Tuning & Hardening Trellis for Large Scale Deployment

Kalicharan Vuppala
Hariprasad Rajendran
INFOSYS
Tuning & Hardening

What do you do to your stock tires if you want to drive to Lake Tahoe in Winter??...

* Defaults are like stock tires...

* Understanding the environment and product capability is extremely important to arrive at right tuning of parameters and hardening features
Data Plane

- Leaf-Spine-Leaf network Topology
- One leaf layer connected to Servers running CMTS VNFs and other leaf layer connected to the R-Phy devices, with the fabric interconnecting them.
- End services (CM & CPEs) connect through the R-Phy device and get tunneled to the CMTS VNFs and reach the Internet through the upstream routers.
- R-Phy, CM, CPEs use DHCP for IP assignment

Control Plane

- 3 Node ONOS cluster.
- The ONOS Cluster uses Distributed stores to reflect the network state in terms of Link store, Host store, Route store, Flow & Group stores etc.,
- Each instance may assume a role of device master or distributed store partition leader and have corresponding back-up/Follower instances
Tuning Parameters
GC & Distributed Cluster Timers Tuning

- ONOS Distributed Cluster Keep-alive, Election timers play a crucial role in keeping the cluster up and preventing network meltdown.
- The GC stop the world pause whenever it goes beyond cluster timeout, causes network partition that can result in network meltdown depending on the volume of data to be synced.

* GC pause induced partition happens due to all Processing resources allocated for GC thereby missing cluster keep-alive processing and resulting in network meltdown
GC & Distributed Cluster Timers Tuning

- Tune the cluster timers and the GC parameters to avoid the longer Pause!
  - GC Pause < Heart Beat Timer < Election Timer

- CMS (Concurrent Mark Sweep) doesn’t provide much flexibility with controlling pause timers so changed GC mechanism to G1GC, that has control up to 200ms of pause.
  - G1GC almost reduced all the network partitions with a 200ms pause timer which is lesser than the cluster communication timeout.

- However with increased scale more tuning of G1GC parameters and Java JVM heap size was required to achieve route scale of 150k. Tuning Explained comprehensively here:
  - GC Pause < Heart Beat Timer < Election Timer
  - G1GC parameter Tuning(G1HeapRegionSize, ParallelGCThreads, ConcGCThreads etc.)
  - JVM heap size changes made it possible to achieve 120 to 150K Route scale with stable soak testing

https://docs.google.com/document/d/1bY6dyl57GqqXFVYPaLcQPyu74eeU9B1nofcnbJ1EdLk/edit?usp=sharing
LLDP StaleLinkAge Tuning

- When the controller doesn’t see LLDP messages on a link within the StaleLinkAge time, the link is marked as broken. When controller is busy with other processing including GC, it can cause wrong StaleLinkAge timeout and link removal.

- Multiple false link removals at scale can result in total topology re-convergence computation which can cause Network meltdown.

* Increase LLDP StaleLinkAge more than known controller processing tolerance at scale
Hardening Features
Route/Flow Store Explosion

When the number of indirect hosts keep increasing with scale, it causes route/flow store explosion and also increases switch processor utilization, costly service bring-up and topology convergence.

Leaf-Spine-Leaf Data Plane | 3 Node ONOS Cluster Control Plane

* 120k Indirect Host Service Bring-up with Route/Flow programming time in 8 Leaf switches = 45mins
* System not in stable condition with 120k routes with Soak Testing
Route/Flow Store Explosion

Route Simplification – Program Indirect hosts only where the next hop resides!

Leaf-Spine-Leaf Data Plane | 3 Node ONOS Cluster Control Plane

- Leaf1 Flow Store: {DH1Flow}
- Leaf2 Flow Store: {DH1Flow, IDH1Flow, ... IDHnFlow}
- Leaf3 Flow Store: {DH2Flow, IDH1Flow, ... IDHnFlow}
- Leaf4 Flow Store: {DH3Flow}

ONOS

- ONOS 1
- ONOS 2
- ONOS 3

Host Store

{DH1, DH2, DH3}

Route Store

{<IDH1 NH>, <IDH2 NH>, ... <IDHn NH>}

- * 120k Indirect Host Service Bring-up with Route/Flow programming time in 8 Leaf switches = 25mins
- * System in stable condition with 120k routes Soak Testing for 2 weeks+
Critical Event Prioritization

- As scale increases critical Service bring-up events which depends on Packet-in Processing requires prioritization.

- Without prioritization time taken for bringing up all services becomes underwhelming and can cause the overlay applications to trigger retries resulting in a choked bottle-neck situation.

Timeout due to slower processing results in re-tries that pumps even more packets that further slows

* DHCP based customer Service bring-up delayed up to 2 hours + and further worsens due to re-tries with 20K customers coming online at the same time.
Critical Event Prioritization

- Multi-Threaded! Helps fasten packet-in processing thereby indirectly achieving prioritization for service bring-up event.
- The solution avoids time-outs and resulting retries on the application side!

* DHCP based customer Service bring-up taken care smoothly up to 120k customers coming only at the same time due to multi-threaded processing.
Distributed Store Operation for Non-Critical Data

- Storing non-critical data using distributed store implementing strong/eventual consistency is costly at large scale and causes performance degradation.
- Especially in a critical event processing cycle, store operations and locks will cause noticeable performance throughput variations.

* DHCP based customer Service bring-up delayed up to 2 hours + even after multi-threading due to consistent store based dhcp counter processing
Distributed Store Operation for Non-Critical Data

- Local Stores! Store non-critical data using local stores. This will greatly avoid the number of store operations across instances and deliver better throughput.

* DHCP based customer Service bring-up taken care smoothly up to 120k customers coming only at the same time due to multi-threaded processing and converting dhcp counters to local stores achieving a turn around time within 25mins

* Distributed store operation for DHCP counters for every DHCP transaction slows down the processing speed of each service bring-up
Resource Deprivation Due to Infinite Event Queue

The infinite event queue length for event processing, results in processor and memory resources wasted on faulty event processing like:

1) Wrong configuration – same IP/MAC for two hosts results in continuous host movement, host probing for discovery and related Route/Flow processing
2) Flapping links – continuous link entry addition/removal and corresponding topology convergence processing each time
3) Packet-in Storm – Rogue hosts sending NDP/ARP/DHCP can hog the controller resources due to continuous packet-in processing

Leaf-Spine-Leaf Data Plane

3 Node ONOS Cluster Control Plane

- Continuous host moves and related host probing with duplicate IP/Mac have filled the queue so deep that even host entry is removed processing continued for hours.
1) Packet Throttling – To avoid continuous packet-in processing during a storm
2) Symmetric Probing – To avoid too many host probe discovers by assuming symmetric host connectivity
3) Event Rate limiting – To avoid continuous event processing by monitoring for same event type occurrence
4) Duplicate Host Detection – To avoid processing duplicate hosts due to configuration mistakes.

Leaf-Spine-Leaf Data Plane  3 Node ONOS Cluster Control Plane

* Symmetric Host probing and Duplicate host detection stops hours of unnecessary processing.
Packet throttling to avoid continuous Packet-in Processing
Route/Flow Calculation Suppression

- Current Route Simplification suppresses pushing flows, but at higher scale even calculation is costly.
- Suppress calculation for devices which don’t need the flows!

Leaf-Spine-Leaf Data Plane  3 Node ONOS Cluster Control Plane

- Calculate and push only for required device
Optimize Flow Stat Collection

- Periodic Flow stat collection from devices and syncing across distributed cluster is a costly process at very high scale.
- Only collect when there is a flow change or device mastership change!

* Collect Device flow only when there is flow change or device mastership change
Avoid Network Meltdown

• Humungous flow object synchronization failures during device mastership change handling causing network meltdown.

  * Flow object size optimization to avoid Humungous object creation for cluster sync-up Netty messaging

• When all instances in the distributed cluster has a partition, it results in link store clean-up forcing a complete topology convergence. At very high scale this topology convergence can cause network meltdown.

  * Local Link store of one of the instance can be taken as source of truth and avoid the clean-up
Thank You