PROCEEDINGS OF SCIENCE

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

T. Kishimoto[*1], **T. Mashimo**[1], **N. Matsui**[1], **T. Nakamura**[2], **H. Sakamoto**[1]

[1] *International Center for Elementary Particle Physics, The University of Tokyo*
[2] *Computing Research Center, KEK*

E-mail: tomoe@icepp.s.u-tokyo.ac.jp

The Tokyo Tier-2 center, which is located in the International Center for Elementary Particle Physics at the University of Tokyo, is providing computing resources for the ATLAS experiment in the Worldwide LHC Computing Grid. In order to utilize the computing resources at the Tokyo Tier-2 center more efficiently, the dynamic partitioning of worker nodes for the single- and multi-core jobs has been implemented using the HTCondor batch job scheduler. In the dynamic partitioning, draining of the single-core jobs is necessary in order to dispatch a new multi-core job. This paper reports optimization studies with respect to the job draining and shows an improvement of the resource utilization by introducing the dynamic partitioning.

International Symposium on Grids and Clouds 2017 -ISGC 2017-
5-10 March 2017
Academia Sinica, Taipei, Taiwan

[*] Speaker.

PoS(ISGC2017)001

1. Introduction

The Tokyo Tier-2 center, which is located in the International Center for Elementary Particle Physics (ICEPP) [1] at the University of Tokyo, is providing computing resources for the ATLAS experiment [2] in the Worldwide LHC Computing Grid (WLCG) [3]. The official site operation in the WLCG was launched in 2007 after developments since 2002, and the site has been achieving a stable operation since then.

The ATLAS experiment is aimed at performing various physics studies, such as measurements of the Higgs boson, searches for new phenomena, as well as measurements of the standard model processes. The experiment developed the multi-core implementation of their software framework for the reconstruction of physics objects, the detector simulation and so on, which provides an efficient memory sharing [4]. In 2014, the experiment started to submit multi-core jobs, which uses eight CPU cores, using the above software framework to the WLCG. The Tokyo Tier-2 center has been processing this multi-core job and standard single-core job (e.g. user analysis job) separately using dedicated worker nodes and computing elements (static partitioning). However, we have often observed idle CPU cores in the worker nodes due to this static partitioning when either multi-core or single-core jobs are not assigned to the site. Therefore, we started to evaluate an implementation of the dynamic partitioning of the worker nodes using HTCondor [5] batch job scheduler in order to increase the utilization. In November 2016, a small cluster of the HTCondor has been deployed into the production.

For the dynamic partitioning, draining of the single-core jobs is necessary in order to reserve job slots for the multi-core jobs when the worker node is filled by only single-core jobs. This job draining should be performed until the number of running multi-core jobs reaches a target share. In order to perform an efficient draining, we need to optimize several parameters, such as the number of draining machines at the same time, based on properties of the jobs.

In this paper, improvements of the CPU utilization by introducing the dynamic partitioning and optimization studies of the drain parameters in the Tokyo Tier-2 center will be reported. The paper is organized as follows, Section 2 provides the configuration of the Tokyo Tier-2 center. Section 3 shows the site status in the ATLAS experiment. Section 4 describes the issue in the static partitioning and configuration of the dynamic partitioning. Section 5 discusses optimizations of the draining parameters. Section 6 and 7 give results and a summary.

2. Site configuration

In the Tokyo Tier-2 center, almost all hardware devices are replaced in every three years in order to satisfy the requirement of the ATLAS experiment. The last hardware upgrade was done in January 2016, and the new system (so called 4th system) is running stably. Table 1 summarizes the available computer resources of the 4th system in comparison with the previous system (3rd system operated from 2013 to 2015). The total number of CPU cores, which includes CPU cores for service instances, was not increased from the 3rd system, and also the performance of individual CPU core is approximately same to the previous system according to HEP-SPEC06 [6] benchmark test. Each worker node consists of 24 CPU cores. The disk storage consists of 80 sets of a disk array and a file server. The total capacity of the disk storage is 10560 TB. 6144 CPU cores (256

worker nodes) and 7392 TB disk storages are reserved for the ATLAS experiment in the WLCG. The remaining resources are dedicated to Japanese collaborators as local resources.

	CPU cores	HEP-SPEC06	Disk [TB]
3rd system	9984 (3840 for WLCG)	18.03	6732
4th system	9984 (6144 for WLCG)	18.11	10560

Table 1: The computer resources in the 3rd and 4th systems.

The worker nodes and the file servers are connected to a central network switch. For 160 worker nodes, the internal network bandwidth to the central switch is 10 Gbps per two worker nodes. For the other 96 worker nodes, the bandwidth is 10 Gbps per four worker nodes. Each file server is also connected to the central switch by 10 Gbps . The central switch is connected to SINET5 [7], which is a Japanese academic backbone network, by 20 Gbps.

3. Status in ATLAS experiment

Figure 1 shows percentages of the number of completed jobs in the Tokyo Tier-2 center in February 2011 to February 2017. The dashed and solid line take the number of completed jobs in the all Tier-0+Tier-1+Tier-2 sites and the only Tier-2 sites as the denominator of the percentage, respectively. The ATLAS jobs are divided into two categories: one is production jobs (blue), which is managed by the experiment for such as the reconstruction process, and the other is user analysis jobs (red). The figure indicates that the percentages was decreasing in the period of 3rd system operated from 2013 to 2015 because the total resource in the WLCG was increasing, but it has been recovered at the 4th system since 2016. For example, the site has completed 4.4% of total production jobs in Tier-2s and 4.7% of total analysis jobs in Tier-2s in the last six months. These numbers show our sufficient contributions to the experiment since the fraction of ATLAS-Japan authors in the experiment roughly corresponds to 3–4%. Also the high site availability, which is greater than 99%, has been achieved using the 4th system.

4. Dynamic partitioning

Table 2 summarizes the types of ATLAS jobs and the target resource shares for each job type. The production job is divided further into the single-core job and the multi-core job. This multi-core job was developed by the ATLAS experiment for the efficient memory usage as described in Section 1.

	Required cores	Target share	
Production jobs	1	20%	Event generation, etc
	8	60%	Simulation, Reconstruction, etc
Analysis jobs	1	20%	User analysis

Table 2: Summary of the ATLAS job types.

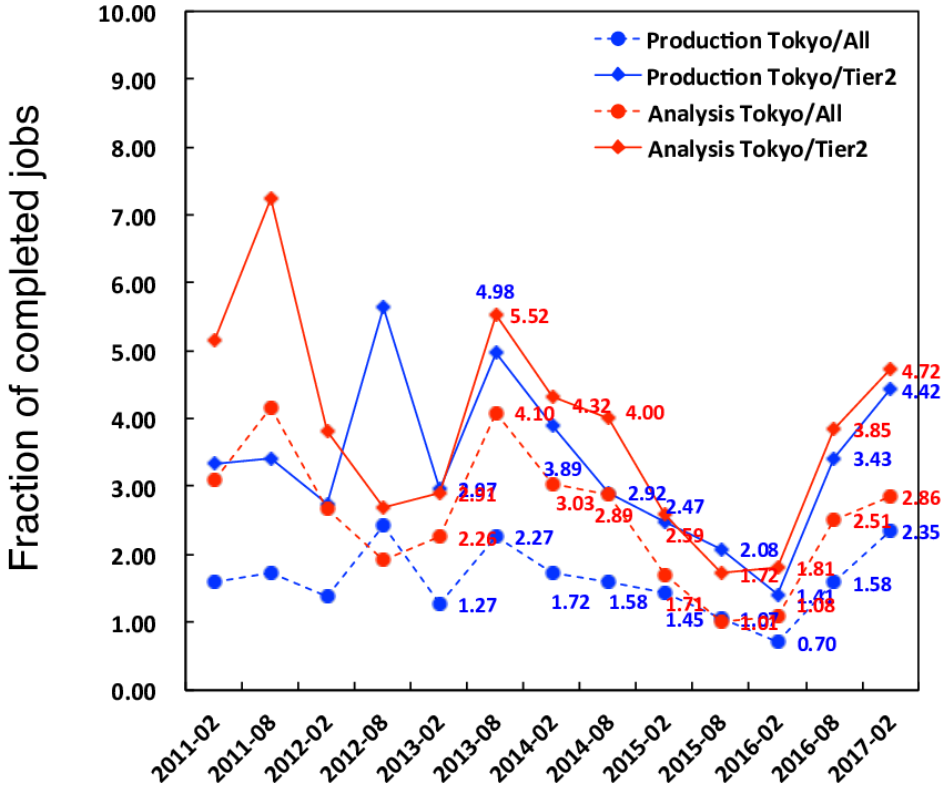
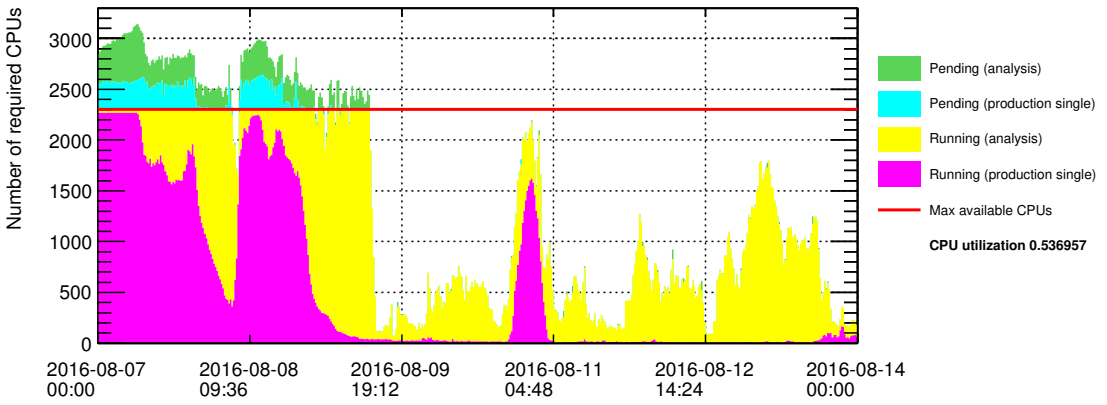


Figure 1: The percentages of the number of completed jobs in the Tokyo Tier-2 center for the last 7 years.

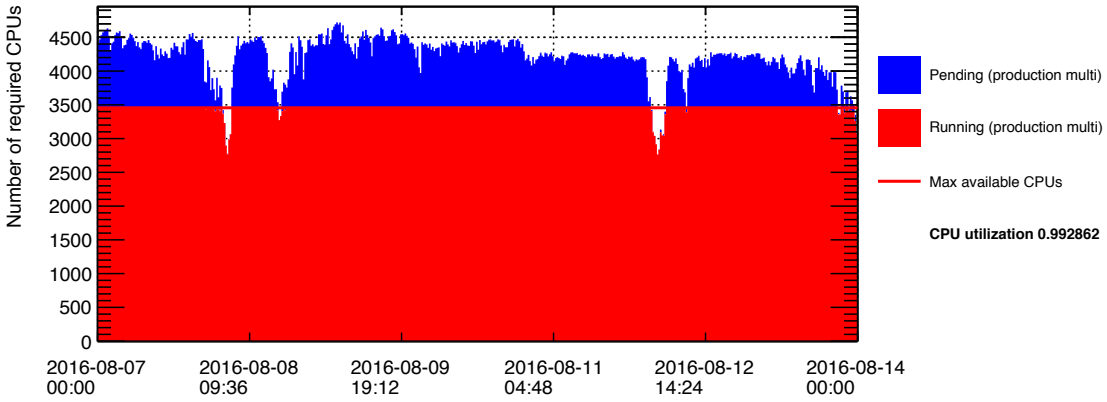
The Tokyo Tier-2 center has been processing the single-core job and the multi-core job separately in the static partitioning, which is managed by Torque [8] and Maui [9] as the batch system in combination with CREAM [10] computing element. However, we have often observed idle CPU cores in the worker nodes due to this static partitioning. Figure 2 shows the number of required CPU cores for running and pending jobs in the queues for the single-core job (a) and the multi-core job (b) respectively during a week in August 2016. The red horizontal lines indicate the number of maximum available CPU cores in the queues. One can find that there were idle CPU cores in the single-core job queue because the single-core jobs were not assigned to the site in this period. Meanwhile, there were many pending jobs in the multi-core job queue since the queue was full. In order to solve this issue and increase the utilization, we started to evaluate an implementation of the dynamic partitioning of the worker nodes using HTCondor batch job scheduler.

In November 2016, a small cluster of the HTCondor pool was deployed into the production. Figure 3 shows the configuration of the HTCondor in the Tokyo Tier-2 center. ARC-CE [11] is used as the computing element. Two ARC-CEs are installed for redundancy. The high availability configuration of the central manager of the HTCondor is also implemented. A feature of the HTCondor called Accounting Group is used to perform the dynamic partitioning. This Accounting Group is configured to share the total computing resources among the ATLAS job types based on the required targets as summarized in Table 2. A job type, e.g. multi-core job, is allowed to exceed

its target share if the resources are free. The Defrag daemon in the HTCondor manages the draining of the single-core jobs, which is discussed in Section 5. Table 3 summarizes the configuration of the computing resources in the production.



(a)



(b)

Figure 2: The number of required CPU cores for running and pending jobs in single-core job queue (a) and multi-core job queue (b).

CE	Batch system	CPU cores	Job type
CREAM	Torque/Maui	2304	single-core job
		2304	multi-core job
ARC	HTCondor	1536	single- and multi-core jobs

Table 3: Summary of the computing resources in the production.

5. Optimization of draining

The multi-core job requires eight CPU cores, which corresponds to eight single job slots in the Tokyo Tier-2 center. In the dynamic partitioning, if the slots are occupied by the single core jobs,

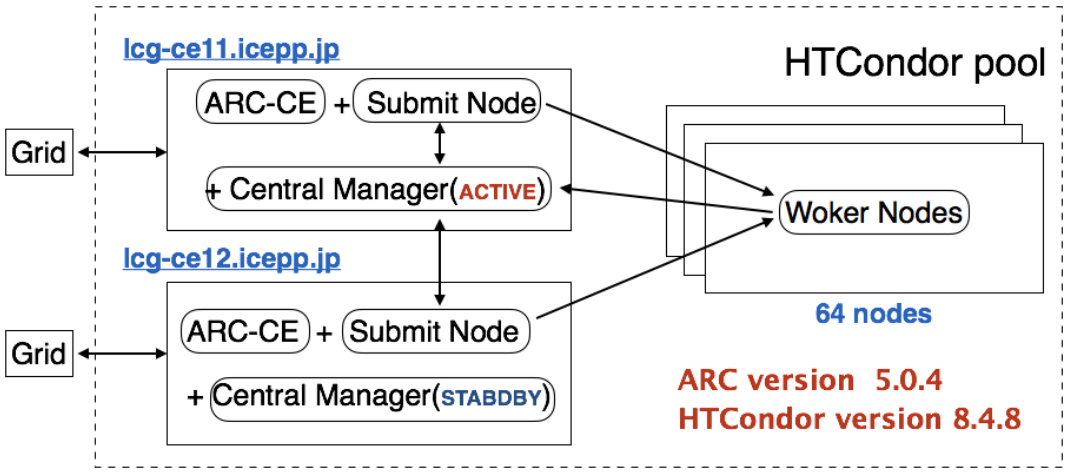


Figure 3: The Configuration of ARC-CE + HTCondor in the Tokyo Tier-2 center.

the single core-jobs need to be drained until eight slots are available in a worker node to dispatch a new multi-core job. This job draining is performed until the number of running multi-core jobs reaches the target share. In order to perform an efficient job draining, several parameters need to be optimized:

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

Optimization studies to determine these parameters are described in the following subsections.

5.1 When should draining start and end?

The job draining should start when the number of running multi-core jobs is less than the target share and there are pending multi-core jobs. The criteria for staring the draining are defined as,

$$(N^{\text{multi-core}}_{\text{running}} < N^{\text{multi-core}}_{\text{target}}) \;\&\&\; (N^{\text{multi-core}}_{\text{pending}} > 0), \tag{5.1}$$

where $N^{\text{multi-core}}_{\text{running(pending)}}$ is the number of running (pending) multi-core jobs, respectively. $N^{\text{multi-core}}_{\text{target}}$ is the target share of the number of running multi-core jobs. The job draining should be terminated when the number of running multi-core jobs reaches the target share, which is described as,

$$N^{\text{multi-core}}_{\text{running}} \geq N^{\text{multi-core}}_{\text{target}}. \tag{5.2}$$

A cron script has been implemented to monitor $N^{\text{multi-core}}_{\text{running}}$ and $N^{\text{multi-core}}_{\text{pending}}$ in the HTCondor pool, and control the job draining. Figure 4 shows a queue status in the dynamic partitioning after introducing these draining criteria. The yellow and pink histograms indicate the number of running single-core jobs and the red histogram indicates the number of running multi-core jobs. The target share is set as 50% in this case. It can be seen that the queue is initially filled by the single jobs, then the the number of running multi-core jobs are increasing to the target share. Thus, it has been confirmed that the criteria are working well at the system in the production.

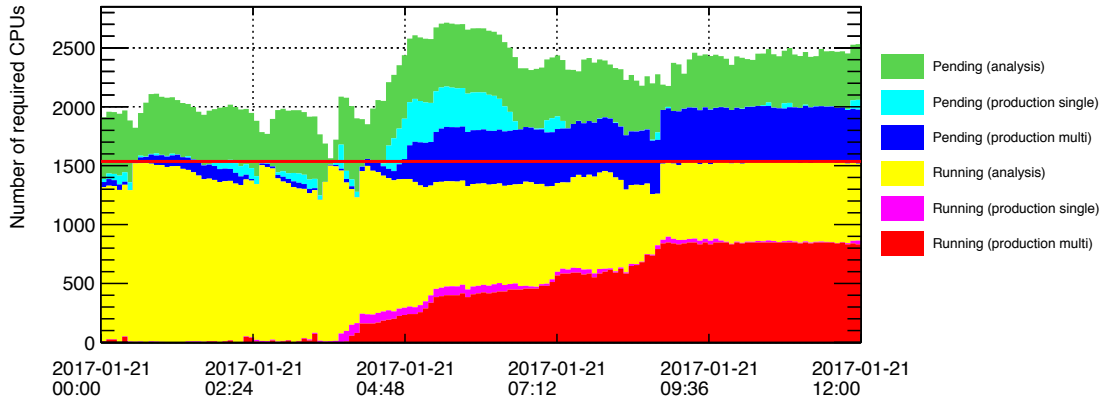
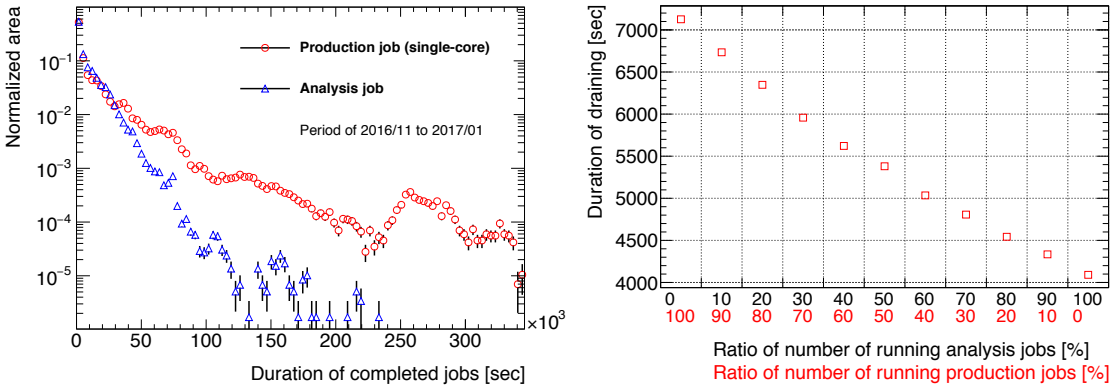


Figure 4: The number of running and pending multi-core jobs in the period of 2017-01-21 00:00:00 to 2017-01-21 12:00:00 (UTC).

5.2 Which machines are more desirable to drain?

The job scheduling with shorter time for the job draining is necessary to avoid the waste of CPU cores. Figure 5 (a) shows distributions of the observed duration time of the single-core jobs for the three months. The red and blue dots indicate the duration time of the production and analysis jobs, respectively. Also Table 4 summarizes the average job duration time. One can see that the user analysis jobs tend to finish earlier than the production jobs in the Tokyo Tier-2 center.



(a) (b)

Figure 5: (a) The observed job duration time of the single-core jobs in the Tokyo Tier-2 center. (b)

	Nov. 2016	Dec. 2016	Jan. 2017
Production job	1.2×10^4 sec	1.8×10^4 sec	1.0×10^4 sec
Analysis job	0.7×10^4 sec	0.8×10^4 sec	0.7×10^4 sec

Table 4: Average job duration time of the single-core jobs in the Tokyo Tier-2 center.

We simulated the time required for the job draining to obtain eight available job slots in a

worker node using the observed job duration time as inputs. Figure 5 (b) shows the results in terms of the ratio of the number of running analysis (or production) jobs in the initial state. There is almost factor two difference of the time for job draining if we compare the case of all jobs are the analysis jobs with the case of all jobs are the production jobs. Thus, the worker node, which has many analysis jobs, is more desirable to drain. A cron script has been developed to calculate a rank (DEFRAG_RANK) of the worker node. The worker node with many analysis jobs gets the higher rank by simple calculation as,

$$\text{DERRAG\_RANK} = (\text{N}_{\text{running}}^{\text{production}} \times 1) + (\text{N}_{\text{running}}^{\text{analysis}} \times 2) + (\#\text{ of free slots} \times 3), \qquad (5.3)$$

where $\text{N}_{\text{running}}^{\text{production(analysis)}}$ is the number of running production (analysis) jobs in the worker node. The worker node with the higher rank will be chosen for the job draining by the Defrag daemon. This simple calculation works well in the case of all worker nodes are homogeneous.

5.3 How many machines are drained at once?

We also simulated the queue status during the job draining. As a simple configuration, if all machines (64 worker nodes in this case) are drained constantly, which is shown in Figure 6 (a), the draining finishes the earliest but the the CPU utilization is the lowest because of unnecessary job slots reservations at the end of drain completion. In order to achieve the higher utilization, the number of concurrent draining machine ($\text{N}_{\text{draining}}^{\text{machine}}$) should not be constant but should be dynamically changed based on the progress of job draining. $\text{N}_{\text{draining}}^{\text{machine}}$ is defined as,

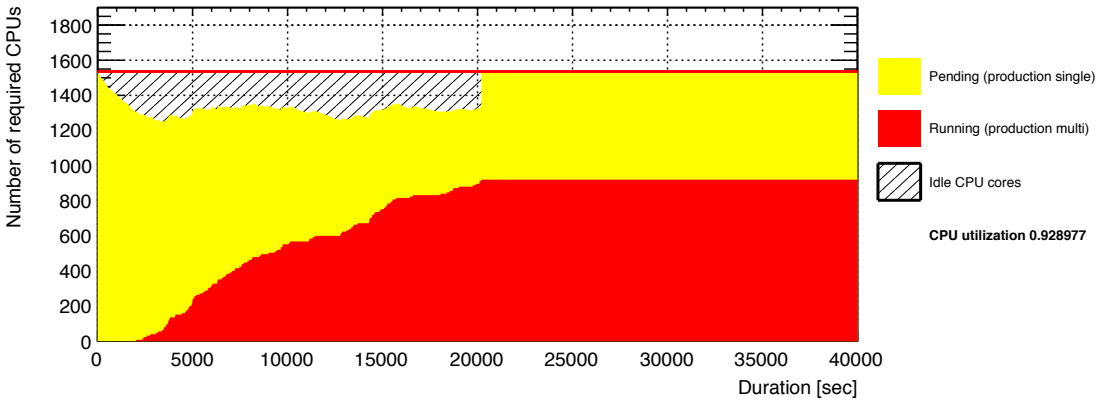
$$\text{N}_{\text{draining}}^{\text{machine}} = \text{N}_{\text{target}}^{\text{multi-core}} - \text{N}_{\text{running}}^{\text{multi-core}}. \qquad (5.4)$$

The simulated queue status during the job draining with this dynamic value of $\text{N}_{\text{draining}}^{\text{machine}}$ is shown in Figure 6 (b).

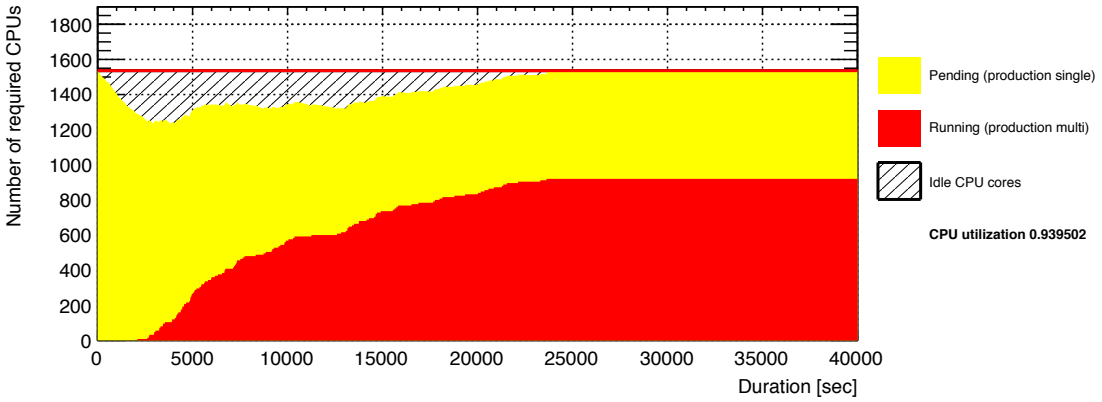
Figure 7 summarizes the CPU utilization for 1.0×10^6 seconds versus the time for job draining estimated by the simulation. The black circles show the results obtained by the constant parameter with respect to the number of draining machine. As already mentioned, if 64 worker nodes are drained constantly ($\text{N}_{\text{draining}}^{\text{machine}} = 64$), the time for job draining is the shortest but the CPU utilization is the lowest. On the other hand, if only one worker node is drained ($\text{N}_{\text{draining}}^{\text{machine}} = 1$), the CPU utilization is the highest but the time for job draining is too long. By introducing the dynamic value of $\text{N}_{\text{draining}}^{\text{machine}}$ as defined as Equation 5.4, the better CPU utilization and time for job draining is achieved, which is shown as the red star on Figure 7. This dynamic value of $\text{N}_{\text{draining}}^{\text{machine}}$ has been implemented to the system in production by monitoring $\text{N}_{\text{running}}^{\text{multi-core}}$ and calculating $\text{N}_{\text{draining}}^{\text{machine}}$.

6. Results

Figure 8 shows a comparison of the observed queue share between the static partitioning and the dynamic partitioning at the system in production. The yellow, pink and red histograms indicate the number of running analysis, single-core production and multi-core production jobs. The green, light blue and blue histograms indicate the number of pending analysis, single-core production and multi-core production jobs. The red horizontal lines indicate the maximum available CPU core in the queues. For example, idle CPU cores were observed in the static partitioning around



(a)



(b)

Figure 6: The simulated queue status during the draining. The yellow histogram shows the number of running single-core jobs and the red histograms shows the number if running multi-core jobs. (a) All machines (64 worker nodes) are drained constantly. (b) The number of concurrent draining machines is dynamically changed by following Equation 5.4.

2016-11-24 due to the multi-core jobs were not assigned to the site. Meanwhile, the number of idle CPU cores in the dynamic partitioning was small compared to the static partitioning since the single-core jobs were filled over the target share. Table 5 summarizes the CPU utilization in the static partitioning and the dynamic partitioning for the three months after introducing the dynamic partitioning. The CPU utilizations have been improved by introducing the scheme of dynamic partitioning. For example, the improvement was about 4% in 2016-11-15 to 2016-11-30. Therefore, we have decided to migrate all CPU cores from Torque/Maui to HTCondor in the near future.

7. Summary

The Tokyo Tier-2 center is providing sufficienct computing resources to the ATLAS experiment in the WLCG, and is keeping the high site availability. The dynamic partitioning for the

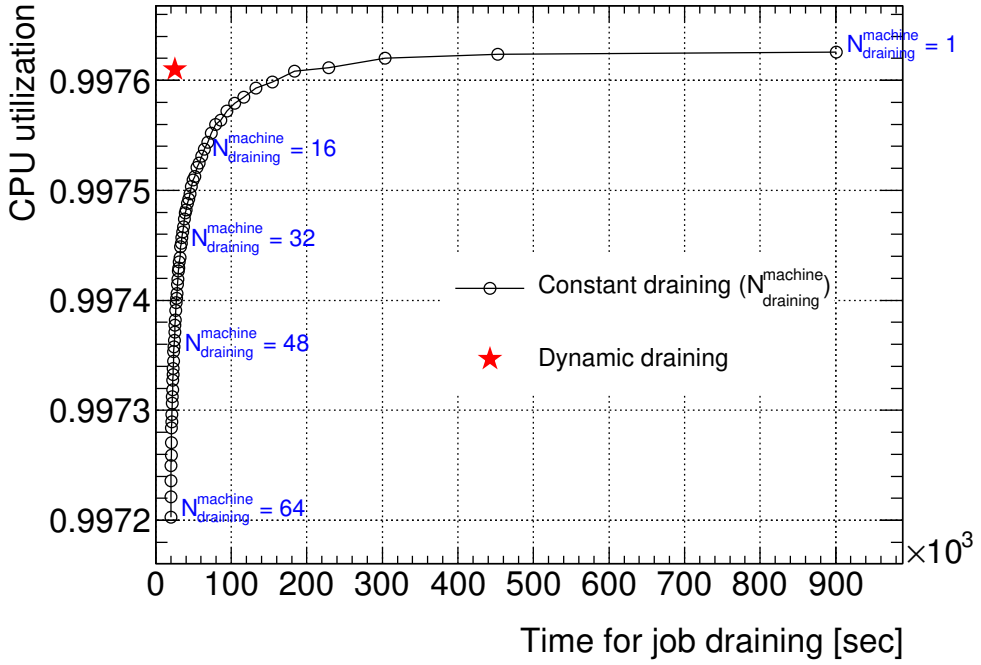
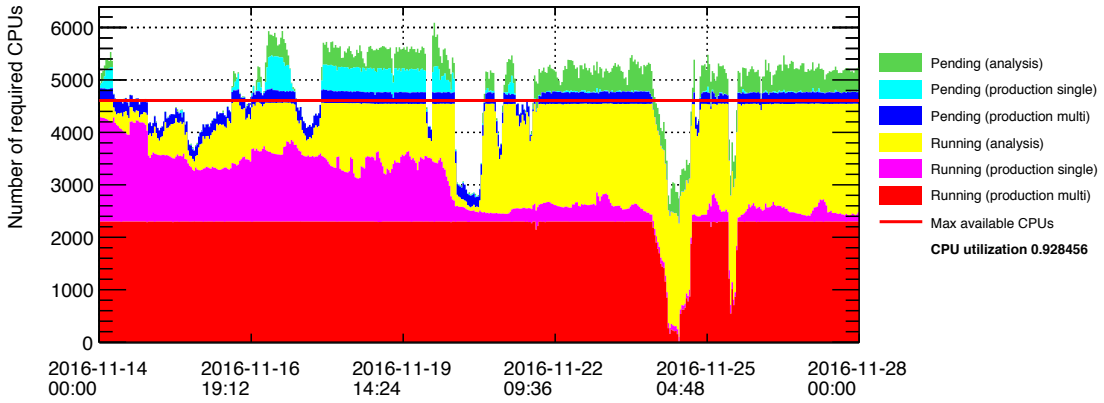


Figure 7: The CPU utilization versus the time for job draining duration estimated by the simulation. 50% production (single) jobs and 50% analysis jobs are filled initially.

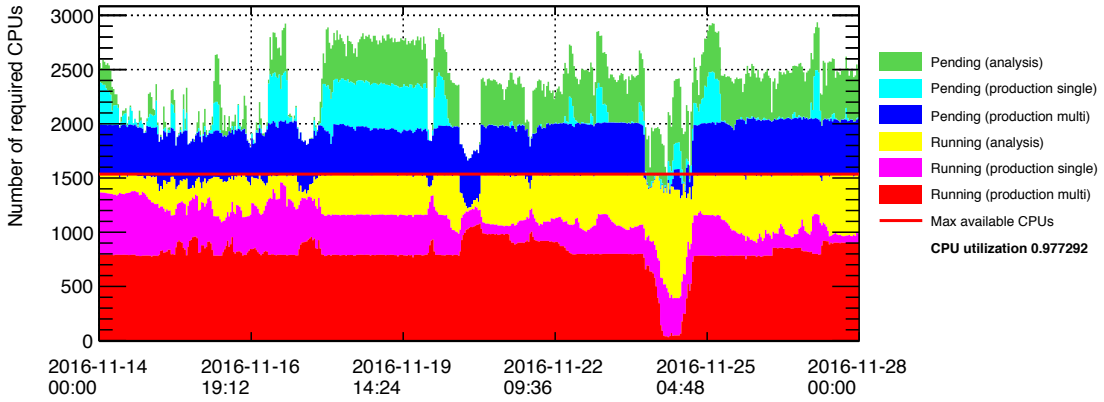
	Nov. 2016		Dec. 2016		Jan. 2017	
	week 1,2	week 3,4	week 1,2	week 3,4	week 1,2	week 3,4
Static partitioning (Torque/Maui)	98.8%	93.9%	*88.8%	94.2%	95.1%	**75.0%
Dynamic partitioning (HTCondor)	**99.4%**	**98.0%**	94.8%	**98.5%**	**98.4%**	**91.1%

Table 5: The observed CPU utilization in the static partitioning and the dynamic partitioning. * There was a pbs_server crush. * There were software maintenances. Effects of the draining optimizations, which are discussed in Section 5.2 and 5.3, are not included in the table since the optimizations were deployed in the production recently (February 2017).

multi-core jobs has been implemented using the HTCondor batch job scheduler in order to utilize the computing resources at the site more efficiently. In the dynamic partitioning among the job types, the draining of the single-core jobs is necessary to dispatch a new multi-core job if the worker node is filled by only the single-core jobs. In order to perform the efficient job draining, the simulation studies have been performed. Based on the simulation studies, we developed a scheme to control the timings to start and finish the job draining, assign the rank of worker nodes for the job draining and calculate the number of max concurrent draining machines by monitoring the queue status. We observed about 4% improvement on the CPU utilization by introducing the dynamic partitioning in 2016-11-15 to 2016-11-30. Therefore, we will migrate all CPU cores from



(a)



(b)

Figure 8: The observed queue status in the static partitioning (a) and dynamic partitioning (b) in the period of 2016-11-15 00:00:00 to 2016-11-28 00:00:00.

the Torque/Maui to the HTCondor in the near future.

References

[1] ICEPP web page, https://www.icepp.s.u-tokyo.ac.jp/en/index.html

[2] ATLAS experiment web page, http://atlas.cern/

[3] WLCG web page, http://wlcg.web.cern.ch/

[4] Crooks, D et al., "Multi-core job submission and grid resource scheduling for ATLAS AthenaMP", ATL-SOFT-PROC-2012-029, https://cds.cern.ch/record/1449084/

[5] HTCondor web page, https://research.cs.wisc.edu/htcondor/

[6] HEP-SPEC06 web page, http://w3.hepix.org/benchmarks/doku.php

[7] SINET web page, https://www.sinet.ad.jp/en/top-en

[8] Torque web page, http://www.adaptivecomputing.com/products/open- source/torque

[9] Maui web page, http://www.adaptivecomputing.com/products/open- source/maui/

[10] CREAM web page, https://wiki.italiangrid.it/CREAM

[11] ARC-CE web page, http://www.nordugrid.org/arc/ce/