ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)
On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Dynamic Resource Management System for Cloud User Using Virtual Machine Live Migration

Kamakshi R[1], Jony N[2], Malar Priya R[3], KrishnaKumar K[4]

PG Scholar, Dept. of Computer Science and Engineering, Adithya Institute of Technology, Coimbatore, Tamilnadu, India[1, 2, 3, 4]

Abstract— Cloud computing allows cloud user to scale up and down their resource usage based on needs. Many of the cloud models come from resource multiplexing through virtualization technology. In this paper, we present a system that uses virtualization technology to allocate resource dynamically based on application demands and support green computing by optimizing the number of servers in use. We introduce the concept of "Temperature" to measure the unevenness in the multidimensional resource of a server. The result of temperature value, we can combine different types of workloads dynamically 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.

Keyword— Cloud computing, virtual machine live migration, hot and cold spots, resource management, green computing.

I. INTRODUCTION

The elasticity and the lack of upfront capital investment offered by cloud computing is appealing to many businesses. The cloud model comes from multiplexing of virtual resources. 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.

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. 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]. 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.

We aim to achieve a primary goal 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.

We aim to achieve a secondary goal in our algorithm:

- *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.

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.

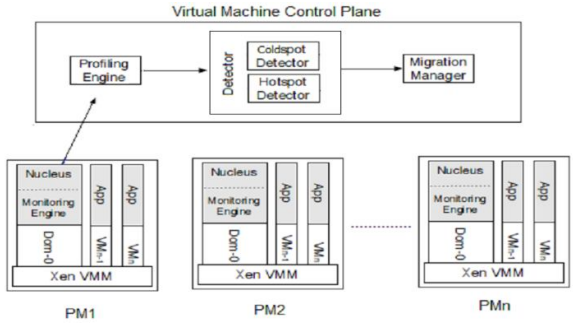


Fig. 1 System architecture

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)
On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

We make the following contribution:

- We develop a resource management system that can avoid overload in the system effectively.
- Using "Temperature" value we can easily detect both hot and cold spot of server.
- We can achieve the number of PMs used should be minimized.
- Idle PMs can be turned off to save energy.
- We introduce the concept of "Temperature" to measure the uneven utilization of a server.

The rest of the paper is organized as follows. Section 2 provides an overview of our system and Section 3 describes our algorithm to dynamically allocate the resource. Section 4 simulation and experiment results, Section 5 discusses related work and Section 6 concludes.

II. SYSTEM OVERVIEW

The architecture of the system is presented in Fig. 1. Each PM runs the Xen hypervisor (VMM) which supports a privileged domain 0 and one or more virtual machine [7], [8]. Each virtual server runs an application or an application component. Xen to implement such architecture. Each virtual server is assumed to be allocated a certain slice of the physical server resources. A application component is called the nucleus on each physical server; the nucleus runs inside a special virtual server (domain 0 in Xen) and is responsible for gathering resource usage statistics on that server (see Fig 1). It employs a monitoring engine that gathers processor, network interface and memory swap statistics for each virtual server. The nuclei periodically relay these statistics to the virtual machine control plane. It comprises four components: a profiling engine, a hotspot detector, cold spot detector and a migration manager (see Fig 1). The profiling engine uses the statistics from the nuclei to construct resource usage profiles for each virtual server and aggregate profiles for each physical server.

The hotspot detector continuously monitors these usage profiles to detect hotspots informally, a hotspot is said to have occurred if the aggregate usage of any resource (processor, network or memory) exceeds a threshold. Thus, the hotspot detection component determines when to signal the need for migrations and invokes the migration manager upon hotspot detection, which attempts hotspot mitigation via dynamic migrations. It implements algorithms that determine what virtual servers to migrate from the overloaded servers, where to move them, and how much of a resource to allocate the virtual servers once the migration is complete. The cold spot detector continuously monitors these usage profiles to detect cold spots informally, a cold spot is said to have occurred if the aggregate usage of any resource (processor, network or memory) is below a threshold. Thus, the cold spot detector if the average utilization of actively used PMs 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. The migration manager assumes that the virtual machine monitor implements a migration mechanism that is transparent to applications and uses this mechanism to automate migration decisions.

III. VIRTUAL MACHINE LIVE MIGRATION ALGORITHM

Virtual machine migration takes a running virtual machine and moves it from one physical machine to another. This migration must be transparent to the guest operating system, applications running on the operating system, and remote clients of the virtual machine. It should appear to all parties involved that the virtual machine did not change its location. The only perceived change should be a brief slowdown during the migration and a possible improvement in performance after the migration because the VM was moved to a machine with more available resources. The migration system presented in this paper is part of the VMware Virtual Center product that manages VMware ESX Server [3]. VMware ESX Server consists of two main components that implement the virtual platform: the virtual machine monitors (VMM) and the vmkernel. A guest operating system such as Windows or Linux runs on top of this virtual platform (see Fig 2). The VMM handles the execution of all instructions on the virtual CPU and the emulation of all virtual devices. The vmkernel schedules the VMM for each virtual machine and allocates and manages the resources needed by the virtual machines.

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)
On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

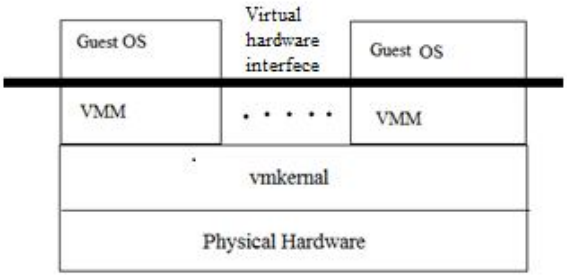


Fig. 2 VM platform layers in VMware ESX Server

Virtual machines provide a natural platform for migration by encapsulating all of the state of the hardware and software running within the virtual machine. There are three kinds of state that need to be dealt with when migrating a VM:

- The virtual device state including the state of the CPU, the motherboard, networking and storage adapters, floppy disks, and raphics adapters.
- External connections with devices including networking, USB devices, SCSI storage devices, and removable media such as CD-ROMs.
- The VM's physical memory.

The actual migration process involves several steps:

- Initiating the migration by selecting the VM to migrate and its destination.
- Pre-copying the memory state of the VM to the destination while the VM is running on the source.
- Quieting the VM and sending the non-memory state.
- Transferring control of the VM to the destination and resuming it at the destination.
- Sending any remaining memory state and removing the dependency on the source machine.

A. 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 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. Different types of resources can have different thresholds. For example, we can define the hot thresholds for CPU and memory resources to be 90 and 80 percent, respectively. Thus a server is a hot spot if either its CPU usage is above90 percent or its memory usage is above 80 percent.

B. 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 go alis 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 temperature 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 temperature can be reduced the most by accepting this VM. Note that this reduction can be negative which means we select the server whose temperature 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

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)
On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

move onto the next hot spot. Note that each run of the algorithm migrates away at most one VM from the overloaded server. This does not necessarily eliminate the hot spot, but at least reduces its temperature. If it remains a hot spot in the next decision run, the algorithm will repeat this process. It is possible to design the algorithm so that it can migrate away multiple VMs during each run. But this can add more load on the related servers during a period when they are already overloaded. We decide to use this more conservative approach and leave the system sometime to react before initiating additional migrations.

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. 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 shut down an underutilized We eliminate cold spots in the system only when the average load of all active servers (APMs) is below the green computing threshold. Otherwise, we leave those cold spots there as potential destination machines for future offloading.

IV. SIMULATION AND EXPERIMENT RESULTS

We evaluate the performance of our algorithm using trace driven simulation. 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 Info Mall: The largest online Web archive in China.
- 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 desktopcomputers in our university including one of our mailservers, the central DNS server, and desktops in ourdepartment.

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 live migration is determined mostly by its memory footprint. The resource utilizations of the server after accepting the VM must be below the warm threshold. While we can save energy by consolidating underutilized servers, overdoing it may create hot spots in the future. The warm threshold is designed to prevent that. If multiple servers satisfy the above criterion, we prefer one that is not a current cold spot. This is because increasing load on a cold spot reduces the likelihood that it can be eliminated.

TABLE 1
Parameter in our simulation

Symbol	Meaning	value
h	hot threshold	0.9
c	cold threshold	0.25
W	Warm threshold	0.65
G	Green computing threshold	0.3

A. *Scalability of the Algorithm*

We evaluate the scalability of our algorithm by varying the number of VMs in the simulation between 200 and 1,400.The ratio of VM to PM is 10:1. The results are shown in Fig. 3. Fig. 3a shows that the average decision time of our algorithm increases with the system size. The speed of increase is between linear and quadratic. We break down the decision time into two parts: hot spot mitigation (marked as "hot") and green computing (marked as "cold").We find that hot spot mitigation contributes more to the decision time. We also find that the decision time for the synthetic workload is higher than that for the real trace due to the large variation in the synthetic workload. Fig. 3b shows the average number of migrations in the whole system during each decision. The number of migrations is small and increases roughly linearly with the system size. We find that hot spot contributes more to the number of migrations. We also find that the number of migrations in the synthetic workload is higher than that in the real trace.

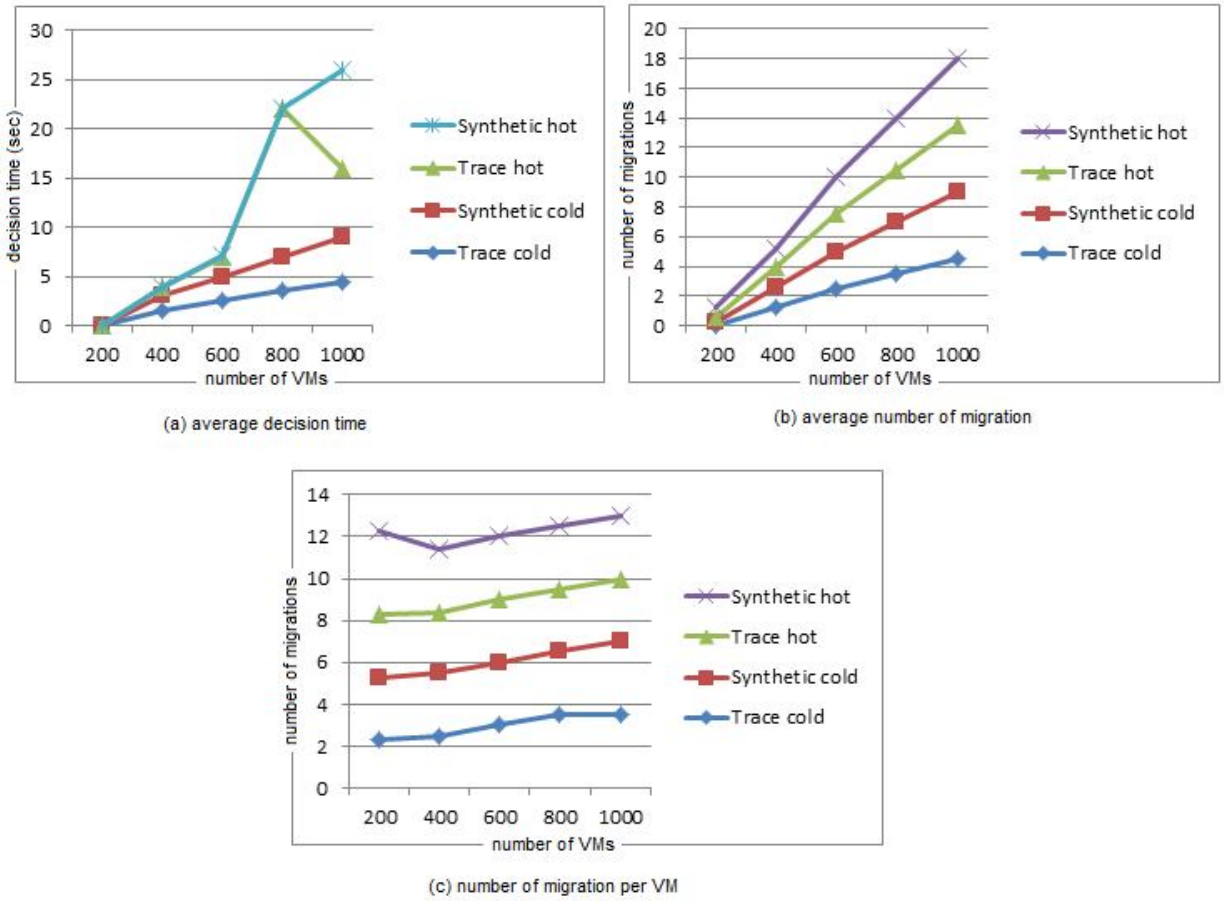
ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)
On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India



(a) average decision time

(b) average number of migration

(c) number of migration per VM

Fig. 3 Migration effectiveness

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

With 140 PMs and 1,400 VMs, on average each run of our algorithm incurs about three migrations in the whole system for the synthetic workload and only 1.3migrations for the real trace. This is also verified by Fig. 5cwhich computes the average number of migrations per VMin each decision.

B. Algorithm Effectiveness

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 wecan present the results for all servers in Fig. 4. Different shades are used for each VM. All VMs are configured with128 MB of RAM. An Apache server runs on each VM. We use http erf to invoke CPU intensive PHP scripts on the Apache server.

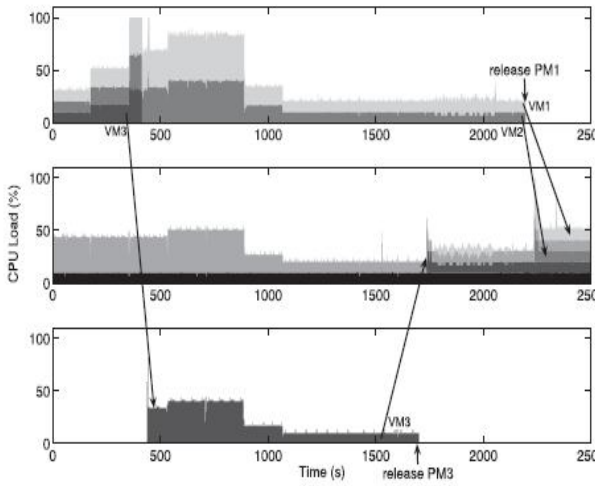


Fig. 4 Algorithm effectiveness

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 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. Around 1,700 seconds, VM3 is migrated from PM3 toPM2 so that PM3 can be put into the standby mode. Around2,200 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 repeat the above process: spread over or consolidate the VMs as needed. Fig. 5 shows how the algorithm spreads the VMs to other PMs over time. As we can see from the figure, the algorithm balances the two types of VMs appropriately. The figure also shows that the load across the set of PMs becomes well balanced as we increase the load.

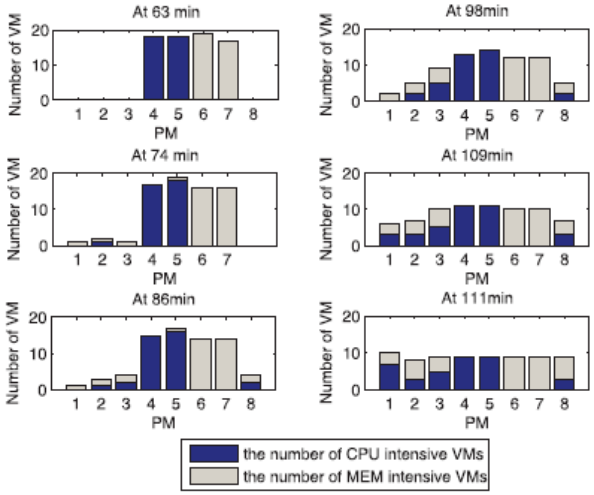


Fig. 5 VM distribution over time

V. RELATED WORK

A. Resource Allocation at the Application Level

Automatic scaling of Web applications was previously studied in [11] and [12] for data center environments. In MUSE [11], each server has replicas of all web applications running in the system. The dispatch algorithm in a frontendL7-switch makes sure requests are reasonably served while minimizing the number of underutilized servers. Work [12]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 [9] 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 multitier architecture with load balancing provided through an front end dispatcher. A VM is treated like a black box. Resource management is done only at the granularity of whole VMs. Map Reduce [13] is another type of popular Cloud service where data locality is the key to its performance. Quincy[14] adopts min-cost flow model in task scheduling to maximize data locality while keeping fairness among different jobs. The "Delay Scheduling" algorithm [15] trades execution time for data locality.

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Work [16] assign dynamic priorities to jobs and users to facilitate resource allocation.

B. Resource Allocation by Live VM Migration

VM live migration is a widely used technique for dynamicresource allocation in a virtualized environment [7], [10],[17]. Our work also belongs to this category. Sandpipercombines multidimensional load information into a singleVolume metric [7]. It sorts the list of PMs based on theirvolumes and the VMs in each PM in their volume-to-sizeratio (VSR), their work has no support forgreen computing and differs from ours in many otheraspects such as load prediction.The HARMONY system applies virtualization technologyacross multiple resource layers [17]. It uses VM and datamigration to mitigate hot spots not just on the servers, butalso 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 [18] formulate dimensional knapsack problem. Unlike our system, their system does not support green computing and load prediction is left as future work. Dynamic placement of virtual servers to minimize SLA violations is studied in [10]. 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.

C. Green Computing

Green computing, also called green technology, is the environmentally responsible use of computers and related resources. Such practices include the implementation of energy-efficient central processing units (CPUs), servers and peripherals as well as reduced resource consumption and proper disposal of electronic waste (e-waste).Many efforts have been made to curtail energy consumption in data centers. Hardware-based approaches include novel thermal design for lower cooling power, or adopting power proportional and low-power hardware. PowerNap [19] resorts to new hardware technologies such as solid state disk (SSD) and Self-Refresh DRAM to implement rapid transition(less than 1ms) between full operation and low power state, so that it can "take a nap" in short idle intervals. When a server goes to sleep, Somniloquy [20] notifies an embedded system residing on a special designed NIC to delegate the main operating system. It gives the illusion that the server is always active. Our work belongs to the category of pure-software low cost solutions [9], [10], [11], [21], [22], [23]. Similar to Somniloquy [20], Sleep Server [22] initiates virtual machines on a dedicated server as delegate, instead of depending on a special NIC.

VI. CONCLUSION

We have presented the design, implementation, and evaluation of a resource management system for cloud user. Our system multiplexes virtual to physical resources adaptively based on the user needs. We use the temperature metric to combine VMs with different resource characteristics appropriately so that the capacities of servers are well utilized. Our algorithm achieves main goal as overload avoidance and secondary goal as green computing for systems with multisource constraints.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable feedback. The authors also express our sincere thanks to Asst.Prof. Mrs.M.Sangeetha Department of Computer Science and Engineering, Adithya Institute of Technology, Coimbatore, has shown keen interest throughout our process. With her potent ideas and excellent guidance we are able to comprehend the essential aspects involved.

REFERENCES

[1]. M. Armbrust et al, "Above the Clouds: A Berkeley View of Cloud Computing," technical report, Univ. of California, Berkeley, Feb. 2009.

[2]. L. Siegele, "Let It Rise: A Special Report on Corporate 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 ofVirtualization," Proc. ACM Symp. Operating Systems Principles(SOSP '03), Oct. 2003.

[4]. "Amazon elastic compute cloud (Amazon EC2)," http://aws. amazon.com/ec2/, 2012.

[5]. C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines,"Proc. Symp. Networked Systems Design and Implementation (NSDI '05), May 2005.

[6]. M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," Proc. USENIX Ann. TechnicalConf., 2005.

[7]. 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 VMware ESX Server," Proc. Symp. Operating Systems Design and Implementation (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 Dispatching for Connection-Intensive Internet Services,"

ISSN (Online) : 2319 – 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization, Volume 3, Special Issue 1, February 2014

International Conference on Engineering Technology and Science-(ICETS'14)

On 10th & 11th February Organized by

Department of CIVIL, CSE, ECE, EEE, MECHNICAL Engg. and S&H of Muthayammal College of Engineering, Rasipuram, Tamilnadu, India

Proc. USENIX Symp. Networked Systems Design and Implementation(NSDI '08), Apr. 2008.

[10]. N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," Proc. IFIP/IEEE Int'l Symp. Integrated Network Management (IM '07), 2007.

[11]. 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.

[12]. 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.

[13]. 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.

[14]. M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters," Proc. ACM Symp. Operating System Principles(SOSP '09), Oct. 2009.

[15]. M. Zaharia, D. Borthakur, J. SenSarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique forAchieving Locality and Fairness in Cluster Scheduling," Proc. European Conf. Computer Systems(EuroSys '10), 2010.

[16]. T. Sandholm and K. Lai, "Mapreduce Optimization Using Regulated Dynamic Prioritization," Proc. Int'l Joint Conf. Measurement and Modeling of Computer Systems(SIGMETRICS '09), 2009.

[17]. Singh, M. Korupolu, and D. Mohapatra, "Server-Storage Virtualization: Integration and Load Balancing in Data Centers,"Proc. ACM/IEEE Conf. Supercomputing, 2008.

[18]. Y. Toyoda, "A Simplified Algorithm for Obtaining Approximate Solutions to Zero-One Programming Problems," Management Science, vol. 21, pp. 1417-1427, Aug. 1975.

[19]. D. Meisner, B.T. Gold, and T.F. Wenisch, "Powernap: Eliminating Server Idle Power," Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '09), 2009.

[20]. Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce Pc Energy Usage," Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI '09), 2009.