World Applied Sciences Journal 13 (8): 1781-1786, 2011
ISSN 1818-4952

Performance Analysis of Competing Tasks and Processors in Heterogeneous Earliest Finish Time Algorithm

Altaf Hussain, Ehsan Ullah Munir,
Muhammad Wasif Nisar and Muhammad Waqas Anwar

Department of Computer Science,
COMSATS Institute of Information Technology, Wah Cantt. 47040, Pakistan

Abstract: Heterogeneous Earliest Finish Time (HEFT) algorithm outperforms many existing list scheduling algorithms for precedence constraint tasks allocation onto computationally diverse set of machines. However, if two or more tasks with the same rank or processors with the same minimum earliest finish time compete, known a tie-break, then various random and deterministic selection policies can bring a significant difference in the final schedule length (makespan). This paper considers a number of tie-breaking schemes for task and processor selection. The performance of the proposed schemes has been extensively studied with various simulation parameters on randomly generated as well as some real world application directed acyclic graphs (DAGs). Our experimental investigation shows that random selection based policy is not always an efficient choice. Furthermore, the implementation of some deterministic approaches can increase the overall cost or complexity of the algorithm but it may be a trade-off worth making.

Key words: Heterogeneous computing · DAG scheduling · Task priority · Schedule length

INTRODUCTION.

Heterogeneous computing system (HCS) is a computing p latform which consists of different kinds of resources interconnected with a high speed network. Due to diverse and sufficient computational resources, task scheduling mechanism in HCS becomes one of the key factors for achieving high performance. The common objective of scheduling is to map tasks onto available set of machines and order their execution so that task precedence requirements are satisfied and there is a minimum schedule length (makespan). The problem of finding the optimal schedule is shown to be NP-complete [1, 2].

An application with computationally intensive tasks can be parallel executed in the HCS. Directed acyclic graph (DAG) is a well accepted representation of a parallel application in which the nodes represent application tasks and the directed edges represent inter-task dependencies, such as task's precedence [3]. List scheduling algorithms [4-11] have shown to be the most promising heuristics for DAG scheduling. The basic idea of list scheduling is to prioritize the tasks of the given application DAG in a non increasing order arranged list. A task with a higher priority is scheduled before a task with a lower priority and ties are broken using some agreed method. Among the list scheduling algorithms *Heterogeneous Earliest Finish Time* (HEFT) algorithm [4] has shown best results both in terms of cost and quality of schedule. In HEFT, priorities are given to a set of tasks on the basis of task average communication and computation costs. The algorithm uses a recursive procedure to compute the rank of a task by traversing the graph upwards from the exit task. However if two or more tasks having same rank compete for the best available processor, known as tie break, then the algorithm uses random policy for task or processor selection.

In this paper we have considered various deterministic tie breaking mechanisms for competing tasks and processors. These strategies have been implemented separately in task prioritizing as well as processor selection phases. Our simulation results show that the random tie breaking policy for competing tasks and processors, as adopted by HEFT, does not always give an optimal solution. The rest of the paper is organized as follows: In section 2, HEFT scheduling attributes have been discussed. In section 3, we give the background and motivation of our work.

Corresponding Author: Altaf Hussain, Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantt. 47040, Pakistan. E-mail: altafhussain@comsats.edu.pk.

In section 4 we discuss the methodology, section 5 provides results and discussion. Finally, section 6 concludes the paper.

HEFT Scheduling Attributes: Before proceeding to the next section, it is necessary to discuss some basic scheduling attributes such as rank upward, rank downward, earliest startup time and earliest finish time, which were used in the HEFT scheduling algorithm.

The upward rank, $rank_u$, of the task v_n is calculated using the following equation

$$rank_u(v_n)\overline{W}_n + v_{m^2} \max_{succ(v_n)} \left\{\overline{d_{nm}} + rank_n(v_m)\right\} \quad (1)$$

where $\overline{W}_n$ is the average computation cost of the task v_n, $\overline{d_{nm}}$ is the average communication cost of the edge from task v_n to task v_m, $succ(v_n)$ is the set of all

Immediate successors of task v_n For the entry task, the upward rank is

$$rank_u(v_n) = \overline{W}_{entry} \quad (2)$$

Similarly, the downward rank of task v_n is defined by the following equation

$$rank_d(v_n) = v_{m^2} \max_{pred(v_n)} \left\{\overline{w_n} + \overline{d_{nm}} + rank_d(v_m)\right\} \quad (3)$$

where $pred(v_n)$ is the set of all immediate predecessors of the task v_n $EST(v_n,p_k)$ and $EFT(v_n,p_k)$ are the earliest startup time and earliest finish time of the task v_n on the processor p_k, respectively. For the entry task v_{entry},

$$EST(v_{entry}, p_k) = 0, \quad (4)$$

For the other tasks in the task graph, the EST and EFT values are computed recursively, starting from the entry task. EST and EFT are defined by

$$EST(v_n,p_k) = \max\left\{avail[k], v_{m^2} \max_{pred(v_n)} \left(AFT(v_m) + d_{m,n}\right)\right\} \quad (5)$$

$$EST(v_n,p_k) = w_{n,m} + EST(v_n,p_k) \quad (6)$$

where $avail[k]$ is the earliest time at which processor p_k is ready to execute the next task. In order to compute the EFT of a task v_n all the immediate predecessor tasks of v_n must have been scheduled.

Background and Motivation: Selection of an appropriate tie break mechanism in case of competing tasks or processors has been an important consideration in well known heuristics for tasks scheduling in heterogeneous computing systems. *Modified critical path* (MCP) [5] algorithm resolves a tie by looking at the latest starting time of the descendants. The *Earliest time ?rst* (ETF) [6] algorithm selects the task with higher static level and D*ynamic level schedule* (DLS) [7] algorithm takes into account the descendants dynamic level to break the ties. HEFT algorithm adopts the random tie breaking policy for competing tasks as alternative policies increase the complexity of the algorithm [12]. However, the processing power of the existing machines has increased considerably over the last decade. In this scenario, a minute increase in algorithmic lines of code can lead to only fractional increase in its complexity but can contribute an enormous decrease in the final schedule of the given task graph.

The schedule of the DAG shown in Fig. 1 explains the benefits of alternate tie breaking mechanisms in detail. The given DAG is scheduled among three processors with different computational capabilities. The communication costs among the nodes are represented as weighted directed edges while the computation costs are given in the provided table. In the task prioritizing phase, the non increasing order of upward rank of the tasks is {1, 5, 6, 2, 4, 3, 9, 8, 7, 10}. Two alternate policies have been implemented in the processor selection phase i.e. *random selection* policy as adopted by the original HEFT algorithm and *predecessor task processor* policy for suitable processor selection (that will be elaborated in the next section). The processor with the minimum earliest finish time (EFT) is selected among the three competing processors. While scheduling the tasks for the best available processors, the task 6 has the EFT values for the three processors as {78, 78, 81} respectively. The processor 1 and 2 are the competing processors for the task 6. The previous task 5 has been scheduled on processor 2. The r*andom selection* policy specifies that processor 1 or 2 can be selected arbitrarily with equal probability. Now if the processor 1 is selected (random selection policy) for scheduling task 6 then the final schedule length so obtained is 231. However *predecessor task processor* policy identifies processor 2 as the best available processor with the schedule length 178. This significant improvement in the result is only a small evidence for the overall benefit of selecting appropriate tie breaking policy. The motivation of the present work is to

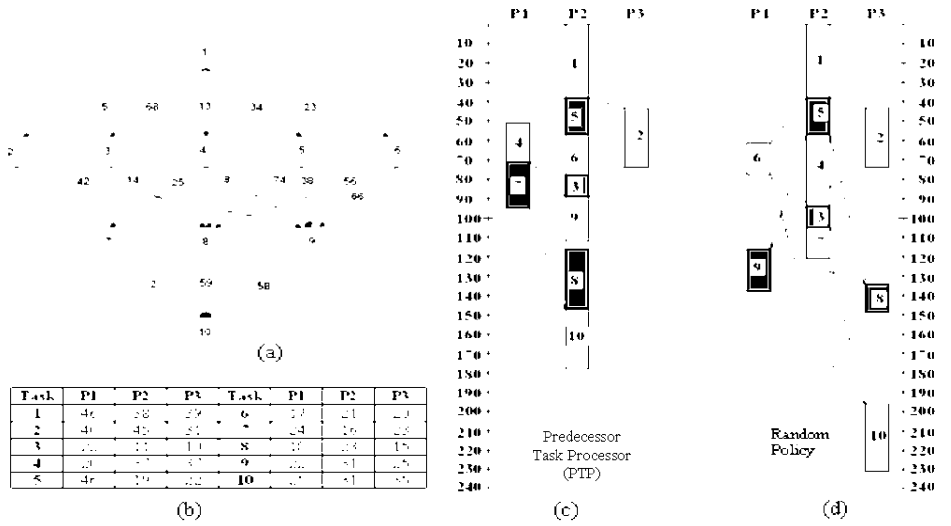


Fig. 1: (a) A sample DAG with ten tasks (b) table showing computation costs of each task (c) Predecessor task selection policy schedule (d) rendom selection policy schedule

investigate different tie breaking mechanisms and compare the effect of these approaches on the performance of the HEFT.

Methodology: HEFT consists of two major phases, task prioritization and processor selection. In task prioritization phase, the algorithm computes the rank of all the given tasks in non increasing order based on some criterion. In the processor selection phase the tasks in the sorted order are scheduled on the best available processor, which minimizes the finish time of the ranked task. We have partitioned the problem into two sections and considered different options for competing tasks and processor selection. In task prioritizing phase, we have implemented three policies i.e. *random selection (RS)* policy, *maximum immediate successor/predecessor (MISP)* policy and *first in order (FIO)* policy. In *RS*, if a tie happens among competing tasks then any of the tasks will be selected for scheduling on adhoc or arbitrary fashion. In *MISP*, a task with the higher immediate successor or predecessor rank will be selected if the tasks are listed according to upward rank or downward rank respectively. In *FIO*, the task which is coming first in the original DAG order will be selected for scheduling.

1. Set the computation and communication costs of the tasks and edges, respectively, with the mean value
2. Compute the upward or downward rank of tasks by using equations (1) & (2)
3. Sort the tasks in a priority queue by non-increasing order of rank values.
4. While ?k is not the exit task do
5. If two or more tasks have the same rank value then
6. mark the tasks as competing tasks.
7. resolve the tie among the competing tasks by some appropriate scheme.
8. re-arrange the priority queue.
9. end if
10. end while
11. Initialize the priority queue with the entry task.
12. While there are unscheduled tasks in the priority queue do
13. Select the highest priority task ?i from the queue
14. for each processor pk in the processor set (pk e Q) do
15. compute the EFT by equation (6)
16. find the minimum EFT processor pk.
17. if two or more processors have the same minimum EFT value then
18. mark the processors as competing processors.
19. resolve the tie among the competing processors by some appropriate policy.
20. end if
21. end for
22. assign the task vi to the selected processor pk.
23. end while

Algorithm 1: Tie breaking pseudo code for HEFT algorithm

Similarly, in the processor selection phase, if a task has the same minimum earliest finish time for two or more processors then three policies will be considered i.e. *random selection (RS)* policy, *predecessor task processor (PTP)* policy and *first in set (FIS)* policy. *RS* is the same as in the task prioritizing phase. In *PTP,* the processor on which the previous task is scheduled (if it happens) is selected for the current task scheduling. In *FIS*, the processor which is coming first in the processor set is selected for scheduling.

In order to investigate the performance of the above mentioned schemes, following comparison metrics were used:

Average Percentage Degradation (APD): [12, 13] this metric indicates the average degradation in the schedule of a particular scheme as compared to the best schedule so obtained by any of the other schemes.

Number of Best Solutions (NB): The number of times a particular scheme gives the minimum schedule length compared to other schemes.

Number of Best Solutions Equal with Another Scheme (NEB): The number of times a particular scheme produces the minimum schedule length but another scheme also exhibits the same performance.

Worst Percentage Degradation (WPD): The maximum degradation in the schedule length of a particular scheme from the schedule length of the best one.

RESULTS AND DISCUSSION

In order to illustrate the impact of different tie breaking chemes mentioned in section 3, we implemented a simulation of HEFT on randomly generated *Directed Acyclic Graphs (*DAGs) with parameters described by [4]. In addition to randomly generated task graphs, we also considered task graphs of real world application problem, Gaussian elimination algorithm. The experimental results are organized into two major test suits.

Test Suit 1: In this test suit, in order to analyze the performance of six schemes we generated random task graphs with two main characteristics; the number of tasks in the graph and number of available processors to schedule these tasks. These two parameters are so chosen because increase in the number of tasks or number of processors can increase the probability of the competing tasks and processors. The communication to computation ratio (CCR) was set to 1.0 and the shape parameter of the graphs (α) was also taken as 1.0 to generate standard task graphs with equal degree of parallelism [4]. A total of six different sets of experiments were performed (three schemes in task prioritizing phase + three schemes in processor selection phase). Granularity of the tasks was varied form 20 to 100 with a step of 20 to generate diverse set of DAGs. Furthermore, number of processors was varied from 4 to 32 {4, 8, 16, 32}. All these schemes were implemented separately for upward as well as downward ranks.

The *average percentage degradation* for each of the above mentioned schemes with randomly generated DAGs is shown in Figure 2 (a, b & c). The suffix '&U' or '&D' with the given schemes is the shorthand representation if the ranking is performed upward or downward, respectively. From the figure, *random selection* policy outperformed all other policies. In (a), RS&U is better than FIS&U by 4.4 %; in (b) by 11.9 % and in (c) by 12%. For downward ranking in task prioritization phase, *random selection* policy is not always the best one. In (b) & (c) RS&D is worse than FIS&D by 5.9 % and 1.3 % respectively.

Table 1 shows the results of three other metrics, NB, NEB and WPD. NB and NEB metrics show that the best single scheme is RS&U but on the other hand it also exhibit its *worst percentage degradation* of 68.4 % against the best scheme in the set.

Test Suit 2: The tie breaking schemes were implemented for the Gaussian elimination task graph with matrix size varied from 6 to 14 with a step of 2. The numbers of processors were varied from 4 to 32 {4, 8, 16, 32}. All these schemes were implemented separately for upward as well as downward ranks.

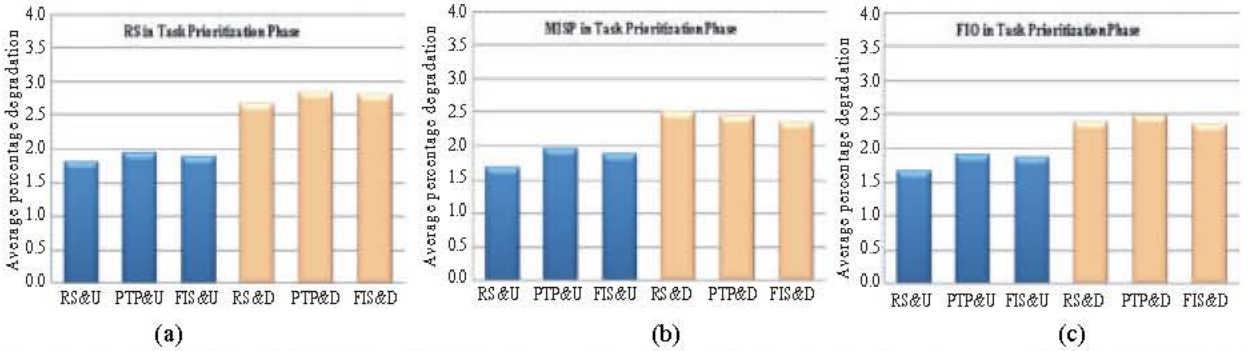
The results for APD are given in the Figure 3 (a, b & c) and for three other metrics i.e NB, NEB and WPD are presented in Table 2. As in the case of random DAGs, *predecessor task processor* and *first in se*t schemes does not exhibit large variations from the best scheme.

Table 1: NB, NEB & WPD for Random DAGs

	RS task priority			MISP task priority			FIO task priority		
Scheme	NB	NEB	WPD	NB	NEB	WPD	NB	NEB	WPD
RS&U	951	6	22.6	976	13	19.7	1003	5	30.4
PTP&U	103	799	22.5	99	802	25.7	108	775	25.1
FIS&U	143	801	27.0	111	813	25.7	118	762	23.4
RS&D	990	11	68.4	993	8	24.5	938	7	54.6
PTP&D	15	981	46.7	102	769	28.9	16	1021	40
FIS&D	9	980	46.6	128	777	27.2	14	1036	42.5

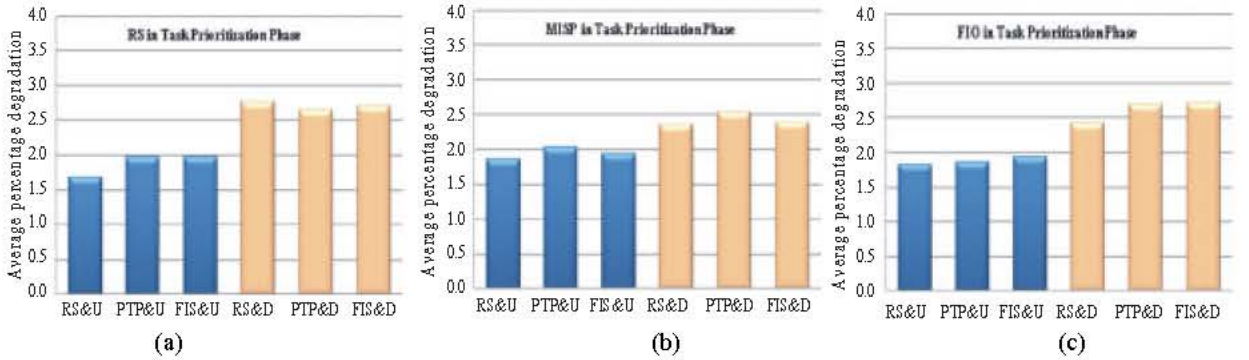
Table 2: NB, NEB & WPD for Gaussian DAGs

	RS task priority			MISP task priority			FIO task priority		
Scheme	NB	NEB	WPD	NB	NEB	WPD	NB	NEB	WPD
RS&U	998	3	19.5	937	9	27.0	959	5	18.8
PTP&U	99	812	21.9	107	819	25.3	122	812	26.9
FIS&U	96	799	21.9	128	828	21.8	99	823	26.9
RS&D	979	1	56.5	979	4	21.5	995	11	66.8
PTP&D	9	999	67.9	115	745	27.3	13	973	69.7
FIS&D	12	1000	63.7	148	767	27.3	10	980	68.2



(a) (b) (c)

Fig. 2: Random DAGs: 20-100 Tasks, 4-32 Machines, with policies in task prioritization phase as (a) RS (b) MISP (c) FIO



(a) (b) (c)

Fig. 3: Gaussian DAGs: 6-14 Matrix, 4-32 Machines, with policies in task prioritization phase as (a) RS (b) MISP (c) FIO

CONCLUSIONS

From the results presented above, it is clear that the performance of HEFT algorithm significantly depends upon various tie breaking schemes implemented in task prioritizing as well as processor selection phases. It is also evident from the experimental investigation that the adhoc or random schemes are always not a best choice. The performance of the HEFT algorithm can further be improved using suitable deterministic approaches for tie breaking if so happens. This would increase the cost or complexity of the algorithm, but can profoundly minimize the schedule length of given DAG.

REFERENCES

1. Gary, M.R. and D.S. Johnson, 1979. "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman and Company,
2. Ibarra, O.H. and C.E. Kim, 1977. "Heuristic algorithms for scheduling independent tasks on non-identical processors", J. ACM, pp: 280-289.
3. Kwok, Y.K. and I. Ahmad, 1999. "Static scheduling algorithms for allocating directed task graphs to multiprocessors", ACM Computing Surveys, pp: 406-471.

4. Topcuoglu, H., S. Hariri and M.Y. Wu, 2002. "Performance Effective and Low Complexity Task Scheduling Algorithm scheduling for heterogeneous computing", IEEE Transaction on Parallel and Distributed System, 13: 3.
5. Wu, M.Y. and D.D. Gajski, 1990. "Hypertool: A Programming Aid for Message-Passing Systems," IEEE Trans. Parallel and Distributed Systems, 1(3): 330-343.
6. Hwang, J.J., Y.C. Chow, F.D. Anger and C.Y. Lee, 1989. "Scheduling Precedence Graphs in Systems with Inter processor Communication Times", SIAM J. Computing, 18(2): 244-257.
7. Sih, G.C. and E.A. Lee, 1993. "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," IEEE Trans. Parallel and Distributed Systems, 4(2): 75-87.
8. Daoud, I. and N. Kharma, 2008. "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems", J. Parallel Distrib. Comput., pp: 399-409.
9. Liu, G.Q., K.L. Poh and M. Xie, 2005. "Iterative list scheduling for heterogeneous computing", J. Parallel Distrib. Comput. pp: 654-665.
10. Tang, X., L. Kenli and D. Padua, 2009. "Communication contention in APN list scheduling algorithm", Science in China Series F: Information Sci., pp: 59-69.
11. Tang, X., L. Kenli, L. Guiping and L. Renfa, 2010. "List scheduling with duplication for heterogeneous computing systems", J. Parallel Distrib. Comp., pp: 323-329.
12. Kwok, Y.K. and I. Ahmad, 1999. "Benchmarking and comparison of the task graph scheduling algorithms", J. Parallel Distrib. Comput., pp: 381-422.
13. Zhao, H. and R. Sakellariou, 2004. "A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems", Proc. 18th Parallel and Distributed Processing Symposium, pp: 26-30.