Energy Savings in Processor Based on Prediction Technique

Dinh-Mao Bui*, Thien Huynh-The*, Sungyoung Lee*, YongIk Yoon†, SungIk Jun‡
*Computer Engineering Department, Kyung Hee University, Suwon, Korea
{mao, thienht, sylee}@oslab.khu.ac.kr
†Department of Multimedia Science, SookMyung Women's University, Seoul, Korea
{yiyoon}@sm.ac.kr
‡ HPC system research section/Cloud computing department/ETRI, Daejeon, Korea
{sijun}@etri.re.kr

***Abstract*—Green computing has become one of the hottest trends in recent years. In this research area, the major purpose is to reduce the energy consumption as well as the CO_2 emission. Obviously, this topic has been the important issue in the field of electronic and computer engineering. In fact, energy factor might be considered to be a significant cost when running any computing system. Basically, energy savings can be obtained in many parts of the system including memory, peripheral devices, hard disk drive and processor. In processor or CPU level, there is a number of solutions to handle the power consumption. However, most of them based on reactive model which engages the thresholds. Obviously, these techniques are not accuracy and limited to save the power. In this research, a proactive solution based on prediction technique is proposed. Firstly, the utilization of each core of processor is anticipated by using Gaussian process regression. Subsequently, a migration mechanism can be used to migrate the system-level processes between these cores. Finally, the idle cores can be turned off to save the power while still maintaining an acceptable performance.**

Keywords*—Processor core, monitoring, Gaussian process regression, energy efficiency, CPU utilization.

I. Introduction

Recently, the investment on energy efficiency has been increased at a stable rate. Most of the budget has been spent on improved hardware as an unavoidable cost. Consequently, this trend leads to the rapid growth in operational cost for the computing system. In computer system, energy is used to maintain the operations of all the components. Even when the system is idle, the energy is still consumed to keep the system alive. Because of that, saving energy solutions mostly concern reducing the computing units that consume the power, especially in the processor. Nevertheless, this effort is very challenge because besides energy efficiency issue, another problem is to maintain the workload at an acceptable rate.

To solve above problems, there are reactive solutions such as Dynamic Voltage and Frequency Scaling (DVFS), power capping, or CPU idle power savings states. However, these solutions have a drawback of inaccuracy or expensive to apply massively in large scale system. Motivated by that reason, we would like to propose a proactive solution to achieve the energy efficiency in processor level. Based on prediction technique, the energy-aware component anticipates the utilization of processor cores. Subsequently, the process migration technique is engaged to migrate the process between these cores in order to increase the number of empty cores. Lastly, these empty cores are turned off to achieve the overall energy efficiency. For verification the effectiveness of our idea, an application is implemented for evaluation purpose.

This paper is organized as follows. In Section 2, some related works are included to show the state of the art in energy efficiency for processor. The proposed approach is included in detail in Section 3. In Section 4, we conduct the performance evaluation for the energy saving application. Our conclusion is summarized in Section 5.

II. Related Works

A. Dynamic Voltage and Frequency Scaling

Dynamic voltage and frequency scaling (DVFS) [1] [2] is technique to automatically change the frequency and voltage 'on the fly'. The purpose of this technique is to save the power or to mitigate the heat of the processor. Less heat generated allows the cooling system to be under-performed or turned off. This leads to the noise reduction and even better energy saving. The main drawback of this technique is to reduce the performance proportionally to the saved energy. Because of this reason, DVFS usually used in laptops and hand-held devices which prefer the long-lasting battery rather than the power of computation [3]. Besides, this technique is also used in the system which prefers low heat emission. For High Performance Computing system or computation intensive system, DVFS might be not a suitable solution.

The DVFS technique can be implemented either in hardware and software levels. For example, Intel and AMD applied DVFS in their hardware throttling technologies, namely SpeedStep and Cool'n'Quite respectively. On software level, the Linux OS governors can control the frequency of processor *via* ACPI interfaces.

B. Power capping

Power capping [4] or Power budgeting is usually used in the power-limited data centers. This solution enables the capability of controlling the power budget on system-level or rack-level. In fact, power capping is considered to be a key factor for engaging power shifting, which is the technique to individually allocate the power to each server in the cluster to achieve the global constraint of using power. According to the original design of this idea, the high priority servers might receive more power than the lower priority ones. Because power capping can be done out-of-band, the failure of Operating System does not affect the functionality of the power management. The implementation of power capping can be found in HP Intelligent Power Discovery and IBM Systems Director.

C. *CPU idle power savings states*

Most of the recent processors are equipped with power saving modes, namely C-states [5]. This configurations enable the capability for idle processor to turn off unused components in order to save the power. Although C-states existed in laptops for lengthening the battery life for a while, they are recently used in the server because of the disadvantage of deep sleep. This issue happens when the processor sticks in some energy saving states and yet requires for a long period to wake up. Clearly, the problem of deep sleep states makes the server unable to fulfill the requirement of some computation intensive tasks. Another issue of C-states is that this mechanism depends on the functionality of the operating system kernel which sometimes fails to provide suitable instruction.

III. Proposed Method

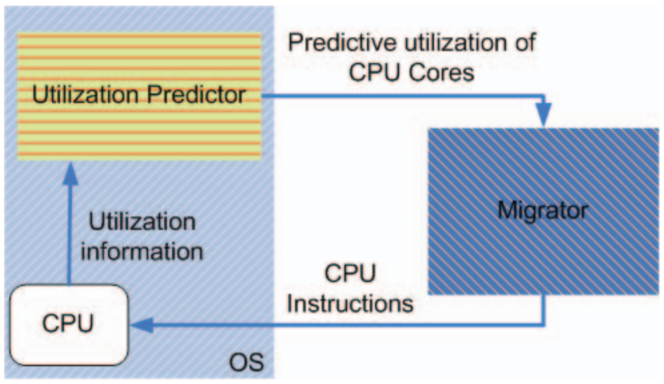


Fig. 1: The architecture of energy saving application.

A. *Working mechanism*

The architecture of energy saving solution for processor is described in Figure 1. As discussed above, the target of this proposed solution is to pro-actively mitigate the power consumption of the processor. An example of energy saving mechanism performing on quad-cores processor can be found in Figure 2. In fact, the number of cores in processor is not an important factor. Basically, the daemon application tries to empty the workload of the processor cores at first. Subsequently, the idle cores are turn off to save the energy. To overcome the delay in issuing the instruction for CPU, the migrator component is implemented to depend on the predictive utilization of the processor cores to determine the source and the destination for process migration. By definition, the aforementioned processes in this paper is the system-level process, which is being considered for migration. Primarily based on the utilization information polling from processor, which consists of the status of CPU cores, the predictive utilization is calculated in the utilization predictor component and keeps an important position in creating the migrating instructions for saving energy. In the next section, the method making the predictive utilization is introduced by using the Gaussian process regression technique.

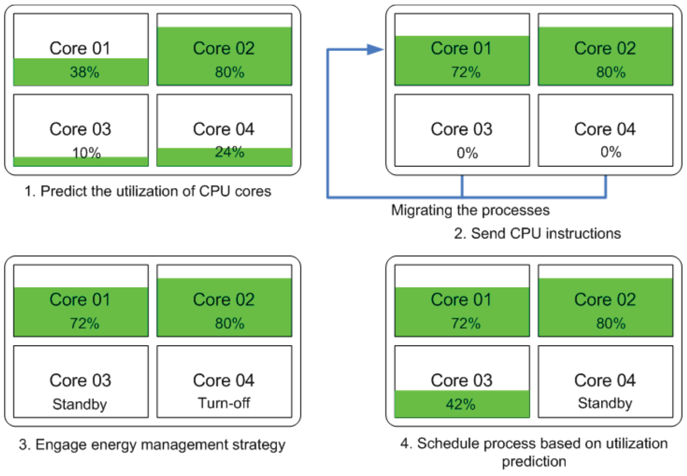


Fig. 2: Working mechanism of proposed solution.

B. *Prediction technique*

The objective of the prediction is to predict the utilization of the processor cores. To do that, the Bayesian learning [6] and Gaussian process regression (GPR) [7] [8] are coupled as the inference technique and probability framework, respectively. Because GPR is non-parametric method, the prediction can be achieved without specifying the parameters in the beginning. The prior knowledge of the system can be integrated into model distribution when analyzing the training data. For prediction perspective, GPR is considered to be one of the most accurate and flexible in comparison to other methods such as linear regression, k-nearest neighbour, multivariate regression splines, multi-layer perceptron and support vector regression [9]. Further, the result of GPR is tractable. It means that the result is always stable and existed. This result comprises the mean and variance of the Gaussian distribution, which represent the predictive CPU utilization and the confidence of the prediction, respectively. Because the input data for this model is the time series variable [10], curve-fitting is preferred over function mapping for the mapping approach. It is important to note that the curve-fitting is more flexible with regards to the time series data and non-stationary model.

Given a training data set $\mathcal{D} = (\mathbf{x}, y)$, where $\mathbf{x}$ is a limited collection of time location $\mathbf{x} = [x_1, x_2, x_3, \cdots x_n]$ and y is a finite set of random variable y=[y_1, y_2, y_3,$\cdots$ y_n] representing the Gaussian distribution of incoming system-level processes. This distribution running over time actually forms up the Gaussian process [11] [6] as below:

$$f(y|\mathbf{x}) \sim \mathcal{N}\big(m(x), k(x,x')\big) \tag{1}$$

with

$$m(x) = \mathbb{E}\big(f(x)\big) \tag{2}$$

$$k(x,x') = \mathbb{E}\Big(\big(f(x) - m(x)\big)\big(f(x') - m(x')\big)\Big) \tag{3}$$

in which, $m(x)$ is the mean function, evaluated at the time variable x, and $k(x, x')$ is the covariance function. By definition, the covariance function is a non-singular, positive-definite function [12], used to define the prior knowledge of the underlying relationship between the predictive function $f(y|\mathbf{x})$ and the realistic one. Theoretically, the covariance function can be dropped by directly calculating the sample if there is a existence of finite dimensional form of the feature space. Unfortunately, the feature space dimension is usually infinite, which leads to the fact that the covariance function cannot be skipped. Because of that, the covariance function is mandatory to build the Gaussian process regression model. For an aforementioned set of time location $\mathbf{x}$, the covariance should be defined in matrix form as follows.

$$\mathbf{K} = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{pmatrix} \tag{4}$$

By definition, the Radial Basis Function (RBF) [9] [13] is chosen as a covariance function. In reality, the RBF is favored in most Gaussian process applications, because its parameter requirements are limited to a few main characteristics of training data. Besides, there is another reason to engage RBF, as it is a universal kernel suitable for any continuous function when having enough data. The formula for RBF is described as follows:

$$k_{RBF}(x, x') = s^2 \exp\left(- \frac{(x_i - x_j)^2}{2\lambda^2} \right) \tag{5}$$

in which, s is the output scale of the covariance function and λ is a time-scale of the input x from two historical moments. Parameter s is used to control the magnitude of the fitting curves, while parameter λ stands for the bandwidth of the kernel. In addition, parameter λ can be used to govern the judgment for Automatic Relevance Detection (ARD). This extra functionality is mainly used to clear out the irrelevant input data with regards to the curve-fitting.

After building the covariance function, the posterior distribution of Gaussian process can be started to estimate. Denoting the set of the input data is (x, y), and the desired predictive value is y_*, we formulate the distribution of GPR as follows.

$$p\left(\begin{bmatrix} y \\ y_* \end{bmatrix} \right) = \mathcal{N}\left(\begin{bmatrix} m(x) \\ m(x_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K}(x, x') & \mathbf{K}(x, x_*) \\ \mathbf{K}(x_*, x) & \mathbf{K}(x_*, x_*) \end{bmatrix} \right) \tag{6}$$

here, $\mathbf{K}(x_*, x_*) = k(x_*, x_*)$, $\mathbf{K}(x, x_*)$ is the column vector made from $k(x_1, x_*), k(x_2, x_*) \cdots, k(x_n, x_*)$. In addition, $\mathbf{K}(x_*, x) = \mathbf{K}(x, x_*)^\top$ is the transpose of $\mathbf{K}(x, x_*)$. Subsequently, the posterior distribution over y_* can be evaluated with the below mean m_* and covariance C_*.

$$m_* = m(x_*) + \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}(y - m(x)) \tag{7}$$

$$C_* = \mathbf{K}(x_*, x_*) - \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1}\mathbf{K}(x, x_*) \tag{8}$$

then

$$p(y_*) \sim \mathcal{N}(m_*, C_*) \tag{9}$$

The optimal estimation of y_* can be obtained by using $\overline{y}_*$ which is the mean of this distribution. This optimal estimation is also recognized as the predictive CPU utilization in the next epoch and can be obtained as below:

$$\overline{y}_* = \mathbf{K}(x_*, x)\mathbf{K}(x, x')^{-1} y \tag{10}$$

IV. Performance Evaluation

A. Experiment

TABLE I: System configuration

	Configuration
Platform	64bit
CPU	Intel®Core™ i7-3770, 3.40GHz
Storage	400GB
Memory	8GB
OS	CentOS 6.5 (final)
Benchmark	*stress*-1.0
Power stat	*powerstat*-0.01
System stat	*sysstat*-9.0.4

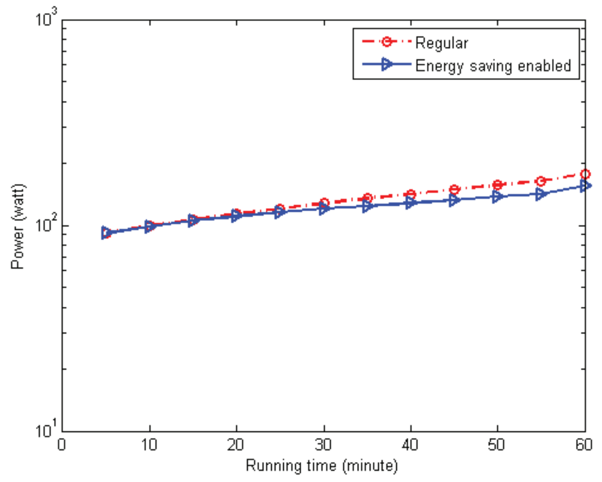
In the experiment, the effectiveness of proposed method is investigated with regards to the energy efficiency. Basically, the workload is generated into two analogous systems *via* the CPU intensive benchmark software for one hour to evaluate the power consumption. The software *stress*-1.0.4 is chosen for benchmarking because of the convenience in controlling the number of tasks as well as the intensiveness of workload. Normally, this benchmark software is used for pushing a large number of system-level processes. For collecting the experimental result, *powerstat* and *sysstat* are used to monitor the power consumption and workload statistics, respectively. The full configuration of the benchmarking system can be found in Table I.

B. Result

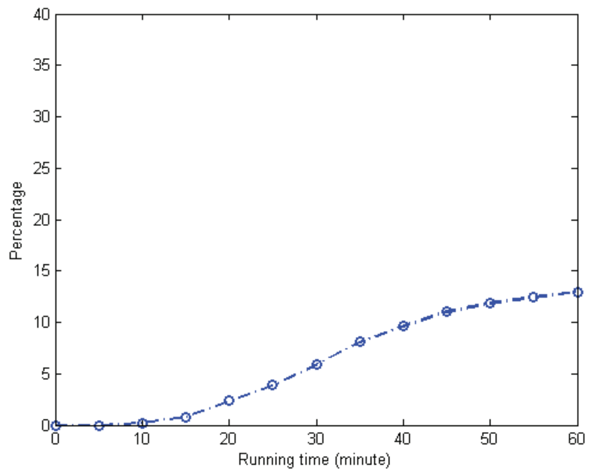
Two systems begin with the stand-by mode, which costs 91.49 watts to keep the systems in idle status (Figure $3a$). When the experiment starts, a number of concurrent stress tests are performed as mentioned above for one hour. In the end of the test, the system with energy saving daemon enabled finishes the experiment by consuming 154.93 watts. Meanwhile, the regular system consumes an amount of power up to 177.96 watts in the same experiment. It means that 23.03 watts are saved (up to 12.94% of power consumption). This result can be seen in Figure $3b$. In processor architecture, an energy savings of 12.94% can be considered as a significant improvement.

V. Conclusion

By proposing the prediction technique, we prove that the proactive approach can be use to achieve the energy efficiency for processor. Besides, the mechanism only consists of three steps, which is simple enough for implementation: first predict the utilization of CPU cores, then migrate the target processes between cores, and finally turn off the empty cores to save the energy. In the near future, the prediction technique is going to upgrade in terms of reaction rate to enhance further the performance. In addition, the proposed prediction technique is also considered to apply to other research fields such as cloud computing and big data analysis.



(a) Comparison between two systems on energy consumption(lower is better)



(b) Energy savings measurement

Fig. 3: Energy efficiency measurement

VI. Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government (MSIP) (No.R0101-15-237, Development of General-Purpose OS and Virtualization Technology to Reduce 30% of Energy for High-density Servers based on Low-power Processors).

References

[1] T. D. Burd, T. Pering, A. J. Stratakos, R. W. Brodersen *et al.*, "A dynamic voltage scaled microprocessor system," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 11, pp. 1571–1580, 2000.

[2] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: understanding the runtime effects of frequency scaling," in *Proceedings of the 16th international conference on Supercomputing*. ACM, 2002, pp. 35–44.

[3] J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with pace," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1. ACM, 2001, pp. 50–61.

[4] C. Lefurgy, X. Wang, and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.

[5] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation." in *NSDI*, vol. 8, 2008, pp. 323–336.

[6] S. Brahim-Belhouari and J. Vesin, "Bayesian learning using gaussian process for time series prediction." in *Statistical Signal Processing, 2001. Proceedings of the 11th IEEE Signal Processing Workshop on*, 2001, pp. 433–436.

[7] D. Petelin, B. Filipič, and J. Kocijan, "Optimization of gaussian process models with evolutionary algorithms." in *Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms - Volume Part I*, ser. ICANNGA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 420–429. [Online]. Available: http://dl.acm.org/citation.cfm?id=1997052.1997098

[8] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data." *Advances in neural information processing systems*, vol. 16, pp. 329–336, 2004.

[9] C. E. Rasmussen, "Evaluation of gaussian processes and other methods for non-linear regression." Ph.D. dissertation, Toronto, Ont., Canada, Canada, 1997, aAINQ28300.

[10] J. Cunningham, Z. Ghahramani, and C. E. Rasmussen, "Gaussian processes for time-marked time-series data." in *AISTATS*, ser. JMLR Proceedings, N. D. Lawrence and M. Girolami, Eds., vol. 22. JMLR.org, 2012, pp. 255–263.

[11] M. E. Tipping, "Bayesian inference: An introduction to principles and practice in machine learning." in *Advanced Lectures on Machine Learning*, ser. Lecture Notes in Computer Science, O. Bousquet, U. von Luxburg, and G. Rtsch, Eds., vol. 3176. Springer, 2003, pp. 41–62. [Online]. Available: http://dblp.uni-trier.de/db/conf/ac/ml2003.html

[12] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms." *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 181–201, Mar 2001.

[13] M. A. Álvarez and N. D. Lawrence, "Computationally efficient convolved multiple output gaussian processes." *J. Mach. Learn. Res.*, vol. 12, pp. 1459–1500, Jul. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2021048