Build and Execution Environments (BEE) : an Encapsulated Environment Enabling the HPC Application Running Everywhere

Anonymous

ABSTRACT

Since different HPC system usually have differet software and hardware configurations, configure and build before running HPC application can take a lot of time and effort for application users. Also, when a HPC system is shared between users and each user gets limited time slots, which is usually not enough to finish their computation process. This requies them to implement checkpoint/restoration feature, so that they can resume work when resources are available. However, these kind of checkpoint cannot be easily migrate to other available machines. Container technology like Docker brings great benefits to application's development, build and deployment process, which can effiectivly solve issues that HPC users are facing. While most cloud computing systems are already equipped with Docker, not much work has been done to deploy Docker on HPC systems. In this work, we propose a Docker-enable build and execution environment for HPC system – BEE. It brings all the benefits of Docker and does not requir system admin level onfiguration. we show that current HPC application can be easily configured to run BEE. It elimites the need for application configurationa and build, also provides comparable performance.

KEYWORDS

High Performance Computing, Cloud Computing, Container.

ACM Reference format:
Anonymous . 2017. Build and Execution Environments (BEE) : an Encapsulated Environment Enabling the HPC Application Running Everywhere. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 2 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

High Performance Computing(HPC) systems have been extensivly used by researchers in academic and industrial of many fields. For example, domian experts uses HPC systems to run large scale physical simulations, big data analysis, or large multi-layer artificial nerual networks. On HPC system, users fisrt need to configure and build their application before they can run their application, since different HPC system usually have differet software and hardware configurations. However, configuring and building large scale applications can take a lot of time and effort. Also, it also requires application users to have enough application technical knowledge. Even worse, when users want to run lagecy application developed years or even decades ago, required dependecy libraries maybe hard to find or have difficulteis compataible with current software stack. Even we can find compataible libraries, the lagency application may also output differently than years before due to today's different hardware. Moreover, commonly used institutional HPC systems usually have limite resource, so that they cannot serve all users at the same time. So, some kind of resource manger is usually intalled in HPC system. so that each user can share part of resource of the HPC system without interfering with each other. However, large scale applications usually requires hours, days or even monthes before they can finish. But allocating such a long time slot is not easy on a busy HPC system. Once a user's allocation is near end, the user must take a checkpoint of the application to avoid lost of progress. Then, the user can chooes whether to wait for another time slot or maunally migrate to another HPC system. However, it imposes several challenges: First, checkpoint feature is not available in HPC applications. Developing checkpoint/restoration requires extensive programming efforts. Second, due to the intrinstic chartacterisitc of computing pattern of HPC applications, taking a checkpoint anytime is not always possible. For example, large scale application usually process stage by stage. Checkpoingting in the middle of a stage is difficult since it is hard to store complicated status information during the computation. Checkpoinging is only possible in between stages. However, if current time slots is not enough to finish the current stage, it can be in serious trouble. Third, migrating to a different HPC system can be difficult too. Different HPC system may have different software stack, e.g. some required libraries may not available or library version is not compatiable. Those problem cannot be fix unless the user has adminstration previlege, which is usually impossible. Also, different hardware architecture or configureation may yeils unexpected results. Even if we have all the compatable execution environment, build the application from scratch is still necessary, which may cost a lot of time for large applications. Cloud computing system usually offers more reliable and easily deployable environment. For example, Linux container technology like Docker enables consistent software and hardware environment for development, build and delopyment. By developing using Docker, developers only need to build their application once in Docker on their local machine, then the application can run on any Docker-enabled machine. It only needs to ship the Docker images to the target machine. Since Docker only create a thin vritualization layer between the host and guest, the performance penalty is negligible. So, Docker can not only benefit cloud user, but also HPC users.

However, Docker is not usually installed on current HPC systesm. The main reason behind it is that Docker requires Linux kernel version to be higher than 3.8, but current main stream Linux kernel version is 2.6. For compatilibly and security consideration, it

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

would take years before HPC systems can upgrade their Linux kenel. Although Docker-enabled HPC systems have been build like Singularlity [] and Shifter [], they either requires specially customized HPC system or Docker daemon. However, those constimizations would limite the praciticalibility to deloping such system to other machines. First, most HPC systems are not allowd to be consomized either in software or hardware by normal users. Second, constomized Docker daemon could bring compaitliy, security issue. Also, maintaining constomization project could be limited, which may affect users. In this work, we propose a new build and execution envrionment that can enable Docker on almost any current HPC systems. We call it Build Execution Environment(BEE). To overcome the Linux kernel version issue and provide more fexlible design environment, we frist create a extra virtual machine layer on top of the host and then deploy Docker on the virtual machine layer. This is a more practical solution beacuse: first, QEMU is usually enabled in current Linux-based HPC system, which allows us to create a virtual machine layer; second, we can easily coustomize virtual machines to make them fully conpatilble with latest vanllia Docker; third, our solution does not requires root/admin previleges of the HPC system, so any user can delopy our BEE to HPC systems. Forth, since we do not consomize Docker, users can always get benefits from the features of the lastes Docker. More specifically, the contributions of this paper include:

(1) **Docker-enabled environment on HPC system** By using BEE, we can provide Docker-enable environment into current HPC systems. HPC users can easily Dockerize their application and run on BEE without worrying about
(2) **User space delopyment on unmodified HPC systems**
(3) **Stadard latestest Docker support**
(4) **Additional hardware virtualization**
(5) **Cross platform live migration**
(6) **Hybrid HPC and Cloud environment**

2 RELATED WORK

3 DESIGN

4 EVALUATION

5 CASE STUDY

6 FUTURE WORK

7 CONCLUSIONS