2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) | 978-1-6654-6367-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/CloudCom55334.2022.00031

Consideration of a Supercomputing System with Cloud Bursting Functionality from an Operational Perspective

Arata Endo
Information Initiative Center
Nara Institute of Science and Technology
Nara, Japan
endo.arata@itc.naist.jp

Shinji Yoshida
Graduate School of Information Science and Technology
Osaka University
Osaka, Japan
yoshida.shinji@ais.cmc.osaka-u.ac.jp

Shuichi Gojuki
Microsoft Japan Co., Ltd.
Tokyo, Japan
shuichi.gojuki@microsoft.com

Hiroaki Kataoka
NEC Corporation
Tokyo, Japan
hkataoka@nec.com

Yoshihiko Sato
RTi-cast Inc.
Sendai, Japan
y-sato@rti-cast.co.jp

Akihiro Musa
RTi-cast Inc.
Sendai, Japan
a-musa@rti-cast.co.jp

Susumu Date
Cybermedia Center
Osaka University
Osaka, Japan
date@cmc.osaka-u.ac.jp

***Abstract*—Cloud bursting functionality provides the administrators of on-premise supercomputing systems with an on-demand way of dynamically integrating the computing resources on the cloud into them when the computing demands from the users dramatically increase. In the Cybermedia Center at Osaka University, we have deployed a cloud bursting environment between a supercomputing system named SQUID and Microsoft Azure by respecting the transparency in usage rather than performance. Technically, the transparency allows users to execute their jobs on the cloud computing resources without having to be aware of the cloud in terms of usage. However, how we encourage users of our supercomputing system to use the cloud computing resources is a realistic and important problem to be solved from a practical operational perspective. In this paper, we investigate the cloud bursting environment on SQUID in terms of performance, monetary cost, and transparency on usage. Finally, we discuss how we as the administrators of on-premise supercomputing systems can provide a better way to encourage users to execute their jobs on the cloud computing resource instead of on the on-premise computing resource under our service policy.**

***Index Terms*—Supercomputing System, Cloud Bursting, IaaS Cloud Service**

I. Introduction

On-premise supercomputing systems are sometimes faced with a situation where the computing demand from users exceeds the computing resources in amount. The surge of computing demand may cause a longer wait time from when a user submits a job until when the job starts. On the other hand, IaaS cloud services such as Microsoft Azure and Oracle Cloud Infrastructure are expected to provide enough computing resources to satisfy the computing demand from users in an elastic manner. From this situation on on-premise supercomputing systems and the expectation for the cloud, the idea or concept of offloading the workloads on on-premise computing resources to the cloud computing resources in an on-demand fashion has emerged. This idea or concept is referred to as *cloud bursting* [1]–[3].

In our previous work summarized in [4], we have built a cloud bursting environment between our supercomputing system named OCTOPUS [5] and the IaaS cloud service by Microsoft Azure, in the hope that our environment allows users to execute their jobs on the cloud computing resources without having to be aware of the cloud in terms of usage. More technically, in the cloud bursting environment, the computing resources of OCTOPUS and those of the Azure are connected on a VPN. Also, the job manager deployed onto OCTOPUS is extended to use the cloud computing resources when peak computing demands occur. The evaluation in [4] shows that the cloud bursting environment provides the users with a transparent way of executing their jobs on the cloud in the same way as the on-premise supercomputing system, according to the administrator's service policy.

This cloud bursting environment is operated practically; however, we still have to technically or operationally tackle issues caused by the difference in performance and cost between on-premise and cloud computing resources. For the transparency described above, we designed the cloud bursting environment so that jobs executed on the IaaS cloud service use input and output data placed on the on-premise supercomputing system. This design may decrease computing performance and consequently may slow job executions on the IaaS cloud service. Also, the service charge of the cloud computing resources is generally more expensive than that of the on-premise computing resources. Therefore, to encourage users to use the cloud computing resources instead of on-premise computing resources, we must explore a better way to build and operate the cloud bursting environment so that both the administrators and the users can benefit.

In this paper, we discuss how the administrator should

978-1-6654-6367-6/22/$31.00 ©2022 IEEE
DOI 10.1109/CloudCom55334.2022.00031

build and operate a cloud bursting environment through a case study of our new supercomputing system SQUID [6], the supercomputing system installed at Osaka University after OCTOPUS. We refer to the cloud bursting environment established between SQUID and an IaaS cloud service as the SQUID-Azure cloud bursting environment. Specifically, to encourage users to use cloud computing resources, we consider how the cloud bursting environment benefits administrators and users in terms of performance, cost and usage. For this discussion, we preliminarily investigate how the design of the data placement increases the execution time of jobs performed on the IaaS cloud service and how costly the cloud computing resources are. For these investigations, we observe and evaluate the performance and cost of several benchmark software and applications. Next, we investigate the SQUID-Azure cloud bursting environment from a practical operational viewpoint. We hope that the experience and expertise summarized in this paper are useful for those who are interested in cloud bursting functionality from an actual operational point of view.

The rest of this paper is organized as follows: Section II reviews related work. Section III describes the overview of the SQUID-Azure cloud bursting environment. Section IV preliminarily investigates the performance and cost of the SQUID-Azure cloud bursting environment. In Section V, we explore a better way to provide the cloud bursting environment. Finally, in Section VI, we conclude this paper.

II. Related work

High-performance computing cloud (HPC cloud) has been studied by many researchers as indicated in a survey study [7]. Of course, the characteristics of cloud computing resources have been investigated for the HPC cloud [8], [9]. However, these research studies do not assume jobs running on an IaaS cloud service access to data on an on-premise supercomputing system because the HPC cloud recognized performance on the cloud rather than transparency on usage. Therefore, for our discussion, we investigate the performance and cost characteristics of the cloud bursting environment which recognizes transparency in usage.

In a cloud bursting environment, an offloading mechanism for deciding which on-premise computing resources or cloud computing resources a job scheduler uses to perform jobs is important. In [10], such an offloading mechanism has been proposed. Using deep reinforcement learning, this offloading mechanism allows administrators to adjust the trade-off between the wait time of jobs and the cost to perform the jobs. However, this research study does not assume that the execution time of a job may be increased due to the performance characteristic of the cloud bursting environment when jobs are offloaded to cloud computing resources. In this paper, we also verify whether an offloading mechanism targeting a cloud bursting environment must consider the performance characteristic in the future or not.

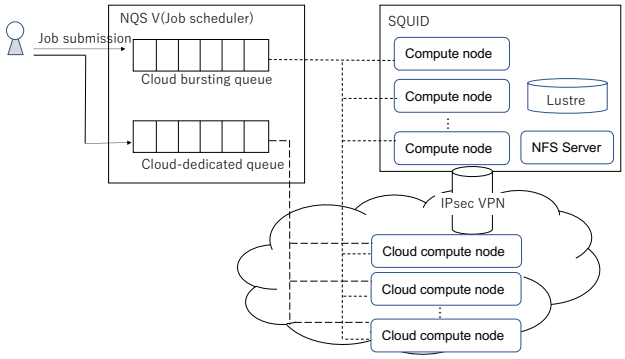


Fig. 1. Overview of the SQUID-Azure cloud bursting environment.

III. SQUID-Azure cloud bursting environment

A. Overview of the SQUID-Azure cloud bursting environment

Fig. 1 illustrates a brief overview of the SQUID-Azure cloud bursting environment. The architecture of this environment is the same as the cloud bursting environment summarized in [4], except for the internal mechanism of the job scheduler. In this cloud bursting environment, the cloud computing resources on the Azure and the computing resources composing SQUID are used for cloud compute nodes and on-premise compute nodes, respectively, and these are connected on a VPN established over the Internet. Specifically, the cloud bursting environment uses the cloud computing resources provided by a data center located in east Japan, approximately $400\,\mathrm{km}$ far from SQUID. The file system on the supercomputing system (Lustre) is mounted from the cloud compute node to enable the submitted jobs to be executed on either the on-premise supercomputing system or the IaaS cloud service. Furthermore, NQSV [11], a job scheduler, deployed onto the on-premise supercomputing system was designed so that it treats the cloud computing resources on the IaaS cloud service in the same way as the on-premise computing resources. Technically, a *cloud bursting queue* is deployed and configured so that the jobs submitted to it are assigned to either cloud compute nodes on the IaaS cloud service or on-premise compute nodes on the on-premise supercomputing system. Furthermore, for this research, we set up a *cloud-dedicated queue* whose submitted jobs are assigned only to the cloud compute node.

In this cloud bursting environment, jobs running on either on-premise or cloud compute nodes generally read input data and write output data on Lustre. Therefore, file I/O overhead to Lustre occurs if jobs are run on cloud compute nodes. For the avoidance of the file I/O overhead, we allow users to use the *local storage* of the on-premise and cloud compute nodes as scratch space. As long as jobs read and write data on the local storage, jobs do not need to access data on Lustre. To use the local storage, users must use the *staging operation*, which moves the input and output data between Lustre and the local storage before and after job execution.

B. Cloud compute nodes on the Azure

Table I shows the specification of the on-premise compute node on SQUID and the cloud compute nodes on Azure. Originally, we wanted to use the cloud compute nodes that provide more processor cores and memory than an on-premise compute node so that the user jobs to be executed on on-premise compute nodes can be accommodated on cloud compute nodes, considering that users are allowed to transparently use the cloud compute node without having to be aware of the difference between the on-premise and cloud compute nodes. However, we could not help selecting such cloud compute nodes due to the unavailability of the cloud compute nodes in building the cloud bursting functionality. Therefore, we chose these cloud compute nodes that provide processor core and memory as close as possible to an on-premise compute node.

C. Service policy

The service policy in our center is simple and we charge a service fee corresponding to the consumed power for the use of the supercomputing systems. More precisely, the service charge is calculated based on PUE (Power Usage Effectiveness) to contain the consumed power for the cooling facility as well. For this reason, the current service charge per node-hour usage in our center is much lower than that in the IaaS cloud service, because it does not include every expense necessary for providing our services, such as the construction cost of data center building and facilities, and an engineering labor fee. Therefore, this gap becomes a problem from both the administrator's and the user's perspectives.

If we as the administrator can pay the gap between them on behalf of the users when the user jobs submitted to the cloud bursting queue are executed on the cloud instead of the on-premise supercomputing system, many users would accept the offloading of their jobs onto the cloud when the on-premise supercomputing system is heavily loaded. In fact, we plan to pay the gap on behalf of the users, when we enable the cloud bursting queue and as a result, the user jobs are executed on the cloud. For the users, this benefits the reduction of wait time until the user jobs start without an additional service charge. However, this operation can be possible only for a short term when surgical demand happens. This reason is we plan to use the money possibly saved through the energy-saving operation of the supercomputing system.

However another problem arises; namely, in the case of operating a cloud-dedicated queue, how to encourage the users to use the cloud resources instead of the heavily-loaded on-premise supercomputing resources. The current operational assumption behind the cloud-dedicated queue is as follows: When the on-premise supercomputing system is heavily loaded, the administrators enable the cloud-dedicated queue and expect the users to use the cloud resources with a higher cost instead of the on-premise resources. This situation works to some degree because some portions of users want to use computing resources for their computation even with higher cost if it is quickly finished. Of course, if the administrator can pay the gap, more users would be encouraged to use the cloud. However, this situation is impossible under our service policy.

IV. Preliminary investigation

A. Investigation overview

In this preliminary investigation, we evaluate the performance and cost characteristics of the SQUID-Azure cloud bursting environment. For this evaluation, we perform benchmark software and an application described later as a job on on-premise and cloud compute nodes. Through this job execution, we measure the execution time of the job as the criteria of performance and then estimate the cost of the job execution.

The cost of a job execution using N nodes is given as:

$$Cost = \frac{T_j + T_s}{3600} CN$$

where T_j (s) is the job execution time, T_s (s) is stage-in/out time, and C ($/node-hour) is the service charge per node-hour usage. When the job is executed without local storage, the stage-in/out time is 0. The service charge per node-hour usage in our center and Azure is shown in Table II. Note that the approximate exchange rate dollar to yen on August 8th, 2022 was referenced.

B. Benchmark software and application

In this investigation, we observe the performance and cost characteristics of the cloud bursting environment. For this reason, we used different types of benchmarks listed in Table III to reproduce three types of jobs: CPU-intensive, communication-intensive and I/O-intensive jobs.

1) EP: We adopted Embarrassingly Parallel (EP), a benchmark in NAS parallel benchmark [14], as a CPU-intensive job. EP is a Monte Carlo simulation application generating a few communications between compute nodes so it can evaluate the scalability of CPU performance. EP requires 2^n cores to execute itself (n is a natural number).

2) CG: We adopted Conjugate Gradient (CG), a benchmark in the NAS parallel benchmark, as a communication-intensive job. CG solves an unstructured sparse linear system using a conjugate gradient method and is suitable for evaluating the scalability of communication performance because it generates a lot of irregular point-to-point communications between compute nodes. CG requires 2^n cores to execute itself.

3) BT-IO: Block Tridiagonal Solver Input/Output (BT-IO), a benchmark in NAS parallel benchmark, was adopted as an I/O-intensive job. BT-IO solves a Computational Fluid Dynamics (CFD) application using multiple and independent systems of non-diagonally dominant and block-tridiagonal equations with a (5×5) block size and can measure file I/O time [15]. BT-IO requires n^2 cores to execute itself.

4) The tsunami simulation: We adopted the tsunami simulation [16], an MPI application to estimate the damage caused by a tsunami, as a CPU-intensive job. The tsunami simulation is suitable as a CPU-intensive job due to the small I/O data size. We executed the tsunami simulation using all cores in a compute node.

TABLE I
SPECIFICATION OF THE ON-PREMISE COMPUTE NODE AND THE CLOUD COMPUTE NODE.

Node	Processor	# of cores on a node	Base frequency	Memory	SSD	Interconnect
SQUID	Intel Xeon Platinum 8368 (Ice Lake) × 2	76	2.40 GHz	256 GB	85 GB	InfiniBand HDR (200 Gbps)
D64ds_v4	Intel Xeon Platinum 8272CL (Cascade Lake)	32	2.60 GHz (estimated)	256 GB	2400 GB	Ethernet (30 Gbps)
HC44rs	Intel Xeon Platinum 8168 (Sky Lake)	44	2.70 GHz	352 GB	700 GB	InfiniBand EDR (100 Gbps)

TABLE II
SERVICE CHARGE PER A NODE-HOUR USAGE IN D64DS_V4, HC44RS AND SQUID (1 USD = 135 JPY) [12], [13].

The compute node	The cost per node-hour
D64ds_v4	$4.672
HC44rs	$4.595
SQUID	$0.111

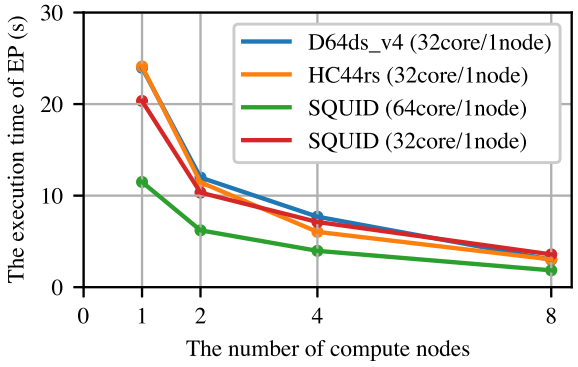
TABLE III
LIST OF BENCHMARKS USED FOR THIS INVESTIGATION.

Benchmarks	Classification of jobs
EP	CPU-intensive job
CG	Communication-intensive job
BT-IO	I/O-intensive job
The tsunami simulation	CPU-intensive job

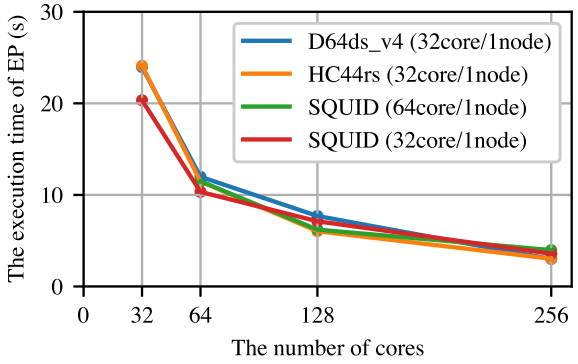
C. Performance measurement result

Fig. 2 shows the execution time of EP (class D) without local storage by using the number of compute nodes (Fig. 2 (a)) and cores (Fig. 2 (b)) as shown in Table IV. The reason why we show the execution time of EP executed on SQUID using 32 cores per compute node is for comparing CPU performance among D64ds_v4, HC44rs and SQUID using the same number of cores. This result shows the execution time of EP on D64ds_v4 and HC44rs is almost the same as that on SQUID when D64ds_v4, HC44rs and SQUID use the same number of cores. Also, the increase in the number of compute nodes and cores makes the execution time of EP short. Therefore, the scalability of the CPU performance in D64ds_v4 and HC44rs exhibits a similar trend to that in SQUID despite the difference in processor generation. This result implies that the performance overhead due to the file I/O from the cloud compute node does not affect the total performance of CPU-intensive jobs.

Fig. 3 shows the execution time of CG (class D) without local storage by using the number of compute nodes and cores as shown in Table IV in the same way as EP. This result shows that the execution time of CG on HC44rs and SQUID becomes short with the increase in the number of compute nodes and cores, while that of CG on D64ds_v4 becomes long in the case of eight compute nodes despite the increase in the number of nodes. This is because the bandwidth between compute nodes in D64ds_v4 is lower than that in HC44rs and SQUID. This result implies that D64ds_v4 is not suitable for the jobs that necessitate communications between multiple compute nodes, for example, communication-intensive jobs.



(a) Execution time depicted per the number of compute nodes.



(b) Execution time depicted per the number of cores.

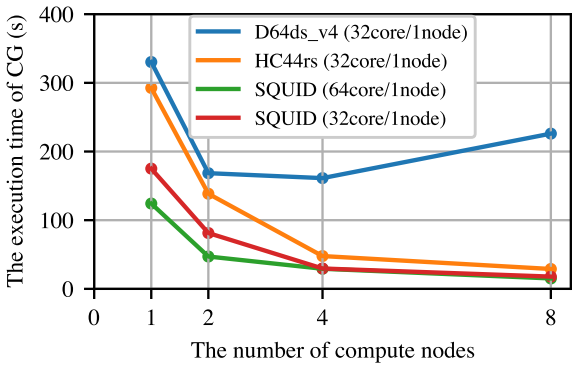
Fig. 2. Execution time of EP to evaluate the scalability of CPU performance.

BT-IO (class C, input data size 0 GB, output data size 6.4 GB) was measured on 25 cores in D64ds_v4, 36 cores in HC44rs and 64 cores in SQUID for using as many cores as possible in a compute node. Also, we used two methods which measure BT-IO without local storage (the direct I/O method) and with local storage (the staging method). Fig. 4 (a) shows the time for file I/O operations during a job execution (I/O time) with the two methods and Fig. 4 (b) shows the I/O time and the time for a file transfer to the on-premise storage after a job execution (stage-out time) with the staging method.

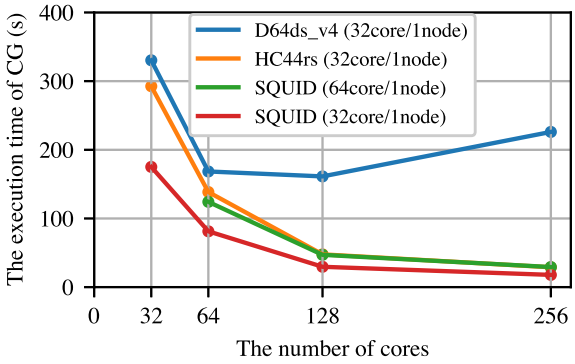
The I/O time of D64ds_v4 and HC44rs with the staging

TABLE IV
NUMBER OF COMPUTE NODES AND CORES TO EXECUTE EP AND CG.

# of compute nodes	D64ds_v4	HC44rs	SQUID (64 cores / node) # of cores	SQUID (32 cores / node)
1	32	32	64	32
2	64	64	128	64
4	128	128	256	128
8	256	256	512	256

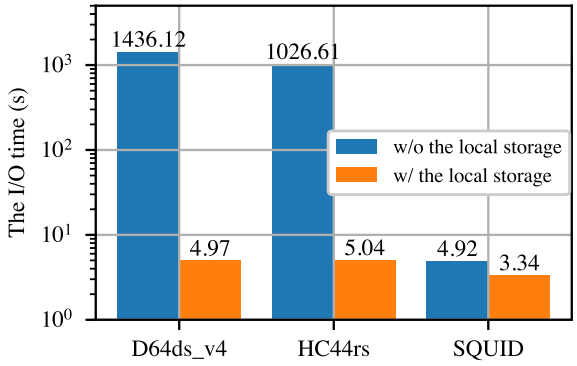


(a) Execution time depicted per the number of compute nodes.

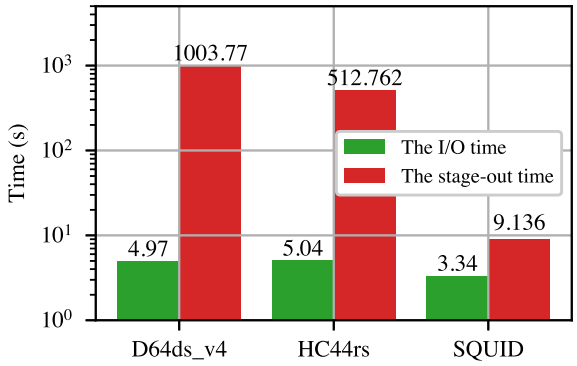


(b) Execution time depicted per the number of cores.

Fig. 3. Execution time of CG to evaluate the scalability of communication performance.



(a) I/O time with and without local storage.



(b) I/O time and stage-out time with local storage.

Fig. 4. I/O time and stage-out time of BT-IO to evaluate I/O performance.

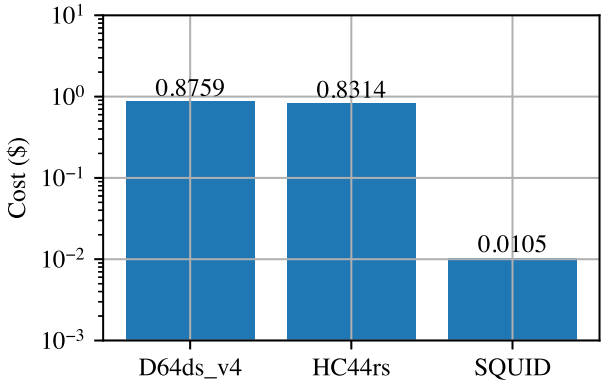
method show a similar tendency as the case of SQUID with the direct I/O method. Furthermore, in the case of D64ds_v4 and HC44rs, the total of the I/O time and the stage-out time with the staging method is shorter than the I/O time with the direct I/O method. These results apparently show that the use of local storage improves the I/O performance of D64ds_v4 and HC44rs, so that the staging operation makes it suitable to execute I/O intensive jobs on the cloud computing resources. In particular, because in the case of the staging method, the stage-in/out time of D64ds_v4 and HC44rs is much longer than their I/O time, the staging operation makes them more suitable for I/O-intensive jobs which generate big file I/O data and small stage-in/out data.

D. Cost estimation result

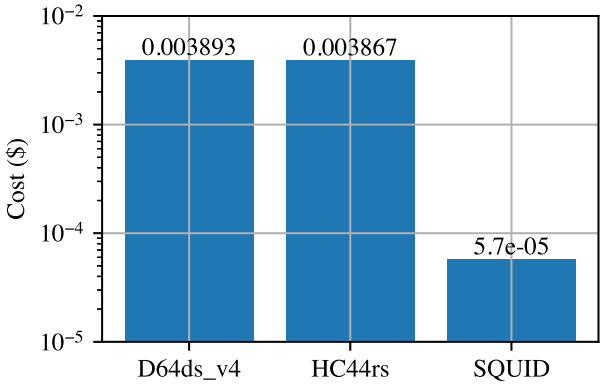
Fig. 5 (a) shows the cost of the tsunami simulation based on the execution time of the tsunami simulation without local storage. This result shows that the cost of the tsunami simulation on D64ds_v4 and HC44rs is 79 times or more than SQUID.

Fig. 5 (b) shows the cost of EP calculated from the performance measurement result when using eight compute nodes in Fig. 2. This result shows that the cost of EP in D64ds_v4 and HC44rs is 68 times or more than SQUID.

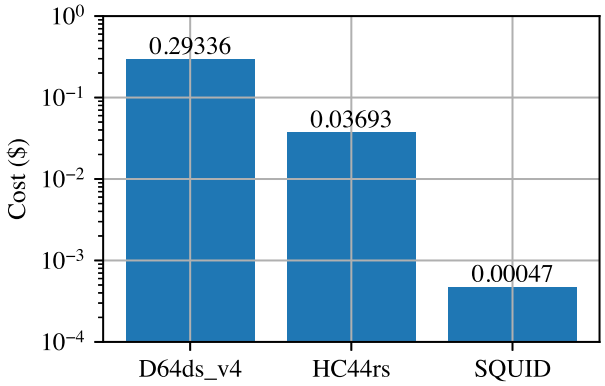
Next, Fig. 5 (c) shows the cost of CG calculated from the performance measurement result in Fig. 3. Fig. 6 summarizes



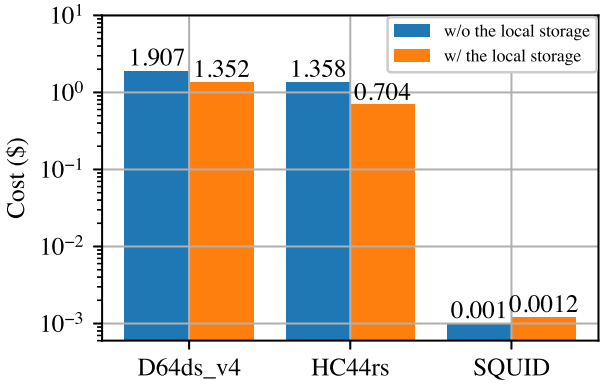
(a) Tsunami simulation (CPU-intensive job).



(b) EP using eight compute nodes (CPU-intensive job).



(c) CG using eight compute nodes (Communication-intensive job).



(d) BT-IO with and without local storage (I/O-intensive job).

Fig. 5. Cost of the CPU-intensive, communication-intensive, and I/O-intensive jobs.

the cost of CG when D64ds_v4, HC44rs and SQUID use eight compute nodes in Fig. 5 (c). These results show that the increase in the number of the compute nodes of HC44rs reduces the cost of CG. However, the cost of CG using eight compute nodes in HC44rs is approximately 78 times that in SQUID. On the other hand, the increase in the number of the compute nodes of D64ds_v4 makes the cost of CG high. As a result, the cost of CG using eight compute nodes of D64ds_v4 is approximately 624 times as long as that of SQUID.

Finally, Fig. 5 (d) shows the cost of BT-IO with and without local storage calculated from the performance measurement result in Fig. 4. This result shows the cost of BT-IO without local storage in D64ds_v4 and HC44rs is 1357 times or more than SQUID. Also, the cost of BT-IO in D64ds_v4 and HC44rs with local storage is lower than that without local storage. Note that although we did not estimate the cost for the traffic from the cloud to the on-premise in this measurement, the cost will be more than this estimation.

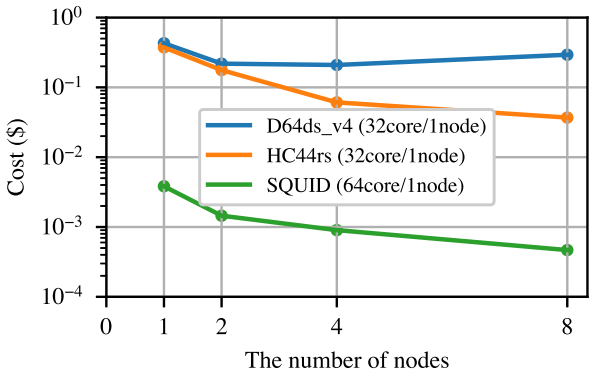


Fig. 6. Cost of CG depicted per the number of nodes.

V. Discussion

In this section, based on the investigation result, we discuss how to build and operate a cloud bursting environment that encourages users to use the cloud computing resources. Specifically, as an insight, we consider how the four possible cloud bursting environments with four settings regarding the queues and the use of local storage benefit administrators and users in terms of performance, cost, and usage. Subsequently, we

explore future work to improve the cloud bursting environment from a practical operational point of view in the Cybermedia Center at Osaka University.

A. Insights

1) Cloud bursting queue with local storage disabled: From the investigation result, the group classified into the I/O-intensive job is not suitable to be executed on the Azure in the current cloud bursting environment in terms of the transparency in performance. This is easily explained by the fact that the file I/O traffic traverses the VPN connection established over the Internet. The result that the execution time of I/O-intensive jobs is increased endorses this reason. On the other hand, this cloud bursting environment is relatively good and acceptable by users for both job classes of CPU-intensive and communication-intensive. This reason is because the performance profile of the Azure, except for D64ds_v4, shows a similar trend in scalability to our on-premise supercomputing system although I/O performance degradation is observed.

Next, we consider how to operate this cloud bursting environment. As described, our service charge for the on-premise supercomputing system is based on the consumed power for computation and we currently plan to pay the gap between the service charge of SQUID and the Azure on behalf of users who submit jobs to the cloud bursting queue. Under this policy, this cloud bursting environment is acceptable to the users. Furthermore, the transparency in usage that allows the users to execute their jobs without being aware of the difference between the cloud and on-premise supercomputing system is highly helpful for users without detailed knowledge of the cloud bursting environment. Also, from the administrator's perspective, this cloud bursting environment can be operated as long as our budget for enabling it lasts.

Nonetheless, however, this cloud bursting environment is difficult to practically operate in the long term due to our service policy. More specifically, the budget to use cloud resources is limited to the money saved through the energy-saving function. If the administrator wants to provide this cloud bursting environment for a longer term, they must ask the users to pay the gap when jobs are offloaded. Otherwise, the operation cost of our supercomputing system is run short. For this service operation, a mechanism to ask the users to permit the use of the cloud is essential. For this purpose, we have extended NQSV so that users can specify their intentions such as "cloud=yes" in their job scripts. Even in this case, for practical operations, we need to further extend a peripheral system that meters the use of the cloud computing resources.

2) Cloud bursting queue with local storage enabled: From the evaluation result, this cloud bursting environment is expected to achieve the transparency in performance for the class of I/O-intensive jobs as well as CPU-intensive and communication-intensive classes. On the other hand, this cloud bursting environment loses the transparency in usage because it requires the users to be aware of the local storage of virtual compute nodes on the cloud or compute nodes on the on-premise supercomputing system to avoid file I/O overhead. Also, any difference between the on-premise and cloud in the storage area where data is staged cannot be accepted because the jobs assuming the staging operation for the local storage on the on-premise cannot be executed on the cloud. Therefore, this cloud bursting environment requires administrators to set up the same staging configuration on both on-premise and cloud. Moreover, this environment must encourage users to use the local storage although users can submit jobs without the local storage by accepting low transparency in performance for I/O-intensive jobs.

Also, in this case, the discussion on how we should operate this cloud bursting environment is the same as the first setting. The users would be encouraged to use this environment if we pay the gap. In the case that we must ask the users to pay the gap, more users are encouraged to use the cloud than the first setting due to the improved transparency in performance for I/O-intensive jobs.

3) Cloud-dedicated queue with local storage disabled: In the cloud bursting environment with this setting, the user has to explicitly specify the cloud-dedicated queue in submitting a job by being aware of the difference between the cloud and on-premise environments. As the evaluation result showed, on the other hand, the transparency in performance is expected to be achieved for CPU-intensive and communication-intensive job classes although it is not achieved for the I/O-intensive job class. In other words, this cloud bursting environment with this setting can be technically operated in the case that many users submit CPU-intensive and communication-intensive jobs to the cloud-dedicated queue. However, this cloud bursting environment with this setting would not be used in the case that many users have no detailed knowledge of it.

Furthermore, from the administrator's perspective, how to charge is a big concern. Under this setting, our service policy that the administrators pay the gap between the cloud and on-premise service charge is practical only for the short term. In the case of the cloud-dedicated queue to which users can freely submit their jobs while this queue is enabled, it would be natural and reasonable for the administrator that users need to pay the gap when submitting jobs to this queue. Therefore, this cloud bursting environment with this setting is useful only for users who can tolerate the high cost of avoiding a wait time until a job execution starts as shown in the evaluation result. More specifically, when a user submits an I/O-intensive job to the cloud-dedicated queue, the user must pay an additional high cost because the execution time of the job is increased.

4) Cloud-dedicated queue with local storage enabled: The cloud bursting environment with the cloud-dedicated queue and local storage is expanded from the previous one so that the local storage can be used. Therefore, compared to the previous one, this cloud bursting environment additionally achieves transparency in performance for an I/O-intensive job because the use of the local storage improves I/O performance. Instead, in this cloud bursting environment, users must learn about the staging operation for the avoidance of file I/O overhead. As a result, usage of this cloud bursting environment becomes more complicated than the previous setting. However, for users who

are strongly willing to use cloud computing resources even with a high cost, the effort to learn the staging operation and to learn the performance and cost characteristics is negligible. As a result, the cloud bursting environment with this setting is the best solution for such users, although it lacks transparency in usage.

B. Future work

From the above insights, a good solution to encourage users to use cloud computing resources would be that the cloud bursting queue with local storage disabled and the cloud-dedicated queue with local storage enabled are used in combination. In this solution, the former setting contributes to transparency in cost and usage within a limited budget and the latter setting contributes to the long-term operation of the cloud bursting environment. While the former setting accommodates many users who have no detailed knowledge of the cloud bursting environment and the staging operation, the latter setting satisfies the demands of reducing wait time for job executions even if knowledge of such executions and higher cost are required.

Also, in the cloud bursting environment with the cloud bursting queue, it is helpful from a practical operational perspective to develop an offloading mechanism to efficiently allocate user jobs to cloud computing resources. In detail, in terms of transparency in performance, the offloading mechanism would be required to decide which user job is a CPU-intensive job, a communication-intensive job, or an I/O-intensive job and to decide whether the job should be allocated to cloud computing resources on behalf of users. Therefore, we consider that the offloading mechanism proposed in [10] should be enhanced so that I/O-intensive jobs are not allocated to cloud computing resources.

VI. Conclusion

Cloud bursting functionality provides the administrators of on-premise supercomputing systems with an on-demand way of dynamically integrating the computing resources on the cloud when computing demands from users dramatically increase. In this paper, the cloud bursting environment between SQUID and Azure built based on the idea described above was investigated as a case study from a practical operational perspective. More specifically, how the administrators encourage users to use the cloud computing resources instead of the on-premise ones was investigated in terms of performance, cost, and usage through the case of the SQUID-Azure cloud bursting environment. Based on this investigation, we can gain insight how the cloud bursting queue without local storage will encourage users who have no knowledge of the cloud bursting environment and the staging operation to use the cloud computing resource due to its transparency in cost and usage. Also, the cloud-dedicated queue with local storage is best for users who prioritize wait time reduction over the transparency because the cloud bursting queue can be practically operated only for a short term. From these insights, it would be a good solution for encouraging users to use the cloud computing resources that both the cloud bursting queue without local storage and the cloud-dedicated queue with local storage are operated under our service policy.

In most cases, cloud computing resources are costly in comparison with on-premise computing resources. Therefore, in computer centers where on-premise computing resources are provided as a paid service, the difference in cost becomes a problem when the cloud bursting service that provides the cloud computing resources in an on-demand way is used. We hope that this work is helpful as a case study for those who consider using the cloud bursting environment.

Acknowledgment

This work was supported by Social Smart Dental Hospital, a collaborative project between Osaka University and NEC Corp. Also, this work was supported by JSPS KAKENHI Grant Numbers JP16H02802, JP16K12419 and JP26330145.

References

[1] T. Guo, U. Sharma, T. Wood, S. Sahu, and P. Shenoy, "Seagull: Intelligent cloud bursting for enterprise applications," in *Proceedings of the 2012 USENIX Annual Technical Conference*, Jun. 2012, pp. 361–366.

[2] H. Wu, S. Ren, G. Garzoglio, S. Timm, G. Bernabeu, H. W. Kimy, K. Chadwick, H. Jang, and S. Noh, "Automatic cloud bursting under FermiCloud," in *Proceedings of the 2013 International Conference on Parallel and Distributed Systems*, Dec. 2013, pp. 681–686.

[3] F. J. Clemente-Castello, B. Nicolae, M. M. Rafique, R. Mayo, and J. C. Fernandez, "Evaluation of data locality strategies for hybrid cloud bursting of iterative MapReduce," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2017, pp. 181–185.

[4] S. Date, H. Kataoka, S. Gojuki, Y. Katsuura, Y. Teramae, and S. Kigoshi, "First experience and practice of cloud bursting extension to OCTOPUS," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science*, May 2020, pp. 448–455.

[5] "OCTOPUS." [Online]. Available: http://www.hpc.cmc.osaka-u.ac.jp/en/octopus/

[6] "SQUID." [Online]. Available: http://www.hpc.cmc.osaka-u.ac.jp/squid/

[7] M. A. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. Cunha, and R. Buyya, "Hpc cloud for scientific and business applications: taxonomy, vision, and research challenges," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–29, Jan. 2018.

[8] M. Ferretti and L. Santangelo, "Cloud vs on-premise hpc: A model for comprehensive cost assessment," in *Parallel Computing: Technology Trends*, vol. 36, Apr. 2020, pp. 69–80.

[9] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," in *Proceedings of the 2011 Sixth Open Cirrus Summit*, Oct. 2011, pp. 22–26.

[10] S. Yasuda, C. Lee, and S. Date, "An adaptive cloud bursting job scheduler based on deep reinforcement learning," in *Proceedings of the 2021 International Conference on High Performance Big Data and Intelligent Systems*, Dec. 2021, pp. 217–224.

[11] "NEC network queuing system V (NQSV) user's guide [introduction]." [Online]. Available: https://www.hpc.nec/documents/nqsv/pdfs/g2ad01e-NQSVUG-Introduction.pdf

[12] "Cost of linux virtual machines." [Online]. Available: https://azure.microsoft.com/ja-jp/pricing/details/virtual-machines/linux/#pricing

[13] "What 'point' is in shared use." [Online]. Available: http://www.hpc.cmc.osaka-u.ac.jp/system/manual/point/

[14] D. Bailey, T. Harris, W. Saphir, R. Van Der Wijngaart, A. Woo, and M. Yarrow, "The NAS parallel benchmarks 2.0," NASA Ames Research Center, Tech. Rep. NAS-95-020, Dec. 1995.

[15] P. Wong, R. VanderWijngaart, and B. Biegel, "NAS parallel benchmarks I/O version 2.4," NAS Technical Report Server (NTRS), Tech. Rep., Nov. 2002.

[16] A. Musa, T. Abe, T. Kishitani, T. Inoue, M. Sato, K. Komatsu, Y. Murashima, S. Koshimura, and H. Kobayashi, "Performance evaluation of tsunami inundation simulation on SX-Aurora TSUBASA," in *Proceedings of 19th International Conference on Computer Science*, Jun. 2019, pp. 363–376.