PROCEEDINGS OF SCIENCE

Architecture of Job Scheduling Simulator for Demand Response Based Resource Provisioning

Shogo Matsui[*a], **Yasuhiro Watashiba**[b], **Susumu Date**[b], **Jason Liu**[c], **Kaname Harumoto**[d] **and Shinji Shimojo**[b]

[a] *Graduate School of Information Science and Technology, Osaka University, Japan*
[b] *Cybermedia Center, Osaka University, Japan*
[c] *Knight Foundation School of Computing and Information Sciences, Florida International University, USA*
[d] *Institute for Datability Science, Osaka University, Japan*
E-mail: matsui.shogo@ais.cmc.osaka-u.ac.jp

We study a new service model based on the Demand Response (DR) resource provisioning at High Performance Computing (HPC) centers. This DR-based resource provisioning model allows administrators of HPC centers to provide computing services with incentives to users to compensate for the performance loss due to power saving operations. In a power conservation mode, a job's performance may decrease, both in terms of a job waiting time and a job execution time. With DR-based resource provisioning, the submitted jobs are divided into two categories, allowed jobs and disallowed jobs, depending on the user's tolerance in the performance degradation. The allowed jobs, if indeed affected by the power saving operations, will receive compensation in accordance with an incentive system which determines the reward to the user. For designing an appropriate demand response model, we need to focus on the increase in the job's execution time and the job's waiting time, and the corresponding decrease in the power consumption. These are important factors in deriving an incentive system. Currently, no existing approaches can reliably quantify the effectiveness and the contribution of these factors in HPC job scheduling and resource provisioning. In this paper, we propose a newly developed job scheduling simulator that can evaluate DR-based resource provisioning approach under various operating conditions. We designed and implemented the job scheduling simulator for HPC demand-response resource provisioning using a general-purpose discrete-event simulator. Our experiments show that the job scheduling simulator can properly represent the demand response resource provisioning using different job scheduling scenarios.

International Symposium on Grids & Clouds 2020, ISGC2021
22-26 March, 2021
Academia Sinica, Taipei, Taiwan (online)

[*] Speaker.

PoS(ISGC2021)006

1. Introduction

High Performance Computing (HPC) centers operate large-scale cluster systems composed of many computing nodes, which are allocated as computing resources to fee-paying users. The power consumption of large-scale cluster systems is of important concern. To conserve power, administrators at HPC centers may choose to either shut down an idle part of the computing cluster or slow down the CPU frequency on selected computing nodes. These power conservation operations performed by the administrators may result in a decrease in available computing resources or performance degradation in the CPU processing. As such, these power conservation operations at HPC centers are called "fallbacks". A fallback may be in the form of prolonged waiting time of submitted jobs or prolonged execution time of running jobs. Most existing job schedulers, which manage submitted jobs and control the provisioning of computing resources to jobs, do not take into account the fallbacks due to power conservation operations.

A new service model based on Demand Response (DR), which attempts to better match power demand with power supply, has attracted the attention of operators and researchers at High Performance Computing (HPC) centers [1, 2, 3]. In this new service model, which we call DR-based resource provisioning, submitted jobs are divided into two categories, allowed jobs and disallowed jobs, based on the user input. The users declare whether the submitted jobs can tolerate fallbacks if power conservation is applied. The disallowed jobs will remain the same as traditional HPC jobs. They do not participate in power conservation operations. Consequently, they do not suffer from fallbacks. The allowed jobs will participate in power conservation operations. These jobs can tolerate longer turnaround times. Administrators at HPC centers devise an incentive system, providing compensations to users who submitted allowed jobs impacted by fallbacks. In doing so, the HPC centers encourage users to consider submitting more allowed jobs.

To design such an incentive system to encourage users to submit as many allowed jobs as possible while achieving larger profit from the reduced energy cost, one needs to consider important factors, including the overall reduction in power consumption of large-scale cluster systems and the increase in the jobs' waiting time and execution time. These factors depend on the fallback methods, the structure of large-scale cluster systems, and the characteristics of submitted jobs. They play an important role in determining a design of the incentive system. Currently, no mechanisms exist to reliably measure the impact of these factors on DR-based resource provisioning methods under various HPC operating conditions. Although job scheduling simulators are widely used as mechanisms to evaluate how submitted jobs are processed on large-scale cluster systems, most job scheduling simulators do not have functionalities to capture the power consumption of large-scale cluster systems, the increased job waiting time and execution time, and their impact on DR-based resource provisioning. This paper presents a new job scheduling simulator with functionalities for managing jobs in accordance with DR-based resource provisioning, in an effort to study and evaluate the effectiveness of various design alternatives for proper incentive systems.

The rest of this paper is organized as follows. Section 2 describes the operations of an HPC center and DR-based resource provisioning as background. Section 3 mentions an analysis of the required functionalities for job processing in DR-based resource provisioning and proposes our job scheduling simulator. Section 4 describes the implementation of the proposed job scheduling simulator. Section 5 presents case study results to evaluate the job scheduling simulator. Section 6

concludes the paper with a summary.

2. Background

2.1 Operations of a High Performance Computing center

In High Performance Computing (HPC) centers that operate large-scale cluster systems, the power consumption of large-scale cluster systems is an important concern. An HPC center provides computing resources for fee-paying users from large-scale cluster systems composed of many computing nodes. Cluster systems consume a huge amount of power. Table 1 shows a list of TOP500 [4] computer systems which are top-ranked on theoretical peak performance and the number of processor cores. The table shows the power consumption of these cluster systems. For example, Tianhe-2A, ranked at 6^{th} place, consumes as much electricity as to power roughly twenty thousand homes. Power consumption undoubtedly is an important issue for HPC centers.

HPC centers attempt to reduce power consumption by performing fallback operations. Two main fallback methods exist: Dynamic Voltage Frequency Scaling (DVFS) [5] and "node stop". DVFS is a method of reducing CPU power consumption by lowering CPU frequency and power voltage supplied to a CPU. Node stop is a method to suppress a power consumption of a cluster system by making computing nodes turning off or idling.

Unfortunately, fallbacks can potentially increase the waiting time of submitted jobs (due to the unavailability of computing nodes that have been turned off or idle) or the execution time of running jobs (due to the reduced CPU processing power at lower frequency or voltage). In general, since users prefer a short turnaround time of jobs, an increase in job's waiting time and execution time presents an unfavorable proposition to the users.

2.2 Demand Response Based Resource Provisioning

To accommodate power conservation operations with fallbacks described in Sec. 2.1, a new service model, called Demand Response (DR)-based resource provisioning, has attracted considerable attention recently [1, 2, 3]. The DR-based resource provisioning is a service model that applies demand response to limit power consumption and balance supply and demand between power companies and power consumers. Power companies maintain a balance by adjusting the supply of power according to the power consumers' declarations for permission to reduce the supply of power. The more power consumers that allow power supply reductions, the easier it is for

Rank	System	Power [kW]
1	Supercomputer Fugaku	29,899
2	Summit	10,098
3	Sierra	7,438
4	Sunway	15,371
5	Selene	2,646
6	Tianhe-2A	18,482

Table 1: part of the TOP500, NOVEMBER 2020.

the power companies to balance the supply and demand. To encourage power conservation, an incentive system is in place that generally translates to a reduced cost for the consumers to willingly participate in the demand response scheme.

DR-based resource provisioning contains new actions in addition to existing resource provisioning. Actions for DR-based resource provisioning are shown in Fig. 1. In DR-based resource provisioning, a user declares whether she would allow her jobs to participate in the power conservation operation. If so, it is expected that the waiting time of the submitted jobs and the execution time of the running jobs may take longer if power conservation is activated. In Fig. 1, "User 1" declares that his job is not allowed to participate in power conservation, while "User 2" allows it. In this case, the allocation of computing resources to User 2's job might be delayed due to unavailability or the computing resources with reduced processing performance might be allocated to User 2's job. To compensate for User 2's participation, User 2 will be provided with a reward, maybe in terms of reduced usage fee, if the job suffers from prolonged waiting time or execution time due to power conservation. The system incorporates an incentive system to calculate the rewards given to users affected by fallbacks.

In DR-based resource provisioning, it is important to design an incentive system so that more users are willing to allow their jobs to participate in power conservation. To design such an incentive system, it is necessary to take into account the benefit of both a HPC center (in terms of profit) and its users (in terms of rewards). There are important factors to consider. For HPC centers, one needs to consider the overall reduction in power consumption of large-scale cluster systems. For HPC users, one needs to consider the increased turn-around time of the allowed jobs (including both the waiting time of submitted jobs and the execution time of running jobs).

Currently, no mechanisms exist to reliably measure the impact of these factors on DR-based resource provisioning, under various HPC operating conditions. These factors depend on the fallback methods, the structure of large-scale cluster systems, and the characteristics of submitted jobs. The structure of a cluster system is determined by the number and type of computing nodes and a type of interconnect between the computing nodes. The type of a computing node is defined by the type of a CPU, the use of accelerators, and the memory capacity. The characteristics of submitted jobs depend on jobs themselves as well as the behavior of the users (such as job arrivals and job types, etc.) Therefore, we aim to realize a mechanism for HPC centers to study and evaluate demand response based resource provisioning and in particular, the design of proper incentive systems. For this purpose, we developed a job scheduling simulator so that we can examine the handling of the different structures of large-scale cluster systems and the various situations in job

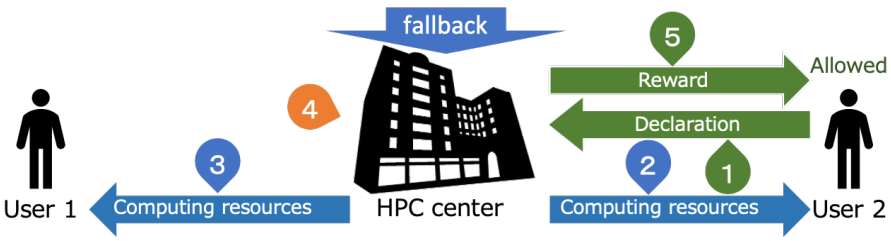

Figure 1: Actions for DR-based resource provisioning.

processing.

3. Job Scheduling Simulator

3.1 Existing Job Scheduling Simulator

Many job scheduling simulators have been researched and developed to evaluate how jobs are allocated in computer systems which include large-scale cluster systems. For example, the gem5-gpu [6] is a mechanism to model and simulate large-scale cluster systems composed of computing nodes with GPU. MERPSYS [7] is a mechanism to simulate the execution time of the message-passing parallel applications on large-scale cluster systems.

Obaida et al. proposed a job scheduling simulator using Performance Prediction Toolkit (PPT) [8]. It supports rapid assessment and performance prediction of jobs. The job scheduling simulator can deal with various job scheduling and task mapping algorithms. Buyya et al. proposed GridSim to evaluate the basic configuration and scheduling algorithms in a grid environment [9]. Skimakov et al. proposed the Slurm Simulator to simulate a large-scale cluster system where jobs are managed by the job scheduler Slurm [10]. It can assist in the selection of Slurm parameters. These existing job scheduling simulators do not have a functionality for managing jobs in DR-based resource provisioning manner.

3.2 DR-based Job Scheduling Simulator

We propose a job scheduling simulator with functionalities for managing jobs in DR-based resource provisioning manner and evaluating the effectiveness of alternative incentive systems. To realize the proposed job scheduling simulator, a functionality required for job processing is derived against the job processing flow of the existing resource provisioning. The structure and processing flow of the job scheduling simulator with functionalities for DR-based resource provisioning is shown in Fig. 2.

As seen in Fig. 2, the proposed job scheduling simulator is composed of four components: job queue, resource allocator, status manager, and computing resources. The job queue stores submitted jobs until they are executed. The resource allocator is used to judge whether it is possible to allocate computing resources to a job in the job queue and allocates computing resources to that job. The status manager has the role of monitoring and changing the status of computing resources. The status of computing resources is categorized into two types: changeable status and unchangeable status. The changeable status includes the number of available computing resources and the processing performance of CPUs in each computing node. The unchangeable status includes the power consumption of the large-scale cluster system and the job ID allocated to each computing resource. The INPUT illustrated in Fig. 2 has four elements: "jobsets", "power limitation", "fallback method", and "Machine config". The jobsets and the Machine config are generally required by the job scheduling simulator. The power limitation and the fallback method are for managing jobs in DR-based resource provisioning. The jobsets describe the jobs which contain the following information for each job: the job ID, the job submission time, the job execution time, and the number of computing nodes required. The Machine config is the structure of the large-scale cluster system. A power limitation consists of values of the upper power limit that a large-scale cluster

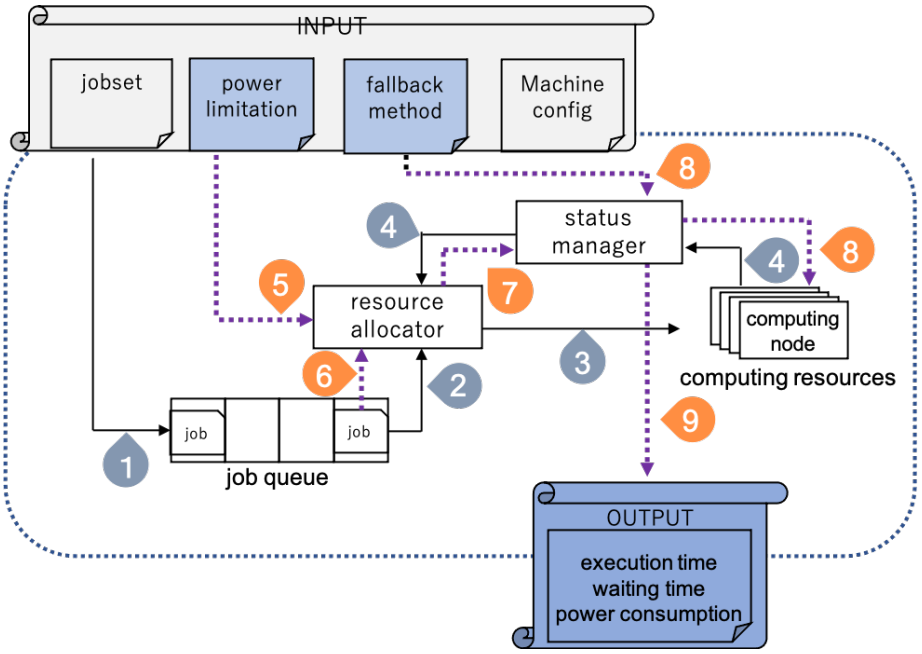

Figure 2: Structure and job processing flow of job scheduling simulator.

system can consume at a given time. A power limitation is used as a threshold for switching to the fallback. The fallback method, such as DVFS and node stop described in Sec. 2.1, is a way to reduce the power consumption of a large-scale cluster system so that they can operate within the power limitation. The OUTPUT illustrated in Fig. 2 is defined as the indexes in this study. The OUTPUT includes the job execution time, the job waiting time, and the power consumption of large-scale cluster systems.

The arrow 1, arrow 2, arrow 3, and arrow 4 in Fig. 2 are processes for existing job processing. The arrow 1 is a process of placing jobs from the jobsets to the job queue in the order of submission time. The arrow 2 is a process by which the resource allocator judges whether or not it is possible to allocate computing resources to the job in the job queue. The arrow 3 is a process in which computing resources the resource allocator allocates to a job. The arrow 4 is a process in which the status manager monitors the status of the computing resources. Such status is provided to the resource allocator on demand.

The arrow 5, arrow 6, arrow 7, arrow 8, and arrow 9 in Fig. 2 are processes which are required for DR-based resource provisioning. The arrow 5 is a process in which the resource allocator receives the power limitation and then determines whether it is necessary to switch to the fallback mode. The arrow 6 is a process in which the resource allocator determines whether a job is allowed to run on computing nodes under the fallback mode. The arrow 7 is a process in which the resource allocator instructs the status manager to update the computing resources. When the fallback method is DVFS, this instruction is to change the processing performance of CPUs which consists of the computing resources allocated to a job. If an adopted the fallback method is node stop, the instruction is to make some computing nodes turned off or idle. The arrow 8 is a process in which the status manager refers to the fallback method and then changes the status of the computing re-

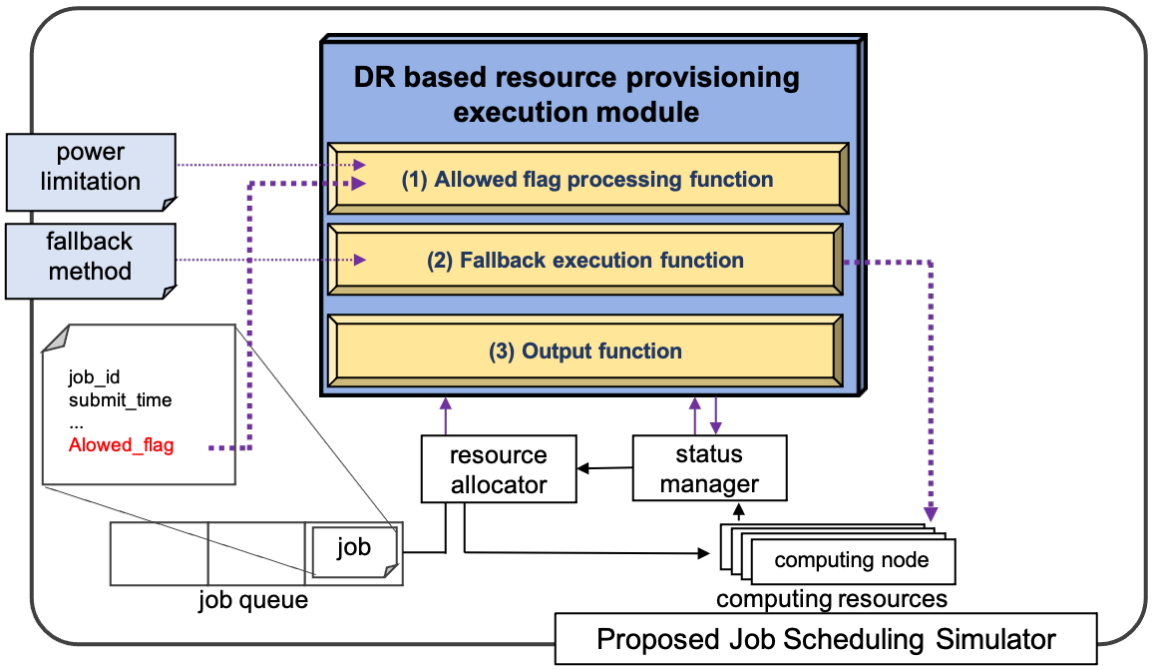

Figure 3: Design of proposed job scheduling simulator.

sources based on the fallback method. The arrow 9 is a process in which the status manager outputs the job execution time, the job waiting time, and the power consumption of the large-scale cluster system.

To realize these functionalities, we have designed DR-based resource provisioning execution module. The proposed job scheduling simulator is realized by linking this module with the existing job processing. Fig. 3 shows the design of the proposed job scheduling simulator. This module consists of three functions: (1) Allowed flag processing function, (2) Fallback execution function, (3) Output function. Function (1), function (2), and function (3) enable the processes of Fig. 2-5,6,7, Fig. 2-8, and Fig. 2-9 for existing job processing, respectively. Function (1) handles the power limitation as a criterion for executing DR-based resource provisioning and the allowed flag. Function (2) enables the status manager to change the status of the computing resources based on the fallback method. Function (3) receives the status of computing resources from the status manager and refers to the job information by using job ID. Function (3) outputs the job execution time, the job waiting time, and power consumption of large-scale cluster systems.

4. Implementation of the Job Scheduling Simulator

4.1 Overview of the Job Scheduling Simulator

We implement the proposed job scheduling simulator based on the design described in Sec. 3.2. The implementation structure of the proposed job scheduling simulator is shown in Fig. 4. The proposed job scheduling simulator consists of two components: simulus [11], which is an exist-

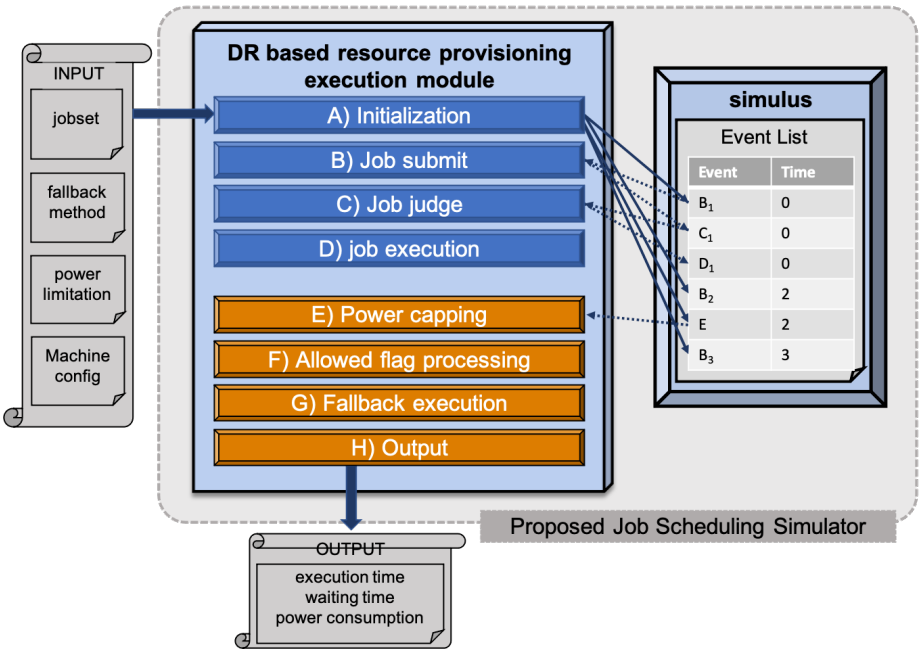

Figure 4: Implementation Structure of Proposed Job Scheduling Simulator.

ing event-driven general-purpose simulator, and DR-based resource provisioning execution module shown in Fig. 3.

Simulus is employed as the foundation of the proposed job scheduling simulator. Simulus performs simulations by executing the events described in the event list. The event list is described as pairs of events and their execution time. A function is assigned to the event as the execution content. In the actual simulation, the function assigned to the event is executed at the designated simulation time. Simulus facilitates the implementation of various event-driven simulators by developing functions defined as processes of independent functions. Thus, simulus is highly extensible. Is suitable to implement the foundation functions of the proposed job scheduler, which are expected to be extended in the future. We intend to use the proposed job scheduler in the future to evaluate a selection algorithm to choose which jobs with allowed flags are affected by the fallback, or to incorporate new fallback methods into the proposed job scheduler.

The DR-based resource provisioning execution module contains eight functions. They are divided into two groups. One group is for processing jobs with the existing job scheduling simulator. The other group is for performing the functions (1)-(3) shown in Fig. 3. The events can be described in the event list by the eight functions described in more detail below.

The first group contains Initialization (Function A in Fig. 4), Job submit (Function B), Job judge (Function C), and Job execution (Function D). Function A is a function for the initial setup of the simulation. Function A has two roles for receiving input data and for adding fixed events to the event list. A fixed event is one that has a fixed start time and is not subject to change, such as a job submission event. Function A adds Function B and Function E as fixed events to the event list. Function B adds a job to the job queue, which is the process when a job is submitted. If a job is added to the head of the job queue, Function B describes Function C to judge whether the job

is executable. Function C has the role of judging whether a job is executable or not. Function C describes Function D based on judgment. Function D is responsible for the job execution (process 3 in Fig. 2). At the start of a job, Function D removes a job from the job queue and increases the number of computing resources allocated to jobs. At the end of the job, Function D reduces the number of computing resources allocated to jobs.

The second group includes Power capping (Function E in Fig. 4), Allowed flag processing (Function F), Fallback execution (Function G), and Output (Function H). Function (1) described in Sec. 3.2 is satisfied by Function E and Function F, and Function (2) is satisfied by Function G. Function (3) is satisfied by Function H. Function E has a role in generating power saving situations. Based on the value of power limitation, Function E changes the upper limit of the power, that can be consumed by all computing resources, a parameter of computing resources. Function F judges whether the allowable flag is true or not. If it is true, Function F registers function G as an event. Function G is for executing the fallback. Function G changes the job execution time and the number of computing nodes. Function H outputs the power consumption in large-scale cluster systems, the job waiting time, and the job execution time.

4.2 Method of Creating Event List for DR-based Resource Provisioning

The procedure for creating an event list in the proposed job scheduling simulator is shown in Fig. 5. This flowchart contains eight actions. The actions can be divided into two groups. The first group is represented as (ii), (iii), (v), and (viii), which describes the behavior of the existing job scheduling simulators. The second group is (i), (iv), (vi), and (vii), which describes the behavior of DR-based resource provisioning. The second group is shown in "Area 1" of Fig. 5.

To create the event list, Action (i) and Action (ii) are executed first. Action (i) has the role of describing Function E illustrated in Fig. 4 when the power limit changes based on the power

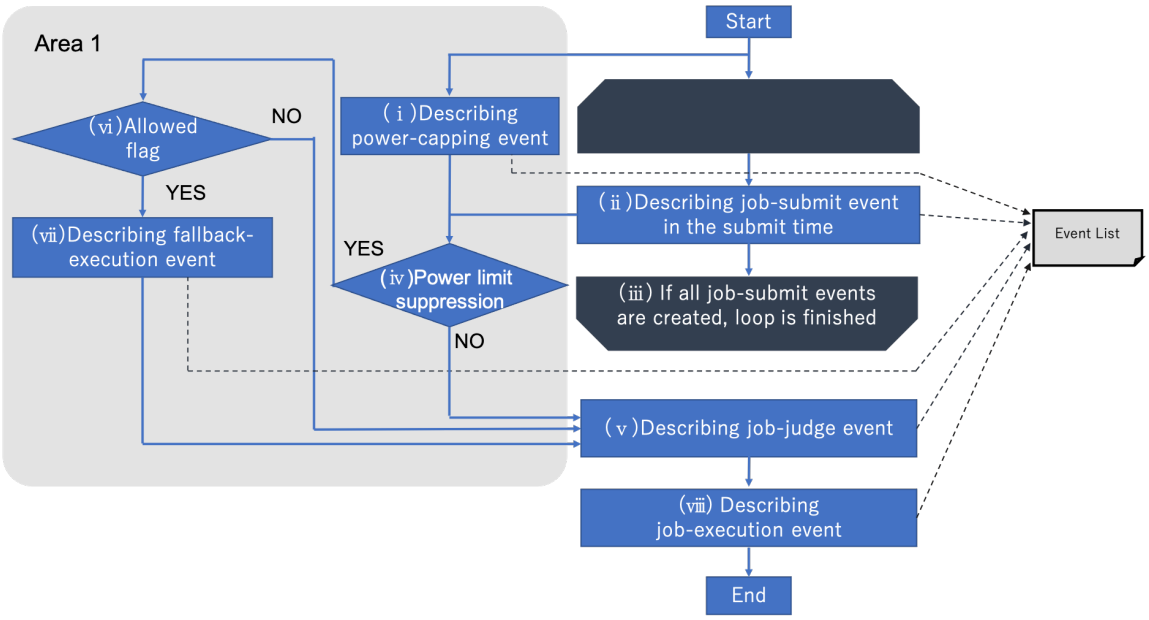

Figure 5: Flow of creating event list.

limitation. Action (ii) has the role of describing Function B as illustrated in Fig. 4 at the submission time of each job in the jobsets. Action (iii) performs Action (ii) as many times as the number of jobs. These actions are done by Function A as illustrated in Fig. 4. After completing these actions, a process of running simulus is performed by Function A.

When simulus reaches the registered event, the function registered as the event is called by simulus. Action (iv) is performed in Function B as illustrated in Fig. 4. Action (iv) has the role of judging whether the power limit suppression is occurring based on the power limitation. If the judgment result is true, Action (iv) executes the action (vi). If false, action (v) is executed. Action (vi) is executed by Function F as illustrated in Fig. 4. Action (vi) judges the allowed flag. If the result is true, Action (vi) executes the action (vii). If false, the action (v) is performed. Action (vii) describes Function G illustrated in Fig. 4 and executes Action (v). Action (v) describes Function C illustrated in Fig. 4. When Action (v) is executed by simulus, Function C is called, and Action (vii i) is executed by Function C. Action (viii) describes Function D illustrated in Fig. 4.

5. Evaluation

5.1 Experiment Setup

Two experiments were conducted for evaluating the proposed job scheduling simulator. The purpose of the first experiment is to make sure the proposed job scheduling simulator works properly. The purpose of the second experiment is to confirm that the indexes for designing an incentive system are presented. In the first experiment, we set a power limit and checked whether the proposed job scheduling simulator is able to provide resources within that limit. We set up a simplified evaluation condition for the simulation so that the behavior could be easily checked. In the second experiment, we conducted a case study to observe how much the amount of increase in the job execution time, the job waiting time, and the reduction of power consumption in large-scale cluster systems would be for a certain percentage of allowed users. Since an incentive system is designed based on the simulation results of the actual large-scale cluster systems, the simulation is set up in an evaluation condition that models the actual workload.

The two experiments were both conducted on the same PC. The processor used is Intel Core i7-4578U, which is 3 GHz. The memory in the used PC is two 8 GB 1600 MHz DDR3. Table 2 shows the common simulation conditions for the two experiments. All jobs in the jobsets given as input data for the simulation are assumed to be submitted within 24 hours. For the configuration

Submit time of jobs	0 - 24	[h]
Number of computing nodes	151	[nodes]
Number of CPU cores per a computing node	14	
Power Consumption of Idling CPU	66	[W]
Power Consumption of Running CPU	240	[W]
Maximum power limitation of large-scale cluster systems	509	[kW]
Power limitation of large-scale cluster systems for power saving	407	[kW]

Table 2: Common simulation conditions.

of the computing resources, the number of computing nodes is 151, each with 14 CPU cores. The power consumption of a CPU is configured as 66 W if in idle state and 240 W otherwise. From this condition, power consumption is about 507 kW when all computing nodes were in use. Thus, the maximum power limitation is set to 509 kW. The power limitation for power saving is set at 407 kW, which is about 80% of the maximum value.

5.2 Experiment Results

Table 3 shows the simulation conditions specific to the first experiment. The number of jobs in the jobsets was 100, the execution time of each was between 2 and 6 hours. The number of parallel computing nodes ranged from 1 to 16. Fig. 6 shows each simulator's response to power saving in the same condition (Table 4). Fig. 6-(a) shows the simulation results of the job scheduling simulator which does not support DR-based job processing. Fig. 6-(b) shows the simulation results of the proposed job scheduling simulator. The horizontal axis is the time and the vertical axis is the power consumption. The line parallel to the x-axis in Fig. 6 is the power limitation for power saving. Under power saving, the job scheduling simulator that does not support DR-based job processing cannot adapt to the power limitation for power saving. On the other hand, the proposed job scheduling simulator can provide resources within the power limitation for power saving. This confirms that DR-based resource provisioning is working at the preliminary level in the proposed job scheduling simulator.

Table 4 shows the simulation conditions specific to the second experiment. The number of jobs in the jobsets was 3000, the execution time of each job was between 100 and 3000 seconds, and the number of parallel computing nodes ranged from 1 to 4. DVFS was set as the fallback method.

Number of jobs	100	
Execution time of jobs	2 - 6	[h]
Number of parallel computing nodes	1 - 16	

Table 3: Simulation conditions specific to first experiment.

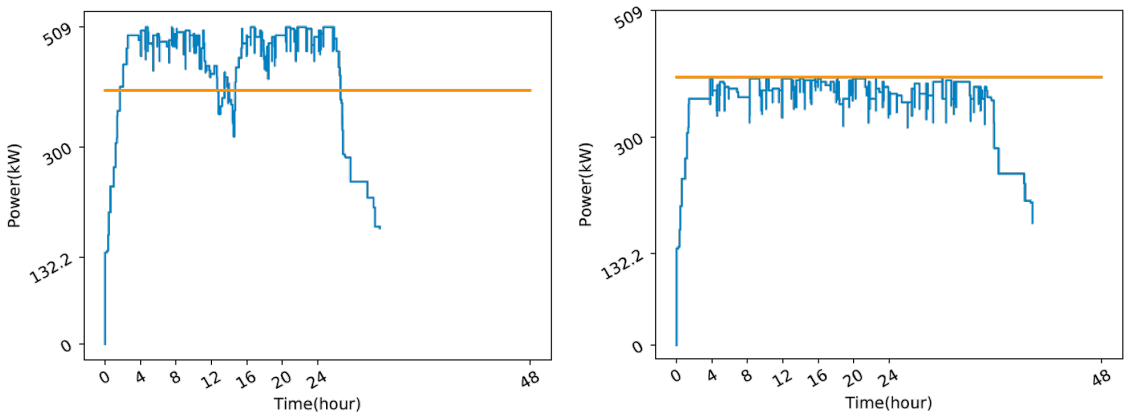

Figure 6: Each simulator's response to power saving.

Number of jobs	3000	
Execution time of jobs	100 - 3000	[s]
Number of parallel computing nodes	1 - 4	
the fallback method	DVFS	

Table 4: Simulation conditions specific to second experiment.

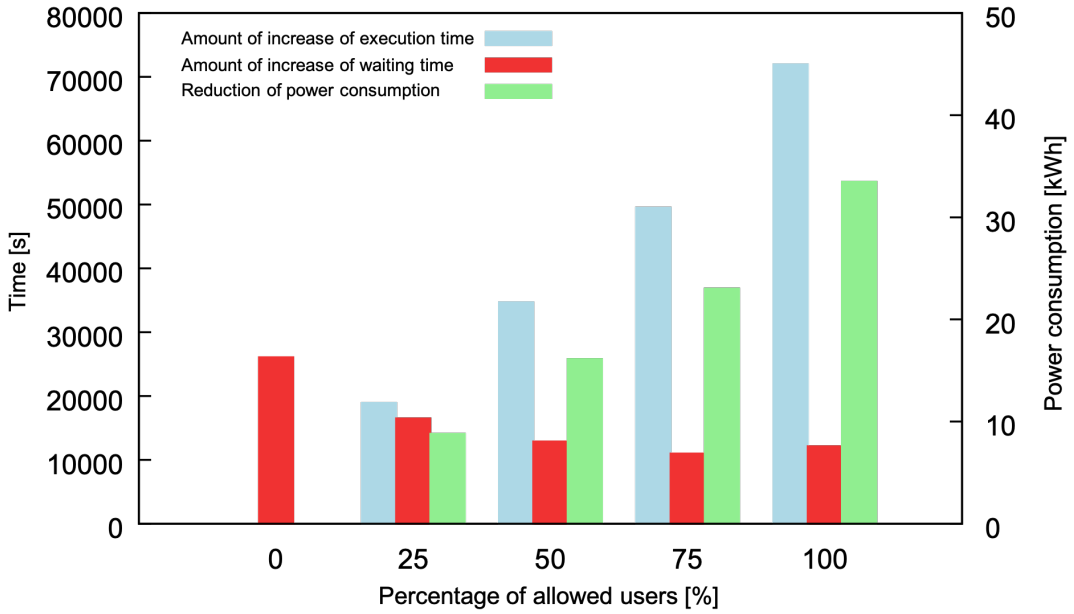

Figure 7: Results of each case study with different percentages of allowed users.

In the case study, some situations with a different portion of allowed users/jobs are conducted. The results of each case study are shown in Fig. 7. The amount of increase in job waiting time decreases relative to the percentage of allowed users. However, the amount of increase in job waiting time increases between 75% and 100%. To minimize the amount of increase in job waiting time, we can design an incentive system so there is 75% of allowed users.

6. Conclusion

To support designing an appropriate incentive system for DR-based resource provisioning, we proposed a new job scheduling simulator which presents the power consumption in large-scale cluster systems, the job waiting time, and the job execution time. To realize the proposed job scheduling simulator, we derived functionalities required for the job processing in DR-based resource provisioning against the job processing flow of the existing resource provisioning. We designed DR-based resource provisioning execution module with the derived functionalities and then implemented the proposed job scheduling simulator by linking DR-based resource provisioning execution module with simulus, a generic discrete-event simulator. In the evaluation, we confirmed that the proposed job scheduling simulator supported job processing for DR-based resource

provisioning and output data that can be used as indexes for the design of an incentive system.

For future work, we plan to enhance the processing capability of the allowed flags. In the proposed job scheduling simulator, the simple job scheduling algorithm that allocates computing resources according to the input order is adopted to simplify the implementation. Research and development of a new job scheduling algorithm that modifies the order of the job execution based on the allowed flags are required to minimize the impact of "fallback" on disallowed users.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number JP17KT0083.

References

[1] K. Ahmed, J. Liu and K. Yoshii, *Enabling Demand Response for HPC Systems through Power Capping and Node Scaling*, in *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 789–796, Jun., 2018.

[2] K. Ahmed, J. Liu and X. Wu, *An Energy Efficient Demand-Response Model for High Performance Computing Systems*, in *Proceedings of the 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 175–186, Sep., 2017.

[3] A. Kishwar and L. Jason, *Simulation of Energy-Efficient Demand Response for High Performance Computing Systems*, in *Proceedings of the 31st Winter Simulation Conference(WSC2019)*, pp. 1–12, Dec., 2019.

[4] "Top500." https://www.top500.org, Nov., 2020.

[5] W. Bao, C. Hong, S. Chunduri, S. Krishnamoorthy, L.-N. Pouchet, F. Rastello et al., *Static and Dynamic Frequency Scaling on Multicore CPUs*, *ACM Transactions on Architecture and Code Optimization (TACO)* **13** (2016) 1.

[6] J. Power, J. Hestness, M. S. Orr, M. D. Hill and D. A. Wood, *GEM5-GPU: A Heterogeneous CPU-GPU Simulator*, *IEEE Computer Architecture Letters* **14** (2015) 34.

[7] P. Czarnul, J. Kuchta, M. Matuszek, J. Proficz, P. Rościszewski, M. Wójcik et al., *MERPSYS: An Environment for Simulation of Parallel Application Execution on Large Scale HPC Systems*, *Simulation Modelling Practice and Theory* **77** (2017) 124.

[8] M. A. Obaida and J. Liu, *Simulation of HPC Job Scheduling and Large-Scale Parallel Workloads*, in *Proceedings of the 50th Winter Simulation Conference (WSC)*, pp. 920–931, Dec., 2017.

[9] R. Buyya and M. Murshed, *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, *Concurrency and Computation: Practice and Experience* **14** (2002) 1175.

[10] N. A. Simakov, M. D. Innus, M. D. Jones, R. L. DeLeon, J. P. White, S. M. Gallo et al., *A SLURM Simulator: Implementation and Parametric Analysis*, in *Proceedings of the 8th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, vol. 10724, pp. 197–217, Dec., 2017.

[11] J. Liu, *Simulus: Easy breezy simulation in python*, in *2020 Winter Simulation Conference (WSC)*, pp. 2329–2340, 2020, DOI.