Application of Container Technology in Numerical Ocean Model: a Kind of High-performance ROMS Containerized Architecture

Wentai Cao, Ying Guo*, Hu Zhang, Minghui Sun and Meng Yuan
Qilu University of Technology (Shandong Academy of Sciences), Jinan, Shandong, 250000, China
Shandong Computer Science Center (National Supercomputer Center in Jinan), Jinan, Shandong, 250000, China
Shandong Provincial Key Laboratory of Computer Networks, Jinan, Shandong, 250000, China
*Corresponding author’s e-mail: guoying@sdas.org

Abstract. Numerical ocean models are an important means for humans to study ocean activities today. For decades, with the continuous improvement of supercomputer performance, the application of numerical ocean models in high-performance computing has also developed rapidly. The birth and progress of high-performance cloud computing provide an opportunity for the development of portability and reproducibility of the ocean model. This paper uses container technology to construct the working environment of the Regional Oceanic Modeling System (ROMS), and rapidly completes the calculation jobs of the ocean model on the high-performance cloud computing platform. At the same time, we compare and analyze the operating performance of traditional HPC VM clusters. The new ocean model system we constructed based on high-performance cloud computing can provide better portability and convenience without wasting performance on traditional HPC clusters. It provides a new tool for scientists to quickly build ocean model clusters suitable for different hardware configurations, and to perform numerical ocean model calculations more conveniently.

1. Introduction
The ocean, which occupies about 71% of the earth's surface area, contains 97% of the earth's water resources and the same abundant ocean mineral resources, marine biological resources, and marine power resources. From the perspective of natural science, the ocean plays an important role in buffering and regulating the climate of the earth and plays an important role in the water cycle, carbon cycle, and nitrogen cycle. Thanks to the appearance of computers, it provides a means for us to understand and further explore ocean activities. At the end of the 1960s, the numerical ocean models were born from the Geophysical Fluid Dynamics Laboratory (GFDL) of the United States to digitalize ocean research. And supplemented by the high performance of computer technology to improve the efficiency of research [1].

With the development of computer design and manufacturing level, in recent decades, High-performance computing (HPC), an important branch of computer science, has gradually occupied a non-negligible position in the field of scientific computing [2]. The organic combination of numerical ocean models and high-performance computing has become an important means of marine scientific research.
The Regional Ocean Modeling System (ROMS), as a kind of ocean model, contains a variety of accurate and effective physical and numerical algorithms, as well as multiple couplings applied to biogeochemical, bio-optical, sediment, and sea ice applications. The model has been widely used in scientific research circles. However, the deployment of ROMS in HPC clusters is relatively unfriendly to ocean scientific researchers who are not computer professionals due to the variety of environments and applications it relies on, and there is a certain degree of difficulty in operation and workload. At the same time, it is difficult to transplant and save the deployed environment in the leased HPC cluster, which increases the cost of scientific research in disguise and improves work efficiency.

Based on container virtualization technology and orchestration technology, this paper proposes an architecture that takes into account the portability, flexibility, and reproducibility of scientific computing. As a result, a regional ocean model system (ROMS) model based on containerization technology is constructed, which can be easily and quickly deployed in various popular public cloud or private cloud environments. And compared to the traditional ROMS use environment and the scheme of this article, the feasibility and reality of the application of containerization technology and ROMS are discussed.

2. Ocean Models

2.1. General Numerical Ocean Models
To meet the needs of humans to explore the ocean, the Ocean Prediction System provides predictions for the future success of the ocean environment from the scale of time and space and guides human activities in the ocean [3]. Since its birth in 1969, Numerical Ocean Models or ocean general circulation modeling (OGCM) has gradually attained a core position in the Ocean Prediction System and has developed multiple branches depending on the different needs of scientific research. Solve various ocean problems [4]. Nowadays according to their respective approaches to spatial discretion and vertical coordinate treatment, the types of ocean models include NCOM, MOM, POM, ROM, etc. [5].

2.2. Regional Ocean Modeling System in HPC
The Regional Ocean Modeling System (ROMS) is a free-surface, terrain-following, primitive equations ocean model, which is widely used by the scientific community. It uses vertical terrain tracking and horizontal curve coordinates. ROMS is a modular code compiled in F90/F95 and uses the Message Passing Interface (MPI) protocol for parallel processing. The entire input and output data structure of the model is carried out through NetCDF (Network Common Data Form) [6].

With the rapid development of High-Performance Computing (HPC), the scale and accuracy of ROMS have significantly improved. In the current popular HPC cluster workflow, ROMS deployment is very complicated [7]. Due to differences in the HPC cluster operating system version and hardware environment, the corresponding compiler and application version have to be adapted accordingly, which increases the cost of environment configuration. At the same time, due to the dependence of numerical models such as ROMS on computer architecture, executable programs compiled on a cluster cannot be shared with other users. Therefore, we need to find a way to solve the complicated and time-consuming problem of preparing a numerical model environment based on virtual machines or bare metal clusters on public or private clouds.

3. Ocean Model Containerization

3.1. Container Technology
As the amount of calculations that HPC needs to process continues to increase, new requirements are also placed on the stability and portability of the cluster. The advantages of container technology, such as simplified deployment, multi-environment support, and ease of migration, have gradually attracted attention in HPC [8]. There are sufficient research examples to prove that while container technology basically retains the performance of the original high-performance computing system, it has greatly enhanced the portability and compatibility of the system.
In addition, different container technologies have different usage scenarios. Singularity is a container technology developed by Lawrence Berkeley National Laboratory in the United States for large-scale, cross-node HPC and DL workloads [9]. It has advantages such as lightweight, rapid deployment, and convenient migration, and it has good performance in operating large-scale parallel computing [10-11]. Through the application of Singularity container technology in the high-performance computing industry and its comparison with traditional container technology [12-15]. We believe that Singularity, as a container tailored specifically for high-performance computing, can support our research.

3.2. Container Orchestration Technology

Container orchestration refers to the automation of container deployment, management, expansion, and networking. Through multiple comparisons, we chose Kubernetes, an open-source container orchestration system. The main reasons for choosing Kubernetes in this study are as follows:

- High market share, fast update, and active community: Kubernetes is an open-source container cluster management platform created by Google using the Go language [16]. By 2019, Kubernetes has achieved 77% of the market share and has become the de-facto standard in the container orchestration field. Therefore, the Kubernetes community is vibrant, which facilitates problem-solving.
- More suitable for large-scale scenarios: Compared with other container orchestration management tools, Kubernetes pays more attention to the use of large-scale cluster scenarios and is especially good at deploying applications in complex scenarios. Because Kubernetes has a large number of components with different functions, and it also supports custom interfaces for scheduling policies. And through the CRI interface can support a variety of container mixed management [17]. And it has better support for ocean model applications that often require massively parallel computing.
- Powerful automation mechanism and microservice management mechanism: Kubernetes supports a series of automated mechanisms such as automatic boxing, self-repair, automatic horizontal expansion, automatic release, and rollback, which greatly reduces the management cost of large-scale clusters and improves operational stability [18,19]. And Kubernetes modules are more detailed and more diverse than tools such as Mesos and Swarm. The modules are directly loosely coupled and easy to customize. This idea of micro-service management greatly reduces the difficulty and cost of later system operation and maintenance [20].

3.3. Construction of ROMS Containerized Model

Our study uses Kubernetes and Singularity containers to design and implement a runnable model of ocean mode. In short, the system environment required by ROMS is first made into a SIF (Singularity Image File) format image file and based on the characteristics of Singularity container technology, the image file can completely save the system state at the time of production and export it as local files for reuse. Then package the image file and upload it to each worker node of the Kubernetes cluster, and use the kubectl command to run the YAML file to deploy the image loaded with the compiled ROMS to the Kubernetes cluster. You can perform ROMS tasks in a Kubernetes cluster. Moreover, since the Singularity container image SIF file can be changed and saved at any time, we can modify only a few configuration items in the image file to realize the operation of other ROMS calculations or jobs with different resource configurations. The exported SIF file loaded with ROMS can also be deployed to run in a new Kubernetes cluster without reinstalling and compiling. The system architecture diagram is shown in Figure 1.

Firstly, we install the Singularity container environment in the Linux system (see the Singularity official website for the installation method: https://sylabs.io/guides/3.5/admin-guide/installation.html). The version selected for this article is Singularity 3.5.3. Obtain a standard Linux system image from the Singularity mirror library and save it locally in a sandbox (the sandbox is a unique mode of the Singularity container. The image can be stored in the file system of the Singularity container host in the form of a folder. All operations done will be saved). The system image we use is Ubuntu, version 16.04,
Figure 1. Architecture of ROMS Containerized System named "ubuntu-roms" sandbox file. Use the "singularity shell --writable ubuntu-roms/" command to enter the mirror. The applications and versions that need to be installed are shown on Table 1.

Table 1. List of required applications.

| Application    | Version | Function                                      |
|----------------|---------|-----------------------------------------------|
| gcc/gfortran   | 7.5.0   | Provide a compilation environment             |
| MPICH          | 3.3     | Provide parallel computing support            |
| JasPer         | 2.0.10  | Process JPEG images                           |
| libPNG         | 1.6.37  | Process PNG images                            |
| ZLIB           | 1.2.11  | Processing compressed data                    |
| NetCDF         | 4.6.3   | Process scientific calculation output files   |
| HDF5           | 1.10.5  | Store images and data                         |

After installing the required dependent environment, you can install the ROMS main program. This article chooses Coupled-Ocean-Atmosphere-Wave-Sediment Transport Modeling System, COAWST for short. It is a comprehensive modeling system that integrates a variety of scientific calculation numerical models such as ocean, atmosphere, waves, and sediment movement [21]. COAWST, through the function of the coupler MCT, can not only use a single type of modeling system such as ocean model ROMS and atmospheric model WRF for scientific calculations but also couple two or more models to provide a simulation environment closer to the real scene of nature. This article chooses to install COAWST in a containerized environment, rather than a separate ROMS, in order to provide more types of calculations in the future. Next, we need to install the COAWST main program on the Singularity image file where the dependent environment is installed and configured and configure and compile the ROMS calculation example. The required operation is shown in Figure 2.

After the compilation is complete, we get a SIF image file with the ROMS environment configured. Use the "singularity build" command to save the sandbox image "ubuntu-roms/" as a separate SIF image file "ubuntu-roms.sif". At this point, this image can directly run the compiled ROMS calculation example in any environment with a Singularity container and output the calculated data.
After that, we need to download and install singularity-CRI. Then modify the runtime interface in the "KUBELET_OPTS" item in the kubelet configuration file kubelet.conf, and restart the kubelet service.

At this point, we can put the ubuntu-roms.sif file into the file system of the Kubernetes cluster, and deploy it to the working node with singularity-CRI according to the general way of deploying Pod in the Kubernetes cluster.

### 4. Experiments

In order to evaluate the results, we implemented some experiments. The experiment content and results are as follows.

#### 4.1. Experimental Environment Preparation

We have prepared a Linux image for installing and compiling COAWST for this study. The image is Ubuntu 16.04 downloaded through DockerHub. The Singularity container also supports the installation of Docker format images. With the help of Docker's active ecology, it provides better compatibility. At the same time, we built a Kubernetes cluster with 5 nodes through the Quan-Cloud cloud computing platform of the Shandong Computer Science Center, one of which is the Master node and 4 is the Worker node. The environmental configuration of each node is shown in Table 2. And designed a series of experiments to verify the feasibility of this research design.

| Configuration         | Value                      |
|-----------------------|----------------------------|
| CPU                   | Intel Xeon E3-1220 (Sandy Bridge) |
| RAM                   | 4 GB                       |
| OS                    | Ubuntu 16.04.3 LTS         |
| Orchestration tool    | Kubernetes v1.16.0         |
| Container             | Singularity v3.5.3         |

#### 4.2. Portability Test

Container technology provides good portability for HPC. We installed the Ubuntu 18.04.1 operating system on our persona computer. The computer's CPU is Intel Core i7-8750H 2.20GHz, which is different from the hardware configuration compiled in the Singularity image file Ubuntu-roms.sif. We installed the Singularity container on this computer, and then use 'singularity run' command to directly run the ocean mode calculation example in the Ubuntu-roms.sif image through the local file system, and got accurate output results. This shows that the ocean mode compiled in our Ubuntu-roms.sif image file can not only run computing jobs in the same hardware environment as when compiling but also directly...
run computing jobs in a heterogeneous environment. There is no need to recompile, which saves a lot of time and reduces the possibility of experimental errors.

4.3. Flexibility Test
We uploaded Ubuntu-roms.sif to the Kubernetes cluster file system of the Quan-Cloud cloud computing platform. The image can be deployed to the working node where Singularity-CRI is installed through the YAML file. Container clusters managed and arranged through Kubernetes are easy to run in environments with different operating systems and different hardware configurations. We can effortlessly use various private or public server resources for ocean model jobs.

4.4. Performance Test
We used the Marsh Dynamics Test in COAWST for testing. In order to control the experiment, we deployed a virtual machine cluster with the same hardware configuration on the Quan-Cloud cloud computing platform and installed and compiled the same version of the COAWST environment in the Ubuntu system of the virtual machine. First, we chose 3000 timesteps, and compared a single virtual machine in a virtual machine cluster with a single Ubuntu-roms Pod in a container cluster managed by Kubernetes. The comparative experiment was repeated 12 times, and the experimental results are shown in Figure 3. It can be seen that the ocean mode Marsh Dynamics calculation using container technology is roughly the same as the efficiency in the Linux virtual machine of the same configuration. Comparing the results of 12 times, it can be seen that the average running time difference is only 2.85%, and the running time of the container cluster is more stable, and the standard deviation is smaller than the virtual machine cluster, as shown in Table 3. This can indicate that the container cluster managed by Kubernetes can have a more stable performance, which is conducive to the controllability of the experimental process. Similarly, the turbulent kinetic energy, potential vorticity energy, and other results calculated by the two clusters are exactly the same (because the same input file and compilation environment are used, only the ocean mode system is installed in the container or the virtual machine is different).

Table 3. Comparison of average and standard deviation.

| Cluster  | Average(s) | Standard Deviation |
|----------|------------|--------------------|
| VM       | 1082.169   | 23.6222            |
| Singularity | 1113.004   | 13.9468            |

Later, we used MPICH to compute Marsh Dynamics Test in parallel on multiple virtual cores. The timesteps value is 36000. We used 4, 8, 16, and 32 vcores for calculations, and the results are shown in Figure 4. We found that when the number of parallelisms is 32, the performance of the container cluster is the highest compared to the virtual machine, which is 2.09%. In the calculation of the number of parallels from 4 to 32, the average performance loss of the container cluster compared to the virtual machine cluster is 1.57%. Considering the other advantages brought by the portability and flexibility of the container cluster, the experimental results are basically within an acceptable range.

4.5. Results and Discussion
In this study, we used Singularity container technology and Kubernetes container orchestration technology suitable for HPC scenarios to greatly improve the portability, flexibility and reproducibility of the numerical model system. The Ubuntu-ROMS.sif image file we encapsulated can directly run jobs on any Kubernetes cluster configured with Singularity-CRI, even heterogeneous clusters with different hardware configurations. In the Kubernetes cluster cloud computing environment, we conducted a comparative experiment using a typical example of ROMS and a virtual machine cluster with the same hardware performance. In the experiment, we found that the results of a container-based cluster and a Linux virtual machine running a ROMS computing job under the same hardware configuration are almost the same. It proves that the ROMS based on Singularity container technology and Kubernetes container orchestration technology proposed in this paper is feasible.
5. Conclusions and Future Work

In this research, the Ocean Model System environment constructed based on Singularity container technology and container orchestration technology can better complete the calculation jobs required by the scientific numerical model. Its performance loss is low, even lower than the performance error between the same VM cluster running the same computing task multiple times. And it is fully suitable for meeting the performance requirements of ocean models. Moreover, Singularity container technology is more applicable to HPC, such as no need to always run the daemon process, more flexible packaging and transplantation capabilities, etc. Therefore, it has become the preferred container for this article. Supplemented by Kubernetes' easy management, resource monitoring, and other advanced features in the cloud environment. It can meet the diverse needs of scientific researchers from all walks of life. Similarly, from the perspective of scientific researchers engaged in scientific numerical computing, the new architecture of this study does not require users to have the knowledge and skills of traditional high-performance cluster construction, and it reduces the time and steps of computing environment construction and deployment, allowing natural Scientists to devote more energy to the research of their profession.

Therefore, we can deduce that the use of containerization technology and container orchestration to build a numerical model cluster can substantially solve the problems of difficult installation and deployment on traditional HPC clusters, difficulty transplanting, and poor repeatability. In this study, we only used a single ROMS calculation in the COAWST model. We will further study the multi-mode coupling in the COAWST. In the future, it will be expanded to more types of numerical models to provide better support for researchers to explore the unknown in scientific research.

Acknowledgments

Funding from the project ZR2019QF014 supported by Shandong Provincial Natural Science Foundation is gratefully acknowledged. And we wish to thank Quan-Cloud provides us with the computing resources needed for research.

References

[1] Tonani, M., Balmaseda, M., Bertino, L., Blockley, E., Brassington, G., Davidson, F., Drillet, Y., Hogan, P., Kuragano, T., Lee, T., Mehr, A., Paranathara, F., Tanajura, C., Wang, H. (2015). Status and future of global and regional ocean prediction systems. Journal of Operational Oceanography, 8, s201 - s220.
[2] Jin, Z., Lu, Z., Li, H., Chi, X., Sun, J. (2019). Origin of High Performance Computing —Current Status and Developments of Scientific Computing Applications. Bulletin of the Chinese Academy of Sciences (06), 625-639.

[3] Fang, C., Zhang, X., Yin J. (2013). Development status and trends of ocean forecasting system in the 21st Century. Marine Forecasts (04), 93-102

[4] Zheng, P., Song, J., Zhang, F., Bao, X. Common Instruction of Some Ogcm. Marine Forecasts (04), 108-120

[5] Introduction. (2007) Ocean-Modeling. https://www.ocean-modeling.org/index.php

[6] Regional Ocean Modeling System (ROMS). (2021) ROMS. http://www.myroms.org/index.php

[7] Bhaskaran, G., Gaurav, P. (2015). Optimizing Performance of ROMS on Intel Xeon Phi. ICCS.

[8] Telfer S. (2018) The State of HPC Containers. https://www.stackhpc.com/the-state-of-hpc-containers.html.

[9] Kurtzer, G.M., Sochat, V., Bauer, M.W. (2017). Singularity: Scientific containers for mobility of compute. PLoS ONE, 12.

[10] Le, E., Paz, D. (2017). Performance Analysis of Applications using Singularity Container on SDSC Comet. Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact.

[11] Wang, Y., Evans, R., Huang, L. (2019). Performant Container Support for HPC Applications. Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning).

[12] Veiga, V.S., Simon, M., Azab, A., Fernández, C., Muscianisi, G., Fiameni, G., Marocchi, S. (2019). Evaluation and Benchmarking of Singularity MPI containers on EU Research e-Infrastructure. 2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), 1-10.

[13] Younge, A., Pedretti, K., Grant, R.E., Brightwell, R. (2017). A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds. 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 74-81.

[14] Rudyy, O., Garcia-Gasulla, M., Mantovani, F., Santiago, A., Sirvent, R., Vázquez, M. (2019). Containers in HPC: A Scalability and Portability Study in Production Biological Simulations. 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 567-577.

[15] Jordan, B.R., Barrett, D., Burke, D., Jardin, P., Littrell, A., Monticciolo, P., Newey, M., Piou, J., Warner, K. (2019). Singularity for Machine Learning Applications - Analysis of Performance Impact. 2019 IEEE High Performance Extreme Computing Conference (HPEC), 1-6.

[16] Hightower, K., Burns, B., Beda, J. (2017). Kubernetes: Up and Running: Dive into the Future of Infrastructure. O'Reilly Media.

[17] Noel, B., Michelino, D., Velten, M., Trigazis, S. (2017). Integrating Containers in the CERN Private Cloud.

[18] Bernstein, D. (2014). Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing, 1, 81-84.

[19] Qu, C., Calheiros, R., Buyya, R. (2016). Auto-scaling Web Applications in Clouds: A Taxonomy and Survey. arXiv: Distributed, Parallel, and Cluster Computing.

[20] Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., Steinder, M. (2015). Performance Evaluation of Microservices Architectures Using Containers. 2015 IEEE 14th International Symposium on Network Computing and Applications, 27-34.

[21] Warner, J., Armstrong, B., He, R., Zambon, J.B. (2010). Development of a Coupled Ocean-Atmosphere-Wave-Sediment Transport (COAWST) Modeling System. Ocean Modelling, 35, 230-244.