Important Aspects of Parallel Computing

Rafiqul Zaman Khan, Javed Ali

Department of Computer Science,
Aligarh Muslim University, Aligarh

Abstract-Parallel computing is a form of computation in which many calculations are carried out simultaneously. It operates on the principle that large problems can often be divided into smaller ones, which can be solved concurrently. In a distributed system, all processing elements are connected by a network. Parallel computing becomes the dominant paradigm in computer architecture, mainly in the form of multi-core processors. Data parallel programming is one which each process executes the same action concurrently, but on different parts of shared data. While in task parallel approach, every process performs a different step of computation on the same data.

Keywords: Directed Acyclic Graph, Parallel Computing, Throughput, Scheduling etc.

1.1 INTRODUCTION

In traditional scheduling, scheduling is defined as the allocation of operations to processing units. One of the primary differences between traditional and parallel computing scheduling is that the parallel computing scheduler does not have full control over parallel computing systems. More specifically, local resources are not controlled by the parallel computing scheduler, but by local scheduler. Another difference is that parallel computing scheduler cannot assume that it has a global view of parallel computing. The difference occurs mainly due to the dynamic nature of resources and constraints in parallel computing environment. Issues on the basis of which differences could be considered are as follows:

1.2 Task partitioning constraints in parallel computing: Scheduling constraints can be hard or soft. Hard constraints are rigidly enforced. Soft constraints are those that are desirable but not absolutely essential. Soft constraints are usually combined into an objective function.

1.3 Optimization criteria for parallel computing scheduling: A variety of optimization criteria for parallel computing scheduling is given below:

1. Minimization of inter-process communication cost, sleek time (processor utilization time) and NSL (Normalized Schedule Length).
2. Maximization of personal or general utility, resource utilization, fairness etc.

1.4 Data scheduling: A variety of data is necessary for a scheduler to describe the jobs and resources [7] e.g. Job length, resource requirement estimation, time profiles, uncertain estimation etc.

1.5 Methodologies: It is obvious that there is no deterministic scheduling method which shows ideal results for parallel computing scheduling. But in conventional scheduling many appropriate methods are available.

2.1 ISSUES AND CHALLENGES IN PARALLEL COMPUTING

Parallel computing is emerging as a viable option for high performance parallel computing. Sharing of resources in parallel computing provides improved performance at low cost. In parallel computing there are various issues such as:

2.1.1 Role of server system: Server system plays a major part in the parallel computing heterogeneous system as today's server systems are relatively small and they are connected to networks by the interfaces [14]. So these systems must be supporting for high performance computing architectures.

3.1 THREADS IN PARALLEL PROCESSING ALGORITHM

A program in execution is known as a process. Light weight processes are called threads. The commonly used term thread is taking from thread of control which is just a sequence of statements in a program. In shared memory architectures, a single process may have multiple thread of control. Dynamic threads are used in shared memory architectures. In dynamic threads, the master thread controls the collection of workers threads. The master thread forks worker threads, these threads complete assigned request and after termination it again joins a master thread. This facility makes the efficient use of system resources, because only at the time of running state resources are used by threads. This phenomenon reduces idle time of the participating processors. These facilities of threads maximize throughput and try to minimize the communication overhead. Static threads are generated by the required setup which is known in advance.

3.1.1 Threads benefits: Multiple threads programming have the following benefits.

3.1.2 Responsiveness: If some threads are damaged or blocked in multithreading applications then it allows a program to run continuously. It facilitates user interaction even if a process is loaded in another thread. So it increases the responsiveness of users.

3.1.3 Economy and resource sharing: During the execution of the process it's very costly to allocate memory in traditional parallel computing environments. In multithreading, all threads of a process can share allocated resources. So the creation and context switching of the threads of a process is economical. Because the creation and allocation of a process are much more time consuming than threads. So thread generates a very low overhead in the

comparison of processes. Creating a process is thirty time slower than creating a thread in Solaris (2) systems. Context switching is 5 times slower in processes than threads. The thread code sharing facility provides an application to have several different threads of activities within the same address space. These threads may share resources effectively and efficiently.

3.1.4 Utilization of multiprocessor architectures: Each thread of all processes may run simultaneously upon different processors. Kernel level threads parallelization increases the usage of processors. Kernel level threads are supported directly by the operating systems without user thread interventions. While in traditional multiprogramming, kernel level processes can't be generated by the operating systems.

4.0 DAG IN DIFFERENT SCHEDULING ENVIRONMENTS

Any parallel program may be represented by DAG. In the execution of parallel tasks more than one computing units required concurrently [21]. DAG scheduling has been investigated on a set of homogeneous and heterogeneous environments. In heterogeneous systems, it has different speed and execution capabilities. The computing environments are connected by the networks of different topologies. These topologies (ring, star, mesh, hypercube etc.) may be partially or fully connected. Minimization of the execution time is the main objective of DAG. It allocates the tasks to the participating processors by preserving precedence constraints. Scheduling length of a schedule (program) is the overall finish time of the program. This scheduling length is termed as the mean flow time [2, 17] or earliest finish time in many proposed algorithms.

In DAG a parallel program is represented by $G = (V, E)$, where V is the set of nodes (n) and E is the set of directed edges (e).A set of instructions which can be sequentially on the same processor without preemption is known as a node on DAG. Every edge (n_i, n_j) is the corresponding computation message amongst node by preserving the precedence constraints. The communication cost of the nodes is denoted by (n_i, n_j). Sink node is called child node and source node is called the parent node of a DAG. A node without child node is known as an exit node while a node without parent node is called an entry node. Precedence constraints mean that child node can't be executed without the execution of parent node. If two nodes are assigned on same processor then communication cost between them is assumed to be zero.

The loop structure of the program can't be explicitly modeled in the DAG. Loop separation techniques [4, 18] divide the loop into many tasks. All iterations of loop started concurrently with the help of DAG. This model can be used for large matrix multiplications and data flow problems. Fast Fourier Transformation (FFT) or Gaussian Elimination (numerical applications) loop bound must be known at the time of compilation. So many iterations of a loop can be encapsulated in a task. Message passing primitives and memory access operations are means of calculation of node weight and edge weight [15]. The granularity of tasks is also divided by the programmer [16]. The scheduling algorithms [22] are used to refine the granularity of DAG.

Preemptive and non-preemptive scheduling approaches investigated by [3] in homogeneous computing architectures. They used independent tasks without having precedence constraints. In DAG, condition of precedence constraints is inserted by Chung and Ranka [6] for preemptive and non-preemptive scheduling. Earliest time factor inserted by them in list scheduling strategies.

In preemptive scheduling, a partial portion of the task can be reallocated to the different processors [11, 13,19]. While in the case of no-preemptive scheduling, allocated processors can't be re-allocated until it finishes assigned tasks. Flexibility and resource utilization of the preemptive scheduling is more than non-preemptive scheduling in a theoretical manner. Practically, re-assigning partial part of task causes extra overhead. Preemptive scheduling shows polynomial time solutions while non-preemptive are NP-complete [5, 10]. Scheduling with duplication upon different processors is also NP-complete. Communication delay amongst preemptive tasks is more due to preemptions of processors.

Problems of conditional branches may be analyses by DAG. Scheduling of probabilistic branches reported in[20]. In this scheduling, each branch is associated with some non-zero probability having some precedence constraints. Towsley [20] used two step methods to generate a schedule. In first step, he tries to minimize the amount of indeterminism. While in second step, reduced DAG may be generated by using pre-processor methods. Problem of conditional branches is also investigated in [9] by merging schedules to generate a unified schedule.

CONCLUSION

This paper represents the fundamental concepts of parallel computing in efficient and effective manner. Thread concept is highly important in high performance computing systems. To maximize throughput threads play an important role to reduce inter-process communication cost. A researcher may use thread to calculate high computing problems. Therefore DAG and thread are highly useful parameters for parallel computing environment

REFERENCES

[1] G. M. Amdahl, "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities," In *AFIPS Conference Proceedings*, (1967), 480–481.

[2] J. Bruno, E. G. Coffman and R. Sethi, "Scheduling Independent Tasks to Reduce Mean Finishing Time," *Commun. ACM 17*, 7 (July), (1974), 380–390.

[3] J. Blazewicz, J. Weglarz and M. Drabowski, "Scheduling Independent 2-Processor Tasks to Minimize Schedule Length," *Inf. Process.Lett. 18*, 5 (June) (1984), 267–273.

[4] M. Beck, K. Pingali and A. Nicolau, "Static Scheduling for Dynamic Dataflow Machines," *J. Parallel Distrib. Comput. 10*, 4 (Dec.) (1990), 275–291.

[5] E. G. Coffman and R. L. Graham, "Optimal Scheduling for Two-Processor Systems," *Acta Inf. 1*, (1972), 200–213.

[6] Y. C. Chung and S. Ranka, "Applications and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithm on Distributed Memory Multiprocessors," In *Proceedings of the 1992 Conferenceon Supercomputing* (Supercomputing '92, Minneapolis, MN, Nov. 16–20), R. Werner, Ed. IEEE Computer Society Press, Los Alamitos, CA, (1992), 515–525.

[7] R. Cheng, M. Gen and Y. Tsujimura, "A Tutorial Survey of Job-Shop Scheduling Problems using Genetic Algorithms," *Comput. Ind. Eng. 30*, 4, (1996), 983–997.

[8] A. W. David and D. H. Mark, "Cost-Effective Parallel Computing," *IEEE Computer*, (1995), 67–71.

[9] El-Rewini and M. A. Hesham, "Static Scheduling of Conditional Branches in Parallel Programs," *J. Parallel Distrib. Comput. 24*, 1 (Jan. 1995), 41–54.

[10] T. Gonzalez and S. Sahni, "Preemptive Scheduling of Uniform Processor Systems," *J.ACM 25*, 1 (Jan), (1978), 92–101.

[11] Gustafson, J.L., "Reevaluating Amdahl's Law," Ames lab web link http://www.scl.ameslab.gov/Publications/AmdahlsLaw/Amdahls.html, June 1996.

[12] E. C. Horvath, S. Lam and R, "Sethi, A Level Algorithm for Preemptive Scheduling," *J. ACM 24*, 1 (Jan.), (1977), 36–47.

[13] T. Haluk, H. Salim and Y. W. Min, "Performance-Effective and Low Complexity Task Scheduling for Heterogeneous Computing," IEEE transactionson parallel and Distributed systems, Vol. 13, No. 3, (2002) 262-279.

[14] H. Jiang, L. N. Bhuyan, and D. Ghosal, "Approximate Analysis of Multiprocessing Task Graphs," In *Proceedings of the International Conference on Parallel Processing* (Aug), (1990), 230–238.

[15] Y. K. Kwok and I. Ahmad, "Efficient Scheduling of Arbitrary Task Graphs to Multiprocessors using a Parallel Genetic Algorithm," *J. Parallel Distrib. Comput. 47*, 1, (1997) 55–78.

[16] J. Y. T. Leung and G. H. Young, "Minimizing Schedule Length Subject to Minimum Flow Time," *SIAM J. Comput. 18*, 2 (Apr), (1989), 310–333.

[17] B. Lee, A. R. Hurson and T. Y. Feng, "A Vertically Layered Allocation Scheme for Data Flow Systems," *J. Parallel Distrib. Comput. 11*, 3 (Mar.), (1991), 172–190.

[18] V.J. Rayward-Smith, "The Complexity of Preemptive Scheduling Given Inter-processor Communication Delays," *Inf. Process. Lett. 25*, 2 (6 May), (1987), 120–128.

[19] D. Towsley, "Allocating Programs Containing Branches and Loops within a Multiple Processor System," *IEEE Trans. Softw. Eng. SE-12*, 10 (Oct.), (1986), 1018–1024.

[20] Q. Wang, and K. H. Cheng, "List scheduling of parallel tasks," *Inf. Process. Lett. 37*, 5 (Mar.), (1991), 289–295.

[21] T. Yang and A. Gerasoulis, "PYRROS: Static Task Scheduling and Code Generation for Message Passing Multiprocessors," In *Proceedings of the 1992 international conference on Supercomputing* (ICS '92, Washington, DC , K. Kennedy and C. D. Polychronopoulos, Eds. ACM Press, New York, NY, (1992), 428–437.

AUTHORS:

Dr. Rafiqul Zaman Khan:

Dr. Rafiqul Zaman Khan, is presently working as a Associate Professor in the Department of Computer Science at Aligarh Muslim University, Aligarh, India. He received his B.Sc Degree from M.J.P Rohilkhand University, Bareilly, M.Sc and M.C.A from A.M.U. and Ph.D (Computer Science) from Jamia Hamdard University. He has 18 years of Teaching Experience of various reputed International and National Universities viz King Fahad University of Petroleum & Minerals (**KFUPM**), K.S.A, Ittihad University, U.A.E, Pune University, Jamia Hamdard University and AMU, Aligarh. He worked as a **Head** of the Department of Computer Science at Poona College, University of Pune. He also worked as a **Chairman** of the Department of Computer Science, AMU, Aligarh. His Research Interest includes Parallel & Distributed Computing, Gesture Recognition, Expert Systems and Artificial Intelligence. Presently **04** students are doing PhD under his supervision.

He has published about **45** research papers in International Journals/Conferences. Names of some Journals of repute in which recently his articles have been published are International Journal of Computer Applications (ISSN: 0975-8887), **U.S.A**, Journal of Computer and Information Science (*ISSN: 1913-8989*), **Canada**, International Journal of Human Computer Interaction (*ISSN: 2180-1347*), **Malaysia,** and Malaysian Journal of Computer Science(*ISSN: 0127-9084*), **Malaysia**. He is the Member of Advisory Board of International Journal of Emerging Technology and Advanced Engineering (IJETAE), Editorial Board of International Journal of Advances in **Engineering** & Technology (IJAET), International Journal of Computer Science Engineering and Technology (IJCSET), International Journal in Foundations of Computer Science & technology (IJFCST) and Journal of Information Technology, and Organizations (JITO).

Javed Ali:

Javed Ali is a research scholar in the Department of Computer Science, Aligarh Muslim University, Aligarh. His research interest include parallel computing in distributed systems. He did Bsc(Hons) in mathematics and MCA from Aligrah Muslim University ,Aligarh. He published seven international research papers in reputed journals. He received state level scientist award by the government of India.