Reproducible and Accurate Parallel Triangular Solver

Chemseddine Chohra, Philippe Langlois, David Parello

Reproducible and Accurate Parallel Triangular Solver

Chemseddine Chohra[1],
Philippe Langlois[2] and David Parello[2]

[1]University 8 Mai 1945 Guelma, LabSTIC, Algeria

[2]DALI-LIRMM, U. Perpignan Via Domitia, U. Montpellier, CNRS, France

July, 17^{th} 2019

ICIAM 2019, Valencia, Spain.

Table of Contents

Rounding Errors and Reproducibility

IEEE-754 Floating-Point Numbers

- Approximate real numbers on computer.
- $f = \pm mantissa \cdot 2^{exponent}$.
- IEEE-754 standard defines formats and rounding modes.
- *binary*64 and RTN in this talk.

Floating-Point Operations

- For $x, y \in \mathbb{F}$ and $x + y \notin \mathbb{F}$, $x + y \neq x \oplus y = round(x + y)$. .
- The same applies for $\ominus$, $\otimes$ and $\oslash$.

Operation Order Matters: FP Addition is not Associative

- $a \oplus (b \oplus c) \neq (a \oplus b) \oplus c$.
- For *binary*64's round-off unit $u = 2^{-53}$:
  $0 = -1 \oplus (1 \oplus u) \neq (-1 \oplus 1) \oplus u = u$.

Does Numerical Reproducibility Matter?

Numerical Reproducibility and HPC

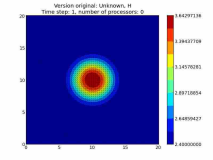
- Reproducibility: bitwise *identical* results for every p-parallel run, $p \geq 1$
- Reproducibility $\neq$ Accuracy
- How to **debug**? to **test**? to **validate**? to receive **legal agreements**?
  - Debug: rounding errors *vs.* bugs? reproduce errors?
  - Validate: reproduce *the* reference result? the same results from one run to another?

In Practice?

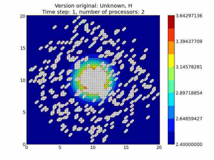
- Failures reported in numerical simulation for climate modeling (2001), energy (2009), dynamic molecular (2010), dynamic fluid (2011), hydrodynamic (2016)

Telemac2D simulation: a white plot displays a non reproducible result (Nheili *et al.*, 2016)

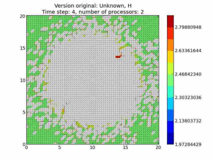
1 proc., t = 0.2sec.



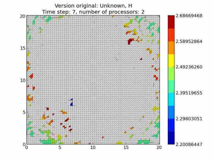
2 proc., 0.2 sec.



2 proc., 0.8 sec.



2 proc., 1.4 sec.



How to Solve Numerical Reproducibility Problems?

Strategies for Reproducibility

- Static order of operations
  - Static scheduling.
  - Deterministic Reduction.
  - Intel MKL Conditional Numerical Reproducibility (CNR).
- Pre-rounding Techniques.
  - ReprodSum and FastReprodSum (Demmel *et al.*, 2013).
  - **Indexed (Binned) floating-point format** (Demmel *et al.*, 2016).
    - Used in ReproBLAS library[a].
- Higher precision (Villa *et al.*, 2009, Iakymchuk *et al.*, 2015).
- **Correctly rounded** (Chohra *et al.*, 2016).

[a] http://bebop.cs.berkeley.edu/reproblas/

Our Aim

RARE-BLAS (2017-)

- Reproducible, Accurately Rounded and Efficient BLAS[a]
- Parallel BLAS 1: correctly rounded `dot` and `asum`, reproducible and faithfully rounded `nrm2`
- Parallel BLAS 2: correctly rounded `gemv`
- Accuracy *vs.* efficiency
  - Chose and tune summation algorithms wrt. architecture and problem constraints.
  - SIMD (AVX2-512), openMP, MPI
  - Run-time overhead ratio: $\times 1 \rightarrow \times 10$

[a] https://gite.lirmm.fr/rare-blas-group/rare-blas

Today: Reproducible Parallel `trsv`

- Provide a reproducible, accurate and efficient triangular solver.
- Two different approaches are presented and compared.
- Performance evaluation on CPU and Intel Xeon Phi accelerator.

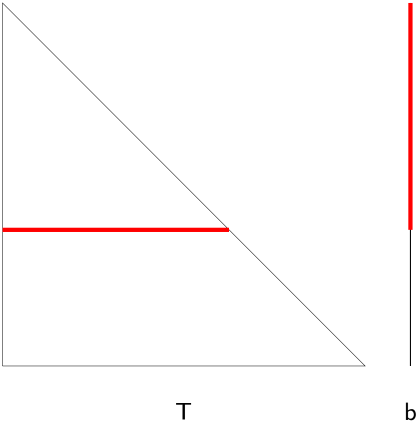
Table of Contents

From Classic Forward Substitution to Parallel `trsv`

Triangular solver

- Given a lower triangular $n \times n$-matrix T and n-vector b.
- Find x such that $Tx = b$.
- Forward substitution: $x_i = \left(b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j \right) / t_{i,i}$.
- Dependency of x_i wrt. x_j, $j < i$.

Triangular Solver: Sequential but SIMD-zed



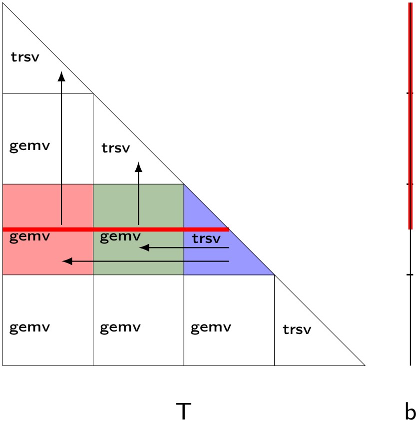
Sequential computation

- $x_1 = b_1 / t_{1,1}$.
- $x_i = (b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j) / t_{i,i}$.

Sources of non reproducibility

- Dot product accumulation
- SIMD lengths
- SIMD reduction schemes

Triangular Solver: Parallel Case



Parallel Process

- *trsv*: sequential.
- *gemv*: parallel.

Parallel computation

- $x_i = (b_i - \sum_{j=1}^{r} t_{i,j} \times x_j - \sum_{j=r+1}^{2r} t_{i,j} \times x_j - \sum_{j=2r+1}^{i-1} t_{i,j} \times x_j \,)/t_{i,i}.$

Sources of non reproducibility

- Dot product: partial accumulations wrt. block size r
- Accumulation order wrt. gemv scheduling
- gemv: SIMD lengths, SIMD reduction schemes

Table of Contents

Reproducible Triangular Solvers

Trade-off

Efficiency *vs.* Accuracy *vs.* Reproducibility

RTrsv

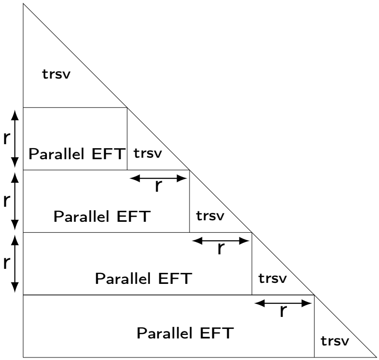
- Correctly rounded $b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j$
- EFT: TwoProd, HybridSum (Zhu-Hayes, 2009)

BinnedTrsvIR

- Reproducibility: BinnedTrsv
  - Binned accumulation *à la* Demmel-Nguyen's ReproBLAS.
  - Efficiency: "only" target reproducibility
- Accuracy: Iterative refinement

Table of Contents

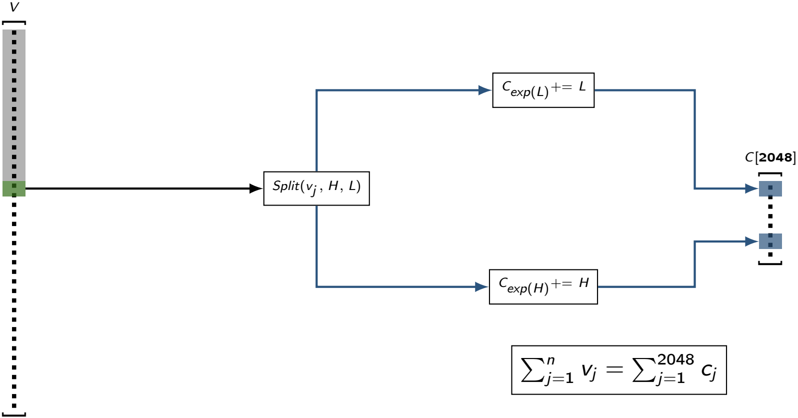
RTrsv: Relies on HybridSum (Zhu-Hayes, 2009)



Parallel Process

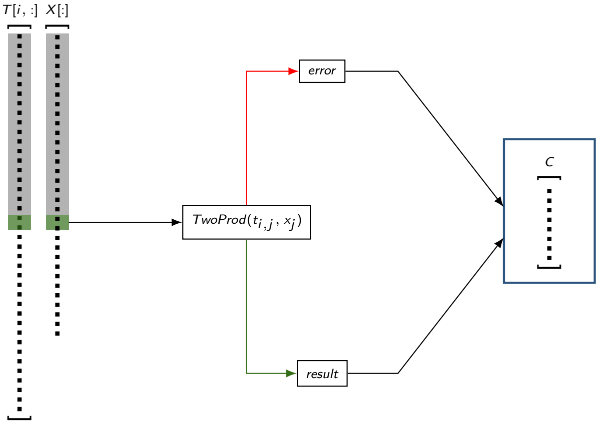
- EFT blocks use HybridSum to transform several rows in parallel.
- *trsv* blocks build on previous transformation to ensure correctly rounded $b_i - \sum_{j=1}^{i-1} t_{i,j} \times x_j$ and then divide it by $t_{i,i}$.

Error-Free Transformation for summation

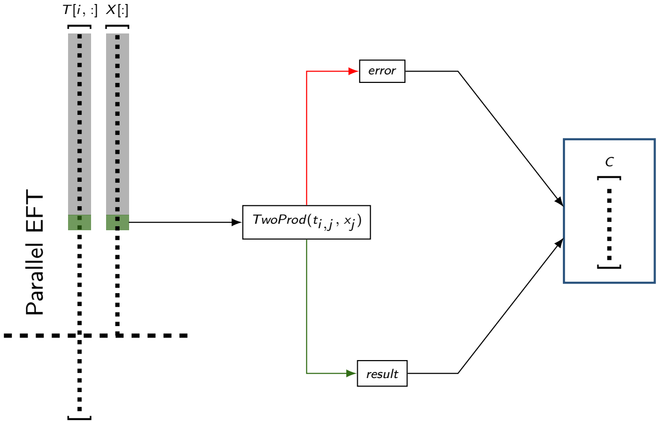


$$\sum_{j=1}^{n} v_j = \sum_{j=1}^{2048} c_j$$

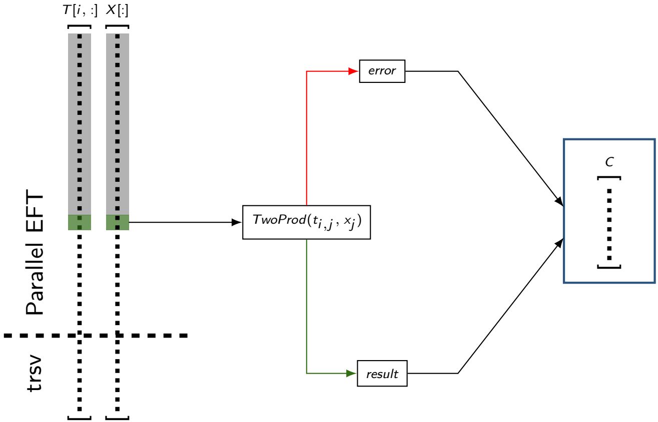
Error-Free Transformation for `trsv`



Error-Free Transformation for `trsv`



Error-Free Transformation for trsv



$$\boxed{b_i - \sum_{j=1}^{m} t_{i,j} \times x_j - \sum_{j=m+1}^{i-1} t_{i,j} \times x_j = \sum_{j=1}^{2048} C_j}$$

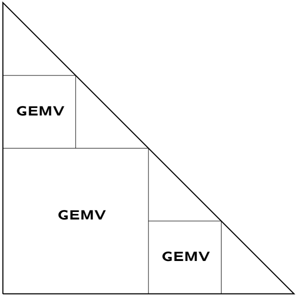
First Results (detail later)

Reproducible but a bit disapointing

- Reproducible solver
- but correctly rounded accumulation $\not\Rightarrow$ solution accuracy improvement
- with run-time overhead.

Table of Contents

BinnedTrsv: Relies on Indexed Floating-Point Format



Parallel Process

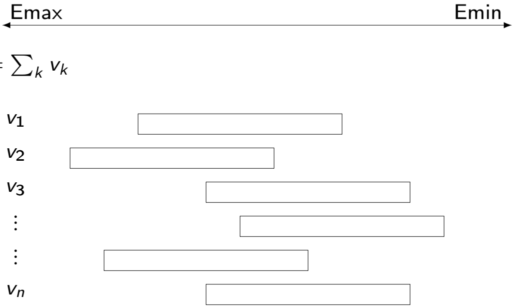
- The input matrix is recursively decomposed into :
  - Square *GEMV* blocks.
  - Triangular *TRSV* blocks.
- Sequential small *TRSV* blocks
- Parallel *GEMV* blocks.

Reproducibility

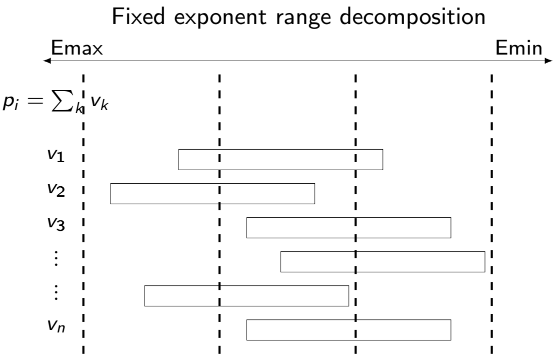
- FP multiplications and divisions
- All accumulations are performed into a n-vector of Indexed FP numbers: one for every x_i

Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

$p_i = \sum_k v_k$

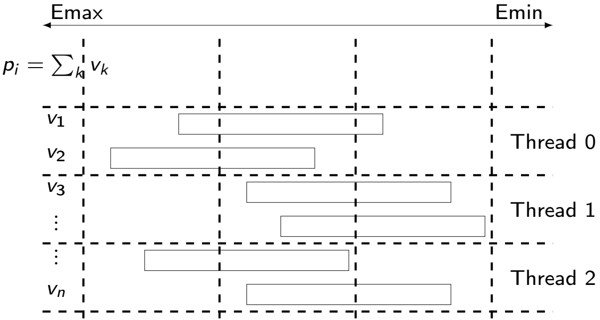


Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

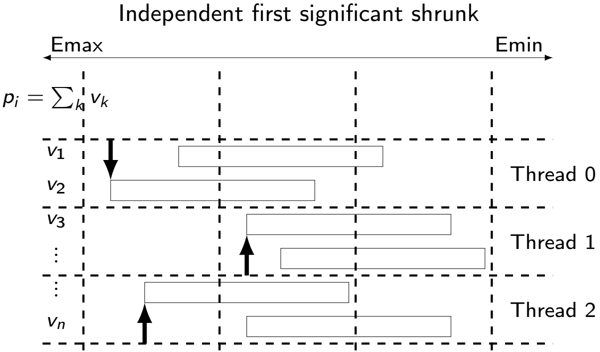


Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

De facto parallel sharing

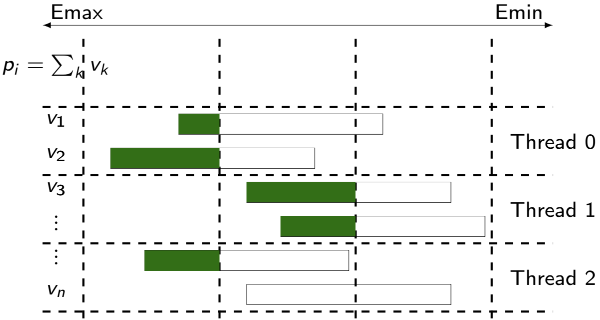


Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

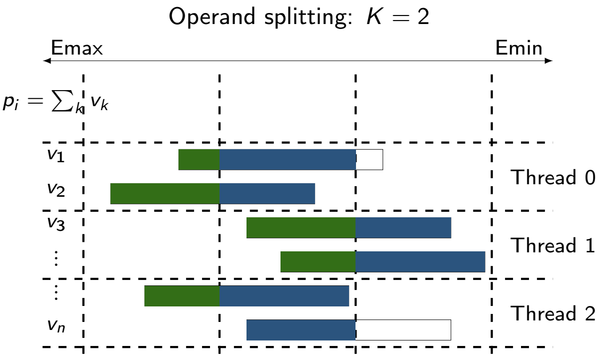


Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)

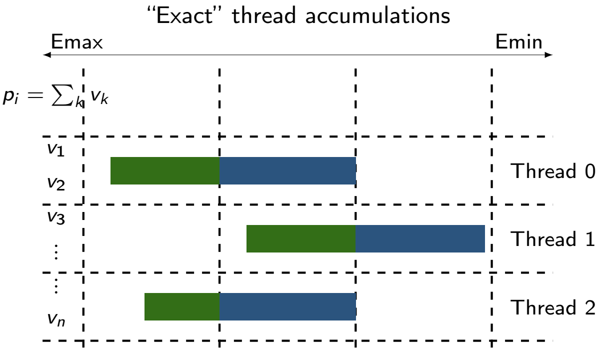
Operand splitting: $K = 1$



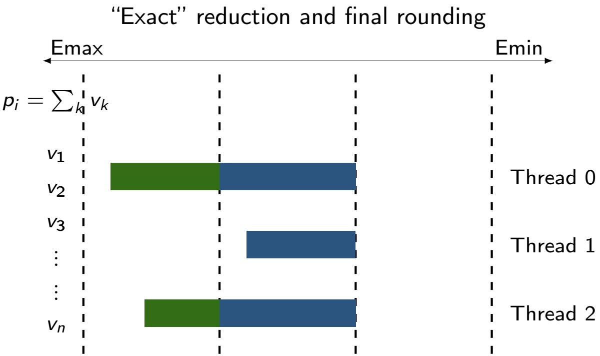
Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)



Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)



Indexed Floating-Point Numbers (Demmel-Nguyen, 2016)



BinnedTrsvIR: BinnedTrsv + Iterative Refinement

Reproducible Iterative Refinement

1. Solve the system with *BinnedTrsv* and $K = 2$.
    - Reproducibility
    - Tradeoff efficiency *vs.*initial accuracy
2. Compute $r^{(i)} = b - T\widehat{x}$ using higher precision.
    - $\times \rightarrow$ TwoProd
    - Higher precision indexed FP numbers: $K = 3$
    - Parallel and reproducible
3. Solve the system $Ad^{(i)} = r^{(i)}$ with reproducible *BinnedTrsv*
4. Update $\widehat{x} = \widehat{x} + d^{(i)}$.
5. Repeat from 2 until $\widehat{x}$ is accurate enough.

Table of Contents

Experimental Framework : Hardware and Software Configurations

CPU Configuration

- Dual Intel Xeon E5-2650 v2 16 cores (8 per socket).
- Memory bandwidth 59,7 GB/s.

Many-core Accelerator

- Intel Xeon Phi 7120 accelerator, 60 cores.
- Memory bandwidth 352 GB/s.

Compiler and Options

- Intel compiler (17.0.1)
- Intel OpenMP 5.0
- `-O3 -fp-model double -fp-model strict -funroll-all-loops`

  `-fp-model double` : rounds intermediate results to 53-bit precision
  `-fp-model strict` : disable contractions

Accuracy and Performance Experiments

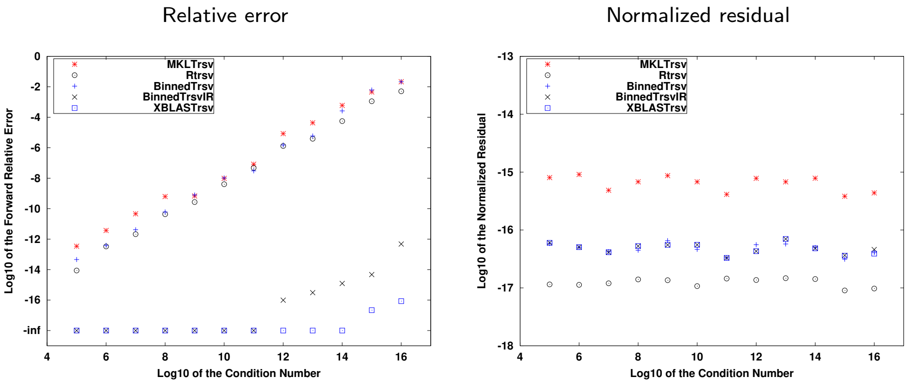
Experiments

- System size
  - Accuracy: $n = 1000$
  - Run-time: $n \in [10000, 15000]$
- $\text{Cond}(T, x) = \| |T^{-1}| \, |T| \, |x| \|_\infty / \| |x| \|_\infty$
  - Accuracy: $\text{Cond} \in [10^5, 10^{15}]$
  - Run-time: $\text{Cond} = 10^8$
- Reference solution : $\widetilde{x} = \text{MPFR}(T^{-1}b)$
- Relative error $= \|\widetilde{x} - \widehat{x}\|_\infty / \|\widetilde{x}\|_\infty$
- Normalized Residual $= \|b - T\widehat{x}\|_\infty / \|b\|_\infty$

Challenging solutions

- Accuracy: Intel MKL Trsv (b64)
- High accuracy: XBLAS double-doubled Trsv
- Performance: Intel MKL Trsv

Reproducible Solvers: Accuracy Results



- Accuracy: correctly rounded (RTrsv) *vs.*. reproducible (BinnedTrsv) dot prods
  → slightly but not significantly better
- More accuracy: similar and classic iterative refinement effect
- Residual: no condition effect, nor solution accuracy significance,
  slight effect of the correctly rounded version (RTrsv)

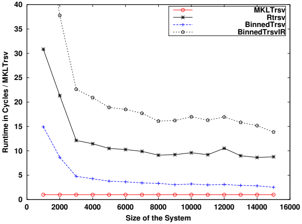
Reproducible Solvers: Performance Results

Run-time overhead ratio *vs.*MKL Trsv, Cond $= 10^8$.

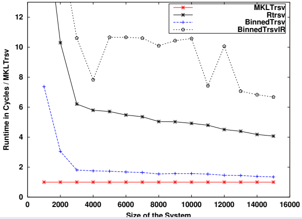
	Sequential	Parallel CPU	Parallel Accelerator
RTrsv	$\times 10$	$\times 5$	$\times 5 - 10$
BinnedTrsv	$\times 2$	$\times 1$	$\times 0.7 - 1.2$
BinnedTrsvIR	$\times 15 - 20$	$\times 8 - 10$	$\times 2$

Note: no accelerator benefit (`trsv` dependencies – as it)

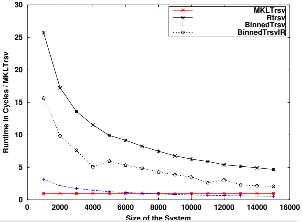
Sequential



Parallel: CPU



Parallel: Xeon Phi



- Parallel solutions scale well
- Reproducibility: from no over-cost to very reasonable cost with BinnedTrsv
- Accuracy cost: not free (iterative refinement) but interesting for accelerator

Table of Contents

Conclusion and Future Works

Conclusion

- Numerical reproducibility for SIMD, multi-core, many-core architectures
- Accuracy: similar to XBLAS
- Run-time performance (*vs.* MKLTrsv):

	Sequential	Parallel CPU	Parallel Accelerator
RTrsv	$\times 10$	$\times 5$	$\times 5 - 10$
BinnedTrsvIR	$\times 15 - 20$	$\times 8 - 10$	$\times 2$

Future Works

- Higher level BLAS: compute bound algorithms $\Rightarrow$ need for other strategies
- Auto tuning to easy optimization
- Build a reproducible LAPACK upon RARE-BLAS