Runtime Performance Projection Model for Dynamic Power Management

Sang-Jeong Lee[1], Hae-Kag Lee[1], and Pen-Chung Yew[2]

[1] Department of Computer Science and Engineering, Soonchunhyang University
Asan, Choongnam, 336-745, Korea
{sjlee,lhk7083}@sch.ac.kr
[2] Department of Computer Science and Engineering, University of Minnesota
Minneapolis, MN 55455
yew@cs.umn.edu

Abstract. In this paper, a runtime performance projection model for dynamic power management is proposed. The model is built as a first-order linear equation using a linear regression model. It could be used to estimate performance impact from different p-states (voltage-frequency pairs). Workload behavior is monitored dynamically for a program region of 100M instructions using hardware performance monitoring counters (PMCs), and performance for the next region is estimated using the proposed model. For each 100M-instructions interval, the performance of all processor p-states is estimated and the lowest frequency is selected within specified performance constraints. The selected frequency is set with a low-overhead DVFS-based (dynamic voltage-frequency scaling) p-state changing mechanism for the next program region. We evaluate the performance degradation and the amount of energy saving of our dynamic power management scheme using the proposed projection model for SPEC CPU2000 benchmark on a Pentium M platform. We measure the execution time and energy consumption for 4 specified constraints – 10%, 20%, 40%, 80%, on the maximum allowed performance degradation. The result shows that our dynamic management scheme saves energy consumption by 3%, 18%, 38% and 48% with a performance degradation of 3%, 19%, 45% and 79% under 10%,20%,40% and 80% constraints, respectively.

Keywords: Dynamic Power Management, Dynamic Voltage-Frequency Scaling, Performance Monitoring.

1 Introduction

Power-aware computing has become a critical component of computer system design. In high-performance systems, thermal dissipation has always been a major challenge. Also, in mobile and embedded systems, energy efficiency is critical to extend battery life. Although hardware design has a direct impact on the system's power consumption, application workload is also an important contributor to power consumption. Therefore, dynamic power management considering application workload is critical to an efficient power management of a computer system [5][7][11][15].

L. Choi, Y. Paek, and S. Cho (Eds.): ACSAC 2007, LNCS 4697, pp. 186–197, 2007.
© Springer-Verlag Berlin Heidelberg 2007

Current power-aware microprocessors provide multiple operating frequency-voltage pairs for dynamic power management using dynamic voltage and frequency scaling (DVFS) techniques. DVFS can trade off system performance with power consumption. Processors that support DVFS have to balance the achieved energy savings with a pre-determined limit of performance impact on applications. Advanced Configuration and Power Interface (ACPI) is one of industrial standards that define active (p-states) and standby power management for the processors [17].

In general, the processor's power consumption is greatly dependent on its executing workload. During a program execution, performance and power consumption can vary widely according to its program behavior. Such workload characteristics can be exploited for power management. When a processor is idle waiting for its memory accesses, power consumption can be reduced by scaling down processor frequency and voltage without a significant performance loss. If we are able to find the slack points in a program region dynamically, power management decisions can be made with the help of an accessing model that estimates their effects on performance and power savings.

In this paper, a runtime performance projection model for dynamic power management is proposed. The model is used to estimate the performance impact of different p-states (voltage-frequency pairs). Workload behavior such as CPI (average cycles per instruction) and the number of memory accesses are monitored dynamically for a program region using hardware performance monitoring counters (PMCs). Performance for the next region is estimated using the proposed performance projection model with the information collected from PMCs. Performance of all processor p-states is estimated and the lowest frequency is selected within the specified performance constraints. The selected frequency is set with a low-overhead DVFS-based p-state change mechanism for the next program region.

We build a linear regression model that relates processor performance to two architectural parameters on a real Pentium-M processor. The proposed performance estimation model is a first-order linear equation that predicts performance impact on CPI by a given p-state using monitored program activities such as CPI and the number of memory accesses. We evaluate the performance degradation and energy saving using the proposed projection model for SPEC CPU2000 benchmark on a Pentium M platform. We measure the execution time and energy consumption under 4 pre-determined constraints – 10%, 20%, 40%, 80%, that specify maximum performance degradation allowed. The results show that our dynamic management scheme could save energy consumption by 3%, 18%, 38% and 48% with a performance degradation of 3%, 19%, 45% and 79%, under 10%,20%,40% and 80% constraints, respectively.

2 Related Work

There have been many studies that investigate power and energy models for power management using DVFS. Among them, Contreras et al. develop a linear power estimation model to estimate run-time CPU and memory power consumption of the Intel PXA255 processor using hardware performance counters. They derive coefficients

of the equation by minimizing the difference between estimated power consumption and actual measured power consumption using linear algebra [2].

Wu et al. propose an analytic DVFS decision model to determine new p-states in a dynamic compilation environment. They develop an equation to select a p-state using CPU execution slack due to asynchronous memory accesses [15]. Isci et al. propose a runtime phase predictor, called GPHT predictor. It monitors workload behavior for dynamic power management on Pentium-M processor [7]. After every 100M microoperations, they predict a new p-state based on Wu's DVFS decision model and apply a new DVFS setting.

Rajamani et al. developed two models based on performance counter events such as decoded instructions per cycle and data cache miss to decide a p-state for power management [11]. Their performance estimation model applies performance projection across all p-states and the p-state with the lowest frequency that meets the specified requirement is selected for next interval.

The power management scheme of Rajamani et al. is the closest to ours. However, their performance model uses both linear and non-linear equations, and they divide the workload into CPU-bound and memory-bound. Their sampling interval is 10ms. Our model is a linear model derived from a regression analysis on different performance parameters. We use a fixed instruction sampling interval that is less sensitive to clock frequency change. Previous research has shown that program phases can alter the timing and values of observed metrics [7]. Therefore, using a fixed instruction interval can eliminate the effect of timing variations.

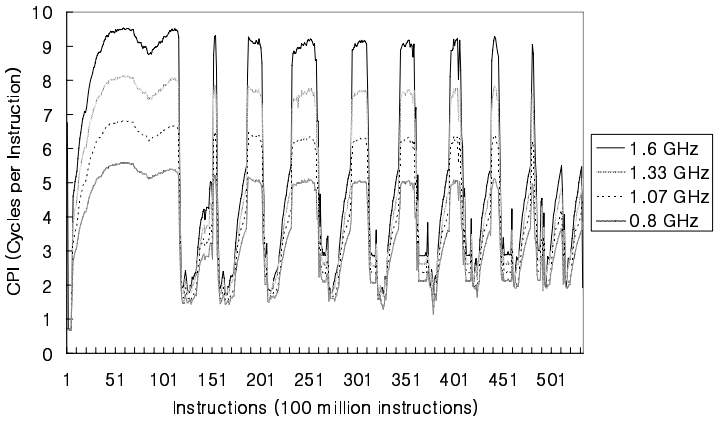
3 Runtime Performance Projection Model

3.1 Workload Behavior

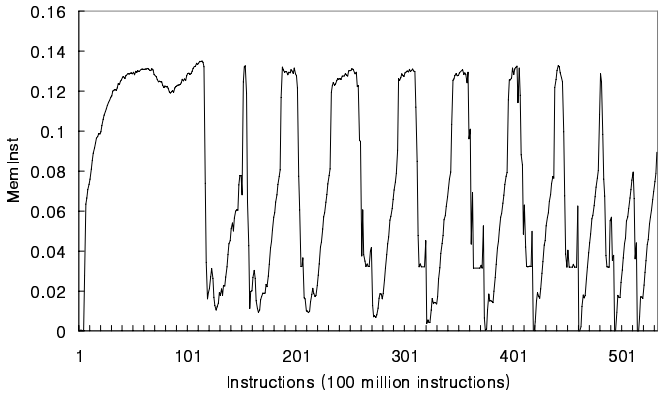
Many studies on program behavior show that programs exhibit repetitive execution phases, and their future behavior could be predicted by their monitored past behavior [6]. Figure 1 shows an example of program phase change for the SPEC CPU2000 benchmark mcf. All results are obtained using PMCs on Pentium-M processor (Model 730) in laptop computer. The Pentium M processor we experimented has 4 different DVFS-based ACPI-defined p-states (voltage-frequency pairs). Table 1 shows the four p-states of Pentium-M 730 model used in this study.

Table 1. Frequency and voltage pairs of Pentium-M (Model 730)

Frequency	Supply Voltage
1.6 GHz	1.308 V
1.33 GHz	1.212 V
1.07 GHz	1.1 V
0.8 GHz	0.988 V



(a) Performance Trace



(b) Memory access trace

Fig. 1. Performance and memory access traces for the SPEC2000 benchmark *mcf*

In Figure 1 (a), the y-axis is cycles per instruction (CPI) measured for every 100M instructions during the program execution. Figure 1 (b) shows the degree of memory accesses, MemInst. It is defined as the ratio between the memory bus transactions to the number of instructions retired. From the figure, the program has many repetitive phases, and its performance has a strong correlation to memory accesses. Figure 2 shows normalized performance and energy consumption across four p-states for mcf. Energy consumption is estimated using power equation described in section 5. All values are normalized toward those obtained using 1.6 GHz. For mcf, the performance has a strong impact on energy consumption across all p-states.

Considering the above workload behavior, we use a linear equation to model program performance for each p-state. Its coefficients will be obtained by a regression model detailed in the next section.

$$CPI_{f'} = \beta_0 + \beta_1 CPI_f + \beta_2 MemInst_f \quad (1)$$

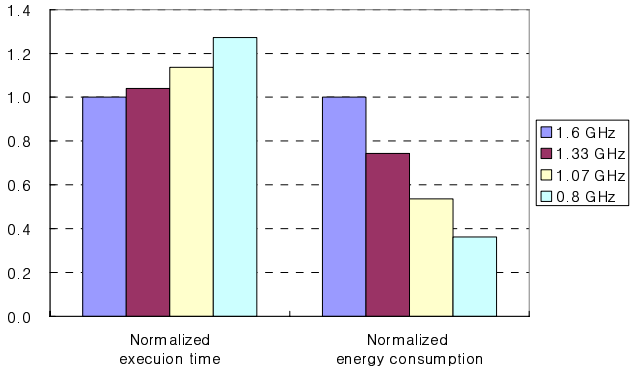


Fig. 2. Normalized performance and energy consumption across four p-states for *mcf*

In the equation (1), f and f' denote the two different frequencies of two different p-states. $CPI_{f'}$ is the estimated CPI at the p-state of f', and CPI_f is the measured CPI at the p-state of f. β_0, β_1, β_2 are regression coefficients. These coefficients indicate the relative significance of the corresponding terms obtained by the regression analysis.

3.2 Linear Regression Model

A regression model is a compact mathematical representation of the relationship between the response variable and the independent variables in a given design space [13]. Linear regression models are widely used to obtain predictions of the response variable at arbitrary points in the design space. Linear regression is represented as follows:

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ji} + e_i \tag{2}$$

where y_i is the i^{th} observation of the response variable which depends on the independent variables $x_1, x_2, \ldots, x_k$. The β_j are the coefficient of the variable x_j and an estimation of the value can provide the useful relationship. The e_i is the error term representing the deviation of the i^{th} observation value (i.e., y_i) from the estimated value by the given linear equation.

A very elegant method for estimating β_j is the method of least squares. This method of estimation, which leads to estimates of certain optimal properties, is based on the appealing idea of choosing β_j to minimize the squares of the error term. That is, determining β_j that minimizes MSE (Mean Squares Error) is one of our goals.

$$MSE = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{k} \beta_j x_{ji})^2 \tag{3}$$

If the relationship equation could be derived, very useful information could also be obtained by using the equation. In this paper, CPI under a certain p-state is predicted by applying the values of CPI and MemInst under a different p-state using equation (1).

Another problem considered in the regression analysis is that of statistical testing for the null hypothesis $H_0 : \beta_j = 0$ (j=1,2,…, k). The F-test is a standard statistical method for testing the regression model, in which the total variation (SST) is decomposed into two terms: the variations due to linear regression (SSR) and the regression error (SSE). The definitions of SST, SSR, and SSE are

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2, \quad SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2, \quad SSE = SST - SSR \tag{4}$$

where $\bar{y}$ is the mean of the observed values of response variables and $\hat{y}_i$ are the predicted values by the regression equation. The sample coefficient of determination R^2 can be calculated by

$$R^2 = \frac{SSR}{SST}, \quad (0 \le R^2 \le 1) \tag{5}$$

R^2 provides the multiple correlation statistic, so the larger the value of R^2 and the smaller the value SSE the better the fit of the predicted value to the observations. F-statistic F_0 for the hypothesis test is defined as

$$F_0 = \frac{SSR}{k} \Big/ \frac{SSE}{n-k-1} \tag{6}$$

If the F-statistic value is lager than the given significance level which is referred to the F-distribution, the null hypothesis $H_0 : \beta_j = 0$ is rejected, in other words, the regression equation has significant meanings.

3.3 Linear Performance Model

We derive coefficients, β_j, for equation (1) by the above regression analysis. The experimental data were obtained from the execution of SPEC CPU2000 benchmark

Table 2. Estimated regression coefficients for each p-state

(a) Response variable $CPI_{1.6GHZ}$ case

f	β_0	β_1	β_2	R^2	F_0
1.3GHz	-0.16	1.01	9.85	0.99	2864261
1.07GHz	-0.02	1.00	20.39	0.98	1272807
0.8 GHz	-0.06	1.02	29.08	0.98	1736878

(b) Response variable $CPI_{1.3GHZ}$ case

f	β_0	β_1	β_2	R^2	F_0
1.3 GHz	0.04	0.96	-8.16	0.99	2417377
1.07GHz	-0.01	0.99	10.30	0.98	1978693
0.8GHz	-0.04	1.02	18.92	0.99	3343031

(c) Response variable $CPI_{1.07GHZ}$ case

f	β_0	β_1	β_2	R^2	F_0
1.6GHz	0.08	0.94	-17.31	0.97	910968
1.3GHz	0.04	0.98	-9.53	0.98	1769944
0.8GHz	-0.02	1.01	8.88	0.98	1707008

(d) Response Variable $CPI_{0.8GHZ}$ case

f	β_0	β_1	β_2	R^2	F_0
1.6 GHz	0.11	0.92	-25.39	0.96	835010
1.3 GHz	0.06	0.96	-17.91	0.99	2272772
1.07 GHz	0.04	0.97	-8.35	0.98	1536522

using the four p-states on Pentium-M processor. The values of each variable are sampled at every 100M instructions interval. And then, we derived the relationships between the CPI values of two different p-states including MemInst. For each pair (f,f'), the coefficients β_j are determined through the linear regression analysis. Table 2 shows the results of the regression analysis. The coefficients are obtained for the performance prediction of each p-state in equation (1). For example, Table 2 (a) shows the coefficients to estimate $CPI_{1.6GHZ}$ (f' = 1.6 GHz) from CPI_f and $MemInst_f$ when f=1.33 GHz, 1.07 GHz, 0.8 GHz in equation (1). Table 2, also, shows the values of R^2 and F_0. For all cases they are sufficiently large, so we can conclude that the null hypothesis $H_0 : \beta_j = 0$ is rejected, that is, the regression equation has significant meanings. Figure 3 shows the plotting of pairs between the observed values and the predicted values. It shows a very good fit of the model to real observed experimental data.

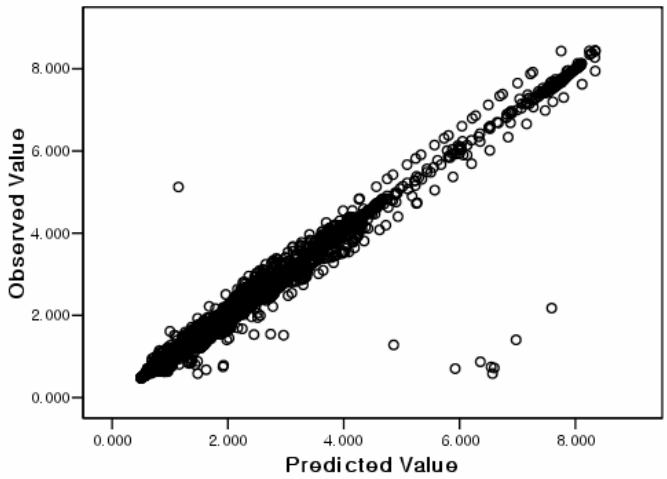


Fig. 3. A plotting of pairs (observed value, predicted value)

4 Dynamic Power Management

Using the linear regression model for performance estimation, we designed a dynamic power management framework on a Pentium-M processor (Model 730) using an off-the-shelf laptop computer (Toshiba Satellite A80) running Linux kernel 2.6-19. Workload behavior is monitored with PMCs dynamically for a program region of every 100M instructions. To eliminate the effect of timing variations across a p-state change during monitoring, we monitor workload behavior at a fixed interval of 100M instructions. After 100M instructions a performance monitoring interrupt (PMI) handler is invoked. The PMI handler is implemented as a loadable kernel module on Linux kernel. The PMI monitors application execution through two PMCs for retired instructions (INST_RETIRED event), memory bus transaction (BUS_TRAN_MEM event) and a time stamp counter (TSC) for clocks on the Pentium-M processor.

Figure 4 shows our dynamic power management framework. From the monitored values, CPI (the ratio of clock cycles to instructions retired) and MemInst (the ratio of memory bus transactions to instructions retired) are calculated. Performance impact is then estimated using our projection model, equation (1), with the calculated parameters to determine a p-state for the next interval. CPIs of all the p-states except current p-state are estimated. After getting CPIs, the expected run times for all p-states are calculated. We choose the p-state with the lowest frequency within the specified performance constraint by comparing their estimated run times. Then, the new p-state with the selected frequency is applied to next interval. Before exiting the PMI handler, the PMCs and a TSC are reinitialized and the counters are restarted.

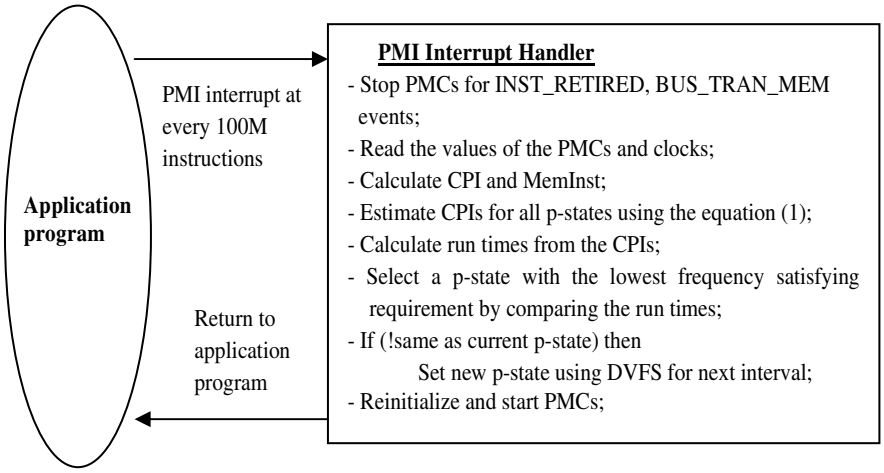


Fig. 4. Our dynamic power management framework with the runtime performance projection model

5 Evaluation Results

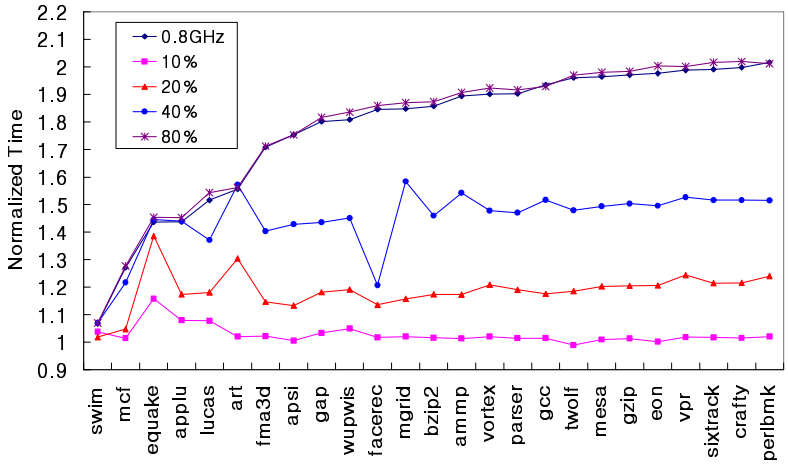
For evaluation, we use SPEC CPU2000 benchmark (178.galgel benchmark is excluded because of compile error). We measured execution time and computed energy consumption for each benchmark program. We could not measure actual energy used but calculated the energy consumption instead using the following equation for CMOS circuit.

$$E = PT = CV^2 fT \tag{7}$$

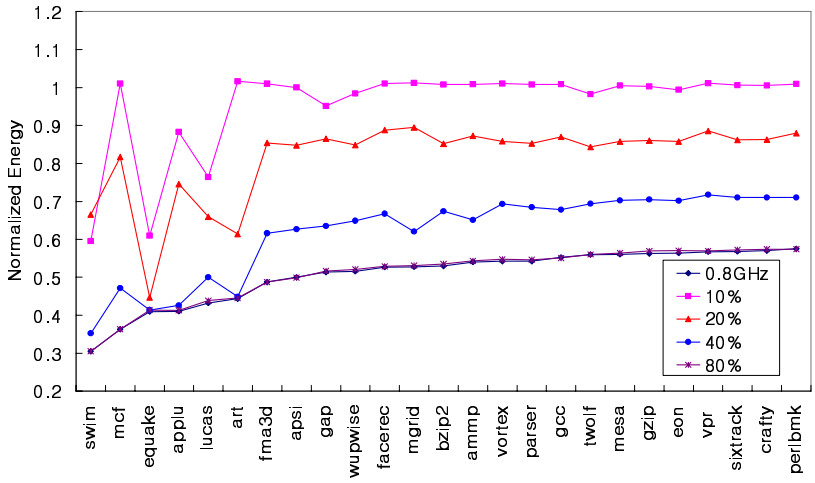
where P is the dynamic power, C is the switched capacitance, V is the supply voltage, f is the clock frequency and T is the total execution time of the program. Let's assume the program is executed on the processor with m p-states and DVFS is applied for dynamic power management. We define

$$E = \sum_{i=1}^{m} E_i = \sum_{i=1}^{m} (C_i f_i V_i^2 T_i) \approx C \sum_{i=1}^{m} f_i V_i^2 T_i \tag{8}$$

where E_i is the total energy consumption using i^{th} p-state, and C_i, f_i, V_i and T_i are switched capacitance, clock frequency, supply voltage and execution time for the i^{th} p-state. We assume all the switched capacitances are the same. By measuring program execution for each p-state, we can get their relative energy consumptions.



(a) Normalized execution time



(b) Normalized energy consumption

Fig. 5. Execution time and energy consumption normalized to 1.6GHz result for SPEC CPU2000 benchmark

Figure 5 shows the results of execution time and energy consumption normalized to the values at 1.6 GHz which is the maximum frequency allowed. It also shows the results at 0.8 GHz, the minimum-frequency among all p-states, and our power

management under the performance constraints of 10%, 20%,40% and 80%, respectively. Here, 10% performance constraint means the performance degradation should be less than 10% comparing to the maximum performance at 1.6GHz. *Swim, mcf, equake, applu, lucas, art, fma3d* on the left are memory-bound applications (MemInst are above 0.01). They have the least performance loss and most energy saving using dynamic power management. The benchmarks on the right are CPU-bound applications. They have the least energy saving with most performance degradation. In Figure 5 (a), on average, the values of normalized time are 1.03, 1.19, 1.45 and 1.79 for 10%, 20%,40% and 80% performance constraints, respectively. There is only one violation at the 40% performance constraint. Almost all benchmarks in Figure 5 (a), except at 20% in which it's results are close to the constraint, are well within their required tolerance. In the 10% case, our performance model estimates the performance too conservatively. For the 40% and 80% cases, memory-bound applications satisfy the requirements well but CPU-bound applications violate the constraints. We think this discrepancy is due to the error caused by four discrete p-states and the error from the CPI term in equation (1). But, in general, our performance projection model estimates performance well. The execution time for the 80% case is 1% greater than the minimum-frequency 0.8GHz because of the overhead of the PMI handler and DVFS overhead. Figure 5 (b) shows the energy consumption normalized to 1.6 GHz. The smaller value means more energy saving. Memory-bound applications show much energy saving with less performance degradation. On average, normalized energy consumption are 0.97,0.82,0.62,0.52 for 10%,20%, 40%,80% constraint, respectively.

Figure 6 shows average execution time and energy consumption normalized to 1.6GHz including the GPHT predictor. GPHT is a dynamic power management framework with a phase predictor that has a similar structure as a branch predictor [7].

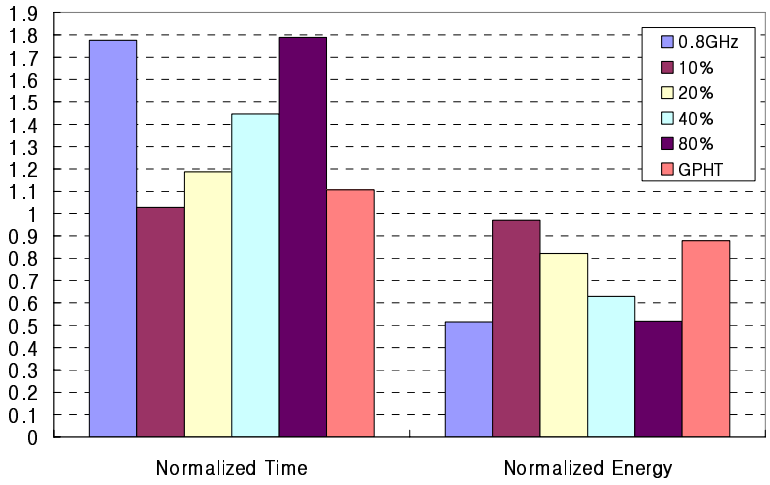


Fig. 6. Average execution time and energy consumption normalized to 1.6GHz including GPHT

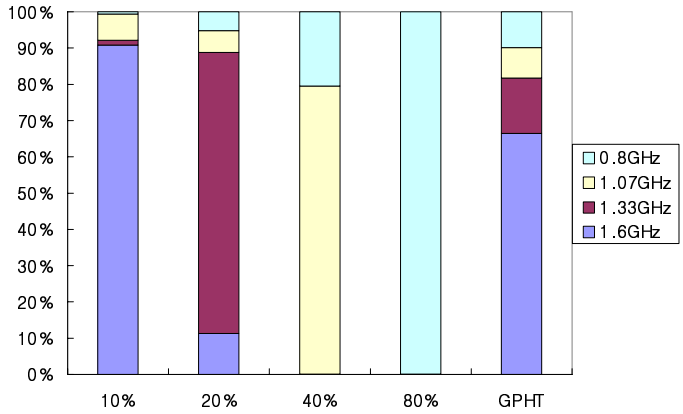


Fig. 7. Average breakdown of p-states for SPEC CPU2000 benchmark

We implemented it using the same value of Mem/micro-op (the ratio of memory bus transactions to micro-operations retired) to classify phases. It has no explicit performance requirement. The result shows that average normalized time is 1.11 and energy consumption is 0.88. GPHT shows better results comparing to the case of our 10% performance requirement. However, its energy saving is limited for less aggressive power management. Our power management for the 20% performance requirement saves much more energy. Figure 7 shows the average breakdown of p-states used during the execution for SPEC CPU2000 benchmark.

6 Conclusion

This paper presents a runtime performance projection model for dynamic power management. This model is used to predict performance impact of a p-state (voltage-frequency pairs). Using linear regression analysis, a first-order linear equation is built to estimate performance impact from monitored activity information such as CPI (cycles per instruction) and the number of memory accesses. We develop a dynamic power management framework using the performance projection model. It monitors and estimates workload behavior through hardware performance monitors sampled at every 100M instructions on Pentium-M processor. At each sample it estimates the performance of all allowable p-states. The lowest frequency that still meets a pre-determined level of tolerance in performance degradation is selected for next execution interval. We experiment our framework with SPEC CPU2000 benchmark. The result shows that our dynamic management saves energy consumption by 3%, 18%, 38% and 48% with a performance degradation of 3%, 19%, 45% and 79% under the pre-determined performance degradation tolerance of 10%,20%,40% and 80%, respectively. When we compare it with the GPHT power management scheme [7] that has 12% energy saving with 11% performance loss, our management framework can provide more energy saving.

References

1. Choi, K., Cheng, W., Pedram, M.: Frame-based Dynamic Voltage and Frequency Scaling for an MPEG Player. Journal of Low Power Electronics, American Scientific Publishers 1(1), 27–43 (2005)
2. Contreras, G., Martonosi, M.: Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events. In: International Symposium on Low Power Electronics and Design (ISLPED'05) (August 2005)
3. Intel Corp.: IA-32 Intel Architecture Software Developer's Manua: vol. 3 System Programming Guide (2005)
4. Intel Corp.: Enhanced Intel SpeedStep ® Technology and Demand-Based Switching on Linux, http://www.intel.com/cd/ids/developer/asmo-na/eng/195910.htm?prn=Y
5. Isci, C., Martonosi, M., Buyuktosunoglu, A.: Long-term Workload Phases: Duration Predictions and Applications to DVFS. IEEE MICRO. 25 (2005)
6. Isci, C., Martonosi, M.: Phase Characterization for Power: Evaluating Control-Flow-Based and Event-Counter-Based Techniques. In: Proceedings of 12th International Symposium on High-Performance Computer Architecture (HPCA-12) (Feburary 2006)
7. Isci, C., Contreras, G., Martonosi, M.: Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. In: Proceedings of the 39th International Symposium on Microarchitecture (MICRO-39) (December 2006)
8. Nienhuser, D.: Power Management Guide, http://www.gentoo.org/doc/en/power-management-guide.xml
9. Poellabauer, C., Zhang, T., Pande, S., Schwan, K.: An Efficient Frequency Scaling Approach for Energy-Aware Embedded Real-Time Systems. In: Beigl, M., Lukowicz, P. (eds.) ARCS 2005. LNCS, vol. 3432, Springer, Heidelberg (2005)
10. Rajamani, K., Hanson, H., Rubio, J., Ghiasi, S., Rawson, F.: Online Power and Performance Estimation for Dynamic Power Management. IBM Technical Report RC24007, IBM Research (July 2006)
11. Rajamani, K., Hanson, H., Rubio, J., Ghiasi, S., Rawson, F.: Application-Aware Power Management. In: IEEE International Symposium on Workload Characterization (IISWC-2006), October 2006, IEEE Computer Society Press, Los Alamitos (2006)
12. Sazeides, Y., Kumar, R., Tullsen, D., Constantinou, T.: The Danger of Interval-Based Power Efficiency Metrics: When Worst Is Best. IEEE Computer Architecture Letters 4 (January 2005)
13. Seber, G., Lee, A.: Linear Regression Analysis. Wiley-Interscience, Chichester (2003)
14. Sprunt, B.: Pentium 4 Performance-Monitoring Features. IEEE MICRO 22(4) (2002)
15. Wu, Q., Reddi, V., Lee, J., Connors, D., Brooks, D., Martonosi, M., Clark, D.: Dynamic Compilation Framework for Controlling Microprocessor Energy and Performance. In: Proceedings of the 38th International Symposium on Microarchitecture (MICRO-38), November 2005 (2005)
16. Zhu, Z., Zhang, X.: Look-ahead Architecture Adaptation to Reduce Processor Power Consumption. IEEE MICRO 25(4) (2005)
17. Advanced Configuration and Power Interface Specification 3.0, http://www.acpi.info