Running Hadoop on the Cloud

Javier López Cacheiro, A. Simón, E. Freire, I. Díaz, A. Feijoo, P. Rey and C. Fernández

Outline

- Introduction
- KVM and VM OS optimizations
- Whirr
- Hadoop results
- Conclusions

Introduction

- CESGA has created hadoop startup scripts for OpenNebula.
- These scripts will be used by internal users to instantiate hadoop clusters on demand.
- It's possible to create **N+1** clusters: 1 **Master** node and **N Slaves** (*tasktrackers* and *datanodes*).
- As example to start 101 hadoop cluster:
  - *start-hadoop -n 100*
- To stop hadoop cluster:
  - *stop-hadoop <cluster_id>*

Introduction

- CESGA is using KVM hypervisor for its cloud infrastructure.
- We have notice that KVM configuration should be tuned up to increase hadoop performance (XenServer already implements these configuration changes by default).
- After these changes we have compared our cluster performance with an commercial provider (Amazon).

KVM and VM OS optimizations

- SL6 KVM version is not optimized by default to run big data.
- We have included these hacks to increase IO and CPU performance.
- These changes were applied to OpenNebula VM templates.
  - Activate **virtio** module for disk and network interfaces.
  - Use small RAW vanilla images instead QCOW2.
  - Generate extra disk storage ans swap locally for each instance.
  - Disable KVM disk cache.
  - Change VM Operating System vd*x* disk scheduling algorithm to deadline.
  - Set *blockdev --setra 8196 /dev/vdx.*
  - Use *host-passthrough* CPU mode.

KVM and VM OS optimizations

```
DISK=[
  BUS="virtio",
  CACHE="none",
  DRIVER="raw",
  IMAGE_ID="hadoop",
  TARGET="vda",
  TYPE="OS" ]

DISK=[
  BUS="virtio",
  CACHE="none",
  FORMAT="ext4",
  SIZE="31480",
  TARGET="vdc",
  TYPE="fs" ]

NIC=[
  MODEL="virtio",
  NETWORK_ID="8" ]

RAW=[
  DATA="<cpu mode='host-passthrough'/>",
  TYPE="kvm" ]
```

KVM and VM OS optimizations

- From VM contextualisation script:

```
#   VM optimizations taken from tuned:
#   From tuned-adm profile virtual-guest
#   and readahead extended to 4KB from 512 bytes
for disk in vda vdb vdc; do
        echo deadline > /sys/block/${disk}/queue/scheduler
        blockdev --setra 8196 /dev/${disk}
done
```

KVM and VM OS optimizations

- From VM /etc/sysctl.conf file:

```
# Minimal preemption granularity for CPU-bound tasks:
# (default: 1 msec#  (1 + ilog(ncpus)), units: nanoseconds)
kernel.sched_min_granularity_ns = 10000000

# This option delays the preemption effects of decoupled workloads
# and reduces their over-scheduling. Synchronous workloads will still
# have immediate wakeup/sleep latencies.
kernel.sched_wakeup_granularity_ns = 15000000

# swapping low.  It's usually safe to go even lower than this on systems with
# server-grade storage.
vm.swappiness = 30

# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 40
```

Apache Whirr

- A cloud-neutral way to run clusters
- It is built on top of Apache jclouds API
- jclouds supports 30 cloud providers including Amazon, Rackspace, OpenStack and CloudStack
- Work in progress to support OpenNebula
- Example:

  whirr launch-cluster --config hadoop-ec2.properties

Hadoop ec2 contrib tools

- Tools included with hadoop distribution
- Legacy whirr code based on shell scripts instead of jclouds
- Easier to customize
- Example:

  hadoop-ec2 launch-cluster hadoop-31 30

Hadoop results

- We have startup the same number of VMs for two different clouds.
  - Private cloud based on OpenNebula framework (CESGA)
  - Public cloud (Amazon).
- We have repeated each test 3 times to instantiate different cluster sizes (10, 21, 51...)

Hadoop results

# VMs	Amazon EBS Whirr (m1.small)	Amazon EBS without Whirr (m1.small)	CESGA ON (small)
10 VMs	14m51s	3m11s	5m40s
21 VMs	27m50s	2m58s	4m53s
51 VMs	**NA**	***9m43s***	13m03s
101 Vms	**NA**	***4m57s***	18m31s

- Small instances (1MB, 1 CPU)

Conclusions

OpenNebula

- Tuning needed to reduce the 2 hours deployment time.
- Combined usage of NFS datastore and local deployment reduces deployment time to 15 min (101 nodes).

Conclusions

Amazon EC2

- Amazon limits the request rate → whirr/jclouds produce too many requests.
- Whirr is not able to launch cluster larger than 20 nodes in Ireland region.
- Hadoop src/contrib/ec2 scripts (legacy whirr) only support very old hadoop versions.
- Customized launch script (based on the ON/Fedcloud one) solved these issues.
- Impressive launch times when using EBS as root: as low as 5 min (101 nodes).
- Unable to launch a complete cluster: from 51 nodes we always get several problematic nodes (usually caused by an internal problem in Amazon with the security group).

Thank You For Your Attention!
Questions?