Unified Workload Management for the Cray XC System with Univa Grid Engine

Daniel Gruber
Senior Solution Architect

Mai, 2016
Who is Univa?

Univa is a leading developer of Workload Optimization solutions

- Global reach – based in Chicago with offices in Markham, Canada, Munich and Regensburg, Germany
- Fast growing enterprise software company
- “Home of Grid Engine.” All Grid Engine developers work at Univa, including founder of the Grid Engine project
- Support global Fortune 500 companies
Customer Use-Cases

The most innovative companies are optimized on our platform:

<table>
<thead>
<tr>
<th>BIG DATA</th>
<th>ENTERPRISE TECHNICAL</th>
<th>ENTERPRISE APPLICATION</th>
<th>ISV/HVS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIMEDES Quantifying Healthcare</td>
<td>bp</td>
<td>sears</td>
<td>SYNOPSYS</td>
</tr>
<tr>
<td>CenturyLink™</td>
<td>NOVARTIS</td>
<td>Danske Bank</td>
<td>Mentor Graphics</td>
</tr>
<tr>
<td>TransUnion.</td>
<td>Saudi Aramco</td>
<td>NUANCE</td>
<td>ANSYS</td>
</tr>
<tr>
<td>epsilon</td>
<td>SAMSUNG</td>
<td>DB</td>
<td>JEPPESSEN</td>
</tr>
<tr>
<td>VERTEX</td>
<td>MARVELL®</td>
<td>BMO</td>
<td>Microsoft</td>
</tr>
<tr>
<td>CSH Cold Spring Harbor Laboratory</td>
<td>BMW</td>
<td>Nielsen</td>
<td></td>
</tr>
</tbody>
</table>
Univa Grid Engine
20 Years of History

• 1992: Initial developments @ Genias in Regensburg
• 1993: First Customer Shipment (as CODINE)
• 1996: Addition of GRD policy module
  • Collaboration with Raytheon & Instrumental Inc
• 1999: Merger with Chord Systems into Gridware
• 2000: Acquisition through Sun
  • Re-launch as Sun Grid Engine
• 2001: Open Sourcing
• Until 2010: Massive growth in adoption (>10,000 sites)
• 2010: Acquisition through Oracle
  • Open Source gets orphaned
• 2011: Key engineering team joins Univa
• 2013: Acquired all IP and assets from Oracle
• Soon: 5th major release of Univa Grid Engine
### Major 8.1 Features & Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Throughput</th>
<th>Utilization</th>
<th>Effort</th>
<th>Cost</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Classes</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td><strong>Templates</strong> providing ease of use, control and best fit</td>
</tr>
<tr>
<td>Resource Maps</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Optimal mapping of resources to jobs</td>
</tr>
<tr>
<td>Core / NUMA Binding</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Tuned utilization of CPU and memory architectures / <strong>per NUMA node</strong> memory reporting and accounting</td>
</tr>
<tr>
<td>Database Spooling</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Increased uptime @ top performance</td>
</tr>
<tr>
<td>Fair Urgency</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Balanced utilization of critical resources</td>
</tr>
<tr>
<td>MPI Integrations</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>Tuned, out-of-the-box integration with MPI versions</td>
</tr>
<tr>
<td>Improved Diagnostics</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Faster time to resolution of issues</td>
</tr>
</tbody>
</table>
## Major 8.2 Features & Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Throughput</th>
<th>Utilization</th>
<th>Effort</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cgroups integration</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Native Windows Support</td>
<td></td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRMAA2 API</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New read-only thread pool</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**How**

- Better workload isolation and resource limitation on Linux
- Supports Windows as submit and execution host without requiring UNIX emulation layer
- Implements open, standardize API for job monitoring, job workflow management, and job submission
- Highly improved scalability due to new, separate components in qmaster
- Dozens of smaller improvements based on direct feedback of our customers reducing management overhead and simplifies product usage
## Major 8.3 Features & Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Throughput</th>
<th>Utilization</th>
<th>Effort</th>
<th>Cost</th>
<th>How</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Jobs Add-on</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Capability of running <strong>100s of jobs per second</strong> through high performance message bus</td>
</tr>
<tr>
<td>Real Preemption</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td>Freeing Grid Engine resources when job is preempted</td>
</tr>
<tr>
<td>Web Services API</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>Access Univa Grid Engine through <strong>REST Style API</strong> (status / configuration)</td>
</tr>
<tr>
<td>Universal Resource Broker Add-on</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td>Running <strong>Spark</strong> or other Mesos frameworks on top of Univa Grid Engine</td>
</tr>
</tbody>
</table>
Special Support for Cray XC Systems
Customer Quote

• “Grid Engine has enabled us to integrate both Cray and non Cray jobs into the same scheduler. It allows us visibility of all our jobs in one place.”

• “It allows us to use the advanced scheduling capabilities of GE to further enhance the Cray internal scheduler. GE effectively acts as a meta-scheduler for the Cray.”

• “The GE environment has proven to be very stable environment scheduling on top of the Cray system.”

Chris Semple, PGS
One Single Shared Resource Pool

- Univa Grid Engine supports heterogeneous clusters
- Global policies and rules for jobs / job classes, users, projects, departments, resources
- Single interfaces for users and admins including new REST API
- Fast, scalable, reliable
Workload @ Cray XC using ALPS / BASIL

- Job Types:
  - Login node jobs (some use it for compilation)
  - Cray XC jobs (using aprun): **Batch** and **Interactive, Array Jobs**

- Cray XC node requests:
  - `-q cray.q` for routing to the Cray machine
  - `-pe cray <nodes>` for selecting the amount of nodes
  - Optional: `mppwidth mppdepth mppnppn cray_nodes`

- Cray XC Job:
  - Runs by using automatically reserved resources via `aprun`
  - Can use subset of **reserved resources** / can use `resources multiple times` sequentially
Workload @ Cray XC

- RUR Integration:
  - Mapping of Cray measurements into Univa Grid Engine counterparts:
    - utime → ru_utime
    - stime → ru_stime
    - max_rss → ru_maxrss
    - wchar → ru_oublock
    - rchar → ru_inblock
  - Get aggregated usage statistics with `qacct`

<table>
<thead>
<tr>
<th>Complex</th>
<th>Semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>cray_nodes</td>
<td>Global consumable resource for compute node limitation</td>
</tr>
<tr>
<td>cray_alps_version</td>
<td>Displays ALPS version detected at installation time</td>
</tr>
<tr>
<td>cray_basil_version</td>
<td>Displays BASIL version detected at installation time</td>
</tr>
<tr>
<td>cray_cores_per_host</td>
<td>Displays the amount of cores on a Cray compute node</td>
</tr>
<tr>
<td>cray_forced</td>
<td>This complex protects cray.q so that no other job runs in</td>
</tr>
<tr>
<td>cray_numa_nodes</td>
<td>Amount of NUMA nodes on a Cray compute node</td>
</tr>
<tr>
<td>mppwidth</td>
<td>Cray reservation request: Amount of Cray PEs to start up</td>
</tr>
<tr>
<td>mppdepth</td>
<td>Cray reservation request: Distance between two Cray PEs</td>
</tr>
<tr>
<td>mppnpppn</td>
<td>Cray reservation request: Amount of Cray PEs on a node</td>
</tr>
</tbody>
</table>
Simple Job Submission

> qsub -q cray.q -pe cray 3 /home/crayadm/job_script.sh
Your job 279 ("job_script.sh") has been submitted

> qstat -j 279
...
hard resource_list:      cray_forced=1,cray_nodes=1,mppwidth=3,mppdepth=32,mppnppn=0

> apstat -a
Total placed applications: 1
Apid  ResId   User  PEs  Nodes  Age  State  Command
3824  4262  crayadm  3   3  0h00m  run worker.sh
Automatic Alignment of Requests

> qsub -q cray.q -pe cray 3 -l mppwidth=4,mppnppn=2 /home/crayadm/job_script.sh
Your job 295 ("job_script.sh") has been submitted

> apstat
Compute node summary

<table>
<thead>
<tr>
<th>arch</th>
<th>config</th>
<th>up</th>
<th>resv</th>
<th>use</th>
<th>avail</th>
<th>down</th>
</tr>
</thead>
<tbody>
<tr>
<td>XT</td>
<td>24</td>
<td>23</td>
<td>2</td>
<td>2</td>
<td>21</td>
<td>1</td>
</tr>
</tbody>
</table>

> qstat -j 295
...
account: sge
hard_resource_list: mppwidth=4,mppnppn=2,cray_forced=1,mppdepth=1,cray_nodes=1
mail_list: crayadm@nid00034
notify: FALSE
job_name: job_script.sh
jobshare: 0
hard_queue_list: cray.q
env_list: 
script_file: /home/crayadm/job_script.sh
parallel environment: cray range: 2
context: reservation_1=4303
...
Interactive Job Example

> qrsh -pe cray 2 -q cray.q

# now on login node

> aprun -B /home/crayadm/installation/examples/jobsbin/lx-amd64/work 10
Forking -1 times.
Forking -1 times.
Application 3872 resources: utime ~200s, stime ~0s, Rss ~3572, inblocks ~0, outblocks ~0

# using the reservation requests: -B
> aprun -B /home/crayadm/installation/examples/jobsbin/lx-amd64/work -w 5
Forking -1 times.
Forking -1 times.
Application 3876 resources: utime ~10s, stime ~0s, Rss ~3572, inblocks ~0, outblocks ~0

> aprun -n 2 -d 32 /home/crayadm/installation/examples/jobsbin/lx-amd64/work -w 5
Forking -1 times.
Forking -1 times.
Application 3877 resources: utime ~10s, stime ~0s, Rss ~3572, inblocks ~0, outblocks ~0

> aprun -n 3 -d 32 /home/crayadm/installation/examples/jobsbin/lx-amd64/work -w 5
apsched: claim exceeds reservations node-count
Univa Grid Engine Features @ Cray XC

- Resource reservation for jobs / Advance Reservation
- Backfilling
- REST API
- Job dependencies between Cray XC and non-Cray jobs
- Job array support
- Job array inter-dependencies on task level
- Resource quotas on job classes, global level (Cray / non-Cray)
- Automatic resource alignment on Cray reconfiguration
- ...lots of policies and more
What is Cooking?

• UGE 8.4 is on the way → June 2016
  • Docker ready: Automatic detection of Docker enabled nodes. Execution of jobs in automatically created containers.
• UniSight 4 is knocking at the door → July 2016
  • Major re-write of UniSight 3
  • Now with live dashboard functionality for monitoring cluster
• Navops.io → Fastest way for deploying Kubernetes on bare metal or on cloud service providers (AWS, Google)
  • Free download! www.navops.io
Questions?
Thank You!

Speak with us! 😊