International Journal of Computers and Applications

ISSN: 1206-212X (Print) 1925-7074 (Online) Journal homepage: http://www.tandfonline.com/loi/tjca20

A study of software pools for seismogeology-related software based on the Docker technique

ShuRen Liu, ChangNing Cai, QiWei Zhu & N. Arunkumar

To cite this article: ShuRen Liu, ChangNing Cai, QiWei Zhu & N. Arunkumar (2017): A study of software pools for seismogeology-related software based on the Docker technique, International Journal of Computers and Applications, DOI: 10.1080/1206212X.2017.1396429

To link to this article: https://doi.org/10.1080/1206212X.2017.1396429

Published online: 14 Nov 2017.

INTERNATIONAL JOURNAL OF COMPUTERS AND APPLICATIONS, 2017
https://doi.org/10.1080/1206212X.2017.1396429

Taylor & Francis
Taylor & Francis Group

A study of software pools for seismogeology-related software based on the Docker technique

ShuRen Liu[a], ChangNing Cai[a], QiWei Zhu[a] and N. Arunkumar[b]

[a]Northwest Branch of PetroChina Research Institute of Petroleum Exploration and Development, Lanzhou, China; [b]Department of Electronics and Instrumentation, Sastra University, Tanjore, India

ABSTRACT

In oil exploration data centers, the seismic exploration process employs multiple applications that are highly specific and replaceable. In addition, the various software versions depend on a particular version of the operating system. As the server purchase from different times, the operating system is also different, with the server scrapped, the newly purchased server operating system version is higher than the original server, resulting in the original application software that cannot be used further. To solve this problem, the software life cycle needs to be extended an independent application software resource pool with a hardware system that is required to be built. Herein, the Docker containerization technology has been used to construct seismic and geologic software pools in a virtualized manner. Furthermore, the hardware and software have been separated from each other, and the hardware resources have been allocated to each container using a scheduling algorithm. By runtime start speed comparison, the physical disk usage comparison test proves that the seismic software has the advantage of being lightweight through the container, and by the analysis of effect of application on conventional module, it proves that the software running in the container has a good effect and is available independence and concurrency. The analysis of the resource-scheduling effect of the scheduling algorithm works well. Thus, we confirmed that the proposed method is effective and has a good promotional value.

ARTICLE HISTORY
Received 28 August 2017
Accepted 8 September 2017

KEYWORDS
Seismic exploration; Docker; virtualization; software pool; MapReduce; K-means

1. Introduction

In the field of oil exploration, seismic exploration [1,2] is the most commonly used method. The process of seismic exploration consists of seismic data collection, processing, interpretation, and other stages of composition. The processing and interpretation of seismic data [3,4] both have their own software, which are highly specific though small.

Exploration data centers run different types of seismic geology software, each depending on the specific version of the operating system. When new servers are purchased, their operating system is often a more modern version than the operating systems running on the old servers. This difference in operating systems often means that the original application software, which ran on the older operating systems, can no longer be used. The traditional method to address this issue is to use a virtual machine, but most of the seismic geological interpretation software operates in an environment that is simple and targeted toward functional applications. The applications do not require complex deployment, and because virtual machines occupy more resources, their deployment is time-consuming. Thus, a virtual machine is not suitable for the software deployment applications that have different operating system platforms in exploration data centers (see Table 1).

As the fields of cloud computing and big data have developed [5–7], many new technologies have arisen in the direction of virtualization. One such technology is Docker containerization technology. Docker is a development platform for developing, packaging, and running a variety of application softwares [8,9] using the Linux operating system kernel LXC [10] (Linux containers). Docker comes with the ability to package the application and the overall production environment with mirroring to achieve lightweight, software-level resource isolation. Docker technology has developed rapidly and been used in applications such as cloud platforms, software development, and Web sites [11–13], to name a few, but it has not yet been used in the energy field.

This article uses Docker containers to package lightweight Linux applications belonging to different software

CONTACT ShuRen Liu 2104218508@qq.com

environment versions at the exploration data center. The use of the Docker containers separates the hardware and software, which allows applications from older servers to continue to run on new servers, thereby extending the lifetimes of the applications and thus reducing the cost of application development.

2. Software pools for seismogeology-related software based on Docker technique

2.1. Docker technical features

Virtual machines are operated by monitoring the underlying hardware resources in order to manage and support multiple instances of operating systems running at the same time. The Docker platform is container-based, lightweight virtualization technology that effectively divides the resources managed by a single operating system into isolated groups to better balance conflicting resource usage requirements between isolated groups [14].

Docker uses the client/server architecture with the Docker Daemon working as the server side to accept and handle client requests (e.g. to create and run a distribution container). Docker can also run on a machine through sockets or RESTful API communication.

2.2. Architecture of software pools for seismogeology-related software

In view of the above problems and considering the characteristics of the Docker platform, the design of the Docker-based seismic geology software pool to achieve hardware and software resource isolation is modeled in Figure 1.

The model includes the following modules:

(1) Hardware resource pool: mainly server-based with different purchase years, and the old and new equipments are coexisting.
(2) Operating system layer: It includes the exploration data center server with various versions

Table 1. Overall logic diagram Dof database management module HBase table.

		Ser_res_usage			Ins_demand_res			Con__res_usage			
Rowkey	TimeStamp	Cpu-usr	Mem (%)	…	Cpu-usr	Mem	…	Cpu-usr	Mem (%)	…	Ser_status
Tag1	20170719113401	40	20	…	22	2G	…	23	10	…	1
Tag2	20170719113505	50	30		22	2G		22	9		1
Tag3	20170719113609	68	35		22	2G		25	13		2

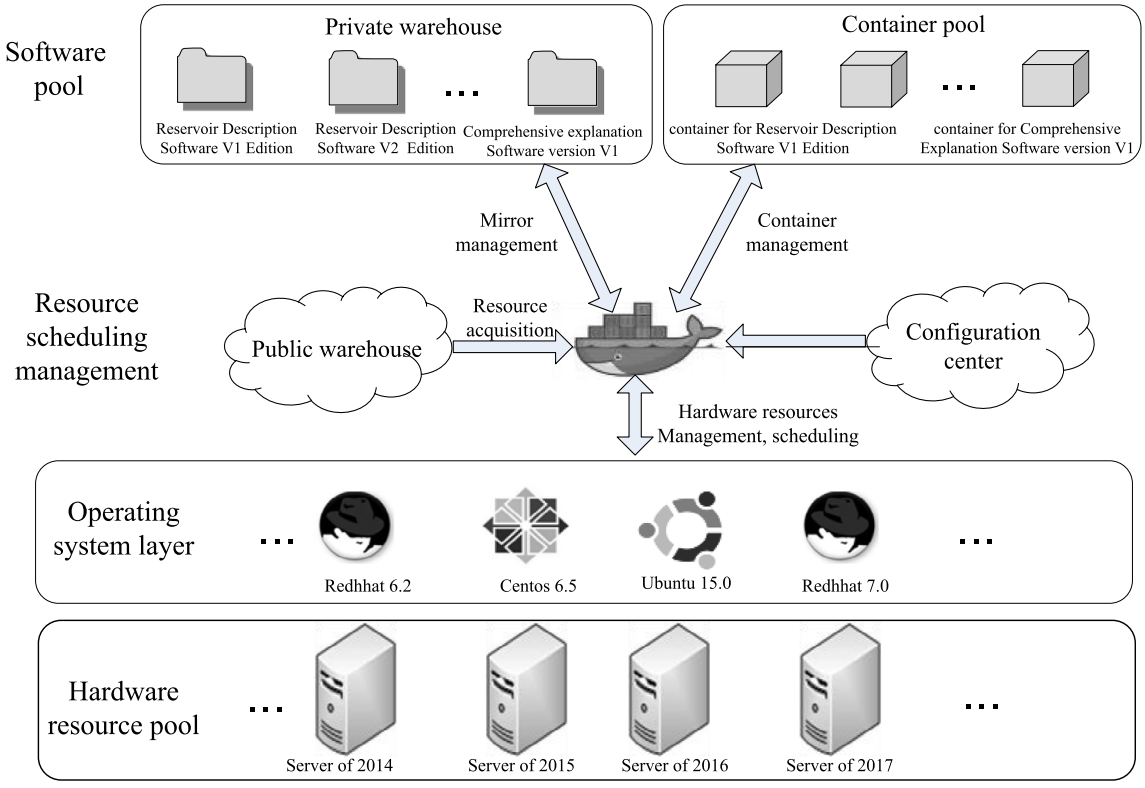

Figure 1. The architecture of the software pools for the seismogeology-related software.

of the operating systems (RedHat and Centos are the most common ones) and also the server with the operating system installed for which the server runs best. This layer can run the Docker container.

(3) Resource-scheduling management: This module is based on the Docker architecture. The module includes hardware resource management and scheduling modules and can access resources from the public warehouse to create the environment needed to run the seismic software and generate a mirror. The mirror is used to generate containers to run various types of software.

(4) Software pool: This module consists of two aspects in which the private warehouse saves each mirror. Each mirror preserves the best operating environment of the light seismogeology-related software. The software is finally run in a container that is instantiated by mirroring, and the container pool has the independence and integrity of the environment.

2.3. Resource-scheduling management

The objective of the resource-scheduling management is to manage the hardware resources and scheduling, mirror generation and management, container operation and management, and more. The resource management model for the whole model at the core position is shown in Figure 2. The resource-scheduling model includes the following:

(1) The server status monitoring module, which uses a Linux Agent for each server configuration resident process, continuously obtains the current consumption of each server's resources, resource features including memory, CPU, network, disk, and graphics. This information is fed back to the resource analysis engine for resource allocation and stored in the state database for subsequent analysis.

(2) The database management module uses the NoSQL database HBase to save the server resource usage information obtained from the server-monitoring module for each time period of the server (see Table 1). The saved information includes the time stamp, the hardware resources needed for each mirror instance, the resources allocated to each container (obtained from the container allocation module), the mirror update, container start, container stop, and other event records.

The database mainly includes three columns: Ser_res_usage indicates the server resource

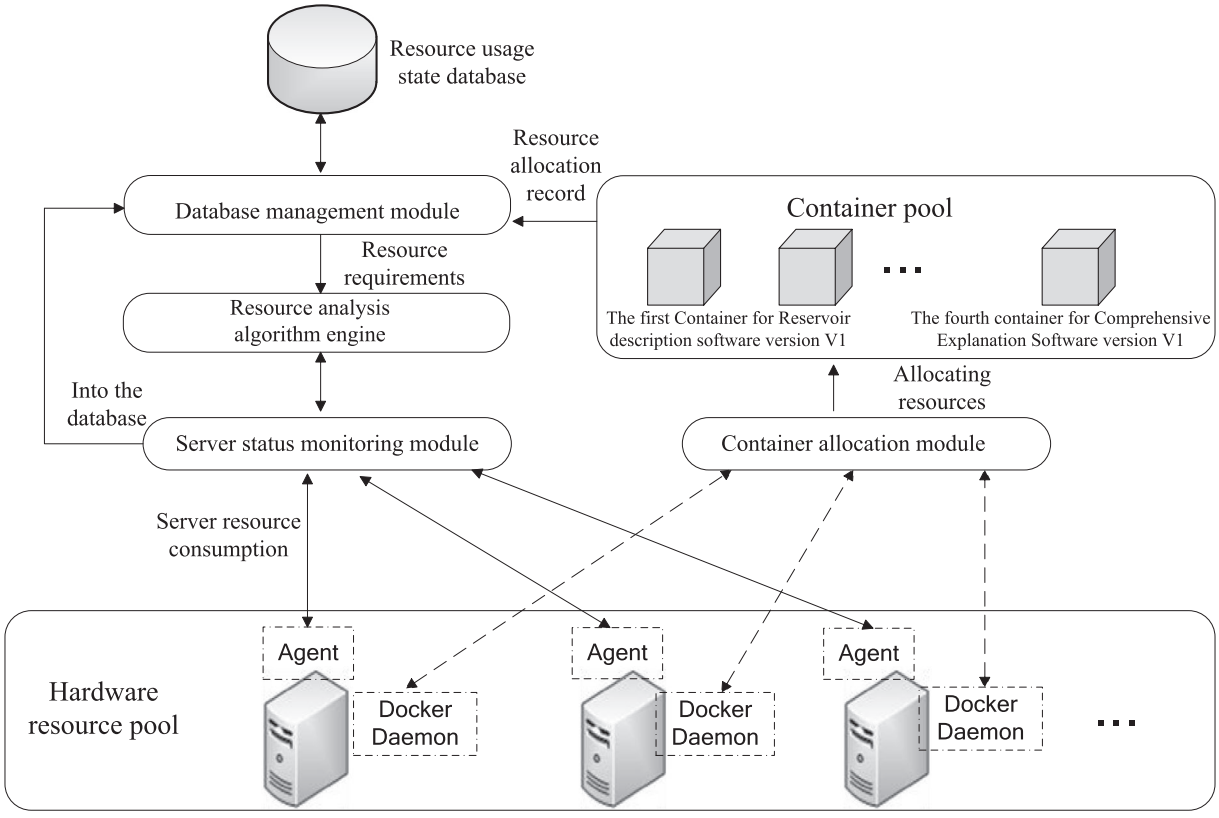

Figure 2. Resource management scheduling model.

usage in terms of the CPU, the memory, and the network resource information; Ins_demand_res indicates the instantiate demand resources; and Con__res_usage indicates the container usage resources. Ser_status represents the server state, which is calculated from the resource analysis engine and is the primary reference parameter during container allocation.

(3) The container allocation module manages and dispatches the entire life cycle (generation, operation, and shutdown) of the container, accepts the data from the resource analysis engine, and determines whether the container is scheduled to run on the specific server, and the resource allocation is fed back to the database management module.

(4) The resource analysis engine analyzes the resource consumption of each server, the resource requirements instantiated by a mirror, and implements the K-means algorithm, which has been studied in other papers, in MapReduce. The server uses the K-means method to find the best dispensable container and feeds back the specific allocation strategy to the container allocation module to realize the load balancing of the resource usage. At the same time, if the server is overloaded or a fault is detected, an alarm is issued.

The K-means algorithm [15] uses distance as the similarity evaluation index and outputs k cluster centers (Figure 3). The steps to implement the process are described below:

(1) K data centers are selected from the data set.
(2) All of the data are used to measure the distance between each center to find the minimum distance, which is included in the minimum class.
(3) All center types are recalculated. Steps 2 and 3 are repeated until the threshold is met. The main function for an appropriate threshold design is to use an iterative process to achieve the Map and Reduce functions and to continue to call the function until the threshold is met.

The separation of the hardware resources and software operating environment can be achieved via resource management. To maximize the performance of a newly purchased server, the operating system that matches the server best is installed on it. Then, the Docker and related components of the monitoring module on the server are deployed and incorporated into the hardware resource pool before the subsequent container deployed. By using the best mirroring environment (determined by the operating system version, various package dependencies, etc.) and running the software, it is possible to instantiate (generate the container) and deploy a large number of homogeneous software runtime environments on the hardware resources to achieve a multi-user software with independent use.

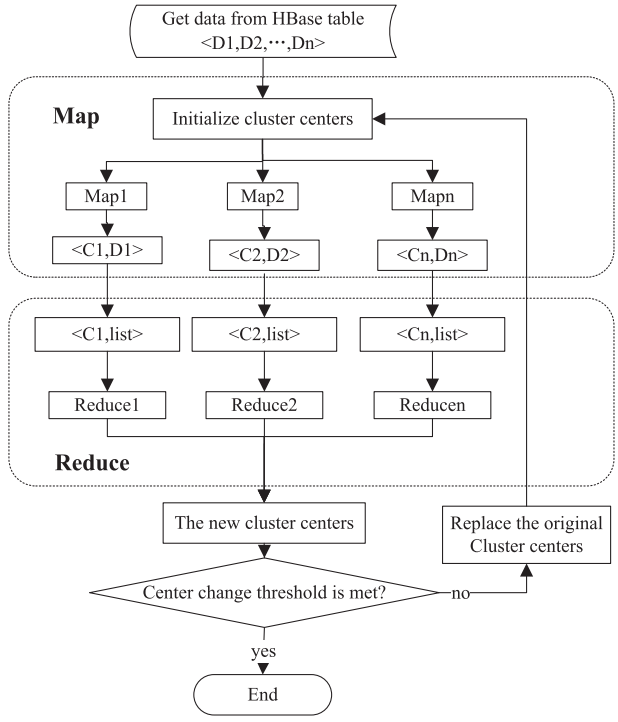

Figure 3. The implementation process of the K-means algorithm in MapReduce.

3. Results and analysis

3.1. Laboratory environment

Experiments were conducted to solve the relevant seismic geology software on the operating system requirements. The experimental configuration is shown in Table 2. The experiment used a virtual operating system with basic mirrors for RedHat 5.5 and 5.9 as the container. The configuration of the operating environment required more steps with the automatic build container than with the Dockerfile.

Because the operation of the seismic geology software module is directly related to the hardware environment (i.e. server configuration, network, and storage), the size of the project area, and the size of the data body, to name a few, the test data varied greatly. In the test, we attempted to ensure that the other experimental conditions remained the same. The tested software does not list the specific name of the application, only the application category, so as to avoid misunderstanding.

3.2. Analysis of experimental results

(1) Runtime start speed comparison

The container start-up speed is tested in the same software running environment (i.e. the same software package installation and the same service package). The test data comparing the physical machine, virtual machine, and Docker container performance are shown in Figure 4. As can be seen from the figure, the container start-up speed is significantly faster than the speeds for the physical and virtual machines, which shows the simplicity of using containers to build a viable environment.

(2) Physical disk usage comparison

The usage of the physical disk for several common seismic geological applications in the basic environment is given in Table 3. As can be seen from the figure, the container has a very clear advantage in terms of the physical disk usage.

3.3. Analysis of Effect of Application on Conventional Module

(1) Software operating efficiency comparison

Figure 5 shows a comprehensive interpretation of the time consumed by the software in different situations given a set area for operation (viz., 200 GB). Because the data storage device, network speed, and other factors have an impact on the module running time, the test data were stored in the network-attached storage (NAS) and the storage network connection line was the same.

The size of the two-dimensional section was 500 M, and the number of earthquakes was 895. The size of the three-dimensional section was 820 M, and the number of earthquakes was 600. The size of the data was 10G, and the number of earthquakes was 612. The load data size was 30G.

As can be seen from the figure, the container performs slightly worse than the physical machine in loading data. The performance of the other modules has a better upgrade over the physical machine and virtual machine.

(2) Performance of concurrent multi-containers

Some applications have a requirement that a given user can only start the software once at any given time. This problem can be solved by initiating multiple containers on the same physical machine (referred to as container concurrency). A comparison of the performance for applications started on the same device is given in Figure 6. The experiment used the same physical machine with 500 GB of working space to start the software several times with the same number of containers and then open a three-dimensional profile.

Table 2. Experimental environment configuration.

CPU	Memory	Hard disk	Operating system	Container operating system	Docker version
16 cores	32G	1T	Centos 7.1	Redhat 5.9/5.5	1.8.3

Table 3. Comparison of disk usage.

	Physical machine(GB)	Virtual machine(GB)	Container (GB)	Software installation package(GB)
Reservoir description	12	1.5	0.8	0.5
Comprehensive interpretation	18	2	1.1	0.8
Seismic interpretation	25	6	2	1.2
Logging interpretation	15	1.96	1.2	0.86

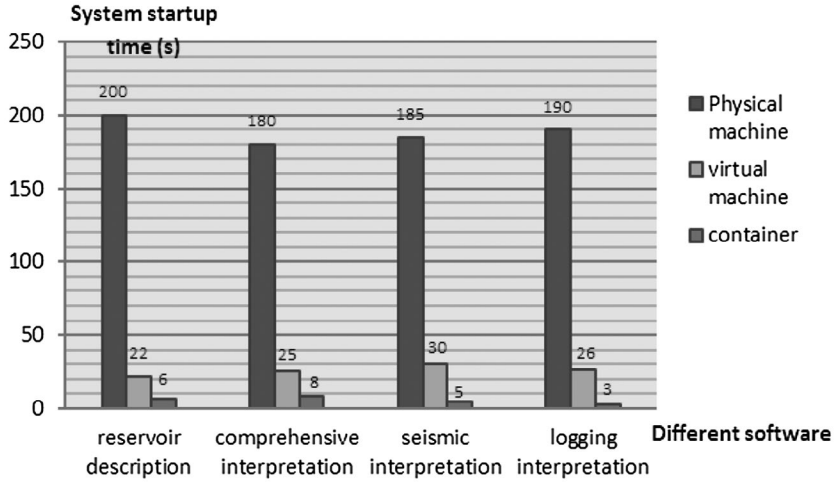

Figure 4. Runtime start-up speed.

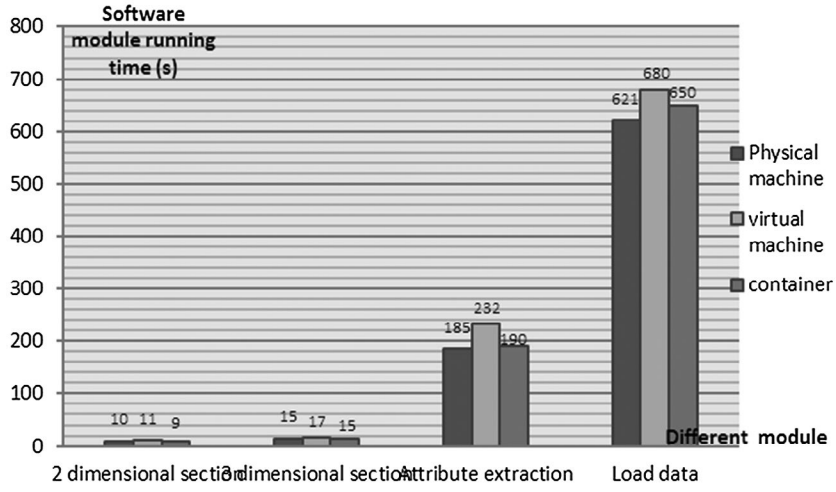

Figure 5. Comparison of operating efficiency of comprehensive interpretation software.

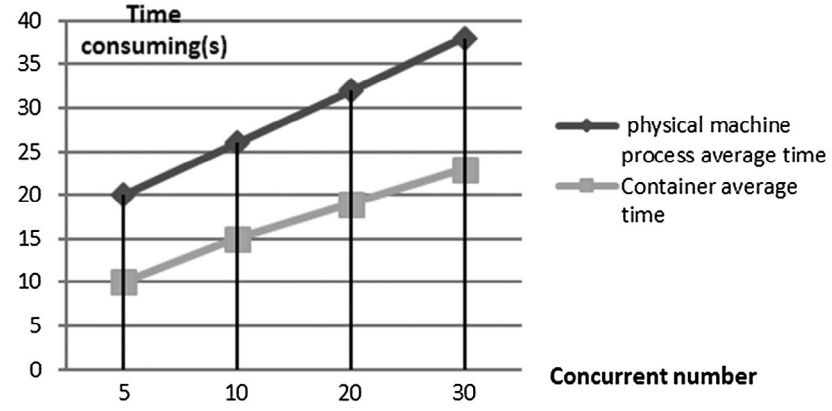

Figure 6. Comparison of container concurrency performance.

Table 4. Analysis of resource-scheduling effect.

	Level 1 idle server	Level 2 idle server	Level 1 busy server	Level 2 busy server	Faulty server
Judgment accuracy	65	62	75	78	60

As can be seen from the figure, the container works well, mainly because each container is independent of the others, so there are no resource contention phenomena. Thus, the container is effective in the case of concurrent applications.

3.4. Analysis of resource-scheduling effect

The classification status of the server is compared with the actual state via the K-means method to classify the state of a certain server. The resulting judgments of the different types of servers are shown in Table 4 for a data analysis with 500 samples.

In the HBase database, Ser_status is divided into five types of tables, from left to right, indicating the business of the server. The table shows that the algorithm is more effective on a busy server, which is related to the classification attributes we choose (such as CPU of memory), to improve efficiency. This will be the focus of future optimizations.

In summary, we have presented a system based on the Docker platform that demonstrates many advantages in terms of the start speed, resource usage, software running independence, concurrency, and other aspects. Additionally, the conventional application module is smooth, the actual operation is good for running the application, and the system is independent of the operating environment.

Finally, through the resource management scheduling, the model has demonstrated good results for ensuring the on-demand allocation of resources.

4. Conclusions

The life cycle of the hardware and software resources for the exploration and data center is inconsistent, because once the hardware is phased out, the corresponding software can no longer be used. To solve this problem and extend the software life cycle, this paper uses Docker containerization technology to package the application environment of different application versions for the exploration data center. The Docker container can then be used to build the seismic and geologic software pool and separate the hardware and software. With the presented method, the application software can continue to be used on the new server, and the server can have the best operating system installed, to extend the software lifetime and thereby reduce the cost of software development investment.

As demonstrated in the relevant tests, the Docker-based, seismic geology software pool solution can solve the above problems to separate the hardware resources and the software operating environment, to ensure the independent operation of the software and to provide rapid deployment and resource scheduling on demand with a better promotion value.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

ShuRen Liu is currently an engineer and researcher in the direction of oil exploration. He graduated from the master's degree in 2013, specializing in computer applications. Since the graduate stage, his research direction was mainly in high-performance computing and storage, and published two EI papers.

Changning Cai is currently a senior engineer and researcher; he is committed to high-performance computing in oil exploration for 30 years. His research direction is high-performance computing network performance, and he published three EI papers.

Qiwei Zhu is a senior engineer and researcher. In 2000, he graduated from the master's degree; his postgraduate research topic is a large-scale computer–machine research in the field of petroleum exploration. After working, he was devoted to the operation and maintenance of oil exploration data center. Currently, he focused on cloud computing.

N Arunkumar is working as an assistant professor in Department of Electronics and Instrumentation, SASTRA University. His research interests include Artificial Intelligence, Machine Learning, and Biomedical Signal Processing.

ORCID

ShuRen Liu http://orcid.org/0000-0001-6445-6670
ChangNing Cai http://orcid.org/0000-0003-4166-7381
QiWei Zhu http://orcid.org/0000-0003-1592-4970

References

[1] Stewart RR, Gaiser JE, Brown RJ, et al. Converted-wave seismic exploration: methods. Geophysics. 2002;67(5):1348–1363.
[2] Weglein AB, Araújo FV, Carvalho PM, et al. Topical Review: inverse scattering series and seismic exploration. Inverse Probl. 2003;19(6):R27.
[3] Robertsson JOA, Rickett J, et al. Seismic data processing. US, US 20080008039 A1. 2008.
[4] Mendel JM. White-noise estimators for seismic data processing in oil exploration. IEEE Trans Autom Control. 1977;22(5):694–706.
[5] Nist SP. A NIST definition of Cloud computing. Commun Acm. 2015;53(6):50–50.
[6] Hashem IAT, Yaqoob I, Anuar NB, et al. The rise of "big data" on cloud computing: Review and open research issues. Inf Syst. 2015. 47:98–115.
[7] Klein D, Tran-Gia P, Hartmann M. Big data. Inform-Spektrum. 2013;36(3):319–323.
[8] Anderson C. Docker [software engineering]. IEEE Softw. 2015;32(3):102-c3.
[9] Peinl R, Holzschuher F, Pfitzer F. Docker cluster management for the cloud - survey results and own solution. New York (NY): Springer-Verlag New York, Inc.; 2016.
[10] Bernstein D. Containers and cloud: from LXC to Docker to Kubernetes. IEEE Cloud Comput. 2015;1(3):81–84.
[11] Pahl C, Felter W, Ferreira A, Rajamony R, et al. Containerization and the PaaS cloud. IEEE Cloud Comput. 2015;2(3):24–31.
[12] Felter W, et al. An updated performance comparison of virtual machines and Linux containers. Lect Notes Comput Sci. 2014;1140:438–453.
[13] Pandey RV, Pabinger S, Kriegner A, et al. DaMold: a data-mining platform for variant annotation and visualization in molecular diagnostics research. Hum Mutat. 2017;38(7):778–787.
[14] Leprevost FD, Gruning BA, Alvers AS, et al. BioContainers: an open-source and community-driven framework for software standardization. Bioinformatics. 2017;33(16):2580–2582.
[15] Al-Zoubi MB, Hudaib A, Huneiti A, et al. New efficient strategy to accelerate k-means clustering algorithm. Am J Appl Sci. 2008;5(9):1247–1250.