Job Management System Extension To Support SLAAC-1V Reconfigurable Hardware

Mohamed Taher[1], Kris Gaj[2], Tarek El-Ghazawi[1], and Nikitas Alexandridis[1]

[1] The George Washington University
[2] George Mason University

Abstract

Reconfigurable Hardware resources or FPGA's can accelerate and improve the performance of a lot of applications, but these resources are very expensive. Job management systems (JMS) are used in resource management and job scheduling. They allow users to execute jobs on a non-dedicated cluster of workstations with a minimum impact on owners of these workstations. By using JMS we can Increase utilization of costly resources, and create a Unified interface to all computing resources.

In our experiments we used LSF Job Management System to manage and utilize the SLAAC-1V FPGA boards. In order to extend LSF to support SLAAC-1V FPGA boards, we need to develop an external resource monitor (External Load Information Manager – ELIM).

We developed the external resource monitor (ELIM). This system permits sharing these FPGA boards. The architecture was verified experimentally for the case of LSF and SLAAC-1V FPGA boards. The utilization of the idle boards was demonstrated to reach up to 95% in our experimental setting which include Linux and Windows NT workstations.

I. INTRODUCTION

This paper reports on a research effort to extend the LSF job management system to support slaac1-v reconfigurable boards [1-5]. The objective is to construct a system that can leverage underutilized resources at a given time to serve other users who currently have the needs, in a grid computing like style. The targeted type of resources are workstations and clusters that are equipped with Field Programmable Arrays (FPGA) boards serving as reconfigurable coprocessors.

Our paper is organized as follows. In Section 2, and section3 we give an introduction to Job Management Systems. In Section 4, we describe our experimental work. Finally, in Sections 5 we present experimental results, and we draw conclusions.

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,

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).

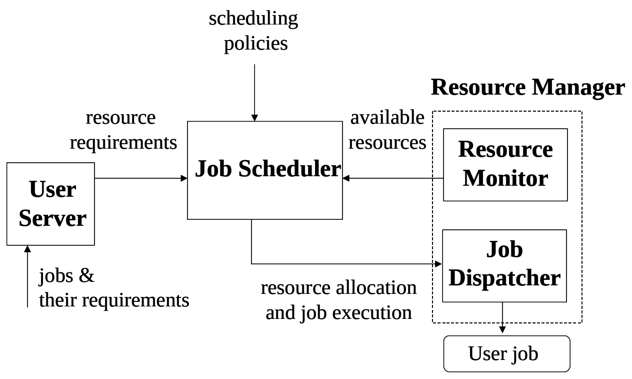

Figure 1. Major functional blocks of a Job Management System

2.2. LSF Job Management Systems

LSF (Load Sharing Facility) is one of the most commonly used commercial JMSs. The common feature of LSF JMS is that it is based on a central Job Scheduler running in a single computational node. LSF (Load Sharing Facility) is a commercial JMS from Platform Computing Corp. It evolved from the operating systems, job types, and features included in the table.

The most important functional characteristics LSF 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.

3. Extending a JMS to support reconfigurable hardware

3.1. JMS features supporting extension

The specific features of Job Management Systems that support extension to reconfigurable hardware include

- capability to define new dynamic resources,
- strong support for stage-in and stage-out in order to allow an easy transfer of the FPGA configuration bitstreams, data inputs, and results between the submission host and the execution host with reconfigurable hardware;

Table 1. Features of LSF JMS

Feature	**Availability**
Distribution	commercial
Linux, Solaris	Yes
Tru64	Yes
Windows NT	Yes
Interactive jobs	Yes
Parallel jobs	Yes
Stage-in and stage-out	Yes
Process migration	Yes
Dynamic load balancing	Yes
Checkpointing	Yes
Daemon fault recovery	master and execution hosts

- support for Windows NT and Linux, which are two primary operating systems running on PCs that can be extended with commercially available FPGA-based accelerator boards with the PCI interface.

An ease of defining new dynamic resources appears to be a minor factor in comparison. LSF seems to be easily extendable with new dynamic resources without the need for any changes in their source code. Stage-in and stage-out are supported by LSF. LSF is fully supports Windows NT

3.2. General architecture of the extended system

General architecture of the extended system is shown in Fig. 2. The primary component of this extension is an external resource monitor that controls the status of an accelerator board, and periodically communicates this status to a resource monitor. The resource monitor transfers this information periodically or by request to a Job scheduler, which uses this information to match each job that requires acceleration with an appropriate host. Job requirements regarding the new reconfigurable resource are specified during a job submission to a user server, and are enforced by a job scheduler the same way as requirements regarding default built-in resources.

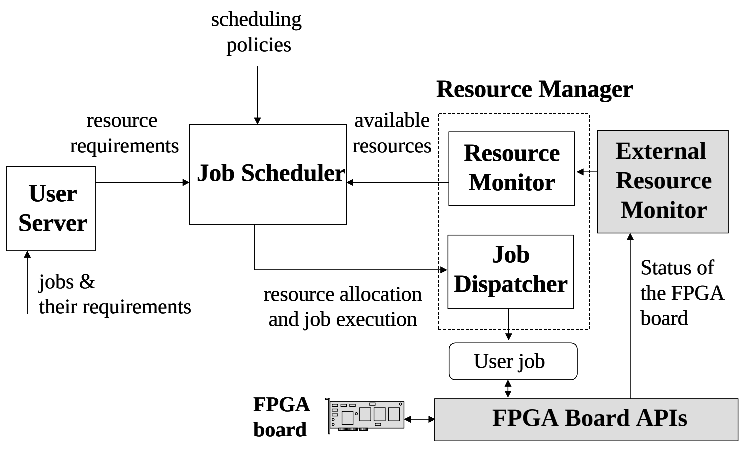

Figure 2. Extension of a JMS to recognize, monitor, and schedule reconfigurable resources

3.3. Extending LSF

Capability of defining new dynamic resources can be used to extend LSF to manage FPGA-based accelerator boards. The new resource that needs to be added to a given JMS represents the availability of the accelerator board for JMS users.

- An external resource monitor (ELIM, External Load Information Manager in LSF) needs to be written according to the specification.

This daemon is started by a local resource manager (LIM in LSF and communicates with the resource monitor using standard output.

This ELIM tries to open the board in exclusive mode, and if it succeed, this mean that the board is available, then it closes it again and reports that the board is available, else if fail this means that the board is not available and it reports that the board is not available. The flow diagram of ELIM is shown in Fig. 3.

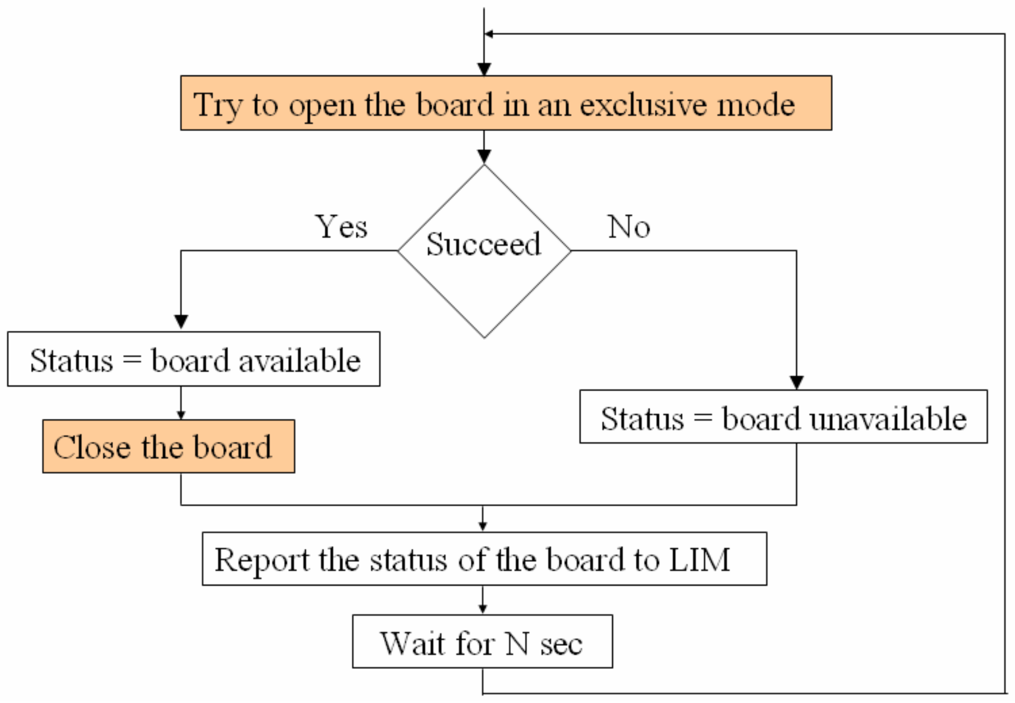
Try to open the board in an exclusive mode
Succeed
Yes
No
Status = board available
Close the board
Status = board unavailable
Report the status of the board to LIM
Wait for N sec

Figure 3. Operation of ELIM

4. Experimental Work

4.1. Extending LSF to support reconfigurable hardware

The general architecture of LSF is shown in Fig. 4. Load Information Monitors (LIMs), running on all execution hosts in the system, monitor and collect information about the current status of all static and dynamic resources available on the execution hosts. This information is periodically forwarded from every LIM to a single Master Load Information Monitor (MLIM) residing on the master host. The combined report about the current status of all system resources, collected by MLIM, is used by the Master Batch Daemon (MBD) to match available resources with resource requirements specified during the job submission. When a job waiting in the queue is

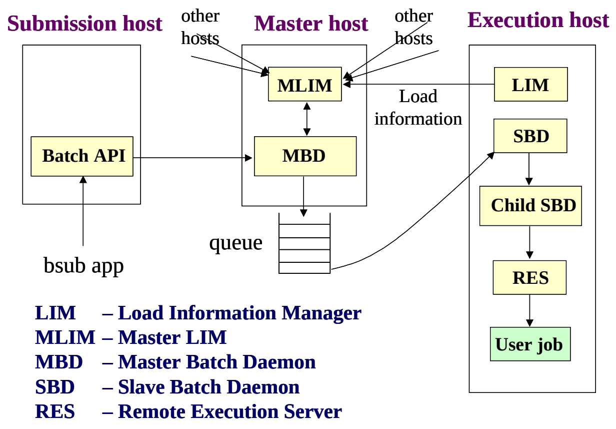

Figure 4. General architecture of LSF

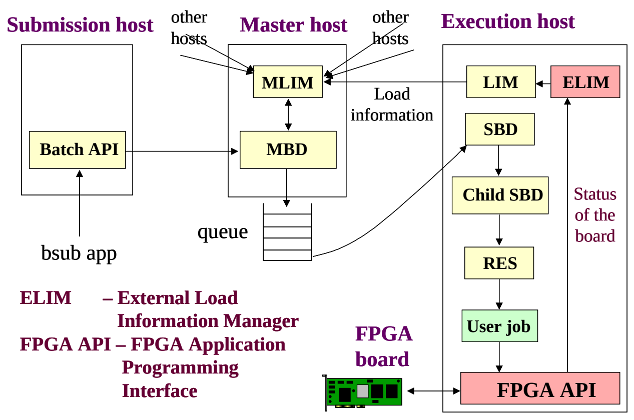

Figure 5. General architecture of LSF after extension to support reconfigurable hardware

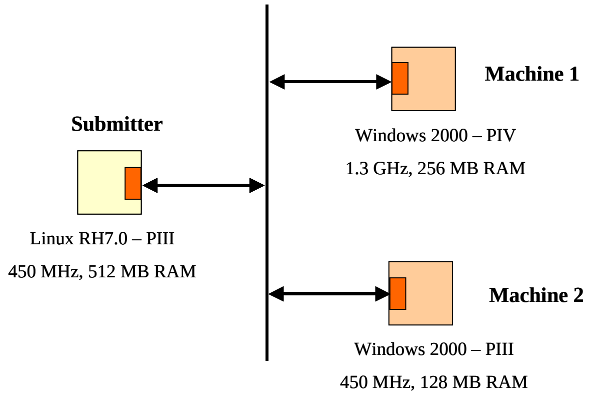

Figure 6. Experimental testbed

matched with an execution host containing the required resources, this job is being dispatched by MBD to the appropriate execution host. The job is prepared for execution by the Slave Batch Daemon (SBD), and started by the Remote Execution Server (RES). SBD is responsible for enforcing local LSF policies and maintaining the status of the job.

To support reconfigurable resources, such as FPGA-based accelerator boards, the LSF system needs to be extended with two extra components: External Load Information Monitor (ELIM) and an FPGA Board Application Programming Interface (API), as shown in Fig. 5. ELIM is a program or script that must be run on each execution host that contains a non-standard dynamic resource, such as an FPGA board. The task of ELIM is to monitor the availability of the FPGA board and to report this availability in the predefined format to LIM. To perform this task, ELIM uses functions of the FPGA Board API. These functions communicate with the FPGA board driver in order to determine whether the board is currently occupied by any job. If this is the case, ELIM reports through LIM to Master LIM (MLIM) that the FPGA board is temporarily unavailable. Otherwise, the information about the availability of the FPGA board is passed to MLIM.

Each user job that makes use of reconfigurable resources needs at the beginning of its execution check the availability of the board. If the board is unavailable, the job exits with an error code, and is resubmitted by LSF at a later time. If the board is available, the job reserves the board for exclusive use, and then configures the board using the configuration bitstream residing on the execution host or downloaded from the submission host using the stage-in capability of LSF. As soon as the board is configured, its clock is started and the FPGA circuit starts communicating with the job running on the execution host. Inputs are sent to the board, and outputs generated by the FPGA circuit are sent back to the job. After the FPGA circuit completes execution, it communicates this fact to the job, which makes final postcomputations, frees the board for use by other jobs, and finishes execution. All described above operations are facilitated by the FPGA board APIs.

4.2 Experimental setup

Our testbed consists of two Windows machines configured as execution hosts, and one Linux machine configured as a job submitter as shown in Fig. 5. Both execution hosts are extended with the SLAAC1-V FPGA accelerator board from the USC-Information Sciences Institute [8, 9].

The benchmark used in our experiments is a hardware implementation of an exhaustive key search attack against Data Encryption Standard (DES). Exhaustive key search is an attack aimed at breaking a cipher by checking all possible keys one by one. To be able to perform this attack, an opponent must know a short fragment of the message and a corresponding fragment of the ciphertext (encrypted message). By decrypting a fragment of the ciphertext with a given key, and comparing the result with a known fragment of the message, a single key can be verified. By repeating the same operation with all possible key values, one is guaranteed to find the correct key. The number of all possible keys in DES is $2^{56} \approx 7.2 \cdot 10^{16}$. This large number of repetitions calls for parallelization of computations. Additionally, since DES was designed to be efficient in hardware rather than in software, an FPGA based hardware accelerator can speed up the required computations by orders of magnitude compared to the purely software parallel implementation.

The inputs to each benchmark are the message block, the ciphertext block, the beginning of the key range, and the key range size. The output is the number and the list of matching keys. The time of the benchmark execution can be set to an arbitrary value, since it is directly proportional to the key range size, and almost independent of other parameters. In our experiments, key range was set to values that guaranteed the execution times listed in Table 2.

Our implementation consists of two parts. Hardware part was written in VHDL, and was transformed into the FPGA configuration bitstream using Xilinx tools. Software part is responsible for reserving an FPGA board for an exclusive use, downloading the configuration bitstream to the board, transferring input parameters to the hardware part, collecting results generated by the board, and releasing the board. During the majority of the time, the program is idle and its only function is to wait for a board to complete execution. This way, the only resource of the execution hosts which is fully utilized during the benchmark execution is the time of the FPGA-based accelerator.

Table 2 Features and parameters of performed experiments

Exp. No.	No. of execution hosts	Number and execution times of jobs	Delay between job submissions
1	2	40 x 20 s	5 s
2	2	8 x 20 s, 8 x 30 s, 8 x 40 s, 8 x 50 s, 8 x 60 s	5 s
3	2	40 x 120 s	5 s
4	2	40 x 300 s	5 s
5	2	3 x 20 s 3 x 40 s 3 x 60 s 3 x 80 s 3 x 100 s 3 x 120 s 3 x 140 s 3 x 160 s 3 x 180 s 3 x 200 s 3 x 220 s 3 x 240 s 3 x 260 s 3 x 280 s 3 x 300 s	5 s

Each experiment consisted of running N jobs chosen from the given set of benchmarks, and submitted one at a time to LSF in the pseudorandom time intervals. All jobs were submitted from the same Linux machine, and belonged to a single user of the system. The rate of the job submissions was chosen to have a Poisson distribution. The submission rate was relatively high with an average interval between consecutive job submissions equal to 5 seconds.

All jobs on the lists were the instances of the exhaustive key search benchmark, and differed only with values of input parameters. All these jobs required acceleration by the SLAAC1-V board. The same Linux machine was used as the submission host and the master host. The primary job requirement specified during the job submission was an availability of the specific type of the FPGA board. The second parameter specified during the job submission was the estimated execution time of the job.

The total number of jobs submitted to a system, N, was chosen based on the expected total time of the experiment, the average execution time of jobs from the given list, and the number of machines in our testbed.

5. Experimental Results

The behavior and performance of the extended Job Management System is shown in Figs. 7-10 and Table 3. For each execution host (Machine 1 and Machine 2) three timing traces are presented. The bottom trace shows timing intervals when jobs dispatched to the given execution host are executed. The numbers above these intervals are the numbers of jobs in the order of their submission. The middle trace shows time intervals when ELIM reports to LIM that the FPGA board is free for use by another job. The top timing trail represents intervals when MLIM is aware that the board is available for use by another job waiting in the queue. The very bottom trace in each figure is common for the entire system, and shows points in time when jobs were being submitted to LSF from the submission host.

In all experiments, all jobs are being submitted to JMS shortly after the beginning of the experiment, and as a result spend most of the time waiting in the queue for their turn to execute. At the beginning of every experiment both ELIM and MLIM report that all FPGA-boards are available for scheduling. As soon as a job is dispatched to the given machine for execution, ELIM running on the same machine becomes aware that the FPGA board is not any longer available. Similarly, as soon as any job completes its execution, ELIM reports to LIM that the board is available for use by another job.

FPGA board utilization is summarized in Table 3.

Table 3. Results of experiments

Experiment No.	**Utilization [%]**		
	iteration 1	iteration 2	iteration 3
1	65.9	78.7	62.8
2	75.6	86.4	84.4
3	90.8	91.6	92.3
4	95	93.6	94
5	91.9	96.2	93.9

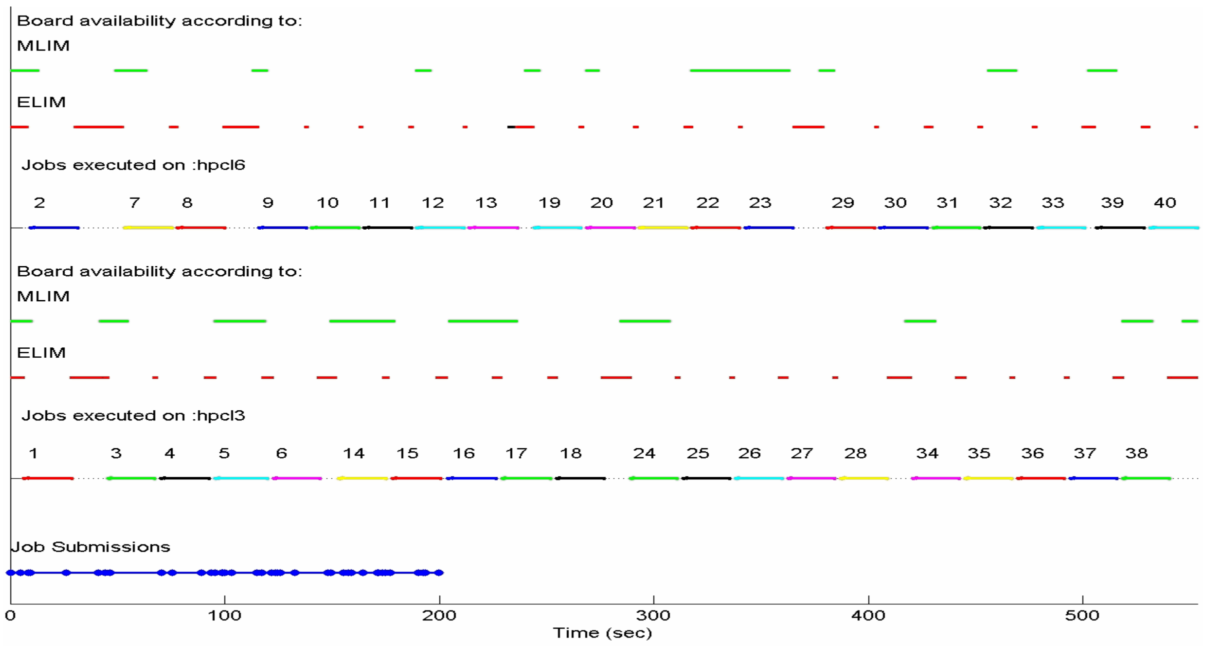

Figure. 7 Utilization of machines in Experiment 1, Iteration 1

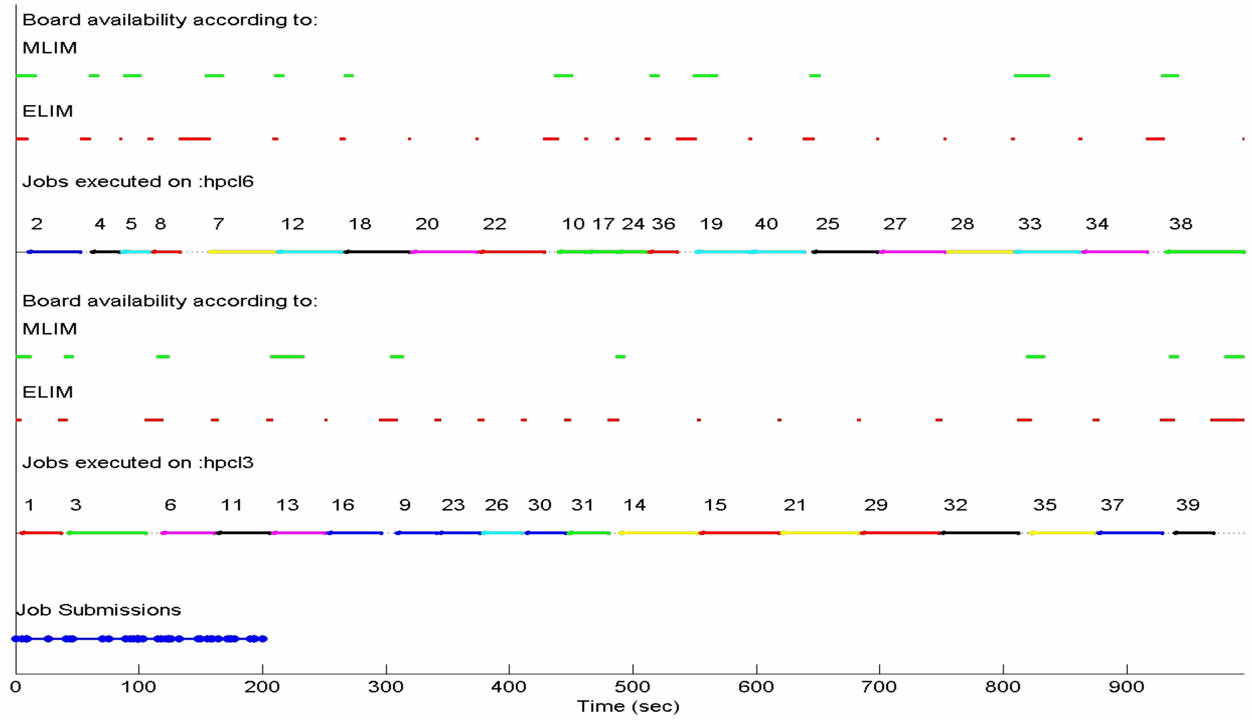

Figure. 8 Utilization of machines in Experiment 2, Iteration 1

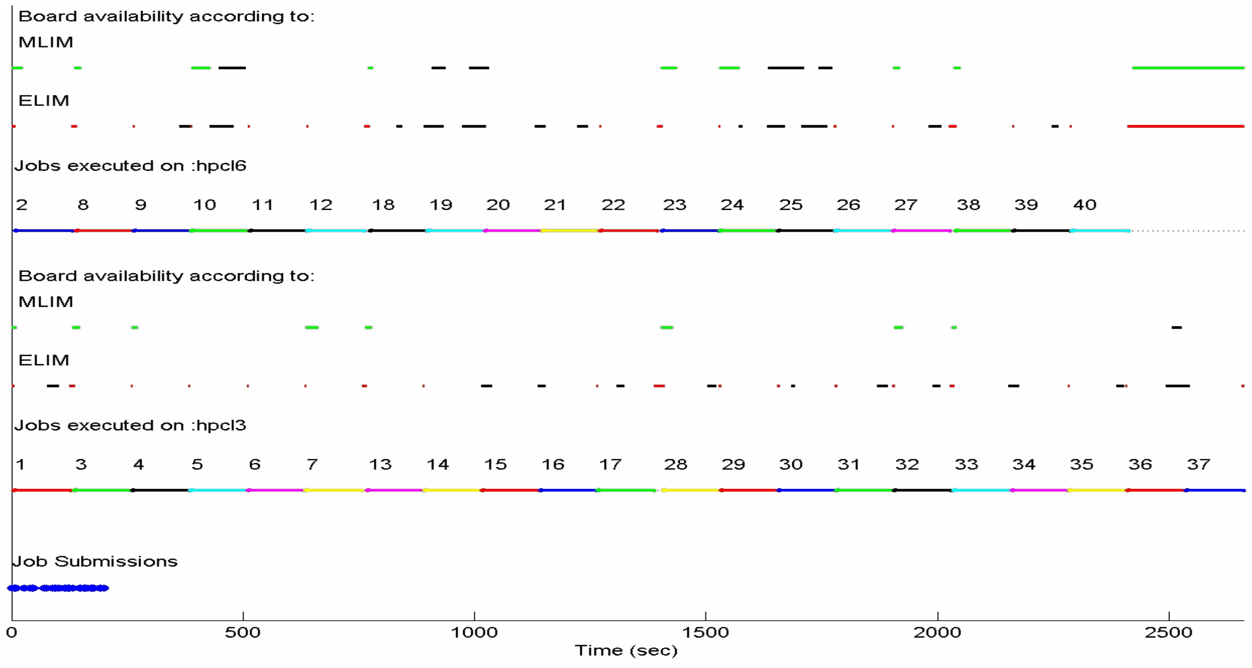

Figure. 9 Utilization of machines in Experiment 3, Iteration 1

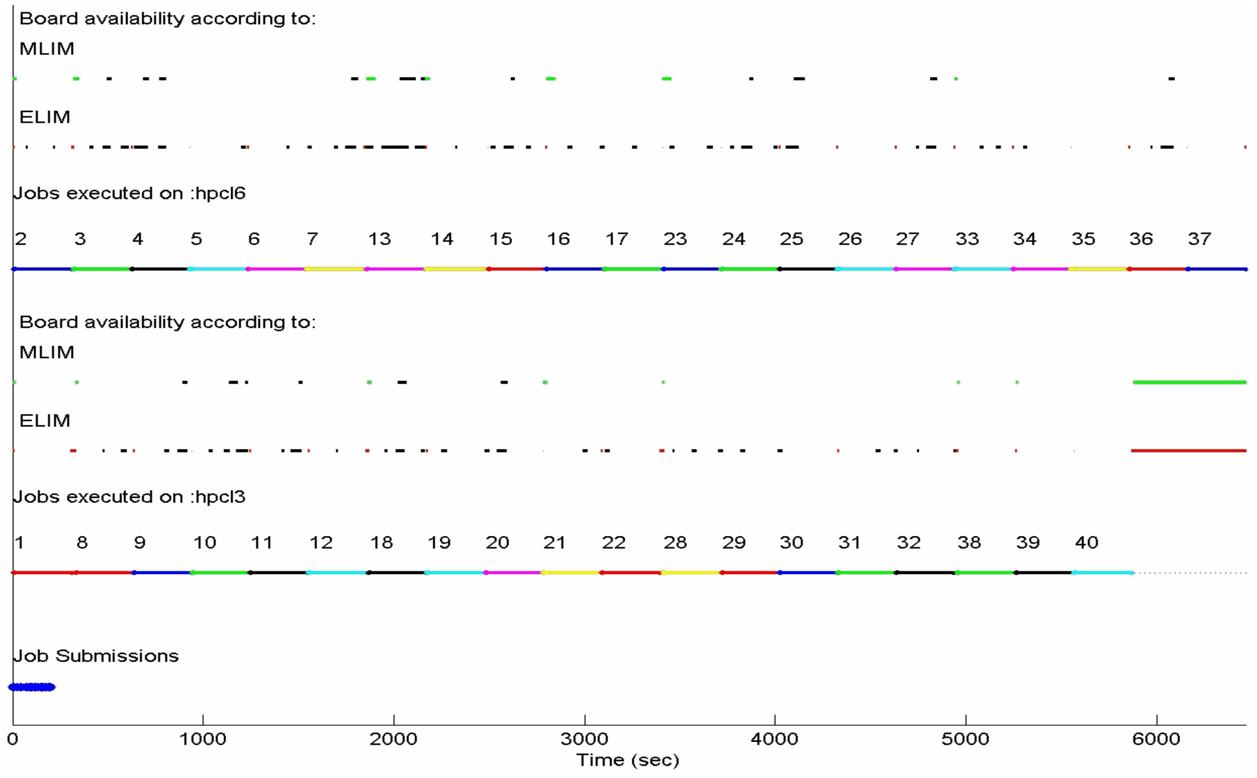

Figure. 10 Utilization of machines in Experiment 4, Iteration 1

6. Conclusions

An extension of LSF, supporting SLAAC-1V FPGA accelerator boards was developed and experimentally tested in a testbed consisting of Windows and Linux workstations. The utilization of the extended system has been improved to reach up to 96%.

References

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

[2] 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

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

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

[5] 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.

[6] 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.

[7] A. V. Staicu, J. R. Radzikowski, K Gaj, N. Alexandridis, and T. El-Ghazawi, "Effective Use of Networked Reconfigurable Resources," Proc. 2001 MAPLD Int. Conf., Laurel, Maryland, Sep. 2001

[8] M. Jones, P. Athanas et al. "Implementing an API for Distributed Adaptive Computing Systems," in *IEEE Workshop on Field-Programmable Custom Computing Machines*, pages 222-230, Napa Valley, CA, April 1999.

[9] B. Schott, S. Crago, et al. "Reconfigurable Architectures for System-Level Applications of Adaptive Computing," In VLSI Design: Special Issue on Reconfigurable Computing, pages 265-280, Volume 10, Number 3, 2000