VAHRM: Variation-Aware Resource Management in Heterogeneous Supercomputing Systems

Kohei Yoshida, Ryuichi Sakamoto, *Member, IEEE*, Kento Sato, *Member, IEEE*, Abhinav Bhatele, *Member, IEEE*, Hayato Yamaki, *Member, IEEE*, Hiroki Honda, *Member, IEEE*, and Shinobu Miwa, *Member, IEEE*

***Abstract*—In this article, we propose a novel resource management technique for heterogeneous supercomputing systems affected by manufacturing variability. Our proposed technique called VAHRM (Variation-Aware Heterogeneous Resource Management) takes a holistic approach to job scheduling on highly heterogeneous computing resources. VAHRM preferentially allocates energy-efficient computing resources to an energy-consuming job in a job queue, considering the impact on both the job turnaround time and the power consumption of individual resources. Furthermore, we have developed a novel approach to modeling the power consumption of computing resources that have manufacturing variability. Our approach called TSMVA (Two-Stage Modeling with Variation Awareness) enables us to generate the first variation-aware GPU power models, which can correctly estimate the power consumption of each GPU for a given job. Our experimental results show that, compared to conventional first-come-first-serve (FCFS) and state-of-the-art variation-aware scheduling algorithms, VAHRM can achieve respective improvements in system energy efficiency of up to 5.8% and 5.4% (4.5% and 4.2% on average) while reducing the average turnaround time of 21.2% and 11.9%, respectively, for various workloads obtained from a production system.**

***Index Terms*—Heterogeneous systems, job scheduling, manufacturing variability, power models, energy efficiency.**

I. Introduction

To ACHIEVE high-performance computing capabilities, modern supercomputing systems consume a significant amount of power. An example is the Frontier supercomputer, which achieves 1.1 EFLOPS at a power consumption of 21.1 MW [1]. Since a system with higher power consumption incurs a higher operational cost, reducing power consumption while maintaining system performance is an important operational issue in many production systems [2], [3], [4], [5], [6], [7]. To improve system's energy efficiency, we need to make use of the power of computing resources such as CPUs and accelerators more effectively because they consume a large fraction of system power. It has been reported that the ratio of the power consumption of computing resources to the overall system power consumption is about 60% in a system without accelerators [8], and this number is expected to substantially increase if a system is equipped with many accelerators.

An effective approach to improving energy efficiency is to manage computing resources while considering the power-efficiency variation among them. Because a typical supercomputer is composed of identical compute nodes in the same stock-keeping unit, it is assumed that these nodes have identical power-efficiency characteristics for a given application. However, the compute nodes used in a production system often exhibit wide variations in power efficiency due to manufacturing variability [9], [10], [11], [12], [13], [14], [15], [16]. For example, Inadomi et al. showed that the power consumption of 1,920 CPU and DRAM modules in the HA8K supercomputer varied by 30% and 184%, respectively, when even running the same DGEMM program [17]. We note that performance variation among compute nodes is negligible in many cases.[1] Based on this observation, variation-aware resource management techniques such as variation-aware job schedulers have been developed [18], [19], [20], [21]. However, all of these techniques only consider the variation in a single type of resource (i.e., CPUs), and the best way to manage inherently heterogeneous resources (e.g., CPUs plus GPUs) while considering their own variations has not yet been established.

In this paper, we propose VAHRM (Variation-Aware Heterogeneous Resource Management) to improve the energy efficiency of heterogeneous supercomputing systems. Nowadays, many production systems employ accelerators (e.g., nine out of the top 10 high-performance computing systems use GPUs as of November 2023 [1]), which have their own variation in power efficiency [22], [23], [24]. Since a typical heterogeneous

Received 15 August 2024; revised 15 May 2025; accepted 29 May 2025. Date of publication 12 June 2025; date of current version 1 July 2025. Part of this research was supported in part by the KDDI Foundation Research Grant, and JST, PRESTO under Grant JPMJPR22P9. This work was supported in part by RIKEN Junior Research Associate Program. Recommended for acceptance by D. Tao. *(Corresponding author: Kohei Yoshida.)*

Kohei Yoshida is with the Department of Computer and Network Engineering, The University of Electro-Communications, Tokyo 182-8585, Japan, and also with RIKEN R-CCS, Kobe 650-0047, Japan (e-mail: yoshida@hpc.is.uec.ac.jp).

Ryuichi Sakamoto is with the Tokyo Institute of Technology, Tokyo 152-8550, Japan.

Kento Sato is with RIKEN R-CCS, Kobe 650-0047, Japan.

Abhinav Bhatele is with the University of Maryland, College Park, MD 20742-5025 USA, and also with RIKEN R-CCS, Kobe 650-0047, Japan.

Hayato Yamaki and Hiroki Honda are with the The University of Electro-Communications, Tokyo 182-8585, Japan.

Shinobu Miwa is with the The University of Electro-Communications, Tokyo 182-8585, Japan, and also with RIKEN R-CCS, Kobe 650-0047, Japan.

This article has supplementary downloadable material available at https://doi.org/10.1109/TPDS.2025.3577252, provided by the authors.

Digital Object Identifier 10.1109/TPDS.2025.3577252

[1] The performance variation between compute nodes is caused by frequency throttling associated with power capping (i.e., the clock frequency in a compute node is degraded when a running application hits the power limit), but this rarely happens in production systems during normal operation.

system has a larger number of accelerators, each consuming more power than a CPU, the resource manager should allocate computing resources to a job while minimizing the impact of the power-efficiency variation among accelerators in addition to among CPUs. VAHRM achieves this by combining *a holistic approach to variation-aware job scheduling in heterogeneous systems* with the very accurate power models created by *Two-Stage Modeling with Variation Awareness (TSMVA)*.

The main contributions of this study are as follows.

- We developed Variation-Aware Heterogeneous Resource Management (VAHRM), a novel resource management technique for heterogeneous supercomputing systems, which suffer from manufacturing variability.
- We developed Two-Stage Modeling with Variation Awareness (TSMVA), a novel approach to modeling the power consumption of computing resources that have manufacturing variability. With this approach, we created the first variation-aware GPU power models, which enable a resource manager to estimate the power consumption of each GPU for a given job before execution.
- We showed that VAHRM had a great ability to improve the energy efficiency of a heterogeneous system. Our experimental results show that, compared to conventional first-come-first-serve (FCFS) and state-of-the-art variation-aware scheduling algorithms, VAHRM can achieve respective improvements in both system energy efficiency of up to 5.8% and 5.4% (4.5% and 4.2% on average) and average turnaround time of 21.2% and 11.9%, respectively, for various workloads obtained from a production system. We note that the saving of system energy of 5% is significant because it corresponds to a reduction in an electricity bill of \$1 million per year in a system consuming 20 MW.
- We showed that our power models generated by TSMVA had a high ability to predict the power consumption of both CPUs and GPUs for various applications. Our experimental results show that our models can make predictions with errors of 1.54% and 3.05% on average for 48 CUDA kernels executed on 256 NVIDIA P100 and V100 GPUs, respectively, and with an error of 1.69% on average for 11 kernels executed on 128 Intel Xeon CPUs.

The remainder of this paper is organized as follows. First, we describe the background of this study in Section II. Next, we propose VAHRM and TSMVA in Sections III and IV, respectively. Then, we present the experimental setup and results in Sections V and VI, respectively. We summarize the related work in Section VII, and present the conclusions of this study in Section VIII.

II. Background

A. Impact of Manufacturing Variability in Supercomputers

Many researchers have reported that production systems show power-efficiency variation across hardware components in the same stock-keeping unit. Rountree et al. executed six NAS Parallel Benchmark applications on CPUs in both 32- and 137-node clusters, and reported that these CPUs showed differences of up to 10 W in power consumption [15]. Inadomi et al. investigated 1,920 CPU and DDR3 DRAM modules in the HA8K supercomputer and found that these modules had differences of up to 184% in power consumption [17]. Hackenberg executed an application on each node in two different supercomputers (SuperMUC and Taurus) and showed that these nodes had power differences of up to 41 W [12]. Marathe et al. showed an increase in the energy-efficiency variation of CPUs by executing various HPC applications in three supercomputers, each having a different CPU architecture [14].

It has also been reported that the power efficiency of GPUs is affected by manufacturing variability. Patki et al. executed a molecular dynamics application on 196 NVIDIA V100 GPUs with power/frequency capping in the Lassen supercomputer deployed at LLNL. They showed that there were differences of up to 1.2$\times$ and 2.0$\times$ in GPU clock frequency and power consumption, respectively [23]. Yoshida et al. executed 48 CUDA kernels on 856 devices selected from three generations of NVIDIA GPUs (P100, V100, and A100) and showed that the power-efficiency variation was strongly affected by both the kernels currently running and the GPU generation [22]. Sinha et al. executed five GPU applications in five supercomputing systems and showed that these systems showed variations of 20% –104% in the power efficiency of GPUs [24].

Some supercomputers use FPGA cards as accelerators. For example, the Noctua 2 supercomputer operated at Paderborn University has a subsystem composed of 36 compute nodes, each having multiple Xilinx Alveo U280 or Intel Stratix 10 FPGA chips [25]. The Cygnus supercomputer at the University of Tsukuba has an Albireo subsystem composed of 80 compute nodes, each having two Intel Stratix 10 FPGA chips in addition to four NVIDIA V100 GPUs [26]. To the best of our knowledge, the power-efficiency variation among FPGA chips has not been reported yet, but we believe that each FPGA chip has its own power-efficiency characteristics because it is also affected by manufacturing variability.

B. Variation-Aware Resource Management

The existence of power-efficiency variation has motivated many researchers to develop various variation-aware resource management techniques. Chasapis et al. proposed new job schedulers to improve the energy efficiency of supercomputers with power-efficiency variation between CPUs [19]. They extended SLURM [27] to a variation-aware job scheduler using two variation-aware CPU power models, which are described in Section II-C. Their experimental results showed that their scheduler achieved energy savings of up to 5.5% and 3.0% on average when compared with a variation-unaware scheduler based on SLURM. Zhou et al. proposed another variation-aware job scheduler that makes scheduling decisions using a combinatorial optimization algorithm [28]. Totoni et al. proposed an integer-linear-programming-based framework that allocates cores to threads while considering the core-to-core variation within a CPU [21]. Shoukourian et al. proposed a configuration adviser to select the most energy-efficient combination of compute nodes and CPU frequencies for a given application while considering the power-efficiency variation between CPU nodes [20].

These previous studies demonstrate that variation-aware resource management is helpful in improving the energy-efficiency of some production systems, but unfortunately, their research focus is limited to situations in which a system has a single type of variation (i.e., the power-efficiency variation among CPUs or cores). This is not applicable to today's highly heterogeneous systems, which are designed to have multiple types of computing resources initially, each individually affected by a different manufacturing process. The resource managers used in such systems need to take into account the multiple variations existing in different types of computing resources for further improvements in energy efficiency.

C. Variation-Aware Power Modeling

To our knowledge, only one previous study has developed two variation-aware power models called the Power Ratio and Variability-Trained Prediction models for CPUs [19]. Both models assume that the weighted sum of the activity ratio of each hardware component (e.g., ALU) within a CPU forms the overall power of the CPU. The activity ratio of each hardware component for a given application is computed with the performance monitoring counter (PMC) profiles obtained through the execution of the application on a single reference CPU, and the weight values are trained using regression analysis. The two models differ in terms of training and prediction methods. The Power Ratio model predicts the power of each CPU by multiplying a CPU-specific power ratio with respect to a reference CPU with the value predicted by a reference power model. The reference power model is trained with sample PMC profiles obtained through the execution of some reference applications on the reference CPU. In contrast, the Variability-Trained Prediction model predicts the power of each CPU with a power model trained with sample data obtained through the execution of the reference applications on the CPU. It was reported that both models achieve an average error below 10% for 12 single-CPU applications executed on 256 CPUs.

The energy-efficient management of heterogeneous computing resources requires precise variation-aware power models for accelerators in addition to CPUs. However, to the best of our knowledge, no researcher has developed such models for accelerators, including GPUs. We therefore propose a new variation-aware power modeling technique, which is available for both CPUs and GPUs.

III. Variation-Aware Job Scheduling in Heterogeneous Systems

A. Prerequisites

The following three types of jobs are executed in supercomputers equipped with accelerators.

- CPU jobs: Perform all computation only on CPUs.
- Accelerator jobs: Perform the main computation on a single type of accelerator (e.g., GPUs). The other type of accelerator (even if available) is not used for computation. CPUs are mainly used to control the accelerators and perform inter-node communication.
- Hybrid jobs: Perform the main computation on 2+ types of computing resources (e.g., CPUs and GPUs) simultaneously.

Because it is not easy for application developers to write programs that are executed on multiple types of computing resources concurrently, hybrid jobs are rarely executed on heterogeneous supercomputers. In this paper, we therefore assume that only CPU and accelerator jobs are submitted to a job scheduler. Nevertheless, we note that our algorithm, presented in Section III-C, can be easily extended to scheduling hybrid jobs.

In addition, we assume that all jobs are exclusively executed each other on the dedicated compute nodes. Like many job schedulers used in production systems, our job scheduler prohibits multiple jobs from concurrently sharing the same compute nodes for performance reproducibility. We assume that CPUs in compute nodes allocated to accelerator jobs consume idle power during execution. This is because such jobs mainly use the accelerators within the nodes and the CPUs maintain an idle state most of the time. We assume that the power of each CPU and accelerator is modeled in a training stage, which is executed during the acceptance testing of a production system.

Many production systems have their own power budgets and need to keep their power consumption under the limit during operation. To achieve this task, some power-aware job schedulers have been proposed that manage not only computing resources but also power pudgets [19], [29], [30], [31], [32]. In this paper, we consider a situation in which a system has only a system-wide power budget (i.e., no power capping is executed for nodes) and the system's power consumption is allowed to exceed the power budget instantaneously. This assumption was also used in previous work [19].

B. Scheduling Policy

To reduce the system energy, we allocate compute nodes to jobs in a greedy manner. More specifically, we pick up a job with the highest energy consumption from the job queue and then allocate the required number of free nodes to it in order of lowest power consumption. This assumes that we save more overall system energy by saving the energy of more energy-consuming jobs. The energy consumption of a job is computed using the power models described in Section IV. These models are also used to predict the power consumption of each node if the job runs on the node. We note that the types of jobs do not affect the scheduling results. Regardless of whether the job is a CPU or accelerator job, the most energy-consuming job has the top priority. For this, we manage submitted jobs with a single job queue, but our proposed approach can be extended to systems composed of multiple job queues that have their own computing resources.

Our scheduling policy ignores the job submission order. It allows a job to be executed before the execution of the preceding

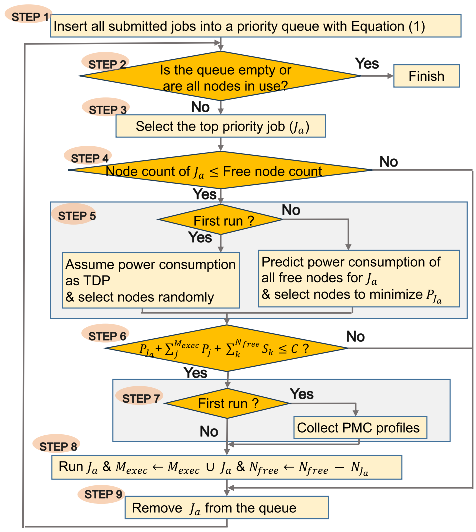

Fig. 1. Flow of job scheduling in VAHRM.

jobs in the job queue, thereby increasing both node utilization and energy efficiency. This policy, however, increases the turnaround time of jobs with small energy consumption because such jobs are frequently overtaken by subsequent jobs with high energy consumption. To improve the turnaround time of jobs, we need to raise the priority of a job that has been kept waiting in the job queue for a long time. For this, we use the following formula to determine the priority of a job.

$$P_j = \beta \times E_j + (1 - \beta) \times W_j \tag{1}$$

where P_j, E_j, and W_j represent the priority, estimated energy consumption, and waiting time (i.e., the elapsed time from submission) of a job j, respectively. To compute E_j, we assume that we can know the execution time of each job before execution because the execution time of parallel applications can be predicted with high accuarcy [33], [34]. β is a weight parameter balancing the system energy efficiency and job turnaround time. The value of β is in the range 0.0–1.0. The closer β is to 1.0, the more energy-efficient a targeted system becomes, whereas the closer it is to 0.0, the smaller the job turnaround time. We assume that the value of β is determined through the operational test of a production system.

Under this scheduling policy, the execution of jobs requesting many nodes is sometimes greatly delayed because a sufficient number of nodes are rarely released and the remaining available nodes are preferentially used by subsequent jobs that request a few nodes. To avoid this situation, we set the upper limit of waiting time (e.g., one week) for each job. If the waiting time of a job exceeds the above limit, our scheduling system stops the algorithm shown in Fig. 1 until the assignment of the unmovable job and then restarts the algorithm after the assignment. We note that no job hits the above limit during our experiments shown in Section VI-C.

C. Scheduling Algorithm

The flow of our job scheduling algorithm is shown in Fig. 1. When one or more running jobs terminate execution or a new job is submitted to the job queue, our algorithm tries to allocate free compute nodes to the jobs in the job queue using the following eight steps.

STEP 1: We first compute the priority of each job in the job queue using (1). For a job of which the PMC profile has not yet been collected, our power model cannot predict the power consumption. In this case, we assume that the power consumption of the job is equal to the product of the TDP of a node and the requested node count. In addition, we create a new empty priority queue and then add all jobs in the job queue to the new priority queue in priority order.

STEP 2: If the priority queue is empty or there is no free node, we stop the job scheduling.

STEP 3: We take a job from the head of the priority queue. The job is referred to as J_a.

STEP 4: We check the availability of nodes. If the number of nodes requested by J_a is greater than the number of available nodes, we jump to Step 9.

STEP 5: If J_a is not first run, that is, our scheduling system has the PMC profile for J_a, we select candidate nodes to be allocated to J_a as follows. First, we estimate the power consumption of each available node when J_a is executed on that node. The prediction for J_a is executed with the corresponding PMC values recorded in the job scheduling system and the CPU or GPU power models generated by the method described in Section IV.[2] Next, we sort the available nodes in ascending order of power consumption, and take the same number of nodes as the requested node count from the head of the sorted list of available nodes.

On the other hand, if J_a is first run, we assume the power consumption of a node for J_a as the TDP so as not to exceed the system power constraints and select candidate nodes to be allocated to J_a randomly.

STEP 6: We check power availability. If the system exceeds a given power constraint when executing J_a on the candidate nodes, we jump to Step 9.

STEP 7: If J_a is first executed, the PMC profile is collected on the candidate nodes. Since the parameters of our model are normalized by the number of instructions per second, as described in Appendix, available online, it is sufficient to obtain a value corresponding to the profile of the first few iterations of the job (i.e., profiling the entire job is unnecessary). Therefore, the performance overhead of profiling is marginal.

STEP 8: We assign the candidate nodes to J_a and remove J_a from the job queue.

STEP 9: We remove J_a from the priority queue and then return to Step 2.

Step 5 requires an accurate power model of each computing resource to predict the power consumption for a given job; however, no variation-aware power model has been developed for accelerators, as described in Section II-C. Because many supercomputing systems use GPUs as accelerators, establishing

[2] Scheduling hybrid jobs requires both CPU and GPU power models. This allows the power consumption of a job to be estimated by summing the predicted values from both the CPU and GPU power models. For CPU and GPU jobs, on the other hand, one of CPU and GPU power models is used.

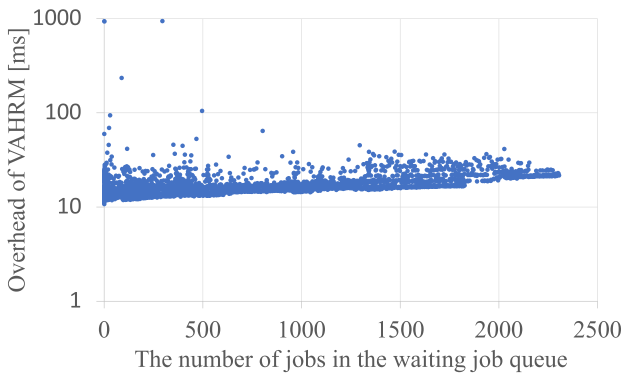

Fig. 2. Execution time overhead of VAHRM.

job scheduling for GPU-based supercomputing systems is an important first step for managing various types of computing resources in heterogeneous systems. For this, we developed variation-aware GPU power models in this study.

We note that the performance overhead of VAHRM is quite small, though it depends on the number of jobs in the job queue. Fig. 2 shows the execution time overhead of VAHRM. The horizontal and vertical axes of this figure represent the number of jobs in the job queue and the execution time of VAHRM, respectively. This figure shows that the execution time of VAHRM increases as the number of jobs in the queue increases; however, the performance overhead of VAHRM is about 0.03 seconds even when there are about 2,000 jobs in the queue. This overhead is only 3% of the scheduling interval (i.e., 1 s) and therefore acceptable for many supercomputing users.

The figure also shows some outliers in the region of 1–500 jobs. They are affected by other processes because VAHRM is executed on a login node. Even in such an exceptional case, our VAHRM shows a performance overhead of only 1 s.

The energy overhead of VAHRM is negligible because our VAHRM needs to modify only the scheduler program that is typically executed on a login node. The modified program itself has no additional impact on the energy consumption of CPUs and GPUs on compute nodes.

As mentioned in Section III-B, our scheduling policy ignores the order in which jobs are submitted, which may delay the start time of the execution of many jobs. Fig. 3 shows the relationship between the number of nodes requested in each job and the waiting time in VU-FCFS-B (see Table V) and VAHRM. This figure indicates that VAHRM tends to show shorter waiting time for jobs requesting a small number of nodes compared to the VU-FCFS-B, whereas waiting time for jobs requesting a large number of nodes is longer (up to 18 times). However, we note that not all large jobs are delayed in execution. More specifically, we analyzed waiting times of jobs requesting more than 15 nodes, and found that some jobs run shorter waiting times than with VU-FCFS-B and the average relative waiting time in this case was 0.99. In addition, even when the execution of a job is delayed, the waiting time is within 24 hours.

Based on these observations, we believe that VAHRM's strategy has a limited impact on the system performance and would be acceptable to many HPC users. Even if this policy were not

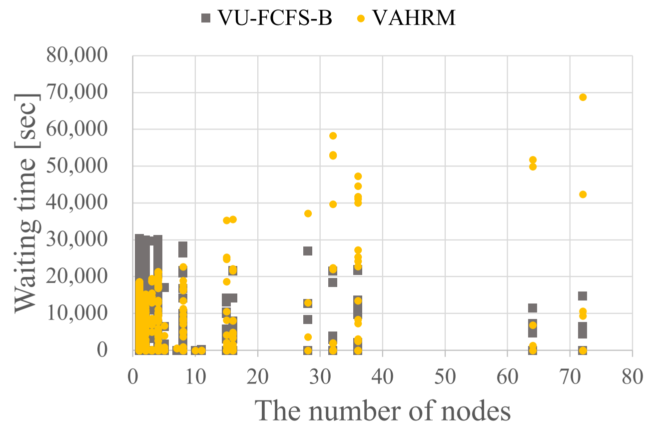

Fig. 3. The relationship between the number of nodes requested and the waiting time for each job.

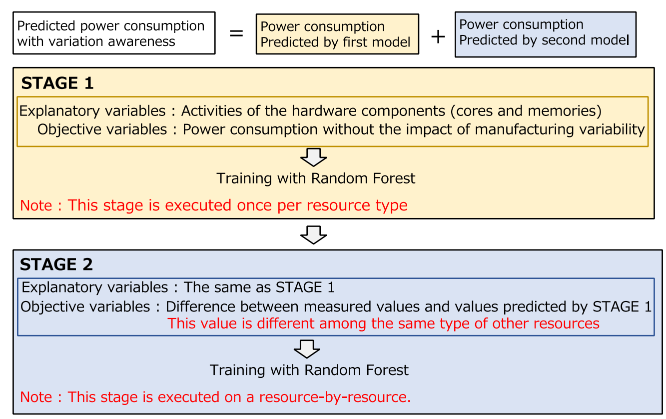

Fig. 4. Overview of proposed approach to modeling the power consumption of computing resources affected by manufacturing variability.

acceptable, VAHRM allows us to reduce the waiting time for jobs with a large number of nodes by setting the variable β close to 0, in exchange for the reduction in energy saving.

IV. Two-Stage Power Modeling With Variation Awareness

A. Overview

We propose a novel approach to modeling the power consumption of computing resources affected by manufacturing variability. Our proposed approach, which was inspired by gradient boosting [35], trains multiple models in an additive manner and uses the sum of the outputs of the trained models as the final output of the prediction, as shown in Fig. 4. Unlike traditional gradient boosting, our approach uses a strong learner like a random forest for each model and trains the first model with intermediate goals given explicitly. This approach enables us to generate very accurate models that predict the power consumption of computing resources affected by manufacturing variability.

Our approach is built on the assumption that the power consumption of a resource (CPU or GPU) affected by manufacturing variability consists of the baseline (average power consumption over all CPUs or all GPUs in the targeted system) and deviation components. This can be represented as follows.

$$P_j^i = B_j^{CPU|GPU} + D_j^i \tag{2}$$

where P_j^i represents the power consumption of the resource i for the application j, $B_j^{CPU|GPU}$ represents the baseline component of CPU or GPU for the application j, and D_j^i represents the deviation component of the resource i for the application j, respectively. We note that $B_j^{CPU|GPU}$ is independent of the resource i. Specifically, $B_j^{CPU|GPU}$ can be defined as follows.

$$B_j^{CPU|GPU} = \frac{1}{N^{CPU|GPU}} \Sigma_{i=1}^{N^{CPU|GPU}} P_j^i \tag{3}$$

where $N^{CPU|GPU}$ represents the number of CPUs or GPUs.

Based on this assumption, our approach uses two models to predict the power consumption of each computing resource i. These models are trained in the following two stages individually.

STAGE 1: We train the first model to output $B_j^{CPU|GPU}$ for any application j. Because $B_j^{CPU|GPU}$ does not depend on individual resources, this stage is executed once per resource type (i.e., once the training for a resource is completed, the trained model can be used to predict the power consumption of the same type of other resources).

STAGE 2: We train the second model to predict the difference in power consumption between the values predicted by the first model and the values measured on the target resource. That is, the second model is trained to produce $(P_j^i - f_1(\boldsymbol{x}_j))$, where $\boldsymbol{x}_j$ represents the explanatory variables of the application j and $f_1(\boldsymbol{x}_j)$ represents the output of the first model for $\boldsymbol{x}_j$.

When predicting the power consumption of resources, we use the two models additively. The final prediction for the application j running on the resource i (denoted y_j^i) can be expressed as follows.

$$y_j^i = f_1(\boldsymbol{x_j}) + f_2(\boldsymbol{x_j}) \tag{4}$$

where $f_2(\boldsymbol{x_j})$ represents the output of the second model for $\boldsymbol{x}_j$.

B. Implementation Details

Although many researchers use linear regression to model the power consumption of computing resources [19], [36], we use random forests because they show a higher prediction accuracy than linear regression for many tasks. Similar to the previous work [19], [36], we use PMC profiles as input to our models, assuming that the power consumption of a computing resource depends on the activity of the hardware components (e.g., the cores and memories) within the resource when running an application. However, the definitions of the explanatory variables are different between the types of computing resources (i.e., CPUs and GPUs). This is because the PMCs available depend on the resource types.

To collect ground truth data (P_j^i), we executed a set of test applications on each computing resource. The details of the test applications are described in Section V-E. We also executed the same applications on a computing resource to collect the PMC profiles and then convert them to x_j.

Our GPU power model consists of six explanatory variables. Three of the six variables are associated with cores (i.e., the number of INT, FP32, and FP64 instructions executed), while the rest are related to memories (the number of GPU, shared, and texture memory accesses). Similarly, our CPU model has six explanatory variables. Three of the six variables are related to cores (the number of instructions, I-TLB loads, and branch misses executed), while the rest are related to memories (D-TLB, L1 data cache, and LLC accesses). When converting PMC profiles to explanatory variables, the corresponding PMC values are normalized to the execution time of the application. For further details, please refer to Appendix, available online.

V. Experimental Setup

A. Simulator and System Configuration

Our experiments are carried out mainly with a simulator, as well as the only comparable previous study [19]. This is because simulation experiments enable us to perform a comprehensive evaluation of VAHRM for various experimental conditions at a realistic cost, so that we can obtain more statistically reliable data. Our simulation results are validated with a real system in Section VI-B.

We use an in-house simulator that mimics the behavior of a job scheduler in a production system at a time step of one second. Our simulator precisely computes the timings of the submission, execution, and termination for each job with various scheduling algorithms while feeding a scheduler's log produced by a production system, and then outputs the system throughput, job turnaround and waiting times, and system's energy consumption for the given workload. We ignored the time required for a job scheduler to make a scheduling decision because it is relatively small compared to our simulation time step. We also ignored the impact of scheduling algorithms on the energy consumption of a node on which a job scheduler is executed. This is because the energy consumption of the scheduler node is substantially smaller than that of the compute nodes.

Our simulator has two parameters that affect the results of job scheduling. One is *shrink ratio of job submission time* (SR), whereas the other is *system's power constraint ratio* (PCR). The details of these parameters are described in Table I.

We use the TSUBAME3.0-like system [37] shown in Table II as a targeted system. We assume that CPU and GPU consume 40 W and 60 W during idle state, respectively. As a result, our targeted system consumes 172.8 kW in total in case that all compute nodes are idle.

B. Workloads

We use nine workloads shown in Table III. They are composed of three workload groups having different ratios of CPU and GPU jobs (1:9, 5:5, and 9:1), and each group consists of three

TABLE I
SIMULATION PARAMETERS

Parameters	Description
SR (shrink ratio of job submission time)	It adjusts the submission time between jobs. The job submission time recorded in a given log file is divided by this ratio. The higher this ratio, the busier the system becomes.
PCR (system's power constraint ratio)	Ratio of the power constraint of the entire system to the TDP. The larger this value, the more jobs can be executed simultaneously.

TABLE II
SYSTEM CONFIGURATION

Number of compute nodes	540
CPU per node	2x Intel Xeon E5-2680 (120 W TDP)
GPU per node	4x NVIDIA Tesla P100 (300 W TDP)
TDP	780 kW

TABLE III
WORKLOADS

ID	Job ratio (CPU:GPU)	Average power consumption
0	9:1	Small
1	9:1	Medium
2	9:1	Large
3	5:5	Small
4	5:5	Medium
5	5:5	Large
6	1:9	Small
7	1:9	Medium
8	1:9	Large

TABLE IV
POWER CONSUMPTION OF JOBS

ID	Job type	Typical power consumption per node [W]
1	CPU	169
2	CPU	206
3	CPU	245
4	GPU	251
5	GPU	597
6	GPU	1095

TABLE V
JOB SCHEDULING SYSTEMS

Name	Description
VU-FCFS	Variation-Unaware FCFS. It allocates compute nodes to jobs in an FCFS manner.
VU-FCFS-B	VU-FCFS with backfilling
CVA-FCFS	CPU-Variation-Aware FCFS, as proposed in the previous work [19]. Power-efficient nodes are allocated to CPU jobs, while randomly-selected nodes are allocated to GPU jobs. All jobs are handled in an FCFS manner.
CVA-FCFS-B	CVA-FCFS with backfilling
CGVA-FCFS	CPU-and-GPU-Variation-Aware FCFS. Any type of job uses the most power-efficient free nodes. All jobs are handled in an FCFS manner.
CGVA-FCFS-B	CGVA-FCFS with backfilling
VAHRM+TSMVA	VAHRM plus power models built by TSMVA
VAHRM+IDEAL	VAHRM plus the ideal power model. Predicted power consumption has no error.

workloads, each having its own job mix. More specifically, we create each workload by differently mixing three CPU and three GPU jobs with different (small, medium, and large) power consumption shown in Table IV. For example, 90% and 10% of the jobs in the workload 0 are CPU and GPU jobs that consume 164 W and 251 W per node, respectively. We note that all jobs that have the same job ID show identical power consumption per node, but the execution time and the number of nodes requested are different among the jobs. As for the submission and execution times and the requested node counts of jobs, we use the information recorded on a scheduler's log in TSUBAME3.0. We use the 10,000 job records written in June and July 2019.

We also note that the power consumption shown in the table is a typical value, and the actual values depend on the computing resources used for execution. We give a more detailed explanation of our power simulation in Section V-D.

C. Job Scheduling Systems

We test eight job scheduling systems including VAHRM. Two of the eight scheduling systems are VAHRM, and each of the two uses a different model to predict the power-efficiency variation between GPUs or CPUs. Four out of the eight are the existing scheduling systems. Two of the four are conventional and unaware of the power-efficiency variation between computing resources, whereas the others are state-of-the-art and take only the power-efficiency variation between CPUs into account. The rest are the extensions of the state-of-the-art to consider the power-efficiency variation between GPUs in addition to CPUs. The details of the job scheduling systems we use are summarized in Table V.

D. Power Simulation

We model the power-efficiency variation between GPUs in the targeted system as follows. First, we select three kernels with different power consumption from 48 diverse kernels shown in the paper [22] as the GPU jobs shown in Table IV. Second, we measure the power consumption of 256 GPUs in TSUBAME3.0 with *nvidia-smi* by executing these kernels on each GPU. We use the resulting values as the power consumption of 256 out of 2,160 GPUs in the targeted system. Finally, we complement the power consumption of the remaining 1,904 GPUs in the targeted system with normal random numbers generated with the mean and variance of the power consumption of 256 GPUs.

Unfortunately, we cannot know the power-efficiency variation between CPUs in TSUBAME3.0 because TSUBAME3.0 does not support user-level CPU power measurement tools such as RAPL [38]. Instead, we statistically model the power consumption of all CPUs in the targeted system, by using the experimental data collected in the Cygnus supercomputer, which is deployed at the University of Tsukuba and supports the RAPL interface. To

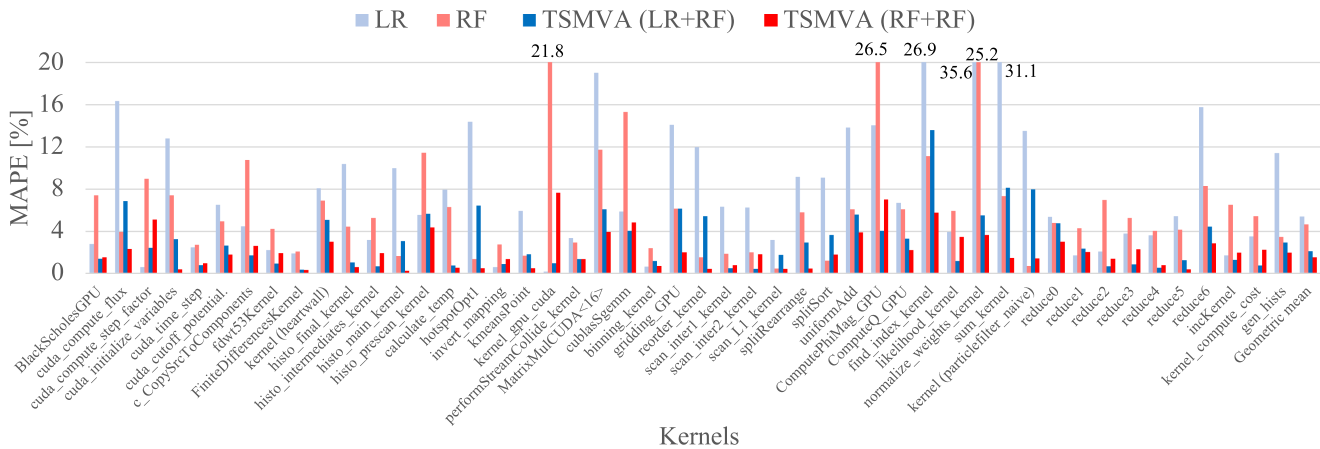

Fig. 5. Mean absolute percentage errors for the prediction of the power consumption of 256 GPUs in TSUBAME3.0. Lower is better.

do this, we first executed three applications with different power consumption on each of the 128 CPUs installed in Cygnus. Two of the three applications are selected from the NAS Parallel Benchmark suite, and the other is the DGEMM kernel implemented with Intel MKL [39]. The power consumption of these applications is shown in Table IV. Next, we computed the mean and variance of the power consumption of the 128 CPUs for each application and then generated the power data for all CPUs in the targeted system, while assuming that the distribution of the power consumption of the CPUs in the targeted system is the same as that in Cygnus.

E. Power Models

We tested four models to predict the power consumption of computing resources. Two of the four models are created by TSMVA, each using a different machine learning algorithm (i.e., linear regression or random forest) for the first stage. The rest are generated by a method similar to Variability-Trained Prediction [19], which trains a single-stage model (linear regression or random forest) to fit sample data collected on each resource. To model the power consumption of GPUs, we trained these models with the power and PMC profiles collected by executing 48 kernels on each of the 256 GPUs in TSUBAME3.0. For modeling CPU power, these models were trained with the power and PMC profiles collected by running 11 kernels on each of the 128 CPUs in Cygnus.

In addition to these models, we tested the ideal model in Section VI-C. The ideal model outputs the actual power consumption that was collected as ground truth for training other realistic models, resulting in no prediction error.

VI. Experimental Results

A. Power Model Validation

Fig. 5 shows the prediction accuracy of the GPU power models. The vertical axis represents MAPEs (Mean Absolute Percentage Errors) in the case of predicting the power consumption of the 256 NVIDIA P100 GPUs in TSUBAME3.0, while the horizontal axis represents test kernels. The light blue and light red bars represent LR (linear regression) and RF (random forest), respectively. Both LR and RF are trained in a single stage. The dark blue bar represents TSMVA (LR+RF), which uses linear regression in the first stage of TSMVA, while the dark red bar represents TSMVA (RF+RF), which uses a random forest in the first stage. Both TSMVA (LR+RF) and (RF+RF) use a random forest in the second stage.

The figure shows that both TSMVA (RF+RF) and (LR+RF) can predict the power consumption of GPUs very correctly. More specifically, TSMVA (RF+RF) and (LR+RF) show errors of 1.54% and 2.12% on average (up to 7.64% and 13.58%), respectively. In contrast to this, LR and RF show errors of 5.4% and 4.6% on average (up to 35.76% and 26.45%), respectively. Thus, we can say that our TSMVA has a great ability to improve the accuracy of predicting the power consumption of P100 GPUs affected by manufacturing variability.

TABLE VI
Mean Absolute Percentage Errors for the Prediction of the Power Consumption of 128 CPUs and 256 GPUs in Cygnus

	LR	RF	TSMVA (LR+RF)	TSMVA (RF+RF)
CPU	5.03	3.18	2.83	1.69
GPU	15.04	8.51	5.31	3.05

To further demonstrate the effectiveness of TSMVA, we evaluated the four power modeling methods with 128 Intel Xeon Gold 6126 CPUs and 256 NVIDIA V100 GPUs installed in Cygnus. Our experimental results are summarized in Table VI. The table shows that our TSMVA (RF+RF) shows the smallest error for both Xeon CPUs and V100 GPUs. Our TSMVA is effective for various types of computing resources.

B. Simulator Validation

To validate our simulation results, we evaluated the impact of our scheduling algorithm on the energy consumption of 128 CPUs and 256 GPUs (i.e., 64 compute nodes) in Cygnus. We used nine workloads, each of which has 300 jobs. The details of these workloads are the same as in Table III. We ran our simulator

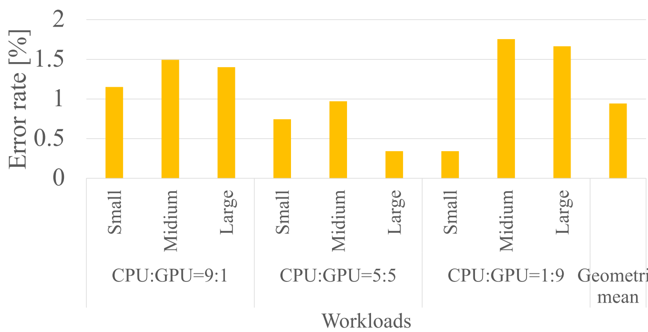

Fig. 6. Error rate between our simulator and the real system. Lower is better.

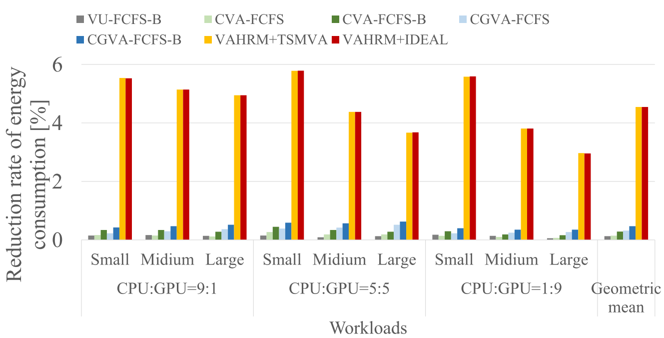

Fig. 7. Energy savings of various job scheduling systems. We normalize the results of eight scheduling systems to the results of VU-FCFS. Higher is better.

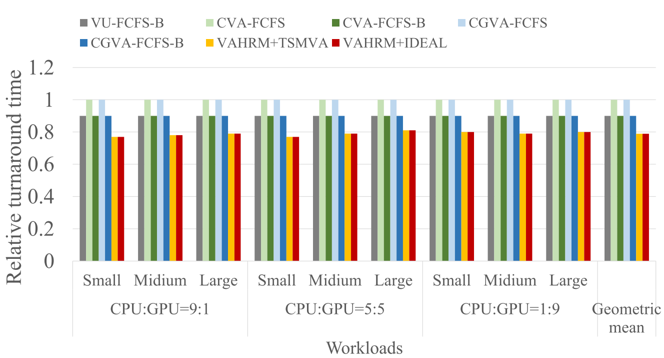

Fig. 8. Job turnaround time of various job scheduling systems. We normalize the results of eight scheduling systems to the results of VU-FCFS. Lower is better.

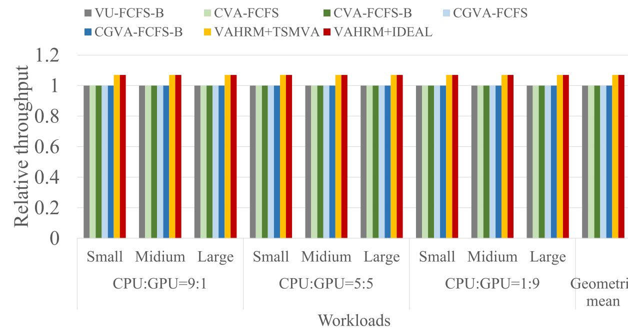

Fig. 9. System throughput of various job scheduling systems. We normalize the results of eight scheduling systems to the results of VU-FCFS. Higher is better.

to determine the execution timing and node assignment for all jobs and then measured the energy consumption of the real nodes in Cygnus while running all jobs according to the pre-determined timing and assignment.

Fig. 6 shows the error rate between our simulator and Cygnus. As you can see, our simulator can accurately estimate the energy consumption of the real system. Our simulator has an error of 0.94% on average (up to 1.76%).

C. Impact of Variation-Aware Job Scheduling

1) Main Results: Figs. 7, 8, and 9 show the energy savings, job turnaround time, and system throughput of the eight job scheduling systems, respectively. The horizontal axes of these figures represent workloads. The vertical axes of Figs. 7, 8, and 9 represent energy saving relative to VU-FCFS (higher is better), average job turnaround time relative to VU-FCFS (lower is better), and system throughput relative to VU-FCFS (higher is better), respectively. VU-FCFS is selected as the baseline for a fair comparison with previous work [19]. We use $\beta = pow(1/4, 6)$, SR=1.0, and PCR=1.0.

Fig. 7 shows that VAHRM can improve the energy efficiency of the targeted system greatly. More specifically, VAHRM+TSMVA (RF+RF) achieves improvements of up to 5.8% and 4.5% on average in the system's energy efficiency for the nine workloads when compared to VU-FCFS. Moreover, VAHRM+TSMVA can save more energy (energy savings of up to 5.4% and 4.2% on average) than the state-of-the-art job scheduling system (CVA-FCFS-B). This is because of two reasons. First, VAHRM allocates more energy-efficient GPUs to GPU jobs than the state-of-the-art by predicting the power-efficiency variation among GPUs. Second, VAHRM allocates more energy-efficient resources to jobs with high energy consumption than the state-of-the-art by ignoring the order of job submission.

Fig. 8 shows that VAHRM also has the ability to improve the average turnaround time of jobs when compared to the existing job scheduling systems. More specifically, VAHRM+TSMVA reduces the average job turnaround time by up to 22.8% and 21.2% on average when compared to the job scheduling systems without backfilling (i.e., VU-FCFS, CVA-FCFS, and CGVA-FCFS). Moreover, VAHRM+TSMVA improve the average job turnaround time by up to 13.8% and 11.9% on average when compared to the job scheduling systems with backfilling (i.e., VU-FCFS-B, CVA-FCFS-B, and CGVA-FCFS-B). This is because VAHRM relaxes the restrictions on job scheduling by ignoring the job submission order. VAHRM allocates jobs to compute nodes as much as possible considering the priority order associated with β, resulting in a great reduction in the turnaround time of jobs.

Fig. 9 shows that VAHRM shows higher throughput than the other job scheduling systems. Specifically, VAHRM+TSMVA achieves an improvement of 7.1% in system throughput compared to VU-FCFS. This is because VAHRM executes as many jobs as possible while ignoring the order of job submission, by

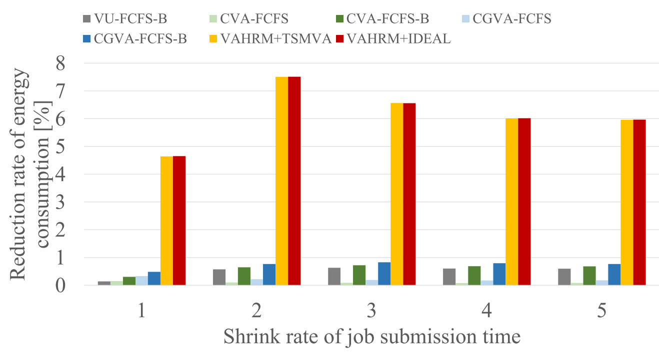

Fig. 10. Energy savings for various shrink ratios of job submission time. Higher is better.

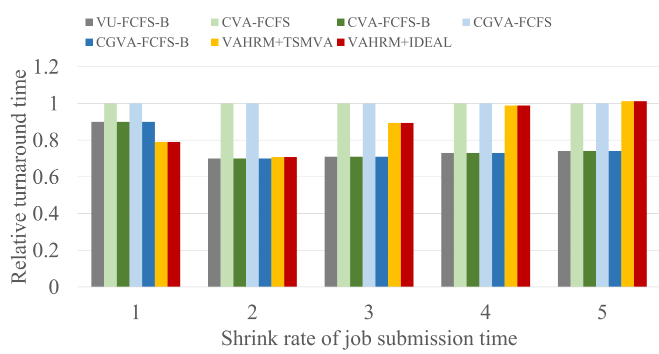

Fig. 11. Job turnaround time for various shrink ratios of job submission time. Lower is better.

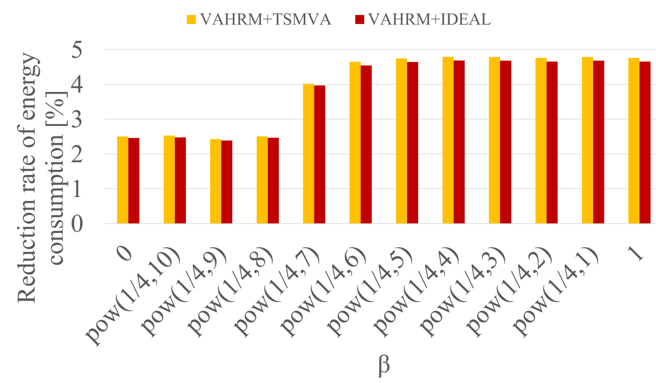

Fig. 12. Energy savings with different values of β. Higher is better.

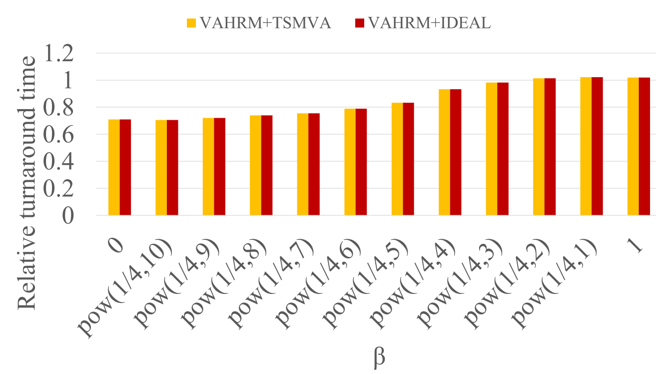

Fig. 13. Job turnaround time with different values of β. Lower is better.

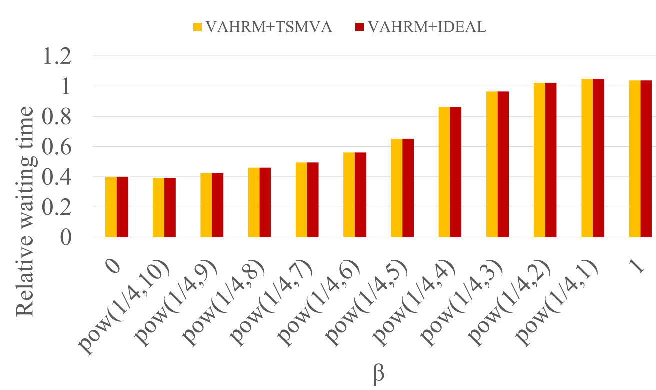

Fig. 14. Waiting time with different values of β. Lower is better.

allocating computing resources to jobs as long as both resource count and power constraints are satisfied.

We note that VAHRM shows very small differences in all evaluation metrics (energy savings, job turnaround time, and system throughput) between realistic and ideal power models. This is because our power models generated by TSMVA have very high accuracy, as shown in Section VI-A. We can say that the predictive accuracy of our power models is sufficient for variation-aware job scheduling.

2) Impact of Congestion: Figs. 10 and 11 show the impact of congestion on energy savings and job turnaround time, respectively. The horizontal axes of these figures represent shrink ratios of job submission time (SR). We note that the higher the SR, the more congested the targeted system. We show the values averaged over the nine workloads.

Fig. 10 shows that VAHRM can save more energy as the target system is congested. This is because VAHRM has a higher chance of allocating energy-efficient computing resources to energy-consuming jobs as the number of jobs in the job queue increases. However, the opportunity to allocate computing resources to jobs with small energy consumption is reduced when the targeted system is congested, and such jobs need to wait in the job queue for a longer time. Therefore, VAHRM increases the average job turnaround time as SR increases, as shown Fig. 11.

3) Impact of β: Figs. 12, 13, and 14 show the impact of β on energy savings, job turnaround time, and waiting time, respectively. The horizontal axes of these figures represent β. A value of β largely influences energy savings, job turnaround, and waiting times. Fig. 12 shows that VAHRM shows higher energy saving by setting a large number of β, just as we thought, whereas Figs. 13, and 14 show that we can reduce average job turnaround and waiting times by using a small number for β, respectively, as expected. Therefore, we strongly recommend that a decision of a value of β is based on not only energy savings but also both job turnaround and waiting times.

Although the optimal value of β depends on the operating policy of the targeted system, we show an example of how to determine β considering the trade-off between system energy savings and job waiting time. In general, larger energy savings

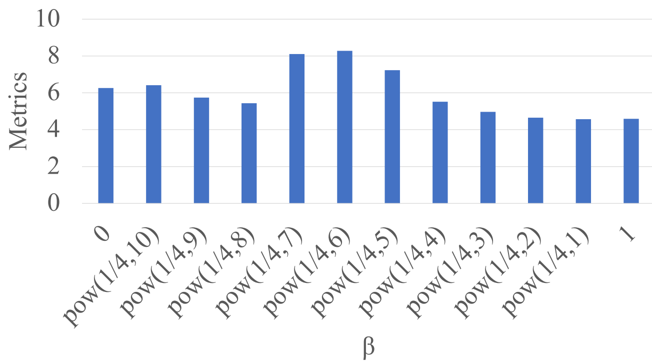

Fig. 15. Metrics for determination of β. Higher is better.

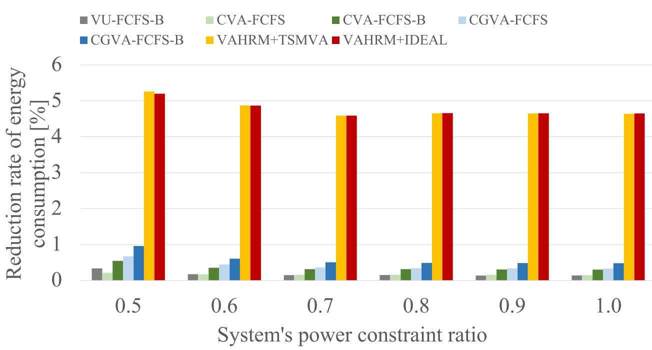

Fig. 16. Energy savings for various power constraints. Higher is better.

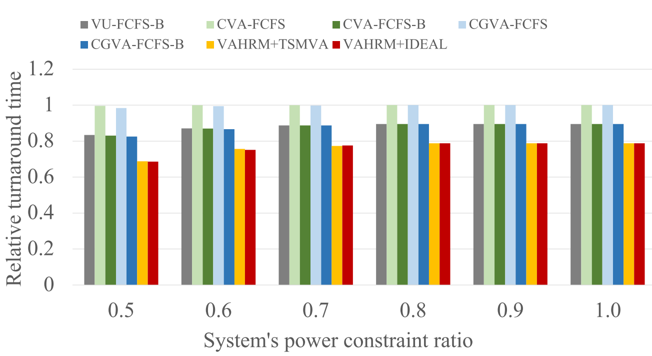

Fig. 17. Job turnaround time for various power constraints. Lower is better.

and shorter waiting times are preferable. To achieve this goal, we can use the following equation to determine β.

$$M_\beta = \frac{Erate}{Wrate} \tag{5}$$

where M_β, $Erate$, and $Wrate$ represent metric, energy reduction rate, and relative waiting time. $Erate$ and $Wate$ are relative to VU-FCFS. Fig. 15 shows the values of this metric for various β. The metric is maximized when $\beta = pow(1/4, 6)$. At this point, we can achieve both high energy efficiency (an energy reduction rate of 4.5%) and small waiting time (a relative waiting time of 0.56).

4) Impact of Power Constraints: Figs. 16 and 17 show the impact of power constraints on energy savings and job turnaround time, respectively. The horizontal axes of these figures represent system's power constraint ratios (PCR).

Fig. 16 shows that VAHRM shows the highest energy saving in the case of PCR$= 0.5$. This is because VAHRM keeps energy-inefficient computing resources unused if a stringent power constraint is given to the targeted system.

Fig. 17 shows that VAHRM reduces the average job turnaround time as PCR decreases. This is because large jobs showing high energy consumption have little chance to overtake small jobs showing low energy consumption in a highly power-constrained system. VAHRM has a similar effect to backfilling in such a system.

TABLE VII
ENERGY, TURNAROUND TIME, THROUGHPUT, AND WAITING TIME UNDER DIFFERENT PREDICTION ERRORS IN OUR POWER MODEL

Error (%)	Energy	Turnaround time	Throughput	Waiting time
5	1.000	1.001	1.000	1.005
10	1.000	1.000	0.999	1.003
15	1.002	1.004	0.997	1.016
20	1.001	1.005	0.998	1.019
25	1.001	1.002	0.998	1.016
30	1.002	0.999	0.997	1.007

We normalize the results to those with no prediction error.

5) Impact of Accuracy of Power Model: We evaluated the impact of the accuracy of power models on the system performance and energy efficiency while adding some noise to the ideal power model. Table VII shows our experimental results. Each value in this table is the average across nine workloads and is normalized by the value with the ideal power model. We use $\beta = pow(1/4, 6)$, SR=1.0, and PCR=0.5. From this table, all values are closer to 1.0 and the accuracy of power models has a small impact in VAHRM.

In VAHRM, the accuracy of power models can mainly affect the selection of nodes to be used in STEP 5 and the check of power constraints in STEP 6. In STEP 5, jobs are assigned by predicting the power consumption of nodes and sorting them in ascending order. Therefore, the accuracy of power models is not affected if the order of the nodes does not change in the sorted node list. In contrast, in STEP 6, power consumption should be accurately predicted. This is because system throughput can be improved by running more jobs within the power constraint.

These results imply that the power model built with TSMVA is inaccurate enough not to affect the order of nodes and is accurate enough to prevent the system power consumption from exceeding the given power constraint.

6) Impact of PMC Collection: Since VAHRM needs to collect PMCs for first-run jobs, the ratio of the number of first-run jobs to the total number of jobs could affect the performance of workloads. Figs. 18, 19, and 20 show the impact of the ratio of first-run jobs on energy savings, system throughput, and job turnaround time, respectively. The horizontal axes represent the ratio of first-run jobs. A value of 1 means that all jobs are first-run. The vertical axes are the same as Figs. 7, 8, and 9, respectively. Each bar represents the average across nine workloads, which is normalized to VU-FCFS. We conservatively assumed that each first-run job required 5% of additional execution

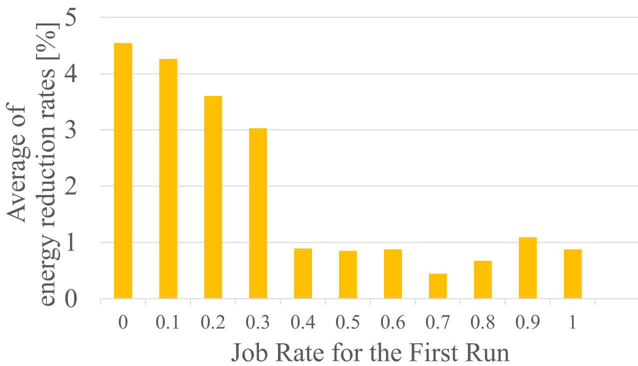

Fig. 18. Energy reduction rate for various rate for the first-run jobs. Higher is better.

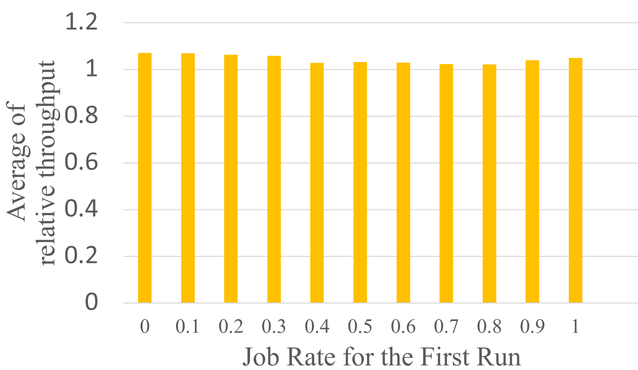

Fig. 19. Throughput for various rate for the first-run jobs. Higher is better.

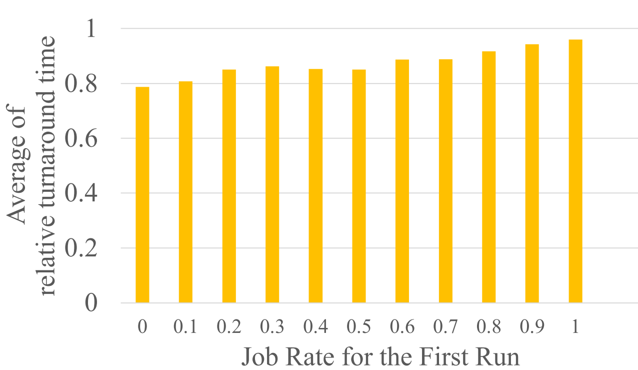

Fig. 20. Turnaround time for various rate for the first-run jobs. Lower is better.

time for PMC collection, although collecting PMCs for an entire application is unnecessary.

Fig. 18 shows that the energy savings gradually decrease as the ratio of first-run jobs increases, and the energy saving is approximately 1% when the number of first-run jobs exceeds 40% of the total number of jobs. This is because an increase in the ratio of first-run jobs leads to a larger number of jobs requiring PCM collection, which increases execution time. This, in turn, results in higher system energy consumption. We note that many jobs are executed multiple times on GPU-based production systems because deep learning and scientific applications account for the majority of jobs and widely use checkpoint/restart techniques for tolerating failures.

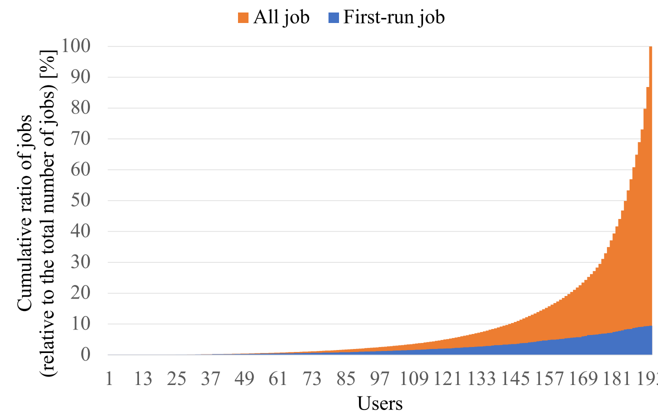

Fig. 21. Cumulative ratio of jobs relative to the total number of jobs per user.

From Figs. 8 and 9, the system throughput is little affected by the ratio of first-run jobs, but the turnaround time increases as the ratio is closer to 1.0. This is due to the larger overhead of PMC collection. In summary, VAHRM is effective when the ratio of first-run jobs is less than or equal to 0.3.

To the best of our knowledge, no existing paper reports the typical proportion of first-run jobs in real-world systems. Accordingly, we estimated this proportion using the job log from TSUBAME3.0. The log, which contains 118,890 job entries from 193 users, each including a job ID, an anonymized user ID, submit time, start time, end time, queue name, and number of cores, was collected during June and July 2019. However, the log does not explicitly indicate whether a job is a first-run or a repeated submission. To address this limitation, we adopted the same assumption as in [40], where two jobs are considered as the same application if they have the same user ID, queue name, core count, and execution time within $\pm$ 10% . Based on this assumption, we estimated the proportion of first-run jobs in the production environment.

Fig. 21 presents a cumulative bar plot of job counts per user. The x-axis represents users sorted in ascending order by the total number of jobs executed, and the y-axis shows the cumulative ratio. The orange bars represent all jobs in the log, while the blue bars represent only first-run jobs. As shown in the figure, the gap between the two bar series becomes wider toward the right, suggesting that heavy users tend to repeatedly execute similar jobs. We found that first-run jobs account for approximately 9% of all jobs, which is considerably less than 40% .

VII. Related Work

A. Job Scheduling

Many researchers have developed various techniques for job scheduling in supercomputing systems. Dai et al. proposed ESlurm, which uses a new communication tree strategy and job runtime estimation to improve communication and scheduling efficiency [41]. Hofmeyr et al. investigated the effectiveness of time-sharing scheduling on supercomputing systems [42]. To improve the response time of interactive jobs, Minami et al. proposed oversubscription scheduling, which allows multiple jobs to share one computing resource [43]. To improve the performance of communication-intensive jobs, Mishra et al. proposed communication-aware job scheduling, which considers the job's communication behavior [44].

To balance competing goals of job scheduling (e.g., improve both job waiting time and system utilization), Tang et al. proposed metric-aware scheduling, which enables a job scheduler to adaptively adjust scheduling policies based on feedback from monitored metrics at runtime [45]. Cao et al. proposed cooling and node location-aware job scheduling, which executes power-hungry jobs on computing nodes having a small impact on the hotspot temperature to prevent the hotspot temperature from increasing while improving system throughput [46].

These studies help to improve the scheduling efficiency in supercomputing systems; however, none of them takes power-efficiency variation among computing resources into account.

B. *Power Modeling*

Many power models have been developed for both CPUs [19], [47], [48], [49] and GPUs [50], [51], [52]. A few of them are aware of power-efficiency variation between CPUs [19], but all GPU power models proposed in these studies are variation-unaware. Therefore, we cannot use them to predict power-efficiency variation between GPUs.

Also, some power models have been proposed for FPGA cards [53], [54]. However, there is no research on FPGA cards that takes power-efficiency variation into account.

VIII. Conclusion

In this study, we proposed VAHRM, a new resource management technique for heterogeneous supercomputing systems. VAHRM is a new holistic approach to variation-aware job scheduling in heterogeneous systems, combined with a new variation-aware CPU and GPU power models generated by our TSMVA. Our experimental results showed that VAHRM has a great ability to improve both the system's energy efficiency and job turnaround time for various workloads. We also showed that our variation-aware CPU and GPU power models generated by TSMVA had high accuracy when predicting the power-efficiency variation between computing resources.

Our future work is to implement VAHRM on real job scheduling systems such as SLURM. We will also evaluate the effectiveness of VAHRM on another heterogeneous system including FPGA.

Acknowledgment

The author are thankful to M. Ono for his contribution to the consideration of scheduling policy and algorithm of this work. This study was carried out using the supercomputer TSUBAME3.0 operatiod by the Tokyo Institute of Technology and the Cygnus system provided by the CCS, University of Tsukuba.

References

[1] TOP500, "November 2023," 2023. [Online]. Available: Internet://www.top500.org/lists/top500/2023/11/

[2] Flagship 2020 Project, "2017 annual report," 2019. [Online]. Available: Internet:www.r-ccs.riken.jp/wp-content/uploads/2019/08/Flagship_2020_Project_2017.pdf

[3] J. Gao et al., "Sunway supercomputer architecture towards exascale computing: Analysis and practice," *Sci. China Inf. Sci.*, vol. 64, no. 4, pp. 1–21, 2021.

[4] T. C. O. Z. Lab, "Energy-efficient computing," 2023. [Online]. Available: Internet://www.zettascale.hpc.cam.ac.uk/themes/energy-efficient-computing/

[5] X.-K. Liao et al., "Moving from exascale to zettascale computing: Challenges and techniques," *Front. Inf. Technol. Electron. Eng.*, vol. 19, no. 10, pp. 1236–1244, 2018.

[6] P. Messina and S. Lee, "The U.S. exascale computing project," in *Proc. 2016 Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2016. [Online]. Available: https://www.exascaleproject.org/wpcontent/uploads/2016/11/Presentation-ECP-SC16-BOF-11-16-16_web-1.pdf?utm_source=chatgpt.com

[7] Z. Wang, Y. Tang, J. Chen, J. Xue, Y. Zhou, and Y. Dong, "Energy wall for exascale supercomputing," *Comput. Inform.*, vol. 35, pp. 941–962, 2016.

[8] S. Wallace, V. Vishwanath, S. Coghlan, Z. Lan, and M. Papke, "Measuring power consumption on IBM blue Gene/Q," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process. Workshops PhD Forum*, 2013, pp. 853–859.

[9] B. Acun, A. Buyuktosunoglu, E. K. Lee, and Y. Park, "Power aware heterogeneous node assembly," in *Proc. 2019 IEEE Int. Symp. High Perform. Comput. Archit.*, 2019, pp. 715–727.

[10] S. Dighe et al., "Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 184–193, Jan. 2011.

[11] F. Fraternali, A. Bartolini, C. Cavazzoni, and L. Benini, "Quantifying the impact of variability and heterogeneity on the energy efficiency for a next-generation ultra-green supercomputer," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1575–1588, Jul. 2018.

[12] D. Hackenberg, "Node power consumption variability," in *Proc. Energy-Efficient HPC WG Workshop*, 2014. [Online]. Available: https://drive.google.com/file/d/1ELSXZzLgCxhpWG90qkcmY6zEs7469qgJ/view

[13] Y. Kodama, T. Odajima, E. Arima, and M. Sato, "Evaluation of power management control on the supercomputer Fugaku," in *Proc. 2020 IEEE Int. Conf. Cluster Comput.*, 2020, pp. 484–493.

[14] A. Marathe, Y. Zhang, G. Blanks, N. Kumbhare, G. Abdulla, and B. Rountree, "An empirical survey of performance and energy efficiency variation on Intel processors," in *Proc. 5th Int. Workshop Energy Efficient Supercomput.*, 2017, pp. 9:1–9:8.

[15] 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 *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops PhD Forum*, 2012, pp. 947–953.

[16] T. Kusaba et al., "Power-efficiency variation on A64FX supercomputers and its application to system operation," in *Proc. 2024 IEEE Int. Conf. Cluster Comput. Workshop*, 2024, pp. 55–65.

[17] Y. Inadomi et al., "Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing," in *Proc. 2015 Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2015, pp. 78:1–78:12.

[18] S. Bhalachandra, B. Austin, and N. J. Wright, "Understanding power variation and its implications on performance optimization on the cori supercomputer," in *Proc. 2021 Int. Workshop Perform. Model. Benchmarking Simul. High Perform. Comput. Syst.*, 2021, pp. 51–62.

[19] D. Chasapis, M. Moretó, M. Schulz, B. Rountree, M. Valero, and M. Casas, "Power efficient job scheduling by predicting the impact of processor manufacturing variability," in *Proc. ACM Int. Conf. Supercomput.*, 2019, pp. 296–307.

[20] H. Shoukourian, T. Wilde, A. Auweter, and A. Bode, "Power variation aware configuration adviser for scalable HPC schedulers," in *Proc. 2015 Int. Conf. High Perform. Comput. Simul.*, 2015, pp. 71–79.

[21] E. Totoni, A. Langer, J. Torrellas, and L. Kale, "Scheduling for HPC systems with process variation heterogeneity," PPL Technical Report, pp. 1–10, 2015.

[22] Y. Kohei, S. Rio, M. Shinobu, Y. Hayato, and H. Hiroki, "Analyzing performance and power-efficiency variations among NVIDIA GPUs," in *Proc. 51th Int. Conf. Parallel Process.*, 2022, Art. no. 65.

[23] T. Patki et al., "Comparing GPU power and frequency capping: A case study with the MuMMI workflow," in *Proc. 2019 IEEE/ACM Workflows Support Large-Scale Sci.*, 2019, pp. 31–39.

[24] P. Sinha, A. Guliani, R. Jain, B. Tran, M. D. Sinclair, and S. Venkataraman, "Not all GPUs are created equal: Characterizing variability in large-scale, accelerator-rich systems," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2022, Art. no. 65.

[25] Paderborn University, "Noctua 2," 2022. [Online]. Available: Internet://pc2.uni-paderborn.de/hpc-services/available-systems/noctua2

[26] University of Tsukuba, "Overview of cygnus: A new supercomputer at CCS," 2019. [Online]. Available: Internet://www.ccs.tsukuba.ac.jp/wp-content/uploads/sites/14/2018/12/About-Cygnus.pdf

[27] A. B. Yoo, M. A. Jette, and M. Grondona, “SLURM: Simple linux utility for resource management,” in *Job Scheduling Strategies for Parallel Processing*, Berlin, Germany: Springer, 2003, pp. 44–60.
[28] J. Zhou, K. Fukazawa, and H. Nakashima, “Energy aware scheduler of single/multi-node jobs exploiting node heterogeneity,” IPSJ Tech. Rep.2021-HPC-178, pp. 1–12, 2021.
[29] T. Cao, Y. He, and M. Kondo, “Demand-aware power management for power-constrained HPC systems,” in *Proc. 16th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, Los Alamitos, CA, USA, 2016, pp. 21–31.
[30] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz, “Dynamic power sharing for higher job throughput,” in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, New York, NY, USA, 2015, Art. no. 80.
[31] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, “Parallel job scheduling for power constrained HPC systems,” *Parallel Comput.*, vol. 38, no. 12, pp. 615–630, 2012.
[32] N. Gholkar, F. Mueller, and B. Rountree, “Power tuning HPC jobs on power-constrained systems,” in *Proc. 2016 Int. Conf. Parallel Architectures Compilation*, New York, NY, USA, 2016, pp. 179–191.
[33] J. Bhimani, N. Mi, M. Leeser, and Z. Yang, “New performance modeling methods for parallel data processing applications,” *ACM Trans. Model. Comput. Simul.*, vol. 29, pp. 1–24, 2019.
[34] J. Zhai, W. Chen, W. Zheng, and K. Li, “Performance prediction for large-scale parallel applications using representative replay,” *IEEE Trans. Comput.*, vol. 65, no. 7, pp. 2184–2198, Jul. 2016.
[35] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Ann. Statist.*, vol. 29, pp. 1189–1232, 2001.
[36] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, “Statistical power modeling of GPU kernels using performance counters,” in *Proc. Int. Conf. Green Comput.*, 2010, pp. 115–122.
[37] Tokyo Institute of Technology, “TSUBAME computing services,” Apr. 5, 2022. [Online]. Available: Internet://www.t3.gsic.titech.ac.jp/en
[38] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, “RAPL: Memory power estimation and capping,” in *Proc. 2010 ACM/IEEE Int. Symp. Low-Power Electron. Des.*, 2010, pp. 189–194.
[39] Intel, “Accelerate fast math with Intel oneAPI math kernel library,” Mar. 21, 2024. [Online]. Available: Internet://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
[40] T. Patel, Z. Liu, R. Kettimuthu, P. Rich, W. Allcock, and D. Tiwari, “Job characteristics on large-scale systems: Long-term analysis, quantification, and implications,” in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2020, pp. 1–17.
[41] Y. Dai et al., “Towards scalable resource management for supercomputers,” in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2022, Art. no. 24.
[42] S. Hofmeyr, C. Iancu, J. Colmenares, E. Roman, and B. Austin, “Time-sharing redux for large-scale HPC systems,” in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun., IEEE 14th Int. Conf. Smart City, IEEE 2nd Int. Conf. Data Sci. Syst.*, 2016, pp. 301–308.
[43] S. Minami, T. Endo, and A. Nomura, “Effectiveness of the oversubscribing scheduling on supercomputer systems,” in *Proc. Int. Conf. High Perform. Comput. Asia-Pacific Region*, 2023, pp. 18–28.
[44] P. Mishra, T. Agrawal, and P. Malakar, “Communication-aware job scheduling using slurm,” in *Proc. Workshop Proc. 49th Int. Conf. Parallel Process.*, 2020, Art. no. 20.
[45] W. Tang, D. Ren, Z. Lan, and N. Desai, “Adaptive metric-aware job scheduling for production supercomputers,” in *Proc. 41st Int. Conf. Parallel Process. Workshops*, 2012, pp. 107–115.
[46] T. Cao, W. Huang, Y. He, and M. Kondo, “Cooling-aware job scheduling and node allocation for overprovisioned HPC systems,” in *Proc. 2017 IEEE Int. Parallel Distrib. Process. Symp.*, 2017, pp. 728–737.
[47] R. Bertran, A. Buyuktosunoglu, M. S. Gupta, M. Gonzalez, and P. Bose, “Systematic energy characterization of CMP/SMT processor systems via automated micro-benchmarks,” in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2012, pp. 199–211.
[48] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, “Decomposable and responsive power models for multicore processors using performance counters,” in *Proc. 24th ACM Int. Conf. Supercomput.*, New York, NY, USA, 2010, pp. 147–158.
[49] W. L. Bircher and L. K. John, “Complete system power estimation using processor performance events,” *IEEE Trans. Comput.*, vol. 61, no. 4, pp. 563–577, Apr. 2012.
[50] L. Braun, S. Nikas, C. Song, V. Heuveline, and H. Fröning, “A simple model for portable and fast prediction of execution time and power consumption of GPU kernels,” *ACM Trans. Archit. Code Optim.*, vol. 18, no. 1, 2021, Art. no. 7.
[51] K. Fan, B. Cosenza, and B. Juurlink, “Predictable GPUs frequency scaling for energy and performance,” in *Proc. 48th Int. Conf. Parallel Process.*, 2019, Art. no. 52.
[52] J. Guerreiro, A. Ilic, N. Roma, and P. Tomas, “GPGPU power modeling for multi-domain voltage-frequency scaling,” in *Proc. 2018 IEEE Int. Symp. High Perform. Comput. Archit.*, 2018, pp. 789–800.
[53] L. Deng, K. Sobti, and C. Chakrabarti, “Accurate models for estimating area and power of FPGA implementations,” in *Proc. 2008 IEEE Int. Conf. Acoust. Speech Signal Process.*, 2008, pp. 1417–1420.
[54] M. Richa, J.-C. Prévotet, M. Dardaillon, M. Mroué, and A. E. Samhat, “High-level early power estimation of FPGA IP based on machine learning,” in *Proc. 29th IEEE Int. Conf. Electron. Circuits Syst.*, 2022, pp. 1–4.

Kohei Yoshida received the BS and MS degrees in engineering from The University of Electro-Communications, Japan, in 2016 and 2023, respectively. He is currently working toward the PhD degree with The University of Electro-Communications, Japan from 2023. He is a junior research associate with the Center for Computational Science, RIKEN (RIKEN R-CCS), Japan from 2024. His research interests are high performance computing, application analysis, and deep learning. He is a member of ACM.

Ryuichi Sakamoto (Member, IEEE) received the PhD degree from the Tokyo University of Agriculture and Technology, in 2015. He was a post-doc with The University of Tokyo, in 2015–2016, assistant professor with The University of Tokyo, in 2016–2021. He is an associate professor with the Department of Mathematical and Computing Science, Tokyo Institute of Technology from 2021. His research interests are computer architecture, system software for large scale systems, and domain-specific architectures.

Kento Sato (Member, IEEE) received the BS degree from the Department of Information Science, Tokyo Tech, in 2008, the MS and PhD degrees from the Department of Mathematical & Computing Sciences, Tokyo Tech, in 2010 and 2014, respectively. He is a team leader of High Performance Big Data Research Team and a unit leader of Data Management Platform Development Unit with the Center for Computational Science, RIKEN (RIKEN R-CCS). His research area is distributed systems and parallel computing, particularly in High Performance Computing (HPC). Major focuses of his research are artificial intelligence, machine learning and deep learning in HPC, application reproducibility (MPI reproducibility, and Validation), scalable fault tolerance (Scalable checkpoint/restart, Fault tolerant MPI, Resilient system design), and I/O optimization (NVRAM, Burst buffer, and Big data), co-designing and cloud computing.

Abhinav Bhatele (Member, IEEE) received the BTech degree in computer science and engineering from I.I.T. Kanpur, India, in May 2005, and the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign, in 2007 and 2010 respectively. He is an associate professor with the Department of Computer Science, and director of the Parallel Software and Systems Group with the University of Maryland, College Park. His research interests are broadly in systems and AI, with a focus on parallel computing and distributed AI.

Hayato Yamaki (Member, IEEE) received the PhD degree in engineering from Keio University, Japan, in 2016. He is now an associate professor with the University of Electro-Communications. His research interests are internet architecture and computer architecture, especially packet switching. He is a member of ACM.

Hiroki Honda (Member, IEEE) received the BS, MS, and PhD degrees in electrical engineering from Waseda University, Japan, in 1984, 1986 and 1991, respectively. He is currently a professor of The University of Electro-Communications, Japan. He was the chair of IEEE Computer Society Japan Chapter from 2009 to 2011. His research interests include parallel processing, paralyzing compiler, parallel computer architecture. He is a member of IEEE-CS and ACM.

Shinobu Miwa (Member, IEEE) received the PhD degree in informatics from Kyoto University, in 2007. He joined Lawrence Livermore National Laboratory as a visiting scientist and professional and is now an associate professor with the University of Electro-Communications. His research interests are computer architecture and high performance computing. He is a member of ACM.