ISSN 2277-3061

BNP TASK SCHEDULING ALGORITHMS FOR PERFORMANCE EVALUATION IN PARALLEL SYSTEM

Akanksha[1], Navdeep S. Sethi[2], Nidhi Arora[3], Amit Makkar[4]
Department of Computer Science and Engineering, Adesh Institute of Engineering & *Tec*, *Faridkot*, *Punjab*[1]
E-mail: er.akanksha1988@gmail.com
Department of Computer Science and Engineering, Adesh Institute of Engineering & *Tec*, *Faridkot*, Punjab[2]
E-mail: navdeepsethi@gmail.com
Department of Computer Science and Engineering, MMEC, Mullana, Ambala, Haryana[3]
E-mail: er.nidhi152@gmail.com
Department of Computer Science and Engineering, Adesh Institute of Engineering & *Tec*, *Faridkot*, Punjab[4]
E-mail: amit_mak@yahoo.com

Abstract

Scheduling is the process to minimize the schedule length by proper allocation of the tasks to the processors and arrangement of execution sequencing of the tasks. Multiprocessor Scheduling using Directed Acyclic Graph (DAG) is used in this research. An important implication of minimization of schedule length is that the system throughput is maximized. The objective of this survey is to describe various scheduling algorithms and their functionalities in a contrasting fashion as well as examine their relative merits in terms of performance and time-complexity. In this research, three BNP Scheduling Algorithms are considered namely HLFET Algorithm, MCP Algorithm and ETF Algorithm to calculate effective output by comparing the algorithms with eight test case scenarios with varying number of nodes and processors.

Keywords

DAG; HLFET; MCP; ETF; Scheduling; Performance Evaluation

ISSN 2277-3061

1. Introduction

Even though the area of parallel computing has existed for many decades, programming a parallel system is still a challenging problem, much more challenging than programming a single processor system. With the current dual-core and multicore processors from IBM, AMD, Intel, and others, mainstream PCs have entered the realm of parallel systems. Parallel computing is a serial computing that uses multiple processing units or computers for a common task. Each processing unit works on its section of the problem. Processing units can exchange information and attempts to imitate many complex, interrelated events happening at the same time, yet within a sequence. Historically, parallel computing has been considered to be "the building block of computing" and has been used to tackle difficult problems in many areas of science and engineering.

In parallel computing, the scheduling of tasks with in a parallel program aims to optimize system performance via the efficient arrangement of the tasks onto the underlying available processors within the parallel system which requires least time for the completion of all processes.

2. The DAG Model

In static scheduling, a parallel program can be represented by a directed acyclic graph (DAG) G = (V,E) , where *V* is a set of *v* nodes and *E* is a set of *e* directed edges. A node in the DAG represents a task which in turn is a set of instructions which must be executed sequentially without pre-emption in the same processor. The weight of a node (n_i) is called the *computation cost* and is denoted by $w(n_i)$. The edges in the DAG, each of which is denoted by (n_i, n_j) correspond to the communication messages and precedence constraints among the nodes. The weight of an edge is called the *communication cost* of the edge and is denoted by C (n_i, n_j) . The source node of an edge is called the parent node while the sink node is called the child node. A node with no parent is called an entry node and a node with no child is called an exit node.

2.1 Task Graph Fundamentals

DAG is also known as Task Graph. There are various ways to determine the priorities of nodes. All these priorities are used while evaluating the algorithms (Hagras and Janeek, 2003) and for finding optimal solution.

Top level

Top-level (t-level) of the node n_i in DAG is the length of the longest path from entry node to n_i not including n_i , i.e. the sum of all the nodes computational costs and edges weights along the path.

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

where n_m is the predecessors of n_i , w_m stands for computational cost, $c_{m,i}$ stands for the communication cost and $t-level(n_{entry}) = 0$.

Bottom-level

Bottom-level (b-level) of node n_i in DAG is the length of the longest path from n_i to the exit node, i.e. the sum of all nodes computational costs and edge weights along the path.

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

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

Static Level (SL)

If the edges weights are not taken while considering the b-level, then it is called Static level.

$$SL(n_i) = w_i + \max(SL(n_m))$$

where n_m is the successors of n_i and $SL(n_{exit}) = w(n_{exit})$

Critical Path (CP)

It is the length of the longest path from standing node to the exit node in DAG (Directed Acyclic Graph).

Earliest Starting Time (EST)

Earliest Starting Time is same as the t-level.

$$EST(n_i) = \max(EST(n_m) + w_m + c_{m,i})$$

where n_m is the predecessors of n_i, w_m stands for computational cost, $c_{m,i}$ stands for the communication cost and $EST(n_{entry}) = 0$

Latest Starting Time (LST)

Latest Starting Time of node is computed by following the path starting from exit node upwards till the desired node is reached.

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

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

Dynamic Level (DL)

Dynamic level of the node is calculated by subtracting the Earliest Start Time from the Static Level.

3. BNP Scheduling Algorithms

In this, we discuss three basic BNP scheduling algorithms: HLFET, MCP, and ETF. All these algorithms are for N number of processors.

The main aim behind the task scheduling problem (Jin et al, 2008) 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.

In this thesis three BNP algorithms are studied and their performance is evaluated taking various possible cases. The possible cases are represented by Directed Acyclic Graphs, G = (N,E,C,W) where, N is the set of nodes, W is the set of computation costs of the nodes, E is the set of communication edges, C is the set of communication costs of the edges. In the order to study these algorithms homogenous computing environment is considered, means processors having same configurations are used for execution. If child task is scheduled on the same processor as that of the parent then the communication cost is not considered and if the child task is scheduled on the different processor as that of the parent, then communication cost is taken into account.

3.1 Algorithm Amplification

These entire algorithms are based on 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 [Arora et al, 2012]. If more than one node has the same priority, ties are broken using some method and then repeatedly execute the following two steps until a valid schedule is obtained:

- Select from the list, the process with the highest priority for scheduling.
- Select a resource to accommodate this process.
- If no resource can be found, we select the next process in the list.

BNP class of algorithms (Hagras and Janeek, 2003) is discussed below that are under consideration:

HLFET Algorithm

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

Algorithm Steps (Kwok and Ahmad, 1999)

i. Calculate the static b-level (static level) of each node.

ii. Make a ready list in a descending order of static b-level. Initially, the ready list contains only the entry nodes. Ties are broken randomly.

Repeat

iii. Schedule the first node in the ready list to a processor that allows the earliest execution, using the non-insertion approach.

iv. Update the ready list by inserting the nodes that are now ready. Until all nodes are scheduled.

MCP Algorithm

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

Algorithm Steps (Kwok and Ahmad, 1999)

i. Compute the ALAP time of each node.

ii. For each node, create a list which consists of the ALAP times of the node itself and all its children in a descending order.

iii. Sort these lists in an ascending order. Create a node list according to this order. Repeat

iv. Schedule the first node in the node list to a processor that allows the earliest execution, using the insertion approach.

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 (Kwok and Ahmad, 1999)

i. Compute the static b-level of each node.

ii. Initially, the pool of ready nodes includes only the entry nodes.

Repeat

iii. 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.

iv. Add the newly ready nodes to the ready node pool.

v. Until all nodes are scheduled.

3.2 Performance Evaluation Criteria

The performance is the most important factor deciding the algorithm better from each other (Kaur et al, 2011). Therefore is order to evaluate the performance, some performance metrics are mentioned below:

i. **Makespan**: It is defined as the completion time of the algorithm. Lesser the Makespan less time to execute the algorithm, more efficient is the algorithm. Makespan is calculated by measuring the finishing time of the exit task by the algorithm.

ii. **Processor Utilization**: In multiprocessor system, processor work in parallel. It may happen in some cases that a large amount of work is done by one processor and lesser by others, if 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.

Processor Utilization (%) = (Total time taken of Scheduled tasks/Makespan)*100

iii. **Speed Up**: It is defined as the ratio of time taken by serial algorithm work to the time taken by the algorithm to perform the same work.

Speed Up = Time taken by serial algorithm/Time taken by parallel Algorithm

iv. **Scheduled length Ratio (SLR):** It is defined as the ratio of Makespan of the algorithm to Critical Path values of the DAG (Hagras and Janeek, 2003). 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

4. Comparative Analysis

In this section all performance parameters i.e. Makespan, SLR, Speedup, Processor Utilization are compared and analyzed for given BNP scheduling algorithms.

4.1 Processor Utilization

Processor Utilization is first most important aspect of determining the performance of algorithms. Lesser the use of number of processors, greater the processor utilization and more efficient is the algorithms.

ISSN 2277-3061

Table 1: Processor Utilization of three Algorithms

Algorithm	10N/ 10 P	20N/ 20 P	30N/ 30 P	40 N/ 40P	50N/ 50 P	60N/ 60 P	70N/ 70 P	80N/ 80P	Average
HLFET	6	12	15	29	35	37	47	55	29.5
ETF	7	16	20	31	40	48	56	65	35.375
MCP	7	13	17	26	28	44	50	64	31.125

Where N= NODES, P= PROCESSORS

Processor Utilization of respective case starting from node 10 to node 80 shown in Figure: 1 which efficiently shows the comparison of all the eight task node cases.

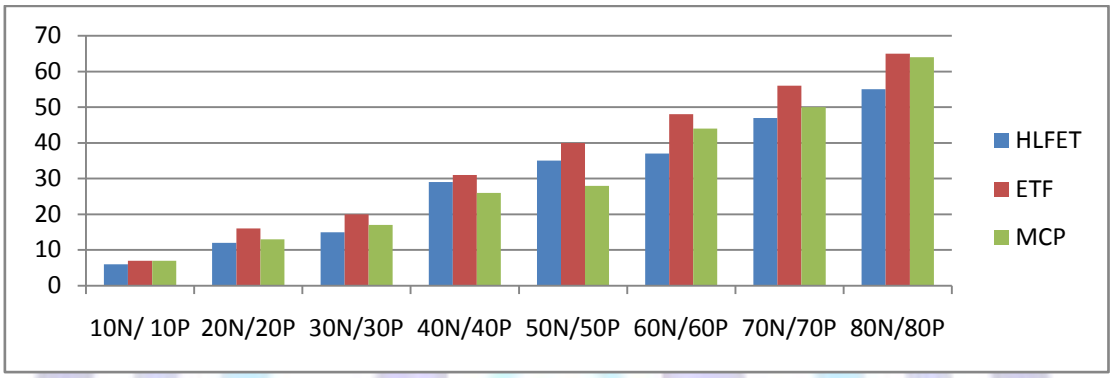

Figure 1: Processor Utilization of 8 cases

The average processor utilization of all the eight task nodes cases 10, 20, 30, 40, 50, 60, 70 and 80 is evaluated and value calculated is shown in the graph. The graph shows that the algorithm with utilization of minimum number of processors is HLFET and is shown graphically in figure 4.35.Hence, HLFET is preferable algorithm.

4.1.1 Average Processor Utilization

It shows the average value of Processor Utilization for all the respective eight cases.

- **Formula to calculate Average is**

  Average= (pu1+pu2+pu3+pu4+pu5+pu6+pu7+pu8)/8

  Where

  pu is maximum processor utilization

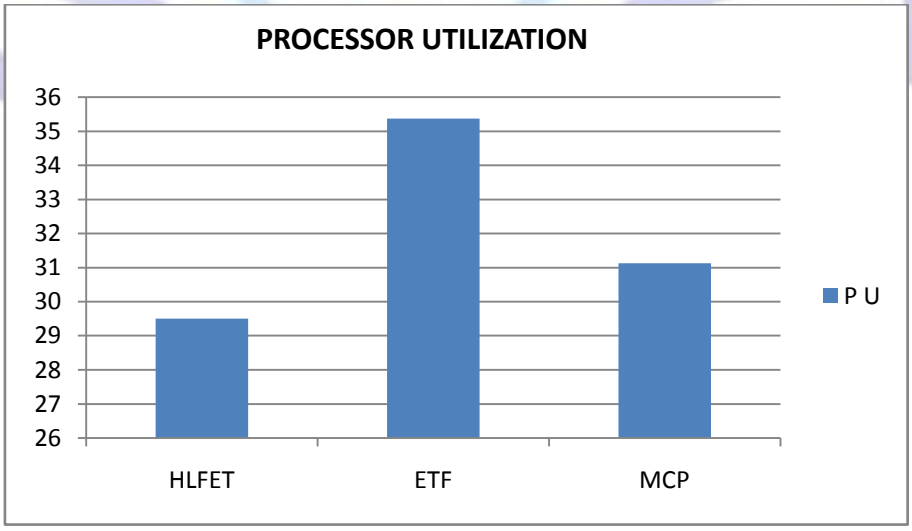

Figure 2: Average Processor Utilization

4.2 Average Makespan, Speedup and SLR

All the related values of eight case scenarios of various algorithms using varying number of processors and nodes are as below:

i) HLFET Algorithm

It is one of the simplest algorithms. Here the HLFET stands for Highest Level First with Estimated Time. The three parameters Makespan, SLR and Speedup is calculated for all the eight cases task node 10, 20 30, 40, 50, 60, 70 and 80 and their respective average value is calculated in the below mentioned Table 2

Table 2: Makespan, Speedup and SLR of HLFET Algorithm

Algo.	Parameter	10N/ 10P	20N/ 20 P	30N/ 30 P	40N/ 40P	50N/ 50 P	60N/ 60 P	70N/ 70 P	80N/ 80P	Average
HLFET	Makespan	60	70	88	110	80	131	117	106	95.25
	SLR	1.71	1.55	1.25	1.15	1.45	1.54	1.46	1.41	1.44
	Speedup	2	2.35	2.84	3.63	4.62	4.58	4.91	6.41	3.9175

Where

N= NODES, P= PROCESSORS

ii) ETF Algorithm

Earliest Time First Algorithm 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 three parameters Makespan, SLR and Speedup is calculated for all the eight cases task node 10, 20 30, 40, 50, 60, 70 and 80 and their respective average value is calculated in the below mentioned Table 3

Table 3: Makespan, Speedup and SLR of ETF Algorithm

Algo.	Parameter	10N/ 10P	20N/ 20 P	30N/ 30 P	40N /40P	50N/ 50 P	60N/ 60 P	70N/ 70 P	80N/ 80P	Average
ETF	Makespan	60	77	104	112	80	146	127	125	103.875
	SLR	1.71	1.71	1.48	1.17	1.45	1.71	1.58	1.66	1.55875
	Speedup	2	2.14	2.4	3.57	4.62	4.1	4.52	5.44	3.59875

Where

N= NODES, P= PROCESSORS

iii) MCP Algorithm

MCP stands for Modified Critical Path. It uses the Latest Start Time attribute for mapping the nodes to processors. The three parameters Makespan, SLR and Speedup is calculated for all the eight cases task node 10, 20 30, 40, 50, 60, 70 and 80 and their respective average value is calculated in the below mentioned Table 4.

Table 4: Makespan, Speedup and SLR of MCP Algorithm

Algo.	Parameter	10N/10P	20N/ 20 P	30N/ 30 P	40N/ 40P	50N/ 50 P	60N/ 60 P	70N/ 70 P	80N/ 80P	Average
MCP	Makespan	60	72	104	106	80	136	120	114	99
	SLR	1.71	1.6	1.48	1.11	1.45	1.6	1.5	1.52	1.49
	Speedup	2	2.29	2.4	3.77	4.62	4.41	4.79	5.96	3.4475

Where

N= NODES, P= PROCESSORS

❖ **Formula to calculate Average is**

Average= (C1+C2+C3+C4+C5+C6+C7+C8)/8

Where C1 to C8 is the respective values from task case1 to task case8

4.2.1 Comparison of Makespan, Speedup and SLR

i) Makespan comparison of three algorithms

Makespan is one of the second most important criteria while evaluating the performance of algorithms. The larger the value of Makespan the more time algorithm takes to execute all the nodes of task graph till its end and vice-versa. It means the algorithm having least values of Makespan is the most efficient of all.

The graph Figure 3 clearly shows the value of makespan for all the eight cases.

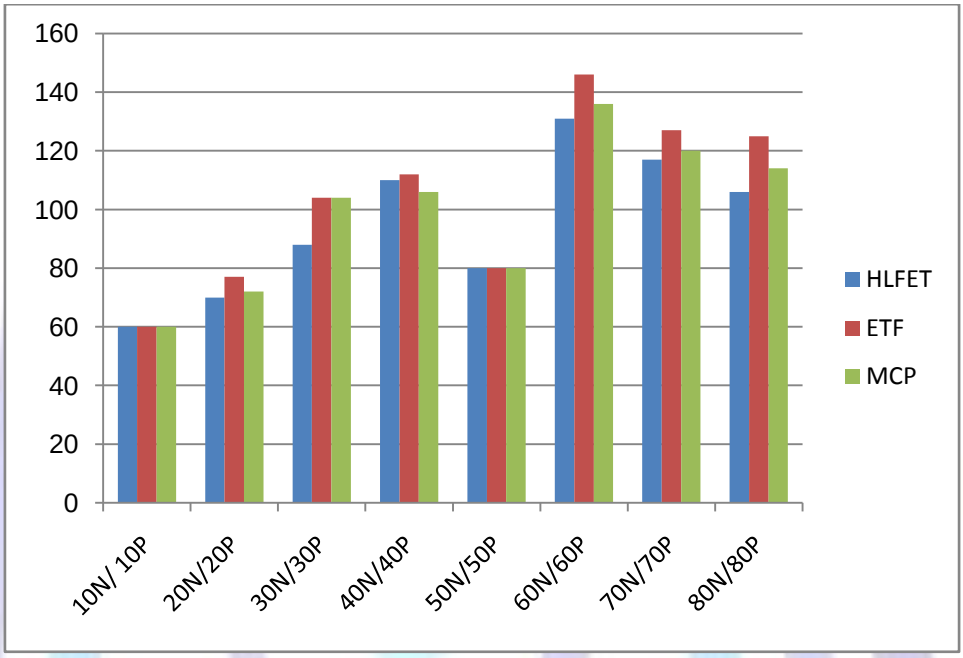

Figure 3: Makespan of 8 cases.

a) Average value of Makespan

Figure 4 shows the average Makespan of the entire algorithm with different nodes cases by calculating it is clear that makespan of HLFET algorithm is less and ETF is maximum.

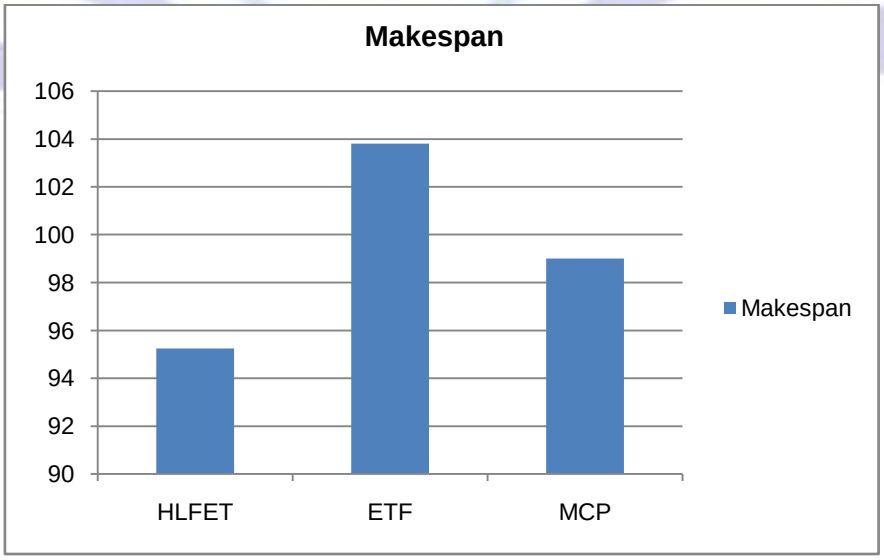

Figure 4: Average value of Makespan

As stated Lesser the Makespan less time to execute the algorithm, more efficient is the algorithm. So HLFET is best algorithm.

ii) SLR comparison of three algorithms

Scheduled Length Ratio is third important aspect. The lesser the values of scheduled length ratio, the lesser is the time taken by the algorithm to execute the entire task by the algorithm till the last. The graph clearly shows the value of SLR for all the eight cases.

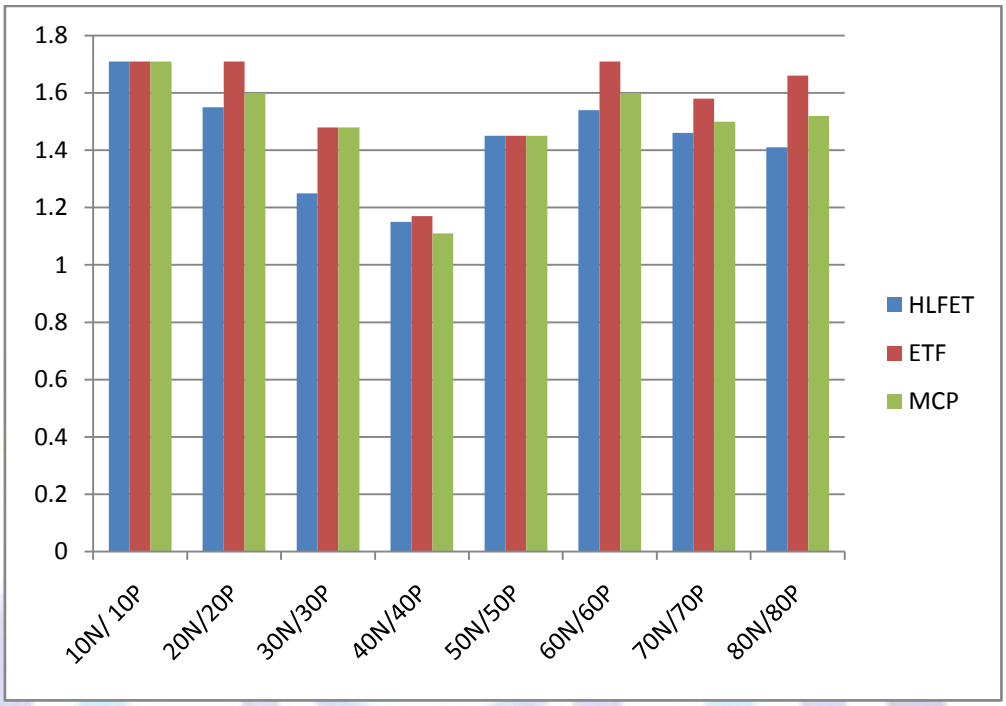

Figure 5: SLR of 8 cases

a) Average value of SLR

Figure 6 shows the average SLR of the entire algorithm with different nodes cases, by calculating it is clear that SLR of HLFET algorithm is less and ETF is maximum.

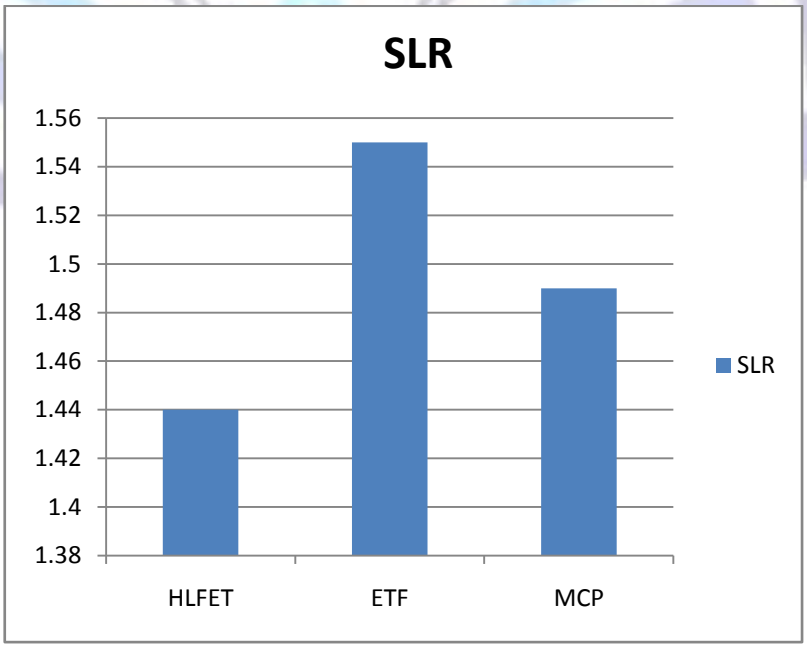

Figure 6: Average value of SLR

As we know lesser the values of SLR the more efficient is the algorithm. So, HLFET is best algorithm.

iii) Speedup comparison of three algorithms

SpeedUp is calculated by dividing the time taken to solve a problem by serial algorithm to the time taken to solve the same problem by parallel algorithms. Here First Come First Serve (FCFS) algorithms are used to determine the time taken to solve the tasks. This time is then divided by the Makespan value which the time taken to solve the task by parallel algorithms, resulting in SpeedUp values.

The graph clearly shows the value of Speedup for all the eight cases.

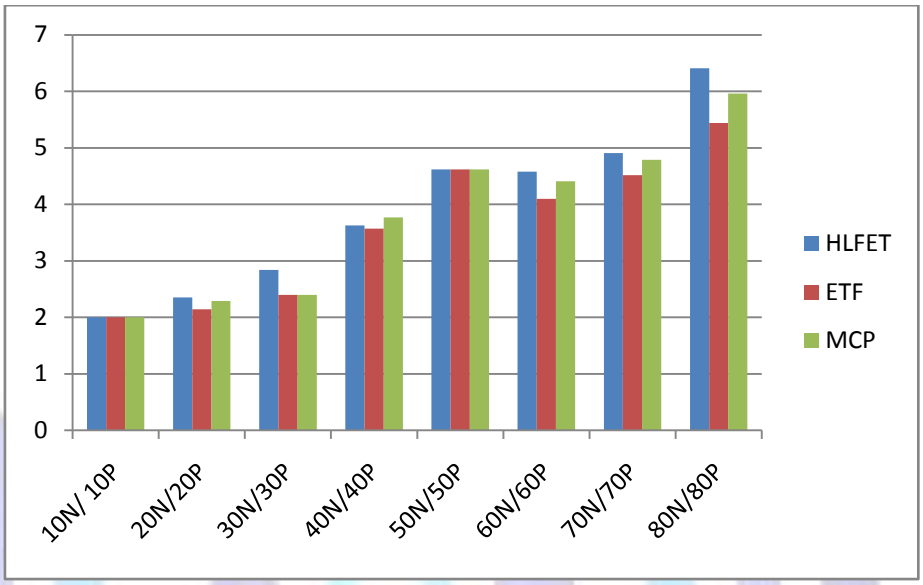

Figure 7: SpeedUp Value of 8 cases

a) Average value of Speedup

Figure 8 shows the average Speedup of the entire algorithm with different nodes cases, by calculating it is clear that Speedup of HLFET algorithm is maximum and MCP is less.

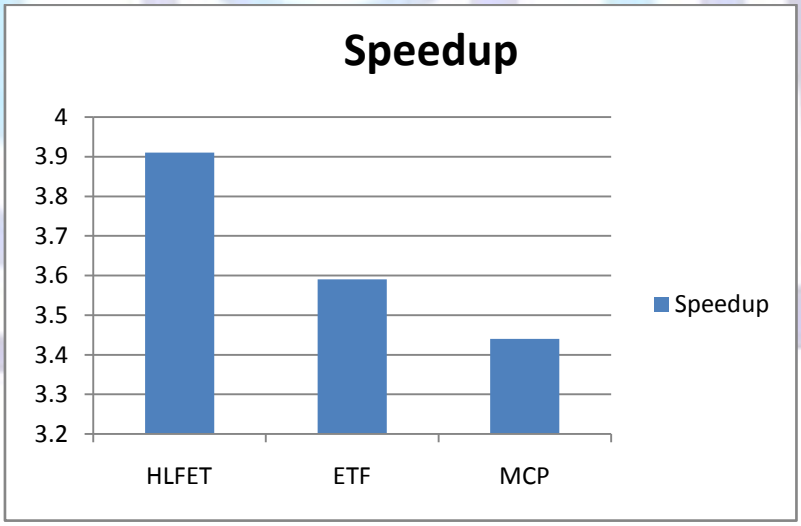

Figure 8: Average value of Speedup

It has been concluded that more is the speed more will be the efficiency. Hence, HLFET is preferred.

5. Conclusion and Future Scope

After comparing the results from all the BNP Scheduling algorithms, it is concluded that **HLFET is one of the efficient algorithms** as it proves to be better in terms of performance with varying number of processors. The research has a vast future scope as parallel computing is the high end of computing in past as well as in present. A lot of work can be done considering more case scenarios:

i. Heterogeneous or Homogeneous environment can be considered with some other scheduling technique.

ii. To extend the current work to distributed heterogeneous supercomputing system (DHSS) which is a suite of machines comprising a variety of sequential and parallel computers provide an even higher level of parallelism.

iii. Various other DAG Scheduling Algorithms and more BNP algorithms can be considered and their performance with other can be estimated.

iv. Further elaboration of various techniques, network topologies and communication traffic can also be considered.

6. References

[1] Arora, N. (2012a) "*Analysis and Performance Comparison of Algorithms for Scheduling Directed Task Graphs to Parallel Processors*", International Journal of Emerging trends in Engineering and Development, vol. 4, Issue 2, pp. 793- 802.

[2] Ahmad, I. and Kwok, Yu-K. (1995) "*Performance Comparison of Algorithm for Static Scheduling of DAGs to multiprocessors*", Proceedings of the Second Australian Conference on Parallel and Real-Time Systems, Perth, Australia, pp. 185-192.

[3] Ahmad, I. and Wu, M. Y. (1996) "*Analysis, Evaluation and Comparison of algorithm for Scheduling Task Graph on Parallel Processor*", IEEE Conference Publications, pp. 1087-4087.

[4] Arora, N. (2012b) "*Comparative study of Task Duplication based Scheduling Algorithms for Parallel Systems*", International Journal of Computer Applications, vol. 58, no. 19, pp. 1-3.

[5] Hagras, T. and Janeek, J. (2003) "*Static Vs. Dynamic List-scheduling Performance Comparison*" Acta Polytechnica vol. 43 no. 6/2003.

[6] Arora, N., Singh, and Kaur, P. (2012) "*Performance Comparison of BNP Scheduling Algorithms in Homogeneous Environment*", Global Journal of Computer Science and Technology, vol. 12, Issue 8, version 1, pp. 41-47.

[7] Hwang, K. and Briggs, F. A. (1984) "*Computer architecture and Parallel Processing*", McGrawHill.

[8] Jin, S, Schiavone, G. and Turgut, D. (2008) "*A performance study of multiprocessor task*" *scheduling algorithm*", Journal of Supercomputing, Springer, vol 43, Issue 1, pp. 77-97.

[9] Kwok, Y. K and Ahmad, I. (1999) "*Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors*", ACM Computing Surveys, vol.31, no.4, pp. 406-471.

[10] Liou, J. C and Palis, M. A. (1997) "*A Comparison of General Approaches to Multiprocessor Scheduling*", IEEE, pp. 152 – 156.

[11] Kaur, P , Singh, D , Singh, G. and Singh, N. (2011) "*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.

[12] Sarkar, V. (1989) "*Partitioning and Scheduling Parallel Programs for Multiprocessor*", MIT Press, Cambridge, MA.

[13] Lin, G. and Rajaraman, R. (2010) "*Approximation algorithm for Multiprocessor scheduling under Uncertainty*", Theory of Computing Systems, vol 47, Issue 4, pp. 856-877.

[14] Silbetschatz, A. and Galvin P. B. "*Operating System Concepts*", Wliley-India Edition.

[15] Shyiyuan Jin, S, Schiavone, S. and Turgut, D. (2008) "A performance study of multiprocessor task scheduling algorithm", Journal of Supercomputing, vol 43, Issue 1, pp. 77-97.