BESIII distributed computing and VMDIRAC

Xiaomei Zhang

Institute of High Energy Physics
BESIII CGEM Cloud computing Summer School
Sep 7~ Sep 11, 2015
• Two ways of scientific applications using cloud resources
  – VMDIRAC is an elastic way for the BESIII application to use cloud

• A real case: BESIII distributed computing
  – built up on DIRAC, VMDIRAC is a cloud extension
  – BESIII users use cloud through this platform
  – Demo: How to submit a job to Cluster and Grid, Cloud

• How VMDIRAC integrate cloud?
  – DIRAC workload management
  – VMDIRAC architecture and implementation
Run scientific applications on clusters

• The feature of Scientific applications
  – Enormous data processing with thousands of jobs to submit and run

• The most common way is to use resource manager to schedule these jobs to proper work nodes
  – PBS, HTCondor, LSF.....
Run scientific applications on clouds

• Build standalone virtual cluster over cloud
  – Everything built over VMs instead of physical machines
  – Transparent to end users
  – Easier, not so flexible
• Based on contextualization technique, we can automatically set up a virtual cluster with “one button”
  – “cernvm-online” in yesterday stefano’s talk and demo
Run scientific applications on clouds

- **On-demand usage**
  - Elastic way to use cloud
  - Don’t occupy resources before jobs are coming
    - Save money when you use commercial cloud
  - VMDIRAC is one of the way allowing to use clouds elastically
    - HTCondor + Cloud scheduler, elastiq
  - Need central task queue and cloud scheduler
BESIII distributed computing

- BESIII distributed computing system provides a way for BESIII physics users to use various distributed computing resource
  - Grid, Cluster, Cloud and Volunteer computing
  - more than 14 sites are joined
  - About 2000 cores CPU resources, 400 TB storage have been integrated
- 60K jobs have been submitted and run over distributed computing resources in recent three months
• Use CVMFS to deploy BESIII experiment software to remote sites
• The system is built up based on DIRAC
• VMDIRAC is a cloud extension of DIRAC
  – Able to integrate both private cloud and commercial cloud, eg. openstack, cloudstack, opennebula, etc
Authentication on BESIII distributed computing

• As a BESIII user, you are allowed to submit jobs to resources

• DIRAC use grid certificate to check if you belong to BESIII
  – First you need to get certificate from one of grid CA (Certification Authority)
    • IHEP CA is the only one in China (https://cagrid.ihep.ac.cn)
  – Second you have to register your certificate in BESIII VO(Virtual Organization)
    • https://voms.ihep.ac.cn

-bash-4.1$ voms-proxy-info -all
......
=== VO bes extension information ===
VO : bes
subject : /C=CN/O=HEP/OU=CC/O=IHEP/CN=Xiao mei Zhang
issuer : /C=CN/O=HEP/OU=CC/O=IHEP/CN=voms.ihep.ac.cn
attribute : /bes/Role=NULL/Capability=NULL
timeleft : 11:59:46
uri : voms.ihep.ac.cn:15001
Demo: How to submit jobs through DIRAC web portal

- Check the permission to use the resources
  - https://dirac.ihep.ac.cn

- Check the available resources
  - https://dirac.ihep.ac.cn:8444/DIRAC/CAS_Production/user/jobs/SiteSummary/display

- Submit a job to resources including cloud

- Monitor job running status

- Get the results from jobs
How to submit jobs to cloud through DIRAC client

• More complicated applications can use command line to submit jobs
  – Source DIRAC environment
  – Initialize your grid certificate to get permission
  – Prepare JDL files
  – dirac-wms-job-submit *.jdl
  – dirac-wms-job-get-output <jobID>

```python
[  
  Executable = "/bin/ls";
  JobRequirements =  
  [  
    CPUPtime = 86400;
    Sites = "CLOUD.CNIC.cn";
  ];
  StdOutput = "std.out";
  StdError = "std.err";
  OutputSandbox =  
  {  
    "std.err",
    "std.out"
  };
]
```
• **Distributed Infrastructure with Remote Agent Control**

• **History**
  – DIRAC project was born as the LHCb distributed computing project
  – Since 2010 DIRAC became an independent project

• **DIRAC has all the necessary components to build ad-hoc infrastructures for distributed computing as a framework**
  – Configuration, agents, services, user interface, databases
  – Allow to customize experiment-specific systems
DIRAC

- DIRAC allows to interconnect computing resources of different types as an **interware**
  - Grid
  - Standalone Cluster
  - Desktop grid
  - Cloud
DIRAC systems

- VMDIRAC is one of DIRAC systems
  - Workload management, Data management....
- Each system consists of similar components
  - services, agents, clients, databases
DIRAC systems

• Services
  – Passive components, permanently running, waiting for queries or requests

• Agents
  – light and active components which run as independent processes to fulfill one or several system functions
A case --- BESIII Transfer system

- Do mass transfers between remote sites
- The Components include:
  - Web interface
    - Request transfers
    - Monitor transfer status
  - Transfer agent
    - Get transfer tasks from DB
    - Start transfers
  - Request service
    - Get requests from users
  - DB
    - Record transfer requests and status
- VMDIRAC is another system in DIRAC, just more complicated
DIRAC workload management

- DIRAC is like a big cluster system over WAN
- Central task queue
  - User jobs are put into the task Queue
  - Job priorities are controlled with VO policies
- Pilot director
  - Connect with resource broker and submit proper pilots
  - Deal with heterogeneous resources
    - Every resource type needs a pilot director
- Match service
  - Cooperate with pilot, Match proper user jobs to resources
Push scheduling

- Two common ways to schedule jobs to resources
  - Push scheduling
  - Pull scheduling

- Push scheduling on clusters
  - User jobs is submitted to the local scheduler
  - Jobs are put into queues
  - Be arranged to WNs directly
• Pull scheduling with pilot paradigm on DIRAC
  – Instead of send use jobs to resources directly
  – Pilot jobs are sent to resource brokers (CE, PBS...) as normal jobs
  – Pilot jobs start job agents
  – Job agents do
    – occupy a resource
    – set up environment
    – pull jobs from central queue

– Advantages
  – Avoid failure of user jobs because of hardware problem
  – Easy to fit in different resource environment
Cloud differences

- Cloud is integrated into DIRAC in similar way, but with some differences
- Local job scheduler and resource manager
  - Cluster: pbs, condor
  - Grid: arcCE, creamCE
  - Cloud: no, only cloud manager to control VMs
- Static and dynamic resources
  - Static WNs in Cluster and Grid
  - No WNs before jobs are coming
Cloud integration

- “VM director” instead of “Pilot director”
  - start VMs, instead of submitting pilot jobs
- VMs at boot time start “pilot job”
  - This makes the instantiated VMs behave just as other WNs with respect to the DIRAC WMS
- VM scheduler need to manage dynamic virtual machines according to job situation
• Integrate Federated cloud into DIRAC
  – OCCI compliant clouds:
    • OpenStack, OpenNebula
    • CloudStack
    • Amazon EC2
• Main functions
  – Check Task queue and start VMs
  – Contextualize VMs to be WNs to the DIRAC WMS
  – Pull jobs from central task queue
  – Centrally monitor VM status
  – Automatically shutdown VMs when no jobs need
• Dirac server side
  – VM Scheduler – get job status from TQ and match it with the proper cloud site, submit requests of VMs to Director
  – VM Manager – take statistics of VM status and decide if need new VMs
  – VM Director – connect with cloud manager to start VMs
  – Image context manager – contextualize VMs to be WNs
Architecture and components

• VM side
  – VM monitor Agent—periodically monitor the status of the VM and shutdown VMs when no need
  – Job Agent—just like “pilot jobs”, pulling jobs from task queue

• Configuration
  – Use to configure the cloud joined and the image

• Work together
  – Start VMs
  – Run jobs on VMs
How to start VMs

- Users submit jobs through DIRAC interface
- Jobs recorded in task queue
- Cloud and VMs status recorded in the database
  - Cloud and images info get from DIRAC CS
  - DIRAC admin has uploaded the proper images in advance by cloud driver
  - VMs status is collected by VM managers
How to start VMs

- VM scheduler gets the list of jobs from the central Task Queues to run by matching the pending tasks with the available cloud.
- VM scheduler also checks if the existing VMs is enough with job info. If not enough and the maximum VMs threshold is not reached, then it submits a request for new VMs.
- The proper VM director connects with Cloud Manager through Cloud API such as rocci, libcloud, EC2.....
- Cloud manager gets the right image and image contextualization to start VMs.
How VMs run jobs

• The VM started is a “full” VM
  • At boot time, it is contextualized and starts DIRAC job Agent and VM Monitor Agent

• Job Agent
  • Cooperate with Job Matcher, and get proper jobs from task queue
  • Start the jobs and supervise their correct execution on the Virtual Machine resource
  • Report periodically to Job state update agent to update job status in DB
How VMs run jobs

• VM monitor agent
  • Report VM running state to VM manager
  • Monitor the CPU load of VM, and when the load is dropped a certain threshold, the VM manager will halt VMs
  • The VM monitor also will help asynchronously uploads the output data when the VM takes new execution
The contextualization mechanism

• The contextualization mechanism allows to configure the VM to start the pilot script at boot time
  – Avoid building and registering enormous number of images
• Ad-hoc image (no contextualization)
  • Install VMDIRAC staffs and security certificate in the images
  • Upload images to every cloud
• Contextualization supported for different cloud manager
  – Generic SSH
  – HEPIX OpenNebula
  – Cloudinit
VMDIRAC configuration

- Collect info of the available clouds and images
- “Endpoint” is used to define the cloud endpoint
- “Image” is to tell you the running env the VM is going to provide
  - Here “image” includes the selection of contextualization methods
VMDIRAC configuration

- “Running Pods” match “Endpoint” and “Image” to define various running conditions
  - Every cloud properly need the special image and contextualization methods
    - Security reason, special format, etc
- “Submit pools” is to collect the info of “Running Pods” for VM Scheduler to choose
VM monitor

- Central monitor
  - Collect info from VM monitor
  - Record in VM DB
- Local monitor
  - Go through web port of the clouds

<table>
<thead>
<tr>
<th>Image</th>
<th>RunningPod</th>
<th>EndPoint</th>
<th>Status</th>
<th>Endpoint VM ID</th>
<th>IP</th>
<th>Load</th>
<th>Uptime</th>
<th>Jobs</th>
<th>Last update (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>c3afdec0-5b06-41d9-bc51-e6b3b65ce5</td>
<td>192.168.61...</td>
<td>1.09</td>
<td>12:30:40</td>
<td>1</td>
<td>2015-08-26 06:53:52</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>87f5c840-87f5-4e4b-b6a1-87f5c84087f5</td>
<td>192.168.61...</td>
<td>1.04</td>
<td>00:30:06</td>
<td>5</td>
<td>2015-08-26 06:53:51</td>
</tr>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>74:55:11</td>
<td>8</td>
<td>2015-08-26 06:53:51</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>75:35:19</td>
<td>5</td>
<td>2015-08-26 06:53:50</td>
</tr>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.05</td>
<td>168:55:33</td>
<td>11</td>
<td>2015-08-26 06:53:49</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>75:25:27</td>
<td>5</td>
<td>2015-08-26 06:53:47</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.02</td>
<td>75:40:26</td>
<td>5</td>
<td>2015-08-26 06:53:44</td>
</tr>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>75:05:12</td>
<td>5</td>
<td>2015-08-26 06:53:41</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>455bc88d-47f5-48fd-b701-e78051496035</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>75:25:24</td>
<td>5</td>
<td>2015-08-26 06:53:40</td>
</tr>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>a1f32b54-326a-4734-8e08-43afbe37b5</td>
<td>192.168.61...</td>
<td>1.17</td>
<td>12:14:51</td>
<td>1</td>
<td>2015-08-26 06:53:39</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>a1f32b54-326a-4734-8e08-43afbe37b5</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>75:05:08</td>
<td>5</td>
<td>2015-08-26 06:53:39</td>
</tr>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>1c88a8b7-141d-4d79-b324-b9c3b036b254</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>75:55:31</td>
<td>6</td>
<td>2015-08-26 06:53:36</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>1c88a8b7-141d-4d79-b324-b9c3b036b254</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>76:45:07</td>
<td>6</td>
<td>2015-08-26 06:53:36</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>1c88a8b7-141d-4d79-b324-b9c3b036b254</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>74:15:54</td>
<td>8</td>
<td>2015-08-26 06:53:35</td>
</tr>
<tr>
<td>SL65-small-rocci-test</td>
<td>IHEP-OpenNebula</td>
<td>rocci-1.1-hep-o...</td>
<td>Running</td>
<td>1c88a8b7-141d-4d79-b324-b9c3b036b254</td>
<td>192.168.61...</td>
<td>1.13</td>
<td>73:50:57</td>
<td>4</td>
<td>2015-08-26 06:53:34</td>
</tr>
<tr>
<td>SL65-cvmfs-cloud</td>
<td>IHEP-OpenStack</td>
<td>nova-1.1-hep-o...</td>
<td>Running</td>
<td>dc07f7e1-8b23-43c1-b328-8debe50808ce5</td>
<td>192.168.61...</td>
<td>1.00</td>
<td>12:05:11</td>
<td>1</td>
<td>2015-08-26 06:53:34</td>
</tr>
</tbody>
</table>
VM monitor

- The total number of VMs by RunningPod
- The total jobs run in the Clouds
Accounting

- A history view of cloud as other resources
• Thank you!
“Image” section

- `bootImageName`
- `FlavorName`
- image name containing
  - OS, software.....
“Endpoint” Section

• Necessary info to connect with Cloud
• cloudDriver is the interface to connect cloud
• It is related directly with cloud name known by users

Code:

```yaml
<table>
<thead>
<tr>
<th>CloudEndpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-1.1-ihep-openstack</td>
</tr>
<tr>
<td>- cloudDriver = nova-1.1</td>
</tr>
<tr>
<td>- vmPolicy = elastic</td>
</tr>
<tr>
<td>- vmStopPolicy = never</td>
</tr>
<tr>
<td>- siteName = CLOUD.IHEP-OPENSTACK.cn</td>
</tr>
<tr>
<td>- osBaseURL = nouse</td>
</tr>
<tr>
<td>- ex_force_auth_url = <a href="http://202.122.35.254:5000/v2.0/tokens">http://202.122.35.254:5000/v2.0/tokens</a></td>
</tr>
<tr>
<td>- ex_force_auth_version = 2.0_password</td>
</tr>
<tr>
<td>- ex_tenant_name = dirac</td>
</tr>
<tr>
<td>- ex_force_service_region = regionOne</td>
</tr>
<tr>
<td>- maxEndpointInstances = 100</td>
</tr>
<tr>
<td>- cmfs_http_proxy = <a href="http://202.122.33.53:3128">http://202.122.33.53:3128</a></td>
</tr>
<tr>
<td>- auth = userpasswd</td>
</tr>
<tr>
<td>- ex_force_ca_cert = /etc/grid-security/certificates/IHEP-2013.pem</td>
</tr>
</tbody>
</table>
```
“Running Pod” section

• Requirements define the running env this RunningPods can provide
• Separate image and requirements? If image doesn’t match the requirements?

![Diagram of RunningPods with image and requirements details]
“SubmitPools”

- Define available resources to VM scheduler
- Different RunningPods are put into SubmitPools for VM scheduler