MySched

Andreas Gellrich

Scientific Computing Group Meeting
12 July 2012, DESY
DESY Grid Center: Grid + NAF

> Grid computing started at DESY in 2004:
  - DESY is the *home* of 10 VOs (site: DESY-HH)
  - (WLCG-)Tier-2 for ATLAS, CMS, and LHCb in Germany (Tier-1: GridKa)
  - HERMES / H1 / ZEUS; ILC / CALICE;
  - BELLE2; IceCube; BIOMED / W-ENMR

> One *complete* *generic* Grid infrastructure for *all* VOs
  - *Federated* resources *w/ opportunistic* usage (“*everybody profits*”)
  - Flexible and scalable to support new VOs
  - Roughly 2/3 of the resources are currently used by the Tier-2 VOs

> Grid is *complemented* by the National Analysis Facility (NAF) [size: ~1 Tier-2]
The Grid infrastructure is the largest Linux installation at DESY

Grid services: (Core servers)
- Servers: ~50
- OS: SL 5.5/64-bit (x86_64)

Computing Resources: (Computing Elements) (CE)
- Compute nodes: 370 hosts, 808 procs, 3504 cores, 4784 job slots
  2GB RAM/slot, 15GB scratch space/slot
- Processing power: ~38 kHEPSPEC
- OS: SL 5.5/64-bit (x86_64)

(Disk) Storage Resources: (Storage Elements) (SE)
- Total: 4300 TB
Job classification

- Computing requests in HEP are parallelized on job-level - not in the application
- HEP jobs are independent and self-contained and can be treated individually

wrsp. to resource requirements jobs can be classified:

- Monte Carlo jobs are CPU-dominated
- Analysis jobs are I/O-dominated and massively use the local disk

- Jobs contain the submitter's credentials as X509 VOMS-proxies
- VOMS-proxies are mapped to UID/GID on the Computing Elements (CE)

- The accounts are the key to distinguish and handle job classes
User credential mapping

- **Job**
  - JDL
  - CE (CREAM)
  - X509 VOMS
  - lcmaps

- **Batch System**
  - queue limits
  - limits & algorithms
  - Queue (torque)
  - Scheduler (...)

- **WN**
  - qrun
  - host uid gid

**Accounting data**

**User credential mapping**

- cms:/cms
- cms:/cms/Role=lcgadmin
- cms:/cms/Role=production
- cms:/cms/de
- cms:/cms/Role=pilot

→ cmsusr175
→ cmssgm007
→ cmsprd018
→ cmsger234
→ cmsplt079
Computing resources

> At Grid sites computing resources are deployed in *batch farms*

> The farm node are called *Worker Nodes* (WN)

> From the batch system point of view the Grid is a simple *'qsub'*

> WNs may be real or *virtual machines* (VM)

> WNs provide CPU-cores, memory, disk scratch space, a network link

> For the batch system each WN provides job slots; usually 1 slot per core

> DESY-HH WNs are heterogeneous reaching from 1 – 48 cores per WN with per slot >=2GB RAM, 20GB scratch space, and 1 GB per WN

> single resources per WN must not be exhausted
Computing resources cont'd

2 GB mem / slot

WN

8 cores

20 GB scratch / job

1 GE / 8 jobs

Andreas Gellrich | MySched | 12 July 2012 | Page 7
48-core WN

- **Load Processes**
  - ![Load Process Graph](Image)
  - *Load/Proc*

- **Memory Usage**
  - ![Memory Usage Graph](Image)
  - *Bytes*
  - *Memory Used, Memory Shared, Memory Cached, Memory Buffered, Total In-Core Memory*

- **CPU Last Week**
  - ![CPU Usage Graph](Image)
  - *Percent*
  - *User CPU, Nice CPU, System CPU, WAIT CPU, Idle CPU*

- **Network Last Week**
  - ![Network Usage Graph](Image)
  - *Bytes/sec*
  - *In, Out*

- **cpu_wio**
  - ![IO Graph](Image)
  - *0 - 300 m*
  - *grid-wn0001.desy.de last week (now 0.00)*
Queuing and scheduling

- Each batch system consists of a queuing system and a scheduler

- Jobs are enqueued by the CEs according to requirements (queue)
- Limits and rules may be individually applied to queues

- The scheduler picks jobs of the queues and starts them on the WNs
- The scheduler bases the job distribution on configurable rules and limits
- The scheduler is the key to optimal utilization of computing resources

- gLite/EMI supports among others PBS/torque and maui
- DESY-HH: torque-2.5.7-7 / maui-3.2.6-p21
Queuing

- The DESY-HH batch system deploys a separate queue for each SE
Scheduling

➢ At DESY-HH as on many other sites maui was deployed as a scheduler

“The Maui Scheduler is a policy engine which allows sites control over when, where, and how resources such as processors, memory, and disk are allocated to jobs. In addition to this control, it also provides mechanisms which help to intelligently optimize the use of these resources, monitor system performance, help diagnose problems, and generally manage the system.”

➢ The scheduler maui has more features than currently needed in HEP

➢ Many sites have problem with the configuration …

➢ DESY-HH as well:
  ▪ Instabilities
  ▪ Blocking of submissions
  ▪ Low occupancy
  ▪ Configuration questions
A new scheduler (*MySched*)

- Occupancy with maui and afterwards:

![Graph showing occupancy with maui and afterwards](image_url)

**Running**

- Y-axis: Number of running jobs (NRunning)
- X-axis: Time (Jul to Jun)
- Legend: Different colors represent different projects (e.g., atlas, belle, biomed, calice, etc.)

**Queueing**

- Y-axis: Number of queued jobs (NQueued)
- X-axis: Time (Jul to Jun)
- Legend: Same as above

Andreas Gellrich | *MySched* | 12 July 2012 | Page 12
Requirements to *MySched*

- Scalable (number of jobs; number of slots)
- Optimizing resource utilization (distribution of jobs to WNs)
- Light-weighted (CPU and memory usage)
- Configurable (config file)
- Tailored to HEP (job-parallel)
- Monitoring
- Limited development/maintenance time
Implementation

- To be based on the torque C-API (libtorque.so.2)
- Run in one process from a script as a cron
- Text logfile and monitoring info to a csv-file

- Base scheduling on a list of jobs and a list of nodes (WN)

- Configuration file with rules and limits

- Simple and obvious job distribution algorithm

- Monitoring via web
Limits and rules

➢ General:
  - `jobMaxTotal` (max number of jobs to be considered) [30000]
  - `jobMaxSubmit` (max number of jobs to be submitted) [1000]
  - `maxNodeSub` (max number of jobs to be submitted per node) [2]
  - `musecSleep` (sleep after submission in musec) [200000]
  - `ndays` (usage statistics of last n*24h) [1]
  - `maxDiversity` (submit to node with max user diversity) [on]

➢ By-pass ('hot'):
  - `hotVo` (hot VO) ["ops"]
  - `hotGroup` (hot group) ["desytst"]
  - `hotRole` (hot role) ["tst"]
  - `hotUser` (hot user) ["cmsusr165"]
  - `hotQueue` (hot queue) ["emi"]
Limits and rules cont’d

- **VO/group**
  - enable/disable  (whole VO or group)
  - max  (absolute number of jobs)
  - fraction  (maximal fraction of number of jobs of online slots)
  - nodemax  (maximal number of jobs per node=WN)
  - nodefrac  (maximal fraction of number of jobs per node)
  - Type  (meet type of node)
  - Share  (relative usage with past time interval)

- **Node**
  - enable/disable  (node)
  - type  (node type)
  - queues  (allow list of queues)
Algorithm

- Create node list
- Create job list
- Sort job list according to share \(\text{(order jobs)}\)
  - Treat job in list
  - Check each job for limits and rules \(\text{(check limits)}\)
  - Find appropriate WN \(\text{(find Node)}\)
  - Start job on WN \(\text{(submit job)}\)
  - Update node list

Find node:
- Online, not-busy, free slots, vo/group/user-on-node limits
- Take node with least occupancy or max diversity
Algorithm cont'd

<table>
<thead>
<tr>
<th>Group</th>
<th>Q&amp;W shar/%</th>
<th>R</th>
<th>max</th>
<th>diff/%</th>
<th>used/%</th>
<th>diff/%</th>
</tr>
</thead>
<tbody>
<tr>
<td>bellesgm</td>
<td>6</td>
<td>0</td>
<td>97</td>
<td>-100</td>
<td>0</td>
<td>-100</td>
</tr>
<tr>
<td>desysgm</td>
<td>1</td>
<td>0</td>
<td>29</td>
<td>-100</td>
<td>0</td>
<td>-100</td>
</tr>
<tr>
<td>opusur</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>-100</td>
<td>0</td>
<td>-79</td>
</tr>
<tr>
<td>biomedusr</td>
<td>29</td>
<td>0</td>
<td>26</td>
<td>48</td>
<td>1</td>
<td>-36</td>
</tr>
<tr>
<td>atlasplt</td>
<td>5</td>
<td>10</td>
<td>347</td>
<td>504</td>
<td>9</td>
<td>-16</td>
</tr>
<tr>
<td>atlastger</td>
<td>3</td>
<td>0</td>
<td>18</td>
<td>13</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>zeususr</td>
<td>673</td>
<td>0</td>
<td>400</td>
<td>96</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>cmsprd</td>
<td>164</td>
<td>0</td>
<td>1306</td>
<td>850</td>
<td>23</td>
<td>28</td>
</tr>
<tr>
<td>cmsusr</td>
<td>2812</td>
<td>0</td>
<td>1046</td>
<td>189</td>
<td>453</td>
<td>6</td>
</tr>
<tr>
<td>hermesusr</td>
<td>198</td>
<td>0</td>
<td>490</td>
<td>96</td>
<td>410</td>
<td>3</td>
</tr>
<tr>
<td>atlaspred</td>
<td>486</td>
<td>0</td>
<td>749</td>
<td>730</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>iceprd</td>
<td>172</td>
<td>0</td>
<td>127</td>
<td>39</td>
<td>226</td>
<td>1</td>
</tr>
<tr>
<td>TOTAL</td>
<td>4550</td>
<td>56</td>
<td>4799</td>
<td>68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Andreas Gellrich | MySched | 12 July 2012 | Page 18
Demo

> root@grid-batch5: [~] less mysched.conf

> root@grid-batch5: [~] less mysched.log

> http://grid-mon1.desy.de/mysched.html
Outlook

➢ It's still a study not a product!

➢ It is hard to judge performance in details.

➢ Looking at inclusive data:
  ▪ Occupancy plots
  ▪ Timing
  ▪ Resource utilization

➢ More analysis needed ...

➢ Multi-core jobs are not treated ...
Monitoring: http://grid.desy.de/mysched.html

Statistics for 1 week:

- Job submissions (success, failure)
- Job submission successes per VO
- Scheduler timing
- Jobs
Monitoring:  http://grid.desy.de/mysched.html

Jobs running/queuing in 24 hours:
Monitoring: [http://grid.desy.de/mysched.html](http://grid.desy.de/mysched.html)

LHC jobs running/queuing per group in 24 hours:
Monitoring: http://grid.desy.de/mysched.html

Non-LHC jobs running/queuing in 24 hours:
Monitoring:  http://grid.desy.de/mysched.html

Jobs running/queuing in 1 week:
Monitoring:  http://grid.desy.de/mysched.html

Jobs ended in 1 week: