Accurate and Reproducible Linear Algebra Operations for Many-core Architectures*

Daichi Mukunoki[1], Takeshi Ogita[2], Katsuhisa Ozaki[3]

RIKEN Center for Computational Science[1], Tokyo Woman's Christian University[2], Shibaura Institute of Technology[3]

1. Introduction

This poster presents our accurate and reproducible implementations of linear algebra kernels for many-core architectures. In this study, "reproducible" means an ability to obtain a bit-level identical result every time on the same input data regardless of the computation environment, and "accurate" means that the accuracy of a computation result is higher than that of the standard floating-point operations. As floating-point computations suffer from rounding errors, the result may be inaccurate. Moreover, the result may be different in each computation (i.e., non-reproducible) depending on many factors such as the implementation of the library, the number of threads, use/non-use of atomicAdd and FMA, etc. on parallel computations. Loss of accuracy and reproducibility may become a crucial issue on the debugging of complex codes as well as the reliability of computations. Therefore, as the first step towards accurate and reproducible numerical computations, several projects have been started to develop accurate and reproducible Basic Linear Algebra Subprograms (BLAS) (e.g., ReproBLAS [1] and ExBLAS [2]).

2. Method

To achieve accurate and reproducible results, we utilized the Ozaki scheme, which is an accurate matrix-multiplication method proposed by Ozaki et al [3]. The method consists of the error-free transformation for matrix-multiplication and an accurate summation method. In this method, a matrix-multiplication is transformed into the summation of multiple error-free matrix-multiplications by splitting the input matrices into several split matrices. The method can achieve not only correct-rounding but also a certain accuracy on demand by changing the number of split matrices used in the computation. The accuracy also depends on the absolute range of the floating-point values in the input matrices. Besides, if the summation is computed by some reproducible method, the accurate matrix-multiplication result also achieves reproducibility. In this study, we used the NearSum, a summation algorithm which can achieve correct-rounding, by Rump et al. [4]. The method is applicable for any inner-product based operations such as DOT, GEMV, and sparse matrix-vector multiplication (SpMV). The notable point of this method is that the matrix-multiplications for the split matrices can be computed using standard floating-point operations, i.e., those operations can be computed with DGEMM for double-precision matrices, and it does not matter whether the DGEMM is reproducible or not. Therefore, we can utilize highly optimized BLAS implementations such as Intel MKL and NVIDIA cuBLAS for accurate and reproducible computation. This is a great advantage from the viewpoint of the development cost.

3. Demonstration

We implemented three BLAS routines, DOT ($r = x^T y$), GEMV ($y = Ax$), and GEMM ($C = AB$), with the Ozaki scheme on x86 CPUs and NVIDIA GPUs. Our implementation with the Ozaki scheme, OzBLAS, computes and returns double-precision floating-point values: the

*This research was partially supported by MEXT as "Exploratory Issue on Post-K computer" (Development of verified numerical computations and super high-performance computing environment for extreme researches) and the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number 19K20286.

Table 1: Maximum relative error compared to MPFR (2048-bit) on DOT ($n = 10000$). All the results including the result by MPFR are rounded to double-precision.

Input range (min – max)	7.6E-06 – 2.4E+01	2.0E-08 – 2.6E+06	2.1E-14 – 2.7E+11	1.2E-17 – 6.2E+14
cuBLAS (double)	1.85E-16	1.21E-16	2.47E-16	2.03E-16
OzBLAS (d=3)	0	3.10E-13	1.59E-10	1.55E-08
OzBLAS (d=4)	0	0	1.60E-15	3.91E-14
OzBLAS (d=5)	0	0	0	0

Table 2: The results of DOT ($n = 10000$, input range: 7.6E-06 – 2.4E+01) with various implementations shown in hexadecimal format. The MPFR result is rounded to double-precision.

Implementation	CPU (Xeon E5-2623 v4)	GPU (Titan V)
MPFR (2048-bit)	0x1.33b6d7b84f15cp+7	N/A
cuBLAS (double)	N/A	0x1.33b6d7b84f15bp+7
MKL (double)	0x1.33b6d7b84f15ep+7	N/A
OzBLAS (d=1)	0x1.33b4bbaep+7	0x1.33b4bbaep+7
OzBLAS (d=2)	0x1.33b6d7b83efffp+7	0x1.33b6d7b83efffp+7
OzBLAS (d=3)	0x1.33b6d7b84f15cp+7	0x1.33b6d7b84f15cp+7

interface is compatible with the standard double-precision BLAS. For the computation of split vectors/matrices, they use Intel MKL and NVIDIA cuBLAS internally on CPUs and GPUs, respectively. Table 1 demonstrates the accuracy of the Ozaki scheme. 'd' corresponds the number of split vectors/matrices used in the computation. The accuracy depends on the range of the absolute values in the input value as well as d. Therefore, it can be less accurate than the standard double-precision DOT routine. Table 2 demonstrates the reproducibility and accuracy of the Ozaki scheme on DOT on CPU and GPU. The OzBLAS may be less accurate than Intel MKL and NVIDIA cuBLAS depending on d but always ensures reproducibility between CPU and GPU. The theoretical execution time overhead compared to the standard double-precision is $4d$ times on memory-bound operations such as DOT and GEMV. On compute-bound operations such as GEMM, while it is d^2 times, it can be reduced to $d(d+1)/2$ times with a small accuracy loss. Our GPU implementation on CUDA achieves a comparable performance with the theoretical overhead. In our poster presentation, we will present the performance evaluation results on both CPUs and GPUs including the comparison with existing implementations as well as the details of the implementations and performance optimizations.

References

1. Ahrens P., Nguyen H.D., Demmel J.: ReproBLAS – Reproducible Basic Linear Algebra Sub-programs, `https://bebop.cs.berkeley.edu/reproblas`.
2. Iakymchuk R., Collange S., Defour D., Graillat S.: ExBLAS – Exact BLAS, `https://exblas.lip6.fr`.
3. Ozaki K., Ogita T., Oishi S., Rump S. M.: Error-free transformations of matrix multiplication by using fast routines of matrix multiplication and its applications, Numer. Algorithms, vol. 59, no. 1, pp. 95–118, 2012.
4. Rump S., Ogita T., Oishi S.: Accurate Floating-Point Summation Part II: Sign, K-Fold Faithful and Rounding to Nearest, SIAM Journal on Scientific Computing, Vol. 31, No. 2, pp. 1269–1302, 2009.