Design and Research of SDN Unified Controller in Large Data Center

Jiye Wang¹, a, Hui Liu², b and Cong Yu², c

¹State Grid Corporation of China, Beijing 100031, China
²State Grid Electric Power Research Institute, Nanjing 211000, China

a jiyewang@sgcc.com.cn, b liuhui7@sgepri.sgcc.com.cn, c yucong@sgepri.sgcc.com.cn

Abstract. Firstly, this paper introduces the current situation and requirements of software defined network (SDN) controller in large data center, and then analyzes the mainstream SDN controller system, and then summarizes the development trends of the data center and SDN controller system. Next, this paper mainly introduces the design architecture and implementation technology of the SDN unified controller. Finally, it summarizes the advantages of the SDN unified controller system and discusses the challenges of future work on the project.

1. Introduction

With the rapid development of the data center, the data center has put forward new requirements from the network architecture, resource allocation, fault tolerance, etc. In face of the explosive growth of data traffic, in order to meet the needs of high computing and storage capabilities, the data center should be expandability, robustness, and efficiency to reply for new challenges [1]. However, the current data center has a problem of low utilization of network resources. A large number of idle resources are usually not fully utilized, mainly because the upper-layer cloud operating system only supports simultaneous docking of one SDN controller [2], and computing resources under different SDN controllers cannot be migrated, intercommunication is inconvenient, and resource utilization is uneven. In order to solve the above problems, the cloud operating system is required to support different types of controllers, support diverse management interfaces, and support unified service orchestration between controllers of different vendors, and unifies the multi-vendor SDN resources to support single domain and Cross-domain network resource collaboration. To meet the needs of high-performance control in large data centers.

2. Related researches about controller

With the development of SDN, the industry has introduced a variety of controller applications, such as Onix [3], Ryu [4], OpenDaylight [5] and so on. These controllers, including programming languages and interfaces, have their own characteristics. Several of them will be briefly introduced below.

2.1. Onix Controller of Google

Onix is a controller platform for the B4 WAN network in Google's data center, and it is high scalability. This controller system is divided into three layers: the physical device layer (Switch hardware), the local network control layer (Site Controller), and the global control layer (global). A
Site is a data center. The first layer of hardware switches and the second layer of controllers are deployed at the internal exits of each data center, while the third layer of SDN gateways and servers are in a globally unified control layer.

The second layer is the most complex. This layer does not have only one server in each data center. Instead, there is a server cluster in it, and each server runs a controller. The controller used by Google is based on the distributed Onix Controller. In the server cluster, a switch can be connected to multiple Controllers, but only one of them is in the working state (Master), one controller controls multiple switches, and a program called Paxos [6] is used to conduct Leader elections (election of Master).

Onix Controller is a controller system based on distributed architecture. Google's data center is big and complex network architecture. Its computing resources and storage resources are distributed in different regions. Google expects to optimize its link management by controlling the priority of application data and the burst traffic on the network edge to alleviate the bandwidth pressure. This enables remote users to backup data to ensure data availability and durability; yet to meet large-scale data synchronization and state synchronization between multiple data centers.

2.2. Ryu Controller of Nippon Telegraph & Telephone (NTT)
Ryu is an open source SDN controller developed by the Japanese operator NTT. The Ryu controller is a module-based SDN controller framework written in Python. It has developed a rich API, a clear functional module, and a community plugging integration into OpenStack. Ryu supports all the latest versions of the OpenFlow [7]. In addition, it supports controller protocols such as OF-Config and NETCONF.

The Ryu controller frame is highly modular, and the different component modules are independent of each other, which can perfectly support OpenFlow. Developers can rectify existing components to develop new applications. Moreover, the Ryu open source framework provides plug-ins for integration with the OpenStack platform, which has good application opportunities in the field of cloud computing and will also provide convenience.

2.3. ODL Controller of Linux Foundation
ODL is a Java-based controller whose project architecture is generally consistent with the SDN architecture, but ODL does not include the lowest infrastructure layer, which means that ODL will not involve the work of forwarding layer’s components. Overall, ODL has the following characteristics:

1) The southbound interface not only supports the OpenFlow, but also supports other standardized protocols such as SNMP, PCEP, BGP-LS, and OF-Config, and even allows privatization interfaces.

2) Introducing the service abstraction layer (SAL), which enables the calls between the upper and lower modules to be isolated from each other, shielding the differences of various southbound protocols, and providing consistent services for the upper functional modules.

3) The north provides an open and extensible API, and the user can develop the application by calling the function or the rest interface as needed.

4) ODL has a modular, scalable controller core. The Open Service Gateway Initiative (OSGi) architecture is used to solve the problem of isolation between functional components, enabling flexible loading of code and functions, and supporting plug-in operations such as installation, update, and deletion of run-time services or applications.

5) The YANG tools can directly generate the “skeleton” of business management. The developer only needs to focus on the specific business and design the interface according to the business-driven model tool to realize the business function.

6) ODL has an open source distributed data grid platform that not only enables data storage, look up and monitoring, but more importantly, it enables ODL to support controller clusters.

3. Development trend
The development of SDN technology and controller technology is largely driven by the development of data center networks. Today, people's lives are receiving services provided by data centers every
day, such as information search, online games, and real-time chat. The size of the data center is growing, and the number of servers in the data center is growing rapidly. This section provides a comprehensive analysis of the trends in data center networks and SDN controllers.

3.1. Development of data center fabric topology
In the traditional data center fabric, a hierarchical structure of trees is adopted, which generally has a two-layer or three-layer tree structure. The two-layer structure is the edge layer and the core layer, which can support thousands of switches. As the size of the data center becomes larger, the aggregation layer is added to the two-layer structure, which constitutes the common three-layer structure. However, due to the limitations of the root node, its scalability is poor, and it is difficult to support hundreds of thousands or even millions of server scales. Most of the traffic in the data center is internal traffic. The traffic changes greatly and is difficult to plan. In a three-layer topology, some links are heavily loaded, while others are lighter. The aggregation layer and core layer switches become bottlenecks affecting service performance, the higher the level, the problem is more serious on performance bottleneck. In addition, as the forwarding point of the underlying switch traffic, if the high-level switch occurs a failure, the exchange of data will be seriously affected, the fault tolerance of this structure is poor.

In order to solve the problems faced by the traditional network structure, the industry has adopted a new SDN technology to transform the data center network architecture. In the SDN architecture, the data forwarding strategy is implemented by a software-based controller. The network device is simply a forwarding tool. The logically central controller maintains the state information of the entire network; controllers interact with devices in the network through standard interfaces.

In the SDN architecture, the controller is responsible for the connection and routing of the network. When faced with complex services, only the controller hardware and software need to be upgraded, and the underlying switch can continue to work. This structure uses an iterative approach to spread traffic across multiple available links managed by the controller, avoiding the performance bottleneck of switches at the core layer in traditional network topology.

3.2. Development of distributed SDN controller
With the development of cloud computing, distributed computing and storage, the traffic characteristics of data center fabric have also changed. Including accesses to distributed database, copies of files, and traffic generated by collaborative processing of each module in the data center, and requests outside the same data center need to be processed by multiple servers inside the data center, this also generates a lot of traffic between servers.

In the SDN architecture, the controller controls the forwarding allocation of this traffic through link and route management. However, in the current distributed deployment network, the link bandwidth utilization is very low, because the load balancing method based on static hash is used now. This method is not absolutely balanced, leading to some paths may have high load while others may have low load, in order to avoid that a large amount of traffic is distributed to the same link, causing the link to be overloaded, the data center has to provide much more bandwidth than actually needed, resulting in a low actual link bandwidth utilization. But some links are still inevitably congested, and the device must support a large packet cache, which is too costly. In addition, in order to increase the visibility, stability of network, and simplify management, distributed clustering SDN controller technology research has been greatly developed.

4. Design of SDN unified controller architecture
The topological feature of the SGCC data center is the distributed cluster deployment of network nodes; and the data center traffic presents high dynamic and high concurrency features, so, after drawing on Google's SDN architecture design for its multiple data centers, combining with the SGCC company information system in the scenario of power applications. This paper proposed the design of a SDN unified controller and studied related technologies.
4.1. Research on Distributed Cluster SDN Unified Controller Technology

Design idea based on SDN: As the core of SDN, the controller is responsible for managing and controlling the underlying forwarding device, and providing network resource calls to upper layer service applications. The unified controller cluster constructed in this design is located in the middle control layer and logically represents a controller [8]. The controller has a modular, plug-gable and flexible component module with topology management, device management, and path forwarding. Basic functions such as management support multiple protocols such as Openflow, and multiple controllers work in a cluster mode. Based on the basic architecture of the SDN controller, combined with the functional requirements of the controller distributed cluster system, the entire controller cluster is built based on the AKKA cluster framework.

The overall architecture is shown in the following figure. The whole system is divided into three layers. The upper layer is the application layer, the middle layer controller cluster layer and the lower layer data forwarding layer. The upper application layer is based on the northward interface Restful API access controller provided by the controller. In order to ensure the flexibility of the service request to the controller group and the flexibility of providing service deployment, based on the analysis and research of the traditional load balancing technology [9], Considering the flexible and versatile characteristics of SDN service requests, this design designs and deploys a load balancer based on HTTP protocol between the upper application layer and the distributed controller cluster.

![Overall Architecture Design of SDN Cluster Controller](image)

**Figure 1. Overall Architecture Design of SDN Cluster Controller.**

In the whole system framework, the design extends the horizontal layer of the SDN controller middle layer. Based on the framework of the cluster model and driver provided by the controller, the load balancing function in the cluster access process is realized. The design expands the connection control module and the configuration management module. Through the design of these two modules, the automatic connection between the controller and the switch can be realized, and load balancing within the cluster can be realized. In addition, based on the API provided by the AKKA cluster function [10], an event notification module is designed in the controller cluster. The module is designed to ensure information consistency between the service module and the controller cluster [11].

The link framework between the system function modules is as follows: The connection control module is responsible for establishing a connection with the switch and interacting with various Openflow messages, and parsing and encapsulating various Openflow messages through the SDN controller Openflow south interface plug-in module. Similarly, the basic device management module of the SDN controller can obtain the information and the connection status of the connected device through the connection control module [12], and save the connection. Therefore, the configuration management module can obtain the connection information of the device through the device
management module, and control the connection. The module informs the controller of the change message of the role, and the remaining HTTP load balancer and event notification module are two independent functional modules.

4.2. Relevant Technology of SDN Unified Controller Compatible with ODL and ONOS

In order to rapidly expand the network and computing node servers in the data center, we designed the southward interface component of docking ONOS [13, 14]/ODL controller in the southward interface layer of the SDN unified controller: controller-driver (see Figure 1). Based on the configuration file of the ONOS/ODL controller in the northward interface model, this component converts related northward requests into northward RESTful API adapted to the corresponding SDN controller, and then publishes the API to ONOS/ODL Controller. By updating the configuration file, we can dynamically increase the types of controllers and business modules that system support.

When the VMs in the same network are wanted to sent to different SDN controllers, the unified controller needs additional business arrangement to achieve normal business operations. For related implementation technologies, see the Layer 2 interconnect technology in the next section.

4.3. Relevant technology of Layer 2 interconnection under SDN unified controller

Currently, the management requirements for disaster recovery, computing and storage resources are getting higher and higher in data center. This requires the data center to have the capability of high availability of server clusters and dynamic migration of virtual servers. This not only requires large Layer 2 network access in the data center, but also requires large-scale Layer 2 interworking (Data Center Interconnection, DCI) between data centers [15].

The previous section introduced the use of the northward RESTful API interface. The SDN unified controller implements the coordination and resource invocation of the multi-SDN controller system. Specifically, the following two problems need to be solved. First: the unified controller pairs the received VM. How to forward the online request to the corresponding underlying controller; Second: how to achieve interoperability with VMs under different underlying control of the network.

How does the unified controller forward the received VM online request to the corresponding underlying controller? The solution is as follows:

![Figure 2. Architecture of Unified Controller Cooperating with Different SDN Controllers.](image-url)

(1) The server accesses the TOR switch and sends LLDP packets, and then the packets carry the server host-id;

(2) The device saves the topology information and notifies the topology management module of the SDN unified controller through the SNMP trap;
(3) The SDN unified controller searches the LLDP adjacency list on the TOR that sends the trap through the MIB, obtains the information of the access server, and updates the topology to the database. At the same time, the controller address corresponding to the TOR is obtained, and the manufacturer to which the TOR belongs is recorded. (The correspondence between the address and the manufacturer needs to be configured. For example, by obtaining the private MIB of a controller of some manufacturer, the manufacturer information can be automatically obtained by looking back);

(4) When the user creates a virtual machine access, nova calls the update interface of neutron, passing the host-id and port-id;

(5) The NEUTORN component of cloud operating system receives messages and sends them to SDN unified controller;

(6) The SDN unified controller searches the topology table according to the host-id, and obtains the TOR information of the host connection and the corresponding manufacturer controller address;

(7) The SDN unified controller sent messages to the corresponding controller according to the obtained information.

Aiming at the second problem: How to realize the interoperability of VM under different underlying control with the same network? The solution is as follows: In order to shield the network from different types and unified networking, the industry introduced the concept of Overlay network. Based on the Overlay network, you can configure VLAN hand-off to implement Layer 2 interworking between virtual machines (VMs) in different data centers. Its configuration is as follows:

As shown in Figure 3, BGP EVPN is configured in data center A and data center B to create a VXLAN tunnel to implement communication between VMs in each data center. Leaf 4 and Leaf 5 access DCI-VTEP1 and DCI-VTEP2 through 2-layer sub-interface. DCI-VTEP1 and DCI-VTEP2 are configured with EVPN protocol to create a VXLAN tunnel to implement communication between data centers. Leaf4/Leaf5 decapsulates the VXLAN packets received from the data center and then sends them to the DCI-VTEP. The DCI-VTEP re-encapsulates the received VLAN packets into VXLAN packets and sends them to the peer DCI-VTEP. The end-to-end bearer of the VXLAN tunnel to the data center is ensured, and the communication between the VMs across the data center is ensured.

![Data center interconnection networking diagram.](image)

In summary, by using the vxlan tunnel technology in the data center to interconnect the large Layer 2 networks, the upper-layer cloud operating system can globally invoke network resources under controllers of different vendors, thereby improving the high availability and dynamic migration capability of the data center.

4.4. High Efficiency Technology for Creating Computing Nodes in Multi-SDN Controller System

When a cloud operating system includes multiple computing node servers and multiple controllers are connected to each other, you need to consider the following business scenarios when creating a VM on the cloud operating system:
(1) When creating a VM requires choosing a computing node server, need to consider the available resources of the currently available computing node server, and ensure the maximum use efficiency of the resources;

(2) The client expects that the VM created under the network falls on the computing node server accessed by the same vendor controller, and it is convenient to use the VM to deploy services of Mutual access with high demand;

(3) The client expects that the VM created under the same network falls on the computing node server accessed by different vendor controllers, and the VM deployed under the network is convenient to deploy the service for disaster recovery.

Therefore, when creating a VM on a cloud operating system, different computing nodes need to be selected according to different scenarios to meet the requirements of high efficiency of business. According to different scenarios, we design corresponding algorithms and filter policies to implement selecting computing node server.

(1) When the physical state of the network is local physical, and the current network has not delivered the VM, the computing node server is randomly selected as the computing node server for this time; otherwise, the computing node server that the VM has delivered in the current network is preferentially selected as the a computing node server created by the secondary VM;

(2) When the physical state of the network is disaster physical, the computing node servers under the controllers of different vendors are alternately selected as the computing node servers created by the VMs;

Finally, add the above filter policy to the configuration file which called nova.conf in the cloud operating system, and update the filter policy in scheduler_default_filters configuration item. Then, the efficient creation of computing nodes can be achieved.

5. Conclusion

This paper has studied SDN unified controller, through the clustered SDN controller design based on distributed collaborative service technology, enables the controller to manage across domains, improves the flexibility and scalability of the system, and reduces the failure of a single node. Leading to the embarrassing problem of the entire network, the reliability of the system is quite high. The SDN unified controller supports the current mainstream SDN operating system framework and is compatible with ODL and ONOS controller plane architectures. At the same time, the design realizes the 2-layer interconnection of the data center through the EVPN DCI technology, and shields the type difference of the interconnection network, and realizes the 2-layer inter-working between different data center VMs. Finally, for different scenarios where VM is created, the corresponding strategy is used to select the optimal computing node, which improves the efficiency of data center business.

The scale of the data center will become larger and larger in the future, and the topology, traffic characteristics of the data center will also change constantly. This will lead to the development of the SDN technology. The SDN unified controller designed for this research project should become more intelligent in network resource scheduling in the future, increase data center network bandwidth utilization, and reduce energy consumption, and reduce costs in the face of new business expansion.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No.2017YFB1010000.

References

[1] Thomas N D, Gray K. SDN: SDNs [M]. 2013.

[2] Nunes A, Mendonca M, Nguyen X N, et al. A Survey of SDNing: Past, Present, and Future of Programmable Networks [J]. Communications Surveys & Tutorials IEEE, 2014, 16 (3): 1617-1634.

[3] Koponen T, Casado M, Gude N, et al. Onix: a distributed controller platform for large-scale
production networks [C]. In Proceedings of the 9th USENIX conference on Operating systems design and implementation USENIX Association, 2010:1-6.

[4] Guohui Wang, T S. Eugene Ng, Anees Shaikh, et al. Programming your network at run-time for big data applications. In Proceedings of the first workshop on Hot topics in SDNs, New York: ACM, 2012: 103-108.

[5] Xu Mingguang, Liu Yaping, Deng Wenping. Research and Analysis of Network Controller OpenDaylight [J]. Computer Science, 2015, 42 (sl): 1-4.

[6] Pin-An Yang, Research and Design of HDFS High Availability Based on Paxos Algorithms [D]. South China University of Technology, 2012.

[7] Tootoonchian A, Ganjali Y. HyperFlow: A distributed controller plane for OpenFlow. In Proceedings of the 2010 internet network management conference on Research on enterprise, networking. USENIX Association, 2010: 30-33.

[8] Thomas D, Nadeau, Ken Gay. SDN: SDN [M]. OREILLY, 2013: 8-20.

[9] Vanbever L, Reich J, Benson T, et al. HotSwap: Correct and efficient controller upgrades for SDNs. In: Proc. of the ACM SIGCOMM Workshop on Hot SDN. 2013: 133-137.

[10] R. Kuhn, J. Boner and P. Tiinder. Typed akka actors. Private communication, 2012: 21-34.

[11] Xue Jun, Li Zengzhi, Wang Yungeng. Development of Load Balancing Technology [J]. Minicomputer System, 2003, 24 (12): 2100-2103.

[12] Marc Mendonca, Bruno Astuto A, Nunes, et al. A Survey of SDNs: Past, Present, and Future of Programmable Network. 2013: 66-73.

[13] Meral Shiraziotir, YingZhang, Neda Beheshti, et al. OpenFlow and Multi-layer Extensions: Overview and Next Steps. European Workshop on SDN, 2012: 14-25.

[14] Aris Cabyadi, Risidianio, Eveving Mulyma. Implementation and Analysis of Control and Forwarding plane for SDN. 7th International Conference on Telecommunication Systems, Services, and Application (TSSA), 2012.

[15] Yu Y, Zhang J, Zhao Y, et al. Field Demonstration of Multi-Domain Software Defined Transport Networking With Multi-Controller Collaboration for Data Center Interconnection [Invited] [J]. Journal of Optical Communications and Networking, 2015, 7 (2): A301-A308.