Optimal Power Allocation in Server Farms

Anshul Gandhi
Carnegie Mellon University
Pittsburgh, PA, USA
anshulg@cs.cmu.edu

Mor Harchol-Balter*
Carnegie Mellon University
Pittsburgh, PA, USA
harchol@cs.cmu.edu

Rajarshi Das
IBM Research
Hawthorne, NY, USA
rajarshi@us.ibm.com

Charles Lefurgy
IBM Research
Austin, TX, USA
lefurgy@us.ibm.com

ABSTRACT

Server farms today consume more than 1.5% of the total electricity in the U.S. at a cost of nearly $4.5 billion. Given the rising cost of energy, many industries are now seeking solutions for how to best make use of their available power. An important question which arises in this context is *how to distribute available power among servers in a server farm so as to get maximum performance.*

By giving more power to a server, one can get higher server frequency (speed). Hence it is commonly believed that, for a given power budget, performance can be maximized by operating servers at their highest power levels. However, it is also conceivable that one might prefer to run servers at their lowest power levels, which allows more servers to be turned on for a given power budget. To fully understand the effect of power allocation on performance in a server farm with a fixed power budget, we introduce a queueing theoretic model, which allows us to predict the optimal power allocation in a variety of scenarios. Results are verified via extensive experiments on an IBM BladeCenter.

We find that the optimal power allocation varies for different scenarios. In particular, it is not always optimal to run servers at their maximum power levels. There are scenarios where it might be optimal to run servers at their lowest power levels or at some intermediate power levels. Our analysis shows that the optimal power allocation is non-obvious and depends on many factors such as the power-to-frequency relationship in the processors, the arrival rate of jobs, the maximum server frequency, the lowest attainable server frequency and the server farm configuration. Furthermore, our theoretical model allows us to explore more general settings than we can implement, including arbitrarily large server farms and different power-to-frequency curves. Importantly, we show that the optimal power allocation can significantly improve server farm performance, by a factor of typically 1.4 and as much as a factor of 5 in some cases.

*Research supported by NSF SMA/PDOS Grant CCR-0615262 and a 2009 IBM Faculty Award.

SIGMETRICS/Performance'09, June 15–19, 2009, Seattle, WA, USA.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques

General Terms

Theory, Experimentation, Measurement, Performance

1. INTRODUCTION

Servers today consume ten times more power than they did ten years ago [3, 21]. Recent articles estimate that a 300W high performance server requires more than $330 of energy cost per year [24]. Given the large number of servers in use today, the worldwide expenditure on enterprise power and cooling of these servers is estimated to be in excess of $30 billion [21].

Power consumption is particularly pronounced in CPU intensive *server farms*, composed of tens to thousands of servers, all *sharing workload and power supply.* We consider server farms where each incoming job can be routed to any server, i.e., it has no affinity for a particular server.

Server farms usually have a fixed peak power budget. This is because large power consumers operating server farms are often billed by power suppliers, in part, based on their peak power requirements. The peak power budget of a server farm also determines its cooling and power delivery infrastructure costs. Hence, companies are interested in maximizing the performance at a server farm given a fixed peak power budget [4, 8, 9, 21].

The power allocation problem we consider is: *how to distribute available power among servers in a server farm so as to minimize mean response time.* Every server running a given workload has a minimum level of power consumption, b, needed to operate the processor at the lowest allowable frequency and a maximum level of power consumption, c, needed to operate the processor at the highest allowable frequency. By varying the power allocated to a server within the range of b to c Watts, one can proportionately vary the server frequency (See Fig. 2). Hence, one might expect that running servers at their highest power levels of c Watts, which we refer to as *PowMax*, is the optimal power allocation scheme to minimize response time. Since we are constrained by a power budget, there are only a limited number of servers that we can operate at the highest power level. The rest of

the servers remain turned off. Thus PowMax corresponds to having *few fast* servers. In sharp contrast is *PowMin*, which we define as operating servers at their lowest power levels of b Watts. Since we spend less power on each server, PowMin corresponds to having *many slow* servers. Of course there might be scenarios where we neither operate our servers at the highest power levels nor at the lowest power levels, but we operate them at some intermediate power levels. We refer to such power allocation schemes as *PowMed*.

Understanding power allocation in a server farm is intrinsically difficult for many reasons: First, there is no single allocation scheme which is optimal in all scenarios. For example, it is commonly believed that PowMax is the optimal power allocation scheme [1, 7]. However, as we show later, PowMin and PowMed can sometimes outperform PowMax by almost a factor of 1.5. Second, it turns out that the optimal power allocation depends on a very long list of external factors, such as the outside arrival rate, whether an open or closed workload configuration is used, the power-to-frequency relationship (how power translates to server frequency) inherent in the technology, the minimum power consumption of a server (b Watts), the maximum power that a server can use (c Watts), and many other factors. It is simply impossible to examine all these factors via experiments.

To fully understand the effect of power allocation on mean response time in a server farm with a fixed power budget, we introduce a queueing theoretic model which allows us to predict the optimal power allocation in a variety of scenarios. We then verify our results via extensive experiments on an IBM BladeCenter.

Prior work in power management has been motivated by the idea of managing power at the global data center level [8, 20] rather than at the more localized single-server level. While power management in server farms often deals with various issues such as reducing cooling costs, minimizing idle power wastage and minimizing average power consumption, we are more interested in the problem of allocating peak power among servers in a server farm to maximize performance. Notable prior work dealing with peak power allocation in a server farm includes Raghavendra et al. [21], Femal et al. [10] and Chase et al. [5] among others. Raghavendra et al. [21] present a power management solution that coordinates different individual approaches to simultaneously minimize average power, peak power and idle power wastage. Femal et al. [10] allocate peak power so as to maximize throughput in a data center while simultaneously attempting to satisfy certain operating constraints such as load-balancing the available power among the servers. Chase et al. [5] present an auction-based architecture for improving the energy efficiency of data centers while achieving some quality-of-service specifications. We differ from the above work in that we specifically deal with minimizing mean response time for a given peak power budget and understanding all the factors that affect it.

Our contributions

As we have stated, the optimal power allocation scheme depends on many factors. Perhaps the most important of these is the specific relationship between the power allocated to a server and its frequency (speed), henceforth referred to as the *power-to-frequency relationship*. There are several mechanisms within processors that control the power-to-frequency relationship. These can be categorized into DFS (Dynamic Frequency Scaling), DVFS (Dynamic Voltage and Frequency Scaling) and DVFS+DFS. Section 2 discusses these mechanisms in more detail. The functional form of the power-to-frequency relationship for a server depends on many factors such as the workload used, maximum server power, maximum server frequency and the voltage and frequency scaling mechanism used (DFS, DVFS or DVFS+DFS). Unfortunately, the functional form of the server level power-to-frequency relationship is only recently beginning to be studied (See the 2008 papers [21, 25]) and is still not well understood. Our first contribution is the investigation of how power allocation affects server frequency in a single server using DFS, DVFS, and DVFS+DFS for various workloads. In particular, in Section 3 we derive a functional form for the power-to-frequency relationship based on our measured values (See Figs. 2(a) and (b)).

Our second contribution is the development of a queueing theoretic model which predicts the mean response time for a server farm as a function of many factors including the power-to-frequency relationship, arrival rate, peak power budget, etc. The queueing model also allows us to determine the optimal power allocation for every possible configuration of the above factors (See Section 4).

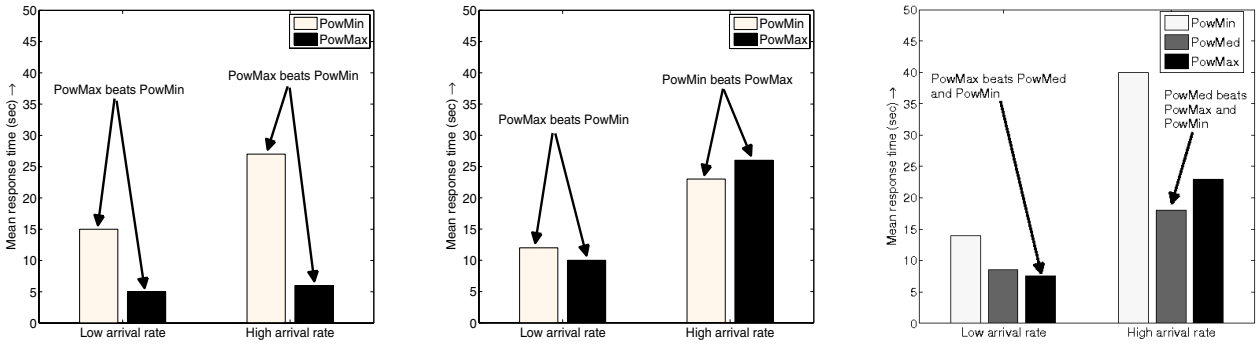
Our third contribution is the experimental implementation of our schemes, PowMax, PowMin, and PowMed, on an IBM BladeCenter, and the measurement of their response time for various workloads and voltage and frequency scaling mechanisms (See Sections 5 and 6). Importantly, our experiments show that using the optimal power allocation scheme can significantly reduce mean response time, sometimes by as much as a factor of 5. To be more concrete, we show a *subset* of our results in Fig. 1, which assumes a CPU bound workload in an open loop setting. Fig. 1(a) depicts one possible scenario using DFS where PowMax is optimal. By contrast, Fig. 1(b) depicts a scenario using DVFS where PowMin is optimal for high arrival rates. Lastly, Fig. 1(c) depicts a scenario using DVFS+DFS where PowMed is optimal for high arrival rates.

We experiment with different workloads. While Section 5 presents experimental results for a CPU bound workload, LINPACK, Section 6 repeats all the experiments under the STREAM memory bound workload, the WebBench web workload, and other workloads. In all cases, experimental results are in excellent agreement with our theoretical predictions. Section 7 summarizes our work. Finally, Section 8 discusses future applications of our model to more complex situations such as workloads with varying arrival rates, servers with idle (low power) states and power management at the subsystem level, such as the storage subsystem.

2. EXPERIMENTAL FRAMEWORK

2.1 Experimental setup

Our experimental setup consists of a server farm with up to fourteen IBM BladeCenter HS21 blade servers featuring two 3.0 GHz dual-core Intel Woodcrest Xeon processors and 1 GB memory per blade, all residing in a single chassis. We installed and configured Apache as an application server on each of the blade servers to process transactional requests. To generate HTTP requests for the Apache web servers, we employ an additional blade server on the same chassis as the workload generator to reduce the effects of network latency. The workload generator uses the web server performance benchmarking tool `httperf` [19] in the open server



(a) PowMax is best.

(b) PowMin is best at high arrival rates.

(c) PowMed is best at high arrival rates.

Figure 1: ***Subset of our results, showing that no single power allocation scheme is optimal. Fig.(a) depicts a scenario using DFS where PowMax is optimal. Fig.(b) depicts a scenario using DVFS where PowMin is optimal at high arrival rates whereas PowMax is optimal at low arrival rates. Fig.(c) depicts a scenario using DVFS+DFS where PowMed is optimal at high arrival rates whereas PowMax is optimal at low arrival rates.***

farm configuration and `wbox` [23] in the closed server farm configuration. We modified and extended `httperf` and `wbox` to allow for multiple servers and to specify the routing probability among the servers. We measure and allocate power to the servers using IBM's Amester software. Amester, along with additional scripts, collects all relevant data for our experiments.

2.2 Voltage and frequency scaling mechanisms

Processors today are commonly equipped with mechanisms to reduce power consumption at the expense of reduced server frequency. Common examples of these mechanisms are Intel's SpeedStep Technology and AMD's Cool 'n' Quiet Technology. The power-to-frequency relationship in such servers depends on the specific voltage and frequency scaling mechanism used. Most mechanisms fall under the following three categories:

Dynamic Frequency Scaling (DFS) a.k.a. Clock Throttling or T-states is a technique to manage power by running the processor at a less-than-maximum clock frequency. Under DFS, the Intel's 3.0 GHz Woodcrest Xeon processors we use allow for 8 operating points which correspond to effective frequencies of 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, and 100% of the maximum server frequency.

Dynamic Voltage and Frequency Scaling (DVFS) a.k.a. P-states is a more efficient power-savings mechanism that reduces server frequency by reducing the processor voltage and frequency. Under DVFS, our processors allow for 4 operating points which correspond to effective frequencies of 66.6%, 77.7%, 88.9%, and 100% of the maximum server frequency.

DVFS+DFS attempts to leverage both DVFS, and DFS by applying DFS on the lowest performance state available in DVFS. Under DVFS+DFS, our processors allow for 11 operating points which correspond to effective frequencies of 8.3%, 16.5%, 25%, 33.3%, 41.6%, 50.0%, 58.3%, 66.6%, 77.7%, 88.9%, and 100% of the maximum server frequency.

2.3 Power consumption within a single server

When allocating power to a server, there is a minimum level of power consumption (b) needed to operate the processor at the lowest allowable frequency and a maximum level of power consumption (c) needed to operate the processor at the highest allowable frequency (Of course the specific values of b and c depend on the application that the server is running). Formally, we define the following notation:

Baseline power: b Watts The minimum power consumed by a fully-utilized server over the allowable range of processor frequency.

Speed at baseline: s_b Hertz The speed (or frequency) of a fully utilized server running at b Watts.

Maximum power: c Watts The maximum power consumed by a fully-utilized server over the allowable range of processor frequency.

Speed at maximum power: s_c Hertz The speed (or frequency) of a fully utilized server running at c Watts.

3. POWER-TO-FREQUENCY

An integral part of determining the optimal power allocation is to understand the power-to-frequency relationship. This relationship differs for DFS, DVFS, and DVFS+DFS, and also differs based on the workload in use. Unfortunately, the functional form of the power-to-frequency relationship is not well studied in the literature. The servers we use support all three voltage and frequency scaling mechanisms and therefore can be used to study the power-to-frequency relationships. In this section, we present our measurements depicted in Figs. 2(a) and (b) showing the functional relationship between power allocated to a server and its frequency for the LINPACK [13] workload. We generalize the functional form of the power-to-frequency relationship to other workloads in Section 6 (See Figs. 8 and 10). Throughout we assume a homogeneous server farm.

We use the tools developed in [17] to limit the maximum power allocated to each server. Limiting the maximum power allocated to a server is usually referred to as *capping*

the power allocated to a server. We run LINPACK jobs back-to-back to ensure that the server is always occupied by the workload, and that the server is running at the specified power cap value. Hence, the power values we observe will be the peak power values for the specified workload. Recall from Section 2 that the voltage and frequency scaling mechanisms have certain discrete performance points (in terms of frequency) at which the server can operate. At each of these performance points, the server consumes a certain amount of power for a given workload. By quickly dithering between available performance states, we can ensure that the server never consumes more than the set power cap value. In this way, we also get the best performance from the server for the given power cap value. Note that when we say power, we mean the *system-level power* which includes the power consumed by the processor and all other components within the server.

Fig. 2(a) illustrates the power-to-frequency curves obtained for LINPACK using DFS and DVFS. From the figure, we see that the power-to-frequency relationship for both DFS and DVFS is almost linear. It may seem surprising that the power-to-frequency relationship for DVFS looks like a linear plot. This is opposite to what is widely suggested in the literature for the *processor* power-to-frequency relationship, which is cubic [11]. The reason why the *server* power-to-frequency relationship is linear can be explained by two interrelated factors. First, manufacturers usually settle on a limited number of allowed voltage levels (or performance states), which results in a less-than-ideal relationship between power and frequency in practice. Second, DVFS is not applied on many components at the system level. For example, power consumption in memory remains proportional to the number of references to memory per unit time, which is only linearly related to the frequency of the processor. Thus, the power-to-frequency curve for both DFS and DVFS can be approximated as a linear function. Using the terminology introduced in Section 2, we approximate the server speed (or frequency), s GHz, as a function of the power allocated to it, $\mathcal{P}$ Watts, as:

$$s = s_b + \alpha(\mathcal{P} - b). \tag{1}$$

where the coefficient α (units of GHz per Watt) is the slope of the power-to-frequency curve. Specifically, our experiments show that $\alpha = 0.008$ GHz/W for DVFS and $\alpha = 0.03$ GHz/W for DFS. Also, from our experiments, we find that $b = 180$W and $c = 240$W for both DVFS and DFS. However, $s_b = 2.5$ GHz for DVFS and $s_b = 1.2$ GHz for DFS. The maximum speed in both cases is $s_c = 3$ GHz, which is simply the maximum speed of the processor we use. Note that the specific values of these parameters change depending on the workload in use. Section 6 discusses our measured parameter values for different workloads.

For DVFS+DFS, we expect the power-to-frequency relationship to be piecewise linear since it is a combination of DVFS and DFS. Experimentally, we see from Fig. 2(b) that the power-to-frequency relationship is in fact piecewise linear (3 GHz - 2.5 GHz and then 2.5 GHz - 1.2 GHz).

Though we could use a piecewise linear fit for DVFS+DFS, we choose to approximate it using a cubic curve for the following reasons:

1. Using a cubic fit demonstrates how we can extend our results to non-linear power-to-frequency relationships.
2. As previously mentioned, several papers consider the power-to-frequency relationship to be cubic, especially for the processor. By using a cubic model for DVFS+DFS, we wish to analyze the optimal power allocation policy for those settings.

Approximating DVFS+DFS using a cubic fit gives the following relationship between the speed of a server and the power allocated to it:

$$s = s_b + \alpha' \sqrt[3]{\mathcal{P} - b}. \tag{2}$$

Specifically, our experiments show that $\alpha' = 0.39$ GHz$/\sqrt[3]{W}$. Also, from our experiments, we found that $b = 150$W, $c = 250$W, $s_b = 1.2$ GHz and $s_c = 3$ GHz for DVFS+DFS.

4. THEORETICAL RESULTS

The optimal power allocation depends on a large number of factors including the power-to-frequency relationship just discussed in Fig. 2, the arrival rate, the minimum and maximum power consumption levels (b and c respectively), whether the server farm has an open loop configuration or a closed loop configuration, etc.

In order to investigate the effects of all these factors on the optimal power allocation, we develop a queueing theoretic model which predicts the mean response time as a function of all the above factors (See Section 4.1). We then produce theorems that determine the optimal power allocation for every possible configuration of the above factors, including open loop configuration (See Theorems 1 and 2 in Section 4.2) and closed loop configuration (See Theorems 3, 4 and 5 in Section 4.3).

4.1 Queueing model

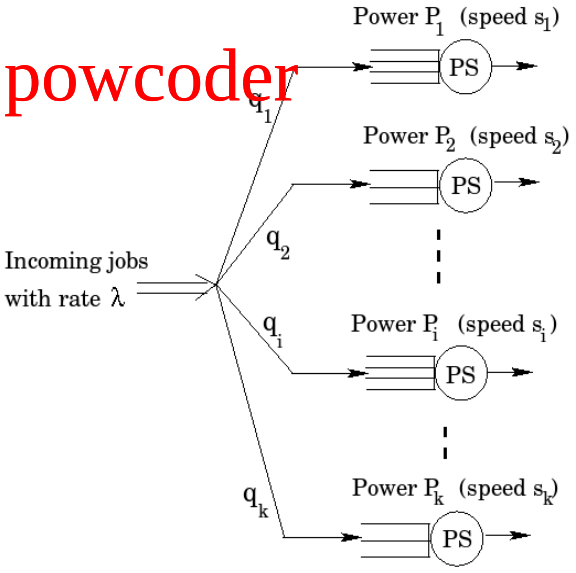
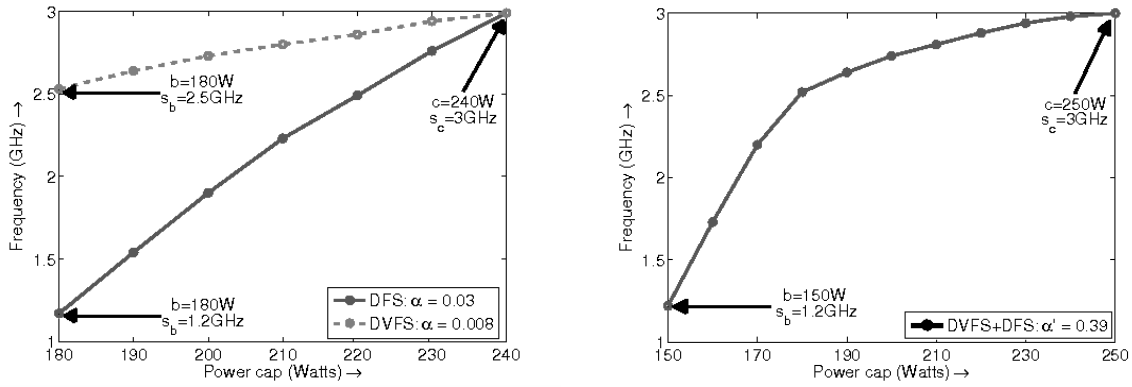


Figure 3: ***Illustration of our k-server farm model***

Fig. 3 illustrates our queueing model for a server farm with k servers. We assume that there is a fixed power budget $\mathcal{P}$, which can be split among the k servers, allocating $\mathcal{P}_i$ power to server i where $\sum_{i=1}^{k} \mathcal{P}_i = \mathcal{P}$. The corresponding server speeds are denoted by $(s_1, \ldots, s_k)$. Each server,



(a) DFS and DVFS (b) DVFS+DFS

Figure 2: ***Power-to-frequency curves for DFS, DVFS, and DVFS+DFS for the CPU bound LINPACK workload. Fig.(a) illustrates our measurements for DFS and DVFS. In both these mechanisms, we see that the server frequency is linearly related to the power allocated to the server. Fig.(b) illustrates our measurements for DVFS+DFS, where the power-to-frequency curve is better approximated by a cubic relationship.***

i, receives a fraction q_i of the total workload coming in to the server farm. Corresponding to any vector of power allocation $(\mathcal{P}_1, \ldots, \mathcal{P}_k)$, there exists an optimal workload allocation vector $(q_1^*, \ldots, q_k^*)$. We derive the optimal workload allocation for each power allocation and use that vector, $(q_1^*, \ldots, q_k^*)$, both in theory and in the actual experiments. The details of how we obtain the optimal $(q_1^*, \ldots, q_k^*)$ are deferred to [12].

Our model assumes that the jobs at a server are scheduled using the Processor-Sharing (PS) scheduling discipline. Under PS, when there are n jobs at a server, they each receive $1/n^{th}$ of the server capacity. PS is identical to Round-Robin with quantums (as in Linux), when the quantum size approaches zero. A job's response time, T, is the time from when the job arrives until it has completed service, including waiting time. We aim to minimize *mean response time*, $E[T]$.

We will analyze our server farm model under both an open loop configuration (See Section 4.2) and a closed loop configuration (See Section 4.3). An *open loop configuration* is one in which jobs arrive from outside the system and leave the system after they complete service. We assume that the arrival process is Poisson with average rate λ jobs/sec. Sometimes it will be convenient to, instead, express λ in units of GHz. This conversion is easily achievable since an average job has size $E[S]$ gigacycles. In the theorems presented in the paper, λ is in the units of GHz. However, in the proofs in [12], when convenient for queueing analysis, we switch to jobs/sec. Likewise, while it is common for us to express the speed of the server, s, in GHz, we sometimes switch to jobs/sec in [12] when convenient. A *closed loop configuration* is one in which there are always a fixed number of users N (also referred to as the multi-programming level) who each submit one job to the server. Once a user's job is completed, he immediately creates another job, keeping the number of jobs constant at N.

In all of the theorems that follow, we find the optimal power allocation, $(\mathcal{P}_1^*, \mathcal{P}_2^*, \ldots, \mathcal{P}_k^*)$, for a k-server farm, which minimizes the mean response time, $E[T]$, given the fixed peak power budget $\mathcal{P} = \sum_{i=1}^{k} \mathcal{P}_i^*$. While deriving the optimal power allocation is non-trivial, computing $E[T]$ for a given allocation is easy. Hence we omit showing the mean response time in each case and refer the reader to [12]. Due to lack of space, we defer all proofs to [12]. However, we present the intuition behind the theorems in each case. Recall from Section 2 that each fully utilized server has a minimum power consumption of b Watts and maximum power consumption of c Watts.

To illustrate our results clearly, we shall assume throughout this section that the power budget $\mathcal{P}$ is such that PowMax allows us to run n servers (each at power c) and PowMin allows us to run m servers (each at power b). This is equivalent to saying:

$$\mathcal{P} = m \cdot b = n \cdot c \tag{3}$$

where m and n are less than or equal to k. Obviously, $m \geq n$.

4.2 Theorems for open loop configurations

Theorem 1 derives the optimal power allocation in an open loop configuration for a linear power-to-frequency relationship, as is the case for DFS and DVFS. In such cases, the server frequency varies with the power allocated to it as $s_i = s_b + \alpha(\mathcal{P}_i - b)$. The theorem says that if the speed at baseline, s_b, is sufficiently low, then PowMax is optimal. By contrast, if s_b is high, then PowMin is optimal for high arrival rates and PowMax is optimal for low arrival rates. If s_i^* is the speed of server i when run at power $\mathcal{P}_i^*$, then the stability condition requires that $\lambda < \sum_{i=1}^{k} s_i^*$.

THEOREM 1. *Given an open k-server farm configuration with a linear power-to-frequency relationship (given by Eq. (1)) and power budget $\mathcal{P}$, the following power allocation minimizes $E[T]$:*

If $\frac{s_b}{b} \leq \alpha$: $\mathcal{P}^*_{1,2,..,n} = c$, $\mathcal{P}^*_{n+1,n+2,..,k} = 0$

If $\frac{s_b}{b} > \alpha$: $\begin{cases} \mathcal{P}^*_{1,2,..,n} = c, \mathcal{P}^*_{n+1,n+2,..,k} = 0 & \text{if } \lambda \leq \lambda_{low} \\ \mathcal{P}^*_{1,2,..,m} = b, \mathcal{P}^*_{m+1,m+2,..,k} = 0 & \text{if } \lambda > \lambda_{low} \end{cases}$

where $\lambda_{low} = \alpha \cdot \mathcal{P}$.

COROLLARY 1. *For DFS, PowMax is optimal. For DVFS, PowMax is optimal at low arrival rates and PowMin is optimal at high arrival rates.*

Intuition For a linear power-to-frequency relationship, we have from Eq. (1) that the speed of a server, s_i, varies with the power allocated to it, $\mathcal{P}_i$, as $s_i = s_b + \alpha(\mathcal{P}_i - b)$. From this equation, it follows that the frequency per Watt for a single server, $\frac{s_i}{\mathcal{P}_i}$, can be written as:

$$\frac{s_i}{\mathcal{P}_i} = \frac{s_b - \alpha b}{\mathcal{P}_i} + \alpha.$$

Hence, maximizing the frequency per Watt depends on whether $s_b \leq \alpha b$ or $s_b > \alpha b$. If $s_b \leq \alpha b$, maximizing $\frac{s_i}{\mathcal{P}_i}$ is equivalent to maximizing $\mathcal{P}_i$, which is achieved by PowMax. Alternatively, if $s_b > \alpha b$, we want to minimize $\mathcal{P}_i$, which is achieved by PowMin. However, the above argument still does not take into account the mean arrival rate, λ. If λ is sufficiently low, there are very few jobs in the server farm, hence, few fast servers, or PowMax, is optimal. The corollary follows by simply plugging in the values of s_b, α and b for DFS and DVFS from Section 3.

Theorem 2 derives the optimal power allocation for non-linear power-to-frequency relationships, such as the cubic relationship in the case of DVFS+DFS. In such cases, the server frequency varies with the power allocated to it as $s_i = s_b + \alpha' \sqrt[3]{\mathcal{P}_i - b}$. The theorem says that if the arrival rate is sufficiently low, then PowMax is optimal. However, if the arrival rate is high, PowMed is optimal. Although Theorem 2 specifies a cubic power-to-frequency relationship, we conjecture that similar results hold for more general power-to-frequency curves where server frequency varies as the n-th root of the power allocated to the server.

THEOREM 2. *Given an open k-server farm configuration with a cubic power-to-frequency relationship (given by Eq. (2)) and power budget $\mathcal{P}$, the following power allocation minimizes $E[T]$:*

$$\begin{array}{llll} \mathcal{P}^*_{1,2,..,n} = c, & \mathcal{P}^*_{n+1,n+2,..,k} = 0 & \text{if } \lambda \leq \lambda'_{low} \\ \mathcal{P}^*_{1,2,..,l} = \frac{\mathcal{P}}{l}, & \mathcal{P}^*_{l+1,l+2,..,k} = 0 & \text{if } \lambda > \lambda'_{low} \end{array}.$$

where $\lambda'_{low} = \frac{nl\alpha'}{l-n}\left(\sqrt[3]{c-b} - \sqrt[3]{\frac{\mathcal{P}}{l}} - b\right)$ *and* $l = \left\lfloor \frac{\mathcal{P}}{b + \left(\frac{\alpha' \mathcal{P}}{3\lambda}\right)^{\frac{3}{2}}} \right\rfloor$.

COROLLARY 2. *For DVFS+DFS, PowMax is optimal at low arrival rates and PowMed is optimal at high arrival rates.*

Intuition When the arrival rate is sufficiently low, there are very few jobs in the system, hence, PowMax is optimal. However, for higher arrival rates, we allocate to each server the amount of power that maximizes its frequency per Watt ratio. For the cubic power-to-frequency relationship, which has a *downwards concave* curve (See Fig. 2(b)), we find that the optimal power allocation value for each server (derived via calculus) lies between the maximum c and the minimum b. Hence, PowMed is optimal.

4.3 Theorems for closed loop configurations

We now move on to closed loop configurations, where the number of jobs in the system, N, is always constant. We will rely on asymptotic operational laws (See [14]) which approximate the performance of the system for very high N and very low N (see [12] for details).

Theorem 3 says that for a closed server farm configuration with sufficiently low value of N, PowMax is optimal.

THEOREM 3. *Given a closed k-server farm configuration with a linear or cubic power-to-frequency relationship (given by Eqs. (1) and (2)), the following power allocation minimizes $E[T]$ for low N, based on the asymptotic approximations in [14]:*

$$\mathcal{P}^*_{1,2,..,n} = c, \quad \mathcal{P}^*_{n+1,n+2,..,k} = 0$$

COROLLARY 3. *For a closed-loop server farm configuration with low N, PowMax is optimal for DFS, DVFS, and DVFS+DFS.*

Intuition When N is sufficiently low, there are very few jobs in the system. Hence few fast servers are optimal since there aren't enough jobs to utilize the servers, leaving servers idle. Thus PowMax is optimal. This is similar to the case of low arrival rate that we considered for an open loop server farm configuration in Theorems 1 and 2.

When N is high, the optimal power allocation is non-trivial. From here on, we assume N is large enough to keep all servers busy, so that asymptotic bounds apply (See [14]). In our experiments, we find that $N > 10k$ suffices.

Theorem 4 says that for high N, if the speed at baseline, s_b, is sufficiently low, then PowMax is optimal. By contrast, if s_b is high, then PowMin is optimal.

THEOREM 4. *Given a closed k-server farm configuration with a linear power-to-frequency relationship (given by Eq. (1)), the following power allocation minimizes $E[T]$ for high N, based on the asymptotic approximations in [14]:*

$$\begin{array}{lll} \text{If } \frac{s_b}{b} < \alpha\text{:} & \mathcal{P}^*_{1,2,..,n} = c, & \mathcal{P}^*_{n+1,n+2,..,k} = 0 \\ \text{If } \frac{s_b}{b} \geq \alpha\text{:} & \mathcal{P}^*_{1,2,..,m} = b, & \mathcal{P}^*_{m+1,m+2,..,k} = 0 \end{array}$$

COROLLARY 4. *For DFS, PowMax is optimal for high N. For DVFS, PowMin is optimal for high N.*

Intuition For a closed queueing system with zero think time, the mean response time is inversely proportional to the throughput of the system. Hence, to minimize the mean response time we must maximize the throughput of the k-server farm with power budget $\mathcal{P}$. When N is high, under a server farm configuration, all servers are busy. Hence the throughput is the sum of the server speeds. It can be easily shown that the throughput of the system under PowMin, $s_b \cdot m$, exceeds the throughput of the system under PowMax, $s_c \cdot n$, when $s_b \geq \alpha b$. Hence the result.

Theorem 5 deals with the case of high N for a non-linear power-to-frequency relationship. The theorem says that if the speed at baseline, s_b, is sufficiently low, then PowMax is optimal. By contrast, if s_b is high, then PowMed is optimal.

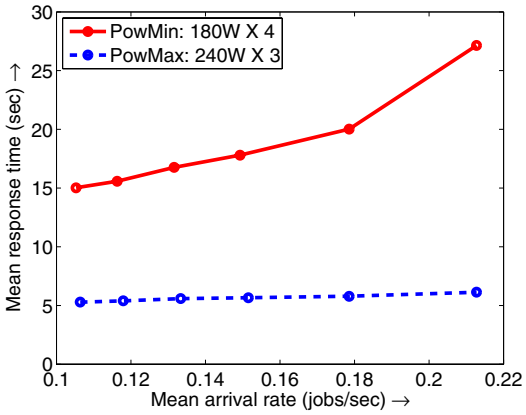
THEOREM 5. *Given a closed k-server farm configuration with a cubic power-to-frequency relationship (given by Eq. (2)), the following power allocation minimizes $E[T]$ for high N, based on the asymptotic approximations in [14]:*

$$\begin{array}{lll} \text{If } s_b < s'\text{:} & \mathcal{P}^*_{1,2,..,n} = c, & \mathcal{P}^*_{n+1,n+2,..,k} = 0 \\ \text{If } s_b \geq s'\text{:} & \mathcal{P}^*_{1,2,..,l} = b + x, & \mathcal{P}^*_{l+1,l+2,..,k} = 0 \end{array}$$

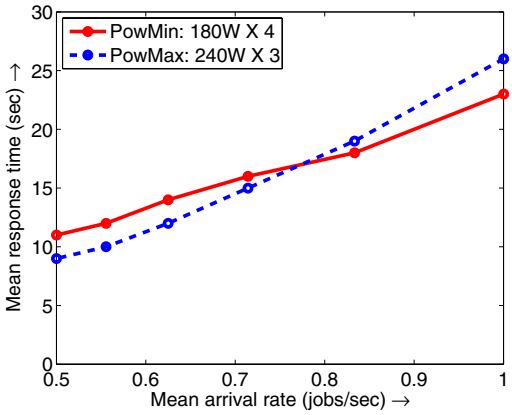
where $l = \left\lfloor \frac{\mathcal{P}}{b+x} \right\rfloor$, $s' = \frac{ms_c}{l} - \alpha' \sqrt[3]{x}$ *and x is the non-negative, real solution of the equation* $b = 2x + \frac{1}{\alpha'}(3x^{\frac{2}{3}} s_b)$.

COROLLARY 5. *For DVFS+DFS, for high N, PowMed is optimal if s_b is high, else PowMax is optimal.*

Intuition As in the case of Theorem 4, we wish to maximize the throughput of the system. When we turn on a new



(a) DFS



(b) DVFS

Figure 4: ***Open loop experimental results for mean response time as a function of the arrival rate using DFS and DVFS for LINPACK. In Fig.(a) PowMax outperforms PowMin for all arrival rates under DFS, by as much as a factor of 5. By contrast in Fig.(b), for DVFS, at lower arrival rates, PowMax outperforms PowMin by up to 22%, while at higher arrival rates, PowMin outperforms PowMax by up to 14%.***

server at b units of power, the increase in throughput of the system is s_b GHz. For low values of s_b, this increase is small. Hence, for low values of s_b, we wish to turn on as few servers as possible. Thus, PowMax is optimal. However, when s_b is high, it pays to turn servers on. Once a server is on, the initial steep increase in frequency per Watt afforded by a cubic power-to-frequency relationship advocates running the server at more than the minimal power, b. The exact optimal PowMed power value ($b + x$ in the Theorem) is close to the knee of the cubic power-to-frequency curve.

5. EXPERIMENTAL RESULTS

In this section we test our theoretical results from Section 4 on an IBM BladeCenter using the experimental setup discussed in Section 2. We shall first present our experimental results for the open server farm configuration and then move on to the closed server farm configuration. For the experiments in this section, we use the Intel LINPACK [13] workload, which is CPU bound. We defer experimental results for other workloads to Section 6.

As noted in Section 3, the baseline power level and the maximum power level for both DFS and DVFS are $b = 180$W and $c = 240$W respectively. For DVFS+DFS, $b = 150$W and $c = 250$W. In each of our experiments, we try to fix the power budget, $\mathcal{P}$, to be an integer multiple of b and c, as in Eq. (3).

5.1 Open server farm configuration

Fig. 4(a) plots the mean response time as a function of the arrival rate for DFS with a power budget of $\mathcal{P} = 720$W. In this case, PowMax (represented by the dashed line) denotes running 3 servers at $c = 240$W and turning off all other servers. PowMin (represented by the solid line) denotes running 4 servers at $b = 180$W and turning off all other servers. Clearly, PowMax outperforms PowMin throughout the range of arrival rates. This is in agreement with the predictions of Theorem 1. Note from Fig. 4(a) that the improvement in mean response time afforded by PowMax over PowMin is huge; ranging from a factor of 3 at low arrival rates (load, $\rho \approx 0.2$) to as much as a factor of 5 at high arrival rates (load, $\rho \approx 0.7$). This is because the power-to-frequency relationship for DFS is steep (See Fig. 2(a)), hence running servers at maximum power levels affords a huge gain in server frequency. Arrival rates higher than 0.22 jobs/sec cause our systems to overload under PowMin because s_b is very low for DFS. Hence, we only go as high as 0.22 jobs/sec.

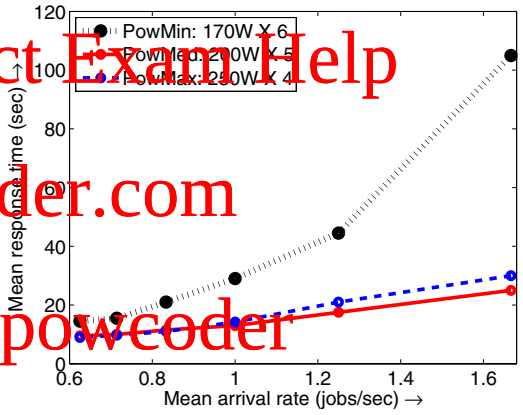
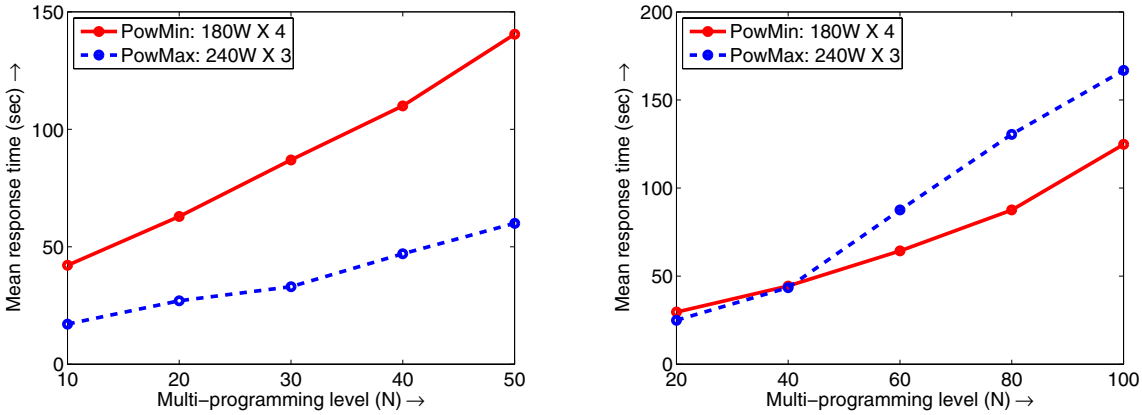


Figure 5: ***Open loop experimental results for mean response time as a function of the arrival rate using DVFS+DFS for LINPACK. At lower arrival rates, PowMax outperforms PowMed by up to 12%, while at higher arrival rates, PowMed outperforms PowMax by up to 20%. Note that PowMin is worse than both PowMed and PowMax throughout the range of arrival rates.***

Fig. 4(b) plots the mean response time as a function of the arrival rate for DVFS with a power budget of $\mathcal{P} = 720$W. PowMax (represented by the dashed line) again denotes running 3 servers at $c = 240$W and turning off all other servers.



(a) DFS (b) DVFS

Figure 6: ***Closed loop experimental results for mean response time as a function of number of jobs*** (N) ***in the system using DFS and DVFS for LINPACK. In Fig.(a), for DFS, PowMax outperforms PowMin for all values of*** N***, by almost a factor of 2 throughout. By contrast in Fig.(b), for DVFS, at lower values of*** N***, PowMax is slightly better than PowMin, while at higher values of*** N***, PowMin outperforms PowMax by almost 30%.***

PowMin (represented by the solid line) denotes running 4 servers at $b = 180$W and turning off all other servers. We see that when the arrival rate is low, PowMax produces lower mean response times than PowMin. In particular, when the arrival rate is 0.5 jobs/sec, PowMax affords a 22% improvement in mean response time over PowMin. However, at higher arrival rates, PowMin outperforms PowMax, as predicted by Theorem 1. In particular, when the arrival rate is 1 job/sec, PowMin affords a 14% improvement in mean response time over PowMax. Under DVFS, we can afford arrival rates up to 1 job/sec before overloading the system. To summarize, under DVFS, we see that PowMin can be preferable to PowMax. This is due to the flatness of the power-to-frequency curve for DVFS (See Fig. 2(a)), and agrees perfectly with Theorem 1.

Fig. 5 plots the mean response time as a function of the arrival rate for DVFS+DFS with a power budget of $\mathcal{P} = 1000$W. In this case, PowMax (represented by the dashed line) denotes running 4 servers at $c = 250$W and turning off all other servers. PowMed (represented by the solid line) denotes running 5 servers at $\frac{b+c}{2} = 200$W and turning off all other servers. We see that when the arrival rate is low, PowMax produces lower mean response times than PowMed. However, at higher arrival rates, PowMed outperforms PowMax, exactly as predicted by Theorem 2. For the sake of completion, we also plot PowMin (dotted line in Fig. 5). Note that PowMin is worse than both PowMed and PowMax throughout the range of arrival rates. Note that we use the value of $\frac{b+c}{2} = 200W$ as the optimal power allocated to each server in PowMed for our experiments as this value is close to the theoretical optimum predicted by Theorem 2 (which is around 192W for the range of arrival rates we use) and also helps to keep the power budget at 1000W .

5.2 Closed server farm configuration

We now turn to our experimental results for closed server farm configurations. Fig. 6(a) plots the mean response time as a function of the multi-programming level (MPL = N) for DFS with a power budget of $\mathcal{P} = 720$W. In this case,

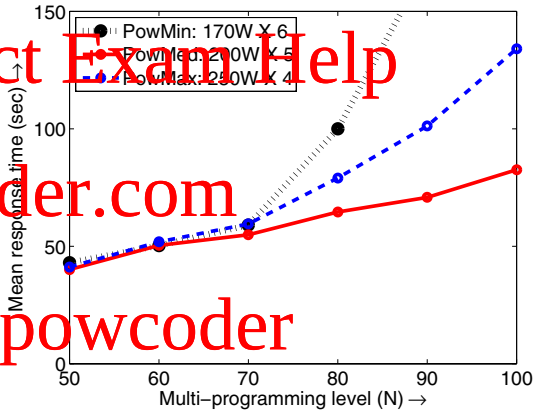
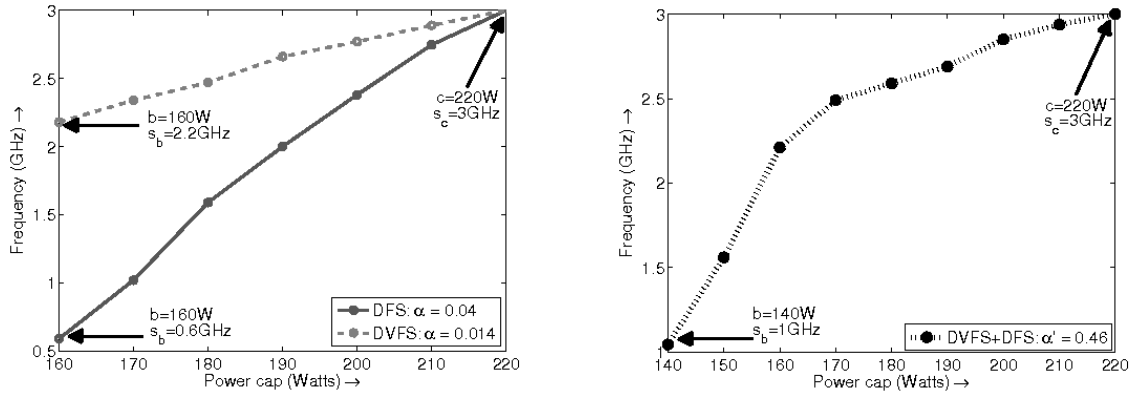


Figure 7: ***Closed loop experimental results for mean response time as a function of number of jobs*** (N) ***in the system using DVFS+DFS for LINPACK. At lower values of*** N***, PowMed is slightly better than PowMax, while at higher values of*** N***, PowMed outperforms PowMax, by as much as 40%. Note that PowMin is worse than both PowMed and PowMax for all values of*** N***.***

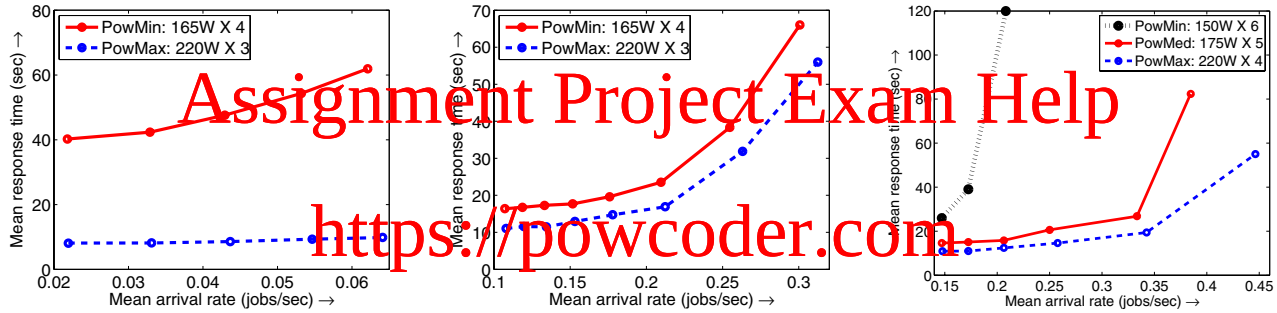
PowMax (represented by the dashed line) denotes running 3 servers at $c = 240$W and turning off all other servers. PowMin (represented by the solid line) denotes running 4 servers at $b = 180$W and turning off all other servers. Clearly, PowMax outperforms PowMin throughout the range of N, by almost a factor of 2 throughout the range. This is in agreement with the predictions of Theorem 3.

Fig. 6(b) plots the mean response time as a function of the multi-programming level for DVFS with a power budget of $\mathcal{P} = 720$W. PowMax (represented by the dashed line) again denotes running 3 servers at $c = 240$W and turning off all other servers. PowMin (represented by the solid line)



(a) DFS and DVFS (b) DVFS+DFS

Figure 8: ***Power-to-frequency curves for DFS, DVFS, and DVFS+DFS for the CPU bound DAXPY workload. Fig.(a) illustrates our measurements for DFS and DVFS. In both these mechanisms, we see that the server frequency is linearly related to the power allocated to the server. Fig.(b) illustrates our measurements for DVFS+DFS, where the power-to-frequency curve is better approximated by a cubic relationship.***



(a) DFS (b) DVFS (c) DVFS+DFS

Figure 9: ***Open loop experimental results for mean response time as a function of the arrival rate using DFS, DVFS, and DVFS+DFS for the CPU bound DAXPY workload. In Fig.(a), for DFS, PowMax outperforms PowMin throughout the range of arrival rates by as much as a factor of 5. In Fig.(b), for DVFS, PowMax outperforms PowMin throughout the range of arrival rates by around 30%. In Fig.(c), for DVFS+DFS, PowMax outperforms both PowMed and PowMin throughout the range of arrival rates. While at lower arrival rates PowMax only slightly outperforms PowMed, at higher arrival rates the improvement is around 60%.***

denotes running 4 servers at $b = 180$W and turning off all other servers. We see that when N is high, PowMin produces lower mean response times than PowMax. This is in agreement with the predictions of Theorem 4. In particular, when $N = 100$, PowMin affords a 30% improvement in mean response time over PowMax. However, when N is low, PowMax produces slightly lower response times than PowMin. This is in agreement with Theorem 3.

Fig. 7 plots the mean response time as a function of the multi-programming level for DVFS+DFS with a power budget of $\mathcal{P} = 1000$W. In this case, PowMax (represented by the dashed line) denotes running 4 servers at $c = 250$W and turning off all other servers. PowMed (represented by the solid line) denotes running 5 servers at $\frac{b+c}{2} = 200$W and turning off all other servers. PowMin (represented by the dotted line) denotes running 6 servers at 170W. We see that when N is high, PowMed produces lower mean response times than PowMax. This is in agreement with the predictions of Theorem 5. In particular, when $N = 100$, PowMed affords a 40% improvement in mean response time over PowMax. However, when N is low, PowMed produces only slightly lower response times than PowMax. Note that throughout the range of N, PowMin is outperformed by both PowMax and PowMed.

6. OTHER WORKLOADS

Thus far we have presented experimental results for a CPU bound workload LINPACK. In this section we present experimental results for other workloads. Our experimental results agree with our theoretical predictions *even in the case of non-CPU bound workloads.*

We fully discuss experimental results for two workloads, DAXPY and STREAM, in this section and summarize our results for other workloads at the end of the section. Due to lack of space, we only show open loop configuration results.

DAXPY

DAXPY [22] is a CPU bound workload which we have sized to be L1 cache resident. This means DAXPY uses a lot of processor and L1 cache but rarely uses the server memory and disk subsystems. Hence, the power-to-frequency relationship for DAXPY is similar to that of CPU bound LINPACK except that DAXPY's peak power consumption tends to be lower than that of LINPACK, since DAXPY does not use a lot of memory or disk.

Figs. 8(a) and (b) present our results for the power-to-frequency relationship for DAXPY. The functional form of the power-to-frequency relationship under DFS and DVFS in Fig. 8(a) is clearly linear. However, the power-to-frequency relationship under DVFS+DFS in Fig. 8(b) is better approximated by a cubic relationship. These trends are similar to the power-to-frequency relationship for LINPACK seen in Fig. 2.

Figs. 9(a), (b) and (c) present our power allocation results for DAXPY under DFS, DVFS, and DVFS+DFS respectively. For DFS, in Fig. 9(a), PowMax outperforms PowMin throughout the range of arrival rates, by as much as a factor of 5. This is in agreement with Theorem 1. Note that we use 165W as the power allocated to each server under PowMin to keep the power budget same for PowMin and PowMax. For DVFS, in Fig. 9(b), PowMax outperforms PowMin throughout the range of arrival rates, by around 30%. This is in contrast to LINPACK, where PowMin outperforms PowMax at high arrival rates. The reason why PowMax outperforms PowMin for DAXPY is the lower value of $s_b = 2.2$ GHz for DAXPY as compared to $s_b = 2.5$ GHz for LINPACK. Since $\frac{s_b}{b} = 0.0137 < \alpha = 0.014$ for DAXPY under DVFS, Theorem 1 rightly predicts PowMax to be optimal. Finally, in Fig. 9(c) for DVFS+DFS, PowMax outperforms both PowMed and PowMin throughout the range of arrival rates. Again, this is in contrast to LINPACK, where PowMed outperforms PowMax at high arrival rates. The reason why PowMax outperforms PowMed for DAXPY is the higher value of $\alpha' = 0.46$ GHz/$\sqrt[3]{W}$ for DAXPY as compared to $\alpha' = 0.39$ GHz/$\sqrt[3]{W}$ for LINPACK. This is in agreement with the predictions of Theorem 2 for high values of α'. Intuitively, for a cubic power-to-frequency relationship, we have from Eq. (2): $s = s_b + \alpha' \sqrt[3]{\mathcal{P} - b}$. As α' increases, we get more server frequency for every Watt of power added to the server. Thus, at high α', we allocate as much power as possible to every server, implying PowMax.

STREAM

STREAM [18] is a memory bound workload which does not use a lot of processor cycles. Hence, the power consumption at a given server frequency for STREAM is usually lower than CPU bound LINPACK and DAXPY.

Figs. 10(a) and (b) present our results for the power-to-frequency relationship for STREAM. Surprisingly, the functional form of the power-to-frequency relationship under DFS, DVFS, and DVFS+DFS is closer to a *cubic relationship* than to a linear one. In particular, the gain in server frequency per Watt at higher power allocations is much lower than the gain in frequency per Watt at lower power allocations. We argue this observation as follows: At extremely low server frequencies, the bottleneck for STREAM's performance is the CPU. Thus, every extra Watt of power added to the system would be used up by the CPU to improve its frequency. However, at higher server frequencies, the bottleneck for STREAM's performance is the memory subsystem since STREAM is memory bound. Thus, every extra Watt of power added to the system would mainly be used up by the memory subsystem and the improvement in processor frequency would be minimal.

Figs. 11(a), (b) and (c) present our power allocation results for STREAM under DFS, DVFS, and DVFS+DFS respectively. Due to the downwards concave nature of the power-to-frequency curves for STREAM studied in Fig. 10, Theorem 2 says that PowMax should be optimal at low arrival rates and PowMed should be optimal at high arrival rates. However, for the values of α' in Fig. 10, we find that the threshold point, λ_{low}, below which PowMax is optimal, is quite high. Hence, PowMax is optimal in Fig. 11(c). In Figs. 11(a) and (b), PowMax and PowMed produce similar response times.

GZIP and BZIP2

GZIP and BZIP2 are common software applications used for data compression in Unix systems. These CPU bound compression applications use sophisticated algorithms to reduce the size of a given file. We use GZIP and BZIP2 to compress a file of uncompressed size 150 MB. For GZIP, we find that PowMax is optimal for all of DFS, DVFS, and DVFS+DFS. These results are similar to the results for DAXPY. For BZIP2, the results are similar to those of LINPACK. In particular, at low arrival rates, PowMax is optimal. For high arrival rates, PowMax is optimal for DFS, PowMin is optimal for DVFS and PowMed is optimal for DVFS+DFS.

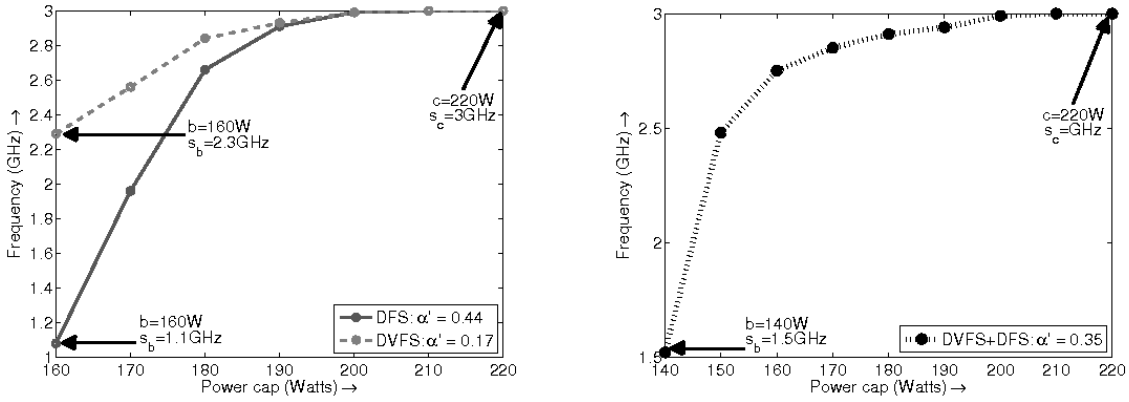
WebBench

WebBench [15] is a benchmark program used to measure web server performance by sending multiple file requests to a server. For WebBench, we find the power-to-frequency relationship for DFS, DVFS, and DVFS+DFS to be cubic. This is similar to the power-to-frequency relationships observed for STREAM since WebBench is more memory and disk intensive. As theory predicts (See Theorem 2), we find PowMax to be optimal at low arrival rates and PowMed to be optimal at high arrival rates for DFS, DVFS, and DVFS+DFS.

7. SUMMARY

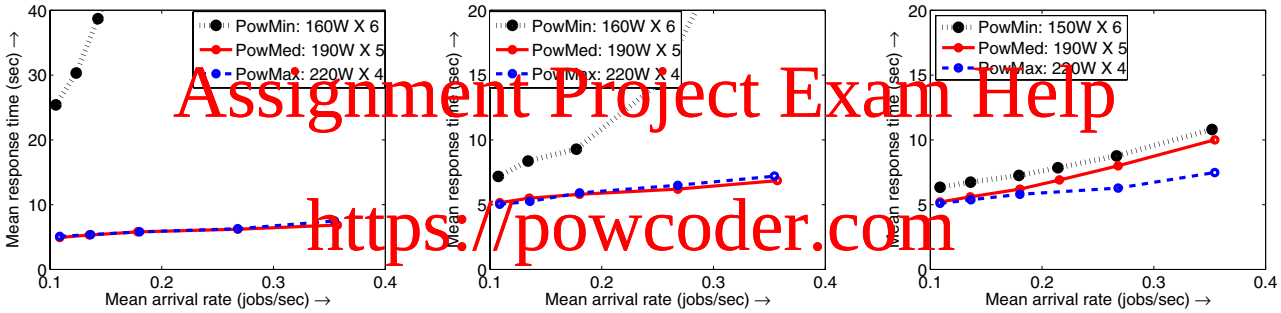
In this paper, we consider the problem of allocating an available power budget among servers in a server farm to minimize mean response time. The amount of power allocated to a server determines its speed in accordance to some power-to-frequency relationship. Hence, we begin by measuring the power-to-frequency relationship within a single server. We experimentally find that the power-to-frequency relationship within a server for a given workload can be either linear or cubic. Interestingly, we see that the relationship is linear for DFS and DVFS when the workload is CPU bound, but cubic when it is more memory bound. By contrast, the relationship for DVFS+DFS is always cubic in our experiments.

Given the power-to-frequency relationship, we can view the problem of finding the optimal power allocation in terms of determining the optimal frequencies of servers in the server farm to minimize mean response time. However, there are several factors apart from the server frequencies that affect the mean response time for a server farm. These include the arrival rate, the maximum speed of a server, the total power budget, whether the server farm has an open or closed configuration, etc. To fully understand the effects of these factors on mean response time, we develop a queueing theo-



(a) DFS and DVFS (b) DVFS+DFS

Figure 10: ***Power-to-frequency curves for DFS, DVFS, and DVFS+DFS for the memory bound STREAM workload. Fig.(a) illustrates our measurements for DFS and DVFS, while Fig.(b) illustrates our measurements for DVFS+DFS. In all the three mechanisms, the power-to-frequency curves are downwards concave, depicting a cubic relationship between power allocated to a server and its frequency.***



(a) DFS (b) DVFS (c) DVFS+DFS

Figure 11: ***Open loop experimental results for mean response time as a function of the arrival rate using DFS, DVFS, and DVFS+DFS for the memory bound STREAM workload. In Figs.(a) and (b), for DFS and DVFS respectively, PowMed and PowMax produce similar response times. In Fig.(c) however, for DVFS+DFS, PowMax outperforms PowMed by as much as 30% at high arrival rates. In all three cases, PowMin is worse than both PowMed and PowMax.***

retic model (See Section 4.1) that allows us to predict mean response time as a function of the above factors. We then produce theorems (See Sections 4.2 and 4.3) that determine the optimal power allocation for every possible configuration of the above factors.

To verify our theoretical predictions, we conduct extensive experiments on an IBM BladeCenter for a range of workloads using DFS, DVFS, and DVFS+DFS (See Section 5 and 6). In every case, we find that the experimental results are in excellent agreement with our theoretical predictions.

8. DISCUSSION AND FUTURE WORK

There are many extensions to this work that we are exploring, but are beyond the scope of this paper.

First of all, the arrival rate into our server farm may vary dynamically over time. In order to adjust to a dynamically varying arrival rate, we may need to adjust the power allocation accordingly. The theorems in this paper already tell us the optimal power allocation for any given arrival rate. We are now working on incorporating the effects of switching costs into our model.

Second, while we have considered turning servers on or off, today's technology [2, 6] allows for servers which are *sleeping* (HALT state or deep C states). These sleeping servers consume less power than servers that are on, and can more quickly be moved into the on state than servers that are turned off. We are looking at ways to extend our theorems to allow for servers with sleep states.

Third, while this paper deals with power management at the server level (measuring and allocating power to the server as a whole), our techniques can be extended to deal with individual subsystems within a server, such as power allocation within the storage subsystem. We are looking at extending our implementation to individual components within a server.

9. REFERENCES

[1] Lesswatts.org: Race to idle. http://www.lesswatts.org/projects/applications-power-management/race-to-idle.php.

[2] Intel: Nehalem. http://intel.wingateweb.com/US08/published/sessions/NGMS001/SF08_NGMS001_100t.pdf.

[3] U.S. Environmental Protection Agency. Epa report on server and data center energy efficiency. 2007.

[4] National Electrical Contractors Association. Data centers - meeting today's demand. 2007.

[5] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, and Amin M. Vahdat. Managing energy and server resources in hosting centers. In *In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP)*, pages 103–116, 2001.

[6] Intel Corp. Intel Core2 Duo Mobile Processor Datasheet: Table 20. http://download.intel.com/design/mobile/datashts/32012001.pdf, 2008.

[7] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for web servers. In *USITS*, 2003.

[8] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. pages 13–23, 2007.

[9] Wes Felter, Karthick Rajamani, Tom Keller, and Cosmin Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *ICS '05: Proceedings of the 19th annual International Conference on Supercomputing*, pages 293–302, New York, NY, USA, 2005. ACM.

[10] Mark E. Femal and Vincent W. Freeh. Boosting Data Center Performance Through Non-Uniform Power Allocation. In *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*, pages 250–261, Washington, DC, 2005.

[11] M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware. System Power Management Support in the IBM POWER6 Microprocessor. *IBM Journal of Research and Development*, 51:733–746, 2007.

[12] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. Optimal power allocation in server farms. *Technical Report CMU-CS-09-113*, 2009.

[13] Intel Corp. Intel Math Kernel Library 10.0 - LINPACK. http://www.intel.com/cd/software/products/asmo-na/eng/266857.htm.

[14] Raj Jain. *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. pages 563–567. Wiley, 1991.

[15] Radim Kolar. Web bench. http://home.tiscali.cz:8080/ cz210552/webbench.html.

[16] Kleinrock L. *Queueing Systems, Volume 2*. Wiley-Interscience, New York, 1976.

[17] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Power capping: a prelude to power shifting. *Cluster Computing*, November 2007.

[18] J.D. McCalpin. Stream: Sustainable memory bandwidth in high performance computers. http://www.cs.virginia.edu/stream/.

[19] David Mosberger and Tai Jin. httperf—A Tool for Measuring Web Server Performance. *ACM Sigmetrics: Performance Evaluation Review*, 26:31–37, 1998.

[20] Vivek Pandey, W. Jiang, Y. Zhou, and R. Bianchini. DMA-Aware Memory Energy Management. *HPCA '06: The 12th International Symposium on High-Performance Computer Architecture*, pages 133–144, 11-15 Feb. 2006.

[21] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "Power" Struggles: Coordinated Multi-Level Power Management for the Data Center. In *ASPLOS XIII: Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, pages 48–59, 2008.

[22] K. Rajamani, H. Hanson, J. C. Rubio, S. Ghiasi, and F. L. Rawson. Online power and performance estimation for dynamic power management. *Research Report RC-24007*, July 2006.

[23] Salvatore Sanfilippo. WBox HTTP testing tool (Version 4). http://www.hping.org/wbox/, 2007.

[24] X. Wang and M. Chen. Cluster-level Feedback Power Control for Performance Optimization. *14th IEEE International Symposium on High-Performance Computer Architecture (HPCA 2008)*, February 2008.

[25] Zhikui Wang, Xiaoyun Zhu, Cliff McCarthy, Partha Ranganathan, and Vanish Talwar. Feedback Control Algorithms for Power Management of Servers. In *Third International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks (FeBid)*, Annapolis, MD, June 2008.