Fast computation of double precision sparse matrix in BCRS and DD vector product using AVX2

Toshiaki Hishinuma[1], Akihiro Fujii[1], Teruo Tanaka[1], and Hidehiko Hasegawa[2]

[1] Major of Informatics, Kogakuin University, Tokyo, Japan
[2] Faculty of Lib., Info. & Media Sci., University of Tsukuba, Tsukuba, Japan
em13015@ns.kogakuin.ac.jp

Abstract. In a double precision sparse matrix and a double-double precision (DD) vector product (DD-SpMV) : $\boldsymbol{y} = \mathbf{A}\boldsymbol{x}$ in CRS format using AVX2, the non-contiguous memory access, processing for the remainder and the summation elements in the SIMD register are factors that affect performance. BCRS format with block size equal to the SIMD register's length or its multiples reduces performance degradation. However, since BCRS consists of zero-elements, it may result in increased computations. We accelerated DD-SpMV in BCRS using AVX2.

Keywords: SpMV, BCRS format, High precision arithmetic

1 Introduction

DD arithmetic is a high precision arithmetic system[1]. It uses two double precision variables to implement one quadruple precision variable.

Intel AVX2 computes four double precision variables simultaneously. In the compressed row storage (CRS) format [2], SpMV using AVX2 requires processing for the remainder (1, 2, 3) in each row. In SpMV in CRS using AVX2, the loading of $\boldsymbol{x}$ is non-contiguous. Therefore, SpMV in CRS must relocate $\boldsymbol{x}$ for using AVX2. In storing $\boldsymbol{y}$, DD-SpMV using AVX2 requires the summation of the elements in the SIMD register in each row. Processing for the remainder and summation of elements in the SIMD register are factors that affect performance.

The block CRS (BCRS) format[2] partitions the matrix A into r $\times$ c small and dense submatrices (called blocks), which may consists of some zero-elements. DD-SpMV using AVX2 in BCRS format can reduce factors that affect performance in DD-SpMV in CRS format. However, since BCRS consists of zero-elements, it may result in increased computations.

We implement BCRS 1x4 and BCRS 4x1. BCRS 1x4 reduces processing for the remainder and improves memory access. BCRS 4x1 reduces processing for the remainder and summation of elements in the SIMD register and improves memory access. We accelerate DD-SpMV in BCRS using AVX2 with the block size optimized for AVX2.

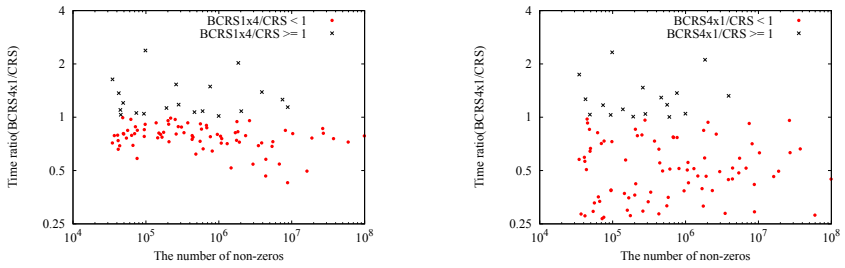

Fig. 1. Elapsed time ratio of DD-SpMV using BCRS format (left : BCRS 1x4 compared to CRS, right : BCRS 4x1 compared to CRS

2 Experimental results

The CPU is a 4-core 8-thread Intel Core i7 4770 3.4 GHz. The size of memory is 16 GB. OS is CentOS 6.4 and the compiler is an Intel C/C++ compiler 13.1.0. Compiler options -O3, -xCORE-AVX2, -openmp, and -fp-model precise are used. We used a set of 100 sparse matrices that obtained from the University of Florida Sparse Matrix Collection.

Figure 1 shows elapsed time ratio of DD-SpMV in BCRS using AVX2. The elapsed time of DD-SpMV in BCRS 4x1 is 0.4-2.4 times that of DD-SpMV in CRS. For 80 matrices, the elapsed time of BCRS 1x4 is less than that of CRS. However, for 20 matrices, since the number of non-zeros is small (less than 10^7), the elapsed time of BCRS 1x4 is more than that of CRS. Elapsed time of DD-SpMV in BCRS 4x1 is 0.3-2.3 times that of DD-SpMV in CRS. For 83 matrices, the elapsed time of BCRS 4x1 is less than that of CRS. However, for 17 matrices, since the number of non-zeros is small (less than 3.0×10^6), the elapsed time of BCRS 4x1 is more than that of CRS. The effect of improving memory access for small size matrices is small. However, DD-SpMV in BCRS yields good performance and is suitable for large size matrices.

3 Conclusion

We accelerated the double precision sparse matrix in BCRS format and DD vector product using AVX2. The best storage format is BCRS 4x1, which reduces factors that affect performance in DD-SpMV in CRS format. The effect of BCRS 4x1 is good and BCRS 4x1 is effective for large size matrices.

References

1. Bailey, D ,H.: High-Precision Floating-Point Arithmetic in Scientific Computation, computing in Science and Engineering, pp. 54-61 (2005).
2. Barrett, R., et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM pp. 57-65 (1994)