Building a Digital Twin of Job Scheduling and Power Management on an HPC System

Tatsuyoshi Ohmura[1], Yoichi Shimomura[2], Ryusuke Egawa[2,3][0000−0001−8966−867X], and Hiroyuki Takizawa[2][0000−0003−2858−3140]

[1] AI Platform Division, NEC Corporation, Japan
tat-ohmura@nec.com
[2] Cyberscience Center, Tohoku University, Japan
{shimomura32,takizawa}@tohoku.ac.jp
[3] School of Engineering, Tokyo Denki University, Japan
egawa@mail.dendai.ac.jp

Abstract. The purpose of this work is to reduce the burden on system administrators by virtually reproducing job scheduling and power management of their target systems and thereby helping them properly configure the system parameters and policies. Specifically, this paper focuses on a real computing system, named Supercomputer AOBA, as an example to discuss the importance of accurately reproducing the behaviors of job scheduling in the simulation. Since AOBA uses some special power saving features that are not supported by any existing job scheduling simulators, we have first implemented a component for a job scheduler to support the special features, and thus to build a "Digital Twin" of AOBA. By using the Digital Twin with actual operation data, a system administrator can check if the system is efficiently used in terms of computational performance and power efficiency. This paper shows a use case of exploring appropriate scheduling and power saving parameters. In the use case, we found that there are more appropriate parameter configurations, which can reduce the job waiting time by 70% at most and the energy consumption by 1.2% at most when the system is busy. By exploiting such a Digital Twin, therefore, it is demonstrated that a system administrator can properly adjust various parameters without disturbing the system operation.

Keywords: job scheduling · simulator · power saving · HPC · parameter survey

1 Introduction

In recent years, large amounts of computing resources have become required for a diverse range of applications such as emerging Artificial Intelligence applications, and traditional numerical simulations in the High Performance Computing (HPC) field. As a result, the operation of a real-world HPC system with a large number of users is required to simultaneously satisfy various requirements

such as users' satisfaction, system utilization, and power consumption. To meet the requirements, a key component is a job scheduler, such as SLURM [24], LSF [8], and Open PBS [19].

In general, job schedulers have many scheduling algorithms and functions to increase users' satisfaction and system utilization, and hence require the system administrators to make various decisions usually in advance of system operation. Typical examples are as follows.

- Backfill Scheduling
  Backfill scheduling algorithms such as EASY backfilling [16] have been studied extensively. A system administrator needs to properly decide the algorithm, parameters and policy for efficient backfill scheduling without knowing the statistical information about future job submissions.
- Multi Queue Scheduling
  A queue can virtually divide computer resources. For example, a queue can be created for each job size. It is not trivial to decide how much computing resources should be allocated to each queue. When a queue is busy, a system administrator may dynamically allocate more computing resource to the queue. However, such a dynamic adjustment further complicates the resource allocation problem.
- Power Saving Scheduling
  Job schedulers can work together with power management mechanisms that turn off some computing resources to reduce the power consumption. However, power management always requires some overheads, and there is a risk of degrading users' satisfaction and system utilization especially if it is inappropriately configured.

System administrators need to understand and use various functions of job schedulers, to fully exploit their HPC systems. To properly configure a job scheduler, they need to analyze a large amount of data, such as user-submitted jobs, node status, and utilization, and keep monitoring whether their HPC systems are being operated efficiently. Moreover, even if they find that the system is being inefficiently used, they cannot easily change the parameters and policies of job scheduling because it is difficult to predict how the changes affect the system operation efficiency, i.e., users' satisfaction, system utilization, and power consumption.

The purpose of this work is to reduce the burden on the system administrators by building a Digital Twin of job scheduling and power management of an HPC system. Our approach is to compare the simulated results of job scheduling and power management to the observed ones on a real-world HPC system, and improve the power-aware job scheduling simulator as the core of the Digital Twin. By using the Digital Twin, system administrators can check if the system operation efficiency is as expected. They can also estimate the system operation efficiency under different parameter configurations to find a better configuration. Moreover, since the Digital Twin enables to estimate how parameter adjustment will affect the system operation efficiency, it will also be helpful to automate

dynamic adjustment of parameters and policies so as to improve the system operation efficiency.

In this paper, we focus on the AOBA system installed at Tohoku University Cyberscience Center as an example of real-world HPC systems, and show the feasibility of building its Digital Twin so as to discuss how such a Digital Twin is helpful to improve the system operation efficiency. First, we describe the real-world target HPC system and its operation. Then, we present extensions of an existing job scheduling simulator to reproduce the behaviors of the AOBA system. Finally, using the simulator, we perform a parameter survey to find appropriate parameters that can maximize the system operation efficiency while minimizing the power consumption of the AOBA system.

The main contributions of this paper are as follows:

- Showing the feasibility of implementing a commercial job scheduler simulation by extending an existing simulator, SLURM Simulator.
- Discussing how helpful the accurate simulation of job scheduling and power management, refereed to as a Digital Twin of HPC systems, is for parameter survey to improve the operation efficiency of a real-world HPC system.

2 Related Work

The most important part of the Digital Twin discussed in this paper is to fully simulate the behavior of the HPC system's job scheduler. Job scheduling simulators have actively been developed, and typical examples of such simulators are ALENA [9] and GridSim [25]. More recently, the SLURM Simulator [23], a job simulator that reproduces the job scheduling of a real-world HPC system, has been introduced. SLURM Simulator is designed based on SLURM [24], which is used in various systems registered in the Top 500 list [27], and has functions to simulate job operations similar to those in real systems, such as priority, partition (queue), and scheduling parameters. Using SLURM Simulator, it is also possible to estimate Quality of Service (QoS) such as the average waiting time of jobs and system utilization with a given parameter configuration.

Maiterth et al. [13] have reported that real-world HPC systems are operated with Power Saving Scheduling. Although the power saving features of SLURM can be simulated in SLURM Simulator, commercial job schedulers often provide some other advanced features that are not supported by the simulator. SLURM's Power Saving Scheduling is controlled mainly by two parameters. One is the idle time until turning off a node, and the other is the number of nodes allowed to start up per minute. On the other hand, other job schedulers could have more parameters than SLURM. For example, a proprietary job scheduler named NQSV [17] has an additional parameter that limits the number of times a node could be turned off in a day to prevent frequent node starts. This could be effective to alleviate the risk of hardware failures caused by frequently stopping and starting a node. Moreover, NQSV is capable of scheduling jobs with considering the startup time of a stopped node. If there is a running job whose

remaining elapsed time is shorter than the startup time, NQSV does not start up a stopped node and waits for the running job to end. As a result, it can avoid from excessively starting up a stopped node. System administrators can control such behaviors by adjusting NQSV's configuration parameters, based on the statistical behaviors of recent job requests.

There are many studies for reducing the operational burden of HPC systems. Recently, scheduling algorithms [2, 4, 6, 14, 15, 22, 20, 28] based on reinforcement learning have been proposed. Fan et al. [6] proposed an automated HPC job scheduling agent named DRAS (Deep Reinforcement Agent for Scheduling). DRAS realized HPC job scheduling features such as resource reservation and backfilling with a hierarchical neural network. DRAS performs 45% better than the algorithm used in the conventional job scheduler. However, to the best of our knowledge, there is no practical AI scheduler that can fully support Power Saving Scheduling.

Several studies such as in [3, 7, 11, 12, 21] have discussed the optimization of job scheduling parameters for maximizing QoS and system utilization. Powers [21] proposed a tool for exploring system scheduler parameters such as scheduling intervals. Kondameedi et al. [11] presented that the waiting time for jobs can be reduced by dynamically changing the queue settings. Chahal et al. [3] introduced a simulation-based scheduling method for workflow jobs. However, these studies focus only on Backfill Scheduling or Multi Queue Scheduling, and not on Power Saving Scheduling.

In this work, we develop a simulator of Power Saving Scheduling on a real-world HPC system, AOBA. Then, we survey scheduling parameters for Power Saving Scheduling and Multi Queue Scheduling to improve the system operation efficiency while reducing power consumption.

3 A Real-world HPC System and its Operation

3.1 Overall System Configuration

In this paper, we show a case study of building a Digtal Twin of an HPC system, taking the AOBA system [5] as a concrete example. Therefore, this section describes the AOBA system and its operation.

AOBA is an HPC system consisting of two subsystems. One is a vector-type computing system called AOBA-A, which consists of 72 nodes of SX-Aurora TSUBASA [10, 18]. The other is a scalar computing system called AOBA-B. In this paper, we target the job simulation of the AOBA-A system.

SX-Aurora TSUBASA is equipped with Vector Engines (VEs) for executing user applications and Vector Hosts (VHs) for running OS and hosting VEs. A collection of one or more VHs, VEs hosted by those VHs, and InfiniBand HCAs is called a Vector Island (VI) [26]. In the case of the AOBA system, one VI has one VH, eight VEs, and two InfiniBand HCAs. In this paper, a VI is referred to as a node.

3.2 Queue Configurations

The AOBA-A system provides six job queues listed in Table 1. To maximize job throughput, job queues are classified into two types. One is for jobs of using only one VE, and the other is for jobs of using multiple VEs. A job submitted to the free queue can run for one hour at a maximum. Users are supposed to select a job queue, considering the expected job execution time and the job size, which is defined by the number of VEs requested by the job. The system administrator could manually reallocate the resources to each queue. In practice, when a medium-sized job queue (sx_m in Table 1) is busy, the system administrator will allocate more resources to that queue by reducing resources of other queues.

Table 1: Queue Configurations

Usage Type	Queue name	Number of VEs per job	Allocated resources
Free	sxf	1VE	4VHs
Paid use	sxl	1VE	58VHs
	sxmix_s	1VE	
	sx_s	2-8VEs (1VH)	
	sx_m	9-64VEs (2-8VHs)	
	sx_l	64-256VEs (9-32VHs)	
Reserved for specific users	-	-	10VHs
Urgent	-	-	4VHs or 24 VHs. Shared with paid use queues.

3.3 Job Scheduling and Parameters

The AOBA system uses a proprietary scheduler, NQSV [17], developed by NEC. In this section, we describe the job scheduling and parameters.

Basic Job Scheduling NQSV's job scheduling adopts an extended algorithm of EASY backfilling [16]. Based on the resource information (the number of CPU cores, the number of VEs, estimated execution time, etc.) specified at job submission, the algorithm determines where and when to execute each job. The nodes for executing a job are selected from the resources allocated to the queue, to which the job has been submitted. NQSV has some advanced features to improve system utilization, such as the feature of starting subsequent jobs when a job finishes earlier than estimated. System administrators can adjust the following parameters and options to control EASY backfilling (the descriptions in () indicate the parameter values and options adopted in the AOBA system).

- Map Size: how far ahead in the future a job can be scheduled (one month).

- Scheduling Interval: the interval at which job scheduling is performed (30 seconds).
- Reordering Policy: a policy to determine the execution order of jobs in a queue. The supported policies are First Come First Serve (FCFS), Smallest Job First, Largest Job First, etc (the FCFS policy).
- Resource Allocation Policy: a policy to determine how to allocate nodes to jobs. There are two options; whether jobs are concentrated on as few nodes as possible, or not (the concentration policy).

Multi Queue Scheduling NQSV can virtually separate computing resources by linking each queue with a particular set of resources. NQSV considers the inter-queue priority and linked resources at job scheduling. Multi Queue Scheduling can be tuned by mainly adjusting the following parameters and options.

- Allocated Resources: the resources on which jobs in the queue are executed. In the case of the AOBA system, the resource allocation is as described in Table 1.
- Queue Type: whether a job is urgent or not. An urgent job can be executed as early as possible on the AOBA system [1].

NQSV also has some other advanced features such as limiting the number of jobs in each queue that are being executed by one user at the same time.

Power Saving Scheduling A node of the AOBA system can be in either of two states, "active state" and "power-saving state." A job can be assigned only to nodes in the active state because the power supply to nodes in the power-saving state is stopped for power saving. If no job is assigned to a node for a certain period, the power supply to the node is stopped, and the node is transiting to the power-saving state. The state transition takes a certain time period as the overhead.

Power Saving Scheduling of NQSV has two functions. The first function is to detect an idle node and change its state from active state to power-saving state. The second function is to systematically schedule a job considering the timing overhead of waking up a power-saving state node. Suppose that a job is submitted and a node to be allocated is in the power-saving state. Then, NQSV can decide the execution start time of a job by estimating how long it takes for the node to transit to the active state and thus become ready for the job execution. Power Saving Scheduling can be tuned by adjusting the parameters shown in Fig. 1 and Table 2.

4 Job Scheduling Simulation of a Real-World HPC System

In this section, we first show an overview of the job scheduling simulator, which is an extension of SLURM Simulator to accurately simulate job scheduling and power management on a real-world HPC system, AOBA-A. Then, we describe the implementation details of the simulator.

Table 2: Configurable Parameters for Power Saving

Item	Description	Setting Value
Min idle time	When the idle state continues for a certain period of time, the node is shutdown.	7200s
Margin for stop host	Time needed to stop a node.	600s
Margin for start host	Time needed to start a node.	1200s
Dcoff limit	Maximum number of times to stop nodes per day.	5
Min operation hosts	Number of nodes that should always be running	0
Estimated dcoff time	If the execution start time of a subsequent job is longer than this parameter, the node enters the power-saving state.	3600s

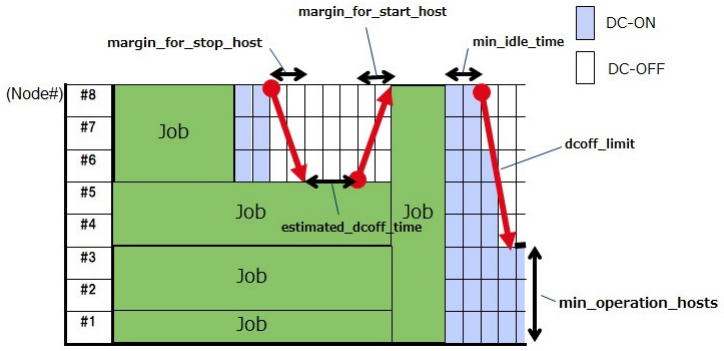


Fig. 1: Parameters of Power Saving Scheduling.

4.1 Extended SLURM Simulator

SLURM is an extensible job scheduler by using plug-ins. The scheduler used in SLURM Simulator can be replaced with a custom scheduler that uses the same scheduling algorithm as NQSV for simulating AOBA-A's job scheduling. Job scheduling policies and features of SLURM and NQSV are significantly different. For example, in SLURM, task based resources (i.e., the number of processors per task) assigned to a job can be specified at its submission. On the other hand, in NQSV, logical hosts which are a set of resources virtually divided into execution hosts can be specified by users, i.e, the number of CPU cores per a logcial host. Therefore, a SLURM plug-in needs to be developed with considering all the differences.

In this work, we develop a SLURM plug-in named NQSV Plug-in, which implements NQSV's job scheduling algorithm, and also a translator to convert the job and partition (queue) information of SLURM to those of NQSV and vice versa.

4.2 Node State Control

SLURM manages the node operation state by a power management thread. At a certain interval, the thread decides whether a node is turned on or off. On the other hand, power management of NQSV works together with job scheduling to determine when a node is turned on or off. Due to the differences above, SLURM's power management thread is not able to simulate the power management of NQSV. Therefore, we develop *Node State Simulator* for SLURM Simulator to implement the transition between the active and power-saving states in NQSV. Node State Simulator can also consider the timing overheads for starting and stopping a node due to the state transition. The extended SLURM Simulator developed in this work is called AOBA-A Job Simulator, because it is developed to be used as a Digital Twin of the AOBA-A system.

Fig. 2 shows an overview of our simulator. "Simulator" in the SLURM daemon (slurmctd) is a component provided by the original SLURM Simulator to manage time, nodes, partitions, and jobs of the simulation. Collaborating with the Simulator component, AOBA-A Job Simulator works as follows.

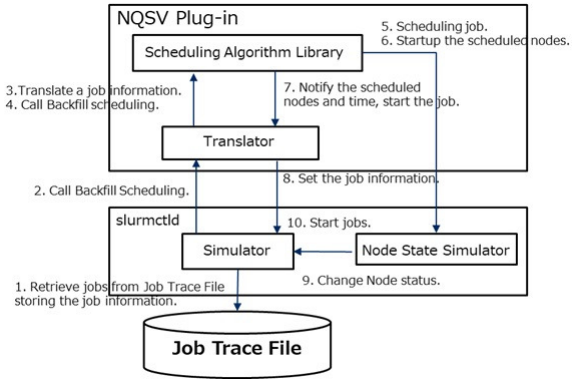


Fig. 2: Overview of AOBA-A Job Scheduler.

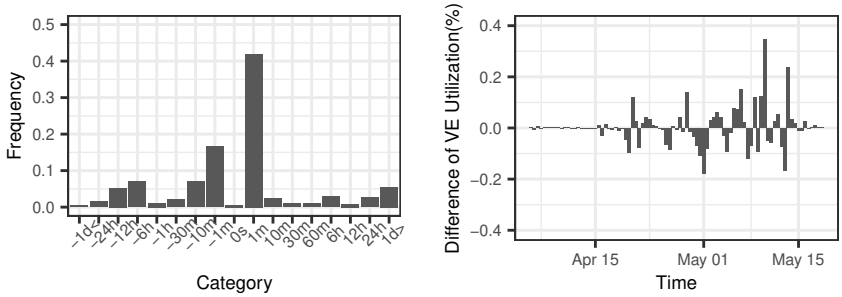
1. Simulator retrieves jobs from Job Trace File, storing the job information.
2. When the scheduling interval is expired, Simulator calls Backfill Scheduling.
3. Translator in NQSV Plug-in converts the job information for SLURM to that for NQSV.
4. Translator calls Scheduling Algorithm Library in NQSV Plug-in.
5. Scheduling Algorithm Library schedules each job by determining its execution start time and nodes which is equipped with VEs to be allocated for the execution.
6. If an allocated node is in the power-saving state, the startup time of the node is calculated, and the execution start time is delayed.
7. Scheduling Algorithm Library reports the execution start time and allocated nodes to Translator.

8. Translator converts and sends the reported information to Simulator.
9. When the startup time of the sleeping node is expired, Node State Simulator changes the node state to active.
10. Simulator starts each job at its execution start time.

The above steps are repeated to simulate the job scheduling with power management for the given job traces.

5 Simulation Accuracy of AOBA-A Job Simulator

In this section, we evaluate the accuracy of the job scheduling simulation with AOBA-A Job Simulator described in Section 4. In the evaluation, the simulation is performed using job traces, i.e., the information about actual jobs executed at the AOBA-A system. Then, the simulation results are compared with the observed job scheduling results. Initially, all the job queues are empty in the simulation. To start the simulation with the same initial condition as the job traces, we use the data of 10,000 jobs observed right after the system maintenance period, in which no job is submitted.



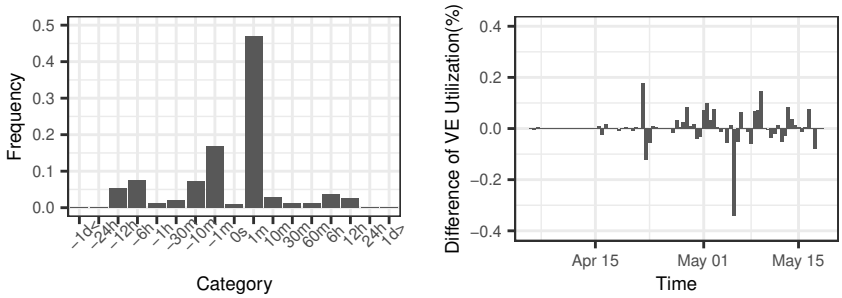
(a) Differences of start time. (b) Differences of VE utilization.

Fig. 3: Simulation with original job traces. Differences between observed and simulated scheduling results.

Fig. 3 shows the simulation results. Fig. 3(a) shows the histogram of differences in start time between the observed and simulation results. The vertical axis shows the frequency, and the horizontal axis shows the class of the difference in starting time. A negative difference means that the job execution has been delayed in the simulation. For example, the frequency of class "-1m" means the number of jobs, for which the time difference between the observation and simulation is in between one second to one minute. We can see that about 60% of the jobs are in the classes of "-1m," "0," and "1m." Therefore, the simulator can accurately simulate the majority of jobs recorded in the job traces. Fig. 3(b)

shows the differences between the observed utilization ratio of VEs and the corresponding simulated ratio. The vertical axis indicates the utilization ratio, which is calculated by subtracting the simulation data from the original data, and the horizontal axis shows the time sequence. We can see that there is no difference in the VE utilization rate at first, and the difference gradually becomes visible for the job scheduling on April 15 and later. In practice, the simulation results are accurate enough except for some special cases described below.

The current implementation of AOBA-A Job Simulator supports only the core scheduling algorithm of NQSV, and does not support some features to strictly simulate the AOBA-A system operation. One example of unsupported features is that NQSV allows users to specify the execution start time at job submission. Another example is the workflow support to guarantee that one job is executed after another job. The difference in waiting time between observed and simulated results becomes large for some jobs, and exceeds one day for about 5% of jobs in Fig. 3(a). This is because the workflow management feature delayed to start executing the job until the execution of its preceding job is finished. The information about use of such unsupported features is not recorded in the job traces. Therefore, we do not implement the features in AOBA-A Job Simulator at present.



(a) Differences of start time. (b) Differences of VE utilization.

Fig. 4: Simulation with corrected job traces. Differences between observed and simulated scheduling results.

For considering the workflow management, job traces are modified. Specifically, the submission time of a job is replaced with the execution start time recorded in the job traces if the difference between the submission time and the execution start time is longer than 12 hours. The simulation results using the corrected job trace are shown in Fig. 4. Comparing Fig. 3(a) and Fig. 4(a), the frequencies of jobs with larger time differences are reduced. The same can be seen in Fig. 3 (b) and Fig. 4 (b). Therefore, these results indicate that data correction under a simple assumption allows the proposed simulator to simu-

late the job scheduling more accurately, even though the workflow management information is not recorded in the job traces.

In Fig. 4(b), there is a time period in which the differences between the observed utilization ratio and the simulated one is -0.3%. This is due to the execution of urgent jobs [1]. The urgent job suspends some running jobs with a lower priority. However, both of regular and urgent jobs are kept running in the observed results. As a result, the utilization ratio appears to be large in the observed job traces.

In Fig. 3(a), there are jobs whose simulation results are more than 10 minutes faster than the observed results. This is because another feature of NQSV is not supported by the proposed simulator. NQSV limits the number of concurrently running jobs of each user to avoid the computing resource from being occupied by specific users. The execution of some jobs of a user would be delayed at the AOBA system if the user has submitted too many jobs. However, the proposed simulator does not support this feature, and excessively submitted jobs are not delayed in the simulation. As a result, there is the difference between the observation and simulation.

Although no hardware failure happened during the simulated period, it could happen in practice. Currently, the proposed simulator does not consider any hardware failures. Since the failure is recorded in the job traces, it would be effective to make the simulation more accurate by simulating the failures in the future.

All the results mentioned above demonstrate that AOBA-A Job Simulator can reproduce the job scheduling results observed at the AOBA-A system. It should be emphasized again that the AOBA-A system is in operation with the power saving mechanism. Therefore, the evaluation results clearly demonstrate that the proposed simulator can well simulate not only the job scheduling but also the power saving mechanism of a real-world HPC system. With the simulator, we can explore the optimal configuration of scheduling and power saving parameters as discussed in Section 6.

6 Survey of Scheduling and Power Saving Parameters

6.1 Parameter settings and job submission behaviors

Scheduling policies and parameters related to scheduling and power saving would usually be decided at the system design. The same applies to the allocation of resources to the job queues. When a queue is extremely busy, the system administrators may manually change the resource allocation to increase the resource for the busy queue. However, changes in scheduling and power saving parameters could have a significant impact on the QoS visible from users, and thus the system administrators are generally conservative to change the parameters during the system operation. Those parameters would empirically be configured based on past operation experience, such as the job submission history, and thus the parameter configuration is not necessarily optimal for the system. Therefore, a

parameter survey with an accurate job scheduling simulator is helpful to check whether the current parameter configuration is appropriate for the system operation that could change dynamically. In this section, we describe a parameter survey of scheduling and power saving parameters.

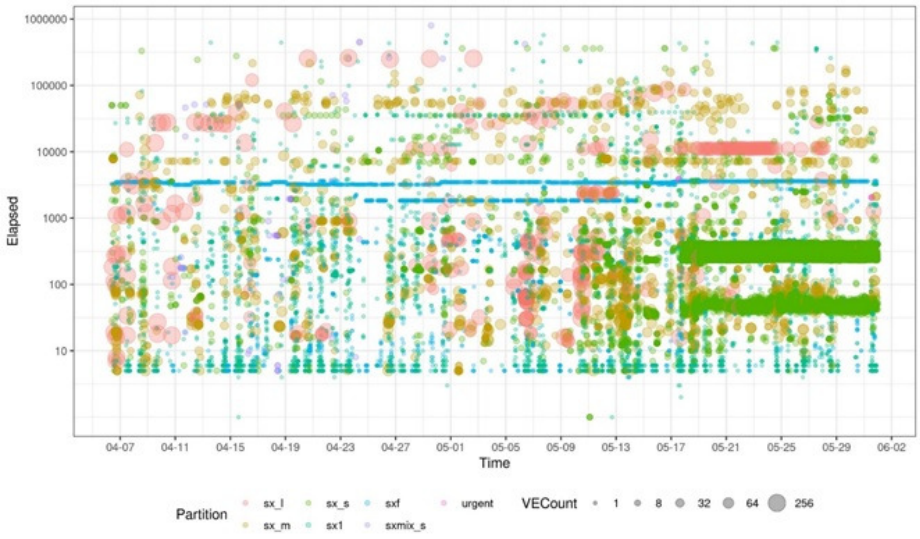


Fig. 5: Submission times, elapsed times and sizes of jobs.

First, we describe the statistical analysis of job submissions observed in a real-world system. Fig. 5 shows the job submission times, elapsed times, and sizes of jobs submitted to the AOBA-A system in April and May, 2021. The vertical axis indicates the elapsed time of each job, the horizontal axis shows the time sequence, and the circle size represents the number of VEs requested by the job. Overall, more jobs were submitted and executed in the second half of May. There is a trend that users are likely to submit short-running jobs to the free queue in the daytime but not on holidays. In addition, some users keep submitting jobs at a regular interval by probably running a job submission script, and hence their jobs are constantly executed on the system regardless of the date and time.

In this paper, the data in Fig. 5 are split into two, off period and busy period data. The data from April to mid-May can be seen as the off period job submission data, while the data in late-May are the busy period ones. Since the system is not so busy in the off period, some of compute nodes could be turned off for power saving. With the off period job data, hence, we can discuss the effect of power saving on the system operation efficiency by adjusting relevant parameters in the simulation. On the other hand, with the busy period data, we

can discuss if the power saving mechanism has a negative effect on the system operation efficiency when the system is keeping busy.

6.2 Parameter survey of job scheduling in off period

Using the data of 10,000 jobs submitted from April 6 to May 17, we investigate the effects of changing power saving parameters on the system operation efficiency. The parameter configurations used in the evaluation are listed in Table 3. In the table, Min Idle Time means the minimum idle time, and a node enters the power-saving state if the node is idle for the minimum idle time. Dcoff limit means the maximum number of times for each node to enter the power-saving state. Nodes and power supply units could be damaged by too frequently turning on and off the nodes, and thus the number of times for each node to enter the power-saving state per day is limited at the AOBA system. Est. dcoff time means the estimated DC off time, which is the minimum sleep time. If a node is expected to be unused for the estimated DC off time or longer, the node enters the power-saving state.

Table 3: Configurable Settings for Power Saving

Parameter Settings	Power Saving	Min Idle Time(s)	Dcoff limit	Est. dcoff time(s)
AOBA	On	7200	5/day	3600
A	Off	7200	5/day	3600
B	On	3600	5/day	3600
C	On	10800	5/day	3600
D	On	7200	200/day	3600
E	On	7200	5/day	1800
F	On	7200	5/day	7200

The power consumption of a node has been measured during the job execution, and is not equal to the value shown in the specification sheet of SX-Aurora TSUBASA. Specifically, the power consumption of an idle node is about 600W, while the power consumption reaches about 2100W when the node runs a well-vectorized job. In addition, the startup time and shutdown time of a node are set to be 60 seconds and 120 seconds, respectively. These values are decided based on the actual measurement, instead of using not the scheduler parameters "Margin for start host" and "Margin for stop host" in Table 2.

The simulation results with different parameter configurations are shown in Fig. 6. The left vertical axis indicates the average waiting time of a job, the right vertical axis indicates the total energy consumption, and the horizontal axis shows parameter configurations. In the figure, the differences between AOBA and Condition A shows the effects of enabling the power saving mechanism, meaning that the mechanism can save about 11 MWh without increasing the average waiting time. The results with Condition B show that if the minimum

idle time is decreased, the energy consumption is further reduced by 1.2 MWh, but the average waiting time becomes longer, degrading the system operation efficiency. The results with Condition D show that increasing the maximum number of times for each node to enter the power-saving state has no effect on the system operation efficiency including the energy consumption. The results with Condition E show that decreasing the minimum sleep time by 30 minutes has no effect. On the other hand, the results with Condition F show that increasing the minimum sleep time by 1 hour makes the average waiting time longer. This is because the nodes that have a running single-node job and a multi-node job scheduled for subsequent time periods can enter the power-saving state less frequently. The other jobs are scheduled before the multi-node job. If the single-node job finishes earlier than estimated, the execution of the multi-node job is inhibited by the newly scheduled jobs.

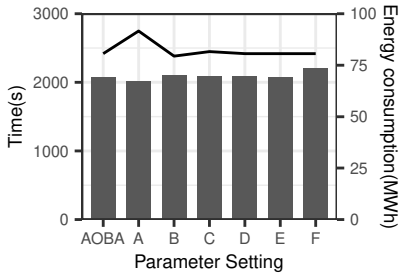


Fig. 6: Results of off-period simulation. The bars indicate the average waiting time of a job, and the line indicates the total energy consumption.

6.3 Parameter survey of job scheduling in busy period

The job traces from May 28 to 29 are shown in Fig. 7. The meanings of the vertical axis, the horizontal axis, and the circle size are the same as in Fig. 5. The figure shows that various jobs were intensively submitted in the daytime on May 28. Another characteristic point is that small jobs are submitted to the sx_s queue continuously the whole day. In addition, long-running jobs of small and medium sizes were submitted on May 28.

A submitted job can be in either of running or queued state, while a node can be in either of active or power-saving state. The state changes of jobs and nodes simulated with AOBA's parameter configuration are shown in Fig. 8. The vertical axis indicates the number of jobs or nodes in each state, and the horizontal axis shows the time sequence. Due to the existence of long-running jobs, a large number of small jobs are in the queued state and waiting for execution. Although the number of queued jobs almost reaches 500 in Fig. 8(a), some of the nodes are always in the power-saving state in Fig. 8(b). This is because resource allocation

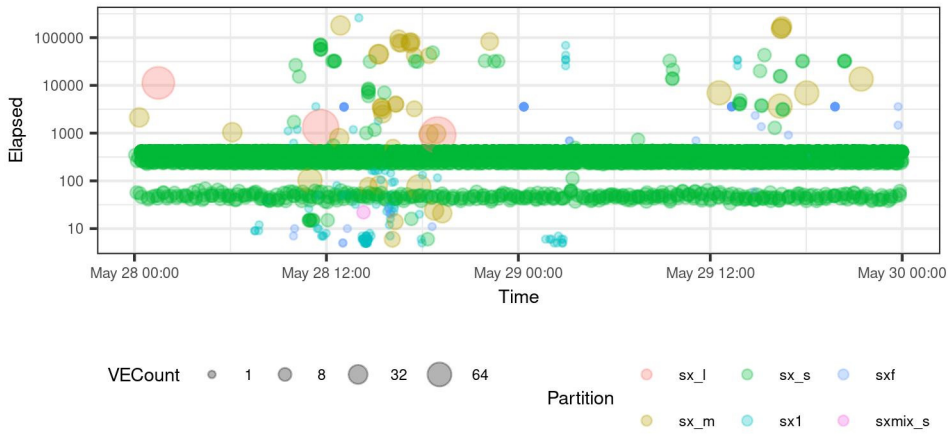


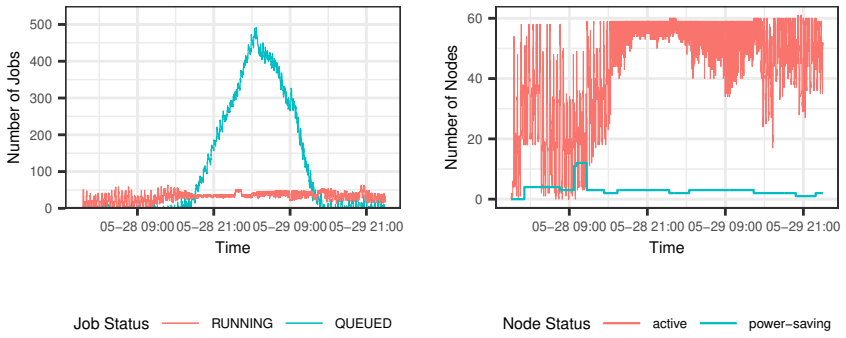
Fig. 7: Job pattern of May 28 to 29.

is not appropriate for the situation. As shown in Table 1, four nodes are available for jobs in the sxf and sx1 queues at the AOBA system, while 58 nodes are allocated to queues of parallel jobs. Since many parallel job are submitted, all the 58 nodes are in use on May 28. On the other hand, only short-running jobs are submitted to sx1 and sxf, four nodes allocated for those queues could sometimes be idle and in the power-saving state. There is room to improve the utilization ratio by allocating those sleeping nodes to the queues, in which many jobs are waiting.

Table 4: Resource Balancing in the Simulation

Item	Description
(1) Move 2 nodes (2) Move 3 nodes	Move 2 or 3 nodes of sx1 and sxf to sx_s, sx_m and sx_l queue.
(3) Share 2 nodes (4) Share 3 nodes	Share 2 or 3 nodes of sx1 and sxf to sx_s, sx_m and sx_l queue.
(5) Share all nodes	Paid use queue and free queue uses all nodes.

Motivated by the above discussion, we improve the system operation efficiency by adjusting the resource balance among queues as shown in Table 4. In the table, "Move 2 nodes" means that two nodes out of four are moved from sx1 and sxf to queues of parallel tasks in Table 1. "Share 2 nodes" means that two nodes out of four are shared by all the queues, and the other two nodes remain dedicated to sx1 and sxf. "Share all nodes" means that all the four nodes originally allocated to sx1 and sxf are shared by all queues.



(a) Job status. (b) Node status.

Fig. 8: Transition of the numbers of running and waiting jobs, and the number of nodes in each power state.

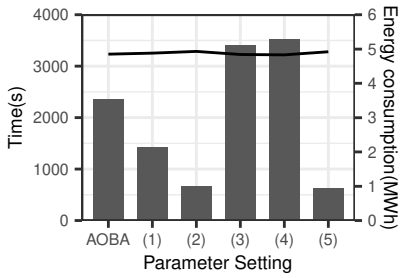


Fig. 9: Results of busy-period simulation.

The results of scheduling parameter tuning are shown in Fig. 9. The left vertical axis indicates the average waiting time of a job, the right vertical axis indicates the energy consumption, and the horizontal axis shows the resource allocation. The average waiting time decreases as the number of nodes increases by moving nodes from the sx1 and sxf queues to the others. On the other hand, the average waiting time increases if two or three nodes are shared by multiple queues, worsening the system operation efficiency. The average waiting time is minimized if all the nodes are shared by queues. In this simulation, we can see that the average waiting time is significantly reduced with a little increase in the energy consumption in "Move 3 nodes" and "Share all nodes" cases. In the following discussion, we focus on the "Move 3 nodes" case because system administrators will likely avoid using "Share all nodes" to ensure the QoS of paid use by limiting resources for the free queue.

Considering the results in Fig. 6, we simulate the job scheduling by changing the minimum idle time in the case, as shown in Fig. 10. The vertical and horizon-

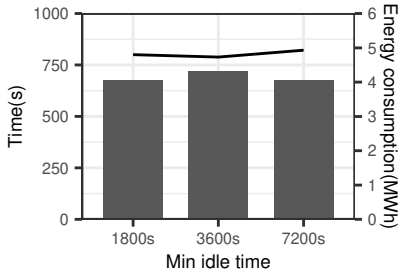
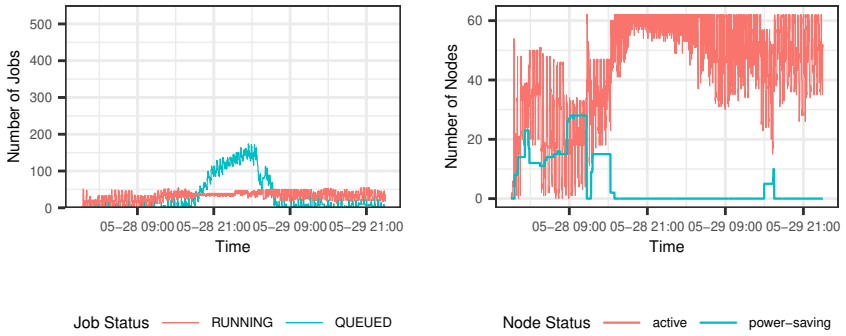


Fig. 10: Results of simulation when changing parameters and resource allocation.

tal axes are the same as those in Fig. 9. We can see that the energy consumption is smaller than that with the minimum idle time of 7200 seconds. The scheduling parameter tuning of the minimum idle time is possible even when the resource balance is adjusted.



(a) Job status. (b) Node status.

Fig. 11: Status transition of jobs and nodes when resource balance and power saving parameter is changed.

The status changes in jobs and nodes in the case with the minimum idle time of 1800 seconds are shown in Fig. 11. The vertical and horizontal axes are the same as those in Fig. 8. In Fig. 11(a), the number of jobs in the queued status is decreased compared to that in Fig. 8(a), indicating that this resource allocation can start jobs earlier than that in the AOBA-A system. In Fig. 11(b), we can see that the number of nodes in the power-saving status has increased compared to that in Fig. 8(b). Even in the busy period, the average waiting time can be reduced from 2370 seconds to 678 seconds, while reducing the energy consump-

tion from 4.85 MWh to 4.80 MWh in comparison with the parameter settings adopted by the AOBA-A system in Table 3. These results clearly indicate that we are able to achieve better efficiency with less energy consumption than the current parameter settings of the AOBA system in those two days if the resource balance and parameter configurations can properly be adjusted. In this way, the proposed simulator is helpful for system administrators to adjust the scheduling and power saving parameter configuration to achieve better system operation efficiency and/or less energy consumption. Furthermore, these results show the effectiveness of a Digital Twin of a real-world HPC system because it predicts daily, weekly, and monthly data and suggests suitable parameters to system administrators.

7 Conclusions

In this paper, we have first implemented a component for a job scheduler to support the special features, and thus to build a "Digital Twin" of a real-world HPC system, named AOBA. By using the Digital Twin with actual job traces, a system administrator can check whether the system is being efficiently operated as expected. Although we have implemented only the core algorithm of NQSV, the simulation results are accurate enough except for some special cases such as hardware failures, and hence the proposed simulator is useful to explore the optimal parameter configurations for individual situations.

This paper has shown a use case of exploring appropriate scheduling and power saving parameters. In the use case, we found that there are more appropriate parameter configurations, which can reduce the job waiting time by 70% at most and the energy consumption by 1.2% at most when the system is busy. By exploiting such a Digital Twin, therefore, it is demonstrated that a system administrator can properly adjust various parameters without disturbing the system operation.

In order to make the simulation more accurate, we will implement unsupported features of NQSV such as an urgent job execution and hardware failures. A more accurate power model would also be effective to improve the simulation accuracy. Moreover, we will consider how to identify underlying job patterns in a short period of time and find an appropriate parameter configuration for the patterns by using machine learning. These will be further discussed in our future work.

Acknowledgements

This work is partially supported by MEXT Next-Generation High-Performance Computing Infrastructures and Applications R&D Program R&D of a Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications.

References

1. Agung, M., Watanabe, Y., Weber, H., Egawa, R., Takizawa, H.: Preemptive parallel job scheduling for heterogeneous systems supporting urgent computing. IEEE Access **9**, 17557–17571 (2021). https://doi.org/10.1109/ACCESS.2021.3053162
2. Baheri, B., Guan, Q.: Mars: Multi-scalable actor-critic reinforcement learning scheduler. ArXiv **abs/2005.01584** (2020)
3. Chahal, D., Mathew, B., Nambiar, M.: Simulation based job scheduling optimization for batch workloads. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. p. 313âĂŞ320. ICPE '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3297663.3310312, https://doi.org/10.1145/3297663.3310312
4. Cunha, R.L.F., Chaimowicz, L.: Towards a common environment for learning scheduling algorithms. 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) pp. 1–8 (2020)
5. Egawa, R., Fujimoto, S., Yamashita, T., Sasaki, D., Isobe, Y., Shimomura, Y., Takizawa, H.: Exploiting the potentials of the second generation sx-aurora tsubasa. In: 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). pp. 39–49 (2020). https://doi.org/10.1109/PMBS51919.2020.00010
6. Fan, Y., Lan, Z., Childers, T., Rich, P., Allcock, W., Papka, M.E.: Deep reinforcement agent for scheduling in hpc (2021)
7. Gaussier, E., Lelong, J., Reis, V., Trystram, D.: Online tuning of easy-backfilling using queue reordering policies. IEEE Transactions on Parallel and Distributed Systems **29**(10), 2304–2316 (2018). https://doi.org/10.1109/TPDS.2018.2820699
8. IBM Spectrum LSF Suites: https://www.ibm.com/products/hpc-workload-management
9. Klusàček, D., Rudovà, H.: Alea 2 - job scheduling simulator. SIMUTools 2010 - 3rd International ICST Conference on Simulation Tools and Techniques p. 61 (01 2010). https://doi.org/10.4108/ICST.SIMUTOOLS2010.8722
10. Komatsu, K., Momose, S., Isobe, Y., Watanabe, O., Musa, A., Yokokawa, M., Aoyama, T., Sato, M., Kobayashi, H.: Performance evaluation of a vector supercomputer sx-aurora tsubasa. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 685–696 (2018). https://doi.org/10.1109/SC.2018.00057
11. Kondameedi, V., Vadhiyar, S.S.: Adaptive hybrid queue configuration for supercomputer systems. 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) pp. 90–99 (2017)
12. Krishnamurthy, D., Alemzadeh, M., Moussavi, M.: Towards automated hpc scheduler configuration tuning. Concurrency and Computation: Practice and Experience **23** (2011)
13. Maiterth, M., Koenig, G., Pedretti, K., Jana, S., Bates, N., Borghesi, A., Montoya, D., Bartolini, A., Puzovic, M.: Energy and power aware job scheduling and resource management: Global survey âĂŤ initial analysis. In: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 685–693 (2018). https://doi.org/10.1109/IPDPSW.2018.00111
14. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot

Topics in Networks. p. 50âĂŞ56. HotNets '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/3005745.3005750, https://doi.org/10.1145/3005745.3005750
15. Mao, H., Schwarzkopf, M., Venkatakrishnan, S.B., Meng, Z., Alizadeh, M.: Learning scheduling algorithms for data processing clusters. In: Proceedings of the ACM Special Interest Group on Data Communication. p. 270âĂŞ288. SIGCOMM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341302.3342080, https://doi.org/10.1145/3341302.3342080
16. Mu'alem, A., Feitelson, D.: Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. Parallel and Distributed Systems, IEEE Transactions on **12**, 529 – 543 (07 2001). https://doi.org/10.1109/71.932708
17. NEC Network Queuing System V (NQSV) User's Guide [Introduction]: https://www.hpc.nec/documents/nqsv/pdfs/g2ad01e-NQSVUG-Introduction.pdf
18. NEC SX-Aurora TSUBASA: https://www.nec.com/en/global/solutions/hpc/sx
19. OpenPBS: https://www.openpbs.org/
20. Peng, Y., Bao, Y., Chen, Y., wu, C., Meng, C., Lin, W.: Dl2: A deep learning-driven scheduler for deep learning clusters. IEEE Transactions on Parallel and Distributed Systems **PP**, 1–1 (01 2021). https://doi.org/10.1109/TPDS.2021.3052895
21. Powers, S.: A study of the impact of scheduling parameters in heterogeneous computing environments. Proceedings - Winter Simulation Conference **2015**, 933–942 (01 2015). https://doi.org/10.1109/WSC.2014.7019953
22. Ryu, B., An, A., Rashidi, Z., Liu, J., Hu, Y.: Towards topology aware pre-emptive job scheduling with deep reinforcement learning. In: Proceedings of the 30th Annual International Conference on Computer Science and Software Engineering. p. 83âĂŞ92. CASCON '20, IBM Corp., USA (2020)
23. Simakov, N., Innus, M., Jones, M., Deleon, R., White, J., Gallo, S., Patra, A., Furlani, T.: A Slurm Simulator: Implementation and Parametric Analysis, pp. 197–217. High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation (01 2018). https://doi.org/10.1007/978-3-319-72971-8_10
24. SLURM: https://www.schedmd.com/
25. Sulistio, A., Cibej, U., Venugopal, S., Robic, B., Buyya, R.: A toolkit for modelling and simulating data grids: An extension to gridsim. Concurrency and Computation: Practice and Experience **20**, 1591–1609 (09 2008). https://doi.org/10.1002/cpe.1307
26. SX-Aurora TSUBASA Architecture Guide Revision 1.1: https://www.hpc.nec/documents/guide/pdfs/Aurora_ISA_guide.pdf
27. Top500: https://www.top500.org/
28. Zhang, D., Dai, D., He, Y., Bao, F.S., Xie, B.: Rlscheduler: An automated hpc batch job scheduler using reinforcement learning. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '20, IEEE Press (2020)