Bright Cluster Manager
Using Slurm for Data Aware Scheduling in the Cloud

Martijn de Vries
CTO
Bright Computing

1. Develops and supports Bright Cluster Manager for HPC systems, server farms, grids and clouds
2. Incorporated in USA and The Netherlands (Offices in San Jose and Amsterdam)
Bright Cluster Architecture

Cluster Management GUI

Cluster Management Shell

Web-Based User Portal

Third-Party Applications

CMDaemon

head node

node001

node002

node003

SOAP+SSL

SOAP+SSL

SOAP+SSL

Cluster Management GUI

Cluster Management Shell

Web-Based User Portal

Third-Party Applications

Bright Cluster

CMDaemon

head node

node001

node002

node003

SOAP+SSL

SOAP+SSL

SOAP+SSL
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
<table>
<thead>
<tr>
<th>Modified</th>
<th>Job ID</th>
<th>Queue</th>
<th>User</th>
<th>Queue</th>
<th>Status</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>763</td>
<td>slurm</td>
<td>marijn</td>
<td>cloudtransfers</td>
<td>RUNNING</td>
<td>demo</td>
</tr>
<tr>
<td></td>
<td>766</td>
<td>slurm</td>
<td>marijn</td>
<td>cloudtransfers</td>
<td>RUNNING</td>
<td>demo</td>
</tr>
<tr>
<td></td>
<td>767</td>
<td>slurm</td>
<td>marijn</td>
<td>cloudtransfers</td>
<td>RUNNING</td>
<td>demo</td>
</tr>
<tr>
<td></td>
<td>768</td>
<td>slurm</td>
<td>marijn</td>
<td>cloudtransfers</td>
<td>RUNNING</td>
<td>demo</td>
</tr>
<tr>
<td></td>
<td>769</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>770</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>771</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>772</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>773</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>774</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>775</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>776</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>777</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>778</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>779</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>780</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>781</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>782</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>783</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>784</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>785</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>786</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>787</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>788</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>789</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>790</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>791</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>792</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>793</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
<tr>
<td></td>
<td>794</td>
<td>slurm</td>
<td>defq</td>
<td>cloudtransfers</td>
<td>PENDING</td>
<td>(Dependency)</td>
</tr>
</tbody>
</table>

**EVENT VIEWER**

- **09/Oct/2012 14:01:08**: Bright 6.0 Demo Cluster, cnode001, Check DevicesUp is in state PASS on cnode001.
- **09/Oct/2012 14:01:08**: Bright 6.0 Demo Cluster, cnode003, Check DevicesUp is in state PASS on cnode003.
- **09/Oct/2012 13:58:14**: Bright 6.0 Demo Cluster, eu-west-1-director, Service named was restarted on eu-west-1-director.
- **09/Oct/2012 13:58:13**: Bright 6.0 Demo Cluster, demo, Service named was restarted on demo.
- **09/Oct/2012 13:58:00**: Bright 6.0 Demo Cluster, cnode002, Check DevicesUp is in state PASS on cnode002.
- **09/Oct/2012 13:58:00**: Bright 6.0 Demo Cluster, cnode004, Check DevicesUp is in state PASS on cnode004.
Integration with Slurm

- Slurm default choice for workload management system
- Slurm up and running at first boot
- Node & partition configuration
- Topology configuration
- HA configuration
- Workload management metrics
- Health checking
- Job monitoring and control
- Integrated in Cluster Management API
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.
Cluster On Demand Scenario

- 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
Cluster Extension Scenario

Cloud Bursting
Cloud Network Map

- internalnet
- head node
- externalnet
- VPN X
- EC2 region X
  - node001..node004
- cloud director
- node005..node008
- VPN Y
- EC2 region Y
  - node009..node012
Cloud Director acts as a head node in the cloud
- Provides gateway between local and cloud nodes
- Provisions software image to cloud nodes
- Serves shared storage for cloud nodes
- Mirrors network services for the cloud nodes (e.g. LDAP, DNS)

Cloud node booting process
- Instances are created with 1GB EBS and nGB 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
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
Workload Driven Cloud Resizing

- Nodes are created in the cloud:
  - Manually by administrator using CMGUI/CMSH
  - Automatically based on workload by cloud-resize utility
- Cloud-resize called periodically from crond
- Three inputs to cloud-resize:
  - Current workload
  - Current number of cloud nodes
  - Policy (Python module)
- When more cloud nodes are needed (as determined by policy), more nodes are created in the cloud based on configured node properties
- When more nodes come online (~2-5m), Slurm will schedule jobs onto nodes
Workload Management in the Cloud

- Typical setup: one workload management queue per region
- Jobs that may run in the cloud should be submitted to one of the cloud queues
- Alternatively, cloud nodes and regular nodes can be combined in same queue.
- Cloud nodes also have workload management features which can be used as job-constraints
- Example:
  ```bash
  #!/bin/sh
  #SBATCH -J TestJob
  #SBATCH --ntasks=16
  #SBATCH --constraint=us-east-1
  ```
- Workload management system will schedule jobs onto cloud nodes the same way as on local nodes
- Nodes NFS mount /home and /cm/shared:
  - Local nodes mount from head node
  - Cloud nodes mount from a cloud director
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
- Data moving mechanics should be hidden from users as much as possible

Solution:
- Bright introduces job submission utility *cmsub* which allows data dependencies of jobs to be made explicit in Slurm
- Useful for cloud, but can also be useful for e.g.
  - Fetching data from tape archive
  - Staging data to local compute nodes to overcome throughput limitations of parallel filesystem (needed for exascale)
Data-Aware Scheduling to the Cloud

1. User submits job to queue
2. Bright creates “data-transfer” job
3. Slurm runs compute job when data-transfer job is complete
4. Bright transfers output data back after completion
### Example

```bash
#!/bin/sh

#SBATCH -J Data-Transfer-Test
#SBATCH --ntasks=1

#CMSUB --input=/home/martijn/data-transfer-test/inputfile.txt
#CMSUB --regions=eu-west-1

# Do the heavy work of reversing the lines
tac inputfile.txt >outputfile-$SLURM_JOB_ID.txt

# Schedule output file to be transferred back
CM_SCHEDULE_TRANSFER(/home/martijn/data-transfer-test/outputfile-$SLURM_JOB_ID.txt)

echo Processed data on `hostname`
```
Data Aware Workload Management

- User submits job to workload management system using cmsub
- The cmsub utility will:
  - Submit input data transfer job to Slurm
  - Submit compute job with dependency on input transfer job
  - Submit output data transfer job with dependency on user job
- Data transfer jobs run on head node, so compute nodes need not be allocated while data is being transferred in/out of cloud
- Option to remove or keep data in the cloud after job completed
- Cmsub prevents multiple transfers of same data
- Partial data transfers are handled elegantly
- Users may also take responsibility for transferring data outside of cmsub
Future Directions

- Scheduling priorities of data transfers and compute jobs should be interdependent
- Order in which data should be transferred depends on:
  - Estimated transfer time (data size, target location)
  - Estimated job run time
  - Job priority
  - Resources requested by job
- Simple example:
  - Job 1: run time: 1h  input data: 10GB (10h)
  - Job 2: run time: 10h  input data: 1GB  (1h)
  - Naïve scheduling: 10h + 1h + 10h = 21h
  - Optimal scheduling: 1h + 10h + 1h = 12h
- Making things worse: what about priority for output data?
Questions?

Martijn de Vries
martijn.devries@brightcomputing.com