Minimal-Precision Computing for High-Performance, Energy-Efficient, and Reliable Computations

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

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

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

1 INTRODUCTION

In numerical computations, the 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, the 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., [12] [2]), but the scope of those studies is limited to the precision-tuning alone.

Our project aims to propose a more broad concept of the minimal-precision computing system with precision-tuning, involving both hardware and software stack. Specifically, our system combines (1) a precision-tuning method based on Discrete Stochastic Arithmetic (DSA) [14], (2) arbitrary-precision arithmetic libraries, (3) fast and accurate numerical libraries, and (4) Field-Programmable Gate Array (FPGA) with High-Level Synthesis (HLS). As a result, our approach aims to achieve the following goals:

- **High-performance**: performance can be improved through the minimal-precision as well as fast numerical libraries and accelerators (FPGA and GPU)
- **Energy-efficient**: through the minimal-precision as well as energy efficient hardware acceleration with FPGA and GPU
- **Reliable**: to ensure the requested accuracy, the precision- tuning is processed based on numerical validation, guaranteeing also reproducibility
- **General**: our scheme is applicable for any floating-point computations. It contributes to low development cost and sustainability (easy maintenance and system portability)
- **Comprehensive**: we propose a total system from the precision-tuning to the execution of the tuned code, combining heterogeneous hardware and hierarchical software stack
- **Realistic**: our system can be realized by combining available in-house technologies

2 PROPOSED APPROACH

Our approach targets a heterogeneous system equipped with FPGAs (and GPUs as an option). Figure 1 shows the system stack and workflow. The below explains the total procedure and some key technologies in the system. The system is a general scheme for any floating-point computations, but we currently target on linear algebra computations as the first step.

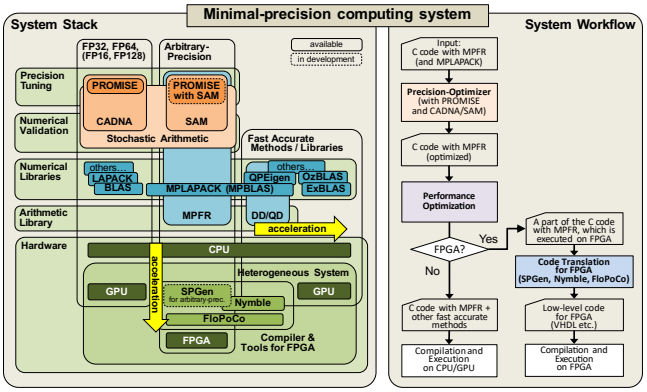

Figure 1: System overview

(1) We target IEEE-754 2008 floating-point numbers. An input C code and 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) [3] – a C library for multiple (arbitrary) precision floating-point computations on CPUs. For codes using FP32/FP64, we can also rely upon MPFR or MPFR-nize them. For linear algebra operations, we can utilize MPLAPACK [11] – a multi-precision Linear Algebra PACKage (LAPACK) including Basic Linear Algebra Subprograms (BLAS) based on some high-precision arithmetic libraries including MPFR.

(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 validated computation result 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, 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

*All authors contributed equally to this research.

needed. Besides, it can be performed at a reasonable cost in terms of both performance and development compared to the other numerical verification or validation methods.

(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), some fast high-precision arithmetic libraries, and some accurate numerical libraries. We assume that this step is processed by hand for now, but we plan to develop some tool 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 HLS technology, we can use FPGA through existing programming languages such as C/C++ and OpenCL. In our scheme, the portion of the code we want to execute on FPGA is compiled with some HLS compiler and executed on FPGA. The compiler can utilize some tools to generate arbitrary-precision floating-point operators on FPGA such as FloPoCo [1].

3 OUR CONTRIBUTIONS

The proposed system can be constructed by combining available technologies, and many of them are developed by us. Below, we list some of our contributions: studies, projects, and tools.

- **DSA libraries**: CADNA (Control of Accuracy and Debugging for Numerical Applications) [9] and SAM (Stochastic Arithmetic in Multiprecision) [5] have been developed; both libraries are developed at Sorbonne University (SU). CADNA is a DSA library for FP16/FP32/FP64/FP128 (for the moment it supports FP16 on GPU for CUDA codes and FP128 on CPU for C/C++ codes). It supports Fortran and C/C++ codes with OpenMP and MPI. SAM is a DSA library for arbitrary-precision with MPFR.
- **Precision-tuner**: PROMISE (PRecision OptiMISE) [4] is a precision-tuner for C/C++ codes developed at SU. It optimizes the precision of variables in codes by referencing the validated computation result with CADNA. We are currently going to extend it for supporting arbitrary-precision with SAM.
- **Fast and accurate numerical libraries**: QPEigen [6] is a quadruple-precision eigen solver based on double-double arithmetic for CPUs developed at JAEA and RIKEN. It relies upon a quadruple-precision BLAS (QPBLAS). ExBLAS [8] and OzBLAS [10] are accurate and reproducible BLAS implementations for CPUs and GPUs developed at SU and Tokyo Woman's Christian University, respectively. These libraries can be used to accelerate accurate linear algebra computations.
- **HLS compilers for FPGA**: Nymble [7], developed at TU Darmstadt and RIKEN, is a compiler which accepts C codes and supports arbitrary-precision floating-point. SPGen (Stream Processor Generator) [13] is another compiler developed at RIKEN, which uses a data-flow graph representation suitable for FPGA. It currently supports FP32 only, but we will extend it to support arbitrary-precision floating-point.
- **CPU-GPU-FPGA system**: Cygnus, installed in University of Tsukuba, is the world first supercomputer system equipped with both GPUs (4x Tesla V100) and FPGAs (2x Stratix 10). This project utilizes the system as a target platform.

4 CONCLUSION

We proposed a new systematic approach for minimal-precision computations. This approach is reliable, general, comprehensive, high-performance, energy-efficient, and realistic. Although the proposed system is still in development, this presentation showed that the system could be constructed by combining already available (developed) in-house technologies as well as extending them. Our ongoing step is to demonstrate the system on a small application.

ACKNOWLEDGMENTS

This research was partially supported by the European Union's 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. 2011. Designing Custom Arithmetic Data Paths with FloPoCo. *IEEE Design Test of Computers* 28, 4 (2011), 18–27.
[2] F. Févotte and B. Lathuilière. 2019. Debugging and optimization of HPC programs in mixed precision with the Verrou tool. *hal-02044101* (2019).
[3] 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).
[4] 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* (2019). (accepted).
[5] S. Graillat, F. Jézéquel, S. Wang, and Y. Zhu. 2011. Stochastic Arithmetic in Multiprecision. *Mathematics in Computer Science* 5, 4 (2011), 359–375.
[6] Y. Hirota, S. Yamada, T. Imamura, N. Sasa, and M. Machida. 2016. Performance of quadruple precision eigenvalue solver libraries QPEigenK and QPEigenG on the K computer. HPC in Asia Poster, International Supercomputing Conference (ISC'16). (2016).
[7] J. Huthmann, B. Liebig, J. Oppermann, and A. Koch. 2013. Hardware/software co-compilation with the Nymble system. In *Proc. 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. 1–8.
[8] R. Iakymchuk, S. Collange, D. Defour, and S. Graillat. 2015. ExBLAS: Reproducible and Accurate BLAS Library. In *Proc. Numerical Reproducibility at Exascale (NRE2015) at SC'15*.
[9] F. Jézéquel and J.-M. Chesneaux. 2008. CADNA: a library for estimating round-off error propagation. *Comput. Phys. Commun.* 178, 12 (2008), 933–955.
[10] D. Mukunoki, T. Ogita, and K. Ozaki. 2019. Accurate and Reproducible BLAS Routines with Ozaki Scheme for Many-core Architectures. In *Proc. International Conference on Parallel Processing and Applied Mathematics (PPAM2019)*. (accepted).
[11] M. Nakata. 2010. The MPACK (MBLAS/MLAPACK); a multiple precision arithmetic version of BLAS and LAPACK. http://mplapack.sourceforge.net. (2010).
[12] 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. International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13)*. 1–12.
[13] K. Sano, H. Suzuki, R. Ito, T. Ueno, and S. Yamamoto. 2014. 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)*. 43–48.
[14] J. Vignes. 2004. Discrete Stochastic Arithmetic for Validating Results of Numerical Software. *Numerical Algorithms* 37, 1 (2004), 377–390.