Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 05, May-2016

A Survey of Scheduling Algorithms for Heterogeneous Systems and Comparative Study of HEFT and CPOP Algorithms

Baldeep Singh
M. Tech Student
Dept. of Computer Science and Engg.
UGI Lalru, PTU Jalandhar,
Punjab, India

Priyanka Mehta
Assistant Prof.
Dept. of Computer Science and Engg.
UGI Lalru, PTU Jalandhar,
Punjab, India

***Abstract*- Scheduling computation tasks on processors is the key issue for an advanced computing. List scheduling has constantly been a topic of conversation for the researchers due to its nature of solving high complexity troubles with minimum complexity and to estimate the additional scheduling problems for the applied matrix. The task in hand is to do reduce the overall time utilizes for task completion. A lot of earlier research works have also included prioritization on the jobs to reduce the computation cost and earliest finish time of the system. This paper introduces a discussion about recently studied scheduling techniques namely: HEFT, CPOP. These two techniques are compared with each other in rest of Schedule Length, Speedup and Efficiency on different number of directed acyclic graphs.**

Keywords: CPOP, HEFT scheduling, DAG.

I. INTRODUCTION

Heterogeneous Distributed computing includes resources of varying capacity forming a fast system to execute computationally intensive parallel and simultaneous applications. It becomes necessary to have the capacity to increase the use of computing and communication infrastructure for justifying heterogeneity the expense that may have gone into making of these resources. The computational requests of jobs and applications have been increasing quickly and the presentation of a figure rigorous application on such stages is intensely reliant on the portion of the tasks on these assets. One of the major issues in Heterogeneous Distributed Computing is to schedule the tasks of requests such that general running time is minimized. The task scheduling problem is known to be N-P complete [1, 2]. Many heuristics have been designed to solve the problem but still no satisfactory results have been found. Task scheduling is divided into two classes: static scheduling and dynamic scheduling. In static scheduling, which is done before the task is run, all the data associated with a parallel program, for example, task managing time, communication time and data dependency are already available. In dynamic scheduling, many scheduling decisions are taken on runtime. Hence the target of dynamic scheduling is to generate a schedule with in the scheduling overhead limit, in addition to internal failure Concerns and other aspects [3]. Special scheduling heuristics have been planned in the writing on principle of the methodologies these heuristics use. They have been characterizing into four clusters: List scheduling algorithms [3 4 7 8 10 12 14 18], Duplication based algorithms [15 20], Cluster based algorithms and Random guided search algorithms [4]. In list based heuristics, tasks are placed in a priority list with every task having special priority importance. Priority of a task depends on priority of its predecessor. A current task from the priority list all scheduled onto a suitable processor by using tasks EST and EFT. Clustering heuristics were mostly planned for standardized frameworks and the point is to frame clusters of tasks that are then transfer to processors. Clustering is the best way to minimize the communication delay within DAG's using grouping closely tasks within a cluster on the same processor. Duplication is very useful in parallel processing. Using duplication duplicate the task between idle times slots. The duplication heuristics deliver the most limited makespan yet they have a disadvantage i.e. the higher time complexity. Guided random scheduling techniques make use of the theory of evolution and ordinary genetics to produce near optimal task schedules. These genetic algorithms are very useful and most popular in list based scheduling.

II. TASK MODEL

An application is divided into set of tasks on a parallel and distributed computing environment. That application is represented with help of DAG (Directed Acyclic Graph) G = (V, E). Where V is a set of v nodes and each node $v_i \in V$ represents set of nodes of an application. E is a set of e coordinated edges between tasks, each e (i, j) $\in$ E represent nodes in the graph and their dependencies. Edges e in the DAG speaks to the connection messages and is spoken to as (n_i, n_j). n_i is a protector node or parent node. n_j is child node. n_j is executed after the execution of n_i. The node with no section or child is called exit node [6]. At a time frame n_i nodes needs to send data elements to n_j with the exception that if both the nodes n_i and n_j are assigned to the same processor then there communication cost must be negligible. Mapping of nodes on best processor and for minimizing the execution time is totally based on ranking. Ranking is defined in many ways in literature. Weight w of every node communicates with the vertices to know the processing expenses w (n). W is a (v*q) describe computation cost matrix in each $w_{i,j}$ gives EET (estimated execution time) to complete a task n_i on p_j processor. Average execution cost of an application or task n_i is describe as

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 05, May-2016

$$wi = \sum_{j=1}^{q} w_{i,j}/q \quad (1)$$

Transfer time of data among Processors are stored in a matrix B, and their size is q*q. Processors communication startup time is given in q_dimensional vector L. Task n_i to task n_k, (which is scheduled onto p_m and p_n) is communication cost among edges, describe as

$$C_{i,k}=L+data_{i,k}/B_{m,n} \quad (2)$$

Both the tasks are assigning onto same processor, their inter-processor communication is zero. The average communication cost between edges (i, k) is describe as

$$C_{i,k}=\bar{L}+data_{i,k}/B \quad (3)$$

Where transfer average rate onto processors is defined by and average communication startup time defined by. Now we describe the earliest startup time (EST (n_i, p_j)) AND earliest finish time (EFT (n_i,p_j)) of application ni onto processor p_j, their entry task is

$$EST(n_{entry}, p_j) = 0. \quad (4)$$

In order to calculate the EFT of a task n_i, all immediate parent tasks of n_i must have been scheduled shown in (5) and (6).

$$EST(n_{i,j})=\max\{avail[j],\max/n_m \in pred(n_j) (AFT(n_m)+c_{m,i})\} \quad (5)$$

$$EFT(n_i, p_j) = w_{i,j} + EST(n_i, p_j) \quad (6)$$

Where pred(n_i) is the list of immediate parent applications of task ni and avail[j] is the minimum time which the p_j is ready for execution. This process is not executes in case of all the tasks are assigned to processors.
After the scheduling of all tasks onto available processors, they calculate their EST and EFT i.e. equal to the actual start time (AST) and actual completion time (ACT), or we can say overall completion time (n_{exit}), i.e. schedule length or makespan, sometimes called critical path length. The makespan is describe as

$$Makespan = \max\{AFT(n_{exit})\} \quad (7)$$

Node with higher need is analyzed for Scheduling before a node with lower need. The main objective of scheduling problem is to assigning the tasks of a given application for execution against suitable processors and tries to minimize the schedule length or makespan.

III. SCHEDULING PROBLEM

The problem presented in this paper is the static scheduling of a reserved application in a heterogeneous structure with P set of processors, V set of vertices, E set of edges between two vertices. Overall mathematically it can be explained as G = (V, E) where V is the set of vertices and E is the edge between two vertices. As said above, task scheduling can be separated into Static and Dynamic methodologies [8]. Dynamic scheduling is satisfactory for situation where the framework and task parameter are not known at compile time, which make choices to be made at runtime however with extra overhead. A dynamic algorithm is obliged on the basis that the workload is just known at runtime, similar to the status of every processor when new tasks arrive. As a consequence of this, a dynamic algorithm not ensures that it have all work essentials accessible among scheduling and can't promote in light of the entire workload. By separation, a static methodology can expand a schedule by allowing for all tasks freely of execution request or time in light of the fact that the schedule is created before execution start and present no operating cost at runtime.

The main problem of scheduling is to divide an application into different parts, these parts are known as nodes or jobs, problem is to find the priority of all nodes and fit into best suitable processors for reducing the running time of an application. We discus two list based scheduling heuristics (HEFT and CPOP) in heterogeneous environment and discus different type of scheduling problems with the help of different parameters like: Schedule Length, Speedup, and Efficiency.

IV. IMPLEMENTED ALGORITHMS

I. HEFT (Heterogeneous Earliest Finish Time)

HEFT is a simple and best scheduling technique in static task scheduling in heterogeneous as well as homogeneous environment for limited number of processors. HEFT has two stages: prioritization phase and processor selection stage.
Prioritization stage: first HEFT calculates the priority using upward ranking ($rank_u$). An application is traversed in upward direction and find out the rank of all nodes in a list with the help of mean communication and mean computation cost. Generated list is arranged in decreasing order of $rank_u$.
HEFT uses a Tie breaking policy for selecting the nodes, which node or successor selects whose rank value is highest. Upward rank of task ni is described as:

$$Rank(n_i) = W_i + \max_{nj \in succ(ni)} (c_{ij} + rank_u(n_j)) \quad (8)$$

W_i is the mean computation cost, succ (n_i) is the immediate child of node n_i, $c_{i,j}$ is the mean communication cost of node (i,j). In case of two nodes have equal rank value selects randomly. In upward ranking, graph is traversed from exit node to entry node. Highest rank is same with exit node:

$$rank_u(n_{exit}) = W_{exit} \quad (9)$$

$rank_u(n_i)$ is total critical path from source node to exit node including communication and computation cost of tasks.

HEFT ALGORITHM:
Input a graph along with communication cost computation and number of processors.
Compute the average mean value of communication and computation costs of each node.
Compute upward rank ($rank_u$) by traversing the graph from exit node to entry node.
Generate a priority queue in decreasing order according to their $rank_u$.
While
Unscheduled tasks in the queue Do

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 05, May-2016

Select first rank task for scheduling and remove from queue.
Assign task (n_i) to processor p_j
Schedule all the tasks and compute their EST and EFT.
END.

II. CPOP (Critical Path on Processor)

HEFT is also called CPOP. CPOP used both ranking techniques upward and downward. CPOP computes the rank value of each node by adding both the techniques $rank_u$+ $rank_d$ and set into a queue. An application is traversed from entry node (n_i) to exit node (n_j) is called downward ranking and traverse exit node to entry node is called upward ranking. CPOP has two steps: task prioritization phase and task allocation phase.
In task prioritization phase, tasks are prioritize according to their rank value ($rank_u$+ $rank_d$) with the help of communication and computation costs of DAG then set into a queue (decreasing order). CPOP used critical path (CP) of an application to find the longest path starting from entry node to exit node.
In second phase, tasks are selected according to higher rank value and selects for scheduling to best suitable processor which minimizes the execution time of task. CP nodes are scheduled on a processor which has less mean computation cost then other processors. CP of given DAG (shown in figure 1) is N1, N2, N9 and N10 and their mean computation cost on each processor is 66, 54 and 63 (P1, P2 and P3). CPOP chooses minimum processor cost from all i.e. called CP-P (Critical Path-Processor).

CPOP ALGORITHM:
Input a graph along with communication cost computation and number of processors.
Compute the upward rank ($rank_u$) and downward rank ($rank_d$). Downward rank is computed by traversing the graph from entry node to exit node.
Compute the priority ni = ranku+ rankd and arrange in a list.
|CP|= rank of entry node. [CP-Critical Path]
SETCP = set all nodes on critical path
$n_k \leftarrow n_{entry}$
While
n_k is not exit node do
Select n_j where (n_j€ succ (n_k)) and (priority (n_j== |CP|))
Select the Critical Path Processor (CP-P) $\sum_{ni \in CP\ node} w_{ij}$ w_{ij}← computation cost.
Initialize the priority list with starting node.
While
There are unscheduled nodes in list do
Select the highest rank node from list and ready to schedule then remove from list.
If n_i€ CP node then
Assign task n_i to CP-P
Else
Assign task to processor pj, which reduces the EFT (n_i, p_j)
Update the priority list
End.

These two algorithms are based on insertion based policy; a task is scheduled in processor earliest idle time slot which has already scheduled tasks, that large enough to hold a task. These tasks are schedule on the same processor.

TABLE: 1 computation cost

P1	P2	P3
14	16	9
13	19	18
11	13	19
13	8	17
12	13	10
13	16	9
7	15	11
5	11	14
18	12	20
21	7	10

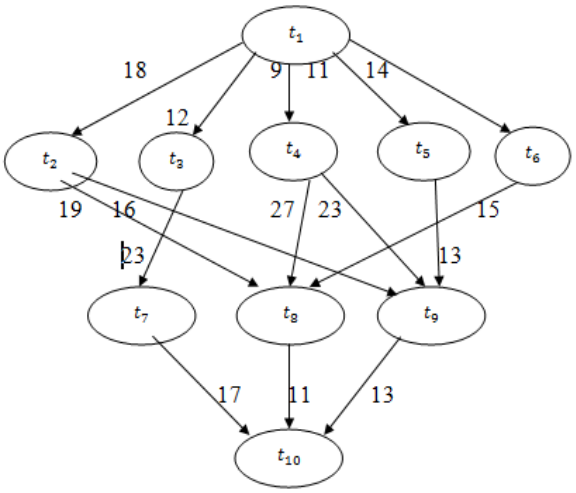


Figure: 1: Directed Acyclic Graph

Table 1 represents the computation cost of each processor on every processor and Figure 1 represents an application shows various type of nodes with their communication cost. Communication is the transfer rate between two nodes on different processors. We discuss this application or DAG on HEFT and CPOP, in heterogeneous environment.

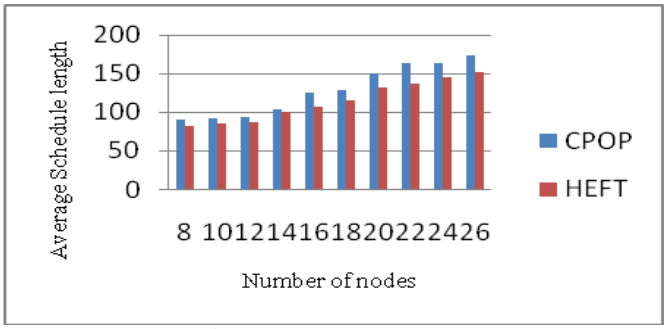
V. EXPERIMENTAL RESULTS AND ANALYSIS

The results are discussed of HEFT and CPOP algorithms under three parameters namely: schedule length, speedup and efficiency.
Comparison metrics: Using these metrics, we discuss comparison between above two algorithms based on:

1. Schedule Length: Schedule length (makespan) is the total execution time of an application or a DAG.
2. Speedup: Speedup is defined as the ratio of given schedule length is divided with obtained fastest processor.
3. Efficiency: speedup is divided with number of processors in each run

We analyze the results on 50 different acyclic graphs with the variation in increasing the number of nodes (8 10 12 14 16 18 20 22 24 26). Performance is increased with increasing the number of nodes.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 05, May-2016



Graph: 1 comparison of HEFT and CPOP w.r.t to average schedule length.

Graph1 shows HEFT gives better results then CPOP for average schedule length. HEFT is 12% more efficient than CPOP. If number of nodes is increased then schedule length is also increased. Graph 2 represents average speedup. HEFT is 15% more efficient then CPOP. Graph 3 represents HEFT is 11% better then CPOP in case of efficiency.

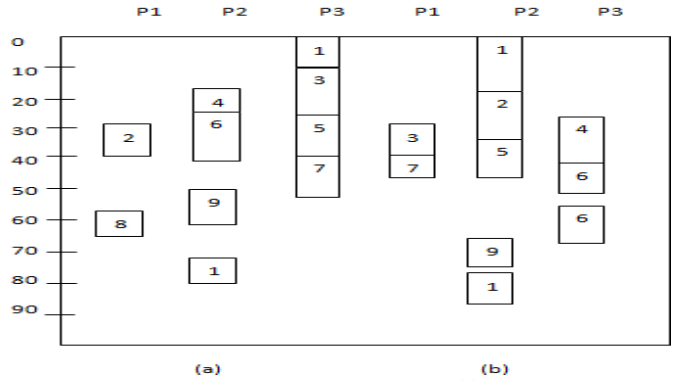
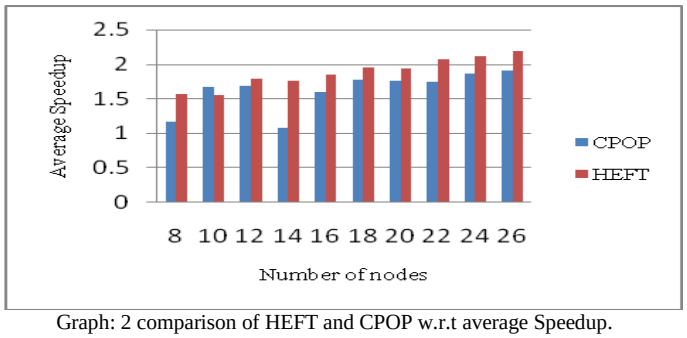
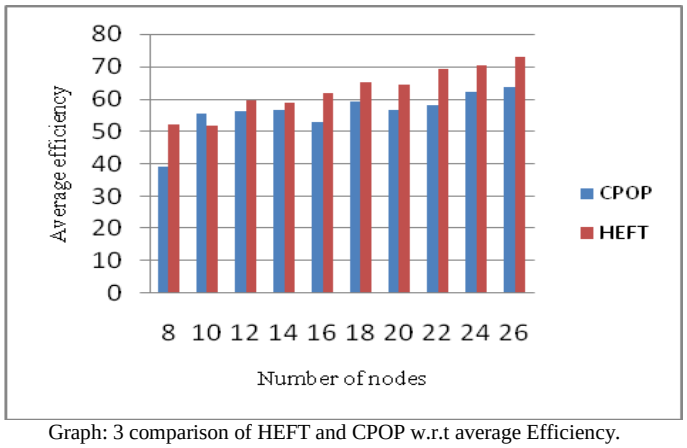


Figure: 2 represent a scheduling graph with the help of figure 1 and table 1
(a) HEFT and (b) CPOP.



Graph: 2 comparison of HEFT and CPOP w.r.t average Speedup.



Graph: 3 comparison of HEFT and CPOP w.r.t average Efficiency.

VI. CONCLUSION AND FUTURE WORK

In this paper we discussed two algorithms namely HEFT and CPOP on different parameters like Schedule Length, Speedup and Efficiency. These algorithms are scheduled on different number of DAGs in static task scheduling algorithms in heterogeneous environment. Results show us HEFT is better than CPOP for all discussed parameters. As consider increasing the number of node. It provides better results for all the parameters.. The results given in the paper demonstrate the fact that there is still a scope of improvement in many aspects for all the algorithms in the literature. Although list scheduling is a vast area of research keeping in view the findings in the given survey it is clear that there is a need of developing a technique which can produce an efficient priority list for tasks to develop an assignment based algorithm so as to reduce the overall execution time (makespan).

REFERENCES

[1] Garey, M. R. e D. S., "A Guide to the Theory of NP-Completeness", W. H. Freeman & Co, New York, NY, USA , 1979.
[2] Yang, T. and Gerasoulis, "A DSC Scheduling parallel tasks on an unbounded number of processors", IEEE Transaction Parallel Distributed System, 5(9), pp:951–967, 1994.
[3] Ilavarasan, E. and Thambidurai, "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments", Journal of Computer Sciences 3 (2), pp: 94- 103, 2007.
[4] Topcuoglu, H. Hariri, S. and Wu, M.Y. "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE Trans. Parallel Distributed System, 13(3), pp: 260–274, 2002.
[5] Adam, T. L. Chandy K. M. and Dickson J. R. "A comparison of list schedules for parallel processing systems", Communication ACM, 17(12), pp: 685–690, 1974.
[6] Kwok, Y.K. and Ahmad, I., "Benchmarking and comparison of the task graph scheduling algorithms", J. Parallel Distrib. Comput, 59(3), pp: 381–422, 1999.
[7] Liu, G. Q. Poh, K. L. and Xie, M. "Iterative list scheduling for heterogeneous computing", J. Parallel Distrib. Comput, 65(5), pp: 654–665, 2005.
[8] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee, "Scheduling precedence graphs in systems with interprocessor communication times," SIAM Journal on Computing, vol. 18, no. 2, pp. 244–257, 1989.
[9] M. Y. Wu and D. D. Gajski, "Hypertool a programming aid for message-passing systems," IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 3, pp. 330–343, 1990.
[10] G. C. Sih and E. A. Lee, "Compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures," IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 2, pp. 175–187, 1993.
[11] H. El-Rewini and T. G. Lewis, "Scheduling parallel program tasks onto arbitrary target machines,"Journal of Parallel and Distributed Computing, vol. 9, no. 2, pp. 138–153, 1990.
[12] M. Iverson, F. Özgüner, and G. Follen, "Parallelizing existing applications in a distributed heterogeneous environment," in Proceedings of the IEEE International Conference on Heterogeneous Computing Workshop (HCW '95), pp. 93–100, 1995.
[13] E. Ilavarasan, P. Thambidurai, and R. Mahilmannan, "High performance task scheduling algorithm for heterogeneous computing system", Volume 3719 of Lecture Notes in Computer Science, pp.193-203. Springer, 2005.
[14] Luiz F. Bittencourt, Rizos Sakellariou and Edmundo R. M. Madeira," DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm", 18th Euromicro Conference on Parallel Distributed and Network-based Processing, pp. 27-34, 2010.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 05, May-2016

[15] Savina Bansal, Padam Kumar and Kuldip Singh, "Dealing with heterogeneity through limited duplication for scheduling precedence constrained task graphs", J. Parallel Distrib. Comput, vol. 65, pp. 479 – 491, 2005).
[16] Tomasz Kalinowski , Iskander Kort and Denis Trystram ,"List scheduling of general task graphs under LogP" ,Parallel Computing, vol.26, pp. 1109-1128, 2000.
[17] R. Eswari and S. Nickolas, "A Level-wise Priority Based Task Scheduling for Heterogeneous Systems" International Journal of Information and Education Technology, Vol. 1, No. 5, pp. 371-375, December 2011.
[18] Mohammad I. Daoud and Nawwaf Kharma, "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems" J. Parallel Distrib. Comput, 68, pp. 399 – 409, 2008.
[19] Samia Ijaz, Ehsan Ullah Munir, Waqas Anwar, and Wasif Nasir, "Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment" The International Arab Journal of Information Technology, Vol. 10, No. 5, pp. 486-492, September 2013.
[20] Doruk Bozdag and Fusun Ozguner, "Comparison of Schedules and a Two-Stage Approach for Duplication-Based DAG Scheduling" IEEE Transactions on Parallel and Distributed System, VOL. 20 NO.6, pp: 857-871, June 2009.