Bright Cluster Manager
Advanced system management & monitoring made easy
... on Cray Systems

Dr Matthijs van Leeuwen
CEO

Martijn de Vries
CTO
The Commonly Used “Toolkit” Approach

- Most HPC cluster management solutions use the “toolkit” approach (Linux distro + tools)
  - Examples: Rocks, PCM, OSCAR, UniCluster, CMU, etc.
  - Tools typically used: Ganglia, Cacti, Nagios, Cfengine, System Imager, xCAT, Puppet, Cobbler, Hobbit, Big Brother, Zabbix, Groundwork, etc.

- Issues with the “toolkit” approach:
  - Tools rarely designed to work together
  - Each tool has its own command line interface and GUI
  - Each tool has its own daemon and database
  - Tools rarely designed to scale
  - Tools rarely designed for HPC

- Making a collection of unrelated tools work together
  - Requires a lot of expertise and scripting
  - Rarely leads to a really easy-to-use and scalable solution
About Bright Cluster Manager

- Bright Cluster Manager takes a much more fundamental & integrated approach
  - Designed and written from the ground up
  - Single cluster management daemon provides all functionality
  - Single, central database for configuration and monitoring data
  - Single CLI and GUI for ALL cluster management functionality

- Which makes Bright Cluster Manager ...
  - Extremely easy to use
  - Extremely scalable
  - Secure & reliable
  - Complete
  - Flexible
  - Maintainable
Architecture

Bright Cluster

- CMDaemon
- head node
- node001
- node002
- node003

Cluster Management GUI

Cluster Management Shell

Web-Based User Portal

Third-Party Application

procedure call

SOAP+SSL

event

procedure call

SOAP+SSL

event
Bright Cluster Manager – Elements

Cluster Management GUI

User Portal

Cluster Management Shell

SSL / SOAP / X509 / IPtables

Cluster Management Daemon

Provisioning

PBS Pro
Torque
Maui/MOAB
Grid Engine
SLURM
LSF

Monitoring
Automation
Health Checks
Management

Compilers
Libraries
Debuggers
Profilers

SLES / RHEL / CentOS / SL

CPU

GPU

Memory

Dis

Eth

Int

iLO

IPM

PDU
**MESSAGE OF THE DAY**

This is the message of the day. Feel free to edit this to your liking (in \form{https://www.brightcomputing.com/tdm/index.php}).

On the right, you will see download and contact information. If there is no contact information available, you can set it in CMGUI/CMSH. Alternatively, you can modify /var/www/html/contact.php.

**DOCUMENTATION**

- Bright Computing website
- Administrator manual
- User manual

**CONTACT**

James Smith  
System Administrator  
Tel: (408) 399-1922  
james.smith@uni.edu

---

**CLUSTER OVERVIEW**

<table>
<thead>
<tr>
<th>Uptime</th>
<th>9 days 8 hours 31 min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1.2 GiB out of 8.3 GiB total</td>
</tr>
<tr>
<td>Nodes</td>
<td>2 ↑ 6 ↓ 1 Θ</td>
</tr>
<tr>
<td>Devices</td>
<td>0 ↑ 1 ↓ 0 Θ</td>
</tr>
<tr>
<td>Cores</td>
<td>3 ↑ 3 total</td>
</tr>
<tr>
<td>Users</td>
<td>0 out of 2 total</td>
</tr>
<tr>
<td>Phase Load</td>
<td>N/A ampere</td>
</tr>
<tr>
<td>Occupation Rate</td>
<td>3.3%</td>
</tr>
</tbody>
</table>

**WORKLOAD OVERVIEW**

<table>
<thead>
<tr>
<th>Queue</th>
<th>Scheduler</th>
<th>#Slots</th>
<th>#Nodes</th>
<th>#Running</th>
<th>#Queued</th>
<th>#Failed</th>
<th>#Completed</th>
<th>Avg. Duration</th>
<th>Est. Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>short.q</td>
<td>Slurm</td>
<td>0</td>
<td>258</td>
<td>32</td>
<td>43</td>
<td>0</td>
<td>482</td>
<td>00:07:27</td>
<td>00:09:05</td>
</tr>
<tr>
<td>medium.q</td>
<td>Slurm</td>
<td>0</td>
<td>123</td>
<td>5</td>
<td>11</td>
<td>0</td>
<td>41</td>
<td>02:15:00</td>
<td>04:16:00</td>
</tr>
<tr>
<td>long.q</td>
<td>Slurm</td>
<td>0</td>
<td>123</td>
<td>8</td>
<td>13</td>
<td>0</td>
<td>01</td>
<td>08:00:00</td>
<td>15:13:00</td>
</tr>
</tbody>
</table>
Management Interface

**Graphical User Interface (GUI)**
- Offers administrator full cluster control
- Standalone desktop application
- Manages multiple clusters simultaneously
- Runs natively on Linux & Windows

**Cluster Management Shell (CMSH)**
- All GUI functionality also available through Cluster Management Shell
- Interactive and scriptable in batch mode
Architecture – Monitoring

- Cluster Management GUI
- Web-Based User Portal

Bright Cluster

CMDaemon

- head node
- node001
- node002
- node003

- metrics
- monitoring data
- events

- BMC

- raw data
- consolidated data

- BMC

- BMC

- BMC
Scenario I - “Cluster on Demand”
Scenario II - “Cluster Extension”
Cray & Bright Cluster Manager
Bright Cluster Manager default for Cray External Service nodes since 2010:
  • esMS
  • esLogin
  • esAUX
  • esDM
  • Lustre MDS
  • Lustre OSS

Bright Cluster Manager considered for integration with Sonexion

Bright Cluster Manager considered for Cray mainframe
Cray/Bright Customers
1. esMS servers are Bright head nodes in failover mode
2. All other servers are Bright slave nodes
3. Bright does provisioning, monitoring, alerting, automation, health checking, access control, Lustre failover, etc.
Bright on Cray XE6 / XK6
Project goal:
Investigate whether Bright Cluster Manager can be used to manage and monitor a Cray XE6

Effort made:
Worked almost 2 weeks with 2 developers

Result:
Bright 6.0b successfully manages XE6 making use of Cray Linux Environment infrastructure
CMDaemon

Bright head node

provisioning node, router between HSN and SMW
Booting Nodes

- Default Cray kernel is used
- Bright Node Installer called from code in cpio boot image
- Node Installer:
  - Starts when node boots
  - Determines node identity based on Gemini MAC (could be NID)
  - Provisions software image into tmpfs filesystem
- Default software image about 2.7GB (can be reduced heavily)
- Parts of software image can be imported over NFS/DVS (minimal setup requires ~80MB)
- Root over NFS/DVS (now) also possible
Points of Integration

- **Power management:**
  - allow components (e.g. nodes) to be reset
  - allow entire system to be reset powered on/off
  - using xtbootsys (probably need lower level utilities in future)

- **Remote console:**
  - allow console of nodes to be accessed
  - using xtcon

- **Monitoring:**
  - allow Cray hardware metrics to be monitored
  - using SEDC (would be good to get direct access to HSS instead)

- **Health checking (not done yet):**
  - Cray hardware health checks in Bright health checking framework
  - using xthealth
Why run Bright on Cray?

- Less steep learning curve for Administrators
- Single interface for managing mainframe and external service nodes (e.g. login, storage)
- Single head node (SMW) which manages everything (HA possible)
- Consistent software image on login and compute nodes
- Same solution across the data-centre which makes integration of Cray system easier
- Access to cutting edge features (e.g. cloud bursting, monitoring, health checking, GPU management, role based access control)
Bright Scalability

- Ability to scale cluster usually limited by head node providing vital services
- Bright philosophy: allow all services provided by head node to be off-loaded to multiple dedicated nodes
- Allow (re-)configuration on the fly by assigning roles to nodes
- Example: node can be turned into provisioning node by assigning it the Provisioning role
- Goal: Linear scaling in terms of node-count
- In large clusters head node is not responsible for anything
Compute Node Footprint

- CMdaemon resident memory size: 31MB
- 7.5 CPU core-seconds per day
- On 16 core node, less than 0.5s wall-clock time per day
- Just 15m wall-clock time lost over 5 years
- Metrics are sampled out-of-band where possible (e.g. through SEDC)
- Other metrics are sampled from within CMdaemon process (i.e. no fork())
- Monitoring configuration highly tunable
- Metric sampling synchronized as much as possible
- No measurable OS jitter at small scale, large scale remains to be tested
Work Remaining

- Create clean installation procedure which integrates nicely into Cray installation procedure
- Migrate some services (e.g. named, LDAP) from SMW to boot node
- Support Cray component hierarchy natively:
  - Cabinet -> Cage -> Slot -> Node (Cray)
  - Rack -> Chassis -> ??? -> Node (Bright)
- Integrate Cray health checks into Bright health checking framework
- Let CLE tools such as “xtcli status” recognize nodes running Bright (currently reports nodes as down)
- Rack view which resembles physical layout of Cray system
Work Remaining

- Allow nodes to be easily switched between classic Cray mode and Bright mode
- Extend range syntax in CMSH to support Cray-style hostnames
- Tighter integration with CLE (e.g. directly calling HSS)
- Integrate with Cray user environment (compilers, libraries, MPI)
- Improve integration of power management for individual components
- Test everything at scale
Questions?