Adaptive Resource and Job Management for Limited Power Consumption

Yiannis Georgiou, David Glesser, Denis Trystram

Adaptive Resource and Job Management for limited power consumption

Yiannis Georgiou
BULL
Yiannis.Georgiou@bull.net

David Glesser
BULL
and Univ. Grenoble-Alpes
David.Glesser@bull.net

Denis Trystram
Univ. Grenoble-Alpes
Institut Universitaire de France
trystram@imag.fr

***Abstract*—The last decades have been characterized by an ever growing requirement in terms of computing and storage resources. This tendency has recently put the pressure on the ability to efficiently manage the power required to operate the huge amount of electrical components associated with state-of-the-art computing and data centers. The power consumption of a supercomputer needs to be adjusted based on varying power budget or electricity availabilities. As a consequence, Resource and Job Management Systems have to be adequately adapted in order to efficiently schedule jobs with optimized performance while limiting power usage whenever needed.**

This paper introduces a new scheduling strategy that provides the capability to autonomously adapt the executed workload to a limited power budget. The originality of this approach relies upon a combination of DVFS (Dynamic Voltage and Frequency Scaling) and node shutdown techniques for power reductions. It is implemented into the widely used resource and job management system SLURM. Finally, it is validated through large scale emulations using real production workload traces of the petaflopic supercomputer *Curie*.

I. Introduction

Energy consumption has become one of the most crucial issues in the evolution of High Performance Computing systems. The increase in computation performance of such platforms has come with an even greater increase in energy consumption, turning energy an undisputed barrier towards the exascale challenge. The power demand of such HPC systems has made electricity one of the largest cost component for the lifetime of these systems and it is important, almost critical, to improve electrical system utilization and operation profiles. Since the operational cost of a computing center for HPC systems is directly related to the energy consumption, such centers have additional motivations to regulate the energy usage. However, controlling the energy for a certain duration demands novel approaches to adjust the instantaneous power on a daily basis.

Research efforts upon all different abstraction layers of Computer Science, from hardware, to middleware up to applications, strive to improve energy consumption and provide efficient power management. The advances at the hardware layer need to be followed by evolutions on systems software and middleware in order to provide efficient results. Various techniques such as *speed scaling* (DVFS) and node shutdown have been broadly used for energy reductions [1,2]. Nevertheless, the regulation of power needs to pass from a centralized system middleware able to monitor power consumption and adjust it through coordinated actions across the whole infrastructure. The most adequate software for this type of coordination is the Resource and Job Management System (RJMS) which plays a crucial role in the HPC software stack since it is aware of both the hardware components state and information, along with details upon the users workloads and the executed applications.

In this work we propose a powercapping mechanism implemented upon a Resource and Job Management System based on a combination of offline and online job scheduling approaches, making use of both DVFS and node shutdown power-reduction techniques. We study a generic model which shows that in some cases, the best solution is to mix both techniques. As a consequence, the RJMS has to be adequately adapted in order to efficiently schedule the jobs with optimized performance while limiting power usage whenever needed. More precisely, the main contribution is the introduction of a new power consumption adaptive scheduling strategy that provides the capability to autonomously adapt the executed workload to the available or planned power budget. We have studied the impact of DVFS on several actual applications along with effects of using grouped shut-down of nodes and considered them in our model. The new scheduling algorithms have been implemented upon the widely used open source workload manager SLURM. As our aim is to integrate this mechanism into large-scale HPC supercomputers such as *Curie*, we have validated our model, algorithms and implementations using large-scale experimentations based upon real production workload traces of *Curie* petaflopic supercomputer.

The content of the paper is as follows. We start by an overview of the most relevant approaches used so far for reducing the energy consumption and powercapping in Section II. Section III presents our model for determining the best combination between DVFS and shutdown for power limitations. The scheduling algorithm is presented in Section IV while section VI provides the analysis of its implementation. Section V presents the adaptation of the mechanisms using real supercomputer's characteristics. We evaluate our algorithms and report the results of our experiments in Section VII. Finally, conclusions and perspectives are discussed in Section VIII.

II. Related Works

A. Controling the power

Dynamic Voltage and Frequency Scaling (*DVFS* in short) is the most popular mechanism used so far for controlling the power in computing systems and as a consequence, reduce the energy. There exist a lot of works oriented toward theoretical results on speed-scaling (for instance continuous speeds). We are targeting here more realistic issues. The first series of

papers intended to determine the right frequency. Energy-centric DVFS controlling method was proposed in [1] for single CPU multi-core platforms. The idea was to monitor the traffic of data from the cores to the memory and to update the DVFS accordingly (i.e. reduce it if the traffic is high or expand if it is low). This was extended in [3] for more general platforms. Schöne and Hackenberg [4] used register measurements for determining the frequency. Kimura et al. [5] provided also a new mode of control of DVFS through a code-instrumented DVFS control. Then, Gandhi et al. [6] considered a mechanism that switch between the maximum DVFS to the idle state, in order to minimize the energy consumption. DVFS has also been studied to predict its impact on the whole system. Rountree et al. [7] developed a performance prediction model outperforming previous models. Etinski et al. [8] studied how to improve the trade-off energy versus completion time on applications. Freeh et al. [9] provided a huge number of experiments for measuring the Energy over Time. An interesting feature was studied in the case where a node is removed from the system (moldable jobs).

Another complementary mechanism consists in switching off some nodes (also called *shutdown*). Lawson et al. [2] proposed an opportunistic shutdown mechanism, which switches off a node after a significant idle period. Aikema et al. [10] studied the energy from the view point of a cost function. Here, a node which becomes idle is considered as a zero-cost in term of energy. Under the assumption of under-loaded cluster, Hikita et al. [11] presented a batch scheduler that minimizes the number of active nodes while keeping the same performances. In [12], Demaine et al. took into account both the cost of energy and of switching (off/on) the processors. They proposed a theoretical algorithm that minimizes the number of such switches.

Despite the two first mechanisms, other options have been studied including network frequency scaling [13] or temperature-aware scheduling [14]. An incentive mechanism for reducing energy has also been proposed in [15].

B. Powercapping

In the following of this paper, we are focusing on powercap as the main topic. Some papers are considering powercapping at the node level, for instance [16] used a new feature available in Intel processors to achieve a local powercap and [17] packed threads together in order to tune the DVFS. Fan et al. defined in [18] a methodology to reduce the global cost of data-centers by buying more processors and capping their power consumption. In the context of cloud computing, Geronimo et al. proposed a virtual machine manager that can use DVFS, update the virtual machine resources, migrate them and shutdownopportunistically some processors [19]. Powercap has been studied by Etinski et al. in a series of papers [20]–[22] where DVFS is the only mechanism used to achieve powercapping while keeping good performances. We consider here a more sophisticated mechanism including also shutdown.

To the best of our knowledge, there is no similar work which consider DVFS and shutdown simultaneously in order to adapt the power consumption of a cluster. Pierson and Casanova proposed a theoretical approach based on mixed Integer Linear Programs restricted to a single application [23]. We propose here a generic mechanism which selects the best strategy among DVFS, shutdown or both together.

III. Energy and Power analysis

In this section, we describe a new model that enables to determine when to switch off nodes and to determine the right frequency.

A. Tradeoff Switch-off between DVFS

Let us define W as the maximum amount of computations that could be performed during a given period of time T:

$$W = T.(\frac{N - N_{off} - N_{dvfs}}{1} + \frac{N_{dvfs}}{deg_{min}}) \qquad \text{(C1)}$$

Where N is the total number of nodes; N_{off} is the total number of nodes which are switched off and N_{dvfs} is the total number of nodes whose frequency has been decreased.

deg_{min} represents the percentage of *degradation* of the completion time at the minimum frequency compared to the maximum one. The justification to take deg_{min} at the minimum frequency is to consider the maximum possible impact on applications. This choice will be discussed in Section VI.

Obviously,

$$N_{dvfs} + N_{off} \leq N. \qquad \text{(C2)}$$

Without loss of generality, T will be set to 1.

The consumed power should be lower than the powercap:

$$\begin{array}{l} N_{off}.P_{off} + N_{dvfs}.P_{min} + \\ (N - N_{off} - N_{dvfs}).P_{max} \leq P \end{array} \qquad \text{(C3)}$$

Where P_{off}, P_{min} and P_{max} are respectively the power consumed by a switched-off node, by a node in the lowest frequency and by a node in the maximum frequency. P is the powercap, *i.e.* the global available power at this moment.

The previous constraints C1 and C3 correspond respectively a plane and an half-space in the 3 dimensional space (W, N_{off} and N_{dvfs}). We are looking for points that maximize W within the previous constraints. The intersection of the two previous surfaces is a segment. We then consider the point that maximizes W on this segment. Then, there are four cases to distinguish:

1) There is only some switched-off nodes (the best point is on located on the plane (W, N_{off}).
2) There is only use of DVFS on some nodes (the best point is on located on the plane (W, N_{dvfs}).
3) Both previous options lead to the same maximum computational load (all the points of the segment are eligible)
4) The powercap is too low and both mechanisms should be used to reach the maximal W (the best point is at the intersection of the segment and the constraint C2).

The value for the two first cases can be easily computed thanks to the following formulas:

$$\begin{cases} N_{off} = \frac{P - N.P_{max}}{P_{off} - P_{max}} \\ N_{dvfs} = 0 \end{cases} \quad \text{or} \quad \begin{cases} N_{off} = 0 \\ N_{dvfs} = \frac{P - N.P_{max}}{P_{min} - P_{max}} \end{cases}$$

Let W_{dvfs} be available computational load available using only the DVFS mechanism (similarly W_{off} for the switch off mechanism). DVFS is better to use when $W_{dvfs} > W_{off}$. Which is equivalent to $\rho > 0$, where $\rho = 1 - \frac{1}{deg_{min}} - \frac{P_{max} - P_{dvfs}}{P_{max} - P_{off}}$. In the third case, we choose

either one or the other mechanisms. Thus, for the three first cases, it is easy to determine which mechanism to use depending on ρ. Let us focus on the last case when the powercap is too low. How low this powercap has to be to reach this limit? The powercap is too low when $P < N.P_{off}$ or $P < N.P_{min}$. The first expression can not happen practically since the powercap will be less than the clusters completely switched-off. Let us define $P = \lambda N P_{max}$, where λ is the powercap normalized by the maximum power consumption of the cluster. The second expression becomes $\lambda < \frac{P_{min}}{P_{max}}$, which means that the powercap can not be less than $\frac{P_{min}}{P_{max}}$ if DVFS is the only mechanism used to control power. In this case, the best choice for N_{off} and N_{dvfs} is:

$$\begin{cases} N_{off} = N - N_{dvfs} \\ N_{dvfs} = \frac{P - N.P_{off}}{P_{min} - P_{off}} \end{cases}$$

We are now able determine which mechanism to use depending on the job, the cluster and the powercap. In the following section we will present an optimization of the switch-off mechanism. Then we will discuss the algorithm from the the described model.

B. Power Bonus when switching off hardware components

In HPC clusters, there are several hardware levels from a power consumption point of view. A level is defined as a group of different hardware components that can be switched-off simultaneously.

The configuration of which hardware components participate at each level depends on the architecture of the cluster. For instance, if the architecture is such that nodes are grouped in order to mutualize the first layer of administration and interconnection networks switches, nodes belonging to a same group can be powered off along with their respective switches without preventing the correct usage of the remaining groups. In this example, such a group will compose a particular power level. The extra power consumption gained by the network switches when powered off is then called a 'power bonus'. This method allows us to reduce even more the power consumed by a cluster when disabling part of its compute power.

In modern architectures, typical HPC clusters will have different 'power bonus' related to the different levels of components aggregation. Correctly selecting the computing elements to switch-off when coping with a power constraint will thus enable to sum up the different bonus at the different levels and maximize the power available to the active compute elements and their hardware dependencies. The lowest level considered in the our model is the multicore node. No actual power bonus is currently provisioned at this level. Individual cores and sockets can not be switched-off individually, hence they cannot comprise a lower level on their own.

In Section VI-A we provide more precise details for the 'power bonus' related to the Curie cluster architecture.

IV. Scheduling under power control

A. Preliminaries on scheduling features

From a high level point of view, scheduling in a Resource and Job Management System can be decomposed into two successive phases: first, the jobs should be selected after prioritization among the group of pending jobs, then, the resources should be selected upon which the job will be dispatched. In more detail, the first phase may involve various mechanisms to select the next job to be treated. For instance, the usual backfilling [24] may be enriched with multifactor priorities such as job age and job size or even more sophisticated features like fair-sharing and preemption. The second phase is related to the actual allocation of resources according to their characteristics such as internal node topology, network topology, usage of accelerators. The proposed powercapping algorithm takes place during this second phase of scheduling. One of the main building blocks of this algorithm relies on the fact that power is treated as a new type of resources characteristics. According to its state (PowerDown, Idle, Busy, etc.), the resource will consume a different amount of power. At any moment, the RJMS keeping the state of each resource internally can deduce the power consumption of the whole cluster. The characteristic of power can be related to any different component of the cluster but for the sake of clarity, we consider here the power of a whole node.

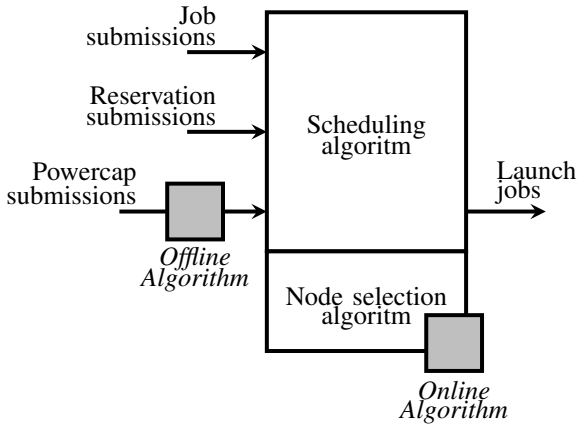


Fig. 1: SLURM architecture. In grey, modified part.

The calculation of the power consumption of the cluster is simply obtained by summing up the power consumptions of each node. For instance, the nodes that are executing jobs will be counted with the maximum power consumption (except in cases of DVFS usage); the nodes that are kept idle will be counted with the minimum power consumption and those that have been switched-off are counted with no power consumption (it could be non-zero in case where the BMC (Baseboard Management Controller) is still on). The algorithm goes one step further by considering the setting of the different CPU frequencies as different power states. Hence, the power consumption of each node will change depending on the CPU frequency at which the job is running.

The power values related to the state of each node can either be measured or be given by the constructor (this information can be configured and kept internally into the RJMS).

B. Scheduling algorithm for powercapping

The powercapping algorithm that we propose is composed of two successive parts: A first offline part where the decisions for power management are taken in advance (to better prepare future actions) followed by an online part where the power distribution and management take place.

The algorithm is activated as soon as a powercap reservation is provided. This powercap value can be either set for *now* (i.e. the moment when the command is launched) with

no time restriction/limitation or as a reservation for a certain time window in the future. When a new job arrives during the allocation phase, the algorithm examines if there is any powercap limit for the time being or if the job may overlap with any future reservation of power at some point in the future. If any of these cases holds, the power consumption of the cluster is computed by considering as busy the nodes that the job will use. Then, the different values of the a-posteriori power consumption of the cluster are examined by measuring the power with all the different CPU frequencies where the allocated nodes may run. If there is no CPU frequency to allow the future power consumption of the cluster to be less than the power budget then the job remains pending. In the opposite case, the job is executed at the maximum CPU frequency that allows the job to be executed keeping the cluster in the power budget.

Input: The user creates a powercap reservation, indicating the interval time and the value of the powercap (P).
if $P < N.P_{min}$ **then**
  $N_{dvfs} = \frac{P - N.P_{off}}{P_{min} - P_{off}}$
  $N_{off} = N - N_{dvfs}$
  Make a switch-off reservation of N_{off} nodes during the powercap.
else
  $\rho = 1 - \frac{1}{deg_{min}} - \frac{P_{max} - P_{dvfs}}{P_{max} - P_{off}}$
  if $\rho <= 0$ **then**
    $N_{off} = \frac{P - N.P_{max}}{P_{off} - P_{max}}$
    Make a switch-off reservation of N_{off} nodes during the powercap.
  end
end

Algorithm 1: Offline algorithm to control the nodes switch-off reservations.

One of the important parts of the algorithm is the selection of the CPU frequency. Selecting a value close to the maximum will make the power consumption of the cluster to increase faster (some nodes will have to be kept idle) producing starvation of following jobs and dropping the utilization of the system. Considering that jobs may run at a lower CPU frequency (which means that nodes will consume less power) gives us extra flexibility for scheduling more jobs. However, since only one job is treated at a time and we cannot know how many jobs will follow, we need to select the best possible value of CPU frequency whenever the powercapping is activated. The optimal CPU frequency is the maximum allowed frequency that all idle nodes could run such that the total power consumption of the cluster remain within the powercap value. Since the number of idle nodes may change, the optimal CPU frequency may also change from one job to another.

The adequate energy saving mechanism is chosen in the offline step. System administrators can force one or another mechanism. We defined three policies *SHUT*, *DVFS* and *MIX*. *SHUT* means that the system will have the possibility to switch-off nodes and keep others in an idle state if needed. *DVFS* policy means that the system will have the possibility to oblige jobs to be executed at lower CPU frequencies. Finally, *MIX* is a mixed *DVFS* and *SHUT* strategy, which considers both possibilities of saving power. In *DVFS*, *SHUT* or *MIX* modes, the system will decide which mechanism is the most suitable one using the equations introduced in Section III-A.

Input: The job trying to be scheduled.
job.DVFS = highest possible DVFS
while $currentPower + N_{job.DVFS} * job.requiredNodes > P$ **do**
  if $job.DVFS == minimumDVFSpossible$ **then**
    return *Impossible to schedule the job now.*
  end
  job.DVFS = a slower value of job.DVFS
end

Algorithm 2: Simplified online algorithm to control the DVFS of jobs.

If the powercap value is set for *now* then there could be a problematic scenario where the cluster is currently above the powercap. In this case, by default, no extreme actions are taken with the running jobs. This means that no additional jobs will be scheduled and the scheduler will wait until some jobs are completed to eventually have the power consumption of the cluster dropped to a value lower than the powercap. However, we argue that the above default way may not be accepted by some sites that may want to have extreme actions when the powercap value is set. In this case, the necessary number of jobs will be killed until the power consumption of the cluster drops instantaneously.

V. Implementation upon SLURM

The above scheduling algorithm has been implemented upon the open-source resource and job management system SLURM [25]. As of the June 2014 Top 500 computer list [1], SLURM was performing workload management on six of the ten most powerful computers in the world including the number 1 system, Tianhe-2 with 3,120,000 computing cores.

In a nutshell, SLURM is designed as a client-server distributed application: a centralized server daemon, also known as the controller, communicates with a client daemon running on each computing node. Users can request the controller for resources to execute interactive or batch applications, referred as jobs. The controller dispatches the jobs on the available resources, whether full nodes or partial nodes, according to a configurable set of rules.

The power awareness of SLURM has recently been enhanced by introducing the capability to regularly capture the instantaneous consumed power of nodes [26]. Coupled with the introduction of a speed scaling logic, enabling to modify the frequencies of the cores involved during the execution, this new feature helps to identify the behavior of applications in terms of power consumption when varying the frequency.

To achieve the targeted goal of powercapping, a new parameter called PowerCap is added to the controller's set of states. It represents the allowed power budget of the cluster in watts. Also, SLURM reservation characteristics have been extended by a new Watts parameter in order to specify a particular amount of power reserved for a specific time slot.

To compute the maximum power amount required to operate a cluster, new parameters are associated to the compute

[1] http://www.top500.org/list/2014/06/

nodes to provide the different maximum amounts of watts consumed. Thus, IdleWatts, MaxWatts, DownWatts will respectively correspond to the amounts of watts required to operate a node in idle, fully used and down states. The down state corresponds to the state the controller uses to characterize a node not being currently accessible within SLURM (e.g. in the case of node shutdown). Furthermore, other parameters such as CpuFreqXWatts may be used to characterize a node that its CPUs runs at a specific X Frequency. While computing the instantaneous maximum amount of power of the cluster, the controller will use the known states of the nodes in order to sum up the individual maximum amounts of watts and produce a global power value for the whole cluster.

The choice of powercap scheduling mode (*SHUT*, *DVFS* or *MIX*) has been implemented as a configurable SchedulerParameter option and can be dynamically altered by the administrator without restarting SLURM services. The offline part of the scheduling algorithm is triggered only in the case of powercap reservations and has the ability to reserve the shutdown of nodes. In our context, the goal is to regroup the shutdown of contiguous nodes in order to benefit of power bonus possibilities as described in Section III. Hence, we coupled this feature in the offline scheduling part and the shutdown of nodes is triggered through a specific type of reservations in SLURM.

The online part is implemented in the central part of SLURM scheduling mechanism upon the controller. When evaluating the impact of the start of a pending job, the controller will temporarily alter the states of the candidate nodes, compute the resultant consumption and compare it to the defined and planned powercap. In case of *DVFS* or *MIX* scheduling mode, the evaluated job is controlled for all different CPU-Frequencies and it stays pending only if the estimated power consumption with the lower permitted CPU Frequency is larger than the power envelope it may use. The target CPU-Frequency is selected based on the Algorithm 1. Since the *DVFS* is actually altered by the controller during the online scheduling phase, the walltime of the job needs to be adapted respectively. Based on similar studies [20], we consider that the walltime should be increased up to 60% for the minimum CPU Frequency, while intermediate values of walltimes are linearly interpolated. Note that the current code does not make any difference in power requirements whether nodes are fully or partially used. The evaluation of new jobs only filling partially used nodes will always pass the powercapping criteria as no extra power will be required. As a result, the scheduler will tend to fill the compute nodes up to the targeted power budget.

VI. Adapting powercapping logic for Curie

The activation of the adaptive power control of SLURM for a certain infrastructure requires an initial configuration where the maximum power consumptions of each implicated components, along with other important parameters are defined. In this section we present the study made for the adaptation of SLURM powercapping logic for Curie supercomputer.

A. Details on Curie

Curie[2] is owned by GENCI[3], it is the first french Tier-0 system opened to scientists through the participation into the PRACE[4] research infrastructure. Since its upgrade in February 2012, Curie consists of 280 Bullx B chassis housing 5,040 Bullx B510 nodes, each with two 8-core Intel Sandy Bridge processors for a total of 80640 cores. Curie was ranked 26th among the 500 most powerful supercomputers on June's 2014 Top500 list[5].

Configuration details and workload traces of Curie have been extracted at some points of its early lifetime and are used in this study to specify hardware characteristics and evaluate the behavior of the proposed mechanisms.

We have seen in Section III-B that the architecture of an HPC cluster plays an important role when considering which nodes to switch-off in order to maximize the 'power bonus'. In Curie, we distinguish 4 hierarchical levels that can be switched-off. Table 2 gives the power consumption of each level.

- The first one is the node level. A node is composed of 2 sockets and each socket contains 8 processors. When a node is powered-off the BMC is kept active so that the node can be powered back on through the network. This explains the consumption of 14W in table 2 when a node is down.
- The second level is the chassis level. Each Chassis contains 18 nodes and the power bonus is composed by global cooling fans, network switches installations such as Ethernet and Infiniband switches, optical cables, network ports. These hardware components consume an extra amount of power of 248 Watts with a power bonus of 500 Watts as we see in 2.
- Rack level is the third one. It is composed of 5 chassis, which contain fans and a cold door related to the liquid cooling equipment. The power consumption of these components is 900 Watts and the bonus when switching off a complete rack will be 3400 Watts.
- Finally, the last level is the whole cluster, which is composed of 56 racks.

Level	Power consumption	Power bonus	Acummulated Power
Node (down)	14 W	-	-
Node (max)	358 W	-	344 W
Chassis (18 nodes)	248 W	248+18*14= **500 W**	344*18+500= **6692 W**
Rack (5 chassis)	900 W	900+500*5= **3400 W**	6692*5+900= **34360 W**

Fig. 2: Power consumption and the possible saved watts when various levels of the cluster are switched-off.

The power bonus values of table 2 imply that if we switch off a complete chassis this will allow us to completely use at least 1 extra node where-as if we switch-off a complete rack this will allow us to use at least 9 extra nodes. For example, if our model needs a power reduction of 6600 Watts and we

[2]http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm
[3]http://www.genci.fr/en/our-computers
[4]http://www.prace-project.eu/
[5]http://www.top500.org

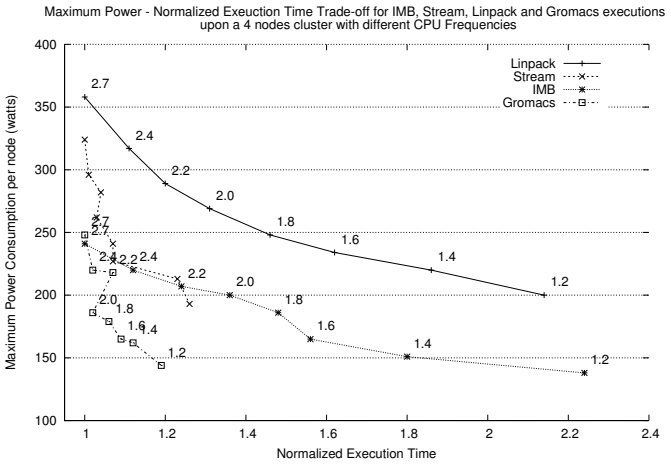


Fig. 3: Maximum Power - Execution Time Tradeoffs for Linpack, Stream, IMB and Gromacs benchmarks at different CPU frequencies

consider that when a node is powered-on it will be used with the maximum consumption of 358 Watts then we need to switch-off 20 single nodes to reduce the power for 6880 Watts (=344*20). However if we make sure that we power off the 18 nodes of a complete chassis this will allow us to make use of the bonus and reduce the power for 6692 Watts which is even more than what we actually need. This allows us to use 2 extra nodes. In order to take advantage of the power bonus and keep more nodes powered-on, we need to prepare an efficient grouping of nodes to switch-off. Hence that is why the choice of which nodes will be switched off takes place during the offline part of the algorithm.

B. Control and Measure power

The power consumption of the different states of a node may be given by the constructor or calculated through experimentations. In our case we perform experiments of various known benchmarks using the exact same models of computing nodes and hardware components that are used on Curie.

We have run three different widely used benchmarks and one application to measure the characteristics of Curie cluster. We have chosen a first benchmark to stress all computing resources (Linpack [27]), a second one targeting memory (Stream [28]), one focused on network (IMB [29]) and the last one is a widely used application for molecular dynamics simulation (GROMACS [30]). Measurements have been done through the IPMI[6] interface of SLURM power profiling mechanisms which have been shown to provide precise results for the consumption at the node level [26]. DVFS is a way to obtain a trade-off between power and completion time. It has been widely studied in the literature (see Section II). Figure 3 gives the evolution of the completion time and the maximum watts consumed at different DVFS values.

Table 4 presents the maximum power consumption observed on a node for each DVFS value based upon the results of all 4 applications. We have also added the observed power consumption of switched-off and idle states. These measurements set the maximum Watts that a node can consume. There is huge gap between switched-off and idle nodes. Both of them do not produce computational power but a switched-off node consumes one order of magnitude less power.

Node state	Maximum power consumption
Switch-off	14 W
Idle	117 W
DVFS 1.2 GHz	193 W
DVFS 1.4 GHz	213 W
DVFS 1.6 GHz	234 W
DVFS 1.8 GHz	248 W
DVFS 2.0 GHz	269 W
DVFS 2.2 GHz	289 W
DVFS 2.4 GHz	317 W
DVFS 2.7 GHz	358 W

Fig. 4: Table of the maximum power consumption of a Curie node in different states.

Also, from these benchmarks we compute the performance impact of DVFS. For simplicity, only the maximum and minimum DVFS frequencies are taken into account. Thus, in the following the *degradation* of performance is between the maximum and the minimum DVFS values. As seen in Section II, the *degradation* of performances has already been studied. In [9], the authors measured the *degradation* for the NAS benchmark, the SPEC float and integer benchmarks. A *degradation* of 163% is assumed to be a good approximation [20]. Figure 5 summarizes the data obtained on Curie for various benchmarks. Common values of *degradation* clearly show that shutdown is the best mechanism to use, at least for reasonable values of powercap.

Benchmark	deg_{min}	ρ	Best mechanism
NA	2.27	0	-
linpack	2.14	-0.027	Switch-off
IMB	2.13	-0.029	Switch-off
SPEC Float [9]	1.89	-0.088	Switch-off
SPEC Integer [9]	1.74	-0.134	Switch-off
Common value [20]	1.63	-0.174	Switch-off
NAS suite [9]	1.5	-0.225	Switch-off
STREAM	1.26	-0.350	Switch-off
GROMACS	1.16	-0.422	Switch-off

Fig. 5: Comparison between DVFS and switch-off in Curie for various benchmarks.

In our context, the *SHUT* policy appears to be the best one. Hence the offline algorithm would never mix *DVFS* and *SHUT* modes. However, we observe in the benchmarks' results that, unlike the power/performance trade-off, the energy/performance trade-off is not monotonic. The most optimal points are between 2.7 GHz and 2.0 GHz. As a consequence, we consider the *MIX* mode with higher DVFS values. The aim of this *MIX* mode is to improve performance and energy consumption while remaining under the power budget. This algorithm is the same as the one previously described but the minimum DVFS frequency is 2.0 GHz instead of 1.2 GHz. Both mechanisms should be used together when the powercap is inferior to 75% of the maximum power.In the remainder of the paper all references to *MIX* policy consider always the high DVFS values (2.0-2.7GHz).

In case we cannot switch-off nodes, the *SHUT* mode can be implemented by keeping nodes idle. In this case, ρ becomes

[6]IPMI (Intelligent Platform Management Interface)

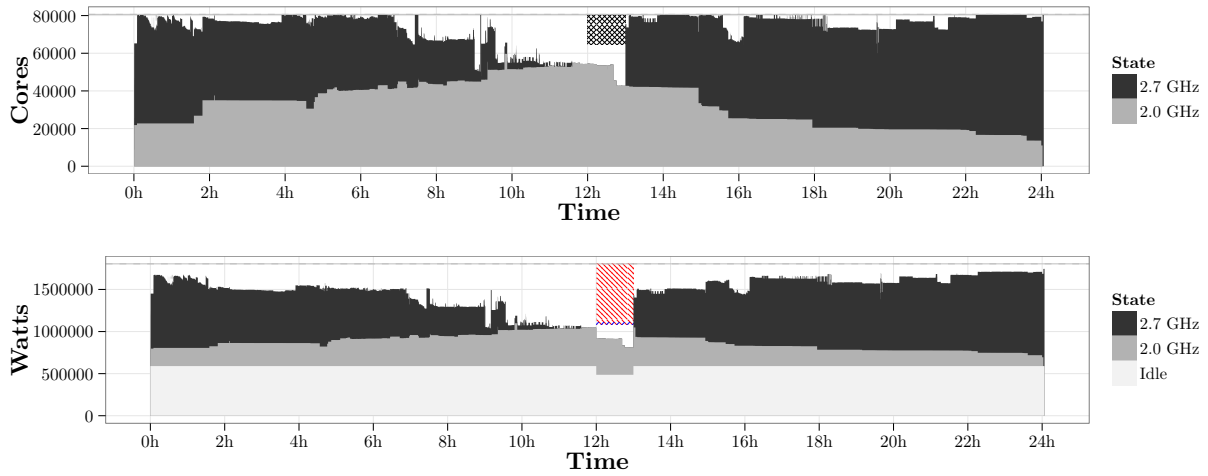


Fig. 6: System utilization for the *MIX* policy in terms of cores (top) and power (bottom) during the 24 hours workload with a reservation (hatched area) of 1 hour of 40% of total power. Cores switched-off represented by a dark-grey hatched area.

positive for all *degradation* values of benchmarks. Thus, *DVFS* turns out to be the best policy in all cases.

VII. Experimental Evaluations

Our choices for experimental evaluations were to: i) use the real workload trace of Curie for approximating production executions of a large-scale supercomputer, ii) take into account the real power consumption data of Curie as discussed in Section III and iii) make use of an emulation technique to study SLURM by using only a small fraction of physical machines.

A. Platform and Testbed

The experiments have been performed upon Nova2 platform which is an internal BULL cluster dedicated for experimentations. The cluster is composed by Intel Sandy Bridge processors with 65 GB of Memory and Infiniband network.

In order to enable real-scale experiments of Curie's workloads with SLURM we need to deploy a configuration of the same size. This is done by making use of an internal SLURM emulation technique called *multiple-slurmd*. This technique is described and validated in [31]. In our context, we deploy 5040 nodes of Curie with only 16 physical nodes of our experimental platform Nova2.

B. Methodology

We propose to replay time intervals extracted from a real workload trace of Curie supercomputer in 2012[7]. In more detail, we select three intervals of 5 hours and one interval of 24 hours with high utilization (most cores are used for computations), big number of jobs in the queue and short inter-arrival time. The intervals used in the following experiments are: i) *medianjob*, with jobs that are representative of the whole workload, ii) *smalljob*, with more small jobs than in the *medianjob* interval, iii) *bigjob*, with more big jobs than in the *medianjon* interval, iv) *24h*, with jobs that are representative of the whole workload. In the extracted traces, Curie is overloaded: there are always at least enough jobs in the submission queues to fill a second cluster of the same size. Most of the jobs are small compared to Curie size, 69% are jobs that need less than 512 cores and ran for less than 2 minutes. 0.1% of jobs are huge, these jobs use more than the equivalent of the whole cluster for 1 hour. It is important to note that in these particular traces, users estimate runtimes badly. In average, they request about 12670 times more walltime than needed (median: 12000). This leads to difficulties for the system to schedule correctly jobs [32].

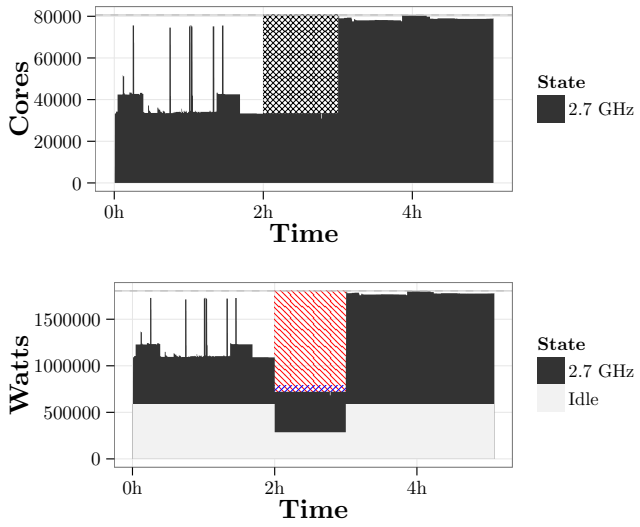
As we are only interested in the RJMS internal decisions, hence the jobs are replaced by simple "sleep" commands. On the original workload, all the jobs were run at maximum DVFS. If our powercap scheduling algorithm decides to run a job at a lower speed, the emulated job will be executed slower. We choose to use a performance *degradation* of 1.63 (1.29 with *MIX*) for all jobs, as our experiments and related works agree that it is a reasonable value (see Section VI-B). This performance degradation is computed with the maximum speed (2.7 GHz) and minimum speed (1.2 GHz or 2.0 GHz with *MIX*), the intermediate values have been linearly interpolated.

The replay of the time interval is based on the four following phases: i) the environment setup: SLURM is set in the closest state of the original run. We put in place the original SLURM configuration of Curie and modified only the parameters that allow our replay (node names, characteristics and power values), ii) the interval initial state setup: runtime characteristics are put in place (queued and running jobs, fairshare values for each user), iii) the actual workload replay: jobs are submitted with the same characteristics as they were on the original run (simple 'sleep' jobs, not real executions), iv) data post-treatment: once the replay of the time interval end, we stop SLURM and then collect and gather information about the replay by the end of the interval (jobs state, outputs and characteristics).

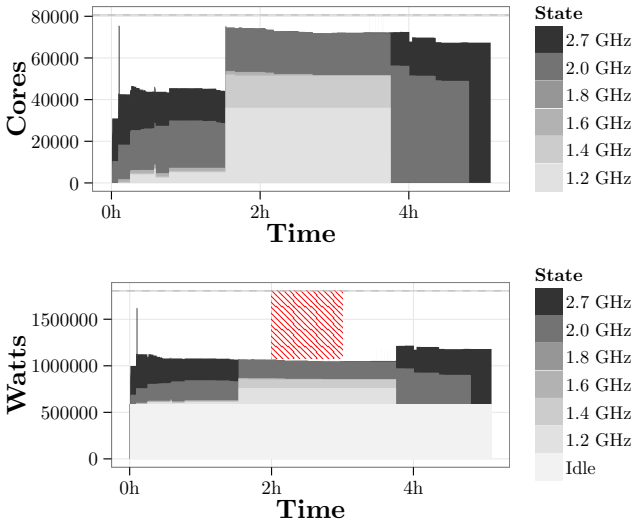
This methodology has some limitations. The initial placement of jobs is not always respected, and we do not replay node failures. Furthermore, job submissions depend of the response time experienced by users [33]. These limitations imply that comparisons to the original traces can not be conducted, but, as the replay is deterministic, we can compare the different replays.

In the experiments reported in the next section, we are eval-

[7] http://www.cs.huji.ac.il/labs/parallel/workload/l_cea_curie/index.html



(a) Powercap of 60% with mainly big jobs and *SHUT* policy



(b) Powercap of 40% with mainly small jobs and *DVFS* policy

Fig. 7: System utilization for the diferent runs in terms of cores (top) and power (bottom) during 5 hours workload with a powercap reservation (hatched area) of 1 hour of 60% or 40% of total power. Cores belonging to switched-off nodes are in cross hatched area.

uating the three different policies *DVFS*, *SHUT* and *MIX*. The policies are tested under three powercap scenarios reserving respectively 80%, 60% and 40% of the available power budget for one hour in the middle of the replayed interval. Powercap reservations are made in the beginning of the workload replay. All experimental results are compared between them along with a simple run where no powercapping takes place. Our goal is to compare system utilization in terms of CPUs and power usage along with the effective work for each scenario.

C. Analysis of results

Figure 6 shows the system utilization (top) and power consumption (bottom) during the replay of the *24h* workload using the *MIX* policy. The grey area in the top figure represents the system utilization of jobs executed upon cores with CPU Frequency of 2.0 GHz whereas the black area represents those running with 2.7 GHz. In the bottom figure the light grey area represents the minimum power consumption of the system if all nodes are idle and no jobs are executed, the grey area represents the additional power consumption of jobs whom the cores compute with 2.0 GHz and the black area reflects the additional power consumed by jobs that compute with 2.7 GHz. The reserved power, is represented by the hatched area in the bottom figure, thus the allowed powercap budget, for that period, is the remaining power below that area. The powercap is triggered in the beginning of workload that is why we observe that jobs are launched with lower CPU frequency directly from the start. Since we are in a *MIX* policy the offline part of the scheduling has reserved a certain number of nodes to shutdown. This can be viewed by the cross hatched area in the top figure during the period of powercap. The small blue cross hatched rectangle below the powercap reservation rectangle represents the bonus power gained back by the grouped shutdown of continuous nodes. This is actually gained power being used by the system for computations but it is plotted upon the reserved power hatched area to better reflect the proportions between them.

It is interesting to observe how while approaching the powercap reservation the system prepares itself for limited power usage by launching more jobs with 2.0 GHz. Similarly after the powercap has passed, utilization of 2.7 GHz cores increases because new jobs are launched with maximum frequency, while older jobs launched with 2.0 GHz, launched before or during the powercap, still remain but gradually decrease. After the powercapped period, we see that the system utilization in terms of cores increases directly to nearly 100%. It seems that a large job allocating more than 40000 cores was scheduled directly after the powercap period. This large job seems to be blocking other smaller jobs that follow and backfilling does not seem to work since thick gaps are witnessed during the powercap interval. Based on previous observations; backfilling is not efficient because of wrong walltime estimations.

If we take a look at other *24h* runs with a powercap of 40%, *DVFS* and *MIX* show similar results: a work around 85% of the total possible work, while *SHUT* has a work of 94% of the total possible work. It is interesting to note that the energy consumption is a the lowest in the *MIX* mode.

Figures 7a and 7b represent the system utilization for the *smalljob* and *bigjob* workloads with different use cases. They are based on the same representations as the previous figure. The difference is that the left one provides results with *SHUT* policy whereas the right one with *DVFS* policy for 60% and 40% powercaps respectively. In the left figure we can observe how the shutdown of nodes makes big space in order to adapt the workload without wasting un-utilized cores. In addition, we can see the value of power bonus due to the regrouping of nodes to be switched-off. Without the offline part of the scheduler this bonus would not be possible. Furthermore, we see how the system utilization increases directly after the powercap interval to 100%. It is interesting to see how backfilling does not take place a lot while preparing for the

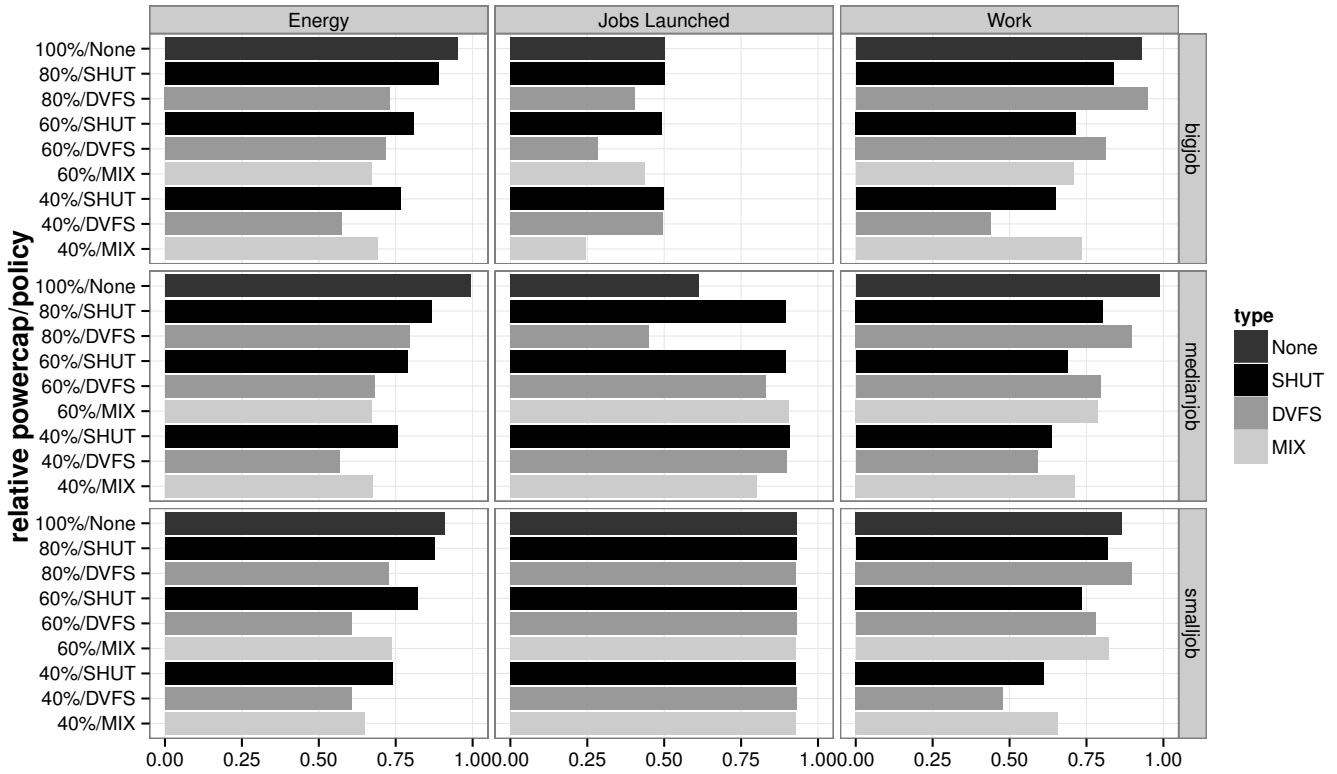


Fig. 8: Comparison of different scenarios of policies and powercaps based on normalized values of total consumed energy, launched jobs and accumulated cpu time during the 5 hours workload interval.

powercap period. This is due to the nature of type of jobs which is mainly big jobs along with the walltime problems that we explained before. In the right figure, different tones of grey represent the different frequencies until the black area which is 2.7GHz. We can see how the appearance of low CPU Frequencies increase while approaching to the powercap period with a total disappearance of 2.7GHz frequencies in close regions to the powercap interval. *DVFS* policy manages to obtain high system utilization with a low power consumption.

We also have done several run with DVFS and switch-off mechanisms deactivated. The only solution for our algorithm is to let nodes idle. As expected, this solution has the worst work (about 40% lower than other modes), while keeping about the same energy consumption.

Let us now look at the impact of the policies for the performances. Figure 8 provides the different runs executed to compare the performance of the different powercap scheduling modes for 5 hours workload. Considering columns we observe the total consumed energy, the number of launched jobs and the total work. In terms of rows we have different groups. The groups based on the workload (left): *bigjob*, *medianjob* and *smalljob* along with the groups representing powercap reservations: 100%, 80%, 60% and 40% which reflect the system power which is allocated for computation. Furthermore distinctions between the different scheduling modes is also made in groups of particular rows in the figure. Only jobs that were running during the replayed time interval are taken into account, and all measures are normalized to the maximal possible value. In the histograms we observe that *DVFS* mode's work is always larger than *SHUT* mode's work and that is because jobs run with lower CPU Frequency and hence the walltime is increased. The *MIX* mode provides most of the time the best energy consumption, while having a work in the same order of others modes.

In the *medianjob* workload, 100%/None and 80%/DVFS runs launched less jobs than others run, while having a high utilization. It seems that in these runs, the algorithm chooses to schedule a huge job preventing a large number of other jobs to be launched.

If we take a look at each mode independently we can see that for every type of workload work and energy decrease proportionally to the powercap diminution. Furthermore, *DVFS* mode seems to be decreasing more rapidly below 60% whereas *SHUT* and *MIX* modes appear to be more consistent. Switch-off mechanism (*SHUT*, *MIX*) seems to be more efficient if we consider the tradeoffs energy/work and this is basically related to the in-advance preparation in the offline part and the gained power due to the bonus.

VIII. Conclusions and Future Works

We presented in this paper a new scheduling algorithm for dealing with power limitations in large scale HPC clusters. The algorithm was developed for a resource and job management system and implemented upon SLURM. It is composed of two phases: an offline part where the planning takes place (choice of policy, selection of group of nodes to be switched-off, etc.) followed by an online part where the power reduction is applied. The implementation upon SLURM resulted into the design of three powercap policies, namely *DVFS*, *SHUT* and *MIX* which respectively take advantage of CPU Frequency scaling, nodes shut down and mixed capabilities in order

to achieve power reductions whenever needed. One of our main objectives was to enable the scheduler to determine automatically the best powercap mechanism for the nodes and we showed how this depends on the architecture, the power consumption of the components and the actual workload. The new developments will appear on the main branch of SLURM in the upcoming version 15.08. As far as our knowledge, this is the first work that considers powercapping techniques in the level of job scheduling for a resource and job management system in HPC. The study allowed us to validate the algorithms and evaluate the different policies through real-scale emulation of a petaflopic supercomputer. In particular we performed experiments with emulation of Curie's characteristics and calculated power values using replay of a real workload trace collected from the production of Curie on 2012. The experimental results validate the model and provide interesting initial insights. Switching-off nodes appear to be the most efficient policy in our use cases of less than 60% powercaps, mixed policy seems to be the more consistent one and finally frequency scaling provides better results with large powercaps of 80%.

The studies will continue to correlate the proposed model with application preferences concerning DVFS. Indeed, if an application is able to provide optimized DVFS values, this should be taken into account by the algorithm. Then, the global performance of the cluster will be improved while respecting the powercap. *SHUT* policy makes an offline selection of the group of nodes to be switched-off in order to take advantage of the power bonus. Nevertheless, this might increase the fragmentation of the system in case of un-homogeneous infrastructures. Hence, deeper studies are needed to compare the rigidity of the selection of the nodes to be switched-off with a more flexible selection of nodes. For future improvements on the code for *DVFS*, we will consider to dynamically change the CPU frequencies while the jobs are running, this will allow nodes to adjust the power consumption instantly whenever it is needed. This will eventually result into faster power decrease when a powercap period is approaching and lower jobs' turnaround time after a powercap period is over. The optimal DVFS choice for the best power/performance trade-offs could be also determined through a particular profiling run as proposed in [34]. In addition, exploring other techniques to limit power consumption upon utilized nodes, such as RAPL, will be also studied. Finally, we would like to adapt the powercapping algorithm in order to consider the real-time power consumption measures of the nodes [26], instead of considering the static values defined during the initialization phase.

REFERENCES

[1] S.-g. Kim, C. Choi, H. Eom, H. Yeom, and H. Byun, "Energy-centric DVFS controling method for multi-core platforms," in *SCC 2012*.

[2] B. Lawson and E. Smirni, "Power-aware resource allocation in high-end systems via online simulation," in *SC 2005*.

[3] D. C. Snowdon, S. Ruocco, and G. Heiser, "Power management and dynamic voltage scaling: Myths and facts," in *PARTS 2005*.

[4] R. Schöne and D. Hackenberg, "On-line analysis of hardware performance events for workload characterization and processor frequency scaling decisions," in *ICPE 2011*.

[5] H. Kimura, M. Sato, T. Imada, and Y. Hotta, "Runtime DVFS control with instrumented code in power-scalable cluster system," in *Cluster 2008*.

[6] A. Gandhi, M. Harchol-Balter, R. Das, J. O. Kephart, and C. Lefurgy, "Power capping via forced idleness," 2009.

[7] B. Rountree, D. Lowenthal, M. Schulz, and B. De Supinski, "Practical performance prediction under dynamic voltage frequency scaling," in *IGCC 2011*.

[8] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "Understanding the future of energy-performance trade-off via DVFS in HPC environments," *Journal of Parallel and Distributed Computing*, 2012.

[9] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," *TPDS*, 2007.

[10] D. Aikema, C. Kiddle, and R. Simmonds, "Energy-cost-aware scheduling of HPC workloads," in *WoWMoM*, 2011, pp. 1–7.

[11] J. Hikita, A. Hirano, and H. Nakashima, "Saving 200kw and $200 k/year by power-aware job/machine scheduling," in *IPDPS 2008*, 2008.

[12] E. D. Demaine, M. Ghodsi, M. T. Hajiaghayi, A. S. Sayedi-Roshkhar, and M. Zadimoghaddam, "Scheduling to minimize gaps and power consumption."

[13] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *HPCA-9 2003*.

[14] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers." in *USENIX annual technical conference, General Track*, 2005.

[15] T. Singh and P. K. Vara, "Smart metering the clouds," 2009.

[16] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, "Beyond DVFS: a first look at performance under a hardware-enforced power bound," in *IPDPSW 2012*.

[17] S. Reda, R. Cochran, and A. K. Coskun, "Adaptive power capping for servers with multithreaded workloads," *IEEE Micro*, 2012.

[18] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *SIGARCH 2007*.

[19] G. A. Geronimo, J. Werner, R. Weingartner, C. B. Westphall, and C. M. Westphall, "Provisioning, resource allocation, and DVFS in green clouds."

[20] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "BSLD threshold driven power management policy for HPC centers," in *2010 IPDPSW*, 2010.

[21] ——, "Optimizing job performance under a given power constraint in HPC centers," in *IGCC 2010*.

[22] ——, "Parallel job scheduling for power constrained HPC systems," *Parallel Computing*, 2012.

[23] J.-M. Pierson and H. Casanova, "On the utility of dvfs for power-aware job placement in clusters," in *Euro-Par 2011*.

[24] A. Mu'alem and D. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling," *TPDS 2001*.

[25] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux utility for resource management," in *JSSPP*, 2003.

[26] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux, "Energy accounting and control with slurm resource and job management system," 2014.

[27] "http://www.netlib.org/linpack/."

[28] J. D. McCalpin, "A survey of memory bandwidth and machine balance in current high performance computers," *IEEE TCCA Newsletter*, 1995.

[29] "https://software.intel.com/en-us/articles/intel-mpi-benchmarks."

[30] H. J. Berendsen, D. van der Spoel, and R. van Drunen, "Gromacs: A message-passing parallel molecular dynamics implementation," *Computer Physics Communications*, 1995.

[31] Y. Georgiou and M. Hautreux, "Evaluating scalability and efficiency of the resource and job management system on large hpc clusters," in *JSSPP*, 2012.

[32] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Modeling user runtime estimates." Springer, 2005.

[33] E. Shmueli and D. G. Feitelson, "Uncovering the effect of system performance on user behavior from traces of parallel systems," in *MASCOTS*, 2007.

[34] A. Auweter, A. Bode, M. Brehm, L. Brochard, N. Hammer, H. Huber, R. Panda, F. Thomas, and T. Wilde, "A case study of energy aware scheduling on supermuc," in *ISC 2014*.