Study of Various Partitioning Policies of Multiprocessor Systems

Harpreet Kaur[#], Sukhpreet Kaur[#]

[#] *Faculity, Department of Computer Science & Engineering, Panjab University SSGRC, Hoshiarpur*
[#] *Faculity, Department of Computer Science & Engineering, Panjab University SSGRC, Hoshiarpur*

Abstract
Many techniques of partitioning the processing elements have been developed since past few years to improve the performance of the system. A common approach is to divide the set of processing elements into independent partitions depending upon the job requirements. This can be done statically, dynamically or adaptively depending upon current requirements and workload characteristics of the particular job. This paper presents several partitioning policies, which are commonly used to partition the set of processing elements to improve the performance of the multiprocessor systems.

Keywords
Adaptive Partitioning, Equipartitioning, Multiprocessors, Partitioning Policies, Scheduling, performance.

I. INTRODUCTION
Parallel job scheduling has got the increasing recognition in recent years. Parallel computing is an efficient technique of achieving the high performance and efficient use of multiprocessor and multicomputer systems over sequential processing on a single processor. The major goal of parallelization is to reduce the overall computation time and improving the performance by distributing or dividing the computation workload amongst the available number of processors [4].

One of the most crucial issues in parallel computing is the efficient distribution of a workload and data (workload balancing) among the processors in multiprocessor and multicomputer systems to achieve optimal performance. The main objective of partitioning the space or resources is to minimize the scheduling overheads [1]. Therefore, it is important to study and implement the efficient decomposition techniques, which play an important role in achieving desired performance and efficient use of multiprocessor and multicomputer systems. Based on whether the workload is distributed before or during run-time, partitioning can be classified as static, adaptive or dynamic [1][6].

II. PARTITIONED SYSTEM
Partitioned systems allow applications with different criticalities or workload characteristics to co-exist and run on the same module without causing any damages to other partitions or applications. A partitioned system is a static system where the time to start and stop an application is predetermined.

The advantages of partitioning the system are [2]:

1. It allows applications with different characteristics to co-exist and run on the same core module without causing any potential damages to other partitions/ applications.

2. Partitions provide the flexibility to add enhancement to an application without modifying the schedule or any other applications.

3. The benefit of partitioning is that one can make modifications in one partition and not have to test the others because each partition has its own space.

III. PARTITIONING STRATEGIES
The parallel application may not be able to efficiently utilize all the processors in the system, it may be better to partition the processor set among the program in a space sharing fashion. Several approaches have been considered in space sharing class [4].

A. Static Partitioning: In this processors are partitioned into a fixed number of disjoint sets, each of which are allocated to individual jobs, This scheduling strategy has often been used in the number of commercial systems. In this policy the system overhead is low and it is a simple policy from both the system and application viewpoints. The static scheduling approach however, can lead to relatively low system throughputs and resource utilizations under non-uniform workloads.

B. Adaptive Partitioning: In this policy, the number of processors allocated to the job is determined when job arrives and depart based on the current system state. This approach outperforms its static counterparts by adapting partitioning sizes to the current load. However the performance benefits of adaptive partitioning can be limited due to its inability to adjust scheduling decisions in response to subsequent workload changes.

C. Dynamic partitioning: In this policy, the size of the partition allocated to the job can be modified during execution, at the expense of increased overhead. The runtime costs of a dynamic partitioning policy are heavily dependent upon the parallel architecture and application workload under consideration. [3],[4]

IV. PERFORMANCE METRICS
As jobs enter and leave the system, the following statistics can be collected [4]:

A. Load average: The nominal load in the system at any time is the fraction of processors that are busy (allocated to a job). The load average is the nominal load averaged over time. Because the jobs in this workload have sublinear speedups, the total allocated time, T, exceeds the sequential lifetime, L, whenever _ > 0 and the cluster size is greater than 1. Thus, the measured load may exceed the offered load.

B. Utilization: Utilization takes into account not only how many processors have been assigned to jobs, but also the efficiency with which those jobs are running. Efficiency is the ratio of speedup to cluster size; utilization is efficiency averaged over processors and time. In most real systems the efficiency of jobs, and therefore the utilization of the system, are not known.

C. Average turnaround time: The turnaround time is the time between the arrival and completion of a job; i.e. the sum of its queue time and its run time.

D. Average slowdown: Slowdown is the ratio of turnaround time to the shortest possible turnaround time, as if the job had run on a dedicated machine. In other words,

$$slowdown = \frac{queuetime + R(n)}{R(N)}$$

where R(n) is the run time on the allocated cluster size, n, and R(N) is the hypothetical run time on all N processors. Slowdown is a useful performance metric because it gives equal weight to all jobs regardless of length, whereas average turnaround time tends to be dominated by long jobs. Also, slowdown may better represent users' perception of system performance, since it measures delays relative to job duration. For example, a long queue time is more acceptable for a long job than for a short one. Slowdown captures this implicit cost function [5].

V. ALLOCATION POLICIES

A. Fixed Processors Per Job (PPJ): In this policy, the processor set is divided into a fixed number of equal sized partitions. When a job arrives, it waits for the next available free partition and then executes in that partition. The maximum number of simultaneously executing jobs is equal to the number of partitions, which is equal to the total number of processors divided by the partition size. The PPJ policy is static as the PARTITION_SIZE remains fixed throughout the lifetime of the system. It achieve good performance under particular conditions.

B. Equal Partitioning with a Maximum (EPM): In this policy, the partition size is computed at allocation time instead of at system configuration time. In EPM policy, the set of currently free processors is equally divided among the jobs in the waiting queue. To limit the maximum size of any allocated partition, an upper bound on the partition size is introduced as a configuration parameter. For example, if MAX= TOT_PEs/2, then no single job is allowed to execute on more than half of the system processors, regardless of the number of free processors and the number of waiting jobs. This allows a mechanism to restrict any job from monopolizing the system by reserving some of the free processors.

The EPM policy is similar to the ASP policy. The difference is that the value of MAX in ASP is the workload's maximum parallelism. The maximum parallelism is defined as the maximum number of simultaneous busy processors during the execution of a program when a sufficiently large amount of processors is available. Here the workload's maximum parallelism may not be known; the value of MAX is set at system configuration time and may be assigned a value between 1 and TOT_PEs.

C. Adaptive Policies

1. Adaptive Policy 1 (AP1): AP1 [6] is specifically used for distributed memory systems. The target partition size at a given time is equal to the total number of processors in the system divided by the number of waiting jobs. If no processors are available when a job arrives, it joins a FIFO queue. Otherwise, it is allocated a number of processors equal to the minimum of its maximum parallelism, the target partition size, and the number of available processors. At each job completion, the same rule is used for the allocation of the released processors to jobs from the FIFO queue. The last job activated gets the remaining available processors, even if that is fewer than either of its maximum parallelism and the target partition size.

Depending upon the specification of procedure compute_target_size, which implements specific split and merge strategies, various adaptive policies are possible.

The compute_target_size specification for AP1 is

$$target\_size \leftarrow max\left(1, \frac{TOT\_PEs}{queue\_length} + 0.5\right)$$

In AP1, the split and merge strategies are directly dependent upon the current value of queue_length. Whenever the queue length increases, the target size decreases.

The compute_target_size specification for AP2 is

$$target\_size \leftarrow max\left(1, \frac{TOT\_PEs}{queue\_length + 1} + 0.5\right)$$

The split strategy of AP2 (when queue length increases) tends towards premature fragmentation.

2. Adaptive Equipartition: The ideal allocation is to divide the processors in the system equally among all the running and waiting jobs [6]. But this cannot be done in a non-preemptive policy. However, one can use as a target partition size the total number of processors divided by the total number of jobs in the system, both waiting and running (whereas both ASP and AP1 use a target allocation that is a number of processors divided by only the number of waiting jobs).

If no processors are available when a job arrives, it joins a FIFO queue. Otherwise, it is allocated a number of processors equal to the minimum of its maximum parallelism, the target partition size and the number of available processors. At each job completion, the same rule is used for allocating the released processors among the queued jobs, in FIFO order.

D. Dynamic Equipartition Policy: In the Dynamic Equipatition Policy (DYN-EQUI), the processors are dynamically partitioned as equally as possible among the applications. Some provisions are taken so that no application is given more processor that it can use. When the number of processors allocated to the application changes, the application readjust the number of running processes accordingly.

VI. ANALYTICAL COMPARISON OF DIFFERENT POLICIES

Scheduling Policy	Load Parameters	Target_Size
PPJ	Waiting Jobs	Statically determined depending upon the job requirements
EPM	Waiting Jobs	$\frac{TOT\_PEs}{waiting\_Queue}$
AP1	Waiting Jobs	$MAX(1, \frac{TOT\_PEs}{queue\_length} + 0.5)$
AP2	Waiting Jobs	$MAX(1, \frac{TOT\_PEs}{queue\_length+1} + 0.5)$
AEP	Waiting + Running Jobs	$\frac{TOT\_PEs}{Waiting\_queue + Running\_queue}$

VII. CONCLUSION

The goal of this paper is to give the brief review of the various partitioning policies. Usually, different types of parameters

like job type, job size, wait time, service time, processors allocated and information are used by different policies, so it is unclear to measure the benefits of each policy under different workload conditions. Adaptive policies perform better than fixed-partitioning and variable-partitioning scheduling policies due to their ability adapt to the current load on the system while calculating partition-size for jobs. Adaptive space-sharing scheduling policies to schedule moldable jobs are widely studied in homogeneous parallel systems (i.e. multiprocessors and clusters) and to less extent in heterogeneous cluster computing systems. Therefore, this paper presents the simple scheduling rules, classification type and information needs of each policy.

REFERENCES

[1] Nedal Kafri, Jawad Abu Sbeih "Simple Near Optimal Partitioning Approach to Perfect Triangular Iteration Space." Proceedings of the 2008 High Performance Computing & Simulation Conference ©ECMS

[2] A. Hariprasad Kodancha, "Time Management in Partitioned Systems" Master's Thesis, Department of Computer Science and Automation Indian Institute of Science Bangalore, October 2007.

[3] Mark S.Squillante, "On the Benefits and Limitations of dynamic Partitioning in Parallel Computer Systems"

[4] E. Rosti, E. Smirni, L.W. Dowdy, G. Serazzi, B.M. Carlson, "Robust Partitioning Policies of Multiprocessor Systems." Matematical Sciences Section of Oak Ridge National Laboratory.1993

[5] Stergios V. Anastasiadis and Kenneth C. Sevcik, "A parallel workload model and its implications for processor allocation" High Performance Distributed Computing, 1997, Proceedings. The Sixth IEEE International Symposium on Aug, 1997.

[6] J.H. Abawajy,"An efficient adaptive Scheduling policy for high performance computing" Future Generation Computer systems, Vol. 25, 364-370, (2009).

[7] Srividya Srinivasan, Vijay Subramani, Rajkuman Kettimuthu, Parveen Holenarsipur and P. Sadayappan," Effective Selection of Partition sizes for Moldable Scheduling of Parallel Jobs" Proceeding in – HiPC '02 Proceedings of the 9th International Conference on High Performance Computing Springer-Verlag London, UK ©2002

[8] Allen B. Downey,"A parallel workload model and its implications for processor allocation" Report No. UCB/CSD-96-922, November 1996.

[9] Amit Chhabra, Gurvinder Singh," An Improved Adaptive Space-Sharing Scheduling Policy for Non-dedicated Heterogeneous Cluster Systems "International Journal of Computer Applications and Technology Volume 1- Issue 2, 2012, 57-63.

[10] S.P. Dandamudi and Z. Zhou, "Performance of Adaptive Space-Sharing Policies in Dedicated Heterogeneous Cluster Systems", Future Generation Computer Systems, 20(5), 895-906 (2004).

[11] Young-Chul Shim, "Performance evaluation of scheduling schemes for NOW with heterogeneous computing power", Future Generation Computer Systems. 20(2): 229-236 (2004).