A Novel Approach for Job Scheduling Optimizations under Power Cap for ARM and Intel HPC Systems

Dineshkumar Rajagopal*, Daniele Tafani†, Yiannis Georgiou*, David Glesser* and Michael Ott†
*Bull Atos Technologies
Email: name.surname@atos.net
†Leibniz Supercomputing Centre
Email: name.surname@lrz.de

Abstract—**The ever-increasing energy demands of modern High Performance Computing (HPC) platforms is undeniably one of the most critical aspects for the future design and evolution of such systems. The capability of managing their energy consumption not only allows for significant reduction in electricity costs but is also a step forward on the road towards the exascale. Powercapping is a widely studied technique that contributes to address this challenge by instantaneously setting and maintaining a predefined power threshold (power cap) that cannot be exceeded. However, the lack of a centralized mechanism responsible for efficiently allocating the available power among resources and jobs may ultimately yield to fragmentation, low system utilization and increased user waiting times. Additionally, power cap violations can lead to high risk scenarios and/or increase operational costs. This paper proposes to prevent such issues with the introduction of the Enhanced Power Adaptive Scheduling (E-PAS) algorithm. The E-PAS algorithm combines scheduling and resource management mechanisms, correlating estimated and real power consumption data in order to optimize the resource utilization of the platform under a predefined power cap. The algorithm has been implemented in the widely used open-source resource and job management system SLURM and is planned to be pushed in a future mainstream version. Its effectiveness has been evaluated through real-scale experiments respectively on an ARM- and an Intel-based cluster of comparable size. All experiments have been performed using synthetic workloads from a set of mini-applications.**

1. Introduction

In the past decades, the evolution of High Performance Computing (HPC) systems has traditionally been associated with an exponential rise of computing power. To give an idea of the tremendous increase in performance that such systems experienced in relatively short time, the current number one system of the TOP500 list, Sunway TaihuLight, peaks at approximately 125 PFlop/s, more than twice as powerful as its number one predecessor in 2015, Tianhe-2 [1]. However, less emphasis has been placed into the power required to run such machines. For example, the aggregate power required to operate Tianhe-2 (including external cooling infrastructure) is about 25 MW, the amount required to power a small town. A more general overview of this issue is provided by the "United States Data Center Energy Usage Report" in 2016 [2]. According to the report, in 2014 U.S data centers consumed an estimated 70 billion kWh, corresponding approximately to 1.8% of the total U.S. electricity consumption. Based on current trend estimates, the authors predict that the energy usage is expected to increase of about 4% for the time frame 2014-2020, yielding to a projected energy consumption associated with U.S. data center operations of approximately 73 billion kWh in 2020. Under these circumstances, it appears obvious that the energy consumption issue represents a major challenge for developing future, energy-efficient exascale HPC platforms.

Over the last decade, the scientific and industrial communities have proposed several solutions to minimize and/or manage the energy consumption of HPC centers, from both hardware and software perspectives. A recent study [3] demonstrates the economic viability of hardware over-provisioned systems in power-constrained HPC systems. On this basis, we expect powercapping in conjunction with adaptive scheduling techniques to become a de-facto operational profile in HPC centers. Hence, scheduling optimizations to increase system utilization and minimize fragmentation under strict power budgets will be of top interest for future HPC platforms, regardless of their architecture. This paper introduces the Enhanced Power Adaptive Scheduling (E-PAS) algorithm which allows system administrators to limit and manage the power consumption of an HPC system, integrating power-aware coordination mechanisms within the open-source Resource and Job Management System (RJMS) SLURM [4].

The E-PAS algorithm is based on the Power Adaptive Scheduling algorithm (previously presented in [5] and included in SLURM since version 15.08), which ensures that the total power consumption of the system does not exceed a predefined power threshold (a powercap) with the employment of opportunistic shutdown of compute nodes and power reduction via popular Dynamic Voltage and Frequency Scaling (DVFS) techniques. Differently from the PAS algorithm, the E-PAS algorithm takes into account real power and energy consumption data from the monitored platform, significantly improving the resource utilization of the system and reducing users' job waiting times. The main contribution of the E-PAS algorithm relies in the acquisition and management of power and energy consumption data in the RJMS, which have been respectively implemented extending the SLURM "power/plugins" framework and exploiting the SLURM "Layouts Framework". In this paper we evaluate two different versions of the algorithm (namely, two different implemented "plugins") associated with two different hardware architectures: one specialized for Intel architectures based on the Intel-processors' Running Average Power Limit (RAPL) technology and one for an ARM-based platform, the Mont-Blanc prototype, based on a customized power monitoring infrastructure. The experiments have been performed with synthetic workloads based on a set of representative mini-applications. The powercap requirement is satisfied for both architectures with an additional guarantee for the Intel platform due to its internal RAPL powercap enforcement.

The remainder of this paper is as follows: in Section 2 we provide a list of proposed solutions for power- and energy-aware scheduling in HPC systems, while Section 3 describes the hardware platforms used for evaluating the proposed algorithm along with their respective power monitoring infrastructure. Section 4 illustrates the core logic of the E-PAS algorithm and Section 5 presents the experimental evaluation for validating its efficacy. Conclusions and future works are provided in Section 6.

2. Related Works

As mentioned in the introduction, power and energy awareness have become topics of significant importance for the scientific community, hence it is without surprise that a variety of different contributions have been proposed over the past years to help addressing this issue.

The survey provided in [6] gives a thorough analysis of related work on power management strategies along with details on the relationship between HPC centers and electricity service providers in the United States. Among the different studied techniques, powercapping in conjunction with job scheduling and node-shutdown mechanisms appeared as the most promising. A recent study [7], implicating a larger group of supercomputers and electricity providers in both the U.S. and Europe, showed that while upper power-bounds play an important role, power variations ultimately do not affect the final energy cost in most use cases. Patki et al. [8] argue that thanks to the control of power consumption, one can buy a bigger cluster for the same annual price while improving scheduling under power budget.

The efficacy of powercap strategies has been further evaluated by Etinski et al. in a series of papers, e.g. in [9], [10] and [11], where DVFS was the only mechanism used to satisfy power thresholds while keeping relatively good performances. Differently from the solution proposed in these papers, we consider a more sophisticated mechanism that uses RAPL internal powercapping in conjunction with DVFS and node idling for Intel platforms while combining DVFS with node idling for ARM platforms. In the context of cloud computing, Geronimo et al. proposed a virtual machine manager that can use DVFS, updates the virtual machine resources, migrates them and shuts down opportunistically some processors [12]. Fan et al. [13] defined a methodology to reduce the global cost of data centers by buying more processors and capping their power consumption. Our work is dedicated to HPC systems, where constraints are very different and, on top of powercap satisfaction, we further improve the workload execution. In [14] the authors tried to optimize throughput under powercap budget showing interesting simulated results. However, their solution employs Integer Linear Programming which may not be practical for realistic scheduling scenarios, as it can introduce delays and may not scale optimally in large systems. Gholkar et al. in [15] presented a 2-level hierarchical powercapping solution based on RAPL which is certainly more promising than employing only DVFS to guarantee a global powercap. Similarly to [13], our main difference is that the E-PAS algorithm attempts to additionally optimize the resource utilization under power constraints.

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 for DVFS tuning. In [18] the authors provide runtime mechanisms that, in correlation with the resource manager system, can dynamically adapt frequency and power allocations across nodes to improve jobs' power efficiency under a powercap budget. Ellsworth et al. in [19] provide an ongoing study based on a SLURM "power plugin" implementation that leverages the RAPL technology for Intel platforms. Although sharing some similarities with our work, all the previously mentioned solutions only apply to Intel architectures and, in the case of [19], users do not have the liberty of deciding whether their applications should run during powercap time frames or not. Finally, the authors in [20] provide machine-learning mechanisms to predict power profiles and introduce a new power-aware scheduler that schedules jobs based on such profiles, satisfying a power-cap while minimizing impact on system utilization. However,the authors do not seem to use any real-production RJMS, but simulations based on a year real production workload traces.

In our previous studies [5], we introduced the PAS algorithm technique at the RJMS level which, by taking into account the application needs and requirements along with the maximum tolerable power consumption at cluster-level, managed to determine if a submitted application could be executed or not selecting optimal CPU frequencies. However, the algorithm has two shortcomings: i) the selection of CPU frequencies does not directly reflect a fixed power consumption and ii) the computation of the actual power consumption is based on maximum estimations and not on real data. Finally, in [21] we proposed a scheduling strategy based on EASY backfilling that coordinates the execution of jobs allowing powercap violations under a fixed energy budget. Similarly to [5], this work was not based on real monitored data. As mentioned in the introduction, the E-PAS algorithm proposes to resolve this issue with the introduction of two "power/plugins".

3. The Experimental Testbeds

In order to better introduce the hardware-specific implementation of the plugins that constitute the core of the E-PAS algorithm, we provide here a brief description of the platforms that have been employed for performing our experiments, namely the ARM-based Mont-Blanc prototype and an Intel-based "Bull internal" cluster. We further include details on how power monitoring is performed on both systems.

3.1. The ARM-based Platform: The Mont-Blanc Prototype

The Mont-Blanc prototype is the result of the successful completion of the first phase of the Mont-Blanc project, which ran from 2011 until 2013. Currently at its third iteration, the Mont-Blanc project [22] is a EU initiative funded under the European 7th Framework Programme and aims at addressing the energy efficiency challenge in exascale systems by proposing a new type of computer architecture employing embedded ARM-based technology.

The Mont-Blanc prototype is hosted in the Barcelona Supercomputing Center and consists of two racks, each hosting four "BullX" chassis with nine blades per chassis. The core of a Mont-Blanc blade resides in the Ethernet Mother Board (EMB), which comprises 15 Samsung Daughter Boards (SDB), interconnected via a 1GbE switch fabric providing two 10 GbE up-links. Each SDB logically corresponds to a compute node and hosts a Samsung Exynos 5 Dual System on a Chip (SoC), which includes a dual core ARM Cortex A-15 CPU and an ARM Mali T-604 GPU. Overall, the entire system fits 1080 compute nodes, for a total of 2160 CPU cores and 1080 GPUs. In order to avoid potential issues with concurrent users of the platform, we decided to run our experiments on a reservation of 260 SDBs (that is, 520 CPU cores and 260 GPUs).

A dedicated power sensor for each SDB has been implemented on the EMB. This last also hosts a Board Management Controller (BMC), which manages the interface between system management software and platform hardware. The blade further hosts an FPGA responsible of handling the parallel collection and pre-processing of power consumption readings. Every 1120ms, the FPGA accesses the power sensors of the SDBs hosted in the blade via I^2C; it then averages 15 power values, each corresponding to the total power consumed by an SDB, and stores them in a FIFO buffer. The BMC polls the FPGA every 500ms retrieving all available power samples, associating a time stamp to the readings and storing them in its DDR2 memory. The data can be then conveniently retrieved via customized Intelligent Platform Management Interface (IPMI) commands.

Due to its prototype nature, a dedicated, out-of-band power monitoring tool has been developed for the Mont-Blanc system in order to automatically acquire and store the power consumption value of each node. The tool is defined by three main software components, namely:

- a key-value store, responsible for storing the monitored power data.
- one or more "pushers", responsible for retrieving power monitored data through different protocols (in the Mont-Blanc system case, via IPMI),
- one or more "collect agents", responsible for gathering all the power data acquired by the pushers and storing it in the key-value store of the tool.

A set of implemented command-line API conveniently allows for acquisition of the raw monitored data collected by the tool. The interested reader can find a more detailed evaluation of the Mont-Blanc prototype and of the power monitoring tool respectively in [23] and in [24].

3.2. The Intel-based Platform: the Bull Internal Cluster

The Intel-based Bull internal cluster is composed of 26 Intel Ivy Bridge compute cards interconnected via InfiniBand. Each node hosts two sockets, with 10 cores per socket, for a total of 520 cores. The Intel Ivy Bridge processor supports the Running Average Power Limit (RAPL) technology, which allows in-band accurate energy monitoring based on Intel-specific software models. The Model-Specific Registers (MSRs) on Intel platforms provide privileged functionalities for monitoring and controlling CPU related properties and can be used to export RAPL related features through the Linux "msr" kernel modules. To simplify the usage of RAPL and to improve portability, we employed the "libmsr" library which provides a set of API to control RAPL features. Power acquisition and powercap allocations functions have been implemented through SLURM remote procedure calls.

The RAPL technology used as monitoring infrastructure returns energy consumption per socket. The default configuration in our case has been to retrieve the average power consumption of a socket by correlating its energy consumption every 4 seconds. In contrast to the ARM-based Mont-Blanc platform, where the dedicated power monitoring tool provides power data at the granularity of a compute node, in the Intel platform the acquisition of energy data happens at socket-level and is distributed evenly among cores. This means that each obtained power value does not include the power consumed by the rest of the components (e.g., motherboard, disk, NIC, etc.) but reflects only the power consumed by the associated core.

4. The Enhanced Power Adaptive Scheduling (E-PAS) Algorithm

In this section we provide a detailed explanation of the core functionalities of the E-PAS algorithm. As mentioned in the introduction of this paper, the E-PAS algorithm is an extension of the PAS algorithm. Hence, for the sake of completeness, we first provide the reader with a brief description of the PAS algorithm's main logic (depicted in Figure 1). For a fully detailed explanation of the PAS algorithm, the reader is invited to refer to [5] and [25].

4.1. The Power Adaptive Scheduling (PAS) Algorithm

The Power Adaptive Scheduling (PAS) algorithm gets activated whenever the system administrator applies a desired power cap, which can be valid immediately or for a future, pre-defined time frame. When a job is submitted, the scheduler predicts the power consumption of its execution by summing the maximum power consumption of all the requested resources. Subsequently, it adds it up to the monitored power consumption of the whole system and verifies that this last remains under the allowed power threshold. If this is the case, the resources requested by the job will be allocated for its execution, otherwise the job returns in a pending state to prevent violations of the configured power constraint. However, the user can increase the chance of executing her job if she provides a CPU frequency window as an additional job parameter. In this case, the scheduler will attempt to execute the job at the maximum CPU frequency that will still maintain the total cluster power consumption under the defined power threshold (in order to do so, the scheduler calculates the power consumed by the job at all provided CPU frequencies considering once again the maximum power consumption of each requested resource). If not, the job remains in the queue. Furthermore, in case resources remain idle, nodes can be turned off and by doing so, the scheduler redistributes available power which could help jobs start faster.

The PAS algorithm has been in the mainstream SLURM version since August 2015. The global details on power data related to resources are made available through the Layouts Framework [26], which provides a key-value store and a set of API that conveniently allows for advanced management and scheduling of new type of resources. As mentioned above, the main assumption of the PAS algorithm is that the instantaneous power consumption of the whole cluster can be calculated at any moment by retrieving the power consumption of each resource connected to the cluster. Nevertheless, the power consumption data is defined statically within SLURM Layouts Framework configuration files and is based on static, theoretical (or measured only once) maximum values. In this way, violations of the powercap are prevented but the scheduler always overestimates the system's power consumption. This behavior is inevitably prone to inaccuracies, leading mainly to potential low system utilization and longer job waiting times.

4.2. Enhancing PAS: the E-PAS Algorithm

The E-PAS algorithm proposes to resolve this issue by extending the PAS algorithm with the implementation of "plugins" for retrieving real monitored power consumption data from the system and correlating the estimations with the dynamically updated power data. The development of such plugins is based on the "power/plugins" feature of SLURM introduced by SchedMD for Cray architectures in [27]. Figure 2 illustrates the high level architecture of the E-PAS algorithm. As it can be observed, the core logic of the

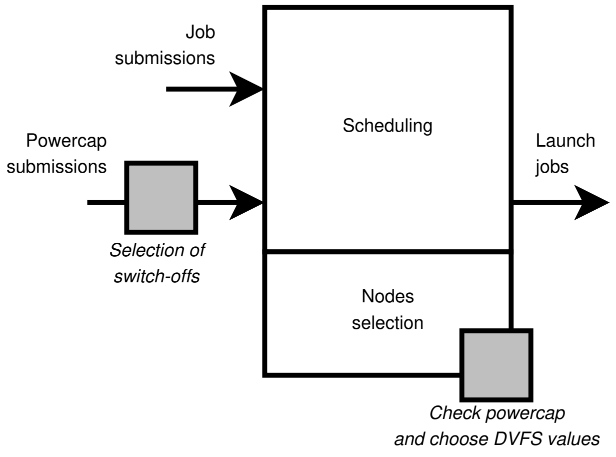

Figure 1. The core logic of the Power Adaptive Scheduling algorithm.

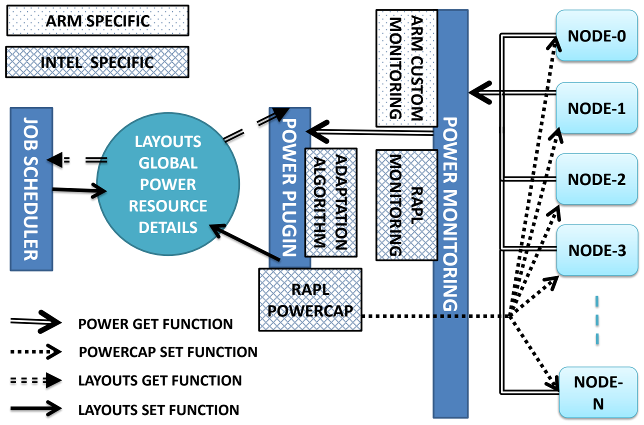

Figure 2. High level architecture of the E-PAS algorithm.

scheduler (which is largely based on the PAS algorithm) still involves an interaction with the Layouts Framework for acquiring power data and updating the power cap. The generic E-PAS algorithm schedules jobs and estimates their associated power consumption in the same fashion of the PAS algorithm. However, the power data is dynamically updated through platform-specific "plugins" which enable the scheduler to make more accurate estimates and improve system-wide metrics. Obviously, implementing the logic of E-PAS in generic terms for different architectures requires addressing the development of specific power data acquisition mechanisms that are inevitably architecture-specific. In order to validate our algorithm cross-platform, we implemented two different versions based on each hardware-specific power monitoring infrastructure presented in the previous section, specifically "power/arm" for ARM and "power/rapl" for Intel. In the following sections we describe the implementation details of both versions.

4.3. ARM Version

The ARM version of the E-PAS algorithm employs the "power/arm" plugin which constitutes the interface between the SLURM Layouts Framework and the power monitoring infrastructure of the Mont-Blanc prototype. Due to the customized nature of the power monitoring infrastructure of the system, the acquisition of power data within the "power/arm" plugin has been realized in the most convenient and less intrusive way by implementing a lightweight HTTP server within the power monitoring tool. Specifically, the submission of a new job triggers the procedure of updating the power consumption of the system by retrieving the latest power consumption values of each compute node. These values are collected with the power monitoring tool and immediately cached within the "collect agent". Subsequently, the RJMS can retrieve the power data issuing a simple HTTP GET request message. For the Mont-Blanc prototype case, each power value returned is averaged over an interval of time of 5 seconds. Once acquired by the RJMS, the power values are finally saved in the key-value store of the SLURM Layouts Framework and used by the E-PAS ARM algorithm to verify if the power consumption associated with the execution of the submitted job will determine a violation of the defined powercap.

4.4. Intel Version

Differently from the "power/arm" plugin, the Intel-based "power/rapl" plugin ensures that the defined maximum power threshold is not exceeded by exploiting the implemented RAPL powercap functionality that is integrated at socket level, allowing for better power balancing between compute nodes. Hence, RAPL is not used only as a monitoring feature but also as a hardware powercap setting. However, as briefly mentioned in Section 3, despite offering a set of configurable parameters that improves the overall monitoring accuracy, RAPL fails to encapsulate the power consumption contribution of other system components (as opposed to the Mont-Blanc power monitoring system that instead encompasses the power consumed by the entire compute node). The main logic behind the "power/rapl" plugin is encapsulated into an additional "adaptation algorithm", whose main steps are listed in the pseudo-code of Algorithm 1. Specifically, let us define the following variables:

- P_i^k and PC_i^k are respectively the power consumption and the powercap values associated with socket i at step k,
- P is the total power unused by sockets and available for redistribution among them,
- p is the maximum average power that can be made available to a socket,
- t and T represent respectively the lower and upper thresholds of a range of power values expressed as percentages of the socket-level powercap PC_i^k. Their values are comprised between 0 and 1,
- r and R represent the rate at which each socket-level powercap value is respectively decreased or increased. Similarly to t and T, both variables are expressed as a percentage of the powercap PC_i^k and their values are within 0 and 1,
- n is the number of sockets that are eligible for raise or their associated powercap.

The algorithm follows a two-phase procedure: in phase 1, the algorithm assesses for each socket whether its power consumption is close to its associated powercap or not (through comparisons with the predefined upper and lower power thresholds t and T). If this is the case, then the socket is eligible to a raise or a decrease of its powercap, otherwise nothing changes. The maximum average power available per socket p is then calculated (all the steps of phase 1 are listed from line 1 to 8). Phase 2 (from line 9 to 21) takes care of the reconfiguration of the internal powercaps associated with the eligible sockets identified in phase 1. It should be noted that if $t \cdot PC_i \leq P_i \leq T \cdot PC_i$, then the powercap PC_i associated with socket i will not change. After calculating it, the new powercap value PC_i^{k+1} will be saved in the RAPL powercap register of the compute node through SLURM remote procedure calls. The complexity of Algorithm 1 is $O(number\_of\_sockets)$, favoring scalability for clusters with a large number of sockets.

5. Experimental Evaluation and Results

In this section we describe the experimental methodology and present results of the evaluation of the E-PAS algorithm when employed upon the hardware platforms presented in Section 3. The methodology consists of deploying the same synthetic workload based on a variation of the Effective System Performance (ESP) benchmark [28], [29]

Algorithm 1 Adaptation algorithm for the "power/rapl" plugin

```
1:  for all i do
2:      if $P_i^k < (PC_i^k * t)$ then
3:          $P \leftarrow P + (PC_i^k - P_i^k)$
4:      else if $P_i^k > (PC_i^k \cdot T)$ then
5:          $n \leftarrow n + 1$
6:      end if
7:  end for
8:  $p \leftarrow P/n$
9:  for all i do
10:     if $P_i^k < (PC_i^k \cdot t)$ then
11:         $PC_i^{k+1} \leftarrow PC_i^k - (PC_i^k \cdot r)$
12:         $PC_i^{k+1} \leftarrow max(PC_i^{k+1}, PC_i^k - (PC_i^k - P_i^k)/2)$
13:     else if $P_i^k > (PC_i^k \cdot T)$ then
14:         $PC_i^{k+1} \leftarrow PC_i^k + (PC_i^k \cdot R)$
15:         $PC_i^{k+1} \leftarrow min(PC_i^{k+1}, (PC_i^k + p))$
16:         $n \leftarrow n - 1$
17:         $p \leftarrow (P - (PC_i^{k+1} - PC_i^k))/n$
18:     else
19:         $PC_i^{k+1} \leftarrow PC_i^k$
20:     end if
21: end for
```

and composed of three mini-applications. In the evaluation, we compare the efficacy of the new E-PAS algorithm against the PAS algorithm. The comparisons take place upon both platforms through the same experiments.

5.1. Application Modeling

The experimental methodology is based on the execution of three mini-applications namely CoMD, Lulesh and MP2C, all deployed through a synthetic workload based on the LightESP benchmark [30]. The LightESP benchmark, which is based on ESP, provides a synthetic workload of 230 jobs of different classes, where each class has the same fixed execution time and number of allocated resources. It is worth noting that the workload is adapted to the total number of the cluster resources.

Both algorithms refer to the maximum power consumption value at each CPU frequency of compute nodes to predict the power consumption of a submitted job; for this reason, we profiled the power consumption and the execution time of all the mini-applications at each CPU frequency and for both hardware architectures. The resulting trade-off graphs are shown in Figure 3 for the Mont-Blanc prototype and in Figure 4 for the Intel cluster. For both platforms it emerges that the maximum power consumption curve is given by Lulesh, hence we populate the SLURM "Layouts Framework" configuration file with the power data associated with this mini-application at all frequencies. Additionally, we note that in both graphs all the trade-off curves constitute Pareto points, meaning that for each CPU frequency we can either optimize execution time or power consumption. In our experiments we decided to prioritize performance over power consumption, hence we defined a CPU frequency window for submission of jobs of 1GHz to 1.6GHz for the Mont-Blanc system and of 2.2GHz to 2.8GHz for the Intel platform.

5.2. Results

In these experiments our goal is to evaluate the differences in system utilization, maximum power consumption and job waiting time between the PAS and the E-PAS algorithms. As extensively mentioned throughout the paper, the main advantage of the E-PAS algorithm relies on the ability of distributing the power resources over jobs under powercap regime on the basis of real power consumption data. We would also like to draw the attention of the reader to the fact that the main reason behind testing the algorithms on both platforms does not lie in an assessment of the performance and/or energy efficiency of the two different systems, but rather to validate the effectiveness of the algorithms and to demonstrate that the obtained results are consistent independently of the employed architecture. On top of this, to the best of our knowledge, these power-aware scheduling optimizations are probably the first of their kind applied on an ARM-based system of the size of the Mont-Blanc platform.

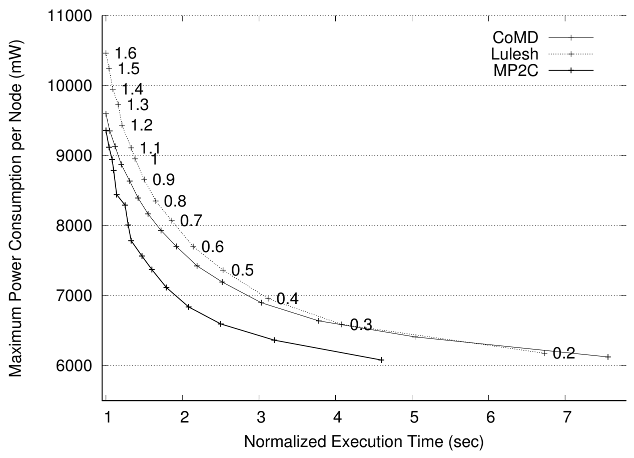

Figure 3. Trade-offs between execution time and maximum power consumption per node: ARM-base platform case.

The same LightESP workload defined in Section 5.1 is executed for both algorithms on both platforms. The powercap is set for both architectures at 80% of their maximum cluster power consumption and is valid for the entire workload execution. This means that the RJMS can redistribute no more than 80% of the total power consumption of the cluster among jobs. For the Intel-based platform the maximum power consumption is 5200W, hence the configured powercap for the experiments is set at 4200W. We further set the RAPL variables defined in Algorithm 1 at $t = 0.92$, $T = 0.96$, $r = 0.1$ and $R = 0.4$. Conversely, the Mont-Blanc testbed reaches a maximum power consumption of 3120W, hence its associated system powercap is set at 2500W. Note that, as mentioned in Section 3, the power monitored data of the ARM platform reflects the aggregate power consumption of all compute nodes (including all additional components hosted by an SDB), providing a more comprehensive view of the total system power profile than the one offered by the Intel cluster.

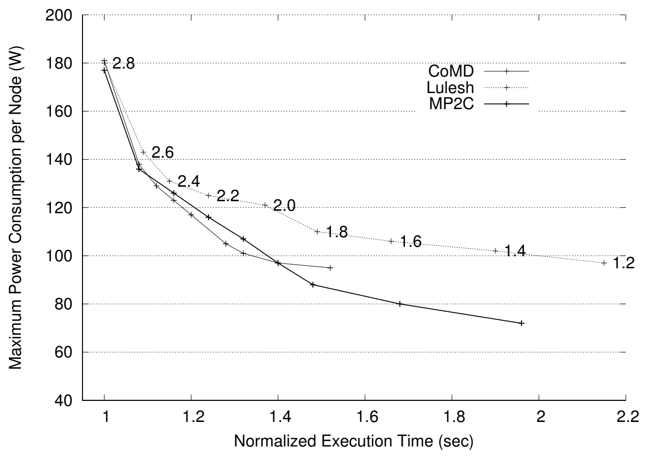

Figure 4. Trade-offs between execution time and maximum power consumption per node: Intel-base platform case.

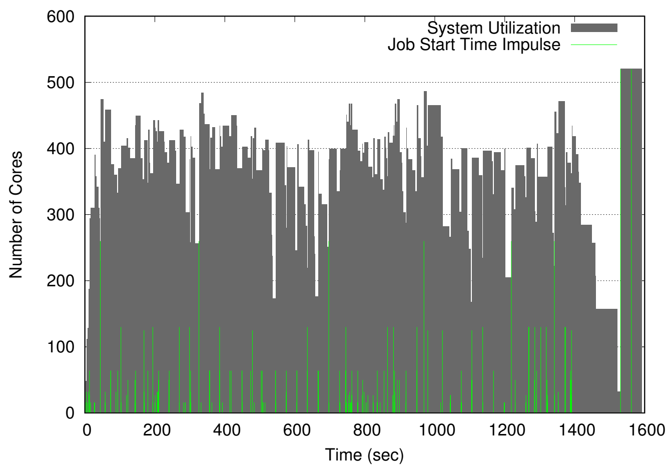

Figure 5. System Utilization of the Intel-based platform for the PAS algorithm. The total number of cores is 520 and the powercap is set at 4200W. Job impulses represent jobs starting at a certain instant of time, with the number of requested cores reflected by the height of the impulse.

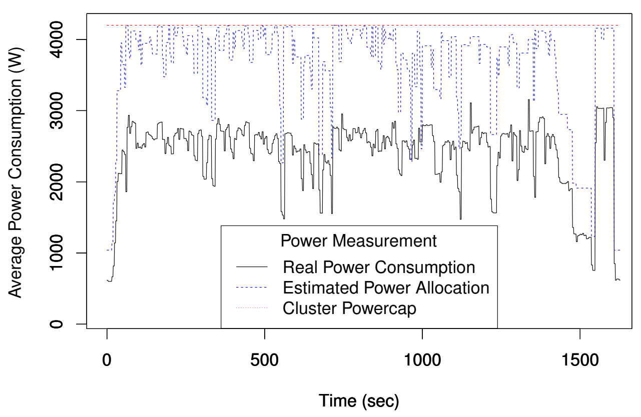

Figure 6. Power consumption profile of the Intel-based platform for the PAS algorithm. The total number of cores is 520 and the powercap is set at 4200W.

Figures 5 and 7 show the system utilization when executing the same workload under the same power cap on the Intel platform, respectively for the PAS and the E-PAS algorithm case. Similarly, Figures 10 and 12 illustrate the exact same situation for the Mont-Blanc platform. It is interesting to acknowledge the positive effect that the redistribution of power driven by real data has for the overall system utilization in both architectures when employing the E-PAS algorithm. For example, in Figures 7 and 12 the scheduler manages to pack the jobs in a better way utilizing much more available resources than it does in Figures 5 and 10 for the PAS algorithm case (e.g., yielding to a resource utilization gain of approximately 10% for the Intel case and 2% for the ARM case). This is due to the fact that PAS largely overestimates the maximum power consumption of the system, achieving a much lower resource utilization. On the contrary, by capturing with higher precision the

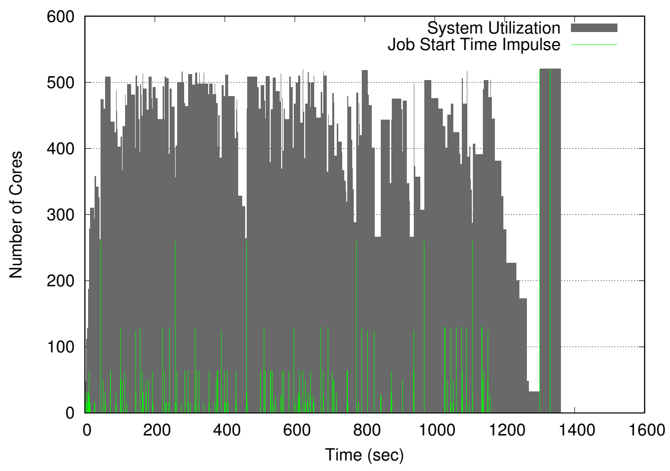

Figure 7. System Utilization of the Intel-based platform for the E-PAS algorithm. The total number of cores is 520 and the powercap is set at 4200W. Job impulses represent jobs starting at a certain instant of time, with the number of requested cores reflected by the height of the impulse.

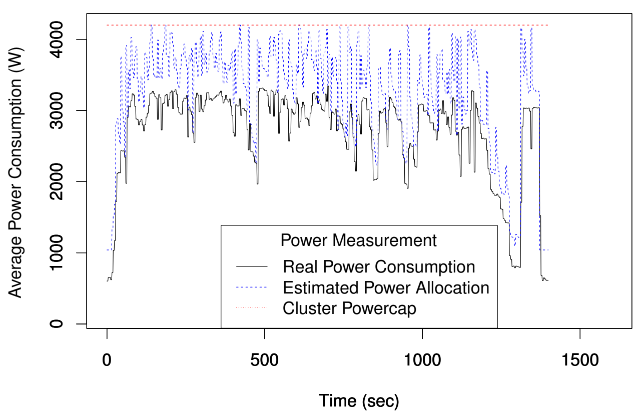

Figure 8. Power consumption profile of the Intel-based platform for the E-PAS algorithm. The total number of cores is 520 and the powercap is set at 4200W.

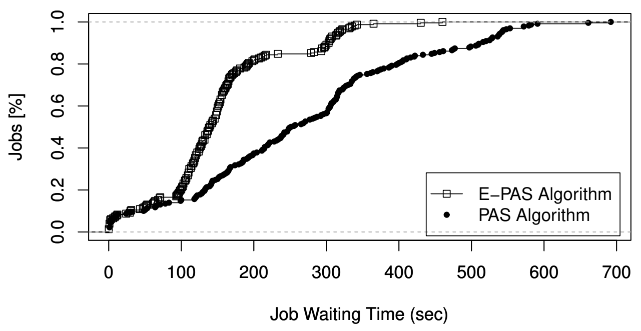

Figure 9. Cumulative distribution function of the job waiting time on the Intel-based platform.

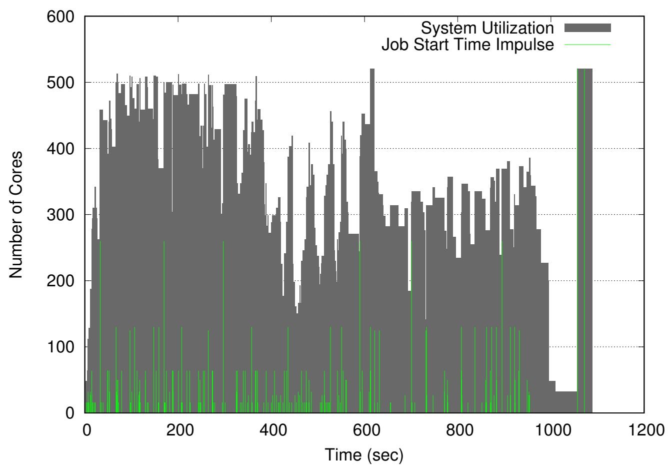

Figure 10. System Utilization of the ARM-based platform for the PAS algorithm. The total number of cores is 520 and the powercap is set at 2500W. Job impulses represent jobs starting at a certain instant of time, with the number of requested cores reflected by the height of the impulse.

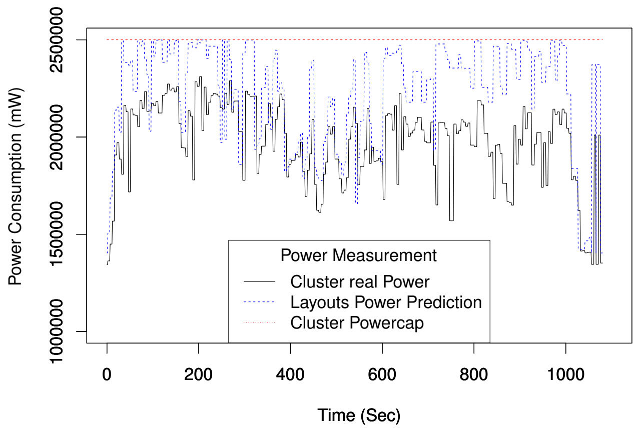

Figure 11. Power consumption profile of the ARM-based platform for the PAS algorithm. The total number of cores is 520 and the powercap is set at 2500W.

power consumption of the computing resources, the E-PAS algorithm ultimately manages to save power that can be used to execute a larger number of jobs, finalizing the same workload in less time.

This is depicted in a clearer way in Figures 6 and 8 for the Intel case, where we illustrate the cluster power consumption profile during the execution of the workload, respectively for the PAS and the E-PAS algorithm. Again, Figures 11 and 13 depict the same scenario for the ARM architecture. The dotted straight line represents the stable power cap (that is 4200W for the Intel platform and 2500W for the ARM platform), while the varying dotted and solid lines are respectively the estimated and the real power consumption of the system. Bearing in mind that our goal is to utilize as much power as possible without violating the defined power cap, we can clearly observe how E-PAS outperforms PAS in all experiments. Specifically, we note that the power consumption in Figures 6 and 11 remains

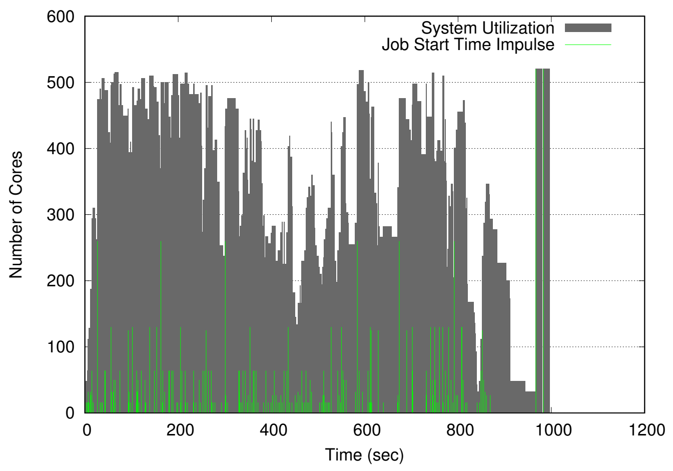

Figure 12. System Utilization of the ARM-based platform for the E-PAS algorithm. The total number of cores is 520 and the powercap is set at 2500W. Job impulses represent jobs starting at a certain instant of time, with the number of requested cores reflected by the height of the impulse.

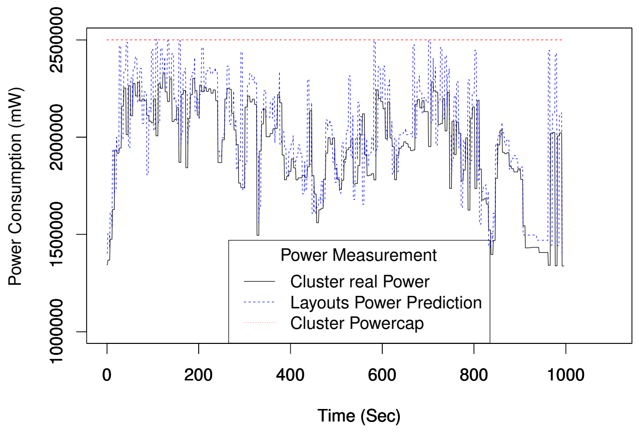

Figure 13. Power consumption profile of the ARM-based platform for the E-PAS algorithm. The total number of cores is 520 and the powercap is set at 2500W.

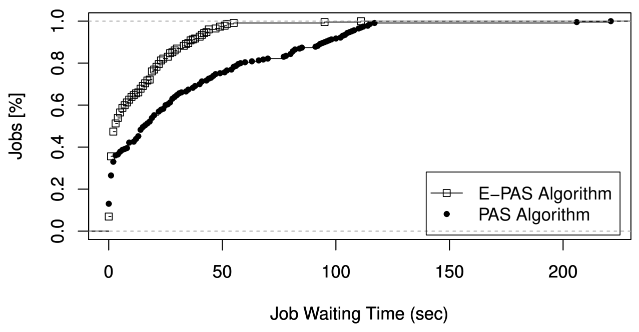

Figure 14. Cumulative distribution function of the job waiting time on the ARM-based platform.

relatively low, resulting in a big gap between the real power consumption and the maximum allowed threshold or even the estimated power profile. Conversely, as illustrated in Figures 8 and 13, the E-PAS algorithm allows for a better reduction of the gap between the real power consumption and the estimated one, meaning that more power is made available for job submissions, consequently speeding up the total workload execution (e.g., E-PAS allows for approximately 15% gain in total workload turnaround time compared to PAS for the Intel case; the same gain is around 10% for the ARM case). Furthermore, due to its faster workload execution, E-PAS uses approximately 9% less energy than the PAS algorithm in the ARM case (3.2% in the Intel case).

Finally, Figures 9 and 14 show the cumulative distribution function of the job waiting times for both algorithms, respectively on the Intel and on the ARM platform. Similarly to the previously analyzed results, also in this case we acknowledge the benefits introduced by the E-PAS algorithm, which allows for a drastic reduction of the average job waiting as opposed to PAS. Specifically, we observe an average reduction of waiting time of approximately 38% when using E-PAS on the ARM platform. For the Intel case, the same reduction gain amounts at approximately 56%, meaning that, when employing E-PAS, jobs will have to wait on average approximately half the time they have to wait when employing PAS. This is once more due to the fact that realistic power data allows for more accurate job power requirements, facilitating their executions and a faster emptying of the scheduler waiting queue.

6. Conclusions and Future Works

In this work we developed a new power-aware algorithmic mechanism named E-PAS and we integrated it within the popular job scheduler SLURM. The E-PAS algorithm is capable of efficiently improving the system resource utilization and significantly reducing job waiting times by redistributing the system power under strict powercap regime. E-PAS extends the previously conceived PAS algorithm by performing power aware optimizations on the basis of real power monitoring data and has been evaluated on both Intel and ARM architectures.

Experimental assessments with synthetic workloads validate the cross-architecture effectiveness of the E-PAS algorithm and proved its superior performance over PAS in terms of increased resource utilization, efficient system power usage and reduced job waiting times. Future works will deal with further improvements of the algorithm by taking into consideration high resolution monitoring infrastructures such as HDEEM [31] and additional integrated powercap mechanisms such as the one offered by NVIDIA GPUs. Further investigations will also deal with potential integrations of the E-PAS algorithm with the GEOPM runtime engine [18] and the development of dynamic power cap adjustments. The current E-PAS implementation will be pushed in a future official version of SLURM.

Acknowledgments

The research leading to these results has received funding from the European Communitys Seventh Framework Programme (FP7/2007-2013) under the Mont-Blanc Project (http://www.montblanc-project.eu), grant agreement n 610402.

References

[1] “Top500. The List,” https://www.top500.org/, last accessed: June 2017.

[2] A. Shehabi *et al.*, “United States data center energy usage report 2016,” Lawrence Berkeley National Laboratory, Tech. Rep., June 2016.

[3] T. Patki, D. K. Lowenthal, B. L. Rountree, M. Schulz, and B. R. de Supinski, “Economic viability of hardware overprovisioning in power-constrained high performance computing,” in *4th International Workshop on Energy Efficient Supercomputing, E2SC@SC 2016, Salt Lake City, UT, USA, November 14, 2016*, 2016, pp. 8–15.

[4] A. B. Yoo, M. A. Jette, and M. Grondona, “SLURM: simple linux utility for resource management,” in *Job Scheduling Strategies for Parallel Processing, 9th International Workshop, JSSPP 2003, Seattle, WA, USA, June 24, 2003, Revised Papers*, 2003, pp. 44–60.

[5] Y. Georgiou, D. Glesser, and D. Trystram, “Adaptive resource and job management for limited power consumption,” in *IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW) 2015*, Los Alamitos, CA, USA, 2015, pp. 863–870.

[6] N. J. Bates, G. Ghatikar, G. Abdulla, G. A. Koenig, S. Bhalachandra, M. Sheikhalishahi, T. Patki, B. Rountree, and S. W. Poole, “Electrical grid and supercomputing centers: An investigative analysis of emerging opportunities and challenges,” *Informatik Spektrum*, 2015.

[7] T. Patki, N. J. Bates, G. Ghatikar, A. Clausen, S. Klingert, G. Abdulla, and M. Sheikhalishahi, “Supercomputing centers and electricity service providers: A geographically distributed perspective on demand management in europe and the united states,” in *International Supercomputing Conference (ISC)*, 2016.

[8] T. Patki, D. K. Lowenthal, A. Sasidharan, M. Maiterth, B. L. Rountree, M. Schulz, and B. R. de Supinski, “Practical resource management in power-constrained, high performance computing,” in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC ’15, 2015, pp. 121–132.

[9] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, “BSLD threshold driven power management policy for HPC centers,” in *IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW) 2010*, Atlanta, Georgia, USA, 19-23 April 2010, pp. 1–8.

[10] ——, “Optimizing job performance under a given power constraint in HPC centers,” in *IEEE International Conference on Green Computing 2010*, Chicago, Illinois, USA, 15-18 August 2010, pp. 257–267.

[11] ——, “Parallel job scheduling for power constrained HPC systems,” *Parallel Computing*, vol. 32, no. 12, pp. 615–630, December 2012.

[12] G. A. Geronimo, J. Werner, R. Weingartner, C. B. Westphall, and C. M. Westphall, “Provisioning, resource allocation, and DVFS in green clouds,” *IJANS*, 2014.

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

[14] 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*, ser. SC ’14, 2014, pp. 807–818.

[15] N. Gholkar, F. Mueller, and B. Rountree, "Power tuning HPC jobs on power-constrained systems," in *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, ser. PACT '16, 2016, pp. 179–191.

[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 *IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW) 2012*, Shanghai, China, 21-25 May 2012, pp. 947–953.

[17] S. Reda, R. Cochran, and A. K. Coskun, "Adaptive power capping for servers with multithreaded workloads," *IEEE Micro*, vol. 32, no. 5, pp. 64–75, September-October 2012.

[18] J. Eastep, S. Sylvester, C. Cantalupo, B. Geltz, F. Ardanaz, A. Al-Rawi, K. Livingston, F. Keceli, M. Maiterth, and S. Jana, "Global extensible open power manager: A vehicle for HPC community collaboration on co-designed energy management solutions," in *High Performance Computing - 32nd International Conference, ISC High Performance 2017, Frankfurt, Germany, June 18-22, 2017, Proceedings*, 2017, pp. 394–412.

[19] D. A. Ellsworth, T. Patki, M. Schulz, B. Rountree, and A. D. Malony, "A unified platform for exploring power management strategies," in *4th International Workshop on Energy Efficient Supercomputing, E2SC@SC 2016, Salt Lake City, UT, USA, November 14, 2016*, 2016, pp. 24–30.

[20] S. Wallace, X. Yang, V. Vishwanath, W. E. Allcock, S. Coghlan, M. E. Papka, and Z. Lan, "A data driven scheduling approach for power management on hpc systems," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2016, pp. 656–666.

[21] P. Dutot, Y. Georgiou, D. Glesser, L. Lefèvre, M. Poquet, and I. Raïs, "Towards energy budget control in HPC," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017, Madrid, Spain, May 14-17, 2017*, 2017, pp. 381–390.

[22] "The Mont-Blanc Project," http://www.montblanc-project.eu/, last accessed: June 2017.

[23] N. Rajovic *et al.*, "The Mont-Blanc prototype: An alternative approach for HPC systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16, 2016, pp. 38:1–38:12.

[24] D. Tafani and A. Auweter, "Deliverable 5.8: Prototype demonstration of energy monitoring tools on a system with multiple ARM board," The Mont-Blanc Project, Tech. Rep., September 2014.

[25] Y. Georgiou *et al.*, "Power adaptive scheduling," The Slurm User Group Meeting 2015, Washington DC, USA, 15-16 September 2015, last accessed: June 2017. [Online]. Available: https://slurm.schedmd.com/SLUG15/Power_Adaptive_final.pdf

[26] M. Hautreaux, "Slurm Layouts Framework," The Slurm User Group Meeting 2015, Washington DC, USA, 15-16 September 2015, last accessed: January 2017. [Online]. Available: https://slurm.schedmd.com/SLUG15/slurm_layouts_framework.pdf

[27] M. Jette, "Slurm power management support," The Slurm User Group Meeting 2016, Athens, Greece, 26-27 September 2015, last accessed: June 2017. [Online]. Available: https://slurm.schedmd.com/SLUG15/Power_mgmt.pdf

[28] A. T. Wong, L. Oliker, W. T. C. Kramer, T. L. Kaltz, and D. H. Bailey, "Esp: A system utilization benchmark," in *Supercomputing, ACM/IEEE 2000 Conference*, Nov 2000, pp. 15–15.

[29] W. T. C. Kramer, "Percu: A holistic method for evaluating high performance computing systems," Ph.D. dissertation, Berkeley, CA, USA, 2008, aAI3388282.

[30] Y. Georgiou and M. Hautreux, "Evaluating scalability and efficiency of the resource and job management system on large HPC clusters," in *Job Scheduling Strategies for Parallel Processing, 16th International Workshop, JSSPP 2012, Shanghai, China, May 25, 2012. Revised Selected Papers*, 2012, pp. 134–156.

[31] T. Ilsche *et al.*, "Power measurement techniques for energy-efficient computing: Reconciling scalability, resolution, and accuracy," in *2nd Workshop on Energy-Aware High Performance Computing, EnA-HPC, Frankfurt, Germany, June 22, 2017*, 2017.