Turkish Online Journal of Qualitative Inquiry (TOJQI)

Volume 12, Issue 3, June 2021: 4553 - 4565

Research Article

BUDGET-BASED SERVICE ALLOCATION AND LOAD BALANCING WITH MEMORY DE-DUPLICATION FOR AUTO SCALING IN CLOUD COMPUTING

Dr.K.Sheela Sobana Rani[1], Dr.M.Yuvaraju[2]

Automatic Scaling provides the opportunity for the applications where their resource utilization can be automatically scaled up and scaled down by the cloud service provider. But achievement of high demand satisfaction ratio is a difficult problem. So, the existing research Class-Constrained Bin Packing (CCBP) problem is formulated in which every server is denoted as bin and every class denoted as an application. An efficient Semi-Online Color Set algorithm (SOCS) is presented for solving the CCBP problem. But the drawback in this method is the demand satisfaction is less when the resources become tight and load balancing among the VMs is not considered that is essential for improving efficiency. In this manuscript an innovative technique called Optimal Selection of Cloud Service with Price for QoS Concession Budget-based Service Allocation and Load Balancing with Memory De-Duplication (OSCSPQC-BSA-LBMD) method is introduced for improving demand satisfaction ratio and achieves load balancing. In this method, the customers are sorted according to their budgets and allocate the resources for the high premium customers. If the load of a particular server is increased, live migration is applied for balancing load among the servers. For that, the statistical information like CPU utilization and memory consumption are gathered and assign the VMs to servers. During the live migration, in order to avoid unnecessary migrating memory pages the Memory De-Duplication method is used. In this method, the duplicate memory pages are identified and stored only once. Experimental results show that the proposed method achieves high demand satisfaction ratio and reduce the complexity.

INTRODUCTION

Cloud Computing is a popular enterprise model to provide on demand services as per the user requirements. In cloud computing, the virtualization technology acts as a role for providing the physical resources like processors, disk storage, and broadband network. Virtualization denotes to the abstraction of physical resources for users. On the other hand, a user demand on resources can be different in different time. Based on these issues, the cloud computing model provides a feature called auto scaling. Auto scaling provides an opportunity for the customers to scale up scale down the resources as per their requirements. The Amazon EC2 service (2012) facilitates customers to get more VM instances as their desire. Various Internet applications can profit auto

[1]Associate Professor, Department of Electrical and Electronics Engineering Sri Ramakrishna Institute of Technology, Coimbatore
[2]Assistant Professor, Department of Electrical and Electronics Engineering Anna University Regional Centre, Coimbatore

scaling property where the utilization of the resources can be scaled up and down automatically by the cloud service provider.

Although the cloud computing model provides infinite capacity based on demand, the capacity of data centers is finite only. If there is a large amount of applications require resources at the same time, then the available resources should be limited. Hence, some of the demands of the users are not satisfied. The demand satisfaction ratio is defined as the percentage of application that fulfilled effectively. Moreover, some of the application needs less amount of resources so it is essential to save energy by diminishing number of servers used. Similarly, in many Internet data centers the average server utilization is very low: real world estimation range is from 5% to 20% Armbrust et al (2009), Siegele (2008). Chase et al (2001) has developed a method that provides an efficient way to save energy by turn off the whole server.

The existing research develops a method that provides auto scaling for the applications in the cloud computing. The instance of the application is implanted inside a VM and virtualization is utilized for providing fault segregation. The CCBP (Xiao et al 2014) problem is originated where a server is indicated as a bin and an application is indicated as a class. But the restrictions of the applications cause impacts in the number of applications a server can run concomitantly. A proficient SOCS algorithm is presented for achieving high demand satisfaction ratio and save energy by reducing the number of servers if the requirement is low. The disadvantage in this method the demand satisfaction is not accomplished when the resource becomes tight. Additionally, the load balancing is not achieved by this method. In the proposed research, a new method is introduced called OSCSPQC-BSA-LBMD for achieving high demand satisfaction ratio and load balancing. The following contributions are:

(1) Firstly, the customers are sorted based on their budget given for the resources. The preference is given to the customers who paid high budget for the resources.
(2) Secondly, the resources are allocated by using the effectual semi-online color set al gorithm which facilitates an automatic scaling feature where their resource usage can be scaled up and down automatically by the cloud service provider.
(3) During the resource allocation, if the server load is high live migration is applied for balancing the load among the servers.
(4) The Memory De-Duplication method is used in the migration process. This method facilitates the identification of duplicate memory pages and stored only once. This eliminates the migration of unnecessary memory pages.

LITERATURE RVIEW

This section, describes the previous research related to bin packing problems for resource allocation.

The conventional bin packing problem has been comprehensively studied in the literature Galambos et al (1995). The vector bin packing problem considers multi-dimensional restraints when items are packed into a less number of bins Chekuri & Khanna (2004). In the vector bin packing problem, the CPU demand and memory obligation are considered as an

individual elements and utilize vector bin packing to handle this problem. But the problem is the memory obligation of Internet applications has to be satisfied as a whole: a main segment of the memory is consumed anyway even when the application receives little load.

Shachnai & Tamir et al (1998) presented the class-constrained multiple knapsack problems that intent to increase the total number of packed items under the constraint that each knapsack has inadequate capacity and a bound on the number of different types of items it can hold (Shachnai & Tamir 2001). But this method does not consider to reducing the number of knapsacks used.

Resource provisioning for web server has been examined. Some allocate resources in the granularity of whole servers which can lead to ineffective resource utilization. Some of the methods do not handle the practical limit on the number of applications a server can run concurrently Wolf & Yu PS (2001).

Bhuvan et al (2002) developed a method that supports shared hosting, but an instance of every application manages autonomously Urgaonkar et al (2002). This method does not provide auto scaling property. If the application demand varies, this method does not attempt to diminish the placement alterations. Zhang et al (2009) focused a set of shared clusters into a network and resource allocation among the clusters. The process of migration has been studied in different contexts Osman (2002). When compared to the virtualization technology, this method does not consider the execution environment of the running processes. This also not supports auto scaling process according to the demand.

CCBP FOR AUTOMATIC SCALING OF RESOURCES

In this method, the automatic scaling problem is defined and it formulate as a Class-Constrained Bin Packing (CCBP) problem in which every server is denoted as a bin and every application is denoted as a class. A new auto scaling algorithm is developed for handling the CCBP problem. This method can support for increasing or decreasing the resource usage based on the customer requirements. The frequent placement variation of the application is to be reduced by using this method. This method also reduces the energy consumption by turn off the server whenever the resource usage level.

Let us consider, Server Set S and set of applications is denoted as A. The server CPU capacity is denoted as C_s and the number of applications executed in a server concurrently based P is denoted as the application placement matrix $P_{a,s} = 1$ denoted as that the application a has an instance running on server, otherwise $P_{a,s} = 0$ and L denotes the application load distribution matrix. $L_{a,s}$represents the CPU resource allotted on server for a particular application. E represents the energy consumption for an active server during the running process. The current application placement matrix is represented by P^* and the predicted CPU demand for a particular application is denoted by C_a and the CPU and memory capacity for the server is

denoted by C_s andM_s. Based on the application requirements a new application placement matrix is created that is represented by P and load distribution matrix is denoted as L.

Increasing demand satisfaction ratio, the variation in application placement reduces and diminish the energy consumption. The objective functions are as follows:

(1) **Maximize** $\sum_{\mathbf{a}\in\mathbf{A}}\sum_{\mathbf{s}\in\mathbf{S}}\mathbf{L_{a,s}}$
(2) **Maximize** $\sum_{\mathbf{a}\in\mathbf{A}}\sum_{\mathbf{s}\in\mathbf{S}}|\mathbf{P_{a,s}}-\mathbf{P^{*}_{a,s}}|$,
(3) **Minimize** $\mathbf{E}*|\{\mathbf{s}|\sum_{\mathbf{a}\in\mathbf{A}}\mathbf{P_{a,s}}>\mathbf{0}\}|$,and the restraints are,

$$L_{a,s}\geq 0\ \forall a\in A,\forall s\in S$$

$$P_{a,s}=0\Rightarrow L_{a,s}=0\ \forall a\in A,\forall s\in S,$$

$$\sum_{a\in A}I_{a,s}\leq M_s,\forall s\in S,$$

$$\sum_{a\in A}L_{a,s}\leq C_s,\forall s\in S,$$

$\sum_{s\in S}L_{a,s}\leq c_a,\forall a\in A,$ (1)

This auto scaling problem is same as the Class-Constrained Bin Packing (CCBP) problem in which each application is labeled as a class and CPU demands for the entire classes is denoted as items which need to be packed into S bins. To solve the CCBP problem, an efficient Semi-Online Color Set al gorithm (SOCS) is presented.

In this algorithm, a series of items require to be packed into a small number of bins. The class restrained version of this problem separates the items into classes or colors. Each bin has the capacity which is denoted by v and handles items from at most c different classes. Items from a particular class denotes the resource demands for the regarding application. The size of an item denotes an amount of load for the regarding application. Consider the entire items have the similar unit size; the item size is represented as a unit of load. This is defined as the "load unit". The bin capacity is denoted by amount of units of load a server can handle. The requirement of resources is varied with time to time. This can be modeled as item arrivals and departures: if the new item arrives, the load is increased, whereas load decreases correspond to departure of previously packed items.

Item arrival: If a new item arrives from a particular color set, it is packed into the analogous unfilled bin. If the entire bins are occupied, a new bin is opened for handling the new item.

Item departure: The departure algorithm is as follows: If the color set does not have an unfilled bin, the item of particular color is detached and the resulting bin becomes the unfilled bin. Suppose, if there is departing color in the unfilled bin, an equivalent item is eliminated directly and shut down the corresponding server to reduce the energy consumption.

OPTIMAL SELECTION OF CLOUD SERVICE WITH PRICE FOR QOS CONCESSION -BUDGET-BASED SERVICE ALLOCATION AND LOAD BALANCING WITH MEMORY DE-DUPLICATION (OSCSPQC-BSA-LBMD) METHOD

A new method introduced is Optimal Selection of Cloud Service with Price for QoS Concession, Budget-based Service allocation and load balancing with Memory De-Duplication (BSA-LBMD) method for achieving high demand satisfaction ratio when the resource becomes tight and also improving load balancing among the servers. In this method, firstly the budget given by the customers are analyzed. Then, the service provider selects the premium customers based on their budgets. If the resource has high demand, the resources are allocated to the premium customers first. An effectual semi-online color set al gorithm is used to allocate the resources. This method facilitates an automatic scaling feature where their resource utilization can be increased or decreased automatically by the cloud service provider. If the sever has high load during the resource allocation process, the live migration is applied for balancing the load among the servers. In the VM migration process, the Memory De-Duplication method is used for finding the duplicate memory pages and stored only once. By using the Memory De-Duplication method, the migration of unnecessary memory pages is eliminated.

Optimal Selection of Cloud Service with Price for QoS Concession Budget-based Service allocation and Load balancing with Memory De-Duplication (OSCSPQC-BSA-LBMD) algorithm

Input: Server Set $S = \{s_1, s_2, \ldots s_k\}$, Set of applications=A, Set of customers$CU = \{cu_1, cu_2, \ldots cu_n\}$, Budget given by the customers $B = \{b_1, b_2, \ldots b_m\}$ for the resources

Output: Resource allocation with load balancing among the servers

1. Deploy the N number of servers and N number of VMs
2. CUs give their budget B for the required resources
3. If resource becomes tight then
4. Select premium CUs first based on their B //CUs=Customers, B=Budget
5. Selection of Premium Customers
6. Analyze the B given by the CUs
7. Sort the users in descending order based on their budgets
8. $\mathbf{New\ List = Sort(B\_CUs)}$ (2)
   B_CUs=Budget of the customers
9. Allocate the resources for the premium CUs
10. Else
11. Allocate resources to all the users
12. End if

13. Resource Allocation using effectual semi-online color set method
14. Set of servers $\mathbf{S} = \{\mathbf{s_1}, .. \mathbf{s_k}\}$ and set of applications, CPU capacity of server $\mathbf{C_s}$ and memory factor of the server is $\mathbf{M_s}$
15. Objective functions are as follows:
16. **Maximize** $\sum_{\mathbf{a}\in\mathbf{A}} \sum_{\mathbf{s}\in\mathbf{S}} \mathbf{L_{a,s}}$ (3)

    $\mathbf{L_{a,s}}$=CPU resource allotted on server for a particular application
17. **Minimize** $\sum_{\mathbf{a}\in\mathbf{A}} \sum_{\mathbf{s}\in\mathbf{S}} |\mathbf{P_{a,s}} - \mathbf{P^*_{a,s}}|$ (4)

    $\mathbf{P_{a,s}}$= New application placement matrix $\mathbf{P^*_{a,s}}$=Current application placement matrix
18. **Minimize** $\mathbf{E} * |\{\mathbf{s}| \sum_{\mathbf{a}\in\mathbf{A}} \mathbf{P_{a,s}} > \mathbf{0}\}|$, (5)

    E= energy consumption for an active server
19. This problem modeled as CCBP which class represents an application and bin represents the server
20. Each bin has capacity v and accommodate the set of items //item= Resource demands of particular application
21. Label class of items with a color and organize them into color sets
22. Every bin runs the application based on the v
23. Application load increase
24. If new item arrives from particular color set then
25. Shift some of the items to unfilled bin and accommodate new item in currently full bin
26. Else if (Entire bins are occupied)
27. Open new bin
28. End if

Application load decrease

29. If the color set does not have an unfilled bin then
30. Remove the item of particular color
31. Shut down the corresponding server
32. End if

    Load balancing
33. Take the N number of servers
34. Compute the LF for all the servers //LF=Load factor
35. For i=1 to k

    $\mathbf{LF_i} = \mathbf{CPU_U} + \mathbf{D_U} + \mathbf{M} + \mathbf{NB}$ (6)

    $\mathbf{CPU_U}$=CPU usage, $\mathbf{D_U}$=Disk usage, M=Memory, NB=Network bandwidth
36. **IF**$(\mathbf{LF} > Th)$ then //Th=Threshold

    VM is migrated to lightly loaded server
37. Call Memory De-duplication algorithm
38. End if
39. End for
40. Memory De-duplication algorithm
41. Snapshot the memory pages of the VMs
42. Compute the **h_value** using MD5 hash algorithm for all the memory pages and create hash table // **h_value**=hash value
43. All the servers store the hash value and all saved page information
44. If VM is migrated from one server to another server then

45. Target server calculate the hash value and then search the hash table
46. If Already found then
47. Not saved the memory pages
48. Else
49. Save the information and update the table
50. End if
51.

Optimal Selection of Cloud Service with Price for QoS Concession Budget-based Service allocation and load balancing with Memory De-Duplication Algorithm (OSCSPQC-BSA-LBMD). The steps are as follows

Customer Budget analysis and selection of premium customers

In this algorithm, the input taken is Server Set $S = \{s_1, s_2, \ldots s_k\}$, Set of applications=A, Set of customers $CU = \{cu_1, cu_2, .. cu_n\}$, budget provided by the customers $B = \{b_1, b_2, .. b_m\}$ for the resources. The Server Set is initialized and the set of VMs are initialized. The customers give their budgets according to the required resources. If the resource becomes tight, the high premium resources are selected first and allocate the resources for achieving high demand satisfaction ratio. The premium customers are selected according to their budgets. The customers are sorted in descending order based on the given budgets. The sorted customers are saved in the $New\ List$. Allocate the resources to the premium customers. If the resource is not tight, randomly allocate the resources.

Auto scaling based Resource allocation

After computing the premium customers, the resources are allocated. In this method, an effectual semi-online color set al gorithm is used for increasing and decreasing the resource usage of the customers. The CPU capacity and memory capacity of the server is denoted as C_s andM_s. The objective functions are to maximize the resource allocation, minimize the frequency of placement changes and minimize the energy consumption. To achieve these objective functions, a Class-Constrained Bin Packing is formulated in which class represents an application and bin represents the server. Each bin has the capacity v and accommodates the set of items. The Item denoted is the resource demand for the equivalent application. The class of items is labeled with color and organizes them into color sets. Every bin has executes the application based on the application. If the application load increases, some of the items are shifted to unfilled bin and accommodate new item in currently full bin. If there is no unfilled bin, the new bin is opened. If the application load is decreases, then remove the item of particular color and then shut down the equivalent server for reduce energy consumption.

Load balancing among the servers by VM migration

During the resource allocation, if the server load is high live migration is applied for balancing the load among the servers. In this method, the set of servers are taken and the load factor (LF) is computed. The LF is computed by,

$$LF\_i = CPU_U + D_U + M + NB \quad (7)$$

In this equation, CPU_U denotes the CPU usage, D_U represents the disk usage. M denotes the memory and NB represents the network bandwidth. If the LF is greater than threshold, the particular server is highly loaded. So, it is necessary to migrate the VM to lightly loaded server. During the migration process, to avoid the unnecessary memory pages, the Memory De-Duplication method is used. This method facilitates the identification of duplicate memory pages and stored only once. This eliminates the migration of unnecessary memory pages.

Memory De-duplication process for VM migration

In the Memory De-Duplication method, firstly snapshot the memory pages of the VM. The hash value is evaluated for all the memory pages and the hash table is created. All the servers store this information. The hash value is computed by using MD5 hash algorithm. The MD5 hash algorithm completes a sequence of 64 process on each 512-bit (64-byte) chunk of the input. The resultant hash of one chunk is then feed as an input to the algorithm as it processes the next chunk. Each and every 4K memory page can be analyzed as 64 chunks of 64 bytes each. For a message with multiples of l-bit length, the iterating process is followed as: objective in this paper is to optimize the weight coefficients of multi-point decision making algorithm and ensure maximum user satisfaction ratio to select best RAT. For this a hybrid methodology integrating a Non-Homogenous Biogeography Based Optimization (NHBBO) is proposed. Thus the parallel fuzzy systems are employed to determine the probability of RAT selection, which acts as an input to the non-homogenous biogeography based optimization procedure. Several experiments are carried out using the proposed NHBBO – PFS technique to demonstrate the effectiveness and robustness in producing solutions.

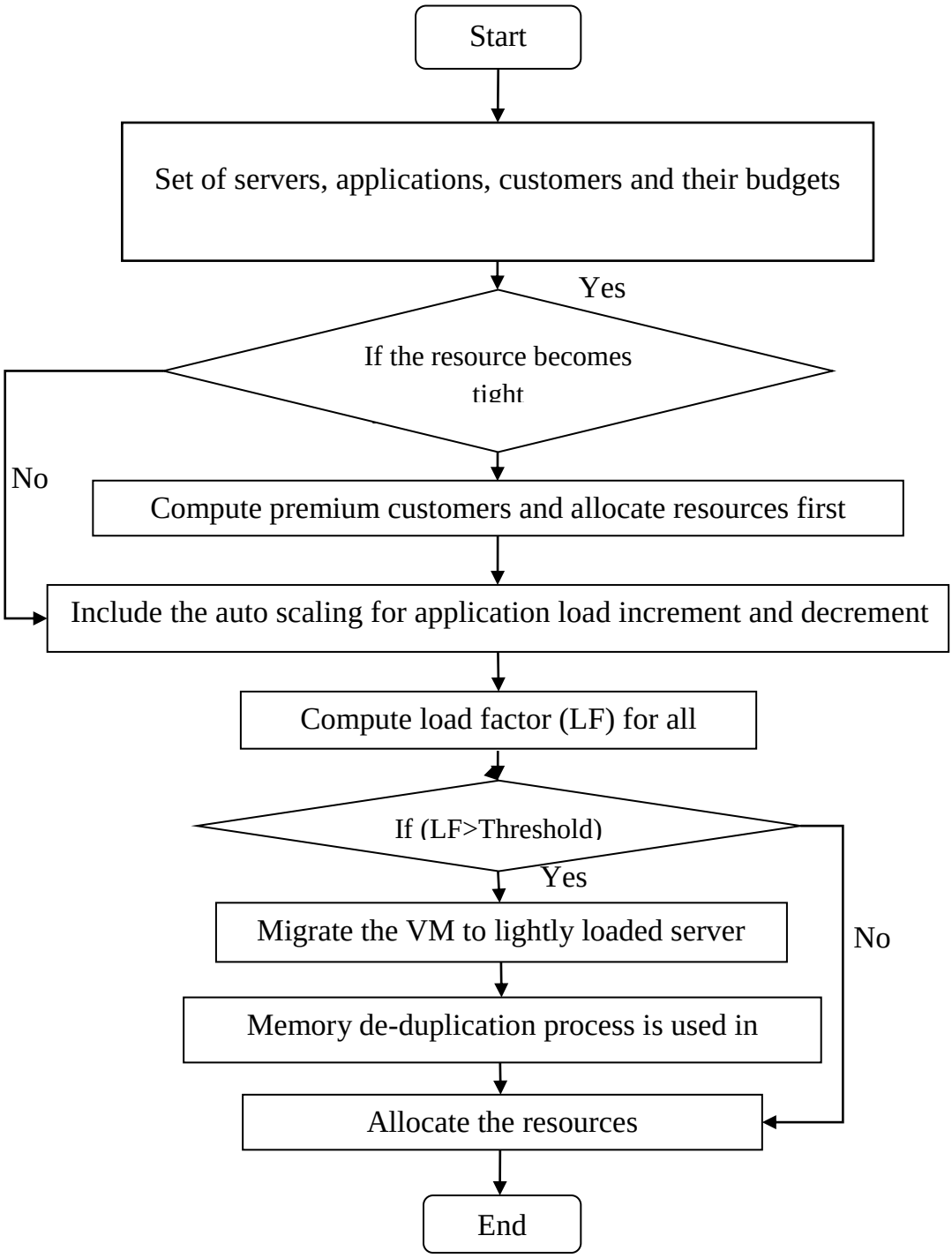

Figure 1 Flow chart for Budget-based Service allocation and Load balancing with Memory De-Duplication (BSA-LBMD)

$$H_{i+1} = f(H_i, M_i), 0 \leq i \leq t-1. \quad (8)$$

Here, $M = M_0, .. M_{t-1}$ and $H_0 = IV_0$ is the initial value for the hash function.

A hash based method is used in which the hash table is shared by all the servers and accumulates all saved page information. The information contains the hash value and also the equivalent file name of the memory page. If VM is migrated from one server to another server, the target server evaluates the hash value and then searches the hash table. Before a page being saved by the target server, it evaluates the hash value and then searches the hash table using the calculated hash value. If these values are already fond, the memory pages are not saved. Otherwise, save the memory pages and update the hash table.

NUMERICAL RESULTS

In performance analysis, the existing SOCS and the proposed BSA-LBMD method are compared in terms of response time and satisfaction probability. The SOCS method is used to increase or decrease the load of the application according to the requirements. In the BSA-LBMD method, the customers are classified as premium customers based on their budget and allocate the resources. If the load is not balanced among the servers, the VM migration is used to balance the load.

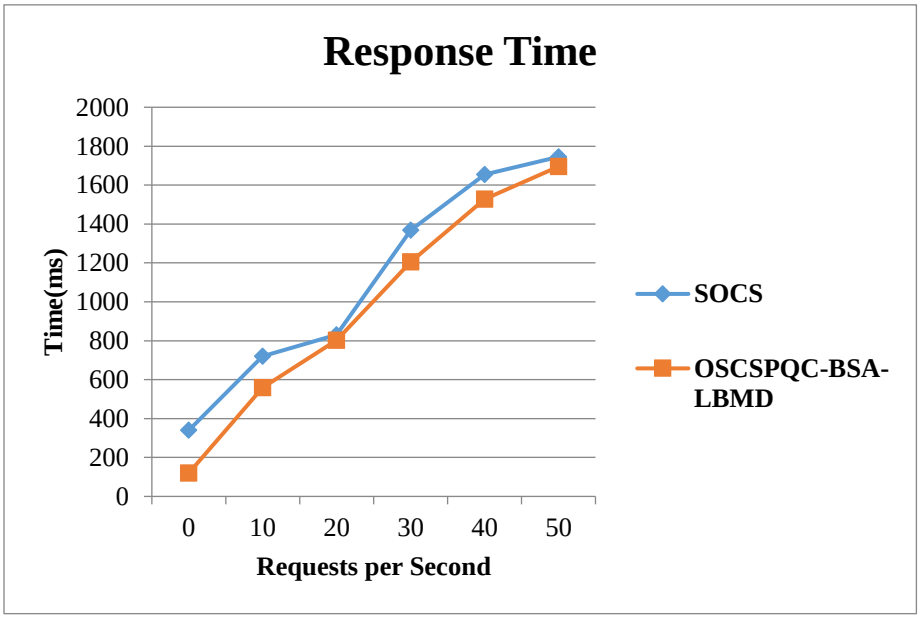

Figure 2 Response Time

Figure 2 shows the response time comparison for the SOCS and the OSCSPQC-BSA-LBMD methods. Response time is defined as the time taken for response with respect to

the customer request. In the existing system, SOCS method is used and in the proposed system OSCSPQC-BSA-LBMD method is presented. In the X-axis time in minutes is taken. In the Y-axis response time in ms is taken. When compared to the existing SOCS method, there is less response time in the proposed OSCSPQC-BSA-LBMD method.

Table 1 shows the comparison of ex- dynamic resource allocation that is important area in the cloud computing environment. In this work, DRM is presented which can use virtualization technology to allocate data center resources dynamically based on the application demands of the users and support green computing by optimizing the number of servers that are currently being used. This work considers the overload avoidance and green computing. By this process it accomplishes the better tradeoff between the two metrics. The skewness concept is used to measure the unevenness in the multi-dimensional resource utilization of a server. By reducing the effect of skewness, the different types of workloads can be combined well. This can improve the overall utilization of server resources. A set of heuristics are introduced to prevent overload in the system significantly. The distributed VM placement algorithms by the process of considering distributed cloud data centers with the objective can minimize the overload and also can minimize the number of PMs. Experimental results shows that the proposed method achieves high efficiency in terms of CPU load and decision time.

Table Error! No text of specified style in document. **Response time**

Request per second	SOCS(ms)	OSCSPQC-BSA-LBMD(ms)
0	340	120
10	720	560
20	830	802
30	1369	1206
40	1654	1528
50	1745	1695
60	1910	1885

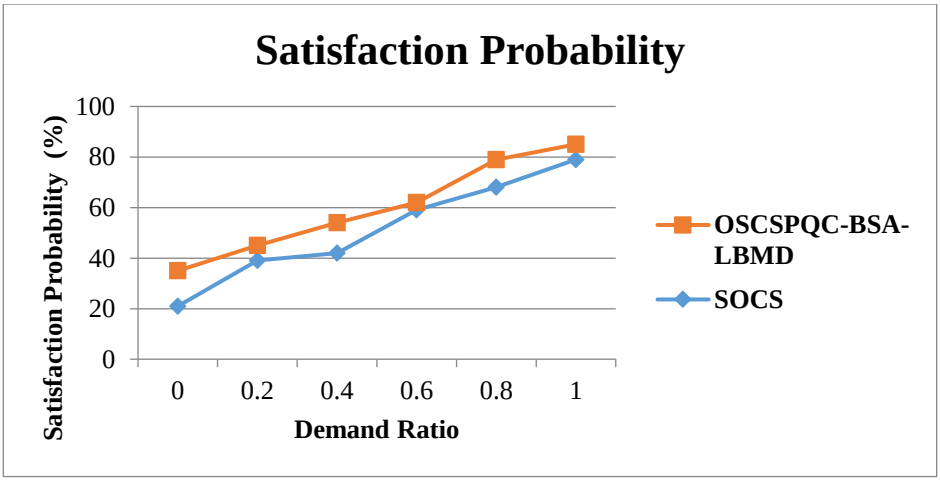

Figure 3 Satisfaction Probability

Figure 3 shows the satisfaction probability comparison for the SOCS and the OSCSPQC-BSA-LBMD methods. Satisfaction probability is defined as the probability that demands satisfied successfully. In the existing system, SOCS method is used and in the proposed OSCSPQC-BSA-LBMD method is presented. In the X-axis demand ratio is taken. In the Y-axis satisfaction probability in percentage is taken. When compared to the existing SOCS method, there is high satisfaction probability in the proposed OSCSPQC-BSA-LBMD method.

Table 2 Satisfaction Probability

Demand ratio	SOCS(%)	OSCSPQC-BSA-LBMD(%)
0	21	35
0.2	39	45
0.4	42	54
0.6	59	62
0.8	68	79
1	79	85

Table 2 shows the comparison for the existing SOCS and the OSCSPQC-BSA-LBMD in terms of satisfaction probability. If the demand ration is 1, the satisfaction probability is 79% in the SOCS method and 85% in the BSA-LBMD method.

CONCLUSION

Auto scaling is a significant cloud computing technique that dynamically allocates resources to the customers based on their requirements. In the presented work, Optimal Selection of Cloud Service with Price for QoS Concession-Budget-based Service Allocation and Load Balancing with Memory De-Duplication (OSCSPQC-BSA-LBMD) is introduced that allocated the resources with high demand satisfaction ratio. If the resources are very tight, the premium customers are selected and allocate the resources. The premium customers are selected based on the budget given for the resources by the customers. Then, the resources are allocated to the customers with auto scaling feature. The resource usage can be increased or decreased dynamically. If the load of a particular server is increased, the VM is migrated from one server to another. During the migration process, the Memory De-Duplication process is applied to identify the duplicate memory pages are identified and stored only once. An experimental result shows that the proposed OSCSPQC-BSA-LBMD method achieves high demand satisfaction ratio and less response time.

References

1. Galambos, G &Woeginger, GJ 1995, 'On-line bin packing—a restricted survey', Zeitschriftfür Operations Research, vol. 42, no. 1, pp. 25-45.
2. Shachnai, H &Tamir, T 1998, 'Noah's bagels-some combinatorial aspects', in In Proc. 1st Int. Conf. on Fun with Algorithms.

3. Zhang, C, Lesser, VR &Shenoy, PJ 2009, 'A Multi-Agent Learning Approach to Online Distributed Resource Allocation', in IJCAI, pp. 361-366.

4. Wolf, JL & Yu, PS 2001, 'On balancing the load in a clustered web farm', ACM Transactions on Internet Technology (TOIT), vol. 1, no. 2, pp. 231-261.
5. Chekuri, C &Khanna, S 2004, 'On multidimensional packing problems', SIAM journal on computing, vol. 33, no. 4, pp. 837-851.
6. Xiao, Z, Chen, Q &Luo, H 2014, 'Automatic scaling of internet applications for cloud computing services', Computers, IEEE Transactions on, vol. 63, no. 5, pp. 1111-1123.
7. Chase, JS, Anderson, DC, Thakar, PN, Vahdat, AM & Doyle, RP 2001, 'Managing energy and server resources in hosting centers', in ACM SIGOPS Operating Systems Review, vol. 35, pp. 103-116.
8. Siegele L 2008. Let it rise: a special report on corporate IT, The Economist newspaper, London, UK; 389, 3–16.
9. Tang, C, Steinder, M, Spreitzer, M &Pacifici, G 2007, 'A scalable application placement controller for enterprise data centers', in Proceedings of the 16th international conference on World Wide Web, pp. 331-340.
10. Kossmann, D, Kraska, T &Loesing, S 2010, 'An evaluation of alternative architectures for transaction processing in the cloud', in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 579-590.
11. Nguyen, H, Shen, Z, Gu, X, Subbiah, S & Wilkes, J 2013, 'Agile: Elastic distributed resource scaling for infrastructure-as-a-service', in Proc. of the USENIX International Conference on Automated Computing (ICAC'13). San Jose, CA.

12. Nguyen Van, H, Dang Tran, F &Menaud, J-M 2009, 'Autonomic virtual resource management for service hosting platforms', in Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 1-8.

13. Kriushanth M, L. Arockiam& G. JustyMirobi 2013, 'Auto Scaling in Cloud Computing: An Overview', International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no 7.

14. Amazon Elastic Compute Cloud (Amazon EC2) 2012. http://aws.amazon. com/ec2/.
15. Shajan Joseph, A Rajaram, "A Novel secure and efficient clustering based routing protocol for MANET with certificate Revocation," International Journal of Computer Technology and Applications (IJCTA), 10(20):41-52, 2017.