Research Article

Consideration of Docker based network deployment for a data center: GSDC in Korea

Jin Kim[1] and Jinsoo Park*[2]

[1] Global Science Experimental Data Hub Center, Korea Institute of Science and Technology Information, 245 Daehak-ro Yuseong-gu Daejeon, 34141, Korea
[2] Dept. of Management Information Systems, Yong In University, 134 Yongindaehak-ro Cheoin-gu Yongin, 17092, Korea
*Corresponding author. Tel.: +00-0000-0000; Email address:

Article History:Received:11 november 2020; Accepted: 27 December 2020; Published online: 5 April 2021

Abstract: For various operational reasons, this study presents a design and testbed environment for applying Docker-based system to a data center in Korea. We describe the design process of Docker based network. First, abstract structure is presented. Second, two conceptual routing models are provided. Third, the network plugin selection is stated. Then, we build Docker based system on a testbed environment as a pre-study for deployment to the entire system. There are plenty of factors to consider for the overall deployment. Representative factors include the routing method inside the server, the number of containers in a server, service arrangement by container, and response to equipment replacement. To evaluate those factors, various study should be performed with key system parameters. The proposed testbed environment can be utilized to obtain key system parameters. Applying Docker system to a data center can be remarkably efficient in terms of operations. The workload of operator will decrease, and scalability and flexibility will increase.

Keywords: Docker network, Network switch, Data center, Routing table, Network plugin.

1. Introduction

The research paradigm in the recent field of basic science has been changed to the data-intensive type. Due to the proliferation of large-scale experimental data, the data center is required for sharing and managing the data. Under these circumstances, Korea Institute of Science and Technology Information (KISTI) have operated Global Science Experimental Data Hub Center (GSDC) as the 11th CERN WLCG Tier 1 Center since 2010 [1]. GSDC stores mass experimental data generated from the Large Hadron Collider (LHC) and basic science data such as structural biology. Since GSDC also provides computing environment, it has supported a variety of research fields in South Korea.

The data center consists of three computing elements: server, storage, and network. On the server side, an operator who manages multiple servers increases a lot of workload under the following circumstances: 1) when a new system is added while operating a system in the same environment, 2) when it is required to change the environment of multiple systems at the same time, 3) system rollback due to errors and failures in the new environment, 4) detailed system version control, 5) repetitive setup. In terms of storage and network, the environments are changing rapidly as the scale of data remarkably increases. Therefore, KISTI intends to introduce a Docker based system in GSDC to flexibly cope with the rapidly changing service environment. In terms of service operation, Docker has the various advantages such as convenience of system construction, system expandability, solidity and flexibility on operation, system test, error detection, and so on. Docker is a set of OS (operating system) level virtualization platform that automates the deployment of applications into software containers [2-3]. Recently, the scope of its application becomes remarkably wide. Accordingly, various researches of Docker based systems have been performed. In the big data platform, Docker can support the construction of distributed computing systems [4]. The network communication is the major focus of numerous studies on Docker based systems [5-9].

In the deployment of Docker based system, the network structure is one of the most important factors for smooth communications between the containers. Therefore, we propose an abstract architecture of Docker based network for GSDC and two conceptual models according to the physical network aspect.

This paper is organized as follows. Section 2 presents the abstract design of Docker based network for GSDC, two conceptual switching models, and the network plugin. Section 3 describes some issues and problems that may occur in the deployment. Finally, section 4 draws our conclusions and future works.

2. System Design

In this section, we describe the Docker network design process for GSDC. After explaining the background of applying Docker to GSDC, the system design is developed in detail. The abstract structure of Docker network for GSDC is described and two conceptual models according to the routing perspective are proposed. Finally, we consider the network plugin that controls the communication between containers in a cluster on Docker.

2.1 Backgrounds

2.1.1 General Benefits of docker

Docker is an open source virtualization platform based on Linux container technology. Since Docker can implement virtualization by separating only the necessary parts. The virtualization part can be more

***Corresponding author:**

concentrated and necessary service expansion may be simple. In particular, Docker has a number of advantages when operating servers that are difficult to manage. It can also be useful to developers. In general, Docker has the following advantages. First, it is easy and fast to build an execution environment. By deploying the initial environment as a Docker container, we can force the execution environment. Second, Docker is lighter and faster than a virtual machine. Since the virtual machine has to restart the OS, the start itself is relatively heavy. However, Docker containers start up quickly due to their light weight. Third, there is a reduction in hardware resources. The container size is small, and multiple containers can be run on one physical server. Fourth, it provides a strong sharing environment. Through Docker Hub, we can get proven results from developers around the world. Finally, the development environment is simple to deploy. This part is also a big advantage in terms of service operation. However, as with the other great technologies, there are some disadvantages and limitations in using Docker. Docker has a lot of overhead in the initial stage of developments. Therefore, it may not be suitable for small systems or projects. In addition, since Docker is Linux-friendly, we can use it properly in a Linux based OS to take advantages of the benefits.

2.1.2 Docker Usage for Large Scale of Computing Farm

There are chronic problems with way traditional data centers operate the server farms. At first, there can be the problem between server farm operator and user applications. In operating a server farm, respective servers are established according to the time or version. However, applications under the respective servers require the same environment. In this case, the snowflake server problem is remarkably difficult for the server farm operator to deal with. Since Docker produces and distributes images, it can address this issue. The second problem is the horizontal expansion of the environment in which the same application runs. There are various ways to configure and manage servers in code in order to operate a server farm of various servers. For example, specific tools can be applied such as tmux, vagrant, chef, puppet, ancible, etc. Docker employs the Docker file as a server operation record and the Docker image as a point of execution. Although the systems are deployed at respective time points, the applications on the systems should run at the same time. Therefore, Docker can simply solve the problem at the horizontal expansion. In addition, Docker provides robustness and flexibility when constructing server farms in code, and guarantees great convenience in scalability. Along with these reasons, introducing the Docker system to GSDC has the following advantages. It is convenient to build through server coding. The system scalability is excellent and operational robustness and flexibility can be guaranteed. System test and error correction become simple. In addition, since hardware is not virtualized, performance is improved for a variety of tasks performed by the OS such as memory access, networking, and so on.

2.2 Docker Deployment in Network Aspect

The configuration of the Docker based network for GSDC proposed in this paper is as follows. The network is a spine-leaf structure with two tiers, and there are 10 physical servers in the respective racks. The servers in a rack are connected with two leaf switches for redundancy. Two containers are generated on a physical server. Basically, although it is a two-tier structure, the server and storage zones are logically distinct. Figure 1 illustrates the abstract structure of Docker based network for GSDC. In Figure 1, FW means firewall and SW implies switch. Note that Figure 1 does not show the physical appearance but a logical structure. Therefore, there can be some differences between Figure 1 and the system description. The top of rack (ToR) switches are connected to the core switch used as a backbone, and respective ToR switch is connected to ten servers in a rack.

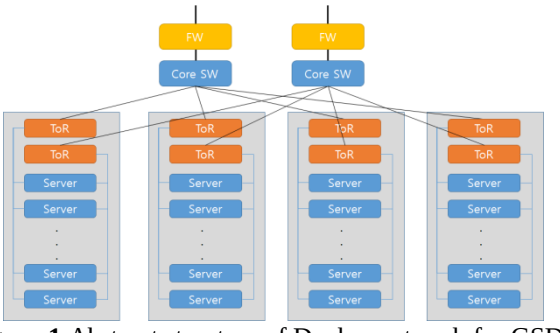

Figure 1 Abstract structure of Docker network for GSDC

Figure 2 shows the abstract structure of a physical server. In Figure 2, N containers are created in a physical server. Since a container has its own IP address, one container becomes an independent network host. In other words, respective container becomes a destination in the Docker network. Therefore, we need a virtual switch (V-SW in Figure 2) to identify the route to containers in a server. A routing table containing entries for respective containers is also required. As shown in Figure 2, the containers can be aggregated to one or more containers.

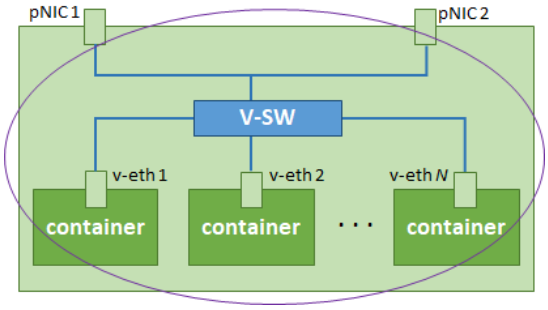

Figure 2 Abstract structure of a physical server

2.3 Two Conceptual Routing Models

The routing model can be divided into two types according to the network layer linked between core switch and ToR switch. There are two conceptual models; layer-2 (L2) and layer-3 (L3) models. As is well known, L2 is the data link layer and L3 is the network layer. For host identification, L2 usually employs MAC address and L3 uses IP address. For routing table in the virtual switch, every server in a network domain must have the information of all entries that exist in the domain. The servers connected to the same ToR switch also need to have the route of everyone.

The L2 model consists of an L2 connection between the backbone switch and the ToR switch, and uses MAC address to reach the destination. Since there are no routings between them, we can aggregate the ToR switches to get a double bandwidth using Link Aggregation Control Protocol (LACP) in a Rack. Therefore, the number of routing entries for respective server is expressed as $(s \times c \times r - 1)$ where s is the number of servers, c is the number of containers, r is the number of racks. The advantage of L2 model is that the LACP technology increases bandwidth and simplifies deployment. Under the L2 model environment, however, the server must broadcast when it is first started.

The L3 model has L3 connection between the backbone switch and the ToR switch, and identifies the destination with IP address. There is no aggregation between the physical network interface controller (NIC) of the server and the ToR switches. The number of routing entries for respective server is calculated by $(s \times c \times r \times l - 1)$ where l is the number of server links to connect network device. In the L3 model, by using the routing table, the broadcast packets in the network inside the data center can be reduced.

2.4 Network Plugin

In Docker, Kubernetes is a system that manages the containerized applications. This system provides a platform for automating deployment, scaling, and operation of application containers between hosts in multiple clusters. Kubernetes ecosystems have been developed by a number of open sources developers due to its advantages. Accordingly, they have been changed rapidly and complexly. For communication between containers in a cluster running on Docker, a network plugin should be chosen from among a variety of network plugins such as Calico, Ciliium, Flannel, and so on. Figure 3 shows a comparison table between the network plugins investigated by Kubedex [10]. Figure 3 is a table cited from [10] as it is. Accordingly, we can check the original and recent table in [10].

	Flannel	Calico	Weave Net	Cilium	Kube Router	Romana	Contiv
Company	Redhat	Tigera Inc	WeaveWorks	Covalent	CloudNative Labs	Pani Networks Inc	Cisco
Latest Stable Version	0.10.0	3.3.1	2.5.0	1.3.0	0.2.1	2.0.2	1.2
Start Date	July 2014	July 2014	August 2014	December 2015	April 2017	November 2015	December 2014
Language	Go	Go	Go	Go	Go	Bash	Go
Minimum OS Version		RHEL 7, Centos 7, Ubuntu 16.04, Debian 8	Linux Kernel > 3.8	Linux Kernel > 4.9			CentOS 7, Ubuntu 16.04
Minimum Kubernetes Version	1.6	1.6	1.6	1.8	1.6	1.8	1.8
IP Version	ipv4	ipv4, ipv6	ipv4	ipv4, ipv6	ipv4	ipv4	ipv4, ipv6
Open Source	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Encryption	Experimental	No	NaCl library	No	No	No	No
Network policy	No	Ingress, Egress	Ingress, Egress	Ingress, Egress	Ingress, Egress	Ingress, Egress	Ingress, Egress
Recommended Max Nodes		5000	500				
Default Network Model	Layer 2 VXLAN	Layer 3	Layer 2 VXLAN	Layer 2 by default. Layer 3 optional.	Layer 3	Layer 3	Layer 2, Layer 3 or ACI options
Layer 2 Encapsulation	VXLAN	-	VXLan	VXLan, Geneve	-	-	VXLAN
Layer 3 Routing	iptables	iptables, kubeproxy	iptables, kubeproxy	BPF, kubeproxy	IPVS, iptables, ipsets	iptables	iptables
Layer 3 Encapsulation	-	IPIP (optional)	Sleeve (fallback)	-	IPVS/LVS DR mode, GRE/IPIP	-	VLAN
Layer 4 Route Distribution	-	BGP	-	-	BGP	BGP, OSPF	BGP
vnic per container	no	yes	yes	yes	yes	no	yes
Multicast Suport	no	no	yes	no	no	no	yes
Subnet Per	Host	One or more of Cluster / Host / Namespace / Deployment	Cluster	Host	Host	Host	Overlapping IP pools
Isolation	cidr	label, host, cidr, network sets	cidr, network	label, services, entities, cidr, dns	cidr	cidr	label, cidr
Load Balancing	no	yes	yes	yes	yes	yes	yes
Multi Cluster Routing	no	yes	yes	yes (see comment)	yes	yes	yes
Partially Connected Networks	no	no	yes	no	no	no	no
IP Overlap Support	no	no	no	no	no	no	yes
Name Service	no	no	yes	no	no	no	no
Datastore	kubernetes CRDs or etcdv3	kubernetes CRDs, or etcdv3	file inside pods	kubernetes etcd	kubernetes etcd	kubernetes etcd	kubernetes etcd, etcd or consul
Paid Support	No	Yes	Yes	No	no	Yes	No
Docs	https://coreos.com/flannel/docs/latest/	https://docs.projectcalico.org/v3.3/introduction/	https://www.weave.works/docs/net/latest/overview/	http://docs.cilium.io/en/v1.3/	https://github.com/cloudnativelabs/kube-router	https://romana.readthedocs.io/en/latest/	http://contiv.github.io/documents/
Platforms	Linux, Windows	Linux, Windows	Linux	Linux	Linux	Linux	Linux

Figure 3 Kubernetes networks comparison

3. Deployment and Discussion

3.1 Deployment

It is not a simple task to deploy the Docker system into the entire GSDC in KISTI. As mentioned previously, it is strictly complicated to configure the initial environment of the Docker in a huge system. In particular, current GSDC network structure has a terrible complexity, and there are a number of servers in operation. Therefore, there are plenty of factors to consider for the overall deployment. The detailed factors to consider for deployment will be discussed later in this section. In this paper, we introduce a testbed environment that considers the characteristics of the current GSDC network. Through operating this testbed environment system, the factor values suitable for GSDC operation will be determined and reflected into the entire system later.

Figure 4 illustrates an instance of the proposed Docker network for GSDC. The core switch described as a backbone switch in Figure 4, is connected by two 100Gbps uplinks, and is connected by 80Gbps link to respective ToR (top of rack) switch. The connection between a server and a ToR is 25Gbps. For internal traffic management due to the increase of computing nodes, we use the L3 model. Respective rack is equipped with 10 physical servers and respective server generates two containers. As described in the previous section, although it is a two-tier structure basically, the server and storage zones are logically distinct. The containers will be assigned 172 bandwidth IPs via DHCP (Dynamic Host Configuration Protocol) server, and use 4 bytes for autonomous system number (ASN). For routing entry in the virtual switch, we assign a last five-digit number of 4 bytes to identify respective container. The fifth digit is the containment number, the third and fourth digits are the rack numbers. The first two digits imply the actual location of the server. Through this information, the container exchanges routing information with the ToR switch through the external Border Gateway Protocol (eBGP) [11]. Considering large container networking, calico [12-13] with near native network performance is selected as the network plugin.

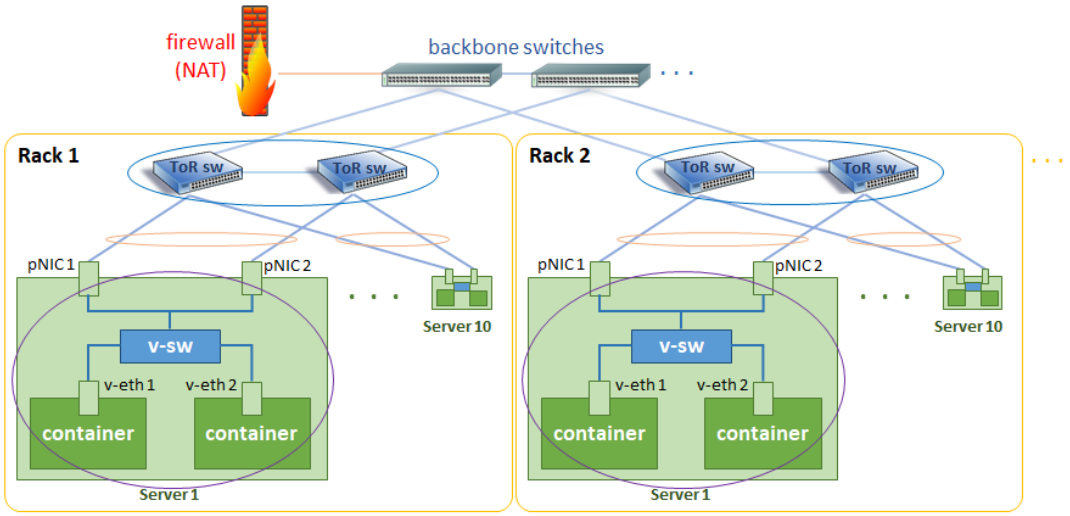

Figure 4 An instance of proposed GSDC Docker network

3.2 Issues and Discussions

As stated previously, there are a number of factors to consider when we deploy the Docker based system into the entire GSDC. In this paper, we discuss some of the representative factors.

In GSDC, a static IP address must be assigned to the server, and the server must communicate irregularly with the outside. Therefore, it is necessary to manage the location and IP address where containers are created in the Docker environment. The network generates an internal routing table for communication between containers. For special reason, all the bits in the IP address becomes subnet identifier for routing (CIDR expression: /32). Accordingly, the number of routing entries in a routing table is identical to the number of containers as described in section 2.3.

Determining the number of containers is one of the most important factors to consider. In general, one server with twice the performance is more powerful than two servers with a single performance. Accordingly, when one server is divided into several containers, it can be considered that there is a performance penalty. In the Docker system, however, the container has the advantage of possibility to provide more than one type of service in a physical server. This situation is expected to be frequent due to the nature of GSDC. In addition, if the workload of respective container server is not heavy, it is possible to improve performance by providing different types of services for respective container. For example, when one container is responsible for I/O and

another container provides only computing resources, we can get a performance boost. On the other hand, increase of the number of containers may degrade the performance. Therefore, it is necessary to find the optimal number of containers in consideration of various environmental variables. The factor to be considered simultaneously with the number of containers is the arrangement of services by container. Service placement by container will be an important factor in determining system performance.

Finally, we should consider replacing the servers or network equipment. KISTI is actually planning to replace the network equipment in GSDC sooner or later. The rack will be equipped with 20 servers, and the ToR will have 40 ports. Therefore, it is necessary to change the physical structure without any logical change. This issue is strictly important and must be considered.

Through simulation studies, numerous problems with Docker based system deployment into GSDC can be solved. However, to perform the simulation, it is essential to find the input parameters such as user arrivals, service times, and so on. Therefore, to obtain these parameters, the proposed testbed system must be actively utilized. This can be another significance of this paper.

4. Conclusions

This paper proposed an abstract design of Docker based network for GSDC. First, we explained the background of the Docker system introduction and presented an abstract Docker based network model for GSDC. In the network, we suggested two conceptual layer models according to the physical network aspect. For respective models, the numbers of routing entries were also calculated. Next, we presented the necessary parts to determine the network plugin. Although it is not yet possible to deploy into the entire GSDC, we build a testbed environment and discussed the factors to be considered later. Representative factors include the routing method inside the server, the number of containers in a server, service arrangement by container, and response to equipment replacement.

Docker based system can support to solve the various operational problems. In the Docker based network, however, load balancing is remarkably important for smooth communication between containers. In addition, various studies such as simulation are required to deploy the Docker system to the entire GSDC. Therefore, the next step on this study will cover this part.

5. Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) through contract N-20-NM-CR02-S01 and the Program of Construction and Operation for Large-scale Science Data Center (K-20-L02-C04-S01).

6. References

1. Park SO, Ahn SU, Yeo I. Survey of an Online-Offline Project for KISTI-GSDC. Platform Technology Letters. 2017 Sep;4(3):17-9.
2. Anderson C. Docker [software engineering]. IEEE Software. 2015;32(3):102-c3.
3. Ismail BI, Goortani EM, Ab Karim MB, et al. Evaluation of docker as edge computing platform. In: 2015 IEEE Conference on Open Systems (ICOS). 2015 Aug. 24-26. IEEE;2015. p.130-5.
4. Varma PC, Kumari VV, Raju SV. Analysis of a network io bottleneck in big data environments based on docker containers. Big Data Research. 2016 Apr;3:24-8.
5. Marmol V, Jnagal R, Hockin T. Networking in containers and container clusters. In: Proceedings of netdev 0.1. 2015 Feb 14-17. Ottawa;2015.
6. Dusia A, Yang Y, Taufer M. Network quality of service in docker containers. In: 2015 IEEE International Conference on Cluster Computing, 2015 Sep 8-11. IEEE,2015. p.527-8.
7. Xu Y, Mahendran V, Radhakrishnan S. SDN docker: Enabling application auto-docking/undocking in edge switch. In: 2016 IEEE conference on computer communications workshops (INFOCOM WKSHPS). 2016 Apr 10-14. IEEE;2016. p.864-9.
8. Xingtao L, Yantao G, Wei W, et al. Network virtualization by using software-defined networking controller based Docker. In: 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. 2016 May 20-22. IEEE;2016. p.1112-5.
9. Zeng H, Wang B, Deng W, Zhang W. Measurement and evaluation for docker container networking. In: 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). 2017 Oct 12-14. IEEE;2017. p.105-8.
10. Kubernetes Network Plugins. Kubedex [Image on Internet]. 2018 [updated 2018 Dec 10; cited 2020 Oct 20]. Available from: https://kubedex.com/kubernetes-network-plugins/ (website).
11. Understanding the Enterprise Data Center Solution. Juniper Networks. 2018 [updated 2018 Jan 29; cited 2020 Oct 20]. Available from: https://www.juniper.net/documentation/en_US/release-independent/solutions/topics/concept/solution-enterprise-data-center-understanding.html (website)

12. Yong C, Lee GW, Huh EN. Proposal of container-based HPC structures and performance analysis. Journal of Information Processing Systems. 2018 Dec;14(6):1398-404.
13. Determine best networking option. Project Calico [Internet]. 2020 [updated 2020 Jun 19; cited 2020 Oct 20]. Available from: https://docs.projectcalico.org/networking/determine-best-networking/ (website)