International Journal of Hybrid Information Technology
Vol.8, No.3 (2015), pp.9-20
http://dx.doi.org/10.14257/ijhit.2015.8.3.02

An Enhanced Cloud Network Load Balancing Approach Using Hierarchical Search Optimization Technique

Debabrata Sarddar[1] Rajesh Bose[2] and Sudipta Sahana[3]

Department of Computer Science and Engineering University of Kalyani, Nadia, West Bengal, India
[1]dsarddar@rediffmail.com, [2]bose.raj00028@gmail.com, [3]ss.jisce@gmail.com

Abstract

As one of the driving forces changing the way research and industry uses virtualization, distributed computing, internet, software and web services today, cloud computing stands tall. A cloud is an ecosystem of data centers, distributed servers, and clients all interconnected to each other. The unique selling point of a cloud is its reduced cost of ownership in comparison to traditional models. This primary advantage is complemented by fault tolerance, high availability opportunity, scalable and flexible structure, reduced infrastructure overheads for users, and services that can be accessed as and when required. One of the challenges that face cloud computing is load balancing. Load balancing assures optimum use of available resources, thereby, enabling consistency and performance of the overall system. An imbalance of load causes a single node or nodes to operate beyond its optimum levels. As a result, there could be either a gradual or a rapid loss in overall efficiency of the system leading to increase in emission rates and inefficient use of energy. In this paper, we have focused on resourceful load balancing coupled with a technique which reduces flooding. We have discussed how a combination of these is able to ensure efficient routing with reduced carbon emission.

Keywords: *Cloud computing, static load balancing, dynamic load balancing algorithm, honey bee algorithm, packet flooding, search optimization*

1. Introduction

The popularity of cloud computing has witnessed a surge over the last few years. Among the repertoire of services that it provides, cloud computing facilitates a resilient and yet a flexible system according ease of use in storing and retrieving data and information. This is evident when handling very large sets of files and data among users located at various points across the globe. Managing large volumes of data is a task that demands constant attention. To this effect, several techniques are employed to minimize resistance in operations and to optimize user experience to accord a level of performance that is deemed satisfactory. It becomes imperative, therefore, to conduct research in areas related directly to cloud computing to enhance utilization of storage and boost download times for users. One of the most stern challenges in this field happens to be is the balancing of load factors in real-time or dynamically, with scheduling of task. Algorithms for load balancing were examined almost extensively in various settings. With cloud computing, however, additional obstacles must be addressed and solved for smooth functioning.

Diverse constraints and high latency in communications must be overcome so that cloud nodes can be assigned tasks as quickly as possible. This is a prime concern in cloud computing resource optimization. Algorithms are classified as either static or dynamic in the context of load balancing. Static algorithms are suited where conditions are homogenous and stable. In such cases, static algorithms are able to deliver consistent results which are usually satisfactory. They do suffer in terms of their inflexible nature

ISSN: 1738-9968 IJHIT

and inability to cope with dynamically changing attributes caused during process execution. On the other hand, dynamic algorithms are more flexible. These take into account the variations in attributes in a system before and in run-time. Further, these algorithms are able to accommodate alterations and have the ability to produce better results insofar as mixed dynamic environments are concerned where multiple products from assorted vendors are made to run at same or different levels.

With increasing complexity and vigorous nature of distribution attributes, a number of these algorithms can begin to perform below par. These then, may be themselves be deemed as unnecessary with service levels and overall performance dipping down below what might be viewed as being acceptable. In this paper, we present a literature survey of an assorted number of load balancing algorithms developed and targeted specifically for cloud computing systems. While bringing forth an overview of these algorithms, we also discuss the properties that each of these possess.

An interesting algorithm that we discuss in our work is the honey bee algorithm [13]. This, as evident from the name itself, is derived from the behavior of the honey bees. These bees follow a sequence of actions for searching, locating, and harvesting food sources. A somewhat similar approach is also applicable in cases of distributed systems where one of the primary concerns is to balance load. Broadly, the honey bees can be classified into two groups comprising forager bees and scout bees. The former wanders in search of food, and upon locating a source returns to the beehive to deliver information on the quality, quantity and the distance from the hive to the food source. The latter then follows the forager bees to extract the food. Upon returning to the hive, the scout bees share with the other worker bees, information on the food that remains to be gathered. This information is used by the honey bees to make a decision to either revisit the food source or abandon it. This is an example of a decentralized load balancing procedure. Performance of system is augmented with increase in variations and disparateness. But throughput is not increased with an increase in size of system [14]. The algorithm is able to deliver the best performance where varying forms of services are needed.

In our proposed method, we have used Billboard Manager (BM) [15], as an electronic device which is able to obtain data involving capacity, shortest node distance and related information of a cloud server. The BM, in order to balance load optimally, is placed in a position to directly control the cloud server. This is done to distribute load among the cloud servers as quickly as possible. This leads to reduction of power consumption which, in turns, minimizes the rate of carbon emissions. While not being a cost-effective approach, it does, however, have a positive effect on our environment as it assists in reducing carbon emissions.

Our paper is segregated into several sections wherein we discuss related work in Section 2. Subsequently, in Section 3, we discuss our proposed method and approach. In Section 4, we present an algorithm for our proposed method. And in Section 5, we follow that up with a flow chart explaining the flow of steps. In Section 6, we analyze the results that we have observed. We arrive at our paper's conclusion in Section 7. Finally, we present a list of references, in Section 8, of authors whose works we have referred to in our work.

2. Related Work

In this section, we discuss acknowledged and published works in the annals of load balancing in cloud computing. We segregate load balancing algorithms into two types – static algorithms and dynamic algorithms. We begin by discussing static load balancing algorithms targeted at meeting cloud computing requirements. Following which, we will discuss dynamic load balancing algorithms.

1.1 Static Load Balancing Algorithms

Assigning tasks to nodes subject to the ability of the node to process new requests is what static load balancing algorithms are designed for. This activity involves prior knowledge of the resources and capabilities of each node. Parameters such as processor performance, memory available, storage capacity, and the last known communication history are also key factors included by such algorithms. While existing communication performance history might be considered, static algorithms are usually not concerned with dynamic alterations to these attributes at run-time. Algorithms such as these are unable to cope with changes in load while processes are running.

The Central Load Balancing Decision Model, or CLBDM [1], was an algorithm suggested by Radojevic. This algorithm is essentially a modified version of the Round Robin algorithm which works on the principle of session switching at the application layer. The CLBDM offers improvement over this functioning of Round Robin. The latter [2] itself is a well-known load balancing algorithm. It functions by forwarding requests to the node which has the least number of connections. CLBDM works by calculating the time it takes to connect the client and the cloud node. If this connection time exceeds a set threshold, then it warns of an issue. This is a marked improvement over the Round Robin algorithm. In case such an issue is detected, termination of the particular connection takes place. The task is then forwarded to another node based on stipulations of the Round Robin algorithm. CLBDM can, therefore, be considered as an administrator which functions autonomously and requires no intervention or constant attention. This is analogous to a human administrator and the idea behind creation of CLBDM came from the actions and perspective of a human administrator.

The algorithm as proposed by Kumar [3] is an improvement over the algorithm [4]. Both these algorithms use behavior of ants to collect information on cloud nodes before assigning tasks to particular nodes. The algorithm [4], however, suffers from the ants' synchronization problem. The author [3] has tried to work out this problem by incorporating the "suicide" feature in the ants' behavior. The two algorithms begin working once a request is made. This is similar to release of pheromones when the ants begin to work their way forward. In the case of the algorithms, these two move forward from the "head" node.

A forward movement implies that an ant is proceeding from an overloaded node, and it is on the lookout for the next available node to see if it is overloaded or not. In case, it finds a node that is not loaded to the mark, or is under-loaded, it will move ahead to check the next one. If the next one is an overloaded node, the ant will move backwards to arrive to the previous node from where it last left. The algorithm [3] adds to this mechanism the action of committing suicide by the ant once it locates the suitable node target. This has the effect of preventing unwanted movements in the reverse direction.

Proposed in this algorithm [5] is a supplement to the Map Reduce algorithm [6]. The Map Reduce model does two actions. It maps tasks, and then follows it up by reduction of the tasks results. The model offers three methods in addition to the tasks. These three methods are the part method, the comp method, and the group method. First, the part method is executed by Map Reduce to begin task mapping. In this step, the request entity is segregated into different sections using the Map tasks. Next, the keys to corresponding parts are saved in a hash key table. It is the comp method which conducts comparisons of the parts. After comparing of the parts, the group method takes over. It assembles the parts, using the Reduce task, into entities that closely resemble each other. It is possible for the Reduce tasks to become overloaded since more than one Map task can read entities simultaneously for processing. In view of this, it has been proposed in this paper, to add another load balancing level layered between the Map and the Reduce tasks. This median load balancing would divide only large tasks into smaller ones. In turn, the smaller tasks would be sent to reduce tasks subject to availability.

In his work, Junjie [7] has proposed an algorithm for load balancing for private cloud incorporating mapping of virtual machines to physical ones. This algorithm is based on an architecture which is composed of a central scheduling controller and a resource monitor.

The calculations are performed by the scheduling controller. These calculations entail identifying the resource capable to take on the task and, subsequently, assigning the task to that resource. While that may be the job of the scheduling controller, the actual job of gathering details on resource availability falls on the resource monitor. The process of mapping tasks is funneled through four distinct phases. These begin at accepting request from the virtual machine; obtaining details of resources using the resource monitor; calculating resource availability and ability to handle tasks by the controller. The resource which chalks up the highest score is the one which is allotted the task. The final phase is the one where the client is actually able to access and use the application.

2.2 Dynamic Load Balancing Algorithms

Dynamic load balancing algorithms work through an inclusive approach of combining the different properties and attributes related to capabilities and network bandwidth in respect of each node. A sizeable number of these algorithms depend on an amalgam of knowledge built on information collected about each node present in the cloud. In addition to that, the algorithms also take into account the properties collected during run-time even as the selected nodes go about processing components of tasks assigned. These algorithms allocate tasks and may progressively reassign the same tasks to nodes subject to calculations on properties collected. Algorithms such as these are considerably more complicated to put in practice; and these demand continual observation and supervision of the progress of the tasks and the state of the nodes. The gains of such algorithms are usually evident in the accuracy offered and increased efficiency in balancing of load.

Supported by an existing algorithm known as weighted least connection (WLC) [9], Ren [8] has designed a dynamic load balancing algorithm applicable in the context of cloud computing. The WLC algorithm functions by assigning tasks to nodes taking into account the number of connections which exist for it at a particular point of time. This is achieved by comparing the sum total of connections of every cloud node and, subsequently, assigning a task to the node running with least number of connections. WLC, however, does not account for node attributes such as speed of processing, storage capacity, and network bandwidth.

The algorithm which has been proposed is called ESWLC (Exponential Smooth Forecast based on Weighted Least Connection). This proposed algorithm is an improvement over WLC and works by taking into account the time series and trials. In other words, ESWLC arrives at the point of assigning a certain task to a particular node after assigning a set number of tasks to it in order to obtain a fix on the capabilities of the node.

ESWLC makes a decision by compiling historical records covering processing power, memory available, number of connections, the amount of disk space utilized of a given node. ESWLC can then forecast which node can be selected by exponential smoothing. The algorithm proposed [10] is a dual direction downloading algorithm from FTP servers (DDFTP). This can also be implemented where load balancing for cloud computing is concerned. DDFTP splits a file size of, say, m into m/2 partitions. Next, each server node begins working on the tasks assigned to it by following a set pattern. A server, for example, will start from block 0 and keep on downloading in increments. Simultaneously, another server begins from block m and goes on downloading in a gradual decrement order. As a result, while both servers work independently, they would download the entire file to the client within a time frame that is optimum considering the combined performance and attributes of the two servers. In this manner, when two servers download consecutive blocks, the task is considered complete. Other tasks can be allotted to the servers, thereafter. The proposed algorithm is able to diminish dependence on network

communication required between client and nodes. This, therefore, has the effect of reducing network overhead. In addition, the proposed algorithm automatically considers attributes such as network bandwidth, network load, and node load without requiring run-time monitoring of the nodes.

An algorithm called Load Balancing Min-Min (LBMM) has been proposed in paper [11]. This algorithm is built on a framework which has a three-level load balancing format. It works using the Opportunistic Load Balancing algorithm (OLB) [12]. As a static load balancing algorithm, OLB aims at keeping each cloud node busy. The execution time of a node, however, is not considered by OLB. This, at times, causes tasks to run slower. Consequently, bottlenecks are not uncommon as requests queue up for nodes to be free to take up as and when available.

With the introduction of LBMM, OLB is significantly enriched by infusion of a three-layered architecture to the algorithm. A request manager is the first level of LBMM architecture. This handles actions of receiving tasks and assigning it to a service manager in the second level of LBMM. After a service manager receives a request, it partitions it into separate tasks to boost the speed of processing that request. The job of the service manager is to assign a sub-task to a service node. In turn, the latter would be responsible for execution of the given task. The service manager goes about in assigning the tasks after taking into consideration attributes covering CPU space remaining (node availability), free memory, and rate of transmission.

3. Proposed Method

In this paper, we have put forth our proposal of a technique aimed at refining search optimization. This proposed technique would ensure less flooding and an appropriate load balancing approach while sifting through data or managing a file of enormous proportions. Adoption of an appropriate search technique is one of the primary drivers in ensuring that the network has a reduced carbon emission footprint.

We foresee that as a result of adoption of this algorithm, the possibility of not being able to gainfully utilize available bandwidth would be significantly reduced. In this model, a cloud node first requests for resources from a cloud server using the shortest available path. Upon receiving such a request, the cloud server then checks its database for availability of the resource. If the resource is found, then the cloud server responds with the resource data and information. Otherwise, the request is forwarded to Billboard Manager. When the request is received by it, the BM transmits the request to all cloud servers registered with it. The routing of the requests takes place using a method which calculates channel capacities from the routes available to the BM. A cloud server which hosts the resources being sought, sends an appropriate response to the BM as the latter is responsible for the delivery to the desired appropriate location. In the event the data is not available to that BM, it then requests the Central Manager (CM) to locate from its links whether the requested resource can be sourced. The CM looks into its buffer. If it finds that its buffer does not contain sufficient information to service the request, it queries all other BMs. The path to the BMs is computed using the honey bee algorithm. In case the resource is found available with more than one CS then two factors are considered to choose the CS from which the resource can be retrieved. The first factor is the channel capacity; the other one being the CS with the most resources. The CM retrieves the resource from the selected CS and delivers that to the CN which placed the original request.

Our algorithm is built around the following units which make up the entire proposed architecture. Simple computing devices which could be handheld computers, tablets, mobile phones, and commodity computers form a collection of cloud nodes (CN). Several CNs connect to a single cloud server (CS). Each CS harvests information on data storage space available, shortest distance to a CN from the CS, and relevant hardware information

of each CN. This information is constantly monitored and updated at regular intervals by each CS.

A Billboard Manager (BM) is used to store information on each CS connected to it. Such information consists of capacity, IP address, all possible routes to the CS, maximum server efficiency, and other relevant hardware information. A BM may have more than one CS attached to it.

At the heart of this lattice formed by several BMs, CSs, and CNs is the central manager (CM). It is the job of the CM to maintain a buffer that contains relevant information on nodes which requested access through it, and nodes which came up in search requests forwarded to CM. The CM, while maintaining these records, also monitors location of each BM and the shortest distance from the CM to each such. The CM maintains and constantly updates the routes to each BM.

4. Algorithm

1. A cloud node (CN) requests for a file.
2. The request is directed to the associated cloud server (CS).
3. If the requested information is available, the CS sends the file from its repository to the CN.
4. The request session is closed and the CS waits for the next request.
5. In case the requested information is unavailable, the CS forwards the request to its associated BM.
6. The BM responds by querying all the CSs registered with it to check whether the data sought can be located and retrieved. In doing so, the BM begins sending the request to the CS with the shortest path to the greatest one.
7. The CS which has the data responds by notifying the BM of availability.
8. The desired data is routed by the BM from the nearest CS to the CN which had placed the request for the file.
9. The request session is then closed and the BM and CSs wait for the next request.
10. If no data is found, the BM notifies the central manager (CM) that it needs to find out whether the requested file can be found in its archives.
11. The CM starts off by first checking its buffer of recently serviced requests.
12. In case it locates a matching request, the data is retrieved from the appropriate location – a CS – and sent back to the BM. The BM, in turn, forwards it to the CS which raised the query in response to the initial request placed by the corresponding CN.
13. If the CM does not find any matching request in its buffer, it sends out a common query to all the BMs connected to it. In doing so, it follows the honey bee algorithm to ascertain the best fit path and resource availability.
14. The BMs receiving the queries, passes the request to the group of CSs attached to each corresponding BM.
15. The CSs which have the desired content respond on receiving the request.
16. The responses – which include the actual file(s) being sought - are forwarded by the corresponding BM to the CM.
17. The CM, finally, processes and closes the session request after transmitting this data, through the corresponding BM, to the destination CN that had originally placed the request.
18. In case, the requested file is nowhere to be found within the entire network comprising of the CM, BMs, CSs and CNs, a message stating unavailability of the desired data is posted to the querying CN. The request session is closed, thereafter, by the CM.

19. Following completion of a cycle which begins by a CN requesting data, through to final closure of the request with a success or failure result, the CM and the BMs wait for to begin processing the next incoming request.

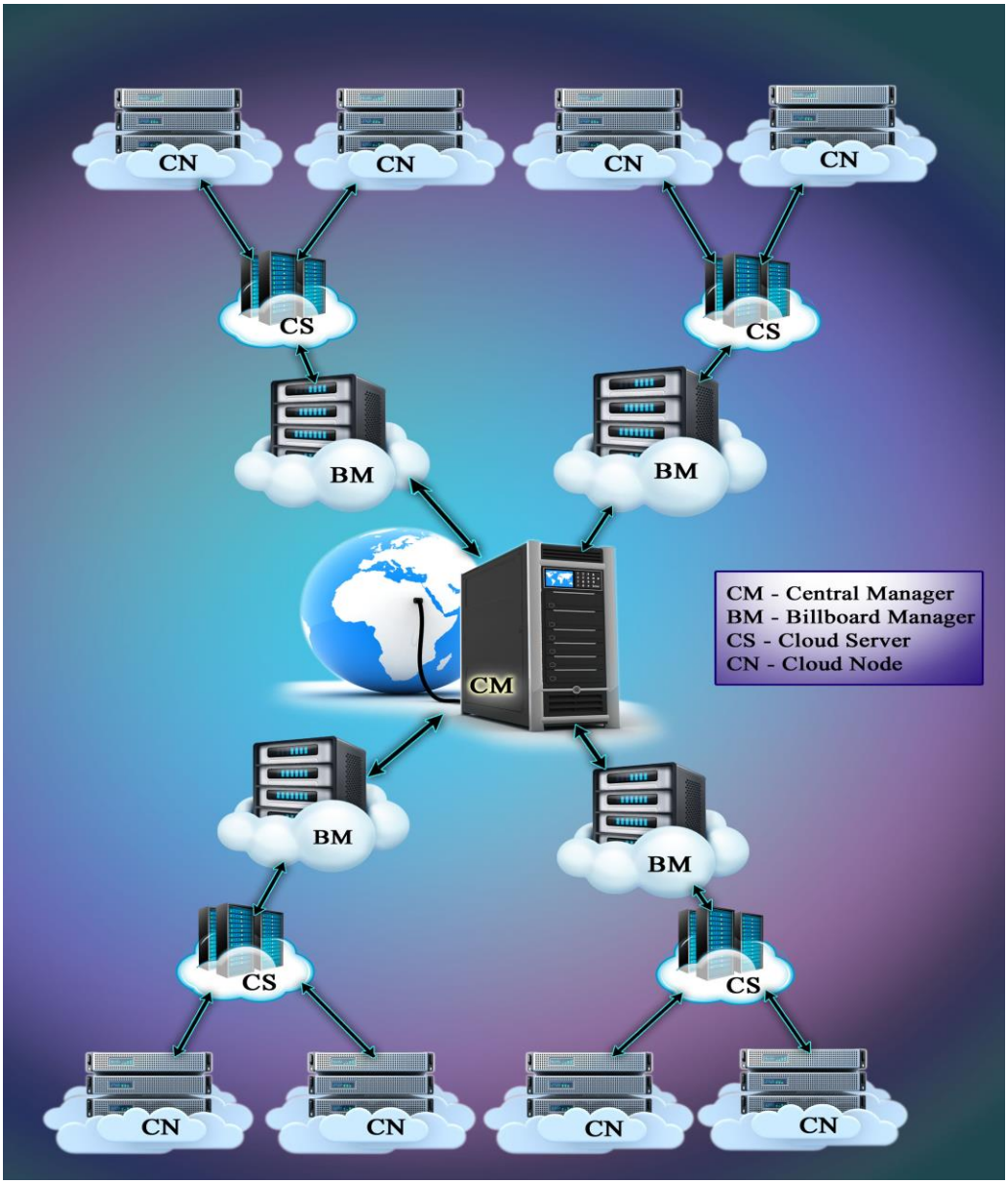
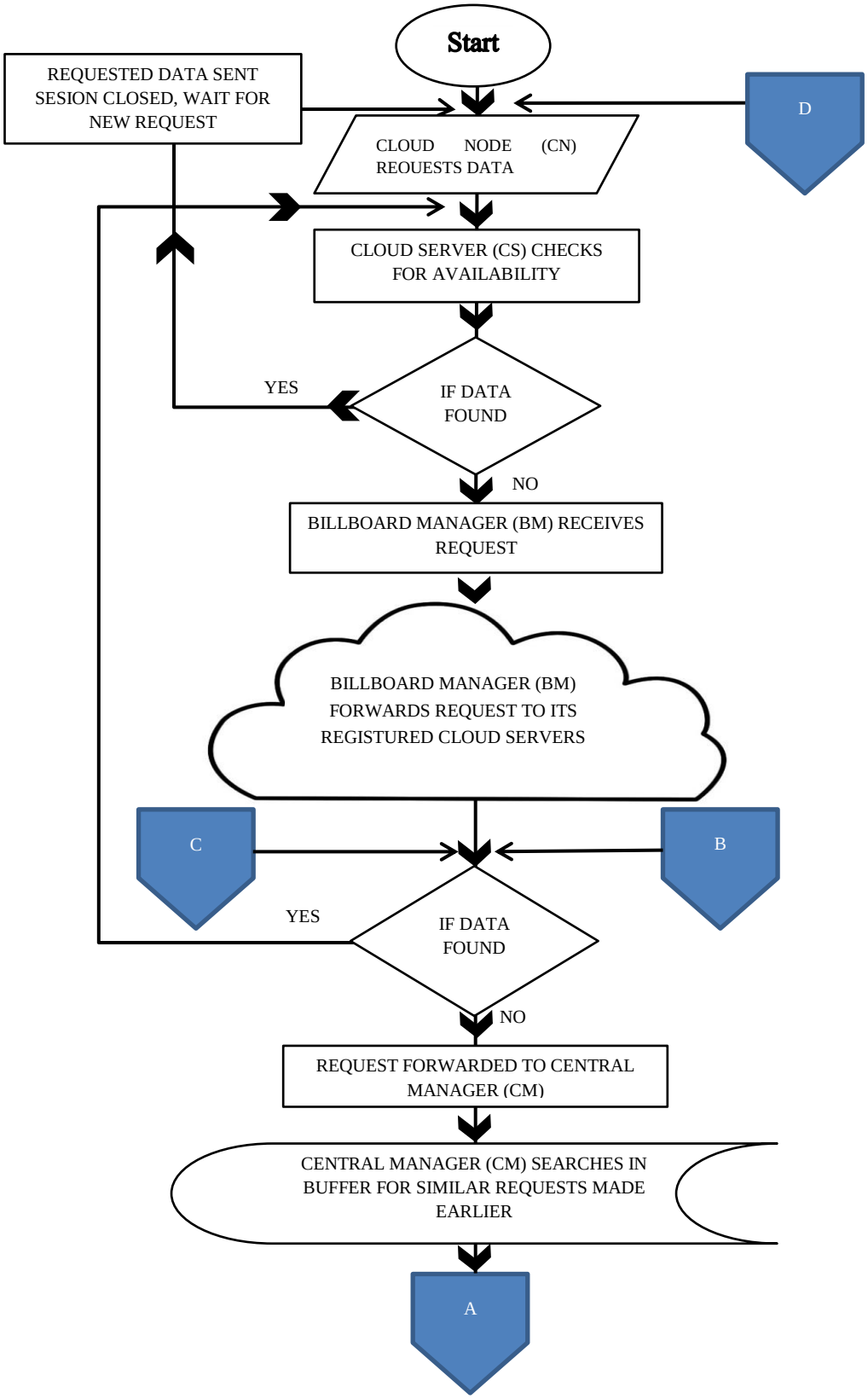
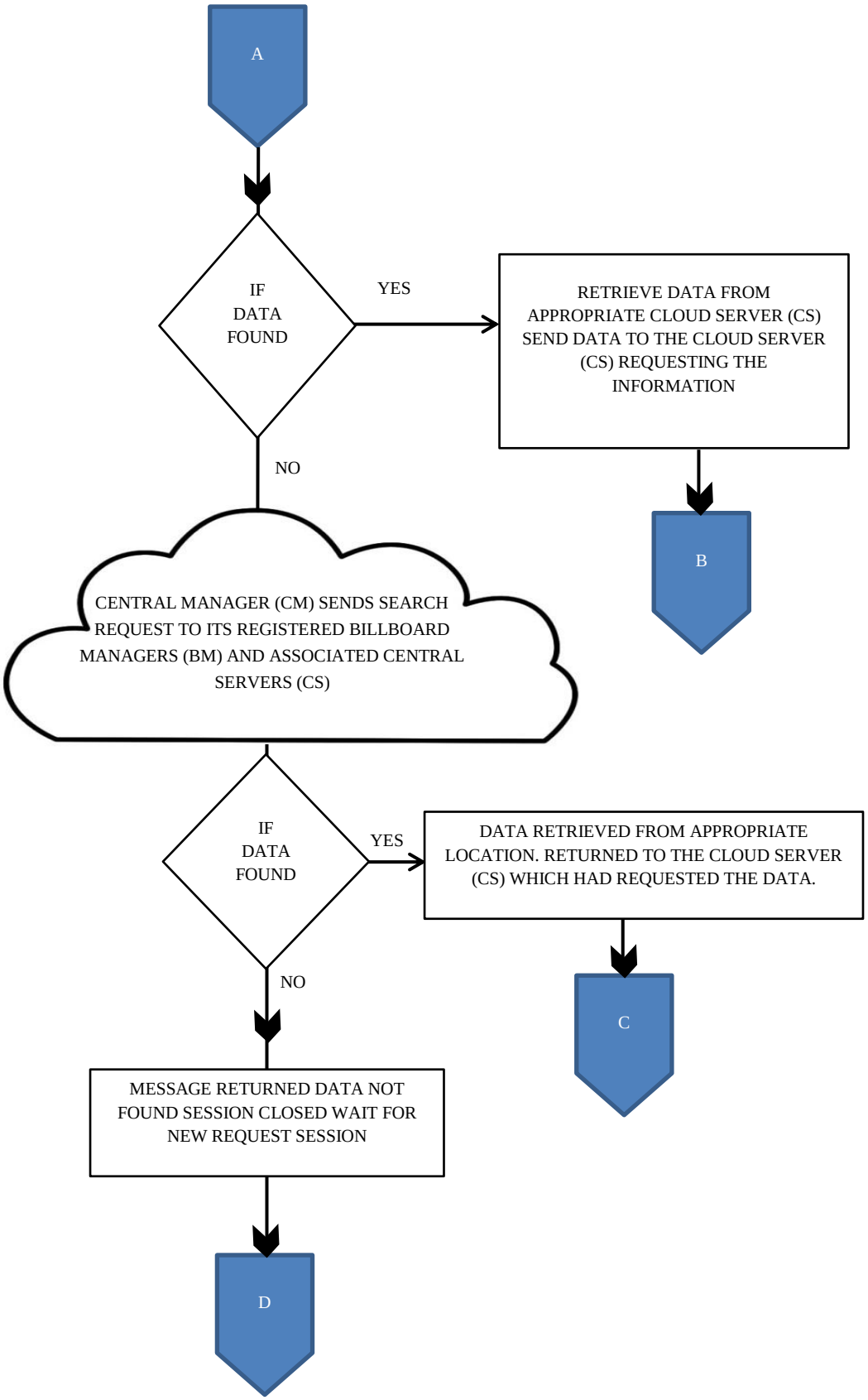


Figure 1. Architecture for the Proposed Method

5. Flow Chart





6. Result Analysis

The primary goal of this paper is to demonstrate an efficient resource searching policy combined with an appropriate load balancing technique. The procedure to search for resources is broken down into several stages. In other words, a hierarchical structure is maintained. Survey conducted on research papers and published works reveals that authors tend to concentrate on load balancing and search methods. In our view, attention must be paid to packet flooding. Studies on packet flooding would reveal that absence of control leads to wastage of bandwidth and fuels higher rate of carbon emissions. From the graph produced below, it can be seen that the rate of carbon emissions increase with a proportionate increase in hop count. Hop counts increase with the number of searches at varying levels. Our proposed method demonstrates that a better result can be obtained with lowering of emission rate.

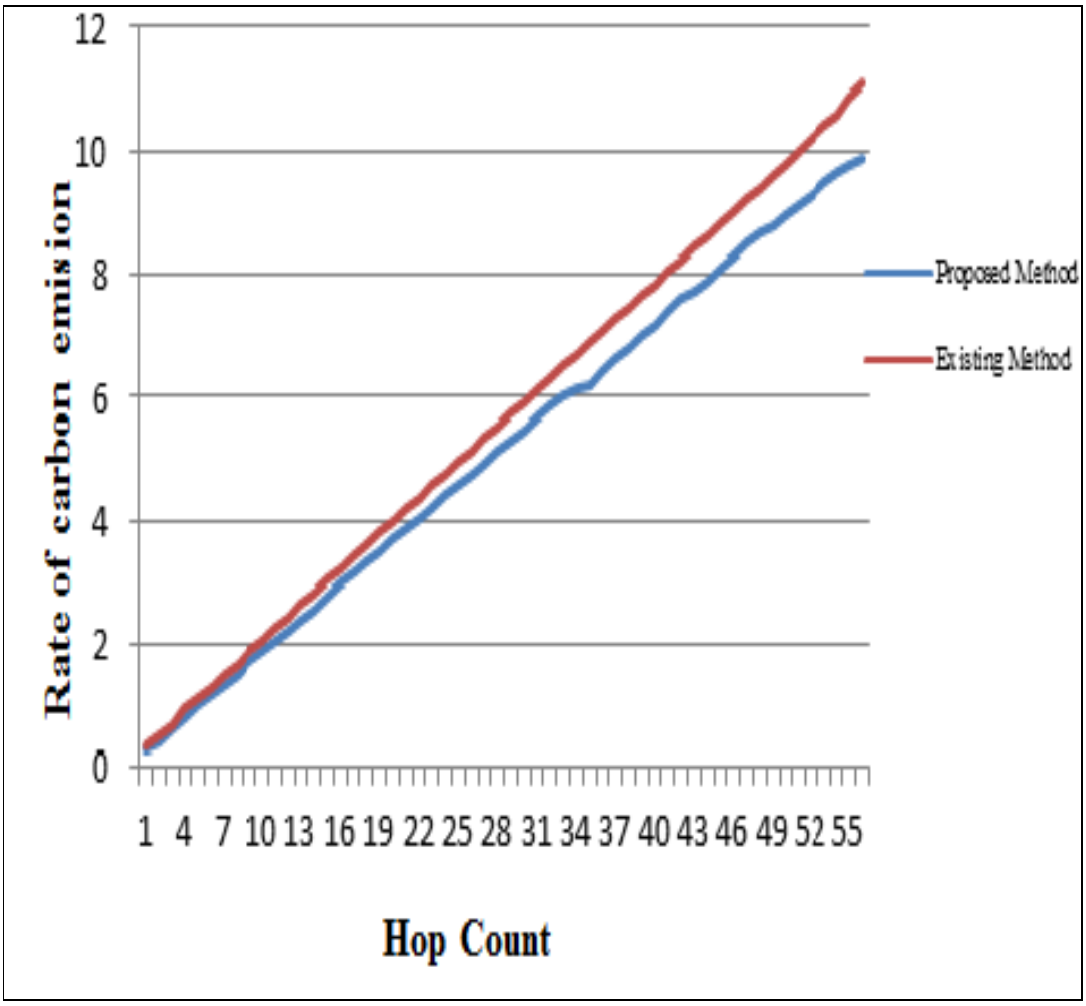


Figure 2. Carbon Emission Rate verses Hop count

7. Conclusion

While cloud computing has seen wide ranging acceptance from across several industry verticals, there exists issues such as load balancing, consolidation of servers, virtual machine migration policies and infrastructure, and energy management. These are relevant issues in the context of what we now see in the field of cloud computing, and which now need to be studied extensively. Of these, the core challenge is search optimization and distribution of processing tasks such that all the relevant nodes in a

cloud computing ecosystem shoulder an almost equal responsibility. Achieving this would result in consistent high levels of user satisfaction and a healthy ratio of resource utilization to availability of resources. Current techniques of balancing load have mostly stressed on reducing overhead, diminishing service response times, and improving performance. But none of these have taken into consideration consumption of energy and the factor of carbon emission. This, therefore, has given an impetus to design and develop an energy efficient model for load balancing coupled with search optimization. These are factors that can raise the level of cloud computing performance, and utilize available resources to the fullest extent possible. As a result of reducing energy consumption and reining in carbon emissions, achieving Green Computing will not be difficult.

References

[1] B. Sotomayor, R. S. Montero, I. M. Llorente and I. Foster, "Virtual infrastructure management in private and hybrid clouds," in IEEE Internet Computing, vol. 13, no. 5, **(2009)**, pp. 14-22.
[2] Nakrani and C. Tovey, "On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers", Adaptive Behavior 12, **(2004)**, pp. 223-240.
[3] D. Sarddar, S. Das and S. K. Sikdar, "Cost Analysis of Algorithm Based Billboard Manger Based Handover Method in LEO satellite Networks", (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 3, no. 11, **(2012)**.
[4] B. Radojevic and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments." In proc. 34th International Convention on MIPRO, IEEE, **(2011)**.
[5] Nishant, K. P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization." In proc. 14th International Conference on Computer Modelling and Simulation (UKSim), IEEE, **(2012)** March, pp. 3-8.
[6] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation." In proc. 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), IEEE, vol. 2, **(2010)** May, pp. 240-243.
[7] L. Kolb, A. Thor and E. Rahm, "Load Balancing for MapReduce- based Entity Resolution," in proc. 28th International Conference on Data Engineering (ICDE), IEEE, **(2012)**, pp. 618-629.
[8] T. Gunarathne, T.-L. Wu, J. Qiu and G. Fox, "MapReduce in the Clouds for Science," in proc. 2nd International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, **(2010)** November/December, pp. 565-572.
[9] J. Ni, Y. Huang, Z. Luan, J. Zhang and D. Qian, "Virtual machine mapping policy based on load balancing in private cloud environment," in proc. International Conference on Cloud and Service Computing (CSC), IEEE, **(2011)** December, pp. 292-295.
[10] X. Ren, R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on Cloud Computing and Intelligent Systems (CCIS), IEEE, **(2011)** September, pp. 220-224.
[11] R. Lee and B. Jeng, "Load-balancing tactics in cloud," in proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, **(2011)** October, pp. 447-454.
[12] J. Al-Jaroodi and N. Mohamed, "DDFTP: Dual-Direction FTP," in proc. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE, **(2011)** May, pp. 504-503.
[13] S.-C. Wang, K.-Q. Yan, W.-P. Liao and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in proc. 3rd International Conference on Computer Science and Information Technology (ICCSIT), IEEE, vol. 1, **(2010)** July, pp. 108-113.
[14] A. Sang, X. Wang, M. Madihian and R. D. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in Wireless Networks, vol. 14, no. 1, **(2008)** January, pp. 103-120.
[15] M. Randles, D. Lamb and A. Taleb-Bendiab, "A Comparative Study into Distributed Load balancing Algorithms for Cloud Computing" IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, **(2010)**, pp. 1-10.

Authors

Debabrata Sarddar, Assistant Professor in the Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nadia, West Bengal, INDIA. He has done Ph.D. at Jadavpur University. He completed his M. Tech in Computer Science & Engineering from DAVV, Indore in 2006, and his B.E in Computer Science & Engineering from NIT, Durgapur in 2001. He has published more than 75 research papers in different journals and conferences. His research interest includes wireless and mobile system and WSN, Cloud computing.

Rajesh Bose is currently pursuing Ph.D. from Kalyani University. He is an IT professional employed as Senior Project Engineer with Simplex Infrastructures Limited, Data Center, Kolkata. He received his degree in M. Tech. In Mobile Communication and Networking from WBUT in 2007. He received his degree in B.E. in Computer Science and Engineering from BPUT in 2004. He has also several global certifications under his belt. These are CCNA, CCNP-BCRAN, and CCA(Citrix Certified Administrator for Citrix Access Gateway 9 Enterprise Edition),CCA(Citrix Certified Administrator for Citrix Xen App 5 for Windows Server 2008).His research interests include cloud computing, wireless communication and networking.

Sudipta Sahana is an assistant professor of a renowned engineering college of West Bengal. For more than 4 years, he has worked in this region. He has passed his M.Tech degree in Software Engineering and B.Tech Degree in Information Technology from West Bengal University of Technology with a great CGPA/DGPA on 2010 and 2012 respectively. He is recently working in Ph.D. in the domain of “Cloud Computing”. He is a member of the Computer Science Teachers Association (CSTA), and also a member of International Association of Computer Science and Information Technology (IACSIT).

