ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

An Efficient Approach for Virtual Machine Migration of Cloud Systems

S.Akhila
PG Scholar,
Department of CSE,
Aurora's Scientific Technological & Research Academy.

K.Ramakanth
Assistant professor,
Department of CSE,
Aurora's Scientific Technological & Research Academy.

Abstract:

Cloud computing allows business customers to scaleup and down their resource usage based on needs., we present asystem 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 theunevenness in the multidimensional resource utilization of a server By minimizing imbalance, wewill mix completely different 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. Many of the touted gains in the cloud model come from resource multiplexing through virtualization technology. In this paper•Trace driven simulation and experiment results demonstrate thatour algorithm achieves good performance.

Index Terms:

Cloud computing, resource management, virtualization, green computing.

I. INTRODUCTION:

THE physical property and therefore the lack of direct capital investment offered by cloud computing is appealing to several businesses. There's lots of dialogue on the and prices of the cloud model and on a way to move inheritance applications onto the cloud platform. Here we have a tendency to study a special problem: however will a cloud service supplier best multiplex its virtual resources onto the physical hardware. This is often necessary as a result of a lot of of the touted gains within the cloud model return from such multiplexing. Studies have found that servers in many existing data centers are often severely underutilized 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 alsobe heterogeneous because multiple generations of hardware coexist in a data center. We aim to achieve two goals in our algorithm:Overload avoidance. The capacity of a PM shouldbe 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 tradeoff 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. In this paper, we present the design and implementation of an automated resource management system that achieves a good balance between the two goals.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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 s erver. By minimizing skewness, we can improve the overall utilization of servers in the face of multidimensional resource constraints. We design a load prediction algorithm that can capture the future resource usages of application accurately without looking inside the VMs. The algorithm can capture the rising trend of resource usage patterns and help reduce the placement churn significantly.

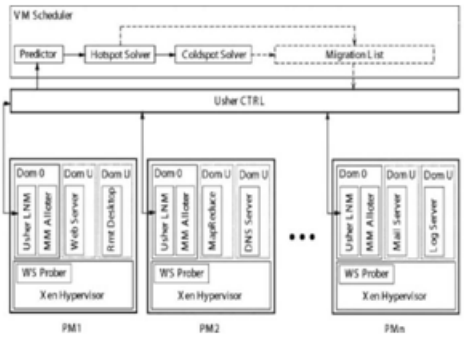


Fig. 1. System architecture

The rest of the paper is organized as follows. Section 2 provides an overview of our system and Section 3 describes our algorithm to predict resource usage. The details of our algorithm are presented in Section 4. Sections 5 and 6 present simulation and experiment results, respectively. Section 7 discusses related work. Section 8 concludes. 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.

2.EXISTING SYSTEM :

The number of servers is comparatively small, typically below 10, which makes them unsuitable for performance analysis of cloud computing data centers. Approximations are very sensitive to the probability distribution of task service times. User may submit many tasks at a time because of this bags-of-task will appear. Due to dynamic nature of cloud environments, diversity of user's requests and time dependency of load is high. The coefficient of variation of task service time is high.

3.PROPOSED SYSTEM :

In Proposed system, the task is sent to the cloud center is serviced within a suitable facility node; upon finishing the service, the task leaves the center. A facility node may contain different computing resources such as web servers, database servers, directory servers, and others. A service level agreement, SLA, outlines all aspects of cloud service usage and the obligations of both service providers and clients, including various descriptors collectively referred to as Quality of Service (QoS). QoS includes availability, throughput, reliability, security, and many other parameters, but also performance indicators such as response time, task locking probability, probability of immediate service, and mean number of tasks in the system, all of which may be determined using the tools of queuing theory

We model a cloud server system which indicates that the inter arrival time of requests is exponentially distributed, while task service times are independent and identically distributed random variables that follow a general distribution with mean value of u. The system under consideration contains m servers which render service in order of task request arrivals (FCFS).The capacity of system is m þ r which means the buffer size for incoming request is equal to r. As the population size of a typical cloud center is relatively high while the probability that a given user will request service is relatively small, the arrival process can be modeled as a Markovian process.

4.THE SKEWNESS ALGORITHM :

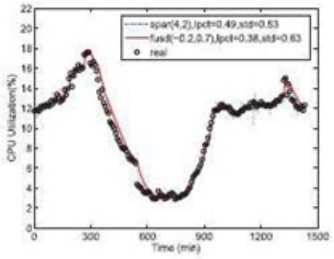
We introduce the concept of skewness to quantify the unevenness in the utilization of multiple resources on a server. Let n be the number of resources we consider and ri be the utilization of the ith resource. We define the resource skewness of a server p as

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

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. In the following, we describe the details of our algorithm.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Analysis of the algorithm is presented in Section 1 in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.283.



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:

$$temperature(p) = \sum_{r \in R} (r - r_t)^2,$$

where R is the set of overloaded resources in server p and rt is the hot threshold for resource r. (Note that only overloaded resources are considered in the calculation.) The temperature of a hot spot reflects its degree of overload. If a server is not a hot spot, its temperature is zero. We define a server as 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. However, 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

5.HOT SPOT MITIGATION:

We sort the list of hot spots in the system in descending temperature (i.e., we handle the hottest one first). Our goal is 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 onto the next VM in the list and try to find a destination server for it. As long as we can find a destination server for any of its VMs, we consider this run of the algorithm a success and then move onto the next hot spot. Note that each run of the algorithm migrates away at most one VM from the overloaded server

Green Computing:

When the resource utilization of activeservers is just too low, a number of them may be turned off to avoid wasting energy. This can be handled in our inexperienced computing rule. The challenge here is to scale back the amount of active servers throughout low load while not sacrificing performance either currently or within the future. We want to avoid oscillation within the system. Our inexperienced computing rule is invoked once the typical utilizations of all resources on active servers are below the inexperienced computing threshold. We tend to type the list of cold spots within the system supported the ascending order of their memory size. Since we want to migrate away all its VMs before we are able to finish off associate degree underutilized server, we tend to outline the memory sizeof a chilly spot because the mixture memory size ofall VMsrunning thereon. Recall that our model assumes all VMs hook up with shared back-end storage. Hence, the value of a VM live migration is decided largely by its memory footprint. Section seven within the supplementary file explains why the memory could be a smart live full. we tend to attempt to eliminate the cold spot with the bottom price initial. For a chilly spot p, we tend to check if we are able to migrate all its VMs elsewhere. For every VM on p, we tend to attempt to realize a destination server to accommodate it. The resource utilizations of the server once acceptive the VMshould

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

be below the nice and cozy threshold. Whereas we are able to save energy by consolidating underutilized servers, overdoing it should produce hot spots within the future. The nice and cozy threshold is intended to forestall that. If multiple servers satisfy the higher than criterion, we tend to like one that's not a current cold spot. This may be as a result of increasing load on a chilly spot reduces the chance that it can be eliminated. However, we'll settle for a chilly spot because the destination server if necessary. All things being equal, we tend to choose a destination server whose lopsidedness may be reduced the foremost by acceptive this VM. If we are able to realize destination servers for all VMs on a chilly spot, we tend to record the sequence of migrations and update the anticipated load of connected servers. Otherwise, we tend to don't migrate any of its VMs. The list of cold spots is additionally updated as a result of a number of them might not be cold because of the projected VM migrations within the higher than method.

Consolidated Movements:

The movements generated in eachstep above are not executed until all steps have finished. The list of movements is then consolidated so that each VM is moved at most once to its final destination. For example, 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.

6.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" comman ds (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. RealCourse. The largest online distance learning system inChina with servers distributed across 13 major cities. AmazingStore.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 post processed the traces based on days collected and use random sampling and linear combination of the data sets to generate the workloads needed. All simulation in this section uses the real trace workload unless otherwise specified. Simulation in this section uses the real trace workload unless otherwise specified..

7.EXPERIMENTS:

Our experiments are conducted using a group of 30 Dell PowerEdge blade servers with Intel E5620 CPU and 24 GB 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.Algorithm Effectiveness: We evaluate the effectiveness of ouralgorithm 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 Fig. 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. We first increase the CPU load of the three VMs on PM1 to 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. Around 1,700 seconds, VM3 is migrated from PM3 to PM2 so that PM3 can be put into the standby mode. Around 2,200 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 repeatthe above process: spread over or consolidate the VMs as needed.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Impact of Live Migration: One concern about the use of VMlive migration is its impact on application performance. Previous studies have found this impact to be small [5]. Weinvestigate this impact in our own experiment. We extract the data on the 340 live migrations in our 30 server experiment above. We find that 139 of them are for hot spot mitigation. 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 seven 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 rerun the session with the sameparameters but perform no migration and use the resulting performance as the baseline. WIPS is the performance metric used by TPC-W. The figure shows that most live migration sessions exhibit no noticeable degradation in performance compared to the baseline: the normalized WIPS is close to

1. The only exception is session 3 whose degraded performance is caused by an extremely busy server in the original experiment Next we take a closer look at one of the sessions in and show how its performance vary over time in The dots in the figure show the WIPS every second. 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. With self-ballooning enabled, the amount of memory transferred during the migration is about 600 MB. 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.

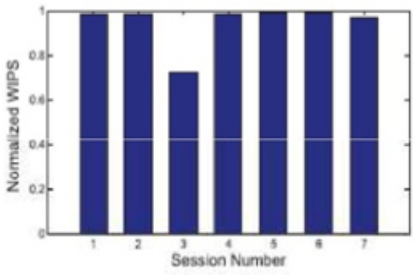


Fig2. Impact of live migration on TPC-W performance.

Resource Balance:

Recall that the goal of the skewnessalgorithm is to mix workloads with different resource requirements together so that the overall utilization of server capacity is improved.

In this experiment, we see how our algorithm handles a mix of CPU, memory, and network intensive workloads. We vary the CPU load as before. We inject the network load by sending the VMs a series of network packets. 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 Fig. 11. The two rows represent the two PMs. The two columns represent the CPU and network dimensions, respectively. The memory consumptionis kept low for this experiment. 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 500 seconds, 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 i ntensive VM and a network intensive VM. Later we decrease the load of all VMs gradually so that both PMs become cold spots. We can see that the two VMs on PM1 are consolidated to PM2 by green computing.

8.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]M. Armbrust et al., "Above the Clouds: A Berkeley V iew of Cloud Computing," technical report, Univ. of California, Berkeley, Feb. 2009.

[2]L. Siegele, "Let It Rise: A Special Report on Corpo rate IT," The Economist, vol. 389, pp. 3-16, 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," Proc. ACM Symp. Operating Systems Principles (SOSP '03), Oct. 2003.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[7]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 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.

[8]C.A. Waldspurger, "Memory Resource Management in VM ware ESX Server," Proc. Symp. Operating Systems Design and I mplementation (OSDI '02), Aug. 2002.

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

[10] 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," Proc. ACM European conf. Computer Systems (EuroSys '09), 2009.

[11]N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," Proc. IFIP/I EEEInt'lSymp. Integrated Network Management (IM '07), 2007.

[12]"TPC-W: Transaction Processing Performance Council, " http:// www.tpc.org/tpcw/, 2012.

[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.