REPRISES Meeting

7 April 2022

Adaptive Precision Sparse Matrix–Vector Product

and its Application to Krylov Solvers

Roméo Molina

LIP6, Sorbonne Université

Service Online, Département Informatique, IJCLab

Joint work with

Stef Graillat, **Fabienne Jézéquel**, and **Theo Mary**

Today's floating-point landscape

		Bits Signif. (t)	Exp.	Range	$u = 2^{-t}$
bfloat16	B	8	8	$10^{\pm 38}$	4×10^{-3}
fp16	H	11	5	$10^{\pm 5}$	5×10^{-4}
fp32	S	24	8	$10^{\pm 38}$	6×10^{-8}
fp64	D	53	11	$10^{\pm 308}$	1×10^{-16}
fp128	Q	113	15	$10^{\pm 4932}$	1×10^{-34}

- Low precision increasingly supported by hardware
- **Great benefits:**
  - Reduced **storage**, data movement, and communications
  - Reduced **energy** consumption ($5\times$ with fp16, $9\times$ with bfloat16)
  - Increased **speed** on emerging hardware ($16\times$ on A100 from fp32 to fp16/bfloat16)
- **Some limitations too:**
  - Low accuracy (large u)
  - Narrow range

Mixed precision algorithms

Mix several precisions in the same code with the goal of

- Getting the performance benefits of low precisions
- While preserving the accuracy and stability of high precision

Terminology varies: Mixed precision, Multiprecision, Adaptive precision, Variable precision, Transprecision, Dynamic precision, . . .

Mixed precision algorithms

Mix several precisions in the same code with the goal of

- Getting the performance benefits of low precisions
- While preserving the accuracy and stability of high precision

Terminology varies: Mixed precision, Multiprecision, Adaptive precision, Variable precision, Transprecision, Dynamic precision, ...

How to select the right precision for the right variable/operation

- **Precision tuning:** autotuning based on the source code, my thesis area: CADNA / PROMISE...
  - ▲ Does not need any understanding of what the code does
  - ▼ Does not have any understanding of what the code does
- **This work:** another point of view, **exploit as much as possible the knowledge we have about the code**

Adaptive precision algorithms

- Given an algorithm and a prescribed accuracy ε, adaptively select the minimal precision for each computation

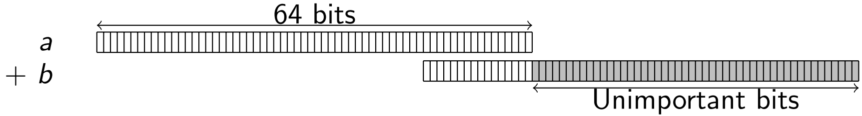
$\Rightarrow$ **Why does it make sens to make the precision vary?**

Adaptive precision algorithms

- Given an algorithm and a prescribed accuracy ε, adaptively select the minimal precision for each computation

$\Rightarrow$ **Why does it make sens to make the precision vary?**

- Because not all computations are equally "important"!
  Example:



and small elements produce small errors :

$$|\,\mathrm{fl}(a \operatorname{op} b) - a \operatorname{op} b| \le u|a \operatorname{op} b|, \qquad \operatorname{op} \in \{+, -, *, \div\}$$

$\Rightarrow$ **Opportunity for mixed precision:** adapt the precisions to the data at hand by storing and computing "less important" (usually smaller) data in lower precision

Adaptive precision at the variable level?

- Pushing adaptive precision to the extreme: can we benefit from storing **each variable** in a (possibly) different precision?
- Example: $Ax = b$ with adaptive precision for each A_{ij}
  - **Is it worth it?**
    Need to have elements of widely different magnitudes
  - **Is it practical?**
    Probably not for compute-bound applications, but could it work for memory-bound ones?

$\Rightarrow$ Natural candidate: **sparse matrices**

Sparse matrix–vector product (SpMV)

$$y = Ax,\ A \in \mathbb{R}^{m \times n}$$

for $i = 1: m$ **do**
    $y_i = 0$
    for $j \in nnz_i(A)$ **do**
        $y_i = y_i + a_{ij}x_j$
    end for
end for

- Standard error analysis for $y = Ax$ performed in a uniform precision ε gives,

$$|\widehat{y}_i - y_i| \le n_i \varepsilon \sum_{j \in nnz_i(A)} |a_{ij}x_j|$$

- **Idea:** store elements of A in a precision inversely proportional to their magnitude (**smaller elements in lower precision**)

Adaptive precision SpMV

```
for i = 1: m do
    y_i = 0
    for k = 1: p do
        y_i^(k) = 0
        for j ∈ nnz_i(A) do
            if a_ij x_j ∈ B_ik then
                y_i^(k) = y_i^(k) + a_ij x_j at precision u_k
            end if
        end for
        y_i = y_i + y_i^(k)
    end for
end for
```

- Split row i of A into p buckets B_{ik} and sum elements of B_{ik} in precision u_k
- Error analysis: $|\widehat{y}_i^{(k)} - y_i^{(k)}| \le n_i^{(k)} u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j|$

Building the buckets

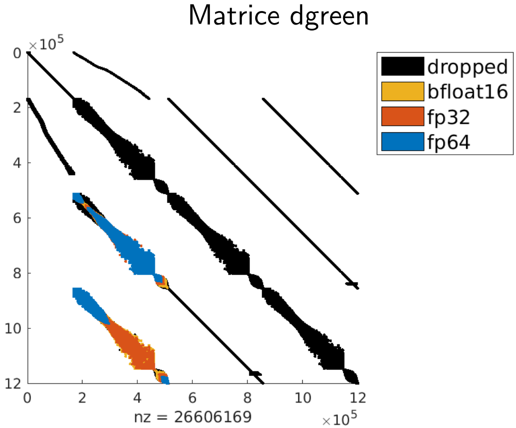
- $|\widehat{y}_i^{(k)} - y_i^{(k)}| \le n_i^{(k)} u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j|$

$\Rightarrow$ Build the buckets such that $u_k \sum_{a_{ij}x_j \in B_{ik}} |a_{ij}x_j| \approx \varepsilon \sum_j |a_{ij}x_j|$

- By setting B_{ik} to the interval $(\varepsilon\beta_i / u_{k+1}, \varepsilon\beta_i / u_k]$, we obtain $|\widehat{y}_i^{(k)} - y_i^{(k)}| \le n_i^{(k)} \varepsilon\beta_i$ and so $|\widehat{y}_i - y_i| \le n_i \varepsilon \beta_i$

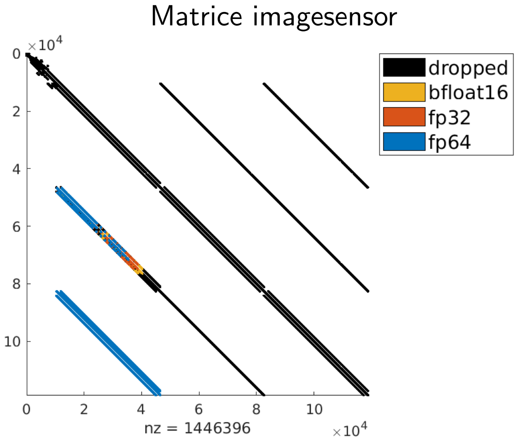
- Two possible choices for β_i:
  - $\beta_i = \sum_j |a_{ij}x_j| \Rightarrow$ guarantees $O(\varepsilon)$ componentwise error: $|\widehat{y}_i - y_i| \le n\epsilon \sum_j |a_{ij}x_j| \quad \forall i \in \{1, \ldots, n\}$
  - $\beta_i = \|A\|\|x\| \Rightarrow$ guarantees $O(\varepsilon)$ normwise error: $|\widehat{y}_i - y_i| \le n\epsilon \|A\|\|x\|$

Visualise mixed-precision gains



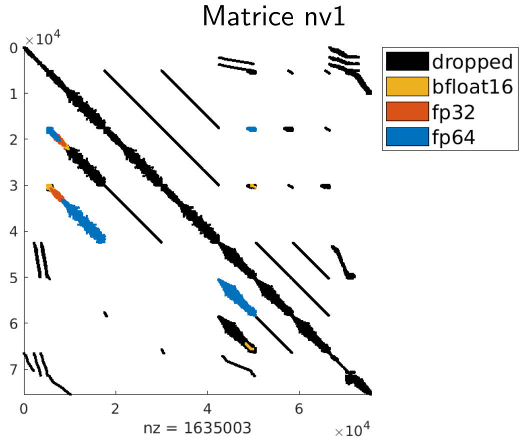
For some matrices, many elements can be dropped that leads to major gains.

Visualise mixed-precision gains



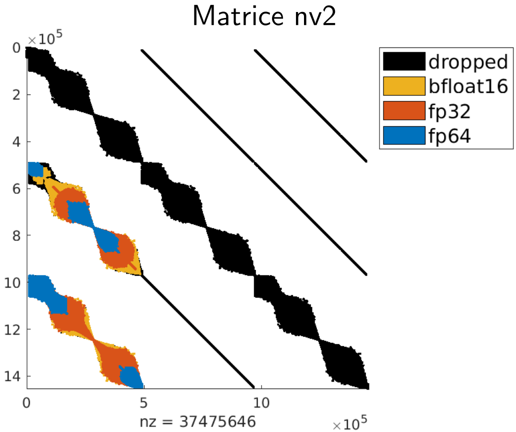
For some matrices, many elements can be dropped that leads to major gains.

Visualise mixed-precision gains



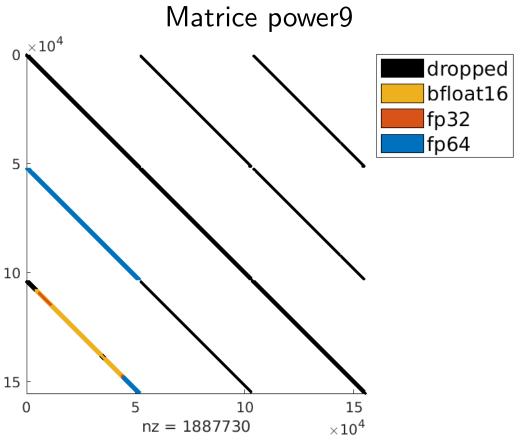
For some matrices, many elements can be dropped that leads to major gains.

Visualise mixed-precision gains



For some matrices, many elements can be dropped that leads to major gains.

Visualise mixed-precision gains



For some matrices, many elements can be dropped that leads to major gains.

SpMV experimental settings

- 34 matrices coming from SuiteSparse collection and industrial partners with at most 166M non-zeros

SpMV experimental settings

- 34 matrices coming from SuiteSparse collection and industrial partners with at most 166M non-zeros
- 3 different **accuracy targets**

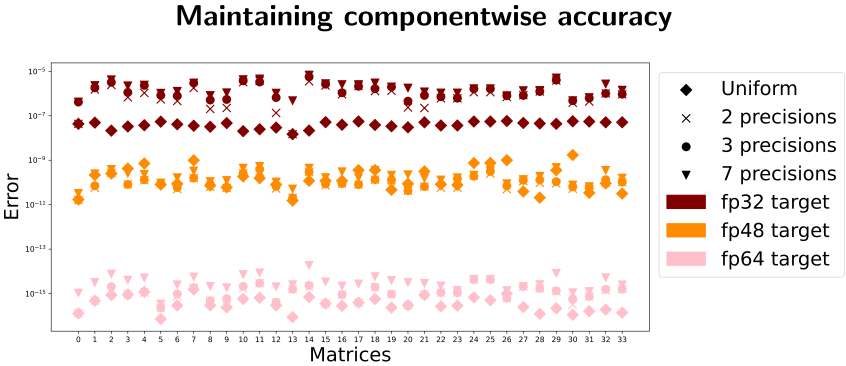
Target	$u = 2^{-t}$
fp32	6×10^{-8}
"fp48"	8×10^{-12}
fp64	1×10^{-16}

SpMV experimental settings

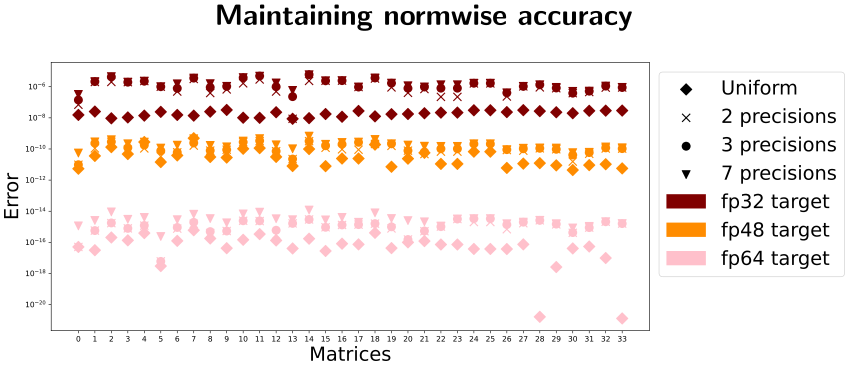
Possibility to use

- 2 **precisions**: fp32, fp64
- 3 **precisions**: bfloat16, fp32, fp64
- 7 **precisions**: bfloat16, "bfloat24", fp32, fp64, "fp40", "fp48", "fp56"

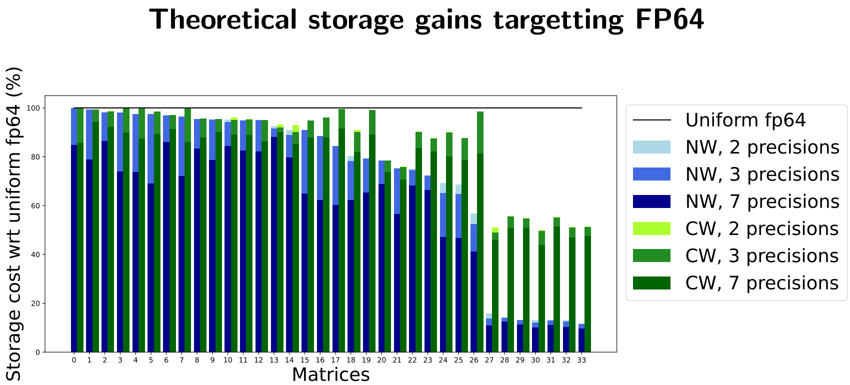
	Bits	
	Mantissa	Exponent
bfloat16	8	8
"bfloat24"	8	8
fp32	24	8
"fp40"	29	11
"fp48"	37	11
"fp56"	45	11
fp64	53	11



Adaptive methods preserve an accuracy close to the accuracy of uniform methods.



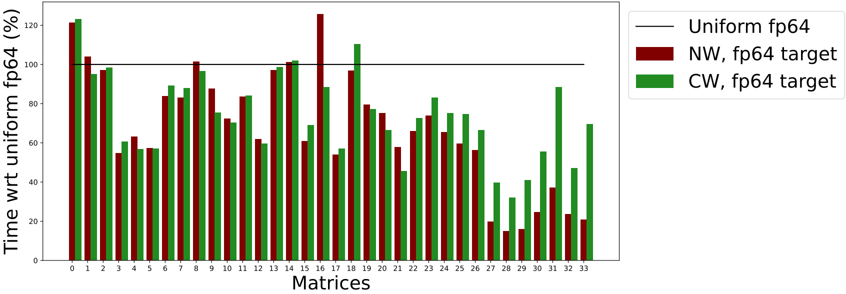
Adaptive methods preserve an accuracy close to the accuracy of uniform methods.



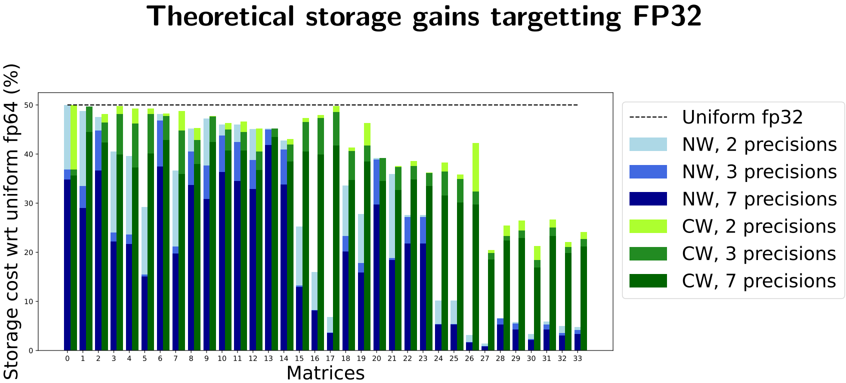
Up to **88%** of storage reduction

SpMV experiments

Actual time gains targetting FP64



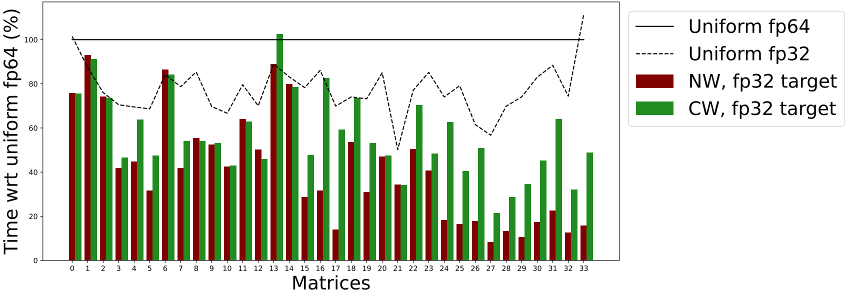
Up to **85%** of time reduction



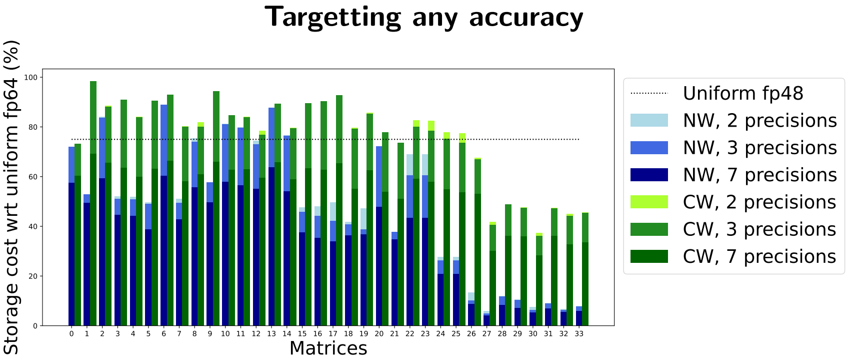
Up to **97%** of storage reduction

SpMV experiments

Actual time gains targetting FP32

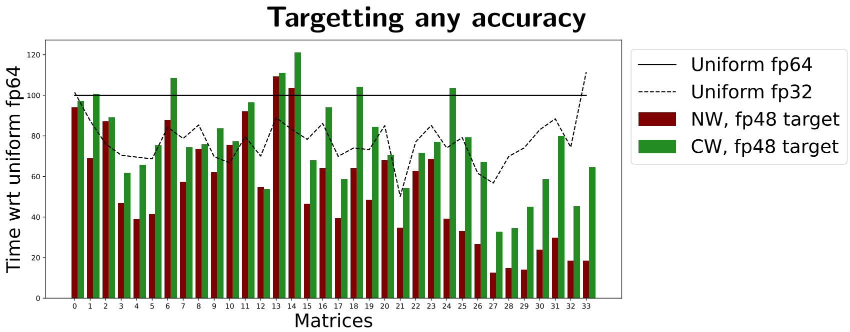


Up to **88%** of time reduction



We are able to target any kind of accuracy with only natively supported precisions.

SpMV experiments



We are able to target any kind of accuracy with only natively supported precisions.

Plug SpMV into GMRES

Performance of GMRES rely on SpMV

$r = b - Ax_0$
$\beta = \|r\|_2$
$q_1 = r/\beta$
for $k = 1, 2, \ldots$ **do**
  $y = Aq_k$
  for $j = 1 : k$ **do**
    $h_{jk} = q_j^T y$
    $y = y - h_{jk} q_j$
  end for
  $h_{k+1,k} = \|y\|_2$
  $q_{k+1} = y / h_{k+1,k}$
  Solve the least squares problem $\min_{c_k} \|Hc_k - \beta e_1\|_2$
  $x_k = x_0 + Q_k c_k$
end for

How does the adaptive method affect the convergence?

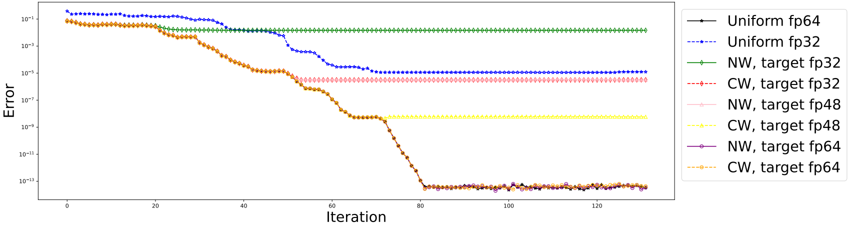
Application to GMRES: experimental settings

Assessing the potential of adaptive precision for GMRES is not straightforward:

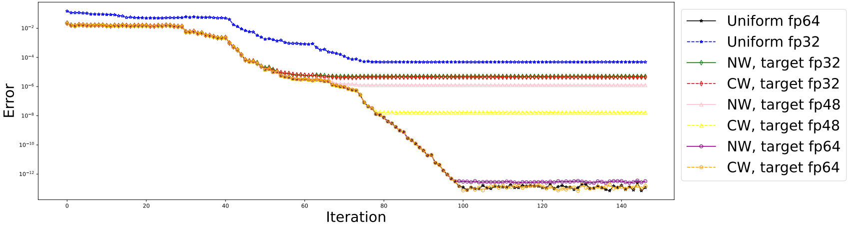
- **Highly matrix dependent**, need to cover a wide range of applications
- For a given matrix, **hard to know what a good accuracy is**
  - What storage precision?
  - What tolerance threshold for GMRES convergence?
  - Normwise or componentwise stable SpMV?
  - How small should the error be?
- Comparison further muddled by possible use of
  - Preconditioners
  - Iterative refinement (i.e., restarted GMRES)

Application to GMRES: maintaining convergence scheme

Adaptive GMRES follows convergence shemes of uniform GMRES

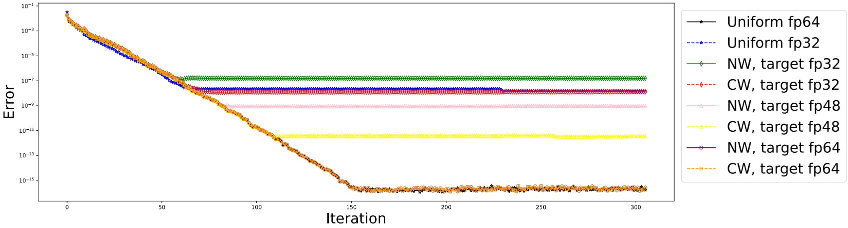


Adaptive GMRES follows convergence shemes of uniform GMRES



Application to GMRES: maintaining convergence scheme

Adaptive GMRES follows convergence shemes of uniform GMRES



Conclusion: take-home messages

- **Adaptive precision SpMV**
- Application to Krylov solvers: significant reductions of the data movement at equivalent accuracy
- Article in preparation

Thank you! Any questions?