

Utility Based Consolidation of Cloud Servers Using Memory Harnessing

S.Usmin, K.P.Vijayalakshmi, U.Muthaiah

Dept of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology,
Sankari, Salem (DT), India.

Dept of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology,
Sankari, Salem (DT), India.

Dept of Computer Science and Engineering, Sri Shanmugha College of Engineering and Technology,
Sankari, Salem (DT), India.

Abstract— Cloud Data Centers provides a range of solutions for systems deployment and operation. It is used to provide hosted applications for a third party to offer services to their customers by multiplexing server resources. Virtualization creates a virtual version of the resources by dividing the execution units into one or more execution units. In this approach, we present an idea that uses virtualization technology to allocate cloud data center resources dynamically based on application demands and upkeep green computing by augmenting the number of servers actively used. The resource allocation problem can be demonstrated as the bin packing problem where each server is considered as a bin and each virtual machine (VM) is the item to be packed in the server. Virtualization technology makes it easy to move running application across physical machines without any interruption. It abstract this as a variant of the relaxed classical on-line bin packing problem and develop a concrete, efficient algorithm that works well in an actual system according to Service Level Agreements. It adjust the resources available to each VM both within and across physical servers with memory de-duplication technologies. This can be used to adjust the VM layout for load balancing and energy saving purpose. Extensive simulation and experiment results demonstrate that our system achieves good performance compared to the existing work.

Keywords — Cloud Computing, Virtualization, Green Computing, Live Migration

INTRODUCTION

Cloud computing is a computing model that allows better management, higher utilization and reduced operating costs for datacenters while providing on demand resource provisioning for different customers. Data centers are often massive in size and complication.

In order to fully realize the cloud computing model, efficient cloud management software systems that can deal with the datacenter size and complexity need to be considered and built.

This paper studies programmed cloud elasticity management, one of the main and crucial datacenter management capabilities. Elasticity can be defined as the ability of cloud infrastructures to swiftly change the amount of resources allocated to an application in the cloud according to its claim.

Amazon EC2 provides set of connections of virtual machines (VM) for lease. This model, named as Infrastructure as a Service (IaaS), is broadly accepted in the industry. It is estimated that the number of daily VM launches in the “us-east-1” region has grown-up from 6,738 in year 2008 to 70,528 in year 2010. The total number of physical servers is close to half a million in 2012.

The load of applications encapsulated in the VMs may transform dynamically. The resource demands of applications can be highly changeable due to the time of the day effect, rapid surge in user requests, and other

reasons. Moreover, some applications such as on-demand data processing and virtual desktop have indiscriminate load. It is hard to forecast their resource demands. Although server consolidation has the effect of captivating load fluctuation, the sum of the peak resource demand of VMs sharing a physical machine (PM) can be much higher than the mean. Thus, it is often inefficient to over-provision resources based on the peak demand because doing so will leave the resources under-utilized most of the time. Live migration consent to a VM to be moved from one server to another without disrupting the application running inside. This can be used to adjust the VM layout for load balancing or energy saving purpose. For example, when the resource utilization of a server becomes too high, some VMs running on it can be migrated away to reduce its load. On the contrary, when the average server utilization in the system turn out to be too low, the VMs can be aggregated to a portion of the servers so that idle servers can be put to sleep to save power. Previous work has shown that energy is a noteworthy part of the operational cost in data centers and that it takes a server only several seconds to sleep or to wake up. Novel hardware assisted servers can bring about the transition between sleep and active states in negligible milliseconds. Compared with application layer solutions, this approach enjoys the simplicity of being application neutral. However, live migration deserves overhead and may last some period of time depending on the workload inside the VM and the coexisting network traffic.

Various migration policies have been designed in the literature, a server whose resource utilization is above some pre-defined thresholds is called a “hot spot”. Its black box/gray box (BG) scheduling algorithm periodically detects hot spots in the system and decides which VMs should be migrated away to relieve the hot spots. It does not consolidate VMs at valley load for green computing purpose, since its main focus is overload avoidance. Nor does VectorDot perform green computing. The resource allocation difficulty can be modeled as the bin packing problem where each server is a bin and each VM is an item to be packed. The approach on hand in periodically performs an offline bin packing algorithm to determine a new VM layout, where hot spots are eradicated and the number of servers in use is minimized. VMs are then migrated accordingly. The new layout, however, may be quite diverse from the existing one, which may acquire a large number of VM migrations. Wang et al. approves an online bin packing algorithm which incurs fewer migrations. The focus of that work is on the network bandwidth only and does not deem multi-dimensional resource constraints. In this paper, we present an approach that exercises live migration to allocate data center resources dynamically based on application demands and supports green computing by optimizing the number of servers used. We make the following contributions:

- We develop a practical bin packing resource allocation algorithm. It is capable of both surplus avoidance and green computing. It is appropriate to scenarios with multi-dimensional resource constraints.

- We give a theoretical proof that the number of servers in use is bounded by $3/2 * d$, where d is the number of restricted access of resources. Especially, with only one bottleneck resource, the number of servers in use is delimited by 1.5 times of the optimal value. We also prove that the number of VM migrations is bounded as well.
- We conduct extensive simulation and experiments. The results exhibit good performance in contrast with the existing work.

II. OVERVIEW

Figure 1 gives an outline of a typical virtualized data center where our approach applies. The components of our approach are manifested with gray color. Each physical machine (PM) runs a Virtual Machine Monitor (VMM) like Xen, KVM or VMware. Several virtual machines (VM) share a PM. Applications such as Web server, remote desktop, DNS, Mail server, Map/Reduce, etc., run inside the VMs. The VMM collects the usage statistics of resources on corresponding PM such as CPU utilization, memory utilization, network sends and receives, etc. These statistics are forwarded to a centralized VM scheduler, the key component that is accountable for load balance and green computing by adjusting the mapping of VMs to PMs.

The VM Scheduler is invoked every so often and receives the following inputs:

- the resource demand history of VMs
- the capacity and the load history of PMs
- the current assignment of VMs to PMs

The VM Scheduler has two modules. The load predictor approximates the resource demands in the near expectations. The scheduling algorithm optimizes VM layout so that the resource demands are satisfied and resource desecrate is minimized. Although the VM scheduler may seem to be a single tip of failure, it is not involved in the data path during normal processing. In other words, its failure will cause the system to end adjusting the VM to PM layout in response to load changes in the system, but the existing layout can continue functioning on average. Should a higher degree of reliability be needed, we can use the hot mirroring technology or the fast restart technology. The VM Scheduler is invoked periodically. In every round, the load predictor estimates the resource demands of all applications (VMs) and the load of all PMs based on precedent statistics. The estimates are symbolized by the number of CPU ticks, the amount of memory, and the bandwidth of network I/O. Based on the estimates, the scheduling algorithm checks whether every PM has sufficient resources for the appointed demands. If so, the VMM is able to adjust resource allocation locally.

Otherwise, the algorithm achieves overload avoidance by migrating away some of the VMs on the overloaded PMs. The scheduling algorithm also consolidates VMs on the under-utilized PMs so that some of the PMs become unused and can be set into the standby mode to save

energy.

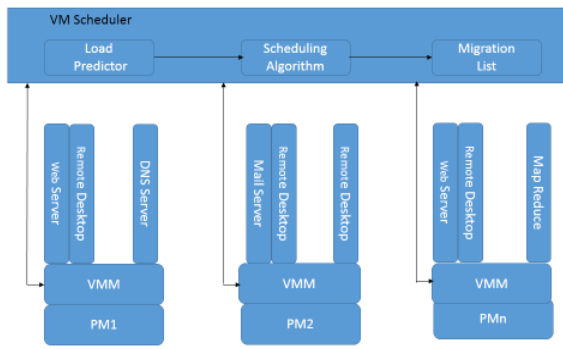


Fig. 1 Overview of the System Architecture

The yield of the scheduling algorithm is a VM migration list which is then sent off to the VMM on the involved PMs for execution. We suppose that all PMs share a back-end storage. Using live migration as a tool for resource allocation seems challenging at first glance. But this classical architecture is broadly adopted in the literature. It is also demonstrated that live migration incurs only insignificant overhead on applications. We will show that it is realistic to use live migration in our system.

We use a variant of the exponentially weighted moving average (EWMA) load predictor. The ESX Server uses traditional EWMA formula like the following:

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

Where $E(t)$ and $O(t)$ are the estimated and the observed load at time t , respectively. α reflects a tradeoff between stability and responsiveness. ESX calculates three EWMA estimates with different parameters and takes the maximum of them as a conventional estimate on the VM's memory needs. Although seemingly acceptable, it does not capture the rising trends of resource usage. For example, when we see a sequence of $O(t) = 10, 20, 30$, and 40 , it is realistic to predict the next value to be 50 . Unfortunately, when α is between 0 and 1 , the predicted value is always between the historic value and the observed one. To reflect the "acceleration", we take a novel approach by setting α to a negative value. When $-1 \leq \alpha \leq 0$, the above formula can be transformed into the following:

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

On the other hand, when the observed resource usage is going down, we want to be conventional in reducing our estimate. Hence, we use two parameters, t_a and t_d , to control how swiftly $E(t)$ adapts to changes when $O(t)$ is rising and decreasing, respectively. We call this the FUSD (Fast Up and Slow Down) algorithm.

So far we take $O(t)$ as the last observed value. Most applications have their SLAs specified in terms of a certain percentiles of requests meeting a explicit performance level. More generally, we keep a window of

W recently observed values and take $O(t)$ as a high percentile of them. As we will see later in the paper, the prediction algorithm plays an important part in improving the stability and performance of our resource allocation resolution.

III. RESOURCE ALLOCATION AS BINPACKING

The classical bin packing problem consists of packing a sequence of items with sizes in the interval $[0, 1]$ into a minimum number of bins with power one. We can model resource allocation as the bin packing difficulty where each PM is a bin and each VM is an item to be packed. We assume that all PMs are standardized with unit capacity. We regularize the resource demands of VMs to be a fraction of that capacity. For example, if a VM requires 20% of the physical memory of the underlying PM, then it corresponds to an item with size 0.2 . It is well-known that the problem is NP-hard. The quality of a polynomial time approximation algorithm A is calculated by its rough calculation ratio $R(A)$ to the finest algorithm OPT:

$$R(A) = \lim_{n \rightarrow \infty} \frac{\sup A(L)}{OPT(L)} \quad (1)$$

where L is the list of the input sequence and $A(L)$ and $OPT(L)$ are the number of bins used under the A algorithm and the finest algorithm, respectively.

Although the bin packing problem has been studied extensively in the literature, the existing solutions do not work well in data center environments. Offline algorithms can achieve an approximation ratio very close to one. This obviously leads to an approach that periodically invokes an offline algorithm to adjust the VM layout. The drawback of this approach is that it may acquire a large number of VM movements when the load of VMs changes dynamically because an offline algorithm by its nature does not consider the existing VM layout when packing the VMs into the set of PMs.

There are also online bin packing algorithms which pack the present item without knowledge of subsequent items. Strict online algorithms do not allow moving any previously packed item and have a theoretical lower bound of 1.536 for the approximation ratio. This is overly restrictive in our situation since virtualization technology enables VM migrations in real time. As we will see later, doing so allows us to achieve a better approximation ratio even though we do not effort to pack the bins almost as full.

Close to our work are the existing relaxed online algorithms which allow a constant number of movements of already packed items in response to the arrival of a new item. Unfortunately, those algorithms are not applicable to our settings either, because they do not support the size changes of already packed items. Note that the resource demands of VMs can modify over time (which motivated us to multiplex data center resources in the first place), the sizes of items in our bin packing problem are not fixed. One might think that we can handle the size change of a previously packed item by removing it from the bin and repack it, since item

removal can be supported by the delete operation in dynamic bin packing algorithms. Unfortunately, it is easy to construct counterexamples where the correct approach is to repack some other items in that bin instead of the changed item. In other words, when the resource demand of a VM changes, we may decide to migrate some other VMs giving out the same PM instead of migrating the changed VM. One solution is to repack all items in the changed bin, but doing so causes too many movements and defeats the purpose of an online algorithm. Moreover, many existing algorithms work by maintaining certain properties of the used bins (to keep them sufficiently full in order to prove the approximation bound). Eliminating an item (or reducing its size) can break those properties, leading to the reshuffle of many other items (including those from the unchanged bins). In short, the size change of an item cannot be accommodated in the existing algorithms easily.

In order to handle the changing resource demand of a VM, we enhance a relaxed online bin packing algorithm called Enhanced Variable Item Size Bin Packing (EVISBP). It features carefully constructed categories of items and bins. Moderate size change can be absorbed as long as the category rules are kept. It is vital to realize that our design principle is different in the face of the highly variable resource demands of data center applications. While the classical bin packing algorithms (online or not) consider packing a bin completely occupied a success, in data center environments keeping servers running at 100% utilization is detrimental to the stability of the system.

IV. EVISBP

The key idea of EVISBP is to trade the approximation ratio for system constancy. We divide items into four types according to their sizes. We restrict the mixture of item types in a bin to reduce the amount of space worn out.

- insert(item): put item into some bin
- change(old item, item): an already filled old item has become a new item (differ only in size)

The EVISBP is relaxed online in that (1) no knowledge of the future is assumed when managing the current event; (2) a little number of movements of already packed items are allowed. It differs from the existing (relaxed) online algorithms in the context of the change operation. The insert operation is similar to offline algorithm. In the following, we present an algorithm with an approximation ratio of 1.5 which confines the number of movements for insert to at most 3 and for change to at most 7. We begin with some definitions.

A. Definitions

1) Definition 1:

- Let b be a bin. $\text{gap}(b)$ indicate the space available in bin b .
- Let i be an item. $\text{size}(i)$ indicate the size of the item (always $\in (0,1]$).
- Let g be a group. $\text{size}(g)$ indicate the size of the group, which is the total size of all items in that group.

- Let x be an item or a group. $\text{bin}(x)$ indicate the bin it is in.

2) Definition 2: We divide items into some types based on their sizes.

- T-item (tiny): size $\in (0,1/3)$
- S-item (small): size $\in (1/3,1/2)$
- L-item (large): size $\in (1/2,2/3)$
- B-item (big): size $\in (2/3,1)$

The above classification of items help the construction of a bin whose gap is less than $1/3$ such as a bin with two T items or one B-item. The EVISBP algorithm attempts to maintain this property for all bins so that its approximation ratio is bounded.

3) Definition 3: We divide bins into several types based on their content.

- B-bin: has only one B-item.
- L-bin: has only one L-item.
- LT-bin: has only one L-item and a certain number of T-items.
- S-bin: has only one S-item.
- SS-bin: has only two S-items.
- LS-bin: has only one S-item and one L-item.
- T-bin: has only a certain number of T-items.

It is called unoccupied if the accessible space is no less than $1/3$. Otherwise, it is called filled.

4) Definition 4: We group a certain number of T-items in the same bin into a set. All T-items in a bin form some non-overlapping groups such that: 1) the size of any group is no more than $1/3$, and 2) the size of any two groups is larger than $1/3$. This grouping is convenient in that those tiny items in a group can be considered as a whole and moved in a single step. It is also realistic since the overhead of VM migration depends mostly on the size of its memory footprint. (Assume common network storage.) Hence, migrating a large VM can have a comparable overhead to the migration of several small ones.

5) Some notations: the item and bin types in the following are examples only. Other types are inferred in a similar way.

- $x \in \text{T-item}$: x is a T-item.
- $x \in b$: x is an item in bin b .
- $b \in \text{S-bin}$: b is an S-bin.
- $g \in b$: g is a group in bin b .
- $x \in (\text{S-item}, b)$: x is an S-item in bin b .
- UT-bin: unfilled T-bin.
- ULLT-bin: L-bin or LT-bin and the available space is no less than $1/3$ (i.e., $\text{gap} > 1/3$).

B. Algorithm

We are now prepared to build the insert and change operations. The two operations must remain the gap of most bins within $1/3$.

insert (item)

if item $\in$ B-item then new(item);

if item $\in$ L-item then $b = \text{new}(\text{item})$; fill(b);

if item $\in$ S- item then insert_S- item(item);

if item $\in$ T-item then fillwith(item);

Now we describe the change operation which is the core of our algorithm. We use $X \rightarrow Y$ to denote the types of the item before and after the change as X and Y , respectively. For example, $B \rightarrow S$ means the item changes from a B-item to an S-item. Not all type changes requires adjustment. On the other hand, adjustment may be needed even if the type does not change. In the following, any condition “ $\exists X\text{-bin}$ ” does not consider the bin where the change is happening.

C. Analysis on the number of movements

Based on the sort of the item before and after the size transform, we list the number of movements in the worst case in Table 1. The analysis is related to the previous theorem.

TABLE 1
Number of Movements in Worst Case

Old\New	B-item	L-item	S-item	T-item
B-item	0	3	1	1
L-item	2	3	3	3
S-item	1	4	0	2
T-item	5	7	6	5

V. SIMULATIONS

A. Existing algorithms for comparison

In this section, we contrast EVISBP with three algorithms using outline driven simulation: the Black-box and Gray-box algorithm, the Vector Dot algorithm, and the offline bin packing algorithm.

The Black-box and Gray-box algorithm (denoted as BG) unites utilization of CPU, memory, and network I/O bandwidth into a single “volume” metric, which enumerates the resource demands of a VM or the load of a PM. When solving a hot spot, the BG algorithm attempts to migrate away a VM with the highest volume to memory size ratio to a PM with the lowest degree. This maximizes the volume brought away by every migrated byte.

In HARMONY, PMs are connected by data center network to a central storage management device. When a PM or a switch in the data center network becomes a hot spot, the VM that contributes the most to the hot spot is moved elsewhere. VectorDot introduces an “EVP” metric, which is calculated by the dot product of the consumption vector of a VM and that of a PM hot spot, to quantify the contribution of the VM to the hot spot. It also helps make a decision the migration destination. This simulation does not involve data scheduling.

For the offline bin packing algorithm (denoted as offline-BP), during each scheduling interval, all VMs are sorted by the resource demand in downward order. The First-Fit heuristic is then used to compute an approximate solution.

B. Design of the simulator

We estimate the effectiveness of EVISBP using trace driven simulation. We use the framework from an open source cloud management software called CloudSim. Our simulation uses the same code base for the algorithm as the

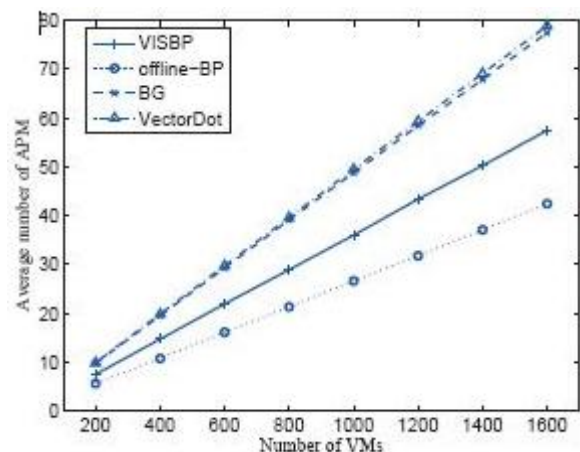
real implementation. This ensures the reliability of our simulation results.

We use random sampling and linear combination of the datasets to produce the VM traces. Our experiment demonstrates the efficacy of local resource adjustment. We run four VMs on a server. Each VM runs a copy of the server side. To simulate rising and declining of the load, we prepare four successive waves of hybrid workload including both the uploading and downloading of files at different throughput levels.

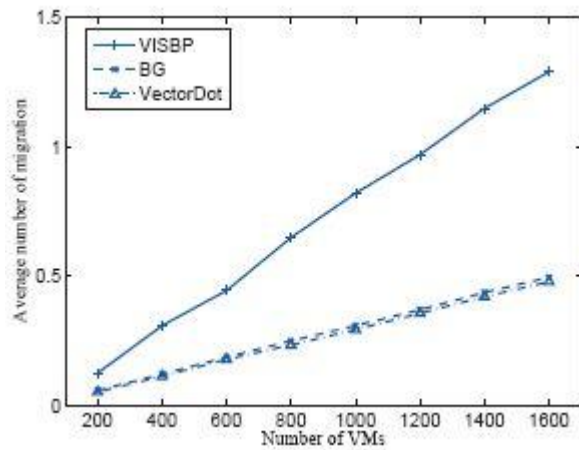
C. Scalability

We evaluate the scalability of the algorithms by escalating the number of VMs in the simulation from 200 to 1600 while keeping the VM to PM ratio fixed at 20:1 as said before. Figure 5 (a) shows that, for all four algorithms, the average number of APMs raises linearly with the system size. Our EVISBP algorithm uses about 73% of total PMs, much smaller number than 98% of the BG algorithm and 99% of the VectorDot algorithm. This demonstrates that our EVISBP algorithm achieves good green computing effect. Although the offline BP algorithm uses 19% fewer PMs than ours, as exposed later in this section, it incurs too many migrations to be practical.

We also calculate the total number of migrations at varying scale. As shown in Figure 5 (b), for all algorithms, the number of migrations increases approximately linearly with the system size. In our EVISBP algorithm, each VM on average experiences 8×10^{-4} migrations during each decision run, independent of the system size. This interprets into roughly one migration per thousand decision intervals. The constancy of our algorithm is good enough. Figure 5 (b) shows that the number of migrations of offline-BP algorithm is much more than that of ours.



5 (a) NUMBER OF APM



5 (b) NUMBERS OF MIGRATIONS

VI. RELATED WORKS

A. Resource Allocation

The problem of adaptive resource allocation in the cloud environment has been examined in the literature. To achieve application level quality of service (QoS), Padala models application performance as a linear function of allocations of CPU time and disk I/O so that the resource allocator on each physical server can smartly regulate the allocation to get expected performance. In Seawall, a centralized controller forecasts congestion in data center network from traffic among application components. It carefully throttles the network I/O of involved virtual machines to optimize network resource utilization. The above approaches spotlight on fairness in resource allocation across the applications. Our work, in contrast, focuses on hot spot avoidance and green computing. EVISBP replicates web applications to meet their resource requirement. It allows web applications to offer for server resources. A resource allocation algorithm chooses which applications are served to maximize profit by minimizing the number of profitable active servers. Hot spots are evaded since the algorithm prohibits over-commitment. However, this algorithm assumes that the memory footprint of applications is small enough that a single server can host all applications in a data center, which is no more appropriate nowadays. In modern systems, memory is treated as the bottleneck resource. A server can only host an inadequate number of applications and the replacement of applications is an expensive operation. As such, they have a strong motivation to minimize application placement changes. In our previous work, we use a CCBP model to present automatic scaling for web applications in the cloud environment. It aims at good demand satisfaction ratio, minimal replacement, and power consumption. These systems do not use virtualization and apply only to explicit applications. In contrast, we treat each VM as a black box and have no knowledge of encapsulated applications.

MapReduce is also a popular framework that supports the resource allocation in the cluster. However, it often runs in higher point than our system. A lot of research has run MapReduce as an application in the cloud, e.g. Amazon EC2 and MCP. Therefore, our EVISBP approach

can still be used to adjust the resource allocation effectively in these cases.

Similar to our work, Sandpiper and HARMO exploit live migration to offload hot spots. However, they do not support green computing. Some approach call upon classical offline bin packing algorithm periodically to optimize resource allocation. Although the offline algorithm does well in green computing, the number of migrations it incurs is virtually prohibitive. Interestingly, Rahman et al. tries to prevent unnecessary migration by solving contention locally.

B. Green Computing

There are many approaches to green computing in data centers. One sort leverages hardware technologies to improve power efficiency of each server. Another adopts low power agencies to minimize idle power consumption.

Close to our approach, a bunch of resource scheduling systems try to minimize the number of active physical servers. Idle servers may be put into sleep state or even shutdown to save power. This approach is known as server consolidation. As shown in VirtualPower, server consolidation accounts for the popular of power saving effect. In addition, this approach does not rely on any special hardware or specific settings like *Sleep Server*. VMAP incorporates the idea that resource utilization of servers in better cooled areas of a data center can be higher than that of the others for improved power effectiveness. This idea can be espoused in our future work.

VII. CONCLUSIONS

We have presented the intend, implementation, and evaluation of a practical online bin packing algorithm called the EVISBP which can allocate data center resources dynamically through live VM migration based on SLA. Compared to the existing algorithm, the EVISBP stand out in hotspots mitigation and load balance. It also bring together the demands on both green computing and stability.

In the future, we plan to incorporate application performance model and allocating resource when there is heavy traffic in the network. In addition, we sketch to contrast the proposed system with other approaches such as linear programming and examine the impact of cooling system to reduce the heat dissipated.

REFERENCES

- [1] G. Rosen. (2010, Dec.) Recounting ec2 one year later. [Online]. Available: <http://www.jackofallclouds.com/2010/12/recounting-ec2/>
- [2] C.Tang (2012, Mar.) Amazon data center size. [Online]. Available: <http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/>
- [3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in Proc. of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05), 2005.
- [4] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in Proc. of the 18th ACM Symposium on Operating System Principles (SOSP'01), 2001.
- [5] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: augmenting network interfaces to reduce PC energy usage," in Proc. of the 6th USENIX symposium on Networked systems design and implementation (NSDI'09), 2009.

- [6] Y. Agarwal, S. Savage, and R. Gupta, "Sleepserver: a software-only approach for reducing the energy consumption of PCs within enterprise environments," in Proceedings of the USENIX conference on USENIX annual technical conference, 2010.
- [7] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in Proc. of the 14th international conference on Architectural support for programming languages and operating systems (ASPLOS'09), 2009.
- [8] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in Proc. of the 16th International World Wide Web Conference (WWW'07), 2007.
- [9] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and Gray-box strategies for virtual machine migration," in Proc. of the 4th Symposium on Networked Systems Design and Implementation (NSDI'07), 2007.
- [10] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in Proc. of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07), 2007.
- [11] A. Singh, M. Korupolu, and D. Mohapatra, "Server-storage virtualization: integration and load balancing in data centers," in Proc. of the 20th ACM/IEEE conference on Supercomputing (SC'08), 2008.
- [12] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in Proc. of 30th IEEE International Conference on Computer Communications (INFOCOM'11), 2011.
- [13] H. Xu and B. Li, "Egalitarian stable matching for VM migration in cloud computing," in Proc. of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2011.
- [14] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: coordinated multi-level power management for the data center," in Proceedings of the 13th international conference on Architectural support for programming languages and operating systems, 2008.
- [15] S.-H. Lim, S.-J. Huh, Y. Kim, and C. R. Das, "Migration, assignment, and scheduling of jobs in virtualized environment," in proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, 2011.