Accurate Component-level Energy Modelling of Parallel Applications on Modern Heterogeneous Hybrid Computing Platforms using System-level Measurements

Muhammad Fahad

UCD student number: 15207261

The thesis is submitted to University College Dublin
in fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science

School of Computer Science

Head of School: Assoc. Professor Chris Bleakley

Research Supervisor: Assoc. Professor Alexey Lastovetsky

August 2020

Acknowledgements

All praises and thanks be to Allah the Almighty for everything and more He blessed me with, and make everything always easier for me.

I would like to acknowledge the support, wisdom, and encouragement given by many respected and loving people around me. I would like to start by expressing my deepest gratitude to my supervisor Dr. Alexey Lastovetsky for his invaluable support, mentorship and the immense knowledge he provided me over the years. His continuous guidance and insights always kept me on track and do the right thing even when the road got tough. A very grateful thanks to Dr. Ravi Reddy Manumachu for his assistance, support and encouragement throughout my Ph.D. His insights have immensely improved my writing skills.

A deepest gratitude to Arsalan Shahid for his collaboration and support; thank you Arsalan for always being there whenever it was needed. A big thank you to all my colleagues in UCD's Heterogeneous Computing Laboratory for their fruitful collaborations: Semen Khokhriakov, Hamidreza Khaleghzadeh, Tania Malik, and especially to Emin Nuriyev for all the chats we had over many lunches on almost everything: Culture, History, Science, the emerging technologies, and the list goes on.

The last four years of my life have been truly rewarding for honing my soft skills. I would like to thank all the professors from UCD School of Computer Science and UCD Lochlann Quinn School of Business who provided me the opportunities to be a teaching assistant and demonstrate the world-class modules. To all the staff of the School of Computer Science, especially to my DSP members, Dr. Chris Bleakley and Dr. Brett Becker. Thank you also to the school's support staff: Lorraine McHugh, Léan Ní Chléirigh, Rosemary Deevy, Carl Lusby, D'Arcey Jackson, Paul Martin and Tony O'Gara for their consistent helpfulness over the years.

This research has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474. I would like to thank SFI and University College Dublin for their financial support in the form of scholarship awards.

A special tribute and respect to all my teachers and mentors whose affection, encouragement and guidance have shaped me and made a lasting impact on me. My grateful thanks are also extended to all the amazing people that I got to know during my stay in Dublin, especially to Mr. Waqas Ansari for his moral support and guidance. I would also like to extend my gratitude to Mr. Gerald Griffin for providing me a wonderful home (away from home!).

I am eternally grateful to my loving parents for their unconditional love and providing me with everything possible at their disposal. I would not be the same without your support and encouragement. A very special thanks to my brother and sisters for all the support and unconditional love they have for me. I am forever indebted to you for everything and being always there for me no matter what.

To my wife for her understanding, continuous support and bearing with me throughout these years. You have been there for me every step of the way; your appreciation and love make it all worthwhile. Last but by no means least, I am thankful to our little shining stars Hussain and Abbas, for bringing so much happiness and joy into our life.

To,

the light of the wisdom and the conscience

Abstract

It is predicted that (in a worst-case scenario) Information and Communications Technology (ICT) systems and devices could contribute up to 23% of greenhouse gas emissions, and could share up to 51% of global electricity in 2030. Considering such dire consequences on environment and economy, energy efficiency in ICT is becoming a grand technological challenge and is now a first-class design constraint in all computing settings. Modern high performance computing (HPC) platforms, cloud computing systems, and data centers are highly heterogeneous containing nodes where a multicore CPU is tightly integrated with one or more co-processors/accelerators such as graphical processing units (GPUs), Intel Xeon PHIs, and field-programmable gate arrays (FPGAs) to address the twin critical concerns of performance and energy efficiency. Application component energy profiles are essential to energy optimization of hybrid applications executing in parallel on such heterogeneous computing platforms. Accurate measurement of energy consumption during an application execution is the key to application-level energy minimization techniques. There are three popular approaches to providing it: a). System-level physical measurements using external power meters, b). Measurements using on-chip power sensors, and c). Energy predictive models. While the first approach is known to be accurate, it can only provide the measurements at a computer level and therefore lacks the ability to provide fine-grained device-level decomposition of the energy consumption by an application executing on several independent computing devices (CPU,GPU, etc.) in a computer.

In this thesis, a methodology is proposed to determine the accurate fine-grained device-level energy consumption by an application employing system-level power measurements provided by external power meters. Then, a comprehensive study is presented comparing the accuracy of energy measurements using state-of-the-art on-chip power sensors (provided by RAPL, NVML, Intel MICSMC) and energy predictive models employing performance monitoring counters (PMCs) as predictor variables, against the ground truth (system-level physical measurements using external power meters). An important finding of this study is that the dynamic energy profile patterns of the on-chip sensors and energy predictive models differ significantly from the patterns obtained with the ground truth. This suggests that the measurements using on-chip sensors and energy predictive models do not capture the holistic picture of the dynamic energy consumption during an application execution. Another key finding is that owing to the nature of the deviations of the energy measurements provided by on-chip sensors from the ground truth, calibration cannot improve the accuracy of the on-chip sensors to an extent that can allow them to be used in optimization of applications for dynamic energy. The thesis demonstrates that using inaccurate energy measurements provided by on-chip sensors for dynamic energy optimization of an application can result in significant energy losses (up to 84% in our case). These are important discoveries to keep in mind when basing new research using these sensors and predictive models based on PMCs.

This thesis also addresses two important challenges for energy optimization of hybrid applications running in parallel on modern heterogeneous NUMA computing platforms: a) Accurate modelling of the energy consumption of application kernels when executing in parallel on multiple compute devices, b) Accurate modelling of the energy consumption of different applications executing in parallel on a multi-socket multi-core CPU platform. It proposes a method for accurate estimation of the application component-level energy consumption employing system-level power measurements with power meters. The method is experimentally validated on a cluster of two hybrid heterogeneous computing nodes using three parallel applications matrix-matrix multiplication, 2D fast Fourier transform and gene sequencing. The experiments demonstrate a high estimation accuracy of the proposed method, with the average estimation error ranging between 2% and 5%. The average error demonstrated by the state-of-the-art estimation methods for the same experimental setup ranges from 15% to 75%, while the maximum reaches 178%. It is demonstrated that the use of the state-of-the-art estimation methods instead of the proposed one in the energy optimization loop leads to significant energy losses (up to 45% in our case).

Finally, this thesis highlights and proposes a solution to an understudied yet fundamentally important question of how to measure the goodness of energy profiles. Accurate energy profiles are essential to optimization of parallel applications for energy through workload distribution. Since there are many model-based methods available for efficient construction of energy profiles, we need an approach to measure the goodness of the profiles compared with the ground-truth profile, which is usually built by a time-consuming but reliable method. Correlation coefficient and relative error are two such popular statistical approaches, but they assume that profiles be linear or at least very smooth functions of workload size. This assumption does not hold true in the multicore era. Due to the complex shapes of energy profiles of applications on modern multicore platforms, the statistical methods can often rank inaccurate energy profiles higher than more accurate ones and employing such profiles in the energy optimization loop of an application leads to significant energy losses (up to 54% in our case). In this thesis, we present the first method specifically designed for goodness measurement of energy profiles. First, it analyses the underlying energy consumption trend of each energy profile and removes the profiles that exhibit a trend different from that of the ground truth. Then, it ranks the remaining energy profiles using the Euclidean distances between them and the ground truth. The proposed method is found to be more accurate than the statistical approaches and can save a significant amount of energy. Furthermore, it is shown how the proposed method can help determine if an energy model that provides the inaccurate energy profile has deficient or redundant information.[1]

[1]This research is financially supported by Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

Contents

List of Figures

List of Tables

Statement of Original Authorship

I hereby certify that the submitted work is my own work, was completed while registered as a candidate for the degree stated on the Title Page, and I have not obtained a degree elsewhere on the basis of the research presented in this submitted work.

Chapter 1

Introduction

Multicore architectures are now prevalent in all computing settings ranging from a handheld mobile device to HPC computing platforms and supercomputers. The advent of multicore architectures has boosted the performance of the applications by providing the opportunities of executing them with a higher degree of parallelism. This has opened up a new era for High Performance Computing (HPC).

In today's data-driven era, HPC is considered as a key strategic resource for innovations, economic competitiveness, and major advances of a country's future [3] [4]. It offers an immense computational power to the industry and small and medium enterprises to find the high-value innovative solutions, lower the production cost, streamline the whole production process, minimize the time to design and test the prototypes, and reduce the time to market for products. Furthermore, nearly all scientific disciplines are relying on very high computing power for scientific discoveries. Consider, for example, the Particle Physics experiments at the Large Hadron Collider require a great amount of computing power for simulation, data processing and analysis [5]. Similarly, the scientists needs HPC to accelerate genome sequencing by two orders of magnitude in order to crack cancer diseases [3]. To summarize, HPC is the core of the industrial, societal, and scientific advancements in modern digital world.

For decades, the performance maximization has been the chief concern of both the hardware architects and the software developers. As per Moore's law [6], the number of transistors in a dense integrated circuit doubles every 18 to 24 months. Dennard scaling [7] relates to Moore's law by postulating that the power use of the transistors stays in proportion to their area. While taking the advantage of Moore's law and Dennard's scaling for decades, there has been an exponential improvement in the performance of Central Processing Unit (CPU) by increasing their clock frequencies without any significant additional power consumption. However, this period of "Free lunch" is over [8] now due to the end of Dennard's scaling in around 2006 [9]. This has led the semiconductor industry to transit from single-core to multi-core and many-core architectures. However, the signs of failure of Moore's law indicates an end of this type of performance scaling as well [10].

As a result, the hardware acceleration and the use of co-processors together with CPU are becoming a popular choice to gain the performance boost while keeping the power budget low. This includes both the new customized hardware for particular application domain such as Tensor Processing Unit (TPU), Vision Processing Unit (VPU) and Neural Processing Unit (NPU); and the modifications in existing platforms such as Intel Xeon Phi co-processors, general purpose GPUs and Field Programmable Gate Array (FPGA)s.

Such accelerators together with main processors and memory, constitute a heterogeneous system and are greatly prevalent now in modern ICT devices ranging from handheld mobile devices to HPC systems for many reasons such as growing accelerated computational needs and power constraints. Figure 1.1 illustrates the increasing trend of accelerators/co-processors based supercomputers in TOP500 [1] which has been publishing an overview on the 500 most powerful computer systems twice a year since June 1993. The plot shown in figure 1.1 is based on the data gathered from the lists published in November every year since 2007. As per the list published in November 2019, a total of 29% of the systems are using accelerators/co-processors where no supercomputer on the list was accelerated nearly a decade ago. However, this heterogeneity has raised unprecedented difficulties posed to performance and energy optimization of modern heterogeneous HPC platforms.

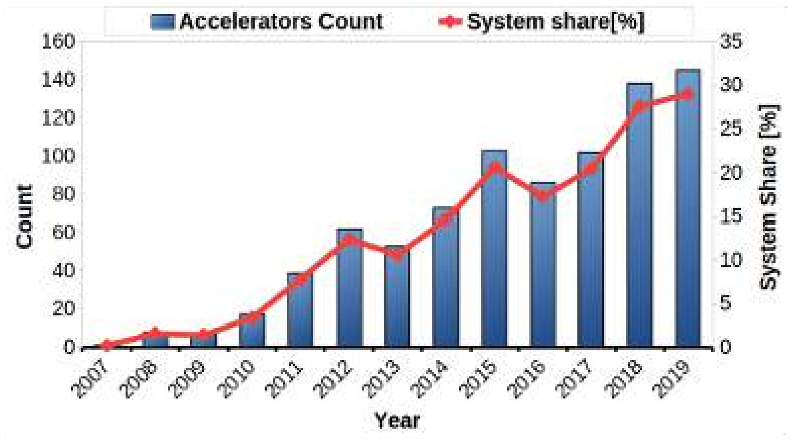
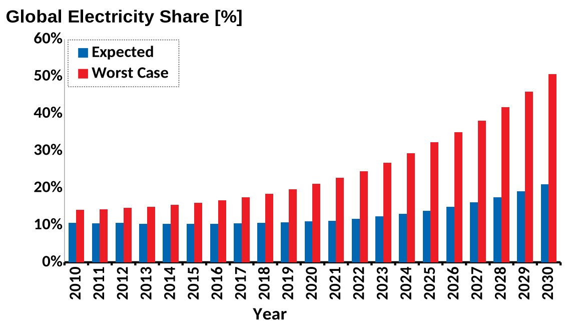


Figure 1.1: System share of accelerators and co-processors in TOP500 supercomputers over time, based on the results published from top500.org [1].

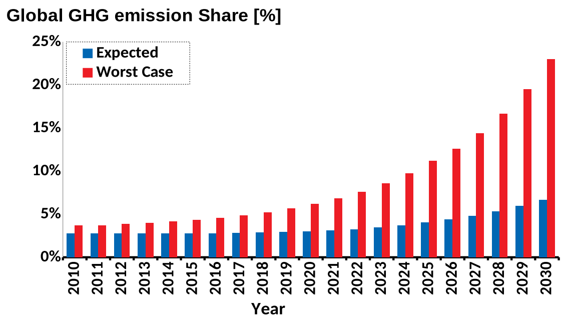
1.1 Motivations for This Thesis

1.1.1 Energy Efficiency: Challenges

This section presents the overview of general challenges to energy efficiency in computing. Energy accounts for two-third of total greenhouse gas [11], and therefore is identified by International Energy Agency (IEA) as a major contributor to climate change. In a recent report, IEA states that energy related CO_2 emission has risen 1.7% to an unprecedentedly high 33.1 gigatonnes due to high energy demands in 2018 [12]. ICT systems and devices (including the personal digital devices, communication networks, televisions, HPC and data centers) are large contributors with a consumption of about 2000 terawatt hours (TWh) per year which is about 10% of the global electricity demand [13]. With a share of 200 TWh, data centers are a big contributor to this high electricity consumption. ICT is generating approximately more than 2% of overall global CO_2 emissions, which is on par with global aviation industry emissions due to fuel combustion [14]. It is predicted that (in a worst case scenario) ICT could use up to 51% of global electricity in 2030, and it could contribute up to 23% of globally released greenhouse gas emission [2]. Figures 1.2(a) and 1.2(b) illustrates the projected share of ICT in global electricity and green house gas (GHG) emissions respectively, for the next ten years.



(a) Percentage share of ICT of global electricity usage.



(b) Percentage share of ICT of global green house gas (GHG) emissions.

Figure 1.2: Projected share of ICT in global electricity and GHG emissions, based on the data from [2].

As a result of Dennard scaling breakdown, energy efficiency is becoming an equally important design concern with performance in ICT. It is becoming a grand technological challenge and is now a first-class design constraint in all computing settings ranging from handheld mobile devices to supercomputers [15, 16]. The advent of multicore architectures has resulted in several inherent complexities such as severe resource contention, NUMA, etc. As a result, the shapes of speed and energy profiles of the applications executing on these platforms are highly non-smooth and non-linear with drastic variations [17]. The studies [17] [18] report that the traditional load-balancing solutions (based on the equal distribution of the workload between identical processors) with such performance and energy profiles do no assure the minimization of execution time or energy consumption. Hence, the traditional methods and algorithms employed for performance and/or energy optimization of the applications executing on modern multicore-based platforms are not applicable anymore [19].

More often than not, modern HPC platforms, cloud computing systems, and data centers are composed of the nodes where a multicore CPU is tightly integrated with one or more compute devices (such as GPUs, FPGAs, Xeon Phis, et cetra) to address the twin critical concerns of performance and energy efficiency. However, this tight integration of multicore CPUs with accelerators has resulted in several inherent complexities, which are: a) Severe resource contention due to the tight integration of tens of cores contending for shared on-chip resources such as Last Level Cache (LLC), interconnect (For example: Intel's Quick Path Interconnect (QPI), AMD's Hyper Transport), and DRAM controllers; b) NUMA where the main memory is distributed between locality domains called NUMA nodes which allows a NUMA node to access its own local memory faster than the non-local memory (memory shared between the NUMA nodes or local to another NUMA node); and c) Dynamic Power Management (DPM) of multiple power domains such as CPU sockets, DRAM.

A parallel application executing on such modern hybrid heterogeneous computing platforms, consists of multiple kernels (generally speaking, multi-threaded), running in parallel on different computing devices of the platform. We term these kernels as application components for illustration purposes in this thesis. As a result of the inherent complexities of heterogeneous hybrid computing platforms, the workload of one application component may significantly impact the performance and energy consumed by the others when running in parallel on different compute devices. This has made the optimization for performance and/or energy optimization of the (hybrid) applications executing parallel on modern multicore-based platforms even more challenging [20][21].

The focus of maximizing the performance of HPC in terms of completing the hundreds of trillion Floating Point Operations Per Second (FLOPS) has led the supercomputers to consume an enormously high amount of energy in terms of electricity and for cooling down purposes. As a consequence, current HPC systems are already consuming Megawatts of energy. For example, the world's most powerful supercomputer as of 2019, Summit, consumes around 13 Megawatts (MW) of power [1] which is roughly equivalent to the power draw of over 10000 households. The power consumption by the top 10 in the TOP500 list has increased over a span of 10 years from about 25 MW in 2008 to around 83 MW in 2018 (as illustrated in figure 1.3) which is an increase of 232%. As per the list of November 2019, the top 4 supercomputers consume a total of over 50 MW of electricity. Because of such high power consumption, future HPC systems are highly likely to be power constrained. For example, US department of energy (DOE) aims to deploy an exascale supercomputer capable of performing 1 million trillion (10^{18}) FLOPS in a power envelope of 20-30 MW [22].

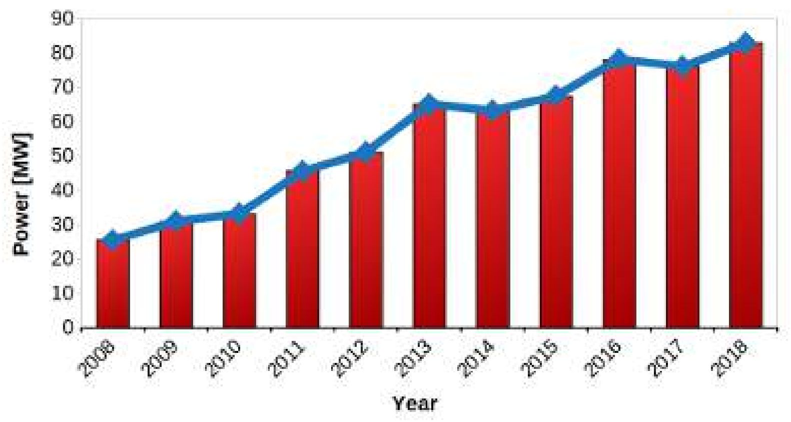


Figure 1.3: Power consumption by top 10 supercomputers over time, based on the results published from top500.org.

GREEN500 [23] is the complementary part of TOP500, which ranks the top 500 supercomputers by energy efficiency since 2007. The disparity between the energy-efficient and the fastest supercomputers can be noted by their corresponding positions in both lists. For example, the top five energy efficient supercomnputers in GREEN500 list published in November 2019 are ranked as {159,420,24,373,1} respectively in its corresponding TOP500 list published in November 2019.

To summarize, the energy of computing is now a serious problem having its dire consequences on environment and economy, and to mitigate it is critically important. Energy efficiency (in transport, industry, and buildings) is the central to the efforts of IEA to combat with climate change [24]. It has become now a first-class design constraint at both hardware and software levels. Energy envelop, carbon footprints, financial implications (electricity bills) and cooling cost bear a serious burden on high performance computing platforms. Energy efficiency is one of the main challenges hindering HPC community from breaking the exascale barrier [25]. In a report published by DOE, the energy efficiency has been listed as a primary constraint for a productive and economically viable exascale system [22].

1.1.2 Energy Efficiency: Approaches in ICT

This section presents a general overview of approaches to achieve energy efficiency in ICT. In general, energy efficiency in ICT can be achieved at following two levels: i) Hardware, and ii) Software. The first approach is driven by the innovations in hardware represented by the micro-architectural and chip-design advancements. The new customized and specialized hardware such as TPU, VPU, NPU, and etc.; and the modifications in existing platforms such as Intel Xeon Phi co-processors, general purpose GPUs and FPGAs are some examples of the innovations at hardware level to achieve the energy efficiency while leveraging the performance. Recent advancements in nanotechnology has also enabled semiconductor fabrication industry to use the unique ways to revolutionize the energy-efficient architectures through innovative transistors and memory technologies. For example, multi-gate transistors are being considered by Complementary Metal-oxide-semiconductor (CMOS) manufacturers to produce smaller microprocessors and memory cells. The multigate metal oxide semiconductor field effect transistor (MOSFET) devices such as gate-all-around field-effect transistors (GAAFET) and fin field-effect transistor (FinFET) reduce the leakage and thus overall power consumption of the incorporating device while enhancing its performance.

The second approach based on software-level can be grouped into two categorized: (a) System-level energy optimization, and (b) Application-level energy optimization. The system-level energy optimization approach focuses on minimizing the energy consumption of the whole node by employing techniques such as clock and power gating, DPM, Dynamic Voltage and Frequency Scaling (DVFS). However, the Application-level optimization methods use application-level parameters and models to maximize the energy efficiency of the applications. The main idea is to capture the real-life performance or energy consumption behavior of the application when executing on a computing platform, and then use such characteristics that influence its performance and energy for optimization purposes. These characteristics include workload size, number of processors, number of threads, loop tile size and etc. are referred as application-level parameters in this thesis for illustration purposes.

In this thesis, we will focus exclusively on the application-level approach. This approach, in general, is comparatively understudied. However, several approaches are proposed recently such as [26, 18, 19, 20, 27, 28, 29, 30], which use the performance and/or energy profiles of an application to optimize its performance and/or energy consumption by employing application-level parameters as decision variables.

1.1.3 Challenges to Application-level Energy Optimization

This section covers the challenges to application-level energy optimization approaches. A clear manifestation of the inherent complexities (introduced by the multi-core architectures) is a complex and non-linear functional relationship between the energy consumption and workload size of the applications executing on these platforms. The shape of speed and energy profiles of the applications executing on modern mutlicore platforms is reported to be highly non-linear and non-smooth with drastic variations [18]. This provides an opportunity for application-level energy optimization through workload distribution as a decision variable [19][20].

We elucidate the energy optimization problem for such applications with an example. Consider an application workload of size n executing on p processors. The problem is to find such a partitioning, $d = x_1, ..., x_q$, of the workload size n between q processors (where $q \leq p$) that minimizes the energy consumption by parallel execution of the workload. A straightforward approach to solve the application-level energy optimization problem using workload size as a decision variable, would be to examine exhaustively all the possible combinations of the workload distributions on q processors and then find the potential workload distribution with minimum dynamic energy consumption during its parallel execution on q processors. The computational complexity of this naïve approach, nevertheless, will be nonlinear.

To reduce this complexity, we need energy profile of the application that is input to a data partitioning algorithm to determine such a workload distribution which minimizes the dynamic energy consumption during its parallel execution. Consider, for example, the model-based data partitioning algorithm [30] to compute an optimal distribution of a given workload size N amongst p heterogeneous processors that minimizes the total dynamic energy when running in parallel. The algorithm takes as input p dynamic energy profiles of the application kernels running parallel on p compute devices. The output is the optimal distribution of the workload amongst the p compute devices. More details on the algorithm and its complexity can be found in [20].

Component level performance and energy models are two key principal building blocks of workload partitioning algorithms employed to partition applications on heterogeneous platforms to optimize the execution time and dynamic energy consumption of the given applications. To accurately measure the execution time of the application components running in parallel, one can use the built-in high precision clocks incorporated within the processors. However, there is no such effective equivalent for measuring the energy consumption by them. As a consequence, while there many model based solutions [18][31][32][33][34][35][36] have been proposed to find such an optimal workload distribution of an application so that its performance can be optimized, a very little progress is seen for model-based energy optimization within a sufficient accuracy.

A parallel application executing on modern hybrid heterogeneous computing platforms, consists of multiple kernels (generally speaking, multi-threaded), running in parallel on different computing devices of the platform. We term these kernels as application components for illustration purposes in this work. As a result of the inherent complexities of heterogeneous hybrid computing platforms, the workload of one application component may significantly impact the performance and energy consumed by the others when running in parallel on different compute devices. Therefore, a fundamental challenge is to determine the decomposition of energy consumption by a hybrid parallel application into application component profiles. The key building block to addressing this challenge is the accurate measurement of energy consumption during the application execution. Accurate energy measurements of an application are critically important also for many other interesting applications such as energy centric performance optimization, auto-tuning of the applications, debugging, to study the trade-off between performance and energy, and identifying the most energy hungry components of the hybrid application. In [37], we find that the use of inaccurate energy measurements to optimize the energy consumption by an application can lead to significant energy losses of up to 84%.

We present a use case, here, to highlight the importance of accurate measurement of dynamic energy during the execution of an application for its optimization for dynamic energy. Consider a real-life dynamic energy consumption profile segment of an application computing 2D fast Fourier transform using FFTW3.3.7 of a complex signal matrix of dimension $N \times N$. The Figure 1.4 shows the dynamic energy profiles of the application for problem sizes (N) ranges from 25984 to 26432 with a step size of 64 constructed with Intel RAPL and HCLWattsUp [38] which provides system-level power measurements using external power meters and which we consider to be the ground truth. The profile is obtained on a modern Intel skylake server comprising of two sockets of 28 cores each. One can observe that Intel RAPL over-reports the dynamic energy consumption for all the problem sizes. Now, consider the workload sizes (or image sizes) N = 26,432 and N = 25,984. The dynamic energy consumption for workload size N = 26,432 is reported by Intel RAPL as 1026 J and 591 J by HCLWattsUp, and for N = 25,984 is 735 J by Intel RAPL and 745 J by HCLWattsUp. If Intel RAPL is used for dynamic energy optimization of an image processing application employing the 2D FFT dynamic energy profile for workload size (or image size) N = 26,432, an optimization method for dynamic energy using Intel RAPL profile as an input could use the solution for the workload size, N = 25,984, aiming to reduce dynamic energy consumption by 28%. Instead, solving this workload size will result in increase of dynamic energy consumption by 26% according to the ground truth.

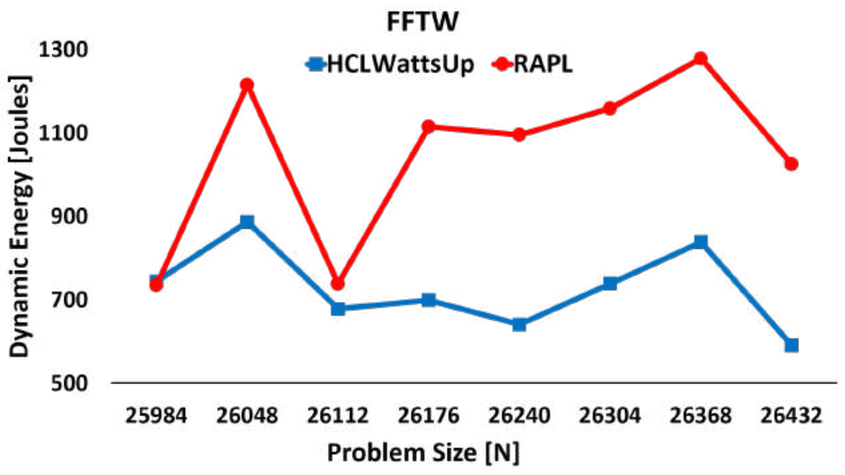


Figure 1.4: Dynamic energy consumption profile segments of HCLWattsUp and Intel RAPL for 2D FFT computation using FFTW-3.3.7 on Intel Intel Xeon Platinum 8180.

In this thesis, we address these challenges by proposing solutions to accurately and reliably measuring the energy consumption by an application during its execution, and to accurately decompose the energy consumption by a hybrid parallel application into application component profiles.

1.1.4 State-of-the-art Approaches to Measure the Energy Consumption of Computing

Energy of computing is now considered as an equally important problem as the performance. Researchers [19, 20, 39] has shown that both of them are tightly coupled and conflicting objectives. Improving performance can yield negative results on energy efficiency and vice versa. Therefore, the accurate measurements of energy are critically important for an insight of trade-off between both objectives and other interesting applications such as energy optimization and etc. The popular approaches to providing the energy consumption by an application during its execution can be broadly categorized into following three categories: a). System-level physical measurements using external power meters, b). Measurements using on-chip power sensors, and c). Energy predictive models.

System-level physical measurements using external power meters: The power measurements by the power meters are considered to be accurate at system level[40]. In this thesis, we consider this approach to be the ground truth.

Measurements using on-chip power sensors: The integrated on-chip power sensors are now prevalent in mainstream processors such as Intel and AMD Multicore CPUs, Intel Xeon Phis and Nvidia GPUs. There are vendor-specific libraries to obtain the power data from these sensors. For example, Intel CPUs offer RAPL [41] to monitor power and to control frequency and voltage. Intel System Management Controller chip (SMC) [42] and NVIDIA Management Library (NVML) [43] provide the power consumption by Intel Xeon Phi and Nvidia GPUs respectively. AMD starting from Bulldozer micro-architecture equip their processors with an estimation of average power over a certain interval through the Application Power Management (APM) [44] capability.

Energy Predictive Models: The third approach is based on software energy predictive models, which emerged as a popular alternative to determine the energy consumption of an application. A vast majority of such models is linear and uses PMCs as predictor variables. PMCs are special-purpose registers provided in modern microprocessors to store the counts of software and hardware activities. In this thesis, we use the acronym PMCs to refer to software events, which are pure kernel-level counters such as *page-faults*, *context-switches*, etc. as well as micro-architectural events originating from the processor and its performance monitoring unit called the hardware events such as *cache-misses*, *branch-instructions*, etc. The most common approach proposing an energy predictive model is to determine the energy consumption of a hardware component based on linear regression of the performance events occurring in the hardware component during an application run. The total energy consumption is then calculated as the sum of these individual energy consumption. Therefore, this approach constructs component-level models of energy consumption and composes them using summation to predict the energy consumption during an application run.

1.1.5 Open Challenges with State-of-the-art Approaches

This section covers the issues and challenges in general, with aforementioned state-of-the-art methods for energy measurements of computing.

System-level physical measurements using external power meters: While the system-level physical measurements provided by the external power meters are considered to be accurate at system level, they lack the ability to provide fine-grained component-level decomposition of the energy consumption of an application. This is a serious drawback. Consider, for example, a computer consisting of a multicore CPU and an accelerator (GPU or Xeon Phi), which is representative of nodes in modern supercomputers. While using the system-level measurements it is easy to determine the total energy consumption of a hybrid application run that utilizes the both compute devices (CPU and the accelerator), it is difficult to determine their individual contributions.

This decomposition is essential to construct the energy profiles that are the key inputs to data partitioning algorithms, and thus are fundamental building blocks for optimization of the application for energy. Without the ability to determine accurate decomposition of the total energy consumption, one has to employ an exhaustive approach (involving huge computational complexity as explained in section 1.1.3) to determine the optimal data partitioning that optimizes the application for energy.

This thesis, however, contributes to fill this gap. The methodologies proposed in this thesis address following three challenges for energy optimization of parallel applications running on modern heterogeneous computing platforms:

1. Accurate measurement of the energy consumption by an application using the system level power measurements provided by the external power meters.

2. Accurate modelling of the energy consumption of application-components when executing a hybrid application in parallel on multiple compute devices on a computer.

3. Accurate modelling of the energy consumption of different applications executing in parallel on a multi-socket multicore CPU platform.

On-chip built-in power sensors: On chip power sensors provide the component level power consumption during the application run which can be accessed using vendor specific libraries such as RAPL, NVML, and etc. However, in general, there is not much documentation available on the accuracy of these vendor-specific libraries. The reported accuracy of the instant current readings in the NVML manual is ($\pm 5\%$) [43]. The accuracy of Intel SMC is not available. For the GPU and Xeon Phi on-chip sensors, there is no information about how a power reading is determined that would allow one to determine its accuracy. However, for the CPU on-chip sensors, RAPL uses voltage regulators (Voltage Regulator (VR) IMON) for CPU and DRAM. VR IMON is an analog circuit within VR, which keeps track of an estimate of the current [45]. It, however, adds some latency because the measured current-sense signal has a delay from the actual current signal to CPU. This latency may affect the accuracy of the readings. The accuracy of VR IMON for different input current ranges is not known. According to [45], DRAM and CPU IMON report higher errors when the system is idle and DRAM VR inaccuracy can be large if the system is allocated memory capacity much lower than its capability. Furthermore, it is reported that VRs from the same manufacturer lot may exhibit different accuracies, and less accurate VRs (for example within an accuracy of $\pm 20\%$) are used by original equipment manufacturer (OEM) for cost-saving purposes [46].

Apart from accuracy, there are some other issues associated with on-chip power sensors. First, how to relate the energy consumption of an application and the energy consumption of the computing elements that are involved in the execution of the application and containing the sensors. While sensors may provide the power consumption of a component within sufficient accuracy, they may not determine the energy consumed by an application when executing on the same component within the same accuracy window. For example, while the accuracy of a power reading is reported by NVML for an Nvidia GPU to be $\pm 5\%$, researchers found that when an application is executed on the GPU, the accuracy is often lower [47] [37]. The topological granularity of power readings are also vitally important to take into consideration while measuring the energy consumption by the application. Sensors only provide the power drawn by a group of computing elements but not the individual contributions of the elements. For example, RAPL reports the overall power consumption by the CPU socket and does not provide the details of the contributions by the individual CPU cores.

The power readings by the sensors also lack details such as update frequency and suffer from potential complications such as sampling interval variability or sensor lag as reported by Reference [47]. Portability is another issue with on-chip sensors due to vendor-specific but non-standardized programmatic interfaces. Existing data center management standards such as IPMI (Intelligent Platform Management Interface) [48] and DCMI (Data Center Manageability Interface) [49]) provide low-resolution data for supported motherboards only. Furthermore, all hardware are not equipped with power sensors and thus the lack of pervasiveness is another important factor limiting their efficacy as a viable approach to determine the energy consumption by an application.

Finally, we discuss the issues with topological granularity of on-chip sensors. Consider, for example, a hybrid matrix-matrix application executing on three compute devices, a multicore CPU and two accelerators (GPU and Xeon Phi). One CPU core acts as a host for each accelerator kernel. Execution of an application using GPU/Xeon Phiinvolves the CPU host-core, DRAM and PCIe to copy the data between CPU host-core and GPU/Intel Xeon Phi. However, the on-chip power sensors (NVML and Manycore Platform Software Stack (MPSS)) provide the power consumption by GPU or Xeon Phi only. One can use RAPL as an aide to NVML/MPSS to determine the energy contribution of CPU and DRAM to obtain the dynamic energy profiles of applications. But, RAPL provides the energy consumption at the socket level which includes also the contribution by other CPU cores executing the CPU kernel and the host-core involved in the execution of other accelerator kernel. Therefore, it is not possible currently with on-chip sensors to accurately attribute the individual contribution of each computation kernel to total energy consumption by a hybrid application executing the kernels in parallel on several heterogeneous compute devices.

To summarize, a good understanding and validation of energy measurement instrumentation systems and on-chip power sensors is necessary for trusting and employing their readings in application-level energy optimization techniques. Furthermore, instead of an instrumentation system or component-level power sensors that measure the instantaneous power drawn by the component, sufficiently accurate measurements of the energy consumed by the application are required for energy optimization and energy-centric performance analysis of applications.

Energy Predictive Models: While the models provide fine-grained component-level energy consumption during the execution of an application, there are research works highlighting their poor accuracy[50, 51, 15, 52, 37]. The sources of this inaccuracy are the following:

1. Model parameters (PMCs) in most cases are not deterministic.

2. In general, the model parameters (PMCs) are selected following the techniques such as principal component analysis or on the basis of their high positive correlation with energy consumption. However, such techniques lack the insights of the physical significance of the model variables originating from fundamental physical laws such as conservation of energy of computing [53].

Economou et al. [50] highlight the fundamental limitation of PMC-based models, which is the restricted access to read PMCs (generally four at a single run of an application). Therefore, it is a tedious task to carefully select the best subset of PMCs as suitable contenders to be used as predictor variables in a model.

McCullough et al. [51] report the predictions errors of such predictive energy models for modern node architectures to be as high as 150%. O'Brien et al. [15] highlight, in their survey on predictive energy models for heterogeneous and hierarchical node architectures, the poor prediction accuracy and ineffectiveness of such models to accurately predict the dynamic energy consumption of modern nodes due to the inherent complexities (such as NUMA, DPM, the contention for shared resources such as LLC, and etc.). Fahad et al. [37] present a comparative study to analyze the accuracy of linear energy predictive models based on PMCs. They report that the average error of the platform-specific energy predictive models ranges from 14% to 32% and the maximum reaches up to 100%. The average error for application-specific energy predictive models reaches up to 27% and the maximum reaches up to 218%.

Shahid et al. [52] question the reliability and reported prediction accuracy of these models, in general. They report that many PMCs that are used as key predictor variables in state-of-the-art predictive models found in literature are not reproducible and does not satisfy the criterion of *additivity* which is derived from the application of theory of energy predictive models for computing (which is explained in [53]). The criterion is based on an experimental observation that dynamic energy consumption of serial execution of two applications is equal to the sum of the dynamic energy consumption of those applications when they are executed separately. The criterion, therefore, is based on a simple and intuitive rule that if the parameter is intended for a linear predictive model, the value of a PMC for a serial execution of two applications is equal to the sum of its values obtained for the individual execution of each application. The PMC is branded as *non-additive* on a platform if there exists an application for which the calculated value differs significantly from the value observed for the application execution on the platform. The authors found that the use of *non-additive* PMCs in a model impairs its prediction accuracy.

Shahid et al. [54] compared the techniques for energy predictive modelling using PMCs on modern multicore CPUs. The authors studied two following types of energy predictive models: i) linear regression models employing PMCs based on the property of additivity, and ii) the sophisticated statistical learning models (random forest and neural network) employing PMCs based on correlation and principal component analysis. The authors conclude that a strong positive correlation of model variables (PMCs) with energy consumption is not sufficient enough to provide good prediction accuracy, and thus it should be combined with methods such as additivity [53] that consider the physical significance of the model variables.

Apart from accuracy, energy predictive models employing PMCs as predictor variables exhibit high implementation complexity due to the following reasons:

1. There is a large number of PMCs provided in a modern multicore processor. For example: Likwid tool [55] provides 164 PMCs and 385 PMCs for the Intel Haswell and the Intel Skylake multicore processors respectively.

2. The accuracy of models is highly dependent on the selection of the PMCs used as predictor variables. However, it is not a trivial task to find the best set of PMCs which reflects the energy consumption for all types of workloads in equally effective way.

3. Tremendous programming effort and time are required to automate and collect all the PMCs on a platform. This is because of the limited number of hardware registers available on plat-

forms for storing the PMCs. Typically, only 3-4 PMCs can be collected in a single run of an application. Moreover, some PMCs can only be collected individually or in sets of two or three for an application run. Therefore, each application must be executed for several times to collect all the PMCs.

4. An energy predictive model purely based on PMCs lacks portability. The reason is because the PMCs available for a CPU processor may not be present in a GPU processor due to inherent architectural differences, or even for the next-generation CPU processor from the architecture space.

These restrictions make highly complex the process of employing PMCs as a predictor variable in models. Hence, it is not a trivial task to build a energy predictive model employing PMCs for all of the heterogeneous compute devices (such as CPUs, GPUs, Intel Xeon Phis, FPGAs, etc.) on thousands of the servers of a data center due to such restrictions.

While the general accuracy of the models has been widely researched, their application-specific accuracy has not been studied and, therefore, needs further validation. In this thesis, a comprehensive study is presented to bridge the gap to compare the accuracy of state-of-the-art on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, which we consider to be the ground truth.

1.1.6 Goodness of Energy Profiles of Applications Executing on Multicore Computing Platforms

Accurate energy profiles or models that are functions of workload size are essential to optimization of parallel applications for energy through workload distribution. However, a fundamental yet understudied challenge is how to measure the goodness of these models. There are many model-based methods for efficient construction of energy profiles but none of them is accurate in all situations. Therefore, to pick the best method in a given situation, we need a way to measure the goodness of energy profiles produced by different methods with the ground-truth profile, which is usually built by a time-consuming but reliable method. We define the goodness as the accuracy of a profile against the ground truth profile. Here, the ground-truth refers to the baseline profile or the reference value for the comparison. Inaccurate energy measurements during an application run can drastically affects the efforts for energy efficient computing.

Pearson correlation coefficient [56] and average prediction error (also known as relative error) [57] are the most commonly used statistical measurements to determine the accuracy of energy profiles. A plethora of research work including [15, 37, 58, 59, 51] use average, maximum and minimum prediction errors to determine the accuracy of energy profiles. References [60, 61, 62, 63] are some of the notable works which used the correlation coefficient to determine whether the energy profiles follow the ground truth.

However, there are research works questioning the effectiveness of both techniques for using goodness measurement of energy profiles. For example, Rico-Gallego et al. [64] argue that the relative error is lower for a profile that underestimates than for a profile that overestimates, and thus can negatively impact the interpretation of the results. Similarly, Fahad et al. [21] demonstrate that the two statistical measures do not capture the holistic picture of the energy consumption trend of the profiles, and thus are blind to the qualitative differences of the energy profiles and the ground truth. As a result, stat-of-the-art but inaccurate energy measurements used in energy optimization of applications can result in significant energy losses [21], up to 84% in some real-life settings [37].

In general, both popular statistical techniques are highly sensitive to outliers and rely on the assumption of linear or smooth increase of energy consumption by applications with the increase of workload size. However, the energy profiles of applications on modern multicore platforms are highly non-smooth and non-linear. Therefore, the existing statistical measures can rank an inaccurate energy profile higher than the accurate ones. The reason is two-fold. First, in the presence of significant variations in the energy profiles, they do not capture the difference in the general trend of energy consumption. Second, they do not capture the similarities in variations.

While the general direction of energy profiles of applications on multicore platforms is reported as a near-linear increasing function of workload size, the shape of the profile can be highly non-linear and non-smooth [18]. We distinguish the terms *trend* and *shape* using the following example. Consider the sample energy profiles shown in figure 1.5. The general direction of all three profiles, which represents the underlying energy consumption trend, is increasing with the increase in workload. However, their shapes are different. The energy profile *Model1* is linear whereas the shapes of *Real* and *Model2* are non-linear and non-smooth.

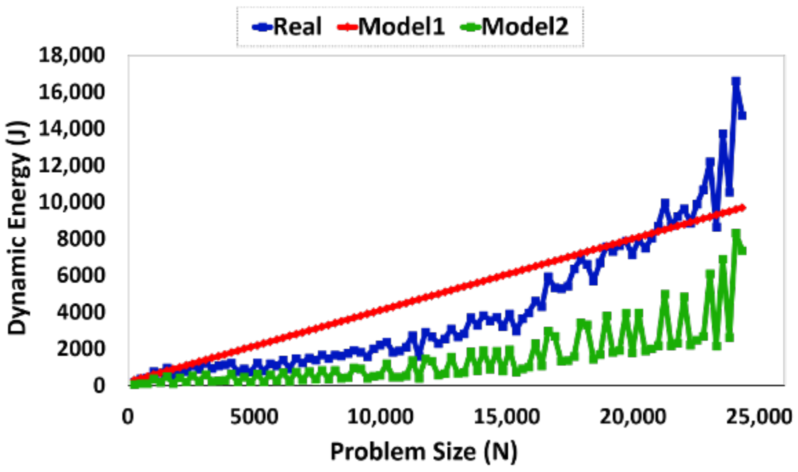


Figure 1.5: Sample Dynamic Energy Profiles

We present a case study to demonstrates that a non-similar energy profile used as an input to an energy optimization algorithm can cause significant energy losses. In figure 1.5, two sample energy profiles are compared against the ground truth (labeled as *Real*). The average errors of profiles *Model1* and *Model2* against the ground truth are 62% and 64% respectively. The Euclidean distance between profiles *Model1*, *Model2*, and the ground truth is 18108 and 33550 respectively. *Model1* and *Model2* are equally strongly correlated with the ground truth with the correlation coefficient equal to 0.91.

While *Model1* is ranked better than *Model2* by both the Euclidean distance and average error, it exhibits different energy consumption behavior for more than 40% of data points as compared with ground truth. Hence, it causes a significant loss of energy when input to the energy optimization algorithm [30], which employs the workload size as the decision variable for energy optimization of an application. For example, *Model1* only provides 21% of workload distributions that are the same as those provided by ground truth when used as an input to this algorithm for energy optimization. In contrast, *Model2* provides the same workload distributions as of the ground truth for 79% of problem sizes despite its higher average error and greater Euclidean distance. Therefore, *Model2* is better than *Model1* for the use in energy optimization or energy consumption analysis of the application.

To summarize, the average error, Euclidean distance, and correlation coefficient are not sufficient to measure the similarity between energy profiles despite being the most used statistical measures for this purpose. The average error and Euclidean distance are highly sensitive to outliers and do not capture the similarity of energy consumption trends. They are also highly sensitive to the transformations such as uniform amplitude/time scaling, shifting, etc. Pearson correlation coefficient, on the other hand, assumes a linear relationship between the variables which might not be always true. It can also be easily misinterpreted as the high correlation coefficient does not necessarily mean a strong linear relationship or high similarity between two profiles. Furthermore, it does not handle a non-linear relationship between the energy profiles. Finally, they can mislead in many cases by erroneously grading an energy profile as the best and causing significant energy losses when used it for the energy optimization of an application.

While the goodness measuring problem is comparatively less-studied for the energy of computing, a plethora of different methods and approaches have been proposed to solve this problem in many other fields such as data mining, time series similarity analysis, and graph (matching) theory. Popular similarity measures for pattern matching are cosine similarity [65], Dynamic Time Warping [66], angular metric for shape similarity (AMSS) for time series data [67], and autoregressive integrated moving average (ARIMA) method [68, 69, 70, 71, 72]. Distance metrics used to determine the pattern matching include Euclidean distance [73, 74, 75] and graph-edit-distance (GED) [76]. In Section 2.5, we provide an overview of the popular approaches in these faculties and why they are not applicable straightforwardly for determining the goodness of energy profiles.

In summary, there is no effective metric to measure the goodness of energy profiles. We present a novel methodology, in this thesis, called **T**rend-based **S**imilarity **M**easure (TSM) of energy profiles, which measures the similarity between a given energy profile and the ground truth. TSM is designed to capture the underlying energy consumption trend of the profiles, and is composed of the following four stages: i). The regression model of the energy profile is learned, ii). The regression fits of this energy profile and the ground truth are compared to determine if they exhibit the same trend, iii). If they do not, then the energy profile is branded fundamentally inaccurate; iv). If they do, the distance between the regression models of the energy profile (that follows the same trend-line as of the ground truth) and the ground truth is determined using Euclidean distance as a metric of goodness of the energy profile. To the best of our knowledge, this is the first work to estimate the goodness of energy profiles by taking into consideration the qualitative difference of the underlying energy consumption trends. Also, unlike other statistical methods used for goodness estimation, it uses the Euclidean distance metric for quantitative estimation of similarity between non-linear and non-smooth profiles, increasing the accuracy of estimation.

1.2 Thesis Contributions

The main contributions by this thesis are followings:

1. A detailed methodology for accurate and reliable measurement of the energy consumption by an application during its execution, using the system-level physical measurements provided by external power meters.

2. The first comprehensive comparative study of the accuracy of state-of-the-art on-chip power sensors against the system-level physical measurements provided by external power meters. The following important discoveries are made:

    - The dynamic energy profile patterns of the on-chip sensors differ significantly from the patterns obtained using the ground truth, which suggests that the measurements using on-chip sensors do not capture the holistic picture of the dynamic energy consumption during an application execution.
    - Owing to the nature of the deviations of the energy measurements provided by on-chip sensors from the ground truth, calibration can not improve the accuracy of the on-chip sensors to an extent that can favour their use in optimization of applications for dynamic energy.

3. An additive energy modelling approach that employs the system-level power measurements to build accurate energy profiles of the individual application-components of a hybrid application executing in parallel on several independently powered compute devices such as CPUs, GPUs, Xeon Phis, and sockets of multi-socket CPUs in a hybrid heterogeneous computing platform.

4. A comparative study of the accuracy of additive energy models of application-components of a hybrid application constructed with state-of-the-art integrated power sensors and the ground truth for two scientific hybrid applications (matrix-matrix multiplication an 2D fast Fourier transform) on a modern hybrid heterogeneous computing platform containing an Intel multicore CPU, an Nvidia GPU, and an Xeon Phi. An important finding is that likewise the application-level energy profiles, the energy consumption behavior of the application-component energy profiles composed with state-of-the-art integrated power sensors also differ significantly from the patterns of the ground truth.

5. Studying the implications of employing in-accurate energy profiles constructed with state-of-the-art approaches, in energy optimization loop of an application. An important finding is that a significant amount of energy is lost by employing the inaccurate energy profiles obtained with the energy measurements provided by state-of-the art approaches, for dynamic energy optimization of an application.

6. A novel methodology to measure the similarity between an energy profile and the ground truth. To the best of our knowledge, the proposed methodology is the first work that takes into consideration the qualitative differences of the energy consumption trend of the profiles and ranks the energy profiles based on their similarity with the ground truth.

    - A comprehensive comparative analysis of the proposed methodology with popular statistical approaches such as correlation, average error, and Euclidean distance, which are commonly used to compare the accuracy and similarity of energy profiles as well as time series of equal lengths in general. An important finding is that all three statistical approaches fail to capture the qualitative difference of an energy profile and the ground truth, and thus fail to distinguish the energy profiles based on their energy consumption trend. As a consequence, they can mislead to rank an inaccurate energy model as better than more accurate ones.

- We demonstrate how the proposed methodology can help in determining whether the energy model that is used to construct the energy profile, includes some extraneous contributors that do not reflect the energy consumption by the application, or it lacks some essential contributor to the energy consumption by the application.

1.3 Thesis Structure

The structure of this thesis is as follows. In Chapter 2, we discuss the existing techniques to measure the energy of computing, power saving mechanisms, notable energy optimization approaches, and common practices and notable works to measure the goodness of the energy profiles. In Chapter 3, a comprehensive methodology is presented to accurately and reliably determine the energy consumption by an application using system-level power measurements. The comprehensive study comparing the accuracy of state-of-the-art integrated on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, is presented in Chapter 4. In Chapter 5, we present a novel methodology called Additive Energy Modelling of Hybrid Applications *AnMoHA* to decompose the energy consumption by a hybrid application at component-level using the system-level power measurements. In Chapter 6, a novel methodology is presented to determine the goodness of energy profiles of an application built with different tools (integrated power sensors, PMC based energy predictive models, system-level power measurements). Finally, we conclude the thesis in chapter 7.

Chapter 2

Background and Related Work

This chapter is organized as follows. First, we introduce some basic taxonomy in section 2.1, related to HPC, power and energy, and terminologies that are used in this thesis. Afterwards, the state-of-the-art techniques to measure the energy of computing are presented in section 2.2. The existing power saving mechanisms in HPC, and some notable energy optimization approaches are presented in section 2.4.1, and 2.4 respectively. Finally, the common practices and notable works to measure the accuracy of the energy profiles is presented in section 2.5.

The most of the chapter is based on the following papers that I authored or co-authored. In [37], we presented a comprehensive study comparing the accuracy of state-of-the-art on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, which we consider to be the ground truth. In [21], we propose a novel method for accurate estimation of the application component-level energy consumption employing system-level power measurements with power meters. This additive energy modelling methodology is used for the optimization of data-parallel applications on heterogeneous HPC platforms for dynamic energy through workload distribution published in [39] and [30]. The same additive energy modelling methodology also laid the basis of the study of bi-objective optimization for performance and energy on heterogeneous processors published in [20]. In [17], we elucidated the challenges to energy and performance optimization of parallel application introduced by the advent of multi-core architectures. In [77], we presented the first method specifically designed for goodness measurement of energy profiles. In [52], we presented a novel selection criterion for PMCs called additivity, which can be used to determine the subset of PMCs that can potentially be used for reliable energy predictive modeling. In this work, we study the additivity of PMCs provided by mainstream performance monitoring tools. In [53], we presented the theory of energy of computing and its practical implications. In [78], we study how that the accuracy of state-of-the-art energy predictive models can be improved by selecting performance monitoring counters based on a property of additivity. In [54], we compare two types of energy predictive models (linear regression and sophisticated statistical learning models (random forest and neural network)) employing PMCs selected by different criterion such as additivity, correlation coefficient and principal component analysis. In [169], we study the prediction accuracy of linear models employing utilization variables only, PMCs only, and combination of both (the utilization variables and PMCs) on modern multicore CPU platforms.

2.1 Terminologies and Taxonomy

In this section, first, we explain the terminologies that are used in this thesis. Then, we provide a general overview of HPC architectures, and energy and power within the context of computing.

Modern HPC computing systems today feature tight integration of multicore CPU processors and accelerators (mix of GPUs, Xeon Phis, FPGAs, et cetra) empowering them to provide not just unprecedented computational power but also to address the newly established critical concerns of power and energy efficiency. We term such heterogeneous nodes incorporating a mix of multicore CPUs and accelerators or co-processors as *hybrid heterogeneous computing platforms*. We broadly refer the constituent processing units (including the multicore-CPU, accelerators, and the co-processors) of such hybrid node as the *compute devices*. A parallel *hybrid application* executing on such a hybrid node, consists of multiple kernels (generally speaking, multi-threaded), running in parallel on different computing devices of the platform. We term these kernels as *application components*.

2.1.1 HPC Architectures

Michael J. Flynn proposed an enhanced version [79] of his initially proposed methodology [80] to classify the computer architectures based on the number of concurrent operations they provide for handling the instructions and data streams. Based on Flynn's taxanomy, the computer architectures can be classified into following four categories:

1. **Single Instruction Single Data (SISD):** It represents the computing systems which can handle only one instruction and one data at a time, and therefore do not exhibit any form of parallelism. Examples include single core and uniprocessor systems (which has a single CPU to execute the computational tasks).

2. **Single Instructions Multiple Data (SIMD):** The architectures which can operate single instruction on multiple data streams fall into this category. However, such architectures do not exhibit concurrency because they execute instructions or processes sequentially but they operate each instruction on multiple data in parallel. Multimedia applications are typical examples to take advantage of such architectures where a common operation (such as adjusting the color) is needed to be applied to a large number of data points such as pixels. Examples include Intel's latest Advanced Vector Extensions and the GPUs.

3. **Multiple Instructions Single Data (MISD):** It represents the architectures which can operate multiple instructions on a single data stream in parallel. This type of architectures are considered as uncommon in general. However, systolic Arrays [81] are often considered as a classic example of MISD.

4. **Multiple Instructions Multiple Data (MIMD):** It represents the architectures where different instructions are operated on different data in parallel. Multi-core/ multiprocessor systems are the typical examples of such architectures.

Modern computing systems in general and HPC systems are in particular represent the MIMD architectures. All of the current TOP500 [1] supercomputers are based on MIMD architecture. Base on memory organizations, MIMD systems can further be classified into following two categories:

1. Distributed Memory: It represents the systems where multiple computing nodes are connected with each other using some shared interconnect. Each node in the network has its own private memory, processors, operating systems and peripherals. Local data (held in the memory) of a node can not be accessed by other nodes directly. However, the nodes can share their data with other nodes using message passing interfaces such as Messgae Passing Interface (MPI).

2. Shared Memory: It represents the systems where a number of processors are connected to a large pool of memory called main memory. It offers a single address space to all processors.

Shared memory based systems can further be classified into following three categories based on the memory organisation and the access time to it.

1. Uniform Memory Access (UMA): The memory is evenly shared among all the processors. All the processors can access the memory uniformly, and the latency is the same for all processors to access a memory word. This type is mainly suitable for time sharing and general purpose applications.

2. Non-Uniform Memory Access (NUMA): Unlike UMA, the main memory is distributed between locality domains called NUMA nodes. It allows a NUMA node to access its own local memory faster than the non-local memory which is shared between the NUMA nodes or local to another NUMA node.

3. Cache-Only Memory Architecture (COMA): Unlike NUMA, the whole shared memory is used as cache where each memory module acts as a huge cache memory and each block has a tag with the address and the state.

The hardware acceleration and the use of co-processors together with CPU are becoming a popular choice to gain the performance boost while keeping the power budget low. Hence, contemporary computational clusters, data centres and supercomputers are getting highly heterogeneous such that 102 out of top 500 supercomputers are heterogeneous [1]. As per the list published in November 2019, a total of 29% of the systems are using accelerators/co-processors where no supercomputer on the list was accelerated nearly a decade ago. The hardware accelerators can be classified into many forms such as

1. Fully customized application specific integrated circuits (ASIC): This includes the customarily designed chips for a particular application domain. For example, artificial intelligence (AI) accelerators are particularly designed to boost the artificial intelligence applications such as VPUs for machine vision, TPUs for machine learning, and NPUs for artificial neural networks.

2. The modifications in existing platforms: This includes modifications in existing hardware in order to decrease latency and increase throughput to support more general applications than ASICs. For example, GPUs were originally used to accelerate the graphical computations. However, they have extensively been evolved to support many other scientific and engineering compute intensive applications.

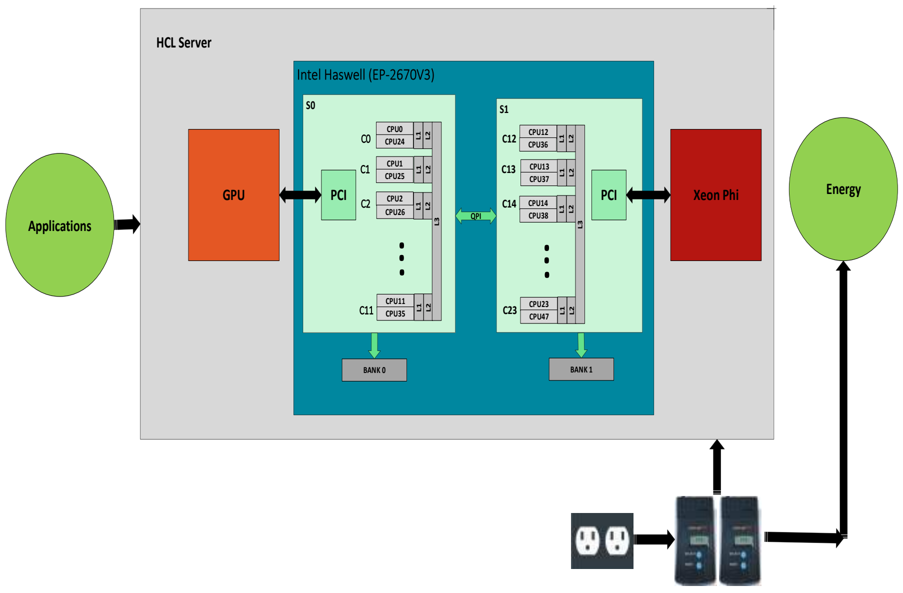


Figure 2.1: Block diagram of a hybrid heterogeneous NUMA node.

3. Re-configurable hardware accelerators: These devices can be tailored dynamically to suit the needs of the specific application. They represent a blend of customization (to gain the performance and energy efficiency) and generality (and thus programmability). FPGAs are the typical example of this class.

While modern CPUs represent both SIMD and MIMD (using vector instructions and mutlicores respectively), the accelerators such as GPUs represent SIMD and Xeon Phi represents MIMD architectures respectively. Currently, GPUs can only be used as an accelerator where the workload is offloaded to it by the CPU, however Xeon Phi can either be used as a co-processor or an accelerator. Figure 2.1 illustrates the block diagram of a typical hybrid heterogeneous NUMA node incorporating one multicore CPU and one or more accelerators.

In following sections, a brief overview of relationship between energy, time and power is presented within the context of computing.

2.1.2 Energy and Power

Energy (*E*) can be defined as the total amount of work performed by the system over a time period (*T*), whereas power (*P*) is the rate at which the work is done by the system. Energy is measured in Joules; power in Watts and time period in seconds. The relationship can be expressed as:

$$E = P \times T \tag{2.1}$$

Energy is also considered as the integration of power values for a time period T, starting from $t1$ till $t2$. Hence,

$$E = \int_{t1}^{t2} P \, dT \tag{2.2}$$

2.1.3 Instantanious Power vs Average Power

Instantaneous power is the amount of power in a circuit at any instant of time, whereas the *average power* consumption is the average of the instantaneous power over one complete cycle. Because of its variations in magnitude and sign over a cycle, instantaneous power is considered as less important. However, average power remains exactly the same for over all the time as the average power given over one cycle. Furthermore, instantaneous power is hard to work with due to its measurement limitations and highly dynamic nature. Therefore, average power is generally used to determine the energy using the numerical approximation of the equation 2.2. In this thesis, power indicates the average power over a given time period, unless specified otherwise.

2.1.4 Static vs Dynamic Power and Energy

CMOS is the main technology used to construct the integrated circuits such as microprocessors, memory chips and other digital logic circuits, because of its high noise immunity and low static power consumption. *Total Power consumption* (P_{total}) in a CMOS component is attributed to the total current that flows within the component for/by different physical processes such as sate changes, clock signaling, etc. Total power consumption in CMOS can be classified into two main types: static power and dynamic power, as shown in equation 2.3.

$$P_{total} = P_{static} + P_{dynamic} \tag{2.3}$$

Static power P_{static} is the idle power or base power, and is consumed because of the non-ideal behavior of transistors. It is also called the leakage power because it is dissipated due to the current leakage through transistors. Subthreshold conduction, gate leakage and reverse biased junction band-to-band tunneling (BTBT) are the dominant sources of the current leakage for sub-100nm technology [82][83][84]. Static power is temperature dependent [85] and has no relation to the clock frequency. It is the product of supply voltage and the device leakage current. Total static power can be obtained using the following equation [86]:

$$P_{static} = I_{CC} \times V_{CC} \tag{2.4}$$

Here, I_{CC} is the leakage current and $V_{CC}$is the supply voltage.

Dynamic Power $P_{dynamic}$ is considered as the primary source of total power dissipation of CMOS circuits. It is consumed due to the switching activities of the transistors. It can further be classified into following two parts:

1. Transient or Switching power: The power used to charge/discharge the capacitive load that drives the logic gate within the circuitry. Switching is quadratically dependent on the supply voltage, and linearly dependent on the frequency of switching activities. It can be expressed as:

$$P_{switching} = \alpha \frac{1}{2} V^2 C f \tag{2.5}$$

Here, V is the supply voltage, C is the electrical capacitance, f is the clock frequency, and α represents the switching or activity factor [87] where $0 \leq \alpha \leq 1$. The variable α is added because most gates do not change their state every clock cycle.

2. Short-circuit power: It is caused by the shorting of the supply to ground when both transistors in a CMOS gate are conducted simultaneously momentarily during the state changes of the logic gates.

Hence, the equation 2.3 can be expressed as:

$$P_{total} = P_{static} + P_{switching} + P_{shortcircuit} \tag{2.6}$$

Switching power is considered as the most significant contributor to the total power dissipation by a well designed CMOS . From the component point of view, we define the dynamic and static power consumption of the CMOS component as the power consumption of the component with and without the given application utilizing the component during its execution respectively. From an application point of view, we define dynamic and static power consumption as the power consumption of the whole system with and without the given application execution respectively.

Similarly, we classify energy into two types: *dynamic energy* and *static energy*. From the platform point of view, we define dynamic and static energy consumption as the energy consumption of the whole system with and without the given application execution respectively. Here, the static energy is the base energy of the system which is an intrinsic property of the system, as it consumes this base energy all the times regardless of doing any work or performing any computation. However, the dynamic energy is the largest component of the total energy consumption by the application when running on a system. That is why, it is the main target of application level energy optimization techniques such as [18, 19, 20, 88, 30].

2.2 Common Practices to Measure the Energy of Computing

Energy of computing is determined as the product of power and time, as shown in equation 2.1. While the time can be measured accurately using high precision CPU clocks, there is no equivalently effective way to measure the power consumption by an application. In this section, we present common techniques and approaches to measure the energy of computing.

2.2.1 Power Instrumentation Systems

Fine-grained power instrumentation systems provide the accurate insights of power consumption by HPC systems and applications. Sophisticated custom-designed Power Monitoring Infrastructures (PMI) are used for power/energy profiling of HPC applications and multiple system components.

PowerPack [89] is the one of the first instrumentation frameworks for direct and automatic profiling of power consumption by parallel HPC applications. It is a combination of hardware (such as sensors, external power meters, data acquisition devices which allow direct instrumentation and direct power measurements) and software (such as drivers for sensors and meters, benchmarks, instrumentation APIs, analysis tools, etc.). It was extended to support multicore and multiprocessor systems [90]. Sense resistors are tapped into each DC power rail, and digital meters are used to measure the voltage difference of two ends of the resistors. Powerpack measures the power consumption of components such as motherboard, CPU, disk, memory, CPU fans and system fans using direct or derived measurements. However, it directly measures the power and energy consumption of one node at a time. The authors [90] suggest a remapping approach to obtain the total power consumption of a whole cluster, where the direct measurement of power consumption can be applied to other identical nodes in the cluster.

Bedard et al [91] [92] present two low-cost power monitoring devices: PowerMon and PowerMon2 to monitor the voltage and current to six and eight DC rails to components respectively, by inserting them in between the power supply and motherboard. The sampling rate of PowerMon is 50 samples per second whereas PowerMon2 can provide up to 1024 samples per second on one channel, or 3072 samples per second when divided among multiple channels. The devices employ an Analog Device ADM1191 [93] which is connected to a sense/shunt resistor on each power rail for detecting the voltage and current traversing the resistor.

Laros et al [94] present PowerInsight which is composed of following three major components: 1) a BeagleBone [95], 2) a carrier board which provides the connection for 15 sensor modules, and 3) a harness which contains the sensor modules to be integrated between the motherboard and the power supply. Hence, it can support the instrumentation of up to 15 DC rails. Unlike PowerMon line of devices, it uses Hall effect sensor due to its low impact on the power rail being measured.

Hackenberg, et al [96] present High Definition Energy Efficiency Monitoring (HDEEM) infrastructure that facilitates energy-aware optimization of parallel codes. HDEEM is implemented on a similar approach to PowerInsight with few different hardware design decisions. For example, HDEEM combines an FPGA and existing baseboard management controller (BMC) instead of using an autonomous measurement board. This reduces the hardware cost and the overall complexity for the measurement equipment. HDEMM provides a higher temporal resolution (the sampling rate) of 8k samples per second, and the finer spatial granularity at the level of per CPU. The authors claim a higher quality of results due to the noise filtering and sensor calibration.

IBM server blades JS22 and HS211-XM are equipped with on-board power-measurement circuits [97]. Low impedance resistors in series are placed with a power rail using circuits feeding the voltage regulators modules powering the system. The power data can be sampled using an IBM proprietary tool Amester [97]. The sampling rate of Amester is up to 1k per second.

To summarize, PMI are custom-designed integrated systems to provide the fine-grained component-level energy consumption of the System-Under-Test (SUT). However, apart from issues related to temporal resolution (sampling rate), and spatial/topological granularity (for example, whether it can measure the energy consumption by the cores individually or it reports the aggregated energy consumption by all the cores at socket level), PMIs suffer from another important disadvantage, which is to manually instrument the hardware that is a highly specialized skill for computer scientists. Mostly PMIs employ sense/shunt resistors and hall effect sensors to determine the power draw in the component/system. Following are some of the drawbacks of using sense/shunt resistors:

- The voltage drop is a big concern for low voltage and high current applications.
- A direct electrical connection is required between the power supply and the component which requires the expertise. Any mishandling may cause damage to hardware.
- They increase the overall power draw by introducing the resistance into the circuit.

- Their resistance is temperature dependent which, therefore, affects the accuracy of power data.

The disadvantages of using hall effect sensors include:

- External magnetic fields can interfere and thus bias the measurements. Therefore, the sensor position is very important to consider within the system.
- The accuracy is highly influenced by the temperature.
- The occurrence of offset voltage.

2.2.2 Power meters

System level power consumption details can be obtained using physical measurements employing external AC power meters. Dedicated power meters are installed between the input power sockets to the system and the wall AC outlets. The total energy consumption by the system using the external power meters is considered to be highly accurate [40] [60]. The revenue-graded power meters such as Yokogawa WT5000 [98] offer the basic power accuracy of up to $\pm 0.03\%$ with a very high sampling rate of 10M per second. Other examples include Watts Up Pro [99] and ZES Zimmer power analyzers [100]. While the accuracy of different models of Watts Up Pro is reported as $\pm 1.5\%$ and $\pm 3\%$ with a sampling rate of 1 sample per second, ZES Zimmer's products such as LMG671 offers an accuracy of $\pm 0.025\%$ with a gap-less sampling up to 18 bit and a minimal cycle time of 10 ms.

However, power meters can provide the measurements only at a system level and cannot, therefore, provide the fine-grained decomposition of the energy consumption by an application executing on multiple independent computing devices in a system. This is a serious drawback.

2.2.3 On-chip power sensors and vendor specific libraries

On-chip integrated power sensors are now prevalent in mainstream processors such as Intel and AMD Multicore CPUs, Nvidia GPUs, and Xeon Phis.

Intel CPUs offer Running Average Power Limit (RAPL) [41] to monitor power and control frequency (and voltage). RAPL is based on a software model using performance monitoring counters (PMCs) as predictor variables to measure the energy consumption for CPUs and DRAM for processor generations preceding Haswell such as Sandybridge and Ivybridge E5 [101]. For latest generation processors such as Haswell and Skylake, however, RAPL uses separate voltage regulators (VR IMON) for CPU and DRAM.VR IMON is an analog circuit within the voltage regulator (VR), which keeps track of an estimate of the current. It, however, adds some latency because the measured current-sense signal has a delay from the actual current signal to CPU. This latency may affect the accuracy of the readings. RAPL samples this reading periodically (100μs to 1 ms) for calculating the power [45]. The accuracy of VR IMON for different input current ranges is not known. According to Reference [45], DRAM and CPU IMON report higher errors when the system is idle or if the system is allocated memory capacity much lower than its capability.

Furthermore, it is reported that VRs from the same manufacturer lot may exhibit different accuracies [46]. Less accurate VRs (for example within an accuracy of $\pm 20\%$) are used by original equipment manufacturer (OEM) for cost-saving purposes [46]. However, to compensate the reported inaccuracies of output current (IMON) from a VR to a processor, an approach is recently proposed to use a programmable load line from BIOS instead of actual implemented load line. This programmed load line values adds an offset to the determined inaccuracy of the VR to increase its accuracy. [46].

Hackenberg et al. [60] report systematic errors in RAPL energy counters and find that it is inclined towards certain types of workload and can give poor power predictions for others. However, in another study later [62], they demonstrate that RAPL improves the accuracy of energy measurements for Haswell generation processors due to employment of VR IMON for power measurement [45]. Khan et al. [63] study the RAPL accuracy on Haswell generation processors by running different benchmarks. While Hackenberg et al. [62] run micro-benchmarks using different thread configurations, Khan et al. run benchmarks using different frequency configurations. However, both of them compare the RAPL readings with total system (AC) power consumption using power meters and report that the RAPL readings are in a strong correlation with AC measurements.

AMD starting from Bulldozer micro-architecture equip their processors with an estimation of average power over a certain interval through the Application Power Management (APM) [44] capability. Reference [60] reports that APM provides highly inaccurate data particularly during the processor sleep states.

Xeon Phi co-processors are equipped with on-board Intel System Management Controller chip (SMC) [42] providing energy consumption that can be programmatically obtained using Intel many-core platform software stack (MPSS) [102]. The accuracy of MPSS is not available.

NVML [43] provides programmatic interfaces to obtain the energy consumption of an Nvidia GPU from its on-chip power sensors. The reported accuracy of the instant current readings in the NVML manual is $\pm 5\%$. However, experimental results demonstrate the accuracy as worse [37]. Burtsher et al. [47] examine the power profiles of three different Nvidia GPUs (Tesla K20c, K20m and K20x) when executing an N-body simulation benchmark. They find multiple anomalies when using the on-chip sensors on K20 GPUs, and inaccurate power readings on K20c and K20m that lag behind the expected power profile based on a software model (which they believe to be the ground truth). Furthermore, the authors observe that the power sampling frequency on K20 GPUs varies greatly and the GPU sensor do not update the power readings regularly.

Instantaneous power usage data on the motherboards can be acquired through Intelligent Platform Management Interface (IPMI) [48] using Baseboard Management Controller (BMC) monitoring chip on supported motherboards. There are a number of vendor specific extensions such as implemented by the Intel Node Manager technologies or Dell Remote Access Controller to manage and monitor the server power.

2.2.4 Energy Predictive Models

Software based energy predictive models emerged as a predominant alternate approach to predict the energy consumption by an application. A vast majority of such models is linear and uses PMCs as predictor variables to predict the energy consumption. PMCs are special-purpose registers provided in modern microprocessors to store the counts of software and hardware activities. In this thesis, PMCs refer to collectively i) software events which are pure kernel-level counters such as *page-faults*, *context-switches*, etc., and ii) micro-architectural events originating from the processor and its performance monitoring unit called the hardware events such as *cache-misses*, *branch-instructions*, etc.

The most common approach proposing an energy predictive model is to determine the energy consumption of a hardware component based on linear regression of the performance events occurring in the hardware component during an application run. The total energy consumption is then calculated as the sum of these individual energy consumption. Therefore, this approach constructs component-level models of energy consumption and composes them using summation to predict the energy consumption during an application run.

Now, we present an overview of the tools widely used to obtain PMCs, notable energy predictive models, and critical reviews of PMCs.

Tools to obtain PMCs: Perf [103] can be used to gather software events such as *context-switches*, *minor-faults*, etc., and hardware events such as *instructions retired*, *L1 cache misses*, etc., for Linux-based systems.

PAPI [104] is a well-known portable API for reading PMCs found in majority of the microprocessors. The latest version of PAPI (PAPI6.0) also includes the support for monitoring the power consumption by AMD GPUs, Power9 and succeeding generations of IBM PowerPC architectures.

The latest fork of Intel PCM [105] provides a set of tools to monitor the energy and performance metrics of a range of Intel processors including Intel Core, Xeon, Atom and Xeon Phi processors. It supports PCM Linux, Windows, Mac OS X, FreeBSD and DragonFlyBSD operating systems.

Likwid [55] is a lightweight command line tool which can be used to obtain PMCs for for a range of Intel, AMD, ARMv8 and POWER9 processors on the Linux operating system. It provides the energy information by accessing RAPL counters.

For Nvidia GPUs, CUDA Profiling Tools Interface (*CUPTI*) [106] provides a set of APIs and tools to profile and trace the CUDA applications. These APIs can be used to collect the PMCs. For example, CUPTI Event API allows to read the event counters on a CUDA-enabled device, and CUPTI Metric API allows to collect the application metrics such as cf_executed (number of executed control-flow instructions), dram_read_transactions (device memory read transactions), and etc.

The main techniques used to select PMCs for modeling can be divided into following four categories:

- Techniques that consider all the PMCs offered for a computing platform with the goal to capture all possible contributors to energy consumption. To the best of our knowledge, we found no research works that adopt this approach because of the models' complexities.

- Techniques that use a statistical methodology such as correlation, principal component analysis and so forth to choose a suitable subset [107, 59].

- Techniques that use expert advice or intuition to pick a subset of PMCs and that, in experts' opinion, are dominant contributors to energy consumption [108].

- Techniques that select parameters with physical significance based on fundamental laws such as the energy conservation of computing [52]. Shahid et al. [52] introduced a new property that is based on an experimental observation that dynamic energy consumption of serial execution of two applications is equal to the sum of the dynamic energy consumption of those applications when they are run separately. The property is based on a simple and intuitive rule that if the PMC is intended for a linear predictive model, the value of it for a serial execution of two applications should be equal to the sum of its values obtained for the individual execution of each application. The PMC is branded as *non-additive* on a platform if there exists an application for which the calculated value differs significantly from the value observed for the application execution on the platform. The authors report that the use of *non-additive* PMCs in a model impairs its prediction accuracy.

To facilitate clarity of exposition, the mathematical form of the linear regression models can be stated as follows:

$\forall a = (a_k)_{k=1}^{n}, a_k \in \mathbb{R},$

$$f_E(a) = \beta_0 + \beta \times a = \sum_{k=1}^{n} \beta_k \times a_k \tag{2.7}$$

where β_0 is the intercept and $\beta = \{\beta_1, ..., \beta_n\}$ is the vector of coefficients (or the regression coefficients). In real life, there usually is stochastic noise (measurement errors). Therefore, the measured energy is typically expressed as

$$\tilde{f}_E(a) = f_E(a) + \epsilon \tag{2.8}$$

where the error term or noise ϵ is a Gaussian random variable with expectation zero and variance σ^2, written $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Additive energy models for the entire system: One of the simplest additive models was presented by Roy et al. [109] which determine the energy consumption by CPU and memory when running an algorithm. The model determine the energy consumption by CPU and memory during the execution of an algorithm as a weighted sum of time complexity and the number of parallel I/O operations of the algorithm.

Lewis et al. [110] proposed a system-wide energy model which relates the energy consumption by the server to its overall thermal envelope. They presented a linear regression model using PMCs as predictor variables within an error of up to 4%. In this model, the system-wide energy consumption is presented as a summation of power models of processor, memory (DRAM), fans, motherboard (chipset) peripherals, and hard-disk drive.

Basmadjian et al. [111] presented a similar model which include some additional components (such as network interface card and power supply unit) to the equation for constructing a similar aggregated power model of the server as a function of resource utilization by its sub-components. They report the error rate of their model ranging between 2% and 10%.

Bircher et al. [112] proposed an iterative procedure to predict the power of the entire system using PMCs that trickle down from the processor to other subsystems such as disks, CPU, memory, I/O and chipset. They estimated the power of entire system as a summation of six subsystems CPU, memory, chipset, I/O, disk, and GPU, and validated their model on two platforms (server and desktop). The average error of their model is reported as 9% per subsystem.

Heath[113] propose a linear model using the utilization of CPU, disk, and the network to estimate the total power consumption of the entire system. The average and maximum error of their proposed model is reported as 1.3% and 2.7% respectively. Economou et al. [50] proposed a more complex power model called as Mantis which is a non-intrusive method and requires a one-time model fitting.

It employs the utilization metrics of CPU, disk, and network components and hardware performance counters for memory as predictor variables.

Rivoire et al. [114] study and compare five full-system real-time power models using a variety of machines and benchmarks. One of them includes CPU PMCs and the OS-reported utilization of CPU and disk in the model variable set whereas the remaining four models are utilization-based. They report that the PMC-based model is the best overall in terms of accuracy because it accounts for the majority of the contributors (particularly the memory activity) to the system's dynamic power.

Energy predictive models for CPU: One of the first models correlating PMCs to energy values was developed by Bellosa et al. [115]. Their model is based on events such as memory requests due to cache misses, integer operations, floating-point operations, etc., which they reported as strongly correlated with energy consumption. An elaborated methodology employing PMCs is proposed by Icsi et al. [116] to determine the component-level power which is estimated from the access rates of the components.

References [113], [50], present component (CPU, fans, memory, and hard disk drive) wide energy predictive models based on highly correlated performance events such as cache misses, floating-point operations and integer operations. Dargie et al. [58] quantify the relationship between the workload and power consumption of the multicore processor by using the statistics of CPU utilization. Lastovetsky et al. [18] propose an application-level energy model by modelling the dynamic energy consumption of a multicore CPU as a highly non-linear function of problem size.

Lee et al. [117] propose a statistically rigorous approach to derive regression models using PMCs to predict power. The median and maximum error of their method is 4.3% and 24.5% respectively. Li et al. [118] report a strong correlation between instructions per cycle (IPC) and operating system (OS) routine power, and thus propose power models for the OS. They reported the estimation error of their power model as less than 6%.

Fan et al. [119] propose a simple linear model that correlates the power consumption of a single-core processor with its utilization. Singh et al. [120] develop per-core power models based on multiple linear regression using PMCs. Powell et al. [121] use a linear regression model to estimate activity factors and power for a large number of micro-architectural structures using a small number of PMCs. Goel et al. [122] derive per-core power models using PMC values and temperature readings. A linear model that takes into account CPU utilization and I/O bandwidth is described in [123] to predict the power consumption of a server. A power model that provides a per-component power breakdown of a multicore CPU is presented by Bertran et al. [124]. It is based on activity factors obtained from PMCs for various components in a multicore CPU.

Basmadjian et al. [125] report that the summation of power consumption of all active cores to derive the total power consumption is inaccurate. They take into account, therefore, the resource sharing in their power prediction model for multicore processors. McPAT [126] is an integrated power, area, and timing modeling framework for multicore, manycore and multithreaded architectures. It supports the estimation of power consumption for various components in a multiprocessor which includes in-order and out-of-order processor cores, shared caches, integrated memory controllers and networks-on-chip. However, Xi et al. [127] report the limitations of McPAT in power estimation.

Dargie et al. [58] use the CPU utilization statistics to model the relationship between the power consumption of the multicore processor and workload quantitatively. They demonstrate that the relationship is quadratic for a single-core processor and linear for multicore processors. Haj-Yihia et al.[108] present a linear regression model for Intel Skylake processors based on PMCs. They selected the PMCs which are popular in well-known energy and power models.

Lastovetsky et al. [18] present an application-level energy model where the dynamic energy consumption of a processor is represented by a function of problem size. They report a highly non-linear and non-convex nature of the relationship between energy consumption and problem size. They use this model for solving the optimization problems of data-parallel applications on homogeneous multicore clusters for energy.

Energy predictive models for accelerators: Hong et al. [128] present an energy model for an Nvidia GPU based on a similar PMC-based power prediction approach of [116]. The power consumption of the GPU is calculated as the summation of power consumption of all the components composing the GDDR memory and Streaming Multiprocessor (SM) with an error of 8.94%. However, it contains a large set of parameters and requires the detailed architectural information. This is the main factor affecting the portability of this model. Nagasaka et al. [129] present a statistical approach to predict power consumption of GPU kernels. Their model employs GPU PMCs exposed for CUDA applications. The average and maximum errors reported in prediction of total power consumption are 4.7% and 23% respectively.

Song et al. [130] present power and energy prediction models that are based on machine learning algorithms such as back propagation in artificial neural networks (ANNs). The selected 10 GPU PMCs as the predictor variable of their model. The PMC values are collected using CUPTI during the application run. They report an average prediction error rate for their power model as 2.1%, and 11% as maximum for their energy model.

Shao et al. [131] develop an instruction-level energy consumption model for a Xeon Phi processor. They report the error of their model ranging between 1% to 5% for real-world applications. Khatib et al. [132] present a linear instruction-level model to predict the dynamic energy consumption by soft processors in FPGA. Their model considers both the the operand values of the instructions and the inter-instruction effects. They report an average error rate for their model as 4.7% where the maximum reaches up to 12%.

2.3 Critiques of built-in power sensors and PMC based predictive modelling

While the built-in power sensors and PMC based predictive models can provide the fine-grained component-level power consumption, many researchers have highlighted their limitations and poor prediction accuracy.

2.3.1 On-chip integrated power sensors

Burtsher et. al [47] find inaccurate power readings on Nvidia K20c and K20m GPUs which lag behind the expected profile. Furthermore, the authors observe that the power sampling frequency on K20 GPUs varies greatly and the GPU sensors do not update the power readings regularly.

In [37], we present a comprehensive study comparing the accuracy of state-of-the-art on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, which we consider to be the ground truth. It is found that the dynamic energy profile patterns of the on-chip sensors (of CPU, GPU, and Xeon Phi) differ significantly from the patterns obtained using the ground truth, which suggests that the measurements using on-chip sensors do not capture the holistic picture of the dynamic energy consumption during an application execution. Furthermore, it is demonstrated that inaccurate energy measurements with on-chip sensors for dynamic energy optimization can result in a significant loss of energy. The average error of dynamic energy measurements with on-chip power sensors is found to be as high as 73%, and can be 32% for energy predictive models employing PMCs as predictor variables for the benchmark suite used in their experiments.

Another important finding is that owing to the nature of the deviations of the energy measurements provided by on-chip sensors from the ground truth, calibration can not improve the accuracy of the on-chip sensors to an extent that can allow them to be used in optimization of applications for dynamic energy. We also demonstrate that using inaccurate energy measurements provided by on-chip sensors for dynamic energy optimization can result in significant energy losses up to 84%. In [21], the similar results are found demonstrating the poor accuracy of the energy measurements provided by on-chip power sensors for constructing the additive energy models of application-components of a hybrid application.

Our works [37] and [21] differ from that in References [62] and [63] in several ways:

- The authors compare the total power consumption by the system with AC power and power consumption by the micro-benchmarks with RAPL and therefore use different reference domains. However, we compare the dynamic energy consumption by applications with both tools (RAPL and power measurements using external power meters) and thus compare the measurements using the same reference domain.
- The authors run benchmarks in different threading/frequency configurations. In contrast, we build the energy profiles of scientific applications representing real-world workloads using different configurations such as problem size, CPU Cores, CPU Threads.
- The authors run their benchmarks on the Haswell platform only. However, our experiment testbed is more diverse and includes advance generations of Intel CPU micro-architecture.
- The authors find a correlation between the measurements with both tools (power meters and RAPL) on Haswell. However, they could not confirm if RAPL can be calibrated owing to different reference domain. We further extend the knowledge-base by showing that the measurements with both tools can not be calibrated due to their qualitative differences and interlacing behavior of the profiles built with them.
- We demonstrate that a high correlation coefficient does not necessarily mean a strong linear relationship or high similarity between two energy profiles [77].

Summary: To summarize, on-chip integrated power sensors provide fine-grain component level power consumption details. However, there are some issues with the power data values provided by these vendor-specific libraries. The fundamental issue with this measurement approach is the lack of information about how a power reading for a component is determined during the execution of an application utilizing the component. Apart from accuracy, the other issues include the lack of details on update frequency of power readings, portability, poor documentation, etc. as discussed in section 1.1.5.

2.3.2 PMC based Energy predictive models

Economou et al. [50] highlight the fundamental limitation of PMC-based models, which is the restricted access to read PMCs (generally four at a single run of an application). Therefore, It becomes an extremely important task to carefully select the best subset of PMCs as suitable contenders to be used as predictor variables in a model.

In a study on the accuracy of predictive power models for multicore architectures, McCullough et al. [51] report that the prediction errors of linear regression-based models employing PMCs can be as high as 150%. In a survey on predictive energy models for heterogeneous and hierarchical node architectures, O'Brien et al. [15] highlight the poor prediction accuracy and ineffectiveness of such models to accurately predict the dynamic power consumption by modern heterogeneous hybrid NUMA nodes. They highlight the main causes of this inaccuracy as the severe resource contention due to the tight integration of tens of cores contending for shared on-chip resources such as LLC; interconnect (For example: Intel's QPI, AMD's Hyper Transport), and DRAM controllers; Non-uniform memory access (NUMA); and DPM of multiple power domains (CPU sockets, DRAM).

Arsalan et al. [52] also question the reliability and reported prediction accuracy of PMC based energy predictive models. They report that many PMCs that are used as key predictor variables in state-of-the-art predictive models are not deterministic. To improve the accuracy of PMC based energy models, they propose a novel selection criterion called the *additivity* for selecting a subset of PMCs to be used in linear energy predictive models by conforming to the theory of energy predictive models for computing. The criterion is based on an experimental observation that dynamic energy consumption of serial execution of two applications is equal to the sum of the dynamic energy consumption of those applications when they are run separately. The PMC is branded as *non-additive* on a platform if there exists an application for which the calculated value differs significantly from the value observed for the application execution on the platform. The use of *non-additive* PMCs in a model impairs its prediction accuracy. In another study [78], the authors demonstrate how the accuracy of state-of-the-art energy predictive models based on three popular techniques (Linear regression, Neural networks, and Random forests) can be improved by selecting PMCs based on the property of additivity.

Arsalan et al. [54] compared the techniques for energy predictive modelling using PMCs on modern multicore CPUs. The authors studied two following types of energy predictive models: i) linear regression models employing PMCs based on the property of additivity, and ii) the sophisticated statistical learning models (random forest and neural network) employing PMCs based on correlation and principal component analysis. The authors conclude that a strong positive correlation of model variables (PMCs) with energy consumption is not sufficient enough to provide good prediction accuracy. Therefore, the author suggested that it should be combined with methods such as additivity that consider the physical significance of the model variables originating from the theory of energy predictive models for computing.

Summary: To summarize, energy optimization at system-level and application-level rely crucially on accurate measurement of energy consumption during an application execution. Energy predictive models using performance monitoring counters emerged as a dominant measurement method. Its main advantage compared to the ground truth (system-level physical measurements using power meters) is the fine-grained decomposition of energy consumption during the execution of an application. However, there are several shortcomings with this approach including:

- Model parameters in most cases are not deterministic.
- Complexity of model construction and lack of consensus among the research works, which report prediction accuracies ranging from poor to excellent.
- A vast majority of research works select PMCs solely on the basis of their high positive correlation with energy consumption without any deep understanding of the physical significance of the model variables.
- A sound theoretical framework to understand the fundamental significance of the model variables with respect to the energy consumption and the causes of inaccuracy or the reported wide variance of the accuracy of the models is lacking.

In [21], the authors discussed the implementations complexity and lack of portability of PMC based energy predictive models which hinders their efficacy to adopt them as a viable approach to predict the energy consumption by the hybrid applications running in parallel on different compute devices. Energy predictive models employing PMCs as predictor variables exhibit high implementation complexity due to the following reasons:

1. There is a large number of PMCs provided in a modern multicore processor to be considered.
2. Tremendous programming effort and time are required to automate and collect all the PMCs. This is because of the limited number of hardware registers available on platforms for storing the PMCs. Only 3-4 PMCs can be collected in a single run of an application. Moreover, some PMCs can only be collected individually or in sets of two or three for an application run. Therefore, each application must be executed for a number of times to collect all the PMCs available on a specific platform.
3. An energy predictive model purely based on PMCs lacks portability. This is because all the PMCs available for a CPU processor may not be present in a GPU processor due to inherent architectural differences, or even for the next-generation CPU processor from the architecture space.

2.4 Energy Optimization Approaches

In this section, we present some notable power mechanism existing in modern hardware to save the overall power consumption by the system. Afterwards, we provide an overview of some notable energy optimization approaches to achieve the bi-objective optimization for performance and energy consumption on modern computing platforms.

2.4.1 Power Saving Mechanisms

Several power saving mechanisms are provided to control and configure the power consumption in modern computing hardware.

Clock and power gating: *Clock gating* [133] is a popular technique which is used in many synchronous circuits for reducing the dynamic power dissipation. It saves power by pruning the clock tree when the circuit is not in use. This disables the portions of the circuitry which stops the flip-flops in them from switching the states. As a result, the switching (dynamic) power consumption goes to zero. *Power gating* is a technique to reduce the stand-by or leakage power by shutting off the current to the inactive blocks of the circuit. In order to achieve this, sleep transistors are used to connect the circuitry blocks to the power supply [134]. Hence, both techniques together save the overall (static and dynamic) power consumption by the circuitry. However, the studies [135] have highlighted the negative effects of these techniques such as performance penalty, area and power overhead, etc.

DPM: DPM techniques target to reduce the power dissipation by selectively turning off the idle components [134]. The basic idea is to put the components to sleep; switched to low power modes; or turned off the current when idle, and then bring them back into active state when required. However, it may hit the performance and can introduce the energy overheads for sleep-state transition.

Advanced Configuration and Power Interface (ACPI): Advanced Configuration and Power Interface (ACPI) is an open standard co-developed by Intel, Microsoft, HP, Toshiba Phoenix and Huawei to configure and perform power management of the system [136]. It is the central to operating system directed configuration and power management. Here, we briefly present the configuration and control of the processors power and performance states as per described in section 8 of ACPI version 6.3 [136].

It defines four following global (system-level) states:

1. **G0 Working**: where the system is executing the user tasks.

2. **G1 sleeping**: where the system is not executing the user tasks, and appears to be off. The system consumes a small amount of power in this state. Example are sleep/standby and hibernate modes in modern OS.

3. **G2 Soft Off**: where the system consumes minimal amount of power. It is almost the same as G3 Mechanical Off. However, in this state the PSU still supplies power to return to G0 state.

4. **G3 Mechanical Off**: where the system power is totally removed.

The power states for the processors {C0, C1,$\cdots$,Cn} are defined by ACPI during G0 Working. Please note here that the Cx states only apply to G0 state. **C0** state is defined as *active power state* where the CPU is executing the instructions. **C1** is defined as halt power state where the CPU is not executing any instruction but it can return to C0 instantaneously. **C2** is defined as stop clock state which is similar to C1 state but it takes longer to return to C0. **C3** is defined as (deep) sleep state which offers improved power savings than C1 and C2. However, it requires more time to return to C0 than required by C2. Additional C states can be defined by manufacturers to provide more control on power savings of processors. However, all the power states from C1 through Cn are related to processor sleep states where the processor consumes less energy and dissipate less power in comparison with C0.

C-sates are further divided into core states (CC-state) and package states (PC-states) to support core-level power management in modern multi-core processors. **CC-states** {CC0, CC1,$\cdots$,CCn} manages the C-states at CPU core-level, whereas PC-states {PC0, PC1,$\cdots$,PCn} manages the power states of other common (shared) resources in processor such as shared cache. xC0 (where x denotes the initial of package state or core state i.e. P or C respectively) state represents the active state for package and core states, whereas all other CCx and PCx states from xC1 through xCn represent different levels of sleep states. PC-sates are dependent on CC-states and thus cannot be interacted directly. They are changed based on the CC-states of the CPU cores. Hence, package state is active when there is at least one CPU core state is active.

The performance states {P0, P1,$\cdots$,Pn} are defined as the power consumption and capability states within the active state for processors or device. **P0** is when the device or processor uses its maximum performance capability and thus may consume the maximum allowed power. **P1** and other subsequent P-states represent different level of states when the performance capability and power consumption by a processor/device is limited below its maximum performance capability and power consumption. **Pn** represents the P-state when the performance capability and power consumption by the processor or device is at its minimum level while remaining in an active state. While C-states mechanism saves the power consumption by putting the idle components into lower power states, P-states allows to save the power consumption by decreasing the voltage or/and frequency of the processor. Therefore, P-states are also known as power-performance states and represents the pair of specific voltage and frequency.

DVFS: DVFStechnique allows changing the processors voltage and frequency at run time [137], and therefore allows the processors to run at different clock speeds and supply voltages. However, there is trade-off between the time and energy by reducing the clock frequency and the voltage of the processors. Furthermore, running an application on lower frequency does not necessarily reduce its overall energy consumption. This is because the execution time of the application may be increased when setting the clock frequency of the processor running it at a lower speed.

The voltage input scales up/down in accordance with the change in clock frequency of processor cores. CPU frequency can be scaled up/down dynamically in response to ACPI events, manually by userspace programs, or automatically depending on the system load. It allows to save the power consumption of the processor by decreasing its clock frequency (and eventually the voltage input) while in active state (i.e. xC0). CPU frequency scaling is implemented in Linux kernels by using the power schemes known as *governors* [138]. Typically, there are following governors supported by Linux kernel: 1) **Performance:** The CPU frequency is set to be at highest possible frequency, 2) **Powersave:** The CPU frequency is set to be at lowest possible frequency, 3) **Ondemand:** The CPU frequency is set at depending on the current system load, 4) **Userspace:** The CPU frequency is set at user specific value. There are some additional governors such as *conservative*, *schedutil*, etc. also supported in some Linux kernels. However, only one governor is active at a time.

Voltage change of the components are supported through voltage regulators (VRs) [139]. A VR performs two following major functions in providing voltage to a computing device: i) it stabilizes the supplied voltage, and ii) it changes the supplied voltage according to the needs of the device. It is important to note that the same voltage is supplied to all components sharing the same *voltage domain*. A *voltage domain* can be defined as a set of components that is attached to a common supply voltage. The number of voltage domains depends upon the number of VRs and their architectural implementation scheme. For example, modern Intel processors are based on one of the two following power delivery schemes: i) **Integrated Voltage Regulator (IVR) scheme**: It uses one motherboard VR and six different on die/package fully integrated VRs, and ii) **Motherboard Voltage Regulator (MBVR) scheme**: It uses four motherboard VRs and three on-die power-gates for the Ring domains and Core0/1. It is important to note that the components which share the same voltage domain cannot regulate their voltage independently from each other. Therefore, only the frequency can be scaled dynamically for the cores/components sharing the same voltage domain.

2.4.2 Multi-objective Optimization Methods Involving Energy

System-level multi-objective optimization: System-level multi-objective optimization methods aim to optimize several objectives of the system or the environment (for example: clouds, data centers, etc) where the applications are executed. Mezmaz et al. [140] propose a parallel bi-objective genetic algorithm to maximize the performance and minimize the energy consumption in cloud computing infrastructures. The parameters used in their method are the computation cost of a task and the communication costs between two tasks. The decision variable is the supply voltage of the processor. Beloglazov et al. [141] propose heuristics that consider twin objectives of energy efficiency and Quality of Service (QoS) for provisioning data center resources. The decision variables are the number of VMs and clock frequencies. Durillo et al. [142] propose a multi-objective workflow scheduling algorithm that maximizes performance and minimizes energy consumption of applications executing in heterogeneous high-performance parallel and distributed computing systems. A machine is characterized using nine parameters (from technology(nm) to TDP). They study the impact of different decision variables such as number of tasks, number of machines, DVFS levels, static energy, and types of tasks. Kolodziej et al. [143] propose multi-objective genetic algorithms that aim to maximize performance and energy consumption of applications executing in green grid clusters and clouds. The performance is modeled using computation speed of a processor. The decision variable is the DVFSlevel.

Application-level multi-objective optimization: Application-level solution methods optimize applications rather than the executing environment. Marszalkowski et al. [144] analyze the impact of memory hierarchies on time-energy trade-off in parallel computations, which are represented as divisible loads. They represent execution time and energy by two linear functions of problem size, one for in-core computations and the other for out-of-core computations. Lastovetsky et al. [18] and Reddy et al. [145] propose data partitioning algorithms that solve single-objective optimization problems of data-parallel applications for performance or energy on homogeneous clusters of multicore CPUs. They take as an input, discrete performance and dynamic energy functions with no shape assumptions and that accurately and realistically account for resource contention and NUMA inherent in modern multicore CPU platforms.

Reddy et al. [19] propose a solution method to solve bi-objective optimization problem of an application for performance and energy on homogeneous clusters of modern multicore CPUs. They demonstrate that the method gives a diverse set of Pareto-optimal solutions and that it can be combined with DVFS-based multi-objective optimization methods to give a better set of (Pareto-optimal) solutions. The methods target homogeneous HPC platforms. Chakraborti et al. [26] consider the effect of heterogeneous workload distribution on bi-objective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters.

Fredy et al. [146] proposed a dynamic energy-aware scheduler for parallel task-based application in cloud computing. The scheduler aims to minimize a multi-objective function that combines the energy consumption and the total execution time by an energy-performance importance factor. The authors also proposed a model to estimate the energy consumption by an application by aggregating the utilization of different elements during its task execution such as VM management, data transfers and the background services of the node. Khaleghzadeh et al. [20] propose a solution method solving the bi-objective optimization problem on heterogeneous processors and comprising of two principal components. The first component is a data partitioning algorithm that takes as an input discrete performance and dynamic energy functions with no shape assumptions. The second component is a novel methodology [21] employed to build the discrete dynamic energy profiles of individual computing devices, which are input to the algorithm.

Energy Proportionality of Multicore Processors: Architects of modern multicore processors follow a key design goal called energy proportionality (EP) defined by Barroso et al. [147] as designing microprocessors composed of components that consume energy proportional to the amount of work performed. Wong et al. [148] show that EP is not uniform across various server utilization levels by proposing metrics that they believe accurately quantifies EP. Lo et al. [149] present a server power management solution that adjusts the power in a fine-grained manner based on latency statistics with an aim to achieving EP objective. Hsu et al. [150] examine a range of metrics for quantifying EP. Sen et al. [151] extend the definition for re-configurable processors.

Khokhriakhov et al. [88] study the practical implication of EP for applications, which is signified by a monotonically increasing relationship between energy and execution time. A breach of this relationship represents a violation of EP. They discover through their novel application-level method for bi-objective optimization of the applications for energy and performance on a single multicore processor, that EP does not hold true for modern multicore processors.

2.5 Approaches for Measuring the Goodness of Energy Profiles of Applications Executing on Multicore Comptuing Platforms

In this section, we first present the overview of approaches that are commonly used to measure the goodness of energy profiles. Then, we present the notable approaches used for closely related problem of pattern matching in other faculties such as text mining, graph theory and time-series analysis.

2.5.1 Accuracy measurement approaches used in energy modeling

The average prediction error (also known as relative error) is the most commonly used statistical measure to determine the accuracy of energy models against the ground truth. Let $E(x)_M$ represents the model of dynamic energy consumption by an application workload size x and $E(x)_R$

represents the real dynamic energy consumption by the same application workload size x, then the prediction error in percentage is calculated as [57]

$$\epsilon = |(E(x)_R - E(x)_M)|/E(x)_R \times 100 \tag{2.9}$$

and the average error in percentage for n data values is calculated as $1/n \sum_{i=1}^{n} \epsilon_i$. A plethora of research work including [15, 21, 37, 58, 59, 51] use average, maximum and minimum prediction errors to determine the accuracy of energy profiles.

Pearson's correlation coefficient is also a common and widely used similarity measure. The correlation between (two series) the ground truth (x) and a model (y) both of length n is represented as [56]:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{2.10}$$

where $\overline{x}$ and $\overline{y}$ represents the sample means of x and y. The coefficient value ranges between -1 and 1 where -1 represents a perfect negative relationship (as one of x or y increases, the other decreases), 1 represents a perfect positive relationship (as both x and y decrease or increase together) and 0 represents an absence of any relationship. References [60, 61, 62, 63] are some of the notable works which used the correlation coefficient to determine whether the energy profiles follow the ground truth.

Critiques of Average Error and Correlation Coefficient: Juan-Antonio et al. [64] argue that the relative error is low for a model that underestimates than a model that overestimates when the proportion for the underestimated and the overestimated values of $E(x)_M$ with $E(x)_R$ is the same. This can negatively impact the interpretation of the results. The authors propose the proportional error μ to as an alternate to relative error. The proportional error for model prediction $E(M)$ with the ground truth $E(R)$ is a ratio of the maximum of the two values with the minimum of the two values. Let μ represents the proportional error which can be calculated as $\mu = \frac{max(E(R),E(M))}{min(E(R),E(M))}$. μ is always greater than 1 if there exists an error, and equal to 1 otherwise.

Fahad et al. [21] highlighted the inadequacies of relative errors and correlation coefficient in capturing the holistic picture of the energy consumption trend of the profiles. The authors demonstrate that the two statistical measures are blind to the qualitative differences of the energy models and the ground truth. They used mean absolute deviation (MAD) around the sample mean and maximum absolute deviation around sample maximum, and percentage increase and decrease between two successive data-points of the energy profiles to compare the qualitative behavior (such as the energy consumption trend and variations) of the energy profiles.

2.5.2 Pattern matching approaches in other fields for closely related problem

This similarity measure problem, in general, can be classified as *pattern matching*. A plethora of different methods and approaches have been proposed to solve this problem in different fields such as data mining, time series similarity analysis, and graph (matching) theory. The proposed solutions can be categorized into a non-exhaustive list of categories such as lock-step, pattern-based, model based, elastic, feature-based and et cetra.

However, the main difference of similarity matching in energy modeling with time-series analysis is that the energy profiles are (usually) a function of application configuration parameters (such as problem sizes, CPU threads, CPU cores, etc.), whereas the time series data is uni-variate (only one variable varies over time). Furthermore, the energy profiles are of the same cardinality as of the ground truth (i.e. they have equal lengths) whereas it is not necessary in case of graph matching, time series, and data mining. Therefore, we do not cover the similarity measures commonly known as elastic similarity measures (such as dynamic time warping (DTW) [66]) where a data point of a time series is compared with many data points of other time series and vice-versa i.e. one-to-many mapping and many-to-one mapping. Now we present some of the popular similarity measures and distance metrics used to determine the pattern matching.

Similarity Measures: Similarity measures establish the resemblance between the ground truth and a model as an absolute value usually within the interval of [0,1] or [-1,1] where 0 or -1 indicates an absolute opposite and 1 denotes a maximum similarity. A common and most widely used similarity measure is Pearson's correlation coefficient or cross-correlation coefficient. The correlation coefficient between two energy profiles can be determined using the equation 2.5.1.

Cosine similarity is another popular similarity measure which measures the orientation of two non-zero n-dimensional vectors irrespective of their magnitude. It is mainly used in text mining problems such as text classification, text summarization, information retrieval, question answering, etc. [65]. It measures the cosine of the angle between the both vectors. It is calculated by the dot product of the vectors and normalized by the product of their lengths. Let x and y represents two non-zero n-dimensional vectors. Then, the cosine similarity between both vectors can be determined as [73]:

$$cos(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{||\boldsymbol{x}||\cdot||\boldsymbol{y}||} = \frac{\sum\limits_{i=1}^{n} x_i y_i}{\sqrt{\sum\limits_{i=1}^{n} x_i^2}\sqrt{\sum\limits_{i=1}^{n} y_i^2}} \tag{2.11}$$

T. Nakamura et al. [67] proposed a shape-based similarity measure called the Angular Metric for Shape Similarity (AMSS) for time series data. It adopts cosine similarity as a principle measure which disregards exact points or vector length.

Distance Metrics: Another way to compare a profile against the ground truth is the use of a distance metric which establishes an absolute value of how far the two objects are. The most commonly used distance metric is Euclidean distance which is based on Pythagoras' theorem and measures the distance between two vectors or two points in a Euclidean space. It is one of the most popular distance metrics in data mining and time series similarity comparison. It is also used for clustering of time series data [74] [75] because of its indexing capabilities and simple computations. It can be determined between two vectors or two points using following equation [73]:

$$d_E(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.12}$$

The average geometric distance or root mean square distance is:

$$d_{rms}(x,y) = \frac{d_E(x,y)}{n} \tag{2.13}$$

Manhatten distance is related to Euclidean distance, and also called City-block or Taxicab distance. It measures the distance between two points as sum of absolute differences of their Cartesian product:

$$d_{Man}(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.14}$$

The generalized form of Euclidean distance and Manhatten distance is Minkowski distance which is also called L_p norm, and can be computed as [152]:

$$d_{Mink}(x,y) = \sqrt[1/p]{\sum_{i=1}^{n} |x_i - y_i|^p} \tag{2.15}$$

For Manhatten distance p=1, and for Euclidean distance p=2. However, Minkowski distance variants are blind to capture the data correlation. Minkowski measures belong to the family of lock-step measures which compare each element (data-point) in (energy) model with their corresponding element (data-point) in ground truth. Mahalanobis distance is a variant of Euclidean distance which takes the data correlation into account.

$$d_{Mahal}(x,y) = \sqrt{(x-y)S^{-1}(x-y)^T} \tag{2.16}$$

where S is the co-variance matrix.

Graph-Edit-Distance (GED) [76] is a widely used distance metric in pattern recognition or graph matching. It is related to string-edit-distance between two strings. It computes the the cost of recognition of nodes and minimum number of modifications (such as deletion,insertion, substitution) required to transform the input graph into the referenced one. However, the complexity of computing GED is non-polynomial [153].

Auto-regressive (AR) modeling is a model-based approach that extracts the features from time-series to use their underlying models to determine the similarity between them. AR modeling specifies that the current value in a data-set (time-series) depends linearly on its preceding value(s). The autoregressive integrated moving average (ARIMA) also called Box-Jenkins [68] method, is a popular approach used in time-series analysis and for anomaly detection [69]. The main idea of using AR modelling to measure the goodness is to learn the models of time-series and then compute the goodness using the model parameters.

Several approaches can be found in the literature which use AR to find the similarity between two time-series such as [70], [71], [72]. However, the basic assumption of AR modelling is that the data (time-series) is uni-variate and the future value depends upon the past value(s). This is not the case with energy modeling because application energy profiles are (usually) a function of application parameters, and each data-point in an energy profile is distinct and independent of all other data-points in that profile. Furthermore, AR modelling assumes that the data is stationary which means that the statistical properties such as mean and variance of the time-series do not change over time. On the contrary, the energy profiles are not stationary and there also exists an energy consumption trend. Therefore, AR modelling approach is not applicable straightforwardly for determining the goodness of energy profiles.

Chapter 3

A Methodology to Determine the Energy Consumption by An Application Using System-level Measurements

3.1 Introduction

Accurate measurement of energy consumption during an application execution is pivotal to many interesting applications such as application-level energy optimization approaches, energy efficiency analysis, auto-tuning, and energy aware dynamic task scheduling. However, a fundamental challenge is how to measure the energy consumption by an application during its execution accurately and reliably.

The power measurements by the power meters are considered to be the most accurate at system level [40]. However, they only provide the energy consumption details on system-level, and therefore lacks the ability to provide fine-grained component-level decomposition of the energy consumption by an application. This is a serious drawback. Because, this decomposition is essential to energy models that are the key inputs to data partitioning algorithms that are fundamental building blocks for energy optimization of an application. Without the ability to determine accurate decomposition of the total energy consumption, one has to employ an exhaustive approach (involving huge computational complexity) to determine the optimal data partitioning that optimizes the application for energy as discussed in section 1.1.3. We bridge the gap in this chapter by presenting a methodology to accurately and reliably determine the energy consumption by an application using the system-level power measurements.

The rest of the chapter is organized as follows. First, we present the details on static energy and dynamic energy consumption and the rationale of using dynamic energy consumption in section 3.2. Then, the methodology and implementations details of HCLWattsUp API to determine the energy consumption by an application during its execution are explained in section 3.3. Finally, the details on precautions and the statistical methodology to ensure the reliability and accuracy of the energy measurements using the proposed methodology are explained in sections 3.4 and 3.5 respectively.

3.2 Energy Consumption by the application

Energy (*E*) can be defined as the total amount of work performed by the system over a time period (*T*), whereas power (*P*) is the rate at which the work is done by the system. Energy of computing is determined as the product of power and time, as shown in equation 2.1. It is also considered as the integration of power values for a time period T, starting from $t1$ till $t2$ and depicted in equation 2.2. Thus, it is computed indirectly from power and time.

While the execution time can be measured accurately using high precision CPU clocks, there is no equivalently effective way to measure the power consumption by an application. The popular approaches to measure it can be categorized as follows: i) System-level physical power measurements using external power meters, ii) Measurements using on-chip power sensor, and iii) Energy predictive models. The power measurements by the power meters are considered to be accurate at system level [40]. However, it provides the physical power measurement at a computer level only and therefore lacks the ability to provide the fine-grained decomposition of the energy consumption by the application executing on several independent computing devices on a computer.

In section 2.1.4, the two components of total energy are presented as: i) *dynamic energy*, and ii) *static energy*. From the platform point of view, we define dynamic and static energy consumption as the energy consumption of the whole system with and without the application execution respectively. Here, the static energy is the base energy of the system which is an intrinsic property of the system, as it consumes this base energy all the times regardless of doing any work or performing any computation. However, the dynamic energy is the largest component of the total energy consumption by the application when running on a system.

Linux kernels supports different governors to control the CPU frequency as explained in section 2.4.1. While *Performance* and *Powersave* scale the CPU frequency at highest possible or lowest possible value respectively in order to maximizing the performance or minimizing the energy, *Ondemand* provides a balance between the two. It sets the CPU frequency depending on the current system load. The current settings of CPU frequency governor also influence the energy consumption by the system. In a typical settings of Ondemand CPU frequency governor, the system in its idle state consumes the lowest and steady amount of (static) energy (due to deep-sleep power/performance state). However, the processor's power and performance state gets changed to xC0 (active state) as soon as it gets a workload to execute. Consequently, it consumes more energy than static energy to get the work done to execute the application for a time period Δt as shown in figure 3.1. This rise in energy consumption of the system attributes to the work done by the platform components during the executing of the application. We call this difference as dynamic energy consumption by the application (or dynamic energy for the sake of simplicity).

The total energy consumption is the sum of dynamic and static energy consumption. The static energy is the intrinsic property of system power and independent of any application configuration. Hence, the static energy consumption can be determined as the idle power of the platform (without application execution) multiplied by the execution time of the application. Whereas, dynamic energy can be determined by subtracting the static energy for this time period Δt from the total energy of the system that it consumed during the application execution.

Consider, for example, an application workload of size x to be executed by a given platform. Let the dynamic energy function of the system executing this workload size x can be represented by

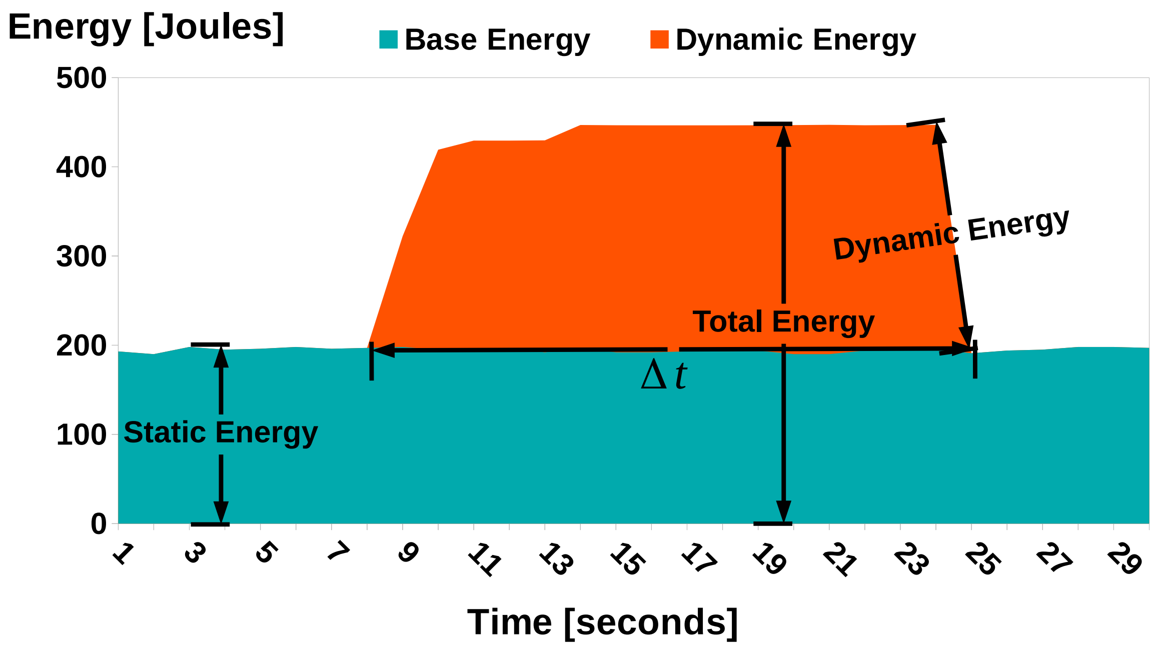


Figure 3.1: Static and Dynamic energy consumption

$E(x)$. Now, if t is the execution time of the application then dynamic energy $E(x)$ can be determined as

$$E(x) = E_{total} - (P_{base} \times t) \tag{3.1}$$

Here, P_{base} is the base or static power of the platform and E_{total} is total energy consumption of the platform during the execution of the application workload size x. The total energy consumption E_{total} is the area under the discrete function of the power samples provided by the power meter versus the time intervals between the samples. Well-known numerical approaches such as trapezoidal rule can be used to numerically approximate this area. The trapezoidal rule works by approximating the area under a curve using trapezoids rather than rectangles to get better approximations.

The execution time t of the application can be determined accurately using the processor clocks. The accuracy of the static power consumption P_{base} is equal to the accuracy provided in the specification of the power meter. Hence, the accuracy of obtaining the total energy consumption E_{total} employing the system-level measurements using the external power-meters is subject to the accuracy of the power-meter. The revenue-graded power meters such as Yokogawa WT5000 [98] offers the basic power accuracy of up to $\pm 0.03\%$ with a very high sampling rate of 10M per second. Hence, one can determine the dynamic energy consumption by the application within the accuracy provided by the external power-meter power readings. Therefore, this approach is considered as the ground truth through out this thesis.

The advancements in power saving mechanisms has resulted in minimizing the energy consumption by a system in its idle state. However, an application utilizing the system at its full capacity consumes a much higher energy such that the static energy becomes merely a low fraction of the total energy. Consider, for example, the workload size 25600×25600 of the hybrid application DGEMM executing on all three compute devices (CPU, GPU, and Xeon Phi) on HCLServer01 (technical description are provided in table 4.1). It consumes 837 watts on average during its execution on all three compute devices. The idle power consumption of HCLServer01 is 201 watts (which is just about 24% of the total power consumption on average). Thus, it consumes 3x more power than the static power to execute DGEMM at its full capacity. Hence, the dynamic energy is a dominating factor of the total energy consumed by HCLServer01 during the execution of given workload size of DGEMM. Therefore, we consider only the dynamic energy consumption by an application in this thesis including the following reasons:

1. Static energy consumption is a constant (or a inherent property) of a platform that can not be optimized. It does not depend on the application configuration.

2. Although static energy consumption is a major concern in embedded systems, it is becoming less compared to the dynamic energy consumption due to the advancements in hardware architecture design in HPC systems as explained in section 2.4.1.

3. We target applications and platforms where dynamic energy consumption is the dominating source of energy dissipation.

4. Finally, we believe its inclusion can underestimate the true worth of an optimization technique that minimizes the dynamic energy consumption. We elucidate this using two examples from published results.

   - In our first example, consider a model that reports predicted and measured total energy consumption of a system to be 16,500 J and 18,000 J. It would report the prediction error to be 8.3%. If it is known that the static energy consumption of the system is 9000 J, then the actual prediction error (based on dynamic energy consumption only) would be 16.6% instead.
   - In our second example, consider two different energy prediction models (M_A and M_B) with same prediction errors of 5% for an application execution on two different machines (A and B) with same total energy consumption of 10,000 J. One would consider both the models to be equally accurate. But supposing it is known that the dynamic energy proportions for the machines are 30% and 60%. Now, the true prediction errors (using dynamic energy consumption only) for the models would be 16.6% and 8.3%. Therefore, the second model M_B should be considered more accurate than the first.

Now, we explain the methodology to obtain the dynamic energy consumption reliably and accurately employing the system-level power measurements.

3.3 API for Power Measurements Using External Power Meter Interfaces (HCLWattsUp)

In a typical settings, an HPC system have a dedicated power meter installed between its input power sockets and wall A/C outlets as shown in figure 2.1. The power meter captures the total power consumption of the node. In case of Watts Up Pro? power meter which is installed with

HCLServers (technical descriptions of each node is presented in tables 4.1, 4.2 and 4.3), a data cable is connected to the USB port of the node. A perl script collects the data from the (Watts Up Pro?) power meter using the serial USB interface. The execution of this script is non-intrusive and consumes insignificant power.

A fundamental challenge is how to gather the readings from power meter to determine the energy consumption by an application during its execution. Here, we explain the design principals of *HCLWattsUp* API [38] which gathers the readings from a power meter to determine the average power and energy consumption during the execution of an application on a computing platform. HCLWattsUp API provides following four types of measures during the execution of an application:

- *TIME*—The execution time of the application (seconds).
- *BPOWER*—The average base (idle) power of the system (watts).
- *TENERGY*—The total energy consumption (joules).
- *DENERGY*—The dynamic energy consumption (joules).

The overhead due to the API is very minimal and does not have any noticeable influence on the main measurements. The API is confined in the *hcl* namespace. It is important to note that the power meter readings are only processed if the measure is not *hcl::TIME*. Therefore, there are two runs for each measurement. First run for measuring the execution time, and the second for energy consumption.

The example provided in figure 3.2 illustrates the use of statistical methods to measure the dynamic energy consumption during the execution of an application. Lines 10–12 construct the Wattsup object. The inputs to the constructor are the paths to the scripts and their arguments that read the USB serial devices containing the readings of the power meters.

The principal method of *Wattsup* class is *execute*. The inputs to this method are the following:

1. Type of measure: (one of the aforementioned four measurements)
2. Path to the application: *executablePath*
3. Arguments to the executable (application specific parameters): *executableArgs*
4. The statistical thresholds: (*pIn*)

The followings are the outputs which are calculated during the execution of the executable (application):

1. The achieved statistical confidence *pOut*
2. The estimators
3. The sample mean (*sampleMean*)
4. The standard deviation (*sd*)

```
#include <wattsup.hpp>
int main(int argc, char** argv)
{
std::string pathsToMeters[2] = {
"/opt/powertools/bin/wattsup1",
"/opt/powertools/bin/wattsup2"};
std::string argsToMeters[2] = {
"--interval=1",
"--interval=1"};
hcl::Wattsup wattsup(
2, pathsToMeters, argsToMeters
);
hcl::Precision pIn = {
maxRepeats, cl, maxElapsedTime, maxStdError
};
hcl::Precision pOut;
double sampleMean, sd;
int rc = wattsup.execute(
hcl::DENERGY, executablePath,
executableArgs, &pIn, &pOut,
&sampleMean, &sd
);
if (rc == 0)
std::cerr << "Precision NOT achieved.\n";
else
std::cout << "Precision achieved.\n";
std::cout << "Max repetitions "
<< pOut.reps_max
<< ", Elasped time "
<< pOut.time_max_rep
<< ", Relative error "
<< pOut.eps
<< ", Mean energy "
<< sampleMean
<< ", Standard Deviation "
<< sd
<< std::endl;
exit(EXIT_SUCCESS);
}
```

Figure 3.2: Example illustrating the use of HCLWattsUp API for measuring the dynamic energy consumption

The *execute* method repeatedly invokes the executable until one of the following conditions is satisfied:

- The maximum number of repetitions specified in $maxRepeats$ is exceeded.
- The sample mean is within $maxStdError$ percent of the confidence interval cl. The confidence interval of the mean is estimated using Student's t-distribution.

- The maximum allowed time $maxElapsedTime$ specified in seconds has elapsed.

If any one of the conditions are not satisfied, then a return code of 0 is output suggesting that statistical confidence has not been achieved. If statistical confidence has been achieved, then the number of repetitions performed, time elapsed and the final relative standard error is returned in the output argument $pOut$. At the same time, the sample mean and standard deviation are also returned. For our experiments, we use values of (1000, 95%, 2.5%, 3600) for the parameters $(maxRepeats, cl, maxStdError, maxElapsedTime)$. Since the Student's t-distribution is used for the calculation of the confidence interval of the mean, it is confirmed specifically that the observations follow normal distribution by plotting the density of the observations using *R* tool. We also use Pearson's chi-squared test to ensure that the observations follow normal distribution.

3.4 Component-Level Energy Consumption Using HCLWattsUp API

We provide here the details of how system-level physical measurements can be used to determine the energy consumption by a compute device such as a CPU, GPU , etc. during an application execution, using HCLWattsUp API.

For illustration purposes, we define the group of components running a given application as an *abstract processor*. Consider, for example, a matrix multiplication application running on a multicore CPU. The abstract processor for this application, which we call *AbsCPU*, comprises of the multicore CPU processor consisting of a certain number of physical CPU cores and DRAM. In this work, we use only such configurations of the applications which execute on *AbsCPU* and do not use any other system resources such as solid state drives (SSDs), network interface cards (NIC) and so forth. Therefore, the change in energy consumption by the system reported by HCLWattsUp reflects solely the contributions from CPU and DRAM. To eliminate any potential interference form the energy consumption of the computing elements that are not part of the abstract processor such as *AbsCPU*, following several precautions are took:

1. It is ensured that the platform is reserved exclusively and fully dedicated to our experiments.
2. The disk consumption is monitored before and during the application run. It is ensured using the tools such as *sar*, *iotop*, etc. that there is no I/O performed by the application.
3. It is ensured that the problem size used in the execution of an application does not exceed the main memory and that swapping (paging) does not occur.
4. It is ensured that network is not used by the application by monitoring using tools such as *sar*, *atop*, etc.
5. The application kernel's CPU affinity mask is set using SCHED API 's system call SCHED_SETAFFINITY().
6. Fans are also a great contributor to energy consumption. On our platforms, fans are controlled in two zones: (a) zone 0: CPU or System fans, (b) zone 1: Peripheral zone fans. There are following four levels to control the speed of fans:

- Standard: BMC control of both fan zones, with CPU zone based on CPU temp (target speed 50%) and Peripheral zone based on PCH temp (target speed 50%).
- Optimal: BMC control of the CPU zone (target speed 30%), Peripheral zone fixed at low speed (fixed 30%).
- Heavy IO: BMC control of CPU zone (target speed 50%), Peripheral zone fixed at 75%.
- Full: all fans running at 100%.

In all speed levels except the full, the speed is subject to be changed with temperature and consequently their energy consumption also changes with the change of their speed. Higher the temperature of CPU, for example, higher the fans speed of zone 0 and higher the energy consumption to cool down. This energy consumption to cool the server down, therefore, is not consistent and is dependent on the fans speed and consequently can affect the dynamic energy consumption of the given application kernel.

Hence, to rule out the fans' contribution in dynamic energy consumption, we set the fans at full speed before launching the experiments. When set at full speed, the fans run consistently at a fixed speed until we set them to another speed level. Hence, fans consume same amount of power which is included in static power of the platform.

7. The temperature of the platform and speed of the fans (after setting it at full) are monitored with help of Intelligent Platform Management Interface (IPMI) sensors, both with and without the application run. In both scenarios, no considerable difference is found in temperature as a result of setting the fans speed as full.

Convincingly following this methodology, the dynamic energy consumption obtained using HCLWattsUp API reflects the contribution solely by the *abstract processor* executing the given application kernel.

3.5 Methodology to Obtain a Reliable Data Point using HCLWattsUp API

To ensure the reliability of the results, following detailed statistical experimental methodology is followed:

- It is ensured that there are no drastic fluctuations in the load due to abnormal events in the server by monitoring its load continuously using the tool *sar*. Insignificant variation in the load is observed during this monitoring period suggesting normal and clean behaviour of the server.
- The server is fully reserved and dedicated to the experiments during their execution.
- To make sure that pipe-lining, cache effects and so forth do not happen, the experiments are not executed in a loop and a sufficient time (120 s) is allowed to elapse between the successive runs. This cool-down time is based on observations of the times taken for the memory utilization to revert to base utilization and processor (core) frequencies to come back to the base frequencies.

- The typical settings of CPU freq governor (such as Ondemand) keeps the processor at the lowest possible frequency by putting it in deep sleep state (such as xC6) when it is idle. It scales up the CPU frequency at the highest possible level by putting the processor in active state (i.e. xC0) as soon as it gets a workload to compute. On HCLServers, we find following C states C0, C1, C1E, C3 and C6. The transitions between the xC-states may influence the dynamic energy. The aforementioned cool-down time ensures to rule out the entry latency (the latency to enter into idle state) whereas the exit latency (the latency to exit out of an idle state such as xC6) on HCLServers is negligibly minimal. For each aforementioned C-state, the exit latency on HCLServers is as follows:
  - C0: 0 microseconds
  - C1: 2 microseconds
  - C1E: 10 microseconds
  - C3: 33 microseconds
  - C6: 133 microseconds

Furthermore, following detailed statistical experimental methodology is employed to ensure that each data point is reliable and accurate. To achieve this, the application is repeatedly executed to obtain each data point until the sample mean lies in the 95% confidence interval and a precision of 0.025 (2.5%) has been achieved. For this purpose, Student's t-test is used assuming that the individual observations are independent and their population follows the normal distribution. The validity of these assumptions are verified by using Pearson's chi-squared test by plotting the distributions of observations.

The function $MeanUsingTtest$, shown in Algorithm 1, describes this step. The inputs to the function $MeanUsingTtest$ are:

- The application to execute, app
- The minimum number of repetitions, $minReps \in \mathbb{Z}_{>0}$
- The maximum number of repetitions, $maxReps \in \mathbb{Z}_{>0}$
- The maximum time allowed for the application to run, $maxT \in \mathbb{R}_{>0}$
- The required confidence level, $cl \in \mathbb{R}_{>0}$
- The required accuracy, $eps \in \mathbb{R}_{>0}$

The outputs by the function $MeanUsingTtest$ are:

- The number of experimental runs actually made, $repsOut \in \mathbb{Z}_{>0}$
- The confidence level achieved, $clOut \in \mathbb{R}_{>0}$
- The accuracy achieved, $epsOut \in \mathbb{R}_{>0}$
- The elapsed time, $etimeOut \in \mathbb{R}_{>0}$

- The mean, $mean \in \mathbb{R}_{>0}$

For each data point, the function is invoked, which repeatedly executes the application app until one of the following three conditions is satisfied:

1. The maximum number of repetitions ($maxReps$) have been exceeded (Line 3).
2. The sample mean falls in the confidence interval (or the precision of measurement eps has been achieved) (Lines 13–15).
3. The elapsed time of the repetitions of application execution has exceeded the maximum time allowed ($maxT$ in seconds) (Lines 16–18).

So, for each data point, the function $MeanUsingTtest$ is invoked and the sample mean $mean$ is returned at the end of invocation. The function $Measure$ measures the execution time or the dynamic energy consumption using the HCL's WattsUp library [38] based on the input, $TIME$ or $ENERGY$. The input minimum and maximum number of repetitions, $minReps$ and $maxReps$, differ based on the problem size solved. For small problem sizes ($32 \leq n \leq 1024$), these values are set to 10,000 and 100,000. For medium problem sizes ($1024 < n \leq 5120$), these values are set to 100 and 1000. For large problem sizes ($n > 5120$), these values are set to 5 and 50. The values of $maxT$, cl and eps are set to 3600, 0.95 and 0.025. If the precision of measurement is not achieved before the maximum number of repeats have been completed, we increase the number of repetitions and also the maximum elapsed time allowed. However, we observed that condition (2) is always satisfied before the other two in our experiments. The complexity of the function $MeanUsingTtest$ is O(N).

Algorithm 1 Function determining the sample mean using Student's *t*-test.

```
1: procedure MEANUSINGTTEST(
        $app, minReps, maxReps,$
        $maxT, cl, accuracy,$
        $repsOut, clOut, etimeOut, epsOut, mean$)
Input:
    The application to execute, $app$
    The minimum number of repetitions, $minReps \in \mathbb{Z}_{>0}$
    The maximum number of repetitions, $maxReps \in \mathbb{Z}_{>0}$
    The maximum time allowed for the application to run, $maxT \in \mathbb{R}_{>0}$
    The required confidence level, $cl \in \mathbb{R}_{>0}$
    The required accuracy, $eps \in \mathbb{R}_{>0}$
Output:
    The number of experimental runs actually made, $repsOut \in \mathbb{Z}_{>0}$
    The confidence level achieved, $clOut \in \mathbb{R}_{>0}$
    The accuracy achieved, $epsOut \in \mathbb{R}_{>0}$
    The elapsed time, $etimeOut \in \mathbb{R}_{>0}$
    The mean, $mean \in \mathbb{R}_{>0}$

2:      $reps \leftarrow 0$; $stop \leftarrow 0$; $sum \leftarrow 0$; $etime \leftarrow 0$
3:      while ($reps < maxReps$) and (!$stop$) do
4:          $st \leftarrow$ MEASURE($TIME$)
5:          EXECUTE($app$)
6:          $et \leftarrow$ MEASURE($TIME$)
7:          $reps \leftarrow reps + 1$
8:          $etime \leftarrow etime + et - st$
9:          $ObjArray[reps] \leftarrow et - st$
10:         $sum \leftarrow sum + ObjArray[reps]$
11:         if $reps > minReps$ then
12:             $clOut \leftarrow$ fabs(gsl_cdf_tdist_Pinv($cl$, $reps - 1$))
                    $\times$ gsl_stats_sd($ObjArray$, 1, $reps$)
                    / sqrt($reps$)
13:             if $clOut \times \frac{reps}{sum} < eps$ then
14:                 $stop \leftarrow 1$
15:             end if
16:             if $etime > maxT$ then
17:                 $stop \leftarrow 1$
18:             end if
19:         end if
20:     end while
21:     $repsOut \leftarrow reps$; $epsOut \leftarrow clOut \times \frac{reps}{sum}$
22:     $etimeOut \leftarrow etime$; $mean \leftarrow \frac{sum}{reps}$
23: end procedure
```

3.6 Summary

System-level power measurements are considered as the most accurate approach to measure the power consumption at node-level. In this chapter, we present a methodology to compute the energy consumption by an application reliably and accurately using the system-level measurements. Then, HCLWattsUp API [38] is presented that gathers the power-readings from the power-meter and return the energy consumption by the application during its execution within user-defined precision settings and confidence interval. The measurements are highly accurate within the accuracy provided by the power-meter used to measure the power consumption.

The modern sophisticated revenue-graded power meters such as Yokogawa WT5000 [98] offers the basic power accuracy of up to $\pm 0.03\%$ with a very high sampling rate of 10M per second. To ensure the reliability of the measurements, a detailed statistical approach and number of precautions are presented. To summarize, using the presented methodology and HCLWattsUp API , one can determine the dynamic energy consumption by the application in an accurate and reliable manner during its execution. The approach is generic and thus can be applied to any power meter connected with a computing node.

Chapter 4

A Comparative Study of Methods for Measurement of Energy of Computing

This chapter is mainly based on [37].

4.1 Introduction

Energy of computing is a serious environmental concern and mitigating it has become an important technological challenge as explained in chapter 1. Energy efficiency in computing is driven by innovations in hardware represented by the micro-architectural and chip-design advancements and software that can be grouped into two categories: (a) System-level energy optimization and (b) Application-level energy optimization. System-level optimization methods aim to maximize energy efficiency of the environment where the applications are executed using techniques such as DVFS, DPM and energy-aware scheduling.

Application-level optimization methods use application-level parameters and models to maximize the energy efficiency of the applications. Accurate measurement of energy consumption during an application execution is the key to energy minimization techniques at software level. There are three popular approaches to providing it: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors and (c) Energy predictive models.

The general issues and challenges with all three popular approaches to measure the energy of computing are discussed in sections 1.1.5 and 2.3. Briefly, the decomposition of energy consumption into the energy consumption of the components involved in executing the application is not straightforward using the system-level power measurements. On-chip integrated power sensors, on the other hand, provide fine-grain component level power consumption details. However, there are some issues with the power data values provided by these vendor-specific libraries. The fundamental issue with this measurement approach is the lack of information about how a power reading for a component is determined during the execution of an application utilizing the component. Apart from accuracy, the other issues include the lack of details on update frequency of power readings, portability, poor documentation, etc. as discussed in section 1.1.5. Therefore, a good understanding and validation of energy measurement instrumentation systems and on-chip power sensors is necessary for trusting and employing their readings in application-level energy optimization techniques.

Energy predictive models are typically trained using a large suite of diverse benchmarks and validated against a subset of the benchmark suite and some real-life applications. While the general accuracy of the models has been widely researched, their application-specific accuracy has not been studied, and therefore needs further validation. In this chapter, we present a comprehensive study comparing the accuracy of state-of-the-art integrated on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, which we consider to be the ground truth.

The rest of the chapter is organized as follows. First, we explain in section 4.2 the common terminologies that are used in this chapter. The experimental platforms, applications and methodology to ensure the reliability of our results are explained in section 4.3. The dynamic energy profiles of our testbed applications with on-chip sensors provided by RAPL and system-level power measurements provided by HCLWattsUp are compared in section 4.4. In section 4.5, the comparison of dynamic energy profiles using on-chip sensors on GPUs (NVML) and HCLWattsUp and Intel Xeon Phi sensors (MPSS) and HCLWattsUp is presented. Then, the comparison of PMC-based dynamic energy predictive models with RAPL and HCLWattsUp is presented in section 4.6. In section 4.7, we study the optimization of a parallel matrix-matrix multiplication application for dynamic energy using RAPL and HCLWattsUp, followed by the comparison of cost of measurements and the other issues with state-of-the-art approaches in section 4.8. Section 4.9 covers the lessons learned, our recommendations for the use of on-chip sensors and energy predictive models and future directions. Finally, the chapter is concluded in section 4.10.

4.2 Terminologies

In this thesis, only the dynamic energy consumption is considered instead of total energy consumption or static energy. We explain the rationale behind using dynamic energy consumption in section 3.2. It can be determined by subtracting the static energy from the total energy of the system that it consumed during the application execution, using the equation 3.1. The execution time of the application execution can be determined accurately using the processor clocks. The accuracy of the static power consumption is equal to the accuracy provided in the specification of the power meter. Hence, the accuracy of obtaining the total energy consumption employing the system-level measurements using the external power-meters is subject to the accuracy of the power-meter. The revenue-graded power meters such as Yokogawa WT5000 [98] offers the basic power accuracy of up to $\pm 0.03\%$ with a very high sampling rate of 10M per second. Hence, one can determine the dynamic energy consumption by the application within the accuracy provided by the external power-meter power readings. Therefore, we consider this approach as the ground truth for the comparative study of methods for measurement of energy of computing.

Using the system-level physical measurements provided by external power meters, the dynamic energy consumption can be determined during an application execution as explained in chapter 3. State-of-the-art on-chip power sensors (RAPL for CPUs, NVML for GPUs, MPSS for Xeon Phis) provide power measurements at a high sampling frequency that can be obtained programmatically. The dynamic energy consumption during an application execution on a compute device equipped with on-chip sensors can also be calculated using the same methodology as explained in chapter 3 using the equation 3.1.

The term ”calibration” is used throughout this chapter. we define it as a constant adjustment (positive or negative value) made to the data points in a dynamic energy profile obtained using a measurement approach (on-chip sensors or energy predictive models) with the aim to increase its accuracy or reduce its error against the ground truth (the physical measurements using power meters).

Let $E(x)_{sensors}$ represents the dynamic energy consumption by an application workload size x with on-chip sensors or predictive energy model. Let $E(x)_{hclwattsup}$ represents the dynamic energy consumption by the same application workload size x with system-level physical measurements using external power meters (HCLWattsUp). Then, the prediction error is calculated as $|(E(x)_{hclwattsup} - E(x)_{sensors})|/E(x)_{hclwattsup} \times 100$.

4.3 Experimental Setup for Comparing On-Chip Sensors and System-Level Physical Measurements Using Power Meters

Following three nodes are employed for this study: (a) HCLServer01 (Table 4.1) has an Intel Haswell multicore CPU having 24 physical cores with 64 GB main memory and integrated with two accelerators: one Nvidia K40c GPU and one Intel Xeon Phi 3120P, (b) HCLServer02 (Table 4.2) has an Intel Skylake multicore CPU consisting of 22 cores and 96 GB main memory and integrated with one Nvidia P100 GPU and (c) HCLServer03 (Table 4.3) has an Intel Skylake multicore CPU having 56 cores with 187 GB main memory. These nodes are the representative of computers used in cloud infrastructures, supercomputers and heterogeneous computing clusters.

Each node has a power meter installed between its input power sockets and the wall A/C outlets. HCLServer01 and HCLServer02 are connected with a *Watts Up Pro* power meter; HCLServer03 is connected with a *Yokogawa WT310* power meter. *Watts Up Pro* power meters are periodically calibrated using the ANSI C12.20 revenue-grade power meter, Yokogawa WT310. The calibration details are provided in appendix C.

The maximum sampling speed of *Watts Up Pro* power meters is one sample every second. The accuracy specified in the data-sheets is $\pm 3\%$. The minimum measurable power is 0.5 watts. The accuracy at 0.5 watts is ± 0.3 watts. The accuracy of Yokogawa WT310 is $\pm 0.1\%$ and the sampling rate is 100K samples per second.

Following four applications are used for this study: (a) *OpenBLAS DGEMM*, OpenBLAS library routine to compute the matrix product of two dense matrices, (b) *Intel MKL-Double-precision General Matrix Multiplication (MKL-DGEMM)*: Intel Math Kernel Library (Intel Math Kernel Library (MKL)) routine, which computes the product of two dense matrices, (c) FFTW 2D: two dimensional FFT routine to compute the discrete Fourier transform of a complex signal and (d) MKL-FFT 2D: two dimensional FFT routine provided by Intel MKL to compute the discrete Fourier transform of a complex signal. The DGEMM applications computes $C = \alpha \times A \times B + \beta \times C$, where A, B and C are matrices of size $M \times N$, $N \times N$ and $M \times N$ and α and β are constant floating-point numbers. The FFT applications compute 2D-DFT of a complex signal matrix of size $M \times N$. The Intel MKL version on the three nodes is 2017.0.2. The FFTW version used is 3.3.7. We choose applications employing matrix-matrix multiplication and fast Fourier transform routines since they are fundamental kernels

Table 4.1: HCLserver01: Specifications of the Intel Haswell multicore CPU, Nvidia K40c and Intel Xeon Phi 3120P.

Intel Haswell E5-2670V3	
Launch Date	Q3'14
No. of cores per socket	12
Socket(s)	2
CPU MHz	1200.402
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30,720 KB
Total main memory	64 GB DDR4
Memory bandwidth	68 GB/sec
Nvidia K40c	
Launch Date	Q4'13
No. of processor cores	2880
Total board memory	12 GB GDDR5
L2 cache size	1536 KB
Memory bandwidth	288 GB/sec
Intel Xeon Phi 3120P	
Launch Date	Q2'13
No. of processor cores	57
Total main memory	6 GB GDDR5
Memory bandwidth	240 GB/sec

employed in scientific applications [154].

To obtain the energy consumption provided by RAPL, a well known package Intel PCM[105] is used. To confirm that the RAPL values output by this package are correct, they are compared with values given by other well known package, PAPI [104].

To obtain the power measurements from the WattsUp Pro power meters and Yokogawa WT310, HCLWattsUp interface [38] is used. HCLWattsUp interface has no extra overhead and therefore does not impact the dynamic energy consumption by the application kernel. The interface and the methodology used to obtain a data point are explained in sections 3.3 and 3.4. We use HCLWattsUp throughout this thesis to represent the system-level power measurements for the sake of brevity.

A detailed sophisticated methodology is followed (as explained in section 3.5) to ensure reliability of our experimental results. The methodology determines a sample mean (execution time or dynamic energy or PMC) by executing the application repeatedly until the sample mean meets the statistical confidence criteria (95% confidence interval, a precision of 0.025 (2.5%)). Student's *t*-test is used to determine the sample mean. The test assumes that the individual observations are independent and their population follows the normal distribution. We use Pearson's chi-squared test to ensure that the observations follow normal distribution.

Table 4.2: HCLserver1: Specifications of the Intel Skylake multicore CPU and Nvidia P100 PCIe.

Intel Xeon Gold 6152	
Launch Date	Q3'17
Socket(s)	1
Cores per socket	22
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30,976 KB
Main memory	96 GB
Nvidia P100 PCIe	
Launch Date	Q2'16
No. of processor cores	3584
Total board memory	12 GB CoWoS HBM2
Memory bandwidth	549 GB/sec

Table 4.3: HCLServer3: Specifications of the Intel Skylake multicore processor (CPU) consisting of two sockets of 28 cores each.

Technical Specifications	**Intel Xeon Platinum 8180**
Launch Date	Q3'17
Socket(s)	2
Cores per socket	28
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	1024 KB, 39,424 KB
Main memory	187 GB

We explain the experimental results into following two sections: i) First, we present the experimental results comparing the energy profiles constructed with integrated on-chip sensors against the ground truth, then ii) the dynamic energy prediction using PMC-Based energy predictive models are compared with the ground truth.

4.4 Comparison of Energy Measurements Using RAPL and HCLWattsUp

We first present a brief on RAPL before introducing the methodology to compare the measurements of dynamic energy consumption by RAPL and HCLWattsUp.

RAPL (Running Average Power Limit) [41] provides a way to monitor and dynamically set the power limits on processor and DRAM. So, by controlling the maximum average power, it matches the expected power and cooling budget. RAPL exposes its energy counters through model-specific registers (MSRs). It updates these counters once in every 1 ms. The energy is calculated as a multiple of model specific energy units. For Sandy Bridge, the energy unit is 15.3 μ J, whereas it is 61 μ J for Haswell and Skylake. It divides a platform into four domains, which are presented below:

1. *PP0* (Core Devices): Power plane zero includes the energy consumption by all the CPU cores in the socket(s).
2. *PP1* (Uncore Devices): Power plane one includes the power consumption of integrated graphics processing unit – which is not available on server platforms– uncore components.
3. *DRAM*: Refers to the energy consumption of the main memory.
4. *Package*: Refers to the energy consumption of entire socket including core and uncore: `Package = PP0 + PP1`.

PP0 is removed in the the Haswell E5 generation [45]. For our experiments, we use *Package* and *DRAM* domains to obtain the energy consumption by CPU and DRAM when executing an application.

4.4.1 Experimental Methodology

The following workflows of the experiments is followed for comparing the RAPL and HCLWattsUp energy measurements. The workflow to determine the dynamic energy consumption by an application using RAPL follows:

1. Using Intel PCM/PAPI, the *base power* of CPUs (core and un-core) and DRAM (when the given application is not running) is obtained.
2. Using HCLWattsUp API, the *execution time* of the given application is obtained.
3. Using Intel PCM/PAPI, the *total energy* consumption of the CPUs and DRAM, during the execution of the given application is obtained.
4. Finally, the *dynamic energy* consumption (of CPUs and DRAM) by subtracting the *base energy* from *total energy* consumed during the execution of the given application is calculated.

The workflow to determine the dynamic energy consumption using HCLWattsUp follows:

1. Using HCLWattsUp API, the *base power* of the platform (when the given application is not running) is obtained.
2. Using HCLWattsUp API, the *execution time* of the application is obtained.
3. Using HCLWattsUp API, the *total energy* consumption of the server, during the execution of the given application is obtained.
4. Finally, the *dynamic energy* consumption by subtracting the *base power* from *total energy* consumed during the execution of the given application is calculated.

Any error in energy readings is either due to the time or power readings because the energy is calculated as the product of power and time. It is confirmed that the execution time of the application kernel is the same for dynamic energy calculations by both tools. So, it is important to note here that any difference between the energy readings of the tools comes solely from their power readings.

Either of the following configuration parameters is used to build an energy profiles with RAPL and HCLWattsUp: (a) Problem size ($M \times N$) where $M \leq N$, (b) Number of CPU threads, (c) Number of CPU Cores. Hence, 51 energy profiles of different application configurations are analyzed in total for aforementioned set of applications.

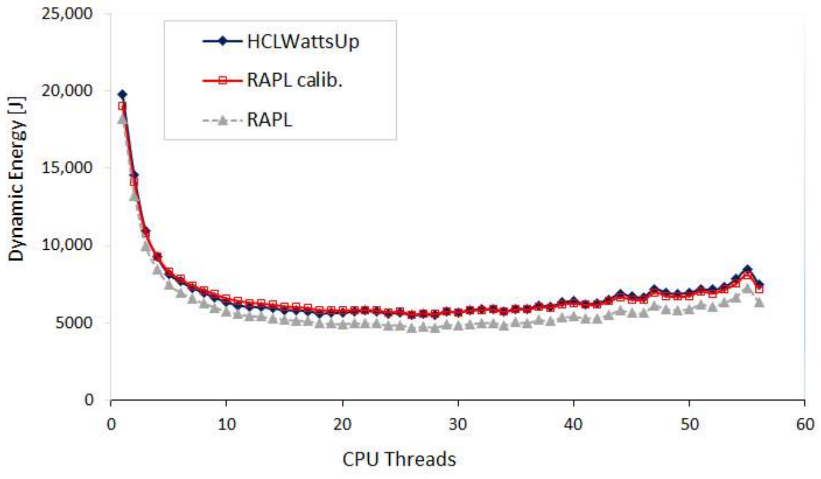
The cost in terms of number of measurements to determine the dynamic energy consumption of the application using sensors is same for both tools as we need three (*Base power, Execution Time and Total Energy*) measurements to obtain a single data point of the application dynamic energy profile.

4.4.2 Experimental Results on HCLServer03

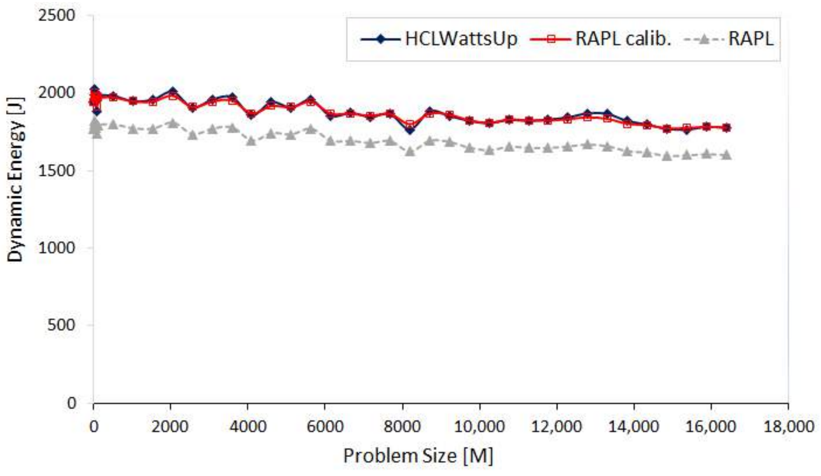
In the first set of experiments, we explore the FFTW and Intel MKL-Fast Fourier Transform (MKL-FFT) energy consumption by a given workload size N = 32,768 and N = 43,328 as a function of logical threads (1 to 112) and CPU cores (1 to 56) of CPU respectively. For next run of experiments, we study the FFTW energy consumption by two teams of 28 cores each when distributing the workload sizes range from $0 \times N/2$ to each team with a step size of 512 for the first dimension M whereas the second dimension N is fixed. We run this configuration for 18 different problem sizes N ranging from 20,480 to 21,568 with a step size of 64. In our next run of experiments, we explore the FFTW energy consumption behavior when both the teams of 28 cores has different workloads. To achieve this, we distribute the first dimension M of the problem size 32,768 $\times$ 32,768 so that we assign the workload size ranges from $0 \times N$ to $N/2 \times N$ to the first team whereas the workload size of second team ranges from N/2 to N.

We find that RAPL reports less dynamic energy consumption for all aforementioned application configurations than HCLWattsUp. RAPL profile follows the same pattern as that of HCLWattsUp for most of the data points. One can significantly reduce the average error between both the profiles by calibrating the RAPL readings. For example, one can reduce this average error from 13.05% to 2.19% for the dynamic energy profile of MKL-FFT as a function of CPU cores for the workload size N = 43,328 as illustrated in figure 4.1a. Similarly, the average error of dynamic energy profile of FFTW as a function of CPU threads for the workload size N = 32.768 can be reduced from 12.68% to 3.69% by calibrating RAPL readings as illustrated in figure 4.1b. This calibration, nevertheless, is application dependent and is also different for different configurations of the same application.

In the second set of experiments, the dynamic energy profiles of different configurations of OpenBLAS DGEMM is examined. Each application configuration is executed using G thread-groups where each group contains equal number of T threads. We study eight such configurations, (G,T) = {(2,56), (4,28), (7,16), (8,14), (14,8), (16,7), (28,4), (56,2)}. We construct the dynamic energy profile for each configuration as a function of problem size. The problem size $N \times N$ ranges from 10,240 $\times$ 10,240 to 26,112 $\times$ 26,112 with a constant step size of 512. Likewise the first set of experiments, RAPL profiles lag behind the HCLWattsUp profiles. Unlike the first set of experiments where one can reduce the error between both the profiles significantly by calibration, one can only reduce up to half



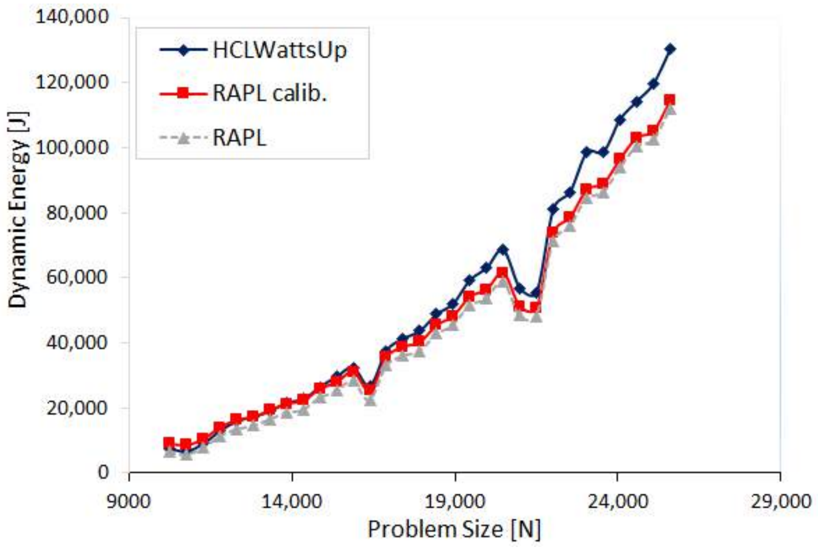
(a) MKL-FFT, N = $43,328$



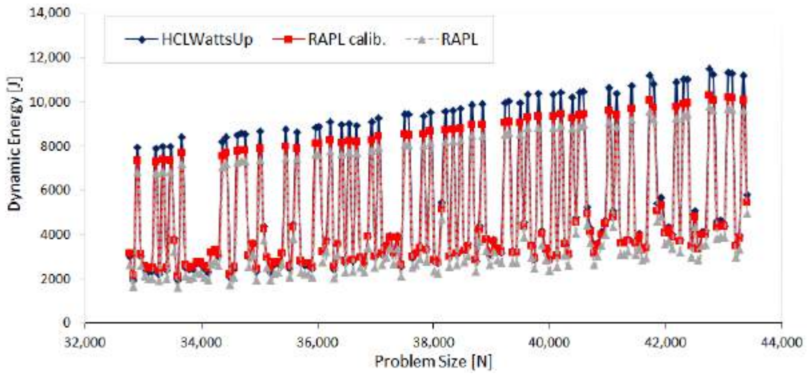
(b) FFTW, N = $32,768$, Threadgroups = 7, Threads = 16

Figure 4.1: Dynamic energy profiles with Running Average Power Limit (RAPL) and HCLWattsUp on HCLServer03, class A. RAPL calib. *means that RAPL readings have been calibrated.*

of the average error for most of the application configurations. It is important to note that likewise the first set of experiments, the calibration is not the same for all the application configurations.



(a) Threadgroups=28, Threads=4



(b) Threadgroups=28, Threads=2

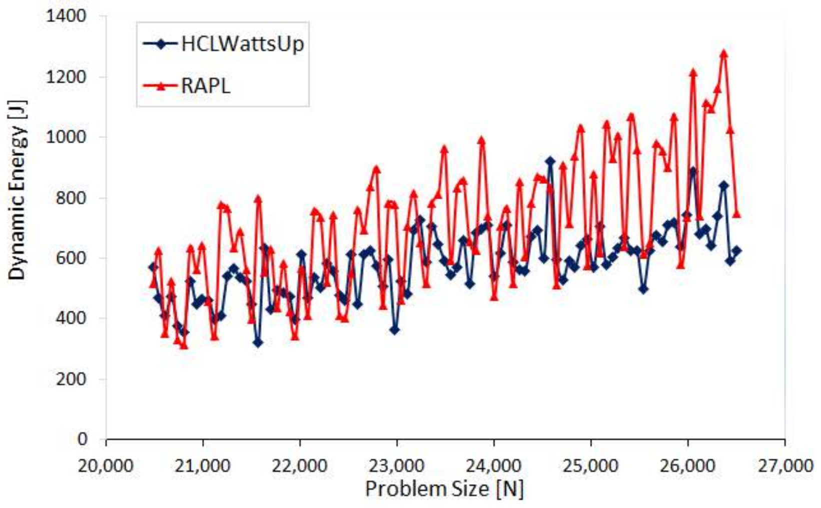
Figure 4.2: Dynamic energy profiles by RAPL and HCLWattsUp on HCLServer03, class B. RAPL calib. *means that RAPL readings have been calibrated.* (***a***) *DGEMM, N = 10,240–25,600,* (***b***) *MKL-FFT, N = 32,768–43,456.*

In our third set of experiments, the dynamic energy behavior of FFTW is examined as a function of problem size $N \times N$. We make three sets of application configurations using three different problem ranges: (i) 35,480$\times$ 41,920 , (ii) 30,720 $\times$ 34,816 and (iii) 20,480 $\times$ 26,560, all with a constant step size of 64. We group the 112 CPU threads into the teams considering following factors {112, 56, 28, 16, 14, 8, 7}. In this way, we build dynamic energy profiles for 21 different application configurations. Unlike the previous sets of experiments, we observe different behavior of RAPL profiles. For most of the application configurations, RAPL over-reports the energy consumption than HCLWattsUp. Furthermore, RAPL exhibits different trend for dynamic energy consumption than HCLWattsUp. Consider, for example, the dynamic energy profile with RAPL and HCLWattsUp of application configuration: problem size range = 20,480 $\times$ 26,560, groups = 16, number of CPU threads = 7. The average and maximum difference of RAPL with HCLWattsUp is 31% and 147%. Figure 4.3a illustrates its dynamic energy profile. One can observe that for many data points, RAPL reports an increase in dynamic energy consumption with respect to the previous data point in the profile whereas HCLWattsUp reports a decrease and vice versa. One can not therefore use calibration to reduce the average error between the profiles because of their interlacing behavior.

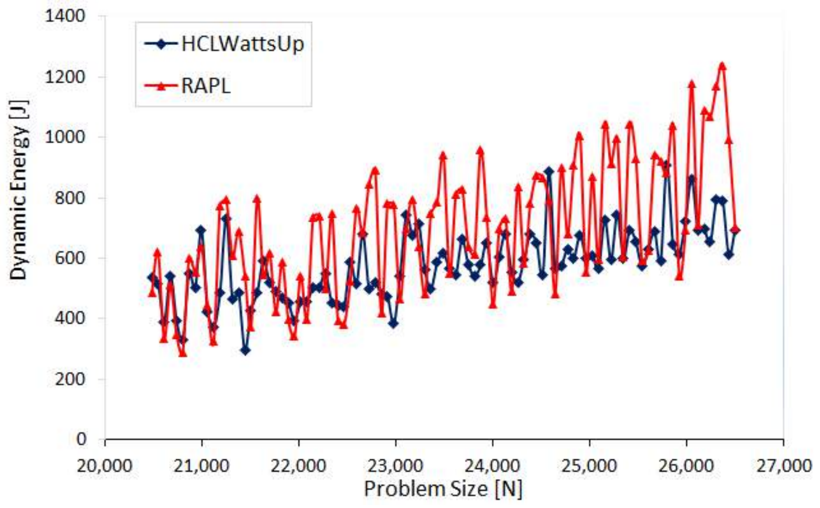
Figures 4.2b and 4.3a,b show drastic variations (energy drops and jumps) in the dynamic energy profiles for OpenBLAS DGEMM, MKL-FFT and FFTW. There are following causes behind the sudden spikes and variations in energy profiles:

- According to the Intel MKL documentation [155], Intel MKL-FFT takes less time of transform for the problem sizes which are the multiple of supported radices (such as 2, 3, 4, 5, 7, 8, 11, 13, 16 and several other larger size kernels), and 2-power sizes. It takes more time of transform for the other problem sizes. Similarly, FFTW [156] is reported to be the best at handling sizes of the form $2^a\ 3^b\ 5^c\ 7^d\ 11^e\ 13^f$ where e and f is either 0 or 1, and the other exponents are arbitrary. Other sizes are computed by means of a slow, general-purpose routine. That is why, it consumes less dynamic energy when the problem sizes is one of the supported indices due to less execution time, and more dynamic energy for the other problem sizes due to the higher execution time. Hence, it reflects into the spikes and variations in the dynamic energy profiles of the fast Fourier transform routines.
- Modern multicore platforms have many inherent complexities, which are: a) Severe resource contention due to tight integration of tens of cores contending for shared on-chip resources such as last level cache (LLC), interconnect (For example: Intel's Quick Path Interconnect, AMD's Hyper Transport), and DRAM controllers; b) Non-uniform memory access (NUMA); and c) Dynamic power management (DPM) of multiple power domains (CPU sockets, DRAM). The complexities were shown to result in complex (non-linear and non-smooth) functional relationships between performance and workload size and between dynamic energy and workload size for real-life data-parallel applications. Reference [18] highlights the performance drops and energy jumps due to the inherent complexities in modern multicore platforms.
- References [19, 145] demonstrate by executing real-life multi-threaded data-parallel applications on modern multicore CPUs that the functional relationships between performance and workload size and between energy and workload size have complex (non-linear) properties.

Tables 4.4, 4.5 and 4.6 present the statistics of prediction error between RAPL and HCLWattsUp on HCLServer03. The error reduction between the dynamic energy profiles with RAPL and HCLWattsUp after calibration is also presented.



(a) Threadgroups=16, Threads=7



(b) Threadgroups=14, Threads=8

Figure 4.3: Dynamic energy profiles of FFTW (N = $20,480-26,560$*) by RAPL and HCLWattsUp on HCLServer03, class C*

Table 4.4: Percentage error of dynamic energy consumption with RAPL and HCLWattsUp on HCLServer03. Here, G = Thread-groups, T = Threads, and SS = Step Size.

Application	Problem Size, Step-Size	Configuration Parameter	Avg Actual Error	Avg. Error after Calibration	Reduction after Calibration
FFTW	$N = 32,768$	CPU Threads (1–112)	12.68%	3.69%	70.9%
MKL-FFT	$N = 43,328$	CPU Cores (1–56)	13.05%	2.19%	83.22%
FFTW	N = 20,480–21560, SS = 512	problem size $(M \times N)$ where $0 \geq M \leq N/2$	8.15%	5.56%	31.78
FFTW	$N = 32,768$, SS = 16	Load Imbalance: problem size $(M \times N)$ where $0 \geq M \leq N/2$	10.45%	0.6%	94.26%

Table 4.5: Percentage error of dynamic energy consumption with RAPL and HCLWattsUp on HCLServer03. Here, G = Thread-groups, T = Threads, and SS = Step Size.

Application	Problem Size, Step-Size	Configuration Parameter	Avg Actual Error	Avg. Error after Calibration	Reduction after Calibration
OpenBlas DGEMM	N = 10,240–25,600, SS = 512	CPU Threads			
		G = 56, T = 2	12.84%	6.66%	48.13%
		G = 28, T = 4	13.28%	8.58%	35.39%
		G = 16, T = 7	14.02%	8.54%	39.09%
		G = 14, T = 8	13.61%	7.98%	41.37%
		G = 8, T = 14	18.59%	9.64%	48.14%
		G = 7, T = 16	19%	9.7%	48.95%

Application	Problem Size, Step-Size	Configuration Parameter	Avg Actual Error	Avg. Error after Calibration	Reduction after Calibration
		G = 4, T = 28	20.89%	10.38%	50.31%
		G = 2, T = 56	23.41%	11.21%	52.11%
MKL-FFT	N = 32,768–43,456, SS = 64	problem size			
		G = 28, T = 2	15.08%	4.91%	67.4%
		G = 14, T = 4	13.63%	4.97%	63.54%
		G = 8, T = 7	13.24%	5.25%	60.35
		G = 7, T = 8	13.21%	5.4%	59.12%
		G = 4, T = 14	13.03%	5.65%	56.64%
		G = 2, T = 28	13.02%	5.64%	56.68%
		G = 1, T = 56	14.12%	6.22%	55.95%
MKL-FFT	N = 25,600–46,080, SS = 512	problem size			
		G = 28, T = 2	14.46%	5%	65.42%
		G = 14, T = 4	13%	4.51%	65.31%
		G = 8, T = 7	12.4%	4.49%	63.79%
		G = 7, T = 8	12.34%	4.45%	63.94%
		G = 4, T = 14	11.97%	4.58%	61.74%
		G = 2, T = 28	12.35%	4.8%	61.13%
		G = 1, T = 56	13.56%	6.27%	53.76%
FFTW	N = 35,480–41,920, SS = 64	problem size			
		G = 16, T = 7	12.4%	10.35%	16.53%
		G = 14, T = 8	13.19%	11.54%	12.51%
		G = 8, T = 14	13.66%	12.73%	6.81%%
		G = 7, T = 16	14.59%	13.3%	8.84%
		G = 4, T = 28	13.73%	12.78%	6.92%
		G = 2, T = 56	12.3%	5.58%	54.63%
		G = 1, T = 112	24.62%	3.9%	84.16%

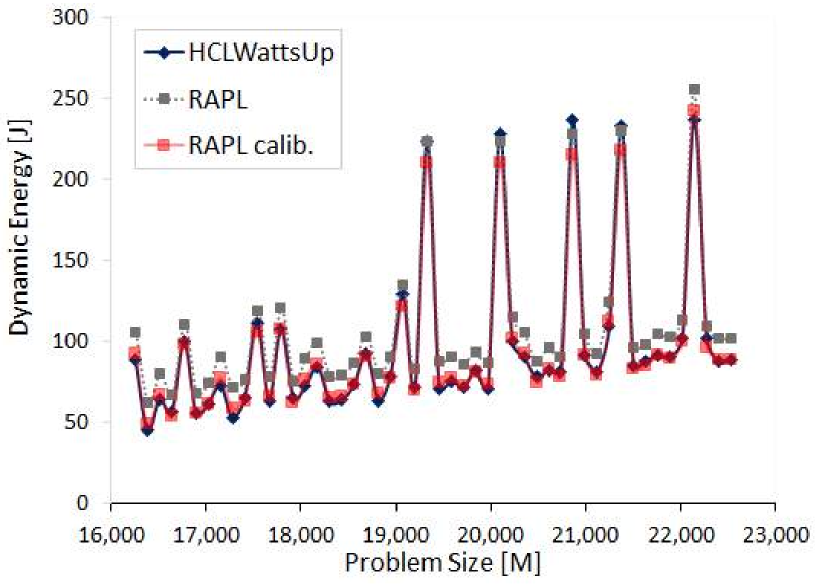
Table 4.6: Percentage error of dynamic energy consumption with RAPL and HCLWattsUp on HCLServer03. Here, G = Thread-groups, T = Threads, and SS = Step Size. '-' denotes that calibration does not be improve the difference.

Application	Problem Size, Step-Size	Configuration Parameter	Avg Actual Error	Avg. Error after Calibration	Reduction after Calibration
FFTW	N = 30,720–34,816, SS = 64	problem size			
		G = 16, T = 7	14.51%	-	-
		G = 14, T = 8	16.32%	-	-
		G = 8, T = 14	16.15%	-	-
		G = 7, T = 16	14.89%	-	-
		G = 4, T = 28	9.32%	-	-
		G = 2, T = 56	10.94%	5.34%	51.19%
		G = 1, T = 112	25.05%	10.44%	58.32%
FFTW	N = 20,480–26,560, SS = 64	problem size			
		G = 16, T = 7	31%	-	-
		G = 14, T = 8	28.16%	-	-
		G = 8, T = 14	21.59%	-	-
		G = 7, T = 16	17.76%	-	-
		G = 4, T = 28	7.6%	4.83	36.45%
		G = 2, T = 56	9.76%	6.12%	37.3%
		G = 1, T = 112	25.63%	10.22%	60.12

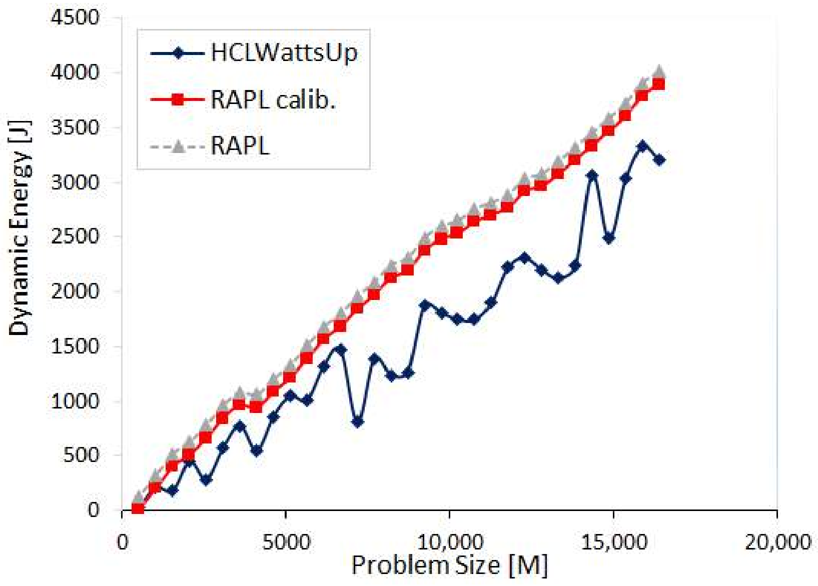
4.4.3 Experimental Results of RAPL and HCLWattsUp on HCLServer01 and HCLServer02

For our sets of experiments on HCLServer01 and HCLServer02, the problem size is used as an application configuration parameter. On HCLServer01, the workload sizes for MKL-DGEMM range from 512 $\times$ 16,384 to 16,384 $\times$ 16,384 with a step size of 512 and for 2D MKL-FFT range from 16256 $\times$ 22,528 to 22,528 $\times$ 22,528 with a step size of 128. On HCLServer02, the workload sizes for MKL-DGEMM range from 6400 $\times$ 6400 to 29,504 $\times$ 29,504 with a step size of 64. For 2D-FFT executed on HCLServer02, the workload sizes range from 22,400 × 22,400 to 41,536 × 41,536 with a step size of 64.

Figure 4.4a,b show the dynamic energy profiles of 2D MKL-FFT and MKL-DGEMM on HCLServer01 respectively. For most of the data points in dynamic energy profile of 2D MKL-FFT, RAPL over-reports the dynamic energy consumption than HCLWattsUp. There are, however, some data points where it reports otherwise. The maximum and average errors of RAPL with HCLWattsUp is 37.1% and 16.01%. One can reduce them to 9.93% and 3.48% by calibrating RAPL readings.



(a) 2D MKL-FFT

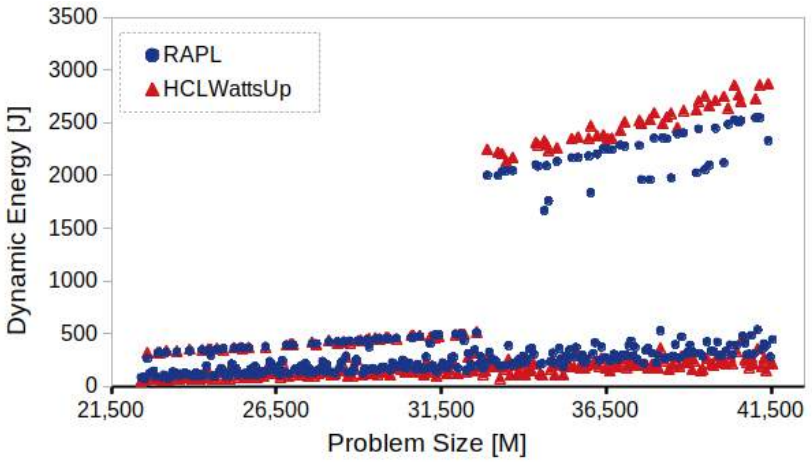


(b) MKL-DGEMM

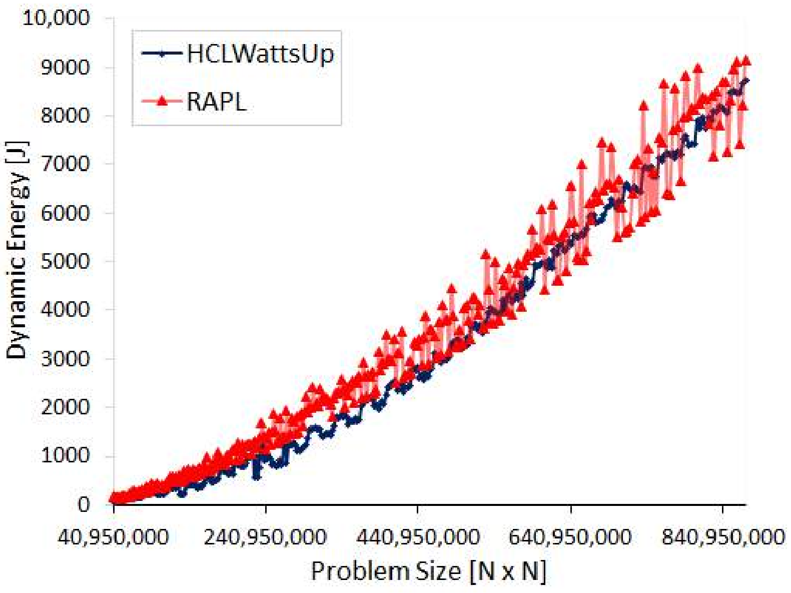
Figure 4.4: Dynamic energy consumption of RAPL, RAPL calibrated and HCLWattsUp on HCLServer01.

For MKL-DGEMM on HCLServer01, we find that RAPL readings are leading the HCLWattsUp readings. Further, both profiles do not exhibit the same pattern. One can observe many data points such as 1024 × 16,384, 2048 × 16,384, 6656 × 16,384, 14,336 × 16,384 and etc., where

HCLWattsUp suggests an increase of 14.67%, 81.53%, 55.05%, 23.2% in dynamic energy consumption whereas RAPL suggest a decrease of 38.25%, 8.05%, 19.89%, 3.76%. The maximum and average difference of RAPL with HCLWattsUp is 266.42% and 62.42%. However, one can reduce this error to 130.53% and 42.87% using calibrating RAPL readings. But, this increases the divergence between the data points where both the tools provide dynamic energy consumption values oppositely. Consider, for example, the data point 1024 $\times$ 16,384. RAPL, after calibrating, suggests a decrease of 49.12% which was 38.25% in the absence of calibration.



(a) 2D MKL-FFT



(b) MKL-DGEMM

Figure 4.5: Dynamic energy consumption of RAPL and HCLWattsUp on HCLServer02.

Figure 4.5a,b show the dynamic energy profiles of 2D MKL-FFT and MKL-DGEMM on HCLServer02 respectively. For most of the data points of MKL-DGEMM profile, RAPL suggests a decrease in dynamic energy consumption whereas HCLWattsUp reports the otherwise and vice versa. Consider, for example, the problem sizes 43454464, 125440000, 228130816, 270536704 and others where RAPL suggests a decrease of 11.92%, 15.29%, 8.68%, 27.6% whereas HCLWattsUp suggests an increase of 41.11%, 30.59%, 70.24%, 37.94%; and the problem sizes, for example, 42614784, 170459136, 249892864, 268435456 and others where RAPL suggests an increase of 14.09%, 17.3%, 20.92%, 38.99% whereas HCLWattsUp suggests a decrease of 29.67%, 19.51%, 11.83%, 28.75%. The maximum and average difference between both profiles is 205% and 36.13%.

We also find many such data points in MKL-FFT profile where both the tools reports the dynamic energy consumption oppositely. Consider, for example, the problem sizes 916393984, 1167998976, 1425817600, 1450086400 where RAPL suggests a decrease of 12.39%, 31.33%, 18.77%, 5.4% whereas HCLWattsUp reports an increase of 11.68%, 35.84%, 6.46%, 31.25%; and the the problem sizes such as 507510784, 800210944, 1150566400, 1099055104 and others where RAPL suggests an increase of 1.26%, 19.9%, 40.87%, 4.74% whereas HCLWattsUp suggests a decrease of 22.24%, 4.79%, 9.54%, 6.02%, 28.75%. The maximum and average difference between both profiles is 156.38% and 28.67%.

Table 4.7 presents the prediction errors of RAPL against HCLWattsUp on HCLServer01 and HCLServer02. We also present the percentage of error reduction between the dynamic energy profiles with RAPL and power meter after using calibration.

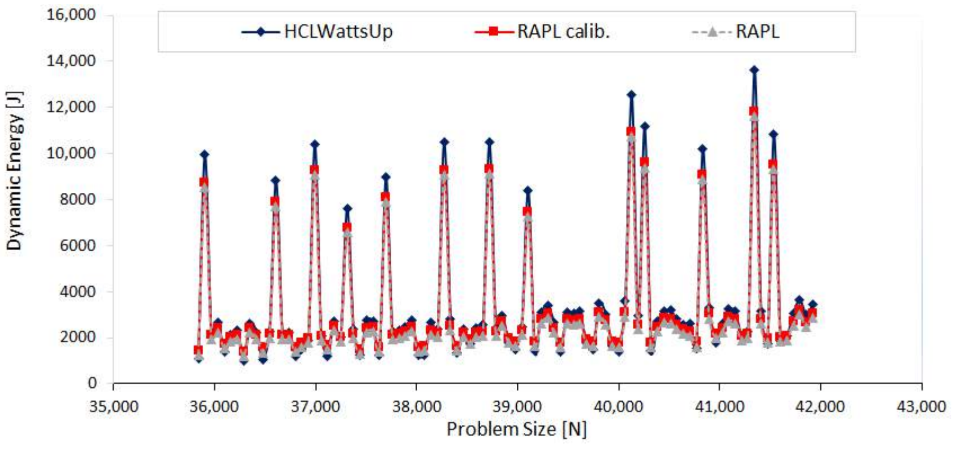
Table 4.7: Percentage error of dynamic energy consumption with RAPL and HCLWattsUp on HCLServer01 and HCLServer02. '-' denotes that calibration does not improve the difference. 01 and 02 denotes HCLServer01 and HCLServer02 respectively.

Application	Platform	Average	Maximum	Minimum	Avg after Calibration	Reduction after Calibration
FFT	01	16.01%	37.1%	0.01%	3.48%	78.26%
DGEMM	01	62.42%	266.42%	12.54%	42.86%	31.34%
FFT	02	28.67%	156.38%	0.03%	-	-
DGEMM	02	36.13%	205%	0.39%	-	-

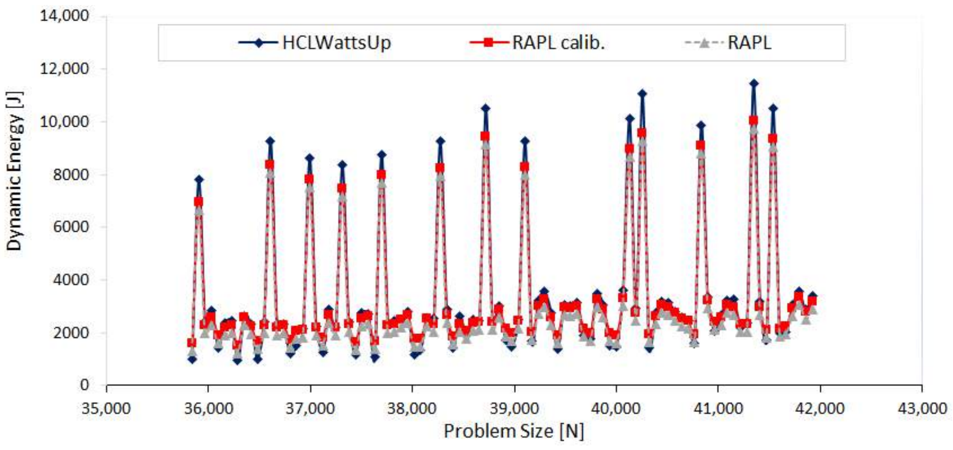
4.4.4 Discussion

In summary, we use a diverse set of application configurations to study their energy consumption behavior, and compare the results of RAPL with different power meters on three different Intel architectures. The applications can be classified into following three broad categories with respect to RAPL:

- Class A: RAPL follows most of the energy consumption pattern of the power meter. One can reduce more than 75% difference between RAPL and power meter readings after calibration. Figures 4.1a,b and 4.4a are the examples representing this class.
- Class B: RAPL does not follow most of the energy consumption pattern of the power meter. The difference between both profiles can be reduced to some extent using calibration. Figures 4.2a,b, 4.6a,b and 4.4b represent this class.
- Class C: RAPL does not follow the energy consumption pattern as of exhibited by the power meter and therefore can not be calibrated. Figures 4.3a,b and 4.5b are the examples representing this class.



(a) Threadgroups = 4, Threads = 28



(b) Threadgroups = 7, Threads = 16

*Figure 4.6: Dynamic energy profiles by RAPL and HCLWattsUp on HCLServer03 falling into Class B. (**a**) FFTW, N = 35,480–41,920, (**b**) FFTW, N = 35,480–41,920.*

Some other important findings are that the calibration is not fixed for an architecture; is not application independent; and is specific to an application configuration. For example, some application configurations can be calibrated using positive offset whereas the other configurations of the same application executing on the same platform needs negative offset for calibration.

4.5 Comparison of Energy Measurements Using GPU and Xeon Phi Sensors with HCLWattsUp

We present in this section a comparative study of energy consumption measurements by on-chip sensors for Nvidia GPUs and Intel Xeon Phi processors and HCLWattsUp. We use two applications, matrix multiplication (DGEMM) and 2D-FFT, for the study. To obtain the dynamic energy consumption of application executing on a GPU, we follow the same methodology as explained in section 4.3.

The experiments are run on two different Nvidia GPUs (K40c on HCLServer01, P100 PCIe on HCLServer02) and one Intel Xeon Phi 3120P (on HCLServer01). The DGEMM application computes the matrix product of two dense matrices *A* and *B* of sizes $M \times N$ and $N \times N$ where $M <= N$. ZZGEMMOOC out-of-card package [157] is used to compute DGEMM on Nvidia GPU K40c and CUBLAS DGEMM for Nvidia P100. The ZZGEMMOOC package reuse CUBLAS for in-card DGEMM calls. Intel MKL FFT for Xeon Phis and CUFFT for Nvidia GPUs are used to compute 2D Discrete Fourier Transform of a complex signal matrix of size $M \times N$ where $M <= N$. The Intel MKL version on both nodes is 2017.0.2, and CUDA versions on HCLServer01 is 7.5 and on HCLServer02 is 9.2.148.

Nvidia NVML [43] is used to acquire the power values from on-chip sensors on Nvidia GPUs. Intel SMC [42] is used to obtain the power values from Intel Xeon Phi that can be programmatically obtained using Intel MPSS [102]. The steps (methodology) taken to compare the measurements using GPU and Xeon Phi sensors and HCLWattsUp are similar to those for RAPL (section 4.4.1) and presented in following section.

4.5.1 Methodology To Compare Measurements Using Sensors and HCLWattsUp

To analyze the dynamic energy consumption by a given component when running an application, we need to build application profiles on them. Execution of an application using GPU/Xeon Phi involves the CPU host-core, DRAM and PCIe to copy the data between CPU host-core and GPU/Intel Xeon Phi. HCLWattsUp API provides the dynamic energy consumption of the application instead of the component. It, therefore, contains the contributions by other components including CPU host-core and DRAM. Built-in sensors, on the other hand, provide the power consumption of accelerator (GPU or Xeon Phi) only (we offload the applications to run on Intel Xeon Phi. So, they does not include the CPU host core, DRAM and PCIe to copy and migrate the data between CPU host core and the accelerator). Therefore, to compare the both methodologies in a most fair equitable way and to obtain the dynamic energy profiles of applications, we use RAPL as an aide to sensors to determine the energy contribution of CPU and DRAM.

Now, we present the work-flow of experiments that we follow to determine the dynamic energy consumption of the application. To obtain the CPU host-core and DRAM contribution in dynamic energy consumption of the application, we use RAPL in following way:

1. Using Intel PCM/PAPI, we obtain the *base power* of CPU and DRAM (when the given application is not running).
2. Using HCLWattsUp API, we obtain the *execution time* of the given application.
3. Using Intel PCM/PAPI, we obtain the *total energy* consumption of the CPU host-core (because all other cores are idle) and DRAM, during the execution of the given application.
4. Finally, we calculate the *dynamic energy* consumption (of CPU and DRAM) by subtracting the *base energy* from *total energy* consumed during the execution of the given application.

To obtain the GPU/Xeon Phi contribution, we use NVML/Intel SMC in following way:

1. Using NVML/Intel SMC, we obtain the *base power* of GPU/Xeon Phi (when the given application is not running).
2. Using HCLWattsUp API, we obtain the *execution time* of the given application.
3. Using NVML/Intel SMC, we obtain the *total energy* consumption of GPU/Xeon Phi during the execution of the given application.
4. Finally, we calculate the *dynamic energy* consumption GPU/Xeon Phi by subtracting the *base energy* from *total energy* consumed during the execution of the given application.

Now, we present the workflow of the experiments to determine the dynamic energy consumption by the given application kernel, using HCLWattsUp:

1. Using HCLWattsUp API, we obtain the *base power* of the platform (when the given application is not running).
2. Using HCLWattsUp API, we obtain the *execution time* of the application.
3. Using HCLWattsUp API, we obtain the *total energy* consumption of the platform, during the execution of the given application.
4. Finally, we calculate the *dynamic energy* consumption by subtracting the *base power* from *total energy* consumed during the execution of the given application.

Important to note here, the execution time of the application kernel is the same for dynamic energy calculations by all tools. So, any difference between the energy readings using these tools comes solely from their power readings. The cost in terms of number of measurements to determine a single data point of an application dynamic energy profile using sensors is higher than using HCLWattsUp API. Because we need at least five (*Base power with RAPL, Total Energy with RAPL, Base power with NVML/Intel SMC, Total Energy with NVML/Intel SMC, and Execution Time*) measurements to obtain the data point of the given application dynamic energy profile using RAPL and GPU/Xeon Phi internal sensors. However, we just need three measurements to obtain it using HCLWattsUp.

4.5.2 Experimental Results Using GPU Sensors (NVML)

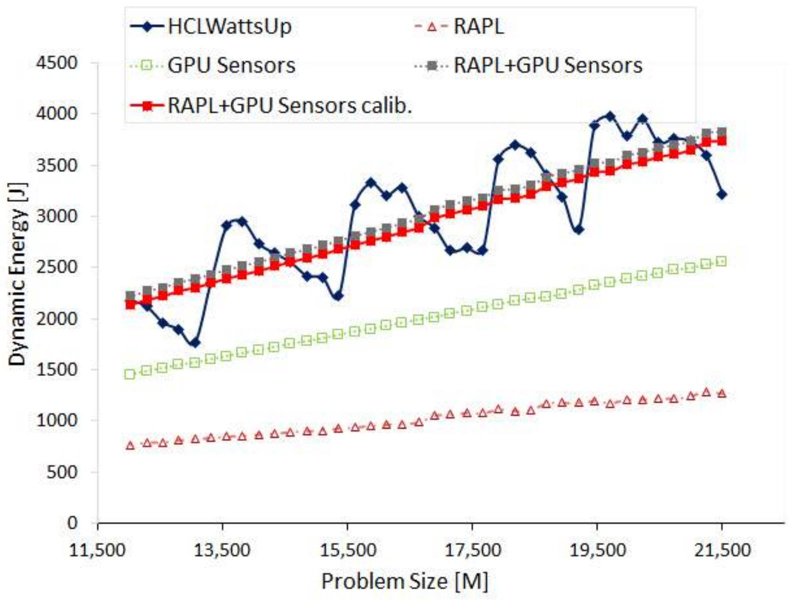
On HCLServer01, DGEMM is executed using the workload sizes ranging from 12,032 $\times$ 21,504 to 21,504 $\times$ 21,504 with a constant step size of 256. Figure 4.7a illustrates the dynamic energy profiles of DGEMM using HCLWattsUp and sensors (RAPL and NVML). The energy readings from the sensors exhibit a linear profile whereas HCLWattsUp does not. We find that sensors do not follow the application behavior exhibited by HCLWattsUp for 67.58% of the data points. Consider, for example, the data points ($M \times N$): 20,480 $\times$ 21,504, 19,200 $\times$ 21,504 and 16,640 $\times$ 21,504 where HCLWattsUp demonstrates a decrease of 6.11%, 11.09% and 9.26%, whereas sensors exhibit an increase of 1.33%, 1.07% and 1.46% respectively. We find the maximum and average errors to be 35.32% and 10.62%. We can reduce marginally the maximum and average errors using calibration to 30.5% and 10.44% respectively but the minimum error increases from 0.08% to 0.19%.

One can observe in Figure 4.7a that the combined dynamic energy profiles with RAPL and NVML follow the same trend for 90.6% of the data points. This can mislead to assume that the difference of sensors with HCLWattsUp comes from both RAPL and NVML together. But, we find that dynamic energy profile with RAPL exhibits opposite behavior to NVML for 28.12% of those data points whereas combined profile with RAPL and NVML exhibits different trends with HCLWattsUp. Consider, for example, the data points ($M \times N$): 21,504 $\times$ 21,504, 19712 $\times$ 21,504, 18,176 $\times$ 21,504. RAPL suggests a decrease of 1.81%, 1.93% and 1.56% respectively in comparison with previous data points in dynamic energy profile whereas NVML suggest an increase of 1.25%, 1.34% and 1.48% respectively for these data points. But, the combined profile follows the trend of NVML and we observe an overall increase in combined dynamic energy profile. This is because NVML has the higher contribution in dynamic energy consumption than RAPL and therefore drives the overall trend. Hence, the difference between dynamic energy profiles with RAPL and NVML and HCLWattsUp is mainly due to NVML.

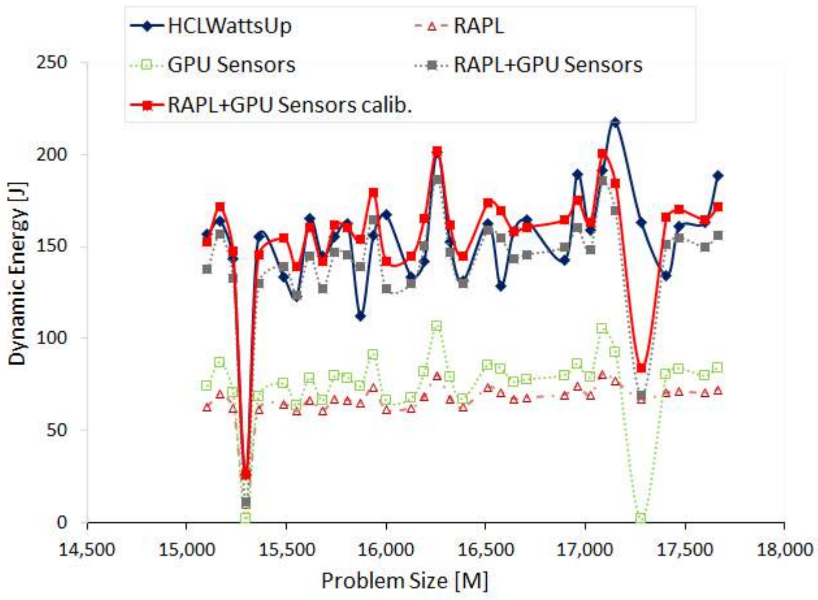
Figure 4.7b shows the dynamic energy profiles of 2D FFT on HCLServer01 with NVML and HCLWattsUp. Measurements by NVML follow the same trend for 71.88% of the data points. Consider, for example, the data points ($M \times N$): 15,360 $\times$ 23,552, 16,000 $\times$ 23,552, 16,704 $\times$ 23,552 and 17,280 $\times$ 23,552 where HCLWattsUp exhibits a decrease of 22.08%, 33.78%, 21.66% and 29.14% but NVML displays an increase of 9.24%, 2.86%, 4.04% and 81.77%. Similarly, HCLWattsUp shows an increase of 7.17%, 11.5%, 29.83% and 25.7% for the data points ($M \times N$): 15744 $\times$ 23,552, 15,936 $\times$ 23,552, 16,576 $\times$ 23,552 and 17,088 $\times$ 23,552. However, NVML exhibit a decrease of 1.48%, 37.47%, 10.91% and 16.27% for them.

We find an average and maximum error of NVML with HCLWattsUp is 12.45% and 57.77% respectively. One can slightly reduce the average error using the calibration to 10.87%. But, it increases the maximum error up to 94.55%.

We find that RAPL and NVML both exhibit the same trend for FFT. Therefore, the difference with HCLWattsUp come from both sensors collectively. Table 4.8 illustrates the errors using on-chip sensors with and without using calibration.



(a) DGEMM



(b) CUDA FFT

Figure 4.7: Dynamic energy consumption profiles of DGEMM and CUDA FFT on Nvidia K40c GPU on HCLServer01. RAPL+GPUSensors calib. *means that RAPL+GPUSensors values have been calibrated.*

Table 4.8: Percentage error of dynamic energy consumption by Nvidia K40c GPU with and without calibration and HCLWattsUp on HCLServer01.

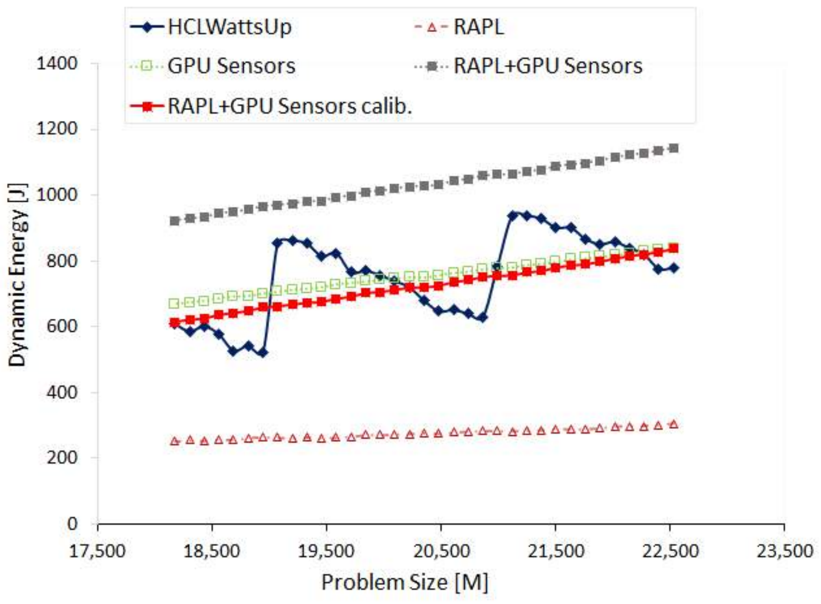
Without Calibration			
application	**Minimum**	**Maximum**	**Average**
DGEMM	0.076%	35.32%	10.62%
FFT	0.52%	57.77%	12.45%
With Calibration			
application	**Minimum**	**Maximum**	**Average**
DGEMM	0.19%	30.50%	10.43%
FFT	0.18%	94.55%	10.87%

On Nvidia P100 GPU on HCLServer02, the dynamic energy profile of 2D FFT is built as a function of problem size $M \times N$ ranging from $21,504 \times 25,600$ to $25,600 \times 25,600$ with a constant step size of 64, using sensors and HCLWattsUp. Figure 4.8b illustrates the dynamic energy profiles of 2D FFT using sensors and HCLWattsUp. We find that sensors follow the trend of HCLWattsUp for 57.14% of the data points. Consider, for example, the data points $22,016 \times 25,600$, $22,080 \times 25,600$ and $23,360 \times 25,600$ where HCLWattsUp suggests an increase of 33.53%, 15.4% and 18.49% whereas sensors suggest an increase of 2.32%, 3.21% and 6.54% respectively. The maximum and average errors are 175.97% and 73.34%. We can reduce them using calibration to 51.24% and 16.95% respectively.

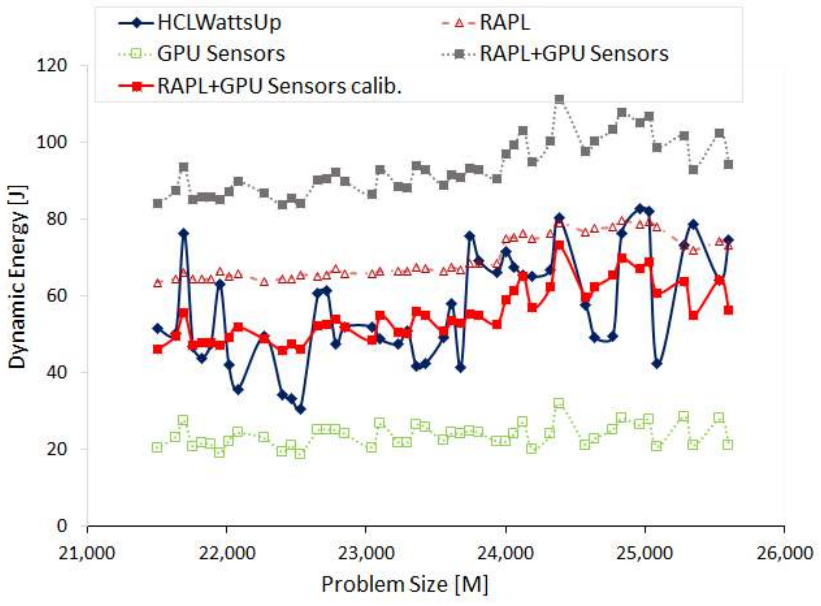
RAPL and GPU sensors follow the same trend for 88.89% of the data points. It reflects that the difference with HCLWattsUp comes from both together. But, the combined profile follows the GPU sensors trend and diverges with HCLWattsUp for 51.11% of the data points. Hence, the difference between (RAPL and GPU) sensors and HCLWattsUp is mainly from GPU sensors because they are driving the combined profile. Consider, for example, the data point $25,280 \times 25,600$. RAPL suggests an increase of 6.42% in dynamic energy consumption with respect to the previous data point. However, GPU sensors suggest a decrease of 38.12% for it and we find a decrease of 2.91% in combined profile of sensors.

Another observation is that RAPL reports more dynamic energy than GPU sensors. It suggests that for this application configuration, data transfer between CPU host-core, DRAM and GPU consumes more dynamic energy than the computation on P100 GPU.

On HCLServer02, DGEMM is executed using workload sizes ranging from 18,176 $\times$ 22,528 to 22,528 $\times$ 22,528 with a constant step size of 128. Figure 4.8a illustrates the dynamic energy profiles of DGEMM using HCLWattsUp and sensors (RAPL and GPU sensors). Likewise K40c GPU on HCLServer01, the combined energy profile of DGEMM with (RAPL and NVML) sensors exhibit a linear profile whereas HCLWattsUp exhibit differently. We find that combined sensors do not follow the application behavior exhibited by HCLWattsUp for 64.71% of the data points. Consider, for example, the data points ($M \times N$): 218,560 $\times$ 22,528, 18,944 $\times$ 22,528 and 22,400 $\times$ 22,528 where HCLWattsUp demonstrate a decrease of 4.14%, 3.8% and 5.32% whereas sensors exhibit an



(a) CUDA DGEMM



(b) CUDA FFT

Figure 4.8: Dynamic energy consumption profiles of Nvidia P100 PCIe GPU on HCLServer02.

increase of 1.23%, 1.06% and 0.59% respectively. The maximum and average errors are 84.84% and 40.06%. They can be reduced using calibration to 26.07% and 11.62% respectively.

Table 4.9: Percentage error of dynamic energy consumption by Nvidia P100 PCIe GPU with and without calibration and HCLWattsUp on HCLServer02.

Without Calibration			
Application	**Minimum**	**Maximum**	**Average**
DGEMM	13.11%	84.84%	40.06%
FFT	17.91%	175.97%	73.34%
With Calibration			
Application	**Minimum**	**Maximum**	**Average**
DGEMM	0.07%	26.07%	11.62%
FFT	0.025%	51.24%	16.95%

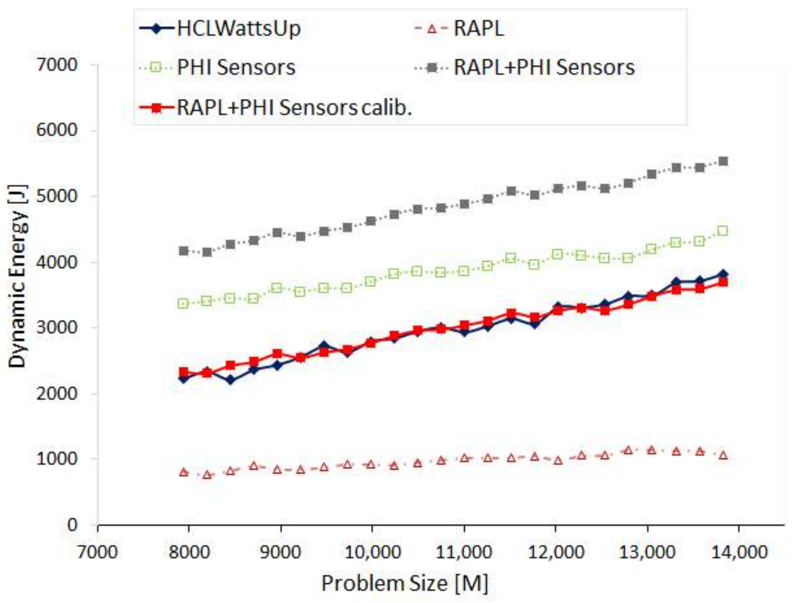
4.5.3 Experimental Results Using Intel Xeon Phi Sensors (Intel MPSS)

The dynamic energy profile of DGEMM is constructed on Intel Xeon Phi using the Intel MKL-DGEMM routine. The profile is a discrete function of problem sizes ranging from 7936 $\times$ 13,824 to 13,824 $\times$ 13,824 with a constant step size of 256. Figure 4.9a illustrates the dynamic energy profiles with sensors (RAPL and MPSS) and HCLWattsUp. We find that sensors follow the trend exhibited by HCLWattsUp for 73.91% of the data points. However, sensors reports higher dynamic energy than HCLWattsUp. The average and maximum error of sensors with HCLWattsUp is 64.5% and 93.06%. But one can reduce this error significantly using calibration to 2.75% and 93.06%.

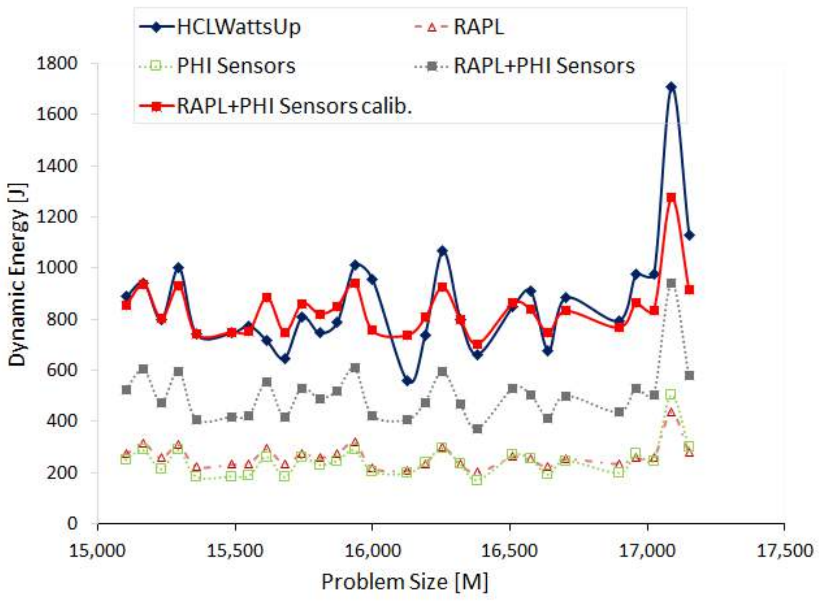
MPSS shows a trend opposite to HCLWattsUp for 26.09% of the data points. Consider, for example, the data point $9216 \times 13,824$ where MPSS exhibits a decrease of 1.24% whereas HCLWattsUp shows an increase of 4.88%. Similarly, MPSS suggests an increase of 1.1% for the data point 9728 $\times$ 13,824 whereas HCLWattsUp suggest a decrease of 3.2%.

RAPL do not follow the trend of MPSS for 53.85% of the data points. We do not, however, find this effect reflected in the profile of the combined sensors. Consider, for example, the data point 12,032 $\times$ 13,824 where RAPL suggests a decrease of 3.98% in dynamic energy consumption with respect to previous data point in the profile whereas MPSS suggests an increase of 6.1% and we find an increase of 1.86% in the combined profile. This is because of the fact that MPSS has higher contribution in combined dynamic energy profile and thus drives the overall trend. Hence, the difference between dynamic energy profiles with MPSS and HCLWattsUp comes mainly from MPSS.

Intel MKL FFT is used to compute the discrete Fourier transform of 2D signal of complex data type and build the dynamic energy profiles with HCLWattsUp and sensors (RAPL and MPSS) as a function of problem size ($M \times N$) ranges from 15,104 $\times$ 23,552 to 17,152 $\times$ 23,552 with a constant step size of 64. Figure 4.9b illustrates the dynamic energy profiles with sensors and HCLWattsUp. We find that sensors follow the trend of HCLWattsUp for 92.59% of the data points. However, sensors



(a) MKL DGEMM



(b) MKL FFT

Figure 4.9: Dynamic energy consumption profiles of Intel MKL DGEMM and Intel MKL FFT on Xeon Phi co-processor. RAPL+PHISensors calib. *means that RAPL+PHISensors values have been calibrated.*

behave oppositely with HCLWattsUp for the data points such as 15,616 $\times$ 23,552 where they display an increase of 30.95% with respect to previous data point whereas HCLWattsUp exhibits a decrease of 7.55%. Similarly, sensors show a decrease of 4.75% for data point 16,576 $\times$ 23,552 with respect to previous data point whereas HCLWattsUp exhibits an increase of 6.98%.

We find RAPL and MPSS exhibit the same trend of dynamic energy consumption. Hence, the difference between the dynamic energy profiles with combined sensors and HCLWattsUp comes from both sensors collectively. However, given the fact that MPSS has higher contribution in comparison with RAPL, therefore, MPSS drives the trend and influences the overall dynamic energy consumption reported by combined sensors. We also observe that RAPL and MPSS consume almost same amount of dynamic energy for running FFT. It shows that data transfer between the CPU host core, DRAM and Xeon Phi consumes almost as much dynamic energy as it takes to compute the given FFT plan.

Overall, MPSS reports less dynamic energy consumption than HCLWattsUp. The average and maximum error are 40.68% and 55.78% respectively. We can reduce them significantly to 9.58% and 32.3% by using calibration. Table 4.10 illustrates the statistics for dynamic energy profiles of DGEMM and FFT on Xeon Phi with and without using calibration.

Table 4.10: Percentage error of dynamic energy consumption with and without calibration and HCLWattsUp on Intel Xeon Phi.

Without Calibration			
Application	**Minimum**	**Maximum**	**Average**
DGEMM	45.1%	93.06%	64.5%
FFT	22.58%	55.78%	40.68%
With Calibration			
Application	**Minimum**	**Maximum**	**Average**
DGEMM	0.06%	9.54%	2.75%
FFT	0.06%	32.3%	9.58%

4.5.4 Discussion

We observe that the average error between measurements using sensors and HCLWattsUp can be reduced using calibration in some cases. However, the calibration value is specific to an application configuration, and different for the energy consumption of different application configurations computed with the same sensors. Another important finding is that CPU host-core and DRAM consume equal or more dynamic energy than the accelerator for FFT applications (FFTW 2D and MKL FFT 2D). We find that the data transfers (between CPU host-core and an accelerator) consume same amount of energy as consumed by the computations on the accelerators for older generations of Nvidia Tesla GPUs such as K40c and Intel Xeon Phi such as 3120P. However, for newer generations of Nvidia Tesla GPUs such as P100, the data transfers consume more dynamic energy than computations. It suggests that optimizing the data transfers for dynamic energy consumption is important.

4.6 Comparison of dynamic energy consumption using PMC-based energy predictive models and HCLWattsUp

This section presents a short summary of the comparison of the prediction accuracy of linear energy predictive models employing performance monitoring counters (PMCs) as predictor variables, with the ground truth. The experiments and findings are mainly done by my colleague Arsalan Shahid. An overview of the findings is presented as an complement to the comparative study of the accuracy of methods for measurements of computing. The details can be found in [37], and also presented in appendix A for the convenience of reader.

The overview is categorized into two classes: a). Class A contains platform-level linear regression models, and b). Class B contains application-specific models. We use a diverse application suite containing highly optimized compute-bound and memory-bound scientific routines such as DGEMM and FFT from Intel Math Kernel Library (MKL), Intel HPCG, benchmarks from NASA Application Suite (NAS), stress, naive matrix-vector multiplication, and naive matrix-matrix multiplication. We present the details on our application suit and experiment setup in appendix A.

For class A, we build a data-set of 277 points using different problem sizes as application configuration parameters. Each point represents the dynamic energy consumption and the PMC counts of one application. We used 227 points for training the models and 50 points to test the accuracy of the models. We build 6 linear models {A, B, C, D, E, F} using regression analysis. Model A is based on all the selected PMCs as predictor variables. Model B is based on five best PMCs with the PMC with the least positive correlation with dynamic energy is removed. Model C uses four PMCs with two least positively correlated PMCs removed and so on until Model F, which contains just one most positively correlated PMC. We find that the average error of the platform-specific energy predictive models and the ground truth ranges from 14% to 32% and the maximum reaches up to 100%.

For class B, we choose one compute-bound (MKL-DGEMM) and one memory bound (MKL-FFT) application. We choose six PMCs that have been employed as predictor variables in state-of-the-art energy predictive models (Section 2.2.4). We build a dataset containing 362 and 330 points representing different configurations of DGEMM and FFT for a range of problem sizes from 6400×6400 to 29504×29504 and 22400×22400 to 41536×41536, with a constant step size of 64. We use 271 and 255 points of DGEMM and FFT, to train the energy predictive models, and 91 and 75 points for testing the accuracy of models. We build two linear models for both applications: i). Model MM, and ii). Model FT. We find that the average and maximum errors for DGEMM using *Model MM* against the ground truth are 26% and 218%. In the case of FFT, the average and maximum errors using *Model FT* against the ground truth are 27% and 147%.

4.7 Energy Losses From Employing an Inaccurate Measurement Tool

In this section, we demonstrate that using inaccurate energy measuring tools in energy optimization methods may lead to significant energy losses. We study the optimization of a parallel matrix-matrix multiplication application for dynamic energy using two measurement tools, RAPL and system-level physical measurements using HCLWattsUp (which is considered as the ground truth).

A parallel application DGEMM is run which uses IntelMKL routine to compute the dot product of two dense matrices A and B of sizes $N \times N$ on two Intel multi-core processors: HCLServer01 and HCLServer02. The matrix A is partitioned on both aforementioned processors into A_1 and A_2. So that the product of matrices B and A_1 of size $M \times N$ is computed by HCLServer01 and HCLServer02 computes the product of matrices B and A_2 of size $K \times N$. There is no communication involve in these experiments.

The decomposition of the matrix A is computed using a profile-based data partitioning algorithm. The inputs to the algorithm are the number of rows of the matrix A, N and the dynamic energy consumption functions of the processors, $\{E_1, E_2\}$. The output is the partitioning of the rows, (M, K). The discrete dynamic energy consumption function of processor P_i is given by $E_i = \{e_i(x_1, y_1), ..., e_i(x_m, y_m)\}$ where $e_i(x, y)$ represents the dynamic energy consumption during the matrix multiplication of two matrices of sizes $x \times y$ and $y \times y$ by the processor i. The dimension y ranges from $14,336$ to $16,384$ in steps of 512. For HCLserver01, the dimension x ranges from 512 to $y/2$ in increments of 512. For HCLserver01, the dimension x ranges from $y - 512$ to $y/2$ in decrements of 512.

Figures 4.10a, 4.10b,4.10c and 4.10d illustrate the dynamic energy profiles for workload sizes (N):{14,336, 14,848, 15,360, 16,384} using RAPL and HCLWattsUp, respectively. To ensure the reliability of our experiments, the same strict methodology is followed as explained in sections 3.5.

One can observe that RAPL over-reports dynamic energy consumption than HCLWattsUp for each configuration. Table 4.11 provides the error of RAPL against HCLWattsUp.

Table 4.11: Prediction errors of RAPL against HCLWattsUp for dynamic energy consumption by DGEMM.

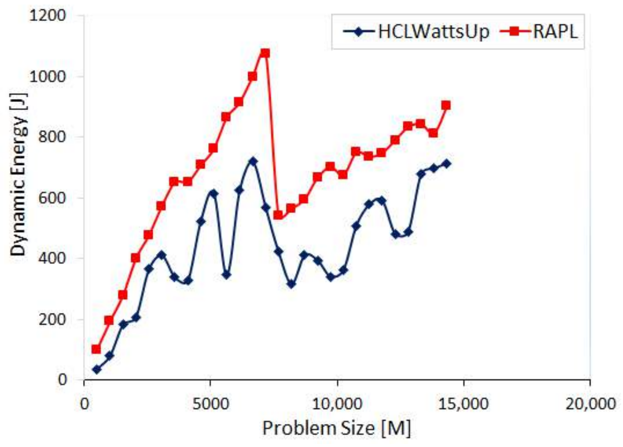
Problem Size (N)	**Minimum**	**Maximum**	**Average**
14,336	17%	172%	65%
14,848	12%	153%	58%
15,360	13%	240%	56%
16,384	2%	300%	56%

The workload partitioning algorithm takes the dynamic energy functional model as an input and finds the optimal workload configuration which optimizes the total dynamic energy consumption for the given application using load imbalance technique. The main steps of the workload partitioning algorithm are as follows:

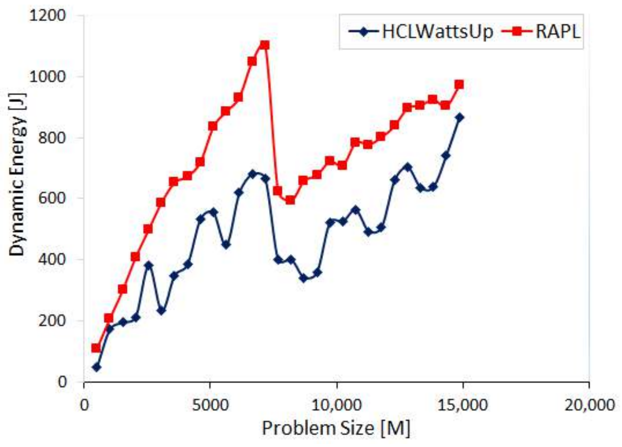
1. Plane intersection of dynamic energy functions: Dynamic energy consumption functions $\{E_1, E_2\}$ are cut by the plane $y = N$ producing two curves that represent the dynamic energy consumption functions against x given y is equal to N.

2. Determine M and K:

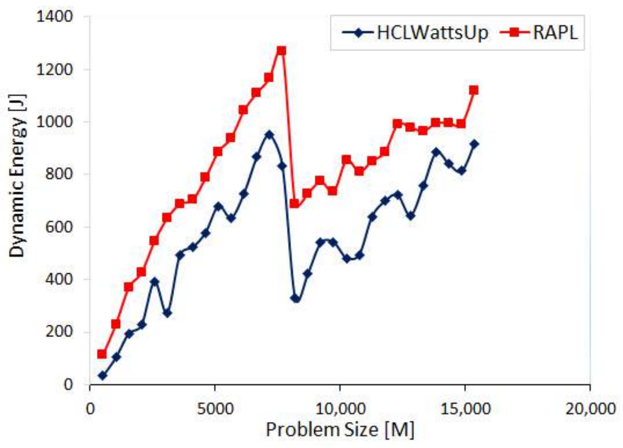
$$(M, K) =_{M \in (512, N/2), K \in (N-512, N/2), M+K=N} (e_1(M, N) + e_2(K, N)) \tag{4.1}$$



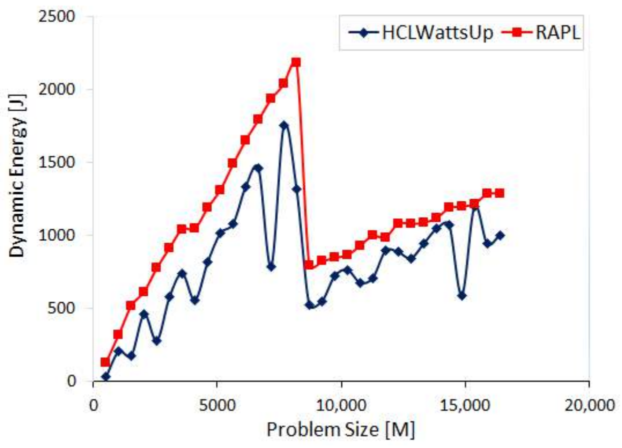
(a) N = 14,336



(b) N = 14,848



(c) N = 15,360



(d) N = 16,384

Figure 4.10: Dynamic energy consumption profiles of DGEMM on HCLServer01 and HCLServer02.

We determine the workload distribution for each workload size using the dynamic energy profiles with RAPL and HCLWattsUp as an input to the data partitioning algorithm. Using this workload distribution, we run the application in parallel on both servers and determine its dynamic energy consumption with RAPL and HCLWattsUp separately. Let $(e_{rapl}$ represent the total dynamic energy consumption by the given workload distribution on both servers with RAPL and $e_{hclwattsup})$ represent the total dynamic energy consumption by the same workload distribution on both servers using HCLWattsUp. Then, we can calculate the percentage loss of total dynamic energy consumption with RAPL compared with HCLWattsUp as $(e_{rapl} - e_{hclwattsup})/e_{hclwattsup} \times 100$.

Table 4.12 illustrates the total dynamic energy losses by using RAPL in comparison with HCLWattsUp, which are $\{54, 37, 31, 84\}$. After calibrating RAPL with HCLWattsUp on both platforms, we can reduce the losses to $\{16, 8, 12, 40\}$.

Table 4.12: Dynamic energy losses in percentage with RAPL in comparison with HCLWattsUp.

Problem Size (N)	[%] Energy Loss without Calibration	[%] Energy Loss after Calibration
14,336	54	16
14,848	37	8
15,360	31	12
16,384	84	40

4.8 Comparison of Costs of Measurement and Implementation Complexity

In this section, we compare the cost in terms of number of measurements to determine a single data point of an application dynamic energy profile constructed with all three aforementioned approaches. To determine the dynamic energy consumption by a given workload size of an application, one needs following three measurements with RAPL and HCLWattsUp:

1. Base power
2. Execution time of the application
3. Total Energy consumed by the application during the execution

However, the cost of determining the dynamic energy consumption with (RAPL and on-chip GPU/Xeon Phi) *sensors* is comparatively higher, because one needs at least following five measurements:

1. Base power with RAPL
2. Total Energy with RAPL
3. Base power with NVML/Intel SMC
4. Total Energy with NVML/Intel SMC

5. Execution Time

Only one measurement is needed to predict the dynamic energy consumption with PMC based energy predictive models. However, the cost of building the model is relatively higher as discussed in next section.

4.8.1 Implementation Complexity

Energy predictive models employing PMCs as predictor variables exhibit high implementation complexity due to the following reasons:

1. There is a large number of PMCs provided in a modern multicore processor. For example: Likwid tool [55] provides 164 PMCs and 385 PMCs for the Intel Haswell (Table 4.1) and the Intel Skylake multicore processors (Table 4.2) respectively.
2. Tremendous programming effort and time are required to automate and collect all the PMCs. This is because of the limited number of hardware registers available on platforms for storing the PMCs. Only 3-4 PMCs can be collected in a single run of an application. Moreover, some PMCs can only be collected individually or in sets of two or three for an application run. Therefore, each application must be executed about 53 and 99 times on the Intel Haswell and Intel Skylake platforms respectively to collect all the PMCs available on them.
3. An energy predictive model purely based on PMCs lacks portability. This is because all the PMCs available for a CPU processor may not be present in a GPU processor due to inherent architectural differences or even on in a next-generation CPU processor from the architecture space.

4.8.2 Topological granularity issues with sensors

Finally, we discuss the issues with topological granularity of on-chip sensors. Consider, for example, a hybrid application DGEMM executing in parallel on three compute devices, a multicore CPU and two accelerators (GPU and Xeon Phi). One CPU core acts as a host for each accelerator kernel. Execution of an application on GPU/Xeon Phi involves the CPU host-core, DRAM and PCIe to copy the data between CPU host-core and GPU/Intel Xeon Phi. However, the on-chip power sensors (NVML and MPSS) only provide the power consumption of GPU or Xeon Phi. Therefore, to obtain the dynamic energy profiles of applications, one can use RAPL to determine the energy contribution of CPU host-core and DRAM. But RAPL provides the energy consumption at the socket level, which includes also the contribution by other CPU cores involved in the execution of kernels running parallel on CPU and other accelerators. Therefore, it is not currently possible using on-chip sensors to accurately attribute the individual contribution of each computation kernel to total energy consumption by a hybrid application executing the kernels in parallel on several heterogeneous compute devices.

4.9 Current Picture, Recommendations and Future Directions

In this section, we cover the lessons learned and our recommendations for the use of on-chip sensors and energy predictive models before indicating some future directions.

Based on our study, we can not recommend use of state-of-the-art on-chip sensors (RAPL for multicore CPUs, NVML for GPUs, MPSS for Xeon Phis). The fundamental issue with this measurement approach is the lack of information about how a power reading for a component is determined during the execution of an application utilizing the component. While the accuracy of this information is reported in the case of NVML, experimental results demonstrate that practical accuracy is worse. Moreover, the dynamic energy profile patterns of the on-chip sensors differ significantly from the patterns obtained using the ground truth, which suggests that the measurements using on-chip sensors do not capture the holistic picture of the dynamic energy consumption during an application execution. At the same time, we observed that the energy measurements reported by the on-chip sensors are deterministic and hence can be used as parameters in energy predictive models.

Integrated on-chip sensors typically use voltage regulators to determine the power consumption by an component [45]. Hence, the accuracy of on-chip power sensors depends upon the accuracy of voltage regulators. It is reported that VRs from the same manufacturer lot may exhibit different accuracies [46]. Less accurate VRs (for example within an accuracy of $\pm20\%$) are used by original equipment manufacturer (OEM) for cost-saving purposes [46]. However, to compensate the reported inaccuracies of output current (IMON) from a VR to a processor, an approach is recently proposed to use a programmable load line from BIOS instead of actual implemented load line [46]. This programmed load line values adds an offset to the determined inaccuracy of the VR to increase its accuracy. We envisage the chip manufacturers to adopt such an approach to address the poor reliability and accuracy of power values provided by integrated on-chip sensors.

Energy predictive models based on PMCs are plagued by poor accuracy [50, 51, 15, 52]. The sources of this inaccuracy are the following: (a) Model parameters in most cases are not deterministic and reproducible and (b) Model parameters are selected chiefly based on correlation with energy and not their physical significance originating from fundamental physical laws such as conservation of energy of computing.

We will now state our recommendations and possible future directions. Since system-level physical measurements based on power meters are accurate and the ground truth, we recommend using this approach as the fundamental building block for the fine-grained device-level decomposition of the energy consumption during the parallel execution of an application executing on several independent computing devices in a computer.

We envisage hardware vendors maturing their on-chip sensor technology to an extent where energy optimization programmers will be provided necessary information of how a power measurement is determined for a component, the frequency or sampling rate of the measurements, its reported accuracy and finally how to programmatically obtain this measurement with sufficient accuracy and low overhead.

Linear energy predictive models can be employed in the optimization of applications for dynamic energy provided they meet the following criteria: (a) Model parameters employed in the models must be deterministic, (b) Model parameters are selected based on physical significance originating from fundamental physical laws such as conservation of energy of computing. Both the criteria are contained in the *additivity* test proposed in Reference [52]. Use of parameters with high additivity improves the prediction accuracy of the model. Additivity test can also be employed to select parameters for machine learning (or black box) methods such as neural networks, random forests, etc., provided the methods use linear functional building blocks internally. While there is experimental evidence demonstrating good accuracy for these types of models, a sound theoretical analysis is lacking. At this point, we do not recommend the use of non-linear energy predictive models since they lack serious theoretical and experimental analysis.

We believe that high-level model parameters designed by combining PMCs (using functions based on physical significance with dynamic energy) may be deterministic and reproducible instead of individual PMCs, which are raw counters. PMCs traditionally have been developed to aid low-level performance analysis and tuning but have been widely adopted for energy predictive modeling. We would call the high-level model parameters, energy monitoring counts (EMCs), that are discovered from insights based on fundamental physical laws such as conservation of energy of computing and that are ideal for employment as predictor variables in energy predictive models.

4.10 Summary

In this chapter, we present a comprehensive study comparing the accuracy of state-of-the-art on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, which we consider to be the ground truth. The measurements provided by on-chip sensors are obtained programmatically using RAPL for Intel multicore CPUs, NVML for Nvidia GPUs and Intel System Management Controller chip (SMC) for Intel Xeon Phis. To compare the approaches reliably, we presented a methodology to determine the component-level dynamic energy consumption of an application using system-level physical measurements using power meters, which are obtained using HCLWattsUp API.

For the study comparing the accuracy of on-chip power sensors with the ground truth, we employ 61 different application configurations of two scientific applications, dense matrix-matrix multiplication and 2D fast Fourier transform, executed on one Intel Haswell and two Intel Skylake multicore CPUs, two Nvidia Graphical Processing Units (GPUs) (Tesla K40c and Tesla P100 PCIe) and one Intel Xeon Phi accelerator. We show that the average error between the dynamic energy profiles obtained using on-chip power sensors and the ground truth ranges from 8% and 73% and the maximum reaches 300%.

For 2D-FFT applications executing on accelerators (GPUs or Intel Xeon Phis), we find that RAPL reports higher dynamic energy than the on-chip power sensors in the accelerators. It should be noted that RAPL reports energy consumption for only CPU and DRAM domains. It shows that the data transfers (between CPU host-core and the accelerator) consume more dynamic energy than the computations on the accelerator. This suggests that we can reduce the dynamic energy consumption by optimizing the dynamic energy of data transfer operations. Furthermore, we found that for 2D FFT, Intel MPSS and NVML provide more accurate dynamic energy consumption and exhibit similar trend as that of HCLWattsUp. For DGEMM, however, we find that RAPL measurements are significantly less than on-chip sensor values of the accelerators, which suggests that computations are the main contributor to the total dynamic energy consumption.

We show that, owing to the nature of the deviations of the energy measurements provided by on-chip sensors from the ground truth, calibration can not improve the accuracy of the on-chip sensors to an extent that can allow them to be used in optimization of applications for dynamic energy.

For the study comparing the prediction accuracy of energy predictive models with the ground truth, we use an experimental platform containing a test-suite of seventeen benchmarks executed on an Intel Haswell multicore CPU and an Intel Skylake multicore CPU. The average error between energy predictive models employing performance monitoring counters (PMCs) as predictor variables and the ground truth ranges from 14% to 32% and the maximum reaches 100%. We highlighted one of the causes of the inaccuracy in PMC based models, which is that they do not take into account the physical significance of the parameters based on fundamental law of conservation of energy of computing. Our experimental results illustrated that methods solely based on correlation with energy to select PMCs are not effective in improving the average prediction accuracy.

We demonstrated through a parallel matrix-matrix multiplication on two Intel multicore CPU servers that using inaccurate energy measurements provided by on-chip sensors for dynamic energy optimization can result in significant energy losses up to 84%.

Chapter 5

Accurate Energy Modelling of Hybrid Parallel Applications on Modern Heterogeneous Computing Platforms using System-Level Measurements

This chapter is mainly based on [21].

5.1 Introduction

Modern HPC platforms, cloud computing systems, and data centers are commonly composed of heterogeneous nodes where a multicore CPU is tightly integrated with one or more accelerators to address the twin critical concerns of performance and energy efficiency. A parallel application executing on such a hybrid node, consists of multiple kernels (generally speaking, multi-threaded), running in parallel on different computing devices of the platform. We term these kernels as application components for illustration purposes in this chapter.

An important challenge for energy optimization of hybrid parallel applications on such platforms is how to accurately estimate the energy consumption of its application components executing in parallel on different compute devices of the platform. The details of the issues involved in addressing the challenge are presented in section 1.1.3. In a nutshell, a naïve approach addressing this challenge has exponential complexity. An efficient solution method must take into account the inherent complexities in modern heterogeneous platforms and employ an accurate measurement method for energy consumption by an application.

There are three mainstream approaches for determining the energy consumption by an application: a). System-level physical measurements using external power meters, b). Measurements using on-chip power sensors, and c). Energy predictive models. System-level physical measurements using external power meters is considered the ground truth. A comparative study of on-chip sensors and energy predictive models against the ground truth is presented in chapter 4. In brief, the two state-of-the-art approaches (on-chip sensors and energy predictive models) suffer from poor accuracy and high implementation complexity [37]. To summarize, there exists no solution method to the best of our knowledge that employs system-level power measurements using external power meters to accurately determine the application component-level decomposition of the energy consumption of an application executing on multiple independent computing devices in a computer.

We propose a novel solution method called (Additive Energy Modelling of Hybrid Parallel Applications (AnMoHA)) to fill the gap. It comprises of two main stages. In the first stage, individual computing elements executing a given application kernel are grouped in such a way that we can accurately measure the energy consumption of the groups. The groups are termed as abstract processors. In the second stage, the discrete dynamic energy functions of the abstract processors are constructed using an additive modelling approach. We do not make any assumption about the shape of the energy profiles of application kernels. They can be linear or non-linear. Using this method, we address two challenges for energy optimization of hybrid parallel applications running on modern heterogeneous NUMA computing platforms:

1. Accurate modelling of the energy consumption of application components when executing a hybrid application in parallel on multiple compute devices on a computer.
2. Accurate modelling of the energy consumption of two different applications executing in parallel on a dual-socket multicore CPU platform.

The rest of the chapter is organized as follows. The so1lution method, AnMoHA, is explained in section 5.2. The experimental validation of AnMoHA is presented in section 5.3. Then, we discuss the trade-offs between the accuracy and time-space, and between the accuracy and design-space of AnMoHA in sections 5.4 and 5.5. We explore the hardware topological granularity and the scalability of AnMoHA in section 5.6. Then, we study the accuracy of additive energy modelling using on-chip sensors against the ground truth, and dynamic energy optimization with on-chip sensors and the ground truth in section 5.7. We discuss the scope, limitations and the efficacy of AnMoHA in section 5.8. Finally, the chapter is concluded in section 5.9.

5.2 AnMoHA: Additive Energy Modelling of Hybrid Applications on Heterogeneous Computing Platforms

In this section, we present our solution method, *AnMoHA*, to determine the dynamic energy consumption by a hybrid application executing in parallel on multiple heterogeneous computing devices such as multi-core CPU, GPU, Xeon Phi, etc., in a computing platform. The method is purely based on system-level power measurements using power meters.

The inputs to AnMoHA are the hybrid application comprising of multi-threaded kernels, the number of compute devices, and the precision settings (such as 2.5%) to be satisfied during the construction of the energy profiles. There is a one-to-one mapping between the application kernels and compute devices. The output of AnMoHA is the energy profiles of the application kernels satisfying the input precision settings.

AnMoHA is composed of two main stages. In the first stage, individual computing elements executing a given application kernel are grouped in such a way that we can accurately measure the energy consumption of the groups. The groups are termed as *abstract processors*. In the second stage, the discrete dynamic energy functions of the abstract processors are constructed using an *additive* modelling approach.

5.2.1 Grouping of Computing Elements

The rationale behind grouping the compute devices is to address one of the fundamental problems while modeling: to decide the granularity level to model the energy consumption by an application kernel. Unfortunately, there is not much privilege to model the accurate energy consumption by an application at a very fine granularity. Consider, for example, two applications executing on two different cores of a CPU socket in parallel. Currently, there is no possible way to determine the energy consumption of these applications accurately at the core level. Similarly, we cannot measure the energy consumption of certain data banks of DRAM used in the execution of an application, or for the PCIe links offloading application data to and from host CPU core to an accelerator. Therefore, it is quite important to formulate such an abstraction of all these components that allow modelling the energy consumption of such components sufficiently accurate.

Furthermore, modern multicore platforms have many inherent complexities such as severe resource contention for shared on-chip resources (Last Level Cache, Interconnect) and Non-Uniform Memory Access (NUMA). As a result, the workload of one computational kernel of a hybrid application (consists of a number of application components) may significantly impact the performance and energy consumed by the others due to tight integration and high resource contention in underlying heterogeneous hybrid platforms. Therefore, the computation kernels cannot be considered fully independent and their energy consumption should not be measured separately. To address this issue, we only consider such configurations of hybrid applications in this work where individual kernels are coupled loosely enough to allow us to construct their individual energy functions. This is an important constraint for AnMoHA. We term this constraint as *loose coupling* for illustration purposes.

To satisfy this constraint, we consider only such configurations in this work where there is one-to-one mapping between the given (CPU or accelerator) kernel and its corresponding device. Hence, each kernel runs only on its corresponding device and there is no more than one CPU kernel or accelerator kernel is running on the corresponding device. Then, each group of computing elements executing an individual kernel of the application together (such as the group of CPU cores executing the CPU kernel together) is modelled as an abstract processor [158]. This way, the executing hybrid heterogeneous computing platform is represented as a set of heterogeneous abstract processors such that the group of computing elements executing an application kernel together belong to only one abstract processor. This mutual exclusion of the compute elements ensures that the sharing of system resources is maximized within the groups representing the abstract processors and minimized between them. Hence, by definition an abstract processor contains solely the computing elements which execute an application kernel, and there is a one-to-one mapping between an abstract processor and its corresponding application kernel.

While the performance of each abstract processor can be measured individually when running the applications in parallel, it is not possible to measure their individual energy consumption if they are not independently powered. This is due to the fact that the abstract processors which share the same power domain cannot be considered fully independent and their energy can not be measured separately when executing the applications in parallel. Therefore, the abstract processors should be independently powered to measure the individual energy consumption by them when executing the applications in parallel.

Hence, the following two constraints must be satisfied when formulating the abstract processors for measuring the energy consumption by them using AnMoHA when executing the application kernels in parallel:

1. **Independently powered:** For a given platform, the abstract processors should be independently powered such that no two abstract processors share the same power domain.
2. **Loose coupling:** For a given hybrid application, the constituent components (kernels) are independent such that a component does not interfere the execution of other components during their parallel execution on their corresponding abstract processors. That is, if we run two applications A and B in parallel on two abstract processors AP_a and AP_b, then the dynamic energy consumption by A executing on AP_a is not affected by the application B executing in parallel on abstract processor AP_b.

We illustrate this concept by using an example. Consider HCLServer01 (technical specifications are provided in table 4.1) that is used in our experiments, containing an Intel Haswell multicore CPU with two sockets of twelve cores each, an Nvidia K40 GPU and an Xeon Phi 3120P coprocessor. Now, consider a hybrid parallel application DGEMM which computes the matrix multiplication of two dense matrices. Let the application uses MKL-DGEMM for CPU and Xeon Phi, and CUBLAS for Nvidia GPU. We formulate three abstract processors $\{CPU1, GPU1, PHI1\}$ that satisfy the aforementioned two constraints.

The abstract processor CPU1 contains 22 CPU cores executing the multi-threaded CPU kernel. The abstract processor GPU1 comprises of the Nvidia K40c GPU along with its dedicated host CPU core executing the GPU kernel, and the dedicated PCIe link between them. The third abstract processor PHI1 consists of the Xeon Phi 3120P co-processor along with its dedicated host CPU core executing the Xeon Phi kernel, and the dedicated PCIe link between them.

The dedicated host CPU core is responsible for sending the data from host to accelerator, kernel invocations on the accelerator and then copying the results back from the accelerator to host via the dedicated PCIe link between them. Therefore, the pair consisting of an accelerator (or co-processor) and its dedicated host core executing one accelerator kernel, and the dedicated PCIe link between the host core and accelerator is modelled by an abstract processor. The application kernel executing on an accelerator uses all the cores of the accelerator.

Hence, this formulation satisfies the both aforementioned constraints i.e. each abstract processor on HCLServer01 is independently powered and the application kernels of DGEMM are independent such that the execution of one does not interfere with the execution of the other. Based on this grouping of compute devices into abstract processors, the total dynamic energy consumption by an application executing on p abstract processors will be equal to the sum of dynamic energies consumed by all p abstract processors running the application. So, if E_{total}(x) is the total dynamic energy consumption by workload size x executing in parallel on p abstract processors $\{AP_1, \cdots, AP_p\}$, then

$$E_{total}(x) = \sum_{i=1}^{p} E_{AP_i}(x) \tag{5.1}$$

where E_{AP_i}(x) is the dynamic energy consumption by the workload size x executing on abstract processor AP_i. Table 5.1 describes the notations employed in this section.

5.2.2 Energy Models of Abstract Processors

The second main stage of AnMoHA consists of building the dynamic energy models of p abstract processors running parallel to the application kernels. We represent the dynamic energy model of an abstract processor with a discrete function composed of a set of points of cardinality m. There are $(2^p - 1) \times m$ number of total experiments available to build the dynamic energy profiles of a hybrid parallel application (containing, generally speaking, p number of independent multi-threaded application kernels) executing on p abstract processors containing m number of data points. There is a one-to-one mapping between the application kernel and an abstract processor.

Consider, for example, the abstract processors on HCLServer01. For illustration purposes, we represent the three abstract processors {CPU1, GPU1, PHI1} by {A, B, C}. Now, consider a hybrid parallel application DGEMM which computes the matrix multiplication of two dense matrices. Let the application uses MKL-DGEMM for CPU and Xeon Phi, and CuBLAS for Nvidia GPU. The goal is to construct the dynamic energy profiles of the application kernels running on three abstract processors {A, B, C} within sufficient accuracy. We can classify the total number of experiments into following categories: {A, B, C, {A,BC}, {AB,C}, {AC,B}, ABC}. The category {A, BC} represents the independent execution of application kernels on A, and parallel execution of application kernels on B and C. All categories hold commutative properties. That is, for example, the categories {BC, A} and {CB, A} are indistinguishable because they consume the same dynamic energy in both cases.

Table 5.1: Table of notations used in equations 5.1 and 5.2.

Notation	Description
AP_i	ith abstract processor.
x	workload size.
$E_{total}(x)$	Total dynamic energy consumption by workload size x.
$E_{AP_i}(x)$	The dynamic energy consumption by workload size x executing on ith abstract processor.
m	Cardinality (total number of data points in) of discrete energy function.
p	Total number of abstract processors/independent multi-threaded application kernels.
$E_{ABC}(x)$	Total dynamic energy consumption by parallel execution of the same application kernels of the workload size x on the abstract processors A, B, and C.
$E_A(x)$, $E_B(x)$, $E_C(x)$	The dynamic energy consumption by the application kernels of workload size x executing sequentially on abstract processors A, B, and C respectively.

We consider a hypothesis that will reduce the number of experiments to $p \times m$. We call it as the *additive hypothesis*. It states that the total dynamic energy consumption by a hybrid application consists of several (generally speaking, multithreaded) kernels running in parallel on p abstract processors equals the sum of dynamic energy consumption by all p abstract processors when running the same application kernels sequentially.

Let $E_A(x)$, $E_B(x)$, and $E_C(x)$ be the dynamic energy consumption by the application kernels of workload size x executing sequentially on abstract processors A, B, and C. Let $E_{ABC}(x)$ be the total dynamic energy consumption by parallel execution of the same application kernels of the workload size x on the abstract processors A, B, and C. Then, the additive hypothesis means the following:

$$E_{ABC}(x) = E_A(x) + E_B(x) + E_C(x) \tag{5.2}$$

So, if the additive hypothesis is validated, we can build the energy model of the abstract processor A independent of energy model of B or energy model of C. These models (discrete functions) then can by supply as an input to a workload partitioning algorithm to optimize the dynamic energy and total energy consumption of the computing platform composed of the given abstract processors. The additive hypothesis holds the associative property and commutative property of addition. We do not make any assumption about the shape of energy profiles of application kernels. They can be linear or non-linear. Furthermore, the additive hypothesis does not make any assumption about the workload distribution to their corresponding compute devices (abstract processors). That is, the applications and workloads executed by the abstract processors can be different.

5.3 Experimental Validation of AnMoHA

In this section, we experimentally validate AnMoHA.

5.3.1 Experiment Platforms and Applications

We use three applications for our experiments using a diverse range of problem sizes: (i). Matrix-matrix multiplication (DGEMM) which computes the matrix product of two dense matrices of size N $\times$ N, (ii). 2D fast Fourier transform (2D-FFT) which computes the discrete Fourier transform of a complex signal matrix of size M $\times$ N, and (iii). a gene sequencing application executing the Smith-Waterman algorithm (SW) algorithm ([159, 160]). Highly optimized kernels for the CPUs and the accelerators are used. For the CPUs, Intel MKL is employed and the version on both nodes is 2017.0.2. Accelerators typically have limited in-card memory and cannot run workload sizes that exceed the memory. Therefore, out-of-card packages, ZZGEMMOOC [157] for Nvidia GPUs and XeonPhiOOC [157] for Xeon Phis, are used. The ZZGEMMOOC and XeonPhiOOC packages reuse CUBLAS and MKL BLAS for in-card DGEMM calls.

We choose matrix-matrix multiplication and fast Fourier transform routines because they are fundamental kernels employed in scientific applications [154]. Furthermore, matrix-matrix multiplication is highly compute-intensive whereas 2D-FFT is memory-intensive. Hence, they exhibit different application characteristics. The gene sequencing application deals with alignment of DNA or protein sequences. It employs the Smith-Waterman algorithm which uses a dynamic programming (DP) approach to determine the optimal local alignment score of two sequences: i) a query sequence of length m, and ii) a database sequence of length n. The time and space complexities of the Smith-Waterman dynamic programming algorithm are $O(m \times n)$ and $O(m)$, where $m < n$, assuming the use of refined linear-space methods. The application uses optimized SW routines provided by SWIPE for Multicore CPUs [161], CUDASW++3.0 for Nvidia GPU accelerators [162], and SWAPHI for Xeon Phi accelerators [163]. We present the detailed description of the application in appendix B.

Following two nodes are employed for comparative study: (a) HCLServer01 (Table 4.1) has an Intel Haswell multicore CPU having 24 physical cores with 64 GB main memory and integrated with two accelerators: one Nvidia K40c GPU and one Xeon Phi 3120P, (b) HCLServer02 (Table 4.2) has an Intel Skylake multicore CPU consisting of 22 cores and 96 GB main memory and integrated with one Nvidia P100 GPU. These nodes are the representative of computers used in cloud infrastructures, supercomputers and heterogeneous computing clusters.

Each node has a *Watts Up Pro* power meter installed between its input power sockets and the wall A/C outlets. *Watts Up Pro* power meters are periodically calibrated using the ANSI C12.20 revenue-grade power meter, Yokogawa WT310. The calibration details are provided in appendix C. The maximum sampling speed of *Watts Up Pro* power meters is one sample every second. The accuracy specified in the data-sheets is $\pm 3\%$. The minimum measurable power is 0.5 watts. The accuracy at 0.5 watts is ± 0.3 watts. The accuracy of Yokogawa WT310 is $\pm 0.1\%$ and the sampling rate is 100K samples per second.

To obtain the power measurements from the WattsUp Pro power meters, an automated tool HCLWattsUp interface [38] is used. HCLWattsUp interface has no extra overhead and therefore does not impact the dynamic energy consumption by the application kernel. The interface and the methodology used to obtain a data point are explained in sections 3.3 and 3.4.

HCLWattsUp follows a detailed sophisticated methodology (as explained in section 3.5) to ensure the reliability of our experimental results. The methodology determines a sample mean (execution time or dynamic energy or PMC) by executing the application repeatedly until the sample mean meets the statistical confidence criteria (95% confidence interval, a precision of 0.025 (2.5%)) for the experiments used in this chapter unless specified otherwise. Student's *t*-test is used to determine the sample mean. The test assumes that the individual observations are independent and their population follows the normal distribution. We use Pearson's chi-squared test to ensure that the observations follow normal distribution. To eliminate the potential contribution by other components such as SSD (Solid State Drives), fans, etc. when measuring the energy consumption by an abstract processor, we follow a strict experimental methodology explained in section 3.4.

5.3.2 Formulation of Abstract Processors on HCLServers

We group the compute devices of both the platforms into following five abstract processors satisfying the constraints of *independently powered* and *loose coupling*.

- Abstract processors, CPU1, GPU1 and PHI1 on HCLServer01, as explained in section 5.2.1.
- CPU2 containing 21 CPU cores executing the multi-threaded CPU kernel on HCLServer02.
- GPU2 comprising of the Nvidia P100 GPU, the dedicated host CPU core executing the GPU kernel, and the dedicated PCIe link between the host core and the GPU on HCLServer02.

Let $E_A(x)$, $E_B(x)$, and $E_C(x)$ be the dynamic energy consumption by the application kernels of workload size x executing sequentially on abstract processors CPU1, GPU1, and PHI1, and $Combined_{ABC}(x)$ represents the sum value of their dynamic energy consumption. Let $E_{ABC}(x)$

be the total dynamic energy consumption by parallel execution of the same application kernels of the work load size x on the abstract processors CPU1, GPU1, and PHI1, which is represented by $Parallel_{ABC}(x)$. Then, the additive hypothesis holds only if $Parallel_{ABC}(x) = Combined_{ABC}(x)$. The description of notations used in additive hypothesis is provided in table 5.3. For illustration purposes, we refer the additive energy models composed using AnMoHA as *Combined*, and compare their accuracy against the *Parallel* energy profiles which we consider as ground truth.

To determine if the additive hypothesis is valid, we build four dynamic energy profiles for HCLServer01 (one parallel and one for each of the three abstract processors), and three profiles for HCLServer02 (one parallel and one for each of the two abstract processors) for each application configuration. Then we sum the dynamic energy consumption by sequential execution of the application and compare the value with dynamic energy consumption by parallel execution of the same application.

This abstract processor formulation also minimizes the contention and mutual dependence between the kernels. We find no difference between the execution times of the application kernels when running sequentially and parallel. The parallel execution time of the hybrid application is equal to the maximum of the execution times of the kernels run sequentially.

5.3.3 Results and Analysis

HCLServer01 and HCLServer02 both have different architectures and CUDA versions on their respective GPUs. We use a hybrid configuration of multi-threaded DGEMM which runs in parallel on different compute devices (i.e. CPUs, GPUs, Xeon Phi) of the given platform. We observe that the GPU kernel completes its execution faster than the CPU on both platforms. However, in contrast with HCLServer01, DGEMM does not destroy its context on GPU during the parallel execution on HCLServer02 and keeps on consuming a constant power which is a bit higher than the base (or idle) power of GPU until all other kernels complete their execution on their respective abstract processors. Nevertheless, the combined profile (composed by summing the dynamic energy consumption of serial execution of the application on abstract processors) does not capture this behavior. Therefore, to calibrate the combined dynamic energy profile of DGEMM with a parallel dynamic energy profile on HCLServer02, we add the dynamic energy consumption to keep alive this context with dynamic energy consumption by the GPU application kernel. Hence, the equation 5.2 can be extended as

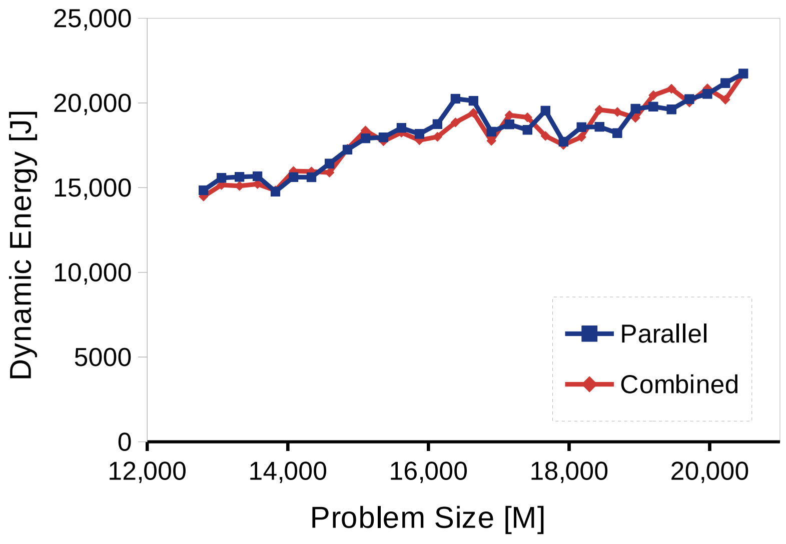
$$\begin{aligned} &E_{ABC}(x) = E_A(x) + E_B(x) + E_C(x) + \overline{e} \\ &\text{where } \overline{e} \geq 0 \end{aligned} \tag{5.3}$$

where $\overline{e}$ denotes the dynamic energy consumed by the application kernel to keep its context alive. The description of notations used in equation 5.3 is provided in table 5.3.

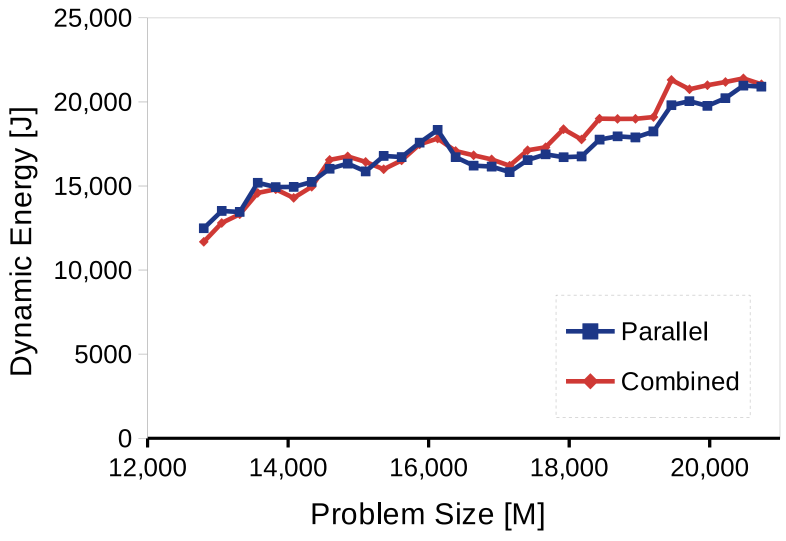
We run three different workload configurations (M $\times$ N where M $\leq$ N) of DGEMM executing on HCLServer01 and for each configuration four dynamic energy functions of DGEMM are built as explained in section 5.3.2. The workload sizes range for all three configurations are as follows: i) from 12800 $\times$ 20224 to 20224 $\times$ 20224 with a constant step size of 128 for the dimension M, ii). from 12800 $\times$ 20480 to 20480 $\times$ 20480 with a constant step size of 256 for the dimension M, and, iii). 12800 $\times$ 20736 to 20736 $\times$ 20736 with a constant step size of 256 for the dimension M. Figure

5.2(a), 5.1(a) and 5.1(b) illustrate the parallel and combined dynamic energy profiles for N=20224. N=20480, and N=20736 respectively.

We find the average and maximum errors between combined and parallel dynamic energy profiles to be 2.24% and 5.56% for N=20224, 3.07% and 7.6% for N=20480, and 3.87% and 9.97% for N=20736, respectively. Furthermore, we find that the execution time of a hybrid application is the maximum of all the execution times of its constituent kernels.



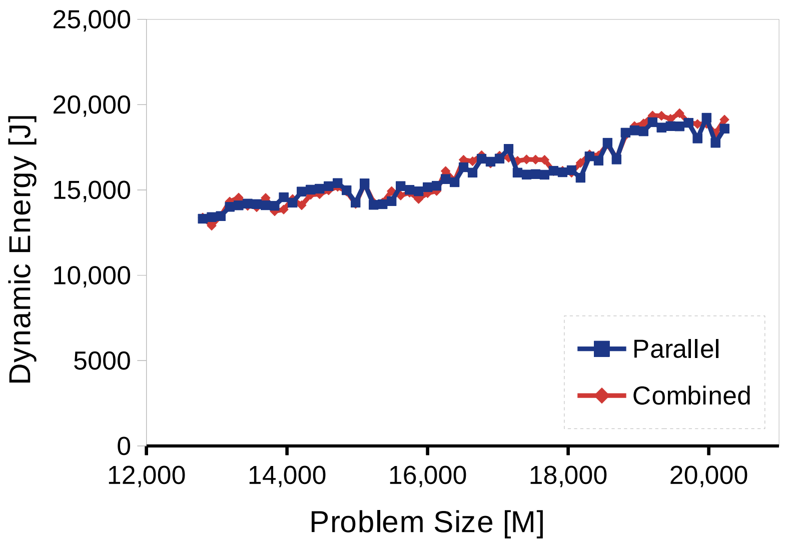
(a) N=20480



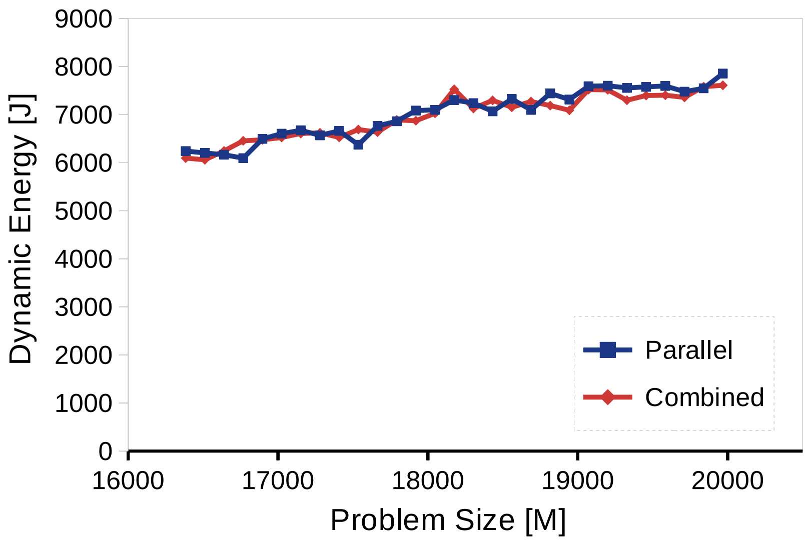
(b) N=20736

Figure 5.1: Dynamic energy profiles of DGEMM on HCLServer01.

We then compute 2D-FFT for the problem sizes ranging from 15104 $\times$ 23552 to 18560 $\times$ 23552 with a constant step size of 64. Figure 5.3(a) shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer01. The average and maximum errors between parallel and combined dynamic energy profiles are 4.32% and 8.91%. For our next batch of experiments on HCLServer01, we run SW application for the problem sizes ranging from 16384 $\times$ 16384 to 18240 $\times$ 16384 with a constant step size of 64. Figure 5.4(a) shows the dynamic energy parallel and

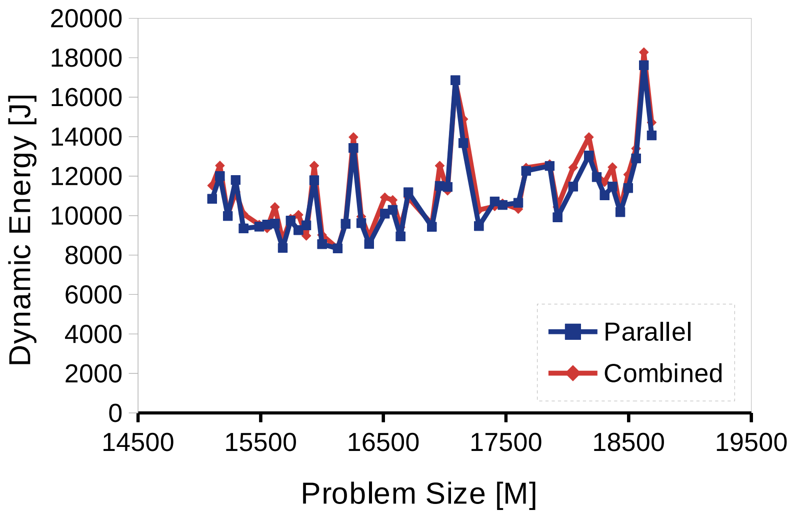


(a) N=20224, HCLServer01

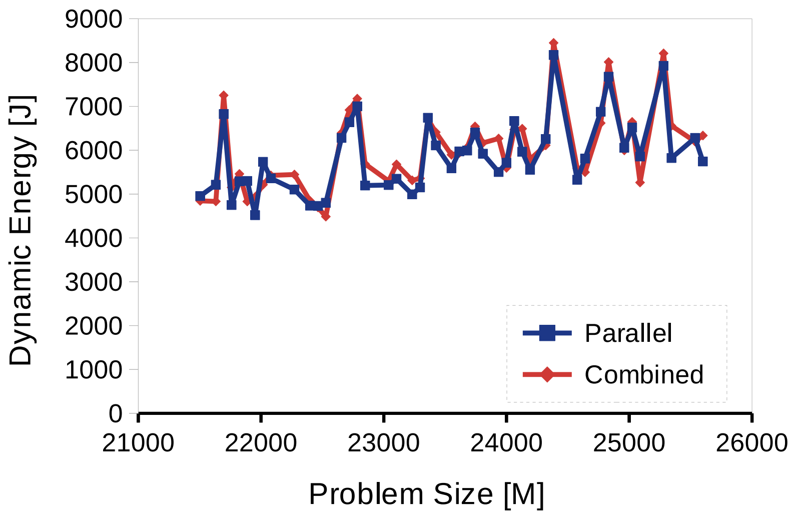


(b) N=22528, HCLServer02

Figure 5.2: Dynamic energy profiles of DGEMM on HCLServers.



(a) N=23552, HCLServer01



(b) N=25600, HCLServer02

Figure 5.3: Dynamic energy profiles of 2D-FFT on HCLServers.

combined profiles of SW executing on HCLServer02. The average and maximum errors between parallel and combined dynamic energy profiles are 2.14% and 5.4%.

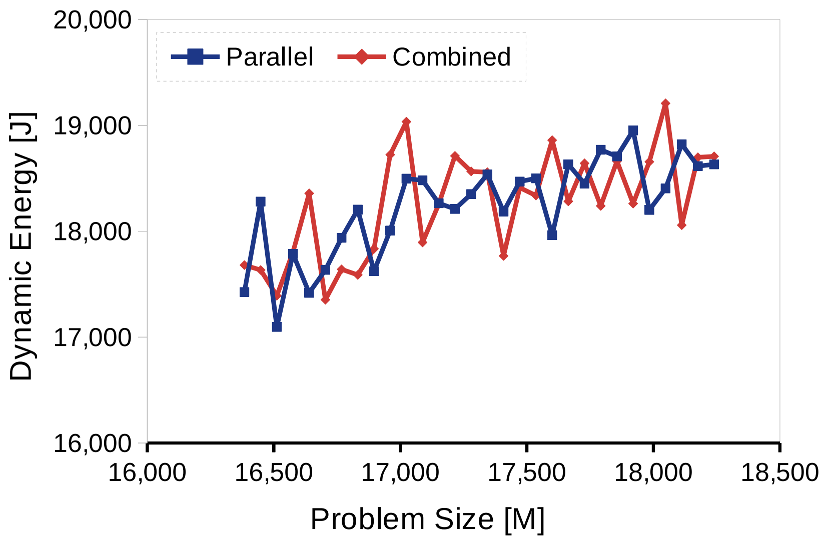
On HCLServer02, we run DGEMM with workload sizes range from 16384 $\times$ 22528 to 20096 $\times$ 22528 with a constant step size of 128. Figure 5.2(b) shows the dynamic energy profiles of parallel and combined on HCLServer02. We find the average and maximum errors between combined and parallel dynamic energy profiles to be 2.32% and 6.6%. We then compute 2D-FFT on HCLServer02 for the problem sizes ranging from 21504 $\times$ 25600 to 25600 $\times$ 25600 with a constant step size of 64. Figure 5.3(b) shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer02. The average and maximum errors between parallel and combined dynamic energy profiles are 4.87% and 13.87%. For our final batch of experiments for this study, we run the SW application for the problem sizes ranging from 40000 $\times$ 16384 to 42624 $\times$ 16384 with a constant step size of 64. Figure 5.4(b) shows the dynamic energy parallel and combined profiles of SW executing on HCLServer02. The average and maximum errors between parallel and combined dynamic energy profiles are 1.22% and 3.6%. Table 5.2 represents the percentage error between parallel and combined dynamic energy profiles of DGEMM, 2D-FFT and SW on HCLServers.

Table 5.2: Percentage error between parallel and combined dynamic energy profiles on HCLServers.

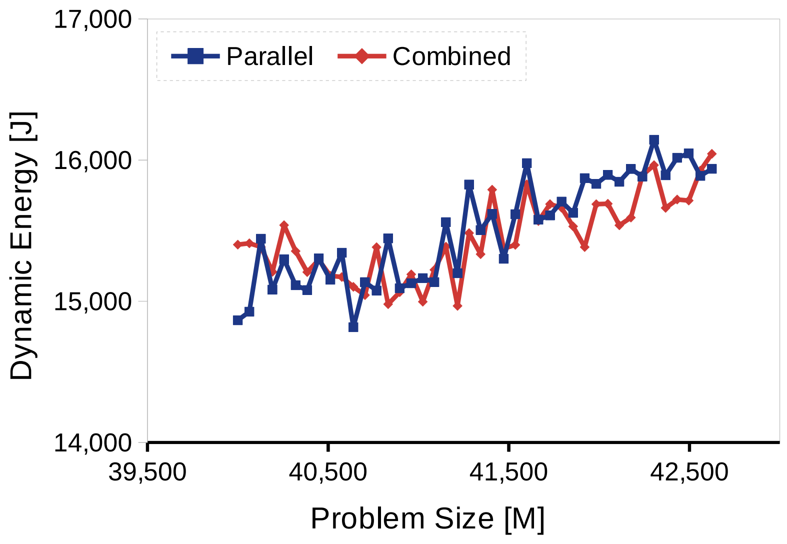
DGEMM				
Platform	**Problem Size [N]**	**Min%**	**Max%**	**Avg%**
HCLServer01	20224	0.016	5.56	2.24
HCLServer01	20480	0.07	7.6	3.07
HCLServer01	20736	0.56	9.97	3.87
HCLServer02	22528	0.29	6.60	2.32
2D-FFT				
HCLServer01	23552	0.005	8.91	4.32
HCLServer02	25600	0.46	13.87	4.87
SW				
HCLServer01	16384	0.05	5.4	2.14
HCLServer02	16384	0.07	3.61	1.22

5.4 Trade-off between accuracy and time space of additive modelling

We illustrate the impact of the precision settings of an experiment for obtaining the data points of the dynamic energy profile of an abstract processor with an example. Let the execution time of a workload size j on an abstract processor i be t seconds. Let HCLWattsUp repeat the application for n times to obtain the average dynamic energy consumption within the precision settings of ϵ. Assume the cool-down period allowed to exclude the pipeline and cache effects between two successive runs of the application is s seconds. Then, it takes $n_{ij} \times (t_{ij} + s)$ seconds in total to obtain the average dynamic energy consumption by the workload size j on abstract processor i. If m is total data points in dynamic energy profile of each of p abstract processors, then the total time to build their dynamic energy profiles can be calculated by using the following equation:



(a) N=16384, HCLServer01



(b) N=16384, HCLServer02

Figure 5.4: Dynamic energy profiles of Smith-Waterman application on HCLServers.

Table 5.3: Table of notations used in equations 5.3 and 5.4.

Notation	**Description**
$Parallel_{ABC}(x)$	The total dynamic energy consumption by parallel execution of the same application kernels of the work load size x on the abstract processors A, B and C.
$Combined_{ABC}(x)$	The sum value of the dynamic energy consumption by the application kernels of workload size x executing sequentially on abstract processors A, B and C .
$\overline{e}$	The dynamic energy consumed by the application kernel to keep the context alive on GPU.
T	Total time to build the dynamic energy profiles
j	The workload size
n_{ij}	number of repetitions to obtain the average dynamic energy consumption by the workload size j on ith abstract processor within the user defined precision settings.
$epsilon$	The precision setting
s	The cool-down period between two successive repetitions
t_{ij}	The execution time of a workload size j on ith abstract processor.

$$T = \sum_{i=1}^{p} \left(\sum_{j=1}^{m} n_{ij} \times (t_{ij} + s) \right) \tag{5.4}$$

The description of notations used in equation 5.4 is provided in table 5.3. We observe that the precision settings highly impacts the overall time T required to build the dynamic energy profiles of abstract processors. Consider, for example, two accuracy settings 97.5% and 90% represented as ϵ_A and ϵ_B respectively. Let T_A and T_B be the total times required to build the dynamic energy profiles of p abstract processors within the accuracy of ϵ_A and ϵ_B respectively. We experimentally observe that T_A is much higher than T_B. This is because it requires more iterations to converge the sample mean for each data point of the profile due to the higher precision settings. As a result, it will take a longer time to build the dynamic energy profiles of p abstract processors. This will reduce the practical viability of AnMoHA.

In this section, we explore if we can relax the precision settings sufficiently enough to get the application convergence faster but, most importantly, without compromising the application trend and behavior (in terms of variations). That is, the combined dynamic energy profile must exhibit the same trend and must have similar variations as that of the parallel dynamic energy profile. Therefore, the following two conditions must be satisfied to use the additive dynamic energy profiles as an input to an energy optimization algorithm (such as [18], [20]) that uses the workload size as a decision variable.

1. **Trend of the profile**: Combined dynamic energy profile must follow the similar application trend as the parallel dynamic energy profile, and

2. **Variations**: Combined dynamic energy profile exhibits similar variations as of parallel dynamic energy profile.

We term it as *usability* test, for illustration purposes. Consider a dynamic energy profile consists of n data points $\{x_1 \cdots x_n\}$. For each pair of consecutive data points in profile, we calculate the percentage difference as $\frac{x_i - x_{i-1}}{x_{i-1}} \times 100$ where $i \in \{2 \cdots n\}$. If the result is positive then it suggests a percentage increase otherwise if the result is negative then it suggests a percentage decrease in dynamic energy consumption with respect to the immediate preceding data point. We determine if the less accurate combined dynamic energy profile follows the same trend as the accurate parallel dynamic energy profile. In an ideal case, the combined dynamic energy profile exhibits an increase/decrease in dynamic energy consumption following the parallel dynamic energy profile for all data points.

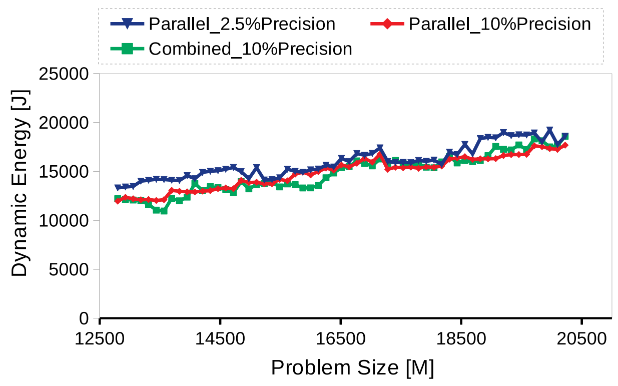
To analyze if both profiles exhibit the same variations, the deviations of the combined profile is compared with the parallel profile. To achieve this, we measure the degree to which each data point in every energy profile deviates from its average and maximum energy measurement (i.e. mean absolute deviation (MAD) around the sample mean and maximum absolute deviation around sample maximum). Let x represents a single data point of a dynamic energy profile consists of n data points, and $\overline{X}$ represents the mean dynamic energy consumption of that profile. We calculate the average absolute percent deviation from mean of that profile as $D_{avg}(\%) = \frac{MAD}{\overline{X}} \times 100$. where MAD is calculated as $MAD = \frac{1}{n}\sum_{i=1}^{n} |x_i - \overline{X}|$. Similarly, we calculate the maximum absolute percent deviation from the maximum of a profile using the formula for the average absolute percent deviation as above with $\overline{X}$ as max(X) where max(X) is the sample maximum of the profile.

5.4.1 Results and Discussion

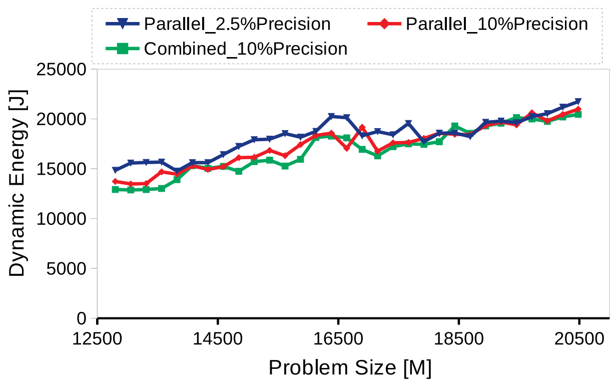
For this study, we repeat the same experiments to build energy profiles as explained in section 5.3.3, and keep all the experiment settings the same. However, we relax the precision settings from 2.5% to 10% to determine whether the combined dynamic energy profiles exhibit a similar trend as that of the parallel profile of the hybrid application. We term the dynamic energy profiles constructed with precision settings 2.5% and 10% as *accurate* and *less accurate* respectively for illustration purposes. We build four dynamic energy profiles for each application configuration similar to as explained in section 5.3.2 to compare the accuracy of AnMoHA with relaxed precision settings and to study the trade-off between the accuracy of the energy profiles and time taken to build them.

For our first batch of experiments, we run following three workload configurations of DGEMM ranging from i) 12800 $\times$ 20224 to 20224 $\times$ 20224 with a constant step size of 128, ii). 12800 $\times$ 20480 to 20480 $\times$ 20480 with a constant step size of 256 for the dimension M, and, iii). 12800 $\times$ 20736 to 20736 $\times$ 20736 with a constant step size of 256 for the dimension M. Figures 5.5(a), 5.5(b) and 5.5(c) illustrate the parallel and combined dynamic energy profiles for all three workload configurations with both precision settings. The average and maximum errors between the accurate parallel and less accurate combined dynamic energy profiles are 7.8% and 22.73% for N=20224, 12.08% and 24.74% for N=20480, and 7.14% and 17% for N=20736.

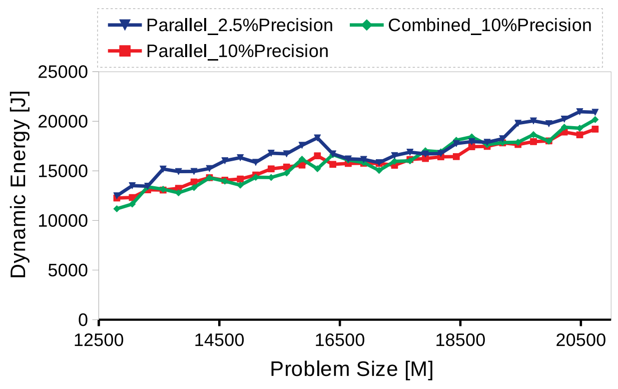
We then compute 2D-FFT on HCLServer01 for the problem sizes ranging from 15104 $\times$ 23552 to 18560 $\times$ 23552 with a constant step size of 64. Figure 5.6(a) shows the dynamic energy parallel



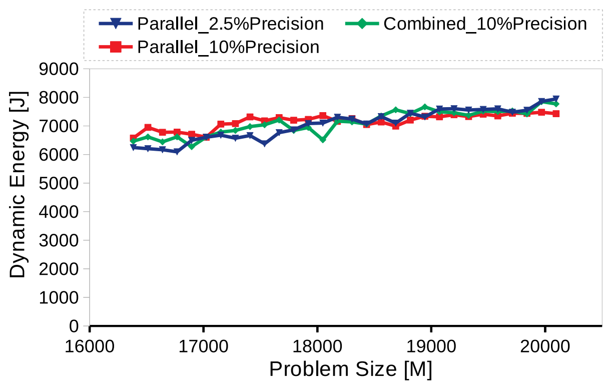
(a) N=20224, HCLServer01



(b) N=20480, HCLServer01



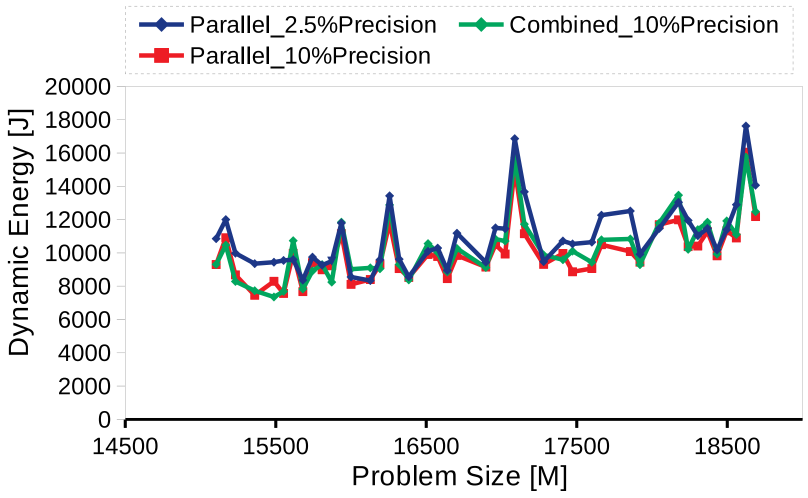
(c) N=20736, HCLServer01



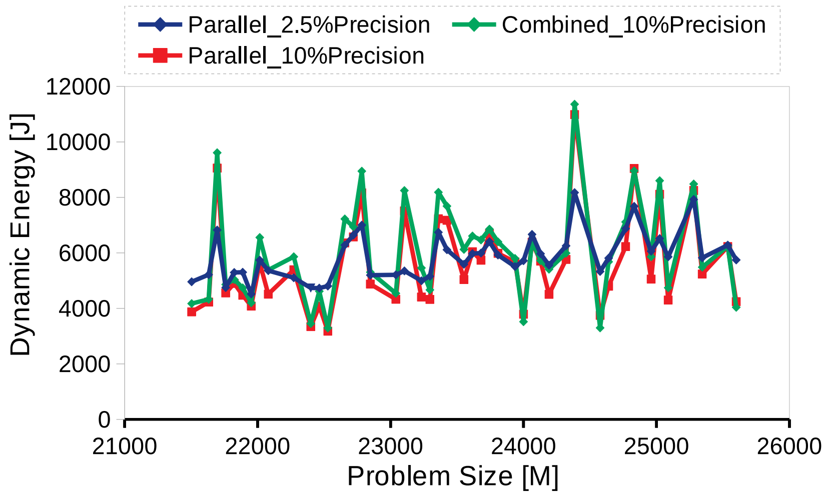
(d) N=22528, HCLServer02

Figure 5.5: Dynamic energy consumption by DGEMM on HCLServers.

and combined profiles of 2D-FFT executing on HCLServer01. The average and maximum errors between the accurate parallel and less accurate combined dynamic energy profiles are 8.16% and 22.1%. For our next batch of experiments on HCLServer01, we run SW application on HCLServer02 for the problem sizes ranging from 16384 $\times$ 16384 to 18240 $\times$ 16384 with a constant step size of 64. Figure 5.7(a) shows the dynamic energy parallel and combined profiles of SW executing on HCLServer01. The average and maximum errors between the parallel and combined dynamic energy profiles are 7.78% and 12%.



(a) N=23552, HCLServer01

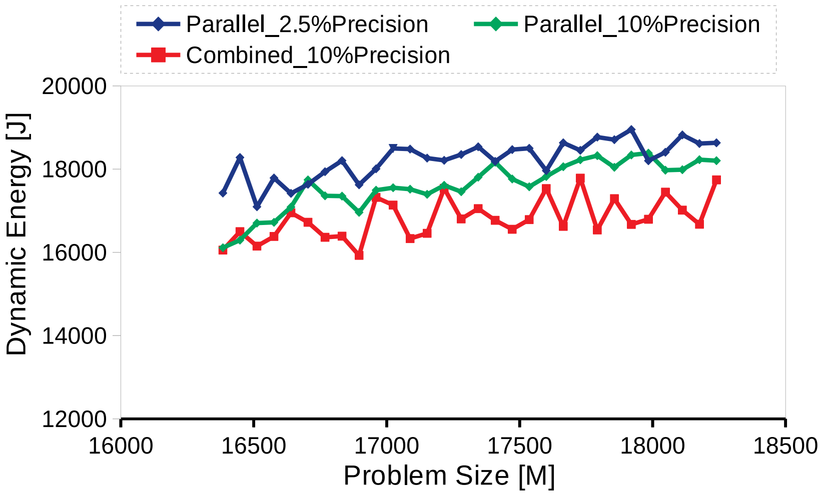


(b) N=25600, HCLServer02

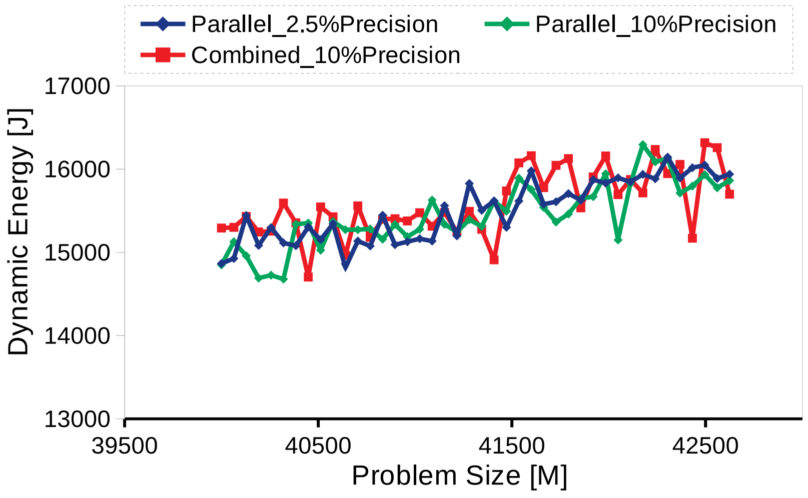
Figure 5.6: Dynamic energy consumption by FFT on HCLServers.

On HCLServer02, we run DGEMM with workload sizes ranging from 16384 $\times$ 22528 to 20096 $\times$ 22528 with a constant step size of 128. Figure 5.5(d) shows the parallel and combined dynamic energy profiles constructed with both precision settings on HCLServer02. The average and maximum error between less accurate combined and accurate parallel dynamic energy profiles are 3.08% and 10.43%. For our next batch of experiments, we compute the 2D-FFT on HCLServer02 for the problem sizes ranging from 21504 $\times$ 25600 to 25600 $\times$ 25600 with a constant step size of 64. Figure 5.6(b) shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer02 under both aforementioned precision settings. The average and maximum errors between the parallel and combined dynamic energy profiles are 14.56% and 54.18%.

For our final batch of experiments for this study, we run SW application on HCLServer02 for problem sizes ranging from 40000 $\times$ 16384 to 42624 $\times$ 16384 with a constant step size of 64. Figure 5.7(b) shows the dynamic energy parallel and combined profiles of SW executing on HCLServer02. The average and maximum error between parallel and combined dynamic energy profiles are 1.77% and 5.29%. Table 5.4 shows the percentage errors of combined profiles with parallel ones on both platforms.



(a) N=16384, HCLServer01



(b) N=16384, HCLServer02

Figure 5.7: Dynamic energy consumption by Smith-Waterman application on HCLServers.

Table 5.4: Percentage errors between parallel and combined dynamic energy profiles with 10% precision setting.

DGEMM				
Platform	**Problem Size [N]**	**Min**	**Max**	**Avg**
HCLServer01	20224	0.02%	22.7%	7.8%
HCLServer01	20480	0.21%	24.74%	12.08%
HCLServer01	20736	0.44%	17%	7.14%
HCLServer02	22528	0.05%	10.43%	3.08%
2D-FFT				
HCLServer01	22528	0.38%	22.1%	8.16%
HCLServer02	25600	0.24%	54.18%	14.56%
SW				
HCLServer01	16384	2.4%	12.03%	7.78%
HCLServer02	16384	0.07%	5.29%	1.77%

Table 5.5: Percentage absolute mean and maximum deviations of dynamic energy consumption by accurate parallel (with 2.5% precision setting) and less accurate combined profiles with 10% precision settings on HCLServers. Here 's01' denotes HCLServer01 and 's02' denotes HCLServer02.

Platform and Application	Problem Size [N]		Parallel _acc	Combined _lacc
s01-DGEMM	20224	**Avg**	8.56	11.84
		Max	17.36	20.5
s01-DGEMM	20480	**Avg**	8.27	11.9
		Max	16.3	22.28
s01-DGEMM	20736	**Avg**	9.5	12.16
		Max	19.11	21.4
s01-FFT	23552	**Avg**	13.86	13.8
		Max	37.82	34.64
s02-DGEMM	22528	**Avg**	6.5	5.23
		Max	11.3	9.4
s02-FFT	25600	**Avg**	11.2	23
		Max	28.3	46.7
s01-SW	16384	**Avg**	1.96	2.34
		Max	3.8	5.47
s02-SW	16384	**Avg**	2.11	2.07
		Max	4	4.46

One can observe that for all application configurations on HCLServers, combined dynamic energy profiles exhibit a similar energy consumption behavior as of parallel dynamic energy profile. Table 5.5 presents the absolute percent deviations from the mean and maximum of dynamic energy consumption by accurate parallel and less accurate combined profiles. The less accurate combined dynamic energy profiles of DGEMM on HCLServer01 for the batch of experiments where dimension N is {20224,20480,20736} follow the similar energy consumption trend as their corresponding accurate parallel dynamic energy profiles for {86,80,83} percent of data points. On HCLServer02, the less accurate DGEMM combined dynamic energy profile exhibits a similar energy consumption trend as its accurate parallel profile for 81% of the data points. For 2D-FFT, the less accurate dynamic energy profiles on HCLServer01 and HCLServer02 show a similar energy consumption trend as their respective parallel profile for {88,89} percent of the data points. Hence, despite relaxing the precision settings of the experiments to 10%, the combined dynamic energy profiles still follows the application trend and shows similar variations. Therefore, the additive dynamic energy profiles qualify the usability criterion and can be employed as input to the energy optimization algorithm [20].

As explained in equation 5.4, the precision settings highly impact the overall time T to construct the dynamic energy profile of an application, and higher precision settings take longer to build the dynamic energy profile of an application. In figure 5.8, we compare the accuracy against the time to build the dynamic energy profiles of DGEMM, FFT and SW on HCLServers under the precision settings of 2.5% and 10%. While the energy profiles are highly accurate under the precision settings of 2.5%, the time to construct them is relatively much higher.

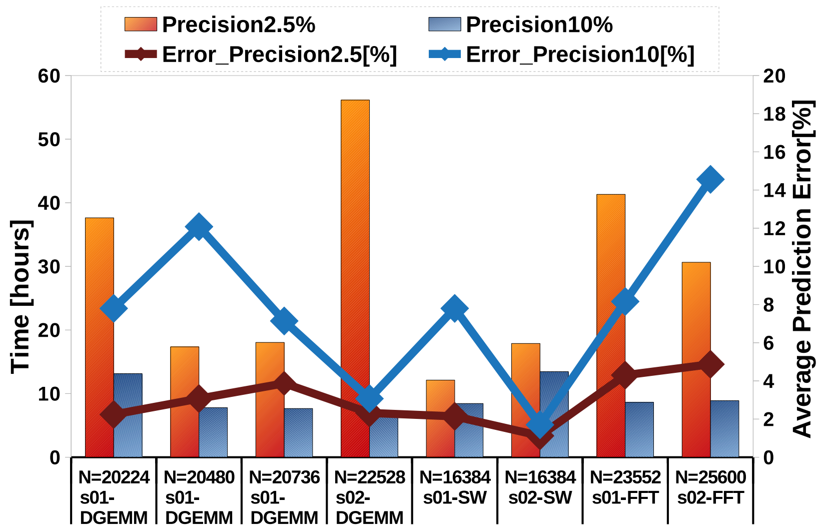


Figure 5.8: Trade-off between the time to build energy models using AnMoHA and their accuracy.

An important finding is that we can significantly reduce the time to build the energy profile of an application by slightly compromising the accuracy. Consider, for example, the average error of DGEMM (N=22528) on HCLServer02 is 2.32% under precision settings 2.5%. However, it takes more than 56 hours to build all three dynamic energy profiles on HCLServer02. In contrast, it takes around 7 hours to build the energy profiles (with an average error of 3.07%) of the same application under the precision settings 10%. Similarly, it takes more than 41 hours to construct all four energy profiles of FFT on HCLServer01 with an average error of 4.3%. However, it takes about 8 hours to build the less accurate profiles of the same application with an average error of 8.16%. For SW on HCLServer02, it takes about 18 hours to construct all three dynamic energy profiles under precision settings of 2.5%. The average error of the combined profile is 1.12%. However, it takes about 13 hours to construct the same profiles under precision settings 10% and the average error of the combined energy profile is 1.17%. The relaxed precision settings for SW reduced the time to build the same energy profiles by 27% whereas the average error of the combined profiles is increased by just 4.5%.

It is important to note here that the high execution times (in hours) taken to build energy models are mainly due to the high execution time of the workload size and high precision settings of the experiments as discussed earlier in equation 5.4. We use a detailed methodology to ensure the reliability of our results as explained in section 3.5. Briefly, to obtain a data point for each energy function, the software follows Student's t-test and executes the application repeatedly until the sample mean lies within user-defined confidence interval (CI) and a user-defined precision has been achieved. The software starts measuring precision and CI after taking the mean of five repetitions of the application.

We illustrate the impact of precision settings and the execution time of the workload size on total time to build an energy profiles by an example. Consider, for example, the gene sequencing application SW. The CPU takes on average 112 seconds to execute each data point of the energy profile of SW on HCLServer02 for the experiments discussed in section 5.3.3. Hence, it takes at least 560 seconds to reliably determine a single data point within user-defined accuracy, and more than 6 hours for a profile comprises of 43 points. However, it takes 3 seconds on average by the GPU for each data point for a single run, and about 10 minutes to obtain the energy profile with same cardinality and precision settings. Similarly, it takes more than 6 hours to build the energy profile with same caridnality and precision settings when executing workloads parallel on CPU and GPU due to the higher execution time by CPU kernels. Therefore, it takes at least about 13 hours to construct all three energy profiles of SW of cardinality 43 on HCLServer02 (for given configuration settings discussed in section 5.3.3). It can take more time if the application repetitions are more than five to get the convergence. It would have taken far less time, however, had the workload is executed for just once.

It is important to note that the usability test just presents a criterion to determine the degree to which an additive model exhibits the similar energy consumption behavior to the ground truth. The percentage that indicates whether the test is passed, is highly dependent on application domain and a matter of choice. Consider, for example, the applications such as signal processing or multimedia processing. Such fault-tolerant applications belong to the approximate computing domains. A possibly inaccurate result is also acceptable in such domains. Therefore, a comparatively relaxed precision settings and relatively inaccurate model can serve the purpose in this case. However, high precision settings are required for the applications such as cryptography or hard real-time applications. Therefore, one will set a comparatively higher percentage to ensure whether the additive models qualifies the usability test. That is why the usability test does not define a percentage limit to indicate the passing threshold.

5.5 Trade-off between accuracy and design space of additive modelling

In this section, we analyze and compare the accuracy of different experiment configurations to build the additive dynamic energy model of an abstract processor. Each experiment configuration is an independent experiment. The objective of this study is to determine such an experiment configuration that provides the most accurate model of the dynamic energy profile of an application by exploring all possible combinations of independent experiments.

We run our experiments on HCLServer01 for this study, and follow the same analogy: {A,B,C} to represent the abstract processors of HCLServer01 as explained in sections 5.2.2 and 5.3.2. Using the additive hypothesis, we can build the dynamic energy profile of abstract processors A, B, and C considering different experiment configurations. The independent experiments providing the dynamic energy consumption by a workload size x executed on abstract processor A are:

$$E_A(x)_1 = E_A(x) \tag{5.5}$$

$$E_A(x)_2 = E_{AB}(x) - E_B(x) \tag{5.6}$$

$$E_A(x)_3 = E_{AC}(x) - E_C(x) \tag{5.7}$$

$$E_A(x)_4 = E_{ABC}(x) - E_{BC}(x) \tag{5.8}$$

$$E_A(x)_5 = E_{ABC}(x) - E_B(x) - E_C(x) \tag{5.9}$$

Likewise, the dynamic energy consumption by workload x executed on the abstract processors B or C using the independent experiments: $\{E_B(x), \{E_{AB}(x) - E_A(x)\}, \{E_{BC}(x) - E_C(x)\}, \{E_{ABC}(x) - E_{AC}(x)\}, \{E_{ABC}(x) - E_A(x) - E_C(x)\}\}$, and $\{E_C(x), \{E_{AC}(x) - E_A(x)\}, \{E_{BC}(x) - E_B(x)\}, \{E_{ABC}(x) - E_{AB}(x)\}, \{E_{ABC}(x) - E_B(x) - E_A(x)\}\}$. Hence, we can compose five combined profiles using these experiment configurations as {Combined$_1$, Combined$_2$, Combined$_3$, Combined$_4$, Combined$_5$} such as Combined$_1$ = $E_{A1} + E_{B1} + E_{C1}$.

For this study, we build the dynamic energy profiles of DGEMM on HCLServer01 for the workload sizes ranging from 12800 $\times$ 20224 to 20224 $\times$ 20224 with a constant step size of 256. We use the same experimental settings as explained in section 5.3.1. However, it takes significant time to run all possible experiment configurations. Because, we build all possible $2^p - 1$ profiles (each composed of a set of data points of cardinality m) and, therefore, we have to run $(2^p - 1) \times m$ experiments in total for this study.

5.5.1 Results and Discussion

Figure 5.9 represents the parallel and combined profiles composed of all possible independent experiments on HCLServer01. One can observe that, overall, Combined$_1$ and Combined$_5$ dynamic energy profiles has less percentage error with parallel dynamic energy profile, whereas Combined$_2$ dynamic energy profile has the largest percentage error.

The Combined$_1$ experiment configuration (using direct measurements) requires just 3 independent experiments to compose the combined profile which is 98% accurate, whereas Combined$_5$ experiment configuration needs 9 independent experiments for composing the combined profile which is 96% accurate. All other experiment configurations {Combined$_2$, Combined$_3$, Combined$_4$} needs 6 independent experiments to compose the combined dynamic energy profile which are {92.26%, 93.41%,94%} accurate.

Table 5.6 provides the percentage errors for each combined dynamic energy profile (composed by using different experimental design configurations) with the parallel dynamic energy profile. The

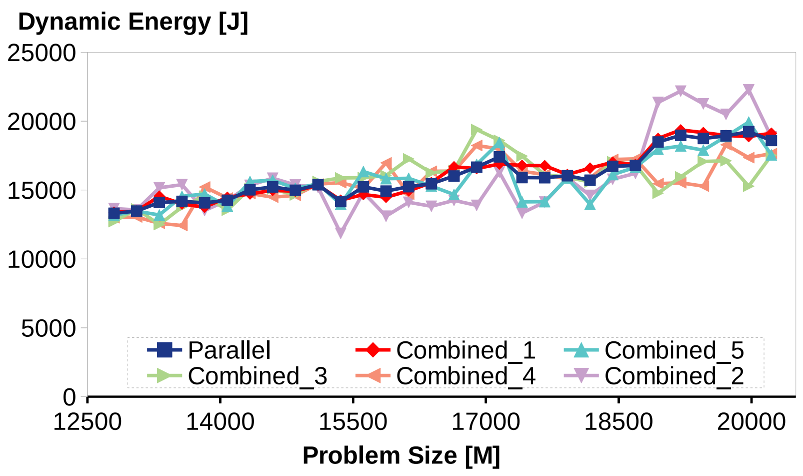


Figure 5.9: Dynamic Energy Profiles of DGEMM for all possible independent experiments on HCLServer01.

Table 5.6: Percentage errors between DGEMM combined and parallel dynamic energy profiles on HCLServer01.

Experiment Configuration	**Min**	**Max**	**Avg**
Combined$_1$	0.02%	5.56%	2.02%
Combined$_2$	0.02%	16.9%	7.74%
Combined$_3$	0.38%	20.56%	6.59%
Combined$_4$	0.3%	18.46%	5.99%
Combined$_5$	0.03%	11.12%	4.03%

average and maximum errors of the best dynamic energy profile (Combined$_1$) are {2.02%, 5.56%} and the worst dynamic energy profile (Combined$_2$) are {7.74%, 16.9%} respectively.

Table 5.7: Percentage deviations from mean and maximum of dynamic energy consumption by parallel and combined profiles.

Experiment Configuration	**Max**	**Avg**
Parallel	16.96%	8.74%
Combined$_1$	17.02%	9.76%
Combined$_2$	28.78%	13.48%
Combined$_3$	18.58%	7.79%
Combined$_4$	17.63%	8%
Combined$_5$	20.85%	9.07%

Table 5.7 presents the percentage deviations from the mean and maximum of dynamic energy consumption by parallel and each combined dynamic energy profiles. Let Dev_{par} and Dev_{com} represent the percentage deviation from the mean of dynamic energy consumption by parallel and combined profiles respectively. Then, the absolute percentage error between average deviations of parallel and combined profiles is calculated as $|(Dev_{par} - Dev_{com})|/Dev_{par} \times 100$. Similarly, the absolute percentage error between maximum deviations of parallel and combined profiles is also calculated in the same way. We find the absolute percentage error between average and maximum deviations of each aforementioned combined profiles and parallel profile as {11.6,54.2,10.87,8.47,3.78} and {0.33,69.67,9.56,3.91,22.93}.

The Combined$_1$, Combined$_2$ and Combined$_5$ dynamic energy profiles exhibit the same dynamic energy consumption as of parallel profile for more than 83% of the data points. In contrast, Combined$_3$, and Combined$_4$ dynamic energy profiles follow the application trend of parallel dynamic energy profile for 62% and 58.6% of the data points. Figure 5.10 illustrates the trade-off between the number of experiments to construct the combined dynamic energy profile using an experiment configuration and its percentage error with parallel dynamic energy profile.

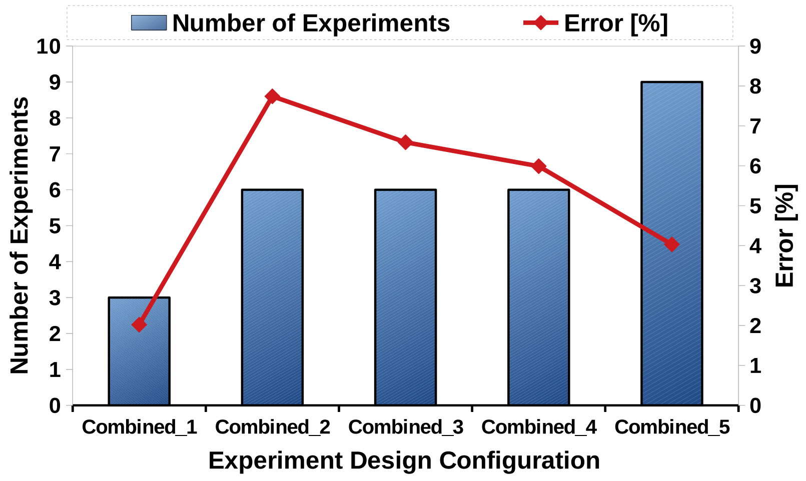


Figure 5.10: Trade off between number of experiments and accuracy.

To summarize, we find that experiment configuration (in equation 5.5) using direct energy measurements during the application run provides the most accurate dynamic energy consumption by an application kernel, and requires the least number of independent experiments. This is because of the fact that only the experiment configuration (in equation 5.5) measures the energy consumption during the application run, and therefore require only one experiment to determine it. In contrast, all other experiment configurations determine the energy consumption by the application kernel indirectly, and therefore require more numbers of independent experiments.

We illustrate this by an example. Consider the experiment configuration explained in equation 5.6. To determine the energy consumption by the abstract processor A when executing the workload size x, it requires following two independent experiments: i) parallel execution of workload size x on abstract processors A and B, and ii) serial execution of workload size x on abstract processor B. In order to determine the energy consumption by the abstract processor A when running the workload size x, it subtracts the energy consumption by the serial execution of workload size x on abstract processor B from the energy consumption by abstract processor A and B when running the workload size x in parallel on both of them. Consequently, two independent experiments are required to determine the energy consumption by abstract processor A when running the workload size x. That is why Combined$_1$ (which is composed of additive energy profiles using experiment configuration explained in equation 5.5) requires the least number of experiments. Furthermore, Combined$_1$ has the least error because unlike other experiment configurations, the energy consumption is measured during the execution of application.

The experiment configuration Combined$_5$ also provides accurate enough dynamic energy consumption by the application kernel. While it requires the most number of experiments to compose the combined dynamic energy profile, it exhibits a more similar application trend for 96.24% of the data points as of parallel dynamic energy profile than direct measurement and has the same percentage deviations with its mean dynamic energy consumption on average as of the parallel dynamic energy profile. However, it provides a relatively poor maximum percentage deviation with a difference of 23% from parallel dynamic energy profile. In contrast, the Combined$_1$ dynamic energy profile provides the same maximum variations as of parallel dynamic energy profile. All other experiment configurations provide relatively worst accuracy, different application trend, and variations from their mean. However, they require relatively less number of experiments than Combined$_5$ for composing the combined dynamic energy profile.

In conclusion, one can opt for the experiment configuration to compose the combined dynamic energy profile by considering the best suitable combination of the number of experiments, accuracy, application trend, percentage deviations from mean and maximum dynamic energy consumption.

5.6 Workload Types and Ganularity Limitations of AnMoHA

In this section, we study the limits for the topological granularity of a computing platform, and the viability and efficacy of AnMoHA when different workload types (applications) and sizes are executing on their corresponding different independently powered compute devices.

5.6.1 Workload Types and Granularity Limitations

To explore the granularity limitations, we first study the AnMoHA at socket-level and then at the granularity level of CPU cores. If the additive modelling hypothesis holds for socket-level dynamic energy consumption, then we can model and attribute the dynamic energy consumption to an individual application when running two different applications parallel on two sockets.

Socket-Wide Additive Energy Modelling: For this study, we run our experiments on HCLServer01. We formulate the socket-wide abstract processors: *AbsCPU* to measure the dynamic energy consumption by the application kernel running on it. Hence, first abstract processor: *AbsCPU1* contains all 12 cores of socket-1, and the second abstract processor: *AbsCPU2* contains all 12 cores of socket-2.

We use only such configurations of the applications, for our experiments, which execute on AbsCPU and do not use any other system resources such as solid-state drives (SSDs), network interface cards (NIC), GPU, and etc. Therefore, the change in energy consumption of the system reported by HCLWattsUp reflects solely the contributions from CPU socket and DRAM. For our experiments, we use MKL routines of the application kernels of DGEMM and FFT as explained in section 5.3.1. To study the real-time CPU usage scenario, we run three batches of experiments to explore the viability of additive modelling for the following three different case studies:

1. Same application kernel with the same workload sizes on both sockets in parallel.
2. Same application kernel: a) MKL-DGEMM, b) MKL-FFT with different workload sizes (for example, workload N on socket-1 and workload M on socket-2) in parallel. where M $\neq$ N and $M > 0, N > 0$.
3. Two different application kernels (such as MKL-FFT and MKL-DGEMM) in parallel on two sockets.

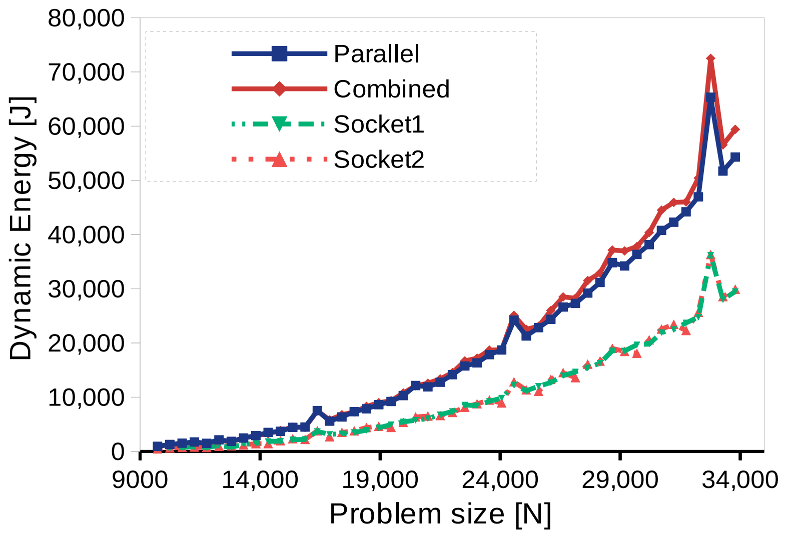


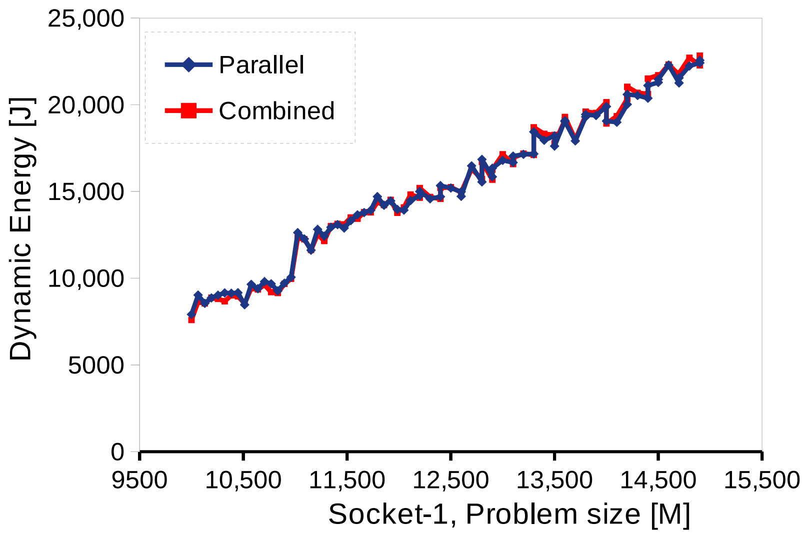
Figure 5.11: Case study A: Socket-wide dynamic energy profiles of same application and same workloads.

Results and Discussion: For our first batch of experiments, we run MKL-DGEMM on both abstract processors each with the same workload size (M $\times$ N) ranging from 9728 $\times$ 9728 to 33792 $\times$ 33792. Fig. 5.11 illustrates the parallel and combined dynamic energy profiles of both application configurations. One can observe that combined dynamic energy is exhibiting the same application trend as of parallel. We find that both sockets consume an equal amount of dynamic energy. This is because they both execute the same workload sizes of the same application kernels. The average and maximum errors between parallel and combined dynamic energy profiles are 4.56% and 10.98%.

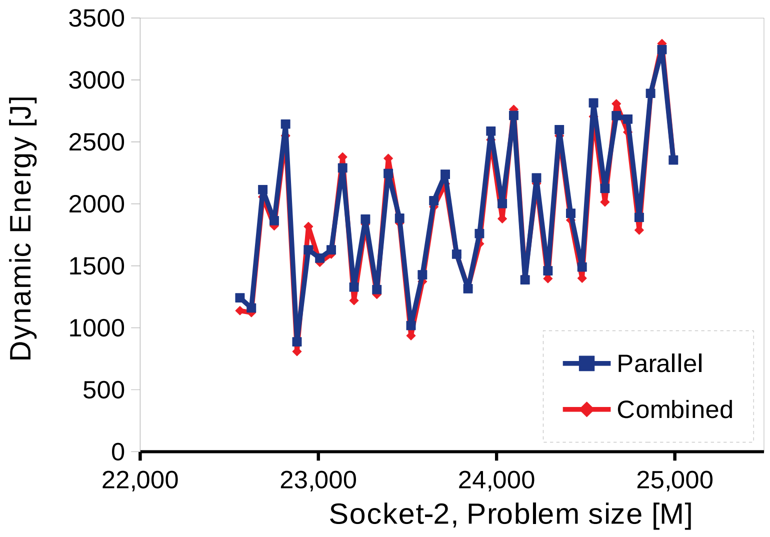
For our next batch of experiments to study the second use case, we run MKL-FFT on both abstract processors each with the different workload size (N $\times$ N). The workload size (N $\times$ N) for AbsCPU1 ranges from 20000 $\times$ 20000 to 22432 $\times$ 22432 with a constant step size of 64. The workload size for AbsCPU2 ranges from 22560 $\times$ 22560 to 24992 $\times$ 24992 with a constant step size of 64. Fig. 5.12(b) illustrates the parallel and combined dynamic energy profiles of both application. The x-axis in the plot shows the problem size range of FFT on AbsCPU2. We find the average and maximum errors between both parallel and combined dynamic energy profiles as 3.7% and 11.56% respectively.

For our batch of experiments to study the second use case, we run MKL-DGEMM on both abstract processors each with a different workload size. The workload size (M $\times$ N) for AbsCPU1 ranges from 7680 $\times$ 30720 to 16896 $\times$ 30720 with a constant step size of 512; and for AbsCPU2, it ranges from 23040 $\times$ 30720 to 32256 $\times$ 30720 with a constant step size of 512. Fig. 5.12(a) illustrates the parallel and combined dynamic energy profiles of both application configurations. The x-axis in the plot shows the problem size range of DGEMM on AbsCPU1 for the dimension M. One can observe that combined dynamic energy profile is exhibiting the same application trend as of parallel. We find the average and maximum errors between parallel and combined dynamic energy profiles to be 1.27% and 5.23%.

For our last batch of experiments to study the third use case, we run MKL-FFT on AbsCPU1 with a workload size (N $\times$ N) ranging from 20000 $\times$ 20000 to 22432 $\times$ 22432 with a constant step



(a) DGEMM. x-axis scale represents the problem size range for DGEMM executed on AbsSoc1



(b) 2D-FFT. x-axis scale represents the problem size range for 2D-FFT executed on AbsSoc2

Figure 5.12: Case study B: Socket-wide dynamic energy profiles of same application, different workloads.

size of 64; and MKL-DGEMM on AbsCPU2 (N $\times$ N) with a workload size ranging from 10000 $\times$ 10000 to 12432 $\times$ 12432 with a constant step size of 64. Fig. 5.13 illustrates the parallel and combined dynamic energy profiles of MKL-FFT and MKL-DGEMM. The x-axis on the plot shows the problem size range of MKL-FFT on AbsCPU1. Similar to previous experiments results, HCLWattsUp combined dynamic energy is exhibiting the same application trend as of HCLWattsUp parallel. We find the average and maximum errors between parallel and combined dynamic energy profiles to be 1.5% and 5%.

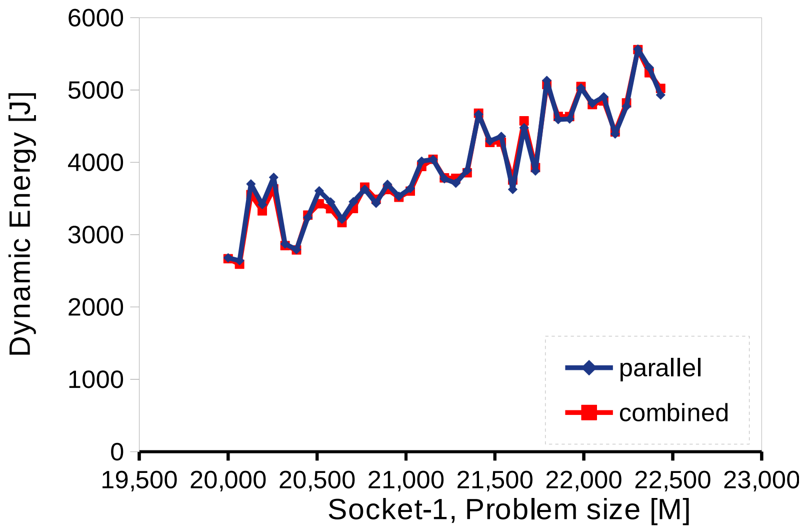


Figure 5.13: Case study C: Different applications. x-axis scale represents the problem size range for DGEMM executed on AbsSoc1.

Summary: To summarize, we find the similar results for all use case scenarios. The combined dynamic energy profiles exhibit the same application trend as of parallel for all case studies with an average error ranges between 1.27% and 4.56%. Table 5.8 presents the percentage errors between the parallel and combined dynamic energy profiles. This suggests that the dynamic energy consumption can be attributed to the individual application using AnMoHA, when two different applications are running in parallel on a dual-socket multicore CPU platform. Furthermore, we can determine and model socket-wide application dynamic energy consumption with additive modelling in an equally effective way as device-wide (CPU, GPU, Xeon Phi as explained in section 5.3). This is because, each CPU socket of Intel Haswell E5-2670V3 is independently powered, and there is minimal resource sharing when two different applications are pinned on individual sockets during their parallel execution. Hence, the abstract processor AbsCPU satisfies both the constraints of AnMoHA: *independently powered* and *loose-coupling* (as explained in section 5.2.1), and that is why we observe that additive hypothesis holds for socket-wide dynamic energy consumption.

5.6.2 State-of-the art Energy Measurement tools

In this section, we compare the socket-wide measurements of dynamic energy consumption by RAPL with HCLWattsUp for aforementioned case-studies. RAPL [41] is explained in section 4.4. Briefly, RAPL is a popular tool to obtain energy consumption by an application running on Intel CPUs. It provides socket-level energy consumptions. To obtain the energy consumption provided by

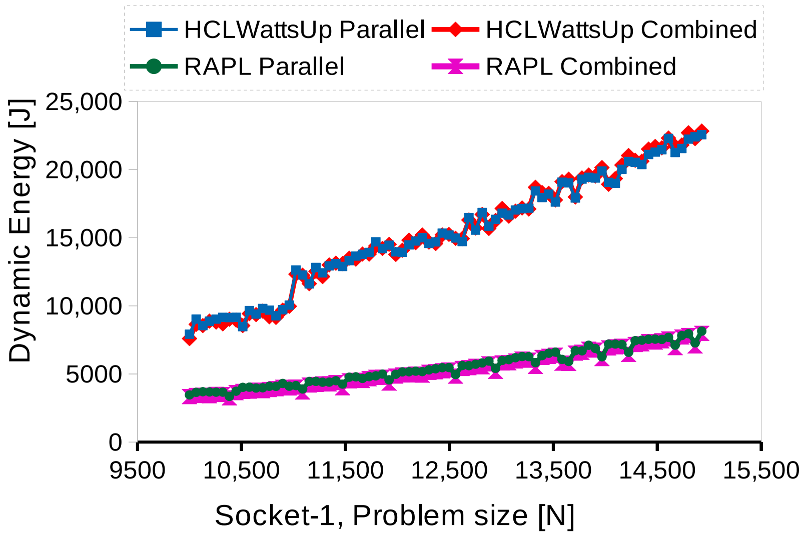
Table 5.8: Percentage errors between socket-wide parallel and combined dynamic energy profiles built with HCLWattsUp.

Experiment Configuration	**Min**	**Avg**	**Max**
Same application (DGEMM), Same workload	0.09%	4.56%	10.98%
Same application (DGEMM), Different workload	0.04%	1.27%	5.23%
Same application (FFT), Different workload	0.06%	3.75%	11.56%
Different applications (DGEMM, FFT)	0.08%	1.46%	4.96%

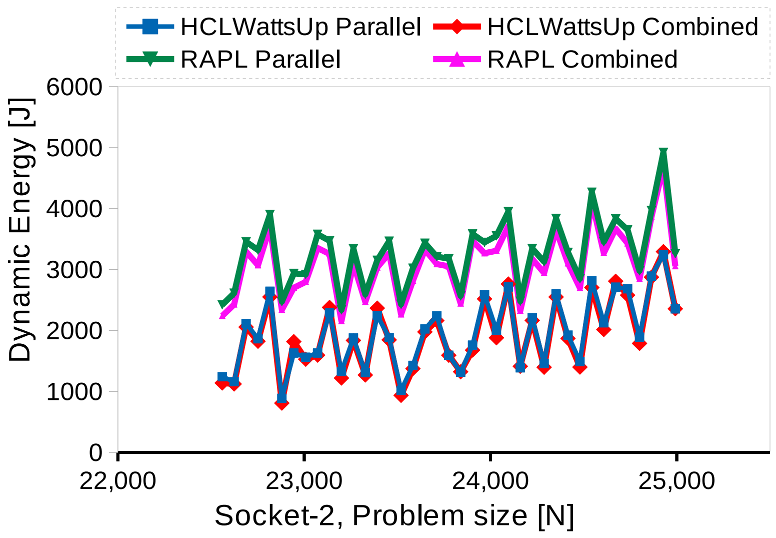
RAPL, we use a well-known package, Intel PCM [105]. We ensure that the RAPL values output by this package is correct by comparing with values given by another well-known package, PAPI [104]. To compare the energy measurements using RAPL against HCLWattsUp, the detailed methodology explained in Chapter 3 and 4.4 is strictly followed. It is important to note here that the execution time of the application kernel is the same for dynamic energy calculations by all tools. So, any difference between the energy readings using these tools comes solely from their power readings.

For our first batch of experiments to study the second use case scenario, we run MKL-DGEMM on both abstract processors each with the different workload sizes (N $\times$ N). The workload size (N $\times$ N) for AbsCPU1 ranges from 10000 $\times$ 10000 to 14928 $\times$ 14928 with a constant step size of 64. The workload size for AbsCPU2 ranges from 15000 $\times$ 15000 to 19928 $\times$ 19928 with a constant step size of 64. Fig. 5.14(a) illustrates the parallel and combined dynamic energy profiles of both applications build using RAPL and HCLWattsUp. The x-axis in the plot shows the problem size range of DGEMM on AbsCPU1. One can observe that HCLWattsUp combined dynamic energy is exhibiting the same application trend as of HCLWattsUp parallel. We find the average and maximum errors between both parallel and combined dynamic energy profiles built using HCLWattsUp to be 1.27% and 5.23% respectively. In contrast, RAPL under-reports the dynamic energy consumption as compared with HCLWattsUp. We find the average and maximum errors between parallel dynamic energy profiles built with RAPL and HCLWattsUp to be 64% and 69% respectively. The average and maximum errors between parallel dynamic energy profile built with HCLWattsUp and combined dynamic energy profile built with RAPL are 65% and 70% respectively. However, we can reduce the average and maximum errors to 18% and 59% respectively between the both profiles by calibrating the RAPL readings.

For our next batch of experiments, we run MKL-FFT on both abstract processors each with the different workload sizes (N $\times$ N). The workload size (N $\times$ N) for AbsCPU1 ranges from 20000 $\times$ 20000 to 22432 $\times$ 22432 with a constant step size of 64. The workload size for AbsCPU2 ranges from 22560 $\times$ 22560 to 24992 $\times$ 24992 with a constant step size of 64. Fig. 5.14(b) illustrates the parallel and combined dynamic energy profiles of both applications build using RAPL and HCLWattsUp. The x-axis in the plot shows the problem size range of MKL-FFT on AbsCPU2. HCLWattsUp combined dynamic energy is exhibiting the same application trend as of the



(a) DGEMM



(b) 2D-FFT

Figure 5.14: Case Study B: Dynamic energy profiles of same application with different workload sizes built with RAPL and HCLWattsUp on HCLServer01.

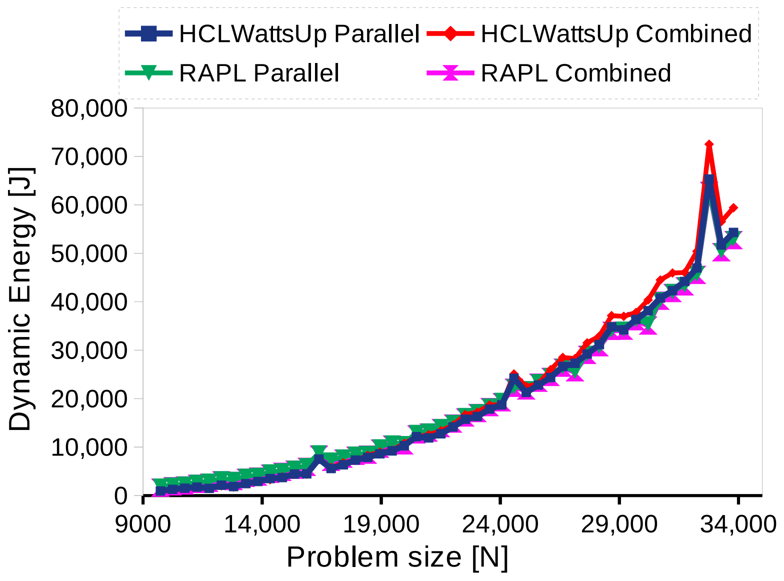
HCLWattsUp parallel. We find the average and maximum errors between both parallel and combined dynamic energy profiles built using HCLWattsUp to be 3.75% and 11.56%. In contrast, RAPL overestimates dynamic energy consumption as compared with HCLWattsUp. We find the average and maximum errors between parallel dynamic energy profiles built with RAPL and HCLWattsUp to be 75% and 178%. The average and maximum errors between parallel dynamic energy profile built with HCLWattsUp and combined dynamic energy profile built with RAPL are 66% and 164%.

For our batch of experiments to study the first use case scenario, we run MKL-DGEMM on both abstract processors each with the same workload size (M $\times$ N) ranging from 9728 $\times$ 9728 to 33792 $\times$ 33792. Fig. 5.15(a) illustrates the parallel and combined dynamic energy profiles of both application configurations. We find that HCLWattsUp combined dynamic energy exhibit the same application trend as of HCLWattsUp parallel for 94% of the data points. In contrast, RAPL parallel and combined dynamic energy profiles exhibit different application behavior with HCLWattsUp parallel profile for 13% of the data points. Consider, for example, the data points (N) {19968,27136,30208} where HCLWattsUp suggests a percentage increase of {12,2,5} in dynamic energy consumption with respect to their corresponding immediate preceding data points. In contrast, RAPL suggests a percentage decrease of {1,4,2} for the same data points. Similarly, HCLWattsUp suggests a percentage decrease in the dynamic energy consumption of the data points such as {11776,20992} by {14,2} with respect to their corresponding immediate preceding data points. However, RAPL suggests a percentage increase of {9,2} for the same data points.

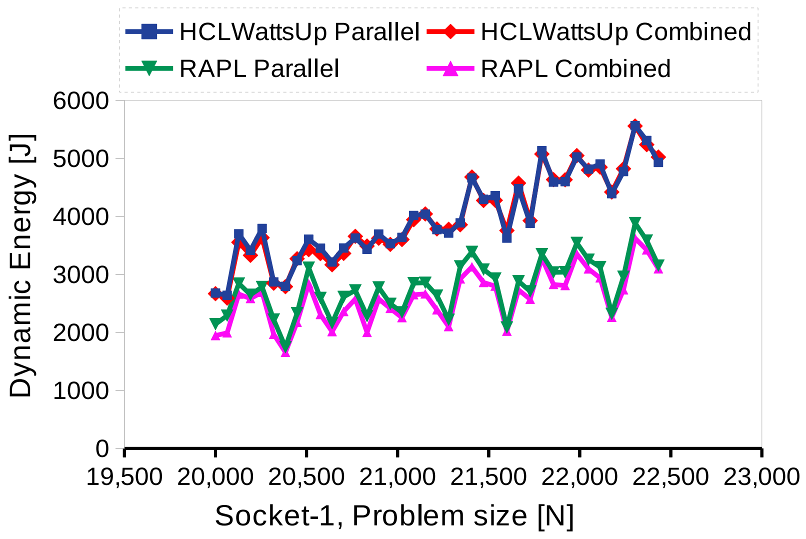
The average and maximum errors between parallel dynamic energy profiles built with RAPL and HCLWattsUp are 21% and 109% respectively. The average and maximum errors between parallel dynamic energy profile built with HCLWattsUp and combined dynamic energy profile built with RAPL are 16% and 77% respectively. However, the average and maximum errors between parallel and combined dynamic energy profiles built with HCLWattsUp are 4.56% and 10.98% respectively. This suggests that the dynamic energy profiles built with HCLWattsUp are more accurate and exhibit similar energy consumption behavior as of ground truth.

For our batch of experiments to study the third use case, we run MKL-FFT on AbsCPU1 with a workload size (N $\times$ N) ranging from 20000 $\times$ 20000 to 22432 $\times$ 22432 with a constant step size of 64; and MKL-DGEMM on AbsCPU2 (N $\times$ N) with a workload size ranging from 10000 $\times$ 10000 to 12432 $\times$ 12432 with a constant step size of 64. Fig. 5.15(b) illustrates the parallel and combined dynamic energy profiles of MKL-FFT and MKL-DGEMM. The x-axis on the plot shows the problem size range of MKL-FFT on AbsCPU1. Similar to previous experiments results, HCLWattsUp combined dynamic energy is exhibiting the same application trend as of HCLWattsUp parallel. We find the average and maximum errors between parallel dynamic energy profiles built with RAPL and HCLWattsUp to be 30% and 47% respectively. The average and maximum errors between parallel dynamic energy profile built with HCLWattsUp and combined dynamic energy profile built with RAPL are 35% and 49% respectively.

Another interesting finding is that we find a strong positive correlation between RAPL and HCLWattsUp energy readings for all case studies. The Pearson correlation coefficient between RAPL and HCLWattsUp parallel dynamic energy profiles for first case study is 0.99. However, both profiles disagree on energy consumption behavior for 13% of the data points. Similarly, the correlation coefficient between RAPL and HCLWattsUp parallel dynamic energy profiles for third use case



(a) Case Study A: Same Application (DGEMM) with same workload size.



(b) Case Study C: Different applications

Figure 5.15: Socket-wide dynamic energy profiles built with RAPL and HCLWattsUp on HCLServer01.

scenario is 0.86, but both profiles disagree on energy consumption behavior for 11% of the data points. Interestingly, the correlation coefficient is 0.89 between RAPL combined dynamic profile and HCLWattsUp prallel profile for the same case study. However, both profiles exhibit different application trend for more than 13% of the data points. Similarly, the correlation coefficient between RAPL combined and HCLWattsUp parallel dynamic energy profiles is 0.9 for second use case when running MKL-FFT with different workload sizes on both sockets. However, both profiles exhibit different energy consumption trend for more than 18% of the data points. This all suggests that two energy profiles can exhibit different trend for a range of data points even if they have a strong positive correlation between them. Hence, the correlation coefficient alone is not sufficient enough for comparing the similarity between energy profiles.

Similarly, there exists a strong positive correlation between RAPL and HCLWattsUp energy readings (the Pearson correlation coefficient between them is 0.97) for MKL-DGEMM with different workload sizes. However, both profiles disagree on energy consumption behavior for more than 48% of the data points. Consider, for example, the data points (N) {11152,11984,12624,13712} where HCLWattsUp suggests a percentage decrease of {5,3,2,6} in dynamic energy consumption with respect to their corresponding immediate preceding data points. In contrast, RAPL suggests a percentage increase of {14,9,13,13} for the same data points. Similarly, HCLWattsUp suggests a percentage increase in the dynamic energy consumption of the data points {12944,13328,13584,13968} by {3,7,8,3} with respect to their corresponding preceding data points. However, RAPL suggests a percentage decrease of {9,7,8,8} for the same data points.

Furthermore, the orientation (i.e. the overall energy consumption trend) of both profiles is also different and the divergence between the both profiles increases with an increase in problem sizes. It shows that on-chip power sensors do not capture the holistic picture of the energy consumption trend when running two applications in parallel each on a different socket. Owing to the nature of the deviations of the energy measurements provided by the RAPL from the ground truth, calibration can not improve the qualitative difference of the energy profiles build with on-chip sensors to an extent that can allow them to be used in optimization of applications for dynamic energy. This is because the inaccurate energy measurements can cause a significant loss of energy when employed for the energy optimization of an application [37].

While the average error between the both profiles constructed with RAPL and HCLWattsUp can be reduced by calibrating the RAPL readings, the overall qualitative difference of energy consumption behavior of both profiles can not be improved. One can observe in figure 5.16 that the overall energy consumption trend of both profiles remains different even after calibrating the RAPL readings. It suggests that the correlation coefficient and average error are not sufficient enough for comparing the similarity between energy profiles.

5.6.3 Core-wide dynamic energy consumption modelling

In this section, we explore if we can further fine the topological granularity to core-wide energy measurements using system-level measurements. For this study, we explore different core-wide combinations (such as 1-core, 2-core, etc) on HCLServer01 and HCLServer02 (technical details are provided in tables 4.1 and 4.2 respectively) to formulate the abstract processors *(AbsCore)* for measuring the dynamic energy consumption by the application running on them. We follow the

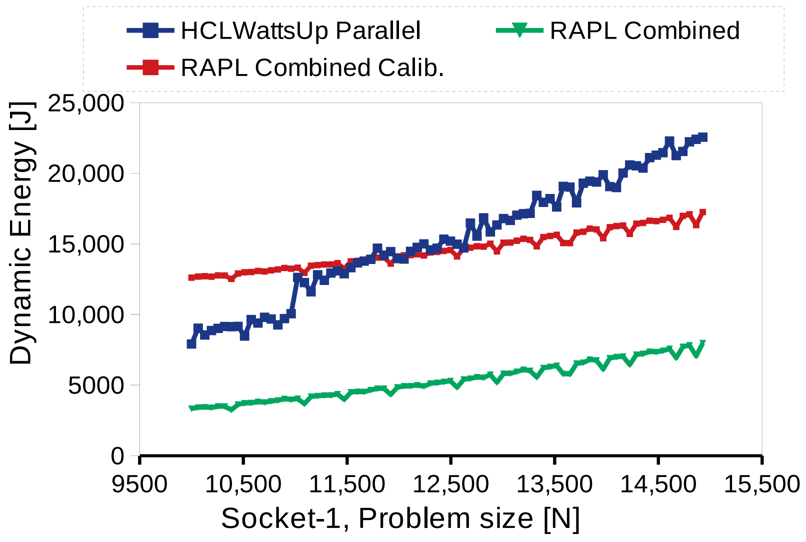


Figure 5.16: Dynamic energy profiles of same application with different workload sizes built with RAPL and HCLWattsUp on HCLServer01 with calibrated RAPL readings.

same methodology to ensure the reliability of our experiment results that we use for socket-wide modelling.

For all our experiment sets, we find that combined dynamic energy consumption is relatively higher than parallel dynamic energy consumption. However, the difference is not the same across the workload sizes or AbsCore configurations. Consider, for example, the problem size 13312 $\times$ 13312 where each AbsCore contains 11 cores on HCLServer02. The dynamic energy consumption when running parallel both kernels is 1982 joules whereas the combined dynamic energy consumption is 3115.87 joules which is 57% higher than the parallel one. Now, consider another problem size 10752 $\times$ 10752 for the same AbsCore formulation. The dynamic energy consumption by the parallel executing kernels is 736 joules whereas the combined dynamic energy is 1887.28 joules which is 157% higher than the parallel one. We find the similar results for other combinations of CPU cores for example where each AbsCore contains one, two or more number of CPU cores. The similar results are also reported for HCLServer01.

To discuss the reasons for this energy consumption behavior of CPU cores, we present a brief on power consumption by CMOS and advance power mechanisms to control and configure the power consumption in modern computing hardware as discussed in sections 2.1.4 and 2.4.1.

In section 2.4.1, we discussed that the voltage change of the components are supported through voltage regulators (VRs) [139]. Briefly, a VR performs two following major functions in providing voltage to a computing device: i) it stabilizes the supplied voltage, and ii) it changes the supplied voltage according to the needs of the device. It is important to note that the same voltage is supplied to all components sharing the same voltage domain. The voltage domain can be defined as a set of components that is attached to a common supply voltage. The components which share the same voltage domain cannot regulate their voltage independently from each other. Therefore, only frequency can be scaled dynamically for the components sharing the same voltage domain. Hence, all the CPU cores belonging to the same voltage domain share the same voltage.

The advancements of power saving mechanisms such as Advanced Configuration and Power Interface (ACPI), allow the OS to change the clock frequency of the components. For example, CPU frequency governors [138] allows one to dynamically change the CPU frequency. The voltage input also scales up or down in accordance with the change in clock frequency of CPU cores. The usual settings of CPU frequency governor (such as *Ondemand* or *conservative* governors in Linux) sets the clock frequency of the cores depending on the current system load which also affects the voltage supply as a consequence. The system, under these settings, keeps the voltage and clock frequency of cores at low when idle, and scale them up as soon as some workload is there for execution. It is important to note here while frequency can be scaled up ot down at core-wide, the CPU cores cannot regulate their voltages independent of each other because the same voltage is supplied across all the cores sharing the same voltage domain.

Total power consumption by a CMOS can be roughly considered as the sum value of idle power and dynamic power as discussed in section 2.1.4. The idle power is the (leakage) power dissipation when it is not running the application. While the dynamic power part is proportional to the square of the voltage, the (leakage or idle) power scales exponentially with voltage [139].

The package C-state (power state) is active when there is at least one CPU core state is active as discussed in section 2.4.1. It consumes more power in active C-state (C0) than other sleep C-states (i.e. C1 through Cn). Therefore, the voltage of entire package is scaled up as a result of scaling up in clock frequency of active CPU cores when a workload is executed on some (or even one) of the CPU cores of a socket. However, the idle fraction of the CPU cores of the socket dissipates its power as leakage. Because of relatively higher voltage supply, this power dissipation by idle CPU cores is higher than the power dissipation when the voltage supply was low. The dynamic energy consumption by the application running sequentially on core-wide abstract processors also includes the idle (static) energy contributed by the idle cores. As a result, we observe a higher combined dynamic energy consumption than the parallel one.

This suggests that power dissipation by the idle cores contribute significantly to the total power consumption by the CPU when running an application on some of the CPU cores of a socket. Hence, we can optimize dynamic energy consumption by the socket by switching the idle cores off. However, it can introduce an overhead. Alternatively, we can execute the application on all of its cores, but it may introduce diminishing returns for some workload types. We leave this study for the future.

In conclusion, we can not model the dynamic energy consumption by the computing elements with system-level measurements using external power meters, which are tightly coupled or which are not independently powered (thus violating the constraints of AnMoHA as explained in section 5.2.1).

5.7 Study of additive energy modelling and dynamic energy optimization with On-chip sensors and HCLWattsUp

First, in this section, we study the additive energy modelling with on-chip built-in sensors for hybrid applications and analyze its accuracy against HCLWattsUp (which we consider as ground truth). Then, we study the dynamic energy optimization of a 2D-FFThybrid application and DGEMM with both aforementioned tools and demonstrate that we can lose significant energy by using inaccurate energy measurements provided by on-chip sensors for dynamic energy optimization.

5.7.1 Additive energy modelling with on-chip sensors

For this study, we use RAPL to obtain the power consumption by CPU, Nvidia NVML [43] to acquire the power values from on-chip sensors on Nvidia GPUs, and Intel SMC [42] to obtain the power values from Xeon Phi that can be programmatically obtained using Intel MPSS [102]. All the experiments are executed on HCLServer01 (technical details are provided in table 4.1). For illustration purposes, we refer these on-chip sensors collectively as *sensors* for the rest of this study.

We run 2D-FFT for the workload sizes ranging from 15104×23552 to 18688×23552 with a constant step size of 64. We build the dynamic energy profile when executing the application kernels in parallel on their respective abstract processors, and call it *parallel* dynamic energy profile with HCLWattsUp. Then, we build dynamic energy functions for each abstract processor executing the 2D-FFT individually with on-chip sensors and HCLWattsUp separately and compose the combined dynamic energy profiles using the equation 5.2, and call them *sensors combined* and *HCLWattsUp combined*. Figure 5.17 shows the dynamic energy profile of 2D-FFT with sensors and HCLWattsUp.

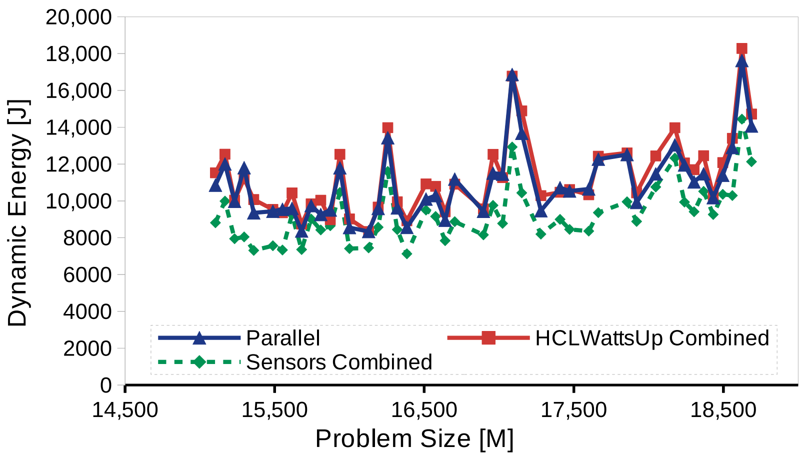


Figure 5.17: Dynamic energy profiles of 2D-FFT on HCLServer01.

One can observe that sensors combined dynamic energy profile lags behind the parallel dynamic energy profile and has a higher error rate with the ground truth than the HCLWattsUp combined. The average and maximum errors of sensors combined dynamic energy profile and HCLWattsUp profile with parallel dynamic energy profile are {15.1%, 31.87%} and {4.34%, 8.91%} respectively. We find that sensor combined profile in comparison with HCLWattsUp combined profile exhibits relatively poor similarity of variations on average and maximum as of parallel dynamic energy profile. The absolute percentage error between the average and maximum deviations of HCLWattsUp combined dynamic energy profile and parallel dynamic energy profiles is {4.97%, 0.64%}, and between sensors combined dynamic energy profile and parallel dynamic energy profiles is {5.67%, 5.3%}. Table 5.9 provides the percentage deviations from mean and maximum dynamic energy consumption by parallel and both combined dynamic energy profiles.

For our next batch of experiments, we run DGEMM for the workload sizes ranging from 12800 $\times$ 20480 to 20480 $\times$ 20480 with a constant step size of 256. Figure 5.18 shows the dynamic energy profile of DGEMM with sensors and HCLWattsUp.

One can observe that sensors combined dynamic energy profile is lagging behind the parallel dynamic energy profile and has a higher difference with it than the HCLWattsUp combined. The av-

Table 5.9: Percentage deviations from mean and maximum of dynamic energy consumption by parallel and combined profiles of 2D-FFT (composed with Sensors and HCLWattsUp) on HCLServer01.

		Combined	
	Parallel	**HCLWattsUp**	**Sensors**
avg	13.72%	14.4%	12.94%
max	37.71%	37.95%	35.71%

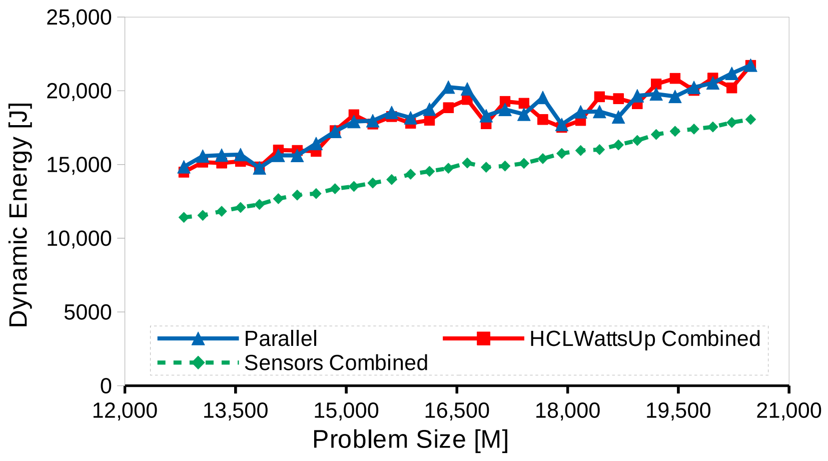


Figure 5.18: Dynamic energy profiles of DGEMM on HCLServer01.

erage and maximum errors between the combined dynamic energy profiles composed with sensors and the parallel dynamic energy profile with HCLWattsUp are {19%,27.18%} respectively. However, the average and maximum error between the parallel and combined dynamic energy profiles with HCLWattsUp are {3.07%, 7.63%} respectively.

We find that sensor combined profile in comparison with HCLWattsUp combined profile exhibits relatively poor similarity of variations on average and maximum as of parallel dynamic energy profile. The absolute percentage error between the average and maximum percentage deviations of HCLWattsUp combined dynamic energy profile and parallel dynamic energy profiles is {5.44%, 2.86%}, and between the sensors combined dynamic energy profile and parallel dynamic energy profiles is {33.08%, 12.46%}. Table 5.10 provides the percentage deviations from mean and maximum dynamic energy consumption by parallel and both combined dynamic energy profiles.

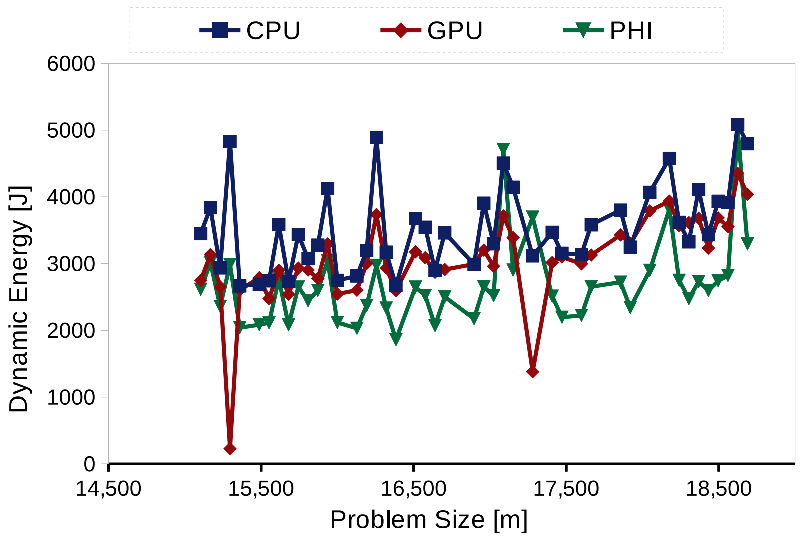
To summarize, we find that the additive energy models with sensors provide poor accuracy and do not follow the energy consumption trend of the ground truth. In contrast, HCLWattsUp combined dynamic energy profile is more accurate and follows the energy consumption trend of the ground truth. In next section, we study the consequences of using inaccurate energy profiles constructed

Table 5.10: Percentage deviations from mean and maximum of dynamic energy consumption by parallel and combined profiles of DGEMM (composed with Sensors and HCLWattsUp) on HCLServer01.

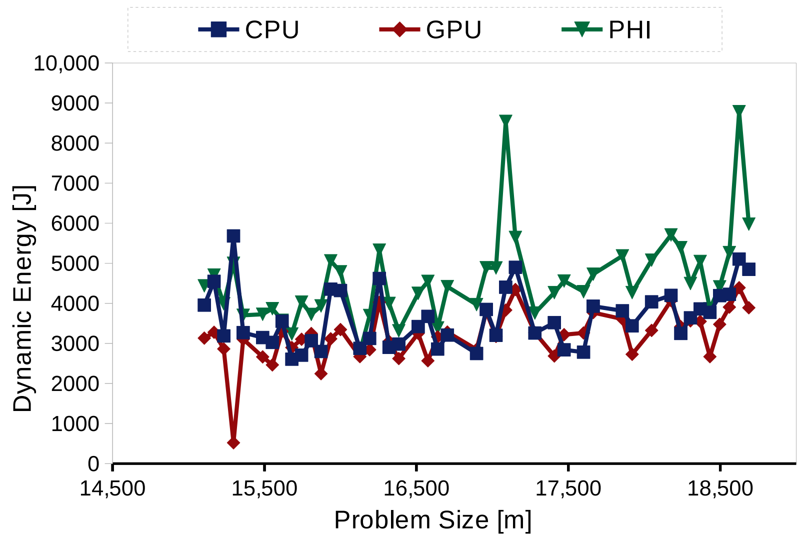
		Combined	
	Parallel	**HCLWattsUp**	**Sensors**
avg	8.27%	8.72%	11%
max	16.31%	16.77%	18.34%

with on-chip power sensors for the energy optimization purposes of an application.

5.7.2 A study of dynamic energy optimization with on-chip sensors and HCLWattsUp



(a) with Sensors



(b) with HCLWattsUp

Figure 5.19: Dynamic energy profiles of 2D-FFT on HCLServer01.

In this section , we study the optimization of two hybrid applications i) 2D-FFT, and ii) DGEMM for dynamic energy measured with (on-chip built-in) sensors and system-level physical measurements using HCLWattsUp.

Dynamic Energy Optimization of 2D-FFT: We run the parallel hybrid application 2D-FFT as explained in section 5.3.1 on HCLServer01 for the problem sizes ranging from 45312 $\times$ 23552 to 56064 $\times$ 23552 with a constant step size of 192. There is no communication involved in these experiments. Figures 5.19(a) and 5.19(b) illustrate the dynamic energy profiles for workload sizes (m) with sensors and HCLWattsUp. One can observe that in comparison with HCLWattsUp, sensors

Table 5.11: Percentage error of sensors against HCLWattsUp for dynamic energy consumption by 2D-FFT.

abstract processor	Min	Max	Avg
CPU1	0.25%	36.37%	8.25%
GPU1	0.52%	57.78%	11.2%
PHI1	1.64%	55.78%	40.87%

under-report the dynamic energy consumption for the range of all problem sizes. Table 5.11 provides the percentage error of dynamic energy measurements of with sensors against HCLWattsUp on each abstract processor. The average errors are {8.25%,11.2%,40.87%} for CPU1, GPU1 and PHI1 respectively.

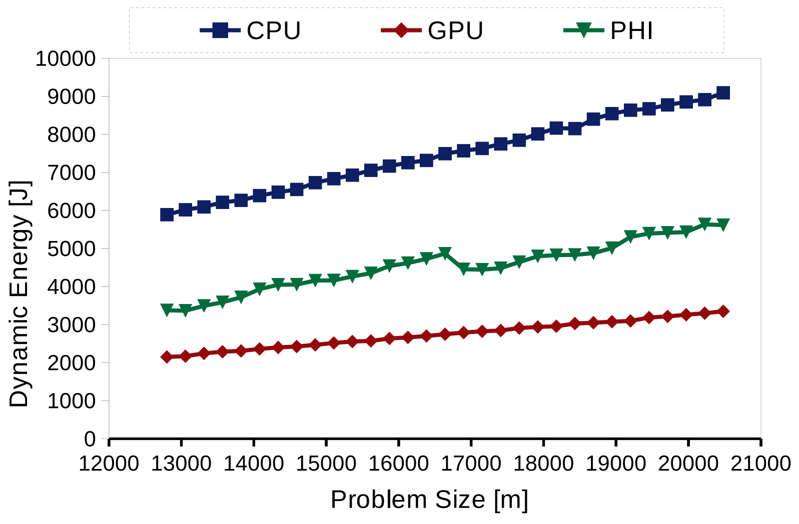
We equally partition the dimension M on all three abstract processors into M_1, M_2 and M_3 such that the 2D Fourier Transforms of signal matrix $M_1 \times N$, $M_2 \times N$ and $M_3 \times N$ are computed by abstract processor CPU1, GPU1, and PHI1. We use a model-based data partitioning algorithm [20] to compute the decomposition of dimension M. The algorithm takes the following inputs: i). the workload size, ii). number of abstract processors, iii) cardinality of energy profiles, iv) the functions of execution time, and v) the discrete dynamic energy profiles of the abstract processors: $\{E_{CPU1}, E_{GPU1}, E_{PHI1}\}$. The output is the optimal workload partitioning allocated to abstract processors: $(m_{CPU1}, m_{GPU1}, m_{PHI1})$. One or more abstract processors may be allocated the workload of size zero. More details on the algorithm and its complexity can be found in [20].

The discrete dynamic energy consumption function of abstract processor AP_i is given by $DE_i = \{e_i(m_1, n_1), ..., e_i(m_x, n_y)\}$ where $e_i(m, n)$ represents the dynamic energy consumption during the Fourier transform of sizes $m \times n$ by the abstract processor i. The dimension n is fixed as 23552, and the dimension m ranges from 15104 to 18688 with a constant step size of 64 for each AP_i.

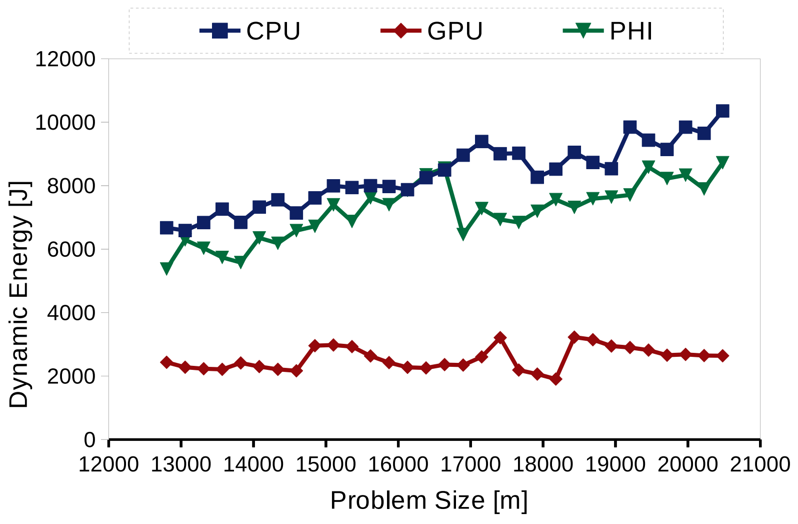
We determine the workload distribution for workload sizes where the dimension M is {46656,46848,48768,52800,53568,53760,54528} using the dynamic energy profiles with sensors and HCLWattsUp as an input to the data partitioning algorithm [20]. For each workload distribution, we run the application in parallel on all abstract processors and determine the dynamic energy consumption with sensors and HCLWattsUp separately. We find the total dynamic energy losses by using sensors in comparison with HCLWattsUp for the aforementioned workload sizes is {42%,39%,45%,38%,37%,38%,36%} respectively.

Dynamic Energy Optimization of DGEMM: We run a parallel hybrid application DGEMM on HCLServer01 for the problem sizes ranging from 38400 $\times$ 20480 to 61440 $\times$ 20480 with a constant step size of 768. Figures 5.20(a) and 5.20(b) illustrate the dynamic energy profiles for workload sizes (m) with sensors and HCLWattsUp. One can observe that in comparison with HCLWattsUp, sensors under-report the dynamic energy consumption for the range of all problem sizes. Table 5.12 provides the percentage difference of measurements of dynamic energy consumption by DGEMM with sensors against HCLWattsUp on each abstract processor. The average errors are {10.25%,14.62%,38.06%} for CPU1, GPU1 and PHI1 respectively. There is no communication involved in these experiments.

We equally partition the dimension M of matrix A on all three abstract processors into M_1, M_2 and M_3, so that the matrix $M_1 \times N$, $M_2 \times N$ and $M_3 \times N$ are computed by abstract processor CPU1, GPU1, and PHI1 respectively. We use the same aforementioned model-based data parti-



(a) with Sensors



(b) with HCLWattsUp

Figure 5.20: Dynamic energy profiles of DGEMM on HCLServer01.

Table 5.12: Percentage difference of sensors against HCLWattsUp for dynamic energy consumption by DGEMM.

abstract processor	Min	Avg	Max
CPU1	0.12%	10.25%	18.74%
GPU1	0.5%	14.62%	55.22%
PHI1	31.02%	38.06%	46.53%

tioning algorithm [20] to compute the decomposition of dimension M. The discrete dynamic energy consumption function of abstract processor AP_i is given by $DE_i = \{e_i(m_1, n_1), ..., e_i(m_x, n_y)\}$ where $e_i(m, n)$ represents the dynamic energy consumption during the matrix multiplication of sizes $m \times n$ by the abstract processor i. The dimension n is fixed as 20480, and the dimension m ranges from 12800 to 20480 with a constant step size of 256 for each AP_i.

The data partitioning algorithm takes the dynamic energy functional models as an input and finds the optimal workload configuration which optimizes the total dynamic energy consumption for the given application using load imbalance technique. We determine the workload distribution for the workload sizes where the dimension M is {40704,41472,42240,43008,44544} using the dynamic energy profiles with sensors and HCLWattsUp as an input to the data partitioning algorithm. For each workload distribution, we run the application in parallel on all abstract processors and determine its dynamic energy consumption with sensors and HCLWattsUp separately. We find the total dynamic energy losses by using sensors in comparison with HCLWattsUp for the aforementioned workload sizes as {22%,24%,21%,22%,24%} respectively.

5.8 Scope and Limitations of AnMoHA

This section covers the scope, viability and limitations of AnMoHA.

Constraints: AnMoHA is the first solution method to effectively address the challenge of how to model accurately the energy consumption of application components when executing them in parallel on several independently powered compute devices such as CPUs, GPUs, Xeon Phis, and sockets of multi-socket CPUs. It is proven to work accurately and efficiently for the case of loosely coupled kernels where the additive hypothesis is satisfied within a given user tolerance. AnMoHA is based on following two essential constraints (as discussed in sections 5.2.1):

1. **Independently powered:** For a given platform, the abstract processors should be independently powered such that no two abstract processors share the same power domain.

2. **Loose coupling:** For a given hybrid application, the constituent components (kernels) are independent such that a component does not interfere the execution of other components during their parallel execution on their corresponding abstract processors. That is, if we run two applications A and B in parallel on two abstract processors AP_a and AP_b, then the dynamic energy consumption by A executing on AP_a is not affected by the application B executing in parallel on abstract processor AP_b.

Input types for AnMoHA: AnMoHAdoes not make any assumption about the nature of applications or the type of workloads and their sizes executed by their corresponding compute devices. The workload types or the applications can be of any type such as compute-intensive, communication-intensive, etc as demonstrated in section 5.3. Different compute devices can execute different types and sizes of workloads/applications in parallel as demonstrated in section 5.6. One can accurately and effectively compute their individual energy consumption using (AnMoHA) as long as they satisfy the aforementioned constraints of AnMoHA.

Testing an application for its amenability for AnMoHA: AnMoHA takes a hybrid parallel application as an input and builds the individual energy profiles for its constituent components. The hybrid application can be a set of data-parallel or task-parallel component profiles or a mix of both. However, it must satisfy the constraint *loose-coupling*. By default, it is an integral part of the hybrid application design. Nevertheless, it can be learned empirically whether the application components are loosely coupled or not by employing different (simple) approaches such as:

1. To test an application for its amenability for AnMoHA, one can compare the parallel execution time of the application components with the serial execution times of the same components of a hybrid application. In case of independent components or loose-coupling, the parallel execution time of a hybrid application is equal to the maximum of the serial execution times of its constituent components (as discussed in sections 5.3.3, 5.4.1, 5.5.1, and 5.7.1). To test the amenability for AnMoHA, a dry run of the application components using HCLWattsUp [38] can be done to compare their (parallel and serial) execution times. A longer parallel execution time indicates the tight-coupling and race conditions between the application components.

2. It can also be learned by determining the communication between the application components. There is no communication involved when executing the independent application components in parallel on independently powered compute devices as discussed in sections 4.7, 5.6.2 and 5.7.

The process of profile generation: Once it is determined that the application components are loosely coupled, the process of generation of profiles is quite natural and can be automated by writing a script. The application components are mapped to their corresponding compute devices to formulate an *abstract processor* by following the constraint called as *independently powered*. We explain the formulation of abstract processors to build the individual profiles of application components in sections 5.2.2 and 5.3.2.

A unifying framework can also be developed which takes the (hybrid) application as an input; builds the (component) profiles; determines the goodness of profiles (using the approach proposed in [77]); map the profiles to all identical devices in the cluster; and then passes the (mapped) profiles to an energy optimization algorithm such as [30] or a multi-objective optimization workload partitioning algorithm such as [20]. The key components of this framework are AnMoHA the goodness measuring approach [77] (which is explained in Chapter 6). Once these components are in place, the framework can be easily developed. However, we leave the development of such a framework as a possible extension of this thesis.

The applications of AnMoHA: The application component profiles are inputted to an (energy/multi-objective) optimization algorithmx. In other words, the optimization method employs AnMoHA to build the component profiles accurately and efficiently offline or at run-time and employs them for energy optimization of the hybrid parallel application. The references [20] [30] [39] employ

AnMoHA to build the individual profiles of the components of a hybrid parallel application to optimize its energy and also to study the trade-off between performance and energy optimization of such hybrid parallel applications. AnMoHA can also be employed for energy-aware scheduling of virtual machines deployment and resource allocations. Likewise the application components, the virtual machines utilize the resources of the same physical machine. A scheduler can employ AnMoHA and an optimization method to efficiently schedule these virtual machines. However, we leave this study as a possible extension of this thesis.

The granular limitations: One can build as fine-grain additive energy profiles as the level of granularity provided by the tool which is used to measure the power consumption as discussed in section 5.6. For example, it is not possible currently to determine the dynamic energy consumption by the computing elements which are tightly coupled or which are not independently powered, using system-level measurements provided by external power meters. Likewise, the popular tools such as RAPL provide also the socket-wide power consumption details only and do not provide core-wide power consumption. Therefore, it is not possible at present, to build the additive energy models of the computing elements which are tightly coupled or which are not independently powered (such as the CPU cores) using AnMoHA. In such scenarios, AnMoHA can be erroneous due to the violation of both aforementioned constraints by a large extent as discussed in section 5.6.3.

Scalability of AnMoHA on large clusters: One can use a remapping approach to build the energy profile of a large cluster by constructing the energy profiles of a small number of devices (which can be done offline) and then apply these profiles for all nodes of the same type in the cluster. To illustrate this, consider a cluster of n identical nodes (servers) sharing the same software and hardware configurations. Let each node contain m heterogeneous compute devices. There are m application components executing in parallel on m compute devices of a node. Then, only m additive energy profiles are needed to be constructed on one node using AnMoHA. The additive energy models constructed for that particular node can then be reused for the other $n-1$ identical nodes. The profile based workload partitioning algorithm [20] takes the $n \times m$ discrete energy profiles as an input to determine the optimal distribution of the workload amongst the $n \times m$ processors such that the energy consumption by the application is minimized.

Energy measurement tools for AnMoHA: One can use any energy measurement tool to determine the energy consumption by the application component in order to construct the additive energy models using AnMoHA. However, we recommend using the system-level power measurements provided by external power meters because of their accuracy as discussed in section 4.9. In case of a large cluster, only p heterogeneous nodes are required to be equipped with external power meters for building the additive energy profiles. The same energy profiles can then be mapped to the nodes identical to them in the cluster. However, one can also use other state-of-the-art approaches (such as integrated power sensors or PMC based energy predictive models) to measure the energy consumption by the application components. AnMoHA can be employed in equally effective way with such state-of-the-art techniques to build the additive energy models. But, such additive models will not be accurate due to the inherent inaccuracy of the underlying measurement approaches employed to construct them (as discussed in Chapter 4, and sections 5.6 and 5.7.1).

5.9 Summary

In this work, we presented a novel methodology called ***A**dditive e**n**ergy **Mo**delling of **H**ybrid **A**pplications* (AnMoHA) to addresses the following challenges: i). Accurate modelling of the energy consumption of application components when executing in parallel on multiple compute devices, ii). Accurate modelling of the energy consumption of different applications executing in parallel on a multi-socket multi-core CPU platform. AnMoHA is an additive modelling approach that constructs the discrete dynamic energy profiles of the application components using system-level physical power measurements using power meters and satisfying a user-specified precision setting.

We experimentally validated AnMoHA on a cluster of two hybrid heterogeneous computing nodes using three highly optimized parallel applications, matrix-matrix multiplication, 2D fast Fourier transform, and a gene sequencing application for a diverse range of problem sizes. The estimation accuracy of the method ranges between 2% and 5%. We analyze and compare different experiment configurations (the number of independent experiments) to build the additive dynamic energy model of an application kernel, and their accuracy. We show that the additive models constructed with direct energy measurements during the application run provides the best accuracy and require the least number of experiments.

We demonstrated that AnMoHA takes less time to construct the energy profiles when precision settings is reduced. A usability test is introduced to explore the trade-off between the accuracy and time needed to build additive energy models of a hybrid application. We demonstrate that an additive model with a relaxed precision setting of the experiment takes less time to build and can be used for all pragmatic purposes (such as energy optimization) only if it qualifies the usability test. Furthermore, it is found that the time to build the energy profile of an application can significantly be reduced by slightly compromising the accuracy.

We explore the hardware topological granular limitations of AnMoHA and show that we cannot model the dynamic energy consumption by the components which are tightly coupled or not independently powered, which violate the conditions of additive modelling. An important finding is that the base energy (or the energy due to leakage power) of idle CPU cores contributes significantly to the total energy consumption by the CPU, if an application is executed on some of the CPU cores of a socket. We also find in some cases that it can even exceed the dynamic energy consumption by the active CPU cores.

We compare the accuracy of additive energy models constructed using state-of-the-art on-chip power sensors with system-level physical measurements using external power meters (which we consider to be the ground truth). The average error of the models built with on-chip power sensors ranges from 15% to 75% and the maximum reaches 178%. Furthermore, we demonstrate that the correlation between the dynamic energy profiles is not sufficient enough to compare the similarity between the models and ground truth. A model having a strong positive correlation with ground truth can exhibit a different energy consumption behavior for 48% of the data points.

We find that a significant loss of energy (up to 45% for the applications used in our experiments) occurs when employing state-of-the-art estimation methods instead of our proposed method for dynamic energy optimization of an application. Finally, we discuss different aspects of AnMoHA including its scope, limitations, the type of input applications, the scalability on large clusters and its efficacy for energy optimization of hybrid parallel applications.

Chapter 6

A Statistical Learning Based Novel Similarity Measuring Methodology for Energy Profiles of Parallel Applications

This chapter is mainly based on [77].

6.1 Introduction

Accurate energy profiles as functions of the workload are essential to the optimization of parallel applications for energy through workload distribution [30]. There are many model-based methods for efficient construction of energy profiles but none of them is accurate in all situations. Therefore, to pick the best method in a given situation, we need a way to measure the goodness of energy profiles produced by different methods when the ground-truth profile, built by a time-consuming and expensive but reliable method, is available. We define the *goodness* as the accuracy of a profile against the ground truth profile. Here, the ground-truth refers to the baseline profile or the reference value for the comparison. Inaccurate energy measurements during an application run can hamper the efforts for energy-efficient computing. Studies [21] [37] show that inaccurate energy measurements for dynamic energy optimization can result in significant energy losses. In [37], the authors report a loss of up to 84% when using state-of-the-art but inaccurate energy models as an input to an optimization algorithm to minimize the energy consumption by an application.

There is no effective metric to measure the goodness of energy models. Pearson correlation coefficient and average prediction error are the most commonly used statistical measures to determine the accuracy of energy models against the ground truth. However, both techniques have their shortcomings to determine the goodness of energy models. In Chapter 5, we highlight the inadequacies of average prediction error and correlation coefficient to measure the goodness of energy models with the ground truth. In section 6.3, a detailed study is presented to highlight the challenges to energy optimization using either of the aforementioned approaches to measure the accuracy of energy models against the ground truth. In general, both popular statistical techniques are highly sensitive to outliers and rely on the assumption of linear or smooth increase of energy consumption by applications with the increase of workload. However, the energy profiles of applications on modern multicore platforms are highly non-smooth and non-linear. Therefore, the existing statistical measures can rank an inaccurate energy profile higher than accurate ones. The reason is two-fold. First, in the presence of significant variations in the energy profiles, they do not capture the difference in the general trend of energy consumption. Second, they do not capture the similarities in variations.

While the general direction of energy profiles of applications on multicore platforms is reported as a near-linear increasing function of workload size, the shape of the profile can be highly non-linear and non-smooth [18]. We distinguish the terms *trend* and *shape* using the following example. Consider the sample energy profiles shown in figure 6.1. The general direction of all three profiles, which represents the underlying energy consumption trend, is increasing with the increase in workload. However, their shapes are different. The energy profile *Model1* is linear whereas the shapes of *Real* and *Model2* are non-linear and non-smooth.

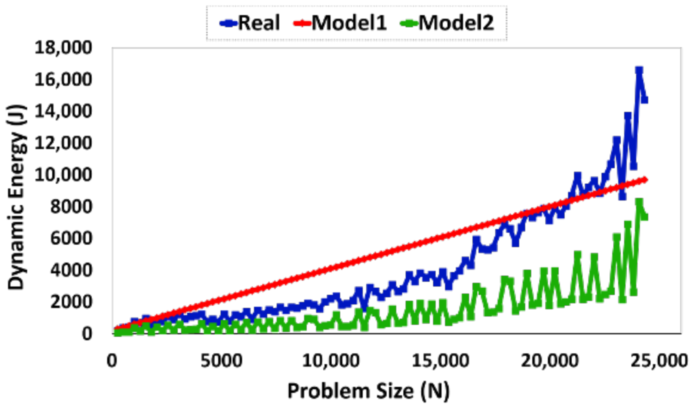


Figure 6.1: Sample dynamic energy consumption profiles.

While the goodness measuring problem is comparatively less-studied for the energy of computing, a plethora of different methods and approaches have been proposed to solve this problem in many other fields such as data mining, time series similarity analysis, and graph (matching) theory. Popular similarity measures for pattern matching are cosine similarity [65], Dynamic Time Warping [66], angular metric for shape similarity (AMSS) for time series data [67], and auto-regressive integrated moving average (ARIMA) method [68, 69, 70, 71, 72]. Distance metrics used to determine the pattern matching include Euclidean distance [73, 74, 75] and Graph-Edit-Distance (GED) [76]. We provide an overview of the popular approaches in these faculties and why they are not applicable straightforwardly for determining the goodness of energy profiles in section 2.5.

To summarize, there is no effective metric to measure the goodness of energy profiles. In this Chapter, we present a novel methodology called **T**rend-based **S**imilarity **M**easure (TSM) of energy profiles, which measures the similarity between a given energy profile and the ground truth. TSM is designed to capture the underlying energy consumption trend of the profiles, and is composed of the following four stages: i). The regression model of the energy profile is learned, ii). The regression fits of this energy profile and the ground truth are compared to determine if they exhibit the same trend, iii). If they do not, then the energy profile is branded fundamentally inaccurate; iv). If they do, the distance between the regression models of the energy profile (that follows the same trend-line as of the ground truth) and the ground truth is determined using Euclidean distance as a metric of goodness of the energy profile.

To the best of our knowledge, this is the first work to estimate the goodness of energy profiles by taking into consideration the qualitative difference of the underlying energy consumption trends. Also, unlike other statistical methods used for goodness estimation, it uses the Euclidean distance metric for quantitative estimation of similarity between non-linear and non-smooth profiles, and thereby increasing the accuracy of estimation. We compare TSM with popular approaches such as Euclidean distance, average and maximum prediction errors, and correlation coefficient for a diverse set of 235 application energy profiles obtained on multicore heterogeneous hybrid computing platforms using three popular energy measurement approaches i). System-level measurements using power meters, ii). Integrated on-chip power sensors, and iii). Energy predictive models. It is discovered that the popular statistical approaches do not capture the underlying energy consumption trend and thus erroneously rank an inaccurate energy profile as better than more accurate ones in some cases. We find that using inaccurate profiles, obtained with state-of-the-art measurement tools, in energy optimization loop may lead to significant energy losses (up to 54% in our case). We find TSM to be more effective when employing in the energy optimization loop than the popular statistical approaches.

The rest of the chapter is organized as follows. The challenges with common practices to measure the goodness of energy models are presented in section 6.3. The formal description of similarity matching problem is presented in section 6.2. The proposed solution method is presented in section 6.4, and the experimental validation of the proposed method is presented in section 6.5. We compare TSM with stat-of-the-art statistical approaches for energy optimization of an application in section 6.6. Finally, the summary of the chapter is presented in section 6.7.

6.2 Goodness Measuring Problem Formulation

Accuracy or goodness of energy profiles of an application can be defined as the degree to which the energy consumption data of the methods which are employed to produce these profiles, conform to the ground truth. Hence, the similarity of an energy profile with ground truth is also implicitly determined when calculating its accuracy. An energy profile of an application is represented as a function of workload size. We define the *goodness* as a measure that provides an absolute value of resemblance between two vectors (ground truth and an energy profile) in a solution space (i.e. set of energy profiles of an application (EPS)).

Goodness Measuring Problem: Consider an energy profile E(A) of an application A given by a discrete set, $E(A) = \{e(x_1), e(x_2), ..., e(x_n)\}$ where $e(x_i)$ $i \in \{1, 2, \cdots, n\}$ is the energy consumption by the workload size x_i. Let there be m energy profiles of the same application A for the same range of problem sizes constructed with different energy measurement approaches. Let EPS_A denotes the set of energy profiles of an application A. Then, the problem is to find the best energy profile in EPS_A which has maximum resemblance with ground truth among all energy profiles.

6.3 Challenges With State-of-the-art Practices To Measure The Goodness Of Energy Models

Multicore architectures are now prevalent in all computing settings ranging from a handheld mobile device to HPC computing platforms and supercomputers. However, the advent of the multicore era has also introduced several inherent complexities, which are: a) Severe resource contention due to the tight integration of tens of cores contending for shared on-chip resources; b) Non-Uniform Memory Access (NUMA), and c) Dynamic Power Management (DPM) of multiple power domains such as CPU sockets, DRAM. The functional relationship between the energy and workload size has complex (non-linear and non-smooth) properties on modern multicore CPUs. Profile-based energy optimization algorithms [30, 18] leverage the profile variations to find energy-efficient workload distributions. At the same time, the state-of-the-art statistical approaches consider energy profiles as linear or smooth functions of workload sizes to find their goodness. The failure of capturing the qualitative differences in energy profiles can drastically affect the energy optimization efforts and can cause significant energy losses [21, 37].

We present two case studies to highlight the inadequacies of the aforementioned statistical measurement. The first is based on the results of [21]. Consider an energy profile of multiplication of two dense $N \times N$ matrices on an Intel Haswell server comprising of two CPU 12-core sockets, using Intel Math Kernel Library (MKL) DGEMM routine (see Figure 6.2). We run two MKL-DGEMM routines in parallel on both sockets, each with a different workload size ($N \times N$), ranging from 10000×10000 to 14928×14928 with a constant step size of 64 on socket1 and from 15000×15000 to 19928×19928 with a constant step size of 64 on socket2. The x-axis in the plot shows the problem size range for socket1. We measure the total dynamic energy consumption by these two parallel workloads using RAPL [41] and HCLWattsUp [38]. The latter provides system-level power measurements using external power meters and is considered to be the ground truth.

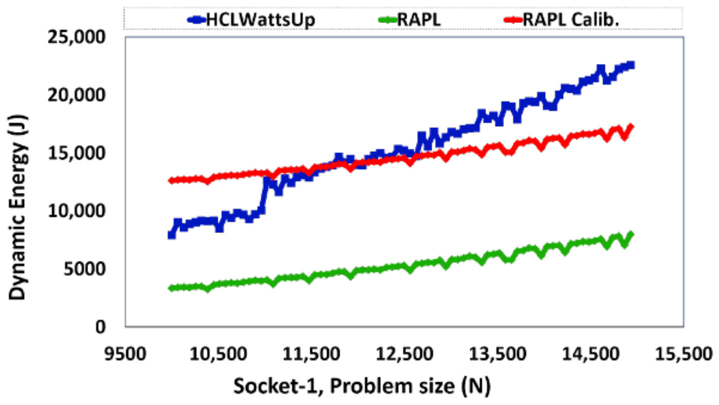


Figure 6.2: Dynamic energy consumption profile segments of matrix-matrix multiplication using Intel MKL constructed with HCLWattsUp and Intel RAPL on HCLServer01.

One can observe that RAPL under-reports the dynamic energy consumption for the range of all problem sizes, resulting in the average and maximum errors of 64% and 69% respectively. The Euclidean distance of 92104 between these profiles is large, but there exists a strong positive correlation of 0.97 between them given by the Pearson correlation coefficient. Despite the strong positive correlation, the profiles disagree on energy consumption behavior for almost 50% of the data points.

For example, for data points (N) {10512,11152,11984,12624,13712} HCLWattsUp suggests a percentage decrease of {8,5,3,2,6} in dynamic energy consumption with respect to immediately preceding data points. In contrast, RAPL suggests a percentage increase of {8,14,9,13,13}. Similarly, while HCLWattsUp suggests a percentage increase for data points {12944,13328,13584,13968} by {3,7,8,3}, RAPL suggests a decrease of {9,7,8,8}.

Furthermore, although both profiles exhibit an overall rising trend of energy consumption, the degrees of the slopes are significantly different. The divergence between the profiles increases with the increase of the problem sizes. All three aforementioned statistical measurements, however, fail to capture this behavior.

The high positive correlation coefficient between the profiles indicates that the RAPL profile can be calibrated. Hence, its average and maximum errors can be reduced to 18% and 59% from 64% and 69% respectively by calibrating the RAPL readings with HCLWattsUp (using a constant positive offset). The Euclidean distance also reduces from 92104 to 26502 after calibration. However, the calibration does not improve the overall qualitative difference in energy consumption behavior. The overall energy consumption trend remains different even after calibrating RAPL readings. It suggests that the correlation coefficient, average error, and Euclidean distance are not sufficient measures for comparing the similarity of energy profiles.

The next case study demonstrates that a non-similar energy profile used as an input to an energy optimization algorithm can cause significant energy losses. In figure 6.1, two sample energy profiles are compared against the ground truth (labeled *Real*). The average errors of profiles *Model1* and *Model2* against the ground truth are 62% and 64% respectively. The Euclidean distance between profiles *Model1*, *Model2*, and the ground truth is 18108 and 33550 respectively. *Model1* and *Model2* are equally strongly correlated with the ground truth with the correlation coefficient equal to 0.91.

While *Model1* is ranked better than *Model2* by both the Euclidean distance and average error, it exhibits different energy consumption behavior for more than 40% of data points as compared with ground truth. Hence, it can cause a significant loss of energy when used as an input to the energy optimization algorithm [30], which employs the workload size as the decision variable for energy optimization of an application. For example, *Model1* only provides 21% of workload distributions that are the same as those provided by ground truth when used as an input to this algorithm for energy optimization. In contrast, *Model2* provides the same workload distributions as of the ground truth for 79% of problem sizes despite its higher average error and greater Euclidean distance. Therefore, *Model2* is better than *Model1* for the use in energy optimization or energy consumption analysis of the application.

To summarize, the average error, Euclidean distance, and correlation coefficient are not sufficient to measure the similarity between the energy profiles despite being the most commonly used statistical measures for this purpose. In general, the average error and Euclidean distance are highly sensitive to outliers and do not capture the similarity of energy consumption trends. They are also highly sensitive to the transformations such as uniform amplitude/time scaling, shifting, etc. Pearson correlation coefficient, on the other hand, assumes a linear relationship between the variables which might not be always true. It can also be easily misinterpreted as the high correlation coefficient does not necessarily mean a strong linear relationship or high similarity between two variables. Similarly, a lesser average error or Euclidean distance do not indicate a higher similarity of energy profiles.

Finally, they can mislead in many cases by erroneously grading an energy profile as the best and thereby causing significant energy losses when used for the energy optimization of an application.

6.4 Trend-based Similarity Measuring Methodology for Energy Profiles

In this section, we present our solution method called **T**rend-based **S**imilarity **M**easure for energy profiles (TSM) to determine the similarity between the energy profiles and the ground truth. We use the term *model* to represent the regression model of an energy profile in the rest of this Chapter for illustration purposes, unless stated otherwise.

The inputs to TSM are the precision settings and a set of energy profiles (constructed with different energy measurement approaches) and the ground truth (EPS). The precision settings are the same as the experimental settings used to construct the energy profiles of the application as explained in Chapters 4 and 5. For example, for each data point in the energy function of an application, we repeatedly execute the application until the sample mean lies within the 95% confidence interval and a precision of 0.025 (2.5%) has been achieved. The output of TSM is the ranking of energy profiles based on their distances which reflect their resemblance with the ground truth.

TSM is composed of the following four stages: i). The underlying regression model of the energy profile is learned, ii). The regression fits of this energy profile and the ground truth are compared to determine if they exhibit the same trend, iii). If they do not, then the energy profile is branded fundamentally inaccurate; iv). If they do, the distance between the regression models of the energy profile (that follows the same trend-line as of the ground truth) and the ground truth is determined using Euclidean distance as a metric of goodness of the energy profile.

6.4.1 Model Fitting

Energy profiles are usually constructed as a function of problem size, CPU threads/cores, or CPU frequency. The configuration parameters have a strong influence on the overall energy consumption behavior of an application. The experimental observation in our previous Chapters (4 and 5) is that the overall trend of the energy profile of an application is a monotonically increasing function of workload size. Likewise the energy profiles, their underlying energy consumption trend can be linear or non-linear, however, the general direction of energy consumption is monotonically increasing.

The authors in [18] also report the same. They find that the dynamic energy profiles of an application in the single-core era increase monotonically with problem size and are smooth linear functions of problem size. However, multicore CPUs exhibit inherent complexities including a) Non-uniform memory access (NUMA); b) Severe resource contention due to the tight integration of tens of cores contending for shared on-chip resources such as last level cache (LLC), interconnect, and DRAM controllers; and c) Dynamic power management (DPM) of multiple power domains (CPU sockets, DRAM). Due to these complexities, while the trend of the energy profiles is still a monotonically increasing function of workload size, the functional shape is non-smooth and non-linear.

Therefore, the first step of TSM is based on the regression analysis of the energy profiles to examine their underlying regression models using the aforementioned application configuration parameters as predictor variables to model the energy consumption. We use polynomial regression to model the relationship between the energy consumption of an application and its configuration parameter.

Polynomial regression fits the relationship between the dependent variable (the energy consumption) and the predictor variable (the application configuration parameter), as an n-th degree polynomial. Therefore, it can estimate both linear and non-linear models. For example, the linear models are fit as polynomial regression of degree 1 whereas the non-linear models are fit as higher (i.e. greater than 1) degrees of polynomial regression (such as quadratic, cubic, etc.). To facilitate clarity of exposition, the mathematical form of kth order of polynomial regression model [164] can be stated as follows:

$$f_E(y) = c_0 + \sum_{i=1}^{n} c_i x_i^k \tag{6.1}$$

where $x = \{x_1, ..., x_n\}$ is the predictor variable; c_0 is the intercept; k is the degree of polynomial; and $c = \{c_1, c_2, ..., c_n\}$ is the vector of coefficients (or the regression coefficients). In real life, there usually is stochastic noise (measurement errors). Therefore, the model can be expressed [165] as

$$\tilde{f}_E(y) = f_E(y) + \epsilon \tag{6.2}$$

where the error term or noise ϵ is a Gaussian random variable with expectation zero and variance σ^2, written $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

In [53], the authors report that the dynamic energy models having a non-zero intercept violate the basic principle of the theory of energy predictive models for computing. This is because dynamic energy is consumed by the CMOS component due to the switching activity when executing an application. There is no switching activity in the absence of workload execution, and therefore the system dissipates static energy only. Hence, regression models should predict no dynamic energy consumption when there is no workload. Therefore, to conform to this principle, we force the intercept to be zero while fitting the regression models.

To choose the order of the polynomial regression model that reflects the best fit, we follow a systematic approach called the forward selection procedure. In this approach, the models are successively fit in an increasing order of polynomial and the significance of regression coefficients is tested at each step of model fitting. The order is kept increasing until F-test for the highest order term is non-significant. Briefly, the F-test of overall significance indicates whether the regression model provides a better fit to the data than a model that contains no independent variables. It has the following two hypotheses: i). The Null hypothesis: It states that the model with no independent variables (intercepts only) fits the data equal to the regression model, and ii). The alternative hypothesis: It states that the regression model fits the data better than the intercept-only model. A regression model is considered as significant if the p-value of the F-test is less than the significance level (i.e. 95% of the confidence interval or 0.05 level). We find that the first or third-order polynomials as the best fit for all the energy profiles in our application suite.

We want to emphasize here that the purpose of fitting the regression model is not to build an offline energy model to predict the energy consumption by employing a predictor variable. Instead, the regression analysis is performed to examine the underlying model of energy profiles in an EPS to facilitate the comparison of their energy consumption trend. Therefore, we fit the regression model on the whole data-set instead of splitting it into training and test datasets.

6.4.2 The Discrepancy Analysis

In the second step, we compare the regression models of energy profiles and the ground truth in an EPS. We term this test as the *discrepancy analysis*. The following conditions must hold for the regression models of two energy profiles to be ideally alike:

1. The regression models of the energy profile and the ground truth must follow the same orientation. Both regression models must exhibit the same increase and decrease for the range of all the data points.
2. The regression models of the energy profile and the ground truth must not intersect at any point.
3. The distance between the regression models of the energy profile and the ground truth must be the same for the range of all the data points.

All of these properties must be satisfied by a regression model to be considered as ideally similar to that of the ground truth.

Mathematically, the idea can be expressed as the slope and its direction must be the same for the regression models of an ideally similar energy profile and of the ground truth. The slope and its direction can be determined by taking the first and second derivatives of the regression models. Therefore, we compare the first and second derivatives of the regression equations of an energy profile and the ground truth. While the first derivative indicates whether the energy consumption trend is increasing or decreasing, the second derivative tells about the shape of the underlying regression model of the energy function. Two regression models do not follow the same trend if the second derivative is positive (greater than zero) for one of them and the negative (less than zero) for the other one, or vice versa. However, they follow the same direction if the second derivatives of both regression models are either positive (greater than zero) or negative (less than zero). The regression models that do not follow the same direction are classified as *opposite* and consequently removed from the EPS.

In the next step, we compare the coefficients of the second derivatives of the regression models that follow the same direction. Two models are considered as *same* if the difference of coefficients of their derivatives are within an interval of input precision settings. To illustrate this, consider two third-order (cubic) polynomial regression models $r1$ and $r2$. Let $r1$ is the regression model of the ground truth. Let the coefficients of the second derivatives of both models are $c1$ and $c2$ respectively. Then, the difference between the coefficients of the second derivatives of both models is calculated as $\epsilon = |(c1 - c2|/c1 \times 100$. Now, if ϵ lies within the input precision settings, then both regression models are considered as the *same*. Otherwise, they are classified as *similar*.

To summarize, we analyze the qualitative behavior of regression models of the energy profiles and the ground truth by comparing the derivatives of their polynomial functions. As a result, we classify the energy models into one of the following three categories:

1. *Opposite*: The slopes of the regression models of an energy profile and the ground truth are in the opposite direction. The regression models of an energy profile and the ground truth exhibit opposite behavior such that one of them is increasing with an increase in x and the other one is decreasing with an increase in x. Furthermore, the shape of the regression fit is concave up for one of them and concave down for the other one.

2. *Same*: The slopes of the regression models of an energy profile and the ground truth are identically the same and follow the same direction. This class represents the energy profiles which are ideally the same to their corresponding ground truths.

3. *Similar*: The slopes of the regression models of an energy profile and the ground truth are different, however, they follow the same direction. It indicates that the energy profile is neither the *same* as the ground truth nor in the *opposite* direction to it.

As a result of this step, the energy profiles that have regression fits in the opposite direction to that of the ground truth are removed from the EPS. Consequently, the resulting EPS contains only the *same* or *similar* energy models. The goodness of the remaining energy profiles to the ground truth is quantified in the third step.

6.4.3 The Distance Metric

In this step, we determine the distance between the regression fit of each energy profile and the ground truth in the remaining EPS. For this purpose, we use Euclidean distance as a distance metric to establish an absolute value of the distance of the regression fit of each profile and the ground truth. Because of its triangular-inequality property, Euclidean distance is used to index the model space which speeds-up the search and matching in general, especially for the huge model space. Furthermore, it helps in ranking the similar profiles based on their distance with the ground truth in an EPS.

Hence, to rank the profiles based on their similarity, first, the Euclidean distance between the regression models of the profiles and the ground truth belong to the same EPS is computed. Then, the energy profiles are ranked according to the Euclidean distance between their regression models and that of the ground truth. The energy profile whose regression model has the least Euclidean distance with that of the ground truth is considered as the most similar profile in that particular EPS. The final output of this step is the sets of the energy profiles with similarity ranks. Two profiles may have the same rank if their Euclidean distance differs by less than or equal to the input precision.

6.5 Experimental Validation of TSM

6.5.1 Experimental Platform and Applications

The dataset used in this work comprises of the energy profiles of different application configurations executed on multicore heterogeneous hybrid computing platforms and constructed with on-chip sensors, power meters, or energy predictive modes employing PMCs as predictor variables. The profiles are constructed as the results of the experiments explained in 4 and 5. The details on experimental set-up, platforms, application suite, configuration parameters, and the boundary conditions to construct the dataset are the part of the experiments presented in the 4 and 5. Briefly, the following application configuration parameters are employed to build the energy profiles: (a) Problem size, (b) Number of CPU threads, or the number of CPU cores. The application suite used to construct the profiles contains highly optimized memory bound and compute-bound scientific routines such as OpenBLAS DGEMM, FFTW 2D, MKL-DGEMM and MKL-FFT from Intel Math Kernel Library (MKL), benchmarks from NASA Application Suite (NAS), Intel High Performance Conjugate Gradients (HPCG) HPCG, *stress*, *naive* matrix-matrix multiplication and *naive* matrix-vector multiplication.

We employ three nodes for constructing the energy profiles: (a) HCLServer01 has an Intel Haswell multicore CPU having 24 physical cores with 64 GB main memory and integrated with two accelerators: one Nvidia K40c GPU and one Intel Xeon Phi 3120P, (b) HCLServer02 has an Intel Skylake multicore CPU consisting of 22 cores and 96 GB main memory and integrated with one Nvidia P100 GPU and (c) HCLServer03 has an Intel Skylake multicore CPU having 56 cores with 187 GB main memory. Technical description of each node is presented in tables 4.1, 4.2 and 4.3 respectively. HCLServer01 and HCLServer02 are connected with a *Watts Up Pro* power meter; HCLServer03 is connected with a *Yokogawa WT310* power meter. Watts Up Pro power meters are periodically calibrated using the ANSI C12.20 revenue-grade power meter, Yokogawa WT310. The maximum sampling speed of Watts Up Pro power meters is one sample every second. The accuracy specified in the data-sheets is $\pm 3\%$. The minimum measurable power is 0.5 watts. The Intel MKL on all HCLServers is 2017.0.2, whereas the CUDA versions used on HCLServer01 is 7.5 and 9.2.148 on HCLServer02.

We use RAPL [41], NVIDIA NVML [43] and Intel System Management Controller chip (SMC) [42] (using Intel manycore platform software stack (MPSS) [102]) to determine the energy consumed by the application kernels executing on Intel CPUs, Nvidia GPUs and Intel Xeon Phi respectively. HCLWattsUp interface [38] is used to obtain the power measurements from the WattsUp Pro power meters. The details on methodology used to obtain a reliable data point using different tools (on-chip power sensors and power meters), and HCLWattsUp interface are explained in 3. Briefly, we follow a statistical methodology to ensure reliability of our experimental results. The methodology determines a sample mean (for the execution time, dynamic energy or PMC) by executing the application repeatedly until the sample mean meets the statistical confidence criteria (95% confidence interval, a precision of 0.025 (2.5%)). Student's *t*-test is used to determine the sample mean. The test assumes that the individual observations are independent and their population follows the normal distribution. We use Pearson's chi-squared test to ensure that the observations follow normal distribution. The details on experimental setup and the methodology to construct an energy profile is explained in sections 4.3 and 5.3.

6.5.2 Experimental Methodology to validate TSM

We classify our suite of energy profiles (EPS) into the following two groups:

1. Group A (Sets of many energy profiles): Group A comprises of the EPS where there is more than one energy profile of the same application constructed with different approaches such as on-chip power sensors, system-level power measurements provided by power meters, etc.

2. Group B (Sets of single energy profiles): Group B comprises of the EPS where only one energy profile is compared with the ground truth.

For each group, we fit the regression models as n-th order of polynomial for each energy profile and their corresponding ground truths belonging to the same EPS. To choose the best order of polynomial approximation, we follow the forward selection procedure as explained in section 6.4.1. Intuitively, the polynomial order should be the same for the regression models of an energy model and its corresponding model of the ground truth belonging to the same EPS. As a result of this sanity check, we reject the energy functions which have a different order of polynomial as a best fit than the regression model of the ground truth.

In the next step, we analyze the qualitative behavior of regression models of the energy profiles and their corresponding ground truths by comparing the derivatives of their polynomial functions as explained in section 6.4.2. The energy profiles classified as opposite are removed from their respective EPS, as a result of this step. In the third step, we determine the similarity between the remaining energy profiles and the ground truth using Euclidean Distance. We compare the results of TSM with other statistical approaches such as correlation, Euclidean distance, and average error to compare the accuracy and similarity of energy profiles (and also for the time series of equal lengths in general). The Euclidean distance that is compared with TSM is the distance between the energy profiles and the ground truth. In contrast, TSM uses the Euclidean distance as a metric to determine the distance between the regression models of the energy profiles and the ground truth in an EPS.

6.5.3 Results and Discussion

Group A (Sets of many energy profiles): The similarity ranking by TSM and popular statistical approaches for the energy profiles in each EPS that belongs to group A are provided in table D.1 in appendix D. One can observe that the correlation coefficient does not always distinguish much between the energy profiles. Consider, for example, the profiles in EPS DGEMM_EqualLoad. The profiles RAPL_Parallel and RAPL_Combined both have a correlation of positive 0.9993 with the ground truth (HCLWattsUp_Parallel). Similarly, the correlation coefficient for RF_Additive and NN_Additive is 0.9999 with the ground truth in EPS DGEMM_Predictive Models.

Similarly, the average prediction error also misleads in many cases. Consider, for example, the profiles in EPS DGEMM_EqualLoad. The average prediction error suggests the HCLWattsUp_Combined as the most similar profile with the ground truth. However, TSM suggests RAPL_Combined as the most similar, and HCLWattsUp_Combined as the most different among all three profiles in EPS DGEMM_EqualLoad. A visual illustration of regression models of the profiles in EPS DGEMM_EqualLoad as presented in Figure 6.3, also conforms to the TSM. In general, one can observe that regression models of all three profiles in EPS DGEMM_EqualLoad follow the same pattern as of the ground truth (HCLWattsUp_Parallel). However, both the RAPL_Parallel and RAPL_Combined exhibit the closest resembling pattern with the ground truth for the range of all problem sizes as illustrated in figure 6.3(b). HCLWattsUp_Combined, on the other hand, exhibits a slightly different pattern at both ends of data points (that is the range of very small problem sizes and very large problem sizes) as shown in figure 6.3(a). Therefore, while it follows the same orientation, it is ranked as the least similar profile in its EPS.

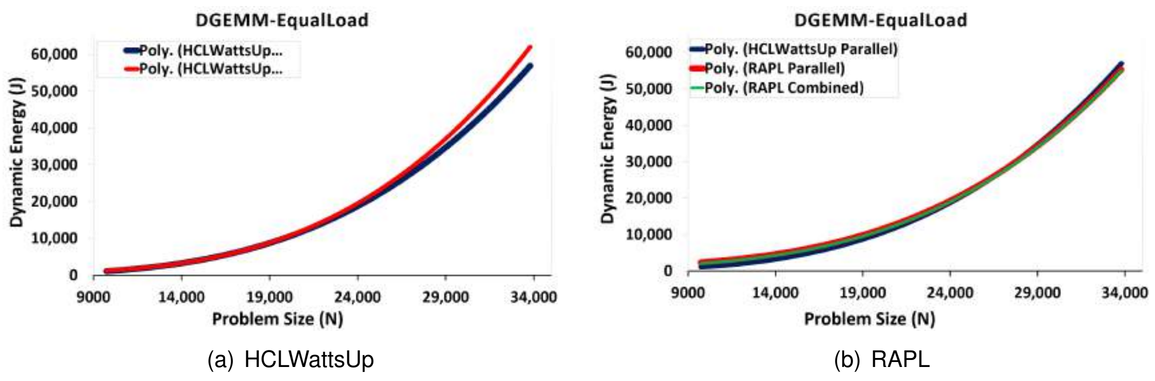
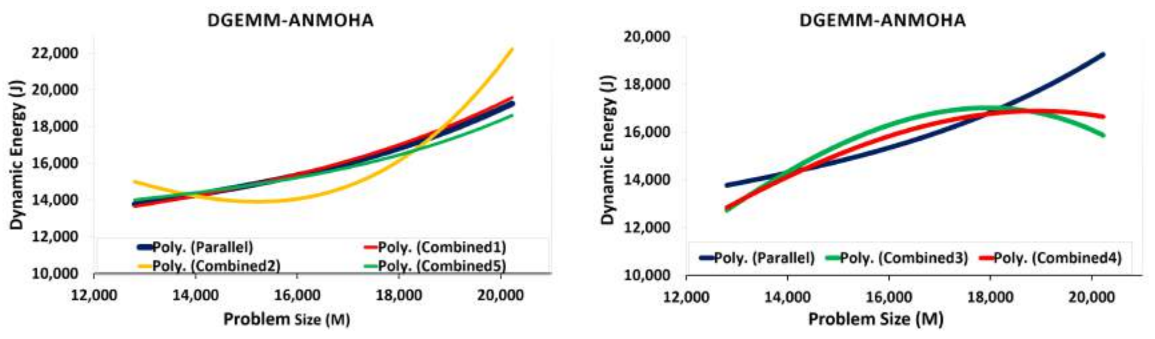


*Figure 6.3: Regression models of energy profiles in EPS DGEMM_EqualLoad constructed with **a)** HCLWattsUp, and **b)** RAPL*

However, the similarity results as presented in table D.1 suggest that overall the Euclidean distance between the energy profiles proves to be more efficient than the correlation coefficient and average prediction error. In most of the cases, it suggests the similarity ranking in line with TSM. However, it also misleads in some of the cases. Consider, for example, the similarity ranking for the profiles in EPS DGEMM_Predictive Models. Euclidean Distance between the profiles ranks LM_additive as the third most similar profile, whereas TSM ranks it as the fifth most similar profile. Similarly, Euclidean distance between the profiles ranks RF_NonAdditive as the second most similar in EPS FFT_Predictive Models, whereas TSM ranks it as the third most similar to the ground truth in its EPS.

It is important to note that none of the aforementioned statistical measurement and metric captures the holistic picture of the energy consumption trend of the profiles. Consider, for example, the profiles in EPS DGEMM_AnMoHA. Both the Euclidean distance and average prediction error consider the profiles Combined_3 and Combined_4 as the third most similar and fourth-most similar profiles with the ground truth (Parallel) as presented in table D.1. However, one can observe in figure 6.4(b) that the qualitative comparison of the regression fit of both profiles and the ground truth by TSM suggests them to have a different energy consumption trend and thus drop them from the EPS. The correlation coefficient ranks the profiles in this particular EPS in line with TSM and ranks them as least similar. But, it also does not provide the details on their qualitative difference of the underlying energy consumption behavior. Similarly, both the Euclidean distance and average prediction error rank Combined_2 as the least similar profile. In contrast, TSM ranks it as the third most similar profile.

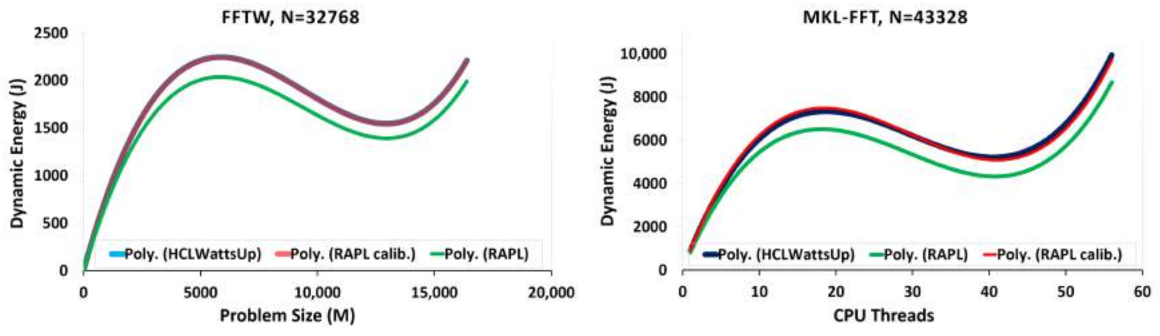


(a) Similar (b) Opposite

Figure 6.4: Regression models of energy profiles in EPS DGEMM_AnMoHA. ***a)*** *Combined_1 and Combined_5 follows the* same *trend, Combined_2 follows* similar *trend, and* ***b)*** *all profiles exhibit* opposite *trend as of the ground truth*

Graphical illustration of regression models of the profiles in EPS DGEMM_AnMoHA as presented in figure 6.4 also confirms the same results. One can observe that Combined_3 and Combined_4 exhibit different energy consumption behavior as of the ground truth (Parallel). However, Combined_1 and Combined_5 follow the same direction with the same slope, whereas Combined_2 follows the same direction but exhibits different orientation.

Group B (Sets of single energy profile): For group B, TSM classifies the similarity of energy profiles with ground truths (after comparing their regression fits) into the three similarity categories explained in section 6.4.2. The similarity ranking by TSM and popular statistical approaches for the energy profiles in each EPS that belongs to group B, are provided in table E.1 in appendix E. One can observe that likewise group A, all three aforementioned statistical approaches fail to capture the qualitative difference of the regression models of the energy profiles and the ground truth belong to the same EPS.



(a) FFTW (b) MKL-FFT

Figure 6.5: Group B (Sets of single energy profile), Class ***same*** *Similarity. Group B comprises of sets of energy profiles where only one energy profile is compared with the ground truth*

Consider, for example, the regression models of the energy profiles illustrated in figures 6.5 and 6.6 representing the classes *same* and *similar* respectively. The regression models of the energy profiles follow the same trend as the ground truths in both cases. However, the slopes of the regression models presented in figure 6.6 are different from their corresponding ground truths. Figure 6.7 illustrates the regression models representing the class *opposite*. It can be observed that

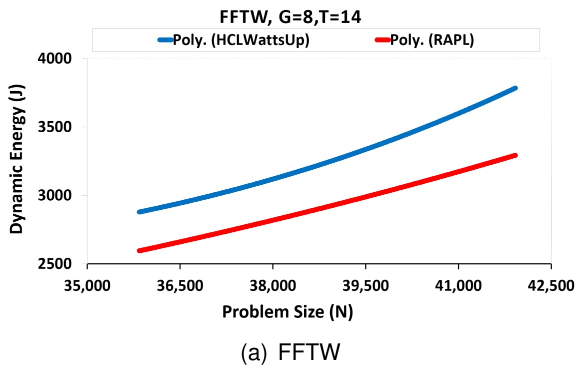


*Figure 6.6: Group B (Sets of single energy profile), Class **similar** Similarity. Here, G= CPU thread groups, and T=CPU threads*

the regression models of the energy profiles and their corresponding ground truths exhibit different trends. Consider, for example, the regression models of the EPS {FFTW,G=16,T=7}. Here, G and T represent the number of thread groups and the number of threads per group respectively. The slopes of the regression models of both profiles are different and have different signs, positive for RAPL and negative for HCLWattsUp. Figure 6.7(c) also shows the same results. One can observe while the shape of the regression model of RAPL is concave up, it is concave down for HCLWattsUp. However, the popular statistical approaches do not capture this behavior.

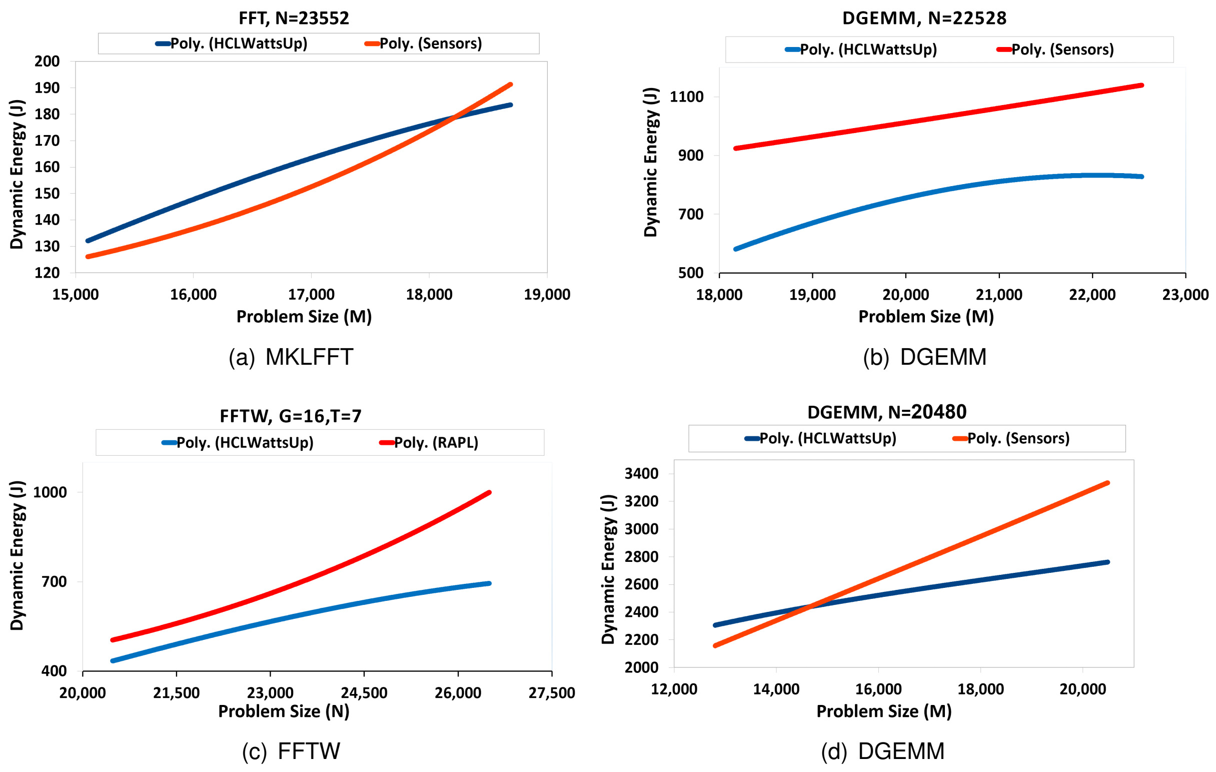


*Figure 6.7: Group B, Class C: **opposite** Similarity*

6.5.4 Discussion

Average error and Euclidean distance do not indicate whether the calibration can improve the average error or Euclidean distance between two similar energy profiles, and thus can mislead to consider an accurate energy profile as inaccurate. However, unlike average prediction error and Euclidean distance, TSM can indicate if the average prediction error and Euclidean distance can be reduced by calibrating the energy profile with the ground truth in an EPS.

Consider, for example, the energy profiles in EPS FFTW where the configuration parameter is the problem size $M \times N$ where $M \leq N$, and N = 32768. Figure 6.5(a) illustrate the regression models of the profiles. The difference between the slopes of the regression fit of the RAPL energy profile and the ground truth is very close to zero, and thus TSM classifies their similarity as the *same*. This *same* similarity suggests that the regression models of both the RAPL energy profile and the ground truth exhibit the same energy consumption behavior. Therefore, one can reduce the average prediction error and Euclidean distance between the RAPL energy profile and the ground truth from 10.5% to 0.6% and from 1134.9 to 94 respectively after calibrating it with the ground truth. That is an improvement of 94% in average prediction error and 92% in the Euclidean distance between the profile.

Similarly, consider the EPS IntelMKLFFT where the configuration parameter is CPU cores and problem size N is 43328. Figure 6.5(b) illustrates the regression models of the profiles. The difference between the slopes of the regression models of both the RAPL energy profile is close to zero, but slightly more than the difference between the energy profiles belong to the aforementioned EPS FFTW. TSM classifies both the RAPL energy profile and the ground truth as *same*. After calibration, one can reduce the average prediction error and the Euclidean distance between the profiles from 13% to 2.19% and from 6700 to 1495.83 respectively. That is an improvement of 83% in average prediction error and 78% in Euclidean distance between the RAPL energy profile and the ground truth in that particular EPS.

It is important to note while the similarity classes such as *opposite* and *same* are more useful for group B, the similarity class *similar* provides less information. It does not present any threshold to indicate the absolute value of the similarity between the energy profile and the ground truth. The percentage or threshold that indicates the value of absolute similarity is highly dependent on the application domain and, therefore, is a matter of choice.

Consider, for example, applications such as signal processing or multimedia processing. Such applications are considered as fault-tolerant and belong to the approximate computing domains. A possibly inaccurate result is also acceptable in such domains. Therefore, a comparatively less similar energy profile can also serve the purpose in this case. In contrast, the high similarity value is required for applications such as cryptography or hard real-time applications. Therefore, one sets a comparatively higher value of the similarity to ensure whether the energy profiles are similar. That is why, TSM does not define a threshold to indicate the degree to which an energy profile exhibits a similar energy consumption behavior to the ground truth. Instead, it just compares the energy consumption behavior and the shapes of the regression models of the energy profile with the ground truth in an EPS and determines whether or not both have a similar shape and energy consumption behavior.

To quantify the *similar* energy profiles, one can take the difference of the polynomials or the derivatives of the regression models of the energy profile under consideration and the ground truth. A zero value of the difference between the polynomials or derivatives indicates the same polynomials and thus the same regression models. One can give some weight to the energy profile under consideration indicating how large is it from zero value of the difference, and thus how less similar is it with the ground truth.

Unlike the profiles classified as *same*, there is little to none margin for average error and the Euclidean distance reduction after calibration, if the profile is classified as *similar*. This is because the derivatives of the regression model of the energy profile under consideration and the ground truth have different slopes. Therefore, the calibration can reduce the average error and Euclidean distance between the energy profile and the ground truth only to an extent. However, it highly depends on the value of the similarity between the polynomials/derivatives of the regression models of the energy profile and the ground truth in an EPS.

Consider, for example, the EPS FFTW where problem size N ranges from 35840 to 41920 and the configuration of CPU threads are grouped into 8 and there are 14 CPU threads in each group. We refer to this EPS as EPS_1 for illustration purposes. Figure 6.6(a) illustrates the regression models of the profiles in EPS_1. One can reduce the average prediction error and Euclidean distance between the RAPL energy profile and the ground truth from 13.66% to 12.73% and from 5569.4 to 4520.9 after calibration. That is an improvement of 6.81% in average prediction error and 18% in Euclidean distance between the RAPL energy profile and the ground truth in EPS_1.

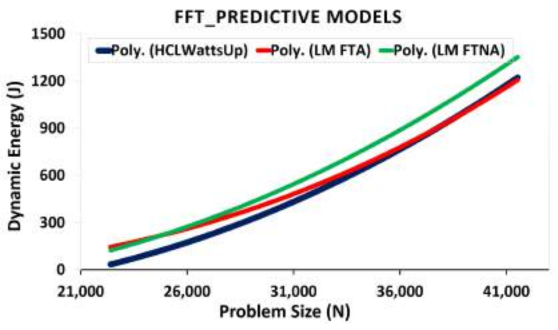
In contrast, consider the EPS FFTW where problem size N ranges from 35840 to 41920 and all 112 CPU threads are grouped into 1 group. Let this EPS be EPS_2 for illustration purposes. Figure 6.6(b) illustrates the regression models of the profiles in EPS_2. The average prediction error and Euclidean distance between the RAPL energy profile and the ground truth can be reduced from 24.62% to 3.9% and from 78669 to 23184.7 after calibration. That is an improvement of 84.16% in average prediction error and 70.5% in Euclidean distance between the RAPL energy profile and the ground truth in EPS_2. This is because the difference in polynomials and derivatives of the regression models of both profiles in EPS_2 is less than the regression models in EPS_1.

Another important finding is that the calibration of less similar profiles with the ground truth can increase the maximum prediction error between them in some cases when trying to reduce the average error and Euclidean distance. Consider, for example, the aforementioned EPS termed as EPS_1. The maximum prediction error between the RAPL energy profile and the ground truth is 29.8% which increases to 55.65% after calibration (using the same offset that reduces the average prediction error and Euclidean distance). That is an increase of 87% in the maximum error. We observe similar findings with other less similar energy profiles. However, it is not the case where the similarity is higher or the *same* between the energy profile under consideration and the ground truth in an EPS. The calibration improves the maximum error, average prediction error, and Euclidean distance between such profiles.

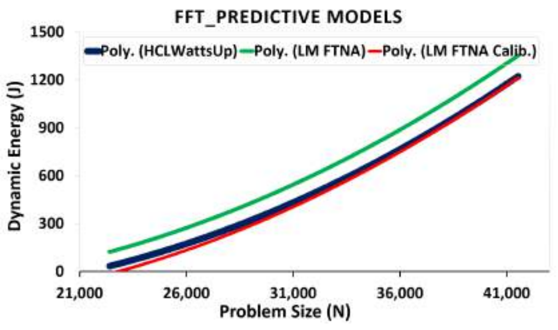
The calibration should only be applied to the profiles that exhibit a similar energy consumption trend as of the ground truth. Because, it only improves the Euclidean distance and prediction error, and thus does not improve the qualitative difference of the energy consumption trend of the profiles.

TSM also indicates whether the predictive model which is employed to construct the energy profile, includes some extraneous contributor that does not reflect the energy consumption by the application, or it lacks some essential contributor to the energy consumption by the application. Consider, for example, the similarity results for EPS FFT_Predictive Models as presented in appendix D. The profile LM_NonAdditive has the highest average error and the greatest Euclidean distance with the ground truth. However, the slopes of the regression models of LM_NonAdditive are in the same direction as the ground truth (HCLWattsUp). Furthermore, the difference between its polynomials and the slopes is close to that of the LM_Additive. However, LM_NonAdditive predicts energy consumption more than the ground truth and LM_Additive. It suggests that the predictive model of LM_NonAdditive includes some extraneous PMC which does not reflect the energy consumption by the application.

Therefore, we apply a constant negative offset to its predictions to calibrate them with ground truth. As a result of this calibration, the average error and Euclidean distance of LM_NonAdditive energy profile with the ground truth is reduced from 92% to 39%, and from 3321 to 2722. This is an improvement of 58% in average error and 18% in Euclidean distance. Consequently, the calibrated energy profile of LM_NonAdditive is closer to LM_Additive is in terms of its average error and Euclidean distance with ground truth. One can observe in figure 6.8 that the regression model of calibrated LM_NonAdditive is in a closer approximation of the ground truth (HCLWattsUp) and the profile LM_Additive.



(a) LM



(b) LM_NonAdditive Calibrated

*Figure 6.8: Regression models of energy profiles in EPS FFT_Predictive Models, **a)** Linear Models, and **b)** Linear Model NonAdditive Calibrated*

This suggests that the prediction error of the energy profile LM_Additive can be improved by removing non-relevant PMCs from the set of explanatory variables. This finding conforms to the results as presented in [53], where the authors present a study to demonstrate how the prediction errors of PMC based energy predictive models can be improved significantly by removing irrelevant PMCs (which does not reflect the energy consumption by the application) from the set of predictor variables.

Likewise the indication of extraneous PMCs, TSM can also indicate whether the predictive model which is employed to construct the energy profile, lacks some essential contributor that strongly reflects the energy consumption by the application. Consider, for example, the regression models of energy profiles of FFTW_32768 and MKLFFT_43328 as shown in figure 6.5. The energy profiles of the application with RAPL exhibit the *same* energy consumption patterns as of the ground truth. However, RAPL under-reports energy consumption in comparison with the ground truth. It suggests that the energy profiles of both applications lack the contributions by some essential components.

The prediction errors and Euclidean distance of both profiles can be reduced significantly by applying a constant positive offset to its predictions to calibrate them with the ground truth. The calibration improves the average prediction error and Euclidean distance of FFTW_32768 by 94% and 92% respectively, and by 83% and 78% respectively for MKLFFT_43328. Hence, TSM may be used as a selection criterion of PMCs in energy predictive models to predict the energy consumption by an application. However, we leave this investigation as a possible future extension of this thesis.

To summarize, the statistical approaches (correlation coefficient, average prediction error, and the Euclidean distance between the energy profiles) fail to distinguish the energy profiles based on their underlying energy consumption trend. They erroneously rank an inaccurate energy profile as better than more accurate ones in some cases. TSM, on the other hand, proves to be more effective in capturing the energy consumption behavior of the profiles and comparing their qualitative differences. It provides more information about the energy consumption behavior of the profiles and thus ranks them based on their proximity with energy consumption behavior of the ground truth. Furthermore, it can also suggest if the calibration can improve the Euclidean distance, and the average and maximum prediction errors between the energy profile under consideration and the ground truth.

6.6 Comparison of TSM and State-of-the-art Statistical Approaches for Energy Optimization

In this section, we compare the effectiveness of TSM with other popular statistical approaches using a profile-based energy optimization algorithm as a yardstick that employs the workload size as a decision variable. Furthermore, we demonstrate that inaccurate energy profiles can cause a significant amount of energy loss when used for the optimization of an application for dynamic energy.

The profile-based energy optimization algorithms [18][30] leverage the variations (jumps and drops) of the energy profiles and determine the workload distributions that optimize the total dynamic energy consumption for the given workload size. These variations in energy profiles are caused by the intrinsic complexities in modern hybrid heterogeneous computing platforms such as resource contention due to non-uniform memory access (NUMA) and the tight integration of multi-core CPU with one or more accelerators. The algorithm provides different workload distributions for non-similar energy profiles used as input for the range of the same workload sizes. This is because non-similar energy profiles exhibit different variations in their energy consumption behavior for the same set of data-points. However, it provides the same workload distributions for the identically same energy profiles used as an input for the range of the same workload sizes.

We use the profile-based data partitioning algorithm [30] to compute the decomposition of workload size to optimize the total dynamic energy consumption of an application. We compare the output workload distributions provided by the algorithm when using as an input the dynamic energy profiles ranked as similar to ground truth by popular statistical approaches and TSM. It is important to note here that the energy optimization algorithm does not make any assumptions about the shape of input energy profiles. The algorithm takes the following inputs: i). the workload size, ii). the number of processors, and iii). the discrete dynamic energy functions of individual processors. The output is the optimal workload distribution that provides minimal dynamic energy consumption for the input workload size. One or more processors may be allocated the workload of size zero. The algorithm has a polynomial complexity of $\mathcal{O}(m^3 \times p^3)$. More details on the algorithm and its complexity can be found in [30].

For our first case study, consider the profiles in the EPS, DGEMM_AnMoHA as illustrated in figure 6.9. The combined energy profiles are constructed following the additive energy modelling approach as presented in 5.2. Briefly, the approach is based on the hypothesis that the total dynamic energy consumption during an application execution will be equal to the sum of energies consumed by all the individual application components executing on processors in the case of loosely-coupled application components. Let $E_A(x)$, $E_B(x)$, and $E_C(x)$ be the dynamic energy consumption by the application kernels of workload size x executing sequentially on processors CPU1, GPU1, and PHI1, and $Combined_{ABC}(x)$ represents the sum value of their dynamic energy consumption. Let $Parallel_{ABC}(x)$ be the total dynamic energy consumption by parallel execution of the same application kernels of the workload size x on the same processors. Then, the additive hypothesis holds only if $Parallel_{ABC}(x) = Combined_{ABC}(x)$.

We run a parallel hybrid application DGEMM (which multiplies two dense matrices A and B of sizes $M \times N$ where $M \leq N$) as explained in 5.3.1 on HCLServer01 (technical specifications are provided in table 4.1) for the workload sizes ranging from 38400×20224 to 60672×20224 with a constant step size of 256. The dimension M is equally partitioned among three aforementioned processors (CPU1, GPU1, PHI1) into M_1, M_2 and M_3 such that the matrix $M_1 \times N$, $M_2 \times N$ and $M_3 \times N$ (i.e 12800×20224) are computed by processor CPU1, GPU1, and PHI1 respectively. There is no communication involved in these experiments. The DGEMM energy profiles in DGEMM_AnMoHA are constructed using different combinations of additive models of application-components executing on processors. More details on additive energy modelling of hybrid parallel applications and the design configurations of independent experiments to construct the energy profiles in DGEMM_AnMoHA can be found in [21].

Fig. 6.9 illustrates the energy profiles in DGEMM_AnMoHA. The average prediction error, correlation coefficients and Euclidean distance of all energy profiles are {2%,8%,7%,6%,4%}, {0.9762,0.8641,0.5741,0.6741,0.8945} and {2258,8795,8421,7523,4515} respectively as presented in table D.1 in appendix D. The Euclidean distance and average prediction error rank the profiles Combined_3 and Combined_4 as the most similar with the ground truth (Parallel) after Combined_1 and Combined_5. However, one can observe in figure 6.9 that the qualitative comparison of both profiles with the ground truth by TSM suggests them to have a different energy consumption trend and thus drop them from the EPS. The correlation coefficient ranks the profiles in this particular EPS inline with TSM and ranks them as the least similar. But, it does not provide the details on their qualitative difference such as the underlying energy consumption trend of the profiles. Similarly, both the Euclidean distance and average prediction error ranks Combined_2 as the least similar profile in its EPS. In contrast, TSM ranks it as the third most similar profile. However, all three statistical approaches likewise TSM rank Combined_1 as the most similar energy profile.

We determine the workload distributions for workload sizes ranging from 38400×20224 to 60672×20224 using the individual additive dynamic energy profiles of each processor CPU1, GPU1, and PHI1 as an input to the data partitioning algorithm [30]. Combined_2 provides 32% of the workload distributions the same as of Combined_1 whereas Combined_3 and Combined_4 provide 29%

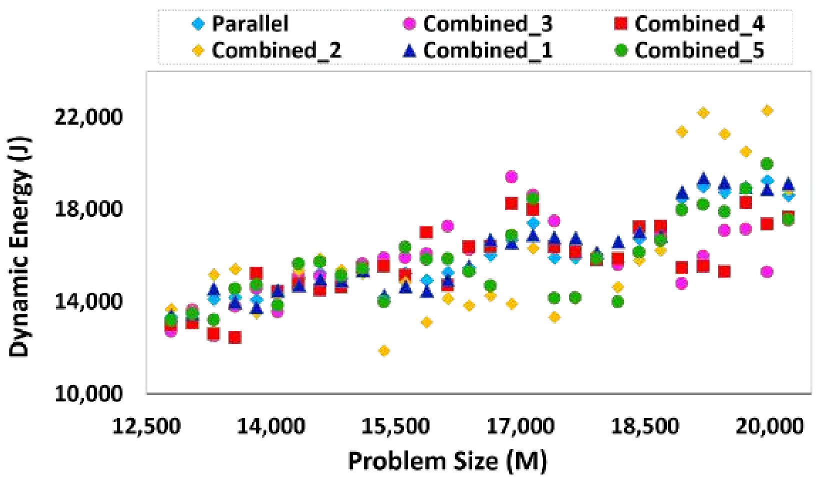


Figure 6.9: Dynamic energy profiles in EPS DGEMM_AnMoHA

and 20% same workload distributions as of Combined_1. This conforms to the results of TSM, which ranks Combined_2 as better than Combined_3 and Combined_5.

We find that overall the workload distributions provided by the algorithm when using Combined_3 and Combined_5 as inputs, consume more dynamic energy for 82% and 81% of the data points respectively in comparison with Combined_2 for the aforementioned workload sizes. Consider, for example, the workload sizes {47616,48128,49664,50176,50688,51200,51712}. The workload distributions of Combined_3 consume {52%,51%,52%,51%,54%,49%,50%} respectively more dynamic energy than the corresponding workload distributions of Combined_2 for the aforementioned workload sizes. Similarly, consider the workload sizes {47104,47616,48128,49152,49664,50176,51200}. The workload distributions of Combined_4 consume {39%,38%,40%,37%,36%,38%,40%} respectively more dynamic energy than the corresponding workload distributions of Combined_2 for the aforementioned workload sizes.

For our next case study, consider the profiles in the EPS DGEMM_EqualLoad in figure 6.10. The energy profiles of DGEMM in this EPS are constructed with RAPL and HCLWattsUp when running equal workload sizes on each CPU socket of a dual-socket multi-core Intel Haswell platform (technical specifications are provided in table 4.1). The details on energy profiles and their construction procedure can be found in [21]. Briefly, we equally partition the workload sizes ($M \times N$) ranging from 19456×9728 to 67584×33792 on both CPU sockets such that the matrix $M_1 \times N$ and $M_2 \times N$ are computed by processor CPU socket1 and CPU socket2 respectively. There is no communication involved in these experiments. Fig. 6.10 illustrates the parallel and combined dynamic energy profiles constructed with RAPL and HCLWattsUp.

One can observe that all three energy profiles have almost the same strong positive correlation with the ground truth. The average errors of HCLWattsUp_Combined, RAPL_Parallel, and RAPL_Combined with HCLWattsUp_Parallel are 4.6%, 21.2% and 16.1 respectively. The correlation coefficient is the same (0.9993) for both RAPL_Parallel and RAPL_Combined, and 0.9995 for HCLWattsUp_Combined. Hence, both the correlation coefficient and average prediction error ranks HCLWattsUp_Combined as the most accurate energy profile in its EPS. However, TSM ranks it as the least similar. It ranks, in contrast, RAPL_Combined as the most similar to the ground truth in that EPS.

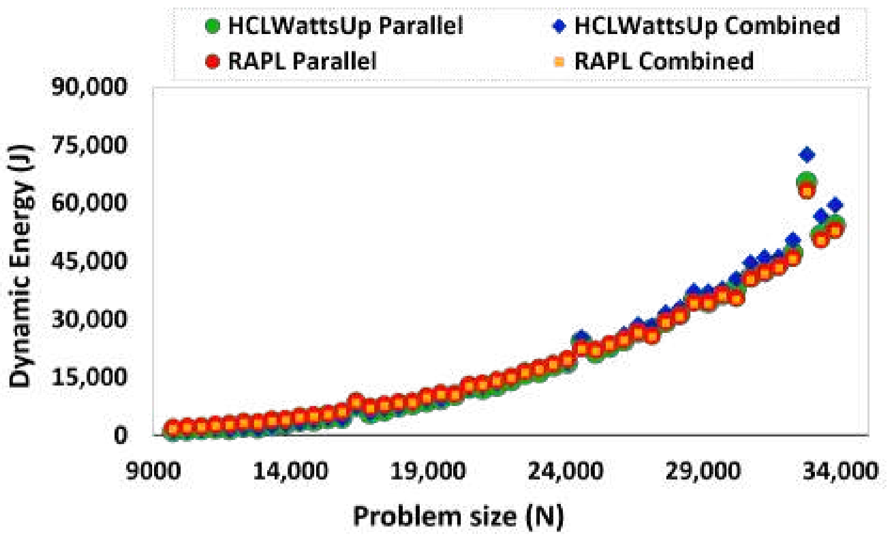


Figure 6.10: Dynamic energy profiles of DGEMM application in the EPS, DGEMM_EqualLoad.

We determine the workload distributions for workload sizes ranging from 19456×9728 to 67584×33792 using the dynamic energy profiles constructed with RAPL and HCLWattsUp as an input to the data partitioning algorithm [30]. For each workload distribution, we run the applications in parallel on both sockets and determine its dynamic energy consumption with RAPL and HCLWattsUp separately. We find that the workload distributions when using HCLWattsUp_Combined consuming more dynamic energy for 65% of the data points of the aforementioned range. Consider, for example, the workload sizes {56320,56832,57344,57856,58368,58880,59392,59904,60928}. The total dynamic energy losses by using HCLWattsUp_Combined in comparison with RAPL_Combined to optimize the dynamic energy consumption of DGEMM for the aforementioned workload sizes are {17%,18%,18%,17%,18%,18%,18%,18%,17%} respectively.

To summarize, we use an energy optimization algorithm [30] as a yardstick to evaluate the effectiveness of TSM and popular statistical approaches to be used in an energy optimization loop of an application. In all the presented case scenarios, TSM proves to be more effective. The energy profiles ranked as similar by TSM provide more number of same workload distributions as of the ground truth when using as an input to the energy optimization algorithm. Another important finding is that the energy profiles erroneously ranked as similar by popular statistical approaches can cause a significant amount of energy loss when used for the energy optimization of the application.

6.7 Summary

In this work, we presented a novel similarity measuring technique which takes into account the underlying energy consumption trend of the energy profiles. The proposed method captures the qualitative differences of the energy consumption behavior of energy profiles and ranks them based on their similarity with the ground truth. It effectively addresses the challenge of determining the goodness of application energy profiles on multicore computing nodes omnipresent in cloud infrastructures, supercomputers, data centers, and heterogeneous computing clusters where the shapes of energy profiles are non-smooth and non-linear. We compared the proposed method with popular statistical approaches, which are used to estimate the similarity between energy profiles, for a diverse set of 235 energy profiles (constructed on multicore heterogeneous hybrid computing platforms using state-of-the-art energy measurement techniques such as integrated power sensors, external power meters, or energy predictive models using PMCs as predictor variables). We demonstrated that the use of the state-of-the-art similarity approaches instead of the proposed one in the energy optimization loop leads to significant energy losses (up to 54% in our case).

We also showed that the proposed method can help determine whether the prediction model (that is employed to construct the profile) includes some extraneous contributor that does not reflect the energy consumption by the application or lacks some essential contributor to the energy consumption by the application. This finding further helps in determining whether the calibration can improve the average and maximum errors, and Euclidean distance between the energy profiles (constructed with over-estimated or under-estimated energy measurements) and the ground truth. Future work would include studying the efficiency of the proposed solution method in selecting the predictive model variables such as performance monitoring counters (PMCs) in order to improve their prediction accuracy.

Chapter 7

Summary, Current Picture and Future Directions

7.1 Summary

Energy is identified by IEA as a major contributor to climate change [11, 12]. Energy efficiency is central to the efforts of IEA to combat climate change [24]. The ICT are predicted in the worst-case scenario to use up to 51% of global electricity in 2030 and contribute up to 23% of globally released greenhouse gas emission [2]. Therefore, energy efficiency in ICT is becoming a grand technological challenge and is now a first-class design constraint in all computing settings [15, 16].

Energy efficiency in ICT can be achieved at the hardware level (or system level) and software level (or application level). While the system-level energy optimization approach focuses on minimizing the energy consumption of the whole node by employing techniques such as the clock and power gating, dynamic voltage and frequency scaling, etc. [166, 167, 168], application-level energy optimization techniques use application-level models and model variables such as workload distribution, number of processes, number of threads, etc. [27, 18] as decision variables for energy optimization of applications. However, this approach is comparatively understudied and is the main focus of this thesis.

Modern HPC, cloud computing systems, and data centers are highly heterogeneous containing nodes where a multicore CPU is tightly integrated with accelerators. Accurate measurement of energy consumption during an application execution is pivotal to energy minimization techniques at software level and many other interesting applications including the energy-centric performance analysis, auto-tuning, energy aware dynamic task scheduling. However, a fundamental challenge is how to measure the energy consumption by an application during its execution accurately and reliably. It is even more challenging for energy optimization of hybrid parallel applications consists of multiple kernels (generally speaking, multi-threaded), to accurately estimate the energy consumption by individual application components running in parallel on different compute devices of modern heterogeneous hybrid computing platforms.

There are three popular approaches to providing it: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors and (c) Energy predictive models. The first approach employing the system-level physical measurements using external power meters is considered to be accurate at system level. However, it lacks the ability to provide fine-grained component-level decomposition of the energy consumption of an application. This is a serious drawback.

On-chip integrated power sensors, on the other hand, provide fine-grain component level power consumption details. However, there are some issues with the power data values provided by these vendor-specific libraries. The fundamental issue with this measurement approach is the lack of information about how a power reading is determined during the execution of an application. Apart from accuracy, the other issues include the lack of details on update frequency of power readings, portability, poor documentation, etc. Therefore, a good understanding and validation of energy measurement instrumentation systems and on-chip power sensors is necessary for trusting and employing their readings in application-level energy optimization techniques.

Energy predictive models are typically trained using a large suite of diverse benchmarks and validated against a subset of the benchmark suite and some real-life applications. While the general accuracy of the models has been widely researched, their application-specific accuracy, however, has not been studied and therefore needs further validation.

We bridge this gap in this thesis, by proposing a comprehensive methodology in Chapter 3 to compute the energy consumption by an application reliably and accurately using the system-level measurements. In this chapter, HCLWattsUp API is presented which gathers the power-readings from the power-meter and return the energy consumption by the application during its execution within user-defined precision settings and confidence interval. The measurements are highly accurate within the accuracy provided by the power-meter used to measure the power consumption. The modern sophisticated revenue-graded power meters such as Yokogawa WT5000 [98] offers the basic power accuracy of up to $\pm 0.03\%$ with a very high sampling rate of 10M per second. To ensure the reliability of the measurements, a detailed statistical approach and number of precautions are followed. To summarize, using the presented methodology and HCLWattsUp API, one can determine the dynamic energy consumption by the application in an accurate and reliable manner during its execution. The approach is generic and thus can be applied to any power meter connected with a computing node.

After that, in chapter 4, a first comprehensive study is presented comparing the accuracy of state-of-the-art integrated on-chip power sensors and energy predictive models against system-level physical measurements using external power meters, which we consider to be the ground truth. The average error of the dynamic energy profiles obtained using on-chip power sensors is found to be as high as 73% and the maximum reaches 300% for two scientific applications, matrix-matrix multiplication and 2D fast Fourier transform for a wide range of problem sizes. The applications are executed on three modern Intel multicore CPUs, two Nvidia GPUs and a Xeon Phi accelerator. The average error of the energy predictive models employing PMCs as predictor variables is found to be as high as 32% and the maximum reaches up to 100% for a diverse set of seventeen benchmarks executed on two Intel multicore CPUs (one Haswell and the other Skylake). It is demonstrated that using inaccurate energy measurements provided by on-chip sensors for dynamic energy optimization can result in significant energy losses up to 84%.

In chapter 5, a novel method is proposed for accurate estimation of the application component-level energy consumption employing system-level power measurements with power meters. The proposed method address two challenges for energy optimization of hybrid parallel applications running on modern heterogeneous NUMA computing platforms: i) Accurate modelling of the energy consumption of individual application components when executing a hybrid application in parallel on multiple compute devices of a heterogeneous computing node, and ii) Accurate modelling of the energy consumption of different applications executing in parallel on a multi-socket multicore CPU platform. The proposed method is validated on a cluster of two hybrid heterogeneous computing nodes using three hybrid parallel applications matrix-matrix multiplication, 2D fast Fourier transform and gene sequencing. The experiments demonstrate a high estimation accuracy of the proposed method, with the average estimation error ranging between 2% and 5%. The average error demonstrated by the state-of-the-art estimation methods for the same experimental setup ranges from 15% to 75%, while the maximum reaches 178%. It is found that the use of the state-of-the-art estimation methods instead of the proposed one in the energy optimization loop leads to significant energy losses (up to 45% in our case). An important finding is that if an application is executed on some of the CPU cores of a socket then the base energy (or the energy due to leakage power) of idle CPU cores can contribute significantly to the total energy consumption by the CPU, and can even exceed the dynamic energy consumption by the active CPU cores in some cases.

Accurate energy profiles are essential to optimization of parallel applications for energy through workload distribution. Since there are many model-based methods available for efficient construction of energy profiles, we need an approach to measure the goodness of the profiles compared with the ground-truth profile, which is usually built by a time-consuming but reliable method. Correlation coefficient and relative error are two such popular statistical approaches, but they assume that profiles be linear or at least very smooth functions of workload size. This assumption does not hold true in the multicore era. Due to the complex shapes of energy profiles of applications on modern multicore platforms, the statistical methods can often rank inaccurate energy profiles higher than the more accurate ones. It leads to the significant energy losses (up to 54% in our case) when employing such inaccurate profiles in the energy optimization loop of an application.

In chapter 6, we present the first method specifically designed for goodness measurement of energy profiles. First, it analyses the underlying energy consumption trend of each energy profile and removes the profiles that exhibit a trend different from that of the ground truth. Then, it ranks the remaining energy profiles using the Euclidean distances between them and the ground truth. We demonstrate that the proposed method is more accurate than the popular statistical approaches and can save a significant amount of energy. It effectively addresses the challenge of determining the goodness of application energy profiles on multicore computing nodes omnipresent in cloud infrastructures, supercomputers, data centers, and heterogeneous computing clusters where the shapes of energy profiles are non-smooth and non-linear.

We also showed that the proposed method can help determine whether the prediction model (that is employed to construct the profile) includes some extraneous contributor that does not reflect the energy consumption by the application or lacks some essential contributor to the energy consumption by the application, thereby inaccurately capturing the energy consumption behavior by the application. This finding further helps in determining whether the calibration can improve the average and maximum errors, and Euclidean distance between the energy profiles (constructed with over-estimated or under-estimated energy measurements) and the ground truth.

In conclusion, this thesis emphasizes the importance of accurate measurement of energy consumption by an application during its execution on modern multicore (and hybrid heterogeneous) computing platforms for its energy optimization purposes. It calls attention to the inadequacies and inaccuracy of state-of-the-art approaches in measuring the energy consumption by an application and highlights the implications (in terms of energy losses) of employing them in energy optimization loop of an application. For application-level energy optimization purposes, the methodologies proposed in this thesis suggest the solutions to the challenges of accurate measurement of energy consumption by an application during its execution and to determine their goodness in the presence of different measurement approaches, and thus pave the way for more green-computing in general and for energy-efficient high performance computing in particular.

Research Impact – publications emanated from this thesis: The methodology to determine the component-level energy measurements using the system level measurements (Chapter3) laid the basis of following publications. The results of comparative analysis of methods for measurements of energy of computing (Chapter 4) are published in [37]. The additive energy modelling methodology for accurate estimation of the application component-level energy consumption employing system-level power measurements with power meters presented in Chapter 5 is published in [21]. This additive energy modelling methodology is used for the optimization of data-parallel applications on heterogeneous HPC platforms for dynamic energy through workload distribution published in [39], and the extended version of this work is published in [30]. The same additive energy modelling methodology also laid the basis of the following studies: a) the bi-objective optimization for performance and energy on heterogeneous processors [20], and b) a comprehensive study of linear energy predictive models employing utilization and PMCs on modern multicore CPUs [169]. The methodologies presented in Chapters 3 and 5 are also followed in the study [170] to determine the energy of communication of performance of optimal matrix partitioning for parallel computing on a hybrid heterogeneous server. The methodology to measure the goodness of energy profiles of parallel application presented in Chapter 6 is published in [77].

One of the recommendations by the reference [37] is that linear energy predictive models can be employed in the optimization of applications for dynamic energy provided they meet the following criteria: (a) Model parameters employed in the models must be deterministic and reproducible, (b) Model parameters are selected based on physical significance originating from fundamental physical laws such as conservation of energy of computing. Both the criteria are contained in the *additivity* test proposed in Reference [52]. In [17], we elucidated the challenges to energy and performance optimization of parallel application introduced by the advent of multi-core architectures. In [78], we study how that the accuracy of state-of-the-art energy predictive models can be improved by selecting performance monitoring counters based on a property of additivity. In [54], we compare following two types of energy predictive models: i) linear regression, and ii) sophisticated statistical learning models (random forest and neural network)) employing PMCs that are selected by different criterion such as additivity, correlation coefficient and principal component analysis. For all the aforementioned research works, the component-level energy consumption is determined using the system-level power measurements by following the methodology presented in Chapter 3.

7.2 Future Works

Following can be the potential future works which could be relevant in the extension of this thesis:

1. The energy modeling of rather complex hybrid applications consists of a number of kernels (application-components) that are tightly coupled or cannot be scaled neatly as a function of problem size.
2. Determine the dynamic energy consumption by the computing elements which are not independently powered.
3. An approach viable to optimizing the dynamic energy consumption by the CPU socket when running an application on some of the CPU cores, by optimizing the contribution by the idle CPU cores.
4. A comprehensive study to analyze the energy consumption by intra-node and inter-node communication of a hybrid application.
5. A comprehensive study providing the insight of decomposition of energy consumption by a hybrid application's intra-node communication and computation.
6. Using the proposed similarity matching approach in Chapter 6 as an aide to the selection criterion of model parameters (PMCs) set to construct the energy predictive model.
7. A comprehensive study to explore the application of proposed goodness measuring approach (TSM) in other fields such as time-series analysis for closely related problem of similarity matching, and its comparison with the stat-of-the-art similarity finding approaches employed in such fields.
8. A middle-ware that unifies the component-level decomposition of energy consumption (Chapter 3), the additive energy modelling (Chapter 5) and the similarity matching approach for accurate measurement of the application-component level energy consumption (Chapter 6) with a model-based workload partitioning algorithm to optimize the energy consumption of a hybrid application executing on a hybrid heterogeneous computing platform.
9. An energy-aware scheduling of virtual machines deployment and resource allocations using the methodology presented in Chapter 5.
10. The results obtained with HCLWattsUp can be compared with a measurement system designed specifically for HPC infrastructure.

7.3 Current Picture, Recommendations and Future Directions

Finally, we conclude with presenting the current picture of energy of computing in general, lessons learned from this thesis and our recommendations for the use of on-chip sensors and energy predictive models, and some likely future directions of the methods for measurements of energy of computing.

7.3.1 Current Picture

In modern data-driven world, HPC is the core of the industrial, societal, and scientific advancements. For decades, the performance maximization has been the chief concern of both the hardware architects and the software developers. The hardware acceleration and the use of co-processors together with CPU are becoming a popular choice to gain the performance boost while keeping the power budget low. This includes both the new customized hardware for particular application domain such as TPU, VPU and NPU; and the modifications in existing platforms such as Xeon Phi co-processors, general purpose GPU and FPGA. Such accelerators together with main processors and memory, constitute a heterogeneous system and are greatly prevalent now in modern ICT devices ranging from handheld mobile devices to HPC systems for many reasons such as growing accelerated computational needs and power constraints.

As a result of Dennard scaling breakdown, energy efficiency is becoming an equally important design concern with performance in ICT. The focus of maximizing the performance of HPC in terms of completing the hundreds of trillion floating-point operations per second (FLOPS) has led the supercomputers to consume an enormously high amount of energy in terms of electricity and for cooling down purposes. Current HPC systems are already consuming Megawatts of energy. For example, the world's most powerful supercomputer, as of 2019, Summit consumes around 13 Megawatts of power [1] which is roughly equivalent to the power draw of roughly over 10000 households. As per TOP500 list of November 2019, the top 4 supercomputers consume a total of over 50 mega watts (MW) of electricity. The power consumption by the top 10 in the list has increased over a span of 10 years from about 25 MW in 2008 to around 83 MW in 2018 which is an increase of 232%. Because of such high power consumption, future HPC systems are highly likely to be power constrained. For example, DOE aims to deploy an exascale supercomputer capable of performing 1 million trillion (10^{18}) floating-point operations per second in a power envelope of 20-30 megawatts [22]. Owing to the superior performance per watt in state-of-art accelerators, contemporary computational clusters, data centres and supercomputers are getting highly heterogeneous so that 102 out of top 500 supercomputers [1] are heterogeneous. Therefore, improving performance and energy consumption has turned into one major issue in heterogeneous HPC. It is becoming a grand technological challenge and is now a first-class design constraint in all computing settings ranging from handheld mobile [15, 16].

Integrated on-chip power sensors are now prevalent in mainstream processors and accelerators such as Intel and AMD Multicore CPUs, Nvidia GPUs, and Xeon Phis. There are vendor-specific libraries available to acquire the power data from these integrated power-sensors. However, based on our study, we can not recommend use of state-of-the-art on-chip sensors (RAPL for multicore CPUs, NVML for GPUs, MPSS for Xeon Phis). The fundamental issue with this measurement approach is the lack of information about how a power reading for a component is determined during the execution of an application utilizing the component. The lack of information on accuracy of power data acquired using vendor-specific libraries (such as in case of RAPL, and Intel-MPSS) is another fundamental issue. While the accuracy of this information is reported in the case of NVML, experimental results demonstrate that practical accuracy is worse. Moreover, the dynamic energy profile patterns of the on-chip sensors differ significantly from the patterns obtained using the ground truth, which suggests that the measurements using on-chip sensors do not capture the holistic picture of the dynamic energy consumption during an application execution. Furthermore, owing to the nature of the deviations of the energy measurements provided by on-chip sensors from the ground truth, calibration can not improve the accuracy of the on-chip sensors to an extent that can allow them to be used in optimization of applications for dynamic energy. At the same time, we observed that the energy measurements reported by the on-chip sensors are deterministic and reproducible and, therefore can be used as parameters in energy predictive models.

Voltage change of the components in modern computing hardware are supported through voltage regulators (VRs). VR performs two following major functions in providing voltage to a computing device: i) it stabilizes the supplied voltage, and ii) it changes the supplied voltage according to the needs of the device. Several power saving mechanisms are provided to control and configure the power consumption in modern computing hardware using the VRs. Integrated on-chip sensors typically use VRs to determine the power consumption by an component. Hence, the poor accuracy of on-chip power sensors can be attributed mainly to employing inaccurate VRs. It is reported that VRs from the same manufacturer lot may exhibit different accuracies [46]. Instead of higher accurate VRs (such as within an accuracy of $\pm 5\%$), less accurate VR (for example within an accuracy of $\pm 20\%$), are integrated by chip manufacturers for cost saving purposes [46]. However, to compensate the reported inaccuracies in for current output (IMON) from a VR to a processor, an approach is recently proposed to use a programmable load line from BIOS instead of actual implemented load line. This programmed load line value adds an offset to the determined inaccuracy of the VR to increase its accuracy [46].

Energy predictive models using performance monitoring counters emerged as a dominant measurement method. Its main advantage compared to the system-level physical measurements using power meters is the fine-grained decomposition of energy consumption during the execution of an application. This approach, however, has several shortcomings:

- **Accuracy**: The accuracy of energy predictive model is very poor, in general.
- **Reliability**: Model parameters (PMCs) in most cases are not reproducible.
- **Standardization**: There is a lack of consensus among the research works which report prediction accuracies ranging from poor to excellent.
- **Model parameter selection criterion**: The accuracy of models is dependent on the selection of the PMCs used as predictor variables. It is not a trivial task to find a best set of PMCs which reflects the energy consumption for all types of workloads in equally effective way.
- **Compliance with fundamental physical laws**: In general, the model parameters (PMCs) are selected following the techniques such as principal component analysis or on the basis of their high positive correlation with energy consumption without any insight of the physical significance of the model variables originating from fundamental physical laws such as conservation of energy of computing.
- **Implementation Complexity**: The model construction process is highly complex and requires a tremendous programming effort and time to automate and collect all the PMCs available on a computing platform.

- **Portability**: An energy predictive model purely based on PMCs lacks portability. This is because all the PMCs available for a CPU processor may not be present in a GPU processor due to inherent architectural differences, or even for the next-generation CPU processor from the architecture space.

These limitations make them a poor choice for modelling a hybrid application executing in parallel on different compute devices of a heterogeneous computing platform. These are important discoveries to keep in mind when basing new research using the integrated on-chip power sensors and predictive models based on PMCs.

Another fundamental yet understudied challenge for energy-efficient computing is how to measure the goodness of energy profiles? While there is a plethora of different methods and approaches have been proposed to solve the goodness measuring (similarity/pattern matching) problem in many other fields such as data mining, time series similarity analysis, and graph (matching) theory, it is comparatively less-studied for the energy of computing. The two popular statistical approaches (correlation coefficient and relative error) employed to determine the goodness of energy profiles of an application mainly rely on the assumption that the profiles be linear or at least very smooth functions of workload size. This assumption does not hold true in the multicore era. Due to the complex shapes of energy profiles of applications on modern multicore platforms, the statistical methods can often rank inaccurate energy profiles higher than more accurate ones and employing such profiles in the energy optimization of an application leads to significant energy losses.

7.3.2 Recommendations and Future Directions in General

Since system-level physical measurements based on power meters are accurate and reliable, we recommend using the approach presented in Chapter 3 as the fundamental building block for the fine-grained device-level decomposition of the energy consumption during the execution of an application. Additive Energy modelling approach (AnMoHA) proposed in chapter 5 can be used to construct the accurate energy profiles of individual application (components) of (a hybrid) parallel executing application(s) on multiple independent compute devices.

We envisage hardware vendors maturing their on-chip sensor technology to an extent where energy optimization programmers will be provided necessary information of how a power measurement is determined for a component, the frequency or sampling rate of the measurements, its reported accuracy and finally how to programmatically obtain this measurement within a sufficient accuracy and low overhead. Furthermore, we envisage the chip manufacturers to adopt such an approach which addresses the poor reliability and accuracy of VRs and the power values provided by the integrated on-chip sensors. Furthermore, we envisage the standardization of programmatic interfaces of current vendor-specific libraries and tools to acquire the power-data from integrated power sensors. As a result, we envisage that the integrated on-chip power sensors would emerge as a predominant method for measuring the energy of computing once the chip-makers address the poor reliability and accuracy of power data provided by them.

Linear energy predictive models can be employed in the optimization of applications for dynamic energy provided they meet the following criteria: (a) Model parameters employed in the models must be deterministic, (b) Model parameters are selected based on the physical significance originating from fundamental physical laws such as conservation of energy of computing. Both the criteria are contained in the *additivity* test proposed in Reference [52]. Use of parameters with high additivity improves the prediction accuracy of the model. Additivity test can also be employed to select parameters for machine learning (or black box) methods such as neural networks, random forests, etc., provided the methods use linear functional building blocks internally [78, 54]. While there is experimental evidence demonstrating good accuracy for these types of models, a sound theoretical analysis is lacking. At this point, we do not recommend the use of non-linear energy predictive models since they lack serious theoretical and experimental analysis. It can be one of the future research directions for researchers.

We find that high-level model parameters designed by combining PMCs (using functions based on physical significance with dynamic energy) may be deterministic instead of individual PMCs, which are raw counters. PMCs traditionally have been developed to aid low-level performance analysis and tuning but have been widely adopted for energy predictive modeling. We would call the high-level model parameters, energy monitoring counts (EMCs), that are discovered from insights based on the theory of energy predictive models for computing proposed in [53] and that are ideal for employment as predictor variables in energy predictive models.

Finally, we recommend using such an approach instead of correlation coefficient and relative error that takes into account the qualitative differences of the energy profiles when measuring their goodness with the ground-truth profile, which is usually built by a time-consuming but reliable method. TSM proposed in Chapter 6 is the first attempt to address this fundamental yet understudied problem. We envisage more research works to contribute by proposing the techniques while considering different service-related constraints such as energy or/and performance optimization of applications executing on modern multicore architectures.

Bibliography

[1] Top500, “Top500 list,” November, 2019. https://www.top500.org/lists/2019/11/.

[2] A. Andrae and T. Edler, “On global electricity usage of communication technology: Trends to 2030,” *Challenges*, vol. 6, p. 117157, Apr 2015.

[3] H. P. Computing and Q. T. U. C.2), “Shaping europe’s digital future, high-performance computing,” November 2019. https://ec.europa.eu/digital-single-market/en/policies/high-performance-computing.

[4] T. W. H. O. of the Press Secretary, “Executive order – creating a national strategic computing initiative,” 2015. Executive Order by President Barack Obama https://obamawhitehouse.archives.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative.

[5] F. Bührer, F. Fischer, G. Fleig, A. Gamel, M. Giffels, T. Hauth, M. Janczyk, K. Meier, G. Quast, B. Roland, and et al., “Dynamic virtualized deployment of particle physics environments on a high performance computing cluster,” *Computing and Software for Big Science*, vol. 3, May 2019. http://dx.doi.org/10.1007/s41781-019-0024-5.

[6] G. E. Moore, “Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff.,” *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, 2006.

[7] R. H. Dennard, F. H. Gaensslen, H. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, “Design of ion-implanted mosfet’s with very small physical dimensions,” *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, 1974.

[8] H. Sutter, “The free lunch is over: A fundamental turn toward con- currency in software,” *Dr. Dobb’s Journal*, vol. 30, p. 202210, March 2005.

[9] W. Haensch, E. J. Nowak, R. H. Dennard, P. M. Solomon, A. Bryant, O. H. Dokumaci, A. Kumar, X. Wang, J. B. Johnson, and M. V. Fischetti, “Silicon cmos devices beyond scaling,” *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 339–361, 2006.

[10] M. M. Waldrop, “The chips are down for moore’s law,” *Nature*, vol. 530, pp. 144–147, February 2016.

[11] IEA, “Climate change,” 2020. https://www.iea.org/topics/climate-change.

[12] IEA, “Global energy & co2 status report 2019,” 2019. `https://www.iea.org/reports/global-energy-and-co2-status-report-2019`.

[13] N. Jones, “How to stop data centres from gobbling up the worlds electricity,” *Nature*, vol. 561, pp. 163–166, 2018.

[14] ATAG, “Facts and figures,” january 2020. `https://www.atag.org/facts-figures.html`.

[15] K. O’brien, I. Pietri, R. Reddy, A. Lastovetsky, and R. Sakellariou, “A survey of power and energy predictive models in hpc systems and applications,” *ACM Comput. Surv.*, vol. 50, pp. 37:1–37:38, June 2017.

[16] G. Fagas, J. P. Gallagher, L. Gammaitoni, and D. J. Paul, “Energy challenges for ict,” in *ICT - Energy Concepts for Energy Efficiency and Sustainability* (G. Fagas, L. Gammaitoni, J. P. Gallagher, and D. J. Paul, eds.), ch. 1, Rijeka: IntechOpen, 2017. `https://doi.org/10.5772/66678`.

[17] A. Lastovetsky, M. Fahad, H. Khaleghzadeh, S. Khokhriakov, R. Reddy, A. Shahid, L. Szustak, and R. Wyrzykowski, “How pre-multicore methods and algorithms perform in multicore era,” in *International Conference on High Performance Computing*, pp. 527–539, Springer, 2018.

[18] A. Lastovetsky and R. R. Manumachu, “New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1119–1133, 2017.

[19] R. R. Manumachu and A. Lastovetsky, “Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy,” *IEEE Transactions on Computers*, vol. 67, no. 2, pp. 160–177, 2018.

[20] H. Khaleghzadeh, M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, “Bi-objective optimization of data-parallel applications on heterogeneous hpc platforms for performance and energy through workload distribution,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 543–560, 2021.

[21] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, “Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements,” *IEEE Access*, pp. 1–1, 2020.

[22] DOE, “Preliminary conceptual design for an exascale computing initiative,” 2014. `https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20141121/Exascale_Preliminary_Plan_V11_sb03c.pdf`.

[23] Green500, “The green500,” 2020. `https://www.top500.org/green500/`.

[24] IEA, “Iea sets out pillars for success at cop21,” june 2015. `https://www.iea.org/news/iea-sets-out-pillars-for-success-at-cop21`.

[25] J. Hsu, “Three paths to exascale supercomputing,” *IEEE Spectrum*, vol. 53, no. 1, pp. 14–15, 2016.

[26] A. Chakrabarti, S. Parthasarathy, and C. Stewart, “A pareto framework for data analytics on heterogeneous systems: Implications for green energy usage and performance,” in *Parallel Processing (ICPP), 2017 46th International Conference on*, pp. 533–542, IEEE, 2017.

[27] J. Lang and G. Rünger, “An execution time and energy model for an energy-aware execution of a conjugate gradient method with CPU/GPU collaboration,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 9, pp. 2884 – 2897, 2014.

[28] J. Demmel, A. Gearhart, B. Lipshitz, and O. Schwartz, “Perfect strong scaling using no additional energy,” in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pp. 649–660, May 2013.

[29] B. Subramaniam and W. Feng, “Statistical power and performance modeling for optimizing the energy efficiency of scientific computing,” in *2010 IEEE/ACM Int'l Conference on Green Computing and Communications and Int'l Conference on Cyber, Physical and Social Computing*, pp. 139–146, Dec 2010.

[30] H. Khaleghzadeh, M. Fahad, R. R. Manumachu, and A. Lastovetsky, “Novel data partitioning algorithm for dynamic energy optimisation on heterogeneous hpc platforms,” *Concurrency and Computation: Practice and Experience*, 2020.

[31] V. Cardellini, A. Fanfarillo, and S. Filippone, “Heterogeneous sparse matrix computations on hybrid GPU/CPU platforms.,” in *PARCO*, pp. 203–212, 2013.

[32] J. Colaço, A. Matoga, A. Ilic, N. Roma, P. Tomás, and R. Chaves, “Transparent application acceleration by intelligent scheduling of shared library calls on heterogeneous systems,” in *Parallel Processing and Applied Mathematics*, pp. 693–703, Springer, 2013.

[33] A. L. Lastovetsky and R. Reddy, “Data partitioning with a realistic performance model of networks of heterogeneous computers,” in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, p. 104, IEEE, 2004.

[34] M. Radmanović, D. Gajić, and R. Stanković, “Efficient computation of galois field expressions on hybrid CPU-GPU platforms.,” *Journal of Multiple-Valued Logic & Soft Computing*, vol. 26, 2016.

[35] A. Lastovetsky and R. Reddy, “Data partitioning with a functional performance model of heterogeneous processors,” *International Journal of High Performance Computing Applications*, vol. 21, no. 1, pp. 76–90, 2007.

[36] H. Khaleghzadeh, R. R. Manumachu, and A. Lastovetsky, “A novel data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous HPC platforms,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 10, pp. 2176–2190, 2018.

[37] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, “A comparative study of methods for measurement of energy of computing,” *Energies*, vol. 12, no. 11, 2019. `https://www.mdpi.com/1996-1073/12/11/2204`.

[38] Heterogeneous Computing Laboratory, University College Dublin, “HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter,” 2020. `https://csgitlab.ucd.ie/ucd-hcl/hclwattsup`.

[39] H. Khaleghzadeh, M. Fahad, R. R. Manumachu, and A. Lastovetsky, “Optimization of data-parallel applications on heterogeneous hpc platforms for dynamic energy through workload distribution,” in *Euro-Par 2019: Parallel Processing Workshops* (U. Schwardmann, C. Boehme, D. B. Heras, V. Cardellini, E. Jeannot, A. Salis, C. Schifanella, R. R. Manumachu, D. Schwamborn, L. Ricci, O. Sangyoon, T. Gruber, L. Antonelli, and S. L. Scott, eds.), (Cham), pp. 320–332, Springer International Publishing, 2020.

[40] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, “Energy consumption estimation in embedded systems,” *IEEE Transactions on Instrumentation and Measurement*, vol. 57, pp. 797–804, April 2008.

[41] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, “Power-Management architecture of the intel microarchitecture Code-Named sandy bridge,” *IEEE Micro*, vol. 32, pp. 20–27, March 2012.

[42] I. Corporation, “Intel xeon phi coprocessor system software developers guide,” 06 2014. `https://software.intel.com/sites/default/files/managed/09/07/xeon-phi-coprocessor-system-software-developers-guide.pdf`.

[43] Nvidia, “Nvidia management library: NVML reference manual,” 10 2020. `https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf`.

[44] A. M. Devices, “Bios and kernel developer’s guide (bkdg) for amd family 15h models 00h-0fh processors,” 2012. `https://www.amd.com/system/files/TechDocs/42301_15h_Mod_00h-0Fh_BKDG.pdf`.

[45] C. Gough, I. Steiner, and W. Saunders, *Energy Efficient Servers Blueprints for Data Center Optimization*. Apress, 2015. isbn:978-1-4302-6637-2.

[46] T. Rahal-arabi and M. S. Bashir, “Programmable imon accuracy in power systems,” March 2019. `https://patentscope.wipo.int/search/en/detail.jsf?docId=US239827974&_cid=P21-KAVQ9S-45020-1` and US Patent 20190094948.

[47] M. Burtscher, I. Zecena, and Z. Zong, “Measuring gpu power with the k20 built-in sensor,” in *Proceedings of Workshop on General Purpose Processing Using GPUs*, GPGPU-7, pp. 28:28–28:36, ACM, 2014.

[48] I. Corporation, “Intelligent platform management interface spec,” 10 2013. `https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ipmi-second-gen-interface-spec-v2-rev1-1.pdf`.

[49] I. Corporation, "Dcmi – data center manageability interface specification," 08 2011. `https://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/dcmi-v1-5-rev-spec.pdf`.

[50] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *In Proceedings of Workshop on Modeling, Benchmarking, and Simulation*, pp. 70–77, 2006.

[51] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, pp. 12–12, USENIX Association, 2011.

[52] A. Shahid, M. Fahad, R. Reddy, and A. Lastovetsky, "Additivity: A selection criterion for performance events for reliable energy predictive modeling," *Supercomputing Frontiers and Innovations*, vol. 4, no. 4, 2017.

[53] A. Shahid, M. Fahad, R. R. Manumachu, and A. Lastovetsky, "Energy of computing on multicore cpus: Predictive models and energy conservation law," 2019.

[54] A. Shahid, M. Fahad, R. R. Manumachu, and A. Lastovetsky, "A comparative study of techniques for energy predictive modelling using performance monitoring counters on modern multicore cpus," *IEEE Access*, vol. 8, pp. 143306–143332, 2020.

[55] J. Treibig, G. Hager, and G. Wellein, "Likwid: A lightweight performance-oriented tool suite for x86 multicore environments," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pp. 207–216, IEEE, 2010.

[56] J. W. Kuzma, *Basic statistics for the health sciences (3rd ed.)*. Mayfield Publishing Company, 1280 Villa Street, Mountain View, CA 94041, 1998. isbn:1-55934-951-4.

[57] M. Abramowitz, *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., 31 E. Second St. Mineola, NY, United States, 1974. isbn:0486612724.

[58] W. Dargie, "A stochastic model for estimating the power consumption of a processor," *IEEE Transactions on Computers*, vol. 64, no. 5, 2015.

[59] Z. Zhou, J. H. Abawajy, F. Li, Z. Hu, M. U. Chowdhury, A. Alelaiwi, and K. Li, "Fine-grained energy consumption model of servers based on task characteristics in cloud data center," *IEEE access*, vol. 6, pp. 27080–27090, 2018.

[60] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *Performance analysis of systems and software (ISPASS), 2013 IEEE international symposium on*, pp. 194–204, IEEE, April 2013.

[61] B. Subramaniam and W.-c. Feng, "Towards energy-proportional computing for enterprise-class server workloads," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE 13, (New York, NY, USA), p. 1526, Association for Computing Machinery, 2013. `https://doi.org/10.1145/2479871.2479878`.

[62] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the intel haswell processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pp. 896–904, May 2015.

[63] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "Rapl in action: Experiences in using rapl for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, pp. 9:1–9:26, Mar. 2018.

[64] J.-A. Rico-Gallego, J.-C. Díaz-Martín, and A. L. Lastovetsky, "Extending τ-lop to model concurrent MPI communications in multicore clusters," *Future Gener. Comput. Syst.*, vol. 61, p. 6682, Aug. 2016. `https://doi.org/10.1016/j.future.2016.02.021`.

[65] B. Li and L. Han, "Distance weighted cosine similarity measure for text classification," in *Intelligent Data Engineering and Automated Learning – IDEAL 2013* (H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, and X. Yao, eds.), (Berlin, Heidelberg), pp. 611–618, Springer Berlin Heidelberg, 2013.

[66] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS94, p. 359370, AAAI Press, 1994.

[67] T. Nakamura, K. Taki, H. Nomiya, K. Seki, and K. Uehara, "A shape-based similarity measure for time series data with ensemble learning," *Pattern Analysis and Applications*, vol. 16, pp. 535–548, 2013.

[68] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., 500 Sansome St. San Francisco, CA, United States, 1976. isbn:978-0-8162-1104-3.

[69] Q. Yu, L. Jibin, and L. Jiang, "An improved arima-based traffic anomaly detection algorithm for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, p. 9653230, 2016. `https://doi.org/10.1155/2016/9653230`.

[70] D. Piccolo, "A distance measure for classifying arima models," *Journal of Time Series Analysis*, vol. 11, no. 2, pp. 153–164, 1990. `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1990.tb00048.x`.

[71] M. Ramoni, P. Sebastiani, and P. Cohen, "Bayesian clustering by dynamics," *Machine Learning*, vol. 47, no. 1, pp. 91–121, 2002. `https://doi.org/10.1023/A:1013635829250`.

[72] E. A. Maharaj, "Cluster of time series," *Journal of Classification*, vol. 17, no. 2, pp. 297–314, 2000. `https://doi.org/10.1007/s003570000023`.

[73] H. Anton and C. Rorres, *Elementary Linear Algebra: Applications Version, 11th Edition*. John Wiley & Sons, Inc., Hoboken, New Jersey, United States, 2013. isbn:978-1-118-43441-3.

[74] T. Warren Liao, “Clustering of time series data-a survey,” *Pattern Recogn.*, vol. 38, p. 18571874, Nov. 2005. `https://doi.org/10.1016/j.patcog.2005.01.025`.

[75] F. Iglesias and W. Kastner, “Analysis of similarity measures in times series clustering for the discovery of building energy patterns,” 2013.

[76] A. Sanfeliu and K. Fu, “A distance measure between attributed relational graphs for pattern recognition,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, pp. 353–362, May 1983.

[77] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, “A novel statistical learning-based methodology for measuring the goodness of energy profiles of applications executing on multicore computing platforms,” *Energies*, vol. 13, no. 15, 2020. `https://www.mdpi.com/1996-1073/13/15/3944`.

[78] A. Shahid, M. Fahad, R. R. Manumachu, and A. Lastovetsky, “Improving the accuracy of energy predictive models for multicore cpus using additivity of performance monitoring counters,” in *Parallel Computing Technologies* (V. Malyshkin, ed.), (Cham), pp. 51–66, Springer International Publishing, 2019.

[79] M. J. Flynn, “Some computer organizations and their effectiveness,” *IEEE Transactions on Computers*, vol. C-21, no. 9, pp. 948–960, 1972.

[80] M. J. Flynn, “Very high-speed computing systems,” *Proceedings of the IEEE*, vol. 54, no. 12, pp. 1901–1909, 1966.

[81] H. Kung and C. E. Leiserson, “Systolic arrays (for vlsi),” *Sparse Matrix Proceedings - SIAM*, pp. 256–278, 1978.

[82] Y.-J. Xu, Z.-Y. Luo, X.-W. Li, L.-J. Li, and X.-L. Hong, “Leakage current estimation of cmos circuit with stack effect,” *Journal of Computer Science and Technology*, vol. 19, p. 708717, Sep 2004. `http://dx.doi.org/10.1007/BF02945598`.

[83] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, “Leakage current: Moore’s law meets static power,” *Computer*, vol. 36, no. 12, pp. 68–75, 2003.

[84] A. Agarwal, S. Mukhopadhyay, C. H. Kim, A. Raychowdhury, and K. Roy, “Leakage power analysis and reduction: models, estimation and tools,” *IEE Proceedings - Computers and Digital Techniques*, vol. 152, no. 3, pp. 353–368, 2005.

[85] K. DeVogeleer, G. Memmi, P. Jouvelot, and F. Coelho, “Modeling the temperature bias of power consumption for nanometer-scale cpus in application processors,” in *2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, pp. 172–180, 2014.

[86] A. Sarwar, “Cmos power consumption and cpd calculation,” Tech. Rep. SCAA035B, Texas Instrument, June 1997. `https://www.ti.com/lit/an/scaa035b/scaa035b.pdf?&ts=1589553055343`.

[87] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A framework for architectural-level power analysis and optimizations,” *SIGARCH Comput. Archit. News*, vol. 28, p. 8394, May 2000. `https://doi.org/10.1145/342001.339657`.

[88] S. Khokhriakov, R. R. Manumachu, and A. Lastovetsky, “Multicore processor computing is not energy proportional: An opportunity for bi-objective optimization for energy and performance,” *Applied Energy*, vol. 268, p. 114957, 2020. `http://www.sciencedirect.com/science/article/pii/S0306261920304694`.

[89] Xixhou Feng, Rong Ge, and K. W. Cameron, “Power and energy profiling of scientific applications on distributed systems,” in *19th IEEE International Parallel and Distributed Processing Symposium*, pp. 10 pp.–, 2005.

[90] R. Ge, X. Feng, S. Song, H. Chang, D. Li, and K. W. Cameron, “PowerPack: Energy profiling and analysis of high-performance systems and applications,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 658–671, May 2010.

[91] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, “Powermon: Fine-grained and integrated power monitoring for commodity computer systems,” in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, pp. 479–484, 2010.

[92] D. Bedard, A. Porterfield, R. Fowler, and M. Y. Lim, “Powermon 2: Fine-grained, integrated power measurement,” Tech. Rep. TR-09-04, RENCI, North Carolina, October 2009. `https://www.renci.org/wp-content/pub/techreports/TR-09-04.pdf`.

[93] AnalogDevices, “Adm1191.” `https://www.analog.com/en/products/adm1191.html#product-overview`.

[94] J. H. Laros, P. Pokorny, and D. DeBonis, “Powerinsight - a commodity power measurement capability,” in *2013 International Green Computing Conference Proceedings*, pp. 1–6, 2013.

[95] beagleboard.org, “Beaglebone.” `https://beagleboard.org/bone`.

[96] D. Hackenberg, T. Ilsche, J. Schuchart, R. Schöne, W. E. Nagel, M. Simon, and Y. Georgiou, “Hdeem: High definition energy efficiency monitoring,” in *Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing*, E2SC 14, p. 110, IEEE Press, 2014. `https://doi-org.ucd.idm.oclc.org/10.1109/E2SC.2014.13`.

[97] L. Brochard, R. Panda, D. DeSota, F. Thomas, and R. H. Bell, “Power and energy-aware processor scheduling,” in *Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering*, ICPE 11, (New York, NY, USA), p. 227234, Association for Computing Machinery, 2011. `https://doi.org/10.1145/1958746.1958780`.

[98] YOKOGAWA, “Wt5000 precision power analyzer.” `https://tmi.yokogawa.com/eu/solutions/products/power-analyzers/wt5000/`.

[99] Vernier, “Watts up pro.” `http://www.vernier.com/files/manuals/wu-pro.pdf`.

[100] Z. Z. P. P. Measurement, “Precision power analyzers.” https://www.zes.com/en/Products/Precision-Power-Analyzers.

[101] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, “RAPL: Memory power estimation and capping,” in *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 189–194, Aug 2010.

[102] I. Corporation, “Intel manycore platform software stack (Intel MPSS),” 06 2014. https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss.

[103] Perf Wiki, “perf: Linux profiling with performance counters,” 2017. https://perf.wiki.kernel.org/index.php/Main_Page.

[104] PAPI, “Performance application programming interface 5.4.1,” 2015. http://icl.cs.utk.edu/papi/.

[105] IntelPCM, “Processor counter monitor (pcm),” 2020. https://github.com/opcm/pcm.

[106] CUPTI, “CUDA profiling tools interface,” 2017. https://developer.nvidia.com/cuda-profiling-tools-interface.

[107] J. Mair, Z. Huang, and D. Eyers, “Manila: Using a densely populated pmc-space for power modelling within large-scale systems,” *Parallel Computing*, vol. 82, pp. 37–56, 2019.

[108] J. Haj-Yihia, A. Yasin, Y. B. Asher, and A. Mendelson, “Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems,” *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 13, no. 4, p. 56, 2016.

[109] S. Roy, A. Rudra, and A. Verma, “An energy complexity model for algorithms,” in *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS ’13, (New York, NY, USA), pp. 283–304, ACM, 2013.

[110] A. Lewis, S. Ghosh, and N.-F. Tzeng, “Run-time energy consumption estimation based on workload in server systems,” in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower’08, (Berkeley, CA, USA), pp. 4–4, USENIX Association, 2008.

[111] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani, “A methodology to predict the power consumption of servers in data centres,” in *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking*, e-Energy ’11, (New York, NY, USA), pp. 1–10, ACM, 2011.

[112] W. L. Bircher and L. K. John, “Complete system power estimation using processor performance events,” *IEEE Transactions on Computers*, vol. 61, pp. 563–577, Apr. 2012.

[113] T. Heath, B. Diniz, B. Horizonte, E. V. Carrera, and R. Bianchini, “Energy conservation in heterogeneous server clusters,” in *10th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP)*, pp. 186–195, ACM, 2005.

[114] S. Rivoire, P. Ranganathan, and C. Kozyrakis, “A comparison of high-level full-system power models,” in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower’08, USENIX Association, 2008.

[115] F. Bellosa, “The benefits of event-driven energy accounting in power-sensitive systems,” in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, ACM, 2000.

[116] C. Isci and M. Martonosi, “Runtime power monitoring in high-end processors: Methodology and empirical data,” in *36th annual IEEE/ACM International Symposium on Microarchitecture*, p. 93, IEEE Computer Society, 2003.

[117] B. C. Lee and D. M. Brooks, “Accurate and efficient regression modeling for microarchitectural performance and power prediction,” *SIGARCH Comput. Archit. News*, vol. 34, pp. 185–194, Oct. 2006.

[118] T. Li and L. K. John, “Run-time modeling and estimation of operating system power consumption,” *SIGMETRICS Perform. Eval. Rev.*, vol. 31, pp. 160–171, June 2003.

[119] X. Fan, W.-D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in *34th Annual International Symposium on Computer architecture*, pp. 13–23, ACM, 2007.

[120] K. Singh, M. Bhadauria, and S. A. McKee, “Real time power estimation and thread scheduling via performance counters,” *SIGARCH Comput. Archit. News*, vol. 37, pp. 46–55, July 2009.

[121] M. D. Powell, A. Biswas, J. S. Emer, S. S. Mukherjee, B. R. Sheikh, and S. Yardi, “CAMP: A technique to estimate per-structure power at run-time using a few simple parameters,” in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, pp. 289–300, Feb 2009.

[122] B. Goel, S. A. McKee, R. Gioiosa, K. Singh, M. Bhadauria, and M. Cesati, “Portable scalable per-core power estimation for intelligent resource management,” in *Portable, Scalable, per-Core Power Estimation for Intelligent Resource Management*, Green Computing Conference, 2010 International, 2010-08-16 2010.

[123] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, “Distributed systems meet economics: pricing in the cloud,” in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, USENIX Association, 2010.

[124] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, “Decomposable and responsive power models for multicore processors using performance counters,” in *Proceedings of the 24th ACM International Conference on Supercomputing*, pp. 147–158, ACM, 2010.

[125] R. Basmadjian and H. de Meer, “Evaluating and modeling power consumption of multi-core processors,” in *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pp. 1–10, May 2012.

[126] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 1, 2013.

[127] S. L. Xi, H. Jacobson, P. Bose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in McPAT and potential impacts on architectural studies," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pp. 577–589, IEEE, 2015.

[128] H. Hong, Sunpyand Kim, "An integrated GPU power and performance model," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, 2010.

[129] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of GPU kernels using performance counters," in *International Green Computing Conference and Workshops (IGCC)*, IEEE, 2010.

[130] S. Song, C. Su, B. Rountree, and K. W. Cameron, "A simplified and accurate model of power-performance efficiency on emergent gpu architectures," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pp. 673–686, May 2013.

[131] Y. S. Shao and D. Brooks, "Energy characterization and instruction-level energy model of intel's xeon phi processor," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 389–394, Sep. 2013.

[132] Z. Al-Khatib and S. Abdi, "Operand-value-based modeling of dynamic energy consumption of soft processors in fpga," in *Applied Reconfigurable Computing* (K. Sano, D. Soudris, M. Hübner, and P. C. Diniz, eds.), pp. 65–76, Springer International Publishing, 2015.

[133] Qing Wu, M. Pedram, and Xunwei Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 3, pp. 415–420, 2000.

[134] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-v power supply high-speed digital circuit technology with multithreshold-voltage cmos," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, 1995.

[135] Hailin Jiang, M. Marek-Sadowska, and S. R. Nassif, "Benefits and costs of power-gating technique," in *2005 International Conference on Computer Design*, pp. 559–566, 2005.

[136] I. Unified Extensible Firmware Interface (UEFI) Forum, "Advanced configuration and power interface (acpi) specification," January 2019. `https://uefi.org/sites/default/files/resources/ACPI_6_3_final_Jan30.pdf`.

[137] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[138] R. J. W. Dominik Brodowski, Nico Golde and V. Kumar, "Linux cpufreq." `https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt`.

[139] J. Haj-Yahya, E. Rotem, A. Mendelson, and A. Chattopadhyay, “A comprehensive evaluation of power delivery schemes for modern microprocessors,” in *20th International Symposium on Quality Electronic Design (ISQED)*, pp. 123–130, 2019.

[140] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E.-G. Talbi, A. Zomaya, and D. Tuyttens, “A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1497 – 1508, 2011.

[141] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012. Special Section: Energy efficiency in large-scale distributed systems.

[142] J. J. Durillo, V. Nae, and R. Prodan, “Multi-objective energy-efficient workflow scheduling using list-based heuristics,” *Future Generation Computer Systems*, vol. 36, pp. 221 – 236, 2014.

[143] J. Kołodziej, S. U. Khan, L. Wang, and A. Y. Zomaya, “Energy efficient genetic-based schedulers in computational grids,” *Concurrency: Practice and Experience*, vol. 27, pp. 809–829, Mar. 2015.

[144] J. M. Marszalkowski, M. Drozdowski, and J. Marszalkowski, “Time and energy performance of parallel systems with hierarchical memory,” *Journal of Grid Computing*, vol. 14, no. 1, pp. 153–170, 2016.

[145] R. Reddy Manumachu and A. L. Lastovetsky, “Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 4, p. e4958, 2019.

[146] F. Juarez, J. Ejarque, and R. M. Badia, “Dynamic energy-aware scheduling for parallel task-based application in cloud computing,” *Future Generation Computer Systems*, vol. 78, pp. 257 – 271, 2018. `http://www.sciencedirect.com/science/article/pii/S0167739X1630214X`.

[147] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” *Computer*, no. 12, pp. 33–37, 2007.

[148] D. Wong and M. Annavaram, “Knightshift: Scaling the energy proportionality wall through server-level heterogeneity,” in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 119–130, 2012.

[149] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, “Towards energy proportionality for large-scale latency-critical workloads,” in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 301–312, 2014.

[150] C.-H. Hsu and S. W. Poole, “Measuring server energy proportionality,” in *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, ICPE 15, (New York, NY, USA), p. 235240, Association for Computing Machinery, 2015. `https://doi.org/10.1145/2668930.2688049`.

[151] R. Sen and D. A. Wood, “Energy-proportional computing: A new definition,” *Computer*, vol. 50, no. 8, pp. 26–33, 2017.

[152] N. J. SALKIND, *Encyclopedia of Measurement and Statistics Spurious Correlation*, vol. 1. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks, California 91320, 2007. isbn:1-4129-1611-9.

[153] C.-L. Lin, “Hardness of approximating graph transformation problem,” in *Algorithms and Computation* (D.-Z. Du and X.-S. Zhang, eds.), (Berlin, Heidelberg), pp. 74–82, Springer Berlin Heidelberg, 1994.

[154] Asanovic, Krste and Bodik, Ras and Catanzaro, Bryan Christopher and Gebis, Joseph James and Husbands, Parry and Keutzer, Kurt and Patterson, David A. and Plishker, William Lester and Shalf, John and Williams, Samuel Webb and Yelick, Katherine A., “The landscape of parallel computing research: A view from berkeley,” Tech. Rep. UCB/EECS-2006-183, Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, United States, December 2006. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html`.

[155] I. Corporation, “Fft-length-and-layout-advisor,” October 2017. `https://software.intel.com/content/www/us/en/develop/articles/fft-length-and-layout-advisor.html`.

[156] F. org, “Fftw reference,” November 2003. `http://www.fftw.org/fftw2_doc/fftw_3.html`.

[157] H. Khaleghzadeh, Z. Zhong, R. Reddy, and A. Lastovetsky, “Out-of-core implementation for accelerator kernels on heterogeneous clouds,” *The Journal of Supercomputing*, vol. 74, no. 2, pp. 551–568, 2018.

[158] Z. Zhong, V. Rychkov, and A. Lastovetsky, “Data partitioning on multicore and multi-GPU platforms using functional performance models,” *Computers, IEEE Transactions on*, vol. 64, no. 9, pp. 2506–2518, 2015.

[159] T. Smith and M. Waterman, “Identification of common molecular subsequences,” *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195 – 197, 1981.

[160] O. Gotoh, “An improved algorithm for matching biological sequences,” *Journal of Molecular Biology*, vol. 162, no. 3, pp. 705 – 708, 1982.

[161] T. Rognes, “Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation,” *BMC bioinformatics*, vol. 12, no. 1, p. 1, 2011.

[162] Y. Liu, A. Wirawan, and B. Schmidt, “CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions,” *BMC bioinformatics*, vol. 14, no. 1, p. 1, 2013.

[163] Y. Liu and B. Schmidt, “SWAPHI: Smith-Waterman protein database search on Xeon Phi coprocessors,” in *2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors*, pp. 184–185, IEEE, 2014.

[164] J. Rawlings, S. Pantula, and D. Dickey, *Applied regression analysis, a research tool*. Springer texts in statistics, New York, NY: Springer, 2. ed ed., 1998. isbn:0387984542.

[165] E. Ostertagová, "Modelling using polynomial regression," *Procedia Engineering*, vol. 48, pp. 500 – 506, 2012. Modelling of Mechanical and Mechatronics Systems and `http://www.sciencedirect.com/science/article/pii/S1877705812046085`.

[166] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment," *Journal of Grid Computing*, vol. 14, pp. 55–74, Mar 2016.

[167] T. Cao, Y. He, and M. Kondo, "Demand-aware power management for power-constrained hpc systems," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 21–31, May 2016.

[168] P. Arroba, J. M. Moya, J. L. Ayala, and R. Buyya, "Dynamic voltage and frequency scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, p. e4067, 2017.

[169] A. Shahid, M. Fahad, R. R. Manumachu, and A. Lastovetsky, "Improving the accuracy of energy predictive models for multicore cpus by combining utilization and performance events model variables," *Journal of Parallel and Distributed Computing*, vol. 151, pp. 38–51, 2021.

[170] T. Malik and A. Lastovetsky, "Towards optimal matrix partitioning for data parallel computing on a hybrid heterogeneous server," *IEEE Access*, vol. 9, pp. 17229–17244, 2021.

[171] P. Alonso, R. M. Badia, J. Labarta, M. Barreda, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí, and R. Reyes, "Tools for power-energy modelling and analysis of parallel scientific applications," in *2012 41st International Conference on Parallel Processing*, pp. 420–429, Sep. 2012.

[172] F. Mantovani and E. Calore, "Performance and power analysis of HPC workloads on heterogeneous multi-node clusters," *Journal of Low Power Electronics and Applications*, vol. 8, no. 2, p. 13, 2018.

[173] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[174] D. W. Mount, *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press, 2004. isbn:9780879696870.

[175] NVIDIA, "Parallel Thread Execution ISA Version 7.0," 2020. `http://docs.nvidia.com/cuda/parallel-thread-execution/#axzz4WCeB6m8U`.

Appendix A

Comparison of dynamic energy consumption using PMC-based energy predictive models and HCLWattsUp

A.1 Introduction

This chapter is based on mainly the research work done by my colleague Arsalan Shahid and as presented in [37]. It presents the prediction accuracy of linear energy predictive models employing PMCs as predictor variables with HCLWattsUp and Intel RAPL. In section 2.2.4, we present the popular tools to read the PMCs on a given platform such as Likwid [55], Linux Perf [103], PAPI [104] and Intel PCM [105]. Extrae and Paraver tools [171, 172] can also be used to gather the PMCs. These tools are built on top of PAPI.

A.1.1 Experimental Setup

The experimental setup is composed of two multicore CPU platforms (specifications given in Tables 4.1 and 4.2).Table A.1 presents the list of applications employed in our experimental suite. The application suite contains highly optimized memory bound and compute bound scientific routines such as DGEMM and FFT from Intel Math Kernel Library (MKL), benchmarks from NASA Application Suite (NAS), Intel HPCG, *stress*, *naive* matrix-matrix multiplication and *naive* matrix-vector multiplication. The reason to select a diverse set of applications is to avoid bias in our models and to have a range of PMCs for different executions of diverse applications.

Table A.1: List of Applications.

Application	Description
MKL FFT	Fast Fourier Transform
MKL DGEMM	Dense Matrix Multiplication
HPCG	High performance conjugate gradient
NPB IS	Integer Sort, Kernel for random memory access
NPB LU	Lower-Upper Gauss-Seidel solver
NPB EP	Embarrassingly Parallel, Kernel
NPB BT	Block Tri-diagonal solver
NPB MG	Multi-Grid on a sequence of meshes
NPB FT	Discrete 3D fast Fourier Transform
NPB DC	Data Cube
NPB UA	Unstructured Adaptive mesh, dynamic memory access
NPB CG	Conjugate Gradient
NPB SP	Scalar Penta-diagonal solver
NPB DT	Data traffic
stress	CPU, disk and I/O stress
Naive MM	Naive Matrix-matrix multiplication
Naive MV	Naive Matrix-vector multiplication

For a given application, we measure three quantities during its execution on our platforms. First is the dynamic energy consumption provided by HCLWattsUp API [38] using the methodology explain in Chapter 3. Second, we measure the execution time. Lastly, we collect all the PMCs available on our platforms using Likwid tool [55].

Likwid can be used using a simple command-line invocation as given below where the *EVENTS* represents PMCs (4 at maximum in one invocation) of the given application, *APP*:

likwid-perfctr -f -C S0:0-11@S1:12-23 -g EVENTS APP

The application (APP) during its execution is pinned to physical cores (0–11, 12–23) of our platform. Likwid use *likwid-pin* to bind the application to the cores on any platform and lack the facility to bind an application to memory. Therefore, we have used *numactl*, a command-line linux tool to pin our applications to available memory blocks.

For Intel Haswell and Intel Skylake platform, Likwid offers 164 PMCs and 385 PMCs, respectively. We eliminate PMCs with counts less than or equal to 10 since we found them to have no physical significance for modeling the dynamic energy consumption and they are non-reproducible over several runs of the same application on our platforms.

The reduced set contains 151 PMCs for Intel Haswell and 323 for Intel Skylake. As in a single application run we can collect only 4 PMCs, it takes a huge amount of time to collect all of them. Moreover, some PMCs can only be collected individually or in sets of two or three for an application run. Therefore, we observe that each application must be executed about 53 and 99 times on Intel Haswell and Intel Skylake platform, respectively, to collect all the PMCs.

We now divide our experiments in to following two classes, Class A and Class B, as follows:

1. Class A: In this class, we study the accuracy of platform-level linear regression models using a diverse set of applications.
2. Class B: In this class, we study the accuracy of application-specific linear regression models.

A.1.2 Accuracy of Platform-Level Linear PMC-Based Models

We select Intel Haswell multicore CPU platform (Table 4.1) for this class of experiments. We select the PMCs commonly used by these models and they are listed below:

- IDQ_MITE_UOPS (X_1)
- IDQ_MS_UOPS (X_2)
- ICACHE_64B_IFTAG_MISS (X_3)
- ARITH_DIVIDER_COUNT (X_4)
- L2_RQSTS_MISS (X_5)
- FP_ARITH_INST_RETIRED_DOUBLE (X_6)

These PMCs count floating-point and memory instructions and are considered to have a very high positive correlation with energy consumption. Table A.2 shows the correlation of the PMCs with the dynamic energy consumption.

Table A.2: Correlation of performance monitoring computers (PMCs) with dynamic energy consumption (E_D). Correlation matrix showing relationship of dynamic energy with PMCs. 100% correlation is denoted by 1.

	E_D	X_1	X_2	X_3	X_4	X_5	X_6
E_D	1	0.53	0.50	0.42	0.58	0.99	0.99
X_1	0.53	1	0.41	0.25	0.39	0.45	0.44
X_2	0.50	0.41	1	0.19	0.99	0.48	0.48
X_3	0.42	0.25	0.19	1	0.21	0.41	0.40
X_4	0.58	0.39	0.99	0.21	1	0.57	0.56
X_5	0.99	0.45	0.48	0.41	0.57	1	0.99
X_6	0.99	0.44	0.48	0.40	0.56	0.99	1

We used all the applications listed in Table A.1 with different configurations of problem sizes to build a data-set of 277 points. Each point represents the data for one application configuration containing its dynamic energy consumption and the PMC counts. We split this data-set in two subsets, one for training (with 227 points) the models and the other to test (50 points) the accuracy of models. We used this division based on best practices and experts' opinion in this domain.

Using the dataset, we build 6 linear models {A, B, C, D, E, F} using regression analysis. Model A employs all the selected PMCs as predictor variables. Model B is based on five best PMCs with the least energy correlated PMC (X_3) removed. Model C uses four PMCs with two least correlated PMCs (X_2, X_3) removed and so on until Model F, which contains just one the most correlated PMC (X_6)

The models are summarized in the Table A.3. We also show the minimum, average and maximum prediction errors of RAPL.

We will now focus on the minimum, average and maximum prediction errors of these models. They are (2.7%, 32%, 99.9%) respectively for Model A. Model B based five most correlated PMCs has prediction errors of (0.53%, 21.80%, 72.9%) respectively. The average prediction error significantly dropped from 32% to 21%. The prediction errors for Model C are (0.75%, 29.81%, 77.2%) respectively. The average prediction error in this case is in between that of Model A and Model C. Model F with just one most correlated PMC (X_6) has least average prediction error of 14%. The prediction errors of RAPL are (4.1%, 30.6%, 58.9%) From these results, we conclude that selecting PMCs using correlation with energy does not provide any consistent improvements in the accuracy of linear energy predictive models.

Table A.3: Linear predictive models (A-F) with intercepts and RAPL with their minimum, average and maximum prediction errors.

Model	PMCs	Intercept Followed by Coefficients	Percentage Prediction Errors (min, avg, max)
A	$X_1, X_2, X_3, X_4, X_5, X_6$	10, 3×10^{-9}, 1.9×10^{-8}, 3.3×10^{-7}, -1×10^{-6}, 6×10^{-8}, -9.3×10^{-11}	(2.7, 32, 99.9)
B	X_1, X_2, X_4, X_5, X_6	3×10^{-9}, 1.9×10^{-8}, -1×10^{-6}, 6.2×10^{-8}, -1.2×10^{-10}, 230	(0.53, 21.80, 72.9)
C	X_1, X_4, X_5, X_6	3.7×10^{-9}, 7.9×10^{-9}, 7.5×10^{-8}, -5.1×10^{-10}, 270	(0.75, 29.81, 77.2)
D	X_4, X_5, X_6	6.7×10^{-8}, 9.4×10^{-8}, -9.7×10^{-10}, 490	(0.21, 23.19, 80.42)
E	X_5, X_6	9.7×10^{-8}, -1.02×10^{-9}, 520	(2, 21.03, 83.40)
F	X_6	1.5×10^{-9}, 740	(2.5, 14.39, 34.64)
RAPL			(4.1, 30.6, 58.9)

We also identified a few more causes of inaccuracy in linear regression based models by looking at the coefficients of PMCs employed in them. Salient observations of these models are outlined below:

- All the models have a significant intercept (β_0). Therefore, the model would give predictions for dynamic energy based on the intercept values even for the case when there is no application executing on the platform, which is erroneous. We consider this to be a serious drawback of existing linear energy predictive models (given in section 2.2.4), which do not understand the physical significance of the parameters with dynamic energy consumption.
- Model A has negative coefficients ($\beta = \{\beta_1, ..., \beta_6\}$) for PMCs, X_4 and X_6. Similarly, Model B have negative coefficients for PMC X_4 and X_6, and Models C-E, X_6 has negative coefficient. The negative coefficients in these models can give rise to negative energy consumption pre-

dictions for specific applications where the counts for X_4 and X_6 are relatively higher than the other PMCs.

A.1.3 Accuracy of Application-Specific PMC-Based Models

In this section, we study the accuracy of application specific energy predictive models built using linear regression. We choose a single-socket Intel Skylake server (Table 4.2) for the experiments. We choose two highly optimized scientific kernels: Fast Fourier Transform (FFT) and Dense Matrix-Multiplication application (DGEMM), from Intel Math Kernel Library (MKL).

We select six PMCs (Y1-Y6) listed in the Table A.4, which have been employed as predictor variables in energy predictive models given in literature (section 2.2.4).

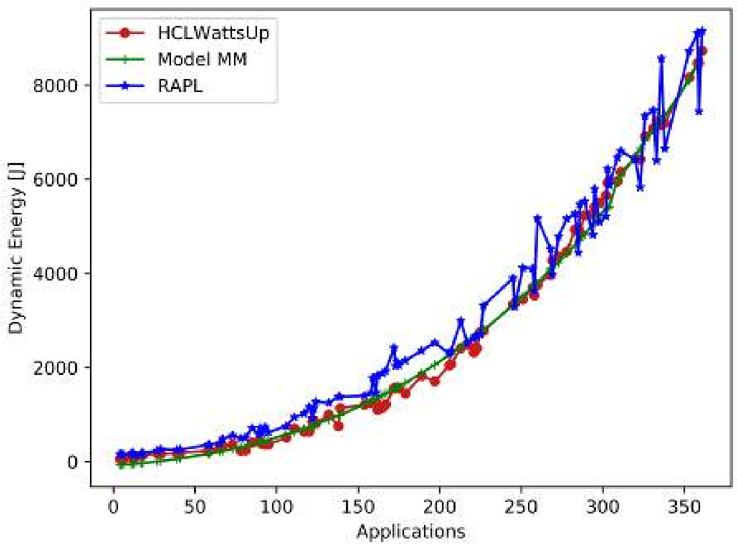
Table A.4: Selected PMCs for Class B experiments along with their energy correlation for DGEMM and FFT. 0 to 1 represents positive correlation of 0% to 100%.

	Selected PMCs	Corr DGEMM	Corr FFT
Y1	FP_ARITH_INST_RETIRED_DOUBLE	0.99	0.98
Y2	MEM_INST_RETIRED_ALL_STORES	0.99	0.99
Y3	MEM_INST_RETIRED_ALL_LOADS	0.98	0.55
Y4	MEM_LOAD_RETIRED_L3_MISS	0.60	0.99
Y5	MEM_LOAD_RETIRED_L1_HIT	0.98	0.34
Y6	ICACHE_64B_IFTAG_MISS	0.99	0.77

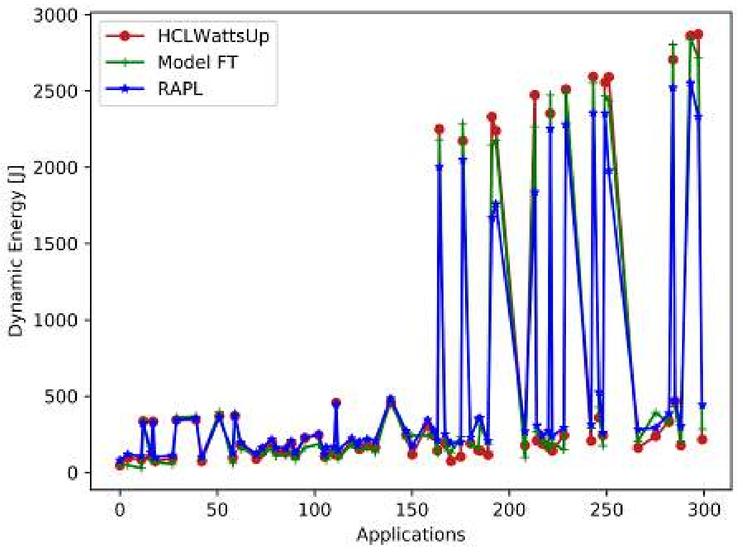
We build a dataset containing 362 and 330 points representing DGEMM and FFT for a range of problem sizes from 6400×6400 to 29,504 $\times$ 29,504 and 22,400 $\times$ 22,400 to 41,536 $\times$ 41,536, respectively, with a constant step sizes of 64. We split the dataset into training and test datasets. Training dataset for DGEMM and FFT contains 271 and 255 points used to train the energy predictive models. Test dataset contains 91 and 75 points for both applications respectively.

Using the datasets, we build two linear models for both applications. These are *Model MM* and *Model FT*. Figure A.1a,b shows the of dynamic energy profiles constructed with PMC base predictive models (*Model MM* and *Model FT*), RAPL and HCLWattsUp for DGEMM and FFT, respectively.

Comparing with HCLWattsUp, the minimum, average and maximum error for DGEMM using *Model MM* and RAPL are (0, 26, 218) and (0.4, 35, 161), respectively. In case of FFT, the minimum, average and maximum error using *Model FT* and RAPL is (0.8, 27, 147) and (0.3, 31, 155) respectively. We observe that both the models perform better in terms of average prediction accuracy than RAPL.



(a) Model MM



(b) Model FT

Figure A.1: Dynamic Energy profiles constructed with predictive models RAPL and HCLWattsUp.

Appendix B

Parallel Gene Sequencing Application

We use a gene sequencing application HCLSW executing the Smith-Waterman algorithm ([159, 160]) for our experiments. The application deals with alignment of DNA or protein sequences; a sequence is an ordering of DNA letters or amino acid letters. Sequence alignment or comparison refers to comparing two (or more) sequences by searching for a series of individual characters or patterns that are in the same order in the sequences. When sequences are aligned, matches, mismatches, spaces, and gaps are allowed. A gap is defined to be any maximal, consecutive run of spaces in a single string of a given alignment. A gap may be as small as a single space.

There are two types of alignment, global alignment and local alignment. A global alignment [173] of two strings S_1 and S_2 is obtained by first inserting chosen spaces, either onto or at the ends of S_1 and S_2, and then placing the two resulting strings one above the other so that every character or space in either string is opposite a unique character or a unique space in the other string. A local alignment of two strings S_1 and S_2 [173] is to find two substrings α and β of S_1 and S_2, respectively, whose similarity (optimal global alignment value) is maximum over all pairs of substrings from S_1 and S_2.

The SW algorithm uses a dynamic programming (DP) approach to determine the optimal local alignment score of two sequences. The recurrence relations of the algorithm [159] with modifications due to [160] using affine gap penalty functions are shown below ([161]):

$$H_{i,j} = \begin{cases} max \begin{cases} H_{i-1,j-1} + P[q_i, d_j] \\ E_{i,j} \\ F_{i,j} \\ 0 \end{cases} , & i > 0 \cap j > 0 \\ 0, & i = 0 \cup j = 0 \end{cases} \tag{B.1}$$

$$E_{i,j} = \begin{cases} max \begin{cases} H_{i,j-1} - Q \\ E_{i,j-1} - R \\ 0 \end{cases} , & j > 0 \\ 0, & j = 0 \end{cases} \tag{B.2}$$

$$F_{i,j} = \begin{cases} max \begin{cases} H_{i-1,j} - Q \\ E_{i-1,j} - R \ , \\ 0 \end{cases} & i > 0 \\ 0, & i = 0 \end{cases} \tag{B.3}$$

$$S = \max_{1 \leq i \leq m \cap 1 \leq j \leq n} H_{i,j} \tag{B.4}$$

The two sequences targeted for alignment are q, known as query sequence and d, known as database sequence. The query sequence q is of length m and contains residues q_i. The database sequence d is of length n and contains residues d_j. $H_{i,j}$ is the score for aligning the prefixes of q and d ending in the alignment of residues q_i and d_j. $E_{i,j}$ and $F_{i,j}$ are the scores of aligning the same prefixes of q and d but ending with a gap in the query and database sequence, respectively. $P[q_i, d_j]$ is the score of aligning residues q_i and d_j with each other according to a substitution score matrix P. Q is the sum of gap open and extension penalties while R is the gap extension gap penalty. S is the overall optimal local alignment score.

For alignment of protein sequences, two well-known families of substitution scoring matrices, PAM and BLOSUM, are used. Each value in a matrix represents an odds score, the likelihood that the two amino acids will be aligned in alignment of similar proteins divided by the likelihood that they will be aligned by chance in an alignment of unrelated proteins. The PAM matrices are based on a mutational model of evolution that assumes amino acid changes occur as a Markov process, where each amino acid change at a site is considered to be independent of previous changes at that site. The BLOSUM matrices are derived from considering all amino acid changes observed in an aligned region from a related family of proteins [174].

The time and space complexities of the SW DP algorithm are $O(m \times n)$ and $O(m)$, where $m < n$, assuming the use of refined linear-space methods. The performance of the SW DP algorithm is usually measured in GCUPS, which stands for Billions of Cell Updated per Second (a cell here refers to a cell in the DP table of dimensions $m \times n$).

For the parallel implementation of the application on our platform, we use the following packages:

- SWIPE for Multicore CPUs [161]. This package contains highly optimized implementation of SW algorithm using SIMD parallelization (for example: using the SSE3 intrinsic offered by latest Intel processors).

- CUDASW++3.0 for nVidia GPU accelerators [162]. This package contains highly optimized implementations of SW algorithm using SIMT (Single Instruction, Multiple Thread) and virtualized SIMD (Single Instruction, Multiple Data) abstractions using CUDA PTX SIMD video instructions [175] for nVidia Tesla GPUs.

- SWAPHI for Intel Xeon Phi accelerators [163]. This package contains highly optimized implementations of SW algorithm using tiled parallelization approach where instruction-level parallelism using SIMD vectorization (512-bit SIMD instructions) and thread-level parallelism (using OpenMP) are employed.

Appendix C

Calibration of WattsUp Pro power-meter

HCLServer01 (Table 4.1) has an Intel Haswell multicore CPU having 24 physical cores with 64 GB main memory and integrated with two accelerators: one Nvidia K40c GPU and one Xeon Phi 3120P. HCLServer02 (Table 4.2) has an Intel Skylake multicore CPU consisting of 22 cores and 96 GB main memory and integrated with one Nvidia P100 GPU. These nodes are representative of computers used in cloud infrastructures, supercomputers and heterogeneous computing clusters.

Each node has a *Watts Up Pro* power meter installed between its input power sockets and the wall A/C outlets. *Watts Up Pro* power meters are periodically calibrated using the ANSI C12.20 revenue-grade power meter, Yokogawa WT310. The maximum sampling speed of *Watts Up Pro* power meters is one sample every second. The accuracy specified in the data-sheets is $\pm 3\%$. The minimum measurable power is 0.5 watts. The accuracy at 0.5 watts is ± 0.3 watts. The accuracy of Yokogawa WT310 is $\pm 0.1\%$ and the sampling rate is 100K samples per second.

The dynamic energy consumption during the application execution is measured using the WattsUp Pro power meter on each node using HCLWattsUp interface [38]. In this chapter, we explain our methodology and some results to calibrate our power-meters.

The WattsUp Pro power-meter power measurements are compared with Yokogawa under following three settings:

1. Naked-eye visual monitoring
2. Monitoring server base (idle) power
3. Measurement of total energy consumption by scientific applications

The rationale to these settings is to compare the power readings of both meters in all typical scenarios.

C.1 Naked-eye visual monitoring

The procedure is presented as follows:

- First, one WattsUp Pro power-meters is plugged in between the input power sockets and the wall A/C outlets with each server.

- Once the servers are turned on and are in stable condition, the power readings in Watts as displayed on LCDs of the meters are monitored and noted for each server.
- The WattsUp Pro power meters are carefully plugged off.
- The Yokogawa power meter is plugged in between the each server is plugged in between the input power sockets and the wall A/C outlets with each server.
- Once the servers are turned on and are in stable condition, the power readings in Watts as displayed on LCDs of the meters are monitored and noted for each server.
- We find a difference of 2 and 3 watts for HCLServer01 and HCLServer02 respectively.

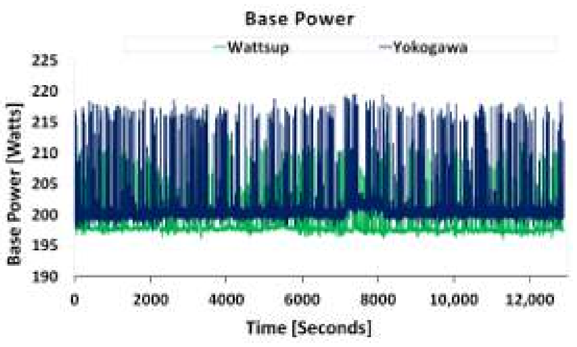
C.2 Monitoring server base power

The procedure is the same for both WattsUp Pro power meters and Yokogawa WT310. In general:

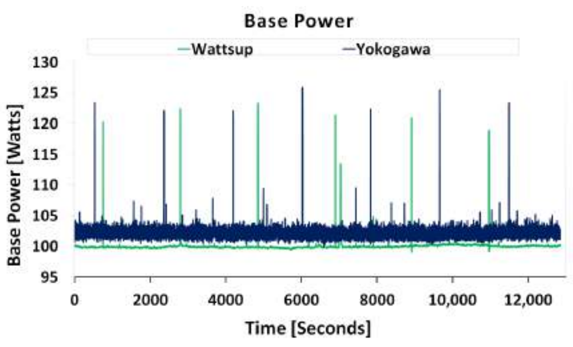
- Both servers are connected with power meters and Yokogawa WT310.
- Once the servers are stable, the power consumption by each server is obtained progrmatically from each power meters.
- For WattsUp Pro, a Perl script is used to obtain the power readings within the frequency of 1 second. Similarly, a python API is used to obtain the power readings within the frequency of 1 second from Yokogawa.
- We compare the power readings obtained from with naked-eye visual monitoring with software. We find no significant overhead of software.
- The base power of each server is monitored for more than 3 hours on different times. Figure C.1 illustrate the base power profiles of HCLServer01 and HCLserver02 using both power meters. One can observe that the base power profiles constructed with both power meters on each server exhibit the same pattern. However, HCLServer01 exhibits more power variations than HCLServer2. For HCLServer2, there are power spikes after every half an hour for a couple of seconds. However, this difference is due to the difference in OS and software configuration on both servers.
- Table C.1 presents the minimum, average, and maximum power consumption by each servers as reported by WattsUp Pro and Yokogawa power meters. In a nut shell, there is a difference of 1 or 2 watts between them.

C.3 Measurement of total energy consumption by scientific applications

For this study, we choose two scientific applications: 1) DGEMM and 2) FFT from Intel MKL. On HCLServer01, we execute DGEMM for problem sizes ranging from 4096$\times$4096 to 30720$\times$30720



(a) HCLServer01



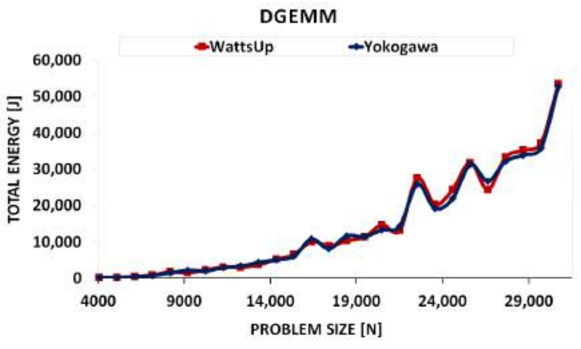
(b) HCLServer02

Figure C.1: Base power of HCLServers with HCLWattsUp and Yokogawa.

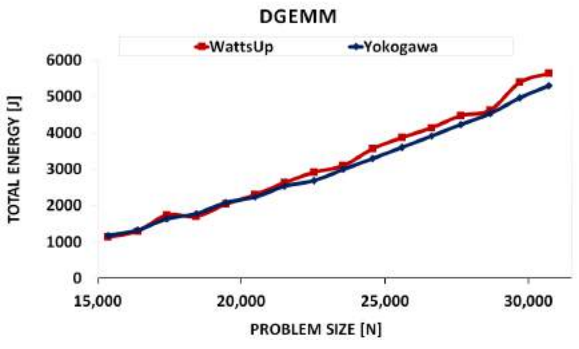
Table C.1: Minimum, maximum and average of idle power using WattsUp Pro and Yokogawa power meters on HCLServer01 and HCLServer02

	HCLServer01		HCLServer02	
	WattsUp Pro	Yokogawa	WattsUp Pro	Yokogawa
Min	196	197.72	99.1	99.93
Average	198.2	201.0	100.03	102.06
Max	212.8	219.47	123.3	125.6

with a constant step size of 1024. On HCLServer02, DGEMM is executed for problem sizes ranging from 15360×15360 to 30720×30720 with a constant step size of 1024. We build the total energy consumption profiles of DGEMM on each server using both power meters. For our next batch of experiments on HCLServer1, we execute FFT for problem sizes ranging from 4096×4096 to 30720×30720 with a constant step size of 1024. On HCLServer02, FFT is executed for problem sizes ranging from 8384×8384 to 62880×62880 with a constant step size of 2096. We build the total energy consumption profiles of DGEMM on each server using both power meters. Figures C.3 and C.2 illustrate the total energy consumption profiles for FFT and DGEMM on both servers, respectively. Table C.2 presents the average error in percentage for total energy consumption by DGEMM and FFT on HCLServer1 and HCLServer2 as reported by each power meter. The average error for DGEMM is 4.95% and 9.25% on HCLServer2 and HCLServer1, respectively. For FFT, the average measurement error is 6% and 7.4% on HCLServer2 and HCLServer1, respectively.

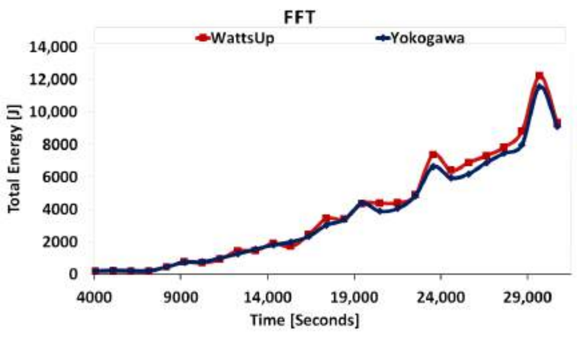


(a) HCLServer01

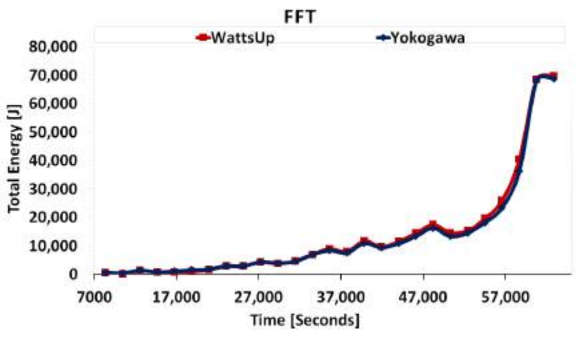


(b) HCLServer02

Figure C.2: Total energy consumption by DGEMM with HCLWattsUp and Yokogawa on HCLServers.



(a) HCLServer01



(b) HCLServer02

Figure C.3: Total energy consumption by FFT with HCLWattsUp and Yokogawa on HCLServers.

Table C.2: Comparison of minimum, average, and maximum measurement errors for DGEMM and FFT on HCLServer01 and HCLServer02 using WattsUp Pro and Yokogawa

	HCLServer01 Errors [%] (Min, Avg, Max)	HCLServer02 Errors [%] (Min, Avg, Max)
DGEMM	(0.58, 9.25, 33.18)	(2.01, 4.95, 8.16)
FFT	(0.20, 7.41, 16.90)	(0.11, 6.02, 15.84)

Appendix D

Similarity Results of Group A

Group A comprises of the EPS where there are more than one energy profile of the same application constructed with different approaches such as on-chip power sensors, system-level power measurements provided by power meters, etc.

Table D.1: Similarity results for Group A.

Correlation	Average Error[%]	Euclidean Distance Between Profiles	Similarity Class	TSM Rank
DGEMM_DiffLoad				
0.9992	1.2646	1975.9796	same	**1**
0.9668	63.5344	90472.0176	opposite	-
0.9682	64.9294	92104.4449	opposite	-
DGEMM_EqualLoad				
0.9995	4.5639	13623.2651	similar	3
0.9993	21.2295	8270.0584	similar	2
0.9993	16.1469	7625.6795	similar	**1**
FFT_Different Load				
0.9933	3.7506	483.9803	same	**1**
0.8983	75.6650	8394.0909	similar	3
0.9002	65.9681	7318.8504	similar	2
FFT(socket1)_DGEMM(socket2)				
0.9960	1.4562	441.0981	same	**1**
0.8596	29.9055	8014.0644	similar	2
0.8900	34.5172	9025.7489	similar	3
FFT(HCLServer01)-Sensors				
0.9785	4.3116	3806.0259	similar	**1**
0.9105	15.0796	12702.5932	similar	2
DGEMM(HCLServer01)- Sensors				
0.9383	3.0720	3803.1372	similar	**1**
0.9037	19.0641	19732.2589	opposite	-
DGEMM_AnMoHA				

0.9762	2.0190	2257.6144	same	**1**
0.8641	7.7355	8794.7977	similar	3
0.5741	6.5949	8420.5404	opposite	-
0.6741	5.9963	7522.9645	opposite	-
0.8945	4.0381	4515.2345	similar	2
FFT_Predictive Models				
0.9887	37.2600	2414.0882	similar	5
0.9924	91.8419	3321.1547	similar	6
0.9994	7.8324	472.2268	similar	2
0.9991	11.8382	593.4430	similar	4
0.9998	3.8569	293.9054	similar	**1**
0.9997	5.1557	334.3252	similar	3
DGEMM_Predictive Models				
0.9993	27.3336	6217.9387	similar	5
0.9973	39.0893	10576.7382	similar	6
0.9999	8.5995	2076.8247	similar	**1**
0.9985	13.1182	6574.3586	similar	3
0.9999	3.0024	1276.4288	similar	2
0.9986	6.2412	6556.0057	similar	4

Appendix E

Similarity Results of Group B

An EPS is a constituent of Application, Configuration Parameters, Profile, Platform , Problem Size and Step Size.

Table E.1: Similarity Results of Group B. Here Problem Size is $(M \times N)$ where $0 \geq M \leq N$.

Application, Configuration Parameter, Profile, Platform	**Problem Size, Step Size(SS)**	**Correlation**	**Average_ Error[%]**	**Euclidean Distance Between Profiles**	**TSM Similarity**
DGEMM, Problem Size, RAPL, HCLServer01	M=12800-20480, N=20480, SS=256	0.9319	10.2457	5161.1469	opposite
FFT, Problem Size, RAPL, HCLServer01	M=15104-18688, N=23552, SS=64	0.8315	9.2421	92.3809	similar
DGEMM, Problem Size, Sensors, HCLServer01	M=10752-21504, N=21504, SS=256	0.7945	53.7876	3841.1001	opposite
FFT, Problem Size, Sensors, HCLServer01	M=15104-18688, N=23552, SS=64	0.7779	11.2001	152.0388	opposite
DGEMM, Problem Size, Sensors, HCLServer02	M=18176-22528, N=22528 SS=128	0.5959	13.1062	1745.9799	opposite
FFT, Problem Size, Sensors, HCLServer02	M=21504-25600, N=25600, SS=128	0.6419	73.3393	258.5867	opposite
DGEMM, Problem Size, Sensors, HCLServer01	M=12800-20480, N=20480, SS=256	0.8791	37.0786	14275.7493	similar
FFT, Problem Size, Sensors, HCLServer01	M=15104-18688, N=23552, SS=64	0.8534	40.8715	2715.9380	similar
DGEMM, Problem Size, RAPL, HCLServer01	M=512-16384, N=16384, SS=512	0.9694	62.4245	4014.0942	similar
FFT, Problem Size, RAPL, HCLServer01	M=16256-22528, N=22528, SS=128	0.9964	16.0109	92.5880	similar
DGEMM, Problem Size, RAPL, HCLServer02	N=6400-29504, SS=64	0.9857	36.1267	10085.1871	similar
FFT, Problem Size, RAPL, HCLServer02	N=22400-41536, SS=64	0.9928	28.6694	2557.1955	opposite
IntelMKLFFT, CPU Cores, RAPL, HCLServer03	N=43328	0.9976	13.0454	6699.9534	similar
FFTW, CPU Threads, RAPL, HCLServer03	N=32768	0.9999	7.4316	5485.4512	similar
FFTW, Problem Size, RAPL, HCLServer03	N=32768	0.9836	10.4683	1134.8875	similar

MKL, Problem Size, RAPL, HCLServer03	N=25600-46080, SS=512,G=1,T=56	0.9998	13.5649	1489.3704	opposite
	N=25600-46080, SS=512,G=2,T=28	0.9999	12.3451	1375.7246	similar
	N=25600-46080, SS=512,G=4,T=14	0.9999	11.9745	1398.6977	similar
	N=25600-46080, SS=512,G=7,T=8	0.9999	12.3395	1603.8175	similar
	N=25600-46080, SS=512,G=8,T=7	0.9999	12.4057	1695.1576	similar
	N=25600-46080, SS=512,G=14,T=4	0.9998	12.9987	2153.5891	similar
	N=25600-46080, SS=512,G=28,T=2	0.9997	14.4593	3179.7384	similar
FFTW, Problem Size, RAPL, HCLServer03	M=512 - 10240, N=20480, SS=512	0.9977	6.5117	937.3472	similar
	M=544-10272, N=20544, SS=512	0.9674	11.1643	360.9246	similar
	M=576-10304, N=20608, SS=512	0.8846	7.0593	114.4169	same
	M=308-10336, N=20672, SS=512	0.4054	8.0483	218.7999	similar
	M=128-10368, N=20736, SS=512	0.5962	5.9544	83.1244	similar
	M=108-10400, N=20800, SS=512	0.7604	6.7705	100.8612	similar
	M=192-10432, N=20864, SS=512	0.8973	10.5817	326.0428	similar
	M=224 10464, N=20992, SS=512	0.6522	9.9545	285.7523	similar
	M=256-10496, N=20992, SS=512	0.5333	9.1475	274.5072	similar
	M=288-10528, N=21056, SS=512	0.9991	5.9861	819.9989	similar
	M=320-10560, N=21120, SS=512	0.1895	5.6268	81.5864	similar
	M=352-10592, N=21184, SS=512	0.9967	6.3244	928.4300	similar
	M=384-10624, N=21248, SS=512	0.9480	12.5488	471.9435	similar
	M=416-10656, N=21312, SS=512	0.9964	6.4423	900.8481	similar
	M=448-10688, N=21376, SS=512	0.9461	10.0455	333.6082	similar
	M=480-10720, N=21440, SS=512	0.9487	10.2235	281.0485	similar
	M=512-10752, N=2154, SS=512	0.8474	6.1786	97.7390	similar

OpenBlas, Problem Size, RAPL, HCLServer03	N=10240-26112, SS=512,G=2,T=56	0.9999	23.3422	15628.3773	similar
	N=10240-26112, SS=512, G=4,T=28	0.9998	20.8883	31537.2291	similar
	N=10240-26112, SS=512,G=7,T=16	0.9998	22.7078	48717.0221	similar
	N=10240-26112, SS=512, G=8,T=14	0.9998	21.5933	58193.8600	similar
	N=10240-26112, SS=512,G=14,T=8	0.9997	15.8470	95839.8982	similar
	N=10240-26112, SS=512,G16,T=7	0.9997	15.8663	143722.2497	similar
	N=10240-26112, SS=512,G=28,T=4	0.9998	15.4408	258623.5761	similar
	N=10240-26112, SS=512,G=56,T=2	0.9994	16.1594	31431.2584	similar
MKL, Problem Size, RAPL, HCLServer03	N=32768-43456, SS=64,G=1,T=56	0.9999	14.1218	7423.8074	same
	N=32768-43456, SS=64, G=2,T=28	0.9998	13.0212	7452.8136	similar
	N=32768-43456, SS=64, G=4,T=14	0.9998	13.0325	7923.3728	similar
	N=32768-43456, SS=64,G=7,T=8	0.9999	13.2150	8457.6352	same
	N=32768-43456, SS=64,G=8,T=7	0.9999	13.2373	8563.3480	same
	N=32768-43456, SS=64,G=14,T=4	1.0000	13.6266	9325.5557	similar
	N=32768-43456, SS=64,G=28,T=2	0.9999	15.0864	11717.0005	same
FFTW, Problem Size, RAPL, HCLServer03	N=35480-41920, SS=64,G=1,T=112	0.9978	24.6166	78669.0124	same
	N=35480-41920, SS=64,g=2,T=56	0.9995	12.2964	5993.3535	same
	N=35480-41920, SS=64,G=4,T=28	0.9976	13.7285	6227.9184	similar
	N=35480-41920, SS=64,G=7,T=16	0.9966	14.5904	5915.2791	similar
	N=35480-41920, SS=64,G=8,T=14	0.9970	13.6615	5569.3958	similar
	N=35480-41920, SS=64, G=14,T=8	0.9946	13.1908	5102.1465	similar
	N=35480-41920, SS=64,G=16,T=7	0.9947	12.3994	4850.3314	similar

FFTW, Problem Size, RAPL, HCLServer03	N=30720-34816, SS=64,G=1,T=112	0.9986	25.0459	81473.9811	similar
	N=30720-34816, SS=64,G=2,T=56	0.9984	10.9418	3060.2949	similar
	N=30720-34816, SS=64,G=4,T=28	0.9840	9.3161	3639.7016	similar
	N=30720-34816, SS=64,G=7,T=16	0.9945	14.8833	3013.0125	similar
	N=30720-34816, SS=64,G=8,T=14	0.9942	16.1527	2955.8465	similar
	N=30720-34816, SS=64,G=14,T=8	0.9912	16.3173	2595.7632	similar
	N=30720-34816, SS=64,G=16,T=7	0.9917	42.8061	2416.1999	similar
FFTW, Problem Size, RAPL, HCLServer03	N=20480-26560, SS=64,G=1,T=112	0.9996	25.6272	86911.0622	similar
	N=20480-26560, SS=64,G=2,T=56	0.9994	9.7569	1502.1569	similar
	N=20480-26560, SS=64,G=4,T=28	0.9985	7.5984	1324.9766	similar
	N=20480-26560, SS=64,G=7,T=16	0.9549	17.7605	1611.0385	opposite
	N=20480-26560, SS=64,G=8,T=14	0.9427	21.5876	1806.2769	similar
	N=20480-26560, SS=64,G=14,T=8	0.6497	28.1641	2012.2371	similar
	N=20480-26560, SS=64,G=16,T=7	0.6010	30.9980	2163.2482	opposite
AnMoHA [2.5% precision]					
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01	M=12800-20224, N=20224,SS=128	0.9750	2.2414	3398.2869	same
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01	M=12800-20480, N=20480, SS=256	0.9383	3.0720	3803.1504	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01	M=12800-20736, N=20224, SS=256	0.9739	3.8751	4349.8794	same
FFT, Problem Size, HCLWattsUp_Combined, HCLServer01	M=15104-18688, N=23552,SS=64	0.9785	4.3116	3806.0146	same
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer02	M=16384-20096, N=22528, SS=128	0.9504	2.1708	956.8094	similar
FFT, Problem Size, HCLWattsUp_Combined, HCLServer02	M=21504-25600, N=25600, SS=64	0.9387	4.8698	2252.6855	similar

AnMoHA Less Accurate (10% prercision)					
DGEMM, Problem Size, HCLWattsUp_Parallel, HCLServer01	M=12800-20224, N=20224,SS=128	0.9134	6.8688	9937.8680	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01		0.9216	7.7965	10996.5086	similar
DGEMM, Problem Size, HCLWattsUp_Parallel, HCLServer01	M=12800-20480, N=20480,SS=256	0.9094	5.9996	6914.4728	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01		0.9023	8.9318	9266.7659	similar
DGEMM, Problem Size, HCLWattsUp_Parallel, HCLServer01	M=12800-20736, N=20736	0.9110	8.1647	8258.8659	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01		0.9456	7.5815	7766.5003	similar
FFT, Problem Size, HCLWattsUp_Parallel, HCLServer01	M=15104-18688, N=23552,SS=64	0.7930	4.4804	2080.0339	similar
FFT, Problem Size, HCLWattsUp_Combined, HCLServer01		0.8625	3.0740	1516.4442	similar
DGEMM, Problem Size, HCLWattsUp_Parallel, HCLServer01	M=16384-20096, N=22528,SS=128	0.9308	8.4808	7966.4267	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01		0.9010	8.1606	7241.1619	similar
FFT, Problem Size, HCLWattsUp_Parallel, HCLServer01	M=21504-25600, N=25600,SS=64	0.8667	13.4232	7170.1068	similar
FFT, Problem Size, HCLWattsUp_Combined, HCLServer01		0.9010	8.1606	7241.1619	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01	M=64- 20288, N=10112,SS=64	0.9963	8.0723	7060.6467	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer02		0.9994	5.0213	1184.0506	similar
DGEMM, Problem Size, HCLWattsUp_Combined, HCLServer01, HCLServer02		0.9972	7.6614	7598.0246	similar

FFT, Problem Size, HCLWattsUp_Combined, HCLServer01	M=1024-10160, N=51200, SS=16	0.9991	14.5338	6279.0562	similar
FFT, Problem Size, HCLWattsUp_Combined, HCLServer02		0.9971	5.7143	147.5044	similar
FFT, Problem Size, HCLWattsUp_Combined, HCLServer01, HCLServer02		0.9992	14.0973	6383.0253	similar

Acronyms

ACPI Advanced Configuration and Power Interface.

AnMoHA Additive Energy Modelling of Hybrid Parallel Applications.

API Application Programming Interface.

CMOS Complementary Metal-oxide-semiconductor.

CPU Central Processing Unit.

DGEMM Double-precision General Matrix Multiplication.

DPM Dynamic Power Management.

DVFS Dynamic Voltage and Frequency Scaling.

EPS Set of Energy Profiles of an Applicaiton Constructed with Different Energy Measurement Techniques.

FFT Fast Fourier Transform.

FFTW Fastest Fourier Transform in the West.

FLOPS Floating Point Operations Per Second.

FPGA Field Programmable Gate Array.

GPU Graphics Processing Unit.

HPC High Performance Computing.

ICT Information and Communication Technologies.

IEA International Energy Agency.

LLC Last Level Cache.

MKL Intel Math Kernel Library.

MKL-FFT Intel MKL-Fast Fourier Transform.

MPI	Messgae Passing Interface.
MPSS	Manycore Platform Software Stack.
NPU	Neural Processing Unit.
NUMA	Non-Uniform Memory Access.
NVML	NVIDIA Management Library.
PMC	Performance Monitoring Counter.
PMI	Power Monitoring Infrastructures.
QPI	Quick Path Interconnect.
RAPL	Running Average Power Limit.
SMC	System Management Controller chip.
SW	Smith-Waterman algorithm.
TPU	Tensor Processing Unit.
TSM	Trend-based Similarity Measuring Methodology for Energy Profiles.
VPU	Vision Processing Unit.
VR	Voltage Regulator.
Xeon Phi	Intel Xeon Phi.