A Proactive Energy-Efficient Technique for Change Management in Computing Clouds

Hady AbdelSalam Kurt Maly
Ravi Mukkamala Mohammad Zubair
Computer Science Department, Old Dominion University
Engineering & Computational Sciences Building,
Norfolk, VA 23529, USA
{asalam, maly, mukka, zubair}@ cs.odu.edu

David Kaminksy
Software Strategy and Technology, IBM
Research Triangle Park, NC 27709, USA
dlk@us.ibm.com

***Abstract*—The intensive use of portable and thin client devices along with the continuously increasing cost of IT management have pushed large IT companies to look for solutions that allow the use of such devices to access the massive computing power of supercomputers available at the company. The cloud computing concept has emerged with promises to simplify and speed up application deployment and maintenance. All these benefits should be provided for a small fraction of current maintenance cost. To be able to provide services to its customers, a cloud requires high level of maintenance and an appropriate strategy for change management. Replacing defective items (hardware/software), applying security patches, or upgrading firmware are just few examples of typical maintenance procedures needed in such environment. While taking resources down for maintenance, applying efficient change management techniques is a key factor to the success of the cloud. Moreover, the increasing cost of energy consumption in such systems has imposed an additional constraint on proposed techniques making the problem more challenging. In this paper, we propose a proactive energy efficient technique for change management in cloud computing environments. We formulate the management problem into an optimization problem to minimize the total energy consumption of the cloud. We validate our analytical model by providing scenarios that illustrate the mathematical relationships for a sample cloud and that provide a range of possible power consumption savings for different environments.**

***Keywords*-Cloud Computing, Autonomic Manager, Policy languages, Change Management, Energy Efficient.**

I. Introduction

A computing cloud [4] can be defined as a pool of computer resources that can host a variety of different workloads, ranging from long-running scientific jobs (e.g., modeling and simulation) to transactional work (e.g., web applications and payroll processing). A cloud computing framework should be able to autonomously and dynamically provision, configure, reconfigure, and deprovision servers as needed in order to satisfy the needs of the cloud users. Servers in the cloud can be physical machines or virtual machines. Cloud-hosting facilities, including many large businesses that run clouds in-house, became more common as businesses tend to outsource their computing needs more and more.

While intermixing workload can lead to higher resource utilization, we believe that the use of subclouds will be more appropriate for future clouds. Not all hardware is created equal: high-end workstations often contain co-processors that speed scientific computations; lower-end workstations can be appropriate for limited I/O requirements; mainframe computers are designed for efficient intensive computing; and so on. For efficiency reasons, we believe that workloads will be partitioned and assigned to subclouds comprised of homogeneous hardware, which is suitable for executing the assigned workloads. A cloud infrastructure can be viewed as a cost-efficient model for delivering information services and reducing IT management complexity. Several commercial realizations of computing clouds are already available today (e.g., Amazon, Google, IBM, Yahoo, etc.) [7].

Managing large IT environments can be expensive and labor intensive [1], [2]. Typically, servers go through several software and hardware upgrades. Maintaining and upgrading the infrastructure, with

minimal disruption and administrative support, can be extremely challenging especially for complex IT environments where a change in some part of the network may have unexpected impacts on other parts due to the complex connectivity and dependency relationships. Currently, most IT organizations handle change management through human group interactions and coordination. Such a manual process is time consuming, informal, and not scalable, especially for a cloud computing environment. The continuously increasing cost of energy consumption in IT systems has imposed an additional constraint on the management problem making it more challenging [5], [6].

The impact of high power consumption is not just limited to the energy cost but extends to the cost of initial investment of the cooling systems needed to get rid of the generated heat and the continuous cost needed to power these systems. To reduce operational cost at these centers while meeting any performance based SLAs (Service Level Agreement), efficient techniques are needed to provision the right number of resources at the right time.

Several software techniques like operating system virtualization, and Advanced Configuration and Power Interface (ACPI) have been proposed to reduce server energy consumption. Other hardware techniques have also been proposed like processor throttling, dynamic voltage and frequency scaling (DVFS), and low-power DRAM states. Modern computing devices have the ability to run at various frequencies each one with a different power consumption level. Hence, the possibility exists to choose frequencies at which different servers run to optimize total power consumption while staying within the constraints of the SLA that govern the running applications.

The process of updating both software and hardware as well as taking them down for repair and/or replacement is commonly referred to as change management. In an earlier work [1], we proposed and implemented an infrastructure-aware autonomic manager for change management. In [2], we enhanced our proposed autonomic manager by integrating it with a scheduler that can simplify change management by proposing open time slots in which changes can be applied without violating any of SLAs reservations represented by availability policies requirements. Motivated by the importance of developing energy efficient techniques, we extend our previous work by proposing a pro-active energy-aware technique for change management in a cloud computing environment.

In this paper, we analyze the mathematical relationship of these SLAs and the number of servers that should be used and at what frequencies they should be running. We discuss a proactive provisioning model that includes hardware failures, devices available for services, and devices available for change management, all as a function of time and within constraints of SLAs. We validate our analytical model by providing scenarios that illustrate the mathematical relationships for a sample cloud and that provides a range of possible power consumption savings for different environments. In other ways, we simply develop a mathematical model that will - under certain assumptions - allow system administrators to calculate the optimal number of servers needed to satisfy the aggregate service needs committed by the cloud owner along with the computation of the frequencies the servers should use.

It is instructive for the reader at this point to note the differences between the analytical model presented in this paper and the model we previously published in [3]. In the previous model, the derivation for the optimal number of servers, k, was based on two main assumptions: (1) The expected cycles per instruction is 1, (2) The approximation that for the same cloud load, the sum of normalized frequencies $(L = \sum_i f_i)$ is independent from the number of running servers k. Unfortunately, this turns out not to be true when servers running frequencies were normalized within $[0, 1]$ range. In this paper, we get rid of this approximation by representing k as a function of the running frequencies. We also generalize the model for processors with $CPI > 1$. Although, the resulting formula we obtained from new model are much more complex than the simplified model, the results turns out to be more accurate. Furthermore, we tried to validate the new model by providing more scenarios.

The remainder of this paper is organized as follows. In Section II, we describe how interactive jobs can be distributed over servers in clouds. In Section III we provide our assumptions about the underlying infrastructure, and the power consumption analysis model we assume in the paper. In Section IV, we provide a robust analysis for energy consumption within the cloud. In particular, we provide two formulas that can be used to obtain optimal number of servers and optimal running frequencies for these servers. In Section V, we apply the equations from our analysis to change management and provide various scenarios to illustrate the power savings one can expect for various cloud environments. Section VI gives our conclusions and future work.

II. Job Distribution Problem

It is important to understand the complexity of the problem of distributing jobs to servers. Actually, this problem can be viewed as a modified instance of the bin packing problem [8] in which n objects of different sizes must be packed into a finite number of bins each with capacity C in a way that minimizes the number of bins used. Similarly, we have n jobs each with different processing requirements; we would like to distribute these jobs into servers with limited processing capacity such that the number of servers used is kept to the minimum. Being an NP-hard problem, there is no fast polynomial time algorithm available to solve the bin packing problem. Next, we attempt to simplify the general problem to a more constrained version for which we can obtain a solution efficiently.

Basically, applications that run in a cloud computing environment can be broadly classified into two different types. The first type includes applications that require intensive processing and usually these are non-interactive applications. The best strategy to run such applications in a cloud environment is to dedicate one or more powerful servers to each of these applications. Obviously, the number of dedicated servers depends on the underlying SLA and the availability of servers in the cloud. These servers should run at their top speed (frequency) so the application can finish as soon as possible. The reason behind this strategy is to allow dedicated servers to be idle for longer periods saving their total energy consumption.

On the other hand, the second type includes applications that heavily depend on user interaction. Web applications and web services are typical examples of this type of applications. Although, in general, interactive applications do not require intensive processing power, they have many clients. If the number of clients for any of these applications is large, then it might be appropriate to run multiple instances of the same application on different servers and balance the load of each server to satisfy the required response time determined by the SLA. Due to the overwhelming number of web based applications available today, it is highly expected to find these applications more common in a cloud computing environment; hence, in this paper we focus on user interactive applications.

As shown below, by focusing on interactive applications, we simplify the problem of distributing jobs into servers. The key idea behind this simplification is to make a job divisible over multiple servers. To clarify this point, we introduce the following example. Assume that, based on its SLA; Job X requires s seconds response time for u users. From the historical data for Job X, we estimate the average processing required for a user query to be I instructions. Assume that job X is to be run on a server that runs on frequency F and on the average requires CPI clock ticks (CPU cycles) to execute an instruction. Within s seconds the server would be able to execute $\frac{s \cdot F}{CPI}$ instructions. Thus, the server can execute $q = \frac{s \cdot F}{I \cdot CPI}$ user queries within s seconds. Basically, if $q < u$, then the remaining $(u - q)$ user requests should be routed to another server. This can be done through the load balancer module at the cloud gateway. When a new job is assigned to the cloud, the job scheduler analyzes the associated SLA and processing requirements of the new job. Based on this information and the availability of servers, the job scheduler module estimates total processing requirements and assigns this job to one or more of the cloud servers.

III. System Model and Assumptions

A cloud consists of a number of server groups; each group has a number of servers that are identical in hardware and software configuration. All the servers in a group are equally capable of running any application within their software configuration. Cloud clients sign a service level agreement SLA with the company running the cloud. In this agreement, each client determines its needs by aggregating the processing needs of its user applications, the expected number of users, and the average response time per user request.

To be able to estimate the computing power (MIPS) needed to achieve the required response time, the client should provide the cloud administrators with any necessary information about the type of the queries expected from its users. One way of doing this is through providing a histogram that shows the frequency of each expected query. Cloud administrators run these queries on testing servers and estimate their computing requirements from their response time. Based on the frequency of each query, cloud administrators can estimate average computing requirement for a user query.

Average response time for a user query depends on many factors, i.e., the nature of the application, the configuration of the server running the application, and the load on the server when running the application. To reduce the number of factors and to simplify our mathematical model, we replace the minimum average response time constraint in SLA by the minimum number of instructions that the application is allowed to execute every second. This kind of conversion can be easily done as follows. If user query has average response time of t_1 seconds when it runs solely on a server configuration with x MIPS (million instructions per second, this can be benchmarked for each server configuration), then to have an average response time of t_2 seconds, it is required to run the query such that it can execute a minimum of $\frac{t_1 \cdot x}{t_2}$ million instructions per second. We assume that each application can be assigned to more than one server to achieve the required response time. When a server runs, it can run on a frequency between F_{min} (the least power consumption) and F_{max} (the

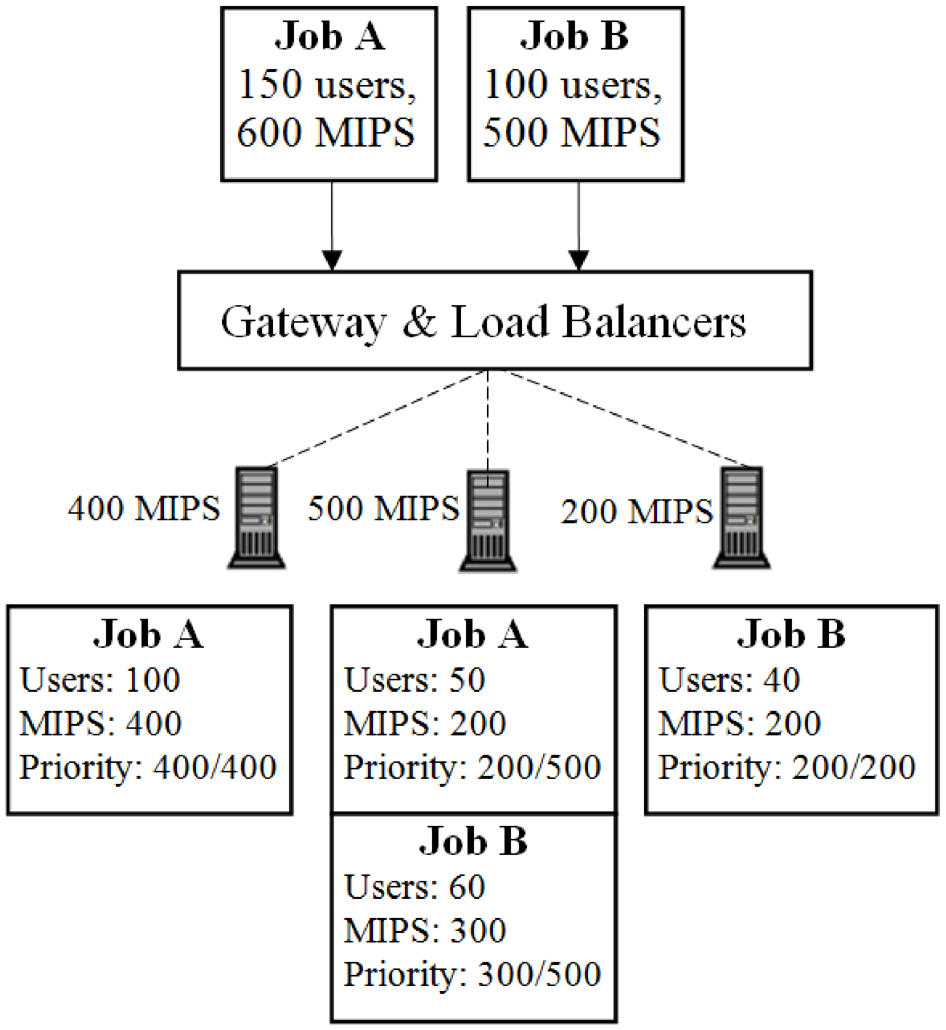


Fig. 1. Distribution of jobs onto servers

highest power consumption), with a range of discrete operating frequency levels in-between.

In general, there are two mechanisms available today for managing the power consumption of these systems: One can temporarily power down the blade, which ensures that no electricity flows to any component of this server. While this can provide the most power savings, the downside is that this blade is not available to serve any requests. Bringing up the machine to serve requests would incur additional costs, in terms of (i) time and energy expended to boot up the machine during which requests cannot be served, and (ii) increased wear-and-tear of components (the disks, in particular) that can reduce the mean-time between failures (MTBF) leading to additional costs for replacements and personnel. Another common option for power management is dynamic voltage/frequency scaling (DVS).

The dynamic power consumed in circuits is proportional to the cubic power of the operating clock frequency. Slowing down the clock allows the scaling down of the supply voltages for the circuits, resulting in power savings. Even though not all server components may be exporting software interfaces to perform

DVS, most CPUs in the server market are starting to allow such dynamic control [5], [6]. The CPU usually consumes the bulk of the power in a server (e.g., an Intel Xeon consumes between 75-100 Watts at full speed, while the other blade components including the disk can add about 15-30 Watts) [5]. The previous example shows that DVS control in cloud computing environment can provide substantial power savings.

When the machine is on, it can operate at a number of discrete frequencies where the relationship between the power consumption and these operating frequencies is of the form [3], [5], [6]

$$P = A + B \cdot F^3$$

so that we capture the cubic relationship with the CPU frequency while still accounting for the power consumption of other components (A) that do not scale with the frequency. In Section V, we shall provide sample values of the constants A and B.

Cloud Environment And Assumptions: for a cloud, requests from cloud clients flow to the system through a cloud gateway. After necessary authentication and based on the current load on the cloud servers, a load balancing module forwards client requests as described in Section II to one of the cloud servers dedicated to support this type of requests. This implies that the load balancing module at the cloud gateway should have up-to-date information about which client applications are running on which servers and the load on these servers. In addition, the system has a 'power-optimizer' module that computes the optimal number of servers and operational frequencies for a particular load requirement. Client applications are assigned to servers based on the requirements of the SLA for each client. This process may involve running the same application on several servers and distributing requests of the same client over different servers based on the load on these servers. To distribute the load on cloud servers correctly, the gateway and the load balancers must have access to the traditional schedule information as well as the information from the power-optimizer.

Homogeneity: in the introduction we described the motivation for using homogeneous sub-clouds that exist within a larger cloud infrastructure. Within each sub-cloud, we assume that resources can be treated homogeneously. That does not mean that all computing devices in a sub-cloud are the same, only that all computing devices in the sub-cloud are capable of executing all work assigned to that sub-Cloud. With the increasing adoption of virtualization technology, including Java JVM and VMware images, we believe that this assumption is valid. For the rest of the paper we shall assume that a cloud is homogeneous

Interactive Applications: Applications that run in a cloud computing environment can be broadly classified into two different types. The first type includes applications that require intensive processing; such applications are typically non-interactive applications. The best strategy to run such applications in a cloud environment is to dedicate one or more powerful servers to each of these applications. Obviously, the number of dedicated servers depends on the underlying SLA and the availability of servers in the cloud. These servers should be run at their top speed (frequency) so the application will finish as soon as possible. The reason behind this strategy is to allow dedicated servers to be idle for longer periods saving their total energy consumption.

The second application type is those that depends heavily on user interaction. Web applications and web services are typical examples. Although, in general, interactive applications do not require intensive processing power, they have many clients, leading to a large aggregate processing demand. If the number of clients for any of these applications is large, to satisfy the required response time determined by the SLA, it might be appropriate to run multiple instances of the same application on different servers, balancing the load among them.. Due to the overwhelming number of web based applications available today, such applications are likely to be prevalent in a cloud computing environment; hence, in this paper we focus on user interactive applications. We leave analysis of the former application type to future work.

Power consumption in our model will be manipulated by changing the frequencies at which instructions are executed at a server. As SLAs are typically expressed in many different ways we need to map these compute requirements into a standard form that relates to the number of instructions executed over a period of time. We chose to represent the load an application will put on the cloud in terms of the familiar MIPS. For example, in Fig. 1 we show how a particular client of the cloud has at that time 150 users who require a total of 500 MIPS for the next period of time.

To estimate the computing power (MIPS) needed to achieve the required response time, the client must provide the cloud administrators with any necessary information about the type of the queries expected from its users. One approach is to provide a histogram that shows the frequency of each expected query. Cloud administrators run these queries on testing servers and estimate their computing requirements from their response time. Based on the frequency of each query, cloud administrators can estimate average computing requirement for a user query.

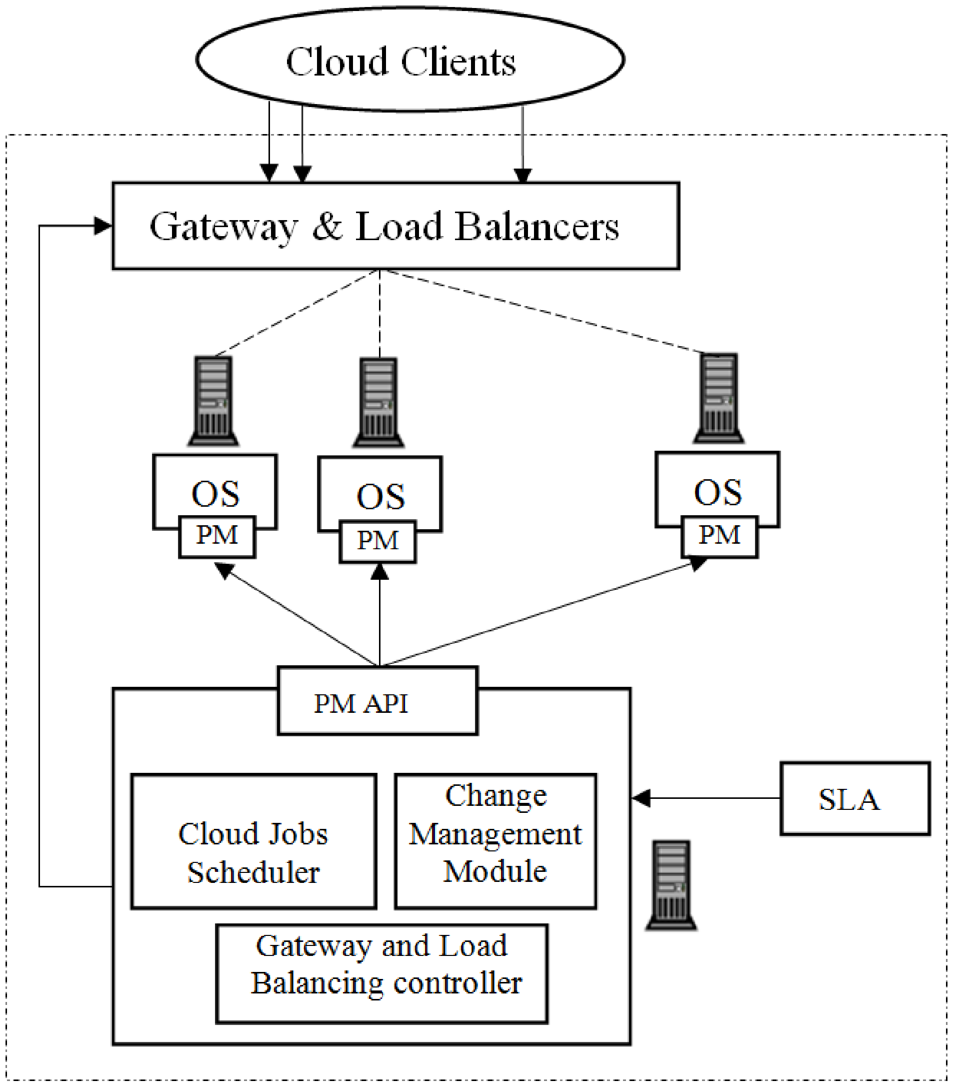


Fig. 2. Cloud Architecture

IV. Power Consumption

To summarize the model assumptions: a cloud consists of a number of server groups; each group has a number of servers that are identical in hardware and software configuration. All the servers in a group are equally capable of running any application within their software configuration. Cloud clients sign a service level agreement SLA with the company running the cloud. In this agreement, each client determines its needs by aggregating the processing needs of its user applications, the expected number of users, and the average response time per user request. When a server runs, it can run on a frequency between (the least power consumption) and (the highest power consumption), with a range of discrete operating frequency levels in-between. In general, there are two mechanisms available today for managing the power consumption of these systems: One can temporarily power down the blade, which ensures that no electricity flows to any component of this server.

While this can provide the most power savings, the downside is that this blade is not available to serve any requests. Bringing up the machine to serve requests would incur additional costs, in terms of (i) time and energy expended to boot up the machine during which requests cannot be served, and (ii) increased wear-and-tear of components (the disks, in particular) that can reduce the mean-time between failures (MTBF) leading to additional costs for replacements and personnel. Another common option for power management is dynamic voltage/frequency scaling (DVS). The dynamic power consumed in circuits is proportional to the cubic power of the operating clock frequency. Slowing down the clock allows the scaling down of the supply voltages for the circuits, resulting in power savings. Even though not all server components may be exporting software interfaces to perform DVS, most CPUs in the server market are starting to allow such dynamic control [5], [6]. The CPU usually consumes the bulk of the power in a server (e.g., an Intel Xeon consumes between 75-100 Watts at full speed, while the other blade components including the disk can add about 15-30 Watts) [5]. The previous example shows that DVS control in cloud computing environment can provide substantial power savings.

When the machine is on, it can operate at a number of discrete frequencies where the relationship between the power consumption and these operating frequencies is of the form [5], [6]

So that we capture the cubic relationship with the CPU frequency while still accounting for the power consumption of other components (A) that do not scale with frequency.

Let $F(t)$ be the total computing load of the cloud at time t. To provide the required computing load, the cloud has k servers that run on frequencies $F_1, F_2, \cdots, F_k$ respectively. We normalize frequencies in the range $[0, 1]$ using,

$$f_i = \frac{F_i - F_{min}}{F_{max} - F_{min}} \tag{1}$$

Define,

$$L = \sum_{i=1}^{k} f_i \tag{2}$$

where f_i is the normalized frequency on which server i runs. Assuming that the average clocks per instruction for the cloud servers is CPI, we can relate normalized frequencies to cloud total load as follows.

$$\begin{aligned}
L &= \sum_{i=1}^{k} f_i \\
L &= \sum_{i=1}^{k} \frac{F_i - F_{min}}{F_{max} - F_{min}} \\
\sum_{i=1}^{k} F_i &= (F_{max} - F_{min})L + k \cdot F_{min} \\
\frac{\sum_{i=1}^{k} F_i}{CPI} &= \frac{(F_{max} - F_{min})L + k \cdot F_{min}}{CPI} = F(t) \qquad (3)
\end{aligned}$$

and from equation (3), we can express L as follows,

$$\begin{aligned}
F(t) &= \frac{(F_{max} - F_{min})L + k \cdot F_{min}}{CPI} \\
L &= \frac{F(t) \cdot CPI - k \cdot F_{min}}{F_{max} - F_{min}} \\
&= C_1 + C_2 \cdot k \qquad (4)
\end{aligned}$$

where

$$\begin{aligned}
C_1 &= \frac{F(t) \cdot CPI}{F_{max} - F_{min}} \qquad (5) \\
C_2 &= -\frac{F_{min}}{F_{max} - F_{min}} \qquad (6)
\end{aligned}$$

Total energy consumption is given by,

$$\begin{aligned}
P &= \sum_{i=1}^{k} \left[A + B \cdot f_i^3\right] \\
&= k \cdot A + B \sum_{i=1}^{k-1} f_i^3 + B \left[L - \sum_{i=1}^{k-1} f_i\right]^3
\end{aligned}$$

Now our interest is to evaluate the optimal values for f_i such that the total energy consumption is minimum. Assuming a continuous frequency spectrum, we evaluate the first partial derivative of the total energy consumption with respect to each frequency.

$$\frac{\partial P}{\partial f_i} = 3Bf_i^2 - 3B \left[L - \sum_{i=1}^{k-1} f_i\right]^2 \qquad \forall i \in \{1, \cdots, k-1\}$$

To minimize P, we set $\frac{\partial P}{\partial f_i} = 0$.

$$\begin{aligned}
\frac{\partial P}{\partial f_i} &= 3Bf_i^2 - 3B\left[L - \sum_{i=1}^{k-1} f_i\right]^2 = 0 \\
3Bf_i^2 &= 3B\left[L - \sum_{i=1}^{k-1} f_i\right]^2 \\
f_i &= \left[L - \sum_{i=1}^{k-1} f_i\right] = f_k \qquad \forall i \in \{1, \cdots, k-1\} \\
f_i &= \frac{L}{k} \qquad \forall i \in \{1, \cdots, k\} \qquad (7)
\end{aligned}$$

In other words, to minimize the total energy consumption, cloud servers must run at frequency $\frac{L}{k}$.

Now we turn our interest to evaluate, k, the optimal number of servers to run. Using a similar approach, we first rewrite the equation of total energy consumption after substitution each frequency using equation (7) as follows.

$$\begin{aligned}
P &= \sum_{i=1}^{k} \left[A + B \cdot f_i^3\right] \\
&= k \cdot A + k \cdot B \left[\frac{L}{k}\right]^3 \\
&= k \cdot A + \frac{B(C_1 + C_2 k)^3}{k^2} \qquad (8)
\end{aligned}$$

After that we obtain the first derivative of the total energy consumption with respect to k. Mathematically, this can be expressed as,

$$\frac{\partial P}{\partial k} = A + \frac{3BC_2(C_1 + C_2k)^2k^2 - 2B(C_1 + C_2k)^3k}{k^4}$$

Setting $\frac{\partial P}{\partial k} = 0$,

$$\begin{aligned}
Ak^3 &= 2B(C_1 + C_2k)^3 - 3BC_2(C_1 + C_2k)^2k \\
\frac{Ak^3}{B} &= (C_1 + C_2k)^2(2(C_1 + C_2k) - 3C_2k) \\
\frac{Ak^3}{B} &= (C_1 + C_2k)^2(2C_1 - C_2k) \\
\frac{Ak^3}{B} &= (C_1^2 + 2C_1C_2k + C_2^2k^2)(2C_1 - C_2k) \\
\frac{Ak^3}{B} &= 2C_1^3 + 4C_1^2C_2k + 2C_1C_2^2k^2 - C_1^2C_2k \\
&\quad -2C_1C_2^2k^2 - C_2^3k^3
\end{aligned}$$

After rearranging terms,

$$\begin{aligned}
\left[\frac{A}{B} + C_2^3\right]k^3 - 3C_1^2C_2k &= 2C_1^3 \\
k^3 - \frac{3C_1^2C_2}{\frac{A}{B} + C_2^3}k &= \frac{2C_1^3}{\frac{A}{B} + C_2^3} \\
k^3 + pk &= q \qquad (9)
\end{aligned}$$

where

$$p = -\frac{3C_1^2C_2}{\frac{A}{B} + C_2^3} \qquad (10)$$

$$q = \frac{2C_1^3}{\frac{A}{B} + C_2^3} \qquad (11)$$

To solve equation(9), we can use Vieta's substitution [9] by defining x as follows,

$$k = x - \frac{p}{3x} \qquad (12)$$

After substituting k into equation (9),

$$\begin{aligned}
\left[x - \frac{p}{3x}\right]^3 + p(x - \frac{p}{3x}) &= q \\
x^3 - px + \frac{p^2}{3x} - \left[\frac{p}{3x}\right]^3 + px - \frac{p^2}{3x} &= q \\
x^3 - \left[\frac{p}{3x}\right]^3 &= q \qquad (13)
\end{aligned}$$

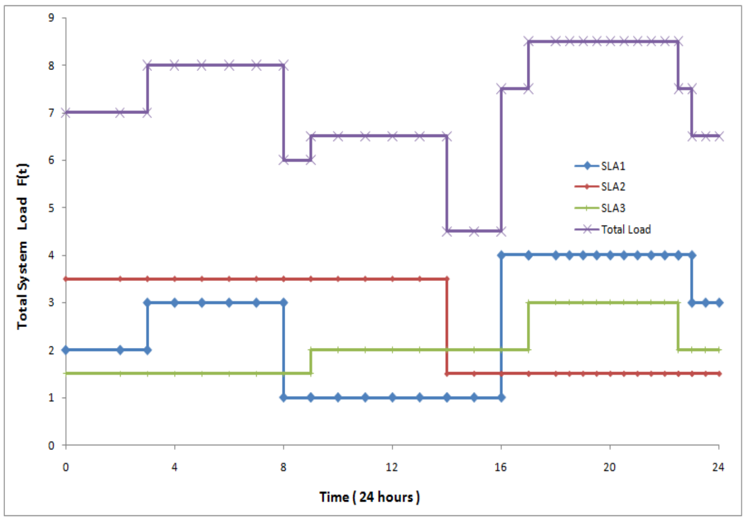
Multiplying both sides of equation (13) by x^3 converts it into the standard quadratic form.

$$\left(x^3\right)^2 - q(x^3) - \frac{p^3}{27} = 0 \qquad (14)$$

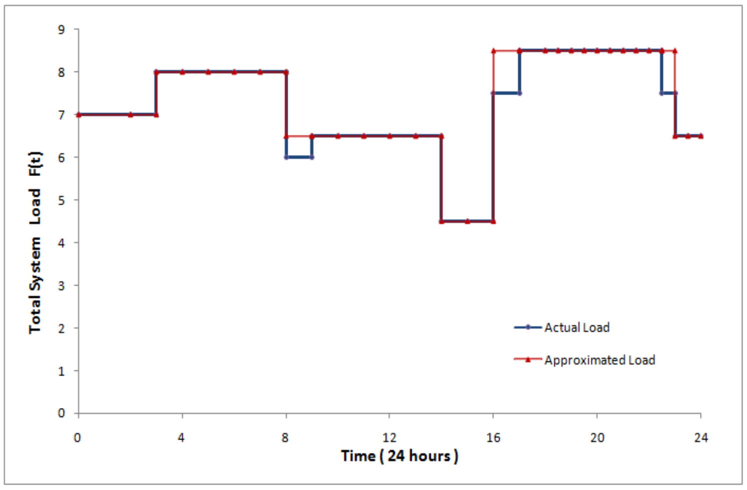
Solving equation (14),

$$\begin{aligned}
x^3 &= \frac{q + \sqrt{q^2 + \frac{4p^3}{27}}}{2} \\
&= \frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \frac{p^3}{27}} \\
&= \frac{C_1^3}{\frac{A}{B} + C_2^3} + \sqrt{\frac{\left(\frac{A}{B} + C_2^3\right)C_1^6 - C_1^6C_2^3}{\left(\frac{A}{B} + C_2^3\right)^3}} \\
x^3 &= \frac{C_1^3}{\frac{A}{B} + C_2^3} + C_1^3\sqrt{\frac{\frac{A}{B}}{\left(\frac{A}{B} + C_2^3\right)^3}} \\
x &= C_1 \cdot \sqrt[3]{\frac{1}{\frac{A}{B} + C_2^3} + \sqrt{\frac{\frac{A}{B}}{\left(\frac{A}{B} + C_2^3\right)^3}}} \qquad (15)
\end{aligned}$$

From equations (12) and (15), we can evaluate k, the optimal number of servers, that should by running in the cloud to optimize power consumption. Interestingly, the value of k depends on the constants A and B, the frequency range of server processors (i.e., F_{min}, F_{max}), and the total computing load $F(t)$.



(a)



(b)

Fig. 3. (a) Cloud total load as imposed by SLAs (b) Actual vs. Approximated total load due to several SLAs

V. Change Management Scenarios

The computing load of the cloud can be expressed as a function of time, and it usually changes when a new application is started or completed. Typically, the sum of the commitments in service level agreements is periodic with a daily, weekly or even monthly period. To include the dynamic nature of the cloud load into our model, we divide the time line into slots. During one slot, the cloud total load does not change. To eliminate minor changes in the total cloud load curve, we approximate the load on the cloud by an upper bound envelope of this curve such that the length of any slot is larger than a predetermined threshold.

The derivations in Section IV for the optimal number of servers and optimal running frequencies should not change for this simplification. All we need to do is to replace L by $L(t)$. Thus, in each time segment, the number of idle servers in the cloud equals the difference between the total number of cloud servers and $k(t)$. An idle server is a candidate for change management. Figure 4, shows a plot for the number of servers available for change management

based on the cloud load determined in Figure 3-a.

We define change management capacity as the average number of servers available for changes per time unit. Based on this definition, change management capacity can be evaluated as the area under the curve of Figure 4 divided by the periodicity of the SLAs. For example, for the cloud load shown in Figure 3-b, change management capacity as follows,

$$\begin{aligned}\textbf{Capacity} &= \frac{3\cdot 5+5\cdot 4+6\cdot 5+2\cdot 7+7\cdot 3+1\cdot 5}{24}\\ &= \frac{75}{24} = 3.125 \textbf{ servers/hour.}\end{aligned}$$

From historical and statistical change management information about the applications running on the cloud, administrators can estimate change management capacity requirements for these applications. This can be used to determine the optimal number of servers to have in the cloud for a particular set of clients. Particularly, the area under the curve in Figure 4 is proportional to the number of available servers for maintenance in the cloud. Basically, this area can be adjusted by raising the curve up or down, which in turn can be done by changing the total number of servers in the cloud. For example, in Figure 4, we estimated cloud change management capacity to be 3.125 servers/unit time. If statistics showed that applications running on the cloud require an average of 4 servers to be available for changes per unit time, then the total number of servers in the cloud should be increased by one server to satisfy change management requirements.

It is worthwhile to mention that under our proposed model it is straightforward to incorporate hardware failures into the model, increasing the reliability of the cloud. Hardware failure rates can be statistically estimated using information about previous hardware failures, expected recovery rates, hardware replacement strategies, and expected lifetime of the hardware equipments. Given hardware failure rate expressed in terms of failed servers per unit time, cloud applications change requirements can be adjusted to reflect hardware failures. The new change management capacity is estimated based on the sum of application changes requirements and hardware failure requirements. In the previous example of Figure 3-b, if the hardware failures rate is less than 0.875 failed servers/hr, then an average of 4 servers available per unit time is enough to satisfy change and hardware failure requirements. However, if hardware failures rate goes above 0.875 servers/hr, then additional servers are needed.

In this section, we compare the performance of our proposed change management technique against other techniques based on over-provisioning. The main idea behind these techniques is to overly provision computing resources to compensate for any failure or change management requirements. Our calculations for over-provisioning techniques assumes a 5% over-provisioning rate, which means that the available computing power available at any time is 5% higher than what is needed to satisfy the service level agreements.

In our scenario, we assume a computing cloud with 125 servers. Each server has a range of discrete running between $1-3$ GHz. Maximum processing power is achieved when the server runs at 3 GHz. We assume that if the required running frequency is unavailable, the next higher available frequency will be used. To numerically correlate the running frequency with the achieved computing power, we must estimate the average number of cycles needed to execute one instruction on any of these servers. Given running frequency, F, and number of cycles per instruction, CPI the computing power of a server can be estimated in MIPS, million instructions per second, as $\frac{F}{CPI\cdot 10^6}$. In this scenario, we set CPI to 3.00 cycles/instructions. To be able to measure energy consumption, we assume the energy model described in Section III $(i.e., P = A + BF_n^3)$ where A and B are system constants, is the normalized running frequency. In our scenario, we use the same values of the constants A and B as was published in [6]. This also requires to normalize the running frequency to the range [0,1], where stands for the minimum running frequency (1.0 GHZ), and stands for the maximum running frequency. Mathematically this can be obtained through, $F_n = \frac{F-F_{min}}{F_{max}-F_{min}}$.

We assume that the cloud administrators have determined that they will need a change management capacity of 1.2 servers/hr. For a more realistic scenario, we include also server failures in our model. We assume a failure rate of 0.6 servers per hour, which is included as an additional computing requirement. In terms of periodic load, here we assume a period of one day.

During the day, the cloud total load changes as described in Table I. To remind the reader, this information is obtained from client applications historical data and is expressed in SLAs.

Under these assumptions, we compare total energy consumption using our proposed approach against using a 5% over-provisioning. Figure 7 shows how using our approach can reduce cloud energy consumption against over-provisioning. Both approaches can achieve the required level of change management capacity.

Figures 5 and 6 respectively show the total energy consumption and the associated number of servers needed for different frequency ranges. From the figures, it is easy to verify that the optimal running

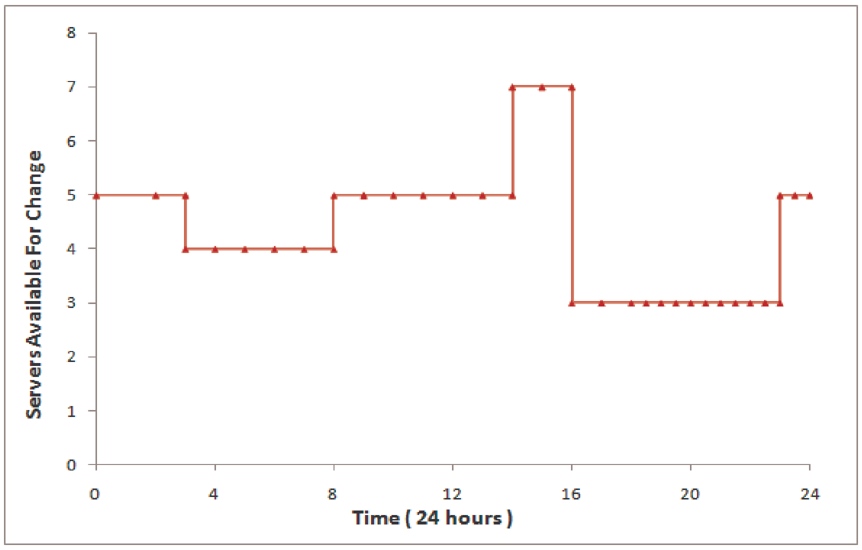


Fig. 4. Servers available for changes as a function of time

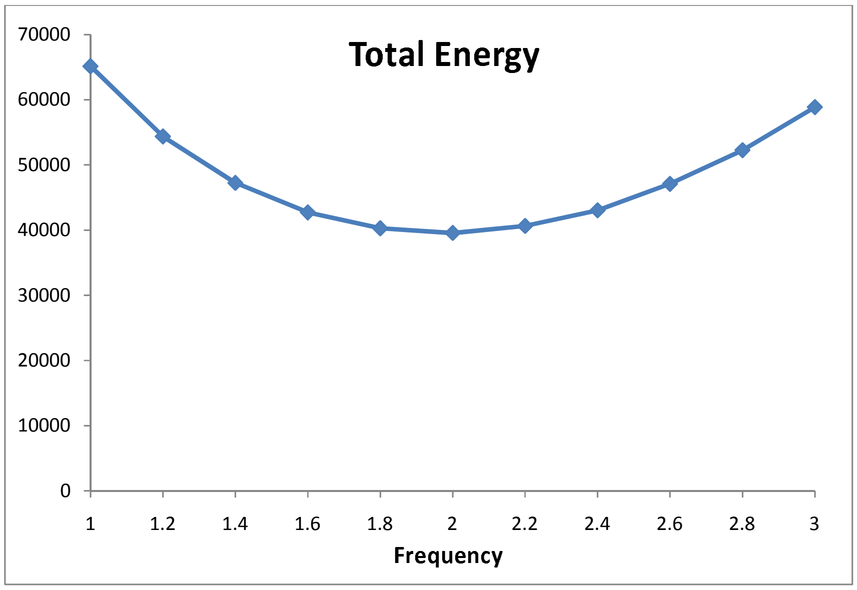


Fig. 5. Total energy consumption when using different frequencies

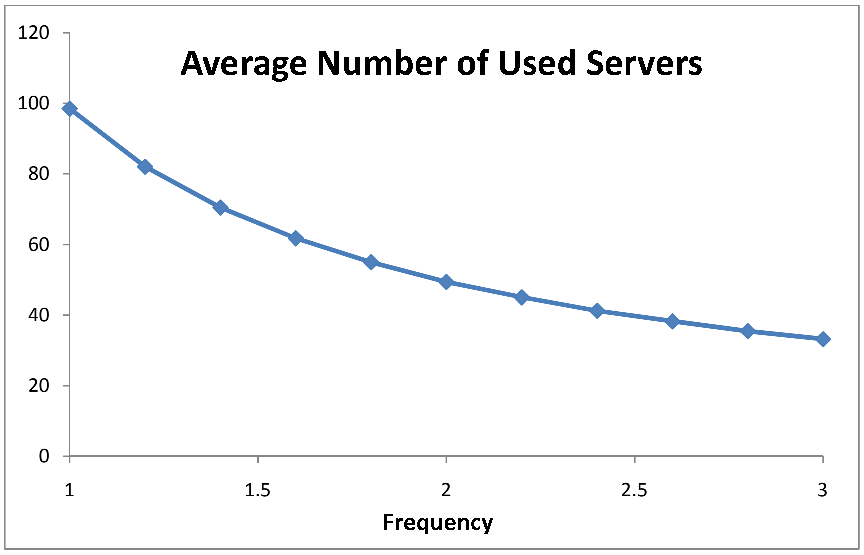


Fig. 6. Average number of servers when using different frequencies

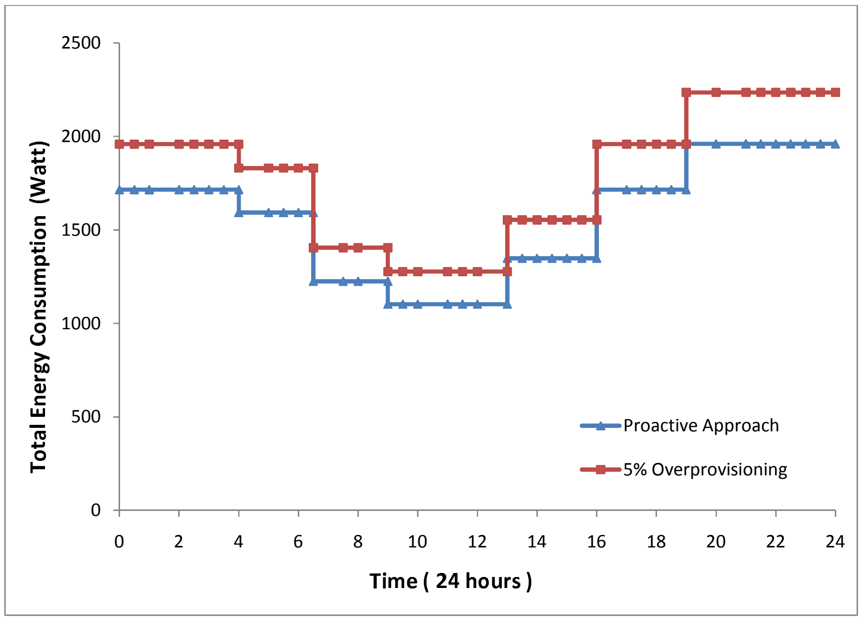


Fig. 7. Proactive approach vs. 5% Over-Provisioning

TABLE I
CLOUD LOAD DURING DIFFERENT TIMES OF THE DAY

From	To	Cloud Load (BIPS)
12:00 AM	04:00 AM	70
04:00 AM	06:30 AM	65
06:30 AM	09:00 AM	50
09:00 AM	01:00 PM	45
01:00 PM	04:00 PM	55
04:00 PM	07:00 PM	70
07:00 PM	12:00 AM	80

TABLE II
TOTAL ENERGY CONSUMPTION IN THE CLOUD DURING ONE DAY ASSUMING 5% PROVISIONING

Frequency	Total(Watt.Hour)	Average(Watt)
1.0 GHZ	64861	2703
2.0 GHZ	39325	1639
2.4 GHZ	42743	1781
3.0 GHZ	58246	2427

frequency is around 2 GHZ as determined by our proposed formula.

We also compare the total energy consumption during one period (one day) using both approaches. Under the proactive approach, the total energy consumption is evaluated to be 37304.76 Watt.Hour, for an average of 1554.36 Watt. Table II, summarizes the total and the average energy consumption when using 5% over-provisioning. Table II shows that using the proactive approach, cloud total energy consumption is smaller than energy consumption using 5% over-provisioning for different running frequencies.

VI. CONCLUSIONS

In this paper we have created a mathematical model for power management for a cloud computing environment that primarily serves clients with interactive applications such as web services. Our mathematical model allows us to compute the optimal (in terms of total power consumption) number of servers and the frequencies at which they should run. We show how the model can be extended to include the needs for change managements and how the other type of typical applications (computing intensive) can be included. Further, we extend the model to account for hardware failure. In Section V, we compare our scheme against various over-provisioning scheme For example, with a cloud of 125 servers, a change management capacity of 1.2 servers/hr, a failure rate of 0.6 servers/hr, the total power consumption for a day with our scheme is $37,305$ Watt.Hour versus $64,861$ Watt.Hour at 5% over-provisioning, and the savings range from $5-74\%$ for various parameters of the cloud environment. In the future we plan to relax some of the model's simplifying assumptions. In particular we can rather straightforwardly adapt the model to have the frequencies assume discrete values rather than be part of a continuous function.

REFERENCES

[1] H. AbdelSalam, K. Maly, R. Mukkamala, and M. Zubair. Infrastructure-aware autonomic manager for change management. In POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, pages 66–69, Washington, DC, USA, 2007. IEEE Computer Society.

[2] H. AbdelSalam, K. Maly, R. Mukkamala, M. Zubair, and D. Kaminsky. Scheduling-capable autonomic manager for policy based it change management system. In EDOC: Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference, pages 386–392, München, Germany., September 15-19 2008. IEEE Computer Society.

[3] H. S. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, and D. Kaminsky. Analysis of energy efficiency in clouds. In COMPUTATIONWORLD '09: Proceedings of the 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pages 416–421, Washington, DC, USA, 2009. IEEE Computer Society.

[4] G. Boss, P. Malladi, D. Quan, L. Legregni, and H. Hall. Cloud computing. High Performance On Demand Solutions (HiPODS), Web Resource available online at: http://www.ibm.com/developerworks/websphere/zones/hipods/, October 2007. IBM.

[5] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pages 303–314, New York, NY, USA, 2005. ACM.

[6] M. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In PACS'02: Proceedings of the 2nd Workshop on Power-Aware Computing Systems, in conjunction with HPCA-8., pages 179–196, Cambridge, Ma, USA, February 2002.

[7] H. Erdogmus. Cloud computing: Does nirvana hide behind the nebula? IEEE Software, 26(2):4–6, March–April 2009.

[8] M. R. Garey and D. S. Johnson. Computers and Intractability; A Guide to the Theory of NP-Completeness, pages 10–20. W. H. Freeman & Co., New York, NY, USA, 1990.

[9] E. W. Weisstein. Vieta's substitution. From MathWorld – A Wolfram Web Resource available online at: http://mathworld.wolfram.com/VietasSubstitution.html, June 2010.