THE DESIGNING OF CLOUD INFRASTRUCTURE CONSISTING OF GEOGRAPHICALLY DISTRIBUTED DATA CENTERS

P.V. Fedchenkov [1], S.E. Khoruzhnikov [1], N.Y. Samokhin [1], A.Y. Shevel [1,2,a]

[1] *Department of Network and Cloud Technologies, ITMO University, St.-Petersburg, 197101, Russia*

[2] *National Research Centre "Kurchatov Institute" PETERSBURG NUCLEAR PHYSICS INSTITUTE, Gatchina, 188300, Russia*

E-mail: [a] shevel_a_y@niuitmo.ru

Contemporary tendency of use the cloud architectures is quite common. Many tests show that the price for using a commercial cloud is feasible for most of applications including High Energy Physics. At the same time, many users are still using the large data centers (DC). The using just one data center (DC) is not completely sufficient in several aspects. First is lack of isolation, and second is lack of elasticity. Lack of isolation means situation when many users share one DC and violate SLA of each other from time to time. Lack of elasticity means situation when many users while sharing one DC cannot change SLA at required level. In context of service availability, one DC for many users is also not perfect solution as technical, nature, and social incidents might affect the DC. As a result, cloud solution based on geographically distributed DCs looks preferable. The running project addressed to implementation of such the cloud is described in the paper.

Keywords: clouds, SDN, NFV, distributed, QKD, software defined, linux

1. Introduction

The number of DCs in the world is steadily growing. There are many attempts in many areas to move applications from concrete DC to public cloud. Using multiple geographically distributed DCs looks preferable from reliability and resource availability points of view. The network connection between DCs included into the cloud should correspond with a SLA for the cloud: if a data transfer speed is serious priority, the data links might have a bandwidth of 100 Gbit or more. In order to implement the cloud, the special management system is required.

2. Main architecture approaches and technical requirements

The number of modern approaches: Software Defined Storage (SDS), Software defined Network (SDN), Network Function Virtualization (NFV), Infrastructure as Code (IaC) [1,2] are proved in many developments. Technical requirements are necessary for describing the future system features as well. Among technical requirements for system management of Scalable Geographically Distributed DCs (SGDD), the following is considered as most important:

- Data links
  - Hardware security.
  - Data compression/decompression.
  - Data encryption/decryption.
- Service reliability
  - Steady system functioning.
  - Gradual degradation with hardware or software malfunction.
- Monitoring of physical and virtual infrastructure.
- Data storage features, including the long-term data storage with total volume up to 1 EB.
- SLA support.
- Data Center Infrastructure Management (DCIM).
- Automatic deployment of the system management

System Management Architecture of SGDD is implemented in form of separate program agents communicating with each other by the special designed protocol. System logical structure is shown in fig. 1. In the paper, the technical term ***program agent or agent*** means program daemon which is running in isolated operating environment. Any request to the agent and answer form the agent could be performed with appropriate protocol. Such implementation gives many advantages:

- independence for agent development: developer can use any programming language and/or library or versions of it;
- independence during system production: if one agent is down, other agents might continue to work, i.e. system is just partially degraded;
- simplification of horizontal scalability: it is possible to set up an additional agent instance with same functions in order to improve service bandwidth and to provide high availability and load balancing.

The system functioning principles are shown in fig. 1 as well. The user (client) can be signed up at the portal of registration and the accounting agent. After registration user is able to create a subscription where required virtual facilities are described. Such facilities may include virtual machines, virtual storages, virtual data links. All facilities may be accompanied by specific SLAs. Initially, all requests are headed to orchestration agent which does preliminary check of available resources. After that, user subscription request is routed to appropriate service agent: VM allocation agent, virtual data links allocation agent, storage allocation agent. Those service agents perform the creation of required virtual objects and send back the technical information how to access created virtual objects to the user.

For example, if creation of virtual storage with specific SLA is required, the storage allocation agent does create the CEPH cluster with required SLA including special gateway, which allows the user to read and write the data to and from virtual storage.

Virtual data links allocation agent does create new virtual links with specific SLA. SLA for virtual links may include encryption/decryption, compression/decompression, etc. The implementation of virtual data links is shown in fig. 2. The agent issues some commands for RYU SDN controller [9] which at the same time does control an Open Virtual Switch (OVS) for choosing an appropriate network segment. All network segments must be created before the system is in operation. Practically data links creation is performed in form of the choice of appropriate network segment.

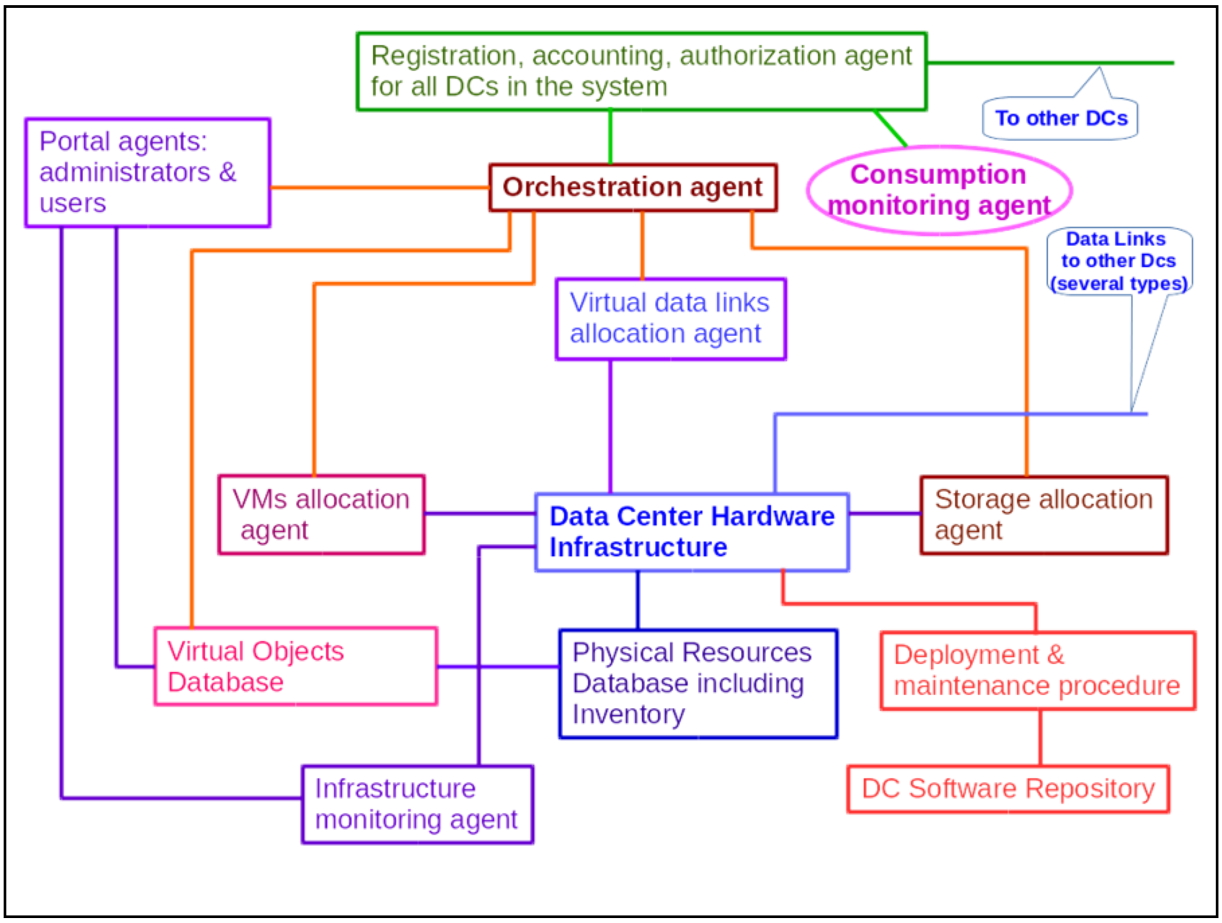
Registration, accounting, authorization agent for all DCs in the system
To other DCs
Portal agents: administrators & users
Orchestration agent
Consumption monitoring agent
Data Links to other Dcs (several types)
Virtual data links allocation agent
VMs allocation agent
Data Center Hardware Infrastructure
Storage allocation agent
Virtual Objects Database
Physical Resources Database including Inventory
Deployment & maintenance procedure
Infrastructure monitoring agent
DC Software Repository

Figure 2. Logical structure of SGDD

Virtual objects of type VM are created in similar ways.

3. Technical solutions for system management of SGDD

Main technical solutions within technical designing are:

- The system is based on Free and Open Source Software (FOSS) components.
- Operating System: Naulinux (compatible with RedHat, Scientific Linux, CentOS) [3].
- Computing, portals: Openstack [4].
- Data Storage - Software Defined Storage (SDS): CEPH [5].

- Monitoring of physical and virtual infrastructure, billing data: Zabbix [6].
- Visualization: Grafana [7]/Kibana [8].
- Database keeping the hardware information, virtual objects information, detailed requests and answers information: PostgreSQL.
- Communication within SGDD: RabbitMQ with special protocol developed.
- Data transfer security: symmetric cryptography with OpenSSL and Quantum Key Distribution (QKD) procedure [9].
- Independence of external components changes (versions, polices, etc): own versions of Linux distribution (Naulinux) + code repository for all developed components.
- Each type of service must be feasible for accounting/billing purposes.

Current debugging of the system is performed providing the following hardware test configuration:

- Several micro DCs consisting of servers [6 + 2 + 2] type HPE DL380 Gen10 8LFF CTO (each server has 2x Intel Zeon Gold 6130, main memory: 128 GB, HD storage 32 TB).
- Several switches: HP FF 5700-32XGT-8XG-2QSFP+ Switch.
- Other communication equipment.

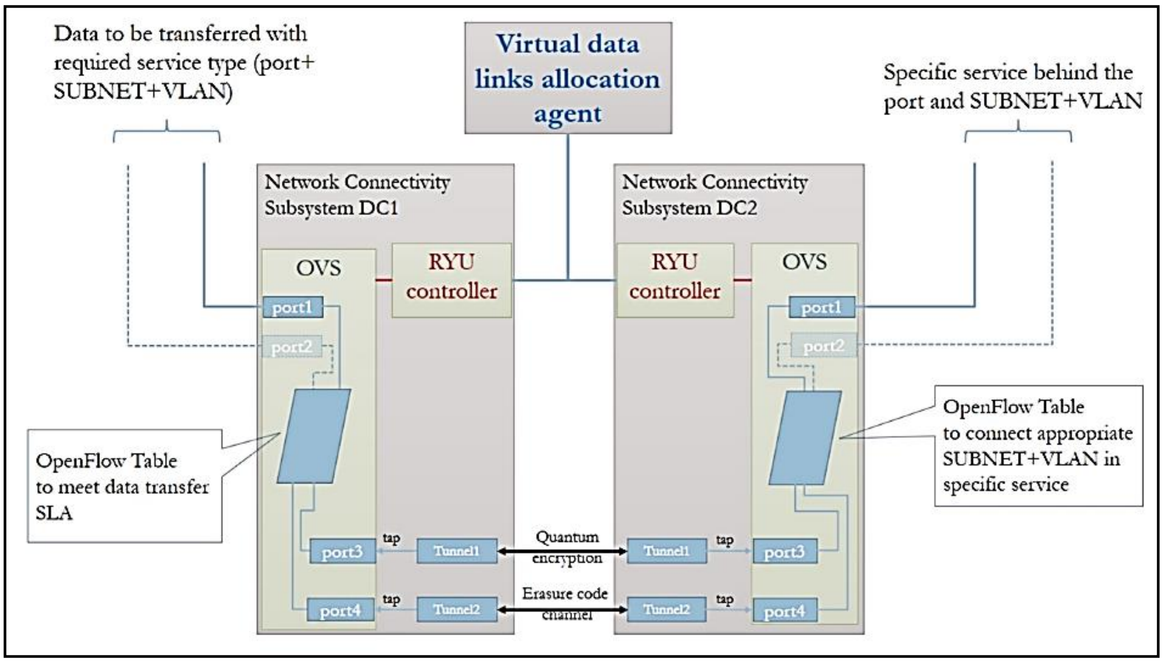
Data to be transferred with required service type (port+ SUBNET+VLAN)
Virtual data links allocation agent
Specific service behind the port and SUBNET+VLAN
Network Connectivity Subsystem DC1
Network Connectivity Subsystem DC2
OVS
RYU controller
RYU controller
OVS
port1
port2
port1
port2
OpenFlow Table to meet data transfer SLA
OpenFlow Table to connect appropriate SUBNET+VLAN in specific service
port3
tap
Tunnel1
Quantum encryption
Tunnel1
tap
port3
port4
tap
Tunnel2
Erasure code channel
Tunnel2
tap
port4

Figure 3. Schematic view of virtual data links

Automated deployment of the system has been implemented with SALTstack [10] and special scripts developed.

4. Conclusion

Main features of the project are: the use of FOSS components, system architecture based on the independent agents, QKD procedure, own code repository, automated deployment of the system [11]. An engineering infrastructure monitoring might permit to use data centers as ***dark*** data centers. Here ***dark*** data center means absence of permanent maintaining staff on the data center. It may be some technical persons are visiting such the data center to fix something by demand. All features together provide more security and reliability. The project is in active debugging stage now. The system is being prepared for a preproduction stage by first half of the 2019.

5. Acknowledgement

The development is performed in accordance with the contract № 03.G25.31.0229 “Development of new technological components for management systems of geographically distributed Data Centers”

References

[1] Matej Artac et al // Infrastructure-as-Code for Data-Intensive Architectures: A Model-Driven Development Approach // 2018 IEEE International Conference on Software Architecture (ICSA) 30 April-4 May 2018 // DOI: 10.1109/ICSA.2018.00025.

[2] Kief Morris // Infrastructure as Code: Managing Servers in the Cloud // ISBN-13: 978-1491924358, ISBN 9781491924358 // O’Reilly Media // 2016.

[3] Naulinux. Available at: http://naulinux.ru/ (accessed 01 November 2018).

[4] OpenStack. Available at: http://openstack.org/ (accessed 01 November 2018).

[5] Wong M.-T. et al // Ceph as WAN Filesystem – Performance and Feasibility Study through Simulation // Network Research WorkshopProceedings of the Asia-Pacific Advanced Network 2014 v. 38, p. 1-11. http://dx.doi.org/10.7125/APAN.38.1 ISSN 2227-3026.

[6] Zabbix. Available at: http://zabbix.org/wiki/Main_Page (accessed 01 November 2018).

[7] Grafana. Available at: https://grafana.com/ (accessed 01 November 2018).

[8] Kibana. Available at: https://www.elastic.co/products/kibana (accessed 01 November 2018).

[9] G. P. Miroshnichenko, A. V. Kozubov, A. A. Gaidash, A. V. Gleim, and D. B. Horoshko, “Security of subcarrier wave quantum key distribution against the collective beam-splitting attack” Opt. Express 26, 11292-11308 (2018).

[10] SaltStack. Available at: https://saltstack.com/community/ (accessed 01 November 2018).

[11] P.V. Fedchenkov et al // APPROACHES TO THE AUTOMATED DEPLOYMENT OF THE CLOUD INFRASTRUCTURE OF GEOGRAPHICALLY DISTRIBUTED DATA CENTERS // these proceedings.