Noname manuscript No.
(will be inserted by the editor)

A Case Study of a Shared/Buy-in Computing Ecosystem

Christopher Liao and Yonatan Klausner, David Starobinski, Eran Simhon, and Azer Bestavros

the date of receipt and acceptance should be inserted later

Abstract Many research institutions are deploying computing clusters based on a shared/buy-in paradigm. Such clusters combine shared computers, which are free to be used by all users, and buy-in computers, which are computers purchased by users for semi-exclusive use. The purpose of this paper is to characterize the typical behavior and performance of a shared/buy-in computing cluster, using data traces from the Shared Computing Cluster (SCC) at Boston University that runs under this paradigm as a case study. Among our main findings, we show that the semi-exclusive policy, which allows any SCC user to use idle buy-in resources for a limited time, increases the utilization of buy-in resources by 17.4%, thus significantly improving the performance of the system as a whole. We find that jobs allowed to run on idle buy-in resources arrive more frequently and run for a shorter time than other jobs. Finally, we identify the run time limit (i.e., the maximum time during which a job is allowed to use resources) and the type of parallel environment as two factors that have a significant impact on the different performance experienced by shared and buy-in jobs.

1 Introduction

The *Shared Computing Cluster (SCC)* at Boston University (BU) [21] implements a *shared/buy-in* system. Shared users access the cluster for free, while buy-in users pay a fee for priority usage of a set of resources. In addition, the SCC allows any user to utilize buy-in resources when they are idle (also known as *public queues*), which greatly increases the system's utilization. BU's adoption of the shared/buy-in architecture is an instance of an increasingly popular two-tiered service model in research computing clusters. Other instances of shared/buy-in systems include University of Wisconsin-Madison's High Performance Computing (HPC) Cluster [15] and University of Arizona's HPC Cluster [19]. As the purpose of the SCC is similar to other university computing clusters in that they support research in a large university environment, we use the SCC as a representative case study to gain insight into the benefits of its shared/buy-in architecture in large research university computing clusters.

The object of this paper is to better understand the behavior of shared/buy-in ecosystems, through measurements performed on the SCC. Specifically, we aim (i) to investigate the differences in job characteristics between shared and buy-in resources, (ii) to determine how public queues benefit the system, and (iii) to quantify the delay that shared users are willing to tolerate. Our study combines analysis of exogenous parameters, such as the running time and interarrival time, together with analysis of endogenous parameters, such as the waiting time and utilization.

Toward goals (i) and (ii), we compare the utilization, demand and workload characteristics of shared and buy-in resources. Through this analysis, we show that implementation of public queues benefits shared/buy-in systems by significantly improving the utilization of buy-in resources. This aspect improves the Quality of Service (QoS) in general, since it makes more resources available to all users. We define the QoS as the degree of satisfaction for users of the SCC as measured by waiting time. We also show that, compared to shared and buy-in jobs, jobs scheduled to public queues tend to arrive more frequently and run for shorter time.

Boston University, Boston, MA
E-mail: cliao25@ bu.edu, yklausner@gmail.com, staro@bu.edu, simhon@bu.edu, best@bu.edu

Toward goal (iii), we examine in detail how the run time limit and the parallel environment affect the waiting time for various jobs. Our analytical methodology provides insight into the QoS experienced by users in shared/buy-in clusters and helps in identifying policy changes that could lead to better waiting time performance. The main contribution of this study is a comprehensive comparison of workload characteristics between the shared and buy-in parts of the SCC, which should inform future computing clusters considering a similar policy.

In the following, we first discuss previous workload characterization studies involving other computing clusters. We next provide an overview of the SCC's architecture, including both physical and logical aspects. Then, we present a statistical characterization of the SCC. This section begins by characterizing exogenous parameters and concludes with a performance evaluation of the SCC based on utilization and waiting time. We end with concluding remarks and suggestions for policy changes that might benefit shared/buy-in systems.

2 Background and Related Work

Before discussing related work, we introduce general terminology that applies to the SCC:

- The number of *slots* is the number of Central Processing Units (CPU) required for a job to run.
- A *node* is an independent computer with multiple slots.
- A *waiting list* is the list of jobs waiting for execution at any point in time.
- A *queue* is a logical abstraction that aggregates a set of slots across one or more nodes [16]. A queue has certain configurations that define which jobs it accepts. A queue is different from a waiting list.
- A *project* is a group consisting of one or more users.
- The *run time limit (RTL)* is the maximum amount of time a job can run before it is killed.

We will further elaborate on these terms throughout the paper.

Our work relates to the workload characterization of computing clusters [1]. Such clusters are generally grouped into two main categories: *grids*, heterogeneous clusters generally used in scientific and academic settings, and *clouds*, large homogeneous clusters generally used in commercial settings [5]. The Grid Workloads Archive (GWA) [20] and Parallel Workloads Archive (PWA) [7] contain workload traces of grid clusters. In relation to other clusters, the SCC should be viewed as a grid-like computing cluster.

The study in [9] shows that usage of real workload traces in grid computing research has become increasingly popular in the past decade and summarizes the workload traces publicly available via the GWA and the PWA. Works that use data traces from the GWA and PWA generally have one of three focuses: exogenous parameter modeling, performance evaluation, or scheduling optimization. Only the first two focuses are relevant to our study.

Parameter modeling studies include [9,8,14]. These studies fit distributions to essential parameters such as interarrival time and running time of traces taken from the GWA and PWA with the goal of creating a synthetic workload trace and improving performance by optimizing system policy. Performance evaluation studies include [11,13]. Specifically, [11] plots the relationship between performance metrics and the load of the system using heat maps, while [13] attempts to predict the waiting time by using stochastic processes and machine learning to identify "quick starters." In our study, we combine the above two focuses.

Our study is one of the first to focus specifically on a shared/buy-in system, though there are a few studies that include similar multi-tiered concepts. As a practical example, the Linux cluster at Lawrence Livermore National Lab (LLNL) is divided into two sub-clusters: Atlas, for large parallel computing tasks, and Thunder, for more frequent small jobs [7]. This two-tiered architecture, although different from the SCC in policy has similarities as will be briefly explored in Section 4.7. Refs. [12] and [23] conduct theoretical studies that deal with multi-tiered service concepts. Ref. [12] suggests and analyzes a service-oriented architecture approach to HPC clusters. Ref. [23] proposes dividing resources into two groups, one that handles peak workload and one that handles regular workload. While these two studies are theoretical investigations of multi-tiered service architectures, our study is a statistical analysis of the performance of shared/buy-in architecture.

While the scheduler by itself is not the focus of this paper, scheduling software does have a direct effect on workload characteristics. Much of the recent research in scheduling algorithms focus on the capability to efficiently schedule highly parallel jobs along smaller jobs. Ref. [2] gives a survey of state-of-the-art scheduling algorithms with parallel-job support, with a focus on those designed around task graphs. The authors in [2] assert high autonomy, scalability, and heterogeneity as benchmarks for a good scheduler. Ref. [6] proposes *SEParAT*, an efficient scheduler for mixed parallel workloads. Ref. [17] compares four popular schedulers, including *Sun Grid Engine (SGE)*, an older version of *Open Grid Scheduler*, the scheduler used by the

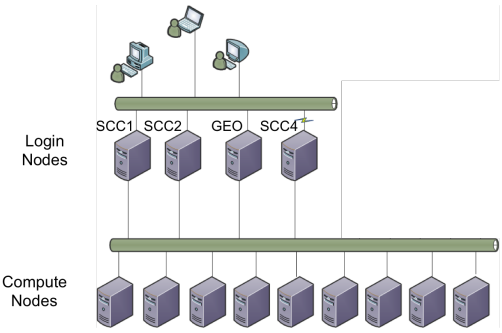

Fig. 1 SCC Architecture. There are 4 login nodes and 467 compute nodes. [21]

SCC (see Section 3.2). Multiple performance parameters are investigated, and the authors found that SGE performed well when given parallel workloads and small jobs. A brief analysis of the parallel workload in the SCC is given in Section 4.4.2.

An abbreviated version of this paper is published in the 2016 IEEE MIT Undergraduate Research Technology Conference Proceedings [10].

3 Cluster Architecture

A user logs on to the SCC by accessing one of the 4 login nodes through a secure shell. *Login nodes* are used for transferring files to and from the SCC, program compilation, text editing, and light debugging. These nodes are not included in our analysis. Instead, we analyze the 467 compute nodes. *Compute nodes* allow users to submit computationally intensive jobs that generally run for a longer amount of time and require more resources. Fig. 1 shows the relationship between the login nodes, compute nodes, and file storage in the SCC.

3.1 System Overview

As mentioned in Section 1, the SCC at BU implements a shared/buy-in system. In order to become a buy-in project, a project owner purchases new nodes for the SCC in exchange for priority access to these nodes. Buy-in projects benefit because they experience shorter waiting times on their own nodes as seen in Section 4.6. Additionally, all projects benefit since buy-in resources are reclaimed when they are not utilized by their owners as is explained next.

Compute nodes are divided into two categories: shared and buy-in. Any job can run on the shared nodes. Any job can run on a buy-in node as long as no users of that buy-in project are running a job on that node. Since buy-in projects receive priority for their own resources, shared users can only run jobs on buy-in nodes if the run time limit is 12 hours or less without exception. Buy-in users are able to run jobs on the shared nodes like shared users.

Table 1 General overview of SCC from 1/1/15 to 7/10/15. The statistics on number of jobs and unique users are totaled over the entire period. The statistics on nodes and cores refer to the end of the period.

	System	Shared	Buy-in	Public
Total Jobs	17,420,081	6,477,145	3,352,421	7,590,515
Unique Users	1,055	995	470	549
Nodes	471	169	298	
Cores (CPUs)	6,312	2,536	3,776	

Table 2 List and description of variables for Section 3.2

Notation	Description
p_i	Priority of a job.
f_i	Number of tickets given to job.
p_t	Adjusted number of tickets.
u_i	Function of recent usage.
p_u	Urgency factor.
p_p	User Assigned Priority.
n	Normalization function.

In general, each buy-in node has two queues: a buy-in queue and a public queue. The *buy-in queue* only accepts jobs from a user of that specific buy-in project. The *public queue* accepts jobs from any user as long as the run time limit is 12 hours or less. The public queue for a node is automatically disabled when a buy-in user of that specific node submits a job on the SCC for that project. When disabled, the queue stops accepting jobs from users outside of the buy-in project. Any job already running when a buy-in user submits a job is completed before the buy-in user receives access to his/her resources. This policy prevents buy-in users from waiting for a long time for their own resources, but allows their resources to be utilized when idle. The benefits of this aspect is further discussed in Section 4.5.2.

Upon submitting a job, users are able to specify certain attributes for their jobs such as the run time limit, the number of slots, and the parallel environment. In Section 4.6, we investigate how some of these attributes affect the waiting time. Table 1 gives a general overview of the SCC. Note that buy-in and public jobs share the same physical resources.

3.2 Scheduler

The SCC operates using the *Open Grid Scheduler*, an open-source scheduling software [16]. A scheduling round occurs every 15 seconds by default, but is also triggered

by events such as job submissions and job completions. This means that priority is recalculated at least every 15 seconds. During each scheduling round, the priority of waiting jobs may change slightly, as jobs entering and leaving the waiting list have an effect on the number of tickets and the normalization functions discussed below. During every scheduling round n, where $n = 1, 2, 3, ...N$ and N is the total number of scheduling rounds, (for simplicity, the index is dropped for the remainder of this section) the following occurs:

- Priority (as explained below) is assigned to each waiting job and all waiting jobs are sorted accordingly. Let the set of jobs in the waiting list be denoted by $i = 1, 2, 3, ...m$, where m is the total number of jobs waiting. The priority of job i is denoted by p_i.
- Available resources are allocated to jobs with the highest priorities.

Three factors are used to calculate the priority p_i of job i: p_t^i, p_u^i, and p_p^i, as explained below:

1. The number of tickets p_t^i is the most complicated part of the final priority assigned to a job. The scheduler starts with 10^4 tickets and distributes tickets to the jobs in the waiting list using a share tree policy. The share tree policy first distributes tickets to projects based on proportions specified by the administrator. The tickets of each project are then distributed equally to its users. The goal of the share tree policy is to ensure that all users and projects on the SCC receive their respective target share of resources over time. Let f_i be the number of tickets a job is given, then the adjusted number of tickets p_t^i is approximated by:

$$p_t^i = \frac{f_i}{u_i}, \tag{1}$$

   where u_i is a function of the recent usage of the user submitting the jobs and the recent usage of the project. This leads to the following:
   - If the user's recent usage has been high, the job will have a lower priority than a job submitted by a user whose recent usage has been low.
   - If the project's recent usage has been high, the project's job will have a lower priority than a job submitted by a project that has a low recent usage.
2. The urgency factor p_u^i is equal to the number of requested slots. This factor causes jobs that request more slots to have a slightly higher priority. Since larger parallel jobs are harder to schedule, putting them at the front of the waiting list prevents them from being starved of resources.
3. The user assigned priority p_p^i is the priority given to job i by the user. If the user submits multiple jobs at once, the user may want a certain job to be completed before another. A user is only able to decrease a job's priority. This feature is rarely used.

A normalization function is used to ensure that all three parts of the priority are on the same scale. The normalization function $n(p, S)$ is defined in terms of the value to be normalized p and the set S to which p belongs:

$$n(p, S) = \frac{\max(S) - p}{\max(S) - \min(S)}, \qquad \forall p \in S, \tag{2}$$

The following equation is used to evaluate the priority of a job on the SCC:

$$p_i = 0.2 \times n(p_p^i, S_p) + 0.02 \times n(p_u^i, S_u) + n(p_t^i, S_t), \tag{3a}$$

where

$$S_p = \{p_p^1, p_p^2, p_p^3 ... p_p^m\}, \tag{3b}$$
$$S_u = \{p_u^1, p_u^2, p_u^3 ... p_p^m\}, \tag{3c}$$
$$S_t = \{p_t^1, p_t^2, p_t^3 ... p_t^m\}. \tag{3d}$$

After sorting by priority, the scheduler assigns each job to an available queue. Each queue is configured with a list of parallel environments that it accepts. Other user requests such as the run time limit, CPU architecture, and available memory also directly affect which queues can accept a particular job. In the case where multiple queues can accept a job, the job is assigned to a queue using an algorithm which takes into account the load and the sequence number of a queue.

4 Performance Analysis

4.1 Method

The SCC has been completely operational since July 2013. We analyze a 4.2 GB data trace collected from January 1, 2015 through July 20, 2016 using R and Python. This workload trace represents jobs from a variety of departments across all of BU, including Medicine, Engineering, Physics, Biology and Chemistry. Therefore, the data trace used should sufficiently generalize the workload of a large research university.

Table 3 Exogenous Parameter Comparison. Demand refers to typical request only.

	System			Shared			Buy-in			Public		
	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD
Interarrival time (s)	2.81	0.64	17.32	7.55	1.31	50.00	14.59	3.25	62.98	6.45	0.92	61.89
Waiting time (hr)	5.32	0.87	18.03	6.21	1.21	20.90	4.99	0.56	18.27	4.71	0.76	15.00
Running time (hr)	0.95	0.03	8.68	1.51	0.06	11.96	1.13	0.02	10.48	0.40	0.02	1.23
Slots	2.52	1.00	4.99	2.20	1.00	5.58	3.25	1.00	5.67	2.46	1.00	4.02
Workload (slot-hr)	2.64	0.05	50.54	3.75	0.09	54.67	4.24	0.04	85.93	0.99	0.03	6.72
Demand of Typical Request (thousand requests/ 12 hrs)	11.11	5.64	17.64	3.64	1.40	6.82	1.75	0.46	5.32	6.30	2.97	11.70

Note that the SCC supports submission of *array jobs*, a group of individual tasks that require identical resources. While the series of tasks are assigned one job number, they are scheduled separately. Therefore, in our analysis, these individual tasks are considered as separate jobs. Consequently, the workload characterization which follows exhibits shorter interarrival times and shorter running times than it would if array jobs were considered a single job. Thus, appropriate considerations have to be made when comparing the analysis presented below with other workload characterization studies.

In our analysis, *shared jobs* are defined as jobs that run on a shared node excluding jobs that run on a public queue. *Buy-in jobs* are defined as jobs that users of buy-in projects run on their own buy-in nodes, and *public jobs* are defined as jobs that run on public queues on buy-in resources.

In the sections that follow, we perform a comprehensive statistical analysis of the SCC, starting with exogenous characteristics and concluding with endogenous characteristics. Table 3 summarizes the statistics on exogenous parameters.

In addition to comparing the distributions of the parameters between shared and buy-in jobs, we also present the distribution fittings for these parameters. We fit the empirical data with common distributions and use the distribution that best fits the empirical data to formulate a random variable. This random variable is helpful to:

- Inform theoretical models with parameters taken from an actual system.
- Predict future usage.
- Experiment with new policies without changing the policy in the real system.

Four distributions were considered: Weibull, log-normal, gamma, and exponential. A skewness-kurtosis plot determined that Weibull and log-normal distributions best fit each of the parameter distributions given in the following sections. Skewness and kurtosis together give a quantitative description of the shape of a distribution. The studies in [3,22] both considered the Weibull distribution when modeling similar parameters such as interarrival time when investigating Bag of Tasks (BoT) workloads. Ref. [22] also considers the log-normal, exponential and gamma distributions. In our study, the distribution fitting process is carried out with the `fitdistrplus` package [4]. Then, we use the one-sample Kolmogorov-Smirnov test (the test used in [22]), a test which uses the vertical distance between two CDFs to determine goodness-of-fit, to compare the three distribution fits for each parameter (see `ks.test` function in R [18]). Only the distribution with the best fit is given in this paper.

4.2 Arrival Pattern

The *interarrival time* is the time elapsing between job submissions. This parameter is calculated by counting the number of job arrivals a_n during each hour $n = 1, 2, 3, ..., N$, where N is the total number of hours in the trace analyzed. The interarrival time during each hour n is then given by $t_n = 3.6 \times 10^3 / a_n$. Table 3 shows that the mean interarrival time is 7.55 seconds for shared jobs, 14.59 seconds for buy-in jobs, and 6.45 seconds for public jobs. Public jobs arrive most frequently, while buy-in jobs arrive least frequently. We present the interarrival times separately for each of the three types of jobs to highlight the difference in workload characteristics. However, it is not appropriate to describe the workload with three separate arrival processes, as the three groups of jobs overlap in physical resources and user populations.

The interarrival time T is best modeled by a log-normal distribution, i.e.

$$P(T \le t) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}[\frac{\ln t - \mu}{\sqrt{2}\sigma}], \qquad \sigma = 1.30, \mu = -0.22, \tag{4a}$$

where

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{e}^{-\tau^2} \, \mathrm{d}\tau. \tag{4b}$$

Table 4 Distribution of Jobs over Parallel Environments by Quantity and Workload

	Number of Jobs				Workload			
	System	Shared	Buy-in	Public	System	Shared	Buy-in	Public
Single-slot	84.5%	87.0%	76.9%	85.7%	31.0%	35.5%	21.5%	34.3%
OMP	15.2%	12.5%	22.6%	14.3%	45.0%	37.3%	47.2%	65.8%
MPI	0.28%	0.48%	0.52%	0.010%	24.0%	27.1%	31.4%	0.85%

The demand is directly related to the interarrival time, as both are measures of how frequently jobs are submitted. The demand is measured by counting the number of requests per 12 hours. Here, we examine the demand for typical requests, that is, single-slot jobs with a run time limit of 12 hours or less. This type of request accounts for 79% of all jobs, so considering the demand of these jobs independently from the rest of the trace is a key component to a complete workload characterization. Table 3 gives the mean, median and standard deviation of the demand of the typical request. These statistics show that the average demand of the typical request is 3.64×10^3, 1.75×10^3, and 6.30×10^3 requests per 12 hours for shared, buy-in and public jobs respectively. Consistent with interarrival time, these observations show that public jobs arrive more frequently than shared jobs, which in turn arrive more frequently than buy-in jobs.

We model the demand of the typical request with a random variable Z defined in terms of the Weibull distribution, i.e.

$$P(Z \leq z) = 1 - e^{(-z/\lambda)^k}, \qquad \lambda = 9.0 \times 10^3, k = 0.73. \tag{5}$$

While this demand model accounts for 79% of the requests, it only accounts for 14% of the workload. It is indeed difficult to capture a large portion of the workload with few request types, as detailed in Section 4.4.

4.3 Running Time

The *running time* is the time elapsing between the start time and the end time of a job. The default run time limit is 12 hours, and the maximum run time limit is 30 days. Table 3 shows that the average running times are 1.51, 1.13, and 0.40 hours for shared, buy-in and public jobs, respectively. Therefore, while on average shared jobs only run 22.8 minutes longer than buy-in jobs, public jobs are on average 64.6% shorter than buy-in jobs and 73.5% shorter than shared jobs. The running time distribution of public jobs also has the lightest tail, with a standard deviation of 1.23 hours. This special characteristic is a result of the 12 hour run time limit policy for all public queue jobs as explained in section 3.1. We also note that 97.7% of shared jobs and 98.3% of buy-in jobs run for under 12 hours, while only 72.8% of shared jobs and 75.7% for buy-in jobs have a run time limit of 12 hours or under. This means that users tend to overestimate their resource needs by requesting a run time limit longer than the actual running time of their jobs.

The running time R is best modeled by a log-normal distribution , i.e.

$$P(R \leq r) = \frac{1}{2} + \frac{1}{2} erf[\frac{\ln r - \mu}{\sqrt{2}\sigma}], \qquad \sigma = 2.60, \mu = 5.04. \tag{6}$$

4.4 Workload

We define the *workload* as the product of the number of slots used by a job and the job running time (in hours). The workload distribution has the heaviest tail of all distributions examined in this paper, i.e. the standard deviation is the largest when compared to the mean, as seen in Table 3.

4.4.1 Slots

Table 4 shows that 87.0% of shared jobs, 76.9% of buy-in jobs, and 85.7% of public jobs use a single slot. Table 3 shows that shared, buy-in and public jobs use averages of 2.20, 3.25, and 2.46 slots, respectively. It is clear from these numbers that a greater percentage of buy-in jobs are parallel jobs. There are two types of parallel jobs in the SCC: *Open Multi-Processing (OMP)* run on one node while *Message Passing Interface (MPI)* jobs run on multiple nodes. Table 4 shows that 98.5% of parallel jobs are OMP jobs.

Table 5 Utilization Analysis of SCC

	Shared	Buy-in	Public
Actual Workload (slot-hr)	24,314,786	14,207,485	7,512,121
Workload Capacity (slot-hr)	34,509,888	43,184,064	43,184,064
Utilization (%)	70.46	32.90	17.40

4.4.2 Parallel Environment

Table 4 shows that OMP jobs account for 45% of the workload of the SCC which is more than MPI or single slot jobs. Although there is no explicit connection between the number of slots and the parallel environment, 16 slots can be considered the boundary case between OMP and MPI jobs. i.e. 99.0% of MPI jobs use 16 slots or more and 99.97% of OMP jobs use 16 slots or less. Table 4 shows that MPI jobs account for 0.48% of shared jobs, 0.52% of buy-in jobs, and 0.01% of public jobs. However, they account for 27.1% of the shared workload, 31.4% of the buy-in workload, and 0.08% of the public workload. These statistics show that while it is tempting to ignore MPI jobs when creating a model of the SCC, they still constitute 20% of the workload.

4.4.3 Shared/Buy-in Comparison

Table 3 shows that the difference between the workload distributions of shared jobs and buy-in jobs is small, with means of 3.75 and 4.24 hours for shared and buy-in jobs, respectively and standard deviations of 54.67 and 85.93 for shared and buy-in, respectively. A more dramatic difference in workload distribution is observed in the distribution of public jobs. The standard deviation of public workload is 6.72 slot-hours, only 13.3% of the system workload deviation; the mean is 0.99 slots-hours, 37.5% of the system mean. According to these statistics, the workload of public jobs, similar to the running time has a much lighter tail than other jobs. This observation in addition to the difference in the other exogenous parameters, reveal that in general, public jobs are submitted more frequently, but take up less temporal and physical resources.

4.5 Utilization

In this section, we analyze the utilization of different parts of the system and the distribution of workload among parts of the system. The *workload capacity* is the workload achieved if all nodes are completely utilized, and the *actual workload* is the sum of the workload of all jobs that actually ran. The *utilization* is the fraction of the workload capacity taken up by the actual workload over a period of time.

4.5.1 System View

Table 5 compares the workload capacity of different types of resources with their actual workloads. This table shows that the workload capacity of shared nodes over the period is 79.9% of buy-in nodes, while their actual workload is 111.9% of buy-in nodes. Thus, workload is distributed more heavily to shared nodes than to buy-in nodes.

4.5.2 Public Queue

Fig. 2 shows the monthly mean utilization trend. It clearly illustrates the benefits of implementing public queues on buy-in nodes. According to Table 5, without public queues, buy-in nodes would only have an average utilization of 32.9%. Public queues reclaim 17.4% of the workload capacity of buy-in nodes, resulting in an average utilization of 50.3% on buy-in nodes. By utilizing buy-in nodes when they are idle, public queues prevent demand for shared resources from exceeding their capacity and balance out the workload more evenly between shared and buy-in nodes.

4.5.3 Pattern

Fig. 3 shows the weekly pattern of utilization. This graph is obtained by splitting the 2015 - 2016 utilization data into a recurring weekly pattern. The pattern is achieved by using the Seasonal Decomposition algorithm in R. This algorithm derives a weekly pattern by removing high frequency and low frequency fluctuations in the utilization time series (see `stl` function in R [18]). It shows that the system's utilization is lowest over the weekend and that the daily utilization peaks in the late afternoon. Shared and buy-in utilization exhibit the same weekly pattern.

Table 6 Factors Affecting Waiting time (hours)

	Shared			Buy-in			Public		
	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD
Overall	6.21	1.21	20.90	4.99	0.56	18.27	4.71	0.76	15.00
OMP	6.53	0.74	18.90	5.28	0.12	21.81	14.80	1.93	33.70
MPI	15.62	0.58	53.70	20.33	0.02	72.85	—	—	—
Single-slot	6.12	1.28	20.83	4.80	0.79	16.03	3.02	0.68	7.27
RTL >12 hrs	10.60	1.71	36.18	4.67	0.60	18.07	—	—	—
RTL $\leq$ 12 hrs	4.58	1.08	10.07	4.40	0.73	13.94	4.71	0.76	15.00

Table 7 Percent of Jobs in the Tail of Waiting Time Distribution

	Shared		Buy-in		Public	
	>12 hr	>1 week	>12 hr	>1 week	>12 hr	>1 week
Overall	12.0%	0.30%	8.8%	0.27%	8.1%	0.058%
OMP	12.2%	0.13%	8.2%	0.34%	23.5%	0.40%
MPI	22.4%	1.9%	18.1%	3.5%	—	—
Single-slot	11.9%	0.31%	8.9%	0.22%	5.6%	0%
RTL >12 hrs	16.6%	1.1%	10.8%	0.67%	—	—
RTL $\leq$ 12 hrs	10.3%	0.003%	8.2%	0.14%	8.1%	0.058%

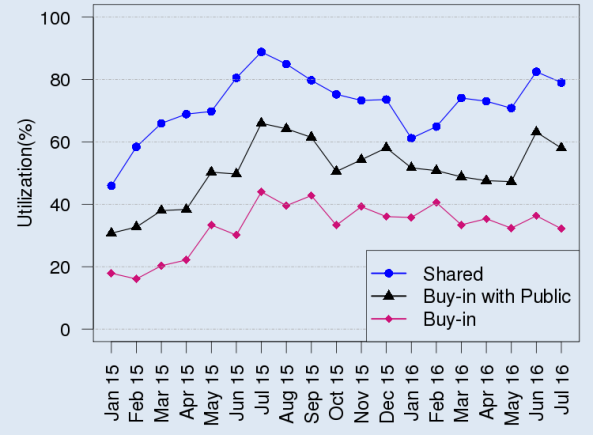

Fig. 2 The average monthly utilization trend comparison between shared and buy-in resources.

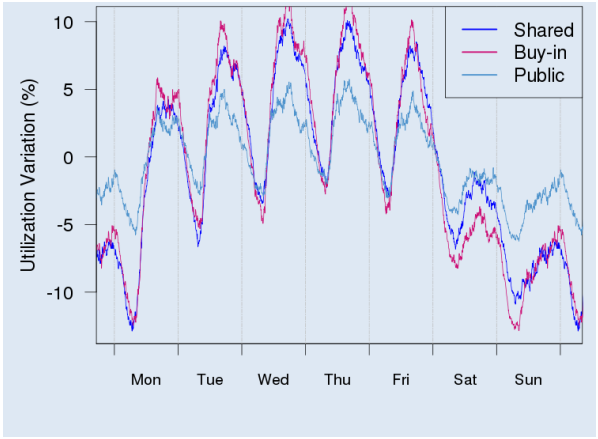

Fig. 3 Weekly utilization pattern.

4.6 Waiting Time

The waiting time of a job is the time elapsing from its submission till its start. Table 6 shows that shared jobs tend to wait more than buy-in jobs, as the median waiting time for shared jobs is 73 minutes, while the median waiting time for buy-in jobs is 34 minutes. Additionally, Table 7 shows that for shared jobs, 12.0% of jobs wait more than 12 hours and 3.0% of jobs wait more than one week, while for buy-in jobs, 8.8% of jobs wait more than 12 hours and 2.7% of jobs wait more than one week. Thus, the waiting time distribution of shared jobs has a heavier tail than that of buy-in jobs.

The mean waiting time for public jobs is close to the mean waiting time for buy-in jobs, but its standard deviation is shorter, at 15 hours, compared to 20.9 and 18.3 hours for shared and buy-in jobs, respectively. Table 8 shows that 8.1% of public jobs wait more than 12 hours and 0.058% wait more than a week. The waiting time is mainly affected by the run time limit and the parallel environment as discussed below. The waiting time W is best modeled with a Weibull distribution, i.e.

$$P(W \leq w) = 1 - e^{(-w/\lambda)^k}, \qquad \lambda = 6.2 \times 10^3, k = 0.42. \tag{7}$$

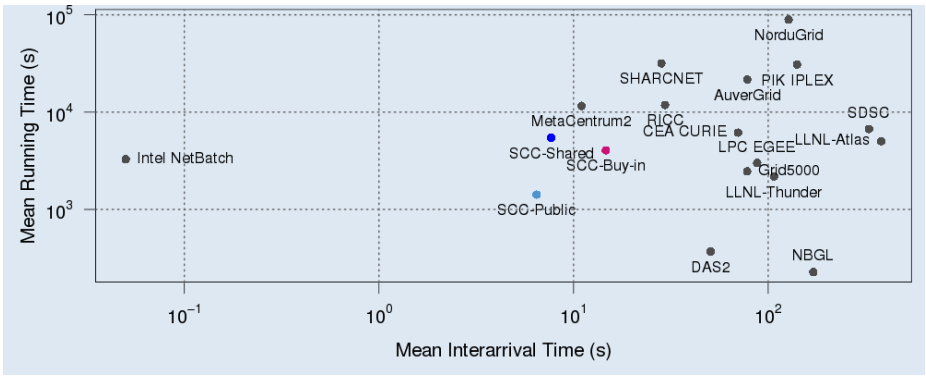

Fig. 4 Comparison of interarrival time and running time averages among three sections of the SCC and 15 other grids from the PWA [7] and GWA [20].

4.6.1 Run Time Limit (RTL)

As mentioned in Section 3.1, users set the RTL when submitting a job. Table 6 shows that when the RTL of a buy-in job exceeds 12 hours, this has negligible effect on the median and mean waiting time but increases the standard deviation from 13.9 to 18.1 hours. However, if the RTL of a shared job exceeds 12 hours, then the median waiting time increases from 1.1 to 1.7 hours, the mean from 4.6 to 10.6 hours, and the standard deviation from 10.1 to 36.2 hours. Hence the difference in performance between shared and buy-in jobs is more significant once the RTL exceeds 12 hours.

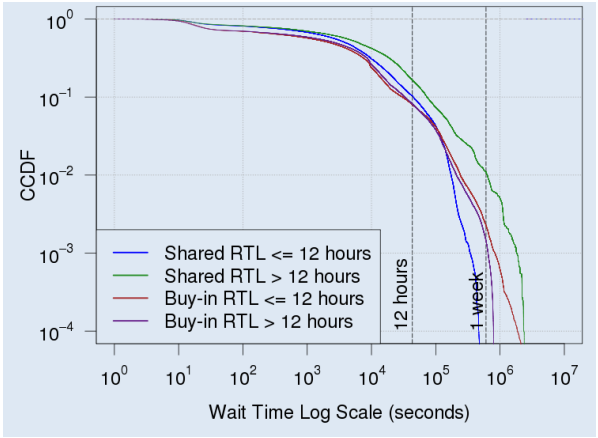

Fig. 5 Comparison of waiting time distribution between jobs that request a run time limit (RTL) of more than 12 hours and jobs that request a run time limit (RTL) of less than 12 hours.

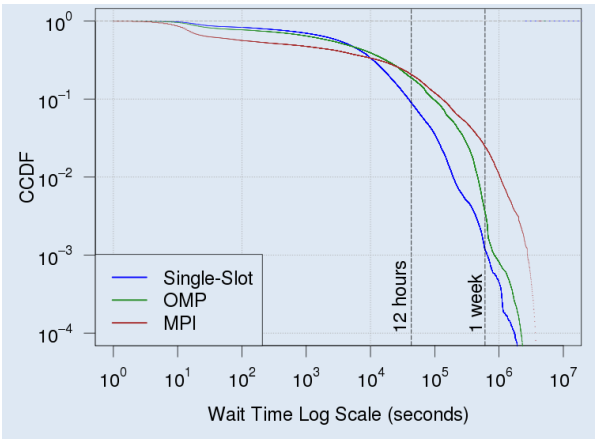

Fig. 6 Waiting time distribution for OMP, MPI, and single-slot requests.

Table 7 shows that for shared jobs, requesting an RTL greater than 12 hours gives a 16.6% chance of waiting more than 12 hours and a 1.1% chance of waiting more than one week, while jobs with RTL of 12 hours or less have a 10.3% chance of waiting more than 12 hours and a 0.003% chance of waiting more than one week. Fig. 4 illustrates this large difference in the tails of the waiting time distributions, which is most likely because more resources are available to jobs with RTL less than or equal to 12 hours via the public queues, resulting in shorter waiting times (recall that jobs with less than 12 hours RTL can be scheduled to shared resources or public resources).

4.6.2 Parallel Environment

Table 6 shows that the difference in the median waiting time among single-slot, OMP, and MPI jobs is within one hour for both shared and buy-in jobs. However, the waiting time standard deviation of single-slot jobs is

20.8 hours for shared jobs and 16.0 hours for buy-in jobs, while the standard deviation of MPI jobs is 53.7 hours for shared jobs and 72.9 hours for buy-in jobs. Hence, submitting an MPI job increases the probability of waiting for a long time, especially for buy-in jobs. This may cause some users to submit a longer single-slot job, which should experience a shorter waiting time, rather than a shorter MPI job, which is prone to longer waiting times.

Table 6 and 7 also show a drastic difference in waiting time performance between single-slot and OMP jobs running on public queues. Public single-slot jobs have an average wait time of 3.02 hours, half that of shared single-slot jobs. On the contrary, public OMP jobs wait an average of 14.8 hours with a standard deviation of 33.7 hours, 80% more than the waiting time standard deviation of shared OMP jobs. Public OMP jobs also have a 23.5% chance of waiting more than 12 hours, a higher chance than other types of jobs listed in Table 7. The difference in the tail of the waiting time distributions is illustrated in Fig. 5.

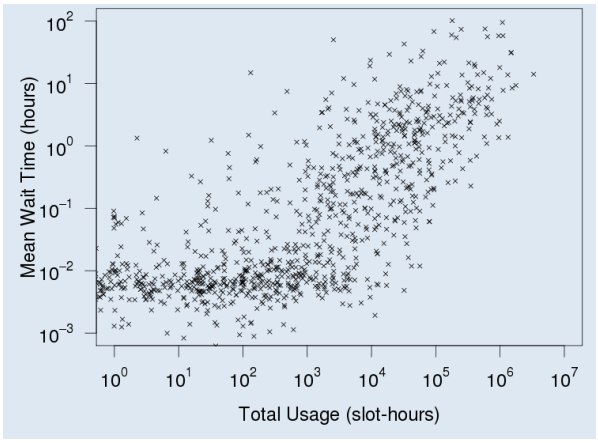

Fig. 7 The mean waiting time of a user's job vs. his/her total usage of the SCC.

4.6.3 User Usage

Fig. 6 shows a positive correlation between a user's mean waiting time and the user's total usage of the SCC. The correlation coefficient is 0.39. We hypothesize that this correlation is due to the priority assigned by the scheduler. As alluded in Section 3.2, the priority of a job is lowered if the user's recent usage is high, resulting in a longer waiting time. Therefore, if a user uses the SCC more frequently, his or her average waiting time would increase, which would also be an incentive to become a buy-in user.

4.6.4 Limitations

We conclude this section by noting the following experimental limitations that may skew the waiting time data trace:

- If a user submits multiple jobs at once, the user may wait for his/her own jobs due to the 512 slot limit for users on shared resources.
- Users have the option of holding their own jobs after submitting them. There is no way to extract the time that a user holds his/her own jobs from the waiting time.
- Users are able to submit jobs which depend on the output of other jobs. This dependency is not explicitly recorded in the data trace. Consequently, the time a job spends waiting for other jobs to complete is included in the waiting time.

Also note that the factors affecting waiting time presented in this section are not exhaustive. Requesting resources such as additional memory or GPUs may increase the waiting time. However, only 3.6% of jobs request more than the default 3 GB of memory and only 0.09% request GPUs.

4.7 Comparison with Other Clusters

From the comparison of the interarrival time, typical demand and running time in this section, we can conclude that public jobs arrive more frequently and run for a shorter time than other jobs. As a result, while public jobs constitute 44% of the jobs, they make up only 16% of the workload. This uniqueness in job characteristic is highlighted in Fig. 7. Public jobs on the SCC have the shortest interarrival time of all the clusters displayed except for Intel NetBatch, a large production grid, and one of the shortest average running time. Fig. 7 shows that in relation to other grids, the SCC as a whole tends to have a relatively short interarrival time and relatively low running times. Note a similar difference in workload characteristic between LLNL-Atlas and LLNL-Thunder, mentioned in Section 2. Relative to each other, Thunder's trace consists of small frequent jobs while Atlas's trace consists of long infrequent jobs, similar to our observations about public jobs in the SCC.

5 Conclusion

Many prior studies have focused on characterizing the workload of computing clusters, (e.g., using real workload traces from the GWA and PWA). However, to

the best of our knowledge, no previous work focuses on shared/buy-in systems such as the SCC. Our study focuses on investigating differences in job characteristics between shared and buy-in resources, identifying requests with high waiting times, and quantifying the advantages of public queues.

We show that public jobs arrive more frequently than shared and buy-in jobs, but are smaller when measured by running time or workload. Particularly, when considered independently, public jobs have the shortest interarrival time of 14 clusters taken from the PWA and GWA. Implementation of public queues increases the performance of the SCC by increasing utilization of buy-in nodes and preventing shared nodes from exceeding their capacity. Public queues also increase the QoS for shared users by granting them access to portions of buy-in resources that would otherwise be idle. Additionally, we identify RTL and parallel environment as the two main factors affecting waiting time. For requests exceeding 12 hours, the mean waiting time of shared jobs is more than twice that of buy-in jobs. Parallel jobs, especially MPI jobs, have longer mean waiting times than one-slot jobs for all job types. These results should provide useful guidelines to users as to whether to purchase buy-in resources or not. Specifically, users should consider buying-in if they need to submit many MPI jobs or jobs that run for over 12 hours.

Our analysis of utilization and waiting time can help the SCC in implementing policies that maximize the system's utilization and minimize waiting time for users. For example, our analysis of waiting time factors shows that a large amount of resources needs to be dedicated to parallel jobs, which experience longer waiting times. Furthermore, since waiting times are longer over periods of high utilization, users could be encouraged to run their jobs during periods of low utilization (late night and weekends). Possible approaches include relaxing the limit on the number of slots used by a user over low utilization periods or decreasing the charge when using a slot over these periods. The evaluation and implementation of such policies represent interesting areas for future work.

Acknowledgment

This research was supported in part by the NSF under grants 1717858, 1012798, 1117160, 1414119, and 1430145, and by the Hariri Institute for Computing at BU. The authors would also like to acknowledge the Research Computing Services group at Boston University, including Glenn Bresnahan, Mike Dugan, and Katia Oleinik, for their guidance and technical support.

References

1. Calzarossa, M., Massari, L., Tessera, D.: Workload characterization issues and methodologies. In: Performance Evaluation: Origins and Directions, pp. 459–482. Springer (2000)
2. Cao, J., Chan, A.T.S., Sun, Y., Das, S.K., Guo, M.: A taxonomy of application scheduling tools for high performance cluster computing. Cluster Computing **9**(3), 355–371 (2006). DOI 10.1007/s10586-006-9747-2. URL https://doi.org/10.1007/s10586-006-9747-2
3. Delamare, S., Fedak, G., Kondo, D., Lodygensky, O.: Spequlos: a qos service for hybrid and elastic computing infrastructures. Cluster Computing **17**(1), 79–100 (2014). DOI 10.1007/s10586-013-0283-6. URL https://doi.org/10.1007/s10586-013-0283-6
4. Delignette-Muller, M.L., Dutang, C., Pouillot, R., Denis, J.B., Delignette-Muller, M.M.L.: Package 'fitdistrplus' (2015)
5. Di, S., Kondo, D., Cirne, W.: Characterization and comparison of cloud versus grid workloads. In: Cluster Computing (CLUSTER), 2012 IEEE International Conference on, pp. 230–238. IEEE (2012)
6. Dümmler, J., Kunis, R., Rünger, G.: Separat: scheduling support environment for parallel application task graphs. Cluster Computing **15**(3), 223–238 (2012). DOI 10.1007/s10586-012-0211-1. URL https://doi.org/10.1007/s10586-012-0211-1
7. Feitelson, D.: Parallel workloads archive (2005). URL http://www.cs.huji.ac.il/labs/parallel/workload/
8. Feitelson, D.G., Tsafrir, D., Krakov, D.: Experience with using the parallel workloads archive. Journal of Parallel and Distributed Computing **74**(10), 2967–2982 (2014)
9. Iosup, A., Li, H., Jan, M., Anoep, S., Dumitrescu, C., Wolters, L., Epema, D.H.: The grid workloads archive. Future Generation Computer Systems **24**(7), 672–686 (2008)
10. Klausner, Y., Liao, C., Starobinski, D., Simhon, E., Bestavros, A.: Workload characterization of the shared/buy-in computing cluster at boston university. In: 2016 IEEE MIT Undergraduate Research Technology Conference. IEEE (2016)
11. Krakov, D., Feitelson, D.G.: Comparing performance heatmaps. In: Workshop on Job Scheduling Strategies for Parallel Processing, pp. 42–61. Springer (2013)
12. Kübert, R., Wesner, S.: High performance computing as a service with service level agreements. In: Services Computing (SCC), 2012 IEEE Ninth International Conference on, pp. 578–585. IEEE (2012)
13. Kumar, R., Vadhiyar, S.: Identifying quick starters: towards an integrated framework for efficient predictions of queue waiting times of batch parallel jobs. In: Workshop on Job Scheduling Strategies for Parallel Processing, pp. 196–215. Springer (2012)
14. Li, H., Groep, D., Wolters, L.: Workload characteristics of a multi-cluster supercomputer. In: Workshop on Job Scheduling Strategies for Parallel Processing, pp. 176–193. Springer (2004)
15. Livny, M.: Hpc cluster buy in options - center for high throughput computing (2016). URL http://chtc.cs.wisc.edu/hpc-buy-in.shtml
16. Oracle-Corporation: Beginner's guide to oracle grid engine 6.2 (2010). URL http://www.oracle.com/technetwork/oem/host-server-mgmt/twp-gridengine-beginner-167116.pdf
17. Qureshi, K., Shah, S.M.H., Manuel, P.: Empirical performance evaluation of schedulers for cluster of workstations. Cluster Computing **14**(2), 101–113 (2011). DOI 10.1007/s10586-010-0128-5. URL https://doi.org/10.1007/s10586-010-0128-5
18. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2016). URL https://www.R-project.org/
19. Russell, J.: Buy-in — university of arizona research computing (2016). URL http://rc.arizona.edu/buy-in

20. Shanny Anoep Catalin Dumitrescu, D.E.A.I.M.J.H.L.L.W.: Grid workloads archive (2016). URL http://gwa.ewi.tudelft.nl/datasets/
21. and Technology, B.U.I.S.: Research computing support. URL http://www.bu.edu/tech/support/research/
22. Tran, N.M., Wolters, L.: Towards a profound analysis of bags-of-tasks in parallel systems and their performance impact. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing, HPDC '11, pp. 111–122. ACM, New York, NY, USA (2011). DOI 10.1145/1996130.1996148. URL http://doi.acm.org/10.1145/1996130.1996148
23. Zhang, H., Jiang, G., Yoshihira, K., Chen, H., Saxena, A.: Intelligent workload factoring for a hybrid cloud computing model. In: Services-I, 2009 World Conference on, pp. 701–708. IEEE (2009)