The Numerical Accuracy Challenge for Code Modernization

Verificarlo: checking floating point accuracy through Monte Carlo Arithmetic

Pablo Oliveira[1], Yohan Chatelain[1];
Eric Petit[2];
Christophe Denis[3], Lin Guo[3],

[1]UVSQ - [2]Intel, IPAG EU, DCG - [3]CMLA

May 13th, 2017

Credits

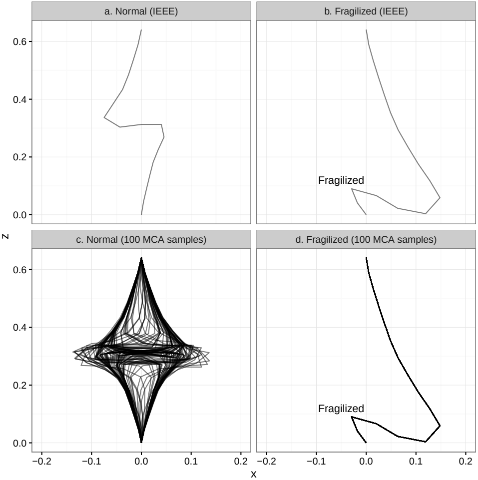
- The work presented in this slides has been done by the following institution and people
  - LiParad UVSQ : Pablo de Oliveira, Eric Petit (now at Intel), Yohan Chatelain
  - CMLA ENS-CACHAN: Christophe Denis
  - Intel: Eric Petit
- The tool is freely available opensource on github under GPL3 license.
- A more detailed paper and research reports can be found on Arxiv, HAL, and Arith23 proceedings.

Reproducibility versus precision

Portability across architecture, heterogeneity, compiler, optimizations level, languages etc. might result is slightly or even totally different results.

- Ensuring the numerical reproducibility is not always a requirement!
  - Most the HPC users want to be conservative
  - However does different results means wrong results?
- Precision analysis is required
  - For a given algorithm, precision bounds accuracy
  - Estimate the significant digits of a computation
  - Find the best compromise between performance, precision and reproducibility

Motivating Example About Reproducibility



Estimating the numerical precision by using Monte Carlo Arithmetic (MCA) [PARKER97]

- Stochastically simulate rounding and catastrophic cancellation errors
- Introduce a uniformly-distributed error at a virtual precision t

$$inexact(x) = x + 2^{e_x - t}\xi$$

  - e_x exponent of x, ξ uniform random variable in $[-\frac{1}{2}, \frac{1}{2}]$
- Each floating point operation is transformed in a MCA operation:

$$x \circ y \rightarrow round(inexact(inexact(x) \circ inexact(y)))$$

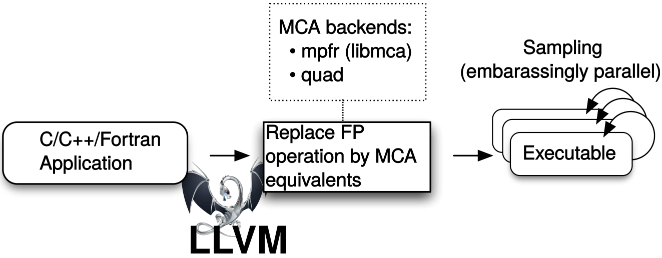
- Distribution of the errors is estimated using N Monte Carlo samplings $\mathbf{x}$
  - Costly in time, but not in memory and embarrassingly parallel
- $\hat{s}(\mathbf{x})$: estimation of s computed as follows:

$$\hat{s}(\mathbf{x}) = -log_{10}\frac{\hat{\sigma}(\mathbf{x})}{\hat{\mu}(\mathbf{x})}$$

  - $\hat{\mu}$: empirical mean value; $\hat{\sigma}$: empirical standard deviation

Verificarlo: an Automatic LLVM Tool for FP Accuracy Checking using MCA

- Support MCA analysis of large code-bases without any source code modification
  - eg. LAPACK, EDF code ASTER and Telemac, CEA Europlexus, CEA DAM Abinit...
- Instrumentation occurs after the optimization passes, just before the back-end ISA code generation
  $\rightarrow$ Verificarlo analyzes the code which is executed



Verificarlo: an Automatic LLVM Tool for FP Accuracy Checking using MCA

- Using LLVM brings advantages:
  - The instrumentation library is an independent module which can be tuned for other tools
  - LLVM supports multiple languages and multiple ISA
  - It benefits from the powerful analysis of the LLVM compiler based on code semantics
    - e.g. per function/loop analysis, access to debug info to relate the observation to the source code...
- But also some constraints:
  - Tied to LLVM compiler, addressing a new compiler would require to rewrite the compiler pass (but it is a short and simple piece of software)
  - Cannot handle precompiled libraries

Concluding remarks and future work

- The assessment of the numerical accuracy of scientific codes becomes crucial
  - When porting a scientific code on another programming language or on different computing resources
  - To find the best compromise between performance and precision
- The current version of Verificarlo is a fully automatic tool to estimate the numerical precision, but it still require expertise...
- Future research direction
  - Extract additional metric and improve the post-treatment toolbox to go beyond the standard deviation analysis of MCA runs
  - Methodologies to pinpoint the exact operation, loop, or routine that is to blame for a precision loss
  - Extend our experience on numerical verification and optimization of full-scale applications