Server Power Modeling for Run-time Energy Optimization of Cloud Computing Facilities

Patricia Arroba , José L. Risco-Martín , Marina Zapater , José M. Moya
José L. Ayala , Katzalin Olcoz

Abstract

As advanced Cloud services are becoming mainstream, the contribution of data centers in the overall power consumption of modern cities is growing dramatically. The average consumption of a single data center is equivalent to the energy consumption of 25.000 households. Modeling the power consumption for these infrastructures is crucial to anticipate the effects of aggressive optimization policies, but accurate and fast power modeling is a complex challenge for high-end servers not yet satisfied by analytical approaches. This work proposes an automatic method, based on Multi-Objective Particle Swarm Optimization, for the identification of power models of enterprise servers in Cloud data centers. Our approach, as opposed to previous procedures, does not only consider the workload consolidation for deriving the power model, but also incorporates other non traditional factors like the static power consumption and its dependence with temperature. Our experimental results shows that we reach slightly better models than classical approaches, but simultaneously simplifying the power model structure and thus the numbers of sensors needed, which is very promising for a short-term energy prediction. This work, validated with real Cloud applications, broadens the possibilities to derive efficient energy saving techniques for Cloud facilities.

Keywords:
Green Data Centers, Sustainable Cloud Computing, Power modeling, Particle Swarm Optimization

1. Introduction

One of the big challenges in data centers is to manage system resources in a power-efficient way. Data centers consume from 10 to 100 times more power per square foot than typical office buildings [1] even consuming as much electricity as a city [2]. Consequently, these infrastructures need to be managed in a power-efficient manner to drive Green Cloud computing [3].

Besides economic incentives, the Cloud model provides also benefits from the environmental perspective, since the computing resources are managed by Cloud service providers but shared among all users, which increases their overall utilization [4]. This fact is translated into a reduced carbon footprint per executed task, diminishing CO_2 emissions. The Schneider Electric's report on virtualization and Cloud computing efficiency [5] confirms that about 17% of annual savings in energy consumption were achieved by 2011 through virtualization technologies.

However, data center designers have collided with the lack of accurate power models for the energy-efficient provisioning and the real-time management of the computing facilities. These power models facilitate the analysis of several architectures from the perspective of the power consumption, and they allow us to devise efficient techniques for energy optimization.

The work proposed in this paper makes substantial contributions in the area of power modeling of Cloud servers taking into account these factors. We envision a powerful method for the automatic identification of fast and accurate power models that target high-end Cloud server architectures. Our methodology considers the main sources of power consumption as well as the architecture-dependent parameters that drive today's most relevant optimization policies.

Analytical models, as closed form solution representations, require the classification of the parameters that regulate the performance and power consumption of a computer system. Also, it is mandatory to find the complex relationships between these parameters to build the analytical functions [6]. Thus, describing complex systems using analytical models is a hard and time-consuming task because it requires knowledge of the dynamics of the specific problem. Therefore, the modeling of scalable, distributed and highly heterogeneous systems is unfeasible for analytical methods. On the other hand, metaheuristics are higher-level procedures that make few assumptions about the optimization problem, providing adequately good solutions that could be based on fragmentary information [7, 8]. They are particularly useful in solving optimization problems that are noisy, irregular and change over time. In this way, metaheuristics appear as a suitable approach to meet our optimization problem requirements.

In the last years, there has been a rising interest in developing simple techniques that provide basic power management for servers operating in a Cloud, i.e. turning on and off servers, putting them to sleep or using Dynamic Voltage and Frequency Scaling (DVFS) to adjust servers' power states by reducing clock frequency. Many of these recent research works have focused on reducing power consumption in cluster systems [9, 10, 11, 12]. In general, these techniques take advantage of the fact that application performance can be adjusted to utilize idle time on the processor to save energy [13]. However, their application in Cloud servers is difficult to achieve in practice as the service provider usually over-provisions its power capacity to address worst case scenarios. This often results in either waste of power or severe under-utilization of resources. Thus, it is critical to quantitatively understand the relationship between power consumption, temperature and load at the system level by the development of a power model that helps on optimizing the use of the deployed Cloud services. Finally our work makes the following contributions:

- We propose an accurate power model for high-end servers in Cloud facilities. This model, as opposed to previous approaches, does not only consider the workload assigned to the processing element, but also incorporates the need of considering the static power consumption and, even more interestingly, its dependence with temperature.

- Moreover, this power model, applied to both the processing core and the memories of the system, includes voltage and frequency as parameters to be tuned during run-time by the DVFS policies.

- The model has been built and tested for an enterprise server architecture and with several real applications that can be commonly found in nowadays' Cloud server machines, achieving low error when compared to real measurements.

- We have developed the power model for our target architecture using the Optimal Multi-Objective Particle Swarm Optimization (OMOPSO), a novel technique to perform the curve fitting. This algorithm allows the simplification of our model attending to each server architecture.

The remainder of this paper is organized as follows: Section 2 gives further information on the related work on this topic. The power model is presented in Section 3. Section 4 provides the background algorithm used for the model optimization. In Section 5 we describe the algorithm setup to adapt its parameters to our optimization problem. Section 6 describes profusely the experimental results. Finally, in Section 7 the main conclusions are drawn.

2. Related Work

Currently the state of the art offers various power models. However these models are analytical, architecture-dependent and do not include the contribution of static power consumption, or the capability of switching the frequency modes (DVFS). The authors develop linear regression models that present the power consumption of a server as a linear function of the CPU usage of that server [14, 15, 16].

Some other models can be found where server power is formulated as a quadratic function of the CPU usage [17, 18, 19]. Still, as opposed to ours, these models do not include the estimation of the static power consumption (which has turned to have a great impact due to the current server technology). Besides, these models have not been exploited in a multi-objective optimization methodology to minimize the power consumption of servers for Cloud services.

Bohra et al. [20] propose a robust fitting to calculate their model that takes into account the correlation between the total system power consumption and component utilization. Our work follows a similar approach but also incorporates the contribution of the static power consumption, its dependence with temperature, and the effect of applying voltage and frequency scaling techniques.

Interestingly, one key aspect in the management of a data center is still not very well understood: controlling the ambient temperature at which the data center operates. Data centers operate in a broad temperature range from 18°C to 24°C but some can be as cold as 13°C [21, 22]. However, due to the lack of accurate power models, the effect of ambient temperature on the power consumption of the servers has not been clearly analyzed, preventing the application of optimization models to save energy. On the contrary, the experimental work presented in this paper has been performed in ambient temperatures ranging from 18°C to 25°C. The range selected follows nowadays' practice of operating at higher temperatures [23] and close to the limits recommended by ASHRAE. Although this practice obtains energy savings in the cooling expense [24], the lack of a detailed power model prevents to apply optimization policies.

The work presented in this paper outperforms previous approaches in the area of power modeling for enterprise servers in Cloud facilities in several aspects. Our approach consists on an automatic method for the identification of an accurate power model particularized for each target architecture. We propose an extensive power model consistent with current architectures. It is based on a generic analytical model where the main power consumption sources are considered. The model is multiparametric to allow the development of power optimization approaches. Our generic power model is optimized using metaheuristics, resulting in a specific model instance for every target architecture. Also the execution of the resulting power model is fast, making it suitable for run-time optimization techniques.

3. Power modeling

Dynamic consumption has historically dominated the power budget in electronic systems. But when the integration technology scales below the 100*nm* boundary, static power consumption becomes much more significant and reaches 30-50% [25] of the total power under nominal conditions. This issue is intensified by the influence of temperature on the leakage current behavior and its exponential dependency.

Leakage current increases strongly with temperature [26], also in deep sub-micron technologies [27], consequently increasing power consumption. Therefore, it is important to consider the strong impact of static power consumed by devices as well as its dependence with temperature, and the additional effects influencing their performance. In this section, we derive a leakage model for the static consumption of servers attending to these concepts. The model is validated with real measurements taken in the enterprise server of our case study.

The current that is generated in a MOS device due to leakage is given by

$$I_{\text{leak}} = I_{\text{s}} \cdot e^{\frac{V_{\text{GS}}-V_{\text{TH}}}{nkT/q}} \cdot (1 - e^{\frac{-V_{\text{DS}}}{kT/q}}) \tag{1}$$

Research by Rabaey [26] shows that if $V_{\text{DS}} > 100mV$ the contribution of the second exponential in (1) is negligible, so the previous formula can be rewritten as

$$I_{\text{leak}} = I_{\text{s}} \cdot e^{\frac{V_{\text{GS}}-V_{\text{TH}}}{nkT/q}} \tag{2}$$

where leakage current depends on the slope factor n, the surface mobility of the carriers μ, the capacitance of the insulating material for the oxide gate C_{ox} and the ratio between the width and length of the transistors $\frac{W}{L}$ as can be seen in the following equation. Technology-dependent parameters can be grouped together to obtain an α constant.

$$I_{\text{s}} = 2 \cdot n \cdot \mu \cdot C_{\text{ox}} \cdot \frac{W}{L} \cdot \frac{kT}{q}^{2} \qquad I_{\text{leak}} = \alpha \cdot T^{2} \cdot e^{\frac{V_{\text{GS}}-V_{\text{TH}}}{nkT/q}} \tag{3}$$

Using (3) in the leakage power equation $P_{\text{leak}} = I_{\text{leak}} \cdot V_{\text{DD}}$, the leakage power for a particular machine m can be derived:

$$P_{\text{leak}}(m) \quad = \quad \alpha(m) \cdot T^{2}(m) \cdot e^{\frac{V_{\text{GS}}-V_{\text{TH}}}{nkT/q}} \cdot V_{\text{DD}}(m) \tag{4}$$

Since our goal is to fit a model for the leakage power, we expand the polynomial function (4) into its Taylor third order series in order to easily regress the function, which leads to

$$P_{\text{leak}}(m) \quad = \quad \alpha_1(m) \cdot T^{2}(m) \cdot V_{\text{DD}}(m) + \alpha_2(m) \cdot T(m) \cdot V_{\text{DD}}^{2}(m) + \alpha_3(m) \cdot V_{\text{DD}}^{3}(m) \tag{5}$$

where $\alpha_1(m)$, $\alpha_2(m)$ and $\alpha_3(m)$ define the specific constants due to the manufacturing parameters of a server.

Incorporating frequency and voltage dependence in models is interesting due to the current trend of using DVFS modes to control the power consumption of servers.

Two of the main contributors to power consumption in servers are the CPU and the memory subsystem. We can easily find DVFS modes in CPUs, but there are currently very few memory devices with these capabilities. Power consumption of both disk and network have not been taken into account because of their lower impact in our scenario, high variability and heterogeneity of their technology in data centers.

Below is the formulation of the static consumption in a scenario with a CPU providing $k \in \{1 \ldots K\}$ different DVFS modes and a memory performing at a constant voltage. The model considers the different contributions due to temperature dependency. Also $\gamma(m)$ has been taken into account as it represents the fan power contribution constant. As seen in [28] fan power is a cubic function of fan speed represented as $FS(m)$. $\lambda(m)$ represents the total consumption of the rest of the server resources and devices that operate at a constant voltage and frequency.

$$\begin{aligned} P_{\text{leak}}(m,k) \quad &= \quad \alpha_1(m) \cdot T_{\text{cpu}}^{2}(m) \cdot V_{\text{DD}}(m,k) + \alpha_2(m) \cdot T_{\text{cpu}}(m) \cdot V_{\text{DD}}^{2}(m,k) + \alpha_3(m) \cdot V_{\text{DD}}^{3}(m,k) \\ &+ \quad \beta_1(m) \cdot T_{\text{mem}}(m) + \beta_2(m) \cdot T_{\text{mem}}^{2}(m) + \gamma(m) \cdot FS^{3}(m) + \lambda(m) \end{aligned} \tag{6}$$

As temperature-dependent leakage cannot be measured separately from the dynamic power in a server, we execute the *lookbusy*[1] synthetic workload to stress the system during monitored periods of time. *Lookbusy* can stress all the hardware threads to a fixed CPU utilization percentage without memory or disk usage. The use of a synthetic workload to derive the leakage model has many advantages, the most important of which is that dynamic power can be described as linearly dependent with CPU utilization and Instructions Per Cycle (IPC). Equation 7 provides the formula for dynamic power consumption.

$$P_{\text{cpu}}^{\text{dyn}}(m,k) = \alpha_4(m) \cdot V_{\text{DD}}^{2}(m,k) \cdot f(m,k) \cdot u_{\text{cpu}}(m,k) \tag{7}$$

In the previous formula $\alpha_4(m)$ is a constant that defines the technological parameters of the machine m, $V_{\text{DD}}(m,k)$ is the CPU supply voltage and $f(m,k)$ is the working frequency of the machine in a specific k DVFS mode. $u_{\text{cpu}}(m,k)$ is the averaged CPU percentage utilization of the specific physical machine m that operates in the k DVFS mode. $u_{\text{cpu}}(m,k)$ is proportional to the number of cycles available in the CPU and accurately describes power consumption.

In order to stress the memory system we have developed a specific benchmark based on *RandMem*[2]. The program accesses random memory regions of an explicit size to explore the memory power consumption.

[1]http://www.devin.com/lookbusy/
[2]http://www.roylongbottom.org.uk

Dynamic power consumption depends on the high level data cache misses characterized during profiling. As memory performs at a constant frequency and voltage, equation 8 describes its dynamic power consumption.

$$P_{\text{mem}}^{\text{dyn}}(m,k) = \beta_3(m) \cdot u_{\text{mem}}(m,k) \tag{8}$$

The constant $\beta_3(m)$ is defined by the technological features of the device encompassing both the constant frequency and voltage and $u_{\text{mem}}(m,k)$ represents the memory utilization expressed in memory accesses per cycle in a k DVFS mode ($k = 1$ represents a powered down server).

Finally, total power can be described as in equation 9.

$$\begin{aligned}
P_{\text{tot}}(m,k) &= P_{\text{cpu}}(m,k) + P_{\text{mem}}(m,k) + P_{\text{others}}(m,k) && (9)\\
P_{\text{cpu}}(m,k) &= \alpha_1(m) \cdot T_{\text{cpu}}^2(m) \cdot V_{\text{DD}}(m,k) + \alpha_2(m) \cdot T_{\text{cpu}}(m) \cdot V_{\text{DD}}^2(m,k) + \alpha_3(m) \cdot V_{\text{DD}}^3(m,k) &&\\
&+ \alpha_4(m) \cdot V_{\text{DD}}^2(m,k) \cdot f(m,k) \cdot u_{\text{cpu}}(m,k) && (10)\\
P_{\text{mem}}(m,k) &= \beta_1(m) \cdot T_{\text{mem}}(m) + \beta_2(m) \cdot T_{\text{mem}}^2(m) + \beta_3(m) \cdot u_{\text{mem}}(m,k) && (11)\\
P_{\text{others}}(m,k) &= \gamma(m) \cdot FS^3(m) + \lambda(m) && (12)
\end{aligned}$$

4. Model identification

As stated above, our proposed power model consists of 9 parameters. Depending on the target architecture, some parameters might have more impact than others, as shown in our results. In our case, identification is performed as a multi-objective optimization and compared with a classical regression method. With a multi-objective optimization, we simultaneously optimize average and maximum errors to avoid peaks in the error function. To this end, we have selected a multi-objective Particle Swarm Optimization (PSO) algorithm to identify our power model. The reason for selecting multi-objective PSO is that this stochastic evolutionary computation technique, based on the movement and intelligence of swarms, has obtained excellent results specially in instances with real variables [29]. Next we provide a brief background about multi-objective optimization and the algorithm selected.

4.1. Multi-objective optimization

Multi-objective optimization tries to simultaneously optimize several contradictory objectives. For this kind of problems, single optimal solution does not exist, and some trade-offs need to be considered. Without any loss of generality, we can assume the following multi-objective minimization problem:

$$\begin{aligned}
&\text{Minimize} && \vec{z} = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}))\\
&\text{Subject to} && \vec{x} \in X
\end{aligned} \tag{13}$$

where $\vec{z}$ is the objective vector with m objectives to be minimized, $\vec{x}$ is the decision vector, and X is the feasible region in the decision space. A solution $\vec{x} \in X$ is said to dominate another solution $\vec{y} \in X$ (denoted as $x \prec y$) if and only if the following two conditions are satisfied:

$$\forall i \in \{1, 2, \ldots, m\}, f_i(\vec{x}) \leq f_i(\vec{y}) \tag{14}$$

$$\exists i \in \{1, 2, \ldots, m\}, f_i(\vec{x}) < f_i(\vec{y}) \tag{15}$$

A decision vector $\vec{x} \in X$ is non-dominated with respect to $S \subseteq X$ if another $\vec{x}' \in S$ such that $\vec{x}' \prec \vec{x}$ does not exist. A solution $\vec{x}^* \in X$ is called Pareto-optimal if it is non-dominated with respect to X. An objective vector is called Pareto-optimal if the corresponding decision vector is Pareto-optimal.

The non-dominated set of the entire feasible search space X is the *Pareto-Optimal Set (POS)*. The image of the POS in the objective space is the *Pareto-Optimal Front (POF)* of the multi-objective problem at hand. A multi-objective optimization problem is solved, when its complete POS is found.

4.2. *PSO and OMOPSO*

Particle Swarm Optimization (PSO) is a heuristic search technique that simulates the movements of a flock of birds that aim to find food. The relative simplicity of PSO and the fact that is a population-based technique have made it a natural candidate to be extended for multi-objective optimization [30].

In PSO, particles are "flown" throughout a hyper-dimensional search space. Changes to the position of particles within the search space are based on social-psychological tendencies of individuals to emulate the success of other individuals. Hence, the position of each particle is changed according to its own experience and its neighbors. Let $x_i(t)$ denote the position of particle p_i, at time step t. The current position of p_i is then changed by adding a velocity vector $v_i(t)$ to the previous position, i.e.:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{16}$$

The velocity vector reflects the socially exchanged information and is defined in the following way:

$$\vec{v}_i(t) \quad = \quad W\vec{v}_i(t-1) + C_1\vec{r}_{i1}(\vec{x}_{ipbest} - \vec{x}_i(t-1)) + C_2\vec{r}_{i2}(\vec{x}_{ileader} - \vec{x}_i(t-1)) \tag{17}$$

where:

- W is the inertia weight and controls the impact of the previous history of velocities.
- C_1 and C_2 are the learning factors. C_1 is the cognitive learning factor and represents the attraction that a particle has towards its own success. C_2 is the social learning factor and represents the attraction that a particle has towards the success of its neighbors.
- $\vec{r}_{i1}$, $\vec{r}_{i2}$ are random vectors, each component in the range $[0, 1]$.
- $\vec{x}_{ipbest}$ is the personal best position of p_i , namely, the position of the particle that has provided the greatest success.
- $\vec{x}_{ileader}$ is the position of the particle that is used to guide p_i towards better regions of the search space.

Particles tend to be influenced by the success of any other element they are connected to. These neighbors are not necessary particles close to each other in the decision variable space, but instead are particles that are close to each other based on a neighborhood topology, which defines the social structure of the swarm [30].

M. Reyes y C. Coello proposed a multi-objective PSO approach based on Pareto dominance, named OMOPSO [29]. This algorithm uses a crowding factor for the selection of leaders. This selection is made by binary tournament. This proposal uses two external archives: one for storing the leaders currently being used for performing the flight and another one for storing the final solutions. Only the leaders with the best crowding values are retained. Additionally, the authors propose a scheme in which the population is subdivided in three different subsets. A different mutation operator is applied to each subset. We use OMOPSO in the identification of our proposed power model, identifying the set of parameters that are representative for each target architecture.

5. Algorithm setup

PSO, as a metaheuristic, makes few assumptions about the optimization problem. As a consequence, the algorithm requires a preliminar configuration to provide adequate solutions. In this section we explain both the constraints and the parameter setup to adapt the metaheuristic to our optimization problem.

5.1. *Multi-objective function*

The problem to be solved is the estimation of the power consumption in virtualized enterprise servers performing Cloud applications. Our power model considers the heterogeneity of servers, as the specific technological features of each processor architecture result in a different power consumption. The resultant power model is non-linear (as shown in the previous section) and presents a large set of constraints. As stated above, the model identification is tackled as a multi-objective optimization simultaneously minimizing both the average and maximum errors :

$$\begin{array}{rrcl} \text{Minimize} & \vec{z} & = & (e_{\text{avg}}(\vec{x}), e_{\max}(\vec{x})) \\ \text{Subject to} & \vec{x}_{\min} & \leq & \vec{x} < \vec{x}_{\max} \\ \text{where} & \vec{x} & = & (\alpha_1, \ldots, \alpha_4, \beta_1, \ldots, \beta_3, \gamma, \lambda) \in X \end{array} \tag{18}$$

$\vec{x}$ is the vector of n decision variables and $\vec{z}$ is the vector of 2 objectives function. $e_{\text{avg}}(\vec{x})$ is the average relative error percentage, $e_{\max}(\vec{x})$ is the maximum of the relative error percentage (equation 19) and X is the feasible region in the decision space. Although we are interested in the minimization of the average relative error, we also use the maximum error percentage to avoid singular high peaks in the estimated model.

$$e_{\text{avg}}(\vec{x}) = \frac{1}{N} \cdot \sum_{n} \frac{(P - P_{\text{tot}}) \cdot 100}{P} \qquad e_{\max}(\vec{x}) = \max \frac{(P - P_{\text{tot}}) \cdot 100}{P} \tag{19}$$

P is the power consumption measure given by the current clamp, P_{tot} is the power consumption estimated by our model (equation 9) and n is each sample of the entire set of N samples used for the algorithm training. We use OMOPSO [31] to obtain a set of candidate solutions in order to solve our problem. Using this formulation, we are able to obtain a power consumption that is realistic with the current technology.

5.2. *Algorithm parameters*

Our power modeling problem requires a set of solutions with low error when compared to the real power consumption measures. In order to obtain suitable solutions we tune the OMOPSO algorithm using the following parameters:

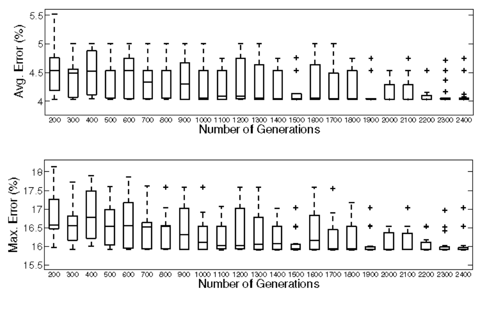
- Swarm size: 100 particles.
- Number of generations: 2000. We avoid the PSO algorithm to be trapped in a local minimum by exhaustively analyzing this parameter. We have performed 20 optimizations for each number of generations ranging from 200 to 2400 as can be seen in Figure 1(a).
- Perturbation: Uniform and non-uniform mutation. Both with a perturbation index of 0.5 and with mutation probability inversely proportional to the number of decision variables, 1/9 in our case.
- W, C1 and C2 are generated for each particle in every iteration as a random value in [0.1,0.5], [1.5,2] and [1.5,2], respectively.

6. Experimental results

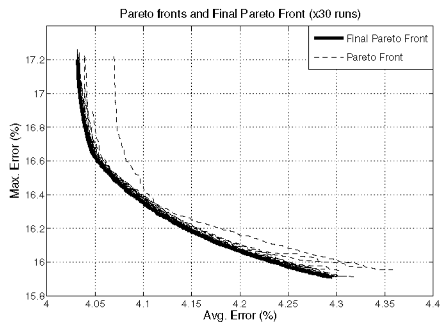
6.1. *Training*

Tests have been conducted gathering real data from an enterprise server. The Fujitsu RX300 S6 server is based on an Intel Xeon E5620 processor operating at f_{m2} = 1.73 GHz, f_{m3} = 1.86 GHz, f_{m4} = 2.13 GHz, f_{m5} = 2.26 GHz, f_{m6} = 2.39 GHz and f_{m7} = 2.40 GHz running on a 64bit CentOS 6.4.

Server virtualization has been performed using the QEMU-KVM hypervisor. The operating system installed in each virtual machine is a 64bit CentOS 6.4. In order to adapt the problem to Cloud computing environments, our model constants are calculated for the data obtained during the execution of the workload simultaneously in four KVM virtual machines using all the available CPU and memory resources in the



(a) 20 optimizations for each number of generations.



(b) Pareto Fronts for 30 optimizations.

Fig. 1. Analysis of Number of generations and Pareto Front final solutions.

Table 1. Constants obtained for Power curve fitting

Algorithm	α_1	α_2	α_3	α_4	β_1	β_2	β_3	γ	λ
OMOPSO	0	0	0	3.32	0	$1.63{\cdot}10^{-3}$	0	$4.88{\cdot}10^{-11}$	0
lsqcurvefit	$2.71{\cdot}10^{-12}$	$3.70{\cdot}10^{-10}$	$1.48{\cdot}10^{-8}$	3.50	$2.55{\cdot}10^{-10}$	$1.60{\cdot}10^{-3}$	$7.63{\cdot}10^{-9}$	$5.12{\cdot}10^{-11}$	$3.76{\cdot}10^{-10}$

server. In the case of the two dual core CPUs Intel Xeon in Fujitsu RX300 S6 server, each VM is assigned to a core and the 16GB RAM memory is divided into blocks of 4GB.

All the data included in the model have been collected via real measurements in the server using on board sensors. Power consumption has been measured using a current clamp. Hardware counters are collected using *perf* to monitor memory and CPU utilization. Frequency and voltage are modified via *cpufreq-utils* linux package. Measurements have been gathered during the execution of *lookbusy* and modified *RandMem* at different utilization levels ranging from 0% to 100% of the available resources of every VM. All data have been gathered for a range of room temperatures between 18°C and 25°C (CPU temperature ranges from 33°C to 64°C), and for each frequency and voltage supply level available in the server. We use these data to perform a regression to our model by applying both nonlinear curve-fitting algorithms. First, we fit the power curve using OMOPSO optimizations and then we compare the results with MATLAB *lsqcurvefit* fitting function to analyze its benefits. The function *lsqcurvefit* is defined to solve nonlinear curve-fitting problems in least-squares sense.

The data collected during the execution of the training set are used to perform 30 iterations of the OMOPSO optimization. We obtain 30 sets of solutions, each of them defining a Pareto front for the two objectives defined in our problem, as seen in Figure 1(b). The hypervolume of these Pareto fronts shows an average value of -1.0109 and a standard deviation of 0.0229; hence, it can be concluded that the algorithm is not trapped into a local minimum. Once we combine these Pareto fronts into a final one, we achieve the final set of solutions for our power modeling problem, also shown in Figure 1(b).

6.2. Results

In order to present some results that support the benefits reached by OMOPSO applied to our optimization problem, we choose a solution from the final Pareto front. We also obtain the only solution of the *lsqcurvefit* optimization applied to the same training data set so that we can compare both approaches. Table 2 shows the values of both the average relative error and maximum relative error percentages obtained applying OMOPSO and *lsqcurvefit*, whereas Table 1 shows the corresponding solution for these two objectives, i.e., the best values reached for the 9 constants included in our power model. These results show that, while *lsqcurvefit* uses all the constants of the model, OMOPSO provides nonzero values to three constants simplifying the power model. This also means that for *lsqcurvefit* we need to collect information from seven sensors and, for OMOPSO, only from five sensors, resulting in computational savings in the monitoring system. We validate the solutions obtained for the power model with both algorithms, OMOPSO

Table 2. Objectives for Power curve fitting

Algorithm	Avg.Error	Max.Error
OMOPSO	4.0328%	17.0693%
lsqcurvefit	4.8501%	16.9401%

Table 3. Average error percentage comparison for the tests performed

Workload	Training	Web Search	SPEC mcf	SPEC perlbench
OMOPSO	4.0328%	4.6028%	4.1242%	5.1148%
lsqcurvefit	4.8501%	4.4253%	6.1736%	5.2453%

and *lsqcurvefit*, using real Cloud computing workload. The validation of the model is conducted by executing three different tests that represent real workload of a Cloud computing data center: The *Web Search* application from the *CloudSuite*[3] benchmark suite, *SPEC_CPU2006 mcf* and *SPEC_CPU2006 perlbench*[4].

Web Search aims to characterize web search engines, typically used in Cloud infrastructures, that process client requests indexing data harvested from online sources. Our *Web Search* benchmark consists of four VMs. Three of the VMs perform as index serving nodes (ISNs) of Nutch 1.2 for a distributed file system with a data segment crawled from the public Internet of about 6 MB, and an index of 2 MB. One of the ISNs also performs as a Tomcat 7.0.23 frontend that sends index search requests to all the ISNs, collects their responses and sends them back to the requesting client. The clients behavior is simulated in the fourth VM using Faban 0.7 with a number of clients ranging from 100 to 300. The four VMs use all available memory and CPU in each server, as in the profiling tests. *Web Search* test set includes measurements of CPU temperatures from 37°C to 55°C and both CPU and memory loads range from 0% to 100%.

SPEC_CPU2006 mcf consists on a large-scale minimum-cost flow problem solved with a network simplex algorithm accelerated with a column generation. On the other hand, *SPEC_CPU2006 perlbench* is a mail based benchmark which applies a spam checking software to randomly generated email messages. Both the *SPEC_CPU2006 mcf* and the *perlbench* tests are conducted in parallel in 4VMs using entirely the available resources of the server. We choose these tests to represent HPC over a Cloud computing infrastructure as they are memory and CPU-intensive, and CPU-intensive applications, respectively. SPEC_CPU2006 data set includes measurements of CPU temperatures from 33°C to 54°C.

We calculate the values of $e_{\text{avg}}(\vec{x})$ and $e_{\max}(\vec{x})$ for the test data sets using the solutions of both OMOPSO and *lsqcurvefit* algorithms. The average percentage error results, $e_{\text{avg}}(\vec{x})$, can be seen in Table 3. These results show that OMOPSO not only simplifies the optimization problem for our power model but also provides better error results than *lsqcurvefit* for three of the four tests conducted. *Web Search* presents higher peaks of memory accesses per cycle in comparison with the rest of the tests. *lsqcurvefit* algorithm takes into account additional power contributions, that are not present in the OMOPSO formulation, which are more sensitive to the high variability in the memory utilization. However, the difference in the power estimated by both algorithms in this test is only 0.3W. Given the obtained results, we can conclude that the proposed methodology based on OMOPSO algorithms is an efficient technique for the envisioning of complex, multi-parametric power models for state-of-the-art Cloud computing servers. Moreover, the proposed technique allows to target several optimization problems that work on setting an energy-efficient working point by deciding the optimal clock frequency, voltage supply level and thermal-aware workload assignment.

7. Conclusions

This work has made successful advances in the provisioning of accurate power models of enterprise servers in Cloud services. As opposed to previous approaches, the work presented here targets different server architectures with no effort for the designer, as the models can be automatically generated. The use of multi-objective meta-heuristic optimization algorithms allows to include the traditional and non-traditional sources of power consumption, as well as the effect of several system-level parameters that affect the energy footprint. The experimental work, conducted with realistic workload, has shown the accuracy of the proposed methodology as compared with traditional regression algorithms. In addition, the multi-objective optimization approach followed in this paper opens the door to future energy minimization techniques where the parameters are considered as decision variables.

[3]http://parsa.epfl.ch/cloudsuite
[4]http://www.spec.org

8. Acknowledgment

Research by Marina Zapater has been partly supported by a PICATA predoctoral fellowship of the Moncloa Campus of International Excellence (UCM-UPM). This work has been partially supported by the Spanish Ministry of Economy and Competitiveness, under contracts TIN2008-00508, TEC2012-33892 and IPT-2012-1041-430000, and INCOTEC. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa).

References

[1] P. Scheihing, Creating energy efficient data center, in: Data Center Facilities and Engineering Conference, Washington DC, USA, 2007.
[2] J. Markoff, S. Lohr, Intel's huge bet turns iffy, New York Times Technology Section.
[3] R. Buyya, et al., Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges, in: PDPTA 2010, Las Vegas, USA, 2010.
[4] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, K. Pentikousis, Energy-efficient cloud computing, Comput. J. 53 (7) (2010) 1045–1051.
[5] P. Niles and P. Donovan, Virtualization and Cloud Computing: Optimized Power, Cooling, and Management Maximizes Benefits. White paper 118. Revision 3, Tech. rep., Schneider Electric (2011).
[6] A. L. Anthony, H. K. Watson, Techniques for developing analytic models., IBM Systems Journal 11 (4) (1972) 316–328.
[7] L. Bianchi, M. Dorigo, L. M. Gambardella, W. J. Gutjahr, A survey on metaheuristics for stochastic combinatorial optimization, Natural Computing: An international journal 8 (2) (2009) 239–287. doi:10.1007/s11047-008-9098-4.
[8] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308. doi:10.1145/937503.937505.
[9] N. Adiga, et al, An Overview of the BlueGene/L Supercomputer, in: SC '02, Salt Lake City, USA, 2002, pp. 60–60.
[10] M. Warren, et al., High-density computing: A 240-processor beowulf in one cubic meter, in: SC '02, Salt Lake City, USA, 2002, pp. 61–61.
[11] R. Ge, et al., Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters, in: SC '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 34–34.
[12] C.-H. Hsu, W.-C. Feng, A power-aware run-time system for high-performance computing, in: SC '05, IEEE Computer Society, Washington, DC, USA, 2005.
[13] G. Contreras, M. Martonosi, Power prediction for intel xscale processors using performance monitoring unit events, in: ISLPED, New York, NY, USA, 2005, pp. 221–226.
[14] A. Lewis, et al., Run-time energy consumption estimation based on workload in server systems, in: HotPower, Berkeley, CA, USA, 2008, pp. 4–4.
[15] S. Pelley, et al., Understanding and abstracting total data center power, in: WEED, 2009.
[16] F. Bellosa, The benefits of event: driven energy accounting in power-sensitive systems, in: ACM SIGOPS, New York, NY, USA, 2000, pp. 37–42.
[17] X. Fan, et al., Power provisioning for a warehouse-sized computer, in: ISCA, New York, NY, USA, 2007, pp. 13–23.
[18] D. Meisner, et al., Peak power modeling for data center servers with switched-mode power supplies, in: ISLPED, New York, NY, USA, 2010, pp. 319–324.
[19] G. Warkozek, et al., A new approach to model energy consumption of servers in data centers, in: ICIT, 2012, pp. 211–216.
[20] A. Bohra, V. Chaudhary, Vmeter: Power modelling for virtualized clouds, in: IPDPSW, 2010, pp. 1–8.
[21] J. Brandon, Going green in the data center: Practical steps for your sme to become more environmentally friendly, Processor 29 (39) (2007) 1–30.
[22] R. Miller, Google: Raise your data center temperature., `http://www.datacenterknowledge.com/archives/2008/10/14/google-raise-your-data-center-temperature/` (2008).
[23] Google Data Centers, Efficiency: How we do it. Temperature control, `http://www.google.com/intl/en_ALL/about/datacenters/efficiency/internal/#temperature` (Jan. 2014).
[24] N. El-Sayed, I. A. Stefanovici, G. Amvrosiadis, A. A. Hwang, B. Schroeder, Temperature management in data centers: why some (might) like it hot, SIGMETRICS Perform. Eval. Rev. 40 (1) (2012) 163–174.
[25] S. Narendra, A. Chandrakasan, Leakage in Nanometer CMOS Technologies, Integrated Circuits and Systems, Springer, 2010.
[26] J. Rabaey, Low Power Design Essentials, Engineering (Springer-11647), Springer, 2009.
[27] M. Bao, A. Andrei, P. Eles, Z. Peng, Temperature-aware idle time distribution for leakage energy optimization, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 20 (7) (2012) 1187–1200.
[28] M. Zapater, J. L. Ayala, J. M. Moya, K. Vaidyanathan, K. Gross, A. K. Coskun, Leakage and temperature aware server control for improving energy efficiency in data centers, in: DATE '13, EDA Consortium, San Jose, CA, USA, 2013, pp. 266–269.
[29] C. Reyes-Sierra, C. A. C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, International Journal of Computational Intelligence Research 2 (2006) 287–308.
[30] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence 1 (1) (2007) 33–57.
[31] M. R. Sierra, C. A. Coello Coello, Improving pso-based multi-objective optimization using crowding, mutation and dominance, in: EMO'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 505–519. doi:10.1007/978-3-540-31880-4-35.