Estimation of numerical reproducibility using stochastic arithmetic

Fabienne Jezequel
Associate Professor at LIP6 (Paris 6)

Differences in simulation results may be observed from one architecture to another or even inside the same architecture. Such reproducibility failures are often due to different rounding errors generated by different orders in the sequence of arithmetic operations. It must be pointed out that the cause of differences in results may be difficult to identify: rounding errors or bug? Such differences are particularly noticeable with multicore processors or GPUs (Graphics Processing Units).

In this talk, we describe the principles of DSA (Discrete Stochastic Arithmetic) which enables one to estimate rounding error propagation in simulation programs. We show that DSA can be used to estimate which digits in simulation results may be different from one environment to another because of rounding errors. We present the CADNA library (http://www.lip6.fr/cadna), an implementation of DSA that controls the numerical quality of programs and detects numerical instabilities generated during the execution. A particular version of CADNA which enables numerical validation in hybrid CPU-GPU environments is described. The estimation of numerical reproducibility using DSA is illustrated by a wave propagation code which can be affected by reproducibility problems when executed on different architectures.

Improving Performance Portability and Exascale Software Productivity with the ∇ Numerical Programming Language

Jean-Sylvain Camier
Engineer-Researcher at CEA-DAM

Addressing the major challenges of software productivity and performance portability becomes necessary to take advantage of emerging computing architectures. There is a growing demand for new programming environments in order to improve scientific productivity, to ease design and implementation, and to optimize large production codes.

We introduce the numerical analysis specific language Nabla (∇) which is an open-source (nabla-lang.org) Domain Specific Language (DSL) whose purpose is to translate numerical analysis algorithmic sources in order to generate optimized code for different runtimes and architectures. ∇ raises the level of abstraction, following a bottom-up compositional approach that provides a methodology to co-design between applications and underlying software layers for existing middleware or heterogeneous execution models.

One of the key concept is the introduction of the hierarchical logical time within the high-performance computing scientific community. This new dimension to parallelism is explicitly expressed to go beyond the classical single-program multiple-data or bulk-synchronous parallel programming models. Control and data concurrencies can be combined consistently to achieve statically analysable transformations and efficient code generation. Shifting the complexity to the compiler offers an ease of programming and a more intuitive approach, while reaching the ability to target new hardware and leading to performance portability.