Efficient Randomized Honeybee Algorithm for Allocation of Cloud Servers
Sakshi Dubey¹, Hari Mohan Singh²
Mtech Student, Assistant Professor
Department of Computer Science and IT
SHIATS, Deemed University, Allahabad, India

Abstract:
Most discussed topic nowadays “Cloud Computing” is becoming buzzword. The rapid improvement of the capacity of the online connectivity in this era gave birth to the cloud computing. Cloud computing is distributed architecture that enables organization or individual to provide on demand services and resources that are centralized on scalable platform in a seamless and cost effective manner. Load balancing is one of the most challenging task in Cloud computing in terms of utilization, throughput or fault tolerance etc. To maximize revenue, hosting centers must allocate servers to clients a cost effective way that the performance and utilization become better. Unpredictable arrival of requests, limited number of servers and cost of reallocation of servers make server allocation optimization difficult. The decentralized honeybee algorithm based on the similarity between server behavior and natural honeybee colony forager allocation which dynamically allocates server to balance request load. But the honeybee algorithm suffers problems with over allocation of servers to profitable virtual machines. In this paper, we enhanced the basic honeybee approach and proposed efficient randomized honeybee algorithm. The main idea of the proposed work is to release the servers from overloaded VMs and reallocate them to underutilized VMs. The simulation result shows that the proposed algorithm performs better than the honeybee algorithm, in case of arrival pattern is not predictable.

Keywords: Cloud Computing, Load Balancing, Honeybee Algorithm, Enhanced Randomized Honeybee Algorithm

I. INTRODUCTION
Technology innovation and its adaptation are two critical successful factors for any business and organization. Apart from construction and operation of extremely large scale commodity datacenter, additional technology trends, new business model and application opportunity also played a key role in making cloud computing a reality this time around. Cloud computing consists of clusters of distributed computers (cloud) providing on demand services or resources over a network with the scale and reliability of a data center. Most applications do not make equal use of computation, network bandwidth, storage or processing; some are network bound, others CPU-bound and so on, and may saturate one resources while underutilizing other. Cloud computing refers to both the applications delivered as service over the internet and the hardware and systems software in the data centers that provide those service. Pay-as-you-go can charge the applications separately for each type of resource. Cloud provides three types of services i.e. SaaS, PaaS and IaaS. The software services have been referred to as Software as a Service. SaaS clients rent usage of application running within the cloud provider infrastructure, for example- salesforce. One big benefit of SaaS is all that all clients are running the same software version and new functionality can be easily integrated by the provider. PaaS service offer an application platform as a service that enable clients to deploy custom software using the tools and programming languages offered by provider, for example Google App Engine. IaaS delivers hardware resources such as CPU, memory, or network component as a service on virtualization platform. When a cloud is made available in a pay-as-use manner to the general public, we call it Public Cloud. An internal datacenter of a business or other organization is refereed as Private Cloud. A hybrid cloud environment consisting of multiple internal and/or external provider. As Cloud computing tags “pay-as-use”, this is generally based on concept of virtualization. Virtualization in cloud computing is creation of virtual of something such as storage, network device, hardware, software or an operating system. Scalability and availability of resources is possible by the use of virtualization. In a virtualized environment IT enterprise has to manage many changes as the changes occur More quickly in virtual environment than in physical environment. The recent trend in virtualization is consolidation of datacenter thus, reducing the managing cost. If failure happen to the server or virtual machine, the cloud computing system transfer and backup those data to other machines and delete those failure node from the system automatically, it makes cloud reliable and more usable.

In Cloud computing, since resources are sharable, and number of servers are limited, load balancing is one of the major challenge for the cloud. Load balancing is required to achieve equally distribution of load among all the nodes, to ensure efficient utilization of resources and minimize the response time. It simultaneously removes a condition in which some of the nodes are overloaded while some others are under loaded. It is one of the element which shows the impact on performance stability and resource balancing of cloud computing. Load balancing is all about the availability, scalability and performance of cloud resources. Depending on the current load of the system, load balancing can be divided into two types: Static and Dynamic. Static load balancing depend on the current state of the system. It requires prior
knowledge of the system resources. A load balancing algorithm which is dynamic in nature does not depend on previous state or behavior of the system. The load is amount of work that a computation system performs, that can be classified in terms of network load, storage capacity, memory capacity and CPU load. Many researchers have been proposed various techniques to improve the load balancing.

Internet hosting centers host services for weather, e-banking, ticket reservation, e-auction, online stock trading and others. There are many considerable things for developing dynamic load balancing, for example- estimation and comparison of load, stability and performance of different system, interaction between nodes and selection of nodes and types of service which is to be assigned. Studies of social animals and social insects have been resulted in a number of computational model like swarm intelligence. Researchers have developed computation optimization method on biology such as Practical swarm optimization, Genetic algorithm, Ant Colony. One of the most popular nature-inspired approach is Honeybee algorithm which mimic the idea of real bee colony. But the problem with honeybee is that the over allocation of servers to profitable virtual machines started decreasing the revenue rate. To overcome the excess allotment problem of servers, we proposed a new Randomized Honeybee approach.

The aim of the paper is to enhance honeybee algorithm and describe an optimization algorithm called Randomized Honeybee Algorithm, inspired from the natural foraging behavior of the honeybees, to find the optimal solution. The algorithm works based on the classical honeybee algorithm with random selection of unutilized virtual servers.

II. Related work

There are numerous challenges faced by the cloud computing, the most critical challenge is load balancing of the servers and resources in cloud environment. Since arrival of the pattern is not predictable and therefore dynamic algorithms are widely expressed by load balancer in cloud computing. Many research works have done in this field.

M.Randles et al. [12] investigated a decentralized honey-bee based load balancing algorithm that is nature inspired algorithm for self-organization. Performance of the system is enhanced with increased system diversity but throughput is not increased with an increase in system size. To overcome this problem Dinesh Babu and P. Vankata Krishna [1] have proposed nature based Honeybee algorithm for load balancing in virtual environment of cloud computing which aims to achieve maximum throughput by well balancing of tasks among virtual machines. Pham, Afify and Koc [2] have applied the Honeybee algorithm to solve the problem of cell formation in cellular manufacturing system. Pham Soroka, Chhambarzadeh, Koc, Otri and Packianather [4] have used Bees algorithm for the optimization of neural network to detect wood defects. Marinakis and Marinaki have studied the issues of the probabilistic travelling salesman which is the variation of the classic traveling salesman problem. Sunil Nakrani and Craig Tovey [9] worked on honeybee and Dynamic server allocation in internet hosting centers to satisfy request load and compare it against a omniscient optimal algorithm, a conventional greedy algorithm and an algorithm that computes omnisciently the optimal static allocation. Erik Cuevas, Denial Zakdivar, Marco Perez-Cisneros Huberto Sossa, Valentin Osuna [5] introduced a block matching for motion estimation based on Artificial Bee Colony. By consideration of fitness calculation, number of search locations are drastically reduced which indicates when it is feasible to calculate or only estimate new search locations. Guoqiu.Zha, Sam Kwong [11] proposed guided artificial bee Colony algorithm for numerical function optimization incorporating the information of global best (gbest) solution into the solution search equation to improve the exploitation. GABC algorithm consists of three different stages that are the employed bee stage, onlooker stage and the scout stage. The onlooker stage tends to select the good solution to further update, while both the employed bee stage and update every individual in the population but this algorithm overhead the computational complexity. Pei-Wei Tsai, Jeng-Shyang Pan, Bin- Yih Leao and Shu-Chau chu [8] introduce an Enhanced Artificial Bee Colony Optimization. In this algorithm, onlookers bee is designed to move straightly to the picked co-ordinate indicated by the employed bees and evaluate the fitness value near it in the original ABC algorithm in order to reduce computational complexity.

Practical Swarm Optimization is proposed by Ayed Salman, Iniwallah Ahmed [3] which combines local search method with global search method through neighborhood experience and attempt to balance exploration and exploitation. The algorithm is proposed for task assignment problem for homogeneous distributed computing system. This algorithm gives faster response with less complexity. Shagufta Khan, Nireesh Sharma [6] proposed Ant Colony Optimization for effective load balancing of nodes. ACO inspired from ant colonies that work together in foraging behavior. It works efficiently and gives better utilization of resources but increase overhead during runtime. Yun-Han Lee, Seiven Leu, Ruay-Shiung Chang [7] proposed an improved hierarchical load balancing algorithm to schedule jobs in Grid computing where it assign the fittest resource to jobs based on current status and compare the cluster’s load with the adoptive load on the system.

III. Overview of Honeybee Algorithm

A centralized internet hosting centers are containing an ensemble of commodity servers on which internet services are hosted for a fee [9]. In honeybee algorithm, the key concept is to allocating the servers among host customers of web services to maximize the revenue and better utilization of resources. Each server plays the role of either forager or scout and the advert board is where servers, successfully fulfilling the request, may post the adverts, with the probability p. A forager server may read the advert board according to their reading probability and scout server simply choose a random virtual server group. The server $S_i$ assesses its profit by comparing its profit $P_i$ with the overall hosting profit $P_{hosting}$ and adjust the probability of its reading the advert board accordingly. Initially, each server will choose random virtual server to serve request from, with service time determined by that virtual server. The server then decides whether to post an advert board for its virtual server depending on the generated profit probability. If a server reads the advert board, it will follow an advert and serve a request, entering a new foraging cycle otherwise it will revert to scout behavior to choose a random virtual server to service a request. The working of the honeybee is shown by the given algorithm-

Assume that a server $S_c\in V_c$ serving requests from service request queue $Q_c\in \mathbb{N}$, the value-per-request-served [10]. A server $S_c\in V_c$, on completion of each request from $Q_c$, will n
attempt with probability $p$ to post an advert on the advert board with duration $D = c_i A$, where $A$ denotes the advert scaling factor. Also, it will attempt with probability $r_i$ to read a randomly selected advert from the advert board as shown in Figure 3.2, if it is a forager or randomly select a $V_i, i: 0 ... (M – 1)$ if it is a scout. The probability $r_i$ is dynamic and changes as a function of the forager/scout server’s down revenue rate and hosting center’s overall revenue rate. The revenue rate, $P_i$, for a server $s_i$ is given by $P_i = \frac{C_i R_i}{T_i}$ where $R_i$ is the total number of requests served by a given server in the time interval $T_i$. The hosting center’s overall revenue rate, $P_{hosting}$, is given by $P_{hosting} = \sum_{s_j} ^{M} \frac{C_j R_j}{T_{hosting}}$ where $R_j$ denotes the total number of requests served by $V_j$ in the time interval $T_{hosting}$. A server $S_i \in V_j$ serving queue $Q_j$ determines its profitability by comparing the revenue rate $P_i$ with $P_{hosting}$. It updates $r_i$ according to the advert reading probability.

Initialization – $S_i \in V_j$ serving $Q_j$, Advert posting probability $p_{i}$, Advert reading probability $r_i$, Revenue rate interval $T_{pr}$, Advert reading interval $T_r$.

```c
while Qj Empty do
    serve request;
    If $T_{pr}$ expired then
        compute revenue rate;
        adjust $r_i$ from lookup table;
    If Flip($p_i$) == TRUE then
        Post Advert;
    If $T_r$ expired && Read($ri$) then
        /* randomly select an advert or a virtual server */
        Select advert( if forager) or Virtual Server $V_k$(if scout);
        Read advert id $V_j$(if forager);
    If $V_j \neq V_k$ then Switch($V_k$);
endwhile
endwhile
```

I.) Efficient Randomized Honeybee Algorithm

In the classical honeybee algorithm, if for a given service, the number of server necessary to serve the current demand is not fulfilled, then requests in the queue will spend time waiting to be served. This will increase the average service response time and may lose the revenue opportunity if a customer balks. Another main server allocation problem is that if, more servers are allocated than necessary to a service, the revenue rate will start degrading due to each server having to wait longer to find a request to serve. Over allocation may lead to increasing waiting time of server. To resolve the problem of excess allotment of servers, we propose the enhanced effective randomized honeybee algorithm. In this, once servers are allocated to the virtual servers, the over allotted virtual machines release their servers and then servers are reallocated to those virtual servers which are unutilized in compare to profitable virtual server. The allocation of those servers are performed on randomly selected unutilized virtual machine. The working idea of proposed algorithm is as follows-

After allocation of servers through honeybee algorithm the number of required server $Req_{server}$ load is calculated for each servicing VM and compare with the threshold value $TRS_{HIGH}$, if $Req_{Server}$ is high, then randomly select VM whose $Req_{Server}$ is less than $TRS_{LOW}$. $TRS_{HIGH}$ defines the highest threshold value of number of servers that could be allocated while the $TRS_{LOW}$ defines the minimum value of allocated servers. The new location where servers have to be allocated, is found by calculating newbee_position. To compare the profit, the cost of the servicing the request of the new virtual server must be calculated. If the cost is less than or equal to $avg_{cost}$, servers are allocated to that VM from overloaded VM. By transferring the servers whose revenue rate started decreasing, to the unutilized servers, the sever allocation is balanced amongst VM and their profit is also increased. The proposed randomized honeybee algorithm in the form of pseudo code is given below-

/*After excess number of server allocation to VMs, the profitable processor overloaded as compare to others while some VMs remains underutilized. This leads to wastage of processor time Check*/

```
for (i=1 to Max_iteration)
If ($Req_{server}$ ≥ $TRS_{HIGH}$) then
    Select VM($Req_{server}$ ≤ $TRS_{LOW}$);
    Perform Randomization on VM;
    Chose $k$ randomly not equal to $i$;
    $K = k$ [randi(1 nume 1 (k))];
    /*Newbee Position*/
    newbee_position = pop(i) + $\phi$*(pop(i).position - pop(k).position);
    Allocate servers to VM;
    newbee_Cost = CostFunction(newbee_position);
    /*Comparison*/
    If newbee_Cost <= $avg_{cost}(i)$ then
        pop(i) = newbee;
    else
        Switch($V_{k+1}$);
endfor
```

V.) Simulation and Results

The implementation of improved randomized honeybee is written and tested in MATLAB. For modelling and simulation of proposed algorithm in cloud computing infrastructure and services, CloudSim-3.0.3 framework is used. The programming language of CloudSim is JAVA. CloudSim provides system and behavioral modelling of the cloud computing component. Simulation of cloud environment and algorithm to evaluate performance can provide useful insights to explore such dynamics, massively distributed and scalable environment. The performance of the proposed algorithm is shown in some considerable metrics of load balancing.

A.) Resource Utilization Performance

An important parameter employed in this work to investigate the load balancing strategy of the projected rule is that the average resource utilization. Resource utilization is achieved by -

Resource Utilization = VM demand / range of tasks

Figure 1 shows the resource utilization rate of projected technique with random stealing that is comparatively high compared to existing honey bee load balancing technique. With randomization technique, tasks area unit purloined from a
random Virtual machine once a VM is idle. It therefore saves the idle time of the process parts within the Virtual machine.

Figure 1: Resource Utilization Performance

B. Response Time or Latency performance
The Figure-2 shows the response time in milliseconds for honeybee and enhanced randomized honeybee. The x-axis vary tasks and y-axis represent response time in milliseconds. The reduction of waiting time is beneficial in up the responsiveness of VMs. From this graph, it’s clear that basic honey bee technique in compare to improved honeybee is a lot of sensible and provides higher response. The virtual machines which remained unutilized and less responsive in honeybee, by proposed algorithm effectively respond and the response time of over loaded VMs is also reduced.

Figure 2: Response Time Performance

C. Idle Time Performance
Idle time is that the time between the days at that task is arrived on a virtual machine and time of task to be assigned to at least one sure virtual machine. The comparison is formed in terms of range of tasks and therefore the idle time and therefore the results area unit shown in Figure-3 of these results verifies that the proposed honey bee technique with randomization performs higher than the prevailing honey bee load balancing algorithm.

Figure 3: Idle Time Performance

V. Conclusion
We proposed a new enhanced randomize honeybee algorithm for reallocation of servers from over-loaded virtual machines to unutilized virtual machines. The problem that we are facing with the existing honeybee load balancing algorithms was that it was not able to perform well for underutilized processors in all the required areas of load balancing. For example, honeybee algorithm is good in the throughput with increased system size but its resource utilization is low and computation of profit on each node causes additional overhead which degrades the performance of the cloud. The basic idea of the proposed work is to dealloclote the servers from overloaded virtual machines and allocate them to underutilized VMs which are randomly selected. The proposed algorithm increases the resource utilization and overall performance of the system since balking behavior of severs and customers waiting in queue are reduced. Results to date support the effectiveness of the algorithm, particularly in the highly dynamic and unpredictable internet environment.

REFERENCES


