Numerical Reproducibility based on Minimal-Precision Validation

Toshiyuki Imamura
Daichi Mukunoki
RIKEN Center for Computational Science (R-CCS)
Kobe, Japan

Roman Iakymchuk
Fabienne Jézéquel
Stef Graillat
Sorbonne University, CNRS, LIP6
Paris, France

I. Introduction

In general, *reproducibility* refers to a capability of obtaining the identical result, but it often means "re-playability" or "re-traceability." In computational science, reproducibility is considered from several view points depending on the context and demand. For instance, the following definition of reproducibility is actually desired: **bit-level reproducibility** *is the capability to reproduce the bit-wise identical result with the same input on any hardware/software configuration. But, the internal procedure of the computation is not necessarily required to be identical.* To establish this bit-level reproducibility, several efforts have been conducted so far. For example, in linear algebra, there are some implementations of Basic Linear Algebra Subprograms (BLAS) such as ReproBLAS [14] and ExBLAS [7]. However, most such existing efforts rely on a special mechanism designed for each mathematical problem. As far as we know, no general approach for any floating-point computation has been proposed yet. We consider that, in terms of the development cost, it is non-realistic to support bit-level reproducibility on all floating-point computations through such existing approaches.

In this study, we define the notion of **weak numerical reproducibility**, i.e. the reproducibility, (up to a high probability) of the computation result with a certain accuracy demanded by the user. The underlying numerical validation is performed using a statistical approach that estimates with a high probability the number of correct digits in the computation result. This weak numerical reproducibility can be established through the extension of our minimal-precision computing scheme [10], which validates the accuracy (demanded by the user) of the result through the minimal-precision use.

II. Minimal-precision computing

The minimal-precision computing has been proposed for realizing high-performance and energy-efficient as well as reliable (accurate, reproducible, and validated) computations. It is a systematic approach combining hardware, software, and algorithmic approaches internally. Specifically, our system combines: i) a precision-tuning method [3], [4], [8] based on Discrete Stochastic Arithmetic (DSA) [13], ii) arbitrary-precision arithmetic libraries, iii) fast and accurate numerical libraries, and iv) Field-Programmable Gate Array (FPGA) with High-Level Synthesis (HLS). Thus, our approach provides the following features:

1) *Reliable*: Precision Tuning is processed to obtain the accuracy requested by the user based on a numerical validation method. It can meet the demands for accurate and reproducible computation.
2) *General*: the scheme is applicable to any floating-point computation. It contributes to low development costs and sustainability such as easy maintenance and system portability.
3) *Comprehensive*: we propose a complete system that emerges from Precision Tuning to the execution of the tuned code, combining heterogeneous hardware and hierarchical software stack.
4) *High-performance*: our scheme considers utilizing fast numerical libraries and accelerators (FPGAs and GPUs).
5) *Energy-efficient*: through the minimal-precision as well as energy efficient hardware acceleration with FPGA and GPU
6) *Realistic*: our system can be realized by combining available technologies.

Figure 1 presents the minimal-precision computing system with the software and hardware stack, and its workflow. The system is a general scheme for any floating-point computations, but we currently target linear algebra computations as the first step (thus, we explain a case for linear algebra computations). Below we explain the total procedure and some key technologies in the system.

- **Setup of inputs**: The system accepts as input C code using IEEE 754-2008 floating-point numbers. Also, the requested accuracy of the computation result is given by the user. Then, all the floating-point variables and operations in the code can be translated to those of the GNU Multiple Precision Floating-Point Reliably (MPFR) [2] – a C library for multiple (arbitrary) precision floating-point computations on CPUs. MPFR is used for precision-tuning with arbitrary-precision and executing operations on FPGAs or CPUs as FPGAs are not necessarily required. For linear algebra operations, we can utilize MPLAPACK [11], a multi-precision Linear Algebra PACKage (LAPACK) including BLAS based on some high-precision arithmetic libraries including MPFR, and similar high precision or reproducible numerical libraries such as QPEigenK and QPEigenG [5], ExBLAS [7], and OzBLAS [9].
- **Precision Tuning**: The precision-tuner determines the optimal

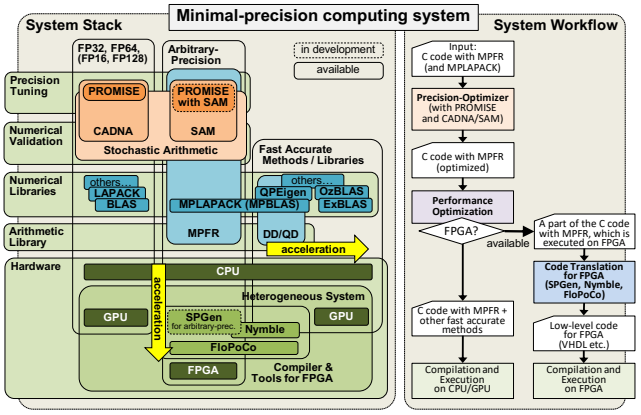

Fig. 1. System overview of minimal-precision computing system

floating-point precisions for all variables in the code, functions, or code segments, which are needed to achieve the computation result with the requested accuracy. Tuning is performed by comparing with a computation result validated by DSA. Thus, the optimized code is reliable. Simply speaking, DSA estimates the rounding errors of floating-point operations with the guarantee of 95% by executing the same code three times with random-rounding (randomly round-down or -up). Then, these three results are used to estimate the number of correct digits. It is a general scheme applicable to any floating-point operation: it requires no special numerical methods and needs only few code modifications that can be automatized. Besides, it can be performed at a reasonable cost in terms of both performance and development cost compared to the other numerical verification or validation methods.

- **Acceleration**: Before executing the tuned-code, if it is possible to speed up some portions of the code with some fast computation methods (including GPU acceleration), these parts are replaced with them. The method must be at least as accurate as that of the required-precision. We may be able to use hardware-native floating-point operations (e.g., FP16/FP32/FP64), some fast high-precision arithmetic libraries, and some accurate numerical libraries. Additionally, we can utilize FPGAs. They enable us to implement and perform arbitrary-precision floating-point operations: FPGAs realize the ultimate minimal-precision computing and achieve better performance and energy-efficiency than software implementations on general processors. Owing to the HLS technology, we can use FPGAs with existing programming languages such as C/C++ and OpenCL [6], [12].

III. Weak numerical reproducibility on minimal-precision computing

The minimal-precision computing system can be considered from a user perspective as a black box where the user provides input parameters, including the desired accuracy, and invokes an algorithm. Under the hood, we may select different paths for execution due to the fact that different working precisions as well as different hardware devices may be used either to speed up computations and/or ensure energy-efficiency. Under any circumstances, scenarios, execution paths, we make sure that the weak numerical reproducibility is guaranteed with the defined input accuracy requirements by using all the tools and features of the system.

On the above system, we can guarantee weak numerical reproducibility by validating the requested accuracy of the computation demanded by the user. Here, if the computation method can achieve the required result, any methods, any computation environments, and any computation conditions can be accepted. We no longer need to develop some reproducible variant(s) for each computation method or mathematical problem. On the other hand, some accurate computation methods may be required to achieve the requested accuracy. However, in general, it is easier to develop accurate methods (not necessarily reproducible) than reproducible methods. Comparing with re-playable and re-traceable methods, it is easier to adapt to different (parallel) architectures. Besides, existing methods and software for ensuring bit-level reproducibility are still able to contribute: for instance, to skip the validation of some parts of algorithms and/or to accelerate the accurate computations needed to ensure the demanded accuracy, if such method relies on some accurate method.

IV. Summary

We proposed a new concept of weak numerical reproducibility and a systematic approach for it with a support of minimal-precision tuning and validation. We expect that the concept of weak numerical reproducibility covers most of the demands for reproducibility in computational sciences. Besides, the minimal-precision computing system can address the demands for accuracy, high-performance, and energy efficient computation as well; especially, when the system can be realized with FPGAs. Although the proposed system is still in development, some key components are already available, and many of them have been developed by us.

Acknowledgment

The authors would like to thank Dr. Kentaro Sano and Dr. Yiyu Tan from RIKEN CCS for their technical supports, and Prof. Taisuke Boku and Dr. Norihisa Fujita from CCS, University of Tsukuba for their fruitful discussions. This research was partially supported by the European Unions Horizon 2020 research, innovation programme under the Marie Skłodowska-Curie grant agreement via the Robust project No. 842528, the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant No. 19K20286, and Multidisciplinary Cooperative Research Program in CCS, University of Tsukuba.

References

[1] F. de Dinechin and B. Pasca, Designing Custom Arithmetic Data Paths with FloPoCo. IEEE Design Test of Computers 28, 4, 18-27, 2011.

[2] L. Fousse, G. Hanrot, V. Lefévre, P. Pélissier, and P. Zimmermann, MPFR: A Multiple-precision Binary Floating-point Library with Correct Rounding. ACM Trans. Math. Softw. 33, 2 (2007).

[3] S. Graillat, F. Jézéquel, R. Picot, F. Févotte, and B. Lathuilière., Auto-tuning for floating-point precision with Discrete Stochastic Arithmetic. Journal of Computational Science (2019). (accepted).

[4] S. Graillat, F. Jézéquel, S. Wang, and Y. Zhu, Stochastic Arithmetic in Multiprecision. Mathematics in Computer Science 5, 4 (2011), 359-375.

[5] Y. Hirota, S. Yamada, T. Imamura, N. Sasa, and M. Machida, Performance of quadruple precision eigenvalue solver libraries QPEigenK and QPEigenG on the K computer. HPC in Asia Poster, International Supercomputing Conference (ISC16) (2016)

[6] J. Huthmann, B. Liebig, J. Oppermann, and A. Koch. Hardware/software co-compilation with the Nymble system. In Proc. 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC). (2013), 1-8

[7] R. Iakymchuk, S. Collange, D. Defour, and S. Graillat. ExBLAS: Reproducible and Accurate BLAS Library. In Proc. Numerical Reproducibility at Exascale (NRE2015) at SC15, (2015)

[8] F. Jézéquel and J.-M. Chesneaux. CADNA: a library for estimating round-off error propagation. Comput. Phys. Commun. 178, 12 (2008), 933-955.

[9] D. Mukunoki, T. Ogita, and K. Ozaki., Accurate and Reproducible BLAS Routines with Ozaki Scheme for Many-core Architectures. In Proc. International Conference on Parallel Processing and Applied Mathematics (PPAM2019). (accepted).

[10] D. Mukunoki, T. Imamura, Y. Tan, A. Koshiba, J. Huthmann, K. Sano, F. Jézéquel, S. Graillat, R. Iakymchuk, N. Fujita, and T. Boku, Minimumal-Precision Computing for High-Performance, Energy-Efficient, and Reliable Computations, International Conference on High Performance Computing, Networking, Storage and Analysis (SC19) (poster submitted).

[11] M. Nakata. The MPACK (MBLAS/MLAPACK); a multiple precision arithmetic version of BLAS and LAPACK (2010). http://mplapack.sourceforge.net.

[12] K. Sano, H. Suzuki, R. Ito, T. Ueno, and S. Yamamoto, Stream Processor Generator for HPC to Embedded Applications on FPGA-based System Platform. In Proc. First International Workshop on FPGAs for Software Programmers (FSP 2014) (2014), 43-48.

[13] J. Vignes, Discrete Stochastic Arithmetic for Validating Results of Numerical Software. Numerical Algorithms 37, 1 (2004), 377-390

[14] J. Demmel, P. Ahrens, H. D. Nguyen, Efficient Reproducible Floating Point Summation and BLAS, Tech. Rep. UCB/EECS-2016-121, EECS Department, University of California, Berkeley (2016).