Global Journal of Computer Science and Technology
Volume 12 Issue 8 Version 1.0 April 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Performance Comparison of BNP Scheduling Algorithms in Homogeneous Environment

By Nidhi Arora Navneet Singh & Parneet Kaur
Maharishi Markandeshwar University

Abstract - Static Scheduling is the mapping of a program to the resources of a parallel system in order to minimize the execution time. This paper presents static scheduling algorithms that schedule an edge-weighted directed acyclic graph (DAG) to a set of homogeneous processors. The aim is to evaluate and compare the performance of different algorithms and select the best algorithm amongst them. Various BNP algorithms are analyzed and classified into four groups - Highest Level First Estimated Time (HLFET), Dynamic Level Scheduling (DLS), Modified Critical Path (MCP) and Earliest Time First (ETF). Based upon their performance considering various factors, best algorithm is determined.

Keywords : *DAG, Task graphs, Parallel Processing, List Scheduling, Multiprocessor, Speed up.*

GJCST Classification: *D.4.8*

Strictly as per the compliance and regulations of:

Performance Comparison of BNP Scheduling Algorithms in Homogeneous Environment

Nidhi Arora [α], Navneet Singh [σ] & Parneet Kaur [ρ]

Abstract - Static Scheduling is the mapping of a program to the resources of a parallel system in order to minimize the execution time. This paper presents static scheduling algorithms that schedule an edge-weighted directed acyclic graph (DAG) to a set of homogeneous processors. The aim is to evaluate and compare the performance of different algorithms and select the best algorithm amongst them. Various BNP algorithms are analyzed and classified into four groups - Highest Level First Estimated Time (HLFET), Dynamic Level Scheduling (DLS), Modified Critical Path (MCP) and Earliest Time First (ETF). Based upon their performance considering various factors, best algorithm is determined.

Keywords : DAG, Task graphs, Parallel Processing, List Scheduling, Multiprocessor, Speed up.

I. Introduction

Parallel processing is the simultaneous use of more than one processor to execute a program in order to get faster results. Given an directed acyclic graph (DAG), also called task graph, in which the nodes represent the tasks and edges represent the communication costs as well as the dependencies among the tasks. The problem deals with the scheduling of the tasks onto 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 paper is organized as follows. In the next section, we describe the generic DAG model and its suitability to different situations. In section 3, basic scheduling attributes are being discussed. Classification of BNP scheduling algorithms is given in section 4. Section 5, presents a performance comparison of various BNP scheduling algorithms and results are derived. Last section concludes the paper and presents the scope of this work in future.

II. Dag Model

The DAG [Kaur et al, 2011][Ahmad and Kwok,1998] 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 nodes $\{ n_1 , n_2 , n_3 \ldots\ldots\ldots n_n \}$ are connected by a set of 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 node without child 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 communication time, 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 interprocessor communication time. The node and edge weights are usually obtained by estimations. Some variations in the generic DAG model are described below:

Accurate Model [Kaur et al, 2011]: 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 [Kaur et al, 2011]: 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 [Kaur et al, 2011]: 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

Author α : Department of Computer Science and Engineering M.M.U, Mullana (Ambala) India. E-mail : er.nidhi152@gmail.com
Author σ : Department of Information Technology Adesh Institute of Engineering & Tech Faridkot-Punjab.
Author ρ : Department of Computer Science and Engineering Adesh Institute of Engineering & Tech Faridkot-Punjab.

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. List Scheduling

Most scheduling algorithms are based on list scheduling technique [Kwok and Ahmad, 1999]. 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 node with higher priority will be examined for scheduling before a node with a lower priority. If more than one node has the same priority, ties are broken using some method. List scheduling consists of two phases:

1. Task prioritizing phase: - In this phase the priority of each node in DAG is computed and assigned.
2. Processor selection:-Each task is assigned processor with minimum execution time.

The two main attributes [Hagras and Janeek, 2003] for assigning priority are the t-level (top level) and b-level (bottom level).

Top level : The t-level of a node n_i is the length of the longest path from an entry node to n_i (excluding n_i). Here, the length of a path is the sum of all the node and edge weights along the path. The t-level is computed recursively by traversing the DAG downward starting from the entry node n_{entry}.

$$\text{t-level}(n_i) = \max(\text{t-level}(n_m) + w_m + c_{m,i})$$

where n_m is predecessors of n_i, w_m stands for computational cost, $c_{m,i}$ stands for communication cost and t-level $(n_{entry}) = 0$. The t-level of n_i highly correlates with n_i's earliest start time, denoted by EST (n_i) , which is determined after n_i is scheduled to a processor.

Bottom level: The b-level of a node n_i is the length of the longest path from node n_i to an exit node. The b-level is computed recursively by traversing the DAG upward starting from the exit node n_{exit}.

$$\text{b-level}(n_i) = w_i + \max(\text{b-level}(n_m) + c_{m,i})$$

where n_m is successor of n_i, w_m stands for computational cost, $c_{m,i}$ stands for communication cost and b-level $(n_{exit}) = w(n_{exit})$.

The b-level of a node is bounded by the length of the critical path. A critical path (CP) of a DAG, is the longest path from an entry node to an exit node.

Static b-level: Some BNP scheduling algorithms do not consider the edge weights in computing the b-level. In that case, b-level does not change throughout the scheduling process, therefore it is called static b-level or static level (SL).

$$\text{SL}(n_i) = w_i + \max(\text{SL}(n_m))$$

where n_m is successor of n_i and $\text{SL}(n_{exit}) = w(n_{exit})$

ALAP start time: The ALAP (As-Late-As-Possible) start time of a node is measure of how far the node's start time can be delayed without increasing the schedule length. It is also known as latest start time (LST).

$$\text{LST}(n_i) = \min(\text{LST}(n_m) - c_{m,i}) - w_i$$

where n_m is successor of n_i, w_m stands for computational cost, $c_{m,i}$ stands for communication cost and $\text{LST}(n_{exit}) = \text{EST}(n_{exit})$.

Dynamic Level: It is the difference of Static level and Earliest Start Time.

Some algorithms assign higher priority to a node with smaller t-level while some algorithms assign higher priority to a node with larger b-level. A priority table is designed for all the nodes in DAG.

IV. Classification of Bnp Scheduling Algorithms

BNP refers to Bounded Number of Processor (BNP) Scheduling Algorithms [Hagras and Janeek, 2003][Kaur et al, 2011]. These algorithms schedule the DAG to a bounded number of processors directly. The processors are assumed to be fully connected. BNP scheduling algorithms are based on the list scheduling technique in which nodes are assigned some priorities. To study these algorithms, homogeneous environment is considered in which processors having same configuration are used for execution. 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. Following are static scheduling algorithms.

Highest Level First with Estimated Times (HLFET) algorithm [Kwok and Ahmad, 1999]**:** It is one of the simplest list scheduling algorithms that uses static b-level as node priority and ignores the communication costs on the edges. Following steps describe the HLFET algorithm in detail:

1. Calculate the static b-level of each node.
2. Make a ready list in descending order of static b-level. The ready list contains only the entry nodes initially. Ties are broken randomly.

Repeat.
3. Schedule the first node in the ready list to a processor that allows the earliest execution, using the non-insertion approach.
4. Update the ready list by inserting the nodes that are now ready.

Until all nodes are scheduled.

Modified Critical Path (MCP) algorithm [Kwok and Ahmad, 1999]**:** This algorithm uses an attribute called ALAP time of a node as a priority. The ALAP time of a node is computed by first computing the length of CP and then subtracting the b-level of the node from it. Therefore, the ALAP times of the nodes on the CP are just their t-levels. Following steps describe the algorithm.

1. Compute the ALAP time of each node.
2. For each node, create a list which consists of the ALAP times of the node itself and all its children in a descending order.
3. Sort these lists in an ascending order. Create a node list according to this order.

Repeat.
4. Schedule the first node in the node list to a processor that allows the earliest execution, using the insertion approach.
5. Remove the node from the node list.

Until the node list is empty.

Dynamic Algorithms: These algorithms also use list scheduling approach. In Dynamic algorithms both the task prioritization phase and processor selection phase goes on side by side. Following are dynamic scheduling algorithms.

The Earliest Time First (ETF) algorithm [Kwok and Ahmad,1999]**:** It computes, at each step, the earliest start times for all ready nodes and then selects the one with the smallest start time, which is computed by examining the start time of the node on all processors exhaustively. The algorithm is described below.

1. Compute the static b-level of each node.
2. Initially, the pool of ready nodes include only the entry nodes.

Repeat.
3. Calculate the earliest start time on each processor for each node in the ready pool. Pick the node-processor pair that gives the earliest time using the non insertion approach. Ties are broken by selecting the node with a higher static b-level. Schedule the node to the corresponding processor.
4. Add the newly ready nodes to the ready node pool.

Until all nodes are scheduled.

Dynamic Level Scheduling (DLS) algorithm [Kwok and Ahmad, 1999]**:** This algorithm uses as node priority an attribute called dynamic level (DL) which is the difference between the static b-level of a node and its earliest start time on a processor. The stepwise description of the algorithm is given below.

1. Calculate the b-level of each node.
2. Initially, the ready node pool includes only the entry nodes.

Repeat.
3. Calculate the earliest start time for every node on each processor. Hence, compute the DL of every node processor pair by subtracting the earliest start time from the node's static b-level.
4. Select the node processor pair that gives the largest DL. Schedule the node to the corresponding processor.
5. Add the newly ready nodes to the ready pool.

Until all nodes are scheduled.

V. Performance Comparison and Results

The performance is the most important factor in every algorithm [Kaur et al, 2011] [Hagras and Janeek, 2003]. In this section, we present performance comparison of above discussed BNP scheduling algorithm. The performance comparison is based upon various comparison metrics discussed below.

Makespan: Makespan is defined as the completion time of the algorithm. It is calculated by measuring the finishing time of the exit task by the algorithm.

Speed Up: The Speed Up value is computed by dividing the sequential execution time by the parallel execution time.

Scheduled length ratio (SLR): It is defined as the ratio of the Makespan of the algorithm to Critical path values of the DAG.

Processor Utilization: It means that how processors are being utilized by different processes. It is good when maximum number processors are utilized.

Above metrices are compared for 10 nodes, 15 nodes, 20 nodes and 25 nodes in homogeneous environment and results are shown graphically.

Case 1 : In first case, results are shown for 10 nodes and 5 processors. Makespan and SLR is same for HLFET, MCP and ETF, but DLS algorithm shows highest makespan value and lowest speed up value. All processors are best utilized in case of HLFET and MCP.

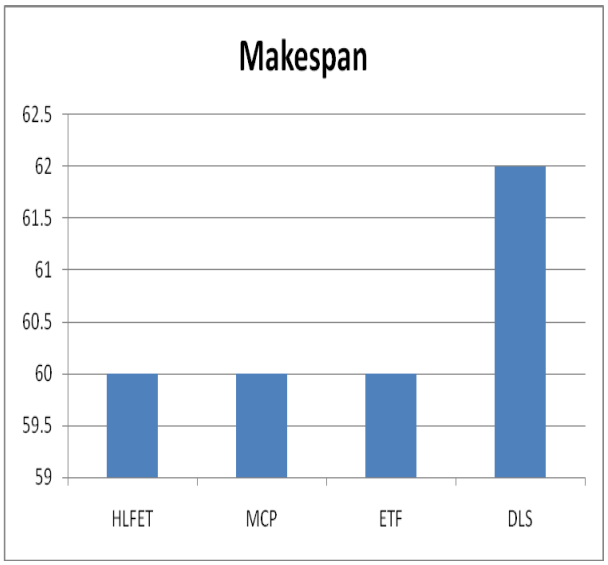

Figure 1: Makespan values for 10 nodes

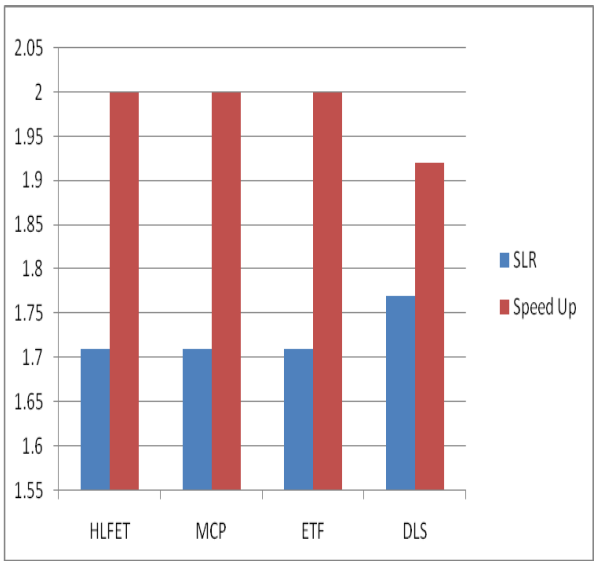

Figure 2 : SLR and SpeedUp for 10 nodes

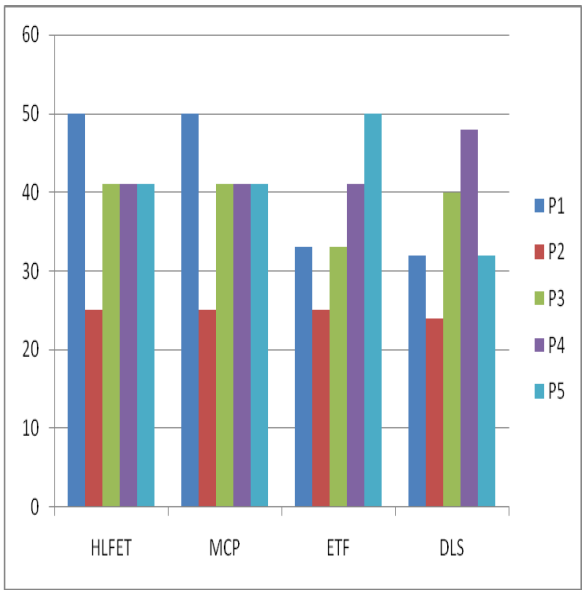

Figure 3 : Processor Utilization for 10 nodes

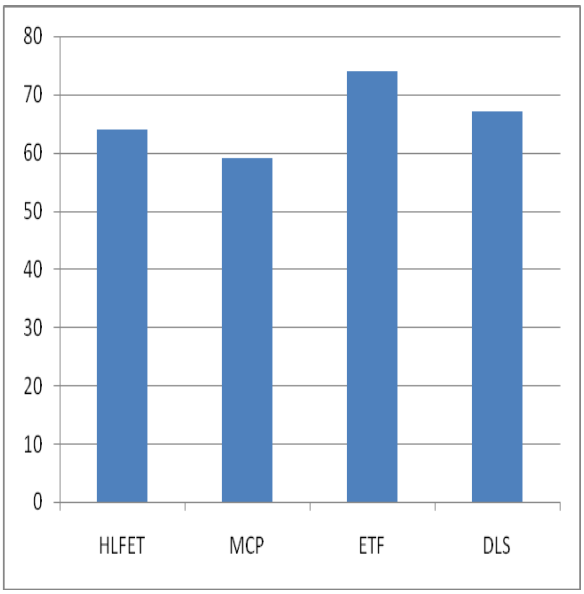

Figure 4 : Makespan for 15 nodes

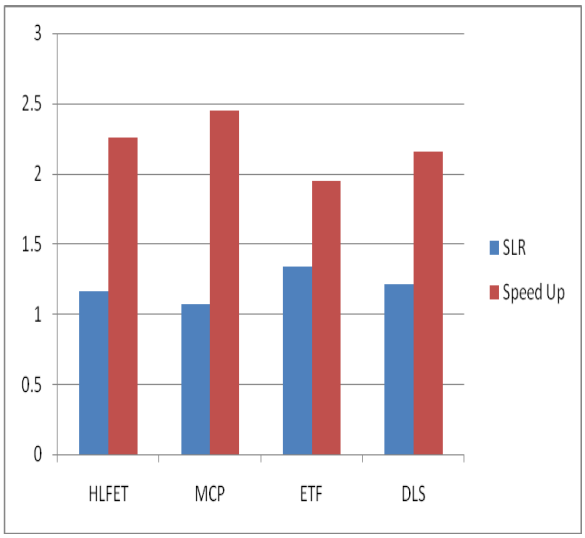

Figure 5 : SLR and SpeedUp for 15 nodes

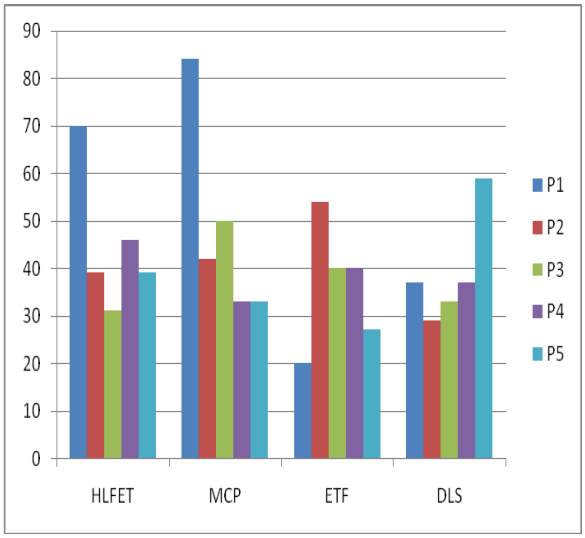

Figure 6 : Processor Utilization for 15 nodes

Case 2: In this case, results are compared for 15 nodes and 5 processors. Makespan is less for MCP and increases in order from HLFET, DLS and ETF. Same results are obtained for SLR values, but processor utilization is best for HLFET and ETF. Speedup is good in case of MCP and HLFET.

Case 3: Here 20 nodes are considered. Makespan time and SLR is less and processor utilization is good in case of DLS. HLFET and DLS shows higher value of Speedup.

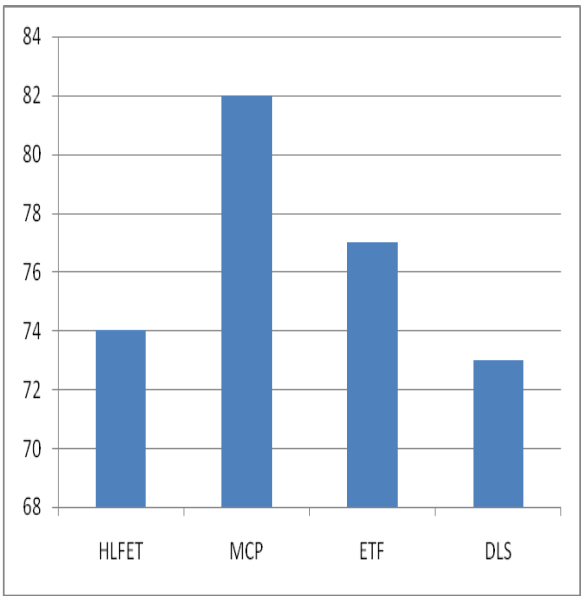

Figure 7 : Makespan for 20 nodes

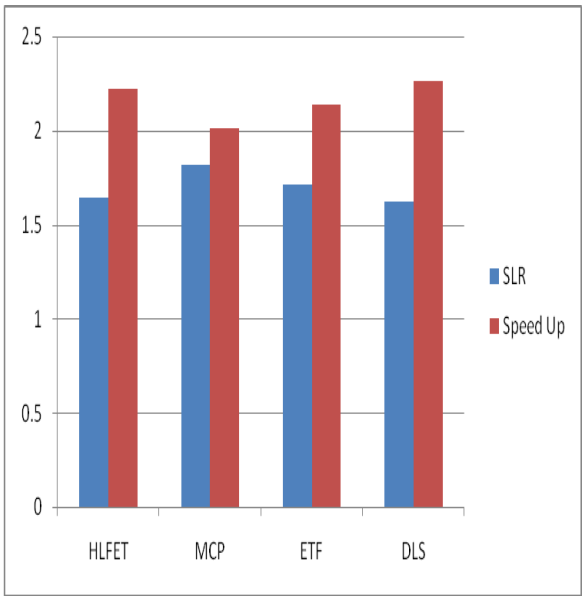

Figure 8 : SLR and SpeedUp for 20 nodes

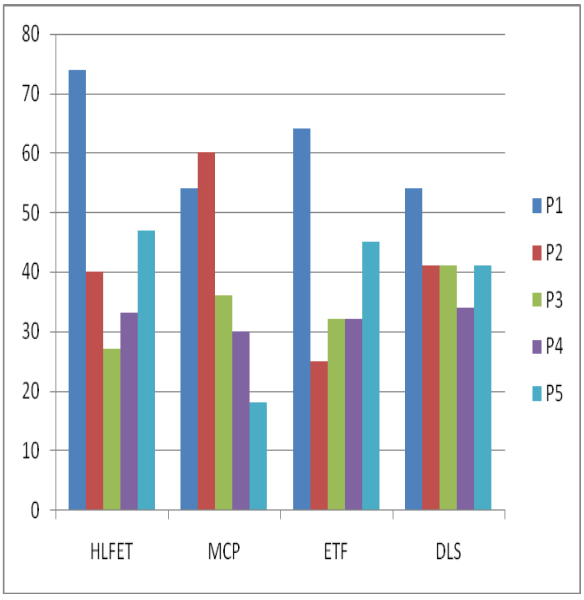

Figure 9 : Processor Utilization for 20 nodes

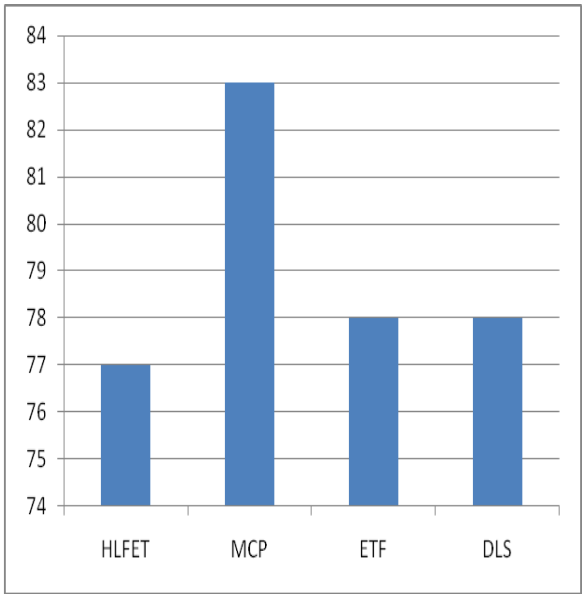

Figure 10 : Makespan for 25 nodes

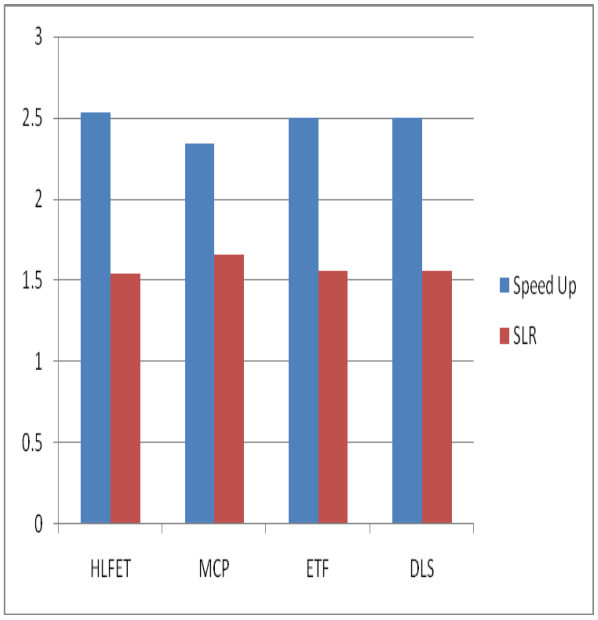

Figure 11: SLR and SpeedUp for 25 nodes

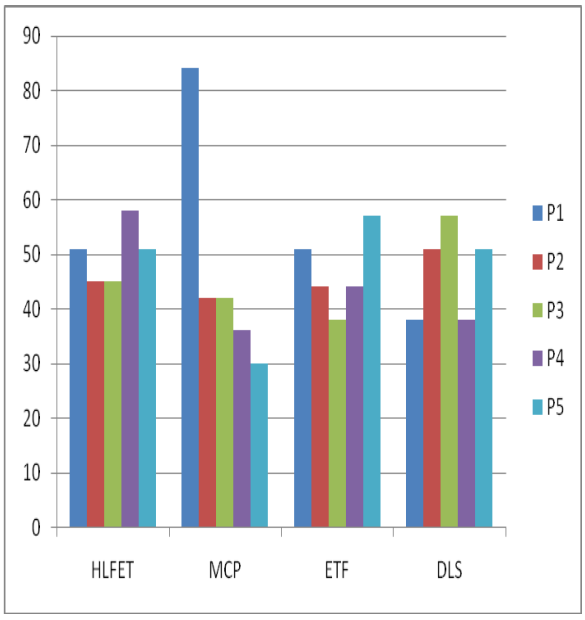

Figure 12 : Processor Utilization for 25 nodes

Case 4: In case of 25 nodes, Makespan time is less for HLFET, same applies for SLR. Processor utilization and Speedup is best in case of HLFET algorithm.

VI. Conclusion and Future Scope

Makespan of MCP and ETF showed large increase in value while increasing the tasks from 20 to 25 compared to other algorithms. The average processor utilization remained same for HLFET and MCP algorithms with 10 tasks. MCP utilized processor efficiently than HLFET with 15 tasks. With 20 and 25 tasks, HLFET proved to be better than other algorithms. The SLR remained almost the same for HLFET, MCP and ETF with 10 tasks. It is maximum in case of DLS for 10 tasks. With 15 tasks MCP shows the lowest value. As the tasks are increased HLFET shows the lesser value in case of 20 and 25 tasks. Same is the case with Speed Up. With 10 tasks speedup of HLFET, MCP and DLS algorithms is same. As the tasks increase from 20 to 25, Speed Up value of HLFET hikes. It can be concluded from the above results, that HLFET is one of the efficient algorithms, considering the data gathered using the scenarios and the performance calculated from them.

The thesis has vast future scope. A lot of work can be done considering more case scenarios. Heterogeneous environment can be considered, in which multiple processors having different configuration are used. The comparison of these algorithms can be done for any number of processors in different environments.

References Références Referencias

1. Parneet Kaur, Dheerendra Singh, Gurvinder Singh and Navneet Singh "Analysis, Comparison and Performance Evaluation of BNP Scheduling Algorithms in Parallel Processing" International Journal of Information Technology and Knowledge Management, Volume 4, No. 1, pp. 279-284, January-June 2011
2. Ishfaq Ahmad, Yu-Kwong Kwok "Performance Comparison of Algorithm for Static Scheduling of DAG to multiprocessor" citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8979...pdf.
3. Ishfaq Ahmad, Yu-Kwong Kwok "Benchmarking and comparison of the Task Graph Scheduling algorithms" pp1063-7133,IEEE 1998.
4. Ishfaq Ahmad and Min-You Wu "Analysis, Evaluation and Comparison Of algorithm for Scheduling Task Graph on Parallel Processor" pp 1087-4087, IEEE 1996.
5. T.Hagras, J.Janeek "Static Vs. Dynamic List-scheduling Performance Comparison" Acta Polytechnica Vol. 43 No. 6/2003.
6. Dror G. Feitelson and Larry Rudolph "Parallel Job Scheduling: Issues and Approaches" www.cse. Hcmut.edu.vn/~ptvu/ppds/ParallelJobScheduling.pdf
7. Min You Wu "On parallelization of Static Scheduling Algorithm" IEEE, vol 23, pp 517 – 528, Aug 1997.
8. Shyiyuan Jin, Guy Schiavone, Damla Turgut "A performance study of multiprocessor task scheduling algorithm" vol 43, Page 77-97, Issue1 (Jan -2008).
9. Thomas L. Casavant "A taxonomy of Scheduling in General-Purpose Distributed Computing Systems" IEEE Transactions on Software Engineering, vol. 14, no. 2, February 1988.
10. T.L Adam, K.Chandy and J. Dickson " A Comparison of List scheduling for Parallel Processing Systems" Communications of the ACM, Vol. 17, no.12, pp 685-690, Dec 1974.

11. Yu-Kwong Kwok and Isfaq Ahmad "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors" ACM Computing Surveys, Vol.31, No.4, December 1999
12. Eryk Laskowski "Fast Scheduling and Partitioning algorithm in the multiprocessor system with Redundant Communication Resources" www.springerlink.com/content/0np00wu1r0c7te8m/fulltext.pdf.
13. Igor Grudenic "Scheduling Algorithm and Support Tools for parallel System" www.fer.hr/_download/repository/Grudenickvalifikacijski.pdf
14. 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.
15. Jing-Chiou Liou, Micheal A.Palis "A Comparison of General Approaches to Multiprocessor Scheduling" IEEE, pp 152 – 156,15 April 1997.

This page is intentionally left blank