A Container-based Approach to HPC Cloud

Korea Institute of Science and Technology Information
www.kisti.re.kr

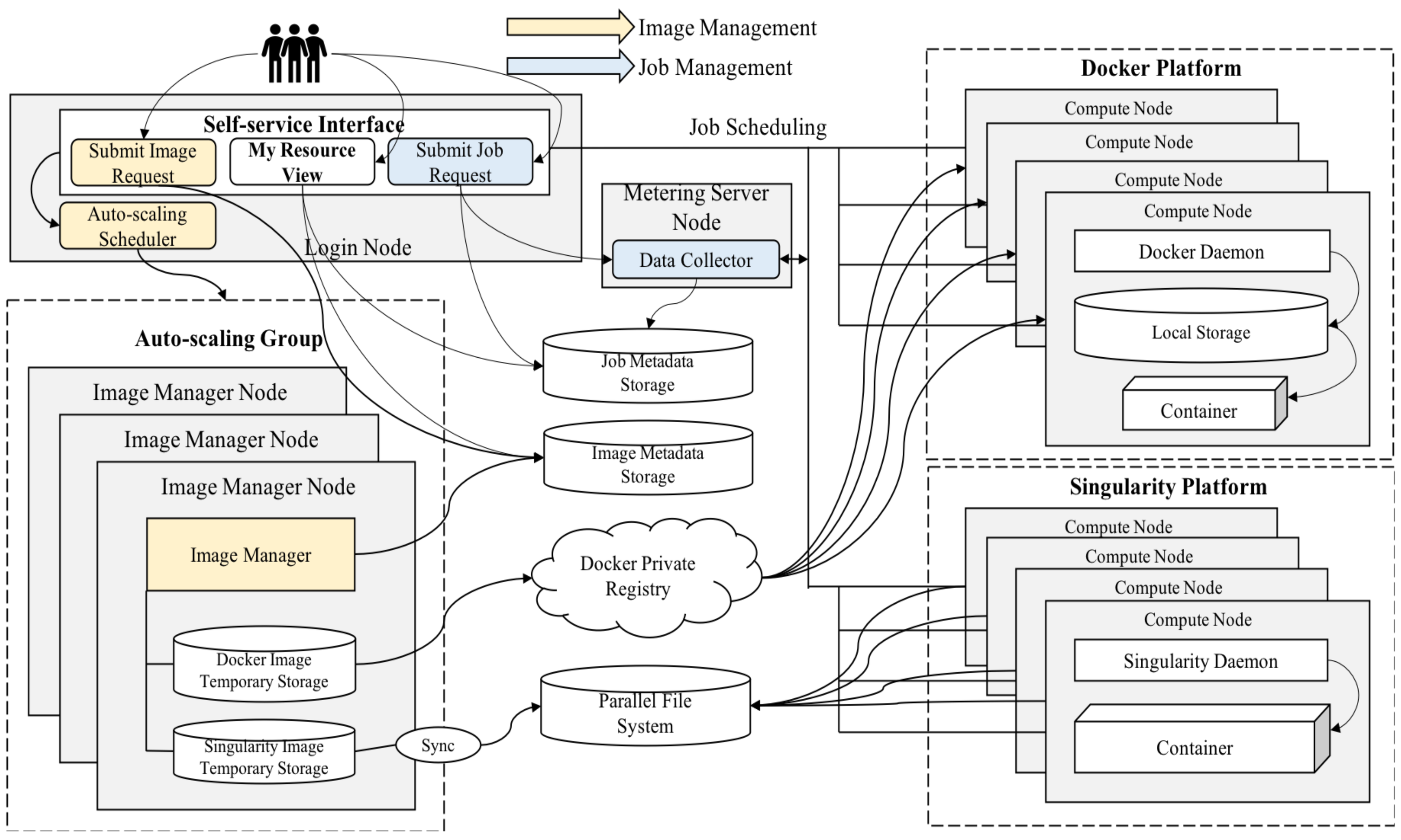
Guohua Li (ghlee@kisti.re.kr), Joon Woo and Taeyoung Hong
Korea Institute of Science and Technology Information (KISTI), Daejeon, Korea

1. Introduction

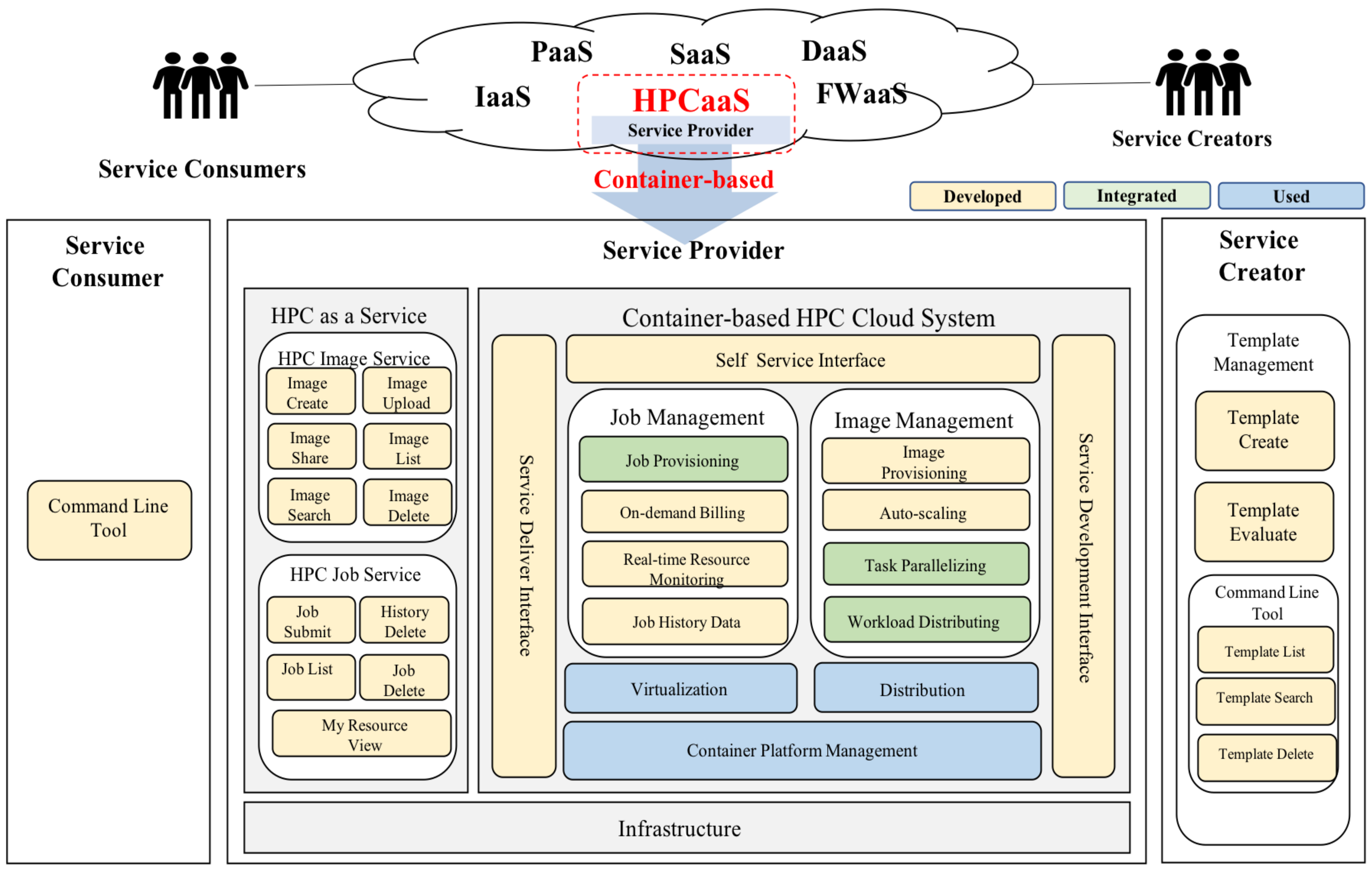
Recently, the virtual machine (VM)-based high-performance computing (HPC) service has been provided in the cloud environment to satisfy portability, flexibility, scalability, and reduction of deployment costs in the HPC field. However, performance issues and workload management issues due to the limitations of VM are reducing the resource utilization of HPC users. Therefore, we aim to provide a lightweight container-based cloud environment to HPC users. This container-based approach consists of two main components: the image management system and the workload management system. We have designed and implemented the system workflow and architecture considering ease of use and efficiency of management. The results have been obtained by comparing network performance, MPI performance and a simple machine learning code – MNIST between bare-metal and container-based (both in Docker and Singularity) environments.

2. Workflow and Architecture

- System Workflow

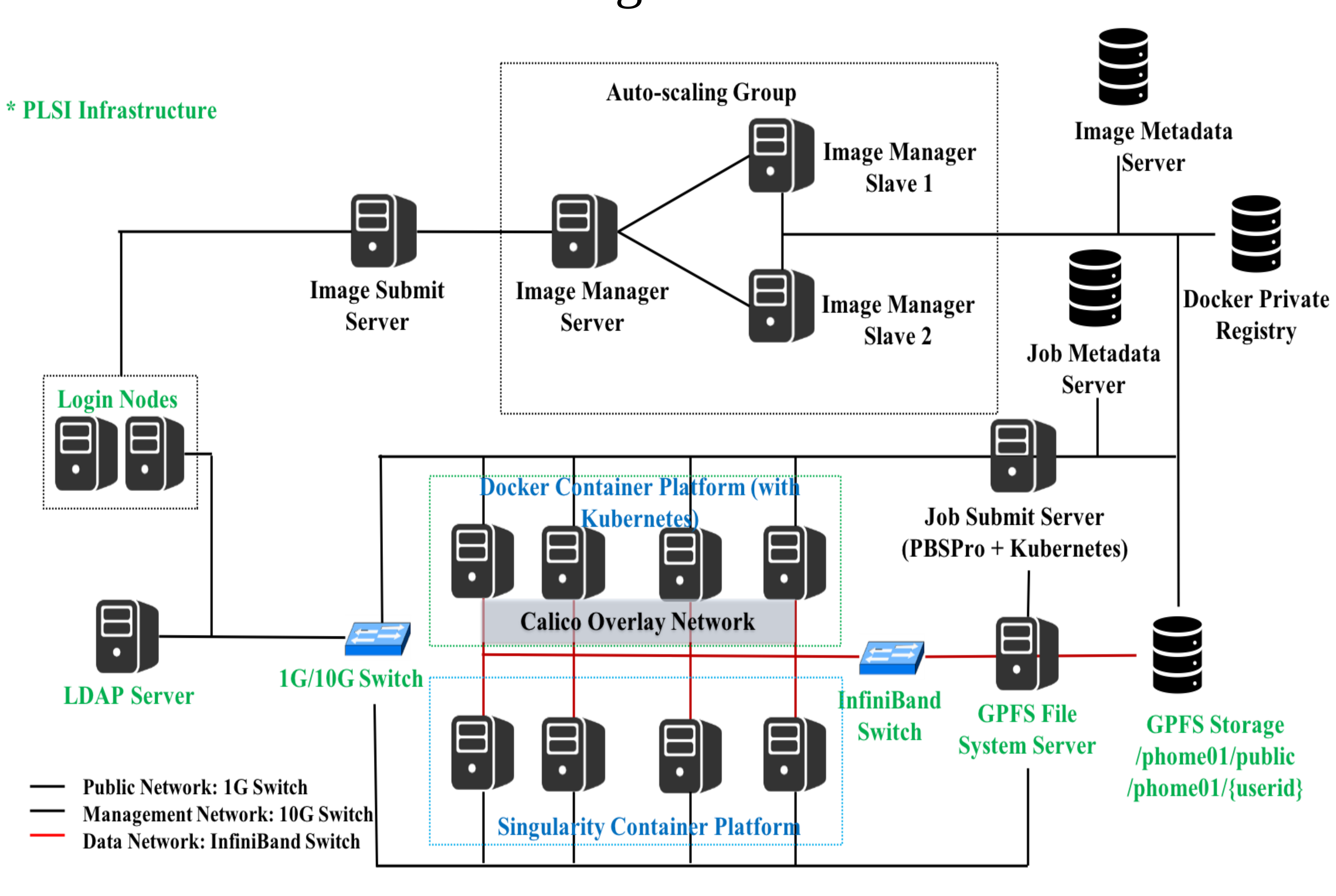


- System Architecture



3. Implementation

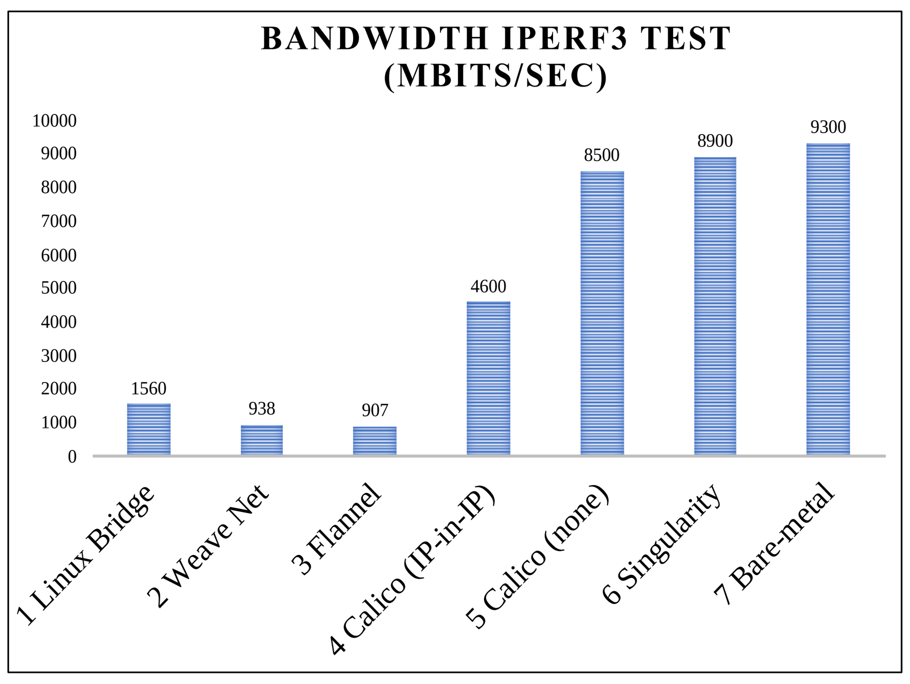
- Test-bed Cluster Configuration



4. Results

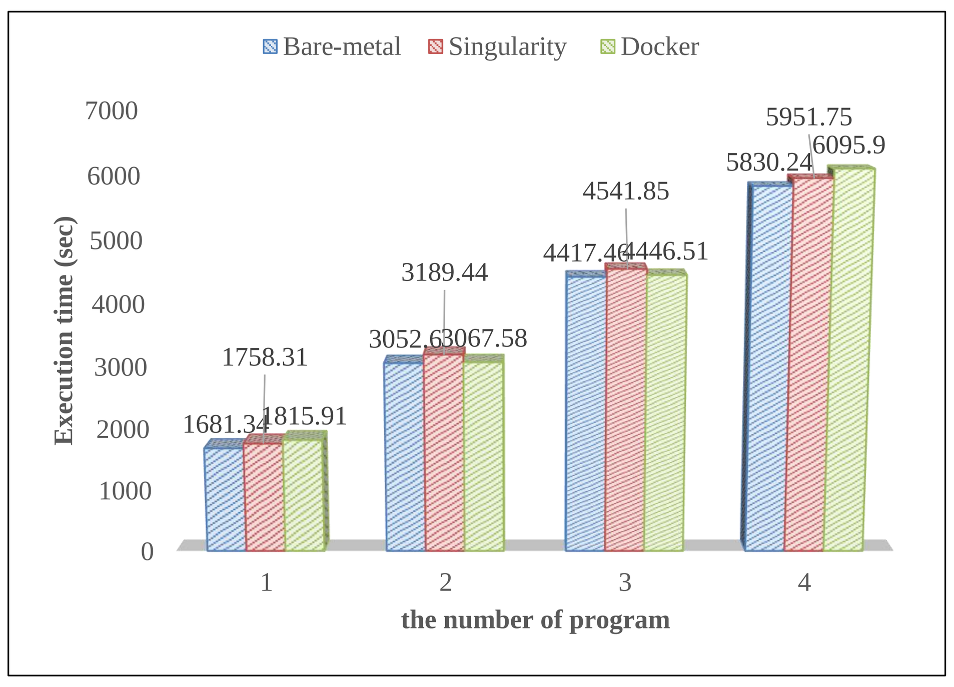
- Network Bandwidth Test

(deployed with one container per node using iperf3)



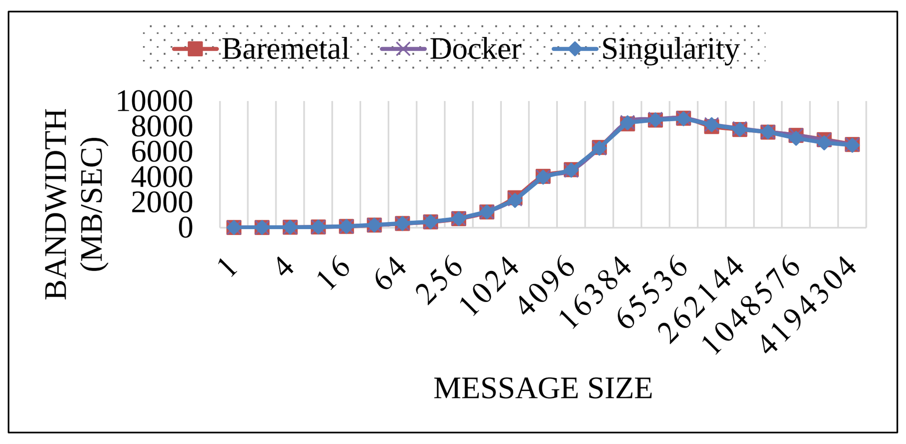
- MNIST Execution Time Test

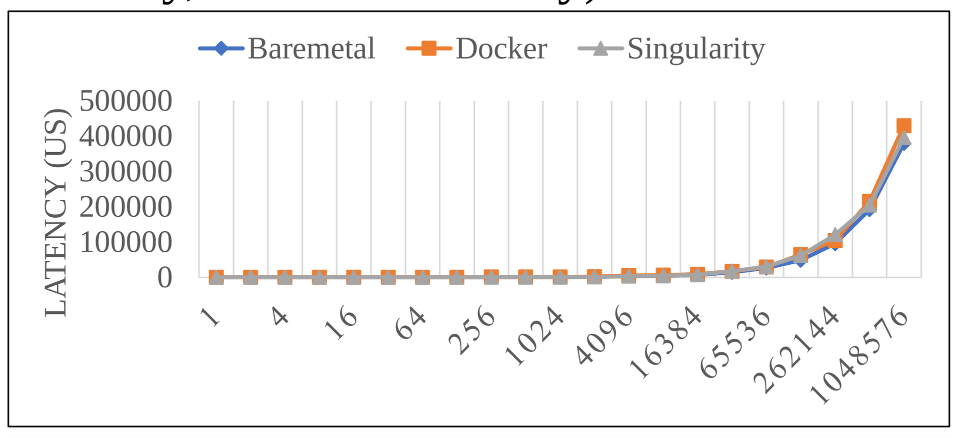
(deployed with example code MNIST – a simple classifier using TensorFlow)

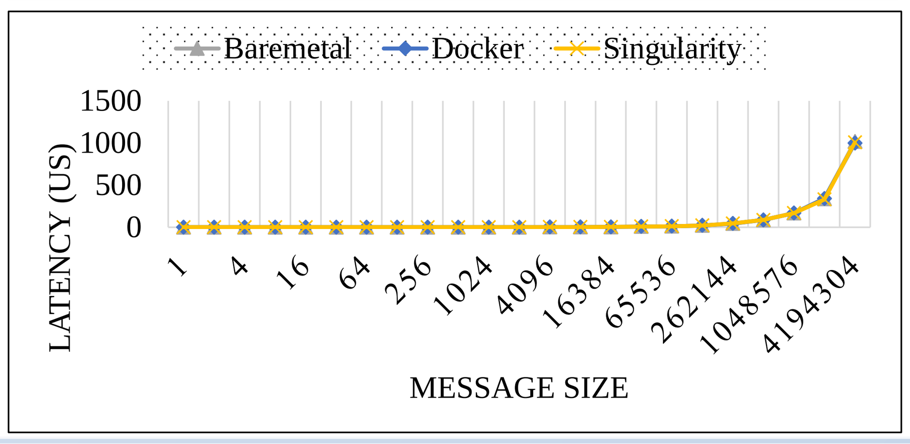


- MPI Performance Test

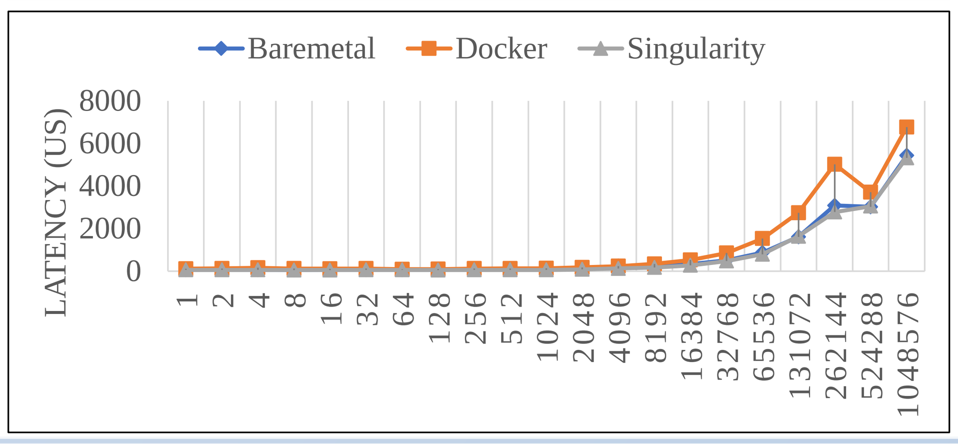
(deployed with two nodes using OSU MPI benchmark tool)







(deployed with 4 nodes (with 40 cores) for running MPI parallel tasks to check MPI all-to-all personalized exchange latency, broadcast latency)



5. Conclusions

This container-based approach to HPC cloud contributes to solving performance and workload management issues for serviceability. Performance issues are verified by comparing the results of network bandwidth, MPI bandwidth/latency, TensorFlow-based MNIST code execution time in Bare-metal, Docker and Singularity* environments. Image workload management has been developed based on Celery-Redis framework to distribute users' image requests. MPI Job workload management has been developed based on integrating some functions of Kubernetes (Docker Orchestration Platform) and PBSPro (Job Scheduler). For ease of use, all workload-related commands are provided through a simple CLI tool (developed in Python). This research is expected to be an important study for providing cloud services in a supercomputing infrastructure environment.

* Singularity can be used with any Job Scheduler directly.

[Acknowledgement]

This study has been performed as a subproject of KISTI's project "The National Supercomputing Infrastructure Construction and Service [K-19-L02-C01-S01]".