Container4NFV: How refactoring VM-based VNF to Container-based microservice VNF

Fu Qiao (fugiao@chinamobile.com)
Jia Xuan (jiaxuan@chinamobile.com)
Agenda

• Background

• Design container-based microservice VNF

• Q/A
Cloud Native is on the way!

Container brings new thought and new method to build up application. From OpenStack to Kubernetes, we find the focus is moving from resource-oriented thought to service-oriented thought.
From monolithic to virtualization

• The architecture of application is same
• Add a new driver
• Still using the old name like “interface board” or “master board”
• P2V tools can reduce work effort!! Very powerful tools.
From virtualization to container

- The application architecture is totally different than before
- Each application is made up of several functions.
- Each function can run well, be self-healing.
Agenda

• Background

• Design container-based microservice VNF

• Cloud Native Technology for VNF.

• Q/A
VNF Design: Stateless vs. Stateful

Stateful Applications:
States and Processing Intertwined

Stateless Applications:
States and Processing Separated and Interconnected over High Performance Network

Processing
States
States
Processing

High Performance Interconnect
States
States
Stateful application

• **Persistent state**
  the states that will survive app restarts and outages;

• **Configuration state**
  non-domain data, like IP addresses or credentials;

• **Session state**
  like temporary state for a session;

• **Connection state**
  states associated with some connection-oriented protocols for communication

• **Cluster state**
  Some applications run as multiple instances in a cluster, for availability and scale, and require shared knowledge of cluster membership and state.
Stateless VNF Design Strategies

1. **State Decoupling**
   
   A stateful VNF has been designed with runtime data (i.e. states) directly embedded into their codes. Stateless VNF requires redesigning the network functions to decouple the state.

2. **Distributed Architecture**
   
   A distributed architecture can improve application resiliency. The single point of failure will be eliminated if an application functionality is spread across containers residing on multiple servers. Having distribution is one of the important factors for better horizontal as well as vertical application scaling.

3. **Microservices Architecture**
   
   Most network functions are deployed in single monolithic space. This can lead to maintenance challenges, as well as slow down the trialing of new technologies. Microservices-based decomposition is a significant step in evolving VNFs to be cloud-native and much more agile and scalable.

4. **Performance Achievement**
   
   Many network functions will sit in the data path, and be required to process a large number of packets per second; we must ensure that efforts to improve performance do not compromise the consistency of the states.
State Decoupling

• Cache State
  • The better design choice is still to keep them internal to the instance of a network function. This can optimize the processing performance
  • The “cached states” are usually short-lived, being ephemeral. The ‘cached states’ are usually short-lived, being ephemeral.

• Shared State
  • It shall be decoupled (extracted) from the instance (of a network function)
  • Stored/migrated to an external replica or persisted to a backend store (to implement failure recovery)
Distributed Architecture for State Storage

- The states are also replicated to multiple storages, then the data-access performance and scalability would also be at advantageous.
- The distributed fault-tolerant state store can be implemented as in-memory data grid, cache and persistent object stores, such as flash/SSD or even hard disks.

Challenge

- Information in DB associated with the corresponding micro-services needs to be synchronized; this could present lots of overhead and even introduce the element of risk (like out-of-sync caused system failure, etc.)
- Another concern is the potential performance hit if a complete workflow has to access info from different DB locations.
Microservices Architecture

VNF decomposition generally indicates the breaking of a monolithic VNF into a set of collaborating services, with each service implementing a single function, or a set of related functionalities.

**Benefit**

- Microservices architecture offers the ability to scale independently of another; services can be deployed on hardware that is best suited to its resource requirements; monolithic has limitation “all-in-one” HW.
- Microservices architecture shrinks the failure domain and improves fault isolation, facilitating easy troubleshooting and high availability; monolithic could bring down “whole”.
- Microservices allows for common functions to be stripped away from the core logic of applications, so called “lightweight” – this makes functions easier and quicker to develop, manage and deploy.
- Decomposition (into microservices) allows VNFs to remove unneeded functionalities – lightweight, smaller footprint, less Capex, etc.
Performance Consideration

- State-decoupling with distributed architecture and storage will inevitably introduce communication latency, which will definitely post a challenge for achieving the similar or even better runtime performance.

- Many network functions are sitting in the data path, and required to process a large number of packets per second.

---

- Add a caching layer:
  - An element can look in the cache for a matching data, and if found, returns the value immediately. Then for writes, it will update the cache and perform a cache coherency algorithm to the target.

- Specific acceleration technologies
  - DPDK/VPP: Network acceleration
  - FaRM: Fast Remote Memory from USENIX NSDI'2014 is a new main memory distributed computing platform that exploits RDMA to improve both latency and throughput by an order of magnitude relative to state-of-the-art main memory systems.
Clover: building VNFs with Service Mesh backed micro-services

https://wiki.opnfv.org/display/CLOV

Control flow during request processing

Discovery & Config data to Proxy

Policy checks, telemetry

TLS certs to Proxy

Example VNFs

Common cloud native toolset for tracing, logging, monitoring ...

DNS  DHCP  CA Broker

Test  Debug

Common Networking micro-services
Container4NFV: The extensions to Kubernetes and container runtime

Container4NFV is the first container project since 2016. It provides an environment where containerized VNF can run.

https://github.com/opnfv/container4nfv

- Multus CNI Plugin
- OVS-DPDK
- Istio
- Stor4NFV (doing)

- Kata Container
- Virtlet
- X86 Platform
- ARM Platform
- SGX (Security)
Thanks