Pushing the Roofline: A Modular Precision Ecosystem Based on a Memory Accessor

NHR PerfLab Seminar Talk

May 18th 2021

Helmholtz Young Investigator Group FiNE

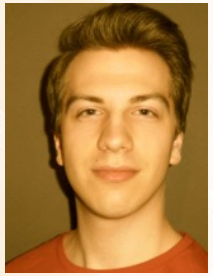
Steinbuch Centre for Computing (SCC), Karlsruhe Institute of Technology

Hartwig Anzt

Isha Aggarwal

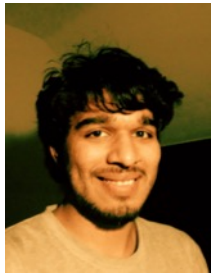
Terry Cojean

Fritz Göbel



T. Grützmacher

Aditya Kashi



Pratik Nayak

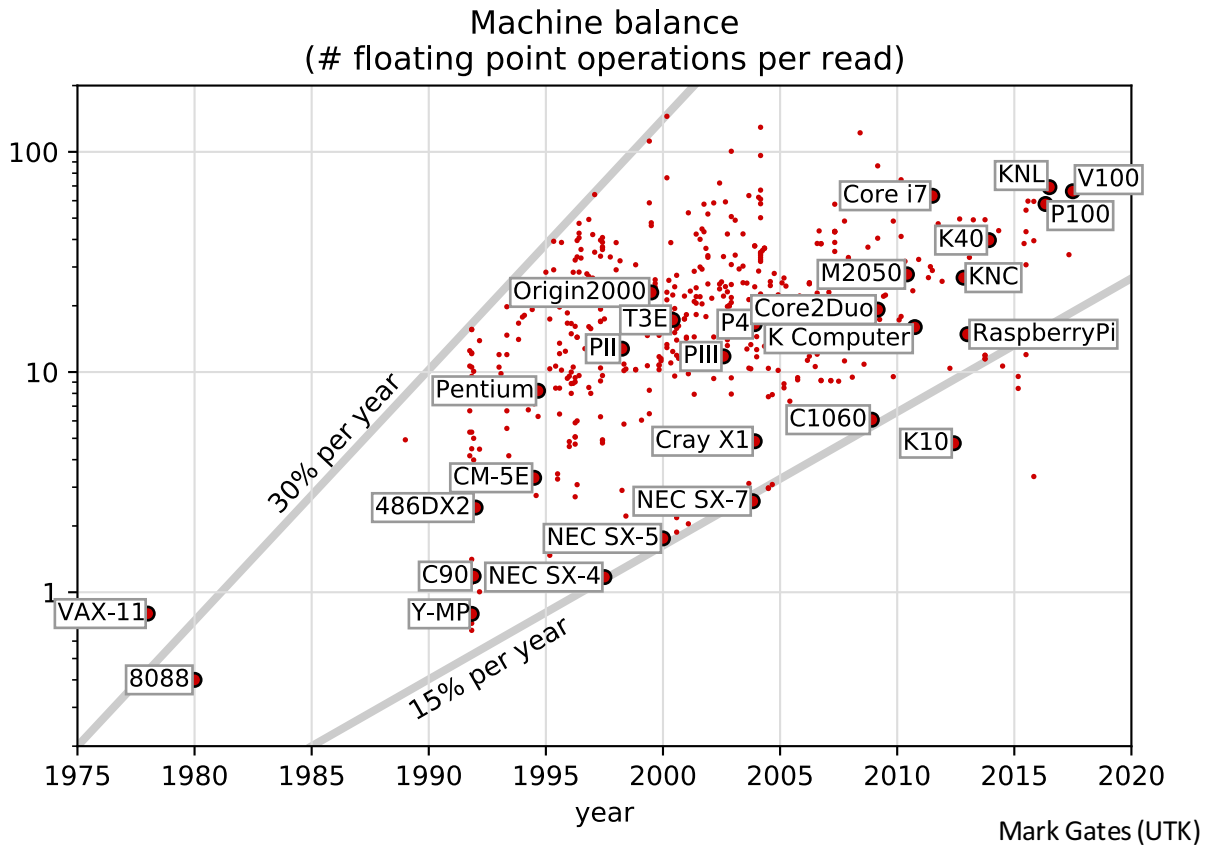
Gregor Olenik

Tobias Ribizel

Mike Tsai

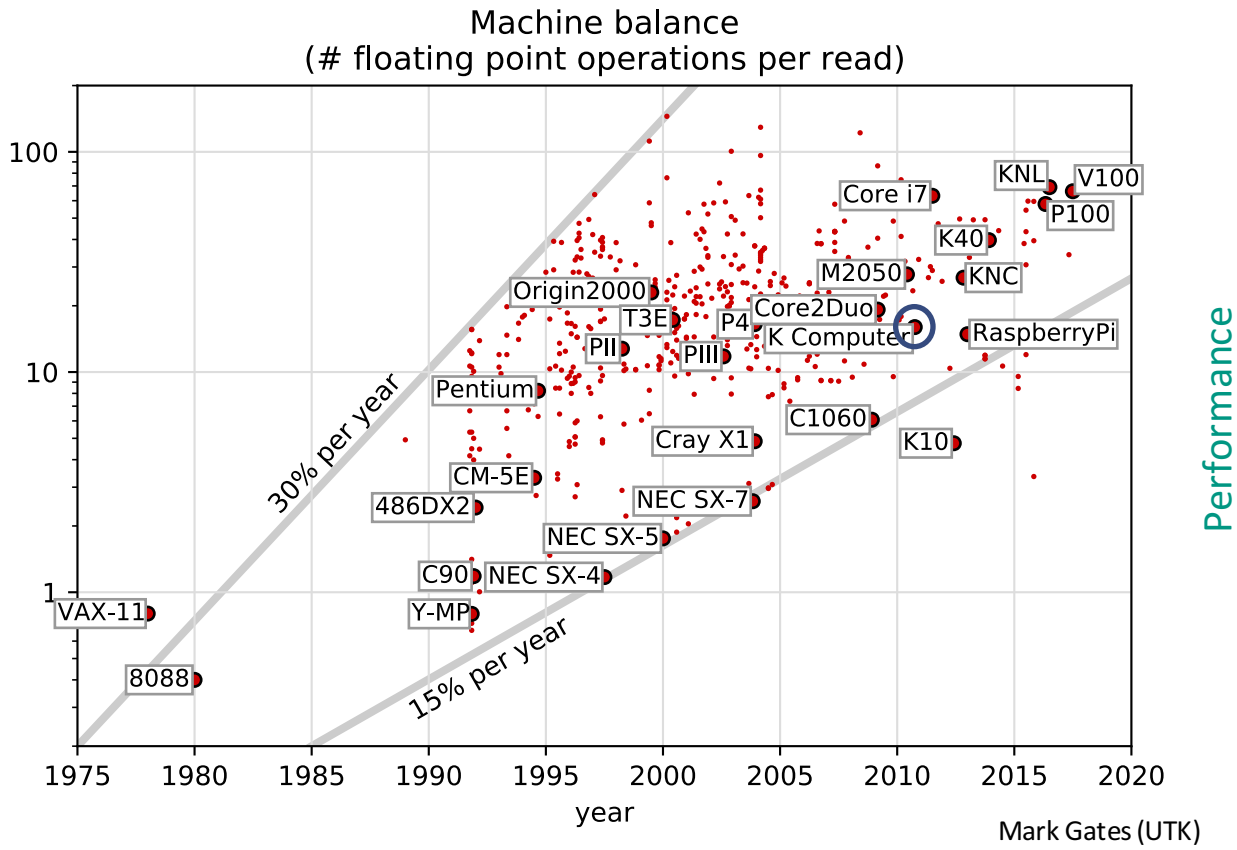
Collaborations with P. Luszczek, E. Quintana-Orti, S. Li, J. Dongarra, U.Yang, E. Boman, J. Loe, E. Carson, N. Higham, G. Wellein, E. Chow, G. Hager, J. Aliaga, A. Tomas, S. Tomov, G. Flegar, S. Pranesh and many others...

Trends in the HPC Landscape

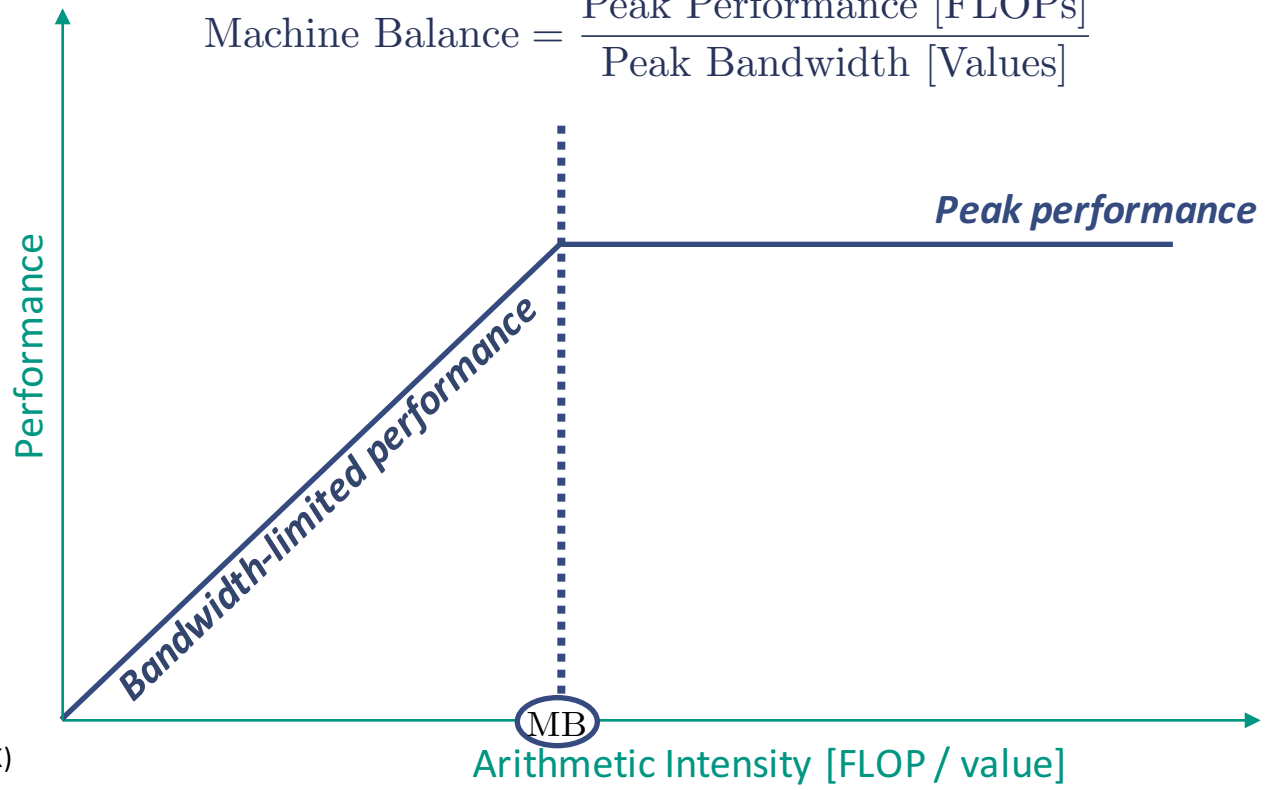


- Compute performance grows faster than memory bandwidth.

Trends in the HPC Landscape

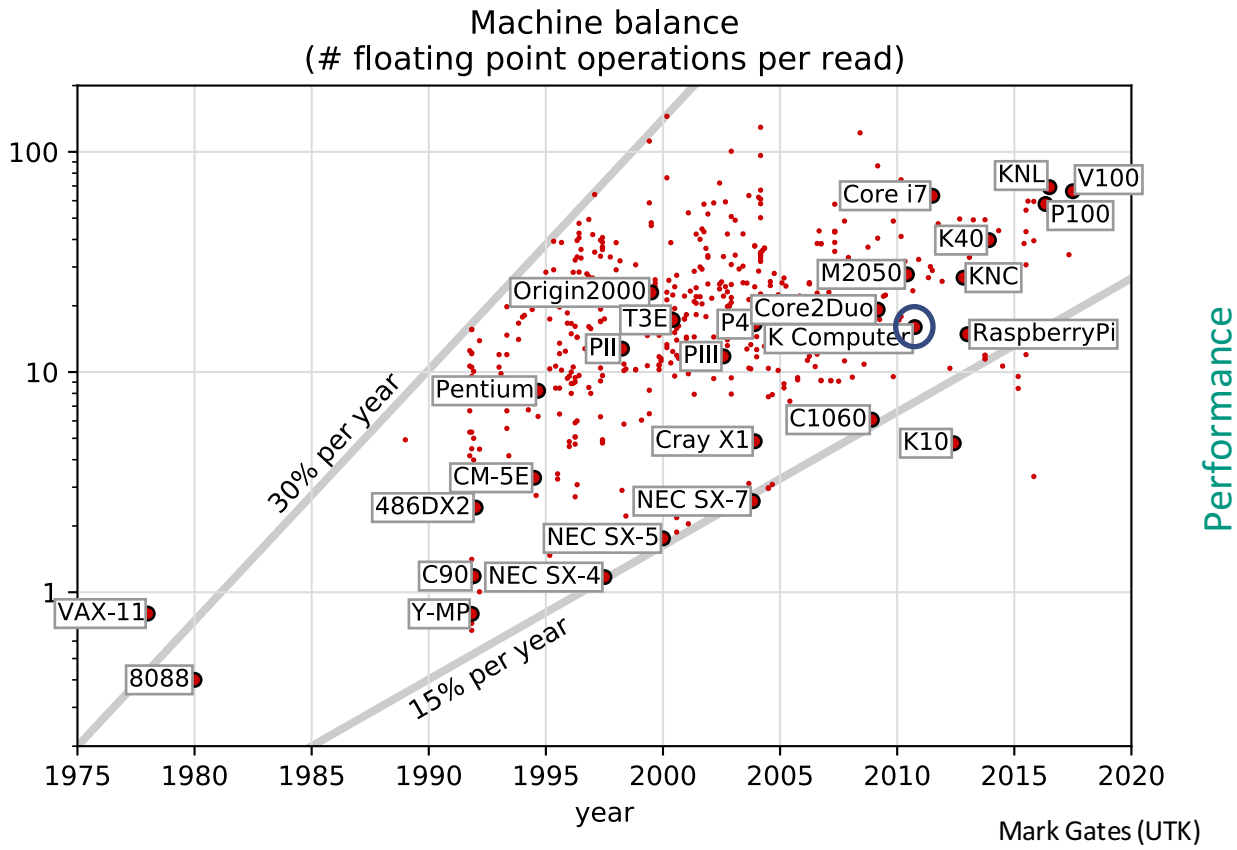


$$\text{Machine Balance} = \frac{\text{Peak Performance [FLOPs]}}{\text{Peak Bandwidth [Values]}}$$

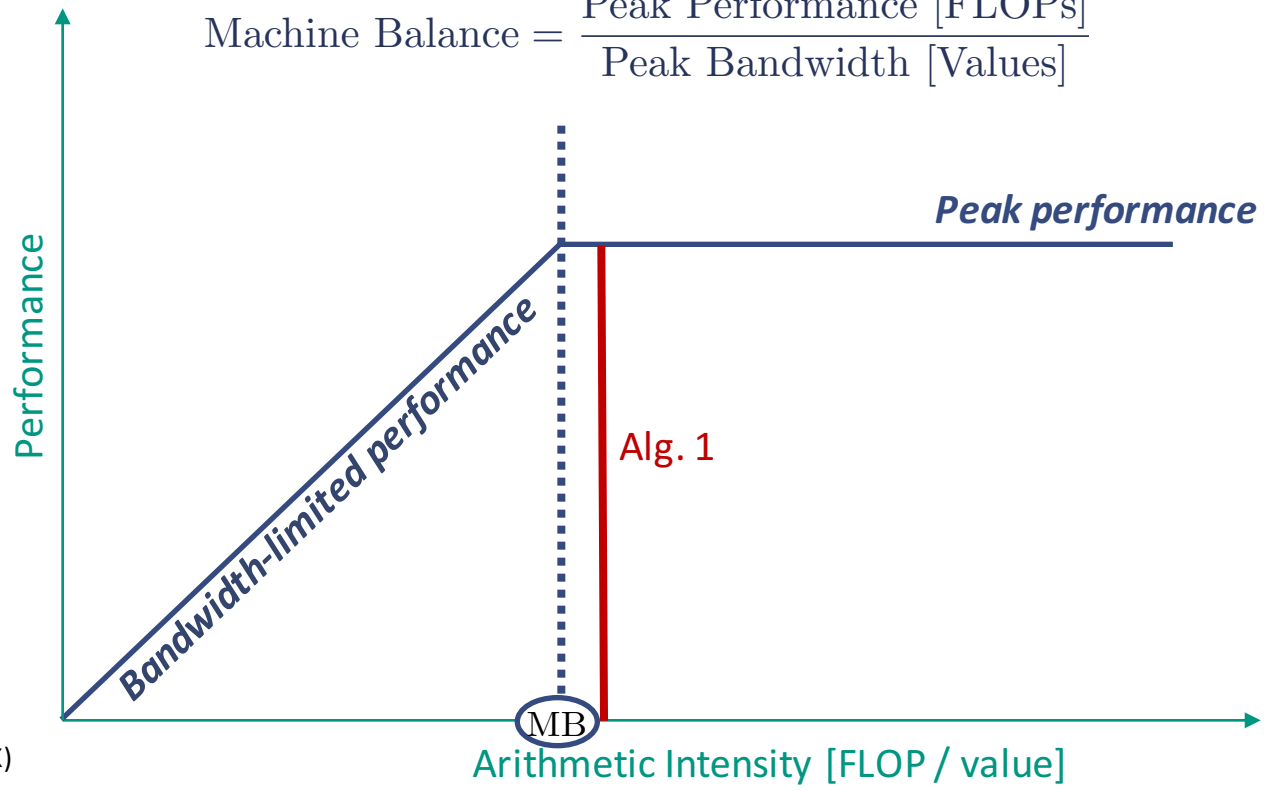


- Compute performance grows faster than memory bandwidth.

Trends in the HPC Landscape

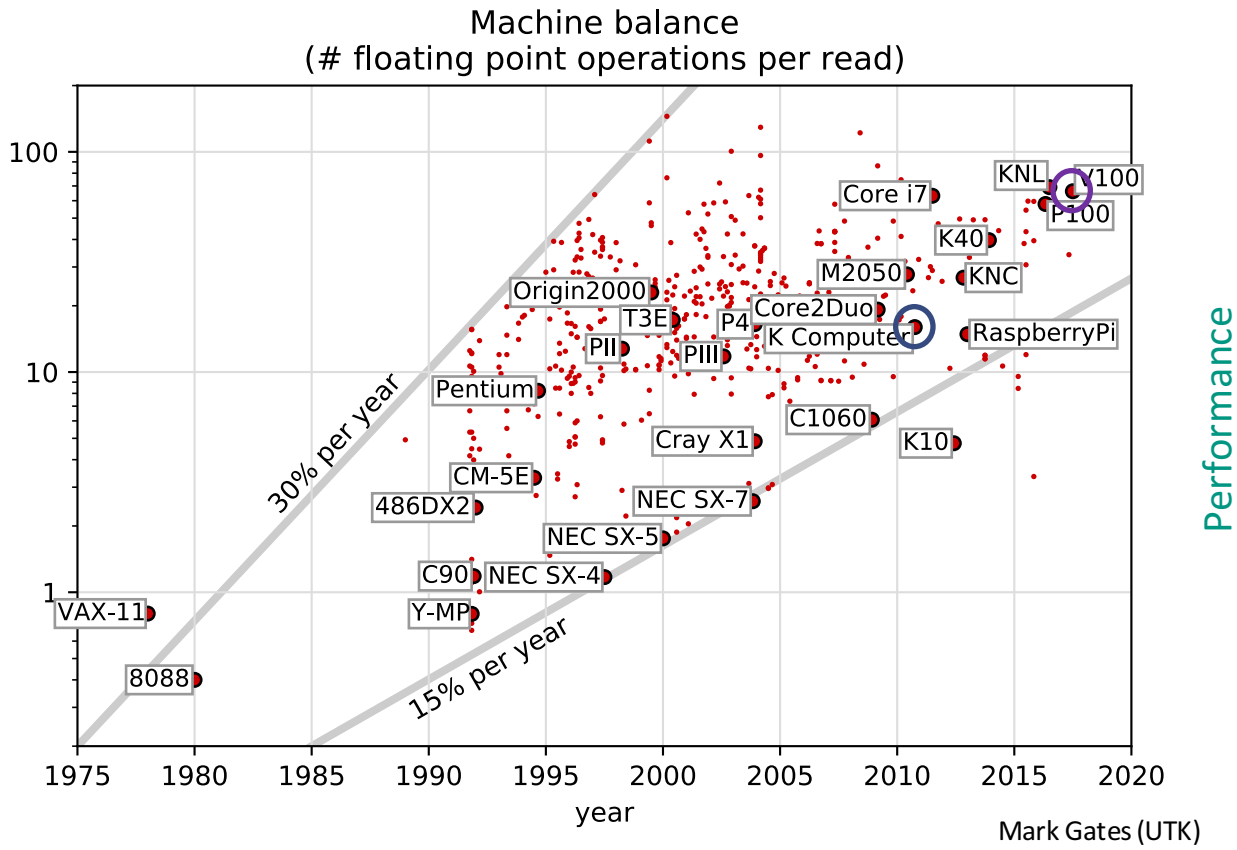


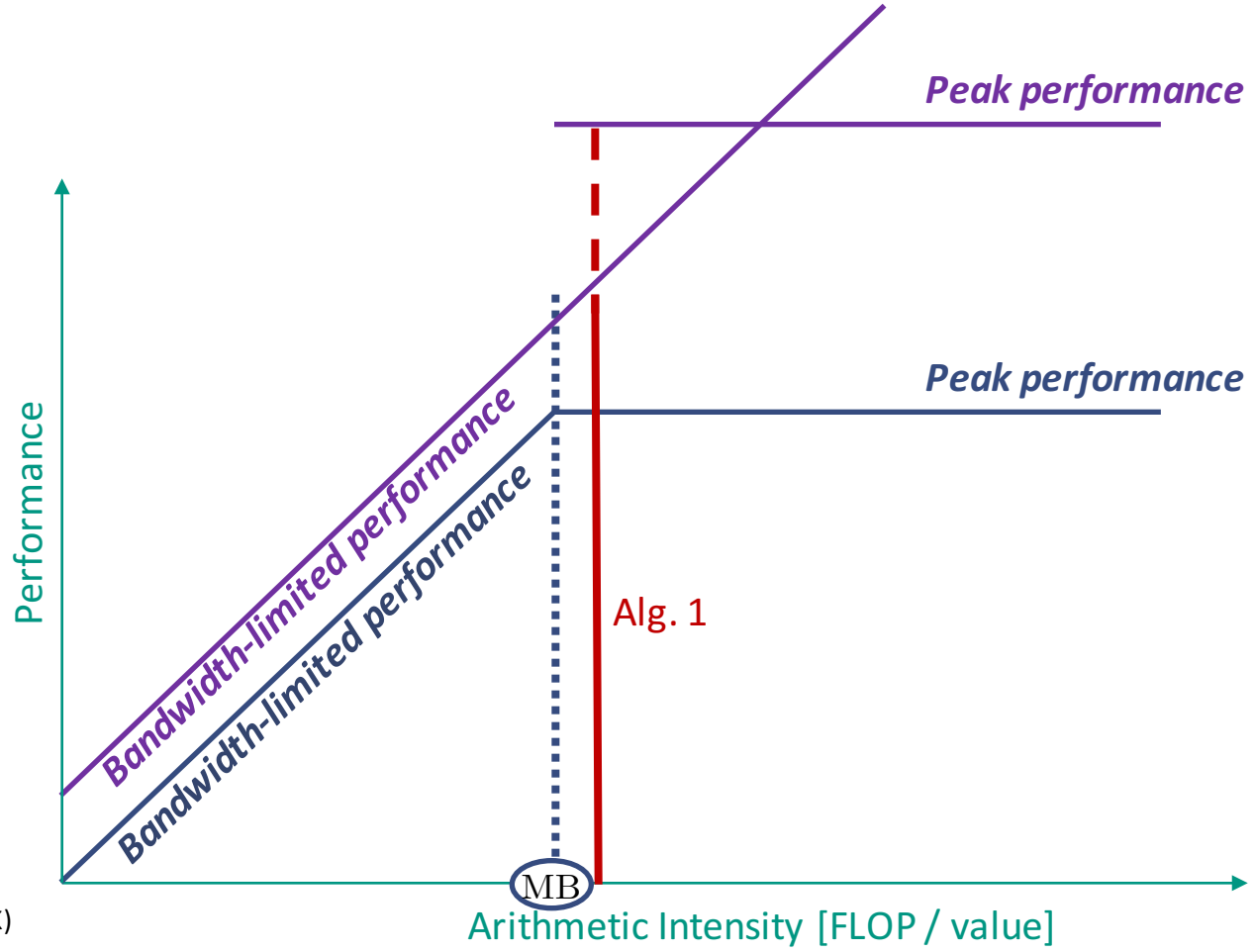
$$\text{Machine Balance} = \frac{\text{Peak Performance [FLOPs]}}{\text{Peak Bandwidth [Values]}}$$



- Compute performance grows faster than memory bandwidth.

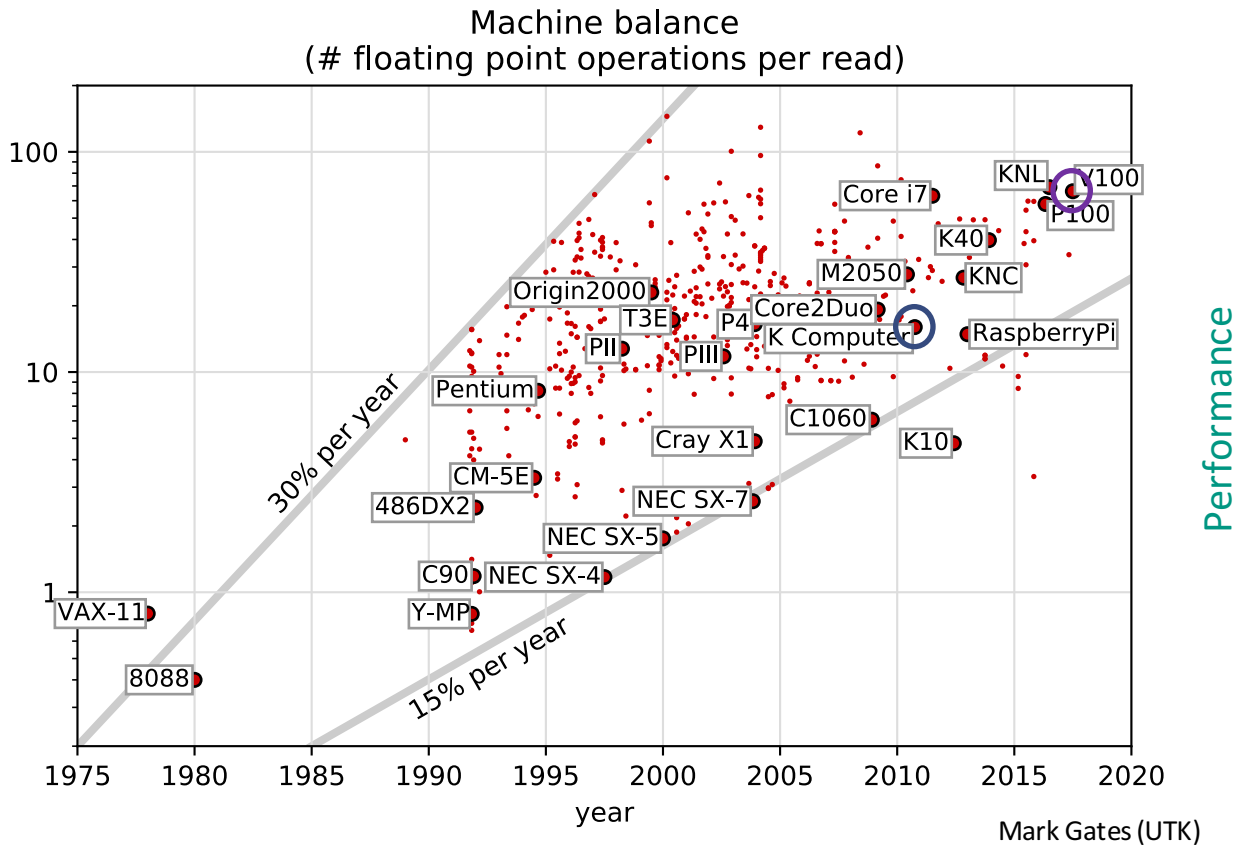
Trends in the HPC Landscape

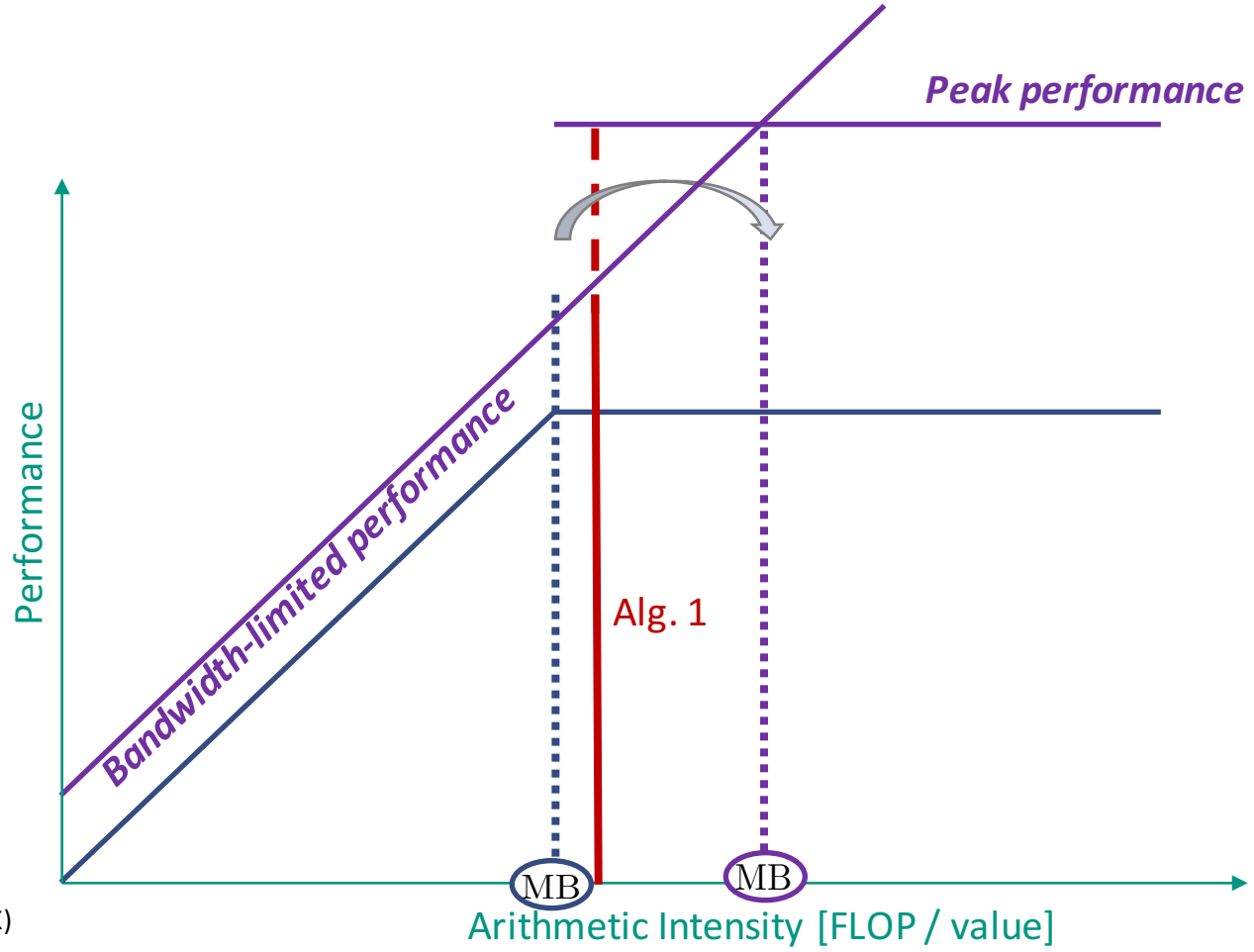




- Compute performance grows faster than memory bandwidth.
- Communication is the new bottleneck!

Trends in the HPC Landscape



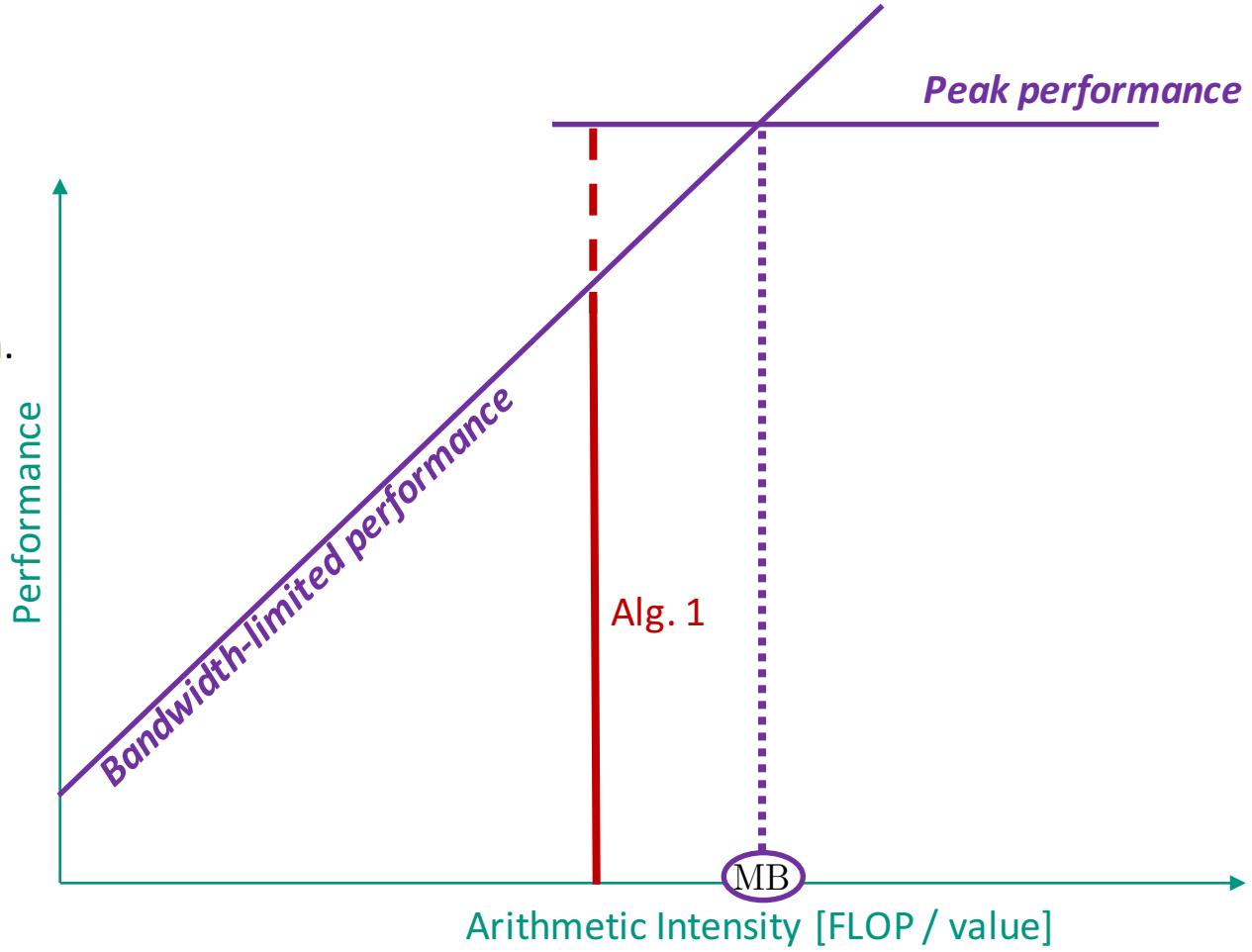


- Compute performance grows faster than memory bandwidth.

- Communication is the new bottleneck!
- The memory wall endangers continued performance growth.

Pushing the Roofline

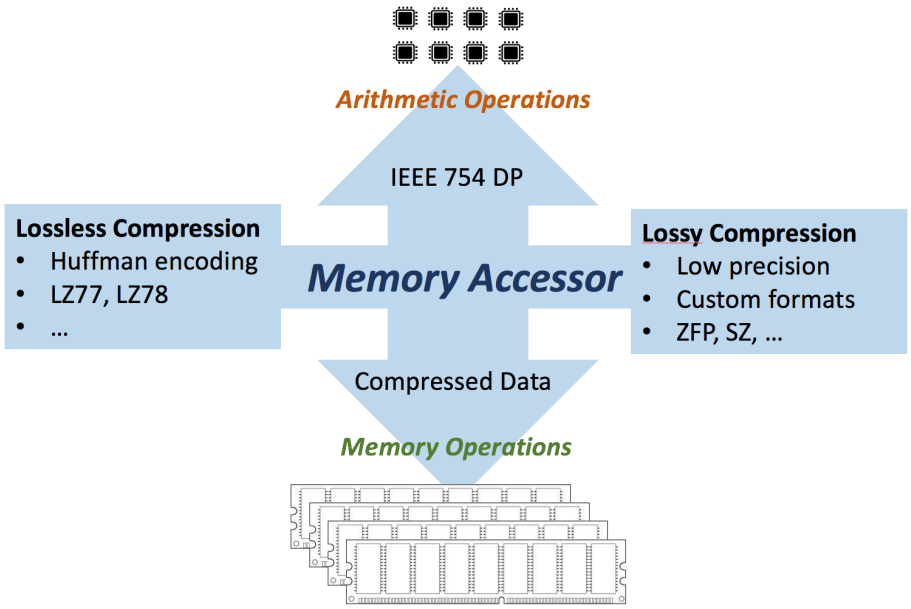
- Traditionally, we use a strong coupling between the precision formats used for arithmetic operations and storing data.
- The arithmetic operations are free, use high precision formats.
- **Data access** should be as cheap as possible, **reduced precision**.

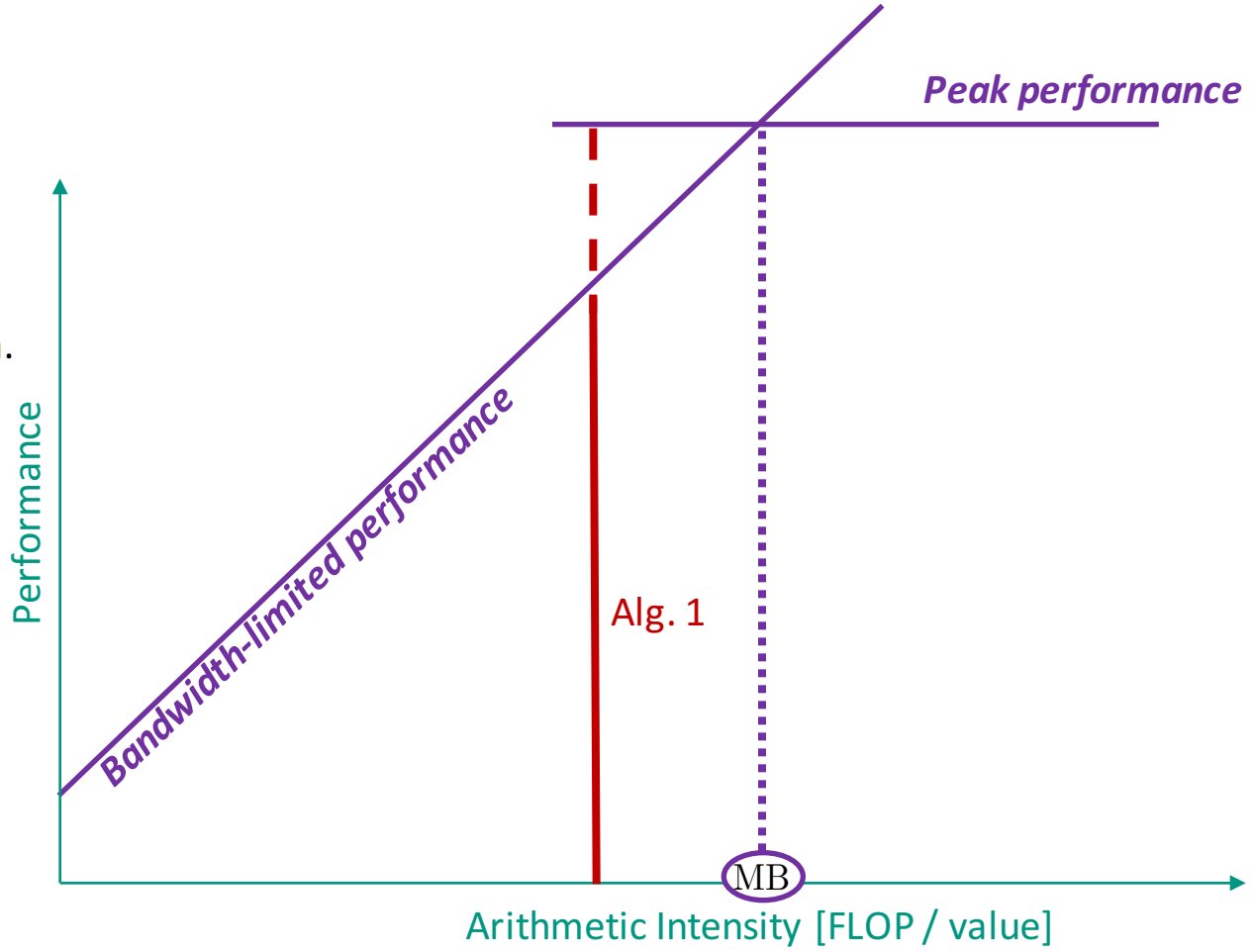


- Communication is the new bottleneck!
- The memory wall endangers continued performance growth.

Pushing the Roofline

- Traditionally, we use a strong coupling between the precision formats used for arithmetic operations and storing data.
- The arithmetic operations are free, use high precision formats.
- **Data access** should be as cheap as possible, **reduced precision**.

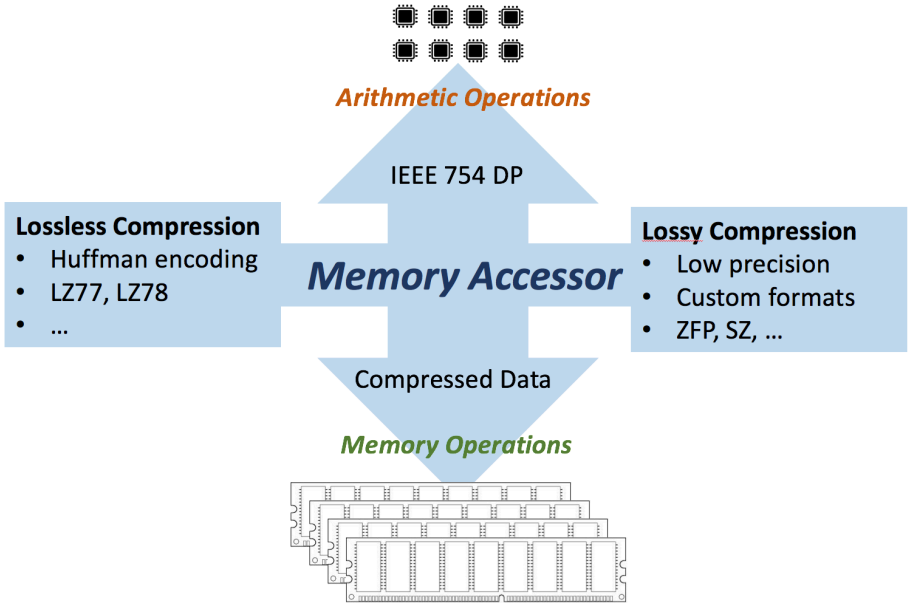


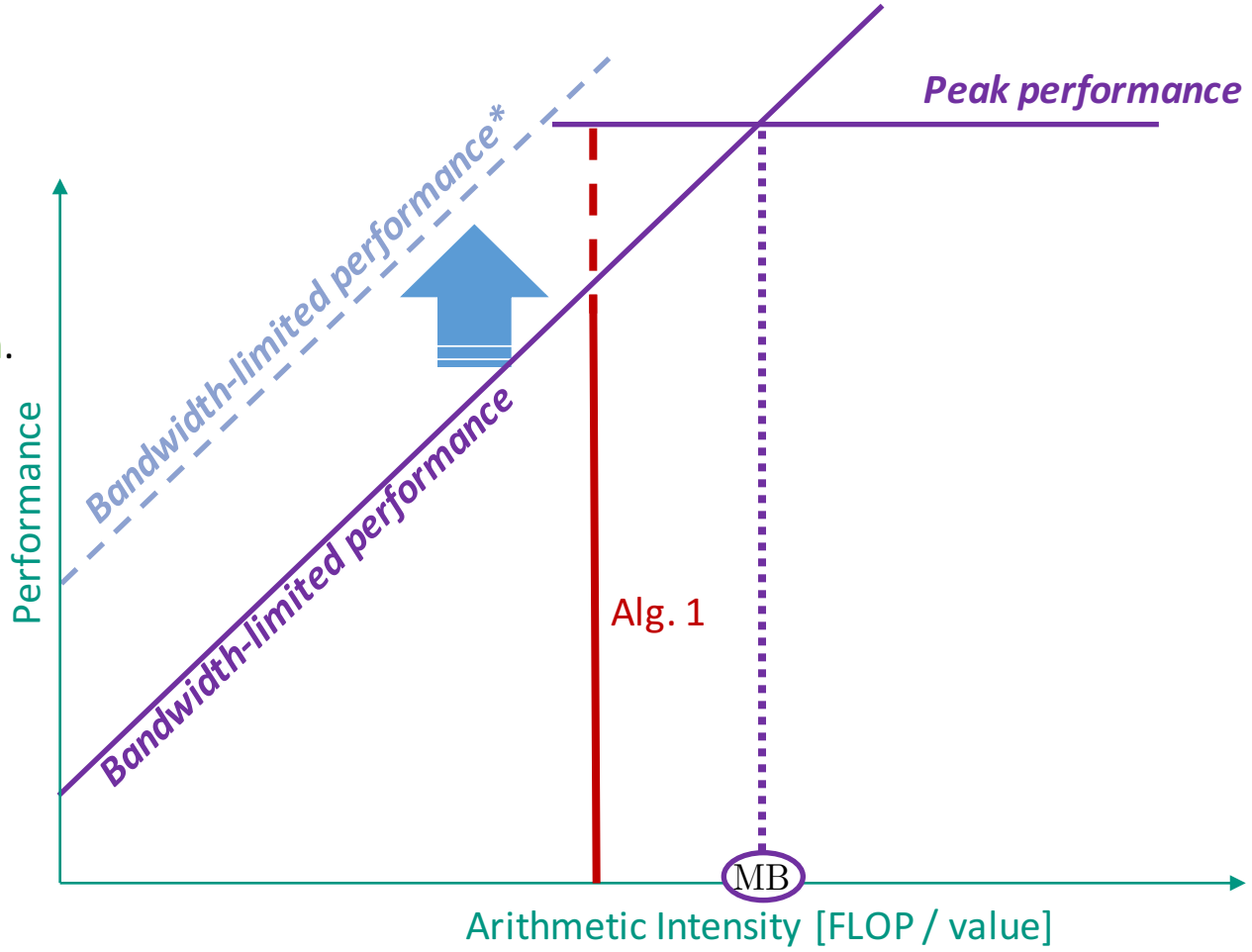


- Communication is the new bottleneck!
- The memory wall endangers continued performance growth.

Pushing the Roofline

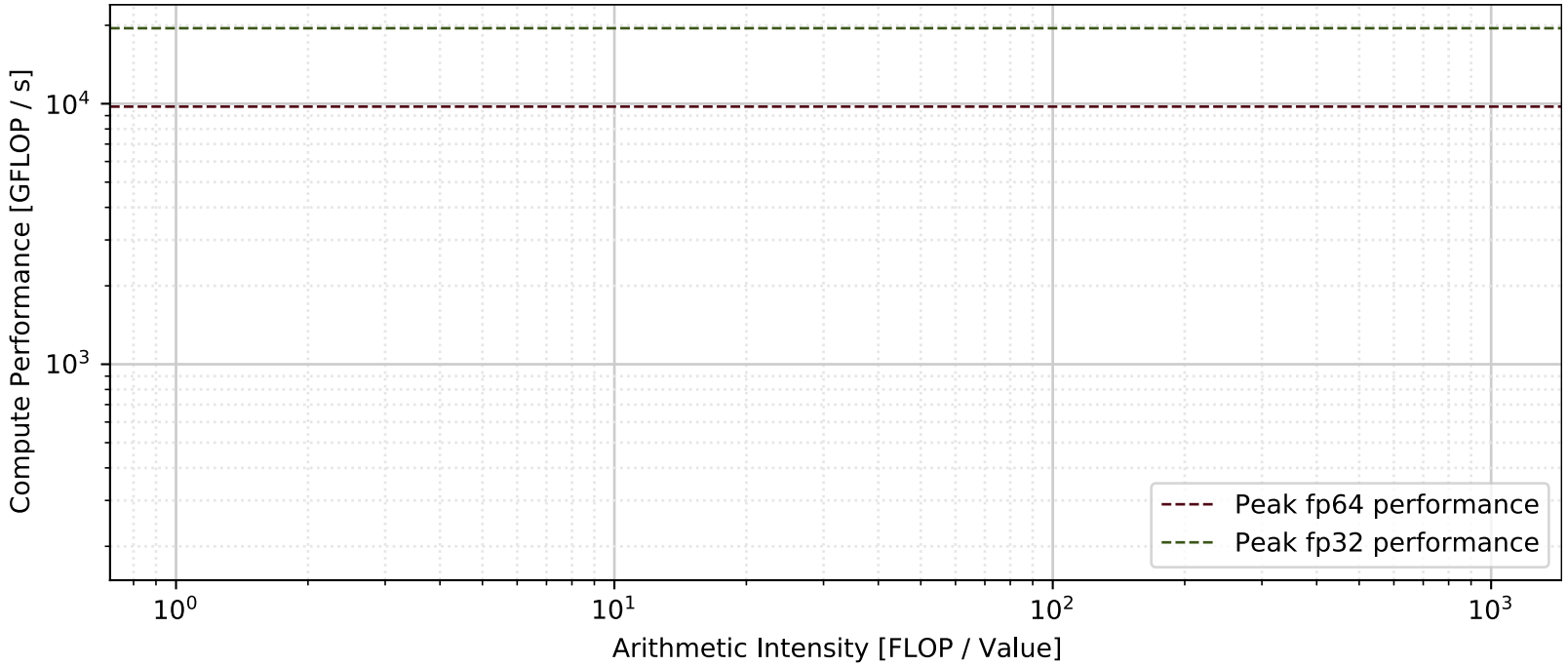
- Traditionally, we use a strong coupling between the precision formats used for arithmetic operations and storing data.
- The arithmetic operations are free, use high precision formats.
- **Data access** should be as cheap as possible, **reduced precision**.



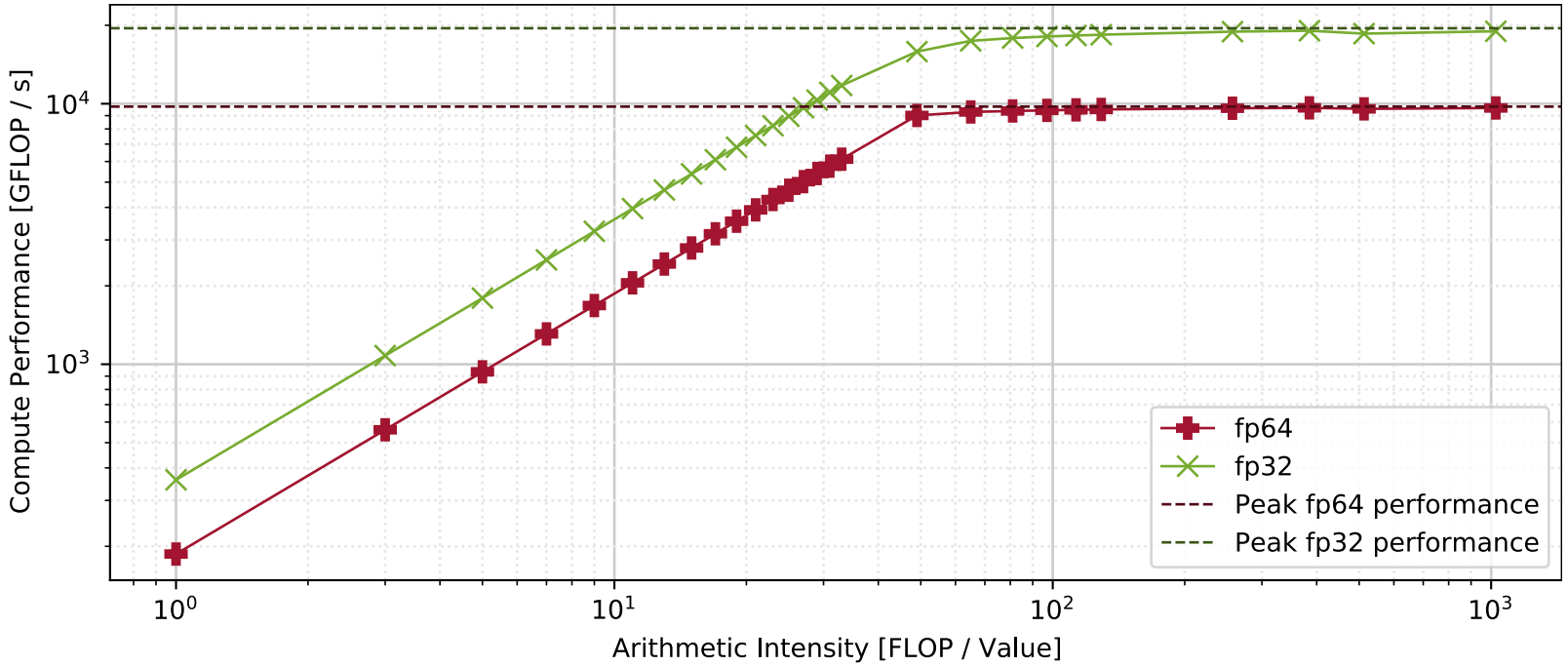


- Communication is the new bottleneck!
- The memory wall endangers continued performance growth.

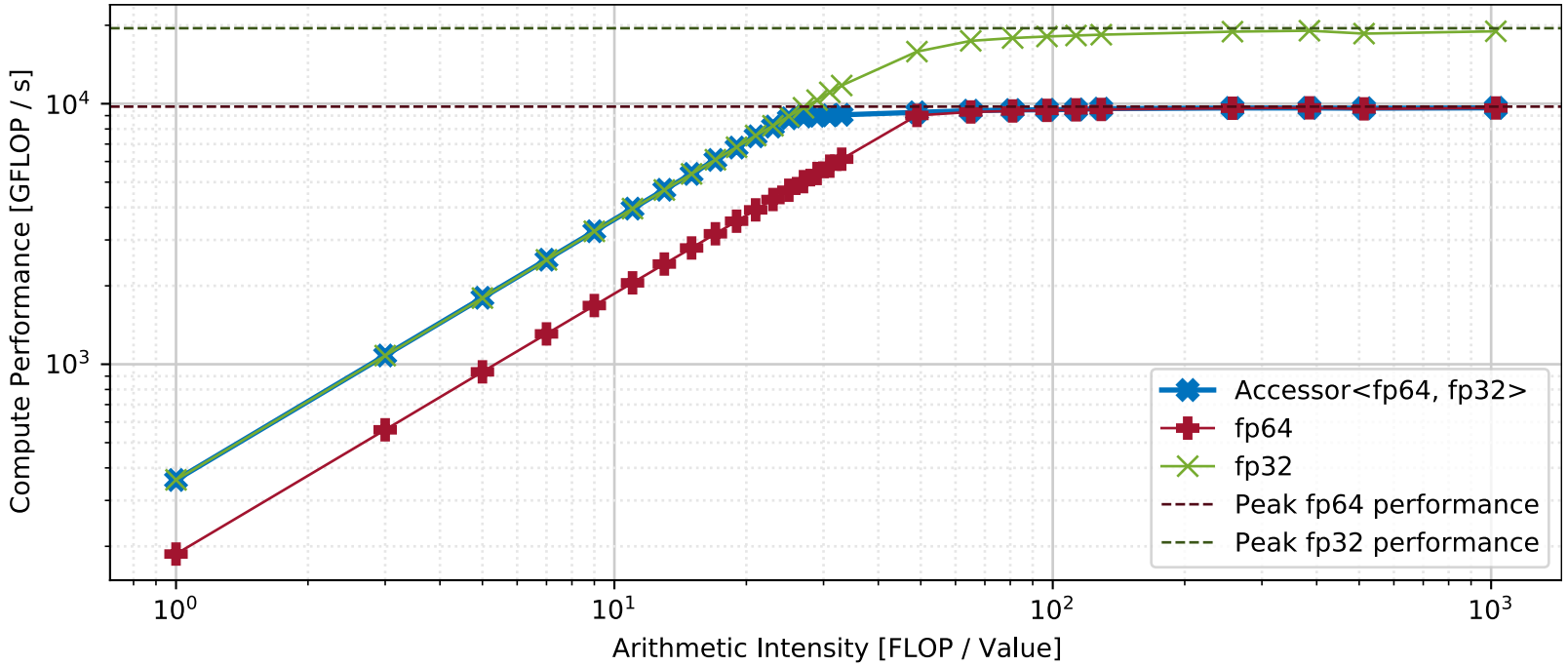
Accessor for NVIDIA A100 GPU



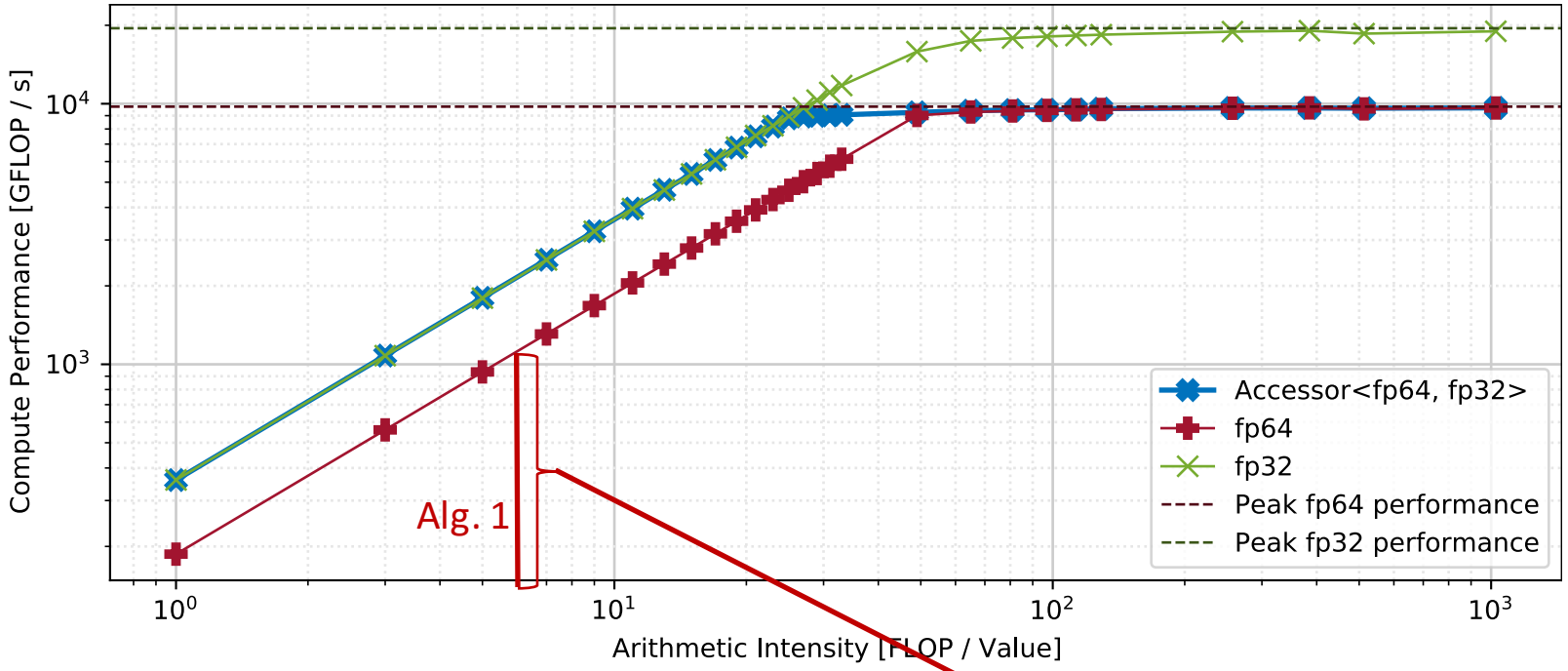
Accessor for NVIDIA A100 GPU



Accessor for NVIDIA A100 GPU

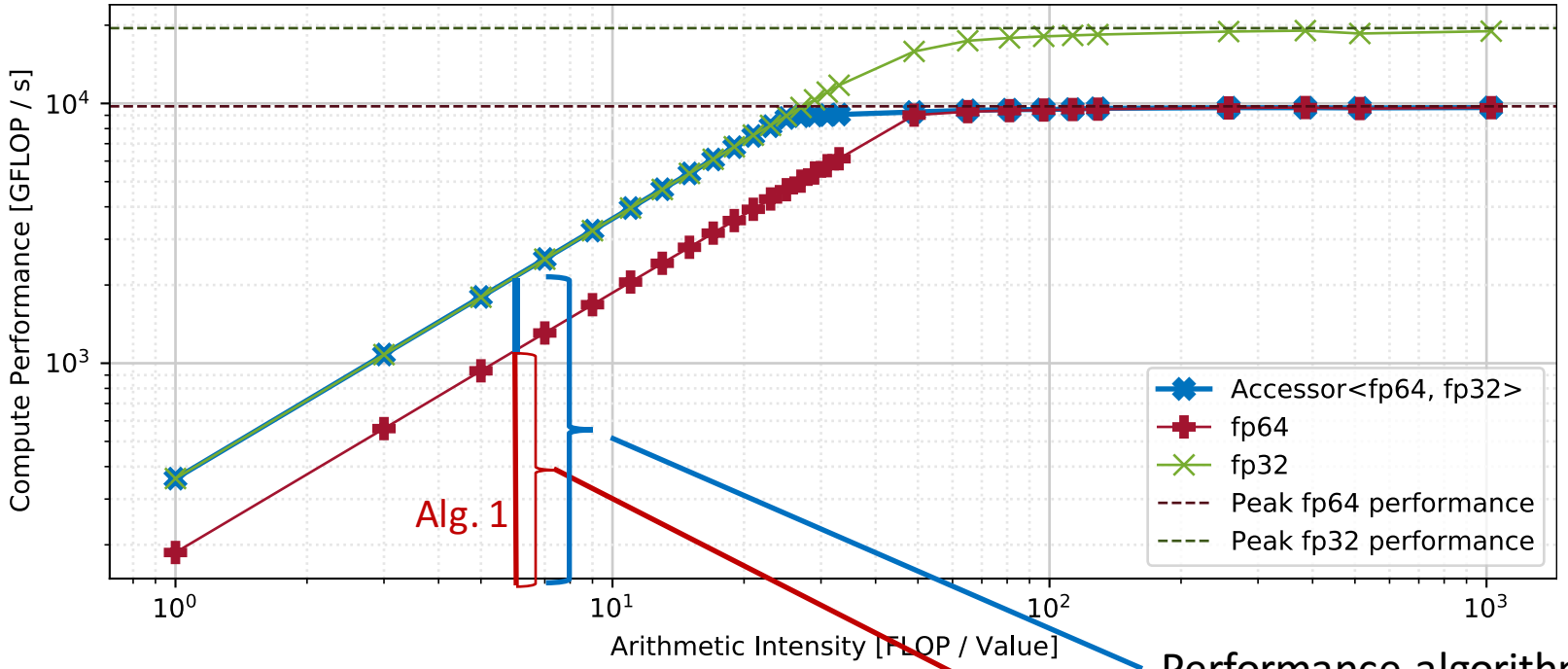


Accessor for NVIDIA A100 GPU



Performance algorithm achieves with standard memory access.

Accessor for NVIDIA A100 GPU



Performance algorithm achieves with memory accessor.

Performance algorithm achieves with standard memory access.

Memory accessor for memory-bound routines

Design

- Memory access in low precision (e.g. fp32);
- Computations in high precision (e.g. fp64);

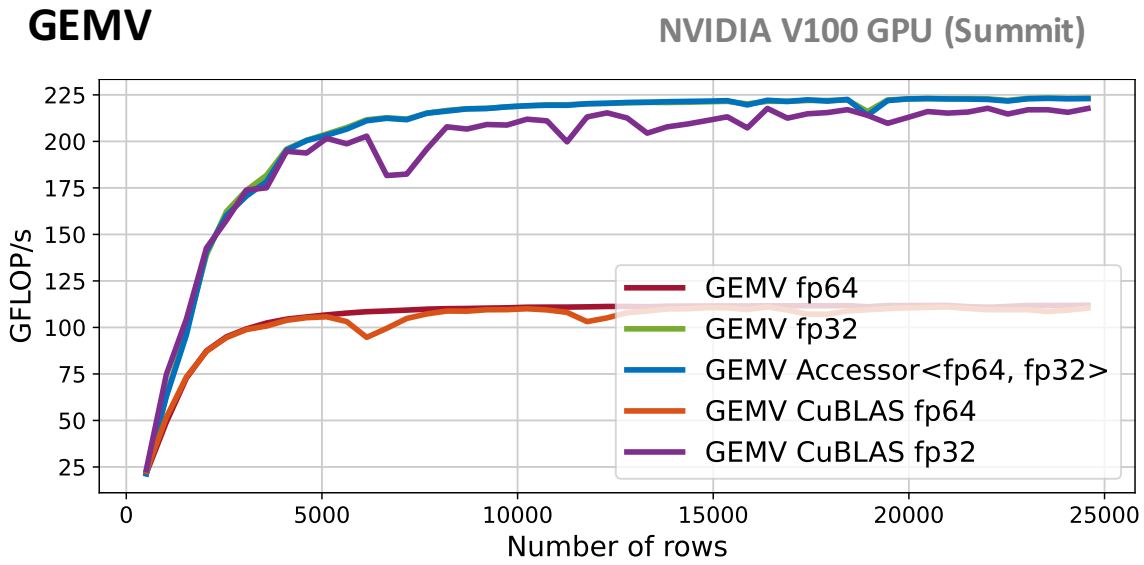
Characteristics

- Performance of low precision routine;
- Higher accuracy of low precision routine;

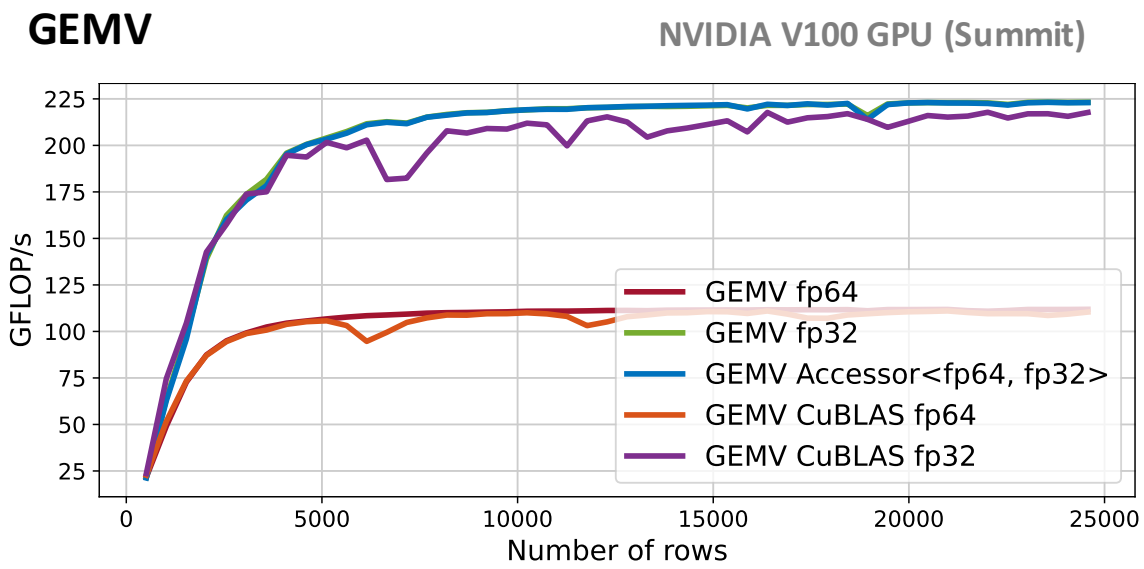
Usage

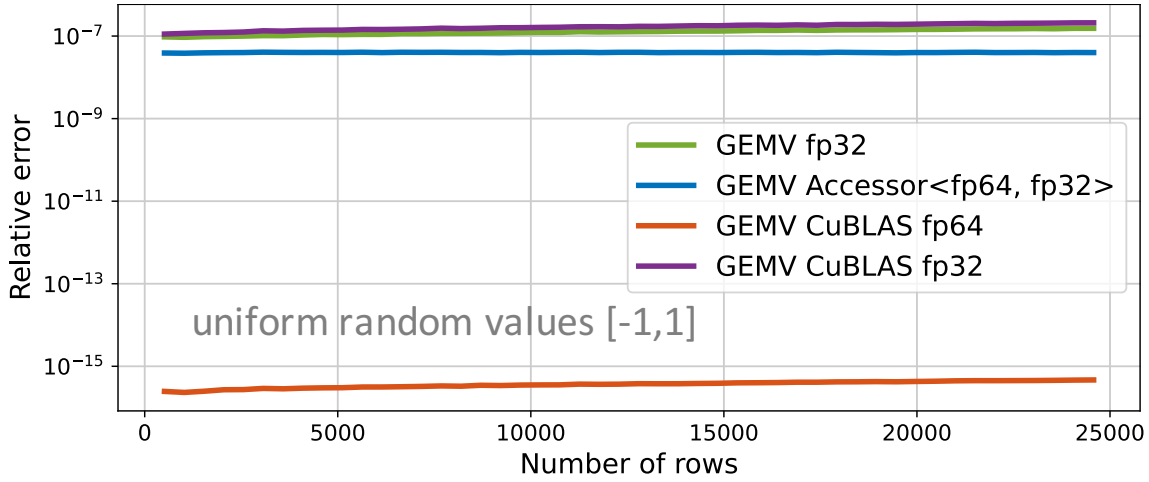
1. Accessor-BLAS can replace low precision BLAS to increase accuracy;
2. Accessor-BLAS can replace high precision BLAS if information loss is acceptable;
   (without having to deal with explicit mixed precision usage)

Accessor-BLAS: Replacing LP BLAS to improve accuracy



Accessor-BLAS: Replacing LP BLAS to improve accuracy

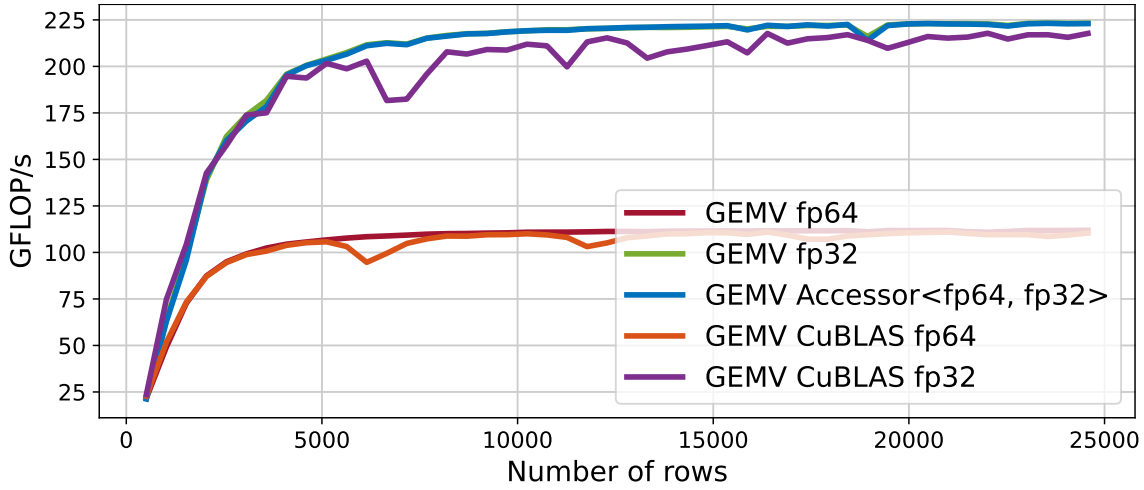


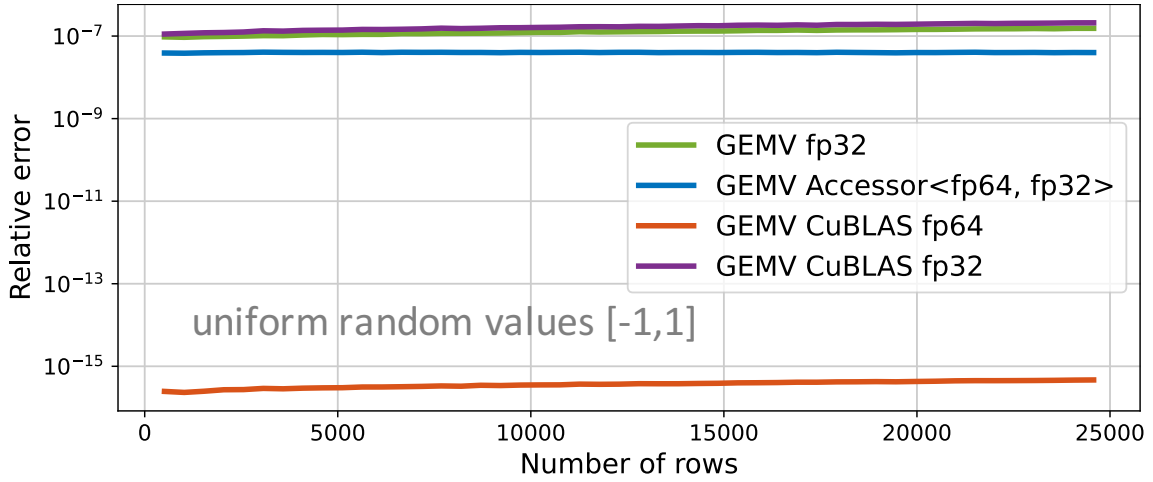


Accessor-BLAS: Replacing LP BLAS to improve accuracy

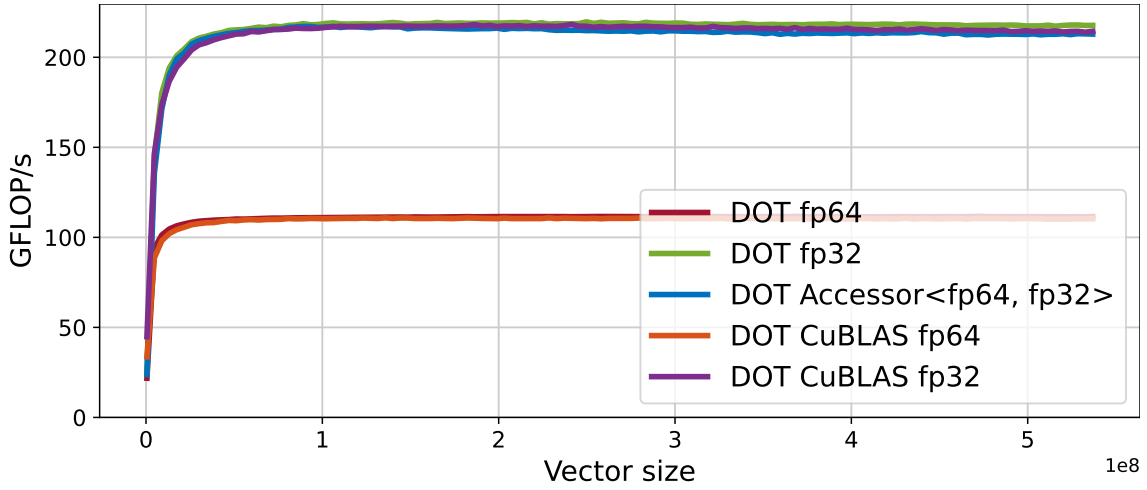
GEMV

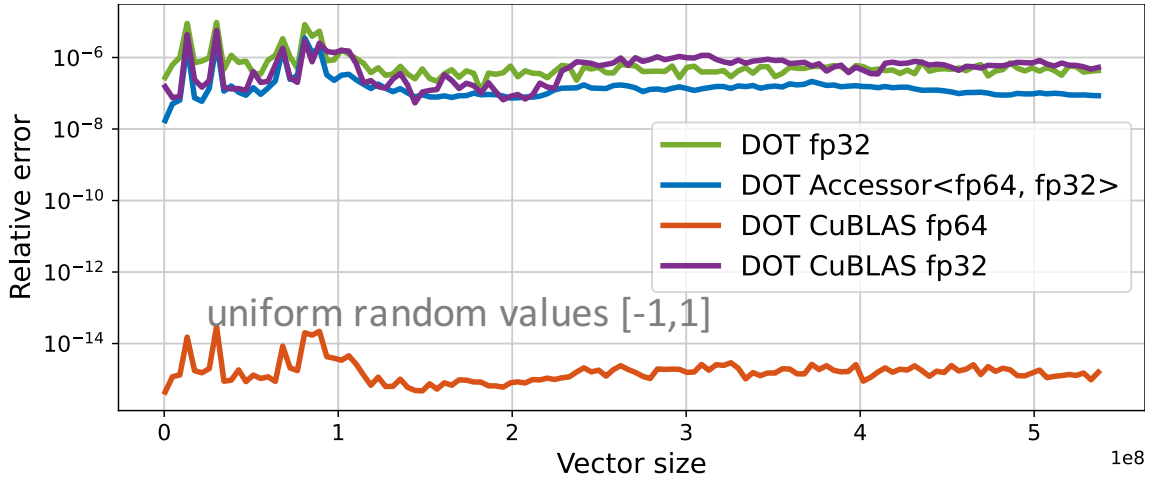
NVIDIA V100 GPU (Summit)





DOT

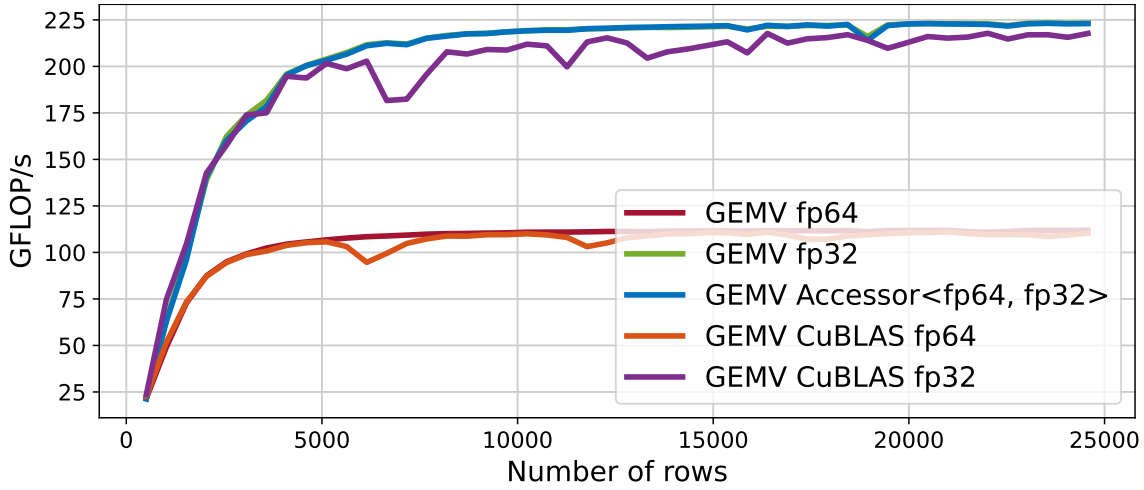


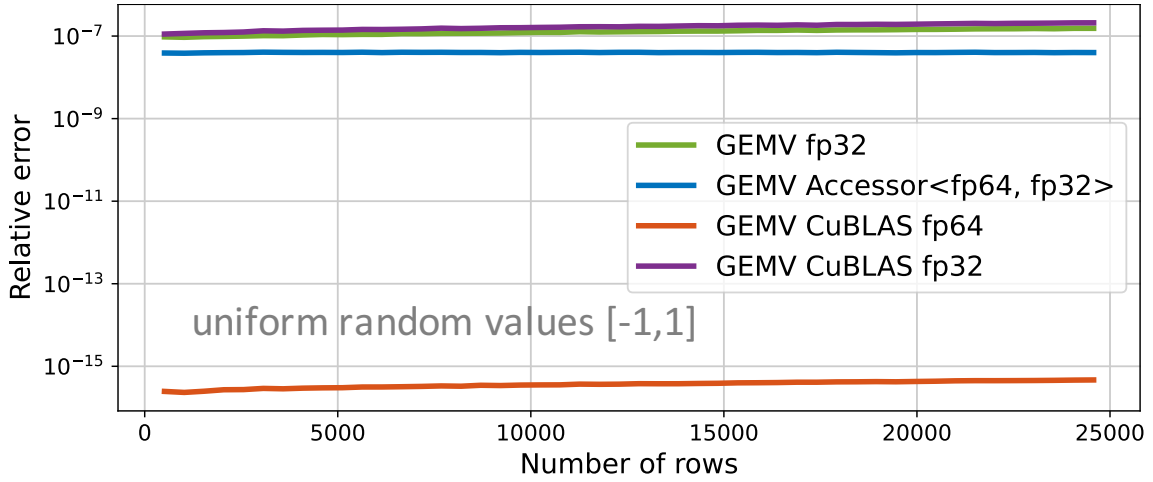


Accessor-BLAS: Replacing LP BLAS to improve accuracy

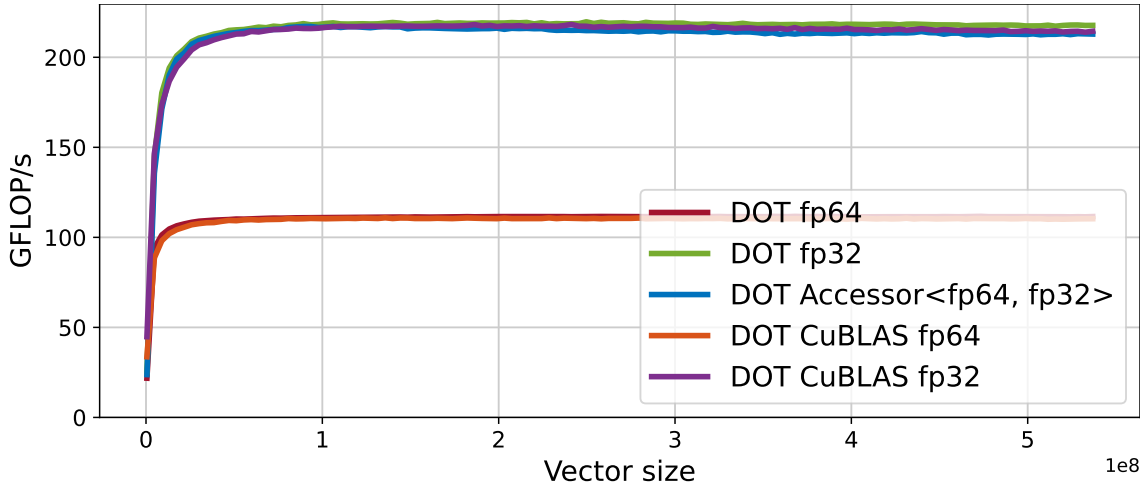
GEMV

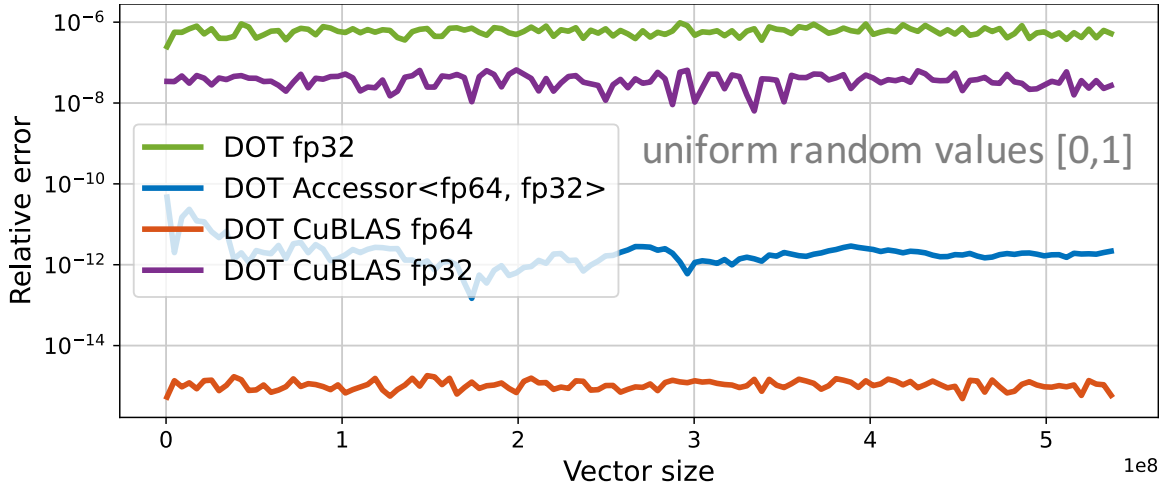
NVIDIA V100 GPU (Summit)





DOT





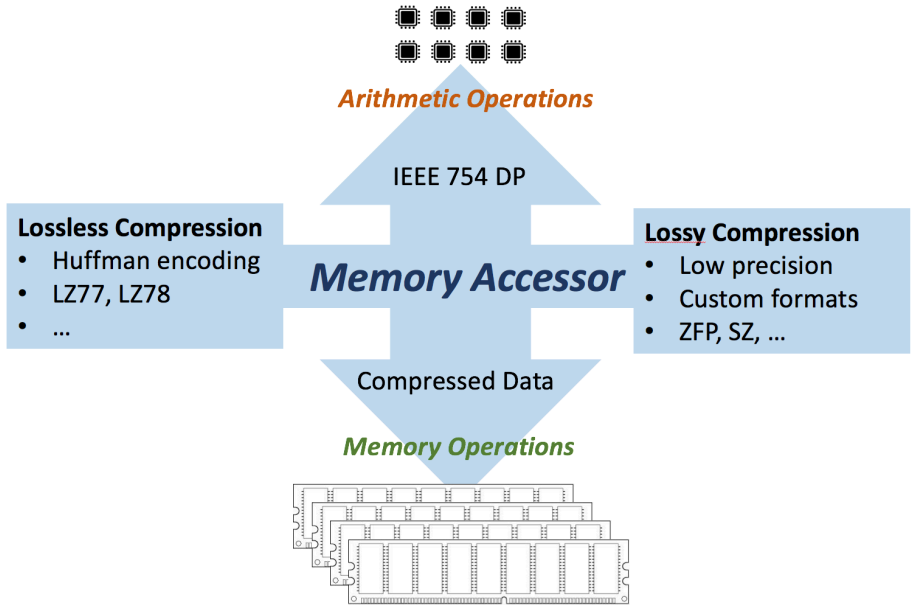
Rethinking Algorithms: Use the accessor to boost performance

Can we use the memory accessor to accelerate applications by compressing data without changing the application output?

- No, not in general.
- Yes, in some cases.
- We need to adapt the approach to the application & data.

  Two possibilities in the context of solving linear systems:

  - *"Self-healing" iterative methods;*
  - *Approximate linear operators;*



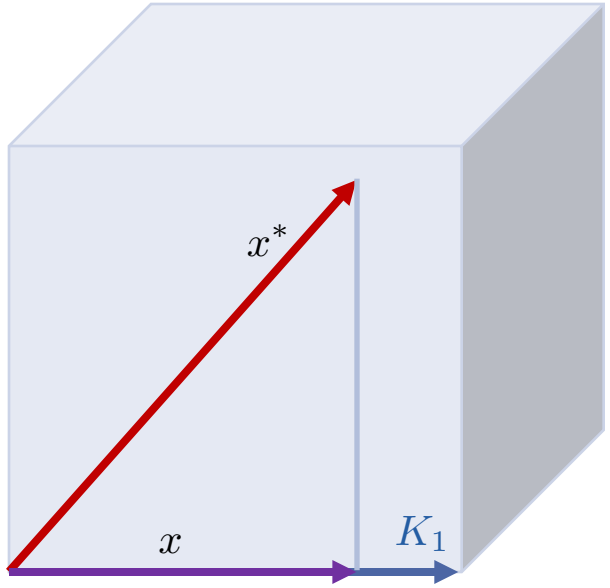
Rethinking Algorithms I: Self-Healing Iterative Methods

Rethinking Algorithms I: Self-Healing Iterative Methods

- **Krylov iterative solvers**
- Krylov methods aim at approximating the solution to a linear problem in a subspace.
- Over the iterations, a nested sequence of Krylov subspaces is generated, adding one basis vector in each iteration.
- Orthonormalization ensures a orthonormal basis is formed (Classical Gram-Schmidt, Modified Gram Schmidt…).

$$K_0 \subset K_1 \subset K_2 \subset \dots$$

$$K_i(A, r) = span\{b, Ab, A^2b, \dots A^{i-1}b\}$$



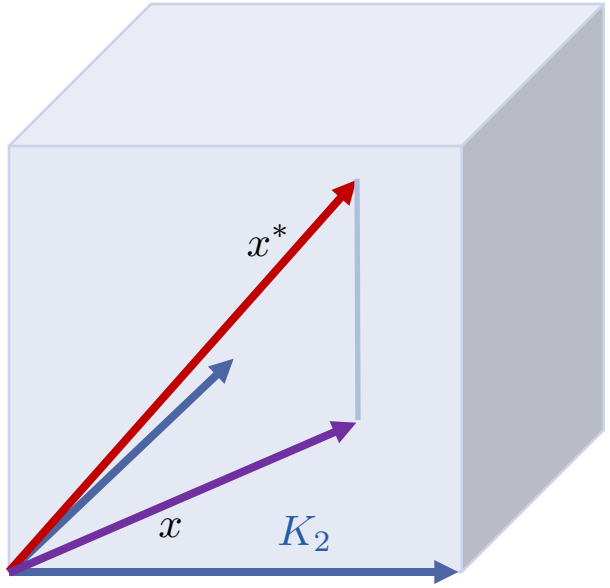
**GMRES in the modular precision ecosystem*

Rethinking Algorithms I: Self-Healing Iterative Methods

- **Krylov iterative solvers**
- Krylov methods aim at approximating the solution to a linear problem in a subspace.
- Over the iterations, a nested sequence of Krylov subspaces is generated, adding one basis vector in each iteration.
- Orthonormalization ensures a orthonormal basis is formed (Classical Gram-Schmidt, Modified Gram Schmidt…).

$$K_0 \subset K_1 \subset K_2 \subset \dots$$

$$K_i(A, r) = span\{b, Ab, A^2b, \dots A^{i-1}b\}$$



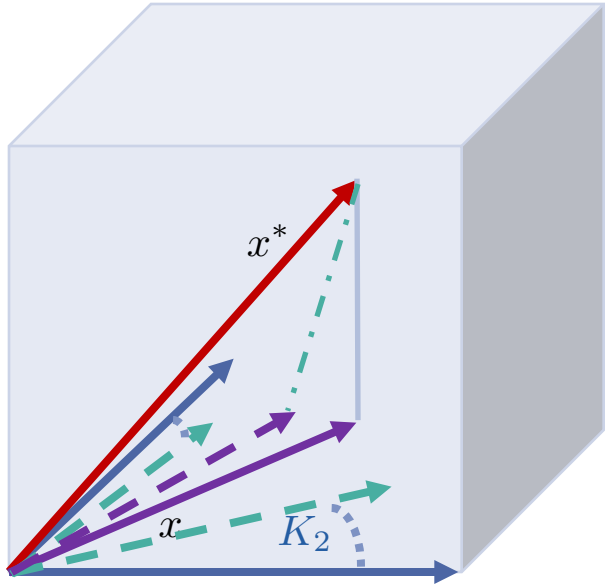
**GMRES in the modular precision ecosystem*

Rethinking Algorithms I: Self-Healing Iterative Methods

- **Krylov iterative solvers**
- Krylov methods aim at approximating the solution to a linear problem in a subspace.
- Over the iterations, a nested sequence of Krylov subspaces is generated, adding one basis vector in each iteration.
- Orthonormalization ensures a orthonormal basis is formed (Classical Gram-Schmidt, Modified Gram Schmidt…).

$$K_0 \subset K_1 \subset K_2 \subset \dots$$

$$K_i(A, r) = span\{b, Ab, A^2b, \dots A^{i-1}b\}$$



Compressed Basis (CB-) GMRES

- Use double precision in all arithmetic operations;
- Store Krylov basis vectors in lower precision;
  - Search directions are no longer DP-orthogonal;
  - Hessenberg system maps solution to "perturbed" Krylov subspace;
  - Additional iterations may be needed;
  - As long as the loss-of-orthogonality is moderate, we should see moderate convergence degradation;

Compressed Basis (CB-) GMRES

- **Krylov iterative solvers**
- Krylov methods aim at approximating the solution to a linear problem in a subspace.
- Over the iterations, a nested sequence of Krylov subspaces is generated, adding one basis vector in each iteration.
- Orthonormalization ensures a orthonormal basis is formed (Classical Gram-Schmidt, Modified Gram Schmidt…).

Ginkgo

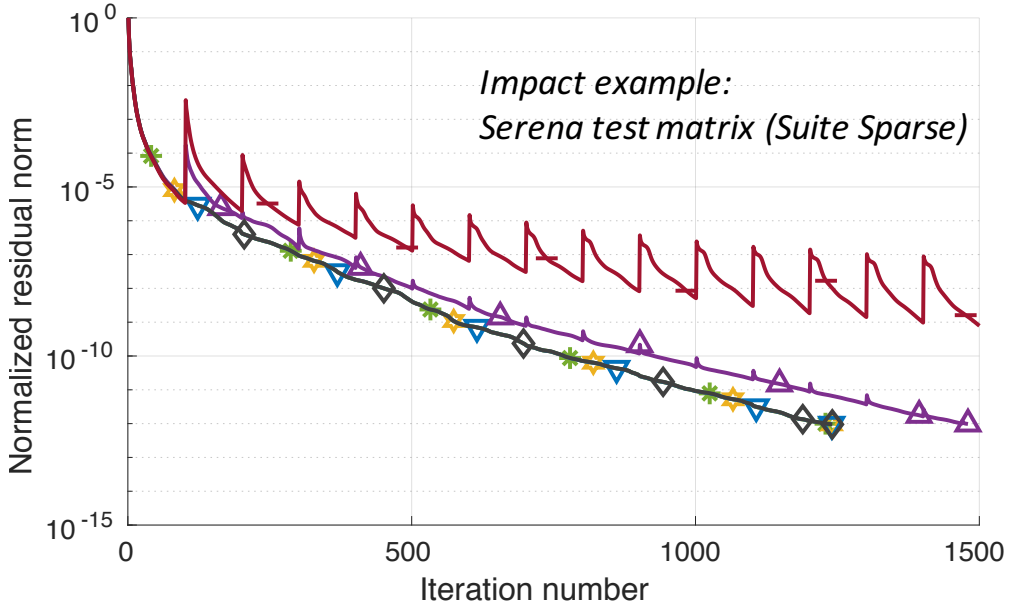
https://ginkgo-project.github.io/

MGS-GMRES<fp64,fp64>
GMRES<fp64,fp64>
GMRES<fp64,fp32>
GMRES<fp64,fp16>
GMRES<fp64,int32>
GMRES<fp64,int16>

arithmetic precision memory precision

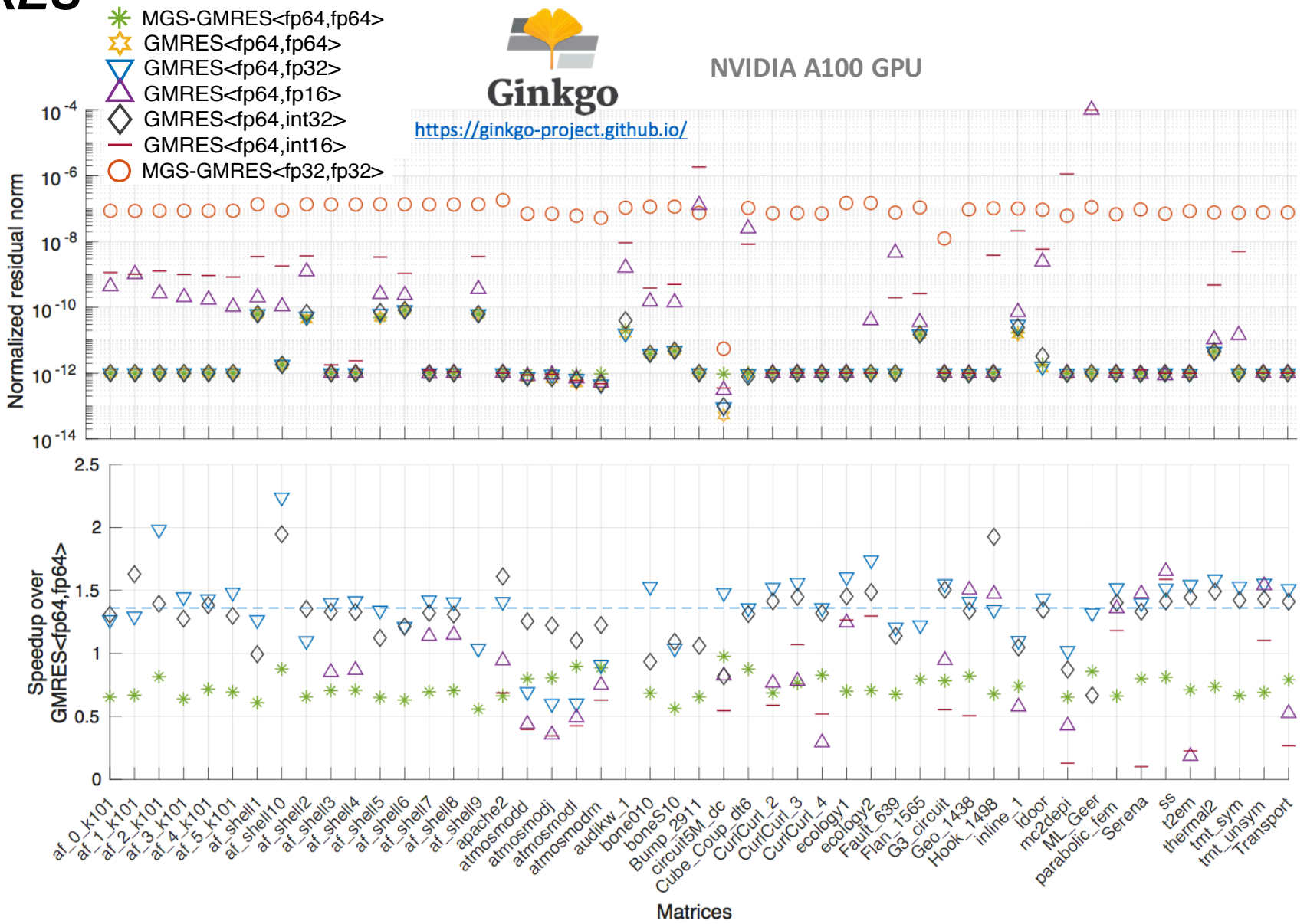
Compressed Basis (CB-) GMRES

- Use double precision in all arithmetic operations;
- Store Krylov basis vectors in lower precision;
  - Search directions are no longer DP-orthogonal;
  - Hessenberg system maps solution to "perturbed" Krylov subspace;
  - Additional iterations may be needed;
  - As long as the loss-of-orthogonality is moderate, we should see moderate convergence degradation;



Compressed Basis GMRES

- CB-GMRES using 32-bit storage preserves DP accuracy (SP-GMRES does not)
- Speedups problem-dependent
- Speedup ∅1.4x (for restart 100)
- 16-bit storage mostly inefficient



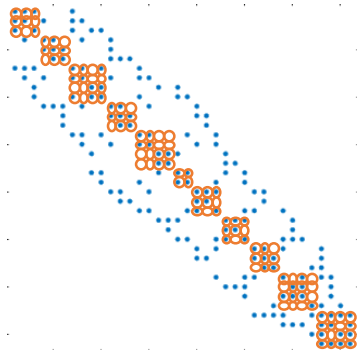
Rethinking Algorithms II: Approximate Linear Operators

Rethinking Algorithms II: Approximate Linear Operators

- **Preconditioning iterative solvers.**
  - Idea: Approximate inverse of system matrix to make the system "easier to solve": $P^{-1} \approx A^{-1}$

    and solve $Ax = b \quad \Leftrightarrow \quad P^{-1}Ax = P^{-1}b \quad \Leftrightarrow \quad \tilde{A}x = \tilde{b}\,.$

- **Block-Jacobi preconditioner** is based on **block-diagonal scaling**: $P = diag_B(A)$
  - Each block corresponds to one (small) linear system.
    - *Larger* blocks typically improve convergence.
    - *Larger* blocks make block-Jacobi more expensive.

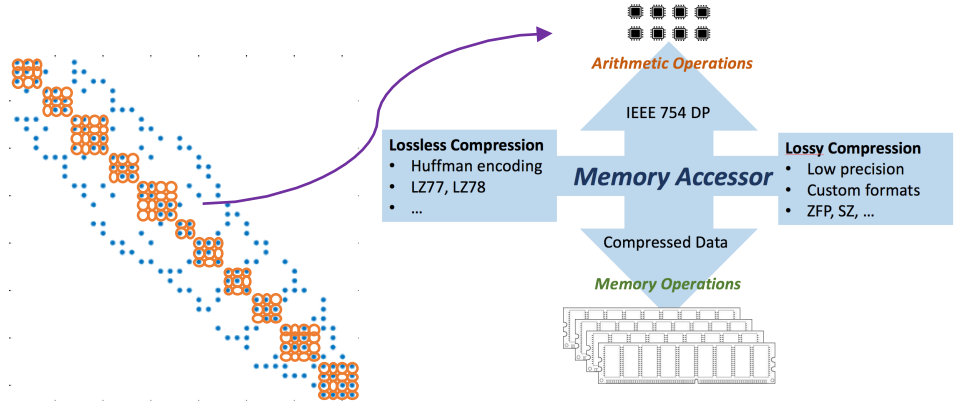


Rethinking Algorithms II: Approximate Linear Operators

- **Preconditioning iterative solvers.**
  - Idea: Approximate inverse of system matrix to make the system "easier to solve": $P^{-1} \approx A^{-1}$

    and solve $Ax = b \quad \Leftrightarrow \quad P^{-1}Ax = P^{-1}b \quad \Leftrightarrow \quad \tilde{A}x = \tilde{b}\,.$

- **Block-Jacobi preconditioner** is based on **block-diagonal scaling**: $P = diag_B(A)$
  - Each block corresponds to one (small) linear system.
    - *Larger* blocks typically improve convergence.
    - *Larger* blocks make block-Jacobi more expensive.

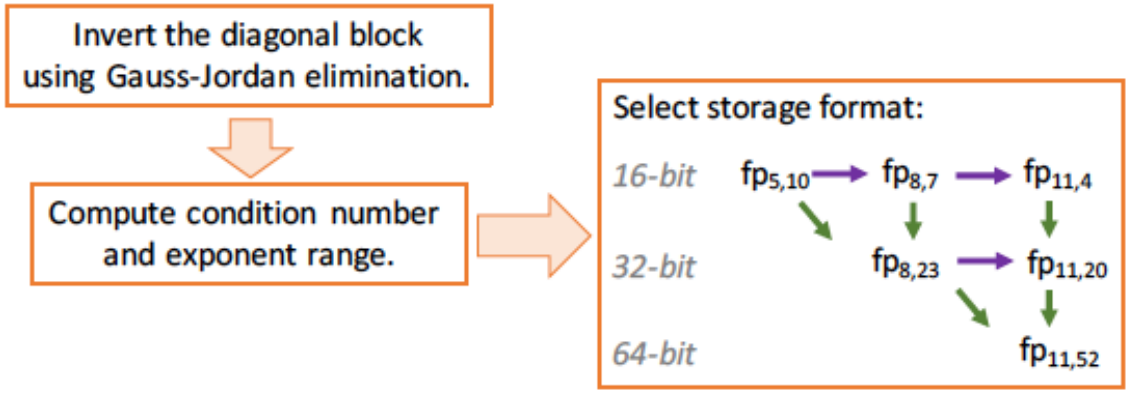


- *Why should we store the preconditioner matrix P^{-1} in full (high) precision?*

- Use the accessor to store the inverted diagonal blocks in lower precision.
  - *Be careful to preserve the regularity of each inverted diagonal block!*

Mixed Precision Preconditioning

- Choose how much accuracy of the preconditioner should be preserved in the selection of the storage format.
- All computations use double precision, but store blocks in lower precision.



\+ **Regularity preserved;**

\+ Flexibility in the accuracy;

\+ ”Not a low precision preconditioner”

  \+ Preconditioner is a constant operator;

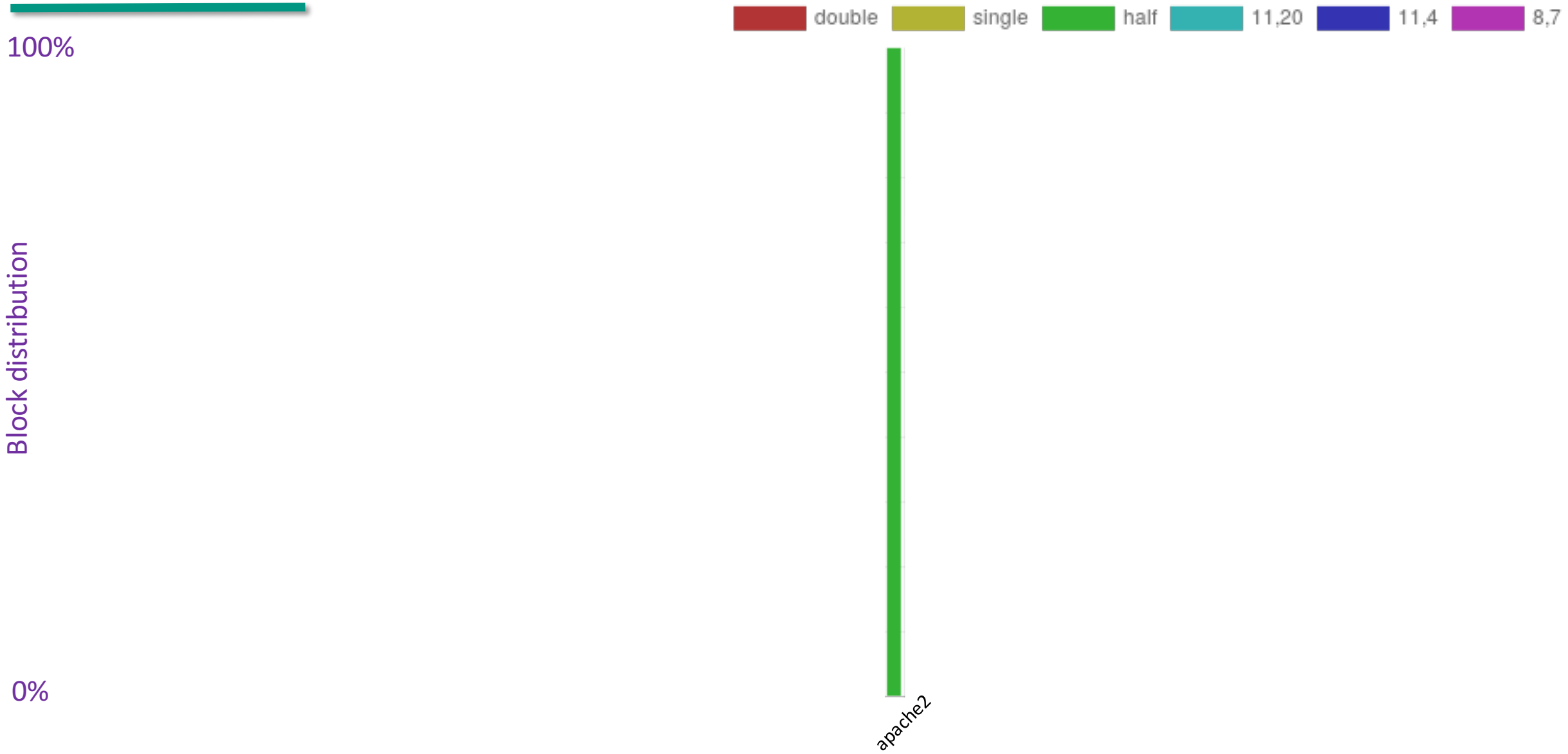
  \+ No flexible Krylov solver needed ;

\- **Overhead** of the **precision detection** (condition number calculation);

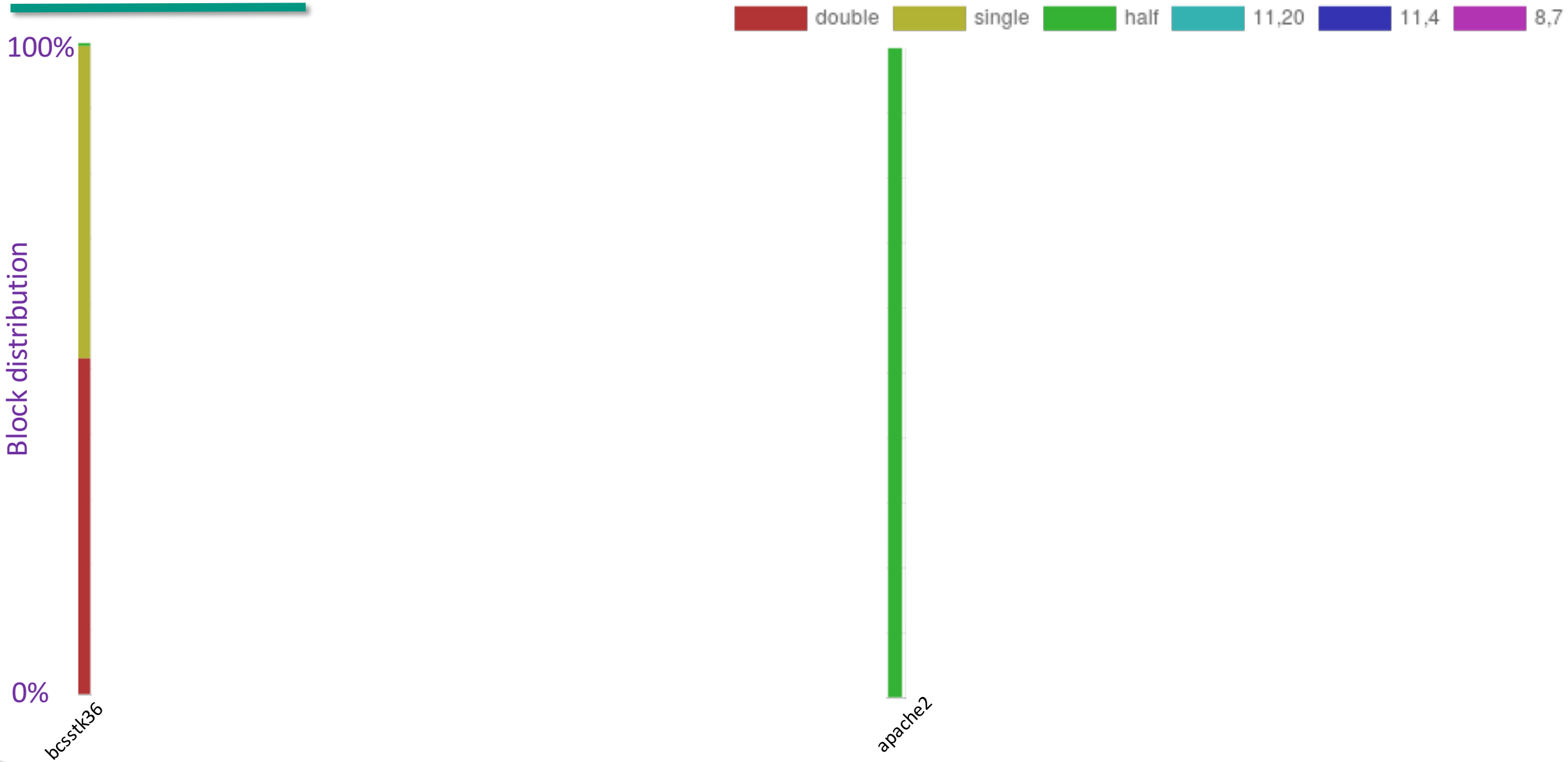
\- **Overhead** from storing **precision information** (need to additionally store/retrieve flag);

\- Speedups / preconditioner quality **problem-dependent**;

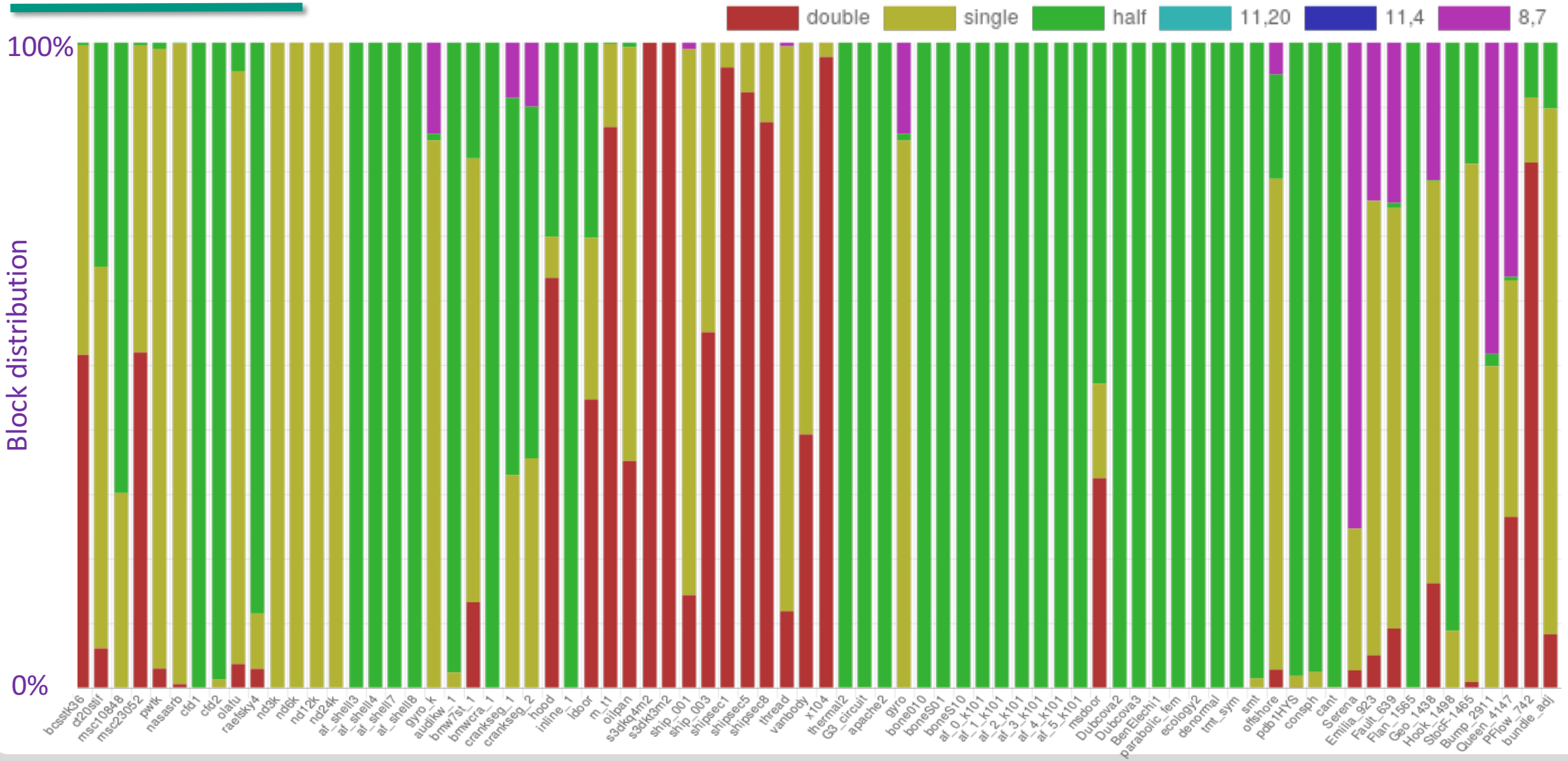
Mixed Precision Preconditioning



Mixed Precision Preconditioning



Mixed Precision Preconditioning



Mixed Precision Preconditioning

Linear System Ax=b with cond(A) ≈ 10^7 *(apache2 from SuiteSparse)* NVIDIA A100 GPU

Double Precision CG + Double Precision Preconditioner

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.97985e-06
CG iteration count:     4797
CG execution time [ms]: 2971.18
```

Accuracy improvement ~10^{-9}

relative residual accuracy ≈ **(unit round-off)** * **(linear system's condition number)**

N. Higham: Accuracy and stability of numerical algorithms. SIAM, 2002.

Experiments based on the Ginkgo library https://ginkgo-project.github.io/
ginkgo/examples/adaptiveprecision-blockjacobi/adaptiveprecision-blockjacobi.cpp

Ginkgo

Mixed Precision Preconditioning

Linear System Ax=b with cond(A) ≈ 10^7 *(d*

```
…
    - ValueType = double; //u=1e-16
    + ValueType = float;  //u=1e-7
…
```

Double Precision CG + Double Precision P

```
Initial residual norm sqrt(r^T r)
%%MatrixMarket matrix array real
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.97985e-06
CG iteration count:     4797
CG execution time [ms]: 2971.18
```

Accuracy improvement ~10^{-9}

Experiments based on the Ginkgo library https://ginkgo-project.github.io/
ginkgo/examples/adaptiveprecision-blockjacobi/adaptiveprecision-blockjacobi.cpp

Ginkgo

Mixed Precision Preconditioning

Linear System Ax=b with cond(A) ≈ 10^7 *(apache2 from SuiteSparse)* **NVIDIA A100 GPU**

Double Precision CG + Double Precision Preconditioner

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.97985e-06
CG iteration count:     4797
CG execution time [ms]: 2971.18
```

Accuracy improvement ~10^{-9}

Single Precision CG + Single Precision Preconditioner

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1588.77
CG iteration count:     8887
CG execution time [ms]: 2972.46
```

No improvement

relative residual accuracy ≈ **(unit round-off)** * **(linear system's condition number)**

N. Higham: Accuracy and stability of numerical algorithms. SIAM, 2002.

Experiments based on the Ginkgo library https://ginkgo-project.github.io/

ginkgo/examples/adaptiveprecision-blockjacobi/adaptiveprecision-blockjacobi.cpp

Ginkgo

Mixed Precision Preconditioning

Linear System Ax=b with cond(A) ≈ 10^7 *(apache2 from SuiteSparse)* NVIDIA A100 GPU

Double Precision CG + Double Precision Preconditioner

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.97985e-06
CG iteration count:     4797
CG execution time [ms]: 2971.18
```

Accuracy improvement ~10^{-9}

Double Precision CG + **Mixed Precision Preconditioner**

Experiments based on the Ginkgo library https://ginkgo-project.github.io/
ginkgo/examples/adaptiveprecision-blockjacobi/adaptiveprecision-blockjacobi.cpp

Ginkgo

Mixed Precision Preconditioning

Linear System Ax=b with cond(A) ≈ 10^7 *(apache2 from SuiteSparse)* NVIDIA A100 GPU

Double Precision CG + Double Precision Preconditioner

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.97985e-06
CG iteration count:     4797
CG execution time [ms]: 2971.18
```

Accuracy improvement ~10^{-9}

Double Precision CG + **Mixed Precision Preconditioner**

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.98574e-06
CG iteration count:     4794
CG execution time [ms]: 2568.1
```

Accuracy improvement ~10^{-9}

Experiments based on the Ginkgo library https://ginkgo-project.github.io/
ginkgo/examples/adaptiveprecision-blockjacobi/adaptiveprecision-blockjacobi.cpp

Ginkgo

Mixed Precision Preconditioning

Linear System Ax=b with cond(A) ≈ 10^7 *(apache2 from SuiteSparse)* NVIDIA A100 GPU

Double Precision CG + Double Precision Preconditioner

```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.97985e-06
CG iteration count:      4797
CG execution time [ms]: 2971.18
```

Accuracy improvement ~10^{-9}

Double Precision CG + **Mixed Precision Preconditioner**

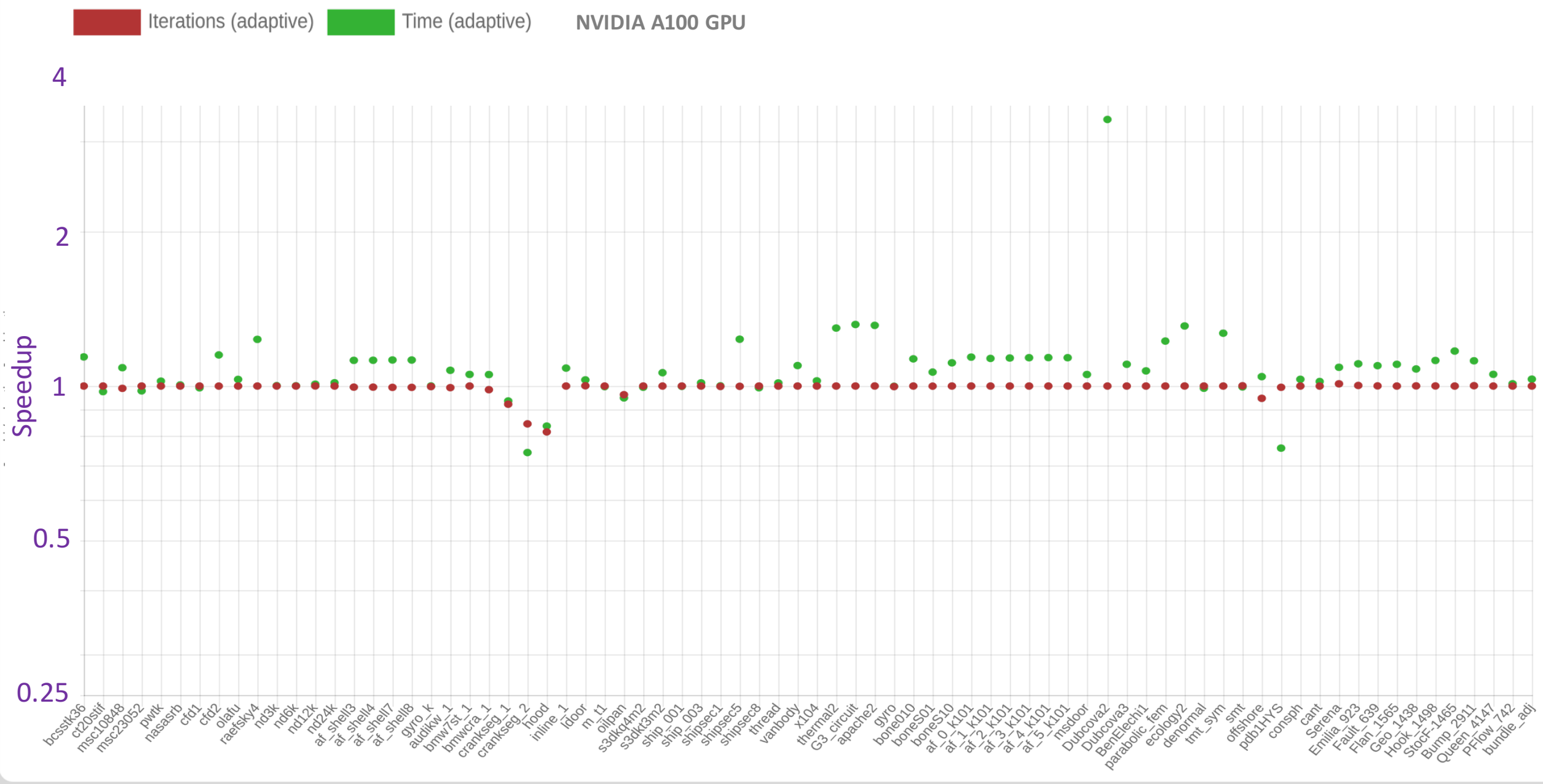
```
Initial residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
1390.67
Final residual norm sqrt(r^T r):
%%MatrixMarket matrix array real general
1 1
3.98574e-06
CG iteration count:      4794
CG execution time [ms]: 2568.1
```

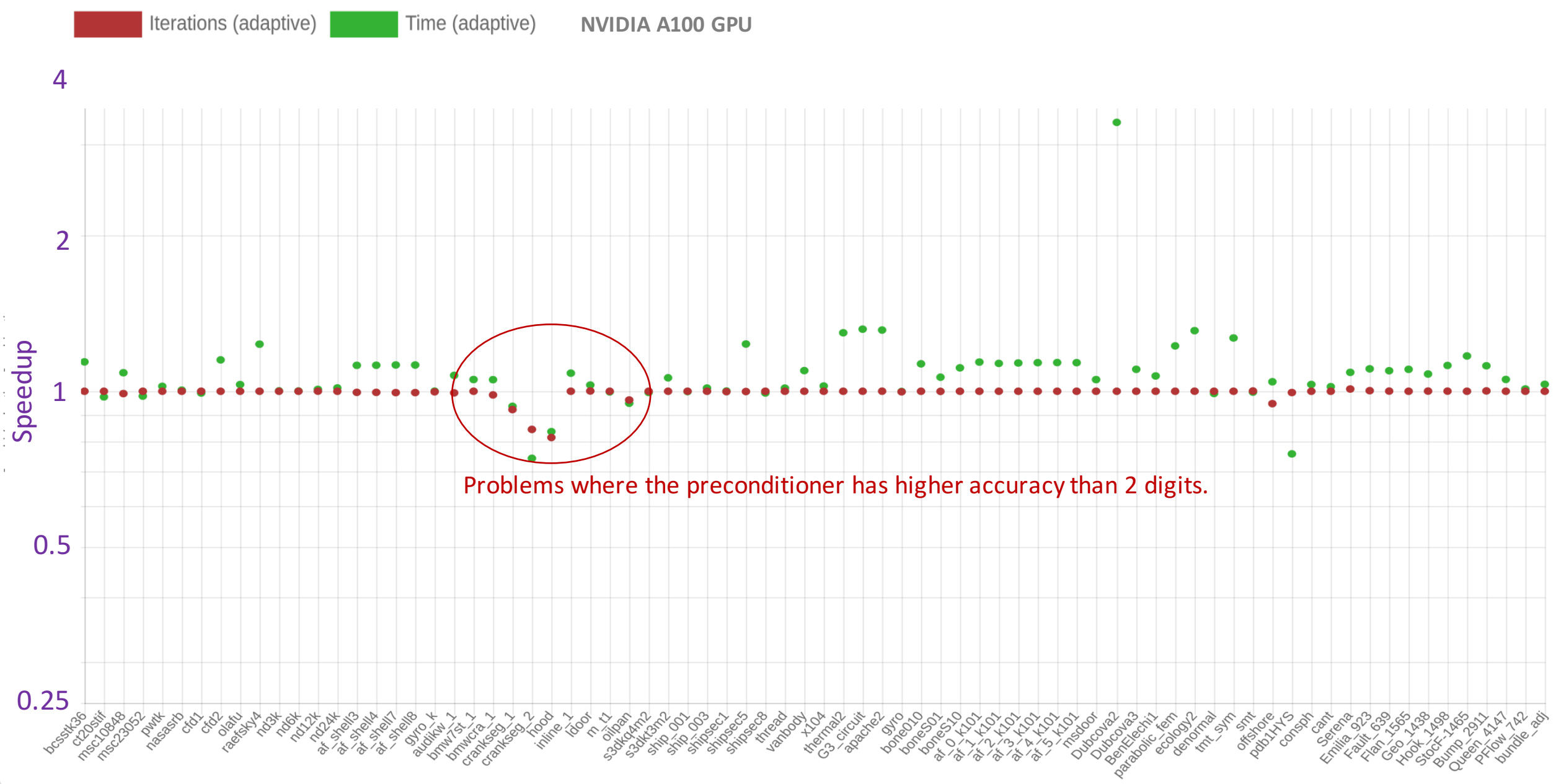
Accuracy improvement ~10^{-9}

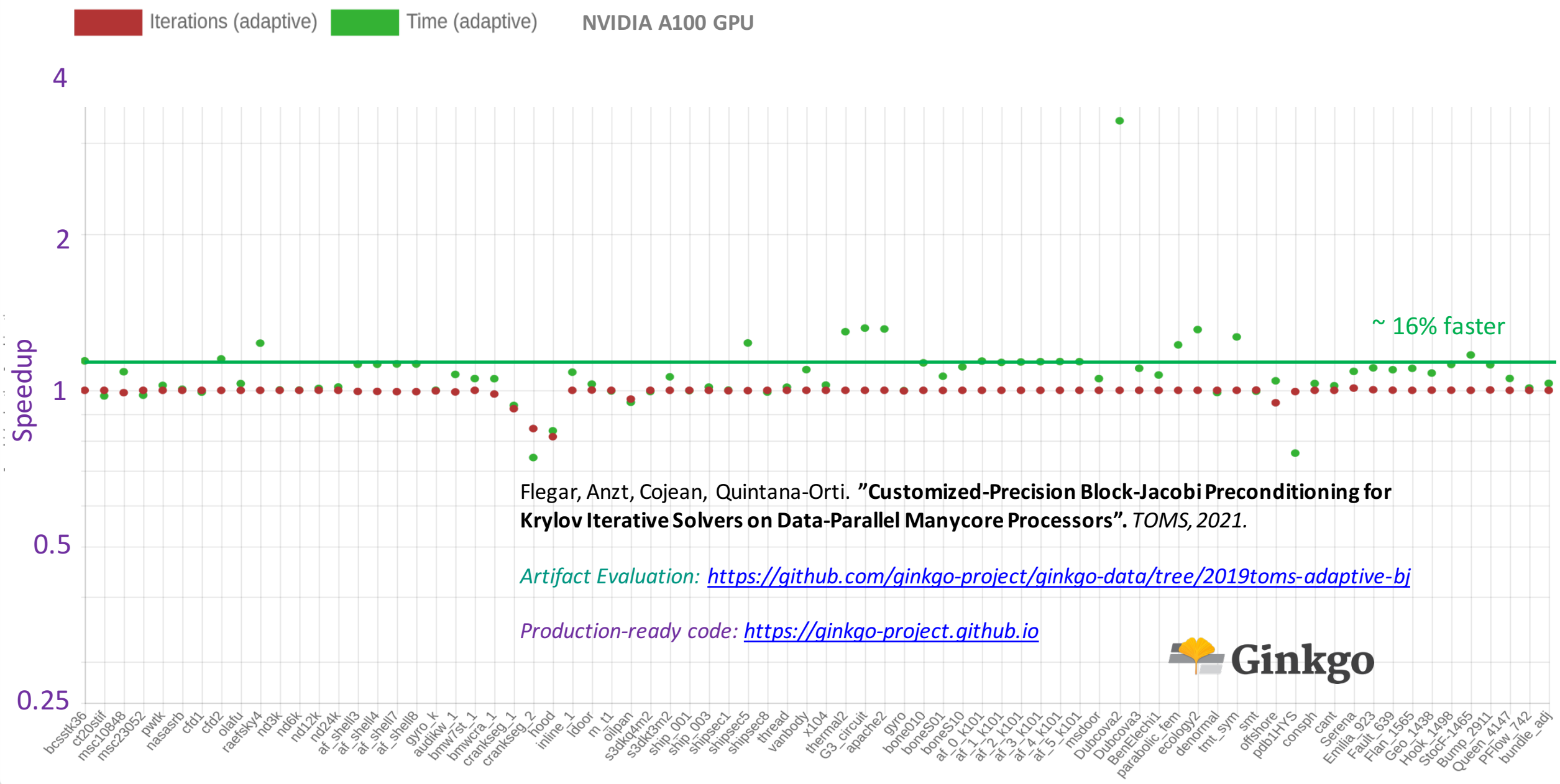
16% runtime improvement

Experiments based on the Ginkgo library https://ginkgo-project.github.io/
ginkgo/examples/adaptiveprecision-blockjacobi/adaptiveprecision-blockjacobi.cpp

Ginkgo


Iterations (adaptive)
Time (adaptive)
NVIDIA A100 GPU
Speedup
4
2
1
0.5
0.25
Problem
bcsstk36
ct20stif
msc10848
msc23052
pwtk
nasasrb
cfd1
cfd2
olafu
raefsky4
nd3k
nd6k
nd12k
nd24k
af_shell3
af_shell4
af_shell7
af_shell8
gyro_k
audikw_1
bmw7st_1
bmwcra_1
crankseg_1
crankseg_2
hood
inline_1
ldoor
m_t1
oilpan
s3dkq4m2
s3dkt3m2
ship_001
ship_003
shipsec1
shipsec5
shipsec8
thread
vanbody
x104
thermal2
G3_circuit
apache2
gyro
boneS01
boneS10
af_0_k101
af_1_k101
af_2_k101
af_3_k101
af_4_k101
af_5_k101
msdoor
Dubcova2
Dubcova3
BenElechi1
parabolic_fem
ecology2
denormal
tmt_sym
smt
offshore
pdb1HYS
consph
cant
Serena
Emilia_923
Fault_639
Flan_1565
Geo_1438
Hook_1498
StocF-1465
Bump_2911
Queen_4147
PFlow_742
bundle_adj


Iterations (adaptive)
Time (adaptive)
NVIDIA A100 GPU
Speedup
4
2
1
0.5
0.25
Problems where the preconditioner has higher accuracy than 2 digits.
Problem
bcsstk36
ct20stif
msc10848
msc23052
pwtk
nasasrb
cfd1
cfd2
olafu
raefsky4
nd3k
nd6k
nd12k
nd24k
af_shell3
af_shell4
af_shell7
af_shell8
gyro_k
audikw_1
bmw7st_1
bmwcra_1
crankseg_1
crankseg_2
hood
inline_1
ldoor
m_t1
oilpan
s3dkq4m2
s3dkt3m2
ship_001
ship_003
shipsec1
shipsec5
shipsec8
thread
vanbody
x104
thermal2
G3_circuit
apache2
gyro
bone010
boneS01
boneS10
af_0_k101
af_1_k101
af_2_k101
af_3_k101
af_4_k101
af_5_k101
msdoor
Dubcova2
Dubcova3
BenElechi1
parabolic_fem
ecology2
denormal
tmt_sym
smt
offshore
pdb1HYS
consph
cant
Serena
Emilia_923
Fault_639
Flan_1565
Geo_1438
Hook_1498
StocF-1465
Bump_2911
Queen_4147
PFlow_742
bundle_adj



Flegar, Anzt, Cojean, Quintana-Orti. **"Customized-Precision Block-Jacobi Preconditioning for Krylov Iterative Solvers on Data-Parallel Manycore Processors".** *TOMS, 2021.*

Artifact Evaluation: https://github.com/ginkgo-project/ginkgo-data/tree/2019toms-adaptive-bj

Production-ready code: https://ginkgo-project.github.io

Ginkgo

Mixed Precision Sparse Approximate Inverse

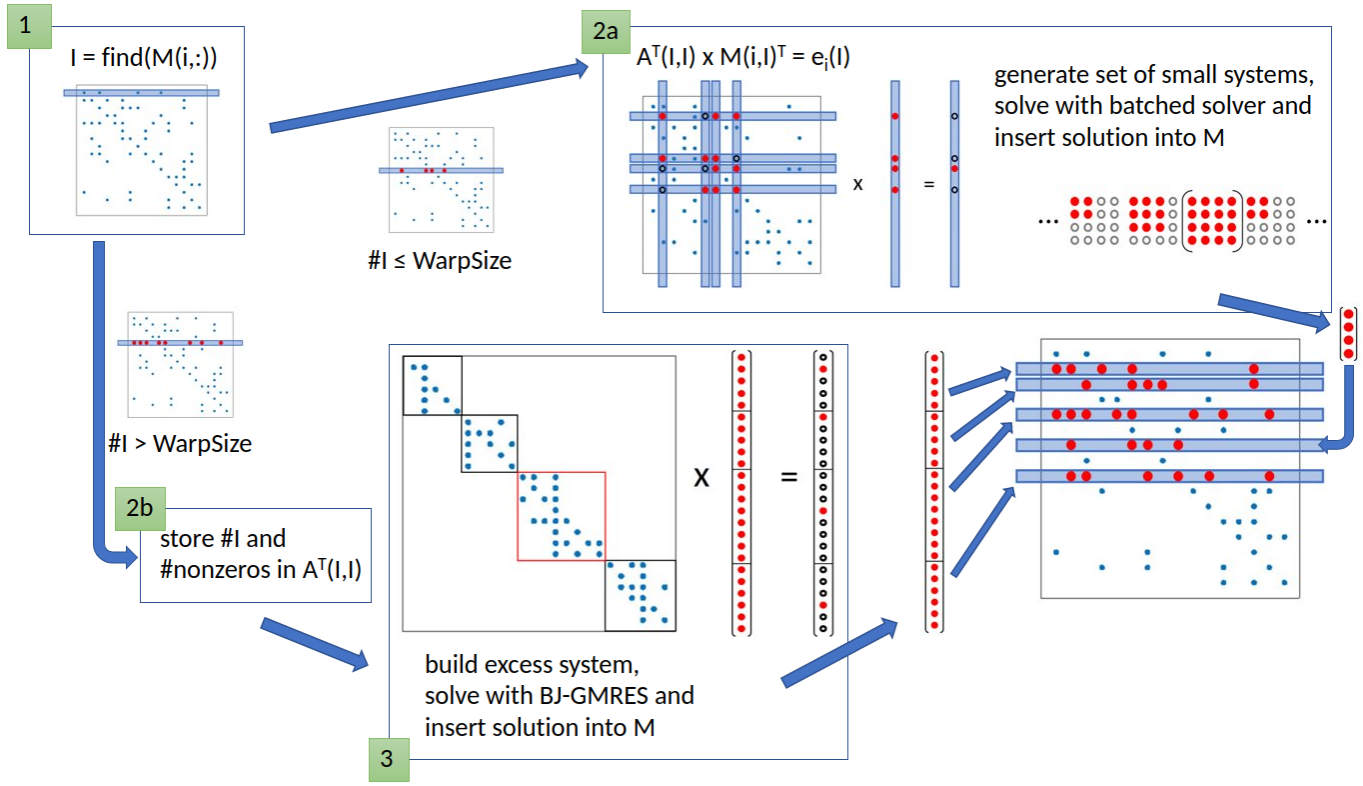
- **Preconditioning iterative solvers.**
  - Idea: Approximate inverse of system matrix to make the system "easier to solve": $P^{-1} \approx A^{-1}$

    and solve $Ax = b \quad \Leftrightarrow \quad P^{-1}Ax = P^{-1}b \quad \Leftrightarrow \quad \tilde{A}x = \tilde{b}.$

- **Sparse Approximate Inverse Preconditioner**
  - $M \approx A^{-1}$ and sparse
  - Incomplete Sparse Approximate Inverse (ISAI)

    uses sparsity pattern of A;
  - Factorized Sparse Approximate Inverse (FSPAI)

    stores inverse approximation in factorized form;



Mixed Precision Sparse Approximate Inverse

- **Preconditioning iterative solvers.**
  - Idea: Approximate inverse of system matrix to make the system "easier to solve": $P^{-1} \approx A^{-1}$

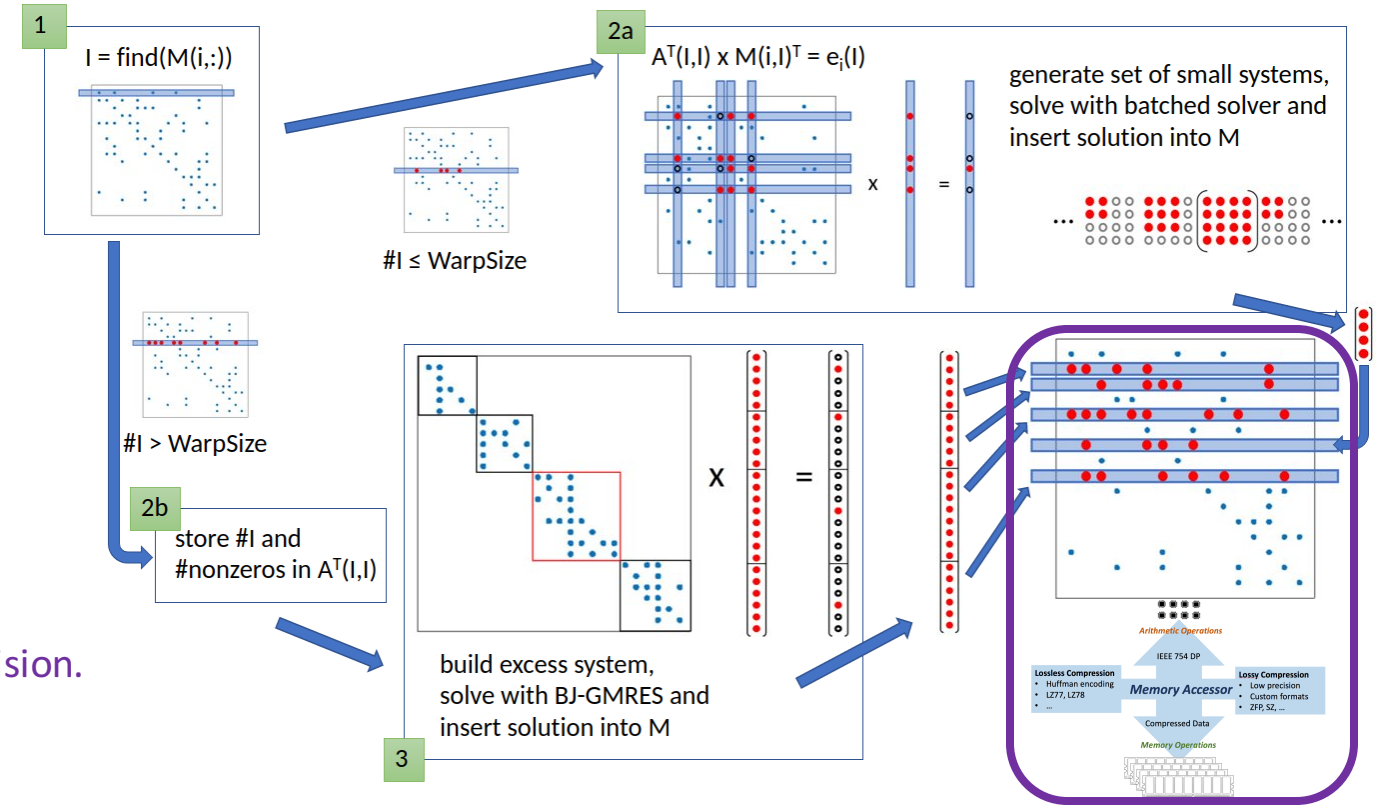
    and solve $Ax = b \quad \Leftrightarrow \quad P^{-1}Ax = P^{-1}b \quad \Leftrightarrow \quad \tilde{A}x = \tilde{b}.$

- **Sparse Approximate Inverse Preconditioner**
  - $M \approx A^{-1}$ and sparse
  - Incomplete Sparse Approximate Inverse (ISAI)

    uses sparsity pattern of A;
  - Factorized Sparse Approximate Inverse (FSPAI)

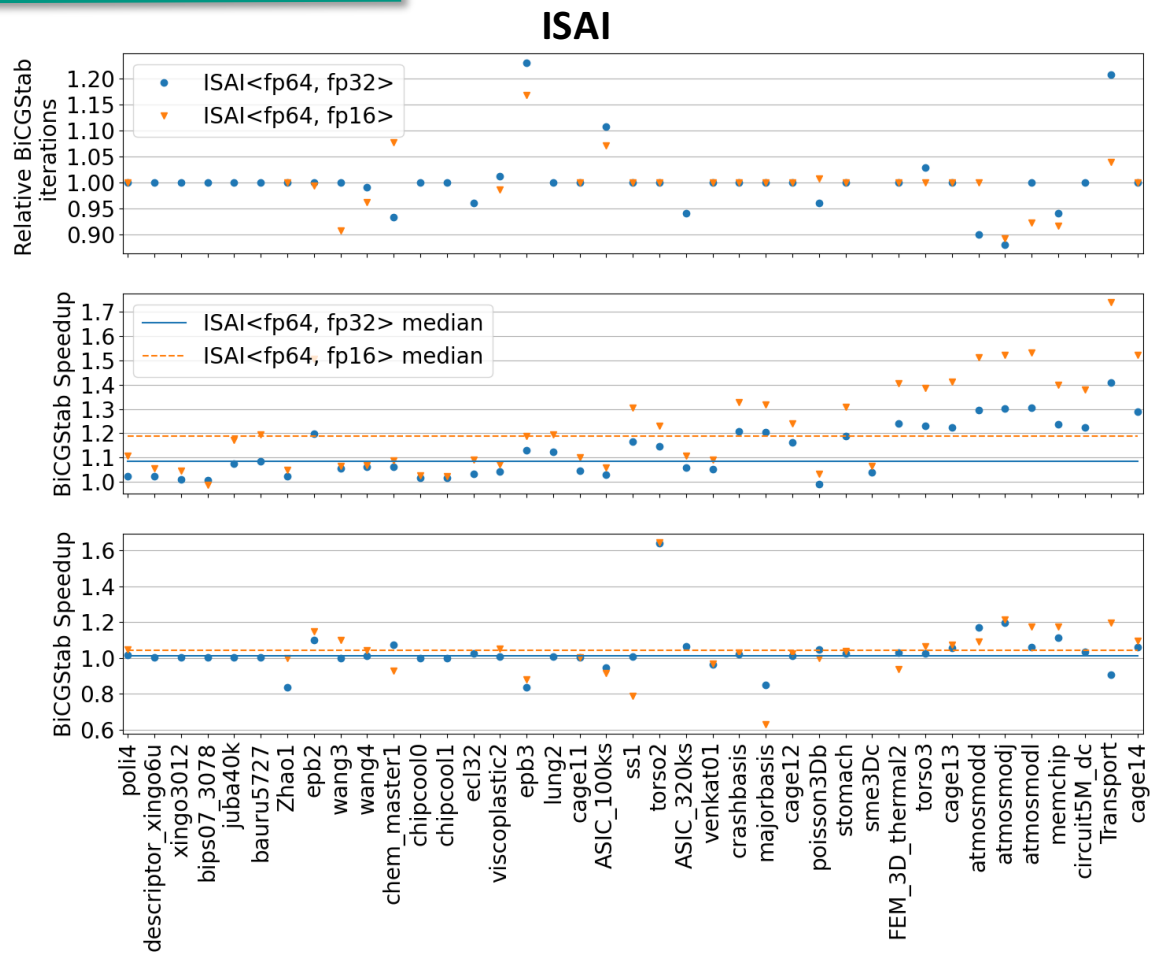
    stores inverse approximation in factorized form;

- Use the accessor to store the preconditioner in lower precision.



Göbel at al: Mixed Precision Incomplete and Factorized Sparse Approximate Inverse Preconditioning on GPUs, EuroPar 2021.

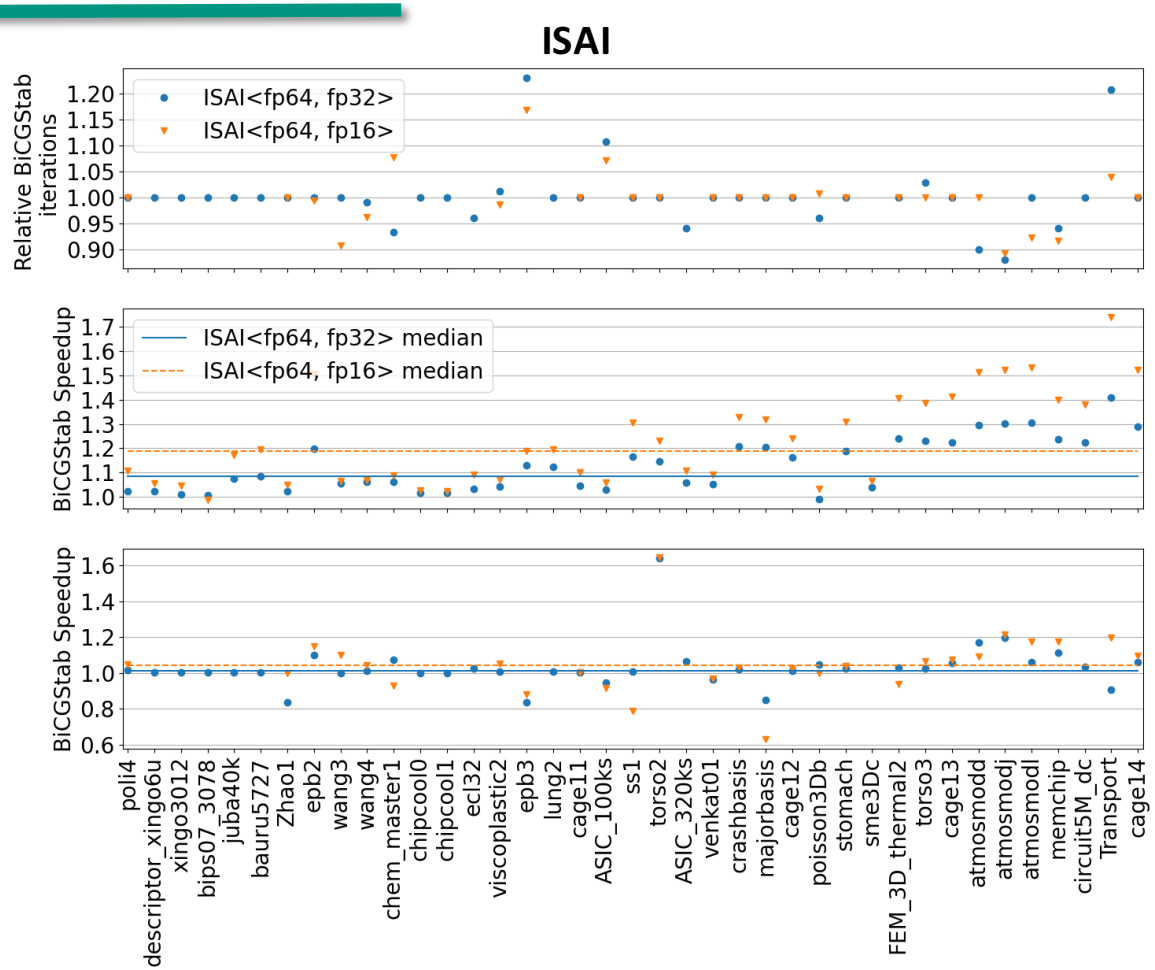
Mixed Precision Sparse Approximate Inverse

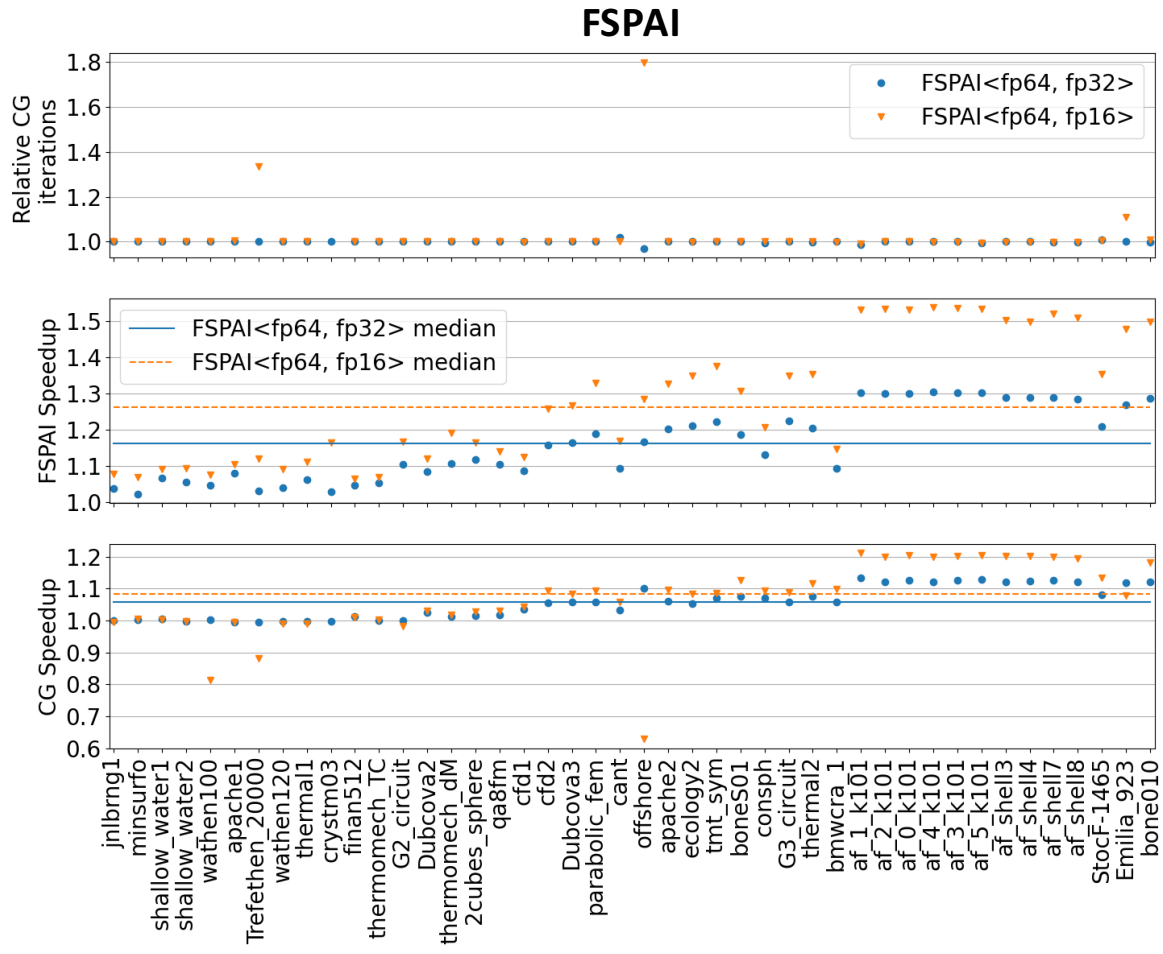


- ~10% speedup for ISAI<fp64,fp32>
- ~20% speedup for ISAI<fp64,fp16> - but unstable!

Göbel at al: Mixed Precision Incomplete and Factorized Sparse Approximate Inverse Preconditioning on GPUs, EuroPar 2021.

Mixed Precision Sparse Approximate Inverse





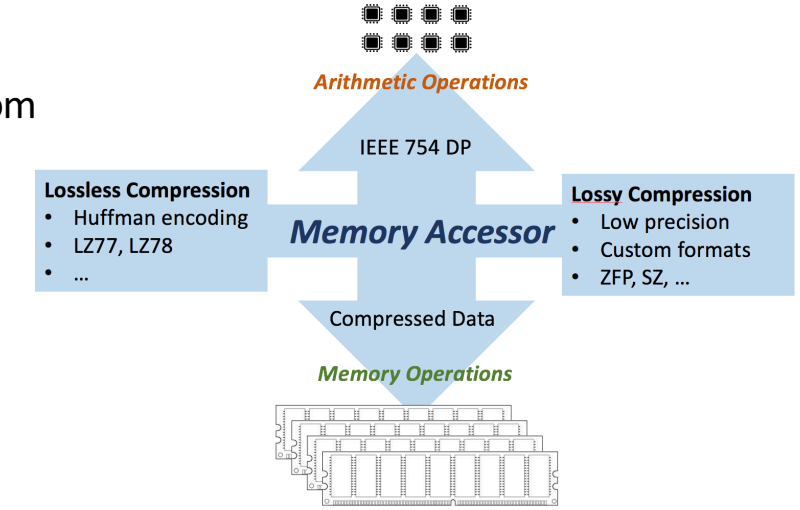
- ~10% speedup for ISAI<fp64,fp32>
- ~20% speedup for ISAI<fp64,fp16> - but unstable!

- ~15% speedup for FSPAI<fp64,fp32>
- ~25% speedup for FSPAI<fp64,fp16>

Göbel at al: Mixed Precision Incomplete and Factorized Sparse Approximate Inverse Preconditioning on GPUs, EuroPar 2021.

Numerical Algorithms in a Modular Precision Ecosystem

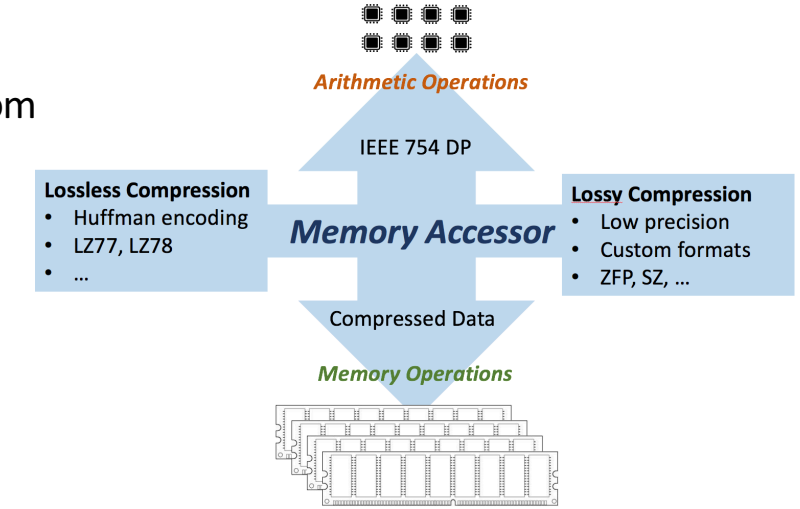
- Need to decouple arithmetic precision from memory precision and consider on-the-fly compression of data;
- Need to re-engineer algorithms to cope with mixed precision / data compression;



- Accessor-BLAS can for free provide higher accuracy for BLAS-1 & BLAS-2 operations;
- If data compression is acceptable, accessor can accelerate memory-bound applications;
  - *self-healing iterative methods*
  - *preconditioners*

Numerical Algorithms in a Modular Precision Ecosystem

- Need to decouple arithmetic precision from memory precision and consider on-the-fly compression of data;
- Need to re-engineer algorithms to cope with mixed precision / data compression;



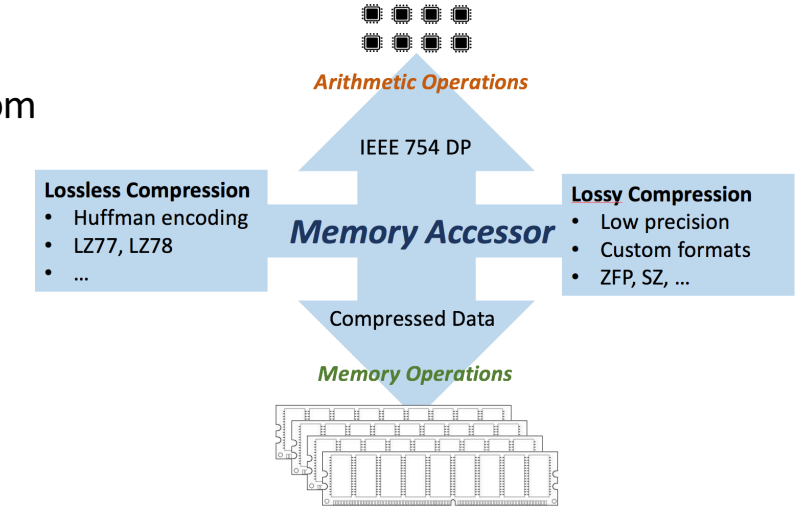
- Accessor-BLAS can for free provide higher accuracy for BLAS-1 & BLAS-2 operations;
- If data compression is acceptable, accessor can accelerate memory-bound applications;
  - *self-healing iterative methods*
  - *preconditioners*

Moving forward

- Accessor-BLAS coverage for BLAS-1 & BLAS-2 ;
- Non-standard formats tuned to data;
- Compression techniques;
- Accessor in other applications & algorithms;

Numerical Algorithms in a Modular Precision Ecosystem

- Need to decouple arithmetic precision from memory precision and consider on-the-fly compression of data;
- Need to re-engineer algorithms to cope with mixed precision / data compression;



- Accessor-BLAS can for free provide higher accuracy for BLAS-1 & BLAS-2 operations;
- If data compression is acceptable, accessor can accelerate memory-bound applications;
  - *self-healing iterative methods*
  - *preconditioners*

Moving forward

- Accessor-BLAS coverage for BLAS-1 & BLAS-2 ;
- Non-standard formats tuned to data;
- Compression techniques;
- Accessor in other applications & algorithms;

Multiprecision Focus Effort within ECP

- Collaborative Effort within US Exascale Computing Effort
- Develop &deploy mixed / multiprecision algorithms for HPC

ECP EXASCALE COMPUTING PROJECT

arXiv.org > cs > arXiv:2007.06674

Search... Help | Advance

Computer Science > Mathematical Software

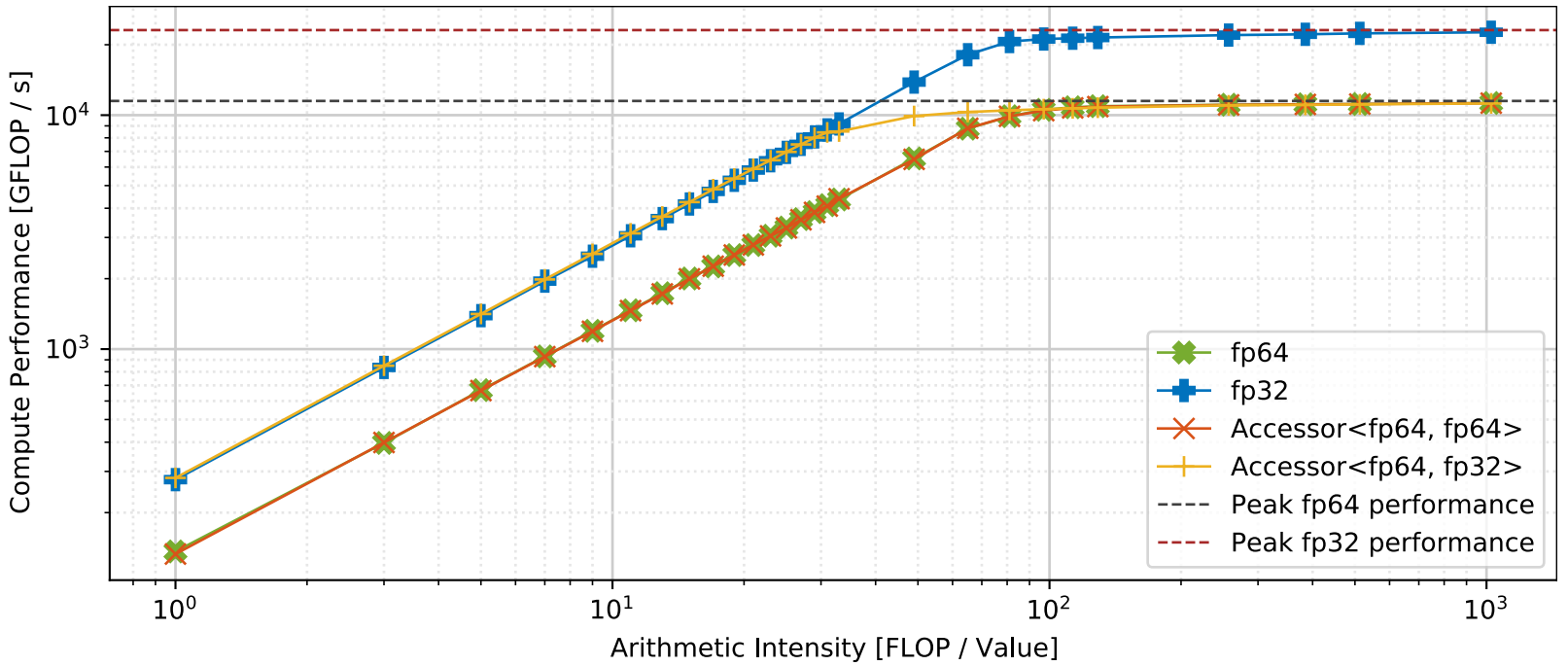
[Submitted on 13 Jul 2020]

A Survey of Numerical Methods Utilizing Mixed Precision Arithmetic

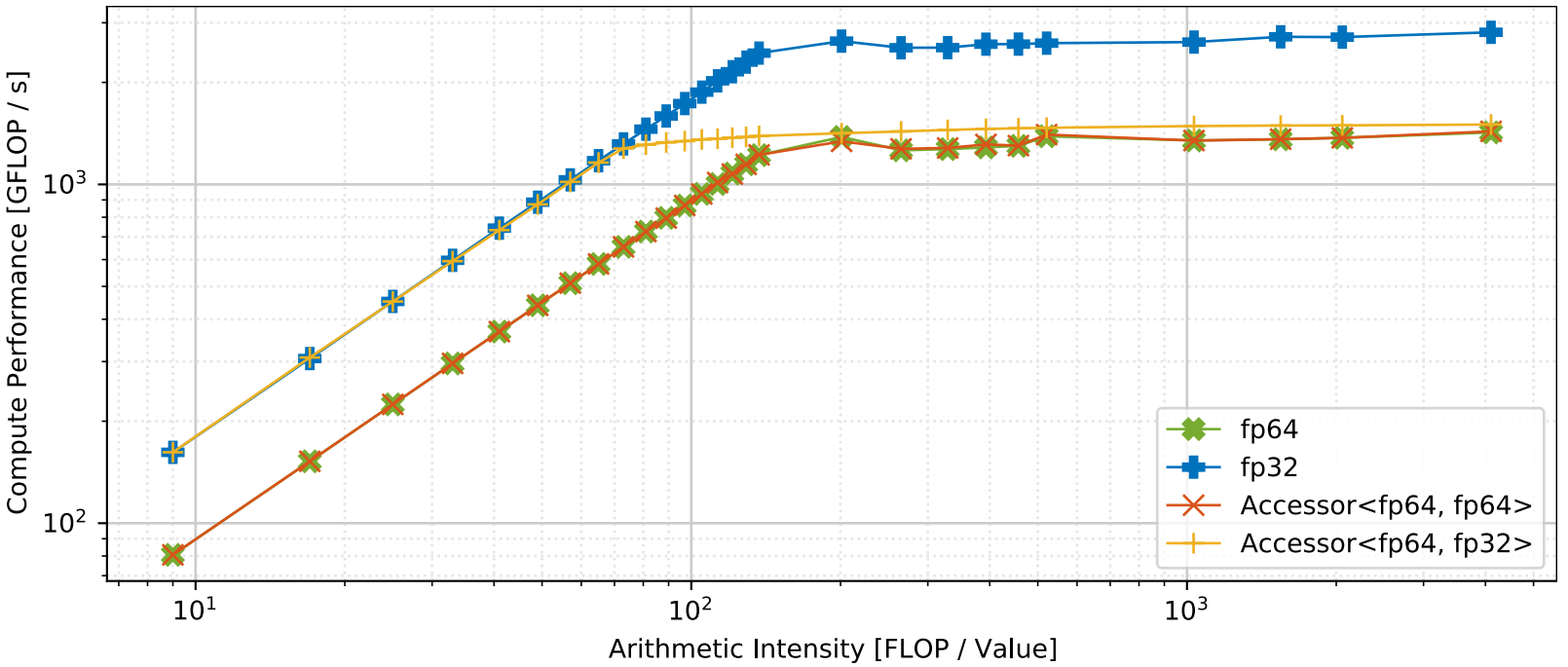
Ahmad Abdelfattah, Hartwig Anzt, Erik G. Boman, Erin Carson, Terry Cojean, Jack Dongarra, Mark Gates, Thomas Grützmacher, Nicholas J. Higham, Sherry Li, Neil Lindquist, Yang Liu, Jennifer Loe, Piotr Luszczek, Pratik Nayak, Sri Pranesh, Siva Rajamanickam, Tobias Ribizel, Barry Smith, Kasia Swirydowicz, Stephen Thomas, Stanimire Tomov, Yaohung M. Tsai, Ichitaro Yamazaki, Urike Meier Yang

Within the past years, hardware vendors have started designing low precision special function units in response to the demand of the Machine Learning community and their demand for high compute power in low precision formats. Also the server-line products are increasingly featuring low-precision special function units, such as the NVIDIA tensor cores in ORNL's Summit supercomputer providing more than an order of magnitude higher performance than what is available in IEEE double precision. At the same time, the gap between the compute power on the one hand and the memory bandwidth on the other hand keeps increasing, making data access and communication prohibitively expensive compared to arithmetic operations. To start the multiprecision focus effort, we survey the numerical linear algebra community and summarize all existing multiprecision knowledge, expertise, and software capabilities in this landscape analysis report. We also include current efforts and preliminary results that may not yet be considered "mature technology," but have the potential to grow into production quality within the multiprecision focus effort. As we expect the reader to be familiar with the basics of numerical linear algebra, we refrain from providing a detailed background on the algorithms themselves but focus on how mixed- and multiprecision technology can help improving the performance of these methods and present highlights of application significantly outperforming the traditional fixed precision methods.

Accessor for AMD MI100 GPU



Accessor for Intel Skylake CPU



Accessor for AMD EPYC CPU