Baker: Scaling OVN with Kubernetes API Server

Han Zhou
OpenStack Summit Boston 2017
Why OVN?

OVS is GREAT.

OVN makes it GREATER!
OVN Challenges

- OVN is distributed, but not fully ...
  - Can we distributed Northd?
OVN Challenges

- OVSDB SB
  - No clustering (yet)

It is nothing but **distributed state management** ...
Scale-out with Baker

- Distributed northd
  - Computes lflows for local only
- Scale-out central cluster
  - K8S API server framework
  - Backed by ETCD
  - Clustering
- Distributed agents
  - Watch for local objects only
  - Translate objects to NB DB
One more thing ...

- Northd and ovn-controller are all distributed
- They process data related to local HV only

But what does this mean?
In terms of overlay ...

- Logical-to-physical mapping states (port-binding) for connectivity
- Doesn’t scale when everyone talks to everyone else in a *large* zone
  - Maybe not the case for public cloud, or small-to-medium enterprise cloud.
  - But it is typical use case for eBay’s private cloud.
Are we solving the right problem?

- Connectivity v.s. Segmentation
- L2 Segmentation v.s. L3 segmentation
- Address sets (L3) based segmentation
  - ACL: default deny, whitelist access
  - IPAM:
    - Use ip efficiently
    - Summarized CIDRs to reduce address set size
Flat network

- Reuse OVN abstraction and pipeline
  - Port security
  - ARP proxy
  - ACL
  - LB
  - ...
  - But NOT overlay
- Use localnet port to connect to physical network directly

- Data to be processed by each HV depends on size of AddressSet used by ACLs that apply to ports on the HV
Baker Object Model

- Similar as OVN NB Schema
  - Logical Port
    - Addresses
    - Port security
  - ACL
  - Address Set
  - Load balancer (TBD)
  - ...

- Differences
  - No Logical Switch (local)
  - Port-SecGroup binding
  - ACL: SecGroup instead of individual ports in inport/outport
Neutron Plugin

- Support security group, with API extensions
  - Address set - support external IPs from legacy systems
  - Security group rule packet logging
Scalability - Control plane throughput

- **Test**
  - E2E: Neutron - Baker - OVS
  - Simulated 1k HVs on 10 BMs
    - OVS/OVN 2.7
  - 1 node Neutron + mysql
  - 1 node Baker API server + ETCD
    - K8s 1.6 pre-release, etcd 3.0
- **Result for single client (parallel test TBD)**
  - Result impacted by SG (address set) size
  - ~3 ports/sec for SG size 1K
Scalability - Latency

● Test
  ○ E2E from Neutron to OVS flow installation for the port created
    ■ Create port from neutron, bind port in ovs on HV
    ■ Wait:
      ● `ovn-nbctl wait-until Logical_Switch_Port <port> up=true`
      ● `ovn-nbctl --wait=hv sync`
  ○ Create ports on top of existing 10K ports, 1K HVs, SG size 1K
  ○ 10K * 3 (flows/ACL) = 30K flows / ovs port

● Result
  ○ Avg 2 sec
Improvement - ovn-controller

- Flow computation blocks flow installation
- Improvement: avoid repeated computation when in-flight messages to OVS pending
- Test result (SG size 10k, flow installation for 10 ports on HV):
  - $10k \times 3 \times 10 = 300k$ OVS flows
  - Before: 50 min
  - After: 16 sec
Other Lessons learned

● Postpone ACL expanding from Neutron to HV
  ○ Introduce port-group binding object in Baker
  ○ Use port-group instead of lport in “inport/outport” in ACL
  ○ Baker agent expand ACL on HV for local lports only
  ○ Benefit:
    ■ Reduced Neutron overhead
    ■ Reduced API calls from Neutron to Baker
    ■ Reduced data size in Baker
Other Lessons learned

- Baker RESTful API: use Protobuf instead of JSON-RPC
  - 10 - 20% throughput increase for SG size 1k - 10k
  - Lower CPU cost on API-server
Thanks!