International Journal of Computer Applications (0975 – 8887)
Volume 122 – No.9, July 2015

Load Balancing Approach in Cloud Computing using Improvised Genetic Algorithm: A Soft Computing Approach

Garima Joshi
CSE Dept. GBPEC,
Ghurdauri, U.K, India

S. K. Verma
CSE Dept. GBPEC,
Ghurdauri, U.K, India

ABSTRACT

The concept of Cloud computing has significantly changed the field of parallel and distributed computing systems. The major issues to the cloud are resource discovery, fault tolerance, load balancing, safety measure, task scheduling, dependability, data backup, and data portability. Load balancing is one of the essential responsibilities of the cloud computing. In current situation, the load balancing algorithms built should be very efficient in allocating the request. It also ensures the usage of the resources in an intelligent way so that underutilization or overutilization of the resources does not occur in the cloud environment. In this paper, a soft computing based load balancing approach has been proposed called Improvised Genetic Algorithm (IGA), for allocation of incoming jobs to the servers or virtual machines (VMs). The proposed algorithm considers the cost value as a fitness function, of an individual node while performing load balancing. The proposed strategy has been simulated using MATLAB toolkit.

Keywords

Cloud Computing, Data Center, Genetic Algorithm, Load Balancing, Response Time

1. INTRODUCTION

Typically, the Internet is depicted in network diagram as a cloud [1]. The term “cloud” [2] grew up from the domain of telecommunications when suppliers began utilizing virtual private network (VPN) services for data communications. Cloud computing merely intends as Internet Computing, the Internet is seen as an accumulation of clouds. Thus, the cloud computing is outlined as employing the internet to render technology-enabled services to the people and organizations [3]. The general view of cloud in fig 1.

The concept of cloud computing has significantly changed the field of parallel and distributed computing systems [4]. It has emerged as a modern solution to provide cheap and easy access to externalized IT resources [5]. Cloud computing addresses with virtualization, scalability, interoperability, caliber of service and the delivery models, namely public, private and hybrid [2]. Thus in spite of divine prospective of Cloud Computing, much vital concern still needs to be examining for its complete acceptance. One of these issues is Load balancing.

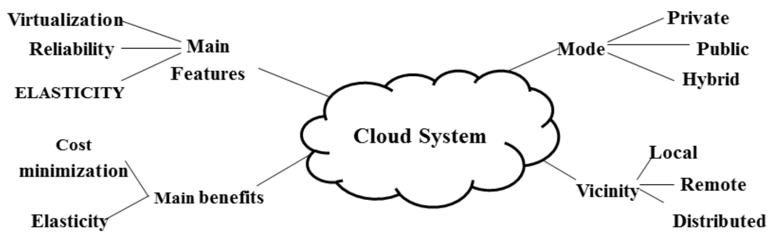

Figure1: Overview of Cloud

Load balancing, as the name depicts, is a proficiency that allows workloads to be disseminated across numerous resources, to make effectual resource employment. It also enhance response time by dealing a situation in which some of the nodes are extremely loaded while some others are under loaded. The intention of load balancing is to optimize usage and throughput while cutting down the reaction time. As on-demand in service model, weight comes arbitrarily or dynamically in cloud computing surroundings, which induces some VM/servers to be loaded largely, while others idle or gently loaded. This successively contributes to pitiful performance and make user unsatisfied.

Therefore, if we distribute the weight in a proper way, it will improve system performance, throughput, response time, etc. to cope up user expectation. The crucial matters to conceive while formulating such algorithm are approximation and equivalence of load, consistency, and functioning of the different system, node selection and many other ones. Load balancing mechanisms can be categorized or static, and periodic or non-periodic.

The introduced algorithm is an improvement over existing Genetic Algorithm (GA) based load balancing strategy for cloud computing [6] called Improvised Genetic Algorithm (IGA). It has considered the fitness function of each task before scheduling it to a particular node for load balancing in the cloud environment. So, that load is equally distributed among various processing nodes.

The proposed algorithm (IGA) is a soft computing approach, which uses a mechanism of natural selection strategy. The remaining paper is prepared as follow: Section 2 gives an overview of previous researchers in the field of load balancing in the cloud environment. Section 3 discusses IGA algorithm for load balancing. Section 4 presents the simulation results, and its analysis with an overview of MATLAB toolkit and section 5 concludes the paper.

2. RELATED WORK

Intensive explore has been carried on over the past decades in cloud load balancing area to solve the problem of distributing load among various nodes present in the cloud so that every node is utilized correctly. It has been proved that load balancing is an NP-complete problem. In this section, we discuss existing load balancing techniques in cloud computing. Here we classify load-balancing algorithm in two main types that are Static load balancing and Dynamic load balancing.

B. Sotomayor et al. [7] introduced a static well-known load balancing technique called Round Robin (RR), in which all processes are distributed among all applicable processors. The allotment sequence of operations is preserved locally, which is autonomous of the allocation of the distant processor. In this approach, the demand is sent to the node bearing minute connections, and as a consequences of this at some point of stage, some node may be massively burdened and other remain idle.

This complication was figure out by S.C. Wang [8] by presenting a dynamic load-balancing algorithm called Load Balancing Min-Min (LBMM) techniques that are based on three- level frameworks. In this, each node is given a proper opportunity that keeps each node busy in the cloud without considering execution time of the node. First layer request manager that is responsible for receiving task and assigning it to one service manager to the second level. On acquiring application, service manager segregates it into subtasks. After that, the service manager will assign subtask to the service nodes to execute the task.

Further, B. Radojevic in [9] proposed a static load-balancing algorithm called CLBDM (Central Load Balancing Decision Model). It is an enhancement of round robin technique. RR is enhanced and in CLBDM, the estimation of network point between the applicant and the node is done and if the connection period rose above the threshold then complication raises. If the problem appears, then the relationship between the client and node is adjourned, and the task is dispatched to another node using RR law.

Lars Kolb in [10] proposed the Map Reduce Based Entity Resolution Load Balancing technique that is based on large datasets. Here two main tasks are done, Map function and Reduce task. For mapping work, the first PART method is executed where the request entity is partitioned into segments, and entirely similar individuals are grouped by GROUP method and by applying, REDUCE function. Map task reads the objects in parallel and process them so that overburden of the work is shrunk.

Further, J. Hu in [11] introduces a static scheduling strategy for load balancing in virtual machine resources. This technique considers the historical data and current state of the organization. Here, essential scheduler and resource supervisor is used. The scheduling organizer analyses the availability of resources to operate a task and assign the same.

Al-Jaroodi in [12], proposed a dynamic load balancing technique named DDFTP (Dual Direction Downloading Algorithm from FTP server). Here files of size m are divided into the m/2 partition. For example, if one server starts from zero to incremental order then other will begin from m to decremented order independently of each other. As on downloading two consecutive blocks the task is considered as finish and assign next task to the server. Because of reduction in the network, communication to the server between client and nodes network overhead is reduced.

Nishant et al. in [13], further extend the work and introduced Ant Colony Optimization (ACO) technique for load balancing in the cloud environment. Here ant initiates the movement as the request is initiated. This method uses the ant action to gather knowledge of cloud node to designate the task to the appropriate node. In this performance, once the demand is admitted, the ant and pheromone start the forward movement in the pathway from the head node. The ants move in the forward direction from the overloaded load looking for next node to check whether it is overloaded node or not. Now if ant, finds under-loaded node still it moves in the forward direction in the path. In addition, if it finds the overloaded load then it starts the backward movement to the last under-loaded node it found previously. Here, if ant found the target node ant will commit suicide, so that it will prevent unnecessary movement.

M. Brototi et al. in [14], presented a Stochastic Hill Climbing based approach on soft computing for solving the optimization problem. It solves the problem with high probability. It is a simple loop moving in a direction of increasing values, which is uphill. Moreover, this makes minor changes in the original assignment. According to some criteria, designed One is candidate generator to arrange feasible successor and the other is interpretation principle, which ranks each logical solution. M. Randles et al. [15], proposed a honeybee behavior inspired load balancing (HBB-LB) technique that help to achieve even load balancing across virtual machines to maximize throughput. It considers the priority of the task is waiting in the queue for execution in the virtual machine. After that workload on VM calculated decides whether the system is under-loaded, overloaded or balanced. In addition, based on this VM are grouped.

The above analysis on various load balancing strategies present some of their advantages and disadvantages and may prompt others to work further on improving and designing the scheduling strategies for load balancing in the cloud environment. No load-balancing model discussed above that takes into consideration of minimization cost function and to improve the response time of the task. In this paper, we consider fitness function of each task before scheduling them to a particular processing node. The proposed algorithm improves the overall response time of task while scheduling them in different VM.

3. PROPOSED ALGORITHM

This section describes the proposed algorithm with a brief introduction.

Genetic Algorithm

This technique requires a coding scheme that can represent all legal solutions to the optimization problem. The general scheme of the genetic algorithm is given in figure 2. The algorithm starts with a set of solutions called population and represented by chromosomes.

The progression generally drawn from a population of randomly achieve individuals, and is a monotonous process, with the people in each recurrence is called a generation. In each generation, the fitness of each person in the population is estimated; the fitness is usually the rate of the objective function in the optimization issue being determined.

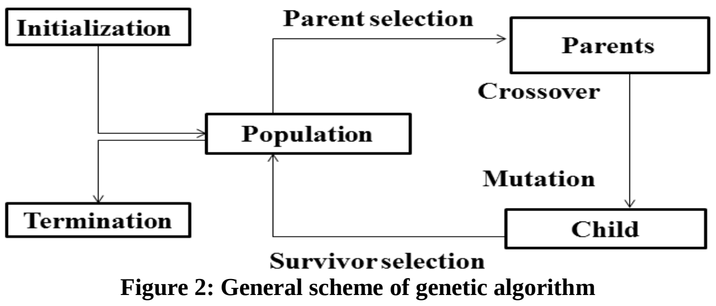

Figure 2: General scheme of genetic algorithm

The fit individuals are stochastically selected from the current population, and particular person genome is altered to form a recent generation. The new generation of candidate solutions is used in the adjoining repletion of the algorithm. More often, either the algorithm wraps up when a maximal figure of generations is produced, or adequate fitness level is attained for the population. The Improvised Genetic Algorithm Load-Balancing issue can be work out by allocating X number of jobs to Y number of processing units in the cloud. Each of the processing units will have a processing unit vector (P_{UV}) showing the status of processing unit utilization. (P_{UV}) consists of MIPS indicating how many million instructions that machine per second, can execute, C_{ex} cost of execution of information and delay cost, C_d. It is an assessment of penalty, which Cloud service holder needs to reward to the customer in the situation of job accomplishing particular time being higher than the limit displayed by the maintenance provider.

$\boldsymbol{P_{UV}}$= f (MIPS, $\boldsymbol{C_{ex}}$, $\boldsymbol{C_d}$) (1)

sSimilarly, a job unit vector (J_{UV}) represents each job submitted by cloud user. Thus, the attribute of different situation can be represented by 2.

$\boldsymbol{J_{UV}}$= f ($\boldsymbol{T_s}$, N, α,$\boldsymbol{W_C}$) (2)

Where,T_s, represents the nature of service needed by the job, Software as a Service (SaaS),Infrastructure as a Service (IaaS) and Platform-as-a-Service (PaaS) [10]. N represents the number of instructions present in the job; this is the count of instruction in the task determined by the processor. Job arrival time (α) indicates wall clock time of arrival of job in the system and worst-case completion time (W_C) is the least possible time recommended to complete the job by a processing unit.

The Cloud business jobholder needs to designate these X jobs among Y number of processors such that cost function as recorded in equation 3 is lessened.

Ͼ= [$\boldsymbol{W_1}$*$\boldsymbol{C_{ex}}$(N/ MIPS) + $\boldsymbol{W_2}$*$\boldsymbol{C_d}$]/ $\boldsymbol{i}$ (3)

Here, i is the index of job present in a virtual machine belonging to a single data center. Where, $\boldsymbol{W_1}$ and are predefined weights. The logic of choosing weight is according to user preference. The weights here are considered as $\boldsymbol{W_1}$= 0.8 and $\boldsymbol{W_2}$= 0.2 such that their summation is 1. GA is

recognizing as one of the most extensively used artificial intelligent techniques that uses an approach for adequate search and optimization. It is a stochastic seeking algorithm based on the tool of natural selection and genetics. GAs has been determined sufficient and reliable in seeking out overall optimum solutions, especially in the compound and enormous research field. In this paper, IGA has been designed to a load balancing procedure for the cloud computing to discover a global optimum processor for the task in a cloud. The arrival of job is treated continuous, and rescheduling of appointment is not considered.

Algorithm Used

A genetic algorithm (GA) is composed of three procedures: selection, crossover, and mutation. The influence of this approach is that it can grasp a boundless search space, pertinent to compound objective function and can escape being trapping into regional optimal solution. The working principle of GA used for the load balancing in Cloud computing is depicted in figure 2 and details of GA are described as follows.

Initial population generation: GA works on the fixed bit string representation of the individual solution. In this phase, data from various sources is been sent to cloud storage where the data is encoded into binary strings. From the initial population, fixed amount of chromosomes is chosen randomly.

Crossover: To generate the new string

(offspring) for the next generation crossover operation is performed. In this, first two parent are selected, then a random crossover point is chosen and finally crossover operation with a given crossover probability to create the new string is performed. The principle behind crossover is “by mating two individuals with different but desirable features, an offspring is produced which combines both of those features”.

In this phase, we choose the best-fitted pair of individual chromosomes. The fitness value of each chromosome is calculated using the fitness function as given in 3. This pool of chromosomes experiences an unplanned single point crossover, where depending upon the crossover point, the fragment lying on one side of crossover site is exchanged with the other side. Thus, it brings out an advanced pair of individuals.

Mutation: After the crossover, each of the individuals of the chromosomes will be mutated to any one of the codes with a given mutation probability. Here, (0.05) is picked up as mutation probability. It is a local optimization procedure and creates a new individual by applying a random variation between arbitrarily selected individuals. Depending on the difference in value, the bits of the chromosomes are toggled from 1 to 0 or 0 to 1. The result of this is a new mating pool ready for crossover. The objective with robust probabilities of crossover and mutation are to sustain the genetic dissimilarity in the population and forbid the genetic algorithms to assemble prematurely to local minima.

This GA process is replicated until the fittest chromosome (optimal solution) is found, or the termination condition (maximum number of iteration) is exceeded.

The proposed algorithm is as given below:

Step 1: [Start] generates the random population of n-chromosome and encodes them into binary string.

Step 2: [Fitness] evaluate the fitness value of each population using equation 3.

Step 3: [New Population] while either maximum number of iteration is reached or optimum solution is found Do:

Step 3(a): [Selection] Consider chromosome with lowest fitness twice and eliminate the chromosome with highest fitness value to construct the mating pool

Step 3(b): [Crossover] Perform single point crossover by randomly selecting the crossover point to form new offspring.

Step 3(c): [Mutation] with a mutation probability of (0.05) mutate new offspring at each locus (position on a chromosome).

Step 3(d): [Accepting] place new offspring as new population and use this community for next round of iteration.

Step 4: [Reinitiate] Use new develop population for a fresh run of the algorithm.

Step 5: [Test] if the end condition is satisfied, stop and return the best solution in the current population.

Step 6: [Loop] Go to Step 2.

4. SIMULATION RESULTS AND ANALYSIS

The proposed Improvised Genetic Algorithm (IGA) is simulated in the MATLAB R 2010 toolkit. MATLAB is a high-level language for numerical computation visualization and application development. Math Works developed it. Here, a hypothetical configuration has been generated using MATLAB Toolkit. The data from six different geographical locations is collected. Such user bases used for experimentation are described in Table 1.

Table 1: Simulation configuration

S No.	USER BASE	REGIONS	ONLINE USERS DURING PEAKS HOURS	ONLINE USERS DURING OFF-PEAKS HOURS
1.	UB1	1. NORTH AMERICA	6,000	600
2.	UB2	2. SOUTH AMERICA	2,000	200
3.	UB3	3. EUROPE	5,000	500
4.	UB4	4. ASIA	7,000	700
5.	UB5	5. AFRICA	1,000	100
6.	UB6	6. OCEANIA	1,500	150

A particular time zone is examined for all the user bases, and it is expected that

there are mixed numbers of customers during peak hours and off-peak hours. Particular simulated "data center hosts" has a specific bulk of virtual machines (VMs) dedicated to the application. Each of the Machines has 4 GB of RAM and 100GB of storage, and each machine has 4 CPUs, having a capacity power of 258 MIPS for 400 MHz processor.

4.1 Simulation setup

The experimentation scenario starts with only a single centralized cloud Data Center (DC), which consists of (25, 50, 75, 100, and 125); VMs and varying number of processing nodes. Data from various social networking sites like Facebook, Google+ etc., is collected. Furthermore, each user makes a request every 5 minutes. Cloud processes all user requests around the world Configuration (CCs) and allocates them to particular processing nodes. This simulation setup consisting of a single data center as described in Table 2 with calculated overall average Response Time (RT) in ms for IGA, and the result are compared with existing technique of GA and RR. A performance analysis graph of the same is depicted in figure 3.

Table 2: Simulation scenario and calculated overall average response time (*RT*) in (ms)

S. NO	CLOUD CONFIGURATION	DC SPECIFICATION EACH WITH	RESPONSE TIME USING ROUND ROBIN	RESPONSE TIME USING GENETIC ALGORITHM	RESPONSE TIME USING IMPROVISHED GENETIC ALGORITHM
1.	CC1	25 VMs	97.63	85.12	29.22
2.	CC2	50 VMs	83.11	71.11	27.45
3.	CC3	75 VMs	61.09	53.23	24.33
4.	CC4	100 VMs	58.18	45.15	20.29
5.	CC5	125 VMs	38.11	25.16	18.26

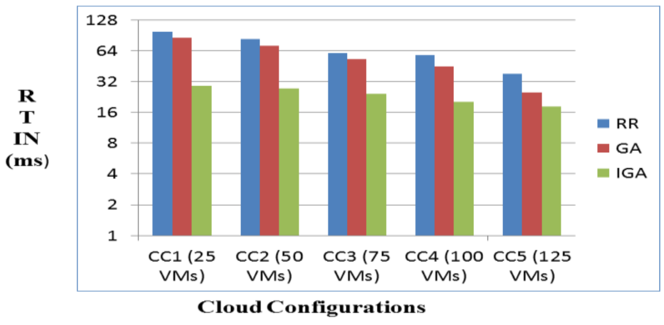

Figure 3: Performance analysis of proposed GA with IGA

Table 1 and graph in figure 3 shows the average response time of the load balancing techniques. The graph shows the average response time obtained in a simulation setup consisting of the single data center, which consist of (25, 50, 75, 100, and 125) VMs. The data from different source node is collected and is then deployed to cloud storage where data is encoded into binary format. Now, the fitness value of each node present in the virtual machine is calculated. Based upon the minimum fitness value of the node crossover is performed, mutation value is set to (0.05), and the process will continue until the end condition is met or the best solution is obtained. Then finally, a new matting pool is created from where the nodes are disseminated to their processing nodes to perform load balancing. It is clear from our graph that our proposed algorithm when simulated in MATLAB toolkit gives better response time than existing GA and RR when simulated in Cloud analyst simulator.

5. CONCLUSION

In this paper, a load balancing strategies called Improvised Genetic Algorithm in a cloud environment is proposed. The proposed soft computing approach is simulated in Matlab toolkit and the result are compared with existing GA and RR which are simulated in cloud analyst simulator. The experiment is been conducted by varying the number of nodes in a VM present in a cloud configuration of single data center. The results are quite encouraging however use of other soft computing techniques are needed to be studied for further improvement.

6. REFERENCES: -

[1] Velte, A.T., Veltey, T.J., and Elsenpeter, R., 2010, "Cloud Computing: A Practical Approach," Tata McGraw-Hill Education Private Limited, New Delhi, Edition.

[2] Jadeja, Y., and Modi, K., 2012, "Cloud Computing-Concepts, Architecture and Challenges," International Conference on Computing, Electronics and Electrical Technologies, pages 877-880.

[3] Shaikh, F.B., and Haider, S., 2011, "Security Threats in Cloud Computing", Internet Technology and Secured Transactions, 214-219.

[4] Srinivas, J., Reddy, K.V.S., and Qyser, A.M., July 2012, "Cloud Computing Basics", International Journal of Advanced Research in Computer and Communication Engineering, volume 1, issue 5.

[5] Ray, S., and Sarkar, A.D., October 2012, "Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment," International Journal of Cloud Computing Services and Architecture, volume 2, issue 5.

[6] Dasgupta,K., Mandal, B., Dutta, P., Mondal, J.K., Dam, S., 2013, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA), volume 10, page 340-347.

[7] Sotomayor, B., Montero, R.S., Llorente, I.M., and Foster, I., 2009, "Virtual infrastructure management in private and hybrid clouds," in IEEE Internet Computing, vol. 13, no. 5, pages 14- 22.

[8] Wang, S.C., Yan, K.Q., Liao, W.P., Wang, S. S., 2010, "Towards a Load Balancing in a three level cloud computing network," in Proc. Third International Conference Computer Science and Information Technology (ICCSIT), IEEE, vol. 1, pages 108—113.

[9] Radojevic, B., and Zagar, M., 2011, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," In Proc. 34th International Convention on MIPRO, IEEE.

[10] Lars, K., Andreas,T., Erhard,R., 2012, "Load Balancing for Map Reduce-based Entity Resolution", in Proc. 28th International Conference on Data Engineering (ICDE), IEEE, pages 618-629.

[11] Gu, J., Hu, J., Zhao, T., Sun, G., January 2012, "A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment," journal of computers, vol. 7, NO. 1.

[12] Jaroodi, Al., Mohamed, N., May 2011, "DDFTP: Dual-Direction FTP," in Proc. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE, pages 504-503.

[13] Nishant, Sharma, K. P., Krishna, V., Gupta, C., Singh, KP., Nitin, N., and Rastogi, R., March 2012, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization," In Proc. 14th International Conference on Computer Modeling and Simulation (UKSim), IEEE, pages 3-8.

[14] Brototi, M., Dasgupta, K., Dutta, P., 2012, "Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach", in Proc. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT).

[15] Randles, M., Lamb, D., and Taleb-Bendiab, A., April 2010, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, pages 551-556.