2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery

Measurement and Evaluation for Docker Container Networking

Hao Zeng, Baosheng Wang, Wenping Deng, Weiqi Zhang
College of Computer
National University of Defense Technology
Changsha, China
zenghao@nudt.edu.cn

***Abstract*—Container technology is a virtualization technology at the operating system level. It is more in line with future data center needs for its lightweight resource management, second-level start-up features. As the bridge between containers, container network plays a very important role in container life cycle system. This paper investigates and analyzes the current status of the development of container network, mainly container network model and network solutions. Besides we design experiments to measure and evaluate the performance of three mainstream network solutions (Flannel, Swarm Overlay, Calico). Our experiments show that Calico is of the highest performance and its TCP throughput is almost the same to host, but the configuration is the most complicated;Flannel configuration is simple, but it is designed for Kubernetes and the performance is not high; Swarm overlay configuration is simple with low performance. For it is originally native to Docker, its potential applications are very considerable.**

***Keywords*-Cloud Computing, Data Center, Container, Container Network, Model, Performance.**

I. Introduction

Today, cloud computing is rapidly developing and large-scale application. As a kind of network-based computing, cloud computing can provide shared software and hardware resource to users by using network, which realizes the reasonable distribution of information and resource. Meanwhile, cloud computing brings along the changes of the traditional data center, which produces a new generation of data center: cloud data center.

By building computing resources, storage resources and network resources into dynamic virtual resource pool using virtualization technology, cloud data center can realize automatic deployment, dynamic extensions and allocation on demand of resources. So virtual resource management becomes hotspot and difficulty of the construction of the cloud data center. But now cloud data center faces with many problems such as low resource utilization, application and platform can not be decoupled, application runtime environment limitations are strong, and operational staff control decrease, which limit the development of it. Container[1][2] technology with lightweight resource management and second-level start-up features can solve the problems mentioned above, which provide a new direction of development for cloud data center.

Container technology is a virtualization technology at the operating system level, it can provide a separate file system, network and process context space for each running server without modifying the host operating environment. So that, each service can arbitrarily change the contents of the operating system files, configuration of network and user in their own virtual running space, without damage to real host system files. Although an instance of the operating system is shared between containers, they run independently and without interference. Besides, do not relaying on the integrity of the operating system makes container more lightweight and fast start than virtual machine, which meets the needs of cloud data center.

As an important part of container technology, container network mainly solves the problem of communication between containers in the case of ensure container isolation. Since container is a new technology, container network is not yet mature, and many container network solutions were proposed to solve the problems of container network. Each solution has its own characteristics, and how to choose appropriate solution according to different scenarios is not only very important for the efficiency of communication between containers but also of great significance in improving cloud data center performance. In order to provide experiences and references for choosing and deploying appropriate container network, we introduce three kinds of mainstream container network solutions (Flannel[3], Docker Swarm Overlay[4][5], Calico[6]), and design experiments to measure and evaluate the performance of them.

The remainder of the paper is organized as follows. We introduce container network model in Section II. In Section III, we analyse three kinds of network solutions from different angles. In Section IV, aiming at the network solutions mentioned above, we design experiments to measure and evaluate the performance of them. Finally we make the conclusion in Section V.

II. Container Network Models

For container network solutions, we should focus on container network models. There are two mainstream models called Container Network Model (CNM) and Container Network Interface (CNI).

978-1-5386-2209-4/17 $31.00 © 2017 IEEE
DOI 10.1109/CyberC.2017.78

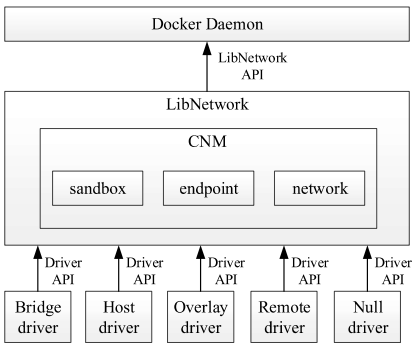

Figure 1. CNM Architecture.

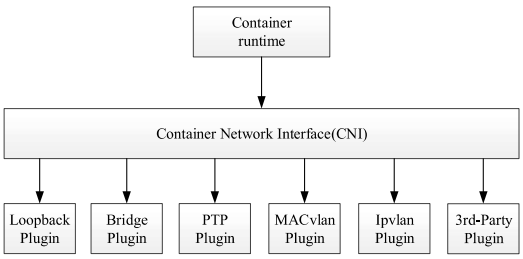

Figure 2. CNI Architecture.

A. Container Network Model

CNM[7] is proposed by Docker's Libnetwork project, which focuses on container networks research, providing standardized interfaces and components between the Docker daemons and network drivers. CNM architecture is depicted in Figure 1.

AS network core to provide services, LibNetwork can meet the needs of the upper and lower levels. For the upper application, Docker daemon can perform the creation and management of network by calling the APIs providing by LibNetwork; for the lower application, the five drivers built into LibNetwork can provide different types of network services. CNM occupies the core of LibNetwork, which is the key to provide network services. CNM mainly includes three components named sandbox, endpoint and network.

Sandbox is an isolated network operating environments that preserves the configuration of container network stack, so that different containers can be completely isolated, including network interface, routing and DNS configuration management. A sandbox can include multiple endpoints from multiple networks.

Endpoint represents the point which container access to network, we can think of an endpoint as a physical network card. An endpoint can only belong to a sandbox and a network, by adding a number of endpoints to the sandbox, we can join a sandbox to multiple networks.

Network represents a set of endpoints that can communicate directly to each other, which based on Linux bridge or vlan. On the host, each network is a separate network namespace, containers on the same network can be connected to the network namespace.

The emergence of CNM makes it possible that different containers within the same subnet can run on different hosts. The structure of CNM is clear, so is the upper and lower service management. CNM is closely integrated with Docker container life cycle, which leads to its lack of compatibility and diversity.

B. Container Network Interface

CNI[7][8] is a container network specification with compact structure proposed by CoreOS. The key to CNI lies in the simple contract between the container Runtime and the network plugin. Besides, CNI defines the input and output needed by the CNI plugin through the JSON syntax which construct the entire CNI architecture. The CNI plugin only needs to provide two commands throughout the model: one for adding network interfaces to the specified network and the other to remove it. These two commands are called when container is created and destroyed. CNI architecture is depicted in Figure 2.

CNI provides a universal container network model, container can be added to multiple networks driven by different plugin, and each network has its own corresponding plugin and unique name. Therefore, CNI can be compatible with other container technology and the upper system.

CNM and CNI are not two completely contradictory models, they can be transformed into each other. Comparing Figure 1 to Figure 2, Sandbox in CNM is consistent with Container Runtime in CNI. The Endpoint in CNM is implied in the ADD/DELETE operation in CNI.

III. Container Network Solutions

Container network mainly solves the problem of communication between containers in the case of ensure container isolation. Container network is not yet mature, and many solutions have been proposed for it. Here we introduce three mainstream container network solutions called Flannel, Docker Swarm Overlay and Calico.

A. Flannel

Flannel[9] is a network planning service designed for Kubernetes[10][11]. The deployment and configuration for Flannel is relatively simple, through the Etcd assigned IP address, it can manage multiple containers cross host. Flannel is designed to be lightweight and superior in performance. It can solve the IP conflict in Docker default configuration by replanning the allocation of IP address. When communication, Flannel will assign each Docker daemon an ip segment.

Through Etcd to maintain a cross host routing table, the IP between containers can be connected to each other. When two containers across the host communicate with each other, they will modify the destination address and source address of the packet, and unpack it after sending to the target host through routing table.

Flannel is a Overlay Network mode that allows TCP data to be encapsulated in another network packet for routing forwarding and communication. Currently, it supports UDP, Vxlan, AWS VPC and GCE. Kubernetes comes with Flannel for network communication, so Flannel is compatible with Kubernetes, and Flannel can be used to enhance Kubernetes' orchestrating and scheduling efficiency under the Kubernetes cluster network.

Flannel is the most mature in container network solutions, but since each host is a separate subnet network architecture, it is not possible to achieve fixed IP container drift and lack of flexibility. Beside, because of the lack of multi-subnet isolation mechanism, high dependency on the upper design and other defects, Flannel can only be used for the container cluster which have low requirement for flexibility.

B. Docker Swarm Overlay

Docker Swarm Overlay[4][12] is a network solution that Docker uses the Swarm framework to solve network communication problems between containers. Swarm Overlay was originally native to Docker, the deployment and configuration is the simplest. Docker 1.12 previously, Overlay needed to add additional key-value storage (Consul, Etcd etc) in the Swarm cluster to synchronize the network configuration to ensure that all containers were in the same network segment. This storage was built in Docker 1.12 and integrated the support of Overlay Networks, which simplifies the complexity of network configuration.

Overlay integrates load balancing and service address function through the Swarm, it provides a DNS address for each service and maintains a common port. Besides, it implements the monitoring and management mechanism to maintain the operation of service state. However, Overlay solution uses Vxlan for cross host communication and requires encapsulation and decapsulation, which lose the performance of network, so it is just suitable for the construction of a simple network with low performance.

C. Calico

Calico[6] is a high performance data center solution base on BGP protocol, implemented through three layer routing. By compressing the entire Internet's extensible IP network principles to the data center level, Calico uses vRouter for data forwarding at each compute node and spreads the workload routing information to the entire Calico network through the BGP protocol to ensure that all data traffic are completed through IP packet. Similar to the general routing scheme, Calico runs a large number of routing tables on the host, these routing tables are dynamically generated and managed by Calico through its own components, with no involvement of tunnels or NATs and less performance loss. These make sure that Calico is of high performance.

The Calico solution is suitable for networks that require high performance and high isolation. Each container in Calico has its own network protocol stack, which facilitates node interconnection. Because Calico directly uses the data center network structure, when deploying network, there is no need to rely on independent network equipment, so the transfer efficiency is higher. Small-scale deployment can be directly interconnected, when deploying a large-scale network , you need to specify the BGP route reflector to complete. Since Calico has a separate IP address for each container, the routing table is relatively large, and Calico is inadequate in terms of security and mobility support.

IV. Measurement and Evaluation

As described in Section III, each solution has its own characteristics. In this section, we design experiments to measure and evaluate the performance of them. The experiments contains three parts: delay of ping, tcp throughput and udp throughput. The experimental results is below.

A. Delay of Ping

The result is depicted in Figure 3, from which we can know that: Calico, Calico ipip and Flannel VxLAN are better than Swarm overlay; Flannel udp is the worst.

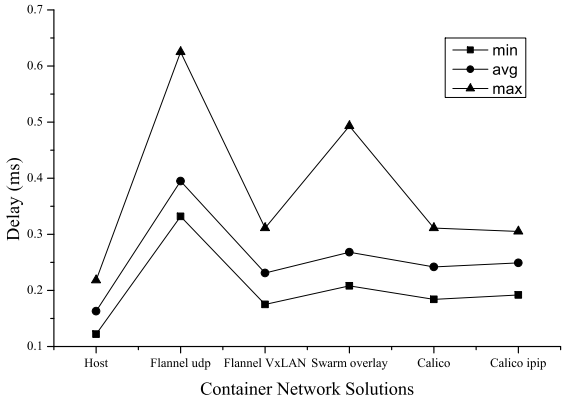

Figure 3. Delay of Ping.

B. TCP Throughput

Figure 4 shows the result of TCP Throughput. We can see that: Calico is the best, which is close to Host; Calico ipip, Flannel VxLAN and Swarm overlay are almost the same, which are slightly poor than Cilico; Flannel udp is the worst.

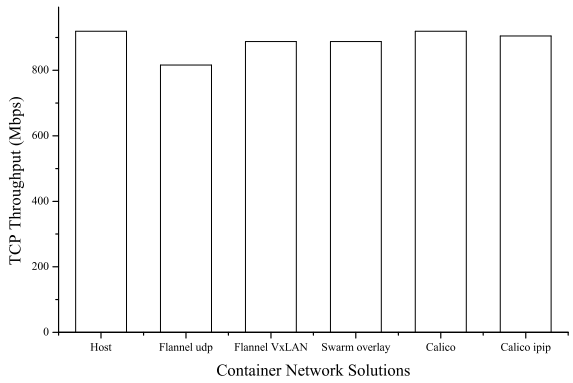

Figure 4. TCP Throughput.

C. UDP Throughput

Figure 5 shows the result of UDP Throughput. We can see that: all the solutions are not effective. But Calico is still the best; Calico ipip, Flannel VxLAN and Swarm overlay are far worse than Calico; Flannel udp is still the worst.

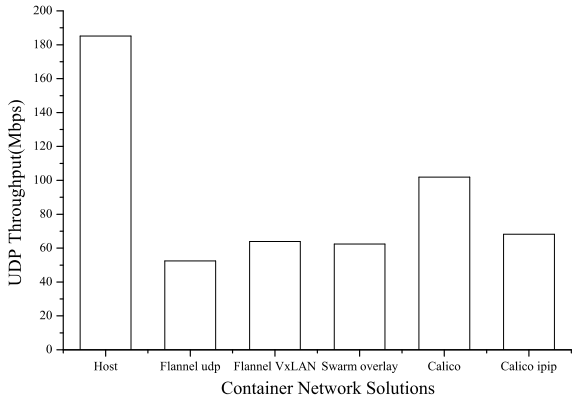

Figure 5. UDP Throughput.

D. Comparison and Analysis

As shown in Section III, each of these solution has its own characteristics. Combining with our experimental results, we carry out qualitative comparison and analysis on these three solutions. The comparison results are depicted in Table I.

V. Conclusion

In this paper, we present a comprehensive overview of two mainstream container network models (CNM, CNI) and three mainstream container network solutions (Flannel, Docker Swarm Overlay, Calico). Beside, we design experiments to measure and evaluate these three container network solutions. The TCP throughput of host is 919.4Mbps, Calico is 919Mbps, Flannel is 815.9Mbps, Swarm overlay is 887.6Mbps. Our experimental results reveal that Calico is of the highest performance and its TCP throughput is almost the same to host, but the configuration is the most complicated; Flannel configuration is simple, but it is designed for Kubernetes and the performance is not high; Swarm overlay configuration is simple and its performance is close to host. For it is originally native to Docker, its potential applications are very considerable.

Table I
Comparison to Different Container Network Solutions

Network Solutions	Applicable Scene	Performance	Deployment	Configuration
Flannel	CoreOS	Middle	Easy	Easy
Swarm Overlay	Docker	Middle	Easy	Easy
Calico	BGP	High	Hard	Hard

References

[1] R. Haines, "Interaction object tracking analysis docker image 1.0," *Journal of Pharmaceutical Sciences*, 2016.
[2] T. Bui, "Analysis of docker security," *Computer Science*, 2015.
[3] "http://dockone.io/article/618."
[4] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," 2014.
[5] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, 2014.
[6] N. Mangano, A. Baker, and M. Dempsey, "Software design sketching with calico," *IEEE/ACM International Conference on Automated Software Engineering*, 2010.
[7] "http://www.dockerinfo.net/3772.html."
[8] S. Nesmachnow and G. Tancredi, "Scheduling algorithms for distributed cosmic ray detection using apache mesos," 2016.
[9] C. Boettiger, "An introduction to docker for reproducible research," *Acm Sigops Operating Systems Review*, 2015.
[10] D. Vohra, "Kubernetes microservices with docker," *Apress*, 2016.
[11] E. Brewer, "Kubernetes and the path to cloud native," *ACM Symposium*, 2015.
[12] J. Baier, "Getting started with kubernetes," *Packt Publishing*, 2015.