

Learnings from scaling IroniC at Yahoo

Arun S A G
saga@yahoo-inc.com
zer0c001 on freenode

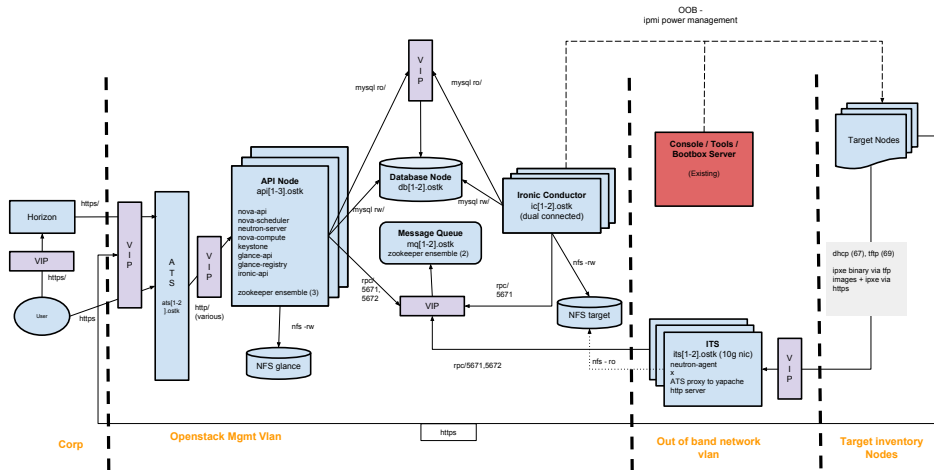
Yahoo Inc

May 08, 2017

<https://github.com/sagarun/presentations/>

Background and Architecture

Cluster Architecture



Migrating to Ironic

- ▶ Import nodes from old system into Ironic

Migrating to Ironic

- ▶ Import nodes from old system into Ironic
- ▶ Create neutron port for the node

Migrating to Ironic

- ▶ Import nodes from old system into Ironic
- ▶ Create neutron port for the node
- ▶ If the node is already active in the old system, 'fake' boot it with fake_pxe driver

Migrating to Ironic

- ▶ Import nodes from old system into Ironic
- ▶ Create neutron port for the node
- ▶ If the node is already active in the old system, 'fake' boot it with fake_pxe driver
- ▶ Once everything is successful, switch to pxe_ipmitool driver

Ironic

Ironic Setup

- ▶ Ironic API runs behind Apache Server

Ironic Setup

- ▶ Ironic API runs behind Apache Server
- ▶ Ironic Conductors(2)

What could possibly go wrong?

What could possibly go wrong?

- ▶ Boots started to fail

What could possibly go wrong?

- ▶ Boots started to fail
- ▶ **Ironic API calls took too long**

Solutions

- ▶ Sync_Power_State periodic task

Solutions

- ▶ Sync_Power_State periodic task
- ▶ Increase the number of Ironi Conductors

Solutions

- ▶ Sync_Power_State periodic task
- ▶ Increase the number of Ironi Conductors
- ▶ Run multiple conductors on the same host

Neutron

Neutron setup

- ▶ All 3 API servers run neutron-server

Neutron setup

- ▶ All 3 API servers run neutron-server
- ▶ 24 API/RPC workers

Neutron setup

- ▶ All 3 API servers run neutron-server
- ▶ 24 API/RPC workers
- ▶ 4 neutron dhcp agents

Neutron setup

- ▶ All 3 API servers run neutron-server
- ▶ 24 API/RPC workers
- ▶ 4 neutron dhcp agents
- ▶ All networks/subnets are managed by all 4 agents (HA)

Neutron setup

- ▶ All 3 API servers run neutron-server
- ▶ 24 API/RPC workers
- ▶ 4 neutron dhcp agents
- ▶ All networks/subnets are managed by all 4 agents (HA)
- ▶ ISC DHCPD driver instead of dnsmasq

What is sync state?

A tale of two drivers

- ▶ OMShell driver

A tale of two drivers

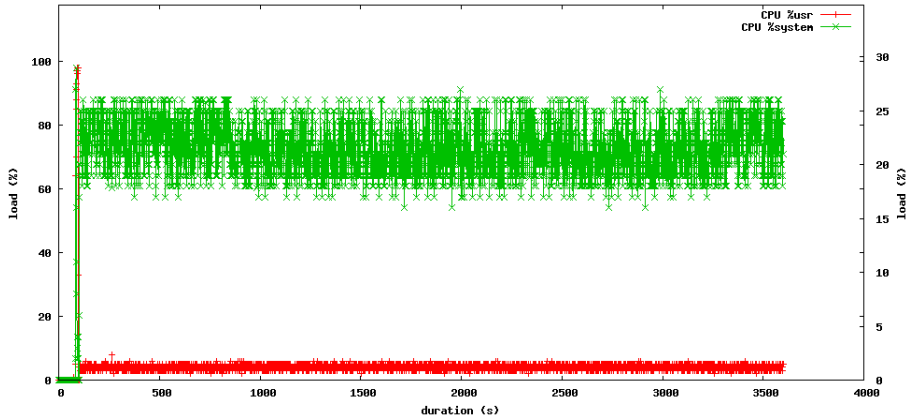
- ▶ OMShell driver
- ▶ Pypureomapi driver

OMShell

```
-bash-4.1$ omshell
> server 127.0.0.1
> port 7911
> key keyname secret
> connect
obj: <null>
> new host
obj: host
> set hardware-address = 00:1c:1a:1d:10:54
obj: host
hardware-address = 00:1c:1a:1d:10:54
> open
obj: host
hardware-address = 00:1c:1a:1d:10:54
ip-address = 0a:d7:a6:b1
name = "hostname.yahoo.com-0"
hardware-type = 00:00:00:01
>remove
```

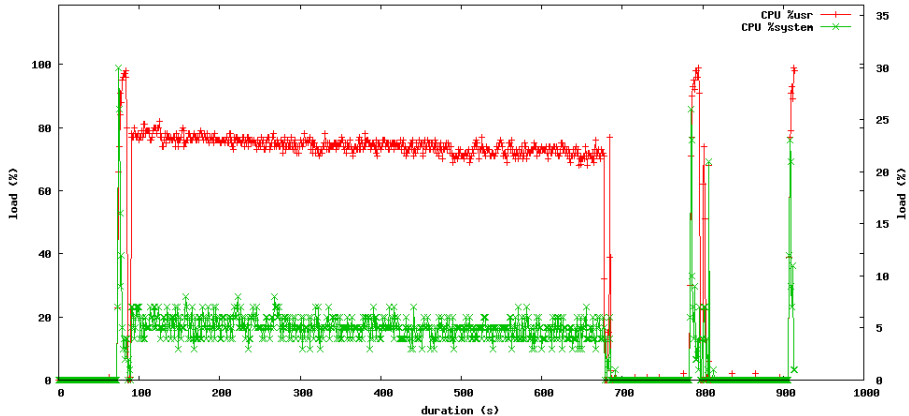
Sync State with OMShell

LOAD for cmd: /home/y/share/venv/openstack_yahoo_neutron/bin/python2 /home/y/share/venv/openstack_yahoo_neutron/bin/neutron-dhcp-agent --conf



Sync State with PypureOMAPI

LOAD for cmd: /home/y/share/venv/openstack_yahoo_neutron/bin/python2 /home/y/share/venv/openstack_yahoo_neutron/bin/neutron-dhcp-agent --conf



Where do we go from here?

- ▶ ISC DHCPD restarts are not ideal

Where do we go from here?

- ▶ ISC DHCPD restarts are not ideal
- ▶ VIP thinks dhcpd is down whenever it restarts

Where do we go from here?

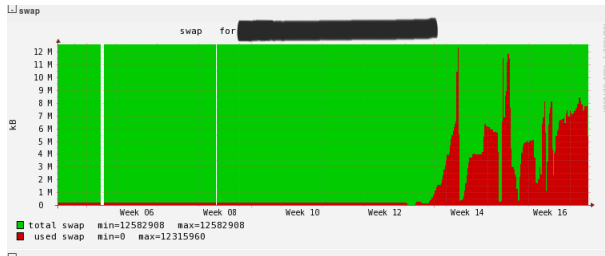
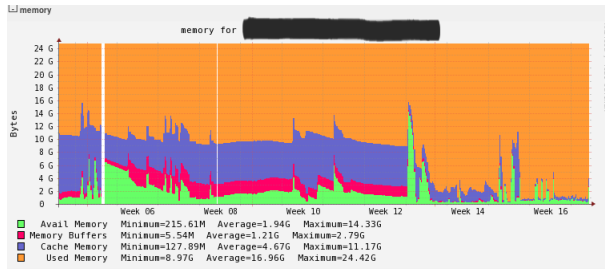
- ▶ ISC DHCPD restarts are not ideal
- ▶ VIP thinks dhcpd is down whenever it restarts
- ▶ Move to Kea DHCP Server if possible

Density Test

When did things started to break?

- ▶ At 24500 nodes, API servers started swapping

Swap and memory usage on API nodes



Memory usage

- ▶ Neutron the biggest user of memory: 1.4 GB per process

Memory usage

- ▶ Neutron the biggest user of memory: 1.4 GB per process
- ▶ Subnets: 2500, Ports: 43000

Memory usage

- ▶ Neutron the biggest user of memory: 1.4 GB per process
- ▶ Subnets: 2500, Ports: 43000
- ▶ Easy fix: Reduce number of `api_workers` and `rpc_workers`

Memory usage

- ▶ Neutron the biggest user of memory: 1.4 GB per process
- ▶ Subnets: 2500, Ports: 43000
- ▶ Easy fix: Reduce number of api_workers and rpc_workers
- ▶ Long Term Fix: Investigate memory usage, isolate neutron

Learnings

Learnings

- ▶ Do a **density** and **scale** testing before taking on production

Learnings

- ▶ Do a **density** and **scale** testing before taking on production
- ▶ Avoid spawning processes, try and use native python libraries whenever possible

Learnings

- ▶ Do a **density** and **scale** testing before taking on production
- ▶ Avoid spawning processes, try and use native python libraries whenever possible
- ▶ Pay attention to periodic tasks

Learnings

- ▶ Do a *density* and *scale* testing before taking on production
- ▶ Avoid spawning processes, try and use native python libraries whenever possible
- ▶ Pay attention to periodic tasks
- ▶ Be prepared to scale horizontally

Learnings

- ▶ Do a **density** and **scale** testing before taking on production
- ▶ Avoid spawning processes, try and use native python libraries whenever possible
- ▶ Pay attention to periodic tasks
- ▶ Be prepared to scale horizontally
- ▶ Pay attention to number of workers,conductors,rpc_workers

Learnings

- ▶ Do a **density** and **scale** testing before taking on production
- ▶ Avoid spawning processes, try and use native python libraries whenever possible
- ▶ Pay attention to periodic tasks
- ▶ Be prepared to scale horizontally
- ▶ Pay attention to number of workers,conductors,rpc_workers
- ▶ Don't forget to have fun :)

Questions