International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 2; Issue: 4; April-2014

AN AUTOMATED VIRTUAL MACHINES ALLOCATION FOR USER REQUESTS IN CLOUD WEB-SERVERS

Rashmi Chaurasia
Student, CS/IT
Govt. Mahila Engg. College
chaurasia.rashmi88@gmail.com

Bhawana Maurya
Asst. Professor, CS/IT,
Govt. Mahila Engg. College
bhawanamaurya1@gmail.com

***Abstract*—Now-a-days everything is getting centralized and people are much habituated of doing anything to everything online. There are lots of advantages of this approach as it saves time and also there is 24 / 7 accessibility of information globally. Cloud is comparatively a new technology which allows using its infrastructure as a service. Virtualization adds more advantages as it allows resources to be utilized from multiple virtual servers simultaneously. Though virtual servers are extracted from a single physical server but they have their own allocations of resources and individual configurations. A major issue rises in it when there is disproportionate load on any of the VMs. It is quite obvious that the VM having more load but weak resource would be slow enough to cause delay in processing of any user request .Here in this paper we are going to study load balancing problems and propose a solution which may help in balancing the load on multiple VMs with significant less processing time.**

Key Words – Load Balancing, Cloud Computing, Parallel Processing, Server Virtualization

1. INTRODUCTION

The elasticity and the lack of upfront capital investment offered by cloud computing is appealing to many businesses. There is a lot of discussion on the benefits and costs of the cloud model and on how to move legacy applications onto the cloud platform. Here we study a different problem: how can a cloud service provider best multiplex its virtual resources onto the physical hardware? This is important because much of the touted gains in the cloud model come from such multiplexing. Studies have found that servers in many existing data centers are often severely under-utilized due to over-provisioning for the peak demand [1] [2]. The cloud model is expected to make such practice unnecessary by offering automatic scale up and down in response to load variation. Besides reducing the hardware cost, it also saves on electricity which contributes to a significant portion of the operational expenses in large data centers.

2. RELATED WORK

2.1 Resource allocation at the application level

Automatic scaling of Web applications was previously studied in [3] [4] for data center environments. In MUSE [3], each server has replicas of all web applications running in the system. The dispatch algorithm in a frontend L7-switch makes sure requests are reasonably served while minimizing the number of under-utilized servers. Work [4] uses network flow algorithms to allocate the load of an application among its running instances. For connection oriented Internet

services like Windows Live Messenger, work [5] presents an integrated approach for load dispatching and server provisioning. All works above do not use virtual machines and require the applications be structured in a multi-tier architecture with load balancing provided through an front-end dispatcher. In contrast, our work targets Amazon EC2-style environment where it places no restriction on what and how applications are constructed inside the VMs. A VM is treated like a blackbox. Resource management is done only at the granularity of whole VMs.

MapReduce [6] is another type of popular Cloud service where data locality is the key to its performance. Qunicy adopts min-cost flow model in task scheduling to maximize data locality while keeping fairness among different jobs [7]. The "Delay Scheduling" algorithm trades execution time for data locality [8]. Work [9] assign dynamic priorities to jobs and users to facilitate resource allocation.

2.2 Resource allocation by live VM migration

VM live migration is a widely used technique for dynamic resource allocation in a virtualized environment [10] [11] [12]. Our work also belongs to this category. Sandpiper combines multi-dimensional load information into a single *Volume* metric [10]. It sorts the list of PMs based on their volumes and the VMs in each PM in their volume-to-size ratio (VSR). This unfortunately abstracts away critical information needed when making the migration decision. It then considers the PMs and the VMs in the pre-sorted order. We give a concrete example in Section 1 of the supplementary file where their algorithm selects the wrong VM to migrate away during overload and fails to mitigate the hot spot. We also compare our algorithm and theirs in real experiment. The results are analyzed in Section 5 of the supplementary file to show how they behave differently. In addition, their work has no support for green computing and differs from ours in many other aspects such as load prediction.

The HARMONY system applies virtualization technology across multiple resource layers [12]. It uses VM and data migration to mitigate hot spots not just on the servers, but also on network devices and the storage nodes as well. It introduces the *Extended Vector Product(EVP)* as an indicator of imbalance in resource utilization. Their load balancing algorithm is a variant of the Toyoda method [13] for multi-dimensional knapsack problem. Unlike our system, their system does not support green computing and load prediction is left as future work. In Section 6 of the supplementary file, we analyze the phenomenon that *V ectorDot* behaves differently compared with our work and point out the reason why our algorithm can utilize residual resources better.

Dynamic placement of virtual servers to minimize SLA violations is studied in [11]. They model it as a bin packing problem and use the well-known first-fit approximation algorithm to calculate the VM to PM layout periodically. That algorithm, however, is designed mostly for off-line use. It is likely to incur a large number of migrations when applied in on-line environment where the resource needs of VMs change dynamically.

3. PROBLEMS IN EXISTIG SYSTEM

Instead of dynamic resource allocation, current data processing frameworks rather expect the cloud to imitate the static nature of the cluster environments they were originally designed.

At the moment the types and number of VMs allocated at the beginning of a compute job cannot be changed in the course of processing, although the tasks the job consists of might have completely different demands on the environment.

The major disadvantages of the existing system are that rented resources may be inadequate for big parts of the processing job**,** it may lower the overall processing performance and it increases the cost.

4. PROPOSED SYSTEM WITH SOLUTIONS

In order to overcome from the existing system problems, researcher has proposed valuable solutions in the current application as follows:

- Overload avoidance: the capacity of a PM should be sufficient to satisfy the resource needs of all VMs running on it. Otherwise, the PM is overloaded and can lead to degraded performance of its VMs.
- The Data processing framework to include the possibility of dynamically allocating/ de-allocating different compute resources from a cloud in its scheduling and during job execution.

The key advantages of the proposed system are as follows:

- IaaS cloud's key feature is the provisioning of compute resources on demand
- VMs can be allocated at any time through a well-defined interface
- VMs available in seconds
- Machines which are no longer used can be terminated instantly and the cloud customer will be charged for them no more
- Moreover, cloud operators let their customers rent VMs of different types, i.e. with different computational power, different sizes of main memory, and storage,
- Clouds are highly dynamic and possibly heterogeneous.

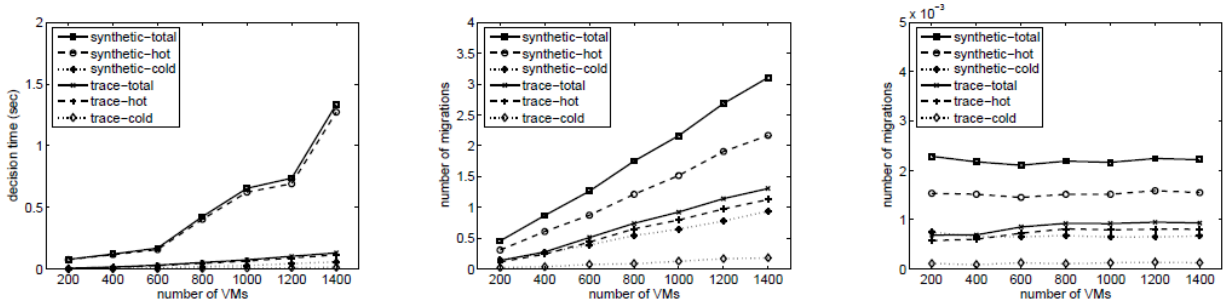
5. PROPOSED ALOGORITHMS

Step 1: Collect user requests

Step 2: Call Dispatcher module ()
Forms batch based on request content type
Calculate total execution time of the batch process
Calls load balancing module ()
Verifies the load on a cluster servers as minimum / maximum

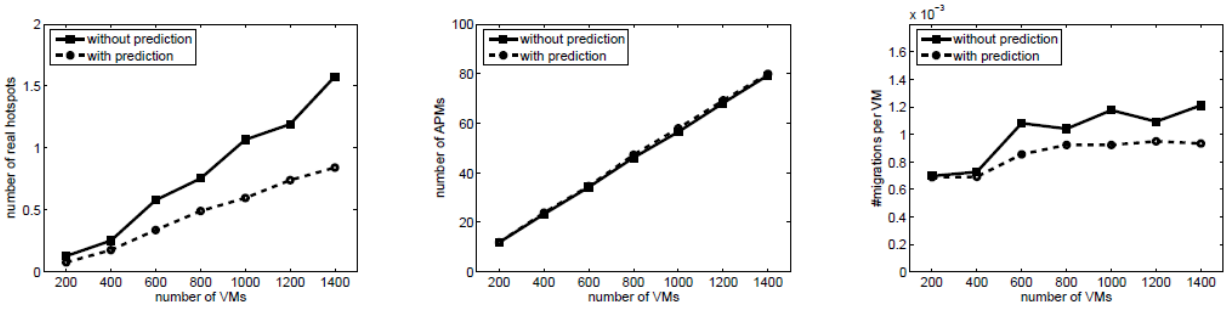
Step 3:Call local cluster server module ()
{
If load on server s_i=minimum then
Call random walk search ()
Update load module table and response table
Else
Call cluster server2
Else
Request execution is rejected
}

6. EXPERIMENTAL RESULTS



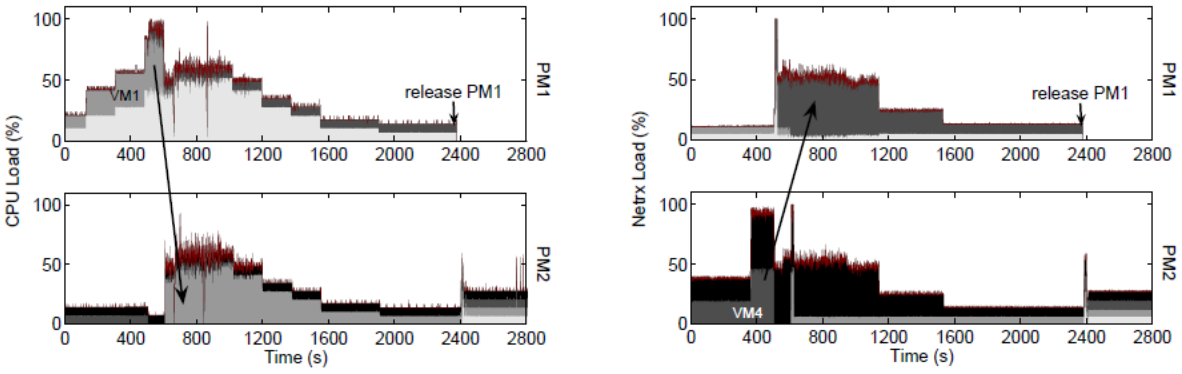
(a) average decision time (b) average number of migrations (c) number of migrations per VM

Scalability of the algorithm with system size

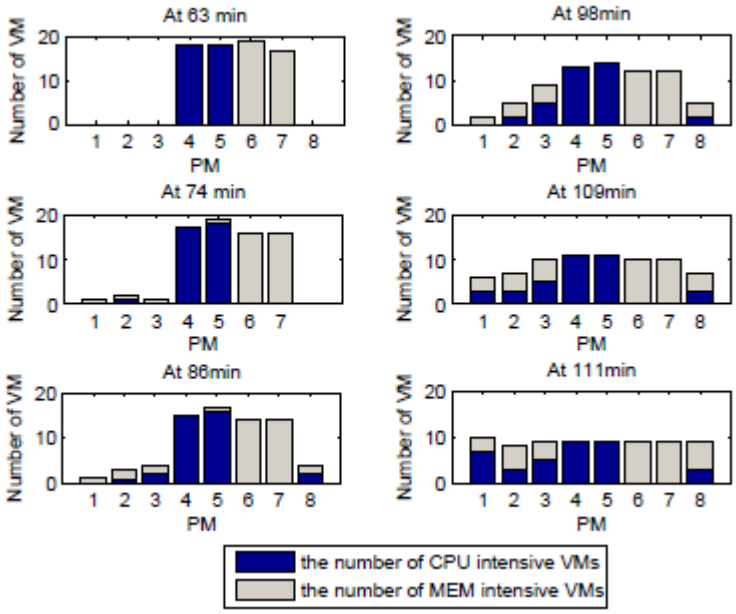


(a) number of hot spots (b) number of APMs (c) number of migration

Effect of Load prediction



Resource balance for mixed workloads



VM distribution over time

7. CONCLUSION

In this paper, we have discussed load balancing problems and proposed a solution which may help in balancing the load on multiple VMs with significant less processing time. Our system multiplexes virtual to physical resources adaptively based on the changing demand. We use the skewness metric to combine VMs with different resource characteristics appropriately so that the capacities of servers are well utilized. Our algorithm achieves both overload avoidance and cloud computing for systems with multi-resource constraints.

8. FUTURE WORK

We may improve the proposed framework ability to adapt to resource overload or underutilization during the job execution automatically. Our current profiling approach builds a valuable basis for this; however, at the moment the system still requires a reasonable amount of user annotations. In general, we think our work represents an important contribution to the growing field of Cloud computing services and points out exciting new opportunities in the field of parallel data processing.

9. BIBLIOGRAPHY

[1] M. Armbrust *et al.*, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep., Feb 2009.

[2] L. Siegele, "Let it rise: A special report on corporate IT," in *The Economist*, Oct. 2008.

[3] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in

hosting centers," in Proc. Of the ACM Symposium on Operating System Principles (SOSP'01), Oct. 2001.

[4] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in Proc. Of the International World Wide Web Conference (WWW'07), May 2007.

[5] 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," in *Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, Apr. 2008.

[6] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in Proc. of the Symposium on Operating Systems Design and Implementation (OSDI'08), 2008.

[7] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair scheduling for distributed computing clusters," in Proc. of the ACM Symposium on Operating System Principles (SOSP'09), Oct. 2009.

[8] 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," in Proc. of the European conference on Computer systems (EuroSys'10), 2010.

[9] T. Sandholm and K. Lai, "Mapreduce optimization using regulated dynamic prioritization," in Proc. of the international joint conference on Measurement and modeling of computer systems (SIGMETRICS'09), 2009.

[10] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. Of the Symposium on Networked Systems Design and Implementation (NSDI'07)*, Apr. 2007.

[11] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*, 2007.

[12] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in Proc. of the ACM/IEEE conference on Supercomputing, 2008.

[13] Y. Toyoda, "A simplified algorithm for obtaining approximate solutions to zero-one programming problems," Management Science, vol. 21, pp.1417–1427, august 1975.