Chapter 1

THE CASE FOR POWER MANAGEMENT IN WEB SERVERS

Pat Bohrer, Elmootazbellah N. Elnozahy, Tom Keller, Michael Kistler, Charles Lefurgy, Chandler McDowell, and Ram Rajamony
IBM Research, Austin TX 78758, USA.
http://www.research.ibm.com/arl

Abstract Power management has traditionally focused on portable and handheld devices. This paper breaks with tradition and presents a case for managing power consumption in web servers. Web servers experience large periods of low utilization, presenting an opportunity for using power management to reduce energy consumption with minimal performance impact. We measured the energy consumption of a "typical" web server under a variety of workloads derived from access logs of real websites, including the 1998 Winter Olympics web site. Our measurements show that the CPU is the largest consumer of power for typical web servers today.

We have also created a power simulator for web serving workloads that estimates CPU energy consumption with less than 5.7% error for our workloads. The simulator is fast, processing over 75,000 requests/second on a 866MHz uniprocessor machine. Using the simulator, we quantify the potential benefits of dynamically scaling the processor voltage and frequency, a power management technique that is traditionally found only in handheld devices. We find that dynamic voltage and frequency scaling is highly effective for saving energy with moderately intense web workloads, saving from 23% to 36% of the CPU energy while keeping server responsiveness within reasonable limits.

1. Introduction

There is a growing industry trend to "outsource" computing services to large data centers accessible through the World Wide Web (WWW). These data cen-

[0]This research has been supported in part by the Defense Advanced Research Projects Agency under contract F33615-00-C-1736.

ters use economies of scale to amortize the cost of ownership and system management over a large number of machines. A typical data center thus deploys hundreds or thousands of computers, densely packed to maximize floor space utilization. This form of server consolidation provides the customer with a more cost-effective approach than the alternative of operating the same services in-house.

Server consolidation, however, has its own problems as large-scale deployment pushes the limits of power supply and cooling systems in a data center. Anecdotal evidence from data center operators already points to the large contribution of power consumption and cooling to operation cost [24], and the intermittent failures of computing nodes in densely packed systems due to insufficient cooling. Furthermore, in many data centers, power supply to the server racks is a key inhibitor to increasing server density. One estimate, based on data from several modern Internet hosting facilities, puts the practical limit in power supply to a rack at around 7KW. This power is often insufficient to allow the rack to be fully populated with "1U" servers. Thus, power consumption increasingly carries an opportunity cost in lost revenue of additional systems. The problem is likely to worsen as newer server-class processors are projected to offer higher levels of performance at the expense of higher power consumption levels. This technology trend is a natural response to the growing needs for more performance. We therefore believe that these technology and market trends will make power management within servers a necessity.

Research in power management has primarily focused on portable and handheld systems. Such systems experience workloads that include substantial interactions with a single user, providing opportunities to turn parts of the system off when the user is "thinking" before issuing the next command. Typical energy saving techniques include turning off the system's display and disk, and slowing or halting the processor [18, 14].

This paper addresses power management in server systems. We focus on web server workloads because of the importance and widespread use of these servers in data centers. We first demonstrate that power management can provide significant benefits by examining workload characteristics from several real web sites, and then support this examination with measurements from an instrumented system in our lab running a workload derived from production Internet web sites. Then, we characterize the power consumption of the web server for these workloads, and use the measurements to validate a simulation model. We then use the simulation model to examine the effect of voltage and frequency scaling on the energy consumption of web servers.

This paper makes three contributions:

- Makes a case for power management in web servers and identifies the processor as the main system component where power management could yield energy savings.

- Introduces a power simulator for web serving workloads that is accurate to within 5.7% over the range of workloads we use in this paper.
- Evaluates the effectiveness of dynamic voltage scaling, an energy-saving technique typically found in handheld devices, using our simulation model.

There are several limitations to a study of this kind, including the inherent inaccuracies in measuring and modeling power consumption. In addition, web servers can be run on a wide variety of machines with vastly different power consumption characteristics. We have chosen to study a "commodity server" (often referred to as a "white box server"), since we believe this is the most common type of web server machine employed in commercial data centers. However, even commodity servers have differing power consumption characteristics, so while we believe our results are generally valid for this space of systems, there are certainly some exceptions.

The remainder of this paper is organized as follows. In Section 2 we present the case for power management in web servers. Section 3 explains our environment for data collection. Section 4 presents a characterization of power consumption in web servers. In Section 5 we describe a simulation model for evaluating various power management policies and our validation of this model. We present the results of simulating a power management policy that dynamically varies the processor voltage and frequency in Section 6. In Section 7 we discuss the implications of energy management on traditional web server performance metrics. A comparison to related work is presented in Section 8. Finally, Section 9 concludes the paper.

2. The Case for Power Management

Web site designers typically plan web server capacity to provide acceptable service even during periods of peak demand. Additional capacity may also be planned in clustered environments for high availability, where it may be necessary to redistribute the load of a failed server to the functioning ones. As a result, a web server may not be operating at its maximum capacity for substantial durations. We show by an analysis of several real internet web servers that this is true in practice. This model of operation has implications on the power consumption of web servers. We then study the efficiency of a web server as it operates under different workloads, and establish the case for power management in such servers.

2.1 Web Server Loads

Consider the design of the web site used for the 1998 Winter Olympics in Nagano [5]. The site consists of a three-tiered structure that is a common

	Avg req / sec	Max req / sec	Avg Server Utilization
Olympics98	459	1840	25%
Wimbledon	778	7166	11%
Finance	16	46	35%
Shopping	25	55	45%
Proxy	15	30	50%

Table 1.1. Analysis of real internet web server workloads

approach to constructing high–volume web sites. The first tier consists of edge servers that interface with clients. These servers provide firewalls, caching, workload distribution, and the HTTP interface for client interaction. The second tier consists of application servers, which perform functions such as content generation and personalization. The last tier is made up of traditional database servers that provide a reliable data repository with sophisticated processing capabilities. The site was replicated over four geographic areas to cover the globe and provide high availability.

The total traffic experienced by the site during the 16 days of the Olympics was roughly 634.7 million hits, and the site successfully handled a peak of over 110,000 hits for a one minute period during the women's freestyle figure skating event [5]. No failures were detected during the 16 days of the Olympics. Thus, while the peak workload was 1840 hits/sec, the average workload over the entire event was only 459 hits/sec. It follows that on average, the site was operating at about 25% of the observed peak capacity. Note that this is not a case of careless over-engineering. The distribution of the workload was not uniform, and depended on the obvious pattern of accesses from around the globe. It was therefore necessary to design the system to meet the peak demand. Well–engineered web sites follow the same design principles. This is particularly important in commercially oriented web sites, where customer dissatisfaction with the service's responsiveness may seriously affect the profitability of the enterprise.

The Winter Olympics example supports the notion that web servers are designed to handle peak loads. A further example is the 1999 Wimbledon web site which experienced an average of 778 hits/per second over two weeks, and a peak of 7166 hits/sec in a one–minute period, implying that the site was operating at only 11% of its observed peak capacity [5].

We have also analyzed the logs of web servers employed in a variety of real internet web sites and found very similar trends in the workload of these servers. These results and the examples given above are summarized in Table 1.1.

Several studies also provide evidence that web workloads are bursty in nature, and support the intuitive notion that web servers tend to be busiest during some

peak hours during the day and almost idle in others [4]. This model of operation has implications with respect to power consumption that we address next.

2.2 Energy Efficiency of Web Servers

Several power management techniques have been developed for portable and desktop computers. One general approach is to reduce the power consumed by components not currently in use. Examples of this kind of technique include placing the CPU in a "halted" state when there are no "runnable" tasks, and turning off the hard drive motor after some period of inactivity. Another category of techniques reduces power consumption of components that are in use, allowing responsiveness or quality to degrade within some acceptable range. Examples here include scaling the CPU frequency and reducing the display brightness.

In contrast, server-class systems currently do very little in the way of power management. In part, this is because most server-class systems do not incorporate the mechanisms required for many power management policies. We believe this is mainly due to the focus on performance as the only metric of relevance for server systems.

We believe that there is ample opportunity to reduce the power consumption of web server systems. As illustrated in the previous section, web server systems are typically designed to handle a peak workload that is much higher than their average workload. As a result, the system will have significant periods of low utilization. During low utilization, some components may be completely idle, and thus could be "spun down", and other components could be operated at reduced power. In web servers, such techniques would most likely be targeted at the CPU, disk, or memory. During periods of peak usage, a web server could trade off quality or responsiveness in order to reduce its power consumption. For example, the server could serve lower resolution images or throttle requests for dynamic content during periods of high activity.

To demonstrate the potential benefits of power management mechanisms, we use simulation studies to evaluate the benefits of scaling the CPU voltage and frequency in response to the server workload. Our results indicate that voltage scaling can result in significant energy savings, without substantially altering the responsiveness of the system.

3. Methodology

We begin our study by characterizing the energy consumption of a web server by measuring the energy used by its components as it executes web server workloads. This section describes the environment for collecting data and the web server workloads.

3.1 Environment

Our instrumented "white box" web server system is made up of the following components:

- Pentium III 2.0 volt 600MHz processor with 512KB off-chip L2 cache and a 100MHz memory bus in a Slot 1 package.
- Tyan S1854 motherboard with 512MB IBM 3.3volt SDRAM (2 DIMMs) – 100MHz, 32Mx72.
- IBM 60GB Deskstar ATA-100, 7200 rpm, IDE disk drive
- Deer Computer Corp DR-250 250 watt ATX power supply with a combined peak output rating of 125 watts.
- Matrox 64/MILA/16/1B2 graphics card.
- Realtek RTL-8029 10/100Mbit Ethernet card.
- Software: Red Hat Linux Release 7.0 with 2.4.3 kernel (no parameters changed) and Apache 1.3.12 web server

The default Apache configuration assumes a modest size web server. Since our goal was to test the limits of the server, we changed the `MaxClients` setting from 100 to 1,500 and increased `MaxRequestsPerChild` from 100 to 10,000. Except for these changes, all other setting were left as in the default case. We sent requests to the server from a client that is more powerful than the server, thereby ensuring that server performance was not artificially limited by client resources. All machines were on a single subnet connected through an ExtremeNetworks Fast Ethernet (100 Mbits/sec) switch.

3.2 Measurement System

We instrumented five power supply leads:

- +3.3V supply to the motherboard, powering the 512MB of RAM, video card, and Ethernet card
- +5V supply to the motherboard, powering the Pentium III processor
- +12V supply to the motherboard, powering the processor cooling fan
- +5V supply to the single disk drive, powering the controller and other components
- +12V supply to the single disk drive, powering the motor and head actuator mechanisms

We determined the energy consumed by each sub-system by measuring the drawn voltage and current. While we could directly measure voltages, we used a sense resistor in series with each sub-system to measure the current. Signals from the sense resistors were filtered with a 10kHz low pass filter, gained to be within $\pm$10V using custom circuitry, and then passed to a PCI-6071E A-to-D board from National Instruments. Each channel was sampled at 1,000 times per second, and custom software was written to gather and process the data. The accuracy of the measurement system is within 5%, with sense resistor accuracy and amplifier voltage offset being the dominant source of errors.

3.3 Workloads

Since our goal is to study the power consumption of web servers under real workloads, we constructed workloads using web server logs obtained from several production internet servers. Each access log represents 24 hours of activity. The first workload is derived from the web server logs of the 1998 Winter Olympics web site. These logs contain all of the requests processed over the course of one day at one of the four geographically distributed sites used for the Olympics. The second workload is derived from the server access log of one day's activity on the web site of a financial services company. The third workload is derived from a log of a proxy server operated by the Information Resource Caching (IRCache) Project [16], which is affiliated with the National Laboratory for Applied Network Research (NLANR). The IRCache Project operates a number of caching proxy servers on the Internet and publishes the logs from these servers as a way to promote research into web caching. All IRCache servers run Squid, which is also supported by the IRCache project. Strictly speaking, a proxy server is not a web server in that it does not have its own content, but stores frequently referenced content of other servers for more rapid access by clients.

For each workload, we construct a stream of HTTP requests for static files based on the server access log. We exclude all requests that provide parameters as part of the URL, since these are obviously for dynamic content. Such requests account for 2.5% or less of the Olympics98 and Finance logs, and none for the Proxy server log. We treat all remaining requests to be for static files. However, to account for any remaining URLs that may generate dynamic content, all file names are augmented with the size of the file returned, so that each dynamic response (of a unique size) is treated as a separate file. We have used this approach to deal with dynamic content because we do not have access to the dynamic scripts used at any of these sites.

Persistent connections, a feature of the HTTP/1.1 protocol [26], allow a client to make multiple requests to the web server over a single TCP connection, thus amortizing the cost of connection establishment and teardown. While

the access logs do indicate whether the client used the HTTP/1.0 or HTTP/1.1 protocol, there is no explicit indication of which requests were grouped onto a single connection. To approximate the effect of persistent connections, we use the following rule to group requests into connections: any request received from the same client within 15 seconds of a prior request is grouped into a connection with that prior request. Since the timestamp in the access log has only a one second resolution, we randomly distribute the arrival time of all HTTP/1.0 requests and new HTTP/1.1 connections within that second. If only one follow-up request in a persistent connection is logged in a second, we assume that it arrived at a random time within that second. We assume that multiple follow-up requests within the same second arrived at the server as a burst at a random time within that second.

It is worth noting that we have used the timestamp in the web server access log to indicate the time the request arrived at the server. In general, this is not accurate, since the timestamp in the access log generally indicates the time that the response was sent, not the time the request was received. For servers that are significantly overloaded, or for very large files, the time of the response might not be a good indicator of the time of the request. We believe neither of these factors were present to a significant degree on the Olympics98 and Finance servers on the days the logs were collected, and therefore we believe the timestamp in the access log is a good indicator of request time for our workloads. For the Proxy workload, we can determine the request time exactly since Squid logs the time required to process the request in addition to the time of the response.

Workload	Olympics98	Finance	Proxy
Avg requests / sec	97	16	15
Peak requests / sec	171	46	30
Avg requests / conn	12	8.5	31
Files	61,807	16,872	698,232
Total file size	705 MB	171 MB	6,205 MB
Requests	8,370,093	1,360,886	1,290,196
Total response size	49,871 MB	2,811 MB	10,172 MB
97%/98%/99% (MB)	24.8 / 50.9 / 141	3.74 / 6.46 / 13.9	2,498 / 2,860 / 3,382

Table 1.2. Characteristics of three web server workloads. Average requests per second is the average request rate over the entire 24 hour period, and peak requests per second is the highest observed rate for a one minute period. The average requests per connection is based on our technique of grouping requests into connections. Files is the number of unique files requested, and Total File Size is the total size of these unique files. Requests is the number of distinct HTTP requests, and Total Response Size is the total size of response data sent for these requests (excluding HTTP headers). 97%/98%/99% data is the amount of memory needed to hold the unique data for 97%/98%/99% of all requests.

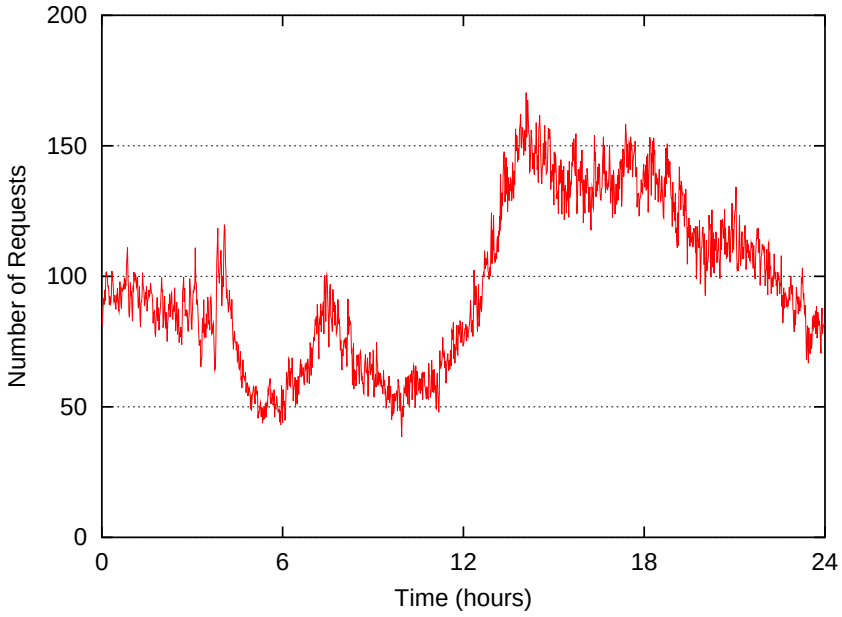

Figure 1.1. Request rate seen at one of the Olympics web sites on February 19, 1998

The characteristics of our three workloads are summarized in Table 1.2. Particularly important for our study is the 97%/98%/99% data size figures, which indicate the amount of memory needed to hold the unique data for 97%/98%/99% of all requests. For example, in the Olympics98 workload, 99% of requests could be served from memory using a cache of only 141 MB. Since our web server has 512MB of memory, caching files in memory could significantly reduce the disk activity required to serve the workload. We should note that it is difficult to make a direct correlation between the 97%/98%/99% data size figures and the cache space required by the web server, since these figures were determined with complete knowledge of the request stream. On the other hand, the figures are for a "static" cache, and could be reduced even further if cache contents could be dynamically managed. On balance, these figures should simply be viewed as indicators of what could be achieved with a reasonably effective caching strategy.

Since caching should significantly reduce disk activity for both the Olympics98 and Finance workloads, we wanted to balance them with a workload that has larger amounts of disk I/O. The Proxy workload has this feature. The total response size of Proxy workload, at 10.2 GB, is less than twice as large the total file size, indicating that first-reference requests alone will generate a significant level of disk activity.

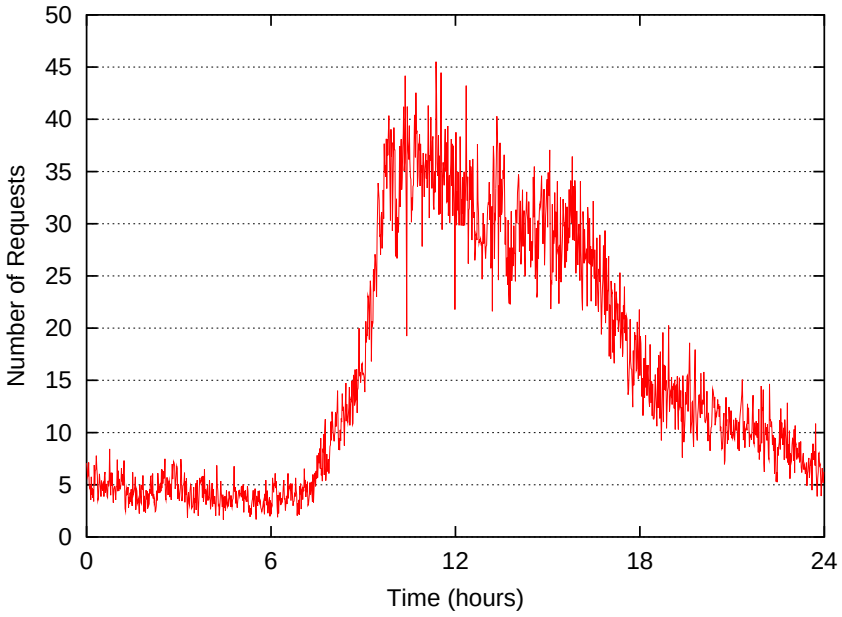

Figure 1.2. Request rate seen at a Financial services company web site on Oct. 19, 1999

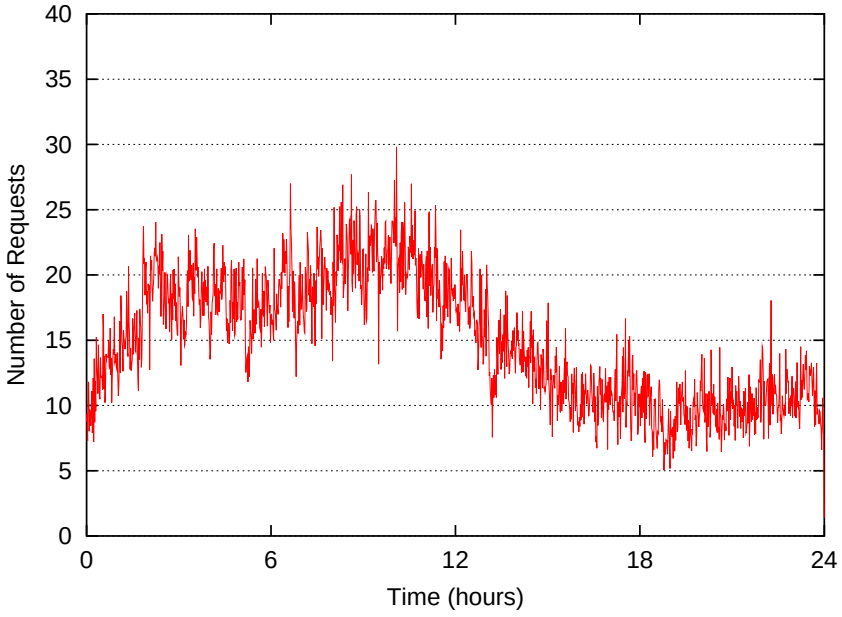

Figure 1.3. Request rate seen at IRCache Project Silicon Valley proxy server on May 2, 2001

Web serving workloads are notorious for having high variability [4] and these three workloads are true to form. As indicated in Table 1.2, the peak request rate measured in one minute intervals is anywhere from 1.5 to 3 times larger than the average request rate. [1] Each workload shows its own distinct pattern of request arrivals, which are illustrated in Figures 1.1, 1.2, and 1.3. This characteristic reflects the challenges in configuring web server systems that have sufficient capacity to handle peak load and are power-efficient when serving the average load.

3.4 Replay Program

We use a modified version of the httperf workload generator [20] to generate an HTTP load on the web server. httperf can simulate a large number of http clients from a single machine by using an event-driven design that minimizes the resources required to maintain each connection. The process described in the previous section converts an access log into a trace file that contains a sequence of connections, each consisting of a sequence of HTTP requests. The trace file specifies the time interval between each connection initiation and the time interval between the initiation of each HTTP request within a connection. We modified httperf to effectively replay a server access log by initiating connections at the appropriate times, generating requests for specific URLs at specified time intervals, and closing the connections after all requests were complete.

In addition, we added the capability to scale the inter-arrival time of connections (but not requests within a connection) by a user–specified amount. A scalefactor of "2×" corresponds to reducing the inter-arrival time of connections by 50%. Since each connection roughly corresponds to a client, reducing the connection inter-arrival time effectively generates a heavier client load, but with the same basic pattern of connection arrivals. Inter-arrival time of requests within a connection are not scaled since these essentially represent user think time or network and client overhead involved in retrieving multiple components of a web page. We use this scaling mechanism to evaluate server performance for a range of client load intensities. Thus we are able to scale the intensity of any workload to "1×", "2×", "2.5×", etc.

4. Power Consumption in Web Servers

In this Section, we characterize the power consumption of a web server using the three workloads described in Section 3.3.

[1] The request rate data for Olympics98 does not match that given in Table 1.1 because the latter is computed over all 16 days of the Winter Olympics.

4.1 Overview of System Power Consumption

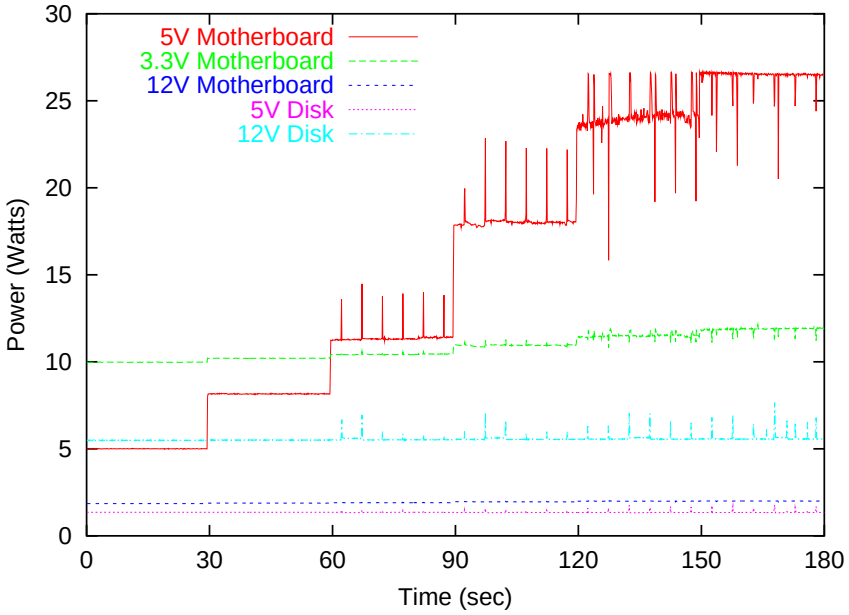

Figure 1.4. Web server power consumption for a steady request stream. The graph shows the power consumption measured on each power supply line over time. Notice that the 5V motherboard power changes significantly as the request rate to the web server increases. The other power supply lines are mostly constant.

Our first measurement is performed using a steady request stream for a single URL at various request rates. The purpose of this measurement is to study the relationship between the load placed on the server, as measured by request rate, and power consumed by the various components of the web server. While a workload of a single URL is clearly not realistic, it nevertheless provides valuable insights into the effect of load on web server power consumption. Figure 1.4 displays the power consumed by our 600MHz web server system over time for the "idle" case and request rates of 100, 200, 400, 600, and 800 req/sec, each executed for 30 seconds. The graph shows the average power consumed from each of the five supply lines for each 100 millisecond interval.

The 5V motherboard supply, which provides power to the 600MHz Pentium III processor as well as other components on the motherboard, changes significantly during the course of the workload. When the system is not servicing requests, the processor spends nearly all the time in a halted state, consuming approximately 5.0 Watts. At 800 requests/sec, the CPU appears saturated, consuming approximately 26.5 Watts. By increasing the sampling rate of our

Request Rate (req/sec)	Idle	100	200	400	600	800
CPU Energy (Joules)/sec	5.0	8.3	11.6	18.2	24.3	26.5
Response time (ms)	-	1.0	1.1	1.2	4.8	908

Table 1.3. CPU Energy consumed (per second) for serving requests of different sizes.

energy measurement equipment, we determined that the CPU typically operated at one of two extremes. At one extreme, the system was idle (waiting for client requests), and the processor was in the halt state consuming only about 5 Watts. At the other extreme, the CPU was active and thus consuming nearly its peak power of 26.9 Watts. The result of this bimodal operation is that CPU energy is simply proportional to the number of cycles spent processing requests. For our simple workload that serves all requests from memory, the number of cycles spent processing requests is essentially linear in the request rate for light to moderate load, and thus energy consumed is essentially a linear function of request rate up to around 600 requests/second. We also measured average response time at each request rate, and have summarized this data along with the average CPU energy consumption in Table 1.3. The two orders of magnitude increase in response time from 600 to 800 reqs/sec confirms that the system is saturated at this level, with the CPU the apparent bottleneck resource.

The CPU and 12V disk power show pronounced spikes every five seconds, which are due to the periodic activity required to flush the web server log to disk. The 5V disk and 12V motherboard energies have almost no variation over the course of the run and are also quite small in comparison to the other three components. The 12V disk power also shows relatively little variation outside of the periodic spikes already mentioned. We note a trend in the 3.3V motherboard supply, where it remains perfectly constant at about 10 Watts while idle, but increases slightly as the load increases. We attribute this to the increased use of memory and to the network interface. Through a separate measurement, we ascertained that the cooling fan draws a constant 1.5 watts from the 12V supply, and the video card, a nearly constant 6.09 Watts from the 3.3 V motherboard supply.

Next we measured the power consumed by our web server for the Olympics98 workload when run at a scalefactor of 4.0, which results in a request rate that stresses the server but does not overload it. These results are shown in Figure 1.5. The figure shows the average energy consumed (in Joules) on each of the five instrumented power leads, averaged over intervals of 60 seconds. The scale for power is the same as in Figure 1.4. As in the single request workload above, the largest and most variable component of power consumption in this workload is CPU power. This result is not surprising, since a high percentage of requests for the Olympics98 workload can be served from the RAM cache. The total

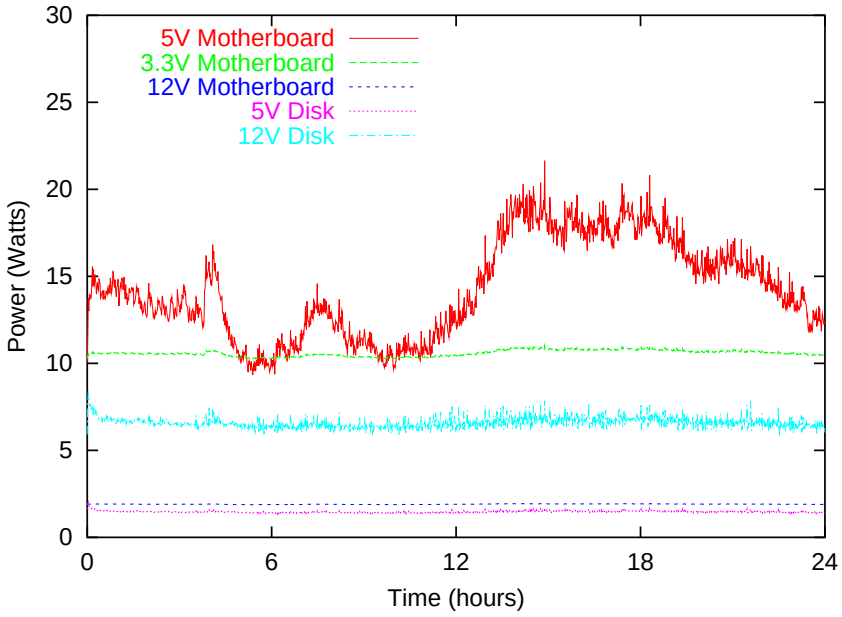

Figure 1.5. Power consumption of the Olympics98 workload (at 6X).

system power consumption of this workload is mostly between the 200 req/sec and 400 req/sec workloads in Figure 1.4. We also measured the response time of each HTTP request over the course of the workload, and found that the 95th percentile of the response time was less than 25 milliseconds, indicating that the server is not overloaded for any significant portion of the run. The same workload executed at a scalefactor of 6.0 resulted in 95th percentile response time of over 130 milliseconds, which indicates significant periods of overload.

We scaled the Finance and Proxy workloads to insure that the server was moderately heavily loaded, in a manner similar to that of the Olympics98 workload. Figure 1.6 shows the power measurements we obtained using the Finance workload at a scalefactor of 20.0. We also measured the power consumption of our web server for the Proxy workload at a scalefactor of 2.0. These results are shown in Figure 1.7.

The Proxy workload differs from the Olympics98 and Finance workloads in that a large fraction of requests result in actual disk I/O. This difference is clearly visible in the power measurements, where the CPU consumes much less power with little variation. On the other hand, the 12V Disk supply (disk mechanics) shows significantly higher and more variable power consumption. Total CPU energy consumption is also significantly smaller than for the Olympics98 workload, with the 3.3V motherboard (memory) being the most dominant sin-

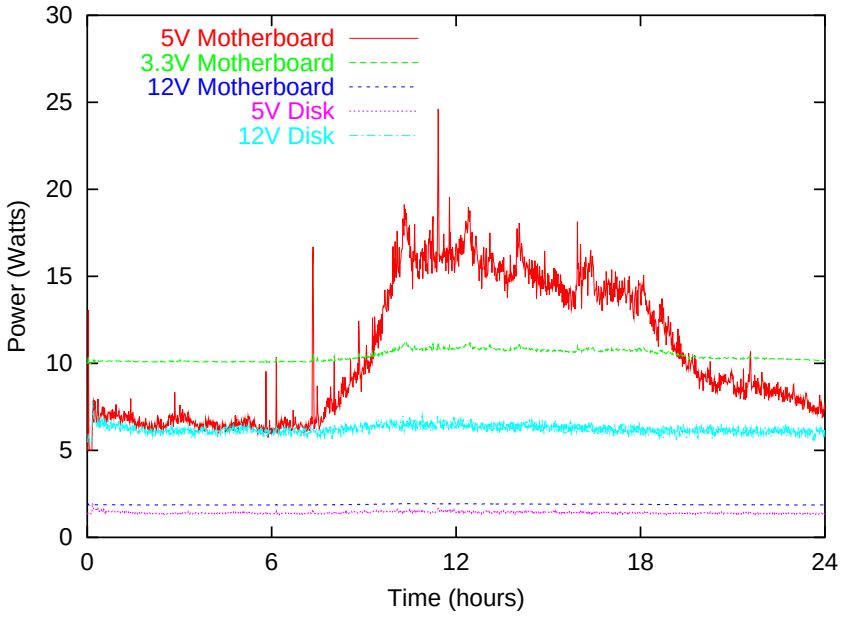

Figure 1.6. Power consumption of the Finance workload (at 20×).

	Olympics98	Finance	Proxy
5V Motherboard	1,232,710	711,415	627,977
3.3V Motherboard	914,116	882,117	869,015
12V Motherboard	164,795	161,997	160,732
5V Disk	127,007	119,610	157,679
12V Disk	569,395	522,065	724,130

Table 1.4. Total Energy consumed (in Joules) of each of the five power supply leads for each of our three workloads.

gle component. Table 1.4 presents the total energy consumption for each of the workloads over the course of a 24 hour run.

4.2 Opportunities for Power Management

Our web server system is made up of several components that consume power. As we observe in the previous section, the CPU (5V Motherboard) is the dominant consumer of power when executing the Olympics98 and Finance workloads. The CPU power consumption with these workloads also exhibits a large variation, presenting an excellent opportunity for power management.

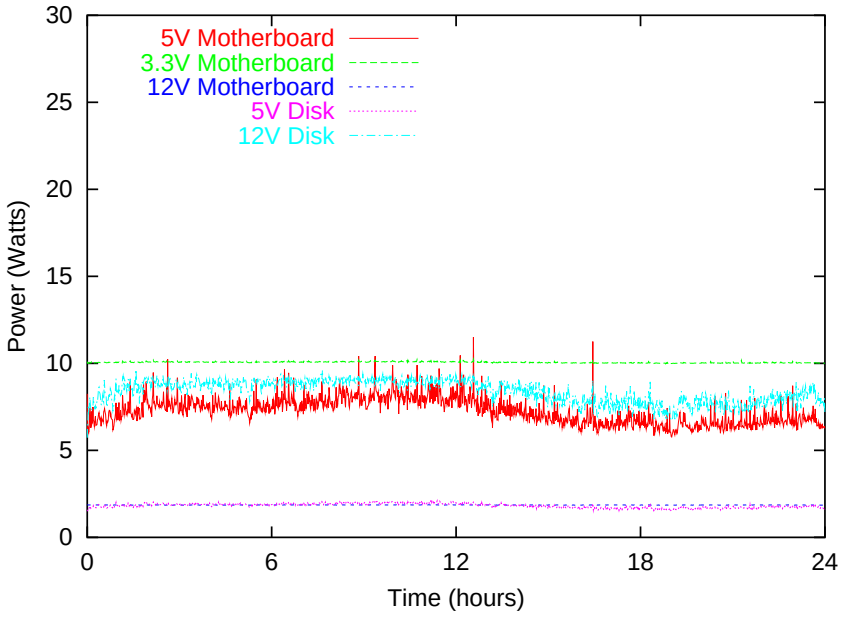

Figure 1.7. Power consumption of the Proxy workload (at 2×).

However, for the Proxy workload, it is the 12V disk and 3.3V motherboard (memory) that consume more power and show a large variation, raising the question of whether these components should be power managed.

The 3.3V motherboard supply powers the chipset, 512MB of memory, and associated components. To determine if it might be worthwhile to employ some form of power management for memory, we performed additional measurements on the 3.3V Motherboard supply. Two 256MB DIMMs make up the memory in our system. We began with two measurements: first, the power consumed (by the 3.3V Motherboard supply) when the system is idle, and next, the power consumed after removing one 256MB DIMM. The reduction in power was measured to be 0.9 Watts, implying that the 512MB of memory in our system consumes 1.8 Watts when idle. We then executed a microbenchmark on the 512MB system which saturated the memory with read and write requests. The difference between the power consumed under the memory test and the idle 512MB measurement was 2.1 watts. This difference represents an upper bound on additional power used by the 512MB of memory when active (that is, we have attributed all the additional power to the memory, even though some might actually be consumed by other components powered by the 3.3V motherboard supply). Thus, the total power consumed by our 512MB of memory (idle + active) is no more than 3.9 watts. Even if a large fraction of this power

could be eliminated without increasing power use in other components, the savings would be small in comparison to the potential savings from CPU power management.

Comparing the power consumed by the 12V disk supply when the disk is spinning idly, with the power consumed when it is busy results in a similar conclusion. The power consumption of the 12V disk supply is 5.6 Watts when idle, rising to 9.9 Watts when serving a random stream of reads. As in the case of the memory subsystem, even if active power management were to reduce this power the maximum possible reduction would only be 4.3 Watts. Although power management based on inactivity may be possible, we expect its effectiveness to be markedly less than that in handheld systems because of the continuous nature of the web serving workload and the steady stream of disk writes (server logs). More complicated schemes that modify the disk's workload, perhaps by delaying or reordering activities, may be possible but are beyond the scope of this paper.

In summary, for web server systems that are built from industry standard, "white box" servers, the greatest opportunity for energy savings will be from CPU power management. Power management of the remaining components are unlikely to provide significant energy savings. We explore one approach for managing CPU power consumption in web servers in Section 6.

5. A Power Simulator for Web Server Workloads

Most web server systems today are not configured for power management, precluding direct measurement of any power management policies on a real system. We therefore constructed a simulator in order to evaluate the potential benefits of power management in this environment. Using this simulator and our three web server workloads, we evaluate dynamic voltage and frequency scaling, which has shown significant benefits in battery powered systems.

Our simulator is based on a queuing model for a server, and uses the CSIM execution engine [15]. In addition to simulating the residence time (and thus the response time) for each request, we also simulate the energy expenditure of the CPU during web serving. The input to the simulator is a stream of time–stamped requests in essentially the same format as used by the httperf tool. Each request is characterized by its arrival time, and its cost as measured by the size of the response. The simulator determines the CPU time and energy consumed to service a request from a model based on measurements of the actual energy consumed by our 600MHz web server system.

To construct the CPU time and energy model, we measured the energy consumed by the CPU when servicing requests with a variety of response sizes. The calibration was accomplished by injecting a stream of requests with a fixed response size at a known rate, and measuring the energy consumed from the

Resp. size (bytes)	100	1000	5000	10,000	50,000	100,000	500,000
$E_{service}$ (Joules)	0.0171	0.0177	.0206	0.0294	0.112	0.277	0.761
CPU cycles (est.)	469K	485K	563K	803K	3.05M	7.57M	20.82M

Part (a). Responses Served from Memory.

Resp. size (bytes)	100	1000	5000	10,000	50,000	100,000	500,000
$E_{service}$ (Joules)	0.0484	0.0865	0.139	0.190	0.382	0.589	2.426
CPU cycles (est.)	1.32M	2.37M	3.81M	5.21M	10.45M	16.11M	66.38M

Part (b). Responses Served from Disk.

Table 1.5. Energy consumed when serving requests of different sizes on the 600MHz system

5V Motherboard power supply over a fixed time interval. From this value, we subtract the CPU idle energy, which is the energy that would have been consumed by the CPU had it been idle for the duration of the interval. The result is the additional CPU energy required to serve requests over the interval. Dividing this by the number of requests served during the interval yields the energy consumed per request, $E_{service}$. We calibrated two cases: one where all the files were served from memory, and one where the files were served entirely from disk. In both cases, we measured the energy expended by the CPU (5V motherboard) while serving the requests. When serving files from disk, the CPU expends more energy because it needs to dispatch requests to the disk and service the ensuing interrupts. The $E_{service}$ values for a variety of response sizes are shown in Table 1.5.

Next, we use $E_{service}$ to calculate the number of CPU cycles it takes to serve a response. We measured the power consumption of the CPU to be $P_{min} = 4.97W$ when idle (i.e., halted) and $P_{max} = 26.9W$ when the CPU is fully busy with no halted cycles. When fully busy, we know the CPU executes F cycles (600 million cycles per second in our case) and consumes 26.9 Joules ($P_{max} \times 1$ second) when doing so. When halted for one second, we know the CPU executes 0 cycles and consumes 4.97 Joules ($P_{min} \times 1$ second). The CPU cycles corresponding to an energy usage of x joules can thus be computed using the following formula, which is used to compute the CPU cycle estimates in Table 1.5.

$$Cycles = x \times \frac{F}{P_{max} - P_{min}}$$

Our simulator combines the model for requests served from memory and the model for requests served from disk by simulating an LRU cache of files from the disk. We use a cache size of 472MB, which leaves 40MB to be used by the operating system and Apache webserver. Each file in the cache consumes an integral number of 4K byte pages. When processing a request, if the target

file is in the RAM file cache, the simulator uses the energy model for requests served from memory. Otherwise, it uses the energy model for requests served from disk. The simulator is extremely fast, allowing us to simulate over 75,000 requests/second on a 866MHz system with a memory footprint of less than 10MBytes.

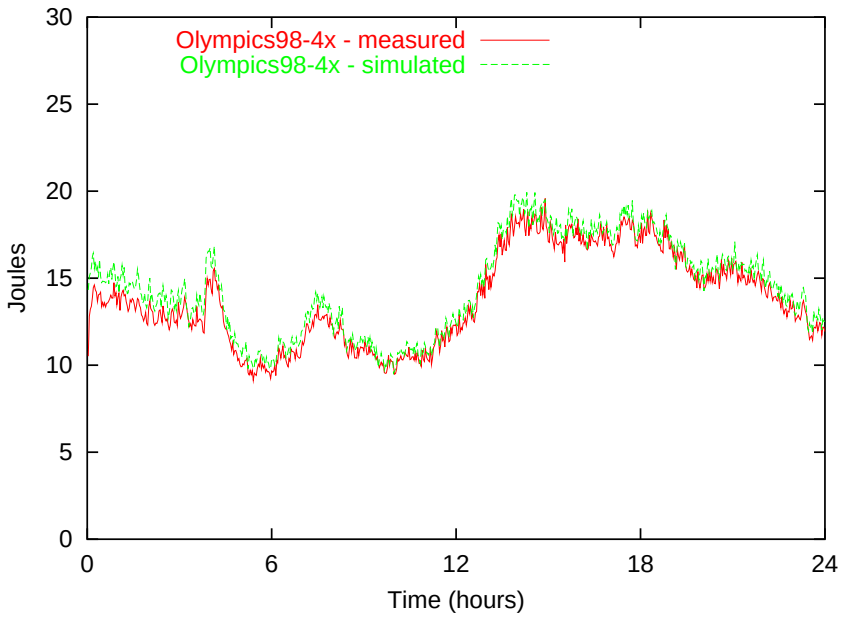

Figure 1.8. Measured vs Simulated Energy Consumption for Olympics98-4× workload.

Workload	Olympics98-4×	Finance-20×	Proxy-2×
Measured CPU Energy (Joules)	1,232,710	925,540	627,977
Simulator CPU Energy (Joules)	1,253,652	946,440	663,648
Error in Total Energy	1.70%	2.26%	5.68%
Correlation Coefficient	0.9846	0.9716	0.8485

Table 1.6. Comparison of Measured to Simulated CPU energy for three workloads. Correlation coefficients were computed based on the energy used in 30 second intervals over the length of the run.

After calibration, we simulated the CPU energy consumption of the Olympics98 workload at a scalefactor of 4×. Figure 1.8 shows the measured CPU energy consumed by the 5V motherboard in the 600MHz system during the execution, overlaid with the simulator output. The simulator over–predicts energy consumption by 1.7%. Furthermore, the correlation coefficient between the

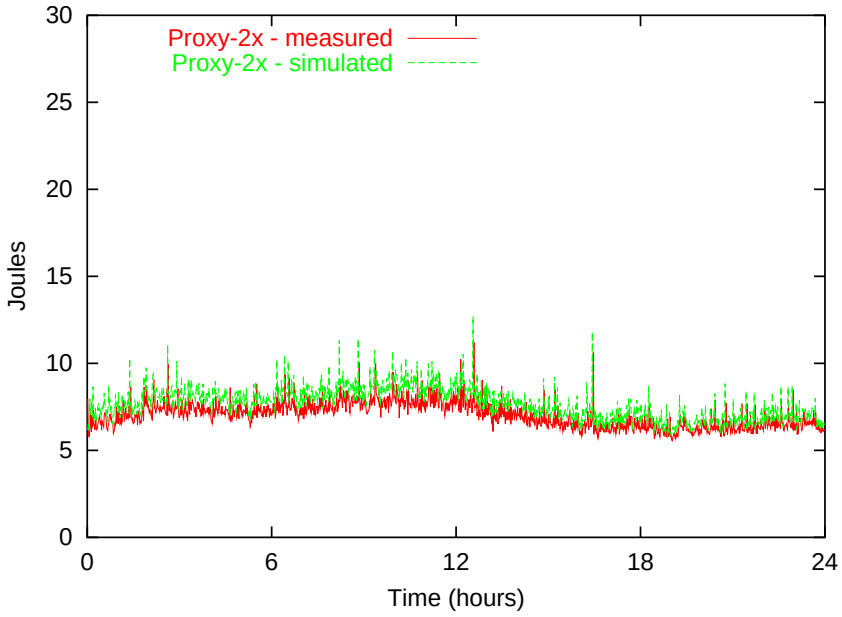

Figure 1.9. Measured vs Simulated Energy Consumption for Proxy-2× workload.

measured and simulated power is 0.9846. We also simulated the Finance workload at a scalefactor of 20× and the Proxy workload at a scalefactor of 2×. Simulated and measured data for the Proxy workload is graphically shown in Figure 1.9. All the simulation results are summarized in Table 1.6. In the interests of brevity we have omitted the graphs of the results for the Finance workload.

6. Dynamic Voltage and Frequency Scaling

One way to reduce the power consumed by the CPU is to lower its operating voltage. A reduction in operating voltage generally also requires a proportional reduction in frequency [22]. This approach of varying the processor voltage in proportion with its frequency is known as voltage scaling. Voltage scaling is advantageous because the energy consumed by a processor is directly proportional to V^2, where V is the operating voltage. By varying the processor voltage and frequency, it is possible to obtain a quadratic reduction in power consumption.

We modified our simulator to support two additional models for CPU power consumption based on data from actual processors that support dynamic voltage scaling. One model is based on publicly available data for the Transmeta

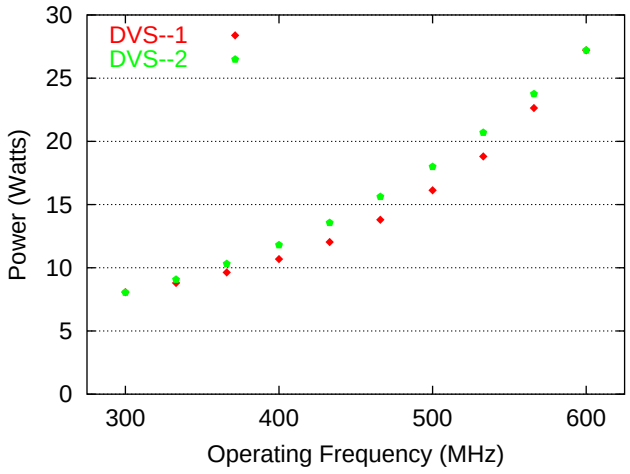

Figure 1.10. Power consumption of two simulated voltage-scaled Pentium 600MHz III processors

Time quantum	20 ms
Frequency Range	300MHz - 600MHz, in steps of 33MHz
Core Operating Voltage Range	1.5V to 2V, linearly with frequency
Busy threshold	Min = 0.80, Max = 0.95

Table 1.7. Parameters of the hypothetical voltage-scaled Pentium III processor used in the simulation

TM5400 processor [9], and the other is based on data for an alternate processor [21]. We scaled the data from both processors to fit the 600MHz Pentium III's maximum frequency and core operating voltage. Our system parameters are shown in Table 1.7. Figure 1.10 shows the power consumption of each simulated processor at the different operating points. We assume that the time and energy required to change frequency and voltage is negligible. In the rest of this paper, we refer to the Transmeta based model as DVS–1 (for Dynamic Voltage Scaling 1), and the other as DVS–2.

The simulator uses a very simple policy based on the recent CPU utilization to determine the processor frequency and voltage. At the beginning of each time *quantum*, we examine the system load during the previous time quantum. If the system utilization is between the threshold values, we do nothing. If the high threshold is exceeded, we step up the frequency of the processor by 33MHz and the voltage appropriately. If the low threshold is exceeded, we step down the frequency and voltage. This type of dynamic voltage scaling algorithm was originally proposed and studied for desktop application workloads in [27].

The minimum and maximum attainable frequencies are 300MHz and 600MHz respectively.

We assume that the number of processor cycles needed to service a request is independent of the processor frequency and is always equal to the number of 600Mhz cycles needed to service the request. This is a conservative assumption since it does not consider energy savings due to a reduction in the CPU stalls that could occur as a result of other system components (particularly memory) appearing relatively faster to the processor as it is slowed down. The CPU time required to service a request is then calculated as the number of cycles to be executed multiplied by the current operating frequency of the processor. The model properly accounts for requests whose service time is spread across several time quantums with different operating frequencies.

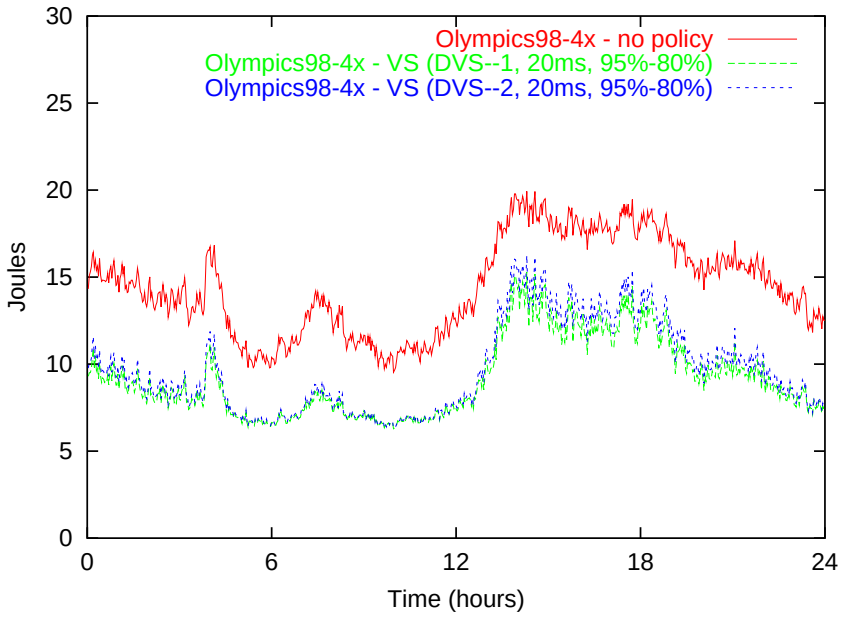

Figure 1.11. Energy savings from voltage and frequency scaling for the Olympics98 workload

Figure 1.11 shows the results of our simulation with the Olympics98 web server workload. The top curve shows the simulated 5V energy consumption of this workload on the 600MHz system without voltage scaling. The two lower curves show the simulated 5V energy consumption of the DVS–1 and DVS–2 voltage scaled Pentium III processors.

Table 1.8 lists the energy savings we obtained for the three workloads. There is a noticeable difference in the energy savings between the two processor designs. This is because the power vs. frequency curve for the DVS–1 design is

Workload		Olympics98-4×	Finance-20×	Proxy-2×
Base energy (J)		1,253,652	946,440	663,348
DVS–1 design	Energy (J)	798,684	636,681	510,788
	Savings	36.3%	32.7%	23.0%
DVS–2 design	Energy (J)	838,436	655,426	512,710
	Savings	33.1%	30.7%	22.7%

Table 1.8. Energy Savings from Dynamic Voltage Scaling

more convex than that for the DVS–2 design. Voltage scaling provides the most energy savings for Olympics98-4× and Finance-20× because those workloads exercise the CPU more than the Proxy workload.

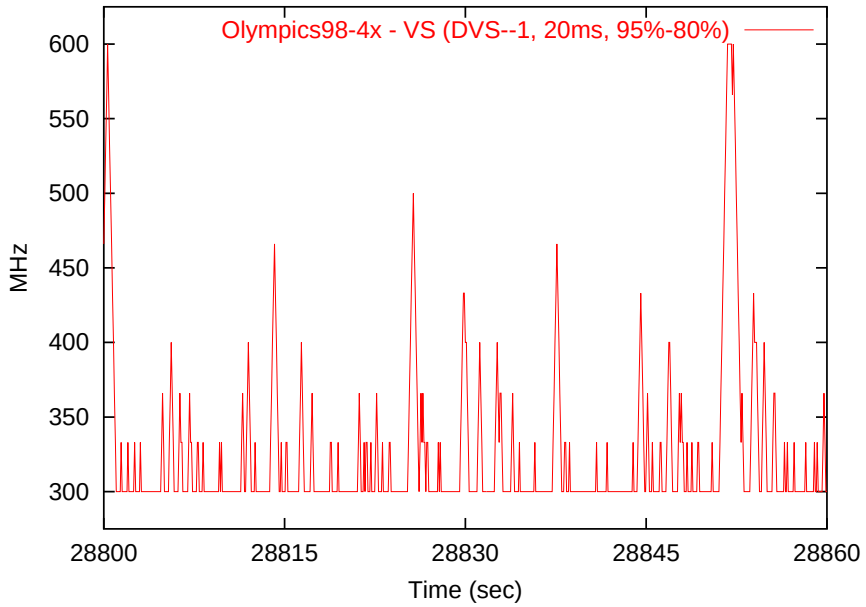

Figure 1.12. Operating frequency of the simulated voltage scaled processor as it executes 60 seconds of the Olympics98 workload. Each data point represents the processor frequency during a 20ms interval.

Over the course of the execution, the average operating frequency of the voltage scaled processor correlates to the incoming request rate, which is a good indicator of the system load for these workloads. In the interests of brevity, we have omitted the graph depicting this behavior. Instead, Figure 1.12 shows the frequency of the DVS–1 voltage scaled processor over the course

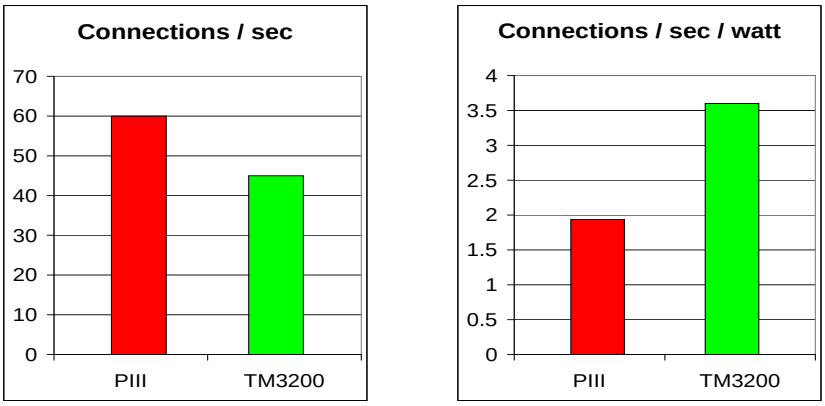

Figure 1.13. Comparison of Two Servers using a Traditional Web Server Performance Metric and our Proposed Energy/Performance Metric

of 60 seconds starting at 8AM as it executes the Olympics98 workload. The frequency varies all the way between 300MHz to 600MHz, demonstrating the processor changing its frequency as it adapts to the bursty workload. The DVS–2 processor exhibits similar behavior.

7. Implications for Web Server Performance Metrics

The benefits of reduced energy consumption must be balanced against the performance impact of the power management mechanisms employed. To allow this trade off to be analyzed in a systematic manner, we need to take a fresh look at the performance metrics that are in use to characterize traditional servers. These metrics focus on absolute performance as the only relevant differentiator, and they ignore issues such as power consumption and cooling requirements. A more relevant evaluation metric for the emerging new applications would be performance relative to power consumption. To illustrate, we have conducted a comparison between two machines running the Apache Web server with the Linux operating system. The first machine is a traditional server, using a 750MHz Intel Pentium^{TM}III processor with a 16 Kbyte L1 I-cache, a 16 Kbyte L1 D-cache, a 1 Mbyte unified L2 cache, and 256MB of main memory. The second machine is power-managed server, which uses a Transmeta Crusoe^{TM}TM3200 processor with a 24 Kbyte L1 cache and 256MB of memory. The TM32000 processor is rated at 400MHz, but because of the code-morphing technology it uses, it is difficult to make comparisons based on operating frequency. The remaining components of the two systems (e.g., disk,

network) are identical or comparable. (The machines used in these measurements differ from those used in the previous sections because this work was done separately, using other machines that were available to us. Our goal in this section is simply to motivate the need for new metrics of web server performance, and not to draw comparisons with the results of previous sections.) Figure 1.13 shows the results of the comparison.

Looking at absolute performance, the traditional server outperforms the power-managed server by a ratio that is roughly proportional to their relative speeds. However, when one considers the performance per unit of power consumption, the power-managed processor gives 3.6 connections/sec/Watt, substantially better than the 2.0 connections/sec/Watt of the traditional server. Since most web applications are inherently highly parallel, and web sites are commonly implemented using clusters of web servers, simply switching to a power-managed system as the building block for these clusters could result in substantial energy savings. This reduced power consumption will also help realize further benefits in reduced cooling, and could allow more efficient packaging of server components, thus reducing the required volume of raised floor space.

8. Related Work

Both the distribution of power consumption and methods for managing it have been studied extensively in the area of portable, battery-powered computers such as laptops and personal digital assistants (PDAs). For example, one power consumption study is a detailed analysis of the energy consumed by the various components of Apple Macintosh laptop computers [19]. While similar in approach to our study, their work focuses on portable computers and workloads typical for such machines, whereas our study focuses on web servers and their typical workloads. This distinction is significant for two reasons. First, the basic set of components is different between these two platforms, and components that are present in both are frequently designed to very different specifications. Second, the workloads and the expectations about their behavior are radically different.

There have been a number of studies of specific power management mechanisms and policies, and a set of standards have been developed for the mechanisms, specifying the interfaces between power-management software and the hardware. Examples of such architectures include the industry standard Advanced Configuration and Power Interface or ACPI [6] and Microsoft's OnNow initiative [2]. Many of these mechanisms could be directly applied to a server system in a web-serving environment although there is no guarantee that the management policies designed for using them in portable, battery powered systems are equally applicable.

Among the common power management techniques is spinning down a hard disk after some period of inactivity [8]. Microprocessors have also received a considerable amount of attention, and many microprocessor architectures and microarchitectures incorporate power-saving features: examples include the mobile processors available from Intel with its SpeedStep(TM) technology and the Transmeta Crusoe processor with LongRun [9]. More recently developed and less widely deployed today are new memory chip architectures that are incorporating similar "spin down" states so that the system can actively manage the power used by main memory [3]. In addition, a number of current research efforts are focusing on new power management mechanisms employed at the operating system [25] and application layers [10] of the system. Techniques for dynamically controlling processor temperature [23] can also be applied to web servers. This results in power savings because CPU activity is decreased to lower processor temperature. Our work complements these studies by demonstrating the value of these performance management mechanisms in a web-serving environment. Furthermore, by measuring the power usage of the specific components of the machine, our work gives a valuable insight into which techniques are likely to provide the greatest benefits.

Our approach of reducing the power consumed by the server by reducing the clock frequency of the processor has been proposed and studied in a variety of contexts [27], but to our knowledge our study is the first to show the benefits of this technique in a web server environment. A variety of approaches have been proposed for determining when to change clock frequency and how to select a new frequency, and a good comparison of these policies is provided in [13]. The policy we use is based on the same basic principle as Transmeta(TM)'s LongRun(TM) power management technique [9].

Studies of web server performance commonly use synthetically generated workloads and workloads based on the web logs of actual servers. Our use of web logs to generate a web server workload differs from many prior studies in that we attempt to recreate the timing of requests exactly as they occurred on the original server. Typically, requests from the log are issued as quickly as possible to determine the maximum throughput of the server. However, there are exceptions to this practice such as the DBench tool [17], which was used in the measurements made of the Harvard Array of Clustered Computers (HACC) [7] locality-based request routing system.

Our proposed SPECWeb/Watt metric was inspired by several other efforts at extended performance metrics to include some measure of energy efficiency. Most notably, [12] considered a number of CPU-centric measures such as SPECint/W and an energy-delay metric, $SPEC^2$/W. For web servers, the SPECweb99 benchmark already includes a notion of delay in that it specifies a maximum response time for all requests in order for the connection to be considered "con-

forming", and thus our SPECWeb/Watt metric captures performance, energy, and delay.

Several researchers have developed tools for simulating the power consumption of computer systems. Brooks et. al. have developed Wattch, a microprocessor power analysis tool based on a microarchitecture simulator [1]. Flinn and Satyanarayanan describe PowerScope, a tool for profiling the energy usage of applications [11]. The power simulator we have developed is substantially faster, because it is targeted specifically for web serving workloads.

9. Conclusions

Over the last several years, the dominant focus of power management research has been on portable and handheld devices. Here, we have presented the case for power management in web serving systems. Typical web servers are designed to operate at a fraction of their peak operating capacity. Such over-engineering is necessary in order to handle the variable and inherently bursty nature of web server workloads. This creates opportunities for energy conservation in environments such as modern Internet data centers where computers are densely packed, and where cooling and electricity delivery are severely constrained.

Using workloads derived from the logs of three production web sites, our direct power measurements show that the CPU consumes the largest fraction of the system power. We use additional measurements to calibrate and validate a power simulator that we created. Our simulator predicts CPU energy consumption for the three workloads with errors ranging from 1.7% to 5.7%. The simulated energy also correlates highly with the measured energy.

We have used the validated simulator to measure the projected effectiveness of a power management policy, typically used only in handheld devices: dynamic voltage and frequency scaling. We find that this technique is effective for saving energy, reducing the CPU energy consumption by up to 36%. Not surprisingly, the energy savings are higher for workloads that exercise the CPU more.

We predict that technology trends will intensify the emerging need for energy-efficient servers, and we suggest that a fundamental change is necessary in the way we design and configure web servers today. Our results provide evidence that in addition to devising server–centric power management strategies, features commonly found in processors intended for mobile computing should be adapted and incorporated as standard features for server processors. The results also suggest that the performance-centric view of designing servers today must give way to a more balanced view in which energy consumption is as important as other goals of the system.

References

[1] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *27th International Symposium on Computer Architecture*, pages 83–94, 2000.

[2] Microsoft Corp. *PC99 System Design Guide*. Microsoft Press, 1999.

[3] Rambus Corporation. Rambus Technology Overview, Feb 1999.

[4] M. Crovella and A. Bestavros. Self-similiarity in World Wide Web Traffic: Evidence and Possible Causes. In *1996 SIGMETRICS Conference on Measurment and Modeling of Computer Systems*, May 1996.

[5] A. Iyengar et. al. High-Performance Web Site Design Techniques. *IEEE Internet Computing*, March/April 2000.

[6] Compaq et al. ACPI Specification, version 2.0, 2000.

[7] X. Zhang et. al. HACC: An Architecture for Cluster-Based Web Servers. In *3rd USENIX Windows NT Symposium*, July 1999.

[8] P. Krishnan F. Douglis and B. Bershad. Adaptive Spin-down Policies for Mobile Computers. In *2nd USENIX Symposium on Mobile and Location-Independent Computing*, April 1995.

[9] M. Fleischmann. Crusoe Power Management: Cutting x86 Operating Power Through LongRun. Embedded Processor Forum, June 2000.

[10] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *17th ACM Symposium on Operating Systems Principles (SOSP'99)*, 1999.

[11] J. Flinn and M. Satyanarayanan. PowerScope: A tool for profiling the energy usage of mobile applications. In *Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 2–10, 1999.

[12] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, September 1996.

[13] K. Govil, E. Chan, and H. Wasserman. Comparing Algorithm for Dynamic Speed-Setting of a Low-Power CPU. In *Mobile Computing and Networking*, 1995.

[14] Akamba Inc. Velobahn product description., 2000.

[15] Mesquite Software Inc. *CSIM18 Simulation Engine*, 1994.

[16] The IRCache Project. http://www.ircache.net/. This project is supported by the National Science Foundation (grants NCR-9616602 and NCR-9521745), and the National Laboratory for Applied Network Research.

[17] A. Wharton J. B. Chen and M. Day. Benchmarking the Next Generation of Internet Servers. Can be obtained by a full text search on 'DBench' on the archives of Iris Today at http://www.notes.net/today.nsf., 1997.

[18] J. R. Lorch and A. J. Smith. Software Strategies for Portable Computer Energy Management. *IEEE Personal Communications Magazine*, June 1998.

[19] J. R. Lorch and A. Jay Smith. Energy Consumption of Apple Macintosh Computers. *IEEE Micro*, 18(6), November/December 1998.

[20] David Mosberger and Tai Jin. httperf: A Tool for Measuring Web Server Performance. In *SIGMETRICS First Workshop on Internet Server Performance*, pages 59—67. ACM, June 1998.

[21] Kevin Nowka. Private communication.

[22] T. Pering, T. Burd, and R. Brodersen. Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System. In *Power Driven Microarchitecture Workshop, attached to ISCA98*, June 1998.

[23] E. Rohou and M. D. Smith. Dynamically Managing Processor Temperature and Power. In *2nd Workshop on Feedback-Directed Optimization*, Nov 1999.

[24] Deo Singh and Vivek Tiwari. Power Challenges in the Internet World. In *Cool Chips Tutorial, held in conjunction with the 32nd Annual International Symposium on Microarchitecture*, November 1999.

[25] A. Vahdat, A. Lebeck, and C. Ellis. Every Joule is Precious: The Case for Revisiting Operating System Design for Energy Efficiency. In *9th ACM SIGOPS European Workshop*, September 2000.

[26] The World Wide Web Consortium (W3C). RFC 2068: Hypertext Transfer Protocol – HTTP/1.1, January 1997.

[27] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced CPU energy. In *First Symposium on Operating Systems Design and Implementation*, pages 13–23, Monterey, California, U.S., 1994.