Dynamic Voltage Scheduling Using Adaptive Filtering of Workload Traces

Amit Sinha and Anantha P. Chandrakasan
Department of EECS
Massachusetts Institute of Technology
Cambridge, MA 02139

e-mail: sinha@mit.edu, anantha@mtl.mit.edu

Abstract *- An adaptive approach for dynamic voltage scheduling on processors is presented based on workload prediction by filtering a trace history. The effects of update frequency and filtering strategy on the energy savings is analyzed. A performance hit metric is defined and techniques to minimize energy under a given performance requirement are outlined. Our results demonstrate that up to two orders of magnitude energy savings is possible with dynamic voltage scheduling depending on workload statistics.*

I. Introduction

Dynamic Voltage Scheduling (DVS) is a very effective technique for reducing CPU energy [1][2]. Most microprocessor systems are characterized by a time varying computational load. Simply reducing the operating frequency during periods of reduced activity results in linear decrease in power consumption but does not affect the total energy consumed per task. Reducing the operating voltage implies greater critical path delays which in turn means that the peak performance is compromised. Significant energy benefits can be achieved by recognizing that peak performance is not always required and therefore the operating voltage and frequency of the processor can be dynamically adapted based on instantaneous processing requirement. Figure 1 shows a 1 minute snapshot of the workload trace for three processors being used for three different types of applications: (i) a dialup server (characterized by numerous users logging in and out independently), (ii) a workstation (characterized by an interactive single user) and (iii) a UNIX file server (characterized by intermittent requests). The varying workload requirements are at once apparent.

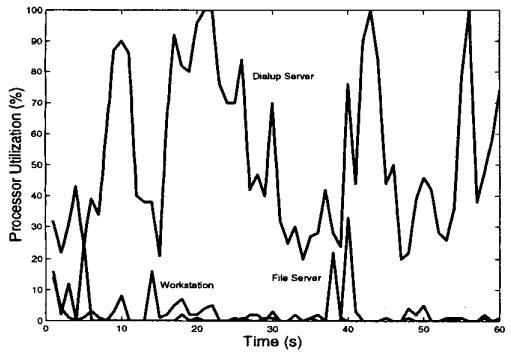

Fig. 1. 60 second workload trace for three processors

The goal of DVS is to adapt the power supply and operating frequency to match the workload such that the visible performance loss is negligible. The crux of the problem lies in the fact that future workloads are often non-deterministic. The rate at which DVS is done also has a significant bearing on performance and energy. A low update rate implies greater workload averaging which results in lower energy (as we will show). The update energy and performance cost is also amortized over a larger time frame. On the other hand a low update rate also implies a greater performance hit since the system will not respond to a sudden increase in workload. In this paper we propose a workload prediction strategy based on adaptive filtering of the past workload profile. Several filtering schemes are analyzed. We also define a performance hit metric which is used to judge the efficacy of the schemes. The evaluation of some DVS algorithms on portable benchmarks was done in [3] and [4]. In [5] the scheduling of hard real-time tasks on variable voltage processors is presented. Our approach is more general and illustrates important tradeoffs between energy savings and performance.

The energy savings from DVS can be substantial depending on workload statistics. Although this is very desirable in portable battery constrained systems, energy savings is becoming equally important in desktops and servers too. A server farm with 10000 units each equipped with 100W processors would require 1MW power!

II. Variable Voltage Processing

A. Energy Workload Model

Using simple first order CMOS delay models it has been shown in [1] that the energy consumption per sample is given by

$$E(r) = CV_0^2 T_s f_{ref} r\left[\frac{V_t}{V_0} + \frac{r}{2} + \sqrt{r\frac{V_t}{V_0} + \left(\frac{r}{2}\right)^2}\right]^2 \qquad (1)$$

where C is the average switched capacitance per cycle, T_s is the sample period, f_{ref} is the operating frequency at V_{ref}, r is the normalized processing rate i.e. $r = f / f_{ref}$ and $V_0 = (V_{ref} - V_t)^2/V_{ref}$ with V_t being the threshold voltage. The normalized workload in a system is equivalent to the processor utilization. The operating system scheduler allocates a time-slice and resources to various processes based on their priorities and state. Often no process is ready to run and the processor simply idles. The normalized workload, w, over an interval is simply the ratio of the non-idle cycles to the total cycles, i.e. w = (*total_cycles - idle_cycles*) / *total_cycles*. The workload is

0-7695-0831-6/00 $10.00 © 2000 IEEE

always in reference to the fixed maximum supply and maximum processing rate. In an ideal DVS system the processing rate is matched to the workload so that there are no idle cycles and utilization is maximum. Figure 2(a) shows the plot of normalized energy versus workload as described by Equation 1, for an ideal DVS system. The important conclusions from the graph are, (i) Averaging the workload and processing at the mean workload is more energy efficient because of the convexity of the $E(r)$ graph and Jensen's inequality [6]: $\overline{E(r)} \geq E(\bar{r})$. (ii) A small number of discrete processing rate levels (i.e supply voltage, V_{dd}, and operating frequency, f) can give energy savings very close to the savings obtained from arbitrary precision DVS.

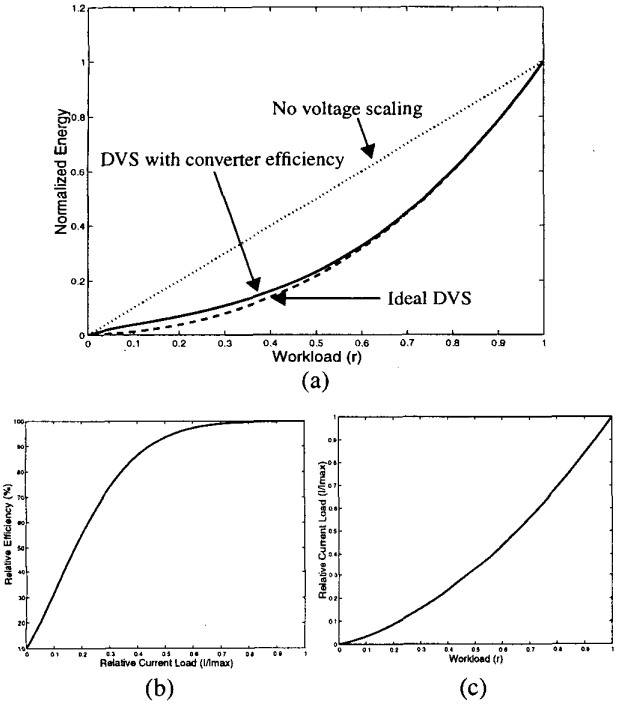

Fig. 2. (a) Energy vs. workload ,(b) Typical DC/DC converter efficiency profile, and (b) Current vs. workload

B. Variable Power Supply

A variable power supply can be generated using a DC/DC converter which takes a fixed supply and can generate a variable voltage output based on a pulse-width modulated signal. It essentially consists of a power switch and a second order LC filter and is characterized by an efficiency which drops off as the load decreases as shown in Figure 2(b). At a lower current load, most of the power drawn from the supply gets dissipated in the switch and therefore the energy gains from DVS are proportionately reduced. Using a technique similar to the one used in the derivation of Equation 1, the a first order current consumption equation can be expressed as

$$I(r) = I_{ref} r \frac{V_0}{V_{ref}} \left[\frac{V_t}{V_0} + \frac{r}{2} + \sqrt{r \frac{V_t}{V_0} + \left(\frac{r}{2}\right)^2} \right] \qquad (2)$$

where I_{ref} is the current drawn at V_{ref}. Using the DC/DC converter efficiency graph and the relative load current $I(r)$, we can predict the efficiency, $\eta(r)$. Figure 2(a) also shows the $E(r)$ curve after incorporating the efficiency of the DC/DC converter as shown in Figure 2(b) while Figure 2(c) shows the relative current consumption as a function of the workload (again assuming an ideal DVS system with $w = r$) as predicted by Equation 2. Efficient converter design strategies have been explored in [7].

III. Workload Prediction

A. System Model

Figure 3 shows a generic block diagram of the variable voltage processing system. The 'Task Queue' models the various events sources for the processor e.g. I/O, disk drives, network links, internal interrupts, etc. Each of the n sources produce events at an average rate of λ_k, ($k = 1, 2, .., n$). An operating system scheduler manages all these tasks and decides which process gets to run on the processor. The average rate at which events arrive at the processor is $\lambda = \sum \lambda_k$. The processor in turn offers a time varying processing rate $\mu(r)$. The operating system kernel measures the idle cycles and computes the normalized workload w over some observation frame. The workload monitor sets the processing rate, r, based on the current workload, w, and a history or workloads from previous observation frames. This rate r in turn decides the operating voltage $V(r)$ and operating frequency $f(r)$ which is set for the next observation slot. The problems that we address in this paper are: (i) What kind of future workload prediction strategy should be used? (ii) What is the duration of the observation slot i.e. how frequently should the processing rate be updated? The overall objective of a DVS system is to minimize energy consumption under a given performance requirement constraint.

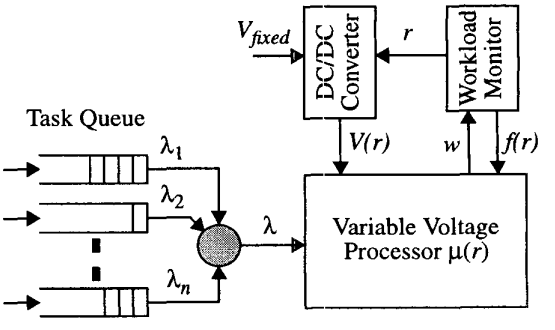

Fig. 3. Block diagram of a DVS processor system

B. Markov Processes

A stochastic process is called a Markov process if its past has no influence on its future once the present is specified [8]. Consider the sequence $X[k] = aX[k-1] + n[k]$, where $n[k]$ is a white noise process. Clearly, at instance k, the process $X[k]$, does not depend on any information prior to instance k-1. The precise

definition of this limited form of historical dependency is as follows: X is an N^{th} order Markov process if its probability distribution function $P_{X[k]}$ satisfies

$$P_{X[k]}(y|X[k-1],X[k-2],...,X[0]) = P_{X[k]}(y|X[k-1],X[k-2],...,X[k-N]) \quad (3)$$

i.e. the present N values contain all the information about the past evolution of the process that is needed to determine the future distribution of the process.

Markov processes have been used in the context of Dynamic Power Management (DPM). In [9] a continuous-time, controllable Markov process model for a power managed system is introduced and DPM is formulated as a policy optimization problem. We propose to use Markov processes in the context of workload prediction i.e. we propose to predict the workload for the next observation interval based on workload statistics of the previous N intervals.

C. Prediction Algorithm

Let the observation period be T. Let $w[n]$ denote the average normalized workload in the interval $(n-1)T \le t < nT$. At time $t = nT$, we must decide what processing rate to set for the next slot, i.e. $r[n+1]$, based on the workload profile history. Our workload prediction for the $(n+1)^{th}$ interval is given by

$$w_p[n+1] = \sum_{k=0}^{N-1} h_n[k]w[n-k] \quad (4)$$

where $h_n[k]$ is an N-tap, adaptable FIR filter whose coefficients are updated in every observation interval based on the error between the processing rate (which is set using the workload prediction) and the actual value of the workload.

Most processor systems will have a discrete set of operating frequencies which implies that the processing rate levels are quantized. The StrongARM SA-1100 microprocessor, for instance, can run at 10 discrete frequencies in the range of 59MHz to 206MHz [10]. As we shall show later, discretization of the processing rate does not significantly degrade the energy savings from DVS. Let us assume that there are L discrete processing levels available such that

$$r \in R_L, R_L = \left[\frac{1}{L}, \frac{2}{L}, ..., 1\right] \quad (5)$$

where we have assumed a uniform quantization interval, $\Delta = 1/L$. We have also assumed that the minimum processing rate is $1/L$ since $r = 0$ corresponds to the complete off state. Based on the workload prediction $w_p[n+1]$, the processing rate $r[n+1]$ is set such that

$$r[n+1] = \left\lceil \frac{w[n+1]}{\Delta} \right\rceil \Delta \quad (6)$$

i.e. the processing rate is set to a level just above the predicted workload.

D. Type of Filter

We have explored four types of filters. In this section we outline the basic motivation behind each of the filters and later present results showing the prediction performance of each of the filters.

Moving Average Workload (MAW) - The simplest filter is a time-invariant moving average filter, $h_n[k] = 1/N$ for all n and k. This filter predicts the workload in the next slot as the average of the workload in the previous N slots. The basic motivation is that if the workload is truly an N^{th} order Markov process, averaging will result in workload noise being removed by low pass filtering. However, this scheme is might be too simplistic and may not work with time varying workload statistics. Also, averaging results in high-frequency workload changes being removed and as a result instantaneous performance hits are high.

Exponential Weighted Averaging (EWA) - This filter is based on the idea that effect of workload k-slots before the current slot lessens as k increases, i.e. it gives maximum weight to the previous slot, lesser weight to the one before and so on. The filter coefficients are $h_n[k] = a^{-k}$, for all n, with a chosen such that $\sum h_n[k] = 1$ and a is positive. The idea of exponential weighted averaging has been used in the prediction of idle times for dynamic power management using shutdown techniques in event driven computation [11]. There too the idea is to assign progressively decreasing importance to historical data.

Least Mean Square (LMS) - It makes more sense to have an adaptive filter whose coefficients are modified based on the prediction error. Two popular adaptive filtering algorithms are the Least-Mean-Square (LMS) and the Recursive-Least-Squares (RLS) algorithms [12]. The LMS adaptive filter is based on a stochastic gradient algorithm. Let the prediction error be $w_e[n] = w[n] - w_p[n]$, where $w_e[n]$ denotes the error and $w[n]$ denotes the actual workload as opposed to the predicted workload $w_p[n]$ from the previous slot. The filter coefficients are updated according to the following rule

$$h_{n+1}[k] = h_n[k] + \mu w_e[n]w[n-k] \quad (7)$$

where μ is the step size. Use of adaptive filters has its advantages and disadvantages. On one hand, since they are self-designing, we do not have to worry about individual traces. The filters can 'learn' from the workload history. The obvious problems involve convergence and stability. Choosing the wrong number of coefficients or an inappropriate step size may have very undesirable consequences. RLS adaptive filters differ from LMS adaptive filters in that they do not employ gradient descent. Instead they employ a clever result from linear algebra. In practice they tend to converge much faster but they have higher computational complexity.

Expected Workload State (EWS) - The last technique is based on a pure probabilistic formulation and does not involve any filtering. Let the workload too be discrete and quantized like the processing rate as shown in Equation 5 with the state 0 also included. The error can be made arbitrarily small by increasing the number of levels, L. Let $\mathbf{P} = [p_{ij}]$, $0 \le i \le L$, $0 \le j \le L$, denote a square matrix with elements p_{ij} such that $p_{ij} = Prob\{\ w[r+1] = w_j \mid w[r] = w_i\ \}$ where w_k represents the k^{th} workload level out of the $L+1$ discrete levels. Therefore $\mathbf{P}$ is the state transition matrix

with the property that $\sum_j p_{ij} = 1$. The workload is then predicted as

$$w[n+1] = \mathbf{E}\{w[n+1]\} = \sum_{j=0}^{L} w_j p_{ij} \quad (8)$$

where $w[n] = w_i$ and $\mathbf{E}\{.\}$ denotes the expected value. The probability matrix is updated in every slot by incorporating the actual state transition. In general the $(r+1)^{th}$ state can depend on the previous N states (as in a N^{th} order Markov process) and the probabilistic formulation is more elaborate.

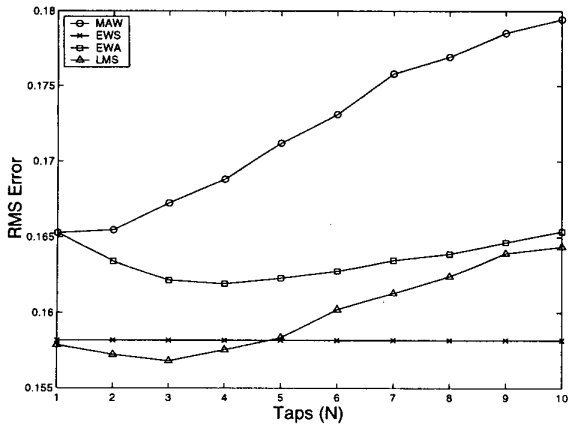

Fig. 4. Prediction performance of the different filters

Figure 4 shows the prediction performance in terms of Root-Mean-Square error for the 4 different schemes. If the number of taps is small the prediction is too noisy and if its too large there is excessive low pass filtering. Both result in poor prediction. In general we found that the LMS adaptive filter outperforms the other techniques and produces best results with $N = 3$ taps. The adaptive prediction of the filter is shown for a workload snapshot in Figure 5.

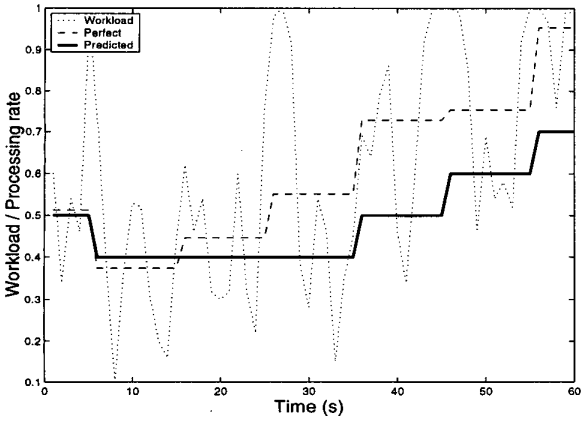

Fig. 5. Workload tracking by the LMS filter

IV. Energy Performance Tradeoffs

A. Performance Hit Function

Definition: The performance hit, $\phi(\Delta t)$, over an a time frame Δt, is defined as the extra time (expressed as a fraction of Δt) required to process the workload over time Δt at the processing rate available in that time frame.

Let $\bar{w}_{\Delta t}$ and $\bar{r}_{\Delta t}$ respectively denote the average workload and processing rates over the time frame of interest Δt. The extra number of cycles required (assuming $\bar{w}_{\Delta t} > \bar{r}_{\Delta t}$) to process the entire workload is ($\bar{w}_{\Delta t} f_{max} \Delta t - \bar{r}_{\Delta t} f_{max} \Delta t$) where f_{max} is the maximum operating frequency. Therefore the extra amount of time required is simply ($\bar{w}_{\Delta t} f_{max} \Delta t - \bar{r}_{\Delta t} f_{max} \Delta t$) / $\bar{r}_{\Delta t} f_{max}$. Therefore,

$$\phi(\Delta t) = \frac{(\bar{w}_{\Delta t} - \bar{r}_{\Delta t})}{\bar{r}_{\Delta t}} \quad (9)$$

If $\bar{w}_{\Delta t} < \bar{r}_{\Delta t}$ then the performance penalty is negative. The way to interpret this is that it is a slack or idle time. Using this basic definition of performance penalty we define two different metrics: $\phi^T_{max}(\Delta t)$ and $\phi^T_{avg}(\Delta t)$ which are respectively the maximum and average performance hits measured over Δt time slots spread over an observation period T.

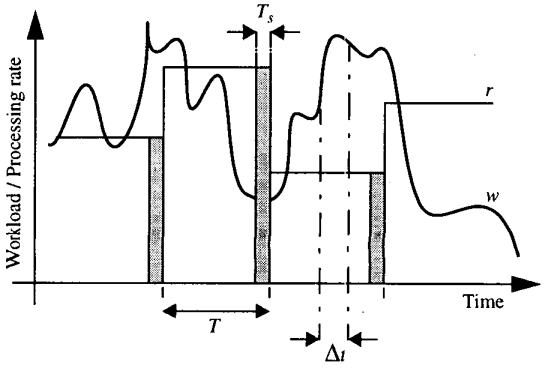

Fig. 6. Performance hit, settling time notions

Figure 7 shows the average and maximum performance hit as a function of the update time T, for a moving average prediction using $N = 2$, 6 and 10 taps. The time slots used were $\Delta t = 1$s and the workload trace was that of the dialup server. The results have been averaged over 1 hour. While the maximum performance hit increases as T increases, the average performance hit decreases. This is because as T increases the excess cycles from one time slot spills over to the next one and if the slot has a negative performance penalty (i.e. slack / idle cycles) then the average performance hit over the two slots decreases and so on. On the other hand, as T increases, the chances of an increased disparity between the workload and processing rate in a time slot is more and the maximum performance hit increases.

This leads to a fundamental energy-performance tradeoff in DVS. Because of the convexity of the $E(r)$ relationship and Jensen's inequality, we would always like to work at the overall

average workload. Therefore, over a 1 hour period for example, the most energy efficient DVS solution is one where we set the processing rate equal to the overall average workload over the 1 hour period. In other words, increasing T leads to increased energy efficiency. On the other hand, increasing T, also increases the maximum performance hit. In other words the system might be sluggish in moments of high workload. Maximum energy savings for a given performance hit involves choosing the maximum update time T such that the maximum performance hit is within bounds as shown in Figure 7.

In most DVS processors, there is a latency overhead involved in processing rate update. This is because there is a finite feedback bandwidth associated with the DC/DC converter. Normally a good voltage regulator can switch between voltage output levels in a few tens of microseconds. Changing the processor clock frequency also involves a latency overhead during which the PLL circuits lock. In general, to be on the safe side, voltage and clock frequency changes should not be done in parallel. While switching to a lower processing rate, the frequency should first be decreased and subsequently the voltage should be lowered to the appropriate value. On the contrary, switching to a higher processing rate requires the voltage to be increased first followed by the frequency update. This ensures that the voltage supply to the processor is never lower than the minimum required for the current operating frequency and avoids data corruption due to circuit failure. However, in [13] the update is done in parallel because the converter and the clock update latency are comparable (approximately 100µs) and it still works.

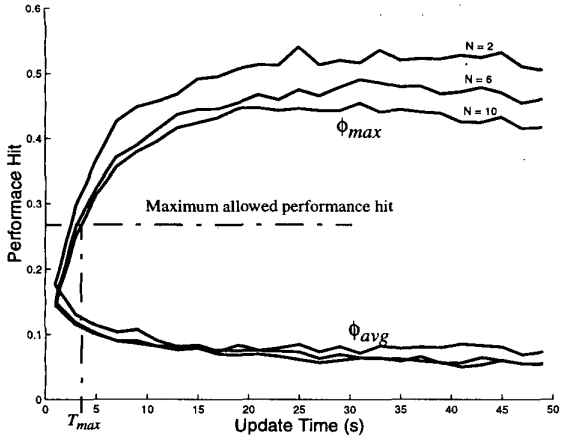

Fig. 7. Average and maximum performance hits

We denote the processing rate update latency by T_s (for settling time). It is possible to incorporate this overhead in the performance hit function. Over the update time T, the extra number of cycles is now equal to ($\bar{w}_{\Delta t} f_{max} T - \bar{r}_{\Delta t} f_{max}(T - Ts)$) and the corresponding performance hit function becomes

$$\phi(T) = \frac{\left(\bar{w}_{\Delta t} - \bar{r}_{\Delta t}\left(1 - \frac{T_s}{T}\right)\right)}{\bar{r}_{\Delta t}} \qquad (10)$$

In our experiments, the time resolution for workload measurement was 1 second. Since we want to work at averaged workload this is not a problem unless there are very stringent real-time requirements. The other advantage of using a lower time resolution is that the workload measurement subroutine does not itself add substantial overhead to the workload if the measurement duty-cycle is small. The update latency is of the order of 100µs and since this is insignificant compared to our minimum update time we have used Equation 9 instead of Equation 10.

B. Optimizing Update Time and Taps

The above conclusion that increasing the update time T results in the most energy savings is not completely true. This would be the case with a perfect prediction strategy. In reality if the update time is large, the cost of an overestimated rate is more substantial and the energy savings decrease. Since we are using discrete processing rates (in all our simulations the number of processing rate levels is set to 10 unless otherwise stated), and we round off the rate to the next higher quanta, using a larger update time results in higher overestimate cost. A similar argument holds for the number of taps N. A very small N implies that the workload prediction is very noisy and the energy cost is high because of widely fluctuating processing rates. A very large N on the other hand implies that the prediction is heavily low-passed and therefore sluggish to rapid workload changes. This leads to higher performance penalty. Figure 8 shows the relative energy plot (normalized to the no DVS case) for the dialup server trace. The period of observation was 1 hour. The energy savings showed a 13% variation based on what N and T were chosen. The filter was once again the moving average type. The implications of the above discussion is at once apparent.

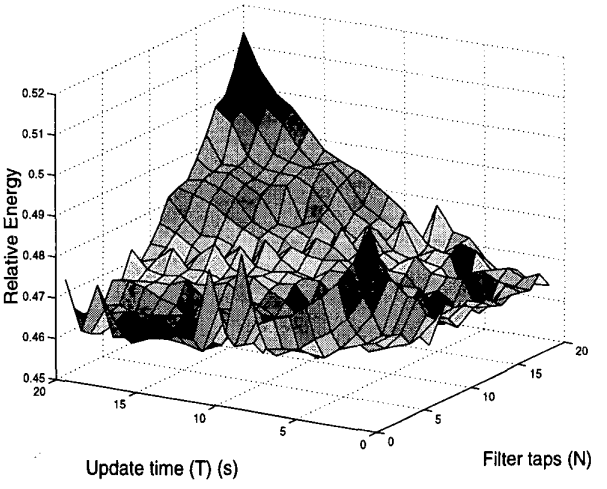

Fig. 8. Average and maximum performance hits

V. RESULTS

Table I summarizes our key results. We have used 1 hour

workload traces from three different types of machines over different times of the day. Their typical workload profiles were shown in Figure 1. The Energy Savings Ratio (ESR) is defined as the ratio of the energy consumption with no DVS to the energy consumption with DVS. Maximum savings occur when we set the processing rate equal to the average workload over the entire period. This is shown in the 'Max' column of ESR and we can see that energy savings from a factor of 2 to a few 100s is possible depending on workload statistics. Maximum savings is not possible because of two reasons: (i) The maximum performance hit increases as the averaging duration is increased, and (ii) It is impossible to know the average workload over the stipulated period *a priori*. The filters have $N = 3$ taps and an update time $T = 5$s, based on our previous discussion and experiments performed. The 'Perfect' column shows the ESR for the case where we had a perfect predictor for the next observation slot. ESR_{Max} / $ESR_{Perfect}$ reflects the factor by which energy savings is reduced because of update every T seconds. The 'Actual' column shows the ESR obtained by the various filters. In almost all our experiments the LMS filter gave the best energy savings. The last two columns are the average and maximum performance hits. The average performance hit is around 10% while the maximum performance hit is about 40%.

TABLE I : DVS ENERGY SAVINGS RATIO ($E_{NO\text{-}DVS}/E_{DVS}$)[N=3, T=5S]

Trace	Filter	Energy Savings Ratio (ESR)			ESR Comparison		ϕ_{avg} (%)	ϕ_{max} (%)
		Max	Perfect	Actual	Max / Perfect	Perfect / Actual		
Dialup Server	MAW	2.9	2.4	2.2	1.2	1.10	10.6	34.8
	EWS			2.1		1.11	10.8	36.3
	EWA			2.2		1.09	10.6	35.4
	LMS			2.3		1.03	14.7	43.1
File Server	MAW	76.7	23.5	16.7	3.3	1.41	12.6	42.8
	EWS			15.7		1.50	7.4	33.8
	EWA			16.7		1.41	9.2	37.4
	LMS			19.6		1.20	14.1	47.7
User Work Station	MAW	445.9	275.2	52.7	1.6	5.22	3.6	35.3
	EWS			59.5		4.63	3.8	35.1
	EWA			52.1		5.28	3.7	35.6
	LMS			53.0		5.19	3.9	36.0

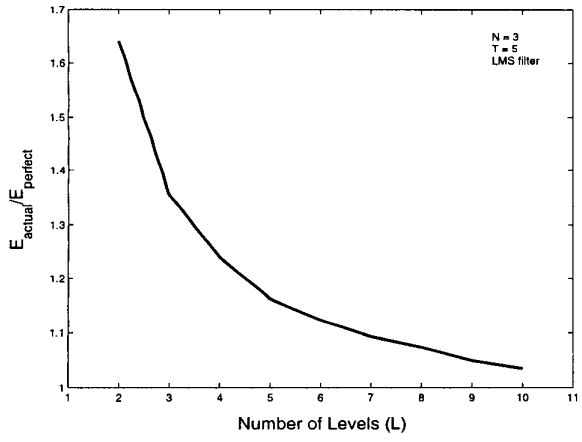

Fig. 9. Effect of number of discrete processing levels, L

Finally, the effect of processing level quantization is shown in Figure 9. As the number of discrete levels, L, is increased, the ESR gets closer to the perfect prediction case. For $L = 10$ (as available in the StrongARM SA-1100) the ESR degradation due to quantization noise is less than 10%.

VI. CONCLUSIONS

Dynamic Voltage Scaling is a very effective technique to reduce processor energy consumption without causing significant performance degradation. Up to two orders of magnitude energy savings is possible on low workload processors. Maximum energy savings occur if the processing rate is set to the overall average workload. This however is generally infeasible *a priori* and even if possible leads to high performance penalties. Frequent processing rate updates ensure that the performance penalty is limited. The faster the update rate, the lower the energy savings and the lesser the performance penalty. Workload prediction is required to set the processing rate for each update slot. Adaptive LMS filtering can be used to predict workloads. Normally a filter with 3-5 taps is good.

ACKNOWLEDGEMENTS

This research is sponsored by the Defense Advanced Research Projects Agency's (DARPA) Power Aware Computing/Communication Program and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0551. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon[1].

REFERENCES

[1] V. Gutnik and A. P. Chandrakasan, "An Embedded Power Supply for Low-Power DSP", IEEE Transactions on VLSI Systems, vol. 5, no. 4, Dec. 1997, pp. 425-435
[2] T. Burd, et. al., "A Dynamic Voltage Scaled Microprocessor System", ISSCC 2000, pp. 294-295
[3] T. Pering, T. Burd and R. Broderson, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms", International Symposium on Low Power Electronics and Design, 1998, pp. 76-81
[4] K. Govil, E. Chan and H. Wasserman, "Comparing Algorithms for Dynamic Speed Setting of a Low-Power CPU", Proc. of the First International Conference on Mobile Computing and Networking, Nov. 1995
[5] I. Hong, M. Potkonjak, M. B. Srivastava, "On-line Scheduling of Hard Real-Time Tasks on Variable Voltage Processor", ICCAD 98, pp. 653-656
[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley and Sons, 1991
[7] G. Wei and M. Horowitz, "A Low Power Switching Power Supply for Self-Clocked Systems", International Symposium on Low Power Electronics and Design, 1996, pp. 313-318
[8] R. Nelson, *Probability, Stochastic Processes and Queueing Theory*, Springer-Verlag,1995
[9] Q. Qiu and M. Pedram, "Dynamic Power Management Based on Continuous-Time Markov Decision Processes", DAC 99, New Orleans, pp.555-561
[10] http://developer.intel.com/design/strong/sa1100.htm
[11] C.-H. Hwang and A. Wu, "A Predictive System Shutdown Method for Energy Saving of Event Driven Computation", Proceedings of the International Conference on Computer Aided Design, 1997, pp. 28-32
[12] P. S. R. Diniz, *Adaptive Filtering Algorithms and Practical Implementation*, Kluwer Academic, 1997
[13] R. Min, T. Furrer and A. P. Chandrakasan, "Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks", Proceedings of the IEEE Computer Society Workshop on VLSI (WVLSI '00), April 200

1. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Project Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.