Conservative Backfilling Job Rejection Policy

Phd Std. Fernando MENDONCA, Dr. Denis Trystram
LIG - Laboratoire d'Informatique de Grenoble

ABSTRACT

Scheduling a job on a high performance PC or any other distributed system is usually done by giving this job a small part of processors and; is done by using First Come First Serve – a fair enough policy.

Each job has to wait for its turn to be executed and has its own running time. If a small job is executed after a large job then the small job has to wait for a longer period of time. This time can be stretched over weeks while the running time would be just in seconds.

This increased stretched time generates a great impact on average stretched time of the scheduler. In addition, this job will reserve some processors while leaving other processors unusable. If this job is rejected then it may reduce the stretch time and improve the utilization.

CONTACT

Waqas IMTIAZ
LIG - Laboratoire d'Informatique de Grenoble
Email: waqas.imtiaz@ensimag.grenoble-inp.fr
Website: https://www.liglab.fr/

INTRODUCTION

The scheduling scheme used on most distributed memory parallel supercomputers is variable partitioning, meaning that each job receives a partition of the machine with its desired number of processors [5]. Such partitions are allocated on a first-come first-serve (FCFS) manner to submitted jobs. But, this approach suffers from fragmentation, where free processors cannot meet the requirements of the next job and therefore remain idle until additional ones become available. As a result, system utilization is typically in the range of 50-80 percent.

It is well known that the best solutions for this problem are to use dynamic partitioning or gang scheduling. However, these schemes have practical limitations. The only efficient and widely used implementation of gang scheduling was the one on the CM-5 Connection Machine. Other commercial implementations are too coarse-grained for real interactive support and do not enjoy much use. To the best of our knowledge, dynamic partitioning has not been implemented on production machines at all.

METHODOLOGY

The experiments are based on an event-based simulation, where events are job arrival and termination. Upon arrival, the scheduler is informed of the number of processors the job needs and its estimated runtime.

It can then either start the job's simulated execution or place it in a queue. Upon a job termination, the scheduler is notified and can schedule other queued jobs on the freed processors The runtime of jobs is part of the input to the simulation but is not given to the scheduler. It is assumed that the runtime does not depend in any way on scheduling decisions.

Traces of the jobs submitted to the following super-Computers:

1.CTC: the Cornell Theory Center 512-node IBM SP2 (79,296 jobs from July 1996 to May 1997)
1.KTH: the Swedish Royal Institute of Technology 100-node IBM SP2 (28,490 jobs from October 1996 to August 1997)

EXPERIMENTS

Traces are simulated using the exact data provided withpossible modifications as noted (e.g., to check the impact of different estimates of runtime). For models, the load on the simulated system is modified by multiplying the inter-arrival times by a certain factor. For example, if, by default, the model produces a load of 0.688, we can create a higher load of 0.8 by multiplying all interarrival times by a factor of 0.68. Using different factors enables the functional relationship of performance on load to be measured.

When using models, 90 percent confidence intervals for the response time were calculated using the batch means method. Each batch size was 3,333 job terminations, with the first batch discarded to account for warmup effects (for the Jann model, batches were just under 1,000 jobs).

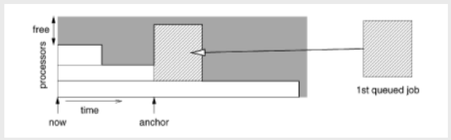

Figure 1. 1st queued job.

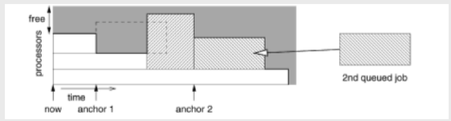

Figure 2. 2nd queued job.

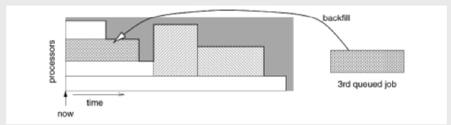

Figure 3. 3rd queued job (not-possible).

RESULTS

The results of simulations using the two models are presented in Fig 4. They indicate that the relative performance of EASY and conservative backfilling depends on the workload used and on the performance metric! Specifically, according to the Feitelson model (F), both schemes are practically identical.

According to the Jann model(J), EASY has better (lower) average response times under high loads, but slightly worse (higher) bounded slowdown.

The results for the actual workload traces are reported for each month individually so as to create multiple data points for somewhat different load conditions. They are shown in Table 1. Again, there is a difference between the different workloads and metrics.

In general, the SP2 workloads favor the EASY backfilling over conservative backfilling.

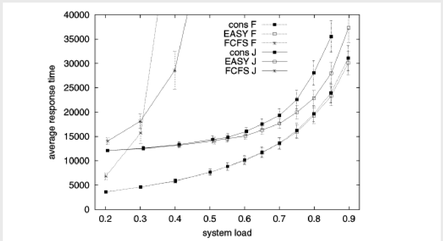

Chart 1. comparison between different policies.

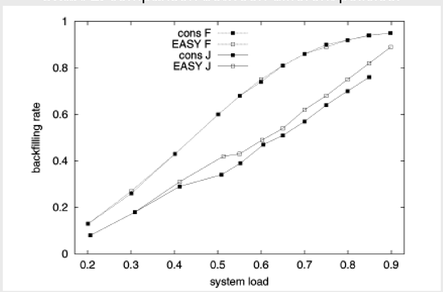

Chart 2. Backfilling rate for cons & EASY policies.

DISCUSSION

o summarize, the simulation results are somewhat inconclusive and depend on the workload and metric being used.

For most of the combinations checked, the performance of the EASY backfilling algorithm was better than that of conservative backfilling. However, in some cases, the two algorithms seemed to provide similar performance and, in one case, conservative was better than EASY.

In order to study user runtime estimates we used workload data from the three IBM SP2 installations mentioned above. The workload data comes in the form of a log of all jobs executed on the machine during a certain period.

The information on each job includes the estimated runtime provided by the user upon submittal and the time the job actually ran. The following is based on a job-by-job comparison of these two times.

CONCLUSIONS

Backfilling is advantageous because it provides improved responsiveness for short jobs combined with no starvation for long ones.

This is done by making processor reservations for the large jobs and then allowing short jobs to leapfrog them if they are expected to terminate in time.

The expected termination time is based on user input

REFERENCES

1. D. Das Sharma and D.K. Pradhan, ™Job Scheduling in Mesh Multicomputers,∫ Proc. Int'l Conf. Parallel Processing, vol. II, pp. 251-258, Aug. 1994.
2. A.B. Downey, ™Using Queue Time Predictions for Processor Allocation,∫ Job Scheduling Strategies for Parallel Processing, D.G. Feitelson and L. Rudolph, eds., pp. 35-57, Springer-Verlag, 1997.
3. A.B. Downey and D.G. Feitelson, ™The Elusive Goal of Workload Characterization,∫ Performance Evaluation Review, vol. 26, no. 4, pp. 14-29, Mar. 1999.