Performance Evaluation of Selected Job Management Systems

Kris Gaj[1], Tarek El-Ghazawi[2], Nikitas Alexandridis[2], Frederic Vroman[2], Nguyen Nguyen[1], Jacek R. Radzikowski[1], Preeyapong Samipagdi[2], and Suboh A. Suboh[2]
[1] George Mason University, [2] The George Washington University
kgaj@gmu.edu, tarek@seas.gwu.edu, alexan@seas.gwu.edu

Abstract

One important component of grid software infrastructure and parallel systems management is the Job Management System (JMS). With many JMSs available commercially and in public domain, it is difficult to choose the most efficient JMS for a given computing environment. All previous comparisons of JMSs had only a conceptual character. In this paper, we present the results of the first empirical study of JMSs reported in the literature. Two most popular commercial systems, LSF and PBS Pro, were included in our study. The study has revealed important strengths and weaknesses of these JMSs under different operational conditions. For example, LSF was shown to exhibit excellent throughput for a wide range of job types and submission rates. On the other hand, PBS appeared to excel in terms of turn-around time. Whenever possible, our study have tried to identify and explain the reasons behind the observed behavior of investigated JMSs.

1. Introduction

A lot of work has been done in grid software infrastructure [1]. One of the major tasks of this infrastructure is *job management*, also known as workload management, load sharing, or load management. Software systems capable of performing this task are referred to as *Job Management Systems* (*JMSs*) [2].

Job Management Systems can leverage under-utilized computing resources to serve remote users who currently have the needs, in a grid computing like style. Most JMSs can operate in multiple environments, including heterogeneous clusters of workstations, supercomputers, and massively parallel systems. The focus of our study is performance of JMSs in a loosely coupled cluster of heterogeneous workstations.

Taking into account the large number of JMSs available commercially and in public domain, choosing the best JMS for particular type of distributed computing environment is not an easy task. All previous comparisons of JMSs reported in literature had only a conceptual character. In [3], selected JMSs were compared and contrasted according to a set of well defined criteria. In [4, 5], the job management requirements for the Numerical Aerodynamic Simulation (NAS) parallel systems and clusters at NASA Ames Research Center were analyzed and several popular JMSs evaluated according to these criteria. In [6, 7], three widely used JMSs were analyzed from the point of view of their use with Sun HPC Cluster Tools. Finally, our earlier conceptual study, reported in [8], gave a comparative overview and ranking of twelve popular systems for distributed computing, including several JMSs.

In this paper, we extend the conceptual comparison with the empirical study based on a set of well defined experiments performed in a uniform fashion in a controlled computing environment [9]. To our best knowledge, this is a first reported experimental study quantifying the relative performance of Job Management Systems.[1]

Our paper is organized as follows. In Section 2, we give an introduction to Job Management Systems, and summarize conceptual functional differences among them. In Section 3, we define metrics used for comparison, present our experimental setup, and discuss parameters and role of all experiments. In Section 4, we describe our methodology and tools used for the measurement collection. Finally, in Sections 5 and 6, we present experimental results, their analysis, and we draw conclusions regarding the relative strengths and weaknesses of investigated JMSs.

2. Job Management Systems

2.1. General architecture of a JMS

The objective of a JMS, investigated in this paper, is to let users execute jobs in a non-dedicated cluster of workstations with a minimum impact on owners of these workstations by using computational resources that can be spared by the owners. The system should be able to perform at least the following tasks:

a. monitor all available resources,

[1] This work was sponsored by the Department of Defense under the LUCITE contract #MDA904-98-C-A081 and by RIACS/NASA/IPG

b. accept jobs submitted by users together with resource requirements for each job,

c. perform centralized job scheduling that matches all available resources with all submitted jobs according to the predefined policies,

d. allocate resources and initiate job execution,

e. monitor all jobs and collect accounting information.

To perform these basic tasks, a JMS must include at least the following major functional units shown in Fig. 1:

1. *User server* – which lets user submit jobs and their requirements to a JMS (task b), and additionally may allow the user to inquire about the status and change the status of a job (e.g., to suspend or terminate it).
2. *Job scheduler* – which performs job scheduling and queuing based on the resource requirements, resource availability, and scheduling policies (task c).
3. *Resource manager*, including

- *Resource monitor* – which collects information about all available resources (tasks a and e), and
- *Job dispatcher* – which allocates resources and initiates execution of jobs submitted to JMS (task d).

2.2. Choice of Job Management Systems

More than twenty JMS packages, both commercial and public domain, are currently in use [2, 3, 8]. For interest of time we selected two representative and most commonly used commercial JMSs

- LSF – Load Sharing Facility, and
- PBS Pro – Portable Batch System

The common feature of these two JMSs is that both of them are based on a central Job Scheduler running in a single computational node.

LSF (Load Sharing Facility) is a commercial JMS from Platform Computing Corp. [10]. It evolved from the Utopia system developed at the University of Toronto [11], and is currently probably the most widely used JMS.

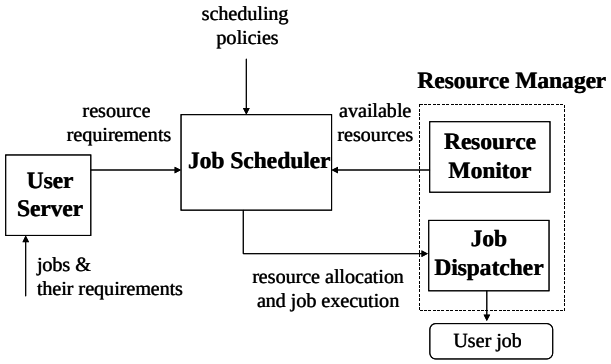

Figure 1. Major functional blocks of a Job Management System

PBS (Portable Batch System) has both a public domain and a commercial version. The commercial version called PBS Pro is supported by Veridian Systems. This version was used in our experiments. PBS was originally developed to manage aerospace computing resources at NASA Ames Research Center.

2.3. Functional similarities and differences among selected Job Management Systems

The most important functional characteristics of selected two JMSs are presented and contrasted in Table 1. From this table, it can be seen that LSF supports all operating systems, job types, and features included in the table. PBS trails LSF in terms of support for parallel jobs, process migration, dynamic load balancing, checkpointing, and master daemon fault recovery. It also does not support Windows NT.

Table 1. Conceptual functional comparison of selected Job Management Systems

	LSF	PBS
Distribution	commercial	commercial and public domain
Linux, Solaris	yes	yes
Tru64	yes	yes
Windows NT	yes	no
Interactive jobs	yes	yes
Parallel jobs	yes	partial
Stage-in and stage-out	yes	yes
Process migration	yes	no
Dynamic load balancing	yes	no
Checkpointing	yes	only kernel-level
Daemon fault recovery	master and execution hosts	only for execution hosts

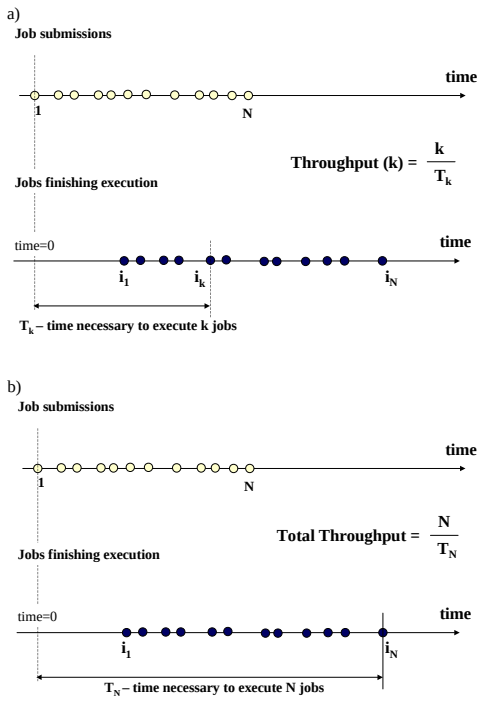

Figure 2. Definition of (a) partial and (b) total throughput

3. Experimental Setup

3.1. Metrics

The following performance measures were investigated in our study:

1. Throughput is defined in general as a number of jobs completed in a unit of time. Nevertheless, this number depends strongly on how many jobs are taken into account. Therefore, we introduce the notion of Partial Throughput with parameter *k*, and define it as *k* divided by the amount of time necessary to complete *k* JMS jobs (see Fig. 2a). We also define Total Throughput as a special case of Partial Throughput for parameter *k* equal to the total number of jobs submitted to a JMS during the experiment, *N* (see Fig. 2b).

In Fig. 3, we show the typical dependence of the partial throughput on the number of jobs taken into account, *k*. It can be seen that the partial throughput increases sharply as a function of *k* until the moment when either all system CPUs become busy, or the number of jobs submitted and completed in a unit of time become equal. When the number of jobs taken into account, *k*, gets close to the total number of jobs submitted during the experiment, the throughput drops sharply and unpredictably.

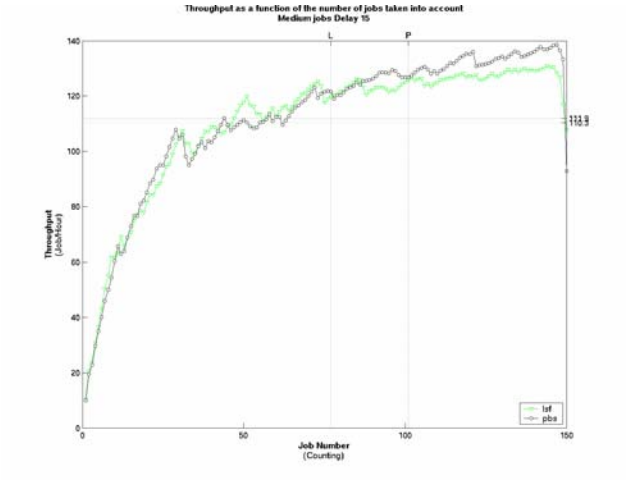

Figure 3. Throughput as a function of the number of jobs taken into account

This drop is the result of a boundary effect and is not likely to appear during the regular operation of a JMS, when the flow of jobs submitted to a JMS continues uninterrupted for a long period of time. The total throughputs are affected by this boundary effect, and as a result do not characterize well the relative performance of JMSs for real-life workloads. Therefore, we decided to use for comparison Average Throughput, defined as the Partial Throughput averaged over all possible values of the job number, *k*.

2. Average turn around time is the time from submitting a job till completing it, averaged over all jobs submitted to a JMS (see Fig. 4).

3. Average response time is the average amount of time between submitting a job to a JMS and starting the job on one of the execution hosts (see Fig. 4).

4. Utilization is the ratio of a busy time span to the available time span. In our experiments, we measured the utilization by measuring the average percentage of the CPU time used by all JMS jobs on each execution host. These average machine utilizations were then averaged over all execution hosts (see Fig. 5).

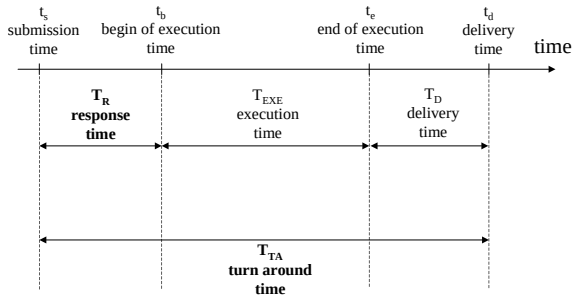

Figure 4. Definition of timing parameters

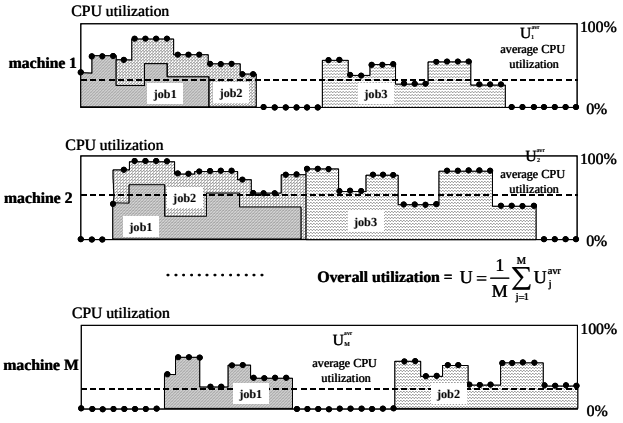

Figure 5. Definition of the system utilization and its measurement using `top`

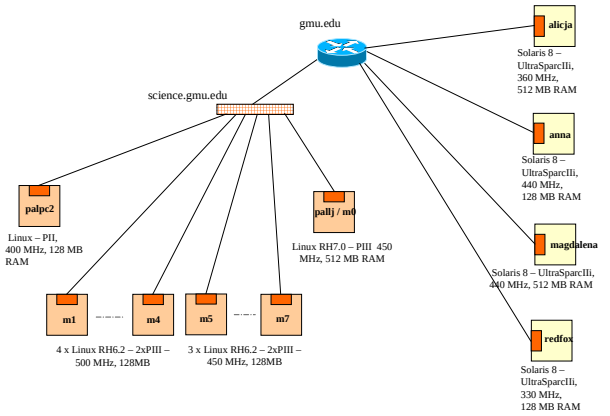

Figure 6. A Micro-grid testbed used in the experimental study

3.2. Our Micro-grid testbed

A Micro-grid testbed used in our experiments is shown in Fig. 6. The testbed consists of 9 PCs running Linux OS, and 4 workstations Ultra 5, running Solaris 8. The total number of CPUs available in the testbed is 20. The network structure of the testbed is flat, so that every machine can serve as both an execution host and a submission host. In all our experiments, `pallj` was used as a master host for all Job Management Systems. All 13 hosts, including the master host, were configured as execution hosts. In all our experiments, `pallj` was also employed as a submission host.

3.3. Application benchmarks

A set of 36 benchmarks has been compiled and installed on all machines of our testbed. These programs belong to the following four classes of benchmarks: NSA HPC Benchmarks, NAS Parallel Benchmarks, UPC Benchmarks, and Cryptographic Benchmarks. Each benchmark has been characterized in terms of the CPU time, wall time, and memory requirements using one of the Linux machines.

All benchmarks have been divided into the following three sets of benchmarks:

1. Set 1 – Short job list – 16 benchmarks with an execution time between 1 second and 2 minutes, and an average execution time equal to 22 seconds.
2. Set 2 – Medium job list – 8 benchmarks with an execution time between 2 minutes and 10 minutes, and an average execution time equal to 7 minutes 22 seconds.
3. Set 3 – Long job list – 6 benchmarks with an execution time between 10 minutes and 31 minutes, and an average execution time equal to 16 minutes 51 seconds.

3.4. Experiments

Each experiment consists of running N jobs chosen pseudorandomly from the given set of benchmarks, and submitted one at a time to a given JMS in the pseudorandom time intervals. All jobs were submitted from the same machine, `pallj`, and belonged to a single user of the system. The rate of the job submissions was chosen to have a Poisson distribution.

Table 2. Characteristics of experiments performed during our study

Experiment Number	Benchmark Set	Average CPU time / Job	Average Time Intervals Between Job Submissions	Total Number of Jobs	Special Assumptions
1	Set 2, Medium job list	7 min 22 s	30 s, 15 s, 5 s	150	one job / CPU
2	Set 3, Long job list	16 min 51 s	2 min, 30 s	75	one job / CPU
3	Set 1, Short job list	22 s	15 s, 10 s, 5 s, 2 s, 1 s	150	one job / CPU

The only job requirement specified during the job submission was the amount of available memory. No information about the expected execution time, or limits on the wall or CPU time were specified.

The total number of jobs submitted to a system, N, was chosen based on the expected total time of each experiment, the average execution time of jobs from the given list, and the number of machines in our testbed. In Experiments 1 and 3, regarding medium and short job lists, the total number of jobs was set to 150, which led to a total experiment time of about two hours. In Experiment 2, regarding the long job list, the total number of jobs was reduced to 75 to keep the time of each experiment within the range of 2 hours.

Each experiment was repeated for both JMSes, under exactly the same initial conditions, including the same initial seeds of the pseudorandom generators. Additionally, all experiments were repeated 3 times for the same JMS to minimize the effects of random events in all machines participating in the experiment.

Additionally, each experiment was repeated for several different average job submission rates. These rates have been chosen experimentally in such a way that they correspond to qualitatively different JMS loads. For the smallest submission rate, each system is very lightly loaded. Only a subset of all available CPUs is utilized at any point in time. Any new job submitted to the system can be immediately dispatched to one of the execution hosts. For the highest submission rate, a majority of CPUs are busy all the time, and almost any new job submitted to a JMS must spend some time in a queue before being dispatched to one of the execution hosts.

The characteristic parameters of three experiments performed during our experimental study are summarized in Table 2.

3.5 Common settings of Job Management Systems

An attempt was made to set both JMSes to an equivalent configuration, using the following major configuration settings:

A. Maximum Number of Jobs per CPU

In all experiments, except Experiment 2, a maximum number of jobs assigned simultaneously to each CPU was set to one. In other words, no timesharing of CPUs was allowed. This setting was chosen as an optimum because of the numerical character of benchmarks used in our study. All benchmarks from the short, medium, and long job lists have no input or output. For this kind of benchmarks, timesharing can improve only the response time, but has a negative effect on two most important performance parameters: turn-around time and throughput.

B. CPU factor of execution hosts

The CPU factors determine the relative performance of execution hosts for a given type of load. Based on the recommendations given in the LSF manual, CPU factors for LSF were set based on the relative performance of benchmarks representing a typical load. For each list of benchmarks, two representative benchmarks were selected, and run on all machines of distinctly different types. The CPU factors were set based on an average ratio of the execution time on the slowest machine to the execution time on the machine for which the CPU factor was determined. Based on this procedure, the slowest machine had always a CPU factor equal to 1.0. The CPU factors of remaining machines varied in the range from 1.2 to 1.7 for a small job list, and from 1.4 to 1.95 for the medium and long job lists. The CPU factors in LSF affect the operation of the scheduler. In PBS, the equivalent parameter has no effect on scheduling, and affects only accounting and time limit enforcement.

C. Dispatching interval

The dispatching interval determines how often the JMS scheduler attempts to dispatch pending jobs. This parameter clearly affects an average response time, as well as scheduler overhead. It may also influence the remaining performance parameters.

LSF and PBS use a different definition of this parameter. In both systems, this parameter describes the maximum time in seconds between subsequent attempts to schedule jobs. However in PBS, the attempts to schedule a job also occur whenever a new job is submitted, and whenever a running batch job terminates. The same is not the case for LSF. On the other hand, LSF has two additional parameters that can be used to limit the time spent by the job in the queue, and thus reduce the response time.

F. Scheduling policies

No changes to the parameters describing scheduling policies were made, which means that the default First Come First Served (FCFS) scheduling policy was used for all systems. One should be however aware that within this policy, a different ranking of hosts fulfilling job requirements might be used by different JMSs.

4. Methodology and measurement collection

Each experiment was aimed at determining values of all performance measures defined in Section 3.1. All parameters were measured in the same way for all JMSs, using utilities and mechanisms of the operating systems only.

In particular, timestamps generated using the C function `gettimeofday()`, were used to determine the exact time of a job submission, as well as the begin and end of the execution time.

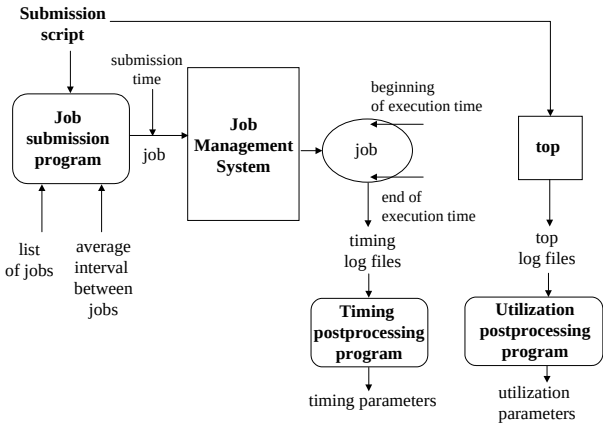

Figure 7. Software used to collect performance measures

The function `gettimeofday()` gets the current time from the operating system. The time is expressed in seconds and microseconds elapsed since Jan 1, 1970 00:00 GMT. The actual resolution of the returned time depends on the accuracy of the system clock, which is hardware dependent. The Unix Network Time Protocol (NTP) was used to synchronize clocks of all machines of our Micro-grid. The protocol provides accuracy ranging from milliseconds (on LANs) to tenths of milliseconds (on WANs).

In order to determine the JMS utilization, the Unix `top` utility was used. This utility records an approximate percentage of the CPU time used by each process running on a single machine averaged over a short period of time, e.g., 15 seconds (see Fig. 5). For each point in time, the sum of percentages corresponding to all JMS jobs is computed. These sums are then averaged over the entire duration of an experiment, to determine an average utilization of each machine by all JMS jobs The execution host utilizations averaged over all execution hosts determine the overall utilization of a JMS.

Three programs were developed to support the experiments and were used in a way shown in Fig. 7. A C++ Job Submission program has been written to emulate a random submission of jobs from a given host. This program takes as an input a list of jobs, a total number of submissions, an average interval between two consecutive submissions, and the name of a JMS used in a given experiment. Two postprocessing Perl scripts, Timing and Utilization postprocessing utilities, have been developed to process log files generated by benchmarks and the `top` utility. These scripts generate exhaustive reports including values of all performance measures separately for every execution host, and jointly for the entire Micro-Grid testbed.

5. Experimental Results

Two most important parameters determining the performance of a Job Management System from the user's point of view are turnaround time and throughput. *Throughput* is particularly important when a user submits a large batch of jobs to a JMS and does not do any further processing till all jobs complete execution. *Turnaround time* is particularly important when a user
tends to work in a pseudo-interactive mode and awaits results of each subsequent experiment.

In Experiment 1, with the medium job list, LSF and PBS are almost identical in terms of the average throughput. In terms of the average turn-around time, PBS is better by a factor ranging from 7 to 17%.

In Experiment 2, with the long job list, the throughputs of LSF and PBS are once again almost identical, and the turn-around time of PBS is better, but this time only by a small margin ranging from 1 to 6%.

In Experiment 3, for short job list, the throughput of LSF was significantly higher than the throughput of PBS. The difference between throughputs of both systems increased as a function of the job submission rate, reaching a factor of 2.1 for the two highest job submission rates. On the other hand, PBS appeared to have a smaller turn-around time for majority of job submission rates. The difference between the turn-around times of PBS and LSF was in the range of 25% for small job submission rates, and reached almost a factor of 2 for the highest investigated job submission rate.

The analysis of the system utilization and job distribution has revealed the following reasons for the different relative performance of investigated systems in terms of throughput and turn-around time. LSF tends to dispatch jobs to all execution hosts, independently of their relative speed as shown in Fig. 11a. It also uses a complex algorithm for scheduling, which guarantees that jobs are executed tightly one after the other. Both factors contribute to a high system throughput. At the same time, distributing jobs to all machines, including slow ones, increases average execution time, and complex scheduling affects average response time. Both factors contribute to the high increase in the average turn-around time. On the other hand, PBS distributes jobs only to a limited number of the fastest execution hosts, as shown in Fig. 11b. As a result, the average execution time is smaller compared to LSF, which contributes to a better average turn-around time. At the same time, the limited utilization of the execution hosts contributes to only average throughput.

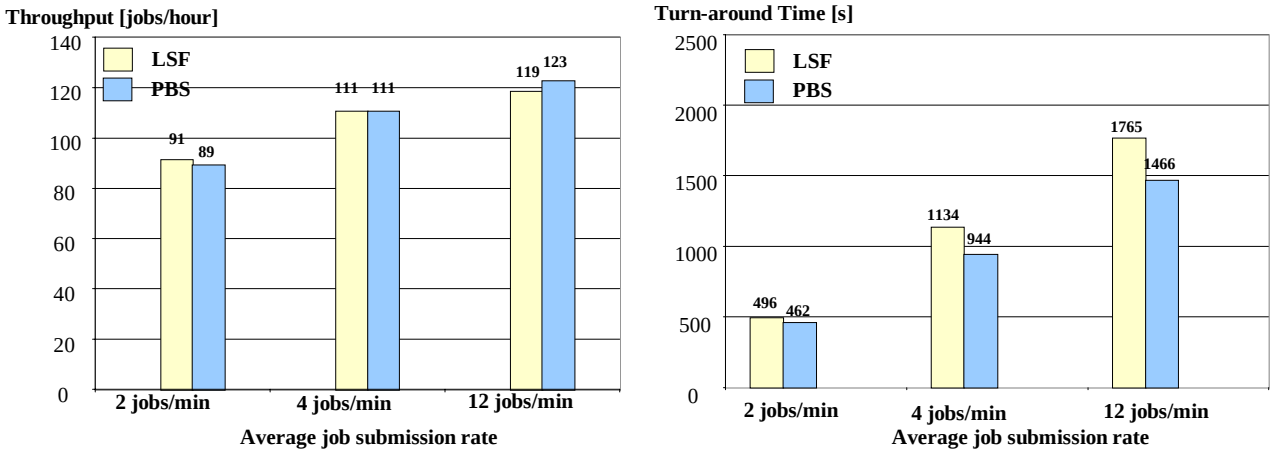

Figure 8. Average throughput and average turn-around time for the medium job list

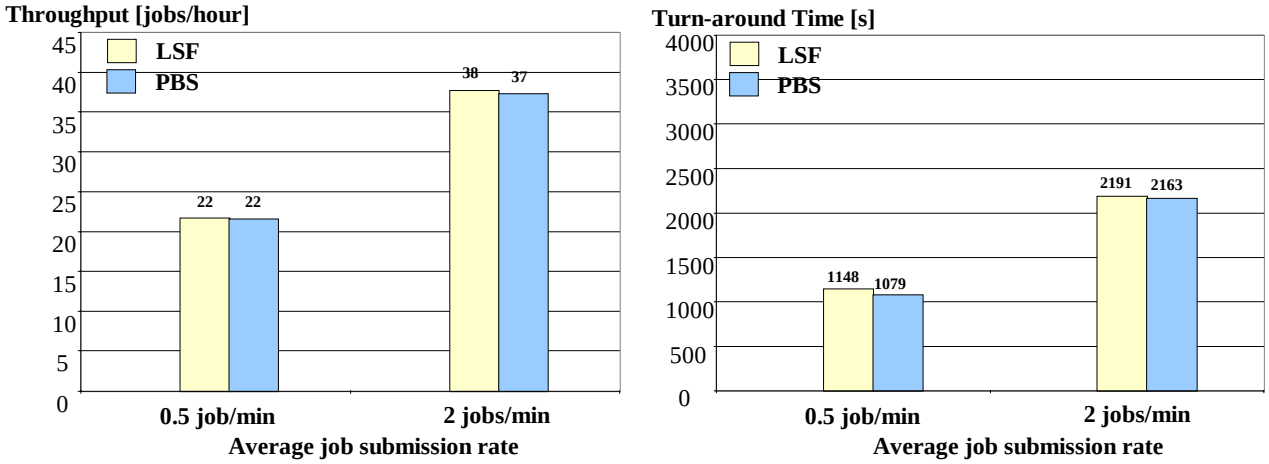

Figure 9. Average throughput and average turn-around time for the long job list

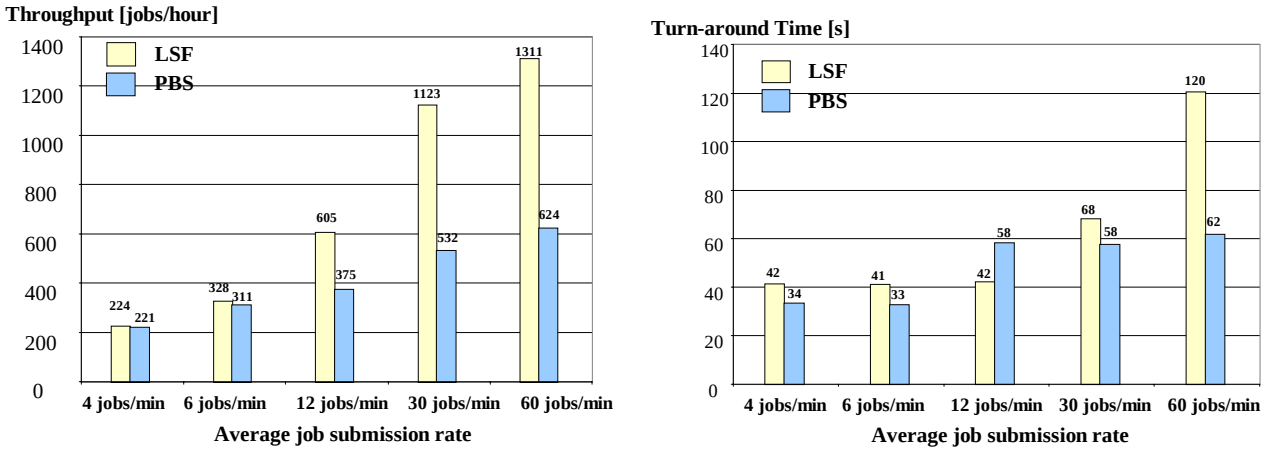

Figure 10. Average throughput and average turn-around time for the short job list

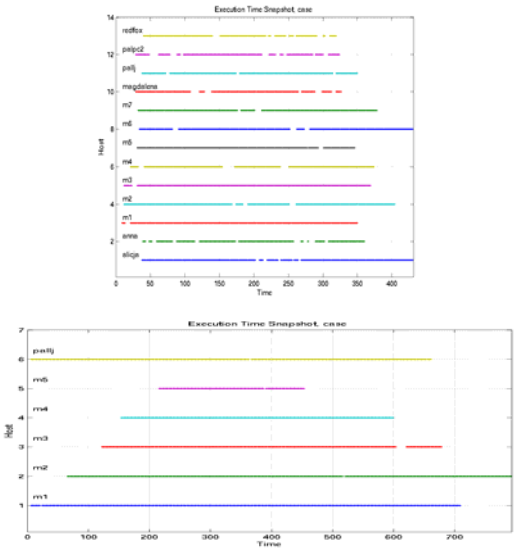

Figure 11. Utilization of machines by a) LSF and b) PBS.

6. Conclusions

Based on Figures 8-11, we can draw the following conclusions. In terms of the average system throughput, LSF and PBS appear to offer almost the same performance for medium and long jobs, and LSF outperforms PBS for short jobs, especially in case of large job submission rates.

In terms of the average turn-around time, PBS is superior to LSF for almost all investigated job sizes and submission rates. The largest difference between turn-around times of both systems appeared for short jobs and large submission rates.

The relative performance of Job Management Systems is similar for medium and large jobs, and changes considerably for short jobs where the job execution times became comparable with the times required for resource monitoring and job scheduling. The obtained results have confirmed that the type of underlying workload and the considered metrics can determine the relative performance standing of different systems.

Despite the limitations resulting from the relatively small size of our Micro-Grid testbed and a limited set of system settings exercised in our experiments, the practical value of our empirical knowledge comes, among the other, from the following factors:

- Even though our benchmarks and experiment times seem to be relatively short compared to the real-life scenarios, we make up for that by setting the average time between job submissions to the relatively small values. As a result, the systems are fully exercised, and our results are likely to scale for more realistic loads with proportionally longer job execution times and longer times between job submissions.
- Typical users rarely use all capabilities of any complicated system, such as JMS. Instead, majority of Job Management Systems deployed in the field use the default values of majority of configuration parameters.

Additionally, to our best knowledge, our study is the first empirical study of Job Management Systems reported in the literature. Our methodology and tools developed as a result of this project may be used by other groups to extend the understanding of similarities and differences among behavior and performance of various Job Management Systems.

Acknowledgments

The authors would like to acknowledge Jearanai Vongsaard, Sébastien Chauvin, and Alexandru V. Staicu for their contribution to the study described in this paper.

References

[1] I. Foster and C. Kesselman (eds.), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kauffman Publishers Inc., 1999.

[2] K. Hwang, Z. Xu, *Scalable Parallel Computing: Technology, Architecture, Programming*, McGraw-Hill 1998.

[3] M. A. Baker, G. C. Fox, and H. W. Yau, "Cluster Computing Review," Northeast Parallel Architectures Center, Syracuse University, Nov. 1995.

[4] J. P. Jones, "NAS Requirements Checklist for Job Queuing/Scheduling Software," NAS Technical Report NAS-96-003 April 1996, available at http://www.nas.nasa.gov/Pubs/TechReports/NASreports/NAS-96 003/

[5] J. P. Jones, "Evaluation of Job Queuing/Scheduling Software: Phase 1 Report," NAS Technical Report, NAS-96-009, September 1996 available at http://www.nas.nasa.gov/Research/Reports/Techreports/1996/nas-96-009-abstract.html

[6] C. Byun, C. Duncan, and S. Burks, "A Comparison of Job Management Systems in Supporting HPC Cluster Tools," Proc. SUPerG, Vancouver, Fall 2000.

[7] O. Hassaine, "Issues in Selecting a Job Management Systems (JMS)," Proc. SUPerG, Tokyo, April 2001.

[8] T. El-Ghazawi, et al., *Conceptual Comparative Study of Job Management Systems*, Technical Report, February 2001, available at http://ece.gmu.edu/lucite/reports.html.

[9] T. El-Ghazawi, et al., *Experimental Comparative Study of Job Management Systems*, Technical Report, July 2001, available at http://ece.gmu.edu/lucite/reports.html.

[10] I. Lumb, "Wide-area parallel computing: A production-quality solution with LSF," Supercomputing 2000, Dallas, Texas, Nov. 2000.

[11] S. Zhou, X. Zheng, J. Wang, and P. Delisle, "Utopia: A load sharing facility for large heterogenous distributed computer systems." *Software Practice and Experience*, Dec. 1993.