Load Balance Aware Improved Honey Bee Algorithm for Task Scheduling
CHAPTER 5

LOAD BALANCE AWARE IMPROVED HONEY BEE ALGORITHM FOR TASK SCHEDULING

As described in the preceding chapters, Task Scheduling is the basic issue in cloud computing. The number of clients of a cloud framework could be billions and the clients may originate from anywhere in the world. Therefore, extensive scale errand planning among the cloud providers often occurs, which turns out to be a pressing issue. The proficiency of the schedule is proficient has no effect on the execution of a cloud system. Therefore, while setting up cloud computing frameworks booking is an imperative concern. Task Scheduling done is to figure out a way to assign specific number of tasks to the suitable resources, and to minimise the total completion time as little as possible, which is a NP-complete issue.

A wide range of scheduling objectives set by clients and cloud providers, besides the most well-known Makespan foundation, goals like proficient use of resources, stack adjust, QoS [GK12] and minimizing total cost with budget constrain [SH10] are considered. However considering the load balance as an objective can give more prominent proficiency. In the current research works, the CPU and memory utilities are utilized as markers of load balancing.

However, due to the principal part of the completion time in the practical application of planning, ascertaining the level of load balance from the view of completion time is a more helpful and significant approach. This concept has been successfully utilized in [ZZY14] who employed Load balance aware Genetic algorithm
(LAGA). It was found that the Genetic algorithm is the one of the simplest and the most basic algorithm which cannot be efficient than most other swarm based optimization algorithms.

The time and memory complexity in GA would be a matter of concern. Introducing efficient optimization algorithms can enhance the task scheduling performance. Hence, the Improved Honey bee behaviour (IHBB) algorithm developed in the previous chapter has been utilized for efficient scheduling with load balance as a parameter. Hence, the time load balance (TLB) model is developed and adopted in the proposed Load Balance Aware Improved Honey Bee Algorithm (LBA-IHB).

5.1. TIME LOAD BALANCE FOR TASK SCHEDULING

Load balance is a circumstance that the vast majority of the resources are relied upon to be occupied as per a schedule, lastly adds to a balanced system, instead of most assets being inactive or overload. In other words, just when every resource is involved in load balancing, the perfect load balance circumstance comes true, resulting in sufficient and proficient Utilization of computing resources, within the minimum total completion time. Load balance model intends to manufacture a quantifiable model to decide the load balancing level of a cloud framework.

The proposed LBA-IHB considers load balance from the view of task completion time, Hence, the new model is called time load balance (TLB) model. It can be noticed that the balance value figured by this model is specified underneath as TLB value (TLBV). The correlation between TLBVs can determine which schedule is better load balanced when compared to other.
The n×m Task Scheduling problem is represented by the corresponding Resource-Task Model, and the characteristics of this problem can be described as an Expected Time of Completion (ETC) matrix, which contains completion time of each task with each resource. There is another matrix known as Expected Scheduling to Compute (ESC) matrix, which describes a solution to the task scheduling problem by recording the matching of tasks and resources.

Let \( T = \{T_1, T_2, T_3, \ldots, T_n\} \) a set of independent tasks, where \( T_i \) is the \( i \)-th task, \( 0 \leq i \leq n \), \( n \) is the total number of tasks and \( R = \{R_1, R_2, R_3, \ldots, R_m\} \) as a set of resources, where \( R_j \) is the \( j \)-th resource, \( 0 \leq j \leq m \), \( m \) is the total number of resources.

Then, \( C = \{T, R\} \) is defined as a \( n \times m \) Resource-Task model with \( n \) tasks and \( m \) resources, where \( T \) is a task set and \( R \) is a resource set.

Let \( ETC = \{ETC_{ij}\}_{n \times m} \) be the matrix of expected time of completion for each tasks in every resources corresponding to a \( n \times m \) Resource-Task model, where \( ETC_{ij} \) is the expected time of completion for task \( T_i \) to be executed by resource \( R_j \). For each Resource-Task Model, there is an \( ETC \) which serves as a fundamental attribute, and helps to describe the problem in mathematical point of view.

Let \( ESC = \{ESC_{ij}\}_{n \times m} \) as a matrix of expected scheduling to compute for an \( n \times m \) Resource-Task model, also known as a solution to the \( n \times m \) task scheduling problem, where \( ESC_{ij} = 1 \) if task \( T_i \) is scheduled on resource \( R_j \), otherwise \( ESC_{ij} = 0 \). Each \( ESC \) represents a possible solution to an \( n \times m \) task scheduling problem. Each matrix can be different from each other. To find the optimal \( ESC \) for a given \( ETC \) is the process of solving task scheduling problem.
Assume that for an $n \times m$ Resource-Task model, every task can only be assigned to one resource and can never be interrupted. The total time load of resource $R_j$ is defined as follows:

$$L_j = \sum_{i=0}^{n} ETC_{ij} \times ESC_{ij} \quad (5.1)$$

Where $ETC_{ij}$ is the matrix denotes the completion time of each task assigned to each resource, $ESC_{ij}$ is the matrix decides which task should be assigned to which resource, $0 \leq i \leq n$ and $0 \leq j \leq m$.

The average time load of all the resources is defined as:

$$EL_j = \frac{1}{m} \sum_{j=0}^{m} L_j \quad (5.2)$$

The difference between time load of resource $R_j$ and the average time load is $|L_j - EL_j|$. After the differences are added, the time load balance value (TLBV) can be defined as follows:

$$time \ load \ balance \ value = \sum_{j=0}^{m} |L_j - EL_j| \quad (5.3)$$

Thus, the load balance model is determined and is ready to be processed using the proposed LBA-IHB algorithms.

5.2. LOAD BALANCE AWARE IMPROVED HONEY ALGORITHM

As stated earlier, load balancing is an important process in cloud computing similar to task scheduling. Both these operations are directly related not only to each other but also to the performance of cloud computing paradigm.
However, Task Scheduling Algorithms are widely employed based on the completion time while compensating quality in load balancing. Hence, the proposed model employs the IHBB proposed in the preceding chapter named as the Load Balance Aware Improved Honey Bee (LBA-IHB) algorithm for the optimization. IHBB is a modified version of the Honey bee algorithm that uses the honey bee foraging behaviour.

The QoS parameters namely response time, availability, throughput and cost are utilized in the proposed IHBB along with load balance as an objective for efficient Task Scheduling. The fitness function of the IHBB checks for the optimal values of these QoS parameters.

Similar to Artificial bee colony and Honey bee behaviour algorithms, the IHB is also based on Honey bee foraging behaviour. The foraging behavior of honey bees was considered to develop of an intelligent optimization technique. The intelligent foraging behavior of the bee swarm focuses on the collection of honey for energy. Foraging begins only when suitable conditions such as temperature, weather, etc. are satisfied.

The bees maintains a thumb rule for foraging area which is be fixed around a hive for two miles, as the food obtained within this area will gain weight while the energy spent beyond this area will be greater than the energy obtained. Foraging at the extreme distances wears out the wings of the individual bees and reduces its life expectancy, thus reducing the efficiency of the bee colony.

The foraging behavior depends on the communication of distance and direction of food source by round dance, waggle dance and shaking signals of the individual bees.

Figure 5.1 shows the proposed LBA-IHB algorithm for efficient Task Scheduling.
Figure 5.1. Overall Structure of LBA-IHB
In the LBA-IHB, n scout bees are selected based on the number of tasks. The number of best locations is denoted as m, while the number of elite locations is denoted as e. The number of recruited bees in elite patches is represented as nep, recruited bees in non-elite patches as nsp and the size of the neighbourhood as ngh. A total of MaxIter iterations are initialized. T tasks are initialized with R resources are considered for which the fitness function is denoted as

$$Fitness = f\{\text{minRT} \cup \text{maxAvail} \cup \text{minCost} \cup \text{maxThroughput}\} + TLBV \quad (5.4)$$

Where minRT is the minimum response time, maxAvail is the maximum availability, minCost is the minimum cost and maxThroughput is the maximum throughput computed for each task T.

Based on this fitness value, the tasks are sorted and the honey bee foraging behaviour is then performed to determine the best VM for each task.

**Algorithm 5.1: LBA-IHB algorithm**

Initialize T, R

Calculate TLBV

Compute QoS parameters

Apply IHB

Generate the initial population size as n

Set m, e, nep, nsp, ngh, MaxIter

Generate initial population.

Evaluate Fitness Value of initial population.
Sort the initial population based on the fitness result.

While \((i \leq MaxIter)\)

\[ i = i + 1; \]

Select the elite patches and non-elite best patches for neighbourhood search.

Recruit the forager bees to the elite patches and non-elite best patches.

Evaluate the fitness value of each patch.

Sort the results based on their fitness.

Allocate the rest of the bees for global search to the non-best locations.

Evaluate the fitness value of non-best patches.

Sort the overall results based on their fitness.

Run the algorithm until termination criteria met.

End

Tasks T are allotted to appropriate VM

End

Thus, the IHB can be employed with the knowledge of Load balance model for efficient Task Scheduling. This approach utilizes the QoS parameters for improved performance.
5.3. PERFORMANCE EVALUATION

In this section, the performance of the proposed LBA-IHB algorithm for Task Scheduling is evaluated in CloudSim. Then the performance is compared with that of the ACTS and IHBB-LB algorithm to determine the efficiency of the proposed algorithm.

The number of VMs is initialized as 10 for the performance evaluation. 100 tasks are considered with each task assumed to be accepted from the user requests and each task is constructed using multiple sub-tasks.

All the sub-tasks correspond to the web services, where they are data transmission tasks or data read/write through disk. The tasks are 10 basic arithmetic computation programs (addition, subtraction, multiplication & division programs) in the form of web services, which can be further combined to construct more complex arithmetic problems. The number of subtasks per task is randomly set in [5, 10], and each subtask is an arithmetic computation selected from the 10 arithmetic computations. For example, a task T is made up of three services: multiplication, addition and subtraction. The arithmetic variables in our experiments are randomly generated with the scales from 5 to 50, and their data sizes range from 24kB to 80MB.

The performance metrics namely Makespan, TLBV, CPU Utilization and Bandwidth Utilization are employed for comparison.

Table 5.1 & 5.2 shows the performance comparison of ACTS, IHBB-LB and LBA-IHB algorithms.
Table 5.1. Makespan & TLBV Comparison of ACTS, IHBB-LB & LBA-IHB

<table>
<thead>
<tr>
<th>Number of tasks</th>
<th>Makespan (s)</th>
<th>TLBV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACTS</td>
<td>IHBB-LB</td>
</tr>
<tr>
<td>10</td>
<td>36.231</td>
<td>31.207</td>
</tr>
<tr>
<td>20</td>
<td>42.125</td>
<td>35.957</td>
</tr>
<tr>
<td>30</td>
<td>48.965</td>
<td>39.526</td>
</tr>
<tr>
<td>40</td>
<td>58.43</td>
<td>48.770</td>
</tr>
<tr>
<td>50</td>
<td>62.12</td>
<td>56.92</td>
</tr>
</tbody>
</table>

Table 5.2. CPU & Bandwidth Utilization Comparison of ACTS, IHBB-LB & LBA-IHB

<table>
<thead>
<tr>
<th>Number of tasks</th>
<th>CPU Utilization (%)</th>
<th>Bandwidth Utilization (bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACTS</td>
<td>IHBB-LB</td>
</tr>
<tr>
<td>10</td>
<td>5.45</td>
<td>4.23</td>
</tr>
<tr>
<td>20</td>
<td>6.01</td>
<td>5.56</td>
</tr>
<tr>
<td>30</td>
<td>7.98</td>
<td>6.12</td>
</tr>
<tr>
<td>40</td>
<td>9.546</td>
<td>7.57</td>
</tr>
<tr>
<td>50</td>
<td>11.345</td>
<td>9.76</td>
</tr>
</tbody>
</table>

**Makespan**

Makespan is defined as the overall time taken for task completion. It can be computed as in equation (4.1).
Figure 5.2 shows the comparison of the ACTS, IHBB-LB and LBA-IHB in terms of the Makespan. The number of tasks is taken in the x-axis, while along the y-axis the Makespan (seconds) is taken. When the number of tasks is 50, the ACTS has Makespan of 62.12 seconds, IHBB-LB has 56.92 seconds and LBA-IHB has 33.12 seconds. Thus the proposed LBA-IHB provides better scheduling with minimal Makespan with efficient consideration of load balance. The important parameter TLBV can evaluate the performance more accurately in the following section.
**Time Load Balance Value (TLBV)**

This parameter defines the efficiency of any task scheduling algorithm which considers the load balance as an objective.

![Figure 5.3. Comparison of TLBV](image)

Figure 5.3 shows the TLBV comparison of the ACTS, IHBB-LB and LBA-IHB. The tasks are taken in the x-axis, while the TLBV is taken along the y-axis. When the number of tasks is 50, the ACTS has TLBV of 5219, IHBB-LB has 4321 and LBA-IHB has 3876. This shows that the LBA-IHB consumes has less values of TLBV which is highly efficient. The evaluations are then made in terms of CPU Utilization as given in the following section.
**CPU Utilization**

CPU Utilization refers to the usage of processing resources, or the amount of work handled by a CPU. CPU Utilization varies depending on the amount and type of managed computing tasks. It is estimated using equation (3.7).

![CPU Utilization Graph](image)

**Figure 5.4. Comparison of CPU Utilization**

Figure 5.4 shows the comparison of the ACTS, IHBB-LB and LBA-IHB in terms of the CPU Utilization. In the x-axis, the number of tasks is taken while along the y-axis the CPU Utilization in % is taken. When the number of tasks is 50, the ACTS has CPU Utilization of 11.345%; IHBB-LB has 9.76% while the LBA-IHB has 8.32%. This shows that the proposed LBA-IHB has less CPU Utilization. Finally the evaluation in terms of bandwidth is given below.
**Bandwidth Utilization**

Bandwidth is the amount of data that can be transmitted in a fixed amount of time. and is given in bits per second (bps). It is estimated using equation (3.8).

![BANDWIDTH UTILIZATION](image)

**Figure 5.5. Comparison of Bandwidth Utilization**

Figure 5.5 shows the comparison of the ACTS, IHBB-LB and LBA-IHB in terms of the Bandwidth. In the x-axis, the number of tasks is taken while along the y-axis the Bandwidth in bps is taken. When the number of tasks is 50, the ACTS shows a Bandwidth of 34.123 bps; IHBB-LB shows 25.25 while the LBA-IHB shows 17.14 bps. Thus, it can be seen that the proposed LBA-IHB has a significantly lower Utilization of bandwidth proving its efficiency.
5.4. SUMMARY

The Utilization of load balance as an objective for efficient task scheduling was described in this chapter. The proposed LBA-IHB based task scheduling algorithm and the evaluation evaluated to determine its efficiency. The performance of the proposed LBA-IHB algorithm has been compared with that of the preceding research methodologies ACTS and IHBB-LB in order to determine the efficiency of the proposed research methodologies. The experimental results justify its performance of LBA-IHB. The following chapter concludes the research and also provides suggestions for future researches.