Internet of Things and Cloud Computing
2019; 7(1): 25-30
http://www.sciencepublishinggroup.com/j/iotcc
doi: 10.11648/j.iotcc.20190701.14
ISSN: 2376-7715 (Print); ISSN: 2376-7731 (Online)

Modifying Broker Policy for Better Distribution of the Load Over Geo-distributed Datacenters

Louai Sheikhani, Weichao Ding, Jonathan Talwana, Chunhua Gu*

School of Information Science and Engineering/East China University of Science and Technology, Shanghai, China

Email address:
Loaiinho@windowslive.com (L. Sheijhani), weich@ecust.edu.cn (Weichao Ding), tal.jonah@hotmail.com (J. Talwana), chgu@ecust.edu.cn (Chunhua Gu)

*Corresponding author

To cite this article:
Louai Sheikhani, Weichao Ding, Jonathan Talwana, Chunhua Gu. Modifying Broker Policy for Better Distribution of the Load Over Geo-distributed Datacenters. *Internet of Things and Cloud Computing*. Vol. 7, No. 1, 2019, pp. 25-30. doi: 10.11648/j.iotcc.20190701.14

Received: March 14, 2019; Accepted: June 11, 2019; Published: June 15, 2019

Abstract: As an increasing number of businesses move toward Cloud based services, issues such as reduce response time, optimize cost, and load balance over data centers are important factor that need to be studied. Selecting the suitable data center to handle the user request is affecting those factors directly. The Broker policy determines which data center should service the request from each user base; so choosing appropriate policy can improve the performance noticeably. One of the benchmarks policies is service proximity-based that routing the request to the data center, which has lowest network latency or minimum transmission delay from a user base. If there are more than one data centers in a region in close proximity, then one of the data centers is selected at random to service the incoming request. However, other factors such as cost, workload, number of virtual machines, processing time etc., are not taken into consideration. Randomly selected data center gives undesirable results in terms of response time, data processing time, cost, and other parameters. this work propose modifying that policy by applying new schedule algorithm that control the load balance. the results showed that the using of this algorithm instead of the random selection would improve the distribution of the workload over the available datacenters noticeably.

Keywords: Cloud Computing, Datacenter Selection, Broker Policy; Min-min Scheduling Algorithm, Load Balance

1. Introduction

Cloud computing represents a new way to deploy computing technology to give users the ability to access, work on, share, and store information using the Internet. The cloud itself is a network of data centers, each composed of many thousands of computers working together that can perform the functions of software on a personal or business computer by providing users access to powerful applications, platforms, and services delivered over the Internet.

Data center is the main resource of the cloud that holds the computing and storage server with number of host machine. The main aim of data centers is to maximize the utilization of computing resources such as storage, CPUs, and network bandwidth as service-by-service providers at less cost. The optimization models aims to optimize both resource centric such as utilization, availability, reliability and user centric like response time, budget spent fairness.

With the increase and rapid usage of the cloud computing, it's become very important to pay attention of the characteristics of data center and it's load. Choosing the appropriate data center to handle the user request is Broker policy responsibility, service proximity-based policy is a benchmarks policy, in scenario that include more than one data center in the same region (geographical region), the Service Proximity Based send the user request randomly to one of these data center.

The random selection of the data center is not a good policy because it sending the user request to the data center without any consideration of the data size or more importantly the data center status which may lead to overload the data center while data centers in better status to handle the user request.

2. Realted Work

Selecting the Data center that can handle the user request

is the main challenge to any broker policy, because during choosing the proper data center, many factors should be taking in consideration such as time, cost and the load distribution over the available data centers in the cloud system. The Proximity-based routing selects the closest region depending upon the least network latency and from that region it selects the data center randomly. However, this policy has many limitations that affect the response time and may lead to overwhelm a certain data center.

We notice variations of service broker policy gives better performance than existing service proximity based policy. Chudasama et al. (2010) presents an enhanced proximity-based routing policy that avoids the direct selection of nearest data center [1]. If more than one data center is in the same region, then the data center having less cost will be selected. Ram Prasad et al. have studied divisible load scheduling theory in cloud computing [2]. Kumar Nishant et al. proposed Ant colony optimization to improve the load balance [3]. In Jasmin James et al. have proposed a better allocation policy called weighted active monitoring load balancing by assigning weights to each VM [4]. Soumya Ray et al. have identified qualitative components for simulation in cloud environment and then based on these components; he has explained execution analysis of load balancing algorithms [5]. Ajith Singh. N et al. have suggested semi-distributed load balancing solution in cloud-based infrastructure [6]. Authors have demonstrated efficient load balancing in cloud computing using Fuzzy logic [7]. H. Mehta et al. have formulated a new content aware load balancing policy named as workload and client aware policy (WCAP). It uses a unique and special property called UPS that defines the requests as well as computing nodes. USP helps the scheduler to decide the best suitable node for the processing the requests [8]. Y. Lua et al. have explained a Join Idle Queue load balancing algorithm for dynamically scalable web services which 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 [9]. J. Hu et al. have investigated the problem of scheduling on load balancing on VM resources that uses historical data and current state of the system [10]. T. S. Wang et al. have formulated a two-phase scheduling algorithm which combines OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms to utilize better executing efficiency and maintain the load balancing of the system [11].

3. Problem Definition

In cloud, from user's end, the important factors are cost optimization and provider that provides utility to the user's need. Thus routing of user's request is a very important aspect in cloud to understand how the user request handle, then we can defined the problem that arise from applying certain policy that route the request. Figure 1 shows the routing of users' requests.

Internet Cloudlet is created by User Base with appropriate parameters such as application ID and name of User Base (for routing back the RESPONSE). REQUEST is sent to the Internet with zero delay. Internet requests service broker to select an appropriate data center depending on the Service broker policy used. Once the Internet receives the information about which data center is to be used, Internet sends the request to that data center after adding appropriate network delay associated with that request. Depending on the load balancing policy, the data center controller routes the request to suitable virtual machine for processing. After processing the REQUEST, RESPONSE is sent to the Internet by selected data center. Then Internet adds network delay to the RESPONSE and sends it to User Base using the originator field in the Cloudlet information.

From the routing of the user requests it is quite evitable that many of the issues arise while:

Selecting the appropriate data center: And this is the responsibility of the broker policy, while having multiple polices have major effect on the performance. Choosing appropriate data center by applying appropriate broker policy is an important step toward providing better performance specially in terms of distributing the load over available datacenters. Presenting appropriate broker algorithm is the work of research.

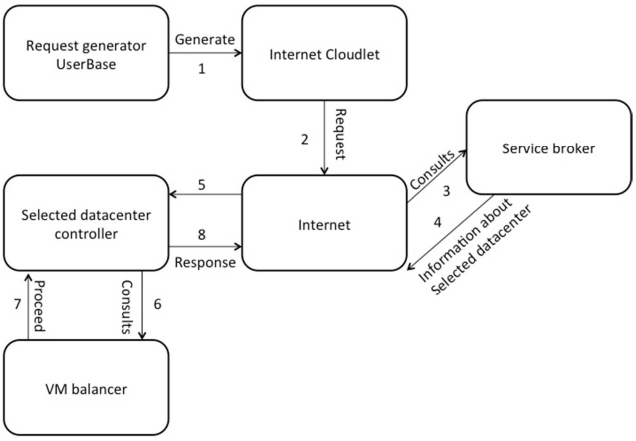

Figure 1. *Routing of the user request.*

Selecting appropriate VM: After selecting the data center it's important to select appropriate VM, this selection will affect directly the load balance within the data center. Various load-balancing techniques are present and proposed to enhance the cloud performance.

The problems may arise from applying some broker policy that may route all the requests to only one data center. As a result, only one data center is highly loaded and others are not. The situation may arise that all the requests may go to only one data center. This scenario may happened if the used policy was proximity based policy that route the user request to the closet data center, but if there are more than one Data center in the same region, the request directed to a random data center.

3.1. Proximity Based Policy

In order to explore the limitation of this algorithm the following steps present how this algorthim works, the following steps show how Service Proximity Based handle the user request [12]:

1) Service Proximity Service Broker maintains an index table of all Data Centers indexed by their region.
2) When the user request is received the Service Proximity Service broker retrieves the sender geographical region and queries for the region proximity list for that region from the Internet Characteristics.
3) The broker then route the sender request to the first earliest/highest region in the proximity list. If more than one data center is located in a region, one is selected randomly.

3.2. Service Proximity Based Drawbacks

The main problem with service proximity-based routing is the random selection of data center when there are more than one data centers present in a particular region with low latency; the results are different even though configurations are kept same. In addition, there is a high probability that the resources that are present are not utilized to their deliverable capability. Also it is possible that the selected data center will increase the response time or might have higher workload or may be of greater cost as compared to those available in same region.

The aim of this study is to show that using a proper scheduling algorithm can guarntee better load distribution over the datacenters in the cloud.

4. Proposed Solution

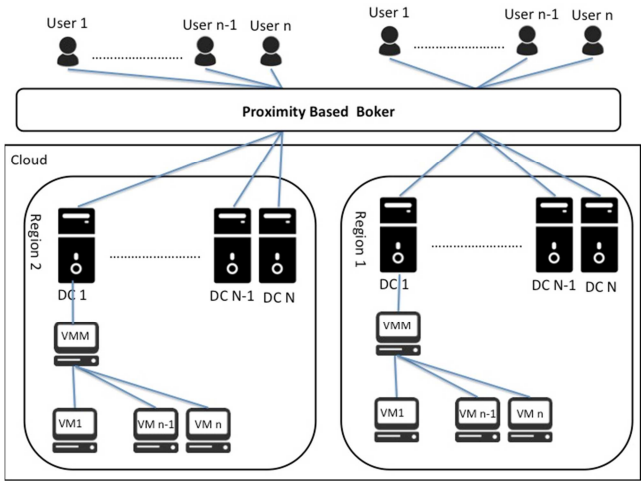

Figure 2. *System architecture.*

In previous study [13] we showed that using mim-min scheduling algorithm instead of random selection would improve the response time noticeably, we will use the same solution to prove that our proposed model not only affect the response time but also can control the load distribution over the whole cloud system, The target system (as shown in Figure 2) in this study is cloud provider that it consist of multiple geo-distributed data centers, those data centers are connected to the users upon the proximity broker, this broker policy is responsible of direct the user request to the data center, this policy route the request randomly to the closet data center.

However, if there is more than one data center in the same geographical region, this policy lead to poor results in term of response time and load over available data centers. So the aim of the proposed solution is to remove this random selection by applying schedule algorithm that can distribute the load over the available data centers in the given region, the proposed min-min based algorithm to improve the current proximity policy.

4.1. Workflow of the Proposed Solution

Our proposed model work according to the following phases:

Phase 1: First computes the completion time of every task on each machine and then for every task select the machine that processes the tasks in minimum possible time. Phase 2: Among all the tasks in Meta task the task with minimum completion time is selected and is assigned to machine on which minimum execution time is expected. The task is removed from the list of Meta Task and the procedure continues until Meta Task list is empty.

Phase 2: Among all the tasks in Meta task the task with minimum completion time is selected and is assigned to machine on which minimum execution time is expected. The task is removed from the list of Meta Task and the procedure continues until Meta Task list is empty.

4.2. Proposed Algorithm Description

The scheduling algorithm takes three kinds of inputs:

1) The task set: which contain the tasks need to be scheduling over the data center (s) in the region that the broker responsible of.
2) The data center (s): which include the all data centers locate in a specific region.
3) Execute Time Matrix (ETM): the execute time matrix which is a matrix of M×N indicate the execution time of M types of tasks running on N types of DCs, for example the entry *eij* in E indicate the required execution time of task type when running on DC type *j*.

Algorithm 1 presents the detailed step to perform the min-min scheduling approach. finish time can not be calculated before the task executed but can just calculate the expected finish time depending on the executable length (MI) and the corresponding server processor speed so once this value is optaind we calculate the expected finish time as following:

1) The task arrives in a form of Internet cloudlet that the size of the task defined by its Executable instruction length (MI) and the size (MB).
2) The recourses (DCs) are defined by its processing speed (MIPS) and bandwidth (Mbps).
3) Assuming having a set of n tasks (T1, T2, T3.... Tn)

needed to be scheduling onto m available resources (R1, R2, R3.... Rm) to calculate the expected time to process the task on each of the resources using equation 1:

$$Ct_{ij} = Et_{ij} + rt_j \quad (1)$$

Where Ct_{ji} is expected running time of task *i* on resource *j*, and r_{tj} indicate the ready time of resource R_j and Et_{ij} represents the execution time of task Ti on resource R_j.

4) So each entity of the ETM matrix is computed as that equation then the algorithm choose the entity with min value, and according to that value assign it to the right data center

Algorithm 1. The proposed scheduling algorithm.

Input: set of tasks, *m* data centers, ETM matrix.
Output: the schedule plan

Initiate the task set P.
While there are tasks not assigned do
  Update task set P.
  For *i*:task v_i do
    Pull all Data centers status.
    Get the earliest resource available time.
    Find the Datacenter $D_{min}(v_i)$ giving the earliest finish time of *vi*.
  End For
  Find the task-Data center pair(v_k,$D_{min}(v_k)$) with the earliest finish time.
  Assign task v_k to cloud $D_{min}(v_k)$.
  Remove v_k from P.
  Update the task set P
End While

5. Evaluations and Analysis

In a real-time environment, the effect of different factors on cloud environments is difficult to determine, costly to perform and risky to apply. For this reason, various simulation tools are used to model and analyze cloud computing environment and applications, graphically analyzing the results before the actual deployment of clouds. This section will present the simulation configurations that been used to evaluate service proximity based routing algorithm, with the proposed algorithm using Cloud Analyst [14] tool, and then the simulation results will be presented for those two algorithms in order to compare.

To explore the proposed algorithm ability we perform the simulation using 2, 3 DCs respectively, then a comparsaion between the proposed policy's results and the proximity based results will be done.

5.1. Load Evaluation Metrics

To show the efficiency of the proposed algorithm comparing to proximity-based in term of load balancing Load balance metrics are used. Load balance metrics characterize how unevenly work is distributed, which provide a detailed picture of load distribution that can indicate whether a distribution has a few highly loaded outliers or many slightly imbalanced datacenters The *percent imbalance* metric, λ (presented in equation 2), is most commonly used:

$$\lambda = \left(\frac{L_{max}}{\bar{L}} - 1\right) \times 100\% \quad (2)$$

Where L_{max} is the maximum load on any datacenter and *L* is the mean load over all datacenters. This metric measures the performance lost to imbalanced load or, conversely, the performance that could be reclaimed by balancing the load. Percent imbalance measures the *severity* of load imbalance.

Another common statistical moments, standard deviation σ, skewness *g*, where *n* is the number of datacenters and L_i is the load on the i^{th} datacenter. Those two metrics are demonstrated in equation 3 and equation 4

$$\sigma = \sqrt[2]{\frac{1}{n}\sum_{i=1}^{n}(L_i - \bar{L})^2} \quad (3)$$

$$g = \sqrt[2]{\sum_{i-1}^{n}(\frac{L_i}{\bar{L}} - 1)^2} \quad (4)$$

The bigger the σ of load balancing is, the more unbalanced the load will be. While Skewness *g* is the measure of unevenness resource utilization of a server, the higher Skewness means that relatively few processes have higher than average load; a normal distribution of load implies Skewness of 0.

5.2. Experiments Configuration

User Base configuration: The user base models a group of users that is considered to be single unit in the simulation, the user base is responsible for generating the traffic in the simulation. The user base represents a single user instead of a group of users, but ideally user base should be represents a large number of users for the efficiency of simulation. Since the scope is to compare the new policy with the existing proximity policy we have to create users that belong to same geographical region. Table 1 shows the user base configurations.

Table 1. *User base configuration.*

Name	Request per user	Data size per request	Peak hour start	Peak hour end	Average peak users	Average off peak user
UB1	60	100	3	9	4000	400
UB2	80	100	2	7	3000	300
UB3	100	120	3	7	5000	500
UB4	40	140	2	9	3000	300

Datacenter configurations: Datacenter is responsible of manage the data activities and routs the requests that is received by the user to the VM depending of the applied policy, We can ignore some parameters like the cost/VM (USD/h) and the data transfer cost (USD/GB) because our scope is not related to the, the most important parameters is the DC MIPS

and the bandwidth because these two parameters are related directly to our policy and the min-min algorithm use them to compute the expected finish time for given task, another thing should be clear in the simulation is that the DCs better to be in the same region because it's our scope. the simulation performed among 2, 3, 5 and 10 DCs; all those DCs have the same configurations represented in Table 2.

Table 2. *Datacenters configurations.*

Datacenter Parameters	Value
Region	0
Speed (MIPS)	5000
Number of cores	2
Number of VMs	5
Image size	10000
Memory	512
Bandwidth	1000
User Grouping Factor in User Base	10
Request Grouping Factor	10
Executable instruction length/request	500
Simulation Duration	24 Hours
VM Image Size	10000
VM Memory	512 Mb
VM Bandwidth	1000

5.3. Results and Analysis

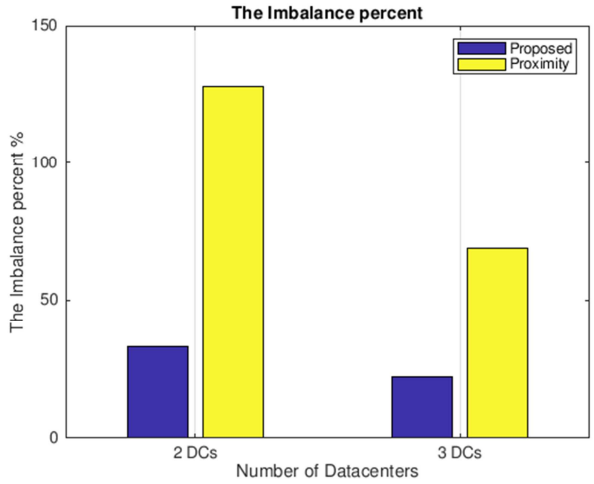

Figure 3. *The imbalance percent.*

After setting the simulation environment parameters, the results and the discussion are detailed in this section. Three performance metrics: imbalance percent, standard deviation and Skewness are introduced to evaluate the proposed policy compared with the proximity based.

Imbalance percent: this metric is used to measure the performance that lost during processing the user request the results shows that using the min-min algorithm reduce the lost performance by lower the value to 33% comparing to 128% using the original proximity based policy in 2 DCs scenario, and also in 3 DCs case the value of imbalance still lower than the proximity based, this improvements are shown in the Figure 3.

Standard deviation and Skewness: the bigger value of the standard deviation reflect more unbalance load over the system, the results show that our proposed solution give more balanced load over the datacenters in both two and three datacenters system. While Skewness is the measure of unevenness resource utilization of a server, the min-min algorithm gives a Skewness value closer to 0 than the proximity based, which mean more normal distribution of load over the system, the results of the standard deviation and Skewenss are presented in Figure 4 and Figure 5 respectively.

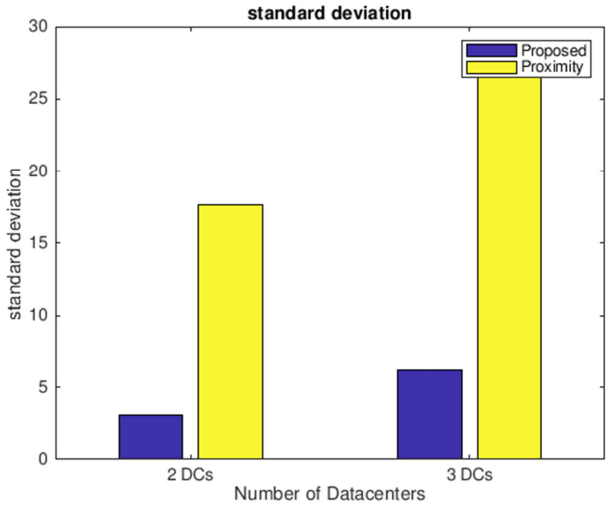

Figure 4. *Standard deviation.*

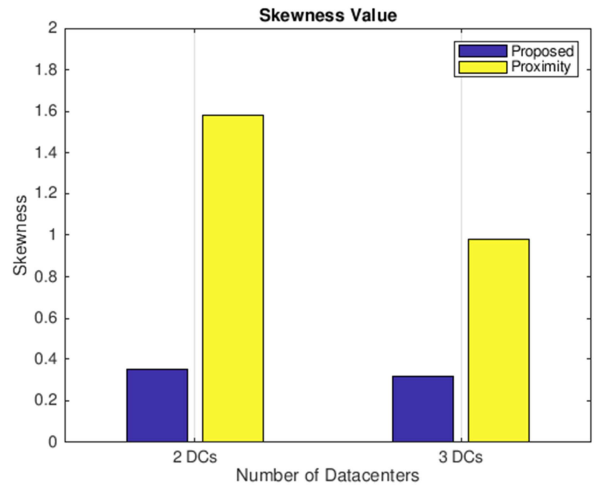

Figure 5. *Skewness.*

5.4. Discussion

The results show a major improvement over three important metrics: 1) The imbalance percent; 2) standard deviation 3) Skewenss.

The result of the measurements (in two datacenters or three datacenters scenarios) demonstrates that our proposed model have the advantage over Proximity-based model in distributing the balance over the datacenters, all the metrics that been used give better values in our model, for example our model minimize the Skewness comparing to Proximity-based model, which indicate less unevenness resource utilization, our Skewness is closer to 0 than Proximity-based, in other

word, closer to normal distribution. Other important metric value is percent imbalance is better in our model since it's noticeably less than Proximity-based (33% to 128%). This results were expected since the random selection of data center in the proximity based policy is raising the chances of overwork one of the available datacenters because this random selection is done without considering the status and characterization of the datacenters in the system.

6. Conclusion

In this study we showed that modifying the broker policy by using the min-min scheduling algorithm which that been proposed in previous work to improve the response time, is also affect the distribution of work load over the datacenters geo- distributed cloud systems comparing to the proximity based. We have used three common metrics to evaluate the improvement in distribution of load over the cloud, the results prove that our proposed model can noticeably give better distribution of the work load since it choose the datacenter to handle the coming requests by considering the status of the datacenter instead if the random selection in proximity based policy, this random selection is the major draw in the proximity based policy that can lead to overwhelming one datacenter and unevenness in distribution the load.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant NO.61472139). The Professor Chunhua Gu is the corresponding author of this paper.

References

[1] C. Devyaniba and T. Naimisha. "Cost effective selection of Data center by Proximity-Based Routing Policy for Service Brokering in Cloud Environment" International Journal of Computer Technology & Applications, Vol 3 (6), 2057-2059.

[2] S. Ranjan Jena and Z. Ahmad, "Response time minimization of different load balancing algorithms in cloud computing environment", IJCA, Volume-69, No-17, May 2013 edition.

[3] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. Pratap Singh and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization", 2012, 14th International conference on modeling and simulation.

[4] J. James and B. Verma, "Efficient VM load balancing algorithm for a cloud computing environment", International Journal on Computer Science and Engineering (IJCSE), 2012.

[5] S. Ray and A. De Sarkar, "Execution analysis of load balancing algorithms in cloud computing environment", IJCCSA, Vol.2, No.5, October 2012.

[6] A. Singh. N and M. Hemalatha, "An approach on semi -distributed load balancing algorithm for cloud computing system", IJCA, Volume 56– No.12, October 2012.

[7] S. Sethi, A. Sahu and S. Kumar Jena, "Efficient load balancing in cloud computing using Fuzzy logic", IOSRJEN, ISSN: 2250-3021 Volume 2, Issue 7, 2012

[8] H. Mehta, P. Kanungo, and M. Chandwani, "Decentralized content aware load balancing algorithm for distributed computing environments", Proceedings of the International Conference Workshop on Emerging Trends in Technology (ICWET), February 2011, pages 370-375.

[9] Y. Lua, Q. Xiea, G. Kliotb, A. Gellerb, J. R. Larusb and A. Greenber, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services", An international Journal on Performance evaluation.

[10] J. Hu, J. Gu, G. Sun, and T. Zhao, A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environ1144 International Journal of Scientific & Engineering Research Volume 5, Issue 3, March-2014 ISSN 2229-5518.

[11] S. Wang, K. Yan, W. Liao, and S. Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, September 2010, pages 108-113.

[12] R. K. Mishra, S. Kumar, B. Sreenu Naik, Priority based Round-Robin service broker algorithm for Cloud-Analyst [C]//Advance Computing Conference (IACC), 2014 IEEE International. IEEE, 2014: 878-881.

[13] L. Sheikhani, Y. Chang, C. Gu and F. Luo, "Modifying broker policy for better response time in datacenters," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 2459-2464.

[14] B. Wickremasinghe, R. Buyya, "CloudAnalyst: A CloudSim-based tool for modelling and analysis of large scale cloud computing environments," MEDC project report, 22 (6), 2009, pp.433-659.