Middle-East Journal of Scientific Research 16 (6): 805-813, 2013
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2013.16.06.11314

Prognostic Load Balancing Strategy for Latency Reduction in Mobile Cloud Computing

Mudassar Ahmad and Ibrahim A. Cheema

Universiti Teknologi Malaysia (UTM)

Abstract: In Mobile Cloud Computing (MCC), load balancing is essential to distribute the local workload evenly across all the nodes either statically or dynamically. A high level of user satisfaction and resource utilization ratio can be achieved by ensuring an efficient and fair allocation of all computing resources. In the absence of proper load balancing strategy/technique the growth of MCC will never go as per predictions. The appropriate load balancing helps in minimizing resource consumption, implementing fail-over, enabling scalability, avoiding bottlenecks. In this paper, a prognostic load balancing strategy is proposed and implemented for computational latency reduction in MCC. Also the results of proposed technique is compared with existing techniques. Finally this study concludes that the proposed predictive technique reduces associated overheads, service response time and improves performance. There are also Various parameters that are identified and used to compare the existing techniques.

Key words: Mobile Cloud Computing (MCC) · Cloud Computing (CC) · Load Balancing Strategy · Load Balancing Techniques · Load Balancing Matrices Experimental · Literary Analysis · Survey/Interview

INTRODUCTION

In last few years, applications targeted at mobile devices have started becoming abundant with applications in various categories such as entertainment, health, games, business, social networking, travel and news. The popularity of these is evident by browsing through mobile app download centers. The reason for this is that mobile computing is able to provide a tool to the user when and where it is needed irrespective of user movement, hence supporting location independence. Indeed, 'mobility' is one of the characteristics of a pervasive computing environment where the user is able to continue his/her work seamlessly regardless of his/her movement.

Recently this problem has been addressed by researchers though cloud computing (CC). CC can be defined as the aggregation of computing as a utility and software as a service [1]. where the applications are delivered as services over the Internet and the hardware and systems software in data centers provide those services [2]. Also called 'on demand computing', 'utility computing' or 'pay as you go computing', the concept behind CC is to offload computation to remote resource providers. The key strengths of CC can be described in terms of the services offered by cloud service providers: software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) [3].

Cloud computing (CC) has widely been adopted by the industry, though there are many existing issues like Load Balancing, Virtual Machine Migration, Server Consolidation, Energy Management, security, etc. that are not fully addressed [4]. Central to these issues is the issue of load balancing that is a mechanism to distribute the workload evenly to all the nodes in the whole cloud to achieve a high user satisfaction and resource utilization ratio. The same challenge is addressed by MCC also where multiple and variable computing requirement comes on datacenters. The present problem with MCC is that bottlenecks of the system which may occur due to load imbalance, computing resource distribution efficiently and fairly, Minimum resource consumption. So, Proper load balancing techniques not only helps in reducing costs but also making enterprises as per user satisfaction [5, 6]. Scalability, one of the very important features of MCC, is also affected by load balancing. Hence, improving resource utility and the performance of a distributed system in such a way will reduce the energy consumption require efficient load balancing.

Corresponding Author: Mudassar Ahmad, Universiti Teknologi Malaysia (UTM)

Long-connectivity application as a representative of Web applications is the research object of this study. The significant features of this type of application are that the users' requests maintain a long connection with web server in a period of time, but they take up very little CPU, memory and other indicators. Wait until a certain point in time arrives; the users will access the application almost at the same time, when indicators on a web server will increase instantly.

Typical of such application is spike in online shopping services. Spike is a sales promotion that Internet sellers release a number of ultra-low-price goods and buyers snapped up by the network at the same time. In web applications such as Internet auction, how to ensure that the back-end web server will not downtime because of overloading? Some of the examples is as the total no of users of web application suddenly increased for some amount of time such as 'Tatkal Ticket Scheme' of Indian Railways where the no of consumers of ticket reservation are increased so high leads to increase in server downtime and the web application slows down regardless of speed of network connectivity and processing capacity of server. At this point, the choice of load balancing strategies is vital.

The paper mainly focuses on implementation of prognostic load balancing strategy named Amplified - ESBWLC which is based on Simple Exponential Smoothing Forecast Technique. A simple exponential smoothing forecast model is a very popular model used to produce a smoothed Time Series. In simple exponential smoothing, however, a "smoothing parameter" or "smoothing constant" is used to determine the weights assigned to the observations. This research shows that how the selection of data center based on predicted response time of available data centers leads to minimization of load on data centers and reduction in latency felt by users.

The rest of the paper is organized as follows: Section II, focuses comparison between cloud computing and MCC. Section III, discusses about the existing load balancing algorithms. Section IV, represents algorithm and pseudo code of proposed algorithm. Section 5 shows the implementation details of algorithm and says about working environment. Section 6 shows the results and comparison analysis. Finally Section 7 concludes the paper with future scope.

Cloud Computing (CC) vs Mobile Cloud Computing (MCC)

Cloud Computing (CC): ''Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services'' [2]. A cluster of computer hardware and software that offer the services to the general public makes up a 'public cloud'. Computing is therefore offered as a utility much like electricity, water, gas etc. where you only pay per use. Virtualization of resources is a key requirement for a cloud provider for it is needed to create the illusion of infinite resources to the cloud user. Ambrust *et al*. [2] holds the view that ''different utility computing offerings will be distinguished based on the level of abstraction presented to the programmer and the level of management of the resources''.

Mobile Cloud Computing (MCC): There are several existing definitions of MCC and different research alludes to different concepts of the 'mobile cloud':

- Commonly, the term MCC means to run an application such as Google's Gmail for Mobile on a remote resource rich server while the mobile device acts like a thin client connecting over to the remote server through 3G. This paper focuses primarily on this type of work.
- Another approach is to consider other mobile devices themselves too as [7] resource providers of the cloud making up a mobile peerto- peer network as in. Thus, the collective resources of the various mobile devices in the local vicinity and other stationary devices too if available, will be utilized.
- The cloudlet concept proposed by Satyanarayanan [8] is another approach to MCC, Where the mobile device offloads its workload to a local 'cloudlet' comprised of several multi-core computers with connectivity to the remote cloud servers. These cloudlets would be situated in common areas such as coffee shops, college campus etc. so that mobile devices can connect and function as a thin client to the cloudlet as opposed to a remote cloud server which would present latency and bandwidth issues.

MCC is the combination of cloud computing and mobile networks to bring benefits for mobile users, network operators, as well as cloud providers. Cloud computing exists when tasks and data are kept on the Internet rather than on individual devices, providing on-demand access. MCC can involve other mobile devices and/or servers accessed via the Internet. MCC would also be based under the basic cloud computing concepts. As discussed by Mei *et al*. in [9] there are

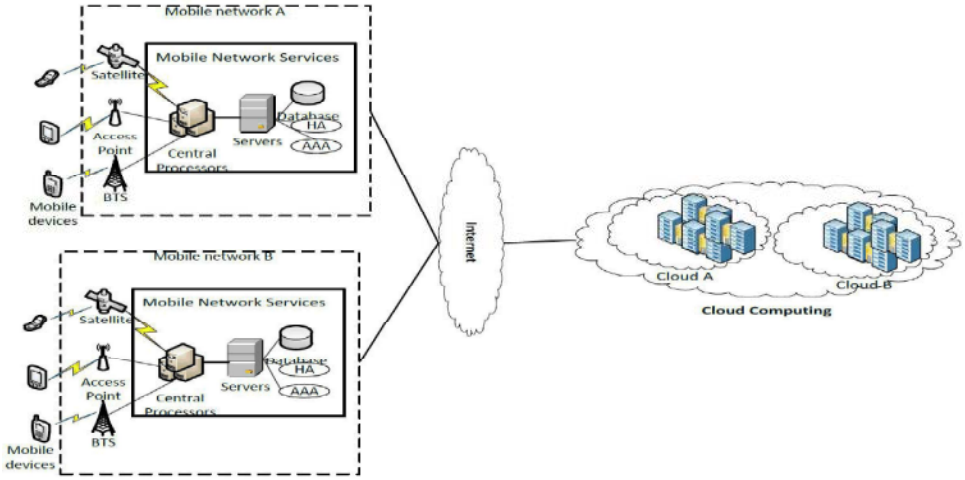


Fig. 1: Mobile Cloud Computing Architecture

certain requirements that need to be met in a mobile cloud such as adaptability, scalability, availability and self-awareness. So Load Balancing [10-12] will play key role in MCC to ensure availability and avoidance of bottleneck.

Related Work: Load Balancing Algorithms in cloud computing environment generally divide in two categories [13] as Static Load Balancing Algorithms and Dynamic Load Balancing Algorithm [14].

Static Load Balancing Algorithm: Static Load balancing algorithms assign the tasks to the nodes based only on the ability of the node to process new requests but they do not consider dynamic changes of these attributes at run-time, in addition, these algorithms cannot adapt to load changes during run-time. The process is based solely on prior knowledge of node's processing power, memory and storage capacity and most recent known communication performance. Round Robin (RR) and Weighted Round Robin (WRR) are most commonly Static Load Balancing Algorithm Used in Cloud Computing. Round Robin Algorithm does not consider server availability, server load, the distance between clients and servers and other factors. In this algorithm server selection for upcoming request is done in sequential fashion. The main problem with this approach is inconsistent server performance which is overcome by WRR. In WRR the weights are added to servers and according to amount of traffic directed to servers however for long time connections it causes load tilt.

Dynamic Load Balancing Algorithm: Dynamic Load Balancing Algorithms considers a combination of knowledge based on prior gathered information about the nodes in the Cloud and run-time properties collected as the selected nodes process the task's components. These algorithms assign the tasks and may dynamically reassign them to the nodes based on the attributes gathered and calculated. However, they are more accurate and could result in more efficient load balancing than Static Load Balancing Algorithm. Least Connection (LC) and Weighted Least Connection (WLC) are commonly used dynamic load balancing algorithm. In LC the total no of connections on server are identified at run time and the incoming request is sent to server with least number of connections. However LC does not consider service capability, the distance between clients and servers and other factors. WLC considers both weight assigned to service node W(Si) and current number of connection of service node C(Si) [15][16]. The problem with WLC is as time progresses static weight cannot be corrected and the node is bound to deviate from the actual load condition, resulting in load imbalance. Xiaona Ren et. al. [17] proposed prediction based algorithm called as Exponential Smoothing forecast- Based on Weighted Least- Connection (**ESBWLC**) which can handle long-connectivity applications well. In this algorithm the load on server is calculated from parameters like CPU utilization, memory usage, no of connections, size of disk occupation. Then load per processor (Load/p) is calculated and this algorithm uses (Load/p) as historical training set, establishes prediction model and predicts the value of next moment. The limitation with this algorithm is this algorithm does not consider the distance between client and servers, network delay and other factors.

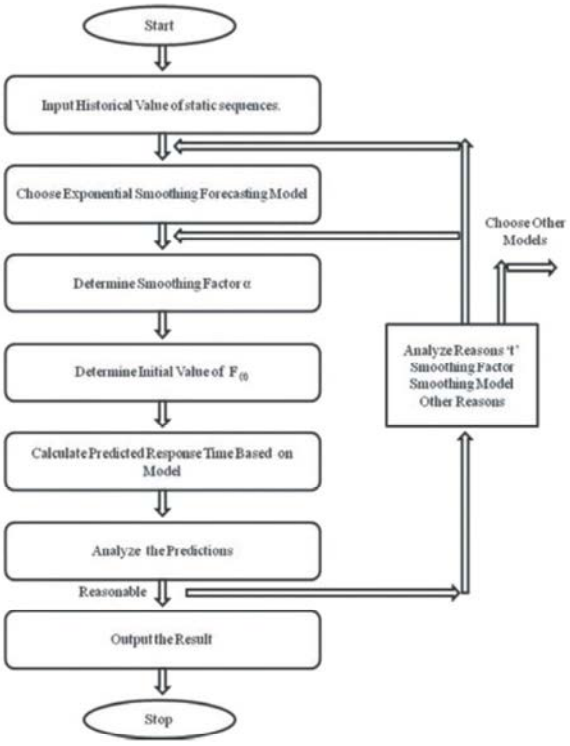


Fig. 2: Flowchart of Proposed Algorithm Amplified - ESBWLC

Paper proposes algorithm named **Amplified-ESBWLC** which overcomes above limitation. In this algorithm we directly calculate the response time we get at client side. This got response time is store for further reference. The response time at time instance 't+1' is predicted by using current response time at time instance 't' and previously predicted response time for time instance 't'.

Proposed Prognostic Algoritm: The presented prediction algorithm is based on simple exponential smoothing forecast model. Simple exponential smoothing forecasting method is one of prediction algorithms based on time series. The algorithm takes advantage of all historical data and distinguishes them through the smoothing factor to let recent data make a greater impact on the predictive value than long-term data. We are sending the observed response time at client side from the current period and the forecast response time from the previous period to come up with a forecast response time for the current period.

Let Assuming time series as x1, x2, x3 … the formula of single exponential smoothing forecasting model is

$$F_{(t)} = \alpha * X_{(t)} + (1-\alpha) * F_{(t-1)}$$

In the formula,

- F (t) represents prediction value of Response Time at t-period.
- X (t) represents observation value of Response Time t-period.
- α represents the smoothing factor ($0 < \alpha < 1$).

The flowchart for Proposed Prediction Algorithm:

Algorithm:

- Input historical value of statistical Response Time as a training set.
- Use single exponential smoothing as a forecasting model.
- Determine the smoothing factor α.
- Determine the initial value of predictive model.
- Calculate based on the prediction model.
- Analyze the predictions.

MATERIALS AND METHODS

The working environment for cloud computing where the proposed algorithm is implemented is done using cloud analyst simulator which is built above "CloudSim", "GridSim" and "SimJava". Cloud-Analyst is built on the top of Cloud-sim. Cloud-sim is developed on the top of the Grid-sim.

- Application users - There is the requirement of autonomous entities to act as traffic generators and behavior needs to be configurable.
- Internet - It is introduced to model the realistically data transmission across Internet with network delays and bandwidth restrictions.
- Simulation defined by time period - In Cloud-sim, the process takes place based on the pre-defined events. Here, in Cloud-Analyst, there is a need to generate events until the set timeperiod expires.
- Service Brokers - DataCeneterBroker in CloudSim performs VM management in multiple data centers and routing traffic to appropriate data centers. These two main responsibilities were segregated and assigned to DataCenterController and CloudAppServiceBroker in Cloud-Analyst.

Pseudo Code for Proposed Prognostic Algorithm:

- Calculate the current response time for each data center and store it.

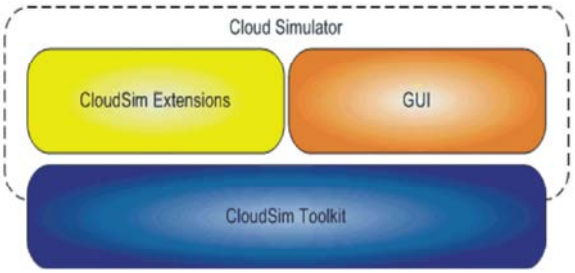


Fig. 3: Cloud-Analyst built on top of Cloud-Sim toolkit

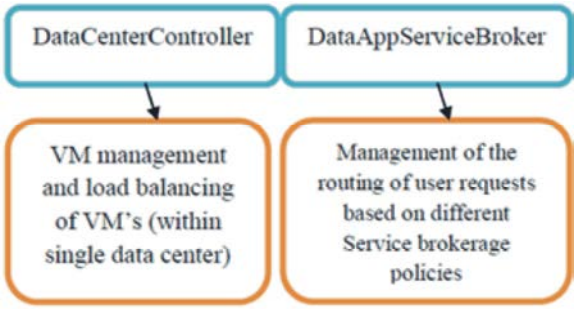


Fig. 4: Responsibilities- Segregation

- Determine the initial value of prediction model.

  for : initial time
  predicted latency = current latency;

- Calculate the predicted latency for next instance using formula

  for : each datacenter
  do
  PredictLatencyNew = alpha * CurrLatency + (1-alpha) * PreviousPredictedLatency;
  Store predictLatencyNew
  Done

- Calculte the minimum PredictLatencyNew from available list of datacenters.
  for (each DataCenter)
  do
  if(minlatency<min)
  min=minlatency;
  done

- Select datacenter with minimum PredictLatencyNew as choosen data center for upcoming service request.
- Return choosendatacenter.

RESULTS AND DISCUSSION

As the algorithm Amplified – ESBWLC is implemented using simulation Cloud-Analyst. The scenario is taken where the data centers are located at different regions with user bases requesting services from different regions. The simulation runs Approximately 60 min amount of time and the final result screen shown as,

In above screen the lines shows that the user base is requesting service from corresponding data center or server. The values shown at boxes at each user bases represents the latency observed by respected user base. The values are the minimum latency calculated at client side wile requesting service from data center in the duration of simulation was running, similarly it shows the maximum latency and the average latency from above two calculated values.

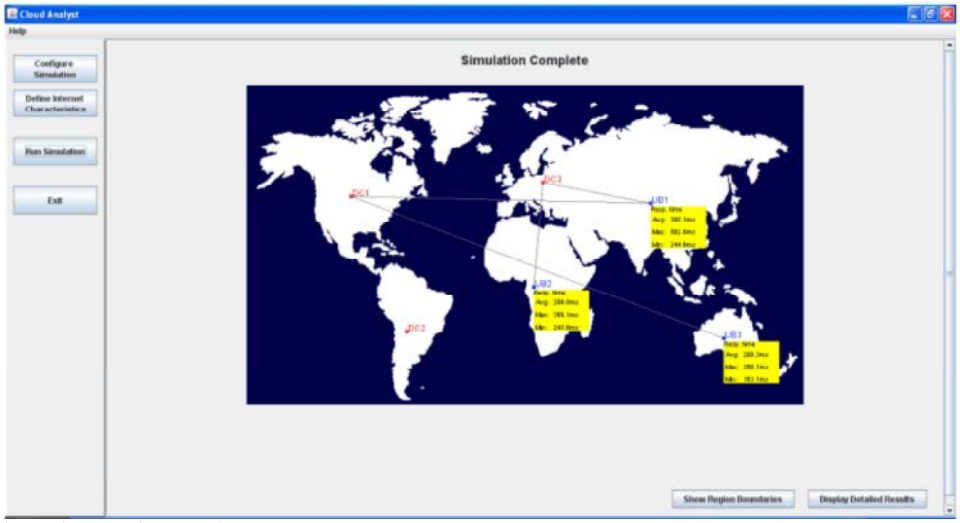


Fig. 5: Cloud Analyst Main Result Screen

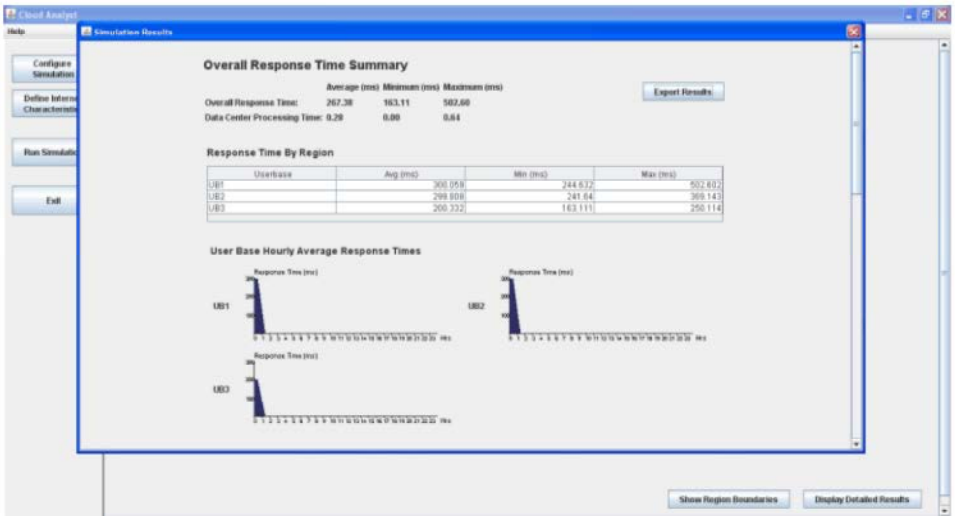


Fig. 6: Result Screen Showing UB Response Time

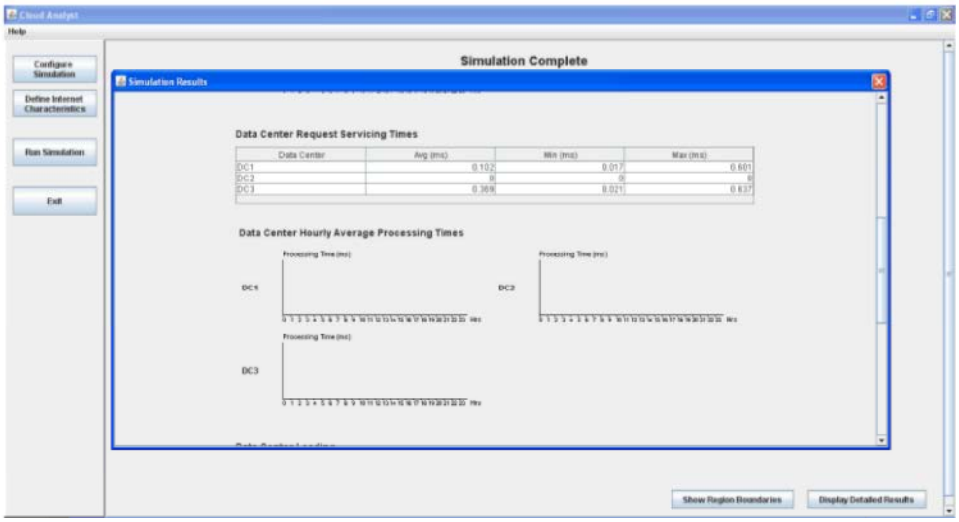


Fig. 7: Result Screen showing DC Service Timing

The Simulation also shows the user base hourly average response time plotted on graph as shown in above figure.

The Simulation calculates the datacenter request service timing in numbers which is in micro seconds. Also it calculates hourly average processing times which are plotted on graph as shown in above figure.

The simulations also shows total loading at each data centers which is plotted on graph on hourly basis as shown in above figure.

The total costing at each data center is calculated as shown above. Total cost inclusive of both virtual machine cost and data transfer cost i.e.

Total Cost = Virtual Machine Cost + Data Transfer Cost.

Comparitive Analysis

Experiment 1 - Verifying the Prediction of Service Capability: Different values of α will lead to different predictions. The Experiments is done with different values of alpha and calculated Response Time is analyzed. The experiment puts the predictive cases for different values of α such as 0.3, 0.5, 0.7, 0.9 and observed response time is plotted on graph for different cases. The resultant graph are shown below.

In above graph, abscissa represents different scenarios and vertical axis represents observed response time in (Ms). The lines with different colors represent different values of α. It can be seen from the figures that line with value of α is 0.9 represents minimum response time at each scenario, which means that the highest prediction accuracy appears when value of α is 0.9.

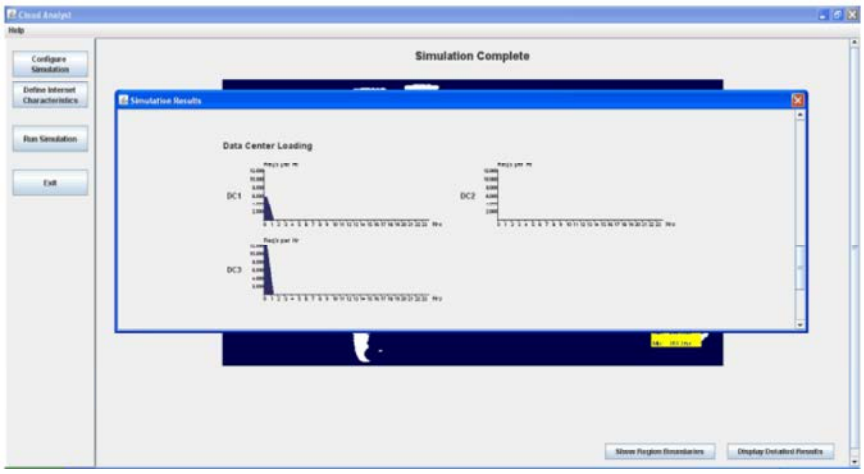


Fig. 8: Result Screen showing DC Loading

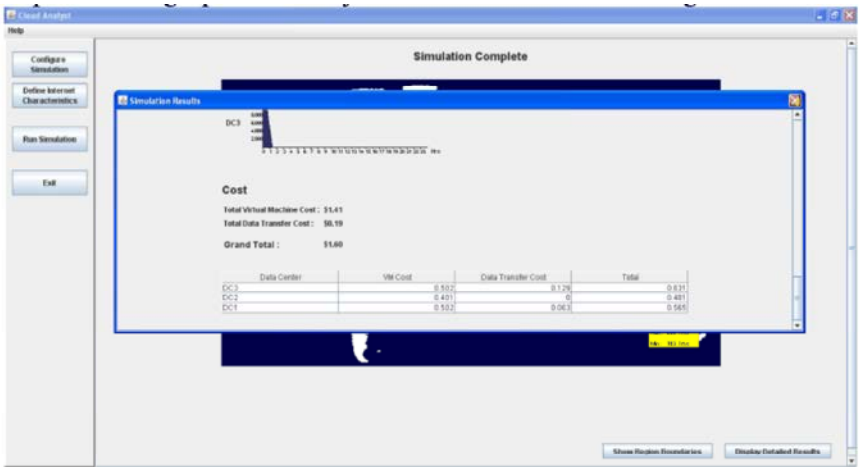
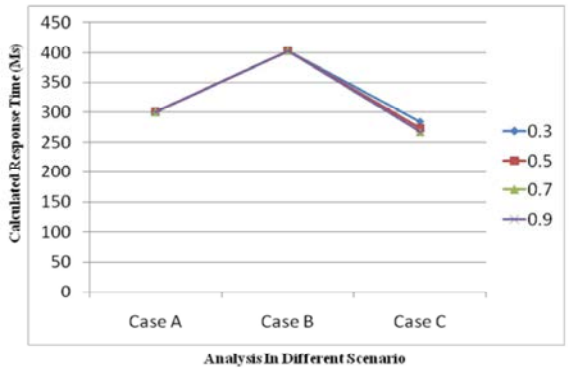


Fig. 9: Result Screen showing Total Costing



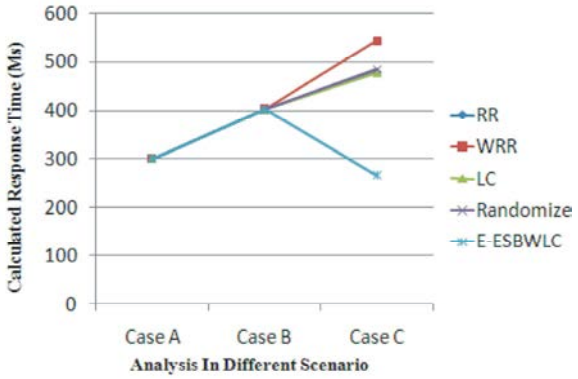


Fig. 10: Graph Showing Prediction Accuracy

Experiment 2 – Comparison with Different Available Algorithm in Terms of Latency: The latency is calculated for different scenarios using different Load Balancing algorithms. The observed latency is compared and analyzed to find out the algorithm which gives minimum latency at client side. The experiment uses different Load Balancing algorithms as Round Robin, Weighted Round Robin, Least Connection and Weighted Least Connection.

In above graph, abscissa represents different scenarios and vertical axis represents observed response time in (Ms). The lines with different colors represent different algorithms implemented. It can be seen from the figures that line with algorithm Amplified - ESBWLC represents minimum latency at each scenario, which means that the latency is reduced with the proposed algorithm named Amplified - ESBWLC.

CONCLUSION

Considering the unique features of long-connectivity applications, an algorithm is proposed Amplified - ESBWLC. ESBWLC optimizes the number of connections and adds single exponential smoothing forecasting. Finally, experiments show that prediction accuracy is maximum when value of α is 0.9 and Amplified-ESBWLC results in reduction in computational latency at client side. The future work may include prediction based load balancing algorithm for multimedia and live streaming web applications.

REFERENCES

1. Vogels, "A head in the clouds the power of infrastructureas as service", Proceedings of the 1st Workshop on Cloud Computing and Applications, CCA'08.
2. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I. Stoica, 2009. Above the clouds : A Berkeley view of cloud computing, Technical Report UCB / EECS -2009-28,2009.
3. Carolan, J., S. Gaede, J. Baty, G. Brunette, A. Licht, J. Remmell, L. Tucker and J. Weise, 2009. Introduction to cloud computing architecture, whitepaper,
4. Rimal, B.P., E. Choi and I. Lumb, XXXX. A Taxonomy and Survey of loud Computing Systems, IEEE 5th International Joint Conference on INC, IMS and IDC, pp: 44-51.
5. Mata-Toledo, R. and P. Gupta, 2010. Cloud Load Balancing Techniques: A Step towards Green, Journal of Technology Research, 2(1): 1-8.
6. Alakeel, A.M., 2010. A Fuzzy Dynamic Load Balancing Algorithm for Homogenous Distributed Systems, International Journal of Computer Science and Network Security, 10(6): 153-160.
7. Marinelli, E., 2009. Hyrax: cloud computing on mobile devices using Map Reduce, Master's Thesis, Carnegie Mellon University.
8. Satyanarayanan, M., P. Bahl, R. Caceres and N. Davies, 2009. The case for VM-based cloudlets in mobile computing, IEEE Pervasive Computing, 8: 14-23.
9. Mei,W., T. Chan, A. Tse, 2008. "Tale of clouds : paradigm comparis on sandsome thoughtson research issues", ProceedingsoftheAsia-PacificServices Computing Conference, APSCC' IEEE, 464-469.
10. Xiaona Ren, Rongheng Lin and Hua Zou, 2011. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast, IEEE ccis2011, pp: 220-225.
11. Qingfeng Liu, Xie Jian and Jicheng Hu, 2009. An optimized solution for mobile environment using mobile cloud computing, IEEE International Conference on Wireless Communications, Networking and Mobile Computing.
12. Qi Cao, Zhi-Bo Wei and Wen-Mao Gong, 2009. An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing, International Conference on Bioinformatics and Biomedical Engineering, pp: 1-3.
13. Luo Yongjun, Li Xiaole and Sun Ruxiang, 2008. Load Balancing Algorithms Overview, Information Development and Economy, 18: 134-136.
14. Randles, M., D. Lamb and A. Taleb-Bendiab, 2010, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, pp: 551-556.
15. Tian Shaoliang, Zuo Ming and Wu Shaowei, 2007. An improved load balancing algorithm based on dynamic feedback, Computer Engineering and Design, 28: 572-573.
16. Zheng Qi., 2006. Load balancing algorithm based on dynamic feedback, Computer Age, pp: 49-51.
17. Xiaona Ren, Rongheng Lin and Hua Zou, 2011. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast, IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), pp: 220-224.
18. Elghoneimy, E., O. Bouhali and H. Alnuweiri, 2012. Resource allocation and scheduling in cloud computing, International Conference on Computing, Networking and Communications (ICNC), pp: 309-314.
19. Khazaei, H. and J. Misic, 2012. Performance Analysis of Cloud Computing Centers Using M/G/m/m+r Queuing Systems", IEEE Transactions on parallel and distributed systems, 23(5): 936-943.

20. Murata, Y., R. Egawa, M. Higashida and H. Kobayashi, 2010. A History-Based Job Scheduling Mechanism for the Vector Computing Cloud, IEEE/IPSJ International Symposium on Applications and the Internet(SAIT), pp: 125-128.
21. Chih-Yung, Chen. and Tseng Hsiang-Yi. 2012. An Exploration of the Optimization of Excutive Scheduling in the Cloud Computing, IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp: 1316-1319.
22. Hu, Wu., Tang. Zhuo and Li Renfa, 2012. A priority constrained scheduling strategy of multiple workflows for cloud computing, IEEE international conference on Advanced Communication Technology (ICACT), pp: 1086-1089.
23. Chih-Yung Chen and Tseng Hsiang-Yi, 2012. An Exploration of the Optimization of Excutive Scheduling in the Cloud Computing, IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp; 1316-1319.
24. Gotoda, S., M. Ito and N. Shibata, 2012. Task Scheduling Algorithm for Multicore Processor System for Minimizing Recovery Time in Case of Single Node Fault, IEEE/ACM International Symposium Cluster, Cloud and Grid Computing (CCGrid), pp: 260-267.