An Approach to the Highest Efficiency of the HPCG Benchmark on the SX-ACE Supercomputer

Kazuhiko Komatsu, Ryusuke Egawa,
Hiroyuki Takizawa, Hiroaki Kobayashi
Tohoku University
6-3 Aramaki Aza Aoba, Aoba-ku
Sendai, 980-8578, Japan
{komatsu,egawa,takizawa,koba}@cc.tohoku.ac.jp

Yoko Isobe, Ryusei Ogata
NEC Corporation
5-7-1, Shiba, Minato-ku
Tokyo, 108-8001, Japan
{y-isobe@pi,r-ogata@ak}.jp.nec.com

ABSTRACT

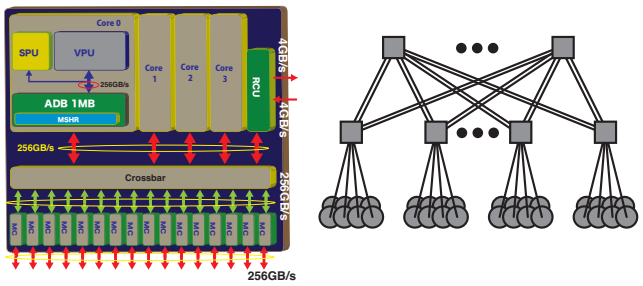
This paper reports how we have achieved the world's highest efficiency on the HPCG benchmark by fully exploiting the high potential of the SX-ACE supercomputer. To achieve efficient vector calculations and memory accesses, several optimization techniques such as ELL sparse matrix data packing, the hyperplane method for eliminating data dependencies, selective caching for the on-chip memory, and problem size tuning have been employed. Evaluation results show that SX-ACE achieves an 11.4% efficiency in the case of 512 nodes, which is the highest efficiency among all of the supercomputers in the latest HPCG ranking. This is because these optimization techniques can effectively exploit the high potential of memory access performance of SX-ACE, which has a strong impact on the overall performance of HPCG.

1. INTRODUCTION

The HPCG (High Performance Conjugate Gradient) benchmark has been developed as a more relevant benchmark to characteristics of practical HPC applications[1]. This poster presents optimization techniques of HPCG to exploit the potential of the SX-ACE supercomputer[2], and then the performances on SX-ACE are examined through experimental evaluations.

2. PRELIMINARY EVALUATION OF THE REFERENCE HPCG ON SX-ACE

The SX-ACE supercomputer in Tohoku University consists of 512 nodes by a two-stage fat-tree custom network as shown in Figure 1. Each node is equipped with an SX-ACE processor whose major features are high vector computational performance by its four vector-cores, high sustained memory bandwidth by a strong memory subsystem and software-controllable on-chip memory called Assignable Data Buffer (ADB). To exploit the potential of SX-ACE, vector calculations by the vector-cores and use of ADB to keep its high sustained memory bandwidth are essential.



(a) SX-ACE processor (b) Two-stage fat-tree network

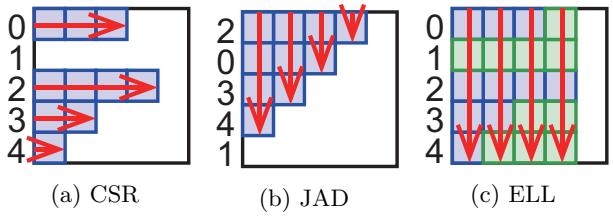
Figure 1: Overview of the SX-ACE supercomputer

Firstly, the preliminary evaluation on SX-ACE has been conducted. From the result, multigrid preconditioned conjugate gradient (MG) and sparse matrix-vector multiplication (SpMV) spend most of the execution time. The performances of SpMV and MG are 0.44 GFLOP/s and 0.59 GFLOP/s that are only 0.17% and 0.23% of the peak performance, respectively. We analysed and identified that these low performances were caused by quite low vectorization rates and inefficient memory accesses.

3. OPTIMIZATIONS OF HPCG FOR SX-ACE

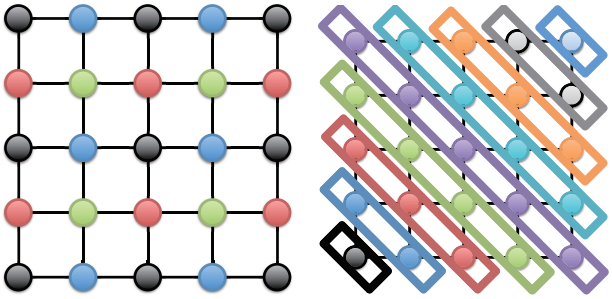
To achieve high sustained performance of HPCG on SX-ACE, several optimization techniques have been examined such as data packing, the multicoloring method, the hyperplane method, selective caching, and tuning of the problem size.

As the first optimization, in addition to the CSR (Compressed Sparse Row) default storage format of HPCG, two other data packing formats, JAD (Jagged Diagonal Format)[3][4] and ELLPACK (ELL)[5][6], have been examined to achieve efficient vector calculations and memory accesses as shown in Figure 2. By reordering the row of CSR in order of the number of non-zero elements, JAD can easily be generated, which leads to efficient vector calculations and memory accesses compared with CSR. Furthermore, JAD can be employed in the SpMV function of a highly-optimized numerical library provided by the vendor. ELL is well known as a suited format for the vector architecture[6]. By padding



(a) CSR (b) JAD (c) ELL

Figure 2: Sparse matrix storage format



(a) Eight-coloring (2D) (b) Hyperplane (2D)

Figure 3: Two parallelization approaches

zero elements into CSR, ELL can achieve more efficient vector calculations and memory accesses than JAD.

Second, to eliminate data dependencies for parallelization, the multicoloring method[7] and the hyperplane method[8][9] have been implemented. For the multicoloring method, eight colors are used as shown in Figure 3(a). The elements with the same color can be calculated in parallel. In the hyperplane method, by accessing elements of an array in the diagonal direction as shown in Figure 3(b), the elements in the hyperplane can be calculated in parallel.

Third, only highly-reusable data are manually stored in ADB by exploiting programmer's knowledge. Compared with storing reusable data judged by the compiler, ADB can be effectively utilized.

Finally, to avoid evicting highly-reusable data from ADB, the problem size of HPCG has been tuned under consideration of both the capacity of ADB and the size of hyperplanes. Especially, the size of each dimension is carefully selected considering the characteristics of the hyperplane method.

4. PERFORMANCE EVALUATION OF HPCG ON SX-ACE

Figure 4 shows the impact of each optimization on performance in the case of a single node of SX-ACE. By employing JAD and the eight-coloring method, the performance improved drastically in comparison with the reference HPCG. Moreover, ELL achieves a better performance than JAD. The hyperplane method could further increase the performance due to the faster convergence. By further employing the selective caching for ADB and the problem size tuning, the performance finally exceeds 30 GFLOP/s. The vectorization ratios of SpMV and MG become 99.37% and 99.23%, respectively, which are drastically improved from those of the reference implementation.

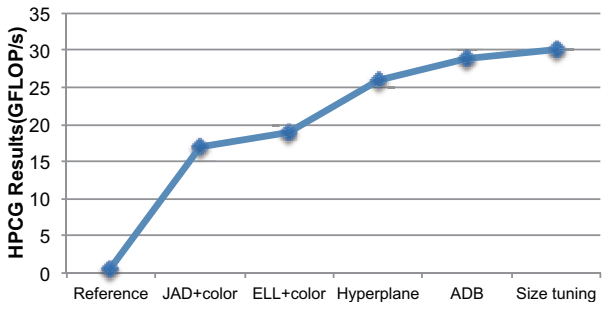


Figure 4: Performance improvement by the optimizations

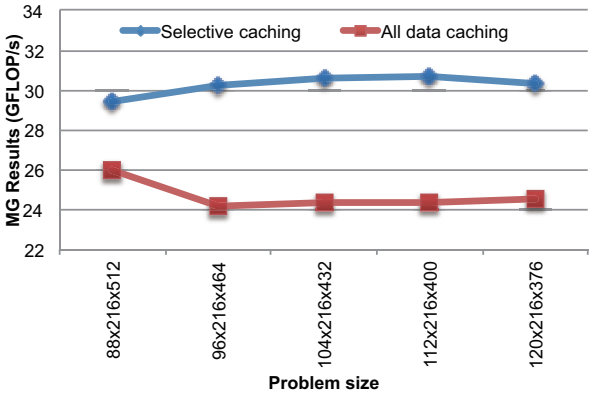


Figure 5: Effects of selective caching and size tuning

Figure 5 shows the effects of the selective caching and the problem size tuning on the single node performance. It is clarified that the selective caching achieves about 30% performance improvement at maximum. Moreover, an appropriate problem size can further improve the performance because of efficient use of ADB and efficient vector calculations.

Figure 6 shows the scalability and efficiency. From the results, it is clarified that the performance scales almost linearly according to the number of nodes. SX-ACE can achieve a high efficiency of 11.4% in the case of using 512 nodes.

5. CONCLUSIONS

To exploit high potential of SX-ACE on HPCG, this poster introduces optimization techniques such as various data packing, the eight-coloring method, the hyperplane method, the selective data caching, and the problem size tuning. By employing the most effective combination of them, SX-ACE successfully achieves the highest efficiency of 11.4% among all of the supercomputers in the latest HPCG ranking.

6. ACKNOWLEDGMENTS

This research was partially supported by JST CREST "An Evolutionary Approach to Construction of a Software Development Environment for Massively-Parallel Heterogeneous Systems".

7. REFERENCES

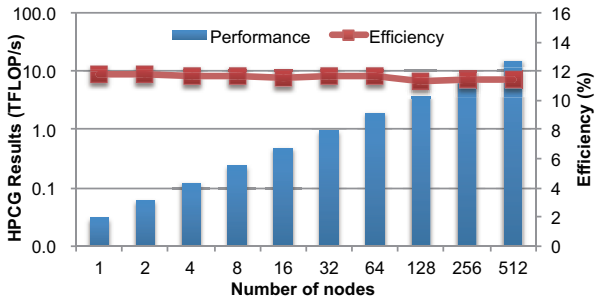


Figure 6: Scalability of HPCG on SX-ACE

[1] HPCG Benchmark. http://www.hpcg-benchmark.org/.

[2] Shintaro Momose, Takashi Hagiwara, Yoko Isobe, and Hiroshi Takahara. The brand-new vector supercomputer, SX-ACE. In *Proceedings of 29th International Supercomputing Conference, ISC 2014, Leipzig, Germany, June 22-26, 2014.*, pages 199–214, 2014.

[3] T. Washio, K. Maruyama, T. Osoda, S. Doi, and F. Shimizu. Efficient implementations of block sparse matrix operations on shared memory vector machines. In *SNA 2000. The fourth international conference on supercomputing in nuclear applications*, 2000.

[4] Eduardo F. D'Azevedo, Mark R. Fahey, and Richard T. Mills. Vectorized sparse matrix multiply for compressed row storage format. In *Proceedings of the 5th International Conference on Computational Science - Volume Part I*, ICCS'05, pages 99–106, Berlin, Heidelberg, 2005. Springer-Verlag.

[5] Roger G. Grimes, David R. Kincaid, and David M. Young. ITPACK 2.0 user's guide. Report CNA-150, August 1979.

[6] Nathan Bell and Michael Garland. Efficient sparse matrix-vector multiplication on CUDA. NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, December 2008.

[7] T. Iwashita, H. Nakashima, and Yasuhito Takahashi. Algebraic block multi-color ordering method for parallel multi-threaded sparse triangular solver in iccg method. In *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 474–483, May 2012.

[8] Y. Oyanagi. Hyperplane vs. multicolor vectorization of incomplete lu preconditioning for the wilson fermion on the lattice. *J. Inf. Process.*, 11(1):32–37, January 1987.

[9] Seiji Fujino, Masatake Mori, and Toshiki Takeuchi. Performance of hyperplane ordering on vector computers. *Journal of Computational and Applied Mathematics*, 38(1-3):125–136, 1991.