Request and Server Consolidation Methods for Cloud Power Management

C.Senthilkumar[a], Mr. A. Rajesh[b], Mr. P. Natarajan[c]

[a,b,c]*Assistant Professor, Department of Computer Applications,*
Erode Sengunthar Engineering College,
Thudupathi, Perundurai, Erode-638 057, India

Abstract

Cloud computing is used to access computing resources owned and operated by a third-party provider. Cloud computing is Internet-based computing to share resources, software and information. Both transactional and long-running analytic computations are comprised into workloads. Scientific simulations to multi-tier transactional applications are referred as workloads. Power management strategies have been proposed for enterprise servers based on Dynamic Voltage And Frequency Scaling (DVFS). DVFS allows the server to transition the processor from high-power states to low-power states. The processors are assigned to sleep states such as deep sleep to reduce energy consumption. In deep sleep the server can be configured to use Direct Memory Access (DMA) to place incoming packets into memory buffers for processing in the active state.

Request batching can be conducted to group received requests into batches and put the processor into sleep between the batches. Virtual Batching is a request batching solution for virtualized servers with primarily light workloads. The system dynamically allocates CPU resources with same performance level and peak values. Server consolidation is performed to fully utilize a small number of active servers in the data center. Static and dynamic server consolidation algorithms are used to assign data centers to the request batches. Static server consolidation algorithm is used for the offline mode in data centers. Online workload variations are managed by the dynamic server consolidation algorithms. Virtual batching is integrated with pMapper (power-aware application placement framework) to assign data centers for the workloads.

The Virtual Batching scheme is enhanced to manage resources with load balancing mechanism. The system is improved with optimization mechanism to manage relative response time. Resource levels and application requirements are integrated in the allocation process. The system is adopted to support Dynamic Random Access Memory(DRAM) and Dual in-line Memory Module(DIMM) components.

Keywords: *DVFS, Virtual Batching, Request Batching, Server ConsolidationCPU resource allocation.*

1. Introduction

Cloud computing is becoming one of the next IT industry buzz words: users move out their data and applications to the remote "Cloud" and then access them in a simple and pervasive way. This is again a central processing use case. Similar scenario occurred around 50 years ago: a time-sharing computing server served multiple users. Until 20 years ago when personal computers came to us, data and programs were mostly located in local resources. Certainly currently the Cloud computing paradigm is not a recurrence of the history. 50 years ago we had to adopt the time-sharing servers due to limited computing resources. Nowadays the Cloud computing comes into fashion due to the need to build complex IT infrastructures. Users have to manage various software installations, configuration and updates. Computing resources and other hardware are prone to be outdated very soon. Therefore outsourcing computing platforms is a smart solution for users to handle complex IT infrastructures.

At the current stage, the Cloud computing is still evolving and there exists no widely accepted definition. Based on our experience, we propose an early definition of Cloud computing is follows. A computing Cloud is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which could be accessed in a simple and pervasive way.

Conceptually, users acquire computing platforms or IT infrastructures from computing Clouds and then run their applications inside. Therefore, computing Clouds render users with services to access hardware, software and data resources, thereafter an integrated computing platform as a service.

Hardware as a Service was coined possibly in 2006. As the result of rapid advances in hardware virtualization, IT automation and usage metering & pricing, users could buy IT hardware, or even an entire data center, as a pay-as-you-go subscription service. The HaaS is flexible, scalable and manageable to meet your needs. Examples could be found at Amazon EC_2, IBM's Blue Cloud project, Nimbus, Eucalyptus and Enomalism.

Software or an application is hosted as a service and provided to customers across the Internet. This mode eliminates the need to install and run the application on the customer's local computers. SaaS therefore alleviates the customer's burden of software maintenance, and reduces the expense of software purchases by on-demand pricing. An early example of the SaaS is the Application Service Provider
(ASP). The ASP approach provides subscriptions to software that is hosted or delivered over the Internet. Microsoft's "Software + Service" shows another example: a combination of local software and Internet services interacting with one another. Google's Chrome browser gives an interesting SaaS scenario: a new desktop could be offered, through which applications can be delivered in addition to the traditional Web browsing experience.

Data in various formats and from multiple sources could be accessed via services by users on the network. Users could, for example, manipulate the remote data just like operate on a local disk or access the data in a semantic way in the Internet. Amazon Simple Storage Service (S_3) provides a simple Web services interface that can be used to store and retrieve, declared by Amazon, any amount of data, at any time, from anywhere on the Web. The DaaS could also be found at some popular IT services, e.g., Google Docs and Adobe Buzzword. Elastic Drive is a distributed remote storage application which allows users to mount a remote storage resource such as Amazon S_3 as a local storage device.

Based on the support of the HaaS, SaaS and DaaS, the Cloud computing in addition can deliver the Infrastructure as a Service (IaaS) for users. Users thus can on-demand subscribe to their favorite computing infrastructures with requirements of hardware configuration, software installation and data access demands. The Google App Engine is an interesting example of the IaaS. The Google App Engine enables users to build Web applications with Google's APIs and SDKs across the same scalable systems, which power the Google applications.

2. Related Work

Virtualization technology has provided a promising way to manage application performance by dynamically reallocating resources to VMs. Several management algorithms have been proposed to control application performance for virtualized applications [3]. For example, Padala et al. [3] used MIMO control theory to control application performance for virtualized servers. In contrast, our solution provides energy savings through DVFS and request batching.

Energy conservation has been one of the most important design constraints for Internet servers. The majority of the prior work has attempted to reduce power consumption by reducing the energy consumption of individual server components. Several algorithms have been proposed to utilize DVFS to manage the energy consumption at the server level, the cluster level [1] and the data center level [4]. Recently, several research projects have addressed power or energy problems in platforms employing virtualization technology by using DVFS techniques. However, these algorithms cannot further reduce energy consumption when the processor is already at the lowest DVFS level.

Elnozahy et al. proposed a request batching technique for nonvirtualized web servers by putting the processor into the sleep mode when the server is idle, significantly reducing the energy consumption of web servers when the workload is light. In contrast, our solution provides additional energy savings by allowing the processor to sleep even when the server is not completely idle. Further, we provide a performance balancing controller to dynamically adjust the CPU resource allocation to the VMs such that request batching can be utilized in virtualized environments. VM migration is an important tool for resource and power management in virtualized computing environments. Several

recent studies propose to dynamically consolidate VMs to a smaller number of servers and putting the unused servers into the sleep mode for energy savings [5]. In contrast to these algorithms, our solution has several advantages. First, the request batching technique used in our solution has a much less overhead than VM migration, thus can be used on a much smaller time scale. Second, due to resource limitations, such as the memory and bandwidth, sometimes it is infeasible to further consolidate VMs to a smaller number of servers even if the utilization of the processor is still low. In this situation, request batching can save additional energy by putting the processor to the sleep mode periodically. Our algorithm is integrated with VM consolidation algorithms to save power in different situations.

Several papers have explored using low power states of server components. AbouGhazaleh and Choi et al. provided studies based on DVFS. Horvath and Skadron proposed using dynamic cluster configuration and multiple ACPI sleep states to reduce energy consumption in multitier server clusters. Zhu et al. [8] and Nathuji et al. [1] proposed to dynamically allocate CPU resources to the operating system or VM. None of these studies, however, consider using a batching technique.

3. Resource Scheduling in Clouds

The need to provide a guaranteed level of service performance is important for data centers. This is largely due to a business model driven by strict service level agreements (SLAs) based on metrics such as response time, throughput, and reserve capacity. However, energy demands and associated costs are increasing at an alarming rate; it is projected that data centers in the US alone will consume 100 billion kWh of energy at a cost of 7.4 billion dollars per year by 2011. This poses a dilemma for data center operators; they must satisfy new and existing service contracts while minimizing energy consumption to reduce cost and strain on power generation facilities.

Data centers generally provision based on a worst-case scenario, which leads to a low-average server utilization in modern data centers. For example, a recent estimation suggests that the utilizations of web servers are often in the 5 to 12 percent range. These underutilized servers spend a large portion of their time in an idle state [7]. Several recent studies have shown that a server uses approximately 60 percent of its required peak power when it is idle. This over provisioning leads to large amounts of energy waste. Therefore, reducing energy waste, while guaranteeing SLA agreements, can lead to significantly reduced operating costs.

A well-known approach to addressing this problem is to transition the processor from high-power states to low power states using Dynamic Voltage and Frequency Scaling (DVFS) whenever the performance allows. This approach effectively reduces the power consumption of the computer systems when the server has a medium intensity workload (we define workload intensity in Section 2). However, the capability of this approach to reduce power consumption is limited when the server has a low-intensity workload due to two reasons. First, when the utilization of the processor is very low, the leakage power, which cannot be significantly reduced by DVFS, contributes a major portion of the power consumption. Second, many high performance processors only allow a small range of DVFS levels and even the lowest level provides a higher speed than is required for some light workloads. For example, in a case study on our testbed, the power consumption of an idle server with an Intel Xeon 5360 processor can only be reduced from 163 to 158 W when the processor is transitioned from the highest DVFS level to the lowest one.

To further reduce energy consumption, processors need to be put into sleep states such as Deep Sleep. In Deep Sleep, the processor is paused and consumes significantly less power. For example, the power consumption of a server with an Intel Xeon 5500 Processor may be reduced to 23 percent of its peak value when the processor is switched to the Deep Sleep state [6]. When the processor is in Deep Sleep, the server can be configured to use Direct Memory Access (DMA) to place incoming packets into memory buffers for processing when the processor is returned to the active state, thus, avoiding harming the functionality of the hosted server applications. Therefore, to save more power for servers with light workloads, we can perform request batching to put the processor into the Deep Sleep state when there are few incoming requests. During the sleep time, we delay and batch the requests when they arrive and wake the processor up when the earliest request in the batch has been kept pending for a certain batching time-out.

However, it is challenging to perform request batching directly on a virtualized

server. Virtualization technologies such as Xen, VMware, and Microsoft Hyper-V allow provisioning multiple virtual machines (VMs) onto a single physical server. However, all the VMs on a single physical server are correlated due to sharing the same physical hardware, i.e., any state changes in the hardware affect all the VMs. Since different VMs may have different workloads and performance requirements, putting the processor into Deep Sleep based on the performance of one VM may affect the application performance of other VMs.

In this paper, we propose Virtual Batching, a novel request batching solution for virtualized enterprise servers with primarily light workloads. Our solution dynamically allocates the CPU resource such that all the VMs can have approximately the same performance level relative to their allowed peak values. Based on the uniform level, our solution then determines the time length for periodically batching incoming requests and putting the processor into sleep. When the workload intensity changes from light to medium, request batching is automatically switched to DVFS to increase processor frequency for performance guarantees.

Virtual Batching is also extended to integrate with server consolidation to achieve maximized energy conservation with performance guarantees for virtualized data centers. Server consolidation can improve server utilization by consolidating VMs onto a smaller number of servers on a long time scale. However, due to conservative resource profiling and various real-world constraints, servers after consolidation can still be underutilized. Virtual Batching can then be adopted to put the processors of active servers into sleep on a shorter time scale for further energy savings due to its much smaller overhead. Specifically, this paper has the following contributions:

- We propose a novel request batching technique in virtualized environments to achieve significant energy conservation when the server workload is light.
- We integrate request batching and DVFS to provide energy conservation for virtualized servers when the workload varies at runtime. Our solution allows energy to be saved across a wide range of workload intensities.
- We design a two-layer control architecture that relies on feedback control theory as a theoretical foundation to achieve analytical assurance of control accuracy and system stability.
- We propose to integrate request batching with server consolidation to achieve maximized energy conservation with performance guarantees for virtualized data centers.
- We conduct experiments on a hardware testbed with real trace files and present empirical results to demonstrate the effectiveness of our control solution to conserve energy for virtualized enterprise servers.

4. Problem Statement

Request batching can be conducted to group received requests into batches and put the processor into sleep between the batches. Virtual Batching is a request batching solution for virtualized servers with primarily light workloads. The system dynamically allocates CPU resources with same performance level and peak values. Server consolidation is performed to fully utilize a small number of active servers in the data center. Static and dynamic server consolidation algorithms are used to assign data centers to the request batches. Static server consolidation algorithm is used for the offline mode in data centers. Online workload variations are managed by the dynamic server consolidation algorithms. Virtual batching is integrated with pMapper (power-aware application placement framework) to assign data centers for the workloads. The following drawbacks are identified in the existing system.

- Complex virtual server sleep process
- Average relative response time is not optimized
- Energy management is tuned for the processor
- Data center load is not managed

5. Architecture of Virtual Batching Scheme

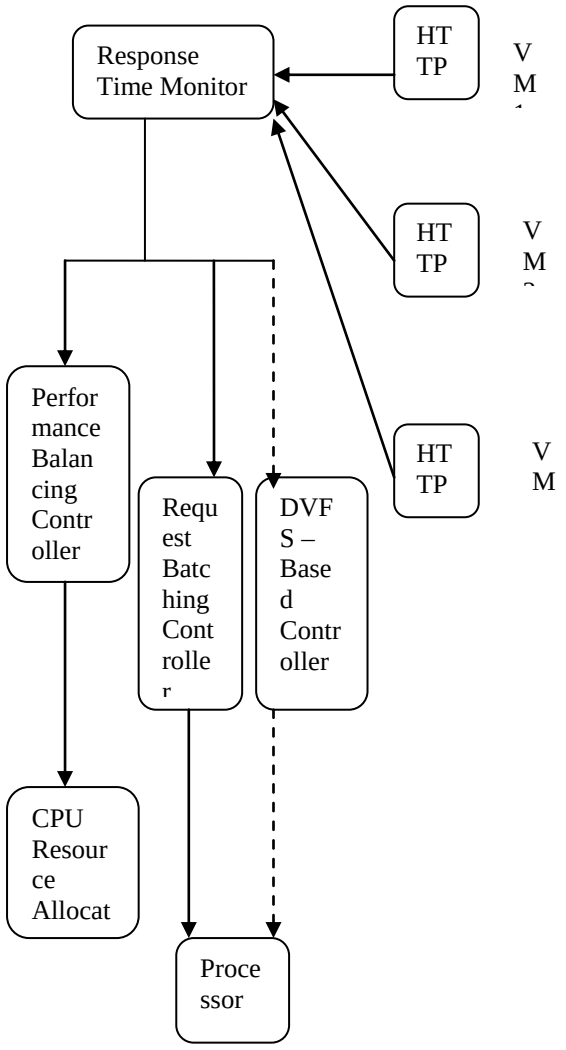

Fig. 5.1. System architecture of Virtual Batching.

In this section, we provide a high-level description of the Virtual Batching system architecture. The goal of Virtual Batching is to control the response times of all the VMs on a physical server to their administrator-defined set points while minimizing the energy consumption of the server. To achieve this goal, we adopt three techniques: Request batching, DVFS, and CPU resource allocation. Fig. 5.1 provides an overview of the system architecture. In this paper, we assume that every VM runs a web application and the applications in different VMs are independent of each other. We also assume that the web requests to different VMs are independent of each other. These assumptions are usually valid because the VMs can belong to different customers.

5.1 Performance Balancing

Implementing a technique such as request batching or DVFS directly in a virtualized environment is challenging because both methods rely on modifying the performance of the physical system (via CPU hardware states) to manage energy consumption. Because hardware state transitions affect the performance of all VMs hosted on a physical server, the VMs are correlated. Without effective resource management, the methods of controlling the response times of enterprise servers inside the VMs are limited; using the average response time has a chance of allowing the busier servers to violate their response time goals while using the longest response time among the servers may waste CPU resource and energy. To overcome this limitation, our design uses a performance balancing controller to balance the relative response times of all the VMs by dynamically allocating the CPU resource. The relative response time is defined as $Rt_{measured,i}/Rt_{setpoint,i}$, where $Rt_{measured,i}$ is the measured response time of the web application in the ith VM and Rtsetpoint;i is the corresponding set point defined by the system administrator for the ith VM. This method allows the web application in each VM to have a different set point, which gives flexibility in how VMs are provisioned on physical servers.

The performance balancing controller periodically uses the response time monitor to measure the average response time of the web applications in each VM hosted on the server. It then adjusts the fraction of CPU time allocated to each VM and gives more CPU time to VMs with relative response times above the average. Once the CPU allocations have been determined, they are enforced by a CPU Resource Allocator.

To design the performance balancing controller, we need to address the following challenges: First, the number of VMs running on the server may change at runtime, so the performance balancing controller should be able to adapt to any of these changes. Second, when the workload is light, it is possible that a VM may become completely idle for a certain interval of time, such that the response time monitor cannot give a reading of the response time. To address the two challenges, we design our performance balancing controller as a collection of VM-level controllers. Every VM-level controller reads the relative response time of the VM from the VM-level response time monitor, collects the average relative response time of all nonidle VMs running on

the server, and controls the VM-level relative response time to the server-level average relative response time.

5.2. Integration of Request Batching and DVFS

With a well-designed performance balancing controller, the relative response times of the individual VMs can be controlled to be close to each other. Thus, the average relative response time of the VMs indicates the server-level performance. We adopt two server-level power management techniques, request batching and DVFS, to minimize the energy consumption while guaranteeing the average relative response times of the VMs hosted on the server. Both techniques provide a compromise between the average relative response time and energy consumption. A request batching-based controller periodically controls the average relative response time by tuning the duty cycle, i.e., the fraction of time that the processor is in the active state. The system then enforces the duty cycle by switching the processor between the sleep and active states. A longer duty cycle results in a shorter response time but higher energy consumption. Similarly, the DVFS-based controller manages the response time by tuning the DVFS level. A higher DVFS level means a shorter response time and higher energy consumption.

The two techniques are targeted for different workload intensity ranges of the server. The request batching-based controller can be more power efficient when the workload intensity is relatively low and the DVFS-based controller works better when the workload intensity is moderate. Virtual Batching uses both controllers, dynamically switching between them. In this paper, the DVFS-based controller is designed based on the control algorithm presented in our previous work. The focus of this paper is on the request batching controller and the integration of request batching with DVFS to handle different workload intensities.

5.3. Request Batching Policy

To minimize the server energy consumption under the response time constraint, we design a request batching policy that includes two steps. In the first step, when the server is completely idle, the processor is put into Deep Sleep until new web requests arrive. This step requires that the server should be able to automatically wake up on demand when a web request comes. These mechanisms are often available in computer systems. For example, network adapters are capable of waking up the processor from sleep modes using the support of Wake-on-LAN feature. The ACPI interface allows waking up the processor at a given point of time in the future. After the processor is waken up, the system enters the second step.

In the second step, when the server is not completely idle, it is possible that the response time is unnecessarily shorter than the desired value at the cost of more energy consumption. In this case, it is desirable that the processor be put into Deep Sleep in short periods to allow reduced energy usage. To achieve response time guarantees, the power state of the processor is switched between Deep Sleep and the active mode on a time scale much shorter than the response time requirement of a web request based on a value called duty cycle, i.e., the fraction of time when the processor is put into Deep Sleep. When the processor is sleeping, incoming requests are batched by the network adapter. When there are no pending requests to process, the system is switched to the first step and put into Deep Sleep again. We design a request batching controller to dynamically tune the duty cycle to control the response time of the VMs to a certain set point. Note that though different VMs sharing the server may have different workloads, they share the same duty cycle. When the processor is in the active mode, the performance balancing controller gives more CPU resource to the VMs with heavy workloads such that all VMs will have their desired performance.

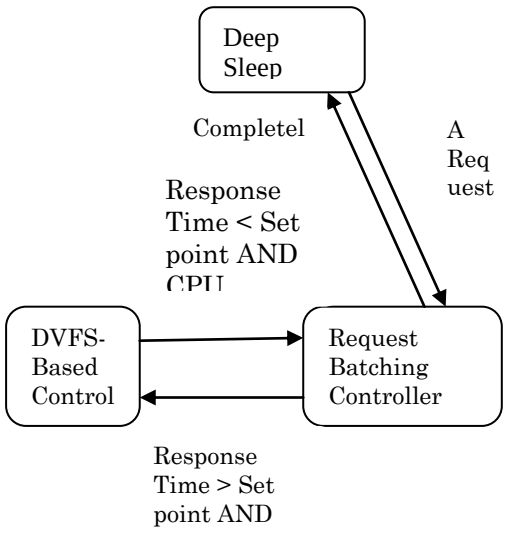

Fig.5.2. Transition Between Request Batching And DVFS Control.

The duty cycle is enforced by switching the processor between the sleep and active states on a small time scale. Based on a recent study [2], the transition time of putting a processor into the Deep Sleep mode is about 30 µs for many processors, such as the Intel Pentium M processor. It is important to note that different processors may have different Deep Sleep modes and Virtual Batching does not assume any specific processors. For processors that do not directly support Deep Sleep, such as Intel Xeon 5400 series, power gating can be used instead, which can effectively reduce the processor power from 80 to 16Win nanoseconds [2]. This small overhead makes it feasible to apply request batching in real server systems. In addition, it has been demonstrated that some other components in a server, such as DRAM DIMM, can also be transitioned into the sleep mode in less than 1 µs in future server design. As a result, the Virtual Batching technique can be extended to put those components into sleep as well, which is our future work. For processors that can achieve very low power consumption at the lowest DVFS level, the Virtual Batching technique can be applied to other components in the server, such as DRAM, for energy savings.

The transition between DVFS control, request batching control, and Deep Sleep is summarized in Fig. 5.2. The details of the design and analysis of the performance balancing controller are available in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.237, of this paper.

6.Virtual Batching with Memory Management

Cloud resources are provided for the users to perform computational tasks. Resources are provided by the providers with different servers. The servers are continuously running to provide resources. Energy consumption level for the servers is increased. Dynamic Voltage Frequency Scaling (DVFS) mechanism is used to adjust the voltage supply for the processors. Power supply is managed under two states. They are sleep state and active state. The power supply is reduced in the sleep state. The resource requests are consolidated with request levels. In the same way the server resource levels are also consolidated with availability details. The request and server consolidation mechanism is used in the Virtual batching technique. The Virtual batching technique is used to manage energy levels in cloud resource sharing environment. The Virtual batching mechanism is enhanced to manage memory resources. The system is designed with the following objectives.

- To manage cloud resources with energy consumption levels
- To apply Virtual Batching technique for energy constrained resource managed process
- To group relevant requests and server resources
- To enhance Virtual Batching with load balancing mechanism
- To incorporate workload variations in dynamic server consolidation process
- To provide power management on memory devices
- To increase the average relative response rate

The Virtual Batching scheme is enhanced to manage resources with load balancing mechanism. The system is improved with optimization mechanism to manage relative response time. Resource levels and application requirements are integrated in the allocation process. The system is adopted to support Dynamic Random Access Memory (DRAM) and Dual in-line Memory Module (DIMM) components.

The Virtual Batching scheme is improved to manage power for computational and storage units. Request consolidation is improved with optimization techniques. Request load is distributed with different servers. The system is divided into five major modules. They are resource management, consolidation process, resource allocation process, load balancing process and power management on memory units.

Resource management module is designed to maintain the resource availability under the providers. Request and server consolidation tasks are carried out under the consolidation module. Resource allocation module handles the scheduling process. Request loads are distributed under load balancing module. Memory devices are managed with power usage levels and memory management module.

7. Conclusion

Cloud resources are managed with energy consumption levels. Virtual Batching scheme is used to allocate resources with request and server consolidation. The system

is improved with optimization schemes to increase response rate. The system is enhanced to support energy management under memory devices. High server utilization is achieved in the system. Energy consumption is minimized by the resource scheduling scheme. The system achieves efficient application performance. The system maximizes the throughput in resource sharing process.

REFERENCES

[1] R. Nathuji, P. England, P. Sharma, and A. Singh, "Feedback Driven QoS-Aware Power Budgeting for Virtualized Servers," Proc. Fourth Int'l Workshop Feedback Control Implementation and Design in Computing Systems and Networks (FeBID), 2009.

[2] D. Meisner, B.T. Gold, and T.F. Wenisch, "PowerNap: Eliminating Server Idle Power," Proc. 14th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2009.

[3] P. Padala, K.-Y. Hou, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated Control of Multiple Virtualized Resources," Proc. Fourth ACM European Conf. Computer Systems (EuroSys), 2009.

[4] X. Wang, M. Chen, C. Lefurgy, and T.W. Keller, "SHIP: Scalable Hierarchical Power Control for Large-Scale Data Centers," Proc. Int'l Conf. Parallel Architectures and Compilation Techniques (PACT), 2009.

[5] A. Verma, G. Dasgupta, T.K. Nayak, P. De, and R. Kothari, "Server Workload Analysis for Power Minimization Using Consolidation," Proc. Conf. USENIX Ann. Technical Conf., 2009.

[6] Intel. Intel 5500 Series Datasheet vol. 1, 2009.

[7] Yefu Wang and Xiaorui Wang, "Virtual Batching: Request Batching for Server Energy Conservation in Virtualized Data Centers", IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 8, August 2013.

[8] X. Zhu, D. Young, B.J. Watson, J. Rolia, S. Singhal, B. Mckee, C. Hyser, D. Gmach, R. Gardner, T. Christian, and L. Cherkasova, "1000 Islands: An Integrated Approach to Resource Management for Virtualized Data Centers," Cluster Computing, vol. 12, pp. 45- 47, 2009.