ENHANCING MINIMAL VIRTUAL MACHINE MIGRATION IN CLOUD ENVIRONMENT

Lidin Das[1], P Mohamed Shameem[2]

[1]*M.Tech Student, Dept. of CSE, TKM Institute of Technology, Kerala, India*
[2]*Associate Professor, Dept. of CSE, TKM Institute of Technology, Kerala, India*

Abstract

Virtualization is a popular solution that acts as a backbone for provisioning requirements of a cloud-based solution. And virtual machine migration is key enabler for dynamic resource management in cloud-based systems. Live virtual machine migration transfers the "state" of a virtual machine from one physical machine to another thereby can mitigate overloaded conditions and enables uninterrupted maintenance activities. In this paper we will come across three main scenarios in virtual machine migration: when, which and where to migrate. Main discussion area in this paper is the scenario, "where to migrate", to choose the destination node to which virtual machine get migrated. A bad choice would lead to a cascade in migration and thereby will create a cyclic effect. So we have to select the better node in order to minimize further migration. For this, we propose a MVMM algorithm to minimize the virtual machine migration.

Keywords: *Cloud Computing, Hot Spot, Live Migration, Virtualization, Virtual Machine*

----------***----------

1. INTRODUCTION

Cloud computing [1] provides a "computing-as- a-service" model in which compute resources are made available as a utility service — an illusion of availability of as much resources (e.g., CPU, memory, and I/O) as demanded by the user. Moreover, cloud users pay only for the amount of resources (a "pay-as-use" model) used by them. This model is different from earlier infrastructure models, where enterprises would invest huge amounts of money in building their own computing infrastructure. Generally, traditional data centers are set to meet the peak demand, which results in wastage of resources during non- peak periods. To mitigate the above problem, modern-day data centers are shifting to the cloud. However, implementing cloud-based data centers requires a great deal of flexibility and agility. For example, the dynamic scaling and shrinking requirement needs compute resources to be made available at very short notice. When computing hardware is overloaded, it may be required to dynamically transfer some of its load to another machine with minimal interruption to the users. Virtualization technology can provide these kinds of flexibilities.

We discuss the use of virtual machine migration [2] for dynamic resource management in virtualized-based cloud systems. As mentioned earlier migration is the process of transferring state of a virtual machine (VM) from one physical machine (PM) to another. Different techniques of migration exits such as suspend-and-copy, pre- copy and post-copy. In suspend-and-copy virtual machine is suspended and copies all its pages and resumes at the destination machine. In pre-copy method it transfers all its pages to the destination without suspending the virtual machine. Once all the necessary pages are transferred VM at the source is suspended and resumes at the new source (destination). Live migration aim to minimize the downtime of virtual machine either by transferring pages before the machine gets suspended or copying minimal state (post-copy) to start the VM and using demand-paging over the network to fetch the remaining state.

In the current cloud computing environments, VM resource scheduling only considers the current system condition and ignores the previous state of system which causes the system load imbalance. Number of VM migrations is more when most of the load balancing takes place. The entire migration cost becomes a problem when most of the VMs are migrated. So it's necessary to minimize the migration of VMs so that we could radically improve the performance of the entire system and saves a much amount of migration cost. So our proposed prediction algorithm provides a solution to the mentioned problem and also embraces multiple aspects and provides an insight into their interactions of today's cloud centers. The main features of our algorithm can be listed as follows:

- Resource usage statics of each VM;
- Predicting job completion time;
- Cascading in migration in avoided;
- Minimizes overloading conditions;

The rest of the paper is organized as follows: In section 2, we survey related work in dynamic resource allocation and live migration in cloud computing environment. In section 3 we will discuss about the proposed system. In section 4 we do the

performance analysis on our proposed algorithm. Finally, in section 4 summarize our findings and conclude the paper.

2. RELATED WORK

Cloud computing has attracted considerable research attention, but only a small portion of the work has been done so far. Many research works carried out in the field of resource allocation and migration. Here, we survey those that proposed certain methods and models for migration and resource allocation.

In [3], the authors studied about resource allocation mechanism that takes place in VM-based data centers. Here they introduce two-tiered on demand resource allocation mechanism that differs from traditional resource allocation mechanisms in adding resource management level for VMs. They proposed on demand resource allocation algorithms and models to make resource allocation more dynamic. But they didn't mention any method to reduce the overloading condition in virtual machines.

In [4], discusses about the framework that is used in managing the clusters of virtual machines. Even though it implements basic management mechanisms such as creating, destroying, and migrating virtual machines, it doesn't mention any method how to minimize the migration of virtual machines.

In [5] and [6], author's mentions about the priority based resource allocation and threats that exits in virtual migration respectively. [7] discusses the design options for migrating OS's running services with liveness constraints, focusing on data center and cluster environments. Author's discusses in detail about the different phases in virtual machine migration. In [8], author discusses about the dynamic resource allocation in cloud environment by using VM migration. They present a system that use virtualization technology to allocate data center resources dynamically based on application demands and support green computing by optimizing the number of servers used. It explains about the situation in which VM get migrated. A VM is migrated when there is hot spot detection. A server is defined as a hot spot when its resource utilization is above a hot threshold value. This indicates that server is overloaded and some of its VMs running on it should be migrated away.

As can be seen, no reported works covers the aspect of minimizing the VM migration. This has motivated us to develop a new algorithm to choose the destination node (server) in such a way that there won't be any cascade in migration. In reference to [8], we also modified the algorithm that to decide when to migrate and which VM to migrate in the way mentioned below.

3. PROPOSED WORK

3.1 When to Migrate

There are many situations when migration of VMs becomes necessary to maintain the overall efficiency of the data center. These situations can be hot spot, periodic, load imbalance, and addition of VMs. We are considering the situation of hot spot over here. As we discussed earlier a server is said to be hot spot if its resource utilization is above a threshold value. Let that threshold value be HT. Now for every node, Ni, we need to find the load of each node and let it be L(Ni). Let the total sum of loads be SL. Now we will calculate the average of loads present in all nodes and introduce another factor called hot spot factor (α). Based on average value and hot spot factor we determine the threshold value. So nodes with loads greater than the threshold value will be considered as hot spot nodes. Step by step algorithm is shown below.

Algorithm

- For each node Ni find load of Ni, i.e. Find L(Ni);
- Set SL=0, α be hot spot factor with value 1.2;
- For each node Ni do the following;
- SL=SL+L(Ni);
- Average (avg)=SL/Ni;
- HT=α * avg; end for;
- For each node Ni repeat up to step 9;
- If L(Ni) > HT then proceed to next step;
- Mark the node as hot spot node; end for;

It is not necessary that all hot spot nodes need to migrate. We will perform the following algorithm on hot spot nodes to determine which node to migrate.

3.2 Which to Migrate

Selecting one or more VMs for migration is a crucial decision of the resource management heuristics. The migration process not only makes the VMs unavailable for a period of time but also consumes resources like network, CPU on source and destination server. So it is important to make the correct decision in choosing which VM to migrate.

Here we would consider only those nodes that are hot spot nodes. Let HN be the list of hot spot nodes. A node or VM is not considered for migration if it is already a migrated one. Here in this algorithm we setup a three threshold values such as threshold-input-length (TIL), min-threshold-input-length (MTIL) and threshold time T. For every node or VM in HN we consider both the total balance input and processed input percentage. We select those nodes or VM with balance input greater than the TIL value and processed input lengths less than MTIL for migration. Step by step algorithm is shown below.

Algorithm

- Initially set values to TIL, MTIL and threshold time T;
- If current-last-migration time > T then go to next step;
- For every node in HN do the following;
- If node (i) not migrated go to next step;
- If balance input (i) > TIL go to next step;
- If processed input percentage (i) < MTIL go to next step;
- Select the node (i) for migration; end for;

3.3 Where to Migrate

During migration destination PM should have enough resources so that it can support incoming migrating VM. Here we will discuss about the MVMM algorithm used to select the destination for migrating VMs. The core part of MVMM algorithm depends upon a VM allocation matrix (VA). So let's see VA matrix in detail.

In VA matrix each column represents VMs in the order of job completion time. This completion time can be calculated on the basis of speed at which each job on the VMs is processed. Each row of VA matrix represents the number of PMs in the entire system. Each value in matrix, say VA[i][j], represents the number of loads pending in PM[i] when VM[j] terminates. Now we will find the HT value using the same method mentioned in the algorithm *when to migrate.* Now we find the number of overloaded nodes on the basis of HT value for each column in VA matrix. We repeat this step for every column and calculate the sum of all overloaded nodes and are set as Migrating Index (MI) value. This MI value plays a pivot role in MVMM algorithm.

The above mentioned VA matrix can be explained with an example. Let VM1, VM2, VM3 and VM4 represents virtual machines in the column of the matrix and is in the order of job completion time. Let PM1, PM2 and PM3 represents the physical machines in the rows of VA matrix.

Table -1: VA matrix

	VM1	VM2	VM3	VM4	
PM1	2	3	1	2	
PM2	4	2	3	3	
PM3	3	3	1	4	
HT	3.6	3.2	2	3.6	
No: of overloaded nodes	1	0	1	1	MI=3

Each value in matrix represents the number of loads pending in each PMs. Here the HT value at the time of termination VM1 is 3.6 (HT=avg * α). Here the avg = 3 ((2+4+3)/3) and value of α is set as 1.2. So at the time VM1 terminates number of nodes that exceeds the HT value is 1. This value is set in the last row.

Now let's see how algorithm works. Let P1, P2.., Pn be the nodes (PM) present in the system. Let MV be the virtual machine to migrate. For each node Pi, let's assume that MV is allocated to node Pi. Now we find the Migration Index (MI) value for every allocation using VA matrix. After calculating all the MI values, we find the node Pi with minimum MI value. So we allocate MV to that node. Hence in following this algorithm we could minimize the migration of VMs. Step by step algorithm is shown below.

MVMM Algorithm

- Let P1, P2,…,Pn be the nodes(PM);
- MV is the VM to migrate;
- For each node Pi, assume MV is allocated to Pi;
- Calculate Migration Index (MI);
- End for;
- Allocate MV to node with minimum MI value;

4. PERFORMANCE ANALYSIS

The resulting algorithms have been implemented and a graph has been plotted. From the table 1, it is clear that Migration Index (MI) is a measure of future migration. And our algorithm considers a different allocation sequence and finally select sequence with minimum MI. But in the case of algorithms used in previous works select any of sequence without considering MI. So as the number of VMs increases there is a drastic increase in migration using normal migration algorithms that leads to a situation that we cannot manage it. But in our algorithm, migration increases only linearly with increase in VMs.

From the graph plotted it is proved that our MVMM algorithm minimizes the virtual machine migration in cloud environment and there by increases overall performance of the cloud system.

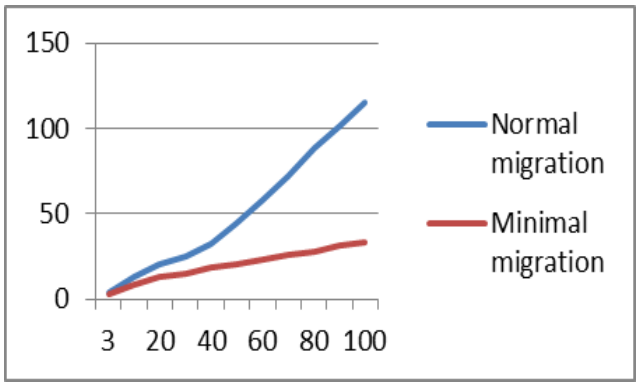

Fig -4: Performance comparison graph

5. CONCLUSIONS

In this position paper, we present a novel approach to minimize virtual machine migration in cloud computing environment. Our modified approach reduces migration overhead up to 75% and the above graph plotted is a proof for that. We only concentrate on minimizing VM migration and eliminate starvation. But we are not considering time complexity factor of our algorithm MVMM

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me help to complete this project. A special thanks to my guide Prof. P Mohamed Shameem, H.O.D., CSE, TKM Institute of Technology.

I am also thankful to staffs of the institution for guiding me and providing me superior computing facilities. Last but not least I would like to thank almighty for making this project a reality.

REFERENCES

[1] M. Armbrust et al., "A view of Cloud Computing," Commun. ACM, vol. 53, no. 4, 2010, pp.50-58.

[2] Michael Nelson, "Fast Transparent Migration for Virtual Machines." 2005

[3] Ying Song et al., "A Two- Tiered On- Demand Resource Allocation Mechanism for VM-Based Data Centers". 2013 IEEE

[4] M. McNett, D. Gupta, A. Vahdat, and G.M. Voelker, "Usher: An Extensible Framework for Managing Clusters of Virtual Machines," Proc. Large Installation System Administration Conf. (LISA '07), Nov. 2007.

[5] Chandrasekhar S. Pawar and Rajnikant B. Wagh, "Priority Based Dynamic resource allocation in Cloud Computing,"

[6] Jon Oberheide, Evan Cooke, and Farnam Jahanian, " Emperical Exploitation of Live Virtual Machine Migration,".

[7] Christopher Clark. 2005 "Live Migration of Virtual Machines". Proc. Symp. Networked Systems Design & Implementation.

[8] Zhen Xiao, Weijia Song, and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment". 2013 IEEE

[9] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," Proc. Symp. Networked Systems Design and Implementation (NSDI '07), Apr. 2007.

[10] C.A. Waldspurger, "Memory Resource Management in VMware ESX Server," Proc. Symp. Operating Systems Design and Implementation (OSDI '02), Aug. 2002.

[11] G. Chen, H. Wenbo, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI '08), Apr. 2008.

[12] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," Proc. IFIP/IEEE Int'l Symp. Integrated Network Management (IM '07), 2007.

[13] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle, "Managing Energy and Server Resources in Hosting Centers," Proc. ACM Symp. Operating System Principles (SOSP '01), Oct. 2001.

[14] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A Scalable Application Placement Controller for Enterprise Data Centers," Proc. Int'l World Wide Web Conf. (WWW '07), May 2007.

[15] M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz, and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," Proc. Symp. Operating Systems Design and Implementation (OSDI '08), 2008.

[16] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters," Proc. ACM Symp. Operating System Principles (SOSP '09), Oct. 2009.

[17] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," Proc. European Conf. Computer Systems (EuroSys '10), 2010.

[18] T. Sandholm and K. Lai, "Mapreduce Optimization Using Regulated Dynamic Prioritization," Proc. Int'l Joint Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '09), 2009.

[19] A. Singh, M. Korupolu, and D. Mohapatra, "Server-Storage Virtualization: Integration and Load Balancing in Data Centers," Proc. ACM/IEEE Conf. Supercomputing, 2008.

[20] Y. Toyoda, "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-One Programming Problems, Management Science, vol. 21, pp. 1417-1427, Aug. 1975.

[21] R. Nathuji and K. Schwan, "Virtualpower: Coordinated Power Management in Virtualized Enterprise Systems," Proc. ACM SIGOPS Symp. Operating Systems Principles (SOSP '07), 2007.

[22] D. Meisner, B.T. Gold, and T.F. Wenisch, "Powernap: Eliminating Server Idle Power," Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '09), 2009.

[23] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce Pc Energy Usage," Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI '09), 2009.

[24] T. Das, P. Padala, V.N. Padmanabhan, R. Ramjee, and K.G. Shin, "Litegreen: Saving Energy in Networked Desktops Using Virtualization," Proc. USENIX Ann. Technical Conf., 2010.

[25] Y. Agarwal, S. Savage, and R. Gupta, "Sleepserver: A Software- Only Approach for Reducing the Energy Consumption of PCS within Enterprise Environments," Proc. USENIX Ann. Technical Conf., 2010.

[26] N. Bila, E.d. Lara, K. Joshi, H.A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan, "Jettison: Efficient Idle Desktop Consolidation with Partial VM Migration," Proc. ACM European Conf. Computer Systems (EuroSys '12), 2012.