Priority based Resource Allocation and Scheduling using Artificial Bee Colony (ABC) Optimization for Cloud Computing Systems

A. Phani Sheetal, K. Ravindranath

Abstract: In the Cloud computing systems, the existing works fails to address the failure rate of servers and no mechanism has been provided for fault recovery. In this paper, we propose a Priority based Resource Allocation & Scheduling Technique using Artificial Bee Colony (ABC) Optimization (PRAS-ABC) for cloud environment. Initially, the work load of server and resource requirements of users are predicted by the scout bees by monitoring the past resource utilizations and size of the allocated VM. With this predicted workload, the expected completion time of each server is estimated. Then the tasks requesting the resources are categorized based on the deadline and resource requirements. Then based on the work load and expected completed time, the servers are categorized. Then each category of task will be allocated to the respective category of servers. The proposed approach is implemented in CloudSim environment of Java and compared with existing techniques in terms of resource utilization, percentage of resources successfully allocated, percentage of missed deadlines, average work load of server etc.

I. INTRODUCTION

A. Cloud Computing

Cloud computing differs from the traditional computing which mainly depends on the personal devices. It allows to share the computing resources from a remote place. It provides flexible and minimum cost access for resources at any place any time. The shared resources can be hardware or software. Cloud offers various services like Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS). [1].

The advantages of Cloud Computing are: (i) Applications can be accessed as utilities throughout the web. (ii) No specific software installation needed for accessing cloud applications. (iii) The PaaS service provides various deployment tools and runtime environments (iv) It provides platform independent access to all clients. (v) Supports load balancing [2].

B. Resource Allocation and Scheduling in Cloud

Resource allocation can be static or dynamic. In static allocation, the cloud user requests for fixed amount of resources proactively. But static allocation causes poor or over utilization of resources [3].

In dynamic allocation, the users request for resources depending on the requirements of application by on demand.

When the requested resources are not available, the cloud service provider (CSP) allocates from other cloud data centers [4].

Advance resource allocation enables to plan the resources for future such that they can be allocated on demand [5].

C. Problem Identification

In multi agent based VM allocation approach [9], the resource utilization and energy consumption parameters are mainly considered for VM allocation. But it fails to consider the work load prediction or future resource requirements. Moreover, the deadline of each task was not considered.

In [11], when a high priority job (with low deadline) comes in, the low priority job (with high deadline) was preempted allowing the high priority job to run in its resource. But it neither checks the work load of the PMs nor checks the size of the requested resources. In [14], the work load based on future resources requirement is predicted. Then it migrates the VM from a hot spot to a cold spot based on resource utilizations. But this approach did not consider the energy cost for the utilized resources. Moreover, the deadline of each task was not considered.

There are some works available on ABC based task scheduling in cloud. Out of these papers, [22] and [24] are same. They consider the VM load in the fitness function for selecting the VMs. [23] considers makespan time and load balancing (they didn’t define these metrics) for fitness function.[25] also considers task completion time and load as fitness function for selecting the VMs. [26] is groups the user requests based on priority. But it considers only make span time as the fitness function for selecting VM.

But the main advantages of our solution over these works are:(i) we have predicted the load and completion time using EWMA from the past values. Only from these predicted values, the fitness function is formed. (ii) Our algorithm does not select individual VM but rather selects a server. (iii) We have classified the servers based on the priority of user requests.

II. RELATED WORKS

Ying Song et al. [6] proposed a two-tiered on-demand resource allocation mechanism, including the local and global resource allocation. A local on demand resource allocation algorithm is applied by the VM with a threshold value. In global resource allocation, the threshold value of local allocation can be adjusted adaptively.
SivaThejaMaguluri et al. [7] have discussed about traffic optimal resource allocation algorithms. Sheng Di et. al. [8] have designed a deadline based resource allocation algorithm and an error free technique for job completion time.

Wanyuan Wang et al [9] have introduced a decentralized multiagent (MA) based VM allocation approach. A local negotiation-based VM consolidation mechanism is developed to exchange the assigned VMs of agents for energy cost saving.

GandhaliUpadhye et al [10] have proposed Utility Accrual (UA) approach to associate each task with a Time Utility Function (TUF). The TUF denotes the utility attained by the utility system at the time when a task is completed to improve the performance.

Chen-Fang Weng et al [12] have designed adaptive neural fuzzy inference system (ANFIS) algorithm for predicting the load and deciding the resource allocation policy for VMS.

GavineKanakadurga et al [13] have presented the many dynamic resource allocation techniques. Resource provisioning was done by parallel processing using different types of scheduling heuristics.

III. PROPOSED SOLUTION

A. Overview

In this paper, we propose a Priority based Resource Allocation &Scheduling Technique using Artificial Bee Colony (ABC) Optimization for Cloud environment. Initially, the work load of server and resource requirements of users are predicted by the scout bees by monitoring the past resource utilizations and size of the allocated VM. With this predicted workload, the expected completion time of each server is estimated. Then the tasks requesting the resources are categorized based on the deadline and resource requirements. Then based on the work load and expected completed time, the servers are categorized. Then each category of task will be allocated to the respective category of servers. The proposed approach is implemented in CloudSim environment of Java and compared with existing techniques in terms of resource utilization, percentage of resources successfully allocated, percentage of missed deadlines, average work load of server etc.

B. Artificial Bee Colony (ABC) Technique

ABC algorithm is a heuristic method of discovering and sharing the food resources of honey bees. Each bee will search for nearby food sources and will share this information to other bees. There are three types of bees involved: Employed, onlooker and scout bees [14][15].

C. Assumptions

Let X = (M, N) be the cloud system
M = {m1, m2, ..., mi} be the set of servers interconnected by the communication network
\( \forall (m_i, m_j) \in N \)

This indicates that mi and mj communicate with each other through only one switch

Let \( c(p_i, p_j) \) be the communication distance between servers mj and mi

This is computed as the number of switches along the shortest path between mj and mi

Let \( \delta = \{\alpha_1, \alpha_2, ..., \alpha_n\} \) be the set of VM resource request required by users.

Let \( w_i \) be the amount of resources required by VM \( \alpha \in \delta \)

Each server owns number of resources that are capable of running multiple VMs.

Let c the amount of resources at the server mj

D. VM Allocation

A VM allocation \( \{\delta(m_i), \delta(m_j), ..., \delta(m_n)\} \) is defined as mapping of server to set a set of VMs that should satisfy the following two conditions:

1. Each VM should be allocated to at least one server and no VM is allocation to more than one server
2. For each server, the total resource requirements of its hosted VM does not exceed its available resources

E. Prediction of Work Load

Each server maintains a resource utilization history (RUT) per VM which contains the total resources utilized and total time taken to complete each services of users.

The workloads of each VM can be predicted using exponentially weighted moving average (EWMA) of past workloads of all VMs

\[ EL(t) = \alpha . EL(t-1) + (1- \alpha) . OL(t) \] (4)

where EL and OL are the estimated and observed loads at time t and \( \alpha \) is a constant.

Similarly, expected completion time of each server can be determined by estimating EWMA of past completion times of all VMs

\[ ECT(t) = \beta . ECT(t-1) + (1- \beta) . OCT(t) \] (5)
Where ECT and OCT are the expected and observed completion time of VMs at time \( t \) and \( \beta \) is a constant.

**F. Classification of Users Tasks**

When the users submit their service requirements, then the cloud server broker will classify the services and allocates priorities as shown in Table 1

<table>
<thead>
<tr>
<th>Dead line</th>
<th>Resources size</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short</td>
<td>High</td>
<td>1</td>
</tr>
<tr>
<td>Short</td>
<td>Medium</td>
<td>2</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>3</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table. 1 Classification of user tasks**

**G. Priority based Resource Allocation & Scheduling Technique**

Let \( Q \) be the solutions

Let \( C \) is the number of optimization parameters

Let \( T \) be the maximum number of cycles that the algorithm would run.

1. Initially scout bees are deployed in the network which collects the past resource utilizations and size of the allocated VM.
2. Based on the collected information, work load of server and resource requirements of users are predicted.
3. A fitness function is formed for each server \( S_n, i=1,2,\ldots,n \) using EL and ECT as

\[
g_{i} = \left( \lambda_{1}^* \text{EL} \right) + \left( \lambda_{2}^* \text{ECT} \right)
\]  

Where \( \lambda_{1} \) and \( \lambda_{2} \) are the weight values

4. Then roulette wheel selection method (of ABC) will be applied to find the best solution. In this method,
   a. The initial positions of food sources are randomly generated.
   b. During each iteration, the employed and onlooker bees seek for better solutions by performing neighbor search.
   c. For each solution \( q_{i} \), determine a neighbor \( b_{j} \), using Eq. (7)

\[
q_{b_{j}} = b_{j} + \sigma \left( b_{j} - b_{i} \right)
\]  

where \( \sigma \in \{1,2,3,\ldots,Q\} \) and \( k \neq i \), \( \sigma \) is a random number in the range \([-1,1]\)

j = 1, 2, 3, ..., V

k and j are randomly selected

d. A better solution is then selected between \( b_{j} \) and \( q_{i} \).

e. The onlooker bees are deployed near the food sources using the roulette wheel selection method.

f. It selects a food source at position \( b_{i} \) with a probability \( Z_{i} \) calculated as follows

\[
\sum_{j=1}^{3} \text{fit}_{j}
\]  

\[
\text{fit}_{i} = \left\{ \begin{array}{ll}
1 & , \quad \text{if } g_{i} \geq 0 \\
\frac{1}{1+g_{i}} & , \quad \text{if } g_{i} < 0
\end{array} \right.
\]  

where \( g_{i} \) = fitness of the solution

**IV. EXPERIMENTAL RESULTS**

The NASA workload [20] has been used as the emulator of Web users requests to the Access Point (AP). This workload represents realistic load deviations over a period time. It comprises 100960 user requests sent to the Web servers during a day. Abnormal deviations in this workload can trigger the load balancing mechanism. Research shows that the pattern of user request arrivals to websites is very much similar to the pattern of this workload. Table 3 shows the experimental parameters assigned in this work.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work load</td>
<td>NADA traces</td>
</tr>
<tr>
<td>Resource Utilization Thresholds</td>
<td>( U^{\text{low-thr}} = 20% ) and ( U^{\text{high-thr}} = 80% )</td>
</tr>
<tr>
<td>Response Time Thresholds</td>
<td>( RT^{\text{low-thr}} = 200ms ) and ( RT^{\text{high-thr}} = 1000ms )</td>
</tr>
<tr>
<td>Scaling Intervals</td>
<td>( \Delta t = 10\min )</td>
</tr>
<tr>
<td>Desired Response Time</td>
<td>DRT = 1000ms=1s</td>
</tr>
<tr>
<td>Load Balancing Policy</td>
<td>Round-Robin</td>
</tr>
<tr>
<td>Configuration of VMs</td>
<td>\text{t2.medium} and \text{t2.Large}</td>
</tr>
<tr>
<td>Maximum On-demand VM Limitation</td>
<td>MaxVM=10VM</td>
</tr>
<tr>
<td>Task and Resources Scheduling Policy</td>
<td>Time-Shared</td>
</tr>
</tbody>
</table>
A. Performance Metrics

This section explains experiments conducted to evaluate the performance of the proposed PRAS-ABC scheme in CloudSim Simulator [21]. The performance metrics considered for evaluation are as follows:

Response delay: The response time $R(t)$ represents the latency in the response to user requests. It is calculated (in minutes) according to Eq. (10).

$$R(t) = \frac{\sum_{j=1}^{Req_{ans}} W_j}{Req_{ans}}$$  \hspace{1cm} (10)

Where $Req_{ans}$ is the number of answered requests

$W_j$ is the waiting time of each answered requests given by

$$W_j = [F_j(t) - A_j(t)] . Ser(t)$$  \hspace{1cm} (11)

Where $F(t), A(t)$ and $Ser(t)$ are the finish time, arrival time and service time. Arrival time and the finish time are the times the request is received from user and the time the response is received by its VM, respectively. Service time is the estimated time required by the incoming user request.

Throughput: It represents the ratio of answered requests ($Req_{ans}$) to the total number of received requests ($Req_{rec}$) in percentage.

Resource utilization: It represents the mean% of the CPU utilization by calculating the average of all VMs utilization.

$$R_u = \frac{\sum_{j=1}^{RVM} U(VM_j)}{RVM}$$  \hspace{1cm} (12)

Here $RVM$ is the number of rented VMs and $U(VM_j)$ represent the utilization of VMj.

Missed deadlines ($M_d$): Requests experiencing the response delay of more than the requested deadline ($R_d$) are considered as missed deadline.

$$M_d = \sum_{j=1}^{Tot\req} W_j - R_d$$  \hspace{1cm} (13)

B. Results

Performance of PRAS-ABC

Fig. 2 Time Vs Response Delay

In this experiment time taken as x-axis and the corresponding delay is calculated. The performance can be evaluated through the graph.

Fig. 4 Time Vs VMs Allocated

In this experiment time taken as x-axis and the corresponding VMs allocation is calculated. The performance can be evaluated through the graph.

Fig. 5 Time Vs Allocation Decision

In this experiment time taken as x-axis and the corresponding allocation decision is calculated. The performance can be evaluated through the graph.

C. Performance of PRAS-ABC with Suprex

In this section, the performance of the proposed PRAS-ABC is compared with the super professional executor (Suprex) [19]. In Suprex, if the resources are under-provisioned, the executor adds a new VM. On the other hand, if the resources are over-provisioned, the executor releases a VM by selecting a VM from on-demand.

Fig. 6 No. of VMs Vs CPU Utilization

In this experiment number of VMs taken as x-axis and the corresponding CPU utilization is calculated. The performance can be evaluated through the graph.
Figure 6 shows the CPU Utilization measured for PRAS-ABC and Suprex when number of VMs are varied. The VMs are increased from 2 to 5, as we can see from the figure, the CPU Utilization of PRAS-ABC decreases from 35 to 17. Hence the CPU Utilization of Suprex is 13% of higher when compared to Suprex.

Figure 7 shows the Throughput measured for PRAS-ABC and Suprex when number of VMs are varied. The VMs are increased from 2 to 5, as we can see from the figure, the Throughput of PRAS-ABC decreases from 100.5 to 100.3, the Throughput of Suprex decreases from 98.6 to 96.2. Hence the Throughput of PRAS-ABC is 4% of higher when compared to Suprex.

Figure 8 shows the CPU Load measured for PRAS-ABC and Suprex when number of VMs are varied. The VMs are increased from 2 to 5, as we can see from the figure, the CPU Load of PRAS-ABC decreases from 160 to 21, the CPU Load of Suprex decreases from 172 to 39. Hence the CPU Load of PRAS-ABC is 42% of higher when compared to Suprex.

Figure 9 shows the percentage of missed deadlines for the requested services when the number of VMs is varied. The figure shows that PRAS-ABC attains 66% lesser missed deadlines than Suprex scheme.

The average values of each metrics for both the schemes are shown in Table 4.

V. CONCLUSION

In this paper, we have proposed a Priority based Resource Allocation & Scheduling Technique using Artificial Bee Colony (ABC) Optimization (PRA-ABC) for cloud environment. Initially, the work load of server and resource requirements of users are predicted by the scout bees by monitoring the past resource utilizations and size of the allocated VM. With this predicted workload, the expected completion time of each server is estimated. Then the tasks requesting the resources are categorized based on the deadline and resource requirements. Then based on the work load and expected completed time, the servers are categorized. Then each category of task will be allocated to the respective category of servers. The proposed approach have been implemented in CloudSim environment of Java and compared with existing techniques in terms of resource utilization, percentage of resources successfully allocated, percentage of missed deadlines, average work load of server etc.

REFERENCES


