Predictive Modeling for Job Power Consumption in HPC Systems

Conference Paper *in* Lecture Notes in Computer Science · June 2016

DOI: 10.1007/978-3-319-41321-1_10

5 authors, including:

Andrea Borghesi
University of Bologna
45 PUBLICATIONS **415** CITATIONS
SEE PROFILE

Andrea Bartolini
University of Bologna
149 PUBLICATIONS **1,750** CITATIONS
SEE PROFILE

Michele Lombardi
University of Bologna
91 PUBLICATIONS **918** CITATIONS
SEE PROFILE

Michela Milano
University of Bologna
252 PUBLICATIONS **3,143** CITATIONS
SEE PROFILE

Predictive Modeling for Job Power Consumption in HPC Systems

Andrea Borghesi†, Andrea Bartolini‡, Michele Lombardi†, Michela Milano†, and Luca Benini‡

{andrea.borghesi3,michele.lombardi2,michela.milano,luca.benini}@unibo.it
barandre@iis.ee.ethz.ch
DISI, University of Bologna†
DEI, University of Bologna‡
Integrated Systems Laboratory, ETH, Zurich‡

Abstract. Power consumption is a critical aspect for next generation High Performance Computing systems: Supercomputers are expected to reach Exascale in 2023 but this will require a significant improvement in terms of energy efficiency. In this domain, power-capping can significant increase the final energy-efficiency by cutting cooling effort and worst-case design margins. A key aspect for an optimal implementation of power capping is the ability to estimate the power consumption of HPC applications before they run on the real system. In this paper we propose a Machine-Learning approach, based on the user and application resource request, to accurately predict the power consumption of typical supercomputer workloads. We demonstrate our method on real production workloads executed on the Eurora supercomputer hosted at CINECA computing center in Bologna and we provide useful insights to apply our technique in other installations.

1 Introduction

Supercomputers peak performance[1] is expected to reach the the ExaFlops (10^{18}) scale in 2023 [20], as revealed by the exponential increase of the worldwide supercomputer installation [16]. A key factor limiting their further growth is the power consumption.

Indeed today's most powerful supercomputer is Tianhe-2 which reaches 33.8 PetaFlops with 17.8 MWatts of power dissipation [15]. Exascale supercomputers built with today's technology would led to an unsustainable power demand (hundreds of MWatts of power) while accordingly to [6] an acceptable range for an Exascale supercomputer is 20MWatts. For this reason, current supercomputer systems must significantly increase their energy efficiency, with a goal of 50GFlops/W. Today "greenest" supercomputers achieve around 9 GFlops/W, thus a wide gap still needs to be closed in order to satisfy Exascale requirements.

The power consumed by a HPC systems is converted into heat therefore, beside the IT power strictly needed for the computation, the additional power

[1] measured as FLOPS (floating point operation per second)

consumption of the cooling infrastructure must be taken into account. The extra infrastructure needed for cooling down the HPC systems has been proved to be a decisively limiting factor for the energy performance; a common approach taken to address this problem is the shift from air cooling to the more efficient liquid cooling. To further reduce the cooling cost HPC systems uses hot water recycling and free-cooling solutions [19]. Indeed when ambient conditions allow it, it is possible to remove heat with direct exchange of heat with the ambient. The amount of heat removed with this approach is proportional to the temperature gradient between the supercomputer outlet temperature (water or air) and the inlet ambient temperature. During cold days the gradient increases, enabling a larger heat portion to be removed without switching on the chillers. At the same time an internal hot-temperature (i.e. hot-water cooling) increases the heat exchanged with the ambient.

A widely used metric for power efficiency is the *PUE index* (Power Usage Effectiveness), i.e. the ratio between the power consumption of the whole data center and the power consumption of the IT equipment alone. Common approaches for the design of the facility as well as for the PUE computation assume average or worst-case ambient parameters (temperature and humidity) as well as peak power consumption. However peak power consumption is rare event, which may not happen for the entire lifetime of the supercomputer. Moreover this static design approach becomes suboptimal when dealing with free-cooling. Indeed as early described in this circumstance the amount of IT power which can be removed without activating the chillers depends on the external temperature and humidity level, and thus varies across daily and night hours and seasons.

Current supercomputers cooling infrastructures are designed to withstand power consumption at the peak performance point. However, the typical supercomputer workload is far below the 100% resource utilization and also the jobs submitted by different users are subject to different computational requirements [31]. Hence, cooling infrastructures are often over-designed. To reduce overheads induced by cooling over-provisioning several works suggest to optimize job dispatching (resource allocation plus scheduling) exploiting non-uniformity in thermal and power evolutions [5].

Hardware heterogeneity as well as dynamic power management have started to be investigated to reduce the energy consumption [17]. The idea is to limit the power consumed by a supercomputer exploiting the power variation which can be found across the computing resources of homogeneous[29] or heterogeneous[18] large-scale systems, in order to create energy and power aware schedulers.

A common approach to limit the amount of power consumed by HPC systems is *power capping*[21], which means forcing a supercomputer not to consume more than a certain amount of power at any given time. Power capping approaches are gaining popularity due the relatively simple implementation and the effectiveness in reducing the total power consumed by a supercomputer. We are especially interested in power capping techniques which do not require any change to the system components, nor to their performance, but only to jobs execution order alone. For these approaches a critical aspect is the possibility to know during the dispatching phase, *before* the actual execution, the amount of power consumed by a job. The key idea of this paper is therefore to provide a predictive model able to estimate with high accuracy the power consumptions of different applications running on a supercomputer. We evaluated our approach on a real supercomputer, with its own peculiar characteristics, but the methodology we employed is general and can be applied to different HPC systems.

The rest of the paper is organized as follows. Section 2 describes in more detail the power capping method and introduces the HPC machine considered in this work. In Section 3 we consider the nature of the applications running on supercomputers and the complications which may arise in a large system where multiple jobs may share the same resources. In Section 4 we present the prediction model and in Section 5 we evaluate the quality of the prediction w.r.t. real historical data. Finally Section 6 draws some conclusions and illustrates future research avenues.

2 Power Capping

Today's most common power budgeting techniques rely on the hardware components capacity to operate at different frequencies and therefore with different power consumptions. The main idea is to limit the computing nodes performance (i.e. through Dynamic Voltage Frequency Scaling, DVFS, the capacity of varying processor clock frequencies) when the total power consumption get closer to the critical threshold [13]. For example, in [26] an Integer Linear Programming (ILP) model is presented to enforce power capping in a HPC cluster. The goal is to combine power aware characteristics with CPU power capping to maximise job throughput of data centres where power is a constraint. Another possibility is the so called "overprovising". Supercomputer components such as CPU, GPU and memory have a vendor-specified Thermal Design Power (TDP) that corresponds to the maximal power required by the subsystem. Currently, maximum power consumption of an HPC system is determined by the sum of the TDP of its subsystems to take into account worst-case scenario where all components work at their TDP level. The ability to constrain the maximum power consumption of the subsystems below the vendor-specified TDP value allows to add more machines while ensuring that the total power consumption of the supercomputer does not exceed its power budget [23]. Today's processing elements can be configured to automatically limit they power consumption to a given budget. However this approach can severely degrade the performance of all the applications which will use the power capped resource, as they will run on a de-facto reduced performance HW.

Another strategy to impose a power constraint is to act on the job execution order alone, without requiring any HW modification nor any change in the operational frequencies of the computing nodes. Actually, this form of power capping may not exclude the HW one, because the HW one can still be required in case of optimistic misprediction. In general HW control can prevent power limit violations at a significant performance cost. Previous works have shown that extending current HPC system job dispatchers with power capping could lead to substantial energy savings without degrading the performance of the supercomputer and the QoS for the users [8, 7]. The approaches studied in these work produce *proactive* schedules: they consider all the jobs which need to be run - and submitted in a job queue by the supercomputer users - and decide the starting time of each one of them in advance, according to a specified objective (i.e. QoS, maximal power savings, etc.). Compared to reactive power capping approaches which cannot prevent dangerous violations of the desired power budget as they are based on a posteriori measurement of the power consumption, proactive approaches can predict which one will be the power consumption of a future schedule and thus can ensure that the power budget is never exceeded.

Typically, job dispatchers need to know the power consumption (or at least an estimate) of each application before deciding a schedule - i.e. the order in which the job are executed. The goal is to guarantee *a priori* that the power constraint will not be violated in any moment (with a certain level of confidence). For this reason the capability of predicting the power consumptions of the jobs which need to be run is extremely important for the optimal implementation of a power capping method, as underlined by several works [22, 30]. Furthermore, a greater prediction accuracy is related to a better performance of a power capped dispatcher (in terms of higher machine utilization and greater energy savings) [12]. Intuitively if we could exactly know the power consumed by each application we could generate optimal schedules and be sure that these schedules will never exceed the power budget; conversely, we may obtain sub-optimal solutions when we deal with imperfect estimates - we may want to be robust and never violate the power constraint (for example, employing a tighter power budget), or we can accept to exceed the power limit from time to time.

A common way to estimate an application energy or power consumption exploits hardware performance counters which monitors the system's components usage during the workload execution [11, 14]. Despite the good accuracy obtained with these models the need to know the performance counters, which should be measured at runtime, clashes with the idea of having power consumption predictions available during the dispatching phase. A model to predict energy and power consumptions is presented in [28]. The authors propose an approach which does not require any application code instrumentation and allows for ahead of time power and energy consumption prediction. The main limit of the described method is that it considers only jobs which occupied entire computational nodes (this is due to the characteristics of the considered supercomputer). This on one hand simplifies the power consumption prediction but on the other hand cannot be directly generalized to different systems where multiple applications can possibly concurrently run on the same node.

Interest in power predictions is not limited to power capping. For example, the authors of [3] use DVFS in order to develop an energy aware scheduler able to reduce energy consumption of supercomputers. For this purpose they introduce a prediction model to forecast power and performance application in case of different execution frequencies. This model relies heavily on precise information about the application executables and requires the user to provide a tag identifying similar jobs. While we think this is an interesting direction, currently users provided information cannot be taken for granted.

In the rest of this section we describe the supercomputer which served as our case study.

2.1 The Eurora Supercomputer

The Eurora supercomputer prototype, developed by Eurotech and Cineca [1] has ranked first in the Green500 list in July 2013, achieving 3.2 GFlops/W on the Linpack Benchmark with a peak power consumption of 30.7 KW. Eurora has been supported by PRACE 2IP project [2] and it serves as testbed for next generation Tier-0 system. Its outstanding energy efficiency is achieved by adopting a direct liquid cooling solution and a heterogeneous architecture with best-in-class general purpose HW components (Intel Xeon E5, Intel Xeon Phi and NVIDIA Kepler K20). For its characteristics Eurora is a perfect vehicle for testing and characterizing next-generation "greener" supercomputers. As described in [4] Eurora has a heterogeneous architecture based on nodes (blades). The system has 64 nodes, each with 2 octa-core CPUs and 2 expansion cards configured to host an accelerator module: currently, 32 nodes host 2 powerful NVidia GPUs, while the remaining ones are equipped with 2 Intel Xeon Phi accelerators. Every node has 16GB of installed RAM memory. A few nodes (external to the rack) allow the interaction between Eurora and the outside world, in particular a login node connects Eurora to the users and runs the job dispatcher (PBS). A key element of the energy efficiency of the supercomputer is a hot liquid cooling system, i.e. the water inside the system can reach up to 50°C.

Jobs are submitted by the users into one of multiple queues, each one characterized by different access requirements and by a different estimated waiting time. Users submit their jobs by specifying 1) the number of required nodes; 2) the number of required cores per node; 3) the number of required GPUs and Xeon Phi per node (never both of them at the same time); 4) the amount of required memory per node; 5) the maximum execution time.

Monitoring Infrastructure Eurora features an integrated and low-overhead monitoring system composed by a set of software daemons and parsing scripts. The SW daemons run periodically (every 5 second) on each node to collect traces of the processing elements (CPUs, GPUs, Xeon Phi) activity by mean of HW performance counters. For each core they gather values from the Performance Monitoring Unit as well as the core temperature sensors, and the time-step counter. In addition, for each CPU it gathers the energy monitoring counters (power unit, core energy, dram energy, package energy) present in the Intel Running Average Power Limit (RAPL) interface. The parsing scripts process off-line the raw log of the performance counters to generate performance metrics (CPI, Load, Temperature, Power, etc.) and relate them with the job running on the node. In addition to the physical information monitoring framework we also gather the information regarding the jobs which executed on the supercomputer;

we consider parameters describing the job, such as the user who submitted it, the requirements (number of core requested, etc.), when it started and when it ended, etc. All the collected data are stored on a database hosted at CINECA.

A critical problem arises from the fact that multiple jobs can run concurrently on the same node: while we can directly measure only the power consumed by the CPUs the jobs can be allocate to single cores. Moreover, we know on which node a job run but we cannot distinguish the exact CPU it used (two CPUs per node). Therefore we cannot directly measure the power consumptions of applications which do not occupy entire nodes. The number of jobs which use only node portions is not negligible at all, since that is actually the majority of jobs which run on Eurora supercomputer[2]. This problem may emerge in many different HPC systems due to the various power measurements methodologies found in current supercomputers[27]. In the following sections we explain how we deal with such a problem.

3 Job Power Profiling

In this section we discuss and empirically validate two extremely important key concepts: 1) the power of a job can be approximated with its average value keeping a good accuracy; 2) a method to compute the power consumed by jobs which required only portion of nodes.

In this work we decided to consider only the power consumption of the CPUs, thus disregarding HW accelerators. We focused on the CPU consumptions because these are the most difficult to deal with, due to the fact multiple jobs can run on the same CPUs. Currently, HW accelerators cannot be shared by multiple jobs. Therefore from now on with “power” we are referring to the CPU power consumption. As already mentioned we must make a distinction between two kinds of problem: jobs which require entire nodes (one or more) and jobs which require only a portion of a node. In the first case we can simply use the power measures we collected; in the latter case we need a method to compute the power consumptions with the data at our disposal. This method is described in Section 3.1.

A HPC application power consumption may vary during its lifetime due to the nature of the applications itself and of the different phases which compose it, but the impact of such variability has not been extensively studied. Conversely, if the power consumption was constant - i.e. if we can associate to each job a precise, single value - the task of a job dispatcher would be greatly simplified[3]. Our idea is to use the *mean* power to represent the power consumption of a job, in the hope that the power variability is relatively low and the power consumption

[2] It is probably due to the fact that Eurora was originally a prototype and only later entered production phase

[3] The proactive job dispatchers aforementioned require to know in advance the job power consumption as a single value; they could theoretically manage the power as a more complex object (i.e. a curve instead of a single value) but we risk to incur in significant performance losses

relatively constant or that when we add all the job power traces the variability of the power of each job is compensated by the others. This mean value is calculated as the average of all power measurements collected during the job lifetime.

In Figure 1 we can see the power consumption profiles of three different jobs, A (Fig 1a), B (Fig 1b) and C (Fig 1c). The first two jobs present a similar profile: their power consumption is quite constant and with a relatively small variability. In particular the power consumption of job B shows very little variability while job A powers have a higher variance - but still quite small compared to the mean value. As job C reveals, this is not always the case: there are also jobs whose power consumption changes more drastically during their lifetime (see the sharp increase in the power consumed by job C). Nevertheless, we can still estimate a job power consumption through its mean (average) value because the jobs with significant power variability are only the minority of all the jobs. The overwhelming majority of jobs show a low standard deviation in their power consumptions.

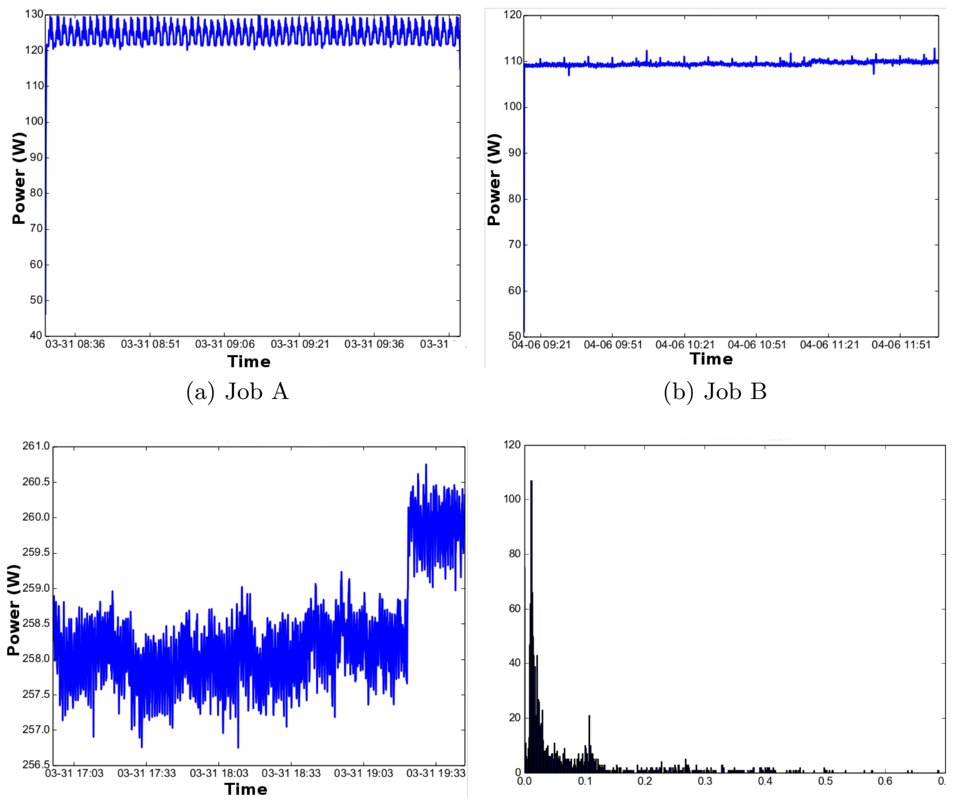
In Fig. 1d finally we see the histogram of the "normalized" standard deviations distribution. For each job we first compute the mean and the standard deviation of all the power measurements then we divide the standard deviation by the mean value to obtain a normalized standard deviation (to let us compare standard deviations of different jobs). We can easily see that for the vast majority of the jobs the standard deviation of the power consumptions is less than 10% of the mean value - and in many cases even less than 5%. This means that the standard deviation is, on average, very small and consequently the power variability is not too big. This is probably a characteristic of HPC jobs, which are generally carefully tuned to avoid big workload changes during the key computation.

This observation allows us to estimate the power consumption with the mean value without losing too much accuracy. It is clear that when associating a single power to a job we are going to lose some information about the real power consumption and therefore commit a certain (small) error. A job dispatcher with power capping is interested in the total power consumption of the system: when we sum all the jobs the errors tend to compensate each other thus the average error we commit is low.

3.1 Power consumption of jobs not using entire nodes

We describe now the technique to estimate the power of jobs running on portions of nodes. As discussed before we do not have a direct measurement of the power consumed by job using only a portion of a node. In this section we present a method to compute a power for these jobs. The main idea behind our approach is that each job consumes an amount of power proportional to its requirements (more specifically the number of requested cores).

An important requirement of our approach is the knowledge of the number of running jobs and "active" cores in each node at every time. Number of active cores means the number of cores that should be used in a node given the number of cores requested by all jobs running on such node. We therefore created



(a) Job A (b) Job B

(c) Job C (d) Standard Deviation Histogram

Fig. 1. Real Power Consumptions of 3 jobs (A, B and C) and standard deviation distribution histogram

these node profiles using the historical job traces. The information regarding the number of cores active on a node at any given time is fundamental for our approach since it allows us to understand the amount of power associated with a job: if a job j runs alone on node i the number of active cores is equal to the number of cores requested by job j and all the power consumption can be attributed to that job. Conversely if more jobs are running concurrently, each job will contribute to only a portion of the whole power (i.e. with 2 jobs sharing a node, using all the node cores, each job can be associated with half the measured power). This reasoning motivates the power we associate to each job as expressed in Equation 1

$$P_j = \overline{P_{ij}} \frac{cr_j}{nca_{ij}} \tag{1}$$

where P_j is the power associated to job j, cr_j is the number of cores required by job j, $\overline{P_{ij}}$ is the average power of the node i on which the job j has run during the job lifespan and nca_{ij} is the weighted average number of cores which were active on node i during the duration of job j. In practice this equations says that the power associated to a job depends on the amount of workload related to the job - i.e. how many of the active cores on the nodes were used by the job.

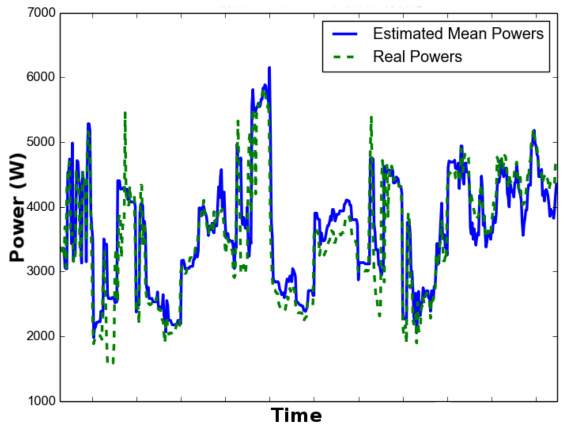
While $\overline{P_{ij}}$ and cr_j are information stored in our database and ready to be used, nca_{ij} needs to be computed. This value tells the average number of cores (related to the number of concurrent jobs) which were active during the lifetime of the job; this number can be computed using the node profiles described previously, which can tell us the number of active cores in any moment of the job duration. More precisely we divide the job lifespan in sub-intervals where the number of active cores is constant and then we compute the average number of cores, weighted by the sub-interval duration. To formalize, given a node i we suppose to have job j, with duration Dur_j, which starts at ST_j and terminates at ET_j. We then have a set of intervals $s \in j$, each one with duration Dur_s; in a sub-interval the number of active cores is nc_s. The weighted number of active cores can be computed as follows:

$$nca_{ij} = \sum_{s \in j} nc_s \frac{Dur_s}{Dur_j} \tag{2}$$

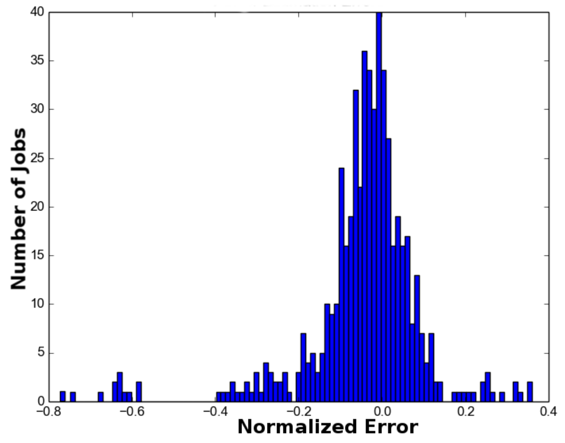
We designed an experiment to to test the accuracy of the approximation method for jobs using node portions. For a given time interval we compare the total real power of the system, computed as the sum of all the node power measurements at each given time (we use time sub-intervals of 5 minutes). Then, for every time considered we also compute the total estimated power, i.e. the sum of the powers of the jobs running at that time; the power of each job is calculated with the method discussed above. A clear limitation in this approach is the fact that we are comparing two aggregate metrics (total real power and total estimated power) and we do not calculate the single job error.

In Figure 2 some results are shown - the trace corresponds to a one-day period. As we can see the results are remarkably good, for example in the period considered we can see that our power estimate has an accuracy around 99%. The results can nevertheless be worse in general, but the accuracy for our whole data set is always between 95% and 96%. Part of the error we observe is not due to the method itself, but actually is caused by the imperfect data in our possession. For example, our algorithm strongly depends on the traces of the jobs which run on Eurora and particularly on the resource requirements declared by the jobs. This can be a source of error because users declare the peak requirement of their application but the average usage may be lower (i.e. asking for 4 cores but actually using just one on average).

The main weak point of our method is its reliance on the resource requirements, which are declared by the users. Hypothetically, users should make truthful declarations, but it turns out - as we discovered through previous analysis - that this is not always the case. This could also be due to the fact that we con-



(a) Computed VS Real Power



(b) Error Histogram

Fig. 2. Comparison between the real aggregated power and the computed aggregated power. Mean Error: 0.011

sider resource requirements as constant during a job lifespan whereas a job may not always employ all the requested resources. Nevertheless, there is a discrepancy between the resource requirements declared by the users and the resource actually used and such discrepancy is going to be a problem for the prediction model. We discuss this issue in Section 4.3.

4 Powers Prediction Model

In the previous sections we have seen that it is possible to describe a job power consumption with a single value - the mean value - and we are able to do that for every job running on the system, whether it occupies an entire node or not. With this information we can now create a prediction model to estimate jobs power consumptions. For this purpose we employed a machine learning approach (ML) which counts on the large amount of historical data in our possession: using the knowledge we have on the applications that run in the past we can learn a model able to predict the consumption of future jobs. We implemented our machine learning models with a Python module called *scikit-learn*[24].

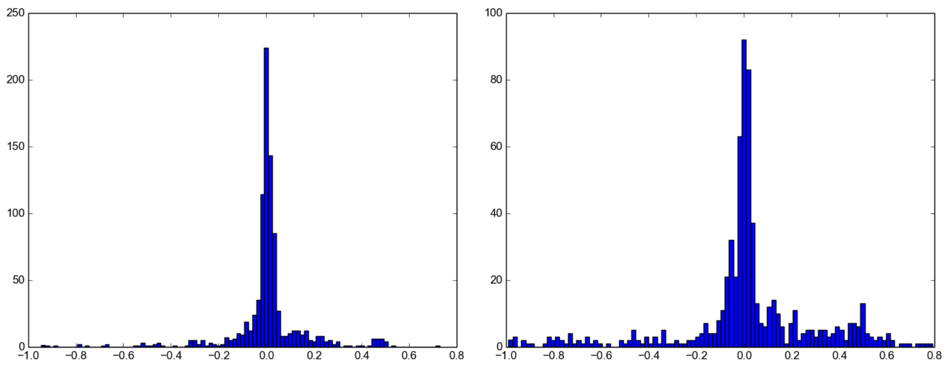
The basic idea of a machine learning predictor is to learn a model which correlates a set of *input features*, or independent variables, with a *target*, or dependent variable. In our particular context we want to correlate the job characteristics (duration, requirements, etc), i.e. the features, with the job power consumptions. This correlation can be learned by the model thanks to the large number of example which constitutes the training set, i.e. the data regarding past jobs, with their characteristics and power consumptions. After the learning phase the model can estimate the powers for new jobs (not seen during the training). A critical element for the success of ML techniques is the availability of large data sets for the training phase; in the Eurora's case we have a data set comprising tens of thousands of jobs (more or less 100k), which is more than enough to obtain good quality predictions.

We developed two different approaches, one to be used with jobs which require entire nodes (Sec. 4.1) and one to be used with jobs occupying only portion of nodes (Sec. 4.2). In this second case, we created multiple prediction models: one for each user with already enough collected data plus a generic one for new users.

4.1 Jobs on entire nodes

In the case of jobs on entire nodes we created a single model which takes into account the following features: user, queue, requested duration, number of requested nodes, number of requested cores, number of requested GPUs, number of requested Xeon Phi[4], amount of requested memory. The ML method we used for this model is called Random Forest Regression [10], which is an evolution of the classical and widely used Decision Tree [25]. Other than Random Forest we tried other supervised regression techniques, using the default implementations provided by sci-kit. We used Generalized Linear Model, Support Vector Machines, Decision Trees and ensemble methods (which combine the predictions of several base estimators) such as Bagging, AdaBoost and Gradient Tree Boosting. We then chose the approach with the better accuracy. The time required to train the model is very low, less than 3 seconds, and the time required to make a prediction is negligible, much less than a second. The training should be performed once with the available historical data and then the model could be updated regularly with the new jobs information collected during the normal execution.

[4] The number of requested HW accelerators is important because GPUs adn Xeon Phi are mounted on computing nodes with different power consumptions, i.e. a job requiring a GPU will necessary run on a CPU consuming more power than those with a Xeon Phi



(a) Test Set A: 943 Jobs (b) Test Set B: 714 Jobs

Fig. 3. Prediction errors histogram for two test sets

A common way to compute the goodness of a prediction model is to split the data set into two subsets: the training set and the test set. With the training set we can learn the prediction model, which is then used to estimate the power consumptions of the jobs in the test set. These estimates are then compared with the real power consumptions of the jobs in the test set, computing the "prediction error" for each job in the test set[5]. The average of these prediction errors then measures the quality of the prediction, with lower values indicating higher quality. The average prediction error computed in this way is around 4%-5%, which guarantees a very accurate prediction. Figure 3a and Figure 3b show the histograms of the prediction errors for two different test sets of jobs on entire nodes. These results were obtained with a training set of around 10k jobs and the test sets are, respectively, of 943 and 714 jobs (test and training sets randomly extracted from the whole data set).

4.2 Jobs on portions of nodes

In the case of jobs using only node portions the prediction turned out to be more difficult than the entire node case and that forced us to try with a different approach. In particular, we chose to create a prediction model for each user, since it is probable that a certain user will submit jobs with a similar pattern. Also in this case the best ML technique has been selected after a preliminary study and the best performance was obtained with Decision Tree Regression [9] - also a Decision Trees based method. The problem with having one predictor per user is that we split our training (and test) set and therefore the dimension of these sets decreases. If the test set dimension decreases too much the learning looses its effectiveness and consequently we cannot make good predictions. This

[5] We actually used a normalized prediction error: $(real\_power - predicted\_power)/real\_power$

problem happens particularly if there are users who submitted few jobs and thus we cannot actually learn a prediction model for them[6]. To solve this issue together with the single user predictors we also have a "general" predictor, devised without splitting the test set and including all users. This generalized predictor is slightly less accurate than the specific ones, but it is an essential component in our prediction mechanism.

The input features of the generalized predictor are the same used for the entire nodes models plus the following: the job name, the number of jobs running on the system at the job start time, the number of cores used on the system at the job start time and the ratio between active cores and total number of cores on the system at job start time. These additional information mainly serve to characterize the supercomputer state and we also added the job name (the application executable) since jobs with similar names - by the same users - usually represent different runs of the same application (and probably are similar in terms of power consumption). The specific (per user) predictors have all the input features of the generalized predictor minus the user name: obviously, since it would be the same for all the jobs in the test set. The training time is very fast also in this case, usually less than 2 seconds for each user predictor.

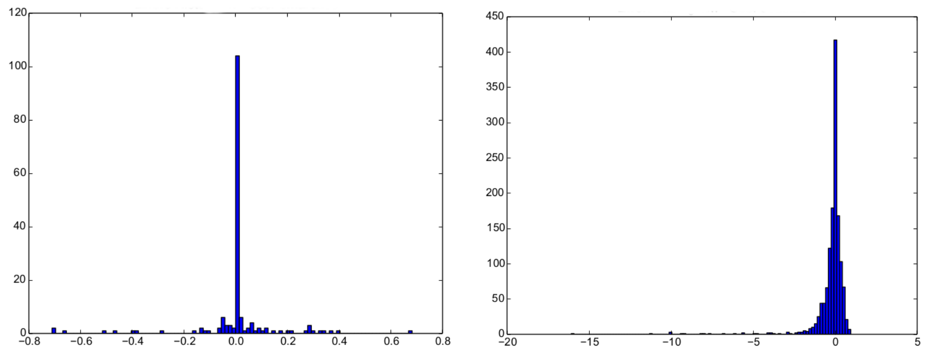
The accuracy of the prediction has been computed as before, i.e. using the average prediction error. Now we have different accuracies for the different user predictors and to obtain an aggregate measure we calculate the mean of all the average prediction errors (one average prediction error user, plus the generalized one). The quality of the prediction is lower than in the case of jobs on entire nodes, with an average error around 15% - for some user the error could be smaller than 3% while for other users it could be up to 35%. In Figure 4 we can see the histograms of the prediction errors for two different users, User A and User B. In the case of User A (Fig. 4a) we see how the accuracy is very good since the errors are very small and centred on zero. Conversely Fig. 4b reveals that for User B the prediction is definitely less precise; it's easy to see that while the majority of jobs are well predicted (small error around zero), a few of them are extremely bad predicted (queue of errors much smaller than -1, on the left of the graph). In the following section we are going to describe the reason of this behaviour.

The predictor models devised for the jobs running on node portions could be also used without modifications also to predict the power of jobs running on entire nodes (jobs on entire nodes are a subset of jobs on node portions). Clearly, the accuracy of the prediction decreases w.r.t. to the dedicated predictor.

4.3 Outliers Management

After collecting the prediction results described previously we investigated them to understand why some predictions were so wrong. The first thing we noticed is that these bad estimated jobs have a small power (compared to the rest of the

[6] This is also a problem for new users to whom we cannot build any prediction model until a sufficient number of jobs are submitted



(a) User A (b) User B

Fig. 4. Prediction error histogram for two different users

jobs). As we see in the next section, these "bad" jobs do not have a great impact on the aggregate prediction precision (that is, when we compare the total real system power consumption and the total predicted power) and could actually be disregarded. Furthermore we claim that these jobs are *outlier*, i.e. jobs not representing a typical Eurora workload. This is due to two factors. First, these jobs show "strange" behaviours: they usually have a very short duration (under 5 minutes and very often even less)[7]. This frequently means that these jobs did not have a correct execution and terminated abruptly, possibly without actually using the requested resources. The second issue is related to the discrepancy between the declared resource requirements and the resources actually used - due to both incorrect estimates by users and varying levels of resource utilization during the job execution. The jobs whose estimates are extremely wrong present a significantly higher level of discrepancy between the declared resource requirements and the resources actually used.

If we discard these outliers from the average prediction error computation, the accuracy increases sharply: the mean error becomes smaller than 4%. This consideration leads us to formulate two simple guidelines to obtain better predictions: 1) it is extremely important that the users declare realistic requirements and should be incentivized to do that; 2) the job monitoring framework has to keep track of those jobs which did not terminate their execution correctly and must be able to distinguish them from the "successful" ones.

5 Results

In this section we present the conclusive results of the methods we introduced in the rest of the paper: estimate of a job power consumption with the mean value

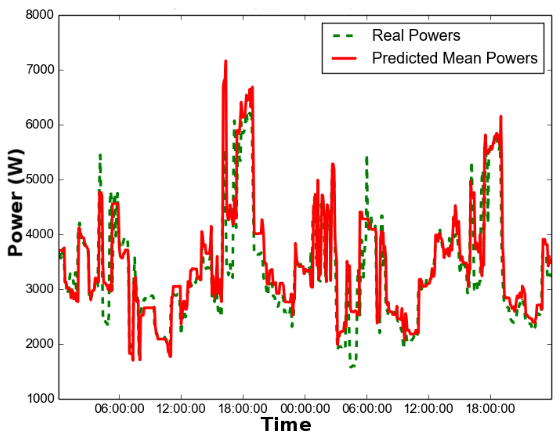
[7] We cannot just delete all jobs with short durations from the train set since in this way we could discard legitimate jobs

of multiple powers measures and prediction of such mean powers with a ML model. We want to know the final error we obtain after having applied all the stages of our method, each of them introducing some inaccuracy. For this purpose we tested our predictions against the real system power consumptions (again, we consider only CPU powers). The experiment set up is the same employed to compare mean power consumptions and real ones, that is we compare the total real power in the system at time t with the sum of all power predictions of job running at time t. The power predictions are obtained with the previously mentioned model and for the sake of simplicity we did not employ the dedicated predictor for the jobs running on entire nodes.

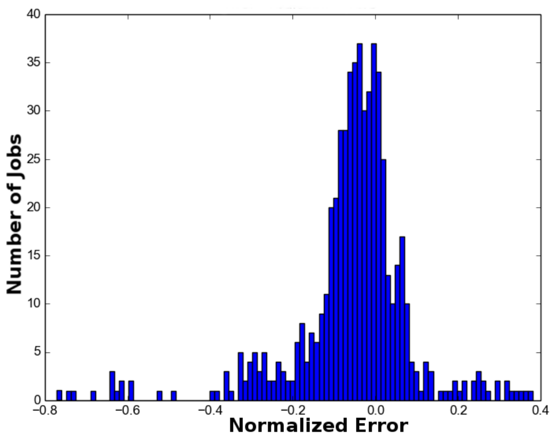
In our accuracy calculations we decided to disregard the outliers, as described in Section 4.3. In Figure 5 we can see some results; the figure corresponds to a two-days period. On the left the predicted trend is compared to the real power trend and on the right we can see the histograms of the prediction errors. As we can see the results are very good, with a mean error smaller than 6%. This is true also if we consider more extended period of times; the average error for the whole test period (a month) is around 8% and 9%.

While the average error is good with must consider the nature of our mispredictions in relation to the power capping. More specifically we can observe *under-prediction* and *over-prediction*. The first one could lead to emergencies (the real power actually exceeds the cap planned in the dispatcher) and the second one leads to machine under-utilization and it is undesirable as well. The concerns about underestimates is that they may lead to power and thermal overshoots. More in detail, when the predicted power is below actual power for a time significantly longer than the thermal time constants of the HPC machine, and the prediction error is above 10%, then hardware power throttling would kick in at run time to prevent temperature overshoots, leading to undesirable performance losses and possible violations of service level agreements. We call these events "critical underestimates". Our error analysis reveals that critical underestimates would happen for less than 2% of the operating time of the machine. For example, in the 2-days period of Figure 5 we registered 37 periods when the predicted power was lower than the real one. While this could seem a high number, the great majority of these under-prediction periods were very short: 90% of the times the under-predictions lasted for less than 2 minutes and 80% of the under-predictions lasted less than 1 minute. The longest under-prediction period lasted for 8 minutes. These values are very short w.r.t. a typical HPC application duration (i.e. a few hours). Even tough these results are very good we can nevertheless take into account under-predictions and there a few ways to cope with them. From one hand, we can back up our dispatcher with a HW power cap mechanism to ensure to never exceed the power budget. Another solution could be to require the dispatcher to respect a power constraint tighter than the real one, thus guaranteeing never to surpass the desired power budget.

An extremely important factor for the quality of the prediction is the availability of information regarding previous jobs. The size of the data set used to train our learning machine models greatly influences the results accuracy. To



(a) Predicted VS Real Power



(b) Error Histogram

Fig. 5. Comparison between the real total power and the predicted total power. Period 2. Mean Error: 0.056

give an idea of the relevance of this point we can look at Figure 6. We have on the x-axis the increasing data set sizes (obtained through random sampling of our original data set) and on the y-axis the corresponding mean normalized error of the prediction. The mean error reported is the aggregated, final one. We can easily see that the prediction accuracy decreases (the error increases) when the data set size shrinks. A good dimension for the data set is around 80k entries (previously observed jobs)[8].

[8] In Eurora's case this corresponds to less than 3 months of observation

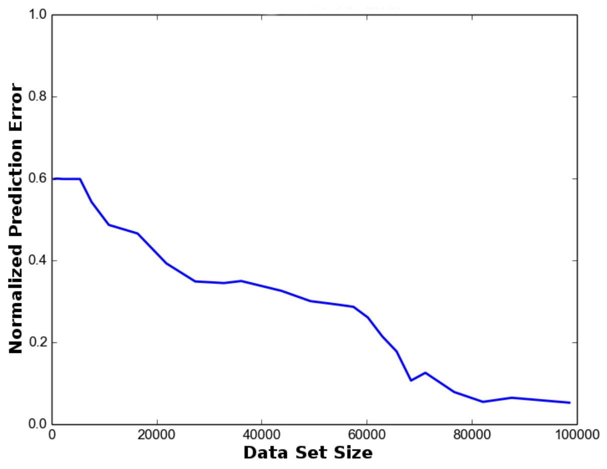


Fig. 6. Data Set size and prediction error

6 Conclusion

In this paper we presented a set of techniques which aim to improve the performance of power capping in HPC systems. Since job dispatchers need to know the power consumptions of the applications in order to create an optimal schedule here we proposed a method to predict such power consumptions. We have done that using techniques borrowed from the Machine Learning field and relying on the big quantity of data we collected on our case-study system - the Eurora supercomputer. The main results we obtained are the following.

First, we can approximate the power consumption of a job with a single value, the average of all the power measurements taken during its lifetime, with only a small loss of precision. Dealing with a single values instead of multiple ones is a great help for the job dispatcher. Then we tackled the problem of co-executing jobs, i.e. applications which run on the same node and using only a portion of the node resources. The problem arised from the mismatch between the minimal allocation unit in the system and the granularity of the power measures collected. We proposed an algorithm to estimated the mean power consumptions of such jobs and proved its efficacy. Finally we devised a method to predict the mean power consumption for every application obtaining a great prediction accuracy and we also provided guidelines to ensure the quality of the predictions.

As future work we plan to integrate the proposed solution in our previous job dispatchers with power capping. In order to do that we are also going to extend our model to consider the HW accelerators in our estimates.

Acknowledgement This work was partially supported by the FP7 ERC Advance project MULTITHERMAN (g.a. 291125). We also want to thank CINECA and Eurotech for granting us the access to their systems.

References

1. Eurora page on the cineca web site. `http://www.cineca.it/en/content/eurora`. Accessed: 2014-04-14.
2. Prace. partnership for advanced computing in europe.
3. A. Auweter, A. Bode, M. Brehm, and et Al. A case study of energy aware scheduling on supermuc. In J. Kunkel, T. Ludwig, and H. Meuer, editors, *Supercomputing*, volume 8488 of *Lecture Notes in Computer Science*, pages 394–409. Springer International Publishing, 2014.
4. A. Bartolini, M. Cacciari, C. Cavazzoni, and et Al. Unveiling eurora - thermal and power characterization of the most energy-efficient supercomputer in the world. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2014*, March 2014.
5. A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller. *IEEE Trans. Parallel Distrib. Syst.*, 24(1):170–183, 2013.
6. K. Bergman, S. Borkar, D. Campbell, and et Al. Exascale computing study: Technology challenges in achieving exascale systems, September 2008.
7. A. Borghesi, F. Collina, M. Lombardi, M. Milano, and L. Benini. Power capping in high performance computing systems. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, pages 524–540, 2015.
8. A. Borghesi, C. Conficoni, M. Lombardi, and A. Bartolini. MS3: A mediterranean-stile job scheduler for supercomputers - do less when it's too hot! In *2015 International Conference on High Performance Computing & Simulation, HPCS 2015, Amsterdam, Netherlands, July 20-24, 2015*, pages 88–95, 2015.
9. L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.
10. Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.
11. GL.T. Chetsa, L. Lefevre, J. Pierson, and et Al. Exploiting performance counters to predict and improve energy performance of hpc systems. *Future Generation Computer Systems*, 36:287–298, 2014.
12. J. Choi, S. Govindan, B. Urgaonkar, and et Al. Profiling, prediction, and capping of power consumption in consolidated environments. In *Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008. MASCOTS 2008. IEEE International Symposium on*, pages 1–10. IEEE, 2008.
13. R. Cochran, C. Hankendi, A. K Coskun, and S. Reda. Pack & cap: adaptive dvfs and thread packing under power caps. In *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*, pages 175–185. ACM, 2011.
14. G. Contreras and M. Martonosi. Power prediction for intel xscale®processors using performance monitoring unit events. In *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, ISLPED '05, pages 221–226, New York, NY, USA, 2005. ACM.
15. J. J. Dongarra. Visit to the national university for defense technology changsha, china. Technical report, University of Tennessee, June 2013.
16. J. J. Dongarra, H. W. Meuer, and E. Strohmaier. 29th top500 Supercomputer Sites. Technical report, Top500.org, November 1994.

17. W. Feng and K. Cameron. The Green500 List: Encouraging Sustainable Supercomputing. *IEEE Computer*, 40(12), December 2007.
18. F. Fraternali, A. Bartolini, C. Cavazzoni, and et Al. Quantifying the impact of variability on the energy efficiency for a next-generation ultra-green supercomputer. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, pages 295–298, New York, NY, USA, 2014. ACM.
19. Jungsoo K., M. Ruggiero, and D. Atienza. Free cooling-aware dynamic power management for green datacenters. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 140–146, July 2012.
20. Peter Kogge and David R. Resnick. *Yearly update: exascale projections for 2013.* Oct 2013.
21. C. Lefurgy, X. Wang, and M. Ware. Power capping: a prelude to power shifting. *Cluster Computing*, 11(2):183–195, 2008.
22. S. Pakin, C. Storlie, M. Lang, and et Al. Power usage of production supercomputers and production workloads. *Concurr. Comput. : Pract. Exper.*, 28(2):274–290, February 2016.
23. T. Patki, D.K. Lowenthal, and et Al. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ICS '13, pages 173–182, New York, NY, USA, 2013. ACM.
24. F. Pedregosa, G. Varoquaux, A. Gramfort, and et Al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
25. J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986.
26. O. Sarood, A. Langer, A. Gupta, and L. Kale. Maximizing throughput of overprovisioned hpc data centers under a strict power budget. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 807–818, Piscataway, NJ, USA, 2014. IEEE Press.
27. T.R.W. Scogland, C.P. Steffen, T. Wilde, and et Al. A power-measurement methodology for large-scale, high-performance computing. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ICPE '14, pages 149–159, New York, NY, USA, 2014. ACM.
28. H. Shoukourian, T. Wilde, A. Auweter, and A. Bode. Predicting the energy and power consumption of strong and weak scaling hpc applications. *Supercomputing frontiers and innovations*, 1(2):20–41, 2014.
29. H. Shoukourian, T. Wilde, A. Auweter, and A. Bode. Power variation aware configuration adviser for scalable hpc schedulers. In *High Performance Computing Simulation (HPCS), 2015 International Conference on*, pages 71–79, July 2015.
30. C. Storlie, J. Sexton, S. Pakin, and et Al. Modeling and predicting power consumption of high performance computing jobs. *arXiv preprint arXiv:1412.5247*, 2014.
31. H. You and H. Zhang. Comprehensive workload analysis and modeling of a petascale supercomputer. In Walfredo Cirne, Narayan Desai, Eitan Frachtenberg, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 7698 of *Lecture Notes in Computer Science*, pages 253–271. Springer Berlin Heidelberg, 2013.