Hindawi
Mobile Information Systems
Volume 2022, Article ID 9139257, 10 pages
https://doi.org/10.1155/2022/9139257

Research Article

Design and Implementation of Software-Defined Data Center (SDDC) for Medical Colleges and Universities

Wei Lin, YuMing Wu, and Ning Jiao

Hebei Medical University, Shijiazhuang, Hebei 050031, China

Correspondence should be addressed to Wei Lin; linwei@hebmu.edu.cn

Received 8 April 2022; Revised 9 May 2022; Accepted 20 May 2022; Published 9 June 2022

Academic Editor: Abid Yahya

The current medical universities culture can reflect their professional nature, showing a strong development trend. The current socialist economy of China is rapidly developing along with cloud computing for data centers. For cultural and educational activities, virtualization and software-defined data center (SDDC) technologies are being used. The most widely used open-source virtualization technologies are SDDC and virtualization. Users may utilize OpenStack to establish private cloud computing environments. The separate control and forwarding architecture of SDDC also make it naturally suitable for the data center's network environment. Ryu controller has become one of the most widely used SDDC controllers because of its lightweight, high efficiency, and modularity features. Due to the large variety of network operations required by the SDDC, appropriate management and control function modules must be developed on Ryu when it is used as the controller. This study investigates the construction of campus culture at medical colleges and universities using SDDC. Our main objective is to improve the efficiency of data construction in medical colleges and universities and creating a positive cultural environment. On the one hand, the addition of functional modules makes the management of the controller itself easier. Ryu lacks an intuitive interactive platform. Although OpenStack provides an interactive interface, it cannot meet the integration requirements of Ryu. The unified control and school scheduling system have some research and application potential. The experimental results show that our proposed approach of designing an SDDC for medical colleges and universities has a significant impact and has vast potential for future studies.

1. Introduction

In recent years, cultural construction projects were launched in local colleges and universities in China. Also, with construction rising tide, medical schools have started their cultural construction projects, with corresponding positive results. The current cultural construction of colleges and universities, on the other side, faces significant difficulties. There will be constraints on the implementation of construction projects due to a variety of factors such as a lack of funding and school transformation, which will result in the construction projects getting terminated. As a result, the socialist market economy is rapidly developing [1]. Data centers have been an important part role of the client-server computing architecture since the 1990s. Due to the rapid improvement of Internet technology in recent years, medical colleges and universities have started to place a higher focus on software data center facilities as a source of cultural development. Therefore, the current campus cultural construction activities need to establish a systematic management to ensure that various construction plans are implemented. Implementation can effectively improve various construction projects, particularly the development of cloud computing technology, which has brought major changes to network services and prompted a climax in data center construction [2]. In the cloud computing environment, data centers provide network services to colleges and universities through the Internet. However, the construction of a data center requires not only various basic hardware facilities but also a software architecture that manages various resources of the data center and provides services to the outside world. At present, the more mainstream cloud architectures include AWS (Amazon Web Services), VMWare (Virmal Machine Ware, Wei Rui), and OpenStack.

Among them, OpenStack, with its open-source features and extensive community support, has gradually become a private cloud construction. OpenStack is mainstream, and it has a much higher level of acceptance than similar open-source projects [3].

In recent years, the software-defined data center (SDDC) has become a popular topic in the field of network technology. The control and forwarding planes are separated in SDDC that adopt network architecture. The southbound communication interface is used by the controller to programmatically control the network devices on the forwarding plane. One of SDDC's earliest standards, OpenFlow was developed initially to provide a real-time experimentation environment for new network protocols, developed by campus network researchers. The SDDC controller controls the OpenFlow switches in the network through the OpenFlow protocol and then manages the entire network [4]. The key problem in the development of traditional data centers (in the cloud computing direction) is that it takes a lot of investment to realize cloud services. The server virtualization carries out the corresponding transformation of the existing network, while SDDC can quickly and efficiently realize the network virtualization and reduces the transformation cost. To solve the difficulties that arose during the transformation, Google's SDDC controller Onix-group has been successfully deployed on the data center's internal backbone network. Because of its near 100% bandwidth utilization, the data center has been the industry's central focus for promoting SDDC technology.

When the data center continues to expand and deploy on a large scale, the management and control of the data center are essential challenges that SDDC needs to solve. Currently, OpenStack is widely used in infrastructure as a service (Iaas) cloud computing data center to provide computing, storage, and virtual resources. The management of application modules has become increasingly complex as a result of the deployment of SDDC in data centers and the continual improvement of controller functionalities. Although OpenStack provides a graphical user interface to assist user operations, SDDC controller integration is problematic. Therefore, the focus of this research is to design a resource management system. That achieves unified management and control of the SDDC controller and OpenStack by evaluating the software-defined data center's present requirements [5].

Academic data center research now focuses mostly on network structure, energy consumption control, and virtual machine scheduling, with just a few studies focusing on management techniques. In China, Huawei's Manage one management system, which automates service deployment and orchestrates the rollout process, is very well. Operators may use this as a starting point to construct their preferred service development methods and quickly launch new services. This solution divides the management system into a business center, IT service management center, and service center. The operation and maintenance center, according to functions, manages data center resource equipment through the cooperation of each part. The management system based on this solution can not only be deployed in the public cloud and the private cloud but can also be applied to traditional data centers. Most of the research on network management systems is based on SNMP (simple network management protocol). Before the concept of SDDC and OpenFlow was proposed, in 2007, Shu Chang realized an SNMP agent by compiling NET-SNMP to manage the network. In 2008, Zhang Jie developed a network management system with network topology discovery and traffic monitoring based on the SNMP protocol system. In 2014, Wang Hongmin developed a management system to manage the devices and resources in the SDDC network. Particularly with the advent of SDDC, the OpenFlow flow table, network topology, tenant network, and so on were all mentioned in the text, but the management of servers and hosts monitor was not [6–8]. In 2015, Zhang et al. realized the real-time management of the physical network topology and the OpenFlow flow table query function through the NOX controller in the SDDC field combined with the SNMP protocol. The data were integrated into SNMP because other systems may get access to the SDDC network through the SNMP interface. However, the NOX controller's support for the OpenFlow protocol is still in an earlier version, so it is more limited [9].

In other foreign countries, Hewlett–Packard Company's Open View, CA of CA Company Uni-center, and IBM's Tivoli are instances of representative network management software. The solutions provided by the three companies provide solutions for data center network management from different perspectives. Although these tools are fully functional, only some of them are used in actual data center management, and researchers are more inclined to lower-cost open-source tools. Furthermore, commercial software was indeed particularly susceptible to scalability problems and will be unable to meet some of the special needs of a software-defined data center [10]. Medical colleges and universities are inseparable from the construction of software facilities in carrying out various teaching activities. At this stage, the strengthening of software facilities in many colleges and universities has made the economic development of software research inevitable for social development. Most of the students use various resources in the construction of the campus, which to a certain extent has also achieved the development of the school's ideological and cultural aspects. As the school develops, it will reflect the corresponding teaching concepts. Medical colleges and universities will reflect the corresponding school-running characteristics and the vigorous development of cultural connotations. The following are the main contributions of this study:

(i) Users may utilize OpenStack to establish private cloud computing environments. The separate control and forwarding architecture of SDDC also make it naturally suitable for the data center network environment.

(ii) The Node and V8 engines are implemented in C and C++ and are positioned for high performance and low memory consumption.

(iii) SDMs (software definition management system, SDDC) three-tier architecture includes application

layer, control layer, and infrastructure layer. It is based on HTML, CSS, Ajax, and JavaScript.

(iv) A complete proxy for OpenStack Dashboard is designed and implemented after an analysis of login, HTTP requests, and WebVNC workflow. The dashboard can connect to the Internet through the management system for users to access.

The rest of this study is arranged in a logical order: Section 2 shows related work, Section 3 shows the related technology, Section 4 shows system structure design, and Section 5 shows the design and implementation of main functional modules. Finally, Section 6 illustrates the research to a conclusion.

2. Related Work

There have been a few related works on network management. Presently, a host of new communication networks is surfacing, and therefore, we must properly manage those [11]. A very well-managed services protocol is the SNMP protocol. Numerous traditional NMSs use SNMP to manage networks. As a result, some studies employ SNMP to manage new network architecture, allowing traditional systems to take control. For cloud network management, CNMM, for example, enhances SNMP [12]. This specifies a management architecture as well as a manager-agent network model for coordinating information stored on multiple portions of the multistage router to provide a unified view to an external network management station issuing SNMP requests [13]. MIBs can be used to designate forwarding planes. In the context of SDN, the ITRI container computer provides an SNMP-based monitoring subsystem. However, it does not have an SNMP-based NMS for managing SDN [9]. It provides an architecture for an integrated network management and control system (INMCS), which combines traditional network management features like discovery and fault detection with SDN-enabled end-to-end flow provisioning and control [14]. It is a hybrid control model. However, neither of these SDN solutions has multiservice management. The testbed in this research is based on previous work. Customers may now design their protocol formats and flow processing rules in physical switches using an SDN-based protocol-independent platform. It describes an autonomous architecture for an SDN-based multiservices network [15].

3. Related Technology

The software-defined data center is designed and developed based on the SDDC architecture and the OpenStack cloud computing platform. The SDDC controller manages and controls the network devices in the data center through the OpenFlow protocol, while OpenStack performs unified regulation on storage, computing, and other virtual resources [16, 17]. The management system is located at the application layer in the SDDC architecture and performs unified scheduling management for the SDDC controller and OpenStack at the control layer. This section mainly provides an overview of SDDC architecture and some components of OpenStack and focuses on the server-side and front-end development technologies used in the development of management systems, as well as Node.js, RabbitMQ, and SNMP protocols [18]. As described in this study, software-defined networking (SDN) provides a logically centralized controller with a global view of the whole data center network. As a result, it provides you with the best of both worlds without requiring the least amount of work. The comprehensive management ability of traditional centralized system cannot be compared with the scalability of the large-scale distributed system. To improve the data quality of stored and processed data and the QoS of multitenanted data center network clouds, we want to adopt an expanded SDN controller architecture [19]. This study identified the first harmonic integration of an optical flexible hardware framework with a rapid application and virtualized environment. On the highest part of the cloud server, the powerful software-defined networking (SDN) central controller makes it possible for the virtual machines of computing and communication resources. To create a virtual data center (VDC) and virtual network functions (VNF), the profoundly programmable all-optical circuit and packet-switched data plane can perform on-demand unicast/multicast switch-over. The authors illustrate realistic intradata center networking with predictable latencies for both unicast and multicast commuters, as well as monitoring and database migration scenarios, all of which are enabled by the network function virtualization (NFV) component [20]. This study presents a unique architecture for spatial orchestrating of network-intensive software applications that are tuned for high service quality. This design includes a global cluster manager for software-defined data centers (SDDCs), a runtime QoS monitoring system, and a QoS modeller and decision-maker for automation tool utility coordination. The developed software automatically selects the optimum spatially accessible computational resources inside the SDDC based on the software component's provided QoS model. This design is event driven since the services are launched and destroyed in real time for each usage event [21].

3.1. SDDC and OpenFlow. The SDDC architecture is shown in Figure 1. The network is divided into three layers: the application layer, the control layer, and the infrastructure layer. The control layer centrally manages and controls the infrastructure layer through the southbound interface, such as the OpenFlow protocol, while the application layer uses the northbound interface [22]. The interaction with the control layer is implemented. In contrast to traditional network distributed control, SDDC extracts the control system of each network device and uses it as the controller, with the network device only acting as a simple forwarding device. The controller formulates the forwarding rules to achieve the control and forwarding separation [18], as shown in Figure 1.

As one of the available southbound interfaces in SDDC implementation, OpenFlow has been widely supported by the industry. The structure of the OpenFlow switch is

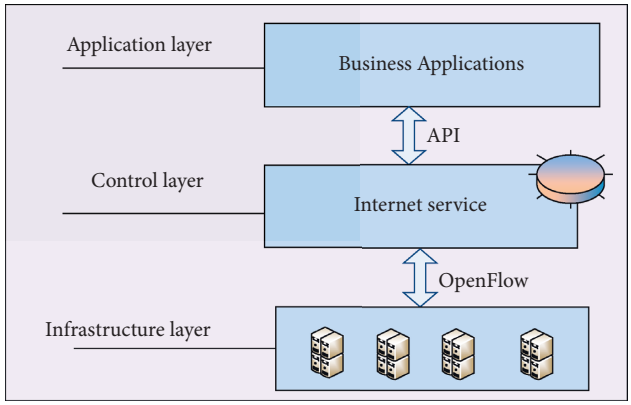

Figure 1: Software-defined network architecture.

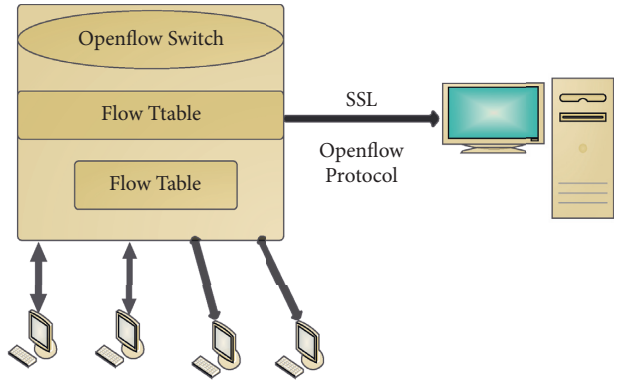

Figure 2: OpenFlow switch structure.

composed of three parts: flow table (flow table), secure channel: road (secure channel 1), and OpenFlow protocol (OpenFlow protocol I), as shown in Figure 2. The flow table is the forwarding rule of the switch, and the secure channel is the connection between the OpenFlow switch and the OpenFlow protocol [23]. The interface of the controller, the switch, and the controller exchange data through the OpenFlow protocol. One controller can be connected to multiple OpenFlow switches.

3.2. Northbound Interface of SDDC Controller. The northbound interface is the interface provided by the controller to other devices for access and management. Although the formulation of the northbound interface scheme and protocol is a hot issue in the current SDDC field, a unified standard has not yet been formed. In general, numerous open-source controllers on the one hand, and the classification framework on the other, promote the development of the current SDDC northbound interface. Different standards organizations have standardized and defined the northbound interface protocol [24]. NETCONF protocol, YANG data model, RESTful protocol, and RESTCONF protocol are the four main northbound interface protocols currently in use. ODL, for instance, uses the YANG data model, while SDDC open-source controllers such as Floodlight and Ryu all use REST-based interfaces. This study partly adopts the REST method when formulating the communication method between the controller and the management system [25].

The concept of representational state transfer (REST) was first proposed by Fielding. The main objective is to understand and evaluate the architecture design of network-based application software while complying with the architectural principle [26]. The RESTful architecture is derived from a powerful, high-performance, and suitable communication architecture. Each URI represents a resource in the RESTful architecture, and the client communicates with the server-side resource using the HTTP protocol. Any text, image, song, or online service that can be represented by a URI is considered a resource. The URI requires the use of a logical identifier instead of a physical identifier. Taking the northbound interface in Ryu as an example, "/stats/flow/1" is a logical identifier, which means obtaining the switch flow table with a depository participant identity (DPID) of 1 instead of using "/stats"/flow/1.html" as such physical identifiers. In the HTTP protocol, GET, POST, PUT, and DELETE are used to describe the operation mode of resources. They correspond to the basic operations of four different types of resources: GET is for getting, POST is for adding or updating, PUT is for updating, and DELETE is for deleting [27]. For example, in Ryu, "GET http://api.ryu.com/stats/flow/1" is used to query or get the switch flow table, then "POST http://api.ryu.com/stats/flow/1" is used, and the flow table information is added to the body of the request to create the flow table. The same is true for PUT and DELETE, but considering that some browsers do not support these two methods, POST is sometimes used instead. For example, in OpenStack, the POST method is used to delete a virtual machine instead of DELETE.

3.3. OpenStack Architecture. The main components of OpenStack include Swift, Keystone, Nova, Neutron, Cinder, and Glance, which correspond to object storage, identity, compute, network, block storage, and image functions, respectively. Figure 3 shows the roles of Horizon and Keystone in the OpenStack stack. Horizon's Dashboard, which serves as the system's browser entry point, has a W-curve-based graphical interface that makes navigation simple [28]. Users can use the dashboard to control the system's computing, storage, and network resources, as well as to operate other components of the system. Keystone controls user information and completes each module's authentication, as shown by the dotted line in the diagram. The solid line in the figure represents the core component of OpenStack that ensures its security.

3.4. Server-Side and Front-End Development Technologies

3.4.1. Node.js and Express. Node is a server-side JavaScript environment based on Google's V8 engine. Both the Node and V8 engines are implemented in C and C++ and are positioned for high performance and low memory consumption. The difference is that the V8 engine mainly serves JavaScript scripts on the browser, while the Node is aimed at the background server process. Its main features are as follows:

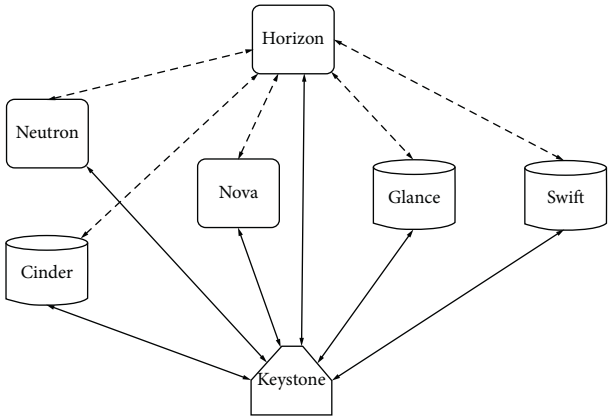

Figure 3: The role of keystone and horizon in OpenStack.

(1) Asynchronous nonblocking single-threaded I/O. Because JavaScript is a single thread, unlike other server-side languages, the Node's processing of concurrent business logic does not depend on multithreading but instead uses an asynchronous I/O event model.

(2) The event front end responds to the user interaction mechanism, and JavaScript uses the event-driven mechanism. When a user interacts with an HTML element, an event is triggered, and the event drives the corresponding function to process information [29]. External requests will be put in the processing queue after the server-side Node introduces the event mechanism. The Node will run the callback code corresponding to the event when an event with a state change is recognized, such as the completion of file reading.

(3) The system is compatible with multi platform nodes, and the project source code currently running on Linux and Ubuntu can be ported and deployed across platforms without any modification. The project in this article is developed on the Windows platform and deployed on the Ubuntu platform without any additional porting effort.

Express is a lean and flexible Node and is a *W*-web program framework. Express, which is based on the Node, provides the fundamental functions required by *W*-curve applications. Express is also an MVC (model view controller) pattern framework, with features such as routing and template engine [30]. In MVC, *M* stands for model, *V* stands for view, and *C* stands for controller. Figure 4 shows the relationship between the three. The view is a template engine such as Jade or EJS, and the controller is the routing mechanism. The model includes multiple Node components, such as a database or a developer-defined module.

3.4.2. Communication between Web Front End and Server. (1) AJAX. The concept of asynchronous JavaScript and XML technology (asynchronous JavaScript, XML, and AJAX) was first proposed by Garrett. AJAX allows web applications to send and receive data from the server asynchronously in the background and dynamically change page content without reloading the entire page by separating the data exchange layer from the presentation layer. In the procedure, XML can be replaced with the JSON data format, which not only reduces the amount of data but also makes it easier to parse with JavaScript [31].

The comparison between AJAX and the traditional model is shown in Figure 5. In the traditional web application model, the server returns the entire web page's HTML and CSS data, and the browser refreshes the entire web page based on the response [32]. In contrast to the traditional model, in the AJAX working model, the browser communicates with the background server by asserting JavaScript scripts. At this time, the server returns XML data, HTML content, or JavaScript data, rather than the entire web page, which is then processed by JavaScript. The script updates the local interface on the web page. AJAX not only reduces the burden on the server but also allows users to avoid experiencing problems caused by frequently refreshing web pages [33].

(2) Socket IO. Web applications that required two-way communication between the client and the server had to depend on HTTP polling to keep the information updated before the advent of online socket. For example, instant messaging and game applications have resulted in significant increases in server and client overhead, as well as increased network traffic [34]. Other improvement schemes such as long polling and streaming essentially use AJAX methods to simulate real-time effects and do not implement real real-time technology. Furthermore, the HTTP protocol was not designed for two-way communication in the first position. The WebSocket protocol is designed to replace HTTP as a bidirectional communication protocol at the transport layer, based on existing infrastructure such as proxying, filtering, and authentication, as shown in Figure 6.

The web socket communication model is like streaming or long-distance communication. The current state is maintained once the connection is established, and data can be pushed from the server to the client at any time thereafter. However, the difference is that although they are all based on TCP, after the connection is established, the data transmission phase of WebSocket does not require the participation of the HTTP protocol. In the long connection mode, the information sent by the server to the client has the complete HTTP header information. The data interaction of WebSocket is complicated only during the first handshake, and the transmission phase is pure WebSocket data flow. WebSocket will inevitably cause compatibility issues on different browsers due to the different support capabilities of browsers for HTML5. Socket and IO were created to aid in the development of programmers. Node is a socket, and the socket is an IO of a Js real-time communication library that loops various protocols, including WebSocket, and makes them available to Node developers for real-time background message push. Switch protocols are automatically based on browser support socket, on the other hand. The disadvantage of IO is that it requires simultaneous use of the front end and the server, and it cannot connect to other WebSocket applications.

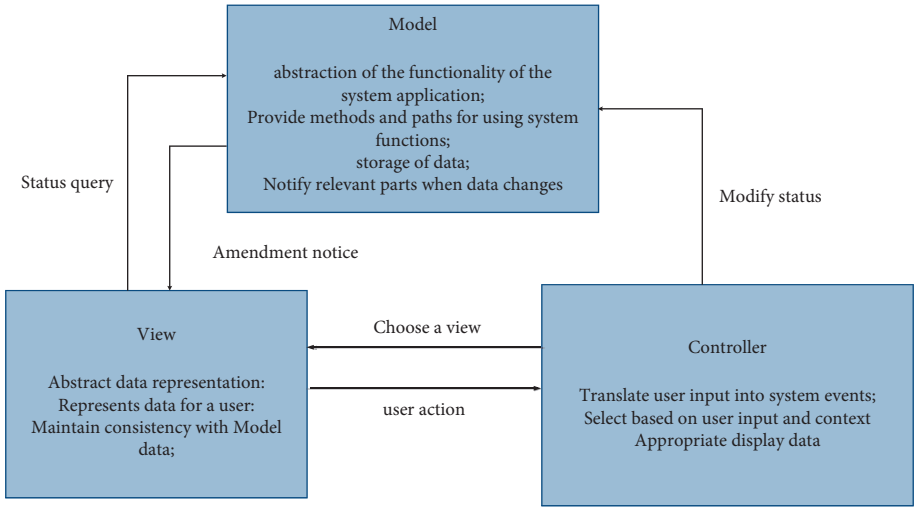

Figure 4: MVC relationship diagram.

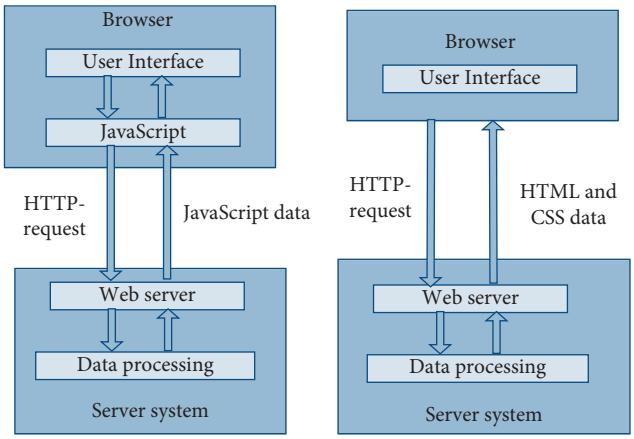

Figure 5: AJAX working model and regular model.

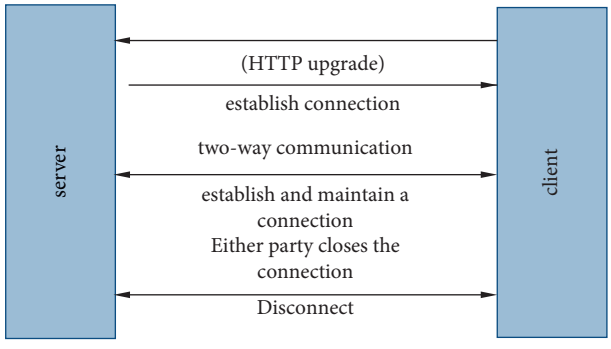

Figure 6: The communication process of WebSocket.

This research focuses on the software-defined data center management system's related technologies. The architecture of SDDC and the controller's northbound interface are explained first, accompanied by the OpenStack module composition and the functions of Horizon and Keystone, as well as their roles in the management system. The Node used in the management system's development is then discussed in detail. In web front-end development, Js, RabbitMQ message queue, AJAX asynchronous communication, and socket are used. The SNMP protocol is used to monitor IO and devices.

4. System Structure Design

The design of the software structure is the basis for the development of the management system, and the realization of business functions needs to be supported by a reasonable software design. This study first analyzes the requirements of the management system of the data center and determines the development language and corresponding modules [35]. Then, the three basic software modules of the management system are expounded in detail: the core scheduling module, the front-end communication module, and the data processing module. The core scheduling module is responsible for user authentication and scheduling of other modules. In response to user requests, the data processing module is responsible for data collection, processing, and persistence. Then, it designs and implements two communication mechanisms, REST interface, and message queue of SDMS system. Finally, it introduces the programming structure of the SDMS system, the component frame of the web front-end, and the structure of the database.

4.1. System Requirements Analysis. Resource management is an important means to ensure the reliable operation of the data center. The management system records the software-defined data center in specific. This improves the data center's availability and reliability while also reducing the data center's monitoring difficulty [36]. Some implemented functions in the software-defined data center do not need to be integrated into the management system because they are implemented in the corresponding SDDC controller.

4.1.1. Resource Management. The management system needs to manage the resources in the data, including network monitoring, link monitoring, configuration management, asset management, and equipment monitoring. Network monitoring includes real-time topology viewing,

message notification, link information, and traffic information. Link monitoring can view the bandwidth statistics of each switch port, and the traffic usage of each core switch, aggregation switch, and edge switch [37]. Configuration management includes the configuration of routing algorithms, server information, cluster information, and switch information. Device monitoring can monitor server performance parameters in the data center.

4.1.2. VDC Mapping Process. In the case of considering only the host failure, the reliability of VDC is defined as the survival rate or reachability rate of VM in the worst case. The actual reliability of VDC_J after mapping should not be less than r_j, as shown in the following formula:

$$r_j \leq \frac{\left|N^j\right| - \max P_j(n)}{\left|N^j\right|}. \quad (1)$$

Among them, $P_j(n)$ represents VDC, the number of VMs allocated to host n. According to formula (1), for VDC, the maximum number of VMs allowed to be placed on a host is shown in the following formula:

$$K = \lfloor \left|N^j\right| \left[1 - r_j\right] \rfloor. \quad (2)$$

The goal is to help InP improve the long-term average return per unit time, which is defined by the following formula:

$$\text{maximize } R = \lim_{t \longrightarrow \infty} \frac{R(t)}{t} = \lim_{t \longrightarrow \infty} \frac{\sum_{j \in \{j \mid t > t_j\}} R_j x_j T_j}{t}. \quad (3)$$

Based on the above requirements, this article selects Node.js, which is compatible with Windows and Linux platforms, as the development language. The Node can not only provide web services as a server but also meet various functional requirements of data center management through the expansion of modules. The database selects the NoSQL database MongoDB to cooperate with Node.js for JSON structure data storage.

4.2. Structural Design of Management System (SDMS). SDDC's three-layer architecture includes an application layer, control layer, and infrastructure layer. The SDMS (software-defined management system, also known as the software-defined management system) system is mainly concerned with the application layer. Ryu, the OpenStack and SDDC controller, is at the control layer, while physical components such as servers and switches are at the infrastructure layer.

The express framework adopts the MVC design pattern, which can make the software structure clearer. However, the information interaction of the management system is relatively complex, and the conventional MVC framework is difficult to meet the requirements of data interaction. Therefore, this study expands and redivides the functions of MVC based on the actual needs and forms a data processing module. The front-end communication module is the structure of the core scheduling module. The detailed design of each module is shown in Figure 7.

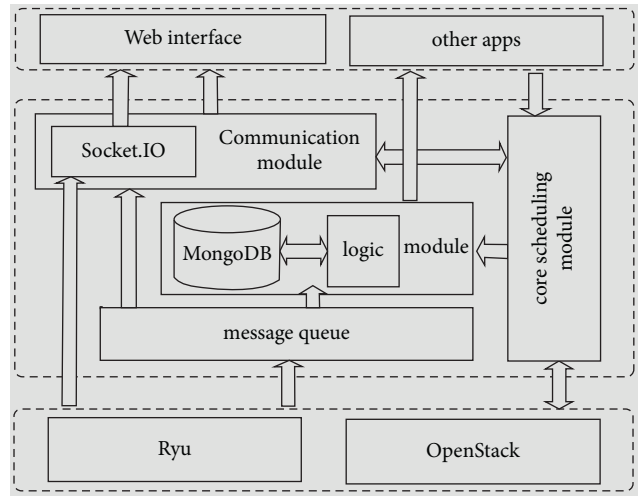

Figure 7: Structural design of SDMS management system.

4.2.1. Data Processing Module. The data processing module, which is at the heart of the system's data processing, reads the data. It is then sent to the front-end communication module for rendering and display, or it is returned to the user directly. It monitors the data from the message queue that needs to be saved and stores it in MongoDB after processing.

4.2.2. Front-End Communication Module. Its component includes data display on the web front-end. The information is embedded in the web template for rendering by the front-end communication module, which then returns it to the user's browser for display. However, some data in the data center are updated frequently and change in real time, such as network node topology, traffic monitoring, and burst error information. The real-time performance of the data will be greatly reduced if all the data are rendered by the front-end communication module, but the SDMS system will be heavily burdened. As a result, WebSocket is used in the view to push data to the front end. WebSocket can realize the function that the server actively pushes messages to the front end. In this study, this function is implemented by the socket of nodes [38, 39]. The IO module is completed: The introduction of WebSocket also requires the occurrence of certain processing capabilities in the web front end. This study mainly uses multiple third-party modules of JavaScript to realize the function of dynamically modifying the content of web pages.

4.2.3. Core Scheduling Module. The core scheduling module authenticates users and manages user permissions as the general system's manager. User requests initiate the operation of data processing modules, front-end communication modules, and OpenStack and Ryu components. The message queue RabbitMQ can distribute the information of the data center control layer to the management system without waiting for the processing result of the data to perform the next task. Data will be correctly received and processed because of RabbitMQ's consumption mechanism.

The entire system is located between the user and the control components of the data center. The SDMS system needs to authenticate any external requests to ensure the data center's security. The user must be notified of the data center response using a mechanism that is convenient for the administrator. However, as a global control, this structure also requires the SDMS system to have a very high load to ensure the normal operation of the system. The mechanism of Node itself can ensure the operating efficiency of the platform, and the introduction of the message queue RabbitMQ can also relieve the load pressure to a certain extent.

5. Design and Implementation of Main Functional Modules

According to the business function of the data center, the SDMS system is divided into multiple functional modules for design and implementation. The web front-end function module, which is based on the bootstrap framework and combined with iQuery and D3, is aimed at the PC side browser. Designing the data interaction between the front end and the backend of the web, the presentation of the page frame, using Js and other third-party JavaScript libraries, and the implementation in combination with specific functions are provided in the following sections. Network monitoring link detection is one of four submodules in the data center resource management function module. Configuration management and asset management are responsible for the front-end display and back-end data management, respectively. User authentication uses the web page login control and the 1 : 3 connection of ordinary data to verify the user's identity in combination. The OpenStack Keystone module implements HTTPS communication on the web front-end to improve security. The device monitoring module collects CPU, memory, and network traffic data through the SNMP agent installed on the server and transmits the above data to the web front-end display through the management system. A complete proxy for OpenStack dashboard is designed and implemented after an analysis of login, HTTP requests, and WebVNC workflow. So, the dashboard can connect to the Internet through the management system for users to access.

The web front-end interface of the management system is mainly designed for PC browsers. In the choice of a front-end framework, this study adopts Bootstrap, combined with jQuery and D3. To achieve various dynamic functions, Js, data tables, and other third-party JavaScript function libraries are used.

5.1. Data Interaction. The template rendering of the view on the backend server-side makes up one part of web front-end rendering. The other part is browser-side real-time rendering and JavaScript modification. The view template engine has the advantage of being able to separate the interface from the data, which improves development efficiency. Here, the default Jade template of the express is used. Real-time rendering is for data messages that are not suitable for rendering with a template engine in the background. The front-end can use the following two methods when acquiring data:

5.1.1. Active Acquisition by the Front End. Actively obtained data include traffic information, configuration information, asset information, and log information. Because the traffic information data are refreshed frequently and need to be updated regularly, the front end can use the polling method. To request data from the specified interface through the HTTP GET method, jQuery is called to modify the DOM node to dynamically add or delete the required content. The other items are controlled by the JavaScript function window. The on-load method is the jQuery function. When the page is loaded, ready () uses the GET method to request data from the corresponding interface. By changing the type parameter, the ajax () function uses the GET, POST, or DELETE method for asynchronous requests using $.

5.1.2. Server-Side Push. Information such as topology changes and error warnings is the main target of this method. The data's most significant feature at present is the randomness, in which it changes. This is incompatible with refreshing the entire page or using regular polling, so the socket is used IO. When the page is loaded, the web front-end will establish a socket with the server. The data pushed by the background are monitored by the IO connection. The page is partially refreshed when the listener receives the data through the use of query. For the drawing of topology, D3 is used in this study. The specific implementation will be described in detail later in conjunction with each module.

5.2. Page Frames. The structure of the overall page includes the left navigation bar, the upper user bar, and the main content. The specific structure is shown in Figure 8. The page template index date provides the code for the title, navigation bar, and user control. After the user controls the system login, the server additionally fills the user's name in the template and sends it to the browser after rendering.

Usually, the active label in the navigation bar is marked with class = "active" in HTML language, and it can also be dynamically implemented using CSS styles or jQuery. Considering that not only the active label needs to be marked but also the partial refresh of the main content according to the user's operation, the jQuery method is adopted.

To make all labels inactive, firstly, jQuery is used to select all list label elements and listen for mouse click events, then the "active" class is removed from all labels. The title triggers the jQuery function after the mouse clicks on the navigation and then adds the label for the current mouse to click the "active" class. Finally, using asynchronous communication, the preset value of the corresponding label as well as the corresponding body content is obtained. The main code is as follows:

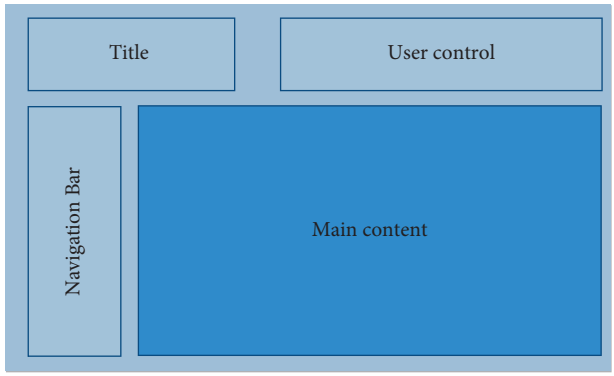

Figure 8: Web page layout.

```
//Select all <li> tags and listen for click events
//Select all <li> tags and listen for click event
S("Ii"). Click (function ()
{
//Select the id currently clicked by the mouse
var target = $(this). attar ("id");
if (target = undefined) return;
//Remove the currently active class tag
S (". active"). remove Class("active");
//Add a class with a value of active to the selected tag.
S(this). add Class("active");
//page path corresponding to the label
var tar = urI[target];
//Get the content of the page through an asynchronous method
S.get (tar, function (data)
{
   //The HTML tag added to the selected content, the id is content
   var content = S("#content");
   //Clear the web page content corresponding to the previous label
   content. Empty (;
   //Add the content corresponding to the current label
   content. Append(data);
})
});
```

6. Conclusion

In the process of strengthening the campus culture's construction, medical colleges and universities are using the network platforms to carry out various educational activities by conducting vivid network education for students through relevant media platforms. According to the school's own data center, relevant professional websites are launched, existing teaching resources are updated, and more effective methods are adopted to allow education to run through the entire construction activities. On the online platform, students can express their opinions and ideas about cultural construction more directly, and teachers can gain a better understanding of students' ideological activities. Various online platforms can help teachers with good moral character to answer questions and solve the problems of students. These platforms cultivate students' ideological and moral aspects, making the Internet at the center of cultural construction. This study first analyzes the current management and control requirements of software-defined data centers (SDDCs) by using the express framework of nodes. The SDMS management system is designed at the application layer of the SDDC architecture to investigate the construction of campus culture at medical colleges and universities. The proposed approach aims to cultivate a cultural environment at institutions across China and around the globe in the future.

Data Availability

The datasets used and analyzed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. J. Ahonen, "On qualitative modelling," *AI & Society*, vol. 8, no. 1, pp. 17–28, 1994.

[2] A. I. Avetisyan, R. Campbell, I. Gupta et al., "Open cirrus: a global cloud computing testbed," *Computer*, vol. 43, no. 4, pp. 35–43, 2010.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: an autonomic QoS policy enforcement framework for software defined networks," in *Proceedings of the 2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, pp. 1–7, Trento, Italy, November 2013.

[4] G. Berndtsson, M. Folkesson, and V. Kulyk, "Subjective quality assessment of video conferences and telemeetings," in *Proceedings of the 19th International Packet Video Workshop (PV)*, pp. 25–30, IEEE, Munich-Garching, Germany, May 2012, Piscataway.

[5] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: a platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, N. Bessis and C. Dobre, Eds., Springer International Publishing, Cham, pp. 169–186, 2014.

[6] C. Chiu, *A Study of Application-Awareness in Software-Defined Data Center Networks*, Louisiana State University, Baton Rouge, Louisiana, 2017.

[7] DMTF, "Software-defined data center (SDDC) definition a white paper from the osddc incubator," pp. 1–22, 2014, https://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0501_1.0.1a.pdf.

[8] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDDC: a software defined datacenter experimental framework," in *Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud*, pp. 189–194, Rome, Italy, August 2015.

[9] Y. Zhang, X. Gong, Y. Hu, W. Wang, and X. Que, "SDNMP: Enabling SDN management using traditional NMS," in *Proceedings of the 2015 IEEE International Conference on Communication Workshop (ICCW)*, pp. 357–362March 2017, London, UK, June 2015.

[10] T. D. Warehouse and T. Common, *IBM Tivoli Netcool/ OMNIbus V7. 2. 1 Delivers Improved Security, Enhanced Process Control, Support for Windows IPv6, and Linux on System z Platform Support*, pp. 1–32, Springer, Wuzhen, 2008.

[11] M. Madan and M. Mathur, "Cloud network management model A novel approach to manage cloud traffic," *International Journal of Cloud Computing: Services and Architecture (IJCCSA)*, vol. 4, 2014.

[12] A. Bianco, R. Birke, F. G. Debele, and L. Giraudo, "SNMP management in a distributed software router architecture," in *Proceedings of the 2011 IEEE International Conference Communications (ICC)*, pp. 1–5, Kyoto, Japan, June 2011.

[13] "Software-defined networking (SDN): Layers and architecture terminology," 2015, http://tools.ietf.org/html/rfc7426.

[14] T.-c. Chiueh, C.-C. Tu, Y.-C. Wang, P.-W. Wang, K.-W. Li, and Y.-M. Huang, "Peregrine: An all-layer-2 container computer network," in *Proceedings of the CLOUD 2012: 2012 IEEE 5th International Conference on Cloud Computing*, pp. 686–693, Honolulu, HI, USA, June 2012.

[15] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, and D. Pinheiro, "Enhancing network management frameworks with SDN-like control," in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 688–691, Ghent, Belgium, May. 2013.

[16] H. Li, X. Que, Y. Hu, X. Gong, and W. Wang, "An autonomic management architecture for sdn-based multi-service network," in *Proceedings of the 2013 IEEE Globecom Workshops (GC Wkshps)*, pp. 830–835, IEEE, Atlanta, GA, USA, Dec. 2013.

[17] J. Li, Z. Zhao, and R. Li, "A machine learning-based intrusion detection system for software-defined 5G network," 2017, https://arxiv.org/abs/1708.04571.

[18] A.-C. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: a software-defined WIreless SEnsor network," 2017, https://arxiv.org/abs/1710.09147.

[19] H. Zhang, Na Liu, K. Long, J. Cheng, V. C. M. Leung, and L. Hanzo, "Energy efficient subchannel and power allocation for software-defined heterogeneous VLC and RF networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 658–670, 2018.

[20] P. Kathiravelu, "Software-defined networking-based enhancements to data quality and QoS in multi-tenanted data center clouds," in *Proceedings of the 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pp. 201–203, Berlin, Germany, April 2016.

[21] G. M. Saridis, S. Peng, Y. Yan et al., "Lightness: a function-virtualizable software defined data center network with all-optical circuit/packet switching," *Journal of Lightwave Technology*, vol. 34, no. 7, pp. 1618–1627, 2016.

[22] U. Paščinski, J. Trnkoczy, V. Stankovski, M. Cigale, and S. Gec, "QoS-aware orchestration of network intensive software utilities within software defined data centres," *Journal of Grid Computing*, vol. 16, no. 1, pp. 85–112, 2018.

[23] S. Xu, X.-W. Wang, and M. Huang, "Software-defined next-generation satellite networks: architecture, challenges, and solutions," *IEEE Access*, vol. 6, pp. 4027–4041, 2018.

[24] P. Giard, G. Sarkis, C. Leroux, C. Thibeault, and W. J. Gross, "Low-latency software polar decoders," *Journal of Signal Processing Systems*, vol. 90, no. 5, pp. 761–775, 2018.

[25] H. Yao, T. Mai, X. Xu, P. Zhang, M. Li, and Y. Liu, "NetworkAI: an intelligent network architecture for self-learning control strategies in software defined networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4319–4327, 2018.

[26] A. Marotta, K. Kondepu, D. Cassioli, C. Antonelli, L. M. Correia, and L. Valcarenghi, "Software defined 5G converged access as a viable techno-economic solution," in *Proceedings of the 2018 Optical Fiber Communication Conference and exposition (OFC)*, Beijing, 2018.

[27] D. Huang, A. Chowdhary, and S. Pisharody, *Software-Defined Networking and Security*, IOPs, Linjiang, 2018.

[28] F. Rebecchi, J. Boite, P.-A. Nardin, M. Bouet, and V. Conan, "DDoS protection with stateful software-defined networking," *International Journal of Network Management*, vol. 29, no. 1, Article ID e2042, 2019.

[29] J. Su, R. Xu, S. M. Yu, B. W. Wang, and J. Wang, "Redundant rule detection for software-defined networking," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 6, 2020.

[30] E. Shenkman, M. Hurt, W. Hogan et al., "OneFlorida clinical research consortium: Linking A clinical and translational science institute with A community-based distributive medical education model," *Academic Medicine*, vol. 93, no. 3, pp. 451–455, 2017.

[31] S. Y. Guraya, M. F. Al-Qahtani, B. Bilal, S. S. Guraya, and H. Almaramhy, "Comparing the extent and pattern of use of social networking sites by medical and non-medical university students: A multi-center study," *Psychology Research and Behavior Management*, vol. 12, pp. 575–584, 2019.

[32] T. Li and Z. Ye, "Campus personal health information reporting system under the covid-19 epidemic: Design and development," *DEStech Transactions on Social Science, Education and Human Science*, vol. 22, 2020.

[33] Z. Zhou, X. Lu, C. Peng et al., "Research on the optimization of the system for the identification, supervision and privacy protection of targeted poverty alleviation for poverty-stricken college students based on big data technology," in *Proceedings of the 2020 International Conference on Computer Engineering and Application (ICCEA)*, Guangzhou, China, March 2020.

[34] X. Zhang, "Design and implementation of university asset management system based on discriminant analysis and decision tree model," in *Proceedings of the 2021 4th International Conference on Information Systems and Computer Aided Education*, Dalian, China, September 2021.

[35] F. Wang and Q. Hu, "Design and implementation of digital platform of academic test in colleges and universities," *Journal of Physics: Conference Series*, vol. 1881, no. 3, Article ID 032055, 2021.

[36] C. Liu, Y. Li, X. Zhao, X. Ren, and M. Tang, "Design and implementation of college students' psychological prediction system based on data mining," in *Proceedings of the 2021 4th International Conference on Information Systems and Computer Aided Education*, Dalian, China, September 2021.

[37] C. Li, D. Ramachandran, K. Rajagopal, S. Jafari, and Y. Liu, "Predicting tipping points in chaotic maps with period-doubling bifurcations," *Complexity*, vol. 2021, Article ID 9927607, 10 pages, 2021.

[38] A. Sadiq, R. A. Khan, B. Mahmood, and M. F. Ashraf, "Assessing institutional preparedness of Pakistani medical schools towards curriculum change using MORC," *Journal of Business and Social Review in Emerging Economies*, vol. 7, no. 3, pp. 687–698, 2021.

[39] X. Wang, J. Li, H. Chen, and J. Li, "Development and implementation of counselor work management information system based on MySQL and data center," in *Proceedings of the 2021 6th International Conference on Communication and Electronics Systems (ICCES)*, Coimbatre, India, July 2021.