Path-based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems

R.Eswari
Department of Computer Applications
National Institute of Technology
Tiruchirappalli
eswari@nitt.edu

S.Nickolas
Department of Computer Applications
National Institute of Technology
Tiruchirappalli
nickolas@nitt.edu

***Abstract*—A vital role of static task scheduling is to obtain high performance in distributed computing environment. Several algorithms have been proposed for homogeneous and heterogeneous distributed computing systems. In this paper we propose a static list Path based Heuristic Task Scheduling (PHTS) algorithm to efficiently schedule tasks on the heterogeneous distributed computing systems. The algorithm is mainly focused on reducing the makespan. It consists of three phases: A path prioritizing phase, selecting all possible paths from the given graph and sorting them by descending order. Secondly, a task selection phase, selecting the tasks from the sorted paths and finally, a processor selection phase, assigning the tasks to processors which minimizes the completion time. From the theoretical analysis of the PHTS algorithm with HEFT for a DAG graph, the better schedule length is observed.**

***Keywords*—static task scheduling, heterogeneous distributed computing systems, Directed Acylic Graph, heuristic algorithm.**

I. INTRODUCTION

A heterogeneous distributed computing system is composed of high performance machines interconnected by a high speed network by promising high-speed processing of computationally intensive applications with diverse computing speeds. The efficient scheduling of application tasks is critical to achieve high performance in parallel and distributed systems. The objective function of scheduling is to map the tasks onto the processors and order their execution so that task precedence requirements are satisfied and minimum schedule length (or makespan) is obtained [1].

Task scheduling algorithms are broadly classified into two classes: static and dynamic. When the characteristics of an application, such as execution times of tasks and data dependencies between tasks are known in advance, the scheduling algorithm is known as static model. Static task scheduling takes place during compile time before running the distributed application. Whereas in the dynamic scheduling decisions are made at run time [2]. An application in static task scheduling is generally represented by a directed acyclic graph (DAG) in which nodes represent application tasks and edges represent intertask data dependencies. The static task scheduling for a heterogeneous distributed computing system is NP-complete problem [1, 3], which means that there is no known algorithm that finds the optimal solution in polynomial time. Various heuristic algorithms have been proposed for homogeneous and heterogeneous systems for finding sub-optimal solutions. These heuristics are categorized into several groups, such as list-based algorithms [1, 2, 7, 8, 9, 12], clustering algorithms [6, 10, 11], and duplication-based algorithms [15, 14]. Among these algorithms, the list-based scheduling algorithms provide good quality of schedules and performance [5, 14].

In this paper we propose a path-based static DAG scheduling algorithm called PHTS for heterogeneous distributed computing environment. They are for a bounded number of processors and are based on list-scheduling heuristics. The motivation behind this algorithm is to generate a better task schedule such that minimum schedule length is achieved.

The remainder of this paper is organized as follows. Section II discusses on problem definition. Section III gives an overview of the related work. Section IV presents our proposed Path-based Heuristic Task Scheduling (PHTS) algorithm with examples. Section V discusses the theoretical analysis and Section VI concludes with scope for future work.

II. PROBLEM DEFINITION

In heterogeneous distributed computing systems, a distributed application is decomposed into multiple tasks with data dependencies among them. It can be represented by a directed acyclic graph (DAG), G (T, E), where T is the set of *n* tasks and E is the set of *e* edges between the tasks. Each task $t_i \in T$ represents a task in the distributed application, and each edge $(t_i, t_j) \in E$ represents a precedence constraint, such that the execution t_j starts after the execution of t_i. A task without any parent is called an entry task (t_{entry}), and a task without any child is called an exit task (t_{exit}). Each edge $(t_i, t_j) \in E$ has a value that represents the communication overhead when data is transferred from task t_i to task t_j. A task can start execution on a processor only when all data from its parents become available to that processor.

A heterogeneous distributed computing environment consists of a set Q of *m* processors connected in a fully connected topology in which all inter-processor communications are assumed to be performed without contention. It is also assumed that computation can be overlapped with communication and task execution of a given application is non-preemptive (the execution of a task

978-0-7695-4201-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ARTCom.2010.19

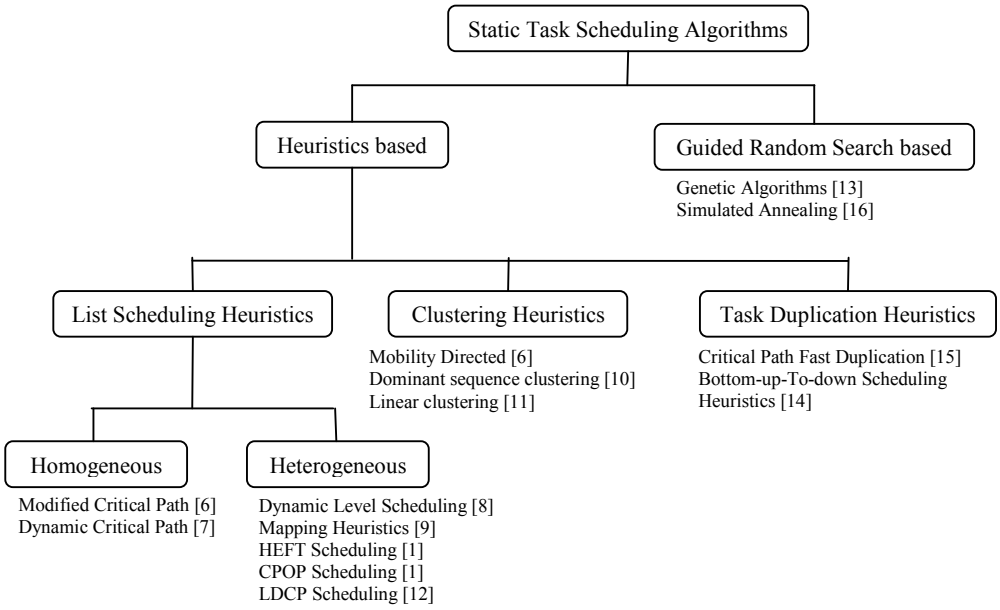

Figure 1. Classification of Static Task Scheduling Algorithms

cannot be interrupted once it starts). The *n x m* computation cost matrix C stores the execution costs of tasks. The average computation cost [1] of a task t_i is computed by

$$\overline{w_i} = \sum_{j=1}^{m} w_{i,j} \; / \; m \qquad (1)$$

where $w_{i,j}$ is the estimated execution time to complete task t_i on processor m_j. The average communication cost of an edge (i, k) is defined by

$$\overline{c_{i,k}} = \bar{L} + \frac{data_{i,k}}{\bar{B}} \qquad (2)$$

where $\bar{L}$ is the average communication startup costs of processor, $\bar{B}$ is the average data transfer rate between the processors and $data_{i,k}$ is the amount of data required to be transmitted from task t_i to t_k. If t_i and t_k are on the same processor then $c_{i,k}$= 0 since intraprocessor communication is negligible. EST(t_i, m_j) and EFT(t_i, m_j) [1] are the earliest execution start time and earliest execution finish time of task t_i on processor m_j.

$$EST\left(t_{entry}, m_j\right) = 0 \qquad (3)$$

$$EST\left(t_i, m_j\right) = \max\left(avail[j], \max_{t_k \in pred(t_i)} \left(AFT(t_k) + c_{k,j}\right)\right) \qquad (4)$$

where $AFT(t_k)$ is the actual finish time of a task t_k on the processor m_j, avail[j] is the time that the processor m_j is free and it is ready to execute task t_i. The inner max block in equation (4) returns the ready time, i.e., the time when all data needed by t_i, has arrived at processor m_j. To compute EFT of a task t_i, all immediate predecessor tasks of t_i must have been scheduled.

$$EFT\left(t_i, m_j\right) = w_{i,j} + EST(t_i, m_j) \qquad (5)$$

After all tasks in a graph are scheduled, the schedule length (overall execution time) will be the AFT of the exit task t_{exit}. The schedule also called makespan [1] is defined as

$$makespan = \max\left(AFT(t_{exit})\right) \qquad (6)$$

The objective function of task scheduling problem is to assign tasks onto the available processors in such a way that minimum schedule length is obtained.

III. Related Work

This section is organized as follows: Subsection A presents the classification of static task scheduling algorithms [1, 4] and Subsection B gives the overview of the Heterogeneous-Earliest-Finish-Time (HEFT) [1] algorithm.

A. Classification of Static Task Scheduling Algorithms

Static task scheduling can be classified into two main groups namely, Heuristics-based and Guided random search based algorithms (Fig. 1). The former can be classified into three groups: List scheduling heuristics, clustering heuristics and task duplication heuristics. List scheduling heuristics maintain a list of all tasks of a given graph according to their priorities. It has two phases: the task selection phase for selecting the highest-priority ready task and the processor selection phase for selecting a suitable processor that minimizes the predefined cost criterion. Classification of list scheduling algorithms can further be classified for supporting homogeneous processors or heterogeneous processors. The processors which are having equal capability and equal communication cost with each other is called homogeneous processors while heterogeneous processors are having different capabilities and different communication costs with each other. In clustering heuristics, a set of tasks that communicate among themselves are grouped together to create a cluster. Clusters are mapped to the available processors and task ordering within each processor is determined. Duplication based scheduling algorithms schedule a task graph by executing the tasks redundantly that have high number of dependent tasks, which reduces the interprocess communication overhead [15, 14]. Guided random scheduling algorithms make use of the principles of evolution and natural genetics to evolve near-optimal task schedules.

```
Set the computation cost of tasks and communication cost of edges
Find all possible paths from entry task to exit task by traversing the graph and sequence the
tasks for each path
Compute rank(p) for all paths
        rank(p) = sum of average computation costs of each task on the path +
                  communication costs along the path
Sort the paths in the path list by decreasing order of rank(p) values
Repeat
   For each path p_j in the sorted path list
      For each unscheduled task in the path p_j
         Select the task t_i from the path p_j
         If the selected task has no parents or scheduled parents then
              For each processor m_k in the processor set (m_k ϵ Q) do
                   Compute EFT(t_i, m_k) value using the insertion based scheduling policy
              Assign task t_i to the processor m_k that minimizes EFT of task t_i
         Else
              Select the next path in the sorted path list
      End for
   End for
Until all tasks are scheduled
```

Figure 2. Path-based Heuristic Task Scheduling (PHTS) Algorithm

B. *The Heterogeneous-Earliest-Finish-Time (HEFT) Algorithm*

HEFT algorithm has two phases: The task prioritizing phase assigns value to each task called upward rank, $rank_u$, which is based on mean computation and mean communication costs. The task list is then generated by sorting the tasks in decreasing order of $rank_u$. In processor selection phase the unscheduled task which has the highest upward rank is selected and assigned to the processor that minimizes its finish execution time, using the insertion-based scheduling policy.

IV. PROPOSED METHODOLOGY

Path-based Heuristic Task Scheduling (PHTS) algorithm, proposed for a bounded number of heterogeneous processors consists of three phases namely, a path prioritizing phase for computing the priorities for all possible paths, a task selection phase for selecting the tasks from the paths and finally a processor selection phase where the selected tasks are scheduled. The proposed algorithm is given in Fig. 2. The steps involved in the three phases of the algorithm are explained below:

A. *Path Prioritizing Phase*

The path prioritizing phase identifies set of all possible paths from a given graph by traversing from entry task to exit task. Each path is assigned by a value called rank(p), which is based on the average computation cost of each task on the path and the communication costs along the path.

$$rank(p_j) = \sum_{t_i \epsilon p_j} \overline{w_i} + \overline{c_{i,succ(t_i)}} \qquad (7)$$

Where $\overline{w_i}$ is the average computation cost of task t_i, $\overline{c_{i,succ(t_i)}}$ is the communication cost of edge from task t_i to its successor, if exists.

The stepwise trace of finding path priority for the sample DAG (Fig. 3) is shown in Table II. Path p_1 has the task sequence of $t_0 - t_3 - t_8 - t_{10}$. The rank(p_1) is computed by using the equation (7) as rank(p_1) = 5 + 5 + 16.25 + 5 + 15 + 11 + 11.25 = 68.5. Likewise, the rank value for all paths in the graph will be calculated. A path list is then generated by sorting all paths by decreasing order of rank(p) values. Ties are decided on a random basis.

B. *Task Selection Phase*

In task selection phase, the algorithm selects the unscheduled tasks from the paths in the sorted path list. During the task selection, the algorithm applies the following condition on each task:

- The task should not be scheduled earlier.
- The task has no parents or its parents are scheduled already.

Any task that satisfies the above mentioned condition is selected for scheduling. The task selection starts from the path which has the highest priority value. The condition is applied to each task of this path sequentially. Any task which satisfies the specified condition will be selected immediately, if not, then the selection process moves on to the next prioritized path starting from the first task. The selection is continued till all tasks are scheduled. For example, the stepwise trace of task selection for the sample DAG is given in Table III. The selection process starts form the path p_1 which has the highest priority among all paths. From this path, task t_0 is selected first since it has no parents. Task t_3 is selected next since its parent t_0 is scheduled already. Next task t_8 is selected since its parent t_3 is scheduled and task t_{10} cannot be selected next since its parents (t_2 and t_6) are not scheduled earlier. So the selection process moves on to the next prioritized path called p_2. It selects the unscheduled tasks sequentially from p_2. If no more tasks found, then the selection process moves on to the next prioritized path and so on. This process will be continued until all tasks from the sorted path list are scheduled. Thus the task sequence selected by PHTS algorithm for the given DAG is $t_0 - t_3 - t_8 - t_1 - t_4 - t_6 - t_2 - t_7 - t_{10} - t_5 - t_9$.

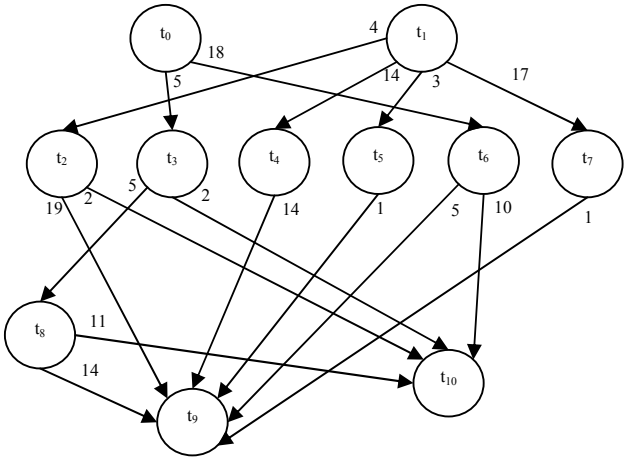

Figure 3. Sample DAG

TABLE I. COMPUTATION COST MATRIX

Task	M_0	M_1
t_0	4	6
t_1	15	22.5
t_2	4	6
t_3	13	19.5
t_4	10	15
t_5	7	10.5
t_6	8	12
t_7	4	6
t_8	12	18
t_9	6	9
t_{10}	9	13.5

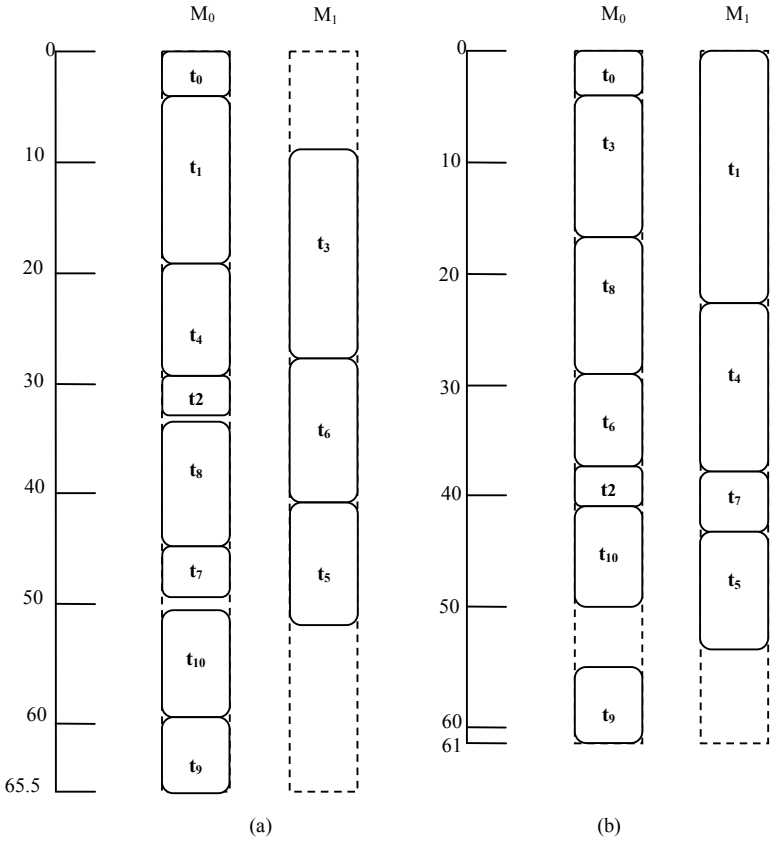

Figure 4. The schedules generated by the (a) HEFT algorithm (b) PHTS algorithm

C. Processor Selection Phase

In this phase, the selected task is assigned to a processor in the set of processors that minimizes its finish execution time using the insertion-based scheduling policy [1]. When a processor m_j is assigned a task t_i, the insertion based scheduling policy considers the possible insertion of a task in an earliest idle time slot between two already-scheduled tasks on a processor. This must be done without violating the precedence constraints among tasks. An idle time slot on processor m_j is defined as the difference between execution start time and finish time of two tasks that were consecutively scheduled on the processor m_j which should be capable of computation cost of the task to be scheduled. The search starts from a time equal to the ready time of t_i on m_j, and proceeds until it finds the first idle time slot with a sufficient large time space to accommodate the computation cost of t_i on m_j. If no such idle time slot is found, the insertion-based scheduling policy inserts the selected task after the last scheduled task on m_j. If two processors are

producing same EFT for a selected task, then the following selection strategies can be followed:

- Select processor randomly
- Select processor that is lightly loaded

TABLE II. PATH PRIORITY COMPUTATION OF SAMPLE DAG

p_i	Path	rank(p_i)
p_1	$t_0 - t_3 - t_8 - t_{10}$	68.5
p_2	$t_0 - t_3 - t_8 - t_9$	67.75
p_3	$t_1 - t_4 - t_9$	66.75
p_4	$t_0 - t_6 - t_{10}$	54.25
p_5	$t_1 - t_2 - t_9$	54.25
p_6	$t_1 - t_7 - t_9$	49.25
p_7	$t_0 - t_6 - t_9$	45.5
p_8	$t_1 - t_2 - t_{10}$	41
p_9	$t_0 - t_3 - t_{10}$	39.5
p_{10}	$t_1 - t_5 - t_9$	38.5

The stepwise trace of processor selection for each task of DAG (Fig. 3) using PHTS algorithm is given in Table III.

TABLE III. TASK SELECTION AND PROCESSOR SELECTION OF SAMPLE DAG

Step	Selected Task	Selected Path	Selected Processor
1	t_0	p_1	M_0
2	t_3	p_1	M_0
3	t_8	p_1	M_0
4	t_1	p_3	M_1
5	t_4	p_3	M_1
6	t_6	p_4	M_0
7	t_2	p_5	M_0
8	t_7	p_6	M_1
9	t_{10}	p_8	M_0
10	t_5	p_{10}	M_1
11	t_9	p_{10}	M_0

In equation (3) the earliest starting time of the first task on any processor is 0. From Table I compared to processor M_1, processor M_0 has less computation time. So task t_0 is scheduled on M_0. For the next task t_3, the earliest finish time (EFT) is calculated by equation (5). The task t_3 is assigned to processor M_0 which has minimum EFT value. Likewise, the selected task sequence is assigned to the available processors.

V. RESULTS AND DISCUSSIONS

Effective task scheduling is important to achieve minimum schedule length in heterogeneous distributed computing system for high performance. In this paper a new PHTS algorithm is proposed and the effectiveness of the algorithm is shown with a sample DAG. Fig. 3 shows the sample DAG and its computation cost matrix is given in Table I. The stepwise trace of PHTS algorithm for the sample DAG is given in Table II & III. The schedules generated by HEFT algorithm and the PHTS algorithm for sample DAG is shown in the Fig. 4 (a), (b). The schedule length generated by the proposed algorithm for the sample graph is 61, whereas the schedule length generated by HEFT algorithm for the same graph is 65.5. Thus the PHTS algorithm schedule tasks with the reduction in makespan.

VI. CONCLUSION

In this paper, a new Path-based Heuristic Task Scheduling (PHTS) algorithm for heterogeneous distributed computing systems has been proposed. The performance of the algorithm is compared with the existing HEFT algorithm. The proposed algorithm is evaluated for different DAGs and found to be giving better schedule length. It can be further strengthened with more real world applications as future work and can be implemented in a simulated environment.

REFERENCES

[1] H.Topcuoglu, S. Hariri, and M.Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel and Distributed Systems, Vol. 13, No.3, pp. 260-274, March 2002.

[2] Y.K. Kwok and I. Ahmad, "Static Scheduling Algorithms for allocating Directed Task Graphs to Multiprocessors", ACM Computing Surveys, Vol.31, No.4, pp. 406-471, December 1999.

[3] J.-C. Liou and M.A. Palis, "A Comparison of General Approaches to Multiprocessor Scheduling", In Proc. International Parallel Processing Symposium, pp. 152-156, 1997.

[4] Liang-Teh Lee, Hung-Yuan Chang, Kang-Yuan Liu, Gei-Ming Chang, and Chin-Chih Lien, "A Dynamic Scheduling Algorithm in Heterogeneous Computing Environments," IEEE W4B-4, ISCIT, pp. 313-318, 2006.

[5] Chih-Hsueh Yang, PeiZong Lee, and Yeh-Ching Chung, "Improving Static Task Scheduling in Heterogeneous and Homogeneous Computing Systems," International Conference on Parallel Processing, pp. 45, 2007.

[6] M. Wu and D. Gajski, "Hypertool: A Programming Aid for Message Passing Systems," IEEE Trans. Parallel and Distributed systems, Vol.1, pp.330-343, July 1990.

[7] Y. Kwok and I. Ahmad, "Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors," IEEE Trans. Parallel and Distributed Systems, Vol.7, no. 5, pp. 506-521, May 1996.

[8] G. C. Sih and E. A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," IEEE Trans. Parallel and Distributed Systems, pp. 175-186, Feb. 1993.

[9] H. EI-Rewini and T. G. Lewis, "Scheduling Parallel Program tasks onto Arbitrary Target Machines," J. Parallel and Distributed Computing, pp. 138-153, Sept. 1990.

[10] T. Yang and A. Gerasoulis, "DSC: Scheduling Parallel Tasks on an Unboounded Number of Processors," IEEE Trans. Parallel and Distributed Systems, Vol.5, no. 9, pp. 951-967, Sept. 1994.

[11] S.J. Kim and J. C. Browne, "A General Approach to Mapping of Parallel Computation upon Multiprocessor Architectures," Proc. International Conf. Parallel Processing, Vol.2, pp. 1-8, 1988.

[12] Mohammad I. Daoud, and Nawwaf Kharma, "Efficient Compile-Time Task Scheduling for Heterogeneous Distributed Computing Systems", Proc. International Conf. Parallel and Distributed Systems, 2006.

[13] L. Wang, H.J. Siegel, and V.P.Roychowdhury, " A Genetic Algorithm-Based Approach for Task Matching and Scheduling in Heterogeneous Computing Environments," Proc. Heterogeneous Computing Workshop, 1996.

[14] Y.-C. Chung and S. Ranka, "Applications and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed memory multiprocessors," In Supercomputing , pp. 512-521, 1992.

[15] I. Ahmad, and Y. Kwok, " A New Approach to Scheduling Parallel Programs Using Task Duplication," Proc. Internaltional Conf. Parallel Processing, Vol.2, pp. 47-51, 1994.

[16] P. Shroff, D.W. Watson, N.S. Flann, and R. Freund, "Genetic Simulated Annealing for Scheduling Data-Dependent Tasks in Heterogeneous Environments," Proc. Heterogeneous Computing Workshop, pp. 98-104, 1996.