Power Efficient Job Scheduling by Predicting the Impact of Processor Manufacturing Variability

Dimitrios Chasapis
Barcelona Supercomputing Center & Polytechnic University of Catalonia
dimitrios.chasapis@bsc.es

Miquel Moretó
Barcelona Supercomputing Center & Polytechnic University of Catalonia
miquel.moreto@bsc.es

Martin Schulz
Technical University of Munich
schulzm@in.tum.de

Barry Rountree
Lawrence Livermore National Laboratory
rountree4@llnl.gov

Mateo Valero
Barcelona Supercomputing Center & Polytechnic University of Catalonia
mateo.valero@bsc.es

Marc Casas
Barcelona Supercomputing Center & Polytechnic University of Catalonia
marc.casas@bsc.es

ABSTRACT

Modern CPUs suffer from performance and power consumption variability due to the manufacturing process. As a result, systems that do not consider such variability caused by manufacturing issues lead to performance degradations and wasted power. In order to avoid such negative impact, users and system administrators must actively counteract any manufacturing variability.

In this work we show that parallel systems benefit from taking into account the consequences of manufacturing variability when making scheduling decisions at the job scheduler level. We also show that it is possible to predict the impact of this variability on specific applications by using variability-aware power prediction models. Based on these power models, we propose two job scheduling policies that consider the effects of manufacturing variability for each application and that ensure that power consumption stays under a system-wide power budget. We evaluate our policies under different power budgets and traffic scenarios, consisting of both single- and multi-node parallel applications, utilizing up to 4096 cores in total. We demonstrate that they decrease job turnaround time, compared to contemporary scheduling policies used on production clusters, up to 31% while saving up to 5.5% energy.

CCS CONCEPTS

• **Computer systems organization** → **Parallel architectures**; • **Hardware** → **Power estimation and optimization**;

KEYWORDS

Manufacturing Variability, Energy Efficient, Power Prediction, Job Scheduling, HPC

ICS '19, June 26–28, 2019, Phoenix, AZ, USA

ACM Reference Format:
Dimitrios Chasapis, Miquel Moretó, Martin Schulz, Barry Rountree, Mateo Valero, and Marc Casas. 2019. Power Efficient Job Scheduling by Predicting the Impact of Processor Manufacturing Variability. In *2019 International Conference on Supercomputing (ICS '19), June 26–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3330345.3330372

1 INTRODUCTION

Power is becoming a major financial and environmental concern, restricting the compute capacity of High Performance Computing (HPC) systems. Today's most power efficient supercomputer operates at 14.1GFLOPS/W [1], but even if we had a system able to operate at the 50GFLOPS/W rate, which is the limit that some funding agencies have set up for building an exascale machine, the full system would consume several tens of MWatts of power, which constitutes a large economic burden. Consequently, a report from the US Department of Energy (DOE) [52] identifies energy efficiency as one of the top ten research challenges on the road to exascale. For similar reasons, the European Union has set up an HPC program low-power systems based on mobile technology [45].

An emerging design practice for HPC systems, known as *overprovisioning* [43], is to have more nodes than the maximum power budget could feed if run at peak capacity, in contrast to traditional approaches, which are focused in having enough power even when all nodes run at their peak. Overprovisioning is driven by the observation that most applications in practice never reach peak power and hence do not fully utilize the available power envelope. In such *overprovisioned* systems, we can lower the average power provisioned to each node, allowing us to power more nodes within the same power budget. This approach is made possible by recent developments in hardware design that enable power management and power capping from user space, as this is necessary to efficiently manage power as a limited and shared resource. As an alternative to restricting power to nodes, we can choose to only operate fewer but at full power. Their total power consumption should not exceed that of the total power budget. This second approach restricts the available

parallelism in the system, but allows for faster execution and does not force us to deal with any complications related slower then expected execution of the system's workload. Which approach is preferable (or a combination of both) should depend on system and workload characteristics. For example, whether the workload would benefit from extra processing units or limitations in the completion time set by the user or administrator.

Workloads at the HPC system level are managed by job schedulers that allocate resources to dispatched jobs. Such jobs can run on distributed memory scenarios and, in this context, MPI [41] is the most common approach to handle distributed memory communications. It is usually coupled with a shared memory programming model, like OpenMP [42] or similar [3]. Either across nodes or within a shared-memory node, both job and runtime schedulers deal with the resource allocation problem, albeit at different levels, offering opportunities to manage power consumption. Indeed, examples of power-aware systems that offer solutions either at the job scheduling [12, 16, 19, 22] or at runtime system [13, 22, 27, 53, 54] levels already exist in the literature.

Manufacturing variability or process variation refers to the power and frequency heterogeneity observed across chips implementing the exact same architecture as a consequence of uncontrollable material differences in the manufacturing process [46]. In order to provide homogeneous performance, chips of the same architecture must hide frequency variability, which can only be achieved via variations in their power consumption. However, in a power constrained environment where all chips need to operate under a certain power cap, this frequency variability can no longer be hidden [46], leading to heterogeneous performance. As a result, a theoretically homogeneous system turns into a heterogeneous one with performance variations of up to 64% [27]. While ignoring this manufacturing variability leads to performance and energy inefficiencies, there are opportunities for achieving improvements at the power budgeting or parallel runtime system levels when variability is properly managed [13, 22, 27, 53, 54]. Moreover, manufacturing variability makes hardware imposed bounds more difficult to deal with, since it has varying and difficult to predict impact on performance. A system originally considered homogeneous in terms of performance, is now an heterogeneous one, after imposing hardware power bounds.

This paper goes beyond the state-of-the-art by proposing job scheduling policies driven by variability-aware power prediction models. We extend power-aware scheduling and power prediction models to deal with manufacturing variabity, producing two novel variability- and power-aware job scheduling policies. We consider the power consumption of the CPU, since it accounts for more than 50% [15, 24] of the total node's power consumption. Our Policies rely on two different models and leverage their power requirement predictions of individual parallel jobs to make scheduling decisions that maximize performance while reducing energy consumption. Many different variability-agnostic power and energy prediction models have been proposed [6, 7, 10, 23, 29] and are often employed

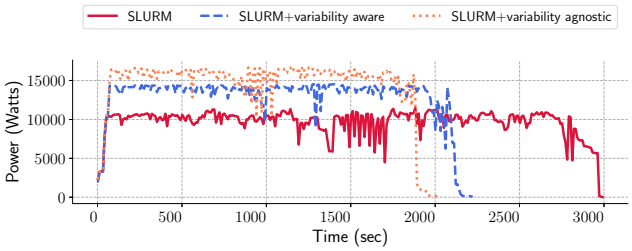

Figure 1: Total power consumption trace for Conservative, Variability agnostic and Variability Aware scheduling policies, when running the same workload. Considering socket variability maximizes performance and meets the power budget.

to manage power distribution on clusters to mitigate the effects of manufacturing variability [4, 13, 16, 22, 27, 53, 54]. In our approach, we do not consider imposing per node hardware power bounds, in order to avoid the complications of a performance heterogeneous system. Instead, we use power prediction models to guide job scheduling and maximize the system's utilization for a given system-wide power budget. Hardware power bounds can be used in our approach however a safety measure, in case of inaccurate predictions.

As a motivation we show Figure 1, where three different scheduling policies are compared. The Conservative simply considers that all jobs consume the same power on all sockets. The Variability agnostic predicts accurately the power consumption of individual jobs, but does not consider socket variability. On the contrary, the Variability Aware policy does also consider socket variability, making a different prediction per socket. As displayed in Figure 1, the Conservative policy is the one providing the worse performance. The Variability Agnostic improves performance by making more accurate predictions but fails to account for the more power consuming sockets, exceeding the 15KWatts power budget. Finally, the Variability Aware policy manages to improve performance, while respecting the power budget. Figure 1 illustrates how accounting for manufacturing variability while scheduling parallel jobs provides performance benefits. Section 4.1 describes the experimental setup we consider to generate Figure 1.

This paper shows how variability-aware power prediction models can be effectively used to guide job scheduling policies and bring significant benefits with respect to the variability-agnostic ones. In particular, this paper makes the following contributions beyond the state-of-the-art:

- Two new variability-aware power prediction models. Both models use Performance Monitoring Counters (PMC) to predict an application's power consumption on a specific socket. PMCs are used to measure the activity of individual architectural components while the targeted application is running and a linear model is then used to find their contribution to power consumption. The first model assumes power variability to impact all applications equally. It uses a single benchmark to measure the power consumption variability

Table 1: Architectural component activity ratios formulas for Intel Broadwell Architecture, inferred from Intel's 64 and IA-32 Architectures Software Developer's Manual [28]

Power Component	Component Activity Formula
Fetch	UOPS_RETIRED.ALL / CPU_CLK_UNHALTED.THREAD_P
Branch Prediction Unit	BR_INST_RETIRED.ALL_BRANCHES / CPU_CLK_UNHALTED.THREAD_P
Arithmetic & Logic Unit	(UOPS_DISPATCHED.PORT_0 + UOPS_DISPATCHED.PORT_1 + UOPS_DISPATCHED.PORT_5) / CPU_CLK_UNHALTED.THREAD_P
Floating Point	FP_COMP_OPS_EXE.X87 / CPU_CLK_UNHALTED.THREAD_P
L1 cache	L1D_ALL.REF / CPU_CLK_UNHALTED.THREAD_P
L2 cache	(L2_RQSTS.ALL_RFO + L2_RQSTS.ALL_DEMAND_DATA_RD) / CPU_CLK_UNHALTED.THREAD_P
L3 cache	LLC.References / CPU_CLK_UNHALTED.THREAD_P
Memory	LLC.Misses / CPU_CLK_UNHALTED.THREAD_P

across sockets and apply it to the Variability Agnostic PMC-based model. The second model extends the PMC-based approach to take power consumption variability into account, as part of the model. It trains the model for each individual socket, using a reduced set of benchmarks.

- Two power- and variability-aware job scheduling policies that optimize job turnaround time and energy efficiency while respecting a system-wide power budget. Unlike previous work that does not consider variability during job scheduling decisions [16, 22, 27, 53], our policies use variability-aware predicton model to guide scheduling.
- A complete evaluation of the two variability-aware policies via a discrete event simulator. We implement additional scheduling policies for our evaluation, which represent traditional and state-of-the-art practices used in today's HPC systems. Our evaluation demonstrates how variability-aware policies achieve energy savings up to 5.5% and job turnaround time reductions up to 31%, considering different power budgets and two workload traffic scenarios (bursty and heavy).

The remainder of this paper is organized as follows. Section 2 presents the two variability-aware power prediction models. Section 3 introduces the variability and power-aware scheduling policies we propose. A validation of the models and evaluation of our job scheduling policies are presented in Section 4. Section 5 discusses the related work on power prediction and power-aware scheduling and, finally, Section 6 provides concluding remarks.

2 POWER VARIABILITY PREDICTION MODELS

Power modeling has received a lot of attention from researchers and developers as it provides a quick and robust way to understand the power behavior of a system. A common approach for predicting power consumption consists in the usage of Performance Monitoring Counters (PMC) [5, 7, 9, 11, 36, 51], since sampling PMC does not introduce significant power interference [29, 32] and PMC-based prediction models decompose a chip into several components in terms of power consumption [7]. While power prediction models are employed to find the best tradeoff between power and performance [22], we are not aware of any previous work that uses variability-aware power models to guide job scheduling decisions.

2.1 Power Ratio Model

Our first (baseline) model attempts to circumvent the relative complexity of dealing with manufacturing variability by assuming that all applications are impacted the same by it. It uses a PMC-based model to predict an application's power consumption, without considering variability. Then it uses a single benchmark, in our case an OpenMP implementation of the *cholesky*[1] decomposition of a dense 65 MB matrix, and measures the average power consumption on each socket we want to generate a model for. Once this information is obtained, we characterize the variability between sockets in terms of power ratios and we apply them to power prediction of a target application (denoted as *app*), made by using the PMC profiles obtained from an execution on a single reference socket (denoted as $socket_{ref}$).

PMC values capture the contribution of each chip's architectural component to power consumption, by modeling their usage as activity ratios. These ratios are defined as the number of retired micro operations relevant to the targeted architectural component per active cycle. For example, for main memory and caches, activity ratios are the number of references or misses per cycle, respectively. The model assumes that a component's contribution to power consumption is proportionate to its usage. The granularity at which we can decompose a chip into architectural components depends on the underlying architecture and the available PMC. Table 1 shows the different components and their corresponding PMC formulas for the Intel's Broadwell architecture.

Figure 2 shows power and activity ratio profiles for the active cores (CORES), fetch unit (FE) and L1 cache, for the *blackscholes*, *bodytrack*, *lu-mz_C.16* and *sp-mz_D.8* parallel codes. For multi-node applications, *lu-mz_C.16* and *sp-mz_D.8*, we show the activity ratios and power consumption of one of the processes, running on one of the sockets.

Each process has its own set of activity ratios that result in individual predictions, per socket that it run on (see Section 4.1 for experimental setup details). All applications have

[1] We chose *cholesky* because it is a dense linear algebra kernel, which stresses both CPU and cache memory accounting for power consumption variability across the sockets.

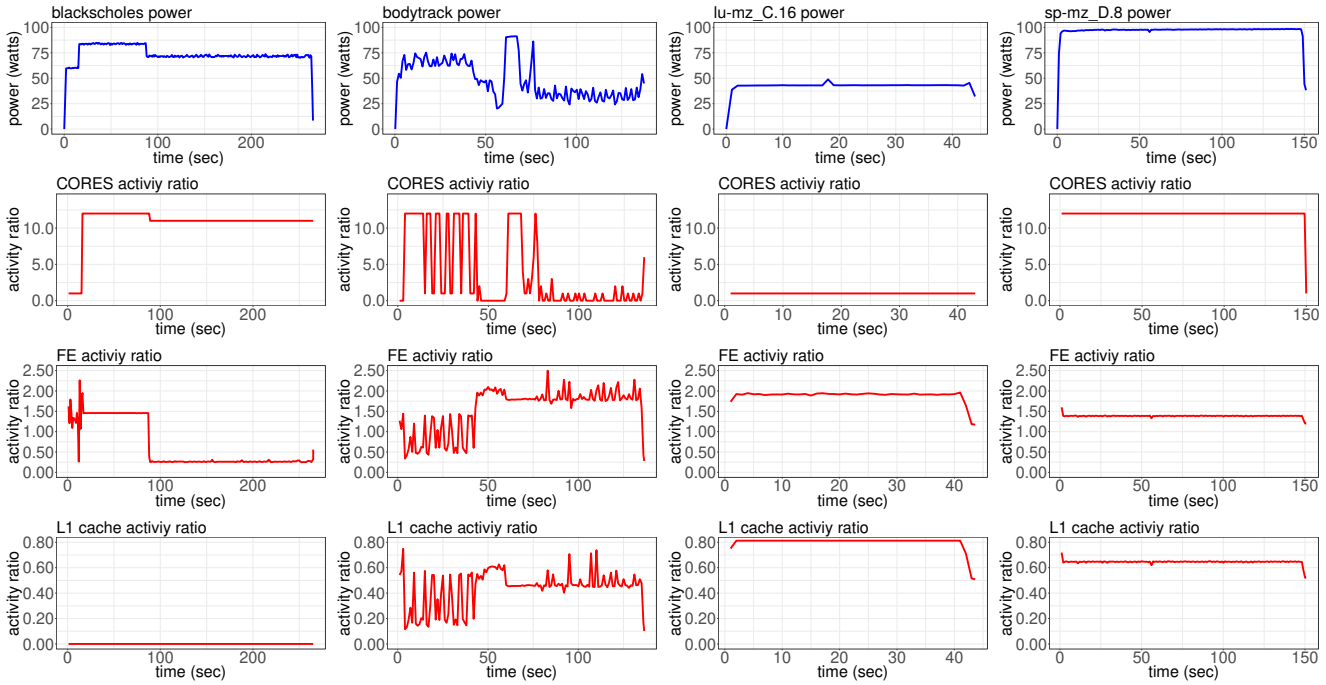

Figure 2: Power, active cores and component activity ratio traces when running on 12 cores of a single socket. Architectural components shown are the fetch unit (FE) and L1 cache. The activity ratios are the number of retired micro operations per unhalted cycle, relevant to each architectural component. In the case of cores, activity ratio is the number of active cores. For memory the activity ratio is measured as the number of references (for caches) or LLC misses (for main memory) per cycle. For multi-node applications, we show the activity on the socket running one of the MPI processes. PMC data is collected for all the processes and individual predictions are made for each socket.

a unique power profile that is the result of the different component activity ratios. For example, *blackscholes* and *sp-mz_D.8* have high CORES activity that contribute to the power consumption. However, *blackscholes* has minimal activity in L1 cache, which results in lower overall power consumption, when compared to *sp-mz_D.8*. Moreover, *lu-mz_C.16* has similar activity ratios to *sp-mz_D.8* but significantly lower activity in cores (only uses 1 core per process), which results in lower power consumption. Finally, we can observe how changes in component activity influences power consumption in the cases of *blackscholes* and *bodytrack*.

We then align the activity ratios with measured power data, which allows us to express the power of an application on a particular socket as:

$$P = AC * W_{cores} + \sum_{c=1}^{N_{comp}} (AR_c * W_c) \quad (1)$$

where AR_i is the activity ratio of power component i, AC is the average number of active cores and P is the power consumption, at a given moment. These values are known for any given application in our training set. However, we need to find the contribution of each component to the total power consumption P. This is expressed using a set of weights, denoted as W_i for architectural component i and W_{cores} for active cores.

We determine the weights using a training stage during which we monitor power along with the architectural component activity for a small set of kernels. The choice of training benchmarks should reflect different application behaviors (e.g., computation vs. memory bound) and stress different architectural components, such as integer or floating point units and the different memory levels. The list of applications used for training can be found in Table 2. To better account for the power contribution of the memory subsystem, this list includes an additional synthetic code, *mem_bench*, a microbenchmark that causes misses on different levels of the memory hierarchy.

2.2 Variability-Trained Prediction Model

With the data measured in this training stage, we then use linear regression to determine the values of W_i and W_{cores} that best fit in Equation 1 on a given socket. The resulting linear model is socket-specific and can predict the power consumption of a generic application assuming its activity ratios for all architectural components and cores are known.

We obtain the values of W_i and W_{cores} for a specific socket we use as reference. To account for manufacturing variability

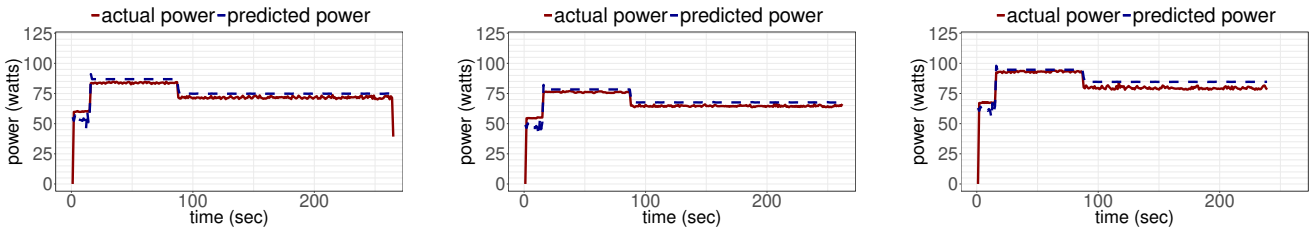

Figure 3: Actual and predicted power consumption, using the PMC-based (Optimized PMC) model, for *blackscholes* under three distinct sockets. Same CPU chip model is mounted on all three sockets, utilizing all 12 available cores.

Table 2: Benchmark training set for PMC-based power prediction model.

Benchmark	Description
cholesky	Floating point cholesky factorazation kernel. Latency-bound.
knn	K-nearest neighbours kernel. CPU-bound.
matmul	Floating point matrix multiplication kernel. Latency-bound.
md5	MD5 message-digest algorithm. Memory-bound.
prk2_stencil	Tests the efficiency with which a space-invariant symmetric filter (stencil) applies to images. Memory-bound.
qr_tile	Tiled QR factorization kernel. CPU-bound.
sparseLU	Sparse LU factorization kernel. Memory-bound.
stap	Space-Time Adaptive RProcessing for radar detection of an objects position. Latency-bound.
symmatinv	Floating point symmetric matrix inversion kernel. CPU-bound.
vector-redu	Computes the sum of the integer elements of a vector. Memory-bound.
mem_bench	A micro-benchmark that stretches different memory levels.

we use the power ratios computed using the *cholesky* benchmark. The power consumption of *app* on any $socket_i$ is then obtained by the following formula:

$$P^{app}_{socket_i} = P^{app}_{socket_{ref}} * \frac{P^{choleksy}_{socket_i}}{P^{cholesky}_{socket_{ref}}} \quad (2)$$

In the case of multi-node applications, we predict the power consumption of each socket an MPI process runs on, by applying the power ratio corresponding to that socket.

Our second model does not assume the impact of manufacturing variability to be independent of the parallel code. Instead, it aims at capturing the impact of manufacturing variability on each specific application. Due to manufacturing variability, power consumption differs between sockets, which means that W_c and W_{cores} are socket-specific and obtained by solving Equation 1 individually for each socket. In terms of the activity ratio values per application, we assume them to be invariant across all sockets featuring the same architectural design. Consequently, the socket-specific Formula 1 can be extended to integrate all sockets featuring the same architectural design:

$$P^{app}_{socket_i} = AC^{app} * W^{socket_i}_{cores} + \sum_{c=1}^{N_{comp}} (AR^{app}_c * W^{socket_i}_c) \quad (3)$$

From this formula we can obtain a power prediction of an application running on any chosen socket, which is characterized in terms of its weights. For each parallel code we just need a single run on a generic socket to compute AR_c and AC, which are socket-independent as they are determined by the architectural design.

For multi-node applications, individual predictions need to be made for each socket an MPI process is run on. A corresponding model needs to be applied to each socket, which has been trained for each socket individually, producing a different set of weights. The resulting prediction is a set of predictions, equal to the number of sockets the application run on, each prediction accounting for the individual socket's own variability.

Figure 3 shows power profiles of the *blackscholes* code (solid line) running on three different sockets, together with the corresponding predicted power consumption (dashed line). Figure 3 displays how the power consumption varies up to 19% for the same application (from 76W to 96W peak power), depending on the socket it runs on. The predicted values capture the power consumption variability for all cases.

To improve the model's precision, per each application, we consider using only a subset of architectural components that provides the most accurate results. This way we mediate any biasing the training set may have. All possible combinations of architectural components are considered and a prediction error is computed for each one. We use the Mean Absolute Percentage Error (MAPE) formula for computing this prediction error

$$M = \frac{100}{n} \sum_{t=1}^{n} |\frac{A_t - F_t}{A_t}| \quad (4)$$

where A_t and F_t are the actual and predicted values for observation t, and n is the total number of observations. The model with the lowest error value is chosen for all future

predictions. This tuning process can be done offline per each targeted application using data obtained from a single parallel execution. The same optimization is also applied to the PR model. From this point on, any mention to the PR and VT models, references the optimized versions of the models (or otherwise unoptimized PR and VT). Section 4.3 shows a detailed evaluation and validation of both models.

2.3 Predicting Power for Multi-Node Applications

For multi-node applications, individual predictions need to me made for each MPI process (process with similar workloads can be treated as the same one). A corresponding model needs to be applied to each socket and process, producing a different set of weights. The result is a prediction of power consumption of each process and each socket. For example, if we have an application with N processes and a system with M sockets, then we make $N \times M$ predictions.

3 JOB SCHEDULING POLICIES

We propose two new variability-aware job scheduling policies, the *Power Ratio Variability Prediction* and the *Variability-Trained Prediction*. These schedulers make use of the power variability prediction models introduced in Section 2. With these models, the schedulers predict the job's power consumption on all available sockets and schedule the job on the most efficient one. Before scheduling the job, the scheduler checks whether the system-wide power budget would be respected once the job starts running. If this is not the case, the job will wait until other jobs finish and more power is available. In the case of multi-node jobs, we consider the total power consumption of all its processes. Our scheduler does not explicitly consider the interconnect topology. We use the default configuration of SLURM, where nodes are viewed as a one-dimensional array and node distances are set to be equal. In such configuration, inter-node communication has always the same cost. Our scheduler always allocates entire nodes to the same application to exploit fast inter-socket communication as much as possible.

Furthermore, we consider three job scheduling policies representative of the state-of-the-art and with increasing complexity: *SLURM extended*, *Power Estimation*, and *Power Estimation+Variability Aware*. The first two are variability-agnostic, while the third is variability-aware. Finally, we also consider an *Ideal Variability Prediction*, which is based on an oracle power variability predictor that knows exactly how much power a job will consume on any processor in the system. We extend the SLURM's logic [30] to implement the various power- and variability-aware job scheduling policies presented in this section. We chose SLURM as our reference because it is widely used on HPC production systems and well studied in the literature. All job scheduling policies are described in the following:

SLURM Extended: This policy implements SLURM scheduler's logic. We extend the default behavior to not exceed the global power budget, by considering the worst case scenario, which is that each job can consume the maximum power budget allowed per socket. Additionally, we extend the scheduler to initiate backfilling for power as well [44]. Typically, if a job requests more sockets than currently available, the scheduler will try to schedule a different job without causing delays. The same will happen if a job requests more power than the system can allocate.

Power Estimation (*SLURM+PE*): This policy extends further *SLURM extended*'s behavior by using a user provided estimation of a job's power consumption. For precision we obtain power profiles of previous execution of the jobs to estimate the power consumption. This is the equivalent of using a variability-agnostic prediction model. This scheduler does not consider manufacturing variability as it assumes all sockets consume the same power for a given job.

Power Estimation+Variability Aware (*SLURM+PEVA*): This policy implements elements from the state-of-the-art practices in power-aware job scheduling [22, 27]. Similarly to *SLURM+PE*, it estimates the power requirements of a job using a power trace from a previous execution. It also orders sockets and allocates first the most power efficient ones to minimize the system's net power consumption. The socket ordering is obtained by running a simple benchmark, the *cholesky* kernel, on all sockets and observing their power consumption. This approach assumes all parallel jobs to be influenced by manufacturing variability in the very same way. Moreover, a job's power estimation depends on the socket used for profiling and thus it is possible to under or over-estimate the final power.

Power Ratio Variability Prediction (*SLURM+PRVP*): This is the first new policy we propose. It relies on our Power Ratio Model, presented in Section 2.1, to guide scheduling decisions. A single power and performance profile of a given job (or one for each socket a process was run on, for multi-node jobs) is required in order to compute the activity ratios, which can be performed on any set of sockets in the system. Running the single benchmark a priori on all sockets is also required.

If the predicted power for a new job makes the system budget to go over its limit, then the job waits until resources are released. The backfilling scheme is the same as with the previous policies. This policy's framework is shown in Figure 4. The lower box shows the training stage corresponding to this policy.

Variability-Trained Prediction (*SLURM+VTP*): Our second proposed policy is similar to the *PRVP* policy, but it uses our VT prediction model (see Section 2.2). This policy requires running the training benchmark set from Table 2 on all sockets in order to train the model. The framework for this policy is described in Figure 4. The box on the upper right corner corresponds to the training process of this policy.

Ideal Variability Prediction (*SLURM+IVP*): This policy is identical to *SLURM+PRVP* and *SLURM+VTP*, but using an oracle power predictor to drive job scheduling decisions. This policy is aimed at showing the impact of using a 100% accurate model to guide scheduling decisions. Thus, *SLURM+IVP* is used for comparison purposes to show

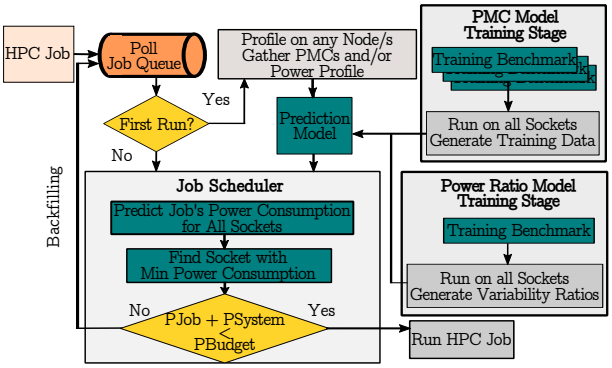

Figure 4: Framework for the *SLURM+PRVP* and *SLURM+VTP*. Training stages for the their respective models are shown in the rightmost boxes.

the maximum benefits that can be achieved by power- and variability-aware job scheduling policies.

4 METHODOLOGY EVALUATION

4.1 Experimental Setup

To evaluate the proposed models and job scheduling policies, we have access to 128 nodes of the Quartz cluster at Lawrence Livermore National Laboratory [37], which contain 4096 cores in total. We use an in-house simulator to mimic the behavior of a job scheduling system on a production platform like Quartz. For training the prediction models, we use the set of kernel and micro-benchmark applications described in Table 2. We maintain socket temperature between 38-42 °C, in order to only observe the power consumption variation relevant to manufacturing variability.

Hardwareand Simulation Platforms: The Quartz cluster [37] consists of 2634 NUMA nodes, each with two 16-core Intel Xeon E5-2695v4 sockets and equipped with 128GB of main memory.

For testing our scheduling policies, we implement a discrete event simulator [2], and implemented our scheduling policies on top of it. It requires all jobs to be first executed on the physical hardware to gather performance and power traces, which are then used to simulate their execution under different scheduling schemes. Idle power is modeled as the average power consumed by each socket, as measured on the actual hardware. Using a simulator allows us to rapidly test and evaluate new policies without any accuracy loss since the simulations are led by power traces obtained from real parallel executions.

Training and Workload Benchmark Applications: To train our models we use a set of small kernel and micro-benchmarks, listed in Table 2, that capture different behaviors. In addition to these kernels, we design a microbenchmark that stresses each level of the memory hierarchy, in order to measure the impact that each cache level has on power consumption. To evaluate the job scheduling policies considered, we use the PARSECSs benchmark suite [14] (task-based implementation of PARSEC [8]) as our set of single socket parallel jobs and the MPI+OpenMP versions of the NAS-MZ benchmarks [31] as our multi-node jobs. The PARSECSs benchmark suite consists of emerging workloads for shared memory architectures, representative of applications run on typical HPC systems. The NAS-MZ benchmarks are also a well known set of kernel applications that can run on a large set of nodes, often used in HPC. The multi-node jobs run with different configurations for 8, 16 and 64 MPI processes, where each process runs on a single socket. All instances and processes run on 16 cores, with the exception of *facesim* (8 cores), *fluidanimate* (8 cores), *lu-mz_D.8* and *lu-mz_D.8* (1 core per MPI process). The numbers at the end of the multi-node application names denote the number of processes instantiated for the given benchmark configuration. Individual application in our diverse set of benchmarks can run from a single core up to 1536 cores, for the larger MPI codes.

Cluster Workload Generation Typically, workload manager schedulers are evaluated using workload traces from the job queues of actual HPC clusters [19, 20]. However, in our case this is not applicable, since these type of traces do not contain information on power and manufacturing variability. Moreover, we are not able to create a job queue trace out of the clusters we have access to, since reading performance counters such as the RAPL interface requires root access. For these reasons we generate our own cluster workload combining single- and multi-node applications, allowing us to measure the performance and power profiles of the workload. A similar methodology is used in other power and manufacturing variability related studies [16, 44], but in our approach we use a wider number and range of applications.

We generate two random job distributions as our workload on the cluster, corresponding to bursty and heavy loads. The bursty scenario consists of 763, periodically creating a heavy load that requires a large number of sockets to be served, even exceeding the systems total capacity, having jobs wait. However, there are also time periods that the system may be idle or have only a few jobs to serve. The heavy load scenario consists of 2286 jobs, where there are always enough jobs to occupy the whole system, for 98% of the total execution (2% corresponds to initial submission when the whole system is idle and the few last jobs remaining at finalization, before all jobs complete and system returns to idle state). In the rest of this document, we use the term traffic when referring to the cluster's load.

Performance and Power Monitoring: Our methodology requires to monitor performance counters plus power consumption rates. We use *perf* version 3.10 and *mpstat* version 10.5.1 for monitoring architectural and core component activity. For measuring power and enforcing power limits we implement a daemon built on top of *libmsr* [49], which is a framework for accessing RAPL registers safely from user space. RAPL registers are known to have very good accuracy [15, 24, 25, 33]. The sample rate is 100ms for power monitoring and 1s for performance counters. Although we are able to monitor an applications real power consumption on

[2]The simulator, along with the traces used in this document, are available at https://bitbucket.org/dchasap/cwm-simulator/src/master/.

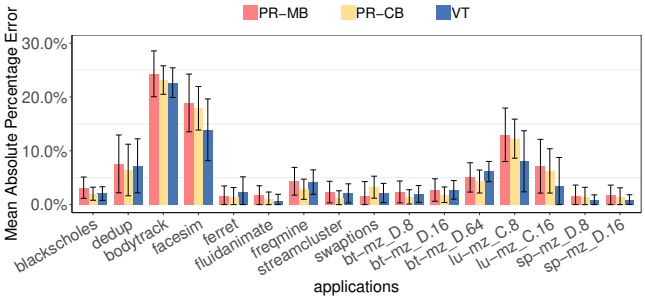

Figure 5: Comparison of average power prediction error for all models over all sockets. The error bars show the standard error deviation across all sockets for the corresponding application.

a finer grain, our predictions are limited to 1s granularity, since they depend on the performance counter data collected at the coarser granularity.

4.2 Simulator Validation

To validate the simulator we run a small scale experiment using 8 nodes (16 sockets) on Quartz. The workload we use consists of a mix of a 100 randomly chosen instances of the PARSECSs benchmarks. We run a bash script that periodically issues 10 instances (every 60s) until all the job instances are issued and wait for all jobs to finish execution. We generate a trace with the timestamp of each job's issue time and the node it was run on. We also keep track of the total time it takes for all the jobs to complete along with the total energy consumed.

We then use the simulator to repeat the experiment and reproduce the results measured on the actual machine. We run again all PARSECSs applications on all 16 sockets to gather the traces and power profiles, this time to give as input to the simulator. Then we issue the same 100 jobs again, this time on the simulator, at the same intervals as with the original run on the actual machine. The simulator will also use the workload trace to get the socket each job run on, and try to replicate the same socket to job allocation. This way, it essentially recreates the same scheduling decisions SLURM took on the actual machine. When all jobs finish execution, we measure the total execution time and energy consumption.

Comparing total execution time and energy consumption between the two experiments shows that the simulator is 1.6% slower than the actual execution on the cluster. Moreover, the jobs' total power consumption is higher on the simulator by 1.1%. These results show that our simulator has very good accuracy and the results discussed in this Section are representative of the impact our scheduling policies would have on the actual machine.

4.3 Model Validation

In this section we experimentally validate the prediction models presented in Section 2. We analyze the models in terms of Mean Absolute Percentage Error (MAPE) (see Equation 4) between predicted and real values. Figure 5 shows the MAPE values of the average power predictions for all applications over all sockets. The error bars show the standard deviation of the MAPE metric, computed over all the 256 sockets. These results correspond to the optimized versions of the Power Ratio (PR) prediction model, presented in Section 2.1, and the Variability-Trained (VT), which is presented in Section 2.2. Since the PR model incorrectly assumes that all applications are affected by manufacturing variability the same way, we show two versions of the PR. Models are denoted as PR-MB and PR-CB, using by a memory bound (*sparseLU*) and a computation bound benchmark (*cholesky*) for computing the variability ratios, respectively. Overall, all models performs well, achieving an average error below 10%. The VT model outperforms the PR models, while PR models varies depending on the benchmark used to compute the variability ratios. PR-MB consistently performs worse than PR-CB, since the memory bound benchmark detects up to 15% less variability than the computation bound one. This disparity among results for the PR model can become a more serious problem in the future, as variability is expected to increase [39]. The more robust VT model will be better suited, since it does not falsely assume that variability is application independent. The unoptimized versions (see Section 2) of the same models reach up to a 16% error (results not shown).

Our results show that all three models are able to predict the power consumption variability in most cases. However, it is possible to misspredict if an application's behavior is not well represented by the benchmarks used for training. Two such cases are *bodytrack* and *facesim*, that although they are effectively using more than 8 cores, their power consumption remains below 60 Watts.

4.4 Variability-Aware Scheduling Evaluation

In this section we evaluate the two novel scheduling policies proposed in this paper, *SLURM+PRVP* and *SLURM+VTP*, which are presented in Section 3. We compare them to four other scheduling policies, *SLURM extended*, *SLURM+PE*, *SLURM+PEVA* and *SLURM+IVP*, which are also described in 3. *SLURM extended* policy implements SLURM's logic in our simulator, with the addition of power-awareness and power backfilling. *SLURM+PE* and *SLURM+PEVA* employ state-of-the-art features of proposed power-aware policies [12, 22, 43], while *SLURM+IVP* demonstrates the ideal scenario, where prediction is 100% accurate. The evaluation is done based on a simulator, as described in Section 4.1, by feeding it performance and power traces from actual executions on the Quartz cluster. For our experiments we generate random job workloads composed of nine applications from the PARSECSs suite and seven multi-node jobs from NAS-MZ, simulating both bursty and heavy traffic scenarios (see Section 4.1). The main objective of each policy is to improve the cluster's performance and energy consumption, while exceeding a global power budget. All policies treat power as a limited

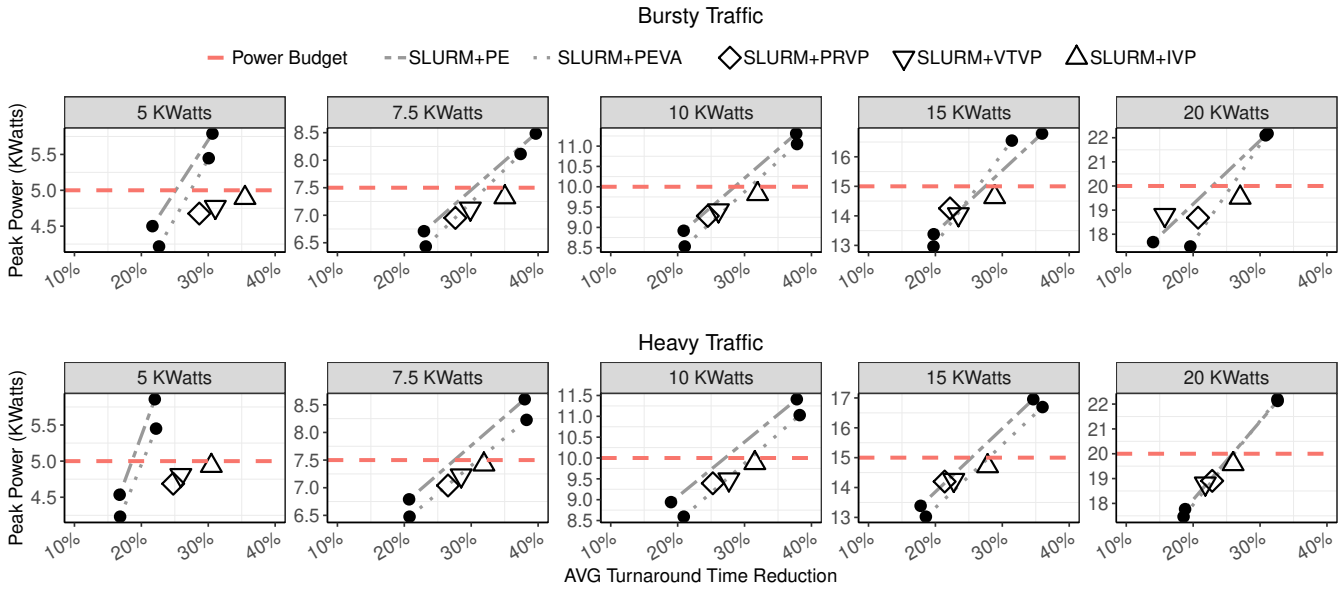

Figure 6: Power and Average turnaround time reduction over SLURM extended. Results SLURM+PE and SLURM+PEVA values range as shown by the corresponding lines, depending on the estimation provided.

resource and depending on a predicted or estimated power peak for each job, they restrict the number of running jobs to only those that can be accommodated by the given power budget.

Figure 6 compares the different policies in terms of average job turnaround time reduction (x axis) and their maximum power consumption (y axis). The turnaround time reduction shown in percentages on x axis is over the *SLURM extended* policy, for the corresponding power budget and traffic. We define job turnaround time as the time a job waits to be scheduled, including scheduler overhead, plus its execution time. The average job turnaround time is computed as the sum of turnaround time of all jobs, divided by the number these jobs. Results consider five different system-wide power budgets, 5K, 7.5K, 10K, 15K and 20KWatts and the bursty and heavy traffic scenarios described in Section 4.1. Our workloads require 25K Watts to run using the whole cluster without restricting any jobs. In the 5KW case, the *SLURM extended* is forced to drop jobs that use 64 sockets, since it estimates that these jobs require more power than available to the system. The minimum budget that allows *SLURM extended* to run jobs that demand 64 sockets is 7.5KW. Since the case of 5KW *SLURM extended* runs a lighter load (dropped 64 socket jobs), results are slightly biased towards *SLURM extended* (just for the 5KW case).

SLURM+PRVP, *SLURM+VTP* and *SLURM+IVP* policies are denoted with different symbols. The two considered traffic scenarios produce similar results. *SLURM+PRVP* performing slightly better, reflecting the better precision of the underlying prediction model. An exception is the case of a 20KWatts power budget, where *SLURM+VTP* performs marginally better than *SLURM+PRVP* by 2%. In this case, almost all jobs can be run without exceeding the power budget, thus rendering the improved precision of *SLURM+PRVP* irrelevant. The lowest reduction in terms of job turnaround time, 15%, is observed in the case of the bursty traffic scenario, under a power budget of 20KWatts, for the *SLURM+VTP* policy. The largest improvement, 31%, is obtained by the *SLURM+VTP* policy when managing the bursty traffic under a power budget of 5KWatts. The benefits of the *SLURM+PRVP*, and the *SLURM+VTP* policies reach an average of 25%, considering all power budgets and traffic scenarios. *SLURM+IVP*, which is the ideal scenario, performs slightly better than *SLURM+PRVP* and *SLURM+VTP* under all power budgets, although the benefits achieved by our two proposed policies are very close to the best possible scheduling. *SLURM+PE* and *SLURM+PEVA* are shown as lines, where their performance and maximum power consumption can lie on any point on the corresponding line. This is because unlike our proposed policies, which use prediction models to compute the power consumption of each job on each socket, the power-estimation-based policies, *SLURM+PE* and *SLURM+PEVA*, use a single power estimation obtained from power profiling on a single socket. Due to the power variability of each socket, it is possible that the estimation varies according to the variability of the socket used for profiling. This variance impacts the policies' efficiency.

Underestimating power by using a power efficient socket allows more sockets to get allocated, but the net power consumption of the system can exceed the system-wide power budget. Contrary, overestimating power can lead to significant performance degradation, since the policy becomes more conservative, underutilizing the available power budget. Using a socket with moderate power consumption to get the

power trace used for the estimation can achieve comparable results to *SLURM+PRVP*, *SLURM+VTP*. However, all points on the lines showing the range of possible results for *SLURM+PE* and *SLURM+PEVA* show worse reduction than *SLURM+PRVP*, *SLURM+VTP* and *SLURM+IVP*, since variability is not considered.

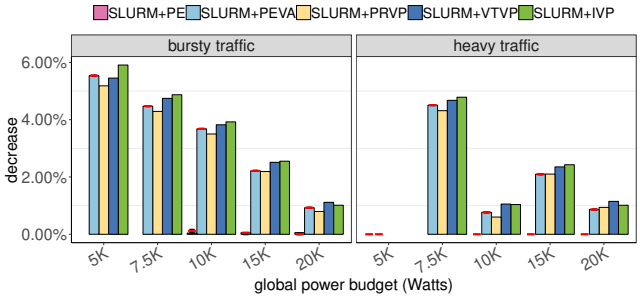

Figure 7: Energy consumption Reduction over *SLURM extended*, under different budgets and traffic scenarios. SLURM+PE, which is Variability Agnostic fail to save any energy, while SLURM+VTVP performs best.

Figure 7 shows the reduction in the system-wide energy consumption. Both traffic scenarios' benefits increase as the system-wide power budget is reduced. When less power is available, considering variability is important for energy saving, since we can choose to always use the most power efficient sockets. Note that under heavy traffic, the 5K scenario offers no benefit. This is because a significant number of multi-node jobs are dropped by the *SLURM extended* since they appear to require more power than available to the system. As a result, *SLURM extended* runs a lighter workload than he rest of the policies. This also happens in the bursty scenario, but since the workload only contains a few 64 socket jobs that are dropped, we still observe significant benefits. *SLURM+PE*, which is variability-agnostic, offers no benefit over the *SLURM extended* policy. *SLURM+PEVA*, which prioritizes allocation of power efficient sockets, matches the energy savings of the *SLURM+PRVP* and *SLURM+VTP* policies. Accounting for power variability can have a significant impact on energy efficiency reaching 5.5% on the most energy-restricted scenarios.

Results shown across Section 4 prove that our proposed policies, *SLURM+PRVP* and *SLURM+VTP*, can improve energy efficiency up to 5.5% (3% on average) over simple solutions commonly used. Moreover, job turnaround time is reduced up to 31% (25% on average). Compared to the *SLURM+PEVA* policy, which is variability-aware, our method is more robust as none of the *SLURM+PE* and *SLURM+PEVA* policies can guarantee that the system-wide power budgets are respected.

5 RELATED WORK

In this Section we present the state-of-the-art approaches on power prediction models and power management in HPC clusters. We compare the related work to our own and discuss how they differ.

5.1 Power Prediction Models

Power prediction models have been extensively studied over the years. In particular, models based on PMC have been very successful in predicting power consumption [6, 7, 10, 23, 29]. Our PMC-based model is based on the work of Bertran et al. [6, 7], which aims at providing insight into the way individual architectural components influence power consumption. Our PMC-based model extends Bertran's model to account for manufacturing variability in terms of power consumption. The original model requires carefully crafting micro-benchmarks that isolate activity per architectural component, in order to train it. We show that comparable results can be obtained by training the model with a small set of kernel applications and a microbenchmark that stresses the memory unit.

5.2 Power-Aware Budgeting and Scheduling

Managing power has become an important issue in HPC. A survey on the techniques developed in nine of the TOP500 HPC centers for improving energy efficient is presented by Maiterth et al. [38]. They identify several emerging techniques, some with common characteristics. Over-provisioning [48] considers building a system where it is not possible to run all the nodes at full capacity. Instead, the system operates under a certain power budget and dynamically distributes the available power among nodes. Nodes can operate under different power caps. For example, a few nodes may operate at full capacity, while the rest are disabled or constrained. Other approaches [40, 44] take advantage of applications that can be considered *moldable*, meaning that these applications can run at different configurations (e.g. number of threads).

A significant body of work examines approaches on how to optimally use DVFS or hardware imposed power constrains (e.g. RAPL) in order to save energy but also optimize performance [2, 17, 19, 21, 26, 44]. A different approach is also identified, where instead of using hardware imposed power caps, energy efficiency is achieved only by job scheduling [21, 34, 35]. Manufacturing variability is also considered in some studies, which exploit the variance in power and performance among nodes to improve energy efficiency [44, 50]. Although the identified techniques are not used in production in any of the HPC centers, they provide some insight on future trends in energy efficient HPC computing.

Etinski et al. [18] propose a job scheduling policy that seeks the optimal frequency for parallel jobs in order to lower power consumption. Sarood et al. [48] use performance modeling to increase job throughput in power constrained systems. A power management of overprovisioned systems has also been studied by Patki et al. [12, 43]. Unlike our work, all sockets are viewed as homogeneous in terms of power consumption, which can lead to suboptimal scheduling decisions.

More recent work identifies the need to consider manufacturing variability when making scheduling decisions or managing a system's power budget [4, 13, 16, 22, 27, 53, 54]. Inadomi et al. [27] extensively study the impact of manufacturing variability on a number of production clusters and propose a variation-aware power budgeting framework. They introduce variability to their prediction model in similar fashion to our *PR* model, by measuring power variability using a single microbenchmark on each socket and then apply it to their original, Variability Agnostic, prediction model. In contrast to our work, their prediction model is used to guide work balancing within an MPI application and not system-wide job scheduling. Unlike our *VT* model, Inadomi's and our *PR* models assume that variability is application independent, which is not correct in general. As the effects of manufacturing variability are expected to increase in future CPUs [39], a more robust model is needed, such as our *VT* model. Chasapis et al. [13] and Totoni et al. [54] also employ runtime scheduling solutions for mitigating manufacturing variability at application level. Teodorescu et al. [53] study the impact of manufacturing variability and propose a linear programming algorithm to find the best parameters for power budgeting with DVFS, instead of optimizing job scheduling. Ellsworth et al. [16] propose a power distribution framework that optimizes an HPC cluster's power consumption under a certain system-wide power budget. In contrast to our work, jobs are scheduled without considering power consumption, but power is redistributed, favoring more power intensive jobs. A two level solution for overprovisioned clusters is presented by Gholkar et al. [22], where a job scheduler is used at system level to allocate nodes and distribute power. The job scheduler predicts the total energy consumption in order to make a scheduling decision, but unlike our work, the prediction model does not consider variability. Individual sockets may run under power constrains and a second runtime scheduler decides the optimal configuration of active processors and the power distribution among them in order to mitigate the power variability.

Our job scheduling policies can also be coupled with energy saving runtimes, to further reduce power consumption. For example, our approach can be coupled with Adagio [47], which detects the critical path of MPI codes and uses DFVS to reduce power consumption of non-critical pieces of work.

6 CONCLUSIONS

In this work, we demonstrate that taking into account manufacturing variability to drive job scheduling policies provides significant benefits in terms of performance and power consumption. We also demonstrate how traditional PMC based power prediction models can be extended to consider and predict manufacturing variability. We propose two job scheduling policies, each one using a different power prediction model: the first assumes that power variability impacts all application equally, while the second one aims at obtaining the power variability impact per application by training the model for each individual socket. We compare both approaches with a range of state-of-the-art approaches as well as an approach using an oracle model.

We examine the benefits of our policies under bursty and heavy traffic scenarios and different power budgets. We observe significant improvements on job turnaround time (up to 31% and 25% on average) and energy consumption (reducing it up to 5.5% and 3% on average) when compared to state-of-the-art approaches that do not consider appropriately the manufacturing variability in existing processors. Moreover, the model-driven policies proposed in this work accurately predict the variability per socket and, as a result, they guarantee that power consumption always remains below the system-wide budget, while the policies that rely on user estimations or prior power profiling fail to do so.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Government (Severo Ochoa grant SEV-2015-0493), by the Spanish Ministry of Science and Technology (contract TIN2015-65316-P), by the Generalitat de Catalunya (contracts 2017-SGR-1414 and 2017-SGR-1328), by the RoMoL ERC Advanced Grant (grant agreement 321253) and by the European Unions Horizon 2020 research and innovation program (grant agreement 779877). D. Chasapis has been partially supported by the Spanish Government (Severo Ochoa grant SVP-2013-067941). M. Moretó has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness under Ramon y Cajal fellowship number RYC-2016-21104. M. Casas has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness under Ramon y Cajal fellowship number RYC-2017-23269. This work was also partially performed under the auspices of Lawrence Livermore National Laboratory, which is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344.

REFERENCES

[1] 2017. The Green500 List. http://www.green500.org. (June 2017).
[2] Axel Auweter, Arndt Bode, Matthias Brehm, Luigi Brochard, Nicolay Hammer, Herbert Huber, Raj Panda, Francois Thomas, and Torsten Wilde. 2014. A Case Study of Energy Aware Scheduling on SuperMUC. In *Supercomputing*, Julian Martin Kunkel, Thomas Ludwig, and Hans Werner Meuer (Eds.). 394–409.
[3] Eduard Ayguadé, Alejandro Duran, Jay Hoeflinger, Federico Massaioli, and Xavier Teruel. 2007. An Experimental Evaluation of the New OpenMP Tasking Model.
[4] Peter E. Bailey, Aniruddha Marathe, David K. Lowenthal, Barry Rountree, and Martin Schulz. 2015. Finding the Limits of Power-constrained Application Performance. Article 79, 12 pages.
[5] Frank Bellosa. 2000. The Benefits of Event: Driven Energy Accounting in Power-sensitive Systems *(EW 9)*. 6. https://doi.org/10.1145/566726.566736
[6] Ramon Bertran, Alper Buyuktosunoglu, Meeta S. Gupta, Marc Gonzalez, and Pradip Bose. 2012. Systematic Energy Characterization of CMP/SMT Processor Systems via Automated Micro-Benchmarks *(MICRO-45)*. 13. https://doi.org/10.1109/MICRO.2012.27
[7] Ramon Bertran, Marc Gonzalez, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. 2010. Decomposable and Responsive Power Models for Multicore Processors Using Performance Counters *(ICS '10)*. 12. https://doi.org/10.1145/1810085.1810108

[8] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. 10. https://doi.org/10.1145/1454115.1454128

[9] W. L. Bircher and L. K. John. 2007. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events. 158–168. https://doi.org/10.1109/ISPASS.2007.363746

[10] William Lloyd Bircher and Lizy K. John. 2012. Complete System Power Estimation Using Processor Performance Events. *IEEE Trans. Comput.* 61, 4 (April 2012), 15. https://doi.org/10.1109/TC.2011.47

[11] W. L. Bircher, M. Valluri, J. Law, and L. K. John. 2005. Runtime Identification of Microprocessor Energy Saving Opportunities *(ISLPED '05)*. 6. https://doi.org/10.1145/1077603.1077668

[12] T. Cao, Y. He, and M. Kondo. 2016. Demand-Aware Power Management for Power-Constrained HPC Systems. 21–31. https://doi.org/10.1109/CCGrid.2016.25

[13] Dimitrios Chasapis, Marc Casas, Miquel Moretó, Martin Schulz, Eduard Ayguadé, Jesus Labarta, and Mateo Valero. 2016. Runtime-Guided Mitigation of Manufacturing Variability in Power-Constrained Multi-Socket NUMA Nodes *(ICS '16)*. Article 5, 12 pages. https://doi.org/10.1145/2925426.2926279

[14] Dimitrios Chasapis, Marc Casas, Miquel Moretó, Raul Vidal, Eduard Ayguadé, Jesús Labarta, and Mateo Valero. 2015. PARSECSs: Evaluating the Impact of Task Parallelism in the PARSEC Benchmark Suite. *ACM Trans. Archit. Code Optim.* 12, 4, Article 41 (Dec. 2015), 22 pages. https://doi.org/10.1145/2829952

[15] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. 2016. A Validation of DRAM RAPL Power Measurements. In *Proceedings of the Second International Symposium on Memory Systems (MEMSYS '16)*. 455–470. https://doi.org/10.1145/2989081.2989088

[16] Daniel A. Ellsworth, Allen D. Malony, Barry Rountree, and Martin Schulz. 2015. Dynamic Power Sharing for Higher Job Throughput *(SC '15)*. Article 80, 11 pages. https://doi.org/10.1145/2807591.2807643

[17] Maja Etinski, Julita Corbalan, Jesus Labarta, and Mateo Valero. 2010. Utilization driven power-aware parallel job scheduling. *Computer Science - Research and Development* 25, 3 (01 Sep 2010), 207–216. https://doi.org/10.1007/s00450-010-0129-x

[18] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. 2011. Linear programming based parallel job scheduling for power constrained systems. https://doi.org/10.1109/HPCSim.2011.5999809

[19] M. Etinski, J. Corbalan, J. Labarta, and M. Valero. 2012. Parallel job scheduling for power constrained {HPC} systems. *Parallel Comput.* 38, 12 (2012), 615 – 630. https://doi.org/10.1016/j.parco.2012.08.001

[20] Dror G. Feitelson, Dan Tsafrir, and David Krakov. 2014. Experience with using the Parallel Workloads Archive. *J. Parallel and Distrib. Comput.* 74, 10 (2014), 2967 – 2982. https://doi.org/10.1016/j.jpdc.2014.06.013

[21] F. Fraternali, A. Bartolini, C. Cavazzoni, and L. Benini. 2018. Quantifying the Impact of Variability and Heterogeneity on the Energy Efficiency for a Next-Generation Ultra-Green Supercomputer. *IEEE Transactions on Parallel and Distributed Systems* 29, 7 (July 2018), 1575–1588. https://doi.org/10.1109/TPDS.2017.2766151

[22] Neha Gholkar, Frank Mueller, and Barry Rountree. 2016. Power Tuning HPC Jobs on Power-Constrained Systems *(PACT '16)*. 13. https://doi.org/10.1145/2967938.2967961

[23] Bhavishya Goel, Sally A. McKee, Roberto Gioiosa, Karan Singh, Major Bhadauria, and Marco Cesati. 2010. Portable, Scalable, Per-core Power Estimation for Intelligent Resource Management *(GREENCOMP '10)*. 12. https://doi.org/10.1109/GREENCOMP.2010.5598313

[24] D. Hackenberg, T. Ilsche, R. Schne, D. Molka, M. Schmidt, and W. E. Nagel. 2013. Power measurement techniques on standard compute nodes: A quantitative comparison. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 194–204. https://doi.org/10.1109/ISPASS.2013.6557170

[25] D. Hackenberg, R. Schne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 896–904. https://doi.org/10.1109/IPDPSW.2015.70

[26] Chung-hsing Hsu and Wu chun Feng. 2005. A Power-Aware Run-Time System for High-Performance Computing. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing (SC'05)*. 1–1. https://doi.org/10.1109/SC.2005.3

[27] Yuichi Inadomi, Tapasya Patki, Koji Inoue, Mutsumi Aoyagi, Barry Rountree, Martin Schulz, David Lowenthal, Yasutaka Wada, Keiichiro Fukazawa, Masatsugu Ueda, Masaaki Kondo, and Ikuo Miyoshi. 2015. Analyzing and Mitigating the Impact of Manufacturing Variability in Power-constrained Supercomputing *(SC '15)*. Article 78, 12 pages. https://doi.org/10.1145/2807591.2807638

[28] Intel 2010. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*. Intel.

[29] Canturk Isci and Margaret Martonosi. 2003. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data *(MICRO 36)*.

[30] Morris A. Jette, Andy B. Yoo, and Mark Grondona. 2002. SLURM: Simple Linux Utility for Resource Management.

[31] Haoqiang Jin and Rob F. Van der Wijngaart. 2006. Performance Characteristics of the Multi-zone NAS Parallel Benchmarks. *J. Parallel Distrib. Comput.* 66, 5 (May 2006), 12. https://doi.org/10.1016/j.jpdc.2005.06.016

[32] Russ Joseph and Margaret Martonosi. 2001. Run-time Power Estimation in High Performance Microprocessors *(ISLPED '01)*. 6. https://doi.org/10.1145/383082.383119

[33] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3, 2, Article 9 (March 2018), 26 pages. https://doi.org/10.1145/3177754

[34] Bhavesh Khemka, Ryan Friese, Sudeep Pasricha, Anthony A. Maciejewski, Howard Jay Siegel, Gregory A. Koenig, Sarah Powers, Marcia Hilton, Rajendra Rambharos, and Steve Poole. 2015. Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system. *Sustainable Computing: Informatics and Systems* 5 (2015), 14 – 30. https://doi.org/10.1016/j.suscom.2014.08.001

[35] Katia Leal. 2016. Energy efficient scheduling strategies in Federated Grids. *Sustainable Computing: Informatics and Systems* 9 (2016), 33 – 41. https://doi.org/10.1016/j.suscom.2015.08.002

[36] Tao Li and Lizy Kurian John. 2003. Run-time Modeling and Estimation of Operating System Power Consumption *(SIGMETRICS '03)*. 12. https://doi.org/10.1145/781027.781048

[37] Livermore Computing. 2014. The Catalyst Supercomputer. http://computation.llnl.gov/computers/catalyst. (2014). http://computation.llnl.gov/computers/catalyst

[38] M. Maiterth, G. Koenig, K. Pedretti, S. Jana, N. Bates, A. Borghesi, D. Montoya, A. Bartolini, and M. Puzovic. 2018. Energy and Power Aware Job Scheduling and Resource Management: Global Survey Initial Analysis. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 685–693. https://doi.org/10.1109/IPDPSW.2018.00111

[39] Aniruddha Marathe, Yijia Zhang, Grayson Blanks, Nirmal Kumbhare, Ghaleb Abdulla, and Barry Rountree. 2017. An Empirical Survey of Performance and Energy Efficiency Variation on Intel Processors. In *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing (E2SC'17)*. ACM, New York, NY, USA, Article 9, 8 pages. https://doi.org/10.1145/3149412.3149421

[40] A. W. Mu'alem and D. G. Feitelson. 2001. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems* 12, 6 (June 2001), 529–543. https://doi.org/10.1109/71.932708

[41] Dan Nagle. 2005. MPI – The Complete Reference, Vol. 1, The MPI Core, 2Nd Ed., Scientific and Engineering Computation Series, by Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra. *Sci. Program.* 13, 1 (Jan. 2005), 7. https://doi.org/10.1155/2005/653765

[42] OpenMP Architecture Review Board. 2013. OpenMP Application Program Interface Version 4.0. (2013). http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf

[43] Tapasya Patki, David K. Lowenthal, Barry Rountree, Martin Schulz, and Bronis R. de Supinski. 2013. Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing. 10. https://doi.org/10.1145/2464996.2465009

[44] Tapasya Patki, David K. Lowenthal, Anjana Sasidharan, Matthias Maiterth, Barry L. Rountree, Martin Schulz, and Bronis R. de Supinski. 2015. Practical Resource Management in Power-Constrained, High Performance Computing. In *Proceedings of the*

24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15). ACM, New York, NY, USA, 12. https://doi.org/10.1145/2749246.2749262

[45] N. Rajovic, P.M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero. 2013. Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC?

[46] Barry Rountree, Dong H. Ahn, Bronis R. de Supinski, David K. Lowenthal, and Martin Schulz. 2012. Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound *(IPDPSW '12)*. 7. https://doi.org/10.1109/IPDPSW.2012.116

[47] Barry Rountree, David K. Lownenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, and Tyler Bletsch. 2009. Adagio: Making DVS Practical for Complex HPC Applications *(ICS '09)*. 10. https://doi.org/10.1145/1542275.1542340

[48] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant Kale. 2014. Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget. 12. https://doi.org/10.1109/SC.2014.71

[49] K. Shoga, B. Rountree, and M. Schulz. 2014. Whitelisting MSRs with msr-safe. (November 2014). https://github.com/LLNL/msr-safe

[50] Hayk Shoukourian. 2015. *Adviser for Energy Consumption Management: Green Energy Conservation*. Dissertation. Technische Universitt Mnchen.

[51] Karan Singh, Major Bhadauria, and Sally A. McKee. 2009. Real Time Power Estimation and Thread Scheduling via Performance Counters. *SIGARCH Comput. Archit. News* 37, 2 (July 2009), 10. https://doi.org/10.1145/1577129.1577137

[52] ASCAC Subcomittee. 2014. *Top ten exascale research challenges*. Technical Report. US Department of Energy.

[53] Radu Teodorescu and Josep Torrellas. 2008. Variation-Aware Application Scheduling and Power Management for Chip Multiprocessors *(ISCA '08)*. 12. https://doi.org/10.1109/ISCA.2008.40

[54] Ehsan Totoni, Akhil Langer, Josep Torrellas, and Laxmikant V.Kale. 2014. *Scheduling for HPC Systems with Process Variation Heterogeneity*. Technical Report. University of Illinois at Urbana-Champaign.