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

Martijn de Vries
CTO
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
Cluster Extension Scenario

Cloud Bursting
Cloud Network Map

- **internalnet**
- **head node**
- **externalnet**
- **VPN X**
- **VPN Y**
- **cloud director**
- **EC2 region X**
- **EC2 region Y**
- **node001-node004**
- **node005-node008**
- **node009-node012**
Data Locality Problem

**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)
Example

#!/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 cmsgub
- The cmsgub 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 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
- Cmsgub prevents multiple transfers of same data
- Partial data transfers are handled elegantly
- Users may also take responsibility for transferring data outside of cmsgub
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