A Novel Power Model and Completion Time Model for Virtualized Environments

Swetha P.T. Srinivasan and Umesh Bellur
Dept. of Computer Science and Engineering
Indian Institute of Technology, Bombay
{swethapts,umesh}@cse.iitb.ac.in

ABSTRACT
As power consumption costs take up half of the operational expenses of a data center, power management is now a critical concern of high performance computing and cloud computing data centers alike. Recent advances in processor technology have provided fine-grained control over the frequency and voltage at which processors operate and increased the dynamic power range of modern computers thus enabling power-performance negotiations. At the same time, the trends towards virtualization of data centers and HPC environments give us a clear boundary for control and helps us map virtual machines (VM) to specific cores of a physical machine. This control coupled with the ability to set the operating frequency of the cores gives us the opportunity to play with the power-performance tradeoffs of the VMs in a relatively simple manner. The complexity in effective power management arises from two sources - being able to predict the power consumption at a given performance level accurately and then to use this prediction in VM provisioning. Although many power and performance models exist, the combined effect of frequency and compute resource variations on the power and performance of VMs are not analyzed thoroughly. Moreover, the models do not perform well for memory or I/O-intensive tasks.

In this paper, we present an empirically derived power model and a completion time model using linear regression with CPU utilization and operating frequency of the server as parameters. We have validated the power model by using several processors in the Intel 'i'-series, the Xeon series as well as the AMD processor in the x4 series and shown that our model predicts power consumption within a range of 2-7% of the measured power. We have also validated the completion time model by predicting the execution time of six CPU, memory and I/O-intensive benchmarks on four heterogeneous systems and our model predicts the completion time within a range of 1-6% of the observed execution time. We are now in the process of employing these models to control VM provisioning to minimize power consumption while at the same time meeting the service level agreements of the applications executing within VMs. Our work applies to any virtualized environment - from a Hadoop running on a HPC cluster to enterprise applications in Cloud data centers.

Categories and Subject Descriptors
C.0 [GENERAL]: Modeling of Computer Architecture; D.4.8 [OPERATING SYSTEMS]: Performance—Modeling and prediction

General Terms
Experimentation, Measurement, Performance

Keywords
Power, Completion Time, Virtualization, Modeling, Prediction

1. INTRODUCTION
Data centers today are equipped primarily with multicore machines which are then virtualized to deliver virtual machines (VMs) of different sizes. Virtualization has also brought in the capability to use the underutilized servers more efficiently. However, power consumption still remains the greatest concern of data center administrators taking 30-50% of the operational costs [24]. Processor manufacturers such as Intel and AMD have introduced products such as AMD PowerNow!, AMD Cool’n’Quiet and Intel SpeedStep that incorporate Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management such as clock gating. Intel Nehalem architecture included per-core Dynamic Frequency Scaling that enables each core with a separate Digital Phase-Locked Loop (DPLL) [18] for clock signal generation rather than using clock gating. Also, finer-grained processor frequency steps, operating points and sleep states have considerably reduced the idle power consumption and greatly expanded the dynamic power range of the processors. With the use of these hardware techniques along with OS-level configurations (CPU governors), the idle power consumed is as low as 38% of the peak power as shown in Figure 1, unlike the earlier architectures that drew about 70% of the peak power [14].

The hypervisors can vary the frequencies of the processor cores the VMs are scheduled on as well as the CPU allocated to individual VMs. These controls can significantly impact both the power drawn by the system and the performance achieved by the applications executing inside the
VMs. Our ultimate goal is to provision VMs while satisfying the twin (and possibly competing) requirements of a power budget and VM performance. As an intermediate step, with a power and completion time model in place, we could optimize for one or the other of these requirements as well and use the resulting model to understand the effect that maximizing performance will have on power or minimizing power consumption will have on the performance. Figure 2 shows the approach we will take towards VM provisioning.

Power consumed by a system depends on the frequency at which the processor operates as well as the CPU utilization itself, as shown in Figure 1. Power varies by as much as 62% across CPU usage and 45% across the highest and the lowest frequencies. A thorough survey of existing literature on power models raised two issues with respect to applicability on modern processors.

- Most models either focused on measuring the power consumed at the component-level using voltage and frequency supplied to the processor, or modeled power as a function of CPU and other resource utilizations but not both.
- The models that consider CPU and frequency as parameters fail to take into account the wide range of frequency settings offered and the kind of frequency scaling (clock gating vs DPLL).

Traditionally, the power consumed is inversely proportional and is thus traded off with application performance which in a data center directly affects the SLAs guaranteed by a service provider. There is a vast collection of literature that quantifies the effect of either the CPU allocation or the processor frequency on the performance of benchmarks. The review of power-aware performance models for virtualized applications evoked two points to focus on.

- The combined effect of CPU and frequency changes on the performance of applications are not studied and analyzed thoroughly.
- While it has been established that frequency changes can be simulated by varying the CPU allocation, other-resource-intensive applications are less susceptible to performance loss due to frequency fluctuations. There is a need to identify the CPU resizing factor for such applications.

The rest of the paper is organized as follows. Section 2 describes the background on the existing hardware-level power management techniques applicable to data centers. Existing power and performance models are presented in Section 3. Section 4 provides the methodology, experimental setup and identifies the parameters needed for predicting the power of a virtualized server. Section 5 derives a power model with CPU% and operating frequency as parameters and validates the model across heterogeneous servers. Section 6 identifies compute resource and server frequency as the input parameters for predicting the performance of a task. Section 7 derives a completion time model from existing work and validates the model across 4 systems and 6 benchmarks. Conclusions and future work are presented in Section 8.

### 2. BACKGROUND

The power consumption of a system is the rate at which the system performs work and has two components: 

$$ P_{\text{total}} = P_{\text{idle}} + P_{\text{dynamic}} \quad (\text{watts}). $$

$P_{\text{idle}}$ is the minimum power that is required by the system to remain active, irrespective of clock rate or usage. $P_{\text{dynamic}}$ is the power consumed while performing computations. $P_{\text{dynamic}}$ varies with clock rate, voltage supplied or utilization of the system. The dynamic power range of a system is defined as the ratio of the difference between peak power and idle power to the peak power. Energy consumption is total work done by system for a time
duration i.e., \( E = \sum_{i=1}^{n} P(i) \) and is measured in watt-hours or joules. Data centers which use modern processors could achieve higher power conservation by understanding and effectively using the Hardware-assisted techniques currently offered.

### 2.1 Hardware-assisted Power Management

Benini et al. [9] classified power management of processors as (1) supply shutdown; (2) clock gating and (3) multiple and variable power supplies for individual components. Modern processors offer dynamic frequency scaling (DFS) either by clock gating or using separate clock signals for each core. DVFS is a powerful dynamic power management technique for conserving the processor power by reducing the voltage and the frequency depending on the CPU-resource utilization. The dynamic power of the processor is given by \( P_{\text{dynamic}} = V^2f \) where \( V \) is the supply voltage and \( f \) is the frequency across the processor. A change in the voltage will have a greater impact on the power drawn than a change of the frequency.

Single-core processors were initially designed with off-chip voltage regulators which caused tens of milliseconds delay during DVFS [32]. The voltage regulators were placed off the chip due to their bulky size and space restrictions of the processor. Global on-chip regulators were designed to moderate the voltage supplied to multicore processors. The voltage is set based on the core operating at the highest frequency. Frequency scaling is done by scaling the clock length, which enables the required threshold voltage to be set. Whereas, the frequencies of the remaining cores are scaled using clock gating i.e., stopping the cores for some cycles [31]. The power conservation through clock gating occurs due to lesser number of instructions issued and not because of voltage scaling. Commercial processors such as Intel Nehalem currently use a DPLL to scale the frequency of individual cores. However, the voltage is still set based on the highest frequency. CPUgovernor has been incorporated into the linux kernel to provide a variety of frequency profiles to the users i.e., the governor chooses which frequency to set based on the CPU utilization and a set of selection policies. The five profiles that are offered are - Conservative, Ondemand, Performance, Powersave and Userspace. The current version of the CPUgovernors used by the hypervisors consider none of the following - the performance achieved by the applications executing inside the VMs, the required performance i.e., SLA or design of the processor.

In this paper, we aim to understand the effect of DVFS technique on the power consumption of the machines and the performance attained by the tasks executing inside the VMs, in order to achieve higher power conservation without SLA violations for VM provisioning.

### 3. RELATED WORK

Power Management has become an essential part of data center operations. While the idle power drawn by the servers and other equipments depend on the material and technology used for fabrication, data centers try to exploit the variation in the dynamic power consumed by the servers. Virtualization has aided in power conservation, as it enabled higher utilization using fewer number of servers. There is scope for further improvement. This section presents the existing power models for modern multicore processors and the available performance model of virtualized applications.

#### 3.1 Power Models

Modeling the power consumption of a system is an essential phase in efficient power management. Earliest works in this area used the processor’s power consumption as a proxy for modeling power drawn by the system. The CMOS circuits that are used for building the processors derive dynamic power as \( P_{\text{dynamic}} = aCV^2f \) where \( a \) is the switching activity, \( C \) is the capacitance, \( V \) is the supply voltage and \( f \) is the frequency or rate of clock signal of the processor. Fan et al. [13] established a linear relationship between power consumption and CPU utilization as

\[
P = C_0 + C_1u
\]

where \( u \) is the fraction of CPU utilization. The authors also proved empirically that the prediction rate of CPU intensive applications follow a non-linear relationship:

\[
P = C_0 + C_1u + C_2u^r
\]

where \( r \) is determined empirically for a system. Belliosa et al. [8] used performance monitoring counters to model the power consumption of the processor. The authors experimented on Intel Pentium II processor and observed a linear relationship between the power consumed and the number of integer micro-operation retired, floating point operations, second-level cache references and main-memory references. The above models focused on CPU utilization as processes consumed about 70% of the total power supplied to the system. However, these models did not hold for non-CPU-intensive applications [25]. Gurumurthi et al. [15] designed SoftWatt simulator to correlate power consumption with utilization of four resources, namely - CPU, memory, I/O and disk. Though the simulation method aids in analyzing the power drawn by individual components, it is empirically infeasible as the modern day processors are made of millions of such components.

Instead of components, Isci and Martonosi [16] built a power model for 22 subunits of Intel Pentium 4 processor using 18 Performance Monitoring Counters (PMC) and linear programming. Bertran et al. [10] used the multicore power model proposed by Singh et al. [26] and identified 41 subsystem components on Intel Core 2 Duo processor and used PMC for predicting of the power drawn, with an overall accuracy of 83%. Pedram and Hwang proposed WorkloadGen [22], a workload generator to model the dynamic power management technique - DVFS. They established that power is linearly proportional to CPU utilization even at high CPU frequency for a 8-core - dual chip, dual socket, dual-core Intel Xeon 5400 sever. This model was based on two assumptions - (i) only two frequencies available - 'high' and 'low'. (ii) if a core is at the 'low' frequency, that core is idle i.e., no process is scheduled on that core. However, there are commercial multicore processors that support a wide range of frequencies to choose from and the cores may be in use even at lower frequencies.

Petrucci et al. [23] addressed this concern when they proposed a power model that support multiple frequency steps. The power \( p_{ij} \) at any given utilization \( u_{ij} \) is given as

\[
p_{ij}(u_{ij}) = p_{m_{ij}} + (p_{M_{ij}} - p_{m_{ij}}) \cdot u_{ij}
\]

\[
p_{m_{ij}} = p_{m_{1i}} + ((p_{M_{ij}} - p_{m_{1i}}) \cdot \frac{(f_{ij} - f_{0i})^2}{(f_{ij} - f_{0i})^2})
\]
\[ PM_{ij} = PM_{i1} + (PM_{F_i} - PM_{i1}) \left( \frac{f_{ij} - f_{i1}}{f_{F_i} - f_{i1}} \right)^2 \]

where \( PM_{i1} \) and \( PM_{F_i} \) are the idle power at maximum and minimum frequencies i.e., \( f_{i1} \) and \( f_{F_i} \) respectively. \( PM_{F_i} \) and \( PM_{i1} \) are the peak power at \( f_{F_i} \) and \( f_{i1} \) respectively. It is to be noted that power is linearly proportional to the frequency or cubically proportional as voltage of the processor is set based on the frequency. Petrucci et al. [23], however assumed a quadratic relationship between power and frequency. In this paper, we empirically establish power's linear dependency on the frequency and since our work is closely related to [23], we show how our model predicts power consumption more accurately than that given in [23]. As highlighted earlier in Figure 1, the challenge lies in modeling power based on the current CPU utilization and the operating frequency.

### 3.2 Power-aware Performance Models

Cardosa et al. [11] and Laszewski et al. [28] have proposed power-performance optimization algorithms with the assumption that the operating point or the frequency required by the VMs is known apriori. Nathuji and Schwan, in VirtualPower [21], implemented soft-scaling and hard-scaling on Xen hypervisor based on the same assumption. ‘Soft Scaling’ - a technique where VM will execute at the required frequency using CPU scheduling policy rather than hypervisor changing the frequency of the processor. Many other authors such as Kamga et al. [17] and Wen et al. [30] proportionally reallocated CPU to simulate frequency changes. Kamga et al. [17] derived the new CPU cap based on the increment in the relative frequency. While proportionally reallocating CPU maintains the power for CPU-intensive applications, Dhiman et al. [12] and Marinoni and Buttazzo [20] experimentally verified that frequency changes have lesser effects on memory-intensive applications and minimally affect network- and disk-intensive applications. Wang and Wang [29] used Model Predictive Control (MPC) theory to design the Controller that changes the CPU allocation of the VM and the frequency of the servers based on a power cap. They modeled the change in the response time at time \( k \) of a 2-tier Apache Web server as a function of the response time at \( k-1 \) and product of inverse of relative frequency and the change in the CPU allocation. Though their performance model considers both the CPU and frequency of the server as parameters, their experimentation were neither performed on non-CPU-intensive applications, which have lesser performance loss with the change in frequency, nor heterogeneous applications with varied SLAs.

Non-CPU-intensive applications were again neglected by Petrucci et al. [23] when they proposed a performance model that depends on the CPU utilization and frequency of the server. They assumed CPU as the bottleneck for httperf tool and predicted the performance \( r_{ij} \) for any given utilization \( u_{ij} \) and frequency \( f_{ij} \) using the following equation.

\[ r_{ij}(u_{ij}) = R_{iF} \cdot u_{ij} \cdot \frac{f_{ij}}{f_{F_i}} \]

where \( R_{iF} \) is the performance at maximum frequency \( f_{F_i} \) and CPU utilization. Our completion time model overcomes this drawback and predicts the performance accurately with upto two orders of magnitude for non-CPU-intensive benchmarks.

### 4. MODELING POWER CONSUMPTION OF VIRTUALIZED SERVERS

Modern processors offer a minimum of 2 (AMD x4 9550) and up to 10 (Intel i7 2600) frequency steps. As discussed earlier, there is a need to consider the effect of operating frequency and the CPU utilization on the dynamic power of the system. Our methodology for empirically modeling power drawn by virtualized servers is described below.

- Observe the power consumption trend of a system and the effect of various parameters such as number of VMs, CPU utilization, frequency of the cores, etc.
- Identify the key parameters that contribute to the power consumption.
- Empirically derive the power model through regression of the input parameters.
- Validate the model on other systems which are heterogeneous in terms of processor architecture, processor manufacturers and chassis.

This section details the experimental setup to collect data to derive the power model. The input parameters of the model are determined through empirical evidence and regression is used to derive the model for Intel i7 2600.

#### 4.1 Experimental Setup

The experiments in this paper are performed on 5 virtualized systems listed in Table 1. All the systems operate on Linux kernel 3.2.0-23-generic-pae and QEMU Kernel-based Virtual Machine 1.0 hypervisor. The VMs also operate on Linux kernel 3.2.0-23-generic-pae. The power is measured using KryKard ALM 10 [1] connected to the input supply of the systems. We experiment with a pseudo microbenchmark performing long double multiplication operations. The four VMs are pinned to four different cores and each VM simultaneously executes the benchmark. The frequency of the cores are set using cpufreq 2.4.2 and the CPU allocation is capped using cpulimit 1.1. All experiments are carried out with the processor frequency and CPU allocation of VMs as independent variables.

<table>
<thead>
<tr>
<th>System</th>
<th>( f_{min} ) (GHz)</th>
<th>( f_{max} ) (GHz)</th>
<th>( f_{step} ) (GHz)</th>
<th>Other fs</th>
</tr>
</thead>
<tbody>
<tr>
<td>i7 2600</td>
<td>1.6</td>
<td>3.4</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>i5 760</td>
<td>1.86</td>
<td>2.79</td>
<td>133</td>
<td>1.2, 2.79+GHz</td>
</tr>
<tr>
<td>Xeon E5507</td>
<td>1.6</td>
<td>2.26</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td>Xeon E5520</td>
<td>1.6</td>
<td>2.26</td>
<td>133</td>
<td>2.26+GHz</td>
</tr>
<tr>
<td>AMD 9550</td>
<td>1.1</td>
<td>2.2</td>
<td>1100</td>
<td></td>
</tr>
</tbody>
</table>

#### 4.2 Number of VMs as a Parameter

In this subsecction, we look at whether the number of active VMs is a parameter that affects the power. The correlation between the number of active VMs and the power is determined by increasing the VMs running long double multiplication operations and the power consumption is observed for Intel i7 2600. In the first scenario, the VMs as allocated a cap of 50% of the total CPU and in the second scenario, no such cap is imposed. The power remains a constant after 2 VMs for 50% overall CPU cap of the system.
and 4 VMs for 100% i.e., no CPU cap on the quad core processor, as shown in Figure 3. This suggests that rather than number of active VMs, the CPU utilization of the system is a more accurate parameter to predict the power consumed by the system.

![Figure 3: Power, CPU % vs Number of VMs](image)

### 4.3 Individual Core Frequencies as a Parameter

We measure the power consumption of a system on which 4 VMs running long double multiplication operations are executing. Each of the 4 VMs are pinned to one of the quad cores of Intel i7 and are allocated 100% of the core. Let $f_i$ be the frequency of core $c_i$. We operate each core on a combination of the following 4 frequencies - 3.4GHz, 2.6GHz, 2GHz and 1.6GHz. For example, configuration C2 has the first core $c_1$ operating at 3.4GHz, $c_2$ at 2GHz, $c_3$ at 1.6GHz and $c_4$ at 2.6GHz. Table 2 lists a few frequency configurations across the 4 cores and shows that even if one of the cores is operating at the highest frequency, the power consumption is the same (99 W). The lowest power of 51 W is reached only when all the cores are operating at 1.6GHz. Moreover, all the cores are homogeneous and the order of the cores does not affect the power i.e., power consumed when operating at C2 is the same as operating at C3 (99 W). Thus, it is sufficient to use the highest operating frequency as the input to model the power consumption of the system rather than the individual operating frequencies of each core.

![Table 2: Power consumption for different frequency settings](table)

### 4.4 Type of Placement - Packing Cores vs Balancing Loads across VMs

The VMs can be placed in many ways across the cores of the processor. Some of the combinations (core-by-core basis (consolidation) or load balancing across the cores) are given in Table 3 along with the corresponding power consumption. C1 represents 4 VMs pinned on 4 cores, each operating at 25% of their capacity. C3 configuration has all 4 VMs pinned to one core. Load balancing the VMs across all the cores at $f_{max}$ can save 1.5% of power (Configuration C4 vs C5) to 8% (Configuration C1 vs C3). It is evident that balancing VMs across all the cores is more power efficient than packing VMs on minimal cores and hence for our power model, we assume that the VMs are placed homogeneously on all cores.

![Table 3: CPU and Power drawn for various placement schemes](table)

### 4.5 Selection of Input Parameters

It is evident from the above experiments that in order to predict the power consumption of a virtualized server, the CPU utilization and the highest currently operating frequency are required. Moreover, voltage of the processor is not considered as modifying the manufacturer-set voltage settings would reduce the life time of the CPU and void the processor warranty. We also assume that the VMs are pinned evenly across all the cores. Therefore, the CPU% and highest currently operating frequency are the two input parameters of our power model. We now proceed to derive our power model in the next section.

### 5. Derivation of Power Model for Virtualized Server

Our aim is to build a model to predict the power consumption of a server for a given CPU utilization and frequency settings with minimal inputs required. Let $P_{cpu}^f$ be the power drawn at CPU allocation and $f$ operating frequency. The total power (TPower) of the system depends both on CPU and $f$, the basic power model is given as

$$TPower_{cpu} = P_{idle} + P_{dynamic}$$

where $P_{idle}$ is the idle power and $P_{dynamic}$ is the dynamic power consumed. We now aim to model the idle power if Intel i7. The $P_{idle}$ for all the frequencies were measured and plotted in Figure 4. It is evident from Figure 4 that $P_{idle}$ is dependent on the frequency of the processor. Moreover, it is trivial that CPU utilization is 0% and hence, does not contribute to $P_{idle}$. In order to make the model system-independent, we use relative change in the frequency as the input parameter instead of the absolute frequency, and apply linear regression on the $P_{idle}$ values. With a linear fitting of 0.91 for the coefficient of determination, $R^2$, it is established
that $P_{idle}$ is linearly dependent on the relative frequency and is modeled as

$$P_{idle} = P_0^{idle} - \alpha \cdot \left(\frac{f_{max} - f}{f_{max}}\right)$$

(2)

where $P_0^{idle}$ is the idle power at $f_{max}$ and $\alpha = 1.237$ (watts) is the idle power slope of Intel i7. Therefore, substituting Equation 2 in Equation 1, we get

$$TPower_{cpu} = P_0^{idle} - \alpha \cdot \left(\frac{f_{max} - f}{f_{max}}\right) + P_{dynamic}$$

(3)

We now empirically analyze and explain our intuition behind deriving a model for $P_{dynamic}$. Figure 1 shows the power consumed for different CPU% and frequency of the system. Intel i7 2600 offers 61.6% of $P_{dynamic}$ and power varies by as much as 44.5% across $f_{min}$ and $f_{max}$ at 100% CPU utilization. It is clear that the power trend for each frequency is linearly proportional to the CPU% i.e., $P_{dynamic} \propto CPU$.

Table 4 shows the fitness of the above equation along with $\beta_f$ values for the ten frequency steps. The absolute difference between $P_{dynamic}$ and $\beta_f$ is very low for most frequencies. Therefore, Equation 3 can be rewritten as

$$TPower_{cpu} = P_0^{idle} - \alpha \cdot \left(\frac{f_{max} - f}{f_{max}}\right) + \beta_f \cdot cpu$$

(4)

To use Equation 4 as such for predicting power, ten $\beta_f$ values are required. Therefore, we further simplify our model by plotting $\beta_f$ values across different frequencies as shown in Figure 5. $\beta_f$ for any frequency is modeled as linearly dependent on the relative frequency with $R^2 = 0.986$ as

| $f$ (GHz) | $\beta_f$ | $P_0^{dyn}$ | $| \beta_f - P_0^{dyn} |$ | $R^2$ |
|----------|----------|-----------|----------------|--------|
| 3.4      | 57.77    | 56.44     | 1.33          | 0.999  |
| 3.2      | 52.49    | 52.28     | 0.21          | 0.999  |
| 3.0      | 45.84    | 45.91     | 0.07          | 0.999  |
| 2.8      | 40.04    | 39.89     | 0.15          | 0.999  |
| 2.6      | 34.93    | 34.77     | 0.16          | 0.999  |
| 2.4      | 30.30    | 30.17     | 0.13          | 0.999  |
| 2.2      | 26.14    | 25.81     | 0.33          | 0.999  |
| 2.0      | 22.47    | 22.24     | 0.23          | 0.999  |
| 1.8      | 18.95    | 18.82     | 0.13          | 0.999  |
| 1.6      | 16.00    | 15.82     | 0.18          | 0.999  |

The above equation requires only three calculated inputs - $A$, $B$ and $\alpha$ and the next subsection explains how they are obtained.

5.1 Obtaining $A$, $B$ and $\alpha$ values

The $A$, $B$ and $\alpha$ values are calculated using Equations 7, 8 and 9.

$$A = ((P_{f_{max}}^{1.0} - P_{f_{min}}^{1.0}) - (P_{f_{max}}^{0.0} - P_{f_{min}}^{0.0})) \cdot \frac{f_{max}}{f_{max} - f_{min}}$$

(7)

$$B = (P_{f_{max}}^{1.0} - P_{f_{min}}^{1.0}) - A$$

(8)

$$\alpha = (P_{f_{max}}^{0.0} - P_{f_{min}}^{0.0}) \cdot \frac{f_{max}}{f_{max} - f_{min}}$$

(9)

We require only 6 inputs - $f_{min}$, $f_{max}$, $P_{f_{min}}$, $P_{f_{max}}$, $P_{f_{max}}^{1.0}$ and $P_{f_{max}}^{0.0}$ to calculate $A$, $B$ and $\alpha$ and to predict the power at a given CPU% and $f$, as shown in Figure 6. Section 5.2 describes the experimental setup for validating our power model and enlists the accuracy of prediction of our power model across heterogeneous systems.

5.2 Validation of the Power Model

The power model that we derived in Equation 6 is validated by predicting power of 4 heterogeneous machines - quad core i5 760 Desktop, dual Xeon E5507 rack mount server, dual Xeon E5520 blade server and AMD x4 9550 quad core Desktop. The systems are heterogeneous in terms of processor architecture, number of processors, processor
manufacturers and chassis and aids in establishing the validity of our power model across a variety of systems.

We observe the idle and total power at $f_{\text{min}}$ and $f_{\text{max}}$. $A$ and $B$ values are calculated using Equations 7 and 8 respectively, and $\alpha$ is obtained using Equation 9. Figures 7(a)-(e) show the power consumption trend for the 5 systems, including Intel i7. Table 5 provides the $A$, $B$, and $\alpha$ values. The deviation of predicted power from the measured power is calculated using Equation 10. Figure 7(f) shows the accuracy of our power model and Table 6 compares prediction accuracy of Petrucci’s model [23] with our proposed model across 5 systems.

$$\text{Error}\% = \frac{|\text{Measured} - \text{Predicted}|}{\text{Measured}} \times 100\%$$ (10)

The maximum error in prediction ranged from 1.89% for AMD x4 9550 to 7.39% for Intel i5. The ‘Turbo’ mode of Intel i7 and Xeon 5520 were assumed to be three frequency steps (one step is 133MHz) higher than the highest frequency 2.793GHz and 2.261GHz respectively i.e, the turbo mode of i5 and E5520 were assumed to be 3.192GHz and 2.66GHz respectively. And the ‘low’ mode of Intel i5, 1.2GHz is assumed to be 5/3 frequency steps lesser than the lowest frequency of 1.862GHz i.e., the ‘low’ mode’s frequency is assumed to be 1.64GHz. This is due to the change in the power is thrice as much as performance while using DVFS [19]. The following were made for the measured power across 5 systems.

- The idle power of the blade and rack-mount servers are significantly higher than desktop machines due to the chassis housing more components, including dual processors, dual fans, more RAM and in the case of Xeon 5520 blade server, integrated management module and ethernet switch.

- Idle power of the Intel processors varies by only about 1W but AMD x4 9550 supports a reduction of 6W between its $f_{\text{min}}$ and $f_{\text{max}}$.

- Even though Intel i7 and AMD 9550 have similar $P^{\text{dyn}}$ at $f_{\text{max}}$ - 56W and 54W respectively, the dynamic range varies by about 11% due to the very low $P_{\text{idle}}$ of Intel i7 of 35W.

- The low error of Xeon 5520 is due to the high idle power (191 W) and a narrow dynamic range of 12.95% while Xeon 5507’s idle power (77 W) is only 40% of Xeon 5520’s idle power and has a wider dynamic range of 48.07%.

The following are the observations of predicting power of the 5 systems using our model and Petrucci’s model [23], as shown in Table 6.

- 4 out of 5 systems have lesser or the same average prediction error % for our model than Petrucci’s model. The 5th system has a higher average error of 0.01%.

- Petrucci’s model does not predict as accurately as our model, given the same input for nearly every system with the highest error being 9.68%. This is because they had modeled power based on the square of the relative change in the frequency while we model it linearly.

- With the maximum error of 7.39%, our power model is able to predict the power for various CPU utilizations and frequency configurations across heterogeneous processors, processor manufacturers and chassis.

### Table 5: Input values of Power Model for Intel and AMD systems

<table>
<thead>
<tr>
<th>Processor</th>
<th>$P_{\text{fmin}}^{\text{th}}$ (Watts)</th>
<th>$P_{\text{fmax}}^{\text{th}}$ (Watts)</th>
<th>$A$</th>
<th>$B$</th>
<th>$P^{\text{dyn}}_{\text{fmax}}$ (Watts)</th>
<th>$P^{\text{th}}_{\text{fmax}}$ (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i7 2600</td>
<td>35.54</td>
<td>36.14</td>
<td>1.09</td>
<td>51.36</td>
<td>92.56</td>
<td>76.72</td>
</tr>
<tr>
<td>i5 760</td>
<td>55.75</td>
<td>56.28</td>
<td>1.59</td>
<td>107.25</td>
<td>149.72</td>
<td>125.82</td>
</tr>
<tr>
<td>Xeon E5507</td>
<td>77.28</td>
<td>77.46</td>
<td>0.61</td>
<td>127.77</td>
<td>148.82</td>
<td>70.95</td>
</tr>
<tr>
<td>Xeon E5520</td>
<td>191.01</td>
<td>191.01</td>
<td>0</td>
<td>208.38</td>
<td>219.45</td>
<td>37.638</td>
</tr>
<tr>
<td>AMD x4 9550</td>
<td>65.63</td>
<td>71.62</td>
<td>11.98</td>
<td>83.88</td>
<td>125.44</td>
<td>71.14</td>
</tr>
</tbody>
</table>

### Table 6: Validation of power model on 5 systems

<table>
<thead>
<tr>
<th>System</th>
<th>Petrucci [23]</th>
<th>Our Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg Error %</td>
<td>Max Error %</td>
</tr>
<tr>
<td>i7 2600</td>
<td>3.83</td>
<td>9.68</td>
</tr>
<tr>
<td></td>
<td>1.89</td>
<td>5.83</td>
</tr>
<tr>
<td>i5 760</td>
<td>2.86</td>
<td>8.45</td>
</tr>
<tr>
<td></td>
<td>2.36</td>
<td>7.39</td>
</tr>
<tr>
<td>Xeon E5507</td>
<td>3.01</td>
<td>6.54</td>
</tr>
<tr>
<td></td>
<td>3.02</td>
<td>6.03</td>
</tr>
<tr>
<td>Xeon E5520</td>
<td>2.41</td>
<td>5.89</td>
</tr>
<tr>
<td></td>
<td>0.97</td>
<td>2.00</td>
</tr>
<tr>
<td>AMD x4 9550</td>
<td>0.83</td>
<td>1.89</td>
</tr>
<tr>
<td></td>
<td>0.83</td>
<td>1.89</td>
</tr>
</tbody>
</table>

Thus, we derived a model to predict the power consumption of a system using only 2 frequencies and 4 power values as inputs. We, then, validated with 5 systems which are heterogeneous in terms of processors, processor manufacturers and chassis. Executing VMs inherently includes the utilization of memory and cache. The power model built in a virtualized environment ensured the inclusion of power consumption of memory and the reusability of the model for power optimal VM placements. However, the derived model will hold for non-virtualized environments such as bare metal, HPC and Hadoop clusters. Their power values can be predicted by supplying the new power values of the HPC and Hadoop systems. Our power model can be used for predicting power as well as setting a power budget for VM placement strategies.

### 6. A COMPLETION TIME MODEL FOR VIRTUAL MACHINES

Service Level Agreements (SLAs) are the operative words of any data center service provider. Consider a scenario where the SLAs are based on the completion time of applications, which is a typical performance metric for HPC tasks and workloads that require batch processing such as Hadoop’s Map-Reduce. The resources are allocated based on the required time for completion. In this section, we aim to understand the effect of compute-resource and frequency modifications on the execution time of the VMs and
experimentally derive a completion time model that can be used for power-optimal placement of VMs without violating execution time SLA constraints. Our methodology for empirically modeling completion time of VMs is described below.

- Observe the completion time of VMs by varying the parameters that affect the completion time of benchmarks executing inside VMs
- Empirically derive the completion time model through regression of the input parameters.
- Validate the model with multiple benchmarks on other systems which are heterogeneous in terms of processor architecture, number of processors, processor manufacturers and chassis.

In this section we identify the parameters that affect the completion time of a task executing inside a virtual machine and empirically derive a completion time i.e., the elapsed wall clock time, prediction model for Intel i5 760 Desktop, as shown in Figure 9. The frequency change does not influence the relative completion time (CT) of randmem32 as much shown in Figure 9. The frequency change does not influence the relative completion time (CT) of randmem32 as much

6.2 Frequency of Processor as Parameter

The effect of frequency modification is observed for cpuTest and randmem32 at 100% CPU and the highest frequency, the completion time of cpuTest and randmem32 are almost the same - 57 and 58 seconds, respectively. At 20% CPU and lowest frequency, the completion time of cpuTest and randmem32 vary drastically at 680 and 413 seconds. This substantial change in the completion time with respect to the frequency and CPU% alterations are analyzed in Sections 6.2 and 6.3. A completion time model with CPU% and frequency as parameters is derived and validated in Section 7.

6.1 Experimental Setup for Completion Time Model Derivation

The experiments below were performed on virtualized Intel i5 760. For derivation of the completion time model, 4 VMs execute cpuTest for 11 billion iterative long double addition and multiplication and randmem32 [2] benchmark that transfers data at increasing data sizes from and to caches and memory. The benchmarks stress test the CPU and the memory subsystems respectively. The benchmarks are executed inside the VMs simultaneously by pinning them to four different cores. The average of the real value of time command of the 4 VMs is noted as the completion time for a specific CPU-frequency combination. The number of VMs is assumed to be 4 as it was established that with 4 VMs, peak CPU utilization and thus peak power is attained.

Figures 8 (a) and (b) graphically represent the completion time values of the 2 benchmarks. At 100% CPU and the highest frequency, the completion time of cpuTest and randmem32 are almost the same - 57 and 58 seconds, respectively. At 20% CPU and lowest frequency, the completion time of cpuTest and randmem32 vary drastically at 680 and 413 seconds. This substantial change in the completion time with respect to the frequency and CPU% alterations are analyzed in Sections 6.2 and 6.3. A completion time model with CPU% and frequency as parameters is derived and validated in Section 7.

\[
U = \left[ CT_{f_{\text{min}}} - CT_{f_{\text{max}}} \right] \cdot \frac{f_{\text{max}} - f_{\text{min}}}{f_{\text{min}}}
\]

and the completion time for any frequency can be predicted using

\[
CT_f = \left[ U \cdot \frac{f_{\text{max}}}{f} + V \right] \cdot CT_{f_{\text{max}}}
\]  

where \( U \) is the fraction of the application that is dependent on the frequency, \( V \) is the fraction that is independent of the frequency changes and \( U + V = 1 \). The \( U \) value was empirically found to be 1.04 and 0.49 for cpuTest and randmem32 respectively. \( U \) is architecture-dependent [20] and needs to
Completion Time Trends for Intel i5 760
(a) Completion Time for cpuTest vs CPU Allocation w.r.t Frequencies
(b) Completion Time for randmem32 vs CPU Allocation w.r.t Frequencies

Figure 8: Completion Time trend of (a) cpuTest (b) randmem32 (c) Relative completion time vs frequency (d) Relative completion time vs CPU% for Intel i5 760

Figure 9: Relative completion time vs frequency for cpuTest and randmem32 on Intel i5 760 be recalculated for different processors.

6.3 CPU Allocation as Parameter

Figure 10: Relative completion time vs frequency for cpuTest and randmem32 on Intel i5 760

Amdahl’s law [7], one of the fundamental laws of Computer Architecture, is used to find the maximum expected improvement to an overall system when only part of the system is improved. i.e.,

$$CT_{new}^{\text{cpu}} = \left(\frac{F_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} + (1 - F_{\text{enhanced}})\right) \cdot CT_{old}^{\text{cpu}}$$

where $F_{\text{enhanced}}$ is the fraction of the application that is enhanced. Rewriting the above equation by expressing speedup $\text{Speedup}_{\text{enhanced}}$ as the CPU% allocated,

$$CT_{cpu}^{\text{CPU}} = \left[\theta \cdot \frac{1}{\text{CPU}} + \mu \right] \cdot CT_{1.0}$$

where $\theta$ quantifies the compute-boundedness of the application that is affected by the change in CPU allocation, $\mu$ is the CPU-independent part of the application and $\theta + \mu = 1$. For a CPU-intensive task, $\theta \approx 1$. This is used by [21], [29] and a host of other authors to calculate the CPU% needed by VMs to execute a task without violating SLA. However, the above equation does not capture the combined effect of frequency and CPU alteration on the completion time. While $\theta$ remains a constant for CPU-intensive tasks, it varies for non-CPU-intensive tasks with change in the frequency as shown in Figure 10. Therefore, $\theta$ has to be recalculated for every $f$.

$$CT_{f}^{\text{CPU}} = \left[\theta f \cdot \frac{1}{\text{CPU}} + \mu f \right] \cdot CT_{f}^{1.0}$$

In order to predict the completion time of tasks executing inside of VMs, the total CPU allocated and the frequency of the system are required and are thus, used as the input parameters of our completion time model. We now proceed to model our completion time prediction and validate the same in the next section.

7. DERIVATION OF COMPLETION TIME MODEL FOR VIRTUALIZED ENVIRONMENTS

Let $CT_{f}^{\text{CPU}}$ be the completion time for a given CPU utilization ratio $\text{cpu}$ and frequency $f$. Equation 11 expresses the frequency dependency and Equation 12 gives the compute-boundedness of an application. Rewriting Equation 11 at 100% utilization as

$$CT_{f}^{1.0} = \left[U \cdot \frac{f_{\text{max}}}{f} + V\right] \cdot CT_{f}^{1.0}_{\text{max}}$$

Equation 13 is plugged into Equation 12 to predict $CT_{f}^{\text{CPU}}$ as

$$CT_{f}^{\text{CPU}} = \left[\theta f \cdot \frac{1}{\text{CPU}} + \mu f \right] \cdot \left[U \cdot \frac{f_{\text{max}}}{f} + V\right] \cdot CT_{f}^{1.0}_{\text{max}}$$

$U$ is calculated using $CT_{f}^{1.0}_{\text{max}}$, $CT_{f}^{1.0}_{2.79\text{GHz}}$, and Equation 13 and is found to be 1.0 and 0.43 for cpuTest and rand-
Table 7: Verification for cpuTest and randmem32

<table>
<thead>
<tr>
<th>Frequency (GHz)</th>
<th>cpuTest U=1.0</th>
<th>randmem32 U=0.43</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\theta_f)</td>
<td>(R^2)</td>
</tr>
<tr>
<td>2.79</td>
<td>1.0</td>
<td>0.999</td>
</tr>
<tr>
<td>2.79</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>2.66</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>2.66</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>2.39</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>2.26</td>
<td>1.02</td>
<td>0.999</td>
</tr>
<tr>
<td>2.13</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>2.0</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>1.86</td>
<td>1.01</td>
<td>0.999</td>
</tr>
<tr>
<td>1.2</td>
<td>1.02</td>
<td>0.999</td>
</tr>
</tbody>
</table>

\(R^2\) as 0.978 suggests that \(\theta_f = K \cdot (\frac{f_{max}}{f_{min}} - 1) + \theta_{fmax}\) and slope \(K = 0.132\) for randmem32. Since \(\theta_f \approx \theta_{fmax}\) for all frequencies of cpuTest, \(K\) is assumed to be 0. Moreover, \(K\) can be approximated as

\[K = (\theta_{fmin} - \theta_{fmax}) \cdot (\frac{f_{min}}{f_{max}} - 1)\]

and the calculated value was found to be \(K = 0.12\). Therefore, our model requires only \(\theta_{fmin}\) and \(\theta_{fmax}\) to calculate all other \(\theta_f\) values. And how we calculate these values is explained in the next subsection.

Figure 11: Proposed Completion Time Model Framework

7.1 Obtaining U and \(\theta_f\) values

\(U\) and \(\theta_f\) values are calculated using Equations 15, 16, 17 and 18.

\[U = \left(\frac{CT_{0.0}^{f_{0.0}} - CT_{1.0}^{f_{max}}}{CT_{1.0}^{f_{max}}}ight) \cdot \left(\frac{f_{max} - f_{min}}{f_{min}}\right)\] (15)

\[\theta_{fmax} = \left(\frac{0.9}{1.0 - 0.9}\right) \cdot \left(\frac{CT_{0.0}^{f_{0.0}} - CT_{1.0}^{f_{max}}}{CT_{1.0}^{f_{max}}}\right)\] (16)

\[\theta_{fmin} = \left(\frac{0.9}{1.0 - 0.9}\right) \cdot \left(\frac{CT_{0.0}^{f_{0.0}} - CT_{0.0}^{f_{min}}}{CT_{0.0}^{f_{min}}}\right)\] (17)

\[\theta_f = (\theta_{fmin} - \theta_{fmax}) \cdot \left(\frac{f_{min}}{f_{max}} - \frac{f_{min}}{f_{max}}\right) - \left(\frac{f_{max}}{f_{max}} - f_{max}\right) + \theta_{fmax}\] (18)

We require only 7 inputs - \(f_{min}, f_{max}, 2\%\) CPU utilization, \(CT_{0.0}^{f_{0.0}}, CT_{0.0}^{f_{max}}\) and \(CT_{1.0}^{f_{max}}\) to predict the power at a given CPU% and \(f\), as shown in Figure 11. Section 7.2 describes the experimental setup for validating our completion time model and Section 7.3 enlists the accuracy of prediction of our completion time model and existing model used by Petrucci et al. [23], across heterogeneous applications and machines.

7.2 Experimental Setup for Completion Time Model Validation

The experiments below were performed on the following virtualized systems - Intel i5 760, Intel i7 2600, dual Intel Xeon E5507 and AMD x4 9550. We have experimented with 6 benchmarks - SysBench CPU test [3] for finding the first 50,000 prime numbers, cpuTest [4], randmem32 [2], kernel compile 3.9.4 [5], IOZone [6] for creating and deleting 1000 files and SysBench File [3] for random read and write of 10 files totaling 1GB. These 6 benchmarks stress test the CPU, memory, I/O and a combination of these resources. The benchmarks are executed inside the VMs simultaneously by pinning them to four different cores. The frequencies of the ‘low’ and ‘turbo’ modes of Intel i5 are assumed to be 1.197GHz and 2.926GHz respectively.

7.3 Validation of Completion Time Model

Table 8: Validation of completion time model using 6 benchmarks on AMD 9550, Intel i5, i7 and E5507

<table>
<thead>
<tr>
<th>Task</th>
<th>System</th>
<th>Error % using</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Petrucci [23]</td>
<td>Our Model</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>cpuTest</td>
<td>AMD</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td>i5</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>i7</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>E5507</td>
<td>2.47</td>
</tr>
<tr>
<td>sys bench</td>
<td>AMD</td>
<td>1.26</td>
</tr>
<tr>
<td></td>
<td>i5</td>
<td>0.52</td>
</tr>
<tr>
<td></td>
<td>i7</td>
<td>1.27</td>
</tr>
<tr>
<td></td>
<td>E5507</td>
<td>1.26</td>
</tr>
<tr>
<td>mem32</td>
<td>AMD</td>
<td>19.32</td>
</tr>
<tr>
<td></td>
<td>i5</td>
<td>29.29</td>
</tr>
<tr>
<td></td>
<td>i7</td>
<td>44.86</td>
</tr>
<tr>
<td></td>
<td>E5507</td>
<td>17.04</td>
</tr>
<tr>
<td>Kernel Compile</td>
<td>AMD</td>
<td>4.76</td>
</tr>
<tr>
<td></td>
<td>i5</td>
<td>4.89</td>
</tr>
<tr>
<td></td>
<td>i7</td>
<td>11.04</td>
</tr>
<tr>
<td></td>
<td>E5507</td>
<td>17.55</td>
</tr>
<tr>
<td>IOzone</td>
<td>AMD</td>
<td>231.46</td>
</tr>
<tr>
<td></td>
<td>i5</td>
<td>204.84</td>
</tr>
<tr>
<td></td>
<td>i7</td>
<td>227.2</td>
</tr>
<tr>
<td></td>
<td>E5507</td>
<td>168.5</td>
</tr>
<tr>
<td>sys bench</td>
<td>AMD</td>
<td>199.17</td>
</tr>
<tr>
<td></td>
<td>i5</td>
<td>229.09</td>
</tr>
<tr>
<td></td>
<td>i7</td>
<td>292.98</td>
</tr>
</tbody>
</table>

The model is validated by predicting completion of CPU- and non-CPU-intensive tasks on AMD x4 9550 and Dual Intel E5507 rack mount server. Figures 12 (a)-(l) show the CDF of the error in predicting the completion time for cpuTest, SysBench CPU, randmem32, kernel compile, IOZone and SysBench File benchmarks using our completion.
time model. Table 8 compares the prediction of our model with the exiting work. We observed the following while predicting the execution time for two CPU-, memory and I/O-intensive benchmarks each.

- The dependency on the processor frequency i.e., the value of $U$ was $\approx 1$ for both the CPU-intensive benchmarks and $\approx 0$ for both the I/O-intensive benchmarks across the 4 heterogeneous servers.

- The $U$ value vary from 0.25 to 0.52 for randmem32 and 0.45 to 0.84 for kernel compile. This indicates that $U$ has to be calculated for individual applications on each system.

- The completion time monotonically increased from 100% to 20% for all frequencies, benchmarks and systems, except for kernel compile. While it still monotonically increased from 80% to 20%, compilation took longer at 100% than 80%. This led to a maximum error in prediction about 16% on Intel i5, i7 and AMD 9550. This outlier could be attributed to the bottleneck created by CPU on the memory and I/O subsystems.

- To overcome the above exception, the base of the prediction to 80%, i.e., the following equation was used to predict the time for kernel compilation where $U$ is obtained from $CT_{cpu}^{0.8}$ and $CT_{cpu}^{f_{min}}$.

  $$CT_{cpu}^{pred} = \lceil f_{cpu} \cdot 0.8 + \mu_f \cdot \left[ U_{f_{max}} + V \right] \cdot CT_{cpu}^{f_{max}}$$

- The above method for prediction led to a maximum error of 5.2% across of 20% to 80% CPU but as much as 30% for 100% CPU utilization. Again, this outlier could be attributed to the bottleneck created on the memory and I/O subsystems.

- The completion time predicted using the inverse of the performance formula described by Petrucci et al. [23] yielded a maximum error of 8.19% for CPU-intensive benchmarks. This is because they used only a single input and failed to consider the compute- and frequency-dependence.

- Our average error is consistently lesser than [23] for CPU-intensive as well as other-resource-intensive benchmarks.

- Compared to our model, existing work predicted almost one magnitude worse for memory-intensive and more than two magnitudes worse for file-intensive benchmarks.

- With the maximum error of 6.81%, our unified completion time model was able to predict the time for various CPU utilizations and frequency configurations across heterogeneous benchmarks and systems.

Thus, we proposed a completion time model to predict the performance of tasks operating inside the VMs using only 2 frequencies, a CPU fraction and 4 execution time values. Unlike learning techniques that require hundreds of data points, our model requires only 4 completion time values per application on each system. We addressed the issue of predicting the running time for memory- as well as file-intensive benchmarks and our model has a high prediction accuracy across heterogeneous virtualized systems. Our completion time model can also be used for Hadoop and HPC environments.

Figure 12: CDF of error in predicting the completion time of (a) cpuTest (b) SysBench CPU (c) randmem32 (d) kernel compile (e) IOzone (f) SysBench File for AMD 9550, Intel i5, Intel i7 and Dual Xeon E5507
8. CONCLUSIONS

In this paper, we proposed a power and a performance model with CPU utilization and frequency as parameters. To achieve that, we studied the power consumption trend of Intel i7, identified the parameters and empirically derived a power model with CPU allocation and frequency as input. We validated the power model by predicting power of 5 systems heterogeneous in terms of the processors, process manufacturers and chassis, with a maximum error of 7.4%. We also derived a completion time model by combining existing models that predict using CPU allocation and frequency of the system individually. We validated the completion time model by predicting the execution time of 6 CPU- and non-CPU-intensive tasks on 4 machines with a maximum error of 6.8%. The two derived models can be combined to power optimally place VMs on servers, without violating their completion time SLAs. They can also be applied to HPC and Hadoop environments with minimal modifications.

For future work, we intend to extend the power model to dual or n-processor systems, encompass memory, file and network resources and propose a VM placement algorithm that is completion-time-aware and power optimal across a cloud setup. The algorithm can also be extended to include dynamic SLA requirement and reallocation of server resources power optimally.

9. REFERENCES