applied sciences

MDPI

*Article*

# Automatically Expanding User-Management System for Massive Users in the Cloud Platform

**Shengyang Li [1], Zhen Wang [2] and Wanfeng Zhang [1,*]**

[1] Key Laboratory of Space Utilization, Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China; shyli@csu.ac.cn
[2] China Manned Space Agency, Beijing 100034, China; wangz@cmse.gov.cn
* Correspondence: wfzhang@csu.ac.cn

**Abstract:** Cloud computing has become one of the key technologies used for big data processing and analytics. User management on cloud platforms is a growing challenge as the number of users and the complexity of systems increase. In light of the user-management system provided by major cloud service providers, which cannot manage multiple types of user systems, this article proposed scale-out automated expansion user management for authorization synchronization to improve the efficiency and scalability of user management on cloud platforms. Three modules for user-automated expansion were designed and implemented to synchronize the authentication information from the cloud platform resource user to the data-processing user. Additionally, an optimized dynamically weighted load-balancing algorithm in Nginx is presented in this article that adjusts the weight according to load information such as CPU and memory usage, and a better load balance can be achieved. The effectiveness of the proposed user-management system is substantiated by comparing it with two existing infrastructures, including multiple data centers and the Huawei cloud platform. The experimental results validate the finding that scale-out automated expansion user management across the Huawei cloud platform can effectively synchronize data accessing authority with cloud resource utilizing authority. Furthermore, the optimized weighted load-balancing algorithm is also valuable for massive concurrent user registration based on limited cloud resources. In the future, this scale-out user-management system could be applied to other cloud platforms and extended by database synchronization to satisfy the needs of data sharing among multiple types of users belonging to different cloud platforms.

**Keywords:** automated expansion; user-management system; cloud computing platform; user authority synchronization



**Citation:** Li, S.; Wang, Z.; Zhang, W. Automatically Expanding User-Management System for Massive Users in the Cloud Platform. *Appl. Sci.* **2024**, *14*, 2549. https://doi.org/10.3390/app14062549

Academic Editor: Andrea Prati

Received: 5 December 2023
Revised: 21 February 2024
Accepted: 22 February 2024
Published: 18 March 2024



## 1. Introduction

Although the term "Big Data" is by no means a novelty in the field of aerospace engineering, there are many challenges that remain unsolved. A vision of data science, based on the activities of people who are "learning from data", was presented to improve that activity in an evidence-based manner. This new field is a better academic enlargement of statistics and machine learning than data science initiatives while being able to accommodate the same short-term goals [1]. As the data volume in aerospace engineering has dramatically increased in the past decade, the data-driven mode is often named the fourth paradigm. Especially for data-intensive science, data capture, curation, analysis, and sharing are facing new challenges [2]. Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Regardless of whether a cloud provider sells services at a low level or a higher level, cloud resources, including computing, storage, and networking, must all focus on the horizontal scalability of virtualized resources rather than on single-node performance [3].

A three-dimensional role-based user-management model is proposed. The three-dimensional role is defined as a vector composed of authority, scope, and permission time. This user-management model, based on the three-dimensional role, can satisfy the requirements of modern application systems and large-scale systems. Considering that users participate in an enterprise system as a particular identity, this results in roles of different users having different authorities, and each authority is valid within a relevant scope during its permission time [4]. In a geospatial hybrid cloud platform based on multi-sourced computing and model resources, a user-management module was proposed to manage the accounts for different types of end users. There were several categories of users, including cloud resource contributors, cloud resource consumers, and cloud administrators, to be managed in this user-management module [5]. Some researchers designed an infrastructure of multiple data centers (MDCs) for managing and processing massive remote-sensing images by introducing access security and information services. Based on this MDC, they succeeded in resolving problems regarding procedures in processing applications collaboratively and transferring massive remote-sensing datasets quickly, with stable cross-MDC with the aid of a data center user-management system [6,7]. Although the conception and application of big data processing have been presented and practiced, user-management systems across distributed data centers or cloud platform nodes have rarely been mentioned.

Especially with the cloud computing technology being put into use for updating the elastic supporting capacity of the cloud platform, this paper proposes an architecture and functional design of the remote-sensing cloud service platform and introduces a prototype. This remote-sensing cloud service prototype allows users to choose required remote-sensing data and software and automatically deploys them to a virtual computer that users can access through the Internet to perform their remote-sensing data processing and application. Experiments show that the remote-sensing cloud service platform can gather remote-sensing information, software, and computing resources from different providers and provide them for sharing on user's demand [8]. By virtue of its excellent scalability and high reliability, the cloud computing system has been introduced to provide significant technical support for improving the service capability of the cloud platform. A hierarchical framework was proposed that allows for an integrated and homogeneous management of heterogeneous platforms but, at the same time, preserves the autonomy of single sites. This paper describes the hierarchical architecture of EcoMultiCloud for efficient management of the workload in a multi-site scenario. Performance analysis has proven that the hierarchical approach achieves nearly the same quantitative results as a centralized architecture. A user-management system was constructed for job submission and monitoring across distributed data centers. Authority for data access was not able to be synchronized with resource access [9]. In a proximity-based self-organizing framework, service information such as user management and resource management is stored in service descriptors that are ordered spatially so that frequently co-used descriptors can be easily found in neighbor hosts. In this paper, the framework is decentralized and self-organizing, which can guarantee good scalability and fault-tolerance characteristics. Performance analysis based on event-based simulation shows that the ant-inspired algorithm succeeds in achieving spatial ordering of descriptors, and the associated discovery algorithm allows search requests to better satisfy user requirements with lower resource consumption [10]. This paper aims to develop a Mobile Cloud Computing-based knowledge management platform to manage lessons learned in different construction projects of small-to-medium enterprises through an iterative and user-centered methodology. The focus is on user-centered design to develop the knowledge management platform [11]. It follows that the user-management system among different geographically distributed data centers and cloud platform nodes was only for job submission in [9–11].

Facing the trouble of which service provider was ideal for cloud users, comparisons were made on a wide range of criteria, including the services and tools offered by each provider, the platforms they run on, the level of security and scalability, and the amount of data that can be stored in the cloud platform [12]. Among all these technologies, an

automated expansion user-management system for massive customer synchronization between the cloud platform and application software has seldom been mentioned.

Cloud-based applications cost less since the customer does not need to pay for all the hardware and software, facilities, or extensive configuration and maintenance of a full technology stack to run them. Cloud provides more scalable, more reliable, and more secure service. The major cloud service providers, such as Alibaba Cloud, Huawei Cloud, and Amazon Cloud, provide available resources and interfaces for computing and storage, so we can develop and deploy a variety of application software on it [13]. This paper identifies the strengths, weaknesses, opportunities, and threats in the cloud computing industry [14]. The three core technologies, including virtualization, multitenancy, and web services, are elaborated upon to unfold the evolution of cloud computing. They summarize the various issues that affect the different stakeholders of cloud computing.

This paper introduces a way to overcome the current cloud service limitations, which cannot provide sufficient quality of service (QoS) guarantees for some applications. The authors' work focuses on how to provide and guarantee quality of service requirements for resource networks within an Infrastructure-as-a-Service (IaaS) framework. Some methods were studied for network resource management and flow control, as well as QoS models. At execution, they developed a framework that enables QoS support for real-time services executing within an IaaS environment [15]. In cloud service management, a cloud service user has several choices for service selection, and the quest to achieve interoperability and compatibility in cloud computing will consequently enable the user to easily migrate between service providers [16]. One paper demonstrated how the solutions to request routing and application placement can be integrated into an overall design for a peer-to-peer management middleware that exhibits properties of self-organization [17]. Through complexity analysis and simulation, the authors chose to which extent the system design was scalable. They built a prototype using accepted technologies and evaluated it using a standard benchmark. The testbed measurements show that the implementation, within the parameter range tested, operates efficiently, quickly adapts to a changing environment, and allows for effective service differentiation by a system administrator.

Under normal conditions, the user-management system of the cloud platform and the user's utility application were independent, respectively. The user-management system was used to allocate computing, storage, and network bandwidth dynamically. The user-management system of the utility application was created to process data using data and invoke the software interface. Because these two types of user-management systems were developed by different departments, they cannot assign or grant operating authorization to each other. Therefore, it is necessary to customize a user-management system for user authorization synchronization.

The OpenStack open-source tool package has also been used to build a private cloud platform to satisfy the requirements for an experimental environment and software system [18,19]. These scientific experiments in our project will bring forth large sets of data. In order to process data generated by space scientific experiments and research programs efficiently, we built a cloud platform based on the technical systems of the OpenStack cloud platform. This cloud platform was named the OpenStack Private Cloud Platform (OPCP) to provide resource pooling for diverse fields of scientific users. For scientific users, data production by means of computing resources and scientific data has a very important significance.

Based on all above-mentioned research and analysis, we can draw the conclusion that there has been no multi-user-management system automatic synchronization in past research, as shown in Table 1.

**Table 1.** List of some system architecture.

| No. | System Architecture | Load Balance of Workload | Multi-User Management |
|---|---|---|---|
| 1 | Infrastructure of multiple data lefts (MDC) | An open-source scheduler was introduced to perform load balance | No multi-user-management system automatic synchronization within all the architecture |
| 2 | Hierarchical architecture of EcoMultiCloud | An efficient management of the workload was included | |
| 3 | Major cloud service providers, such as Alibaba Cloud and Huawei Cloud | Some open-source and commercial scheduling software within cloud service providers | |
| 4 | OpenStack open-source cloud | Some open-source scheduling software was integrated in OpenStack | |

Therefore, this paper will re-establish a scale-out multi-user-management system to break through the obstacle between cloud resource and application data (remote-sensing data and other scientific data).

Moreover, it is impractical to add each user manual as the number of users grows. Especially for our cloud platform, it was calculated that the number of users has reached four thousand at present and is expected to grow with time. An automated expansion multi-user-management system could definitely improve the efficiency of cloud platform resource utilization, but constructing a system that covers user authorization management and user organization relationships is extremely difficult. The user-management system of OPCP was initialized based on the user's architecture of Stack 6.5.1, which belonged to Huawei's private cloud. As a matter of fact, Stack 6.5.1 of Huawei's private cloud was derived from OpenStack architecture. The framework of user management within Stack 6.5.1 was already in use, so we expanded to accommodate plenty of cloud users automatically based on Stack 6.5.1 [20].

This article is structured in sections, as follows: summary with keywords; (1) introduction with literature review; (2) architecture of the user management proposed for implementation of user authority synchronization, dividing it into three parts: user information synchronization module, user identification and authentication module, and user registration and request module; (3) a dynamically weighted load-balancing algorithm presented to adjust the weight of worker node for improving scheduling efficiency; (4) experimental environment and illustrated comparison with other methodologies; (5) conclusions; (6) and bibliographic references.

## 2. Methodologies

### *2.1. Architecture of the User-Management System*

To establish an automated expansion user-management system, we constructed a prototype that can achieve user authority synchronization on the Huawei cloud platform. The version of HuaWei cloud is Cloud Stack 6.5.1. Users in Virtual Data Center (VDC) on Cloud Stack 6.5.1 were registered to utilize the virtual machine and elastic storage services. We can make use of the fundamental methods and interfaces provided by Cloud Stack 6.5.1 to construct an architecture for user authority synchronization.

As shown in Figure 1, on the top of this architecture, it must be convenient to register for data users. In order to register users by means of methods and parameters, the user-register class should be rewritten to realize user registration and requests in the Data User Register Tier. After the user was registered, the function of user authority and user role mapping was invoked in the Authority Management Tier. The class BaseController was the most significant base class for the user-management system. There were

three controller classes inherited from the class BaseController in the User Information Processing Tier. In this paper, we designed three modules to register users, identify user information, and synchronize user authentication. All three modules were inherited from the class BaseController. At the bottom of this architecture, the VDC resource user can be created in ManageOne, which is a management module of HuaWei Cloud Stack 6.5.1. More logical methods and interfaces were provided by ManageOne in the VDC User Resource Tier.

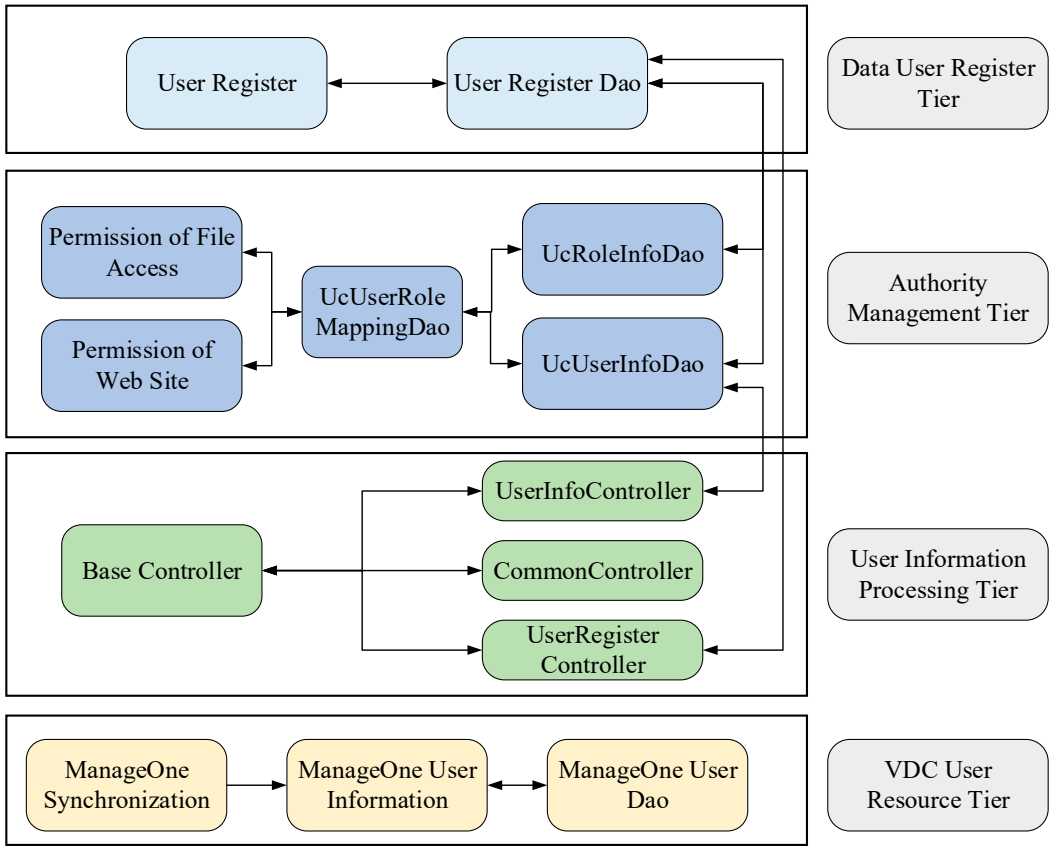


**Figure 1.** The architecture of the user-management system.

In Figure 1, we defined a user role system within the VDC User Resource Tier to represent the VDC users' operations that will be synchronized with ManageOne. In order to give the VDC users the authority to access the database and data files, a decisive functional module named authority management was designed and implemented. Currently, the data storage module inside usually consists of file storage and database storage. So, the authority management will be divided into permission of file access and permission of the web site.

The VDC resource user system contains five levels of resource pools at most. Each level of the resource pool possesses quotas, respectively. When a resource user was created within any level of the pool, all the user information was synchronized to the authority management system.

### *2.2. Implementation of Users' Automated Expansion Module*

The number of OPCP users will increase dramatically as the scale of cloud resources gets bigger. In order to implement the modules of user-automated expansion, we should analyze the basic synchronization steps from Huawei Cloud Stack 6.5.1 users to user authority management clearly. Altogether, the synchronization procedures can be divided into eight steps, from users being created in Huawei Cloud Stack to being authorized to access the data.

Among all the steps, user authority management plays a critical role in user group partition and authority assignment. We have summarized a series of interfaces and classes by analyzing the implementation of VDC user creation. By expanding these interfaces, we can acquire the VDC user attributes and endow the data-access authority to these users.

The module of the users' automated expansion is composed of three parts, that is, the user information synchronization module, the user identification and authentication module, and the user registration and request module. For a large-scale cloud platform, it is indispensable to synchronize user information and authentication between two types of user-management systems. So, we should rewrite some classes and interfaces of user information synchronization. In the meantime, the original function of user identification and authentication within the cloud platform is insufficient to manage massive users and huge amounts of data. Some of the important methods for user identification and authentication are deficient, including user role mapping, user group dividing, and user privilege creation. Therefore, these three categories of user information methods are requisite for automated expansion of users' accounts and synchronizing users' authority. Based on the above definition of three modules, the flow of user authority synchronization is described in Figure 2.

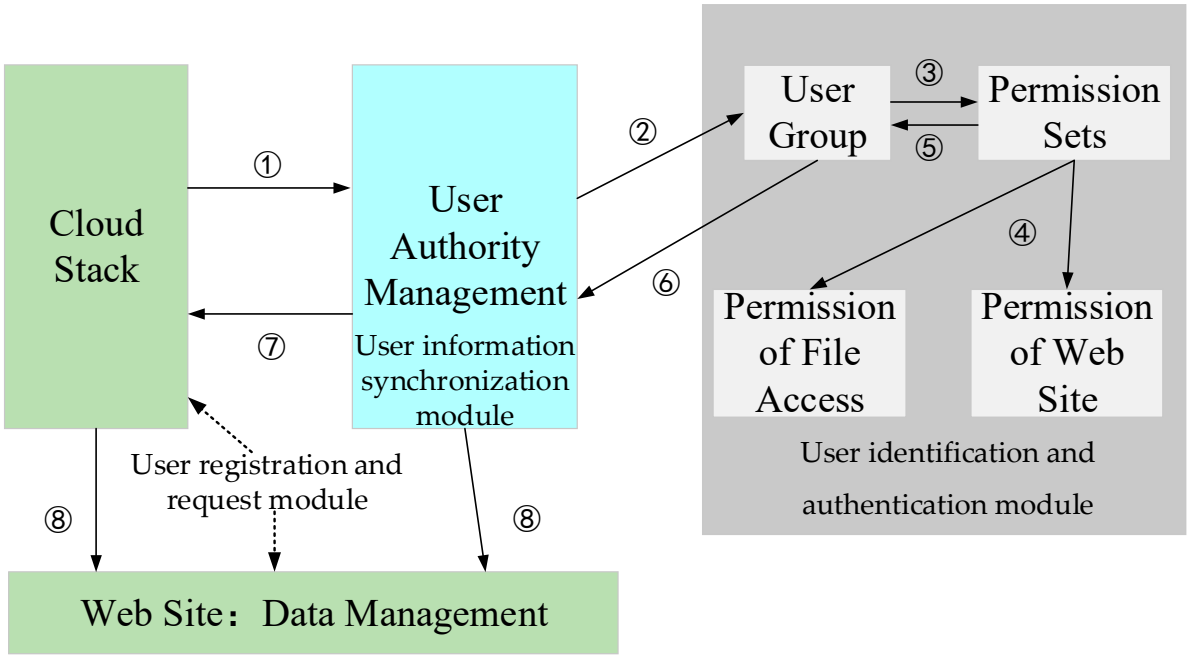


**Figure 2.** Basic synchronization flows from Huawei Cloud Stack users to user authority management.

As shown in Figure 2, the VDC users from HuaWei Cloud were established and were synchronized to user authority management (①). Then, the VDC users were divided into several user groups (②). All the VDC users in user group were attached to permission sets (③). The permission set was composed of accessibility scope of web site and the operating authority on the disk file (④). By means of permission sets, each user group should correspond to one or more types of operating authority (⑤). All the permissions of user group were recorded in user authority management (⑥). After the synchronization, the VDC users from HuaWei Cloud possessed permission to access data and web site (⑦). So, users across the platform can visit each other (⑧).

Among all the base classes provided by OPCP, the class BaseController was the most significant for the user-management system. We defined the class UserInfoController, CommonController, and UserRegisterController, which were inherited from the class BaseController to implement synchronization of user authority between cloud platform and business software.

2.2.1. User Information Synchronization Module

Firstly, we rewrote the class UserInfoController to contain all the public methods, such as naming rulers and login regulations. As shown in Figure 1, the class UserInfoController was inherited from the class BaseController and belonged to the user information process tier. We constructed some classes such as UserInfoService, UserInfoDao, MoUserInfoService, and MoUserInfoDao to synchronize from HuaWei Cloud Stack users to user authority management, as shown in Figure 3.

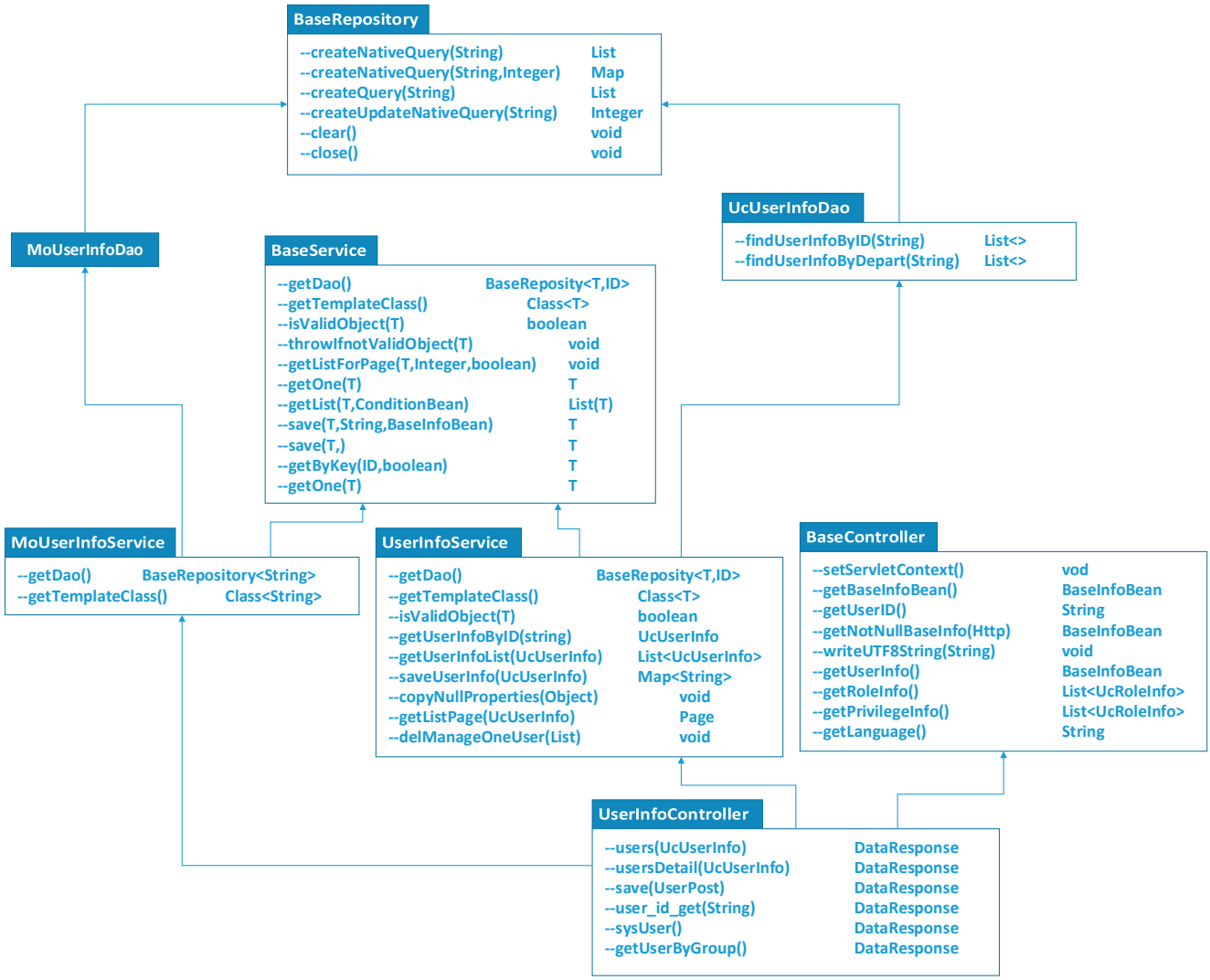


**Figure 3.** The interface class of user information synchronization.

The above methods and interfaces can supply functions for the application of user authority synchronization.

- There are more than a dozen methods in the class BaseController, so we can take advantage of most of them. A crucial method getUserByGroup of class UserInfoController was constructed to divide the user group.
- Another class, UserInfoService, was introduced to the class UserInfoController for processing logical transactions. For instance, when a VDC user sets the account password, the method entryptPassword was used to encrypt the password securely. Meanwhile, user information can also be acquired via the user's identification or user's department through the method getUserInfoById and getUserInfoByDepartmentId.
- The class UserInfoService was inherited from the class BaseService. Some basic logical methods such as the paging querying and database table processing were contained

within the class UserInfoService. These logical methods can be used as the methods of a generic class.

- The class UserInfoDao was a data tier to provide some custom methods. We can invoke these methods from the service tier to complete a variety of complicated functions on database operations.
- To call functions of synchronization service within ManageOne, we constructed a significant service class, MoUserInfoService, which created some interfaces. User name, user permission, and user creation time were all synchronized by means of this class.
- Another class, MoUserInfoDao, was a data-access tier that can obtain information from ManageOne of the HuaWei Cloud Stack user authority management so that user authorization of the user-management system would synchronize with Huawei Cloud Stack.

#### 2.2.2. User Identification and Authentication Module

Secondly, we defined the class CommonController to implement user identity authentication. The class UserLoginInfoService and the class RoleInfoService were constructed to set user privilege, user department, and user id. In the meantime, some of the most important methods, including user role mapping, user group dividing, and user privilege creating, were implemented within these two classes, as shown in Figure 4.

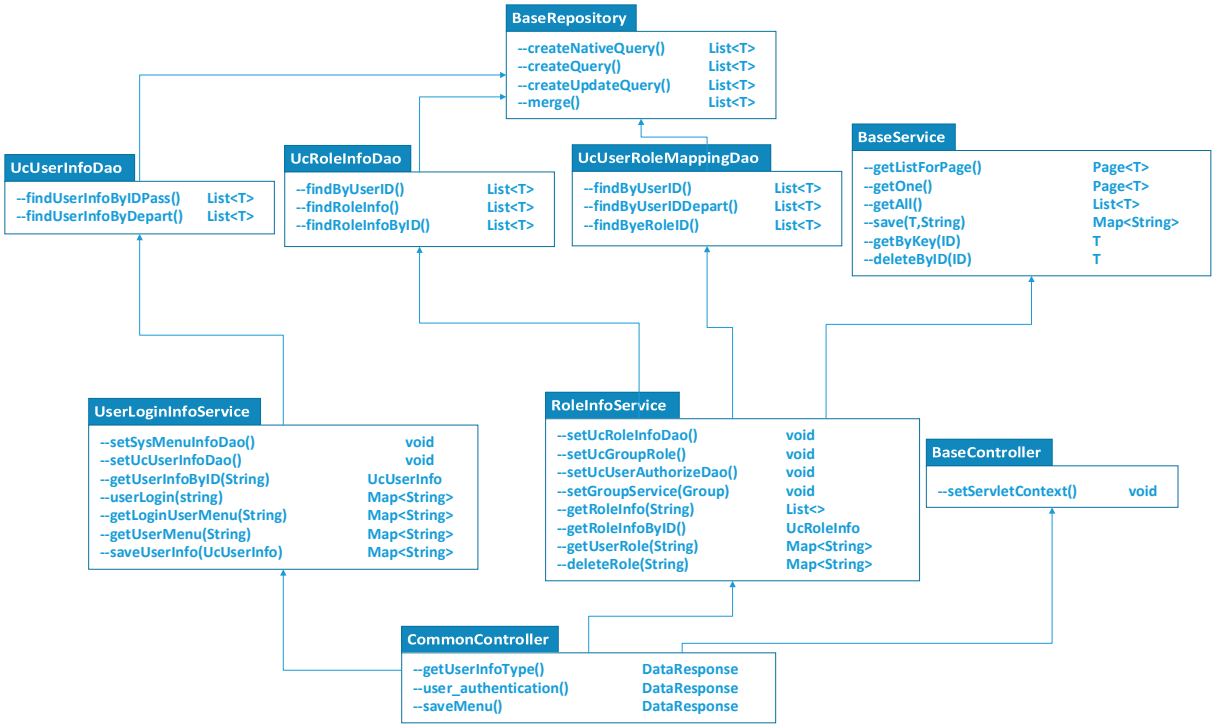


**Figure 4.** The interface class of user identification and authentication.

Using the above methods and interfaces, we created and implemented some classes for the application of user authority identification and authentication.

- The class CommonController was defined as a controller class to implement user identity authentication and was inherited from the class BaseController. All the authentication services interfaces consist of getUserInfoType, putAuthentication, and saveAuthentication, which are provided as Restful API. As shown in Figure 1, the class CommonController was inherited from the class BaseController and belonged to the user information process tier. The class BaseController was a customized parent class of the control tier to provide certain primitive methods in common.

- The class UserLoginInfoService and the class RoleInfoService were created within the class CommonController. These two classes contained a large number of methods for handling the transaction logic between the data user register tier and the user information process tier. The interface setGroupRoleMappingDao and setUserAuthorizeDao were used for user grouping and permission assignment in the class RoleInfoService.
- The classes UcUserInfoDao, UcRoleInfoDao, and UcUserRoleMappingDao were defined within the data-access class for operating the database. We can find the user's department using the method findUserInfobyDepartmentId and the method findUserInfobyIdandPassword. So, when a new user is registered through the cloud resource user's management system, we can map the privilege of cloud resource users to the data users using the class UcUserRoleMappingDao. Once the privileges of these two types of users are bridged, any type of user can transform permission of file accessing and website accessing each other (as shown in Figure 1). Moreover, these classes had the characteristics of some public data operating methods, which were inherited from the class BaseReposity.

#### 2.2.3. User Registration and Request Module

Thirdly, the user registration and request module was implemented within the user information process tier. As shown in Figure 1, the class UserRegisterController was inherited from the class BaseController. On this basis, some classes, including UserRegisterService, BaseService, and BaseRepository, were realized for user registration, as shown in Figure 5.

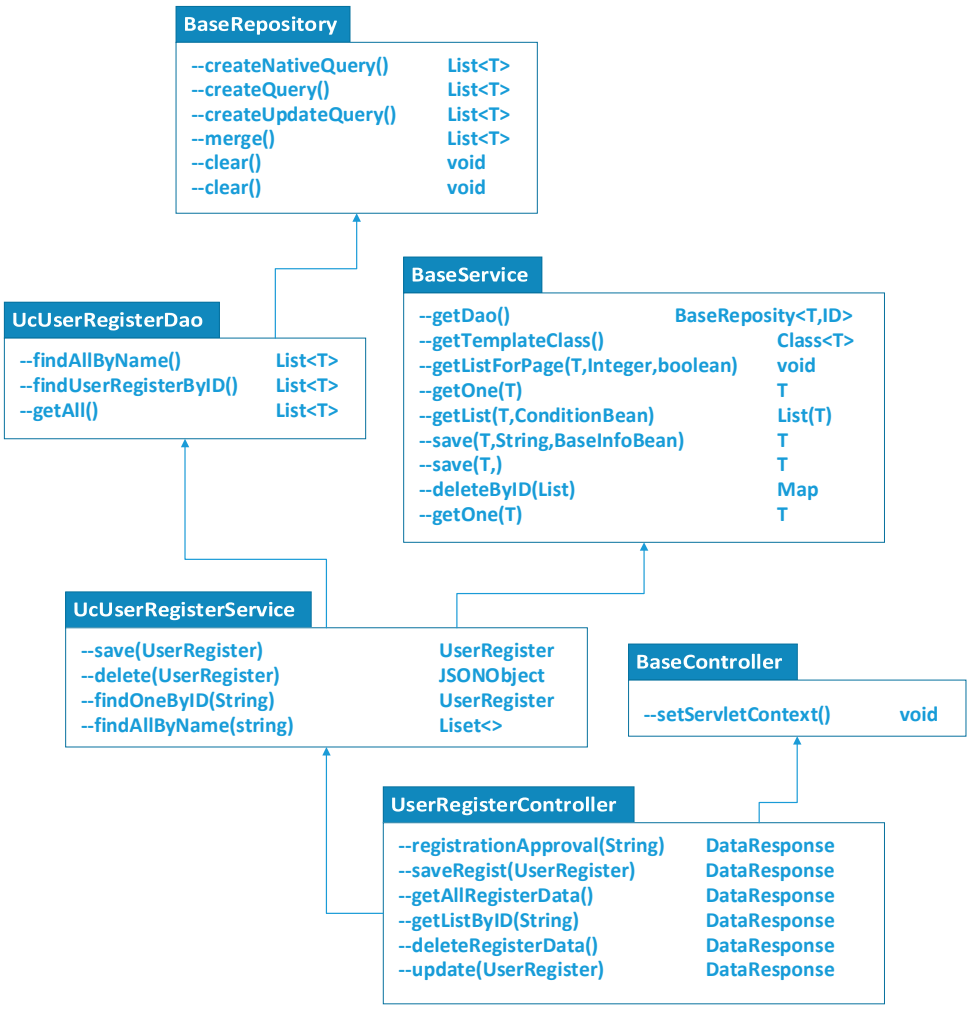


**Figure 5.** The interface class of user registration and request.

- When a user wants to apply for registration of a new account for data application or cloud resource usage, the class UserRegisterController is defined to provide the methods for filling in user approvals, saving user registration, and querying user IDs. The data user registration application and the cloud resource user registration application are differentiated as two registration flows by the class UcUserRegisterService. There are a large number of services and logical methods in this class. Once any type of user has been registered, the user identification and authentication will be synchronized with each other.
- The class UcUserRegisterService was inherited from the class BaseService. Nearly all the methods for user registration were enclosed within the class BaseService. The request for user registration is recorded and examined so that all the registration information is compliant. As shown in Figure 1, the user registration information, after validation, is passed to the class UcUserRoleMappingDao to synchronize user authentication.
- Another class, UcUserRegisterDao, was defined for some customized data-processing methods and complicated database operations. Different user authentications can be merged or detached by means of the methods provided by the class BaseRepository.

## 3. Load Balancing for Massive Concurrent Users

Although the VDC users from HuaWei Cloud can established and synchronized to user authority management, the registration concurrently of a large number of users will cause the automated expansion of user management to be under heavy load. So, a load-balancing strategy is essential for dispersing users' requests and reducing the load of the user registration server.

### *3.1. Load-Balancing Algorithms in Nginx*

Compared with conventional load-balancing tools, Nginx has a better load-balancing efficiency and a lower cost. Nginx uses event-driven architecture, which is a lightweight, scalable, and high-performance HTTP server. It is also very convenient to deploy and manage in our cloud environment. Nginx is an HTTP and reverse proxy server that has the characteristics of load balancing, accelerated reverse proxying with caching, and autoindexing. Some classical load-balancing algorithms have been included in Nginx. DNS refers to the Domain Name System, which is a directory that connects names to the user registration Internet addresses, as shown in Figure 6.

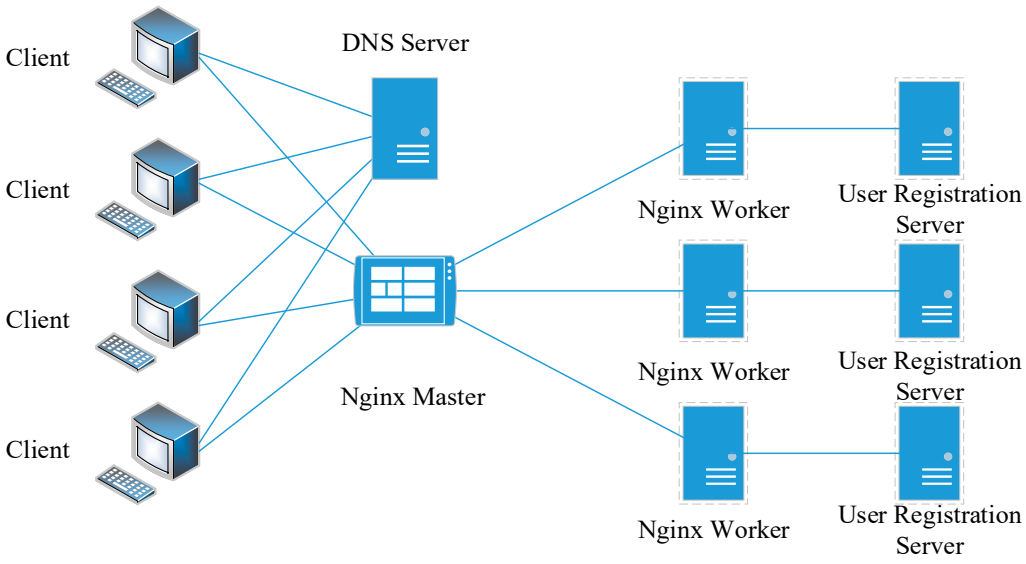


**Figure 6.** The flows of Nginx load balancing.

Figure 6 shows that the Nginx master allocates the user registration requests to other workers. There are some load-balancing algorithms used in Nginx to ensure the Nginx workers running at an equalized level. Typically, the following three load-balancing algorithms can be applied to the large-scale cloud platform.

- Round Robin load balancing: This is a simple way to distribute client requests across a group of servers via the Ngnix master node. A client request is forwarded to each Nginx worker node in turn and submitted to each user registration node. The algorithm instructs the load balancer to go back to the top of the list and repeats again. Round robin is the most widely deployed load-balancing algorithm. Using this method, client requests are routed to available servers on a cyclical basis. Round-robin load balancing works best when servers have roughly identical computing capabilities and storage capacity.
- Weighted Load Balancing: The weighted round robin load-balancing algorithm allows the Nginx master node to assign weights to each Nginx worker node based on criteria like the traffic-handling capacity. Nodes with higher weights receive a higher proportion of client requests.
- IP HASH Load Balancing: The IP Hash policy uses an incoming request's source IP address as a hashing key to route non-sticky traffic to the same user registration server. The load balancer routes requests from the same client to the same backend server as long as that server is available.

Among these load-balancing algorithms, the weighted load-balancing algorithm is more flexible and scalable for the cloud platform. All the Nginx worker nodes can be given different weights depending on their server specifications to realize load balancing. However, the unchanged weights for worker nodes were proved limited for load balancing. So, we chose the weighted load-balancing algorithm for optimization by adjusting the weights of worker nodes.

### *3.2. Optimization of the Weighted Load-Balancing Algorithm*

After the client sends a request to Nginx, all the requests are distributed to worker nodes. This reverse proxy caching mechanism in Nginx can greatly reduce the waiting time for clients. In the weighted load-balancing algorithm, each work node has a fixed weighted value. This paper proposed a dynamically weighted load-balancing algorithm that adjusts the weight according to the load information, such as CPU and memory usage.

All the Nginx worker nodes were provided based on the OPCP, so the network bandwidth and disk I/O of the worker node can be considered uniform. For a worker node, $L_{cpu}$, $L_{mem}$ indicates the real-time load of CPU and memory. $W_{cpu}$ represents the initial weight of the CPU. $(1 - {}_{cpu})$ represents the initial weight of the memory. $P_{cpu}$ and $P_{mem}$ represents the remaining load performance of CPU and memory. So, the dynamic weight of each worker node can be calculated as follows:

$$W_{node} = W_{cpu} \times L_{cpu}/P_{cpu} + (1 - W_{cpu}) \times L_{mem}/P_{mem} \quad (1)$$

If there are i nodes within the worker nodes pool, the set of weights of all the worker nodes can be indicated. The weight of each worker node was updated at 1-min intervals.

$$W_{set} = \{W_{node}\ (1), W_{node}\ (2), W_{node}\ (3), \ldots\ldots, W_{node}\ (i)\} \quad (2)$$

We provided three virtual machines with different specifications as worker nodes in the cloud platform. The virtual machine with high specifications was endowed with a higher initial weight so as to handle more user registration requests. For example, the virtual machine with 64 threads (virtual cores) can be allocated 32 requests, while the virtual machine with 32 threads can be allocated 16 requests.

Each minute, the new weight of each worker nodes was computed via Equations (1) and (2). Then, the three worker nodes were assigned new weights. We

simulated three scenarios to validate the quality of services. This optimized weighted load-balancing algorithm obtains an optimized result, as shown in Table 2.

**Table 2.** Qos of three types of concurrent connections.

| Quality of Service | Concurrent Connections | | |
|---|---|---|---|
| Complete requests | 200 | 500 | 1000 |
| File size (KB) | 1024 | 10,240 | 40,960 |
| Time cost (s) | 1.02 | 1.05 | 1.15 |
| Transfer rate (KB/s) | 1003.90 | 9752.38 | 35,617.39 |

## 4. Results and Discussion

### *4.1. Experimental Environment*

The automated expansion user-management system was designed and implemented based on Stack 6.5.1 in Huawei's private cloud platform. This cloud platform was both the experiment environment and the practical running environment. It is calculated that the number of users has reached four thousand at present and is expected to grow with time. All the users registered in the cloud platform can access the cloud resources and datasets. In order to install the Nginx nodes of the master and worker, we used three different virtual machines provided by the cloud as follows.

As shown in Table 3, the maximum number of threads of the virtual machine was 64. The physical host computing server was assembled with Intel 6100 series CPUs that support the actual maximum frequency of 2666 MHz. The three virtual machines were allocated storage space of the same size. In consideration of the proportional relationship in CPU threads, the above three virtual machines were assigned 4, 2, and 1, respectively, as initial weighted values. Each minute, the weight of three worker nodes was updated.

**Table 3.** List of three virtual machine configurations.

| No. | CPU Threads | Memory (GB) | Disk Capacity (GB) |
|---|---|---|---|
| 1 | 64 | 128 | 100 |
| 2 | 32 | 64 | 100 |
| 3 | 16 | 32 | 100 |

### *4.2. Performance Evaluation*

#### 4.2.1. Performance of Load-Balancing Algorithm

To correctly evaluate the performance of the optimized load-balancing algorithm, it is necessary to contrast the optimized load-balancing algorithm and other algorithms in Nginx. Three virtual machines were allocated from our platform, as shown in Table 3. All the servers have network adapters with a bandwidth of 10 Gbps.

We used round robin load balancing, weighted load balancing, and the optimized weighted load-balancing algorithm to compare the registration time. As shown in Figure 7, we can analyze the case where a better load balance is achieved by an optimized load-balancing algorithm. All the user registrations can be completed within 10 s, as the number of users is less than 10,000.

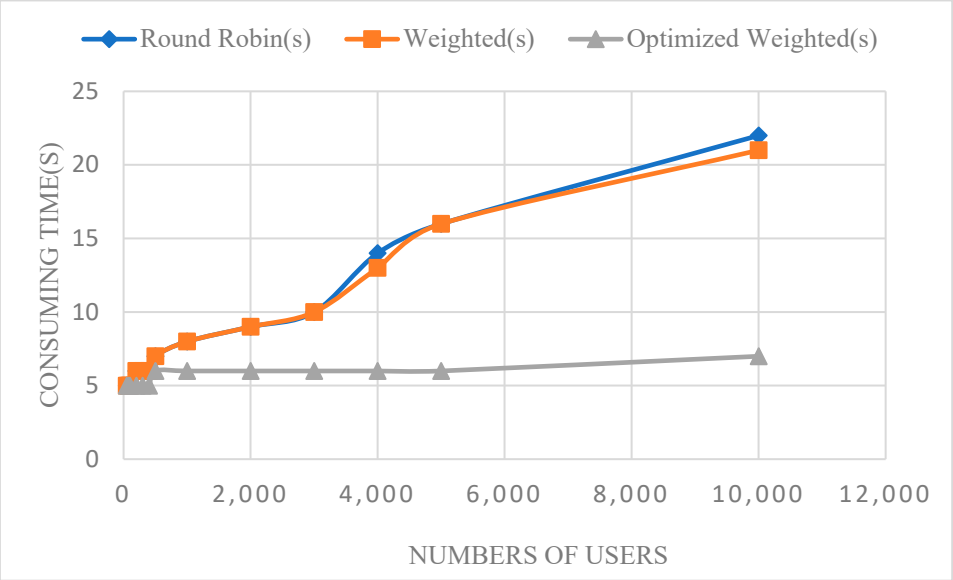


**Figure 7.** Consumption time of three load-balancing algorithms.

4.2.2. Performance of Massive User Registration and Online

After Stack 6.5.1 cloud platform of this project was put into use, massive user registration and authentication were validated by a large number of experiments. Experiment results showed that the achievement time of massive user registration and authentication was successful.

As shown in Figure 8, the experimental result shows that increasing the number of users to ten thousand only needs an extra two seconds for user registration. For this project, several thousands of users were more common. So, registering a new user would take about six seconds in most cases. In order to meet the requirement of time effectiveness for users accessing our cloud platform, we invited two hundred participants to answer questions randomly from thousands of users. According to the user feedback, when they completed the process of user registration and user authority synchronization within 10 s, they had a responsive user experience.

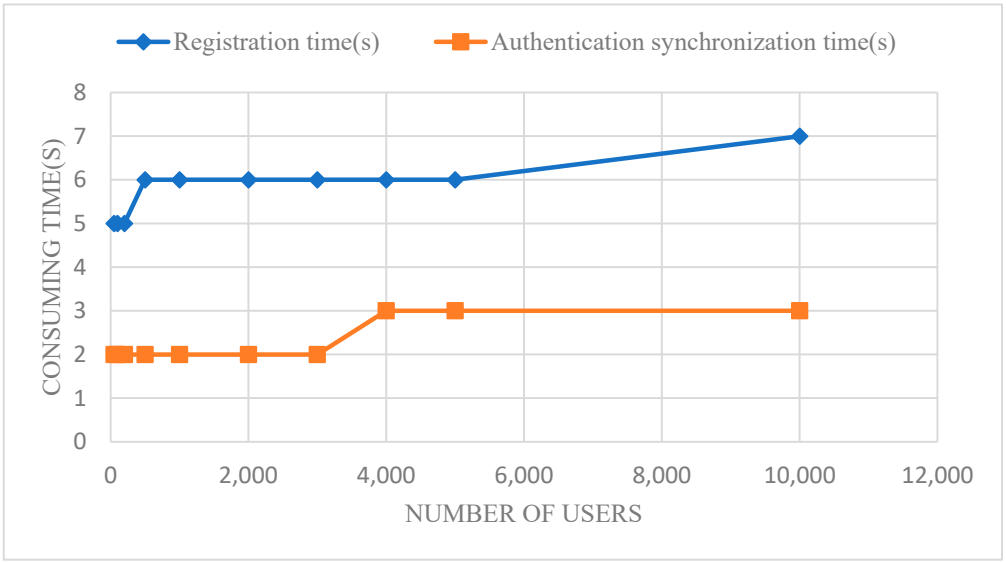


**Figure 8.** Time of user registration and authentication synchronization.

In contrast, we added cloud platform users and synchronized user authentication through ManageOne 6.5.1, which is provided by Huawei Cloud Stack 6.5.1. Because user

authentication synchronization was realized by deploying the ManageOne 6.5.1 software manual, two types of users should be created first. It takes twice the time to create users than the dynamic expansion of the user-management system proposed in this study.

This result reflects that the time of user registration and user authority synchronization once through the dynamical expansion of the user-management system was less than 10 s. It would take 13 s to achieve the same process using ManageOne 6.5.1 software. More importantly, there are five thousand users in our cloud platform at present. In the future, this platform will keep on running for at least ten years and deal with many new user registrations every day.

### *4.3. Comparison with Existing Schemes*

As shown in Table 1, there has been no multi-user-management system that performs synchronization between two types of users automatically in past research works. So, we selected two previous architectures in contrast with the automated expansion user-management system proposed in OPCP.

As shown in Table 4, the time of user registration in OPCP is broadly in line with the two other system architectures. User registration by filling and submitting a user form took approximately 6 s. In the OPCP, user authentication synchronization can be completed by the automated expansion user-management system. In the MDC and Huawei Cloud platforms, resource user authentication can be synchronized with the data user manually. Because the synchronization consists of two distinct phases within the MDC and Huawei Cloud platform, the time of user registration and authentication synchronization is more time-consuming. The total time of user registration and authentication synchronization in the cloud platform can be marked as benchmarking. In comparison, this benchmarking of the automated expansion user-management system in OPCP is generally more effective.

**Table 4.** Time of user registration and authentication synchronization using three types of system architecture.

| No. | System Architecture | Time of User Registration (s) | Time of User Authentication Synchronization (s) | Total Time (s) |
|---|---|---|---|---|
| 1 | OpenStack Private Cloud Platform (OPCP) | 6 | 3 | 9 |
| 2 | Infrastructure of multiple data centers (MDC) | 7 | 10 | 17 |
| 3 | Huawei Cloud platform | 6 | 12 | 18 |

In the meantime, the limitations of the proposed system are that we only completed user registration and authentication synchronization between two categories of user-management systems. For three or more types of user authentication synchronization, one user authority should be synchronized to other users simultaneously.

Therefore, in order to realize massive user registration and authentication synchronization among multiple user-management systems, we should study database synchronization to avoid authentication information inconsistency.

## 5. Conclusions

This research aims to address the challenge encountered while managing multiple types of user-management systems across cloud platforms. In light of the complexity of the two types of users existing on the cloud platform, it becomes a formidable task to implement synchronization of user authority information. This challenge poses significant obstacles to the application of a cloud platform.

The proposed solution entails an architecture that establishes a scale-out automated expansion user-management system, utilizing the fundamental methods and interfaces provided by Cloud Stack 6.5.1 while optimizing the weighted load-balancing algorithm

within Nginx. The study results demonstrate the effectiveness of this architecture in achieving automatic user authority synchronization.

The main contributions of this paper are the three categories of user information modules that can be applied to another large-scale cloud platform to implement the synchronization of massive users. Additionally, the optimized weighted load-balancing algorithm is valuable for massive concurrent user registration based on limited cloud resources.

Ultimately, there is still room for improvement regarding more complex user authority in this study's automated expansion of a user-management system for massive users on a cloud platform. On the one hand, we can apply this architecture and the methods of user authority synchronization to the Alibaba cloud platform. On the other hand, if users want to share their data with other users, database synchronization can be studied to avoid authentication information inconsistency. This can be a topic for further research on the dynamic expansion of user-management systems.

**Author Contributions:** Conceptualization, W.Z.; Funding acquisition, W.Z.; Investigation, W.Z.; Methodology, S.L., Z.W. and W.Z.; Resources, S.L., Z.W. and W.Z.; Software, Z.W. and W.Z.; Supervision, Z.W.; Visualization, W.Z.; Writing—Original Draft, W.Z.; Writing—Review and Editing, S.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by the Nation Natural Science Foundation of China. The funding grant number is 41701468.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Donoho, D. 50 Years of Data Science. *J. Comput. Graph. Stat.* **2017**, *26*, 745–766. [CrossRef]
2. Hey, A.J.G.; Tansley, S.; Tolle, K.M. *The Fourth Paradigm: Data-Intensive Scientific Discovery*; Microsoft Research: Redmond, WA, USA, 2009.
3. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
4. Li, J.; Zhang, C. A three-dimensional role based user management model in web information systems. In *Proceedings of the 2012 International Conference on Information Technology and Software Engineering: Information Technology*; Volume 210 of Lecture Notes in Electrical Engineering; Springer: Berlin/Heidelberg, Germany, 2013; pp. 657–665.
5. Huang, Q.; Li, J.; Li, Z. A geospatial hybrid cloud platform based on multi-sourced computing and model resources for geosciences. *Int. J. Digit. Earth* **2018**, *11*, 1184–1204. [CrossRef]
6. Zhang, W.; Wang, L.; Liu, D.; Song, W.; Ma, Y.; Liu, P.; Chen, D. Towards building a multi-datacenter infrastructure for massive remote sensing image processing. *Concurr. Comput. Pract. Exp.* **2013**, *25*, 1798–1812. [CrossRef]
7. Zhang, W.; Wang, L.; Ma, Y.; Liu, D. Design and implementation of task scheduling strategies for massive remote sensing data processing across multiple data centers. *Softw. Pract. Exp.* **2014**, *44*, 873–886. [CrossRef]
8. Ren, F.; Wang, J. Turning remote sensing to cloud services: Technical research and experiment. *J. Remote Sens.* **2012**, *16*, 1331–1346. [CrossRef]
9. Agostino, F.; Carlo, M.; Michela, M.; Giuseppe, P.; Mehdi, S. Hierarchical Approach for Green Workload Management in Distributed Data Centers. In Proceedings of the Euro-Par 2014 International Workshops, Porto, Portugal, 25–26 August 2014; Revised Selected Papers, Part I. Springer: Cham, Switzerland, 2014. [CrossRef]
10. Agostino, F.; Carlo, M.; Giuseppe, P.; Giandomenico, S. A Proximity-Based Self-Organizing Framework for Service Composition and Discovery. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 17–20 May 2010. [CrossRef]
11. Núñez, D.; Ferrada, X.; Neyem, A.; Serpell, A.; Sepúlveda, M. A User-Centered Mobile Cloud Computing Platform for Improving Knowledge Management in Small-to-Medium Enterprises in the Chilean Construction Industry. *Appl. Sci.* **2018**, *8*, 516. [CrossRef]
12. Anurag, R.; Puja, G.; Prapti, G.; Sunital, V.; Sharma, K.K.; Upendra, S. Study of Cloud Providers (Azure, Amazon, and Oracle) According To Service Availability and Price. In Proceedings of the 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN), Salem, India, 19–20 June 2023. [CrossRef]
13. Lakshmi, D.C. Impact study of cloud computing on business development. *Oper. Res. Appl. Int. J.* **2014**, *1*, 1–7.

14. Marston, S.; Li, Z.; Bandyopadhyay, S.; Zhang, J.; Ghalsasi, A. Cloud computing—The business perspective. *Decis. Support Syst.* **2011**, *51*, 176–189. [CrossRef]
15. Voith, T.; Oberle, K.; Stein, M. Quality of service provisioning for distributed data center inter-connectivity enabled by network virtualization. *Future Gener. Comput. Syst.* **2012**, *28*, 554–562. [CrossRef]
16. Zia-ur, R.; Omar, K.H.; Farookh, K.H. User-side cloud service management: State-of-the-art and future directions. *J. Netw. Comput. Appl.* **2015**, *55*, 108–122.
17. Adam, C.; Stadler, R. Service middleware for self-managing large-scale systems. *IEEE Trans. Netw. Serv. Manag.* **2007**, *4*, 50–64. [CrossRef]
18. Zhao, S.; Li, L.; Ling, X.; Xu, C.; Yang, J. Architecture and scheduling scheme design of TsinghuaCloud based on OpenStack. *J. Comput. Appl.* **2013**, *33*, 3335–3338. [CrossRef]
19. Deng, Z.; Duan, Z.; Li, L. Research of dynamic scheduling of resources under the environment of OpenStack. *J. Northwestern Polytech. Univ.* **2016**, *34*, 650–655.
20. Website of Huawei Cloud Stack Guide, Managing User Groups, Creating a User Group. Available online: https://support.huawei.com/enterprise/en/doc/EDOC1100296026/3a95b21f/managing-user-groups (accessed on 21 December 2023).