Asian Journal of Information Technology 7 (9): 403-407, 2008
ISSN: 1682-3915

An Approach for Parallel Job Scheduling Using Supple Algorithm

S.V. Sudha and K. Thanushkodi
Deprtment of Information Technology,
Coimbatore Institute of Engineering and Information Technology, Coimbatore, India

Abstract: Supple Scheduling is a Scheduling methodology for medium grain applications. The medium grain applications consist of many processes running on different Processors. A way of characterizing a parallel system is to consider the synchronization granularity or frequency of synchronization between processes in a application. A Single application can be effectively implemented as a collection of threads. In this case, the potential parallelism of an application must be explicitly specified by the Programmer. Typically, there will need to be rather a high degree of coordination and interaction among the threads of an application, leading to a medium grain-level of synchronization. The various threads of an application interact so frequently, scheduling decisions concerning one thread may affect the performance of the entire application. The Supple algorithm is fully implemented and compared the results with various other scheduling algorithms like First Come First Served, Gang Scheduling, Flexible Co scheduling. This study describes in detail the implementation of Supple and its performance evaluation with a original workload log LLNL-Thunder-2007-0.

Key words: Gang scheduling, first come first served scheduling, flexible co scheduling, performance metrics, supple algorithm

INTRODUCTION

Large Scale parallel machines are essential to meet the needs of demanding applications at supercomputing environments, such as Lawrence Livermore (LLNL), Los Alamos (LANL) and Sandia National Laboratories (SNL). With the increasing emphasis on computer simulation as an engineering and scientific tool, the load on such systems is expected to become quite high in the near future. As a result, it is imperative to provide effective scheduling strategies to meet the desired quality of service parameters from both user and system perspectives (Kerbyson, 1999; Dan Tsafrir, 2007). We would like to reduce response and wait time for a job, minimize the slowdown that a job with the new algorithm.

Effective Job Scheduling schemes are important for parallel system in order to improve system metrics like utilization and user metrics like turn around time. Most of the studies in literature have reported these metrics averaged over all jobs of simulated traces. When compared different scheduling strategies (Eitan *et al*., 2005), many studies have concluded that the relative effectiveness of different schemes often depends on the job mix (Eitan *et al*., 2005). In order to gain greater insight into the relative effectiveness of different scheduling strategies, we have taken the original log for the medium grain application (Kettimuthu, 2002; Lee, 1997).

Scheduling of processes onto processors of a parallel machine has always been an important and challenging area of research. Its importance stems from the impact of the scheduling discipline on the throughput and response time of a system.

Parallel Job Scheduling has been widely studied in the past (Nissimov, 2007; Anglano, 2000). The simplest way to schedule jobs is to use First Come First Served (FCFS) policy. This approach suffers from low utilization. Gang Scheduling is a scheduling algorithm that schedules related thread or processes to run simultaneously on different processors. Gang Scheduling requires that the schedule of communicating processes be precomputed which complicates the co scheduling of application. Flexible Co scheduling concentrates mainly on the Fine and Coarse grain applications and the algorithm saturates at heavy loads.

The main objectives of this study, are to define, propose, develop and implement the Supple Algorithm using simulation. The implementation is done through simulation.

Corresponding Author: S.V. Sudha, Deprtment of Information Technology,
Coimbatore Institute of Engineering and Information Technology, Coimbatore, India

MATERIALS AND METHODS

Supple algorithm: The algorithm concentrates on the medium grain application (Tsafrir, 2007; Nissimov, 2007) and the entire application workload is being divided in to several slot. The jobs are equally placed in the slots. A Scheduling Matrix is developed for the algorithm implementation and a multiprogramming level of 10 is considered for experiment.

In the slots taken, every 5 jobs are considered as Primary jobs and being allocated in the matrix. Now the next 5 jobs in the workload are taken and allocated in the primary slots which are not used by the primary jobs. The entire process is continued until all jobs are completed in the slot and work continue until all slots are completed. The sample Scheduling Matrix is shown in the (Table 1).

Scheduling of parallel jobs is usually viewed in terms of a matrix called Scheduling Matrix called Ousterhout Matrix that defines the tasks executing on each processor and each time slice (Srinivasan, 2002; Antonopoulos, 2001). The scheduling matrix is as shown Table 1, where each task of a job is represented as pij where pij represents the jth task of the job i. Each row of the matrix defines a 10 Processor virtual machine. All tasks of a parallel job are always co scheduled to run concurrently.

Workload characteristics: The simulation studies were performed using the collection of workload log available from Feitelson's archive. This log contains several months worth of accounting records from a large Linux cluster called Thunder installed at Lawrence Livermore National Lab. This specific cluster has 1024 nodes, each with 4 processors, for a total of 4096 processors. At the time that this log was recorded, Thunder was considered a capacity computing resource, meaning that it was intended for running large numbers of smaller to medium jobs. This is in contrast with the newer Atlas cluster, which is a capability machine, used for running large parallel jobs that cannot execute on lesser machines. The original log is available as LLNL-Thunder-2007-0. This file contains one line per completed job in the Slurm format. The fields are JobID, start time, end time etc.

The difficulty of scheduling parallel resources is deeply interwoven with the inherent variability in parallel workloads. To propose a robust policy, we first examine real parallel workloads of productionsystems (Lublin, 2003; Franchtenberg *et al*., 2002).

Performance metric: The synthetic workload generated Feitelson's archive are used as input to the simulation of various scheduling strategies. We monitor the following parameters the arrival time, start time, execution time, finish time etc. Different Scheduling algorithms have different properties and may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms (Feitelson, 1998; Franchtenberg *et al*., 2003). Many criteria have been suggested for scheduling algorithms. The criteria includes the following:

Table 1: The scheduling matrix

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
P11	p112	p13	p14	p15					
p21	p22	p23	p24	P25	p25				
p31	p32	p33	p34	p35					
p41	p42	p43	p44						
p51	p52	p53	p54	p55	p56	p57			

Mean utilization: We want to keep the CPU as busy as possible. CPU Utilization may range from 0-100%. In a real system, it should range from 40% (for a lightly loaded system) to 90% (for a heavily loaded system).The mean utilization is the ratio of cpu busy time to the number of processors multiplied with Total time for execution.

$$\text{Mean utilization} = \frac{\sum \text{CPU Busy time}}{\text{Number of processors} \times \text{total time}} \quad (1)$$

Mean response time: In an interactive system,Turnaround time may not be the best criteria. Often, a process can produce some output fairly early,and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure is called response time (Eitan *et al*., 2005).

$$\text{Mean response time} = \frac{\sum \text{Job finish time} - \text{Job submit time}}{\text{Number of jobs}} \quad (2)$$

Mean reaction time: The mean job reaction time defined as the mean time interval between the submission and the start of the job.

$$\text{Mean reaction time} = \frac{\sum \text{Job start time} - \text{Job submit time}}{\text{Number of jobs}} \quad (3)$$

Mean slowdown: Mean Slowdown is the sum of jobs response times divided by the job's execution times. This metric emerges as a solution to normalize the high variation of the jobs response time.

$$\text{Mean slow down} = \frac{\sum \text{Job response time/Job execution time}}{\text{Number of jobs}} \quad (4)$$

Turn around time: From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turn around time. Turn around time is the sum of periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O (Eitan *et al*., 2005).

Waiting time: The scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

Scheduling strategy: We now describe and analyze in detail the supple scheduling algorithm in our work. We start with the simple Scheduling algorithm First Come First Serve, executed with the log specified, then the same log with the Gang scheduling and the Flexible co scheduling (Nikolopoulos and Polychronopoulos, 2003). The algorithm analyze the medium grain application indetail. The medium grain applications consists of synchronization at a gross level and this kind of situation is easily handled on a multiprogrammed uniprocessor. The synchronization granularity and processes plays an important role in the performance of the application so, the supple algorithm carefully examines the jobs and schedules neatly so that the performance level reaches a greater height.

Supple algorithm

For (all jobs in a queue)

- Sort the jobs in accordance with the submit time.
- Divide the total number of jobs in to 1000 slots.
- Each job is given a time quantum to execute in each processor so that a strict global Round Robin is followed.
- While (Slots available).
  Scheduling flag = true
  Scheduling count=0
  - While(Number of jobs available in the slot)
    - If Scheduling flag.
      - Primary Jobs=5 Jobs in the slot.
      - Assign the jobs in the Scheduling Matrix.
      - Primary Slots = ideal Processors not used by primary jobs.
      - Scheduling Flag is off.
      - Scheduling Count is incremented.
    - Else
      - Secondary Jobs=next 5 Jobs in the slots
      - If Primary slots available the assign secondary slots in the primary slots.
      - Scheduling flsg is on.
      - Scheduling Count is incremented.
  - If scheduling count is 2.
    - Schedule the jobs to the processor.
    - Scheduling Count is 0.

Experimental results: In this study, we present and analyze the performance of Supple Algorithm. First, for each metric, we present the results by simulation. All simulators are written in Java. The performance analysis graphs in Fig. 1-6 shows that the average waiting time, mean response time, turnaround time, mean reaction time,

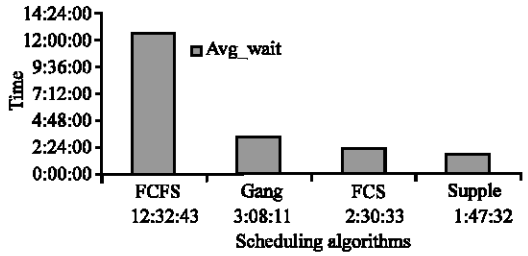

Fig. 1: Average waiting time considering the scheduling algorithms

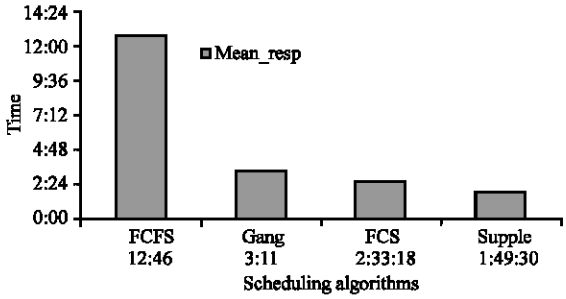

Fig. 2: Mean response time considering the scheduling algorithms

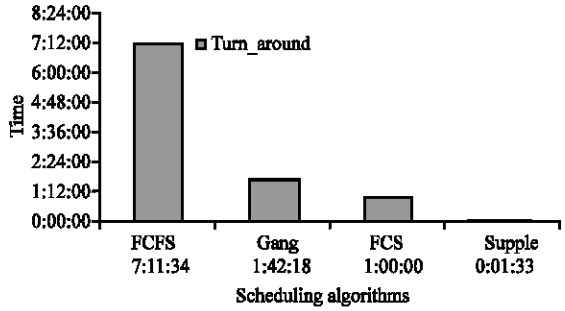

Fig. 3: Turn around time considering the scheduling algorithms

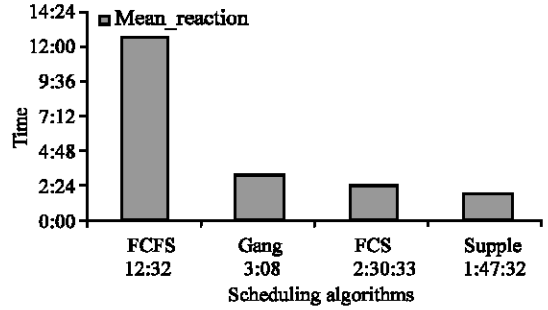

Fig. 4: Mean reaction time considering the scheduling algorithms

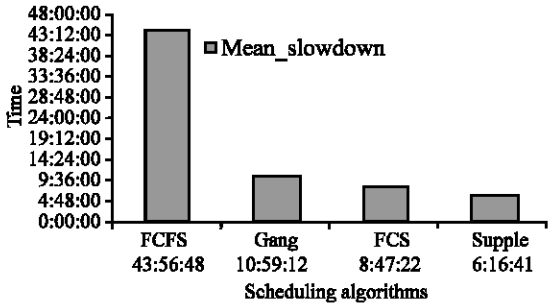

Fig. 5: Mean slow down considering the scheduling algorithms

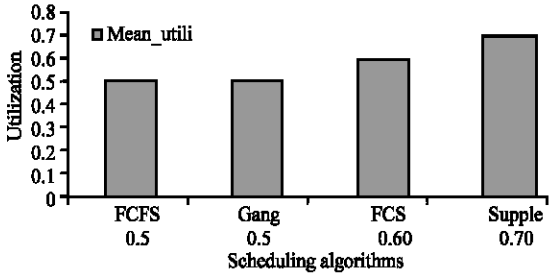

Fig. 6: Mean utilization considering the scheduling algorithms

mean slow down and mean utilization for the supple algorithm gives better results when compared to the scheduling strategies first come first served, gang scheduling and flexible co scheduling.

CONCLUSION

We present a new Scheduling methodology Supple Algorithm for Medium Grain Applications. The Algorithm concentrates mainly on the frequency of synchronization between the processes of the application and the performance is improved and the supple algorithm is compared with Fist Come First Served, Gang Scheduling and Flexible Co scheduling. The workload of 30,000 jobs is considered for the calculation.

The first come first served algorithm executes the work and the overall running time is 7 h and 11 min. Gang scheduling executes at the time 1 h and 42 min, Flexible Co scheduling executes at 1 h and the supple algorithm executes at 1 min and 33 sec. The drawback is overcome with the supple algorithm.

REFERENCES

Anglano, C., 2000. A comparative evaluation of implicit coscheduling strategies for network of workstations. Proc. Ninth Int'l Symp. High Performance Distributed Computing, pp: 221-228.

Antonopoulos, C.D., D.S. Nikolopoulos and T.S. Papatheodorou, 2001. Informing algorithms for efficient scheduling of synchronizing threads on multiprogrammed SMPs. Proc Int'l Conf Parallel Processing, pp: 123-130.

Eitan, F., D.G. Feitalson, F. Petrini and J. Fernandez, 2005. Adaptive parallel job scheduling with flexible coscheduling. IEEE Trans. Parallel and Distributed Sys., 16 (11): 1066-1077.

Feitelson, D.G. and L. Rudolph, 1998. Metric and benchmarking for parallel job scheduling. Job Scheduling Strategies for parallel Processing, pp: 1-24.

Frachtenberg, E., F. Petrini, J. Fernandez, S. Pakin and S. Coll, 2002. STROM: Lightning-Fast Resource Management. Proc. Supercomputing conf.

Franchtenberg, E., D.G. Feitelson, J. Fernandez-Peinador and F. Petrini, 2003. Parallel Job Scheduling under Dynamic Workloads. Proc. Ninth Workshop Job Scheduling Strategies for Parallel Processing, pp: 208-227.

Kerbyson, D., H. Alme, A. Hoisie, F. Petrini, H. Waserman and Miggints, 1999. Predictive Performance and Scalability Modeling of a Large -Scale SMP Clusters. Proceeding of the Symposium, Frontiers of Massively Parallel Computation.

Kettimuthu, R., V. Subramani, Srinivasan, T.B. Gopalsamy, D.K. Panda and P. Sadayappa, 2002. Selective preemption Strategies for Parallel Job Scheduling. Proceeding Int'l Conf Parallel Processing, pp: 55-71.

Lee, W., M. Frank, V. Lee, K. Mackenzie and L. Rudolph, 1997. Implication of I/O for gang scheduled workloads. Job Scheduling Strategies for Parallel Processing, pp: 215-237.

Lublin, U. and D.G. Feitelson, 2003. The workload of super computers. Modeling the characteristics of rigid jobs. J. Parallel and Distributed Computing, 63 (11): 1105-1122.

Nikolopoulos, D.S. and C.D. Polychronopoulos, 2003. Adaptive scheduling under memory constraints on non-dedicated computational farme. Future Generation Comput. Sys., 19 (4): 505-519.

Nissimov, A. and D.G. Feitelson, 2007. Probabilistic backfilling. 13th Workshop on Job Scheduling Strategies for Parallel Processing in Conjuction with 21st ACM International Conference on Super Computing, pp: 102-115.

Srinivasan, S., R. Kettimathu, V. Subramani and P. Sadayappan, 2002. Selective Reservation Strategies for Backfilling Job Scheduling. Job Scheduling Strategies for Parallel Processing, pp: 55-71.

Tsafrir, D., Y. Etsion and D.G. Feitelson, 2007. Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel and Distributed Syst., 18 (6): 789-803.