Memory Accessor for Mixed-precision GMRES-based iterative refinement

Patrick Amestoy[1] Antoine Jego[2] Jean-Yves L'Excellent[1]
Theo Mary[2] , Mumps Technologies[1], Sorbonne Université, CNRS, LIP6[2]

RAIM 2025

Motivation

We want to solve $Ax = b$.

Some key ingredients

- **Direct** methods for robustness.
- **Approximations** to avoid useless compute.
  - Taking sparsity into account
  - (Block) Low-Rank compression
  - Arithmetic compression
- **Iterative** methods to restore an accurate solution.

$\Rightarrow$ Mixed-Precision Preconditioned Iterative Refinement

Motivation

We want to solve $Ax = b$.

Some key ingredients

- **Direct** methods for robustness.
- **Approximations** to avoid useless compute.
  - Taking sparsity into account
  - (Block) Low-Rank compression
  - Arithmetic compression
- **Iterative** methods to restore an accurate solution.

$\Rightarrow$ Mixed-Precision Preconditioned Iterative Refinement

Methods – Iterative refinement

Assuming $\kappa(A) = \|A\|\|A^{-1}\| < \frac{1}{u_{\texttt{fp32}}} = 1.6 \times 10^7$ the following algorithm is guaranteed to converge to a solution of $Ax = b$ in precision **fp64**.

Factorize $A = LU$ in precision **fp32**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ in precision **fp32**
repeat
  $r_i = b - Ax_i$ in precision **fp64**
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision **fp32**
  $x_{i+1} = x_i + d_i$ in precision **fp64**
until converged

- $O(n^3)$ flops **fp32**
- $O(n^2)$ flops **fp64** per iteration

J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, J. Dongarra. *"Exploiting the Performance of 32 bit Floating Point Arithmetic in Obtaing 64 bit Accuracy (Revisiting Iterative Refinement for Linear Systems)"*. Supercomputing 2006.

Assuming $\kappa(A) = \|A\|\|A^{-1}\| < \frac{1}{u_{\texttt{fp32}}} = 1.6 \times 10^7$ the following algorithm is guaranteed to converge to a solution of $Ax = b$ in precision **fp64**.

Factorize $A = LU$ in precision **fp32**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ in precision **fp32**
repeat
  $r_i = b - Ax_i$ in precision **fp64**
  Solve $Ad_i = r_i$ via $d_i = U^{-1}(L^{-1}r_i)$ in precision **fp32**
  $x_{i+1} = x_i + d_i$ in precision **fp64**
until converged

- $O(n^3)$ flops **fp32**
- $O(n^2)$ flops **fp64** per iteration

J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, J. Dongarra. *"Exploiting the Performance of 32 bit Floating Point Arithmetic in Obtaing 64 bit Accuracy (Revisiting Iterative Refinement for Linear Systems)"*. Supercomputing 2006.

Assuming $(u_{\texttt{fp64}} + u_{\texttt{fp64}}\kappa(A))(1 + u_{\texttt{fp32}}^2\kappa(A)^2) < 1$ *i.e.* $\kappa(A) < 10^{10}$ the following algorithm is guaranteed to converge to a solution of $Ax = b$ in precision **fp64**.

Factorize $A = LU$ in precision **fp32**
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ in precision **fp32**
repeat
  $r_i = b - Ax_i$ in precision **fp64**
  Solve $Ad_i = r_i$ via LU-preconditioned GMRES converging to accuracy **fp64**
  $x_{i+1} = x_i + d_i$ in precision **fp64**
until converged

- $O(n^3)$ flops **fp32**
- $O(n^2)$ flops **fp64** per iteration

E. Carson, N.J. Higham. *"Accelerating the solution of linear systems by iterative refinement in three precisions"*. SIAM J. Scientific Computing (2018).

Assuming $(u_g + u_p\kappa(A))(1+u_f{}^2\kappa(A)^2) < 1$ the following algorithm is guaranteed to converge to a solution of $Ax = b$ in precision u.

Factorize $A = LU$ in precision u_f
Solve $Ax_1 = b$ via $x_1 = U^{-1}(L^{-1}b)$ in precision u_f
repeat
  $r_i = b - Ax_i$ in precision u_r
  Solve $Ad_i = r_i$ via LU-preconditioned GMRES cv. to acc. u_g with prec. u_p
  $x_{i+1} = x_i + d_i$ in precision u
until converged

- $O(n^3)$ flops u_f
- $O(n^2)$ flops u_g, u_p, u_r per iteration

P. R. Amestoy, A. Buttari, N. J. Higham, J.-Y. L'Excellent, T. Mary, B. Vieublé. *"Five-Precision GMRES-based Iterative Refinement"*. Journal on Matrix Analysis and Applications (2024).

Performance–robustness tradeoff $u_g = u_{\text{fp64}}$

Assuming $u_f = u_p = u_{\text{fp32}}$

- 👍 LU factors can be applied in their initial precision
- 👎 Bound degrades to $\kappa(A) < 10^7$

Assuming $u_f = u_{\text{fp32}} > u_p = u_{\text{fp64}}$

- 👎 LU factors have to be converted to higher precision.
- 👍 Bound is maintained ($\kappa(A) < 10^{10}$)

Decoupling **storage** and **compute** arithmetics improves the robustness of GMRES-IR.

⚠ Without decoupling, GMRES may not converge without a **flexible** variant.

Alfredo Buttari, Xin Liu, Theo Mary, Bastien Vieublé. *"Mixed precision strategies for preconditioned GMRES: a comprehensive analysis"*. Preprint (2025).

Performance–robustness tradeoff $u_g = u_{\text{fp64}}$

Assuming $u_f = u_p = u_{\text{fp32}}$

- 👍 LU factors can be applied in their initial precision
- 👎 Bound degrades to $\kappa(A) < 10^7$

Assuming $u_f = u_{\text{fp32}} > u_p = u_{\text{fp64}}$

- 👎 LU factors have to be converted to higher precision.
- 👍 Bound is maintained ($\kappa(A) < 10^{10}$)

Decoupling **storage** and **compute** arithmetics improves the robustness of GMRES-IR.

⚠ Without decoupling, GMRES may not converge without a **flexible** variant.

Alfredo Buttari, Xin Liu, Theo Mary, Bastien Vieublé. *"Mixed precision strategies for preconditioned GMRES: a comprehensive analysis"*. Preprint (2025).

Performance–robustness tradeoff $u_g = u_{\text{fp64}}$

Assuming $u_f = u_p = u_{\text{fp32}}$

- 👍 LU factors can be applied in their initial precision
- 👎 Bound degrades to $\kappa(A) < 10^7$

Assuming $u_f = u_{\text{fp32}} > u_p = u_{\text{fp64}}$

- 👎 LU factors have to be converted to higher precision.
- 👍 Bound is maintained ($\kappa(A) < 10^{10}$)

Decoupling **storage** and **compute** arithmetics improves the robustness of GMRES-IR.

⚠ Without decoupling, GMRES may not converge without a **flexible** variant.

Alfredo Buttari, Xin Liu, Theo Mary, Bastien Vieublé. *"Mixed precision strategies for preconditioned GMRES: a comprehensive analysis"*. Preprint (2025).

Performance–robustness tradeoff $u_g = u_{\text{fp64}}$

Assuming $u_f = u_p = u_{\text{fp32}}$

- LU factors can be applied in their initial precision
- Bound degrades to $\kappa(A) < 10^7$

Assuming $u_f = u_{\text{fp32}} > u_p = u_{\text{fp64}}$

- LU factors have to be converted to higher precision.
- Bound is maintained ($\kappa(A) < 10^{10}$)

Decoupling **storage** and **compute** arithmetics improves the robustness of GMRES-IR.

⚠ Without decoupling, GMRES may not converge without a **flexible** variant.

Alfredo Buttari, Xin Liu, Theo Mary, Bastien Vieublé. *"Mixed precision strategies for preconditioned GMRES: a comprehensive analysis"*. Preprint (2025).

Performance-robustness tradeoff – more

Approximate computing techniques make it possible to set any values for u_f.

- Low-Rank approximations : tiles in the sparse factors are represented as

$B \approx X Y^T$ if $\|B_{ij} - X_{ij}Y_{ij}^T\| < u_f\|A\|$

- Arithmetic approximations : tiles in the sparse factors are converted to precision u if $\|B_{ij} - B_{ij}^{(u)}\| < u_f\|A\|$

u_f	$u_p = u_{\text{fp64}}$	$u_p = u_{\text{fp32}}$
10^{-6}	2×10^9	2×10^6
10^{-4}	1×10^8	1×10^5
10^{-2}	4×10^6	6×10^3
10^{0}	2×10^5	3×10^2

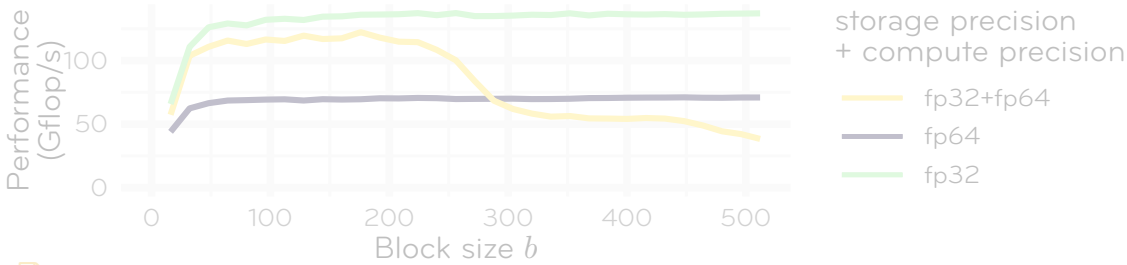
Well-conditioned problems should use more approximations

Memory accessor – Decoupling arithmetics

Decoupling the **storage arithmetic** from the **compute arithmetic** can be achieved by implementing a *Memory Accessor*.

Thomas Grützmacher, Hartwig Anzt, Enrique S. Quintana-Ortí. *"Using Ginkgo's memory accessor for improving the accuracy of memory-bound low precision BLAS"*. Journal of software: practice and experience, 2023.

It is possible to leverage the technique at a coarse block level.

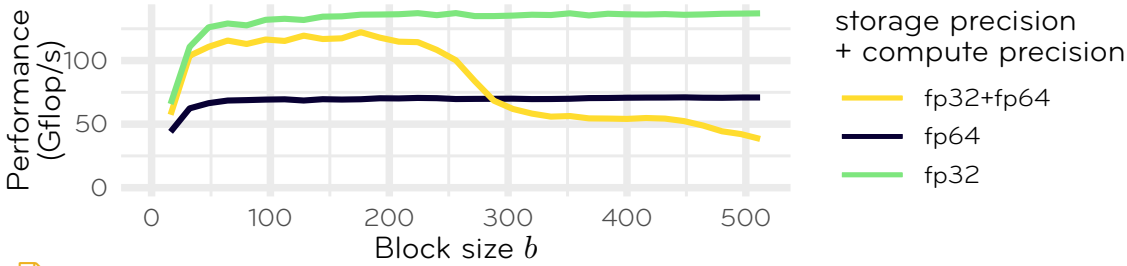


Patrick Amestoy, Antoine Jego, Jean-Yves L'Excellent, Théo Mary, and Grégoire Pichon. *"Blas-based block memory accessor with applications to mixed precision sparse direct solvers"*. Submitted to ACM TOMS.

Decoupling the **storage arithmetic** from the **compute arithmetic** can be achieved by implementing a *Memory Accessor*.

Thomas Grützmacher, Hartwig Anzt, Enrique S. Quintana-Ortí. *"Using Ginkgo's memory accessor for improving the accuracy of memory-bound low precision BLAS"*. Journal of software: practice and experience, 2023.

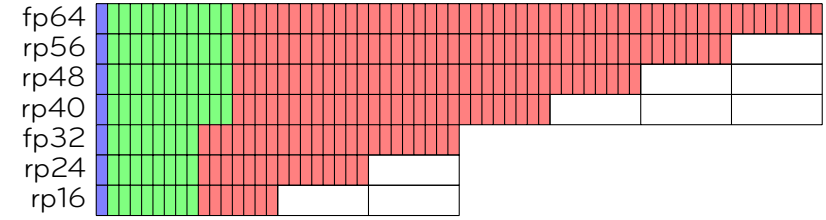
It is possible to leverage the technique at a coarse block level.



Patrick Amestoy, Antoine Jego, Jean-Yves L'Excellent, Théo Mary, and Grégoire Pichon. *"Blas-based block memory accessor with applications to mixed precision sparse direct solvers"*. Submitted to ACM TOMS.

Low precision arithmetics – Reduced precision

IEEE-compliant arithmetic formats have been extended by *chopping* bytes off the mantissa, yielding at least 7 precisions.



D. Mukunoki, T. Imamura. *"Reduced-Precision Floating-Point Formats on GPUs for High Performance and Energy Efficient Computation."*. IEEE CLUSTER 2016.

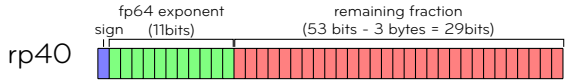
You can produce chopped versions of fp128 but our focus is on BLAS-based operations.

Low precision arithmetics – Custom formats

Type		Exponent	Mantissa	Range $2^{\pm 2^{\text{Exponent}-1}}$	Precision $2^{-\text{Mantissa}}$
fp64	double	11 bits	53 bits	$2^{\pm 2^{11-1}} \approx 10^{\pm 308}$	$2^{-53} \approx 1 \times 10^{-16}$
rp56	custom	11 bits	45 bits	$2^{\pm 2^{11-1}} \approx 10^{\pm 308}$	$2^{-45} \approx 3 \times 10^{-14}$
rp48	custom	11 bits	37 bits	$2^{\pm 2^{11-1}} \approx 10^{\pm 308}$	$2^{-37} \approx 7 \times 10^{-12}$
rp40	custom	11 bits	29 bits	$2^{\pm 2^{11-1}} \approx 10^{\pm 308}$	$2^{-29} \approx 2 \times 10^{-9}$
fp32	single	8 bits	24 bits	$2^{\pm 2^{8-1}} \approx 10^{\pm 38}$	$2^{-24} \approx 6 \times 10^{-8}$
rp24	custom	8 bits	16 bits	$2^{\pm 2^{8-1}} \approx 10^{\pm 38}$	$2^{-16} \approx 1 \times 10^{-5}$
rp16	custom	8 bits	8 bits	$2^{\pm 2^{8-1}} \approx 10^{\pm 38}$	$2^{-8} \approx 4 \times 10^{-3}$

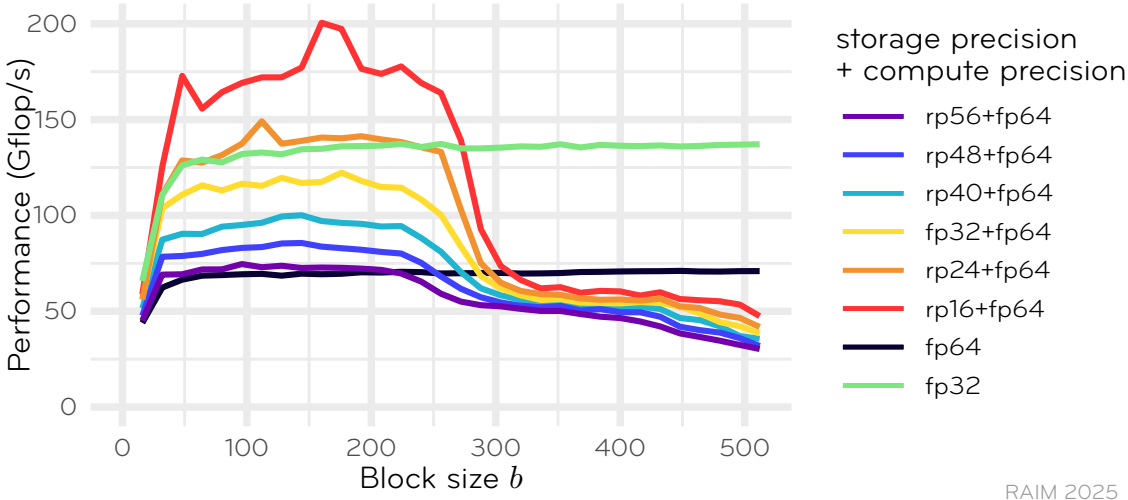
Custom formats can provide a continuum of precisions

- **further reduce storage cost**.
- they are **not natively supported** by hardware.



Low precision arithmetics – Memory accessor results

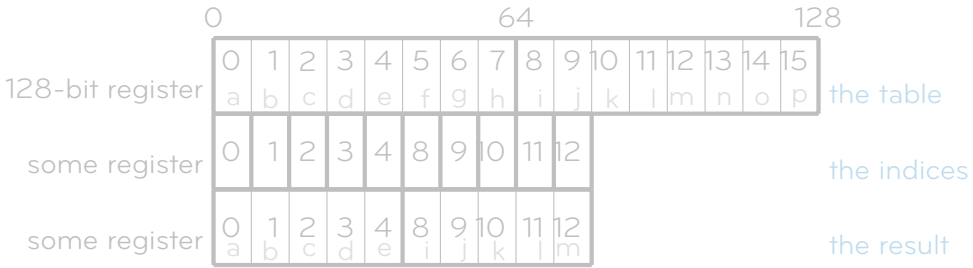
Running triangular solves in parallel.
`chirop` platform on Grid 5000 – L2 per core about 1.4MB – DDR4 RAM.



Efficient reduced-precision (de)compressor

A key ingredient to these bandwidth-bound kernels is conversion efficiency.
We wrote micro kernels through **SIMD** intrinsics to achieve satisfying performance.

The truncature is essentially a bytewise **table lookup** where the table is streamed and the indices are fixed.

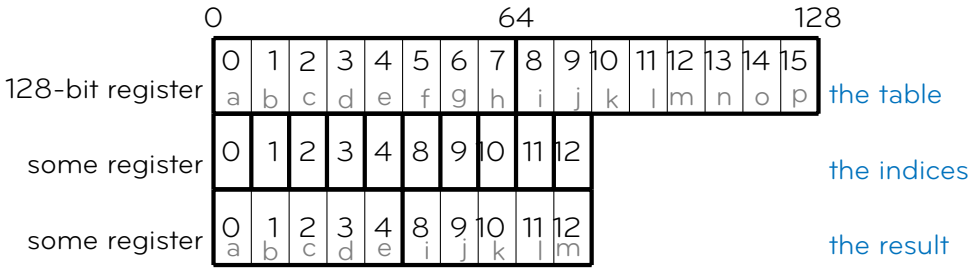


AVX512-*Vector Byte Manipulation Instructions* provides a more straigthforward instruction to implement our memory accessor.

Efficient reduced-precision (de)compressor

A key ingredient to these bandwidth-bound kernels is conversion efficiency.
We wrote micro kernels through **SIMD** intrinsics to achieve satisfying performance.

The truncature is essentially a bytewise **table lookup** where the table is streamed and the indices are fixed.

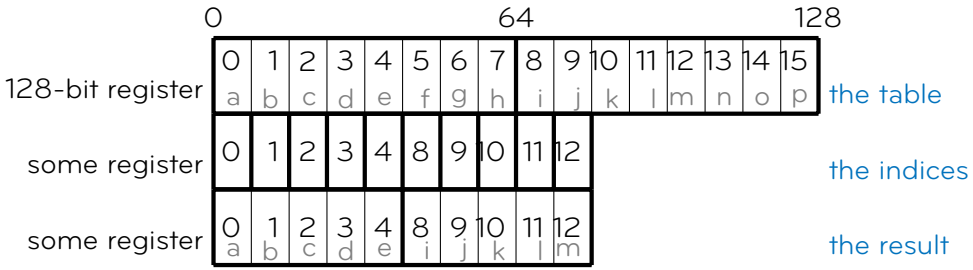


AVX512-*Vector Byte Manipulation Instructions* provides a more straigthforward instruction to implement our memory accessor.

Efficient reduced-precision (de)compressor

A key ingredient to these bandwidth-bound kernels is conversion efficiency.
We wrote micro kernels through **SIMD** intrinsics to achieve satisfying performance.

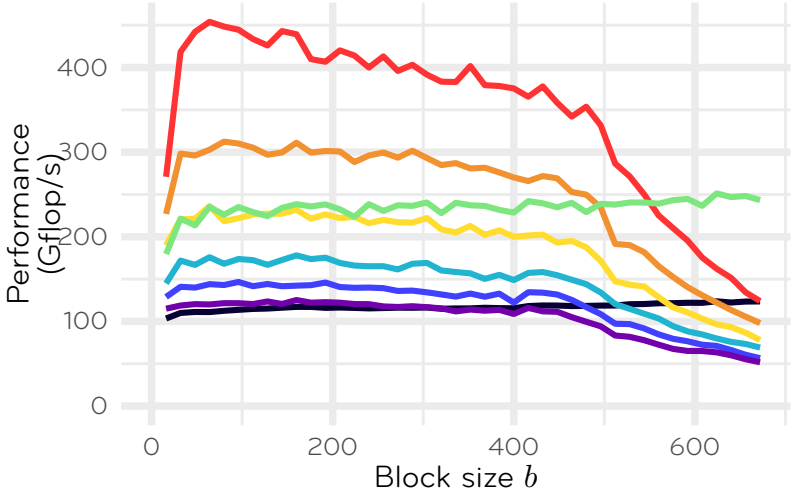
The truncature is essentially a bytewise **table lookup** where the table is streamed and the indices are fixed.



AVX512-*Vector Byte Manipulation Instructions* provides a more straigthforward instruction to implement our memory accessor.

Memory accessor results – on AMD

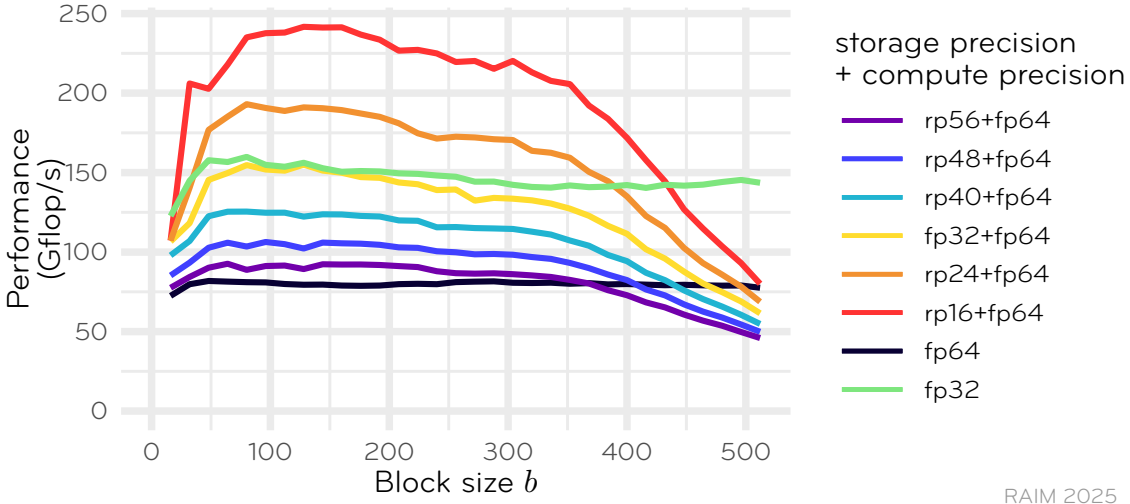
Running triangular solves in parallel.
`grdix` platform on Grid 5000 – L2 per core about 2.1MB – DDR5 RAM.



Memory accessor results – on arm

Running triangular solves in parallel.

`hydra` platform on Grid 5000 – L2 per core about 1.1MB – DDR5 RAM.



Impact on MUMPS sparse direct solver

We now have a mixed-precision direct solver (MUMPS) with **storage** and **compute** precision **decoupling** and with Block Low-Rank **compression**.

Compute precision is fp64. `Queen_4147` SuiteSparse matrix.

	Storage precisions	Accessor method	Backward error	LU factor storage (GB)	Solve time (ms)
Full-rank	fp64	–	10^{-16}	112	871
BLR	fp64	–	10^{-11}	57	505
BLR	fp64, fp32	Naive	10^{-11}	44	922
BLR	fp64, fp32	Our work	10^{-11}	44	509
BLR	fp64, fp56, ..., fp16	Naive	10^{-11}	37	1089
BLR	fp64, fp56, ..., fp16	Our work	10^{-11}	37	479

Comparing iterative refinement schemes

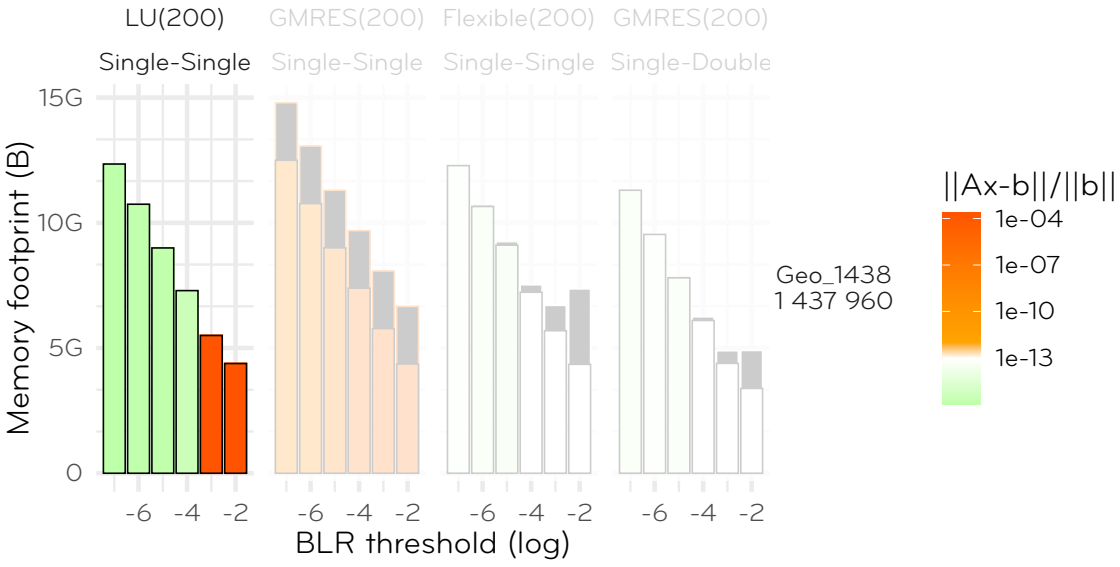
We can benchmark the following schemes to solve systems of equations.

- LU-Iterative Refinement (MUMPS)
- GMRES-Iterative Refinement (PETSc + MUMPS)
- flexible GMRES-Iterative Refinement (PETSc + MUMPS)

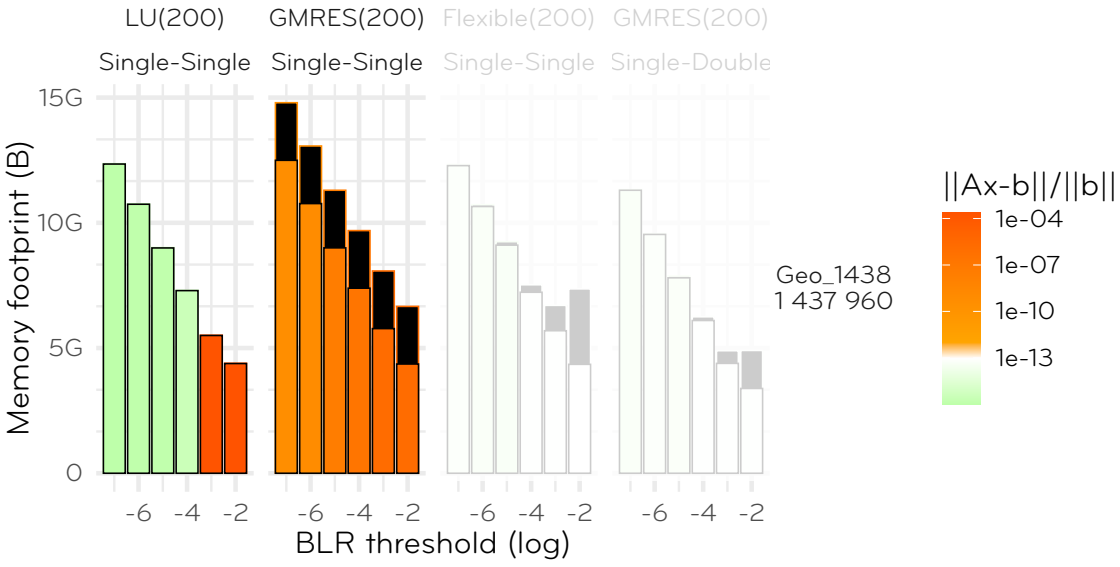
For GMRES-based IR, we compare

- the reference (Single-Single)
- "memory accessor"-enabled runs (Single+custom-Double)

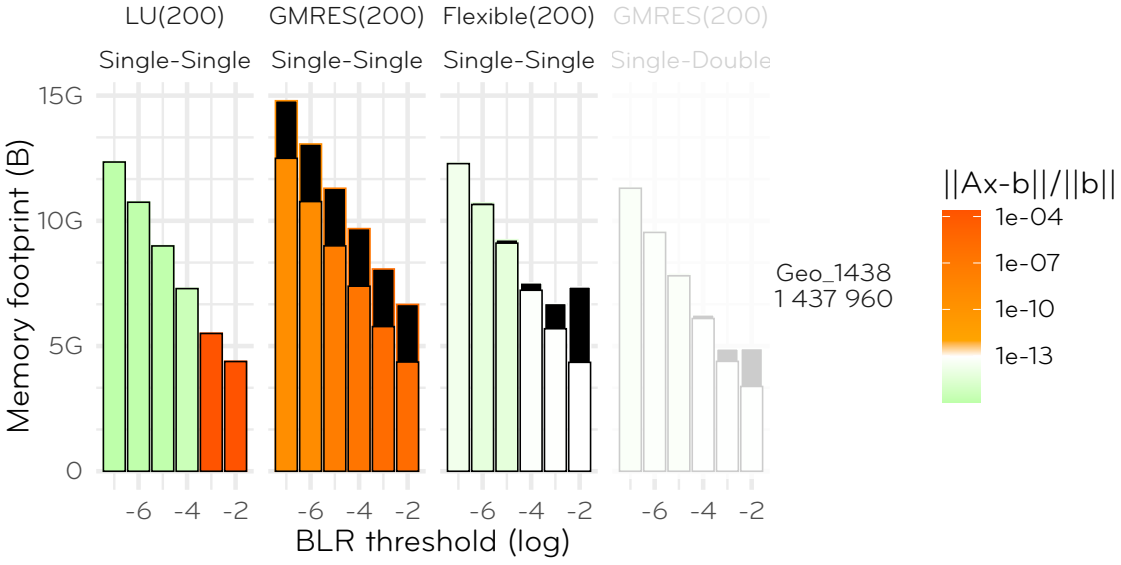
Experimental results – Target accuracy 10^{-13}



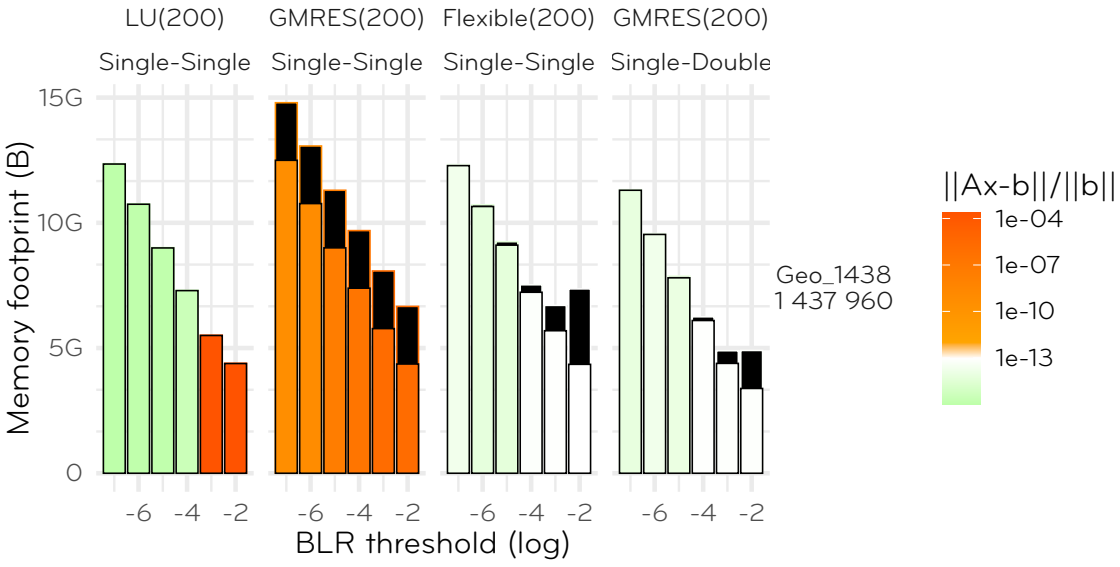
Experimental results – Target accuracy 10^{-13}



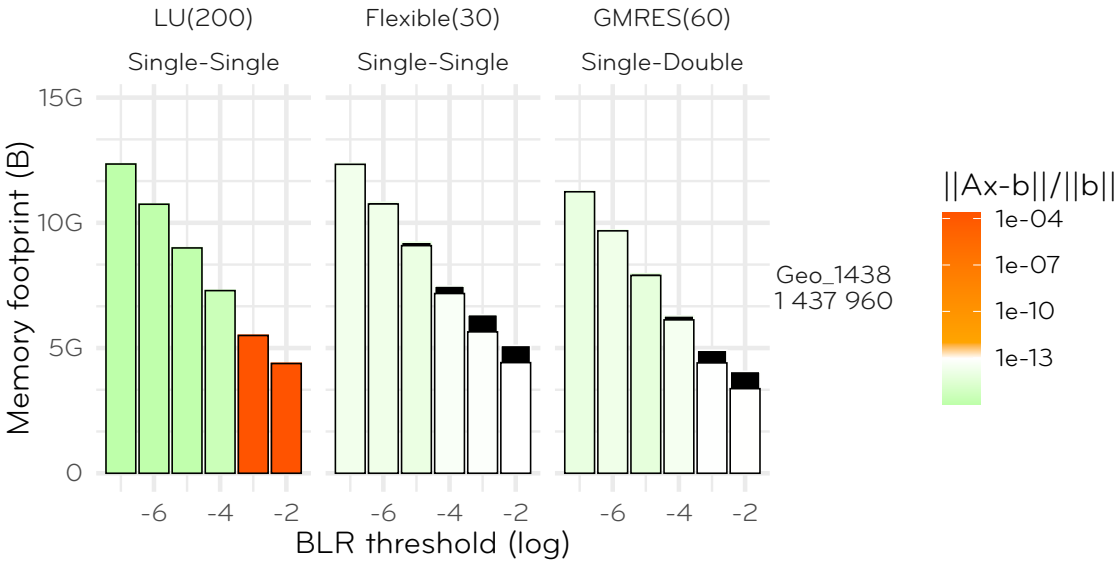
Experimental results – Target accuracy 10^{-13}



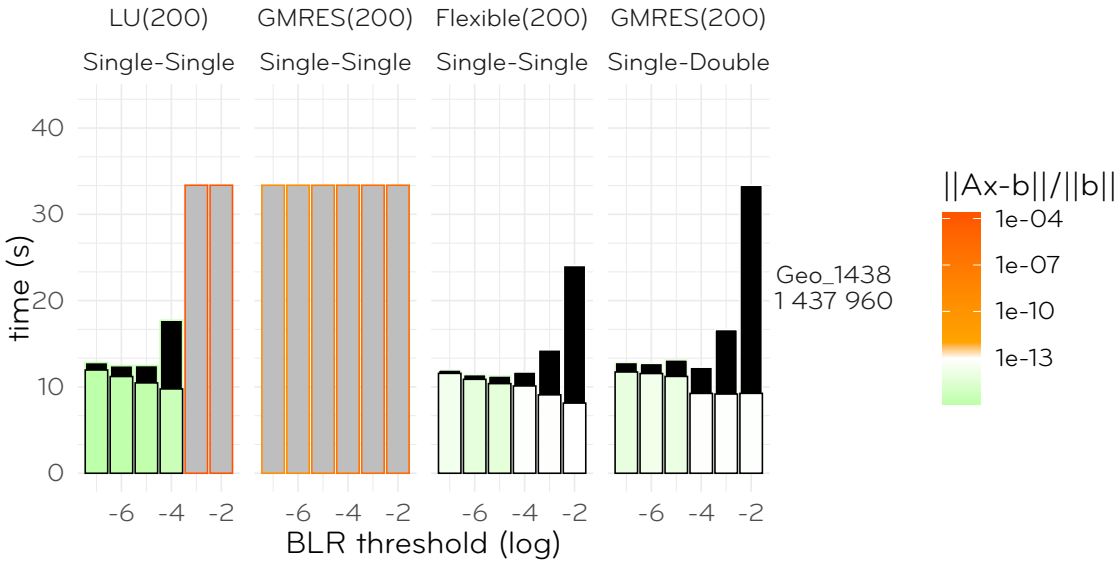
Experimental results – Target accuracy 10^{-13}



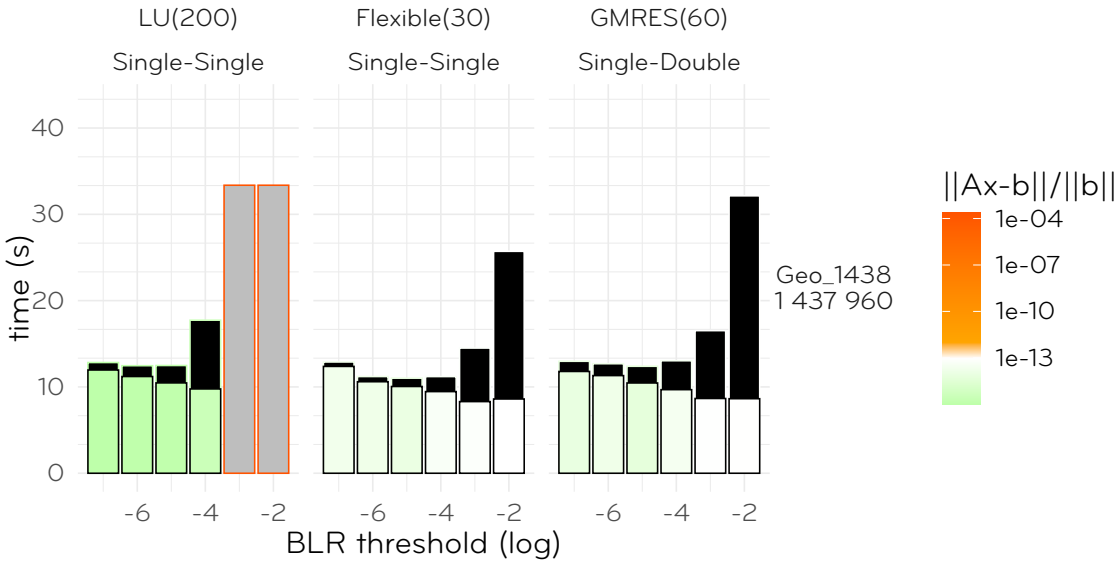
Experimental results – Target accuracy 10^{-13}



Experimental results – Target accuracy 10^{-13}



Experimental results – Target accuracy 10^{-13}



Conclusions

Memory accesor technique combined with GMRES-based Iterative Refinement enables either

- Large memory saving trading-off pure performance by solving highly compressed versions of A
- Notable memory savings and sometimes time savings by solving lightly compressed versions of A

On-going work

Optimization for small frontal matrices

Truncating floats is done through AVX512-*Vector Byte Manipulation Instructions*
⇒ One instruction, multiple (de)compressions

Impact (**Queen** 4, 1 `chirop` node, 4 MPI)			
AVX512	BLR mixed $\epsilon = 10^{-8.5}$		
	Memory	$\|Ax - b\|/\|b\|$	Time
with	35 GiB	10^{-11}	456 ms
w/out	35 GiB	10^{-11}	570 ms

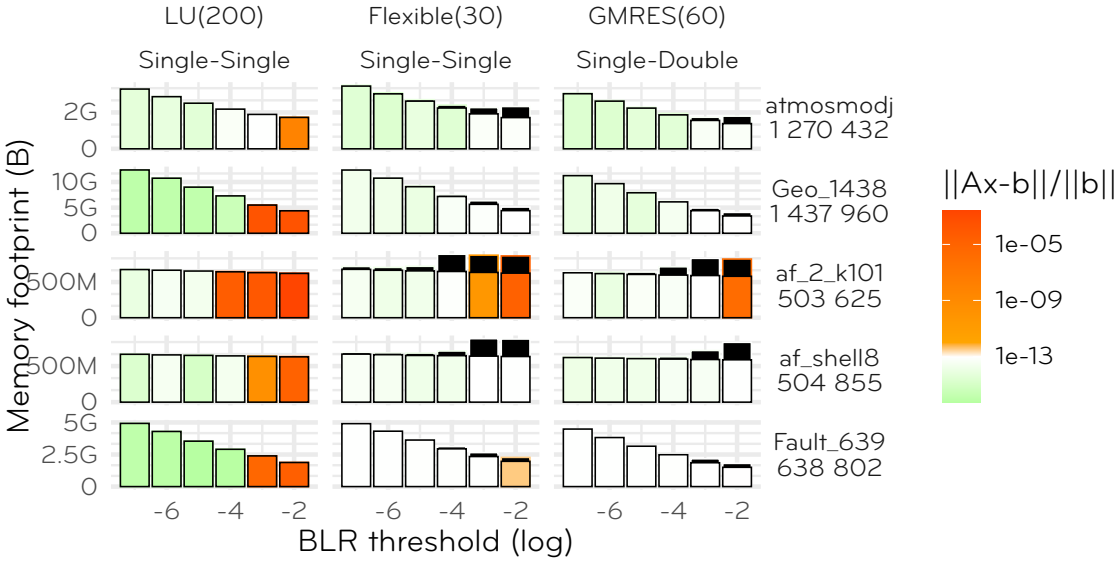
Intrinsics are essential for achieving best performance.

- AVX512-VBMI available since Zen 4 and Tigger/Ice Lake (after 2022)
- arm SVE table lookups (`svtbl_u8`) are not especially modern

MUMPS Single-precision solve step optimization

Impact (Queen, 1 chirop node, 4 MPI)				
Arithmetics		BLR $\epsilon = 4 \times 10^{-3}$		
LU factors	Solve	Memory (GB)	$\frac{\|Ax-b\|}{\|b\|}$	Time (ms)
fp64	fp64	19.0	10^{-6}	214
fp64,..., fp16	fp64	13.6	10^{-6}	192
fp32	fp32	9.5	10^{-6}	125
fp32, fp24, fp16	fp32	7.9	10^{-6}	134
fp32	fp64	8.8	10^{-6}	245
fp32, fp24, fp16	fp64	7.2	10^{-6}	235

Experimental results – Target accuracy 10^{-13} – more!



Experimental results – Target accuracy 10^{-13} – more!