Using on-line power modeling for server power capping

Madhu Saravana Sibi Govidan[1], Charles Lefurgy[2], and Ajay Dholakia[2]
University of Texas at Austin[1] and IBM[2]
sibi@cs.utexas.edu, lefurgy@us.ibm.com, adholak@us.ibm.com

Abstract-- We have developed a self-tuning feedback controller that caps the power consumed by a server to a user-defined budget. The controller's parameters are tuned dynamically by computing an internal model of the server's power consumption and how it changes with frequency. This allows the controller to adapt in response to changing workload and changing system configuration. The power model encompasses both clock modulation and voltage-frequency scaling actuators so they can be used together for power capping. We show the self-tuning controller achieves acceptable settling times and has application performance similar to a controller based on fixed parameters that are determined off-line. The self-tuning controller saves the vendor time in determining good controller parameters and adjusts to unforeseen workloads and server configurations.

1. Introduction

The increasing power consumption of server systems is a serious problem in Internet Data Centers (IDC), leading to increased cooling costs, reduced performance and availability of the servers. One response from industry has been the development of **power capping**, a mechanism to control a server's power consumption to fit within a particular operating power budget. In the last few years IBM, HP, and Intel have offered a power capping feature in their products [4][5][6]. Current power capping solutions work by adjusting the microprocessor speed to keep the overall server power within the available power budget.

An important use of power capping is as a safety net that allows system designers to use cheaper, smaller power supplies that do not provide complete redundancy during failure. If a supply fails, the remaining one can usually run the server at full performance. However, some demanding workloads will require too much power and in this case, the power capping controller slows the system to a safe power consumption level. It is important that a power capping controller settle to a new power budget or adapt to a new workload quickly so that a redundant supply does not experience overload and thereby fail. A prior study [8] placed the power supply time constraint at the order of 1 second for blade servers. Application performance under a power cap is a less important metric, since failures are rare and some degradation in performance is an acceptable alternative to turning off the server completely.

Power capping controllers based on control theory take parameters that determine the controller's settling time under specific loads. These parameters are set based on power consumption models for the server – models that predict the relationship between the actuators for controlling microprocessor speed and the resulting power consumption. The precise relationship depends on multiple dimensions, including the demand of the workload, the configuration of the server (hot-plug components), and environmental conditions (temperature and airflow).

It is typical today for power capping controllers to use static parameters that do not change as the server operates. The parameters may be fixed at manufacturing time, which can add to design and validation time. Alternatively, the firmware runs a calibration workload at boot time, taking power measurements at fast and slow speed settings, and determining the parameters. A static controller that is optimized for a particular workload on a typical hardware configuration, running under typical environmental conditions, is practical only if the server is used in this limited manner. In practice, each of these multi-dimensional parameters can change and it is often possible for multiple parameters to change together. Therefore, a fixed controller is likely not to have acceptable settling time and provide good performance all the time. This is the appeal for a self-tuning controller that learns the server power model dynamically as it changes.

It is important to note that an optimal static controller for a given set of operating parameters may still achieve better metrics than a self-tuning controller which must take some time to learn its parameters. The value of a self-tuning controller is in shortening the vendor's product schedule by eliminating the need for long periods of pre-production and manufacturing time characterization across a large set of workloads, configurations, and environmental conditions. Additionally, the self-tuning controller may deal more appropriately with situations not covered during characterization and testing of fixed controllers.

In this paper, we describe an on-line, power modeling methodology that can be implemented at the system level in a self-tuning power-capping controller. Our results show that the controller achieves acceptable settling times, but without off-line effort to find appropriate fixed parameters.

This paper makes the following contributions:

1. We present the design of a piece-wise power controller that can use both dynamic voltage and frequency scaling (P-states) and clock modulation (T-states) to meet a requested power cap. In section 3, we describe the system under test and its power controller.

2. We present an on-line method of deriving a system-level power model that can be used to tune the power capping controller. In section 4, we discuss the design and operation of the on-line modeler.

3. We directly measure the settling time of the power capping controller and present the first published results.

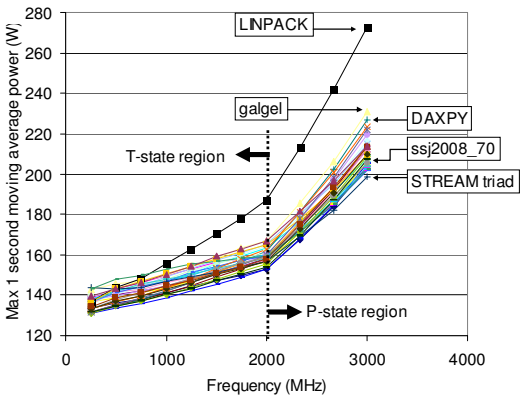

Figure 1: Power consumption of Server X.

4. Our results in section 5 show that an on-line power model can be as effective as an off-line, static model for capping power on our server. The main benefit is to save the manufacturer time in exhaustively searching for good, static controller parameters that work over many workloads and configurations, and guard against unforeseen cases.

2. Related work

Previous power capping studies [1][2][3][10] use fixed constants for the parameters of proportional or proportional-integral controllers and have not fully evaluated settling time for a wide range of application and configurations. We directly measure how settling time changes with a fixed-parameter controller and compare it to a self-tuning controller.

A different type of controller, called "ad-hoc controller", works by incrementing or decrementing the performance state of the processor by one unit to move measured power towards the power cap. Ad-hoc control is not recommended since it exhibits steady-state error that prevents it from settling precisely [1]. However, we find that the ad-hoc controller has useful properties for deriving a new power model when the present model fails. One of the major contributions of our work is to combine an ad-hoc controller together with a standard P-controller to create a self-tuning, power-capping controller.

Another study [2] has looked at using piece-wise linear models characterized off-line for dynamic voltage and frequency scaling (DVFS) and clock modulation actuators. We extend this by developing a method for on-line modeling and demonstrate its operation.

Prior off-line power models of servers relate component utilization to power [7]. We believe our work is unique in that it uses actual power measurements, instead of performance counters or utilization, to compute on-line power models to relate frequency to power.

3. Piece-wise controller

A prior study [1] designs and evaluates a P-controller that maintains the power supply load of a server. The controller's parameter is determined using off-line modeling. This controller only uses the clock modulation actuator of the microprocessor. For our study, we borrow this controller and build on it. In this section, we extend the controller to account for the different system power-performance response seen under clock modulation and DVFS controls. This allows our controller to work on systems that provide both types of control or only one type of control.

Controller selected performance state	Processor P-state	Processor T-state	Effective frequency
0	3.000 GHz	100.0%	3.000 GHz
1	2.667 GHz	100.0%	2.667 GHz
2	2.333 GHz	100.0%	2.333 GHz
3	2.000 GHz	100.0%	2.000 GHz
4	2.000 GHz	87.5%	1.750 GHz
5	2.000 GHz	75.0%	1.500 GHz
6	2.000 GHz	62.5%	1.250 GHz
7	2.000 GHz	50.0%	1.000 GHz
8	2.000 GHz	37.5%	0.750 GHz
9	2.000 GHz	25.0%	0.500 GHz
10	2.000 GHz	12.5%	0.250 GHz

Table 1: Available performance states

3.1. Power actuators

Our primary experiments are performed on an IBM HS21 blade server called "Server X". This server has two 3.0 GHz Intel 5160 dual-core (80 W) processors, 4GB memory, 1 disk, and two 1 Gb/s Ethernet ports. We utilize two mechanisms to control power consumption. First, we use P-states (DVFS) that set both the processor voltage and frequency. Server X has P-states of 2.000, 2.333, 2.667, and 3.000 GHz. The higher the frequency, the higher the corresponding voltage level. Second, we use T-states ("clock modulation" or "clock throttling") to operate the system below 2.0 GHz to extend the power capping range. The T-states use the voltage level of the 2.0 GHz P-state. There are 8 states that modulate the operation of the clock in steps of 12.5%. Table 1 shows how P-states and T-states are combined to provide 11 natural effective frequencies that the controller can select during a control interval.

Server X has no actuators to directly control memory power (20 W maximum). The processors are the primary consumers of power in the system and controlling the processor frequency alone provides a wide range for the power cap setting. Slowing the frequency indirectly affects memory power by slowing the request rate. Prior work in power control has reached similar conclusions [1][2][3].

Figure 1 shows power consumption of Server X for 30 workloads. The maximum 1 second moving average of power for fixed-frequency runs is plotted. We observe 1) the relationship between processor frequency and server power is piece-wise linear (correlation coefficient R^2 values > 0.99 in each region) and it is predictable and 2) the specific power-to-frequency ratio varies across workloads. Above 2 GHz, is the P-state region, and below 2 GHz, is the T-state region. In this server, the use of T-states significantly extends the power capping range of the server over using P-states alone. By using T-states, the minimum server power cap setting is about 150 W. If only P-states

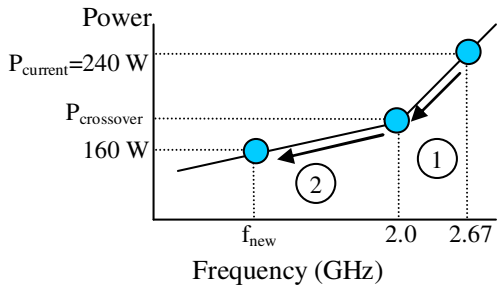

Figure 2: Controller driven by on-line power model.

were used, then the minimum setting would be about 190 W, limited by the LINPACK workload.

3.2. Prior static controller

The prior controller is a P-controller that selects an ideal frequency for each control period. The ideal frequency is realized by a delta-sigma modulator that modulates between natural frequencies over several control periods. The controller relies on the following simple, but effective model that relates power "P" and frequency "f":

$$P = A*f + c \quad (1)$$

This simple model has also proven effective for DVFS-based capping controllers in other studies [2][3][7][10]. The power model for the controller is a single parameter "A" expressing the power-to-frequency ratio. The constant c is not material to the controller's operation or performance [1].

During each control period, the controller measures the current power $P_{current}$ and frequency $f_{current}$ and calculates a new frequency f_{new} to move the system power toward the set point (power cap) P_s according to the equation:

$$f_{new} = 1/A*(P_s - P_{current}) + f_{current} \quad (2)$$

3.3. Piece-wise controller

In Server X, the P-state range and the T-state range have very different power-to-frequency ratios. Our modified controller takes two parameters, A_p and A_t, using $A=A_p$, when operating in the P-state range and $A= A_t$, when operating in the T-state range.

When the current frequency and new frequency are in the same linear region, or the server only has actuators for one region, then our controller operates like the prior controller. When the final frequency is not in the same region, we use a piece-wise linear approach to controlling power. Our current implementation uses two piece-wise linear regions, but it could be adapted to handle more regions if new actuators were added to servers, or to approximate non-linearity in the power consumption.

Figure 2 shows how the piece-wise controller crosses between linear regions. Assume that the system is currently operating at 2.67 GHz and the power cap is suddenly changed from 240 W to 160 W. First, the controller uses $A= A_p$ and equation 2. Since the result is below 2 GHz (the end of the P-state region), the controller notes that the P-state will change to 2 GHz and now it tries to find the appropriate T-state. This is done by applying equation 2 again using $A= A_t$, $f_{current}$ = 2 GHz (the crossover frequency), and $P_{current}$ = P(2 GHz) estimated by using equation 3.

$$P(f) = (2\ GHz - f)*A_p + P_{current} \quad (3)$$

Since our system only has two regions, we simplify our implementation a bit. The controller first uses the current region model to estimate the power at the crossover point of 2 GHz using equation 3 and the appropriate choice for A. Using the crossover power and the desired set point, the controller chooses either A_p or A_t to go in the desired direction from 2 GHz. This works even if the current and new frequencies are in the same region.

3.4. Implementation

Our power controller prototype is implemented in firmware running on the service processor that already exists on the blade server for remote management. The controller uses existing power measurement circuits in the server to read the 12V bulk DC power supply consumption every 1 ms with a precision of 0.1 W. We use this to create the 1 second moving average and 64 ms moving average of power used by the controller. The service processor sets the P-state and T-state used by the microprocessors.

4. On-line power modeling

The prior work on power control for servers has done extensive off-line characterization of the servers to set the power model parameters used by the controller [1][2][3][10]. Selecting good parameters is extremely important to ensure that the system settles to a given power cap within the desired time. If this requirement is not met, then the system will not be stable, and could even oscillate between the highest and lowest frequencies. In this section, we extend the controller introduced in Section 3 to learn an appropriate model that will allow the power to settle quickly.

4.1. Data collection

Prior studies that use off-line models generate them by running the system at different frequencies or utilizations and measuring the resulting power consumption. Our insight for on-line model creation is that the power capping controller itself can provide this data. As the controller is running, it measures power consumed during the last control interval and knows the frequency state of the processors. We use this stream of data for refining the power model and keeping it up-to-date as the workload and system configuration change. This means that the system does not need to spend any overhead time probing different performance states to find their power consumption.

4.2. Algorithm and implementation

Figure 3 illustrates the on-line power model and controller interaction. It operates as follows:

1. Every 64 ms control interval, the current performance state and 64 ms moving average power measurement are recorded into a table in the service processor. This table

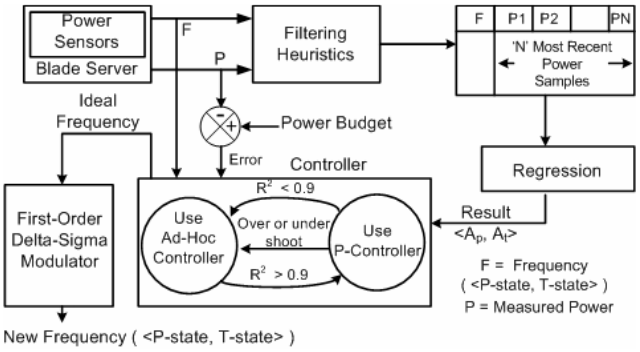

Figure 3: Controller driven by on-line power model.

holds up to four recent samples for a given performance state. As we assume that recent past is a good predictor of the near future, power samples older than 512 ms are discarded.

2. After updating the table, we perform two linear regressions with the current table values to update A_p and A_t. If the correlation coefficient $R^2 > 0.9$, the new power model is accepted for use.
3. Normally, our system uses the piece-wise P-Controller for power capping. Under the following two scenarios, our system discards the table data and starts using the ad-hoc controller:
   a. When the correlation coefficient of the regression R^2 is < 0.9. This occurs when the workload demand changes quickly.
   b. If the 1 second average power has remained above or below the power budget by 1 Watt for the last 6 control periods (empirically found to work well).
4. The ad-hoc controller works as follows: it lowers the processors by 1 performance state if either the 1 second or 64 ms average power is above the power budget. Otherwise, if the 64 ms average power is below the power cap, the ad-hoc controller raises the processors by 1 performance state.
5. While the ad-hoc controller is being used, power and performance state data is collected into the table. When the correlation coefficient (for either A_p or A_t) > 0.9, we use the piece-wise P-controller.

After most regressions, R^2 is greater than 0.9 and the power model is accepted for use. However, when certain scenarios, as described in 3a and 3b, occur, we fall back to the ad-hoc controller because the power models via regression are not accurate enough. Using the ad-hoc controller achieves two things: 1) the ad-hoc controller helps hold the power cap for two control intervals until the new power model is determined and 2) it guarantees a change in processor frequency, which in turn provides new measurement points for rebuilding an accurate power model. Two control intervals after falling back to the ad-hoc controller, the table has two recent power measurements at two different frequencies and regression can proceed. This yields $R^2 = 1$ which triggers use of the piece-wise P-controller. Using the ad-hoc controller for longer periods would result in steady-state error. Thus, our algorithm combines the ad-hoc and P-controllers to achieve better robustness than using either alone.

Server	Region	Minimum (W/GHz)	Maximum (W/GHz)	Average (W/GHz)
X	T-state	8.2	28.6	18.4
	P-state	41.5	86.3	63.9
Y	T-state	4.1	13.6	8.8
	P-state	19.8	41.2	30.5

Table 2: Range of power model across configurations.

In real deployments we suggest using a filtering stage before storing the 64 ms power samples in the table. It is important that the power and frequency data points be correlated with each other to derive an accurate model. If the frequency changed in the middle of a 64 ms control interval for a reason besides capping (due to user control, emergency temperature reduction, etc.), then the data point is discarded since it is not correlated with the desired frequency used by the controller. In our experiments, no data points were filtered out because the frequency was only changed by the power capping controller.

5. Results

Three identical runs are used to determine each experimental value reported. For settling time results, the maximum value was picked (to be conservative). For power measurement results, the average value was used. In addition to running experiments on Server X, we also introduce Server Y which has half the processors and memory of Server X. Our benchmark set is comprised of 30 workloads, including SPEC CPU2000 (rate mode), DAXPY, LINPACK, the triad phase of STREAM, and the SPECpower_ssj2008 70% load phase.

Table 2 shows the range of power models across all workloads on each server. The data for Server X comes from the slopes in Figure 1. The *Average* column shows the average model. The P-controller "A" parameter in [1] is selected using this average. This choice provides similar control performance (settling time) for the minimum and maximum workloads.

5.1. Workload variation

Now we examine how different workloads respond to different choices of static power model. In this experiment we exercise Server X in the P-state region using a broad range of values for A_p and graph the settling time in for power caps of 190, 200, and 210 W. The range of A_p is selected to span the range of models for Server X and server Y in Table 2. A_t is fixed to be 18.4 W/GHz which is the average from Table 2.

Our firmware tracks each episode when the power overshoots the cap and measures the time to return to the cap. The firmware measures settling time by incrementing a counter for each 64 ms control period that the 1 second

moving average power is 1 W above the cap. Once the power is under the 1 W guard band, then the value of the counter is recorded as the settling time and the counter is reset for the next overshoot episode. The maximum settling time observed across 3 identical runs for each workload run is shown in Figure 4.

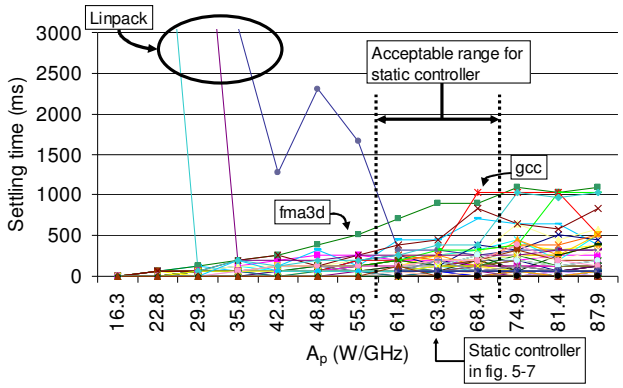

Figure 4: Settling time for Server X with static model.

Our goal is for the 1 second moving average power to settle within 1 second to within 1 W of the power cap. We choose a 1 W target because it is an amount of power that can have a measurable performance impact (~1%) on an application [1]. This aggressive target will become more meaningful as power capping and power shifting is deployed inside future servers to move power between processors and other components on demand as in [9]. Enforcing such precise power budgets ensures every Watt of available power may be used.

The results show that only a narrow range of A_p (61.8 – 68.4 W/GHz) allows the settling time to be 1 second or less across all 30 workloads on Server X. For LINPACK, when A_p is small, the controller exhibits oscillations in frequency and does not settle within 5 seconds. In the worst case, it overshoots a 190 W cap by up to 80 W and on average runs 22% faster than required to meet the cap. Finding the acceptable range is time consuming and error prone for vendors. It took 1 week to collect the data for the single configuration shown in Figure 4. The vendor must additionally understand the range of possible workloads and numerous upgrade configurations over the server's expected lifetime of 3-5 years to have confidence in static parameters.

5.2. Configuration variation

Server configuration has a large effect on acceptable values of A_t and A_p. In general, the model parameters learned for one server configuration do not apply to other server configurations.

For example, Table 2 shows that workloads on Server Y correspond to A_p between 19.8 W/GHz and 41.2 W/GHz. However, if this range were to be used on Server X, it is clear from that LINPACK would fail to settle in time due to oscillation in the controller. The reason for this is that LINPACK on Server X has a maximum power-to-frequency ratio of 86.3 W/GHz, which corresponds to a controller gain of over 2 (86.3 / 41.2). The P-controller must have a gain less than 2 to avoid oscillation [1]. Otherwise, the frequency will change by double the amount required to settle to the cap.

To avoid large differences in gain, a controller using a static model would need to re-evaluate it every time the server is booted or every time hot-plug components are added or removed.

Section 5.1, shows using an average A_p=63.9 W/GHz for Server X provides acceptable settling time. However, it is not clear this method of using an average power-to-frequency ratio will work for all future servers. If the minimum and maximum values for the ratio span a wider range, then a single value of A_p cannot prevent oscillations. We expect this will be the case for large, many-core servers. The promise of the self-tuning controller is that it should adapt with change in core utilization and workload.

5.3. Comparison between static and on-line models

In figures 5-7 we compare a piece-wise controller using the on-line model to a piece-wise controller that uses acceptable, average static parameters (A_t = 18.4 W/Ghz, A_p = 63.9 W/GHz) as suggested by [1].

Since the range for Server X in Table 2 is not extremely wide, a static model can have **very good** control performance across workloads on this configuration. Therefore, we do not expect the on-line modeling to have better settling time or application performance on Server X. Instead, we are looking to see if it can have substantially similar performance without the effort involved in finding and validating fixed controller parameters.

Figure 5 shows the maximum settling time that occurred. Both achieve the desired settling time of 1 second. LINPACK is a difficult workload for the on-line model. LINPACK is a non-steady workload that has dramatic power swings on the order of about 1 second. As soon as the on-line model adjusts to one phase, the phase ends and the model no longer works for the next phase and must be relearned. During the run we observe the controller using values for A_p that span the range in Table 2. The good news is that the controller rejects poor models before settling time becomes too great. For this server, a controller with an average static value for A_p settles better and does not suffer a penalty for temporarily using bad power models found as phases are changing. In the worst-case analysis, across all runs, the maximum settling time for self-tuning (LINPACK @ 190 W) is only 13% longer than the static controller (CFP2000 @ 190 W). We feel this is an acceptable trade-off for reducing vendor testing time.

Figure 6 shows the time the controller spends overshooting the cap by more than 1 W. The on-line model shows improvement for nearly all of the 30 workloads. On average across 30 workloads, this overshoot happens less than 1% of the time.

Figure 7 shows results for workload performance in terms of speedup of the on-line model workloads over the static model workloads. Results greater than 1 mean the on-line controller yielded better workload performance

than the static controller. On average, across 30 workloads, the two controllers have workload performance within 0.5% of each other. This shows that the self-tuning controller can achieve workload performance similar to the fixed parameter controller.

Our results show that we have succeeded in automatically finding controller parameters that achieve acceptable settling time and yield application performance similar to a controller using tuned static parameters. The important point is that the on-line modeling saves the vendor from doing exhaustive testing to find and validate the right static parameters and is expected to deal better with unforeseen workloads or configurations.

6. Conclusion

We developed a self-tuning piece-wise power capping controller that accounts for different behavior of clock modulation and DVFS actuators found in servers. It works by combining a P-controller and an ad-hoc controller in a novel way to adjust controller parameters as the system changes behavior. The ad-hoc controller acts as a temporary backup for the P-controller and provides new measurement data to form a new power model for the P-controller to use. The controller settles within 1 second and achieves application performance within 0.5% of a controller with tuned static parameters, which is acceptable for the scenarios under which power capping is used. Most importantly, the controller saves the vendor time in finding good, fixed controller parameters.

7. References

[1] C. Lefurgy et al., "Power capping: a prelude to power shifting", *Cluster Computing*, Springer Netherlands, 2007.

[2] Z. Wang et al., "Feedback Control Algorithms for Power Management of Servers", *Third International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks* (FeBID), 2008.

[3] R. Raghavendra et al., "No Power Struggles: A Unified Multi-level Power Management Architecture for the Data Center", *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS), 2008.

[4] P. Ainsworth et al., *Going Green with IBM Systems Director Active Energy Manager*, IBM, 2008.

[5] Hewlett-Packard Development Company, *Dynamic Power Capping TCO and Best Practices White Paper*, 2008.

[6] Intel, *Dynamic Power Optimization for Higher Server Density Racks – A Baidu Case Study with Intel Dynamic Power Technology*, 2008.

[7] S. Rivoire et al., "A Comparison of High-Level Full-system Power Models", Workshop on Power Aware Computing and Systems (HotPower), 2008.

[8] T. Brey et al., "BladeCenter Chassis Management", *IBM J. Res. & Dev.*, vol. 49, no. 6, November, 2005.

[9] W. Felter et al., "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems", Proceedings of the International Conf. on Supercomputing, 2005.

[10] X. Wang and M. Chen, "Cluster-level Feedback Power Control for Performance Optimization", *IEEE Intl. Symp. on High-Performance Computer Architecture* (HPCA), 2008.

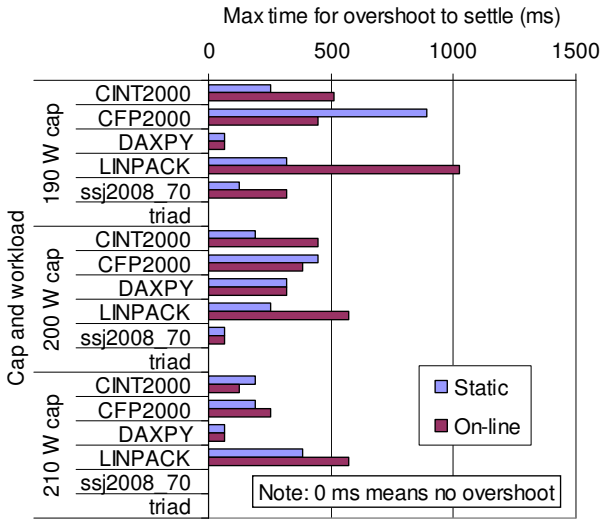

Figure 5: Settling time for Server X using on-line modeling with DVFS and throttling actuators

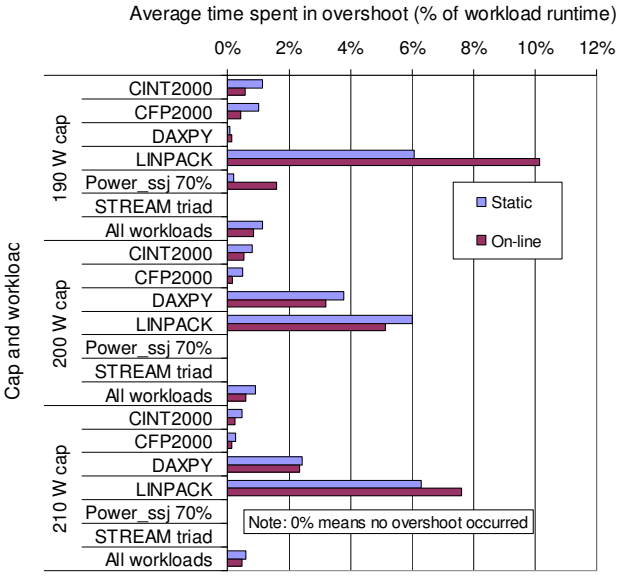

Figure 6: Average time spent in overshoot episodes

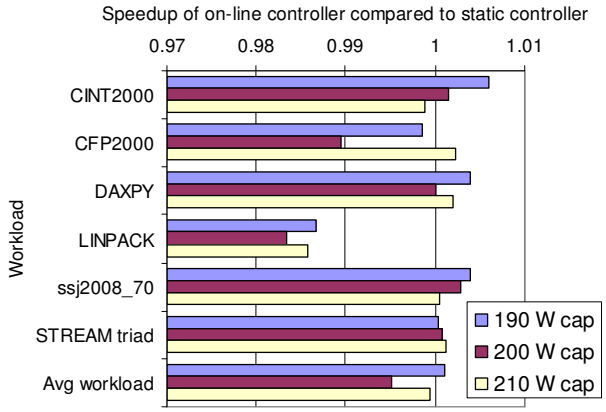

Figure 7: Speedup of on-line model compared to static model