International Journal of Engineering and Technology Innovation, vol. 7, no. 4, 2017, pp. 292 - 307

Scalable Load Balancing Approach for Cloud Environment

Anurag Jain[1,*], Rajneesh Kumar[2]

[1] Department of Virtualization, School of Computer Science & Engineering, University of Petroleum & Energy Studies, Dehradun, Uttarakhand, India.

[2] Department of Computer Science and Engineering, Maharishi Markandeshwar University, Mullana, Ambala, Haryana, India.

Received 29 January 2017; received in revised form 18 April 2017; accepted 19 May 2017

Abstract

Cloud computing is a combination of parallel and distributed system which aims at effective resource utilization, providing uninterrupted services all the time which adapts itself with varying number of users without much capital investment. Ubiquitous, scalability and elasticity are some of the important features of cloud computing. To maintain essential characteristics, there is a need of mechanism which distributes the load efficiently among the available resources. Load balancing means the distribution of tasks among different available resources so that no one is over or under-utilized. Scalable, adaptable, efficient and reliable are some of the desirable features of a load balancing approach.

In this paper, authors have proposed a new load balancing approach named "Weighted Biased Random Walk" for the cloud environment using the concept of biased random walk. Weighted biased random walk approach has been analytically & experimentally analyzed. It has been compared with other load balancing approaches based on biased random walk found in literature. It has been found that weighted biased random walk approach is self-adjustable, distributed, dynamic, scalable and efficient in nature. It outperforms the other load balancing approaches based upon biased random walk. Collective presence of all the desirable features makes the weighted biased random walk approach perfect load balancing approach for the cloud environment.

Keywords: cloud computing, load balancing, biased random walk, scalable

1. Introduction

In 2009, Buyya et al. [1] have discussed the cloud evolution and how it has become the 5th utility of life. Computing represents a goal-oriented activity which includes the design and use of hardware and software for a wide range of activities. The list of activities is endless. It's meaning changes from one context to another. Parallel computing and distributed computing are means of using parallelism in computing to achieve higher performance. Cloud computing is the result of the evolution of mainframe, distributing computing, cluster computing, and grid computing. It is a combination of distributed and parallel interconnected systems which are dynamically providing resources through virtualization according to service level agreements finalized between user and cloud service provider. In 2014, Jain and Kumar [2] have discussed the different type of cloud computing services provided through the internet. Services can be in the form of storage, application, and hardware. The user can access services anytime anywhere without human interaction through the internet. A pool of resources is created. The user can access resources by creating their account and they have to pay as much they have used. In 2016, Khari et al. [3] have discussed how resources can be increased or decreased as per the requirement through virtualization. Virtualization of

* Corresponding author. E-mail address: anurag.jain@ddn.upes.ac.in
Tel.: +91-9466096567

resources has raised the issue of load balancing on cloud service provider end. This has raised the need of efficient load balancer which must do the optimized mapping between resources and tasks.

Organization of paper is as follows: In section 2, authors have covered the basics of load balancing. Section 3 gives the information about the literature review. In section 4, authors have discussed the proposed algorithm – "Weighted Biased Random Walk for load balancing. Section 5 gives the analytical analysis of weighted biased random walk algorithm. In Section 6, authors have described the simulation environment, results and characteristics of the weighted biased random walk algorithm. Conclusion and future scope have been given in section 7.

2. Load Balancing Overview

In 2012, Khiyaita et al. [4] have discussed the load balancing in cloud computing along with its algorithm type and different evaluation parameters. Load balancing means the distribution of tasks among different available resources so that no one is over or under-utilized. Resources can be the data center, physical machine, virtual machine or any application software.

2.1. Load balancing algorithm types

In 2012, Nuaimi et al. [5] have discussed the different types of Load balancing algorithms for the cloud environment. Major classifications of load balancing algorithms for cloud environment are as follows:

- Centralized & Static Approach
- Distributed & Dynamic Approach
- Mixed & Adaptive Approach

In centralized & static approach, single scheduler manages the distribution of load in the system. The logic of resource and task management is pre-defined and installed during the design of scheduling algorithm. An algorithm based on this approach, are suitable for small static and homogeneous environment. They can't match the requirements of the dynamically changing environment.

In distributed and dynamic approach, multiple schedulers manage the distribution of load in the system. Resource and task management logic is although predefined but it should change itself with the changing requirements of the environment. It is suitable for the dynamic and big environment.

In the mixed and adaptive approach, the scheduling algorithm is installed in multiple nodes at different levels and task management and resource management logic changes itself with the changing need of environment. Due to multilevel scheduling, logic becomes complex and processing gets slow but at the same time it increases the reliability and decreases the failure rate. This type of approach is suitable for the large, dynamic and heterogeneous system.

2.2. Load balancing metrics

In 2016, Jain and Kumar [6] have discussed the different quantitative and qualitative parameters that are considered for judging the performance of a load balancing algorithm in a cloud environment. They are as follows [6]:

- Response time: Time that a system takes to respond.
- Migration time: The time required for the transfer of a task from overloaded/failed machine to under-loaded/working machine. It should be least.
- Scalability: It is the ability of an algorithm to scale according to the requirement. How efficient system is, in handling the variation in demand, is a measure of scalability of the system.
- Throughput: Number of tasks completed in unit time interval is called throughput. It should be high.
- Fault tolerant: It is the capability of the load balancing algorithm to perform correctly in the situation of node/system failure. How algorithm is handling the situation of failure, how it is recovering from failure and how it is preventing the failure situation are some points which help in the judging of fault tolerance characteristic of any load balancing algorithm.

3. Literature Review

In this section, authors have analyzed the work done by other researchers in the area of load balancing in cloud computing. Authors have discussed the centralized and distributed load balancing approaches for cloud computing.

3.1. Centralized Approach

In 1998, Armstrong et al. [7] have discussed min-min and max-min approach of load balancing. These two algorithms don't follow first come first serve sequence rather it contains two criteria for task VM mapping:

- Execution time
- Completion time

In min-min, minimum execution time tasks are preferred over the maximum execution time tasks. It is decided on the basis of task size. Tasks are stored in the buffer. When the buffer fills completely, tasks are arranged in increasing order of the size and batch is processed. The concept chooses the task which holds minimum execution time and assigns it to the virtual machine which gives minimum completion time. Minimum completion time is estimated on the basis of VM power and no. of tasks in the queue of VM. It involves two minimum selection criteria, so it is called min-min approach.

In the max-min approach of load balancing, maximum execution time tasks are preferred before the minimum execution time tasks. Tasks are stored in a task allocation table till table fills completely. After this, task in the task-allocation table is sorted in the decreasing order of their size. Then the scheduler chooses the task which holds the maximum execution time. After this, virtual machine which will complete the task in minimum time get selected. Completion time is calculated on the basis of virtual machine capacity and number of tasks in the queue of virtual machine. It involves one maximum and one minimum selection criteria, so it is called max-min approach.

In 2011, Lu et al. [8] have discussed Join Idle Queue (JIQ) scheduling approach for load balancing. JIQ was realized using two level scheduling. To realize the concept of two levels of scheduling, authors have used the distributed scheduler. Multiple schedulers are used. Numbers of schedulers are very less in comparison to the number of virtual machines. Every scheduler will maintain a queue of idle virtual machines. At first level, idle VM is identified to be mapped with the task while at second level idle VM associates itself with any one of the randomly selected scheduler. On receiving a task, scheduler first consults its idle queue. If it finds any virtual machine, which is idle, then it immediately assigns the task to that virtual machine and removes that virtual machine from its idle queue. If it does not find any idle virtual machine, then it randomly maps the task with any VM. A virtual machine, after job completion, updates about its status to any of the randomly chosen idle queues associated with a scheduler. This approach separates the task of the discovery of idle servers from the task of job assignment to a virtual machine. Due to the use of multiple schedulers, this approach is distributed in nature. Failure of one scheduler does not cause the failure of the entire system.

In 2013, Xu et al. [9] have discussed Round Robin (RR) scheduling algorithm for load balancing in a cloud environment. The basis of this algorithm is the principle of time scheduling. The scheduler maintains a list of available virtual machines in a table known as VM allocation table. It assigns the tasks received through the data center controller to a list of virtual machines on a rotation basis. Scheduler initializes the current_vm variable with the id of the first virtual machine. It maps the received task with that VM whose id is stored in a current_vm variable. If the value of current_vm is equal to the id of last VM, it first initializes current_vm with the id of first and does the mapping otherwise it directly maps received task with that VM whose id is stored in a current_vm variable.

In 2013, Xu et al. [9] have discussed Minimum Completion Time (MCT) and Minimum Execution Time (MET) approach for load balancing. In both the approaches, tasks are assigned to resources in first come first serve manner.

In MCT, the virtual machine which takes less completion time for a given task is scheduled first. Completion time is estimated on the basis of VM power and number of tasks in VM queue. In the beginning, when no task is allotted to VM, then VM power is equal to its completion time. For assignment of a task to a virtual machine, MCT Scheduler accesses the VM allocation table. VM allocation table stores the virtual machine id, virtual machine power, number of tasks in queue and completion time of that virtual machine. This approach is dynamic in nature as it considers the current load of virtual machines.

In MET, the virtual machine which takes less Execution Time (ET) for a given task is scheduled first. Execution time is estimated on the basis of the processing capacity of virtual machines. MET Scheduler accesses the VM allocation table for mapping of the task with VM. VM allocation table stores the virtual machine id and virtual machine processing capacity. A virtual machine with more processing power can execute the task fast. So, this centralized load balancing approach is static in nature which neither considers the present load nor considers the task size.

In 2015, Tyagi and Kumar [10] have discussed Throttled load balancing strategy for cloud environments. Throttled load balancer uses a single job scheduler, which makes it centralized in nature. The job scheduler maintains a table named VM allocation table, which stores the id and status of all the virtual machines. A virtual machine can have only two states: occupied or idle, denoted by 1 or 0 respectively in the array. Initially, all virtual machines are idle. On receiving a task, job scheduler searches the virtual machine which is not busy. If it finds an idle virtual machine, then it assigns the task to that virtual machine. If no virtual machines are available to accept the job, then the task has to wait in job scheduler's queue. No queues are maintained at the virtual machine level. A virtual machine can accommodate only one task and another task can be allocated only when the current task has finished.

In 2015, Zaouch and Benabbou [11] have discussed Equally Spread Current Execution (ESCE) load balancing approach for cloud environments. This algorithm uses the spread spectrum approach. It works in such a way that the numbers of active tasks on each virtual machine are same at any time instant. The scheduler maintains VM allocation table which stores VM id and active task count on that VM. With the assignment of new tasks or on task completion, active task count corresponding to that VM in VM allocation table will be updated. In the beginning, active task count of each VM is zero. On arrival of the task, ESCE scheduler finds that VM whose active task counts is lowest. If more than one VM has lowest active counts, then VM which has been identified first is selected for task assignment. Task queues are maintained corresponding to each VM.

3.2. Distributed Approach

In distributed approach based load balancing approach, scheduling can be managed through multiple points. Failure of one point does not return in the failure of entire system. Implementation of distributed approach based scheduling techniques is complex but ideal for big distributed environment like cloud. Different heuristic based approach like honey bee, ant colony, genetic algorithm and random walk are distributed in nature. Discussion of different task scheduling techniques based on distributed approach are given below:

Honey Bee Based Algorithm: In 2004, Nakrani and Tovey [12] have discussed the honey bee approach for solving the optimization problem. Algorithms based upon honeybee approach are motivated by usual foraging activities of bees to discover the best option. During honey collection, bees are divided into two categories: Scout Bee and Foraging Bee respectively. Scout bee act as a navigator which are sent to search for an appropriate food source. Once they discover the appropriate food source, scout bee returns to the hive and do the waggle dance to notify foraging bee about the nectar site, its direction, the quality of food available at nectar site and distance of the hive from the nectar site. Foraging bees follow the path indicated by scout bees to collect the food from nectar site.

In 2013, Babu and Krishna [13] has proposed a honey bee foraging strategy based load balancing algorithm for cloud environments. Their algorithm performs in a dual manner. It not only performs the load balancing act but also considers the task priority while moving tasks from overloaded to the under-loaded machine. The tasks under migration act as a honey bee which globally updates the load information. Their load balancing algorithm improves the priority based balancing, throughput and response time.

Ant Colony Based Algorithm: In 2010, Xue et al. [14] have discussed the ant colony approach for solving optimization algorithm. It is a heuristic-based optimization method which is inspired by a biological system of ants. In food finding process, ants frequently move between the nest and a food source. Ants first move in a random manner. While finding a food source, ants leave special substance called pheromones on the ground. This pheromone guides other ants in finding the food source. After finding a food source, ants first analyze the quality and quantity of food source. While moving back to nest ants again leave pheromone on the ground which is an indicator of quality and quantity of food. Adaptability, parallelism, stochastic, positive feedback and autocatalytic are some of the inherent features of the ant colony based algorithm. Due to possession of stated features, ant colony behavior naturally suited to solve optimization algorithm.

In 2013, Gao et al. [15] have proposed a virtual machine to a physical machine mapping algorithm for optimum utilization of resources, power and load balance. Their algorithm was based on the ant colony approach. Authors have compared their approach with max-min ant system, bin packing algorithm, and multi-objective genetic algorithm approach. In their proposed approach, all parameters are initialized in the beginning and pheromone trails are set to 0. During execution, ant receives a VM request and need of VM host arises. Mapping is done on the basis of the concentration of pheromone concentration, which acts as a heuristic for VM host mapping. After every assignment pheromone is updated locally. When ant has completed the pheromone updating, after this the global pheromone updating is performed.

Genetic Algorithm Based Algorithm: In 1998, Mitchell [16] has discussed the basics of genetic algorithm search technique. A genetic algorithm is a heuristic based tool used to solve search and optimization problem which is based on survival of fittest theory given by Darwin. The concept of genetic algorithm has been derived from natural system evolution. While solving any problem through genetic algorithms, the problem is mathematically formulated in terms of the initial population and fitness value is assigned to each individual using fitness function. Then crossover and mutation operations are applied to individuals and the new population is generated. The fitness value of the new population is also generated and among the total individuals, only fittest individuals are selected for the next phase. The process is repeated till the most optimum solution of the problem is found.

In 2012, Gu et al. [17] have discussed resource scheduling approach for cloud environment based on genetic algorithm. Using previous data, the present state of the system and genetic algorithm approach, their approach computes the effect on the system after assignment of the resource and identify the least affected solution. So in this manner, it achieves the optimum load balancing solution and minimizes the dynamic migration.

Simulated Annealing: In 1995, Fleischer [18] has discussed simulated annealing technique for optimization. Simulated annealing is a search technique which is useful when the search space is discrete and used for searching the global optimization solution. It does not give the guarantee of a best optimal solution. Its name annealing has been inspired from techniques used in metallurgy in which material is heated and cools down in controlled amounts. This increases the crystal size and decreases the defects inside those crystals. Due to heat, atoms become free, change their positions and move randomly from a higher energy state. The slow cooling process helps atoms to find lower energy state relative to their last state. In simulated annealing technique, each point of the search space is representing a state of the physical system. Minimization of the objective function is equivalent to the energy of state in the system. The objective is to bring the system from starting state with minimum energy state.

In 2012, Zhan and Huo [19] have given a task scheduling algorithm for cloud environments. The authors have combined the concept of simulated annealing and particle swarm optimization. Through simulated annealing, they got the characteristic of higher convergence rate and through particle swarm optimization they got improved efficiency. The resultant approach is an improved particle swarm scheduling using simulated annealing. Through experimental results, the author has shown that their hybrid approach has reduced the average running time and increases the availability rate of resources. Their proposed hybrid approach also avoids the problem of local optima.

Random Walk: A random walk is a mathematical representation of a path which is composed of a succession of random steps. A random walk of length l on a graph G is a stochastic process with random variables Y_1, Y_2, … , Y_k such that $Y_1 = 0$ and Y_{m+1} is a vertex chosen randomly from neighbors of Y_m. In biased random walk, neighbors of Y_m are not chosen randomly but they are selected according to pre decided criteria [20].

In 2008, Rahmeh et al. [21] have proposed a load balancing framework for distributed network using the biased random walk. Authors have represented the situation through a virtual graph in which nodes of the graph have represented the servers. This virtual graph was self-organized in nature and it has used only local data for the discovery of resources and load distribution. The biased random walk was started from any randomly chosen node and biasing was achieved on the basis of remaining resources. Through simulation, they have shown their proposed framework as scalable and reliable. The authors have assumed their virtual machine as homogeneous in nature.

In 2010, Randles et al. [22] have given a comparative analysis of distributed load balancing algorithms for cloud environments. Authors have discussed honey bee based approach, biased random walk based approach and active clustering based load balancing approach for cloud environments. In their biased random walk based load balancing approach, they have used the virtual graph to represent the load on the server. Each node of the virtual graph represents a server and numbers of in-degree on a graph represents the available resources on that server. A biased random walk is started to search the highest in-degree node. Biasing was achieved through job token that has stored the information like no. of steps in random walk and node id having maximum in-degree. The value stored in the token helps in the formulation of the path during the biased random walk. Experimentally they have shown their approach as efficient in terms of throughput.

In 2012, Manakattu and Kumar [23] have proposed a load balancing algorithm which has implemented the biased random sampling. The parameters chosen by authors for implementation of biasing were:

- Queuing length
- Processing time

The authors have assumed that each node will maintain its neighbor's list, and nodes are heterogeneous in nature. Whenever a node receives a job, it creates a token. The token contains the queue length of the node having the shortest queue, the id of that node and walk length. The node forwards that token to one of its neighbors from the list of neighbors. Whenever another node receives a token, it compares its queue length with the queue length stored in the token. If it is less, then it updates otherwise no editing is done in the token. This process of updating and forwarding will carry on till walk length becomes log n. So, when token reaches the last node, then the node will forward the job towards that node whose id is stored in the token. Similarly, in place of queue length, processing time can also be used as a node selection parameter. Through simulation, the authors have shown the efficiency of the proposed approach in terms of response time, in comparison to other biased random sampling based approaches.

In 2014, Kumar and Agarwal [24] have presented a random graph based virtual machine migration model for the server network available in the data center. In their model, the authors have tried to balance the number of virtual machines available on different servers at run time by shifting the virtual machines from an overloaded server to under load server. Steps involved in their algorithm are as follows:

- Identify the overloaded server S_m.
- Select a subset of virtual machines which needs to be migrated.
- Identify the underloaded server S_n where migrated set of virtual machines can be deployed

In their model, the biased random walk will be started from an overloaded server in search of under load server. For selecting a subset of virtual machines for migration, authors have chosen the maximum correlation coefficient as a heuristic. For selecting a server for deploying migrated virtual machine, authors have chosen the migration opportunity and server capacity as a heuristic. Through simulation, the authors have shown that their algorithm finds a suitable server for migration of VM in minimum time. Also, they have shown that the average degree of their graph remains same throughout which indicates the proper load balancing in the network.

In 2015, Ariharan and Manakattu [25] have used the concept of the token to implement load balancing through the biased random walk. Each node actively participates in the biased random walk process by maintaining the information about a load of its neighboring node. This information is stored in the token and it is circulated during the biased random walk. A node after receiving a job selects the least loaded node of the neighboring node to continue the biased random walk. So, at the end of the biased random walk, token stores the information of least loaded node and the job is assigned to that node. This results in better response time and data processing time.

Table 1 Analysis of various approaches for load balancing

Approach	Centralized/ Distributed	Static / Dynamic	Concept	Feature	
				Pros	Cons
Honey Bee Approach [12]	Distributed	Dynamic	Waggle dance of scout bee helps in identify the best virtual machine.	• Ensures global load balancing through local load balancing. • Self-organizing in nature. • Performs well for a large diversified system.	• Does not utilize system resources effectively, This results in low throughput.
Ant Colony Approach [14]	Distributed	Dynamic	Pheromones dropped by ants are used in identify the best virtual machine.	• Adaptive, stochastic & parallel in nature. • Less chance of premature convergence. • Have autocatalytic positive feedback feature. • Shows better results relative to genetic algorithm and simulated annealing	• Convergence rate is slow relative to other. • It's difficult to analyze it theoretically. • After every iteration probability distribution changes.
Genetic Algorithm [16]	Distributed	Dynamic	Crossover, mutation and fitness function helps in identifies the best virtual machine.	• Adaptive, stochastic & parallel in nature. • Simple and easy to implement. • Stable and efficient in finding a global optimum solution.	• Does not give a guarantee of a global optimum solution. • Shows the problem of slow and imperfect convergence in a large system.
Biased Random Sampling [20]	Distributed	Dynamic	Random walk based upon biasing parameter helps in identifies the best virtual machine.	• Always ensures global load balancing. • Self-organizing in nature. • Easy to implement • Better utilization of resources results in higher throughput.	• By the increase of load or population diversity performance degrades.

Table 1 shows the comparative analysis of various approaches for load balancing in a cloud environment. After thoroughly reviewing the literature, it was found some essential features for load balancing technique were missing in the existing work. Those features are as follows:

- Distributed: Existing techniques are partially distributed in nature.
- Self-Adaptable: Existing techniques are not self-adaptable. They are using some sort of tokens to share information among virtual machines.
- Scalable: Existing techniques don't have the capability to handle varying requirement effectively.

Lack of above-described feature in existing load balancing approaches has motivated to propose a fully distributed, self-adaptable and efficient load balancing approach for cloud environments.

4. Proposed Algorithm - Weighted Biased Random Walk

Load balancing scenario has been represented in the form of directed graph G = (V, E).

V = Set of existing virtual machines

E = Set of an existing group of edges.

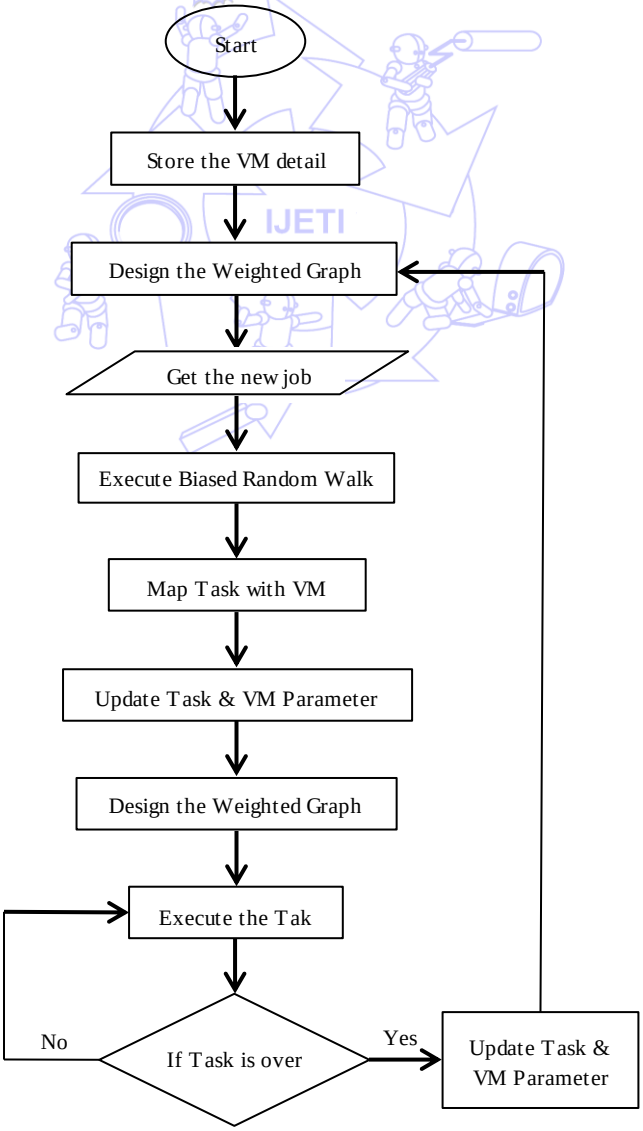


Fig. 1 Flowchart of Weighted Biased Random Walk

The capacity of virtual machines is not identical. All virtual machines are labeled with their weight. The weight of the virtual machine is calculated using a number of tasks in queue, their size, and resources available to virtual machines. An outgoing edge from a virtual machine (node) X to Y will be there if the weight of node X is less than the weight of node Y. Whenever a new task arrives in a data center, **Weighted Biased Random Walk** starts it's working. It selects a node randomly and starts the biased random walk. The biased random walk will end at the highest weighted node. The task is mapped with that virtual machine (node) where biased random walk has ended. After this weight of virtual machine (node) and graph connectivity is updated. On task completion, the weight of virtual machine (node) and graph connectivity is re-updated. Fig. 1 shows the flowchart representation of Weighted Biased Random Walk.

Weighted_Biased_Random_Walk ()

{

***Step 1**: Discover the existing virtual machine and find out their specification. It includes RAM, Storage capacity, No of processors, Processing speed, Queue length and Task Size.*

***Step 2:** Compute the Processing Capacity of all virtual machine using the following formula: Processing Capacity = [{(.5) * (No of processors) * (Processing Speed) / (Maximum Processing Speed)} + {(.4) * (Ram Size) / (Max Ram Size)} + {(.1) * (Storage capacity) / (Max Storage Capacity)}]*100000*

***Step 3**: Compute the weight of all virtual machine (node) in the graph.*

If queue length is null, then Weight=Processing Capacity.

Else Weight = Processing Capacity/Size of all tasks in the queue of that node (virtual machine).

***Step 4**: Use the Weighted Biased Random Walk to design the virtual graph (An outgoing edge from a virtual machine (node) X to all those virtual machines (node) Y will be there whenever the weight of node X is less than the weight of node Y).*

***Step 5**: Whenever a new task arrives, assign it Task id and store its size and arrival time.*

***Step 6**: Use the Weighted Biased Random Walk to map task with a virtual machine (It selects a node randomly and starts biased random walk which will end at the highest weighted node).*

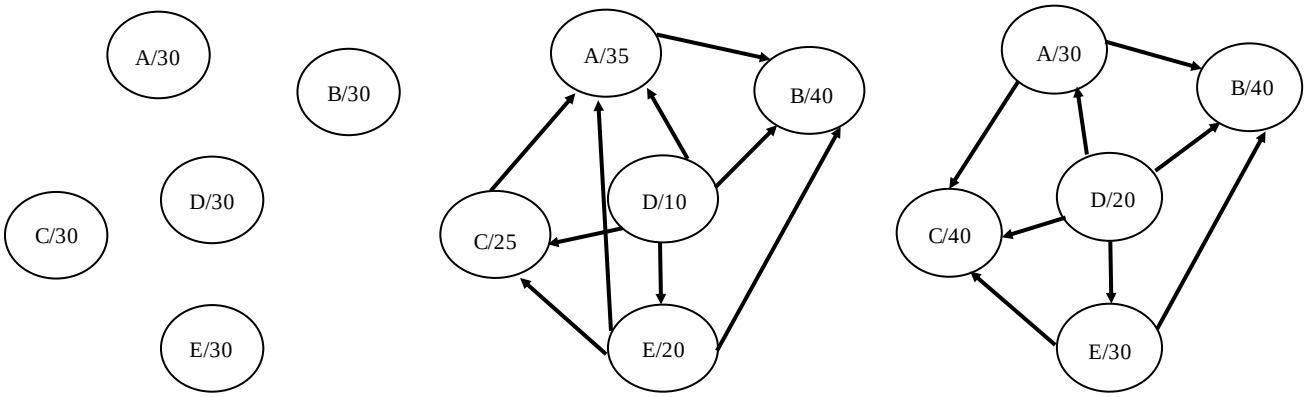
***Step 7**: On receiving a new task, the node updates its weight and this also results in a change of edge connectivity in the graph.*

***Step 8**: On task completion, node re-updates its weight and this again results in a change of edge connectivity in the graph.*

}

5. Analytical Analysis

In 1959, Erdos and Renyi [26] have discussed the properties of graphs. Consider a graph having N vertices, in which each vertex is corresponding to a virtual machine. Virtual machines can be homogeneous or heterogeneous in nature. Then, a maximum number of possible edges in the graph will be N*(N-1)/2. In degree of any node, Y will represent the number of nodes having a weight less than Y. Out-degree of any node Y will represent the number of nodes having weight more than Y. Total Degree (sum of in degree & out degree) of each node will lie between 0 to (N-1). The probability of ending a random walk at a virtual machine (node) is directly related to the distribution of nodes in degree & out-degree. The analysis of walk length in different cases of weighted biased random sampling approach is below.



(a) Maximum possible path length is 0 (b) Maximum possible path length is 4 (c) Maximum possible path length is 2

Fig. 2 Graphical Representation of three different scenario for N=5

Best Case: Capacity of all virtual machines is identical. Then, graph will be a collection of disconnected vertices and random walk length will be zero. Fig. 2 (a) shows the best case.

Worst Case: Capacity of all virtual machines is different. Then, there will be at most N*(N-1)/2 number of edges in the graph and random walk length will be at most (N-1). Fig. 2 (b) shows the worst case.

Average Case: When some nodes have the same capacity and some have different capacity, random walk length will be (N-1)/2. Fig. 2 (c) shows the best case.

So, it can be concluded that on average after a random walk of N/2 length, the Weighted Biased Random Walk will always find the suitable virtual machine. So, time complexity of finding the suitable virtual machine is of linear order which makes the proposed approach efficient in nature.

In the Weighted Biased Random Walk, random walk terminates on a node whose weight is highest in the walk. This ensures the selection of best available virtual machine and improves the performance of the system. On receiving a task, nodes weight will decrease and accordingly in degree and out degree will change. Also, on task completion, nodes weight will increase and accordingly in degree and out degree will change. This regular updating of nodes degree will maintain its degree near to average degree. This makes the graph a regular graph which makes the system stable.

6. Results & Analysis

6.1. Simulation Environment

To analyze the weighted biased random walk approach, simulation environment has been developed using JDK 1.6 and Netbeans IDE 6.0.1. The user can set the number of tasks, their size and tasks arrival time through the task configuration interface. Fig. 3 shows the task configuration interface. The minimum and maximum value of MI range indicates the size of randomly generated tasks in millions of instructions per second. The minimum and maximum value of time range indicates the range of inter-arrival time i.e. time gap after which next task will arrive. Information related to the arrival time, task size and task id are stored as a string in a file.

Fig. 4 shows the interface for setting the virtual machine configuration. It includes no. of processors, their processing speed, RAM, bandwidth, secondary storage. Using this interface, single data center with specified number of pre-configured machines has been developed. Using the formula specified in step 2 of weighted biased random walk algorithm given in section 4, the capacity of each machine is calculated in the beginning. Fig. 5 shows the interface which shows the capacity of each machine. Information related to VM id and its capacity are stored as a string in the file.

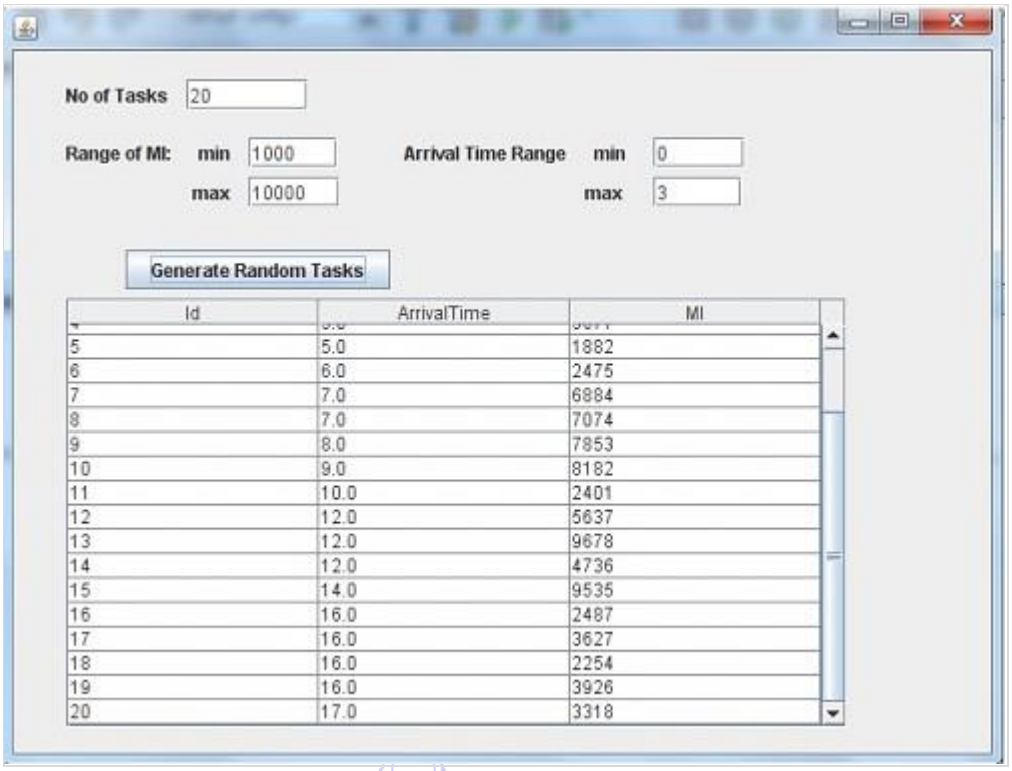


Fig. 3 Task configuration interface

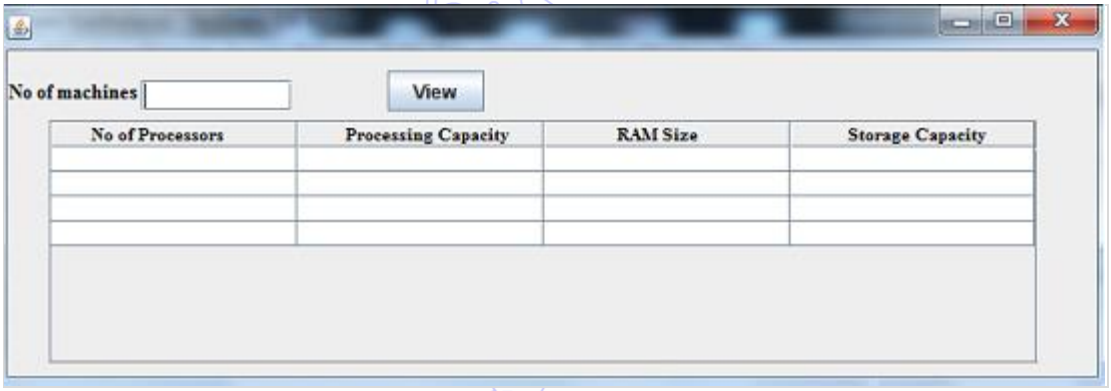


Fig. 4 Virtual Machine Configuration Interface

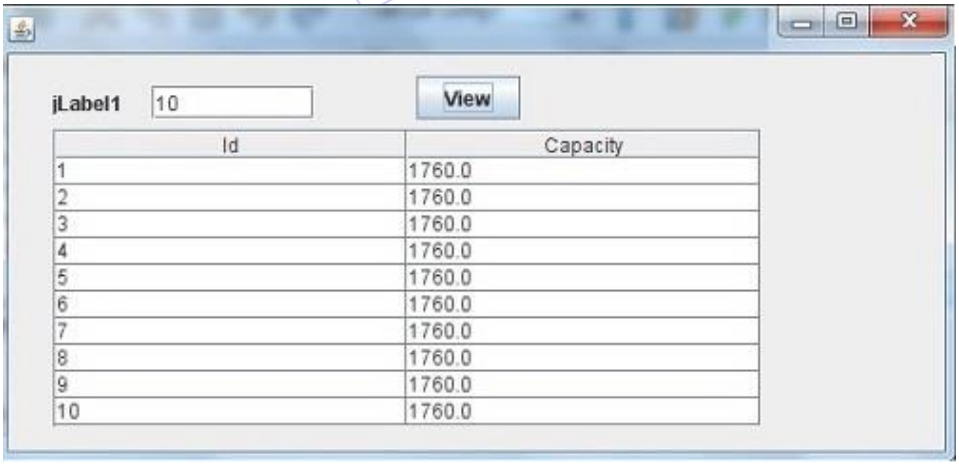


Fig. 5 Capacity of each Machine

Once parameters of tasks and VM have been configured then using the scheduler interface, the simulation will be started. Tasks are arrived according to the arrival time stored in the file and tasks are mapped with the VM according to weighted biased random walk. Weighted biased random walk has been compared with the other approaches based upon biased random walk under the same task set and virtual machine set.

6.2. Results

Following are the graphical results which have been obtained when experiments have been done for the same set of a virtual machine with varying number of tasks and their size. No. of machines are set 10 with same capacity and no of tasks are varying from 100 to 500 with randomly varying capacity and arrival time. Weighted Biased Random Walk will start from the randomly selected node. So, there is no centralized unit for task virtual machine mapping. This makes the proposed approach **distributed** in nature. Whenever a node fails, its weight will become zero and graph will be updated accordingly.

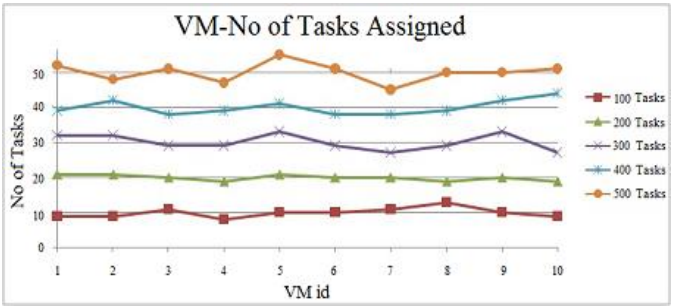


Fig. 6 Task distribution among 10 VM

From Fig. 6, it can be concluded that distribution of load is always even. The increase in a number of tasks does not affect the performance of Weighted Biased Random Walk. So, it can be concluded the weighted biased random walk is **dynamic** and **stable** in nature.

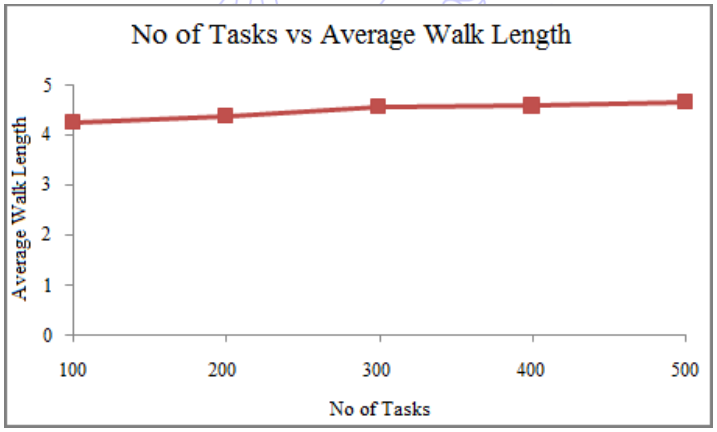


Fig. 7 Average walk length for different number of tasks

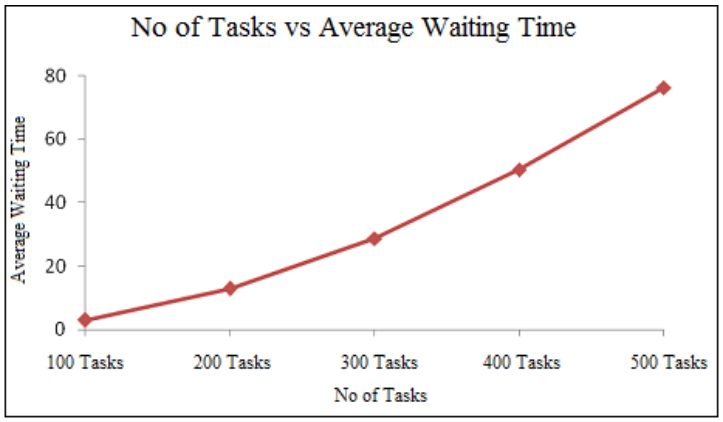


Fig. 8 Average waiting time for different number of tasks

It has been identified through analytical analysis done in section 5, that on average after N/2 no of moves Weighted Biased Random Walk will identify the best virtual machine. From Fig. 7, it has been observed that average walk length in case

of 10 nodes is lying between 4 and 5. Therefore, simulation result has supported the analytical analysis. From Fig. 8, it has been observed that with the increase of a number of tasks, average waiting time is not increasing exponentially. It is increasing gradually. This makes the weighted biased random walk efficient in nature.

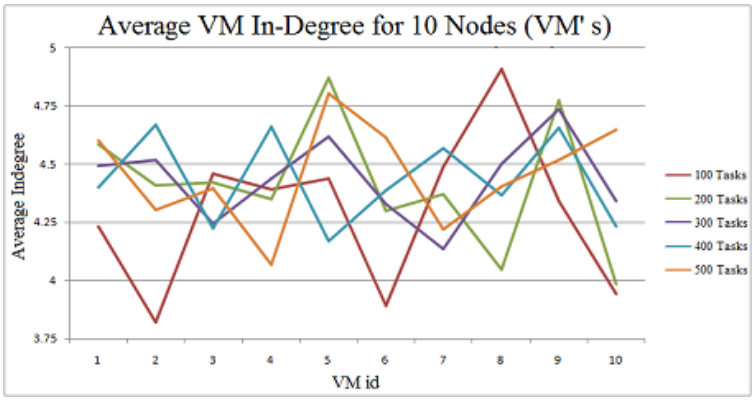


Fig. 9 Average VM in-degree for different task set among 10 VM

Analytical analysis done in section 5 has shown that graph formed during the implementation of Weighted Biased Random Walk is a regular graph. Moreover, it was not affected by variation of tasks and also number of nodes can be increased or decreased according to requirement. Simulation results have supported the analytical analysis. From Fig. 9, it can be re-concluded that in weighted biased random walk, average in-degree of all nodes is nearly same. Also from Fig. 10, it can be concluded that there is very small variation in in-degree with a change of a number of tasks. Therefore, it can be concluded that weighted biased random walk is scalable in nature.

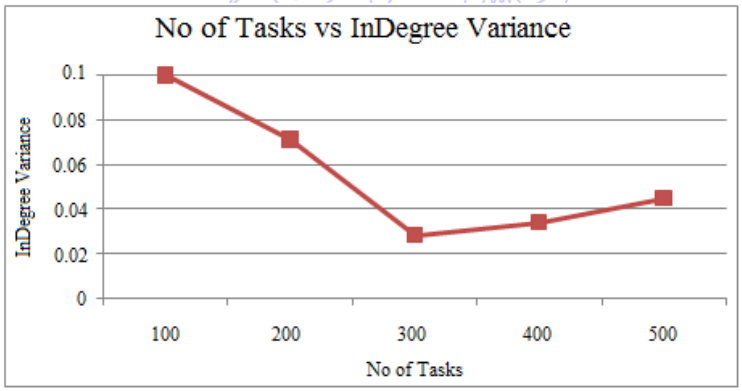


Fig. 10 In-degree variance for different task set among 10 VM

6.3. Comparison with other biased random walk based approach:

Table 2 shows the comparison of a weighted biased random walk with other variants of biased random walk proposed by other researchers:

Table 2 Comparison of proposed approach with other variants of biased random walk

Biased Random Sampling [21]:	Biased Random Sampling based on Queue Length [22]:	Biased Random Sampling based on Processing Time [23]:	Weighted Biased Random Sampling (Proposed Approach)
Load balancing scenario is represented by graph.	Load balancing scenario is represented by graph.	Load balancing scenario is represented by graph.	Load balancing scenario is represented by graph.
Nodes are connected randomly.	Nodes are connected randomly.	Nodes are connected randomly.	There will be an edge from a virtual machine (node) X to Y iff the weight of node X is less than the weight of node Y.
Threshold limit of random walk length is log n where n is a number of nodes.	Threshold limit of random walk length is log n where n is a number of nodes.	Threshold limit of random walk length is log n where n is a number of nodes.	Threshold limit of random walk length is n-1 where n is a number of nodes.

Table 2 Comparison of proposed approach with other variants of biased random walk (continued)

Biased Random Sampling [21]:	Biased Random Sampling based on Queue Length [22]:	Biased Random Sampling based on Processing Time [23]:	Weighted Biased Random Sampling (Proposed Approach)
The weight of a node is based on the available number of resources.	The weight of a node is based on the queue length of the node.	The weight of a node is based on the no of tasks in the queue and their size.	The weight of a node is based on available resources, no of tasks in the queue and their size.
No token is rotated.	The token is rotated among the nodes which store the walk length and id of least weight node found during a random walk.	The token is rotated among the nodes which store the walk length and id of least weight node found during a random walk.	A centralized database containing information about the weight, in degree, out degree and node connectivity is maintained.
Next node during execution of random walk is selected randomly.	Next node during execution of random walk is selected randomly.	Next node during execution of random walk is selected randomly.	Next node during execution of random walk is selected based upon the weight of the node.
Does not give the guarantee of selection of a best available node.	Does not give the guarantee of selection of a best available node.	Does not give the guarantee of selection of a best available node.	Gives the guarantee of selection of a best available node.

Table 3, 4 and 5 show the comparison of a weighted biased random walk with other variants of biased random walk on waiting time, completion time and throughput parameter respectively. It has been found from the analysis of Table 3, 4 and 5 that weighted biased random walk always outperforms the other variants on all parameter.

Table 3 Comparison of proposed approach with other variants of biased random walk on waiting time

Approach	**Average Waiting Time (ms) with 10 virtual machines**				
	100 Tasks	200 Tasks	300 Tasks	400 Tasks	500 Tasks
Biased Random Sampling [21]	3.6444	13.3058	30.9567	52.4491	77.8865
Biased Random Sampling based on queue length [22]	3.3439	13.6245	29.5771	51.6249	77.1236
Biased Random Sampling based on processing time [23]	3.4479	14.644	29.9567	53.2535	78.719
Proposed Approach (Weighted Biased Random Sampling)	2.9956	12.953	28.7442	50.3804	76.2537

Table 4 Comparison of proposed approach with other variants of biased random walk on completion time

Approach	**Average Completion Time (ms) with 10 virtual machines**				
	100 Tasks	200 Tasks	300 Tasks	400 Tasks	500 Tasks
Biased Random Sampling [21]	4.68711	15.3745	33.9719	56.348	82.6011
Biased Random Sampling based on queue length [22]	4.3866	15.6933	32.5923	55.5239	81.8382
Biased Random Sampling based on processing time [23]	4.4906	16.7127	32.9719	57.1524	83.4335
Proposed Approach (Weighted Biased Random Sampling)	4.0383	15.02176	31.7594	54.2793	80.9682

Table 5 Comparison of proposed approach with other variants of biased random walk on throughput

Approach	**Average Throughput with 10 virtual machines**				
	100 Tasks	200 Tasks	300 Tasks	400 Tasks	500 Tasks
Biased Random Sampling [21]	5.26	4.1	2.1	1.9	1.68
Biased Random Sampling based on queue length [22]	6.19	3.61	2.6	2	1.76
Biased Random Sampling based on processing time [23]	6.05	3.21	2.6	1.6	1.69
Proposed Approach (Weighted Biased Random Sampling)	8.45	4.2	3.06	2.3	1.91

7. Conclusions

In this paper, authors have proposed a new load balancing approach named **weighted biased random walk** for the cloud environment using the concept of random walk. Weights are assigned to all virtual machines by considering the parameters

such as the resources allotted to the virtual machine, the number of tasks assigned to a virtual machine and their size. Random walk for the search of the virtual machine will start randomly from any machine and it will end at that virtual machine where weight is the highest among all the virtual machines. In this way, every task is mapped with the best available virtual machine. Through analytical analysis and simulation results, it has been proved that the weighted biased random walk is adaptive, efficient, distributed, dynamic and scalable in nature. It also outperforms the other load balancing approaches based upon the biased random walk. Collective presence of all the desirable features makes the weighted biased random walk approach perfect load balancing approach for the cloud environment.

As a future scope, authors have planned to implement the weighted biased random walk for multiple data center and analyze the effect of bandwidth and delay. Also to increase the user's trust in the capability of cloud, authors have planned to incorporate fault tolerance and fault recovery mechanism in a weighted biased random walk.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, June 2009.

[2] A. Jain and R. Kumar, "A taxonomy of cloud computing," International Journal of Scientific and Research Publications, vol. 4, no. 7, pp. 1-5, July 2014.

[3] M. Khari, S. Gupta and M. Kumar, "Security outlook for cloud computing: A proposed architectural-based security classification for cloud computing," Proc. IEEE Conference. Computing for Sustainable Global Development (INDIACom), March 2016, pp. 2153-2158.

[4] A. Khiyaita, H. El Bakkali, M. Zbakh, and D. El Kettani, "Load balancing cloud computing: state of art," Proc. IEEE Conf. Network Security and Systems (JNS2), April 2012, pp. 106-109.

[5] K. Al Nuaimi, N. Mohamed, M. Al Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: challenges and algorithms," Proc. IEEE Symp. Network Cloud Computing and Applications (NCCA), December 2012, pp. 137-142.

[6] A. Jain and R. Kumar, "A multi stage load balancing technique for cloud environment," Proc. IEEE Conf. Information Communication and Embedded Systems (ICICES), February 2016, pp. 1-7.

[7] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," Proc. IEEE Conf. Heterogeneous Computing Workshop (HCW 98), March 1998, pp. 79-87.

[8] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus and A. Greenberg, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services," Performance Evaluation, vol. 68, no. 11, pp. 1056-1071, November 2011.

[9] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud," Tsinghua Science and Technology. vol. 18, no. 1, February 2013, pp. 34-39.

[10] V. Tyagi and T. Kumar, "ORT broker policy: reduce cost and response time using throttled load balancing algorithm," Proc. Conf. Intelligent Computing, Communication and Convergence (ICCC), 2015, pp. 217-221.

[11] A. Zaouch and F. Benabbou, "Load balancing for improved quality of service in the cloud," International Journal of Advanced Computer Science and Applications, vol. 6, no. 7, pp. 184-189, 2015.

[12] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," International Society for Adaptive Behavior, vol. 12, no. 3, pp. 223-240, December 2004.

[13] D. Babu and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing, vo. 13, no. 5, pp. 292-2303, May 2013.

[14] X. D. Xue, B. Xu, H. L. Wang, and C. P. Jiang, "The basic principle and application of ant colony optimization algorithm," Proc. IEEE Conf. Artificial Intelligence and Education (ICAIE), October 2010, pp. 358-360.

[15] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," Journal of Computer and System Sciences, vol. 79, no. 8, pp. 1230-1242, December 2013.

[16] M. Mitchell, An introduction to genetic algorithms. MIT press; 1998.

[17] J. Gu, J. Hu, T. Zhao, and G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment," Journal of Computers, vol. 7, no. 1, pp. 42-52, January 2012.

[18] M. Fleischer, "Simulated annealing: past, present, and future," Proc. IEEE Conf. Winter Simulation, December 1995, pp. 155-161.

[19] S. Zhan and H. Huo, "Improved PSO-based task scheduling algorithm in cloud computing," Journal of Information and Computational Science, vol. 9, no. 13, November 2012, pp. 3821-3829

[20] "Random Walk," https://en.wikipedia.org/wiki/Random_walk, October 12, 2016.

[21] O. A. Rahmeh, P. Johnson, and A. Taleb-Bendiab, "A dynamic biased random sampling scheme for scalable and reliable grid networks," INFOCOMP Journal of Computer Science, vol. 7, no. 4, pp. 1-10, December 2008.

[22] M. Randles, O. Abu-Rahmeh, P. Johnson, and A. Taleb-Bendiab, " Biased random walks on resource network graphs for load balancing," The Journal of Supercomputing. vol. 53, no. 1, pp. 138-162, July 2010.

[23] S. S. Manakattu and S. D. Kumar, "An improved biased random sampling algorithm for load balancing in cloud-based systems," Proc. ACM Conference. Advances in Computing, Communications, and Informatics, August 2012, pp. 459-462.

[24] N. Kumar and S. Agarwal, "Self-regulatory graph-based model for managing VM migration in cloud data centers," Proc. IEEE Conf. Advance Computing Conference (IACC), February 2014, pp. 731-734.

[25] V. Ariharan and S. S. Manakattu, "Neighbor aware random Sampling (NARS) algorithm for load balancing in cloud computing," Proc. IEEE Conf. Electrical, Computer and Communication Technologies (ICECCT), March 2015, pp. 1-5.

[26] P. Erdos and A. Renyi, "On random graphs I," Publicationes Mathematicae, vol. 6, pp. 290-297, 1959.