Enhancing System Utilization by Dynamic Reallocation of Computing Nodes

Seungmin Lee, Hee Jin Jang, and Min Ah Kim

Abstract The field has expanded as supercomputers deal with different characteristics of workloads such as traditional scientific computing and data intensive computing. A batch queue-based Parallel Batch System (PBS) scheduler that manages high performance computing (HPC) tasks and a Kubernetes platform for managing data intensive applications are applied. Computing nodes are currently divided into static partitions for each workload. However, it provides better overall resource utilization of supercomputer as dynamically reallocating computing nodes to partitions according to the number of waiting jobs. In this work, we propose an approach to dynamic resource reallocation of computing nodes. We distinguish our approach from previous works in that our approach provides isolation of software stack and reallocates resources with a bare-metal environment suitable for preventing conflicts between two heterogeneous platforms. We considered node level reallocation which means that allocation is done to the partition in computing node level rather than sharing components of computing resources. A test scenario is used to demonstrate the process and feasibility of this approach and the result shows that it can improve system utilization.

Keywords Dynamic reallocation · Heterogeneous platform · Supercomputer

S. Lee (✉) · H. J. Jang · M. A. Kim
Korea Institute of Science and Technology Information, Daejeon 34141, Republic of Korea
e-mail: smlee76@kisti.re.kr

H. J. Jang
e-mail: jhj@kisti.re.kr

M. A. Kim
e-mail: petimina@kisti.re.kr

J. S. Park et al. (eds.), *Advances in Computer Science and Ubiquitous Computing*,
Lecture Notes in Electrical Engineering 1028,
https://doi.org/10.1007/978-981-99-1252-0_24

1 Introduction

The field has expanded as supercomputers deal with different characteristics of workloads such as traditional scientific computing and data intensive computing. Data intensive computing uses a data parallel approach to process huge data sets with a great diversity of types so that it requires deferent data processing approaches. A batch queue-based Parallel Batch System (PBS) scheduler that manages high performance computing (HPC) tasks for traditional scientific computing and a Kubernetes platform for managing data intensive applications, especially big data analysis, are applied in one of the largest supercomputer, NURION in KISTI [1].

Computing nodes are currently divided into static partitions to process HPC workloads and big data analytics workloads. However, computing nodes become idle when jobs are not enough for one partition, as a consequence waste resources if there are jobs waiting computing resources in another partition. It provides better overall resource utilization of supercomputer, therefore, as dynamically reallocating computing nodes to partitions according to the number of waiting jobs as shown in Fig. 1.

There is a study that applied dynamic resource partitioning to the Athena development system using Mesos [2] and reported experiences on mixed workload [3]. In our case, problems arise when frameworks using different software stacks are installed and used on the local disk of computing nodes. A machine oriented mini-server (MOM) daemon of PBS scheduler not only runs and manages jobs and monitors resource usage on each computing node but also prohibit user daemon from listening port. However, user daemon is inevitable to enforce the principle of least privilege in Jupyter notebook service that leverage big data tools [4].

In this paper, we propose an approach to dynamic resource reallocation of computing nodes. We distinguish our approach from previous solutions in that it provides isolation of software stack and reallocates resources with a bare-metal environment suitable for preventing problems mentioned above. The remainder of this

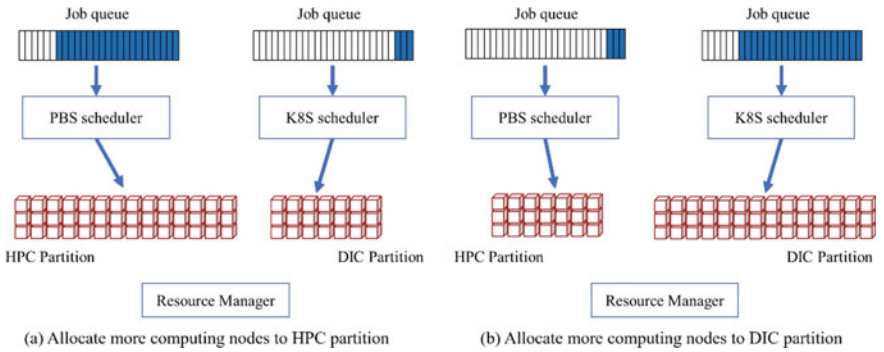

Fig. 1 Dynamically redistributing computing nodes to HPC and DIC partition according to tasks waiting in the job queue

paper is organized as follows: Sect. 2 introduces the proposed approach. Section 3 shows the experimental results and analysis. And Sect. 4 concludes this paper.

2 Methodology

In this section, we describe the key components and the reallocation process of computing nodes. Since HPC and DIC workloads have different characteristics, we first considered node level reallocation which means that allocation is done to the partition in computing node level rather than sharing components of computing resources (i.e., memory and network) so that each application can be executed according to its own characteristics without interruption. Second, the matching between tasks that request resources and available resources is determined by the number of CPUs. Last but not the least, the minimum resource of the partition managed by each scheduler is set as default (i.e., non-swappable and non-reallocatable) to prevent starvation of tasks in one partition. We briefly describe main components of dynamic resource reallocation as follows:

- **Task monitoring** to gather information about waiting tasks periodically
- **A computing node manager** to add and delete computing nodes to and from partitions managed by the scheduler
- **A database** (DB) that stores information and status about computing nodes
- **A dynamic node manager** that determines the need for reallocation through the overall system situation and by combining decision method according to information about available resources

Figure 2 the main components for dynamic resource reallocation and a sequence in the process shows a relation among main components and a sequence in the process. First, the dynamic node manager is a component that performs decision making, and collects information on waiting tasks and available resources through the local scheduler periodically. Then, it calculates the number of nodes required in each partition based on the collected information and decides whether to reallocate currently allocated computing nodes.

When a situation meets the resource reallocation policy, the suitable nodes are selected from the database, and the nodes are rebooted to the new disk image so that the selected computing nodes are allocated to a new partition. After assigned to a new partition, finally the database information is updated.

3 Results and Discussion

Figure 3 shows a testbed to demonstrate the feasibility of our approach. In order to manage heterogeneous platform-based task execution environments and services, bare-metal environment is advantageous in providing stable services. Though the

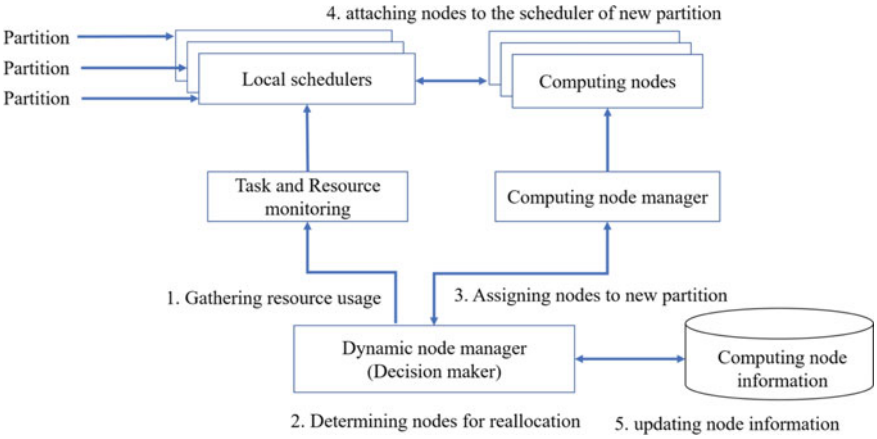

Fig. 2 The main components for dynamic resource reallocation and a sequence in the process

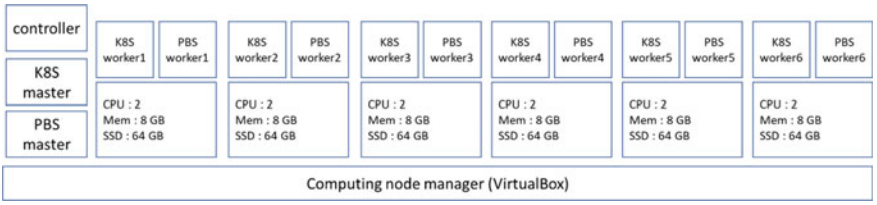

Fig. 3 A testbed environment to demonstrate the feasibility of dynamic resource reallocation

testbed was configured in a virtual machine, we can easily apply this environment to the real computing node of supercomputer by changing computing node manager from a hypervisor of VirtualBox to a MaaS (Metal as a Service) management tool.

Figure 4 confirms that overall throughput is improved by dynamically reallocating resources for the test scenario. In the initial environment, the node managed by the PBS scheduler and the node managed by Kubernetes are 3 nodes respectively, and 2 cpu processors per node. The High Performance LINPACK (HPL) benchmark requests 4 cpu resources per job, and the pbs-worker2 and pbs-worker3 nodes are assigned as shown in Fig. 4a. Among all idle resources, the resources allocated to the HPC partition are not sufficient to process the HPL job requiring 4 cores, so the jobs are waiting in the queue. At this time, based on the resource reallocation policy, the node of the DIC partition is reallocated (pbs-worker4) to the HPC partition. As in Fig. 4b, another job is allocated resources and executed, indicating that it is changed to the running state.

Two jobs using spark framework that request resources of DIC partition through Kubernetes are submitted in Fig. 4c. Each creates one driver and one work process, and each process requires 2 cores. Since there are 4 available cpu cores, 2 nodes with 2 cores are available, so one job can be executed, but the driver process preempts 2

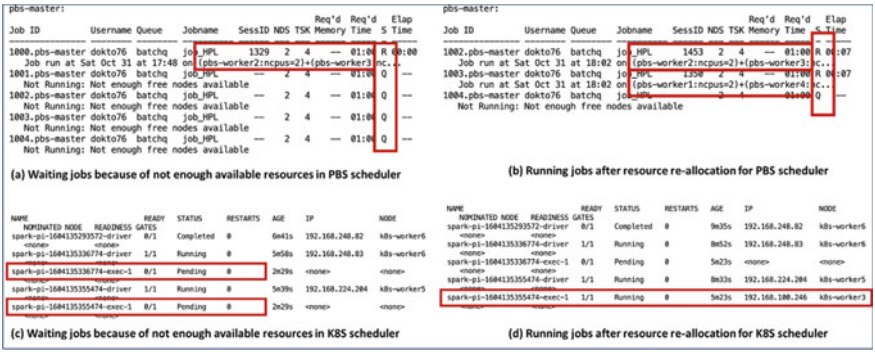

Fig. 4 A sample scenario to show the improved throughput by applying dynamic reallocation of computing nodes to each partition

cores in each job with the Kubernetes scheduler as non-preemption, so the worker process cannot be allocated resources result in pending of the task. After making a decision about the resource reallocation, the pbs-worker3 node in the HPC partition is reassigned to the DIC partition and switched as k8s-worker3. And then the resource is allocated to another worker of the previous pending task in DIC partition and all task changed to running state (Fig. 4d).

4 Conclusion

This work enhances overall system utilization of supercomputer as dynamic reallocating resources and solves the conflict problems between heterogeneous platforms executed on the same computing node by applying node level reallocation with a bare-metal environment. We implement a testbed and simulate a test scenario to demonstrate the feasibility of dynamic resource reallocation and the result shows that our approach can improve system utilization.

Acknowledgements This work was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. CRC21011).

References

1. Lee S, Park JW, Jeong K, Hahm J (2021) Implementation of a container-based interactive environment for big-data analysis on supercomputer. In: Park JJ, Fong SJ, Pan Y, Sung Y (eds) Advances in computer science and ubiquitous computing (Lecture notes in electrical engineering), vol 715. Springer, Singapore. https://doi.org/10.1007/978-981-15-9343-7_58

2. Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz RH et al. (2011) Mesos: a platform for fine-grained resource sharing in the data center. In: USENIX Symposium on Networked Systems Design and Implementation (NSDI)
3. Ayyalasomayajula, West K (2017) Experiences running different work load managers across cray platforms. In: Cray User Group conference (CUG'17)
4. Jupyter Notebook. [Online] Available: https://jupyter.org