ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (506-511), Month: July - September 2014, Available at: www.researchpublish.com

DYNAMIC RESOURCE ALLOCATION FOR CLOUD COMPUTING ENVIRONMENT USING VIRTUAL MACHINES

[1]B. SIREESHA, [2]E. VENKATA RAMANA

[1,2]Department of Computer Science and Engineering, Vidya Vikas Institute of Technology, Hyderabad, India

Abstract: **Cloud computing allows business customers to scale up and scale down their resource usage based on their needs. Many of the gains in the cloud come from resource multiplexing through virtualization technology. In this paper, we present a system that uses virtualization technology to allocate data center resources dynamically based on application demands and support green computing by optimizing the number of servers in use. We develop a set of heuristics that prevent overload in the system effectively while saving the energy.**

Keywords: **Cloud Computing, Resource Management, Virtualization, Green Computing.**

1. INTRODUCTION

The elasticity and the lack of upfront capital investment offered by cloud computing is appealing to many businesses. This is regarding 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.

Servers in many existing data centers are often severely under-utilized due to over-provisioning for the peak demand. Cloud model makes 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 datacenters.

In Virtual machine monitors (VMMs) provide a mechanism for mapping virtual machines (VMs) to physical resources. This mapping is largely hidden from the cloud users. Users with the Amazon EC2 service we do not know where their VM instances run. It is up to the cloud provider to make sure the underlying physical machines have sufficient resources to meet their needs. VM live migration technology makes it possible to change the mapping between Virtual Machines and Physical Machines while applications are running. This is challenging when the resource needs of VMs are heterogeneous due to the diverse set of applications they run and vary with time as the workloads grow and shrink. The capacity of PMs can also be heterogeneous. We aim to achieve two goals in our algorithm:

- 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.

- Green computing: the number of PMs used should be minimized as long as they can still satisfy the needs of all VMs. Idle PMs can be turned off to save energy.

There is an inherent trade-off between the two goals in the face of changing resource needs of VMs. For overload avoidance, we should keep the utilization of PMs Low to reduce the possibility of overload in case the resource needs of VMs increase later. For green computing, we should keep the utilization of PMs reasonably high to make efficient use of their energy.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (506-511), Month: July - September 2014, Available at: www.researchpublish.com

2. SYSTEM OVERVIEW

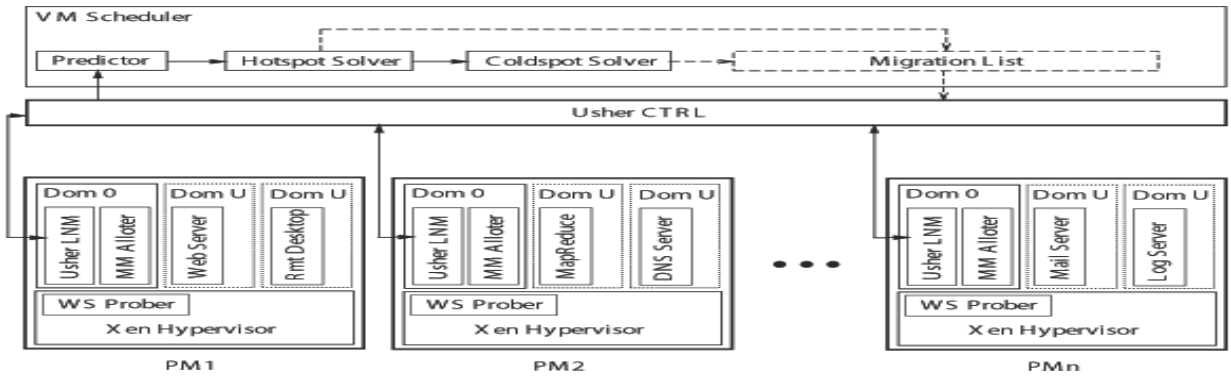


Fig.1

Each PM runs the Xen hypervisor (VMM) which supports a privileged domain 0 and one or more domain U. Each VM in domain U encapsulates one or more applications such as Web server, remote desktop, DNS, Mail, Map/Reduce, etc. We assume all PMs Share backend storage.

The multiplexing of VMs to PMs is managed using the Usher framework. Each node runs an Usher local node manager (LNM) on domain 0 which collects the usage statistics of resources for each VM on that node. The CPU and network usage can be calculated by monitoring the scheduling events in Xen. Working set prober (WS Prober) on each hypervisor estimates the working set sizes of VMs running on it. The statistics collected at each PM are forwarded to the Usher central controller (Usher CTRL) where VM scheduler runs. The VM Scheduler is invoked periodically and receives from the LNM the resource demand history of VMs, the capacity and the load history of PMs, and the current layout of VMs on PMs.

The scheduler has several components. The predictor predicts the future resource demands of VMs and the future load of PMs based on past statistics. The load of the PM will be computed by aggregating the resource usage of its VMs. The LNM at each node first attempts to satisfy the new demands locally by adjusting the resource allocation of VMs sharing the same VMM. Xen can change the CPU allocation among the VMs by adjusting their weights in its CPU scheduler. The MM Allotter on domain 0 of each node is responsible for adjusting the local memory allocation. The hot spot solver in our VM Scheduler detects if the resource utilization of any PM is above the hot threshold. Some VMs running on them will be migrated away to reduce their load. The cold spot solver checks if the average utilization of actively used PMs (APMs) is below the green computing threshold. If so, some of those PMs could potentially be turned off to save energy. It identifies the set of PMs whose utilization is below the cold threshold (i.e., cold spots) and then attempts to migrate away all their VMs. It then compiles a migration list of VMs and passes it to the Usher CTRL for execution.

3. PREDICTING FUTURE RESOURCE NEEDS

We need to predict the future resource needs of VMs. We need to look inside a VM for application level statistics, by parsing logs of pending requests. Doing so requires modification of the VM which may not always be possible. Instead, we make our prediction based on the past external behaviors of VMs. Our first attempt was to calculate an exponentially weighted moving average (EWMA) using a TCP-like scheme:

$$E(t) = \alpha * E(t-1) + (1-\alpha) * O(t),\ 0 \le \alpha \le 1$$

Where E(t) and O(t) are the estimated and the observed load at time t, respectively. _ reflects a tradeoff between stability and responsiveness. We use the EWMA formula to predict the CPU load on the DNS server in our university. We measure the load every minute and predict the load in the next minute.

TABLE 1 Load prediction algorithms

	ewma(0.7) W = 1	fusd(-0.2, 0.7) W = 1	fusd(-0.2, 0.7) W=8
median error	5.6%	9.4%	3.3%
high error	56%	77%	58%
low error	44%	23%	41%

When α is between 0 and 1, the predicted value is always between the historic value and the observed one. To reflect the "acceleration", we take an innovative approach by setting_ to a negative value. When □1 _ _ < 0, the above formula can be transformed into the following:

$$E(t) = -|\alpha| * E(t-1) + (1 + |\alpha|) * O(t)$$

$$= O(t) + |\alpha| * (O(t) - E(t-1))$$

On the other hand, when the observed resource usage is going down, we want to be conservative in reducing our estimation. Hence, we use two parameters, ↑ α and ↓ α to control how quickly E(t) adapts to changes when O(t) is increasing or decreasing, respectively. We call this the FUSD (Fast Up and Slow Down) algorithm. Now the predicted values are higher than the observed ones most of the time: 77% according to Table 1. The median error is increased to 9:4% because we trade accuracy for safety. It is still quite acceptable nevertheless.

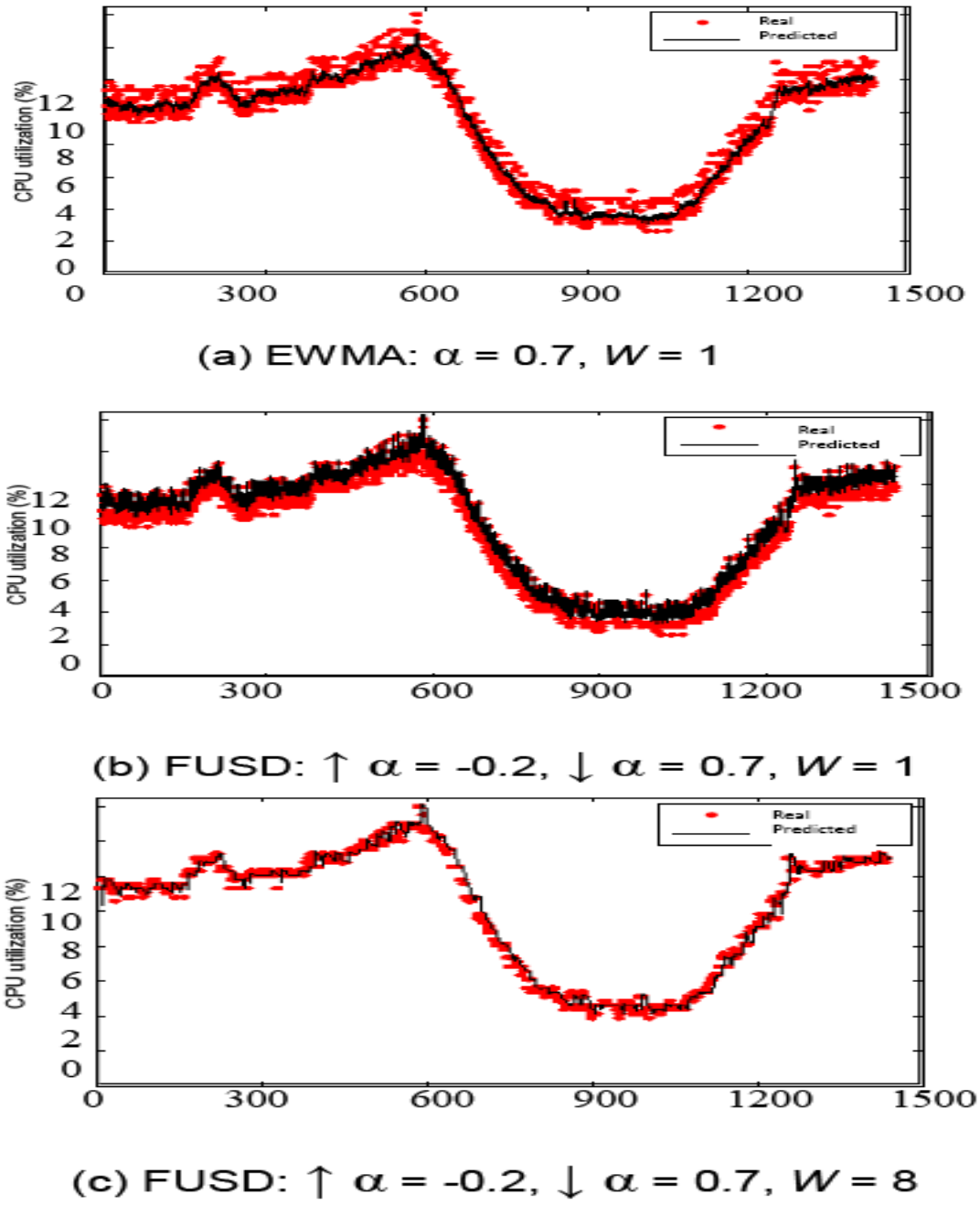


Fig.2. CPU load prediction for the DNS server at our university. W is the measurement window.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (506-511), Month: July - September 2014, Available at: www.researchpublish.com

We compare SPAR(4,2) and FUSD(-0.2,0.7) in figure 3.'lpct' refers to the percentage of low errors while 'std' refers to standard deviation. Both algorithms are used to predict the CPU utilization of the aforementioned DNS server in one-day duration. The predicting window is eight minute.

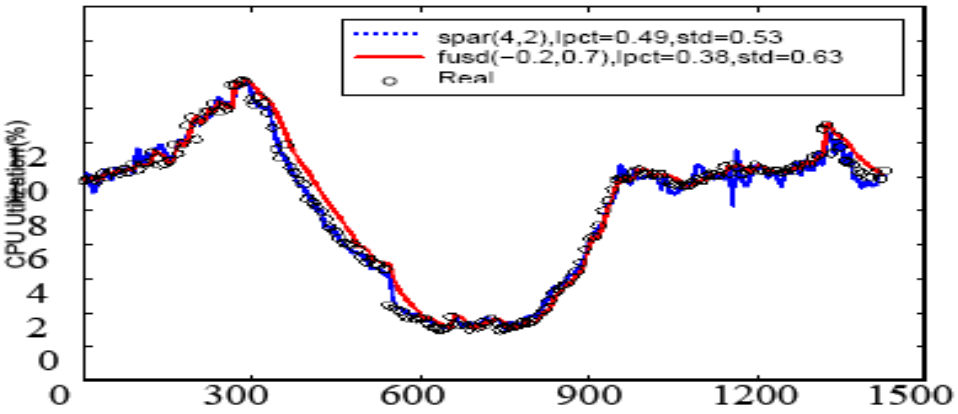


Fig.3. Comparison of SPAR and FUSD

The standard deviation (std) of SPAR (4, 2) is about 16%smaller than that of FUSD (-0.2, 0.7), which means SPAR (4,2) achieves slightly better precision. This is because it takes advantage of tiding pattern of the load. However, SPAR(4,2) neither avoid low prediction nor smooth the load. The requirement of a training phase to determine parameters is inconvenient, especially when the load pattern changes. Therefore we adopt the simpler EWMA variance. Thorough investigation on prediction algorithms are left as future work.

4. THE SKEWNESS ALGORITHM

The skewness quantifies the un evenness in the utilization of multiple resources on a server. Let n be the number of resources we consider and ri be the utilization of the i-th resource. We define the resource skewness of a server p as

$$\text{skewness}(p) = \sqrt{\sum((r_i/\dot{r})-1)^2}$$

Where ṙ is the average utilization of all resources for server p. In practice, not all types of resources are performance critical and hence we only need to consider bottleneck resources in the above calculation. By minimizing the skewness, we can combine different types of workloads nicely and improve the overall utilization of server resources.

4.1. Hot and cold spots

Our algorithm executes periodically to evaluate the resource allocation status based on the predicted future resource demands of VMs. We define a server as a hot spot if the utilization of any of its resources is above a hot threshold. This indicates that the server is overloaded and hence some VMs running on it should be migrated away. We define the temperature of a hot spot p as the square sum of its resource utilization beyond the hot threshold.

A server is a cold spot if the utilizations of all its resources are below a cold threshold. This indicates that the server is mostly idle and a potential candidate to turn off to save energy. This will be done only when the average resource utilization of all actively used servers (i.e., APMs) in the system is below a green computing threshold. A server is actively used if it has at least one VM running. Otherwise, it is inactive. Finally, the warm threshold is a level of resource utilization that is sufficiently high to justify having the server running but not as high as to risk becoming a hot spot in the face of temporary fluctuation of application resource demands.

4.2. Hot spot mitigation

This is meant to eliminate all hot spots if possible. Otherwise, keep their temperature as low as possible. For each server p, we first decide which of its VMs should be migrated away. We sort its list of VMs based on the resulting temperature of the server if that VM is migrated away. We aim to migrate away the VM that can reduce the server's temperature the most. In case of ties, we select the VM whose removal can reduce the skewness of the server the most. For each VM in the list, we see if we can find a destination server to accommodate it. The server must not become a hot spot after

accepting this VM. Among all such servers, we select one whose skewness can be reduced the most by accepting this VM. Note that this reduction can be negative which means we select the server whose skewness increases the least. If a destination server is found, we record the migration of the VM to that server and update the predicted load of related servers. Otherwise, we move on to the next VM in the list and try to find a destination server for it.

4.3. Green computing

When the resource utilization of active servers is too low, some of them can be turned off to save energy. This is handled in green computing algorithm. The challenge here is to reduce the number of active servers during low load without sacrificing performance either now or in the future. We need to avoid oscillation in the system.
Green computing algorithm is invoked when the average utilizations of all resources on active servers are below the green computing threshold. We sort the list of cold spots in the system based on the ascending order of their memory size. Since we need to migrate away all its VMs before we can shut down an under-utilized server, we define the memory size of a cold spot as the aggregate memory size of all VMs running on it. Recall that our model assumes all VMs connect to share back-end storage. Hence, the cost of a VM lives migration is determined mostly by its memory footprint. This consolidation adds extra load onto the related servers. This is not as serious a problem as in the hot spot mitigation case because green computing is initiated only when the load in the system is low

4.4. Consolidated movements

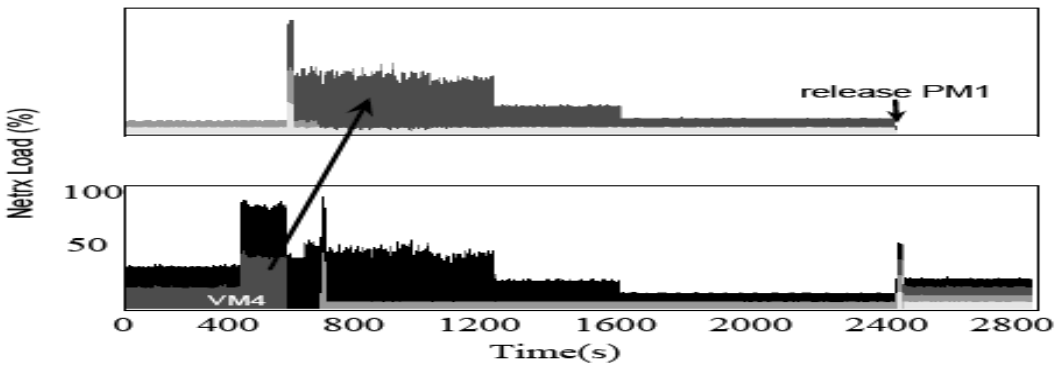
The movements generated in each step above are not executed until all steps have finished. The lists of movements are then consolidated so that each VM is moved at most once to its final destination. Hot spot mitigation may dictate a VM to move from PM A to PM B, while green computing dictates it to move from PM B to PM C. In the actual execution, the VM is moved from A to C directly.

5. RELATED WORK

5.1. Resource allocation at the application level

Automatic scaling of Web applications was previously studied in for data center environments. In MUSE, 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 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 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 a 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 black box. Resource management is done only at the granularity of whole VMs.

Map Reduce 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. The "Delay Scheduling" algorithm trades execution time for data locality. Work assigns dynamic priorities to jobs and users to facilitate resource allocation.



5.2. Resource allocation by live VM migration

VM live migration is a widely used technique for dynamic resource allocation in a virtualized environment. Our work also belongs to this category. Sandpiper combines multi-dimensional load information into a single Volume metric. 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. In addition, their work has no support for green computing and differs from ours in many other aspects such as load prediction.

6. CONCLUSION

Resource management system for cloud computing system multiplexes virtual to physical resources adaptively based on the changing demand. Skewness metric has been used to combine VMs with different resource characteristics appropriately so that the capacities of servers are well utilized. This achieves both overload avoidance and green computing for systems with multi-resource constraints.

REFERENCES

[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] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in Proc. of the ACM Symposium on Operating Systems Principles (SOSP'03), Oct. 2003.

[4] Amazon elastic compute cloud (Amazon EC2), http://aws.amazon.com/ec2/."

[5] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proc. of the Symposium on Networked Systems Design and Implementation (NSDI'05), May 2005.

[6] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in Proc. of the USENIX Annual Technical Conference, 2005.

[7] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker, "Usher: An extensible framework for managing clusters of virtual machines," in Proc. of the Large Installation System Administration Conference (LISA'07), Nov. 2007.

[8] 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.

[9] C. A. Waldspurger, "Memory resource management in VMware ESX server," in Proc. of the symposium on Operating systems design and implementation (OSDI'02), Aug. 2002.

[10] 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.

[11] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in Proc. of the ACM European conference on Computer systems (EuroSys'09), 2009.

[12] 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.

[13] TPC-W: Transaction processing performance council, http://www.tpc.org/tpcw/."

[14] 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.

[15] 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.