International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 03 | June-2015 www.irjet.net p-ISSN: 2395-0072

REVIEW AND ANALYSIS OF TASK SCHEDULING ALGORITHMS

Chinam Bajaj[1], Anu Dogra[2], Dr. Gurvinder Singh [3],

[1] *M.tech, CSE Department, GNDU Amritsar, Punjab, India*
[2] *M.tech, CSE Department, GNDU Amritsar, Punjab, India*
[3] *Professor, HOD, Dr. Gurvinder Singh, GNDU Amritsar, Punjab, India*

Abstract - ***Task scheduling is one of the essential part for proper functioning of parallel processing system. Scheduling and mapping of precedence constrained task graphs to the processors is one of the most critical problems in parallel computing. Due to the NP- complete nature of the problem, a large portion of related work relies on heuristic approaches with the objective of finding better solutions within a reasonable amount of time. Several approaches have been applied to solve this problem. The objective of this paper is to study and explore several task scheduling algorithms and propose a GA based algorithm using various scenarios for scheduling tasks onto processors. Also the criteria, how the parallel task scheduling algorithms can be evaluated is determined. The recommended algorithm has the ability to overcome the limitations which are present in earlier techniques i.e to increase the throughput and decrease the completion time of the system.***

Key Words: *DAG, Genetic algorithm, fitness function*

1. INTRODUCTION

There has been a great demand for high speed computing in many application areas. As the computing power available is increasing day by day, the search for more and more power also keeps increasing. Parallel computing is a form of computation in which many calculations are carried out simultaneously, [1] operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel").Parallel processing is the ability to carry out multiple operations or tasks simultaneously. The simultaneous use of more than one CPU or processor requires executing a program or multiple threads. Ideally, parallel computing makes programs run quicker as there are more engines (CPUs or Cores) running it [2] [3].

In parallel processing systems, scheduling is the most important consideration. The aim of scheduling is to minimize the completion time of a parallel application by properly allocating the tasks to the processors and also sequencing the execution of the tasks. Scheduling is crucial and essential for proper operation of multiprocessor systems and it is performed by the operating system. Every processor needs operating system to hide its details and manage its resources. It can be used for tasks that involve large number of calculations and can be divided into number of smaller tasks. Thus, the efficiency of parallel processing systems depends on how efficient is the scheduling policy used by the operating system.

One of the major problems in parallel processing system is to partition the program into tasks which will be appointed to totally different processors for parallel execution. If a higher degree of parallelism is the objective, the communication that will be required between the tasks is in greater amount. But, if communication is restricted, potential parallelism are going to be lost. The goal of partitioning heuristics is to divide the program into the appropriate size and number of tasks so that the two conflicting necessities of low communication overhead and then the high degree of parallelism should be balanced. The most effective partitioning scheme will minimize the parallel running time .A sequential program once divided is often potrayed by a Program Dependence Graph (PDG) that is a directed acyclic graph (DAG). Every vertex in it denotes a task and its weight, which denotes its processing time. Every edge denotes the precedence relation between the 2 tasks, and therefore the weight of the edge is the communication cost that incurs if the 2 tasks are assigned to completely different processors. Main factor in efficient utilization of multiprocessor systems is proper assignment and scheduling of computational tasks of a DAG among processors.

Within this paper we study the various parallel static task scheduling algorithms and study the characteristics of different scheduling algorithms. In section 2, related work is discussed that determine the criteria on which parallel task scheduling algorithms can be evaluated. Section 3 more formally describes the gaps in existing literature and section 4 proposes a new genetic algorithm for multiprocessor scheduling that solves the gaps. Finally section 5, concludes this paper.

2. RELATED WORK

(Ahmad, Y .K. Kwok and M. Y Wu) [4] Survey algorithms that allocate a parallel program represented by an edge-weighted directed acyclic graph (DAG), also known as a task graph or macro dataflow graph, to a collection of homogeneous processors, with the target of minimizing the

completion time. There are many algorithms and they are divided into four groups. The first group includes algorithms that schedule the DAG to a bounded number of processors directly. These are called (BNP) scheduling algorithms. The algorithms in the second group schedule the DAG to an unbounded number of clusters and are called (UNC) scheduling algorithms. The algorithms in the third group schedule the DAG using task duplication and are called the task duplication based (TDB) scheduling algorithms. The algorithms in the fourth group perform allocation and mapping on arbitrary processor network topologies. These algorithms are called (APN) scheduling algorithms. The design methodologies behind these algorithms are discussed, and the performance of all of the algorithms is evaluated and compared with each other on a unified basis by applying various scheduling parameters.

(Yu-Kwonk Kwok) [5] Classified various algorithms into different categories according to their assumptions and functionalities. A performance measure called scheduling scalability (SS) has also been proposed that captures the collective effectiveness of a scheduling algorithm in terms of its solution quality, the number of processors that will be used, and the running time.

L.D Davis [6] has covered the basic principles of genetic algorithms and the number of variations in population size, in initialization methods, in fitness definition, in selection and replacement strategies in crossover and mutation. Also issues of implementation relating to one of the most popular meta heuristics have been presented.

G.C Sih and E.A Lee [7] have presented a compile-time scheduling heuristic known as dynamic level scheduling, that accounts for interprocessor communication overhead once mapping precedence-constrained, communication tasks onto heterogeneous processor architectures with limited or possibly irregular interconnection structures. This method uses dynamically changing priorities to match tasks with processors at every step, and schedules over each of spatial and temporal dimensions to remove shared resource contention. This method is quick, flexible, wide targetable, and gives good performance.

Bashir, A. F., V. Susarla, and K. Vairavan [8] have shown the results of an experimental study on the performance of the very easy level scheduling algorithm for unit time task systems. Though the matter of problem of the development of better k-processor schedules for unit time tasks is NP-complete, the experimental investigation suggests that the proposed efficient linear time algorithm can give optimal schedules most of the time in the sense that the probability of producing an better schedule using this algorithm is at least 0.9 for over 700 cases randomly generated in experiment.

Kwok, Yu-Kwong, and Ishfaq Ahmad [9] Proposed a novel GA-based algorithm with an aim to simultaneously meet the goals of high performance, scalability, and fast running time. The proposed PGS (Parallel Genetic Scheduling) algorithm is a parallel algorithm and generates supreme quality solutions in a short duration. The PGS algorithm can give a better scheduling list which leads to an optimal schedule.

Ishfaq Ahmad and Yu-Kwong Kwok [10] Proposed a parallel algorithm, called PBSA (Parallel Bubble Scheduling and Allocation), that produces a good quality scheduling solutions. The novelty of the algorithm lies in a systematic dividing of the task graph that guides the concurrent generation of sub schedules. The algorithm works well for regular or irregular graph structures with arbitrary communication and computation costs, handles arbitrarily connected network of target processors, and is appropriate for homogenous and heterogeneous processor systems.

Yu-Kwong Kwok and Ishfaq Ahmad [11] Proposed large spectrum of techniques, including branch-and-bound, integer-programming, searching, graph theory, randomization, genetic algorithms, and evolutionary methods. They also described various scheduling algorithms and their functionalities in a contrasting fashion as well as studied their relative advantages in terms of performance and time-complexity.

Joseph YT Leung, and Gilbert H. Young [12] considers the problem of pre-emptively scheduling a task system and shows the NP hardness of finding a minimum mean flow time schedule for a set of independent tasks with release time and deadline constraints. Also a greedy algorithm to find a better schedule for a large class of task systems has been given.

J. B Sinclair [13] presented an algorithm named as minimum independent assignment cost underestimate (MIACU) to find an optima assignment of distributed tasks. This is a branch and bound with underestimates algorithms and simulation showed that it gives excellent results in reducing average time and space complexities for finding optimal assignments.

A. A Khan, Carolyn L. McCreary, and M. S. Jones [14] illustrate three types of algorithms and their relative capabilities on a variety of graphs. The comparison between a graph based method, a list scheduling technique and three critical path methods have also been made. The strengths and weaknesses of the algorithms have been discussed.

Kamaljit Kaur, Amit Chhabra, and Gurvinder Singh [15] have proposed a new genetic algorithm, Heuristics based Genetic algorithm for scheduling static tasks onto homogeneous parallel system. This algorithm minimizes the completion time and increase the throughput of the system.

3. GAPS IN EXISTING LITERATURE

The considered scheduling problem is NP complete problem. Various methods have been proposed and developed to find the solution in reasonable amount of time which are studied and discussed under literature review section. The gaps in study are identified and are discussed along with found drawbacks of existing methods as follows:

The major drawback of most of the existing heuristics is that they are evaluated with small problems sizes only, thus their scalability is not known. Also, most of the heuristics do not perform well when important details of the application and the target system, are taken into account. As a result, they are not applicable in a real environment with moderately large problems. Few solutions have been proposed using Genetic Algorithms as well, but there is too much scope of improvement in existing implementations. Study of existing methods shows many drawbacks, such as fitness function needs to be improved to make it multi objective function, Roulette wheel selection is used which may lead to early convergence problem on local minima.

4. METHODOLOGY PROPOSAL

It defines how different gaps discussed above will be removed using the proposed algorithm using genetic operators.

The proposed genetic algorithm is given below:

[STEP 1] Read number of tasks n and communication cost in directed acyclic graph (DAG) and build the design database. Also, Read the parameters like population size (pop_size), crossover probability (X_p), mutation probability (M_p), number of maximum generations (max_gen) and the number of processors (P) for GA processing.

[STEP 2] Let generation gcount=0

[STEP 3] Initialization: Initialize the cur_pop by the initial population by selecting chromosomes randomly.`

[STEP 4] While (gcount <= max_gen) Do

[STEP 4.1] Define fitness function to calculate the cost and the fitness value of each chromosome in the cur_pop. Save the fittest current solution in the database.

[STEP 4.2] Select a pair of chromosome from cur_pop probabilistically based on their fitness using stochastic universal sampling strategy as parents to contribute to the next generation.

[STEP 4.3] Perform crossover with probability (C_p) and mutation with probability (M_p) to generate offsprings to form new generation.

[STEP 4.4] Replace the entire cur_pop with the new population

[STEP 4.5] gcount=gcount+1

[STEP 4.6] Endwhile

[STEP 5] If gcount=max_gen, then exit with best solution and stop else when gcount< max_gen goto step 3 .

5. CONCLUSIONS AND FUTURE SCOPE

The review shows the scheduling problem that is how to the schedule the tasks onto multiple processors and order their execution so as to optimize their performance criteria and also how to analyze their performance. Also there are different problems in existing approaches and how these problems will be eliminated or reduced using Genetic Algorithm to determine suitable priorities of tasks by applying list scheduling heuristic as a decoding heuristic which leads to a better solution. The technique is proposed using Genetic Algorithm to determine suitable priorities of tasks by applying list scheduling heuristic as a decoding heuristic which leads to a better solution.

Another improved selection methods such as stochastic universal sampling can be applied and analyzed. Different crossover and mutation probabilities can be applied and analyzed in order to converge to final solution in lesser number of generations.

The review shows the scheduling problem that is how to the schedule the tasks onto multiple processors and order their execution so as to optimize their performance criteria and also how to analyze their performance. Also there are different problems in existing approaches and how these problems will be eliminated or reduced using Genetic Algorithm to determine suitable priorities of tasks by applying list scheduling heuristic as a decoding heuristic which leads to a better solution. The technique is proposed using Genetic Algorithm to determine suitable priorities of tasks by applying list scheduling heuristic as a decoding heuristic which leads to a better solution.

Another improved selection methods such as stochastic universal sampling can be applied and analyzed. Different crossover and mutation probabilities can be applied and analyzed in order to converge to final solution in lesser number of generations.

REFERENCES

1. Gottlieb, Allan; Almasi, George S. (1989). Highly parallel computing. Redwood City, Calif.: Benjamin/Cummings. ISBN 0-8053-0177-1.
2. Principles of Parallel Programming by Lin C. and Snyder L.

3. http://software.intel.com/en-us/articles/how-to-sound-like-a-parallel-programming-expert-part-1-introducing-concurrency-and-parallelism/
4. Ahmad, Ishfaq, Yu-Kwong Kwok, and Min-You Wu. "Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors." In*Parallel Architectures, Algorithms, and Networks, 1996. Proceedings., Second International Symposium on*, pp. 207-213. IEEE, 1996.
5. Kwok, Yu-Kwong, and Ishfaq Ahmad. "Benchmarking and comparison of the task graph scheduling algorithms." *Journal of Parallel and Distributed Computing* 59, no. 3 (1999): 381-422.
6. Davis, Lawrence, ed. *Handbook of genetic algorithms*. Vol. 115. New York: Van Nostrand Reinhold, 1991.
7. Sih, Gilbert C., and Edward A. Lee. "A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures." *Parallel and Distributed Systems, IEEE Transactions on* 4, no. 2 (1993): 175-187.
8. Bashir, A. F., V. Susarla, and K. Vairavan. "A statistical study of the performance of a task scheduling algorithm." *Computers, IEEE Transactions on*100, no. 8 (1983): 774-777
9. Kwok, Yu-Kwong, and Ishfaq Ahmad. "A Parallel Genetic-Search-Based Algorithm for Scheduling Arbitrary Task Graphs to Multiprocessors." In*Proceedings of the 9th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'97)*, pp. 245-248. 1997.
10. Ahmad, Ishfaq, and Yu-Kwong Kwok. "On parallelizing the multiprocessor scheduling problem." *Parallel and Distributed Systems, IEEE Transactions on*10, no. 4 (1999): 414-431.
11. Kwok, Yu-Kwong, and Ishfaq Ahmad. "Static scheduling algorithms for allocating directed task graphs to multiprocessors." *ACM Computing Surveys (CSUR)* 31, no. 4 (1999): 406-471.
12. Leung, Joseph YT, and Gilbert H. Young. "Minimizing schedule length subject to minimum flow time." *SIAM Journal on Computing* 18, no. 2 (1989): 314-326.
13. Sinclair, J. B. "Efficient computation of optimal assignments for distributed tasks." *Journal of Parallel and Distributed Computing* 4, no. 4 (1987): 342-362.
14. Khan, A. A., Carolyn L. McCreary, and M. S. Jones. "A comparison of multiprocessor scheduling heuristics." In *Parallel Processing, 1994. Vol. 1. ICPP 1994. International Conference on*, vol. 2, pp. 243-250. IEEE, 1994.
15. Kaur, Kamaljit, Amit Chhabra, and Gurvinder Singh. "Heuristics based genetic algorithm for scheduling static tasks in homogeneous parallel system."*international journal of computer science and security* 4, no. 2 (2010): 183-198.

