A Survey of Power and Energy Predictive Models in HPC Systems and Applications

KENNETH O'BRIEN, University College Dublin, Ireland
ILIA PIETRI, University of Manchester, UK
RAVI REDDY, University College Dublin, Ireland
ALEXEY LASTOVETSKY, University College Dublin, Ireland
RIZOS SAKELLARIOU, University of Manchester, UK

Power and energy efficiency are now critical concerns in extreme-scale high performance scientific computing. Many extreme-scale computing systems today (For example: Top500) have tight integration of multicore CPU processors and accelerators (mix of GPUs, Intel Xeon Phis, or FPGAs) empowering them to provide not just unprecedented computational power but also to address these concerns. However, such integration renders these systems highly heterogeneous and hierarchical thereby necessitating design of novel performance, power, and energy models to accurately capture these inherent characteristics.

There are now several extensive research efforts focusing exclusively on power and energy efficiency models and techniques for the processors composing these extreme-scale computing systems. This article synthesizes these research efforts with absolute concentration on predictive power and energy models and prime emphasis on node architecture. Through this survey, we also intend to highlight the shortcomings of these models to correctly and comprehensively predict the power and energy consumptions by taking into account the hierarchical and heterogeneous nature of these tightly-integrated high performance computing systems.

CCS Concepts: •**Computing methodologies** → **Parallel computing methodologies; Modeling methodologies;** •**Computer systems organization** → **Parallel architectures;** •**Hardware** → **Power estimation and optimization;**

Additional Key Words and Phrases: Survey, Power models, Energy models, Power consumption, Energy consumption, HPC

ACM Reference Format:
Kenneth O'Brien, Ilia Pietri, Ravi Reddy, Alexey Lastovetsky and Rizos Sakellariou, 2015. A Survey of Power and Energy Predictive Models in HPC Systems and Applications *ACM Comput. Surv.* V, N, Article A (January YYYY), 38 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Energy is now a first-class design constraint along with performance in all computing settings. Energy-proportional designs [Barroso and Hölzle 2007] in servers are now crucial to the operational efficiency of data centers. Most data centers use almost as much non-computing or "overhead" energy (due to power distribution and cooling) as

This research is conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474. It is also supported by the Structured PhD in Simulation Science which is funded by the Programme for Research in Third Level Institutions (PRTLI) Cycle 5 and co-funded by the European Regional Development Fund. This work is partially supported by EU under the COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS)
Author's addresses: Kenneth O'Brien, Ravi Reddy, and Alexey Lastovetsky, School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland
Ilia Pietri and Rizos Sakellariou, School of Computer Science, The University of Manchester; Kilburn Building, Oxford Road, Manchester, M13 9PL, United Kingdom.

DOI: 0000001.0000001

they do to power their servers [Dat 2015]. According to a 2010 DOE Office of Science report [DOE 2010] on the challenges and opportunities of building and using exaflop supercomputers, "All of the technical reports on exascale systems identify the power consumption of the computers as the single largest hardware research challenge. To achieve an exascale system using current technology, the annual power cost to operate the system would be above $2.5 billion per year." Therefore, energy has now become the leading concern for HPC system designs also [DOE 2010].

To address the new concerns of power consumption and energy efficiency (performance/watt) while continuing to provide unprecedented performance, most of the extreme-scale high performance scientific computing systems today (especially supercomputers [Top500 2015]) have multicore CPUs tightly integrated with accelerators (GPUs, Xeon Phis, FPGAs, etc.). However, such tight integration has rendered these systems highly heterogeneous and hierarchical where resource contention, non-uniform memory access (NUMA), limited accelerator memory, and low bandwidth of communication links (PCIe) between the host (multicore CPU) and accelerators have become inherent characteristics.

Figure 1 shows the architecture common in supercomputers today highlighting these complications from modeller's and programmer's point of view:

— The nodes are connected to each other by a fast interconnect where each node has multiple processors with shared memory space. Components in a node are managed by a chipset and are connected via a PCI-Express bus [PCIE 2003].
— A node has multiple CPUs. Each CPU has multiple number of cores, which share resources. Each CPU has multiple levels of caches with smaller, faster caches closer to a core and larger, less fast caches shared by cores.
— Memory access from cores on a CPU to the memory controlled by another CPU happens via a high-speed on-chip interconnect such as
  (1) QuickPath Interconnect (QPI) [QPI 2008] from Intel
  (2) HyperTransport (HT) [AMDHT 2001] from AMD
— CPUs have a memory controller integrated into them. Multiple channels are used to increase memory bandwidth. Multiple DIMMs are used to increase capacity per channel.
— The CPUs are connected to one or more accelerators with limited memory by a low bandwidth communication link (PCIe).
— The accelerators have unique architectures with their own hierarchical memory structures. For example, the NVIDIA Kepler GK210 architecture [GK210 2014] has the following features:
  (1) 15 Streaming Multiprocessors (SMX) and six 64-bit memory controllers.
  (2) The total amount of configurable on-chip memory per SMX is 128 KB. The allowed memory configurations are 112 KB shared memory and 16 KB of L1 cache, 96 KB shared memory and 32 KB L1 cache, and 80 KB of shared memory and 48 KB L1 cache.
  (3) Each SMX also has a read-only data cache of 48 KB.
  (4) 1536 KB of dedicated L2 cache memory.
  (5) 12 GB GDDR5 DRAM memory.
— Two accelerators on the same PCIe bus can transfer data directly between their memories avoiding any copies to system memory. Data can be transferred across a network between two accelerators in different nodes bypassing host memory altogether. For example: GPUDirect versions introduced in CUDA4.0 and CUDA5.0 [CUD 2015] provide these capabilities.

As one can see, the node architecture has become highly heterogeneous and hierarchical (two-level with components and architectural units within a component). Need-

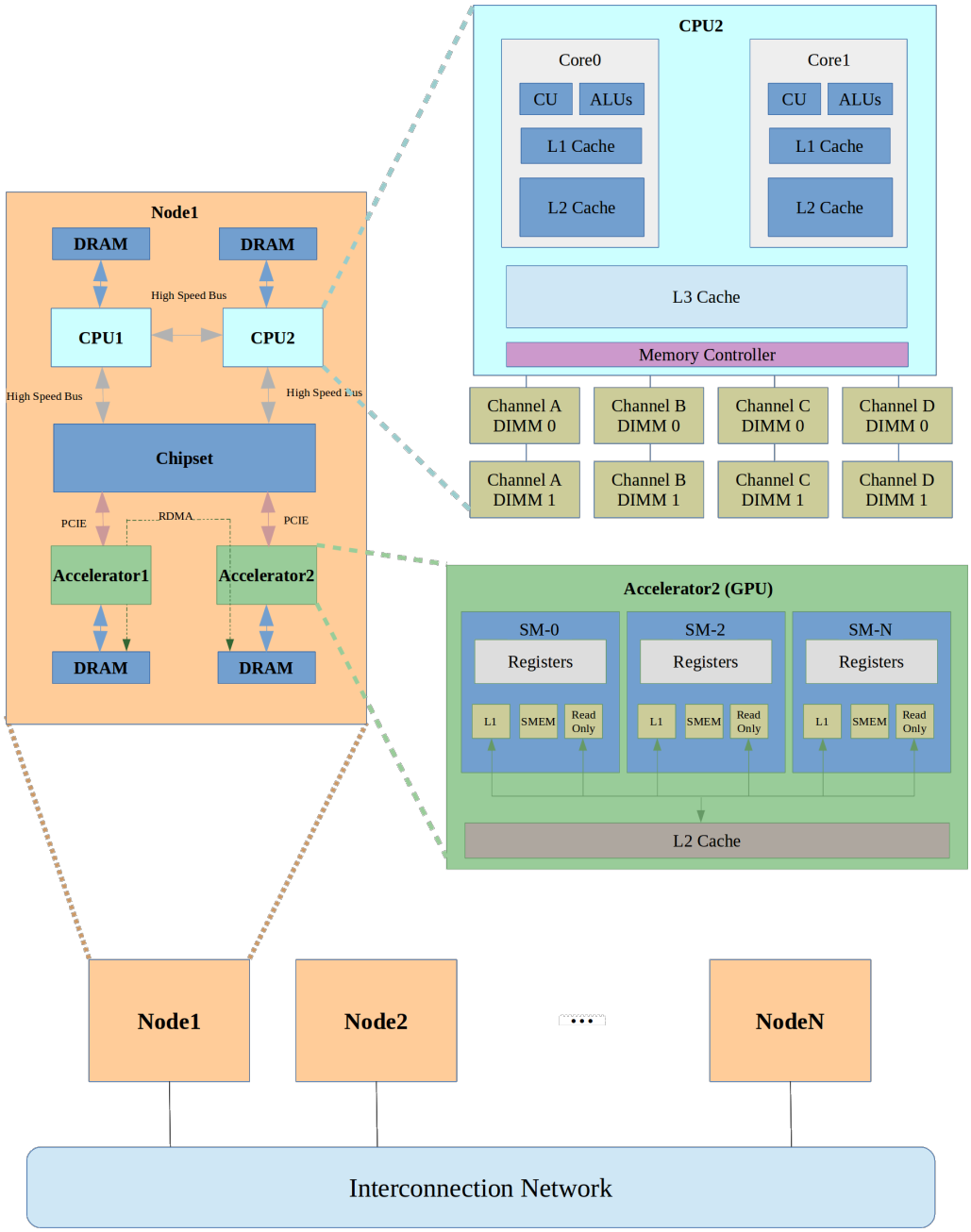

Fig. 1 The hierarchical and heterogeneous node architecture of extreme-scale computing systems.

less to say, modelling the power and energy consumption of such a node is a complex task. There exist several research efforts modelling component-wise power and energy consumptions of a node. Models ([Bellosa 2000], [Isci and Martonosi 2003], [Heath

Table I Table of contents for the survey

Section No.	Description
2	The salient characteristics we use to classify predictive power and energy models in this survey
3	Power and energy predictive models for CPUs
4	Power and energy predictive models for GPUs
5	Power and energy predictive models for Xeon Phis and FPGAs
6	Analytical energy predictive models
7	Power and energy predictive models for high performance computing applications
7.1	Interconnects and Communications
8	Related surveys
9	A case study examining the potential accuracy and relevance of performance monitoring counter (PMC) based models for modern node architectures
10	Discussion
11	Conclusion

et al. 2005], [Economou et al. 2006], [Fan et al. 2007], [Wang et al. 2010], [Basmadjian et al. 2011], [Bircher and John 2012]) deal with power consumption on single-core CPUs and multicore CPUs. Models ([Hong 2010], [Chen et al. 2011], [Kasichayanula et al. 2012], [Nagasaka et al. 2010], [Song et al. 2013], [Wang and Cao 2015]) focus on power consumption of accelerators such as GPUs. Many models attempt to provide per-unit decomposition of a nodal component by correlating the power consumption of a unit with the hardware performance events associated with it ([Isci and Martonosi 2003], [Heath et al. 2005], [Lee and Brooks 2006], [Economou et al. 2006], [Hong 2010], [Chen et al. 2011], [Kasichayanula et al. 2012], [Song et al. 2013]).

Models predicting the power consumption of a node construct the model by a simple summation of the power models of its components. Few research efforts have attempted to model the power consumption of the low-bandwidth links between multicore CPUs and accelerators ([Song et al. 2013]). Many models use linear correlation (multiple linear regression methods) between power and energy consumption and performance counters associated with various architectural units of a component. Some models ([Chen et al. 2011], [Song et al. 2013], [Wang and Cao 2015]) try to capture the hierarchical nature of a node by incorporating non-linearity in their design. However, the non-linearity is built into these models through explicit use of methods such as artificial neural networks, fuzzy wavelet neural networks, random forest methods, etc., where output is expressed as a non-linear function of inputs rather than considering methodically the hierarchical nature and resource contention of components in a node.

Through this survey, we endeavour to synthesize these research efforts by extracting common but crucial features of these models. By doing so, we also aim to capture and highlight their shortcomings.

We classify the models in our survey based on the dominant components at the level of node architecture which is our primary focus area. The rest of the paper is organized as shown in Table I.

2. CLASSIFICATION OF POWER/ENERGY MODELS

Table II presents the most salient characteristics of the power and energy models that we will focus in this survey. A model is represented by its bibliographical reference

Table II Salient characteristics of the power and energy models.

Model Characteristic		Description
Model		Name of the model and year of publication
Parameters		Parameters of the model
Level of abstraction		How the model captures the hierarchical nature of modern processor architectures?
Type of power		Is the predicted power instantaneous or average?
Is energy predicted?		Is Energy predicted? If yes, how?
Decomposition		Does the model provide per-component power and energy breakdown?
Accuracy of power prediction	**Accuracy of dynamic power prediction**	The maximum percent error in the dynamic power prediction calculated from the total power and static power consumptions reported by the authors
	Accuracy of total power prediction	The maximum percent error in the total power prediction reported by the authors
Accuracy of energy prediction		The maximum percent error in the energy prediction reported by the authors
Implementation Complexity (effort-week/effort-month/effort-year (EW/EM/EY))		The implementation effort required to build the model
Portability		Is the model portable to next-gen processors in the same architecture space?

and its *parameters*. Some of the models that we surveyed have a large set of model parameters. To avoid hampering the flow of our paper, self-contained explanations for each of the parameters are not included. We advise the readers to consult the original sources. However, we decided to provide the full list of the parameters (Appendix D) for each model due to two reasons. Firstly, the effort spent to distil the parameters from their source was considerable. Secondly, such complete enumeration of parameters for a model would hopefully assist one in building it with minimal effort.

The characteristic *Level of abstraction* specifies how the model captures the inherent hierarchical and heterogeneous nature of modern processor architectures.

— *Linear Independence* - All the components of a node are modelled independently. The model for a node is a linear combination of the models of its components.
— *Linear Dependence* - The components of a node are modelled taking into account the dependencies (shared structures) between them and expressing these dependencies linearly. For example:
  — The models for CPUs are constructed taking into account the shared resources (Last level cache) between them.
  — For an application employing both CPUs and accelerators, the models for CPUs and the accelerators are constructed taking into account the shared resources (last level cache) between the CPUs and the communication link (PCIe) connecting the CPUs and the accelerators.
— *Non-linear Independence* - All the components of a node are modelled independently. However, the model for a node is a non-linear combination of the models of its components.
— *Non-linear Dependence* - This is the most complex model. The components and dependencies between them (shared resources, communication links) are modelled non-linearly by taking into account their inherent hierarchical and heterogeneous nature.

The model characteristic *Type of power* specifies if the predicted power is instantaneous or average. The average power during a time duration ΔT is defined as the total amount of energy ΔE consumed during the time duration ΔT and is given by $\Delta E/\Delta T$. The instantaneous power is the limit of the average power as the time interval ΔT tends to zero:

$$P_{instantaneous} = \lim_{\Delta T \to 0} P_{avg} = \lim_{\Delta T \to 0} \Delta E/\Delta T = dE/dt \quad (1)$$

In the context of power prediction at application level, average power can be defined as total amount of energy consumed during the execution of an application (ΔE) divided by the total execution time of the application (ΔT). However, there is no clear-cut definition for instantaneous power. We can define it as the power at any instant (infinitesimal time interval) during the execution of an application. For the models classified as predicting instantaneous power in this paper, the time interval is considered to be one second since this was the granularity of system utilization measures (also the sampling interval of power meters) that were the basis of these models.

If a model predicts only the average total power consumption, it is imperative that authors of the model make known the total execution times (ΔT) of their test applications used to build and validate the model. This is because the true accuracy of a model is determined by how closely it exemplifies the fluctuations in power consumptions during smaller durations (small multiple of seconds). For an application executing for a long duration (hours to days), the prediction error reported although accurate for such long runs may not, however, truly represent the model's prediction ability for all ranges of execution times. That is, a model that is built solely from a training set of applications with large execution times (For example: highly compute-bound BLAS3 routines for large dense matrices) may not accurately capture the minute fluctuations in power consumptions of some applications (especially those executing on accelerators that complete within seconds or even fraction of a second).

There are two types of power consumptions in a component: dynamic power and static power. Dynamic power consumption is caused by the switching activity in the component's circuits. Static power is the power consumed when the component is not active or doing work. Static power is also known as idle power or leakage power or base power. Some authors [Molka et al. 2010] differentiate baseline power consumption from idle power consumption. They define baseline power consumption as the "lowest possible power consumption of the whole system with all processor cores in C0 state (operating state, CPU fully turned on)" and idle power consumption as the "lowest possible power consumption of the whole system with all processor cores in deep sleep states (C3-C6) [ACPI 2015]".

To remove any confusion, we would like to clarify our definitions further. From an application point of view, we define dynamic and static power consumption as the power consumption of the whole system with and without the application execution respectively. From the component point of view, we define dynamic and static power consumption of the component as the power consumption of the component with and without the application utilizing the component during its execution respectively. For example, supposing we want to determine the static power consumption of an accelerator in a node. First, the accelerator is removed and the power consumption of the whole system is measured using a power meter. Then, the accelerator is inserted into the node and the power consumption of the whole system is again measured using a power meter. The difference gives the static power consumption of the accelerator. To determine the dynamic power consumption of an accelerator during an application execution, the static power consumption of the accelerator is subtracted from the power consumption of the whole system during the application execution.

In this paper, when we refer to power consumption of a component or an application, we mean the total power consumption, which is a summation of dynamic and static powers. Unless otherwise specified explicitly, the models report predicted power consumption in watts.

The model characteristic *Is energy predicted?* specifies if the models predict the energy consumption. If yes, then how is it predicted?

— *Power* $\times$ *Timing* - By constructing power and timing models and composing the energy model from these individual models.
— *Explicit* - By explicitly modelling the energy consumption without any reference to power consumption.

The model characteristic *Decomposition* specifies the various architectural units of the nodal components whose power and energy consumptions are modelled. Some models model just the power consumption of a node without decomposing it further whereas some models model the power consumption of all the architectural units composing a nodal component (For example, all the architectural units within a CPU or a GPU, etc.). Some models try to model all the components of a node and all the architectural units within a component. In this case, they may take into account the communication links between the architectural units within a component (For example, on-chip interconnect, high-speed buses, etc.) and/or the low-bandwidth links between the components (For example, PCIe link between a multicore CPU and an accelerator). This model characteristic is presented separately in Appendix D.

The accuracy/error of prediction reported by a model is calculated using the formula:

$$\text{Percent error of prediction} = \frac{|\text{Actual value - Predicted value}|}{\text{Actual value}} \times 100 \tag{2}$$

where *Actual value* is the value measured directly using a power meter and *Predicted value* is the value predicted by the model.

The total power consumption of an application is a sum of dynamic and static power consumptions. The static power consumption is a constant and does not change during the execution of the application. Some papers ([Bertran et al. 2013], [Hong 2010], [Chen et al. 2011], [Kasichayanula et al. 2012], [Lim et al. 2014]) report the static power consumption as well as the total predicted power consumption of an application, which allows us to calculate the predicted dynamic power consumption and thereby its prediction error. We therefore split the model characteristic *Accuracy of power prediction* into two categories: *Accuracy of dynamic power prediction* which is the maximum error in prediction of dynamic power consumption calculated from the static power and the total power consumptions reported by the authors and *Accuracy of total power prediction* which is the maximum error in prediction of total power consumption reported by the authors.

We make this differentiation because static power being a constant will be the same for all models and therefore, for the sake of truthfulness, the models should be compared based on their prediction accuracy of dynamic power consumption. We elucidate this subtlety using two examples from published results. In our first example, consider a model that reports predicted and measured total power consumption of a GPU to be 165W and 180W, respectively. It would report the prediction error to be 8.3%. However, if it is known that the static power of the GPU is 90W, then the actual prediction error (based on dynamic powers only) would be double, 16.6%.

In our second example, consider two different power models (A and B) with same prediction errors of 5% for an application execution on two different machines (A and B) with same total power consumption of 100W. One would consider both the models to be equally accurate. But supposing it is known that the dynamic power proportions for

the machines are 30% and 60%. Now, the true prediction errors (using dynamic powers only) for the models would be 16.6% and 8.3% respectively. Therefore, the second model B should be considered more accurate than the first.

However, we also would like to take nothing away from the research works that are dedicated to predicting static power. Predicting static power can be quite challenging. There are numerous models researching this topic in the field of digital design and can be a subject for a separate research survey. Our main goal of this work is application-level models for HPC systems and applications that are solely dedicated to predicting dynamic power.

The model characteristic *Accuracy of energy prediction* is the maximum error in prediction of total energy consumption reported by the authors. Unless otherwise stated explicitly, the models report predicted energy consumption in watt seconds or joules.

The model characteristic *Implementation Complexity* expresses a rough estimate of the effort made by an experienced researcher in implementing the model. Some models that we survey involve painstaking efforts on the part of a programmer to accurately build the model experimentally. For example, models that use PMC-based approach require the model-writer to design a meticulously-written micro-benchmark test suite that stress the various architectural units to determine the access rates of the components. Models that use sophisticated regression and artificial neural network methods demand the model-writer to not only possess or acquire practical knowledge of these methods but also to carefully pick a diverse set of kernels (from well-known benchmarks as well as real applications) to train the models.

The model characteristic *Portability* can have many connotations. Here, we refer to its reuse and its ability to predict the power and energy consumption of next generation processors in the same architecture space. That is, a model designed for a NVIDIA GT200 processor should be applicable for newer generations of processors such as Fermi, Kepler, and Maxwell in order to be considered portable. A model designed specially for a single-core CPU may not be applied straight-forwardly to multicore CPU even though such a model can be used as a foundation for modelling the power and energy consumptions of individual cores in a multicore processor. Such models would be considered non-portable. However, when we define *portability*, we do not mean generality of the model to be able to predict power consumption for all processor architectures. Even though it is ideal to have such a model, we don't expect that a model designed for multicore CPUs would also be used for modelling GPU architectures. If we apply the criterion of generality, all the models in this survey would fail to satisfy it. Most of the models in our survey use hardware performance events as model parameters and we have rated them to be portable since all processors today come with a large set of diverse performance events. However, these models are not portable across architectures and hence not generic since performance events differ between architectures.

3. POWER AND ENERGY MODELS FOR CPUS

For more than three decades prior to mid-2000s, computer users came to expect performance doubling every 18 months due to Moore's law [Moore 1965] and Dennard scaling [Dennard 1974]. Both clock rate and power increased rapidly. Power and energy models appeared in early 2000s addressing the increasing power dissipation for single-core CPU processors.

However, by 2004, computer designers hit the power wall caused by problems stemming from increasing power consumption and increasing power density (amount of power dissipated per unit area, which represents the heat dissipation). The power problem was caused primarily by the breakdown of Dennard scaling, a scaling model whereby the power density of a transistor based processor of a unit area remains con-

stant due to voltage and current scaling down with the length of the transistor. Up until 2004, moving to a smaller transistor process meant frequency could be increased for no increase in heat dissipation. The breakdown of Dennard scaling meant frequency scaling was no longer economical. The chip fabrication industry turned to multicore CPU architectures to address this problem of increased power consumption and power density. Frequency scaling was abandoned in favour of multiple processors per chip. Subsequently, power models for multicore CPUs started appearing towards late 2000s. The majority of these models focused on the CPU since it was the dominant component in terms of power consumption. Table III shows the salient features of these models under the heading "CPU".

Unfortunately, there are limits to multicore scaling too. As it continues, power constraints will prevent "powering of all cores at their full speed, requiring a fraction of the cores to be powered off at all times" [Esmaeilzadeh et al. 2011] forming the "Utilization Wall" or "Dark Silicon". The challenges posed by "Dark Silicon" could be addressed by heterogeneous architectures where general-purpose cores are augmented by specialized accelerators that offer outstanding performance-per-watt. These heterogeneous architectures have now become the dominant paradigm in extreme-scale high performance computing systems. We cover accelerators in later sections.

Before we survey the models, we trace briefly the roots of modelling power consumption. This would give us hints as to how all these models happened to adopt the dominant approach of predicting power consumptions based on utilizations or activity factors estimated from hardware performance counters. In the late 1990s, architecture-level power models were developed in simulators to estimate power and energy mirroring cycle-level architecture simulation of performance. The Cacti tool [Muralimanohar et al. 2007] originally written to study latencies of caches in detail subsequently provided their dynamic power and leakage power models. In 2000, whole-processor power simulators, SimplePower [Vijaykrishnan et al. 2000] and Wattch [Brooks et al. 2000], appeared. SimplePower focused on in-order pipelined processor and provided detailed dynamic power models of integer ALU and other architectural units; the Wattch tool simulated an out-of-order super-scalar pipeline. While both simulators used analytical methods for modelling power, IBM's PowerTimer [Brooks et al. 2001] used empirical techniques. It predicted power consumption of an architectural unit based on measured power consumption of similar unit in an existing microprocessor and scaling it appropriately taking into account variations in size and design. However, simulators had some drawbacks, chief among them being their speed of exploration and their lack of rapid adaptability and sustainability to fast-changing hardware architecture landscape. An appealing alternative route was allowed by direct physical measurements of power consumption using power meters. Although power meters provided the total power consumption of the system, one major challenge remained as to how to decompose this power into unit-level power consumptions. To address this challenge, the performance monitoring counter (PMC) based approach estimating power consumptions of architectural units based on their activity factors was invented and eventually became the core of dominant models today. The accelerated adoption of this approach was however made possible by the simultaneous provision of hardware performance counters (almost akin to standardization) by all the major hardware vendors.

One of the first models correlating hardware event counters to energy values was developed by [Bellosa 2000]. Their model is based on events such as integer operations, floating-point operations, memory requests due to cache misses, etc. that they believed to strongly correlate with power consumption. To trigger these events, special micro-benchmarks are written and executed for several seconds.

Table III Power and Energy Models. '-' indicates not reported.

Model	Level of Abstraction	Type of Power	Is Energy Predicted?	Accuracy of Power Prediction		Accuracy of Energy Prediction	Implementation Complexity	Portability
				Dynamic	Total			
CPU								
[Bellosa 2000]	Linear Independence	Average	Explicit	-	-	-	1 EM	Yes
[Isci and Martonosi 2003]	Linear Independence	Instantaneous, Average	No	-	-	-	3 EM	No
[Heath et al. 2005]	Linear Independence	Instantaneous	No	-	2.7%	-	1 EW	No
[Economou et al. 2006]	Linear Independence	Instantaneous	No	-	15%	-	1 EW	No
[Lee and Brooks 2006]	Linear Independence	Average	Power × Timing	-	24.5%	-	1 EM	No
[Fan et al. 2007]	Linear Independence	Instantaneous	No	-	-	-	1 EW	No
[Fan et al. 2007]	Non-linear Independence	Instantaneous	No	-	-	-	1 EW	No
[Lewis et al. 2008]	Linear Independence	Instantaneous	Yes	-	-	4%	1 EW	Yes
[Wang et al. 2010]	Non-linear Independence	Instantaneous	No	-	-	-	1 EW	No
[Basmadjian et al. 2011]	Linear Independence	Instantaneous	No	-	9%	-	1 EM	Yes
[Basmadjian and de Meer 2012]	Linear Dependence	Average	No	-	4%	-	1 EM	Yes
[Bertran et al. 2013]	Linear Independence	Average	No	14%	10.15%	-	3 EM	Yes
[Bircher and John 2012]	Non-linear Independence	Instantaneous, Average	No	-	14.1%	-	3 EM	No
GPU								
[Hong 2010]	Linear Independence	Average	Power × Timing	18%	8.94%	-	3 EM	No
[Nagasaka et al. 2010]	Linear Independence	Average	No	-	23%	-	3 EM	Yes

[Chen et al. 2011]	Non-linear Independence	Average	No	12.95%	7.77%	-	3 EM	Yes
[Zhang et al. 2011]	Non-linear Independence	Average	No	-	4.34%	-	3 EM	Yes
[Kasichayanula et al. 2012]	Linear Independence	Average	No	24%	12%	-	1 EM	Yes
[Song et al. 2013]	Non-linear Independence	Average	Power × Timing	-	2.1%	11.02%	3 EM	Yes
[Lim et al. 2014]	Non-linear Independence	Average	Yes	15.14%	12.8%	-	3 EM	Yes
[Wang and Cao 2015]	Non-linear Independence	Average	Power × Timing	-	6%	-	3 EM	Yes
Intel Xeon Phi								
[Shao and Brooks 2013]	Linear Independence	Average	Explicit	-	-	5%	3 EM	No
HPC Applications								
[Bui et al. 2008]	Linear Independence	Average	No	-	-	-	3 EM	Yes
[Subramaniam and Feng 2010]	Linear Independence	Average	Power × Timing	-	5.6%	-	1 EM	No
[Tiwari et al. 2012]	Non-linear Independence	Average	Explicit	-	5.5%	7%	3 EM	Yes
[Lively et al. 2012]	Linear Independence	Average	No	-	4.94%	-	2 EW	Yes
[Gamell et al. 2013]	Linear Independence	Average	Power × Timing	-	-	1.41%	1 EM	Yes
[Diouri et al. 2013]	Linear Independence	Average	Power × Timing	-	-	-6.82%	2 EM	Yes
[Kestor et al. 2013a]	Linear Independence	Average	No	-	10%	-	3 EM	Yes
[Gschwandtner et al. 2014]	Linear Independence	-	Explicit	-	-	15%	2 EW	Yes

During the execution of micro-benchmarks, the power and energy consumptions of the whole system are measured using a multimeter. Their synthetic calibration software called Joule Watcher correlates the events and energy measurements. // [Isci and Martonosi 2003] propose a methodology to determine unit-level power estimates based on hardware performance counters. They select 22 strictly collocated physical units based on an annotated P4 die photo. Their power model is constructed as follows:

$$TotalPower = \sum_{i=1}^{22} Power(C_i) + BasePower \tag{3}$$

where a unit power prediction is based on the following equation:

$$\begin{aligned} Power(C_i) = {} & AccessRate(C_i) \times ArchitecturalScaling(C_i) \\ & \times MaxPower(C_i) + NonGatedClockPower(C_i) + BasePower \end{aligned} \tag{4}$$

The parameter, $BasePower$, is the base power consumption of the Pentium 4 platform used in their experiments. The parameter, ($ArchitecturalScaling(C_i)$), is a conditional clock power factor used to model the non-linear behaviour of some issue logic units. The values of the parameters, $MaxPower(C_i)$ and $NonGatedClockPower(C_i)$, are obtained for each of the units using physical areas on the die and training benchmarks.

[Lee and Brooks 2006] adopt a statistically rigorous approach to derive regression models using PMC to predict power. Their approach is two-pronged. First, they derive a baseline model using various statistical techniques to determine the predictors. After having derived this baseline model, they optimize it further by reformulating it using different sampled observations. They report a median error rate of 4.3% and a maximum error of 24.5%.

A linear model that is based on the utilization of CPU, disk, and network is proposed in [Heath et al. 2005]. The power consumed by a node can be described using the formula:

$$P = C_{base} + C_1 \times U_{CPU} + C_2 \times U_{Disk} + C_3 \times U_{Net} \tag{5}$$

where C_{base} is the base power consumption of a node and the coefficients C_1, C_2, and C_3 for power consumptions of CPU, disk, and network, respectively are determined using several benchmarks. The server cluster used in their experiments is composed of four PCs with Pentium III processors and four blade servers with Celeron processors. The power consumed by it is calculated as the sum of power consumptions of all the hardware resources in it. Several micro-benchmarks are used to stress each resource individually in different ways. Total power measurements and utilization data for CPU, disk, and network are collected using a multimeter to compute the least-squares fit and to determine the coefficients of the model. Reported average and maximum prediction errors of the models were 1.3% and 2.7% respectively.

A more complex power model (Mantis) is proposed in [Economou et al. 2006] relying on the utilization metrics of CPU, disk, and network components and hardware performance counters for memory. Here, the general model can be described as follows:

$$P = C_{base} + C_1 \times U_{CPU} + C_2 \times U_{Mem} + C_3 \times U_{Disk} + C_4 \times U_{Net} \tag{6}$$

where C_{base} is the base power consumption of a node and U_{CPU}, U_{Mem}, U_{Disk}, and U_{Net} are the CPU, memory, disk, and network utilizations respectively. The models were calibrated for two server systems using idle runs and different configurations of Gamut [Moore 2004] to emulate applications with varying resource needs. Several benchmarks (SPECcpu2000, SPECjbb2000, and SPECweb2005 suites) [SPEC 2015] and STREAM benchmark [Stream 2015]) were used in the evaluation.

[Fan et al. 2007] propose a simple linear model that correlates power consumption of a single-core processor with its utilization. In this model, power consumption is modelled as follows:

$$P_{CPU} = P_{base} + (P_{max} - P_{base}) \times (U/100) \tag{7}$$

where P_{base} is the base power consumption of a processor and P_{max} represents the power consumption at maximum utilization. The parameter, U, signifies the utilization of the processor. They refine these models further as follows:

$$P = C_{base} + C_1 \times (2 \times U_{CPU} - U_{CPU}^r) \tag{8}$$

where an empirical term is added to increase the accuracy of the model. The tuning parameter r is obtained during calibration using a model similar to Mantis [Economou et al. 2006].

[Lewis et al. 2008] propose a linear regression energy model and validate it for a dual-core AMD Opteron processor architecture. Their model can be described as follows [Lewis et al. 2008]:

$$E = a \times (E_{CPU} + E_{DRAM}) + b \times E_{em} + c \times E_{Support\_Chipsets} + d \times E_{HDD} \tag{9a}$$

$$E_{HDD} = P_{spin-up} \times t_{su} + P_{read} \times \sum N_r \times t_r + P_{write} \times \sum N_w \times t_w + P_{base} \times t_{base} \tag{9b}$$

$$E_{em} = \sum P_{fan} \times t_{ipmi-slice} + \sum P_{optical} \times t_{optical} \tag{9c}$$

E_{CPU} is the energy consumption of the dual-core AMD Opteron processor. It is predicted from four key contributors. These are traffic on the HyperTransport bus [AMDHT 2001], L2 cache misses, CPU core temperatures, disk read and write bandwidths, and ambient temperatures. E_{mem} is the energy consumed by the DRAM banks. E_{em} is the energu consumption of the cooling fans and optical drives. E_{HDD} is the energy consumption of the disk predicted from four components: 1). $P_{spin-up}$, the power consumption to spin the disk to full rotation and t_{su} is the time taken for the spin-up, 2). P_{read} is the power consumption to read kilobyte of data from the disk, 3). P_{write} is the power consumption to write kilobyte of data to the disk, and 4). P_{idle} is the base power consumption of the disk [Lewis et al. 2008]. They report a worse-case prediction error of 4% for common processor benchmarks.

Energy profiling for applications using CPU and disk activity is the subject in [Kansal and Zhao 2008]. The profiling tool consists of a workload manager that triggers the measurement process and event tracing related to OS operations, such as CPU and disk I/O usage, the event logger for event logging using Windows Xperf [WINXPERF 2015], and the energy profiler that correlates resource usage with the event traces and profiles application energy usage across the various resources.

[Rivoire et al. 2008], [Rivoire 2008] study and compare five full-system real-time power models using a variety of machines and benchmarks. Four of these models are utilization-based whereas the fifth includes CPU PMCs in the model parameter set along with the utilizations of CPU and disk. They report that PMC-based model is the best overall in terms of accuracy since it is able to account for majority of the contributors to system's dynamic power (especially the memory activity). They also question the generality of their PMC-based model since the PMCs used in their model parameter set may not have the same essence across different architectures (Intel, AMD).

A linear model that takes into account CPU utilization and I/O bandwidth is described in [Wang et al. 2010] to predict power consumption of a server. The model is

based on the work in [Kansal and Zhao 2008] and can be described as:

$$P = C_{base} + C_1 \times U_{CPU} + C_2 \times U_{I/O} \tag{10}$$

where C_{base} is the base power consumption of a node, U_{CPU} the CPU utilization of the server, $U_{I/O}$ the I/O bandwidth in MB/sec, and C_1, C_2 the coefficients of the model. A wattsup power meter is used to measure overall power consumption of the servers and to calibrate the model.

[Basmadjian et al. 2011] construct a power model of a server as a summation of power models of its components, the processor (CPU), memory (RAM), fans, and disk (HDD). Based on the model evaluations on tower and blade servers, the authors report maximum prediction error rates of 8% and 9% for tower servers and blade servers respectively. This model is presented in detail in Appendix A.1.

[Bertran et al. 2010] present a power model that provides per-component power breakdown of a multicore CPU. It can be described as follows:

$$P_{total} = \sum_{j=1}^{j=cores} ((\sum_{i=1}^{i=ncomponents} AR_{ij} \times P_i)) + P_{base} \tag{11}$$

The parameter AR_{ij} is the activity ratio of component i in core j. The dynamic power consumption of a component i in core j is given by the product, $AR_{ij} \times P_i$. P_{base} represents the base power consumption of the system. The parameter, $ncomponents$, represents the number of components accommodated in their model. They report that apart from facilitating decomposability (per-component power breakdown), their models are perceptive to power phase changes. Based on the validation of their models using SPECcpu2006 benchmarks [SPEC 2015], they report average prediction errors between [1.89-6]% and a maximum error of 10.15%.

[Bircher and John 2012] propose an iterative modelling procedure to predict power using PMCs. They use PMCs that trickle down from the processor to other subsystems such as CPU, disk, GPU, etc and PMCs that flow inward into the processor such as Direct Memory Access (DMA) and I/O interrupts. The highest error reported is 14.1% for the memory controller and the average error reported is less than 9% per subsystem.

[Basmadjian and de Meer 2012] report that summation of power consumptions of all active cores to derive the total power consumption is inaccurate and take into account resource sharing in their power prediction model for multicore processors. They report a maximum prediction error of 5%. [Wang and Shi 2012] present a energy consumption model using three parameters, which are the concurrency level of the workload, the average power dissipation of a thread, and the total time taken to execute the workload. [Liu et al. 2013] present a method to predict the energy consumption of a task executing in a multicore with several other tasks running simultaneously. Their method use the resource utilization and occupancy of a task to predict its energy consumption. On a 32-core multicore, they report an average and maximum errors of 4% and 9% respectively. [Li 2015] use queueing theory to model the power consumption of multicore processors. They treat a multicore server processor as an M/M/m queueing system with multiple servers.

Several other research efforts use PMC approach to model power consumption. [Gurumurthi et al. 2002] use analytical power models for CPU, memory, and disk in their power simulator called SoftWatt to predict power consumptions of applications and OS services. [Li and John 2003] propose power models for the operating system (OS) based on their observations of strong correlation between instructions per cycle (IPC) and OS routine power. [Singh et al. 2009] develop per-core power models based on multiple linear regression using PMCs. [Powell et al. 2009] use a linear regression model to estimate activity factors and power for a large number of micro-architectural structures

using a small number of PMCs. [Goel et al. 2010] derive per-core power models using PMC values and temperature readings. [Roy et al. 2013] propose an energy model for an algorithm, which expresses the energy for an algorithm as a weighted linear combination of the time complexity of the algorithm and the number of "parallel" accesses to the memory. [Spiliopoulos et al. 2011] propose linear regression models that predict power consumption for any DVFS voltage and frequency combination supported by a computing platform.

[McCullough et al. 2011] 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. [Dargie 2015] 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.

Finally, hardware vendors now provide software interfaces to measure and control power and energy consumption of their processors. However, due to the disparate capabilities and the non-uniformity of these interfaces, there is a real need for standardization to facilitate development of supporting infrastructures and tools. [PowerAPI 2016] is the first large-scale effort in this direction. PowerAPI is an interface for standardizing power and energy measurement and control for wide range of systems spanning desktops and datacenters to large-scale HPC systems.

Some key observations follow from the analysis of these models:

— The models [Economou et al. 2006] and [Basmadjian et al. 2011] have high prediction error accuracy than [Heath et al. 2005] even though these models are more comprehensive by accounting for the power consumption from *RAM*.
— There is a noticeable difference between the power prediction accuracies of the models [Economou et al. 2006] and [Basmadjian et al. 2011], even though they both take into account all the major resource utilizations (*CPU*, *RAM*, *Disk*, *NIC*).
— The model [Basmadjian et al. 2011] has a higher prediction error accuracy than the model [Bircher and John 2012], which employs a complex PMC-based approach to model power consumption of all the architectural units of a CPU.

4. POWER AND ENERGY MODELS FOR GPUS

GPUs are now an integral part of high performance computing systems due to their enormous computational powers and energy efficiency (performance/watt). In a node, the GPU is used as a coprocessor and is connected to a CPU through a PCI-Express (PCIe) bus. Work is offloaded to the GPU from a CPU. Table (III) shows the salient features of the GPU models under the heading "GPU".

[Rofouei et al. 2008] use a linear model to calculate the energy consumption of a GPU in their server from real-time energy measurements. The linear relationship is presented below:

$$E_{gpu} = t_{gpu} \times (P_{avg-gpu} + P_{base-cpu}) + E_{transfer} \tag{12}$$

where $t_{gpu}, P_{avg-gpu}, P_{base-cpu}$, and $E_{transfer}$ denote the time of execution of the application on the GPU, average power consumption of GPU, base power consumption of CPU, and the energy consumption of data transfer between CPU and GPU.

One of the first comprehensive models developed for a GPU architecture was by [Hong 2010]. The GPU power consumption in their prediction model is modelled similar to the PMC-based unit power prediction approach of [Isci and Martonosi 2003]. Their model is presented in detail in Appendix B. In their model, the power consump-

tion is calculated as sum of power consumptions of all the components composing the Streaming Multiprocessor (SM) and GDDR memory. To demonstrate the accuracy of their model, NVIDIA GTX280 GPU is used. They report that the prediction error for the total power consumption is 8.94% and the average energy consumption savings are 10.99%. The main factor hindering the portability of this model is that it requires detailed architectural information and contains a large set of parameters.

[Nagasaka et al. 2010] present a statistical approach that uses GPU performance counters exposed for CUDA applications to predict power consumption of GPU kernels. The GPU used in the experiments was NVIDIA GeForce GTX 285. They report an average error of 4.7% and a maximum error of 23% in prediction of total power consumption.

[Chen et al. 2011] use linear regression tree and random forest methods in their models to predict GPU power consumption. The random forest method is used to select the predictors that are the dominant contributors to the power consumption. The Nvidia GTX 280 GPU was used for the experiments. They report an average percentage error (PE) of 7.77% for the total power consumption.

[Kasichayanula et al. 2012] propose an analytical model to estimate activity factors and power for micro-architectural structures on GPUs. The key difference from the model [Hong 2010] is that only two parameters, execution time and load rate, are used to estimate unit-level dynamic power of a GPU. Their model can be described as follows:

$$Total\_power\_consumption = DynamicPower + BasePower \tag{13a}$$

$$DynamicPower = \sum_{i=1}^{n}(N_{SM,i} \times P_i \times U_i) + B_i \times U_i \tag{13b}$$

where n is the number of architectural components, $N_{SM,i}$ is the number of streaming processors utilizing an architectural component, P_i, B_i, and U_i are the dynamic power consumption, base power consumption, and utilization of the architectural component i respectively. A unit is defined as an architectural component such as a floating-point unit (FP), shared memory (Shared), or global memory (GlobalMem). A micro-benchmark is designed for each architectural unit stressing the unit to obtain its P_i and B_i values. The utilization rates U_i of the units are obtained from the application execution. The Fermi C2075 GPU and MAGMA kernels are used for model evaluation in their experiments. The maximum error in predictions for the total power consumption is reported to be 12%.

[Song et al. 2013] propose power and energy prediction models that employ a configurable, back-propagation, artificial neural network (BP-ANN). The parameters of the BP-ANN model are ten carefully selected PMCs of a GPU. The values of these PMCs are obtained using the CUDA Profiling Tools Interface (CUPTI) [CUPTI 2015] during the application execution. Their energy model for a GPU-based cluster can be described as follows [Song et al. 2013]:

$$\forall l, m, l \in \Psi, m \in \Gamma, E = \sum_{l=1}^{\Psi}[(\sum_{m=1}^{\Gamma} \bar{P_{l,m}} \times t_{gpu}) + \bar{P}_{base} \times t_{gpu} + E_{parallel-overhead}] \tag{14a}$$

$$t_{gpu} = t_{pci} + t_{kernel} \tag{14b}$$

Ψ represents the set of hybrid identical nodes where each node has multicore processors and multi-GPUs. The set of GPUs in a node is denoted by Γ. The parameter, t_{gpu}, represents the execution time of the application on the GPU cluster. $\bar{P_{lm}}$ denotes the average dynamic power consumption of the mth GPU on node l. This is predicted using

BP-ANN using the PMCs. $\bar{P}_{base}$ is the average base power consumption of the whole system on node l without the GPU. The total execution time of the application, t_{gpu}, is the sum of PCIe data transfer time, t_{pci}, and the kernel execution time, t_{kernel}. The parameter, t_{pci}, is calculated as the size of the data transferred via the PCIe link divided by the bandwidth of the PCIe link. The parameter t_{kernel} is predicted using a performance model. $E_{parallel-overhead}$ represents the energy consumption from parallel overhead due to network communications in the cluster. A NVIDIA Tesla C2075 is used for model validation and analysis. The authors [Song et al. 2013] report an average prediction error rate of 2.1% for their power model and maximum prediction error percentage of 11% for their energy model.

[Marowka 2013] present analytical models for studying the energy consumption of various architectural design choices for hybrid CPU-GPU chips. Their model for performance per watt for an asymmetric processor follows [Marowka 2013]:

$$\frac{Perf}{W_a} = \frac{1}{P_s + P_c + P_g} \tag{15a}$$

$$P_s = (1-f)(1 + (c-1) \times k_c + g \times w_g \times k_g) \tag{15b}$$

$$P_c = \frac{\alpha \times f}{c} \times (c + g \times w_g \times k_g) \tag{15c}$$

$$P_g = \frac{(1-\alpha) \times f}{g \times \beta} \times (g \times w_g + c \times k_c) \tag{15d}$$

c is equal to the total number of CPU cores and g is equal to the total number of GPU cores. α represents fraction of program's execution time spent in parallel execution on the CPU cores. β is the GPU core's performance normalized to that of the CPU core. f represents is the execution time of the fraction of program that can be parallelized. P_s represents the power consumption during the sequential computation phase. It is equal to power of 1 consumed by one active CPU core plus remaining $c-1$ CPU idle cores consuming a fraction of power, k_c, plus g idle GPU cores consuming fraction of power, $k_g \times w_g$. P_c represents parallel computation on CPU cores only. P_g denotes parallel computation on GPU-cores only [Marowka 2013].

[Lim et al. 2014] enhance McPAT [Li et al. 2013] to write a power model for GPUs. In their power model, the total average power is calculated as a sum of the power consumptions of all the components. It can be described as follows:

$$\begin{aligned} TotalPower = \sum_{i=0}^{n} P\_Component_i = a \times P_{SP\_fpu\_dyn} + b \times P_{SP\_fpu\_lkg} + \\ c \times P_{SP\_alu\_dyn} + d \times P_{SP\_alu\_lkg} + e \times P_{ConstMem} + P_{Others} \end{aligned} \tag{16}$$

where a, b, c, d, and e are scaling parameters. NVIDIAs Fermi architecture is used for building the power model. They report average prediction errors of 7.7% and 12.8% for the micro-benchmarks (used to build the model) and Merge benchmarks [Linderman et al. 2008] respectively.

[Wang and Cao 2015] use the technique of program slicing to model GPU power consumption. The source code of an application is decomposed into slices and these slices are used as basic units to train a power model based on fuzzy wavelet artificial neural networks (FWNN). So, unlike earlier research efforts which use PMCs, slicing features are extracted from the programs and used in their model. They use three GPUs for evaluation of their model but prediction error rates are not reported.

[Ma et al. 2009] use support vector regression (SVR) to predict the power consumption of a GPU based on workload signals. They choose five major workload signals

representing the runtime utilizations of major pipeline stages in a NVIDIA GeForce 8800gt GPU. [Li et al. 2011] present power and performance prediction models to identify an energy-efficient consolidation of workloads. [Xie et al. 2012] build a power model based on native instructions to analyse and estimate the power consumption at an architecture level. With proper profilers and tools, they identify major power contributors in GPU architecture. These contributors are divided into two groups, i.e. computing unit and memory access on which different native instruction were run and measured separately as a foundation for their energy prediction model.

AMD GPUs are also used in many HPC systems. For example: three systems in Top500 list ([Top500 2015] and two in Green500 [Green500 2015] use AMD FirePro GPUs (the AMD Firepro Server S9150 also topped the November 2014 Green500 list). However, power and energy models for these GPUs are abysmally lacking.

[Zhang et al. 2011] employ a rigorous statistical model to predict power consumption of GPGPU applications executing on an ATI Radeon HD5870 GPU. They use a Random Forest method to correlate the execution characteristics and the power consumption of the GPU. They report a median absolute error of 4.34% for total power consumption. [Zhao et al. 2013] examine the power breakdown using McPAT [Li et al. 2013] of different components of NVIDIA and AMD GPUs such as cores and caches, memory controllers, and off-chip memory. Based on experiments on AMD Radeon HD7970, they report that the off-chip DRAM accesses consume a significant portion (32 %) of the total system power. To reduce the memory power consumption without compromising the memory bandwidth, they scale down the supply voltage and frequency of memory interface.

One observation from the analysis of these models is that even though both [Song et al. 2013] and [Wang and Cao 2015] use sophisticated artificial neural network (ANN) methods, there is a remarkable difference in their reported power prediction error accuracies.

5. POWER AND ENERGY MODELS FOR XEON PHI AND FPGA

In this section, we cover the other accelerators that are used in high performance computing systems.

Xeon Phi [XEONPHI 2015] is the competing offering from Intel in the accelerator space based on Intel's Many Integrated Core Architecture or Intel MIC. The Top 500 (November 2015) list contains 29 supercomputers using Xeon Phi. This list includes the No.1 supercomputer Tianhe-2, which uses Xeon Phi 31S1P containing 57 cores running at 1.1GHz. The current list (June 2016) has 23 supercomputers using Xeon Phi with Tianhe2 moving to position No.2. Table (III) shows the salient features of the models for Xeon Phi under the heading "Intel Xeon Phi". We found just one energy prediction model for this accelerator even though it appropriates an appreciable share in the Top500 supercomputers. [Shao and Brooks 2013] construct an instruction-level energy model of a Xeon Phi processor and report an accuracy between 1% and 5% for real world applications.

FPGAs (Field Programmable Gate Arrays) are another acceleration technology that holds immense promise for high performance computing platforms. They have created the Reconfigurable Computing (RC) market segment, which exploited the inherent parallelism and reconfigurability provided by them to "hardware accelerate" software algorithms. High-Performance Reconfigurable Computing (HPRC) [HPRC 2015] brought Reconfigurable Computing into the high-performance computing sphere by combining FPGAs with multicore CPUs. In a node, the FPGA is used as a coprocessor and is connected to a CPU through a PCI-Express (PCIe) bus.

[Mittal and Vetter 2015a] report that for several applications, FPGAs have demonstrated better performance and energy efficiency than CPUs and GPUs. Table (IV)

Table IV Power/Energy measurement infrastructures for the HPC applications on FPGAs

Model	Power/Energy Measurement Infrastructure
[Thomas et al. 2009]	Data sheet TDPs ([TDP 2015]) for FPGA, GPU, and CPU
[Lange et al. 2009]	Xilinx XPower Estimator for FPGA [XPE 2015]. For CPU and GPU, power measured indirectly between mains and Host PC power supply
[Hamada et al. 2009]	No details of how power is measured
[Kestur et al. 2010]	Wattsup? Pro power meter
[Betkaoui et al. 2010]	Olson remote power monitoring meter for the whole system
[Hussain et al. 2011]	Data sheet TDP for GPU. No details of how power is measured for FPGA and CPU
[Schryver et al. 2011]	Xilinx XPower Estimator for FPGA [XPE 2015]. No details of how power is measured for GPU and CPU
[Duan et al. 2011]	Fluke Norma 4000 Power Analyzer for the whole system
[Van Essen et al. 2012]	Xilinx XPower Estimator 13.2 for FPGA [XPE 2015], Data sheet TDPs for GPU and CPU
[Birk et al. 2012]	Xilinx Power Estimater for FPGA [XPE 2015], data sheet TDP for GPU.
[Zou et al. 2012]	PC diagnostics software utility EVEREST for CPU. For FPGAs and GPUs, a pincer galvanometer (equipment type HIOKI3290)
[Benkrid et al. 2012]	Power meter for the whole system
[Pauwels et al. 2012]	No details of how power is measured
[Fowers et al. 2013]	Monitoring a wattmeter

summarizes the power/energy measurement infrastructures used in the studies that have compared the energy efficiency of FPGAs to CPUs and GPUs [Mittal and Vetter 2015a]. All of these works use direct physical measurements and none use models; some use power estimators for FPGAs provided by the vendors; some however do not specify how power consumptions were measured.

To the best of our knowledge, there are no linear regression models using PMCs because PMCs are not yet offered by FPGAs. [Ou and Prasanna 2004] construct a linear energy prediction model based on instruction level energy profiling.[Poon et al. 2005] present a hardware-level model, which uses a placement and routing CAD tool and in-depth knowledge of FPGA architecture. [Wang et al. 2006] propose a linear component-based model to predict energy consumption of a reconfigurable Multiprocessors-on-a-Programmable-Chip (MPoPCs) implemented on Xilinx FPGAs. [Al-Khatib and Abdi 2015] propose a linear instruction-level model to predict dynamic energy consumption for soft processors in FPGA. The model considers both inter-instruction effects and the operand values of the instructions.

6. ANALYTICAL ENERGY MODELS

[Demmel et al. 2013] prove that a region of strong scaling in energy exists for matrix multiplication and N-body problem. That is, they show that for a given problem size n, the energy consumption remains constant as the number of processors p increases and the runtime decreases proportionally to p.

[Choi et al. 2013] present an energy roofline model based on the time-based roofline model [Williams et al. 2009]. Their models for time and energy can be described as follows:

$$\begin{aligned} T &= max(W \times \tau_{flop}, Q \times \tau_{mem}) \\ &= W \times \tau_{flop} \times max(1, \frac{B_\tau}{I}) \end{aligned} \tag{17}$$

T denotes the running time of the algorithm (under the assumption of perfect overlap between computations and memory operations). W and Q are the number of arithmetic and memory operations respectively. I is the algorithm's computational intensity. τ_{flop} and τ_{mem} respectively are the time per flop and time per mop. B_τ is called the time-balance point. Their energy model is presented below [Choi et al. 2013]:

$$\begin{aligned} E &= W \times \epsilon_{flop} + Q \times \epsilon_{mem}) + \pi_0 \times T \\ &= W \times \hat{\epsilon}_{flop} \times (1 + \frac{\hat{B}_\epsilon(I)}{I}) \end{aligned} \tag{18}$$

ϵ_{flop} and ϵ_{mem} respectively are the energy per flop and energy per mop. $\hat{B}_\epsilon(I)$ is called the effective energy balance. They conclude that the balance gap ($\frac{B_\epsilon}{B_\tau}$) represents the difficulty in achieving energy efficiency compared to time efficiency. They validate their model on on a Intel multicore CPU and a NVIDIA Fermi GPU. They use PowerMon2 apparatus [Bedard et al. 2010] for power and energy measurements.

[Choi et al. 2014] extend the roofline model by adding parameters such as power caps, memory hierarchy access costs, and measurement of random memory access patterns. They conduct a microbenchmarking study of time, energy, and power of computation and memory access for over dozen diverse platforms, which include x86, ARM, GPU, and hybrid (AMD APU, SoC) processors. They conclude that constant power (or base power) is a critical limiting factor accounting for about 50% of total power in 7 out of 12 platforms evaluated. Based on these conclusions, they recommend further tighter integration of non-processor and non-memory components.

[Drozdowski et al. 2014] visualize models of energy consumption of computations as two-dimensional maps similar to isotherms or isobars in weather maps. They call these models, isoenergy maps which represent points of equal energy consumption in a multi-dimensional space of system and application parameters. They use isoenergy maps to study energy-performance trade-offs. They present isoenergy maps for three models of parallel computations, Amdahl's law, Gustafsons laws [Gustafson 1988], and divisible loads representing data-parallel computations [Bharadwaj et al. 2003].

[Marszalkowski et al. 2016] analyze the impact of memory hierarchies on time-energy trade-off in parallel computations. They formalize non-linear dependence of execution time and energy on problem size. They use this formalization in their multi-objective optimization problem of minimizing time and energy in parallel processing of divisible loads. The total energy consumed is computed as sum of energy consumed by the originator (initiator or resource allocator) and the energies consumed by the slaves in idle, starting up, networking, and running states.

7. POWER AND ENERGY MODELS IN HIGH PERFORMANCE COMPUTING APPLICATIONS

In this section, we present studies for saving power and energy in HPC applications. Previous sections dwelt on power and energy models for dominant components in a node that predicted power and energy consumptions for all kinds of applications executing on these components. Our focus in this section is application-specific and our purpose for surveying these studies is manifold.

— Study how power and energy consumptions were measured (either via direct measurements or models) for a node and for the whole cluster.
— Study how a power/energy model for a cluster is composed from the power/energy models of nodes.
— Compare the prediction error accuracy of application-specific models to application-agnostic (i.e., full-system or the component-specific) models.

Table (III) summarizes these studies under the heading "HPC Applications". Studies not shown in the table use direct physical measurements.

[Kamil et al. 2008] is a pioneering effort that studied power consumption of applications executing on large-scale HPC systems. They conclude that power consumption of HPL benchmark is a good representative predictor for the power consumption of a HPC workload and that power consumption of a large-scale system can be accurately predicted from power measurements of smaller subsets of the system. Their study was conducted on stand-alone systems and a large-scale NERSC Cray XT4 system. Using results of a single cabinet in the system, they model the system's power usage. The power consumption of a single rack is extrapolated linearly to the 102 racks that compose the HPC system. Their model is described as follows:

$$\begin{aligned} DCWatts_{system} &= 102 \times DCWatts_{rack} + 50KW \\ &= 102 \times DCAmps_{rack} \times Volts_{rack} + 50KW \end{aligned} \tag{19}$$

where 50KW is the power consumed by the disk subsystem.

[Bui et al. 2008] propose a power model using PMCs to predict power consumption of parallel scientific applications running on modern multicore/multiprocessor systems. They modify the power model of [Isci and Martonosi 2003] to model power consumption for a modern Intel Itanium2 processor. For the architecture scaling factor (a parameter modelled in [Isci and Martonosi 2003] using component area ratios), transistor counts are used as initial values. The power for each component is modelled as a linear function of access rates of its architectural units. The total power consumed by a processor is calculated as the sum of power consumptions of its components and its base power. The total power consumption of a multicore system is then calculated as the sum of power consumptions of all the cores/processors that are contained in it.

One of the most popular frameworks used for fine-grained power and energy profiling on parallel and distributed systems is the PowerPack framework [Ge et al. 2010]. This toolkit contains both hardware and software components. An AC power meter (Watt's Up Pro) is used to measure the power draw from the wall and DC power data acquisition devices (Analog Input Module NI 9205NI and cDAQ chassis NI cDAQ9172) are used to obtain component-level power consumptions inside a node through precise instrumentation of all its power rails. The software components provide facilities for synchronized collection of the measurement data from various data streams and analysis of the data. The toolkit is used to measure the power and energy consumption of one node at a time. To obtain total power consumption of a whole cluster, a node remapping approach [Ge et al. 2010] is used.

[Subramaniam and Feng 2010] use multiple linear regression to model the power consumption of the High Performance Linpack (HPL) benchmark [HPL 2008]. Out of the 18 HPL parameters, they select four parameters (problem size, block size, number of process rows, number of process columns) for the regression modelling. To evaluate their model, they perform experiments using 64 nodes from a cluster called SystemG [SYSTEMG 2015]. The power and energy values are obtained using "Watts UP? Pro E" power meter. The power consumption of the cluster is predicted by extrapolating the prediction of power consumption of a single node.

[Dongarra et al. 2012] studied energy consumptions of high performance dense linear algebra libraries LAPACK [LAPACK 2013] and PLASMA [PLASMA 2015] using PowerPack [Ge et al. 2010] and Intel RAPL API [Rotem et al. 2012]. They conclude that, for the applications using these libraries, RAPL API is a good alternative to power meters based on near identical power measurements observed between PowerPack and RAPL.

[Tiwari et al. 2012] develop CPU and DIMM power and energy models using artificial neural networks. They study three important HPC kernels, matrix multiplication, stencil computation, and LU factorization. To derive component-level power measurements, they use the PowerMon2 apparatus [Bedard et al. 2010]. They report an absolute error rate of 5.5% for the total power consumption and energy usage predictions for the three kernels.

[Lively et al. 2012] and [Lively et al. 2014] propose application-centric predictive models for power consumption. For each kernel in an application, multivariate linear regression models for system power, CPU power, and memory power are constructed using PAPI performance events [PAPI 2015] as predictors. To train and validate their model, they conduct experiments in a power-aware cluster called SystemG [SYSTEMG 2015]. They validate the models using four hybrid scientific applications and report a maximum prediction error percentage of 4.93%.

[Kestor et al. 2013a] propose a per-core power model based on a regression analysis of core activity. [Ltaief et al. 2012], [Bosilca et al. 2014] also compare the power consumptions of two high performance dense linear algebra libraries i.e., LAPACK and PLASMA. Their results indicate that PLASMA outperforms LAPACK both in performance as well as energy efficiency.

[Witkowski et al. 2013], [Jarus et al. 2014] propose system-wide power prediction models for HPC servers based on performance counters. They cluster real-life HPC applications into groups and create specialized power models for them. They then use decision trees to select an appropriate model for the current system load. They report average prediction error of 4% based on experiments on four servers, one AMD and three Intel.

[Gschwandtner et al. 2014] present linear regression models based on hardware counters for prediction of energy consumption of HPC applications executing on IBM POWER7 processor. Instead of micro-benchmarks, they use NAS parallel benchmarks to train the models. They pick a small subset from 500 different hardware counters offered by the POWER7 processor. They report a maximum prediction error of 15%.

There are exhaustive studies primarily focusing on the performance analysis of Xeon Phi but very few on power/energy models. [Li et al. 2014] present a detailed study of the performance-energy trade-offs of the Xeon Phi architecture. They propose extensions to PowerPack infrastructure [Ge et al. 2010] to measure component-wise power and energy consumptions of systems hosting accelerators. The energy efficiency of Xeon Phi 5110P is compared to a Tesla Fermi c2050 GPGPU and the results are found to be mixed.

[Lastovetsky and Manumachu 2017] present an application-level energy model where the dynamic energy consumption of a processor is represented by a function of problem size. Unlike PMC-based models that contain hardware-related PMCs and do not consider problem size as a parameter, this model takes into account highly nonlinear and non-convex nature of the relationship between energy consumption and problem size for solving optimization problems of data-parallel applications on homogeneous multicore clusters for energy.

7.1. Interconnects and Communications

We now present research works that have studied power and energy consumptions of HPC interconnects and communication algorithms in HPC systems and applications.

[Murugan et al. 2013] explore power savings in HPC interconnects. They present a power-aware scheduling method in a HPC interconnect where the nodes are interconnected by a 3D torus topology. They show that 30-40% power savings by switching elements of the interconnect to low-power modes without any degradation in performance.

[Diouri et al. 2013] present energy prediction model for MPI broadcast algorithms in large scale HPC systems. They present models for four broadcast algorithms, Scatter and AllGather algorithm (MPI/SAG) [Thakur and Gropp 2003] used in MPICH2 [MPICH 2016], Pipelining algorithm (MPI/Pipeline) provided in OpenMPI 1.4.4, hybrid broadcasting algorithm which combines MPI/SAG and OpenMP, and a hybrid algorithm, which combines MPI/Pipeline and OpenMP [OpenMPI 2016]. They model the energy consumption of a broadcast operation as sum of energy consumptions of nodes and switches involved in the operation. Their model is presented in detail in Appendix C. They validate their energy prediction model on Grid5000 [Cappello et al. 2005] and report a worst prediction error of -6.82%.

[Gamell et al. 2013] explore energy and performance trade-offs of data movement and I/O at extreme scales. Their energy consumption prediction model is presented in detail in Appendix C. It is calculated based on summation of energy consumptions of CPU, DRAM, NIC, and other miscellaneous components. The energy consumption model of a MPI communication operation is calculated as sum of energy consumptions of nodes involved in the operation.

On-chip interconnect now contributes to more than 30% of on-chip power consumption. Strategies such as DVFS [DVFS 2015], on/off control for link power management, and reduction of activity factor on the interconnect have been used to improve power efficiency of interconnect links. [Shang et al. 2003] dynamically adjust frequency and voltage of links to minimize the power consumption of on-chip interconnect. They report a maximum of 6.3x power savings (and 4.3x on average), which is accompanied by moderate increase in performance (15.2% increase in average latency and 2.5% decrease in throughput). [Soteriou and Peh 2007] propose self-regulating power-aware interconnection networks, which turn on/off links based on traffic. They report a reduction in power consumption of 54.4% accompanied by a modest increase in network latency. [Jin et al. 2008] propose a data compression technique in on-chip network architectures that dynamically tracks value patterns in cache traffic and that dynamically applies compression depending on the workload. Based on experiments using a suite of scientific and commercial benchmarks for a 16-core tiled CMP, they report energy savings of 36% while at the same time improving latency by 31%.

7.2. Summary

Our goal in this section was to compare prediction accuracies of power and energy consumptions of HPC applications and the prediction accuracies of application-agnostic power and energy models. We expected application-specific predictive power and energy models to have higher prediction accuracy but our survey shows results to the contrary.

8. RELATED SURVEYS

In this section, we present recent surveys summarizing the power and energy efficiency techniques employed in high performance computing systems and applications. Our survey differs from these surveys by focusing exclusively on power and en-

ergy models in high performance systems and applications whereas the other surveys record all the research efforts covering the entire spectrum of power and energy efficiency, which includes (along with power and energy models) power management techniques such as DVFS (Dynamic Voltage and Frequency Scaling) [DVFS 2015] and AVS (Adaptive Voltage Scaling) [AVS 2015]. The goal of our survey is to understand how power and energy consumption are predicted in these research efforts from the viewpoint of node architecture.

[Kaxiras and Martonosi 2008] survey the most important architectural techniques that have been proposed to reduce both static and dynamic power consumptions in processors and memory hierarchies.

[Benedict 2012] present a survey of various energy measurement methodologies for HPC, Grid, and Cloud applications. They classify methodologies as follows: a). Hardware-based, b). Software-based, and c) Hybrid. In hardware-based energy measurements, they survey the various hardware approaches used such as power meters, iPDUs, datasheets, in-built sensors, and external sensors. In the software-based energy measurements, they survey energy prediction models. And in the hybrid approach, they look at online-based, offline-based, and profile-based methods. They also present a list of energy monitoring tools available such as pTop, PowerTop, IntelPCM, PowerPack, Likwid, and lmsensors and summarize their overhead, portability, and usability of these tools.

[Mobius et al. 2014] 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 Dantas 2014] present a literature survey of works using workload characterization for performance and energy efficiency improvement in HPC, cloud, and big data environments. They report a remarkable increase in research papers proposing energy modelling and energy efficiency techniques from 2009 to 2013 thereby suggesting an increasing importance of energy saving techniques in the HPC, cloud, and big data environments.

[Orgerie et al. 2014] survey techniques for improving the energy efficiency of computing and networking resources in large-scale distributed systems. [Tan et al. 2014] survey the research on saving power and energy for HPC linear algebra applications. They separate the surveyed efforts into two categories: 1) Power management in HPC systems and 2) Power and energy efficient HPC applications (Cholesky, LU, QR). 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 Vetter 2015a] present a survey of Heterogeneous Computing Techniques (HCTs) that enable utilizing both CPUs and GPUs to improve energy efficiency. They classify the research efforts in techniques for saving energy in Heterogeneous Computing Systems (HCSs) as follows: "intelligent workload partitioning and performance improvement, DVFS on both CPU and GPU, DVFS on CPU only, resource scaling or low-power modes". They also present energy efficiency evaluation of fused CPU-FPU chips compared to discrete systems (where GPUs have separate memory spaces from the CPU).

[Mittal and Vetter 2015b] 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. Their classification categories are: "DVFS, CPU-GPU workload division, architectural techniques for specific GPU components such as caches, techniques that

exploit workload variation to dynamically allocate resources, application-specific and programming-level techniques for power analysis and management."

[Dayarathna et al. 2016] 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. While this survey gives an impressive overview from the data-center point of view, our survey is mainly focused from the viewpoint of node architecture.

9. CASE STUDY OF PERFORMANCE MONITORING COUNTER BASED MODELS

From the survey, we find that the most dominant approach employs linear regression using PMC as parameters to model the power/energy consumption of a node. To study the accuracy of this approach for a modern node architecture, we build power and energy models using it on a Intel Haswell platform with the specification shown in Table VI. Likwid API [Haswell 2013] is used to get the PMC values of an application execution. 35 PMCs from 13 performance groups are selected from the total of 390 supported PMCs (Table XII in Appendix E.1). Some groups share some of the selected PMCs. To train the model, applications (Table V) from NAS serial and OpenMP benchmarks [NAS 2015], Rodinia OpenMP benchmarks [Rodinia 2015], STREAM benchmark [Stream 2015], and BLAS double-precision benchmarks [BLAS 2015] are used. Benchmark classes (S, W, A) were used in the NAS benchmark suite. The total number of data points used for training is 187. For the NAS OpenMP and Rodinia OpenMP benchmarks, the number of OMP threads executed is 4, which is equal to the number of physical cores in the Intel Haswell platform. For all the other applications, the number of threads in the application is set to 1.

Since PMC values differ for different runs of the same application, we follow the methodology described below to make sure the experimental results are reliable:

— The Intel Haswell platform is fully reserved and dedicated to these experiments during their execution. We also made certain that there are no drastic fluctuations in the load due to abnormal events in the platform by monitoring its load continuously for a week using the tool *sar*. Insignificant variation in the load was observed during this monitoring period suggesting normal and clean behavior of the platform.
— When an application is executed, it is bound to the physical cores using the *numactl* tool.
— To obtain a data point, the application is repeatedly executed until the sample mean lies in the 95% confidence interval and a precision of 0.025 (2.5%) has been achieved. For this purpose, Students t-test is used assuming that the individual observations are independent and their population follows the normal distribution. We verify the validity of these assumptions by plotting the distributions of observations. To be more precise, for each data point, the application is repeatedly run until one of the following three conditions is satisfied:
  (1) The input maximum number of repetitions have been exceeded.
  (2) The sample mean falls in the input confidence interval of 95% (or the precision of 0.025 has been achieved).
  (3) The elapsed time of the application runs has exceeded the maximum time allowed (3600 seconds).

  Each application is always run for a minimum number of repetitions. The input minimum and maximum number of repetitions differ based on the application or the problem size solved. For small problem sizes (NAS benchmarks class S, W), these values are set to 10000, and 100000 respectively. For medium problem sizes (NAS benchmarks class A), these values are set to 100 and 1000. For large problem sizes (NAS benchmarks class B, C, D), these values are set to 5 and 50. For each data

Table V Applications used for the training set

Benchmark Suite	**Applications**
NPB SER [NAS 2015]	BT.SER.α, CG.SER.α, DC.SER.α, EP.SER.α, FT.SER.α, IS.SER.α, LU.SER.α, MG.SER.α, SP.SER.α, UA.SER.α where α = S, W, A **[NAS 2015]**
NPB OpenMP [NAS 2015]	BT.OMP.α, CG.OMP.α, DC.OMP.α, EP.OMP.α, FT.OMP.α, IS.OMP.α, LU.OMP.α, MG.OMP.α, SP.OMP.α, UA.OMP.α where α = S, W, A **[NAS 2015]**
Rodinia OpenMP [Rodinia 2015]	bfs, heartwall, kmeans, leukocyte, nn, particlefilter, srad, backprop, cfd, hotspot, lavaMD, lud, nw, pathfinder, streamcluster **[Rodinia 2015]**
Stream [Stream 2015]	Stream C
BLAS [BLAS 2015]	daxpy, dgemv, dgemm **[BLAS 2015]**

point, we select the sample mean as the output only when the input precision of 0.025 (2.5%) has been reached. If the precision of measurement is not achieved before the maximum number of repeats have been completed, we increase the number of repetitions and also the maximum elapsed time allowed. However, we observed that condition (2) is always satisfied before the other two in our experiments.

The PMC values, power and energy consumptions, and execution times are obtained separately using three separate programs. This is to isolate the overhead (constant but low) in collecting PMC values using likwid API. The energy consumptions at the socket and DRAM level are obtained using the RAPL PMC values (PWR_PKG_ENERGY:PWR0, PWR_DRAM_ENERGY:PWR3) of the performance group, ENERGY. The average dynamic power consumption during the execution of an application is obtained by dividing the total energy consumption (given by the PMC values) by the total execution time of the application (using *gettimeofday* timer). The power and energy models are built using the GSL multiple linear regression API [GSLMLR 2015]. The main steps of an application execution in the training and validation set are initializing Likwid runtime, starting of monitoring PMC, executing the benchmark code, stopping the monitoring of PMC, querying for values of the PMCs, and finalizing the Likwid runtime.

The model took two weeks (*implementation complexity* of 2 EW) to implement by a senior post-doctoral researcher who has vast experience in software programming. The implementation tasks involved studying Likwid API [Haswell 2013], short-listing the PMC to use in the modelling which includes using rigorous statistical techniques to prune the PMC set, studying GNU scientific library linear regression API [GSLMLR 2015], short-listing the applications (or benchmark test suites) to use in the training and validation sets, writing scripts to automate the model building and validation process, and cleaning the validation data, for example, removing outliers since GSL multiple linear regression (MLR) is quite sensitive to them. The models took 6 hours of experimental time to build. Figure 2 (Appendix E.3) shows the execution times of applications in the training set that are used to calculate their average dynamic power consumptions. It also shows that we have used applications with a wide range of execution times to build the models. This would allow us to determine the true capability of a model to predict accurately the power and energy consumptions for applications with a wide range of execution times.

Table VI Specification of the Intel Haswell workstation used to build the PMC-based power and energy models.

Technical Specifications	**Intel Haswell i5-4590**
Processor	Intel(R) Core(TM) i5-4590 3.3 GHz
Microarchitecture	Haswell
Memory	8 GB
Socket(s)	1
Core(s) per socket	4
L1d cache	32 KB
L11 cache	32 KB
L2 cache	256 KB
L3 cache	6144 KB
TDP	84 W
Base Power	22.3 W
Max Turbo Frequency	3.7 GHz

Table VII Prediction error percentages for NAS Serial and OpenMP Applications (Benchmark Class W)

Benchmark	**Avg. Dynamic Power Prediction Error % (std. dev.)**	**Energy Prediction Error % (std. dev.)**
BT.SER.W	98.716442 (0.10)	235.52 (1.50)
CG.SER.W	96.66 (0.41)	171.02 (5.90)
DC.SER.W	99.76 (0.01)	97.30 (0.20)
EP.SER.W	99.92 (0.01)	99.39 (0.15)
FT.SER.W	99.24 (0.34)	100.34 (4.91)
IS.SER.W	130.57 (18.66)	40.94 (6.79)
LU.SER.W	50.19 (4.5)	542.05 (64.37)
MG.SER.W	98.21 (0.07)	6.70 (1.06)
SP.SER.W	99.43 (0.03)	72.08 (0.50)
UA.SER.W	79.74 (0.94)	12.86 (13.51)
BT.OMP.W	98.68 (0.17)	10.67 (2.52)
CG.OMP.W	100.00 (0.00)	99.99 (0.00)
DC.OMP.W	92.09 (2.33)	85.55 (13.39)
EP.OMP.W	92.96 (0.87)	10.52 (2.53)
FT.OMP.W	99.41 (0.23)	112.76 (3.33)
IS.OMP.W	64.66 (8.48)	31.83 (1.22)
LU.OMP.W	81.96 (3.22)	626.71 (46.07)
MG.OMP.W	99.12 (0.07)	48.53 (1.06)
SP.OMP.W	99.25 (0.64)	561.84 (9.28)
UA.OMP.W	88.29 (1.18)	15.81 (16.90)
daxpy	99.27 (0.19)	45.41 (2.84)
dgemv	89.00 (1.29)	39.87 (18.46)
dgemm	83.53 (1.83)	77.14 (26.16)

One interesting observation from our results is that some PMCs have negative coefficients (Table XIII in Appendix E.2). For example, consider the PMC, L2_RQSTS_MISS. It represents how often it was necessary to get cachelines from memory and therefore

it should only increase the average dynamic power consumption. However, this coefficient is negative for the power model but positive for the energy model. Same applies for PMC, ICACHE_MISSES. Now, at this point, this set of PMC can be pruned further to remove the parameters with negative coefficients or collinearity between parameters using rigorous statistical techniques ([Lee and Brooks 2006]) or detailed iterative methodologies ([Bircher and John 2012], [Bertran et al. 2013]).

To validate the models, we use them to predict power and energy consumptions of NAS serial and OpenMP benchmarks (class W) and BLAS applications. The (minimum, average, maximum) prediction error percentages of average dynamic power and total energy consumptions are (50, 93, 130) and (6, 136, 626) respectively (Table VII with standard deviations). The prediction error percentage for a model is calculated using the actual value provided by Likwid API and the fitted value provided by the GSL MLR function using the model.

During an application execution in the training and validation set, DVFS is disabled by pre-setting the frequencies of CPUs to base frequency using "userspace" [CPUFreq 2015] governor. However, in a real-life situation, dynamic power management (DPM) is not disabled and the default governor is "ondemand" [CPUFreq 2015]. Due to DPM (and runtime DVFS), there can be fluctuations in power consumption. Therefore, we would like to question the rationale behind use of PMCs, which are accumulated for a whole application execution, to model instantaneous dynamic power. Even if PMCs are used to model power for small but distinct phases of an application, the significant overhead of acquiring these PMC values and building phase-level models prohibits their use for this case. For example, CUPTI [CUPTI 2015], due to its design limitation, allows querying for only one PMC (or event) of a NVIDIA GPU platform in a single CUDA kernel invocation [CUD 2015]. One must execute a CUDA kernel many number of times to get the values of all the desired PMCs.

10. DISCUSSION

In this section, we present a summary of all the power/energy models that we have surveyed. Our prominent observations for each of the model characteristics follow:

(1) *Level of Abstraction*:
   — There exists no model today that truly and comprehensively captures the highly heterogeneous and hierarchical architecture of a node illustrated in Figure (1). That is, no model exists that satisfies the property of *Non-linear Independence*.
   — Many models for a node have the property of *Linear Independence*. Such models are constructed by summation of models of the components.

(2) *Type of power*:
   — There are very few models that predict instantaneous power of an application.
   — We expect the models that predict instantaneous power accurately to maintain their accuracy for a wide range of problem sizes and execution times.
   — However, we are sceptical of models that are used for predicting average total power consumption for applications running for long durations (hours to days) to accurately predict power for a wide range of problem sizes and execution times.

(3) *Decomposition*:
   — Many models focus exclusively on modelling power consumption of either the CPU or an accelerator in a node. They can be further classified into:
      (a) Models that adopt linear regression methods using performance-monitoring counters (PMCs) as the parameters.
      (b) Models that adopt non-linear methods such as artificial neural networks, etc. using PMCs as the parameters.
   — Models using PMC employ iterative methodologies or methods such as Principal Component Analysis (PCA), Random Forest etc. to prune the set of PMCs.

— Very few models take into account contention for shared resources between components (for ex: cores sharing last-level cache).
— Very few models study power consumption of communication links in a node such as
  (a) On-chip interconnect between CPUs.
  (b) PCIe bus connecting the multicore CPUs and accelerators.

(4) *Accuracy of power prediction*:
— Almost all the models report prediction error for only the total average power consumption but not the dynamic power consumption. We would expect the prediction errors for dynamic power to be higher.
— There are many models that do not mention clearly the proportion of dynamic to static power consumptions in the total power consumption as can be seen from the empty entries for *dynamic power* in the tables for the models.

(5) *Implementation Complexity*: The implementation complexity for the models employing linear regression or ANNs using the PMC as parameters is quite high. This is due to several reasons:
— The first step in these models is to design and write a micro-benchmark test suite for a component. This test suite contains benchmarks that stress the various architectural units of the component to create a model of the power consumption of the architectural unit as a function of its access rate (activity factor) or its PMC set. To the best of our knowledge, there are no standard micro-benchmark suites. So, if one were to implement one of these models, one has to write a micro-benchmark test suite. We believe that writing micro-benchmarks is a very complex, tedious, and error-prone task. One key design requirement of the test suite is to stress a single architectural unit avoiding completely or minimizing interaction with other architectural units. This is required so that its activity can be decoupled to derive its contribution to the total power consumption. If there is interaction, the micro-benchmark stressing an architectural unit must ensure that the utilization is constant for the other architectural units during its execution. So, this interaction must somehow be removed or accounted in the model for the architectural unit to avoid collinearity problems when multiple linear regression is applied later. This, we believe, is not easy to accomplish. Micro-benchmark test suites are usually written in assembly language. So, the model implementer must now become an expert in assembly language to write a reliable micro-benchmark test suite.
— [Wu et al. 2006], [Bertran et al. 2010], [Molka et al. 2010], [Kestor et al. 2013b], [Pandiyan and Wu 2014] describe in great detail this complex process of writing a micro-benchmark test suite. [Bertran et al. 2010] have designed 97 micro-benchmarks for their power model.
— So, after a micro-benchmark test suite is written, each of the micro-benchmarks must be run for a sufficiently long time to ensure that the power consumption reading (using a power meter) is stable. However, there is no straightforward method to separate the static and dynamic components of this power consumption. The access rates or the PMC values for the architectural unit are also obtained separately.
— Then, the power consumption of an architectural unit is modelled as a function of its PMC. Today, each architectural unit comes with a large set of PMC. For example: Haswell architecture [Haswell 2013] has 390 events. So a question arises as to which events to select from this set of PMCs. Models include some form of Principal Component Analysis (PCA) to prune this set.
— Some models use complex ANNs such as Random Forest (RF), Back-Propagation (BP), Fuzzy Wavelet (FW), etc. to incorporate non-linearity or dependence be-

tween the PMCs. Therefore, implementation of these models necessitates the model writer to possess or acquire knowledge of these methods.

(6) *Portability*:
- — Many models ensure their portability to next-gen processors in the same architecture space by using the PMC approach.
- — However, the PMC approach hinders their generality or applicability to all architectures since PMCs are architecture-specific.

(7) *Scalability*:
- — The standard approach to determine the total power/energy consumption of a large-scale HPC system is based on the following steps:
  - (a) Measure the power/energy consumption of a node (either using direct measurements or power/energy models). Multiply this power/energy consumption by the total number of nodes in the system.
  - (b) Or use a node remapping technique. In this technique, the number of measurements is equal to the number of nodes and each time the node that is measured is mapped to a different physical node. The total power/energy consumption is equal to the sum of all the measurements taken.

PMC-based approach is frequently used to model unit-level power consumptions of a nodal component to predict the total power consumption of it. However, we question its continuing use to predict power consumption of a nodal component or a node due to two reasons. Firstly, it has high implementation complexity. In most cases (For example: Likwid, CUPTI), the values of all the performance events can not be obtained during a single execution of an application. Secondly, while values of some PMCs can be determined from static analysis of the source code of an application, values of most PMCs are gathered during the application execution. Therefore, an application must be executed to get the values of PMCs, which are then input to a PMC-based model to get the predicted power/energy consumption. However, there are software libraries available today on all mainstream commodity processors providing interfaces to determine power consumption at component level during an application execution via in-built power meters or models. The PAPI library ([PAPI 2015], [Weaver et al. 2012]) provides API for energy consumption. For Intel's CPU processors, the library uses the "Running Average Power Limit" (RAPL) component [David et al. 2010], which uses a software model to predict energy. Intel PCM (Performance Counter Monitor) [IntelPCM 2012] is a tool to monitor performance hardware counters on Intel processors, similar to PAPI. The difference between PCM and PAPI is that PCM supports only Intel hardware, but it can monitor also uncore metrics, like memory controllers and QuickPath Interconnect links. The Intel PCM utility *pcm-power* displays energy usage and thermal headroom for CPU and DRAM sockets. For Intel's Xeon PHI processor, the MPSS *MicMgmt* library [IntelMPSS 2014] is used to get the instantaneous power consumption from the on-chip power meters. NVIDIA Management Library (NVML) API [NVML 2011] can be used to determine the power consumption of the NVIDIA GPUs from the on-chip power meter. Although there is some overhead introduced by these library calls, it can be minimized by choosing appropriate sampling frequency. Therefore, when one can get the power/energy consumption from vendor-specific software libraries (that provide access to readings from all on-chip power meters at low sampling frequency), we question the necessity of using performance events for modelling power/energy consumption.

11. CONCLUSION

This survey presented a classification of predictive power and energy models for the major components at the node architecture level in modern HPC computing platforms.

One overarching conclusion can be made from the survey. During the era of single-core processors, models were able to accurately predict the dynamic power of the full-system by having parameters that accurately but separately modelled dynamic power consumption of all components in the system. But now such an approach will be erroneous. Unless the inherent complexities (contention for shared resources, dynamic power management, etc.) of the modern node architectures are methodically taken into account, models aspiring to predict power/energy consumptions for these architectures will be inaccurate.

Power models have been the main research focus for HPC applications and outnumber energy models. There are very few studies that directly model energy. Studies that do not directly model energy predict it using a power model and a timing model (or by expliciting measuring execution time). However, the prediction error in accuracy of energy consumption in these studies is compounded by the errors in accuracies of its constituent models (power and timing). Therefore, we would like to ask why not model energy consumption directly for HPC applications instead of constructing its constituent models. Power models and power management algorithms are necessary for system designers to ensure that an application execution does not exceed the power/thermal constraints of a system. For example, Intel RAPL [David et al. 2010] allows the power consumption of an application to exceed the TDP for short periods of time but monitors the power consumption closely to keep it close to an average limit by controlling frequency. Many research efforts propose power models and use them to find inefficiencies in the system and thereby provide suggestions to designers to improve the system architecture. However, if the goal of energy efficiency of HPC applications is to minimize the total energy consumption without sacrificing performance, one can strive to accomplish it by directly modelling energy consumption.

In the case of accelerators, we found that models for NVIDIA GPUs are predominant. Even though Intel Xeon Phi holds a notable portion (6%) of Top500 list [Top500 2015], very few studies have modelled its power/energy consumptions. Considering that FPGAs are now becoming a reckonable force in the HPC space, there exist no studies dedicated to their power/energy models for HPC applications.

REFERENCES

2015. Compute Unified Device Architecture. (2015). http://www.nvidia.com/object/cuda_home_new.html

2015. Energy Efficiency in Data Centers. (2015). http://www.google.co.in/about/datacenters/efficiency/

ACPI. 2015. Advanced Configuration and Power Interface Specification, Version 6.0. (2015). http://www.uefi.org/sites/default/files/resources/ACPI_6.0.pdf

Zaid Al-Khatib and Samar Abdi. 2015. Operand-Value-Based Modeling of Dynamic Energy Consumption of Soft Processors in FPGA. In *International Symposium on Applied Reconfigurable Computing*. Springer, 65–76.

AMDHT. 2001. HyperTransport. (2001). https://en.wikipedia.org/wiki/HyperTransport

AVS. 2015. Adaptive voltage scaling. (2015). https://en.wikipedia.org/wiki/Adaptive_voltage_scaling

Luiz André Barroso and Urs Hölzle. 2007. The case for energy-proportional computing. *Computer* 12 (2007), 33–37.

Robert Basmadjian, Nasir Ali, Florian Niedermeier, Hermann de Meer, and Giovanni Giuliani. 2011. A methodology to predict the power consumption of servers in data centres. In *2nd International Conference on Energy-Efficient Computing and Networking*. ACM.

R. Basmadjian and H. de Meer. 2012. Evaluating and modeling power consumption of multi-core processors. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*. 1–10.

Daniel Bedard, Min Yeol Lim, Robert Fowler, and Allan Porterfield. 2010. PowerMon: Fine-Grained and Integrated Power Monitoring for Commodity Computer Systems. In *Proceedings Southeastcon 2010*. IEEE.

Frank Bellosa. 2000. 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.

Shajulin Benedict. 2012. Review: Energy-aware Performance Analysis Methodologies for HPC architectures-An Exploratory Study. *J. Netw. Comput. Appl.* 35, 6 (Nov. 2012).

Khaled Benkrid, Ali Akoglu, Cheng Ling, Yang Song, Ying Liu, and Xiang Tian. 2012. High Performance Biological Pairwise Sequence Alignment: FPGA Versus GPU Versus Cell BE Versus GPP. *Int. J. Reconfig. Comput.* 2012, Article 7 (Jan. 2012).

Ramon Bertran, Marc Gonzalez, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. 2010. Decomposable and Responsive Power Models for Multicore Processors Using Performance Counters. In *Proceedings of the 24th ACM International Conference on Supercomputing (ICS '10)*. ACM, 147–158.

Ramon Bertran, Marc Gonzalez Tallada, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. 2013. A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs. *IEEE Trans. Comput.* 62, 7 (July 2013), 1289–1302.

Brahim Betkaoui, David B Thomas, and Wayne Luk. 2010. Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing. In *Field-Programmable Technology (FPT), 2010 International Conference on*. IEEE, 94–101.

Veeravalli Bharadwaj, Debasish Ghose, and Thomas G. Robertazzi. 2003. Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. *Cluster Computing* 6, 1 (Jan. 2003).

William Lloyd Bircher and Lizy K John. 2012. Complete System Power Estimation Using Processor Performance Events. *IEEE Trans. Comput.* 61, 4 (April 2012), 563–577.

Matthias Birk, Matthias Balzer, Nicole Ruiter, and Jurgen Becker. 2012. Comparison of processing performance and architectural efficiency metrics for FPGAs and GPUs in 3D ultrasound computer tomography. In *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*. IEEE, 1–7.

BLAS. 2015. BLAS (Basic Linear Algebra Subprograms). (2015). http://www.netlib.org/blas/

George Bosilca, Hatem Ltaief, and Jack Dongarra. 2014. Power profiling of Cholesky and QR factorizations on distributed memory systems. *Computer Science-Research and Development* 29, 2 (2014), 139–147.

David Brooks, Margaret Martonosi, John-David Wellman, and Pradip Bose. 2001. Power-Performance Modeling and Tradeoff Analysis for a High End Microprocessor. In *Proceedings of the First International Workshop on Power-Aware Computer Systems-Revised Papers (PACS '00)*. Springer-Verlag.

David Brooks, Vivek Tiwari, and Margaret Martonosi. 2000. Wattch: A Framework for Architectural-level Power Analysis and Optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA '00)*. ACM.

Van Bui, Boyana Norris, Kevin Huck, Lois Curfman McInnes, Li Li, Oscar Hernandez, and Barbara Chapman. 2008. A Component Infrastructure for Performance and Power Modeling of Parallel Scientific Applications. In *Proceedings of the 2008 compFrame/HPC-GECO Workshop on Component Based High Performance (CBHPC '08)*. ACM, Article 6, 11 pages.

F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. 2005. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.*

Jianmin Chen, Bin Li, Ying Zhang, Lu Peng, and Jih-Kwon Peir. 2011. Statistical GPU Power Analysis Using Tree-based Methods. In *International Green Computing Conference and Workshops (IGCC)*. IEEE.

Jee Choi, Marat Dukhan, Xing Liu, and Richard Vuduc. 2014. Algorithmic time, energy, and power on candidate HPC compute building blocks. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 447–457.

Jee Whan Choi, Daniel Bedard, Robert Fowler, and Richard Vuduc. 2013. A roofline model of energy. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*. IEEE, 661–672.

CPUFreq. 2015. CPU frequency scaling - ondemand Governor. (2015). https://wiki.archlinux.org/index.php/CPU_frequency_scaling

CUPTI. 2015. CUDA Profiling Tools Interface. (2015). https://developer.nvidia.com/cuda-profiling-tools-interface

Waltenegus Dargie. 2015. A Stochastic Model for Estimating the Power Consumption of a Processor. *IEEE Trans. Comput.* 64, 5 (2015).

Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: memory power estimation and capping. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*. IEEE, 189–194.

Miyuru Dayarathna, Yonggang Wen, and Rui Fan. 2016. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 732–794.

J. Demmel, A. Gearhart, B. Lipshitz, and O. Schwartz. 2013. Perfect Strong Scaling Using No Additional Energy. In *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*.

Dennard. 1974. Dennard scaling. (1974). https://en.wikipedia.org/wiki/Dennard_scaling

M. E. M. Diouri, O. Glück, J.-C. Mignot, and L. Lefèvre. 2013. Energy Estimation for MPI Broadcasting Algorithms in Large Scale HPC Systems. In *Proceedings of the 20th European MPI Users' Group Meeting (EuroMPI '13)*. ACM.

DOE. 2010. The Opportunities and Challenges of Exascale Computing. (2010). http://science.energy.gov/~/media/ascr//pdf/reports/Exascale_subcommittee_report.pdf

J. Dongarra, H. Ltaief, P. Luszczek, and V. Weaver. 2012. Energy Footprint of Advanced Dense Numerical Linear Algebra using Tile Algorithms on Multicore Architecture. In *The 2nd International Conference on Cloud and Green Computing*.

Maciej Drozdowski, Jedrzej M Marszalkowski, and Jakub Marszalkowski. 2014. Energy trade-offs analysis using equal-energy maps. *Future Generation Computer Systems* 36 (2014), 311–321.

B. Duan, W. Wang, X. Li, C. Zhang, P. Zhang, and N. Sun. 2011. Floating-point mixed-radix FFT core generation for FPGA and comparison with GPU and CPU. In *Field-Programmable Technology (FPT), 2011 International Conference on*.

DVFS. 2015. Dynamic voltage scaling. (2015). https://en.wikipedia.org/wiki/Dynamic_voltage_scaling

Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, and Partha Ranganathan. 2006. Full-system power analysis and modeling for server environments. In *In Proceedings of Workshop on Modeling, Benchmarking, and Simulation*. 70–77.

Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark Silicon and the End of Multicore Scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA '11)*. ACM, 365–376.

Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. In *34th Annual International Symposium on Computer architecture*. ACM, 13–23.

Naila Farooqui, Andrew Kerr, Gregory Diamos, S. Yalamanchili, and K. Schwan. 2011. A Framework for Dynamically Instrumenting GPU Compute Applications Within GPU Ocelot. In *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units (GPGPU-4)*. ACM, Article 9.

Jeremy Fowers, Greg Brown, John Wernsing, and Greg Stitt. 2013. A Performance and Energy Comparison of Convolution on GPUs, FPGAs, and Multicore Processors. *ACM Trans. Archit. Code Optim.* 9, 4, Article 25 (Jan. 2013).

Marc Gamell, Ivan Rodero, Manish Parashar, Janine C. Bennett, Hemanth Kolla, Jacqueline Chen, Peer-Timo Bremer, Aaditya G. Landge, Attila Gyulassy, Patrick McCormick, Scott Pakin, Valerio Pascucci, and Scott Klasky. 2013. Exploring Power Behaviors and Trade-offs of In-situ Data Analytics. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13)*. ACM.

Rong Ge, Xizhou Feng, Shuaiwen Song, Hung-Ching Chang, Dong Li, and Kirk W Cameron. 2010. Powerpack: Energy profiling and analysis of high-performance systems and applications. *Parallel and Distributed Systems, IEEE Transactions on* 21, 5 (2010), 658–671.

GK210. 2014. An Overview of Kepler GK110 and GK210 Architecture. (2014). http://international.download.nvidia.com/pdf/kepler/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf

Bhavishya Goel, Sally A. McKee, Roberto Gioiosa, Karan Singh, Major Bhadauria, and Marco Cesati. 2010. Portable, Scalable, per-Core Power Estimation for Intelligent Resource Management. Green Computing Conference, 2010 International.

Green500. 2015. The Green500 List - November 2015. (2015). http://www.green500.org/news/green500-list-november-2015

Philipp Gschwandtner, Michael Knobloch, Bastian Mohr, Dirk Pleiter, and Thomas Fahringer. 2014. Modeling cpu energy consumption of hpc applications on the ibm power7. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*. IEEE, 536–543.

GSLMLR. 2015. Multi-parameter fitting. (2015). https://www.gnu.org/software/gsl/manual/html_node/Multi_002dparameter-fitting.html

Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, N. Vijaykrishnan, Mahmut Kandemir, Tao Li, and Lizy Kurian John. 2002. Using Complete Machine Simulation for Software Power Estima-

tion: The SoftWatt Approach. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture (HPCA '02)*. IEEE Computer Society.

John L. Gustafson. 1988. Reevaluating Amdahl's Law. *Commun. ACM* 31, 5 (May 1988).

Tsuyoshi Hamada, Khaled Benkrid, Keigo Nitadori, and Makoto Taiji. 2009. A Comparative Study on ASIC, FPGAs, GPUs and General Purpose Processors in the Gravitational N-body Simulation. In *Proceedings of the 2009 NASA/ESA Conference on Adaptive Hardware and Systems (AHS '09)*. IEEE Computer Society, 447–452.

Haswell. 2013. Performance Monitoring Counters for Haswell Architecture. (2013). https://code.google.com/p/likwid/wiki/Haswell

Taliver Heath, Bruno Diniz, Belo Horizonte, Enrique V Carrera, and Ricardo Bianchini. 2005. Energy Conservation in Heterogeneous Server Clusters. In *10th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP)*. ACM, 186–195.

Hyesoon Hong, Sunpyand Kim. 2010. An Integrated GPU Power and Performance Model. *SIGARCH Comput. Archit. News* 38, 3 (June 2010), 280–289.

HPL. 2008. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. (2008). http://www.netlib.org/benchmark/hpl/

HPRC. 2015. High-Performance Reconfigurable Computing. (2015). http://www.chrec.org/

Hanaa M Hussain, Khaled Benkrid, Ahmet T Erdogan, and Huseyin Seker. 2011. Highly parameterized k-means clustering on FPGAs: Comparative results with GPPs and GPUs. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*. IEEE, 475–480.

Eduardo Camilo Inacio and Mario A. R. Dantas. 2014. A Survey into Performance and Energy Efficiency in HPC, Cloud and Big Data Environments. *Int. J. Netw. Virtual Organ.* 14, 4 (March 2014), 299–318.

IntelMPSS. 2014. Intel Manycore Platform Software Stack (Intel MPSS). (2014). https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss

IntelPCM. 2012. Intel Performance Counter Monitor - A better way to measure CPU utilization. (2012). https://software.intel.com/en-us/articles/intel-performance-counter-monitor

Canturk Isci and Margaret Martonosi. 2003. Runtime power monitoring in high-end processors: Methodology and empirical data. In *36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 93.

M. Jarus, A. Oleksiak, T. Piontek, and J. Wglarz. 2014. Runtime power usage estimation of HPC servers for various classes of real-life applications. *Future Generation Computer Systems* 36 (2014).

Yuho Jin, Ki Hwan Yum, and Eun Jung Kim. 2008. Adaptive Data Compression for High-performance Low-power On-chip Networks. In *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 41)*. IEEE Computer Society, 354–363.

Shoaib Kamil, John Shalf, and Erich Strohmaier. 2008. Power efficiency in high performance computing. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 1–8.

Aman Kansal and Feng Zhao. 2008. Fine-grained energy profiling for power-aware application design. *ACM SIGMETRICS Performance Evaluation Review* 36, 2 (Aug. 2008), 26.

Kiran Kasichayanula, Dan Terpstra, Piotr Luszczek, Stan Tomov, Shirley Moore, and Gregory D. Peterson. 2012. Power Aware Computing on GPUs. In *Symposium on Application Accelerators in High Performance Computing (SAAHPC)*. IEEE Computer Society.

Stefanos Kaxiras and Margaret Martonosi. 2008. *Computer Architecture Techniques for Power-Efficiency* (1st ed.). Morgan and Claypool Publishers.

Gokcen Kestor, Roberto Gioiosa, Darren J. Kerbyson, and Adolfy Hoisie. 2013a. Enabling Accurate Power Profiling of HPC Applications on Exascale Systems. In *Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers (ROSS '13)*. ACM.

Gokcen Kestor, Roberto Gioiosa, Darren J Kerbyson, and Adolfy Hoisie. 2013b. Quantifying the energy cost of data movement in scientific applications. In *2013 IEEE international symposium on workload characterization (IISWC)*.

Srinidhi Kestur, John D. Davis, and Oliver Williams. 2010. BLAS Comparison on FPGA, CPU and GPU. In *Proceedings of the 2010 IEEE Annual Symposium on VLSI (ISVLSI '10)*. IEEE Computer Society, 288–293.

Holger Lange, Florian Stock, Andreas Koch, and Dietmar Hildenbrand. 2009. Acceleration and Energy Efficiency of a Geometric Algebra Computation Using Reconfigurable Computers and GPUs. In *Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM '09)*. IEEE Computer Society, 255–258.

LAPACK. 2013. Linear Algebra Package. (2013). http://http://www.netlib.org/lapack/

Alexey Lastovetsky and Ravi Reddy Manumachu. 2017. 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.

Benjamin C. Lee and David M. Brooks. 2006. Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction. *SIGARCH Comput. Archit. News* 34, 5 (Oct. 2006), 185–194.

Adam Lewis, Soumik Ghosh, and N.-F. Tzeng. 2008. Run-time Energy Consumption Estimation Based on Workload in Server Systems. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower'08)*. USENIX Association.

Bo Li, Hung-Ching Chang, Shuaiwen Song, Chun-Yi Su, Timmy Meyer, John Mooring, and Kirk W Cameron. 2014. The power-performance tradeoffs of the Intel Xeon Phi on HPC applications. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 1448–1456.

Dong Li, Surendra Byna, and Srimat Chakradhar. 2011. Energy-Aware Workload Consolidation on GPU. In *Proceedings of the 2011 40th International Conference on Parallel Processing Workshops (ICPPW '11)*. IEEE Computer Society, 389–398.

Keqin Li. 2015. Optimal Partitioning of a Multicore Server Processor. *J. Supercomput.* 71, 10 (Oct. 2015).

Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2013. The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing. *ACM Trans. Archit. Code Optim.* 10, 1, Article 5 (April 2013).

Tao Li and Lizy Kurian John. 2003. Run-time Modeling and Estimation of Operating System Power Consumption. *SIGMETRICS Perform. Eval. Rev.* 31, 1 (June 2003), 160–171.

Jieun Lim, Nagesh B. Lakshminarayana, Hyesoon Kim, William Song, Sudhakar Yalamanchili, and Wonyong Sung. 2014. Power Modeling for GPU Architectures Using McPAT. *ACM Trans. Des. Autom. Electron. Syst.* 19, 3, Article 26 (June 2014), 24 pages.

Michael D. Linderman, Jamison D. Collins, Hong Wang, and Teresa H. Meng. 2008. Merge: A Programming Model for Heterogeneous Multi-core Systems. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XIII)*. ACM.

Qixiao Liu, Miquel Moreto, Victor Jimenez, Jaume Abella, Francisco J. Cazorla, and Mateo Valero. 2013. Hardware Support for Accurate Per-task Energy Metering in Multicore Systems. *ACM Trans. Archit. Code Optim.* 10, 4 (Dec. 2013).

Charles Lively, Valerie Taylor, Xingfu Wu, Hung-Ching Chang, Chun-Yi Su, Kirk Cameron, Shirley Moore, and Dan Terpstra. 2014. E-AMOM: an energy-aware modeling and optimization methodology for scientific applications. *Computer Science-Research and Development* 29, 3-4 (2014), 197–210.

Charles Lively, Xingfu Wu, Valerie Taylor, Shirley Moore, Hung-Ching Chang, Chun-Yi Su, and Kirk Cameron. 2012. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. *Computer Science-Research and Development* 27, 4 (2012), 245–253.

Hatem Ltaief, Piotr Luszczek, and Jack Dongarra. 2012. Profiling High Performance Dense Linear Algebra Algorithms on Multicore Architectures for Power and Energy Efficiency. *Comput. Sci.* 27, 4 (Nov. 2012), 277–287.

Xiaohan Ma, Mian Dong, Lin Zhong, and Zhigang Deng. 2009. Statistical power consumption analysis and modeling for GPU-based computing. In *Proceeding of ACM SOSP Workshop on Power Aware Computing and Systems (HotPower)*.

Olli Mämmelä, Mikko Majanen, Robert Basmadjian, Hermann De Meer, Andrè Giesler, and Willi Homberg. 2012. Energy-aware job scheduler for high-performance computing. *Computer Science - Research and Development* 27, 4 (2012).

Ami Marowka. 2013. Analytical Modeling of Energy Efficiency in Heterogeneous Processors. *Comput. Electr. Eng.* 39, 8 (Nov. 2013).

Jedrzej M. Marszalkowski, Maciej Drozdowski, and Jakub Marszalkowski. 2016. Time and Energy Performance of Parallel Systems with Hierarchical Memory. *Journal of Grid Computing* 14, 1 (2016), 153–170.

John C. McCullough, Yuvraj Agarwal, Jaideep Chandrashekar, Sathyanarayan Kuppuswamy, Alex C. Snoeren, and Rajesh K. Gupta. 2011. Evaluating the Effectiveness of Model-based Power Characterization. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'11)*. USENIX Association.

Sparsh Mittal and Jeffrey S. Vetter. 2015a. A Survey of CPU-GPU Heterogeneous Computing Techniques. *ACM Computing Surveys (CSUR)* 47, 4 (July 2015).

Sparsh Mittal and Jeffrey S. Vetter. 2015b. A Survey of Methods for Analyzing and Improving GPU Energy Efficiency. *ACM Computing Surveys (CSUR)* 47, 2 (Jan. 2015).

Christoph Mobius, Waltenegus Dargie, and Alexander Schill. 2014. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (2014).

Daniel Molka, Daniel Hackenberg, Robert Schöne, and Matthias S Müller. 2010. Characterizing the energy consumption of data transfers and arithmetic operations on x86- 64 processors. In *Green Computing Conference, 2010 International*. IEEE, 123–133.

Moore. 1965. Moore's Law. (1965). https://en.wikipedia.org/wiki/Moore's_law

J Moore. 2004. Gamut: Generic Application EMULaTOR. (2004).

MPICH. 2016. MPICH - High performance portable MPI. (2016). http://www.mpich.org/

Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. 2007. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 40)*. IEEE Computer Society.

Muthukumar Murugan, David Hung Chang Du, and Krishna Kant. 2013. On the interconnect energy efficiency of high end computing systems. *Sustainable Computing: Informatics and Systems* 3, 2 (2013), 49 – 57.

Hitoshi Nagasaka, Naoya Maruyama, Akira Nukada, Toshio Endo, and Satoshi Matsuoka. 2010. Statistical power modeling of GPU kernels using performance counters. In *International Green Computing Conference and Workshops (IGCC)*. IEEE.

NAS. 2015. NAS Parallel Benchmarks. (2015). https://www.nas.nasa.gov/publications/npb.html

NVML. 2011. NVIDIA Management Library (NVML). (2011). https://developer.nvidia.com/nvidia-management-library-nvml

OpenMPI. 2016. OpenMPI - Open source high performance computing. (2016). https://www.open-mpi.org/

Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. 2014. A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems. *ACM Comput. Surv.* 46, 4, Article 47 (March 2014), 47:1–47:31 pages.

Jingzhao Ou and V. K. Prasanna. 2004. Rapid energy estimation of computations on FPGA based soft processors. In *SOC Conference, 2004. Proceedings. IEEE International*.

D. Pandiyan and C. J. Wu. 2014. Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms. In *Workload Characterization (IISWC), 2014 IEEE International Symposium on*. 171–180.

PAPI. 2015. Performance Application Programming Interface 5.4.1. (2015). http://icl.cs.utk.edu/papi/

Karl Pauwels, Matteo Tomasi, Javier Diaz, Eduardo Ros, and Marc M. Van Hulle. 2012. A Comparison of FPGA and GPU for Real-Time Phase-Based Optical Flow, Stereo, and Local Image Features. *IEEE Trans. Comput.* 61, 7 (2012), 999–1012.

PCIE. 2003. Peripheral Component Interconnect Express. (2003). https://en.wikipedia.org/wiki/PCI_Express

PLASMA. 2015. Parallel Linear Algebra for Scalable Multi-core Architectures. (2015). http://http://icl.cs.utk.edu/plasma/

Kara K. W. Poon, Steven J. E. Wilton, and Andy Yan. 2005. A Detailed Power Model for Field-programmable Gate Arrays. *ACM Trans. Des. Autom. Electron. Syst.* 10, 2 (April 2005), 279–302.

M. D. Powell, A. Biswas, J. S. Emer, S. S. Mukherjee, B. R. Sheikh, and S. Yardi. 2009. CAMP: A technique to estimate per-structure power at run-time using a few simple parameters. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. 289–300.

PowerAPI. 2016. Sandia National Laboratories: High Performance Computing Power Application Programming Interface (API) Specification. (2016). http://powerapi.sandia.gov/

QPI. 2008. Intel QuickPath Interconnect. (2008). https://en.wikipedia.org/wiki/Intel_QuickPath_Interconnect

Suzanne Rivoire. 2008. Models and Metrics for Energy-Efficient Computer Systems. PhD Thesis. Stanford University, Stanford, California.

Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. 2008. A Comparison of High-level Full-system Power Models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower'08)*. USENIX Association.

Rodinia. 2015. The Rodinia Benchmark Suite, version 3.0. (2015). https://www.cs.virginia.edu/~skadron/wiki/rodinia/index.php/Rodinia:Accelerating_Compute-Intensive_Applications_with_Accelerators

Mahsan Rofouei, Thanos Stathopoulos, Sebi Ryffel, William Kaiser, and Majid Sarrafzadeh. 2008. Energy-aware High Performance Computing with Graphic Processing Units. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower'08)*. USENIX Association.

Efraim Rotem, Alon Naveh, Avinash Ananthakrishnan, Eliezer Weissmann, and Doron Rajwan. 2012. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE Micro* 32, 2 (March 2012), 20–27.

Swapnoneel Roy, Atri Rudra, and Akshat Verma. 2013. An Energy Complexity Model for Algorithms. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS '13)*. ACM, 283–304.

Christian de Schryver, Ivan Shcherbakov, Frank Kienle, Norbert Wehn, Henning Marxen, Anton Kostiuk, and Ralf Korn. 2011. An Energy Efficient FPGA Accelerator for Monte Carlo Option Pricing with the Heston Model. In *Proceedings of the 2011 International Conference on Reconfigurable Computing and FPGAs (RECONFIG '11)*. IEEE Computer Society, 468–474.

Li Shang, Li-Shiuan Peh, and N. K. Jha. 2003. Dynamic voltage scaling with links for power optimization of interconnection networks. In *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*.

Yakun Sophia Shao and David Brooks. 2013. Energy Characterization and Instruction-level Energy Model of Intel's Xeon Phi Processor. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design (ISLPED '13)*. IEEE Press, 389–394.

Karan Singh, Major Bhadauria, and Sally A. McKee. 2009. Real Time Power Estimation and Thread Scheduling via Performance Counters. *SIGARCH Comput. Archit. News* 37, 2 (July 2009), 46–55.

Shuaiwen Song, Chunyi Su, Barry Rountree, and Kirk W Cameron. 2013. A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures. In *27th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 673–686.

Vassos Soteriou and Li-Shiuan Peh. 2007. Exploring the design space of self-regulating power-aware on/off interconnection networks. *IEEE Transactions on Parallel and Distributed Systems* 18, 3 (2007), 393–408.

SPEC. 2015. SPEC's Benchmarks. (2015). https://www.spec.org/benchmarks.html

Vasileios Spiliopoulos, Stefanos Kaxiras, and Georgios Keramidas. 2011. Green governors: A framework for continuously adaptive dvfs. In *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 1–8.

Stream. 2015. STREAM: Sustainable Memory Bandwidth in High Performance Computers. (2015). https://www.cs.virginia.edu/stream/

Balaji Subramaniam and Wu-chun Feng. 2010. Statistical Power and Performance Modeling for Optimizing the Energy Efficiency of Scientific Computing. In *Proceedings of the 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM '10)*. IEEE Computer Society, 139–146.

SYSTEMG. 2015. System G cluster. (2015). https://www.cs.vt.edu/facilities/systemg

Li Tan, Shashank Kothapalli, Longxiang Chen, Omar Hussaini, Ryan Bissiri, and Zizhong Chen. 2014. A survey of power and energy efficient techniques for high performance numerical linear algebra operations. *Parallel Comput.* 40, 10 (Dec. 2014), 559–573.

TDP. 2015. Thermal design power. (2015). https://en.wikipedia.org/wiki/Thermal_design_power

Rajeev Thakur and William D Gropp. 2003. Improving the performance of collective operations in MPICH. In *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*. Springer.

David Barrie Thomas, Lee Howes, and Wayne Luk. 2009. A Comparison of CPUs, GPUs, FPGAs, and Massively Parallel Processor Arrays for Random Number Generation. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '09)*. ACM, 63–72.

Ananta Tiwari, Michael Laurenzano, Laura Carrington, and Allan Snavely. 2012. Modeling power and energy usage of hpc kernels. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*. IEEE, 990–998.

Top500. 2015. Top 500. The List - November 2015. (2015). http://top500.org/lists/2015/11

Brian Van Essen, Chris Macaraeg, Maya Gokhale, and Ryan Prenger. 2012. Accelerating a Random Forest Classifier: Multi-Core, GP-GPU, or FPGA?. In *Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines (FCCM '12)*. IEEE Computer Society, 232–239.

N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye. 2000. Energy-driven Integrated Hardware-software Optimizations Using SimplePower. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA '00)*. ACM.

Haifeng Wang and Yunpeng Cao. 2015. Predicting power consumption of GPUs with fuzzy wavelet neural networks. *Parallel Comput.* 44 (May 2015), 18–36.

Hongyi Wang, Qingfeng Jing, Rishan Chen, Bingsheng He, Zhengping Qian, and Lidong Zhou. 2010. Distributed systems meet economics: pricing in the cloud. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. USENIX Association.

Shinan Wang and Weisong Shi. 2012. Cpt: An energy efficiency model for multi-core computer systems. *Wayne State University, Tech. Rep. MIST-TR-2012-008* (2012).

X. Wang, S. G. Ziavras, and J. Hu. 2006. System-Level Energy Modeling for Heterogeneous Reconfigurable Chip Multiprocessors. In *2006 International Conference on Computer Design*.

Vincent M Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore. 2012. Measuring energy and power with PAPI. In *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*. IEEE, 262–268.

Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (April 2009).

WINXPERF. 2015. Windows Performance Toolkit Technical Reference. (2015). https://msdn.microsoft.com/en-us/library/windows/hardware/hh162945.aspx

M. Witkowski, A. Oleksiak, T. Piontek, and J. Weglarz. 2013. Practical Power Consumption Estimation for Real Life HPC Applications. *Future Gener. Comput. Syst.* 29, 1 (Jan. 2013).

Wei Wu, Lingling Jin, Jun Yang, Pu Liu, and Sheldon X.-D. Tan. 2006. A Systematic Method for Functional Unit Power Estimation in Microprocessors. In *Proceedings of the 43rd Annual Design Automation Conference (DAC '06)*. ACM, 554–557.

XEONPHI. 2015. Intel Many Integrated Core Architecture. (2015). https://en.wikipedia.org/wiki/Xeon_Phi

Qiyao Xie, Tian Huang, Zhihai Zou, Liang Xia, Yongxin Zhu, and Jiang Jiang. 2012. An accurate power model for GPU processors. In *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*. IEEE, 1141–1146.

XPE. 2015. Xilinx Power Estimator (XPE). (2015). http://www.xilinx.com/products/technology/power/xpe.html

Ying Zhang, Yue Hu, Bin Li, and Lu Peng. 2011. Performance and Power Analysis of ATI GPU: A Statistical Approach. In *Proceedings of the 2011 IEEE Sixth International Conference on Networking, Architecture, and Storage (NAS '11)*. IEEE Computer Society, 149–158.

Jishen Zhao, Guangyu Sun, Gabriel H Loh, and Yuan Xie. 2013. Optimizing GPU energy efficiency with 3D die-stacking graphics memory and reconfigurable memory interface. *ACM Transactions on Architecture and Code Optimization (TACO)* 10, 4 (2013), 24.

Dan Zou, Yong Dou, and Fei Xia. 2012. Optimization Schemes and Performance Evaluation of Smith-Waterman Algorithm on CPU, GPU and FPGA. *Concurr. Comput. : Pract. Exper.* 24, 14 (Sept. 2012), 1625–1644.

APPENDIX

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

Online Appendix to: A Survey of Power and Energy Predictive Models in HPC Systems and Applications

KENNETH O'BRIEN, University College Dublin, Ireland
ILIA PIETRI, University of Manchester, UK
RAVI REDDY, University College Dublin, Ireland
ALEXEY LASTOVETSKY, University College Dublin, Ireland
RIZOS SAKELLARIOU, University of Manchester, UK

A. POWER AND ENERGY MODELS: DETAILS

A.1. Power and Energy Models for CPUs

[Basmadjian et al. 2011] construct a power model of a server as a summation of power models of its components, the processor (CPU), memory (RAM), fans, and disk (HDD). The power consumptions for different types of servers are given by the following equations :

$$P_{Blade} = \sum_{i=1}^{l} P_{Mainboard} \tag{20}$$

$$P_{Tower\_or\_Rackable} = \sum_{i=1}^{l} P_{Mainboard} + \sum_{j=1}^{m} P_{Fan} + \sum_{k=1}^{n} P_{PSU} \tag{21}$$

where l, m, and n denote respectively the total number of mainboards, fans, and power supply units. For purposes of brevity, only their power model for *blade* servers is presented here.

The power consumption of the mainboard is given by the following equation [Basmadjian et al. 2011]:

$$P_{Mainboard} = \sum_{i=1}^{l} P_{CPU} + P_{RAM} + \sum_{j=1}^{m} P_{NIC} + \sum_{k=1}^{n} P_{HDD} + c \tag{22}$$

where l, m, and n denote the total number of processors (CPU), the total number of network interface cards (NIC), and the total number of attached hard disk drives (HDD) respectively and c is a constant (55 W for blade servers).

The power consumption of a multicore processor, P_{CPU}, is calculated as the sum of power consumptions of individual cores. The power consumption of each individual core is based on the linear single-core model of [Fan et al. 2007]. [Basmadjian et al. 2011] model the power consumption of a multicore processor containing n cores as follows:

$$P_{C_i} = P_{max} \times (U_{C_i}/100) \tag{23a}$$

$$P_{CPU} = P_{base} + \sum_{i=1}^{n} P_{C_i} \tag{23b}$$

$$P_{max} = V_{max}^2 \times f_{max} \times C_{eff} \tag{23c}$$

DOI: 0000001.0000001

where P_{C_i} indicates the power consumption of a core, p_{base} represents the base power consumption of a processor, V_{max} and f_{max} signify the voltage and frequency at maximum utilization respectively, and C_{eff}, the effective capacitance.

For memory module, the power consumption is given by [Basmadjian et al. 2011]:

$$P_{RAM} = P_{RAM\_base} + \gamma \times \alpha \times \beta \tag{24}$$

where α = 1, 2.3, 1.3, and 1.9 for unbuffered DDR_2 SDRAM, buffered DDR_2 SDRAM, unbuffered DDR_3 SDRAM, and buffered DDR_3 SDRAM respectively. The value of β used is 7.347. A probabilistic approach is used to model γ for a processor not in idle state, otherwise the value of γ used is 0. [Basmadjian et al. 2011] and [Mämmelä et al. 2012] model the idle power consumption of a unbuffered SDRAM of type DDR_2 or DDR_3 as follows:

$$P_{RAM\_base} = \sum_{i=1}^{r} s_i \times p \tag{25}$$

where *r* represents the total number of installed memory modules and s_i indicates the size of memory module i. [Basmadjian et al. 2011] and [Mämmelä et al. 2012] model the power consumption of the hard disk as follows:

$$P_{HDD} = a \times 1.4 \times P_{base} + b \times P_{HDD\_base} + c \times 3.7 \times P_{base} \tag{26a}$$

$$P_{HDD\_base} = P_{base} \times (d + 0.2 \times e) \tag{26b}$$

where a, b, c, d, and e denote probabilities of different states of the disk and P_{base} is the base power consumption.

Based on the model evaluations on tower and blade servers, the authors report maximum prediction error rates of 8% and 9% for tower servers and blade servers respectively.

B. POWER AND ENERGY MODELS FOR GPUS

[Hong 2010] present a GPU power consumption prediction model that is modelled similar to the PMC-based unit power prediction approach of [Isci and Martonosi 2003]. Their model is described below.

$$GPU\_power = DynamicPower + BasePower$$

$$DynamicPower = \sum_{i=0}^{n} DP\_Component\_i = DP\_SMs + DP\_Memory$$

$$\sum_{i=0}^{n} SM\_Component_i =$$

$$DP\_Int + DP\_Fp + DP\_Sfu + DP\_Alu + DP\_Texture\_Cache +$$

$$DP\_Const\_Cache + DP\_Shared + DP\_Reg + DP\_FDS + DP\_Const\_SM$$

$$DP\_SMs = Num\_SMs \times \sum_{i=0}^{n} SM\_Component_i$$

where $BasePower$ is the idle power consumption of a GPU and Num_SMs is the total number of Streaming Multiprocessors (SM) in a GPU. The dynamic power for each

component is calculated as follows [Hong 2010]:

$$
\begin{aligned}
DP\_Component\_i &= MaxPower_{component} \times AccessRate_{component} \\
AccessRate_{component} &= \frac{DAC\_per\_th_{component} \times Warps\_per\_SM}{Predicted\_Exec\_cycles/4} \\
DAC\_per\_th_{component} &= \sum_{i=0}^{n} Number\_Inst\_per\_warps_i(comp) \\
Warps\_per\_SM &= (\frac{\#Threads\_per\_block}{\#Threads\_per\_warp} \times \frac{\#Blocks}{\#Active\_SMs})
\end{aligned}
$$

The parameter, $DAC\_per\_th_{component}$), is calculated as the total number of instructions that access an architectural component. The parameter, $Warps\_per\_SM$, indicate the number of warps that are executed in one SM. The parameter, ($Predicted\_Exec\_cycles$), is calculated using an analytical timing model, which we don't present here since our main focus in this paper is power and energy models. The dynamic power of GDDR memory (DP_Memory) is modelled based on the dynamic powers of the global memory and local memory that share it [Hong 2010]:

$$
DP\_Memory = \sum_{i=0}^{n} Memory\_component_i = DP\_GlobalMem + DP\_LocalMem
$$

The other memories (shared, constant, texture) are modelled separately as SM components. The parameter, $MaxPower$, is empirically determined by stressing different architectural units in a GPU using synthetic micro-benchmarks. A special piecewise linear approach is used for eight power components due to their non-linear behaviour similar to how it was done in [Isci and Martonosi 2003]. Finally, dynamic power is modelled as follows [Hong 2010]:

$$
\begin{aligned}
&DP\_SMs = Max\_SM \times log_{10}(\alpha \times Active\_SMs + \beta) \\
&Max\_SM = Num\_SMs \times \sum_{i=0}^{n} SM\_Component_i \\
&DynamicPower = (Max\_SM + DP\_Memory) \times log_{10}(\alpha \times Active\_SMs + \beta) \\
&\alpha = (10 - \beta)/Num\_SMs, \beta = 1.1
\end{aligned}
$$

where $Active\_SMs$ is the number of active SMs in the GPU.

To demonstrate the accuracy of their model, NVIDIA GTX280 GPU is used. The number of dynamic instructions and instruction types (which are used to calculate the access rates) are determined using a GPU PTX emulator, Ocelot ([Farooqui et al. 2011]). The authors [Hong 2010] report that the IPP prediction error for the total power consumption is 8.94% and the average energy consumption savings are 10.99%. The main factor hindering the portability of this model is that it requires detailed architectural information and contains a large set of parameters.

C. HIGH PERFORMANCE COMPUTING APPLICATIONS

[Diouri et al. 2013] present energy prediction model for MPI broadcast algorithms in large scale HPC systems. They present models for four broadcast algorithms, Scatter and AllGather algorithm (MPI/SAG) [Thakur and Gropp 2003] used in MPICH2 [MPICH 2016], Pipelining algorithm (MPI/Pipeline) provided in OpenMPI 1.4.4, hybrid broadcasting algorithm which combines MPI/SAG and OpenMP, and a hybrid algorithm, which combines MPI/Pipeline and OpenMP [OpenMPI 2016]. The energy

consumption of MPI broadcast operation (MPI/SAG) is modelled as follows [Diouri et al. 2013]:

$$\begin{aligned} E_{MPI/SAG} &= \sum_{i=1}^{N} e_{MPI/SAG}^{Node_i} + \sum_{j=1}^{M} e_{MPI/SAG}^{Switch_j} \\ &= t_{Scatter}(p,N) \times (\sum_{i=1}^{N} \rho_{Scatter}^{Node_i}(p) + \sum_{j=1}^{M} \rho_{Scatter}^{Switch_j}) \\ &+ t_{AllGather}(p,N) \times (\sum_{i=1}^{N} \rho_{AllGather}^{Node_i}(p) + \sum_{j=1}^{M} \rho_{AllGather}^{Switch_j}) \end{aligned} \tag{27}$$

$t_{op}(p,N)$ is the time required to perform op (Scatter, AllGather, Pipeline, or CopyPrivate) over the $N \times p$ processes, where N is the number of nodes. Within each node i, p processes are involved in the execution of op. $\rho_{op}^{Node_i}(p)$ is the power consumption of the node i during t_{op}. $\rho_{op}^{Switch_j}(p)$ is the power consumption of the switch j during t_{op}. They validate their energy prediction model on Grid5000 [Cappello et al. 2005] and they report a worst prediction error of -6.82%.

[Gamell et al. 2013] explore energy and performance trade-offs of data movement and I/O at extreme scales. Their energy consumption model follows:

$$E = E_{system} + E_{comm} \tag{28a}$$

$$E_{system} = T \times (P_{static} + P_{dynamic}) \tag{28b}$$

$$P_{static} = P_{cpu}^{base} + P_{mem}^{base} + P_{nic}^{base} + P_{misc}^{base} \tag{28c}$$

$$P_{dynamic} = a \times P_{cpu}^{active} + b \times P_{mem}^{active} \tag{28d}$$

$$P_{cpu}^{active} = C \times V^2 \times \alpha \times f \tag{28e}$$

where P_{cpu}^{base}, P_{mem}^{base}, P_{nic}^{base}, and P_{misc}^{base} are the base power consumptions of CPU, DRAM, NIC, and other miscellanous components. Dynamic power is predicted by using the capacitance (C), switching activity (α), operational voltage (V), and frequency (f). The parameters a and b are determined using the number of arithmetic operations per second and the number of memory accesses per second. The energy consumption model of a MPI communication operation is constructed as follows [Gamell et al. 2013]:

$$E_{comm} = \sum_{i=0}^{M} \frac{data_i}{BW_{net}} \times P_{transfer}, \quad if \quad smp(src_i) \neq smp(dest_i) \tag{29a}$$

$$E_{comm} = \sum_{i=0}^{M} \frac{data_i}{BW_{mem}} \times (P_{cpu}^{active} + P_{mem}^{active}), \quad if \quad smp(src_i) = smp(dest_i) \tag{29b}$$

where $smp(i) = smp(j)$ indicates that MPI ranks i and j are mapped to cores that share memory, BW_{net} and BW_{mem} are the bandwidth values of network and memory respectively, and $P_{transfer}$ depends on the characteristics of the underlying network (for example, Infiniband, Gemini).

D. POWER AND ENERGY MODELS: PARAMETERS

Table VIII Parameters and Decomposition characteristics of the CPU Power and Energy Models Surveyed.

Model	**Parameters**	**Decomposition**
[Bellosa 2000]	microoperation, floating-point operations, second-level address strobes, memory transactions **[Bellosa 2000]**	single-core CPU
[Isci and Martonosi 2003]	128bit_MMX_uop, 64bit_MMX_uop, BPU Fetch Req, Branch Retired, BSQ Cache Ref, Bus Ratio, Front End Event, FSB Data Activity, IOQ Allocation, ITLB Reference, Ld Port Replay, Machine Clear, MOB Load Replay, packed_DP_uop, packed_SP_uop, scalar_DP_uop, scalar_SP_uop, St Port Replay, TC Deliver Mode, Uop Queue Writes, Uops retired, uop_type, x87_FP_uop, x87_SIMD_moves_uop **[Isci and Martonosi 2003]**	1st Level BPU, 2nd Level BPU, Allocation, Bus Control, Data TLB, FP Exec, FP Regfile, Inst Dec, Inst Queue1, Inst Queue2, INT Exec, INT Regfile, ITLB & Fetch, L1 cache, L2 Cache, MEM control, MOB, Rename, RETIRE, Schedule, Trace Cache, Ucode ROM **[Isci and Martonosi 2003]**
[Heath et al. 2005]	$C_{base}, U_{CPU}, U_{Disk}, U_{Net}$	CPU, disk, network
[Economou et al. 2006]	$C_{base}, U_{CPU}, U_{Mem}, U_{Disk}, U_{Net}$	CPU, memory, disk, network
[Lee and Brooks 2006]	ALU Latency, Branch, Branch Latency, Depth, D-L1 Cache Size, Fixed-Point/Memory, Floating-Point, Floating-Point (FP), FP-Divide Latency, FPU Latency, Functional Units, FX-Divide Latency, FX-Multiply Latency, General Purpose (GP), I-L1 Cache Size, L2 Cache Latency, L2 Cache Size, Load/Store Latency, L/S Reorder Queue, Main Memory Latency, Special Purpose (SP), Store Queue, Width **[Lee and Brooks 2006]**	single-core CPU
[Fan et al. 2007]	P_{base}, P_{max}, U	single-core CPU
[Fan et al. 2007]	C_{base}, U_{CPU}, r	single-core CPU

[Lewis et al. 2008]	Ambient_Temp0, Ambient_Temp1, CPU0 Die Temp, CPU1 Die Temp, HT1 Bus X-Actions, HT2 Bus X-Actions, L1/L2 Cache Miss (Core 0 to Core 3), Disk Bytes Read, Disk Bytes Written, $P_{spin-up}$, t_{su}, P_{read}, N_r, t_r, P_{write}, N_w, t_w, P_{base}, t_{base}, RPM_{Fan}, RPM_{base}, P_{fan}, $t_{ipmi-slice}$, $P_{optical}$, $t_{optical}$, $V_{pow-line}$, $I_{pow-line}$, $t_{timeslice}$	Cor0 to Core 3, DRAM, HDD, Fan, Support Chipsets
[Wang et al. 2010]	C_{base}, U_{CPU}, $U_{I/O}$	CPU, I/O
[Basmadjian et al. 2011]	P_{base}, V_{max}, f_{max}, C_{eff}, U_C, n, s_i, p, γ, β, a, b, c, d, e **[Basmadjian et al. 2011]**	CPU, RAM, NIC, HDD, Fan
[Bertran et al. 2013]	$AR_{(comp,core)}$, P_{comp}, P_{static} where $core$ = 0,1, $comp$ = BPU, FE, FP, FSB, INT, L1, L2, SIMD **[Bertran et al. 2013]**	BPU and branch execution, DECODE, FETCH_UNIT, Floating point arithmetic units, FSB, Integer arithmetic units, L1, L1 DTLB, L1_ICACHE, L1_ITLB, L2, L2 DTLB, LD/ST execution, LSD, memory, MOB, PREDECODE, RAT, RETIRE, ROB, SIMD arithmetic units, SPT, uCODE ROM, uOP BUFFER **[Bertran et al. 2013]**

[Bircher and John 2012]	*For server systems*, parameters are: Cycles, DMA misses, Fetched μops, Halted cycles, Interrupts, L3 Cache misses, Processor memory bus transactions, TLB misses, Uncacheable accesses **[Bircher and John 2012]** *For desktop systems*, parameters are: CPU clock frequency, CPU to I/O transactions, CPU voltage, DC accesses, DCTPageConflicts, DCTPageHits, DCTPageMisses, DRAM active percent, Fetched μops, FP μops retired, GPU nongated clocks, %Halted/%Not-Halted, Link active percent, Spindle active percent, Temperature **[Bircher and John 2012]**	Chipset, CPU, Disk, I/O, Memory, Memory Controller **[Bircher and John 2012]**

Table IX Parameters and Decomposition characteristics of the GPU Power and Energy Models Surveyed.

Model	**Parameters**	**Decomposition**
[Hong 2010]	α, β, Num_SMs, $Active\_SMs$, $Predicted\_Exec\_cycles$, $\#Threads\_per\_block$, $\#Threads\_per\_warp$, $\#Blocks$ $DAC\_per\_th_{component}$, where component = Alu, Const_cache, Const_SM, FDS, Fp, GlobalMem, Int, LocalMem, Reg, Sfu, Shared, Texture_cache **[Hong 2010]**	ALU, Constant cache, FDS (Fetch/Dec/Sch), Floating Point Unit, Global memory, Int. arithmetic unit, Local memory, Register File, SFU, Shared memory, Texture cache **[Hong 2010]**
[Nagasaka et al. 2010]	branch, divergent_branch, gld_128b, gld_32b, gld_64b, gst_128b, gst_32b, gst_64b, instructions, local_load, local_store, tlb_miss, warp_serialize **[Nagasaka et al. 2010]**	GPU

[Chen et al. 2011]	ALU, Atomic, BankConf, Barrier, Branch, CMInst.hit, CMInst.miss, D.Branch, D.FP, GMInst, INT, LMInst, M.Barrier, Occupancy, PMInst, Register, S.FP, SFU, ShMInst, TMInst.hit, TMInst.miss, UncoalesMem **[Chen et al. 2011]**	GPU
[Kasichayanula et al. 2012]	$N_{SM,i}$, P_i, U_i, B_i **[Kasichayanula et al. 2012]**	Floating Point Unit, Global Memory, Shared Memory **[Kasichayanula et al. 2012]**
[Song et al. 2013]	branch, divergent branch, gld request + l1 global load hit + l1 global load miss, global store transaction, inst executed, local load, local store, Shared load + l1 shared bank conflict, tex0 cache sector misses, tex0 cache sector queries, t_{pci}, t_{kernel}, $\bar{P}_{base}$, $E_{parallel-overhead}$ **[Song et al. 2013]**	Floating Point Unit, Global Memory, Shared Memory, Local Memory, Texture Cache **[Song et al. 2013]**
[Lim et al. 2014]	access time, access time, address bus width, associativity, associativity, #banks, #banks, buffer line size, chip coverage, cycle time, cycle time, data bus width, decoded opcode width, duty cycle, #entries, #entries, flit bits, input line width, input line width, I/O buffer entries, link latency, link throughput, #memory channels, output line width, output line width, peak transfer rate, percentage of pipelining, pipeline stages, #ports (in, out), #ports(R, W, RW), #ranks, request window entries, router or bus, selection input size, selection output size, tag width, tag width, #virtual channels, width **[Lim et al. 2014]**	Block/Warp States, Cache Buffers, Constant Cache, Data TLB, Fetch Queue, Instruction Cache, Instruction Decoder, Instruction Issue Selection Logic, Instruction Queue, Instruction TLB, L1 Cache, L2 Cache, LD/ST units, Memory Controller, NoC, PipelineLatches, Register File, Scoreboard, SFU, Shared Memory, SP, Texture Cache **[Lim et al. 2014]**
[Wang and Cao 2015]	A, S_a, N_{inst}, T_{arc} **[Wang and Cao 2015]**	GPU

[Zhang et al. 2011]	ALUBusy, ALUFetchRatio, ALUInsts, ALUPacking, FastPath, FCStacks, FetchInsts, FetchSize, FetchUnitBusy, GPR, LDSBankConfict, LDSFetchInsts, LDSWriteInsts, LocalMemSize, ScratchRegs, Wavefronts, WriteInsts, WriteUnitStalled **[Zhang et al. 2011]**	GPU

Table X Parameters and Decomposition characteristics of the Intel Xeon Phi Power and Energy Models Surveyed.

Model	**Parameters**	**Decomposition**
[Shao and Brooks 2013]	$EPI_{accessmode,optype}$ where $accessmode$ = $Register$, $L1$, $L2$, $Mem\_with\_prefetch$, $Mem\_without\_prefetch$, $Write\_to\_mem$, and $optype$ = $ScalarOp$, $VectorOp$, $vprefetch0\_to\_L1$, $vprefetch1\_to\_L2$ **[Shao and Brooks 2013]**	Compute, Hardware PF, MEM, Private Cache, Redundant SW-PF, Remote Cache, Software PF **[Shao and Brooks 2013]**

Table XI Parameters and Decomposition characteristics of the Power and Energy Models used in the HPC Applications surveyed.

Model	**Parameters**	**Decomposition**
[Bui et al. 2008]	$\Delta Cycles$, L1 Total Cache Access, L2 Total Cache Access, L3 Total Cache Access, Total Instructions Retired **[Bui et al. 2008]**	L1 cache, Core Logic, L3 Cache, L2 Cache **[Bui et al. 2008]**
[Subramaniam and Feng 2010]	N, NB, P, Q	Node
[Tiwari et al. 2012]	ti, tj, tk (i, j, k tiles), ti_1, tj_1, tk_1 (trsm tiles), tj (loop j tile), ti_2, tj_2, tk_2 (MM tiles), CPU freqs ($freq$), matrix sizes ($msize$), ui, uj (i, j unrolls), ui_1, uj_1 (trsm unrolls), ui_2, uj_2 (MM unrolls) **[Tiwari et al. 2012]**	CPU, DIMM

[Lively et al. 2012]	Cache_FLD_per_instruction, LD_ST_stall_ per_cycle, PAPI_BR_INS, PAPI_L1_DCM, PAPI_L1_ICA, PAPI_L1_TCA, PAPI_L1_TCM, PAPI_L2_ICM, PAPI_L2_TCA, PAPI_L2_TCH, PAPI_RES_STL, PAPI_TLB_DM, PAPI_TOT_INS **[Lively et al. 2012]**	CPU, Memory
[Kestor et al. 2013a]	FP operations, last level cache misses, Retired instructions, stalled cycles **[Kestor et al. 2013a]**	Integer, Floating point, L1 cache, L2 Cache, L3 Cache, Memory **[Kestor et al. 2013a]**
[Gschwandtner et al. 2014]	_DISP, _DOUBLE_ISSUED, PAPI_FP_INS, PAPI_INT_INS, PAPI_L1_DCM, PAPI_L2_DCM, PAPI_L3_DCM, PAPI_L3_DCR, PAPI_TOT_CYC, PAPI_TOT_INS, PM_CMPLU_STALL, PM_CMPLU_STALL_THRD, PM_L1_ICACHE_MISS, PM_L2_INST_MISS, PM_L3_MISS, PM_L3_PREF_MISS, PM_LSU_DC_PREF, PM_LSU_FX_FIN, PM_LSU_LDF, PM_LSU_LDX, PM_MEM0_PREFETCH, PM_VSU_FMA_DOUBLE, PM_VSU_SIMPLE_ISSUED, PM_VSU_VECTOR, PM_VSU_VECTOR, _SINGLE_ISSUED, _STREAM_CONFIRM **[Gschwandtner et al. 2014]**	CPU

E. CASE STUDY: PERFORMANCE MONITORING COUNTERS AND REGRESSION MODEL

E.1. Case Study: Performance Monitoring Counters

Table XII shows the Likwid [Haswell 2013] performance groups and performance counters (PMCs) that are used as parameters in the regression model in the experiments.

E.2. Case Study: Regression Model Coefficients

Table XIII shows the multiple linear regression coefficients obtained for power and energy models.

E.3. Case Study: Regression Model Training Times

Figure 2 shows the distribution of execution times of applications in the training set.

Table XII Likwid [Haswell 2013] performance groups and performance counters (PMCs)

Performance Group	**Performance Monitoring Counters**
BRANCH (Branch prediction miss rate/ratio)	BR_MISP_RETIRED_ALL_BRANCHES, BR_INST_RETIRED_ALL_BRANCHES **[Haswell 2013]**
DATA (Load to store ratio)	UOPS_RETIRED_ALL, MEM_UOPS_RETIRED_STORES, MEM_UOPS_RETIRED_LOADS **[Haswell 2013]**
ICACHE (Instruction cache miss rate/ratio)	ICACHE_ACCESSES, ICACHE_MISSES, ICACHE_IFETCH_STALL, ILD_STALL_IQ_FULL **[Haswell 2013]**
L2CACHE (L2 cache miss rate/ratio)	L2_RQSTS_MISS, L2_TRANS_ALL_REQUESTS **[Haswell 2013]**
L2 (L2 cache bandwidth in MBytes/s)	L1D_REPLACEMENT, L2_TRANS_L1D_WB **[Haswell 2013]**
L3CACHE (L3 cache miss rate/ratio)	UOPS_RETIRED_ALL, MEM_LOAD_UOPS_RETIRED_L3_MISS, MEM_LOAD_UOPS_RETIRED_L3_ALL **[Haswell 2013]**
L3 (L3 cache bandwidth in MBytes/s)	L2_TRANS_L2_WB, L2_LINES_IN_ALL **[Haswell 2013]**
TLB_DATA (L1 Data TLB miss rate/ratio)	DTLB_STORE_MISSES_WALK_DURATION, DTLB_STORE_MISSES_CAUSES_A_WALK, DTLB_LOAD_MISSES_WALK_DURATION, DTLB_LOAD_MISSES_CAUSES_A_WALK **[Haswell 2013]**
TLB_INSTR (L1 Instruction TLB miss rate/ratio)	ITLB_MISSES_CAUSES_A_WALK, ITLB_MISSES_WALK_DURATION **[Haswell 2013]**
UOPS_EXEC (UOPs execution)	UOPS_EXECUTED_USED_CYCLES, UOPS_EXECUTED_STALL_CYCLES, CPU_CLOCK_UNHALTED_TOTAL_CYCLES, UOPS_EXECUTED_STALL_CYCLES **[Haswell 2013]**
UOPS_ISSUE (UOPs issueing)	UOPS_ISSUED_USED_CYCLES, UOPS_ISSUED_STALL_CYCLES **[Haswell 2013]**
UOPS_RETIRE (UOPs retirement)	UOPS_RETIRED_USED_CYCLES, UOPS_RETIRED_STALL_CYCLES **[Haswell 2013]**
UOPS (UOPs execution info)	UOPS_ISSUED_ANY, UOPS_EXECUTED_THREAD, UOPS_RETIRED_ALL, UOPS_ISSUED_FLAGS_MERGE **[Haswell 2013]**
Fixed function performance counter registers	CPU_CLK_UNHALTED_CORE, INSTR_RETIRED_ANY, CPU_CLK_UNHALTED_REF **[Haswell 2013]**

Table XIII Multiple linear regression coefficients for power and energy Models

Predictors (Likwid PMCs) [Haswell 2013]	**Model Coefficients for Average Dynamic Power**	**Model Coefficients for Energy**
INSTR_RETIRED_ANY:FIXC0	-4.489847e-09	9.837091e-08
CPU_CLK_UNHALTED_CORE:FIXC1	-8.192357e-09	-4.365642e-08
CPU_CLK_UNHALTED_REF:FIXC2	7.601429e-09	6.070421e-08
BR_INST_RETIRED_ALL_BRANCHES:PMC0	5.102516e-10	-8.391657e-09
BR_MISP_RETIRED_ALL_BRANCHES:PMC1	-1.049940e-11	1.071510e-09
MEM_UOPS_RETIRED_LOADS:PMC0	2.383515e-11	-3.902474e-09
MEM_UOPS_RETIRED_STORES:PMC1	5.102516e-10	-8.391657e-09
ICACHE_ACCESSES:PMC0	-1.049940e-11	1.071510e-09
ICACHE_MISSES:PMC1	5.102516e-10	-8.391657e-09
ICACHE_IFETCH_STALL:PMC2	-1.049940e-11	1.071510e-09
ILD_STALL_IQ_FULL:PMC3	2.383515e-11	-3.902474e-09
L2_TRANS_ALL_REQUESTS:PMC0	-6.054451e-10	3.830272e-07
L2_RQSTS_MISS:PMC1	5.102516e-10	-8.391657e-09
MEM_LOAD_UOPS_RETIRED_L3_ALL:PMC0	-1.049940e-11	1.071510e-09
MEM_LOAD_UOPS_RETIRED_L3_MISS:PMC1	5.102516e-10	-8.391657e-09
DTLB_LOAD_MISSES_CAUSES_A_WALK:PMC0	-1.049940e-11	1.071510e-09
DTLB_STORE_MISSES_CAUSES_A_WALK:PMC1	5.102516e-10	-8.391657e-09
DTLB_LOAD_MISSES_WALK_DURATION:PMC2	-1.049940e-11	1.071510e-09
DTLB_STORE_MISSES_WALK_DURATION:PMC3	2.383515e-11	-3.902474e-09
ITLB_MISSES_CAUSES_A_WALK:PMC0	-6.054451e-10	3.830272e-07
ITLB_MISSES_WALK_DURATION:PMC1	5.102516e-10	-8.391657e-09
UOPS_EXECUTED_USED_CYCLES:PMC0	-1.049940e-11	1.071510e-09
UOPS_EXECUTED_STALL_CYCLES:PMC1	5.102516e-10	-8.391657e-09
CPU_CLOCK_UNHALTED_TOTAL_CYCLES:PMC2	-1.049940e-11	1.071510e-09
UOPS_EXECUTED_STALL_CYCLES:PMC3:EDGEDETECT	2.383515e-11	-3.902474e-09
UOPS_ISSUED_USED_CYCLES:PMC0	-6.054451e-10	3.830272e-07
UOPS_ISSUED_STALL_CYCLES:PMC1	5.102516e-10	-8.391657e-09
UOPS_ISSUED_STALL_CYCLES:PMC3:EDGEDETECT	-1.049940e-11	1.071510e-09
UOPS_RETIRED_USED_CYCLES:PMC0	-6.054451e-10	3.830272e-07
UOPS_RETIRED_STALL_CYCLES:PMC1	5.102516e-10	-8.391657e-09
UOPS_RETIRED_STALL_CYCLES:PMC3:EDGEDETECT	-1.049940e-11	1.071510e-09
UOPS_ISSUED_ANY:PMC0	-6.054451e-10	3.830272e-07
UOPS_EXECUTED_THREAD:PMC1	5.102516e-10	-8.391657e-09
UOPS_RETIRED_ALL:PMC2	-1.049940e-11	1.071510e-09
UOPS_ISSUED_FLAGS_MERGE:PMC3	-6.054451e-10	3.830272e-07

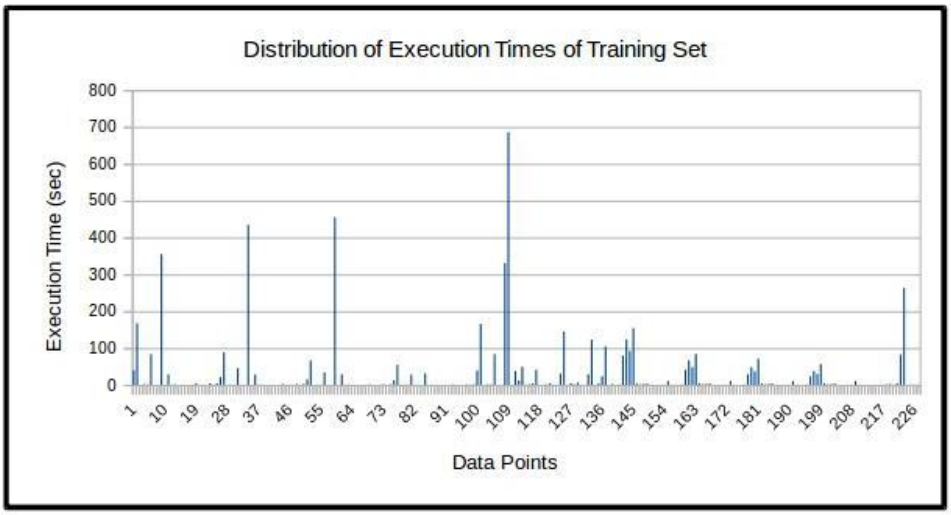

Fig. 2 Execution times of applications in the training set.