Autonomous Management System for Dynamic Resource Allocation in Cloud Computing

Poral Nagaraja,
Dept.ofCS&E,
S.J.M.I.T.Chitradurga-577 502.

Tejaswini.S. C,
P.G.Student,
Dept.ofCS&E,
S.J.M.I.T.Chitradurga-577 502.

ABSTRACT
Cloud Computing allows resource usage based on needs of the business customers. In this paper, we present an autonomous management system that uses virtualization technology to allocate data center resources effectively based on application demands that supports green computing to save energy used by optimizing the number of servers in use. We implement the concept of "skewness" to measure the irregularity in the multi-dimensional resource utilization of a server. By reducing skewness, we can improve the overall utilization of server resources and combine different types of workloads fairly. Cloud SIM simulation and experiment results demonstrate that our algorithm achieves better performance.

General Terms
Algorithms, virtual machine, physical machine, overload.

Keywords
Cloud computing, resource management, virtualization, green computing.

1. INTRODUCTION
The plasticity and the lack of advance 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 transfer legacy applications onto the cloud platform. Here, we analyze a different problem: how can cloud service providers multiplex its virtual resources onto the physical hardware in a better way? This is important because, much of the supported gains in the cloud model come from such multiplexing. Studies have found that servers in many existing data centers are often severely underutilized due to over provisioning of the peak demand [1], [2]. The cloud model is expected to make such practice unnecessary by providing automatic scale up and down in response to load variation. Besides minimizing the hardware cost, it also saves power consumption 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 concealed from the cloud users. Users with the Amazon EC2 service, for example, do not know where their VM samples run. It is up to the cloud provider to make sure that the underlying Physical Machines (PMs) have plenty of resources to meet their demands. 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 needs of VMs are met while the number of PMs used is reduced. This is challenging when the resource needs of VMs are heterogeneous due to the various set of applications they run and differs with time as the workloads grow and shrink. The capacity of PMs can also be heterogeneous because multiple generations of hardware coincide in a data center. We aim to achieve two goals in our algorithm:

- Overload avoidance: The capacity of a PM should be sufficient to fulfill the resource demands 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 fulfill the demands of all VMs. Idle PMs can be turned off to save power.

There is an inherent tradeoff between the two goals in the face of varying resource demands of VMs. For overload avoidance, we should keep the utilization of PMs low t minimize the possibility of overload in case the resource demands of VMs increase later. For green computing, we should keep the utilization of PMs reasonably high to make efficient use of their power. In this paper, we present the design and implementation of an automated resource management system that achieves a good balance between the two goals. 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 used the concept of "skewness" to measure the uneven usage of a server. By minimizing skewness, we can enhance the overall utilization of servers in the face of multi-dimensional resource constraints.
- We design a load prediction algorithm that can predict 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 significantly.

2. RELATED WORKS

2.1 Resource Allocation at the Application Level
Automatic scaling of Web applications was previously studied in [14] and [15] for data center environments. In MUSE [14], each server has duplicates of all web applications running in the system. The dispatch algorithms in a frontend L7-switch ensure that requests are reasonably served while minimizing the number of underutilized servers. Work [15] 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 [10] shows an integrated approach for load transmitting and server provisioning. Virtual machines are not used in above all works and they require the applications be designed in a multitier architecture with load balancing provided through a front-end dispatcher. In contrast, our work targets Amazon

EC2-style environment where it places no limitation on what and how applications are established inside the VMs. A VM is considered as a black box. Resource management is done only at the granularity of all VMs.

Map Reduce [16] is another type of popular Cloud service where data locality is very important to its performance. Quincy [17] adopts min-cost flow model in job scheduling to increase data locality while keeping fairness among various jobs. The "Delay Scheduling" algorithm [18] trades execution time for data locality. Work [19] assigns dynamic priorities to jobs and users to facilitate resource allocation.

2.2 Resource Allocation by Live VM Migration

VM live migration is a popular technique for dynamic resource allocation in a virtualized environment [8], [12], [20]. Our work also belongs to this group. Sand piper merges multidimensional load information into a single Volume metric [8]. It arranges the list of PMs based on their volumes and the VMs in each PM in their Volume-To-Size Ratio (VSR). This unfortunately conceals critical information needed when making the migration decision. It then considers the PMs and the VMs in the prearranged order. An example of the supplementary file is available online, where the algorithms choose the wrong VM to migrate away during overload and fails to reduce the hot spot and also the algorithms and their results in the experiment are compared. In addition, the work has no support for green computing and differs from various other aspects such as load prediction.

The HARMONY system applies virtualization technology across multiple resource layers [20]. It uses VM and data migration to reduce hot spots not only 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 usage. Their load balancing algorithm is an alternative of the Toyoda method [21] for multidimensional knapsack problem. Unlike the system, it does not support green computing and load prediction is left for enhancement. The analysis of the phenomenon Vector Dot behaves differently compared with the work and make out the reason why the algorithm can use residual resources better.

Dynamic placement of virtual servers to reduce Service Level Agreement (SLA) violations is studied in [12]. They model it as a bin packing problem and use the well-defined first-fit approximation algorithm to calculate the VM to PM layout statistically. That algorithm, however, is structured mostly for offline use. It is likely to incur a large number of migrations when applied in online environment where the resource demands of VMs change dynamically.

2.3 Green Computing

Many efforts have been made to minimize power consumption in data centers. Hardware-based approaches include novel thermal design for low-power hardware, or adopting power proportional and lower cooling power. Work [22] uses Dynamic Voltage and Frequency Scaling (DVFS) to adjust CPU power according to its load. We do not use DVFS for green computing. Power Nap [23] resorts to new hardware technologies such as Solid State Disk (SSD) and Self-Refresh DRAM to implement quick transition(less than 1ms) between complete operation and low power state, so that it can "take a nap" in short idle intervals. When the server goes idle, Somniloquy [24] notifies an embedded system situating on a special designed NIC to delegate the main operating system which gives an illusion that the server is always active.

Our work belongs to the category of pure-software low cost solutions [10], [12], [14], [25], [26], [27]. Similar to Somniloquy [24], Sleep Server [26] initiates virtual machines on a dedicated server as delegate, instead of depending on a special NIC. LiteGreen [25] does not use a delegate. Instead it migrates the desktop OS away so that the desktop is in stand by position. It requires that the desktop is virtualized with distributed storage. Jettison [27] invents "partial VM migration," a variance of live VM migration, which migrates away only the necessary working set while leaving infrequently used data behind.

3. SYSTEM OVERVIEW

The architecture of the system is shown in Figure 1.EachPMthat runs Virtual Machine Monitoring (VMM) such as Xen hypervisor supports a privileged datacenter and one or more hosts. Each virtual machine in host, encapsulates one or more applications like remote desktop, Mail, web server, map/reduce DNS, etc. We assume all PMs Share backend storage.

The multiplexing of VMs to PMs is managed using the usher framework in network topology. The main logic of the system is implemented as a set of plug-ins to usher. Each host runs an Usher Local Node Manager (LNM) on datacenter which collects the usage statistics of resources for each VM on that node. The CPU and network utilization can be calculated by monitoring the scheduling events in VMM. The memory utilization within a VM, however, is not visible to the VMM. One approach is to infer lack of memory in VM by observing its swap activities. Unfortunately, the host OS is required to install a separate swap partition. Furthermore, it may be too late to adjust the memory allocation while time swapping occurs. Instead, we implemented a working set prober on each VMM to estimate the working set sizes of VMs running on it.

The statistics collected at each PM is passed to the central controller where our VM scheduler runs. The VM scheduler is invoked periodically and it accepts from the LNM the resource demand history of VMs, the ability and the load history of PMs, and the current layout of VMs on PMs.

The scheduler in Figure 1 has many components. The predictor predicts the future resource demands of VMs and the future load of PMs based on earlier statistics. We compute the load of a PM by aggregating the resource utilization of its VMs. The details of the load prediction algorithm will be described in the next section. The LNM at each host first attempts to satisfy the new demands locally by adjusting the resource allocation of VMs sharing the same VMM. VMM can vary the CPU allocation among the VMs by adjusting their weights in its CPU scheduler. The Memory Manager (MM) Allotter in datacenter of each host is responsible for adjusting the local memory allocation.

The hot spot solver in the VM Scheduler detects if the resource usage of any PM is above the hot threshold (i.e., a hot spot). If so, some VMs running on them will be migrated away to minimize their load. The cold spot solver verifies if the average usage of actively used PMs is below the green computing threshold. If so, some of those PMs could potentially be turned off to save power. It identifies the set of PMs whose usage is below the cold threshold (i.e., cold spots) and then tries to migrate away all their VMs. It then compiles a migration list of VMs and forwards it to the CTRL. for execution.

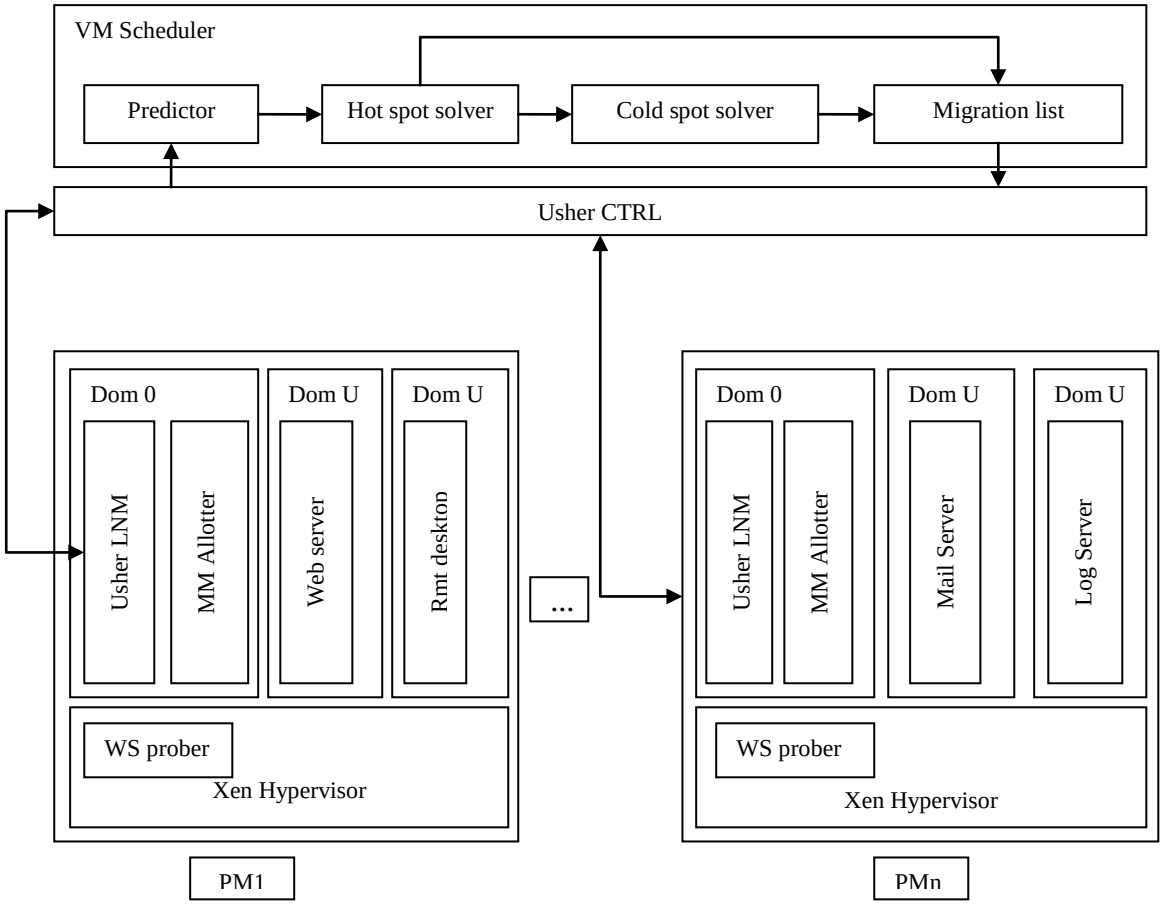

Figure1.System Architecture

4. PREDICTING FUTURE RESOURCE NEEDS

It is required to predict the future resource needs of VMs. Concentrating on Internet applications, one solution is to look inside a VM for application level statistics, e.g., by parsing logs of pending requests. Doing so, requires alteration of the VM which may not always be positive. Instead, we make our prediction based on the historic external behaviors of VMs. Our first attempt was to calculate an Exponentially Weighted Moving Average (EWMA) using a TCP-like scheme

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

Where $E(t)$ and $O(t)$ are the estimated and the monitored load at time t, respectively. α reflects a tradeoff between stability and responsiveness.

Table 1.Load Prediction Algorithms

	ewma(0.7) W=1	fusd(-.2,0.7) W=1	fusd(-.2,0.7) W=8
median error	5.6%	9.4%	3.3%
high error	56%	77%	58%
low error	44%	23%	41%

The parameters in the parenthesis are the α values. W is the length of the measurement window used to represent the recently observed values. The "median" error is calculated as a percentage of the monitored value:$|E(t) - O(t)|/O(t)$. The "higher" and "lower" error percentages shown in Table 1 are the percentages of predicted values that are higher or lower than the monitored values, respectively. The prediction is fairly accurate with approximately equal percentage of higher and lower values.

Although seemingly satisfactory, this formula does not capture the future trends of resource usage. For example, when we see a sequence of $O(t)$ =10; 20; 30, and 40, it is reasonable to predict the future value to be 50. Unfortunately, when α is between 0 and 1, the predicted value is always between the past value and the observed value. To reflect the "acceleration," we considered an innovative approach by setting α to a negative value. When$-1 \leq \alpha < 0$, the above formula can be transformed into the following:

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

When the resource usage is decreasing, we want to be conservative in minimizing our estimation. Hence, we use two parameters, $\uparrow \alpha$ and$\downarrow \alpha$, to know how rapidly $E(t)$ adapts to modification when $O(t)$ isincreasing or decreasing, respectively. This is called as FastUp and Slow Down (FUSD) algorithm.

5. THE SKEWNESS ALGORITHM

The concept of skewnessis used to quantify the unevenness in the utilization of multiple resources on a server [28]. Let n be the number of resources and ri be the utilization of the i[th] resource. The resource skewness of a server p is given as

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

where ris the average utilization of all resources for server p. In practice, not all types of resources are performance crucial and hence we need to consider bottleneck resources in the calculation. By minimizing the skewness, the system can

combine various types of workloads fairly and improve the overall utilization of server resources.

5.1 Hot and Cold Spots

The algorithm is executed periodically to evaluate the resource allocation status based on the predicted future resource needs of VMs. If the server has the utilization of any of its resources above a hot threshold, it is called hot spot. This shows that the server is overloaded and hence some VMs running on it should be moved away. The temperature of a hot spot p is the square sum of its resource utilization beyond the hot threshold given as

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

where R is the set of overloaded resources in server p and $ri$is the hot threshold for resource r. The temperature of a hot spot reveals its degree of overload. If the temperature is zero, then that server is not a hot spot. If server has the utilization of all its resources below a cold threshold, it is called cold spot. This indicates that the server is mostly inactive and a potential candidate to turn off to save power. However, it happens only when the average resource usage 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 idle. Finally, the warm threshold is a level of resource utilization that is high enough to justify having the server running but not as high as risk that becomes a hot spot in the face of temporary variation of application resource demands. Various types of resources can have different thresholds.

For example, consider the hot thresholds for CPU and memory resources to be 80 and 90percent, respectively. Thus a server is a hot spot if either its CPU usage is above80 percent or its memory usage is above 90 percent.

5.2 Hot Spot Mitigation

The list of hot spots in the system is in descending order of temperature (i.e., handle the hottest one first). The goal is to remove all hot spots if possible or else to keep their temperatures as low as possible. For each server p, first decide that which of its VMs should be migrated away. The list of VMs is arranged based on the resulting temperature of the server if that VM is migrated away.

We aim to migrate away the VM that can minimize the server's temperature the most. Here, we tried to select the VM whose elimination can reduce the skewness of the server the most. Each VM in the list is required to find a destination server to accommodate. After accepting this VM, the server must not become a hot spot. Among all such servers, select one whose skewness can be reduced the most by accepting this VM. Note that this reduction can be negative which means the selected server increases the skewness.

If the destination server is found, then record the migration of the VM to particular server and improve the predicted load of related servers. Otherwise, move to the next VM in the list and try to find a destination server for it. As long as the VMs find their destination server, the run of the algorithm is successful and then move on to the next hot spot. Note that each run of the algorithm moves away at most one VM from the overloaded server. This does not necessarily remove the hot spot, but at least reduces its temperature. If it remains a hot spot in the next decision run, the algorithm repeats this process. It is possible to design the algorithm so that it can move away multiple VMs during each run. But this can add more load on the associated servers during a period when they are already overloaded. So it is decided to use this conservative approach and leave the system for some time to react before initiating additional migrations.

5.3 Role of Green Computing Algorithm

When the resource usage of active servers is too low, some of them can be turned off to save power. This is handled in Green computing algorithm[25]. The challenge here is to reduce the number of active servers during low load without compromising performance either now or in the future to avoid fluctuation in the system.

Green computing algorithm is invoked when the average usage of all resources on active servers is below the green computing threshold. The list of cold spots in the system is arranged based on the increasing order of their memory size. Since it is needed to move away all its VMs before shutting down an underutilized server, memory size of a cold spot is defined as the average memory size of all VMs running on it. Recall that model assumes all VMs plug-in to share back-end storage. Hence, the cost of a VM live migration is determined mostly by its memory footprint. It is tried to remove the cold spot with the lowest cost first.

For a cold spot p, check if it can be able to move all its VMs somewhere else. Each VM on p, tried to locate a destination server to accommodate. The resource usage of the server after accepting the VM must be below the warm threshold. So the power is saved by consolidating underutilized servers, overdoing it may create hot spots in the future. The warm threshold is designed to avoid that. If multiple servers satisfy the above criterion, then one of that is not a current cold spot. This is because increasing load on a cold spot minimizes the likelihood that it can be eliminated. However, if necessary choose a cold spot as the destination server. If all things are being equal, then choose the destination server whose skewness can be minimized the most by accepting this VM. In case of destination servers for all VMs on a cold spot, we record the sequence of migrations and improve the predicted load of related servers. Otherwise, do not move any of its VMs. The list of cold spots is also updated because some of them may no longer be cold due to the proposed VM migrations in the above process.

The above consolidation adds an extra load to the related servers. This is not a critical problem as in the hot spot mitigation case because green computing is initiated only when the load in the system is low. Nevertheless, needed to bound for extra load due to server consolidation. By restricting the number of cold spots that can be removed in each run of the algorithm, we find no more active servers in the system. This is called the consolidation limit.

The elimination of cold spots in the system is possible only when the average load of all active servers is below the green computing threshold. Otherwise, leave those cold spots there only as potential destination machines for future offloading.

5.4 Consolidated Movements

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

6. SIMULATION

We evaluate the performance of the skewness algorithm using Cloud Sim simulation. Our simulation uses the same code base for the algorithm as the real implementation in the experiments. This ensures the loyalty of our simulation results. Results are per-minute server resource usage, such as memory usage, CPU rate, and network traffic statistics, that

are collected using tools like "perfmon" (Windows), the "/proc" file system (Linux), "pmstat/vmstat/netstat" commands (Solaris), etc..

We also collected results from different servers and desktop computers. We post processed the results based on days collected and use arbitrary sampling and linear combination of the data sets to generate the workloads needed. All simulations in this section use the actual trace workloads unless otherwise specified.

Table 2. Parameters of Our Simulation

	Meaning	Value
h	Hot threshold	0.7
c	Cold threshold	0.1
w	Warm threshold	0.65
g	Green computing threshold	0.3
l	Consolidation limit	0.05

The default parameters we use in the simulation are shown in Table 2. In a dynamic system, these parameters represent good knobs to tune the performance of the system adaptively. We choose the default parameter values based on real experience working with many Internet applications.

6.1 Effect of Thresholds

We have evaluated the effect of the different thresholds used in our algorithm. We simulate a system with 10 PMs and 100 VMs. We use arbitrary VM to PM mapping in the initial layout. The scheduler is invoked once per minute. The bottom part of Figure 2 shows the resource usage in the system. The x axis is the time in minutes. The y-axis is overloaded with two meanings: the percentage of the resource or the percentage of active PMs in the system. Recall that a PM is active if it has at least one VM running. As can be seen from the Figure 2, the CPU load demonstrates patterns which are substantially fixed even after few minutes. The memory consumption is nicely balanced over the time. The network usage stays very low.

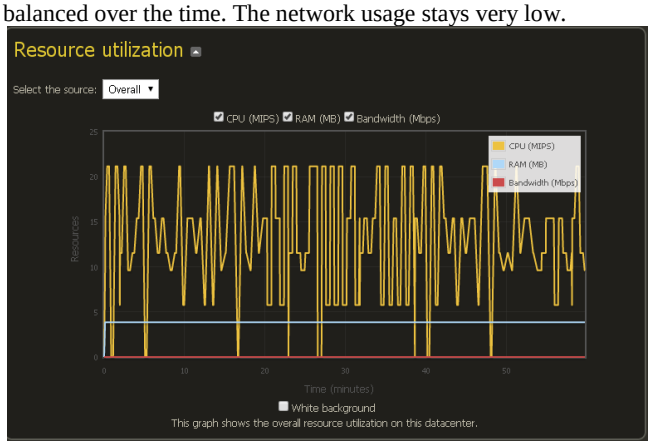

Figure 2. Impact of threshold

Figure 2 shows how the percentage of resource utilization differs with the load for different thresholds in our algorithm. For example, "h0.9 g0.4 c0.25" means that the hot, the green computing, and the cold thresholds are 90, 40, and 25 percent respectively. Parameters not had shown in the figure take the default values in Table 2. Our algorithm can be made more or less aggressive in its migration decision by modulating the thresholds. Figure 2 shows lower hot thresholds because more aggressive migrations to reduce hotspots in the system increase the number of APMs, and higher cold and green computing thresholds cause more dynamic consolidation. With the default thresholds in Table 2, the percentage of resources in the algorithm follows the load pattern closely.

To analyze the performance of the algorithm in more extreme situations, we also create a synthetic workload which imitates the shape of a sine function (only the positive part) and ranges from 15 to 95 percent with a 20 percent random fluctuation. It has a much larger peak-to-mean ratio than the actual result.

7. 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 varying requirements. We use the skewness metric to merge VMs with various resource characteristics appropriately so that the capacities of servers are well utilized. Both overload avoidance and green computing for systems with multi-resource constraints are achieved by the algorithm.

8. ACKNOWLEDGMENTS

The authors would like to thank S.J.M.Vidyapeetha, The Principal and staff of S.J.M.I.T.Chitradurga for their support.

9. 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 of Virtualization," 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. Technical Conf., 2005.

[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), Nov. 2007.

[8] 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.

[9] C.A. Waldspurger, "Memory Resource Management in VMware ESX Server," Proc. Symp. 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," Proc. USENIX Symp. 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 MultipleVirtualized Resources," Proc. ACM European conf. ComputerSystems (EuroSys '09), 2009.

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

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

[14] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P.Doyle, "Managing Energy and Server Resources in HostingCenters," Proc. ACM Symp. Operating System Principles (SOSP '01),Oct. 2001.

[15] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A ScalableApplication Placement Controller for Enterprise Data Centers,"Proc. Int'l World Wide Web Conf. (WWW '07), May 2007.

[16] 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.

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

[18] 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.

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

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

[21] Y. Toyoda, "A Simplified Algorithm for Obtaining ApproximateSolutions to Zero-One Programming Problems," ManagementScience, vol. 21, pp. 1417-1427, Aug. 1975.

[22] R. Nathuji and K. Schwan, "Virtualpower: Coordinated PowerManagement in Virtualized Enterprise Systems," Proc. ACMSIGOPS Symp. Operating Systems Principles (SOSP '07), 2007.

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

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

[25] T. Das, P. Padala, V.N. Padmanabhan, R. Ramjee, and K.G. Shin,"Litegreen: Saving Energy in Networked Desktops Using Virtualization,"Proc. USENIX Ann. Technical Conf., 2010.

[26] Y. Agarwal, S. Savage, and R. Gupta, "Sleepserver: A Software-Only Approach for Reducing the Energy Consumption of PCSwithin Enterprise Environments," Proc. USENIX Ann. Technical Conf., 2010.

[27] N. Bila, E.d. Lara, K. Joshi, H.A. Lagar-Cavilla, M. Hiltunen, andM. Satyanarayanan, "Jettison: Efficient Idle Desktop Consolidationwith Partial VM Migration," Proc. ACM European Conf.Computer Systems (EuroSys '12), 2012.

[28]Zhen Xiao, Senior Member, IEEE, Weijia Song, and Qi Chen "Dynamic Resource Allocation Using VirtualMachines for Cloud Computing Environment" IEEE Transactions On Parallel And Distributed Systems, June 2013.