Journal of Biometrics & Biostatistics
ISSN: 2155-6180

Haq and Shah, J Biom Biostat 2015, 6:5
DOI: 10.4172/2155-6180.1000257

Research Article Open Access

Task Scheduling in Parallel Processing: Analysis

Abdul Haq* and Munam Ali Shah

Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan

Abstract

Task scheduling in parallel processing is a technique in which processes are assigned to different processors. Task scheduling in parallel processing use different types of algorithms and techniques which are used to reduce the number of delayed jobs. Now a days there are different kind of scheduling algorithms and techniques used to reduce the execution time of tasks. As task scheduling the NP-hard problem and no one can say the about the best algorithm proposed so in this paper we will review some of the task scheduling algorithms and other techniques.

Keywords: Task scheduling; Parallel processing; Algorithms and techniques

Introduction

Parallel processing is dividing the process into multiple processes and execute them concurrently by the use of more than one CPU or processor[1]. Before dividing the process it is checked whether the process is divisible or not, if it is not then the process is executed as a whole and if it is divisible then these processes can be mapped among the processors separately, after execution these processes are reassembled and finally the processed is completed as shown in Figure 1.

Parallel processing is used due to some reasons, i.e. it provide concurrency, save time, solve larger problems, maximize load balancing and make a good use of parallel hardware architecture [2]. In multiprocessor environment parallel processing has two kinds of processors heterogeneous and homogeneous , in heterogeneous the processors are of different kind of speed and cost while in homogenous there are same kind of processors in all perspective [3] as shown in Table 1. By adding extra processors it is possible to reduce the execution time of a task [4,5].

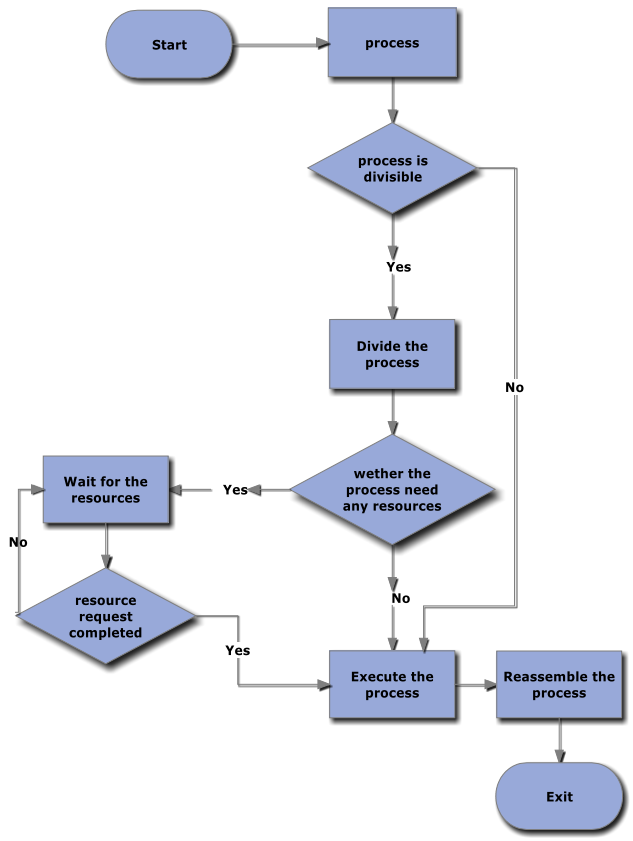

Figure 1: Phenomena of parallel processing.

Other than the environment, the parallel processing focuses on task execution and its speed. This is done by different kinds of task scheduling algorithms and techniques. The main objective of these algorithms are to minimize the overall execution time of the task and maximize the execution speed of the task [6]. The task scheduling is categorized in two section static scheduling and dynamic scheduling. In static scheduling the execution time, the limit and communication information of a task are fixed or pre-defined, In dynamic scheduling these information are not pre-defined till execution [7]. Static scheduling allows one process for one processor which results in reduction of process creation and termination overhead. Static scheduling takes smaller time of execution than dynamic scheduling [8,9].

There are different kinds of algorithms and techniques as follow. A graphical method is use to solve problems called graph theory approach and in that graph colouring is used in task scheduling and resource allocation. The second is Heuristic approach and the third one is mathematical based function and methods [10].

Homogenous Processor	Heterogeneous Processor
Same cores	Different Cores
Symmetric	Asymmetric
Off load of task is easy	Off load of task is complicated
At each CPU the operation is same	The operation at each CPU is different
Compatibility is better	Less compatible (Specialized for specific tasks) [4]

Table 1: Comparison of Homogenous and Heterogeneous Processors.

***Corresponding author:** Abdul Haq, Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan, Tel: 051-9247000-3; E-mail: Abdulhaqkhan49.uom@gmail.com

Received September 15, 2015; **Accepted** October 17, 2015; **Published** October 23, 2015

Citation: Haq A, Shah MA (2015) Task Scheduling in Parallel Processing: Analysis. J Biom Biostat 6: 257. doi:10.4172/2155-6180.1000257

This review is aimed to explore the scheduling algorithm and techniques for parallel processing. Which consist of gang scheduling and back filling. The rest of the paper is organized as follows. In the section gang scheduling and backfilling is reviewed.

Related Work

There are different scheduling techniques used in parallel processing and we are going to review it in the next section. The following are some scheduling algorithms presented in different literatures.

Gang scheduling

In gang scheduling time sharing is used among gangs and schedules the related processes to run concurrently on different processors [11]. This is used for Inter process communication so that the two processes communicate with each other at the same time. When the gang scheduling is not used over a group of processes then the overhead of context switch occurs [12].

Gang scheduling is related with co-scheduling i.e., co-scheduling not only run related processes concurrently but it also allows fragments, fragments run independently of the rest of gang i.e. they do not run concurrently with them [13].

Gang scheduling is represented in two dimensional matrixes known as Outer out matrix, the problem with gang scheduling occur when no job is assigned to a processor and in the meantime the processor is idle which leads to fragmentation [14]. Table 2 is shows Outerhout matrix in a machine with five processors [15].

The problem with gang scheduling occur when no job is assigned to a processor and in the meantime the processor is idle which leads to fragmentation. Several ideas are proposed to reduce fragmentation [16,17]. Some of the proposed ideas includes the use of backfilling with gang scheduling, the Gang EDF scheduling and paired Gang scheduling, etc. [14]. There are also other techniques proposed in gang scheduling as follows

Gang EDF (Earliest Deadline First): In this paper [12] the author proposed a new policy to gang scheduling called Gang EDF (Earliest Deadline First) scheduling which is just like classical Global ED: where the higher priorities are assigned to the jobs with earlier deadlines. But in Gang EDF he proposed a special limitation on available processors, where the number of ready jobs is chosen on the basis of Global EDF. In gang EDF the earliest deadline jobs are selected first and executed concurrently, and if there are some limitation on the processor due to which the job cannot start executing then on the basis of first fit heuristic it select the next job for execution. The author also presents a pseudo-code for the implementation of gang EDF.

Paired gang scheduling: The paper [14] is about paired gang scheduling where he minimize the problem of I/O-bounded jobs to increase the system performance. So to avoid this problem a time quantum is assigned to processes on the basis of their characteristics. As we already study the Outerhout matrix in Figure 1 two rows are selected in paired gang scheduling i.e. two different processes are assigned among the processers and the local scheduler schedules these processes. The author also measures the CPU utilization for the two different jobs i.e. if one process uses some I/O device while other uses CPU. So in this case the I/O activity and CPU utilization must be measured.

Processor1	Jb1	Jb2	Jb3	Jb5
Processor1	Jb1	Jb2	Jb3	Jb5
Processor1	Jb1	*	Jb4	Jb5
Processor1	*	*	Jb4	*
Processor1	*	*	Jb4	*

*processor is idle

Table 2: Outerhout matrix [11].

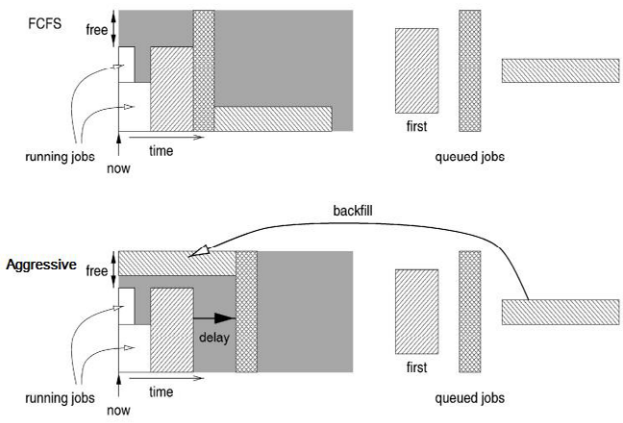

Figure 2: Aggressive backfilling technique [21].

Backfilling

Backfilling is scheduling technique in which the jobs are packed together in order to avoid fragmentation [18].

While using back filling can also defined the run time estimation so that the scheduler can predict the termination of a job and also when the next job will be executed which is already in the queue. The soul of backfilling is that it identifies the "holes" in the schedule and fit the small jobs in those holes. It is necessary for a scheduler with backfilling that it will move short jobs forward to improve responsiveness and utilization and to avoid starvation for large jobs [19].

Aggressive backfilling: The paper [20] is about using different algorithms to improve the backfilling scheduling. In this paper the aggressive backfilling is compared with conservative backfilling. As we discussed earlier in this paper that backfilling identify "holes", Conservative backfilling fill that holes by pushing small jobs if they don't delay other jobs that are queued then it goes forward in queue. While in aggressive backfilling those jobs takes reservation which are on the head of queue. The small jobs will be allowed to go forward only if they do not delay or affect the jobs on the head of the queue. The jobs are actually divided into two parts one is runtime for the jobs and the other is number of processors requested. The aggressive backfilling is shown in Figure 2.

The aggressive backfilling is compared to FCFS scheduling algorithm and shown with the help of the diagram [21].

Conservative backfilling: Mualem et al., [19] evaluated the conservative and easy backfilling techniques. The conservative backfilling is discussed in our paper earlier, the Figure 3 shows conservative backfilling.

The authors [22] compared many algorithms according to load balancing. For these comparisons we have two goals first is performance improvement and other is to maintain the service quality. Each assigned jobs to a processors required a specific time and when it takes

		Gang EDF Scheduling	Paired Gang Scheduling	Conservative Backfilling	Aggressive Backfilling
Advantages	Efficiency	High	-	High	High
	Performance	High	High	High	High
	Load Balancing	Maximum	Maximum	Average	-
	Turnaround Time	-	-	Average	Average
Limitations	Fragmentation	Yes	Yes	No	No
	Context Switching	High	High	Average	Average
	Job Reservation	No	No	Yes	Yes
	Running Time Estimation	No	No	Yes	No
Constraints	Queue	-	-	Priority	Priority
	Tasks	Independent	Independent	Dependent	Dependent

Table 3: Evaluation.

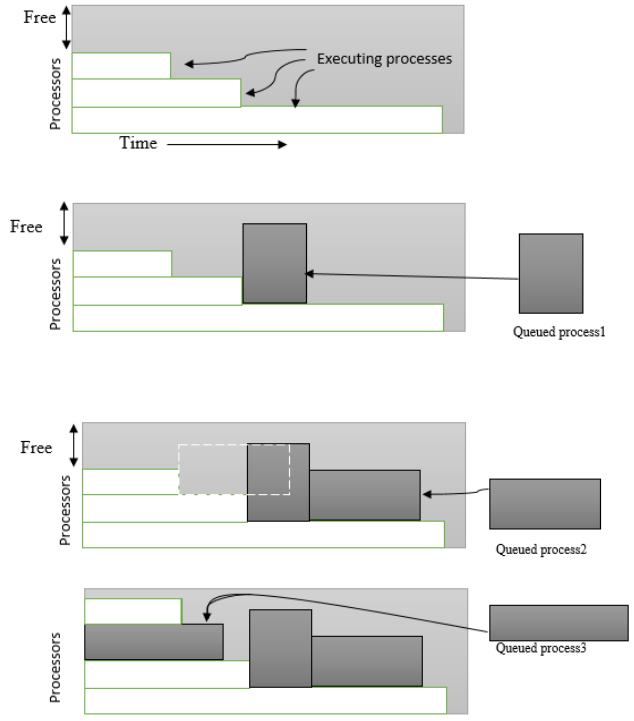

Figure 3: Conservative Backfilling [19].

longer time than that it is killed otherwise. The other algorithm called Flex algorithm use different approach that focus on how to optimize the whole queue not just the head job of the queue. Flex algorithm used slack to avoid starvation i.e. slack is given to each job on arrival and never wait longer than its slack.

Performance Evaluation

On the basis of related work section as we discussed previous work on these algorithms, we draw a performance evaluation table to compare these algorithms. Table 3 show the performance evaluation of these algorithms.

Conclusion

In our paper we study different scheduling algorithms and also show there performance evaluation. From the table it is clear that there are some advantages and limitations in specific constraints. In future work we will focus on having such algorithm that have all the advantages of these algorithms and try to minimize the limitation of that algorithm.

References

1. Rouse M (2015) What is parallel processing? - Definition from WhatIs.com.
2. Barney B (2015) Introduction to Parallel Computing.
3. Tamhane S, Kumar M (2008) Task scheduling on Heterogeneous Devices in Parallel Pervasive Systems {(P2S)}. 15th Annual IEEE International Conference on High Performance Computing.
4. Wolf M (2014) High-Performance Embedded Computing: Applications in Cyber-Physical Systems and Mobile Computing. Newnes.
5. Elnashar Al (2011) To Parallelize or Not to Parallelize, Speed up Issue. International Journal of Distributed and Parallel Systems (IJDPS) 2: 14-28.
6. Sharma M, Kaur H (2012) A Study of Bnp Parallel Task Scheduling Algorithms Metrics for Distributed Database System. International Journal of Distributed and Parallel Systems (IJDPS) 3: 157-166.
7. Cooper K, Crummey MJ, Sarkar V (2011) Languages and Compilers for Parallel Computing. Springer Berlin Heidelberg ,Berlin, Heidelberg.
8. Singh R (2014) Task Scheduling in Parallel Systems using Genetic Algorithm. International Journal of Computer Applications 108: 34-40.
9. Hagras T, Janecek J (2003) Static vs Dynamic List-Scheduling Performance Comparison. Acta Polytechnica 43: 16-21.
10. Riaz F, Ali KM (2011) Applications of Graph Theory in Computer Science. Computational Intelligence, Communication Systems and Networks 2: 142-145.
11. Corbalán J, Martorell X, Labarta J (2011) Improving Gang Scheduling through Job Performance Analysis and Malleability. Proceedings of the 15th International Conference on Supercomputing.
12. Kato K, Ishikawa Y (2009) Gang EDF scheduling of parallel task systems. Real-Time Systems Symposium.
13. Jiang Y, Shen X, Chen J, Tripathi R (2008) Analysis and approximation of optimal co-scheduling on chip multiprocessors.
14. Wiseman Y, Feitelson DG (2003) Paired gang scheduling. IEEE Transactions on Parallel and Distributed Systems 14: 581-592.
15. Zhang Y, Franke H, Moreira JE, Sivasubramaniam A (2000) Improving parallel job scheduling by combining gang scheduling and backfilling techniques. Proc. 14th International Parallel and Distributed Process. Symp. IPDPS 2000.
16. Arpaci-Dusseau AC (2001) Implicit coscheduling: coordinated scheduling with implicit information in distributed systems. ACM Trans Comput Syst 19: 283-331.
17. Frachtenberg E, Feitelson DG, Petrini F, Fernandez J (2003) Flexible coscheduling: mitigating load imbalance and improving utilization of heterogeneous resources. Proc Int Parallel Distrib Process Symp 2003.
18. Shmueli E, Feitelson DG (2003) Backfilling with Lookahead to Optimize the Performance of Parallel Job Scheduling. Job Sched Strateg Parallel Process.
19. Mualem AW, Feitelson DG (2001) Utilization, predictability, workloads, and

user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Trans Parallel Distrib Syst 12: 529-543.

20. Srinivasan S, Kettimuthu R, Subramani V, Sadayappan P (2002) Characterization of backfilling strategies for parallel job scheduling. Proceedings. International Conference on Parallel Processing Workshop.

21. Feitelson DG, Rudolph L, Schwiegelshohn U (2005) Parallel Job Scheduling - A Status Report. JSSPP'04 Proc 10th Int Conf Job Sched Strateg Parallel Process.

22. Talby D, Feitelson DG (2005) Improving and stabilizing parallel computer performance using adaptive backfilling. Proc 19tIEEE Int Parallel Distrib Process Symp IPDPS 2005.