A Methodology for Full-System Power Modeling in Heterogeneous Data Centers

Mauro Canuto[1], Raimon Bosch[1], Mario Macias[1], and Jordi Guitart[1,2]
[1]Barcelona Supercomputing Center (BSC), [2]Universitat Politecnica de Catalunya (UPC)
Barcelona, Spain
{mauro.canuto, raimon.bosch, mario.macias, jordi.guitart}@bsc.es

ABSTRACT

The need for energy-awareness in current data centers has encouraged the use of power modeling to estimate their power consumption. However, existing models present noticeable limitations, which make them application-dependent, platform-dependent, inaccurate, or computationally complex. In this paper, we propose a platform- and application-agnostic methodology for full-system power modeling in heterogeneous data centers that overcomes those limitations. It derives a single model per platform, which works with high accuracy for heterogeneous applications with different patterns of resource usage and energy consumption, by systematically selecting a minimum set of resource usage indicators and extracting complex relations among them that capture the impact on energy consumption of all the resources in the system. We demonstrate our methodology by generating power models for heterogeneous platforms with very different power consumption profiles. Our validation experiments with real Cloud applications show that such models provide high accuracy (around 5% of average estimation error).

1. INTRODUCTION

Energy used by data centers worldwide increased by about 56% from 2005 to 2010, accounting for 1.5% of total energy use in 2010 [28]. This contribution is expected to increase in the following years [16] and this has encouraged the development of techniques to reduce the energy consumption and the environmental footprint of data centers. These techniques require data centers to be energy-aware, that is, they must be able to measure and predict their energy consumption. A common approach toward energy-awareness is through power models, which allow estimating the power consumption by means of indirect evidences (such as resource usage) [31]. This differs from power measurement, which measures the actual power consumption by means of special hardware devices. Power models are especially helpful when direct measurement is not possible (e.g. to assess power of individual software components such as processes or virtual machines) or expensive (e.g. to assess power at low granularities).

A majority of the works that derive full-system power models follow a similar methodology, which consists of the one-time offline execution of the following phases: i) deployment of a platform for power metering and system monitoring; ii) collection of power measurements and resource usage indicators periodically while the system runs a special training workload; iii) model generation by fitting the resource usage indicators to the power measurements with some machine learning technique; iv) model validation.

However, many of those works present noticeable limitations. First, some models only consider the impact of the processor on the power consumption [23]. Whereas the processor has been the main contributor to the power consumed by traditional HPC applications, modern data centers run heterogeneous applications with diverse resource usage [32], including the memory, the disk, and the network. Note that those subsystems apart from the processor have been reported to make up 40%-60% of the total power consumption depending on the workload [18]. In order to avoid models that are specific for CPU-intensive applications, the impact on the power consumption of the rest of subsystems should be also considered.

Second, some authors do not perform an adequate selection of indicators to account for the usage of each resource. On one side, some of them simply apply machine learning techniques with all the indicators provided by the monitoring framework [19], which increases the complexity of the data mining process and makes it harder to find correlations between the resource usage and the power consumption. On the other side, other authors use indicators that are not unequivocally correlated with the power consumption incurred by a given resource. For instance, the utilization is not the best indicator of the processor usage because applications with the same processor utilization can have different processor power consumption depending on what instructions they are executing [27]. For this reason, the utilization cannot be the unique determinant of the processor power consumption, though it could be used to further refine a model that considers, for instance, the number of operations executed. According to this, the modeling methodology should exploit the capabilities of machine learning techniques to systematically filter the relevant indicators.

Third, proposed models have commonly assumed a linear relation between the power consumption and the resource usage [22], which provided a reasonable accuracy with low computational complexity in traditional platforms. However, linear models are weak in modern platforms such as multicore systems [30], thus models able to accurately capture non-linear relations while incurring a reasonable computational complexity are required. Whereas machine learning is a very powerful technique to capture data correlations, detecting complex correlations requires driving the mining process by including derivatives of the resource usage indicators that capture better their relation with the power consumption.

Fourth, some models are flawed because of the design of the training or the validation workloads. On one side, some authors only

UCC '16, December 06-09, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/2996890.2996899

include a given kind of applications in the training workload [34], which results in inaccurate estimations for the rest of applications. The training workload should be generic, able to capture the essentials of the power behavior of the modeled host when running heterogeneous applications. On the other side, some authors validate the model with the same applications used to train it [20], which does not provide any insight about its accuracy with the rest of applications. The validation and the training workloads should be independent.

Finally, some modeling works require a priori knowledge of the target platform, for instance, to select the input performance indicators of the model [18], thus forcing to completely rethink the methodology when a different platform wants to be modeled. According to this, the modeling methodology should be designed from the beginning with platform heterogeneity in mind in order to homogenize the tools in all the platforms and automatize the modeling procedure.

In this paper, we propose a methodology for full-system power modeling in data centers. At first glance, the proposed methodology is similar to former proposals. However, we have revisited, refined, and improved the whole procedure to overcome the limitations of the previous works. In particular, our methodology:

- Derives full-system power models by considering the impact on energy consumption of all the resources in the system, namely processor, cache, memory, disk, and network, and can be easily extended to include other resources (i.e. GPUs) as needed.
- Derives a single model per platform that works with high accuracy for heterogeneous applications with different patterns of resource usage and energy consumption.
- Derives models by using a minimum set of resource usage indicators, which are selected for each platform according to their correlation with the power consumption.
- Derives models that capture non-linear relations between the resource usage and the power consumption.
- Exploits machine learning techniques to drive the selection of resource usage indicators and the capture of non-linear relations without human intervention.
- Gets full-system information needed to generate the models by means of a comprehensive monitoring framework that seamlessly integrates several data sources, including system information, performance counters, and overall power consumption measures.
- Uses the same training set for each platform, which is independent of the applications used to validate the models and comprises various micro-benchmarks that selectively stress each resource at varying levels of utilization.
- Validates the models with real applications, which are commonly found in Cloud data centers, with different patterns of resource usage and energy consumption.
- Achieves platform- and application-agnosticism by using the same tools and systematically following the same steps to derive power models in heterogeneous platforms.

In addition, we demonstrate our methodology by generating power models for heterogeneous platforms with very different power consumption profiles. The evaluated platforms range from high-performance server architectures based on Intel Xeon and AMD Opteron to low-power architectures based on Intel Atom and ARM Cortex-A. None of the proposed methodologies up to now has been demonstrated in so diverse platforms regarding their power consumption.

The remainder of the paper is as follows. Section 2 presents the related work. Section 3 introduces the collection of the data required to build the models, whereas Section 4 describes the steps needed to generate such models from the data collected. The validation of the derived models is presented in Section 5 and the paper is concluded in Section 6.

2. RELATED WORK

Power models have been extensively used in the last years to enable energy-awareness in data centers. They range from models focusing on individual subsystems (especially the processor) to models considering the consumption of the entire system. Mobius et al. [31] presented a qualitative comparison of different models proposed on those areas, as well as models that estimate the power consumption of virtual machines.

Some authors have provided also quantitative comparisons of power models. In particular, Rivoire et al. [33] compared several full-system power consumption models, which had been proposed by Fan et al. [23], Heath et al. [25], and Economou et al. [22]. They concluded that CPU utilization is not a good proxy for full-system power consumption, less detailed models may yield better results if a large contributor to the power is not modeled, the nuances of how the performance counters are defined across platforms do matter, and high-level interfaces to get insights about memory and disk power are required.

McCullough et al. [30] also compared a number of linear and non-linear regression models (including advanced modeling techniques such as Lasso regression) evaluating their accuracy in modern multicore platforms. Their results showed that power models doubled their prediction error in multicore platforms in comparison to their execution on a single core. Prediction errors were noticeably higher for linear models on individual subsystems such as the processor, where non-linear models only provided marginal improvement. McCullough et al. posited that this is due to effects such as cache contention, processor performance optimizations, and hidden device states not exposed to the operating system.

Some authors proposing full-system power consumption models use a similar methodology to our proposal. In particular, Bircher and John [18] estimated the power consumption of a complete system by extending the concept of using performance events as proxies for power measurement beyond the microprocessor to various subsystems. They trained each subsystem with a high-utilization workload, comprising a subset of the validation set and small synthetic workloads, while sampling the performance counters and only tracking the power on the corresponding subsystem. An initial selection of performance counters, which was dictated by an understanding of subsystem interactions, was then correlated with the measured power consumption using linear regression. If accuracy was not enough, the authors tried first to expand the selection of performance counters and then to use polynomial regression.

Cupertino et al. [19] proposed a methodology to model the power consumption of computing systems, which consisted on training the system with custom micro-benchmarks to stress the CPU, the cache, the memory, and the network, while gathering system information, performance counters, and model specific registers, as well as the overall power consumption. Then, they used Artificial Neural Networks to correlate the power consumption and all the captured performance indicators. Whereas neural networks can capture non-linear relations, they require that all variables used during

the training phase cover their entire spectrum of values to achieve an accurate estimation. Because of this and the lack of selection of performance indicators (which complicates the mining process), this approach presented noticeable errors (up to 30%) when validated with a workload comprising several HPC applications.

Economou et al. [22] introduced a method for modeling full-system power consumption so-called Mantis. They used a one-time calibration phase to capture some fixed system utilization metrics and performance counters, as well as the overall power consumption, while running a generic application emulator. Those metrics were then correlated to the power consumption by means of linear programming. Mantis presented up to 15% estimation error, especially on CPU-intensive benchmarks, mainly because its assumption of linearity and the use of the utilization as the sole indicator of the CPU usage.

Jarus et al. [26] introduced a methodology for creating power models based on the analysis of performance counters only related to CPU, cache, and memory behavior, which were captured, together with the overall power consumption, while running some selected HPC programs. Captured data was then clustered to create different models for groups of applications from the training set that share similar characteristics. A decision tree was also derived from captured data to select an appropriate power consumption model according to values of the performance counters. For each cluster of applications, the model that correlates the performance counters with the power consumption was derived by means of stepwise regression with forward selection. To capture non-linear relations, several transformations (none, logarithm, and square root) on the variables were also evaluated and included in the set if accuracy got improved. The training set in this work seems to be very specific for HPC applications. Even within the HPC domain, it is not clear how well other applications would fit in the clusters identified by the authors.

Witkowski et al. [34] presented a practical approach to power consumption estimation of both individual applications and entire nodes, though captured variables were only related to CPU, motherboard, and memory. To this end, they captured performance counters, system statistics, CPU core temperature data, as well as the overall power consumption, while running selected HPC applications. An initial selection of variables, comprising those with the highest correlation with the measured power, was then fitted with the measured power consumption by means of linear regression. Other variables were added as long as the coefficient of determination increased. Non-linear behaviors were captured by including also transformed derivatives of some variables in the regression. The authors simply analyzed the graphs showing their relation with the measured power consumption to determine the correct transformation to apply. Validation was performed by using the same HPC applications comprised in the training set. Whereas the authors claimed that their estimation method was designed for clusters with a relatively constant set of applications, its accuracy for non-HPC applications (and HPC applications not included in the training set) could be questioned.

Da Costa and Hlavacs [20] described a methodology for predicting the power consumption of a standard off-the-shelf PC. To this end, they measured performance counters, host and process related information, as well as the overall power consumption, while running synthetic workloads to stress CPU, memory, network, and disk. Captured explanatory variables (after including also their squared values) were then correlated with the power consumption via regression analysis. The obtained model was validated with the same synthetic workloads used during the training phase, thus nothing can be said about its accuracy with real applications.

3. DATA COLLECTION

Models are generated from a collection of power measurements and resource usage indicators gathered during a training execution which is performed once for each modeled platform. Hardware platforms considered in this work are described in Section 3.1. During the training execution, we periodically capture system information, performance counters, and the overall power consumption by means of a monitoring framework, which is described in Section 3.2. The training execution consists of a special workload that includes micro-benchmarks that selectively stress each resource at varying levels of utilization. By using this approach we aim to capture the essentials of the power behavior of the modeled platform without being tied to specific applications. The benchmarks used to stress each particular resource are described in Section 3.3.

3.1 Hardware platforms

We generate power models for heterogeneous platforms with very different power consumption profiles. As shown in Table 1, they range from high-performance server architectures based on Intel Xeon and AMD Opteron to low-power architectures based on Intel Atom and ARM Cortex-A.

3.2 Power metering and system monitoring

Models are generated from metrics from different layers, which are obtained by means of a comprehensive monitoring framework that seamlessly integrates several data sources including system information, performance counters, and overall power consumption measures. This framework is able to retrieve key resource usage indicators for the main subsystems of a physical host: CPU, cache, memory, disk, and network.

The Ganglia monitoring system [29] has been used as a collector, where all the measured data sources flow to. Ganglia facilitates the integration of all the data sources seamlessly, as well as the adoption of this solution on existing monitoring platforms in current data centers. As shown in Figure 1, Ganglia comprises *gmond* (Ganglia Monitoring Daemon) and *gmetad* (Ganglia Meta Daemon) instances. *gmond* can monitor system metrics from a host and share them with other hosts through a simple listen/announce protocol. *gmetad* polls the *gmond* instances periodically and stores their metrics to the hard disk in RRD format. We have also developed a custom query mechanism integrated in the Ganglia Web (*gweb*) interface that returns a JSON file containing all those metrics. In our setup, Ganglia components do not need to run in the hosts being monitored.

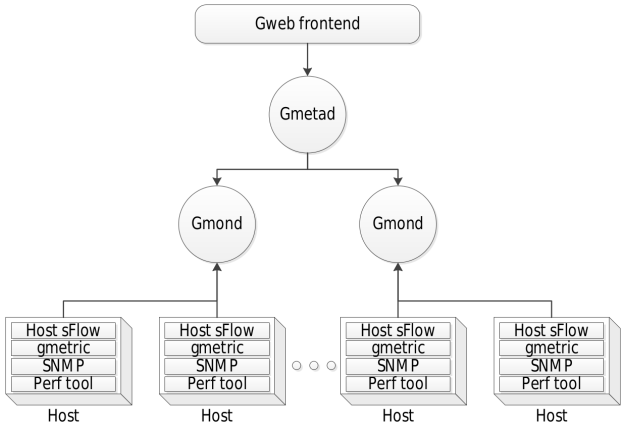

Figure 1: Monitoring architecture

We run Host sFlow [6] on every single host which needs to be

Table 1: Characteristics of modeled hardware platforms

Name	Platform	Processors	CPU Model	Clock Speed	Cores[1]	Threads[1]	Cache[1]	Memory	Disk	Net
srv-opt-1	HP ProLiant DL165 G7	2	AMD Opteron 6140	2.6 GHz	8	8	L1d: 8 x 64 KB L1i: 8 x 64 KB L2: 8 x 512 KB L3: 12 MB	32 GB DDR3-1333	2 TB SAS 7.2K	1 Gbps
srv-opt-2	HP ProLiant DL165 G7	2	AMD Opteron 6234	2.4 GHz	12	12	L1d: 12 x 16 KB L1i: 6 x 64 KB L2: 6 x 2 MB L3: 16 MB	32 GB DDR3-1600	2 TB SAS 7.2K	1 Gbps
srv-xeon-1	HP ProLiant DL160 G6	2	Intel Xeon X5650	2.66 GHz	6	12	L1d: 6 x 32 KB L1i: 6 x 32 KB L2: 6 x 256 KB L3: 12 MB	24 GB DDR3-1333	2 TB SAS 7.2K	1 Gbps
srv-xeon-2	HP ProLiant DL160 Gen8	2	Intel Xeon E5-2650	2 GHz	8	16	L1d: 8 x 32 KB L1i: 8 x 32 KB L2: 8 x 256 KB L3: 20 MB	32 GB DDR3-1600	2 TB SAS 7.2K	1 Gbps
srv-atom	ASUS AT3N7AI Twin-1U Rackmount	1	Intel Atom 330	1.6 GHz	2	4	L1d: 2 x 24 KB L1i: 2 x 32 KB L2: 2 x 512 KB L3: none	4 GB DDR2-800	320 GB SATA 7.2K	1 Gbps
srv-arm	Samsung Exynos 5 Dual Arndale Board	1	ARM Cortex-A15	1.7 GHz	2	2	L1d: 32 KB L1i: 32 KB L2: 1024 KB L3: none	2 GB DDR3-800	8 GB USB	100 Mbps

[1] Per Processor

monitored to capture the system-level metrics about CPU, memory, disk, network, and processes. They include, for instance, the processor time executing user and system code, the number of disk reads and writes, the number of input and output network packets, the amount of free memory, etc. The Host sFlow agent can interact with Ganglia through *gmond* and becomes fully integrated in its architecture. In fact, Host sFlow allows increasing the efficiency of Ganglia-based monitoring platforms [29]. When using Host sFlow, *gmond* is used in deaf mode: it does not collect any metrics by itself but aggregates the ones coming from several Host sFlow agents. In fact, the Host sFlow agent is able to collect a superset of *gmond* base metrics. Not only real-time system-level metrics about the behavior of the physical host, but also metrics related to virtual machines running in the system (captured through *libvirt*) and Docker containers. We will model the power consumption of virtual machines and containers in our future work.

We use *perf*, which is a profiler tool supported by Linux 2.6+ based systems, to capture a rich set of performance and raw performance event counters. They include, for instance, processor (and cache) micro-architectural events such as the number of cycles, instructions retired, L1, L2, and LLC cache loads and stores, L1, L2, and LLC cache misses, branches, etc. The raw performance event counters are additional CPU counters that *perf* does not list out-of-the-box as named counters. Examples of them are the number of floating-point or SIMD instructions executed. To capture them, its hexadecimal code needs to be found out using *perfmon2/libpfm* (as described in [7]) and supplied to *perf*.

We get the overall power consumption by measuring AC power at the outlet by means of different power sensors depending on the platform: a WattsUp sensor, with an accuracy of $\pm 1.5\%$, for low-power platforms, and an HP Intelligent Power Distribution Unit (iPDU), with an accuracy ± 1.5 Watt at current > 20 mA and accessible via SNMP, connected to the rest of the platforms. A separate driver has been developed for each sensor type. Note that measuring AC power at the outlet has been suggested as the most architecture independent and less intrusive method for power metering [19].

We gather also some custom system metrics, such as the CPU utilization per core (captured using the *psutil* Python library), and the number of active sockets, cores, and threads (calculated by cross-linking the information from `/proc/cpuinfo` and the CPU utilization per core).

Those additional metrics to the ones collected by *gmond* (i.e. performance counters, *psutil* metrics, and power metrics) are integrated into Ganglia by using a custom metric collection framework (implemented in Python) that injects them to *gmond* using *gmetric*, which allows each *gmond* instance within a cluster to read and store these new metrics as if they had been originally collected by *gmond*.

Most of the gathered metrics are the same for all the platforms. Only some performance counters can vary, in particular the raw counters that refer to micro-architectural events specific for each processor (e.g. number of floating-point instructions executed by the Streaming SIMD Extensions (SSE) in Intel processors).

Data is sampled with a different frequency depending on the platform: 0.2 Hz for low-power platforms and 0.5 Hz for the rest of platforms. We have set such frequencies based on the sampling speed of the power sensors used (1 second for the WattsUp and 0.5 seconds for the HP iPDU) and to keep the monitoring overhead low. The sampled value for each metric designates the number of events of such metric occurred during the corresponding sampling interval, which allows capturing how the metric varies over time.

3.3 Training micro-benchmarks

The training set comprises several micro-benchmarks that se-

lectively stress different components of the target platform (CPU, cache, main memory, network, and disk) at different intensity levels. We created a series of scripts that automatically execute those benchmarks on the different platforms.

3.3.1 CPU and cache memory

We stress the CPU and cache subsystems by using the following micro-benchmarks:

- Ibench (Integer, FP) [21]: SoI11 and SoI12 benchmarks have been adapted to perform different types of operations (floating-point, integer, and square root).
- Stress-ng CPU v.0.03.03 [13]: nearly 70 CPU-specific stress tests that exercise floating point, integer, bit manipulation, control flow, and cache thrashing.
- Sysbench CPU v.0.4.12 [14]: calculates prime numbers using 64-bit integers.
- Prime95 v.285 [5]: makes heavy use of integer, floating point, and SIMD instructions by computing Mersenne prime numbers. It is not supported in the *srv-arm* platform.
- Linpack-neon [11]: includes double precision and single precision vector operations for ARM v7 CPUs and it has replaced the Prime95 benchmark in the *srv-arm* platform.

We perform several executions of these micro-benchmarks increasing the number of threads involved (running several instances of the benchmark when needed). For those executions, we have increased the intensity for each thread from 0% to 100% through *cpulimit* [3] (excluding the Prime95 and Linpack-neon benchmarks). In such a way, we can stress different number of CPU cores and cache levels and analyze the impact on the power of technologies such as hyperthreading.

3.3.2 Main memory

We stress the main memory subsystem (both memory capacity and bandwidth) by using the following micro-benchmarks. They are 'main-memory-specific' benchmarks, but they also involve cache memory to some extent.

- Stress-ng VM v.0.03.03 [13]: over 20 virtual memory stress tests where different workers modify memory in predefined ways. We perform several executions increasing the number of workers.
- Pmbw v.0.6.2 [9]: measures the parallel memory bandwidth of multi-core machines by executing different modes of memory access (ranging from sequential scanning to pure random access and including different memory transfer sizes) with increasing array size and thread count (evenly dividing the array among threads). We perform several executions increasing the number of threads.
- STREAM v.5.10 [12]: measures sustainable memory bandwidth (in MB/s) for simple vector operations. We perform several executions controlling the array size and the number of threads.

3.3.3 Disk

We stress the disk subsystem with the following micro-benchmarks:

- Stress-ng hdd v.0.03.03 [13]: starts a variable number of workers spinning on write operations followed by unlinking the file. We perform several executions increasing the number of workers.
- Fio v.2.1.9 [4]: spawns a number of threads or processes doing a particular type of I/O actions: sequential reads, sequential writes, random reads, and random writes. We perform several executions increasing the bandwidth rate with steps of 10% of the maximum supported.

3.3.4 Network

We stress the network subsystem using the iperf v.3.0.1 [8] benchmark. We perform several executions increasing the bandwidth with steps of 10% of the maximum network capacity.

4. MODEL GENERATION

Figure 2 shows the procedure used to generate and validate the power model for each platform, which has been automated by using the *R* package, a programming language and software environment for statistical computing [15]. The different phases involved are described in this section.

First, the datasets obtained through the execution of the training micro-benchmarks are pre-processed to set the captured data into a suitable form to work with. This includes i) retrieving the data in a JSON format through the *gweb* component and converting them to *csv* files with the same internal structure, and ii) removing and fixing incomplete records or in a wrong format produced by failures of the monitoring system.

The set of representative features that will be used to generate the model must be then selected. Note that the monitoring system provides a large number of resource usage indicators, but many of them may not be necessary for modeling the power. Including them will unnecessarily slow down the modeling process and can affect the accuracy of the models. An initial selection of features is obtained by excluding those showing not significant variance in the training experiments. This selection is further refined by using a novel approach to analyze the training set, which assesses the relation of each feature with the power while capturing non-linear behaviors. Although the model will finally be built using a training set consisting of the union of all the micro-benchmarks, this analysis is done considering each resource usage indicator on each micro-benchmark separately, because it is harder to recognize correlations among metrics after joining all the training datasets. For instance, Figure 3 shows the relation between power consumption and L1 data cache loads considering the entire training set in *srv-xeon-2* platform. Several patterns can be appreciated, since each benchmark stresses a different resource at various intensities. Therefore, it would be very difficult to model the different patterns at the same time (i.e. with a single coefficient, if using linear regression). Thanks to the per-micro-benchmark analysis, our methodology is able to capture several correlations for each resource usage indicator.

For each resource usage indicator on each micro-benchmark, we perform a regression analysis to find the expression that shows best (in terms of the coefficient of determination) the relation of such metric with the power consumption in that particular micro-benchmark. To capture non-linear behaviors, we assess polynomial (including fractional exponents to represent nth-root relations) and logarithmic transformations of the metric. Note that including such transformations in the training dataset will steer the mining algorithm to recognize the correlation of such metric with the power. If a logarithmic function returns the best correlation, we apply the $log(metric)$ transformation to the data corresponding to that metric. If the relation fits better with a polynomial function, we apply the $metric^{\alpha}$ transformation to the data. The α parameter is obtained through the *nls* function from *stats* package for *R* language, which determines non-linear least-squares estimates of non-linear models. Note that α being 1 means that the metric does not need to be transformed. In

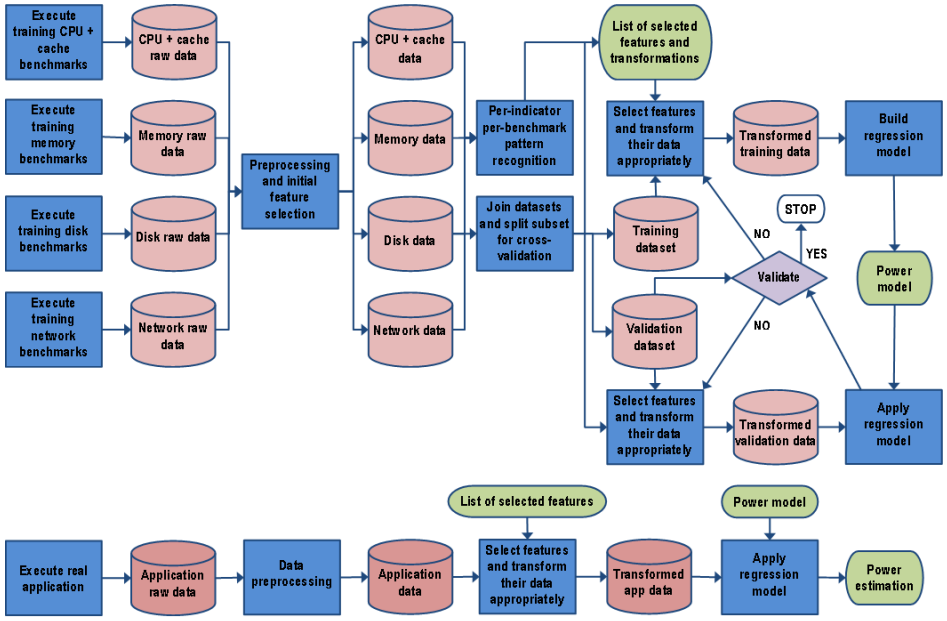

Figure 2: Model generation (top) and validation (bottom) procedures

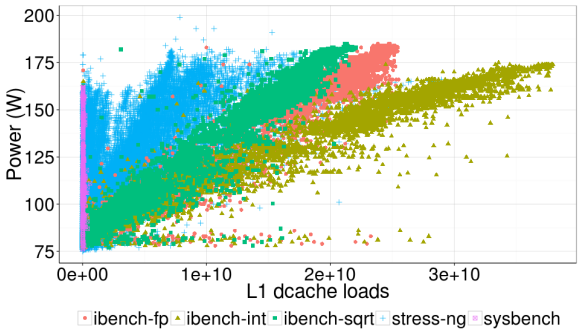

Figure 3: Relation between power consumption and L1 data cache loads for the entire training set in *srv-xeon-2*

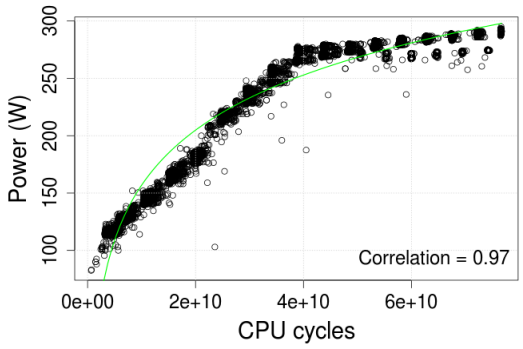

Figure 4: Relation between power consumption and CPU cycles for Prime95 micro-benchmark in *srv-xeon-2*

any case, if the correlation with the power is not significant (lower than 0.95), this resource usage indicator is not selected.

For example, as shown in Figure 4, the relation between the power and the number of CPU cycles for the Prime95 micro-benchmark in *srv-xeon-2* platform is best approximated by using a logarithmic function, so this is transformation that we apply to the CPU cycles data from that benchmark. Note that we do not support piecewise-defined functions, which could fit better in this example, because either human support is required to identify the breakpoints, which would make our methodology not automatic, or the automatic detection of breakpoints tends to overestimate the number of them, which would probably overfit the model. However, this does not significantly impact the resulting accuracy of our models.

The final training dataset to be supplied to the mining algorithm is generated by joining the datasets of all the micro-benchmarks, filtering the selected features, and performing the polynomial and logarithmic transformations studied before. We use then linear regression in order to fit our model (*lm* function from *stats* package for *R* language).

The resulting model can still include some features that do not determine the model significantly, basically because they have essentially the same behavior. We remove those features from the feature set before generating the final model. To this end, we compute the p-value of each feature regarding the power consumption. A large p-value (higher than 0.05) indicates a weak influence of the feature on the model, thus it can be discarded from the feature set.

5. MODELS VALIDATION

We validate the models derived for each hardware platform with real applications, which are commonly found in Cloud data centers, with different patterns of resource usage and energy consumption. Those applications are described in Section 5.1. As shown in Figure 2, we execute the applications in the target platform while capturing their performance indicators by means of the monitoring framework, we filter the selected features and transform the captured data as described in previous section, and we apply the power model of the corresponding platform using as input the transformed data. Note that we do not need to execute the entire application to get power estimations. They can be dynamically calculated over time

by using the captured data at each monitoring interval.

The validation includes a numerical assessment of the models accuracy as well as detailed time plots comparing real and estimated power consumption. We compare the accuracy of our models (labeled 'OURS' in the tables) with some models derived by means of the methodology proposed by McCullough et al. [30], which uses Lasso regression to automatically incorporate just enough features as are necessary. In particular, we use their methodology to build linear models (labeled 'LNLS' in the tables) and non-linear polynomial models (labeled 'PLLS' in the tables) by using the *glmnet* package to perform a Lasso regression on a filtered set of features after discarding those with low correlation with power consumption. As in their paper, the optimal value for the λ parameter is selected by cross validation on the training data and non-linear models are generated including polynomial terms with exponents from 1 to 3.

The accuracy of the models is computed by comparing the power estimations obtained from the model with the actual measured values by using the Mean Absolute Percentage Error (MAPE), which is calculated as in Equation 1.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} \left| \frac{P_{actual_i} - P_{modeled_i}}{P_{actual_i}} \right| * 100 \qquad (1)$$

Note that these accuracy values also include some error caused by external factors to the modeling procedure. For example, they are affected by the accuracy of the power sensor itself and by the multiplexing of hardware counters: the number of events that are physically counted during each sampling period is limited by the number of counters that can be simultaneously accessed and, during this period, the remaining events of the multiplexed event-set are estimated.

5.1 Validation workload

The workload used to validate the power models includes traditional HPC applications, but also novel Big Data programs, which have not been considered in the previous works on full-system power modeling. We have used CloudSuite and NAS benchmarks to provide such real-world application profiles.

CloudSuite 2.0 [24] comprises eight scale-out applications that feature real-world server software stacks and datasets. Refer to [2] for additional installation and configuration details.

- Data Analytics: performs machine learning tasks to analyze a 30 GB Wikipedia dataset by using Mahout libraries running on top of Hadoop. We used a smaller dataset (1.4 GB) in order to overcome the lack of available disk space in some of our platforms.
- Data Caching: simulates a Twitter caching server using a real Twitter dataset by relying on memcached, which is the most widely used data caching server. We ran the workload with eight client threads, 200 TCP/IP connections, and a get/set ratio of 0.8. We changed the number of requests per second from 10% to 90% with steps of 10% of the maximum throughput.
- Data Serving: implements a NoSQL system by means of Cassandra (an open-source NoSQL datastore) stimulated by the Yahoo! Cloud Serving Benchmark. As the standard benchmark creates very small databases, we have overridden the default property to load 10 million of records.
- Graph Analytics: performs data analysis on large-scale graphs by using the GraphLab software to run the TunkRank algorithm, which recursively computes the influence of Twitter users based on the number of their followers. We used a Twitter dataset with 11 million of vertices (Twitter users). This benchmark is not supported in *srv-arm* platform.
- Media Streaming: executes the Darwin Streaming Server stressed by a client emulator. We have tried several real-world scenarios by changing the number of clients simultaneously emulated (up to 7500).
- Software Testing: uses the Cloud9 software testing engine to run large-scale symbolic execution tasks in a Cloud environment. We ran the benchmark with multiple independent Cloud9 workers depending on the available cores of each machine. This benchmark is not supported in *srv-arm* platform.
- Web Search: uses the Nutch/Lucene search engine stimulated by a client representing real-world scenarios. Server and client were installed in two different machines. In addition to the default client, we also used Apache JMeter [1] to emulate different number of clients accessing the search engine simultaneously (up to 100 clients).
- Web Serving: uses CloudStone running the PHP version of Olio (a Web 2.0 social-events application) to generate dynamic web traffic. The benchmark consists of three main components: a web server, a database backend, and a client to emulate real-world accesses to the web server. Client and web server (together with the database) run in two different machines. We ran the benchmark several times changing the number of concurrent users that send requests to the web server (scaling factor up to 200).

NAS Parallel Benchmarks (NPB v.3.3) [17] were used as HPC representatives. We executed the serial, MPI, and OpenMP implementations of three pseudo-applications implementing complex matrix solvers, namely Block Tri-diagonal solver (BT), Scalar Penta-diagonal solver (SP), and Lower-Upper Gauss-Seidel solver (LU), using problem sizes C and D (the latter only for OpenMP implementations in high-performance servers). Refer to [10] for more details on each problem size.

The number of processes of the MPI implementations have been changed (without exceeding the number of CPUs available) in order to try different configurations. Applications SP and BT require a number of processes that must be a square (4, 9, 16, ...), while application LU supports a power-of-2 number of processes (2, 4, 8, ...). OpenMP implementations have been executed using as many threads as the maximum number of cores available in each system.

5.2 Validation of power models for high-performance platforms

Table 2 shows the accuracy of the models derived for the Intel Xeon platforms. Although the high complexity of these platforms raises the difficulty of modeling them (mainly due to the high number of cores they have), the average MAPE of our models is 4.3% for *srv-xeon-1* and 5.7% for *srv-xeon-2* considering all the set of benchmarks. This rate is very accurate considering that we do not focus on any specific type of application but our methodology can be applied to a wide set of applications in different platforms. Lasso models provide good accuracy for some applications, especially NAS HPC jobs in *srv-xeon-1*, but fail to model others, especially CloudSuite benchmarks in *srv-xeon-2*. Polynomial Lasso model cannot achieve the same accuracy as its linear counterpart for NAS HPC jobs in *srv-xeon-2* either.

Table 2: Validation errors for Intel Xeon platforms

Benchmark	*srv-xeon-1*			*srv-xeon-2*		
	LNLS MAPE	PLLS MAPE	OURS MAPE	LNLS MAPE	PLLS MAPE	OURS MAPE
Data Analytics	9.62	6.52	7.32	6.08	13.82	5.58
Data Caching	6.54	5.31	3.44	21.01	15.97	5.59
Data Serving	3.25	2.31	0.95	19.75	14.29	3.27
Graph Analytics	7.69	14.31	3.58	3.88	14.93	5.17
Media Streaming	5.67	3.23	0.76	33.81	4.47	1.59
Software Testing	6.36	9.91	5.04	6.91	5.83	6.92
Web Search	85.48	210.08	6.29	47.71	51.39	8.55
Web Serving	4.4	1.34	0.8	8.39	5	3.07
BT (MPI,C)	2.65	8.95	4.63	8.42	12.2	9.19
LU (MPI,C)	6.6	11.14	5.46	4.98	6.4	8.36
SP (MPI,C)	6.55	6.24	9.52	14.72	14.9	8.6
BT (OMP,D)	5.38	4.47	2.97	5.46	20.9	5.3
LU (OMP,D)	8.9	9.36	3.21	3.69	24.93	2.66
SP (OMP,D)	4.14	4.7	5.56	5.44	20.41	6.06
BT (SER,C)	2.42	2.23	2.01	8.84	13.75	6.68
LU (SER,C)	3.79	4	6.17	6.47	16.47	4.66
SP (SER,C)	2.3	2.18	4.84	4.64	15.85	6.3

Figure 5 and Figure 6 present a comparison of estimated and real power consumption for the Media Streaming benchmark running in *srv-xeon-1* and the Data Analytics benchmark running in *srv-xeon-2*, respectively, demonstrating that the values predicted with our models closely follow the real measurements. The Media Streaming benchmark is executed four times with different number of concurrent clients. The Data Analytics benchmark comprises the generation of a classifier model (up to sample 2500) and the testing of such model.

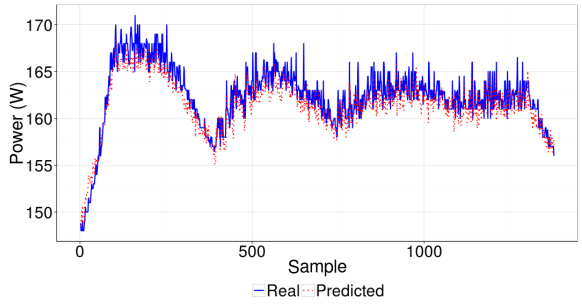

Figure 5: Comparison of power consumption predicted with our model and real measurements on *srv-xeon-1* platform running Media Streaming benchmark

Table 3 summarizes the accuracy of the models derived for the AMD Opteron platforms. Despite the high range of power consumption that can be generated by the two platforms (up to 400 W for *srv-opt-1* and 360 W for *srv-opt2*), the average error of our models is still very low: 5.4% for *srv-opt-1* and 5.7% for *srv-opt-2*. Predictions errors are noticeable for Lasso models, especially with CloudSuite benchmarks.

Figure 7 and Figure 8 compare the power consumption estimation and the actual power for the NAS BT (MPI implementation and class C) running in *srv-opt-1* and the Web Serving benchmark executed in *srv-opt-2*, respectively. The NAS BT benchmark is executed three

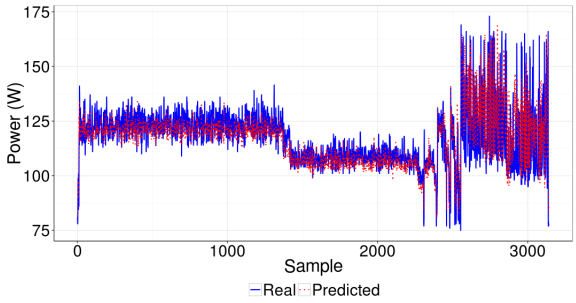

Figure 6: Comparison of power consumption predicted with our model and real measurements on *srv-xeon-2* platform running Data Analytics benchmark

Table 3: Validation errors for AMD platforms

Benchmark	*srv-opt-1*			*srv-opt-2*		
	LNLS MAPE	PLLS MAPE	OURS MAPE	LNLS MAPE	PLLS MAPE	OURS MAPE
Data Analytics	8.28	10.82	6.3	19.25	13.08	9.79
Data Caching	42.81	7.18	6.45	10.97	71.83	4.84
Data Serving	6.93	3.67	9.36	17.76	60.55	9.56
Graph Analytics	12.39	12.25	8.65	13.63	11.16	6.57
Media Streaming	21.07	12.26	1.82	7.87	4.42	2.58
Software Testing	6.87	5.07	3.01	9.45	5.25	3.46
Web Search	6.43	2.87	2.29	11.46	8.71	4.99
Web Serving	13.74	5.37	6.65	7.11	4.87	3.46
BT (MPI,C)	7.54	2.2	1.91	6.81	3.09	3.13
LU (MPI,C)	8.87	11.93	5.28	11.08	6.65	8.22
SP (MPI,C)	8.69	8.84	6.1	13.3	6.86	10.36
BT (OMP,D)	8.47	6.09	5.56	5.15	5.03	5.93
LU (OMP,D)	10.77	2.35	3.2	6.68	5.72	3.43
SP (OMP,D)	4.68	3.09	3.1	4.65	3.98	4.16
BT (SER,C)	9.95	17.31	8.27	19.19	15	5.59
LU (SER,C)	10.31	17.23	8.36	12.66	10.02	5.3
SP (SER,C)	9.01	15.06	5.63	10.67	6.76	4.87

times, the first with 25 processes (up to sample 2100), the second with 4 processes (from sample 2100 to 2375), and the third with 9 processes (from sample 2375 to the end). The Web Serving benchmark is executed nine times with different number of concurrent clients. Results of our models for the AMD Opteron platforms are in line with the Intel platforms previously analyzed and confirm that the proposed methodology can produce good results for different server families and configurations.

5.3 Validation of power models for low-power platforms

The accuracy of our models in low-power platforms is also very promising (as shown in Table 4). The average MAPE for *srv-atom* is 2.6% and for *srv-arm* is 5.2%, which is consistent with the results in the previous section, despite the platforms are very different and the range of power consumption is much lower. Whereas Lasso models provide very good accuracy for NAS HPC jobs in *srv-atom*, they show important errors for CloudSuite benchmarks. They also provide low accuracy in *srv-arm* for all the applications, especially Big Data ones.

The accuracy of our models can be seen in Figure 9, which shows

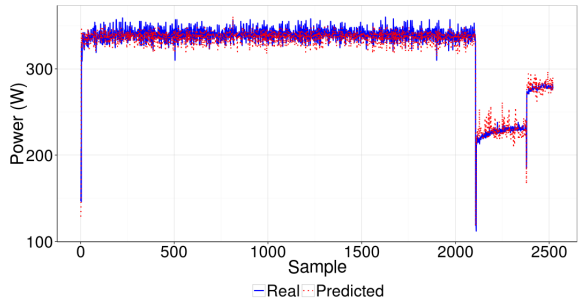

Figure 7: Comparison of power consumption predicted with our model and real measurements on *srv-opt-1* platform running NAS BT (MPI,C) benchmark

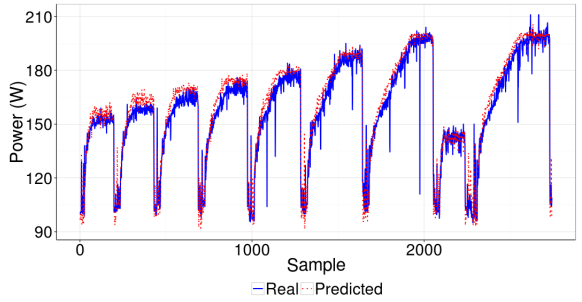

Figure 8: Comparison of power consumption predicted with our model and real measurements on *srv-opt-2* platform running Web Serving benchmark

a comparison of the estimated and real power for the Software Testing benchmark running in *srv-atom*, and Figure 10, which shows the power estimation for the Data Caching benchmark running in *srv-arm*. The Software Testing benchmark is executed four times, with 1, 2, 3, and 4 worker tasks, respectively. The Data Caching benchmark is executed nine times with different number of requests served per second. The estimations closely follow the real measurements with deviations of less than 1 W, which is lower than the accuracy of the power sensor itself.

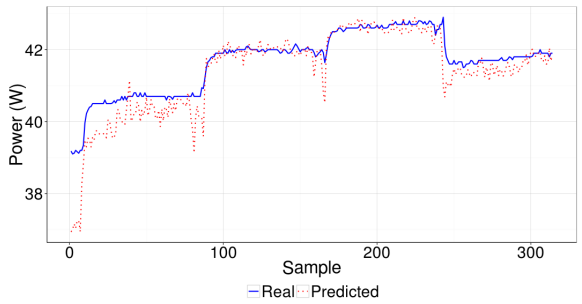

Figure 9: Comparison of power consumption predicted with our model and real measurements on *srv-atom* platform running Software Testing benchmark

6. CONCLUSIONS

In this paper, we have proposed a platform- and application-agnostic methodology for full-system power modeling in heterogeneous data centers based on collecting power and resource usage measurements while running a special training workload and fitting them through machine learning. Our methodology overcomes the limitations of the previous works. It derives a single model per platform by systematically selecting a minimum set of resource usage indicators and extracting complex relations among them that capture the impact on energy consumption of all the resources.

We have demonstrated our methodology by generating power models for heterogeneous platforms with very different power consumption profiles ranging from high-performance server architectures based on Intel Xeon and AMD Opteron to low-power architectures based on Intel Atom and ARM Cortex-A. Our experiments with real Cloud applications with different patterns of resource usage and energy consumption, represented by CloudSuite and NAS benchmarks, show that such models provide high accuracy (around 5% of average estimation error) and that power consumption estimations closely follow real power measurements in the time plots of the executions.

As part of our future work, we will enhance our methodology to model the power consumption of individual virtual machines and containers. We also plan to use the power models to drive energy-

Table 4: Validation errors for low-power platforms

Benchmark	*srv-atom*			*srv-arm*		
	LNLS MAPE	PLLS MAPE	OURS MAPE	LNLS MAPE	PLLS MAPE	OURS MAPE
Data Analytics	1.08	1.17	1.79	31.43	713.5	6.18
Data Caching	8.28	176.5	4.17	18.45	144.44	4.33
Data Serving	2.4	5.92	1.83	20.07	42.01	4.09
Graph Analytics	7.67	3.70	6.52	-	-	-
Media Streaming	62.49	37.06	1.94	24.86	63.64	6.3
Software Testing	1.99	0.51	0.81	-	-	-
Web Search	5.36	4.46	1.6	16.97	28.46	5.72
Web Serving	10.74	4.64	3.14	19.93	11.62	15.61
BT (MPI,C)	0.99	0.85	1.96	10	17.42	3.73
LU (MPI,C)	1.92	2.03	4.09	12.87	15.68	4.69
SP (MPI,C)	1.88	1.60	2.41	13.81	18.78	3.26
BT (OMP,C)	2.06	1.99	2.03	5.95	11.97	5.98
LU (OMP,C)	3.05	2.08	1.44	9.6	12.19	3.26
SP (OMP,C)	4.84	4.18	5.12	15.66	19.77	2.73
BT (SER,C)	1.05	0.82	1.75	6.8	12.81	5.32
LU (SER,C)	2.02	2.13	2.38	10.23	12.73	3.87
SP (SER,C)	2.41	2.65	2	15.76	19.9	2.82

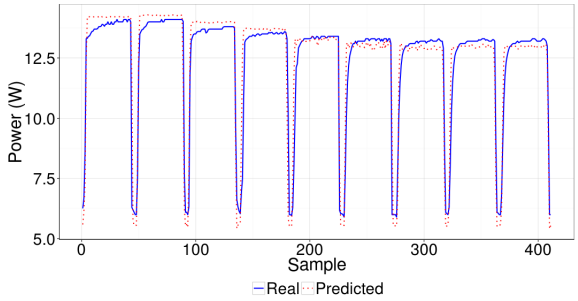

Figure 10: Comparison of power consumption predicted with our model and real measurements on *srv-arm* platform running Data Caching benchmark

aware scheduling policies that select the most convenient host where to execute each application.

Acknowledgements

This work is supported by the Spanish Ministry of Economy and Competitiveness under contract TIN2015-65316-P, by the Generalitat de Catalunya under contract 2014-SGR-1051, and by the European Commission under FP7-SMARTCITIES-2013 contract 608679 (RenewIT) and FP7-ICT-2013-10 contracts 610874 (ASCETiC) and 610456 (EuroServer).

7. REFERENCES

[1] Apache JMeter. http://jmeter.apache.org/.

[2] CloudSuite Benchmarks. http://cloudsuite.ch/.

[3] cpulimit. https://github.com/opsengine/cpulimit.

[4] fio - Flexible I/O Tester. http://git.kernel.dk/?p=fio.git.

[5] Free Mersenne Prime Search Software: Prime95. http://www.mersenne.org/download/.

[6] Host sFlow. http://sflow.net/.

[7] How to monitor the full range of cpu performance events. http://www.bnikolic.co.uk/blog/hpc-prof-events.html.

[8] iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool. https://github.com/esnet/iperf.

[9] pmbw - Parallel Memory Bandwidth Benchmark/Measurement. http://panthema.net/2013/pmbw/.

[10] Problem Sizes and Parameters in NAS Parallel Benchmarks. http://www.nas.nasa.gov/publications/npb_problem_sizes.html.

[11] Roy Longbottom's Raspberry Pi & Raspberry Pi 2 Benchmarks: Linpack NEON Benchmark. http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm#anchor24b.

[12] STREAM: Sustainable Memory Bandwidth in High Performance Computers. https://www.cs.virginia.edu/stream/.

[13] stress-ng. http://kernel.ubuntu.com/~cking/stress-ng/.

[14] sysbench. https://github.com/akopytov/sysbench.

[15] The R Project for Statistical Computing. https://www.r-project.org/.

[16] Make IT Green: Cloud Computing and its Contribution to Climate Change. Technical report, Greenpeace International, 2010.

[17] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks - Summary and Preliminary Results. In *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing (SC'91), Albuquerque, NM, USA*, pages 158–165, November 1991.

[18] W. L. Bircher and L. K. John. Complete System Power Estimation Using Processor Performance Events. *IEEE Transactions on Computers*, 61(4):563–577, April 2012.

[19] L. Cupertino, G. Da Costa, and J.-M. Pierson. Towards a Generic Power Estimator. *Computer Science - Research and Development*, 30(2):145–153, May 2015.

[20] G. Da Costa and H. Hlavacs. Methodology of Measurement for Energy Consumption of Applications. In *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing (GRID'10), Brussels, Belgium*, pages 290–297, October 2010.

[21] C. Delimitrou and C. Kozyrakis. iBench: Quantifying Interference for Datacenter Applications. In *Proceedings of the 2013 IEEE International Symposium on Workload Characterization (IISWC'13), Portland, OR, USA*, pages 23–33, September 2013.

[22] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-System Power Analysis and Modeling for Server Environments. In *Proceedings of the 2nd Workshop on Modeling, Benchmarking and Simulation (MoBS'06), Boston, MA, USA*, June 2006.

[23] X. Fan, W.-D. Weber, and L. A. Barroso. Power Provisioning for a Warehouse-sized Computer. *SIGARCH Computer Architecture News*, 35(2):13–23, June 2007.

[24] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the Clouds: a Study of Emerging Scale-out Workloads on Modern Hardware. In *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '12), London, UK*, pages 37–48, March 2012.

[25] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., and R. Bianchini. Energy Conservation in Heterogeneous Server Clusters. In *Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'05), Chicago, IL, USA*, pages 186–195, June 2005.

[26] M. Jarus, A. Oleksiak, T. Piontek, and J. Weglarz. Runtime Power Usage Estimation of HPC Servers for Various Classes of Real-life Applications. *Future Generation Computer Systems*, 36:299–310, July 2014.

[27] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual Machine Power Metering and Provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC'10), Indianapolis, IN, USA*, pages 39–50, June 2010.

[28] J. Koomey. Growth in Data Center Electricity Use 2005 to 2010. Technical report, Analytics Press, Oakland, CA, USA, August 2011.

[29] M. Massie, B. Li, B. Nicholes, V. Vuksan, R. Alexander, J. Buchbinder, F. Costa, A. Dean, D. Josephsen, P. Phaal, and D. Pocock. *Monitoring with Ganglia*. O'Reilly Media, Inc., 1st edition, 2012.

[30] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta. Evaluating the Effectiveness of Model-based Power Characterization. In *Proceedings of the 2011 USENIX Annual Technical Conference (ATC'11), Portland, OR, USA*, pages 159–172, June 2011.

[31] C. Mobius, W. Dargie, and A. Schill. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. *IEEE Transactions on Parallel and Distributed Systems*, 25(6):1600–1614, June 2014.

[32] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In *Proceedings of the 3rd ACM Symposium on Cloud Computing (SoCC'12), San Jose, CA, USA*, pages 7:1–7:13, October 2012.

[33] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A Comparison of High-level Full-system Power Models. In *Proceedings of the 2008 Workshop on Power Aware Computing and Systems (HotPower'08), San Diego, CA, USA*, December 2008.

[34] M. Witkowski, A. Oleksiak, T. Piontek, and J. Weglarz. Practical Power Consumption Estimation for Real Life HPC Applications. *Future Generation Computer Systems*, 29(1):208–217, January 2013.