Presenting A Method For Load Balancing In Cloud Computing Using Bat Algorithm

Azma Sareh, Akbari Reza, Keshavarzi Amin

Computer engineering group, science and research university of fars, master of science student, marvdasht, Iran.

Corresponding Author: Azma Sareh, Computer engineering group, science and research university of fars, master of science student, marvdasht, Iran.

Article history:
Received 15 April 2014
Received in revised form 22 May 2014
Accepted 25 May 2014
Available online 15 June 2014

Keywords:
Cloud computing, load balancing, bat algorithm, responding time, resource efficiency.

ABSTRACT

Cloud computing is one of the modern technologies in information technology world. Cloud computing provides user information needs in a subscriber setting, in the form of service. Using load balancing is essential for cloud computing service providers. We have implemented balance to provide better service, to create balance between services, and to increase the responding speed of system. Load in load balancing can be represented as cpu load, memory capacity, delay or network load. It has been presented different algorithm about load balancing. The goal of using bat algorithm in this research is to produce balance. Bat algorithm is a type of Metaheuristic algorithm and functions based on probability. The proposed algorithm decreases the speed of responding virtual machines in cloud computing, more than genetic algorithm and pso algorithm; and it works better than all existing resources. Likewise, bat algorithm is more efficient than pso algorithm with its higher isotropic speed.

INTRODUCTION

Cloud computing is one of the most modern technologies that has changed information technology world. Cloud users can use or reserve resources online. Its advantages are scalability, virtualization and lower costs. Cloud computing contains different models such as public cloud, private cloud, hybrid cloud and community cloud [15].

Cloud computing is a model for accessibility to resources based on user demand. It works by a network. It can provide demands of users quickly and does not need much management. Therefore, based on NIST, the basic characteristics of the cloud computing are On-demand self-service, Broad network access, Resource pooling, Rapid elasticity and Measured service[11].

In cloud computing user needs are presented as services and users pay expenses as much as they use the service. Cloud computing services have different types. To separate cloud services XaaS “model or anything as a service can be used. X can be hardware, software, data, platform, etc [5]. To provide resources for users effectively, service providers use load balancing, a challenge existing in cloud computing [4].

Load balancing is a technique which controls load between service providers and resources, and it reduces the load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work [13,2]. In recent years, different ways have been presented to produce load balancing. Some of these algorithms are ant colony algorithm [6] particle swarm optimization [8] and genetic algorithm [1,12]. The objective of the present study is to use bat algorithm to produce load balancing in cloud computing service providers. Using this algorithm to produce load balancing, reduces the responding time, more than genetic algorithm and pso algorithm, and also leads to efficient use of all existing resources and better performance.

The article is structured as below; Section 2 reviews the literature on load balancing. Section 3 outlines the details of bat algorithm and section 4 describes our suggested algorithm and the used equations. Section 5 compares our suggested algorithm with genetic and pso algorithms. And finally, section 6 reports the results along with some suggestions for future studies.

Corresponding Author: Azma Sareh, Computer engineering group, science and research university of fars, master of science student, marvdasht, Iran.

E-mail: Sare.azema@yahoo.com
2. Literature review:

In recent years meta-heuristic ways have been used to provide load balancing. In this section, some of these ways would be reviewed. Honeybee algorithm [10] is used to coordinate the servers which perform the web services. The explorer has been sent to find the suitable food resources; when they found a source, they came back to the hive to declare it with a show which is known as "waggle dance". Based on the waggle dance, and the quantity or quality of nectar, it will be understood that whether the food source is suitable or not or what is its distance to the hive. The servers act as the bee and they have p₁ or p₂ possibility. When variation increases, the honeybee algorithm would have better performance. But when the size of system size changes, the honeybee algorithm does not change and even it works more efficiently[10].

In genetic algorithm [1], at first the question will be coded very well. Then, the suitability function will be chosen to engender their children by Mutation and Crossover function. This operation is repeated several times, until the provided groups have provided the convergence criteria. When the goal is achieved, the repetition will stop. This algorithm tries to achieve responding time saving and source use saving [1].

The efficient innovational algorithm of ant's colony was presented to provide distributed load service in cloud computing architecture [6]. AS such, up-to-dating the pheromone mechanism was improved effectively and it also improved the effect of instruments in load balancing. According to many species, ants provided the pheromone ways while they were moving from colony to the food. They follow this way. The advantage of these series is the points where amount of pheromone is high there. Because the pheromone vapors and vapors and vanishes, this algorithm is based on accident and possibility; and these tow factor have important function in the algorithm [6].

3. Bat algorithm:

Bat algorithm was presented by yang in 2010 who was inspired by the natural behavior of bats. Bats use a type of sonar, called echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Pseudo bat algorithm code is represented in figure1. This algorithm is presented by this instruction [7]:

1) All bats use echolocation to sense distance, and they also ‘know’ the difference between food/prey and background barriers in some magical way;
2) Bats fly randomly with velocity vi at position xi with a fixed frequency f_{min} varying wavelength and loudness A₀ to search for prey. They can automatically adjust the wavelength \( \lambda \) (or frequency) of their emitted pulses and adjust the rate of pulse emission \( r \) \([0, 1]\), depending on the proximity of their target;
3) Although the loudness can vary in many ways, it is assumed that the loudness varies from a large (positive) A₀ to a minimum constant value \( A_{min} \).

Objective function \( f(x), x=(x^1, ..., x^n) \), Initialize the bat population \( x_i \) and \( v_i, i=1,2, ..., n \).
Define pulse frequency \( f_{max}, f_{min} \), \( \forall i=1,2, ..., m \).
Initialize pulse rates \( r \), and the loudness \( A_v, i=1,2, ..., m \).
While \( t<T \)

For each bat \( b_x \) do

Generate new solutions through Equations (1), (2), (3).
If \( rand > r \), then

Select a solution among the best solutions.
Generate a local solution around the best solution.
If \( rand < A \), and \( f(x) < f(\hat{x}) \), then

Accept the new solutions.
Increase rand and reduce \( A_v \).

Rank the bats and find the current best \( \hat{x} \).

Fig. 1: Pseudo bat algorithm [14].

In implementation, the virtual bats are used. The instructions are identified so that position of \( x_i \) and velocity of \( v_i \), bats frequency based on the existing conditions will be update in the pseudo code and in a research space with d dimension. The new solutions \( x_i \) and velocities \( v_i \), and frequency in t step will obtain as follow[14]:

\[
f_i = f_{min} + (f_{max} - f_{min}) \beta \\
v_i(t) = v_i(t-1) + (\hat{x} + x_i(t-1))f_i \\
x_i(t) = x_i(t-1) + v_i(t)
\]
Where $\beta$ will determine in $[0,1]$ distance, $x^j_t(t)$ is variant value of j for I bat in t step. Result of $f_i$ will be used to control bats step and movement domain. Here $\hat{x}$ is the current global best location (solution) which is located after comparing all the solutions among all the n bats [14].

Random walk is a conventional math way which is consisted of series of accidental steps. The random walk explains behavior observations of processes in different fields, thus the fundamental model, would be used to record the possible practices [9].

Yang, offered the random walk, to be able to change the possible solutions. Always, between the best solutions, the usual one will be chosen, then the random walk will be used to create a new solution for each bat [14].

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A$$

$A^i$ is loudness mean of all bats in t time and $\varepsilon [-1,1]$ represents direction and length of random walk. In bat algorithm, loudness is $A_i$, and pulse emission should be update during the frequencies. When the bat finds its prey, the loudness decreases, while pulse emission increases. Loudness can be any value. For example, we can choose $A_o=0$ and $A_{\text{max}}=100$. Loudness $A_i$ and pulse emission $r_i$ in each frequency of algorithm, will update as this [14].

$$A_{i+1}^t = \alpha A_i^t$$

$$r_{i+1}^t = r_i^t[1 - \exp(-\gamma t)]$$

Where $\gamma$ and $\alpha$ are constant here. In this step of algorithm the pulse emission $r_i(0)$ and loudness of $A_i$ is often chosen accidentally for each value, $0 < \alpha < 1$ and $\gamma > 0$ thus:

$$A_i^t \rightarrow 0 \quad r_i^t \rightarrow r_i^0 \quad \text{as} \quad t \rightarrow \infty$$

4. Proposal algorithm:

In cloud computing systems there are nodes which provide users needs. It is possible that some of the nodes have high load and some of them are idle. This factor reduces the system’s performance and increases the responding time. To prevent this problem, we can divide the load of each node, between the idle nodes, produce the load balancing between nodes, and increase the system’s performance. The goal of presenting this algorithm is to produce load balancing between virtual machines and decrease the responding time. Here the system must be able to use all existing virtual machines effectively. The proposed pseudo code of algorithm is shown in figure 2.

**Objective function** $f(x), x=(x^1, ... ,x^n)$.

**Initialize the bat population** $x_i$ and $v_i, i=1,2,...,m$.

**Define pulse frequency** $f, \text{at} x_i$, $i=1,2,...,m$.

**Initialize pulse rates** $r_i$ and the loudness $A_i, i=1,2,...,m$.

While $t<T$:

For each bat $b_i$ do:

- Generate new solutions through Equations(1), (2), (3).
- If rand $>r_i$, then
  - Generate $x_{\text{new}}$ through Equation (4).
- Generate function $f(x_{\text{new}})$.
- If $f(x_{\text{new}})<f(\hat{x})$, then
  - Accept $\hat{x}$.
  - Increase $r_i$.
- If $f(x_{\text{new}})<f(\hat{x})$, then
  - Accept $f(x_{\text{new}})$, $x_{\text{new}}$.

**Rank the bats and find the current best $\hat{x}$**.

**Fig. 2:** The proposal algorithm.

According to above equations and the proposed pseudo code, we can implement the bat algorithm like this:

Step 1) at first, we get value to all frequencies, pulse emission ratio $r_i$ and loudness $A_i$, for each bat.

Step 2) position and velocity of each bat determines accidentally in d dimensional space.

Step 3) fitness function of each bat, will be determine. Fitness function is a function which determines ability of each element.

Step 4) according to fitness function of each bat, the best position of bat is determined.

Step 5) by (1), (2) and (3) equations, frequency, velocity and position of each bat will be update.

Step 6) if pulse emission is less than the accidental value, random walk will be applied for that bat.
Step 7) if the new fitness function is less than the old fitness function, the value of the best solution is replaced with the new solution, and the value of pulse emission ratio and the loudness must be update.

Step 8) the algorithm will frequency from 5 to 7 steps, until the frequencies end in implementation of this algorithm.

Value of $\gamma$ and $\alpha$ are represented $\gamma$ and $\alpha=0.9$. In proposal algorithm the second condition is ignored to obtain the best solution. Figure 2 shows the proposal algorithm. Frequency range in bat algorithm is determined as $[0, 1]$ and the initial pulse emission ratio is determined accidentally for each bat.

For solving each problem, at first the fitness function, with a negative value must be invented. This value illustrates bat’s eligibility. Fitness function of each bat is separate from the other bats. In this implementation, bat’s position is determined in the location and the fitness function is:

$$f(x) = \sum_{i=0}^{d} x_i^2 \quad -10 < x < 10$$

(7)

The fitness function illustrates the implantation time and a bat is chosen which has less implementation time compared with other bats.

5) Simulation result:

To simulate the bat algorithm, the cloudsim simulation has been used [3] and the result is compared with genetic and pso algorithms. Characteristics of simulation machines of this implementation are shown in table (1).

In this section, the result of different experiments which are applied on the bat, genetic and pso algorithms are shown. In the first experiment, the responding time of algorithms have been analyzed. The second experiment calculates efficient use of the both algorithms and at the third experiment compares the convergence velocity of the bat algorithm with that of pso algorithm. The fourth experiment calculates change number of demand and at the end the performance of bat algorithm is compared to pso algorithm.

<table>
<thead>
<tr>
<th>Vmm</th>
<th>Ps number</th>
<th>Bw</th>
<th>Ram</th>
<th>Size</th>
<th>Mips</th>
<th>Virtual machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xen</td>
<td>1</td>
<td>100</td>
<td>512</td>
<td>10000</td>
<td>100</td>
<td>Vm0</td>
</tr>
<tr>
<td>Xen</td>
<td>1</td>
<td>100</td>
<td>512</td>
<td>10000</td>
<td>200</td>
<td>Vm1</td>
</tr>
<tr>
<td>Xen</td>
<td>1</td>
<td>100</td>
<td>512</td>
<td>10000</td>
<td>300</td>
<td>Vm2</td>
</tr>
<tr>
<td>Xen</td>
<td>1</td>
<td>100</td>
<td>512</td>
<td>10000</td>
<td>400</td>
<td>Vm3</td>
</tr>
<tr>
<td>Xen</td>
<td>1</td>
<td>100</td>
<td>512</td>
<td>10000</td>
<td>500</td>
<td>Vm4</td>
</tr>
</tbody>
</table>

5-1) First experiment: responding time:

At the first experiment, virtual machines responding time was studied, which is represented in figure 3. It is shown that bat algorithm's responding time compared to the two other algorithms is better; and it has less responding time as in bat algorithm operations which take more time are performed by a very fast processor.

5-2) Second experiment: efficient use of resources:

At the second experiment effective use of the resources are studied and the result is illustrated in figure (4). According to obtained result, bat algorithm uses all resources while the genetic algorithm and pso algorithm does not use virtual machines effectively.

The two experiments’ results are represented in table (2). As shown below, the bat algorithm uses sources better than the genetic and pso algorithms, and furthermore, it has better responding time. In this table, the responding is based on thirty frequency algorithm. In some performances, the genetic algorithm and pso
algorithm employ all resources but it is not permanent. The bat algorithm, however, uses all sources in all performances. 

![Graph showing performance comparison](image)

**Fig. 4:** Compare of sources benefit.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Responding time</th>
<th>Benefice of sources (has/not)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bat</td>
<td>0.09</td>
<td>Has</td>
</tr>
<tr>
<td>Genetic</td>
<td>0.2</td>
<td>Has not</td>
</tr>
<tr>
<td>PSO</td>
<td>0.22</td>
<td>Has not</td>
</tr>
</tbody>
</table>

**Table 2:** Total compare of algorithm.

5-3) **Third experiment: convergence velocity:**

Experiment 3 studied the convergence velocity. Analysis of the results suggests that the more bat algorithm frequencies, the more efficient response. This result is adoptable in pso algorithm too. But, by comparing the two algorithms and observation about their fitness function it is shown that bat algorithm's convergence velocity is almost better than that of pso algorithm. The result is represented in figure (5).

![Graph showing convergence velocity comparison](image)

**Fig. 5:** Bat algorithm's convergence velocity compare to pso algorithm.

5-4) **Fourth experiment: calculation of changing the number of demands:**

At this experiment, by increasing the number of demands, bat algorithm' responding time increases, while in comparing to pso algorithm, responding time's gradient is diminished, and it performs better. The results are presented in figure (6).

![Graph showing changing number of demands](image)

5-5) **Fifth experiment: performance:**

The final experiment is the calculation of bat and pso algorithm's performance. According to the obtained results, shown in figure (7), bat algorithm's performance is almost the same as pso algorithm. In this experiment and the following experiments, the number of virtual machines is 5.

6) **Results and future work:**

In this article, a new algorithm, i.e. bat algorithm, to calculate the load balancing in cloud computing. This algorithm is an accidental one; and it has been inspired from the bat's natural behavior. At implementation, the loudness variable is ignored and eliminated from the algorithm, as in analyzing the loudness, we would lose some of the best bat positions, and no efficient response is obtained. The bat algorithm, pso algorithm and the genetic algorithm are simulated by cloudsim simulator. They have same simulation machines, and the same
number of works. According to the obtained results, bat algorithm uses all resources to provide better responding time.

![Graph]

**Fig. 6:** Comparing the effect of increase in the demand's number in pso and bat algorithm.

![Graph]

**Fig. 7:** Calculate of bat algorithm and pso algorithm performance.

One of the load balance goals is to reduce the responding time which can be achieved by this algorithm. Another experiment which was implemented on bat and pso algorithms indicates that the performance of bat algorithm is almost better than that of pso algorithm. In future, efficiency and other important factors of this algorithm in load balancing can be studied and compared with those of other evolutionary algorithms such as honeybee and ant's colony.

**REFERENCES**


