Resource Based Load Balanced Min Min Algorithm (RBLMM) for static Meta task Scheduling in Cloud

BJ.Hubert Shanthan\textsuperscript{1}, Dr.L.Arockiam\textsuperscript{2}

Abstract:
Cloud Computing is a new technology that facilitates the IT users for providing large variety of resources. Load Balancing is one of the major issues in cloud. To tackle this problem there is a need for proper scheduling mechanism to match user requests with the available resources. In Load Balanced Min Min algorithm, the make span is high and average resource utilization rate is low. The RBLMM algorithm is proposed to overcome the problem and it outperforms in comparison with the existing Min Min and Load Balanced Min Min algorithms in terms of make span and resource utilization rate. This algorithm consists of two phases. In the first phase, the make span value is calculated using the mechanism of the Min Min algorithm. The second phase uses them as threshold value and it reschedules the tasks with maximum execution time and minimum completion time. Hence, the proposed algorithm balances load equal to all tasks and its resources.

Keywords— Min Min, Load Balance, Load Balanced Min Min, RBLMM.

I. INTRODUCTION
In today’s modern era, cloud is an emerging technology in the field of ubiquitous computing. Cloud computing provides unlimited virtual service to its customers on demand. Cloud Technology enables the user to store, compute and process the data in the internet. Most of IT Companies deploy their software applications in the cloud.

There are three main types of service models provided by cloud, namely IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service). IaaS is mainly used by the IT administrators for data storage in Virtualized servers. PaaS mainly comprises of tools which are more viable for programmers to develop their own applications. SaaS is designed for the normal internet users to utilize application only [1].

Cloud is further classified into three major types, namely public cloud, private cloud and hybrid cloud. Public cloud is a cloud that can be accessed by anybody from anywhere on demand. Private cloud belongs to particular organization and it is more secured than any other cloud. Hybrid cloud is combination of private and public cloud [2][3].

Cloud technology is applicable in several fields to analyse and give solutions for real world problems. Cloud is created as a business application model to satisfy the customer needs on demand. Therefore, cloud consumer requests more number of resources within a stipulated amount of time. So, there is a chance of performance degradation in the cloud service providers.

Scheduling is the process of assigning resources from the server to the user/client [4][5]. Scheduling plays a vital role in the cloud. A Scheduling mechanism consists of three phases. They are searching, matching and allocating the resources to its corresponding requests.

Scheduling is broadly classified into two types, namely dependent scheduling and independent scheduling. Independent scheduling is based on ETC (Expected Execution Time) matrix. ETC [6] table comprises of tasks with execution time of the resource. In dependent scheduling, the tasks execution is done based on preceding tasks (i.e. the tasks are dependent on each other or collaborative task scheduling). Dependent and independent scheduling is classified into sub types such as static and dynamic. In Static Scheduling, the tasks execution time is known to the users and it has low fault tolerance rate [7]. Dynamic Scheduling or Online algorithm is more suitable for real time applications and its run time is not known to users [8]. Make span and Load balancing are two major factors affecting scheduling algorithm [9]. Makespan is the total completion time of the tasks for the required resources. Load balancing is used to manage overloaded resource to the tasks.
Completion time for every task is calculated by the sum of waiting time and execution time. The waiting time is determined by the arrival rate/time of the requests of the tasks by the user.

This paper proposes a novel Resource Based load balanced Min Min algorithm to balance the load and to reduce makespan with proper resource utilization. This algorithm outperforms the existing load balanced min-min algorithms in terms of makespan and resource utilization. Section II narrates related works of the algorithm. Section III describes proposed algorithm with pseudo code. Section IV comprises of results and discussion. Finally, section V provides conclusions with future directions.

II. RELATED WORKS

Kamala, et al. [10] presented a heuristics algorithm to reduce makespan and improve utilization of resources. The algorithm combined the benefits of minimum completion time and minimum execution time of the resources. This algorithm is comprised of two phases: first phase selects the tasks with minimum execution time and the second phase calculates the tasks with the minimum completion time and then assigns the tasks to its resources. This algorithm gave priority for the smaller tasks and so there was rapid increase of waiting time for the larger tasks. Rajasekar, et al. [11] proposed a mechanism to improve makespan for maximum utilization of resources for larger tasks. This algorithm outperformed Min algorithm in terms of makespan. This algorithm had two phases: the first phase was similar to Min Min and in second phase, the tasks with maximum completion time were selected and assigned to their resources. Priority was given to larger tasks as it reduced average waiting time of the resources.

Kokilvani, et al. [12] proposed an efficient load balanced Min Min scheduling algorithm to reduce makespan and improve the resource utilization rate. Min Min algorithm was used in the first stage of this algorithm and the makespan was also calculated in this stage. Makespan value was used as threshold value to reschedule the tasks and the tasks with heavy load of resources were reassigned to the tasks with light load of resources. This algorithm outperformed the standard Min Min algorithm and improved the average resource utilization rate. Patel, et al. [13] suggested a strategy to enhance load balanced Min Min scheduling algorithm to minimize the makespan and increased the average resource utilization rate. This algorithm was more suitable for scheduling Meta tasks. Makespan value was calculated by Min Min algorithm. The tasks were sorted in non-decreasing order with execution time and value was checked with makespan. The sorted value with maximum execution time was reassigned to the tasks with minimum completion time of the resources and load was balanced by selecting minimum value from the completion time of the assigned resources of the tasks.

M. Maheswaran, et al. [14] proposed a suffrage based Meta task scheduling algorithm to reduce the makespan and increased the average resource utilization rate of the servers. The suffrage value was calculated by making difference between first minimum execution time and second minimum completion time. The tasks with high suffrage value were assigned to the resources and also considered minimum completion time of the resources. In this, algorithm maximum suffered tasks were given priority and least suffered tasks were given less importance. This algorithm gave better makespan results than Min Min and Max Min algorithm.

Afab[15] proposed an improved Min Minalgorithm to reduce makespan in the Meta task scheduling in grid computing. This algorithm calculated the arithmetic mean of the minimum execution time value. The mean value was used as a threshold value and tasks were allocated to the resources. This algorithm outperformed standard Min Min and Max Min algorithm. Load balanced Opportunistic approach [16] was easy and simple to implement than any other mechanism. This approach had poor makespan. It allocated the available resources to the tasks and it made the machine busy all the time. This algorithm did not consider the execution and completion time for the resources.

Thomas, et al. [17] proposed a static scheduling algorithm by considering task length and user priority as credit value. The credit value acted as parameter and allocated the task based on priority and duration of the task which was requested by the user. Parsa, et al. [18] proposed a unique resource aware scheduling mechanism for the Meta tasks. The mechanism worked on the available resources for mapping of the tasks. This algorithm used Min Min strategy for the odd and even number of resources. The Max Min mechanism was used for mapping the tasks to its resources. Sharma, et al. [19] suggested a heuristic based task aware scheduling mechanism for cloud. This algorithm checked total number of tasks that were available for mapping its resources. The odd number of tasks used Min Min strategy to map its resources. Max Min mechanism was used for even number of tasks to map its resources.
III. RESOURCE BASED LOAD BALANCED MIN MIN ALGORITHM

The Resource based Load balanced Min Min algorithm is designed to reduce make span and to balance the virtual machines’ load. This algorithm is more viable for static independent meta task scheduling mechanism in cloud computing.

The proposed algorithm comprises of two phases. In the first phase, Min-Min strategy is used to assign task to its resources and the make span value is calculated from the assigned resources. In the second phase, the make span value is considered as a threshold value and tasks are rescheduled to balance load in the virtual machines.

In the proposed RBLMM algorithm, the tasks are sorted in ascending order based on minimum completion time. This algorithm selects tasks that have maximum completion time with minimum execution time from the sorted list. The selected task must have lesser execution time than the make span value. In the sorted list, if there is same number of resources assigned to a particular task the priority will be given to tasks with minimum completion time. The allocated tasks are removed from the sorted list and ready time is updated with

Resource Based Load Balanced Min Min (RBLMM) Algorithm

Input : Meta tasks MT_a, Task Length M_i, Resource Speed MIPS_j, Virtual Machines Resources RS_i

Output : Mapped Schedule of Virtual resources RS, with tasks MT_a

Begin
Initialization
MT_a[MT_1, T_2, ..., T_n] // Meta tasks //
RS_i[R_1, R_2, ..., R_m] // Resources//
R_j[0] // Ready time //
Repeat
Until MT_a ≠ Ø // Meta tasks not empty //
for all tasks T_a,∈MT_a
for all resource R_a,∈RS_i
Compute ETC_ij = M_i / MIPS_j
Sort the tasks in ascending order based on completion time
Select the first task and schedule the task that as minimum execution time and assign task T_a, to the Resource R_a
Remove the assigned tasks to the resources by using formula
MT_a = MT_a - T_a
Update the ready time of all resources R_j
Compute the completion time of the tasks by using formula
CT_a = ETC_ij + R_j
for all resource R_a,∈RS_i
Compute MakeSpan using the formula
Mksp_i = max (CT_a(RS_i))
End for
if more than one task has same MET then
select task T_a with minimum value and assign to min (CT_a)
if (min (CT_a) <= Mksp_i) then
Reschedule the tasks T_a to the resource produces it
Update ready time R_j
End if
End if
End for
End for
End
// MET Maximum Execution Time //
// MCT Minimum Completion Time //

Fig 2: Depicts the Working Mechanism of RBLMM flowchart.

IV. RESULTS AND DISCUSSION

The Algorithm is evaluated by using sample benchmark data described in the table 1.0. The table 1.0 contains tasks with Million instructions Per second (MI) and data volume in Mega bytes (Mb).
The ETC matrix values are presented in table 3 which describe the expected execution time for the tasks. The ETC value is calculated by the following formula: 

\[ \text{ETC} = \frac{(M_1 \text{ Speed})}{\text{MIPS}}. \]

The figure 2 clearly illustrates the comparative analysis of existing scheduling algorithms with the proposed RBLMM scheduling Algorithm. The makespan value for the min min algorithm is 10 secs, and LBMM is 8 secs. The maximum completion time for the proposed RBLMM is 7 secs. This algorithm also balances load equal between two virtual machines.

### Table 1: Tasks Specification

<table>
<thead>
<tr>
<th>Task (Jobs)</th>
<th>Instruction volume (MI)</th>
<th>Data Volume (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>702</td>
<td>129</td>
</tr>
<tr>
<td>T₂</td>
<td>1202</td>
<td>270</td>
</tr>
<tr>
<td>T₃</td>
<td>700</td>
<td>129</td>
</tr>
<tr>
<td>T₄</td>
<td>670</td>
<td>89</td>
</tr>
</tbody>
</table>

### Table 2: Resources Specification

<table>
<thead>
<tr>
<th>Resource</th>
<th>Processing Speed (MIPS)</th>
<th>Bandwidth (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>100</td>
<td>78</td>
</tr>
<tr>
<td>R₂</td>
<td>320</td>
<td>65</td>
</tr>
</tbody>
</table>

### Table 3: ETC Table

<table>
<thead>
<tr>
<th>Task</th>
<th>R₁</th>
<th>R₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>T₂</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>T₃</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>T₄</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

### REFERENCES


