Type 2: Enabling Multidisciplinary Collaboration with Containerization Technologies

Jian Tao*
Strategic Initiatives
Texas A&M Engineering Experiment Station
and
High Performance Research Computing
Texas A&M University
College Station, TX, USA
jtao@tamu.edu

Huasong Shan, Qingyang Wang
Computer Science and Engineering
School of Electrical Engineering and
Computer Science
Louisiana State University
Baton Rouge, LA, USA
hshan1@lsu.edu
qywang@csc.lsu.edu

Q. Jim Chen
Department of Civil and
Environmental Engineering
and
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA, USA
qchen@lsu.edu

***Abstract*—Information technology (IT) plays a more and more important role in scientific research nowadays. It is crucial for researchers whose work relies heavily on computers to take advantage of the latest development of IT to help speed up their scientific discoveries. Cloud computing, especially containers built upon the so-called containerization technologies enable an extremely flexible and portable computing environment that could be tailored to meet a wide spectrum of computing requirements for scientific research. In this paper, we present our efforts to build SIMULOCEAN, a web-based computing platform for scientific applications on containers. SIMULOCEAN enables rapid sharing and integration of simulation applications and computational tools among connected communities. It also provides a collaborative platform to enable multidisciplinary and integrative research.**

I. Introduction

Compared to the quick adoption of cloud computing technology in industry, the academic community, and especially the computational science community as a whole, has been slow to make the move, partially because of the lack of investment in cloud-ready systems from the National Science Foundation (NSF) and other major funding agencies. For years, many researchers and engineers who did not run large-scale applications regularly were inhibited by the efforts needed to gain the specialized knowledge to effectively use high performance computing (HPC) resources for their research. Their time could be better spent on their research if they did not have to worry about how to run their applications. It was not a surprise that under the NSF Cloud initiative, in 2014, NSF funded two $10 million projects *Chameleon* and *CloudLab* to enable the academic research community to drive research on a new generation of innovative applications for cloud computing and cloud computing architectures [1]. In January 2017, NSF moved one step further by announcing the participation of cloud providers, including Amazon Web Services (AWS), Google, and Microsoft, in its flagship research program on big data, Critical Techniques, Technologies and Methodologies for Advancing Foundations and Applications of Big Data Sciences and Engineering (BIGDATA) [2]. Through this program, AWS, Google, and Microsoft will provide cloud credits/resources to qualifying NSF-funded projects, enabling researchers to obtain access to state-of-the-art cloud resources.

The cloud computing technologies have rapidly evolved in recent years due to explosive growth in the demand in the industry. The improvements in the technologies result in greatly reduced overhead caused by virtualization on most cloud systems. Various performance metrics such as network performance, storage speed, memory size, and compute capabilities of cloud computing systems are approaching those of comparable bare metal homogeneous HPC systems. With the latest GPU passthrough/penetration technologies [3], the computing power of GPUs can also be explored in virtualized systems.

For scientific application development and deployment, one of the major benefits of using cloud computing systems is the portability across different cloud platforms via virtualization technologies. In a virtual image, a developer can include both the operating system and software environment that are required for the execution of a scientific software package. End users are guaranteed to have a functional software out of box and run in series or parallel on any cloud systems they are comfortable with. This makes it easier to share not only data but also software tools in the same package. The virtualization are very often in forms of virtual machines via the so-called hypervisors such as Oracle (VirtualBox), VMWare (vSphere), and Microsoft (Hyper-V). Unfortunately, the size of virtual machines as well as virtual images are usually very large as they contain the whole operating system in addition to the required libraries. One way out is to use container-based virtualization or containerization technology, which is an operating system level virtualization method to deploy distributed applications without starting an entire virtual machine for each application. As shown in Figure 1, containers do not depend on the hypervisor and the virtual machines. The server provides the shared instance of the operating systemd and each application stays in its own container.

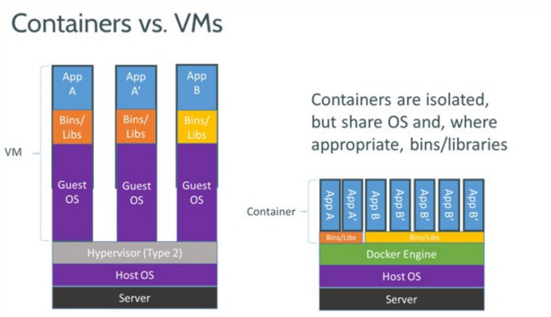


Fig. 1. Container vs virtual machine. Image credit: docker.com

In next several sections, we will discuss the containerization technology as well as the design and development of SIMULOCEAN system which manages and deploys scientific applications as containers.

II. Containerization with Docker

The concept of containerization/isolation has been around for more than a decade. In 2008, Linux Contains (LXC) was added to the kernel. LXC combined the use of kernel cgroups and namespaces to implement lightweight process isolation [4]. However, the idea of containerization was not widely adopted until Docker [5] was developed to highly facilitate the usage of the technologies. As shown in Figure 2, Docker, as a software container platform, provides a suite of software that creates, manages, and distributes applications. The Docker container itself is a complete file system with everything it needs to run: code, runtime, system tools, system libraries, etc.

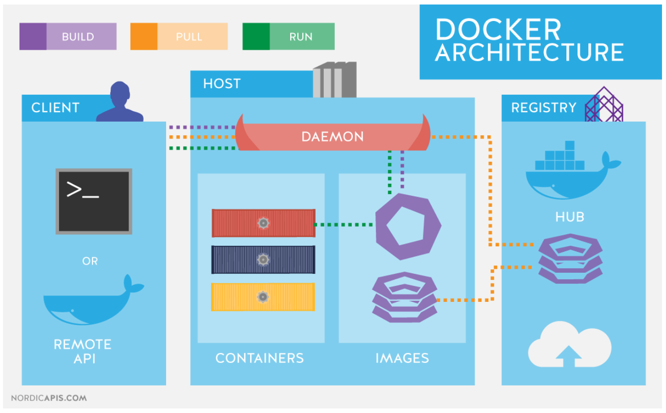


Fig. 2. Docker architecture. Image credit: nordicapis.com

For application developers, Docker enables developers to build, ship, and run distributed applications in self-contained environments. Docker enables executable applications to be quickly assembled from components then run by a user without the need to rebuild or satisfy any external dependencies. As a result, a Docker-enabled application can be reliably executed in a known operating system environment on any system that supports Docker containers.

III. SIMULOCEAN Science Gateway

Science gateways are web-based portals that allow researchers to perform computational science in domain-specific, community-developed interfaces. SIMULOCEAN is one of such portals for researchers to investigate water, air and land dynamics on HPC or cloud-ready systems.

The development of SIMULOCEAN is carried out by the Coastal Hazards Research Collaboratory (CHARCOL) [6] led by Dr. Q. Jim Chen at Louisiana State University (LSU). SIMULOCEAN started in the NSF-funded Northern Gulf Coastal Hazards Collaboratory (NGCHC) project, which used cyberinfrastructure to catalyze collaborative research and education and reduce risks to coastal. The SIMULOCEAN project continues as one of the major components of another NSF-funded Coastal Resilience Collaboratory (CRC) project to enable discoveries for sustainable deltaic coasts.

SIMULOCEAN science gateway (Figure 3) has three primary goals. The first is to increase community and collaboration in the interested field of study. The second is to provide scientists and engineers with easy-to-use computer tools for wetlands restoration and protection. And lastly, the third is to leverage and promote NSF-funded resources like XSEDE, which are provided to the greater science community. Focusing specifically on the "community and collaboration" goal, it is important to enhance the collaboration among earth scientists, computer scientists, cyberinfrastructure specialists

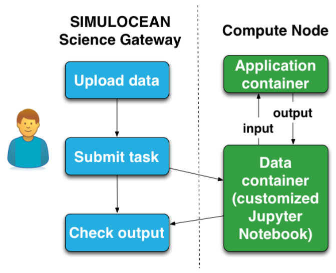


Fig. 3. A dual-container system is designed in SIMULOCEAN to separate the scientific application and data, which is hosted on a customized Jupyter Notebook to enable interface data analysis and visualization.

and coastal engineers tasked with solving the sustainability issues of deltaic coasts like those in Louisiana.

In SIMULOCEAN, ecologists and geologists (or any other type of scientists and engineers) who study the wetlands in Louisiana can use the computing facilities available to explore, and perform numerical simulations of various coastal events like hurricanes and storm surges on large swaths of land like never before. SIMULOCEANs easy-to-use interface can help inexperienced cyberinfrastructure users perform the tasks they need to.

IV. Implementation of SIMULOCEAN

SIMULOCEAN consists of 4 subsystems (Figure 4): (1) observation data assembling, (2) model distribution and deployment, (3) workflow management, and (4) numerical data interpretation and visualization.

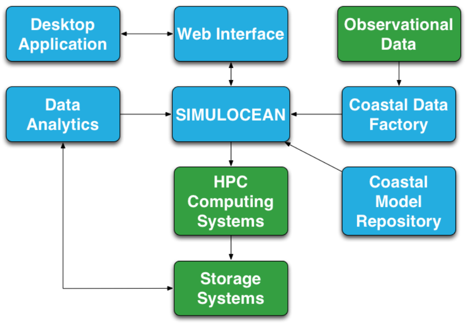


Fig. 4. SIMULOCEAN includes multiple components (as shown above in blue boxes) that are coupled together to provide user interfaces to prepare model input and run models on high performance computing systems.

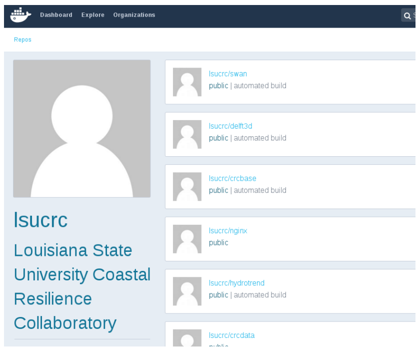
A. Data Assembling with Coastal Data Factory

Collecting and formatting data is a tedious but important task in coastal modeling. Despite many efforts on different levels to standardize presentation of observational data, there is still an expectation that modelers prepare any required input data on their own. The procedure could easily become complicated. For instance, data providers utilize one format for real-time data (typically offered in human readable ASCII format) and another for historical data (likely a compressed binary format). If both real-time and historical data are required, a modeler is expected to download, translate, and merge the data into the form required by their application. Such preprocessing overhead is made worse for coupled simulations where multiple input data sets are required. Further compounding the problem, different modeling applications require different input data formats. To put the problem in perspective, consider that N input data sources and M models would require $N \times M$ different configurations to handle all the potential cases. The preprocessing becomes increasingly burdensome and task management unwieldy as the values of N and M become large.

To solve this $N \times M$ problem, we implemented a data assembling framework named Coastal Data Factory. It consists of N data collectors, M translation adapters, and one local database where all the downloaded data are be held in a predefined format. This effectively isolates the data providers from the application users and allows data to be delivered to a particular model in 2 steps: 1) translate source data into common format for storage, and 2) translate common format into application specific format. This reduces the original $N \times M$ data handling complexity to a much more manageable set of $N + M$ data translation tools. Since only well-defined data is stored, adding new data sources or applications only requires the addition of 1 tool for each, be it a spider or an adapter. A prototype of such a Coastal Data Factory has been developed to provide input data for models managed by SIMULOCEAN.

B. Cloud-ready Coastal Model Repository

The Coastal Model Repository (CMR) is targeting such cloud and cloud-like architectures to enable quick deployment of coastal models and their working environments (Figure 5) [7]. CMR serves as a community repository for precompiled open source models that are widely used by coastal researchers. In addition to the source code, CMR will introduce the distribution of containerized coastal models, which can run on any cloud-like architecture directly, with negligible system overhead. CMR is currently hosted on Docker Hub [8], which is a cloud service offered by Docker. Docker Hub enables application sharing and automates workflows for building and shipping of applications as well. Containerized applications on Docker Hub could be automatically assembled from components, if there is any updates, then run by a user by downloading the updates without the need to rebuild or satisfy any external dependencies. As a result, the coastal models from CMR can be reliably executed in a known operating system environment on any system that supports Docker containers.



lsucrc

Louisiana State University Coastal Resilience Collaboratory

lsucrc/swan
public | automated build

lsucrc/delft3d
public | automated build

lsucrc/crcbase
public | automated build

lsucrc/nginx
public

lsucrc/hydrotrend
public | automated build

lsucrc/crcdata

Fig. 5. Hosted on Docker Hub, CMR serves as a community repository for precompiled open source models.

CMR provides and maintains a minimal operational system image to synchronize with upstream development by the open source community for bug fixes, performance improvement, and feature enhancement. CMR periodically checks community code repositories as well as the download page of each model hosted on CMR. Once a new release of a model is detected, CMR automatically downloads, compiles, and tests the model to update the repository. With the help of CMR, a coastal researcher can start running state-of-the-art models on the latest cloud-ready computing systems in minutes. Workflow management tools can take advantage of CMR to quickly deploy coastal models on academic and commercial cloud platforms while continuing their support on traditional HPC systems. With community involvement, we could build a software ecosystem for coastal models and beyond to serve the computational science and engineering community to speed up scientific discovery and decision-making process.

C. Workflow Management

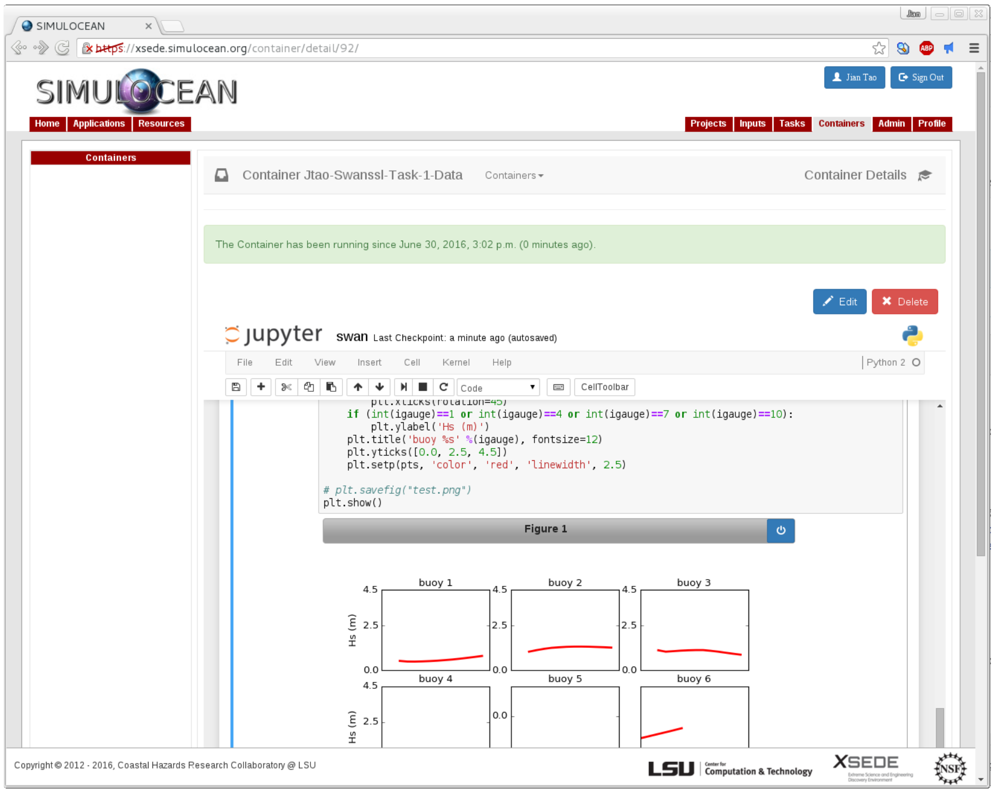
The modules and tools in SIMULOCEAN have been designed to handle different tasks that are crucial to successfully run coastal models and visualize simulation results. The tasks in SIMULOCEAN are managed by Celery [9], an asynchronous task queue/job queue based on distributed message passing. The Celery task queue enables SIMULOCEAN to run its tasks in the background with attached status labels, which are updated in real time by Celery to reflect the progress of each task. Celery requires a message transport/broker to send and receive messages. In SIMULOCEAN, we use RabbitMQ [10] as the primary message broker. RabbitMQ implements the Advanced Message Queuing Protocol (AMQP)[11], a binary, application layer protocol designed to efficiently support a wide variety of messaging applications and communication patterns. The Celery task queue equipped with RabbitMQ can process millions of tasks a minute, with sub-millisecond round-trip latency. The characteristic high throughput and low latency of the Celery task queue are helpful to minimize the system overhead. The web interface, mobile page viewer, and desktop client for SIMULOCEAN will be developed to users access to the computing and storage facilities via web browsers, handheld devices, and a feature-enhanced desktop application.

D. Data Interpretation and Visualization

The high-resolution, multi-physics simulations needed for real world scientific and engineering problems require numerical model generation of large-scale spatial-temporal data sets that require significant computational resources to process, visualize, and archive. In SIMULOCEAN, we designed and implemented a dual-container system to separate the scientific application and data, which is hosted on a customized Jupyter Notebook [12] container to enable interface data analysis and visualization (Figure 6). A Jupyter notebook is an interactive web application for writing and running code interactively and authoring notebook documents. For each supported model in CMR (Section IV-B), a Jupyter notebook document will be provided to help with data analysis and report generation. Such a document contains a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. The notebook could be created, shared, and enhanced by individual user of the model. It provides another mean to share expertise and knowledge among connected science and engineering communities.

V. Collaboration via SIMULOCEAN

In the design of SIMULOCEAN, we took a multi-layer approach to separate concerns at both the development and deployment stage of the project. We aimed at forging a strong interdisciplinary collaboration by standardizing a descriptive interpretation to ensure a smooth communication between experts in different domains. Based on SIMULOCEAN, a virtual organization (VO) could be established to focus on ongoing or planned science or engineering projects.



The Container has been running since June 30, 2016, 3:02 p.m. (0 minutes ago).

```
        plt.xticks(rotation=45)
    if (int(igauge)==1 or int(igauge)==4 or int(igauge)==7 or int(igauge)==10):
        plt.ylabel('Hs (m)')
    plt.title('buoy %s' %(igauge), fontsize=12)
    plt.yticks([0.0, 2.5, 4.5])
    plt.setp(pts, 'color', 'red', 'linewidth', 2.5)

# plt.savefig("test.png")
plt.show()
```

Copyright © 2012 - 2016, Coastal Hazards Research Collaboratory @ LSU

Fig. 6. A customized Jupyter container is created to provide an interactive interface for SIMULOCEAN users for data analysis and visualization.Users can modify the notebook while the application is still running. Changes to the notebook could be saved and exported for later usage.

SIMULOCEAN has a built-in permission grant system to control the access to data, applications, hardware facilities, and workflows at different levels for groups and users. It helps to assign different roles in a VO to support fine-grained control over various aspects of both small and large scale collaborations. As an open source project, SIMULOCEAN is built on top of many other open source projects. The SIMULOCEAN team endorses the open source and open science initiatives. We encourage our users to share their code and data whenever possible as an open and shared software environment becomes more and more important for successful collaborative research.

VI. Conclusion and Future Work

We presented the design and development of the SIMULOCEAN science gateway that takes advantage of the latest development of containerization and web programming technologies to support the management and deployment of scientific and engineering applications on traditional HPC and cloud computing systems. Serving as a platform to enable multidisciplinary collaboration, SIMULOCEAN helps to connect communities with the integration of simulation applications and computational tools. With the rapid growth in both the development and adoption of containerization and cloud computing technologies in academia, more and more tools like SIMULOCEAN will emerge to serve the computational science and engineering community. We are currently working towards a more generic platform to facilitate the packaging and deployment of more applications via SIMULOCEAN.

VII. Acknowledgments

The authors would like to thank our colleagues at LSU and Texas A&M for comments and discussions. This work is supported by NSF Award EPS-1010640, CCF-1539567, and IIS-1723292 and in part by HPC computing resources at LSU and Texas A&M High Performance Research Computing. This work also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575. We acknowledge the assistance and support of the CSDMS Integration Facility and XSEDE Extended Collaborative Support Service (ECSS) program.

References

[1] Enabling a new future for cloud computing. https://nsf.gov/news/news_summ.jsp?cntn_id=132377.

[2] Amazon Web Services, Google Cloud, and Microsoft Azure join NSFs Big Data Program. https://www.nsf.gov/news/news_summ.jsp?cntn_id=190830.

[3] John Paul Walters, Andrew J. Younge, Dong-In Kang, Ke Thia Yao, Mikyung Kang, Stephen P. Crago, and Geoffrey C. Fox. Gpu passthrough performance: A comparison of kvm, xen, vmware esxi, and lxc for cuda and opencl applications. In *2014 IEEE 7th International Conference on Cloud Computing*, 2014.

[4] Linux Containers. https://linuxcontainers.org/.

[5] Docker. https://www.docker.com/what-docker.

[6] Coastal Model Repository at Docker Hub. http://www.coastalhazards.org/.

[7] Coastal Model Repository at Docker Hub. https://hub.docker.com/u/lsucrc/.

[8] Docker Hub: a cloud-based registry service for Docker images. https://hub.docker.com/.

[9] Celery: Distributed Task Queue. http://www.celeryproject.org/.

[10] RabbitMQ: an open source message broker software. https://www.rabbitmq.com/.

[11] The Advanced Message Queuing Protocol (AMQP) for message-oriented middleware. https://www.amqp.org/.

[12] The Jupyter Notebook. http://jupyter.org/.