MULTI OBJECTIVE INTEGRATED CROSSOVER BASED PARTICLE SWARM OPTIMIZATION FOR LOAD BALANCING IN CLOUD DATA CENTERS

NEHA SETHI, SURJIT SINGH, GURVINDER SINGH

1Research Scholar, Department of Computer Application, IKGPTU, Jalandhar, India.
2Professor, Department of Computer Application, IKGPTU, Jalandhar, India.
3Professor, Department of Computer Science and Engineering, GNDU, Amritsar, India.

E-mail: 1neha_tejpai@yahoo.com, 2surjit.gndu@yahoo.com, 3gsbawa71@yahoo.com

ABSTRACT

Cloud computing is becoming popular day by day, due to its wide range of scalable and dynamic characteristics. The increase in number of cloud users leads to the imbalance in resources and cloud data centers utilization and drastically improves the energy consumption. Therefore, it increases the data centers cost and waiting time of cloud users. Therefore, improving the waiting time and optimum usage of cloud resources has become a challenging issue. Many approaches have been designed to balance the user loads between cloud data centers. However, majority of existing load balancing approaches suffer from premature convergence issue, stuck in local optima issue and poor convergence issue. To overcome this issue, in this paper a multi-objective integrated crossover based particle swarm optimization has been proposed for balancing the load between cloud data centers. The proposed technique has been designed and implemented in MATLAB 2013a tool with the help of parallel processing toolbox. Extensive analysis reveals that the proposed technique outperforms existing techniques in terms of average waiting time, makespan and degree of imbalance. Analysis of Variance (ANOVA) based statistical testing has also been utilized to evaluate the significant improvement of the proposed technique.

Keywords: Particle Swarm Optimization, Meta-heuristics, Cloud data centers, Load balancing, Analysis of Variance.

1. INTRODUCTION

With the advancements in cloud environment, several commercial enterprises as well as another computational domain are progressively serious for executing workflow applications with sensitive intermediate information [1]. However, majority of existing scheduling techniques are unable to schedule the jobs between available high-end servers in more significant way [2]. Because, load balancing in cloud environment is an NP-hard problem. Therefore, a genetic based scheduling technique is designed, to schedule the jobs among available high-end servers [3]. Muthiah and Rajkumar [4] utilized Artificial Bee Colony algorithm (ABC) to reduce the makespan of task scheduling method. The authors have proved that ABC produces much better results as compared to Genetic Algorithm (GA).

Xu and Yang [5] designed an algorithm known as Cost-Greedy Price-Adjusting (CGPA) to optimize the incentives for both parties i.e., resource providers and grid users. The basic load balancer model is shown in Figure 1.
It is a multi-objective method which considers execution rate, deviation of profits, and combined cost as fitness functions. Sana and Rezaeian [6] utilized a hybrid meta-heuristic technique i.e., artificial immune system (AIS) to reduce the completion time for load balancing problems. In this approach, limited number of buffers is used between successive machines. Tiwari and Vidyarthi [7] used the concept of lazy ant to improve the auto controlled ACO algorithm for grid scheduling issue. This method has better performance and efficiency as compared to earlier optimization techniques. The agent-based prioritized dynamic round robin method is developed to efficiently schedule the jobs in cloud data centers [8]. But, this technique is limited to a single objective problem only.

Contribution: Our main contributions in this paper are as follows:

- In this paper, we propose a load balancing technique considering particle swarm optimization for cloud computing environment.
- First of all, the crossover based particle swarm can better coordinate the distribution of jobs and the allocation of jobs.
- The loads between the High-end servers (HES) are also balanced to reduce the waiting time further.
- In a word, load balancing and load balancing technique considering crossover based particle swarm with multi-objective fitness function are designed for cloud computing environment.

Rest of paper can be represented as follows: In section 2, related work has been discussed. Section 3, defines mathematical model for cloud computing environment. In Section 4, proposed technique is described. The experimental set-up and results are demonstrated in Section 5. The conclusion and future work are outlined in Section 6.

2. RELATED WORK

A parallel load balancing method is designed to schedule the parallel workload [9]. The list scheduling based server assignment technique is developed which always evaluate an optimal server assignment that minimizes the makespan [10]. The fresh multi-agent strengthening recovering technique, termed Ordinal sharing learning (OSL) technique is suggested for work arrangement problems. The actual OSL technique may solve the purpose of load balancing effectively [11]. A new parallel work arrangement plans which is dependent on integer straight line encoding is proposed. The actual marketing issue can help determine which job opportunities need to function along with the frequency [12]. Multi-objective optimization based load balancing techniques have ability to assign jobs on high-end servers in such a way that two opposite objective functions can be achieved simultaneously [13]. Therefore, multi-objective optimization based techniques can assign jobs between cloud servers in more efficient way by designing an efficient objective function considering various quality measures such as makespan, energy consumption, waiting time, reliability etc. [14]. Dynamic algorithm for resource allocation in cloud using fuzzy logic and pattern recognition based on power and storage parameters. The propose algorithm is derived from Fast Bid algorithm. The algorithm tries to improve the network traffic and communication load over the system [15]. This work contributed a review and comparative study or current state of art cloud resource scheduling and allocation algorithms for cloud. Moreover, this work proposes a taxonomy for resource allocation in cloud environment, which shows various ways to solve the issue of resource allocation and different aspects of resource allocation [15]. Major contribution this work is the broad study and classification of various ways to improve power consumption in cloud environment [17]. In order to boost the search efficiency, the min-min and max-min algorithm are used for the population initialization. But these may stuck in local minima and to find best solution genetic algorithm is proposed [18]. cost efficient data / storage allocation algorithm using genetic algorithm for video and metadata storage over cloud. This algorithm aims to provide heterogeneous memory storage space over cloud with least cost using genetic programming to select cheapest service provider. Output proves that the proposed algorithm proves to provide improved communication costs, data move operating costs and energy performance [19]. Big Bang-Big Crunch optimization algorithm to solve the problem of scheduling classed for timetable. This algorithm has proved to perform better than existing GA based algorithm [20]. Cloud computing, deals with many research issues in load balancing algorithms such as fault tolerance, high performance and increasing of storage [21]. The traditional scheduling methods are used in cloud computing on a first come first served basis. Using this method the current load is divided equally to VMs one by one [22]. Cloud data centre in which a set of application servers is hosted. Each server runs in a virtual machine in the cloud, subjected to a workload is allocated in available VM, whereby
session data on one server is copied to other servers for purposes of high availability [23]. Cloud computing, virtualization is a key concept. The major aspect of virtualization shares the physical resources by many users [24]. The cloud computing provides information and services to the user through the Internet. The scheduling of information resources becomes very complex in cloud computing [25]. The data replication has been widely used to increase the data reliability in cloud storage systems where failures are normal. To identify the failure VM and reliable VM, the reliability method is used [26]. The cloud infrastructure has many design challenges. One of the challenges in cloud is reliability. The reliability can identify the failure VM [27]. The replication of data is an effective solution to achieve efficient performance in reliability. The data replication placement strategy in the system is critical [28].

3. APPLICATION AND CLOUD MODEL

A workflow application is demonstrated by a Directed Acyclic Graph (DAG, (T, E)), represented by a mathematical model C_T (T, E), where T represents group of n jobs \{t_1, t_2, \ldots, t_n\}, and E represents e edges. Each edge demonstrates the dependencies between jobs. Every \( t_j \in T \) demonstrates a job in the service and every edge \( (t_q, t_i) \in E \) demonstrates a precedence assumption, such that burst time of \( t_j \in T \) cannot be initialized before tie \( T \) ends up its burst time [15]. If \( (t_q, t_i) \in E \) then \( t_i \) is parent of \( t_q \), and \( t_q \) is child of \( t_i \). A job with no parent is called an entry job and a job without children is called as exit job. The job size \( (A_{t_j}) \) is demonstrated in Millions of Instructions (MI).

Cloud model demonstrates service provider which provides, \( M \) number of resources \( \{R_1, R_2, \ldots, R_M\} \) at several processing powers and numerous costs. It is supposed that any service from set, \( R_{ce} \) has ability to run all jobs of a service provider. The processing power of a resource \( r_p \in R_{ce} \) is demonstrated by Millions of Instruction per Second (MIPS) and is represented by \( P_{r_p} \). The cost model dependent upon pay-as-you-go basis is similar to present profitable clouds.

Every job can be run on numerous resources. The burst time, \( E_{\text{burst}} T_{\text{init}}(j_p) \) of a job \( t_j \) on a resource \( r_p \) is evaluated as follows:

\[
E_{\text{burst}} T_{\text{init}}(j_p) = \frac{ZT}{P_{r_p} P_{r_p}} \quad \ldots \ldots \quad (1)
\]

And the execution cost \( E_{\text{exe}} C_{(j_p)} \) is calculated as follows:

\[
E_{\text{exe}} C_{(j_p)} = \mu P E_{\text{burst}} T_{\text{init}}(j_p) \ldots \ldots \quad (2)
\]

Here \( \mu P \) is the cost unit of utilizing resource \( r_p \) for each time interval. However, all computation resources of a service provider are supposed to be in similar physical area, so data storage and transmission costs are supposed to be 0 and the mean bandwidth among these resources is supposed to be coarsely equal. Only, time to communicate data among two dependent jobs \( (c_i) \), which are mapped to several resources, is considered through experimentation [16]. Assume \( \text{EST} (t_j, r_p) \) and \( \text{EFT} (t_j, r_p) \) represent the earliest execution initialization time and earliest finish time of a job \( t_j \) on a resource \( r_p \), respectively. For the entry job, \( \text{EST} \) can be calculated by:

\[
\text{EST}(t_{\text{entr}} r_p) = \text{avail}(r_p) \ldots \ldots \quad (3)
\]

For other jobs in DAG, we calculate earliest execution initialization time (EST) and earliest finish time (EFT) recursively as follows:

\[
\text{EST}(t_j, r_p) = \max \left\{ \text{max} \left[ \text{AFT}(t_i) + c_i \right], \text{t}\text{ie per max} \right\} \quad \ldots \ldots \quad (4)
\]

\[
\text{EFT}(t_j, r_p) = \text{ET}(j_p) \ldots \ldots \quad (5)
\]

Here \( \text{ET}(j_p) \) is the time when the resource \( r_p \) is ready for job execution. \( \text{AFT}(t_j, r_p) \) and \( \text{EFT}(t_j, r_p) \) determine actual start time and actual finish time of jobs \( t_j \) on resource \( r_p \), respectively. These may be dissimilar from job’s \( \text{EST}(t_j, r_p) \) and \( \text{EFT}(t_j, r_p) \). The makespan define maximum of load length of exit jobs \( t_{\text{exit}} \) and is evaluated as follows:

\[
M_{\text{SPN}} = \max \left\{ \text{max}(t_{\text{exit}}) \right\} \ldots \ldots \quad (6)
\]

Energy model utilized in this paper is taken from the capacitive power (\( P_{c} \) of
complementary metal-oxide semiconductor based logic circuits [17] which is given by:

\[ p_c = A C_{ap} V_g f_e \ldots (7) \]

Where \( A \) defines number of switches per clock cycle, \( C_{ap} \) represent capacitance load, \( V_g \) is the supply voltage, and \( f_e \) is the frequency. The energy consumed by executing workflow jobs over available resources can be calculated as follows:

\[ E_{gs} = \sum_{i=1}^{m} A C_{ap} V_g^2 f_e = E_{gs} T_{sk(i)} + EST(t_j, r_p) \ldots (8) \]

Where \( V_g \) define supply voltage of high end server on which job \( n_i \) is executed, and \( ET(j, r_p) \) is burst time of job \( j \) on assigned resource \( r_p \).

4. SCHEDULING USING MULTI OBJECTIVE PARTICLE SWARM OPTIMIZATION

PSO is one of the well-known evolutionary approaches inspired by nature and introduced by Kennedy and Elberhart (1995) [5]. PSO is a technique which achieves near to optimal results without the explicit information of the problem [6]. It simulates the process of preying birds swarm. Due to its ability of global searching, it has been implemented in several applications in many areas. A group of particles are randomly developed where position of each particle shows a possible solution point in search space [7]. Each particle stores the coordinates of best global solution (gbest) and the current local best (pbest) solution [8]. The proposed load balancing technique is initialized by developing random chromosome using normal distribution with mean = 0 and variance = 1. Each character of random population represent a given job number. However, jobs whose results are required by more than one job will be duplicated in developed random schedules. The velocity of each developed population is evaluated with the help of multi-objective fitness function. Then, pbest and gbest. t values are calculated to determine the best solution. Subsequent section describes multi-objective particle swarm optimization based scheduling technique.

4.1. Multi-Objective optimization

Load balancing in cloud computing, where both processing and bandwidth constraints at several high-end servers need to be considered, is a challenging problem. Furthermore, directing the optimization of multiple objective functions makes it even more interesting issue. In this paper a load balancing technique based on particle swarm optimization in which multiple and conflicting objectives are concurrently optimized. Precisely, we improve job execution quality while minimizing the energy consumption.

Multi-objective fitness function is mathematically computed as follows:

\[ \text{Max}(x) = \frac{1}{\text{Energy}(y)} + (1 - \omega) \times \text{Quality}(y) \ldots (9) \]

Here, \( x \) stores the objective values of each solution. \( \text{Energy}(x) \) represents energy consumed by schedule \( y \). Quality(\( y \)). Indicate the quality of given schedule. Subsequent sections describe various steps which are used to maximize the designed fitness function.

(a) Velocity updating

The velocity can be updated for each particle as follows:

\[ v_{(i)}^{t+1} = \omega v_{(i)}^t + C_{ap(1)} r_{(i)}^{(t)} (g_{best} - x_{(i)}^{t}) + C_{ap(2)} r_{(i)}^{(t)} (p_{best} - x_{(i)}^{t}) \ldots (10) \]

Here, \( \omega \) represents inertia weight. \( C_{ap(1)} \) and \( C_{ap(2)} \) Symbolize cognitive coefficients based on particle's velocity. This inertia weight, \( \omega \), controls a push with the particle. Enlargement around solutions will be evaluated by reduction of a \( \omega \) linearly looking at the extreme cost to its cheapest price, with creation, it's price, \( \omega \), can be calculated by:

\[ \omega_R = (\omega_1 - \omega_2) \frac{\text{Max}(A) - A}{\text{Max}(A)} + \omega_2 \ldots (11) \]

In the same way, if \( C_{ap(1)} \) reduces the computation cost. Then, \( C_{ap(1)} \) and \( C_{ap(2)} \) can be evaluated as follows:

\[ C_{ap(1)} A = (C_{ap(1)} h_{max} - C_{ap(1)} l_{max}) \frac{A}{\text{Max}(A)} + C_{ap(1)} l_{max} \ldots (12) \]

\[ C_{ap(2)} A = (C_{ap(2)} h_{max} - C_{ap(2)} l_{max}) \frac{A}{\text{Max}(A)} + C_{ap(2)} l_{max} \ldots (13) \]

Here, \( \text{Max}(A) \) is maximum number of generations and \( A \) is the generation number.

(b) Updating Position Vector:

\[ x_{(i)}^{t+1} = x_{(i)}^{t} + v_{(i)}^{t+1} \ldots (14) \]
Here, \( \mathbf{v}_{d_t} \) defines position of particle at \( A_{th} \) generation; velocity of the particle at \( A_{th} \) generation

c) Fitness Function

The fitness utilized in proposed technique is calculated as follows:

\[
\text{fitness} = a \times \text{Time} + (1 - a) \times \text{Cost} \ldots \ldots (15)
\]

Where Time and Cost can be calculated as follows:

\[
\text{Time} = \max\{\text{AFT}(t_{ext})\} \ldots \ldots (16)
\]

\[
\text{cost} = \sum_{j=1}^{M} E(C_{ap\,ji}) \ldots \ldots (17)
\]

To balance the load in an efficient manner, we have designed multi-objective PSO technique. The designed approach has following steps:

4.2. Updating external archive

In proposed technique, elite archives utilized to save the non-dominated particles originated with search procedure. After determining the fitness value, every particle is evaluated for its domination with additional particles.

4.3. Perimeter assignment

When several solutions having similar dominance, then diversity perimeter is used, \( I(\mathbf{x}) \) whose value for any solution \( \mathbf{x} \) can be calculated by:

\[
I(\mathbf{x}) = \sum_{i=1}^{N} \frac{f_{x(i)}(y) - f_{x(i)}(z)}{\max(f_{x(i)}) - \min(f_{x(i)})} \ldots \ldots (18)
\]

4.4. Updating \( p_{best} \) and \( g_{best} \)

The \( g_{best} \) solution is elected from the solutions of present record which is sorted based upon non-dominance and perimeter considering binary tournament selection. For \( p_{best} \), particle positions are compared with most effective location of particle in prior to generation. If current best solution (cbest) has good fitness than best known \( p_{best} \) then \( p_{best} \) will be updated, continue otherwise

In the same way, if \( p_{best} \) has good fitness than best known \( g_{best} \) then \( g_{best} \) will be updated, continue otherwise.

4.5. Crossover operator

The crossover function plays significant role in multi-objective PSO to avoid getting stuck into local minima. The crossover probability, \( p \) (Crossover) in proposed technique can be calculated as follows:

\[
p = (\text{Crossover}) = 1 - \frac{A}{\max(A)} \ldots \ldots (19)
\]

Where \( A \) is present generation and \( \max(A) \) is maximum number of generations taken. For each particle a random number within the range (0, 1) is considered. If rand < \( p \), then randomly a job is elected from the particle for crossover.

Fig.2 shows the effect of crossover on the selected particles. Particles have been selected based upon the \( G_{best} \) values. It has been clearly observed that the crossover returns a single particle combination after mutating two particles. If this particle has significant multi-objective fitness value then will become best particle combination.

Fig.3 shows the diagrammatic flow of the proposed technique. It has been observed that the crossover will be applied on the outcomes of the particle swarm optimization. Parents for crossover operator will be selected based upon the \( G_{best} \) values of the particle swarm optimization.

5. EXPERIMENTAL RESULTS

This section describes the experimental setup for cloud computing environment. MATLAB 2013a tool is use with the help of parallel processing toolbox to balance the load between HESs. The Dell notebook computer is used with 8 GB RAM, 2.4 GHz Intel Core i5 processor with 2GB GPU built in. The proposed and other selected techniques (i.e., PSO [21], ACO [31], MVNS [29], and Game Theory [30]) are designed and
implemented on the same experimental platform. 4000 jobs are tested on every technique. Following subsection describes the comparison of proposed method with existing techniques.
Table 1 demonstrates the comparison between MVNS [29], ACO [31], PSO [21], and Game Theory [30] with proposed technique on average waiting time (in seconds). The Table has shown that the proposed technique takes lesser time compared to existing approaches. Thus, proposed method is more efficient than others techniques in terms of waiting time. It has been observed that average waiting time increases whenever there is increase in number of tasks. But, in the case of proposed technique, it has been found that the proposed method has quite less increase in waiting time than earlier methods.

From Table 1, it is proved that the proposed technique has significantly decreased the average waiting time of jobs. Compared with other methods the proposed method has considerably reduced the mean waiting time i.e. 1.9814%. It shows that proposed technique is more suitable for real-time cloud computing environment. Table 2 demonstrates the comparison between MVNS [29], ACO [31], PSO [21], and Game Theory [30] with Proposed technique regarding makespan time (in seconds).

Table 2 depicts that the proposed technique has lesser makespan when compared with existing technologies. Because the mean reduction in makespan in seconds is approximately 5.0143 %. Therefore, it indicates that the Proposed technique has smaller makespan than earlier techniques. Also, when the logical analysis is considered (i.e., a range of the makespan) it has been observed that proposed method is more significant than earlier techniques. Because average variation in makespan is 141 seconds which were 159, 191,177 and 149 in MVNS [29] , ACO [31], PSO [21], and Game Theory [30], respectively.

\[
D = \frac{\theta_{\text{max}} - \theta_{\text{min}}}{\bar{\theta}_{\text{avg}}}
\]  

(19)

Where \(\theta_{\text{max}}\) and \(\theta_{\text{min}}\) are the maximum and minimum \(\theta_{\text{avg}}\) along with each HESSs, \(\theta_{\text{avg}}\) is the average \(\theta_{\text{avg}}\) of HESSs. Load balancing system increases the degree of imbalance considerably.

Table 3 demonstrates the comparison between MVNS [29] ACO [31], PSO [21], Game Theory [30] with Proposed technique regarding a degree of imbalance. A schedule is said to best if it has close to 0 degrees of imbalance. Therefore, from the Table 3, we have proved that the proposed technique has lesser degree of imbalance. Therefore, proposed technique has balanced the load among HESSs in more efficient way than earlier methods.

From the Table 3 it has been observed that the proposed technique has the lesser degree of imbalance compared to earlier methods. The mean reduction in the degree of imbalance is 0.819 % when proposed technique is compared with other scheduling techniques. Therefore, experimental results clearly show that the proposed technique outperforms other techniques in terms of waiting time, makespan and degree of imbalance.

On comparing the results of proposed approach with the others, it has been observed that the proposed technique is able to achieve significant scheduling results. The mean reduction in results of proposed techniques over others in terms of makespan in seconds is 7.3458 %, waiting time in seconds is 6.7234 %, degree of imbalance is 3.4982 %. Thus, proposed technique is more efficient for scheduling jobs between cloud data centers.

6. Analysis of Variance (ANOVA) Based Significance Testing

ANOVA consists of statistical techniques which are utilized to evaluate the differences between group means and their connected techniques (such as "variation" between these techniques). It contains a statistical test of whether or not the means of different techniques are equally significant. Therefore, it has ability to evaluate the significance analysis of given set of techniques. It is a special form of t-test in which more than two attributes are there.

Fig. 4., Fig.5., and Fig. 6. prove that the proposed technique is significantly different from other techniques. And these figures reveal that the proposed technique has significantly reduced the waiting time, makespan and degree of imbalance. Therefore, proposed technique provides significant results compared to existing techniques.

From Fig. 4 to 6 it has been statistically verified that by using the Analysis of Variance (ANOVA), there is significant improvement in the results using proposed techniques. By reducing makespan, waiting time and degree of imbalance rate, all the proposed techniques indirectly reduced
carbon emissions and cooling requirements of the cloud data centers, leading to a further drop in the energy demand and helps in achieving green computing.

Table 1: Average waiting time analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2.29 ± 0.61</td>
<td>2.13 ± 0.72</td>
<td>2.09 ± 0.71</td>
<td>1.82 ± 0.61</td>
<td>1.79 ± 0.41</td>
</tr>
<tr>
<td>1500</td>
<td>3.71 ± 0.97</td>
<td>2.72 ± 0.71</td>
<td>2.67 ± 0.61</td>
<td>2.48 ± 0.71</td>
<td>2.18 ± 0.51</td>
</tr>
<tr>
<td>2000</td>
<td>4.74 ± 0.54</td>
<td>3.63 ± 0.71</td>
<td>3.38 ± 0.66</td>
<td>3.11 ± 0.54</td>
<td>2.83 ± 0.69</td>
</tr>
<tr>
<td>2500</td>
<td>5.54 ± 0.91</td>
<td>4.11 ± 0.64</td>
<td>4.03 ± 0.63</td>
<td>3.28 ± 0.71</td>
<td>3.07 ± 0.63</td>
</tr>
<tr>
<td>3000</td>
<td>5.16 ± 0.67</td>
<td>5.02 ± 0.74</td>
<td>4.91 ± 0.78</td>
<td>4.34 ± 0.67</td>
<td>3.27 ± 0.74</td>
</tr>
<tr>
<td>3500</td>
<td>6.23 ± 1.08</td>
<td>5.88 ± 1.03</td>
<td>5.67 ± 0.78</td>
<td>4.44 ± 0.86</td>
<td>4.29 ± 0.81</td>
</tr>
<tr>
<td>4000</td>
<td>7.21 ± 1.04</td>
<td>6.78 ± 1.12</td>
<td>5.28 ± 0.81</td>
<td>4.91 ± 0.81</td>
<td>4.81 ± 0.46</td>
</tr>
</tbody>
</table>

Table 2: Comparison of makespan

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>15181 ± 149</td>
<td>14158 ± 73</td>
<td>12357 ± 117</td>
<td>12126 ± 108</td>
<td>11887 ± 99</td>
</tr>
<tr>
<td>1500</td>
<td>21179 ± 158</td>
<td>19215 ± 144</td>
<td>16587 ± 138</td>
<td>15473 ± 125</td>
<td>14949 ± 97</td>
</tr>
<tr>
<td>2000</td>
<td>32186 ± 196</td>
<td>29549 ± 184</td>
<td>27348 ± 168</td>
<td>28981 ± 137</td>
<td>27449 ± 107</td>
</tr>
<tr>
<td>2500</td>
<td>37155 ± 197</td>
<td>35458 ± 192</td>
<td>32657 ± 176</td>
<td>30146 ± 143</td>
<td>29747 ± 118</td>
</tr>
<tr>
<td>3000</td>
<td>43114 ± 208</td>
<td>41145 ± 178</td>
<td>37497 ± 175</td>
<td>36784 ± 158</td>
<td>34994 ± 127</td>
</tr>
<tr>
<td>3500</td>
<td>48164 ± 217</td>
<td>46279 ± 204</td>
<td>38487 ± 187</td>
<td>36498 ± 165</td>
<td>35816 ± 134</td>
</tr>
<tr>
<td>4000</td>
<td>58165 ± 229</td>
<td>51244 ± 226</td>
<td>49146 ± 209</td>
<td>47657 ± 198</td>
<td>47547 ± 156</td>
</tr>
</tbody>
</table>

Table 3: Comparison based on degree of imbalance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1.31 ± 0.48</td>
<td>1.61 ± 0.39</td>
<td>0.81 ± 0.37</td>
<td>0.73 ± 0.36</td>
<td>0.69 ± 0.19</td>
</tr>
<tr>
<td>1500</td>
<td>1.78 ± 0.51</td>
<td>1.79 ± 0.47</td>
<td>0.71 ± 0.45</td>
<td>0.62 ± 0.46</td>
<td>0.61 ± 0.28</td>
</tr>
<tr>
<td>2000</td>
<td>1.16 ± 0.57</td>
<td>1.01 ± 0.47</td>
<td>0.94 ± 0.43</td>
<td>0.84 ± 0.48</td>
<td>0.80 ± 0.48</td>
</tr>
<tr>
<td>2500</td>
<td>1.52 ± 0.62</td>
<td>1.32 ± 0.56</td>
<td>1.15 ± 0.49</td>
<td>0.94 ± 0.47</td>
<td>0.89 ± 0.39</td>
</tr>
<tr>
<td>3000</td>
<td>1.44 ± 0.59</td>
<td>1.38 ± 0.68</td>
<td>1.19 ± 0.58</td>
<td>0.83 ± 0.53</td>
<td>0.79 ± 0.53</td>
</tr>
<tr>
<td>3500</td>
<td>1.91 ± 0.71</td>
<td>1.90 ± 0.79</td>
<td>1.05 ± 0.65</td>
<td>0.72 ± 0.61</td>
<td>0.71 ± 0.61</td>
</tr>
<tr>
<td>4000</td>
<td>1.98 ± 0.68</td>
<td>1.92 ± 0.62</td>
<td>1.15 ± 0.61</td>
<td>0.96 ± 0.56</td>
<td>0.91 ± 0.46</td>
</tr>
</tbody>
</table>

Figure 4: Significance testing of Waiting time
7. CONCLUSION

Cloud computing plays a significant role in storage and communication of big capacity data due to a rapid improvement in size and the number of organizational activities. Due to rapid increase in cloud users, an efficient load balancing technique become a challenging issue. Existing load balancing techniques suffer from premature convergence, stuck in local optima issue, poor convergence speed, etc. Therefore, to overcome these issues an integrated crossover based particle swarm optimization based load balancing technique has been proposed. Due to non-availability of actual cloud environment, simulation environment of cloud computing has been designed in the MATLAB 2013a tool. Extensive experiments have been performed on the proposed and existing load balancing techniques. Extensive experiments reveal that the proposed technique has lesser mean waiting time when compared with existing technologies. Because the mean reduction in mean waiting time in seconds is approximately 2.3416 %. Extensive analysis reveal that the proposed technique outperforms existing techniques in terms of average waiting time, makespan and degree of imbalance. ANOVA based statistical testing has also been utilized to evaluate the significant improvement of the proposed technique.

REFERENCES:


