International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online) 2249 –071X, ISSN (Print): 2278 – 4209

Improvised MTTF in Parallel Processing on the Basis of MINs

Harpreet Kaur
Asst. Professor
M.Tech. (CSE)
SBBSIET College, Jalandhar
er.harpreetarora@gmail.com

Amita Sharma
M.Tech. (CSE)
SBBSIET College,
Jalandhar
sharma.3014@gmail.com

***Abstract* — Parallel program represented by an edge-directed acyclic graph (DAG) based on homogenous processors. The objective is to minimize the execution time, evaluate and compare the performance of the individual algorithms and select the best algorithm. For this purpose various tasks scheduling algorithms are available like BNP, UNC, TDB and APB. All the models of algorithms are used in various cases where BNP is used for bounded number of processors i.e. in this case the number of processors is fixed, UNC is for unbounded number of processors i.e. where the number of processors is not fixed and it changes. For mapping the tasks onto to processor in a parallel system, DAG plays an important role. DAG represents the tasks and their computational times along with the communications times in between them. It makes easy representation of the task in graphical fashion. The execution times of tasks on multiple processors depicted using Gantt charts, help in analyzing the starting and ending times of multiple tasks. The performance factor of task scheduling algorithms is calculated using various measures which includes the Makespan, Average processor utilization, SpeedUp and the Scheduled Length Ratio. All these factors help in effectively measuring the performance of algorithms.**

***Keywords* — DAG, Multiprocessor, Parallel Processing, Task Graph, List Scheduling.**

I. INTRODUCTION

Parallel processing is a computing architecture within a single computer that performs more than one operation at the same time. The computer resources can include a single computer with multiple processors, or a number of computers connected by network, or a combination of both. The processors access data through shared memory. Some parallel processing systems have hundreds of thousands of microprocessors. With the help of parallel processing, a number of computations can be performed at once, bringing down the time required to complete a project. There are basically two reasons for the popularity of parallel processing. First is the declining cost of computer hardware that has made it possible to assemble parallel machines with hundreds of processors. Secondly, the availability of applications which are beyond the capability of conventional computers. Weather forecasting, airplane and space vehicle modeling, genetic engineering, interactive graphics, and managing large databases are some of the applications of parallel processing. Without using parallel processing, these applications would not been possible.

It can also be achieved by using multiple computers clustered together to process one part of a large function simultaneously to obtain results faster. Given an edge directed acyclic graph (DAG), also called task graph, the problem of scheduling it to a set of homogenous processors to minimize the completion time. DAG is generic model of a parallel program consisting of a set of processes. Each process is an indivisible unit of execution, expressed by node. A node has one or more inputs and can have one or more output to various nodes. The main problem is the scheduling of tasks onto multiple processors and how their performance is analyzed, selecting the parallel processor scheduling algorithm, how to schedule tasks on the processors, how can performance of algorithm be analyzed, on which criteria the performance can be calculated, and selecting the best algorithm. Figure1 shows the multiprocessor algorithm classification.

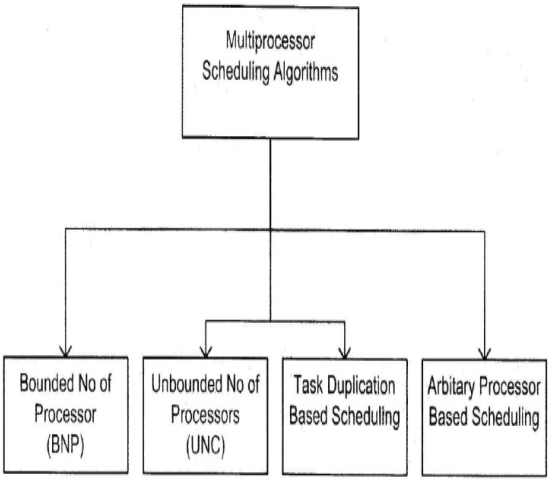


Fig.1. Multiprocessor Algorithm Classification

These algorithms can be classified into following categories:

A. BNP Scheduling Algorithm

BNP stands for Bounded Number of processors. These algorithms schedule the DAG to a bounded number of processors directly. The processors are assumed to be fully-connected. Most BNP scheduling algorithms are based on the list scheduling technique. List scheduling is a class of scheduling heuristics in which the nodes are assigned priorities and placed in a list arranged in a descending order of priority. The main examples of BNP algorithms are the HLFET (Highest Level First with Estimated Times) algorithm, the MCP (Modified Critical path) algorithm, and the DLS (Dynamic Level Scheduling) algorithm and ETF (Earliest Task First) algorithm.

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online) 2249 –071X, ISSN (Print): 2278 – 4209

B. UNC Scheduling Algorithm

UNC stands for Unbounded Number of Processors. These algorithms schedule the DAG to an unbounded number of clusters. The processors are assumed to be fully-connected. The technique employed by these algorithms is also called clustering. The basic technique employed by the UNC scheduling algorithms is called Clustering. The example of UNC scheduling algorithms are LC(Linear Clustering), DSC (Dominant Sequence Clustering) algorithm, the MD (Mobility Directed) algorithm, EZ (Edge-zeroing) algorithm and DCP (Dynamic Critical Path) algorithm.

C. TDB Scheduling Algorithm

The TDB stands for Task Duplication Based scheduling algorithms. The principle behind the TDB algorithms is to reduce the communication overhead by redundantly allocating some tasks to multiple processors [4]. In duplication different strategies can be employed to select ancestor nodes for duplication. Some of the algorithms duplicate only the direct predecessors while some other algorithms duplicate all possible ancestors. The examples of TDB scheduling algorithms are LWB (Lower Bound) algorithm, DSH (Duplication Scheduling Heuristic) algorithm and CPFD (Critical Path Fast Duplication) algorithm.

D. APN Scheduling Algorithm

The APN stands for Arbitrary Processor Network scheduling algorithms. The algorithms in the class take into account specific architectural features as the number of processors as well as well their interconnection topology. These algorithms can schedule tasks on the processors and messages on the network communication links. Scheduling of messages may be dependent on the routing strategy used by the underlying network. The mapping, including the temporal dependencies, is therefore implicit without going through a separate clustering phase. The examples of APN scheduling algorithms are MH (Mapping Heuristic) algorithms, DLS (Dynamic Level Scheduling) algorithm and BSA (Bubble Scheduling Allocation) algorithm.

II. DAG MODEL

The DAG is generic model of a parallel program consisting of a set of processes among which there are dependencies. Each process is an indivisible unit of execution, expressed by node. A node has one or more inputs and can have one or more output to various nodes. When all inputs are available, the node is triggered to execute. After its execution, it generates its output. In this model, a set of node ($n_1, n_2, n_3 \ldots\ldots n_n$) are connected by a set of a directed edges, which are represented by (n_i, n_j) where n_i is called the Parent node and n_j is called the child node. A node without parent is called an Entry node and a without child node called an Exit node. The weight of a node, denoted by w (n_i), represents the process execution time of a process. Since each edge corresponds to a message transfer from one process to another, the weight of an edge, denoted by $c(n_i, n_j)$,is equal to the message transmission time from node n_i to n_j. Thus, $c(n_i, n_j)$becomes zero when n_i and n_j are scheduled to the same processor because intraprocessor communication time is negligible compared with the inter processor communication time. The node and edge weights are usually obtained by estimations. Some variations in the generic DAG model are described below:

Accurate Model: In an accurate model, the weight of a node includes the computation time, the time to receive messages before the computation, and the time to send messages after the computation. The weight of an edge is a function of the distance between the source and destination nodes, and therefore, depends on the node allocation and network topology. It also depends on network contention which can be difficult to model. When two nodes are assigned to a single processor, the edge weight becomes zero, so as the message receiving time and sending time.

Approximate Model 1: In this model, the edge weight is approximated by a constant, independent of the message transmission distance and network contention. A completely connected network without contention fits this model.

Approximate Model 2: In this model, the message receiving time and sending time are ignored in addition to approximating the edge weight by a constant. These approximate models are best suited to the following situations; (i) the grain-size of the process is much larger than the message receiving time and sending time; (ii) communication is handled by some dedicated hardware so that the processor spends insignificant amount of time on communication; (iii) the message transmission time varies little with the message transmission distance, e.g., in a wormhole or circuit switching network; and (iv) the network is not heavily loaded. In general, the approximate models can be used for medium to large granularity, since the larger the process grain-size, the less the communication, and consequently the network is not heavily loaded. The second reason for using the approximate models is that both the node and edge weights are obtained by estimation, which is hardly accurate. Thus, an accurate model is useless when the weights of nodes and edges are not accurate.

III. TASK SCHEDULING ALGORITHM

A. Task Scheduling Problem

The main aim behind the task scheduling problem is to map nodes (tasks) to multiple processors in such way that it requires least time for the completion of all processes, the task dependencies are satisfied and minimum overall scheduling length is achieved. Moreover it also helps in attaining parallelism by executing multiple tasks simultaneously.

B. Algorithm Categorization

BNP class of algorithms is categorized into two categories:

Static Algorithms:- These algorithms use list scheduling approach. Therefore in static algorithms once

the task prioritization phase is finished then and only then the processor selection phase begins.

1. HLFET Algorithm

It is one of the simplest algorithms. Here the HLFET stands for Highest Level First with Estimated Time.

Algorithm Steps:-

a. Calculate the static Level of the nodes in the DAG.
b. Insert all the nodes into a list and sort the list according to descending order of Static Level of the nodes.
c. While not the end of the list do
- Remove the node n_i from the list.
- Compute the earliest start execution time of n_i for all the processor present in the system.
- Map the node n_i to the processor that has the least early start execution time.

2. MCP Algorithm

MCP stands for Modified Critical Path. It uses the Latest Start Time attribute for mapping the nodes to processors.

Algorithm Steps:-

a. Calculate the Latest Start Time (LST) of all the nodes in the DAG.
b. Insert all the nodes into a list and sort the list according order of Latest Start Time.
c. While not the end of the list do
- Remove the node from the list
- Compute the earliest start execution time of n_i for all the processors present in the system.
- Map the node n_i to the processor that has the least early start execution time.

Dynamic Algorithm: - These algorithms also use list scheduling approach. In Dynamic algorithms both the task prioritization phase and processor selection phase goes on side by side.

1. ETF Algorithm

ETF stands for Earliest Task First. This algorithm computes the earliest execution start time for all nodes and selects one with lowest value for scheduling. In this algorithm the ready node stands for that node which has all its parents scheduled.

Algorithm Steps:-

a. Calculate the Static Level of each node in the DAG.
b. In the beginning the ready node list contains only the entry node.
c. While the ready node list is not empty do
- Compute the earliest start time of all the nodes in the ready node list on each processor.
- Select the node with earliest start time. If two or more nodes have same earliest execution start time values then the node with highest Static Level is selected.
- Map the selected node to the processor.
- Add new ready nodes to the ready node list.

2. DLS Algorithm

DLS stands for Dynamic Level Scheduling. It uses the Dynamic Level attribute for scheduling the nodes.

Algorithm Steps:-

a. Calculate the Static Level of node in DAG.
b. In the beginning the ready node list contains only the entry node.
c. While the ready node list is not empty do
- Compute the earliest start time of every node in the ready node list on each processor.
- Calculate the Dynamic Level of every node in the list
- Select the node with the largest Dynamic Level.
- Schedule the node onto the processor.
- Add new ready nodes to the ready node list.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

The analytical results of the entire algorithms under all three case scenarios are determined. The results include the Makespan time of algorithm in seconds, the Scheduled Length Ratio, and the Processor utilization by algorithms.

Scenario 1: Results obtained when 5 task node graphs

The results obtained from all the algorithms using 5 task nodes are shown in table 1. Here it is clearly observed that the Makespan, SLR and the SpeedUp are throughout same for all the algorithms. There is slight variation in processor utilization by ETF algorithm. But rest of the result is mostly the same.

Algorithm	Makespan	SLR	Speed UP	Processor Utilization		
				P1	P2	P3
HLFET	51	1.275	1.27451	58.82%	58.82%	9.80%
MCP	51	1.275	1.27451	58.82%	58.82%	58.82%
ETF	51	1.275	1.27451	58.82%	9.80%	9.80%
DLS	51	1.275	1.27451	58.82%	58.82%	9.80%

Table 1: Makespan, SLR, SpeedUp and Processor Utilization (P1, P2 and P3) of algorithms with 5 tasks

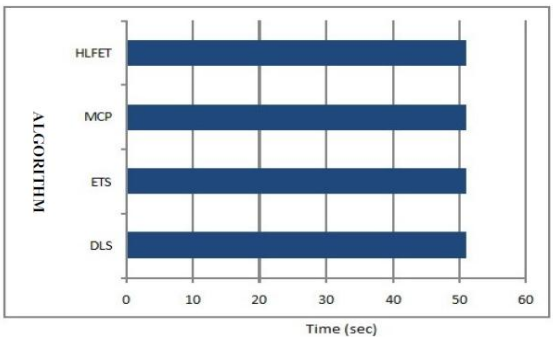


Fig.2. Makespan with 5 nodes

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online) 2249 –071X, ISSN (Print): 2278 – 4209

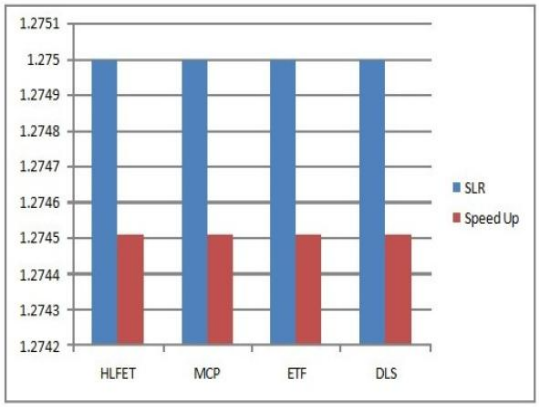


Fig.3. SLR and SpeedUp with 5 nodes

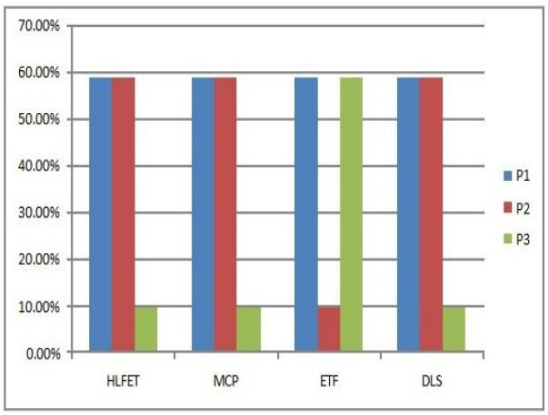


Fig.4. Processor Utilization (P1, P2 and P3) with 5 nodes

Scenario 2: Results obtained when 10 task node graphs

The result obtained from all the algorithms using10 tasks nodes is shown in table5. Here the result is totally different as observed with 5 task nodes. The Makespan of MCP, EFT and DLS algorithm is equal, but the Makespan of HELFET is higher than others. Similarly is the case with SLR is same for MCP, EFT, and DLS algorithms but different in case of HELFET. The same applies to the Speed Up and Processor Utilization.

Algorithm	Makespan	SLR	Speed UP	Processor Utilization		
				P1	P2	P3
HLFET	63	1.8	1.904762	79.37%	63.49%	47.62%
MCP	61	1.742857	1.967213	98.36%	49.18%	49.18%
ETF	61	1.742857	1.967213	98.36%	49.18%	49.18%
DLS	61	1.742857	1.967213	98.36%	49.18%	49.18%

Table 2: Makespan, SLR, SpeedUp and Processor Utilization (P1, P2 and P3) of algorithms with 10 tasks.

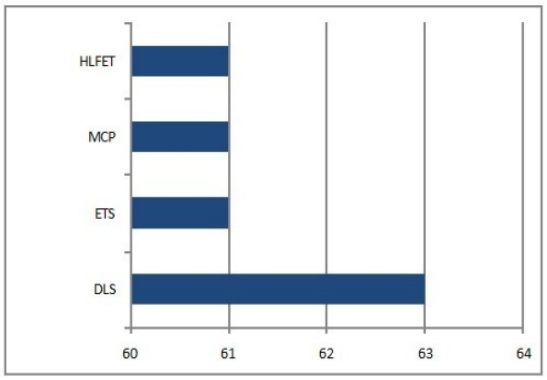
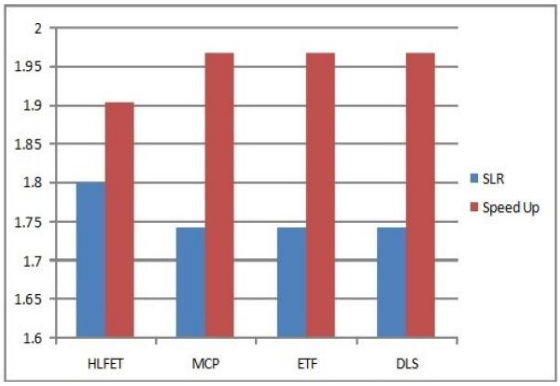


Fig.5. Makespan with 10 nodes



Algorithm

Fig.6. SLR and SpeedUp with 10 nodes

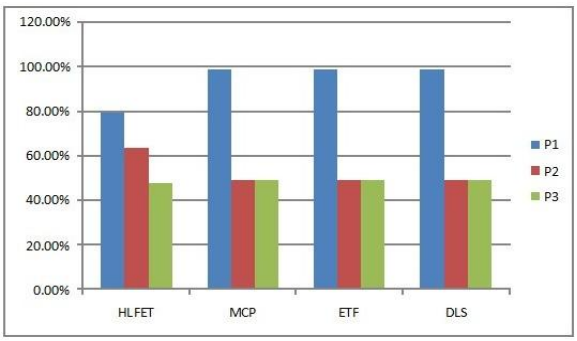


Fig.7. Processor Utilization (P1, P2 and P3) with 10 nodes

V. CONCLUSION

Studying these algorithms shows a wide variety of results. According to these results each of these algorithms provides variable results in different cases.

When tested with 5 tasks:-

- Makespan remains the same for all the algorithms.
- SLR also remains same.
- SpeedUp too remains same.
- Average processor utilization is also the same for all of them.

When tested with 10 tasks:-

- Makespan of MCP, ETF and DLS remains same with HLFET the highest one.
- SLR too remains the same as above with HLFET having the highest.

- These is very small change in SpeedUp with MCP, ETF and DLS having highest SpeedUp and greater than HLFET.
- The average processor utilization also remains highest for the above trio.

It can be concluded from the above results, that DLS is one of the efficient algorithms, considering the data gathered using the scenarios and the performance calculated from them.

REFERENCES

[1] Shyiyuan Jin, Guy Schiavone, Damla Turgut "A performance study of multiprocessor task scheduling algorithm" vol 43, Page 77-97, Issue1 (Jan -2008).

[2] Jorge R.Ramos,Vernon Rego "Efficient implementation of multiprocessor Scheduling algorithm on a simulation tested" SP&E ,Vol 35,Issue 1,pp27-50, Jan 2005.

[3] S.V. Sudha and K.Tjanushkodi "An Approach for parallel job Scheduling Using supple Algorithm" Asian Journal of Information Technology, Volume: 7, Issue: 9, Page No.: 403-407, 2008.

[4] T. Hagras, J. Janecek, "Static vs. Dynamic List-Scheduling Performance Comparison" Acta Polytechnica, 43, No. 6/ 2003.

[5] Ishfaq Ahmad and Min-You Wu, "Analysis, Evaluation and Comparison of Algorithm for Scheduling Task Graph on Parallel Processor" pp 1087-4087, IEEE 1996.

[6] Ishfaq Ahmad and Min-You Wu, "Performance Comparison of Algorithms for Static Scheduling of DAG to Multiprocessors", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8979&rep=rep1&type=pdf

[7] O.C Shih and E.A Lee "A compile-Time Scheduling Heuristic for Interconnection-Constrained heterogeneous Processor Architectures" IEEE Trans. On Parallel and Distributed Systems, Vol. 1, no. 3 pp. 330-343,Jul 1990.

[8] V.Sarkar "Partitioning and Scheduling Parallel Programs for Multiprocessor", MIT Press, Cambridge, MA, 1989.

[9] Eryk Laskowski "Fast Scheduling and Partitioning algorithm in the multiprocessor system with Redundant Communication Resources"

[10] Min You Wu "On parallelization of Static Scheduling Algorithm" IEEE, vol 23, pp 517 – 528, Aug 1997.

[11] Ishfaq Ahmad, Yu-Kwong Kwok "Benchmarking and comparison of the Task Graph Scheduling algorithms" pp1063-7133,IEEE 1998.

[12] Y.C. Chung and S. Ranka, "Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithm on distributed memory multiprocessors," Proc. of Supercomputing'92, pp. 5, 12-521, Nov. 1992.

[13] Parneet Kaur, Dheerendra Singh, Gurvinder Singh and Navneet Singh, "Analysis, Comparison and Performance Evaluation of BNP scheduling algorithms in Parallel Processing," Proceedings of International Journal of Information Technology and knowledge management, January-June 2011, vol. 4, no. 1, pp. 279-284.

AUTHOR'S PROFILE

Harpreet Kaur
Asst. Professor
M.Tech. (CSE)
SBBSIET College, Jalandhar
er.harpreetarora@gmail.com

Amita Sharma
M.Tech. (CSE)
SBBSIET College, Jalandhar
sharma.3014@gmail.com