OpenStack Performance

Ayal Baron and Doug Williams
abaron@redhat.com, dougw@redhat.com
November 6, 2013
Topics

- Conceptualizing OpenStack Performance
- Foundation
  - Keystone Performance
- OpenStack Nova
  - KVM Performance
  - Resource Over-commit
  - Nova Instance Storage – Boot Times and Snapshots
  - Nova Scheduler
- OpenStack Cinder
  - Direct storage integration with QEMU
  - Glusterfs Performance Enhancements in Havana
- Swift Performance
  - Swift and Blocking IO
- What's Next
Background

- Talk reflects work-in-progress
- Includes results from:
  - RHEL-OSP 4 (Havana)
  - RHOS 3 (Grizzly)
  - RHOS 2 (Folsom)
- Items not included in presentation
  - Neutron
  - Heat and most provisioning use-cases
Conceptualizing OpenStack Performance
High Level Architecture

- Modular architecture
- Designed to easily scale out
- Based on (growing) set of core services
Control Plane vs Data Plane

Control Plane
- Create/Delete/Start/Stop/Attach/Detach
- Performance Dictated by OpenStack

Data Plane
- Workloads in steady-state operation
- Performance dictated by components managed by OpenStack

<table>
<thead>
<tr>
<th>Control Plane</th>
<th>Data Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Provisioning</strong></td>
<td><strong>Steady State</strong></td>
</tr>
<tr>
<td>Heat</td>
<td>Nova</td>
</tr>
<tr>
<td>Lifecycle ops</td>
<td>VMs</td>
</tr>
<tr>
<td>Cinder</td>
<td>Instance Storage</td>
</tr>
<tr>
<td>Lifecycle ops</td>
<td>Volumes</td>
</tr>
<tr>
<td>Neutron</td>
<td>Networks</td>
</tr>
<tr>
<td>Lifecycle ops</td>
<td>Swift</td>
</tr>
<tr>
<td>Glance</td>
<td>Objects &amp; Containers</td>
</tr>
<tr>
<td>Images</td>
<td></td>
</tr>
</tbody>
</table>

- Glance
- Neutron
- Cinder
- Nova
Foundational Elements
- Each service has associated databases
- Extensive use of messaging for integration
- Keystone as common identity service
Control Plane and Data Plane Technologies

**Control Plane**
- Nova
  - Lifecycle ops
- Cinder
  - Lifecycle ops
- Neutron
  - Lifecycle ops
- Glance
  - Images
- Keystone
- Ceilometer
- Database
- Messaging

**Data Plane**
- VMs
  - Instance Storage
- Volumes
- Networks
- Objects & Containers

**Foundational Services**
- LDAP
- MariaDB, MySQL, Galera
- QPID, RabbitMQ

**Additional Technologies**
- KVM, Xen, ESXi
- Linux (RHEL)
- XFS, RoC ...
- Glusterfs, Ceph, NetApp, EMC, SolidFire ...
- LinuxBridge, OVS, GRE, VXLAN, Nicera, BigSwitch ...
- Cisco Nexus, other switches

OpenStack Havana Python 2.7
Control Plans and Data Plane Technologies
Technologies Used in Red Hat's Offering (RHEL-OSP 4)

Note: Italicics are examples of partner technologies
Factors influencing control plane performance demands

- Size of Cell in VM's
- Dynamic nature of workload
  - Provisioning operations per hr
  - Continuous Integration
  - Dev / Test
  - Autoscaling Production Workloads
  - Static Production Workloads

Provisioning frequency
Foundational Elements
Keystone Performance

Keystone findings - UUID in Folsom

- Chatty consumers: Multiple calls to keystone for new tokens

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizon Login</td>
<td>3 Tokens</td>
</tr>
<tr>
<td>Horizon Image page</td>
<td>2 Tokens</td>
</tr>
<tr>
<td>CLI (nova image-list)</td>
<td>2 Tokens</td>
</tr>
</tbody>
</table>

- Database grows with no cleanup
  - As tokens expire they should eventually get removed
    - Should help with indexing
    - For every 250K rows response times go up 0.1 secs
  - Can be addressed via cron job
    - `keystone-manage token_f ush`
Keystone
Inefficiencies in CLI due to Python libraries

Inefficiencies in CLI vs curl calls
- nova image-show
  - Executes in 2.639s
- curl -H " "
  - Executes in 0.555s
- Tracing of CLI shows that python is reading the data one byte at a time
  - Known httpplib issue in the python standard library

- Next steps for testing are to move to Havana and PKI tokens
Nova
Understanding Nova Compute Performance
KVM SPECvirt2010: RHEL 6 KVM Post Industry Leading Results

Steady-State Performance of Nova Compute Nodes Strongly Determined by Hypervisor
- For OpenStack this is typically KVM
- Good news is RHEL / KVM has industry leading performance numbers.

http://www.spec.org/virt_sc2010/results/
Understanding Nova Compute Performance

SPECvirt2010: Red Hat Owns Industry Leading Results

Best SPECvirt_sc2010 Scores by CPU Cores

(As of May 30, 2013)

Comparison based on best performing Red Hat and VMware solutions by cpu core count published at www.spec.org as of May 17, 2013. SPEC® and the benchmark name SPECvirt_sc2010® are registered trademarks of the Standard Performance Evaluation Corporation. For more information about SPECvirt_sc2010, see www.spec.org/virt_sc2010/.

OpenStack sweet spot

oVirt / RHEV sweet spot
Understanding Nova Compute Performance

Guest Performance

- Expect the similar out of the box performance as RHEL / KVM
  - Added tuned virtual-host to the Nova compute node configuration
  - RHOS generates its own XML file to describe the guest
    - About the same as virt-manager
    - Of course smart storage layout is critical

Tuning for performance

- Common for OpenStack and standalone KVM
  - Big Pages, NUMA, tuned profiles
- Optimizations for standalone KVM not currently integrated into OpenStack
  - SR-IOV, process-level node bindings
Nova Out of the Box Performance Comparison with Standalone KVM Results

RHOS vs libvirt/KVM

java workload

bops

1vm 4vm 8vm 12vm

RHOS Libvirt/KVM untuned
Nova Compute
Resource Over-Commit

Nova default configuration has some aggressive over commit ratios

- CPU has an over commit of 16
  - The scheduler will need multiple suggestions based on the instance workload

- Memory over commit is a much lower 1.5
  - Again depends on the workload
  - Anything memory sensitive falls off the cliff if you need to swap
Nova Compute
Understanding CPU Over-Commit

RHOS Java workload vm scaling
Impact of overcommitting CPU on aggregate throughput

However beware of cliff in excessive paging
Nova Compute
Ephemeral Storage

Look at ephemeral storage configuration

- Help determine guidelines for balancing ephemeral storage performance vs cost / configuration
  - Trade-off between footprint (number of drives) and performance
    - Initial cost / configuration, rack space 1U vs 2U, Power / cooling
  - How does network based storage perform
    - Need to ensure proper network bandwidth

Configuration for tests

- Each instance uses a different image
- Hardware configs:
  - Single system disk
  - Seven disk internal array
  - Fiber channel SSD drives
Nova Boot Times

Nova Boot Times (Multiple Images) - Folsom

virt-host profile

average boot time (secs)

2vm
16vm

System Disk  Local RAID  Cinder (FC SSD)
Impact of storage config on Nova Boot Times
Local RAID and Cinder

Nova Boot Times (Multiple Images)

- System Disk
- Local RAID
- Cinder (FC SSD array)
Nova Compute
Ephemeral Storage Snapshots Performance

RHOS Concurrent Snapshot Timings
(qemu-img convert only)

Backimg image and qcow =>
new image

• Via qemu-img convert
  mechanism
• Written to temporary snap
directory
• This destination is a tunable
Impact of Storage Configuration on Snapshots

RHOS Snapshot Timings (qem-img convert only) - Grizzly

Avg. Snapshot Time (secs)

1 snapshots
2 snapshots

11
15

Rhos 7 dsk lun
Impact of Adding Guests on Total YCSB Throughput

- **m1.medium**: 12.8 x Baseline, 61% @ 21 VMs
- **m1.large**: 5 x Baseline, 50% @ 10 VMs
- **m1.xlarge**: 3.3 x Baseline, 50% @ 5 VMs

**Nova Compute Single-node Scalability**

- **Medium**: 21 guests
- **Large**: 10 guests
- **Xlarge**: 5 guests

**Throughput**

- **Baseline Throughput**: single guest
- **Total Throughput**: 500000

---

26
Per-Guest Performance Degradation Due to Over-Subscription

Impact of Adding Guests on Average YCSB Throughput

Average Throughput (ops/sec)

- m1.medium
- m1.large
- m1.xlarge

Baseline Throughput
- Single guest

Average Throughput
- Medium = 21 guests
- Large = 10
- Xlarge = 5
Per-Guest Performance Degradation Due to Over-Subscription

Impact of Adding Guests on YCSB Latency

![Chart showing the impact of adding guests on YCSB latency. The chart compares the average app latency per guest across different instance sizes (m1.medium, m1.large, m1.xlarge) to the baseline latency. The chart indicates that adding guests leads to an increase in latency, with the highest increase observed on m1.medium.](image-url)
The default Nova scheduler assigns based on free memory
- Not much concern about other system resources (CPU, memory, IO, etc)
- You can change / tune this
- Be aware if you have machines with different configurations
Nova Scheduler – Heterogeneous Memory Config
Example of Results After Scheduler Tuning

Nova Scheduler Instance Placement at 20 vm launch increments

scheduler_host_subset_size=4

<table>
<thead>
<tr>
<th>Instance Count</th>
<th>1-20</th>
<th>21-40</th>
<th>41-60</th>
<th>61-80</th>
<th>81-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>24cpu/48GB</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>24VMs</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>24cpu/96GB</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>24cpu/96GB</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>
Cinder
Cinder QEMU Storage Integration

- RHEL 6.5 and RHEL-OSP 4 (Havana) now includes tight QEMU integration with Glusterfs and Ceph clients

- Benefits:
  - Direct QEMU integration avoids unnecessary context switching
  - One client per guest configuration may help alleviate client-side performance bottlenecks.

- Integration model
  - QEMU includes hooks to dynamically link with storage vendor supplied client
  - Delivered in a manner than preserved separation of concerns for software distribution, updates and support
  - OpenStack and Linux, including QEMU, provided by Red Hat
  - Storage client libraries provided and supported by respective storage vendors
    - Libgfapi for Glusterfs (RHS) supported by Red Hat
    - Librados for Ceph supported by InkTank
Glusterfs Support for Cinder in Havana With RHEL 6.5

**Configuration**
- 1 Compute node (2s x86)
- 2 Storage nodes (2s x86)
  - 12 LFF drives, RAID06 with WBC
  - 2-way replication across nodes
- 10gE Network

**Workload**
- 8 vLUNS at 48G each
- 64K IO stream per vLUN
- Configured to avoid both client-side and server-side caching

**Conclusion:** RHS runs at hardware limited performance
- Sequential performance limited by network
- Random IO performance limited by drives
Swift
Swift Performance – Glusterfs as Pluggable Backend
Tuning Worker Count, Max Clients, Chunk Size

150 30 MB Objects Transferred Sequentially
10 Ge, Concurrent w Four Other Clients (3K, 30K, 300K, 3M)

Issue: Blocking IO for filesystem operations with Python Eventlet packaged used by OpenStack

Recommended Tuning:
Workers: 10 (was 2)
Max Clients: 1 (was 1024)
Chunk Size: 2M (was 64K)

Reducing Max Clients also helps vanilla Swift
Wrap Up
Wrap Up

- OpenStack is a rapidly evolving platform
- Out of the box performance is already pretty good
  - Need to focus on infrastructure out of the box configuration and performance
- Still just scratching the surface on the testing
  - Control plane performance via emulated Vms
  - Neutron performance (GRE, VXLAN)
  - Ceilometer
  - Performance impact of Active-Active foundational components (DB, messaging)
Questions