Deadline Aware Prioritized Task Scheduling Algorithm in Cloud Computing

R. Barani, S. Suguna

Abstract— Cloud Computing is a computing Paradigm in which services are provided by service providers on pay-per-use. Task Scheduling is the challenging issue in cloud computing. Task scheduling refers to allocating tasks to available resources to achieve better performance of the system. Here we have proposed a Heuristics algorithm to schedule tasks in given resources which satisfies the QoS of system taking Priority and Deadline of tasks as parameters. Our algorithm is compared with existing algorithms like EDF and TLD algorithms. Our algorithm provides better makespan, increases throughput and utilize resources well compared to existing algorithms.

Keywords— Task Scheduling - TLD Algorithm –EDF Algorithm –Proposed Algorithm –Comparative Analysis

I. INTRODUCTION

Cloud computing is the computing service delivered to the user by providers. Cloud providers provide different types of service like Software as a Service(SaaS) where application software and database access are provided. Platform as a Service(PaaS) where computing platform are provided and Infrastructure as a Service(IaaS) where Infrastructure is provided to user which includes processing power and storage of virtual machines.

These services are provided as a pay-per-use service. Task Scheduling refers to allocating tasks to available Virtual machines. A good task scheduling improves the performance of the system and help to achieve good Quality of Service(QoS). The objective of task scheduling is to minimize the makespan and maximize the Resource Utilization. Many Task scheduling algorithms were proposed but we need more improvement to make cloud system a better one.

In this paper we propose a Heuristics algorithm called Deadline Aware Prioritized Task Scheduling Algorithm(DPTS) to schedule tasks in available Virtual machines. In Section 2, we provide the literature survey of available algorithms for task scheduling. In section 3, our proposed algorithm was explained. In Section 4, the comparative analysis of proposed algorithms with the existing algorithms are made and the results are tabulated. Finally we conclude about the merits and demerits of proposed algorithm.

II. LITERATURE SURVEY

Omer Khalid et al[1] designed a deadline-aware algorithm which schedules a virtual machine for minimizing overhead and it is used to minimize job-deadlines when it is given to resources. Patel et al[2] proposed an improved priority based scheduling using iterative methods. This algorithm has better makespan and consistency than other algorithms. NeelimaPriyanka et al[3] proposed an Optimal priority based EDF scheduling algorithm for allocating resources using task priorities, deadlines and its processing times. It shows better performance of turnaround time, waiting time and deadline violation when compared to FCFS, SJF and priority based EDF scheduling.

Deepta Saxena et al[4] proposed a dynamic fair priority optimization task scheduling algorithm in which they classified and grouped all tasks based on their deadline and cost constraints. Here different priority queues (high, mid, low) are implemented in round robin fashion. This algorithm provides better efficiency and fairness at system level. NeelimaVarma et al[5] presented Round robin based Prioritized Earliest Deadline first scheduling model based on time quantum, priorities and deadlines. It is compared with traditional algorithms like FCFS and SJF which reduces the turnaround time, waiting time and average deadline violation.

Pankaj Kumar Chauhan et al[6] proposed a model in which they have assigned a rank for each task and scheduled using min-min strategy. This algorithm is advantageous in terms of task scheduling based on task length and deadline. K. Raja et al[7] proposed an improved credit based algorithm which uses the parameters like user priority, task length and deadline. This algorithm reduces makespan of tasks. Antony Thomas et al[8] proposed a Credit based scheduling algorithm which is based on cloudlet length and priority. In this makespan is decreased and shows improvement in resource utilization.

III.EXISTING METHODOLOGIES

Garima Gupta et al[9] present a priority based earliest deadline first algorithm with task migration. They allocate priority to task based on their deadline and allocate resources to them. The tasks are migrated when they missed their deadline. This improves overall response time and gives better performance. In this algorithm the tasks are allocated only on deadline value. Waiting queue is introduced for pre-empted tasks and processed later based on their priority. This increases the overhead of the system.

Arnab Wadhonkaret al[10] proposed an algorithm which allocates tasks based on their length and deadline. This algorithm has two scheduling criteria. First is task length and other is deadline of task. It is compared with algorithm which is based on task length and algorithm which is based on deadline.
The presented algorithm shows reduction in makespan and average waiting time than compared algorithms. We have proposed a task scheduling algorithm which takes two parameters Deadline and Task length to improve the performance of the system.

3.1 Proposed Methodology

In our algorithm we have considered two constraints. One is task length and other is Deadline. First we are taking the mean for task length and finding the difference between mean value and task length of each task. In the same way, we are finding the mean for deadline value and finding the difference between the mean value and deadline value of each task. Then we are adding the two difference values of task and deadline. These values are arranged in ascending order. We select the first value(low value) from list and assign it to minimum completion time resource if it is available. Then the last value(high value) is selected and assigned to next minimum completion time resource available. This process is repeated until the list is empty.

As the list is arranged based on difference value of mean and we are choosing the low value and high value in the list alternatively, starvation of tasks are reduced.

The flowchart for this algorithm is shown in Figure 1

![Flowchart of Proposed Algorithm](image)

We calculate the Average throughput of the system as

\[
\text{Average Throughput} = \frac{\text{No of tasks}}{\text{makespan}}
\]

and Average Utilization of resources as

\[
\text{Average Resource Utilization} = \frac{\sum \text{time taken by resource to finish all tasks}}{\text{makespan} \times n} \times \frac{1}{n} = \frac{\text{Number of resources}}{n}
\]

The pseudocode of the proposed algorithm (DPTSA) is shown in Algorithm 1.

**Algorithm 1: Pseudocode of Proposed Algorithm**

<table>
<thead>
<tr>
<th>Input:</th>
<th>’n’ number of job request Ti with its priority Pi and deadline DLi.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>Mapping of Tasks in resources considering both Task length as priority and deadline of job.</td>
</tr>
</tbody>
</table>

1. Get each task length (TLi)
2. Get Deadline of each tasks (TDi)
3. Calculate the mean for length of task as
   \[ MT = \frac{\sum TLi}{1 \leq i \leq n} \]
4. Calculate mean for deadline of tasks as
   \[ MD = \frac{\sum TDi}{1 \leq i \leq n} \]
5. Compute difference between task length and mean task length values
   \[ DiffLeni = |MT - TLi| \]
6. Compute difference between task deadline and mean deadline values
   \[ Diffdi = |MD - TDi| \]
7. Add both task length difference value and deadline difference values
   \[ Pri = DiffLeni + Diffdi \]
8. Sort Pri in ascending order
9. Repeat the following steps until no tasks is available in Pri.
10. Check if Minimum completion time resource is available.
11. If yes then
    11.1 Get the task from the List(pri=low value).
    11.2 Assign task Ti to Resource Rj
    11.3 Compute completion time of task as
        \[ Cij = ETij + Wti \]
    11.4 Remove the task Ti from task list.
    11.5 Update Rj & Cij
12. Next choose the task Tn from list(pri=high value)
13. Repeat the steps 11.2 thro 11.5.
14. Alternatively choose the minimum value from Pri and then maximum value from Pri for next iteration.
15. If Minimum completion time resource not available then
16. Choose next available minimum completion time resource to allocate the tasks.
17. If resource not available, wait in queue until resource becomes free.

IV. RESULTS AND DISCUSSION

The algorithm is simulated in simulation environment CloudSim 3.0.3. The simulation is done under following conditions.

**Table 1: Basic Configuration**

<table>
<thead>
<tr>
<th>Number of Datacenters</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Cloudlets</td>
<td>10</td>
</tr>
<tr>
<td>Number of Brokers</td>
<td>1</td>
</tr>
<tr>
<td>Number of Hosts under Datacenter</td>
<td>2</td>
</tr>
</tbody>
</table>

Each datacenter has several hosts. Host configuration is shown as
Table 2: Host configuration

<table>
<thead>
<tr>
<th>Resource</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM(MB)</td>
<td>16384</td>
</tr>
<tr>
<td>MIPS(Lines of Codes)</td>
<td>1000</td>
</tr>
<tr>
<td>Storage(MB)</td>
<td>100000</td>
</tr>
<tr>
<td>Bandwidth(MB/sec)</td>
<td>10000</td>
</tr>
<tr>
<td>Number of Virtual Machines</td>
<td>3</td>
</tr>
</tbody>
</table>

Host consists of several virtual machines. Each virtual machine has its own configuration. Here we consider the same configuration is applied for each VMs varying its processing speed. The configuration of virtual machine are shown as

Table 3: Virtual Machine Configuration

<table>
<thead>
<tr>
<th>Resource</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number cores</td>
<td>2</td>
</tr>
<tr>
<td>MIPS (Lines of Codes)</td>
<td>1000</td>
</tr>
<tr>
<td>Size(MB)</td>
<td>10000</td>
</tr>
<tr>
<td>RAM(MB)</td>
<td>512</td>
</tr>
<tr>
<td>Bandwidth(MB/sec)</td>
<td>1000</td>
</tr>
</tbody>
</table>

We are taking 7 tasks whose task length are (2400,8200,5600,4200,1600,3200,2500) and their deadline (102,151,194,118,57,69,72). We are allocating these 7 tasks in three resources whose processing speed are (300,100,50). The Makespan of TLD algorithm is 120, EDF algorithm is 114 whereas proposed algorithm is 82. This shows the makespan is reduced in proposed algorithm. The comparison table of algorithms for executing 20 tasks in 3 resources is as below

Table 4: Comparison table of algorithms

<table>
<thead>
<tr>
<th>Metrics / Algorithms</th>
<th>TLD</th>
<th>EDF</th>
<th>Proposed (DPTSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan</td>
<td>42.6</td>
<td>34</td>
<td>32</td>
</tr>
<tr>
<td>Throughput</td>
<td>0.47</td>
<td>0.59</td>
<td>0.63</td>
</tr>
<tr>
<td>Resource Utilization</td>
<td>77%</td>
<td>91%</td>
<td>94%</td>
</tr>
</tbody>
</table>

From the above table it is clear that the makespan is reduced and throughput is increased. From the provider point the resource utilization is high compared to other algorithms. The comparison chart for makespan when executing tasks in 3 resources are shown in figure 2.

Figure 2: Comparison chart of makespan

This shows that the makespan is reduced in proposed algorithm. The comparison chart for resource utilization is shown in figure 3.

Figure 3: Comparison chart of Resource Utilization

The above chart shows that the resource utilization is high in proposed algorithm. The comparison chart of throughput is shown in figure 4 as follows

Figure 4: Comparison chart of Throughput

The chart shows that the throughput is increased in proposed algorithm when compared to existing algorithm.

V. CONCLUSION

The Efficient Scheduling of Tasks to available resources will improve performance of the system. Users desire to reduce makespan of executing tasks and provider desire to increase Resource utilization. In our proposed algorithm we have increased the resource utilization on the average of 15.46% than compared algorithms and makespan is reduced on an average of 23.04%. Throughput also increased on an average of 19.04% when compared to other existing algorithms like TLD and Prioritized EDT algorithms. Thus the proposed algorithm improves the performance by an average of 19.18% than existing algorithms. In future we consider the workflow tasks and work to reduce the time latency.

REFERENCES


