Proceedings of the APAN – Research Workshop 2017
ISBN 978-4-9905448-7-4

Prototyping Workload-based Resource Coordination for Cloud-leveraged HPC/BigData Cluster Sharing

Namgon Lucas Kim and JongWon Kim

***Abstract* — Recently high-performance computing (HPC) and BigData workloads are increasingly running over cloud-leveraged shared resources, meanwhile traditionally dedicated clusters have been configured only for specific workloads. That is, in order to improve resource utilization efficiency, shared resource clusters are required to support both HPC and BigData workloads. Thus, in this paper, we discuss about a prototyping effort to enable workload-based resource coordination for cloud-leveraged shared HPC/BigData cluster. By taking OpenStack cloud-leveraged shared cluster as an example, we demonstrate the possibility of workload-based bare-metal cluster reconfiguration with interchangeable cluster provisioning and associated monitoring support.**

***Index Terms* — HPC/HTC workload, BigData workload, cloud-based shared cluster, dynamic resource configuration, and bare-metal cluster provisioning.**

I. Introduction

NOWDAYS we can easily realize diversified applications at a low cost owing to the emerging cloud-first computing paradigm that leverages flexible resource pooling. In particular, high-performance computing (HPC) and BigData workloads are increasingly spreading over cloud-leveraged shared resource infrastructure to enjoy its scaling and reliability benefits. Thus it is important to leverage the resource pooling power of hyper-scale cloud-based shared clusters, while balancing the dedicated engineering for HPC MPI (message passing interface) parallel computing workload and/or data-intensive BigData computing/storage workload.

This work is supported in part by a collaborative research project of PLSI supercomputing infrastructure service and application, funded by Korea Institute of Science & Technology Information (KISTI). Also, this work is supported in part by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. R7117-16-0218, Development of automated SaaS compatibility techniques over hybrid/multisite clouds).

Namgon Lucas Kim and JongWon Kim are with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju, Korea (e-mail: {namgon, jongwon}@nm.gist.ac.kr).

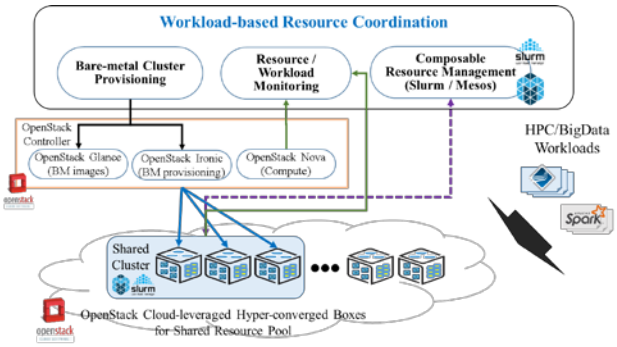


Fig. 1. Workload-based cluster resource configuration: Concept.

However, traditionally, dedicated clusters for HPC and BigData parallel workloads have been separately configured only for chosen workload and thus most of the dedicated clusters could not flexibly match and utilize the full capacity of cluster resources. To improve the efficiency of resource utilization, various types of shared clusters have been proposed [4-9]. Besides, the growing popularity of x86 hardware and Linux operating system is accelerating the increasing adoption toward hyper-converged (i.e., compute/storage/networking integrated) cluster nodes (denoted as boxes in this paper) [1]. Thus, it is becoming cheaper and easier to flexibly support both HPC and BigData workloads on a single cloud-leveraged cluster of hyper-converged boxes, which are to be coordinated with composable resource management.

Thus, in this paper, we discuss about a prototyping effort to enable workload-based resource coordination for cloud-leveraged shared HPC/BigData cluster. The workload-based resource coordination (and thus sharing) is coordinated by an entity called as composable resource management, where software-based resource management for the required dynamic coordination is prototyped with resource management APIs. More specifically, as depicted in Fig. 1, we enable workload-based cluster reconfiguration over hyper-converged SmartX Boxes [2], clustered with open-source OpenStack cloud infrastructure software [3]. That is, workload-based cluster resource coordination is designed

and prototyped over the shared resource pools of OpenStack cloud-leveraged cluster. Especially flexible and automated resource coordination is realized by leveraging bare-metal OpenStack cloud provisioning. By taking OpenStack cloud-leveraged shared cluster as an example, we demonstrate the possibility of workload-based bare-metal cluster reconfiguration with interchangeable cluster provisioning and associated monitoring support.

II. Cloud-leveraged Cluster Sharing for HPC/BigData Workloads

Based on the coordination power of composable resource management, the cloud-leveraged cluster sharing should serve multiple heterogeneous workloads in general. If we configure a dedicated cluster over a pay-per-use (mostly in VM unit) public cloud, it can become easily inefficient when the demanded workload does not match with the configured resources. Thus, in order to efficiently manage and operate cloud-leveraged shared cluster, the composable resource management should manage shared resource pools based on both workload-aware resource coordination (i.e., reconfiguration) and scheduling policy.

As discussed above, the composable resource management for cloud-leveraged shared cluster usually includes workload management to take complex and difficult scheduling logic in charge. The shared cluster needs to flexibly manage the resource allocation for workloads by utilizing resource management APIs. Moreover, there are several early studies to apply one shared cluster for heterogeneous workloads such as HPC and BigData. First, Hadoop on HPC has ported Apache Hadoop as a BigData processing framework to execute on an HPC cluster [4]. Also, Univa supports API-based Apache Mesos resource scheduling [5] through universal resource broker (URB) [6] as a workload-aware tool for grid-computing-style resource management. That is, scheduling-based resource coordination is supported with Mesos software frameworks without modifying Univa Grid Engine. YARN-MPI [7] has modified YARN as a resource management tool for Apache Hadoop cluster that enables the running of MPI-based parallel computing. Note however that most of these proposals require specialized implementation to support other additional (i.e., not designed initially) workloads.

Moreover, we may selectively choose the type of node (i.e., bare-metal or virtual machine) when configuring cloud-leveraged shared clusters. A bare-metal cluster can exhibit more computing power than a virtual machine cluster [8]. For example, we can provision cloud-leveraged HPC cluster with a highlighted focus on bare-metal clustering for HPC workloads [9]. Similarly, BigData clusters can be enabled over workload-customized provisioning of bare-metal resource boxes with iSCSI-based storage to improve its overall performance [10]. This work is indeed quite close to our work, except that it only considers BigData cluster with iSCSI-based storage and PXE+TFTP capability.

In summary, first, without any main update of related software, the resource coordination module in the proposed prototype should execute dynamic provisioning running over reconfigurable HPC/BigData cluster. Also, we could utilize more flexible resource management including cloud-leveraged bare-metal provisioning. Finally, we adopt iPXE+HTTP to deploy bare-metal images to reduce the provisioning time for resource coordination.

III. Prototyping Workload-Based Resource Coordination

A. Cloud-leveraged Cluster and Resource Coordination

As depicted in Fig. 1, the proposed cloud-leveraged shared cluster is built with hyper-converged bare-metal nodes. In order to flexibly install and operate shared cluster over the OpenStack cloud environment, we leverage OpenStack Ironic bare-metal node provisioning to prepare the targeted shared cluster nodes (i.e., bare-metal instances) with bare-metal images including related libraries for a targeted cluster. Thus, the proposed workload-based resource coordination should be able to support bear-metal image generation/management and consistent cluster configuration management, as part of automated cluster provisioning. The automated provisioning is critical since we need to frequently go through so-called CRUD (create, read, update, and delete) of shared cluster nodes to match with time-dependent workload variations. That is, we need to define the life-cycle (tied with CRUD capability) of desired resource coordination, and then design and implement automated cluster provisioning (with reconfiguration). Also, depending on operation policy and resource status of whole shared cluster, the required amount of resource slices is chosen from available resource pools. The selected resource slices allocated and then configured to execute the demanded workloads under the coordination of composable resource management. In addition, the composable resource management needs to be periodically updated about the operation status data of shared cluster so that it can continuously monitor resources and workloads.

This kind of required provisioning for cloud-leveraged shared cluster can be divided into following two stages.

First, we prepare well-arranged bare-metal images and execute the node creation by downloading (from public or private repository) and installing them into the selected bare-metal nodes from OpenStack cloud resource pool. Note that all bare-metal images should be ready to communicate with the proposed resource coordination via proper agents pre-installed.

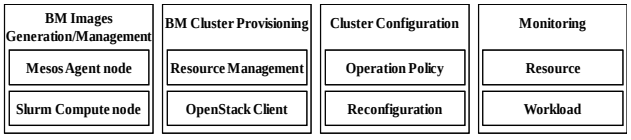
Second, when the shared cluster is ready to start the operation, we need to continuously apply coordination actions to sustain the operation of shared cluster. The key feature of required resource coordination is dynamic cluster reconfiguration supported by composable resource management. Note that in order to match diverse workload demand, the status of resources and workloads should be continuously monitored. Also resource coordination need to cover the required orchestration of shared cluster operation. For this, eventually, resource management APIs are to be implemented to support various reconfiguration of shared

cluster operation. In addition, this kind of resource coordination can be executed only if we have sufficient access/control authority (i.e., privilege) on the underlying shared cluster nodes and its operation. Thus, for each cluster, we typically arrange cluster compute/agent nodes and coordinate them via a cluster master node. Note that, if the master node cannot access/control compute/agent nodes, we may directly control troubled nodes.

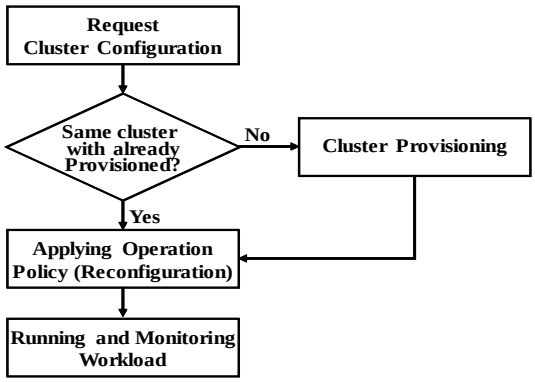
To execute workload on the shared cluster, reconfiguration request file is needed to setup workload type (HPC, BigData, mixed), scheduling type, maximum wait time / min-max requirements for resource, and cluster scaling. Meanwhile the shared cluster operator prepares operation policy to define the number of simultaneous users, maximum amount of resource per user, maximum wait time, and others. Thus, it is required to verify the requested reconfiguration files for its completeness as well as operational conflicts. For example, if some reconfiguration parameters are not reasonable, we need to reject (or change to default) with warning messages.

B. Prototype Implementation

We implement a prototype of coordination of composable resource management using OpenStack-based cloud resource pool. Fig. 2 shows an overview of prototype implementation including building blocks and associated procedures. As explained already, we adopt OpenStack Ironic [11] and use bare-metal node images for HPC cluster with Slurm and BigData cluster with Mesos, respectively. Also, the Mesos agent node image is pre-installed with Mesos and related packages, following OpenStack bare-metal image format. Similarly, the Slurm compute node image is prepared by utilizing OpenHPC [12] packages and configured for efficient HPC cluster operation. One thing to mention is that both OpenStack and OpenHPC packages can support bare-metal node provisioning. Thus, in this work, we only integrate cluster management (i.e., reconfiguration related) features into that Slurm compute image. Slurm requires synchronization files for authenticated cluster operation and the consistency of sync files across cluster nodes is important. Finally, all bare-metal images are created by utilizing open-source Diskimage-builder [13].



(a) Building blocks



(b) Overall procedure

Fig. 2. Prototype implementation of resource coordination.

Each cluster node is provisioned by the cooperation of bare-metal node provisioning and composable resource management. Also, since we utilize OpenStack CLI to execute required OpenStack commands, we internally embed OpenStack client for OpenStack CLI. Note that all OpenStack bare-metal images have cloud-init software and user agent installed so that it can support the centralized control of all cluster nodes. Also the operation policy is pre-installed before creating and running the shared cluster. Finally, at this stage of prototype implementation, we fix both Mesos and Slurm masters on the desired node and run independently. Also, we execute both direct and indirect monitoring every 10 minutes.

IV. Evaluations on Prototype Implementation

We verify the feasibility of proposed workload-based resource coordination by prototyping it over an OpenStack cloud-leveraged shared cluster. Table 1 shows the hardware specification of the shared cluster with three nodes, which is decomposed into a small-size resource pool of 1 master and 2 compute nodes, respectively. All these nodes are installed with CentOS 7.3, OpenStack Ocata, Slurm 16.05, and Mesos 1.1.0. In addition, OpenStack controller node (Intel Xeon X3330 and 8GB DDR2 RAM) is separately setup. Also all cluster nodes can be power controlled by IPMI (Intelligent Platform Management Interface).

TABLE I
Shared cluster hardware specification (2 compute nodes)

	Specification
CPU	12 cores, 24 threads (Intel Xeon D-1528@2.3Ghz)
RAM	DDR4 64GB (PC4-17000)
Storage	400 GB SSD (Intel S3500)
Network	2-port 10GbE (EA), 2-port 1Gbe (EA)

Also, for benchmarking workloads, we use Intel MPI Benchmark (IMB) [14] for HPC workload and Spark-Perf [15] for BigData workload, respectively. For HPC workload, IMB-NBC is selected as the test case and it is executed with default parameters such as 10~1000 iterations, 0B ~ 4MB message size, and 2/4/8 processes. Note that, since each compute node has 12 logical cores, we do not apply Hyper Threading for MPI workload to avoid potential performance degradation. In addition, K-Means test case is selected for BigData workload with Scale_Factor 1, which kind of matches 20 instance workload of Amazon AWS EC2 M1.xlarge.

A. Evaluation: Workload-based Cluster Coordination

First, we check the provisioning performance of bare-metal cluster nodes by adopting OpenStack Ironic bare-metal provisioning tool with pxe_ipmitool driver over 1Gbps network inter-connection: single node and multiple nodes. Fig. 3 shows the total elapsed time comparison for BigData cluster provisioning for different provisioning options. Note that, in this case, the bare-metal image size of BigData cluster is bigger than that of HPC cluster and thus the overall provisioning takes longer time. Also with OpenStack Ironic bare-metal node provisioning, we need two types of images: deploy image and user image. The deploy image includes OpenStack IPA (Ironic Python Agent) that prepares the bare-metal node provisioning itself. This deploy image is identical with both HPC and BigData cluster provisioning and consists mostly of Kernel (33MB) and Ramdisk (334MB). On the other hand, separate user images are selectively used for workload-based cluster configuration, which is decomposed into three partitions for Kernel (5.2MB), Ramdisk (38MB), and User image (714MB).

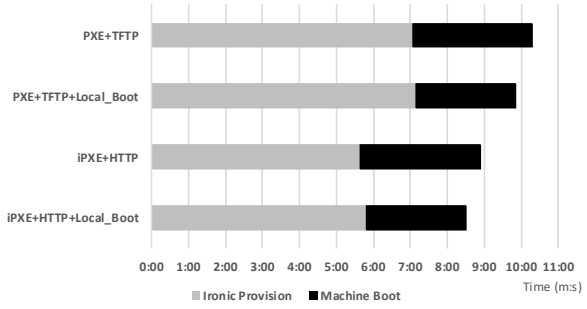


Fig. 3. Provisioning time comparison (bare-metal cluster node for BigData workload).

We represent the overall cluster node provisioning time by separating time associated with OpenStack Ironic bare-metal node provision and node boot. The time for OpenStack Ironic bare-metal provision is measured by OpenStack Nova compute service, which includes times consumed for deploying images and node boot/reboot. In comparison, node boot time indicates the time consumed for final booting after the completion of OpenStack Ironic provisioning process, which cannot be measured by OpenStack Nova. Fig. 4 shows the detailed time comparison of cluster node boot according to several booting options. The firmware option means that we load UEFI firmware before loading OS boot loader. Also, if we do not use local_boot option, boot loader consumes longer time since it needs to receive images from the controller node. User space option commonly takes longer and DHCP-interface option needs longest boot time around 30 secs. Finally, as shown in Fig. 3 and Fig. 4, we can compare the overall reconfiguration time according to the choices on bear-metal image delivery and local_boot option (about writing user images to local storage or not). The comparison result is showing that HTTP is better than TFTP in general, and local_boot option can reduce cluster node boot time.

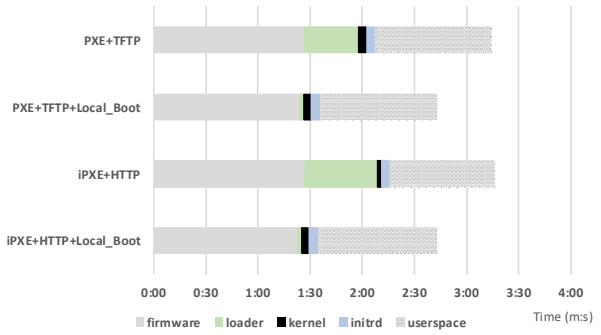
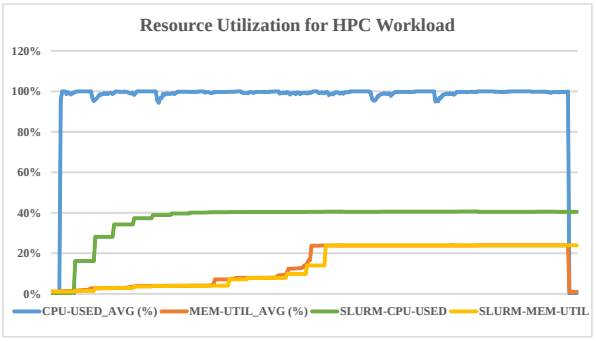


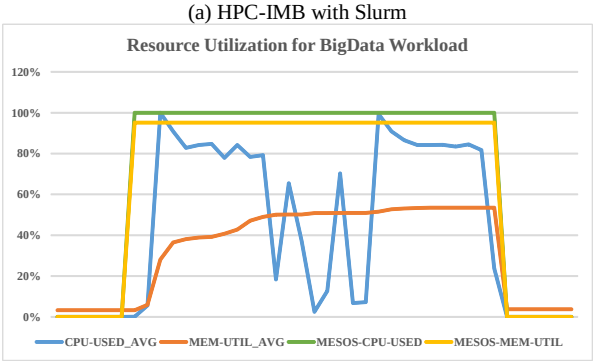
Fig. 4. Boot time comparison (bare-metal cluster node).

B. *Evaluation: Workload Execution and Monitoring*

Now we verify workload execution performance by monitoring the operation (i.e., running) status of workload execution. We independently perform each experiment on the same shared cluster, which is provisioned beforehand. Fig. 5 shows the resource usage patterns to monitor several stages (e.g., before/after and during) of workload execution. We insert 1-min idle time between experiments and reserve key resources (CPU and RAM) for monitoring to avoid unnecessary resource contention.



(a) HPC-IMB with Slurm



(b) Spark-Perf with Mesos

Fig. 5. Monitoring result for workload execution.

From Fig, 5, each workload shows two different monitoring results: one is directly collected from cluster compute nodes and the other is collected from the master node for composable resource management. In case of HPC workload, CPU usage shows different patterns between two monitoring options, since Slurm collects 5-min average load with Linux kernel APIs. On the other hand, Mesos-based master node only manages the allocated resource status, which makes the allocated usage of CPU and RAM fixed. Note also that, with

direct monitoring, the usage impact of operating system kernel and daemons is included. In summary, with two different monitoring results, we can verify that the resource coordination module could figure out and assist the workload-based resource usage.

C. Provisioning Scalability Expectation

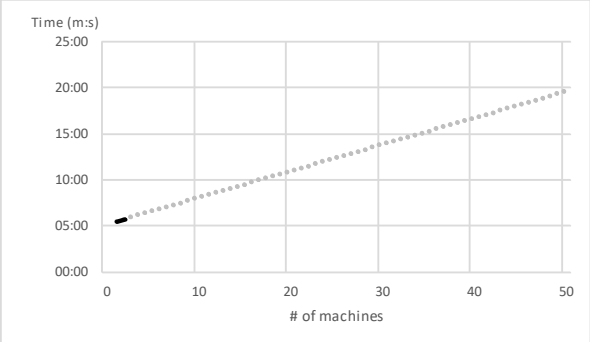


Fig. 6. Estimated node boot time.

Finally we check cluster node scalability by estimating the cluster provisioning time for up to 50 nodes. Based on the provisioning time performance in Section IV and with iPXE local_boot option, we depict the estimated time in Fig. 4. It is estimated that around 20 minutes is consumed to simultaneously provision a 50-node cluster by utilizing the 17-sec delta time for one additional node provisioning. Also, 1GB-size bare-metal image transfer takes around 13 secs over 1Gbps network (assuming 600 Mbps throughput). In addition, we assume involved delay due to OpenStack controller node.

V. Conclusion

We presented a prototype of resource coordination module that performs bare-metal cluster coordination for both HPC and BigData workloads based on an OpenStack cloud-leveraged resource pool. We also verified the possibility of cluster reconfiguration depending on the types of workloads by leveraging automated OpenStack Ironic bare-metal provisioning. Note that the consumed provisioning delay for shared resource clustering is 8 minutes and 30 seconds, only with two compute nodes. It might be a non-negligible overheads for overall shared cluster performance. However, remember that usually the combined workloads of HPC and BigData will be running for several hours.

References

[1] J. Kim, "Realizing Diverse Services Over Hyper-converged Boxes with SmartX Automation Framework," in *Proc. Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2017)*.

[2] A.C. Risdianto, J. Shin, and J. Kim, "Building and Operating Distributed SDN-Cloud Testbed with Hyper-convergent SmartX Boxes," in *Proc.* 6th EAI International Conference on Cloud Computing, Daejeon, Korea, Oct. 2015.

[3] OpenStack, http://openstack.org.

[4] A. Luckow, et al., "Hadoop on HPC: Integrating Hadoop and Pilot-based Dynamic Resource Management," arXiv preprint arXiv:1602.00345 (2016).

[5] B. Hindman et al., "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," In *Proc NSDI* 2011.

[6] Univa URB, http://www.univa.com/resources/files/urb.pdf.

[7] MPICH2-Yarn, https://github.com/alibaba/mpich2-yarn.

[8] C. G. Kominos, N. Seyvet, and K. Vandikas. "Bare-metal, virtual machines and containers in OpenStack." In *Proc. Innovations in Clouds, Internet and Networks (ICIN)*, 2017.

[9] P. Rad, et al. "Benchmarking bare metal cloud servers for HPC applications." In *Proc. Cloud Computing in Emerging Markets (CCEM), 2015*.

[10] A. Turk, et al. "An experiment on bare-metal bigdata provisioning." *In Proc. 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[11] OpenStack Ironic, https://wiki.openstack.org/wiki/Ironic.

[12] Karl W. Schulz, et al., "Cluster Computing with OpenHPC," In *Proc. HPCSYSPROS16*, 2016.

[13] Diskimage-builder, http://docs.openstack.org/developer/diskimage-builder.

[14] Intel, Intel MPI Benchmarks. https://software.intel.com/en-us/articles/intel-mpibenchmarks.

[15] Spark-Perf, https://github.com/databricks/spark-perf.