applied sciences

MDPI

Article

An Efficient Approach to Consolidating Job Schedulers in Traditional Independent Scientific Workflows

Byungyun Kong [1], Geonmo Ryu [1], Sangwook Bae [1], Seo-Young Noh [2] and Heejun Yoon [1,*]

[1] Global Science Experimental Data Hub Center, Korea Institute of Science and Technology Information, 245 Daehak-ro, Yuseong-gu, Daejeon 34141, Korea; kong91@kisti.re.kr (B.K.); geonmo@kisti.re.kr (G.R.); wookie@kisti.re.kr (S.B.)

[2] Department of Computer Science, Chungbuk National University, Chungdae-ro 1, Seowon-Gu, Cheongju, Chungbuk 28644, Korea; rsyoung@cbnu.ac.kr

* Correspondence: k2@kisti.re.kr; Tel.: +82-42-869-1042

Received: 31 December 2019; Accepted: 18 February 2020; Published: 21 February 2020

Abstract: The current research paradigm is one of data-driven research. Researchers are beginning to deploy computer facilities to produce and analyze large amounts of data. As requirements for computing power grow, data processing in traditional workstations is always under pressure for efficient resource management. In such an environment, a tremendous amount of data is being processed using parallel computing for efficient and effective research results. HTCondor, as an example, provides computing power for data analysis for researchers. Although such a system works well in a traditional computing cluster environment, we need an efficient methodology to meet the ever-increasing demands of computing using limited resources. In this paper, we propose an approach to integrating clusters that can share their computing power on the basis of a priority policy. Our approach makes it possible to share worker nodes while maintaining the resources allocated to each group. In addition, we have utilized the historical data of user usage in order to analyze problems that have occurred during job execution due to resource sharing and the actual operating results. Our findings can provide a reasonable guideline for limited computing powers shared by multiple scientific groups.

Keywords: HTCondor; multi-cluster; job-scheduler; preemption

1. Introduction

Research methodology has changed from traditional observational methods to a data-driven research paradigm, which is called the fourth generation research paradigm [1]. According to the paradigm of existing scientific research, there has been a change from the first generation paradigm that describes natural phenomena through observation, the second generation through modeling and generalization, and the third generation research paradigm using computer simulation technology [2]. Recently, a data-driven research paradigm that analyzes and makes discoveries using vast amounts of data from large research equipment has emerged [3,4]. Efficiently analyzing this vast amount of data requires massive amounts of computing power [5,6]. In general, computer clustering technologies are utilized in computing resources of groups of various sizes, from small lab units to data centers. This is called high throughput computing (HTC) [7]. A job management program is required for batching jobs and managing queues to use the HTC system more efficiently, such as HTCondor [8–10], PBSPro [11,12], Slurm [13,14], and Torque [15,16]. HTCondor is an open source program that was released in 1988 (the name was Condor at the time of release [10]) and has an active open source community.

We currently operate a data center to support researchers in basic science [17]. Not only do new researchers continue to ask for support, but the demand for resources annually submitted by existing researchers continues to grow [18]. Figure 1 shows that resource demand steadily increases by about 2000–3000 cores each year [19]. The total amount of demand will increase from 3900 cores in 2019 to 13,700 cores in four years. This trend is not confined to domestic, but also to international research groups [20]. However, due to the limited size of equipment that can be introduced on a limited budget, it is impossible to meet the requirements of all researchers. However, the user's request is to meet peak demand at a specific point in time, and since the utilization rate of the cluster is not always high, we integrated the dedicated clusters of each research group into one in order to satisfy this as much as possible.

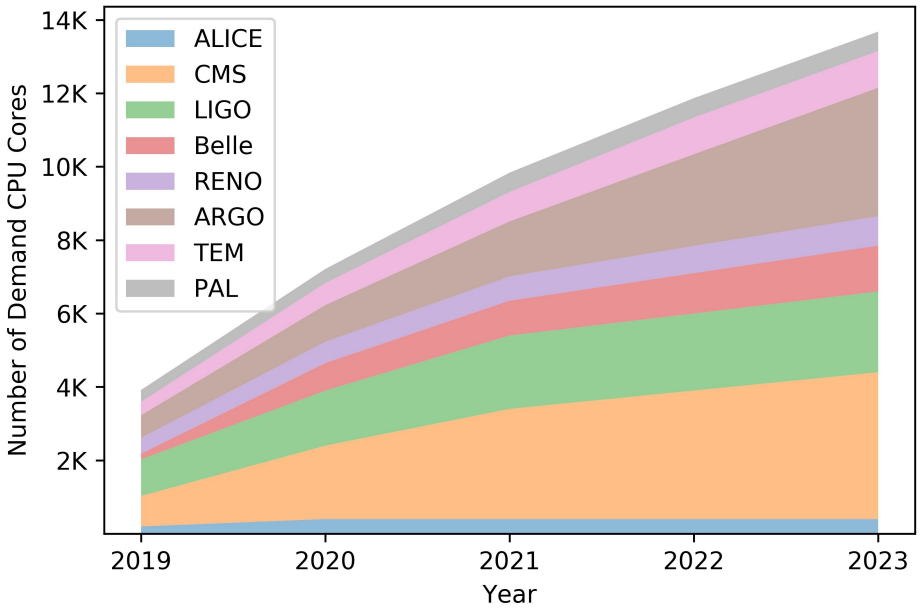


Figure 1. Results of resource demand survey requested by a group of Korean researchers.

First, we chose to change the existing system without deploying a new system as a method for management of integrated resources [21]. Accordingly, we chose to share worker nodes using existing HTCondors but limit quotas of individual groups. In this case, it is difficult to continuously match the information of newly added or deleted users and groups. To solve this problem, we separated the job submission node. Users' access and defined user groups are based on the submission node. We also considered how the temporary resources that occur occasionally can be effectively utilized while maintaining the minimum allocated resources for each group [22].

In this paper, we discuss how shared resources can be used simultaneously and how the minimum allocated resources can be maintained for each group. We also analyzed the user's job histories to examine changes in job processing patterns. We also analyzed TimeLoss according to job characteristics.

2. Materials and Methods

2.1. Job History Information

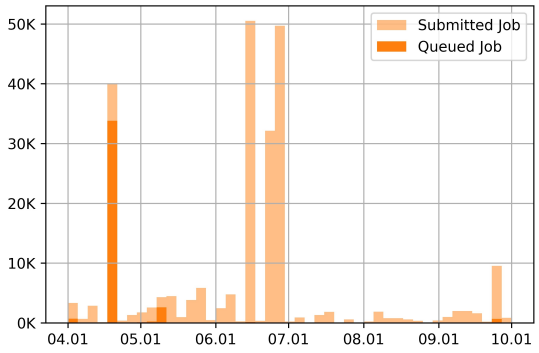
We extracted the job history information through HTCondor's `condor_history` command. The user's job histories were extracted and analyzed from April to September of 2018 before the integration and the same period of 2019 after the integration. The information used in the analysis is `AcctGroup`, `AcctGroupUser`, `CMD` for job categorization and `CommittedTime`, `CumulativeSlotTime`, `JobCurrentStartDate`, `JobCurrentStartExecutingDate`, `JobStartDate`, `NumJobStarts`, and `QDate` for job characteristics. The meaning of each item is summarized in Table 1.

Table 1. Meaning of items extracted with `condor_history` [23].

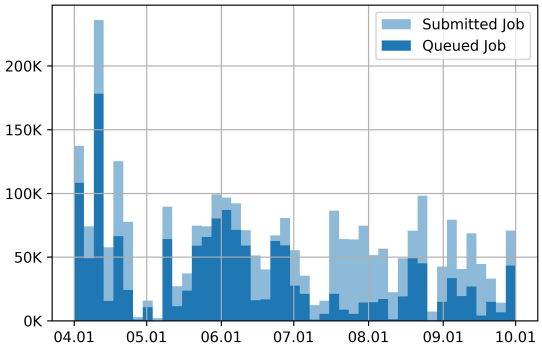
Item	Mean
AcctGroup	Group information for submitted jobs
AcctGroupUser	User information for submitted jobs
CMD	Executed command
CommittedTime	The number of seconds of wall clock time that the job has been allocated to a machine
CumulativeSlotTime	Cumulative number of seconds the job has been allocated to a machine
JobCurrentStartDate	Time at which the job most recently began running
JobStartDate	Time at which the job first began running
NumJobStarts	An integer count of the number of times the job started executing
QDate	Time at which the job was submitted to the job queue

2.2. Status before Integration

Before the integration, each experiment group used dedicated computing resources. Each experimental group could only use the dedicated resources assigned to them. For this reason, the congestion level of the cluster varies according to the group members who follow the same schedule as the conferences or workshops. In other words, the cluster is crowded at certain times, but most of the time it is idle. The trend of job submissions for each group is shown in Figure 2. As shown in Figure 2, Group A actively submitted jobs in mid-April and June, while Group B did in early April. It is important to note that there are time mismatches for the utilization of computing resources dedicated to the two groups.



(**a**) Group A



(**b**) Group B

Figure 2. Submitted and queued jobs in 2018.

In the extracted job records, jobs with different submitted times and job started times were classified. The time difference was assumed to be longer than the cycle reported by the job manager. In the HTCondor Job Manager, this cycle is called the matchmaking cycle [24]. These jobs mean that the job cannot run immediately because there are no slots available when the job is submitted. A percentage of 15.99% of jobs submitted from Group A were accumulated in the job queue, and 55.46% of jobs submitted from Group B

were accumulated in the job queue. The number of jobs submitted is not proportional to the number of jobs waiting, as in Figure 2a,b. This is because the submitted jobs did not overlap in a short time.

Long overlap periods reduce the benefits of cluster consolidation. According to the job histories of the year 2018, the percentage of jobs in the queue overlaps 3.37% of the total for Group A and 0.29% of the total for Group B, and in fact jobs rarely overlap between the two groups. The overlapping periods can be seen in Figure 3. Therefore, we expected high efficiency when the two groups were integrated.

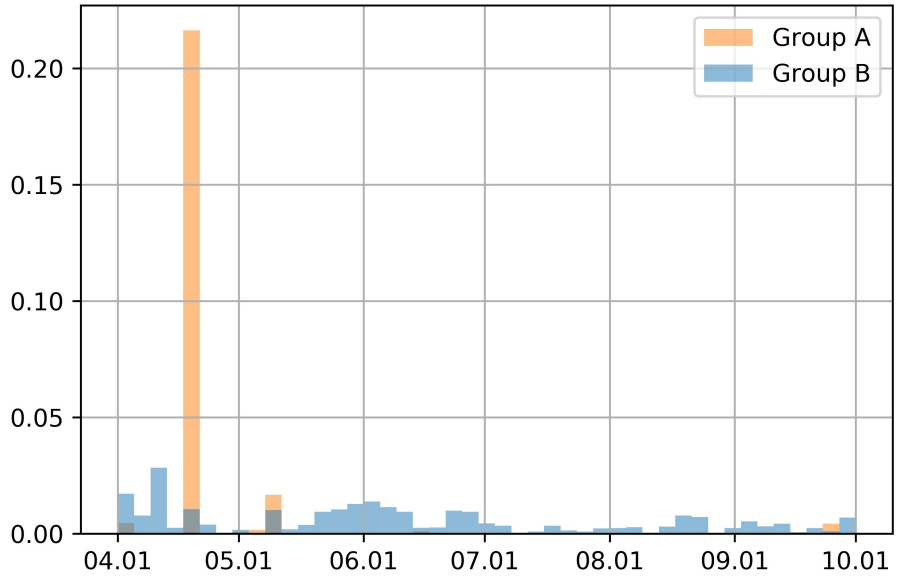


Figure 3. Comparison of the queued job for two groups in 2018.

Despite the above situation, individual users feel that there is a resource shortage as demand increases, and they want to increase the size of the overall cluster. During this period, the utilization rate of Group A is 10.24%, and the utilization rate of Group B is 20.40%, so the utilization rate of the entire cluster is not high. Therefore, it is difficult to guarantee that the overall cluster utilization will increase as you increase the cluster size. In other words, if we provide more resources for each experiment group, resource utilization will still be similar. The integration of the two group's clusters provides a way to meet the needs of users while increasing utilization across the clusters.

2.3. Deployment of the Integrated Cluster

2.3.1. Configuration of the Integrated Cluster

Target clusters for integration have similar properties and provide close functionalities in terms of analytical tools or individual jobs used as experimental groups in the same field. The analysis environment is similar, so no extra action had been required to shared worker nodes. The integrated cluster is composed of independent job submit nodes, and there is a common job management node, and there are worker nodes. The two groups to be consolidated were assigned worker nodes with 400 cores and 1656 cores, respectively. In order to integrate the two groups, we set the minimum guaranteed quota to the number of cores of the worker nodes owned by each group. Storage, not computing resources, is set up to be accessible from the same configuration and all worker nodes. The problem caused by the difference of OS versions before integration was solved by introducing Singularity, a Linux container program [25–27]. Figure 4 is a schematic of the integration cluster.

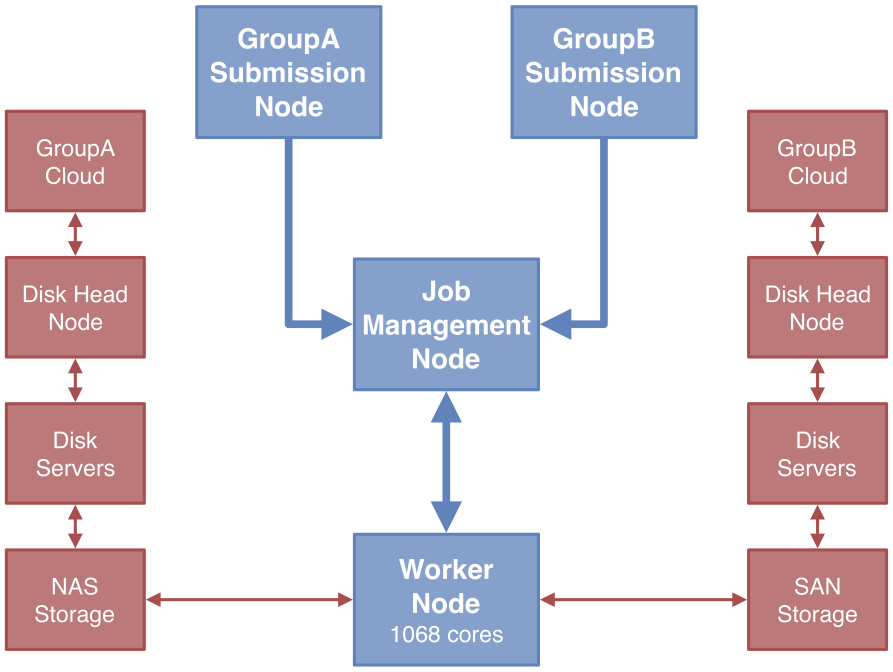


Figure 4. The job management node in the center of the figure and the worker node below it are shared by the two groups, but the job submission node (UI) and storage system are separately organized.

2.3.2. Configuration of Submission Node

Because job submission nodes in each group are separated, we can identify user groups by job submission nodes. The user's group authentication was based on the UI server information submitted by the user, not information such as ID. In this concept, HTCondor does not restrict the group information of the user's submission, so it adds a setting to refuse to submit the job to the user access node such as `GROUP_NAMES = group_a`, `SUBMIT_REQUIREMENT_NAMES = GROUP`, and `SUBMIT_REQUIREMENT_GROUP = (AcctGroup =?= ''group_a'')`. This setting forces the users to clarify a valid group name for their jobs. If a user submits a job with the wrong group name, the job manager rejects the job and informs the correct name to user. For example, `SUBMIT_REQUIREMENT_GROUP_REASON = ''Wrong accounting group. Your group is group_a''`.

2.3.3. Configuration of the Job Management Node

In the job management node, the job is assigned to a worker node by matching the requirements from the submitted job and the information of the worker node. Group information and the quota of each group were set in this node, such as `GROUP_NAMES = group_a, group_b, group_etc`, `GROUP_QUOTA_group_a = 400`, `GROUP_QUOTA_group_b = 1656`, and `GROUP_QUOTA_group_etc = 80`. A setting that can be used in excess of the set quota when there are extra slots in the other group's slot (`GROUP_ACCEPT_SURPLUS = True`) and a setting to preempt a slot when another group uses more than their own quota (`PREEMPT = True`, `NEGOTIATOR_CONSIDER_PREEMPTION = True`, `PREEMPTION_REQUIREMENTS = True`, and `PREEMPTION_REQUIREMENTS = $(PREEMPTION_REQUIREMENTS) && ((SubmitterGroupResourcesInUse < SubmitterGroupQuota) && (RemoteGroupResourcesInUse > RemoteGroupQuota))`) were added. The question of which slot to preempt thus arises. `PREEMPTION_RANK = 2592000 - ifThenElse(isUndefined(TotalJobRuntime),0,TotalJobRuntime)` is in this setting. This setting allows specific jobs to perform beyond the group's quotas. When a job is requested by a different group of users, it is necessary to terminate the running job over the quota. Furthermore, a returned slot is then assigned to that group. In this case, we have set the most recent job to be preempted. Users may think that they are losing time, but it is not at all a loss because it is a job that would have to wait in the queue if resources were not consolidated.

3. Results and Discussion

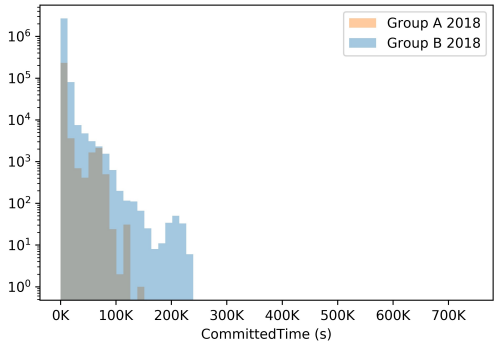
3.1. Characterization of Jobs before and after Integration

We found changes in job processing patterns in both periods through job statistics before and after integration. The statistical information is shown in Table 2. The number of jobs processed increased by about 84% in Group A, from 242,963 in 2018 to 447,944 in 2019, and decreased by about 35% in Group B, from 2,788,872 in 2018 to 1,799,363 in 2019. The wall time of submitted jobs increased by about 587% in Group A, from 647,731,692 s in 2018 to 4,450,001,236 s in 2019, and decreased by about 25% in Group B, from 5,342,505,530 s in 2018 to 4,015,125,597 s in 2019, respectively. Job waiting time increased by about 1203% in Group A, from 201,578,262 s in 2018 to 2,625,918,725 s in 2019, and decreased by about 59% in Group B, from 15,587,662,341 s in 2018 to 6,370,466,789 s in 2019.

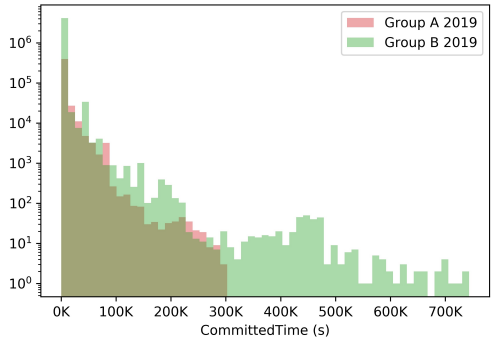
Table 2. Statistics for submission jobs in 2018 and 2019.

	Number of Jobs Processed		Total WallTime (s)		Total WaitingTime (s)	
	Group A	**Group B**	**Group A**	**Group B**	**Group A**	**Group B**
2018	242,963	2,788,872	647,731,692	5,342,505,530	201,578,262	15,587,662,341
2019	447,944	1,799,363	4,450,001,236	4,015,125,597	2,625,918,725	6,370,466,789
	+84.37%	−35.48%	+587.01%	−24.85%	+1202.68%	−59.13%

The execution time of each job was analyzed to characterize the individual jobs. For Group A, the average execution time in 2018 is 2648.35 s, and the quartiles are 28.00, 84.00, and 956.00 s, while the average execution time of 2019 is 6061.84 s, and the quartiles are 95.00, 1482.00, and 6331.00 s. For Group B, the average time of 2018 is 1890.27 s, and the quartiles are 152.00, 402.00, and 965.00 s, and for 2019, the average time is 1414.18 s, and quartiles are 31.00, 130.00, and 645.00 s. In Group A, the execution time of individual jobs more than doubled from the previous year, while in Group B they tended to decrease. The distribution of job execution time is summarized in Figure 5.



(**a**) in 2018



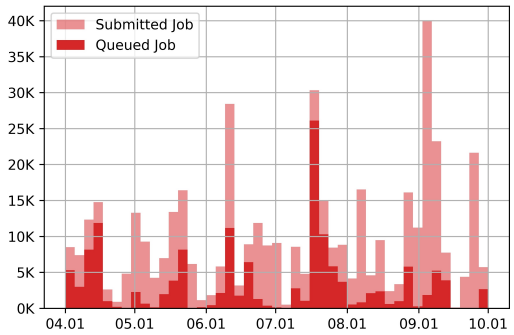
(**b**) in 2019

Figure 5. Individual job committed time distribution.

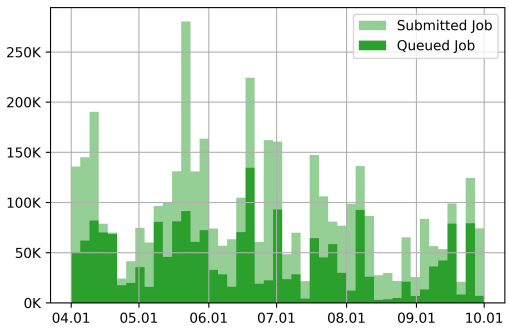
3.2. Analyzing Submitted Jobs after Integration

3.2.1. Job Submission Trends

In the same way that the 2018 job statistics were compared, the trend patterns of the job submitted by each group in 2019 and the level of overlap between the two groups were checked. Comparing Figures 2 and 6 shows that the timing of job submissions changed.



(**a**) Group A



(**b**) Group B

Figure 6. Submitted and queued jobs in 2019.

Similarly, we also compared the trends of queued jobs. In the case of Group A, the ratio in the queue during the submission process increased to 32.97%, and in Group B the ratio in the queue during the submission process decreased to 46.98%. In fact, the overlapping intervals of the jobs stacked in the queue are shown in Figure 7—15.79% of the total in Group A and 1.68% of the total in Group B. Both groups increased compared to before the integration. This part appears to be a natural change due to the increase and decrease in the amount of job processing and wall time of Groups A and B. In particular, users in Group A submitted larger jobs than in 2018 as the total slots grew. In fact, the utilization rate of Group A increased significantly to 70.36%, and the utilization rate of Group B decreased to 15.33%. In conclusion, the utilization rate of the entire cluster was 25.06%, which was 16.64% before the integration, that is higher than before despite the side effect due to integration.

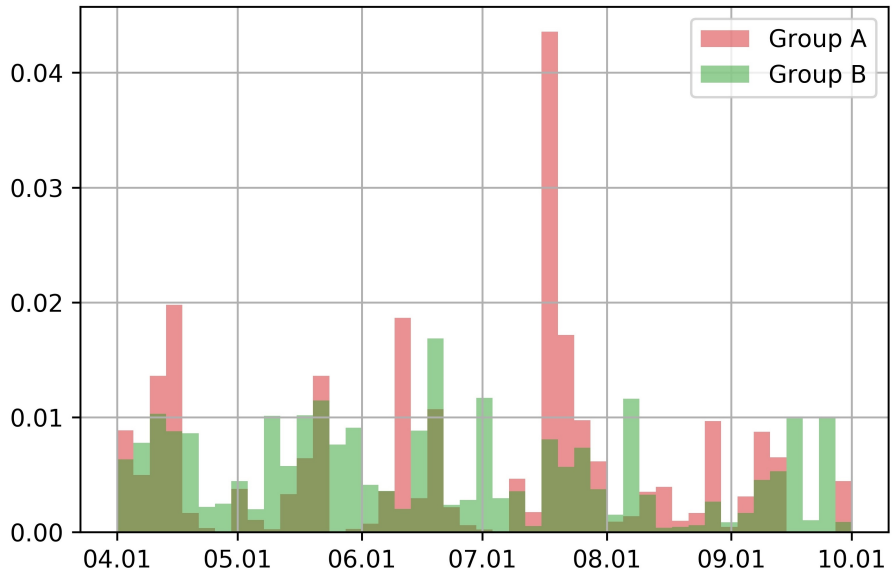
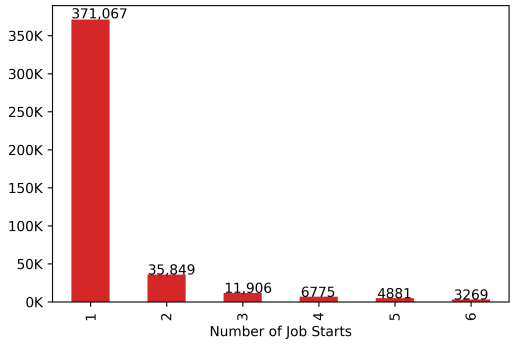


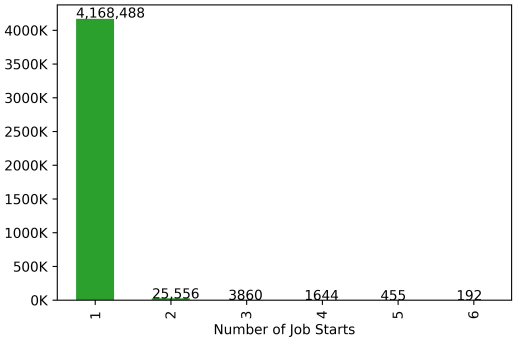
Figure 7. Comparison of queued jobs for two groups in 2019.

3.2.2. Characteristics of Preempted Jobs

When a slot was preempted, the job assigned to the slot had to be restarted when an idle slot was available. Therefore, the job with a `NumJobStarts` value of 2 or more was judged as preempted, and the characteristics of the job were examined. The distribution of `NumJobStarts` used in the analysis is shown in Figure 8. Percentages of 17.10% in Group A and 0.76% in Group B were preempted job ratios. It was noted that the ratio difference between the two groups resulted from a fourfold difference in size.



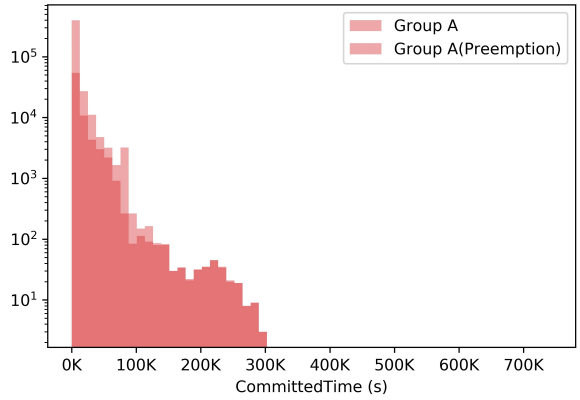
(**a**) Group A



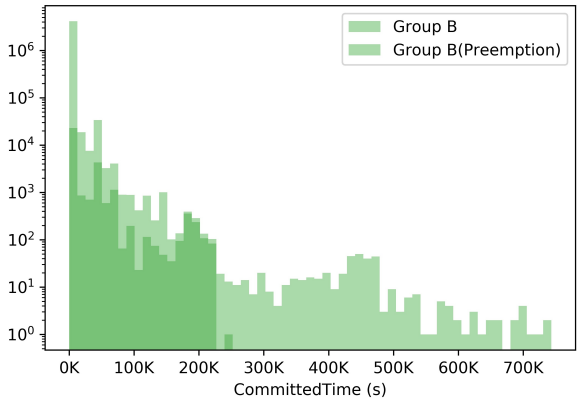
(**b**) Group B

Figure 8. Job-start distribution.

We also looked at the situation of preempted jobs and preempted possibility along their committed time. The result is shown in Figure 9. In both groups, the longer the execution time was, the higher the probability of being preempted was. However, Group A maintained a high probability, while Group B tended to have nothing preempted after a certain period.



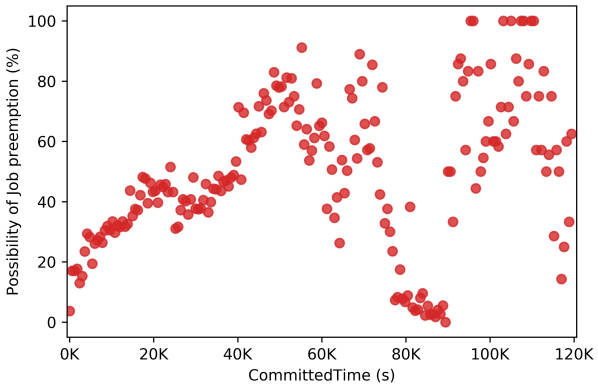
(**a**) Group A



(**b**) Group B

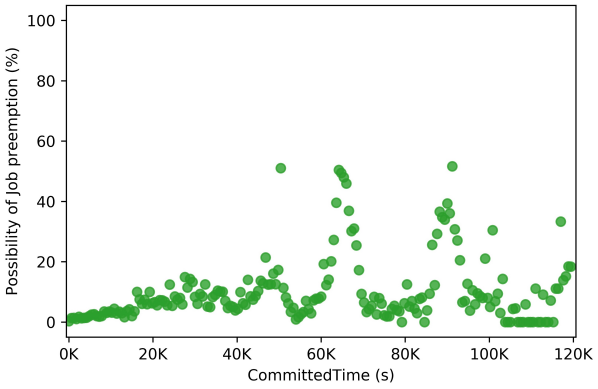
Figure 9. Distribution of preempted jobs by committed time.

In the probability graph at Figure 10, redrawing only up to the percentage of committed time 99 would look like Figure 11. Both groups tend to grow linearly, but Group A grows faster than Group B.



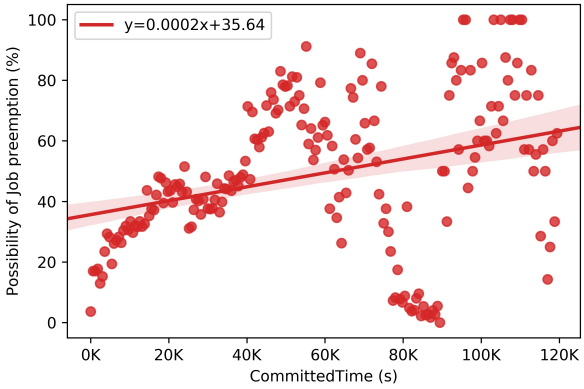
(**a**) Group A

Figure 10. *Cont.*

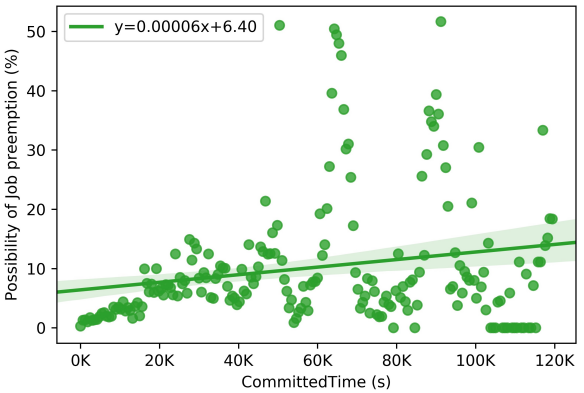


(**b**) Group B

Figure 10. Probability of preempted jobs by committed time.



(**a**) Group A

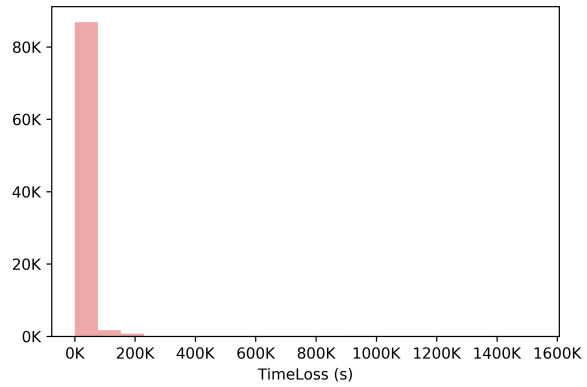


(**b**) Group B

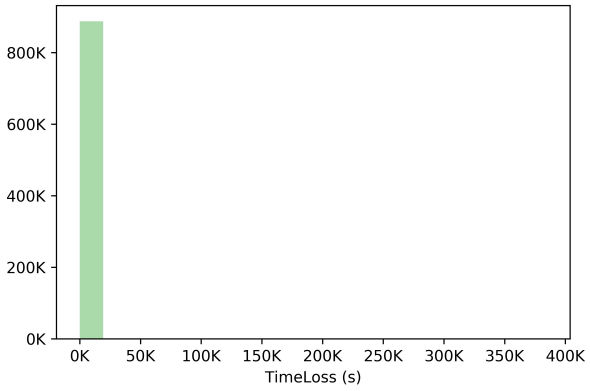
Figure 11. Probability of preempted jobs by committed time until Percentile 99.

3.2.3. Time Loss along Preemption

If a slot is preempted, that is, in excess of the quota, the running job is canceled, and time is lost. We calculated these loss times to measure the inefficiency of the preemption. As Figure 12 shows, most of the time lost in the preemptive slots is small. For Group A, the average 19,003 s quartiles are 502, 3149, and 10,363 s. For Group B, the mean is 419 s, and the quartiles are 1, 2, and 25 s.



(**a**) Group A



(**b**) Group B

Figure 12. TimeLoss of preempted jobs.

4. Conclusions

As scientific discoveries through data analysis form the mainstream of research, there is a need for efficient use of available resources that are distributed and used independently. In the course of the study, we showed examples of practical applications about the sharing of computing systems used in different experimental groups. However, in the process of integrating resources, the effect that can be obtained depending on the size of the resources was different. When integrating two groups of different resources, the small group will have benefits from more resources than before, and the larger group will see little benefit, which is expected. In practice, small groups benefited by up to 527.0% of resource utilization, while large groups benefited by 127.4% and thus saw no significant benefit. Meanwhile, the probability of a job being preempted or of time loss caused by being preempted was greater in the small group than in the large group. Ultimately, when consolidating resources, all groups obtain a reasonable benefit. These benefits will contribute to academic development by increasing the utilization of resources by researchers. In the viewpoint of computing resources, you can use the same resources more efficiently and meet the requirements of users. In addition, the results discussed in the paper might provide a basis for persuading groups of different sizes. In the future, we will integrate groups of various sizes and further study the benefits and effects of each group.

Author Contributions: Conceptualization, S.-Y.N.; data curation, B.K.; funding acquisition, S.-Y.N. and H.Y.; investigation, S.B.; methodology, G.R.; project administration, H.Y.; validation, S.B.; visualization, G.R.; writing—original draft, B.K.; writing—review & editing, G.R., S.B., and S.-Y.N. All authors have read and agree to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Korea (NRF) through contract NRF-2010-0018156 and the Korea Institute of Science and Technology Information (KISTI) through the Program of Construction and Operation for the Large-Scale Science Data Center (K-19-L02-C03 & K-20-L02-C04).

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Hey, A.; Tansley, S.; Tolle, K. *The Fourth Paradigm: Data-Intensive Scientific Discovery*; Microsoft Research: Redmond, WA, USA, 2009.
2. Agrawal, A.; Choudhary, A. Perspective: Materials informatics and big data: Realization of the 'fourth paradigm' of science in materials science. *APL Mater.* **2016**, *4*, 053208. [CrossRef]
3. Trajanov, D.; Zdraveski, V.; Stojanov, R.; Kocarev, L. Dark Data in Internet of Things (IoT): Challenges and Opportunities. In Proceedings of the 7th Small Systems Simulation Symposium, Niš, Serbia, 12–14 February 2018.
4. Oliveira, S.F.; Fürlinger, K.; Kranzlmüller, D. Trends in computation, communication and storage and the consequences for data-intensive science. In Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, Liverpool, UK, 25–27 June 2012; pp. 572–579.
5. Cukier, K.; Mayer-Schoenberger, V. The rise of big data. *Foreign Aff.* **2013**, *92*, 29.
6. Hershey, P.C. Data Analytics Implications. *IEEE Potentials* **2018**, *37*, 10–11. [CrossRef]
7. Livny, M.; Basney, J.; Raman, R.; Tannenbaum, T. Mechanisms for High Throughput Computing. *SPEEDUP J.* **1997**, *11*, 36–40.
8. Thain, D.; Tannenbaum, T.; Livny, M. Distributed computing in practice: The condor experience. *Concurr. Pract. Exp.* **2005**, *17*, 323–356. [CrossRef]
9. Tannenbaum, T.; Wright, D.; Miller, K.; Livny, M. Condor—A Distributed Job Scheduler. In *Beowulf Cluster Computing with Linux*; Sterling, T., Ed.; The MIT Press: Cambridge, MA, USA, 2002; ISBN 0-262-69274-0.
10. HTCondor, High Throughput Computing. Available online: https://research.cs.wisc.edu/htcondor (accessed on 29 December 2019).
11. Henderson, R.; Tweten, D. *Portable Batch System: External Reference Specification*; Technical Report; Ames Research Center: Mountain View, CA, USA, 1996.
12. PBS. Available online: http://pbspro.org (accessed on 29 December 2019).
13. Jette, M.A.; Yoo, A.B.; Grondona, M. SLURM: Simple Linux Utility for Resource Management. In Proceedings of the Job Scheduling Strategies for Parallel Processing (JSSPP), Seattle, WA, USA, 24 June 2003; Lecture Notes in Computer Science; pp. 44–60.
14. Slurm. Available online: https://slurm.schedmd.com (accessed on 29 December 2019).
15. Chlumský, V.; Klusáček, D.; Ruda, M. The extension of TORQUE scheduler allowing the use of planning and optimization algorithms in Grids. *Comput. Sci.* **2012**, *13*, 5–19. [CrossRef]
16. Torque. Available online: https://www.adaptivecomputing.com/products/open-source/torque (accessed on 29 December 2019).
17. Kim, B.; Ahn, S.; Khan, T.; Jnag, H. Introduction of GSDC project and activities. In Proceedings of the International Symposium on Grids and Clouds and the Open Grid Forum, Taipei, Taiwan, 19–25 March 2011; SISSA Medialab: Trieste, Italy, 2011; p. 13.
18. Cho, K. (KISTI, Daejeon, Korea). Personal communication, 2019.
19. Cho, K. *8th GSDC Resources Review Boards*; KISTI: Daejeon, Korea, 2019.
20. Albrecht, J.; Alves, A.A.; Amadio, G.; Andronico, G.; Anh-Ky, N.; Aphecetche, L.; Bagliesi, G. A Roadmap for HEP Software and Computing R&D for the 2020s. *Comput. Softw. Big Sci.* **2019**, *3*, 7.
21. Kong, B.; Ryu, G.; Noh, S. Multi-experimental Support Through HTCondor Scheduling Policy Integrated Pool Configurationr. *Platf. Technol. Lett.* **2019**, *6*, 17–20.
22. Ahn, U.; Jaikar, A.; Kong, B.; Yeo, I.; Bae, S.; Kim, J. Experience on HTCondor batch system for HEP and other research fields at KISTI-GSDC. *J. Phys. Conf. Ser.* **2017**, *898*, 182938. [CrossRef]
23. HTCondor Manual. Available online: https://htcondor.readthedocs.io/en/latest/classad-attributes/job-classad-attributes.html (accessed on 29 December 2019).
24. Raman, R.; Livny, M.; Solomon, M. Matchmaking: Distributed Resource Management for High Throughput Computing. In Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, Chicago, IL, USA, 28–31 July 1998.

25. Ryu, G.; Bae, S.; Noh, S.; Yoon, H. A study on performance degradation when using Singularity container of HTCondor scheduler. *Platf. Technol. Lett.* **2019**, *6*, 1.
26. Kurtzer, G.M.; Sochat, V.; Bauer, M.W. Singularity: Scientific containers for mobility of compute. *PLoS ONE* **2017**, *12*, e0177459. [CrossRef] [PubMed]
27. Singularity. Available online: https://singularity.lbl.gov (accessed on 29 December 2019).