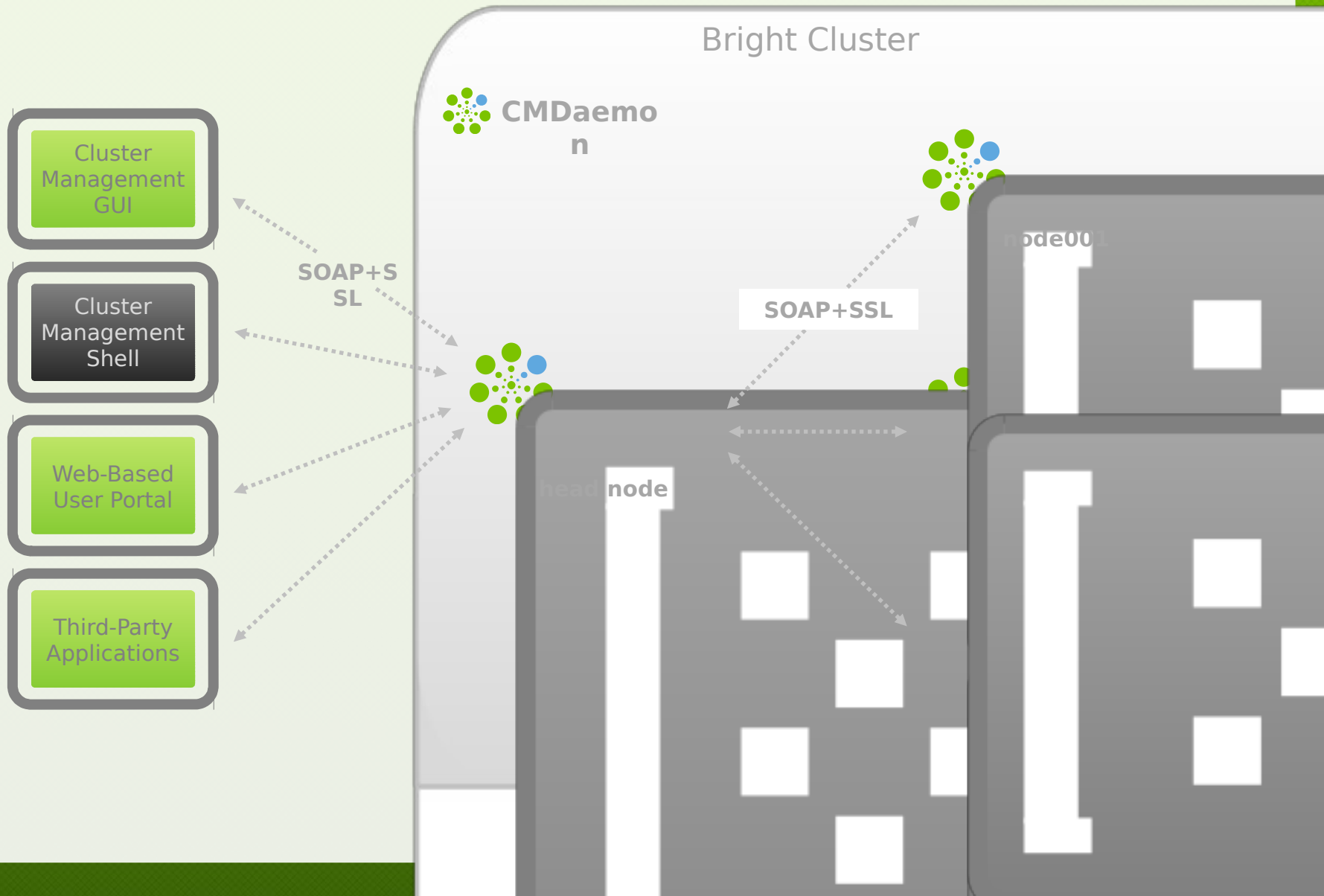
Bright Cluster Manager

Using Slurm for Data Aware Scheduling in the Cloud

Martijn de Vries
CTO

Architecture



Bright Cluster Manager

File Monitoring View Bookmarks Help

RESOURCES

- My Clusters
  - Bright 6.0 Demo Cluster
    - Switches
      - switch01
    - Networks
      - amazon
      - eu-west-1
      - externalnet
      - globalnet
      - internalnet
      - netmap
    - Power Distribution Units
      - apc01
    - Software Images
      - default-image
    - Node Categories
      - cloud-director
      - default
    - Head Nodes
      - demo
    - Racks
      - 1
        - apc01
        - switch01
    - Chassis
    - Virtual SMP Nodes
    - Nodes
      - node001
      - node002
    - Cloud Nodes
      - Amazon EC2
        - cnode001
        - cnode002
        - cnode003
        - cnode004
        - cnode005
        - cnode006

Bright 6.0 Demo Cluster

Overview | Settings | Failover | Rackview | Health | Parallel shell | License | Notes

Uptime: 0 days 8 hours 50 minutes

Nodes: 8 ↑ 4 ↓ 0 ⊖

GPU Units: 0 ↑ 0 ↓ 0 ⊖

Devices: 1 ↑ 1 ↓ 0 ⊖

Jobs: 0 running 0 waiting

Phase load: 1.4 A

CPU Cores: 17 out of 17

GPUs: 0 out of 0

Memory: 2.08 GiB out of 32.91 GiB

Users: 0 out of 2

CPU Usage: 0.28% u 0.23% s 0.08% o 99.41% i

Occupation rate: 0%

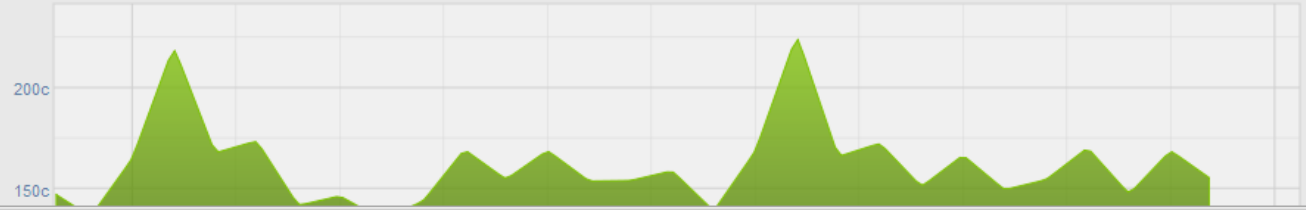
Disk Usage

Mountpoint	Used	Size	Use %
/	18.64 GiB	103.57 GiB	
/boot	109.57 MiB	472.54 MiB	
/tmp	537.48 MiB	7.51 GiB	
/var	1.49 GiB	15.02 GiB	
/var/lib/mysql/cmdaemon_mon	702.14 MiB	9.39 GiB	

Workload Management

Queue	Running	Queued	Error	Completed	Avg. Duration	Est. delay
defq	0	0	2	248	1 seconds	0 seconds
cloudtransfers	0	0	0	488	7 seconds	0 seconds

Metric: CtxtSwitches (ctx_switch/s)



EVENT VIEWER

All Events

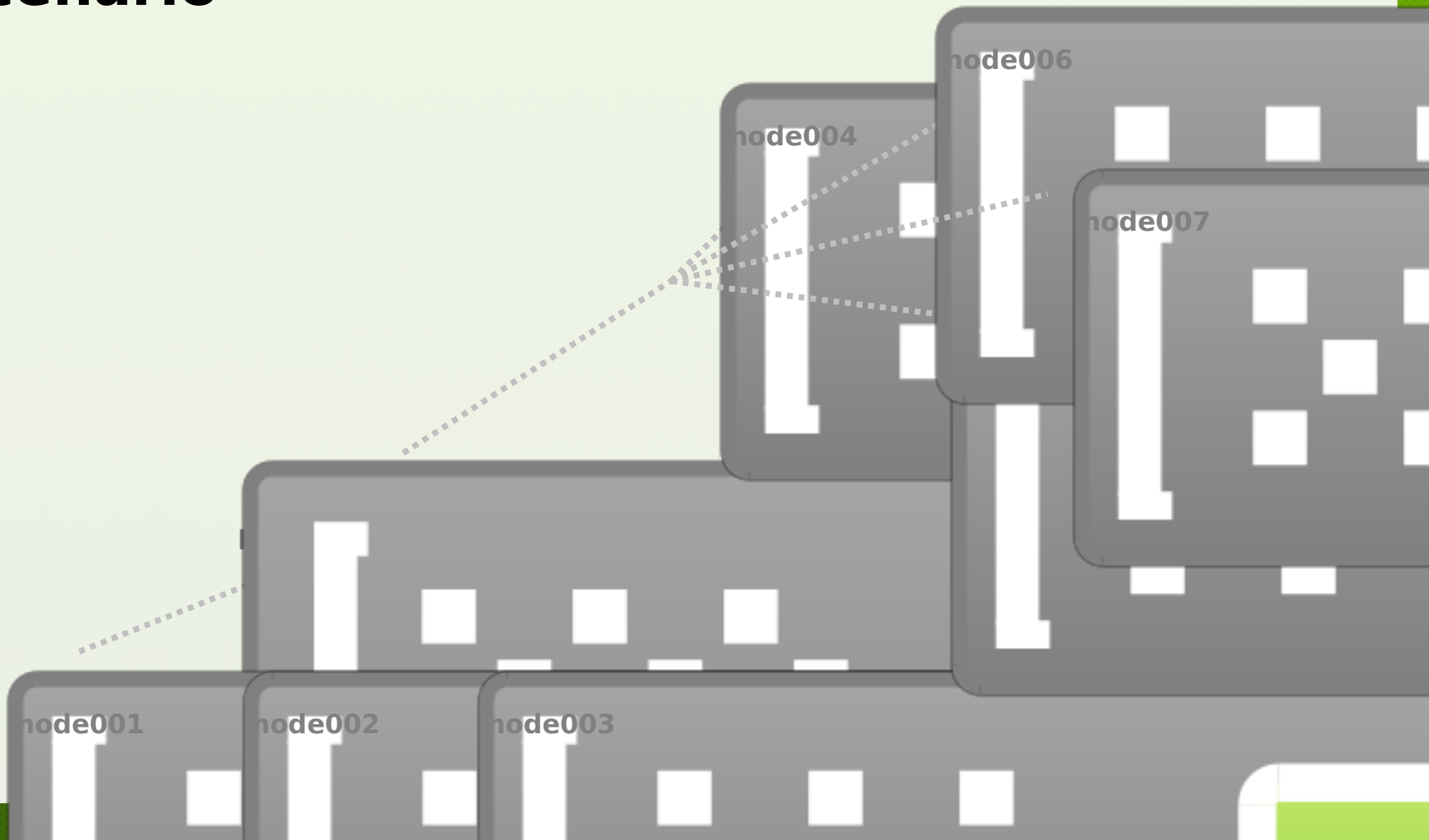
Time	Cluster	Source	Message
09/Oct/2012 14:01:08	Bright 6.0 Demo Cluster	cnode001	Check 'DeviceIsUp' is in state PASS on cnode001
09/Oct/2012 14:01:08	Bright 6.0 Demo Cluster	cnode003	Check 'DeviceIsUp' is in state PASS on cnode003
09/Oct/2012 13:58:14	Bright 6.0 Demo Cluster	eu-west-1-director	Service 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 13:58:06	Bright 6.0 Demo Cluster	cnode002	Check 'DeviceIsUp' is in state PASS on cnode002
09/Oct/2012 13:58:00	Bright 6.0 Demo Cluster	cnode004	Check 'DeviceIsUp' is in state PASS on cnode004

Ready

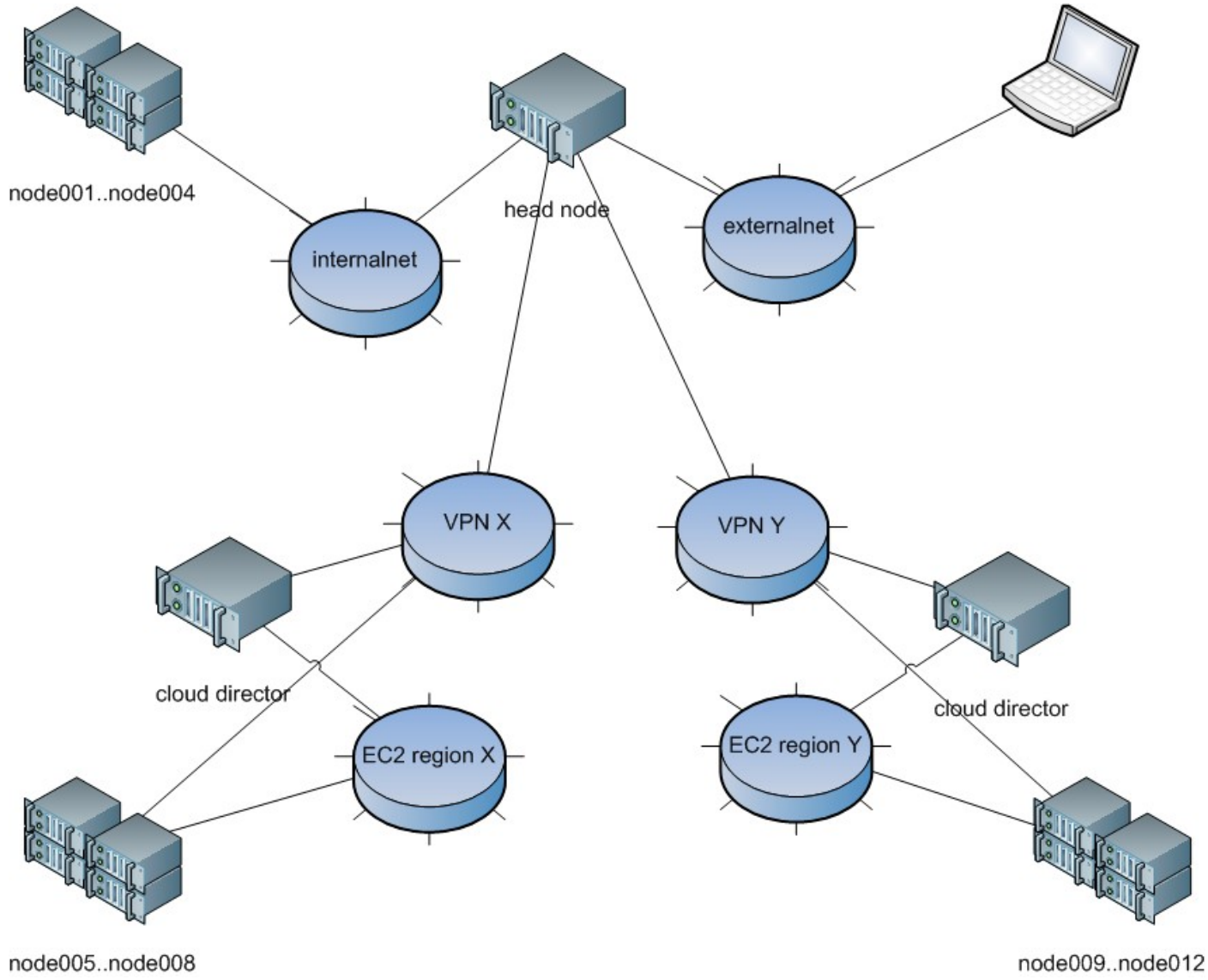
Integration with Slurm

- 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 Network Map



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)

Slurm Job with Data Dependencies

- 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 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 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