

Energy-Efficient Server Clusters ^{*}

E.N. (Mootaz) Elnozahy, Michael Kistler, and Ramakrishnan Rajamony

Low-Power Computing Research Center
IBM Research, Austin TX 78758, USA.
<http://www.research.ibm.com/ar1>

Abstract. This paper evaluates five policies for cluster-wide power management in server farms. The policies employ various combinations of dynamic voltage scaling and node vary-on/vary-off (VOVO) to reduce the aggregate power consumption of a server cluster during periods of reduced workload. We evaluate the policies using a validated simulator that calculates the energy usage and response times of a Web server cluster serving traces culled from real-life Web server workloads.

Our results show that a relatively simple policy of independent dynamic voltage scaling on each server node can achieve savings ranging up to 29% and is competitive with more complex schemes for some workloads. A policy that brings nodes online and takes them offline depending on the workload intensity also produces significant savings up to 42%. The largest savings are obtained by using a *coordinated* voltage scaling policy in conjunction with VOVO. This policy provides up to 18% more savings than just using VOVO in isolation. All five policies maintain server response times within acceptable norms.

Keywords: Power Management, Clusters, Voltage Scaling, Web Servers

1 Introduction

Power consumption is rapidly becoming a key design issue for servers deployed in large data centers and web hosting facilities. Anecdotal evidence from data center operators indicates that a significant fraction of the operation cost of these centers is due to power consumption and cooling. Computing nodes in these densely packed systems also often overheat, leading to intermittent failures. These problems are likely to worsen as newer server-class processors offer higher levels of performance at the expense of increased power consumption.

This paper explores five policies for reducing the energy consumption of server clusters with varying degrees of implementation complexity. The first policy, *independent voltage scaling (IVS)*, simply uses voltage scaled processors that independently vary their voltage and frequency according to the server workload. The second policy also uses voltage scaled processors, but coordinates the processor voltage scaling algorithms. We call this *coordinated voltage scaling (CVS)*.

^{*} This research has been supported in part by The Defense Advanced Research Projects Agency under contract F33615-00-C-1736.

The third policy uses non-voltage scaled processors and turns entire servers on or off depending on the workload intensity. This policy, which we call *vary-on vary-off (VOVO)*, was originally proposed by Pinheiro et. al. [11], and has been evaluated for web-server clusters [4,11] and compute-server clusters [11]. The fourth policy combines IVS and VOVO, while the fifth uses a combination of coordinated voltage scaling and VOVO.

We use workloads constructed from the server access logs of the 1998 Nagano Winter Olympics website and those of a financial services web site to evaluate the five policies in terms of both their response time and energy savings. The evaluation is performed using a simulation model of a web server cluster. This simulation model is an extension of a previously developed single-node web server simulator that has been extensively validated for accuracy in both energy consumption and response times against energy and response time measurements from a commodity server [1].

Our findings show that independent voltage scaling, the simplest of all policies in terms of implementation complexity¹, offers energy savings ranging from 20% to 29%. Coordinated voltage scaling offers slightly better savings than IVS, but this benefit is probably not sufficient to justify the increased implementation complexity. The energy savings afforded by VOVO are workload dependent. For the Finance workload, VOVO saves more energy than IVS. However, IVS saves more energy than VOVO for Olympics98. Combining voltage scaling with VOVO offers the most energy savings with VOVO-IVS saving more energy than either voltage scaling or VOVO in isolation. VOVO-CVS saves the most energy (up to 18% more than VOVO) at the expense of a more complicated implementation. All five policies can be engineered to keep server response times within acceptable norms.

The remainder of this paper is organized as follows. In Section 2 we present five policies for power management in server clusters. In Section 3, we analytically derive an optimal operating frequency range for servers in a cluster with voltage-scaled processors. This derivation forms the basis for the VOVO-CVS policy. Section 4 describes the evaluation methodology for the various cluster power management policies using workloads based on logs from real web servers and presents the results of our evaluation. A comparison to related work is presented in Section 5 and conclusions in Section 6.

2 Cluster Power Management

We explore the benefits of five processor power management policies for clusters. The policies use two basic power management mechanisms, dynamic voltage scaling and node vary-on/vary-off. We begin by briefly describing these mechanisms, and then explain how these mechanisms are employed in the policies.

¹ We use the term “implementation complexity” to refer to the *additional* work required in integrating commodity components to implement a policy. More specifically, we do not consider the complexity of implementing or designing the commodity components.

Dynamic voltage scaling adjusts the operating voltage and frequency of the CPU to match the intensity of the workload. Voltage scaling reduces energy consumption by virtue of the fact that the energy consumed by a processor is typically directly proportional to V^2 , where V is the operating voltage. To ensure reliable operation, processor frequency must also be reduced proportionally with voltage. Setting the processor frequency (and thence the voltage) to the lowest point where the workload can be executed with adequate responsiveness results in reduced energy consumption. In most high performance server systems, the CPU provides the greatest opportunity to reduce power consumption, since other components consume a smaller fraction of the total system power, exhibit much less variation in energy consumption with workload, and could substantially impact system performance and/or reliability if power managed. Voltage scaling works particularly well in Web servers since typically, their configured capacity is significantly larger than the average encountered workload [1]. Dynamic voltage scaling is an intra-node power management mechanism.

Node vary-on/vary-off takes whole nodes offline when the workload can be adequately served by a subset of the nodes in the cluster. Machines that are taken off-line may be powered off or placed in a low power state. Machines that are off-lined are placed back online should the workload increase. Node vary-on/vary-off is an inter-node power management mechanism.

All five policies assume that the incoming workload is balanced across cluster nodes using a mechanism such as weighted round-robin request distribution, with the servers weighted according to the average response time of recent requests.

Independent Voltage Scaling (IVS): In this policy each node independently manages its own power consumption using dynamic voltage scaling. This policy performs no inter-node power management, so all the nodes in the cluster stay in the active state even during periods of low workload. Each node may operate at different frequencies and voltages due to workload variations and the differences in the computational demands of individual requests. However, since the request distribution mechanism balances the workload across all nodes, on average, each node will operate at roughly the same frequency and voltage. While IVS requires that the cluster nodes have processors and infrastructure that support dynamic voltage scaling, no other software support is necessary. In particular, a cluster composed of nodes using Transmeta Crusoe™ processors implements this policy.

Coordinated Voltage Scaling (CVS): This policy uses dynamic voltage scaling in a coordinated manner to reduce cluster power consumption. In contrast to IVS, the cluster nodes coordinate their voltage scaling actions so that all nodes operate very close to the average frequency setting across the cluster. A centralized monitor periodically computes the average frequency setting of all active nodes and broadcasts it to all servers in the cluster. Each node then restricts its voltage scaling policy to a small interval of settings around this average frequency. As in the IVS policy, there is no inter-node power management. Hence, all the nodes in the cluster are active even during periods of low workload. The CVS policy is expected to save more energy than IVS because a cluster where

the nodes operate at a particular frequency/voltage setting S is more efficient than one where the nodes operate at multiple settings whose average is S . To implement the CVS policy, cluster processors must support software controlled dynamic voltage scaling, such as that afforded by AMD'sTM PowerNowTM technology. In addition, CVS requires a central facility that monitors the frequency settings of all nodes and disseminates the average setting to the cluster. This monitoring facility can be implemented as a software service running on one of the cluster nodes (or a separate support server) with software probes running on each of the nodes.

Vary-On Vary-Off (VOVO): This policy, originally proposed by Pinheiro et. al. [11], turns off server nodes so that only the minimum number of servers required to support the workload are kept active. Nodes are brought online as and when required. VOVO does not use any intra-node voltage scaling, and can therefore be implemented in a cluster that uses standard high-performance processors without dynamic voltage scaling. However, some hardware support, such as a Wake-On-LAN network interface, is needed to signal a server to transition from inactive to active state. Node vary-on/vary-off can be implemented as a software service running on one of the cluster nodes (or a separate support server) to determine whether nodes must be taken offline or brought online and requires software probes running on each of the cluster nodes to provide information on the utilization of that node. The load distribution mechanism must be aware of the state of the servers in the cluster so that it does not direct requests to inactive nodes, or to nodes that have been selected for vary-off but have not yet completed all received requests.

Combined Policy (VOVO-IVS): This policy is a combination of the VOVO policy to reduce the number of active servers and the IVS policy to reduce power consumption on individual nodes. VOVO-IVS can be easily implemented using Transmeta CrusoeTM processors in a VOVO cluster. Thus, the implementation complexity of VOVO-IVS is similar to that of VOVO.

Coordinated Policy (VOVO-CVS): Conceptually, this policy is a combination of the VOVO policy to use the fewest number of active servers and the CVS policy to reduce the power consumption of individual active nodes. However, in this policy the power management actions of the two policies are integrated to use the most effective mechanism whenever the maximum cluster capacity is not needed. In particular, VOVO-CVS places a larger emphasis on voltage scaling than VOVO-IVS, since voltage scaling provides a quadratic benefit whereas node vary-on/vary-off provides only a linear benefit. The trade-off between these two mechanisms is heavily dependent on the power consumption characteristics of the processor and system components, and is discussed in detail in 3. Operationally, the policy achieves this integration by constraining the range of frequency and voltage settings based on the number of active nodes in the cluster. This range also dictates when a node is brought online or offline. A new node is brought online when the existing nodes find they have to operate above the allowed frequency range. Similarly, an existing node is taken offline when the nodes attempt to operate below their allowed frequency range.

VOVO-CVS is more complicated to implement than VOVO. The voltage scaling component requires processors that support software controlled dynamic voltage scaling such as that afforded by AMD’s™ PowerNow™ technology. In addition, the CVS component requires a central facility that communicates the optimal operating range to the software that sets the individual processor frequency/voltage settings. The VOVO component requires a central monitoring service to determine whether nodes must be taken offline or brought online and probes to monitor the utilization on each node. Finally, as with the VOVO policy, the load distribution mechanism must be aware of the state of the servers in the cluster so that it does not direct requests to inactive nodes, or to nodes that have been selected for vary-off but have not yet completed all received requests.

3 Details of the Coordinated Policy

In this section we explore in detail how to best integrate the VOVO policy, which expands and contracts the set of active servers based on workload, and the CVS policy, which uses software managed dynamic voltage scaling to ensure all nodes operate at a low average frequency and voltage. We construct a simple model for the power consumption of the cluster and use this model to determine how the two policies should cooperate to minimize power consumption. This theory will then be used as the basis for our coordinated power management policy.

As previously discussed, CPU power is the largest and most variable component of system power consumption in typical server systems. Therefore, the model assumes that power consumption of all other system components is essentially constant regardless of system activity. Our earlier empirical observations support the validity of this assumption [1]. Since our processors support dynamic voltage scaling, CPU power consumption depends on the CPU voltage and frequency as well as CPU utilization. To simplify the model, we will assume the CPU is fully utilized. Another way to view this assumption is that during periods of low workload, CPU frequency can be lowered to the point that the processor is kept fully utilized. The dynamic component of CPU power consumption – the only part affected by system load – is given by the following formula [10]:

$$\text{Dynamic CPU Power} = A C V^2 f$$

where A is an activity factor that accounts for how frequently gates switch, C is the total capacitance at the gate outputs, V is the voltage of the processor, and f is the operating frequency. In voltage scaled processors, decreasing the voltage requires that frequency also be reduced in approximately the same proportion. Furthermore, we can express voltage as a linear function in the frequency $V = \alpha f$. Substituting for V and combining constants, our formula for dynamic power of the CPU can be reduced to a function of processor frequency:

$$\text{Dynamic CPU Power} = c_1 f^3$$

For convenience, we express f in terms of a ratio to the maximum CPU frequency, and thus $0 \leq f \leq 1$. When the CPU runs at a reduced frequency,

it takes longer to complete a given task. We assume that this slow-down is proportional to frequency – that is, the number of cycles required to perform any given task is fixed regardless of processor frequency. This will be generally true for workloads where the CPU is the critical resource, such as web server workloads where the data mostly fits in the filesystem cache. Even for CPU intensive workloads, this is a conservative estimate, since at lower frequencies, the processor spends less time stalled on accesses to memory, network, disk, etc.

Since the power consumption of all other components in the system is essentially constant, we now have the following simple model of the power consumed by one node in the cluster running at frequency f :

$$\text{System Power} = P(f) = c_0 + c_1 f^3 \quad (1)$$

where c_0 is a constant that includes the power consumption of all components except the CPU, plus the base power consumption of the CPU. Power consumption of a cluster is just the sum of the system power consumed by its active nodes. The c_0 and c_1 terms should also include power loss due to power supply inefficiency. This power loss is typically more significant at low power levels, and thus may have a greater effect of the c_0 term than the c_1 term. If a single power supply is used for all systems in the cluster, power loss also depends on the number of active servers. However, power supply efficiency increases quickly with load until it reaches a nearly steady level, so for simplicity we treat power supply loss as a fixed fraction of overall power consumption.

Now consider a cluster of n systems operating at frequency f_1 . Using the model of system power stated above, the power consumed by these n systems is:

$$n \times P(f_1) = n \times (c_0 + c_1 f_1^3)$$

When the frequency f_1 is low, it can be beneficial to vary off (turn off) a server and consolidate the workload on remaining $n - 1$ servers. This would allow us to eliminate one system's fixed power consumption at the cost of slightly increased variable energy consumption on the remaining $n - 1$ systems. In this scenario, to maintain the same response time and performance in the remaining $n - 1$ systems we must increase the processor frequency in each system to $\frac{n}{n-1}f_1$, causing the power consumed by these $n - 1$ systems to become:

$$(n - 1) \times P\left(\frac{n}{n-1}f_1\right) = (n - 1) \times \left(c_0 + c_1 \left(\frac{n}{n-1}f_1\right)^3\right)$$

Thus, the configuration with $n - 1$ servers consumes less energy than with n servers when

$$(n - 1) \times \left(c_0 + c_1 \left(\frac{n}{n-1}f_1\right)^3\right) < n \times (c_0 + c_1 f_1^3)$$

$$(n - 1) \left(\frac{n}{n-1}\right)^3 f_1^3 < \frac{c_0}{c_1} + n f_1^3$$

$$\frac{2n^2 - n}{(n - 1)^2} f_1^3 < \frac{c_0}{c_1}$$

Thus, if the cluster has n active servers, we can reduce power consumption in the cluster by varying off one member of the cluster when the average CPU frequency of the backend servers falls below

$$f_{varyoff}(n) = \sqrt[3]{\frac{c_0}{c_1} \frac{(n - 1)^2}{2n^2 - n}}$$

Likewise, when the frequency f_1 is high, it can be beneficial to vary on (turn on) a server and spread the workload across $n + 1$ systems. The reduced workload on each server allows us to decrease processor frequency in each system to $\frac{n}{n+1} f_1$ while maintaining the same response time. In this configuration, the power consumed by the $n + 1$ systems is:

$$(n + 1) \times P\left(\frac{n}{n + 1} f_1\right) = (n + 1) \times \left(c_0 + c_1 \left(\frac{n}{n + 1} f_1 \right)^3 \right)$$

Constructing an inequality with the power consumed by n systems and solving for f_1 in the same manner as above, we find that when the average CPU frequency of the backend servers rises above $f_{varyon}(n)$, we can reduce power consumption in the cluster by varying on a new member of the cluster.

$$f_{varyon}(n) = \sqrt[3]{\frac{c_0}{c_1} \frac{(n + 1)^2}{2n^2 + n}}$$

Thus, given the constants c_0 and c_1 , the optimal average frequency range for a cluster of n systems is simply

$$f_{varyoff}(n) \leq \text{CPU frequency} \leq f_{varyon}(n)$$

Now we apply this theory to determine the optimal operating frequency for our simulated cluster nodes. The simulated CPU has a frequency range of 600MHz to 1.2GHz, and corresponding operation voltage range of 1.15 V to 1.4 V, and the rest of the system consumes 8.5W regardless of workload. When completely idle, the server consumes approximately 13.5 Watts (5 Watts for the constant processor power, and 8.5 watts for the rest of the system), and when fully loaded (CPU 100% busy), the system consumes 36.2 Watts. The idle power roughly corresponds to system power at CPU frequency $f = 0.0$, from which we can estimate $c_0 = 13.5$ Watts. (This estimate is not entirely accurate, since Linux places the CPU in the Halted state when the system is idle. However, for the purpose of this illustration, this small error can be ignored.) Likewise, the fully loaded power corresponds to the system power at CPU frequency $f = 1.0$, from which we can estimate $c_1 = 22.7$ Watts.

Using these parameters, we can compute the “vary-on” and “vary-off” threshold frequencies for any value of n , and thence the ideal operating frequency for a

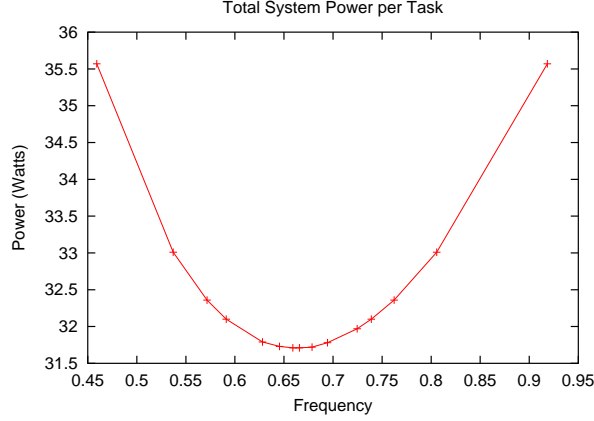


Fig. 1. Optimal Power Consumption for a Server Cluster

given cluster. Table 1 gives some representative values, which are also illustrated graphically in Figure 1. There are two points to note. First, for a particular implementation, the table *can be precomputed* and stored to control the operating range of the cluster and the activity state of each node. Furthermore, for a particular cluster, this theory suggests the power management policy presented as VOVO-CVS in Section 2.

n	$f_{varyoff}$	MHz $_{varyoff}$	f_{varyon}	MHz f_{varyon}
1	0.0000	0	0.9256	1111
2	0.4628	555	0.8119	974
3	0.5413	650	0.7681	922
4	0.5761	691	0.7447	894
5	0.5958	715	0.7302	876
10	0.6329	760	0.6998	840
20	0.6505	781	0.6839	821

Table 1. Optimal average frequency range for a cluster with 1200MHz processors

Our derivation of the optimal frequency range assumes that inactive servers consume no power, but this can easily be adapted for a case where inactive servers are placed in a low power state. For example, a server could be placed in *standby mode*, where the CPU consumes almost no power, but memory and peripherals remain powered so that state is not lost. A server in standby mode can be brought online much more rapidly than a server that is powered off. In this case, the power consumed in standby mode becomes a fixed cost of the cluster that is independent of the number of servers currently active. To determine the

vary-on and vary-off frequencies for the optimal range, we can simply subtract the standby power from c_0 , and then use the originally derived formulas. In other words, the c_0 constant really represents the *additional* fixed cost of the system incurred by placing it into the active state.

Bounds on Operating Voltage/Frequency

In computing the vary-on and vary-off points for the optimal range, we also must consider the bounds on operating voltage and frequency of the processor. There are two cases that must be considered:

$$\frac{n}{n-1} f_{varyoff}(n) > f_{max}$$

A cluster with n nodes operating at the varyoff frequency $f_{varyoff}(n)$ cannot set the frequency of $n-1$ nodes to the prescribed frequency because it exceeds the maximum processor frequency.

$$\frac{n}{n+1} f_{varyon}(n) < f_{min}$$

A cluster with n nodes operating at the varyon frequency $f_{varyon}(n)$ cannot set the frequency of $n+1$ nodes to the prescribed frequency because it is below the minimum processor frequency. In Table 1, this happens with a system containing two active nodes.

In the first case, we must reduce the varyoff frequency for n nodes to the point at which $n-1$, running at f_{max} , can handle the work of n nodes running at the varyoff frequency. In other words, we simply set

$$f_{varyoff}(n) = f_{max} \times \frac{n-1}{n}$$

In the second case, starting a new node at $f_{varyon}(n)$ will result in higher energy consumption, since the processors will be running at a higher frequency than specified by the optimal range. On the other hand, waiting to start a new node until the expected new frequency is f_{min} will also typically waste energy, since the corresponding frequency of the n node cluster could be significantly above the varyon point. Thus, the optimal transition point lies somewhere between these two extremes. At the transition point, we should have

$$n \times P(f) = (n+1) \times P(f_{min})$$

We should expect that our $n+1$ systems are not fully utilized, and therefore we include a utilization factor u in the power equation for $n+1$ systems.

$$\begin{aligned} n * (c_0 + c_1 f^3) &= (n+1)(c_0 + u(c_1 f_{min}^3)) \\ n * c_1 f^3 &= c_0 + (n+1)u(c_1 f_{min}^3) \\ f^3 &= \frac{c_0/c_1 + (n+1)u(f_{min}^3)}{n} \end{aligned}$$

If the two configurations perform equal work, then

$$f = \frac{n+1}{n}u \times f_{min} \quad (2)$$

Substituting into the previous equation:

$$\begin{aligned} \left(\frac{n+1}{n}u \times f_{min}\right)^3 &= \frac{c_0/c_1 + (n+1)u(f_{min}^3)}{n} \\ \frac{(n+1)^3}{n^3}u^3 \times f_{min}^3 &= \frac{c_0/c_1 + (n+1)u(f_{min}^3)}{n} \\ \frac{(n+1)^3}{n^2}f_{min}^3 \times u^3 - (n+1)f_{min}^3 \times u - c_0/c_1 &= 0 \end{aligned}$$

At this point we have a cubic equation in u , which has a closed form solution². Solving for u and substituting into equation 2 gives the optimal value of f_{varyon} .

4 Evaluation

4.1 Methodology

We constructed workloads using web server logs obtained from several production Internet servers. The first workload, Olympics98, is derived from the requests received on Feb 19, 1998 at the Nagano Winter Olympics servers at Columbus [2]. The second workload, Finance, is derived from the requests recorded on Oct 19, 1999 at the web site of a major financial services company.

Workload	Olympics98	Finance
Avg requests / sec	97	16
Peak requests / sec	171	46
Avg requests / conn	12	8.5
Files	61,807	16,872
Total file size	705 MB	171 MB
Requests	8,370,093	1,360,886
Total response size	49,871 MB	2,811 MB
97%/98%/99% (MB)	24.8 / 50.9 / 141	3.74 / 6.46 / 13.9

Table 2. Characteristics of two web server workloads.

Table 2 summarizes the characteristics of the two workloads. The table shows the average and peak request rates per second over the entire day. Additionally, the table shows the average requests per connection, which is based on grouping

² A closed form equation for the roots of a general cubic equation was first published by Girloamo Cardano (1501-1576) in his book on Algebra titled *Ars Magna*

requests into connections. The table includes statistics about the data served, including the number of files requested, their total size, and the total response size in bytes which represents the the total size of response data sent back to clients (excluding HTTP headers). The 97%/98%/99% data is the amount of memory needed to hold the unique data for 97%/98%/99% of all requests and gives an indication of the potential benefits of a RAM-based cache. For example, in the Olympics98 workload, the total size of the data required to serve 99% of requests is only 141 MB. Relatively speaking, modern systems can cache this amount entirely in memory, reducing the disk activity required to serve the workload.

From the two base workloads, we generate workloads of different intensities by scaling the inter-arrival time of connections (but not requests within a connection) by a constant amount. A scalefactor of “2×” corresponds to reducing the inter-arrival time of connections by 50%. This results in an increase in connection arrival rate corresponding to the scalefactor, but maintains the same basic pattern of connection arrivals. Inter-arrival time of requests within a connection are not scaled since these times 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 all three workloads to “2.5×”, “4×”, etc. This methodology has been used by other researchers [4].

Workload	Olympics98	Finance
Measured CPU Energy (J)	1,232,710	711,415
Simulator CPU Energy (J)	1,253,652	739,200
Error in Total Energy	1.70%	3.91%
Correlation Coefficient	0.9846	0.9960

Table 3. Comparison of Measured to Simulated CPU energy for the two workloads. Correlation coefficients were computed based on the energy used in 30 second intervals over the length of the run.

We extended our simulation model for energy consumption and response time of a web server [1] to model a cluster of web servers. The energy consumption reported by the simulator was validated against actual measurements performed on a “commodity” web server system with a 600MHz processor; these results are summarized in Table 3. A complete description of the methodology is presented elsewhere [1]. Our simulated CPU supports dynamic voltage scaling with frequency range of 600MHz to 1.2GHz, and corresponding voltage range of 1.15 V to 1.4 V. These parameters are loosely based on currently available processors designed for mobile systems. We present results for two different values of base system power consumption (the c_0 component of Equation 1): 8.5W, corresponding to current commodity servers, and 5W, which we present as representative of a future power-efficient server system.

The form of dynamic voltage scaling we use permits the CPU frequency to be set to a limited number of discrete settings. The processor voltage is set to the lowest value that allows reliable operation at a given frequency. The processor voltage and frequency are adjusted at regular intervals based on CPU utilization. At the beginning of each interval, the system determines the average CPU utilization over some set of recent intervals, and then uses a simple thresholding scheme to select the CPU frequency for the next interval. If the current CPU frequency is below the maximum setting and CPU utilization exceeds a high threshold, the CPU frequency is set to the next highest discrete frequency setting. Likewise, if the CPU is not currently at the minimum frequency and CPU utilization is less than a low threshold, the frequency is decreased one step. When the processor utilization is between the high and low thresholds, the frequency is left unchanged. In our study we use a low and high threshold of 80% and 95% respectively. This form of dynamic voltage scaling is commonly referred to as PAST and was originally proposed by Weiser et. al. [16].

Servers placed in the inactive state consume no energy. When a server is selected to be varied off, the distribution mechanism stops sending new requests to this server. When the server completes all its outstanding requests, it transitions to the inactive state after a “shutdown” time. When a server is brought online, it transitions to the active state after a “startup” time. We set both startup and shutdown times to 30 seconds in the simulations. When a varyon or varyoff is initiated, our policy inhibits subsequent node power management actions for the next 60 seconds (twice the transition time), to allow the node to transition and the cluster to adjust to its new configuration. In particular, this time should be sufficient for a newly started web server to warm up its file cache.

4.2 Results

Table 4 summarizes the results of our simulations. For both Finance and Olympics98 workloads, the table shows the energy and response times for the five policies described in Section 2. The total energy data is also presented graphically in Figure 2.

For both workloads, the IVS policy exhibits significant savings over the case without any power management. However, while the CVS policy does save more energy than IVS, the extra savings only amount to about 1%. Since the implementation complexity of CVS is far greater than that for IVS, the findings indicate that a first step in saving energy is to simply populate a server cluster with voltage scaled processors. Note that the response time afforded by both policies is well within acceptable limits. Compared to desktop and mobile workloads, relatively larger response times are acceptable for web workloads because the response time perceived by the client is typically much larger when factoring other components such as WAN delays [12].

Comparing IVS with VOVO yields an interesting result. While VOVO saves more energy for the Finance workload, the opposite is true for Olympics98. The switch comes about because of the nature of the two workloads [1]. After a steep increase in intensity at 9:30am, the Finance workload maintains a high intensity

Workload	Policy	Energy ¹			Energy ²			Response Time	
		K Joules	% savings over		K Joules	% savings over		Average (ms)	90%-ile (ms)
			None	VOVO		None	VOVO		
Finance	None	17879	—	—	14785	—	—	5	8
	IVS	14404	19.4	—	11310	23.5	—	10	22
	CVS	14216	20.5	—	11122	24.8	—	12	26
	VOVO	10313	42.3	—	9133	38.2	—	14	26
	VOVO-IVS	9237	48.3	10.4	8062	45.5	11.7	16	34
	VOVO-CVS	8985	49.7	12.9	7469	49.5	18.2	19	38
Olympics98	None	20977	—	—	17925	—	—	5	9
	IVS	15821	24.6	—	12769	28.8	9.2	13	29
	CVS	15445	26.4	0.8	12393	30.9	11.9	15	34
	VOVO	15818	24.6	—	14064	21.5	—	13	24
	VOVO-IVS	14605	30.4	7.7	12852	28.3	8.6	18	34
	VOVO-CVS	13968	33.4	11.7	11533	35.7	18.0	36	38

Table 4. Simulated energy and response time for the different cluster power management policies. Energy¹ uses a fixed cost of 8.5 W, and Energy² uses a fixed cost of 5 W. Except for VOVO-CVS, response times are independent of the fixed cost. For VOVO-CVS, the fixed cost plays a role in determining when nodes are brought online and taken offline, causing the response times to vary. Here, the variation is minor. 90% of responses were satisfied below the 90%-ile response time numbers.

level until about 6pm. Afterward, the Finance workload decreases in intensity. In comparison, the Olympics98 workload is more stable. VOVO saves more energy than IVS for Finance only because during the periods of low intensity (late evening and night), VOVO is able to place machines offline. IVS becomes more favorable as the fixed costs of the cluster components decrease.

VOVO-IVS always performs better than VOVO. For the set of workloads and fixed costs, the improvement ranges from about 8% to 12%. At the same time, the response time degradation is well within acceptable norms. These findings suggest that when building web clusters, using voltage scaled processors in *combination* with VOVO provides increased energy savings with minor (for example, use Transmeta processors) additional implementation cost.

The policy that yields the most energy savings is VOVO-CVS. Compared to VOVO, VOVO-CVS is able to provide savings ranging from 12% to 18%. When compared to VOVO-IVS, the savings range from 3% to 10%. While VOVO-CVS has an increased implementation complexity compared to VOVO, the savings it affords increase as the fixed costs of the system decrease. VOVO-CVS saves up to 50% of the energy expended by the cluster with no power management.

Figures 3 through 5 present the request rate, power consumption, and number of active servers for the VOVO and VOVO-CVS policy over the course of both workloads. Note that for both policies, power consumption closely follows the workload on the cluster, with the VOVO-CVS policy consuming less power at almost all workload intensities. The graph shows two or three instances in which the VOVO-CVS consumes more power for a short period, all of which occur

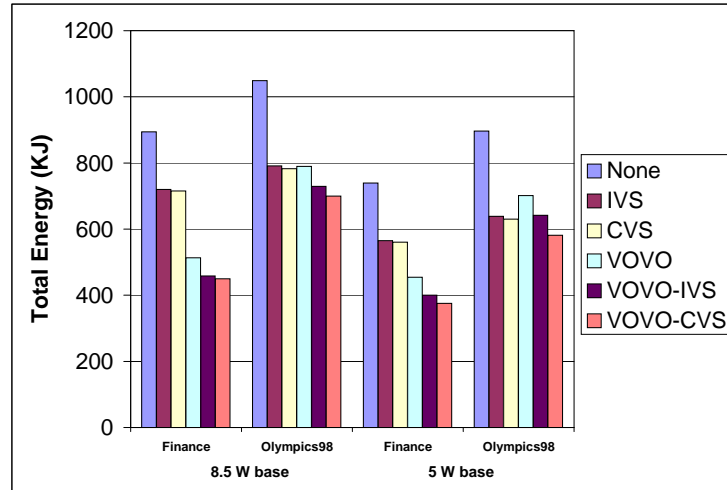


Fig. 2. Total Energy Consumed by workload and base power consumption for our five cluster power management policies.

when workload levels off after a period of rapid increase. In Figure 5 shows that VOVO-CVS often uses more servers than VOVO; this is the fundamental source of power savings provided by VOVO-CVS. By using more servers, the variable component of power consumption is reduced quadratically by voltage scaling, to the point where it balances against the additional fixed cost of a larger active set.

Figure 6 presents the power savings from the CVS, VOVO, and VOVO-CVS policies over a server cluster without power management over the course of both workloads. These graphs indicate that at low workload intensities, the VOVO policy provides large savings since the workload can be adequately served by a small set of machines. However, these savings drop significantly as workload increases, and VOVO provides relatively little benefit at moderately heavy workloads. In contrast, power savings from CVS are modest at low workload intensities, since the CPU is idle most of time, and thus obtains no benefit from voltage scaling. As workload increases, average CPU utilization increases, yielding increased savings from the CVS policy. Finally, by employing both vary-on/vary-off and voltage scaling, the combined VOVO-CVS policy achieves significant power savings across a broad range of workload intensities.

Another way to interpret these results is that node vary-on/vary-off is a coarse-grained mechanism that provides power savings in a few large increments, with corresponding large affects in cluster capacity. In contrast, voltage scaling is a fine-grained mechanism, since each individual power management action saves only a small amount of power, but also results in a very small decrease in cluster capacity. Furthermore, many components, particularly disk drives, are adversely affected by frequent power cycles, and thus node vary-on/vary-off

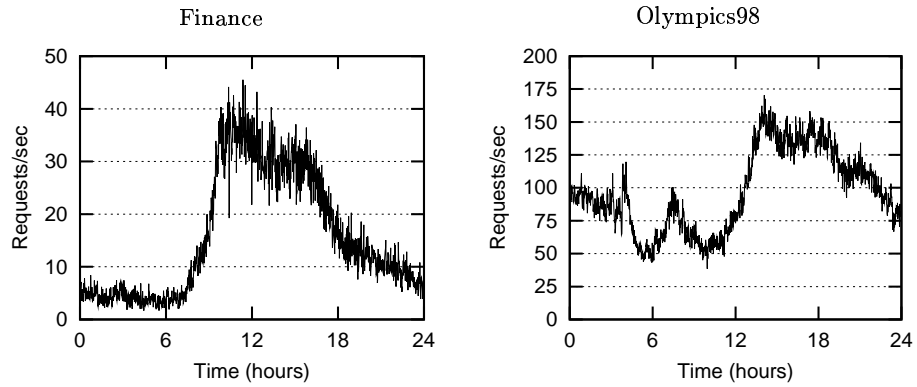


Fig. 3. Request rate

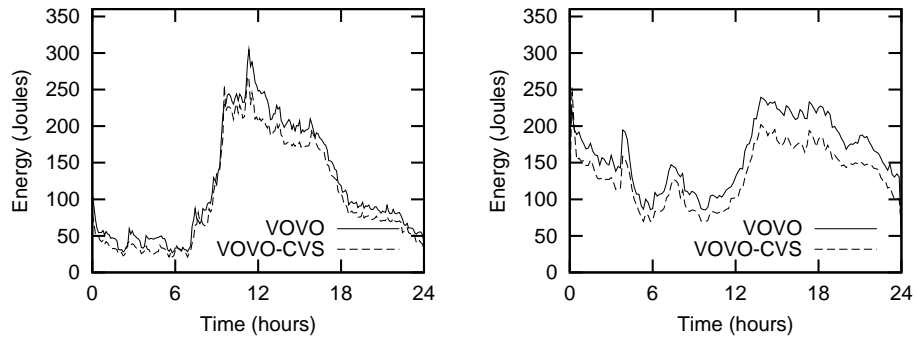


Fig. 4. Power consumption

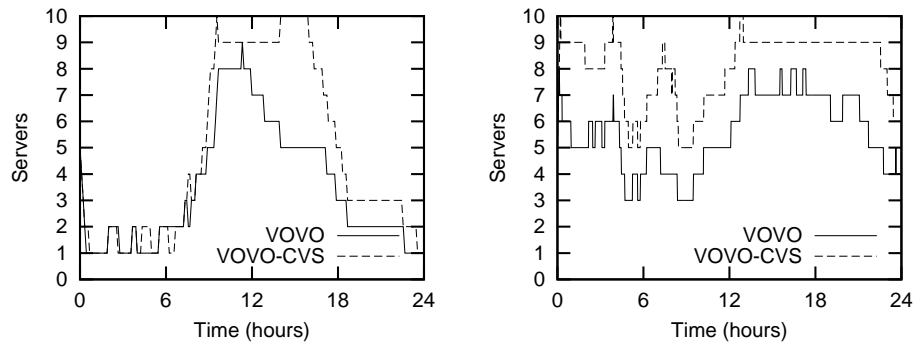


Fig. 5. Number of active servers

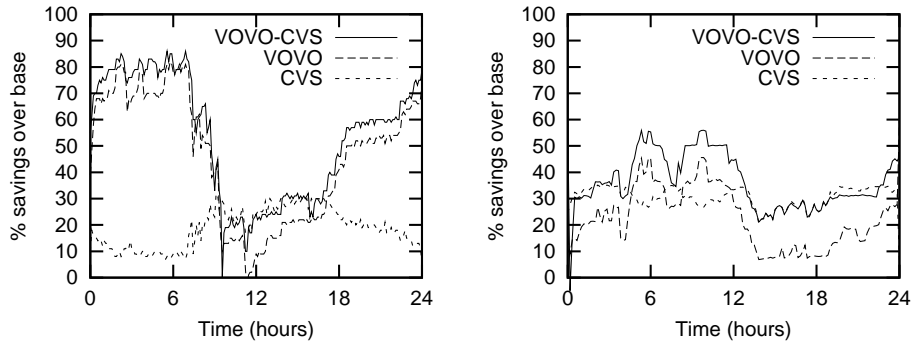


Fig. 6. Energy savings from three policies over time

must be constrained in the frequency of power management actions. Voltage scaling has no such adverse affect, and thus can be applied much more liberally. Thus, VOVO-CVS represents a combination of coarse- and fine-grained power management that outperforms either mechanism used alone.

5 Related Work

Reducing power consumption by reducing the clock frequency of the processor has been widely studied [16,8]. Until recently, this mechanism was employed almost exclusively in portable, battery powered systems as a means of extending battery life. However, power consumption and the related costs of cooling and reliability have led to a new focus on server systems with voltage scaled processors. Mudge [10] observed that the combination of voltage scaling and parallelism (e.g. clusters) could be applied to reduce energy consumption. Indeed, there is an emerging class of servers that are designed around processors with dynamic voltage scaling mechanisms [14], though to the best of our knowledge, there has been no formal study of the potential energy savings in clusters of such systems.

Flautner et. al. [5] explored a software-managed dynamic voltage scaling policy that sets CPU speed on a task basis rather than by time intervals. Lorch et. al. [9] describe how a task-based, software-managed voltage-scaling policy can be optimized when task completion times cannot be predicted accurately. Both of these techniques perform well for desktop application workloads, but it is unclear how they could be applied to server environments. Our work demonstrates how the same basic mechanism of software-managed dynamic voltage scaling can be used to achieve significant power savings in server clusters.

Pinheiro et. al. [11] proposed a simple policy for managing energy use in server clusters by powering machines on and off (similar to the VOVO policy). They examined this policy in the context of both web-server and compute-server clusters. Chase et. al. have employed this mechanism in the context of an economic framework in which web sites “bid” on resources based on their current workload. They report savings of 29%, 38%, and 78% for three different workloads. We believe the high energy savings they report for the third workload

(from the World Cup Trace) can be attributed to the significant variation in intensity of that workload. Our work complements and extends these studies by comparing VOVO to two forms of dynamic voltage scaling, and extending VOVO with voltage scaling mechanisms.

A wide range of techniques have been explored for power management in microprocessors, 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 [6]. 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 [15] and application layers [7] of the system. Techniques for dynamically controlling processor temperature [13] can also be applied to web servers. This results in power savings because CPU activity is decreased to lower processor temperature. Some of these mechanisms could be used to extend the gains from power management demonstrated in this paper.

6 Conclusions

In this paper, we have described and analyzed five distinct policies for power management in server clusters with varying degrees of implementation complexity. The policies employ various combinations of intra-node (dynamic voltage scaling) and inter-node (node vary-on/vary-off) power management mechanisms to reduce the aggregate power consumption of a server cluster during periods of reduced workload. We use a validated simulator to study the potential benefits of these policies for workloads derived from the server access logs of the 1998 Nagano Winter Olympics and a financial services web site.

We find that a relatively simple policy of independent dynamic voltage scaling on each server node yields energy savings between 20% and 29%. A policy that varies nodes on or off based on the workload intensity achieves between 22% and 42% savings at the expense of cluster-wide coordination. However, for some cases, we find that independent voltage scaling is more efficient without the need for cluster-wide coordination. The most energy savings are obtained by a policy that combines node vary-on/vary-off with voltage scaling with pre-computed optimal transition points. This combined policy achieves up to 18% more savings than pure vary-on vary-off, albeit at the expense of increased implementation complexity. Compared to a cluster that is not power managed, the combined policy saves between 33% and 50% of the cluster energy.

References

1. Pat Bohrer, Mootaz Elnozahy, Tom Keller, Michael Kistler, Charles Lefurgy, Chandler McDowell, and Ramakrishnan Rajamony. The Case for Power Management

- in Web Servers. In Robert Graybill and Rami Melhem, editors, *Power-Aware Computing*, Kluwer/Plenum Series in Computer Science, to appear. January 2002.
2. Jim Challenger, Paul Dantzig, and Arun Iyengar. A scalable and highly available system for serving dynamic data at frequently accessed web sites. In *Proceedings of 1998 ACM/IEEE Supercomputing (SC98)*, Orlando, Florida, November 1998, 1998.
 3. Rambus Corporation. Rambus Technology Overview, Feb 1999.
 4. Jeff Chase et. al. Managing Energy and Server Resources for a Hosting Center. In *18th Symposium on Operating Systems Principles (SOSP)*, October 2001.
 5. K. Flautner, S. Reinhardt, and T. Mudge. Automatic Performance Setting for Dynamic Voltage Scaling. In *Proceedings of the 7th ACM Int. Conf. on Mobile Computing and Networking (MOBICOM)*, July 2001.
 6. M. Fleischmann. Crusoe Power Management: Cutting x86 Operating Power Through LongRun. Embedded Processor Forum, June 2000.
 7. J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *17th ACM Symposium on Operating Systems Principles (SOSP'99)*, pages 48–63, 1999.
 8. K. Govil, E. Chan, and H. Wasserman. Comparing Algorithm for Dynamic Speed-Setting of a Low-Power CPU. In *Mobile Computing and Networking*, 1995.
 9. Jacob R. Lorch and Alan Jay Smith. Improving Dynamic Voltage Scaling Algorithms with PACE. In *ACM SIGMETRICS 2001*, June 2001.
 10. Trevor Mudge. Power: A First Class Architectural Design Constraint. *IEEE Computer*, 34(4):52–57, April 2001.
 11. E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Workshop on Compilers and Operating Systems for Low Power*, September 2001.
 12. R. Rajamony and M. Elnozahy. Measuring Client Perceived Response Times on the WWW. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01)*, Mar. 2001.
 13. Erven Rohou and Michael D. Smith. Dynamically Managing Processor Temperature and Power. In *2nd Workshop on Feedback-Directed Optimization*, November 1999.
 14. RLX Technologies. <http://www.rlxtechnologies.com/home.html>.
 15. 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.
 16. 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.