ITBL Community

DD-AVX Library

SC2017

Hidehiko Hasegawa[†,††], Toshiaki Hishinuma[†], and Teruo Tanaka[††]

[†]University of Tsukuba [††]Kogakuin University

Introduction

- High precision arithmetic is effective for reducing rounding errors and improving the convergence of Krylov methods[1].
- DD-AVX Library includes Double-Double precision[2] vector and Sparse Matrix operations optimized for SIMD AVX2[3].
- DD-AVX enables Double and Double-Double Mixed precision arithmetic by using operator and function overloading in C++.

What is DD-AVX Library?

```
#include<dd-avx.hpp>
int main(){
  double alpha = 1.0;
  DD_Scalar beta=5.0, gamma=0.0;
  D_Vector x;
  DD_Vector y;
  D_Matrix A;
  DD_AVX_input(A, “input.mtx”, “bcrs4x1”);
  DD_AVX_vector_calloc(x, A.N);
  DD_AVX_vector_calloc(y, A.N);
  gamma = beta * 5.0 + DD(alpha);
  axpy(-gamma, x, y); //y += -gamma * x
  spmv(A,x,y);
  gamma.print();
}
```

Fig.1 Sample code of DD-AVX

A set of Double precision Sparse Matrix and DD precision vector operations:

- DD-AVX accelerates Double and DD precision arithmetic for SIMD AVX2/AVX/SSE2.
- DD-AVX supports
  - Vector operations : axpy, axpyz, xpay, dot, nrm2, and scale,
  - SpMV and transposed SpMV (Matrix of Double precision),
  - Two Sparse Matrix Storage format (CRS and BCRS4x1).
- A combination of BCRS4x1 and AVX2 can be good performance [3].

EasyUI :

- Argument of functions does not depend on precisions.
- Interface of Functions is the same for D and DD.
- Mixed precision arithmetic is possible without changing the code.
- Arithmetic operators (+, -, *, /) can be used for scalar arithmetic.
- SIMD intrinsics are used for avoiding overhead of C++ calls.
- Users can write code without knowledge of C++ programming language.

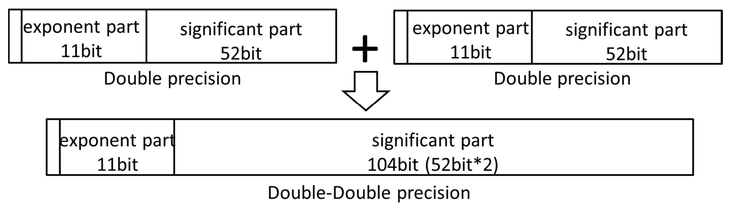

Fig.2 Double-Double precision

Performance of $y_{DD} = A_D x_{DD}$ and $y_{DD} = A_D^T x_{DD}$ using DD-AVX

Intel Core i7 4770K 3.4GHz 4core, 16GB, CentOS 6.4, intel C++ Compiler 13.1.0

Design policy: Precision of Matrix is Double.

- Input Matrices are given in Double precision in many cases.
- The bytes/flops of a product of Sparse Matrix and Vector $y_{DD} = A_D x_{DD}$ becomes small.
- Precision of matrix has small influence for their convergence.

Table 1 bytes/flops of $y = Ax$

	bytes / flops	
$y_D = A_D x_D$	14 (28 bytes / 2 flops)	
$y_{DD} = A_{DD} x_{DD}$	2.26 (52 bytes / 23 flops)	94%
$y_{DD} = A_D x_{DD}$	2.09 (44 bytes / 21 flops)	

DD-AVX 2.0.0 (alpha)

Download DD-AVX library 2.0.0 (alpha)

http://www.slis.tsukuba.ac.jp/~s1530534/DD-AVX.html

Contact: hishinuma@slis.tsukuba.ac.jp

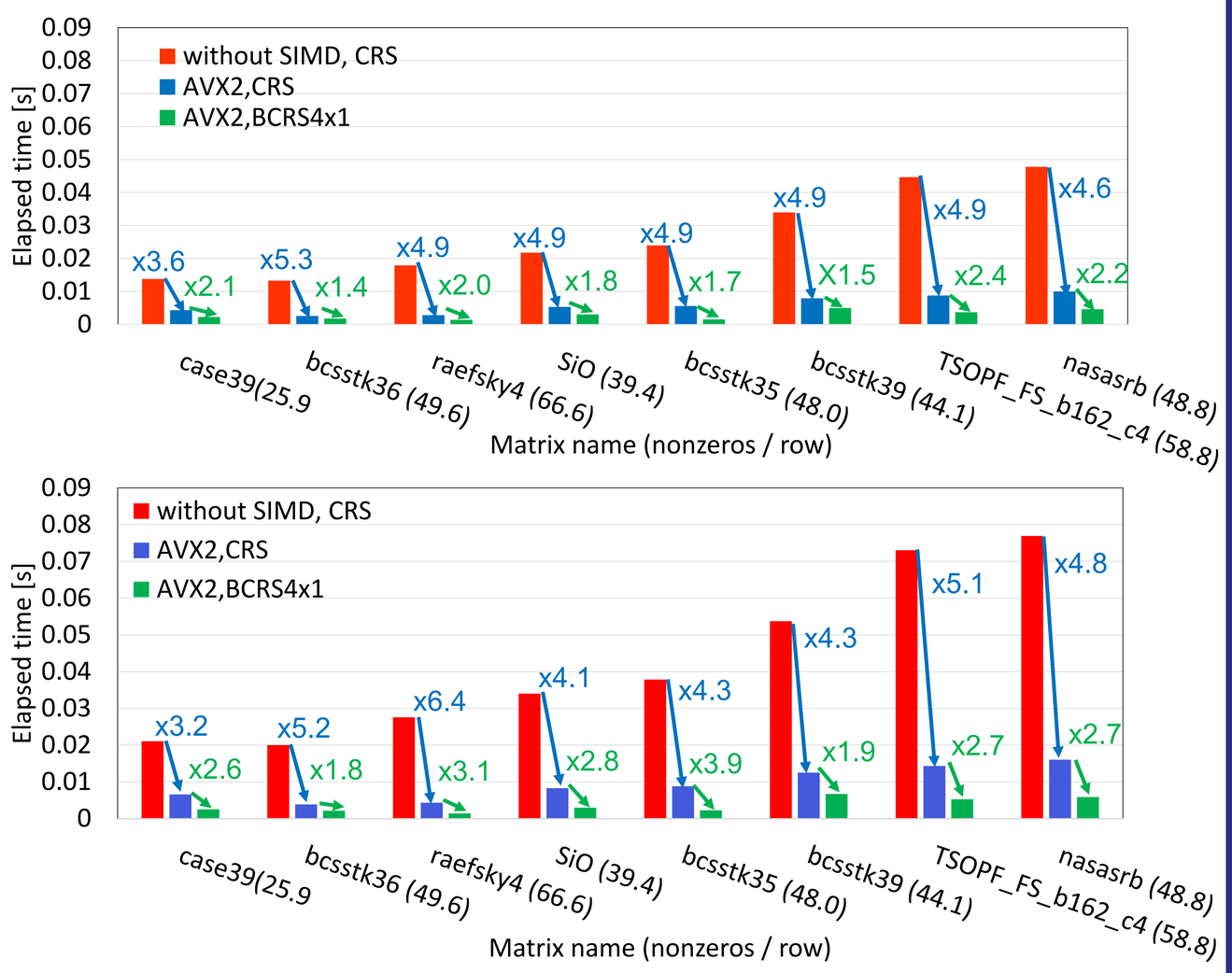

Fig.3 Elapsed time of $y_{DD} = A_D x_{DD}$ (above) and $y_{DD} = A_D^T x_{DD}$ (below)

References

[1] T. Kouya : A Highly Efficient Implementation of Multiple Precision Sparse Matrix-Vector Multiplication and Its Application to Product-type Krylov Subspace Methods, IJNMA, Vol. 7, Issue 2, pp. 107-119 (2012).

[2] Bailey, D, H.: High Precision Floating-Point Arithmetic in Scientific Computation, Computing in Science and Engineering, pp. 54-61 (2005).

[3] T.Hishinuma, et al.: AVX acceleration of DD arithmetic between a sparse matrix and vector, Lecture Notes in Computer Science 8384, pp. 622-631 (2014).

- DD-AVX can accelerate the product of Double precision Sparse Matrix in BCRS4x1 and DD precision Vector by using SIMD AVX2 (x1.4~x3.9 speedup!).
- This makes Krylov Subspace methods more stable with a small extra cost.
- Mixed precision arithmetic will be possible to reduce computation time of High precision Klylov methods.