Standard Deviation of Probability of Processor Availability based Task Scheduling Algorithm

Vijayalakshmi A. Lepakshi [1]
[i]*Research Scholar, NHCE, Bangalore, VTU, India*
vjlepakshi@gmail.com

Dr. Prashanth C S R [2]
[2]HOD, Department of Computer Science and Engineering, NHCE, Bangalore, VTU, India
drprashanthcsr@gmail.com

Abstract **–** Cloud [2] is an interconnection of clusters of different heterogeneous processors that provides different services such as SaaS, PaaS, IaaS etc., to different users. User's jobs are diverse and have different requirements. Jobs can be represented as Directed Acyclic Graphs and scheduling these jobs is a known NP-Complete problem. There are many heuristics developed earlier for job scheduling, that work differently in different environments with different requirements. A new heuristic SDPA, using Standard Deviation of probability of resource availability is developed for scheduling tasks which reduces makespan. It works in two stages. During first stage jobs are arranged based on the upward ranks and during the second stage jobs are allotted to the processor based on the standard deviation of probability of processor availability. Performance metrics such as Speedup, SLR and makespan are used to compare the performance of the new heuristic with existing one.

Keywords- Cloud Computing, Cloud Services, DAGs, Makespan, Resource Availability, Task Scheduling,

I. INTRODUCTION

In a Heterogeneous environment like cloud, users require various resources for computation depending on their job. The resources range from Infrastructure to Computation or Services on payment basis. When user requests for certain resources, resources will be allocated for job execution based on availability. Multiple resources may be available on heterogeneous network for execution with different speeds and costs of execution. Resources in Cloud are globally placed, temporal in nature and the load on the resources such as CPU may increase due to various reasons such as pre-allocation of resources to jobs or local scheduling of jobs to those resources. The present task is selecting resources and scheduling the job to those resources that minimizes the makespan by reducing the cost. A job may be a single or parallel application. Scheduling parallel applications modelled by Directed Acyclic Graphs onto a network of heterogeneous computers is a NP - Complete problem. Different scheduling algorithms can be used depending on the type of application to be scheduled. Existing algorithms may not work in all the environments efficiently where resource availability constraint is considered. So, in Cloud computing there is a need for developing new heuristics for scheduling tasks, since the efficiency of cloud depends on these scheduling algorithms.

In this paper we present a new heuristic called SDPA Task Scheduling algorithm, for scheduling task graphs with precedence constraints and scheduling tasks to a processor not only based on Earliest Finish time but also considering the probability of availability of processors based on its Standard Deviation.

The problem statement is defined in the next section. In Section 3, we discuss related work. Section 4 introduces a new task scheduling heuristic SDPA. Section 5 presents comparative study of SDPA and HEFT. The present research summary is given in Section 6.

II. SCHEDULING PROBLEM

Problem can be formally defined as follows: "Scheduling parallel applications modelled by Directed Acyclic Graph on to a Cloud, which is a heterogeneous interconnection of clusters, such that precedence constraints are satisfied and makespan is reduced".

A Directed Acyclic Graph is used to represent a task which is a large parallel application subdivided into sub-tasks, where subtasks are arranged in different levels with predefined precedence constraints. A sub task can be processed only when its precedence constraints are satisfied. When a task is ready for execution it is placed in a queue and upon satisfying the precedence constraints and based on availability of processors, sub tasks are placed in the ready queue of allotted processor. Tasks placed in the ready task list are processed based on the local scheduling policy along with

978-1-4673-6667-0/15/$31.00©2015 IEEE

local jobs. When local jobs are scheduled on a processor sometimes processor may not be available for global jobs, due to which delays may occur in completion of a task. In the following Figure 1 an example task graph[1] is shown and Table 1 shows computation cost matrix.

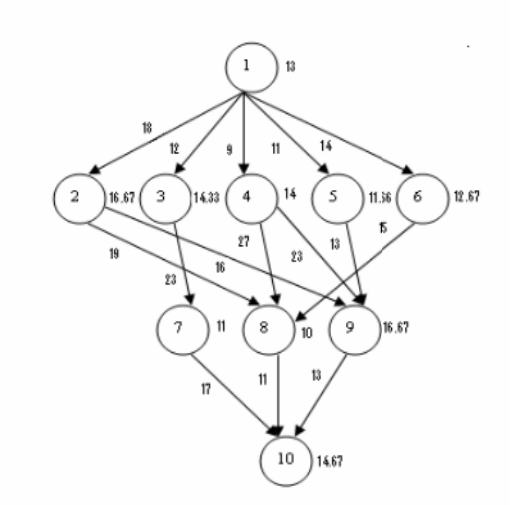

Figure 1 Example Task Graph

Table 1 Computation Cost Matrix

Task	P_1	P_2	P_3
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

III. RELATED WORK

An application Scheduling algorithm Heterogeneous Earliest Finish Time [HEFT][1] for bounded number of heterogeneous processors has two major phases. The first phase is a task prioritising phase for computing the priorities of all tasks. The second phase is known as processor selection phase in which tasks are selected in the order of their priorities and scheduled on best processor, which minimises the task's finish time.

Task Prioritizing[1]: Each task is set with a priority using an upward rank value, $rank_u$, based on mean computation and mean communication costs. The sorted task list is prepared in the decreasing order of $rank_u$ providing a topological order of tasks and preserving the precedence constraints.

Processor Selection[1]: In this phase tasks are selected in the order of their priority and the selected task is assigned to the best processor, which minimises its earliest finish time using an insertion based approach.

The time complexity of HEFT algorithm with e number of edges and q number of processors is $O(e \times q)$.

IV. STANDARD DEVIATION OF PROBABILITY OF PROCESSOR AVAILABILITY (SDPA) BASED TASK SCHEDULING ALGORITHM

ALGORITHM :
Begin SDPA
// N represents set of Nodes
// P represents set of Processors
// $pap(n_i,p_j)$ is probability of availability of processor p_j for given task n_i
// SD is standard deviation of probability of processor availability on available processors
For all n_i **in** N
Compute $rank_u(n_i)$
End For
ReadyTaskList ← *Start Node*
While *ReadyTaskList* ***is*** *NOT NULL* **do**
n_i ← Node in the *ReadyTaskList* with the maximum *ranku*
For all p_j **in** P

$$EST(n_i, p_j) = \max(T_{available[j]}, \max_{n_m \in pred(n_i)}(EFT(n_m, p_k) + c_{m,i}))$$

$$EFT(n_i, p_j) = w_{i,j} + EST(n_i, p_j)$$

End For
For all *pj* **in** P
ESD= $Mpap(n_i)$ - $SD(n_i)$ // Mpap is Mean of $pap(n_i)$
If ((EFT(n_i,p_j) ~= max(EFT(n_i))) AND ($pap(n_i,p_j)$ < ESD))

$$EFT_p(n_i, p_j) = EFT(n_i, p_j) + (EFT(n_i, p_j) * SD(ni))$$

Else

$$EFT_p(n_i, p_j) = EFT(n_i, p_j)$$

End If
End For
Map node n_i on processor p_j which provides its least EFT_p
Update *T_Available[pj]* and *ReadyTaskList*
End While
End SDPA

Figure 2 The SDPA Algorithm

The SDPA algorithm is an application scheduling algorithm for static scheduling that works in two stages that considers resource availability [7]. During first stage jobs are arranged based on the upward ranks and during the second stage jobs are allotted to the processor based on the standard

deviation of probability of processor availability. In this stage processor is selected by checking the probability of processor availability with the Standard deviation and if it is outside the Standard Deviation range EFT is adjusted with the Standard Deviation factor to avoid delays in execution.

The time complexity of SDPA algorithm with e number of edges and q number of processors is O(e x q).

V. EXPERIMENTAL RESULTS:

We present the comparative result of our algorithm SDPA and HEFT in this section. For testing the algorithms we used application graphs generated randomly. Following sections present algorithm used for Random DAG generation and metrics used for performance evaluation.

a. Comparison Metrics[1]

The SDPA and HEFT algorithms are compared based on the following metrics:

Makespan: Makespan is the main performance measure of a scheduling algorithm or Schedule Length of its output schedule.

Schedule Length Ratio: The SLR of an algorithm is defined as follows:

$$SLR = \frac{Makespan}{\sum_{n_I \in CP_{\min}} \min_{p_J \in Q} \{w_{ij}\}}$$

The best scheduling algorithm is the one that gives the lowest SLR of a graph. Average SLR values are considered for performance evaluation of task graphs.

Speedup: Speedup of scheduling algorithm is computed by dividing sequential execution time by the parallel execution time (makespan). The sequential execution time is calculated by assigning all sub-tasks to a single processor which minimizes the cumulative computation costs.

b. Random Graph Generator: A random graph generator algorithm given in [6], generates random directed acyclic graphs, in which it takes number of nodes as input and generates a weighted directed acyclic graph, where number of edges are generated randomly, based on number of nodes.

Our simulated framework first executes Random Directed A-cyclic Graph Generator Program followed by execution of SDPA and HEFT scheduling algorithms.

It takes number of nodes, number of processors, randomly generated computation cost matrix, and communication cost matrix and probability of processor availability matrix as input to generate schedule outputs. Performance metrics are computed based on schedules.

c. Results: The performance of algorithms is compared by generating set of graphs of different sizes with 10, 20,30,40,50 nodes with random values. Average values of different DAGs are considered for comparison and the graphs are shown in figure 3, 4 and 5.

Table 2 Comparison of Average Makespan

No. Of Nodes	Average Makespan	
	HEFT	SDPA
10	96.4	105.5
20	188.90	197.81
30	283.27	297.54
40	393.00	407.18
50	439.45	456.81

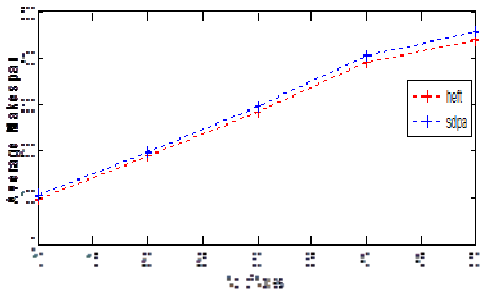

Figure 3 Average Makespan of Randomly Generated DAGs

Table 3 Comparison of Average SLR

No. Of Nodes	Average SLR	
	HEFT	SDPA
10	0.66	0.72
20	0.62	0.66
30	0.58	0.61
40	0.58	0.61
50	0.57	0.59

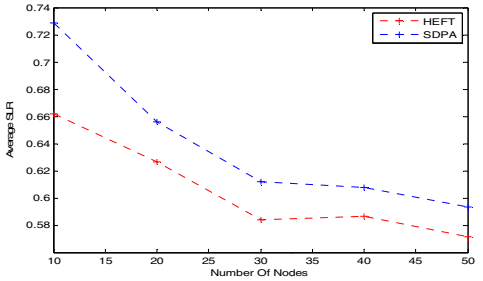

Figure 4 Average SLR of Randomly Generated DAGs

A static scheduling algorithm HEFT, for bounded number of processors, in which tasks are prioritized and scheduled to a processor based on Earliest

Finish Time. HEFT assumes that all processors are 100% available for allocation. In real life situation if a resource is not available due to overloading of jobs or other reasons, delays may occur and it may not achieve the required makespan.

Table 4 Comparison of Average Speedup

No. Of Nodes	Average Speedup	
	HEFT	SDPA
10	0.94	0.84
20	0.97	0.93
30	0.99	0.95
40	0.91	0.88
50	1.05	1.01

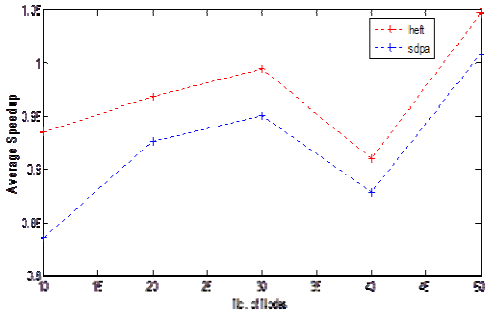

Figure 5 Average Speedup of Randomly Generated DAGs

A static scheduling algorithm SDPA for bounded number of processors, in which tasks are prioritized and scheduled to a processor not only based on Earliest Finish time but also considers processor availability based on probability of availability of processors, so that delays can be avoided. SDPA algorithm produces makespan, SLR and Speedup comparable with HEFT by considering the probability of availability of processors where as in HEFT probability of availability of processors is not considered.

VI. CONCLUSIONS

We present a new heuristic called SDPA Task Scheduling algorithm for scheduling task graphs with precedence constraints and scheduling tasks to a processor not only based on Earliest Finish time but also considering the probability of availability of processors. Our algorithm is based on Standard Deviation of probability of availability of processors. We computed Average makespan, Average SLR and Average SpeedUp of the generated Random Graphs of different sizes and compared the results. 10% of the Random Graphs generated produced better results than HEFT and 40% of the Graphs generated produced makespan, SLR and Speed up comparable with HEFT.

Makespan, SLR and Speedup produced by SDPA algorithm is comparable with HEFT by considering the probability of availability of processors. This helps in avoiding delays that may occur due to non-availability of processors.

SDPA algorithm considers the resource availability factor where as in HEFT it is not considered. Our SDPA Algorithm is reliable compared to HEFT for static scheduling.

REFERENCES

[1] Haluk Topcuoglu, Salim Hariri, Min-You Wu "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing" 1045-9219/02/$17.00 © 2002 IEEE

[2] Xu Wang, Beizhan Wang, Jing Huang, "Cloud Computing and its key techniques", 978-1-4244-8728-8 ©2011 IEEE

[3] LI Wenhao "A Community Cloud Oriented Workflow System Framework and its Scheduling Strategy" 2010 IEEE

[4] A Marinos and G Briscoe. "Community Cloud Computing", First International Conference on Cloud Computing. ACM Press 2009.

[5] Gerasoulis, A. and Yang, T "A comparison of clustering heuristics for scheduling directed graphs multiprocessors" Journal of Parallel and Distributed Computing, 16(4):276-291.

[6] Yinfeng Wang, Zhijing Liu, Wei Yan, "Algorithms for Random Adjacency Matrixes Generation Used for Scheduling Algorithms Test", International Conference on Machine Vision and Human-Machine Interface (MVHI), 2010

[7] Brent Rood and Michael J. Lewis," Multi-State Grid Resource Availability Characterization", IEEE 8th Grid Computing Conference, 2007

[8] Nitish Chopra, Sarbjeet Singh, "Survey on scheduling in hybrid clouds", International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2014

[9] Sanjeev Baskiyar, Prashanth C. SaiRanga: Scheduling Directed A-cyclic Task Graphs on Heterogeneous Network of Workstations to Minimize Schedule Length. ICPP Workshops 2003: 97-103

[10] Sanjeev Baskiyar, Prashanth C. SaiRanga: Scheduling DAGs on Heterogeneous Network of Workstations to Minimize Finish Time. I. J. Comput. Appl. 13(4): 168-175 (2006)