Examination of dynamic partitioning for multi-core jobs in the Tokyo Tier-2 center

Tomoe Kishimoto
ICEPP, The University of Tokyo

Mar. 10 2017
International Center for Elementary Particle Physics

R&D for ILC
MEG at PSI

ATLAS

TGC (KEK, Tokyo, TMU, Sinsyu, Nagoya, Kyoto, Kobe...)

DAQ (KEK, Sinsyu, Hiroshima-IT, Nagasaki-IAS)

High Level Trigger (KEK, TITEck, TITEch, Waseda, Kobe...)

Muon TDC (KEK)

Computing Center (Tokyo ICEPP)

Tokyo Tier-2 is the the only WLCG site in ATLAS-Japan

SCT (KEK, Tsukuba, TITEch, Ochanomizu, Kyusyu, Osaka...)

Solenoid (KEK)
ICEPP regional analysis center

✓ Resource overview
  - Support only ATLAS VO in WLCG (Tier2) and provide ATLAS–Japan dedicated resources (local use)
  - Hardwares are leased, and are replaced in every three years
  - 4th system is running since Jan. 2016
    - ~10000 CPU cores including service instances and ~10 PB disk storage (T2 + local use)

Single VO and uniform architecture

✓ Operation team
  - 5 university staffs + 2 SEs from company
Tier2 configuration of the 4th system

**Disk server (×48)**
- 132TB × 48 servers
- **Total capacity is 6.336PB** (DPM)
- 10Gbps NIC (for LAN)
- 8G–FC (for disk array)
  - 500~700MB/sec (sequential I/O)

**Worker node (×256)**
- 24 CPU cores/node, **total 6144 CPU cores**
- Memory: 2.66GB/core
- 10Gbps pass through module (SFP+ TwinAx cable)
- Rack mount type 10GE switch (10G BASE SR SFP+)
- Band width:
  - For 160 WNs: 10Gbps/2nodes (max 10Gbps, min 5Gbps)
  - For 96 WNs: 10Gbps/4nodes (max 10Gbps, min 2.5Gbps)
Tier2 configuration of the 4th system

Network

- 20Gbps to WAN
- Brocade MLXe-32 x 2
  Non-blocking 10Gbps
- Inter link
  16 x 10Gbps
- 10GE (SFP+)
  176 ports

Main switches: continued use from 3rd system

Tier2
- DPM file servers
- LCG service nodes
- LCG worker nodes

Non-grid
- GPFS/NFS file servers
- Tape servers
- Non-grid service nodes
- Non-grid computing nodes
Status in ATLAS

✓ Fraction of number of completed jobs

Results in the last six months:
- Production, 4.4% (Tier2) – 2.4% (All)
- Analysis, 4.7% (Tier2) – 2.9% (All)

← Good contributions

# of ATLAS–J authors ~ 100
# of ATLAS authors ~ 3000

✓ > 99 % site availability has been achieved using the 4th system
ATLAS job types

✓ Summary of ATLAS job types:

<table>
<thead>
<tr>
<th>Required cores</th>
<th>Target share</th>
<th>Production jobs</th>
<th>Analysis jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20%</td>
<td>Event generation, etc</td>
<td>User analysis</td>
</tr>
<tr>
<td>8</td>
<td>60%</td>
<td>Simulation, Reconstruction, etc</td>
<td></td>
</tr>
</tbody>
</table>

✓ Tokyo Tier2 site has been processing single-core jobs and multi-core jobs separately using dedicated WNs and CEs
- Simple configuration
- CREM-CE + Torque/Maui

✓ However, we have often observed idle CPU cores due to this static partitioning
Problem in static partitioning

Single-core job queue

Idle CPUs due to single-core jobs were not assigned

But, many multi-core jobs were waiting...

Multi-core job queue

CPU utilization 0.992862
Dynamic partitioning

✓ We started to evaluate an implementation of the dynamic partitioning using HTCondor batch scheduler
  - HTCondor is becoming standard in WLCG
  - Well documented

✓ In Nov. 2016, we deployed a small cluster (1536 cores) of HTCondor (+ ARC–CE) into the production:

<table>
<thead>
<tr>
<th>CE</th>
<th>Batch system</th>
<th>CPU cores</th>
<th>Job types</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREAM</td>
<td>Torque/Maui</td>
<td>4608 (192 WNs)</td>
<td>Single- and Multi-core jobs (static partitioning)</td>
</tr>
<tr>
<td>ARC</td>
<td>HTCondor</td>
<td>1536 (64 WNs)</td>
<td>Single- and Multi-core jobs (dynamic partitioning)</td>
</tr>
</tbody>
</table>
ARC–CE + HTCondor configuration

- Two ARC–CEs for redundancy
- High availability of central managers
- Draining for multi–core job is managed by Defrag daemon
  - Need optimizations

ARC version  5.0.4
HTCondor version  8.4.8
Optimization of draining

✓ In the dynamic partitioning, draining of single core jobs is necessary in order to dispatch new multi jobs

24 cores (slots) per WN

Draining

= single core job

= eight core job

✓ Defrag daemon in HTCondor is used to manage this draining

✓ Need to optimize several parameters

- When should draining start/end?
- Which machines are more desirable to drain?
- How many machines are drained at once?
When should draining start/end?

✓ Criteria for stating the draining:
  - \((N^{\text{multi-core running}} < N^{\text{multi-core target}}) \land (N^{\text{multi-core waiting}} > 0)\)

✓ Criteria for finishing the draining
  - \((N^{\text{multi-core running}} >= N^{\text{multi-core target}})\)

A cron script monitors \(N^{\text{multi-core running}}\) and \(N^{\text{multi-core waiting}}\), and sets DEFRA\_WHOLE\_MACHINE\_EXPR in Defrag daemon

→ The criteria are working well in the production system
Which machines are desirable to drain?

✓ Observed job duration in the HTCondor system:

![Graph showing job duration]

Period of 2016/11 to 2017/01

<table>
<thead>
<tr>
<th></th>
<th>Production (single-core)</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov.</td>
<td>$1.2 \times 10^4$</td>
<td>$0.7 \times 10^4$</td>
</tr>
<tr>
<td>Dec.</td>
<td>$1.8 \times 10^4$</td>
<td>$0.8 \times 10^4$</td>
</tr>
<tr>
<td>Jan.</td>
<td>$1.0 \times 10^4$</td>
<td>$0.7 \times 10^4$</td>
</tr>
</tbody>
</table>

✓ In the Tokyo Tier2 center, user analysis jobs tend to finish earlier than production jobs
Which machines are desirable to drain?

- Duration of draining until 8 slots are available in a WN:
  - Estimated by simulation, using the observed job duration as inputs

→ Desirable to drain WNs, which have many analysis jobs

✓ A cron script has been developed:
  - Monitors WNs status, and set `DEFRAG_RANK` for each WN
  - `DEFRAG_RANK = (N_{production\ running}) * 1 + (N_{analysis\ running}) * 2 + (# \ of \ free \ slots) * 3`
  - WNs with higher rank will be chosen for draining by Defrag daemon
  - All WNs are homogeneous in Tokyo Tier-2

* Implemented into the production at Feb.
How many machines are drained at once?

✓ Simulated queue status during the draining:
  - $N_{\text{machine}}^{\text{draining}}$ : number of concurrent draining machines

$N_{\text{machine}}^{\text{draining}} = 64$

If all machines (64 WNs) are drained constantly:

- **Pros**: the draining finishes earliest
- **Cons**: CPU utilization is lowest because of useless drainings

→ $N_{\text{machine}}^{\text{draining}}$ is dynamically changed based on progress of draining

$N_{\text{machine}}^{\text{draining}} = N_{\text{multi-core target}} - N_{\text{multi-core running}}$

---

Simulated queue status during the draining:

- $N_{\text{machine}}^{\text{draining}} = XX$

**constant**

**dynamic**
How many machines are drained at once?

- CPU utilization VS draining duration:
  - Estimated by simulation, 50% production (single) jobs and 50% analysis jobs are filled initially
  - X axis: Duration of draining until target share is achieved
  - Y axis: CPU utilization for $1.0 \times 10^6$ seconds

- Good CPU utilization and draining duration (★) are achieved by introducing the dynamic value of $N_{\text{machine draining}}$

A cron script monitors $N_{\text{multi-core running}}$ and calculates $N_{\text{machine draining}}$, then sets `DEFRAG_MAX_CONCURRENT_DRAINING` in Defrag daemon

* Implemented into the production at Feb.
Results by introducing dynamic partitioning

Static partitioning (Torque/Maui)

Dynamic partitioning (HTCondor)

Idle CPUs were compensated by the other type of jobs

The Dynamic partitioning is working very well
Observed CPU utilization

**Improvement of CPU utilization has been observed thanks to the dynamic partitioning**

**HTCondor is stable so far**

- We plan to migrate all CPUs from Torque/Maui to HTCondor in the near future

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>week1,2</td>
<td>week3,4</td>
<td>week1,2</td>
</tr>
<tr>
<td>Static partitioning (Torque/Maui)</td>
<td>98.8%</td>
<td>93.9%</td>
<td><strong>88.8%</strong></td>
</tr>
<tr>
<td>Dynamic partitioning (HTCondor)</td>
<td><strong>99.4%</strong></td>
<td><strong>98.0%</strong></td>
<td>94.8%</td>
</tr>
</tbody>
</table>

* Test jobs (i.e. ops job) are overcommitted in HTCondor system
Summary

✓ Tokyo Tier2 with the 4th system is running
  - Providing enough computing resources for ATLAS
  - > 99% site availability is achieved

✓ Dynamic partitioning has been introduced using HTCondor
  - Parameters of draining have been optimized based on the job features at Tokyo Tier2 center
    ▸ Draining starts/ends based on # of running/waiting multi-core jobs
    ▸ Analysis jobs tend to finish earlier
    ▸ $N_{\text{machine,draining}}$ is dynamically changed based on progress of draining
  - Improvement of CPU utilization has been observed
    ▸ e.g. ~5% improvement in 2016/12/15 to 2016/12/31
  - Will migrate all CPUs from Torque/Maui to HTCondor
Backup