Using Grammatical Evolution Techniques to Model the Dynamic Power Consumption of Enterprise Servers

Juan C. Salinas-Hilburg , Marina Zapater , José L. Risco-Martín , José M. Moya , José L. Ayala

Abstract—**The increasing demand for computational resources has led to a significant growth of data center facilities. A major concern has appeared regarding energy efficiency and consumption in servers and data centers. The use of flexible and scalable server power models is a must in order to enable proactive energy optimization strategies. This paper proposes the use of Evolutionary Computation to obtain a model for server dynamic power consumption. To accomplish this, we collect a significant number of server performance counters for a wide range of sequential and parallel applications, and obtain a model via Genetic Programming techniques. Our methodology enables the unsupervised generation of models for arbitrary server architectures, in a way that is robust to the type of application being executed in the server. With our generated models, we are able to predict the overall server power consumption for arbitrary workloads, outperforming previous approaches in the state-of-the-art.**

I. Introduction

Modern data centers are a huge source of power consumption and, hence, generate a tremendous amount of heat. The popularization of Cloud Computing and next-generation applications such as Smart Cities or e-Health, has dramatically increased the computational needs of data center facilities, and suposes an important challenge from the energy perspective. In 2010, data center electricity represented 1.3% of all the electricity use in the world, and 2% of all electricity use in the US [1]. In year 2012 alone, global data center power consumption increased to 38GW, and further rise of 17% to 43GW was estimated in 2013 [2].

Data center power budget is mainly devoted to the energy drawn by servers and the cooling needed to keep IT equipment under safe environmental conditions, avoiding thermal redlining [3]. In the last years Power Usage Efectiveness (PUE), defined as the ratio between total facility power and IT power, has become an important metric to measure the energy efficiency of these facilities. In year 2013, world average PUE reached 1.65 [4], whereas some major players such as Google are already reporting PUE values of around 1.13 [5]. Even though cooling efficiency minimizes the electricity bill, reducing PUE alone is not enough, as the major contributor to data center power is IT equipment, mainly enterprise servers.

Both industry and academy have focused their efforts on the development of data center optimization strategies to minimize energy from the computational and cooling perspective. In order to propose such policies we need to predict, with sufficient accuracy, the power consumption of the enteprise servers and the temperature attained when running a certain workload. Prediction enables the deployment of proactive optimization policies. Moreover, prediction is considered a must when data centers participate in demand-response programs for Smart Grid integration, as the facilities need to constantly forecast, track and adjust their power consumption. These techniques have recently proved to yield substantial energy savings, but require overall data center power prediction [6].

Recent research has shown the importance of splitting the various contributors to power in enterprise servers to leverage energy minimization strategies [7]. However, even though these strategies propose several models that isolate and quantify the various contributors to power, they lack the prediction of dynamic server power consumption, i.e. they cannot estimate server dynamic power given workload characteristics. Other techniques in the state of the art that model dynamic power make use of classical approaches based on data regression, but require manual model tuning. Data Center facilities are heterogeneous by nature, and a large set of servers from different architectures and manufacturers are usually found in the same data room. Therefore, in order to predict overall data center power, a model for each server needs to be created. In this sense, the use of models that require human interaction to be generated is not feasible.

Our work proposes the use of Genetic Programming techniques -and more specifically, Grammatical Evolution- to obtain a model for server dynamic power consumption in an unsupervised way, with minimal user interaction, as a first step towards data center wide power prediction. Our solution is robust to server architecture, workload allocation and applications. To develop our models we collect a significant amount of performance counters during runtime execution of the application, using them to predict server dynamic CPU power. By applying our dynamic power model with previous temperature, leakage and cooling models in the state of the art, we are able to estimate overall server power consumption with high accuracy.

The main contributions of our work can be summarized as follows:

- We propose an unsupervised modeling methodology based on Grammatical Evolution (GE) that uses Fea-

ture Engineering as a way to automatically extract relevant features, while obtaining a mathematical expression for dynamic power.

- We show how our model is able to predict the dynamic power as a function of performance counters with high accuracy, improving the Root Mean Square Error of classical approaches by more than 7%.
- We validate our approach on a presently-shipping enterprise server for a wide range of sequential and parallel applications, and show how our methodology can be extended to arbitrary scenarios.

Our work contributes to the state of the art by proposing the use of Gramatical Evolution as a way to predict, in an unsupervised way, dynamic power consumption of enterprise servers in data centers. Our solution enables the generation of models in heterogeneous data center environments without human interaction, leveraging the use of proactive optimization policies.

The remainder of the paper is organized as follows: Section II discusses the related work. Section III shows our experimental methodology. A classical modeling approach for dynamic power consumption is presented in Section IV, whereas Section V describes the Gramatical Evolution approach. Results are presented in Section VI and Section VII concludes the paper.

II. Related work

Previous work on server power modeling commonly focuses on estimating the dynamic power consumption of enterprise servers assuming that leakage has minimal impact. Lewis et al. [8] build a linear regression model based on performance events to determine run-time system-wide power prediction. Other models define overall server power as a quadratic function of CPU utilization [9]. The power modeling methodology vMeter [10], detect a correlation between the overall system power consumption and component utilization, and develops a linear total server power model. Cochran et al. [11] determine a set of relevant workload metrics for energy consumption minimization and handle tradeoffs between energy and delay. Our work, as opposed to others, takes in count the leakage power consumption in order to model the dynamic power consumption of an enterprise server.

Several works present models that isolate contributors to the overall server power consumption. Economou et al. [12] measure the total and component-level power for a set or workloads. Overall power is calculated with a non-intrusive method called Mantis. This model relies on the component-level power and a series of metrics such as utilization or hardware counters. Arjona et al. [13] isolate the power from CPU, disks and network, and propose a power and energy characterization of different type of servers. Isci et al. [14] proposed a methodology for runtime power monitoring with intrusive techniques. They isolate the overall CPU power value placing a clamp ammeter through the CPU power lines. At the same time, hardware counters are gathered to build a complete model of the overall CPU power. Even though there exist models in the literature to split the contributors to power consumption, those models do not describe the contributors to dynamic power consumption in an unsupervised way, i.e dynamic CPU power.

In terms of profiling, Ren et al. [15] present a continuous profiling infrastructure for data centers called Google Wide Profiling. The system uses *Oprofile* to sample hardware counters across different machines in multiple data centers. Profiling is performed on a small subset of machines in the data center to reduce profiling overhead. Schubert et al. [16] developed *eprof*, a software profiler that relates overall dynamic consumption energy to specific code sections. In order to work, *eprof* requires changes in the kernel of servers. They use CPU and memory linear models based on hardware counters to compute energy. Other works, like Bruening et al. [17] proposes the use of dynamic binary translation to make an instant profiling of selected phases of execution. The main drawback of the previous solutions is that they either need code instrumentation or changes in the server platform, which prevents automatization. Moreover, these techniques disregard the trade-offs in terms of temperature, leakage and fan power that affect overall server power.

As opposed to others, our work uses previous work on accurate server power modeling to first split the contributors to power consumption of enterprise servers, isolating dynamic power. Then, we apply an unsupervised modeling methodology to predict dynamic power consumption. As we prove later on, our devised methodology constitutes an effective technique for the power modeling of enterprise servers, as it exhibits higher accuracy than traditional approaches, and can be adapted to different and complex architectures in an unsupervised way. Because of the heterogeneous nature of data center, the proposed technique is particularly useful when new power sources need to be incorporated, and allows to model all servers in a data room, predicting overall data center power consumption.

III. Experimental methodology

A. Overview

In this paper we propose a model for server dynamic power consumption. To this end, we first follow the methodology in [7] to isolate and quantify the contributors to power. For instance, Equation 1 shows the overall server power and how its various contributors can be split. Our methodology is able to calculate overall server, fan, memory and disk power, allowing us to isolate the dynamic power consumption of the server.

$$P_{total} = P_{CPU} + P_{fan} + P_{memories} + P_{disks} \quad (1)$$

We execute a wide range of workloads and sample during runtime a diverse set of hardware counters for each workload. Also, we collect different parameters of the server associated with every workload: CPU temperature, fan speed, overall power, memory, fan and disk power.

To model dynamic power consumption in an unsupervised way, we propose the use of Gramatical Evolution. Our goal is to obtain a mathematical expression of dynamic power as a function of performance counters. To validate our approach we also generate a linear regression model following a classical approach.

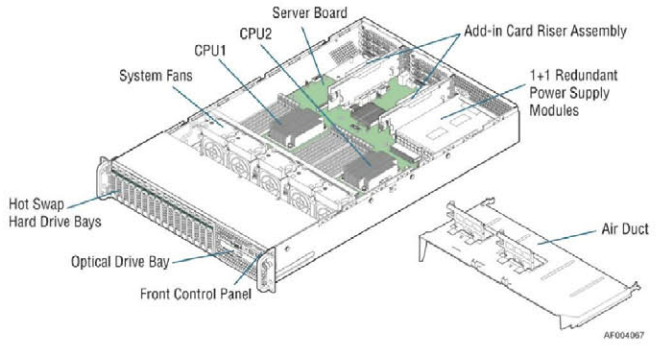


Fig. 1. Decathlete server internal diagram. Taken from [18]

B. Experimental setup

Our experiments take place on an Intel SandyBridge-EP server belonging to the Open Compute Project (OCP) [1] initiative, led by Facebook Inc. The idea behind choosing an OCP server is to exploit the benefits of flexibility and scalability brought by open-hardware designs, allowing to extend our customized monitoring, modeling and optimization setup easily to other platforms.

The server chosen is an Intel S2600GZ, whose design is based on an Intel OCP v2.0 Decathlete board. The board has two sockets, each can be equipped with a 6-core Intel SandyBridge-EP processor providing up to 12 hardware threads. The server is equipped with one Intel SandyBridge-EP processor, eight 4GB memory DIMMs, four 1TB hard drives, two PSUs and five fans. Figure 1 shows a diagram of the server internals.

The server runs a CentOS 6.5 Linux operating system. We use IPMI[2] to poll the available server sensors: i) CPU temperature, ii) fan speed and iii) overall server power consumption. Fan speed in this server can be controlled by setting different PWM values to the fan controllers via the BIOS [18]. For workload monitoring we use the *Oprofile* tool to poll the server hardware counters during runtime [3].

The original server monitoring via IPMI does not provide values for the CPU, fan, memory or disk power. In order to apply our modeling methodology we need to be able to split and quantify all these contributors to power. To this end, we deploy intrusive current measument sensors in the critical board components: i) fans, ii) memory DIMMs and iii) hard disk drives. This way we are able to isolate the contribution from cooling power, memory and disk from that of CPU power.

To measure power consumption, we use the commercial chip from Texas Instruments INA219. This chip uses an integrated power measurement circuit that measures the voltage drop in a *shunt* resistor placed in series with the power supply of the device to be measured. This setup allows to measure the power drawn by one memory DIMM, one fan, and the disks. The *shunt* resistor selected for each component must ensure a voltage drop low enough to keep the devices working.

[1]http://www.opencompute.org

[2]Intelligent Platform Management Interface

[3]http://oprofile.sourceforge.net/news/

Because fans and disks are powered directly via the PSU of the system, we can insert our sensor in between the power supply wires. However, because the memory DIMMs are powered via the motherboard, we need to insert a memory expander that incorporates the shunt resistors to enable power measurement.

To measure overal fan power we simply multiply the fan power of a single fan by the number of fans, as the server default fan control policy always drives all fans at the same speed. After running several experiments with memory-intensive benchmarks, we see that the power consumption of the memory is equally spread accross DIMMs, therefore we can also obtain total memory power multiplying by the amount of DIMMs in our system.

The INA219 current measurement chip has an I2C interface that we connect to a wireless microcontroller node that retrieves all the information from the chip and sends it wirelessly to a gateway [19]. The wireless node can be placed either inside or on top of the server, allowing to place the server inside a rack.

Finally, all the collected values are sent periodically via UDP to a monitoring tool called *graphite*. Every value can be retrieved from *graphite* on *csv* format file for post processing. In this scenario, power samples are gathered in 10-second intervals.

For more information on the monitoring setup the reader is referred to [20].

C. Test and Training sets

To develop the CPU dynamic consumption model we run a set of workloads to train and test our models. We use the following set of benchmarks:

- All the benckmarks from the CPU- and memory-intensive SPEC CPU2006 [21] benchmark suite.
- All benchmarks of PARSEC, a multi-threaded benchmark suite [22] that assesses the performance of multiprocessor systems.

In order to train our model we select a subset of 6 Integer benchmarks and 6 Floating Point benchmarks of SPEC CPU2006. This selection is made using the dendrogram provided by Phansalkar et.al. [23]. In the case of PARSEC, we select a set of benchmarks showing different computational behavior between them, according to [22].

- PARSEC: blackscholes, facesim, ferret, swaptions, vips and streamcluster.
- SPEC CPU2006: gcc, mcf, hmmer, sjeng, libquantum, xalancbmk, milc, cactusADM, soplex, povray, lbm and wrf.

To validate our model we use all remaining SPEC CPU2006 and PARSEC benchmarks:

- PARSEC: bodytrack, freqmine, raytrace, fluidanimate, x264 and canneal.
- SPEC CPU2006: perlbench, bzip2, gobmk, h264ref, omnetpp, astar, bwaves, gamess, zeusmp, gromacs,

leslie3d, namd, dealII, calculix, GemsFDTD, tonto and sphinx3.

D. Profiling methodology

In order to gather the hardware counters for modeling the dynamic power consumption we use *ocount*, an *Oprofile* tool that can be used to count hardware events for any specific application, and is inspired by the Digital Continuous Profiling Infrastructure [24]. The *ocount* tool takes samples from the hardware counters periodically and stores every set of values in a file. Hardware counters are a special set of registers that collect information on the performance of servers, and are generally used for power modeling.

We execute every SPEC CPU2006 and PARSEC benchmarks with *ocount* taking samples every 1 second of 21 hardware counters. PARSEC benchmarks were executed for every possible number of threads in our system, from 1 to 12 threads. SPEC CPU2006 benchmarks were executed for 1, 2, 3, 4, 5, 6 and 12 copies. Then, we calculate a 10-second mean on all the hardware counters collected. We do this in order to match the timestamp of the hardware counter event values and server sensors parameters extracted from *graphite*.

We calculate the correlation between counters and remove those whose correlation exceeds the 0.9 threshold. This threshold is heuristically selected to remove counters with high correlation between them. This way, we are able to reduce the set of hardware counters to 13, as shown in Table I.

To maintain a robust set of samples we discard all the benchmarks with less than 10 samples. Since the power phases of all the benchmarks are stable we select a representative set of samples from every benchmark. Also, we need approximately the same amount of samples from every benchmark to train the model in a balanced way. To this end, we perform a decimation on every benchmark with a factor of 10. This means every benchmark has approximately 10 samples.

E. CPU dynamic consumption

The CPU power component of Equation 1 can be divided into three components: (i) $P_{CPU,idle}$, which contains a temperature-independent leakage component plus the power consumption due to the Operating System running, (ii) $P_{CPU,leakT}$, a temperature-dependent leakage component, and (iii) $P_{CPU,dyn}$, the dynamic power of the CPU due to an application or workload execution:

$$P_{CPU} = P_{CPU,idle} + P_{CPU,leakT} + P_{CPU,dyn} \quad (2)$$

CPU power can be obtained through the overall power and the other components power from the server, as shown in Equation 1. The $P_{CPU,idle}$ can be easily calculated through Equations (1) and (2), when no workload is executed on the server. For the component $P_{CPU,leakT}$ we build a model and obtain a linear regression relating the power of leakage with the temperature of the CPU, as shown on Equation 3. The leakage model is obtained through the following methodology: we execute a CPU-intensive workload on the server, change step-by-step the speed of the fans (through the PWM parameter in the BIOS), and collect the CPU power consumption. This consumption is directly related to temperature-dependent leakage, as changing fan speed leads to different CPU temperatures for the same workload.

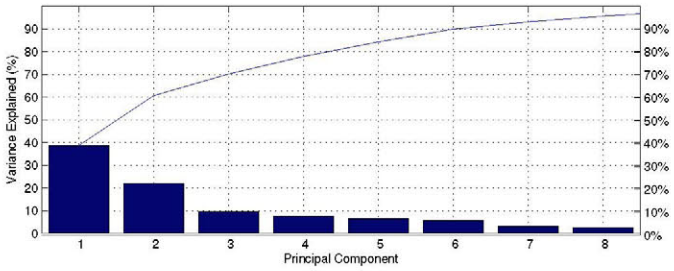


Fig. 2. Percentage of the total variance explained by each principal component.

$$P_{CPU,leakT} = \alpha_0 + \alpha_1 \cdot T_{CPU} + \alpha_2 \cdot T_{CPU}^2 \quad (3)$$

where regression coefficients are: $\alpha_0 = 27.5$, $\alpha_1 = -1.016$ and $\alpha_2 = 0.0112$.

To calculate CPU dynamic consumption, $P_{CPU,dyn}$, we use Equations (1) and (2), with the component $P_{CPU,idle}$ previously calculated. The quantization error of the total power consumption sensor of our server, P_{total}, is of 4W. Thus, the accuracy of our models is limited by this value.

IV. Server power modeling: Classical Approach

This section presents a partial least squares regression model for the dynamic power consumption of the server, that we use as a baseline for comparison. This approach provides an analytic expression of the dynamic consumption power based on the hardware counters. Additionally, it is a straightforward method with minimal computation overhead.

A. Feature selection

We use a *Principal Component Analysis* (PCA) to reduce the set of hardware counters to a lower dimension. Figure 2 shows the total variance explained by each component. Since the first 3 principal components explain 70% of the variance we plot the 3 principal components of the PCA output and choose the largest and most separated vectors, as shown in Figure 3. Those vectors represent the hardware counters that are highly independent from each other. The final hardware counter set is composed of the 1, 2, 5, 10, 11 counters shown in Table I.

B. Partial least squares regression model

We used the MATLAB function *plsregress* to find the analytic expression of the dynamic consumption power based on 5 hardware counters. We train and validate our model with the test and training sets explained in Section III. Equation 4 shows the dynamic power consumption linear regression expression:

$$\begin{aligned} P_{CPU,dyn} =& \beta_0 + \beta_1 \cdot C_1 + \beta_2 \cdot C_2 + \beta_3 \cdot C_5 + \\ & \beta_4 \cdot C_{10} + \beta_5 \cdot C_{11} \end{aligned} \quad (4)$$

TABLE I. SUMMARY OF RELEVANT HARDWARE COUNTERS

Counter	Number	Description
CPU_CLK_UNHALTED	1	Clock cycles when not halted
INST_RETIRED	2	Number of instructions retired
LLC_MISSES	3	Last Level cache demand requests from this core that missed the LLC
LLC_REFS	4	Last Level cache demand requests from this core
BR_INST_RETIRED	5	Number of branch instructions retired
BR_MISS_PRED_RETIRED	6	Number of mispredicted branches retired (precise)
misalign_mem_ref_1	7	Speculative cache-line split load uops dispatched to the L1D
misalign_mem_ref_2	8	Speculative cache-line split Store-address uops dispatched to L1D
arith	9	Number of times that the divider is actived, includes INT, SIMD and FP
resource_stalls	10	Core resource stalls (Cycles Allocation is stalled due to Resource Related reason)
uops_dispatched	11	Counts total number of uops dispatched from any thread
mem_trans_retired	12	Count memory transactions
mem_uops_retired	13	Count uops with memory accessed retired

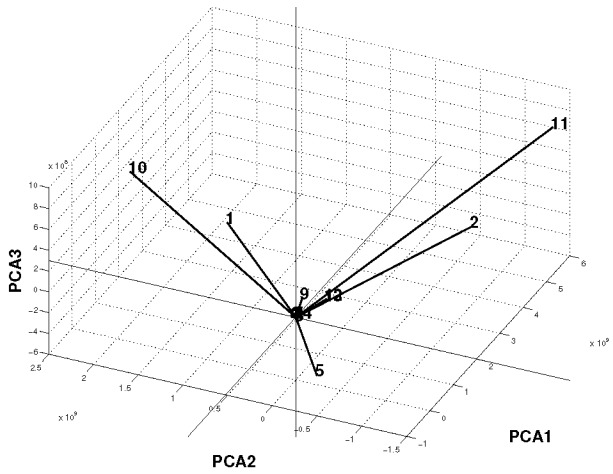


Fig. 3. Principal Components Analysis (PCA) for the hardware counters data set. Plot of the first 3 principal component axis.

where β_n corresponds to regression coefficients and C_m corresponds to hardware counters shown in Table I. The values of the regression coefficients are: $\beta_0 = 16.22$, $\beta_1 = 1.5 \cdot 10^{-9}$, $\beta_2 = 1.3 \cdot 10^{-9}$, $\beta_3 = -4.0 \cdot 10^{-10}$, $\beta_4 = -4.2 \cdot 10^{-12}$ and $\beta_5 = -6.0 \cdot 10^{-10}$. We compare the performace of this model vs. the model obtained by GE in Section VI.

V. SERVER POWER MODELING: GRAMATICAL EVOLUTION

In this section we describe our proposed modeling approach, as well as a brief description of grammatical evolution. Moreover, we describe the feature selection process and the fitness function used.

A. Feature selection and model definition

Evolutionary algorithms use the principles of evolution, i.e. survival of the fittest and natural selection, to turn one population of solutions into another, by means of selection, crossover and mutation. Among them, Grammatical Evolution (GE) [25] is a simple yet effective evolutionary computation technique based on Genetic Programming to perform symbolic regression [26]. GE is inspired in the biological process of generating a protein given the genetic material (DNA) of an organism. GE evolves computer programs given a set of rules, adopting a bio-inspired genotype-phenotype mapping process.

In every algorithm iteration, GE computes the fitness function for every iteration and extracts the mathematical expression given by an individual (phenotype) by applying a mapping process to the chromosome (genotype). This mapping process is achieved by defining a set of rules to obtain the mathematical expression, using grammars in Backus Naur Form (BNF) [25].

The process does not only perform parameter identification like in a classical regression method. In this sense, GE tries to simultaneously obtain a mathematical expression with the set of features that best fit the target system. This process is called Feature Engineering, and is a particularly useful technique to select the set of features and combination of variables that best describe a model.

GE is particularly useful to provide solutions that include non-linear terms offering Feature Engineering capabilities and removing analytical modeling barriers. Also, designer's expertise is not required to process a high volume of data as GE is an automatic method.

In this work, we propose the usage of Grammatical Evolution to obtain a mathematical expression for server dynamic CPU power. This expression is derived from experimental measurements of performance counters and power consumption values in a presently shipping enterprise server. Our goal is to develop a methodology for the unsupervised modeling of server CPU power, so that all servers in a data center facility can be modeled in an automatic way.

The goal of using GE is to obtain accurate models, thus, our fitness function needs to express the error resulting in the estimation process. To measure the accuracy in our prediction, we select the Root Mean Square Error (RMSE) as a fitness function.

B. Proposed solution

In order to develop the aforementioned model by means of Grammatical Evolution, we use the training set described in Section III, i.e. a decimated set of samples belonging to SPEC CPU and PARSEC benchmarks. As previously described, we use the 13 performance counters that have a correlation with power above 0.9. Because of the Feature Engineering capability of GE, we do not need to further reduce the features. We directly feed the GE algorithm with a grammar, the performance counters and the training set. As a result, after

a number of generations, we obtain a mathematical model that includes the most relevant features for power prediction. To analyze the error evolution, we let the models run for a large number of generations, until fitness value (i.e. RMSE in our case) does not improve further. Note that the lack of improve in fitness does not ensure a global optimum, it only shows that the algorithm has reached a local minimum.

Even though this paper focuses on CPU power prediction, it needs to be taken into account that the procedure to predict the dynamic power of other components (i.e. memory, network or disk) is completely equivalent. We only need to feed the algorithms with the adequate set of power data and the same performance counters.

In the next section we describe the grammars used and show results on the performance of our models as compared to the classical regression methodology.

VI. Results

In this section we present the results obtained for the modeling of dynamic CPU power with Grammatical Evolution techniques, and we also compare both the classical and GE approaches. First, we describe the algorithm setup as well as the grammars used and the results obtained in power prediction for the PARSEC and SPEC benchmarks. Finally, we discuss the applicability and overhead of our approach.

A. Algorithm setup and grammars

After evaluating the performance of our model with several setups, we select the following one for all models in this paper:

- Population size: 200 individuals
- Chromosome length: 100 codons
- Mutation probability: inversely proportional to the number of rules.
- Crossover probability: 0.9
- Maximum wraps: 3
- Codon size: 8 bits (values from 0 to 255)
- Tournament size: 2 (binary)

We perform variable standardization for every feature (in the range $[1, 2]$) to assure the same probability of appearance for all the variables and to enhance the GE symbolic regression. We have trained our models using two different grammars:

- GE Linear: this grammar is described in Grammar 1 and constraints the possible solutions to linear models. In this sense, GE selects the most relevant features and adjusts the constants, obtaining a model similar to the classical regression approach, i.e: $P_{CPU,dyn} = k_0 + \sum_{i=0}^{N}(k_i C_i)$
- Non-linear: the second grammar adds two new operands (product and division), to obtain more complex relations between performance counters, searching for a non-linear model. The only changes needed are the modification of Rules I and II, as shown in Grammar 2, whereas the remaining Rules keep unchanged. In this case GE selects not only the performance counters, but also their products. If the relation between dynamic power and performance counters is not linear, this grammar is expected to outperform both the GE linear and the classical regression approaches.

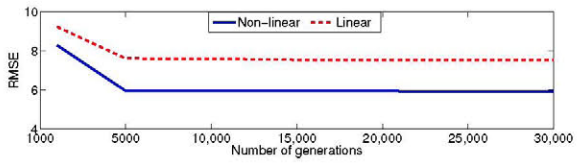


Fig. 4. RMSE evolution for training set when using the linear and non-linear grammars

TABLE II. Normalized phenotype obtained for both grammars used in dynamic power prediction

Model	Phenotype
Linear	$(0.4 \cdot C_1 + 1.2 - 0.3 \cdot C_8)$
Non-linear	$(1.2 + (C_2 \cdot C_{10}/((C_3/C_6) + ((1 + C_{10} + C_3 /(C_4 - C_{10}^2 + (C_1 \cdot C_4^2 - (C_{11}/C_{12}))/(C_{11}^2/C_8) /(((C_1 \cdot C_7 + (C_1/C_6)) \cdot C_1) - (C_5/C_9))))) \cdot C_{11} \cdot C_4))))$

Grammar 1 GE linear grammar used for dynamic CPU power modeling

$\langle expr \rangle$ ::= $\langle expr \rangle \langle op \rangle \langle expr \rangle$ | $\langle cte \rangle * \langle var \rangle | \langle var \rangle | \langle cte \rangle$ (I)

$\langle op \rangle$ ::= +|- (II)

$\langle var \rangle$ ::= C1|C2|...|C13 (III)

$\langle cte \rangle$ ::= $\langle dgt \rangle . \langle dgt \rangle$ (IV)

$\langle dgt \rangle$ ::= 0|1|2|3|4|5|6|7|8|9 (V)

Grammar 2 GE Non-linear grammar used for dynamic CPU power modeling

$\langle expr \rangle$ ::= $\langle expr \rangle \langle op \rangle \langle expr \rangle$ | $\langle var \rangle * \langle var \rangle | \langle var \rangle / \langle var \rangle | \langle var \rangle | \langle cte \rangle$ (I)

$\langle op \rangle$ ::= +|-|*|/ (II)

Figure 4 shows the evolution of the RMSE with the number of generations for both models. As can be seen, error stabilizes after 5,000 generations, being lower for the non-linear case. This indicates that second order relations exist between performance counters that are not being considered by linear grammars. The models obtained using both grammars are shown in Table II. As can be seen, the GE linear model selects less features than the classical regression approach, i.e. only C_1 and C_8. On the other hand, the non-linear model is more complex and incorporates multiplications and divisions of counters.

B. Dynamic power prediction

Table III shows a comparison in terms of RMSE, MAE and maximum error of the classical approach, the GE linear and GE non-linear models. As can be seen, the minimum

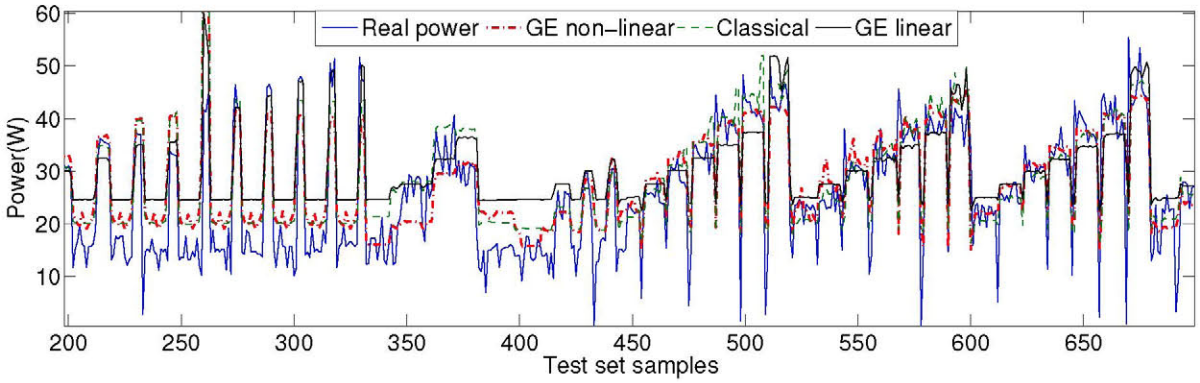


Fig. 5. Dynamic CPU power prediction for test set samples with GE and classical regression techniques

TABLE III. RMSE, MAE AND MAXIMUM ERROR FOR TRAINING AND TEST SET IN CLASSICAL AND GRAMMATICAL EVOLUTION MODELS

Model	Training set			Test set		
	RMSE	MAE	Max.	RMSE	MAE	Max.
Classical	6.6	4.8	37.1	7.8	5.4	79.3
GE Linear	7.5	5.6	46.4	7.9	5.7	35.5
GE Non-linear	5.9	4.1	40.8	7.21	5.5	37.1

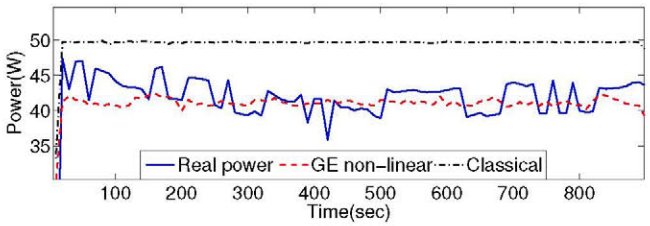
MAE obtained is 4.1W, being RMSE around 5.5 in the best-case scenario. Because the sensor used has an accuracy of 4W, we consider that our models are accurate enough. The GE linear model is worse than the classical approach in the training, but very similar in the test set. In fact, it significantly reduces maximum error in the test set compared to the classical approach, which is unable to follow spikes in power. The feature engineering performed by GE, which only selects two performance counters, outperforms the classical approach. On the other hand, the non-linear model outperforms both linear approaches, as it is able to incorporate more information.

Figure 5 shows the prediction of the samples in our test set, for the three models developed. As can be seen, the non-linear model is able to follow the trend of power more easily. However, all models are unable to follow very low power values, performing poorly when dynamic power is less than 15W.

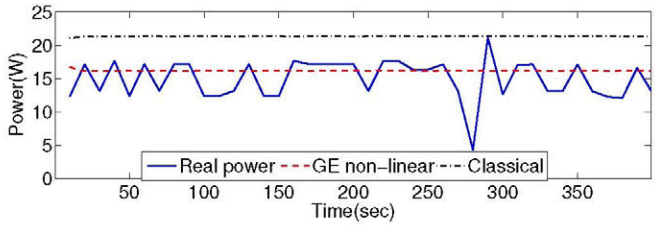
The previous Figure only shows the decimated samples of our test set. In Figure 6 we show the evolution of power consumption for all the samples in two particular tests: i) the benchmark Namd from SPEC CPU launched with 12 copies, and ii) the benchmark fluidanimate from PARSEC launched only with 1 thread. As opposed to the classical regression model, the GE Non-linear model is able to track the power consumption of both benchmarks during their entire execution, providing good results both for high and low power values.

C. Applicability and computational effort

The proposed modeling approach allows the prediction of dynamic CPU power consumption without user interaction, in a way that is robust to the workload and the server architecture. The methodology to model any other server is completely equivalent. Having predicted dynamic CPU power, the result can be used to predict temperature, leakage and overall server power. This enables the development of workload and cooling-aware strategies in the data center.



(a) Dynamic CPU power prediction for 12 copies of Namd



(b) Dynamic CPU power prediction for 1-thread Fluidanimate

Fig. 6. Dynamic CPU power prediction for Namd and Fluidanimate for the classical regression and the GE Non-linear model

Our approach is computationally intensive during the training stage. At this point, we need to launch as many models as different server architectures we have in the data centers. According to our results, the GE model needs to evolve a random initial population of 200 individuals for 5,000 generations to obtain accurate results. In our experiments, training four models in parallel in a QuadCore Intel i7 CPU@3.4GHz and 8GB of RAM takes 3 hours.

As for the model testing, we predict a new power sample every 10 seconds, which is the maximum granularity of power samples. The overhead to test one model is found to be negligible.

VII. CONCLUSIONS

In this paper we have presented a modeling technique based on Gramatical Evolution to model the dynamic power

consumption of enteprise servers in data centers. The proposed models are generated in an unsupervised way, by means of the Feature Enginnering capabilities of GE techniques. We have trained and tested two different models for CPU dynamic power based, using a linear and a non-linear model, and showed how adding complex relations between counters improves accuracy. We have compared our results with a classical feature selection and least-squares modeling approach.

Our models have been validated with a wide range of sequential and parallel applications running with different copies and number of threads, in a real enteprise server, obtaining an average error of 5.5W for the test set. Our modeling methodology outperforms classical regression techniques, improving RMSE by 7.5% while drastically reducing maximum error by 53%.

In the future, we plan to extend our methodology to the modeling of memory, network and disk, using a wider variety of workloads that include IO-intensive applications. Moreover, we also plan to validate our methodology in heterogeneous architectures. Because of the unsupervised nature of GE techniques, the changes needed to train new models are minimal.

ACKNOWLEDGMENT

This project has been partially supported by the Spanish Ministry of Economy and Competitiveness, under contracts TEC2012-33892, IPT-2012-1041-430000 and RTC-2014-2717-3.

REFERENCES

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," Analytics Press, Oakland, CA, Tech. Rep., 2011.

[2] A. Venkatraman, "Global census shows datacentre power demand grew 63% in 2012," http://www.computerweekly.com/news/2240164589/Datacentre-power-demand-grew-63-in-2012-Global-datacentre-census, October 2012.

[3] M. Iyengar and R. Schmidt, "Analytical modeling for thermodynamic characterization of data center cooling systems," *Journal of Electronic Packaging*, vol. 113, Feb. 2009.

[4] J. K. Matt Stansberry, "Uptime institute 2013 data center industry survey," Uptime Institute, Tech. Rep., 2013.

[5] G. Inc., "Efficiency: How we do it." [Online]. Available: http://www.google.com/about/datacenters/efficiency/internal/

[6] H. Chen, M. C. Caramanis, and A. K. Coskun, "The data center as a grid load stabilizer," in *19th Asia and South Pacific Design Automation Conference, ASP-DAC 2014, Singapore, January 20-23, 2014*, 2014, pp. 105–112.

[7] M. Zapater, O. Tuncer, J. L. Ayala, J. M. Moya, K. Vaidyanathan, K. Gross, and A. K. Coskun, "Leakage-aware cooling management for improving server energy efficiency," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2014, in Press, to appear in 2014.

[8] A. Lewis and et al., "Run-time energy consumption estimation based on workload in server systems," in *HotPower*, Berkeley, CA, USA, 2008, pp. 4–4.

[9] X. Fan and et al., "Power provisioning for a warehouse-sized computer," in *ISCA*, New York, NY, USA, 2007, pp. 13–23.

[10] A. Bohra and V. Chaudhary, "VMeter: Power modelling for virtualized clouds," in *IPDPSW*, 2010, pp. 1–8.

[11] R. Cochran, C. Hankendi, A. Coskun, and S. Reda, "Identifying the optimal energy-efficient operating points of parallel workloads," in *ICCAD*, 2011, pp. 608–615.

[12] D. Economou, S. Rivoire, and C. Kozyrakis, "Full-system power analysis and modeling for server environments," in *In Workshop on Modeling Benchmarking and Simulation (MOBS*, 2006.

[13] J. Arjona Aroca, A. Chatzipapas, A. Fernández Anta, and V. Mancuso, "A measurement-based analysis of the energy consumption of data center servers," in *Proceedings of the 5th International Conference on Future Energy Systems*, ser. e-Energy '14. New York, NY, USA: ACM, 2014, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/2602044.2602061

[14] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," Tech. Rep., 2003.

[15] G. Ren, E. Tune, T. Moseley, Y. Shi, S. Rus, and R. Hundt, "Google-wide profiling: A continuous profiling infrastructure for data centers." *IEEE Micro*, vol. 30, no. 4, pp. 65–79, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/micro/micro30.html#RenTMSRH10

[16] S. Schubert, D. Kostic, W. Zwaenepoel, and K. G. Shin, "Profiling software for energy consumption," in *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications*, ser. GREENCOM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 515–522. [Online]. Available: http://0-dx.doi.org.tkplib01.tut.ac.za/10.1109/GreenCom.2012.86

[17] D. Bruening, S. Mahlke, and R. Hank, "Instant profiling: Instrumentation sampling for profiling datacenter applications."

[18] Intel, "Server Board S2600GZ/GL. Technical Product Specification," 2014 (Revision 2.1).

[19] J. Pagán, M. Zapater, O. Cubo, P. Arroba, V. Martín, and J. M. Moya, "A Cyber-Physical approach to combined HW-SW monitoring for improving energy efficiency in data centers," in *Conference on Design of Circuits and Integrated Systems*, ser. DCIS'13, 2013, pp. 140–145.

[20] Juan C. Salinas-Hilburg, "Analysis and characterization of a high performance server and its impact on the energy consumption of a data center," Master Thesis, Universidad Politécnica de Madrid, Tech. Rep., 2014.

[21] SPEC CPU Subcommittee and John L. Henning, "SPEC CPU 2006 benchmark descriptions," http://www.spec.org/cpu2006/.

[22] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *PACT*, 2008, pp. 72–81.

[23] A. Phansalkar, A. Joshi, and L. K. John, "Subsetting the spec cpu2006 benchmark suite," *SIGARCH Computer Architecture News*, vol. 35, no. 1, pp. 69–76, 2007.

[24] J. M. Anderson, L. M. Berc, J. Dean, S. Ghemawat, M. R. Henzinger, S.-T. A. Leung, R. L. Sites, M. T. Vandevoorde, C. A. Waldspurger, and W. E. Weihl, "Continuous profiling: Where have all the cycles gone?" in *ACM Transactions on Computer Systems*, 1997, pp. 1–14.

[25] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, Aug 2001.

[26] C. Ryan, J. Collins, and M. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *Genetic Programming*, ser. Lecture Notes in Computer Science, W. Banzhaf, R. Poli, M. Schoenauer, and T. Fogarty, Eds. Springer Berlin Heidelberg, 1998, vol. 1391, pp. 83–96.