Energetic Reserve Distribution using near Machine for Cloud Computing Environment

Ganesh.S
AP/CSE
SCET,SGI
Villupuram ,
India
ganeshsu2006@gmail.com

Elakkiya Selvi.S
Computer Science and Engineering
SCET,SGI
Villupuram,
India
elakki.san@gmail.com

Prema.J
Computer Science and Engineering
SCET,SGI
Villupuram,
India
ekprema.18@gmail.com

Abstract— **Energetic introduce the concept of "skewness" to measure the unevenness in the multi-dimensional reserve consumption of a server. By minimizing skewness, we can combine different types of workloads nicely and improve the overall utilization of server resources. We develop a set of heuristics that prevent overload in the system successfully while saving energy used resource allocation using near equipment 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. Trace driven simulation and experiment results demonstrate that our algorithm achieves good routine.**

Index Terms—Cloud Computing, Resource Management, Virtualization, Green Computing.

I. INTRODUCTION

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. Virtual machine monitors (VMMs) like Xen provide a mechanism for mapping virtual machines (VMs) to physical resources. The capacity of PMs can also be heterogenous because multiple generations of hardware co-exist in a data center. We aim to achieve two goals in our algorithm. Overload avoidance, green computing. We make the following contributions. We develop a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers used. We introduce the concept of "skewness" to measure the uneven utilization of a server. By minimizing skewness, we can improve the overall utilization of servers in the face of multi-dimensional resource constraints. We design a load prediction algorithm that can capture the future resource usages of applications accurately Without looking inside the VMs. The algorithm can capture the rising trend of resource usage patterns and help reduce the placement churn significantly.

II.SYSTEM OVERVIEW

The architecture of the system is presented in Figure 1.Each PM runs the Xen hypervisor (VMM) which support a privileged domain 0 and one or more domain. 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 a backend storage.

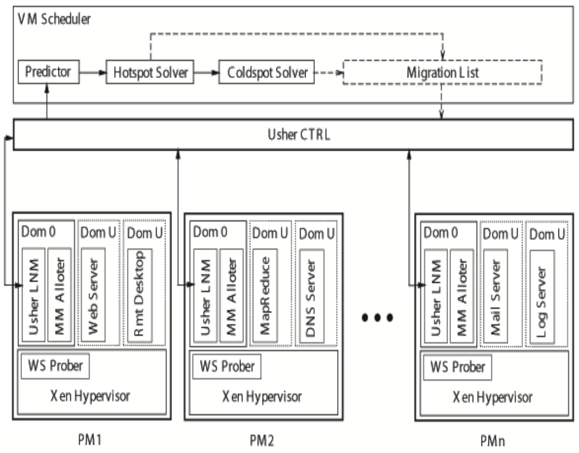

Fig.1. System Architecture.

The multiplexing of VMs to PMs is managed using the Usher framework. The main logic of our system is implemented as a set of plug-ins to Usher .The scheduler has several components. The predictor predicts the future resource demands of VMs and the future load of PMs based on past statistics. We compute the load
of a PM 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* (i.e. hot spot). If so, 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 UsherCTRL for execution. The statistics collected at each PM are forwarded to the Usher central controller (Usher CTRL) where our VMscheduler 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.

III.THE SKEWNESS ALGORITHM

We introduce the concept of *skewness* to quantify the unevenness 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. By minimizing the skewness, we can combine different types of workloadsNicely and improve the overall utilization of server resources. In the following, we describe the details of our algorithm.

A. Hot and cold spots

We define a server as a hot spot if the utilization of any of its resource 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. We do so 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, we define the warm threshold to be a level of resource utilization that is sufficiently high to justify having the server running but not so high as to risk becoming a hot spot in the face of temporary fluctuation of application resource demands. Thus a server is a hot spot if either its CPU usage is above 90% or its memory usage is above 80%.

B. Hot spot mitigation

Our goal is to eliminate all hot spots if possible.Otherwise,keep their temperature as low as possible. We or tits 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, 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.

C. 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 our 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.Our 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 shutdown 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 a shared
back-end storage.

IV.EXPERIMENTS AND RESULTS

The servers are connected over a Gigabit Ethernet to a group of four NFS storage servers where our VM Scheduler runs. We use the same default parameters as in the simulation.

A. Algorithm effectiveness

We evaluate the effectiveness of our algorithm in overload mitigation and green computing. We first increase the CPU load of the three VMs on PM1to create an overload. Our algorithm resolves the overload by migrating VM3 to PM3. It reaches a stable state under high load around 420 seconds. Around 890 seconds, we decrease the CPU load of all VMs gradually. Because the FUSD prediction algorithm is conservative when the load decreases, it takes a while before green computing takes effect. Around1700 seconds, VM3 is migrated from PM3 to PM2 so that PM3can be put into the standby mode. Around 2200 seconds, the two VMs on PM1 are migrated to PM2 so that PM1 can be released as well. As the load goes up and down, our algorithm. Will repeat the above process, spread over or consolidate the VMs as needed.

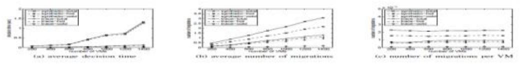

Fig.2 Scalability of the algorithm with system size.

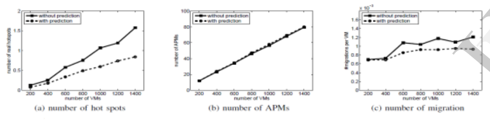

Fig.3 Effect of load prediction

B. Impact of live migration

The use of VM live migration is its impact on application performance. We focus on these migrations because That is when the potential impact on application performance is the most. Among the 139 migrations, we randomly pick 7 corresponding TPC-W sessions undergoing live migration. All these sessions run the "shopping mix" workload with 200 emulated browsers. As a target for comparison, we re-run the session with the same parameters but perform no migration and use the resulting performance as the baseline.

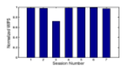

Fig.4 Impact of live migration on TPC-W performance.

The two curves show the moving average over a 30 second window as computed by TPC-W. We marked in the figure when live migration starts and finishes. The figure verifies that live migration causes no noticeable performance degradation. The duration of the migration is under 10 seconds. Recall that our algorithm is invoked every 10 minutes.

C.Resource balance

The memory intensive applications are created by allocating memory on demand. Again we start with a small scale experiment consisting of two PMs and four VMs so that we can present the results for all servers in Figure 5.Initially,the two VMs on PM1 are CPU intensive while the two VMs on PM2 are network intensive. We increase the load of their bottleneck resources gradually. Around 500seconds, VM4 is migrated from PM2 to PM1 due to the network overload in PM2. Then around 600 seconds, VM1 is migrated from PM1 to PM2 due to the CPU overload in

PM1. Now the system reaches a stable state with a balanced resource utilization for both PMs – each with a CPU intensive VM and a network intensive VM.

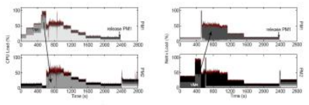

Fig.5 Resource balance for mixed workloads.

V. RELATED WORK

A. Resource allocation at the application level

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 black box. Resource management is done only at the granularity of whole VMs.

B. Resource allocation by live VM migration

VM live migration is a widely used technique for dynamic resource allocation in a virtualized environment. 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. 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.

VI.CONCLUSION

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 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.