Unsupervised Power Modeling of Co-Allocated Workloads for Energy Efficiency in Data Centers

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

*LSI, Electronic Engineering Dpt. - Center for Computational Simulation,
Universidad Politécnica de Madrid, Madrid 28040, Spain, jcsalinas@die.upm.es, josem@die.upm.es
†DACYA, Universidad Complutense de Madrid, Madrid 28040, Spain, marina.zapater@ucm.es, jlrisco@ucm.es, jayala@fdi.ucm.es

***Abstract*—Data centers are huge power consumers and their energy consumption keeps on rising despite the efforts to increase energy efficiency. A great body of research is devoted to the reduction of the computational power of these facilities, applying techniques such as power budgeting and power capping in servers. Such techniques rely on models to predict the power consumption of servers. However, estimating overall server power for arbitrary applications when running co-allocated in multi-threaded servers is not a trivial task. In this paper, we use Grammatical Evolution techniques to predict the dynamic power of the CPU and memory subsystems of an enterprise server using the hardware counters of each application. On top of our dynamic power models, we use fan and temperature-dependent leakage power models to obtain the overall server power. To train and test our models we use real traces from a presently shipping enterprise server under a wide set of sequential and parallel workloads running at various frequencies We prove that our model is able to predict the power consumption of two different tasks co-allocated in the same server, keeping error below 8W. For the first time in literature, we develop a methodology able to combine the hardware counters of two individual applications, and estimate overall server power consumption without running the co-allocated application. Our results show a prediction error below 12W, which represents a 7.3% of the overall server power, outperforming previous approaches in the state of the art.**

I. Introduction

Data centers are outstanding power consumers and generate a tremendous amount of heat, making energy efficiency in these facilities an important research challenge. In 2010, data centers were responsible for 1.3% of all the electricity use in the world [1]. Nowadays data centers consume between 2 and 2.5% of the electricity in Europe, and keep rising faster than any other industry [2]. The major contributors to data center power are the energy drawn by computational (IT) equipment and the cooling needed to prevent IT thermal redlining. Both research and industry have proposed solutions to reduce data center cooling costs [3]. Thus, Power Usage Effectiveness (PUE) improved from 2.5 in 2007 to 1.65 in 2013 [4]. Despite the increase in energy efficiency, IT power, dominated by enterprise servers [5], still represents one of the top challenges for exascale research [6].

To reduce IT power consumption, techniques such as power budgeting propose methods to partition the total power budget among servers [3], usually ensuring that maximum per-server power remains below a cap [7]. Resource management aims to optimize the allocation of tasks to servers, according to their power consumption [8]. When data centers are integrated in the Smart Grid, the regulation capabilities of servers enable the participation in demand-response programs [9].

Previous techniques rely on models to predict server power consumption. Estimating power in highly-multithreaded enterprise servers running arbitrary workloads under a given frequency setup is not a trivial task. Simplified linear or quadratic models [10] exhibit high errors in these scenarios. They disregard the impact of leakage and fan power, and assume the same trend for CPU and memory power. Due to the complexity of modeling resource contention when various CPU or memory intensive tasks run on the same server, task co-allocation in non-virtualized scenarios is either not considered or limited to a particular set of known workloads [11], usually profiled off-line [3]. The validity of these approaches is limited, not matching real data center conditions.

In our previous work [12], we showed how Evolutionary Techniques, such as Grammatical Evolution (GE), could be used to predict CPU power during runtime in multi-threaded enterprise servers as a function of hardware (HW) counters, with negligible overhead. In this paper, we leverage this concept, and propose a methodology and model to predict the overall power consumption of an enterprise server running a realistic set of two co-assigned tasks, given the parameters (i.e. HW counters) of individual applications. Our contributions are:

- We model CPU and memory power of an enterprise server given the per-application HW counters, using GE techniques. We use analytical fan and leakage power models to obtain overall server power. Our models are trained and tested using a wide range of sequential and parallel workloads, under various DVFS setups, improving error by a 32% when compared to a traditional approach.
- We show that our model is robust enough to predict the power consumption of two different tasks when they run co-assigned in the same server, given the HW counters of the overall server.
- We develop a methodology to, given the HW counters of individual tasks, obtain the HW counters when both applications are co-allocated (without executing the co-allocated applications).

The previous contributions allow us to predict the overall server power consumption for two arbitrary tasks co-allocated

978-3-9815370-6-2/DATE16/

in the same server, given only the HW counters of individual applications. Once our model is trained, the power consumption of any pair of tasks can be estimated without the need to run them together. As our work targets the type of workloads found in High Performance Computing (HPC) data centers, we can restrict our analysis to the co-allocation of pairs of different workloads as it is unlikely to find more than 2 different types of workloads running on the same server in an HPC facility. Our models are tested in an Intel Sandy Bridge enterprise server, obtaining an error below 7.3% of overall server power.

The paper continues by discussing the related work. Section III describes the experimental methodology. Section IV and Section V present our modeling techniques. Results are presented in Section VI, and Section VII concludes the paper.

II. Related work

Energy-minimization resource management techniques usually require the *a priori* knowledge of the power consumption attained by servers when running a specific workload. Utilization and Instructions per Cycle (IPC) have traditionally been used as metrics to predict the power [13], [14], using linear or quadratic models [10]. However, these metrics are not sufficient to derive accurate models for the power consumption of highly-multithreaded enterprise servers [15]. To overcome this limitation, HW counters have been used to model CPU and memory power [16], as they provide information on workload characteristics. Previous works perform classical feature selection (e.g. correlation analysis, or Principal Component Analysis) to extract relevant parameters for a set of applications. Then, models are fitted using classical regression methods such as linear regression or multiple linear regression [17]. These methods require user interaction to discover the relevant features, train and test the models. Our work, on the contrary, leverages the usage of unsupervised techniques that automatically perform feature selection for power prediction. Moreover, we also consider the contribution of leakage and fan power to overall server power.

Other power-saving mechanisms, such as power-capping, reduce the voltage-frequency setup (DVFS) of servers to minimize power. Due to the lack of models able to predict power consumption under arbitrary workloads and frequency constraints, some approaches experimentally derive the power-frequency curves of each application [9]. Others enforce a power cap by using feedback controllers [18]. Newer mechanisms such as Intel's Running Average Power Limit (RAPL) [19] automatically implement that feature on newer SandyBridge systems. However, the former case requires profiling all the incoming tasks, whereas the latter sets an arbitrary power limit by decreasing frequency, potentially degrading performance. Our work, on the contrary, predicts the power consumption of tasks under various frequency setups, by using application characteristics measured via HW counters, overcoming the limitations of previous approaches. Moreover, it can be combined with frequency optimization policies to leverage energy efficiency.

Our work also tackles power prediction in task co-allocation. Research in this area is mainly focused on consolidation in virtualized environments [20]. However, due to the high number of cores in today's servers, it is common in HPC clusters to find a certain degree of task co-allocation (i.e. two different multi-threaded tasks running on the same server). Previous work addresses this challenge by trying to attribute the total CPU power to each of the tasks [21], or by deriving the power consumption of co-allocated tasks given the power of individual tasks using regression techniques [11], obtaining errors around 10% in overall server power. As opposed to our work, theirs require profiling all new incoming tasks to obtain per-application power profiles. Because our power model is robust to task co-allocation, we only need application parameters. Thus, power consumption for co-allocated tasks can be predicted using the parameters of individual applications, reducing error when compared to previous approaches.

III. Experimental methodology

This work proposes a methodology to model the overall power of servers running single or co-allocated applications, given the parameters (HW counters) of each individual task. Fig. 1 shows how the models developed in this paper can be used together to predict overall server power. To predict the power attained by a server when jointly running Tasks 1 and 2, we first gather the HW counters of each application separately (i.e., when not sharing the server with other tasks). Then, we develop a model to estimate the HW counters when both tasks run together in the same server (i.e. co-allocated tasks, as described in Sec. V). Finally, we use the predicted HW counters to model the power of the co-allocated tasks by using the overall server power model (described in Sec. IV).

A. Experimental Setup

The experiments take place on an Intel S2600GZ server, which is based on the Intel Decathlete v2.0 Open Compute Project server board. The server is equipped with one Intel 6-core SandyBridge-EP processor providing up to 12 HW threads, 8 4GB memory modules, 4 hard disk drives, 5 fans and 2 PSUs. The server runs a CentOS 6.5 Linux OS. Fig. 2 shows a diagram of the server internals. We use the IPMI to poll three server sensors: i) overall server power consumption (W), ii) CPU temperature (^{o}C) and iii) fan speed (RPM). Since the server is not equipped with power sensors for CPU, fans, memory or disks, we deploy intrusive current measurement sensors in three board components: i) one fan, ii) one memory DIMM and iii) the 4 disks. To obtain power values, we use the commercial chip from Texas Instruments INA219. To monitor the fan and disks we placed the sensor in series with the power supply of each component. Only one power sensor is used for the fans, as the server fan control firmware always drives all fans to the same speed. Thus, total fan power can be obtained by multiplying the measured fan power by the number of fans. Regarding memory, to measure the power of one memory DIMM, we inserted a memory extender that incorporates a shunt resistor to enable power measurement. We characterized memory power by running several memory-intensive experiments with the synthetic benchmark Randmem [1], and changing the DIMM we were monitoring. This way, we experimentally validated that memory power consumption is equally spread across all DIMMs, regardless of the workload run. Therefore, to measure overall memory power we can multiply the value from the power sensor by the number of DIMMs in the server.

To gather the HW counters of applications, we use *ocount*, an *Oprofile* [2] tool used to count HW events during runtime.

[1] https://github.com/greenlsi/randmem

[2] http://oprofile.sourceforge.net/about/

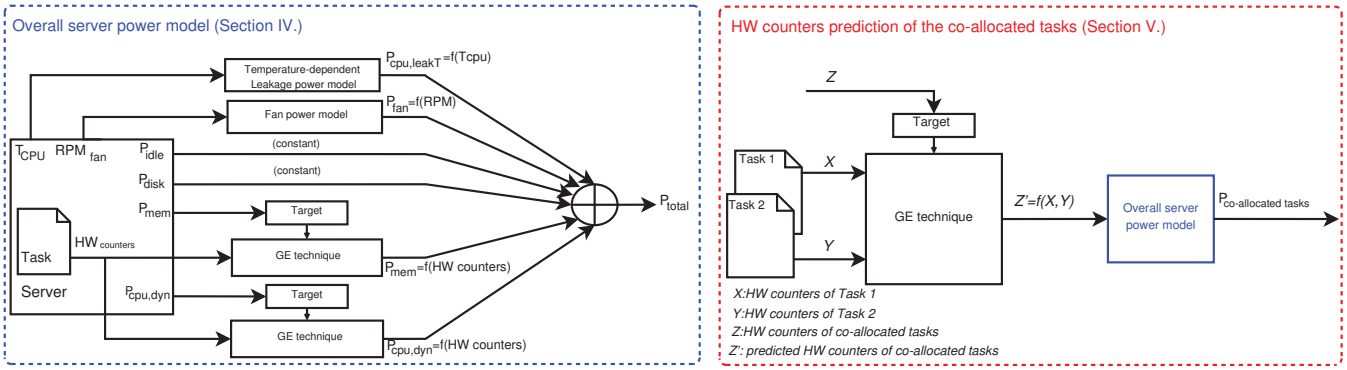


Fig. 1. Overview of the co-allocated tasks power model methodology

TABLE I. HW COUNTERS COLLECTED IN EACH EXECUTION

	Description
C1	Clock cycles
C2	Instructions retired
C3	LLC misses
C4	L2D Cache misses
C5	Branch instructions retired
C6	Mispredicted branches retired
C7	Speculative cache-line split load μops dispatched to the L1D
C8	Speculative cache-line split Store-address μops dispatched to L1D
C9	Number of times the divider is active
C10	Resource stalls
C11	μops dispatched
C12	Memory transactions
C13	μops with memory accessed retired
C14	L1D Cache misses

Table I shows the HW counters collected in each execution. This set of HW counters are proved to be correlated with the power consumption of running tasks [22]. All the parameters (CPU temperatures, fan speed, HW counters, and overall, fan, disk and memory power) are gathered every 10 seconds.

To develop our overall server power model we run a subset of workloads from the SPEC CPU 2006 [23] benchmark suite, the PARSEC [24] benchmark suite and Randmem. The SPEC and PARSEC benchmarks are selected according to their CPU and memory power characteristics, i.e. to train our model we choose 6 benchmarks with very different CPU and memory power profiles that sweep a wide range of values. The training set consists on: *lbm*, *calculix*, *gcc*, *bodytrack*, *blackscholes*, *streamcluster* and *randmem*. To validate our model we use a different set of benchmarks: *mcf*, *perlbench*, *bzip2*, *freqmine*, *raytrace* and *facesim*. Each SPEC/PARSEC benchmark is run for a different number of copies/threads: 1, 2, 3, 6 and 12. We use Randmem to stress various memory usages, ranging from 1 to 32GB. All experiments were run for the following frequency setups available in the server: 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000 MHz and the Turbo Boost frequency of 2001 MHz. Turbo Boost technology allows processors to run at a higher frequency than 2001 MHz, in our case at 2400MHz.

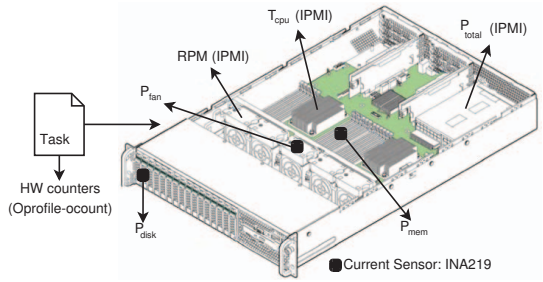


Fig. 2. Experimental Setup. Parameters collected during the experiments

We measure the real frequency by running several benchmarks while collecting Clock cycles (C1) every second.

To predict the HW counters of a co-allocated task given the HW counters of the individual tasks we run a set of experiments with co-allocated applications. We choose only two applications running together since is a typical HPC scenario. The training set is formed by the following co-allocated set of benchmarks: *perlbench-mcf*, *bzip2-lbm*, *perlbench-calculix*, *lbm-mcf*, *gcc-perlbench* and *calculix-mcf*. The test set is formed by: *perlbench-lbm*, *calculix-lbm*, *gcc-bzip2*, *calculix-bzip2* and *mcf-gcc*. Each experiment is run for 1, 2 and 3 copies of each application (e.g *Experiment 1:* 1 copy of *perlbench* and 1 copy of *mcf*, *Exp. 2:* 2 copies of *perlbench* and 2 copies of *mcf*, and so on.). Experiments were run under two frequencies: 1700 and 2001 MHz. The train and test set were selected by randomly combining benchmarks of their respective sets.

IV. SERVER POWER MODELING

The proposed server power model isolates and separately quantifies the main contributors to power consumption: i) CPU power P_{CPU}, ii) memory power P_{mem}, iii) fans P_{fan} and iv) disks P_{disk}. Eq. 1 describes the power consumption breakdown. CPU power can be further divided into three contributors: i) P_{idle}, is the power consumption of the CPU and other components of the server (motherboard, service processor, network interface, etc) when no workload is executed; it contains a temperature-independent leakage component plus the power consumption due to the OS running; ii) $P_{CPU,leakT}$, represents temperature-dependent leakage, which has an exponential dependence with temperature [15]; and iii) $P_{CPU,dyn}$, the dynamic power consumption of the CPU due to a workload execution. As mentioned previously, we are not able to measure directly CPU power consumption (P_{CPU}), so we calculate this term by subtracting all other power values (fan, memory and disk power) from overall server power (P_{total}). The quantization error of P_{total} is equal to 4W, this establishes a floor to the accuracy of the P_{CPU} value.

$$P_{total} = \underbrace{P_{idle} + P_{CPU,leakT} + P_{CPU,dyn}}_{P_{CPU}} + P_{fan} + P_{mem} + P_{disk} \quad (1)$$

Leakage power was modeled using the same methodology than in previous work by Zapater *et al.* [15]. To this end, we run a CPU-intensive workload under various fan speeds (fan speed can be manually set through the server BIOS), while collecting CPU power and temperature. Because leakage power depends exponentially on CPU temperature, changing

TABLE II. COEFFICIENT VALUES FOR ALL MODELS

	Coefficient Values										
	0	1	2	3	4	5	6	7	8	9	10
α	27.50	-1.02	0.01	-	-	-	-	-	-	-	-
β	0	$4.21E{-4}$	$-6.11E{-8}$	$1.14E{-11}$	-	-	-	-	-	-	-
γ	4.10	$-1.04E{-10}$	$8.92E{-11}$	$2.58E{-10}$	$8.94E{-12}$	-	-	-	-	-	-
δ	5.13	$-4.75E{-10}$	$2.33E{-11}$	$6.06E{-10}$	$1.06E{-10}$	-	-	-	-	-	-
x	53.10	$1.00E{-10}$	$1.05E{-10}$	9.83	$4.03E{-12}$	$3.72E{-10}$	$2.49E{-12}$	$1.12E{-12}$	58.19	-	-
y	24.80	$1.00E{-10}$	$1.77E{-11}$	5.76	$1.00E{-10}$	$2.49E{-12}$	$3.72E{-10}$	24.80	-	-	-
η_0	$1.70E9$	0.78	1.29	0.06	0.08	-0.02	-0.08	-0.05	-0.10	-	-
η_1	$7.10E9$	0.01	0.32	1.47	0.63	-0.46	-0.43	-0.53	0.25	-	-
η_2	$-1.35E9$	-0.19	-0.07	-0.34	0.05	1.22	1.17	0.34	-0.01	-	-
η_3	$9.82E9$	0.37	1.32	1.17	0.12	-1.12	-1.23	-0.23	0.55	-	-
m	0.44	0.86	$5.15E9$	-	-	-	-	-	-	-	-
n	$1.44E9$	$6.24E{-10}$	$4.68E{-12}$	$5.34E{-12}$	$1.39E9$	-	-	-	-	-	-
p	0.62	0.50	$4.70E8$	-	-	-	-	-	-	-	-
q	1.27	$1.60E10$	$2.83E{-11}$	$2.23E{-10}$	$9.62E9$	-	-	-	-	-	-
r	$8.24E10$	$1.69E{-11}$	$1.88E10$	$5.41E{-12}$	$2.23E{-10}$	$1.73E{-11}$	$5.34E{-12}$	$1.02E11$	-	-	-
s	$5.97E10$	$5.41E{-12}$	$3.03E{-11}$	$2.65E{-10}$	$6.24E{-10}$	$1.73E{-11}$	$2.23E{-10}$	$2.75E11$	$4.68E{-12}$	$2.65E{-10}$	$3.06E11$
w	1.44	$1.47E9$	$4.68E{-12}$	$2.23E{-10}$	$2.59E{-10}$	$3.03E{-11}$	$9.92E8$	$2.65E{-10}$	$6.24E{-10}$	$5.41E{-12}$	$2.71E7$

fan speed under a constant workload varies CPU temperature, and thus, leakage. We regress leakage power to the second order Taylor series expansion of an exponential (see Eq. 2), obtaining the coefficients shown in Table II. This process is done through a normalization of the T_{CPU} values. The leakage power model has an RMSE (Root Mean Square Error) of 1.14W and a MAE (Mean Absolute Error) of 1.03W.

$$P_{leak} = \alpha_0 + \alpha_1 \cdot T_{CPU} + \alpha_2 \cdot T_{CPU}^2 \tag{2}$$

Fan power consumption was modeled by changing fan speed while gathering fan power values via the deployed sensor. As fan power has a cubic relation with fan speed, we regress fan power to a third order polynomial, as in Eq. 3. The regression coefficients obtained are shown in Table II. The fan power model has an RMSE of 0.09W and a MAE of 0.05W.

$$P_{fan} = \beta_0 + \beta_1 \cdot rpm_{fan} + \beta_2 \cdot rpm_{fan}^2 + \beta_3 \cdot rpm_{fan}^3 \tag{3}$$

Since we do not use disk-intensive benchmarks, the power consumption of disks (P_{disks}) is constant, with a mean power consumption of 16.75W and a standard deviation of 1.3W. Power consumption when no workload is running (P_{idle}) is also constant through all the experiments, with a mean power consumption of 50.0W and a standard deviation of 2.0W.

In this paper, we present a methodology to obtain the models for the memory (P_{mem}) power consumption and the dynamic CPU power consumption ($P_{CPU,dyn}$) as a function of HW counters. We propose an unsupervised modeling technique based on Grammatical Evolution (GE) that automatically extracts relevant features, while obtaining a mathematical expression for dynamic CPU and memory power. We compare our solution to a classical modeling approach based on classical feature selection and regression.

In summary, we claim that to model server power we only need to collect the following parameters: HW counters, CPU temperature and fan speed, as shown in Eq. 4:

$$P_{total} = f(HW_{counters}, T_{CPU}, rpm_{fan}) \tag{4}$$

A. Classical Approach

We first present a partial least squares regression to model dynamic CPU and memory power ($P_{CPU,dyn}$, P_{mem}). We use Principal Component Analysis (PCA) for feature selection, reducing the set of HW counters. In our experiments, the first 3 principal components explain 70% of the variance, so we plot the first 3 components and choose the HW counters that are highly independent from each other. The final HW counter set is composed of: C1, C2, C10, C11. We train and validate our model with the training and test sets described in Sec. III. Equation 5 shows the dynamic CPU and memory power linear regression expression. γ_n and δ_n are the regression coefficients and C_m correspond to HW counters. Regression coefficients for all models are summarized in Table II.

$$\begin{aligned} P_{CPU,dyn} =& \gamma_0 + \gamma_1 \cdot C_1 + \gamma_2 \cdot C_2 + \gamma_3 \cdot C_{10} + \gamma_4 \cdot C_{11} \\ P_{mem} =& \delta_0 + \delta_1 \cdot C_1 + \delta_2 \cdot C_2 + \delta_3 \cdot C_{10} + \delta_4 \cdot C_{11} \end{aligned} \tag{5}$$

B. Grammatical Evolution

Grammatical Evolution (GE) is an evolutionary technique, inspired on the biological process of generating a protein from DNA, that performs symbolic regression [25]. Given a set of composition rules that describe the mathematical relations and the variables of a model, GE generates and evolves a model until it fits the training data with the minimum error possible. As opposed to a classical approach, GE performs automatic feature selection, reducing the initial set of HW counters. We feed the GE algorithm with all the HW counters and variables we want to model (i.e. the target): CPU dynamic power ($P_{CPU,dyn}$) and memory power (P_{mem}). This process is done separately for both magnitudes, obtaining one model for $P_{CPU,dyn}$ as a function of HW counters and another for P_{mem}, as shown in Eq. 6 and Table II.

$$\begin{aligned} P_{CPU,dyn} =& x_0 \cdot \frac{x_1 \cdot C14 + 1.0}{x_2 \cdot C4 + 1} + \\ & x_3 \cdot \frac{(x_4 \cdot C10 + 1)(x_5 \cdot C3 + 1.0)(x_6 \cdot C2 + 1)^2}{x_7 \cdot C11 + 1} - x_8 \\ P_{mem} =& y_0 \cdot \frac{y_1 \cdot C14 + 1}{y_2 \cdot C5 + 1} + \\ & y_3 \cdot (y_4 \cdot C14 + 1)(y_5 \cdot C2 + 1)(y_6 \cdot C3 + 1) - y_7 \end{aligned} \tag{6}$$

V. POWER PREDICTION FOR CO-ASSIGNED TASKS

To predict the overall power of co-allocated applications we first predict the HW counters when two tasks run co-allocated. To this end, we model each counter of the CPU and memory models as a function of those same counters on the individual tasks. For the case of the classical model, counters C1, C2, C10 and C11 were selected to obtain the models of $P_{CPU,dyn}$ and P_{mem} (see Sec. IV). Thus, we predict the HW counters of the co-allocated tasks ($\hat{Ci}$) as a function of the HW counters of single tasks (Ci_{t_j}), using classical regression to fit the regression coefficients η (see Table II), as shown:

$$\begin{bmatrix}\hat{C1}\\ \hat{C2}\\ \hat{C10}\\ \hat{C11}\end{bmatrix}=\begin{bmatrix}\eta_{01} & \eta_{02} & \eta_{03} & \eta_{04} & \eta_{05} & \eta_{06} & \eta_{07} & \eta_{08}\\ \eta_{11} & \eta_{12} & \eta_{13} & \eta_{14} & \eta_{15} & \eta_{16} & \eta_{17} & \eta_{18}\\ \eta_{21} & \eta_{22} & \eta_{23} & \eta_{24} & \eta_{25} & \eta_{26} & \eta_{27} & \eta_{28}\\ \eta_{31} & \eta_{32} & \eta_{33} & \eta_{34} & \eta_{35} & \eta_{36} & \eta_{37} & \eta_{38}\end{bmatrix}\begin{bmatrix}1\\ C1_{t1}\\ C1_{t2}\\ C2_{t1}\\ C2_{t2}\\ C10_{t1}\\ C10_{t2}\\ C11_{t1}\\ C11_{t2}\end{bmatrix}$$

For the GE model, the counters C2, C3, C4, C5, C10, C11 and C14 were automatically chosen as features to model $P_{CPU,dyn}$ and P_{mem}. We feed the GE algorithm with these counters of the single tasks to obtain the counters of the co-allocated tasks (target). We derive 7 models, one per HW counter, as shown next:

$$\begin{aligned}
\hat{C2}&=m_0.C11_{t1}+m_1.C11_{t2}-m_2\\
\hat{C3}&=n_0\frac{(n_1.C3_{t2}+1)(n_2.C11_{t1}+1)}{n_3.C2_{t1}+1}-n_4\\
\hat{C4}&=p_0.C4_{t2}+p_1.C4_{t1}+p_2\\
\hat{C5}&=q_0.C5_{t2}+q_1\frac{q_2.C5_{t1}+1}{q_3.C14_{t1}+1}-q_4\\
\hat{C10}&=r_0(r_1.C10_{t2}+1)+\\
&\quad r_2\frac{(r_3.C2_{t2}+1)(r_4.C14_{t1}+1)(r_5.C10_{t1}+1)}{r_6.C2_{t1}}-r_7\\
\hat{C11}&=\frac{s_0(s_1.C2_{t2}+1)(s_2.C5_{t2}+1)(s_3.C4_{t2}+1)(s_4.C3_{t2}+1)(s_5.C10_{t1}+1)}{s_6.C14_{t1}+1}+\\
&\quad s_7\frac{s_8.C11_{t1}+1}{s_9.C4_{t2}+1}-s_{10}\\
\hat{C14}&=w_0.C4_{t2}+w_1\frac{(w_2.C11_{t1}+1)(w_3.C14_{t1}+1)(w_4.C4_{t1}+1)}{w_5.C5_{t2}+1}-\\
&\quad w_6\frac{(w_7.C4_{t2}+1)(w_8.C3_{t2}+1)}{w_9.C2_{t2}+1}+w_{10}
\end{aligned}\tag{7}$$

VI. Results

In this section we present the results obtained for the overall server power model and for the power prediction of co-assigned tasks obtained by following the process in Fig. 1. We compare the GE technique against a classical modeling approach using PCA analysis. The main advantage of the GE technique is the automatic feature selection, as opposed to the classical approach where the feature selection is not done in an automatic way.

A. Overall server power model

The first row of Table III shows the RMSE and MAE of the overall server power model without task co-allocation, for both the GE and classical models. We observe that the GE technique has a MAE of 5.82W and a RMSE of 7.40W for the test set, improving by 32.11% the RMSE value with respect to the classical approach.

Fig. 3 shows the value of the RMSE of the overall server power model for each frequency and copy/thread configuration. For instance, the RMSE value of the test set when the frequency is 1300MHz and we have one copy/thread is around 8W. The RMSE value decreases when the frequency increases, for 1, 2, 3 and 6 copies/threads configuration. This means that overall server power model is more accurate for higher frequencies than for lower frequencies. Again, higher frequencies increase the power consumption of applications, increasing the accuracy of our model. As the main usage of this modeling would be to estimate if a workload exceeds a certain cap, obtaining higher errors at low power values is not an important limitation. We see that accuracy is lower in the case of 12 copies/threads. This is mainly caused by the impact of resource contention, which is very extreme in these cases, as we are assigning highly CPU and memory bounded applications to all HW threads in the system. However, even in these adverse conditions our model is able to obtain errors below 10%. Another interesting case is the 2001MHz frequency (i.e. the Turbo Boost frequency, that corresponds to 2400MHz), where the RMSE rises as the number of copies/threads increases. This is caused both by resource contention and due to the underfitting of the model. In this sense, for the 2001MHz frequency we have less samples in our training sets compared to other frequencies (due to applications running faster and finishing early). Moreover, as the there is an important gap (400MHz) between the 2000MHz and the 2001MHz frequency, instead of 100MHz, our models tend to overfit lower frequencies, obtaining less accuracy for the 2001MHz. This issue is inherent to any regression technique that tries to minimize the RMSE across samples. We observe the same behavior for the classical modeling approach, but the GE techniques improve the error obtained.

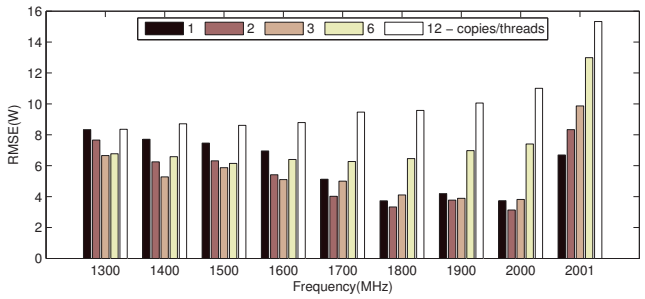
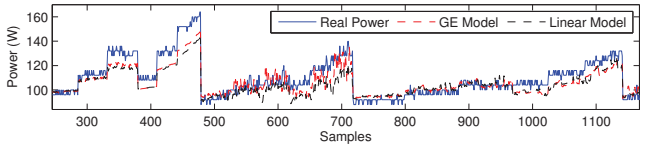
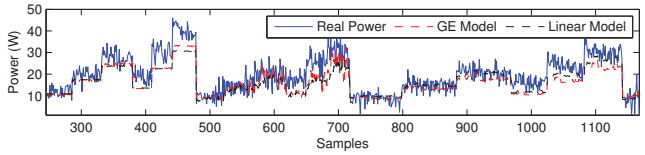


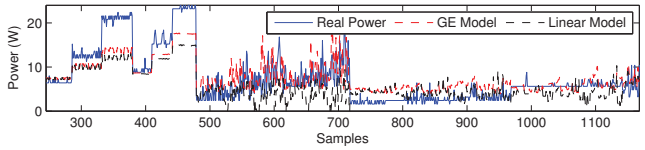
Fig. 3. RMSE of the overall server power model for each frequency configuration (Test set)



(a) Overall Server Power



(b) Dynamic CPU Power



(c) Memory Power

Fig. 4. Test set samples (subset). Power prediction of co-allocated tasks using HW counter prediction

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

	Training set				Test set			
	RMSE (W)		MAE (W)		RMSE (W)		MAE (W)	
	GE	Classical	GE	Classical	GE	Classical	GE	Classical
Overall Power	7.15	8.75	5.56	7.20	7.40	10.90	5.82	7.68
Overall power of co-allocated applications	8.06	13.15	6.13	10.46	9.52	12.64	7.48	9.25
Overall power of co-allocated applications (2vs1)	-	-	-	-	7.77	8.76	6.26	6.92
Overall power of co-allocated applications (4vs2)	-	-	-	-	12.86	15.91	10.14	12.50

B. Overall power of co-assigned applications

The third row of the Table III shows the RMSE and MAE in overall server power when two tasks are running at the same time in the server. As expected, the GE model outperforms the classical approach, with a MAE of 7.48W and a RMSE of 9.52W for the test set. Fig. 4 shows the prediction of the overall, dynamic CPU and memory power for the co-allocation scenario. We can see that both the GE and the classical model are able to predict fairly accurately the overall server power using HW counters prediction.

The last two rows of the Table III show the RMSE and MAE for an asymmetric task co-allocation. In the *2vs1* scenario, we run 2 copies/threads of one benchmark of the test set together with 1 copy/thread of another benchmark of the test set. For example, 1 copy of *perlbench* together with 2 copies of *lbm*, and also 2 copies of *perlbench* with 1 copy of *lbm*, and so on. In the *4vs2* scenario the process is identical. There are no training sets for the *2vs1* and *4vs2* scenarios, as the model is robust enough to predict the HW counters for an asymmetrical scenario and no re-training is needed. The RMSE value for the *2vs1* scenario is fairly low when compared to the *4vs2* scenario. The overall RMSE values are below 12W when the unsupervised GE model is used, which represents a 7.3% of the total server power.

The results of the models show errors with enough accuracy to apply different state-of-the-art policies to reduce the IT power consumption, thus improving energy efficiency in data centers. Policies like power-capping or resource management could benefit from the models obtained in this work.

VII. CONCLUSIONS

In this paper, we present a model based on Grammatical Evolution (GE) techniques to predict dynamic CPU and memory power in enterprise servers as a function of HW counters. Using leakage and fan power models from our previous work we develop an overall server power model that is robust enough to predict the power of co-allocated tasks without the need to train the model under that situation. In order to predict the power consumption of co-allocated tasks, we develop a methodology to predict the HW counters of two co-allocated tasks given the HW counters of the individual tasks. All the models only need to be trained one time per server. Moreover, we can predict the power consumption of a task that was not previously trained in the models.

The proposed models are generated in an unsupervised way, leveraging the usage of the automatic feature selection and extraction of GE techniques. Models have been trained and tested using real traces from a multi-threaded enterprise server, running a wide range of sequential and parallel applications, under various DVFS setups. Results have been compared against a classical feature selection and least-squares modeling approach. The overall RMSE values of the models are below 12W when the unsupervised GE model is used, which represents a 7.3% of the total server power.

ACKNOWLEDGMENT

This project has been 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] G. D. Net, "Introduction to the Green Data Net Project ," http://www.greendatanet-project.eu/, 2015.

[3] X. Zhan and et al., "Techniques for energy-efficient power budgeting in data centers," ser. DAC, 2013.

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

[5] D. Floyer, "Networks go green," http://wikibon.org/wiki/v/Networks_Go_GrEEN, 2011.

[6] DOE ASCAC Subcommittee, "Top ten exascale research challenges," U.S. Department of Energy (DOE), Tech. Rep., february 2014.

[7] O. Sarood and et al., "Optimizing power allocation to cpu and memory subsystems in overprovisioned hpc systems," in *IEEE CLUSTER*, 2013.

[8] X. Zheng and et al., "Markov model based power management in server clusters," in *GREENCOM*, 2010, pp. 96–102.

[9] H. Chen and et al., "Dynamic server power capping for enabling data center participation in power markets," in *ICCAD*, 2013, pp. 122–129.

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

[11] J. Choi *et al.*, "Power consumption prediction and power-aware packing in consolidated environments," *IEEE Trans. on Computers*, 2010.

[12] JC Salinas and et al., "Using grammatical evolution techniques to model the dynamic power consumption of enterprise servers," in *CISIS*, 2015.

[13] A. Bohra and et al., "Vmeter: Power modelling for virtualized clouds," in *IEEE (IPDPSW)*, 2010, pp. 1–8.

[14] R. Ayoub and et al., "Temperature aware dynamic workload scheduling in multisocket cpu servers," *IEEE TCAD*, 2011.

[15] M. Zapater and et al., "Leakage-aware cooling management for improving server energy efficiency," *TPDS*, vol. PP, no. 99, 2014.

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

[17] C. Mobius and et al., "Power consumption estimation models for processors, virtual machines, and servers," *IEEE TPDS*, 2014.

[18] R. Raghavendra and et al., "No "power" struggles: Coordinated multi-level power management for the data center," in *ASPLOS*, 2008.

[19] H. David and et al., "Rapl: Memory power estimation and capping," in *ISLPED*, 2010, pp. 189–194.

[20] G. Dhiman and et al., "A system for online power prediction in virtualized environments using gaussian mixture models," in *DAC*, 2010.

[21] Y. Zhai and et al., "Happy: Hyperthread-aware power profiling dynamically," in *USENIX Annual Conference*, 2014, pp. 211–218.

[22] R. Cochran and et al., "Identifying the optimal energy-efficient operating points of parallel workloads," ser. ICCAD '11, 2011, pp. 608–615.

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

[24] C. Bienia and et al., "The PARSEC benchmark suite: characterization and architectural implications," in *PACT*, 2008, pp. 72–81.

[25] C. Ryan and et al., "Grammatical evolution: Evolving programs for an arbitrary language," in *Genetic Programming*, ser. LNCS, 1998.