Particle Swarm Optimization with SIMD-Oriented Fast Mersenne Twister on the Cell Broadband Engine

Jun Igarashi[1], Satoshi Sonoh[1], and Takanori Koga[2]

[1] Graduate School of Life Science and Systems Engineering
Kyushu Institute of Technology
2-4 Hibikino, Wakamatsu-ku, Kitakyushu, 808-0196, Japan
{igarashi@,sonoh@edu.}brain.kyutech.ac.jp
[2] Graduate School of Science and Engineering
Yamaguchi University
1677-1 Yoshida, Yamaguchi-shi Yamaguchi, 753-8511, Japan
koga@ic.sci.yamaguchi-u.ac.jp

Abstract. We introduce a processing performance of Particle Swarm Optimization with SIDM-oriented Fast Mersenne Twister on the Cell Broadband Engine. Extreme-high processing performance is demanded for solving very complex optimization problem in a small amount of time. In this research, we verified the effectiveness of employing SIMD-oriented Fast Mersenne Twister on the Cell Broadband Engine for the processing of Particle Swarm Optimization by numerical simulations.

1 Introduction

Optimization problem is the problem of finding out the best solution from all feasible solutions. Particle Swarm Optimization (PSO) [1], which is a meta-heuristic inspired by a swarm of insects or a pod of fish, shows excellent performance for optimization problems and has been successfully applied to function optimization [2] and neural network training [3]. PSO achieves to search the best solution by using several particles that represent candidate solutions. PSO has a good feature that the algorithm is described by simple equations that update positions of the particles using a pseudo random number. However, its computational cost drastically increases according to the number of the particles, search dimensions and search trials. PSO requires vast amount of calculations including a generation of pseudo random number. The generation of pseudo random number is very important to the performance of PSO and relatively occupies a large amount of processing time in PSO processing. Thus, a fast generation method of the pseudo random number is essential for improvement of PSO in aspect of the processing time.

SIMD-oriented Fast Mersenne Twister (SFMT) is designed for fast generation of the pseudo random number using Single Instruction Multiple Data (SIMD) operation and multi-stage pipelines of modern CPUs. SFMT generates pseudo random numbers extremely fast compared with conventional ones. Furthermore, quality of the generated numbers is superior as random numbers. In order to bring out the performance of PSO with SFMT, a suitable processing unit should be used. In this paper, we propose

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1065–1071, 2009.
© Springer-Verlag Berlin Heidelberg 2009

to apply the Cell Broadband Engine (Cell/B.E.) processor to PSO with SFMT. The Cell/B.E. processor is a heterogeneous processor that targets video games, multimedia applications, and scientific computation. The Cell/B.E. has excellent floating-point processing performance compared with other old supercomputers.

Additionally, the Cell/B.E. is available for a scientific research with a reasonable price because it is mounted on the game console, PLAYSTAION3. In this paper, we implement PSO algorithm with SFMT on the Cell/B.E. processor for speed-up of PSO. Finally, we show that the Cell/B.E. effectively perform PSO with SFMT compared with a general processor.

2 Implementation of Particle Swarm Optimization with SIMD-Oriented Fast Mersenne Twister on the Cell Broadband Engine

2.1 Particle Swarm Optimization and DeJong's Benchmark

Fig. 1 shows the schematic picture of PSO. Each individual in the particle swarm is composed of three d-dimensional vectors, where d is the dimensionality of the search space. The current position, the previous best position and the velocity are denoted by $\boldsymbol{x}_i$, $\boldsymbol{p}_i$ and $\mathbf{v}_i$, respectively. Here, i is index of the individual. In the particle swarm optimization process, the velocity of each particle is iteratively adjusted so that the particle stochastically oscillates around $\boldsymbol{p}_i$ and $\boldsymbol{p}_g$. $\boldsymbol{p}_g$ is the best of all $\boldsymbol{p}_i$. At each step, the desired optimization fitness function in d variables is evaluated, then, $\boldsymbol{p}_i$ and $\boldsymbol{p}_g$ are renewed. The equation of PSO is described as follow,

$$\begin{cases} \mathbf{v}_i^{k+1} = \gamma \mathbf{v}_i^k + c_1 \mathbf{U}(0,\phi_1) \otimes \left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2 \mathbf{U}(0,\phi_2) \otimes \left(\mathbf{p}_g^k - \mathbf{x}_i^k\right) \\ \mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^k. \end{cases}$$

Here, gamma = 0.99, C1 = 0.9 and C2 = 0.9 are parameters which govern search policy of each particle. k is time step. U() is a vector of pseudo random numbers uniformly distributed in [0, φ]. $\otimes$ is component-wise multiplication.

As a benchmark of PSO, DeJong's benchmarks [4] were used. In the experiments, 100 and 500 iterations were performed for F1-F3 and F4-F5, respectively. 2400 trials were done in each benchmark.

2.2 SIMD-Oriented Fast Mersenne Twister

SFMT is proposed by M. Matsumoto and M. Saito in 2006 [5]. SFMT is a linear feed back shift register type pseudo random number generator. SFMT is designed considering new features of modern CPUs, such as Single Instruction Multiple Data (SIMD) operation and multi-stage pipeline processing.

We coded SFMT for the Cell/B.E. by modifying SFMT version 1.3.3 for PowerPC Altivec provided on the website [6]. In the case of using SFMT on the Core 2, SIMD processing was disabled. The fill_array32 function was used in SFMT for batch processing of pseudo random number generation. The rand() function of standard library of ANSI C was used for performance comparison with SFMT.

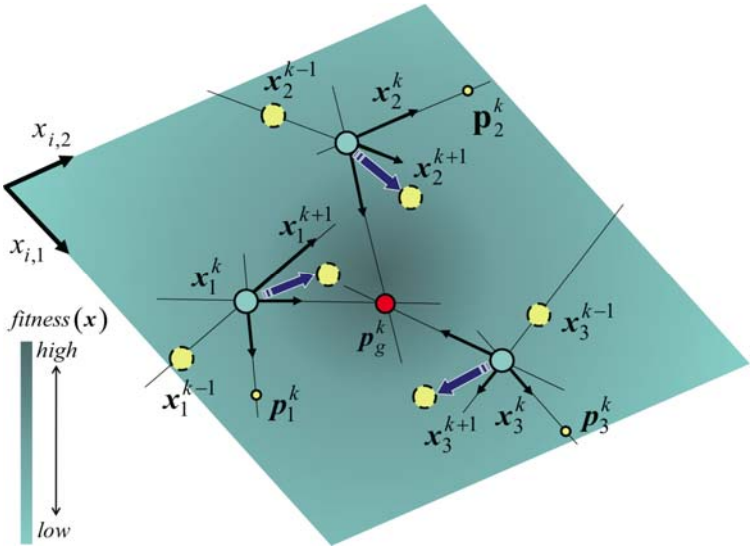

Fig. 1. The schematic picture of PSO. Particles represented as circles that search the optimum value in cooperation with other particles.

2.3 Cell Broadband Engine Processor

The Cell/B.E. is a heterogeneous microprocessor that results from a joint effort among the Sony Group, Toshiba and IBM. The Cell/B.E. has a PowerPC processor Element (PPE) designed for general-purpose processing and 8 Synergistic Processor Elements (SPEs) designed for high performance floating-point computation [7]. In the Linux programming environment of PLAYSTATION3, the one PPE and the six SPEs of the Cell/B.E. are available. The peak performance of the Cell/B.E. processor of PLAYSTATION3 is about 200 GFLOPS. In this research, we mainly used the SPEs for PSO processing except for starting and ending process of the SPE threads by the PPE. Numerical calculation was performed with single precision. Table.1 shows equipment of the Cell/B.E. Machine and Core 2 Machine.

Table 1. The equipment of the Cell/B.E. Machine and the Core 2 Machine

	Cell/B.E. Machine: PLAYSTATION3	PC: IBM PC/AT Compatible
CPU	Cell B.E. 3.2 GHz (PPE +6SPE)	Core 2 Duo 2.66GHz (1core usage)
Memory	XDR 256MB	DDR2 2GB
OS	FedoraCore6 (kernel 2.6.18-53.1.4.el5)	CentOS5 (kernel2.6.18-53.1.4.el5)
Compiler	GCC 4.1.1 for Cell B.E (-O3)	Intel Compiler ICC (-O3)
Library	Cell SDK 2.1, libspe2, SIMD Math Library	

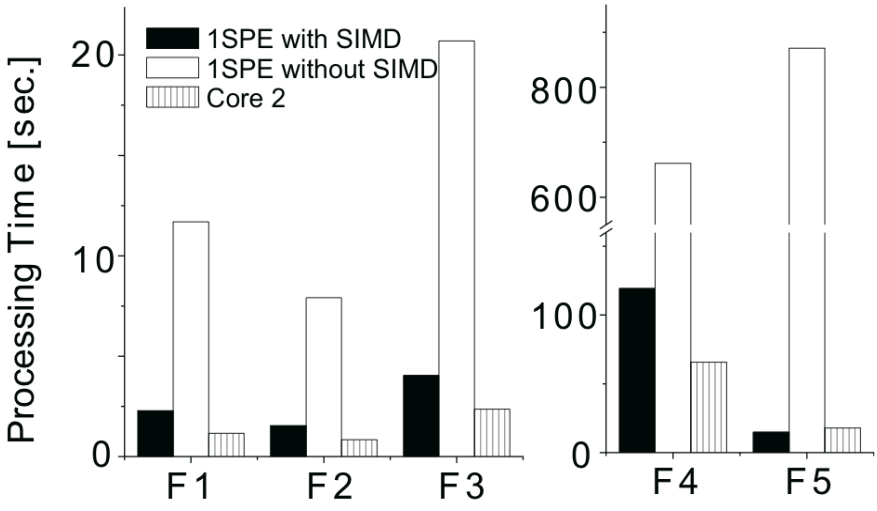

Fig. 2. The reduction of processing time by using SIMD processing on the one SPE in De-Jong's benchmark

3 Experimental Results and Discussions

First, we compared the processing time using the one SPE of the Cell/B.E. with that using the Core 2 processor when the rand() function of ANSI C was used for pseudo random number generation in PSO processing. Fig. 2 shows the results of DeJong's benchmarks of PSO using the one SPE with SIMD instructions, the one SPE without SIMD instructions and the Core 2 with normal C program. In the SIMD processing, 4

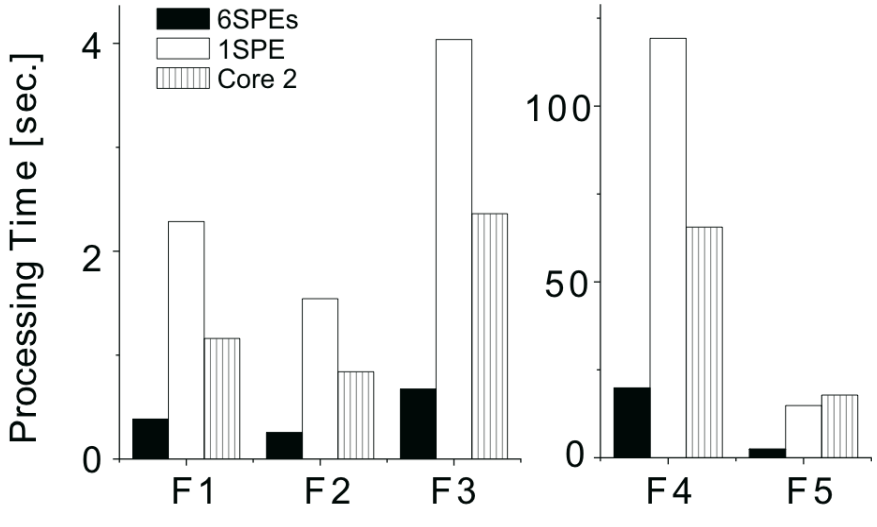

Fig. 3. The performances calling with increase in the number of SPE

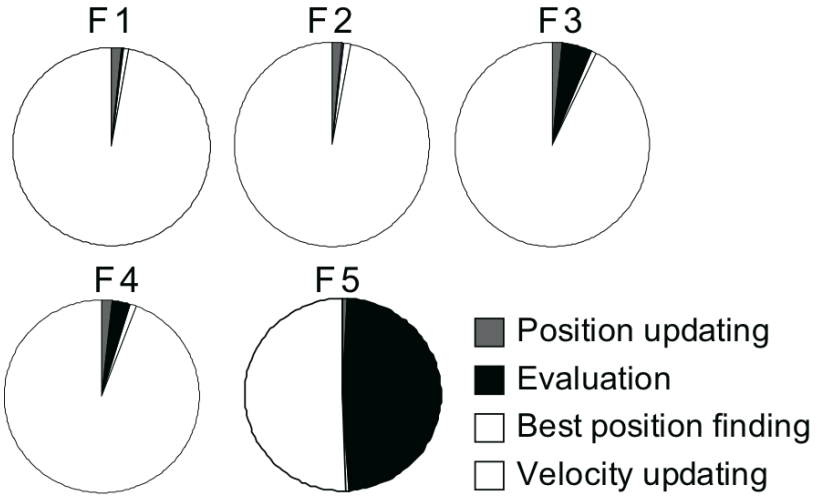

Fig. 4. Pie charts for processing time ratios

trials were assigned to each element of one vector, and performed in parallel. In the all benchmarks, by using SIMD, the processing times of Cell/B.E. were shortened more than 5 times. These were because of parallelism effect by SIMD instruction and efficient execution of intrinsic functions for the SIMD instruction. The processing times using the one SPE with SIMD instructions took twice as much as that of the Core2.

Fig. 3 shows the results of DeJong's benchmarks of PSO when the one SPE, the six SPEs or the Core 2 was used with the rand() function. The six SPEs computed PSO 6 times faster than the one SPE. This performance scaling occurred due to parallel execution of independent trials on the six SPEs. As compared with the Core 2, the six SPEs took 1/7 to 1/3 of processing time. Fig. 4 shows a pie chart showing processing time ratio. PSO processing mainly consisted of 4 parts, updating of particle position, evaluation, finding of the best value and updating of velocity. The longest time in all benchmarks was taken in the part of updating of velocity that mainly performed pseudo random number generation. Speeding up of random number generation should be effective for shortening processing time of PSO.

Table 2. The processing time ratios between the Cell/B.E. and the Core 2 with SFMT or the rand() function

	F1	F2	F3	F4	F5
Core 2-rand/ Cell-SFMT	41.4	42.0	29.1	33.5	13.9
Core 2-SFMT/ Cell-SFMT	13.9	15.0	13.7	12.1	12.0
Core 2-rand/ Cell-rand	3.0	3.3	3.5	3.3	7.2

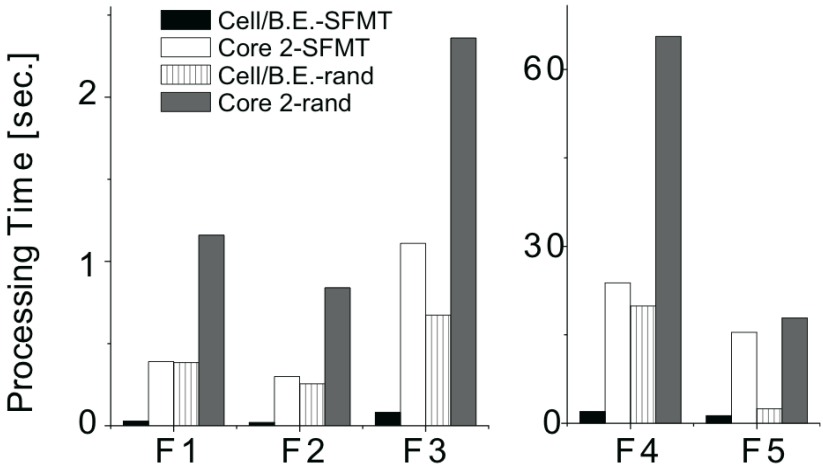

Fig. 5. The high speed processing of PSO by the Cell/B.E. with SFMT

Next, we introduced SFMT into PSO processing. Fig. 5 shows processing times of DeJong benchmarks of PSO using the six SPE of the Cell/B.E. or the Core 2. In the cases, SFMT or rand() function was used for pseudo random number generator. In the both case of the Cell/B.E. and the Core 2, processing times were drastically shorten by employing SFMT as compared with the rand() function. Combination of SFMT and the Cell/B.E. performed fastest computation in the all benchmarks. Table 2 shows processing time ratios between the Cell/B.E. and the Core 2 shown in Fig. 5. The combination of the Cell/B.E. and SFMT were approximately 14 to 42 times faster than the combination of the Core 2 and the rand() function (see Table 2 row 1). Table 2 also shows effectiveness of combination of SFMT and the Cell/B.E. When the rand() function was used for both processors, the Cell/B.E. calculated faster just 3 to 7 times than the Core 2 (see Table 2 row 3). On the other hand, when SFMT was used for the both processors, the Cell B.E. calculated 12 to 15 times faster than the Core 2 (see Table 2 row2). This result suggests that SFMT brought out high floating-point performance of the Cell/B.E. because of its optimization for SIMD and pipeline feature, comparing with the rand() function that performed scalar instruction and was not specially designed for pipeline features.

4 Conclusion

In this paper, we showed that fast processing of PSO using SFMT on the Cell/B.E.. By only applying multi-core processing and SIMD processing technique of Cell/B.E. for PSO processing, PSO processing by the Cell/B.E. were from 3 to 7 times faster than the general processor, Core 2. Furthermore, by applying SFMT on the Cell/B.E., the Cell/B.E. calculated PSO processing faster from approximately 14 to 42 times than Core 2 using rand() function.

Acknowledgments. This work is partially supported by Kitakyushu Foundation for the Advancement of Industry, Science and Technology.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE the International Conference on Neural Networks, pp. 1942–1948 (1995)
2. Clerc, M.: Particle Swarm Optimization. ISTE USA, USA (2006)
3. Cai, X., Zhang, N., Venayagamoorthy, G.K., Wunsch, D.C.: Time series prediction with recurrent neural networks trained by a hybrid PSO–EA algorithm. Neurocomputing 70, 2342–2353 (2007)
4. DeJong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)
5. Saito, M., Matsumoto, M.: SIMD-oriented Fast Mersenne Twister: a128-bit Pseudorandom Number Generator. In: Keller, A., Heinrich, S., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carl Methods 2006, pp. 607–622. Springer, Heidelberg (2008)
6. SIMD-oriented Fast Mersenne Twister (SFMT), `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index-jp.html`
7. Johns, C.R., Brokenshire, D.A.: Introduction to the Cell Broadband Engine Architecture. IBM Journal of Research and Development 51, 503–520 (2007)