Multiple HPC Environments-Aware Container Image Configuration for Bioinformatics Application

Kento Aoyama
Tokyo Institute of Technology / RWBC-OIL, AIST
Tokyo, Japan
aoyama@bi.c.titech.ac.jp

Hiroki Watanabe
Tokyo Institute of Technology / RWBC-OIL, AIST
Tokyo, Japan
watanabe@bi.c.titech.ac.jp

Masahito Ohue
Tokyo Institute of Technology
Tokyo, Japan
ohue@c.titech.ac.jp

Yutaka Akiyama
Tokyo Institute of Technology
Tokyo, Japan
akiyama@c.titech.ac.jp

ABSTRACT

Containers have a considerable advantage for application portability in different environments by isolating process with a small performance overhead; thus it has been rapidly getting popular in a wide range of science fields. However, there are problems in container image configuration when run in multiple HPC environments, and it requires users to have knowledge of systems, container runtimes, container image formats, and library compatibilities in HPC environments.

In this study, we introduce our HPC container workflow in multiple supercomputing environments that have different system/library specifications (ABCI, TSUBAME3.0). Our workflow provides custom container image configurations for HPC environments by taking into account differences in container runtime, container image, and library compatibility between the host and inside of the container. We also show the parallel performance of our application in each HPC environment.

KEYWORDS

Linux containers, Docker, Singularity, Bioinformatics, Multiple environments

1 INTRODUCTION

Containers that contribute to application portability through process isolation are now being widely used in the field of computational science. Today, many researchers run containers in various computing environments such as laptops, clouds, and supercomputers, and that is becoming essential to retain our scientific reproducibility and impact. Contrarily, there are problems in the configuration of container images for HPC applications that run in multiple HPC environments. It requires users to understand the know-how of systems, container runtimes, container image format, and compatibility of those used in HPC environments. In addition, when system hardware, namely, GPU/MPI with host-side library is used, compatibility between the host and the inside of container is required. It makes porting of HPC container hard; thus, these problems are a major obstacle for extensive use of container technology.

In this study, in order to introduce container's techniques and its benefits to one of our HPC applications, MEGADOCK [1], we propose a workflow for container image configuration by considering differences between hosts and container runtime/image/library in multiple HPC environments. In addition, we also show the parallel performance of our application execution using a benchmark dataset in multiple HPC environments.

2 CONTAINER ON HPC ENVIRONMENT

Currently, there are two container environments mainly used by researchers: Docker is the most general container that provides useful toolsets for managing container networks, volumes, resources, clustering, web service provisioning, and many more to be widely used in various computing environments. Singularity is mainly used in HPC systems, which is focused for use by users in the scientific field. It does not require privileges for users and the main focus is on performance, portability, security, and so on.

In that case, Singularity's function of converting Docker container image into Singularity format is helpful. It enables users to download container image from Docker Hub and allows it be available in singularity environment.

2.1 Problems of container image configuration

The number of HPC systems that provide container environments is constantly increasing; thanks to widespread use of Singularity; however, there are several difficulties of the deployment of containers:

1. **Cost for container image preparation**
   Container image recipe formats of Docker and Singularity are different, which allows users cost for preparing container images in each format.
2. **Container image compatibility**
   Singularity supports the conversion of Docker container images into Singularity format, but several errors may occur and may result in failure in some cases. Even if the process succeeds, the application may not be able to run due to differences in security policy.
3. **Library compatibility between host and container**
   HPC applications usually require host hardware resources such as GPU and interconnect with optimized libraries for MPI communications. Runtime errors can occur if there is no compatibility between the libraries of the host's and of the container's.

3 MEGADOCK: A HIGH PERFORMANCE PROTEIN-PROTEIN INTERACTION PREDICTING APPLICATION

MEGADOCK [1] is a protein-protein interaction (PPI) prediction application for large-scale computing environments. Its internal process is mainly based on Fast Fourier Transform (FFT) for docking calculation using FFTW library (or CUFFT library if GPU is available). Each docking calculation is parallelized by OpenMP and CUDA; also task distribution is controlled by the built-in master-worker framework implemented with MPI.

We have been working to improve the application not only in performance but also in portability. Now Docker container images and its recipe (Dockerfile) for GPU-enabled environment are provided to users, and we have performed those parallel performance in the cloud environment of Microsoft Azure [2]. For further advances of MEGADOCK project, we are working to provide container workflow for both Docker and Singularity environments with MPI parallelization that assumes multiple-target environments.

4 HPC CONTAINER WORKFLOW WITH HPC CONTAINER MAKER

To integrate the container workflow of MEGADOCK on multiple HPC environments, we decided to use HPC Container Maker (HPCCM) [3]. It is an open source tool written in Python, which can generate container image recipes for both Docker and Singularity environments from a single Python code. Introducing the HPCCM into our workflow has the following advantages.

a) **Decreasing the preparation cost of container images**
HPCCM can generate a container image recipe for both Docker and Singularity, which will result in the reduction of management costs for recipes and will be helpful for continuous container image maintenance.

b) **Avoiding container image compatibility problems**
With HPCCM, each Docker/Singularity container image can be easily built from each of its own recipe file. There is no need to convert the container image format that led us to avoid image compatibility problems.

c) **Avoiding library compatibility problems**
HPCCM's Building Block feature allows users to introduce software dependency libraries into container images with minimal parameter descriptions. In addition, using the user argument with Building Block, we can easily specify the target library version while generating a container recipe. Explicit and easy specification for library versions for image building is helpful to solve library compatibility problems.

Users can specify the OpenMPI version from user arguments while generating a container recipe with HPCCM; thus, by specifying the version of the MPI library similar to that in the HPC environment, the compatibility problem of the MPI library in each HPC environment can be avoided.

5 EXPERIMENTS AND PERFORMANCE RESULTS

We performed a benchmark docking calculation of using container image generated from our workflow. The HPC environments used for experiments are TSUBAME 3.0, located at Tokyo Institute of Technology, and ABCI (AI Bridging Infrastructure) located at National Institute of Advanced Industrial Science and Technology, both of which provide Singularity environment. Since there are differences in the MPI library environments provided as modules between the two systems, each container image configured as the version of MPI library is the same as the MPI module provided by each host.

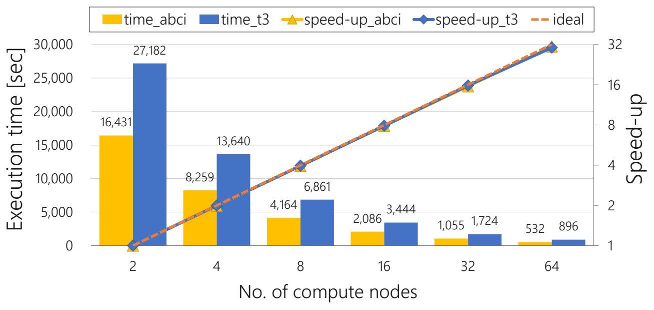

Figure 1: Performance result of MEGADOCK with ZDOCK Benchmark 5.0 (all-to-all, 230 × 230 pairs) on TSUBAME 3.0 (t3) and ABCI (abci).

In the experiment, all calculations were successfully completed with no fatal errors in both environments. The result of compute-node on ABCI that has four NVIDIA Tesla V100 devices showed better performance compared to the result of compute-node on TSUBAME 3.0 that has four NVIDIA Tesla P100 devices. Parallel performances in both environments were confirmed as almost equivalent and over 0.9 in strong scalability on the executions with 64 nodes, this indicates the task load distribution among worker nodes was properly balanced.

6 CONCLUSION

We introduced the HPCCM framework into our HPC container workflow to integrate the container image building flows over multiple HPC environments, and this helped us in successfully preventing MPI compatibility problems. Furthermore, we measured the parallel performance of our HPC container execution using a benchmark calculation in both TSUBAME 3.0 and ABCI environments. We believe container techniques would be beneficial for contributing to the portability of scientific achievements and retaining its reproducibility.

ACKNOWLEDGMENTS

This work was partially supported by JSPS KAKENHI (No. 17H01814, 18K18149), MEXT Program for Building Regional Innovation Ecosystems, and JST Research Complex Program of Japan.

REFERENCES

[1] M. Ohue, T. Shimoda, S. Suzuki, *et al.* "MEGADOCK 4.0: An ultra-high-performance protein-protein docking software for heterogeneous supercomputers," *Bioinformatics*, vol. 30(22), pp. 3281–3283, 2014.
[2] K. Aoyama, Y. Yamamoto, M. Ohue, Y. Akiyama, "Performance evaluation of MEGADOCK protein–protein interaction prediction system implemented with distributed containers on a cloud computing environment," In *Proc of PDPTA'19*, pp. 175-181, 2019.
[3] S. McMillan, "Making Containers Easier with HPC Container Maker," In *Proc of HPCSYSPROS'18*, 2018.