Multicore Processor Computing is not Energy Proportional: An Opportunity for Bi-objective Optimization for Energy and Performance

Semyon Khokhriakov, Ravi Reddy Manumachu, and Alexey Lastovetsky

School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland

Abstract

Energy proportionality is the key design goal followed by architects of multicore processors. One of its implications is that optimization of an application for performance will also optimize it for energy.

In this work, we show that energy proportionality does not hold true for multicore processors. This finding creates the opportunity for bi-objective optimization of applications for energy and performance. We propose and study a novel application-level bi-objective optimization method for energy and performance for multithreaded dataparallel applications. The method uses two decision variables, the number of identical multithreaded kernels (threadgroups) executing the application and the number of threads per threadgroup, with a given workload partitioned equally between the threadgroups.

We experimentally demonstrate the efficiency of the method using four popular and highly optimized multithreaded data-parallel applications, two employing two-dimensional fast Fourier transform and the other two, dense matrix multiplication. The experiments performed on four modern multicore processors show that the optimization for performance alone results in increase in dynamic energy consumption by up to 89% and optimization for dynamic energy alone results in performance degradation by up to 49%. By solving the bi-objective optimization problem, the method determines up to 11 Pareto-optimal solutions.

Finally, we propose a qualitative dynamic energy model employing performance events as variables to explain the discovered energy nonproportionality. The model shows that the energy nonproportionality on our experimental platforms for the two data-parallel applications is due to disproportionately energy expensive activity of the data translation lookaside buffer.

Keywords: energy proportionality, multicore processor, energy optimization, energy predictive models, bi-objective optimization, fast Fourier transform, matrix multiplication

1. Introduction

Computing is omnipresent in our digital era, and the share of computing platforms in the total energy consumption is rapidly increasing. Nicola Jones [1] reports that the energy consumption of computing systems and devices accounts for 20% of the global electricity demand. Andrae and Edler [2] predict that computing systems and devices will consume up to 50% of global electricity in 2030 with a contribution towards greenhouse gas emissions of 23%. This makes the energy of computing the next grand technological challenge. Multicore CPUs are at the heart of modern computing platforms. Therefore, their energy efficiency is critical for addressing the challenge of energy of computing.

Architects of modern multicore processors (CPUs) follow a key design goal called energy proportionality (defined by Barroso and Hölzle [3] and extended by Sen and Wood [4]), which means designing microprocessors composed of components that consume energy proportional to the amount of work performed. One of the implications of energy proportionality is that optimization of an application for performance will also optimize it for energy.

Modern multicore CPUs however have many inherent complexities, which are: a) Severe resource contention due to tight integration of tens of cores organized in multiple sockets with multi-level cache hierarchy and 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) where the time for memory access

between a core and main memory is not uniform and where main memory is distributed between locality domains or groups called NUMA nodes; and c) Dynamic power management (DPM) of multiple power domains (CPU sockets, DRAM). The complexities were shown to result in complex (non-linear and non-smooth) functional relationships between performance and workload size and between dynamic energy and workload size for real-life data-parallel applications on modern multicore CPUs (Lastovetsky and Reddy [5], Reddy and Lastovetsky [6], Reddy and Lastovetsky [7]). Motivated by these research findings and based on further deep exploration, we show that energy proportionality does not hold true for multicore CPUs. This creates the opportunity for bi-objective optimization of applications for energy and performance on a single multicore CPU.

We present now an overview of notable state-of-the-art methods solving the bi-objective optimization problem of an application for energy and performance on multicore CPU platforms. System-level methods are introduced first since they dominated the landscape. This will be followed by recent research in application-level methods. Then we describe the proposed solution method solving the bi-objective optimization problem of an application for energy and performance on a single multicore CPU.

1.1. Overview of System-level Solution Methods

Solution methods solving the bi-objective optimization problem for energy and performance can be broadly classified into *system-level* and *application-level* categories. System-level methods aim to optimize energy and performance of the environment where the applications are executed. The methods employ application-agnostic models and hardware parameters as decision variables. They are principally deployed at operating system (OS) level and therefore require changes to the OS. They do not involve any changes to the application. The methods can be further divided into the following prominent groups:

I. Thread schedulers that are contention-aware and that exploit cooperative data sharing between threads (Petrucci, Loques, Mossé, Melhem, Gazala, and Gobriel [8], Kim, Kim, and Chung [9]). The goal of a scheduler is to find thread-to-core mappings to determine Pareto-optimal solutions for energy and performance. The schedulers operate at both user-level and OS-level with those at OS-level requiring changes to the OS. Thread-to-core mapping is the key decision variable. Performance monitoring counters such as LLC miss rate and LLC access rate are used for predicting the performance given a thread-to-core mapping.

II. Dynamic private cache (L1 and L2) reconfiguration and shared cache (L3) partitioning strategies (Wang, Mishra, and Ranka [10], Zhuravlev, Saez, Blagodurov, Fedorova, Prieto [11], Cheng, Huang, Huang, and Knoll [12]). The proposed solutions in this category mitigate contention for shared on-chip resources such as last level cache by physically partitioning it and therefore require substantial changes to the hardware or OS.

III. Thermal management algorithms that place or migrate threads to not only alleviate thermal hotspots and temperature variations in a chip but also reduce energy consumption during an application execution (Yang, Zhou, Chrobak, Zhang, and Jin [13], Ayoub and Rosing [14]). Some key strategies are dynamic power management (DPM) where idle cores are switched off, Dynamic Voltage and Frequency Scaling (DVFS), which throttles the frequencies of the cores based on their utilization, sand migration of threads from hot cores to the colder cores.

IV. Asymmetry-aware schedulers that exploit the asymmetry between sets of cores in a multicore platform to find thread-to-core mappings that provide Pareto-optimal solutions for energy and performance (Li, Baumberger, Koufaty, and Hahn [15], Humenay, Tarjan, and Skadron [16]). Asymmetry can be explicit with fast and slow cores or implicit due to non-uniform frequency scaling between different cores or performance differences introduced by manufacturing variations. The key decision variables employed here are thread-to-core mapping and DVFS. Typical strategy is to map the most power-intensive threads to less power-hungry cores and then apply DVFS to the cores to ensure all threads complete at the same time whilst satisfying a power budget constraint.

1.2. Overview of Application-level Solution Methods

In the second category, solution methods optimize applications rather than the executing environment. The methods use application-level decision variables and predictive models for energy and performance of applications to solve the bi-objective optimization problem. The dominant decision variables include the number of threads, loop tile size, workload distribution, etc. Following the principle of energy proportionality, a dominant class of such solution methods aim to achieve op-

timal energy reduction by optimizing for performance alone. Definitive examples are scientific routines offered by vendor-specific software packages that are extensively optimized for performance. For example, Intel Math Kernel Library [17] provides extensively optimized multithreaded basic linear algebra subprograms (BLAS) and 1D, 2D, and 3D fast Fourier transform (FFT) routines for Intel processors. Open source packages such as OpenBLAS [18], FFTW [19], and ZZGemmOOC [20] offer the same interface functions but contain portable optimizations and may exhibit better average performance than a heavily optimized vendor package as demonstrated by Khaleghzadeh, Reddy, and Lastovetsky [21], Khokhriakov, Reddy, and Lastovetsky [22]. The optimized routines in these software packages allow employment of one key decision variable, which is the number of threads. A given workload is load-balanced between the threads. *In this work, we show that the optimal number of threads (and consequently load-balanced workload distribution) maximizing the performance does not necessarily minimize the energy consumption of multicore CPUs.*

State-of-the-art research works on application-level optimization methods (Lastovetsky and Reddy [5], Reddy and Lastovetsky [6], Reddy and Lastovetsky [7]) demonstrate that due to the aforementioned design complexities of modern multicore CPU platforms, the functional relationships between performance and workload size and between dynamic energy and workload size for real-life data-parallel applications have complex (non-linear and non-smooth) properties and show that workload distribution has become an important decision variable that can no longer be ignored. The research works [5, 6, 7] propose model-based data partitioning methods that take as input discrete performance and dynamic energy functions without any assumptions on their shapes. The functions accurately and realistically account for resource contention and NUMA inherent in modern multicore CPU platforms. Using a simulation of the execution of a data-parallel matrix multiplication application based on OpenBLAS DGEMM on a homogeneous cluster of multicore CPUs, Lastovetsky and Reddy [5] show that optimizing for performance alone results in average and maximum dynamic energy reductions of 24% and 68%, but optimizing for dynamic energy alone results in performance degradations of 95% and 100%. For a 2D fast Fourier transform application based on FFTW, the average and maximum dynamic energy reductions are 29% and 55% and the average and maximum performance degradations are both 100%.

Reddy and Lastovetsky [6] proposes a solution method to solve bi-objective optimization problem of an application for energy and performance on homogeneous clusters of modern multicore CPUs. This method is shown to determine a diverse set of Pareto-optimal solutions whereas existing solution methods give only one solution when the problem size and number of processors are fixed. The methods [5, 6, 7] target homogeneous high performance computing (HPC) platforms. Khaleghzadeh, Fahad, Shahid, Reddy, and Lastovetsky [23] propose a solution method solving the bi-objective optimization problem on heterogeneous processors. The authors prove that for an arbitrary number of processors with linear execution time and dynamic energy functions, the **Pareto front is piece-wise linear and contains** an infinite number of solutions out of which one solution is load balanced while the rest are load imbalanced. A data partitioning algorithm is presented that takes as an input discrete performance and dynamic energy functions with no shape assumptions and outputs Pareto front of solutions (workload distributions).

The research works (Lastovetsky and Reddy [5], Reddy and Lastovetsky [6], Reddy and Lastovetsky [7], Khaleghzadeh, Fahad, Shahid, Reddy, and Lastovetsky [23]) are theoretical demonstrating energy and performance improvements based on simulations of clusters of homogeneous and heterogeneous nodes. Khokhriakov, Reddy, and Lastovetsky [22] present two novel optimization methods to improve the average performance of the FFT routines on modern multicore CPUs. The methods employ workload distribution as the decision variable and are based on parallel computing employing threadgroups. They utilize load imbalancing data partitioning technique that determines optimal workload distributions between the threadgroups, which may not load-balance the application in terms of execution time. The inputs to the methods are discrete 3D functions of performance against problem size of the threadgroups, and can be employed as nodal optimization techniques to construct a 2D FFT routine highly optimized for a dedicated target multicore CPU. The authors employ the methods to demonstrate significant performance improvements over the basic FFTW and Intel MKL FFT 2D routines on a modern Intel Haswell multicore CPU consisting of thirty-six physical cores.

1.3. Motivation

The findings in (Lastovetsky and Reddy [5], Reddy and Lastovetsky [6], Reddy and Lastovetsky [7], Khokhriakov, Reddy, and Lastovetsky [22], Khaleghzadeh, Fahad, Shahid, Reddy, and Lastovetsky [23]) motivate us to study the influence of three-dimensional decision variable space on bi-objective op-

Table 1: Specifications of the four Intel multicore CPUs, {HCLServer1, HCLServer2, HCLServer3, HCLServer4}.

Technical Specifications	**HCLServer1 (S1)**	**HCLServer2 (S2)**	**HCLServer3 (S3)**	**HCLServer4 (S4)**
Processor	Intel Xeon Gold 6152	Intel Haswell E5-2670V3	Intel Xeon CPU E5-2699	Intel Xeon Platinum 8180
Core(s) per socket	22	12	18	28
Socket(s)	1	2	2	2
L1d cache, L1i cache	32 KB, 32 KB	32 KB, 32 KB	32 KB, 32 KB	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30720 KB	256 KB, 30976 KB	256 KB, 46080 KB	1024 KB, 39424 KB
Total main memory	96 GB	64 GB	256 GB	187 GB
Power meter	WattsUp Pro	WattsUp Pro	-	Yokogawa WT310

timization of applications for energy and performance on multicore CPUs. The three decision variables are: a). The number of identical multithreaded kernels (threadgroups) involved in the parallel execution of an application; b). The number of threads in each threadgroup; and c). The workload distribution between the threadgroups. We focus exclusively on the first two decision variables in this work. The number of possible workload distributions increases exponentially with increasing number of threadgroups employed in the execution of a data-parallel application and it would require employment of threadgroup-specific energy and performance models to reduce the complexity. It is a subject of our future work.

We propose and study a novel application-level method for bi-objective optimization of multithreaded data-parallel applications for energy and performance. The method introduces a new direction towards energy-optimal design of multithreaded data-parallel applications. It employs two optimization objectives, dynamic energy and performance, and uses two decision variables, the number of identical multithreaded kernels (threadgroups) executing the application in parallel and the number of threads in each threadgroup. The workload distribution is not a decision variable. It is fixed so that a given workload is always partitioned equally between the threadgroups. The method allows full reuse of highly optimized scientific codes and does not require any changes to hardware or OS. The first step of the method includes writing a data-parallel version of the base kernel that can be executed using a variable number of threadgroups in parallel and solving the same problem as the base kernel, which employs one threadgroup.

We demonstrate our method using four multithreaded applications: a) 2D-FFT using FFTW 3.3.7; b) 2D-FFT using Intel MKL FFT; c) Dense matrix multiplication using OpenBLAS; and d) Dense matrix multiplication using Intel MKL FFT.

Four different modern Intel multicore CPUs are used in the experiments: a) A single-socket Intel Skylake consisting of 22 physical cores; b) A dual-socket Intel Haswell consisting of 24 physical cores; c) A dual-socket Intel Haswell consisting of 36 physical cores; and d) A dual-socket Intel Skylake consisting of 56 cores. Specifications of the experimental servers S1, S2, S3, and S4 equipped with these CPUs are given in Table 1. Servers S1, S2, and S4 are equipped with power meters and fully instrumented for system-level energy measurements. Server S3 is not equipped with a power meter and therefore is not employed in the experiments for single-objective optimization for energy and bi-objective optimization for energy and performance.

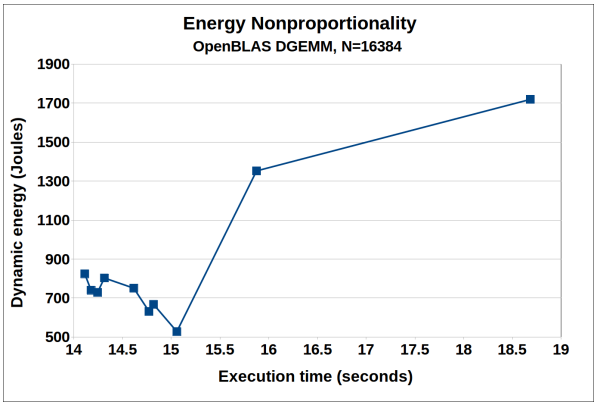


Figure 1: Energy nonproportionality on S2 found by our method for OpenBLAS DGEMM application solving workload size, N=16384. Energy and execution time do not exhibit a monotonically increasing relationship.

Figure 1 illustrates the energy nonproportionality on S2 found by our method for OpenBLAS DGEMM application solving workload size, N=16384. Data points in the graph represent different configurations of the multithreaded application solving exactly the same problem. *Energy proportionality is signified by a monotonically increasing relationship between energy and execution time. This is clearly not the case for the relationship shown in the figure.*

The average and maximum performance improvements for performance optimization on a single-socket multicore CPU (S1) are (7%, 26.3%), (5%, 6.5%) and (27%, 69%) for the OpenBLAS DGEMM, Intel MKL

DGEMM and Intel MKL FFT applications against their best single threadgroup configurations. Along with performance optimization, the energy improvements for OpenBLAS DGEMM and Intel MKL DGEMM are (7.9%, 30%) and (35.7%, 67%) against their best single threadgroup configurations.

At the same time, the optimization for performance alone results in average and maximum increases in dynamic energy consumption of (22.5%, 67%) and (87%, 89%) for the Intel MKL DGEMM and Intel MKL FFT applications in comparison with their energy-optimal configurations. The optimization for dynamic energy alone results in average and maximum performance degradations of (27%, 39%) and (19.7%, 38.2%) in comparison with their performance-optimal configurations. The average and the maximum number of Pareto-optimal solutions for Intel MKL DGEMM and Intel MKL FFT are (2.3, 3) and (2.6, 3).

On the 24-core dual-socket CPU (S2), the average and maximum performance improvements of (16%, 20%) and (8%, 21%) for the OpenBLAS DGEMM and Intel MKL DGEMM applications against their best single-threadgroup configurations. Even higher average and maximum performance improvements of (30%, 50%) are achieved for the FFTW application on the 56-core dual-socket CPU (S4). Again, the improvements are measured against the original single-threadgroup basic routine employing optimal number of threads.

At the same time, we find that optimization of the OpenBLAS DGEMM and Intel MKL DGEMM applications on S2 for performance only, results in average and maximum increases in dynamic energy consumption of (15%, 35%) and (7.1%, 49%) in comparison with their energy-optimal configurations, and optimization of the Intel MKL FFT and FFTW applications on S4 for performance alone results in average and maximum increases in dynamic energy consumption of (7%, 25%) and (15%, 57%).

On S2, the optimization of the OpenBLAS DGEMM and Intel MKL DGEMM applications for energy only, results in average and maximum performance degradations of (2.5%, 6%) and (3.7%, 11%). On S4, the average and maximum performance degradations are (20%, 33%) and (31%, 49%) for the Intel MKL FFT and FFTW applications. The performance degradations are over the performance-optimal configuration.

By solving the bi-objective optimization problem on three servers $\{$S1,S2,S4$\}$, the average and the maximum number of Pareto-optimal solutions determined by out method are (2.7, 3), (3,11), (2.4, 5) and (1.8, 4) for Intel MKL FFT, FFTW, OpenBLAS DGEMM and Intel MKL DGEMM applications.

1.4. Qualitative Dynamic Energy Model to Explain Energy Nonproportionality

There are three mainstream approaches to providing measurement of energy consumption during an application execution: (a) System-level physical measurements us ing external power meters, (b) Measurements using on-chip power sensors, and (c) Energy predictive models. Fahad, Shahid, Reddy, and Lastovetsky [24] present a comprehensive comparative study of the approaches. A vast majority of software energy predictive models employ performance monitoring counters (PMCs) as predictor variables. PMCs are special-purpose registers provided in modern microprocessors to store the counts of software and hardware activities. To explain the discovered energy nonproportionality, we propose a qualitative dynamic energy model based on linear regression and employing PMCs as variables.

There are few research works that highlight the poor accuracy of state-of-the-art linear regression models employing PMCs as predictor variables and demonstrate how the accuracy can be improved using model variables that are deterministic and reproducible and that are selected based on physical significance originating from fundamental physical laws such as conservation of energy of computing. Economou, Rivoire, Kozyrakis, and Ranganathan [25] highlight the fundamental limitation, which is the inability to obtain all the PMCs simultaneously or in one application run. They also mention the lack of PMCs to model the energy consumption of disk I/O and network I/O. McCullough, Agarwal, Chandrasekhar, Kuppuswamy, Snoeren, and Gupta [26] evaluate the competence of predictive power models for modern node architectures and show that linear regression models show prediction errors as high as 150%. They suggest that direct physical measurement of power consumption should be the preferred approach to tackle the inherent complexities posed by modern node architectures. O'Brien, Pietri, Reddy, Lastovetsky, and Sakellariou [27] survey the state-of-the-art energy predictive models in HPC and present a case study demonstrating the ineffectiveness of the dominant PMC-based modeling approach for accurate energy predictions. Shahid, Fahad, Reddy, and Lastovetsky [28] propose a novel property of PMCs called *additivity*, which can be used to determine the subset of PMCs that can potentially be used for reliable energy predictive modeling. It is based on the experimental observation that the energy consumption of a serial execution of two applications is the sum of energy consumptions observed for the individual execution of each application. A linear predictive energy model is consistent if

and only if its predictor variables are *additive* in the sense that the vector of predictor variables for a serial execution of two applications is the sum of vectors for the individual execution of each application. The use of *non-additive* PMCs in a model impairs its prediction accuracy. Shahid, Fahad, Reddy, and Lastovetsky [29] demonstrate that correlation with dynamic energy consumption alone is not sufficient to provide good average prediction accuracy of models but should be combined with methods such as *additivity* that take into account the physical significance of the predictor variables originating from fundamental laws such as energy conservation of computing.

Our proposed dynamic energy model employs PMCs that are based primarily on the property of additivity followed by high positive correlation with dynamic energy consumption. It reveals the cause behind the energy nonproportionality and demonstrates the same trend as the measured dynamic energy using system-level power measurements based on power meters, which is considered to be the ground truth [24].

1.5. Summary of Contributions

The main contributions in this work are the following:

- We show that energy proportionality does not hold true for multicore CPUs thereby affording an opportunity for bi-objective optimization for energy and performance.
- We propose and study a novel application-level method for bi-objective optimization of multithreaded data-parallel applications for energy and performance. The method introduces a new direction towards energy-optimal design of multithreaded data-parallel applications. Using four highly optimized data-parallel applications, the proposed method is shown to determine good numbers of Pareto-optimal solutions (number of threadgroups, number of threads per threadgroup) of the applications providing the programmers better trade-offs between energy and performance.
- To explain the discovered energy nonproportionality, a qualitative dynamic energy model based on linear regression and employing performance monitoring counters (PMCs) as variables is proposed. The model shows that the energy nonproportionality on our experimental platforms for the two data-parallel applications is due to disproportionately high energy consumption by the data translation lookaside buffer (dTLB) activity.

The rest of the paper is organized as follows. Section 2 presents terminology related to energy consumption. Section 3 presents the related work. Section 4 describes our solution method. Section 5 describes the first step of our solution method for two data-parallel applications, 2D fast Fourier transform and dense matrix multiplication. Section 6 contains the experimental results. Section 6.4 presents our dynamic energy model employing PMCs as variables to explain the cause behind the energy nonproportionality on our experimental platforms. Section 7 concludes the paper.

2. Terminology

There are two types of energy consumptions, static energy, and dynamic energy. The total energy consumption is the sum of dynamic and static energy consumptions. The static energy consumption is calculated by multiplying the idle power of the platform (without application execution) with the execution time of the application. The dynamic energy consumption is calculated by subtracting this static energy consumption from the total energy consumed by the platform during the application execution. That is, if P_S is the static power consumption of the platform, E_T is the total energy consumption of the platform during the execution of an application, which takes T_E seconds, then the dynamic energy E_D can be calculated as,

$$E_D = E_T - (P_S \times T_E) \tag{1}$$

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

1. Although static energy consumption is a major concern in embedded systems, it is becoming less compared to the dynamic energy consumption due to advancements in hardware architecture design in HPC systems.
2. We target 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 minimizes the dynamic energy consumption. 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 16500J and

18000J. It would report the prediction error to be 8.3%. If it is known that the static energy consumption of the system is 9000J, then the actual prediction error (based on dynamic energy consumptions 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 10000J. 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 consumptions only) for the models would be 16.6% and 8.3%. Therefore, the second model M_B should be considered more accurate than the first.

A background on multi-objective optimization is presented in the supplemental [30]. In multi-objective optimization, there is no natural ordering in the objective space because it is only partially ordered. Therefore we must treat the concept of optimality differently from single-objective optimization problem. The generally used concept is *Pareto-optimality*. In this work, we consider bi-objective optimization where dynamic energy and performance are the objectives.

3. Related Work

We divide our literature review into the following categories:

- Software based energy predictive models for multicore CPUs;
- Surveys on energy efficiency in computing;
- Single-objective optimization methods for energy and performance;
- Multi-objective system-level and application-level optimization methods with energy or performance or both as important objectives.

3.1. Energy Predictive Models of Computation

Software based energy predictive models emerged as a predominant approach to predict the energy consumed by a given platform during the execution of an application. A vast majority of such models are linear and use performance monitoring counters (PMCs) to predict the energy consumption.

Bellosa [31] proposes an energy model based on performance monitoring counters such as integer operations, floating-point operations, memory requests due to cache misses, etc. that they believed to strongly correlate with energy consumption. Icsi and Martonosi [32] employ access rates of the components determined using performance monitoring counters (PMCs) to model component-level power consumption. Li and John [33] employ instructions per cycle (IPC) as predictor variable to model power consumption of the operating system (OS). Lee and Brooks [34] propose regression models using performance events to predict power. A linear model that is based on the utilization of CPU, disk, and network is presented by Heath, Diniz, Horizonte, Carrera, and Bianchini [35]. A more complex power model (Mantis) studied by Economou, Rivoire, Kozyrakis, and Ranganathan [25] employs utilization metrics of CPU, disk, and network components and hardware performance counters for memory as predictor variables.

Fan, Weber, and Barroso [36] propose a simple linear model that correlates power consumption of a single-core processor with its utilization. Bertran, Gonzalez, Martorell, Navaroo, and Ayguade [37] present a power model that provides per-component power breakdown of a multicore CPU. Basmadjian, Ali, Niedermeier, de Meer, and Giuliani [38] model power consumption of a server as sum of power consumption of its components, the processor (CPU), memory (RAM), fans and disk (HDD). Bircher and John [39] present an power predictive model based on PMCs that capture interdependence between subsystems such as CPU, disk, GPU and so forth. Dargie [40] use the statistics of CPU utilization (instead of PMCs) to model the relationship between the power consumption of multicore processor and workload quantitatively. They demonstrate that the relationship is quadratic for single-core processor and linear for multicore processors.

Energy predictive models predominantly employ performance monitoring counters (PMCs) as variables. Haj-Yihia, Yasin, Asher, and Mendelson [41] propose a linear power predictive model for Intel Skylake based CPUs based on selected PMCs that are highly positively correlated with power consumption. Mair, Hyang, and Eyers [42] present Manila, which is a power model based on PMC space generated as densely populated points gathered via a large number of synthetic applications. Lastovetsky and Reddy [5] present an application-level energy model where the dynamic energy consumption of a processor is represented by a discrete function of problem size, which is shown to be

highly non-linear for data-parallel applications on modern multicore CPUs.

Unlike the energy predictive models surveyed in this section, we propose a qualitative dynamic energy model employing PMCs as variables that explains the cause of energy nonproportionality in our experimental platforms.

3.2. Surveys on Energy Efficiency in Computing

Kaxiras and Martonosi [43] survey the most important architectural techniques that have been proposed to reduce both static and dynamic power consumptions in processors and memory hierarchies. Benedict [44] present a survey of various energy measurement methodologies for HPC, Grid, and Cloud applications. Mobius, Dargie, and Schill [45] present a survey of power consumption models for single-core and multicore processors, virtual machines, and servers. They conclude that regression-based approaches dominate and that one prominent shortcoming of the these models is that they use static instead of variable workloads for training the models.

Inacio and Dantes [46] present a literature survey of works using workload characterization for performance and energy efficiency improvement in HPC, cloud, and big data environments. Orgerie, Assuncao, and Lefevre [47] survey techniques for improving the energy efficiency of computing and networking resources in large-scale distributed systems. Tan, Kothapalli, Chen, Hussaini, Bissiri, Chen et al. [48] survey the research on saving power and energy for HPC linear algebra applications. They construct a linear model of a HPC system as a summation of power consumptions of all the nodes in the system. The power consumption of a node is modelled as the sum of all the major components (CPU, GPU, RAM) of a node.

Mittal and Vettel [49] present a survey of research works analysing and improving energy efficiency of GPUs. In this survey, they also present works that compare the energy efficiency of GPUs with other computing systems such as CPUs, Cell processor, FPGA etc. Dayarathna, Wen, and Fan [50] present an in-depth survey on data center power modelling. They organize the power models based on two classifications: a). hardware-centric, and b). software-centric.

O'Brien, Pietri, Reddy, Lastovetsky, and Sakellariou [27] survey the state-of-the-art energy predictive models in HPC and present a case study demonstrating the ineffectiveness of the dominant PMC-based modeling approach for accurate energy predictions. In the case study, they use 35 carefully selected PMCs (out of a total of 390 available in the platform) in their linear regression model for predicting dynamic energy consumption. The authors show that the linear regression models give prediction errors as high as 100%.

3.3. Energy Optimization Methods

There are three important categories dealing with energy optimization on multicore CPU platforms. The software category contains research works that propose shared resource partitioners. The two hardware categories concern research works that employ Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) and thermal management. Zhuravlev, Saez, Blagodurov, Fedorova, and Prieto [51] survey the prominent works in all the three categories.

Wang, Mishra, and Ranka [10], Cheng, Huang, Huang, and Knoll [12] propose dynamic reconfiguration of private caches and partitioning of shared caches (last level cache, for example) to reduce the energy consumption without hurting performance.

DVFS and DPM allow changing the frequencies of the cores and to lower their power states when they are idle. Considering the enormity of literature in this category, we will cover only works that take into account resource contention and thread-to-core mapping while employing DVFS. Kadayif, Kandamir, and Kolcu [52] exploit the heterogeneous nature of workloads executed by different processors to set their frequencies so as to reduce energy without impacting performance. Kondo, Sasaki, and Nakamura [53], Watanabe, Kondo, Nakamura, and Nanya [54] employ DVFS to reduce resource contention and energy consumption.

The main goal of thermal management algorithms proposed by Yang, Zhou, Chrobak, Zhang, and Jin [13], Ayoub and Rosing [14] is to find thread-to-core mappings (or even thread migration) to remove drastic variations in temperatures or thermal hotspots in the chip and at the same time reduce the energy consumption without impacting the performance. They employ as inputs thermal models that are built using temperature measurements provided by on-chip sensors. The algorithms are chiefly employed at the OS level.

Asymmetry-aware schedulers have been proposed for energy optimization on asymmetric multicore systems, which feature a mix of fast and slow cores, high-power and low-power cores but that expose the same instruction-set architecture (ISA). Fedorova, Saez, Shelepov, and Prieto [55] propose a system-level scheduler that assigns sequential phases of an application to fast cores and parallel phases to slow cores to maximize the energy efficiency. Herbert, Garg, and Marculescu [56] employ DVFS to exploit the core-to-core variations

from fabrication in power and performance to improve the energy efficiency of the multicore platform.

In this work, we propose the first application-level method for energy optimization that finds the optimal configuration of the application minimizing the dynamic energy consumption during the execution of the application rather than the optimal configuration of the multicore CPU operating environment.

3.4. Performance Optimization Methods

There are three dominant approaches in this category. First category contains research works [57, 58] that have proposed contention-aware thread-level schedulers that try to minimize performance losses due to contention for on-chip shared resources.

The second category includes DRAM controller schedulers that aim to efficiently utilize the shared resource, which is the DRAM memory system, and last level cache partitioning that physically partition the shared resources to minimize contention. DRAM controller schedulers proposed by Ebrahimi, Miftakhutdinov, Fallin, Lee, Joao, Mutlu et al. [59], Jeong, Yoon, Sunwoo, Sullivan, Lee, and Erez [60] improve the throughput by ordering threads and prioritizing their memory requests through DRAM controllers. Last level cache partitioners proposed by Li, Lu, Ding, Zhang, Zhang, and Sadayappan [61], Tam, Azimi, Soares, and Stumm [62] explicitly partition the cache when the default cache replacement policies (such as least-recently-used (LRU)) do not result in efficient execution of applications. These partitioners, however, must be used in conjunction with schedulers that mitigate contention for memory controllers and on-chip interconnects.

The final category includes research works (Tang, Mars, Vachharajani, Hundt, and Soffa [63], Mars, Tang, Hundt, Skadron, and Soffa [64]) that focus on thread-level schedulers that exploit data sharing between the threads to co-schedule them. A key building block in the schedulers are performance models based on PMCs that can predict performance loss due to co-scheduling or migrating threads between cores.

3.5. Multi-Objective Optimization

In this section, we survey the state-of-the-art multi-objective optimization methods.

3.5.1. System-level Methods

System-level multi-objective optimization methods aim to optimize several objectives of the system or the environment (for example: clouds, data centers, etc) where the applications are executed.

Rong, Feng, Feng, and Cameron [65] present a runtime system (CPU MISER) based on DVFS that provides energy savings with minimal performance degradation by using a performance model. Huang and Feng [66] propose an eco-friendly daemon that employs workload characterization as a guide to DVFS to reduce power and energy consumption with little impact on application performance. Mezmaz, Melab, Kessaci, Lee, Talbi, Zomaya et al. [67] propose a parallel bi-objective genetic algorithm to maximize the performance and minimize the energy consumption in cloud computing infrastructures. The parameters used in their method are the computation cost of a task and the communication costs between two tasks. The decision variable is the supply voltage of the processor. Fard, Prodan, Barrionuevo, and Fahringer [68] present a four-objective case study comprising performance, economic cost, energy consumption, and reliability for optimization of scientific workflows in heterogeneous computing environments. The parameters are the computation speeds of the processors and the bandwidths of the communication links connecting a pair of processors. The decision variable is the task assignment or mapping. The energy consumption of computations is modeled as cube-root of clock frequency. Beloglazov, Abawajy, and Buyya [69] propose heuristics that consider twin objectives of energy efficiency and Quality of Service (QoS) for provisioning data center resources. The decision variables are the number of VMs and clock frequencies. The energy consumption is modeled as a linear function of CPU utilization. Kessaci, Melab, and Talbi [70] present a multi-objective genetic algorithm that minimizes the energy consumption, CO2 emissions and maximizes the generated profit of a cloud computing infrastructure. The parameters are the execution time of an application, the number of processors used in the execution of an application, and the deadline for completion of the application. The decision variable is the arrival rate.

Durillo, Nae, and Prodan [71] propose a multi-objective workflow scheduling algorithm that maximizes performance and minimizes energy consumption of applications executing in heterogeneous high-performance parallel and distributed computing systems. A machine is characterized using nine parameters (from technology(nm) to TDP). They study the impact of different decision variables: number of tasks, number of machines, DVFS levels, static energy, and types of tasks. The execution time and energy consumption are predicted using neural networks. Das, Kumar, Veeravalli, Bolchini, and Miele [72] propose task mapping to optimize for energy and reliability on multiprocessor systems-on-chip (MPSoCs) with performance as a con-

straint. Zhang and Chang [73] present a DVFS scheduler that makes sure the multiple user applications executing on multicores in clouds meet their SLA requirement, which is the specific allowed performance loss.

Kolodziej, Khan, Wang, and Zomaya [74] propose multi-objective genetic algorithms that aim to maximize performance and energy consumption of applications executing in green grid clusters and clouds. The performance is modeled using computation speed of a processor. The decision variable is the DVFS level. Vaibhav and Masha [75] present a runtime system that performs both processor and DRAM frequency scaling and demonstrate total energy savings with minimal performance loss. Yuichi, Patki, Inoue, Aoyagi, Rountree, Schulz et al. [76], Neha, Muller, and Rountree [77] consider the fluctuations in performance arising from manufacturing and thermal variations and propose approaches that take into account these variations to assign jobs to machines, which have a specified power budget. In our work, we consider the variations in performance caused by severe resource contention and NUMA inherent in modern multicore platforms during the execution of highly multithreaded scientific data-parallel applications. Sheikh, Ahmad, and Fan [78] propose task scheduler employing evolutionary algorithms to optimize applications on multicore CPU platforms for performance, energy, and temperature. Abdi, Girault, and Zarandi [79] propose multi-criteria optimization where they minimize the execution time under three constraints, the reliability, the power consumption, and the peak temperature. DVFS is a key decision variable in all of these research works.

3.5.2. Application-level Methods

Subramaniam and Feng [80] use multi-variable regression to study the performance-energy trade-offs of the high-performance LINPACK (HPL) benchmark. They study performance-energy trade-offs using the decision variables, number of threads and number of processes. Marszalkowski, Drozdowski, and Marszalkowski [**?**] analyze the impact of memory hierarchies on time-energy trade-off in parallel computations, which are represented as divisible loads. They represent execution time and energy by two linear functions on problem size, one for in-core computations and the other for out-of-core computations.

Lastovetsky and Reddy [5], Reddy and Lastovetsky [7] propose data partitioning algorithms that solve single-objective optimization problems of data-parallel applications for performance or energy on homogeneous clusters of multicore CPUs. They take as an input, discrete performance and dynamic energy functions with no shape assumptions and that accurately and realistically account for resource contention and NUMA inherent in modern multicore CPU platforms. Reddy and Lastovetsky [6] propose a solution method to solve bi-objective optimization problem of an application for performance and energy on homogeneous clusters of modern multicore CPUs. They demonstrate that the method gives a diverse set of Pareto-optimal solutions and that it can be combined with DVFS-based multi-objective optimization methods to give a better set of (Pareto-optimal) solutions. The methods target homogeneous HPC platforms. Chakraborti, Parthasarathy, and Stewart [81] consider the effect of heterogeneous workload distribution on bi-objective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters. The performance is represented by a linear function of problem size and the total energy is predicted using historical data tables. Khaleghzadeh, Fahad, Shahid, Reddy, and Lastovetsky [23] propose a solution method solving the bi-objective optimization problem on heterogeneous processors and comprising of two principal components. The first component is a data partitioning algorithm that takes as an input discrete performance and dynamic energy functions with no shape assumptions. The second component is a novel methodology employed to build the discrete dynamic energy profiles of individual computing devices, which are input to the algorithm.

None of the state-of-the-art methods optimise for a single multicore CPU processor. They all target either homogeneous or heterogeneous platforms with multiple multicore CPUs. Our proposed method is the first application-level solution method for bi-objective optimization of a data-parallel application for energy and performance that targets a single multicore CPU.

4. Solution Method Solving Bi-objective Optimization Problem on a Single Multicore CPU

In this section, we describe our solution method, BOPPETG, for solving the bi-objective optimization problem of a multithreaded data-parallel application on multicore CPUs for performance and energy (BOPPE). The method uses two decision variables, the number of identical multithreaded kernels (threadgroups) and the number of threads in each threadgroup. A given workload is always partitioned equally between the threadgroups.

The bi-objective optimization problem (BOPPE) can be formulated as follows: Given a multithreaded data-parallel application of workload size n and a multicore CPU of l cores, the problem is to find the Pareto-optimal

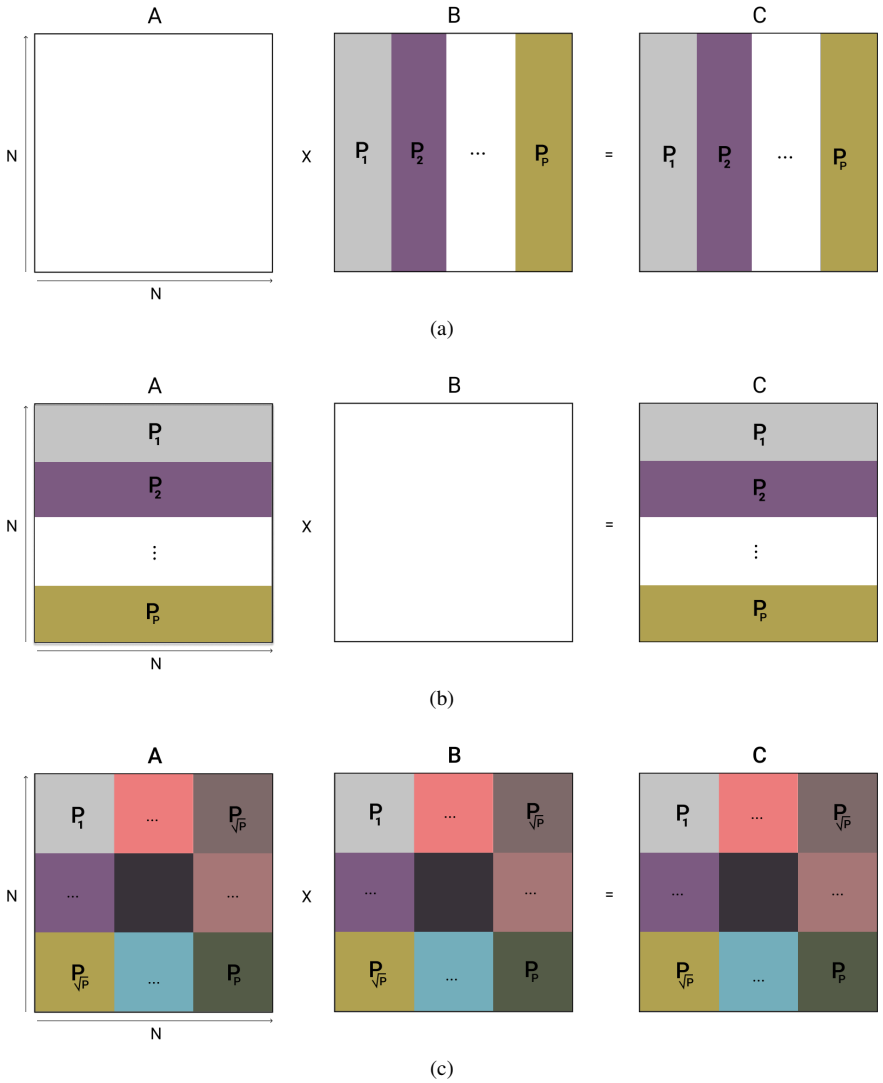


Figure 2: Decomposition of the matrices, A, B, and C, in our parallel matrix multiplication. (a). PMMTG-V: Matrices B and C are vertically partitioned among the threadgroups. (b). PMMTG-H: Matrices A and C are horizontally partitioned among the threadgroups. (c). PMMTG-S: The p threadgroups are arranged in a square grid of size $\sqrt{p} \times \sqrt{p}$. All the matrices are partitioned into squares among the threadgroups.

solutions optimizing execution time and dynamic energy consumption during the parallel execution of the workload. Each solution is an application configuration given by (threadgroups, threads per group).

The inputs to the solution method are the workload size of the multi-threaded data-parallel application, n; the number of cores in the multicore CPU, l; the multithreaded base kernel, *mtkernel*; the base power of the multicore CPU platform, P_b. The outputs are the Pareto front of objective solutions, $\mathcal{P}_{opt}$, and the optimal application configurations corresponding to these solutions, C_{opt}. Each Pareto-optimal solution of objectives o is represented by the pair, (s_o, e_o), where s_o is the execution time and e_o is the dynamic energy. Associated with this solution is an array of application configurations, $\mathcal{A}(g_o, t_o)$, containing decision variable pairs, (g_o, t_o), where g_o represents the number of threadgroups each containing t_o threads.

The main steps of BOPPETG are as follows:

Step 1. Parallel implementation allowing (g,t) configuration: Design and implement a data-parallel version of the base kernel *mtkernel* and that can be executed using g identical multithreaded kernels in parallel. Each kernel is executed by a threadgroup containing t threads. The workload n is divided equally between the g threadgroups during the execution of the data-parallel

version. The data-parallel version should essentially allow its runtime configuration using number of threadgroups and number of threads per group with the workload equally partitioned between the threadgroups.

Step 2. Initialize g and t: All the runtime configurations, (g,t), where the product, $g \times t$, is less than or equal to the total number of cores (l) in the multicore platform are considered. $g \leftarrow 1, t \leftarrow 1$. Go to Step 3.

Step 3. Determine time and dynamic energy of the (g,t) configuration of the application: The data-parallel version composed in Step 1 is run using the (g,t) configuration. Its execution time and dynamic energy consumption are determined as follows: $s_o = t_f - t_i$, $e_o = e_f - P_b \times s_o$, where t_i and t_f are the starting and ending execution times and e_f is the total energy consumption during the execution of the application. Go to Step 4.

Step 4. Update Pareto front for (g,t): The solution (s_o, e_o) if Pareto-optimal is added to the Pareto front of objective solutions, $\{\mathcal{P}_{opt}\}$, and existing member solutions of the set that are inferior to it are removed. The optimal application configurations corresponding to the solution (s_o, e_o) are stored in C_{opt}. Go to Step 5.

Step 5. Test and Increment (g,t): If $t < l, t \leftarrow t + 1$, go to Step 3. Set $g \leftarrow g+1, t \leftarrow 1$. If $g \times t \leq l$, go to Step 3. Else return the Pareto front and optimal application configurations given by $\{\mathcal{P}_{opt}, C_{opt}\}$ and quit.

In the following section, we illustrate the first step of BOPPETG for dense matrix multiplication application. The application of BOPPETG for implementing the data-parallel version of 2D fast Fourier transform (PFFTTG) is presented in the supplemental [30]. We show in particular how BOPPETG can reuse highly optimized scientific kernels with careful design and development of parallel versions of the application.

5. Parallel Matrix Multiplication

We illustrate the first step of our solution method (BOPPETG) for implementing the data-parallel version of dense matrix multiplication (PMMTG).

The PMMTG application computes the matrix product ($C = \alpha \times A \times B + \beta \times C$) of two dense square matrices A and B of size $N \times N$. The application is executed using p threadgroups, $\{P_1, ..., P_p\}$. To simplify the exposition of the algorithms, we assume N to be divisible by p.

There are three parallel algorithmic variants of PMMTG. In PMMTG-V, the matrices B and C are partitioned vertically such that each threadgroup is assigned $\frac{N}{p}$ of the columns of B and C as shown in the Figure 2a. Each threadgroup P_i computes its vertical partition C_{P_i} using the matrix product, $C_{P_i} = \alpha \times A \times B_{P_i} + \beta \times C_{P_i}$. In PMMTG-H, the matrices A and C are partitioned horizontally such that each threadgroup is assigned $\frac{N}{p}$ of the rows of B and C as shown in the Figure 2b. Each threadgroup P_i computes its horizontal partition C_{P_i} using the matrix product, $C_{P_i} = \alpha \times A_{P_i} \times B + \beta \times C_{P_i}$. In PMMTG-S, the p threadgroups $\{P_1, ..., P_p\}$ are arranged in a square grid $Q_{st}, s \in [1, \sqrt{p}], t \in [1, \sqrt{p}]$.

```
void *dgemm(void *input)
{
  int i = *(int*)input;
  openblas_set_num_threads(t);
  goto_set_num_threads(t);
  omp_set_num_threads(t);
  if (i == 1)
  {
       cblas_dgemm(CblasRowMajor, CblasNoTrans,
            CblasNoTrans, N/p, N, N, alpha, A1, N,
            B, N, beta, C1, N);
  }
    ...
  if (i == p)
  {
       cblas_dgemm(CblasRowMajor, CblasNoTrans,
            CblasNoTrans, N/p, N, N, alpha, Ap, N,
            B, N, beta, Cp, N);
  }
}

int main() {
  int row;
#pragma omp parallel for num_threads(p*t)
  for (row = 0; row < N/p; row++) {
    memcpy(&A1[row*N], &A[row*N], N*sizeof(double));
    ...
    memcpy(&Ap[row*N], &A[(p-1)*N*(N/p)+row*N],
           N*sizeof(double));
    memcpy(&C1[row*N], &C[row*N], N*sizeof(double));
    ...
    memcpy(&Cp[row*N], &C[(p-1)*N*(N/p)+row*N],
           N*sizeof(double));
  }

  pthread_t t1, ..., tp;
  int i1 = 1, ..., ip = p;
  pthread_create(&t1, NULL, dgemm, &i1);
  ...
  pthread_create(&tp, NULL, dgemm, &ip);
  pthread_join(tp, NULL);
  ...
  pthread_join(t1, NULL);

#pragma omp parallel for num_threads(p*t)
  for (row = 0; row < N/p; row++)
  {
    memcpy(&A[row*N], &A1[row*N], N*sizeof(double));
    ...
    memcpy(&A[(p-1)*N*(N/p)+row*N], &Ap[row*N],
           N*sizeof(double));
    memcpy(&C[row*N], &C1[row*N], N*sizeof(double));
    ...
    memcpy(&C[(p-1)*N*(N/p)+row*N], &Cp[row*N],
           N*sizeof(double));
  }
}
```

Figure 3: OpenBLAS implementation of parallel matrix multiplication using horizontal partitioning of matrices A and C (PMMTG-H). The implementation employs p threadgroups of t threads each.

The matrices A, B, and C are partitioned into equal squares among the threadgroups as shown in the Figure 2c. In each matrix, each threadgroup $P_i(= Q_{st})$ is assigned a sub-matrix of size $\frac{N}{\sqrt{p}} \times \frac{N}{\sqrt{p}}$ and computes its square partition $C_{Q_{st}}$ using the matrix product, $C_{Q_{st}} = \alpha \times \sum_{k=1}^{\sqrt{p}} (A_{sk} \times B_{kt}) + \beta \times C_{Q_{st}}$. A_{sk} is the square block in matrix A located at (s, k). B_{kt} is the square block in matrix B located at (k, t).

5.1. Implementation of PMMTG-H Based on OpenBLAS DGEMM

We describe an OpenBLAS implementation of PMMTG-H (Figure 3) here. The implementations of the other PMMTG algorithms employing Intel MKL and OpenBLAS are described in the supplemental [30].

The inputs to an implementation are: a). Matrices A, B, and C of sizes $N \times N$; b). Constants α and β; c) The number of threadgroups, $\{P_1, \cdots, P_p\}$; d). The number of threads in each threadgroup represented by t. The output matrix, C, contains the matrix product.

The vertical partitions of A and C, $\{A_{P_i}, C_{P_i}\}$, $i \in [1, p]$, assigned to the threadgroups, $\{P_1, ..., P_p\}$, are initialized in Lines 24-34. Then p pthreads representing the p threadgroups are created, each a multithreaded OpenBLAS DGEMM kernel executing t OpenMP threads (Lines 36-43).The p threadgroups compute the matrix product (Lines 1-20). The result is gathered in the matrix C (Lines 45-56).

The implementations using Intel MKL differ from those using OpenBLAS. In Intel MKL, the matrix computation by a threadgroup is performed using an OpenMP parallel region with t threads whereas the same is done in OpenBLAS using a pthread.

6. Experimental Results and Discussion

In this section, we present our experimental results for matrix multiplication (PMMTG) and 2D fast Fourier transform (PFFTTG) employing our solution method.

To make sure the experimental results are reliable, we follow a statistical methodology described in the supplemental [30]. Briefly, for every data point in the functions, the automation software executes the application repeatedly until the sample mean lies in the 95% confidence interval and a precision of 0.025 (2.5%) has been achieved. For this purpose, Student's t-test is used assuming that the individual observations are independent and their population follows the normal distribution. The validity of these assumptions is verified by plotting the distributions of observations and using Pearson's Test. The speed/time/energy values shown in the graphical plots are the sample means.

Four multicore CPUs shown in the Table 1 and described earlier are used in the experiments. Three platforms {S1, S2, S4} have a power meter installed between their input power sockets and the wall A/C outlets. S1 and S2 are connected with a *Watts Up Pro* power meter; S4 is connected with a *Yokogawa WT310* power meter. S3 is not equipped with a power meter and therefore is not employed in the experiments for single-objective optimization for energy and bi-objective optimization for energy and performance.

The power meter provides the total power consumption of the server. It has a data cable connected to one USB port of the server. A script written in Perl collects the data from the power meter using the serial USB interface. The execution of the script is non-intrusive and consumes insignificant power. WattsUp Pro power meters are periodically calibrated using the ANSI C12.20 revenue-grade power meter, Yokogawa WT310. The maximum sampling speed of WattsUp Pro power meters is one sample every second. The accuracy specified in the data-sheets is ±3%. The minimum measurable power is 0.5 watts. The accuracy at 0.5 watts is ±0.3 watts. The accuracy of Yokogawa WT310 is 0.1% and the sampling rate is 100k samples per second.

HCLWattsUp API [82] is used to gather the readings from the power meter to determine the dynamic energy consumption during the execution of PMMTG and PFFTTG applications. HCLWattsUp has no extra overhead and therefore does not influence the energy consumption of the application execution. The API is described in the supplemental [30].

Fans are significant contributors 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%

To rule out the contribution of fans in dynamic energy consumption, we set the fans at full speed before executing the applications. When set at full speed, the fans run

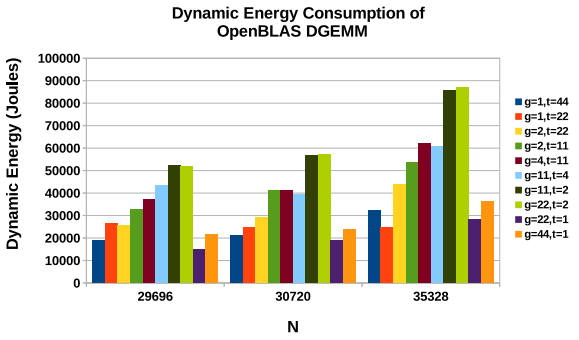


Figure 4: Dynamic energy consumption for PMMTG employing OpenBLAS DGEMM for different (g,t) configurations on S1. g is the number of threadgroups and t is the number of threads per threadgroup.

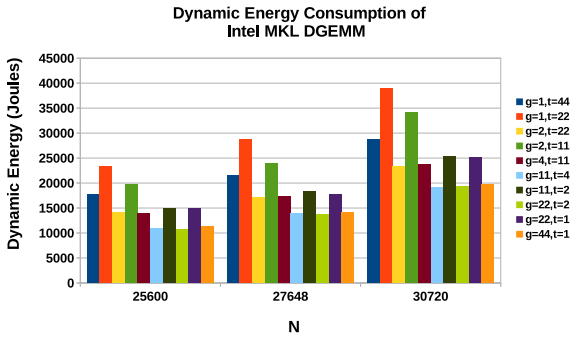


Figure 5: Dynamic energy consumption for PMMTG employing Intel MKL DGEMM for different (g,t) configurations on S1.

constantly at ~13400 rpm until they are set to a different speed level. In this way, energy consumption due to fans is included only in the static power consumption of the platform. The temperature of our platform and speeds of the fans (with *Full* setting) is monitored with the help of Intelligent Platform Management Interface (IPMI) sensors, both with and without the application run. An insignificant difference in the speeds of fans is found in both the scenarios.

6.1. Parallel Matrix-Matrix Multiplication

6.1.1. Energy Optimization on a Single Socket Multicore CPU

Fiqure 4 shows the dynamic energy consumptions for PMMTG using OpenBLAS DGEMM of different threadgroup combinations on a single-socket CPU (S1). The base version corresponds to application configuration employing one threadgroup with optimal number of threads, which is 44 threads. The best combination for sizes N=29696 and N=30720 is (g,t)=(22,1) where g is the number of threadgroups and t is the number of threads per threadgroup. It outperforms the base combination by 20%. The best combination for N = 35328 is (g,t)=(1,22), which outperforms the base combination by 23%. Furthermore, the average improvement (or energy savings) over the base combination for 41 tested workload sizes in the range, $5120 \leq N \leq 35000$, is 8%.

Fiqure 5 shows the dynamic energy consumptions for PMMTG using Intel MKL DGEMM. There are three best combinations for each problem size, (g,t)={(11,4),(22,2),(44,1)}. They outperform the base combination by 35%. Furthermore, the average improvement over the base combination for 21 tested workload sizes in the range, $5120 \leq N \leq 35000$, is 35.7%.

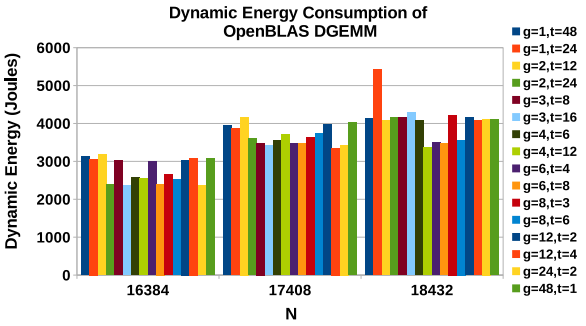


Figure 6: Dynamic energy consumption of PMMTG employing OpenBLAS DGEMM for different (g,t) configurations on S2.

6.1.2. Energy Optimization on a Dual-socket Multicore CPU

Figure 6 show the results for PMMTG based on OpenBLAS DGEMM on S2 with three different workload sizes. There are four best combinations minimizing the dynamic energy consumption for workload size 16384, (g,t)={(2,24),(3,16),(6,8),(24,2)}. The energy savings for these combinations compared with the best base combination, (g,t)=(1,24), is around 21%. For the workload sizes 17408 and 18432, the best combinations are (12,4) and (4,12). The energy savings in comparison with the best base combination, (g,t)=(1,24), for 17408 and (g,t)=(1,44) for 18432, are 15% and 18%. Furthermore, the average improvement over the best base combination for 19 tested workload sizes in the range, $5120 \leq N \leq 35000$, is 10%.

Figure 7 show the results for PMMTG based on Intel MKL DGEMM on S2. The best combination minimizing the dynamic energy consumption for workload size 28672 involves 12 threadgroups with 2 threads each. The energy savings for this combination compared with the best base combination, (1,24), is 10.5%. For the

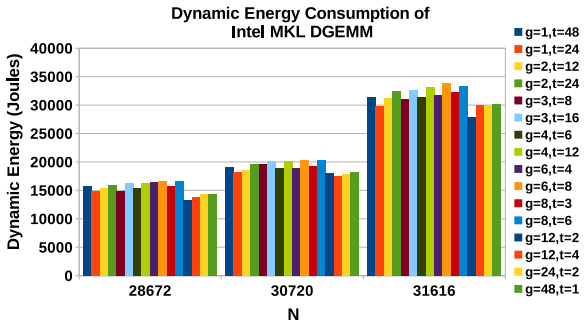


Figure 7: Dynamic energy consumption of PMMTG employing Intel MKL DGEMM for different (g,t) configurations on S2.

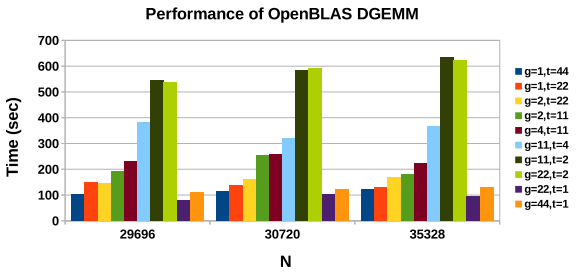


Figure 8: Performance of PMMTG application employing OpenBLAS DGEMM for different (g,t) configurations on S1.

workload sizes 30720 and 31616, the best combinations are (12,4) and (12,2). The energy savings in comparison with the best base combination are 4% and 7%. Furthermore, the average improvement over the best base combination for 19 tested workload sizes in the range, $5120 \leq N \leq 35000$, is 13%.

6.1.3. *Performance Optimization on a Single Socket Multicore CPU*

Fiqure 8 shows the execution times of PMMTG using OpenBLAS DGEMM for different threadgroup combinations on a single-socket CPU (S1). The base version corresponds to the application configuration employing one threadgroup with optimal number of threads, which is 44 threads. The best combination is (g,t)=(22,1) for all the three workload sizes. It outperforms the base combination by 20% for N=29696 and N=35328, and about 11% for N=30720. Furthermore, the average performance improvement over the base combination for 41 tested workload sizes in the range, $5120 \leq N \leq 36000$, is 7%. The starting problem size of 5120 is chosen to ensure that the workload size exceeds the last level cache.

Fiqure 9 shows the execution times of PMMTG using

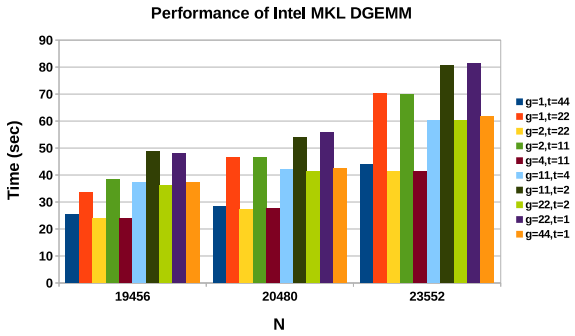


Figure 9: Performance of PMMTG application employing Intel MKL DGEMM for different (g,t) configurations on S1.

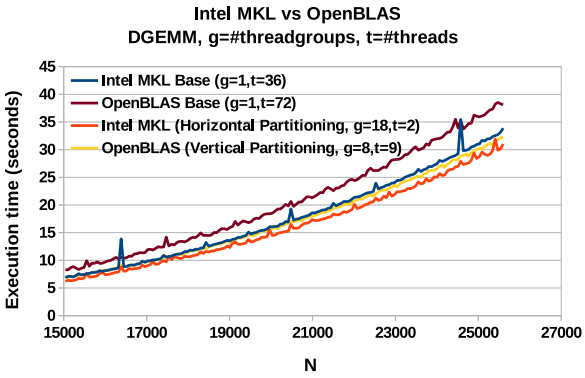


Figure 10: Comparision between the base and best versions for Intel MKL DGEMM and OpenBLAS DGEMM on S3.

Intel MKL DGEMM. The best combinations (g,t) are $\{(4,11),(2,22)\}$ for all the three workload sizes. They outperform the base combination by 6%. The average performance improvement over the base combination for 21 tested workload sizes in the range, $5120 \leq N \leq 36000$, is 5%.

6.1.4. *Performance Optimization on a Dual-socket Multicore CPU*

Figure 10 shows the comparision between base and best combinations for OpenBLAS DGEMM and Intel MKL DGEMM on S3. The base version corresponds to application configuration employing one threadgroup with optimal number of threads.

Unlike the base version, the best combinations for OpenBLAS DGEMM and Intel MKL DGEMM do not have any performance variations (drops). The best combination for Intel MKL DGEMM is 18 threadgroups with 2 threads each. It outperforms the base version by 8% on the average and the next best combination, 12

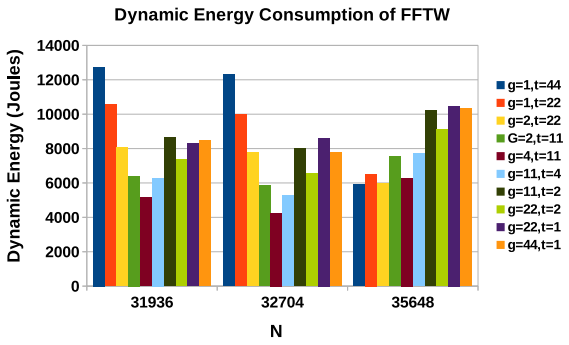


Figure 11: Dynamic energy consumption of PFFTTG employing FFTW for different (g,t) configurations on S1.

threadgroups with 2 threads each, by 2.5%. Our solution method removed noticeable drops in performance for workload sizes 16384, 20480, and 24576, with performance improvements of 36.5%, 14.5% and 21.5%.

6.2. *Parallel 2D Fast Fourier Transform*

In this section, we use 2D fast Fourier transform routines from two packages, FFTW-3.3.7 and Intel MKL. The packages are installed with multithreading, SSE/SSE2, AVX2, and FMA (fused multiply-add) optimizations enabled. For Intel MKL FFT, no special environment variables are used. Three planner flags, {FFTW_ESTIMATE, FFTW_MEASURE, FFTW_PATIENT} were tested. The execution times for the flags {FFTW_MEASURE, FFTW_PATIENT} are high compared to those for FFTW_ESTIMATE. The long execution times are due to the lengthy times to create the plans because FFTW_MEASURE tries to find an optimized plan by computing many FFTs whereas FFTW_PATIENT considers a wider range of algorithms to find a more optimal plan.

6.2.1. *Energy Optimization on a Single Socket Multicore CPU*

Figure 11 shows the dynamic energy comparision for PFFTTG employing FFTW between base and best combinations for workload sizes, 31936, 32704, and 35648 on a single-socket CPU (S1). The best combination (g,t)=(4,11) is the same for workload sizes, 31936 and 32704. The reductions in dynamic energy consumption in comparison with the base combination, (g,t)=(1,44), are 41% and 65%. For workload size 35648, the base combination is the best and outperforms the next best combination (g,t)=(2,22) by 5%. For Intel MKL FFT, the base combination, (g,t)=(1,44), is the best.

6.2.2. *Energy Optimization on a Dual-socket Multicore CPU*

Figures 12a, 12b show the results for PFFTTG employing FFTW on S4 for matrix sizes equal to N=30464 and N=32192. The minimum for dynamic energy is located in {4,7,8} threadgroups with 14 threads in each threadgroup for workload size (N=32192) and 12 threads in each threadgroup for workload size 30464. The minimum for the workload size 30464 is achieved for the combination, (g,t)=(8,12). The dynamic energy consumption for this combination is 661 Joules. The energy saving is around 30% in comparison with the best combination of threads for one group (g,t)=(1,45) whose dynamic energy consumption is 918 Joules. The minimum for the workload size (N=32192) is achieved for the combination, (g,t)=(4,14). The saving is around 35% in comparison with (g,t)=(1,16) where dynamic energy is 2197 Joules.

6.2.3. *Performance Optimization on a Single Socket Multicore CPU*

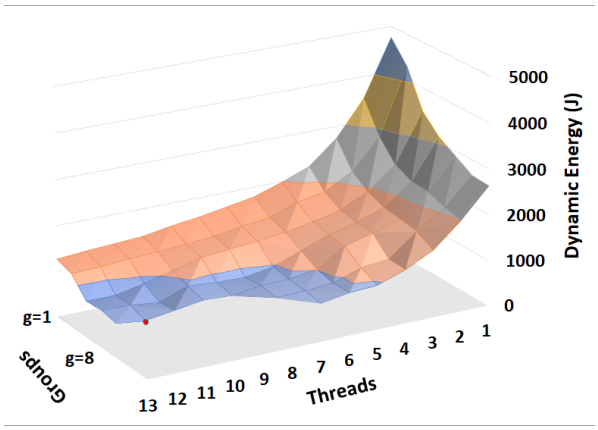
Figure 13 shows the results for PFFTTG employing FFTW on a single-socket CPU (S1). The best combination, (g,t)=(4,11), is the same for workload sizes, N=31936 and N=32704. The improvements over the base combination, (g,t)=(1,44), are 55% and 57%. For matrix dimension, N=35648, the base combination is the best and outperforms the next best combination, (g,t)=(2,22), by 5%.

Figure 14 shows the results for PFFTTG employing Intel MKL FFT. There are three best combinations, (g,t)=(2,22),(2,11),(4,11), for all the three workload sizes, where performances differ from each other by less than 5%. Their improvement over the base combination, (g,t)=(1,44), for N=18432 is 8%. For workload sizes, N=30720 and N=31616, the performance improvements are 25% and 26%. Furthermore, the average performance improvement over the best base combination for 23 tested workload sizes in the range, $5120 \leq N \leq 37000$, is 27%.

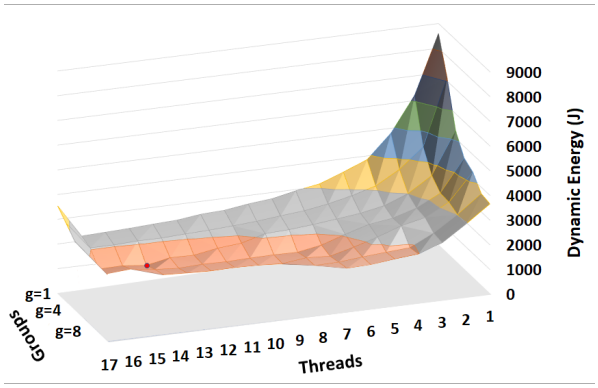
6.2.4. *Performance Optimization on Dual-socket Multicore CPUs*

All results in this section are represented by a 3D surface represented by axes for performance or energy, number of threadgroups (g) and the number of threads in each threadgroup, t. The location of the minimum in the surface is shown by the red dot.

Figure 15a shows the results of PFFTTG using FFTW3.3.7 on S4 for matrix dimension N=30976. The area with minimum execution time is located in the figure in the region containing {4,7,8} threadgroups with



(a)



(b)

Figure 12: (a). Energy profile of FFTW PFFTTG for different (g,t) configurations on S4 for workload size N=30464. (b). Energy profile of FFTW PFFTTG for different (g,t) configurations on S4 for workload size N=32192. Red dot represents the minimum.

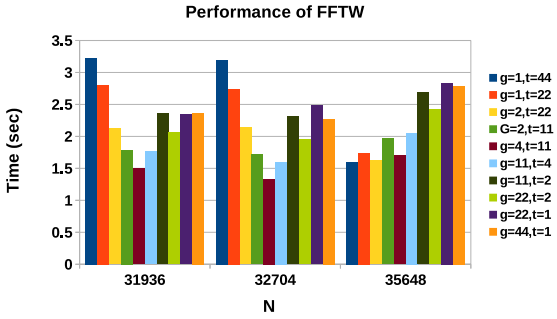


Figure 13: Performance of PFFTTG employing FFTW for different (g,t) configurations on S1.

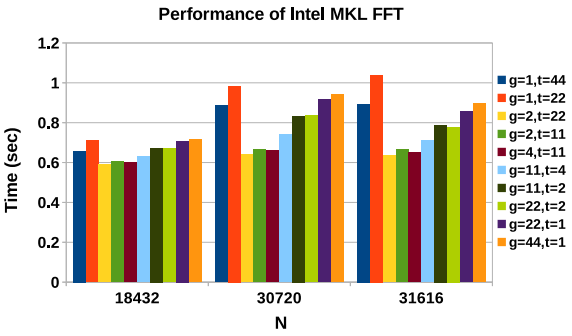


Figure 14: Performance of PFFTTG employing Intel MKL FFT for different (g,t) configurations on S1.

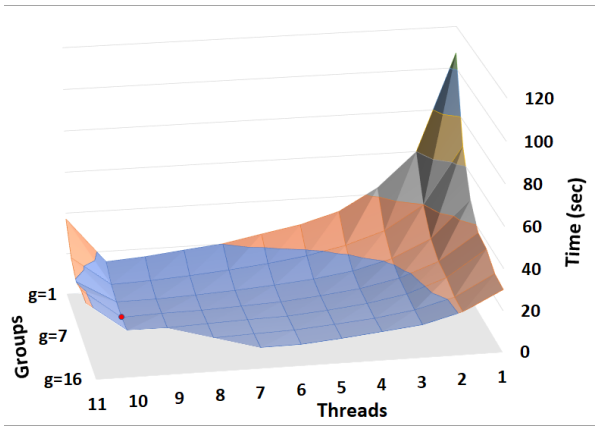
10 threads in each group. The minimum is achieved for the combination (g, t)=(7,10) with the execution time of 8 seconds. The speedup is around 100% in comparison with the best combination of threads for one group (g, t)=(1,10) where the execution time is 16 seconds.

Figure 15b presents the results of PFFTTG using FFTW3.3.7 on S3 for the matrix dimension N=17728. The minimum is centred around number of threads-groups equal to {4,7,8}. The minimum is achieved for the combination, (g, t)=(4,16). The performance improvement is 80% in comparison with (g, t)=(1,72), which is the best combination for one group.

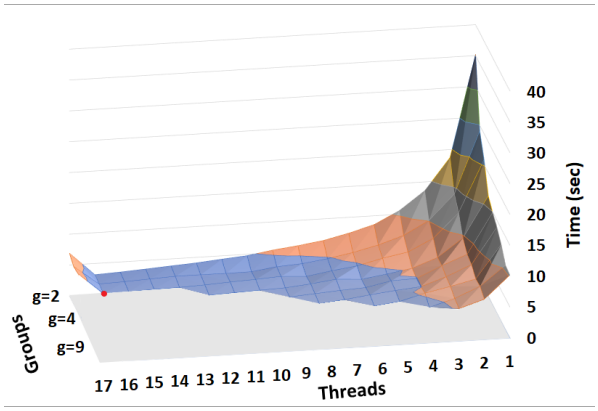
6.3. Bi-Objective Optimization for Dynamic Energy and Performance

6.3.1. Single Socket Multicore CPU

Figure 16a shows the Pareto front for PMMTG employing Intel MKL DGEMM on S1 for workload size

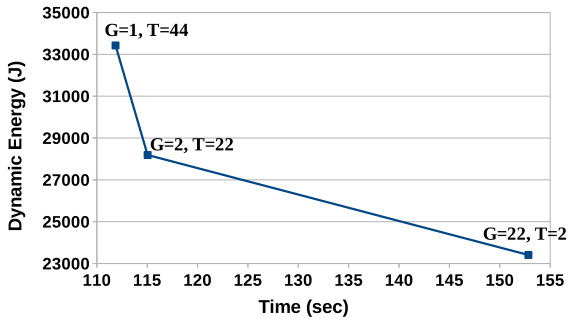


(a)

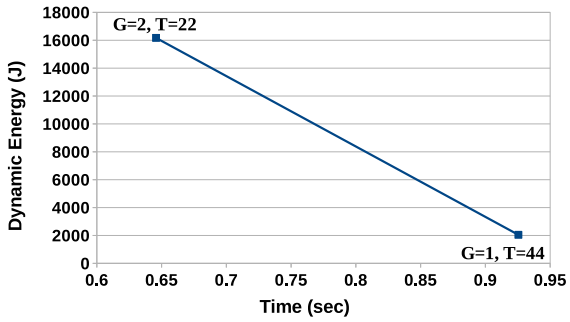


(b)

Figure 15: (a). Performance profile of FFTW PFFTTG for different (g,t) configurations on S4 for workload size, N=30976. (b). Performance profile of FFTW PFFTTG for different (g,t) configurations on S3 for workload size, N=17728. Red dot represents the minimum.



(a)



(b)

Figure 16: (a). Pareto front of Intel MKL DGEMM PMMTG application on S1 for workload size N=32768. (b). Pareto front of Intel MKL FFT PFFTTG on S1 for workload size N=31744.

32768. Optimizing for dynamic energy consumption alone degrades performance by 27%, and optimizing for performance alone increases dynamic energy consumption by 30%. The average and maximum sizes of the Pareto fronts for Intel MKL DGEMM are (2.3,3).

Figure 16b shows the Pareto front for PFFTTG based on Intel MKL FFT on S1 for workload size 31744. There are two Pareto-optimal solutions. Optimizing for dynamic energy consumption alone degrades performance by around 31%, and optimizing for performance alone increases dynamic energy consumption by 87%. The average and maximum sizes of the Pareto fronts for Intel MKL FFT are (2.6,3).

No bi-objective trade-offs were observed for FFTW and OpenBLAS applications. We will investigate two lines of research in our future work. One is the influence of workload distribution; The other is the absence of bi-objective trade-offs for open-source packages such as FFTW and OpenBLAS using a dynamic energy predictive model.

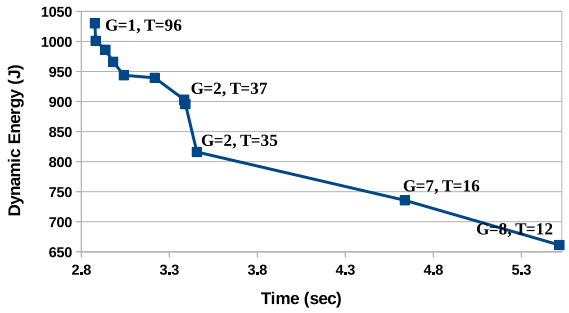
6.3.2. Dual-socket Multicore CPUs

In this section, we will focus on bi-objective optimization on dual-socket CPUs, S2 and S4.

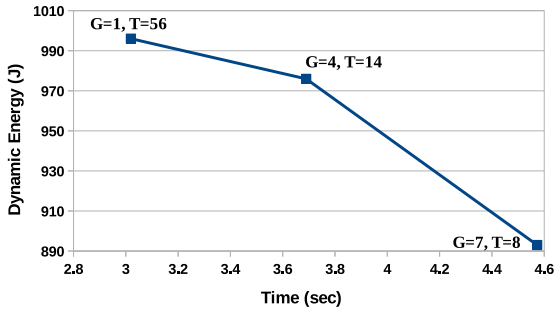
Figures 17a shows the Pareto fronts for PFFTTG FFTW on S4 for workload size, N=30464. The maximum number of Pareto-optimal solutions is 11. The optimization for dynamic energy consumption alone degrades performance by 49%, and optimizing for performance alone increases dynamic energy consumption by 35%.

Figure 17b shows the Pareto front for PFFTTG employing Intel MKL FFT on S2 for workload size, N=22208. Optimizing for dynamic energy consumption alone degrades performance by 33%, and optimizing for performance alone increases dynamic energy consumption by 10%. The average and maximum sizes of the Pareto fronts for FFTW and Intel MKL FFT are (3,11) and (2.7, 3).

Figure 18a shows the Pareto front for PMMTG em-

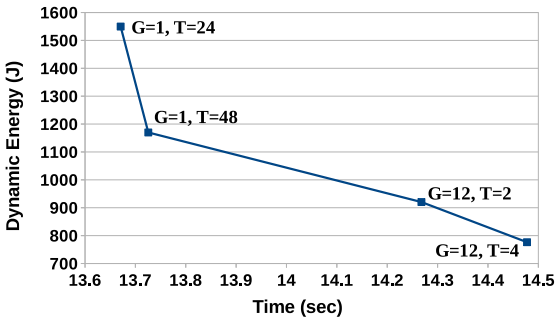


(a)

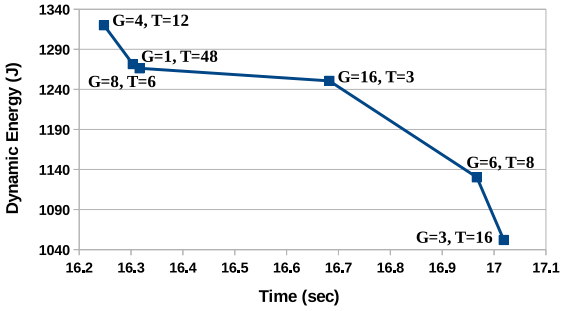


(b)

Figure 17: (a). Pareto front of FFTW PFFTTG on S4 for workload size, N=30464. (b). Pareto front of Intel MKL FFT PFFTTG on S4 for workload size, N=22208.



(a)



(b)

Figure 18: (a). Pareto front of Intel MKL DGEMM PMMTG application on S2 for workload size, N=17408. (b). Pareto front of OpenBLAS DGEMM PMMTG application on S2 for workload size, N=17408.

ploying Intel MKL DGEMM on S2 for workload size, N=17408. Optimizing for dynamic energy consumption alone degrades performance by 5.5%, and optimizing for performance alone increases dynamic energy consumption by 50.7%. The average and maximum sizes of the Pareto fronts are (1.8, 4).

Figure 18b shows the Pareto front for PMMTG based on OpenBLAS DGEMM on S2 for workload size, N=17408. There are six Pareto-optimal solutions. Optimizing for dynamic energy consumption alone degrades performance by around 5%, and optimizing for performance alone increases dynamic energy consumption by 20%. The average and maximum sizes of the Pareto fronts are 2.4 and 5.

The execution time of building the four dimensional discrete graph with dynamic energy and performance as two objectives and the two decision variables can be cost-prohibitive for its employment in dynamic schedulers and self-adaptable data-parallel applications. We will explore approaches to reduce this time in our future work.

6.4. Analysis Using Dynamic Energy and Performance Models

In this section, we present an overview of popular mainstream approaches for measurement of energy consumption during an application execution. We then propose a qualitative dynamic energy model employing performance monitoring counters (PMCs) as variables. The model reveals the cause behind the energy non-proportionality in modern multicore CPUs. The model along with the execution time of the application is used to analyze the Pareto front determined by our solution method on a dual-socket multicore platform.

6.4.1. Mainstream Energy Measurement Methods

There are three popular approaches to providing measurement of energy consumption during an application execution: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors, and (c) Energy predictive models.

While the first approach is demonstrated to be accurate (by Konstantakos, Chatzigeorgiou, Nikolaidis, and Laopoulos [83]), it can only provide the measurement at

a computer level and therefore lacks the ability to provide fine-grained device-level decomposition of the energy consumption of an application executing on several independent computing devices in a computer.

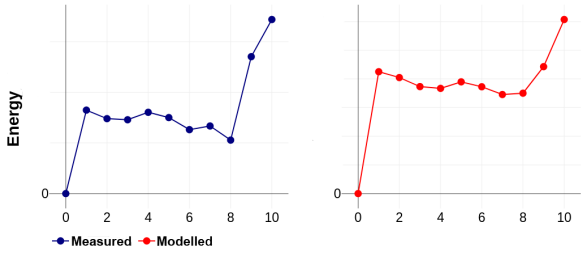
The second approach based on on-chip power sensors is now available in mainstream processors such as Intel and AMD Multicore CPUs, Nvidia GPUs, and Intel Xeon Phis. There are vendor specific libraries to acquire the power data from these sensors. For example, Running Average Power Limit (RAPL) [84] can be used to monitor power and control frequency (and voltage) of Intel CPUs, and Nvidia NVIDIA Management Library (NVML) [85] and Intel System Management Controller chip (SMC) [86] provide the power consumption by Nvidia GPUs and Intel Xeon Phi respectively. While the accuracy of GPU on-chip sensors is reported in the NVML manual (±5%) [85], the accuracies of the other sensors are not known. For the GPU and Xeon Phi on-chip sensors, there is no information about how a power reading is determined that would allow one to determine its accuracy. For the CPU on-chip sensors, RAPL uses separate voltage regulators (VR IMON) for both CPU and DRAM. VR IMON is an analog circuit within voltage regulator (VR), which keeps track of an estimate of the current [87]. There are two issues with these measurements. First, how this estimate is determined. Second, the accuracy of the estimates is not reported in the vendor manual. Fahad, Shahid, Reddy, and Lastovetsky [24] present the first comprehensive comparative study of the accuracy of state-of-the-art on-chip power sensors against system-level physical measurements using external power meters, which is considered to be the ground truth. They show that, owing to the nature of the deviations of the energy measurements provided by on-chip sensors from the ground truth, calibration can not improve the accuracy of the on-chip sensors to an extent that can favour their use in optimization of applications for dynamic energy. We define calibration as a constant adjustment (positive or negative value) made to the data points in a dynamic energy profile of an application obtained using a measurement approach (on-chip sensors or energy predictive models) with the aim to increase its accuracy or reduce its error against the ground truth.

The third approach based on software energy predictive models emerged as a popular alternative to determine the energy consumption of an application. Energy predictive models allow determination of fine-grained decomposition of energy consumption during the execution of an application. A vast majority of such models are linear and employ performance monitoring counters (PMCs) as predictor variables. PMCs are special-purpose registers provided in modern microprocessors to store the counts of software and hardware activities. The acronym PMCs is used to refer to software events, which are pure kernel-level counters such as *page-faults*, *context-switches*, etc. as well as micro-architectural events originating from the processor and its performance monitoring unit called the hardware events such as *cache-misses*, *branch-instructions*, etc. The research works (Shahid, Fahad, Reddy, and Lastovetsky [28], Fahad, Shahid, Reddy, and Lastovetsky [24]) are experimental proofs highlighting the inaccuracy of energy predictive models employing PMCs as predictor variables and that are purely based on positive correlation with dynamic energy consumption and how the accuracy of models can be improved using the property of *additivity* to select the PMCs.

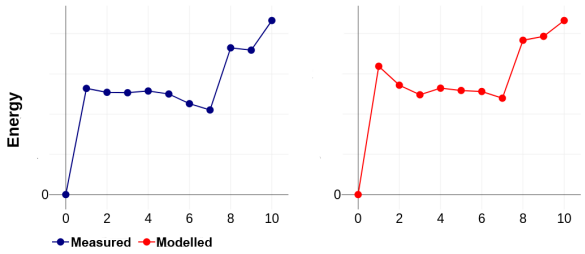
6.4.2. Energy Nonproportionality Detection Using Dynamic Energy and Performance Models

We propose a qualitative dynamic energy model employing performance monitoring counters (PMCs) as variables to explain the discovered energy nonproportionality. The experimental platform S2 and the application OpenBLAS-DGEMM is employed for the analysis. Likwid tool [88] is used to obtain the PMCs. On this platform, it offers 164 PMCs, which are divided into 28 groups (L2CACHE, L3CACHE, NUMA, etc.). The groups are listed in the supplemental [30]. All the PMCs for each workload size executed using different application configurations, (#threadgroups (g), #threads_per_group (t)) are collected. Each PMC value is the average for all the 24 physical cores. We analyzed the data to identify the major performance groups, which are highly correlated with the dynamic energy consumption. The highest correlation is contained in the data provided by TLB_DATA performance group. This group provides data activity, such as load miss rate, store miss rate and walk page duration, in L1 data translation lookaside buffer (dTLB), a small specialized cache of recent page address translations. If a dTLB miss occurs, the OS goes through the page tables. If there is a miss from the page walk, a page fault occurs resulting in the OS retrieving the corresponding page from memory. The duration of the page walk has the highest positive correlation with dynamic energy consumption based on our experiments. The PMCs associated with the TLB_DATA performance group are also highly *additive* (as shown by Shahid, Fahad, Reddy, and Lastovetsky [28]).

Non-negative multivariate regression is employed to construct our model of dynamic energy consumption based on the PMC data from dTLB. The model is shown below:



(a)



(b)

Figure 19: (a). Measured (left) and predicted (right) dynamic energy consumption of OpenBLAS DGEMM on S2 for workload size, N=16384. (b). Measured (left) and predicted (right) dynamic energy consumption of OpenBLAS DGEMM on S2 for workload size, N=17408.

Table 2: L1 dTLB PMC data for size 16384

Combination (g, t)	Dynamic Energy (J)	Time (sec)	L1 dTLB load miss duration (Cyc)	L1 dTLB store miss duration (Cyc)
(1,48)	824.2743	14.112	108.373	124.326
(4,12)	740.0211	14.177	113.515	105.363
(8,6)	729.1005	14.244	104.564	89.3753
(2,24)	802.6687	14.314	105.328	82.5185
(16,3)	750.6159	14.615	100.924	90.2733
(3,16)	631.3098	14.772	97.9180	76.1889
(6,8)	667.4856	14.818	96.8957	58.0210
(12,4)	528.0411	15.057	97.0492	52.8966
(24,2)	1352.141	15.875	100.106	82.7514
(48,1)	1719.012	18.685	111.902	85.9282

Table 3: L1 dTLB PMC data for size 17408

Combination (g, t)	Dynamic Energy (J)	Time (sec)	L1 dTLB load miss duration (Cyc)	L1 dTLB store miss duration (Cyc)
(4,12)	1320.0702	16.2478	105.961	122.191
(1,48)	1271.5506	16.3034	99.5398	63.7090
(8,6)	1266.3294	16.3166	95.7896	58.9096
(2,24)	1287.6882	16.4498	98.2180	74.6859
(16,3)	1250.5616	16.6824	95.2988	58.3551
(6,8)	1130.2412	16.9668	93.4336	47.9097
(3,16)	1052.0283	17.0187	90.5275	45.7483
(24,2)	1824.5795	18.0755	106.804	55.5686
(12,4)	1795.7680	20.5520	93.6595	46.5541
(48,1)	2164.1212	20.9868	96.6999	71.4943

$$E_{dynamic} = \beta_1 \times T + \beta_2 \times L + \beta_3 \times S \qquad (2)$$

where β_1 is the average CPU utilization, β_2 and β_3 are

the regression coefficients for the PMC data. T is the execution time of the application, L is the time of page walk caused by load miss and S is the time of page walk caused by store miss in dTLB. The coefficients of the model ($\{\beta_1,\beta_2,\beta_3\}$) are forced to be non-negative to avoid erroneous cases where large values for them gives rise to negative dynamic energy consumption prediction violating the fundamental energy conservation law of computing.

To test this model, we use two workload sizes 16384 and 17408. The PMC data that is obtained for these sizes and that is used to train the model is shown in the tables 2 and 3. The rows of the tables are sorted in increasing order of time. The blue colour in the tables shows the rows that are in the Pareto front. The time of page walk (last two columns, 4 and 5) is measured in cycles. As can be seen from the tables, the dynamic energy decreases as the number of cycles decreases.There is however a trade-off between the execution time of application and the page walk time. For a Pareto-optimal solution, a long execution time corresponds to smaller number of load and store cycles and thereby less dynamic energy consumption.

Two dynamic energy models for the workload sizes 16384 (Table 2) and 17408 (Table 3) were constructed. The coefficients for the workload size 16384 are $\{\beta_1 = 253.680, \beta_2 = 39.536, \beta_3 = 13.647\}$. The coefficients for the workload size 17408 are $\{\beta_1 = 137.953, \beta_2 = 12.564, \beta_3 = 3.835\}$. We then predict the dynamic energy consumption using the model and compare with the dynamic energy measured using HCLWattsUp, which is based on power meters and which we consider to be the ground truth. The Figures 19a and 19b illustrate the comparison. The x axis represents the number of a row in the Tables 2, 3. The modeled dynamic energy demonstrates the same trend as the measured dynamic energy using HCLWattsUp.

Kadayif, Nath, Kandemir, and Sivasubramaniam [89], Karakostas, Gandhi, Cristal, Hill, McKinley, Nemirovsky et al. [90] have studied TLB activity and have found that the address translation using the TLB consumes as much as 16% of the chip power on some processors. The authors propose different strategies to improve the reuse of TLB caches. Our solution method employing threadgroups (or grouping using multithreaded kernels) allows to fill the page tables more evenly and reduce the duration of page walk resulting in less dynamic energy consumption.

To summarize, our proposed dynamic model based on variables reflecting TLB activity (the duration of page walk) shows that the energy nonproportionality on our experimental platforms for the data-parallel applications is due to the activity of the data translation lookaside buffer (dTLB), which is disproportionately energy expensive. This finding may encourage the chip design architects to investigate and remove the nonproportionality in these platforms. In our future work, we would aim to identify other causes behind the lack of energy proportionality by broadening the range of platforms and applications.

7. Conclusion

The share of computing platforms in the total energy consumption is rapidly increasing thereby making the energy of computing the next grand technological challenge. Multicore processors are at the heart of modern computing platforms, and their energy efficiency is critical for addressing the challenge of energy of computing. Energy proportionality is the key design goal followed by architects of modern multicore processors. One of its implications is that optimization of an application for performance will also optimize it for energy.

In this work, we experimentally demonstrated that energy proportionality does not hold true for modern multicore processors. Based on this discovery, we proposed a novel application-level optimization method for bi-objective optimization of multithreaded data-parallel applications for energy and performance on a single multicore processor. The method uses two decision variables, the number of identical multithreaded kernels (threadgroups) and the number of threads in each threadgroup. A given workload is partitioned equally between the threadgroups.

We demonstrated our method using four highly optimized multithreaded data-parallel applications, two-dimensional fast Fourier transform written using Fastest Fourier Transform in the West package and Intel Math Kernel Library, and dense matrix multiplication written using optimized open-source basic linear algebra subprograms package and Intel Math Kernel Library, on four modern multicore processors one of which is a single socket multicore processor and the other three dual-socket with increasing number of physical cores per socket. We showed in particular that optimizing for performance alone results in significant increase in dynamic energy consumption whereas optimizing for dynamic energy alone results in considerable performance degradation and that our method determined good number of Pareto-optimal solutions.

Finally, we proposed a qualitative dynamic energy model employing performance monitoring counters as variables to explain the discovered energy nonproportionality and the Pareto-optimal solutions determined by

our solution method for modern multicore processors. The model employs performance monitoring counters that are selected primarily based on physical significance originating from fundamental physical laws such as conservation of energy of computing followed by high positive correlation with energy. It showed that the energy nonproportionality on our experimental platforms for the two data-parallel applications is caused by disproportionately high energy consumption by the data translation lookaside buffer activity.

Our proposed method can be used by application software developers to optimize their multithreaded applications on mainstream multicore processors for performance and energy and by hardware architects to find energy disproportional hardware components and improve their design.

The software implementations of the two parallel applications employing our optimization method are available at [91].

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474. We thank Roman Wyrzykowski and Lukasz Szustak for allowing us to use their Intel servers, HCLServer03 and HCLServer04.

References

[1] N. Jones, How to stop data centres from gobbling up the world's electricity, Nature 561 (2018) 163–166. doi:10.1038/d41586-018-06610-y.

[2] A. Andrae, T. Edler, On global electricity usage of communication technology: Trends to 2030, Challenges 6 (1) (2015) 117–157. doi:10.3390/challe6010117.
URL http://dx.doi.org/10.3390/challe6010117

[3] L. A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer (12) (2007) 33–37.

[4] R. Sen, D. A. Wood, Energy-proportional computing: A new definition, Computer 50 (8) (2017) 26–33.

[5] A. Lastovetsky, R. Reddy, 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 28 (4) (2017) 1119–1133.

[6] R. R. Manumachu, A. Lastovetsky, Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy, IEEE Transactions on Computers 67 (2) (2018) 160–177.

[7] R. Reddy Manumachu, A. L. Lastovetsky, Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution, Concurrency and Computation: Practice and Experience 0 (0) e4958.

[8] V. Petrucci, O. Loques, D. Mossé, R. Melhem, N. A. Gazala, S. Gobriel, Energy-efficient thread assignment optimization for heterogeneous multicore systems, ACM Trans. Embed. Comput. Syst. 14 (1) (Jan. 2015).

[9] Y. G. Kim, M. Kim, S. W. Chung, Enhancing energy efficiency of multimedia applications in heterogeneous mobile multi-core processors, IEEE Transactions on Computers 66 (11) (2017) 1878–1889.

[10] W. Wang, P. Mishra, S. Ranka, Dynamic cache reconfiguration and partitioning for energy optimization in real-time multi-core systems, in: 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), 2011, pp. 948–953.

[11] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, M. Prieto, Survey of scheduling techniques for addressing shared resources in multicore processors, ACM Comput. Surv. 45 (1) (Dec. 2012).

[12] G. Chen, K. Huang, J. Huang, A. Knoll, Cache partitioning and scheduling for energy optimization of real-time mpsocs, in: 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors, 2013, pp. 35–41.

[13] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, L. Jin, Dynamic thermal management through task scheduling, in: ISPASS 2008 - IEEE International Symposium on Performance Analysis of Systems and software, 2008, pp. 191–201.

[14] R. Z. Ayoub, T. S. Rosing, Predict and act: Dynamic thermal management for multi-core processors, in: Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED '09, ACM, 2009, pp. 99–104.

[15] T. Li, D. Baumberger, D. A. Koufaty, S. Hahn, Efficient operating system scheduling for performance-asymmetric multi-core architectures, in: SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007, pp. 1–11.

[16] E. Humenay, D. Tarjan, K. Skadron, Impact of process variations on multicore performance symmetry, in: 2007 Design, Automation Test in Europe Conference Exhibition, 2007, pp. 1–6.

[17] Intel® Math Kernel Library (Intel® MKL), Intel MKL FFT - fast fourier transforms (2019).
URL https://software.intel.com/en-us/mkl

[18] Z. Xianyi, Openblas, an optimized blas library (2019).
URL http://www.netlib.org/blas/

[19] FFTW, Fastest fourier transform in the west (2019).
URL http://www.fftw.org/

[20] H. Khaleghzadeh, Z. Zhong, R. Reddy, A. Lastovetsky., ZZGemmOOC: Multi-GPU out-of-core routines for dense matrix multiplization (2019).
URL https://git.ucd.ie/hcl/zzgemmooc.git

[21] H. Khaleghzadeh, Z. Zhong, R. Reddy, A. Lastovetsky, Out-of-core implementation for accelerator kernels on heterogeneous clouds, The Journal of Supercomputing 74 (2) (2018) 551–568.

[22] S. Khokhriakov, R. R. Manumachu, A. Lastovetsky, Performance optimization of multithreaded 2d fast fourier transform on multicore processors using load imbalancing parallel computing method, IEEE Access 6 (2018) 64202–64224.

[23] H. Khaleghzadeh, M. Fahad, A. Shahid, R. Reddy, A. Lastovetsky, Bi-objective optimization of data-parallel applications on heterogeneous hpc platforms for performance and energy through workload distribution, CoRR abs/1907.04080 (2019). arXiv:1907.04080.
URL http://arxiv.org/abs/1907.04080

[24] M. Fahad, A. Shahid, R. R. Manumachu, A. Lastovetsky, A comparative study of methods for measurement of energy of computing, Energies 12 (11) (2019). doi:10.3390/en12112204.
URL https://www.mdpi.com/1996-1073/12/11/2204

[25] D. Economou, S. Rivoire, C. Kozyrakis, 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.

[26] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, R. K. Gupta, Evaluating the effectiveness of model-based power characterization, in: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'11, USENIX Association, 2011.

[27] K. O'Brien, I. Pietri, R. Reddy, A. Lastovetsky, R. Sakellariou, A survey of power and energy predictive models in HPC systems and applications, ACM Computing Surveys 50 (3) (2017). doi:10.1145/3078811.
URL http://doi.org/10.1145/3078811

[28] A. Shahid, M. Fahad, R. Reddy, A. Lastovetsky, Additivity: A selection criterion for performance events for reliable energy predictive modeling, Supercomput. Front. Innov.: Int. J. 4 (4) (2017) 50–65.

[29] A. Shahid, M. Fahad, R. R. Manumachu, A. Lastovetsky, Improving the accuracy of energy predictive models for multicore CPUs using additivity of performance monitoring counters, in: V. Malyshkin (Ed.), Parallel Computing Technologies, Springer International Publishing, Cham, 2019, pp. 51–66.

[30] S. Khokhiakov, R. Reddy, A. Lastovetsky, HCLLIMB: Optimization of multithreaded matrix multiplication and 2d fast fourier transform on multicore cpu processors using parallel computing methods (2020).
URL https://csgitlab.ucd.ie/manumachu/hcllimb/blob/master/docs/supplemental_r1.pdf

[31] F. Bellosa, The benefits of event: driven energy accounting in power-sensitive systems, in: Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, ACM, 2000.

[32] C. Isci, M. Martonosi, Runtime power monitoring in high-end processors: Methodology and empirical data, in: Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36., 2003, pp. 93–104.

[33] T. Li, L. K. John, Run-time modeling and estimation of operating system power consumption, Sigmetrics Perform. Eval. Rev. 31 (1) (2003) 160–171.

[34] B. C. Lee, D. M. Brooks, Accurate and efficient regression modeling for microarchitectural performance and power prediction, Sigarch Comput. Archit. News 34 (5) (2006) 185–194.

[35] T. Heath, B. Diniz, B. Horizonte, E. V. Carrera, 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.

[36] X. Fan, W.-D. Weber, L. A. Barroso, Power provisioning for a warehouse-sized computer, in: 34th Annual International Symposium on Computer architecture, ACM, 2007, pp. 13–23.

[37] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, E. Ayguade, Decomposable and responsive power models for multicore processors using performance counters, in: Proceedings of the 24th ACM International Conference on Supercomputing, ACM, 2010, pp. 147–158.

[38] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, G. Giuliani, A methodology to predict the power consumption of servers in data centres, in: Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking, e-Energy '11, ACM, 2011, pp. 1–10.

[39] W. L. Bircher, L. K. John, Complete system power estimation using processor performance events, IEEE Trans. Comput. 61 (4) (2012) 563–577.

[40] W. Dargie, A stochastic model for estimating the power consumption of a processor, IEEE Transactions on Computers 64 (5) (2015).

[41] J. Haj-Yihia, A. Yasin, Y. B. Asher, A. Mendelson, Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems, ACM Trans. Archit. Code Optim. (TACO) 13 (4) (2016) 56.

[42] J. Mair, Z. Huang, D. Eyers, Manila: Using a densely populated pmc-space for power modelling within large-scale systems, Parallel Computing 82 (2019) 37–56.

[43] S. Kaxiras, M. Martonosi, Computer Architecture Techniques for Power-Efficiency, 1st Edition, Morgan and Claypool Publishers, 2008.

[44] S. Benedict, Review: Energy-aware performance analysis methodologies for hpc architectures-an exploratory study, J. Netw. Comput. Appl. 35 (6) (Nov. 2012).

[45] C. Mobius, W. Dargie, A. Schill, Power consumption estimation models for processors, virtual machines, and servers, IEEE Transactions on Parallel and Distributed Systems 25 (6) (2014).

[46] E. C. Inacio, M. A. R. Dantas, A survey into performance and energy efficiency in hpc, cloud and big data environments, Int. J. Netw. Virtual Organ. 14 (4) (2014) 299–318.

[47] A.-C. Orgerie, M. D. d. Assuncao, L. Lefevre, A survey on techniques for improving the energy efficiency of large-scale distributed systems, ACM Comput. Surv. 46 (4) (2014) 47:1–47:31.

[48] L. Tan, S. Kothapalli, L. Chen, O. Hussaini, R. Bissiri, Z. Chen, A survey of power and energy efficient techniques for high performance numerical linear algebra operations, Parallel Computing 40 (10) (2014) 559–573.

[49] S. Mittal, J. S. Vetter, A survey of methods for analyzing and improving gpu energy efficiency, ACM Computing Surveys (CSUR) 47 (2) (Jan. 2015).

[50] M. Dayarathna, Y. Wen, R. Fan, Data center energy consumption modeling: A survey, IEEE Communications Surveys & Tutorials 18 (1) (2016) 732–794.

[51] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, M. Prieto, Survey of energy-cognizant scheduling techniques, IEEE Transactions on Parallel and Distributed Systems 24 (7) (2013) 1447–1464.

[52] I. Kadayif, M. Kandemir, I. Kolcu, Exploiting processor workload heterogeneity for reducing energy consumption in chip multiprocessors, in: Proceedings Design, Automation and Test in Europe Conference and Exhibition, Vol. 2, 2004, pp. 1158–1163 Vol.2.

[53] M. Kondo, H. Sasaki, H. Nakamura, Improving fairness, throughput and energy-efficiency on a chip multiprocessor through dvfs, SIGARCH Comput. Archit. News 35 (1) (2007) 31–38.

[54] R. Watanabe, M. Kondo, H. Nakamura, T. Nanya, Power reduction of chip multi-processors using shared resource control cooperating with dvfs, in: 2007 25th International Conference on Computer Design, 2007, pp. 615–622.

[55] A. Fedorova, J. C. Saez, D. Shelepov, M. Prieto, Maximizing power efficiency with asymmetric multicore systems, ACM Queue 7 (10) (2009) 30:30–30:45.

[56] S. Herbert, S. Garg, D. Marculescu, Exploiting process variability in voltage/frequency control, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 20 (8) (2012) 1392–1404.

[57] A. Fedorova, M. Seltzer, M. D. Smith, Improving performance isolation on chip multiprocessors via an operating system scheduler, in: Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques, PACT '07, IEEE Computer Society, 2007, pp. 25–38.

[58] S. Zhuravlev, S. Blagodurov, A. Fedorova, Addressing shared

resource contention in multicore processors via scheduling, in: Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems, ASPLOS XV, ACM, 2010, pp. 129–142.

[59] E. Ebrahimi, R. Miftakhutdinov, C. Fallin, C. J. Lee, J. A. Joao, O. Mutlu, Y. N. Patt, Parallel application memory scheduling, in: Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-44, ACM, 2011, pp. 362–373.

[60] M. K. Jeong, D. H. Yoon, D. Sunwoo, M. Sullivan, I. Lee, M. Erez, Balancing dram locality and parallelism in shared memory cmp systems, in: IEEE International Symposium on High-Performance Comp Architecture, 2012, pp. 1–12.

[61] Jiang Lin, Qingda Lu, Xiaoning Ding, Zhao Zhang, Xiaodong Zhang, P. Sadayappan, Gaining insights into multicore cache partitioning: Bridging the gap between simulation and real systems, in: 2008 IEEE 14th International Symposium on High Performance Computer Architecture, 2008, pp. 367–378.

[62] D. K. Tam, R. Azimi, L. B. Soares, M. Stumm, Rapidmrc: Approximating l2 miss rate curves on commodity systems for online optimizations, in: Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XIV, ACM, 2009, pp. 121–132.

[63] L. Tang, J. Mars, N. Vachharajani, R. Hundt, M. L. Soffa, The impact of memory subsystem resource sharing on datacenter applications, in: Proceedings of the 38th Annual International Symposium on Computer Architecture, ISCA '11, ACM, 2011, pp. 283–294.

[64] J. Mars, L. Tang, R. Hundt, K. Skadron, M. L. Soffa, Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations, in: Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-44, ACM, 2011, pp. 248–259.

[65] R. Ge, X. Feng, W.-c. Feng, K. W. Cameron, CPU MISER: A performance-directed, run-time system for power-aware clusters, 2007 International Conference on Parallel Processing, IEEE Computer Society, 2007.

[66] S. Huang, W. Feng, Energy-efficient cluster computing via accurate workload characterization, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, 2009.

[67] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E.-G. Talbi, A. Zomaya, D. Tuyttens, A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, Journal of Parallel and Distributed Computing 71 (11) (2011) 1497 – 1508.

[68] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, T. Fahringer, A multi-objective approach for workflow scheduling in heterogeneous environments, 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid), IEEE Computer Society, 2012, pp. 300–309.

[69] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Generation Computer Systems 28 (5) (2012) 755 – 768, special Section: Energy efficiency in large-scale distributed systems.

[70] Y. Kessaci, N. Melab, E.-G. Talbi, A pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation, Cluster Computing 16 (3) (2013) 451–468.

[71] J. J. Durillo, V. Nae, R. Prodan, Multi-objective energy-efficient workflow scheduling using list-based heuristics, Future Generation Computer Systems 36 (2014) 221 – 236.

[72] A. Das, A. Kumar, B. Veeravalli, C. Bolchini, A. Miele, Combined dvfs and mapping exploration for lifetime and soft-error susceptibility improvement in mpsocs, in: 2014 Design, Automation Test in Europe Conference Exhibition (DATE), 2014, pp. 1–6.

[73] Z. Zhang, J. M. Chang, A cool scheduler for multi-core systems exploiting program phases, IEEE Transactions on Computers 63 (5) (2014).

[74] J. Kołodziej, S. U. Khan, L. Wang, A. Y. Zomaya, Energy efficient genetic-based schedulers in computational grids, Concurrency: Practice and Experience 27 (4) (2015) 809–829.

[75] V. Sundriyal, M. Sosonkina, Joint frequency scaling of processor and DRAM, The Journal of Supercomputing 72 (4) (Apr 2016).

[76] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda, M. Kondo, I. Miyoshi, Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing, International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2015.

[77] N. Gholkar, F. Mueller, B. Rountree, Power tuning HPC jobs on power-constrained systems, International Conference on Parallel Architectures and Compilation, ACM, 2016.

[78] H. F. Sheikh, I. Ahmad, D. Fan, An evolutionary technique for performance-energy-temperature optimized scheduling of parallel tasks on multi-core processors, IEEE Transactions on Parallel and Distributed Systems 27 (3) (2016) 668–681.

[79] A. Abdi, A. Girault, H. R. Zarandi, Erpot: A quad-criteria scheduling heuristic to optimize execution time, reliability, power consumption and temperature in multicores, IEEE Transactions on Parallel and Distributed Systems (2019) 1–1.

[80] B. Subramaniam, W. C. Feng, Statistical power and performance modeling for optimizing the energy efficiency of scientific computing, IEEE/ACM Int'l Conference on Cyber, Physical and Social Computing (CPSCom), 2010.

[81] A. Chakrabarti, S. Parthasarathy, C. Stewart, A pareto framework for data analytics on heterogeneous systems: Implications for green energy usage and performance, in: Parallel Processing (ICPP), 2017 46th International Conference on, IEEE, 2017, pp. 533–542.

[82] HCL, HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter (2016).
URL `http://git.ucd.ie/hcl/hclwattsup`

[83] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, T. Laopoulos, Energy consumption estimation in embedded systems, IEEE Transactions on Instrumentation and Measurement 57 (4) (2008) 797–804.

[84] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, D. Rajwan, Power-Management architecture of the intel microarchitecture Code-Named sandy bridge, IEEE Micro 32 (2) (2012) 20–27.

[85] Nvidia, Nvidia management library: NVML reference manual (10 2018).
URL `https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf`

[86] I. Corporation, Intel® xeon phi™ coprocessor system software developers guide (06 2014).
URL `https://software.intel.com/sites/default/files/managed/09/07/xeon-phi-coprocessor-system-software-developers-guide.pdf`

[87] C. Gough, I. Steiner, W. Saunders, Apress, 2015.

[88] J. Treibig, G. Hager, G. Wellein, Likwid: A lightweight performance-oriented tool suite for x86 multicore environments, in: 2010 39th International Conference on Parallel Processing Workshops, IEEE, 2010, pp. 207–216.

[89] I. Kadayif, P. Nath, M. Kandemir, A. Sivasubramaniam, Re-

ducing data tlb power via compiler-directed address generation, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26 (2) (2007) 312–324.

[90] V. Karakostas, J. Gandhi, A. Cristal, M. D. Hill, K. S. McKinley, M. Nemirovsky, M. M. Swift, O. S. Unsal, Energy-efficient address translation, in: 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2016, pp. 631–643.

[91] S. Khokhiakov, R. Reddy, A. Lastovetsky, HCLLIMB: Optimization of multithreaded matrix multiplication and 2d fast fourier transform on multicore cpu processors using parallel computing methods (2020).
URL `https://csgitlab.ucd.ie/manumachu/hcllimb`