EAP: Energy-Awareness Predictor in Multicore CPU

Dinh-Mao Bui[1], Thien Huynh-The[1], YongIk Yoon[2], SungIk Jun[3], Sungyoung Lee[1]

[1] Computer Engineering Department, Kyung Hee University, Suwon, Korea
{mao, thienht, sylee}@oslab.khu.ac.kr
[2] Department of Multimedia Science, SookMyung Women's University, Seoul, Republic of Korea
yiyoon@sm.ac.kr
[3] HPC system research section/Cloud computing department/ETRI, Daejeon, Korea
sijun@etri.re.kr

Abstract. To deal with inference and reasoning problems, Gaussian process has been considered as a promising tool due to the robustness and flexibility features. Especially, solving the regression and classification, Gaussian process coupling with Bayesian learning is one of the most appropriate supervised learning approaches in terms of accuracy and tractability. Because of these features, it is reasonable to engage Gaussian process for energy saving purpose. In this paper, the research focuses on analyzing the capability of Gaussian process, implementing it to predict CPU utilization, which is used as a factor to predict the status of computing node. Subsequently, a migration mechanism is applied so as to migrate the system-level processes between CPU cores and turn off the idle ones in order to save the energy while still maintaining the performance.
Keywords: Proactive prediction, Bayesian learning, Gaussian process, energy efficiency, CPU utilization

1 Introduction

Research in energy consumption is one of the hot trends in recent years. From time to time, this topic has been the major issue in most of the computing systems. Coming up with the higher performance that the computing node achieves, a large amount of energy is also taken in to account as significant costs. Theoretically, energy is used to conduct the computation, especially in servers. When the servers are up and running, even when they are idle, the energy is still wasted to maintain the power-on status. For that reason, enhancing the energy efficiency in computing system concerns reducing the energy waste on idle computing facilities. However, this effort is challenged by the problem to concurrently maintain the performance.

In this paper, we propose a proactive solution for energy saving in CPU level. By applying the prediction technique which is the Gaussian process regression, the energy application predicts the multicore CPU utilization and activates the process migration between the cores in order to achieve the overall energy efficiency while maintaining a high performance.

This paper is organized as follows. We detail our energy efficiency approach in Section 2. In Section 3, we conduct the performance evaluation of the energy saving application. Our conclusion is summarized in Section 4.

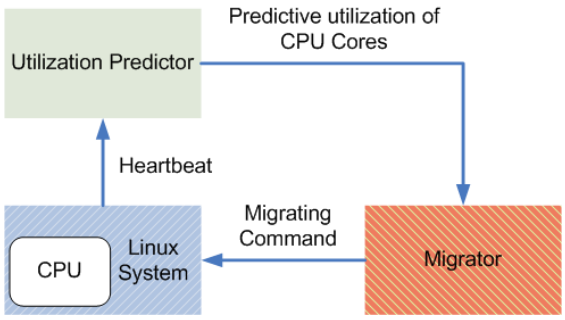


Fig.1 The architecture of energy saving application.

2 Proposed Method

2.1 Target process

The energy efficiency architecture for a multi-cores CPU is proposed in this section and described in detail in Fig.1. Basically, the purpose of this architecture is to pro-actively reduce the energy consumption of the CPU cores. Thus, the main functionality of this application is to empty the workload of the CPU cores and then deactivate or stand-by the idle cores to save energy. To do this effectively, the migration procedure on the Migrator component needs the predictive information of the CPU core utilization to determine the source and the destination in order to migrate the target process (from this point, the system process being considered migration will be known as the target process). Primarily based on the current value of the CPU core's utilization, known as the heart-beat, the predictive utilization is calculated in the Utilization Predictor component and plays a critical role in subsequent decision making.

2.2 Prediction model

In our application, the object of the prediction is to anticipate the utilization of the CPU cores. Bayesian learning and Gaussian [1,2,3,4] process regression are employed as the inference technique and probability framework, respectively. Because the input data for this model is the time series utilization [5], curve-fitting is preferred over

function mapping for the mapping approach. It is important to note that the curve-fitting is more flexible with regard to the time series data and non-stationary model.
Assuming that the input data is a limited collection of time location $x=[x_1, x_2, x_3,\ldots x_n]$, a finite set of random variable $Y=[Y_1, Y_2, Y_3,\ldots Y_n]$ represents the corresponding joint Gaussian distribution of incoming processes with regard to the time order. This set over the time constraint actually forms up the Gaussian process:

$$f(y|x) \sim GP(m(x), k(x, x')) \tag{1}$$

with

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

and

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

in which, $m(x)$ is the mean function, evaluated at the location x variable, and $k(x,x')$ is the covariance function, also known as the kernel function[6,7,8]. By definition, the kernel function is a positive-definite function, used to define the prior knowledge of the underlying relationship. Basically, the kernel function is only a mandatory requirement when there is a lack of finite dimensional form of the feature space. Otherwise, it can be dropped by directly calculating the sample. However, this feature space dimension is frequently infinite, which means that the kernel function cannot be directly calculated. For this reason, the kernel function technique is often chosen to tackle the Gaussian process regression. In addition, the kernel function comprises some special parameters that specify its own shape. These parameters are referred to as hyper-parameters.
Generally, the Square-Exponential (SE) kernel, also known as the Radial Basis Function (RBF) kernel [9,10], is chosen as a basic kernel function. In reality, the SE kernel is favored in most Gaussian process applications, because it requires the calculation of only a few parameters. Moreover, there is a theoretical reason to choose this method, as it is a universal kernel appropriate for any continuous function when enough data is given. The formula for SE kernel is described as follows:

$$k_{SE}(x, x') = \sigma_f^2(-\frac{(x - x')^2}{2l^2}) \tag{4}$$

in which, σ_f is the output-scale amplitude and l is the time-scale of the variable x from one moment to the next. l also stands for the bandwidth of the kernel and, thereby, the smoothness of the function in the model. In addition, l also plays the role of judgement for Automatic Relevance Detection (ARD) [11,12], to discard the irrelevant input dimension. In the next step, we evaluate the posterior distribution of the Gaussian process. Assuming that the incoming value of the input data is (x_*, y_*), the joint distribution of the training output is y, and the test output is y_*, as below.

$$p\begin{pmatrix} y \\ y^* \end{pmatrix} = GP\begin{pmatrix} m(x) & \boldsymbol{K}(x, x')\boldsymbol{K}(x, x_*) \\ m(x_*)\,' & \boldsymbol{K}(x_*, x)\boldsymbol{K}(x_*, x_*) \end{pmatrix} \tag{5}$$

here, $\boldsymbol{K}(x_*,x_*)=k(x_*,x_*)$, $\boldsymbol{K}(x,x_*)$ is the column vector made from $k(x_1,x_*)$, $k(x_2,x_*)\ldots,k(x_n,x_*)$. In addition, $\boldsymbol{K}(x_*,x) = \boldsymbol{K}(x,x_*)^T$ is the transposition of $\boldsymbol{K}(x,x_*)$.

Subsequently, the posterior distribution over y_* can be evaluated with the below mean m_* and covariance C_*.

$$m_* = m(x_*) + \boldsymbol{K}(x_*, x)\boldsymbol{K}(x, x')^{-1}(y - m(x)) \tag{6}$$

$$C_* = \boldsymbol{K}(x_*, x_*) - \boldsymbol{K}(x_*, x)\boldsymbol{K}(x, x')^{-1}\boldsymbol{K}(x, x_*) \tag{7}$$

then

$$p(y_*) \sim GP(m_*, C_*) \tag{8}$$

The best estimation for y_* is the mean of this distribution:

$$\overline{y_*} = \boldsymbol{K}(x_*, x)\boldsymbol{K}(x, x')^{-1}y \tag{9}$$

In addition, the uncertainty of the estimation is captured in the variance of the distribution as follows:

$$var(y_*) = \boldsymbol{K}(x_*, x_*) - \boldsymbol{K}(x_*, x)\boldsymbol{K}(x, x')^{-1}\boldsymbol{K}(x, x_*) \tag{10}$$

3. Performance Evaluation

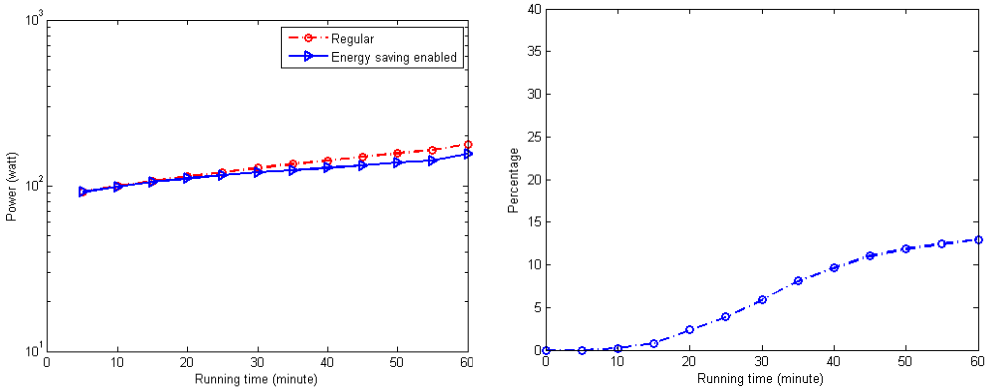
Table 1. System configuration.

	Configuration
Platform	64bit
CPU	Intel Core i7-3770, 3.40GHz, Quad core
Storage	800GB
Memory	16GB
OS	CentOS 6.5 kernel: 2.6.32-431.el6.x86_64
Benchmark	stress-1.0.4
Power stat	powerstat-0.01.30-1
System stat	sysstat-9.0.4-27.el6

3.1 Experiments

For the performance evaluation, our experiments are aimed at investigating the performance of the proposed application in terms of energy efficiency and execution time. In the initial experiments, the workload is generated *via* the CPU intensive benchmark for one hour to determine the energy savings. In this test, in order to more easily control the number and the intensiveness of the workload, a benchmark software, namely *stress*-1.0.4, is used to simulate the incoming processes. To aggregate the results, the *powerstat* and the *sysstat* software are used to log the power consumption and workload statistics, respectively. All of the information of the benchmarking system is described in Table 1.

3.2 Results



(a) Power consumption over one hour running time (lower is better).

(b) Power saving over one hour running time.

Fig.2 Performance evaluation of proposed method.

As seen in the Fig.2a, both systems begin with the stand-by mode, which costs 91.49 watts to maintain. Both systems had simultaneous stress tests for duration of 60 minutes. Subsequently, the system with energy saving enabled ends the benchmark test with a consumption of 154.93 watts, in comparison with 177.96 watts for the regular system. Therefore, 23.03 watts are saved (which is equivalent to an energy savings of 12.94%) (Fig.2b). In processor architecture, an energy reduction of 12.94% is significant.

4 Conclusion

The proposed method proves the capability in improving the power consumption of the computing node. To do that, the strategy is to predict the utilization of CPU cores, migrate the target processes and stand-by the idle cores to save the energy. For further development, as previously mentioned, some parts of the source code that are developed to test this method would be made available under the terms of the GNU general public license (GPL).

5 Acknowledgements

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] Lawrence, N. D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. Advances in neural information processing systems, 16(3), 329-336.
[2] Rasmussen, C. E. (1996). Evaluation of Gaussian processes and other methods for non-linear regression (Doctoral dissertation, University of Toronto).
[3] Chalupka, K., Williams, C. K., & Murray, I. (2013). A framework for evaluating approximation methods for Gaussian process regression. The Journal of Machine Learning Research, 14(1), 333-350.
[4] Williams, C. K., & Rasmussen, C. E. (2006). Gaussian processes for machine learning. the MIT Press, 2(3), 4.
[5] Brahim-Belhouari, S., & Vesin, J. M. (2001). Bayesian learning using Gaussian process for time series prediction. In Statistical Signal Processing, 2001. Proceedings of the 11th IEEE Signal Processing Workshop on (pp. 433-436). IEEE.
[6] Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., & Aigrain, S. (2013). Gaussian processes for time-series modelling. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 371(1984), 20110550.
[8] Petelin, D., & Kocijan, J. (2014, June). Evolving Gaussian process models for predicting chaotic time-series. In Evolving and Adaptive Intelligent Systems (EAIS), 2014 IEEE Conference on (pp. 1-8). IEEE.
[9] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain.
[10] Grande, R. C., Chowdhary, G., & How, J. P. (2013, December). Nonparametric adaptive control using Gaussian Processes with online hyperparameter estimation. In Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on (pp. 861-867). IEEE.
[11] Banerjee, A., Dunson, D. B., & Tokdar, S. T. (2012). Efficient Gaussian process regression for large datasets. Biometrika, ass068.
[12] Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. arXiv preprint arXiv:1309.6835.