ISSN 2277-2685
IJESR/October 2014/ Vol-4/Issue-10/630-633
B. Khalelahamad et al./ International Journal of Engineering & Science Research

DYNAMIC RESOURCE ALLOCATION USING VIRTUAL MACHINES FOR CLOUD COMPUTING ENVIRONMENT

Om Prakash[1], K.Prasanth Kumar[2], B Khalelahamad*[3]

[1]Asst. Prof, JJ Institute of Information Technology, Maheshwarm, R. R. Dist, Telangana, India.

[2]HOD, JJ Institute of Information Technology, Maheshwarm, R. R. Dist, Telangana, India.

[3]M.Tech Pursuing, JJ Institute of Information Technology, Maheshwarm, R. R. Dist, Telangana, India.

ABSTRACT

Cloud computing allows business customers to scale up and down their resource usage based on needs. Many of the touted gains in the cloud model 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 introduce the concept of "skewness" to measure the unevenness in the multi-dimensional resource utilization 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 effectively while saving energy used. Trace driven simulation and experiment results demonstrate that our algorithm achieves good performance.

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.

Virtual machine monitors (VMMs) like Xen provide a mechanism for mapping virtual machines (VMs) to physical resources [3]. This mapping is largely hidden from the cloud users. Users with the Amazon EC2 service [4], for example, do not know where their VM instances run. It is up to the cloud provider to make sure the underlying physical machines (PMs) have sufficient resources to meet their needs. VM live migration technology makes it possible to change the mapping between VMs and PMs while applications are running [5, 6]. However, a policy issue remains as how to decide the mapping adaptively so that the resource demands of VMs are met while the number of PMs used is minimized. 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 because multiple generations of hardware co-exist in a data center.

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.

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 r_i be the utilization of the i-th resource. We define the resource skewness of a server p as

$$skewness(p) = \sqrt{\sum_{i=1}^{n} (\frac{r_i}{\bar{r}} - 1)^2}$$

where r' 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.

SIMULATIONS

We evaluate the performance of our algorithm using trace driven simulation. Note that our simulation uses the same code base for the algorithm as the real implementation in the experiments. This ensures the fidelity of our simulation results. Traces are per-minute server resource utilization, such as CPU rate, memory usage, and network traffic statistics, collected using tools like "perfmon" (Windows), the "/proc" file system (Linux), "pmstat/vmstat/netstat" commands (Solaris), etc.. The raw traces are pre-processed into "Usher" format so that the simulator can read them.

We collected the traces from a variety of sources:

• Web InfoMall: the largest online Web archive in China (i.e., the counterpart of Internet Archive in the US) with more than three billion archived Web pages.

• Real Course: the largest online distance learning system in China with servers distributed across 13 major cities.

• Amazing Store: the largest P2P storage system in China. We also collected traces from servers and desktop computers in our university including one of our mail servers, the central DNS server, and desktops in our department.

We first evaluate the effect of the various thresholds used in our algorithm. We simulate a system with 100 PMs and 1000 VMs (selected randomly from the trace). We use random VM to PM mapping in the initial layout. The scheduler is invoked once per minute. The bottom part of Figure 1 show the daily load variation in the system. The x-axis is the time of the day starting at 8am. The y-axis is overloaded with two meanings: the percentage of the load or the percentage of APMs (i.e., Active PMs) in the system. Recall that a PM is active (i.e., an APM) if it has at least one VM running.

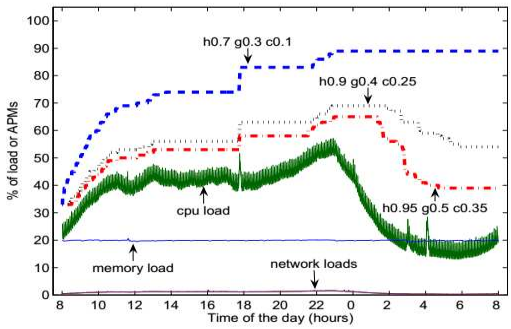
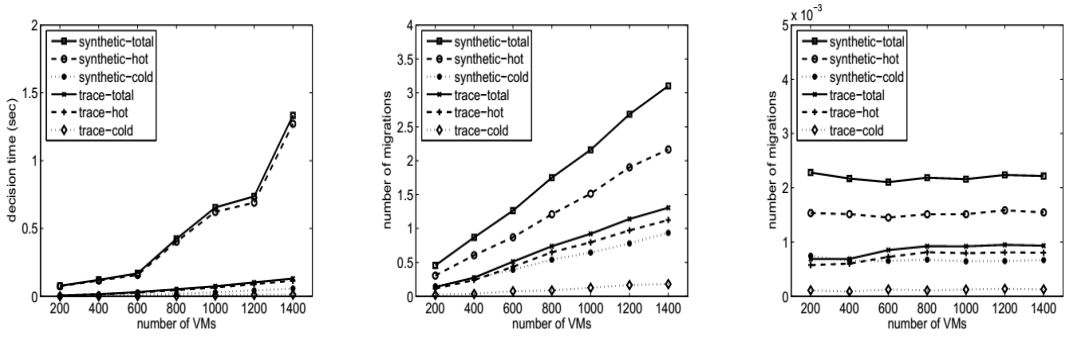


Fig 1: Impact of thresholds on the number of APMs

EXPERIMENTS

Our experiments are conducted using a group of 30 Dell Power Edge blade servers with Intel E5620 CPU and 24GB of RAM. The servers run Xen-3.3 and Linux 2.6.18. We periodically read load statistics using the xenstat library (same as what xentop does). 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.

We evaluate the effectiveness of our algorithm in overload mitigation and green computing. We start with a small scale experiment consisting of three PMs and five VMs so that we can present the results for all servers in figure 7. Different shades are used for each VM. All VMs are configured with 128 MB of RAM. An Apache server runs on each VM. We use httperf to invoke CPU intensive PHP scripts on the Apache server. This allows us to subject the VMs to different degrees of CPU load by adjusting the client request rates. The utilization of other resources are kept low.



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

Fig 2: Scalability of the algorithm with system size

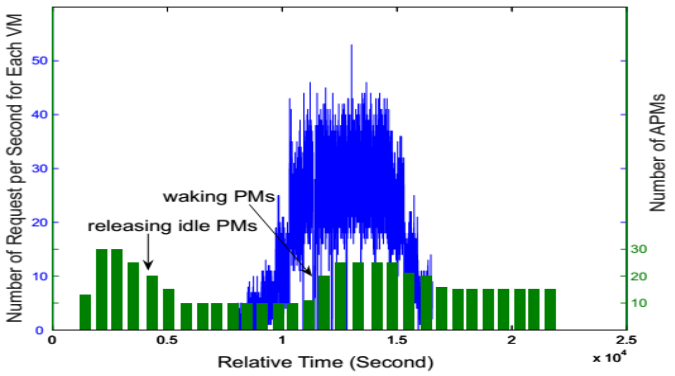


Fig 3. APMs varies with TPC-W load

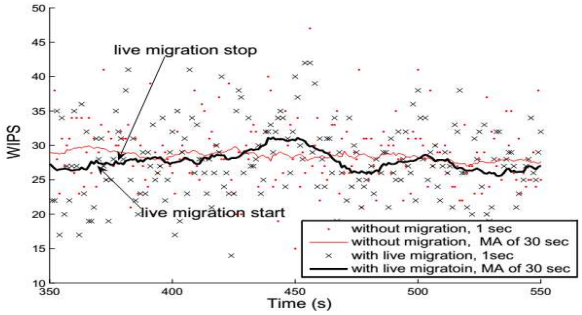


Fig 4: TPC-W performance with and without live migration

CONCLUSION

We have presented the design, implementation, and evaluation of a resource management system for cloud computing services. 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 green computing for systems with multi-resource constraints.

REFERENCES

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

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

[3] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. 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] Clark C, Fraser K, Hand S Hansen JG, Jul E, Limpach C, Pratt I, Warfield A. Live migration of virtual machines. Proc. of the Symposium on Networked Systems Design and Implementation (NSDI'05), May 2005.

[6] Nelson M, Lim BH, Hutchins G. Fast transparent migration for virtual machines. Proc. of the USENIX Annual Technical Conference, 2005.

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

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

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

[10] Chen G, Wenbo H, Liu J, Nath S, Rigas L, Xiao L, Zhao F. Energy-aware server provisioning and load dispatching for connection-intensive internet services. Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), Apr. 2008.

