Optimizing Precision for High-Performance, Robust, and Energy-Efficient Computations

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

Daichi Mukunoki
Toshiyuki Imamura
Yiyu Tan
Atsushi Koshiba
Jens Huthmann
Kentaro Sano
RIKEN Center for Computational Science, Japan

Norihisa Fujita
Taisuke Boku
Center for Computational Sciences, University of Tsukuba, Japan

Introduction. In numerical computations, precision of floating-point computations is a key factor to determine the performance (speed and energy-efficiency) as well as the reliability (accuracy and reproducibility). However, precision generally plays a contrary role for both. Therefore, the ultimate concept for maximizing both at the same time is the minimal-precision computing through precision-tuning, which adjusts the optimal precision for each operation and data. Several studies have been already conducted for it so far (e.g. [1, 9]), but the scope of those studies is limited to the precision-tuning alone. Our project aims to propose a broader concept of the *minimal-precision computing system* with precision-tuning, involving both hardware and software stack.

Minimal-precision computing system. The proposed system [7] combines (1) a precision-tuning method, (2) arbitrary-precision arithmetic libraries, (3) fast and accurate numerical libraries, and (4) heterogeneous architectures with Field-Programmable Gate Array (FPGA). We explain the overall procedure below.

(1) We target IEEE-754 2008 floating-point numbers. An input C code and a requested accuracy are given by the user. We assume that the floating-point variables and operations in the code are defined using the GNU Multiple Precision Floating-Point Reliable (MPFR) library [2]. For codes using FP32/FP64, we can also rely upon MPFR or MPFR-nize them.
(2) The precision-tuner determines the optimal floating-point precisions for all variables in the code, which are needed to achieve the computation result with the requested accuracy. Tuning is performed by comparing with a result validated by Discrete Stochastic Arithmetic (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, the common digits in the three results are assumed to be a reliable result. It is a general scheme applicable for any floating-point operations: no special algorithms and no code modification are needed. We propose to use two DSA libraries, namely CADNA [6] and SAM [4], as well as a precision tuner called PROMISE [3].
(3) The tuned-code (with MPFR) proceeds to the performance optimization phase (and execution). At this stage, if possible to speed up some portions of the code with some fast computation methods (including GPU acceleration), those 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), fast high-precision arithmetic libraries, and accurate numerical libraries. For instance, ExBLAS [5] and OzBLAS [8] libraries for accurate and reproducible BLAS. We assume that this step is processed manually for now, but we plan to automate or assist it.
(4) Another possibility for performance improvement is utilizing FPGA. FPGA enables us to implement and perform arbitrary-precision floating-point operations: it realizes the ultimate minimal-precision computing and achieves better performance and energy-efficiency than software implementations on general processors. Owing to the High-Level Synthesis (HLS) technology, we can use FPGA through existing programming languages such as C/C++ and OpenCL. As a target platform, we plan to utilize the Cygnus supercomputer at the University of Tsukuba that is equipped with GPUs and FPGAs.

Conclusion. We proposed a new systematic approach for minimal-precision computations. This approach is high-performant, robust, energy-efficient, general, comprehensive, and realistic. Although the system is still in development, this presentation showed that the system could be constructed by combining already available in-house technologies as well as extending them.

Acknowledgment: This work was partially supported by EU MSCA-IF Robust Grant No. 842528 and JSPS KAKENHI Grant No. 19K20286.

References.

[1] F. Févotte and B. Lathuilière. 2019. Debugging and optimization of HPC programs in mixed precision with the Verrou tool. *hal-02044101* (2019).
[2] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. 2007. 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. 2019. Auto-tuning for floating-point precision with Discrete Stochastic Arithmetic. *Journal of Computational Science.* 36 (2019), 101017.
[4] S. Graillat, F. Jézéquel, S. Wang, and Y. Zhu. 2011. Stochastic Arithmetic in Multiprecision. *Mathematics in Computer Science.* 5, 4 (2011), 359–375.
[5] R. Iakymchuk, S. Collange, D. Defour, and S. Graillat. 2015. ExBLAS: Reproducible and Accurate BLAS Library. In *Proc. NRE2015 at SC'15.*
[6] F. Jézéquel and J.-M. Chesneaux. 2008. CADNA: a library for estimating round-off error propagation. *Comput. Phys. Commun.* 178, 12 (2008), 933–955.
[7] D. Mukunoki, T. Imamura, Y. Tan, A. Koshiba, J. Huthmann, K. Sano, F. Jézéquel, S. Graillat, R. Iakymchuk, N. Fujita, and T. Boku. 2019. Minimal-Precision Computing for High-Performance, Energy-Efficient, and Reliable Computations. Research Posters at SC'19. (2019). (accepted).
[8] D. Mukunoki, T. Ogita, and K. Ozaki. 2019. Accurate and Reproducible BLAS Routines with Ozaki Scheme for Many-core Architectures. In *Proc. PPAM2019.* (accepted).
[9] C. Rubio-González, Cuong Nguyen, Hong Diep Nguyen, J. Demmel, W. Kahan, K. Sen, D. H. Bailey, C. Iancu, and D. Hough. 2013. Precimonious: Tuning assistant for floating-point precision. In *Proc. SC '13.* 1–12.