Elastic Data Processing
Big Data Frameworks Course
Spring 2015

Lirim Osmani
What do they have in-common.!?
Workloads are evolving
Server vs Cloud

TRADITIONAL WORKLOADS

- Typically resides on a single large server machine;
- Cannot tolerate any downtime;
- Needs expensive high availability tools;
- Application scales up rather than out;

CLOUD WORKLOADS

- Workload resides on multiple Virtual Machines;
- Tolerates VM failure – if one fails, another quickly replaces it;
- Fault tolerance often built into workload;
- Application scales out rather than up;
OpenStack Architecture

- Modular architecture;
- Designed to easy scale out;
- Based on a set of core services;
Why OpenStack?

- Brings Public Cloud like capabilities to the datacenter;
- Provides massive on-demand scalability;
  - 1,000’s → 10,000’s of VMs;
- It’s OPEN!;
  - Provides flexibility to customize and interoperate;
- Community development = higher “feature velocity”;
  - Features and functions you need, faster to spin a proof-of-concept;
-Constantly “under development”…..Achilles heel;
OpenStack Data Processing Sahara

- Provides a scalable data processing stack and associated management interfaces;
- Provision and operate Hadoop/Spark clusters;
- Schedule and operate Hadoop/Spark jobs;
- Self-service provisioning of clusters;
- Utilization of compute capacity for bursty workloads;
- Run big-data workloads in few clicks without expertise in cluster operations;
Sahara Integration with OpenStack

- Sahara
  - User
  - Glance
  - Nova
  - Keystone
  - Swift

Graphical representation:
- Sahara connects to User and Glance.
- Sahara also connects to Nova through a REST API.
- Sahara communicates with Keystone to authenticate user.
- Sahara interacts with Swift for Hadoop jobs.
- Sahara creates VMs and gets VM info from Nova.
- Sahara uses Glance for Hadoop image.
Sahara Features - Jobs

- EDP – API to execute MapReduce jobs without exposing details of underlying infrastructure (similar to AWS EMR)
  - Pluggable data sources: Swift, HDFS, Ceph
  - Supported job types: Jar, Pig, Hive
- User-friendly UI or ad-hoc analytics queries based on Hive or Pig;
- Spark jobs on Spark standalone clusters;
- Access and Storage of job binaries/output in Swift or Sahara’s own database;
- Configuration of jobs at submission time;
- Execution of jobs on existing clusters;
Sahara Features – Cluster Operations

- Hadoop/Spark clusters operation and provisioning
  - Templates for Hadoop/Spark cluster configuration
  - REST API for cluster startup and operations
  - Manual cluster scaling (add/remove nodes)
  - Swift(storage) integration
- UI integrated in dashboard (Horizon)
- Plugin mechanism for integration with different Hadoop distributions – Vanilla apache, Hortonworks, Cloudera, Spark
Sahara - General workflow

- Sahara provides two level of abstractions for API and UI for: cluster provisioning and analytics-as-a-service
- For cluster provisioning:
  - Select cluster version (Hadoop, Spark);
  - Select base image with or without pre-installed software;
  - Define cluster configuration, including size and topology and setting the different type of cluster parameters (e.g. heap size);
  - Provision the cluster: Sahara will provision VMs, install and configure your cluster (Hadoop, Spark);
  - Operation of the cluster: add/remove nodes;
  - Terminate the cluster when it’s not needed anymore;
For analytics as a service the workflow is:

• Select one of the predefined cluster versions;

• Configure the job;
  – Choose type of job: pig, hive, jar-file, etc;
  – Provide the job script source or jar location;
  – Select input and output data location;
  – Select location for logs;

• Set the limit for the cluster size;

• Execute the job:
  – All cluster provisioning and job execution will happen transparently to the user;
  – Cluster will be removed automatically after the job completion;

• Get the results of computation (e.g. Swift)
Sahara - General workflow – Templates

- **Node Group Template:**
  - Describes a group of nodes within a cluster;
  - Contains a list of Hadoop(Spark) processes that will be launched on each instance in a group;
  - Encapsulates hardware parameters (flavor) for the node VM and configuration for Hadoop(Spark) processes running on the node;

- **Cluster Template:**
  - Brings node group templates together to form a Cluster;
  - This template defines what Node Group will be included and how many instances will be created in each;

- **The Cluster:**
  - Represent the Cluster itself with pre-installed images software ready for deployment;
Flexible Scheduler Driver with filters:

- CoreFilter;
- RamFilter;
- RetryFilter;
- AvailabilityZoneFilter;
- ComputeFilter;
- ComputeCapabilitiesFilter;
- ImagePropertiesFilter;
- ServerGroupAffinityFilter;
- ServerGroupAntiAffinityFilter;
Scheduler examples

- **AvalabilityZoneFilter:**
  - return hosts where `node_availability_zone` name is the same as the one requested;

- **RamFilter:**
  - return hosts where `(free ram * ram_allocation_ratio)` is greater than requested ram;

- **ComputeFilter:** `return hosts where asked instance_type (with extra specs) match capabilities;`
# Elastic Data Processing – How?

## Metrics in Ceilometer

**Nova**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
<th>Resource</th>
<th>Origin</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
<td>Gauge</td>
<td>instance</td>
<td>inst ID</td>
<td>both</td>
<td>Duration of instance</td>
</tr>
<tr>
<td>instance:&lt;type&gt;</td>
<td>Gauge</td>
<td>instance</td>
<td>inst ID</td>
<td>both</td>
<td>Duration of instance &lt;type&gt; (openstack types)</td>
</tr>
<tr>
<td>memory</td>
<td>Gauge</td>
<td>MB</td>
<td>inst ID</td>
<td>notification</td>
<td>Volume of RAM in MB</td>
</tr>
<tr>
<td>cpu</td>
<td>Cumulative</td>
<td>ns</td>
<td>inst ID</td>
<td>pollster</td>
<td>CPU time used</td>
</tr>
<tr>
<td>cpu_util</td>
<td>Gauge</td>
<td>%</td>
<td>inst ID</td>
<td>pollster</td>
<td>Average CPU utilisation</td>
</tr>
<tr>
<td>vcpus</td>
<td>Gauge</td>
<td>vcpu</td>
<td>inst ID</td>
<td>notification</td>
<td>Number of VCPUs</td>
</tr>
<tr>
<td>disk.read.requests</td>
<td>Cumulative</td>
<td>request</td>
<td>inst ID</td>
<td>pollster</td>
<td>Number of read requests</td>
</tr>
<tr>
<td>disk.write.requests</td>
<td>Cumulative</td>
<td>request</td>
<td>inst ID</td>
<td>pollster</td>
<td>Number of write requests</td>
</tr>
<tr>
<td>disk.read.bytes</td>
<td>Cumulative</td>
<td>B</td>
<td>inst ID</td>
<td>pollster</td>
<td>Volume of read in B</td>
</tr>
<tr>
<td>disk.write.bytes</td>
<td>Cumulative</td>
<td>B</td>
<td>inst ID</td>
<td>pollster</td>
<td>Volume of write in B</td>
</tr>
<tr>
<td>disk.root.size</td>
<td>Gauge</td>
<td>GB</td>
<td>inst ID</td>
<td>notification</td>
<td>Size of root disk in GB</td>
</tr>
<tr>
<td>disk.ephemeral.size</td>
<td>Gauge</td>
<td>GB</td>
<td>inst ID</td>
<td>notification</td>
<td>Size of ephemeral disk in GB</td>
</tr>
<tr>
<td>network.incoming.bytes</td>
<td>Cumulative</td>
<td>B</td>
<td>iface ID</td>
<td>pollster</td>
<td>number of incoming bytes on the network for a VM interface</td>
</tr>
<tr>
<td>network.outgoing.bytes</td>
<td>Cumulative</td>
<td>B</td>
<td>iface ID</td>
<td>pollster</td>
<td>number of outgoing bytes on the network for a VM interface</td>
</tr>
<tr>
<td>network.incoming.packets</td>
<td>Cumulative</td>
<td>packets</td>
<td>iface ID</td>
<td>pollster</td>
<td>number of incoming packets for a VM interface</td>
</tr>
<tr>
<td>network.outgoing.packets</td>
<td>Cumulative</td>
<td>packets</td>
<td>iface ID</td>
<td>pollster</td>
<td>number of outgoing packets for a VM interface</td>
</tr>
</tbody>
</table>
Elastic Data Processing

Cluster Templates

<table>
<thead>
<tr>
<th>Name</th>
<th>Plugin</th>
<th>Hadoop Version</th>
<th>Node Groups</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>large-cluster</td>
<td>vanilla</td>
<td>2.4.1</td>
<td>control: 1, worker: 10</td>
<td></td>
</tr>
<tr>
<td>small-cluster</td>
<td>vanilla</td>
<td>2.4.1</td>
<td>worker: 3, control: 1</td>
<td></td>
</tr>
</tbody>
</table>

Displaying 2 items
Elastic Data Processing

- Support for external HDFS;
Elastic Data Processing

- Job Type selection;
Elastic Data Processing

Job re-launch, with new data and parameters

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>e3a0f9b6-cea2-4f37-b8f3-fd8491c8a291</td>
<td>SUCCEEDED</td>
<td>![Actions Menu]</td>
</tr>
</tbody>
</table>

Displaying 1 item
EDP – Command line interface

Image Management

$ sahara
...
Positional arguments:
<subcommand>
  image-add-tag    Add a tag to an image.
  image-list       Print a list of available images.
  image-register   Register an image from the Image index.
  image-remove-tag Remove a tag from an image.
  image-show       Show details of an image.
  image-unregister  Unregister an image.
Node group, cluster and job templates

$ sahara

node-group-template-create Create a node group...
node-group-template-delete Delete a node group...
nodel-group-template-list Print a list of available...
nodel-group-template-show Show details of a node...
cluster-template-create Create a cluster template.
cluster-template-delete Delete a cluster template.
cluster-template-list Print a list of available...
cluster-template-show Show details of a cluster...
job-template-create Create a job template.
job-template-delete Delete a job template.
job-template-list Print a list of job...
job-template-show Show details of a job...
EDP – Command line interface

Data sources and job binaries

$ sahara
...

<table>
<thead>
<tr>
<th>Subcommand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-source-create</td>
<td>Create a data source that provides job input receives job output.</td>
</tr>
<tr>
<td>data-source-delete</td>
<td>Delete a data source.</td>
</tr>
<tr>
<td>data-source-list</td>
<td>Print a list of available data...</td>
</tr>
<tr>
<td>data-source-show</td>
<td>Show details of a data source.</td>
</tr>
<tr>
<td>job-binary-create</td>
<td>Record a job binary.</td>
</tr>
<tr>
<td>job-binary-delete</td>
<td>Delete a job binary.</td>
</tr>
<tr>
<td>job-binary-list</td>
<td>Print a list of job binaries.</td>
</tr>
<tr>
<td>job-binary-show</td>
<td>Show details of a job binary.</td>
</tr>
</tbody>
</table>
$ sahara
...

<subcommand>
  cluster-create  Create a cluster.
  cluster-delete  Delete a cluster.
  cluster-list    Print a list of available clusters.
  cluster-show    Show details of a cluster.
  job-create      
  job-delete      Delete a job.
  job-list        Print a list of jobs.
  job-show        Show details of a job.
Thank you.