2011 International Conference on Information Communication and Management
IPCSIT vol.16 (2011) © (2011) IACSIT Press, Singapore

Expected Completion Time based Scheduling Algorithm for Heterogeneous Processors

R.Eswari [1+] and S.Nickolas [2]

[1] Assistant Professor, Department of Computer Applications, National Institute of Technology, Trichy, Tamilnadu, India

[2] Associate Professor, Department of Computer Applications, National Institute of Technology, Trichy, Tamilnadu, India

Abstract. Effective scheduling of a distributed application is one of the major issues in distributed computing systems since scheduling algorithms are playing important role in achieving better performance. In this paper we proposed a static Expected Completion Time based Scheduling (ECTS) algorithm to effectively schedule application tasks onto the heterogeneous processors. The algorithm is mainly focused on minimizing the application execution time. It consists of two phases: First, the order of execution of tasks is computed in the task prioritization phase and secondly, the ordered tasks are assigned to the available processors in the processor selection phase. In comparison with the existing HEFT and PETS algorithms, the proposed ECTS algorithm offers better schedule length.

Keywords: static task scheduling, heterogeneous distributed computing systems, Directed Acylic Graph, heuristic algorithm

1. Introduction

A heterogeneous distributed computing system (HDCS) 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 is obtained. Task scheduling can be performed at compile-time or run time. When the characteristics of an application, such as execution times of tasks and data dependencies between tasks are known in advance, it is represented with a static model. The static task scheduling for a HDCS is NP-complete problem [1, 3]. Various heuristics algorithms have been proposed for homogeneous and heterogeneous systems, and are categorized into several groups, such as list-based algorithms [1, 2, 3], clustering algorithms [6], duplication-based algorithms [4] and genetic algorithms [5]. Among these algorithms, the list-based scheduling algorithms provide good quality of schedules and performance [1, 2].

In this paper we propose a new algorithm which is based on the expected completion time of task for heterogeneous distributed 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.

2. Problem Description

[+] Corresponding author. Tel.: +91-431-2503744; fax: +91-431-2500133.
E-mail address: eswari@nitt.edu

In a distributed environment, an 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 environment consists of a set *Q* of *m* processors connected in a fully connected topology. The following assumptions are made:

1. All inter-processor communications are performed without contention.
2. Computation can be overlapped with communication.
3. Task execution of a given application is non-preemptive.

The average communication cost of sending $\mu_{i,j}$ amount of data from task t_i to task t_j is defined by

$$C_{i,j} = \bar{S} + \bar{R}\, X\, \mu_{i,j} \tag{1}$$

where $\bar{S}$ is the average communication startup costs over all processors, $\bar{R}$ is the average communication cost per transferred unit over all processors. If t_i and t_j are on the same processor then $c_{i,j} = 0$ 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.

$$EST\left(t_{entry}, m_j\right) = 0 \tag{2}$$

$$EST\left(t_i, m_j\right) = \max(avail[j], \max_{t_k \in pred(t_i)}(AFT(t_k) + c_{k,j})) \tag{3}$$

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 is ready to execute task t_i. The inner max block in equation (3) returns the time when all data needed by t_i, has arrived at processor m_j.

$$EFT\left(t_i, m_j\right) = w_{i,j} + EST(t_i, m_j) \tag{4}$$

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 length, also called makespan, is defined as

$$makespan = \max\left(AFT(t_{exit})\right) \tag{5}$$

3. Related Work

This section presents the two existing static task scheduling algorithms for heterogeneous distributed environment.

3.1. The Heterogeneous-Earliest-Finish-Time Algorithm (HEFT)

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 and sorts the tasks in decreasing order of $rank_u$. The processor selection phase assigns the sorted tasks to the processors that minimize their finish time.

3.2. The Performance-Effective-Task-Scheduling Algorithm (PETS)

This algorithm starts with level sorting phase in which tasks at each level are sorted in order to group the tasks that are independent of each other. The priority of each task is computed using various attributes such as Average Computation Cost, Data Transfer Cost and Rank of Predecessor Task. The tasks are selected from each level based on their priority and assigned to the processor which gives minimum finish time.

4. Proposed Algorithm

The proposed Expected Completion Time based Scheduling (ECTS) algorithm comprises of two phases namely, Task Prioritization phase and Processor Selection phase. The algorithm is shown in Fig.2, 3, 4 and 5. The Task Prioritization phase identifies the task sequence by two stages such as level wise task priority stage and task selection stage. In level wise task priority stage, the priority of all tasks at each level is computed by

their Expected Completion Time (ECT). This ECT is based on the tasks Average Computation Cost (Definition 1) and Maximum Data Arrival Cost (Definition 2).

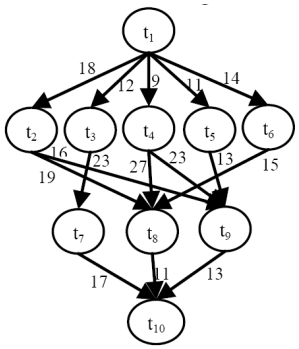

Fig. 1: Sample DAG

Tab. I. Computation cost matrix

Task	M_0	M_1	M_2
t_1	14	16	9
t_2	13	19	18
t_3	11	13	19
t_4	13	8	17
t_5	12	13	10
t_6	13	16	9
t_7	7	15	11
t_8	5	11	14
t_9	18	12	20
t_{10}	21	7	10

Definition 1. Given a DAG with n tasks and m processors, the Average Computation Cost (ACC) of a task (t_i) is computed by dividing the sum of computation cost of the task on each processor by the number of available processors.

$$ACC(t_i) = \overline{w_i} = \sum_{j=1}^{m} w_{i,j} \ / \ m \tag{6}$$

where $w_{i,j}$ is the estimated execution time to complete task t_i on processor m_j.

ECTS_Algorithm()
Read the DAG with the corresponding attributes such as computation cost, communication, number of processors
Call **Level_Task_Priority()** for finding the priority of tasks in each level
Call **Task_Selection(priority_of_tasks)** to select task sequence for execution
Call **Processor_selection(selected_task_seq)** to select best processor for each level

End

Fig. 2: ECTS Algorithm

Level_Task_Priority()
For each level L_i in the DAG do
For each task t_i in the level Li do
Find Average Computation Cost
Find Maximum Data Arrival Cost
Find Expected Completion Time
End For
Sort all tasks in decreasing order of ECT
Assign priority for each task
End For
Return the priority of tasks
End

Fig. 3: Level wise task priority

Task_Selection(priority_of_tasks)

For each level Li in the DAG do
Select tasks from the level by its priority which is calculated by their Expected Completion Time
End For
Return the selected task sequence

End

Fig. 4: Task Selection

Processor_Selection(selected_task_sequence)
For each unscheduled tasks in the task sequence do
For each processor p_k in the processor set do
Compute earliest finish time of t_j on p_k using the insertion based policy
End For
Assign task t_i to the processor p_k that minimizes its finish time
End For
End

Fig. 5: Processor Selection

Definition 2. Given a DAG with n tasks and e edges, the Maximum Data Arrival Cost (MDAC) of a task (t_j) is the highest amount of time that the task needs to spend to receive data among its parents.

$$MDAC(t_j) = max_{t_i \in pred(t_j)}(C_{i,j}) \tag{7}$$

where t_i is the set of predecessors of task t_j.

Now ECT value for all tasks is computed by definition 3. Tasks in each level are sorted in non-increasing order of their ECT value. A task which is having highest ECT value is given the highest priority for selection.

Definition 3. The Expected Completion Time (ECT) of a task (t_j) is computed by summing the average computation cost of that task and maximum data arrival cost of the same task.

$$ECT(t_j) = ACC(t_j) + MDAC(t_j) \tag{8}$$

The computed value of ACC, MDAC and ECT of all tasks for the given graph is shown in Tab.II. In task selection stage, the tasks are selected from each level according to their priority and a task sequence is produced for execution. Thus the task sequence selected by the proposed algorithm for the given graph is $t_1 - t_2 - t_6 - t_3 - t_5 - t_4 - t_9 - t_8 - t_7 - t_{10}$.

Tab. II: Priority computation for the given graph

Level	Task	ACC	Parent tasks	MDAC	ECT	Priority
1	1	13	0	0	13	1
2	2	16.667	1	18	34.667	1
2	3	14.333	1	12	26.333	3
2	4	12.667	1	9	21.667	5
2	5	11.667	1	11	22.667	4
2	6	12.667	1	14	26.667	2
3	7	11	3	23	34	3
3	8	10	2,4,6	27	37	2
3	9	16.667	2,4,5	23	39.667	1
4	10	14.667	7,8,9	17	31.667	1

Tab. III: Task and Processor selection

Selected Task	Selected Processor
t_1	M_3
t_2	M_3
t_6	M_3
t_3	M_1
t_5	M_2
t_4	M_2
t_9	M_2
t_8	M_2
t_7	M_1
t_{10}	M_2

In Processor selection phase, the selected task sequence is assigned to the processors that minimize their execution time using the insertion-based scheduling policy [1] without violating the precedence constraints among tasks. If two processors are producing same EFT for a selected task, then one of the following selection strategies can be followed:

- Select processor randomly
- Select processor that is lightly loaded
- Select processor based on minimum processor utilization.

The stepwise trace of processor selection for each task of the given graph using the proposed algorithm is shown in Tab.III.

5. Experimental Results and Discussions

In this section, the comparative evaluation of our proposed algorithm with the existing PETS and HEFT algorithms is presented. For this purpose, a sample DAG is taken to show the effectiveness of the proposed algorithm theoretically and a real world application graphs for experimental analysis.

5.1. Comparison Metrics

The following metrics have been taken to evaluate the proposed algorithm.

- Schedule Length Ratio (SLR) is the ratio of the parallel time to the sum of weights of the critical path tasks on the fastest processor.
- The Speedup is the ratio of the sequential execution time to the parallel execution time.

5.2. Theoretical Analysis

A sample DAG is shown in Fig.1 and its computation cost matrix is given in Tab.I. The schedule length generated by the ECTS algorithm for the given DAG is shown in Fig.6 and is compared with the existing PETS and HEFT algorithms, which is shown in Fig.7. The schedule length of ECTS algorithm is 73 and is shorter than the schedule length generated by PETS and HEFT algorithms which are 77 and 80 respectively. It is observed that the proposed ECTS algorithm is producing better schedule length compared to the other two algorithms.

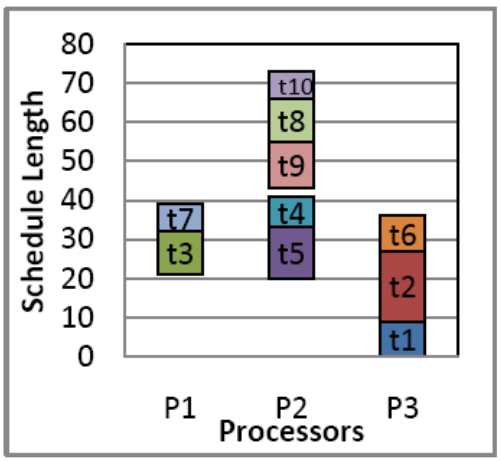

Fig. 6: Performance of ECTS Algorithm

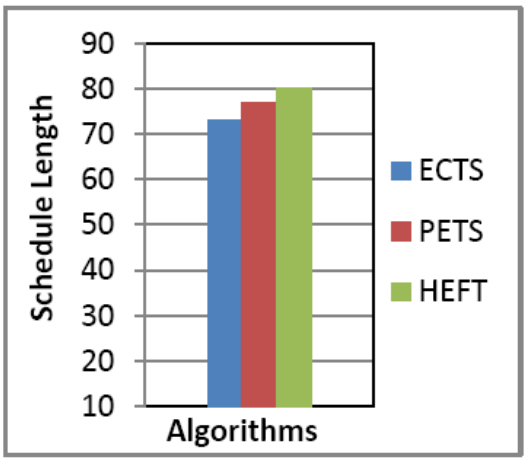

Fig. 7: Comparison of ECTS with PETS and HEFT

5.3. Fast Fourier Transformation

The task graph of the Fast Fourier Transformation (FFT) [1] is characterized by the size of the input vector. For an input vector of size M, the total number of nodes in the task graph is equal to$(2\ x\ M - 1) + (M\ x\ \log_2 M)$. Communication to computation cost ratio (CCR) [1] is the ratio of the average communication cost to the average computation cost and the computation cost of each task t_i on each processor m_j is randomly set from the range given in equation (9). $\overline{w_i}$ is selected randomly from a uniform distribution with range$[0, 2\ X\ \overline{w_{DAG}}]$, where $\overline{w_{DAG}}$ is the average computation cost of the given graph which is set randomly in the algorithm and the heterogeneity factor β is taken from the data set given in [1].

$$\overline{w_i}\ X\ \left(1 - \frac{\beta}{2}\right) \leq w_{i,j} \leq \overline{w_i}\ X\ \left(1 + \frac{\beta}{2}\right) \quad (9)$$

5.4. Performance Results

The algorithm was implemented using C Language in Intel core i7-620M processor. The performance of the algorithm is compared with respect to data points and number of processors. Fig.8 and Fig.9 present the average SLR values for FFT graphs at various sizes of input points and different number of processors, respectively.

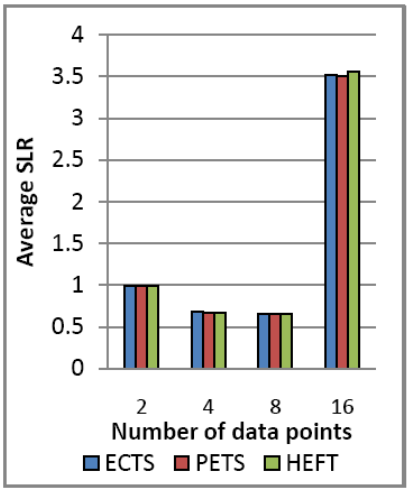

Fig. 8: Average SLR for data points

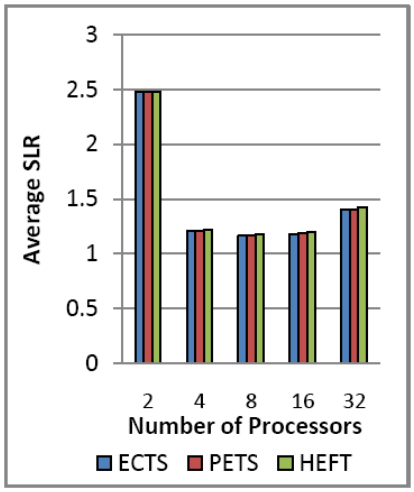

Fig. 9: Average SLR for processors

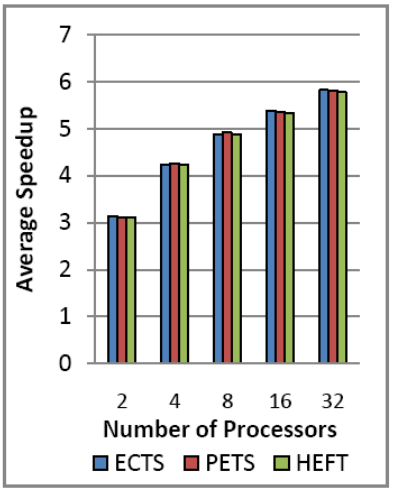

Fig. 10: Average Speedup

The average speedup obtained for each of the algorithms with respect to various numbers of processors with graphs of 16 data points is shown in Fig.10. From the results obtained and the graphs shown in Fig.8, Fig.9 and Fig.10, it can be observed that the proposed ECTS algorithm gives better performance when compared with HEFT algorithm and gives competitive performance when compared with PETS algorithm. Moreover, ECTS algorithm gives better performance when the number of tasks and processors were increased.

6. Conclusion

In this paper, an Expected Completion Time based Scheduling (ECTS) algorithm for heterogeneous environment has been proposed. It is a new method of finding minimum schedule length for static task scheduling problem. The proposed algorithm is evaluated for the real world FFT application graphs. The performance of the algorithm is compared with the existing HEFT and PETS algorithms. It can be tested for more real application graphs and enhancement can be made in terms of time complexity and efficiency which are to be considered for our future course of research work. Also the effect of task duplication in reducing schedule length can be considered in future.

7. 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. 2002, 13 (3): 260-274.

[2] E. Illavarasan and P.Thambidurai. *Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments*. J. Computer Science. 2007, 3(2): 94-103.

[3] Mohammad I. Daoud, and Nawwaf Kharma. *A high performance algorithm for static task scheduling in heterogeneous distributed computing systems*. J. Parallel Distributed Computing. 2008, 68: 399-409.

[4] S. Ranaweera and D. P. Agrawal. *A task duplication based scheduling algorithm for heterogeneous systems*. Proc. of Intl. Parallel and Distributed Processing Symposium. 2000, pp. 445–450.

[5] Kim, S.C. and S. Lee. *Push-pull: Guided search DAG scheduling for heterogeneous clusters*. Intl. Conf. on Parallel Processing. 2005.

[6] Cristina Boeres, Jos'e Viterbo Filho and Vinod E.F.Rebello. *A cluster based strategy for scheduling task on heterogeneous processors*. Proc. of 16th Symp. on Computer Architecture and High Performance Computing (SBAC-PAD). 2004.