A Case Study of the Reproducibility Issues in EigenExa

Roman Iakymchuk
CST/PDC, CSC, KTH Royal Institute of Technology
11428 Stockholm, Sweden
riakymch@kth.se

Imamura Toshiyuki
RIKEN Advanced Institute for Computational Science
Kobe, 650-0047, Japan
imamura.toshiyuki@riken.jp

Stef Graillat
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, 75005 Paris, France
stef.graillat@lip6.fre

Stefano Markidis
CST/PDC, CSC, KTH Royal Institute of Technology
11428 Stockholm, Sweden
markidis@kth.se

Erwin Laure
CST/PDC, CSC, KTH Royal Institute of Technology
11428 Stockholm, Sweden
erwinl@pdc.kth.se

ABSTRACT

In this article, we study the accuracy and reproducibility issues in EigenExa due to the non-associative of floating-point operations, rounding-off errors, dynamic thread scheduling, and different reduction trees, etc. By investigating the EigenExa's algorithmic structure and the corresponding implementation, we find the origins of the non-reproducibility and demonstrate it on the numerical results. As a solution, we, at first, propose to apply the accurate and reproducible parallel reduction from the ExBLAS library. Moreover, we outline possible extensions to ExBLAS that would enhance further the numerical properties of EigenExa.

Keywords

EigenExa, eigenvalue solver, reproducibility, accuracy, superaccumulator, error-free transformation, ExBLAS.

1. INTRODUCTION

In order to facilitate portability of numerical codes, the IEEE-754 arithmetic standard was created in 1985 and then revised in 2008. The standard has led to a considerable increase in the reliability of numerical computations by rigorously specifying the properties of floating-point arithmetic. The standard requires correctly rounded results for the basic arithmetic operations ($+, -, \times, /, \sqrt{}$). It means that the operations are performed as if the result was first computed with an infinite precision and then rounded to the floating-point format.

Although the standard ensures different rounding modes, the round-to-nearest is often used, it does not keep track of the truncated parts of numbers, meaning the floating-point operations are commutative, but non-associative. The non-associativity of floating-point addition occurs when performing addition of numbers with different exponents. It leads to cancellation phenomenon which consist in the elimination of the lowest-order bits of the sum. The non-associativity and the order of operations strongly impact the results of floating-point operations [7]. For instance, the result of summing all elements of a vector differ when the numbers summed in descending or ascending orders. This difference becomes even more noticeable on clusters with thousands of processors that enable dynamic thread scheduling, resources allocation (e.g. for the scalability purposes), various reduction trees, etc. On clusters, it is not only difficult to obtain a result with a certain accuracy, but it is even more difficult to obtain the bit-wise reproducible result from one run to another of the code on the identical input data.

In this article, we investigate the accuracy and reproducibility issues of EigenExa on shared- and distributed-memory architectures. Knowing the sources of these issues, we propose a suitable treatment through the ExBLAS library and its extension. On an example of EigenExa, we aim to provide suitable algorithmic solutions for this type of problems and to develop scalable implementations to guarantee both accuracy and reproducibility.

The paper is organized as follows. The ExBLAS library is described in Section 2. Section 3 presents the EigenExa library and the sources of its non-reproducibility, and introduces our approach to solve this problem. Finally, Section 4 draws the conclusions and outlines the future work.

EASC'16 April 25–29, 2016, Stockholm, Sweden
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

2. EXBLAS LIBRARY

ExBLAS stands for Exact (fast, accurate, and reproducible) Basic Linear Algebra Subprograms [4]. In ExBLAS, we aim at providing new algorithms and implementations for fundamental linear algebra operations – like those included in the BLAS library. We [2] introduced a multi-level approach to compute correctly rounded and reproducible sums. This approach is based on floating-point expansions (FPEs) and superaccumulators. FPEs aim at computing the error which occurred during rounding using FP expansions in conjunction with error-free transformations (EFTs). Thanks to EFTs, when working with the rounding-to-nearest mode, the rounding error of addition and multiplication can be represented as a floating-point number. FPEs represent the result as an unevaluated sum of FP numbers, whose components are ordered by magnitude with minimal overlap to cover a wide range of exponents. Superaccumulators instead

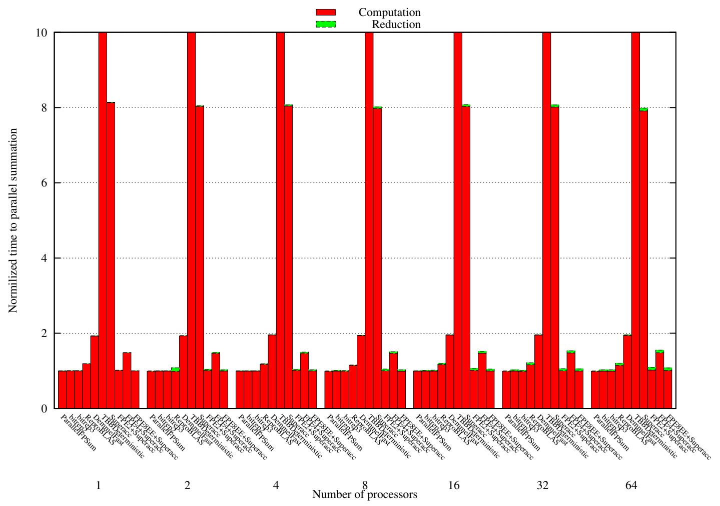

Figure 1: Performance scaling of parallel reduction on the 64 nodes Mesu cluster at UPMC; each node is equipped with two 8−core Intel Xeon E5-4650L (Sandy Bridge) @ 2.6 GHz.

exploit the finite range of representable floating-point numbers by storing every bit of the sum. The superaccumulator covers the range from the minimum representable FP value to the maximum value independently of the sign. We use a superaccumulator of 2098 (emin + emax + mantissa = 1022 + 1023 + 53) bits for double precision FP accumulation.

We provided implementations of the multi-level approach for parallel reduction on a range of architectures: desktop and server CPUs, Intel Xeon Phi co-processors, and both AMD and NVIDIA GPUs. The proposed implementations showed that the numerical reproducibility and bit-perfect accuracy can be achieved at no additional cost for large sums with dynamic ranges of up to 90 orders of magnitude.

Figure 1 compare the performance scaling of the various reduction algorithms on the Mesu cluster. We measure the computation and reduction time of the parallel non-reproducible summation (“ParallelFPSum”), the TBB deterministic summation (“TBBDeterministic”), single-sweep reductions (“bitrep2” and “bitrep3”) from bitrep [1], one-reduction summation (“ReproBLAS”) from ReproBLAS [3], our implementation of their two-reduction algorithm (“DemmelFast”), and our algorithm (“Superacc” with suparaccumulators only and “FPEx+Superacc” that combine superaccumulators and FP of size x with the early-exit technique (EE) [2]). These algorithms compute the local sums on each MPI process. Then, MPI Reduce() or MPI Allreduce() are applied to calculate the final sum. We use two MPI processes per node in our implementations and one MPI process per core in case of ReproBLAS and bitrep.

The number of MPI process varies from 1 to 64, each of them performing the summation of 16M double-precision FP numbers. This dataset size ensure that we fall in the out-of-cache case. For each process, we measure the local summation time, which is colored red, and the MPI reduction time, which is colored green. For the whole range of processors, the execution time of each algorithm is dominated by the local summation time because of the dataset size. In addition, due to the equal distribution of computations among MPI processes, the computation time is roughly equivalent on the whole range of MPI processes, while the reduction time changes according to the number of MPI processes involved. We normalize the total runtime of each algorithm by the total execution time of the parallel FP summation. Since the TBB deterministic sum is roughly 68 times slower than the parallel floating-point summation on one node, we cut this results in order to provided a better view of the rest of the results that do not exceed the slowdown of 10 times, The “bitrep2”, “bitrep3”, “Demmel fast”, and “ReproBLAS” algorithms are 2.9 %, 3.1 %, 95.5 %, and 20.6 %, accordingly, slower than the conventional parallel summation on 64 CPUs. Although “bitrep2” and “bitrep3” show the low performance overhead, their accuracy and reproducibility are not guaranteed, especially for the moderate dynamic ranges. In contrast, our “FPE2 + Superacc” and “FPE8EE + Superacc” deliver both correctly rounded and bit-wise reproducible results with the overhead of 9.2 % and 7.8 %, respectively.

To enhance reproducibility, Intel proposed a “Conditional Numerical Reproducibility” (CNR) in its MKL. However, CNR does not ensure correct rounding and it induces large performance overhead. For instance, for large arrays the MKL’s summation with CNR is $85 - 93$ % slower than both the regular MKL’s and our reproducible summation.

We extended and adapted the multi-level approach to dot product, triangular solver, and matrix-matrix multiplication. However, these routines are only supported on GPUs.

3. TOWARDS REPRODUCIBLE EIGENEXA

EigenExa is a high-performance numerical eigenvalue solver [6, 5]. The EigenExa objective is to achieve an eigenvalue library scalable to operate on future Exascale systems. EigenExa provides functionality for computing all eigenpairs (eigenvalues paired with their respective eigenvectors) for both standard and generalized eigenvalue problems. EigenExa applies both classical and advanced algorithms and enhances the required computation time of its predecessor EigenK for the diagonalization stage.

The development of EigenExa includes the utilization of various parallel programming languages and libraries, encompassing MPI, OpenMP, high-performance BLAS, and SIMD vectorized Fortran90 compiler techniques. In most cases, the EigenExa performance exceeds that of EigenK, ScaLAPACK, and others of the highest-level numerical computation libraries. EigenExa is in operation on many HPC platforms, including the K computer and its Fujitsu PRIMEHPC FX10 commercial variant, various cluster computers using Intel x86 series processors, IBM Blue/Gene Q systems, and the NEC vector computer SX series systems.

We present below the main steps of the EigenExa solver (eigen_sx) for the direct computation of the eigenvalues and eigenvectors of a banded pentadiagonal matrix.

1. Pentadiagonalization of the input matrix by block version of Householder transformations: $Q^TAQ \to B$
2. Computation of the eigenvalues and eigenvectors of a pentadiagonal matrix by the divide- and-conquer method: $By_i = \gamma_i y_i$
3. Back transformation of the eigenvectors: $Qy_i \to x_i$

Since the reproducibility problem comes from the pentadiagonalization step, here we focus on it only. Suppose we have two vectors a_1 and a_2, then we calculate the two consecutive Householder transformations as follows

$$\begin{aligned} H_1 &= I - b_1 u_1 u_1^T, \\ H_1 a_1 &= -t_1 e_1, \\ H_2 a_2 &= c_2, \\ d_2 &= [0; c_2(2:n)], \\ H_2 &= I - b_2 u_2 u_2^T \end{aligned}$$

where

$$\begin{aligned} u_1 &= t_1 e_1 + a_1, \\ t_1 &= sign(norm2(a_1), a_1[0]), \\ b_1 &= 2/norm2(u_1), \\ u_2 &= t_2 e_2 + c_2, \\ t_2 &= sign(norm2(c_2), d_2[1]), \\ b_2 &= 2/norm2(u_2). \end{aligned}$$

Then, we construct matrices $U = [u_1, u_2]$ and C, where C holds $H_2 H_1 = I - UCU$.

In this process, if $[a_1, a_2]$ are very close or almost linearly dependant, d_2 equals zero in the exact calculation. But, numerically d_2 is polluted by the round-off errors and the approximated result of d_2 is not the zero vector.

This situation depends on any possible hardware and software differences, such as the number of threads used and their scheduling, compiler optimization, MPI implementation, reduction trees, etc. If d_2 is not zero, then H_2 is not identical and, therefore, the other matrices $H_i(N/2 - 1 \leq i > 1)$ are not correctly computed. Thus, the computed result is drastically different.

In order to provide the evidence of these accuracy and reproducibility problems in EigenExa, we gather the log files on the EigenExa execution. These log files contain the pentadiagonal matrix represented by three arrays $[D, E, F]$. We expect that these three arrays will be relatively close to each other with respect to the accumulated round-off errors. However, we observe that some elements of F are completely different: one is positive while the other is negative. Below we demonstrate this difference on the highlighted elements of the arrays F_1 and F_4 computed with 1 and 4 processes, accordingly.

$$F_1 = \begin{bmatrix} 0.0000E+0 & 0.0000E+0 & \mathbf{1.2398123} \\ \mathbf{-1.3029823} & \mathbf{-1.5637168} & \mathbf{-1.3334236E-015} \\ 1.4429192 & 1.5458365 & -1.0235326 \end{bmatrix}$$

$$F_4 = \begin{bmatrix} 0.0000E+0 & 0.0000E+0 & \mathbf{-1.8012368} \\ \mathbf{1.1340358} & \mathbf{1.6641588} & \mathbf{1.2585248E-015} \\ 1.4429192 & 1.5458365 & -1.0235326 \end{bmatrix}$$

This result shows that there is a need to replace the dot product and, especially, the parallel reduction operations in the the block-version Householder transformation by the corresponding routines from ExBLAS. This will provide us with the control over the rounding errors and will guarantee the correctly rounded and reproducible results independently of the number of threads and processes, the threads scheduling policies, etc.

4. CONCLUSIONS AND FUTURE WORK

In this paper we studied the accuracy and reproducibility problems of EigenExa. We indicated that these problems originated by the non-associativity of floating-point operations, accumulated round-off errors, dynamic thread scheduling, etc. The sources of these problems are in the collective operations such as MPI reduction and the linear algebra operations provided by the BLAS library. We draw a strategy to ensure the reproducibility and correct rounding of the EigenExa results by plugging in the ExBLAS routines, which already guarantee exact computations and low performance overhead for memory-bound operations such as parallel reduction.

We plan to conduct numerical experiments with EigenExa using the ExBLAS parallel reduction in the near future. Moreover, we foresee to extend the functionality of the ExBLAS library to support more routines as well as large scale computations. This would enable rebasing the major part of the EigenExa computations on top of ExBLAS.

5. REFERENCES

[1] A. Arteaga, O. Fuhrer, and T. Hoefler. Designing bit-reproducible portable high-performance applications. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14, pages 1235–1244, Washington, DC, USA, 2014. IEEE Computer Society.

[2] S. Collange, D. Defour, S. Graillat, and R. Iakymchuk. Numerical Reproducibility for the Parallel Reduction on Multi- and Many-Core Architectures. *Parallel Computing*, 49:83–97, 2015.

[3] J. Demmel and H. D. Nguyen. Parallel Reproducible Summation. *IEEE Transactions on Computers*, 64(7):2060–2070, 2015.

[4] R. Iakymchuk, S. Collange, D. Defour, and S. Graillat. ExBLAS: Reproducible and Accurate BLAS Library. In *Proceedings of the Numerical Reproducibility at Exascale (NRE2015) workshop held as part of the Supercomputing Conference (SC15). Austin, TX, USA, November 15-20, 2015*, Oct. 2015.

[5] T. Imamura, Y. Hirota, T. Fukaya, S. Yamada, and M. Machida. EigenExa: high per- formance dense eigensolver, present and future. In *Proceedings of the 8th International Workshop on Parallel Matrix Algorithms and Applications (PMAA14), Lugano, Switzerland, July 2-4, 2014*.

[6] T. Imamura, S. Yamada, and M. Machida. Development of a High Performance Eigensolver on the Peta-Scale Next Generation Supercomputer System. *Progress in Nuclear Science and Technology, the Atomic Energy Society of Japan*, 2:643–650, 2011.

[7] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, 2010.