Effective Processor selection on Heterogeneous Computing

Balamurugan M
M.Tech Student
Dept. of Computer Science and Engineering
Pondicherry Engineering College
Pondicherry-60514
Email: tmbalagan@pec.edu

Dr. V. Akila
Assistant Professor
Dept. of Computer Science and Engineering
Pondicherry Engineering College
Pondicherry-60514
Email: akila@pec.edu

Abstract—Effective scheduling of a distributed application is one of the major issues in distributed computing systems since scheduling algorithms are playing an important role in achieving better performance. A popular representation of an application is directed acyclic graph (DAG), The DAG scheduling problem has been shown to be NP-complete. To introduce an Effective Processor Selection (EPS) algorithm to effectively schedule the application tasks onto the heterogeneous processors. The algo-rithm is mainly focused on minimizing the application execution time and best processor selection. Although few algorithms in literature for heterogeneous processors, they usually require earliest finish time (EFT) to finish schedule. We present with an objective to meet performance and fast scheduling time, EPS algorithm select the tasks at priority to assign the task. On other hand, processor selection based on constraint satisfaction Problem (CSP) feature.

Keywords—static task scheduling, DAG scheduling, list scheduling, CSP.

I. INTRODUCTION

Different sets of resources can be globally or locally distributed called as heterogeneous computing. The resources with different computing capabilities and different computing speeds. In heterogeneous computing environments, several is-sues need to consider one of as scheduling tasks. The objective of task scheduling is to minimize the overall completion time. Task scheduling problem for heterogeneous is more complex than the homogenous computing systems because in heterogeneous different execution time and communication time among the processors.

It has broadly classified static and dynamic scheduling. In static scheduling, the execution time and communication cost is known beforehand and dynamic scheduling made at runtime. Static scheduling can be classified heuristic based, guided random, search based[1]. Heuristic based give a good solution with polynomial time complexity. The heuristic based classified of three subcategories list, cluster and duplication scheduling[3]. List scheduling consists of two phases: a task prioritization phase, where each task assign certain priority is compute and assigned to each node of the Directed Acyclic Graph (DAG) and process selection (in order of priority) is assigned to processor that minimize execution time. The DAG scheduling problem has been shown to be NP-complete[7], even in this field have been mainly focused on obtaining low complexity heuristics that produce good schedules.

In this paper, we present a new algorithm called Effective Processor Selection is developed for best processor selection. The motivation behind this algorithm to assign the best pro-cessor on each task. The processor selection for neighbour is inconsistent. Consider the same processor have been allocated for parent and child node. When the parent did not complete within Earliest Finish Time (EFT), the child has to wait till the period of EFT. Finally, our algorithm could best processor to fit on each task.

The paper is organized as follows: Section II discuss problem definition. Section III gives an overview of the related works. Section IV presents our developed scheduling algorithm namely EPS. Section V Discussion And Expecting Results and finally concluded in section VI.

II. PROBLEM DEFINITION

We present an algorithm a single job onset of P processor best fit without inconsistent.

A typical workflow application can be represented by a Directed Acyclic Graph[9][8] (DAG), In a DAG, an individual task and its dependency are represented by a node and its edges. A dependency ensures that a child node cannot be exe-cuted before all its parent tasks finish successfully and transfer the required child input data. The task computation time and communication time is modelled by assigning a weight to nodes and edges respectively. A DAG can be modeled by a tuple G(V, E) where V is the set of v nodes and each node v_i 2 V represents an application task, and E is the set of communication edges between tasks. Each edge e(i; j) 2 E represents the task-dependency constraint such that task v_i should complete its execution before task v_j can be started. In a given DAG, a task with no predecessors is called an entry task and a task with no successors is called an exit task. We assume that the DAG has exactly one entry task n_{entry} and one exit task n_{exit}. The speed of the interprocess communication network is negligible. Therefore, two tasks are scheduled on the same processor the communication cost between them is ignored.

A task graph with 10 tasks, and its computation cost matrix given[2], are shown in figure 1 and Table 1.

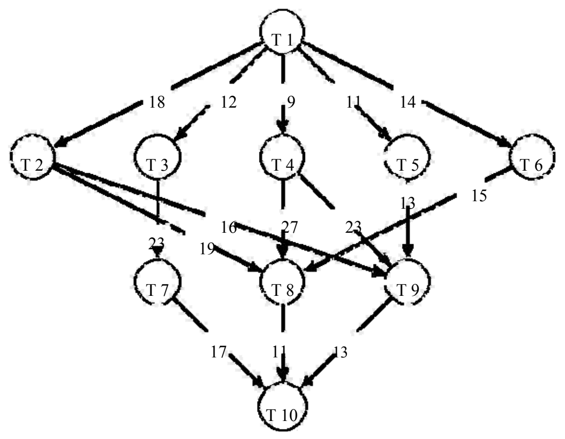

Fig. 1: DAG

TABLE I: Computation Cost Matrix

Task	P1	P2	P3
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	6
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

We present some common attributes used in task schedul-ing:

Definition 1. EST(n_i; p_j) and EFT(n_i; p_j) are the earliest start time and the earliest finish time of task n_i on processor p_j, respectively.

$$EST\ (n_{entry}; p_j) = 0 \qquad (1)$$

Definition 2. EST(n_i; p_j) is defined as

$$EST\ (n_i; p_j) = maxfavail[j]; \ n_m \ \overset{max}{2} \ {}_{pred(n_i)}(AFT\ (n_m)+c_{m;j})g \qquad (2)$$

Where pred(n_i) is set of immediate predecessor tasks n_i and avail[j] is the earliest time at which processor p_j, is ready task execution. If n_k is the last assigned task on the processor p_j, then avail[j] is the time that processor p_j completed the execution of a task when we have a non-insertion based scheduling policy. The inner max block in the EST equation returns the ready time, i.e., the time when all data needed by n_i has arrived at processor p_j.

After a task n_m is scheduled on a processor p_j, the earliest start time and the earliest finish time of n_m on processor p_j is equal to the actual start time, AST (n_m), and the actual finish time, AFT (n_m), of task n_m, respectively. After all tasks in a graph are scheduled, the schedule length (i.e., overall completion

time) will be the actual finish time of the task n_{exit}. If there are multiple exit tasks and the convention of inserting a pseudo

exit tasks is not applied, the schedule length (which is also called makespan) is defined as

$$makespan = maxfAFT\ (n_{exit})g \qquad (3)$$

The objective function of the task scheduling problem is to schedule the tasks of an application to the processor such that is schedule length is minimized.

Definition 3. EFT(n_i; p_j) is defined as

$$EFT\ (n_i; p_j) = EST\ (n_i; p_j) + w_{i;j} \qquad (4)$$

Which is EST of a task n_i on a processor p_j plus the computational cost of n_i on a processor p_j.

Definition 4. Constraint Satisfaction Problem

Which is EFT of a task n_i on a processor p_j best fit to set each task without any inconsistent[11][10]. The selection policy based Forward Checking and Degree Heuristic. The objective function of selecting processor is to schedule the tasks independently, to the processor such that is minimize inconsistent among the adjacent nodes.

III. Related Work

Efficient application scheduling is critical for achieving high performance a heterogeneous computing system, because of its key importance of performance, the scheduling prob-lem extensively studied various heuristics have been in the literature [3][1][5][6][4][2]. The heuristics are classified into a variety of schemes such as priority based, guided random search based, cluster based and duplication based.

Priority for each task that is utilized to assign the tasks to the different processors. Priority based scheduling algorithm such as Heterogeneous Earliest Finish Time (HEFT), Per-formance Effective Task Scheduling (PETS), Predict Earliest Finish Time (PEFT). The complexity of HEFT, PETS, PEFT algorithm is $O(v^2{:}p)$, $O(v^2(p{:}logv))$, $O(v^2{:}p)$ respectively. The improvement of algorithm interms of Speedup, Schedule length, Efficiency.

IV. Effective Process Selection (EPS) Algorithm

A. Task Prioritization Approach

The proposed algorithm consists of three phases: level sorting, task prioritization and processor selection.

In the first phase, the given DAG is traversed in top-down to sort task at each level in order to group tasks that are independent of each other. Level 0 contains entry task and the last level comprises of some of the exit tasks.

In the second phase of the algorithm, priority is compute for each task to assign. Priority is computed based on a com-munication cost and average computation cost. The Average Computation Cost (ACC) of a task is the average computation cost on all the available p processor and it computed by using.

$$ACC(n_i) = (5)\ \frac{\overset{p}{\underset{j=i}{x}} w_{i;j}}{p}$$

We have defined two new attributes of communication cost for a task. Task Initial Cost (TIC) and Task Receiving Cost (TRC). TIC of a task is the amount of cost incurred to transfer the data to all its immediate successor. The exit task is 0, for all tasks at level i, compute by using.

$$TIC(n_i) = \overset{b}{\underset{i=1}{x}} out_{i;j} \qquad (6)$$

b is the number of immediate successor of n_i TRC of a task is the cost incurred to receive data from its immediate predecessor tasks. The TRC compute by using

$$TRC(n_j) = MaxfTEEC(n_j)g \qquad (7)$$

where n_j is the predecessor of n_j

Task Expected Execution Cost (TEEC) of a task n_i ($TEEC(n_i)$) is sum of its TIC, TRC and ACC values of that task. For every task at each level i, ($TEEC(n_i)$) is compute by using.

$$TEEC(n_i) = TIC(n_i) + TRC(n_i) + ACC(n_i) \quad (8)$$

Priority is assigned to all tasks at each level i, based on its TEEC value. At each level, the task with highest TEEC value receives the highest priority followed by task with next highest TEEC value and so on. the computed ACC, TIC, TRC, TEEC and priority value for each task.

B. Process Selection Approach

The processor selection for neighbour is inconsistent. Con-sider the same processor have been allocated for parent and child node. When the parent did not complete within EFT, the child has to wait till the period of EFT. So algorithm based on feature is its processor selection policy. To select the processor according to the "Constraint Satisfaction Problem" (CSP)[11][10], First of all, to calculate EST and EFT each task on processors. And assign the processor to allocate unassigned tasks in search based technique, if any processor has occurred tie (i.e)., the same processor to assigned in the adjacent task, the degree heuristic phase resolve the tie by backtracking. Forward checking as a processor to allocate unassigned tasks. And given scheduling can be done expected EFT with best processor

C. The EPS Algorithm

Algorithm 1 :Task Prioritization

```
1: Generate the DAG
2: Sort the DAG levels according to the dependency ordering
3: For all tasks at each level $L_i$ do
4: Compute ACC, TIC and TRC
5: ComputeTEEC($n_i$) = TIC($n_i$)+TRC($n_i$)+ACC($n_i$)
6: Construct Priority queue using TEEC value
```

Algorithm 2 :Process Selection

```
1: Compute EFT
2: Compute forward checking ($L_i$; $n_i$; $p_j$)
3: While there are unallocated tasks in the list
4: According to the CS problem of search with forward
   checking and adjacent cannot be same processor.
5: if check tie occur in the same level then
6: Backtracking to replace the processor for current task
7: Assign best processor to each task ($n_i$)
8: end while
```

V. Discussion and Expecting Results

To test performance of EPS algorithm with other algorithm a set of p processor allocation that determines, whether its inconsistent or not. According to the CSP to assign the processor to unallocated tasks first, whether its critical path or adjacent had the same processor resolve by backtracking.

Result for PETS:

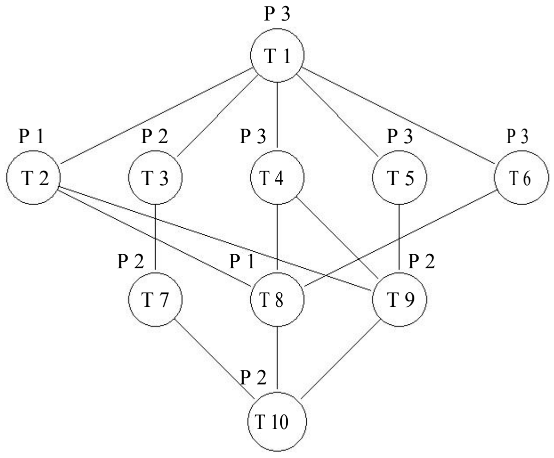

Fig. 2: PETS

Result for our algorithm (EPS) processor selection:

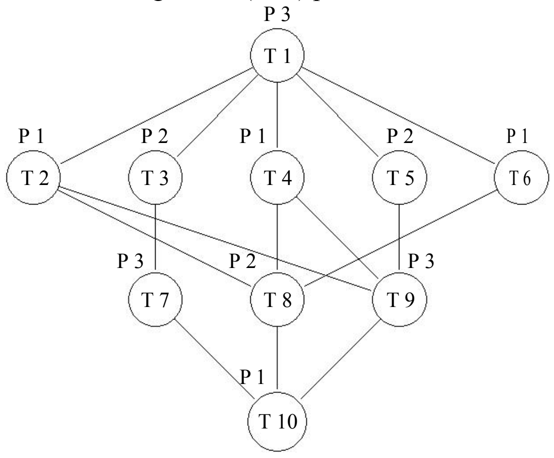

Fig. 3: Processor selection our algorithm (EPS) for PETS

TABLE II: Critical Path for PETS

P3	P3	P2	p2
T1	T5	T9	T10

TABLE III: Critical Path for EPS Processor Selection

P3	P2	P3	p1
T1	T5	T9	T10

VI. Conclusion

The task scheduling algorithm EPS proposed here has been better for scheduling DAG structured application onto heterogeneous computing. The main theme of this scheduling to best fit processor for each task. According to the results to found that algorithm EPS is better way to schedule the tasks among given processor.

References

[1] H. Arabnejad and J.G. Barbosa, "List Scheduling Algorithm for Hetero-geneous Systems by an Optimistic Cost Table",IEEE Trans. Parallel and Distributed Systems, vol. 25, no.3,pp. 682-694, 2014.

[2] E. Ilavarasan P. Thambidurai and R. Mahilmannan, "Performance Effec-tive Task Scheduling Algorithm for Heterogeneous Computing System", ISPDC, 2005.

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

[4] R.Eswari, S.Nickolas, "Path-based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems", IEEE, 2010.

[5] Tarek Hagras, Jan Jane cek, "An Approach to Compile-Time Task Scheduling in Heterogeneous Computing Systems", ICPPW, 2004.

[6] Tarek Hagras, Jan Jane cek, "A Simple Scheduling Heuristic for Hetero-geneous Computing Environments", ISPDC, 2003.

[7] Lance Fortnow, "The Status of the P versus NP Problem", Northwestern University. Available: http://people.cs.uchicago.edu/ fortnow/papers/pnp-cacm.pdf.

[8] Shane Saunders, "Improved Shortest Path Algorithms for Nearly Acyclic Graphs" Ph.D. dissertation, Department of Computer Science, University of Canterbury, 2004.

[9] Yu-Kwong Kwok, "Benchmarking and Comparison of the Task Graph Scheduling Algorithms", Journal of Parallel and Distributed Comput-ing,1999.

[10] Stuart Russell, Peter Norvig, "Constraint Satisfaction Problem", Artifi-cial Intelligence: A Modern Approach, University of California, Berkeley, pp. 137-160, 2015.

[11] Zhe Liu, "Algorithms for Constraint Satisfaction Problems", Ph.D. dissertation, Master of Mathematics in Computer Science University of Waterloo, 1998.