Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070552

Reducing Resource Over-Provisioning Using Workload Shaping for Energy Efficient Cloud Computing

Woongsup Kim * *and Jaha Mvulla*

Department of Computer and Information Communications Engineering, Dongguk University, Seoul, Korea

Received: 15 Feb. 2013, Revised: 15 Jun. 2013, Accepted: 19 Jun. 2013
Published online: 1 Sep. 2013

Abstract: Using workload shaping technology, we present an approach to remove hardware over-provisioning implementing task buffers and scheduler, in terms of energy consumption. Task buffers reorder tasks with various priorities and routes them to appropriate virtual machines. Scheduler monitors the task buffering and hardware load status, and decides the optimal number of active physical and virtual machines. In addition, we designed a mechanism wherein tasks with fast executing are routed in fast and high energy consumption machines and slow tasks to slow and low energy consumption machines. As a result, our approach efficiently can shape workloads and manage the optimal number of active virtual machines and physical machines, in terms of energy consumption. To evaluate our approach, we generated synthetic workload data and evaluated it both in simulating and actual cloud environment. Our experimental results demonstrate our approach outperforms in terms of energy consumption to when not using no workload shaping methodology.

Keywords: Cloud Computing, Traffic Shaping, Green Computing, Distributed Computing, Power Management

1 Introduction

In recent times, cloud computing [1] has attracted considerable attention as it provides a new paradigm for the virtually unlimited provisioning of computing infrastructure [2]. In the cloud, resources are interconnected in parallel and virtually provisioned by the cloud provider. Through cloud computing, users can acquire and release required resources on demand and access data without knowing the exact physical locations and identifiers of the resources. By this means, customers will be able to access applications and data transparently all over the world.

An Internet Data Center (IDC) is a common platform to provide virtualized services for cloud computing [3,4]. An IDC usually deploys hundreds or thousands of densely packed blade servers to save server space. Executing services using IDC servers provides customers with alternatives to operating their own computers; hence, an IDC can reduce the cost of system ownership and system maintenance. The computing paradigm facilitates the reduction of acquisition and management costs of hardware and software resources.

However, the energy costs for operating and cooling cloud resources have increased significantly, to the point where they will surpass the cost of purchasing hardware resources, as the energy consumed by IDCs is directly related to the number of host servers and their workloads [5,6,7]. Thus, reducing energy consumption can significant reduce management costs. Moreover, the heat generated from operating multiple computing devices could increase the probability of system failure. Therefore, reducing energy consumption can help reduce costs by eliminating the need for heat cooling devices for increased system reliability.

There are several approaches for reducing energy consumption [8,9,10]. Existing works in the field of energy management aimed at reducing energy consumption is divided into three levels: physical machine level, by controlling machine components [11]; network traffic level, by controlling network traffic to reduce the number of operating nodes [12]; deployment level, by controlling where to deploy nodes [13]. To reduce energy consumption at the machine level computer, components change their operating speed when they are not in use. At the deployment level, computing nodes in an IDC server farm can be switched to sleep mode or active mode. At the network traffic level, network traffic can be routed such that the number of

* Corresponding author e-mail: woongsup@dongguk.edu

operating nodes are minimized. Energy management at the machine level relies on a single user's local activity, at the network traffic level it depends on incoming traffic, and energy management at the deployment considers the workload dynamics. All of these energy management approach should ensure service availability and on-time service delivery while reducing energy consumption.

One of the major problems of energy consumption in cloud infrastructure is hardware over-provisioning. Typically, cloud computing resources are over-provisioned in order to maintain service availability and prevent service delays. To maintain service availability, the amount of service provisioning in the cloud infrastructure should be able to handle the peak level of incoming demands without resulting in service delays for customers. However, usually, the number of incoming service demands are much lower than the peak level and hence a large portion of computing resources in cloud infrastructure run idle most of time. Therefore, we could achieve optimal energy consumption if we could shut down, hibernate, or switch to idle unused physical machines, while guaranteeing service availability and on-time service delivery.

The basic approach we adopted for energy consumption management is workload shaping [14,15,16] where the workload of various concurrently running applications is shaped to fit the available machine's capacity. Our approach exploits an energy optimization strategy in the cloud infrastructure at the network traffic level. Through workload shaping, the dynamic workloads can be smoothed out to reduce the local peak level of service requests, and the number of active physical machines required to maintain service availability can thus be reduced. The reduced number of operating nodes can result in reduced energy consumption. In this paper, we propose a workload shaping technique for the optimal energy management scheme based on virtualization technology [17] for cloud infrastructure. Virtualization makes use of existing software and hardware, allowing more than one service to operate on the same piece of hardware, thereby improving hardware utilization. As single machines with high loads consume less energy than several lightly loaded machines, using virtualization can reduce the cost of energy maintenance. Through virtualization technology, the number of active physical machines can be controlled dynamically. Dynamic resizing of the number of active physical machines allows optimal number of hosting nodes to be deployed in cloud infrastructure, thereby improving energy efficiency.

To implement workload shaping technology in cloud infrastructure, we use several *task buffer*s that reorder tasks with various priorities and route them to the appropriate virtual machine. In our approach, we implemented a *scheduler* to control the number of active virtual machines and active physical machines dynamically by relocating or turning on/off virtual machines. Our *scheduler* monitored the task buffering and CPU load status of each physical machine and decided which virtual machine to route a task to and whether to turn on/off virtual machines and physical machines, so as to reduce energy consumption.

This paper is organized as follows. The mathematical energy consumption model is described in Section 2. We present this model in order to provide evaluation measurement that can evaluate that our shaping approach works effectively in simulated environment in terms of energy consumption in cloud infrastructure. Section 3 describes our workload shaping methodology using *task buffer* and *scheduler*. Section 4 illustrates our simulated results based on our own cloud simulator and our experimental results for real cloud system configuration. We compared our results to the typical threshold-based cloud system configuration with no energy consumption consideration. Finally, we present some contributions and final discussions in Section 5.

2 Energy Consumption Model for Cloud Environment

In this section, we present an energy model to evaluate the effects on energy consumption in cloud infrastructure. First, we define the energy consumption model in a single physical machine through dynamic speed scaling. The energy consumed by a single physical machine is composed of base energy consumption independent of CPU clock speed and dynamic energy consumption dependent on CPU clock speed changes. Equation (1) describes the energy consumption in a single physical machine. Here, we assume there is steady energy consumption by all hardware components except CPUs.

$$P_i = \int_{t_0}^{t_1} (x_i(t)\cdot(c+\beta_i f_i(t))\,dt \qquad (1)$$

In equation (1), P_i is the energy consumption of a single physical machine i during the time spanning $t_0 \le t \le t_1$. $x_i(t)$ denotes machine on-off states at time t (when a machine i is in on state $x_i(t) = 1$ and when a machine is in off state $x_i(t) = 1$). c_i is the constant energy consumption of the machine i. c_i includes the CPU's base energy consumption and the energy consumption of all the other components in the machine i. In equation 1) above, $\beta_i f_i(t)$ is CPU's consumption, which varies with CPU operating frequency $f_i(t)$. For efficient cloud server consolidation, the system may use computer's stand-by option rather than turn off host nodes to reduce waiting time spent on the host nodes booting. Considering computer stand-by option, the energy consumption model of a physical machine is defined in equation (2).

$$P_i = \int_{t_0}^{t_1} (x_i(t)\cdot\overline{s_i(t)}\cdot(c+\beta_i f_i(t)))\,dt + \overline{x_i(t)}\cdot s_i(t)\cdot st_i\,dt \qquad (2)$$

Here, st_i is the amount of energy that hardware i in in the stand-by mode consumes at time t. $s_i(t)$ denotes stand-by mode activation state at time t (when a machine i is in stand-by mode, then $s_i(t) = 1$, otherwise $s_i(t) = 0$). From equation (2), $x_i(t) = 0$ when standby mode is activated in a physical machine i. The energy consumption of all the physical machines in a cloud infrastructure can be expressed as follows.

$$P_{all} = \sum_{i=1}^{N} P_i \tag{3}$$

Here P_{all} represents the sum of all physical machines' energy consumption and is expressed with the sum of P_i in equation (2). N is the number of available physical machines in a cloud infrastructure.

The service availability conditions are represented as follows.

$$\sum_{i=1}^{N} x_i(t)\lambda_i(t) = \lambda_{cloud}(t) \tag{4}$$

$$x_i(t)(1-x_i(t)) = 0, i = 1,2,3,\ldots,N \tag{5}$$

$$st_i(t)(1-x_i(t)) = 1, i = 1,2,3,\ldots,N \tag{6}$$

$$g(\alpha_i f_i) > h(\lambda_i(t)), i = 1,2,3,\ldots,N \tag{7}$$

where $\lambda_i(t)$ is service requests for the machine i at time t, λ_{cloud} is service requests from all the cloud clients at time t, $g(\alpha_i f_i)$ is machine i's processing capacity when the machine's CPU runs at frequency f_i, and $h(\lambda_i)$ is the required amount of CPU usage to process requests $h(\lambda_i)$ without making delay of service delivery. Equation (4) describes all the service requests should be processed in one of active machines all the time. Equation (5) describes all physical machines have either on or off state. Equation (6) is presented to guarantee that the machine should not be on state when the machine is in stand-by mode. The final condition as shown in equation Equation (7) is that the number of processing requests to machine i must be less than machine i's processing capacity.

The objective of the energy optimization problem is to minimize P_{all} without breaking conditions for service availability. This is reflected in our objective function (Equation (8)).

$$\text{Minimize } P_{all} = \sum_{i=1}^{N} P_i \tag{8}$$

Subject to the following constraints:

$$\begin{aligned}\sum_{i=1}^{N} x_i(t)\lambda_i(t) &= \lambda_{cloud}(t)\\ x_i(t)(1-x_i(t)) &= 0, i = 1,2,3,\ldots,N\\ st_i(t)(1-x_i(t)) &= 1, i = 1,2,3,\ldots,N\\ g(\alpha_i f_i) &> h(\lambda_i(t)), i = 1,2,3,\ldots,N\end{aligned}$$

Constraint represented with Equation (4) - (7) ensures that the capacity of each physical machine is not exceeded, and all service requests are processed during the given time span.

3 Workload Shaping Approach to Prevent Resource Over-provisioning

This section describes our workload shaping technique employed to reduce energy consumption in a cloud infrastructure composed of multiple resources. The workload shaping involves shaping workloads to reduce the peak service demands and routes tasks to active resources, while each workload of every physical resource fits the resource's own processing capacity.

To this end, we implemented *task buffers* and a *scheduler*. The purpose of our *task buffers* in our approach is to smooth out local peak levels of task requests, while ensuring that each machine manages tasks as per its capacity limit. Consequently *task buffers* in each virtual machine holds off task execution by putting task requests into the *task buffer* rather than executing task immediately, The *scheduler* takes charge of server consolidation [18], by deciding the number of active physical and active virtual machines without breaking service availability. In addition, since a physical machine can run one or more virtual machines in the cloud environment, and a virtual machine can move from one machine to another without interruption due to migration technology [19], the *scheduler* dynamically locates virtual machines in any of the available physical machines on the fly, thus minimizing the number of active physical machines.

Fig. 1 describes our approach using workload shaping technology. Our approach maintains several *task buffers* where each *task buffer* is associated with one virtual machine and is labeled with a predetermined priority. Tasks are identified in advance based on the task priority and routed to a *task buffer* with the same priority. When the system receives new request, *job dispatcher* first identifies new request's task priority and assigns it to the appropriate *task buffer*. *Task buffer* then holds the request if it detects that processing any new task request breaks CPU load threshold, in order to smooth out request arrivals. *Scheduler* monitors this *task buffer*'s status and creates new *task buffers* and activates a physical machine

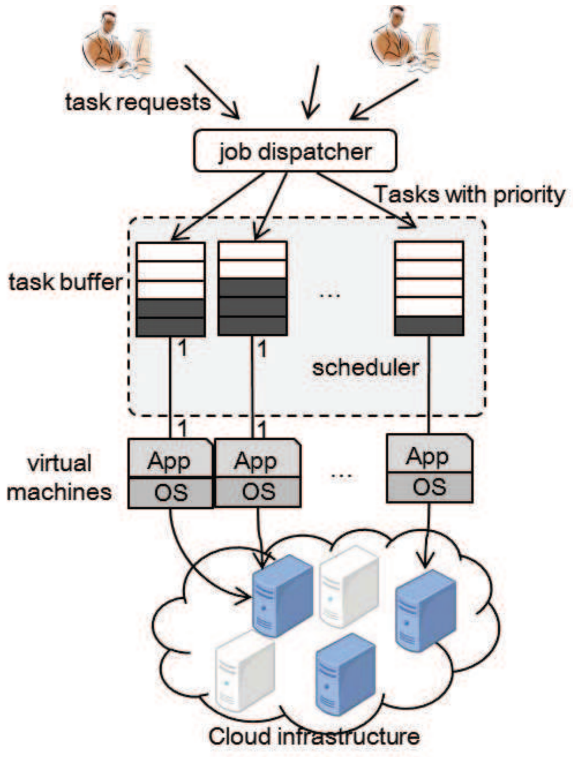

Fig. 1: Workload shaping approach in cloud infrastructure.

if it detects that there is any chance of the any task execution delayed and breaking service availability.

As we do not know the number of requests in advance, initially we activate *n task buffer*s and *n* corresponding virtual machines in a single physical machine which consumes a low level of energy. Here, *n* refers to the number of task priorities in a cloud infrastructure. The number of *task buffers* and virtual machines increases or decreases based on the number of tasks loaded in the *task buffer*. *Scheduler* creates a new *task buffer* and its associated virtual machine when it detects that there is a chance of task execution delay and a breakdown of service availability. Whenever a new *task buffer* and its associated virtual machine are activated, a task priority is assigned to the *task buffer* and tasks of that priority are routed to the new *task buffer*. If the CPU load in a physical machine crosses the threshold limit, *scheduler* activate new physical machine. Based on processing capacity of physical machines, *scheduler* decides which virtual machine is migrated to the new physical machine. By default *scheduler* moves virtual machines with high priority tasks to faster and high energy consuming physical machines.

In our approach, the priority of tasks is determined based on their expected execution time because typical cloud tasks do not have any priority. Thus, tasks with similar execution times are assigned to the same *task buffer*. This implies that a virtual machine always runs tasks with similar expected execution times. To estimate task execution time, we use the execution history recorded in a log file. In our approach, we set the deadline to all the tasks with $\rho\ \times$ expected execution time, and force all the tasks to complete satisfying the deadline. The deadline is the time limit by which a task should be completed. To accomplish this, we put *scheduler* to hold a task request in the *task buffer* until the request's suspended time reaches $(\rho - 1)\ \times$ expected execution time.

The condition in which a new virtual machine is allocated is based on the *task buffer*'s status. Scheduler investigates all the *task buffer*s to check if there is any chance of breaking the deadline of a task through its expected execution time. *scheduler* also considers the number of tasks in the *task buffer*s to decide new virtual machine allocation. Overflows in *task buffer*s result in the allocation of a new virtual machine and its associative new task buffer. In addition, virtual machines are released when the associated task buffers stay empty for pre-defined periods.

The condition in which a new physical machine is allocated is represented as follows.

$$\forall i, h_l(\lambda_i(t)) > \psi \times g_l(\alpha_i f_i), i = 1,2,3,\ldots,k \tag{9}$$

where k is the number of current active physical machines for a task with l priority, $h(\lambda_i(t))$ is the physical machine i's utilization rate at time t, $g_l(\alpha_i f_i)$ is the physical machine capacity at the CPU clock frequency f_i for a task with l priority, and ψ is the threshold to avoid service delay due to task saturation in the virtual machine.

The condition to withdraw the existing physical machine is represented as follows.

$$\begin{aligned}&\exists i, h_l(\lambda_i(t)) < \gamma \times g_l(\alpha_i f_i) \wedge\\&\exists j, j \neq i, h_l(\lambda_i(t)) < \psi \times g_l(\alpha_i f_i),\\&i,j = 1,2,3,\ldots,k\end{aligned} \tag{10}$$

where γ is the low level threshold where the existing physical machine is deactivated when current CPU load is under the threshold.

Scheduler withdraws physical machines when their CPU load in a physical machine falls below the low level threshold. At this point, new incoming task will not be assigned to any of the virtual machines located on the physical machine anymore and eventually virtual machines and their associated *task buffer*s will have no tasks to run. Then *scheduler* stops running the virtual machine and removes the associated *task buffer*. Through the migration technology, ideally virtual machines can move or merge from one machine to another to reduce the number of active physical machines. However, due to an implementation problem, we could not finish implementing ideal virtual machine migration.

Applying our approach makes an idle physical server active longer than using migration technology directly. However, the delay time for using our approach is estimated as at most the executing time of longest current running task in the physical machine, and the amount of energy consumption of delayed idle server would not seems to be significant considering virtual machine migration cost from one physical machine to another.

4 Evaluation

This section presents our experimental result both in simulated environment and in the actual cloud implementation.

4.1 Energy Consumption in a Simulated Environment

We evaluate our approach using a custom-built simulator. We assumed there are 100 physical machines in a cloud infrastructure, where each physical machine can have maximum 10 virtual machines, and all machines are of equal processing capacity (100 load capacity) with different CPU speeds f_i (5-10). The energy consumption of each active machine is calculated as $100 + \beta f_i$ (β = 10). The energy consumption from the off-state machine is set to 0 and that of the stand-by machine is set to 2. We also set the CPU clock frequency level at two levels: 1) when the machine is idle or runs slowly, it consumes 100 units of energy, and 2) when the machine runs at a normal speed, the machine i consumes $100 + \beta f_i$ units of energy.

For the evaluation, we generated workload of 100, 200, 400, 800, 1600, and 3200 tasks (10 mean inter arrival times). Task request intervals were modeled following Poisson process. To generate the workload, we used three types of task patterns: random, high, and low. In the random pattern, the CPU utilization rate for each task is generated at random and uniformly distributed between $5 \sim 30$ load units. In high pattern, the task utilization rate is distributed in $20 \sim 30$ range, while the task utilization rate is in the $5 \sim 15$ load unit range in the low pattern. The execution time for each task is generated randomly from 5 to 50 tick range. For the experiment, we set the upper threshold ψ at 0.8 and the lower threshold at 0.2.

Fig. 2 - 4 shows the energy savings using various parameters (m: task buffer size, ρ: smoothing deadline parameter). To justify our approach, we compared our work to the normal threshold based approach. In threshold approach, there is no workload shaping: a new virtual machine is created only when the CPU usage become above upper threshold and new virtual machines are released only when virtual machines are on idle. We applied various parameter values for *task buffers* to determine the optimal configuration.

As can be seen in our experimental result shown in Fig. 2 - 4, our approach proves to be more effective than using a non-workload shaping approach in terms of energy consumption. Especially when task requests are uniformly distributed (Fig. 2) or the requested tasks does not require heavy CPU processing power (Fig. 4), our approach exhibits more energy saving gains. In addition, we noticed large sized buffers and smoothing parameters help in energy consumption saving. However, a large buffer size and smoothing parameters may cause more delays in service delivery as large-sized buffers can hold off more task requests. Therefore we need to avoid using very large parameters in order to provide faster service delivery.

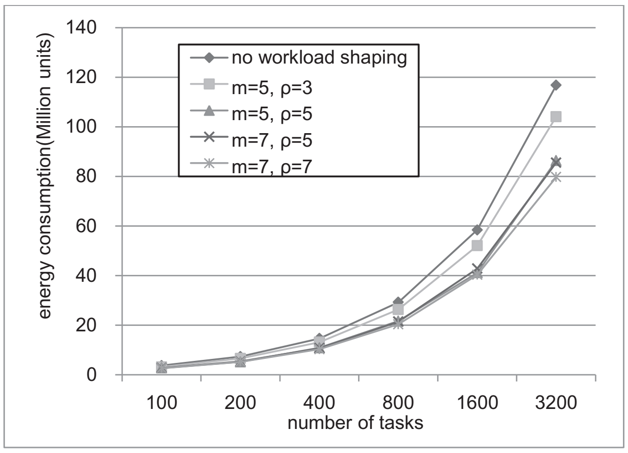

Fig. 2: Energy consumption with synthetic job request (random task pattern)

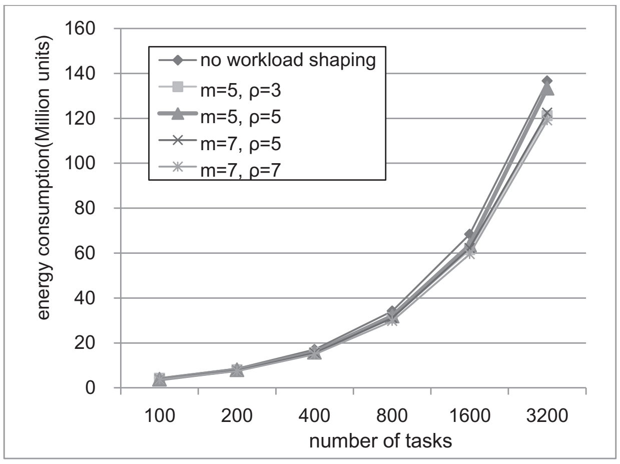

Fig. 3: Energy consumption with synthetic job request (high task pattern)

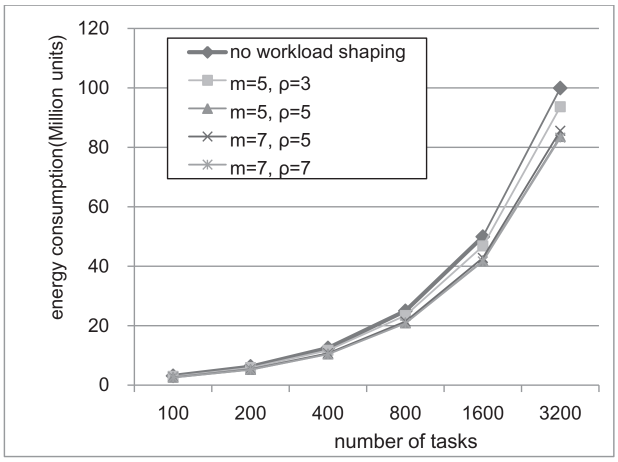

Fig. 4: Energy consumption with synthetic job request (low task pattern)

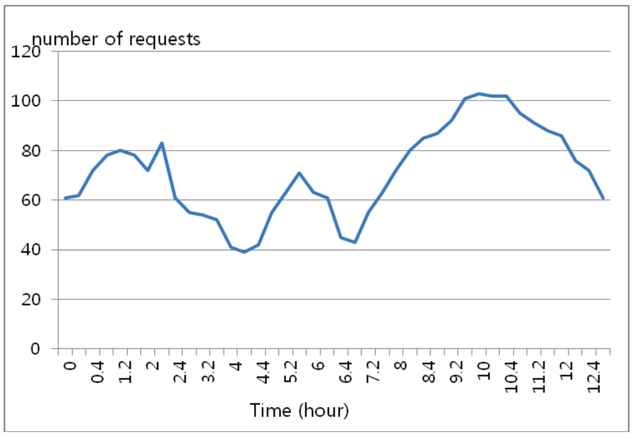

Fig. 5: The number of task requests generated from log traces

4.2 Energy Consumption in Actual Cloud Infrastructure

We also implemented our workload shaping technology approach in an actual cloud infrastructure. Our implementation was based on CentOS 6.2 64bit, 10.04, XenServer 6.0.201 [20] and Apache Cloud Stack 3.0.1 [21]. For the experiment, we implemented three host servers and a management server in which each server has dual core 2.5 GHz Pentium CPU with 8 GB memory. Each host server is interconnected management server and set to hold maximum 6 virtual machines. *Job Dispatcher*, *Scheduler*, *task buffer*s were implemented and located in the management server.

To generate workloads, we developed a load generator that spawns task requests to the management server in our implementation. Fig. 5 illustrates the number of job requests generated by the load generator during a given time span. For realistic playback of task request, we executed several grid-based distributed indexing services for 5, 10, 15, and 20 GB data; measured the execution time for each external resource request or external function calls; and recorded the execution time of external resources or function calls in a log file. Based on the log file, we built job request traces shown in Fig. 5. Fig. 6 illustrates cumulative execution time distribution in our experimental task request list. Using trace data shown in Fig. 5, our load generator playbacks workloads to emulate real clients' task requests. For the experiment, we identified the task priority based on the average execution time in log traces, and route a task request to the corresponding *task buffer*s by estimating the expected execution time using the task's average execution time.

For the evaluation we directly measured the energy consumption every 20 minutes using power measuring

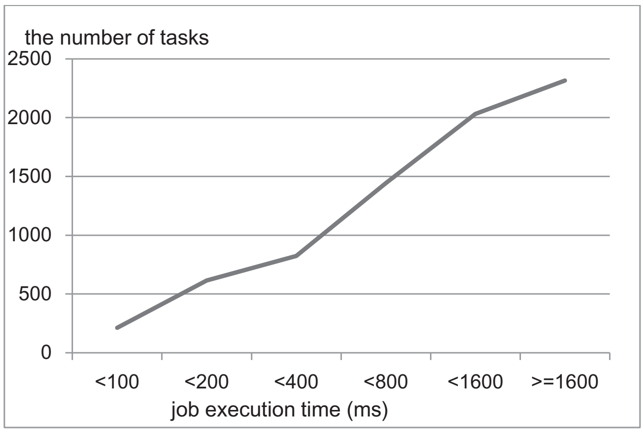

Fig. 6: Cumulative task distribution based on task execution time

device. Table 1 shows a comparison of energy consumption between no workload shaping technology and workload shaping technology. The energy consumption here is the average energy consumption of three host servers. As we can see in table 1, our approach can reduce energy consupmtion by up to 13 % with appropriate parameter values. During the experiment, we noticed task buffering size works as more deciding factor for energy consumption saving. Larger buffers result in greater energy saving as they can hold off more task requests. Consequently larger buffers can suspend new virtual machine's activation and hence the number of active physical machines can be further reduced. That facilitates more effective energy consumption management in cloud infrastructure.

Table 1: Energy consumption comparison in actual cloud environment

Energy Management Strategy	Energy Consumption (avg. W)	Energy Saving (%)
No workload shaping	621	100
$m = 3, \rho = 3$	582.3	93.7
$m = 3, \rho = 5$	568.7	91.6
$m = 5, \rho = 3$	559.1	90.0
$m = 5, \rho = 5$	542.3	87.2

5 Conclusion

We proposed an approach for energy consumption optimization for the cloud environment. through workload shaping. Our approach exploits virtualization technology by utilizing virtual machines and server consolidation. To enable energy-efficient cloud infrastructure, we implemented a workload shaping technique into the cloud infrastructure using task buffers and scheduler. Task buffers reorder task requests to smooth out the peak job requests and enable to host minimum number of active virtual machines. Scheduler monitors CPU loads of physical machines and maintains an optimal size of active physical machines by deciding when physical machines should be turned on/off. In addition, we designed a mechanism wherein tasks with fast executing are routed in fast and high energy consumption machines and slow tasks to slow and low energy consumption machines. As a result, we concluded that our approach is useful and effective in reducing energy consumption in cloud infrastructure.

In the future, we need to use more meaningful and practical data for the cloud environment, in order to justify our work. In addition, we need to compare existing workload shaping algorithms and refine our algorithm to provide a more optimal solution. Finally, we should add more configurable option such as dynamic on-the-fly virtual machine migration to further reduce energy consumption.

Acknowledgement

This work has been supported by Dongguk Research Fund of 2011.

The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] A. Weiss, netWorker, **11**, 16-25 (2007).
[2] C. E. Amrani, K. B. Filali, K. B. Ahmed, A. T. Diallo, S. Telolahy, T. El-Ghazawi, Proceedings of CCGRID **2012**, 690-693 (2012).
[3] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen, Proceedings of the 6th international conference on Autonomic computing and communications industry session, 29-38 (2009).
[4] D. Bein, W. Bein, S. Phoha, Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations, 71-75 (2010).
[5] A. Berl, E. Gelenbe, M. di Girolamo, G. Gioliani, H. de Meer, M.Q. Dang, and K. Pentikousis, The computer journal, **53**, 1045 (2010).
[6] D. M. Quan, A. Somov, C. Dupont, Proceedings of the First international conference on Energy Efficient Data Centers, 129-140 (2012).
[7] K. Ye, D. Huang, X. Jiang, H. Chen, S. Wu, Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, 171-178 (2010).
[8] San Murugesan, IEEE IT Professional, 24 (2008).
[9] W. H. Kemp, The Renewable Energy Handbook: A Guide to Rural Energy Independence, Off-Grid and Sustainable Living, Aztext Press (2006).
[10] C. Dupont, T. Schulze, G. Giuliani, A. Somov, F. Hermenier, Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, **4**, (2012).
[11] T. T. Tesfay, R. Khalili, J. Le Boudec, F. Richter, A. Fehske, Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, 83-92 (2011).
[12] W. Feng, H. Alshaer, J. M. H. Elmirghani, International Journal of Network Management, **18**, 447-464 (2009).
[13] D. Niyato, S. Chaisiri, and L. B. Sung, Proceedings of the 9th IEEE/ACM international symposium on cluster computing and the grid, 84-92 (2009).
[14] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, Proceedings of ICAC-INDST **09**, 19-28, (2009).
[15] P. Kanuparthy and C. Dovrolis, Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, 473-482 (2011).
[16] S. Manolache, P. Eles, Z. Peng, Proceedings of the conference on Design, automation and test in Europe, 718-723 (2006).
[17] H. Meinhard, Proceedings of VTDC, **12**, 27-28 (2012).
[18] A. Corradi, M. Fanelli, L. Foschini, Proceedings of the 2011 IEEE Symposium on Computers and Communication, 129-134 (2011).
[19] Hajjat, X. Sun, Y. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, M. Tawarmalani, Proceedings of ACM SIGCOMM, **2010**, 243-254 (2010).
[20] A. Lenk, M. Klems, J. Nimis, S. Tai, and To. Sandholm, Proceedings of ICSE Workshop of Software Engineering Challenges of Cloud Computing, 23-31 (2009).
[21] P. Barham, B, Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, Proceedings of the 19th ACM symposium on Operating systems principles, 164-177 (2003).

Woongsup Kim was born in Seoul, Korea in 1970. He received the B.S. degree in Computer Engineering from Seoul National University, Korea in 1998, M.S. degree in Computer and Information Science from University of Pennsylvania, USA, in 2001, Ph.D. degree in Computer Science and Engineering from Michigan State University, USA, in 2006 respectively. Since 2007, he has been with Department of Computer and Information Communications Engineering, Dongguk University, Seoul, Korea, where he is currently Assistant Professor. His research interests are in the areas of Software Engineering, Web Services, Semantic Web, Distributed Computing, and Cloud Computing.

Jaha Mvulla was born in Salaam, Tanzania in 1981. He re ceived the B.Sc. degree in Electrical and Computer Systems Engineering from Das es Salaam University, Tanzania in 2007. Since 2012, he has been with Department of Computer and Information Communications Engineering, Dongguk University, Seoul, Korea, where he is currently in Msc degree program. His research interests are in the areas of Computer Networks, Distributed Computing, and Cloud Computing.