International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 7, July 2013 ISSN 2319 - 4847

Implementation of Dynamic Level Scheduling Algorithm using Genetic Operators

Prabhjot Kaur[1] and Amanpreet Kaur[2]

[1, 2] M. Tech Research Scholar Department of Computer Science and Engineering
Guru Nanak Dev University, Amritsar, Punjab, India

ABSTRACT

Scheduling and mapping of precedence constrained task graphs to the processors is one of the most crucial problems in parallel computing. Due to the NP- completeness of the problem, a large portion of related work is based on heuristic approaches with the objective of finding good solutions within a reasonable amount of time. The major drawback with most of the existing heuristics is that they are evaluated with small problem sizes and their scalability is unknown. In this paper, I have implemented APN Dynamic Level Scheduling algorithm by using genetic operators for task scheduling in parallel multiprocessor system including the communication delays to reduce the completion time and to increase the throughput of the system. The parameters in this paper I have used are makespan time, processor utilization and scheduled length ratio. The graphs show better results of dynamic level scheduling with genetic operators as compared to simple dynamic level scheduling algorithm.

Keywords: DLS algorithm, makespan time, processor utilization, schedule length ratio, genetic operators

1. INTRODUCTION

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 core to execute a program or multiple computational threads. Ideally, parallel processing makes programs run faster because there are more engines (CPUs or Cores) running it [2] [3]. Parallel processors are computer systems consisting of multiple processing units connected via some interconnection network and the software needed to make the processing units work together [4].

Parallel processing platform exist in a wide variety of forms [5], [6]. The most common forms are the shared-memory and message-passing architectures. Examples of Shared- memory machines are the BBN Butterfly, the KSR-1 and KSR-2. And examples of Message-passing architectures are Intel iPSC hyper cubes [6], Intel Paragon meshes and CM-5.

2. RELATED WORK

- Dynamic Task Scheduling using Genetic Algorithms for Heterogeneous Distributed Computing has been developed to dynamically schedule heterogeneous tasks on heterogeneous processors in a distributed system. The scheduler operates in an environment with dynamically changing resources and adapts to variable system resources. It operates in a batch fashion and utilises a genetic algorithm to minimise the total execution time. The existing schedulers are compared to six other schedulers, three batch-mode and three immediate-mode schedulers. Simulations with randomly generated task sets, using uniform, normal, and Poisson distributions, whilst varying the communication overheads between the clients and scheduler have been performed. Results are very efficient.
- Another venue of further research is to extend the applicability of the proposed parallel genetic algorithm. While they are targeted to be used in APN scheduling the algorithms may be extended to handle BNP and UNC scheduling as well. Some efficient algorithmic techniques for scheduling messages to links need to be sought lest the time complexity of the algorithm increase.

3. PROBLEM DEFINITION

In general, parallel programs are made up of several smaller tasks that depend on each other. This dependency gives us the precedence relationship among tasks. It can be modelled by using a directed acyclic graph (DAG) called a task graph. A DAG represents set of nodes and directed edges. A node represents task and an edge corresponds to communication messages and precedence constraints among the nodes. A task can only start execution after all the data have arrived from its predecessors in the task execution time. The problem definition is implementing a dynamic level scheduling algorithm with the genetic operators to achieve the minimum makespan time, schedule length ratio and

highest processor utilization to find the best optimal result in all generations and get above parameters by applying genetic operators.

4. PROPOSED DYNAMIC LEVEL SCHEDULING ALGORITHM WITH GENETIC OPERATORS

Genetic Algorithm is guided random search technique that mimics the principles of evolution and natural genetics. It searches optimal solution from entire solution space and from a pool of questionnaire data. The main disadvantage of this process is to spend much time for schedule. So to overcome such type of problem a genetic algorithm is proposed to overcome such type of drawback in realistic world according to the user's satisfactions and conditions. Here the problem can be solved easily by using the power of genetic algorithm in the field of parallel and distributed computing system processor. The DLSG algorithm uses a quantity called Dynamic Level DL(ti,pj), which is the difference between the maximum sum of computation costs from task t*i* to an exit task, and the earliest start execution time of t*i* on processor pj. The DLSG algorithm does not schedule tasks between two previously scheduled tasks. At each scheduling step, the algorithm evaluates the DL values for all combinations of ready tasks and available processors. The best and fittest chromosomes are selected to reproduce offspring.

Algorithm

1. Initialize ready node list by inserting all entry node in the task graph. The list is ordered according to node priorities with the highest priority node.
2. While ready node list is not empty do
3. N(i) first node in the list.
4. Call Generate Schedule
5. Repeat step 6 to 9 generations number of times.
6. Call crossover
7. Call mutation
8. Compute the fitness value of each chromosome store fitness value.
9. Call reproduce
10. Compute fitness value. Store result in fitness value select chromosome with best fitness.
11. End while.

Generate Schedule

1. Schedule n(i) to the processors which gives highest node.
2. Calculate start time of node, Routing table maintain for each processor.
3. Each entry of table indexed by destination processing element.
4. Routing table will direct message from one machine to along a path with minimum communication time.
5. Shortest path with between the processing elements are stored in routing table.
6. Message is sent, the route from source to destination machine become busy, carrying message of certain amount of time.
7. When message is received its route become free and this route can be used for other processor for transmission of message again.
8. Every time a message sent and message received event is processed the routing table will be updated to profile the direction for fastest communication routes at any time.

Crossover

Perform the crossover operation of two chromosome pair A and B.
C1: Select crossover Generate a random number crossover between 0 and the maximum node of task graph.
C2: Do C3 for each processor P_i in chromosome A and B.
C3: Find node n_i in the processor P_i that has Highest priority node and n_j is the node following n_i, where $c = n_i < n_j$ highest for i.
C4: Do C5 for each processor P_i in chromosome A and B.
C5: Using the crossover site selected in C3 exchange the part of chromosome A and B.

Mutation

Mutate a chromosome to form new chromosome.
M1: Randomly pick node n_i
M2: Randomly pick node n_j
Form a new chromosome by exchange the two nodes n_i and n_j in the chromosome.

Reproduce

Population of chromosome Pop and generate new population NEWPOP.

R1: Construct roulette wheel: Let Fitness sum be the sum of all the fitness value of chromosome in POP. Form fitness_ sum slot and assign chromosome to occupy number of slots according to their fitness value of the chromosome.

R2: Select the first chromosome in POP with highest fitness value. Add this chromosome to NEWPOP.

R3: Let NEWPOP be the number of chromosome in POP. Repeat R4 NEWPOP -1 time.

R4: Generate Random number between 1 and fitness_sum. Use it to indexbinto the slots to find the corresponding chromosome. Add the chromosome to NewPop.

5. PERFORMANCE EVALUATION

5.1. Graph for schedule length ratio

The lesser the values of SLR the more efficient is the algorithm, but the SLR cannot be less than the Critical Path values.

Scheduled Length Ratio = Makespan /Critical path

SLR value should be less as the numbers of nodes are increases its makespan time must be decreases. Due to this effect as per number of node are increased its SLR ratio should be decrease. Above graph shows when nodes are 36 the slr value is 5.029 and then at node 40 and 42 there is slight increase in slr value but after that slr value decreases when number of nodes increases. At node 70 slr value is least 4.607.

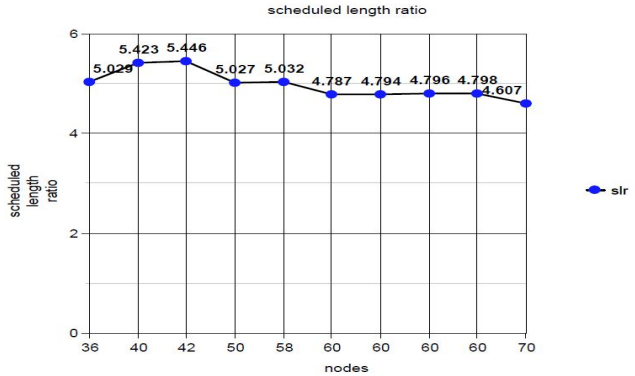

Figure 5.1 schedule length ratio graph for 8 nodes

5.2. Graph for makespan time

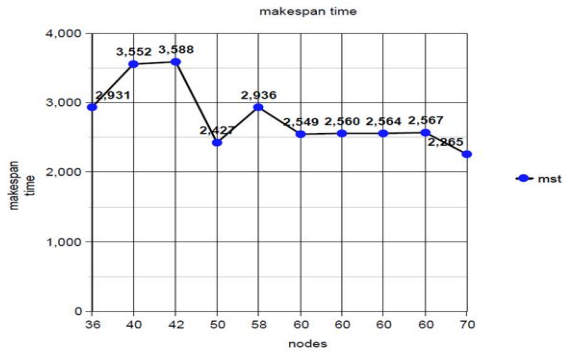

Figure: 5.2 makespan time graph for 8 nodes

In this experiment, only the proposed scheduling order was used, which meant tasks with higher priority level were scheduled first. Apparently, algorithm resulted as with the increasing the number of nodes (tasks) the makespan time decreases because there is arbitrary number of processors and these nodes are assigned to arbitrary processors in mesh network. As number of nodes increases, the more processors are assigned and if more processors are assigned, the makespan time naturally decreases. Due to this concept the above graph shows makespan time decreases as nodes increases. As graph shows for 36 nodes makespan time per generation is 2931 then as number of nodes increases makespan time decreases. For 70 nodes which is highest the makespan time is least that is 2265.

5.3. Graph for processor utilization

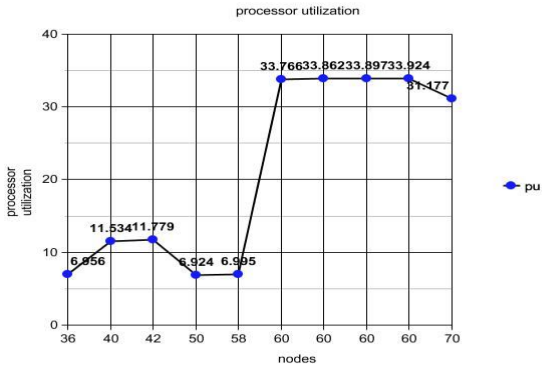

Figure: 5.3 processor utilization graph for 8 nodes

Processor utilization is the most important aspect of determining the performance of algorithm. The greater the processor utilization the more efficient is the algorithm. In multiprocessor system, processor work in parallel. There can be some scenario when large amount of work is done by one processor and lesser by others, which the work distribution is not proportionate. Processor utilization measure the percent of time for which the processor performed. It is calculated by dividing the execution times of the tasks scheduled on the processor with the makespan time of the algorithm. So when execution time is calculated with the makespan time with the increasing number of nodes, its processor utilization decreases. At 36 nodes, processor utilization is approx 7% but as nodes increases to 50 processor utilization decreases because with increasing nodes there is more work has to be done by single processor. At nodes 60 processor utilization increases by 34% and at nodes 70 decreases.

6. CONCLUSION

In this study we have proposed Genetic Algorithm (GA) for dynamic level scheduling to minimize the makespan time, schedule length ratio to increase the throughput of the system. The processor utilization depends upon the problem size therefore their value varies. The proposed method found a better solution for assigning the tasks to the heterogeneous parallel multiprocessor system. The performance study is based on the best randomly generated schedule of the Genetic Algorithm.

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.] Thomas g. Price, "An analysis of central processor scheduling in multi programmed computer systems", digest edition, October 1972.

[5.] H. J. Siegel, et al., "PASM: A Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition," IEEE Transactions on Computers, vol. C-30, no. 12, Dec. 1981, pp. 934-947.

[6.] J. P. Hayes and T. Mudge, "Hypercube Supercomputers," Proceedings of the IEEE, vol. 77, no. 12, Dec. 1989.

[7.] Introduction to Parallel Computing Author: Blaise Barney, Lawrence Livermore National Laboratory

[8.] I. Ahmed, Y. K. Kwok and M. Y. Wu, "Analysis, Evaluation, and Comparison of Algorithms for Scheduling Task Graphs on Parallel Processors", International Symposium on Parallel Architectures, Algorithms and Networks, Beijing, China, pp.207-213, Jun, 1996.

[9.] Yu-Kwong Kwok, "Benchmarking and Comparison of the Task Graph Scheduling Algorithms", Journal of Parallel and Distributed Computing 59, pp.381-422, 1999.

[10.] L. D. Davis (Ed.), The Handbook of Genetic Algorithms, New York, Van Nostrand Reinhold, 1991.

[11.] G. C. Sih and E. A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection Constrained Heterogeneous Processor Architectures", IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 2, Feb. 1993, pp. 75-87.

[12.] F. Gruau, G. R. Joubert, F. J. Peters, D. Trystram and D. J. Evans, "The Mixed Parallel Genetic Algorithm", Parallel Computing: Trends And Applications (Proc. of the International Conference ParCo' 93), 1994, pp. 521-524.

[13.] Gilbert C. Sih and Edward A. Lee, "Dynamic-Level Scheduling For Heterogeneous Processor Networks".

[14.] Yu-Kwong Kwok and Ishfaq Ahmad, "A Parallel Genetic-Search-Based Algorithm for Scheduling Arbitrary Task Graphs to Multiprocessors".
[15.] Ishfaq Ahmad and Yu-Kwong Kwok, "On Parallelizing the Multiprocessor Scheduling Problem".
[16.] Heinz M_uhlenbein, "Evolution in Time and Space- The Parallel Genetic Algorithm".
[17.] Albert Y. Zomaya and Yee-Hwei Teh, "Observations on using Genetic Algorithms for Dynamic Load-Balancing, IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 9, Sep 2001.
[18.] Jasbir Singh and Gurvinder Singh, "Improved Task Scheduling on Parallel System using Genetic Algorithm", International Journal of Computer Applications (0975 – 8887) vol 39, no.17, Feb 2012.
[19.] Kamaljit Kaur, Amit Chhabra and Gurvinder Singh, "Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System", International Journal of Computer Science and Security (IJCSS), Volume (4): Issue (2).

AUTHOR

Prabhjot Kaur (prabhjot.kaur16@gmail.com) received the B.Tech degree in Computer Science and Engineering from Guru Nanak Dev Engineering College Ludhiana in 2011 and M.Tech degree in Computer Science and Engineering from Guru Nanak Dev University Amritsar in 2013 respectively.

Amanpreet Kaur (preet5189@yahoo.com) received the B.Tech degree in Computer Science and Engineering from DAVIET Jalandhar in 2011 and M.Tech degree in Computer Science and Engineering from Guru Nanak Dev University Amritsar in 2013 respectively.