A Scheduling Approach with Processor and Network Heterogeneity for Grid Environment

Dr. Vinay Kumar
School of Computer and Systems Sciences
Jawaharlal Nehru University
Delhi, INDIA
vinay2teotia@gmail.com

Prof. C. P. Katti
School of Computer and Systems Sciences
Jawaharlal Nehru University
Delhi, INDIA

***Abstract*— Processor heterogeneity is an important issue in grid environment. In this paper, a list based task scheduling algorithm, called "critical path scheduling with t-level" (CPST) for grid computing system is proposed. There are no. of scheduling algorithms such as HEFT [1] use mean execution time based b-level for task priority and SHCP [2] use task priority based on simple critical path. In CPST, a critical path based task sequence is generated with t-level value of tasks, where variance based computation and communication cost is used. The experimental results show that CPST algorithm performs better than HEFT, SHCP and HHS algorithm in grid environment for task graphs.**

Keywords* - *Grid Scheduling; Directed Acyclic Graph; Critical Path; Heterogeneity.

I. INTRODUCTION

A grid is a type of heterogeneous computing system for aggregation and management of computational resources that provides shared access of graphically distributed heterogeneous resources that is inter-connect over high speed networks and internet. The main idea behind grid environment is to utilize the idle time of processor cycles. An efficient task scheduling [3] in computational grid environment is essential to obtain the better utilization of resources and achieving high performance. To obtain a optimal scheduling problem is NP-complete [8]. The objective of a task scheduling problem is to map tasks onto the suitable resources and to order their execution on each resource such that precedence relationships between tasks are not violated and the overall execution time, makespan could be minimized.

In general, a grid consists of heterogeneous resource over large geographical region connected through arbitrary topology. It causes more challenges for scheduling applications due to processor and network heterogeneity in grid environment. List-based scheduling algorithms are uses to solve this problem in literature oftenly. In list-based task scheduling, tasks are ordered and selected in non-increasing order of their priorities and scheduled on processors to optimize various performance metrics.

II. PRELIMINARY

A grid application is represented by a directed acyclic graph *(DAG)*, *G= (V,E)*, where *V* is a set of tasks and *E* is a set of communication edges between tasks. Each edge *(i,j)* ϵ *E* represents the precedence constraint such that task n_i should complete its execution before task n_i start. In a *DAG*, a task without any parent is called entry task and a task without any child is called an exit task. If there is more than one entry (exit) task, they are connected to a zero cost entry (exit) task with zero cost edges, which does not affect the given *DAG*. The computation cost of a task *i* is represented by τ_i and communication cost along the edge *(i,j)* is represented by $C_{i,j}$.

A grid resource model can be represented by *G = (P, Q, A, B)*, where *P* is the set of available processors [1]

$$A = [\alpha (P_i) | \alpha (P_i) \epsilon A, i=1, 2\ldots\ldots|P|]$$

$$Q = [q\ (P_i, P_j) |\ q\ (P_i, P_j) \epsilon\ Q\ \&\ i, j{=}1, 2\ldots\ldots|P|]$$

$$B = [\beta\ (P_i, P_j) |\ \beta\ (P_i, P_j) \epsilon\ B\ \&\ i, j{=}1, 2\ldots\ldots|P|]$$

Here *A* is the set of computation rate for processors P_i , *Q* is the set of communication links that connects the processors and *B* is the set of data transfer rates between two processors [1].

Some of the efficient list based heuristics are heterogeneous earliest finish time (HEFT) [1], critical path on a processor (CPOP) [4], hybrid heuristic scheduling (HHS) and scheduling with heterogeneity using critical path (SHCP) [2]. In some algorithm [1, 2, 4], task node value in heterogeneous system is considered as an average, median, best or worst value. In [2], author Zhans gives a relative performance comparison of scheduling algorithms in grid environment. Zhans conclude that HEFT and HHS perform better than level-based scheduling methods on many combination of computing environments.

The HEFT is an insertion based static list heuristic that assign priorities of task on the basis of top-rank calculation. The top priority tasks are selected for schedule on processor which finishes its execution at earliest. The HHS use the hybrid technique of list based and level based scheduling techniques, it partitioned the directed acyclic graph into levels of independent tasks and tasks in each level are ordered and scheduled. The SHCP compute the priority of task by consider heterogeneity of processors. The priorities of tasks decide the execution order of tasks.

In this paper, a new approach for computing the priority of task is adopted considering heterogeneity of processors and *t*-level value for critical path of given *DAG*. A critical path in a *DAG* is the longest path in the *DAG* by considering the computation and communication cost. The *t*-level of a node n_i is the length of longest path from entry node to n_i with excluding computation cost of task n_i. The priorities of tasks decide the execution order of task which reflects the schedule length of task graph. The experimental result shows that CPST algorithm performs better for running large task graphs in grid environment at high CCRs (communication to computation ratio).

Heterogeneity is a type of variability in characteristics of resources. The resources can be computational and communicational of any distributed system. Here the means of characteristic of resources are bandwidth, execution rate etc. we considered a heterogeneity model [2, 5] which define two factor, processor heterogeneity factor and network heterogeneity factor to estimate the expected computation costs of tasks and expected communication cost of edges in the given DAG.

The processor heterogeneity factor ρ can be computed as [2]

$$\rho = \frac{2 \times \sqrt{\sum_{i=1}^{|P|} \{\alpha\ (P_i - \bar{\alpha}(P_i))\}^2 \Big/ |P|}}{\max(\alpha\ (P_i))} \qquad \ldots\ldots (1)$$

Where $\alpha(\overline{P_i})$ is the mean processing rate, which can computed as

$$\alpha(\overline{P_i}) = \frac{\sum_{i=1}^{|P|} \alpha\ (P_i)}{|P|} \qquad \ldots\ldots. (2)$$

High heterogeneity among the processor shows by high value of α.

The network heterogeneity factor σ can be computed as [2]

$$\sigma = \frac{2 \times \sqrt{\sum_{i=1}^{|P|} \sum_{j=i+1}^{|P|} \{\beta\ (P_i, P_j) - \bar{\beta}(P_i, P_j)\}^2 \Big/ \left(\frac{|P|^2 - |P|}{2}\right)}}{\max(\beta\ (P_i, P_j))} \qquad \ldots\ldots (3)$$

Where $\bar{\beta}(P_i, P_j)$ is the mean transfer rate, which can be calculated as

$$\bar{\beta}(P_i, P_j) = \frac{\sum_{i=1}^{|P|} \sum_{j=i+1}^{|P|} \beta\ (P_i, P_j)}{\left(\frac{|P|^2 - |P|}{2}\right)} \qquad \ldots\ldots (4)$$

It is assumed that data transfer rate between two processors without any direct link is zero. So, the expected computation cost of task nodes can be computed as [2]

$$\overline{w}_i = \frac{\tau_i}{\rho\times\min(\alpha\,(P_i) + (1-\rho)\times\max(\alpha\,(P_i)))} \quad \text{…….. (5)}$$

Similarly, expected communication cost of edge from task n_i to n_j can be computed as [2]

$$\varepsilon_{ij} = \frac{C_{i,j}}{\sigma\times\min(\beta\,(P_i,P_j)) + (1-\sigma)\times\max(\beta\,(P_i,P_j))} \quad \text{…….. (6)}$$

Where τ_i is computation cost for task n_i and $C_{i,j}$ is communication cost occurring along the edge *(i,j)*.

III. PROPOSED WORK

In this phase, a critical path based sequence of tasks is generated. The nodes along this path are called *CP*-nodes. After constructing critical path, other nodes are added to keep the precedence constraint order of task execution. In this, the successor nodes are added on the basis of higher *t*-level and ties being solved randomly as in [6] and [9]. Other remaining nodes are added using the same priority as assigned to them at the end of task sequence.

The *t*-level of task n_i is the longest directed path considering computation and communication cost from entry node to node n_i with excluding computation cost of n_i. It can be calculated as

$$t_i = max\,(t_j + \bar{\varepsilon}_{ij}),\ \forall\, n_j \in pred\,(n_i) \quad \text{……. (7)}$$

Where *pred* (n_i) represent the immediate predecessor of task node n_i in the *DAG*. The first unscheduled task in the task sequence is known as candidate task. The unscheduled selected candidate task is mapped to the processor in the processor matrix which allows it to finish at earliest. The task n_i can start its execution on the candidate processor if data arrive from all of its immediate parents so as to meet precedence constraints.

To select best processor for the candidate task , it is necessary to define the earliest start time *(EST)* and earliest finish time *(EFT)* of task n_i on processor p_j. For entry task node

$$EST\,(n_{entry}\,,p_j) = 0 \quad \text{...…. (8)}$$

For other tasks, the *EST* and *EFT* can be calculated as

$$EST\,(n_i\,,p_j) = max\,[avail\,(\text{j}),\ \underset{n_m\,\in\,\text{pred}\,(n_i)}{max}\,(AFT(n_m) + \frac{C_{m,i}}{\beta\,(P_k\,,P_j)})\,] \quad \text{….. (9)}$$

$$EFT\,(n_i\,,p_j) = \frac{\tau_i}{p_j} + EST\,(n_i\,,p_j) \quad \text{….. (10)}$$

Where *AFT* (n_m) is the actual finish time of task n_m, *avail(j)* is the time when processor p_j is ready to execute new task in non-insertion based scheduling policy [10]. After assignment of all tasks in a *DAG*, the makespan of the schedule will be

$$\text{Makespan} = max\,[AFT(n_{exit})] \quad \text{……. (11)}$$

Here, a task scheduling algorithm (CPST) for grid environment presented. The pseudo code of the algorithm is presented in Fig. 1. CPST algorithm use t-level approach for minimizing makespan of a DAG, because *t*-level not include the computation time of current task. So algorithm is free to assign this task to best processor, whenever other algorithms HEFT, CPOP and SHCP are not free to assign the current task to any best processor.

The CPST algorithm works in two phases, in first phase a task sequence generated on basis of critical path with *t*-level for computing priorities of tasks using processor and network heterogeneity factor. In second phase select the tasks in order of their priorities and schedule to a processor which minimizes the task's completion time.

As an illustration, Fig. 3 presents the schedules obtained by CPST algorithm for a sample DAG of Fig. 2. The schedule length, which is 80, is shorter than the schedule lengths of other related work. The Table 1 and 3 gives mean processing rate for processors and t-values for all tasks. The scheduling order of the tasks with respect to CPST algorithm is $\{n_1, n_4, n_5, n_3, n_6, n_2, n_8, n_7, n_9, n_{10}\}$.

```
1)  compute the computation and communication heterogeneity factor by equ (1) and
    (3)
2)  set the weight of tasks in DAG with expected computation cost using equ (5)
3)  set the weight of edges in DAG with expected communication cost using equ (6
4)  construct a critical path with t-level based sequence using equ (7)
5)  while  there are unscheduled tasks in task sequence in task sequence ($n_{i \epsilon N}$) do
6)  select first unscheduled task, $n_i$ from sequence
7)  for each processor $p_i (p_i \epsilon P)$  do
8)  compute EFT ($n_i$ ,$p_j$) with insertion-based task scheduling using equ (10)
9)  assign task $n_i$ to processor $p_j$ which minimize EFT of task $n_i$
10) end while
```

Figure 1: CPST Algorithm

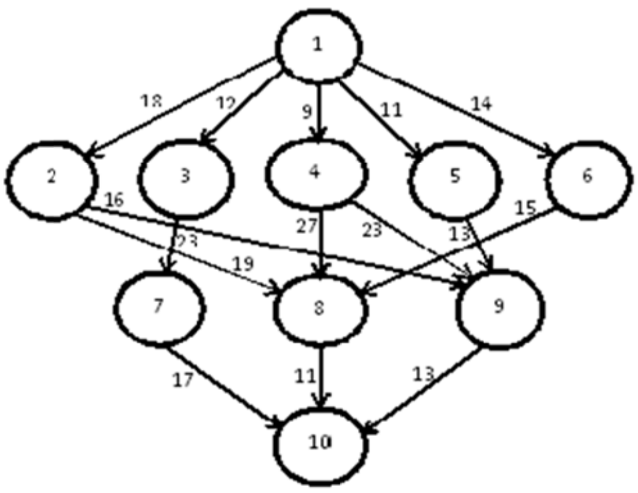

Figure 2: DAG form of a task scheduling problem

Table 1: mean processing rate

processor	α (P_i)	α($\overline{P_i}$)
p_1	127	42.3
p_2	130	43.3
p_3	143	47.6

From equ (1) and (3)

processor heterogeneity factor ρ = 0.713

network heterogeneity factor σ = 0.571

Table 2: Computation Cost Matrix

	p_1	p_2	p_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

Table 3: *t*-values of tasks

task	*t*-level	t_i
n_1	0	0
n_2	31	1.8
n_3	25	1.2
n_4	22	.9
n_5	24	1.1
n_6	27	1.4
n_7	62.3	5.3
n_8	66.6	4.8
n_9	63.6	6.3
n_{10}	93.2	9.01

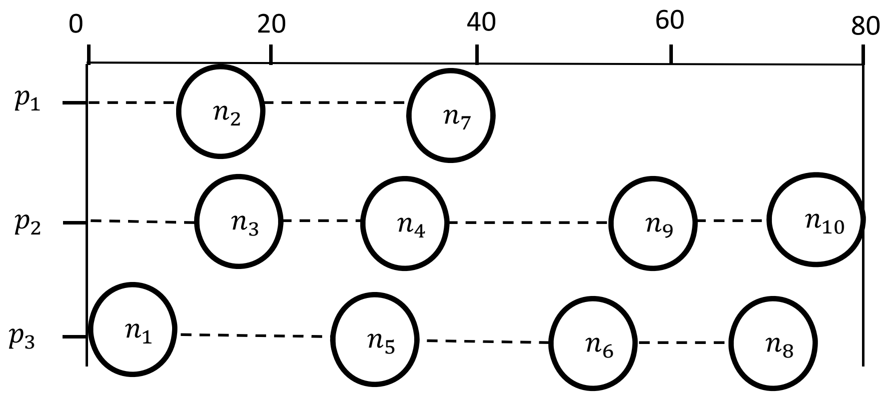

Figure 3: Scheduling length by CPST

Sort task t_i values in increasing order. That is $n_1, n_4, n_5, n_3, n_6, n_2, n_8, n_7, n_9, n_{10}$. Now calculate *EFT* $(n_i\ ,p_j)$ for each task and assign this task to processor that has minimum value for it. By this we get minimum makespan 80 with heterogeneity factors ρ and σ.

IV. EXPERIMENTAL RESULT AND SIMULATION

In the section, we give the comparative evaluation of CPST algorithm with other algorithms like HHS, HEFT and SHCP. Performance of CPST algorithm is evaluated by using experiments on MATLAB platform (version R2009a). We have implemented the CPST algorithm and compared the schedule produced on a variety of random task graphs in heterogeneous grid environment as described in [7].

The following performance metrics are the basis of comparison of algorithms.

<u>*Schedule Length Ratio*</u>: since a large set of task graphs is used. So it's necessary to normalize the schedule length to a lower bound. That is called schedule length ratio (*SLR*) and define as [11]

$$SLR = \frac{makespan}{\sum_{n_i \epsilon CP_{min}} [\ \min_{p_j \epsilon P} (\frac{\tau_i}{p_j})]} \qquad \text{…….. (12)}$$

The denominator is the summation of minimum execution costs of tasks on the CP_{min} (minimum critical path).

<u>*Speedup*</u>: it can be calculated by dividing minimal sequential execution time with the parallel execution time[12].

$$Speedup = \frac{\min_{p_j \epsilon P}[\sum_{n_{i\epsilon N}}(\frac{\tau_i}{p_j})]}{makespan} \qquad \text{....... (13)}$$

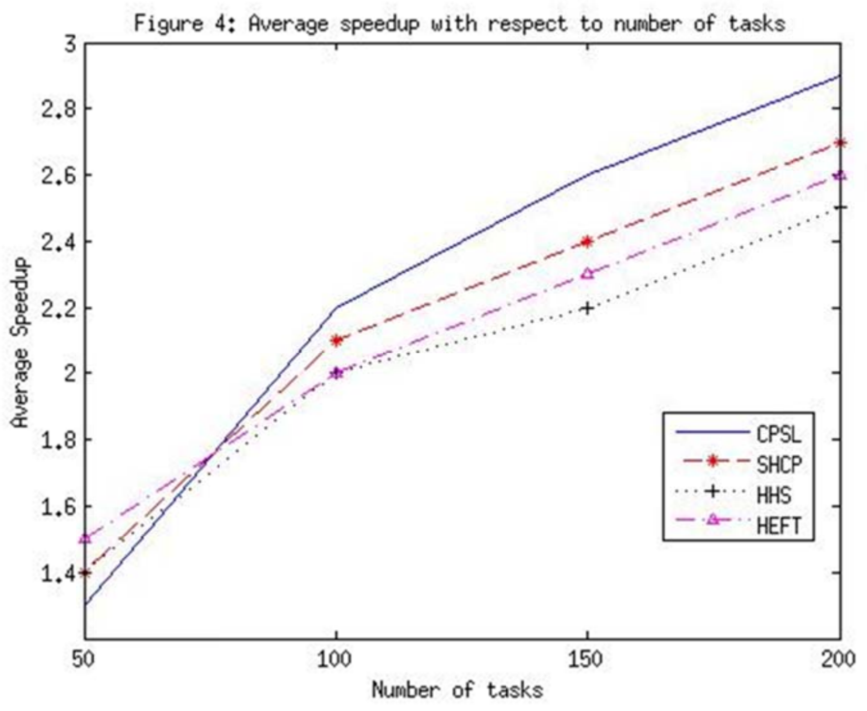

Figure 4: average *speedup* with respect to number of tasks

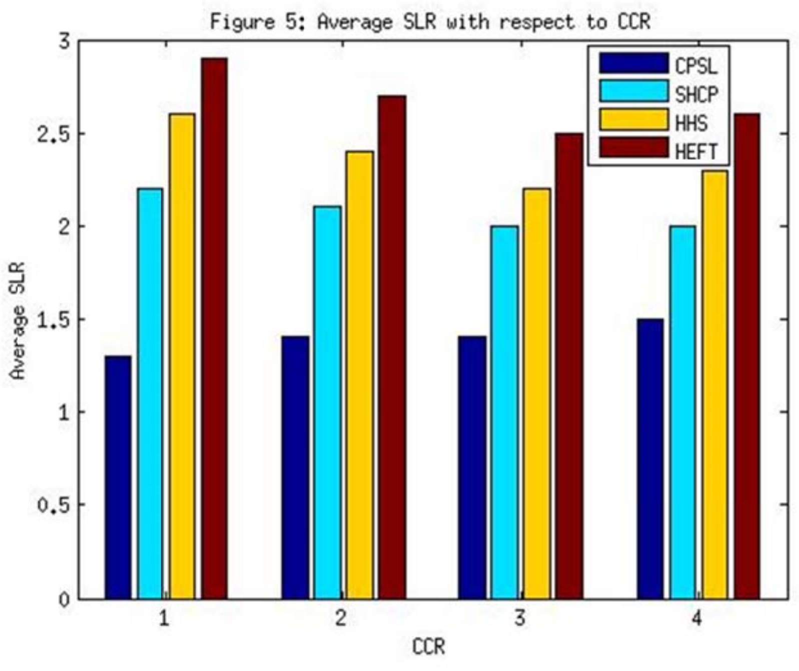

Figure 5: Average *SLR* with respect to *CCR*

The results are obtained with respect to average *SLR* and speedup over different task sizes and *CCRs*. Each result is obtained with respect to graph size in an average of 100 graphs and with respect to *CCR* in an average of 100 graphs. The experimental results (Fig. 4 and Fig. 5) show that CPST algorithm gives better or equal result in comparison of HEFT, HHS and SHCP algorithm for large task graphs and high *CCRs*.

V. CONCLUSION

This paper presents a new approach of task scheduling in grid environment when heterogeneity of resources and heterogeneity of network are two important factors in a task scheduling problem. The *CCR* value is more in heterogeneous environment. The result shows that the CPST algorithm improves with increase of task sizes and *CCRs*. So for large task graphs at higher *CCRs*, CPST gives good results as compare to HHS, HEFT and SHCP.

REFFERENCES

[1] G.C. Sih and E. A. Lee, “A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures”, IEEE Trans. Parallel and Distributed System vol. 5, no. 2, Feb. 1994, pp. 113-120.

[2] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, “Heuristics for Scheduling Parameter Sweep Applications in Grid Environments”, In. Proc. Of the 9th heterogeneous Computing Workshop, Cancun,Mexico, 2000, pp. 349-363.

[3] M. Iverson, F. Ozguner, and G. Follen, “Parallelizing Existing Applications in a Distributed Heterogeneous Environment”, Proc. Hetrogeneous Computing Workshop, 1995, pp. 93-100.

[4] H. El-Rewini and T.G. Lewis,”Scheduling Parallel Program Tasks onto Arbitrary Target Machines”, Journal of Parallel and Distributed Computing, vol. 9, 1990, pp. 138-153.

[5] A. Khokhar, V. K. Prasanna, M. Shaaban, and C. L Wang, “Heterogeneous computing: Challenges and opportunities”, IEEE Computer, Vol. 26, No. 6, June 1993, pp. 18-27.

[6] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen. and R. F. Freund, “Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems”, International Journal of Parallel and Distributed Computing, Vol. 59(2), 1999, pp.107-131.

[7] H. Topcuoglu, S. Hariri, Min-You Wu, “Performance effective and low complexity task scheduling for heterogeneous computing”, IEEE Trans. Parallel and Distributed System vol. 13, no. 3, Mar. 2002, pp. 260-274.

[8] H. J. Siegel, J. K. Antonio, R. C. Metzger, M. Tan, and Y. A. Li, “Heterogeneous computing”, in Parallel and Distributed Computing Handbook, A. Y. Zomaya, ed., McGraw-Hill, New York, NY, 1996, pp. 725-761.

[9] I. Foster and C. Kesselman (editors), “The Grid: Blueprint for a Future Computing Infrastructure”, Morgan Kaufmann Publishers, USA 1999.

[10] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, International Journal of Parallel and Distributed Computing, Vol.61(6), 2001, pp. 810-837.

[11] H. S. Stone, “Multiprocessor Scheduling with the aid of network flow algorithms”, IEEE Trans. Software Eng. 3, 1977, pp. 85-93.

[12] H. J. Siegel, H. G. Dietz, and J. K. Antonio, “Software support for heterogeneous computing”, in The Computer Science and Engineering Handbook, A. B. Tucker, Jr., ed., CRC Press, Boca Raton, FL, 1997, pp. 1886-1909.