Reducing Faulty Jobs by Job Submission Verifier in Grid Engine

Misha Ahmadian
Department of Computer Science
Texas Tech University
Lubbock, TX, USA
misha.ahmadian@ttu.edu

Eric Rees
High Performance Computing Center
Texas Tech University
Lubbock, TX, USA
eric.rees@ttu.edu

Yu Zhuang
Department of Computer Science
Texas Tech University
Lubbock, TX, USA
yu.zhuang@ttu.edu

Yong Chen
Department of Computer Science
Texas Tech University
Lubbock, TX, USA
yong.chen@ttu.edu

ABSTRACT

Grid Engine is a Distributed Resource Manager (DRM), that manages the resources of distributed systems (such as Grid, HPC, or Cloud systems) and executes designated jobs which have requested to occupy or consume those resources. Grid Engine applies scheduling policies to allocate resources for jobs while simultaneously attempting to maintain optimal utilization of all machines in the distributed system. However, due to the complexity of Grid Engine's job submission commands and complicated resource management policies, the number of faulty job submissions in data centers increases with the number of jobs being submitted. To combat the increase in faulty jobs, Grid Engine allows administrators to design and implement Job Submission Verifiers (JSV) to verify jobs before they enter into Grid Engine. In this paper, we will discuss a Job Submission Verifier that was designed and implemented for Univa Grid Engine, a commercial version of Grid Engine, and thoroughly evaluated at the High Performance Computing Center of Texas Tech University. Our newly developed JSV communicates with Univa Grid Engine (UGE) components to verify whether a submitted job should be accepted as is, or modified then accepted, or rejected due to improper requests for resources. It had a substantial positive impact on reducing the number of faulty jobs submitted to UGE by far. For instance, it corrected 28.6% of job submissions and rejected 0.3% of total jobs from September 2018 to February 2019, that may otherwise lead to long or infinite waiting time in the job queue.

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07…$15.00
https://doi.org/10.1145/3332186.3338408

CCS CONCEPTS

• Computing methodologies~Parallel computing methodologies • Computing methodologies~Distributed computing methodologies

KEYWORDS

Job Submission Verifier, Grid Engine, Faulty Jobs

ACM Reference format:

Misha Ahmadian, Eric Rees, Yu Zhuang and Yong Chen. 2019. Reducing Faulty Jobs by Job Submission Verifier in Grid Engine. In *Proceedings of ACM Practice & Experience in Advanced Research Computing Conference 2019 (PEARC'19),* July 28-August 1, 2019*, Chicago, IL, USA, 8 pages.* https://doi.org/10.1145/3332186.3338408

1 Introduction

The Distributed Resource Management (DRM) systems, commonly called Job (or Workload) Schedulers, provide an interactive interface for users to request specific resources on a distributed system such as a high-performance computing (HPC) cluster, a Grid, or a cloud system. Users submit their requests to the DRM system and wait until the amount of physical resources such as CPU cores and memory, or virtual demanded resources such as software licenses are acquired for their jobs. The DRM system will then assign and execute the designated job on the requested resources.

Distributed resource management systems, such as Univa Grid Engine (UGE) [1] , Sun Grid Engine (SGE) [2], PBS Pro [3], and Slurm [4], often force users to adopt different formats for requesting resources based on the DRM's command syntax, design, and policies. Due to the complexity of DRMs' job submission commands and complicated resource management policies, the number of faulty job submissions in data centers increases with the number of jobs being submitted. In order to reduce faulty job submissions, some DRM system, such as Grid Engine, provide a programming interface for developers and system administrators to design and implement a Job Submission

Verifier (JSV) to control users' job submissions by *rejecting*, *correcting*, or *accepting* the jobs based on specific criteria before the DRM receives those jobs.

In this paper, we introduce a research and development effort of designing and implementing a job submission verifier within the UGE job scheduler. We have also carried out extensive evaluation tests on two large-scale, production HPC clusters, Quanah and Hrothgar, located at the High Performance Computing Center of Texas Tech University. Using this method, we have observed a substantial reduction of the number of faulty job submissions and completely wiped out jobs that would have waited forever due to resource requirements that could never be met. It also helps to train users by using a JSV and notifying them of their mistakes in an informative way while significantly reducing system administrators' time and efforts by resolving the faulty job submission issue automatically.

The contribution of this study is three-fold. First, we identify and demonstrate faulty job submission issues commonly observed in data centers, HPC systems, Grid or cloud systems. Second, we propose two novel Job Submission Verifiers (JSV) to address these issues, and we introduce the design and a reference implementation conducted in this research. More specifically, our proposed JSVs were developed in Perl, with around 980 lines of codes for Quanah cluster and about 460 lines of code for Hrothgar cluster. Third, we have conducted extensive evaluation tests on two production HPC systems in our data centers, and the evaluation results confirm that the proposed JSVs have reduced the number of faulty job submissions significantly. Such a reduction of faculty job submissions translates to quick turnaround time for users, reduced time and efforts required from system administrators' attention, and improved system utilization. We believe such a JSV can have an impact on resource management of current and future distributed computing infrastructures.

The rest of this paper is organized as follows. In Section II, we will discuss the background and motivation of this study. In Section III, we will explain the concept of JSV in detail, as well as our design and architecture of JSVs for two production clusters. We will also present our analysis and observations in this section. In Section IV, we will discuss relevant work in this space, and we will conclude this research study in Section V.

2 Background and Motivation

2.1 Job Schedulers and Univa Grid Engine

Job scheduler is a critical component in distributed computing systems and is often the most important factor that determines the overall system efficiency and utilization. Numerous job schedulers exist, such as UGE [1], SGE [2], PBS Pro [3], and Slurm [4]. Most of these job schedulers, however, bear very similar design and implementation philosophy. Thus, in this study, we focus on UGE as it is a commonly-used and a commercial version of job scheduler, Grid Engine. Our proposed Job Submission Verifier provides a general design, which can be applicable to any job schedulers, but the current implementation was carried out specifically for the UGE scheduler.

As a branch of Sun Grid Engine (SGE) [2] and later Oracle Grid Engine (OGE), Univa Grid Engine (UGE) [1] is a well-known DRM system used by many academic and private institutions worldwide. UGE uses a complex set of tunable scheduling and resource allocation policies to allocate the requested resources to jobs. In this context, a *resource* refers to any physical or virtual computing asset such as CPU cores, GPU cores, available memory, storage space, software license or any static characteristic or attributes within the computing environment. A *job* is defined as any process that can be executed from a command-line interface or issued from a GUI such as gateway systems. Sample jobs include executable binaries, shell scripts, MPI (Message Passing Interface) jobs, shared-memory applications such as OpenMP or Pthreads jobs, distributed executable tasks such as MapReduce or TensorFlow applications, or even interactive terminal sessions [5].

UGE maintains a queue of submitted jobs along with a list of available and occupied resources. In order to allow administrators to define how resources should be exploited and how jobs should be prioritized, UGE provides a rule system referred to as *scheduling policies*, similar to other job schedulers. These policies provide a method for prioritizing jobs based on a number of criteria including the job's user, group, queue, project or job type and allow the resource manager to govern the resources in a fair yet predictable manner. These scheduling policies along with the list of used, currently available and projected resources allow UGE to decide which job should take which resource or which job must give up the resources to make them free for other jobs waiting on the list.

To assist better understanding of this study, we briefly explain a list of common but critical terminologies used in UGE policy definitions that we use throughout this paper [5]:

- *Projects:* a set of configuration properties in UGE, which controls resource privileges and policies for a group of users, jobs, or queues.
- *Queue:* maintains a group of hosts for running jobs on those hosts when requested amount of resources are available.
- *Parallel Environment (PE):* a runtime environment for shared-memory or distributed-memory parallel applications, in which users can request for a number of required cores per job.
- *Resource Quota Sets (RQS):* a set of configuration properties to apply limits for the consumption of resources of any job requests.

2.2 Faulty Job Submissions

In this paper, we define faulty job submissions as belonging to one of two categories: 1) jobs that enter into "Error State" immediately after job submission and are shown as "Eqw" (or Error-Queue-Wait) status in UGE jobs list. These faulty jobs may emerge due to unusual behavior of the UGE scheduler or an issue with the user's executable script rather than deterministic

manners such as application runtime error or bad input file; and 2) jobs that are incapable of ever being scheduled and thus go into "queue wait" and stay in "qw" state forever. These faulty jobs often occur either by accident or user ignorance of the scheduler's policies and design.

For instance, if computational nodes in a cluster consist of x CPU cores per node, system administrators may desire to enforce distributed-memory (MPI) jobs consume all the x CPU cores of the allocated node(s). In this case, users are expected to request ($n * x$) CPU cores for their MPI job submission, in which n is the number of nodes that needs to be allocated to the job. However, since UGE accepts any number for CPU cores in job submissions, if user request for an amount of CPU cores which is not a multiple of the number of CPU cores per node (x), UGE receives the job submission, but will leave the job in waiting queue (qw) forever due to the conflict with the design and policy. Furthermore, in such a case, the UGE will fail to inform the user of what caused her/his job to stay in 'qw' mode and has no chance to obtain cluster resources.

This type of faulty job submissions, on average, requires 10 minutes of system administration work per job in order to determine the cause of job failure, find an appropriate solution, inform the user about her/his mistake, clear the faulty job, and respond to the user's feedback. Thanks to the more deterministic behavior of jobs that are unable to schedule and the available interface in UGE for verifying the jobs before entering the UGE, it would be feasible to reduce or eliminate faulty job submissions on HPC clusters. In this paper, we will introduce our proposed Job Submission Verifiers (JSV) to address the challenges of faulty jobs.

3 Job Submission Verifier (JSV)

Job Submission Verifier (JSV) is designed as a shell script or executable binary that runs as a process and communicates with Univa Grid Engine to verify jobs before being sent to the UGE scheduler components [6]. During the job verification process, the JSV manages to modify (*Correct*) the users' requests based on determinable criteria, drop (*Reject*) the potential faulty jobs with a wrong and uncorrectable request format, or allow (*Accept*) the job to be received by UGE components if it is clear of certain mistakes.

JSV helps the system administrator to ensure that submitted jobs are accurate and certain environment variables are passed with jobs. If a user fails to submit a job correctly, then the JSV prevents the job from being sent to the UGE master process and instead informs the user of her/his mistakes by displaying a proper message on the output. Users can also benefit from the JSV and its notification messages not only to correct their mistakes or missing parameters, but also learn from their common faults in order to improve their future job submissions.
Univa Grid Engine supports two types of JSV interface: Client JSV and Server JSV. The Client JSV can be defined by the system administrator as well as normal users who can submit jobs to the UGE. After verifying the job submission, each Client JSV process gets terminated in order to eliminate direct overhead on the cluster throughput. Server JSVs can be defined and executed only by system administrators with the purpose of exchanging information with users under a certain condition or logging users' job activities into a file. Since the Server JSV lives as long as sge_qmaster (UGE master process) is running, it may degrade the submission performance and cluster throughput [6]. In this research, we focus on Client JSV in order to verify, correct or reject job submissions on our HPC clusters based upon certain criteria. We will use the JSV and Client JSV interchangeably for the rest of this paper.

3.1 JSV Design for Quanah Cluster

3.1.1 Resources and Policies on QUANAH Cluster

The Quanah cluster is the newest and busiest HPC cluster at Texas Tech University and was commissioned in 2017. Quanah consists of 467 nodes, with a total of 16,812 cores (36 core per node), 87.56 TB total RAM (192 GB per node), and Intel OmniPath high throughput internal network (100 Gbps). UGE manages the resource on Quanah cluster based on three main policy levels: 1) the Parallel Environment (PE) provides a runtime setting for shared-memory (*sm*) and distributed-memory (*mpi*) applications respectively, 2) the (omni) queue maintains instances of jobs and prioritize them in a fair-share fashion, and 3) (quanah), (xlquanah), and (hep) projects apply specific resource policies to the jobs within the 'omni' queue. More details about the PEs and Projects on Quanah cluster are shown in Tables 1 and 2.

Table 1: Parallel Environments on Quanah

PE	Policies
mpi	• Must request for a multiple of 36 cores • All MPI applications must define the 'mpi' PE
sm	• Can request between 1 and 36 cores • Slots are guaranteed to be in one node.

Table 2: Project Policies on Quanah

	quanah	xlquanah	hep
Max # of cores	16,182	144	720
default runtime	48 hours	72 hours	48 hours
Max runtime	48 hours	120 hours	∞
Allowed (PE)s	'sm', 'mpi'	'sm'	'sm', 'mpi'

3.1.2 JSV Design and Reference Implementation

Univa Grid Engine is capable of handling almost all the resource policy definitions and configurations for system administrators. However, several details remain out of the scope of UGE's configuration settings such as: 1) assigning default runtime and PE values to jobs based on different requested Project or Queue;

2) ensuring a soft runtime (s_rt) is not greater than hard run time (h_rt) for each submitted job; 3) confirming distributed-memory (mpi) jobs request for CPU cores in multiples of a total number of cores per node, and shared-memory (sm) jobs do not request for more than total number of cores per node; 4) ensuring the requested amount of memory does not exceed the overall size of the memory across the requested nodes; and 5) notifying users of their job submission rejection with descriptive error messages.

In order to reduce the number of faulty jobs and notify users of their faults, we have designed and implemented a JSV on Quanah cluster in order to verify the job submissions based on 'Resource time', 'Parallel Environment', and 'Memory' requests against each 'Projects' on Quanah.

If a user defines the job submission project as 'quanah', then the JSV checks the requested runtime (h_rt, s_rt) parameters to ensure they are not greater than 48 hours. If so, then the JSV corrects the parameters and sets them to be 48 hours, which is the maximum allowable runtime per job. Since the soft runtime (s_rt) must not exceed the hard runtime (h_rt), the JSV corrects the s_rt by setting the (s_rt = h_rt) if it surpasses the h_rt. The JSV will also assign the default value of 48 hours to those jobs that are missing the runtime parameters during the submission and requesting for 'quanah' project. In case of entering the runtime in any format other than HHH:MM:SS, the JSV will reject the job submission and users will be notified to correct the runtime format accordingly.

If PE parameter does not appear in job submission script (or command-line), the JSV *corrects* the job submission by assigning the default value of 'sm' (share-memory) PE along with one CPU core to the job. The JSV also checks the number of requesting CPU cores against the 'sm' and 'mpi' PE. If the number of requesting CPU cores was not a multiple of 36 (total number of CPU cores per node on Quanah) for 'mpi' jobs or exceeds 36 CPU cores for 'sm' jobs, then the JSV *rejects* the job submission and informs the user with a proper message.

All the compute nodes on Quanah provide 192GB RAM, which means jobs cannot request more than 192GB memory per each machine. Therefore, the JSV on Quanah *rejects* those 'sm' jobs which request for more than 192GB memory and those 'mpi' jobs that request size of memory greater than the total memory size of all to-be-assigned nodes. If a user forgets to define the memory size (*h_vmem* parameter in UGE), the JSV will correct the job submission by setting the default memory size of 5.3GB, which is the amount of memory per CPU core per node.

3.2 JSV Design for Hrothgar Cluster

3.2.1 Resources and Policies on Hrothgar Cluster

The Hrothgar cluster at Texas Tech University consists of three sub-clusters with the following configurations:

- *Hrothgar (West)*: Commissioned in 2011 with 563 nodes, Xeon X5660 Westmere Processors, with a total of 6,756 cores (12 cores/node), 13.19 TB total RAM (24 GB/node), and DDR 20 GB/second Infiniband fabric.
- *Hrothgar (Ivy)*: Commissioned in 2014 consists of 96 nodes, Xeon E5-2670v2 Ivy Bridge Processors, with a total of 1,920 cores (20 cores/node), 6.14TB Total RAM (64 GB/node), and QDR 40 GB/second Infiniband fabric.
- *Hrothgar (Serial):* Uses identical hardware as Hrothgar West except it lacks an Infiniband fabric. These nodes only support serial jobs and do not allow MPI jobs.

UGE on Hrothgar cluster defines two projects: 'Community-cluster' and 'hrothgar'. However, the JSV on this cluster only supports the 'hrothgar' project, and because of that, we do not explain other projects in this section.

In contrast to the resource allocation on the Quanah cluster, which is mainly governed by policy definitions in each project, the Hrothgar cluster handles the resource requests through Queues' policies for each sub-cluster on Hrothgar (i.e. 'west', 'ivy', and 'serial'). In addition to that, each queue defines only one type of PE for each sub-cluster (i.e. 'west' and 'ivy', and 'sm'). Table 3 and 4 explain the PE and Queues' configuration for 'hrothgar' project on Hrothgar cluster.

Table 3: Parallel Environments on Hrothgar

PE	Policies
west	Must request for a multiple of 12 cores
ivy	Must request for a multiple of 20 cores
sm	Can request between 1 to 12 cores.

Table 4: Queue Policies on Hrothgar

Queue Name	west	ivy	serial
Runtime limit	48 hours	48 hours	120 hours
Allowed PE	'west'	'ivy'	'sm'
# cores per node	12	20	12
Max Memory size	24GB	64GB	48GB

3.2.2 JSV Design and Reference Implementation

The JSV on Hrothgar cluster only verifies the jobs without correcting any parameters and notifies users about their mistakes if their jobs get rejected. The JSV on Hrothgar cluster verifies the PE and memory for any available queue when 'hrothgar' project is requested in job submissions. The JSV confirms that if the queue parameter in job submissions requests for either '*west*', '*ivy*', or '*serial*', then the PE parameter should be defined as '*west*', '*ivy*', or '*sm*' corresponding to the selected queue. If not, then the job submission will be *rejected*. For '*west*'

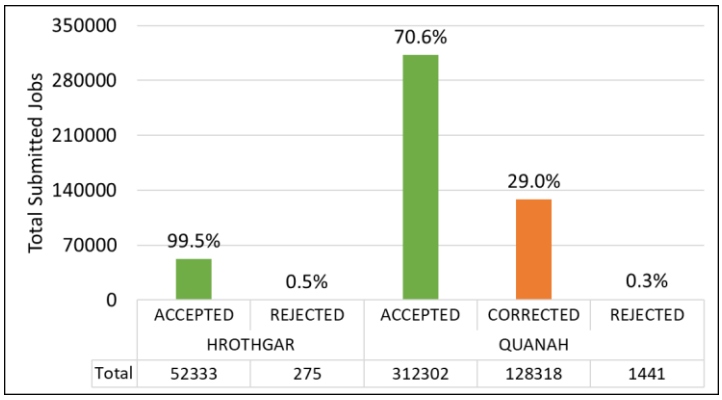

Figure 1: JSV Activities on Hrothgar and Quanah clusters 09/10/2018 – 02/10/2019

and '*ivy*' PE s, the JSV will ensure that the number of requested CPU cores would be a multiple of 12 (for '*west*') or 20 (for '*ivy*') and may not exceed the total of 12 CPU cores for '*sm*' PE per each job submission.

Since the total size of memory per node on '*west*' queue nodes is 24GB, and on '*ivy*' queue nodes is 64GB, the JSV will reject those job submissions which request for these types of queue and a memory size greater than the total memory size of all to-be-assigned nodes. For instance, if a job submission calls for '*hrothgar*' project, '*west*' queue and '*west*' PE along with 24 CPU cores (2 computational node), the maximum allowable memory request for the job is 48GB, and any request for memory size greater than 24G will be *rejected* by the JSV. The JSV also ensures that the memory size will never exceed 48GB (maximum memory size per node on '*serial*' queue) in those job submissions which request for '*serial*' queue and '*sm*' PE.

3.3 Analysis of JSVs

By using Perl as one of the recommended scripting languages for JSVs to communicate efficiently with the UGE interface [6], we have developed two Client JSVs for both Quanah and Hrothgar clusters. Quanah and Hrothgar clusters maintain their own JSV and each JSV runs as a distinct process for every submitted job in order to: 1) verify the accuracy of the job, 2) correct missing or wrong parameters, or 3) accept the job to be sent to the UGE scheduler. JSVs also produce logs for every submitted job including the submission status along with the reason of correcting or rejecting the jobs if any occurs.

Figure 1 depicts the JSV activities on Hrothgar and Quanah clusters for total of 442,061 job submission on Quanah and 52,608 job submissions on Hrothgar from Sept 10th, 2018 to Feb 10th, 2019. As we can observe from this figure, on Quanah 128,318 (29%) submitted jobs were corrected by JSV, 1,441 (0.3%) jobs were blocked from entering the UGE scheduler and all remaining jobs were accepted. Table 5 shows more details about which parameters were corrected for users' job submissions and what reason caused some of the jobs to be rejected. The JSV on Quanah cluster corrected 29% of the job submissions mostly by setting a default memory size and runtime limit. In some cases, users leave the PE undefined, which eventually made the JSV choose the default PE value for their jobs. JSV also enabled the 'Reservation' mode for those jobs which requested for 360 cores or more and did not ask for reservation during submission.

Table 5 also contains the failure reasons of the rejected jobs, including: 1) Non-Existing Project Name: the requested project name does is not a valid name, or the project name is undefined; 2) Non-Existing PE: the requested PE is not valid; 3) Incorrect Number of Requested Slots: the requested number of slots (CPU cores) exceeds 36 cores for 'sm' jobs or is not a multiple of 36 for 'mpi' jobs; 4) Incorrect Run Time Format: the requested runtime is not in the format of 'HHH:MM:SS'; and 5) Memory Size out of Bound: the requested amount of memory for the job is too large and exceeds the total memory size of the to-be-assigned nodes.

In the cases of 1), 2), and 4), UGE will prevent the job from being scheduled even without the JSV. However, the output is not informative enough to make users aware of their faults. In this manner, the JSV is helpful for training users by rejecting their job submissions and providing an informative message. For cases 3) and 5), there is no feature in UGE other than JSV to handle these issues. Without a JSV, those jobs will stay in 'qw' state forever since UGE does not see any problem with those jobs and the scheduler would never find enough resources to allocate them. Therefore, we can infer from Table 5 that the total of 5.6% of the rejected jobs on Quanah (~81 jobs out of 1,441 rejected jobs) are deterministic faulty jobs that were caught by JSV.

The result of JSV's activity on Hrothgar cluster can be observed in Figure 1 as well. Since the JSV on Hrothgar does not correct any job, the data in Figure 1 only represents the results of the 52,333 accepted jobs (99.5%) and 275 rejected jobs (0.5%).

Table 5: Job Submission Status on Quanah 09/10/2018 – 02/10/2019

Submission Status	
ACCEPTED	70.6%
CORRECTED	29.0%
Set default Run Time and Memory Size	66.9%
Set default Memory Size only	30.7%
Set default Run Time, Memory, and PE	1.3%
Set default Run Time, Memory, and Reservati	0.9%
Enable Reservation Only	0.2%
REJECTED	0.3%
Non-Existing Project Name	86.5%
Non-Existing Parallel Environment (PE)	6.1%
Incorrect Number of Requested Slots *	4.7%
Incorrect Run Time Format	1.7%
Memory Size out of Bound *	0.9%
Total	100%

Table 6: Job Submission Status on Hrothgar 09/10/2018 – 02/10/2019

Submission Status	
ACCEPTED	99.5%
REJECTED	0.5%
Incorrect Project Name	24.0%
Non-Existing Queue Name	19.6%
Incorrect Parallel Environment *	19.6%
Undefined Parallel Environment (PE)	10.2%
Incorrect Queue Name *	9.1%
Incorrect Number of Requested Slots	9.1%
Non-Existing Parallel Environment (F	5.1%
Undefined Queue Name	2.2%
Memory Size Out of Bound *	1.1%
Total	100%

Table 6 describes the common reasons that the JSV rejects user jobs on Hrothgar, including: 1) Non-Existing Queue Name: the requested queue name is not a valid queue name; 2) Incorrect PE: the selected PE is valid, but is not being supported by the requested queue; 3) Incorrect Project Name: the project was not defined by the user; 4) Undefined PE: the PE was not defined by the user; 5) Incorrect Queue Name: the selected queue is valid but does not match the requested project; 6) Incorrect number of requested slots: the requested number of slots (CPU cores) exceeds the total number of cores per node for 'sm' jobs, or is not a multiple of 12 for west jobs or 20 for ivy jobs; and 7) Non-Existing PE: the requested PE is not valid.

Similar to Quanah cluster, the rejected jobs in cases of 1), 3), 4), or 7), could be caught by the UGE without JSV. However, UGE's output messages after rejecting a job is not as informative and helpful as our JSV. Moreover, UGE is not capable of recognizing cases of 2), 5), and 6). For instance, if a user requests a valid queue name (e.g. 'west') along with an unmatched PE (e.g. 'ivy'), the UGE will accept the job submission as long as both queue name and PE are valid policy names. UGE will also accept a job submission, in which a valid project name (e.g. 'communitycluster') and a valid queue name (e.g. 'west') are defined, even though the requested queue does not work with the requested project. Furthermore, there is no way in UGE other than JSV to verify the number of requested cores against an acceptable number of cores that can be requested for a particular PE. In all these cases, submitted jobs could be stuck in 'qw' mode forever. As shown in Table 6, we can see that a total of 40% of the rejected jobs on Hrothgar (~110 jobs out of 275 rejected jobs) are deterministic faulty jobs that UGE cannot catch them without JSV.

3.4 Further Analysis of JSVs

Despite the considerably low number of rejected jobs compared to the overall number of accepted jobs on both Quanah and Hrothgar clusters from Sept 10th, 2018 to Feb 10th, 2019, we observed a noticeable improvement in job submissions. This improvement was inferred from the evaluation of users' job submission behavior, and the impact of reducing faulty jobs on system administrators' workload time.

As mentioned earlier, the purpose of a JSV is not only to reduce the 'waiting-forever' faulty jobs, but also to improve user job submission behavior by informing them of their mistakes and correcting missing parameters in their job submission scripts. Figures 2 and 3 outline the percentage of accepted, corrected, and rejected job submissions per month from Sept 10th, 2018 to Feb 10th, 2019. As it is shown in these figures, the total number of submitted jobs to Quanah and Hrothgar clusters in September is relatively low, since our collected data for this month is limited to Sept 10th to Sept 30th, and also users' activities during this month were relatively low. Similarly, our data for February is limited to the first 10 days of this month. We experienced a fair increase in corrected and rejected jobs from September to October since JSVs were helping users understand their mistakes in job submissions. From October to December we observed more accepted job submissions with stable decrease in correcting and rejecting the jobs. However, due to accepting a large number of new users in January (at the beginning of new academic semester), JSVs on both clusters started correcting and rejecting more job submission for new users. Although the total number of job submissions compared to September, a few new users caused a significant increase in corrected and rejected job submissions by submitting many jobs at the same time without paying attention to the JSVs' output messages.

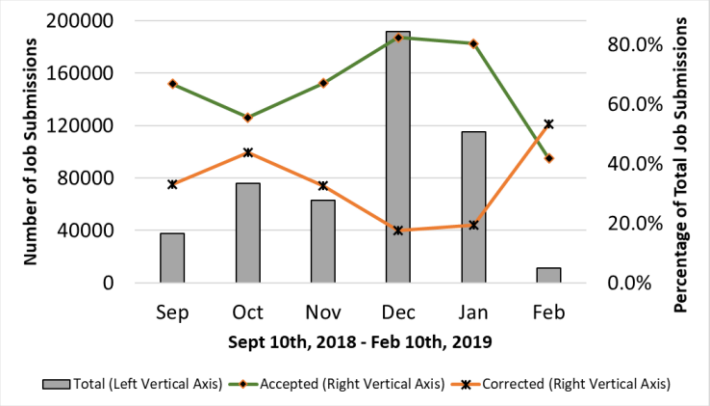

Figure 2: Percentage of accepted and corrected job submissions per month on both Quanah and Hrothgar (09/10/2018 – 02/10/2019)

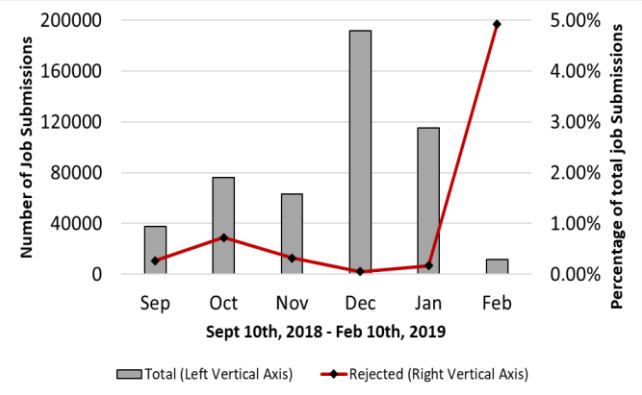

Figure 3: Percentage of rejected job submissions per month on both Quanah and Hrothgar (09/10/2018 – 02/10/2019)

4 Related Work

Failure of job submissions on large-scale systems such as Grid [7] has been studied in several research efforts. Some job failure analysis results show a considerable failure rate of 25% to 33% of all submitted jobs in Grid systems, and 5% to 8% of the job failures occur after beginning execution on compute resources [8]. However, few approaches such as fault-aware scheduling policies and techniques can reduce the job submission failures caused by unexpected resource failure or unavailability [9]. In these techniques, proactive and predictive strategies might be employed to analyze Grid workload traces and discover common patterns of successful or failed jobs [10] or prevent job failure during the execution time [11].

It is also possible to design and develop a tool or plugin as an external component for DRMs and schedulers to manage job submission failures. These tools can be a set of system level script files and end user tools to correct configuration of the user environment and applications [12], or a meta-scheduler that integrates into the source code of Grid brokers (e.g. Condor [13]) to deploy a set of predefined standards and to coordinate users' tasks and resource providers' requirements [14]. For instance, Job Submission Manager (JSM) [15] is a meta-scheduler and can filter the arriving jobs based on parameters such as current system load.

5 Conclusion

Distributed Resource Manager (DRM) systems such as Univa Grid Engine are responsible for managing the distributed resource allocation and policies on HPC clusters. However, in some cases, they fail to give system administrators more advanced control over verifying users' job submissions and blocking potential faulty jobs that may stay in 'qw' mode forever. Therefore, the lack of enough control on job submission confirmation and the absence of a builtin mechanism to automate the detection of faulty jobs and inform users of their errors may lead system administrators to leverage JSVs. Job Submission Verifiers (JSVs) can help system administrators to define extra resource policies, reject potential faulty jobs before entering the scheduler based on specific criteria, and send a message to users' output in order to notify them of their job submission mistakes.

In this paper, we have introduced our research and development efforts in designing, implementing, and analyzing JSVs for two production HPC clusters at Texas Tech University (Quanah and Hrothgar), maintained by the High Performance Computing Center (HPCC). We have described the details about how UGE manages the resource allocation policies on these clusters. We have explained the design and implementation of JSVs for each of these clusters in order to reduce the number of faulty jobs by finding and blocking faulty jobs, correcting some of the missing parameters in users' job submissions, and informing users of their mistakes. We have also analyzed the JSVs' activities and performance results by explaining the data that we have collected from the JSVs' log file. The logs between Sep 10th, 2018 to Feb 10th, 2019 show a significant decrease in rejecting and correcting the job submissions on both clusters before they received a large number of new users. Moreover, 81 rejected job submissions on Quanah cluster and 110 rejected job submissions on Hrothgar cluster were potential 'waiting-forever' faulty jobs, which were caught by our JSV implementation. JSVs were not only successful in rejecting the faulty jobs and correcting some of the user's job submissions' parameters but were also effective in training users on how to prepare their job submission parameters correctly by informing them about their mistakes in a meaningful way.

ACKNOWLEDGMENTS

The authors acknowledge the High Performance Computing Center (HPCC) at Texas Tech University [16] in Lubbock for providing HPC resources that have contributed to the research results reported within this paper. We are also thankful to the anonymous reviewers for their valuable feedback. This research is supported in part by the National Science Foundation under grant CNS-1338078, CCF-1718336, OAC-1835892, CNS-1817094.

REFERENCES

[1] Univa Grid Engine: http://www.univa.com/.

[2] Gentzsch, W. 2001. Sun Grid Engine: towards creating a compute power grid. Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid (2001), 35–36.

[3] Nitzberg, B., Schopf, J.M. and Jones, J.P. 2004. PBS Pro: Grid Computing and Scheduling Attributes. Grid Resource Management: State of the Art and Future Trends. J. Nabrzyski, J.M. Schopf, and J. W\keglarz, eds. Springer US. 183–190.

[4] Yoo, A.B., Jette, M.A. and Grondona, M. 2003. SLURM: Simple Linux Utility for Resource Management. Job Scheduling Strategies for Parallel Processing (Berlin, Heidelberg, 2003), 44–60.

[5] Engineering, U. 2016. Grid Engine Introductory Guide. Univa Corporation.

[6] Engineering, U. 2016. Univa Grid Engine Documentation Univa Grid Engine Administrator ' s Guide. Univa Corporation.

[7] Foster, I., Kesselman, C. and Tuecke, S. 2001. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications. 15, 3 (Aug. 2001), 200–222. DOI:https://doi.org/10.1177/109434200101500302.

[8] Li, H., Groep, D., Wolters, L. and Templon, J. 2006. Job Failure Analysis and Its Implications in a Large-Scale Production Grid. 2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06) (Dec. 2006), 27–27.

[9] Anglano, C., Brevik, J., Canonico, M., Nurmi, D. and Wolski, R. 2006. Fault-aware scheduling for Bag-of-Tasks applications on Desktop Grids. 2006 7th IEEE/ACM International Conference on Grid Computing (2006), 56–63.

[10] Fadishei, H., Saadatfar, H. and Deldari, H. 2009. Job failure prediction in grid environment based on workload characteristics. 2009 14th International CSI Computer Conference (Oct. 2009), 329–334.

[11] Benjamin Khoo, B.T. and Veeravalli, B. 2010. Pro-active failure handling mechanisms for scheduling in grid computing environments. Journal of Parallel and Distributed Computing. 70, 3 (Mar. 2010), 189–200. DOI:https://doi.org/10.1016/j.jpdc.2009.11.003.

[12] Fenglian Xu, Eres, M.H., Baker, D.J. and Cox, S.J. 2004. Tools and support for deploying applications on the grid. IEEE International Conference onServices Computing, 2004. (SCC 2004). Proceedings. 2004 (Oct. 2004), 281–287.

[13] Litzkow, M.J., Livny, M. and Mutka, M.W. Condor-a hunter of idle workstations. [1988] Proceedings. The 8th International Conference on Distributed 104–111.

[14] Colling, D., McGough, A.S., Ma, T., Novov, V., Smith, J.M., Wallom, D. and Xiong, X. 2010. Adding standards based job submission to a commodity Grid broker. Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010. Cit (2010), 1530–1535. DOI:https://doi.org/10.1109/CIT.2010.272.

[15] Saadatfar, H. and Deldari, H. 2014. A job submission manager for large-scale distributed systems based on job futurity predictor. International Journal of Grid and Utility Computing. 5, 1 (2014), 50. DOI:https://doi.org/10.1504/IJGUC.2014.058252.

[16] High Performance Computing Center (HPCC) at Texas Tech University: http://www.depts.ttu.edu/hpcc/.