Cloud Bursting with SLURM and Bright Cluster Manager

Martijn de Vries
CTO
Architecture

Bright Cluster

CMDaemon

procedure call

SOAP+SSL

event

head node

node001

node002

node003

Cluster Management GUI

Cluster Management Shell

Web-Based User Portal

Third-Party Application

Bright Computing
Management Interfaces

Graphical User Interface (GUI)
- Offers administrator full cluster control
- Standalone desktop application
- Manages multiple clusters simultaneously
- Runs on Linux & Windows
- Built on top of Mozilla XUL engine

Cluster Management Shell (CMSH)
- All GUI functionality also available through Cluster Management Shell
- Interactive and scriptable in batch mode
Integration with workload manager:
- All popular workload managers supported
- SLURM default choice during installation
- Automatic installation

Points of integration:
- Automatic node and queue configuration
- Automatic high availability configuration
- Monitoring workload management metrics
- Health checking
- Job monitoring and control
A workload management system is highly recommended to run compute jobs. Please choose the workload management system that should be configured. To prevent a workload management system from being set up, select 'None'. The number of slots per node should ideally be equal to the number of CPU cores available on each node. On small clusters, the head node may also be used for compute jobs.

**Workload management system**  
Slurm(v2.2.4)

**Number of slots/node**  
8

**Use head node for compute jobs**  
Yes  No

---

The Simple Linux Utility for Resource Management (SLURM) is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. The slurm controller daemon will be configured to run on the head node and the slurm daemons will be configured to run on all the nodes. If the master node is required to run jobs, then the slurm will also run on the head node. MySQL will be used to store job accounting information.
Cloud Bursting

Scenario I

- Head node
- node001
- node002
- node003
Mixing Local and Cloud Resources

Cloud does not work well for all HPC workloads
- Sensitive data/computations
- Problems getting huge amounts of data in/out
- Workload may depend on low latency / high bandwidth
- Workload may depend on non-standard compute resources
- Workload may depend on advanced shared storage (e.g. Lustre)

Not everyone will replace HPC cluster with EC2 account
- Allow local cluster to be extended with cloud resources to give best of both worlds
- Allow workload suitable for cloud to be off-loaded
- Allow traditional HPC users to try out and migrate to cloud
Cloud Bursting

Scenario II

Head Node

node001

node002

node003

node004

node005

node006

node007
<table>
<thead>
<tr>
<th>Modified</th>
<th>Hostname</th>
<th>Type</th>
<th>Category</th>
<th>Persistent storage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vnode001</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode002</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode003</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode004</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode005</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode006</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode007</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode008</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode009</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode010</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode011</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode012</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode013</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode014</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode015</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td></td>
<td>vnode016</td>
<td>m1.xlarge</td>
<td>default</td>
<td>EBS (40G)</td>
</tr>
<tr>
<td>Modified</td>
<td>Job ID</td>
<td>Scheduler</td>
<td>User</td>
<td>Queue</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-----------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>modflow</td>
<td>slurm</td>
<td>kate</td>
<td>long.cloud</td>
</tr>
<tr>
<td></td>
<td>modflow</td>
<td>slurm</td>
<td>kate</td>
<td>long.cloud</td>
</tr>
<tr>
<td></td>
<td>modflow</td>
<td>slurm</td>
<td>kate</td>
<td>short.mixed</td>
</tr>
<tr>
<td></td>
<td>modflow</td>
<td>slurm</td>
<td>kate</td>
<td>short.mixed</td>
</tr>
<tr>
<td></td>
<td>mparallelapp</td>
<td>slurm</td>
<td>jodi</td>
<td>medium.local</td>
</tr>
<tr>
<td></td>
<td>serial1</td>
<td>slurm</td>
<td>alice</td>
<td>serial.cloud</td>
</tr>
<tr>
<td></td>
<td>serial1</td>
<td>slurm</td>
<td>alice</td>
<td>serial.cloud</td>
</tr>
<tr>
<td></td>
<td>serial1</td>
<td>slurm</td>
<td>alice</td>
<td>serial.mixed</td>
</tr>
<tr>
<td></td>
<td>serial1</td>
<td>slurm</td>
<td>alice</td>
<td>serial.mixed</td>
</tr>
<tr>
<td></td>
<td>xhpl</td>
<td>slurm</td>
<td>matthew</td>
<td>short.local</td>
</tr>
<tr>
<td></td>
<td>xhpl</td>
<td>slurm</td>
<td>matthew</td>
<td>short.local</td>
</tr>
<tr>
<td></td>
<td>xhpl</td>
<td>slurm</td>
<td>matthew</td>
<td>short.local</td>
</tr>
<tr>
<td></td>
<td>modflow</td>
<td>slurm</td>
<td>kate</td>
<td>long.cloud</td>
</tr>
<tr>
<td></td>
<td>modflow</td>
<td>slurm</td>
<td>kate</td>
<td>short.mixed</td>
</tr>
<tr>
<td></td>
<td>serial1</td>
<td>slurm</td>
<td>alice</td>
<td>serial.mixed</td>
</tr>
</tbody>
</table>
Cloud Network Map

- Cloud Network Map diagram with various network components and connections.
  - Internal network (internalnet) connected to head node and external network (externalnet).
  - Node001 to Node004 connected to internal network.
  - Head node connected to external network.
  - VPN X and VPN Y connected to various server nodes.
  - EC2 region X and EC2 region Y connected to cloud director and server nodes.
  - Node005 to Node008 connected to EC2 region X.
  - Node009 to Node012 connected to EC2 region Y.
Uniformity

Cloud nodes behave the same way as local nodes

- Same method of provisioning
- Same software image and user environment
- Same workload management set-up
- Same management interface that allows to control cluster
- Same monitoring & health checking

Everything can talk to everything

- Accomplished using VPN, routing, network mapping
- VPN set-up automated and does not require firewall set-up
  (requires just **outgoing** access on 1194/udp)
- Single global DNS namespace
Running Cloud Nodes

**Cloud Director has a number of responsibilities:**

- Gateway between local and cloud nodes
- Provision software image to cloud nodes
- Serve shared storage for cloud nodes
- Mirror network services for the cloud nodes (e.g. LDAP, DNS)

**Cloud node booting process**

- Instances are created with 1GB EBS and $n$GB ephemeral/EBS disk
- Bright Node Installer AMI goes on EBS disk
- Node Installer continues with normal procedure to bring up node
- Software image gets provisioned onto second disk
Common setup: one SLURM partition per cloud region

Example:
```
[root@sc11-demo ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
defq*        up   infinite      1   idle node001
california   up   infinite      4   idle cnode[001-004]
oregon       up   infinite      4   idle cnode[005-008]
```

Jobs that may run in the cloud should be submitted to one of the cloud partitions

SLURM will schedule jobs onto cloud nodes the same way as on local nodes

Current situation:
- /cm/shared mirrored and exported by cloud director
- /home mounted over VPN
- Works great, but /home is too slow
Data Locality Problem

- Jobs usually require input data and produce output data
- Input and/or output data may require significant transfer time
- Resources charged by the hour, so input/output data should be transferred while resources are not yet allocated
- Solution to data locality problem should ideally be hidden from users as much as possible
Data Aware Workload Management

- SLURM needs to be made aware of job data dependencies
- Jobs should not be scheduled until data is present on cloud-director
- As part of job script, copy input data in special input directory, copy output directory into output directory
- Workload management environment takes care of transferring input and output directories
- Option A) let SLURM take care of copying data (e.g. using job dependencies)
- Option B) transfer data using separate daemon and set SLURM job attributes to allow/disallow job start
Questions?