Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.501 – 509

RESEARCH ARTICLE

A MODEL FOR EVALUATING AND MAINTAINING LOAD BALANCING IN CLOUD COMPUTING

[1]Pushpendra Verma, [2]Dr. Jayant Shekhar, [3]Amit Asthana

[1]Research Scholar, Department of CSE, Swami Vivekanand Subharti University, Meerut, U.P., INDIA
Correspondence: mail2_pushpendra@rediff.com

[2]Professor, Swami Vivekanand Subharti University, Meerut, Uttar Pradesh, INDIA
Correspondence: jayant_shekhar@hotmail.com

[3]Assistant Professor CSE Department, Swami Vivekanand Subharti University, Meerut, Uttar Pradesh, INDIA
Correspondence: amitasthana80@gmail.com

ABSTRACT

Cloud computing is emerging as a new paradigm of large-scale distributed computing. It is a framework for enabling convenient, on-demand network access to a shared pool of computing resources. Load balancing is one of the main challenges in cloud computing which is required to distribute the dynamic workload across multiple nodes to ensure that no single node is overwhelmed. It helps in optimal utilization of resources and hence in enhancing the performance of the system. The goal of load balancing is to minimize the resource consumption which will further reduce energy consumption and carbon emission rate that is the dire need of cloud computing."Cloud computing" is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, data enter and distributed servers. It includes fault tolerance, high availability, Scalability, flexibility, reduced overhead for users, reduced cost of ownership, on Demand services etc. Central to these issues lies the establishment of an effective load balancing algorithm. The load can be CPU load, memory capacity, delay or network load. Load Balancing is the process of distributing the load among various nodes of a distributed System to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time. This technique can be sender initiated, receiver initiated or symmetric type (combination of sender initiated and receiver initiated types). Our objective is to develop an effective load balancing algorithm using Divisible load scheduling theorem to maximize or minimize different performance parameters (throughput, latency for example) for the clouds of different sizes (virtual topology depending on the application requirement).

Keywords: ABC-Artificial Bee Colony, EA-Evolutionary Algorithm CSP-Cloud Service Provider, CSA- Cloud Security Alliance IaaS-Infrastructure as a Service, IP- Intellectual Property

1. INTRODUCTION

The cloud computing is a recent field in the computational intelligence techniques which aims at surmounting the computational complexity and provides dynamically services using very large scalable and virtualized resources over the Internet. It is defined as a distributed system containing a collection of computing and communication resources located in distributed data enters which are shared by several end users. There are two kinds of the cloud; the former is the public cloud in which services may be sold to anyone on the Internet. Amazon Elastic Compute Cloud (EC2), Google App Engine is large public cloud providers. The second type of the cloud is the private cloud. It is a proprietary network or data center that supplies hosted services to a limited number of clients (end-users). Cloud computing allows users to run applications remotely including information technology services called Software-as-a-Service (SaaS) . Besides, Infrastructure-as-a-Service (IaaS) refers to where the application is carried out via useful functions and services provided in the cloud. Between SaaS and IaaS, there are cloud platform services known as Platform-as-a-Service (PaaS) deliver operating system, programming language and solution stack. It aims to improve the applications deployment cost and reduce its complexity. It can be seen that the cloud resource use can be simplified and efficiency operated by the scheduled services use in an optimal manner in the interest of the end users.

- Authentication: The process of proving one‘s identity
- Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver
- Integrity: Assuring the receiver that the received message has not been altered in any way from the original
- Non-repudiation: A mechanism to prove that the sender really sent this message

Cloud computing is an on demand service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. It’s a term which is generally used in case of Internet. The whole Internet can be viewed as a cloud. Capital and operational costs can be cut using cloud computing. Load balancing in cloud computing systems is really a challenge now. Always a distributed solution is required. Because it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfil the required demands. Jobs can’t be assigned to appropriate servers and clients individually for efficient load balancing as cloud is a very complex structure and components are present throughout a wide spread area. Here some uncertainty is attached while jobs are assigned. This paper considers some of the methods of load balancing in large scale Cloud Systems. Our aim is to provide an evaluation and comparative study of these approaches, demonstrating different distributed algorithms for load balancing and to improve the different performance parameters like throughput, latency etc. for the clouds of different sizes. As the whole Internet can be viewed as a cloud of many connection-less and connection oriented services, thus concept of load balancing in Wireless sensor networks (WSN) proposed in can also be applied to cloud computing systems as WSN is analogous to a cloud having no. of master computers (Servers) and no. of slave computers (Clients) joined in a complex structure.

2. RELATED WORK

2.1 Existing Load Balancing Techniques in Cloud Computing:

Following load balancing techniques are currently prevalent in clouds:-

2.1.1 Decentralized content aware load balancing - H. Mehta et al. [21] proposed a new content aware load balancing policy named as workload and client aware policy (WCAP). It uses a unique and special property (USP) to specify the unique and special property of the requests as well as computing nodes. USP helps the scheduler to decide the best suitable node for the processing the requests. This strategy is implemented in a decentralized manner with low overhead. By using the content information to narrow down the search, this technique improves the searching performance and hence overall performance of the system. It also helps in reducing the idle time of the computing nodes hence improving their utilization.

2.1.2 Server-based load balancing for Internet distributed services - A. M. Nakai et al. [22] proposed a new server based load balancing policy for web servers which are distributed all over the world. It helps in reducing the service response times by using a protocol that limits the redirection of requests to the closest remote servers without overloading them. A middleware is described to implement this protocol. It also uses a heuristic to help web servers to endure overloads.

2.1.3 Join-Idle-Queue - Y. Lua et al. [23] proposed a Join- Idle-Queue load balancing algorithm for dynamically scalable web services. This algorithm provides large scale load balancing with distributed dispatchers by, first load balancing idle processors across dispatchers for the availability of idle processors at each dispatcher and then, assigning jobs to processors to reduce average queue length at each processor. By removing the load balancing work from the critical path of request processing, it effectively reduces the system load, incurs no communication overhead at job arrivals and does not increase actual response time.

2.1.4 A Lock-free multiprocessing solution for LB - X. Liu et al. [24] proposed a lock-free multiprocessing load balancing solution that avoids the use of shared memory in contrast to other multiprocessing load balancing\ solutions which use shared memory and lock to maintain a user session. It is achieved by modifying Linux kernel. This solution helps in improving the overall performance of load balancer in a multi-core environment by running multiple load-balancing processes in one load balancer.

2.1.5 Scheduling strategy on load balancing of virtual machine resources - J. Hu et al. [25] proposed a scheduling strategy on load balancing of VM resources that uses historical data and current state of the system. This strategy achieves the best load balancing and reduced dynamic migration by using a genetic algorithm.\ It helps in resolving the issue of load-imbalance and high cost of migration thus achieving better resource utilization.

2.1.6 Central load balancing policy for virtual machines - A. Bhadani et al. [26] proposed a Central Load Balancing Policy for Virtual Machines (CLBVM) that balances the load evenly in a distributed virtual machine/cloud computing environment. This policy improves the overall performance of the system but does not consider the systems that are fault-tolerant.

2.1.7 LBVS: Load Balancing strategy for Virtual Storage - H. Liu et al. [27] proposed a load balancing virtual storage strategy (LBVS) that provides a large scale net data storage model and Storage as a Service model based on Cloud Storage. Storage virtualization is achieved using an architecture that is three-layered and load balancing is

achieved using two load balancing modules. It helps in improving the efficiency of concurrent access by using replica balancing further reducing the response time and enhancing the capacity of disaster recovery. This strategy also helps in improving the use rate of storage resource, flexibility and robustness of the system.

3. HONEY BEE IN NATURE

The exchange of information among bees is the most important occurrence in the formation of collective knowledge. While examining the entire hive, it is possible to distinguish some parts that commonly exist in all hives. The most important part of the hive with respect to exchanging information is the dancing area. Communication among bees related to the quality of food sources occurs in the dancing area. The related dance is called waggle dance. Since information about all the current rich sources is available to an onlooker on the dance floor, she probably could watch numerous dances and choose to employ herself at the most profitable source. There is a greater probability of onlookers choosing more profitable sources since more information is circulating about the more profitable sources. Employed foragers share their information with a probability, which is proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to profitability of a food source. In order to understand the basic behaviour characteristics of foragers better, let us examine the. Assume that there are two discovered food sources: A and B. At the very beginning, a potential forager will start as unemployed forager. That bee will have no knowledge about the food sources around the nest.

There are two possible options for such a bee:

(i) It can be a scout and starts searching around the nest spontaneously for a food due to some internal motivation or possible external clue.

(ii) It can be a recruit after watching the waggle dances and starts searching for a food source. After finding the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. Hence, the bee will become an ‘‘employed forager’’. The foraging bee takes a load of nectar from the source and returns to the hive, unloading the nectar to a food store. After unloading the food, the bee has the following options:

(i) It might become an uncommitted follower after abandoning the food source (UF).

(ii) It might dance and then recruit nest mates before returning to the same food source (EF1).

(iii) It might continue to forage at the food source without recruiting after bees (EF2). It is important to note that not all bees start foraging simultaneously. The experiments confirmed that new bees begin foraging at a rate proportional to the difference between the eventual total number of bees and the number presently foraging.

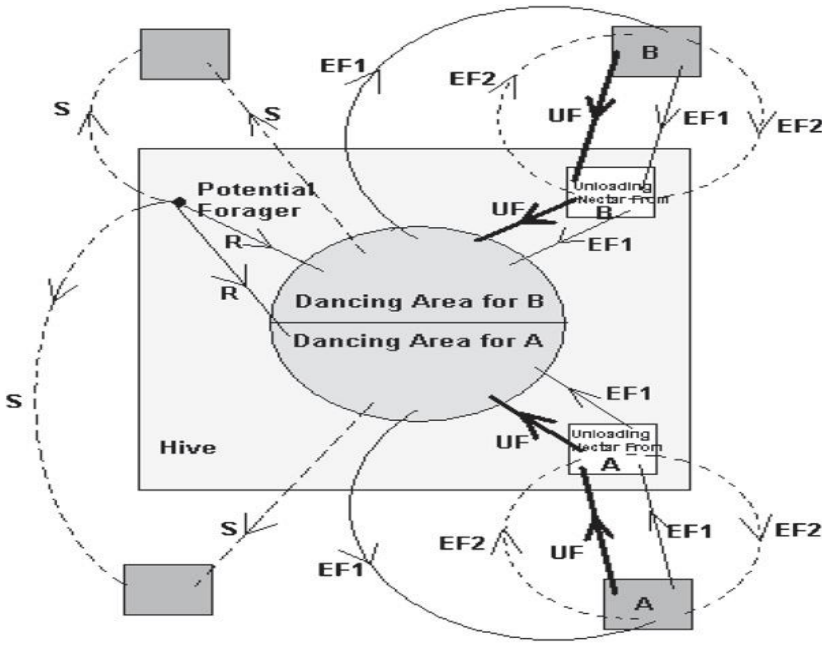

Figure: Behaviour of honeybee foraging for nectar

4. MATERIALS AND METHODS

The Bee Agent Model

In this section, we will briefly describe the organizational principles of bee behaviour that inspired us to transform them into an agent model. Honey bees evaluate the quality of each discovered food site and only perform *waggle dance* for it on the *dance floor* if the quality is above a certain threshold. So not each discovered site receives reinforcement. As a result, quality flower sites are exploited quite extensively. Hence we abstract a *dance floor* into a routing table where *bee agents*, launched from the same source but arrived from different neighbours at a given node, could exchange routing information to model the network state at this node. The agent communication model of *BeeHive* could easily be realized as a *blackboard system*. In comparison *AntNet* utilizes the *principle of stigmergy* for communication among agents.

The majority of foragers exploit the food sources in the closer vicinity of their hive while a minority among them visit food sites far away from their hive. We transformed this observation into an agent model that has two types of agents: *short distance bee agents* and *long distance bee agents*. *Short distance bee agents* collect and disseminate routing information in the neighbourhood (upto a specific number of hops) of their source node while *long distance bee agents* collect and disseminate routing information to all nodes of a network. Informally, the *Beehive* algorithm and its main characteristics could be summarized as follows:

1. The network is organized into fixed partitions called *foraging regions*. A partition results from particularities of the network topology. Each *foraging region* has one representative node. Currently the lowest IP address node in a *foraging region* is elected as the representative node. If this node crashes then the next higher IP address node takes over the job.

2. Each node also has a node specific *foraging zone* which consists of all nodes from whom *short distance bee agents* can reach this node.

3. Each non-representative node periodically sends a *short distance bee agent*, by broadcasting replicas of it to each neighbour site.

4. When a replica of a particular *bee agent* arrives at a site it updates routing information there, and the replica will be flooded again, however, it will not be sent to the neighbour from where it arrived. This process continues until the life time of the agent has expired, or if a replica of this *bee agent* had been received already at a site, the new replica will be killed there.

5. Representative nodes only launch *long distance bee agents* that would be received by the neighbours and propagated as in 4. However, their life time (number of hops) is limited by the *long distance limit*.

6. The idea is that each agent while travelling, collects and carries path information, and that it leaves, at each node visited, the trip time estimate for reaching its source node from this node over the incoming link. *Bee agents* use priority queues for quick dissemination of routing information.

7. Thus each node maintains current routing information for reaching nodes within its *foraging zone* and for reaching the *representative nodes* of *foraging regions*. This mechanism enables a node to route a data packet (whose destination is beyond the *foraging zone* of the given node) along a path toward the *representative node* of the *foraging region* containing the destination node.

8. The next hop for a data packet is selected in a probabilistic manner according to the quality measure of the neighbours; as a result, not all packets follow the best paths. This will help in *maximizing the system performance though a data packet may not follow the best path*, a concept directly borrowed from a principle of bee behaviour: *A bee could only maximize her colony's profit if she refrains from broadly monitoring the dance floor to identify the single most desirable food* (In comparison *OSPF* always chooses a next hop on the shortest path).

BEES ALGORITHM

1. Initialise population with random solutions.

2. Evaluate fitness of the population.

3. While (stopping criterion not met)

//Forming new population.

4. Select sites for neighbourhood search.

5. Recruit bees for selected sites (more bees for best **e** sites) and evaluate fitness.

6. Select the fittest bee from each patch.

7. Assign remaining bees to search randomly and evaluate their fitness.

8. End While.

The algorithm requires a number of parameters to be set:

1. Number of scout bees **n**
2. Number of sites selected **m** out of **n** visited sites
3. Number of best sites **e** out of **m** selected sites

4. Number of bees recruited for best **e** sites **nep** or (**n2**)
5. Number of bees recruited for the other (**m-e**) selected sites which is **nsp** or (**n1**)
6. Initial size of patches **ngh** which includes site and its neighbourhood and stopping criterion
7. Number of algorithm steps repetitions **imax**

5. PROPOSED ALGORITHM

An Extended Bee Algorithm proposed, starts with bee population initialization step which contains N bees (individuals) chosen randomly in the search space. The population fitness is evaluated in the second step. A bee population contains *'1'* queen, *'D'* drones and *'W'* workers in which the fittest bee represents the queen, the *'D'* fittest following bees represent the drones and the remaining bees are the workers. Consequently, the sum of the different bee individuals *(1, D and W)* equal to the population size *(N). 'D'* and *'W'* are considered as a two user-defined parameters. Each cycle of a bee population life consists of two bee behaviours: reproduction and food foraging respectively. In reproduction behaviour, the queen starts mating in the space by mating-flight with the drones using crossover and mutation operators. Next, queen starts breeding *'N'* broods in step 4. Then, the evaluation of the brood fitness is performed (steps 5). If the fittest brood is fitter than the queen, it will be considered as the new queen for the next population. Moreover, *'D'* best bee individuals are chosen among the *'D'* fittest following broods and the drones of the current population to form the drones of the next population. After that, *'W'* best bee individuals are chosen among the *'W'* fittest remaining broods and the workers of the current population in order to ensure the food foraging (steps 6 to 8). In step 9, the *'W'* workers search food source in *'W'* regions of flowers. We consider that each worker represents one region and there are other bees for each region recruited and employed to search the best food source among the different food sources in the region (step 10). The recruited bees represent neighbour solutions in the search space used to ensure neighbourhood search. BLA uses more recruited bees for the *'B'* best regions among *'W'* regions. *'B'* is user-defined parameter. For each region in step 11, only the bee with the highest fitness will be selected to form the next bee population. The evaluation of the new population fitness is executed in step 12. If the stopping criterion is not satisfied, a new bees' life cycle is performed, and then we rerun the third step and so on.

1. Initialize population (*N* bees) at random
2. Evaluate fitness of population (fittest bee is the queen, *D* fittest following bees are drones, *W* fittest remaining bees are workers)
3. ***While*** stopping criteria are not satisfied (Forming new population)

/* **reproduction behavior** */

4. Generate ***N*** broods by **crossover** and **mutation**
5. Evaluate fitness of broods
6. If the fittest brood is fitter than the queen then replace the queen for the next gene

7. Choose *D* best bees among *D* fittest following broods and drones of current population (Forming next generation drones)

8. Choose *W* best bees among *W* fittest remaining broods and workers of current population (to ensure food foraging)

/* **food foraging behavior** */

9. Search of food source in *W* regions by *W* workers

10. Recruit bees for each region for **neighbourhood search** (more bees *(FBest)* for the best *B* regions and *(FOther)* for remaining regions)

11. Select the fittest bee from each region

12. Evaluate fitness of population (fittest bee is the queen, *D* fittest following bees are drones, *W* fittest remaining bees are workers)

13. ***End while***

6. RESULTS

In Extended Bee Algorithm storing best fittest be from each site from selected site. Using this stored value we assigned new bees for next time. Define stopping criteria for each site based on site performance so that overhead must be less and response time must be short. When we assigned new bees to selected site if stopping criteria met then we assigned it for random solutions. Using this we search once for each calculate its stopping criteria and best response time as bit in array. Whenever in future we assigned new populations of bees for this site if these criteria not matched assign new bee or problems to this otherwise assign for random solutions. Using this Extended Bee Algorithm response time and overhead must be less than bee algorithm.

7. CONCLUSION

The Cloud Computing provider must assure the data owner that they provide full disclosure (aka 'transparency') regarding security practices and procedures as stated in their SLAs. Ensure specific identification of all controls used during the data lifecycle. Ensure there specifications of to which entity is responsible for each control between the data owner and cloud services provider.

As there are numbers of issues related to security, we here just focus on data security issues during the different phases of the lifecycle of data in the cloud computing model. For the security purpose of data, below given some recommendations from the researchers, experts and other sources. Both the consumer and provider require using these recommendations make proper secure model for the cloud computing architecture.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", EECS Department, University of California, Berkeley, Technical Report No., UCB/EECS-2009-28, pages 1-23, February 2009.

[2] R. W. Lucky, "Cloud computing", IEEE Journal of Spectrum, Vol. 46, No. 5, May 2009, pages 27-45.

[3] M. D. Dikaiakos, G. Pallis, D. Katsa, P. Mehra, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research", IEEE Journal of Internet Computing, Vol. 13, No. 5, September/October 2009, pages 10-13.

[4] G. Pallis, "Cloud Computing: The New Frontier of Internet Computing", IEEE Journal of Internet Computing, Vol. 14, No. 5, September/October 2010, pages 70-73.

[5] B. P. Rima, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems", Proceedings of 5th IEEE International Joint Conference on INC, IMS and IDC, Seoul, Korea, August 2009, pages 44-51.

[6] R. Mata-Toledo, and P. Gupta, "Green data center: how green can we perform", Journal of Technology Research, Academic and Business Research Institute, Vol. 2, No. 1, May 2010, pages 1-8.

[7] S. K. Garg, C. S. Yeob, A. Anandasivamc, and R. Buyya, "Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers", Journal of Parallel and Distributed Computing, Elsevier, Vol. 70, No. 6, May 2010, pages 1-18.

[8] K. M. Nagothu, B. Kelley, J. Prevost, and M. Jamshidi, "Ultra low energy cloud computing using adaptive load prediction", Proceedings of IEEE World Automation Congress(WAC) , Kobe, September 2010, pages 1-7.

[9] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems, Cloud Computing: Principles, Systems and Applications", Computer Communications and Networks, Chapter 2 , pages 21-46, DOI 10.1007/978-1-84996-241-42, Springer – V erlagLondonLimited, 2010.

[10] S. Kabiraj, V. Topka, and R. C. Walke, "Going Green: A Holistic Approach to Transform Business", International Journal of Managing Information Technology (IJMIT), Vol. 2, No. 3, August 2010, pages 22-31.

