HAL
open science

A Reproducible Accurate Summation Algorithm for High-Performance Computing

Caroline Collange, David Defour, Stef Graillat, Roman Iakymchuk

To cite this version:

Caroline Collange, David Defour, Stef Graillat, Roman Iakymchuk. A Reproducible Accurate Summation Algorithm for High-Performance Computing. EX: Exascale Applied Mathematics Challenges and Opportunities, Jul 2014, Chicago, United States. hal-01267825

HAL Id: hal-01267825

https://hal.science/hal-01267825v1

Submitted on 4 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Reproducible Accurate Summation Algorithm for High-Performance Computing

Sylvain Collange[1], David Defour[2], Stef Graillat[3], and Roman Iakymchuk[3,4]

[1] INRIA – Centre de recherche Rennes – Bretagne Atlantique
Campus de Beaulieu, F-35042 Rennes Cedex, France
sylvain.collange@inria.fr
[2] DALI–LIRMM, Université de Perpignan, 52 avenue Paul Alduy, F-66860 Perpignan, France
david.defour@univ-perp.fr
[3] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005 Paris, France
CNRS, UMR 7606, LIP6, F-75005 Paris, France
{stef.graillat, roman.iakymchuk}@lip6.fr
[4] Sorbonne Universités, UPMC Univ Paris 06, ICS, F-75005 Paris, France

Keywords: Parallel floating-point summation, reproducibility, accuracy, long accumulator, multi-precision, multi- and many-core architectures.

Floating-point (FP) addition is non-associative and parallel reduction involving this operation is a serious issue as noted in the DARPA Exascale Report [1]. Such large summations typically appear within fundamental numerical blocks such as dot products or numerical integrations. Hence, the result may vary from one parallel machine to another or even from one run to another. These discrepancies worsen on heterogeneous architectures – such as clusters with GPUs or Intel Xeon Phi processors – which combine programming environments that may obey various floating-point models and offer different intermediate precision or different operators [2,3]. Such non-determinism of floating-point calculations in parallel programs causes validation and debugging issues, and may lead to deadlocks [4].

The increasing power of current computers enables one to solve more and more complex problems. That, consequently, leads to a higher number of floating-point operations to be performed; each of them potentially causing a round-off error. Because of the round-off error propagation, some problems must be solved with a wider floating-point format. Two approaches exist to perform floating-point addition without incurring round-off errors. The first approach aims at computing the error that is occurred during rounding using *FP expansions*, which are based on an error-free transformation. FP expansions represent the result as an unevaluated sum of a fixed number of FP numbers, whose components are ordered in magnitude with minimal overlap to cover a wide range of exponents. FP expansions of sizes 2 and 4 are presented in [5] and [6], accordingly. The main advantage of this solution is that the expansion can stay in registers during the computations. But, the accuracy is insufficient for the summation of numerous FP numbers or sums with a huge dynamic range. Moreover, their complexity grows linearly with the size of the expansion. An alternative approach to expansions exploits the finite range of representable floating-point numbers by storing every bit in a very long vector of bits (accumulator). The length of the accumulator is chosen such that every bit of information of the input format can be represented; this covers the range from the minimum representable floating-point value to the maximum value, independently of the sign. For instance, Kulisch [7] proposed to utilize an accumulator of 4288 bits to handle the accumulation of products of 64-bit IEEE floating-point values. The Kulisch accumulator is a solution to produce the exact result of a very large amount of floating-point numbers of arbitrary magnitude. However, for a long period this approach was considered impractical as it induces a very large memory overhead. Furthermore, without dedicated hardware support, its performance is limited by indirect memory accesses that make vectorization challenging.

We aim at addressing both issues of accuracy and reproducibility in the context of summation. We advocate to compute the correctly-rounded result of the exact sum. Besides offering strict reproducibility through an unambiguous definition of the expected result, our approach guarantees that the result has

the best accuracy possible. However, the general use of correctly rounded sums has been considered impracticable since computing the exact sum was deemed to be too costly [8]. We revisit this assumption and show that: 1. The computation of the exact sum can be carried out at the affordable cost using a large fixed-point accumulator, which is called a *superaccumulator*; 2. The overhead can be made negligible on large sums with low dynamic range using vectorized FP expansions.

Most prior works on exact accumulation have been targeting ill-conditioned problems that could not be satisfied using the accuracy provided by the conventional floating-point accumulation. We instead focus on computing a deterministic and accuracy-guaranteed result for sums that are typically evaluated using the floating-point arithmetic today. We expect such sums to be well-conditioned and cover a moderate dynamic range.

We introduce a multi-level algorithm, where the accumulation of FP numbers is split into five levels; this decomposition is suitable for the nested parallelism of modern architectures. Thus, the first level consists of FP expansions to filter large numbers, which are expected to be the major part of all inputs. The second level is only invoked whenever the accuracy provided by expansions is not enough. Threads will send the values that are too small to be accumulated exactly in the expansions, as well as the results of partial sums accumulated during the first step, to private superaccumulators. During the third step, inside each processor, private superaccumulators are merged into a single scalar superaccumulator. In the fourth level, all scalar superaccumulators are combined together into one single superaccumulator accessible by all threads. Finally, the fifth level consists in rounding the global superaccumulator back to the target floating-point format in order to produce the correctly rounded result.

We present implementations of the multi-level algorithm on the whole range of parallel platforms: Intel desktop and server processors, Intel Xeon Phi accelerators, and both NVIDIA and AMD GPUs. These implementations strive to use all resources of modern processors: SIMD instructions, fused multiply-add, and multi-threading on multi-core CPUs and Xeon Phi; local memory and atomic instructions on GPUs. We verify the accuracy of our implementations by comparing the computed results with the ones produced by the multiple precision MPFR library [9].

We show that the numerical reproducibility and bit-perfect accuracy can be achieved at no additional cost for large sums that have dynamic ranges of up to 90 orders of magnitude by leveraging arithmetic units that are left underused by standard reduction algorithms. Furthermore, we demonstrate that the multi-level correct summation outperforms the other recently proposed reproducible summation algorithms – like the Demmel's and Nguyen's algorithm [8] – and, moreover, it offers correct rounding. More information can be found in our technical report [10].

References

1. Bergman, K., al.: Exascale computing study: Technology challenges in achieving exascale systems. DARPA Report (2008)
2. Whitehead, N., Fit-Florea, A.: Precision & performance: Floating point and IEEE 754 compliance for NVIDIA GPUs. Technical report, NVIDIA (2011)
3. Corden, M.: Differences in floating-point arithmetic between Intel® Xeon® processors and the Intel® Xeon Phi™ coprocessor. Technical report, Intel (2013)
4. Doertel, K.: Best known method: Avoid heterogeneous precision in control flow calculations. Technical report, Intel (2013)
5. Li, X.S., al.: Design, implementation and testing of extended and mixed precision BLAS. ACM Transactions on Mathematical Software **28**(2) (2002) 152–205
6. Hida, Y., Li, X.S., Bailey, D.H.: Algorithms for quad-double precision floating point arithmetic. In: Proc. 15th IEEE Symposium on Computer Arithmetic, IEEE Computer Society Press, Los Alamitos, CA, USA (2001) 155–162
7. Kulisch, U., Snyder, V.: The exact dot product as basic tool for long interval arithmetic. Computing **91**(3) (2011) 307–313
8. Demmel, J., Nguyen, H.D.: Fast reproducible floating-point summation. In: 21st IEEE Symposium on Computer Arithmetic, Austin, Texas, USA. (2013) 163–172
9. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: MPFR: A multiple-precision binary floating-point library with correct rounding. ACM Trans. Math. Softw. **33**(2) (2007) 13 `http://www.mpfr.org`.
10. Collange, S., Defour, D., Graillat, S., Iakymchuk, R.: Full-Speed Deterministic Bit-Accurate Parallel Floating-Point Summation on Multi- and Many-Core Architectures. Technical Report hal-00949355, INRIA, DALI–LIRMM, LIP6 (2014) `http://hal.archives-ouvertes.fr/hal-00949355/en/`.