A New Time Scheduling Concepts On assorted compute Organization

N.V.Madhu
Associate Professor
Christu Jayanthi Jubilee college
Guntur
Venkatamadhu.neelam@gmail.com

ABSTRACT

There are two algorithms for heterogeneous computing environment. They are Heterogeneous Earliest Finish time (**HEFT**) algorithm and Alternate Heterogeneous Earliest Finish time (**AHEFT**) algorithm. The latter is an extension of the first one. Different weighted Directed Acyclic Graphs (DAG's) can be used for comparing the two algorithms by simulation. A network of fully connected heterogeneous processors is assumed for task scheduling.The performance indices that can be used for comparison of algorithms are Schedule Length, Speedup Factor and Efficiency. For higher values of CCR (Communication to Computation Ratio), AHEFT algorithm performs better than HEFT. Moreover, as the number of tasks in the DAG increase, the performance improvement of AHEFT algorithms is more as compared to HEFT. Also AHEFT algorithm can be modified to have another algorithm called EAHEFT(Extended AHEFT), where all the entry tasks of the input application DAG are scheduled on to the processors, and then the remaining tasks are handled for scheduling. This algorithm on average as the number of tasks and processors increases, produces better results compared to the HEFT and AHEFT algorithms.

1. INTRODUCTION :-

We choose HEFT because it is simple ,popular and was shown to be competent.

The HEFT algorithm has two phases as below.

i) Task Prioritizing Phase
ii) Processor Selection Phase.

i) Task Prioritizing Phase

In this phase, the priority of each task is to be set with the upward rank value ($rank_u$), which is based on mean computation cost as well as mean communication cost. The tasks are sorted by decreasing order of $rank_u$ and a ready list is generated. If tie occurs, policies such as selecting the task whose immediate successors tasks has higher upward ranks can be used. As these policies tend to increase the completion time, so a random selection strategy is normally preferred.

ii) Processor Selection Phase

In this phase, the tasks are scheduled on processors based on the ready list of the tasks. The earliest available time of a processor Pj for a task execution is the time when Pj has completed the execution of its last assigned task. However, the HEFT algorithm has an insertion-based policy, which considers the possible insertion of a task in an earliest idle time slot between two already scheduled tasks on a processor. However, to ensure non-preemptive nature of the scheduling, the length of an idle time-slot i.e, the difference between AST and AFT of two tasks, should be capable of computation cost of the task to be scheduled. In addition, scheduling on this idle time slot should preserve precedence constraints .

HEFT uses the EFT as the objective for selection of best processor. The HEFT algorithm selects the task with the highest upward rank at each step,The selected task is assigned to the processor which minimizes its EFT.

Pseudo Code in detail.

Step 1: The computation costs of the tasks and communication costs of the edges are set to their mean values.

Step 2: The DAG is traversed from bottom to top and ranku of each task is calculated recursively.

Step 3: A ready list is prepared by sorting the tasks in the decreasing order of their ranku values.

Step 4: Till there are unscheduled tasks in the ready list
Do

Step 4.1 From the ready list, select the first task Ti, for scheduling.

Step 4.2 For each processor Pj in the processor set ($Pj \in Q$), do the following

Step 4.3 Calculate the EFT (Ti , Pj) for the selected task by using the insertion-based scheduling policy.

Step 4.4 Task Ti is scheduled on the processor, which minimizes the EFT of that task.

Step 5. End While.

In the HEFT algorithm, the search of an appropriate idle time slot of a task Ti on a processor Pj starts at the time equal to the ready-time of Ti on Pj, i.e., the time when the data required for execution of Ti is sent by its immediate predecessors tasks to processor Pj. The search continues until finding the first idle time slot that is capable of holding the computation cost of task Ti. The HEFT algorithm has an $O(e \times q)$ time complexity for e edges and q processors. For the dense graph when the number of edges is proportional to v^2 , the time complexity

is of the order of $O(v^2 \times p)$.

However, the performance of HEFT algorithm for DAG with high value of CCR can be improved because in HEFT, processor selection phase is based only on the task selected for the schedule. This algorithm does not consider the immediate critical child of the task. The selected task and critical child can be executed on different processors, which involve communication cost for data transfer between the two processors on which these tasks are scheduled. Due to this communication cost, the schedule length of the algorithm of the algorithm is increased. As suggested by the authors of HEFT, the critical child of the task is considered in AHEFT while scheduling it in the processor selection phase to improve the performance of HEFT.

11. THE SIMULATION ENVIRONMENT.

Simulation environment requires an input application in the form of DAG's, which can be randomly generated. To generate different size random DAG's many input parameters are required such as maximum out degree of a task, number of tasks, CCR and average computation cost of DAG. Maximum out degree of a task gives the maximum number of children of the task. Random number of children is generated for a task using a random number function. Number of tasks gives the size of the input application. CCR of DAG gives nature of application i.e., whether it is communication intensive or computation intensive. Simulation environment also requires a network of fully connected heterogeneous processors for executing the input application. To generate a fully connected network, two parameters viz. (i) Number of processors and (ii) Heterogeneity factor (β) are needed. The number of processors gives the size of the network while β is range percentage of computation costs on processors. A high percentage value causes significant difference in a task's computation cost among the processors and low percentage indicates that the expected computation time of task is almost equal on any of the given processors.

The HEFT and AHEFT algorithms are compared on the basis of performance indices by varying the number of tasks in the randomly generated input application. Varying the number of processors in the randomly generated network also does the comparison. The algorithm, which gives low value of schedule length and high values of speedup factor and efficiency, is better one.

While performing the simulation, task execution of a given application is assumed to be non-preemptive . Further, the startup costs

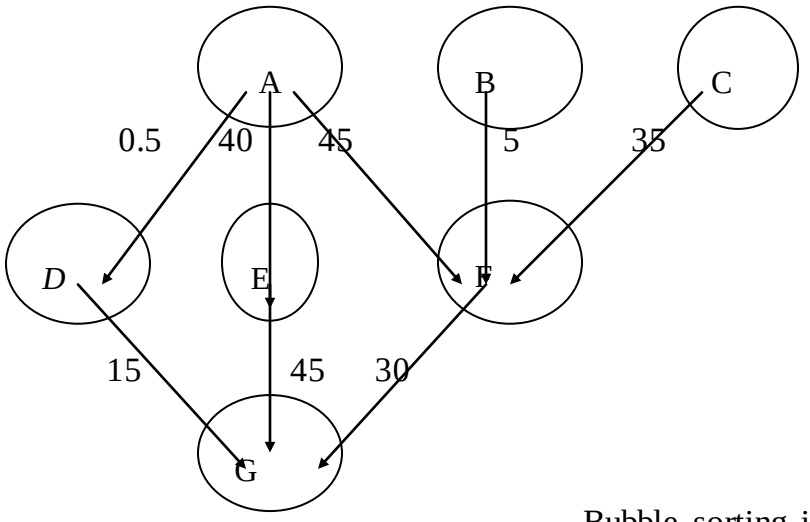
of the processors are also taken to be negligible as compared to the execution cost of tasks. In addition, communication links between processors are ideal which means that inter processor communication is same for any pair of processors.

15. EXTENDED AHEFT TASK SCHEDULING ALGORITHM

As an extension to the AHEFT algorithm, another algorithm called EAHEFT can be proposed by modifying the AHEFT algorithm.

If more than one entry tasks are allowed in the input application DAG, then it produces better results if all the entry tasks of the DAG are scheduled first before any other tasks, since they don't have any parents i.e., no data dependenies. And then scheduling the remaining tasks based on the EFT values on the available processors. This modified algorithm can be named as Extended AHEFT algorithm

INPUT APPLICATION TASK GRAPH (DAG):



EAHEFT algorithm also has two phases similar to HEFT, which are explained as follows,

(i) Task Prioritizing Phase :

EAHEFT uses the similar policy for task prioritizing as in AHEFT. This phase requires the priority of each task to be set with the $rank_u$ value, which is based on mean computation and communication costs. Bubble sorting is used for sorting the tasks by decreasing order of $rank_u$. In the case of a tie (two tasks having same upward rank), the priority is given randomly.

(ii) Processor Selection Phase:

The entry tasks of the input application are scheduled first

(because they do not have any parent tasks) and then the other tasks are scheduled according to their ranks. Further, the critical child is also taken into consideration.

In this phase, a different strategy is undertaken for scheduling the task on the processor. Firstly, the critical child of the task (selected from the scheduled list for scheduling) is found. If all the parent tasks of this critical child other than the selected task are already scheduled then the selected task and its critical child both are scheduled on the same processor, which gives minimum EFT for critical child. Otherwise, the selected task is assigned to the processor, which gives minimum EFT for that task.

For DAGs with lower values of CCR, the EAHEFT performs similar to AHEFT. However, when CCR value is increased, its performance improves over AHEFT because the selected task and its critical child both are executed on the same processor. So, there is saving in the data transfer cost or communication cost and the schedule length is lower than that in AHEFT.

Pseudo Code for EAHEFT in detail

Step 1: The computation costs of the tasks and communication costs of the edges are set to their mean values.

Step 2: The DAG is traversed from bottom to top (from exit task to entry task) and $rank_u$ of each task is calculated recursively.

Step 3: For each task in the DAG, assign priority(Ti) = $rank_u$(Ti).

Step 4: All the entry tasks i.e., which don't have parent tasks are first scheduled based on the best EFT value on to the best suited processors.

Step 5: A ready list is prepared by sorting the remaining tasks in the decreasing order of their priority values.

Step 6: Till there are unscheduled tasks in the ready list

Do

Step 6.1: From the ready list, select the first task Ti, for scheduling.

Step 6.2 : Find critical child Tc of task Ti by using the formula

$$\max \left\{ rank_u(Tc) + Ci,c \right\}$$

$Tc \in succ(Ti)$

Where Ci,c is the communication cost from parent task Ti to the child task Tc.

Step 6.3: If all the immediate predecessors (other than the selected task) of critical child are already scheduled then,

Step 6.3.1: For each processor Pk in the processor set,

Step 6.3.1.1: Compute EFT (Tc , Pk).

Step 6.3.2: Assign task Ti and Tc to the processor Pj that minimize the EFT of task Tc.

Step 6.4: Else

Step 6.4.1: for each processor Pk in the processor set,

Step 6.4.1.1: Calculate the EFT (Ti , Pk).

Step 6.4.2: Assign task Ti to the processor Pj that minimize the EFT of the task Ti.

Step 7: End While.

16. ASSUMPTION CONSIDERED FOR THE ABOVE ALGORITHMS

16.1 HEFT:

- In the task graphs for the input applications, only one entry task and one exit task are present.
- Tasks will get executed with no preemption
- If more than one entry tasks are present, then among them only one is designated as entry task.

16.2 AHEFT:

- In the task graphs for the input applications, only one entry task and one exit task are present.
- If more than one entry tasks are present, then among them only one is designated as entry task.
- For every task, critical child is considered for processor allocation.

16.3 Extended AHEFT:

- More than one entry tasks are allowed and taken into consideration in the input DAG.
- All the entry tasks are scheduled first, since they do not have parents i.e., no dependencies
- Then the remaining tasks are scheduled based on their upward ranks.

CONCLUSION

As the number of tasks is increased the schedule length obtained by using AHEFT is less as compared to that in HEFT. For low value of CCR and less number of tasks both performs similar. Moreover, as the number of processors is increased in the network, the system is used more efficiently when AHEFT algorithm is used for scheduling.

Also, as the number of tasks is increased on average the schedule length obtained by using EAHEFT is less as compared to that in AHEFT. For low value of CCR and less number of tasks both performs similar. Moreover, as the number of processors

is increased in the network, the system is used more efficiently when EAHEFT algorithm is used for scheduling.

REFERENCES

1.”Performance-Efective and low complexity Task Scheduling for Heterogeneous Computing”. IEEE Trans on Parallel & Ditributed System”.Vol.13,no.3,march 2002.

2.”Guest Editorial Heterogeneous Computing”,by Sameershivle,Prasanna Sugavanam, HJ Siegel .IEE Heterogeneous Computing Workshop (Apr,2004).

3.G Gabrani & Abhishek Hare, “An Algorithm For Task Scheduling in Heterogeneous Computing Environment”, IETE Journal of Education ,Vol 45, No 3, July – September 2004,pp 123 – 134.

4. E llavarsan & P Thambidurai, “A New Approach For Scheduling Directed A-cyclic Precedence Constrained Task Graphs onto Distributed Heterogeneous Computing System”, Proceedings of Asia pacific Conference ObCom APC-2004.

5. Babak Hamidzadeh , David J. Lilja &Yacine Atif,” Dynamic Scheduling Techniques for Heterogeneous Computing Systems”, Concurrency: Practice & Experience, October 1995, Minnesota University of Science & Technology.

6.”Bech Marking &Comparision of the Task Graph Scheduling Algorithm”. Journal of Parallel &Distributed Computing(1999).

7.”From Heterogeneous Task Scheduling To Heterogeneous Mixed ParallelScheduling”.Frederic Suter,Frederic Desprez,Henri csanova.

8.” An Experimental Investigation In to rank Function of Heterogeneous Earliest Finish time”by Henanzhao&Rizossakavaroiu.