First French-Japanese Workshop on Numerical Computations
March 22-25, 2015, Paris

Numerical Reproducibility:
Feasibility and Efficiency Issues

Philippe Langlois,
Chems Chohra, Rafife Nheili, David Parello

Equipe-projet DALI, Université de Perpignan Via Domitia
LIRMM, UMR 5506 CNRS - UM2
Acknowledgment

Ch. Denis, EDF R&D (Clamart) and CMLA (Cachan)
J.-M. Hervouet, EDR R&D (Chatou)
Numerical Reproducibility

1. Motivations

2. Feasibility of numerical reproducibility for industrial software

3. Efficiency of reproducible BLAS 1

4. Conclusion and work in progress
Motivations

Exascale HPC and numerical simulation

- Moore’s rule → $10^{18}$ flop/sec in 2020
- Massive and heterogeneous parallelism: 1 million of computing units
- Numerical simulation of complex and sensitive physical phenomena
Motivations

Exascale HPC and numerical simulation
- Moore’s rule $\rightarrow 10^{18}$ flop/sec in 2020
- Massive and heterogeneous parallelism: 1 million of computing units
- Numerical simulation of complex and sensitive physical phenomena

Numerical reproductibility failure of finite precision computations
- Non associative floating-point addition
- Computed value depends on the order of the operations
- Existing reproducibility failures for numerical simulations in energy [12], dynamical weather science [4], dynamical molecular [11], dynamical fluid [8]
Reproducibility failure of an industrial simulation code

- Simulation of free-surface flows in 1D-2D-3D hydrodynamic
- Integrated set of open source Fortran 90 modules, 300 000 loc
- LNHE (EDF R&D) + international consortium, 20 years, 4000 reg. users

Telemac 2D [5]

- 2D hydrodynamic: Saint Venant equations
- Finite element method, triangular element mesh, sub-domain decomposition for parallel resolution
- Mesh node unknowns: water depth (H) and velocity (U, V)
The Malpasset dam break (1959)

- A five year old dam break: **433 dead people and huge damage**
- Simulation mesh: 26000 elements and 53000 nodes
- Simulation: 2200 second with time step = 2 sec
The Malpasset dam break (1959)

- A five year old dam break: 433 dead people and huge damage
- Simulation mesh: 26000 elements and 53000 nodes
- Simulation: 2200 second with time step = 2 sec

<table>
<thead>
<tr>
<th></th>
<th>The sequential run</th>
<th>a 64 procs run</th>
<th>a 128 procs run</th>
</tr>
</thead>
<tbody>
<tr>
<td>velocity U</td>
<td>0.4029747E-02</td>
<td>0.4935279E-02</td>
<td>0.4512116E-02</td>
</tr>
<tr>
<td>velocity V</td>
<td>0.7570773E-02</td>
<td>0.3422730E-02</td>
<td>0.7545233E-02</td>
</tr>
<tr>
<td>depth H</td>
<td>0.3500122E-01</td>
<td>0.2748817E-01</td>
<td>0.1327634E-01</td>
</tr>
</tbody>
</table>
Reproducibility failure I

When? why?
- **Uncertainty** of the operation order for consecutive executions of a given binary file
- Both in parallel and “sequential+SIMD” environments

Parallel loop reduction: SIMD, openMP, MPI, GPU
- The number of computing units $p$ modifies the partial computed values before the reduction
- For a given $p$, the computed reduced value depends on the dynamic scheduling of the reduction (omp, mpi, gpu)

Memory data alignment: parallel and sequential issues
- Different $p$ modifies the data alignment to the cache line boundary
- Vectorized iteration loop vs. prologue-epilogue ones are modified
- “Sequential+SIMD” case: alignment may even depend on external events (OS) [2]
Reproducibility failure II

Reproducibility $\neq$ Portability

- Portability: one source $\rightarrow$ different binaries
- Portability parameters: compilers and their options, libraries, OS, computing units
- Reproducibility may fail for a given set of portability parameters

Reproducibility $\neq$ Accuracy

- Reproducibility: bitwise identical results for every $p$-parallel run, $p \geq 1$
- Full accuracy = unit roundoff accuracy = bitwise exact result
- Improving accuracy up to correct rounding yields reproducibility

Reproducibility

- Pros: numerical debug, validation, legal agreements
- Cons: numerical debug, stochastic arithmetic
Issues of the day

Feasibility

- Do existing techniques easily provide reproducibility to large industrial scientific software?

Efficiency

- Do correctly rounded summation algorithms provide efficient implementations of reproducible parallel BLAS routines?
First issue

1 Motivations

2 Feasibility of numerical reproducibility for industrial software
   - Integer converted assembly
   - Accurate compensated assembly
   - Reproducible algorithm based assembly

3 Efficiency of reproducible BLAS 1

4 Conclusion and work in progress
Do existing techniques easily provide reproducibility to large industrial scientific software?

One industrial scientific software

- Tomawac: Telemac’s module for wave propagation in coastal areas
- Transport equation: first order PDE → ODEs along its characteristic curves
- Integration reduces to accumulation (and interpolation) over the finite element mesh
- Unknowns: significant wave height, mean wave frequency and direction
- The Nice test case: effect of high-speed ferry waves
The two main resolution steps in a parallel FE resolution

**Inner node assembly step:**
\[ X(i) = \sum_{\text{elements}} W_e(i) \]

**Interface point assembly step:**
\[ X(i) = \sum_{\text{subdomain}} X_{\text{subdomain}}(i) \]

\[ X(i) = X^{(1)}(i) + X^{(2)}(i) + X^{(3)}(i) \]
\[ X(i) = X^{(2)}(i) + X^{(1)}(i) + X^{(3)}(i) \]
\[ X(i) = X^{(3)}(i) + X^{(1)}(i) + X^{(2)}(i) \]

Non associativity

Continuity constraint recovering

\[ X(i) = \max( X(i), X(i), X(i) ) \]

the max of the sums is not reproducible sequential vs. parallel

\( p \)-parallel \( D_1..D_p \) vs. \( q \)-parallel \( D_1..D_q \), \( p \neq q \)
Example of reproducibility failure

Sequential vs. Parallel computation for $p = 2, 4, 8, 16$

- Assembly with classical floating-point accumulations
- $FPAss$: parallel $FPSum_p$ and sequential $FPSum$

$$\max |FPAss_p - FPAss|/|FPAss|$$

Mean frequency wave, Nice test case, Tomawac
Do existing techniques easily provide reproducibility to large industrial scientific software?

Existing techniques to recover numerical reproducibility in summation:

- Integer conversion [7]
- Accurate compensated summation [6]
- Demmel-Nguyen’s reproducible sums [3]
Motivations

2 Feasibility of numerical reproducibility for industrial software
   - Integer converted assembly
   - Accurate compensated assembly
   - Reproducible algorithm based assembly

3 Efficiency of reproducible BLAS 1

4 Conclusion and work in progress
8 byte integer conversion of a N-length vector $X$

- **Scaling factor:** $Q_8(X, N) = \frac{\max(\text{INT K8})}{\max(X) \times N}$
- **Conversion:** $IX = X \times Q_8(X, N)$
- $S = SUM$ when $X \times Q_8(X, N) \in \text{INT K8}$
8 byte integer conversion:

\[ Q_8(X, N) = \frac{\text{max}(\text{INT} \ K8)}{\text{max}(X) \times N} \]

**Example**

- \( N = 10 \), let \( X \in [10^2, 10^6] \) so \( \text{max}(X) = 10^6 \)
- Let \( IX \in [0, 10^8] \) so \( \text{max}(IX) = 10^8 \)
- Conversion \( IX = Q \times X \) with \( Q = 10^8 / (10^6 \times 10) = 10 \)

\[ X = \begin{array}{cccccccc}
100.0 & 100.1 & \ldots & 100.11 & \ldots & 100.2 & \ldots & 100.26 \\
\checkmark & \checkmark & \ldots & \times & \ldots & \checkmark & \ldots & \times \\
IX = 1000 & 1001 & \ldots & 1001 & \ldots & 1002 & \ldots & 1002
\end{array} \]
IntAss: reproducible integer assembly

Assembly is easily converted in integer sum
- FE and interface point assembly steps are easily performed in integer
- Tomawac is a favourable case: few terms to accumulate

IntAss
- yields reproducible assembly
- but introduces accuracy lost during conversion
  - \( Q_8(X, N) = 2^\alpha \) and a short X-range improve the accuracy conversion

\[
\begin{align*}
\text{acc} &= \max_{\text{rel}} \text{IntAss}^p, \text{FPAss}^s \quad \text{rep} = \max_{\text{rel}} (\text{IntSum}^p, \text{IntAss}^s)
\end{align*}
\]
Motivations

Feasibility of numerical reproducibility for industrial software
- Integer converted assembly
- Accurate compensated assembly
- Reproducible algorithm based assembly

Efficiency of reproducible BLAS 1

Conclusion and work in progress
Accurate compensated summation . . .

Compensated assembly steps are easy to derive . . .

\[
X(i) = W_i(e_1) + W_i(e_2) + \ldots + W_i(e_n)
\]

\[
E(i) = \Delta_1 + \Delta_2 + \ldots + \Delta_n
\]

Subdomain FE assembly.

Computation and accumulation of rounding errors \( \Delta \).

Interface point assembly

\( i : \) interface point between \( D1, D2, D3 \)

Accumulation of errors \( \Delta \) and \( \varepsilon \).

\[
E(i) = (E(i) + E(i) + \varepsilon_1) + (E(i) + \varepsilon_2)
\]

Compensation of the FE assembly for every mesh node \( i \).
Accurate compensated summation gives reproducibility

Compensated assembly steps are easy to derive and efficient

- Reproducibility and accuracy

\[ A^s: \text{sequential, } A^p: p\text{-parallel} \]
\[ \max_{rel}(A_1, A_2) = |A_1 - A_2|/|A_2| \]

Accuracy (of FPSum):
\[ acc = \max_{rel}(\text{CompAss}^p, \text{FPAss}^s) \]

Reproducibility:
\[ rep = \max_{rel}(\text{CompAss}^p, \text{CompAss}^s) \]

Mean frequency wave, Nice test case, Tomawac
Motivations

Feasibility of numerical reproducibility for industrial software
- Integer converted assembly
- Accurate compensated assembly
- Reproducible algorithm based assembly

Efficiency of reproducible BLAS 1

Conclusion and work in progress
Demmel-Nguyen’s reproducible sums [3]

Parallel K-fold reproducible summation
- Derivation of Rump-Ogita-Oishi AccSum and FastAccSum
- Exact sum of shrunks defined thanks to the $\max x_i$
- $K$-fold process with $K$ a priori chosen
- $\max$ and $\sum$: two reductions in ReprodSum and FastReprodSum

Issues for reproducible assembly steps in Tomawac
- Max of the inner point and interface point contributions
  - Assembly steps are not one-line accumulations
  - Extra storage before EF accumulation: matrix of size $ndp \times nelem$
  - or two outer+inner iterations: $ndp \times nelem$
  - or 1-reduction algorithm
Reproducibility recovered for assembly with $K = 2$

Assembly reproducibility recovered for $K = 2$

$A^s$: sequential
$A^p$: parallel

$$\max_{rel}(A_1, A_2) = \frac{|A_1 - A_2|}{|A_2|}$$

$$acc = \max_{rel}(\text{RepAss}^p, \text{FPSum}^s)$$

$$rep = \max_{rel}(\text{RepAss}^p, \text{RepAss}^s)$$

Mean frequency wave, Nice test case, Tomawac

- For $K = 2$, $\text{RepAss}_p \approx \text{CompAss}$
First issue

Feasibility?

Do existing techniques easily provide reproducibility to large industrial scientific software?
Current feasibility for reproducible industrial software

Reproducibility is recovered for Tomawac

- **Integer conversion**: easily, low over-cost
- **Compensation**: easily, accurate, low over-cost
- **Reproducible sums**: efficient, less easily, more expensive
Second issue

1. Motivations

2. Feasibility of numerical reproducibility for industrial software
   - Integer converted assembly
   - Accurate compensated assembly
   - Reproducible algorithm based assembly

3. Efficiency of reproducible BLAS 1

4. Conclusion and work in progress
Numerical reproducibility for the BLAS

BLAS + correctly rounded sums

- Dot product of length \( n \to \) sum of length \( 2n \)
- A correctly rounded result is reproducible
- A large panel of algorithms for faithful or correctly rounded sums

Motivation

- How to benefit from these CR sums for reproducible BLAS?
- Is the over-cost acceptable in practice for reproducible BLAS?

Work in progress

- BLAS 1 : asum, dot, norm2
- openMP for shared memory
- Hybrid openMP-MPI for shared+distributed memory
Overview

Our methodology

1. Optimization and choice of the best sequential CR sums
2. Deriving parallel CR sums
3. Application to reproducible BLAS-1 routines

The starting point: sequential summation algorithms

- Faithful: AccSum [10], FastAccSum [9]
- Correctly rounded (in RtN): iFastSum, HybridSum [14], OnlineExact sum [15]
Reproducible parallel BLAS-1: algorithmic choice

- Rdot: FastAccSum (small $n$) or modified OnLineExact (large $n$)
- Rnrm2: Rdot+IEEE sqrt $\rightarrow$ reproducible only

All details in [1]
Efficiency of Reproducible Level 1 BLAS,
C. Chohra, Ph. L., D. Parello.
Submitted to SCAN 2014 Post-Conference Proceedings
http://hal-lirmm.ccsd.cnrs.fr/lirmm-01101723
## Experimental framework

### Software

<table>
<thead>
<tr>
<th>Compiler</th>
<th>ICC 14.0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options</td>
<td>-O3 -axCORE-AVX-I -fp-model double -fp-model strict -funroll-all-loops</td>
</tr>
<tr>
<td>Parallel library</td>
<td>OpenMP 4.0</td>
</tr>
<tr>
<td>BLAS library</td>
<td>Intel MKL 11</td>
</tr>
</tbody>
</table>

### Hardware

<table>
<thead>
<tr>
<th>Processor</th>
<th>Xeon E5 2660 (Sandy Bridge) at 2.2 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache</td>
<td>L1: 32KB, L2: 256KB, shared L3 for each socket: 20MB</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>51.2 GB/s</td>
</tr>
<tr>
<td>#cores</td>
<td>2 × 8 cores (hyper-threading disabled)</td>
</tr>
</tbody>
</table>
Parallel BLAS-1: Runtime overcost for reproducibility

Over-cost wrt. optimized non reproducible ones (MKL based)

<table>
<thead>
<tr>
<th>Vector size</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
<th>$10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rasum/asum</td>
<td>2.0</td>
<td>(1/1)</td>
<td>1.5</td>
<td>(4/2)</td>
</tr>
<tr>
<td>Rdot/mkldot</td>
<td>6.4</td>
<td>(8/*)</td>
<td>3.8</td>
<td>(8/*)</td>
</tr>
<tr>
<td>Rnrm2/nOrm2</td>
<td>9.1</td>
<td>(8/*)</td>
<td>7.1</td>
<td>(8/*)</td>
</tr>
</tbody>
</table>
Parallel BLAS-1: Runtime overcost for reproducibility

<table>
<thead>
<tr>
<th>Vector size</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
<th>$10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rasum/asm</td>
<td>2.0 (1/1)</td>
<td>1.5 (4/2)</td>
<td>1.3</td>
<td>1.1</td>
</tr>
<tr>
<td>Rdot/mkldot</td>
<td>6.4 (8/⋆)</td>
<td>3.8 (8/⋆)</td>
<td>1.6</td>
<td>1.1</td>
</tr>
<tr>
<td>Rnrm2/norm2</td>
<td>9.1 (8/⋆)</td>
<td>7.1 (8/⋆)</td>
<td>3.4</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Over-cost wrt. reproducible ones

<table>
<thead>
<tr>
<th>Vector size</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
<th>$10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rasum/FastReprodasm</td>
<td>0.9 (1/1)</td>
<td>0.9 (4/4)</td>
<td>1.0</td>
<td>0.8</td>
</tr>
<tr>
<td>Rdot/FastReprodDot</td>
<td>1.5 (8/1)</td>
<td>1.5 (8/8)</td>
<td>0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>Rnrm2/FastReprodNrm2</td>
<td>1.7 (8/1)</td>
<td>1.5 (8/8)</td>
<td>0.9</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Parenthesis: best number of threads for implementation (if $\neq 16$) and ⋆ for the (hidden) Intel MKL’s choice.
Parallel BLAS-1: Runtime overcost for reproducibility

Runtime/size of parallel level 1 BLAS, up to 16 threads, cond=10^{32}

- **asum**
- **dot**
- **nrm2**
Parallel BLAS-1: scalability

Runtime/size of sequential BLAS-1

![Graphs showing runtime/size of sequential BLAS-1 operations](image)

Runtime/size of parallel BLAS-1, up to 16 threads

![Graphs showing runtime/size of parallel BLAS-1 operations](image)
Motivations

Feasibility of numerical reproducibility for industrial software
- Integer converted assembly
- Accurate compensated assembly
- Reproducible algorithm based assembly

Efficiency of reproducible BLAS 1

Conclusion and work in progress
Conclusion, work in progress and todo list

Reproducibility

- How to remain confident facing the complexity of today’s computational systems? and tomorrow?

Feasability

- Existing techniques are efficient...
- and more or less easy to apply
- TODO: Telemac 2D?

Efficiency

- Convincing BLAS level 1
- Hybrid openMP+MPI and scalability “in the large”: occigen ...
- TODO: BLAS-2, optimistic
- TODO: BLAS-3, no future!
C. Chohra, P. Langlois, and D. Parello.  
Efficiency of Reproducible Level 1 BLAS.  

J. M. Corden and D. Kreitzer.  
Consistency of Floating-Point Results using the Intel Compiler or Why doesn’t my application always give the same answer?  
Intel Corporation, Aug. 2014.

Fast reproducible floating-point summation.  

Y. He and C. Ding.  
Using accurate arithmetics to improve numerical reproducibility and stability in parallel applications.  

J.-M. Hervouet.  
Hydrodynamics of free surface flows: Modelling with the finite element method.  

T. Ogita, S. M. Rump, and S. Oishi.  
Accurate sum and dot product.  


A parallel algorithm for accurate dot product.  

Y.-K. Zhu and W. B. Hayes.  
Correct rounding and hybrid approach to exact floating-point summation.  

Y.-K. Zhu and W. B. Hayes.  
Algorithm 908: Online exact summation of floating-point streams.  