Resource Virtualization Complex Scheduling and Task Execution Strategy Based on CloudSim Simulation Framework

Dongxian Shi[1,*]

1 ZheJiang Economic & Trade Polytechnic
Hangzhou, Zhejiang, China

***Abstract* — Data center inefficiency is a widespread and growing. The paper achieves this through workload consolidation onto a set of servers and powering off servers that become idle after consolidation. The main idea is to reduce power wastage by idle servers that do not have any workload. This paper presents an efficient method of server consolidation through virtual machine migration among server farms. To evaluate the methods of resource virtualization complex scheduling and task execution strategy, the paper uses CloudSim framework which provides basic models and entities to validate and evaluate energy-conscious provisioning of techniques/algorithms. The paper has made a number of extensions to Framework to enable it to simulate efficient energy-conscious provisioning policies at resource, and VM level in a secure way. It does this by extending its classes to include additional power model object for managing power consumption on a per Cloud host basis. The result showed that the CloudSim framework is more accurate and reliable.**

Keywords - cloudSim framework; VM level; resource virtualization complex scheduling; task execution strategy

I. INTRODUCTION

Recent advances in hardware technologies including low-power processors, solid state drives, and energy-efficient monitors have alleviated the energy consumption issue to a certain degree, and a series of software approaches have significantly contributed to the improvement of energy efficiency [1]. Most of the current state-of-the-art research on energy efficiency has predominantly focused on achieving energy efficiency through virtual machine consolidation. Other power saving solutions focused on making the data center hardware components power efficient.

Technologies, such as Dynamic Voltage and Frequency Scaling (DVFS), and Dynamic Power Management (DPM) were extensively studied and widely deployed. Because the aforementioned techniques rely on power-down and power-off methodologies, the efficiency of these techniques is at best limited [2]. In fact, an idle server may consume about 2/3 of the peak load .However most of the methods explored previously by other researchers have security vulnerabilities and low energy saving.

The paper achieves this through workload consolidation onto a set of servers and powering off servers that become idle after consolidation. The main idea is to reduce power wastage by idle servers that do not have any workload [3]. This paper presents an efficient method of server consolidation through virtual machine migration among server farms.

The paper makes major contributions by providing a comprehensive consolidation strategy using secure VM live migration. It discusses efficient ways of deploying virtual machines to servers and migrating virtual machines among servers clusters based on sever workload utilization using dynamic round-robin algorithm. The ultimate result of the method is the reduction of the number of physical machines used since the number of physical machines used greatly affects the overall power consumption. The paper further discusses known security threats to VM's during live migration and presents mitigation strategies to the threats. The methods discussed in the paper preserves privacy and integrity of protected contents by making sure that memory pages of secure processes are kept inaccessible from other processes and operating system kernel during migration. The methods presented eliminate security vulnerabilities faced during live migration.

To evaluate the methods discussed in the paper, the paper uses CloudSim framework which provides basic models and entities to validate and evaluate energy-conscious provisioning of techniques/algorithms [4-5]. The paper has made a number of extensions to Framework to enable it to simulate efficient energy-conscious provisioning policies at resource, and VM level in a secure way. It does this by extending its classes to include additional power model object for managing power consumption on a per Cloud host basis. To support modeling and simulation of different power consumption models and power management techniques such as Dynamic Voltage and Frequency Scaling (DVFS), the paper provides an abstract implementation to handle this. This capability enables the creation of energy-conscious provisioning policies that require real-time knowledge of power consumption by Cloud system components.

Furthermore, it enables accounting of total energy consumed by the system during the simulation period. To make sure that VM is protected against attacks, the paper extended the C1oudSim classes and implements the security module in it. It is the main class, which is responsible for managing event queues and controlling step by step (sequential) execution of simulation events. Every event that is generated by the CloudSim entity at run-time is stored in the queue called future events.

These events are sorted by their time parameter and inserted into the queue. Next, the events that are scheduled on each step of the simulation are removed from the future events queue and transferred to the deferred event queue. Following this, an event processing method and the security module is invoked for each entity, which chooses events from the deferred event queue and performs appropriate actions.

This paper also does significant contributions by doing extensive investigation on various power-aware VM provisioning schemes like DVFS and DNS. For this purposes, the paper presents an optimal power aware provisioning scheme based on comparative evaluation on existing schemes on various data center architectures. The paper has made a number of extensions to CloudSim framework to enable it to simulate efficient energy-conscious provisioning policies at resource, and VM level in a secure way. It does this by extending the Cloudsim's core classes to include an additional Power Model object for managing power consumption on a per Cloud host basis. This is a significant contribution since this modification enables the framework to simulate different power consumption models and power management techniques such as Dynamic Voltage and Frequency Scaling (DVFS) [6].

From the results the paper presents, it is observed that virtual machine migration using Dynamic Round robin algorithm for server consolidation is an extremely feasible solution to reduce energy consumption in a data center without compromising on security.

II. SECURE LIVE MIGRATION

Migration controller is a fuzzy-logic based feedback control loop. An advisor module of the controller continuously monitors the servers' resource utilization and triggers a fuzzy-logic based controller whenever resource utilization values are too low or too high. When the advisor detects a lightly utilized, i. e., under-load situation, or overload situation the fuzzy controller module identifies appropriate actions to remedy the situation. For this purpose, it is initialized with information on the current load situation of all affected servers and workloads and determines an appropriate action [7]. For example, as a first step, if a server is overloaded it determines a workload on the server that should be migrated and as a second step it searches for a new server to receive the workload. Furthermore, these rules initiate the shutdown and startup of nodes.

The implementation of the workload migration controller uses the following rules:

(i) A server is defined as overloaded if its CPU or memory consumption exceeds a given threshold. In an overload situation, first, a fuzzy controller determines a workload to migrate away and then it chooses an appropriate target server. The target server is the least loaded server that has sufficient resources to host the workload. If such a server does not exist, a new server is started and the workload is migrated to the new one.

(ii) An under-load situation occurs whenever the CPU and memory usage averaged over all servers in the server pool drops below a specified threshold. While an overload condition is naturally defined with respect to a particular server, the under-load situation is different. It is defined with respect to the average utilization of the overall system involving all the nodes. In this way, the paper avoids system thrashing: e.g., a new server generally starts with a small load and should not be considered immediately for consolidation. In an under-load situation, first, the fuzzy controller chooses the least loaded server and tries to shut it down. For every workload on this server, the fuzzy controller determines a target server. If a target cannot be found for a workload then the shutdown process is stopped. In contrast to overload situations, this controller does not ignite additional servers.

The main goal is to provide VM migration capability while preserving strict protection during and after the migration without significant performance degradation and migration downtimes. The paper proposes a security preserving VM live migration architecture. Three modules are added in the VMM. The Migration Data Protection Module is responsible for intercepting, encrypting and decrypting contents that belong to the protected processes inside the migrating VM. The Metadata Management Module is responsible for serializing the metadata for transmission and re-constructing the metadata in the migration target machine. The Security Guard protects the live migration system from several security vulnerabilities.

The Migration Manager is the migration tool inside the control VM. This module is not within the trusted computing base. Thus it provides the migration functionality without being trusted. This is possible by encrypting and hashing all the sensitive data before passing to the module. In the following subsections, the paper first presents the major function modules that provide protected data and metadata migration, and then describe the security issues and our solutions in live migration.

The VMM depicted with light grey is the trust computing base. The Migration Data Protection Module is responsible for intercepting and protecting contents that belong to the protected processes inside the migrating VM which is depicted in deep grey. The Metadata Management Module is responsible for packing and unpacking the maintenance metadata inside the VMM. The Security Guard protects the live migration system from several security vulnerabilities. The Secure Migration Manager is the migration tool in the Control VM. It manages the secure migration but is not necessary to be trusted. The solid lines in the graph show the data flow in the migration [8].

Before migration, memory pages of secure processes are kept inaccessible from other processes and operating system kernel. This protection should remain valid during migration. In addition to protection for local data privacy and integrity, protection from several network attacks during migration should be addressed, including spoofing, replay and man-in-middle attacks. Denial-of-service attacks are not considered as the platform owner or a malicious kernel can refuse to provide service anyway.

Then the migration manager transfers the content of the pages via network to the target machine. When the migration manager tries to map secure pages, the protection logic

embedded in the memory mapping module will make an encrypted copy of the page in the memory buffer of the control VM, and map the encrypted page for the migration manager. This interception, encryption and redirection is done transparently to the control VM kernel and the migration manager.

To encrypt the pages, one can use one global migration session key or assign each page with a random key. Other normal pages are left unprotected. Data integrity can be protected by hashing the memory content at page granularity. These hash values together with the keys should be encrypted again using the private platform key (SK) provided by the TPM before sending them over network. This additional encryption is essential because the migration tools are not trusted.

Normal pages are directly mapped in the memory space of the migration manager. Mappings requests for the secure pages are intercepted by the protection module in the VMM. The protection module makes an encrypted copy of the secure page and redirects the mapping to this page. The solid lines show the actual data flow of the memory contents. The dash lines show the data flow in the view of migration manager. The interception and redirection is transparent to upper operating system kernel and the migration manager.

On receiving the memory pages, the migration manager simply puts the received contents in the corresponding memory locations. Before resuming the VM on the target machine, the VMM is responsible for locating and decrypting all secure pages. It first decrypts the keys and hashes by the public platform key (PK) of the source machine. Then it compares the hash value of the secure pages before decrypting them.

This platform/session key based security protocol is naturally preventing the man-in-middle attack which hijacks the communication by cheating both sides after receiving all the memory pages. The VM image is rebuilt on the target machine. As the rebuilding process is under the control of the potentially malicious migration manager, it may switch the content of two protected pages of a protected process to launch a spoofing attack. This is possible as the paper does not hash the VM image as a whole. To prevent this attack, a kind of binding should be established between the page content and its memory location. A simple way is to bind the hash of each page with the page frame number (PFN) and send them in the metadata transfer phase, which is explained in the following section.

There is TOCTTOU security vulnerability in the map-and send mechanism when live migrating the VM. The key difference is the VM is running when the map-and-send is in process. When the migration manager issues a hyper-call to map a batch of memory pages of the migrating VM, the check is performed to see if the pages are protected. Then the VMM encrypts the protected pages. However, if a page is not protected when the migration manager performs the checking and mapping, and then the page is assigned to a secure process in the migrating VM, that page turns into a sensitive page but is still mapped in the space of the migrating tools.

The paper fixes this security hole by revoking and redirecting the memory mapping to a zero page when the page is assigned as a secure page. The moment a page is allocated as a secure page and the Security module will check the page table of migration manager and force any existing mappings to the secure page to be unmapped. To prevent the page fault, the Security module will temporarily map a zero page to the page table entry. This forced redirection is performed no matter the mapped content has been sent, is being sent, or has not been sent yet. It is possible that part or the entire page may be sent in zero value. This is safe because the page is assigned to another process, which indicates the page content is obsolete. And if the protected process writes the page later, it will be dirtied and sent again. This is safe because the transmitted content is in encryption form and will not be decrypted by the VMM at the target machine because the page is no longer a secure page.

Adopting a wrong resuming order may also incur serious security problems. For example, the Xen migration manager will pin all page table pages when a VM is received and about to resume. Page pinning is a process that Xen validates the mappings in the page tables and guarantees the VM has the privilege to map the memory pages. Meanwhile, the metadata that indicate the security level of pages are also set in the resuming phase. If the page tables are pinned before the correct security level of pages are set, it is possible that an unauthorized page table can bypass the security check and illegally map the secure pages. Although the paper can implement a correct hyper-call invocation sequence in the migration manager, it is possible that the migration manager is compromised, and hence breaks the sequence.

This security hole is fixed by enforcing the resuming order in the VMM. Specifically, the VMM ensures the pinning operation cannot be done before it has fully received and restored the metadata of page security information.

In Summary, short simulation times are provided by CloudSim and MDCSim even for very large data centers due to their event-based nature. While CloudSim offers an improvement in the simulation precision keeping and the simulation time at a reasonable level, none of the tools offer user-friendly graphical interface.

CloudSim allows implementation of the complex scheduling and task execution schemes involving resource virtualization techniques. However, its workloads are more relevant in grid networks. The Green Cloud supports cloud computing workloads with deadlines, but only simple scheduling policies for single core servers are implemented. The MDCSim workloads are described with the computational requirements only and require no data to be transferred. Power saving and energy models support are the key strengths of the CloudSim. Its ease of Extensibility and availability made it a tool of choice in our scenario.

III. CLOUDSIM ARCHITECTURE

The C1oudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for virtual machines (VMs), memory, storage, and

bandwidth. The fundamental issues such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state are handled by this layer. A Cloud provider, who wants to study the efficiency of different policies in allocating its hosts to VMs (VM provisioning), would need to implement their strategies at this layer.

Such implementation can be done by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer related to provisioning of hosts to VMs. A Cloud host can be concurrently allocated to a set of VMs that execute applications based on SaaS provider's defined QoS levels. This layer also exposes functionalities that a Cloud application developer can extend to perform complex workload profiling and application performance study. The top-most layer in the C1oudSim stack is the User Code that exposes basic entities for hosts (number of machines, their specification and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. Figure 1 shows layered CloudSim architecture. By extending the basic entities given at this layer, a Cloud application developer can perform following activities: (i) generate a mix of workload request distributions, application configurations;
(ii) model Cloud availability scenarios and perform robust tests based on the custom configurations; and
(iii) implement custom application provisioning techniques for clouds and their federation.

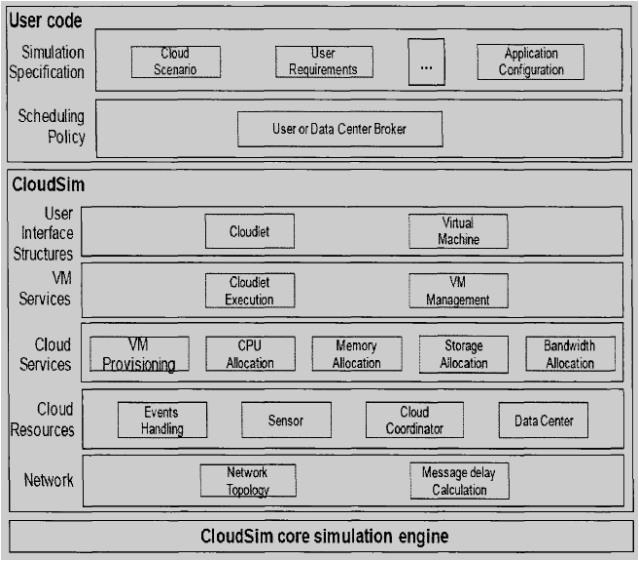

Figure 1. Layered CloudSim architecture.

As Cloud computing is still an emerging paradigm for distributed computing, there is a lack of defined standards, tools and methods that can efficiently tackle the infrastructure and application level complexities. Hence, in the near future there would be a number of research efforts both in academia and industry towards defining core algorithms, policies, and application benchmarking based on execution contexts. By extending the basic functionalities already exposed with CloudSim, researchers will be able to perform tests based on specific scenarios and configurations, thereby allowing the development of best practices in all the critical aspects related to Cloud Computing.

The infrastructure-level services (IaaS) related to the clouds can be simulated by extending the Datacentre entity of C1oudSim. The Data Center entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the Cloud service provider. Here, the VM policy stands for the operations control policies related to VM life cycle such as: provisioning of a host to a VM, VM creation, VM destruction, and VM migration.

Similarly, one or more application services can be provisioned within a single VM instance, referred to as application provisioning in the context of Cloud computing. In the context of C1oudSim, an entity is an instance of a component. A CloudSim component can be a class (abstract or complete), or set of classes that represent one CloudSim model (data center, host).

A Data center can manage several hosts that in turn manage VMs during their life cycles. Host is a C1oudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per second MIPS), memory, storage, and a provisioning policy for allocating processing cores to virtual machines. The Host component implements interfaces that support modeling and simulation of both single-core and mufti-core nodes.

CloudSim supports the development of custom application service models that can be deployed within a VM instance and its users are required to extend the core Cloudlet object for implementing their application services. Furthermore, CloudSim does not enforce any limitation on the service models or provisioning techniques that developers want to implement and perform tests with. Once an application service is defined and modeled, it is assigned to one or more pre-instantiated VMs through a service specific allocation policy. Allocation of application-specific VMs to Hosts in a Cloud-based data center is the responsibility of a Virtual Machine Allocation controller component (called VM Allocation Policy). This component exposes a number of custom methods for researchers and developers that aid in implementation of new policies based on optimization goals (user centric, system centric or both).

By default, VM Allocation Policy implements a straightforward policy that allocates VMs to the Host in First-Come-First-Serve (FCFS) basis. Hardware requirements such as the number of processing cores, memory and storage form the basis for such provisioning. Other policies, including the ones likely to be expressed by Cloud providers, can also be easily simulated and modeled in C1oudSim. However, policies used by public Cloud providers (Amazon EC2, Microsoft Azure) are not publicly available, and thus a pre-implemented version of these algorithms is not provided with CloudSim.

For each Host component, the allocation of processing cores to VMs is done based on a host allocation policy. This

policy takes into account several hardware characteristics such as number of CPU cores, CPU share, and amount of memory physical and secondary) that are allocated to a given VM instance. Hence, CloudSim supports several simulation scenarios that assign specific CPU cores to specific VMs (a space-shared policy) or dynamically distribute the capacity of a core among VMs (time-shared policy); and assign cores to VMs on demand.

Each Host component also instantiates a VM scheduler component, which can either implement the space-shared or the time-shared policy for allocating cores to VMs. Cloud system/application developers and researchers, can further extend the VM scheduler component for experimenting with custom allocation policies. In the next section, the finer level details related to the time-shared and space-shared policies are described. Fundamental software and hardware configuration parameters related to VMs are defined in the VM class. Currently, it supports modeling of several VM configurations offered by Cloud providers such as the Amazon EC2.

IV. VM ALLOCATION AND EXPERIMENT RESULT

One of the key aspects that make a Cloud computing infrastructure different from a Grid computing infrastructure is the massive deployment of virtualization tools and technologies. Hence, as against Grids, Clouds contain an extra layer (the virtualization layer) that acts as an execution, management, and hosting environment for application services. Hence, traditional application provisioning models that assign individual application elements to computing nodes do not accurately represent the computational abstraction, which is commonly associated with Cloud resources. For example, consider a Cloud host that has a single processing core, there is a requirement of concurrently instantiating two VMs on that host. Even though in practice VMs are contextually (physical and secondary memory space) isolated, still they need to share the processing cores and system bus. Hence, the amount of hardware resources available to each VM is constrained by the total processing power and system bandwidth available within the host. This critical factor must be considered during the VM provisioning process, to avoid creation of a VM that demands more processing power than is available within the host. In order to allow simulation of different provisioning policies under varying levels of performance isolation, C1oudSim supports VM provisioning at two levels: first, at the host level and second, at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core will be assigned to each VM. At the VM level, the VM assigns a fixed amount of the available processing power to the individual application services (task units) that are hosted within its execution engine.

For the purpose of this thesis, this thesis considers a task unit as a finer abstraction of an application service being hosted in the VM. At each level, CloudSim implements the time-shared and space-shared provisioning policies. To clearly illustrate the difference between these policies and their effect on the application service performance. A host with two CPU cores receives request for hosting two VMs, such that each one requires two cores and plans to host four tasks units.

An end user or a SaaS provider consumer who is not satisfied with the delivered QoS is likely to switch their Cloud provider hence it is very important requirement that Cloud system simulation frameworks provide facilities for modeling realistic networking topologies and models. Inter-networking of Cloud entities (data centers, hosts, SaaS providers, and end-users) in CloudSim is based on a conceptual networking abstraction. In this model, there are no actual entities available for simulating network entities, such as routers or switches.

This paper considered two sets of experiments to evaluate the proposed approach. The first set of experimental tests was done to test the approach in a two-tier (2T), three-tier (3T), and three-tier high-speed (3Ths) architectures. The second experiments that were done validate the effectiveness of energy-conscious VM provisioning technique proposed.

Proposed algorithms were evaluated through simulations using the CloudSim toolkit which offers the following novel features

(i) Support for modeling and simulation of large scale Cloud computing environments, including data centers, on a single physical computing node.

(ii) A self-contained platform for modeling Clouds, service brokers, provisioning, and allocations policies.

(iii) Support for simulation of network connections among the simulated system elements.

(iv) Facility for simulation of federated Cloud environment that inter-networks resources from private and public domains, a feature critical for research studies related to Cloud-Bursts and automatic application scaling. Some of the unique, compelling features that make CloudSim our chosen simulation framework are:

(i) Availability of a virtualization engine that aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node

(ii) Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

Figure 2 presents a workload distribution among severs.

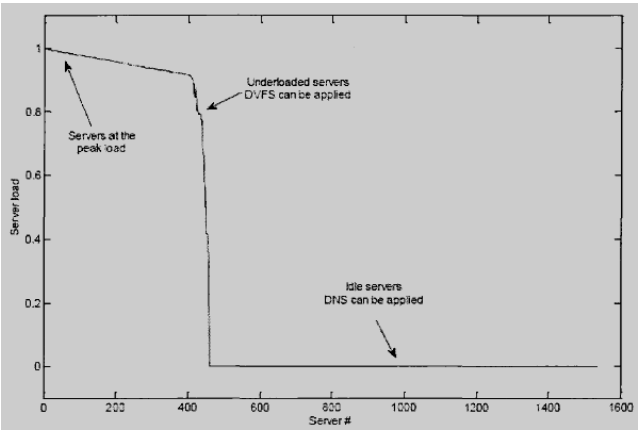

Figure 2. Server workload distribution with a CIoudSim scheduler.

The whole load of the data center (around 30% of its total capacity) is mapped onto approximately one third of the servers maintaining load at a peak rate (left part of the chart). This way, the remaining two thirds of the servers can be shut down using DNS technique. A tiny portion of the approximately 50 out of 1536 servers which load represents a falling slope of the chart are under-utilized on average, and DVFS technique can be applied on them. Figure 3 presents data center energy consumption under variable load conditions for DVFS only and DNS+DVFS power management schemes. The curves are presented for balanced type of the workloads and correspond to the total data center consumption as well as the energy consumed by the servers and switches. The DVFS scheme shows itself little sensitive to the input load of the servers and almost insensitive to the load of network switches. On the contrary, the DNS scheme appears to capture load variation precisely adjusting power consumptions of both servers and switches accordingly. The results reported are averaged over 20 runs with the random seed controlling random number generator. The introduced uncertainty affected mainly the way the workloads arrive to the data center slightly impacting the number of servers and network switches required to be powered.

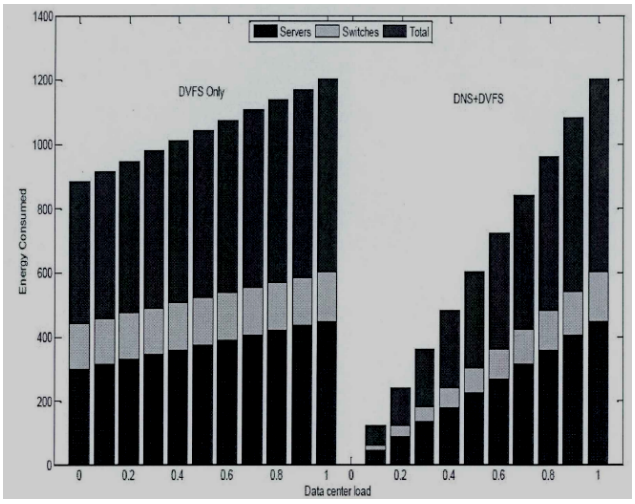

Figure 3. Data center energy consumption comparison.

Figure 4 presents data center energy consumption comparison for different types of user workloads: balanced, computationally intensive and data-intensive workloads.

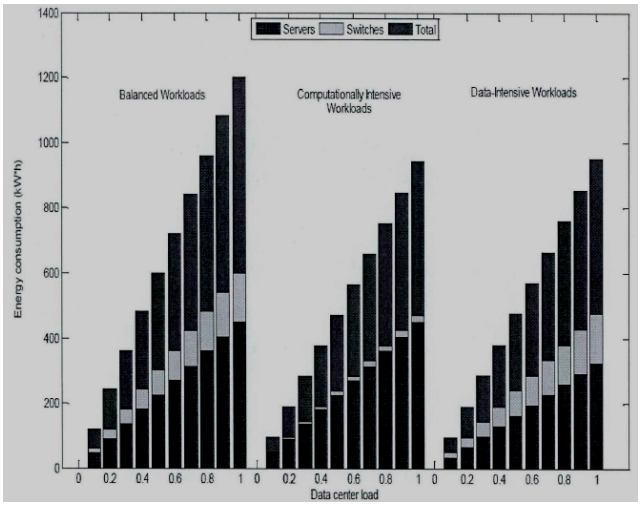

Figure 4. Data center energy consumption for different types of workloads.

Balanced workloads consume the most as the consumptions of both servers and switches become proportional to the offered load of the system. CIWs stress the computing servers and leave data center network almost unutilized. On the contrary, execution of DIWs creates a heavy traffic load at the switches and links leaving the servers mostly idle.

To avoid SLA violations, the VMs were packed on the hosts in such a way that the host utilization was kept below a pre-defined utilization threshold. This threshold value was varied over a distribution during the simulation for investigating its effect on the behavior of the system. The simulation was repeated ten times, the mean values of the results that the thesis obtained are presented in Figures 5.

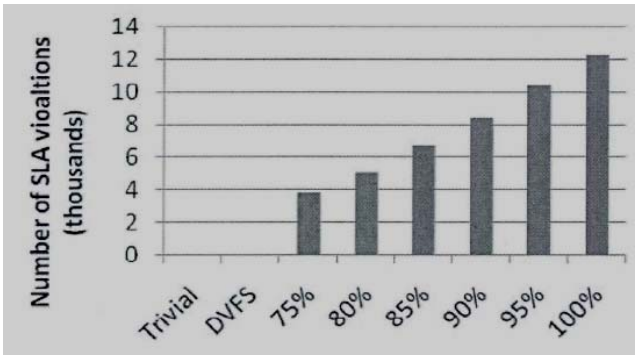

Figure 5. Number of SLA violations.

V. CONCLUSION

This paper has done extensive investigation on various power-aware VM provisioning schemes like DVFS and DNS. The thesis does two sets of experiments to test the presented approach in various architectures like two-tier (2T), three-tier (3T), a three-tier high-speed (3Ths) architectures. Previously, other researcher like Ching-Chet al had performed simulation on two tier architecture only .They overlooked the

face that data centers behave differently when implemented using 2T,3t and 3Ths. The paper is able to present an optimal power aware provisioning scheme based on comparative evaluation the existing schemes on various architectures. The effectiveness of these contributions has been appraised through a comprehensive simulation-driven analysis of the proposed approach based on realistic well-known data center conditions in order to capture the transient behaviors that prevail in existing data center environments. This paper presents an efficient method of server consolidation through virtual machine migration among server farms. To evaluate the methods of resource virtualization complex scheduling and task execution strategy, the paper uses CloudSim framework which provides basic models and entities to validate and evaluate energy-conscious provisioning of techniques/algorithms. The result showed that the CloudSim framework is more accurate and reliable.

REFERENCES

[1] Niroshinie Fernando, Seng W. Loke,Wenny Rahayu, "Mobile cloud computing: A survey". Future Generation Computer Systems, 2, 91-95, 2013.

[2] Daniel R. Williams, Peter Thomond, Ian Mackenzie, "The greenhouse gas abatement potential of enterprise cloud computing". Environmental Modelling and Software, 2:77-89, 2013.

[3] Shih-Wei Lin, Kuo-Ching Ying, "Minimizing shifts for personnel task scheduling problems: A three-phase algorithm". European Journal of Operational Research, 5: 82-90, 2014.

[4] Reza Salimi, Homayun Motameni, Hesam Omranpour, "Task scheduling using NSGA II with fuzzy adaptive operators for computational grids". Journal of Parallel and Distributed Computing, 2014.

[5] Yuming Xu,Kenli Li,Jingtong Hu,Keqin Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues". Information Sciences, 15: 112-118,2014.

[6] M.G. Avram, "Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective". Procedia Technology,12: 298-306, 2014.

[7] Reza Rezaei, Thiam Kian Chiew,Sai Peck Lee,Zeinab Shams Aliee, "A semantic interoperability framework for software as a service systems in cloud computing environments". Expert Systems With Applications,4: 113-119, 2014.

[8] Younis A. Younis,Kashif Kifayat,Madjid Merabti, "An access control model for cloud computing". Journal of Information Security and Applications, 7: 22-33, 2014.