Modified Spline Interpolation Method For Resource Allocation In Heterogeneous Cloud Environment

S. Vimala, V. Nisha

Abstract: Distributed resource allocation is a complex problem in the emerging Cloud Computing strategy, where the fundamental criteria that benefit both the cloud users and the cloud providers are to be identified. The resources are allocated to the tasks depending on the demand in terms of bandwidth and memory by the Users. In this research work, the Spline Linear Interpolation method is used for the resource allocation problem to find an optimum solution. It is keen to provide user-requested bandwidth and memory with less completion time. The proposed method is evaluated and tested in the CloudSim environment. The test results show that the Spline Interpolation method performs efficiently in terms of migration cost, Completion Time, Waiting Time, Turn Around Time and workload balance compared to Modified Round Robin Algorithm.

Index Terms: Cloud computing, Resource allocation, Virtual Machine, Spline interpolation, CloudSim

1 INTRODUCTION
Cloud computing is an infrastructural model, which provides enormous computational power to internet service providers. Resource allocation is a challenging process to allocate resources based on the users’ requirements in an effective manner. The need for reliable, robust, and high-performance scheduling algorithm increases day-to-day. New methods, resource models and standards are being developed continuously. Scalable on-demand Resource provisioning is one of the basic features of Cloud computing technology. The Service Level Agreement (SLA) is a bond between Cloud User and Cloud Provider stating the agreement terms that include Obligations of the user, Quality of Service (QoS) that can be provided by the Cloud Provider and penalties for agreement violation [1]. Resource Provisioning can be done with SLA in the cloud. In the Cloud Provider’s point of view, SLA violation has to be prevented to eliminate penalties and to utilize the resources efficiently. From the Cloud users’ point of view, SLA helps to enjoy the facilities for the cost they are paying for. Thus, scheduling of resources by considering multiple SLA parameters and efficient allocation of resources is necessary [1]. However, the above approaches are limited to simple workflows and single-task applications. Scheduling and deploying the service requests for multiple SLA parameters such as the amount of CPU required, network bandwidth, memory, and storage are still open research challenges [3] [4]. The problem occurs during resource allocation can be solved by choosing a suitable algorithm for resource allocation [5]. In the proposed work, bandwidth and memory are taken into account for resource allocation. Spline Linear Interpolation [6] method is used to find the best matching resources available from the pool of resources. After choosing the VM, the user’s job is kept in the wait queue. Generally, each VM has a separate wait queue. If one of the wait queues is loaded with more user jobs, the VM that completes all the jobs will fetch the jobs from the heavily loaded wait queue for execution after checking the required bandwidth and memory. This algorithm is efficient in allocating user-requested bandwidth and memory with less completion time. The rest of this paper is organized as follows; section 2 presents the existing work, section 3 describes the proposed method, in section 4, the results are discussed, and Section 5 gives the conclusion.

2 RELATED WORK
Author [7] proposes an algorithm to maximize revenue through SLA aware allocation. Time Service factor-based pricing models are used to allocate resources by considering the following Quality of Service (QoS) parameters such as service rates and available resources. Author [8] proposes a two-fold mechanism. In the first phase, jobs are arranged in ascending order according to the priority. Priority is calculated by following QoS parameters such as user level, expected priority, length, waiting time. In the second phase, compute the expected execution time. Then the resources are allocated to the tasks in ascending order of execution time. Author [9] proposes a Multi-objective scheduling method based on Ant Colony Optimization (MOSACO) to optimize the resources in an hybrid cloud by considering the deadline and cost constraint QoS parameter. A component-based model [10] implements the resource estimate using a weighted least square estimation method. The least-square method regression technique is used for predicting resource demands [11]. An integer linear programming approach [12] and Haizea greedy algorithm have been used for optimizing the problem. The number of VMs to be assigned to Processing Element must be determined depending on the capacity of Processing Element. This leads to an efficient utilization of resources. In paper [13] Modified Round Robin algorithm is used to evaluate the solution. It satisfies customer demands by reducing the waiting time.
3 PROPOSED METHOD

3.1 Modified Spline Interpolation Based Resource Allocation Method
The general formula for polynomial equations can be written as:
\[ f(x) = a_0 + a_2 x + a_3 x^2 + \ldots + a_n x^n - 1 \]  
(1)

A single polynomial cannot fit a large number of points, it will produce an oscillating performance in the interpolation. Spline which is in the first order will find the straight lines between each and every points. The lines obtained looks like a broken line. The general formula for spline is:
\[ S_k(x) = y_k + d_k(x-x_k) \]  
(2)

where \( d_k = (y_{k+1} - y_k) / (x_{k+1} - x_k) \)

x value should be between \( x_k \) and \( x_{k+1} \), i.e., \( x_k \leq x \leq x_{k+1} \)

The linear spline function can be written in the form of a set of equations.
\[ S(x) = y_0 + d_0(x-x_0) \quad \text{for} \quad x \in [x_0, x_1] \]
\[ y_1 + d_1(x-x_1) \quad \text{for} \quad x \in [x_1, x_2] \]
\[ y_2 + d_2(x-x_2) \quad \text{for} \quad x \in [x_2, x_3] \]
\[ y_k + d_k(x-x_k) \quad \text{for} \quad x \in [x_k, x_{k+1}] \]
\[ y_{k+1} + d_{k+1}(x-x_{k+1}) \quad \text{for} \quad x \in [x_{k+1}, x_{k+2}] \]

(3)

It is assumed that \( x_0 < x_1 < \ldots < x_k < x_{k+1} < x_{k+2} \).

The Spline method can able to find the optimum schedule using a first-order polynomial equation (line). The proposed scheduling problem consider the parameters such as Bandwidth and Memory. Consider \( n \) as the number of resources, \( \{x_0, x_1, x_2, \ldots, x_n\} \) as resource parameters, \( \{y_0, y_1, y_2, \ldots, y_n\} \) as resource index number, \( x \) as the user requested parameter (i.e. user required bandwidth and memory). Arrange the resource parameters in ascending order. Draw the straight perpendicular line using \( x \) value. Find the vertices where \( x \) hits. Calculate the vertices \( (x_k, y_k) \) and \( (x_{k+1}, y_{k+1}) \) which satisfy the limit \( x_k < x < x_{k+1} \). Find the absolute value of \( (x_k, y_k) \) and \( (x_{k+1}, y_{k+1}) \), if \( x_k, y_k \) is greater than \( x_{k+1}, y_{k+1} \) store the value of \( y_{k+1} \) in first preference array and \( y_k \) in second preference array. Otherwise, store the value of \( y_k \) in first preference array and \( y_{k+1} \) in the second preference array. Similarly, find suitable resources for all the user tasks and select two suitable resources and store the most suitable resources in the first preference array and store the next suitable resource in second preference array. Compare the first preference of memory with the first preference of bandwidth, if match found, allocate the particular Virtual Machine (VM) to that task. Then compare the second preference of memory with the second preference of bandwidth, if match found, allocate the particular Virtual Machine (VM) to that task. Then Compare first preference of memory with the second preference of bandwidth, if match found, allocate the particular Virtual Machine (VM) to that task. Then compare the second preference of memory with the first preference of bandwidth, if match found, allocate the particular Virtual Machine (VM) to that task. Unscheduled tasks are stored in noMatch array. VM starts its execution after the tasks are allocated. Whenever a VM completes all of the assigned tasks, then it fetches the jobs from the noMatch array and checks user requirements (i.e.) Bandwidth and Memory with the resource availability. If match found, allocate the particular Virtual Machine (VM) to that task. If a match is not found, then it fetches the job from heavily loaded VM Queue and checks user requirement (i.e.) Bandwidth and Memory with the resource availability.

3.2 Algorithm

Step 1: Initialize the parameters of bandwidth and memory of the resources.
Step 2: Fetch the Resource parameter (i.e. Bandwidth, Memory), arrange the resources in ascending order. Initialize the resource parameter in \( \{x_0, x_1, x_2, \ldots, x_n\} \), the resource numbers in \( \{y_0, y_1, y_2, \ldots, y_n\} \), \( j \) is the iteration number.

Step 3: Draw the straight line segments using \( (x_0, x_1, x_2, \ldots, x_n) \) and \( (y_0, y_1, y_2, \ldots, y_n) \). (Where \( n \) is the number of resources)

Step 4: Initialize the user request in \( x \). Find the value of \( y \) (Suitable Resource). Find the line where \( x \) hits and it should satisfy \( x_k < x < x_{k+1} \).

\[
\text{if } \text{abs}(x-x_k) > \text{abs}(x-x_{k+1}) \text{ then}
\]
\[
\text{FirstPref} = y_{k+1}
\]
\[
\text{SecondPref} = y_k
\]
\[
\text{if } \text{abs}(x-x_k) < \text{abs}(x-x_{k+1}) \text{ then}
\]
\[
\text{FirstPref} = y_k
\]
\[
\text{SecondPref} = y_{k+1}
\]

Where,
\[
i \rightarrow \text{Cloudlet ID}
\]
\[
j \rightarrow \text{Parameter(i.e. Memory-1,Bandwidth-2)}
\]
\[
k \rightarrow \text{Resource Index Number}
\]
\[
\text{Firstpref}->\text{First most suitable resource}
\]
\[
\text{Secondpref}->\text{Second most suitable resources}
\]
\[
\text{VM}->\text{Ready Queue for each Virtual Machine}
\]
\[
\text{noMatch}->\text{an array used to store remaining jobs which are not allocated}
\]

Step 5: If all the required resource parameters are processed then go to step 6, otherwise go to step 2.

Step 6:
\[
j = 1
\]
\[
\text{for}(i=1;i<n;i++)
\]
\[
\{\text{if}(\text{firstPref} = \text{firstPref}_{i+1} \text{ and matchbit}[i]=0)
\]
\[
\{\text{VM}_{\text{firstPref}} = i
\]
\[
\text{matchbit}[i]=1
\}
\]

Where matchbit is used to identify the job is scheduled or not. If matchbit is 1 job is scheduled.
Step 7:
j=1
for(i=1;i<n;i++)
{
    if(secondPref\textsubscript{ij} = secondPref\textsubscript{ij+1} and
    matchbit[i]=0)
    {
        VM\textsubscript{secondprefij}=i
        matchbit[i]=1
    }
}

Step 8:
j=1
for(i=1;i<n;i++)
{
    if(firstPref\textsubscript{ij} = secondPref\textsubscript{ij+1} and matchbit[i]=0)
    {
        VM\textsubscript{firstprefij}=i
        matchbit[i]=1
    }
}

Step 9:
j=1
for(i=1;i<n;i++)
{
    if(secondPref\textsubscript{ij} = firstPref\textsubscript{ij+1} and matchbit[i]=0)
    {
        VM\textsubscript{firstprefij}=i
        matchbit[i]=1
    }
}

Step 10:
j=0
for(i=1;i<=n;i++)
{
    if (matchbit[i]=0)
    {
        notmatch[i]=i
    }
    j++;
}

Where nomatch is used to hold the jobs that are not scheduled.

Step 11: The VM which complete its tasks will fetch the job from the nomatch array and match the required parameter of user task with VMs parameter and if match found,VM will execute the job.

Step 12: if no jobs are matched, then completed VM fetch the jobs from heavily loaded queue for execution.

4 RESULTS AND DISCUSSION
Datasets for execution is taken from GoCJ dataset [14]. Dataset depicts Google Cluster Traces based on workload behavior. Consider 20 User tasks are in the wait queue and 5 servers available. Server Configurations are depicted below in Table 1.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
VM & MEMORY (KB) & CPU CAPACITY [MHz] & BANDWIDTH \\
\hline
VM1 & 3000000 & 2600 & 2000 \\
VM2 & 702948 & 3000 & 2500 \\
VM3 & 6469632 & 5900 & 5000 \\
VM4 & 4071424 & 5200 & 4700 \\
VM5 & 8388608 & 6500 & 5200 \\
\hline
\end{tabular}
\caption{Server Configuration}
\end{table}

Consider there are 20 users, 5 resources. Bandwidth and memory are the parameter for allocating resources. Cloudlet details are depicted in Table 2. First, it will compare the user required memory with the available VM memory. Choose two options (VMs) which are close to the required memory using Spline Linear Interpolation method. The first matched value is stored in the firstPref array and the second matched value is stored in the secondPref array. Then compare the user required bandwidth with the available VM bandwidth. Choose two options (VMs) which are close to the required bandwidth using Spline Linear Interpolation method. The first matched value is stored in the firstPref array and the second matched value is stored in the secondPref array.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Cloudlet & Instruction (MI) & Memory (KB) & Bandwidth \\
\hline
Cloudlet 1 & 127000 & 720000 & 2100 \\
Cloudlet 2 & 83000 & 2000000 & 2600 \\
Cloudlet 3 & 63000 & 700000 & 5100 \\
Cloudlet 4 & 63000 & 6500000 & 4800 \\
Cloudlet 5 & 65000 & 4100000 & 5200 \\
Cloudlet 6 & 15000 & 8400000 & 2400 \\
Cloudlet 7 & 91000 & 700000 & 2600 \\
Cloudlet 8 & 85000 & 6000000 & 5100 \\
Cloudlet 9 & 101000 & 4000000 & 5000 \\
Cloudlet 10 & 40000 & 8400000 & 4700 \\
Cloudlet 11 & 40000 & 2000000 & 5300 \\
Cloudlet 12 & 712500 & 70000 & 2600 \\
Cloudlet 13 & 59000 & 6500000 & 4900 \\
Cloudlet 14 & 113000 & 2100000 & 5000 \\
Cloudlet 15 & 87000 & 700000 & 2000 \\
Cloudlet 16 & 53000 & 6500000 & 2400 \\
Cloudlet 17 & 47000 & 3800000 & 5100 \\
Cloudlet 18 & 127000 & 8000000 & 4800 \\
Cloudlet 19 & 45000 & 6500000 & 3000 \\
Cloudlet 20 & 27500 & 410000 & 4000 \\
\hline
\end{tabular}
\caption{Cloudlets}
\end{table}

Schedule the jobs by comparing firstPref array of memory with firstPref array of bandwidth. If a match occurs, allocate a particular VM to that task. Compare all the jobs, if possible, allocate the VM to that the job. Similarly, schedule the Jobs to matched VM by comparing the secondPref array of memory and bandwidth. If still job remains unscheduled, compare the firstPref of memory with the secondPref of bandwidth and
schedule the jobs to the matched VM. If still job remains unscheduled, compare the SecondPref of memory with the firstPref of bandwidth and schedule the jobs to the matched VM. If still job remains unscheduled, store the remaining jobs in noMatch array. The initial allocation of jobs to the available VMs is shown in Table 3.

<table>
<thead>
<tr>
<th>VMS</th>
<th>JOBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>2 20</td>
</tr>
<tr>
<td>VM2</td>
<td>1 3 7 12 15</td>
</tr>
<tr>
<td>VM3</td>
<td>8 13 10 4 16 19</td>
</tr>
<tr>
<td>VM4</td>
<td>5 9 17</td>
</tr>
<tr>
<td>VM5</td>
<td>18 6</td>
</tr>
</tbody>
</table>

**Matched**: 11 14

Consider each VM have its own wait queue. Jobs 2 and 20 are scheduled for VM1. Jobs 1, 3, 7, 12 and 15 are scheduled for VM2. Jobs 8, 13, 10, 4, 16 and 19 are scheduled for VM3. Jobs 18 and 6 are scheduled for VM5. Jobs 11 and 14 are held in noMatch array. High-speed VM will complete its job quickly. Generally, high-speed VM will have high memory and high bandwidth. Low-speed VM will have Low memory and Low bandwidth. If VMs completed its job, first it will migrate the job from noMatch wait queue and execute it. If there is no job in the noMatch wait queue then it chooses from the high populated queue which stands in the bottom edge. VM3 completes job8 in 14.41ms, then it fetches job13 for execution. VM1 completes job2 in 15.96ms then it fetches job20 for execution. VM5 completes job18 in 19.54ms, then it fetches job6 for execution. VM3 completes its job in 21.25ms. Then VM1 search the noMatch array and get the matched job11 for execution. VM3 completes its job13 around 24ms and then fetch job10 which resides in the queue for execution. VM4 executes its job5 in 25ms the fetch job5 for execution. After 28.9ms, VM1 search nomatch array and fetches job14 for execution. VM3 fetches job4 for execution from its queue after highly populated queue and checks the available memory and bandwidth with the user required bandwidth and memory (it checks the last job in the queue), VM1 fetches job17 from VM4 queue for execution. VM3 fetches job3 for execution from its queue. VM5 again fetches job12 from VM2 for execution 31.12ms. Again VM3 fetches job16 from its queue for execution after 41.86ms. After completing the job1, VM3 fetches job3 execution in 42.3ms. VM5 fetches job10 for execution from VM2 after checking its own queue, noMatch array and each queue’s last job (check the available memory and bandwidth with the user required bandwidth and memory) in 42.61ms. In 50.67ms VM1 searches noMatch array for in 56ms. VM1 fetches job7 for execution from VM2 in 59.7ms. After execution, jobs are allocated to the VMs are shown in Table 4.

<table>
<thead>
<tr>
<th>ALLOCATION OF JOBS AFTER EXECUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>VM1</td>
</tr>
<tr>
<td>VM2</td>
</tr>
<tr>
<td>VM3</td>
</tr>
<tr>
<td>VM4</td>
</tr>
<tr>
<td>VM5</td>
</tr>
</tbody>
</table>

Job 2, 20, 11, 14, 17, 7 are scheduled to VM1, Job1,3 are scheduled to VM2, Job 8, 13, 10, 14, 16, 19 are scheduled to VM3, Job 5 and 9 are scheduled to VM4, Job 18, 16, 15 and 12 are scheduled to VM5. The average processing time is 17.39 milliseconds. The proposed algorithm is compared with Modified Round Robin algorithm in terms of waiting time, completion time, turnaround time and it is shown in Table 5.

<table>
<thead>
<tr>
<th>RESULT OF PROPOSED METHOD AND MODIFIED ROUND ROBIN METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Number</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

Table 6 shows the overall comparison between the proposed method and the Modified Round Robin. The proposed method shows the best results in terms of completion time, waiting time and turn around time. The proposed method is better than the Modified Round Robin because Migration cost is very less, balance workload to all VMs and no VMs are idle. Advantages of the proposed method are computation complexity is reduced, Migration cost is less because the proposed
algorithm does not migrate the running jobs, and it will migrate the jobs which are waiting in the waiting queue. Cloud providers can provide resources according to user demand. VMs are utilized properly. It minimizes the processing time of Jobs. Proposed work helps to execute all set of jobs such as small, medium and large jobs inefficient manner. The proposed algorithm aims to provide the user with the required memory and bandwidth. It also minimizes the processing time. This algorithm considered only one VM for each server. Additional parameters such as CPU speed, no of VMs required by the user and Response time, etc., may also be taken for further analysis in the future.

<table>
<thead>
<tr>
<th>Task</th>
<th>Average Time Taken</th>
<th>Proposed Method</th>
<th>Modified Round Robin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall completion</td>
<td>17.39ms</td>
<td>20.45ms</td>
<td></td>
</tr>
<tr>
<td>Overall Waiting time</td>
<td>26.66ms</td>
<td>121ms</td>
<td></td>
</tr>
<tr>
<td>Turnaround time</td>
<td>44.05ms</td>
<td>141.45ms</td>
<td></td>
</tr>
</tbody>
</table>

6 CONCLUSION
The proposed Spline Linear Interpolation method reduces the computation complexity, so it is very simple and easy to develop. Proposed work will consider both the benefits of cloud users as well as the cloud provider. Cloud Users can get the required resource in time. Cloud Providers can have efficient utilization of all the servers properly, thus leading to reduction in idle time and increased throughput. Workloads of the VM are properly managed, so completion time is minimized. Migration cost is less because the proposed algorithm does not migrate the running jobs, it will migrate the jobs which are waiting in the waiting queue. Proposed work helps to execute all set of jobs such as small, medium and large jobs in efficient manner. The proposed work completes all the jobs before 3.06ms, waiting time is also very less when compared to the Modified Round Robin algorithm.

7 REFERENCES