Cluster Comput (2014) 17:1279–1293
DOI 10.1007/s10586-014-0373-0

Empirical and analytical approaches for web server power modeling

Leonardo Piga · Reinaldo A. Bergamaschi · Sandro Rigo

Received: 28 May 2013 / Revised: 23 January 2014 / Accepted: 21 March 2014 / Published online: 19 April 2014

Abstract Power-aware computing has emerged as a significant concern in data centers. In this work, we develop empirical models for estimating the power consumed by web servers. These models can be used by on-the-fly power-saving algorithms and are imperative for simulators that evaluate the power behavior of workloads. To apply power saving methodologies and algorithms at the data center level, we must first be able to measure or estimate the power and performance of individual servers running in the data centers. We show a novel method for developing full system web server power models that reduces non-linear relationships among performance measurements and system power and prunes model parameters. The web server power models use as parameters performance indicators read from the machine internal performance counters. We evaluate our approach on an AMD Opteron-based web server and on an Intel i7-based web sever. Our best model displays an average absolute error of 1.92 % for Intel i7 server and 1.46 % for AMD Opteron as compared to actual measurements, and 90th percentile for the absolute percent error equals to 2.66 % for Intel i7 and 2.08 % for AMD Opteron.

1 Introduction

The shift towards an increasing demand in computational resources has forced companies to build facilities hosting hundreds of thousands of computers called data centers. This comes at a price of higher operational costs which include the management of these installations, the electricity costs, and environmental impacts due to the increased power consumption. Energy consumption has emerged as an important concern to the total cost of ownership of data centers [3,12] making energy-efficiency one of the key metrics in the design of large-scale services. The power consumed by servers dictates the power required by auxiliary equipment and cooling; consequently, reducing server power impacts directly on the overall data center power.

Electric utilities and electric installation equipments also impose limits on data center peak power. To avoid surpassing these thresholds, data center administrators keep a safety margin when configuring the server units across the facility. This approach is conservative and usually overestimates the actual power consumed by servers, which rarely operate at their full capacity [12]. Recent studies have emerged to help maintain an average power budget, assert peak power constraints, or optimize performance under a power cap [3,10,12,16]. Most of these techniques require monitoring the power consumption to apply their optimizations.

Most contemporary high-end processors feature sensors for monitoring energy consumption [27]; however, in commodity processors, which are prevalent in Internet-based data centers, this is typically not the case. Nevertheless, these processors usually feature event counters that can be used to estimate power consumption. For example, the second generation Intel Core microarchitecture uses performance events as proxies of power consumption [31]. The technique is based on reading hundreds of internal performance counters and on applying activity energy costs to each event to estimate power. Previous studies have used these probes to indirectly estimate power consumption [4,6,11,17,18]. Their usual approach is to derive linear power models based on the

L. Piga (✉) · R. A. Bergamaschi · S. Rigo
Institute of Computing, University of Campinas UNICAMP, Av. Albert Einstein, 1251 - Cidade Universitaria, Campinas, SP 13083-852, Brazil
e-mail: lpiga@ic.unicamp.br

R. A. Bergamaschi
e-mail: rberga@ic.unicamp.br

S. Rigo
e-mail: sandro@ic.unicamp.br

usage numbers collected for the processor sub-components (e.g. caches and branch predictor).

This paper advances the state-of-the-art in this area by presenting power models for two systems considering all their major parts (i.e. processors, disks, network, memory, and other motherboard components) and all its software stack (when running as web servers) for their different CPU core voltage and frequency states (also known as *P-states*). These models can be used for commodity systems for on-the-fly power-saving algorithms, and among other applications, they can also be used by simulators which evaluate the power behavior of workloads. Our models have been used in a data center simulator to guide the implementation of power/performance optimization algorithms through voltage and frequency state assignment [5]. The models are also used in a global power and performance optimization algorithm.

The power models are developed by using different methods, such as, linear regression, cluster analysis, and machine learning techniques having hardware events (e.g. number of instructions, unhalted cycles, cache misses) and system level measurements (e.g. page-faults, number of context switches) as proxies for power. However, linear power models based on system measurements and performance counters exhibit issues that need to be investigated: excess of parameters and non-linear relation among power and the associated factors (as is the case for I/O bound workloads).

The fewer the number of parameters of a model the faster the power estimation; therefore, we prune model parameters by using a correlation-based feature selection (CFS) [13] algorithm for choosing a subset of them that is most correlated to the power measurements. This approach has reduced the number of parameters preserving accuracy and precision when compared to a model using all the parameters.

To allow the use of power models based on linear regression on these types of workloads, we use k-means clustering to group up the performance measurements. Linear regression is applied on each cluster to dismiss the non-linear relationship among server measurements and power consumption improving model precision when compared to the other models. From our knowledge, this is the first work that applies CFS and k-means clustering to improve linear regression-based power models for computing systems.

The main contributions of this paper are as follows:

- We develop accurate empirical models for estimating the power consumed by web servers considering all their major parts, such as processors, disk, memory, network, and other motherboard components.
- The power of processors is characterized for their different *P-states* and models are developed for each *P-state*.
- We apply CFS to prune correlated model parameters and k-means clustering to soften non-linear relationship among server measurements and power consumption on linear power models improving their accuracy.

The remainder of this paper is organized as following: Section 2 comments out the related work. Section 3 introduces power characterization methodology to measure power consumption of system computers. Section 4 shows the characterization model, Sect. 5 shows our results. Finally, Sect. 6 presents the conclusions.

2 Related work

Linear models are easy to implement, simple to develop and use, fast to run on simulators, and have shown suitable for CPU-bound workloads [4,6,11,17,18]. Previous works on power modeling have used performance monitoring counters (PMC) as proxies to estimate CPU power. Bellosa [4] showed that CPU power correlates to floating point operations, L2 cache references, and memory references. His work was one of the first to propose the use of PMCs to create an energy-aware scheduler.

Joseph and Martonosi [18] introduced a model capable of estimating the power consumption for the processor and its sub-components by using PMCs and some heuristics based on capacitance models. Though accurate, such information might not be available for all processors. Isci and Martonosi [17] introduced a model for the Pentium IV processor that did not rely upon circuit-level information. However, their model required more then 15 PMCs, even though such a large number of counters is not usually available at the same time on most modern processors.

Contreras and Martonosi [11] used PMCs as inputs to a linear model for an Intel mobile processor at three voltage and frequency levels. Their model was validated with benchmarks representing embedded systems. Bertran et al. [6] used PMCs to build power models for contemporary Intel multi-core processors considering all voltage and frequency levels available. The works listed so far focused mainly on CPU benchmarks, whose applications fully utilize the processor. Because web servers typically do not characterize CPU-bound workloads due to the high number of I/O operations performed by such servers, there is pressure on the shared resources creating non-linear effects among power and system measurements.

Chen et al. [9] presented performance and power models in a multi-programmed multi-core environment by addressing the problem of time sharing. The performance model is based on the cache access pattern. They proposed a power model using neural network and another using linear regression on CPU performance counters. The reported prediction error is 3.2 % for the former and 3.8 % for the later.

Lewis et al. [22] also proposed power models using PMCs to enable dynamic control of thermal footprint. They modeled the computer using system of deterministic differential equation in which solution is estimated via time-series approximation. They reported prediction error between 1.6 and 3.3 % for both systems that they evaluated.

In our best web server Power model, the prediction error is 1.92 % for the Intel i7 server and 1.46 % for the AMD Opteron. Therefore, our proposed technique is simpler and equivalent in terms of accuracy to the other alternatives [9, 22].

Rivoire et al. [29] have compared several high-level full-system power models. They used a variety of workloads and architectures and also observed non-linear effects among CPU power and performance measurements due to bottlenecks on shared resources. We soften the non-linear effects among CPU power and performance by using k-means clustering. In addition, our power measuring device allows power breakdown for the individual system components.

Another component that should be taken into account when dealing with web servers is the hard disk drive. Carrera et al. [8] proposed the usage of disks with multiple rotation speeds to reduce the energy consumption in data centers. The authors used power values from data-sheets instead of actually measuring power. They showed that SCSI hard disks can account for up to 24 % of the overall energy consumption of a server. Besides, if a server is built with a higher number of disks, this fraction can increase to 77 %. In our experiments with web servers, we observe that hard disk power account for up to 20 % but does not exhibit wide variation allowing this power component to be modeled as a constant value.

Zedlewski et al. [32] developed tool called *Dempsey*, which can simulate disk operations in order to estimate the power consumption for a given workload. Their model was built with real power measurements. However, the chosen modeling parameters were based upon disk information that might not be available for all classes of hard disks.

In addition to the works above, some authors introduced models for estimating the full system power consumption. Bohrer et al. [7] studied power characterization of web servers considering a hypothetical support to dynamic voltage and frequency scaling (DVFS). They developed a web server simulation tool which could predict the power consumption based on web requests and CPU cycles. The adopted workload was generated from LOG files and static web content. Such workload is not compatible with modern web content, which heavily rely on dynamic content for rendering the pages.

3 Experimental methodology

This section introduces the power characterization methodology to measure the power consumption of commodity system computers. The workloads are SPECint2006 and SPECweb2009. Our main target is web server models, however, we use SPECint2006 to stimulate higher activity level on the CPU. We develop models for two different commodity systems: (1) A server with Intel i7 860 processor, 4 GB of memory, and an Western Digital serial ATA hard disk of 500 GB and 7,200 rpm running Ubuntu 9.10×86. (2) A server with an AMD Opteron 6168 processor containing 12 cores, 16 GB of memory, and a Seagate serial ATA hard disk of 1 TB and 7,200 rpm with Ubuntu 10.04 × 64 as operating system. All servers run the Apache 2.2.16 web server configured to use threads and PHP 5.3.3 for serving SPECweb2009 [1].

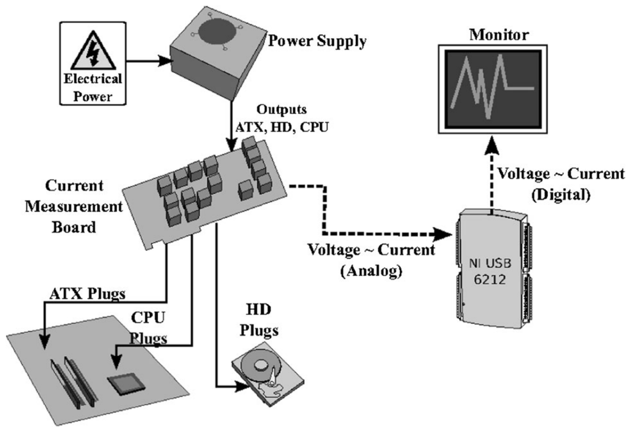


Fig. 1 Custom-made power measuring infrastructure designed to collect power on commodity computer systems

3.1 Measuring power

There are several methods to measure the power delivered to a given component. Some approaches use expensive intelligent power supplies or motherboards that have embedded power meters [20]. In other works [7,32], multimeters connected in series with the circuit are used for current measurements. This approach might add noise into the circuit and may not be suitable for high sampling rates.

We designed a custom-made measuring device similar to the infrastructure described in other works [25,26]. Piga et al. [26] fully describes our power measurement infrastructure shown in Fig. 1. It uses 15 current transducers (LTS 25-NP [21]) in series with the power lines which convert current into proportional values of voltage with accuracy of ±0.2 % and linearity of less than 0.1 %.

As shown in the Fig. 1, our board is installed into the server (plugged into a PCI slot). ATX, HD, and CPU plugs from the power supply are connected to the board's inputs. The output of the transducers are attached to a 16-bit data acquisition system (National Instruments NI USB–6212 [14]) that is capable of acquiring 400k samples per second across all channels. Since 15 channels need to be sampled, the actual sampling rate is 25k per second. Each ATX positive wire is connected in series to a transducer. Given that the voltages of these wires are known upfront, by measuring the currents that flow through them, it is possible to calculate power consumption, which is the product of voltage and current. Eventually, the data are read and stored by a monitoring computer from the acquisition device.

3.2 Collecting system performance

Hardware events and operating system measurements representing higher level system activity are collected to be used as proxies for power. Operating system measurements are important especially for the web server application where many processes of the same type are running in parallel. `Perf` [28] utility is responsible for assessing these events. The models correlate the collected performance rates (events per second) to the power measurements. Section 5 explains how these metrics affect the power models. The metrics are as follows:

3.2.1 Instructions retired per second

The CPI (Cycles Per Instructions) value or the equivalent BIPS (Billions of Instructions per Second) are directly correlated to CPU activity level making them important parameters for power models.

3.2.2 Unhalted cycles per second

This also represents CPU activity level. Higher values are expected when more functional units are working; hence, affecting CPU power consumption.

3.2.3 Last level (L3) cache references per second

Under different *P-states* (i.e. different frequencies) and different workloads, the last-level cache references per unit of time can change significantly, making it an important parameter in power models.

3.2.4 Last level (L3) cache misses per second

When there is a processor cache miss, its pipeline stalls affecting power consumption. This parameter also reflects the demand for memory resources since a miss in the last level cache requires an access to the main memory.

3.2.5 Page faults per unit of time

This metric captures the demand for memory resources with low level of temporal locality, reducing the activity level in the node.

Table 1 Disk activity parameters

Number of reads	Writes completed	Time reading	Time writing
Reads merged	Writes merged	I/O in progress	Time doing I/O
Sectors read	Sectors written	Time in queue	

3.2.6 Context switches per unit of time

This metric is used to capture the activity level of the operating system.

3.2.7 CPU migration per unit of time

This counter increases every time a process changes CPUs. When a process switches CPU, the instruction code cache is flushed, invalidated, and reloaded on the new CPU decreasing the system activity level.

3.2.8 CPU load

The time share that the CPU spends executing useful processes, that is, $(1.0 - t_{Idle}) \times 100\,\%$, where t_{Idle} is the share of time that the idle process is scheduled. The idle process is an infinite loop of halt instructions that changes the core to the HALT state when scheduled by the Operating System.

The disk parameters are collected by reading Linux device block statistics [23]. Table 1 presents the disk events available on the Linux system[1].

In order to synchronize the power numbers collected from our power measuring device with the performance measurements, we simultaneously sample power and performance on fixed rates (1 second in our experiments). When a given workload execution is completed, the monitoring software stores the resulting power values and performance measurements on disk for model generation. Collecting system-level statistics (such as number of page-faults and context switches) requires only small amount of processing because our sampling rate is one second. Moreover, measuring performance counters requires executing only one instruction to start sampling (`wrmsr`) and other to read the register values (`rdpmc`) [15]. We measured the overhead of collecting performance statistics while measuring power and observed that they are negligible corroborating to previous works [6,17,18].

[1] Reads/writes merged count the frequency that two 4 kB operations become one 8 kB operation

4 Characterization model

Our models are derived using regression techniques on experimental data using system-level measurements and performance counters as proxies for estimating the power consumption of the system for each CPU core frequency and voltage state. The following components contribute to the system power: CPU power; chipset, video board, network device, memories, and fans (miscellaneous components); and hard disk power. The total power of the system is computed by simply adding up the estimated power obtained for each component as shown by Eq. 1. Further discussions about the model accuracies and precisions are done in Sect. 5.

$$P_{Total} = P_{CPU} + P_{disk} + P_{miscellaneous} \tag{1}$$

Modern processors support DVFS, which can be exploited to optimize power and performance. Each voltage and frequency operating point represents a so-called power-saving state of the processor. The ACPI [2] is the operating system interface to these power-saving states, usually called *C-states* and *P-states*. C_0 is the active state and $C_1 \ldots C_{n-1}$ are the idle states. The deeper the state the higher the savings, at the cost of increasing time penalty for returning to the active state. When in idle states, the processor normally turns off some of its internal components. In C_0, it is possible to trade-off power consumption for performance by setting the processor to performance states (*P-states*) in accordance with the workload being executed.

On Linux, the *P-states* and their changing policy can be controlled by using the `sysfs` utility, which provides information about devices and drivers from the kernel space to the user space and interfaces to toggle them. There are operating system's utilities such as "On-demand Linux Governor" which can control the *P-state* transitions automatically based on the workload. There is also the "Userspace Linux Governor" utility which allows users to control the *P-states* of the CPU directly. Our models are developed in the context of the latter and they will be used as part of a global power and performance optimization policy [5]. The policy will choose the *P-States* for all CPU nodes and set them for each time-window.

P-states represent different operating points of frequency and voltage with different power and performance characteristics. The Intel i7 860 processor has 14 power states operating from 1.2 to 2.8 GHz. The AMD Opteron processor has 5 power states operating from 800 MHz to 1.9 GHz. We derived models for CPU (i.e. CPU cores, L1, L2 cache, and L3 cache) by measuring power and performance on each CPU *P-state*. Thus, each benchmark run is repeated by setting all cores of the CPU to the same *P-state*. This is not a limitation of the work, as the same methodology described in this paper can be used on a core by core basis.

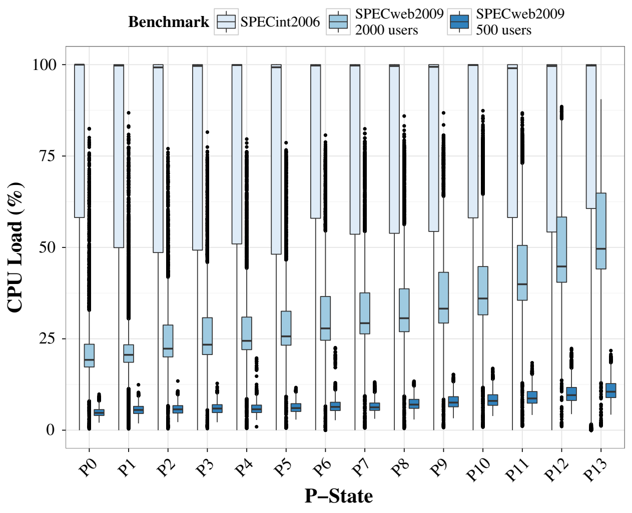


Fig. 2 Box plot for CPU load for SPECint2006, SPECweb2009 with 500 users, and SPECweb2009 with 2,000 users on each i7 *P-State*. Each benchmark stresses different ranges of the CPU load

To develop the models, we run SPECweb2009 and SPECint2006 according to the following strategy. For SPECweb2009, seven computers run as clients simulating 500 and 2000 real clients so as to emulate different server loads. This imposes different activity levels on the server directly impacting its CPU load and network traffic. Since SPECweb2009 benchmark is not able to exercise the full CPU capability due to its I/O boundedness characteristic, SPECint2006 is also executed to stimulate higher CPU activity levels. An instance of each SPECint2006 program runs in parallel on each core to stress all CPU cores. Therefore, by combining CPU-bound (SPECint2006) and I/O bound (SPECweb2009) data-sets, the complete spectrum of CPU utilization is covered.

Figure 2 presents box plots for SPECint2006 and for SPECweb2009 (running with 500 and 2,000 users) for all *P-states* on the Intel i7 server to illustrate the differences on the CPU Load on each benchmark. The plot shows that SPECweb2009 is not able to achieve CPU Load values higher than 80 % even when considering the outliers. On the other hand, CPU Load for SPECint2006 is concentrated from 60 to 100 % with median around 98 %. Therefore, by combining both benchmarks we can cover all CPU Load spectrum.

Considering the SPECweb2009 benchmark in Fig. 2, the outliers for the CPU-utilization are related to the nature of the web server application. For example, a Deposit operation is different from a Login operation, resulting in different CPU usages. Moreover, certain operations are more frequent depending on the operation mix, resulting on the outliers.

Next, we analyze the power variation on CPU, disk, and miscellaneous components. Figure 3 shows the power variation in the box plots. The CPU is the critical component of total power since it has the most contribution to total power

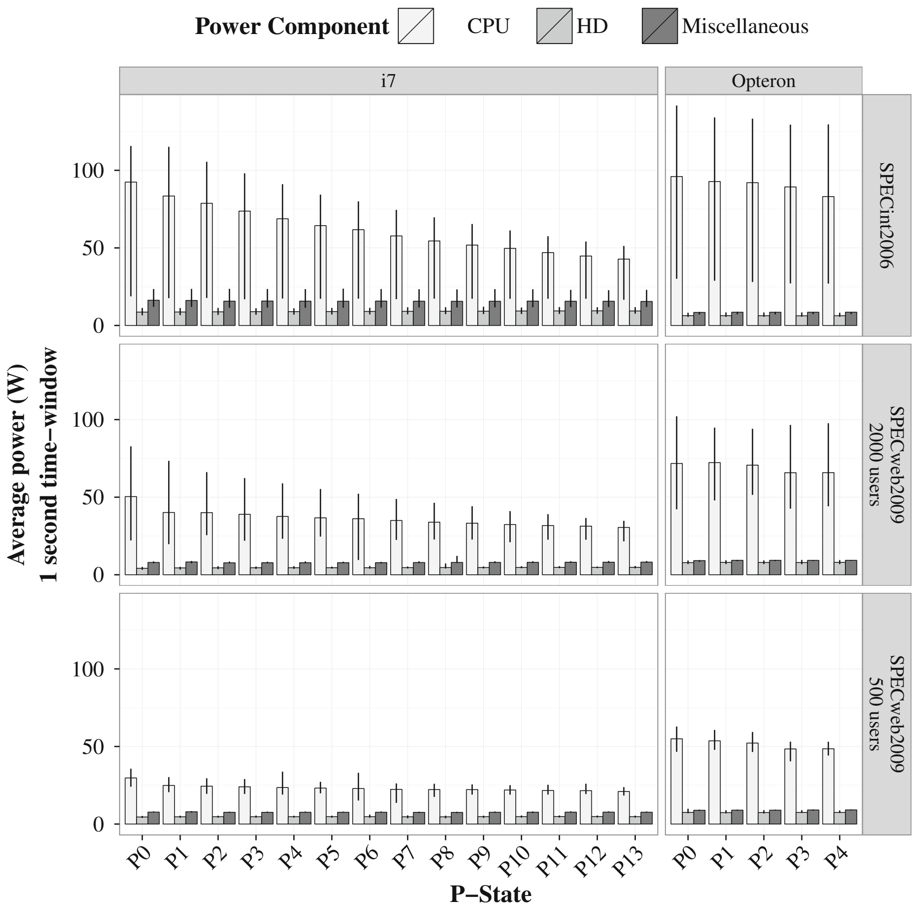


Fig. 3 Average device power measurements for SPECint2006, SPECweb2009 with 500 users, and SPECweb2009 with 2,000 on each *P-State* for AMD Opteron and Intel i7. The *lines* represents the minimum and the maximum device power measurement. CPU presents the most variation and has the most contribution for total power

and is responsible for the most power variation; consequently, it requires more detailed power models.

The work described by Zedlewski et al. [32] presents models using disk time reading and disk time writing to estimate disk power. Following this idea, at first, we developed disk power models based on the disk usage parameters that estimate the serial ATA disk power consumption. We used linear-regression to model disk power as $P_{disk} = a_0 + \sum_{i=1}^{9} a_i \cdot p_i$ where p_i's are the model parameters and a_i's their associated coefficient values shown in Table 2. This model exhibited an average error of 3.5 %

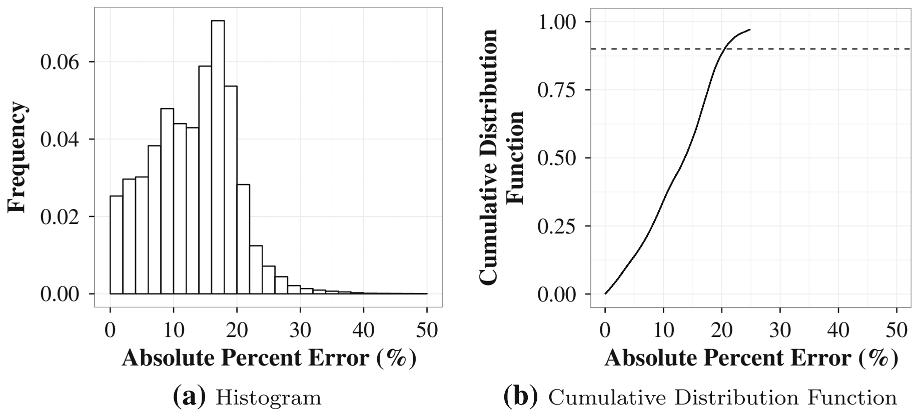
However, we observed that disk power varies very little and the standard deviations are less than 5 % of the mean values for this power component. Figure 3 shows that the disk power is concentrated between 8.0 and 9.0 W for the Opteron-based system and 5.0–6.0 W for the Intel i7-based system. The disk power consumption is responsible for only 10–12 % of the total power; thus, we model disk power as a constant value. The disk constant model displayed the same average error of 3.5 % obtained with the detailed model and it is simpler than the linear regression-based disk model. Hence, the constant model was chosen to be used in the full-system power model.

Table 2 Disk power model for the AMD Opteron server

i	p_i	a_i
0	Constant	8.622
1	Reads per s	0.002
2	Reads merged per s	−0.031
3	Sectors read per s	-9.555×10^{-6}
4	Time reading per s	-54.95×10^{-6}
5	Writes per s	-42.22×10^{-6}
6	Writes merged per s	-70.75×10^{-6}
7	Sectors written per s	10.18×10^{-6}
8	Time writing per s	-9.289×10^{-6}
9	Time I/O per s	0.002×10^{-6}

We grouped all other computer components such as chipset, video board, network device, memories, and fans in a power component called miscellaneous components. Following the same argument used for disk, miscellaneous components power has small contribution to total power and small variance; hence, we also model this component as a constant value equal to the average power of all points.

Fig. 4 Full-system power model for Intel i7 using the parameters described in Sect. 3.2. **a** presents the distribution for the absolute percent error, **b** presents the cumulative distribution function and the 90th percentile of the absolute percent error in *dashed line*



On the AMD Opteron server, the CPU contribution to total power is even higher compared to disk and miscellaneous component power. Therefore, considering constant power models for these components has even smaller impact on total model accuracy when compared to the Intel i7 server.

5 Experimental results

This section presents the results for our experiments and the evaluation of our power models. We split up the data-set into training set (50 % of the points) for building the models and testing set (remaining 50 % of the points) for model validation. We build the power models incrementally starting from a global power model (GPM), which does not distinguish *P-states* nor application running on the server and then we apply enhancements to improve accuracy and precision of the models.

We observe that model generality, excess of parameters, and non-linear relation among power and performance measurements on I/O intensive applications impose limitations on the usage of linear regression techniques. We address these issues by developing computer system power models considering the following: (1) models that consider *P-states* as nominal parameters and the application that the computer is running, (2) models that make use of a machine-learning algorithm (CFS) for selecting the parameters most correlated to power, and (3) models that soften non-linear effects among computer system measurements and power by using k-means clustering.

5.1 Global power model

The first power model, called GPM, is built using linear regression having the miscellaneous component and the hard disk power modeled as a constant value equal to the average power of these power components. We use points from all *P-states* and from both benchmarks (i.e. SPECint2006 and SPECweb2009) to develop this model; hence, we are not distinguishing *P-states* nor applications. For the CPU power component, the performance measurements described in Sect. 3.2 are the dependent variables and the CPU power the independent variable.

Figure 4 shows the histogram and the cumulative distribution function (CDF) for this model on the Intel i7 machine. There are outliers which can reach up to 120 % of the absolute percent error (omitted in Fig. 4a). Even though, most of the points are concentrated within 50 % absolute error. The CDF (Fig. 4b) clarifies this observation. The median is about 15 % and the 90th percentile is about 20 % (dashed line) (i.e. 90 % percent of the points display an absolute percent error lesser than 20 %).

Rivoire et al. [29,30] claims that a power model must have the average for the absolute percent error lesser than 10 % to be consider accurate. Therefore, this model does not fit this requirement. The remaining of this Section discusses enhancements done on the power models to improve their accuracy and precision.

5.2 Nominal parameters and model specificity

In Fig. 3 we have shown that CPU power characteristics change when *P-states* change. Thus considering this parameter as nominal improves precision and accuracy. Hence, the first improvement that we do in GPM is to consider *P-state* as nominal parameter. In this approach, the CPU power is modeled by doing linear regression on the points of each P-state. This model is called P-state-based power model (pSPM). Differently from GPM, in pSPM, hard disk power and miscellaneous component power are modeled to a constant value equal to the average power on *each P-state*.

Figure 5 shows the cumulative distribution function for the GPM (solid line) and the pSPM (dashed line). The steeper the line, the more accurate the model. Thus, we observe that by doing a linear regression for each *P-state*, we improve the precision of the power models. Figure 5 also shows that

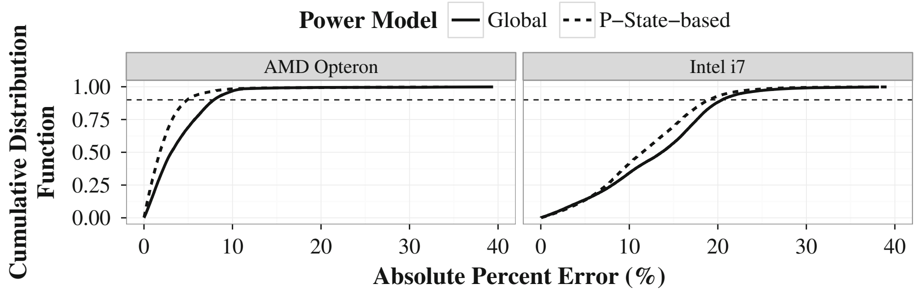


Fig. 5 Comparison between GPM and pSPM. The latter is more precise, since it has narrower steeper CDF curve

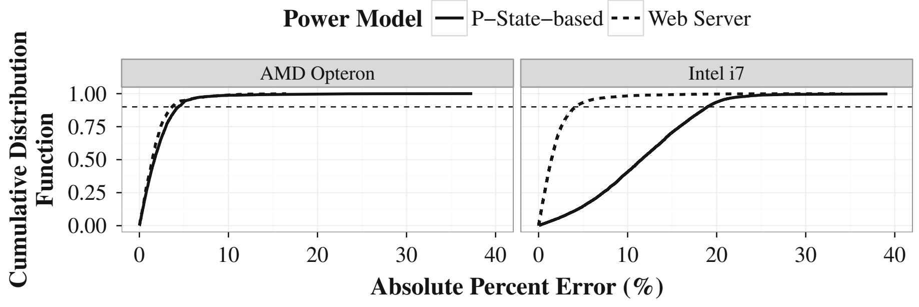


Fig. 6 CDF for the absolute percent error for pSPM and WSPM. The latter is more accurate having a steeper line

for the AMD Opteron server both GPM and pSPM meet the accuracy requirements (i.e. average for the absolute percent error below 10 %). However, they still do not meet the accuracy requirements for the Intel i7 server.

We have observed in our experiments that if a model is general (using points from either SPECint2006 or SPECweb2009), it needs more parameters to have similar accuracy and precision to a model developed for a specific application. To improve the pSPM, we consider the application that the machine is running.

On web server environment the workload is known upfront, the power variation is mostly due to variation on the number of concurrent requests. Thus, we relinquish model generality and focus on an specialized power model for web server application by considering just measurements done when running the SPECweb2009. This model is called web server power model (WSPM). In fact, we are targeting web server power models on our Data Center simulator; in this way, we take advantage of a more specific power model to also improve the simulator performance.

Figure 6 presents the CDF for the pSPM and the WSPM We observe that the general power model is inaccurate specially on the Intel i7 server. On the other hand, an specialized power model has improved accuracy reducing the 90th percentile of the absolute percent error on the AMD Opteron server from 4.4 to 3.5 % and on the Intel i7 server from 18.9 to 4.3 %. The WSPM meets the accuracy requirements, since it displays the average for the absolute percent error below 10 %. However, this model requires all the parameters that were presented in Sect. 3.2. The next section shows how we reduce the number of parameters.

5.3 Pruning model parameters

The WSPM meets the accuracy requirements. However, it requires nine parameters plus the processor *P-state*. During our experiments, we observe that many of the performance measurements are highly correlated with each other. Since we are using linear regression models, we can eliminate highly correlated parameters preserving the model precision and accuracy.

Figure 7 shows examples of parameters that are correlated in the Intel i7 server at the highest frequency state using points from SPECweb2009 and SPECint2006. The line represents the linear regression between the two axes. The stronger the correlation between two variables the closer the points follow the line. In this way, on the left hand side, we notice that CPU load correlates well to unhalted cycles per second; and on the right hand side, we observe that CPU migrations per second correlates to context switches per second.

Consider the WSPM, which uses all proposed parameters in Sect. 3.2. This model can be stated as follows: $P_{CPU} = a_0 + \sum_{i=1}^{8} a_i \cdot p_i$ where p_i's are the performance measurements and a_i's their associated coefficient values. Table 3 shows the coefficients calculated using linear regression for all *P-states* on both architectures.

This model displays high coefficient of determination ($R^2 = 0.943$), which means that most of the cases can be explained by the model, but needs all parameters described in Sect. 3.2 and shown in Table 3.

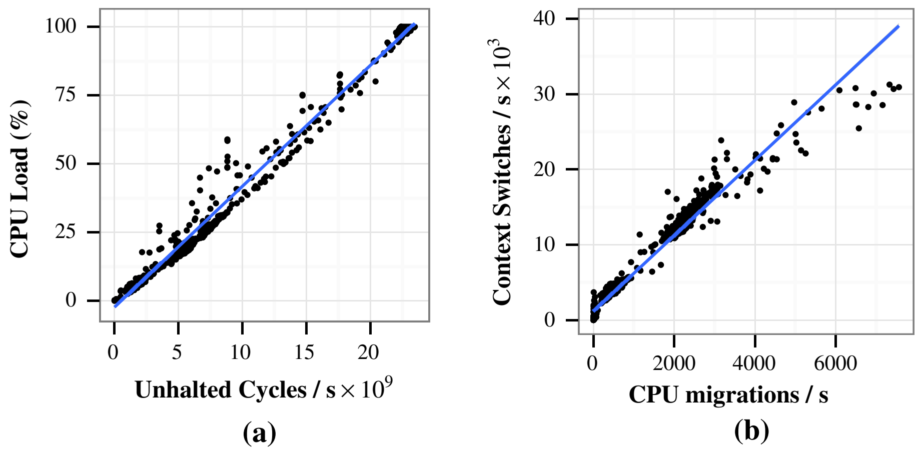


Fig. 7 Correlation between some of the model parameters. **a** CPU load versus unhalted cycles per second and **b** context switches per second versus CPU migration per second

Table 3 Coefficients for the web server CPU power model at each *P-state* for both architectures

Server	P-state	Cte	BIPS	Context swts.(s)	CPU load	Page faults (s)	CPU migs. (s)	Unhalted cycles (s)	LL cache refs (s)	LL cache misses (s)
	P0	20.77	18.83	6.69E−04	−1.2605	4.97E−04	−2.39E−03	−6.23E−09	2.00E−07	−1.89E−06
	P1	18.93	14.52	9.47E−04	−0.4822	2.58E−04	−1.09E−03	−7.28E−09	7.30E−09	1.46E−07
	P2	18.41	12.17	1.13E−03	−0.3459	2.64E−04	−1.74E−03	−6.03E−09	−1.26E−08	−8.27E−08
	P3	18.40	14.81	8.97E−04	−0.2476	2.42E−04	−2.30E−03	−1.03E−08	1.95E−07	7.75E−07
	P4	18.32	13.74	8.25E−04	−0.3675	1.98E−04	−2.54E−03	−8.46E−09	1.81E−07	5.73E−07
Intel	P5	18.53	12.88	9.55E−04	−0.2574	2.14E−04	−1.15E−03	−7.81E−09	2.30E−08	5.35E−07
i7	P6	17.91	11.21	9.52E−04	−0.2387	1.80E−04	−2.40E−03	−6.49E−09	8.23E−08	3.58E−07
	P7	18.06	12.44	5.90E−04	−0.3628	1.59E−04	−1.32E−03	−7.24E−09	1.50E−07	6.23E−07
	P8	17.88	12.03	6.93E−04	−0.2775	1.60E−04	−1.24E−03	−6.56E−09	6.54E−08	3.74E−07
	P9	18.03	10.74	7.14E−04	−0.2064	1.41E−04	−2.06E−03	−5.92E−09	6.28E−08	5.92E−07
	P10	18.19	11.51	6.09E−04	−0.1187	1.25E−04	−1.70E−03	−7.50E−09	9.94E−08	8.15E−07
	P11	18.28	9.59	6.34E−04	−0.1077	8.85E−05	−1.40E−03	−5.82E−09	2.35E−08	1.07E−06
	P12	18.08	10.43	4.45E−04	−0.0269	1.08E−04	−8.90E−04	−7.37E−09	9.62E−08	7.91E−07
	P13	17.35	9.76	3.44E−04	−0.0715	1.19E−04	−9.05E−04	−6.28E−09	1.09E−07	6.70E−07
	P0	46.95	−0.70	1.58E−03	0.4675	−1.03E−05	−3.94E−03	−4.68E−09	9.46E−09	2.32E−07
	P1	43.89	−12.46	2.27E−03	0.5750	1.95E−05	−2.34E−03	3.88E−09	2.73E−08	−7.30E−07
AMD	P2	42.83	−10.84	2.03E−03	0.4801	7.61E−06	−2.15E−03	3.55E−09	2.45E−08	−6.23E−07
Opteron	P3	40.48	4.70	1.35E−03	0.3792	−1.02E−04	−7.51E−04	2.39E−09	−1.54E−08	−3.58E−07
	P4	42.28	3.14	1.10E−03	0.3851	4.64E−05	−2.51E−04	−8.52E−10	4.75E−09	−4.93E−07

In this paper, we use a CFS [13] algorithm to facilitate choosing the parameters mostly correlated to power. The CFS algorithm focus on the job of feature selection for machine learning through a correlation based approach. The main assumption is that high quality feature sets (in our case all performance parameters) carry features that are highly correlated with the class (i.e. power). We believe that CFS is more appropriate for this task them other methods, such as Principal Component Analysis, because it returns a well determined subset of the input parameters (in contrast to other methods that rely the choice of them to experimenters). Therefore, it reduces the design space exploration and time analysing the models' accuracy. From our knowledge, this is the first work that uses this algorithm to support the choice of performance parameters to be used as proxies to power.

We develop the pruned web server power model (PWSPM) by applying CFS on points of each *P-state*, which prune parameters. Then, we apply linear regression using CPU power as the independent variable and the CFS selected parameters as the dependent variables. Table 4 shows the coefficients calculated at each *P-state* for the Intel i7 and AMD Opteron servers. The algorithm is able to reduce from nine parameters to up to three parameters at each *P-state*.

Figure 8 shows the CDF for the WSPM and for the PWSPM (in dashed lines). The average of the absolute percent error for WSPM is 2.08 % for the Intel i7 and 1.81 %

Table 4 Coefficients for the CPU pruned web server power model at each *P-state* for both architectures

Server	Pstate	Cte	BIPS	Context swts. (s)	CPU load	Page faults (s)	CPU Migs. (s)	Unhalted cycles (s)	LL Cache refs (s)	LL cache misses (s)
Intel i7	P0	24.69	1.98	1.17E−03						
	P1	18.92	8.93	1.05E−03	−1.24					
	P2	18.68	6.75	1.13E−03	−0.89					
	P3	18.47	6.85	1.05E−03	−0.86					
	P4	18.56	7.88	8.97E−04	−0.93					
	P5	18.13	7.17	1.00E−03	−0.84					
	P6	18.03	7.07	8.69E−04	−0.74					
	P7	17.99	7.33	7.45E−04	−0.69					
	P8	17.65	7.96	6.95E−04	−0.71					
	P9	19.49	2.63			9.15E−05				
	P10	18.81	1.88	4.62E−04						
	P11	18.38	1.91	4.56E−04						
	P12	18.27	1.90	4.46E−04						
	P13	17.71	1.98	4.28E−04						
	P0	47.83	−1.52	1.03E−03	0.40					
	P1	47.88	−3.38	7.01E−04	0.67					
AMD	P2	47.16			0.68					
Opteron	P3	42.62			0.59					
	P4	42.48			0.54					

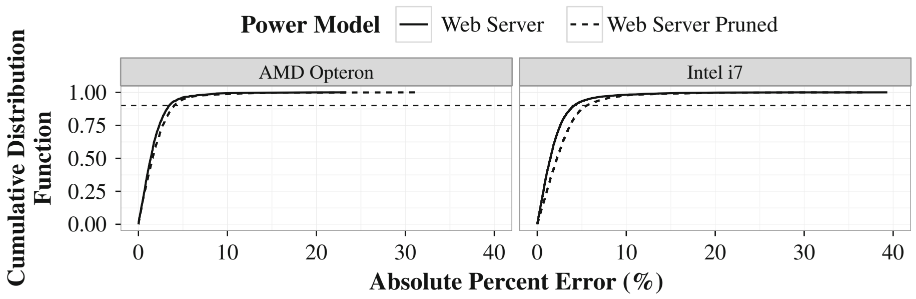


Fig. 8 CDF for the absolute percent error for WSPM and PWSPM

for the AMD Opteron. For the PWSPM, the average of the absolute percent error is 2.86 % for the Intel i7 and 2.07 % for the AMD Opteron. Therefore, PWSPM uses fewer parameters having similar accuracy and precision.

5.4 Softening non-linear effects

A web server needs to answer thousands of clients in a short period of time; hence, multiple processes are spawned creating an environment dominated by resource sharing. When using performance measurements as proxies for CPU power, non-linear relations are observed due to bottlenecks on shared resources, as noted by Rivoire [29].

Figures 9 and 10 show plots of CPU power versus BIPS and context switches, respectively, for Intel i7 sever at some of its *P-states* when running SPECweb2009 benchmark. The solid lines represent the linear model equation, while the dashed lines represent a log-log regression.

The non-linear effects are more remarkable at the higher performance states such as P0, P1, P2, and when running I/O bound workloads (i.e. SPECweb2009). At higher performance states, the frequency is higher and the CPU is relatively faster than the I/O devices; hence, it becomes idle more often waiting for I/O. Moreover, the CPU can handle more requests, increasing the competition for the shared resources, such as memory controllers and hard disk. Therefore, the CPU operates in bursts of processing creating non-linear effects among power and performance measurements.

Applying polynomial regression might be difficult when the polynomial order is unknown. Using logarithm regression results in models that do not take into account "idle power", because when no activity is observed in the computer the model yields a power value that is equal to zero. We use k-means clustering technique [24] to soften the non-linear effects. This approach approximates a non-linear curve using multiple linear segments creating a different equation with a

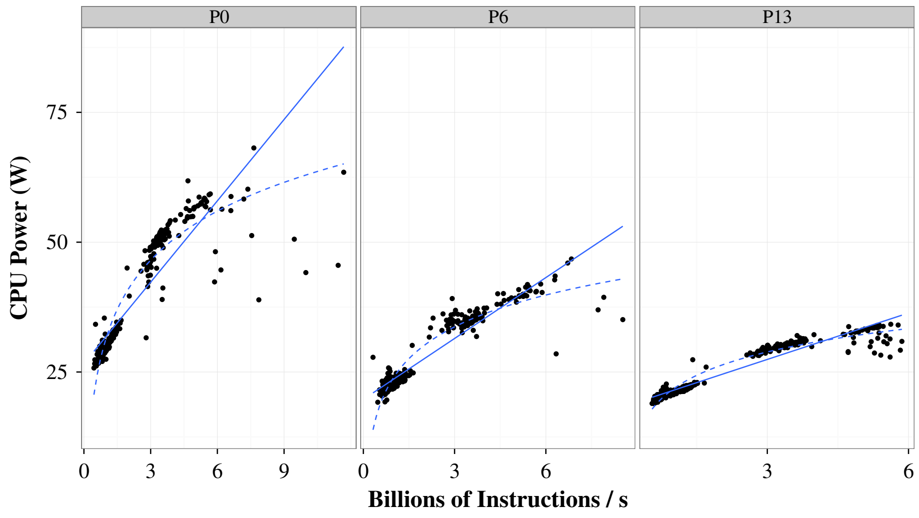


Fig. 9 i7 power versus BIPS. The *lines* represent the linear regression while the *dashed lines* represent a log regression

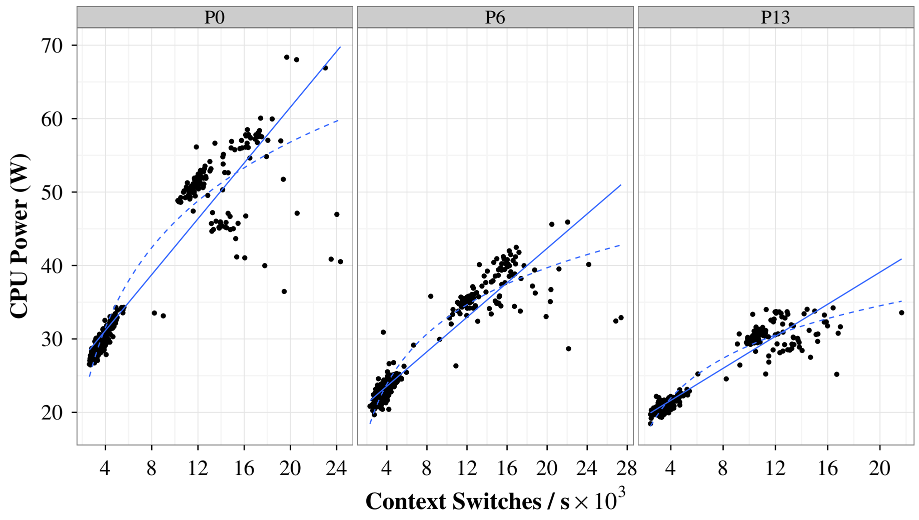


Fig. 10 i7 power versus Context Switches per second. The *lines* represent the linear regression while the *dashed lines* represent a log regression

different slope for each cluster. We use average values of HD power and miscellaneous components power for *each cluster* at each *P-state* to derive these two components power model.

K-means clustering is a cluster analysis method which targets to divide n observations into k clusters where each point lies into the cluster with the nearest mean. In a formal presentation, let $\boldsymbol{X} = \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ be a set of observations, where each observation is a d-dimensional real vector, k-means clustering partitions the n observations into k sets $\boldsymbol{S} = S_1, S_2, ..., S_n$ $(k \leq n)$, minimizing the within-cluster sum of squares, where $\boldsymbol{\mu}_i$ is the mean of points in S_i:

$$\arg\min_c \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in S_i} \parallel \boldsymbol{x}_j - \boldsymbol{\mu}_i \parallel^2 \qquad (2)$$

To obtain models with higher accuracy, we need to select the most pertinent set of variables that is used to create the clusters (based on CFS algorithm), the most suitable distance function, and the most appropriate number of clusters (i.e. k). This model is called cluster web server power model (CWSPM).

5.4.1 *Selecting the most pertinent variables*

When using cluster techniques, we observe that using a large number of attributes (i.e. all the gathered events) yields less precise models. We attribute this fact to the presence of high correlation among clustering variables which might overweight one or more parameters [19]. Hence, we apply the CFS algorithm to obtain a smaller set of variables mostly correlated to power.

5.4.2 *Choice of the most suitable distance function between the points*

We use the euclidean distance for the k-means clustering distance function in our models.

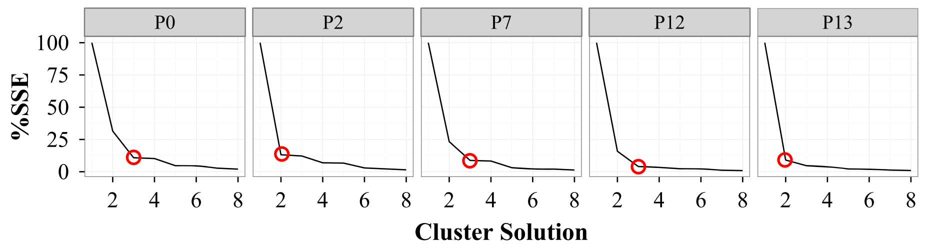


Fig. 11 Elbow plot for determining the best number of clusters for the Intel i7 server power model. The *circles* highlights the number of clusters chosen to partition the data set

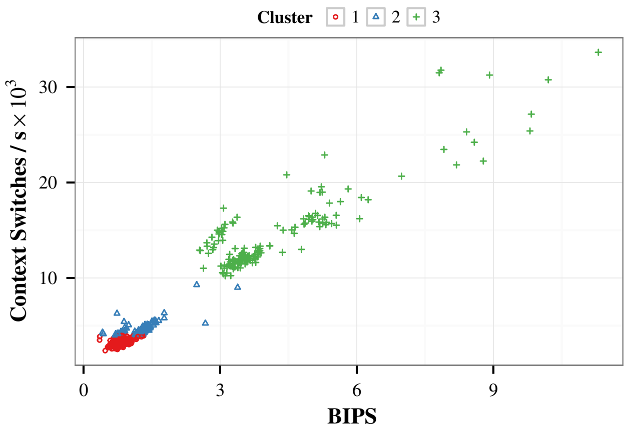


Fig. 12 Result of k-means clustering for the Intel i7 server at P-0 state after CFS selected BIPS and number number of context switches per second as the performance parameters to be used to estimate CPU Power

5.4.3 *Selecting the most suitable number of clusters*

To select an appropriate value of k, we use the sum of the squared errors of prediction (SSE) (also known as an F-test) as a function of the number of clusters and observe when adding another cluster does not yield better modeling of the data. This can be done visually by using "elbow criterion" where small values of k explain most of the variance. At some point the gain drops, resulting in an angle in the graph where the number of clusters is set.

Figure 11 shows the elbow plots for some *P-states* on the Intel i7 server where the circles highlight the number of clusters selected for each *P-state* (i.e. the location of the "elbow"). Note that the CFS algorithm may select different sets of variables for each *P-state* as shown in Table 4.

In order to explain the reasoning behind the usage of the k-means clustering, lets focus on the case of P0-state of the Intel i7 server. In this case, CFS returned BIPS and number of context switches per second as the performance statistics that should be used to estimate the CPU Power. The "elbow criterion" selects $k = 3$. K-means clustering on the 2-dimensional observations (BIPS and number of context switches per second) is applied. Figure 12 shows the result of the clusterization for this case.

Figure 12 shows that the regions are well defined and are making a discretization of the points based on the system activity. The non-linear effects start to be noticed at higher activity levels (higher BIPS and higher number of context switches per second), as noticed in Figs. 9 and 10. The k-means clustering algorithm is making a discretization of the points based on the system activity level, as shown in Fig. 12. Therefore, by applying a different linear-regression for each region, the non-linear effects are softened.

Finally, Fig. 13 shows a plot of CPU power versus BIPS for a subset of the data points for the Intel i7 server for all its *P-states* and we can see the effect of the discretization made by the k-means clustering.

Figure 14 shows the comparison among the three models. By using k-means clustering, we come up with a model that uses fewer parameters and is more accurate than a model that uses all of the others. Using fewer parameters is important for the simulator, since the input data could be reduced and is also important on real-time power estimation because fewer system measurements need to be sampled.

Finally, Fig. 15 shows an analysis for each *P-state* on each architecture. We can see that all models meet the accuracy requirement (i.e. they display average for the absolute percent below 10 %) at all frequency states. Furthermore, the CWSPM model uses fewer parameters than the WSPM and is the most accurate at all *P-states* but P9 on the Intel i7. Therefore, we select the CWSPM as the best model.

6 Conclusion

The stunning increase in the demand for Internet services in the last few decades is the primary reason for the need of increasingly larger data centers, which currently can host several thousands of computers inter-connected within a single facility. This scenario puts metrics such as power in more evidence not only for economical reasons but also for environmental issues. Therefore, reducing the power consumption must be a central role in the design of contemporary data centers.

This paper presented empirical models for estimating the power consumed by web servers. The models were validated with SPECweb2009, a state-of-the-art web benchmark which characterizes different web applications and contains both

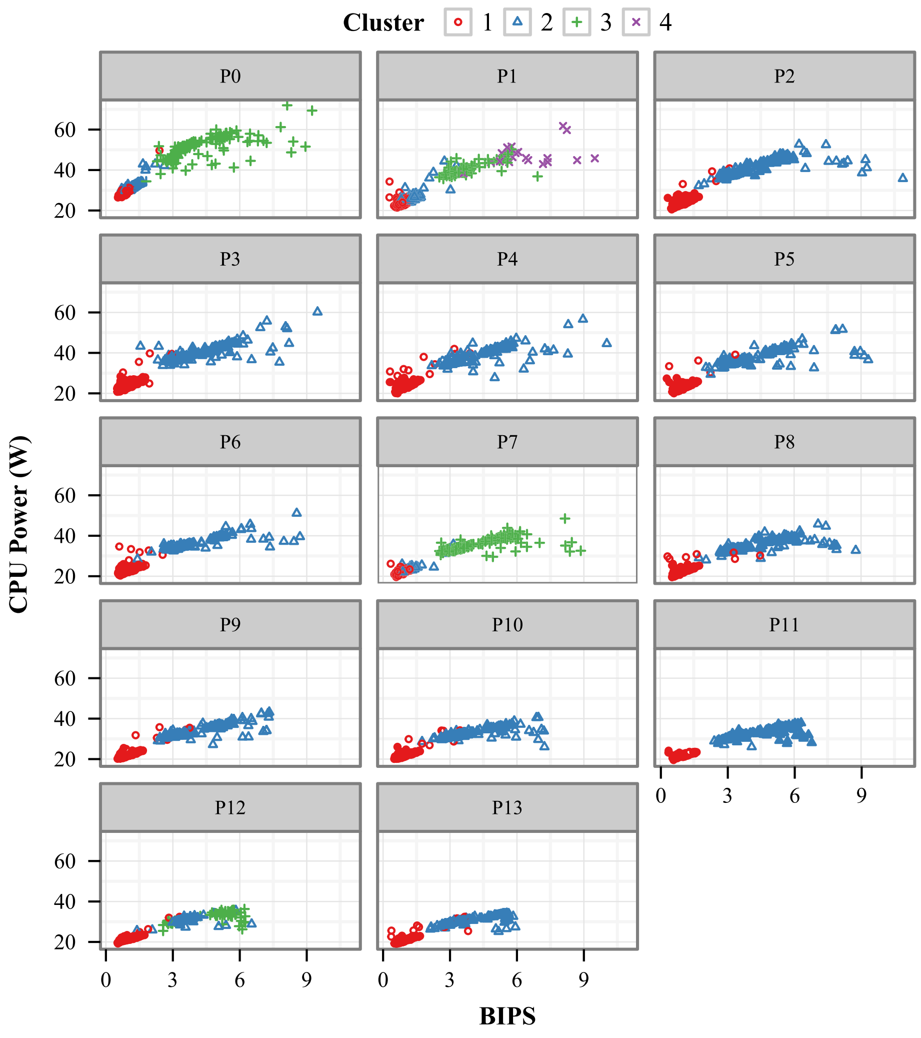


Fig. 13 CPU Power versus BIPS for a subset of the testing set in their respective cluster for the Intel i7 server. K-means cluster groups up the points and linear regression is applied on each cluster to soften non-linear effects among power and the parameters

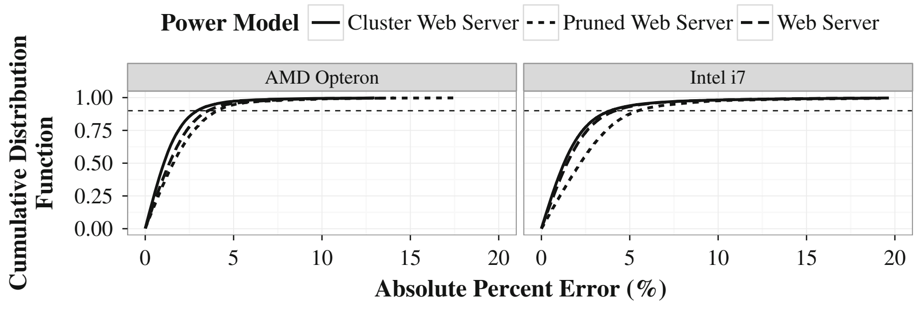


Fig. 14 CDF for the absolute percent error for WSPM, PWSPM, and CWSPM. The latter is the best model in terms of accuracy and also uses fewer parameters

static and dynamic content. We modeled two web servers having different processors and configurations.

The web server power measurements were done by a custom-made infrastructure that enables power breakdown for the individual system components. Hence, we found that the processor is the dominant component in the server's power consumption, corroborating the results presented by others [3]. This result suggests that the processor should be the main target when devising power-aware optimization algorithms.

We also presented a novel approach for modeling full system power based on CFS algorithm and k-means clustering. This new approach softened non-linear effects among system measurements and system power improving models accuracy. Our models considered the different frequency and voltage operating points (*P-states*) of the processor and took into account all major components of the server, such as processor, disk, network, memory, and other motherboard components Our best full-system power models displayed an average absolute error of 1.92 % for the Intel i7 server and

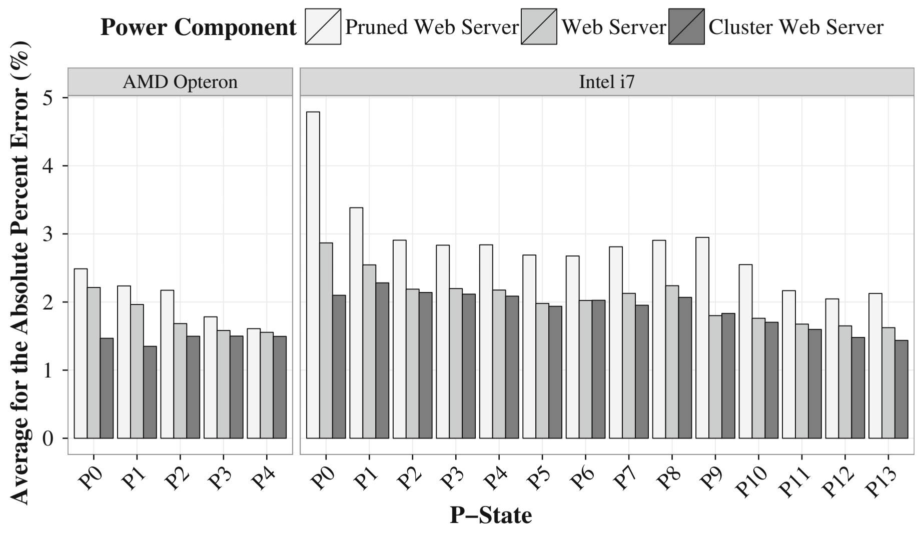


Fig. 15 Average of the the absolute percent error for the absolute percent error for WSPM, PWSPM, and CWSPM

1.46 % for the AMD Opteron as compared to actual measurements, and 90th percentile for the absolute percent error equal to 2.66 % for the Intel i7 and 2.08 % for the AMD Opteron.

Acknowledgments We would like to thank the reviewers for their valuable suggestions. This work was partially supported by FAPESP Grant No. 2010/05389-5.

References

1. Standard performance evaluation corporation (SPEC). http://www.spec.org/web2009 (2009). Accessed 17 March 2009
2. Advanced Configuration and Power Interface Specification. http://www.acpi.info/spec.htm (2011). Accessed 29 November 2011
3. Barroso, L.A., Holzle, U.: The case for energy-proportional computing. IEEE Computer (2007)
4. Bellosa, F.: The benefits of event-driven energy accounting in power-sensitive Systems. In: EW 9: Proceedings of the 9th workshop on ACM SIGOPS European, workshop (2000)
5. Bergamaschi, R.A., Piga, L., Rigo, S., Azevedo, R., Araujo, G.: Data center power and performance optimization through global selection of p-states and utilization rates. Sustain Comput Inf Syst **2**(4), 198–208 (2012)
6. Bertran, R., Gonzalez, M., Martorell, X., Navarro, N., Ayguade, E.: Decomposable and responsive power models for multicore processors using performance counters. In ICS '10: Proceedings of the 24th ACM International Conference on Supercomputing (2010)
7. Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., Rajamony, R.: Power aware computing. The case for power management in web servers (2002)
8. Carrera, E. V., Pinheiro, E., Bianchini, R.: Conserving disk energy in network servers. In ICS '03: Proceedings of the 17th annual international conference on Supercomputing (2003)
9. Chen, X., Xu, C., Dick, R.P., Mao, Z.M.: Performance and power modeling in a multi-programmed multi-core environment. In: Proceedings of the 47th Design Automation Conference (2010), DAC '10.
10. Cochran, R., Hankendi, C., Coskun, A., Reda, S.: Pack & cap: adaptive dvfs and thread packing under power caps. In: 44th Annual IEEE/ACM International Symposium on Microarchitecture (2011)
11. Contreras, G., Martonosi, M.: Power prediction for Intel XScaleprocessors using performance monitoring unit events. In: ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design (2005)
12. Fan, X., Weber, W.-D., Barroso, L. A.: Power provisioning for a warehouse-sized computer. In: ISCA '07: Proceedings of the 34th, annual international symposium on Computer architecture (2007)
13. Hall, M. A.: Correlation-based feature selection for machine learning. Ph.D. Thesis, University of Waikato (1999)
14. Instruments, N.: Bus-Powered M Series Multifunction DAQ for USB - 16-Bit, up to 400 kS/s, up to 32 Analog Inputs, Isolation Data Sheet (2009)
15. Intel.: Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2. Santa Clara, CA, USA (2013)
16. Isci, C., Buyuktosunoglu, A., Cher, C., Bose, P., Martonosi, M.: An analysis of efficient multicore global power management policies: Maximizing performance for a given power budget. In: 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-39 2006) (2006)
17. Isci, C., Martonosi, M.: Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In: MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture (2003)
18. Joseph, R., Martonosi, M.: Run-time power estimation in high performance microprocessors. In: ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design (2001)
19. Ketchen, D.J., Shook, C.L.: The application of cluster analysis in strategic management research: an analysis and critique. Strateg Manag J **17**(6), 441–458 (1996)
20. Laros, J., Pedretti, K., Kelly, S., Vandyke, J., Ferreira, K., Vaughan, C., Swan, M.: Topics on measuring real power usage on high performance computing platforms. In: CLUSTER '09. IEEE International Conference on Cluster Computing and Workshops (2009)
21. LEM Components. Current transducer lts 25-NP data sheet (2008)
22. Lewis, A.W., Tzeng, N.-F., Ghosh, S.: Runtime energy consumption estimation for server workloads based on chaotic time-series approximation. ACM Trans. Archit. Code Optim. **9**, 3 (Oct. 2012)

23. Linux Kernel Organization. Block layer statistics: Linux Documentation Project (2010)
24. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans Inf Theor **28**, 129–137 (1982)
25. Lucer, C.D., Akella, C.: Power profiling for embedded applications. White paper (2009)
26. Piga, L., Bergamaschi, R., Azevedo, R., Rigo, S.: Power measuring infrastructure for computing systems. Institute of Computing, University of Campinas, Tech. rep. (2011)
27. Rajamani, K., Rawson, F., Ware, M., Hanson, H., Carter, J., Rosedahl, T., Geissler, A., Silva, G., Hua, H.: Power-performance management on an IBM POWER7 server. In: ISLPED '10: Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design (2010)
28. Red Hat Inc., Performance counters for linux (2010)
29. Rivoire, S., Ranganathan, P., Kozyrakis, C.A.: comparison of high-level full-system power models. In: HotPower'08 (2008)
30. Rivoire, S.M.: Models and metrics for energy-efficient computer systems. Ph.D. Thesis, Department of Electrical Engineering of Stanford University (2008)
31. Rotem, E., Naveh, A., Rajwan, D., Ananthakrishnan, A., Weissmann, E.: Power management architecture of the 2nd generation intel core microarchitecture, formerly codenamed sandy bridge. In: Hot Chips 23 (2011)
32. Zedlewski, J., Sobti, S., Garg, N., Zheng, F., Krishnamurthy, A., Wang, R.: Modeling hard-disk power consumption. In: FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies (2003)

Leonardo Piga received his Ph.D. in Computer Science at University of Campinas (Unicamp) in 2014 and a Bachelors degree in Computer Engineering from the University of Campinas in 2008. Leonardo was also an intern at AMD Research Labs working with characterization of Cloud Workload. In his Ph.D., he developed researches in power and performance optmizations for Web server. Leonardo is current working at AMD Research labs in Austin, TX, USA.

Reinaldo A. Bergamaschi received his Ph.D. in EECS from the University of Southampton, UK in 1989, a Masters degree from the Philips International Institute, Eindhoven, The Netherlands, and a Bachelors degree in Electronics from ITA, Brazil. He worked for 19 years at the IBM T. J. Watson Research Center, Yorktown Heights, NY, in the areas of CAD, power modeling and SoC design. He was a visiting professor at UNICAMP, Brazil in 2008/2009. He is now Founder and CEO of Odysci, Inc., developing knowledge management systems. He is a Fellow of the IEEE, and an ACM Distinguished Scientist.

Sandro Rigo collected his Ph.D. at University of Campinas (Unicamp) in 2004. He joined the staff at the Institute of Computing, Unicamp, in 2005. From 2009-2010, he served as the Coordinator for the Bachelor of Computer Science program at IC-Unicamp. During his Ph.D., Prof. Rigo worked in the creation of the ArchC architecture description language, earning the Best Paper award at SBAC-PAD 2004. His current research interests are in code generation and optimization, dynamic compilation, power/performance management computing resources, and virtual platform simulation for SoCs. Prof. Rigo has been serving as a PC member for the Brazilian Symposium on Programming Languages and several other workshops.