Abstract

We aim at providing algorithms and implementations for fundamental linear algebra operations—like the ones included in the Basic Linear Algebra Subprograms (BLAS) library—that would deliver reproducible and accurate results with reasonable performance overhead compared to the standard non-reproducible implementations on modern parallel architectures such as Intel processors, Intel Xeon Phi co-processors, and GPU accelerators.

Introduction

In general, BLAS routines rely on the optimized version of parallel reduction and dot product involving floating-point additions and multiplications operations that are non-associative. Due to non-associativity of these operations and dynamic scheduling on parallel architectures, getting a bitwise reproducible floating-point result for multiple executions of the same code on different or even similar parallel architectures is challenging. These discrepancies worsen on heterogeneous architectures—such as clusters composed of standard CPUs in conjunction with GPU accelerators and/or Intel Xeon Phi co-processors—where different programming environments may obey various floating-point models and offer different intermediate precision or different operators. Such non-determinism and non-reproducibility of floating-point computations on parallel machines causes validation and debugging issues, and may even lead to deadlocks. Existing solutions to enhance reproducibility of BLAS routines are:

- **Fixed reduction scheme** such as Intel’s “Conditional Numerical Reproducibility” (CNR) available as an option in the Math Kernel Library (MKL);
- **Avoid rounding error** by using the Kulisch accumulator [3];
- **Mixed solution** as the one proposed by Demmel and Nguyen for BLAS level-1.

Our Approach

We introduced in [1] an approach to compute deterministic sums of floating-point numbers. This approach is based on a multi-level algorithm that combines efficiently floating-point expansion [2], which are placed in registers, and Kulisch accumulator.

Performance Results

We provided implementations of the multi-level summation scheme on a range of parallel platforms: desktop and server CPUs, the Intel Xeon Phi many-core accelerator, and both NVIDIA and AMD GPUs. We relied on the parallel summation algorithm as well as exact multiplication to develop the fast, accurate, and reproducible implementations of fundamental linear algebra operations such as dot product, triangular solver, and matrix-matrix multiplication. We verified the accuracy of our implementations by comparing the computed results with the ones produced by the multiple precision MPFR library.