Retraction

Retraction: Computer Software Design Based on Cloud Platform High-Performance Computing *(J. Phys.: Conf. Ser. 1915 032005)*

Published 9 September 2022

This article has been retracted by IOP Publishing following an allegation that raises concerns this article may have been created, manipulated, and/or sold by a commercial entity. In addition, IOP Publishing has seen no evidence that reliable peer review was conducted on this article, despite the clear standards expected of and communicated to conference organisers.

The authors of the article have been given opportunity to present evidence that they were the original and genuine creators of the work, however at the time of publication of this notice, IOP Publishing has not received any response. IOP Publishing has analysed the article and agrees there are enough indicators to cause serious doubts over the legitimacy of the work and agree this article should be retracted. The authors are encouraged to contact IOP Publishing Limited if they have any comments on this retraction.

Retraction published: 9 September 2022
Computer Software Design Based on Cloud Platform High-Performance Computing

Ning Ding¹*

¹Office of Academic Affairs, Shenyang Polytechnic College, Shenyang, Liaoning, 110045, China

*Corresponding author e-mail: dingning2389@vtcsy.com

Abstract. In modern computer software research, high-performance computing technology has been more and more widely used. In software design, a large number of computing requirements and tasks need to be completed with the help of supercomputers and corresponding computing environments. At the same time, the rapid development of modern high-performance computers provides strong support for improving software design research.

Keywords: High-Performance Computing Applications, Computer Environment, Software Platform Design

1. Introduction

With the rapid development of computer technology and network technology, high-performance computing has gradually become an important means of technological innovation. It is widely used in all walks of life. However, the application of high-performance computing in computational fluid dynamics has some problems, especially for some large-scale complex problems, cutting-edge problems and scientific problems, it is necessary to write code for numerical simulation.

2. Process and architecture

2.1. Operation process

Different applications face different fields and face scientific problems, so the quantity and form of pre-processing, parameters and processing results are different, but after abstraction, the entire processing flow can be summarized as shown in Figure 1, including setting parameters or uploading Input file, input file pre-processing, execution calculation, calculation result post-processing and visualization four steps. Except for the computing tasks performed in the high-performance computing environment, the
remaining steps can be individually customized for their presentation form and operation logic, or the pre-processing or post-processing steps can be omitted [1].

![Diagram of overall service flow chart]

**Figure 1.** Overall service flow chart

The overall processing flow of the platform is completed under the cooperation and interaction of the browser, front-end server, and high-performance computing environment. The browser is mainly responsible for interacting with users, operating parameter settings, task queue selection, task status feedback, historical data management, and result visualization. It is a user-oriented interface. The high-performance computing environment mainly provides computing resources and performs computing tasks. The front-end server is a bridge connecting user operations and high-performance computing environments. It is responsible for process organization, job parameter checking, job status monitoring, and notification message sending.

2.2. Architecture design

The software platform solution is based on the Docker container solution deployed on the LINUX system. The entire system architecture is shown in Figure 2, which is divided into two parts: a front-end server and a back-end computing node. The user accesses the front-end server through a browser to obtain various services uniformly [2]. The front-end server uses the high-performance computing environment API to operate computing nodes, submit and query tasks, etc., and the back-end computing nodes connect to the high-performance computing environment.
Figure 2. The overall architecture of a software platform based on high-performance computing

The front-end server uses a set of Docker containers to provide services. The MySQL and MongoDB containers are responsible for data storage, and the notification service container is responsible for asynchronously sending job status information to users. The application container is shown in Figure 3, which contains the specific business logic of the platform, such as the implementation process of each computing application, the implementation of a general management background, and a public class library. The containers are logically independent of each other, and they can be physically deployed on different servers or on one server. The containers are called through interfaces [3].

Figure 3. Application module structure

3. Design points

3.1. General architecture design and differentiated customization solutions for different computing services

Different computing software has different functions and calculation methods, the operating parameters of the application and the type and quantity of input and output files are different. Therefore, it is
necessary to abstract and encapsulate the application and computing tasks uniformly so that all applications and computing tasks support unified Operation.

On the other hand, because different application output files are different, the result data that needs to be displayed and used are different, and the visualization and post-processing of the result data are even more different. It is necessary to meet the customized input and output of different applications on the basis of unified abstraction [4]. The calculation results are displayed and satisfy other subsequent processing, such as data visualization and other operations. In the design of the general platform, the above problems are solved through the following solutions.

1) Abstract integration of public operating interfaces, modular loading of the front end. For application queue selection, input file upload, input parameter receiving and other frequently used functions of various applications, it is not necessary to implement each application separately. A set of common interface methods are encapsulated in the back-end, and the front-end is called asynchronously through Ajax. The front-end code logic is packaged as asynchronously loaded components, which are configured and loaded separately.

2) Use a document database, because the parameter types, quantities, temporary files, and result file formats of different applications are very different, and the applications have the same attributes such as creation time, run queue, and run time. If you use a relational database It is necessary to create a basic information table, and then create a related storage table structure for each application. It also needs to be processed separately when inquiring, and the operation is cumbersome. Non-relational databases (NoSQL) greatly improve product performance and scalability by reducing functions that are rarely used in traditional relational databases. MongoDB is a database based on distributed file storage, written in C++ language, designed to provide scalable high-performance data storage solutions for Web applications. MongoDB is a database between relational and non-relational databases. Among non-relational databases, it has the most functions and resembles relational databases. The data structure it supports is very loose, and it is a bson format similar to json, so it can store more complex data types, greatly simplify storage complexity and improve storage efficiency [5].

3) Use object inheritance to design code logic and inherited ODM. Because all computing tasks share some attributes and have specific task parameters and information on this basis, it is very consistent with the object inheritance model, that is, each task inherits from the base class TaskBase, and the task creation time and completion time are defined on TaskBase, Running status and other information, and then each different task is derived from the base class, and the corresponding parameters and information are added. In this way, in terms of code organization, it is convenient to perform unified operations on all tasks, and to operate a special task individually. At the same time, based on the use of the document database MongoDB, the system's database operations are completed indirectly through ODM supported by the Mongoose class library. The Mongoose class library also supports inherited database model organization, which is convenient for the development of task models.

4) Use MVC to organize code logic. MVC is the abbreviation of model-view-controller. It is a model of software design. It organizes code by separating business logic, data, and interface display, and gathers business logic into one component. It does not need to rewrite business logic while improving and customizing the interface and user interaction. The server-side design of the general platform uses MVC to organize the code structure to reduce coupling and facilitate reconstruction.
3.2. Unified basic service design

In the design and implementation of the general software platform, abstraction separation includes two levels: encapsulating general class libraries and creating independent service mirrors. Encapsulating general class libraries is to independently create and publish general class libraries in the form of components. For example, the operation class library for high-performance computing environment is packaged as scgrid package, and compression and decompression operations and other general operations are packaged in utilities package. Independently packaged and released component libraries can be directly referenced by other components and modules, which is also easy to maintain and upgrade [6]. Creating an independent service mirror is to publish relatively independent, low-coupled modules through the Docker mirror, and expose the HTTP call interface to interact with other services. For example, the notification service is a separate HTTP service, which contains message queues and mail sending. The logic of database operations, application services add notifications and query status to the queue by calling the interface.

3.3. Good user experience

A good product is not only "available", that is, meeting functional requirements, but also "easy to use", such as the simplicity and convenience of the process, the intuitive and beautiful interface, and the easy accessibility in the form of use. When applications and services are built on a good user experience, the platform will be truly user-friendly, attract more users and promote the application and development of high-performance computing.

In the design and implementation of the universal platform, the user experience is enhanced by using the latest HTML5 and CSS3 features and beautifying the interface with related class libraries. For example, read and upload files asynchronously through the HTML5 File API, beautify the layout by referencing the Bootstrap library, optimize the display of mobile pages, and draw beautiful statistical charts and visualization effects by referencing Echarts, D3.js, etc.

3.4. Easy to deploy new applications

As a general management platform, it is necessary to consider how new applications will be integrated and added in the future. In addition, the current form of software development and release has become a fast update and iterative model. Fast feedback and modification will continuously improve the quality and experience of the application [7]. Therefore, scalability and convenience must be considered when designing. Update the issue of iteration. In the design and implementation of the universal platform, the code is organized through modularization and MVC to facilitate the addition of new applications. At the application layer, when a new application is added to the platform, there is no need to consider how the job is submitted to the computing environment, how to monitor the job status, how to notify the user of the job information and other underlying operations, just write the application submission parameter page according to the template. Yes, the components such as queue selection and file upload can be used directly. Applications that do not need post-processing can directly use the general results display page to obtain the calculation results. If you need post-processing and visualization of the calculation results, you only need to add the corresponding processing logic to the workflow. Realize that the application can be added conveniently only by customizing the input and output logic in the workflow. In the process of platform development and deployment, build tools are used to cooperate with Docker and related
services to achieve continuous integration and deployment iteration. Use the build tool Grunt to cooperate with the configuration script to organize the test environment and compilation and release of the code. The local environment variables and test database are referenced in the test environment to facilitate local debugging. The online environment variables are used when publishing, the debugging information is removed, the official environment database is used, and the compression is merged. Front-end static files to optimize performance. When the compiled code is pushed to the release branch, it will trigger the image building service of Docker Hub to automatically build the relevant Docker image. Then the local server will automatically pull the built image through the built message hook, update the container running the service, and realize the service Automatic update and deployment [8].

3.5. System and service security

With the continuous improvement of computer system functions and the continuous improvement of speed, the composition of the system is becoming more and more complex, and the scale of the system is getting larger and larger, especially the rapid development of the Internet, the unprecedented expansion of software scale, and the security problems of computer systems and services are becoming more and more serious. A qualified software service has to consider security issues and minimize and avoid security risks. Common platform services are released and deployed through Docker. Due to the characteristics of containers, security can be summarized in two points: no impact on the host and no impact on other containers. The services in each container run independently, and the crash of a service in a container will not affect the services of the host and other containers. At the same time, Docker opens and exposes service ports through port mapping and binding, and can only open and expose services on ports 80 and 443 that provide external services. For internal service ports, such as database ports, reference through links, there is no need to bind to external ports, which is further enhanced safety [9,10].

4. Conclusion

This paper proposes a software platform based on a high-performance computing environment, that is, we need to provide users with a more convenient and user-friendly experience through the high-performance computing environment, so as to improve the functional design and use effect of the software platform.

References

[1] Yuan Haigen, Li Hongli. Design and application of software test platform based on cloud computing [J]. Journal of Shanxi Energy Institute, 2018:135-137.

[2] Cao Zheng. Design and Application of Software Testing Platform Based on Cloud Computing [J]. Computer Products and Circulation, 2018:28.

[3] Hu Yilin, Wei Dongdong, Ye Zhu. Design and Application of Arrow-borne Computer Software Testing Platform [J]. Computer Knowledge and Technology: Academic Exchange, 2018:66-68.

[4] LI Jianhua. Virtual Design of Mechanical Properties Using Computational Software [J]. Decision and Information: Late Edition, 2012:230-230.
[5] Zhao Wen. Application Research of Cloud Software Platform in Computer Room [J]. Journal of Chifeng University (Natural Science Edition), 2013: 35-36.

[6] Chen Xinyu, Liu Qian, Zhang Baohua, Jin Zhong. Design and Implementation of Scientific Computing Application Platform Based on High Performance Computing Environment [J]. Technology and Application of Scientific Research Informatization, 2016.

[7] Sun Yinhang, Chen Zhe, Wang Yu, Fan Dejun, Yang Xuanlin. Application of IBM Platform High Performance Computing Platform in Research Cloud [J]. China Science and Technology Expo, 2015: 217.

[8] Lin You, Yu Yang. Design and Application of High Performance Embedded Computing Hardware Platform with Multiple FPGA [C]. 2016.

[9] Wang Qiaozhen. Design and Application of Network Teaching Platform Based on Computer Software Development Technology [J]. Information Systems Engineering, 2018: 170.

[10] Su Hengyang. Discussion on the Application of Virtual Host Platform in High Performance Computer [J]. Computer Knowledge and Technology, 2010: 215-216.