Load Balancing in Cloud Computing

CS218 Topics in Cloud Computing

Term Project 1

Piyush Bajaj

Saketh Saxena

Dr. Melody Moh

CONTENTS

List of Acronyms

1. WCAP : Workload and Client Aware Policy
2. OLB : Opportunistic Load Balancing
3. LBMM : Load Balancing Min-Min
4. LB-BC : Load Balancing using Bayes Clustering
5. LB-VC-FC : Load Balancing Algorithm for Virtual Cluster Using Fuzzy Clustering

Abstract

Cloud computing is a computing model which ensures delivery of computing services such as servers, storage, databases, networking and software over the internet and is commonly referred to as the cloud. Cloud computing brings advantages such as cost, flexibility and availability of service to the users. These advantages make the demand for cloud services to rise which brings along a lot of technical issues such as availability of resources and scalability. Thus, load balancing becomes a major concern and requires to dynamically allocate workload evenly across all nodes. Cloud Load Balancing is a technique to divide the load among different available resources and equalize it across virtual machines to achieve efficiency and increases throughput. There have been many load balancing algorithms and policies, some of which we will be investigating such as the Load balancing Min Min (LBMM), workload and client aware policy (WCAP), join-idle-queue, honey-bee foraging behavior, etc., as part of our project. Many new algorithms have also been proposed with the aim to increase the efficiency of load balancing some of which we will be exploring such as Clustering based Load Balancing using Bayes clustering (LB-BC) and Load Balancing Algorithm for virtual Cluster using Fuzzy Clustering (LB-VC-FC) as part of this research project.

Introduction

Cloud computing is a computing model which enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage devices, applications and services which can be rapidly provisioned and released with minimal management effort and service provider interaction [2]. Cloud computing brings advantages such as virtualization of resources, cost, scalability, flexibility and high availability of service to users [4]. These advantages have driven numerous organizations to port their applications and services, with a huge user base, to the cloud environment which inadvertently has resulted in a high demand for cloud services. Since the resources in an open cloud computing environment are typically shared, spikes in demand lead to technical issues relating to maintaining availability of resources, providing scalability and ultimately resulting in deadlocks.

One way of avoiding this situation is via load balancing, Cloud Load Balancing is a technique to divide the load among different available resources and equalize it across virtual machines to achieve efficiency and increases throughput [4]. It helps in avoiding deadlocks by optimal utilization of resources and increases the system performance. The main challenge in load balancing is to distribute dynamic workload across multiple nodes to ensure no single node is overwhelmed [7]. The goal of load balancing is to reduce resource consumption which will further lead to reducing energy consumption and ensure scalability, avoid bottlenecks, and over-provisioning. Thus, having efficient load balancing techniques in place is extremely important, to meet the growing demand of cloud computing resources.

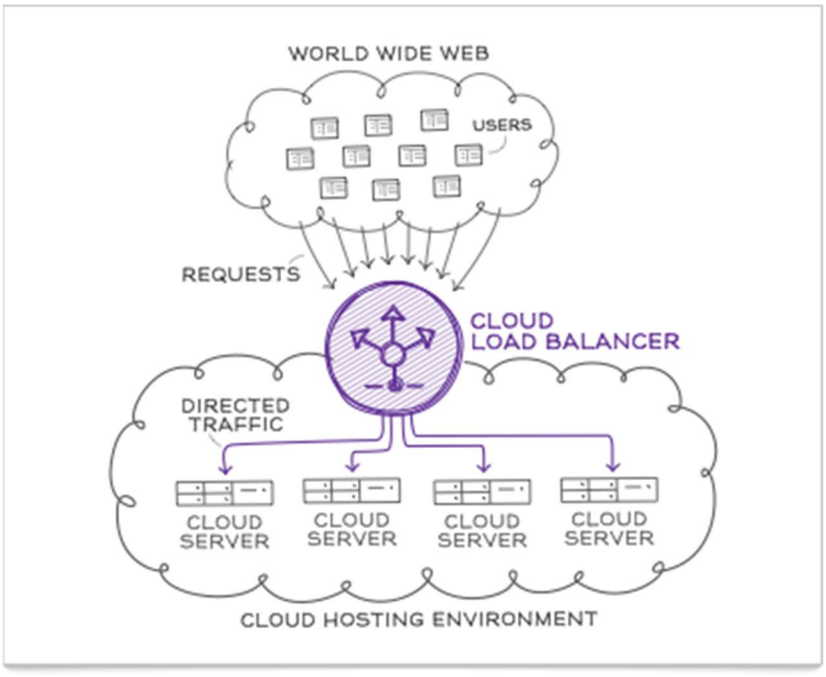

Figure 1: Cloud Load Balancing Schematic Representation [3]

A lot of work has already been done for load balancing in the cloud and, typical load balancing techniques include task scheduling algorithms, load balancing policies, and client-side load balancing [7]. As part of the project report we will be exploring some of the existing load balancing techniques such as workload and client aware policy (WCAP)[4], opportunistic load balancing[1], Load Balancing Min-Min[7, 13], Join-Idle-Queue[8] and Honeybee Foraging Behavior[7]. We will also be exploring some proposed approaches to load balancing in the cloud which employ the use of clustering techniques and heuristics to optimize load balancing [2][4]. All these techniques are elaborated in further chapters, chapter 2 gives an overview of existing load balancing algorithms, chapter 3 describes in detail on a heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment, chapter 4 describes a Load balancing algorithm for virtual cluster using fuzzy clustering and finally chapter 5 summarily concludes the report.

Overview of Existing Load Balancing Algorithms

Workload and client aware policy (WCAP)

WCAP is a decentralized content aware load balancing policy [4]. It employs the usage of a USP - unique and special property to specify the nature of both the requests and computing nodes. The USP enables the scheduler to decide the best suitable host for processing the incoming requests. This strategy has low overhead since it is implemented in a decentralized manner. The basic idea of the technique is to improve the searching performance by using the content information from the USP to search for a suitable host for a specific request. It also helps in reducing the idle time of the computing nodes because it is aware of the current workloads of all hosts and their idle times and thus, improves resource utilization [7].

Opportunistic Load Balancing

Opportunistic load balancing algorithm assigns workload to nodes in free order without considering the expected execution time of each node [1]. OLB is a static load balancing algorithm that has the goal of keeping each node in the cloud busy [1]. The advantage of OLB is that it is very simple to implement. Its shortcoming is that since OLB does not consider the current workload of a node or the expected execution time of a task while assigning tasks to nodes, the overall completion time is very poor. There are variations of this technique such as round-robin load balancing [2].

Load Balancing Min-Min

The load balancing min-min algorithm is a simple task scheduling algorithm which considers the execution time of the tasks and the completion time to assign workload to resources or nodes. It first identifies the task with the minimum execution time and schedules it first, it then checks for the minimum completion time [7]. Since Min-Min chooses the smallest tasks first it loads the fast executing resource more which leaves the other resources idle [7]. In the second round, LBMM reassigns resources from

heavy load and reassigns the workload to resources with light load by considering the completion time of the resources produced in the first round. The process stops when all resources and tasks assigned to them have been considered for rescheduling otherwise the iteration continues. The LBMM algorithm is shown in figure 2.

```
for all tasks Ti
 for all resources
    Cij=Eij+rj
 do until all tasks  are mapped
    for each task find the earliest completion time and the
resource that obtains it
    find the task Tk with the minimum earliest completion time
    assign task Tk to the resource Rl that gives the earliest
completion time
   delete task Tk from list
   update ready time of resource Rl
   update Cil for all i
end do
// rescheduling to balance the load
sort the resources in the order of completion time
for all resources R
Compute makespan = max(CT(R))
End for
for all resources
  for all tasks
    find the task Ti that has minimum ET in Rj
    find the MCT of task Ti
    if MCT <  makespan
       Reschedule the task Ti to the resource that produces it
       Update the ready time of both resources
    End if
  End for
End for
//Where MCT represents Maximum Completion Time
```

Figure 2: LBMM Algorithm [7]

Join-Idle-Queue

This algorithm provides large scale load balancing with distributed dispatchers by, first load balancing idle processors across dispatchers for the availability of idle processors at each dispatcher and then, assigning jobs to processors to reduce average queue length at each processor. By removing the load balancing work from the critical path of request processing, it effectively reduces the system load, incurs no communication

overhead at job arrivals and does not increase actual response time [7]. It is particularly useful for dynamically scalable web services.

Honeybee Foraging Behavior

This is a decentralized honeybee-based load balancing technique for self-organization inspired by the nature. It uses local server actions for efficient load balancing. The system size is considerably increased due to system diversity, in this approach but it is a trivial trade off since the performance of the system is enhanced. It is best suited for the conditions where the diverse population of service types is required [7]. A modified honey bee foraging algorithm has also been proposed which uses the system throughput to intensify the honeybee foraging behavior [2].

A heuristic clustering-based task deployment approach for load balancing using Bayes theorem(LB-BC)

Introduction:

The paper has proposed a heuristic approach to finding the optimal physical hosts for tasks deployment by achieving a load balancing strategy through a long-term algorithm process and thus to obtain optimal performance.

It can be achieved only by deploying requested tasks into the resource pool of the cloud data center efficiently and properly, which can improve computing efficiency of cloud data centers, and provide users with cloud data centers good performance of external service.

Work done related to load balancing:

Currently, most of open-source IaaS platforms have utilized static algorithms to conduct the resource scheduling. For example, Eucalyptus platform uses round robin to assign virtual machines to different physical hosts in sequence to achieve load balancing. Wei et al. have employed the weighted minimum link algorithm, which means that different

weights indicate the performance of the physical host. Then, the virtual machine will be allocated to the physical host which has the smallest ratio of the current number and the weight. The advantage of static scheduling algorithms is that it is simple to use. But in the large-scale cloud data centers whose resource heterogeneity is very strong and user demand isn't consistent, the load balancing effect is not ideal [2].

Dynamic load balancing is the concept which is used to distribute the load among the various hosts more rationally which revolves around the phenomenon that it should not let some computing nodes will be overloaded and some other computing nodes to be light on the load. It should provide a perfect balance in the load across all the nodes across the hosts to achieve the overall performance of the system. During the process of Dynamic load balancing, we will be observing the communication overhead that is produced will downgrade the performance of the system significantly. On top of that, if we consider the increased network latency between the nodes, it will have a consequential impact on the performance of the load balancer.

To address this issue of reducing additional communication overhead in the process of Dynamic load balancing, the probable solution is using greedy algorithms to deal with the scenario. Load Receiver Strategy (LRS) algorithm has used the way that the light load is received and allocated preferentially. Xu et al. have proposed an efficient double-sided combinatorial auction model to dynamically achieve optimal goals. Lau et al. have integrated the two strategies of heavy load priority and light load priority. They have proposed an adaptive load distribution algorithm to effectively reduce communication overhead of the load balancing process. Utilizing the greedy algorithms can solve the problem of load distribution. However, several algorithms above cannot meet greedy choice performance and the nature of optimal substructure at the same time. So, these load distribution approaches often obtain the local optimal solutions. And the effect of solving the problem of load distribution under certain special circumstances is not ideal. Cloud data center cannot reach load balancing of the entire network.

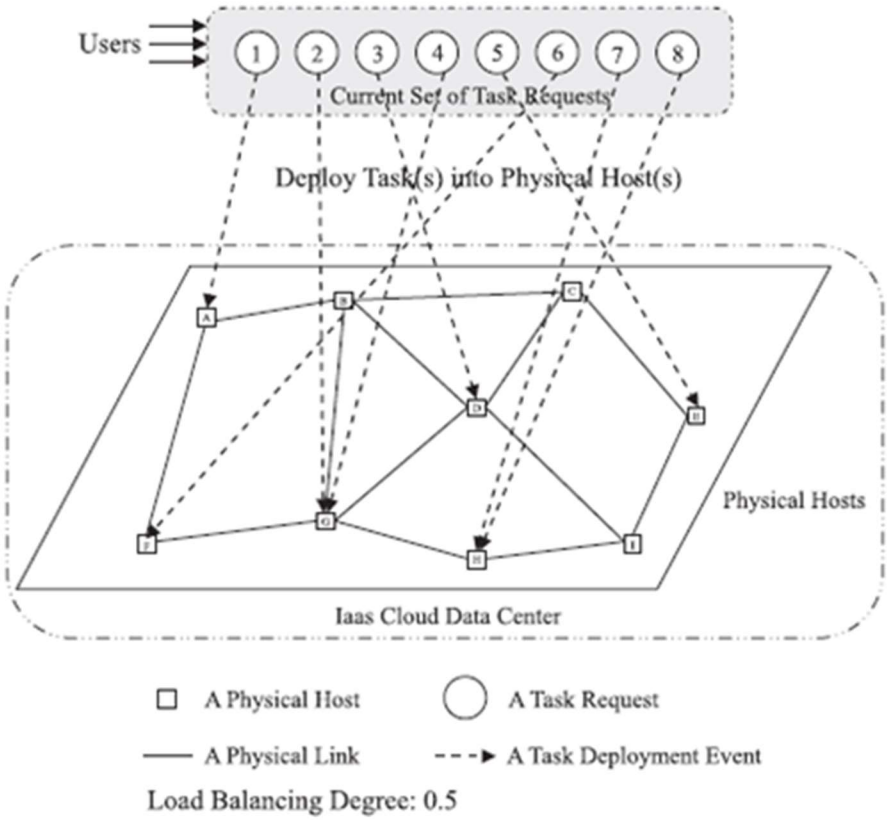

Figure 3. Deploy tasks into physical hosts in the IaaS cloud data center [14]

LB-BC's architecture:

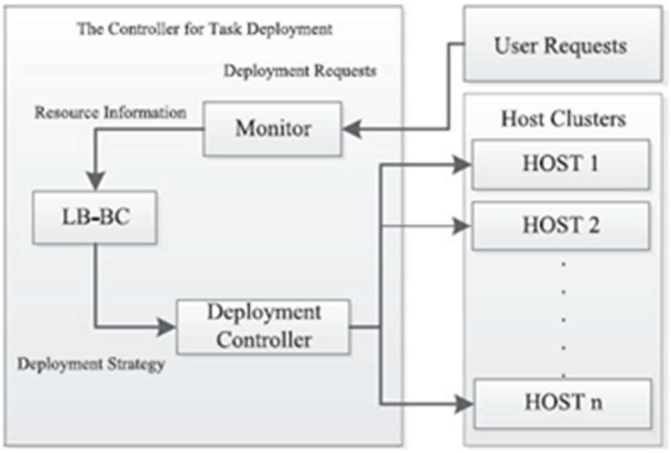

Figure 4. The view of LB-BC's architecture [14]

Fig. 4 describes the architecture of the proposed LB-BC approach in the cloud data center environment. It shows the interaction between LB-BC and other entities and that it plays a very important role in the whole architecture. First, the Monitor acquires the information of resource requested by users' tasks and the status information of remaining resource amount (including CPU and memory) of m available physical hosts in cloud data center. By using the information acquired from Monitor, LB-BC generates the deployment strategy, which is transmitted to Deployment Controller whose function is to control and carry out the deployment of requested tasks. In the end, the task requests accumulated within a (delta)t time are deployed to the corresponding physical hosts in the final optimal set of physical hosts obtained by the proposed LB-BC approach [2].

It calculates the prior probability of each physical host in the network is calculated according to the remaining resource amount of the physical host and the resource amount which the users have requested. Further, Bayes theorem is used to calculate the posterior probability of each physical host. The formulas are as follows:

$$P(B_i|A) = \frac{P(A|B_i) * P(B_i)}{P(A)}$$

$$P(A) = \sum_{i=1}^{m'} P(A|B_i) * P(B_i).$$

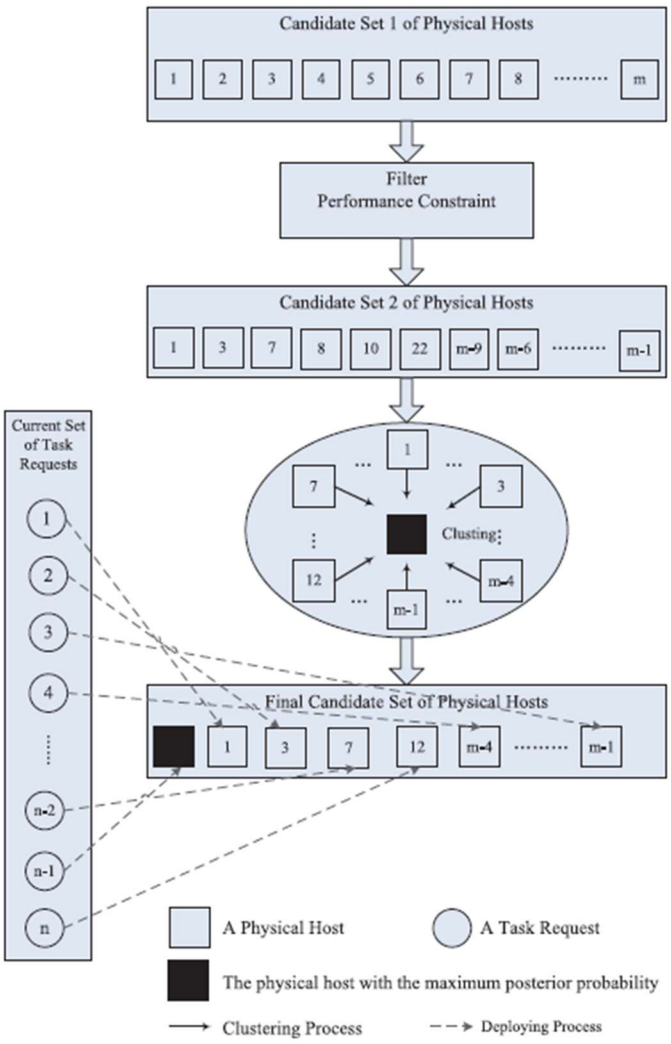

Figure 5. The process of LB-BC task deployment [14]

The task deployment process of LB-BC is shown in Fig. 5. First, these physical hosts, each of which has a larger remaining resource amount than the maximum requested resource amount of all task requests, can be searched out to constitute a new candidate set to meet the performance constraint while making LB-BC have the potential of achieving the long-term load balancing. Second, the k physical hosts in the set of physical hosts can be regarded as k objects waiting for being clustered. Each physical host in the given set is given to a prior probability. The posterior probability of each

physical host's handling tasks can be calculated through Bayes theorem. This probability can be regarded as an attribute of each object while the remaining CPU resource amount and the remaining memory resource amount of each physical host can be regarded as the other two attributes. The similarity degree values between physical hosts are calculated according to the three attributes of each physical host. A threshold value is determined based on these similarity degree values. The physical hosts whose similarity degree values between them are within the given threshold be the optimal clustering to form the final set of candidate physical hosts. Finally, the tasks are placed on the hosts in the final set. And the clustering process of physical hosts in the cloud data center is the process of finding the optimal physical hosts for executing tasks [2].

Algorithm 1. LB-BC(PH,TR,L_c,L_{mem},R_c,R_{mem})

Input: current PH, current L_c, current L_{mem}, received TR, received R_c, received R_{mem};
Output: final deployment solution vector S;

1: $NPH = \phi$, $NPH' = \phi$, S = NULL;
2: **for each** $tr_i \in TR$ **do**
3: $R_i = \alpha R_c^i + \beta R_{mem}^i$;
4: **end for**
5: $L_{Mreq} = \max_{i=1}^{n} R_i$;
6: **for each** $ph_i \in PH$ **do**
7: $L_i = \alpha L_c^i + \beta L_{mem}^i$;
8: **if** $L_i > L_{Mreq}$ **then**
9: $NPH = NPH \cup ph_i$;
10: **end if**
11: **end for**
12: **for each** $nph_i \in NPH$ **do**
13: the posterior probability $P(B_i|A)$ of nph_i is obtained according to the formula (16);
14: **end for**
15: nph_j is the candidate physical host with $P_j = \max_{i=1}^{m'} P(B_i|A)$ in NPH;
16: $NPH' = NPH' \cup nph_j$;
17: **for each** $nph_i \in NPH$ **do**
18: $SD(nph_i, nph_j)$ is obtained according to the formula (17);
19: **if** $SD(nph_i, nph_j) > U_{threshold}^{Similarity}$ **then**
20: $NPH' = NPH' \cup nph_i$;
21: **end if**
22: **end for**
23: **for each** $tr_i \in TR$ **do**
// Assume the n task requests of TR are sorted in terms of FIFO in advance.
24: S[i] = the No. of the physical host nph_i' with the maximum remaining resource amount L_i in NPH';
25: $L_i = L_i - R_i$;
26: **end for**
27: **Return** S;

Figure 6. Algorithm of LB-BC [14]

Summary:

The first step was to use performance values of the nodes and use it in comparison to narrow down the search. In the next step, LBBC obtains the posteriori probability values of all candidate physical hosts using the Bayes theorem. Finally, LB-BC has combined probability theorem and the clustering idea to pick out the optimal hosts set, where these physical hosts have the most remaining computing power currently, for deploying and executing tasks by selecting the physical host with the maximum posteriori probability value as the clustering center and thus to achieve the load balancing effect from the long-term perspective.

Load balancing algorithm for virtual cluster using fuzzy clustering(LB-VC-FC)

Introduction:

There is a vast increase in the usage of the virtual cluster as the virtualization technology develops. It is crucial for virtual clusters to balance the loads of different computing nodes and to maintain high utilization efficiency of system resources. Load balancing algorithms are segmented into two main categories based on the parameters that whether load balancing decisions are based on current node states. The categories are static and dynamic. However, most of the algorithms consider only the loads on the system processing unit to separate the work load, loads of other system resources are ignored, and data links are considered ideal for the network resources and fail to fully consider the features of virtual cluster. Those problems are categorized as follows:

(1) Load balancing for virtual cluster is realized through the transfer of virtual machines, which, compared with the transfer of process or thread, will cause transfer overhead because the amount of transmission data is quite large.

(2) Every virtual machine is a holonomic operating system hosting mangy business applications. Its high-efficiency operation requires not only CPU, but also memory and network.

Overview of fuzzy-based load balancing algorithms: Most existing fuzzy-based load balancing algorithms consider the loads of multi-class system resources, but they cannot truly fuse various loads because the rule based method they utilize is essentially in a combined way. However, the dynamically changing node loads within the virtual cluster cause difficulties in reasonably partitioning the comprehensive loads consisting of multi-dimensional factors. By introducing fuzzy clustering into algorithm design, this paper presents a load balancing algorithm for virtual cluster using fuzzy clustering. To realize the quantization of comprehensive loads, each load of system resources is treated as one dimension of the comprehensive loads and feature weight is attached to each dimension. The improved fuzzy clustering algorithm optimizes distance metric, adds weight constraints and penalty terms, to build the specific objective function for load data of virtual cluster. During load balancing, the suitable objective node for virtual machine migration is located based on clustering partition. Experiments demonstrate that the proposed algorithm can achieve fuzzification of multi-class load data and balance loads within virtual cluster. To evaluate the loads of multi-class resources, recently some researchers have presented algorithms using fuzzy logic to consider the loads of CPU, memory, network and other system resources. a dynamic fuzzy load balancing approach is designed to schedule the tasks to the most suitable virtual machine, based on modelling of loads of memory, bandwidth and discs [4].

Forming objective function for fuzzy clustering:

For node loads data, optimal feature weights in different dimensions are often difficult to obtain, but feature weight preferences, which represent sample characteristics, can be easy to acquire based on existing experience. Therefore, clustering results can be reasonably guided and adjusted according to algorithm presupposition. When calculating the distortion measure between sample points and cluster centers, fuzzy C-

means algorithm does not consider feature weights between different dimensions, failing to reflect the prior knowledge utilized to cluster samples, and causing low algorithm performance. This paper proposes a clustering algorithm that learns and adjusts weight based on priori constraints of feature preferences [4].

Clustering objective function based on feature weight:

$$\min_{\{\omega,\xi\},\{\pi_c\}_{c=1}^{k},\{v_c\}_{c=1}^{k}} \sum_{c=1}^{k}\sum_{x_i\in\pi_c} u_{ci}^{2}\sum_{j=1}^{d}\omega_j d_\phi(x_{ij},v_{cj}) + \lambda_1\sum_{p\in P}\xi_p + \lambda_2\boldsymbol{\omega}^T\boldsymbol{\omega}$$

$$s.t.\quad \boldsymbol{u}\in\Delta_{k\times n}$$

$$\boldsymbol{\omega}\in\Delta_d$$

$$\omega_s-\omega_t\geq\delta-\xi_p \ \ for\ all\ \ p=(s,t,\delta)\in P$$

$$\xi_p\geq 0\ for\ all\ p\in P$$

where lamda1 and lamba2 are non-negative parameters corresponding to feature weight preference and entropy, w is the weight, d is the divergence function used to measure the distortion between the sample data of node loads and clustering centers.

Selection and location strategy of the nodes:

After effectively clustering highly-loaded nodes and lightly-loaded nodes, it needs to select the virtual machine from a highly-loaded node to be transferred to a lightly loaded one. The selection strategy means choosing a virtual machine from highly-loaded nodes as an object to be transferred, based on minimizing migration cost. It is crucial to reduce migration costs, especially transmission data, during load balancing. For the Xen based virtual cluster, the online migration works by iteratively copying the memory mirror to the objective nodes, while Xen hypervisor tracks and re-sends the modified memory pages, that is, dirty pages. Xen hypervisor intercepts the memory access request to synchronize the data between the nodes to be transferred and the nodes to be transferred to, which decreases the performance. By reasonably choosing the virtual

machine for transfer and lowering the transmission data to be copied, the migration time can be effectively reduced, and the influence on applications in virtual machine will be minimized [4].

Summary:

This paper presents a load balancing algorithm for virtual cluster using fuzzy clustering, that considers the loads of CPU, memory, network, and other task related system resources. Compared with existing algorithms that only consider the loads of CPU, the proposed algorithm can calculate the most suitable objective node for virtual machine migration through fuzzy clustering, to realize the balance of CPU loads on all nodes as well as the balance for other system resources. Experiments prove that the algorithm performs more effectively than classical algorithms. The design of simple and efficient algorithm is the next research direction. In addition, efforts should be also dedicated to the optimization of the improved fuzzy clustering method [4].

Conclusion

From our comprehensive study of various classical as well as new cloud load balancing algorithm, we conclude that each algorithm has its own advantages and disadvantages. There is no one solution that solves all the problems but specific solutions focussing on specific issues related to load balancing. It is also important to note, that the selection of of which cloud balancing algorithm to be used depends largely on the type of application and service provided and the load balancing algorithm be chosen needs to be appropriate to the service domain. The classical algorithms are quite simple and quick to implement but disregard major issues such as throughput and focus majorly on execution and completion times. While many improvements on them have been suggested it is not in the scope of this project report to discuss the same. In contrast the clustering based approaches have the following advantages, it efficiently clusters the physical hosts based on the demand of the task resource loads using Bayes probability theorem which reduces the failure number of task deployment events and is designed to achieve the overall load balancing in a long-term process. Load balancing algorithm for virtual cluster evaluates the most suitable objective node for virtual machine migration

through fuzzy clustering to realize the balance of processing unit loads on all nodes and other system resources. The conclusive analysis and comparison of the cloud balancing algorithms will help us in understanding the nuances of cloud load balancing in physical as well as virtual clusters and help in deciding which algorithm is best to use to provide efficient load balancing during tremendous spikes in resource requests.

References

[1] Al Nuaimi, Klaithem, et al. "A survey of load balancing in cloud computing: Challenges and algorithms." Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on. IEEE, 2012.

[2] Bhavya, V. V., K. P. Rejina, and A. S. Mahesh. "An Intensification of Honey Bee Foraging Load Balancing Algorithm in Cloud Computing."

[3] http://www.globaldots.com/knowledge-base/cloud-loadbalancing/cloud_balancer/

[4] Huang, Weihua, et al. "Load balancing algorithm for virtual cluster using fuzzy clustering."Computer and Communications (ICCC), 2016 2nd IEEE International Conference on. IEEE, 2016.

[5] Kansal, Nidhi Jain, and Inderveer Chana. "Cloud load balancing techniques: A step towards green computing." IJCSI International Journal of Computer Science Issues 9.1 (2012): 238-246.

[6] Khara, Satvik, and Umang Thakkar. "A novel approach for enhancing selection of Load Balancing algorithms dynamically in cloud computing." Computer, Communications and Electronics (Comptelix), 2017 International Conference on. IEEE, 2017.

[7] Kokilavani, T., and DI George Amalarethinam. "Load balanced min-min algorithm for static meta-task scheduling in grid computing." International Journal of Computer Applications 20.2 (2011): 43-49.

[8] Lu, Yi, et al. "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services." Performance Evaluation 68.11 (2011): 1056-1071.

[9] Mehta, H., Priyesh Kanungo, and Manohar Chandwani. "Decentralized content aware load balancing algorithm for distributed computing environments." Proceedings of the International Conference & Workshop on Emerging Trends in Technology. ACM, 2011.

[10] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011)

[11] Randles, Martin, David Lamb, and A. Taleb-Bendiab. "A comparative study into distributed load balancing algorithms for cloud computing." Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on. IEEE, 2010.

[12] Wang, S. C.; Yan, K. Q.; Liao, W. P.; Wang, S. S. (2010), "Towards a load balancing in a three-level cloud computing network", Proceedings of the 3rd International Conference on Computer Science and Information Technology (ICCSIT), IEEE: 108–113, ISBN 978-1-4244-5537-9.

[13] Wee, Sewook, and Huan Liu. "Client-side load balancer using cloud." Proceedings of the 2010 ACM Symposium on Applied Computing. ACM, 2010.

[14] Zhao, Jia, et al. "A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment." IEEE Transactions on Parallel and Distributed Systems 27.2 (2016): 305-316.