Elastic Load-Balancing
Using Octavia
deep dive

Dean H. Lorenz, IBM Research – Haifa
Allan Hu, Cloud Networking Services, IBM NSJ
Load Balancing 101

• Users access a service
  – Service hosted on cloud

• **Pool** of back-end servers (aka **members**)
  – High availability:
    • server failure ≠ service failure
  – Performance:
    • add/remove servers to match load

• One service IP (aka **VIP**)
  – Clients do not know which back-end serves them
  – Need to split incoming VIP traffic
Load Balancing 101 (2)

• Load balancer
  – Distribute new VIP connections to members
  – High availability: avoid failed servers
  – Performance: avoid overloaded servers
    • LB is not the pool manager: does not add/remove servers
    • But uses all available servers, reports broken ones
  – Health Monitor + Stats Collector

• LB Algorithm / Policy
  – Balance something
    • # connections, CPU load…
  – Affinity: similar packets go to same back-end
    • All packets from same flow (minimum affinity)
    • All packets from same source (quicker TLS handshakes)
    • All packets from same HTTP user
Load-Balancing as a Service (LBaaS)

- **Neutron LBaaSv2 API**
  - LB (VIP) → Listeners (protocol) → Pool → Members, Health monitor
    - neutron lbaas-{loadbalancer,listener,pool,member,healthmonitor}-CRUD,
      CRUD: {create,delete,list,show,update}

- **Octavia** (operator-grade LB)
  - VM per LB (aka **Amphora**) running HAProxy
    - 2 VMs for active-standby HA (Mitaka)
Load-Balancing as a Service (LBaaS)

- Neutron LBaaSv2 API
  - LB (VIP) → Listeners (protocol) → Pool → Members, Health monitor
    - neutron lbaas-{loadbalancer,listener,pool,member,healthmonitor}-CRUD,
      CRUD: {create,delete,list,show,update}

- Octavia (operator-grade LB)
  - VM per LB (aka Amphora) running HAProxy
    - 2 VMs for active-passive HA (Mitaka)
  - Many pieces under the hood…
    - Lot’s of pluggability
Amphora can do even more

• HAProxy is great
  – L7 Content Switching
  – Monitor back-end health
  – Cookie insertion (session stickiness)
  – SSL termination
  – Authentication
  – Compression
  – …

• Would be nice to include other functions
  – E.g., cache, FW, rewrite, …

⚠️ The more it does, the more resources it needs
Remind me again; why did I need a LB?

- High availability
  - Amphora is single point of failure
  - But active-standby just added in Mitaka

- Performance:
  - Huge, successful service…
  - Amphora might not be able to handle load
Elastic Load Balancing (ELB)

Remind me again; why did I need a LB?

- High availability
  - Amphora is single point of failure
  - *But active-standby just added in Mitaka*

- Performance:
  - Huge, successful service…
  - Amphora might not be able to handle load

Elastic Load-Balancing (ELB)

- Pool of Amphorae
- Need to split incoming VIP traffic over Amphorae pool
- Déjà vu…
**LBaaS Challenge:**

**Cost-effectively provide LBaaS for cloud workloads**

- Customers expect the cloud to support their *elastic* workloads
  - Cheap for small workloads (free tier)
  - Acceptable performance for large workloads
    - No matter how large

- LBaaS should
  - Use as little resources as possible for small workloads
  - Have the resources to handle huge workloads

- Existing Octavia topologies have per LB
  - **One** active VM
    - Too small for large workloads? Too much for free tier? Maybe use containers?
  - *(optionally)* One **idle** standby VM
    - 50% utilization
Introducing:
Active-Active, N+1 Topology

- $N$ Amphorae, all active
  - Can handle large load

- 2-stage VIP traffic splitting
  1) Distributor to Amphorae
  2) Amphora to Back-end servers

- Standby Amphora
  - Ready to replace a failed Amphora
    - Takes over the load
  - Failed Amphora recreated as standby
  - Can generalize to more than one standby
    - $N + k$

Disclaimer: Active-Active topology is still a draft blue-print 😞 ( + demo code 😊 )
The Distributor

• Equivalent to a GW router
  – Should have similar high availability attributes
  – Needs to handle entire VIP load
  – HW is a good match

• “Not so smart” LB
  – More like ECMP
  – L3 only, but **must have per-flow affinity**
    • Cannot break TCP

• Could be shared (multi-tenant)
  – SSL termination is only at Amphora

• Could be DNS
  – If you have enough (public) IPs
Our SDN SW Distributor

- 1-arm Direct Routing
  - Co-located on same LAN as Amphorae
  - L2 forwarding
    - Replace own MAC with MAC of Amphora
  - Direct Server Return
    - Return traffic goes directly to GW
  - Amphorae do not advertise VIP

- OpenFlow rules (using groups)
  - Select Amphora by hash of SrcIP:Port

- OVS VM
  - Can be any OpenFlow switch
  - Multi-tenant
  - No HA for now 😞
Our SDN SW Distributor

$ sudo ovs-ofctl -O OpenFlow 15 dump-groups br-data
OFPST_GROUP_DESC reply (OF1.5) (xid=0x2):
group_id=1, type=select, selection_method=hash, fields(ip_src, tcp_src),
bucket=bucket_id:0, actions=set_field:fa:16:3e:95:86:06->eth_dst,IN_PORT,
bucket=bucket_id:1, actions=set_field:fa:16:3e:9d:c9:d2->eth_dst,IN_PORT,
bucket=bucket_id:2, actions=set_field:fa:16:3e:ef:97:60->eth_dst,IN_PORT
$

• OpenFlow rules (using groups)
  – Select Amphora by hash of SrcIP:Port

• OVS VM
  – Can be any OpenFlow switch
  – Multi-tenant
  – No HA for now 😞
Elastic LB – Auto Scaling

- Amphorae pool is an auto-scale group
  - Use **Heat** to manage Amphora stack
    - Octavia Compute Driver (similar to Nova Driver)
    - Being replaced with a **Cluster Manager Driver**
  - Manage cluster of \( N \) Amphorae
    - Detect & replace failed Amphorae
    - Add/remove Amphorae when overloaded/underloaded

- Use **Ceilometer** to monitor Amphorae

- Octavia controller still does all the work….
  - Configure each Amphora
  - Monitor Amphorae at the application level
    - *Do we need Ceilometer?*
  - Add/remove forwarding rules to **Distributor**
    - Need to handle Affinity!

**Disclaimer:**
Not even a blue-print yet 😒 (but demo code 😊)
IBM Cloud

• Based on open standards
• Several cloud offerings running OpenStack operating system
• A large scale of workloads
• Benefit of load-balancer
  – High-availability
  – Performance
• Benefit of ELB
  – Load-balancer HA
  – Accommodate more workloads
  – Allow pay-per-use (cost efficient)
Demo (screenshots)
https://www.youtube.com/watch?v=l302AURPVii
Demo Story

- Two web flower shops:
  - Red shop
  - Blue shop

- Each “shop” returns a flower page
  - Red or Blue flower
  - Different flower per back-end
  - Back-end IP inserted into page

- # of Amphorae doing LB for the red shop is auto-scaled by Heat (Ceilometer alarms)

- HAProxy injects Amphora ID
  - For demo purposes only
Back-end IP:

```
20.0.0.12
```

Amphora ID:

```
amphora_server: am-pusw-zmth6klpgf2g-roztyebjx
```
Back-end IP

unchanged

changed

Amphora ID

 amphora_server: am-pusw-1iacjmf72ff-n3eh3nlfli

@ [10.0.0.5]
2 Elastic Load-Balancers

$ neutron lbaas-loadbalancer-list

+---------------------------------+--------+----------------+-------------------------------+-----------------+-------+
| id                             | name   | vip_address    | provisioning_status          | provider        |
+---------------------------------+--------+----------------+-------------------------------+-----------------+-------+
| 6379f6f7-9c8b-459a-8469-30e5f08e7da5 | red_lb | 20.0.0.12      | ACTIVE                        | octavia         |
| d3ed8e66-7e35-48dc-8839-fb2768942dd6 | blue_lb| 30.0.0.11      | ACTIVE                        | octavia         |
Elastic Load Balancing Using Octavia

- Management network
- Red shop network
- Blue shop network
- Amphorae
- Distributor
### Stack Details: amphora-cluster_for_loadbalancer_id_d3ed8e66-7e35-48dc-8839-fb2768942dd6

<table>
<thead>
<tr>
<th>Stack Resource</th>
<th>Resource</th>
<th>Stack Resource Type</th>
<th>Date Updated</th>
<th>Status</th>
<th>Status Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>scaleup_policy</td>
<td>d7f5464f5f0de4f849e64a088af528b3d8b28d</td>
<td>OS: Heat: ScalingPolicy</td>
<td>2 hours, 10 minutes</td>
<td>Create Complete</td>
<td>state changed</td>
</tr>
<tr>
<td>cpu_alarm_low</td>
<td>9cb13922-8860-42e8-b304-528ab3e6b1e7</td>
<td>OS: Ceilometer: Alarm</td>
<td>2 hours, 10 minutes</td>
<td>Create Complete</td>
<td>state changed</td>
</tr>
<tr>
<td>scaledown_policy</td>
<td>c70007f90b0d34b4a143c3f3c06d65928d2e53</td>
<td>OS: Heat: ScalingPolicy</td>
<td>2 hours, 10 minutes</td>
<td>Create Complete</td>
<td>state changed</td>
</tr>
<tr>
<td>asg</td>
<td>8ef6dccc-4.765-fc21-9a5a-fa68a8500038</td>
<td>OS: Heat: AutoScalingGroup</td>
<td>2 hours, 10 minutes</td>
<td>Create Complete</td>
<td>state changed</td>
</tr>
<tr>
<td>cpu_alarm_high</td>
<td>b36e1f970-886a-400c-9496-35affb4d14e0</td>
<td>OS: Ceilometer: Alarm</td>
<td>2 hours, 10 minutes</td>
<td>Create Complete</td>
<td>state changed</td>
</tr>
</tbody>
</table>

- **Scale-up Policy**
- **Scale-down Alarm**
- **Scale-down Policy**
- **Amphorae Cluster**
- **Scale-up Alarm**
Scale-up Ceilometer Alarm:
- statistic: avg
- comparison_operator: gt
- type: threshold
- threshold: 40.0
- period: 120
- state: unknown/ok/alarm
- alarm_actions: Scale-up URL

Alarm fires when avg of cpu_util > 40% over 2 minutes
Scale-down Ceilometer Alarm:

- statistic: avg
- comparison_operator: lt
- type: threshold
- threshold: 10.0
- period: 120
- state: unknown/ok/alarm
- alarm_actions: Scale-dn URL

Alarm fires when avg of cpu_util < 10% over 2 minutes
Start the Stress...

Creating ping stress for 600 seconds against 20.0.0.12
ARPING to 20.0.0.12 from 20.0.0.11 via eth0
Sent 1 probe(s) (1 broadcast(s))
Received 1 replies (0 request(s), 0 broadcast(s))
Creating high stress for [1] more seconds (using port 13980)
Waiting (sleeping) for 600 seconds
Waiting (no stress) for [1] more seconds
Elastic Load-Balancers Under Stress

`cpu_util > 40%` (as specified in the alarm) – scale-up alarm triggered

A new Amphora VM will be added to the cluster (by Heat Engine)
Elastic Load-Balancers Stress Free

An existing Amphora VM will be removed from the cluster (by Heat Engine)

cpu_util < 10% (as specified in the alarm) – scale-down alarm triggered
Sample Run (simulated HTTPS load)

![Graph showing New Sessions Per Second (1000) over time with Amphorae and Sessions per second as variables.](image-url)

- **Y-axis:** New Sessions Per Second (1000)
- **X-axis:** Time
- **Legend:**
  - Green: Amphorae
  - Blue: Sessions per second
Equal Balancing at Each Level

```
stack@garda6 [1894] ~@devstack/SharedRepository/CIL/tools (master *)
$ for i in {1..100}; do curl -i 20.0.0.12 2>/dev/null | grep 'backend-server'; done | sort | uniq -c
  34 backend-server: 10.0.0.3
  33 backend-server: 10.0.0.4
  33 backend-server: 10.0.0.5

stack@garda6 [1895] ~@devstack/SharedRepository/CIL/tools (master *)
$ for i in {1..100}; do curl -i 20.0.0.12 2>/dev/null | grep 'amphora'; done | sort | uniq -c
  20 amphora_server: am-pusw-2hpzggzsnomu-b5ewlqrj2f
  22 amphora_server: am-pusw-mikvap7iokv-x6zxxpdpb7
  31 amphora_server: am-pusw-xlzy4aesdwxj-6qrbagv2dz
  27 amphora_server: am-pusw-zmth6klpgf2g-rozteybsjx

stack@garda6 [1896] ~@devstack/SharedRepository/CIL/tools (master *)
```
End of Demo
https://www.youtube.com/watch?v=l302AURPVil
Amphora Containers

• Lower cost per LB instance
  – Containers use less resources
  – Can be packed tighter

• Container less powerful
  – Horizontal scaling allows large workloads

• Faster creation
  – No need for +1 ?

• Better availability
  – Larger $N \rightarrow$ better spread
  – Container migration
Thank you.

Questions?

Blueprints: (active-active-topology, active-active-distributor)
https://review.openstack.org/#/c/234639