Job Scheduling for the BlueGene/L System

Elie Krevat[1], José G. Castaños[2], and José E. Moreira[2]

[1] Massachusetts Institute of Technology, Cambridge, MA 02139-4307
krevat@mit.edu

[2] IBM T. J. Watson Research Center, Yorktown Heights, NY 10598-0218
{castanos,jmoreira}@us.ibm.com

Abstract. Cellular architectures with a toroidal interconnect are effective at producing highly scalable computing systems, but typically require job partitions to be both rectangular and contiguous. These restrictions introduce fragmentation issues which reduce system utilization while increasing job wait time and slowdown. We propose to solve these problems for the BlueGene/L system through scheduling algorithms that augment a baseline first come first serve (FCFS) scheduler. Our analysis of simulation results shows that migration and backfilling techniques lead to better system performance.

1 Introduction

BlueGene/L (BG/L) is a massively parallel cellular architecture system. 65,536 self-contained computing nodes, or *cells*, are interconnected in a three-dimensional toroidal pattern [7]. While toroidal interconnects are simple, modular, and scalable, we cannot view the system as a flat, fully-connected network of nodes that are equidistant to each other. In most toroidal systems, job partitions must be both rectangular (in a multidimensional sense) and contiguous. It has been shown in the literature [3] that, because of these restrictions, significant machine fragmentation occurs in a toroidal system. The fragmentation results in low system utilization and high wait time for queued jobs.

In this paper, we analyze a set of scheduling techniques to improve system utilization and reduce wait time of jobs for the BG/L system. We analyze two techniques previously discussed in the literature, backfilling [4,5,6] and migration [1,8], in the context of a toroidal-interconnected system. Backfilling is a technique that moves lower priority jobs ahead of other higher priority jobs, as long as execution of the higher priority jobs is not delayed. Migration moves jobs around the toroidal machine, performing on-the-fly defragmentation to create larger contiguous free space for waiting jobs.

We conduct a simulation-based study of the impact of those techniques on the system performance of BG/L. We find that migration can improve maximum system utilization, while enforcing a strict FCFS policy. We also find that backfilling, which bypasses the FCFS order, can lead to even higher utilization and lower wait times. Finally, we show that there is a small benefit from combining backfilling and migration.

2 Scheduling Algorithms

This section describes four job scheduling algorithms that we evaluate in the context of BG/L. In all algorithms, arriving jobs are first placed in a queue of waiting jobs, prioritized according to the order of arrival. The scheduler is invoked for every job arrival and job termination event, and attempts to schedule new jobs for execution.

First Come First Serve (FCFS). For FCFS, we adopt the heuristic of traversing the waiting queue in order and scheduling each job in a way that maximizes the largest free rectangular partition left in the torus. If we cannot fit a job of size p in the system, we artificially increase its size and retry. We stop when we find the first job in the queue that cannot be scheduled.

FCFS With Backfilling. Backfilling allows a lower priority job j to be scheduled before a higher priority job i as long as this reschedule does not delay the estimated start time of job i. Backfilling increases system utilization without job starvation [4,9]. It requires an estimation of job execution time. Backfilling is invoked when FCFS stops because a job does not fit in the torus and there are additional jobs in the waiting queue. A reservation time for the highest-priority job is then calculated, based on the worst case execution time of jobs currently running. If there are additional jobs in the waiting queue, a job is scheduled out of order as long as it does not prevent the first job in the queue from being scheduled at the reservation time.

FCFS With Migration. The migration algorithm rearranges the running jobs in the torus in order to increase the size of the maximal contiguous rectangular free partition, counteracting the effects of fragmentation. The migration process is undertaken immediately after the FCFS phase fails to schedule a job in the waiting queue. Running jobs are organized in a queue of migrating jobs sorted by size, from largest to smallest. Each job is then reassigned a new partition, using the same algorithm as FCFS and starting with an empty torus. After migration, FCFS is performed again in an attempt to start more jobs in the rearranged torus.

FCFS with Backfilling and Migration. Since backfilling and migration are independent scheduling concepts, an FCFS scheduler may implement both of these functions. First, we schedule as many jobs as possible via FCFS. Next, we rearrange the torus through migration to minimize fragmentation, and then repeat FCFS. Finally, the backfilling algorithm from Scheduler 2 is performed.

3 Experiments

We used an event-driven simulator to process actual job logs of supercomputing centers. The results of simulations for all four schedulers were then studied to determine the impact of their respective algorithms. The BG/L system is organized as a $32 \times 32 \times 64$ three-dimensional torus of nodes (cells). The unit of allocation for job execution in BG/L is a 512-node ensemble organized in an $8 \times 8 \times 8$ configuration. Therefore, BG/L behaves as a $4 \times 4 \times 8$ torus of these *supernodes*. We use this supernode abstraction when performing job scheduling for BG/L. That is, we treat BG/L as a machine with 128 (super)nodes.

A job log contains information on the arrival time, execution time, and size of all jobs. Given a torus of size N, and for each job j the arrival time t_j^a, execution time t_j^e and size s_j, the simulation produces values for the start time t_j^s and finish time t_j^f of each job. These results were analyzed to determine the following parameters for each job: (1)

wait time $t_j^w = t_j^s - t_j^a$, (2) response time $t_j^r = t_j^f - t_j^a$, and (3) bounded slowdown $t_j^{bs} = \frac{\max(t_j^r, \Gamma)}{max(t_j^e, \Gamma)}$ for $\Gamma = 10$ s. The Γ term appears according to recommendations in [4], because jobs with very short execution time may distort the slowdown.

Global system statistics are also determined. Let the simulation time span be $T = \max_{\forall j}(t_j^f) - \min_{\forall k}(t_k^a)$. We then define system utilization (also called *capacity utilized*) as $w_{\text{util}} = \sum_{\forall j} \frac{s_j t_j^e}{TN}$. Similarly, let $f(t)$ denote the number of free nodes in the torus at time t and $q(t)$ denote the total number of nodes requested by jobs in the waiting queue at time t. Then, the total amount of unused capacity in the system, w_{unused}, is defined as $w_{\text{unused}} = \int_{\min(t_j^a)}^{\max(t_j^f)} \max(0, f(t) - q(t))dt$. This parameter is a measure of the work unused by the system because there is a lack of jobs requesting free nodes. The balance of the system capacity is lost despite the presence of jobs that could have used it. The lost capacity in the system is then derived as $w_{\text{lost}} = 1 - w_{\text{util}} - w_{\text{unused}}$.

We performed experiments on 10,000-job segments of two job logs obtained from the *Parallel Workloads Archive* [2]. The first log is from NASA Ames's 128-node iPSC/860 machine (from the year 1993). The second log is from the San Diego Supercomputer Center's (SDSC) 128-node IBM RS/6000 SP (from the years 1998-2000). In the NASA log, job sizes are always powers of 2. In the SDSC log, job sizes are arbitrary. Using these two logs as a basis, we generated logs of varying workloads by multiplying the execution time of each job by a constant coefficient.

Figure 1 presents a plot of average job bounded slowdown (t_j^{bs}) $\times$ system utilization (w_{util}) for each of the four schedulers considered and each of the two job logs. (B+M is the backfilling and migration scheduler.) We also include results from the simulation of a fully-connected (*flat*) network. This allows us to assess how effective the schedulers are in overcoming the difficulties imposed by a toroidal interconnect. The overall shapes of the curves for wait time are similar to those for bounded slowdown. The most significant performance improvement is attained through backfilling, for both the NASA and SDSC logs. Also, for both logs, there is a certain benefit from migration, whether combined with backfilling or not.

With the NASA log, all four schedulers provide similar average job bounded slowdown for utilizations up to 65%. The FCFS and Migration schedulers saturate at about 77% and 80% utilization respectively. Backfilling (with or without migration) allows utilizations above 80% with a bounded slowdown of less than a hundred. We note that migration provides only a small improvement in bounded slowdown for most of the utilization range. In the NASA log, all jobs are of sizes that are powers of 2, which results in a good packing of the torus. Therefore, the benefits of migration are limited.

With the SDSC log, the FCFS scheduler saturates at 63%, while the stand-alone Migration scheduler saturates at 73%. In this log, with jobs of more varied sizes, fragmentation occurs more frequently. Therefore, migration has a much bigger impact on FCFS, significantly improving the range of utilizations at which the system can operate. However, we note that when backfilling is used there is again only a small benefit from migration, more noticeable for utilizations between 75 and 85%.

Migration by itself cannot make the results for a toroidal machine as good as those for a flat machine. For the SDSC log, in particular, a flat machine can achieve better than 80% utilization with just the FCFS scheduler. However, the backfilling results are closer

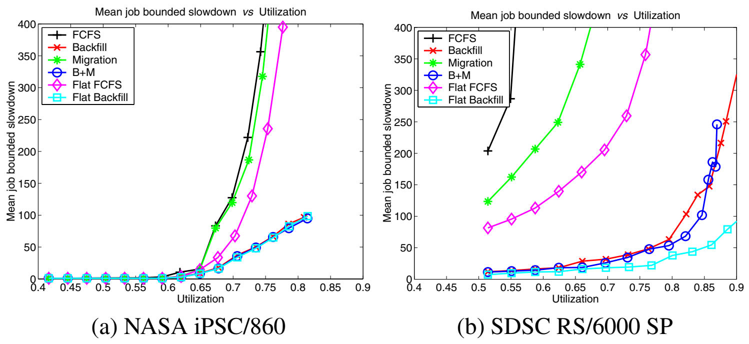

Fig. 1. Mean job bounded slowdown *vs* utilization for the NASA and SDSC logs, comparing toroidal and flat machines.

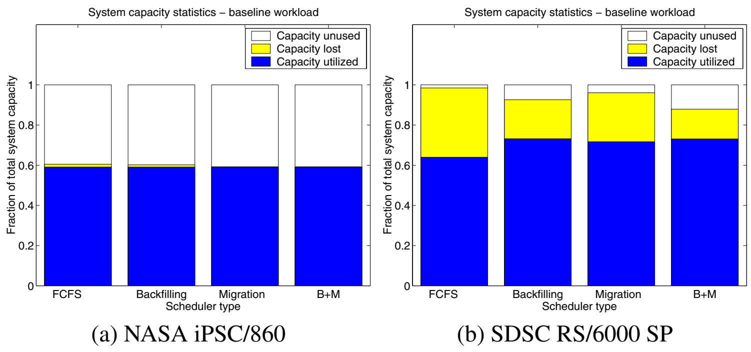

Fig. 2. Capacity utilized, lost, and unused as a fraction of the total system capacity.

to each other. For the NASA log, results for backfilling with migration in the toroidal machine are just as good as the backfilling results in the flat machine. For the SDSC log, backfilling on a flat machine does provide significantly better results for utilizations above 85%.

The results of system capacity utilized, unused capacity, and lost capacity for each scheduler type and both job logs (scaling coefficient of 1.0) are plotted in Figure 2. The utilization improvements for the NASA log are barely noticeable – again, because its jobs fill the torus more compactly. The SDSC log, however, shows the greatest improvement when using B+M over FCFS, with a 15% increase in capacity utilized and a 54% decrease in the amount of capacity lost. By themselves, the Backfill and Migration schedulers each increase capacity utilization by 15% and 13%, respectively, while decreasing capacity loss by 44% and 32%, respectively. These results show that B+M is significantly more effective at transforming lost capacity into unused capacity.

4 Related and Future Work

The topics of our work have been the subject of extensive previous research. In particular, [4,5,6] have shown that backfilling on a flat machine like the IBM RS/6000 SP is an effective means of improving quality of service. The benefits of combining migration and gang-scheduling have been demonstrated both for fully connected machines [10] and toroidal machines like the Cray T3D [3]. This paper applies a combination of backfilling and migration algorithms, exclusively through space-sharing techniques, to improve system performance on a toroidal-interconnected system. As future work, we plan to study the impact of different FCFS scheduling heuristics for a torus. We also want to investigate time-sharing features enabled by preemption.

5 Conclusions

We have investigated the behavior of various scheduling algorithms to determine their ability to increase processor utilization and decrease job wait time in the BG/L system. We have shown that a scheduler which uses only a backfilling algorithm performs better than a scheduler which uses only a migration algorithm, and that migration is particularly effective under a workload which produces a large amount of fragmentation. We show that FCFS scheduling with backfilling and migration shows a slight performance improvement over just FCFS and backfilling. Backfilling combined with migration converts significantly more lost capacity into unused capacity than just backfilling.

References

1. D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. **A worldwide flock of Condors: Load sharing among workstation clusters**. *Future Generation Computer Systems*, 12(1):53–65, May 1996.
2. D. G. Feitelson. **Parallel Workloads Archive**. `http://www.cs.huji.ac.il/labs/parallel/workload/index.html`.
3. D. G. Feitelson and M. A. Jette. **Improved Utilization and Responsiveness with Gang Scheduling**. In *IPPS'97 Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1291 of *Lecture Notes in Computer Science*, pages 238–261. Springer-Verlag, 1997.
4. D. G. Feitelson and A. M. Weil. **Utilization and predictability in scheduling the IBM SP2 with backfilling**. In *12th International Parallel Processing Symposium*, April 1998.
5. D. Lifka. **The ANL/IBM SP scheduling system**. In *IPPS'95 Workshop on Job Scheduling Strategies for Parallel Processing*, volume 949 of *Lecture Notes in Computer Science*, pages 295–303. Springer-Verlag, April 1995.
6. J. Skovira, W. Chan, H. Zhou, and D. Lifka. **The EASY-LoadLeveler API project**. In *IPPS'96 Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science*, pages 41–47. Springer-Verlag, April 1996.
7. H. S. Stone. **High-Performance Computer Architecture**. Addison-Wesley, 1993.
8. C. Z. Xu and F. C. M. Lau. **Load Balancing in Parallel Computers: Theory and Practice**. Kluwer Academic Publishers, Boston, MA, 1996.
9. Y. Zhang, H. Franke, J. E. Moreira, and A. Sivasubramaniam. **Improving Parallel Job Scheduling by Combining Gang Scheduling and Backfilling Techniques**. In *Proceedings of IPDPS 2000*, Cancun, Mexico, May 2000.
10. Y. Zhang, H. Franke, J. E. Moreira, and A. Sivasubramaniam. **The Impact of Migration on Parallel Job Scheduling for Distributed Systems**. In *Proceedings of the 6th International Euro-Par Conference*, pages 242–251, August 29 - September 1 2000.