Scaling the OVN Control Plane in OVS 2.6.0

Liran Schour, Ryan Moats
Topics

- Conditional Monitoring
- Wire Protocol Optimization
- Incremental Processing
- Open Need
Define logical network topology

Update SB tables

Update each HV on any change (ovn-controller)

HIGH PROCESSING OVERHEAD

REDUNDANT FLOWS
Conditional monitoring

Define logical network topology

Update SB tables

Update only HVs that match conditions (ovn-controller)

Add VM2’s network to condition

Add VM1’s network to condition
Add to the OVSDB protocol the following requests:

- **monitor_cond:**
  Allows clients to start a conditional monitor session

- **monitor_cond_change:**
  Allows clients to iteratively change the conditions of the monitor session
ovsdb_idl_add_clause_false(idl, tableA); // Start with empty table

while (1) {
    ovsdb_idl_loop_run(idl);
    ...
    ovsdb_idl_add_clause(idl, tableA, clause1);
    ...
    ovsdb_idl_loop_commit_and_wait(idl);
}
# of Flows:
- Patch
  - Logical flows = 5010
  - Host 1 # flows 835
  - Host 2 # flows 927
  - Host 50 # flows 1111
- Master
  - Logical flows = 5010
  - Host 1 # flows 5793
  - Host 2 # flows 5819
  - Host 50 # flows 5871
Influence of network spread over DC on SB
Wire protocol optimization

• OVSDB protocol options for changing data
  – Read-modify-write
    • Transmits entire row state from client to server for verification to avoid dirty reads
  – Mutate
    • Only transmits row deltas
How to get there

**OVS**
- Extend HPE’s partial map update contribution to cover partial sets
- Expose partial set update capability in Python IDL

**CMS**
- Call new partial set update capability
What does it buy us?

- Rally test adding ports to a local switch and ACL entries
- Sniff protocol stream from CMS to OVN NB DB
Another data point

- CMS: OpenStack Neutron+networking-ovn (Newton)
- Test: Time taken to launch 10 instances from Horizon
  - Using read-modify-write: 60 seconds
  - Using partial set updates: 37 seconds
  - ~40% improvement
• OVN controller process performs a full recalculation of all OVS flows each pass.

• At scale:
  – Pegs a CPU
  – Controller loop time exceeds 1 second, leading to lag in picking up new changes from Southbound Database

• Goal: only recalculate changes
• Attempt didn’t quite work
  – Persisting state is hard
  – Too many “back doors” to full recalculation

• Result:
  – Didn’t provide a gain during scale up/scale down
  – Quiesces OVN controller doing idle time (but there are simpler ways to get there)
Open Need

- **NB and SB DBs**
  - Today, one ovsdb-server process for each
    - Defeats increasing concurrency via horizontal scaling
  
- Clustering for both NB and SB
  - Avoid SPOFs
  - Horizontally scale NB
  - “Shard” chassis among SB
Questions?

- Thanks for listening!