IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2020.DOI

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

MUHAMMAD FAHAD[1], ARSALAN SHAHID[1], RAVI REDDY MANUMACHU[2], ALEXEY LASTOVETSKY[2]

[1]School of Computer Science, University College Dublin (e-mail: {muhammad.fahad, arsalan.shahid}@ucdconnect.ie)
[2]School of Computer Science, University College Dublin (e-mail: {ravi.manumachu, alexey.lastovetsky}@ucd.ie)

Corresponding author: Muhammad Fahad (e-mail: muhammad.fahad@ucdconnect.ie).

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

ABSTRACT Modern high-performance computing platforms, cloud computing systems, and data centers are highly heterogeneous containing nodes where a multicore CPU is tightly integrated with accelerators. An important challenge for energy optimization of hybrid parallel applications on such platforms is how to accurately estimate the energy consumption of application components running on different compute devices of the platform. In this work, we propose a method for accurate estimation of the application component-level energy consumption employing system-level power measurements with power meters. We experimentally validate the method 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%. We also show 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).

INDEX TERMS energy modelling, energy optimization, power meters, on-chip power sensors, heterogeneous platforms, parallel applications, multicore CPU, GPU, Intel Xeon Phi, HPC

I. INTRODUCTION

High energy consumption by Information and Communications Technology (ICT) systems and devices is a serious concern now because of its dire consequences on the economy and environment. ICT systems and devices are already consuming about 2000 terawatt-hours (TWh) per year which is about 10% of the global electricity demand [1]. With a share of 200 TWh, data centers are a big contributor to this high electricity consumption. Andrae and Edler [2] predict that ICT could use up to 51% of global electricity in 2030, and it could contribute up to 23% of globally released greenhouse gas emission. Because of such high power consumption, future HPC systems are highly likely to be power constrained. For example, the Department of Energy (DOE) in the United States aims to deploy an exascale supercomputer capable of performing 1 million trillion (10^{18}) floating-point operations per second within a power envelope of 20-30 megawatts [3].

Several system-level and application-level solution methods have been proposed to reduce the energy consumption. System-level solution methods include clock and power gating, dynamic voltage and frequency scaling (DVFS), dynamic power management (DPM), etc [4]–[6]. In contrast, the application-level approach is comparatively understudied. Application-level solution methods use application-level models and application-level parameters such as number of processors, number of threads, workload distribution, etc., as decision variables for energy optimization of applications [7]–[9]. In this work, we will focus exclusively on the

application-level approach. Accurate measurement of energy consumption during an application execution is pivotal to application-level solution methods.

Modern high-performance computing (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 work.

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. We present the details of the issues involved in addressing the challenge in section II. 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 the section III. In brief, the two state-of-the-art approaches, on-chip sensors and energy predictive models, suffer from poor accuracy and high implementation complexity [10]. 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 **A**dditive e**n**ergy **Mo**delling of **H**ybrid **A**pplications (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 consumptions 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.

We experimentally validate the method on a cluster of two hybrid heterogeneous computing nodes using three parallel applications – matrix-matrix multiplication, 2D fast Fourier transform and a gene sequencing application for a diverse range of problem sizes. The experiments demonstrate a high estimation accuracy of the proposed method, with the 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%. The main contributions of this work are:

- A novel methodology (AnMoHA) to determine the fine-grained decomposition of the energy consumption by a hybrid application executing in parallel on multiple independent computing devices (CPU, GPU, Xeon Phi, FPGA, etc) in a hybrid heterogeneous computing platform within sufficient accuracy, and empirical validation of the methodology on two modern heterogeneous hybrid servers.
- A comprehensive study to analyze the pragmatic aspects of additive energy modeling: i). The trade-off between the time to build the energy functions and their accuracy, and ii). The trade-off between the number of independent experiments to build the energy functions and their accuracy.
- A comprehensive study to find the hardware topological granularity limits of AnMoHA. We demonstrate how 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 find that if an application is executed on some of the CPU cores of a socket, then the base energy (or the idle energy) of idle 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.
- A comparison of the accuracy of additive energy models using state-of-the-art on-chip power sensors against the additive energy models constructed with the ground truth for two scientific hybrid applications (matrix-matrix multiplication and 2D fast Fourier transform) on a modern hybrid heterogeneous computing platform containing an Intel multicore CPU, a Nvidia GPU, and an Intel Xeon Phi.
- We show that significant energy is lost by employing the energy models constructed by using energy measurements provided by on-chip sensors for dynamic energy optimization of an application.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

The rest of the paper is organized as follows. We present the challenges to energy optimization of a hybrid parallel application in section II. The comparative study of prediction accuracy of the state-of-the art approaches to measure the energy consumption is presented in the section III. Related work follows in section IV. Our solution method, AnMoHA, is explained in section V. The experimental validation of AnMoHA is presented in section VI. We discuss the trade-offs between the accuracy and time-space, and between the accuracy and design-space of AnMoHA in sections VII and VIII. We explore the hardware topological granularity and the scalability of AnMoHA in section IX. 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 X. Finally, we conclude the paper in section XI.

II. CHALLENGES TO ENERGY OPTIMIZATION OF A HYBRID PARALLEL APPLICATION

Modern high-performance computing (HPC) platforms, cloud computing systems, and data centers are commonly composed of nodes where a multicore CPU is tightly integrated with one or more 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. A hybrid parallel application consists of application components running in parallel on multiple compute devices of such a heterogeneous hybrid node.

To optimize the application for dynamic energy consumption, a naïve approach explores all possible workload distributions to obtain the optimal workload distribution for the compute devices such that the energy consumption by the application is minimized. For each workload distribution, it determines the total dynamic energy consumption during the parallel execution of the workload. It, then, returns the workload distribution with the optimal total dynamic energy consumption. Briefly, the total energy consumption is the sum of dynamic and static energy consumption. The static energy consumption is the idle power of the platform (without application execution) multiplied by the execution time of the application. The dynamic energy consumption is the total energy consumed by the platform during the application execution minus the static energy consumption. In this work, we consider only dynamic energy. We explain the rationale behind using dynamic energy consumption instead of total energy consumption in appendix F.

The naïve approach, however, has an exponential complexity. To reduce this complexity, we need application component energy profiles that are input to a data partitioning algorithm to determine the workload distribution minimizing the dynamic energy consumption during the parallel execution of the workload. Consider, for example, the model-based data partitioning algorithm [11] to compute an optimal distribution of a given workload size N amongst p heterogeneous processors that minimizes the total dynamic energy during the parallel execution of the application. The algorithm takes as input p dynamic energy profiles of the application components 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 [11].

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. Fahad et al. [10] show that the use of inaccurate energy measurements to optimize the energy consumption by an application can lead to significant energy losses by up to 84%.

We illustrate the energy loss by an example. Consider, for example, a dynamic energy profile of an image processing application for a range of problem sizes$\{x_1, x_2, \cdots, x_n\}$. Let the tool A reports the dynamic energy consumption for problem sizes $\{x_i, x_{i+1}, x_{i+2}\}$ (where $0 < i$ and $i + 2 \leq n$) to be {776 J, 764 J, 634 J}. Whereas, the tool B (the ground truth) reports the dynamic energy consumption by the same problem sizes to be {409 J, 540 J, 568 J}. Now, if the measurements by tool A are used for dynamic energy optimization of the image processing application for workload size (or image size) $N = x_i$, an optimization method for dynamic energy using the profile built with the tool A as an input could use the solution for the workload size $N = x_{i+2}$, aiming to reduce the dynamic energy consumption by 18%. Solving this workload size, however, will result in an increase of dynamic energy consumption by 39% according to the ground truth. Hence, 39% energy loss occurs by using inaccurate measurements to optimize the energy consumption of the application.

Furthermore, the tight integration of multicore CPUs with accelerators in modern hybrid heterogeneous HPC systems has resulted in 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, 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). As a result, the workload of one application component of a hybrid application may significantly impact the performance and energy consumed by the others when running in parallel on multiple compute devices. One can accurately measure the performance of the application components running in parallel using high precision processor clocks. However, there is no such effective equivalent for measuring the energy consumption.

III. COMPARATIVE ANALYSIS OF THE STATE-OF-THE-ART ENERGY MEASUREMENT TOOLS

In this section, first, we survey the state-of-the-art energy measurement approaches. Then, we compare and analyze

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

their accuracy, implementation cost and the cost to determine the energy consumption by an application.

We categorize the dominant approaches for determining the energy consumption by an application as follows: a). System-level physical measurements using external power meters, b). Measurements using on-chip power sensors, and c). Energy predictive models. The first approach using the external power meters is considered to be accurate [12]. It can provide only the measurement at a computer level and cannot, therefore, provide the fine-grained decomposition of the energy consumption by an application executing on multiple independent computing devices in a computer.

The second approach employs on-chip power sensors which are now available in mainstream processors such as Intel and AMD Multicore CPUs, Nvidia GPUs, and Intel Xeon Phis. There are vendor-specific libraries to acquire the power data from these sensors. For example, Running Average Power Limit (RAPL) [13] is used to monitor power and control frequency (and voltage) of Intel CPUs. Similarly, Nvidia NVIDIA Management Library (NVML) [14] and Intel System Management Controller chip (SMC) [15] provide the power consumption by Nvidia GPUs and Intel Xeon Phi. NVML provides the instant power draw values with nominal accuracy up to $\pm 5\%$ [14]. The accuracy of Intel SMC is not available. Apart from insufficient documentation, there are other issues with the power data values provided by these vendor-specific libraries. For example, it lacks details such as update frequency of power readings and also suffers from potential complications such as sampling interval variability of significant sensor lag as reported by [16].

The third approach based on software energy predictive models emerged as a popular alternative to determine the energy consumption of an application. Majority of the models are linear and employ performance monitoring counters (PMCs) as predictor variables. While the models allow determination of fine-grained decomposition of energy consumption during the execution of an application, there are research works highlighting their poor accuracy [17]–[20].

Fahad et al. [10] present the first comprehensive comparative study on 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 is considered to be the ground truth. We illustrate the key results of the study [10], comparing the accuracy of energy measurements using on-chip power sensors and energy predictive models against system-level physical power measurements using external power meters. We run our experiments on two modern hybrid heterogeneous computing platforms: HCLServer01 and HCLServer02. HCLServer01 contains an Intel Haswell CPU, an Nvidia K40 GPU and an Intel Xeon Phi 3120P. HCLServer02 contains an Intel Skylake multicore CPU and an Nvidia P100 GPU. The technical details are provided in table 1. These nodes are representative of computers used in cloud infrastructures, supercomputers, data centers, 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.

TABLE 1: Technical Specifications of HCLServers.

Technical Specifications	**HCLServer01**	**HCLServer02**
Processor	Intel E5-2670 v3 @2.30GHz	Intel Xeon Gold 6152
Micro-architecture	Haswell	Skylake
OS	CentOS 7	CentOS 7
Thread(s) per core	2	2
Cores per socket	12	22
Socket(s)	2	1
NUMA node(s)	2	1
L1d cache / L11 cache	32 KB / 32 KB	32 KB / 32 KB
L2 cache	256 KB	1024 KB
L3 cache	30720 KB	30976 KB
Main memory	64 GB DDR4	96 GB DDR4
TDP	240 W	140 W
Idle Power	58 W	32 W
	NVIDIA K40c	**NVIDIA P100 PCIe**
No. of processor cores	3584	2880
Total board memory	12 GB CoWoS HBM2	12 GB GDDR5
Memory bandwidth	549 GB/sec	288 GB/sec
	Intel Xeon Phi 3120P	
No. of processor cores	57	
Total main memory	6 GB GDDR5	
Memory bandwidth	240 GB/sec	

We employ two popular scientific applications for our experiments: (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. Matrix-matrix multiplication and fast Fourier transform routines are fundamental kernels employed in scientific applications [21]. 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 [22] for Nvidia GPUs and XeonPhiOOC [22] for Intel Xeon Phis, are used. The ZZGEMMOOC and XeonPhiOOC packages reuse CUBLAS and MKL BLAS for in-card DGEMM calls. We follow the detailed methodology explained in appendix I to compare the energy measurements using RAPL, and GPU/Xeon Phi sensors against system-level physical measurements provided by external power meters.

A hybrid parallel application (such as DGEMM and 2D-FFT) executing on a heterogeneous platform, consists of several kernels (generally speaking, multithreaded), running in parallel on different computing devices (CPU, GPU, Intel Xeon Phi, etc.) of the platform. To minimize the resource contention, we consider only such configurations in this work where there is one-to-one mapping between the given (CPU or accelerator) kernel and the corresponding device i.e. each kernel is running only on its corresponding device

and there is no more than one (CPU or accelerator) is running on its corresponding device. To determine the dynamic energy consumption by each application kernel, we model the computing elements executing an application kernel as abstract processors [23] so that the executing platform is represented as a set of heterogeneous abstract processors. Each abstract processor solely constitutes the processing elements and resources which are involved in the execution of the application kernel mapped to it. We explain the details on abstract processors in section V.

We build dynamic energy profiles of DGEMM on Nvidia Tesla P100 GPU on HCLServer02 with sensors (RAPL and NVML) for the workload sizes ranging from 18176×22528 to 22528×22528 with a constant step size of 128. Figure 1 illustrates the dynamic energy profiles. One can observe that the combined dynamic energy profile with RAPL and NVML sensors exhibits a linear profile, however, the ground truth exhibits a different pattern. We find that combined sensor measurements do not follow the trend of dynamic energy consumption exhibited by ground truth for 64.71% of the data points. The maximum and the average error of the sensor profiles are 84.84% and 40.06% respectively.

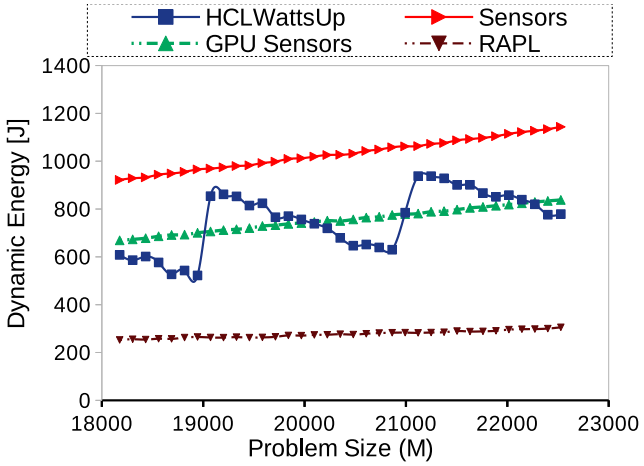


FIGURE 1: Dynamic Energy profiles of DGEMM on Nvidia Tesla P100 GPU.

For brevity reasons, we present the results for our experiments to compare the accuracy of RAPL against the power meters on Intel Skylake Gold 6152 (HCLServer02), and the sensors (RAPL and MPSS) on Intel Xeon Phi 3120P (HCLServer01) in appendix J. In summary, RAPL does not exhibit similar dynamic energy consumption behavior as that of the ground truth for the dynamic energy consumption by MKL-FFT executed on Intel Skylake Gold 6152. We find the maximum and average error of RAPL to be 156% and 29% respectively. Owing to the interlacing nature of both profiles composed by both tools, we can not calibrate RAPL to reduce its average error against the ground truth. For MKL-DGEMM executed on Intel Xeon Phi 3120P, we find that sensors MPSS report on the average higher dynamic energy consumption than the ground truth. The average and maximum error of the sensor (MPSS and RAPL) profile against the profile using power measurements provided by external power meters are 64.5% and 93.06% respectively.

We finish our overview with a comparison of the prediction accuracy of linear energy predictive models employing performance monitoring counters (PMCs) as predictor variables with the ground truth. 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 methodology in appendix L.

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 IV). 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%.

A. IMPLEMENTATION COMPLEXITY AND COST

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

[24] provides 164 PMCs and 385 PMCs for the Intel Haswell and the Intel Skylake multicore processors (Table 1) 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.

The cost in terms of number of measurements to determine a single data point of an application dynamic energy profile with on-chip power sensors is higher than the cost for the ground truth and for the energy predictive models based on PMCs. To determine the dynamic energy consumption by a given workload size using on-chip sensors, we need 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.

However, we need just following three measurements for the ground truth: i) base power, ii) total energy, and iii) execution time. Whereas, just one measurement is sufficient for the PMC-based energy predictive models.

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 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.

In summary, energy measurement methods employing on-chip sensors and energy predictive models suffer from poor accuracy, implementation complexity, and topological granular limitations. Therefore, they are not suitable for optimization of applications for dynamic energy.

IV. RELATED WORK

Additive energy models for the entire system: One of the simplest additive models was presented by Roy et al. [33] which represents an algorithm energy consumption as a weighted sum of the energy consumption by CPU and memory. Lewis et al. [25] proposed a system-wide energy model (based on hardware performance counters) as a summation of power models of processor, memory (DRAM), fans, motherboard (chipset) peripherals, and hard-disk drive. Basmadjian et al. [26] presented a similar model including more 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. Bircher et al. [27] 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. Fatemeh et al. [34] proposes a priority based energy-efficient routing method to optimize the energy consumption of Internet of things (IoT). Venkatraman et al. [35] present Reinforcement Learning (RL) based framework for energy-efficient Mobile Device Clouds (MDC). The framework learns the power-related statistics of the system and generates a group of resources for computation.

Energy predictive models for CPU: References [28], [17], 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. [36] quantify the relationship between the workload and power consumption of the multicore processor by using the statistics of CPU utilization. Lastovetsky et al. [8] 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.

Energy predictive models for accelerators: Burtsher et. al [16] examined the power profiling of three different Nvidia GPUs (Tesla K20c, K20m, and K20x) when computing a given workload (n-body simulation benchmark) using integrated sensors. Hong et al. [29] present an energy model for an Nvidia GPU based on a similar PMC-based power prediction approach of [37]. Nagasaka et al. [30] propose a PMC-based statistical power consumption modelling technique for GPUs that run CUDA applications. Song et al. [31] present power and energy prediction models based on machine learning algorithms such as back propagation in artificial neural networks (ANNs). Shao et al. [32] develop an instruction-level energy consumption model for a Xeon Phi processor.

Critiques of built-in power sensors and PMC based

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

TABLE 2: Summary of notable related-work. Here, '-' indicates the value is not reported. '*' indicates the model is portable to next-generation processors in the same architecture space.

Model	Decomposition	Is Energy/Power Consumption Predicted By The Application Executing Parallel On Compute Devices?	Type of Power (Instantaneous or Average)	Is Energy Predicted?	Accuracy of Energy Prediction	Is Platform Independent?
Lewis et al [25]	Cor0 to Core 3, DRAM, HDD, Fan, Support, Chipsets	No	Instantaneous	Yes	4%	No*
Basmadjian et al. [26]	CPU, RAM, NIC, HDD, Fan	No	Instantaneous	No	-	No*
Bircher et al. [27]	Chipset, CPU, Disk, I/O, Memory, Memory Controller	No	Instantaneous, Average	No	-	No
Heath et al. [28]	CPU, disk, network	No	Instantaneous	No	-	No
Economou et al. [17]	CPU, memory, disk, network	No	Instantaneous	No	-	No
Hong et al. [29]	GPU (ALU, Constant cache, FDS (Fetch/Dec/Sch), Floating Point Unit, Global memory, Int. arithmetic unit, Local memory, Register File, SFU, Shared memory, Texture cache)	No	Average	Yes	-	No
Nagasaka et al. [30]	GPU	No	Average	No	-	No*
Song et al. [31]	GPU (Floating Point Unit, Global Memory, Shared Memory, Local Memory, Texture Cache)	No	Average	Yes	11.02%	No*
Shao et al. [32]	Intel Xeon Phi (Compute, Hardware PF, MEM, Private Cache, Redundant SW-PF, Remote Cache, Software PF)	No	Average	Yes	5%	No
AnMoHA	Independently Powered Compute Devices (CPU, GPU, Intel Xeon Phi, CPU Socket, FPGA etc.)	Yes	Average	Yes	5%	Yes*

predictive modelling: Many researchers have highlighted the poor prediction accuracy and limitations of built-in on-chip power sensors and PMC based models. McCullough et al. [18] report the prediction errors of linear regression-based models as high as 150% in a study on the accuracy of predictive power models for multicore architectures. O'Brien et al. [19] report the average error of linear PMC based energy models used in their experimental study, as high as 60%. Arsalan et al. [20] also question the reliability and reported prediction accuracy of these 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 physical law of conservation of energy.

Burtsher et. al [16] 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.

Research works [38] and [39] study the RAPL accuracy on Haswell generation processors by running different benchmarks. While the former run micro-benchmarks using different thread configurations, the latter run benchmarks using different frequency configurations. Both of them compare the RAPL readings with total system (AC) power consumption using power meters and find the RAPL readings in a strong correlation with AC measurements. Our work differs from that in Reference [38] and [39] in several ways: (a) 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. (b) 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. (c) The authors run their benchmarks on the Haswell platform only whereas our experiment testbed is more diverse and includes advance generations of Intel CPU micro-architecture. (d) They 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. Studies [10] shows, however, that the energy measurements cannot be calibrated with both

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

tools because of their qualitative differences and interlacing behavior. More details on this can be found in appendix J and [10]s.

In [10], the authors report that energy measurements using on-chip sensors (of CPU, GPU, and Xeon Phi) do not capture the holistic picture of the dynamic energy consumption during application execution and therefore should not be used to measure the energy consumption of an application. They demonstrate that inaccurate energy measurements with on-chip sensors for dynamic energy optimization can result in a significant loss of energy. They report that average error of dynamic energy measurements with on-chip power sensors can 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.

In table 2, we compare different power/energy models and AnMoHA based on their characteristics. Here Decomposition indicates the level of component level power/energy breakdown.

V. 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 a additive modelling approach.

A. 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.

To achieve this, 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 devices 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 [23]. This way, the executing hybrid heterogeneous computing platform is represented as a set of heterogeneous abstract processors. We make sure that the sharing of system resources is maximized within the groups representing the abstract processors and minimized between them. To minimize the contention and mutual dependence between abstract processors, following two properties must be satisfied to formulate the abstract processors:

- Completeness: An abstract processor contains solely the computing elements which execute an application kernel. There is a one-to-one mapping between an abstract processor and its corresponding application kernel.
- Loose coupling: Abstract processors do not interfere with each other during the application execution. 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 AP_a when executing the application A is not affected by the activities of abstract processor AP_b when running application B in parallel.

Both properties are essential for composing an abstract processor.

We illustrate this concept using an example. Consider HCLServer01 (technical specifications are provided in table 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 Intel Xeon Phi 3120P coprocessor.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

Now, consider a hybrid parallel application DGEMM which computes the matrix multiplication of two dense matrices. Let the application uses Intel MKL DGEMM for CPU and Intel Xeon Phi, and CUBLAS for Nvidia GPU. We formulate three abstract processors $\{CPU1, GPU1, PHI1\}$ that satisfy the aforementioned two properties.

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 Intel 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.

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

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

where E_{AP_i}(x) is the dynamic energy consumption by the workload size x executing on abstract processor AP_i. Table 3 describes the notation employed in this section.

B. 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 Intel MKL DGEMM for CPU and Intel 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 3: Table of notations used in equations 1 and 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 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:

$$E_{ABC}(x) = E_A(x) + E_B(x) + E_C(x) \tag{2}$$

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

(discrete functions) then are input to a data 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.

VI. EXPERIMENTAL VALIDATION OF ANMOHA

In this section, we experimentally validate AnMoHA.

A. 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 (SW) algorithm ([40], [41]). The platforms are HCLServer01 and HCLServer02 (specifications given in Table 1). The details of DGEMM, 2D-FFT, and the platforms are explained in section III.

We choose matrix-matrix multiplication and fast Fourier transform routines because they are fundamental kernels employed in scientific applications [21]. 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 [42], CUDASW++3.0 for Nvidia GPU accelerators [43], and SWAPHI for Intel Xeon Phi accelerators [44]. We present the detailed description of the application in appendix M.

To ensure the reliability of our experimental results, we use an automated tool HCLWattsUp Interface [45] to determine the dynamic energy consumption by an application kernel. The interface and the methodology used to obtain a data point are explained in the appendix G. HCLWattsUp has no extra overhead and therefore does not impact the dynamic energy consumption by the application kernel. It follows a detailed statistical methodology to ensure the reliability of experimental results which is explained in appendix H. 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 and a user-defined precision has been achieved. We set the confidence interval as 95% and the precision as 0.025 (2.5%) for our experiments. The software returns the sample mean and standard deviation of the dynamic energy consumption.

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 appendix K.

B. FORMULATION OF ABSTRACT PROCESSORS ON HCLSERVERS

We group the compute devices of both the platforms into five abstract processors by following the properties of completeness and loose coupling.

- Abstract processors, CPU1, GPU1 and PHI1 on HCLServer01, as explained in section V-A.
- 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. 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 total parallel execution time of the hybrid application is equal

to the maximum of the execution times of the kernels run sequentially.

C. 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 threads complete their execution on other abstract processors. 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 2 can be extended as

$$E_{ABC}(x) = E_A(x) + E_B(x) + E_C(x) + \overline{e} \quad \text{where } \overline{e} \geq 0 \tag{3}$$

where $\overline{e}$ denotes the dynamic energy consumed by the application kernel to keep its context alive. The description of notations used in equation 3 is provided in table 5.

We run three different workload configurations (M $\times$ N where M $\leq$ N) of DGEMM executing on HCLServer01 and for each configuration build four dynamic energy functions of DGEMM as explained in VI-B. The workload sizes range for all three configurations are i) from 12800 $\times$ 20224 to 20224 $\times$ 20224 with a constant step size of 128, 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 2a illustrate the parallel and combined dynamic energy profiles for N=20224. We show the results for the other two experiment configurations in appendix B. In summary, we find the average and maximum error between combined and parallel dynamic energy profiles i) for N=20224 to be 2.24% and 5.56%, ii) for N=20480, 3.07% and 7.6%, and iii) for N=20736 to be 3.87% and 9.97% respectively. Furthermore, we find that the execution time of a hybrid application is the maximum of all the execution times of its constituent kernels.

We then compute the 2D-FFT for the problem size ranging from 15104 $\times$ 23552 to 18560 $\times$ 23552 with a constant step size of 64. Figure 3a shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer01. The average and maximum error between parallel and combined dynamic energy profiles are 4.32% and 8.91%. For our next batch of experiments on HCLServer01, we run the SW application for the problem size ranging from 16384 $\times$ 16384 to 18240 $\times$ 16384 with a constant step size of 64. Figure 4a 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 2.14% and 5.4%.

On HCLServer02, we run DGEMM with workload size ranges from 16384 $\times$ 22528 to 20096 $\times$ 22528 with a constant step size of 128. Figure 2b shows the dynamic energy profiles of parallel and combined on HCLServer02. We find the average and maximum error between combined and parallel dynamic energy profiles to be 2.32% and 6.6%. We then compute the 2D-FFT on HCLServer02 for the problem size ranging from 21504 $\times$ 25600 to 25600 $\times$ 25600 with a constant step size of 64. Figure 3b shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer02. The average and maximum error 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 size ranging from 40000 $\times$ 16384 to 42624 $\times$ 16384 with a constant step size of 64. Figure 4b 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 1.22% and 3.6%. Table 4 represents the percentage error between parallel and combined dynamic energy profiles of DGEMM, 2D-FFT and SW on HCLServers.

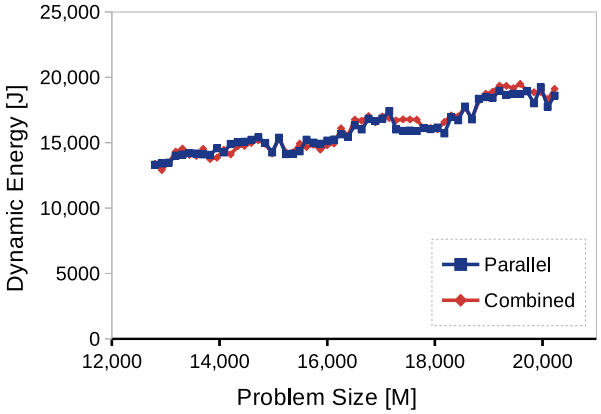
TABLE 4: Percentage error between parallel and combined dynamic energy profiles on HCLServers.

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

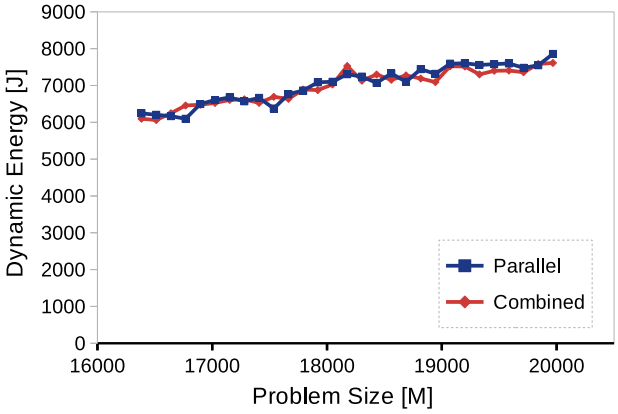
VII. TRADE-OFF BETWEEN ACCURACY AND TIME SPACE OF ADDITIVE MODELLING

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

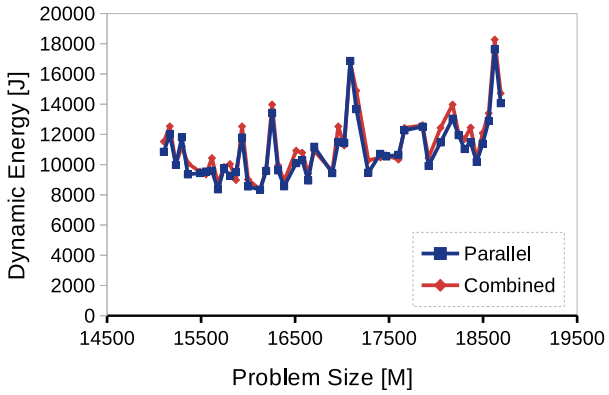


(a) N=20224, HCLServer01

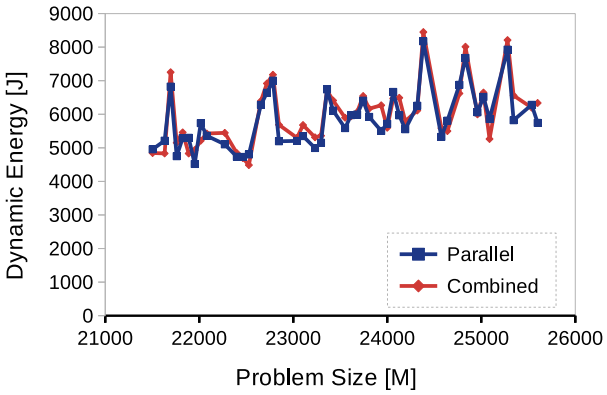


(b) N=22528, HCLServer02.

FIGURE 2: Dynamic energy profiles of DGEMM on HCLServers.

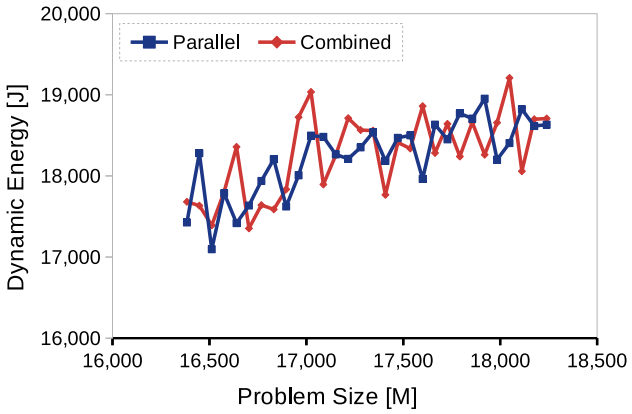


(a) N=23552, HCLServer01

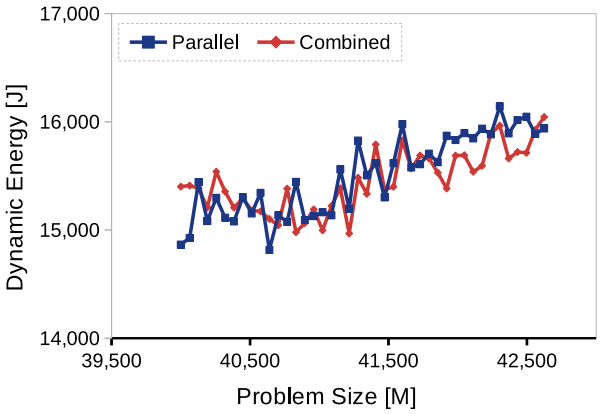


(b) N=25600, HCLServer02

FIGURE 3: Dynamic energy profiles of 2D-FFT on HCLServers.



(a) N=16384, HCLServer01



(b) N=16384, HCLServer02

FIGURE 4: Dynamic energy profiles of Smith-Waterman application on HCLServers.

TABLE 5: Table of notations used in equations 3 and 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.

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:

$$T = \sum_{i=1}^{p} \left(\sum_{j=1}^{m} n_{ij} \times (t_{ij} + s) \right) \tag{4}$$

The description of notations used in equation 4 is provided in table 5. 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 [8], [11]) 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 a 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.

A. RESULTS AND DISCUSSION

For this study, we repeat the same experiments to build energy profiles as explained in section VI-C, 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 VI-B to compare the accuracy of AnMoHA with relaxed precision settings and to study the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

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 5a, 5b and 5c illustrate the parallel and combined dynamic energy profiles for all three workload configurations with both precision settings. The average and maximum error between 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 the 2D-FFT on HCLServer01 for the problem size ranging from 15104 $\times$ 23552 to 18560 $\times$ 23552 with a constant step size of 64. Figure 6a shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer01. The average and maximum error between 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 the SW application on HCLServer02 for the problem size ranging from 16384 $\times$ 16384 to 18240 $\times$ 16384 with a constant step size of 64. Figure 7a shows the dynamic energy parallel and combined profiles of SW executing on HCLServer01. The average and maximum error between parallel and combined dynamic energy profiles are 7.78% and 12%.

On HCLServer02, we run DGEMM with workload size ranges from 16384 $\times$ 22528 to 20096 $\times$ 22528 with a constant step size of 128. Figure 5d 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 size ranging from 21504 $\times$ 25600 to 25600 $\times$ 25600 with a constant step size of 64. Figure 6b shows the dynamic energy parallel and combined profiles of 2D-FFT executing on HCLServer02 under both aforementioned precision settings. The average and maximum error between parallel and combined dynamic energy profiles are 14.56% and 54.18%.

For our final batch of experiments for this study, we run the SW application on HCLServer02 for the problem size ranging from 40000 $\times$ 16384 to 42624 $\times$ 16384 with a constant step size of 64. Figure 7b 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 6 shows the percentage errors of combined profiles with parallel ones on both platforms.

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 7 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

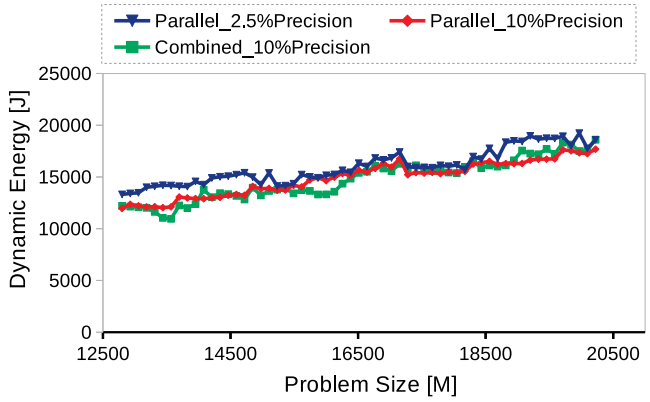
TABLE 6: Percentage errors between parallel and combined dynamic energy profiles with 10% precision setting.

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%

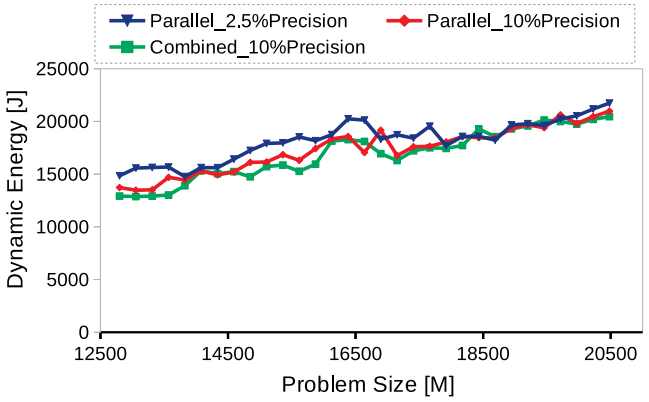
TABLE 7: Percentage absolute mean and maximum deviations of dynamic energy consumption by accurate parallel (with 2.5% precision setting) and less accurate combined profiles with 10% precision settings on HCLServers. Here 's01' denotes HCLServer01 and 's02' denotes HCLServer02.

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

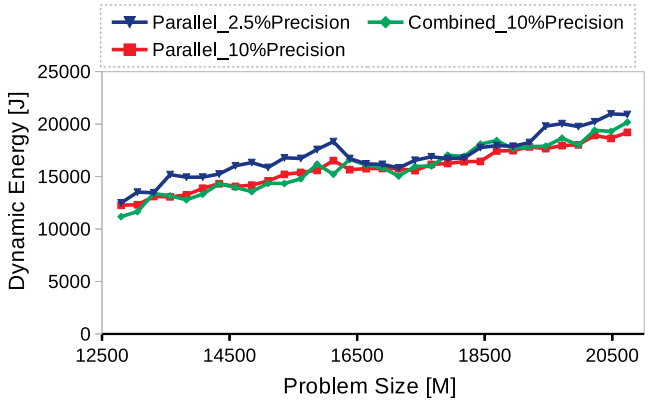
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



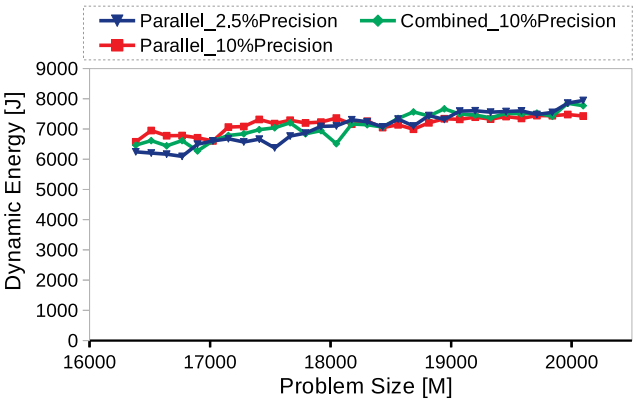
(a) N=20224, HCLServer01



(b) N=20480, HCLServer01

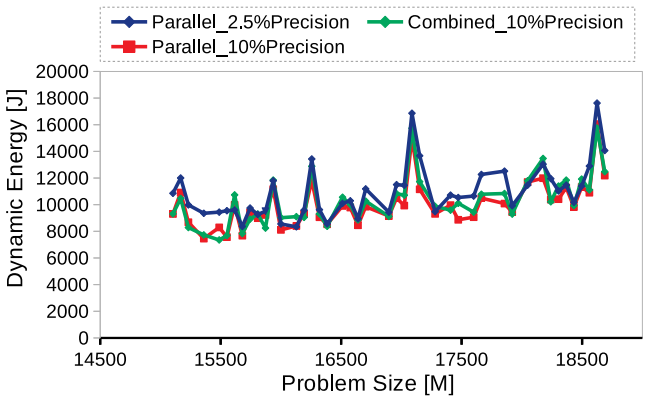


(c) N=20736, HCLServer01

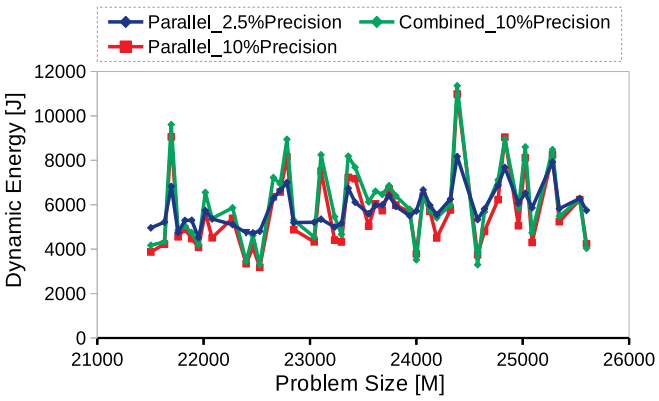


(d) N=22528, HCLServer02

FIGURE 5: Dynamic energy consumption by DGEMM on HCLServers.



(a) N=23552, HCLServer01



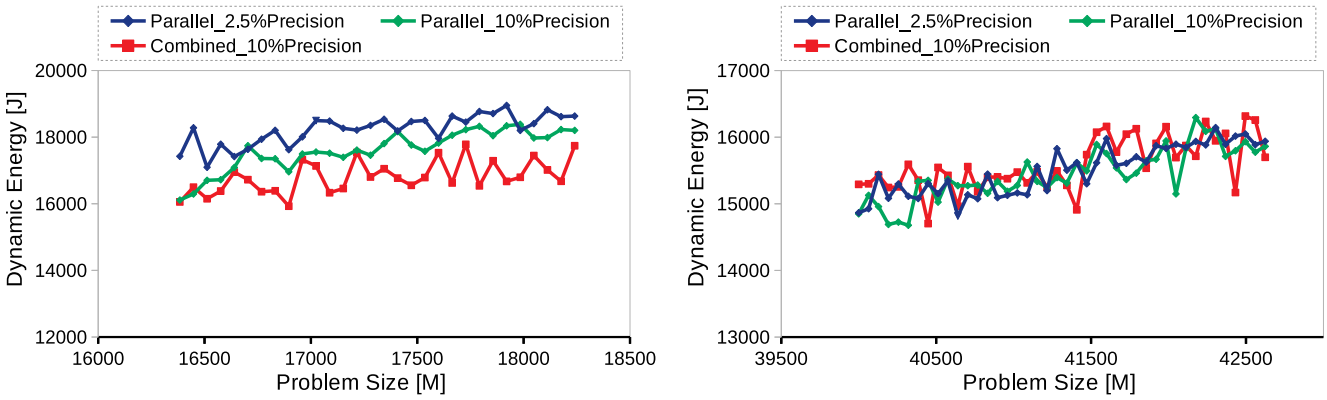
(b) N=25600, HCLServer02

FIGURE 6: Dynamic energy consumption by FFT on HCLServers.

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 [11].

As explained in equation 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 8, we compare the accuracy against the time to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



(a) N=16384, HCLServer01

(b) N=16384, HCLServer02

FIGURE 7: Dynamic energy consumption by Smith-Waterman application on HCLServers.

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.

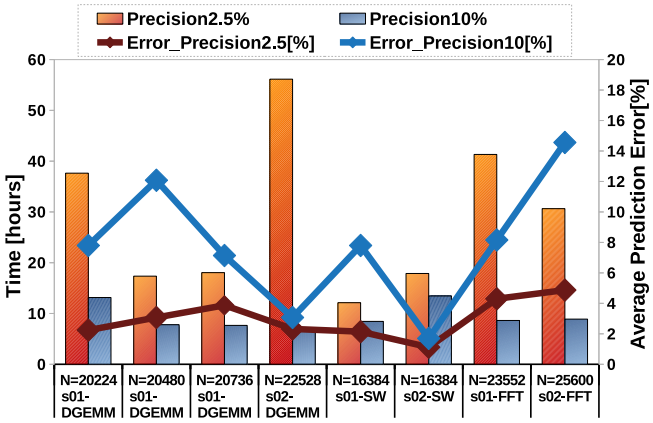


FIGURE 8: Trade-off between the time to build energy models using AnMoHA and their accuracy.

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 where the combined energy profiles has an average error of 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 4. We use a detailed methodology to ensure the reliability of our results as explained in appendix H. 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 VI-C. Hence, it takes at least 560 seconds to reliably determine a single data point within user-defined accuracy, and more than six 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 six 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 VI-C) . It can take more time if the application repetitions are more than five to

get the convergence. However, it would have taken far less time 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 does not define a percentage limit to indicate the passing threshold.

VIII. 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 the 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 VI-B and V-B. 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:

$$E_A(x)_1 = E_A(x) \tag{5}$$

$$E_A(x)_2 = E_{AB}(x) - E_B(x) \tag{6}$$

$$E_A(x)_3 = E_{AC}(x) - E_C(x) \tag{7}$$

$$E_A(x)_4 = E_{ABC}(x) - E_{BC}(x) \tag{8}$$

$$E_A(x)_5 = E_{ABC}(x) - E_B(x) - E_C(x) \tag{9}$$

Likewise, the dynamic energy consumption by workload x executed on the abstract processors B or C can be determined using the independent experiments as explained in appendix C. We can compose five combined profiles using these experiment configurations as {Combined_1, Combined_2, Combined_3, Combined_4, Combined_5} such as $\text{Combined}_1 = E_{A1} + E_{B1} + E_{C1}$.

For this study, we build the dynamic energy profiles of DGEMM on HCLServer01 for the workload size 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 VI-A. 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.

TABLE 8: 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%

A. RESULTS AND DISCUSSION

The detailed results are presented in appendix C. Table 8 provides the percentage errors for each combined dynamic energy profile (composed by using different experimental design configurations) with the parallel dynamic energy profile. The 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.

We explain the percentage deviations from the mean and maximum of dynamic energy consumption by parallel and each combined dynamic energy profiles in appendix C. In summary, the absolute percentage error between average and maximum deviations of each aforementioned combined profiles and parallel profile are {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 9 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.

To summarize, we find that experiment configuration (in equation 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) 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

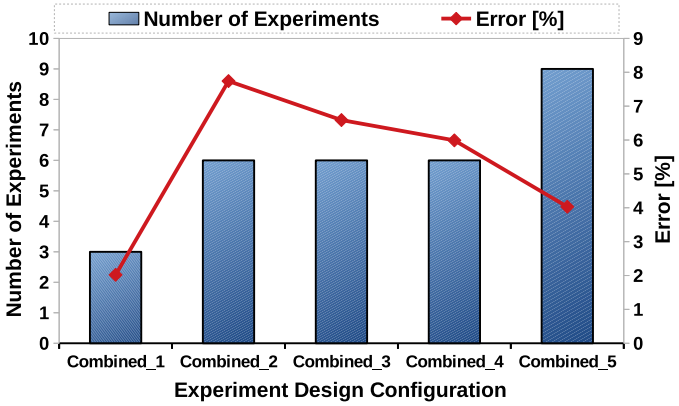


FIGURE 9: Trade off between number of experiments and accuracy.

We illustrate this by an example. Consider the experiment configuration explained in equation 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 each 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) 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, but it requires the most number of experiments to compose the combined dynamic energy profile. However, the Combined_5 dynamic energy profile 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. But, 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.

IX. STUDY OF ADDITIVE ENERGY MODELLING OF DIFFERENT WORKLOAD TYPES, GRANULARITY LIMITATIONS AND SCALABILITY OF ANMOHA

In this section, we explore the limits of the topological granularity of a computing platform on the viability and efficacy of the AnMoHA when different workload types/applications and sizes are executing on their corresponding different independently powered compute devices. 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. 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 Intel MKL routines of the application kernels of DGEMM and FFT as explained in section VI-A. 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.

A. 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. 10 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

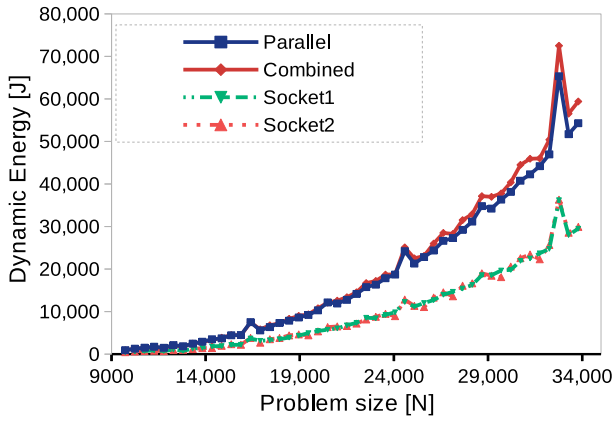


FIGURE 10: Case study A: Socket-wide dynamic energy profiles of same application and same workloads.

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 brevity reasons, we present the details of the experiment results for second and third use case scenarios in appendix D. To summarize, overall, we find 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 9 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 VI). This is because, each CPU socket of Intel Haswell E5-2670V3 is independently powered, and there is minimal resource sharing when running parallel two different applications pinned on individual sockets. Hence, the abstract processor AbsCPU satisfies both the propositions of additive hypothesis: completeness and loose-coupling (as explained in section V-A), and that is why we observe that additive hypothesis holds for socket-wide dynamic energy consumption.

B. STATE-OF-THE ART ENERGY MEASUREMENT TOOLS

We compare the measurements of dynamic energy consumption by RAPL against HCLWattsUp. RAPL [13] 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 RAPL, we use a well-known package, Intel PCM [46]. We ensure that the RAPL values output by this package is correct by comparing with values given by another well-known package, PAPI

TABLE 9: Percentage errors between socket-wide parallel and combined dynamic energy profiles built with HCLWattsUp.

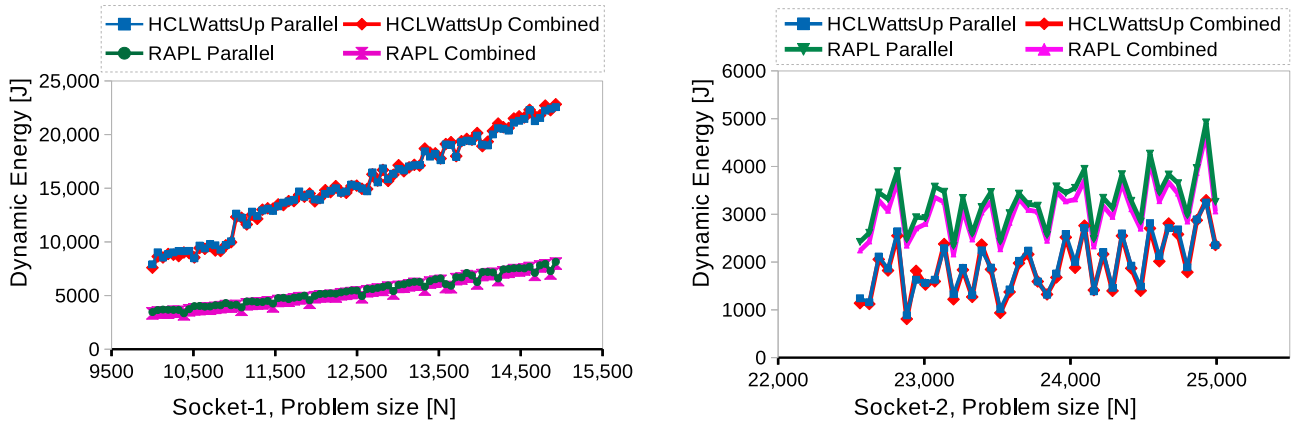
Experiment Configuration	Min	Avg	Max
Same application (DGEMM), Same workload	0.09%	4.56%	10.98%
Same application (DGEMM), Different workload	0.04%	1.27%	5.23%
Same application (FFT), Different workload	0.06%	3.75%	11.56%
Different applications (DGEMM, FFT)	0.08%	1.46%	4.96%

[47]. We strictly follow the detailed methodology explained in [10] and appendix I to compare the energy measurements using RAPL against HCLWattsUp. 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. We present the details on experiment results only for the second use case scenario here. The experiment results for other aforementioned case studies are explained in appendix D.

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. 11a 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 con-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



(a) DGEMM (b) 2D-FFT

FIGURE 11: Dynamic energy profiles of same application with different workload sizes built with RAPL and HCLWattsUp on HCLServer01.

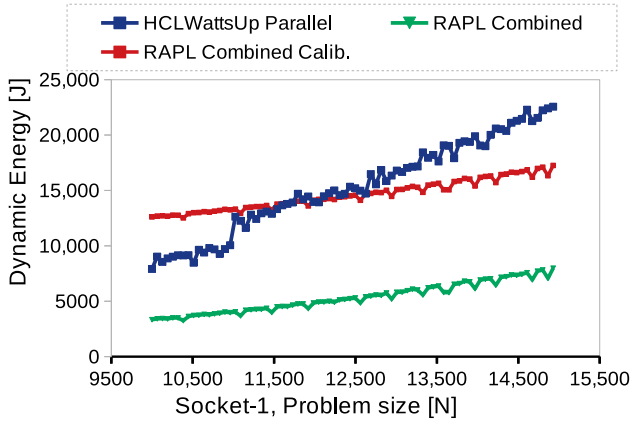


FIGURE 12: Dynamic energy profiles of same application with different workload sizes built with RAPL and HCLWattsUp on HCLServer01 with calibrated RAPL readings.

stant step size of 64. The workload size for AbsCPU2 ranges from 22560 × 22560 to 24992 × 24992 with a constant step size of 64. Fig. 11b 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 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%. We find similar results for our experiments to study the other two aforementioned use case scenarios (the details are presented in appendix D-A).

Another interesting finding is that 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 [10].

While the average error between the both profiles constructed with RAPL and HCLWattsUp can be reduced by calibrating the RAPL readings, the overall qualitative differ-

ence of energy consumption behavior of both profiles can not be improved. One can observe in figure 12 that the overall energy consumption trend (the trend of energy consumption behavior) 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. However, we leave this study for the future.

C. CORE-WIDE DYNAMIC ENERGY CONSUMPTION MODELLING

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. We explain the details in appendix D-B. This suggests that power dissipation by the idle cores can 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.

D. GRANULARITY LIMITATIONS, WORKLOAD TYPES AND SCALABILITY OF ANMOHA

To summarize, AnMoHA is an additive energy modelling approach to address the challenge of how to model accurately the energy consumption of application components when executing them in parallel on multiple independently powered compute devices such as CPUs, GPUs, Xeon Phis, and sockets of multi-socket CPUs. One can use any energy measurement tool to determine the energy consumption by the application component in order to construct the additive energy models.

Currently, it is not possible to determine the dynamic energy consumption by the computing elements which are tightly coupled or which are not independently powered, using system-level measurements using 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 summary, 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.

AnMoHA does not make any assumption about the type of workload and their sizes executed by their corresponding compute devices. Different compute devices can execute different types and sizes of workloads/applications in parallel. One can compute their individual energy consumption using additive approach (AnMoHA), as discussed in section IX.

To illustrate the scalability of AnMoHA on large clusters, 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 functions 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 model based data partitioning algorithm [11] takes the $n \times m$ discrete energy functions 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.

X. STUDY OF ADDITIVE ENERGY MODELLING AND DYNAMIC ENERGY OPTIMIZATION WITH ON-CHIP SENSORS AND HCLWATTSUP

First, we study the additive energy modelling with on-chip built-in sensors and analyze its accuracy against HCLWattsUp (which we consider as ground truth [10]). Then, we study the dynamic energy optimization of a 2D-FFT hybrid 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.

A. ADDITIVE ENERGY MODELLING WITH ON-CHIP SENSORS

HCLServer01 is used for the experiments. We use RAPL to obtain the power consumption by CPU, Nvidia NVML [14] to acquire the power values from on-chip sensors on Nvidia GPUs, and Intel System Management Controller chip (SMC) [15] to obtain the power values from Intel Xeon Phi that can be programmatically obtained using Intel manycore platform software stack (MPSS) [48]. 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 size 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. 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 2, and call them sensors combined and HCLWattsUp combined. Figure 13 shows the dynamic energy profile of 2D-FFT with sensors and HCLWattsUp.

One can observe that sensors combined dynamic energy profile lags behind the parallel dynamic energy profile and has a higher error rate with it 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

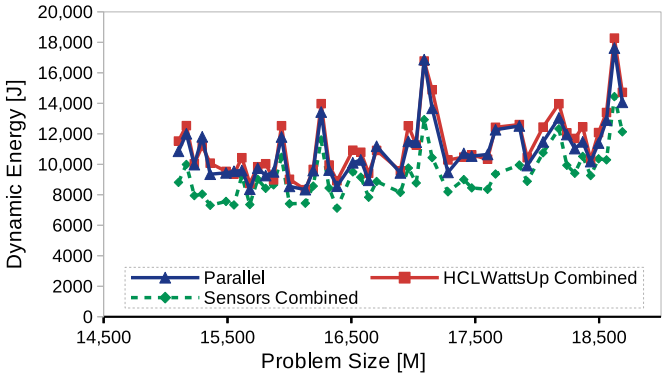


FIGURE 13: Dynamic energy profile of 2D-FFT on HCLServer01.

TABLE 10: 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%

{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 10 provides the percentage deviations from mean and maximum dynamic energy consumption by parallel and both combined dynamic energy profiles.

We present the additive modelling of DGEMM with sensors on HCLServer01 in appendixE-A. In summary, the average and maximum error of sensors combined (DGEMM) dynamic energy profile against parallel profile is {19%, 27.18%}, and HCLWattsUp combined against parallel profile is {3.07%, 7.63%} respectively. The absolute percentage error between the average and maximum percentage deviations of parallel and HCLWattsUp combined dynamic energy profile is {5.44%, 2.82%}, and between parallel and sensors combined dynamic energy profile is {33.08%, 12.45%} respectively.

In summary, additive energy models with sensors provide poor accuracy and poorly model the variations in the combined dynamic energy profile as that of a parallel dynamic energy profile (for the application we used for our experiment). It suggests that the sensor combined dynamic energy profile will not provide similar workload distributions as of parallel profile, if used as an input to the dynamic energy optimization algorithm that leverages the profile variations. In contrast, the combined dynamic energy profile with HCLWattsUp is more accurate and exhibits the better average and maximum variations as of parallel dynamic energy profile.

TABLE 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%

B. A STUDY OF DYNAMIC ENERGY OPTIMIZATION WITH ON-CHIP SENSORS AND HCLWATTSUP

We study the optimization of the parallel hybrid 2D-FFT application and DGEMM for dynamic energy using measurement tools (on-chip built-in) sensors and system-level physical measurements with HCLWattsUp.

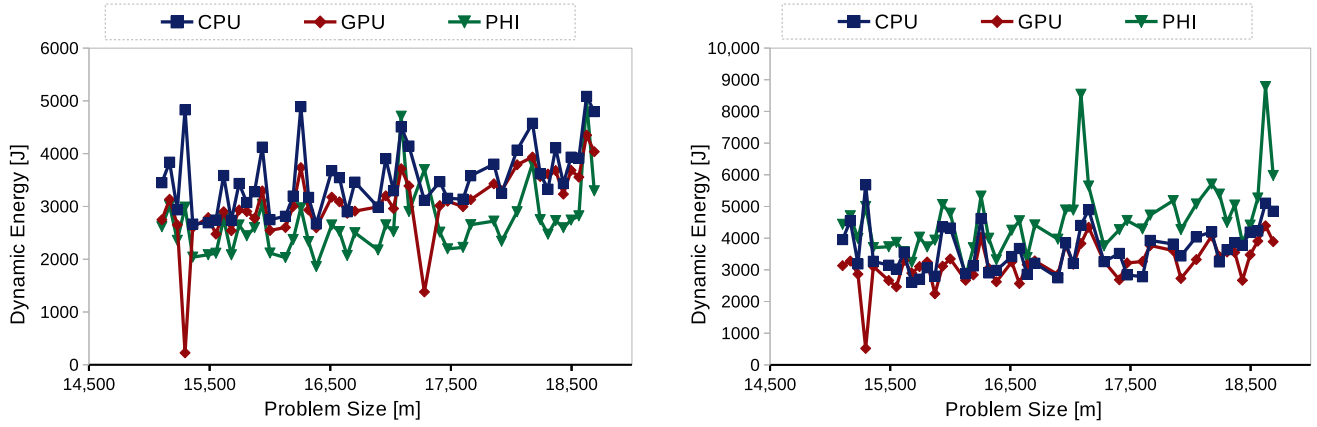
We run the parallel hybrid application 2D-FFT as explained in section VI-A on HCLServer01 for the problem size ranges from 45312 $\times$ 23552 to 56064 $\times$ 23552 with a constant step size of 192. We equally partition the dimension M on all three abstract processors into M_1, M_2 and M_3, so that 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. There is no communication involved in these experiments.

Figures 14a and 14b illustrate the dynamic energy profiles for workload sizes (m) with sensors and HCLWattsUp. One can observe that sensors under report the dynamic energy consumption than HCLWattsUp. Table 11 provides the percentage error of measurements of dynamic energy consumption by 2D-FFT 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 use a model-based data partitioning algorithm [11] 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 functions, iv) the functions of execution time, and v) the discrete dynamic energy functions c 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 [11].

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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



(a) with Sensors.

(b) with HCLWattsUp.

FIGURE 14: Dynamic energy profiles of 2D-FFT on HCLServer01.

We determine the workload distribution for workload sizes {46656,46848,48768,52800,53568,53760,54528} using the dynamic energy profiles with sensors and HCLWattsUp as an input to the data partitioning algorithm [11]. Using this 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 is {42%,39%,45%,38%,37%,38%,36%}. We discuss the results of the dynamic energy optimization of DGEMM on HCLServer01 in appendix E-B. In summary, we find the total dynamic energy losses by using sensors in comparison with HCLWattsUp to optimize the dynamic energy consumption of DGEMM for the workload sizes {40704,41472,42240,43008,44544} is {22%,24%,21%,22%,24%}.

XI. CONCLUSION

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 consumptions of application components when executing different application kernels in parallel on multiple compute devices, ii). Accurate modelling of the energy consumption of two 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 an 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 demonstrated that AnMoHA takes less time to construct the energy profiles when precision settings is reduced. We demonstrated that the average error for the state-of-the-art estimation methods for the same experimental setup ranges from 15% to 75% and the maximum reaches 178%.

Finally, we showed that a significant loss of energy (up to 45% for our applications) occurs when employing state-of-the-art estimation methods instead of our proposed method for dynamic energy optimization of an application.

ACKNOWLEDGEMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

REFERENCES

[1] N. Jones, “How to stop data centres from gobbling up the worldâĂŹs electricity,” Nature, vol. 561, pp. 163–166, 2018.

[2] A. Andrae and T. Edler, “On global electricity usage of communication technology: Trends to 2030,” Challenges, vol. 6, no. 1, p. 117âĂŞ157, Apr 2015.

[3] DOE, “Preliminary conceptual design for an exascale computing initiative,” 2014. [Online]. Available: https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20141121/Exascale_Preliminary_Plan_V11_sb03c.pdf

[4] 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, no. 1, pp. 55–74, Mar 2016.

[5] 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), May 2016, pp. 21–31.

[6] 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, e4067 cpe.4067.

[7] J. Lang and G. RÃijnger, “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.

[8] 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.

[9] 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.

[10] 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. [Online]. Available: https://www.mdpi.com/1996-1073/12/11/2204

[11] H. Khaleghzadeh, M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "Bi-objective Optimisation of Data-parallel Applications on Heterogeneous Platforms for Performance and Energy via Workload Distribution," arXiv:1907.04080 [cs, eess], Jul. 2019, arXiv: 1907.04080. [Online]. Available: http://arxiv.org/abs/1907.04080

[12] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 4, pp. 797–804, April 2008.

[13] 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, no. 2, pp. 20–27, March 2012.

[14] Nvidia, "Nvml reference manual," 10 2018. [Online]. Available: https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf

[15] I. Corporation, "Intel xeon phi coprocessor system software developers guide," 06 2014. [Online]. Available: https://software.intel.com/sites/default/files/managed/09/07/xeon-phi-coprocessor-system-software-developers-guide.pdf

[16] 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, ser. GPGPU-7. ACM, 2014, pp. 28:28–28:36.

[17] 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, 2006, pp. 70–77.

[18] 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, ser. USENIXATC'11. USENIX Association, 2011.

[19] 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 Computing Surveys, vol. 50, no. 3, 2017.

[20] 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.

[21] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The landscape of parallel computing research: A view from berkeley," Tech. Rep. UCB/EECS-2006-183, 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

[22] 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.

[23] 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.

[24] 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. IEEE, 2010, pp. 207–216.

[25] 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, ser. HotPower'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 4–4.

[26] 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, ser. e-Energy '11. New York, NY, USA: ACM, 2011, pp. 1–10.

[27] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," IEEE Transactions on Computers, vol. 61, no. 4, pp. 563–577, Apr. 2012.

[28] 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). ACM, 2005, pp. 186–195.

[29] H. Hong, Sunpyand Kim, "An integrated GPU power and performance model," SIGARCH Comput. Archit. News, vol. 38, no. 3, 2010.

[30] 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.

[31] S. Song, C. Su, B. Rountree, and K. W. Cameron, "A simplified and accurate model of power-performance efficiency on emergent GPU architectures," in 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS). IEEE Computer Society, 2013, pp. 673–686.

[32] Y. S. Shao and D. Brooks, "Energy characterization and instruction-level energy model of Intel's Xeon Phi processor," in Proceedings of the 2013 International Symposium on Low Power Electronics and Design, ser. ISLPED '13. IEEE Press, 2013.

[33] 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, ser. ITCS '13. New York, NY, USA: ACM, 2013, pp. 283–304.

[34] F. Safara, A. Souri, T. Baker, I. Al Ridhawi, and M. Aloqaily, "Prinergy: a priority-based energy-efficient routing method for iot systems," The Journal of Supercomputing, 2020. [Online]. Available: https://doi.org/10.1007/s11227-020-03147-8

[35] V. Balasubramanian, F. Zaman, M. Aloqaily, S. Alrabaee, M. Gorlatova, and M. Reisslein, "Reinforcing the edge: Autonomous energy management for mobile device clouds," in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019, pp. 44–49.

[36] W. Dargie, "A stochastic model for estimating the power consumption of a processor," IEEE Transactions on Computers, vol. 64, no. 5, 2015.

[37] 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. IEEE Computer Society, 2003, p. 93.

[38] D. Hackenberg, R. SchÃűne, 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, May 2015, pp. 896–904.

[39] 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, no. 2, pp. 9:1–9:26, Mar. 2018.

[40] T. Smith and M. Waterman, "Identification of common molecular subsequences," Journal of Molecular Biology, vol. 147, no. 1, pp. 195 – 197, 1981.

[41] O. Gotoh, "An improved algorithm for matching biological sequences," Journal of Molecular Biology, vol. 162, no. 3, pp. 705 – 708, 1982.

[42] T. Rognes, "Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation," BMC bioinformatics, vol. 12, no. 1, p. 1, 2011.

[43] 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.

[44] 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. IEEE, 2014, pp. 184–185.

[45] Heterogeneous Computing Laboratory, University College Dublin, "HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter," 2020. [Online]. Available: https://csgitlab.ucd.ie/ucd-hcl/hclwattsup

[46] IntelPCM, "IntelÂő performance counter monitor - a better way to measure cpu utilization." 2012. [Online]. Available: https://software.intel.com/en-us/articles/intel-performance-counter-monitor

[47] PAPI, "Performance application programming interface 5.4.1," 2015. [Online]. Available: http://icl.cs.utk.edu/papi/

[48] I. Corporation, "IntelÂő manycore platform software stack (Intel MPSS)," 06 2014. [Online]. Available: https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss

[49] 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), March 2019, pp. 123–130.

[50] C. Gough, I. Steiner, and W. Saunders, Energy Efficient Servers Blueprints for Data Center Optimization. Apress, 2015, isbn:978-1-4302-6637-2.
[51] Perf Wiki, "perf: Linux profiling with performance counters," 2017. [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page
[52] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, 1997.
[53] D. W. Mount, Bioinformatics: sequence and genome analysis. Cold Spring Harbor Laboratory Press, 2004, isbn:9780879696870.
[54] NVIDIA, "Parallel Thread Execution ISA Version 5.0," 2016. [Online]. Available: http://docs.nvidia.com/cuda/parallel-thread-execution/#axzz4WCeB6m8U

MUHAMMAD FAHAD is a Ph.D. researcher in Heterogeneous Computing Lab (HCL) at University College Dublin, Ireland. He received his MS degree from KTH - Royal Institute of Technology, Sweden in 2012, and BS degree from International Islamic University Islamabad, Pakistan in 2008. His main research interests include high-performance heterogeneous computing, energy-efficient computing, parallel/distributed and peer-to-peer computing.

ARSALAN SHAHID is a PhD researcher in Heterogeneous Computing Lab at University College Dublin. Arsalan graduated as a gold medalist for the best final year project in BS Electrical Engineering from HITEC University Pakistan, in 2016. His research interests are in energy-aware and high-performance heterogeneous computing.

RAVI REDDY MANUMACHU received a B.Tech degree from I.I.T, Madras in 1997 and a PhD degree from the School of Computer Science, University College Dublin in 2005. His main research interests include high performance heterogeneous computing, distributed computing, energy-efficient computing, and sparse matrix computations.

ALEXEY LASTOVETSKY received a Ph.D. degree from the Moscow Aviation Institute in 1986, and a Doctor of Science degree from the Russian Academy of Sciences in 1997. His main research interests include algorithms, models, and programming tools for high-performance heterogeneous computing. He has published over a hundred technical papers in refereed journals, edited books, and international conferences. He authored the monographs Parallel computing on heterogeneous networks (Wiley, 2003) and High-performance heterogeneous computing (Wiley, 2009).

APPENDIX A SUPPLEMENTAL MATERIAL

The supporting materials for the main manuscript, "Accurate Energy Modelling of Hybrid Parallel Applications on Modern Heterogeneous Computing Platforms in Data Centers" are:

- Empirical validation of additive modelling.
- Trade-off between accuracy and design space of additive modelling.
- Topological granularity limits of additive energy modelling.
- Study of additive energy modelling and dynamic energy optimization with On-chip sensors and HCLWattsUp.
- Rationale behind using dynamic energy consumption instead of total energy consumption.
- Application Programming Interface (API) for measurements using external power meter interfaces (HCLWattsUp).
- Methodology to obtain a reliable data point.
- Methodology to compare measurements using sensors and HCLWattsUp.
- Comparison of dynamic energy consumption using on-chip sensors and HCLWattsUp
- Precautions to prevent interference of other components in dynamic energy consumption
- Accuracy of PMC-based energy predictive models against HCLWattsUp
- Parallel Gene Sequencing Application

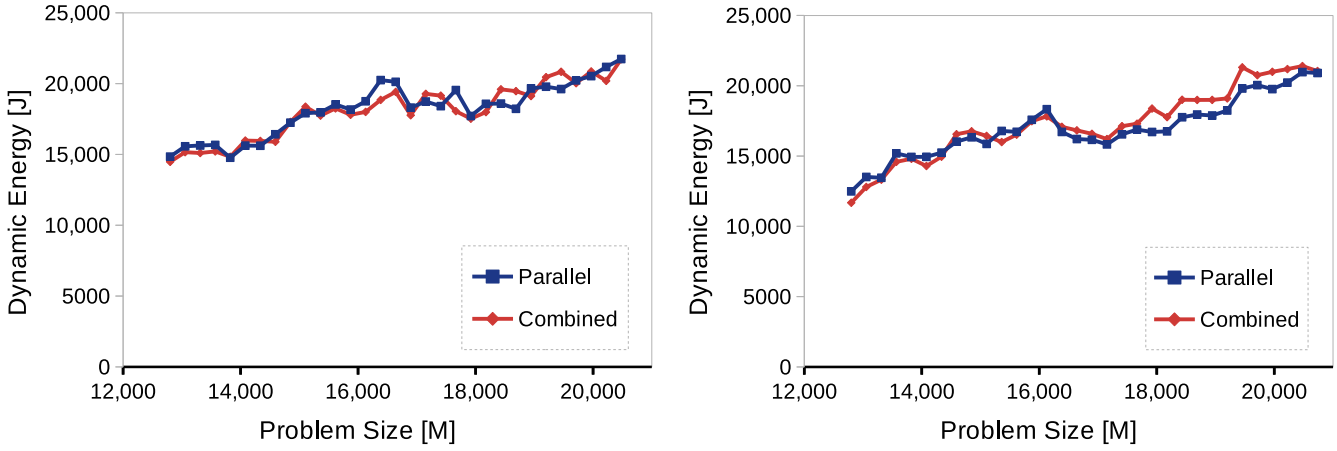
APPENDIX B EMPIRICAL VALIDATION OF ADDITIVE MODELLING

We run three different workload configurations of DGEMM executing on HCLServer01 and for each configuration build four dynamic energy functions of DGEMM as explained in section VI-B. The workload sizes range for all three configurations are: i) from 12800×20224 to 20224×20224 with a constatn step size of 128, ii). from 12800×20480 to 20480×20480 with a constant step size of 256 for the dimension M, and, iii). 12800×20736 to 20736×20736 with a constant step size of 256 for the dimension M. Figure 15a and 15b illustrate the parallel and combined dynamic energy profiles of experiment configurations: 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.

APPENDIX C TRADE-OFF BETWEEN ACCURACY AND DESIGN SPACE OF ADDITIVE MODELLING

We can determine 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 $\{E_C(x), \{E_{AC}(x) - E_A(x)\}, \{E_{BC}(x) - E_B(x)\}, \{E_{ABC}(x) - E_{AB}(x)\}, \{E_{ABC}(x) - E_B(x) - E_A(x)\}\}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



(a) N=20480 (b) N=20736

FIGURE 15: Dynamic energy profiles of DGEMM on HCLServer01.

Figure 16 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.

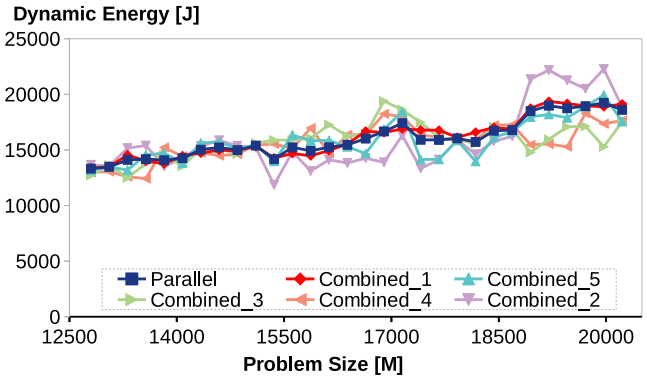


FIGURE 16: Trade off between number of experiments and accuracy.

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 12 presents the percentage deviations from the mean and maximum of dynamic energy consumption by parallel and each combined dynamic energy profiles.

TABLE 12: Percentage deviations from mean 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%

APPENDIX D TOPOLOGICAL GRANULARITY LIMITS OF ADDITIVE ENERGY MODELLING

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 × N). The workload size (N × N) for AbsCPU1 ranges from 20000 × 20000 to 22432 × 22432 with a constant step size of 64. The workload size for AbsCPU2 ranges from 22560 × 22560 to 24992 × 24992 with a constant step size of 64. Fig. 17b 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. 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 3.7% and 11.56% respectively. In contrast, RAPL under-reports the dynamic energy consumption as compared with HCLWattsUp.

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 × N) for AbsCPU1 ranges from 7680 × 30720 to 16896 × 30720 with a constant step size of 512; and for AbsCPU2, it ranges from 23040 × 30720 to 32256 × 30720 with a constant step size of 512. Fig. 17a 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 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. 18 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%.

A. STATE-OF-THE ART ENERGY MEASUREMENT TOOLS

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. 19a 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.

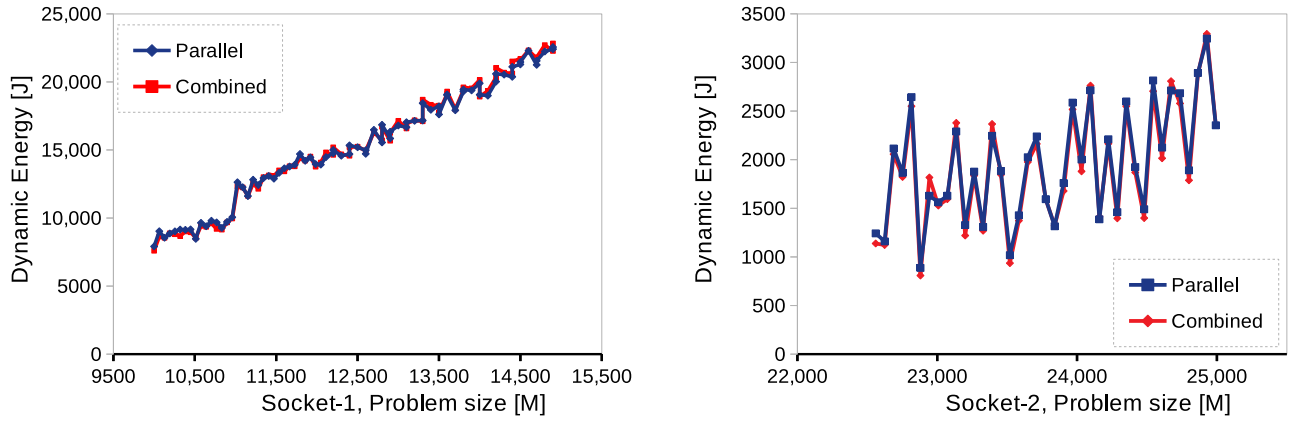
The average and maximum errors between parallel dynamic energy profiles built with RAPL and HCLWattsUp are 21% and 109% respectively. The average and maximum errors between parallel dynamic energy profile built with HCLWattsUp and combined dynamic energy profile built with RAPL are 16% and 77% respectively. 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. 19b 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 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.

To summarize, 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 equally effective way as device-wide (CPU, GPU, Xeon PHI as explained in section VI). This is because, each socket of Intel Haswell E5-2670V3 is independently powered, and there is minimal resource sharing when running parallel two different applications pinned on individual sockets. Hence, AbsCPU holds both the propositions of additive hypothesis: completeness and loose-coupling (as explained in section V-A), and that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



(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 17: Socket-wide dynamic energy profiles for case study B: Same application, Different workloads.

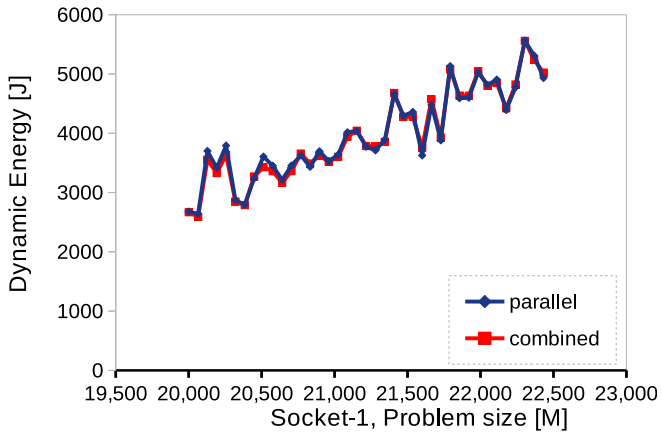


FIGURE 18: Case study C: Different applications. x-axis scale represents the problem size range for DGEMM executed on AbsSoc1.

is why we observe that additive hypothesis is equally validate for socket-wide dynamic energy consumption. Another important finding is that on-chip power sensors does not capture the holistic picture of the energy consumption trend when running two applications in parallel each on a different socket.

B. CORE-WIDE DYNAMIC ENERGY CONSUMPTION MODELLING

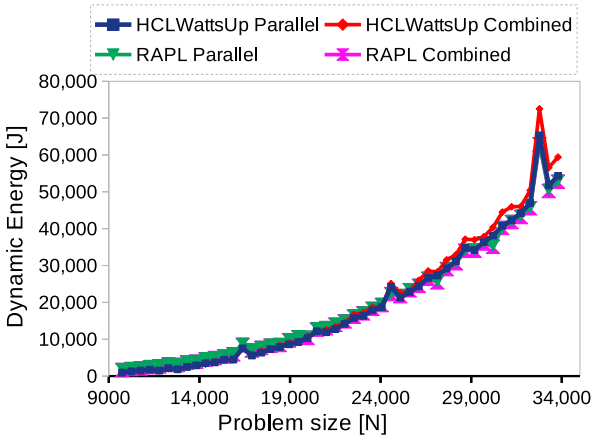
In this section, we explore if we can further fine the topological granularity to core-wide. For this study, we explore different core-wide combinations (such as 1-core, 2-core, etc) on each HCLServer (technical details are provided in table 1) to formulate the abstract processors (AbsCore) for measuring the dynamic energy consumption by the application running on them. We follow the 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 13312×13312 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 10752×10752 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 on HCLServer01.

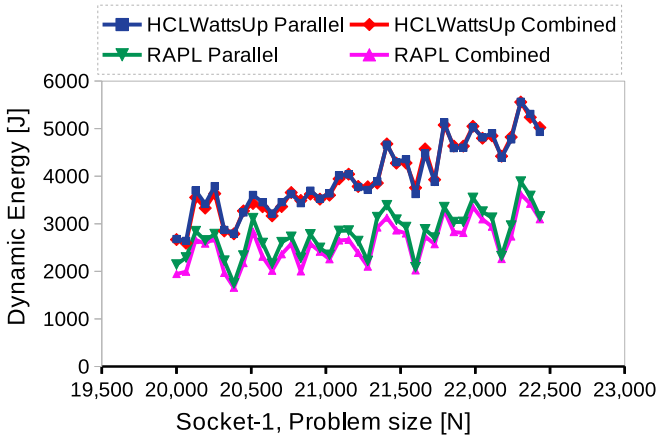
The total power consumption by a complementary metal-oxide-semiconductor (CMOS) can be roughly considered as the sum value of idle power and dynamic power. Here, the idle power is the (leakage) power dissipation when it does not running the application. The usual settings of DVFS governor (such as Ondemand or conservative governors in Linux) sets the voltage and clock frequency of the cores depending on the current system load. That is, it keeps the voltage and clock frequency of cores at low when idle, and scale them up as soon as the workload is supplied. While frequency can be scaled up/down at core-wide, the cores cannot regulate their voltages independent of each other because the same voltage is supplied across all the cores sharing the same voltage domain. While the dynamic power part is proportional to the square of the voltage, the (leakage or idle) power scales exponentially with voltage [49].

When a workload is executed on a some of CPU cores of a socket, then the DVFS governor scales up the voltage of entire socket according to change in clock frequency of active cores. However, the idle fraction of the CPU cores of the socket dissipates its power as leakage. Because of higher voltage supply, this power dissipation by idle CPU

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



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



(b) Case Study C: Different applications

FIGURE 19: Socket-wide dynamic energy profiles built with RAPL and HCLWattsUp on HCLServer01.

cores is higher than the power dissipation when the voltage supply was low. The total dynamic energy consumption by the application running sequentially on AbsCore processors measured with HCLWattsUp includes both the dynamic power consumed by the active CPU cores and higher idle power dissipated by the idle CPU cores. However, this is not the case when we run the applications on both AbsCore abstract processors. As a result, we observe a higher combined dynamic energy consumption than the parallel one. This suggests that power dissipation by the idle cores can contribute significantly to the total power consumption by the CPU when running an application on some of the CPU cores. Hence, the dynamic energy consumption by the socket can be optimized by switching the idle cores off. However it can introduce the overhead of a switching them on/off mechanism. Alternatively, we can leverage the CPU by utilizing all of its cores, however, it may introduce diminishing effects for some workload types. We, nevertheless, leave this study for 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.

APPENDIX E STUDY OF ADDITIVE ENERGY MODELLING AND DYNAMIC ENERGY OPTIMIZATION WITH ON-CHIP SENSORS AND HCLWATTSUP

A. ADDITIVE ENERGY MODELLING WITH ON-CHIP POWER SENSORS AND HCLWATTSUP

We run DGEMM for the workload size ranging from 12800 $\times$ 20480 to 20480 $\times$ 20480 with a constant step size of 256. 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. Then, we build dynamic energy functions for each abstract processor executing the DGEMM individually with on-chip sensors and HCLWattsUp separately, and compose the combined dynamic energy profiles. We call them sensors combined and HCLWattsUp combined. Figure 20 shows the dynamic energy profile of DGEMM with sensors and HCLWattsUp.

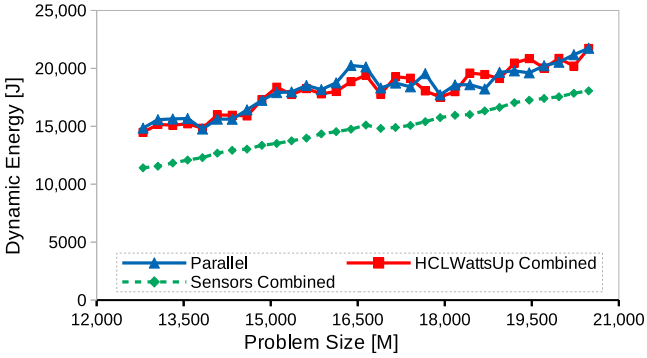


FIGURE 20: Dynamic energy profile of DGEMM on HCLServer01.

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 average 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

TABLE 13: 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%

the sensors combined dynamic energy profile and parallel dynamic energy profiles is {33.08%, 12.46%}. Table 13 provides the percentage deviations from mean and maximum dynamic energy consumption by parallel and both combined dynamic energy profiles.

In summary, additive energy models with sensors have poor accuracy and poorly exhibit the variations in the combined dynamic energy profile as of parallel dynamic energy profile (for the application set we used for our experiment). It suggests that we can not benefit from the sensor combined dynamic energy profile in an equally effective way as parallel dynamic energy profile to be used as an input to the dynamic energy optimization algorithm where workload distribution is used as an decision variable. In contrast, combined dynamic energy profile with HCLWattsUp is more accurate and exhibits the better average and maximum variations as of parallel dynamic energy profile.

B. A STUDY OF DYNAMIC ENERGY OPTIMIZATION WITH ON-CHIP SENSORS AND HCLWATTSUP

In this section, We study the optimization of parallel hybrid DGEMM application for dynamic energy using measurement tools (on-chip built in) sensors and system-level physical measurements with HCLWattsUp.

We run a parallel hybrid application DGEMM as explained in section I on HCLServer01 for the problem size ranges from 38400×61440 to 61440×61440 with a constant step size of 768. 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. There is no communication involved in these experiments.

Figures 21a and 21b illustrate the dynamic energy profiles for workload sizes (m) with sensors and HCLWattsUp. One can observe that sensors under report the dynamic energy consumption than HCLWattsUp. Table 14 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. In section X-A, we explained the percentage differences of average and maximum deviations from mean dynamic energy consumption by sensor and HCLWattsUp combined dynamic energy profiles.

We use a model-based data partitioning algorithm [11] to compute the decomposition of dimension M. The brief

TABLE 14: 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%

details on the inputs and outputs of the algorithm, and the algorithm is presented in section X-B. 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.

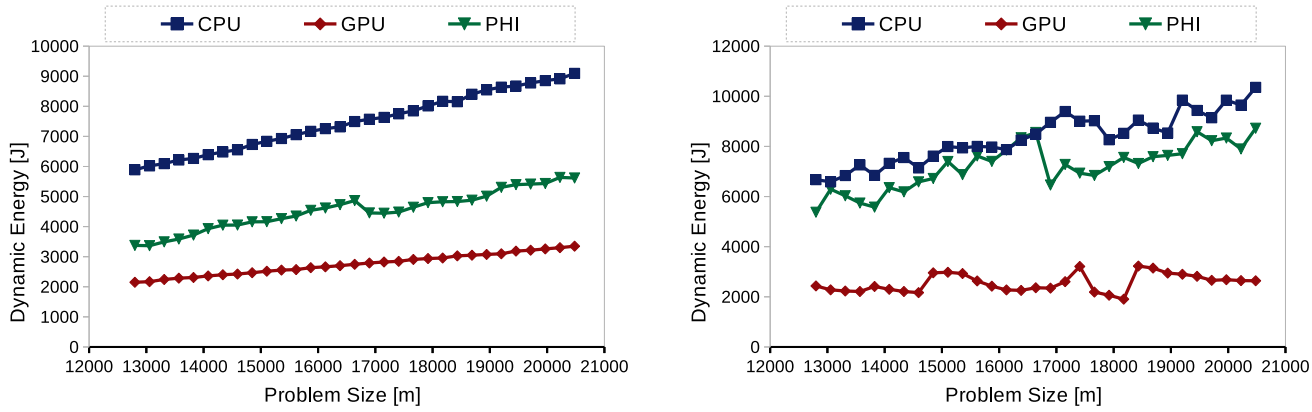
The data partitioning algorithm takes the dynamic energy functional models as an input and finds the optimal workload configuration which optimizes the total dynamic energy consumption for the given application using load imbalance technique. We determine the workload distribution for workload sizes where M {40704,41472,42240,43008,44544} using the dynamic energy profiles with sensors and HCLWattsUp as an input to the data partitioning algorithm. Using this 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%}.

APPENDIX F RATIONALE BEHIND USING DYNAMIC ENERGY CONSUMPTION INSTEAD OF TOTAL ENERGY CONSUMPTION

We consider only the dynamic energy consumption in our work for reasons below:

1) The static energy consumption is a major concern in embedded systems. However, it is becoming less compared to the dynamic energy consumption due to advancements in hardware architecture design in HPC systems.
2) We target the applications and platforms where dynamic energy consumption is the dominating energy dissipator.
3) Finally, we believe its inclusion can underestimate the true worth of an optimization technique that optimizes the dynamic energy consumption (i.e. the minimal energy consumption under service-related constraints such as performance). We elucidate 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 respectively. It would report the prediction error

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access



(a) with Sensors.

(b) with HCLWattsUp.

FIGURE 21: Dynamic energy profiles of DGEMM on HCLServer01.

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% respectively. Therefore, the second model M_B should be considered more accurate than the first.

APPENDIX G APPLICATION PROGRAMMING INTERFACE (API) FOR MEASUREMENTS USING EXTERNAL POWER METER INTERFACES (HCLWATTSUP)

HCLServer01 and HCLServer02 have a dedicated power meter installed between their input power sockets and wall A/C outlets. The power meter captures the total power consumption of the node. It has a data cable connected to the USB port of the node. A perl script collects the data from the power meter using the serial USB interface. The execution of this script is non-intrusive and consumes insignificant power.

We use *HCLWattsUp* API function, which gathers the readings from the power meters to determine the average power and energy consumption during the execution of an application on a given platform. *HCLWattsUp* API can provide following four types of measures during the execution of an application:

- *TIME*—The execution time (seconds).
- *DPOWER*—The average dynamic power (watts).
- *TENERGY*—The total energy consumption (joules).
- *DENERGY*—The dynamic energy consumption (joules).

We confirm that the overhead due to the API is very minimal and does not have any noticeable influence on the main measurements. It is important to note that the power meter readings are only processed if the measure is not $hcl :: TIME$. Therefore, for each measurement, we have two runs. One run for measuring the execution time. And the other for energy consumption. The example provided in figure 22 illustrates the use of statistical methods to measure the dynamic energy consumption during the execution of an application.

The API is confined in the *hcl* namespace. 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 type of measure, the path to the executable *executablePath*, the arguments to the executable *executableArgs* and the statistical thresholds (*pIn*) The outputs are the achieved statistical confidence *pOut*, the estimators, the sample mean (*sampleMean*) and the standard deviation (*sd*) calculated during the execution of the executable.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access


```
#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 22: Example illustrating the use of HCLWattsUp API for measuring the dynamic energy consumption

argument $pOut$. At the same time, the sample mean and standard deviation are returned. For our experiments, we use values of (1000, 95%, 2.5%, 3600) for the parameters $(maxRepeats, cl, maxStdError, maxElapsedTime)$. Since we use Student's t-distribution for the calculation of the confidence interval of the mean, we confirm specifically that the observations follow normal distribution by plotting the density of the observations using *R* tool.

APPENDIX H METHODOLOGY TO OBTAIN A RELIABLE DATA POINT

We follow the following strict methodology described below to make sure the experimental results are reliable:

- The server is fully reserved and dedicated to these experiments during their execution. We also made certain that there are no drastic fluctuations in the load due to abnormal events in the server by monitoring its load continuously for a week using the tool *sar*. Insignificant variation in the load was observed during this monitoring period suggesting normal and clean behaviour of the server.
- We set the application kernel's CPU affinity mask using SCHED API's system call SCHED_SETAFFINITY() Consider for example mkl-DGEMM application kernel running on HCLServer01. To bind this application kernel, we set its CPU affinity mask to 12 physical CPU cores of Socket 1 and 12 physical CPU cores of Socket 2.
- To make sure that pipelining, cache effects and so forth, do not happen, the experiments are not executed in a loop and sufficient time (120 s) is allowed to elapse between successive runs. This 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.
- To obtain a data point, the application is repeatedly executed 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. We verify the validity of these assumptions 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 15–17).
  3) The elapsed time of the repetitions of application execution has exceeded the maximum time allowed ($maxT$ in seconds) (Lines 18–20).

  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 [45] 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).

APPENDIX I METHODOLOGY TO COMPARE MEASUREMENTS USING ON-CHIP 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. HCLWattsUp API provides the dynamic energy consumption of 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, only provide the power consumption of GPU or Xeon Phi only (we offload the applications to run on Intel Xeon Phi, so it includes the CPU host core, DRAM and PCIe to copy and migrate the data between CPU host core and Xeon Phi). Therefore, to compare both methodologies in a most fairly equitable way and to obtain the dynamic energy profiles of applications, we use RAPL as an aide to sensors for determining the application energy. Because, RAPL determine the power consumption of CPU and DRAM using on-chip voltage regulator and current sensor [50].

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.

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

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

$reps \leftarrow 0$; $stop \leftarrow 0$; $sum \leftarrow 0$; $etime \leftarrow 0$
while ($reps < maxReps$) and (!$stop$) **do**
  $st \leftarrow$ MEASURE($TIME$)
  EXECUTE(app)
  $et \leftarrow$ MEASURE($TIME$)
  $reps \leftarrow reps + 1$
  $etime \leftarrow etime + et - st$
  $ObjArray[reps] \leftarrow et - st$
  $sum \leftarrow sum + ObjArray[reps]$
  if $reps > minReps$ **then**
    $clOut \leftarrow$ fabs(gsl_cdf_tdist_Pinv(cl, $reps - 1$)) $\times$ gsl_stats_sd($ObjArray$, 1, $reps$) / sqrt($reps$)
    if $clOut \times \frac{reps}{sum} < eps$ **then**
      $stop \leftarrow 1$
    end if
    if $etime > maxT$ **then**
      $stop \leftarrow 1$
    end if
  end if
end while
$repsOut \leftarrow reps$; $epsOut \leftarrow clOut \times \frac{reps}{sum}$
$etimeOut \leftarrow etime$; $mean \leftarrow \frac{sum}{reps}$
end procedure

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

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 server (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 server, 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.

APPENDIX J COMPARISON OF DYNAMIC ENERGY CONSUMPTION USING ON-CHIP SENSORS AND HCLWATTSUP

We use Nvidia NVML [14] to acquire the power values from on-chip sensors on Nvidia GPUs and Intel System Management Controller chip (SMC) [15] to obtain the power values from Intel Xeon Phi that can be programmatically obtained using Intel manycore platform software stack (MPSS) [48]. The steps (methodology) taken to compare the measurements using RAPL, and GPU/Xeon Phi sensors against HCLWattsUp are presented in appendix I. Briefly, HCLWattsUp API provides the dynamic energy consumption of an application using both CPU and an accelerator (GPU or Xeon Phi) instead of the components involved in its execution. 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. On-chip power sensors (NVML and MPSS) only provide the power consumption of GPU or Xeon Phi only. Therefore, to obtain the dynamic energy profiles of applications, we use RAPL to determine the energy contribution of CPU and DRAM.

First, we discuss a comparison of accuracy of the energy measurements reported by Intel RAPL with those provided by the ground truth. We build the dynamic energy profiles of MKL-FFT on Intel Skylake Gold 6152 (HCLServer02) with RAPL and HCLWattsUp. The workload sizes for MKL-FFT range from 22400×22400 to 41536×41536 with a step size of 64. Figure 23 illustrates the dynamic energy profiles of 2D MKL-FFT. RAPL does not exhibit the same dynamic energy consumption behavior as that of the ground truth.

We find many such data points where the dynamic energy consumptions reported by both the tools disagree. Consider, for example, the problem sizes where M is $30272, 34176, 37760, 38080$. For these problem sizes, RAPL suggests a percentage decrease of $12.39, 31.33, 18.77, 5.4$ with respect to their immediate data points in the energy profile whereas HCLWattsUp reports an percentage increase of $11.68, 35.84, 6.46, 31.25$. Similarly, RAPL suggests an percentage increase of $1.26, 19.9, 40.87, 4.74$ for the problem sizes where M is $22528, 28288, 33920, 33152$ with respect to their immediate preceding data-points in energy profiles whereas HCLWattsUp suggests a percentage decrease of $22.24, 4.79, 9.54, 6.02, 28.75$ for them. Owing to the nature of the interlacing nature of the measurements reported by RAPL, we can not calibrate RAPL to reduce its average error with respect to the ground truth. We find the maximum and average error of RAPL to be 156% and 29% respectively.

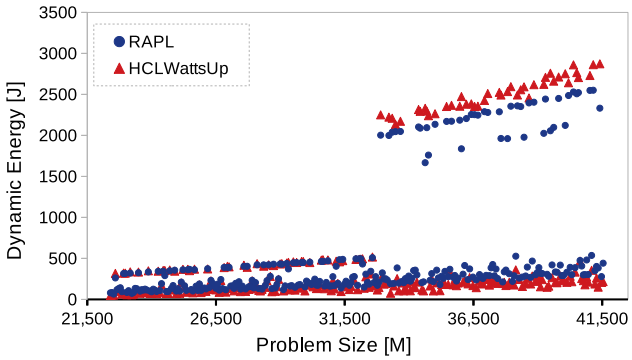


FIGURE 23: Dynamic energy profiles of 2D MKL-FFT on Intel Skylake Gold 6152 with RAPL and HCLWattsUp.

On Intel Xeon Phi 3120P on HCLServer01, we build the dynamic energy profile of Intel MKL-DGEMM with sensors (RAPL and MPSS) as a function of problem size (M $\times$ N) ranges from $7936 \times 13{,}824$ to $13{,}824 \times 13{,}824$ with a constant step size of 256. Figure 24 illustrates the dynamic energy profiles. We find that MPSS reports on the average higher dynamic energy consumption than the ground truth used in our experiments. The average and maximum error of the sensor profile is 64.5% and 93.06% respectively.

APPENDIX K PRECAUTIONS TO PREVENT INTERFERENCE OF OTHER COMPONENTS IN DYNAMIC ENERGY CONSUMPTION

We take several precautions in computing energy measurements to eliminate any potential interference of the computing elements that are not part of the abstract processor running the given application kernel. Consequently, it ensures that the dynamic energy of an abstract processor computed in this way solely represents the dynamic energy consumed by the constituent computing elements of the very abstract processor. For this, we take following precautions:

1) We group abstract processors in such a way that an abstract processor must be constituting solely the com-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

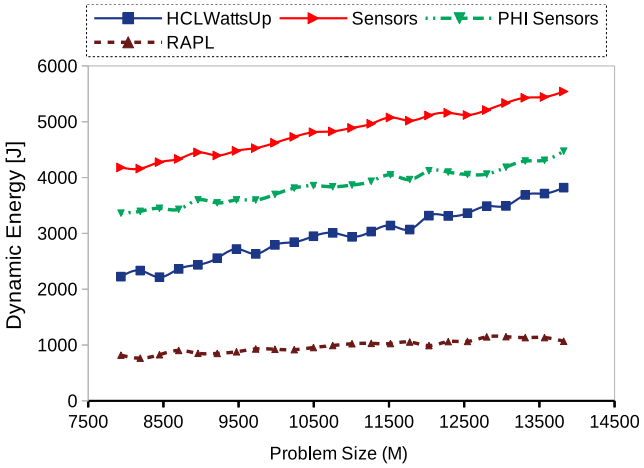


FIGURE 24: Dynamic Energy profiles of MKL-DGEMM on Intel Xeon Phi 3120P.

puting elements which are involved to run an application kernel. The application kernel will, in this way, only use the computing elements of the abstract processor executing it and do not use any other component for its execution. Hence, the dynamic energy consumption will solely reflect the work done by the computing elements of the abstract processor executing the application kernel.

Consider for example mkl-DGEMM application kernel executing on only abstract processor A (comprises of CPU and DRAM). However, HCLWattsUp API gives the total energy consumption of the server during the execution of an application. This includes the contribution from all components such as NIC, SSDs, fans, etc. Therefore, to rule out their contribution in dynamic energy consumption, we ensure all the components other than CPUs and DRAM are not used during the execution of an application. In this way, the dynamic energy consumption that we obtain using HCLWattsUp API reflects only the contribution of CPUs and DRAM. For this, we follow the below steps to verify if these other components are not used:

- We monitor the disk consumption before and during the application run and ensure that there is no I/O performed by the application using tools such as sar, iotop, etc.
- we ensure that the problem size used in the execution of an application does not exceed the main memory and that swapping (paging) does not occur.
- We ensure that the network is not used by the application by monitoring using tools such as sar, atop, etc.
- We set the application kernel's CPU affinity mask using SCHED API's system call SCHED_SETAFFINITY() respecting abstract processors formulation guidelines. Consider for example mkl-DGEMM application kernel executing on only abstract processor A. To bind this application kernel, we set its CPU affinity mask to 12 physical CPU cores of Socket 1, and 12 physical CPU cores of Socket 2.

2) Fans are also a great contributor to energy consumption. On our platform fans are controlled in two zones: a) zone 0: CPU or System fans, b) zone 1: Peripheral zone fans. There are 4 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%), with 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. Consequently, the energy consumption by the fans also changes with the change in their speed. Higher the temperature of CPU, for example, higher the fans speed of zone 0 to cool it down, and as a consequence the energy consumption also gets higher. This energy consumption to cool the server down, therefore, is not consistent and is dependent on the fans speed, and consequently effects the dynamic energy consumption of the given application kernel. Hence, to rule out fans' contribution in dynamic energy consumption, we set the fans at full speed before launching the experiments. When set at full speed, the fans on our platform run consistently at $\sim$13400 rpm, and do not change their speed until we do so to another speed level. In this way, fans consumed same amount of power which is included in static power of the server. We monitor the temperature of server and speed of the fans (after setting it at full) using Intelligent Platform Management Interface (IPMI) sensors, both with and without the application run. We find no considerable difference in temperature, and find the speed of fans same in both scenarios.

Thus, we ensure that the dynamic energy consumption obtained using HCLWattsUp, reflects the contribution solely by the abstract processor executing the given application kernel.

APPENDIX L ACCURACY OF PMC-BASED ENERGY PREDICTIVE MODELS AGAINST HCLWATTSUP

In this section , we compare the prediction accuracy of linear energy predictive models employing performance monitoring counters (PMCs) as predictor variables against HCLWattsUp. Likwid [24], Linux Perf [51], PAPI [47] and Intel PCM [46], are some popular tools which can be used to read the PMCs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2994953, IEEE Access

data on a given platform. However, we use Likwid for PMC collection. We can collect PMC data for an application using Likwid with a simple command-line invocation as shown below: likwid-perfctr -f -C S0:0-11@S1:12-23 -g EVENTS APP here APP represents the test application and EVENTS represents the PMCs. Our 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. Table 15 presents the list of applications employed in our experiments.

TABLE 15: Applications suite for comparing the accuracy of PMC based energy predictive models against Power meters.

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

We measure following three quantities during the execution of an application on our platforms. i) Dynamic energy consumption with HCLWattsUp API, ii) the execution time, and iii) all the PMCs available on our platforms. The applications are pinned to physical CPU cores of our platforms during its execution. We use likwid-pin to bind the applications to CPU cores. We use numactl which is a command-line linux tool to pin the applications to the memory blocks.

Likwid offers 164 PMCs and 385 PMCs for Intel Haswell and Intel Skylake platform respectively. We eliminate PMCs from our dataaset with counts less than or equal to 10 because a) we find them to have no significance for modeling the dynamic energy consumption, and b) they are non-reproducible.

The reduced set contains 323 for Intel Skylake and 151 PMCs for Intel Haswell. It takes significant time to collect all of them because only 4 PMCs can be collected in a single application run. Furthermore, some PMCs can only be collected individually or in sets of two or three for an application run. Therefore, we observe that an application must be run about 53 and 99 times on Intel Haswell and Intel Skylake platforms, to collect all the PMCs.

The mathematical form of the linear regression models can be stated as follows: $\forall a = (a_k)_{k=1}^{n}, a_k \in \mathbb{R}$,

$$f_{_E}(a) = \beta_0 + \beta \times a = \sum_{k=1}^{n} \beta_k \times a_k \tag{10}$$

where β_0 is the intercept and $\beta = \{\beta_1, ..., \beta_n\}$ is the vector of regression coefficients. To capture the stochastic noise (measurement errors), the measured energy is typically expressed as

$$\tilde{f}_{_E}(a) = f_{_E}(a) + \epsilon \tag{11}$$

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

Further details on our experiment methodology and results can be found in [10].

APPENDIX M PARALLEL GENE SEQUENCING APPLICATION

We use a gene sequencing application HCLSW executing the Smith-Waterman algorithm ([40], [41]) 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 [52] 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 [52] 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 [40] with modifications due to [41] using affine gap penalty functions are shown below ([42]):

$$H_{i,j} = \begin{cases} max \begin{cases} H_{i-1,j-1} + P[q_i, d_j] \\ E_{i,j} \\ F_{i,j} \\ 0 \end{cases} & , \quad i > 0 \cap j > 0 \\ 0, & i = 0 \cup j = 0 \end{cases} \tag{12}$$

$$E_{i,j} = \begin{cases} max \begin{cases} H_{i,j-1} - Q \\ E_{i,j-1} - R \\ 0 \end{cases}, & j > 0 \\ 0, & j = 0 \end{cases} \quad (13)$$

$$F_{i,j} = \begin{cases} max \begin{cases} H_{i-1,j} - Q \\ E_{i-1,j} - R \\ 0 \end{cases}, & i > 0 \\ 0, & i = 0 \end{cases} \quad (14)$$

$$S = \max_{1 \le i \le m \cap 1 \le j \le n} H_{i,j} \quad (15)$$

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 sequencies, 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 [53].

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 [42]. This package contains highly optimized implementation of SW algorithm using SIMD parallelization (for example: using the SSE3 intrinsics offered by latest Intel processors).
- CUDASW++3.0 for nVidia GPU accelerators [43]. 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 [54] for nVidia Tesla GPUs.
- SWAPHI for Intel Xeon Phi accelerators [44]. 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.