Horizontal Scaling With OVN Component Templates

Dumitru Ceara
Why “horizontal” scaling?

Node 1
- ext-worker-1
- GR-worker-1
- ovn-worker-1

Node 2
- ext-worker-2
- GR-worker-2
- ovn-worker-2

Node 3
- ext-worker-3
- GR-worker-3
- ovn-worker-3

Node n
- ext-worker-n
- GR-worker-n
- ovn-worker-n
It could have been "vertical" too…

Node 1
ovn-worker-1
GR-worker-1
ext-worker-1

Node 2
ovn-worker-2
GR-worker-2
ext-worker-2

Node n
ovn-worker-n
GR-worker-n
ext-worker-n

… but it wouldn’t fit that well :-(
OVN-K8S network topology

- Distributed:
  - ovn_cluster_router
  - join_switch
- Per node:
  - ovn-worker switch
  - GR-worker router
  - ext-worker switch
"Load balance traffic destined to a cluster-internal IP (and port) to a set of backends (pods)."

- Single OVN Load Balancer applied to all ovn-worker switches
- 1:1 mapping between k8s service object and OVN load balancer object
Service1 =
(.vip=42.42.42.42:4242,
.backends=[pod2IP, pod3IP], tcp)

- pod1IP -> 42.42.42.42:4242
  - DNAT on ovn-worker-1 to either pod2IP or pod3IP
- pod2IP -> 42.42.42.42:4242
  - DNAT on ovn-worker-2 to either pod2IP or pod3IP
- pod3IP -> 42.42.42.42:4242
  - DNAT on ovn-worker-3 to either pod2IP or pod3IP

Scales linearly:
- S services -> S load balancers
- O(S) logical flows
"Exposes the Service on each Node's IP at a static port."

- Unique OVN Load Balancers applied to all N GR-worker routers
- 1:N mapping between k8s service object and OVN load balancer object
OVN-K8S services - NodePort example

Service1_node1 =
(vip=10.0.0.1:4242,
.backends=[pod2IP, pod3IP], tcp)

Service1_node2 =
(vip=10.0.0.2:4242,
.backends=[pod2IP, pod3IP], tcp)

Service1_node3 =
(vip=10.0.0.3:4242,
.backends=[pod2IP, pod3IP], tcp)

Does not scale nicely (S services, N nodes):
• S x N load balancers
• O(S x N) logical flows
Focus on the load balancers.

Almost identical load balancers...

If we highlight the different bits...

And then mask them out...

We get...
OVN Component Templates

Service1 = (vip=\^VIP_VAR:4242,.backends=[pod2IP, pod3IP], tcp)

- A component template has a name
  - template names have similar restrictions to port group and address set names
  - when referring to a template name use the ^ character as prefix
- A component template has (at most) one value on any given chassis in the cluster
  - defined through a new table in the OVN_Northbound database

```
"Chassis_Template_Var": {
  "columns": {
    "chassis": {"type": "string"},
    "variables": {
      "type": {"key": "string", "value": "string"},
      "min": 0, "max": "unlimited"}
  },
  "indexes": ["chassis"],
  "isRoot": true
}
```
OVN-K8S services - NodePort example with templates

Service1 =
(.template=true, .vip=^VIP_VAR:4242,
.backends=[pod2IP:4242, pod3IP:4242], tcp)

Service2 =
(.template=true, .vip=^VIP_VAR:8484,
.backends=[pod2IP:8484, pod3IP:8484], tcp)

Chassis_Template_Var
(.chassis=node1, .variables=[(VIP_VAR: 10.0.0.1)])
(.chassis=node2, .variables=[(VIP_VAR: 10.0.0.2)])
(.chassis=node1, .variables=[(VIP_VAR: 10.0.0.2)])

Scales linearly (S services, N nodes):
- S load balancers (templated)
- O(S) logical flows (templated)
- N Chassis_Template_Var mappings
OVN-K8S services - NodePort example with templates and unique backends

Service1 = (.template=true, .vip=^VIP_VAR:4242, .backends=^BACKEND1, tcp)

Service2 = (.template=true, .vip=^VIP_VAR:4242, .backends=^BACKEND2, tcp)

Chassis_Template_Var
(.chassis=node1, variables=[(VIP_VAR: 10.0.0.1), (BACKEND1:"POD1IP:4242"), (BACKEND2:...)])
(.chassis=node2, variables=[(VIP_VAR: 10.0.0.2), (BACKEND1:"POD2IP:4242"), (BACKEND2:...)])
(.chassis=node3, variables=[(VIP_VAR: 10.0.0.3), (BACKEND1:"POD3IP:4242"), (BACKEND2:...)])

Worst case (S services, N nodes, unique backends per node):

- S load balancers (templated)
- O(S) logical flows (templated)
- O(S x N) Chassis_Template_Var mappings
Simulate an OVN-K8S deployment with N nodes, S NodePort services, unique backend sets: 5 unique backends per service per node.

<table>
<thead>
<tr>
<th>Template</th>
<th>N</th>
<th>S</th>
<th>NB (size on-disk/RSS)</th>
<th>SB (size on-disk/RSS)</th>
<th>ovn-northd loop time</th>
<th>ovn-controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>60</td>
<td>1000</td>
<td>Size: 25MB RSS: 116MB</td>
<td>Size: 118MB RSS: 589MB</td>
<td>2.70s</td>
<td>RSS: 463MB Recompute: 0.52s</td>
</tr>
<tr>
<td>YES</td>
<td>60</td>
<td>1000</td>
<td>Size: 6MB RSS: 25MB</td>
<td>Size: 8MB RSS: 46MB</td>
<td>0.07s</td>
<td>RSS: 44MB Recompute: 0.20s</td>
</tr>
<tr>
<td>NO</td>
<td>120</td>
<td>2000</td>
<td>Size: 67MB RSS: 865MB</td>
<td>Size: 471MB RSS: 9000MB</td>
<td>15.60s</td>
<td>RSS: 1016MB Recompute: 0.40s</td>
</tr>
<tr>
<td>YES</td>
<td>120</td>
<td>2000</td>
<td>Size: 23MB RSS: 96MB</td>
<td>Size: 28MB RSS: 225MB</td>
<td>0.22s</td>
<td>RSS: 83MB Recompute: 0.40s</td>
</tr>
<tr>
<td>YES</td>
<td>120</td>
<td>10000</td>
<td>Size: 118MB RSS: 440MB</td>
<td>Size: 136MB RSS: 668MB</td>
<td>0.72s</td>
<td>RSS: 311MB Recompute: 1.77s</td>
</tr>
<tr>
<td>YES</td>
<td>250</td>
<td>10000</td>
<td>Size: 244MB RSS: 870MB</td>
<td>Size: 263MB RSS: 1502MB</td>
<td>1.26s</td>
<td>RSS: 318MB Recompute: 1.87s</td>
</tr>
</tbody>
</table>

- Templates allow scaling to **x2 nodes** and **x5 services** compared to the current (non-template) deployment while using less resources.
- For the **N=120 S=2000** case:
  - NB size reduced by ~65%, NB RSS reduced by ~90%, SB size reduced by ~95%, SB RSS reduced by ~98%
  - ovn-northd loop time reduced by ~98%, ovn-controller RSS reduced by ~92%
OVN Component Templates - Conclusions

- Significantly improve scalability when resources are distributed uniformly
- Supported for any type of OVN match/actions and Load Balancers
- Require work on the CMS side to define the templates in a way that translates optimally to virtual network resources
- Targeting acceptance in OVN v22.12.0

V1:
https://mail.openvswitch.org/pipermail/ovs-dev/2022-September/398110.html
https://patchwork.ozlabs.org/project/ovn/list/?series=320941&state=*
Thank you!