Addressing the Communication Bottleneck:

Towards a Modular Precision Ecosystem for High Performance Computing

Focus Session: New Approaches, Algorithms Towards Exascale Computing
ISC High Performance 2019, June 19th, Frankfurt

Hartwig Anzt, Terry Cojean, Goran Flegar, Thomas Grűtzmacher, Pratik Nayak, Tobias Ribizel
Steinbuch Centre for Computing (SCC)


Terry Cojean


Goran Flegar


Thomas Grützmacher


Pratik Nayak


Tobias Ribizel

Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

summit

- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

summit

- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

Roofline Model

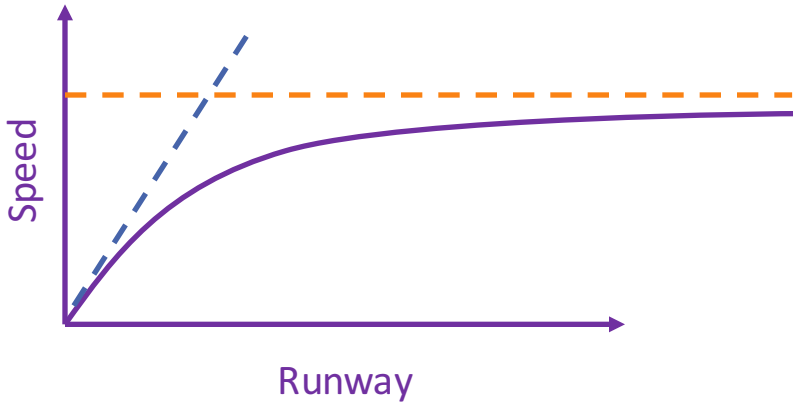
Given certain hardware characteristics:

memory bandwidth,
arithmetic power,

Acceleration
Top Speed

the performance of any operation is

- either bound by the data access/communication (*memory bound*),
- or by the arithmetic operations (*compute bound*).



Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

summit

- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

Roofline Model

Given certain hardware characteristics:

memory bandwidth,
arithmetic power,

Acceleration
Top Speed

the performance of any operation is

- either bound by the data access/communication (*memory bound*),
- or by the arithmetic operations (*compute bound*).

Matrix-Matrix Product (GEMM): $C = A \times B$ $\quad A, B, C \in \mathbb{R}^{n \times n}$

$3n^2$ Memory operations
$2n^3$ Arithmetic operations

Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

summit

- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

Copyright@ORNL

Roofline Model

Given certain hardware characteristics:

memory bandwidth,
arithmetic power,

Acceleration
Top Speed

the performance of any operation is

- either bound by the data access/communication (*memory bound*),
- or by the arithmetic operations (*compute bound*).

Matrix-Matrix Product (GEMM): $C = A \times B$ $\quad A, B, C \in \mathbb{R}^{n \times n}$

$3n^2$ Memory operations
$2n^3$ Arithmetic operations

We just need to increase the size, and at some point the operation becomes compute bound.

"we infinitely extend the acceleration runway"

Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

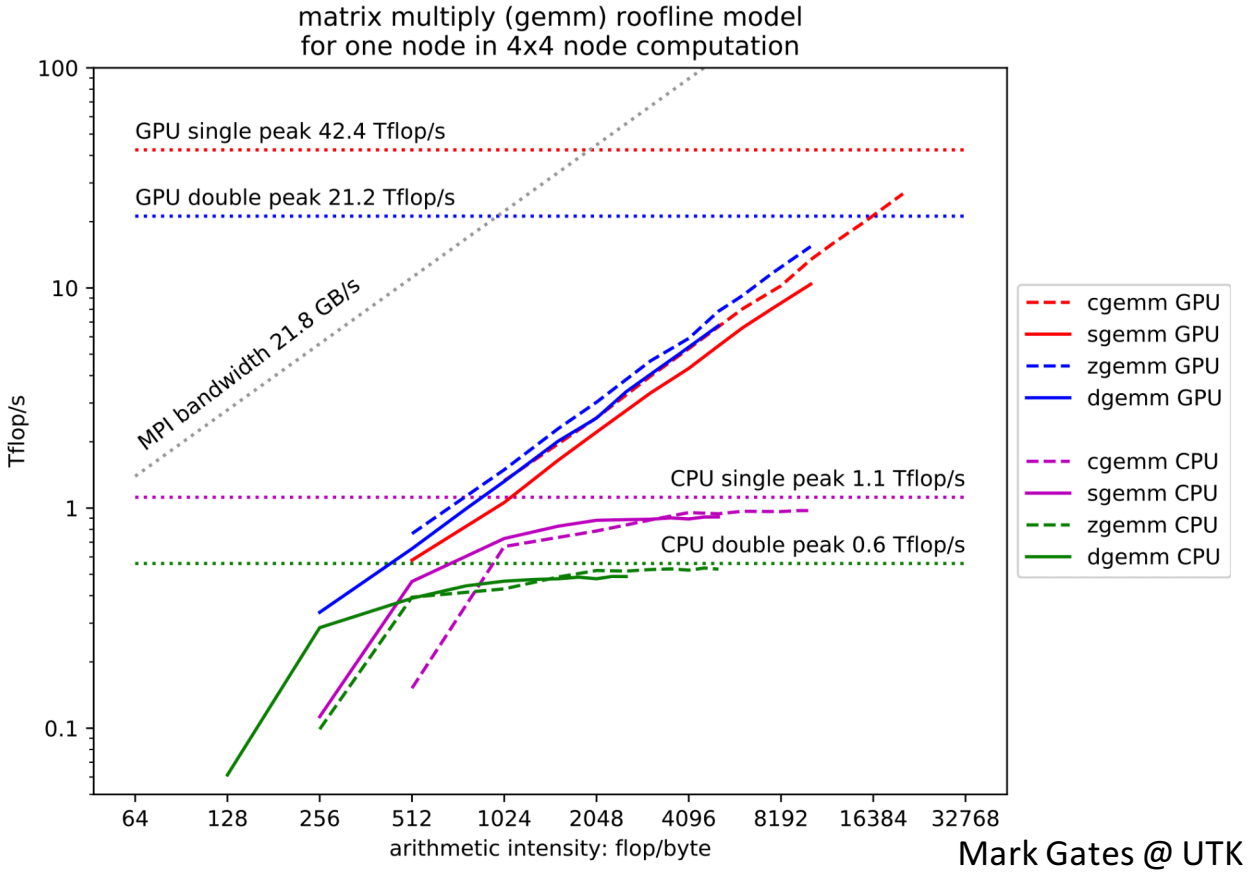
summit

- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

Copyright@ORNL

Dense Matrix Operations?

- The inter-node communication is the limiting resource;
- Each node has more computational power than what we can leverage;



Mark Gates @ UTK

Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

summit

- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

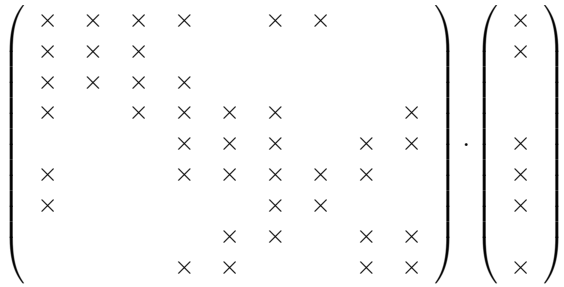
Copyright@ORNL

Dense Matrix Operations?

- The inter-node communication is the limiting resource;
- Each node has more computational power than what we can leverage;

Sparse / Graph Problems?

- ***Sp**arse **M**atrix **V**ector Product* (SpMV) is a central building block;



Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

summit

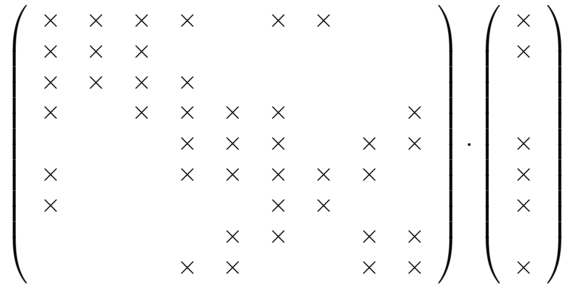
- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

Dense Matrix Operations?

- The inter-node communication is the limiting resource;
- Each node has more computational power than what we can leverage;

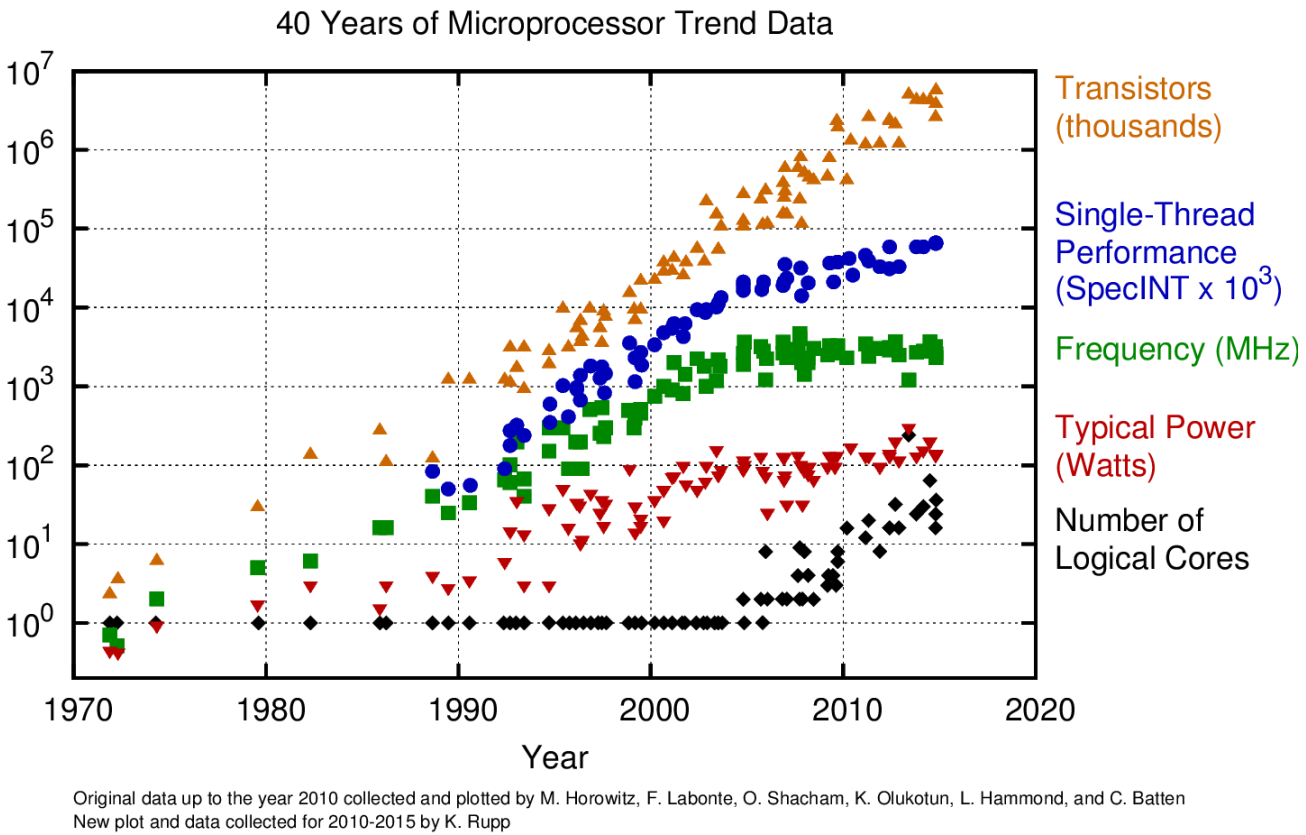
Sparse / Graph Problems?

- ***Sp**arse **M**atrix **V**ector Product* (SpMV) is a central building block;
- It looks like for the test problems at the SuiteSparse Matrix Collection[1], a Multi-node SpMV is slower than a Single-node SpMV;
- The inter-node communication is an order of magnitude slower than the local computations.



[1]SuiteSparse Matrix Collection: https://sparse.tamu.edu/

How did we get there?

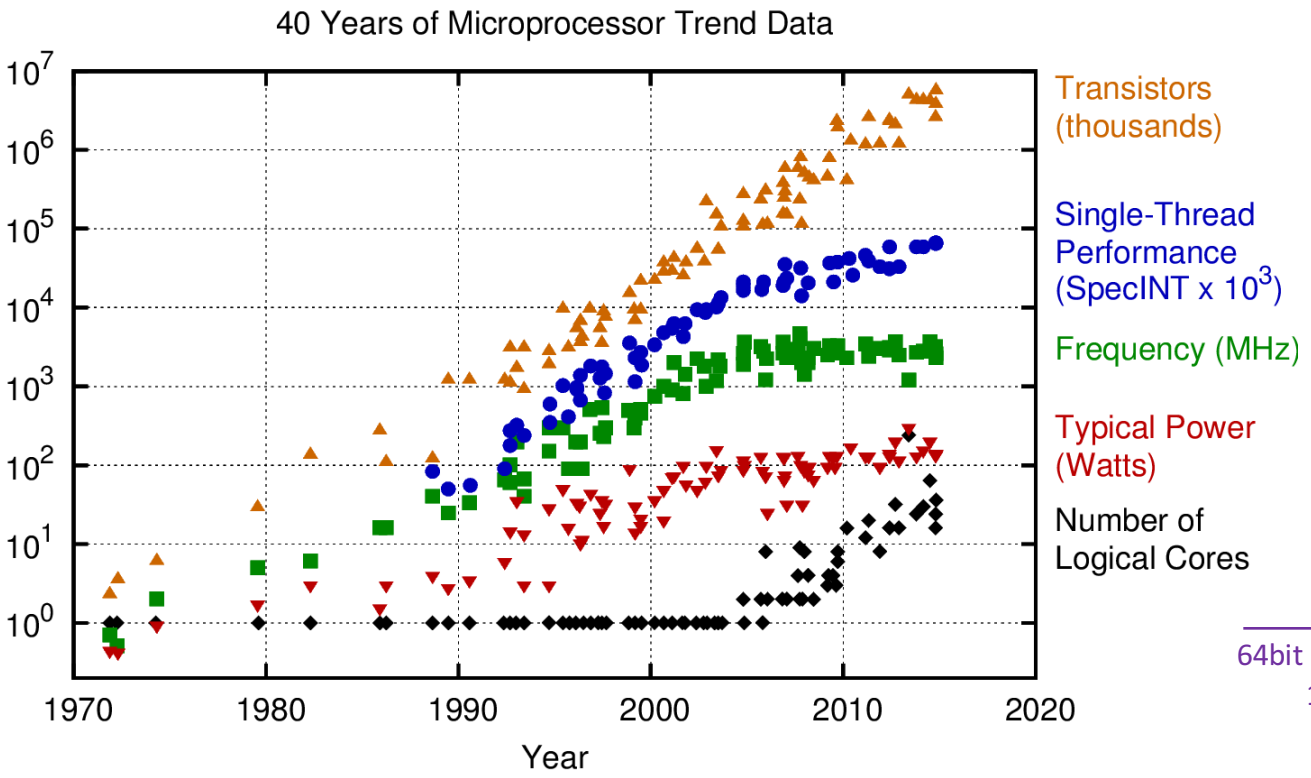


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

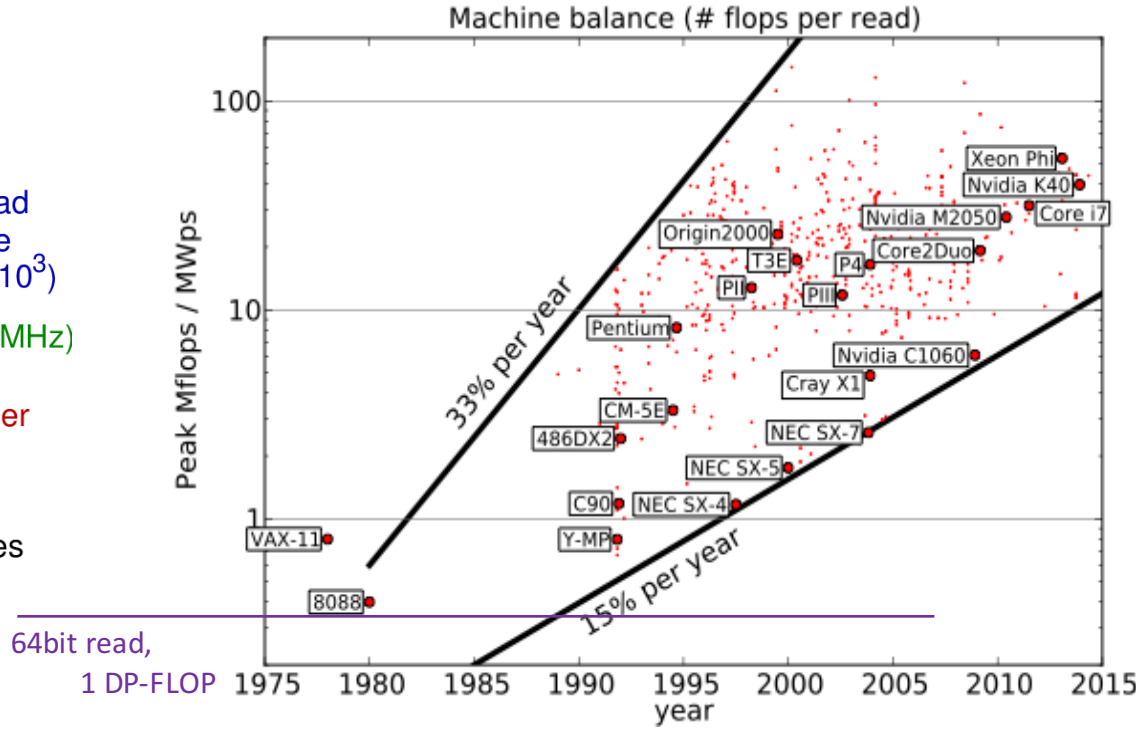
- Explosion in core count.

"Parallelism needed – synchronization kills performance."

How did we get there?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp



John D. McCalpin (TACC)

- Explosion in core count.

  "Parallelism needed – synchronization kills performance."

- Compute power (#FLOPs) grows much faster than bandwidth.

  "Operations are free, mem access and comm is what counts."

How did we get there?

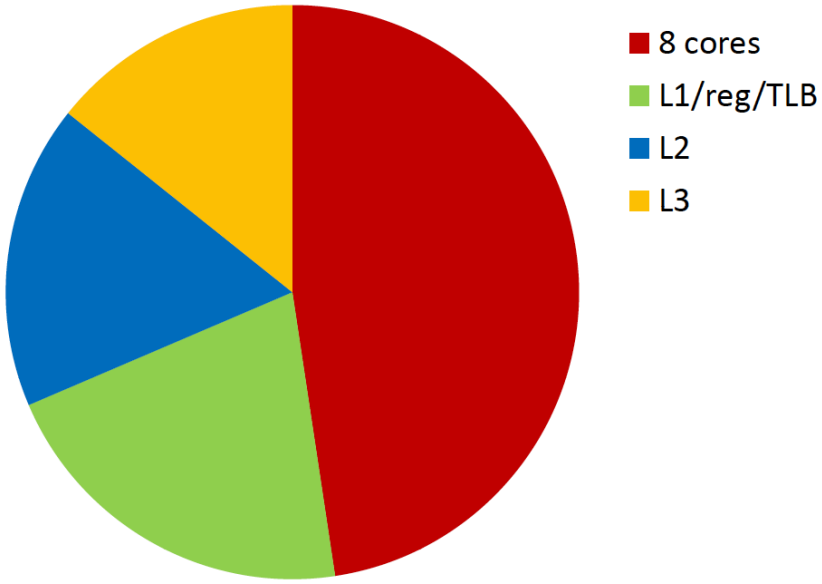
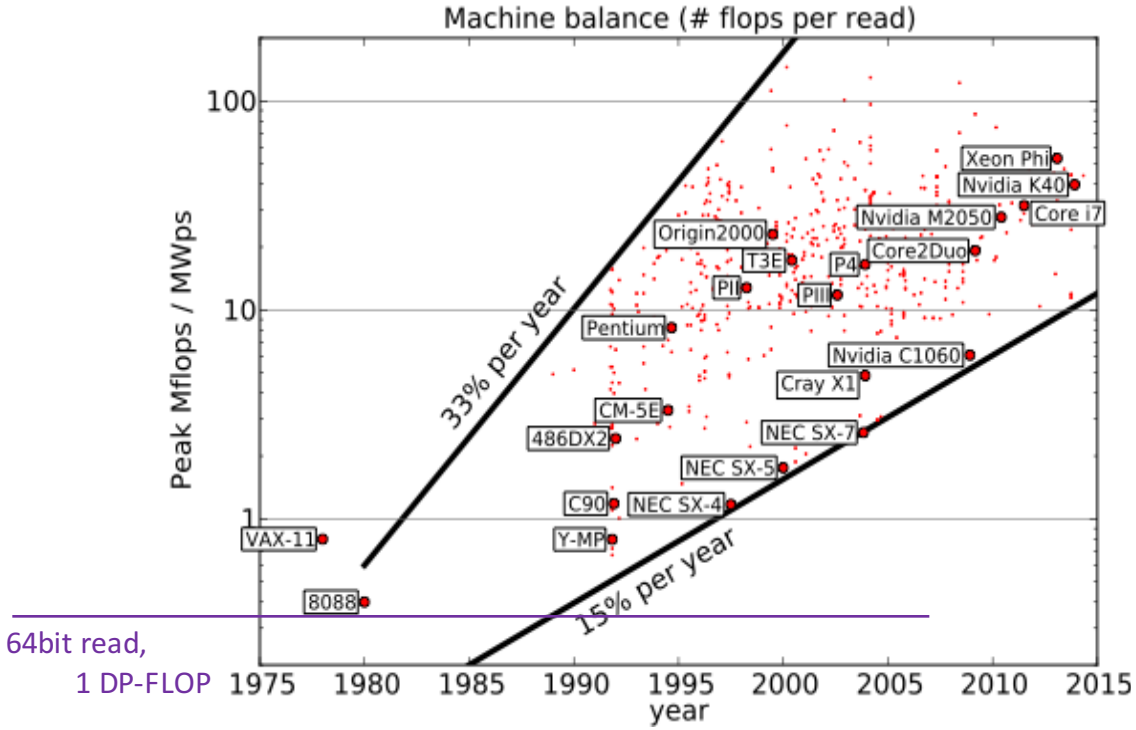


Figure 1.1.7: Power breakdown of an 8 core server chip.

Mark Horowitz (2014): **Computing's energy problem (and what we can do about it)**



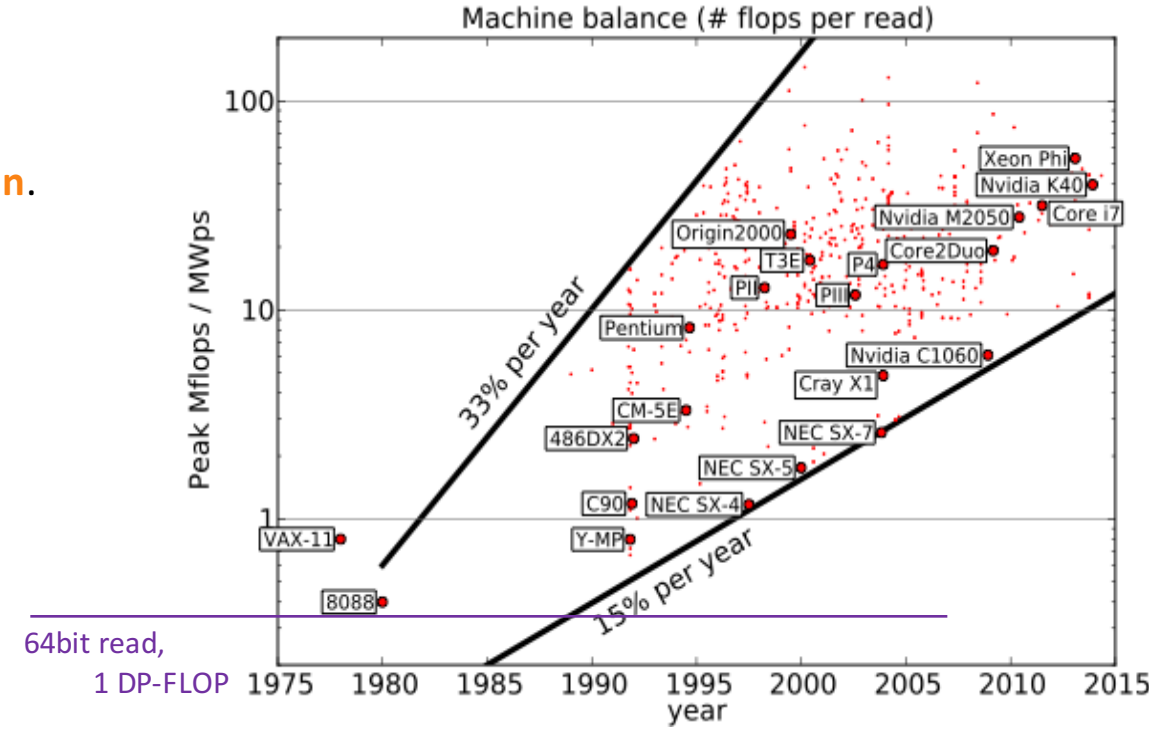
John D. McCalpin (TACC)

- Compute power (#FLOPs) grows much faster than bandwidth.

"Operations are free, mem access and comm is what counts."

Algorithms reflecting Hardware Evolution

- The arithmetic operations should use high precision formats natively supported by hardware.
- **Data access** should be as cheap as possible, **reduced precision**.



John D. McCalpin (TACC)

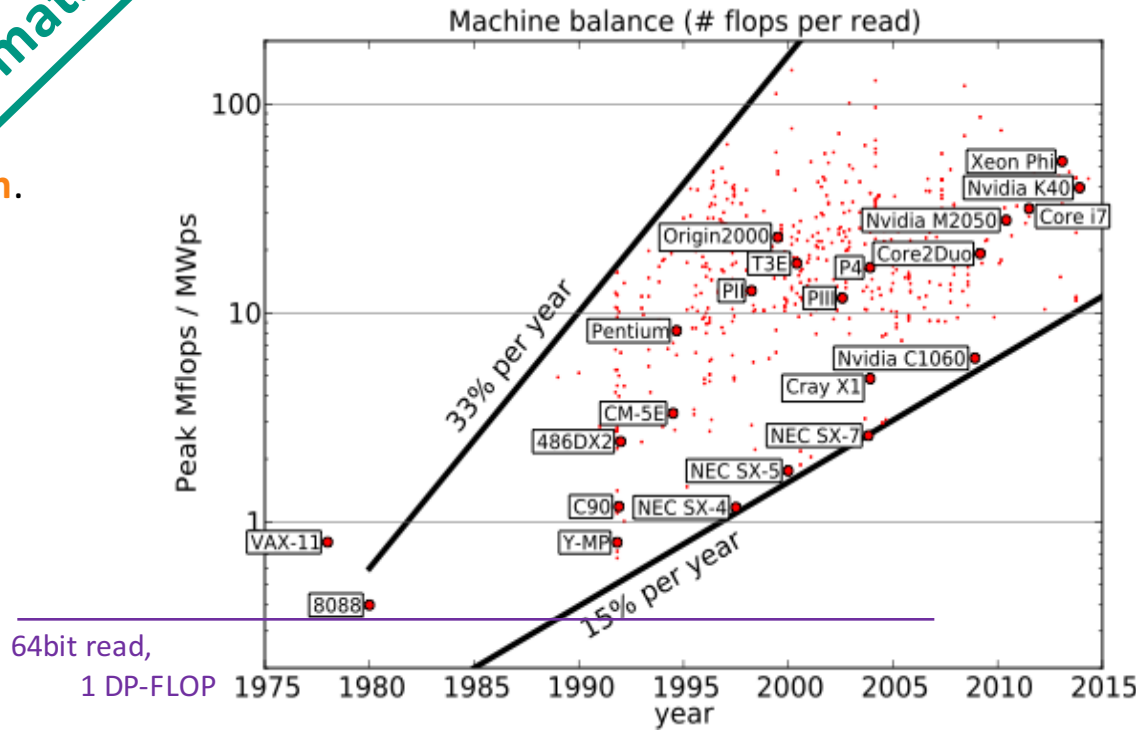
- Compute power (#FLOPs) grows much faster than bandwidth.

"Operations are free, mem access and comm is what counts."

Algorithms reflecting Hardware Evolution

- The arithmetic operations should use high precision forma
  natively supported by hardware.
- **Data access** should be as cheap as possible, **reduc** **sion**.

Radically decouple storage format from arithmetic format.



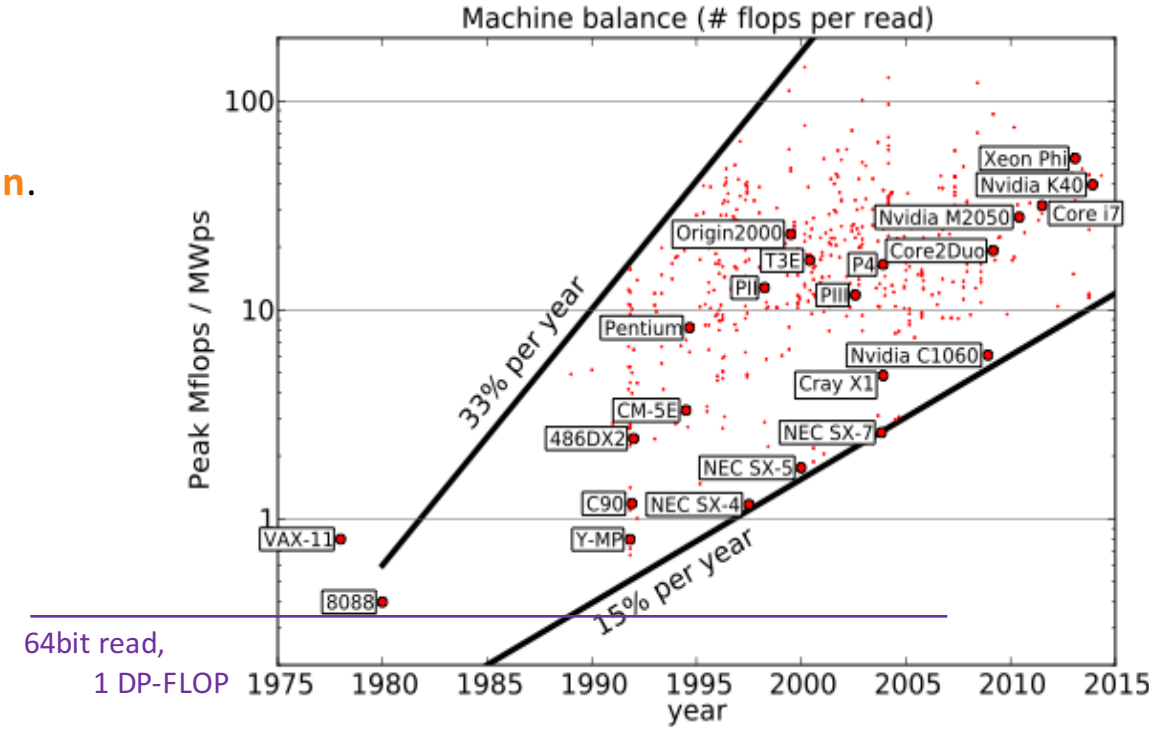
John D. McCalpin (TACC)

- Compute power (#FLOPs) grows much faster than bandwidth.

"Operations are free, mem access and comm is what counts."

Algorithms reflecting Hardware Evolution

- The arithmetic operations should use high precision formats natively supported by hardware.
- **Data access** should be as cheap as possible, **reduced precision**.
- **Radically decouple storage format from arithmetic format.**
  - *Double precision in all arithmetic operations.*
  - *Algorithm-aware precision when accessing data.*



John D. McCalpin (TACC)

- Compute power (#FLOPs) grows much faster than bandwidth.

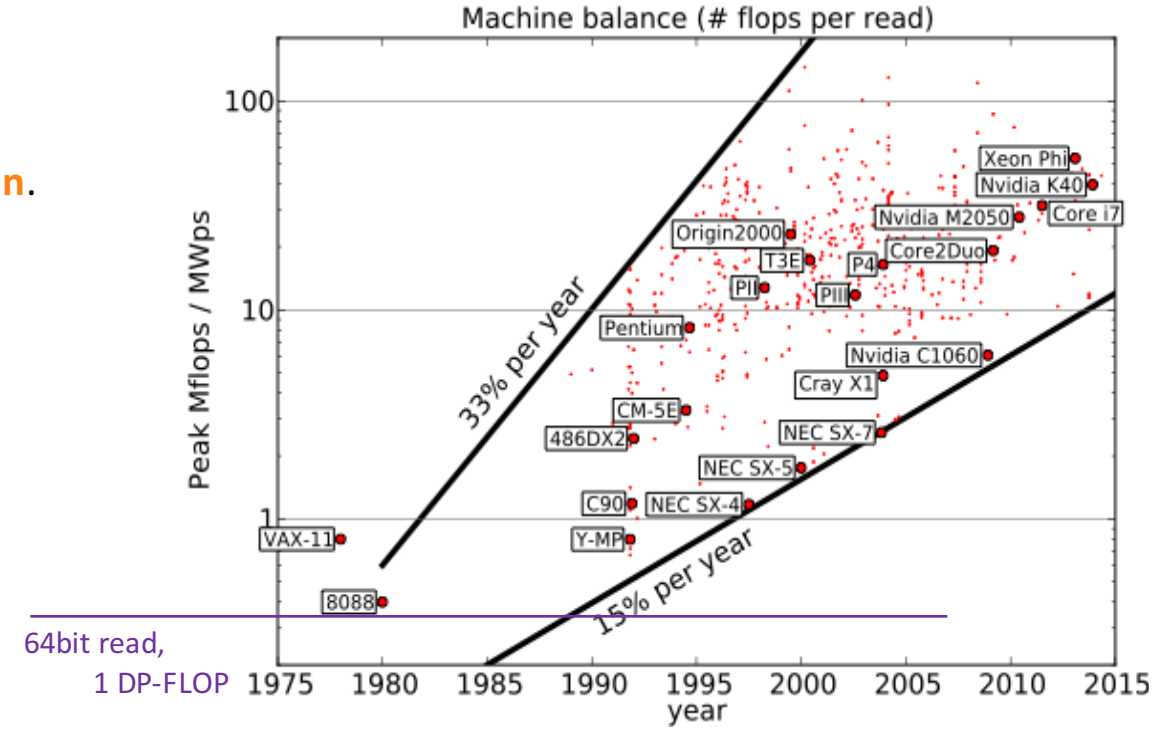
"Operations are free, mem access and comm is what counts."

Algorithms reflecting Hardware Evolution

- The arithmetic operations should use high precision formats natively supported by hardware.
- **Data access** should be as cheap as possible, **reduced precision**.
- **Radically decouple storage format from arithmetic format.**
  - *Double precision in all arithmetic operations.*
  - *Algorithm-aware precision when accessing data.*

Challenges when using double precision
+ IEEE single/half precision:

- Need explicit conversion.
- Data range reduction: protect against under- / overflow.
- Need to duplicate data in memory (half/single/double).



John D. McCalpin (TACC)

- Compute power (#FLOPs) grows much faster than bandwidth.

"Operations are free, mem access and comm is what counts."

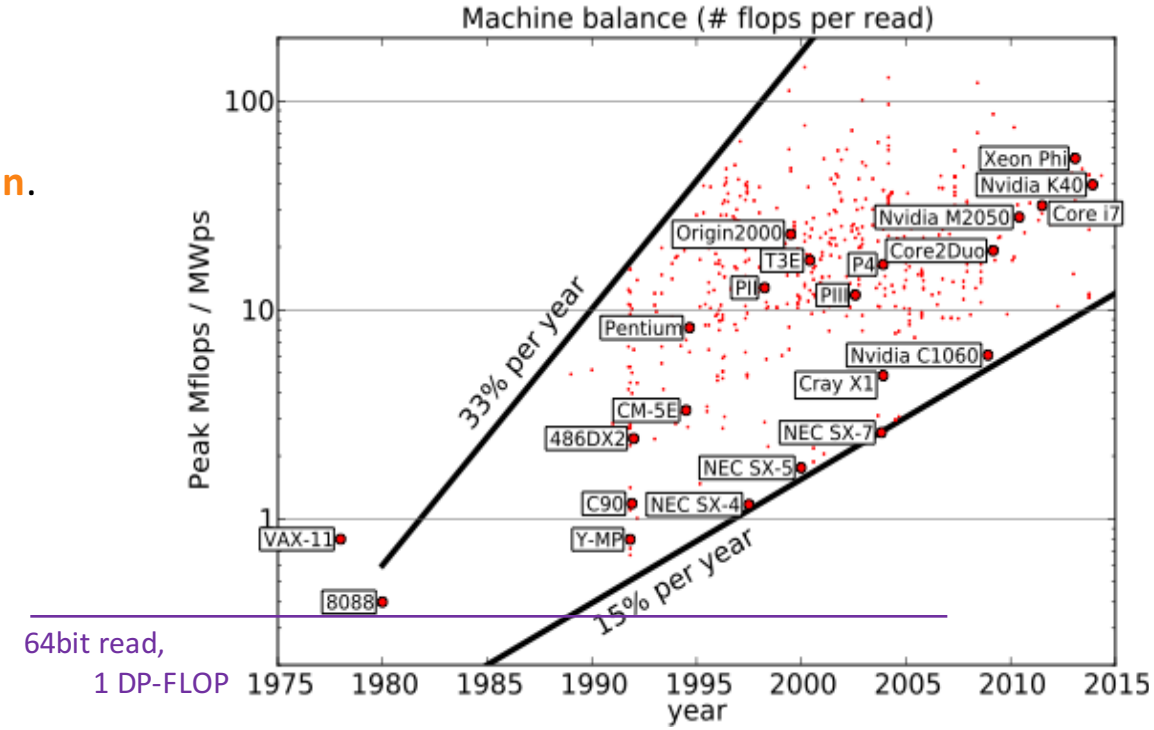
Algorithms reflecting Hardware Evolution

- The arithmetic operations should use high precision formats natively supported by hardware.
- **Data access** should be as cheap as possible, **reduced precision**.
- **Radically decouple storage format from arithmetic format.**
  - *Double precision in all arithmetic operations.*
  - *Algorithm-aware precision when accessing data.*

Challenges when using double precision
+ IEEE single/half precision:

- Need explicit conversion.
- Data range reduction: protect against under- / overflow.
- Need to duplicate data in memory (half/single/double).

- Better: **Customized Precision Format**



John D. McCalpin (TACC)

- Compute power (#FLOPs) grows much faster than bandwidth.

"Operations are free, mem access and comm is what counts."

Use reduced precision for “approximate Operators”

- **Preconditioners for iterative solvers.**

Use reduced precision for “approximate Operators”

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

$$Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}$$

Use reduced precision for “approximate Operators”

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

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

Use reduced precision for “approximate Operators”

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

    $\tilde{A} = P^{-1}A\,, \quad \tilde{b} = P^{-1}b\,,$ and we solve $Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}\,.$
- **Why should we store the preconditioner matrix P^{-1} in full (high) precision?**

Use reduced precision for “approximate Operators”

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

    $\tilde{A} = P^{-1}A\,, \quad \tilde{b} = P^{-1}b\,,$ and we solve $Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}\,.$
- **Why should we store the preconditioner matrix P^{-1} in full (high) precision?**
  - **We have to ensure regularity!**

Use reduced precision for “approximate Operators”

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

    $\tilde{A} = P^{-1}A\,, \quad \tilde{b} = P^{-1}b\,,$ and we solve $Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}\,.$
- **Why should we store the preconditioner matrix P^{-1} in full (high) precision?**
  - **We have to ensure regularity!** (Reducing precision can turn matrix singular)

- **Jacobi method** based on **diagonal scaling:** $P = diag(A)$

Use reduced precision for “approximate Operators”

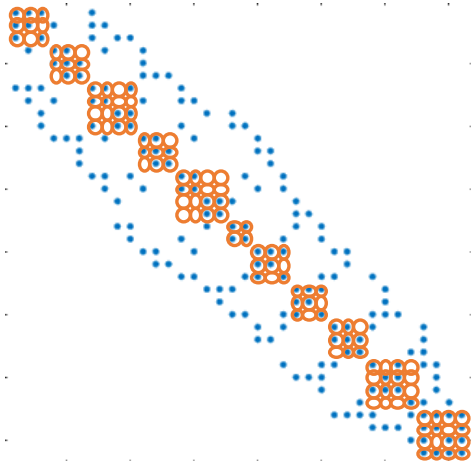
- **Preconditioners for iterative solvers.**
  - Idea: Approximate inverse of system matrix to make the system “easier to solve”: $P^{-1} \approx A^{-1}$

    $\tilde{A} = P^{-1}A\,, \quad \tilde{b} = P^{-1}b\,,$ and we solve $Ax = b \Leftrightarrow \tilde{A}x = \tilde{b}\,.$
- **Why should we store the preconditioner matrix P^{-1} in full (high) precision?**
  - **We have to ensure regularity!** (Reducing precision can turn matrix singular)

- **Jacobi method** based on **diagonal scaling:** $P = diag(A)$

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

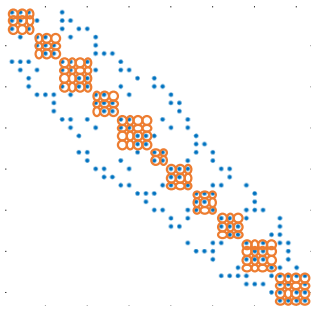
    Extreme case: one block of matrix size.

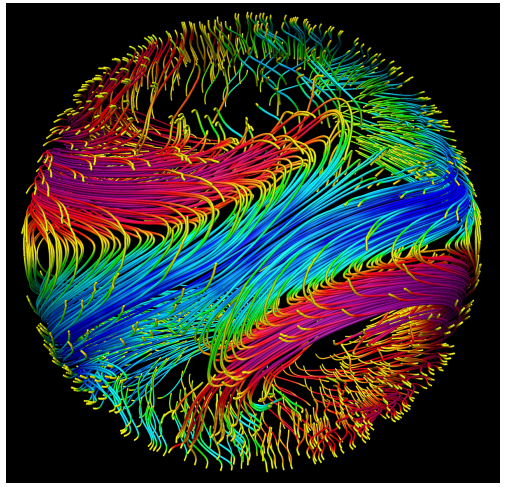


Block-Jacobi Preconditioning

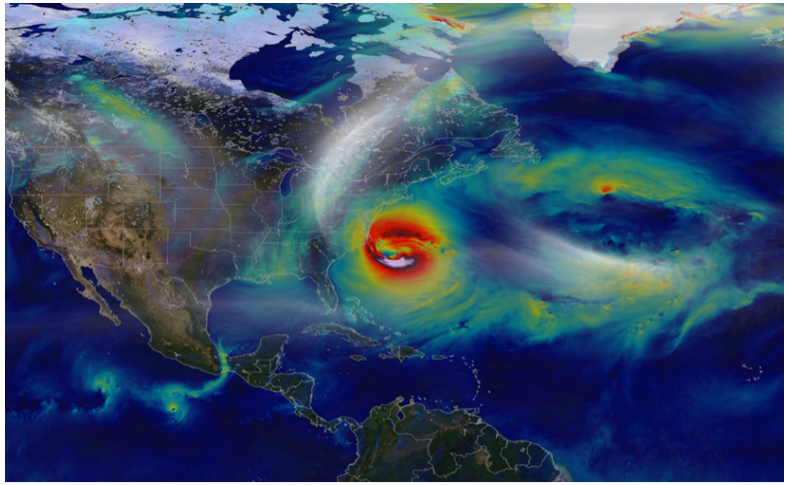
- **Block-Jacobi** method typically used as **preconditioner** inside **linear- / Eigenvalue solvers**.
- Target: large, sparse linear systems.

  discretizations often carry a **block-structure** (multiple variables per node).
- "Natural blocks" of small size (8, 12..).
- System matrix often stored in **sparse data structure** (CSR).

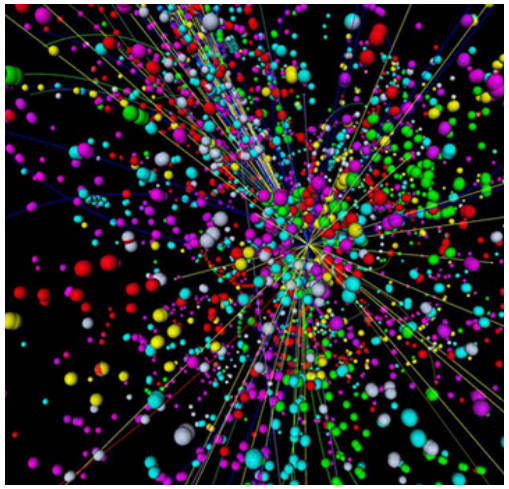




http://www.nas.nasa.gov/SC14/



https://science.nasa.gov/earth-science/focus-areas/earth-weather



http://i.livescience.com

Block-Jacobi Preconditioning

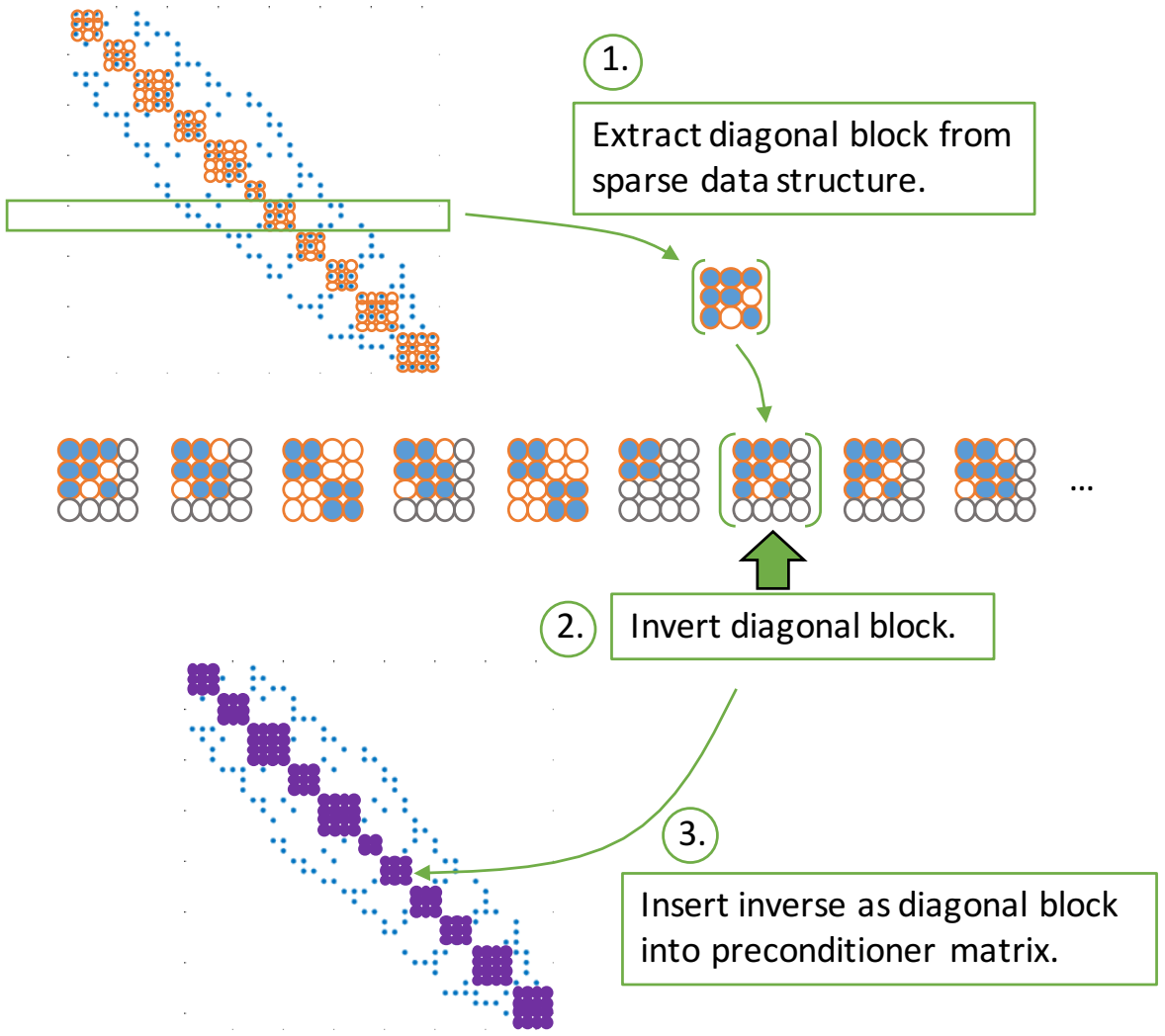
Preconditioner Setup:

- Identify the diagonal blocks $P = diag_B(A)$
- Form the block-Inverse $P^{-1} \approx A^{-1}$

Preconditioner Application:

- Apply the preconditioner in every solver iteration via:

$$y := P^{-1}x$$



Block-Jacobi Preconditioning

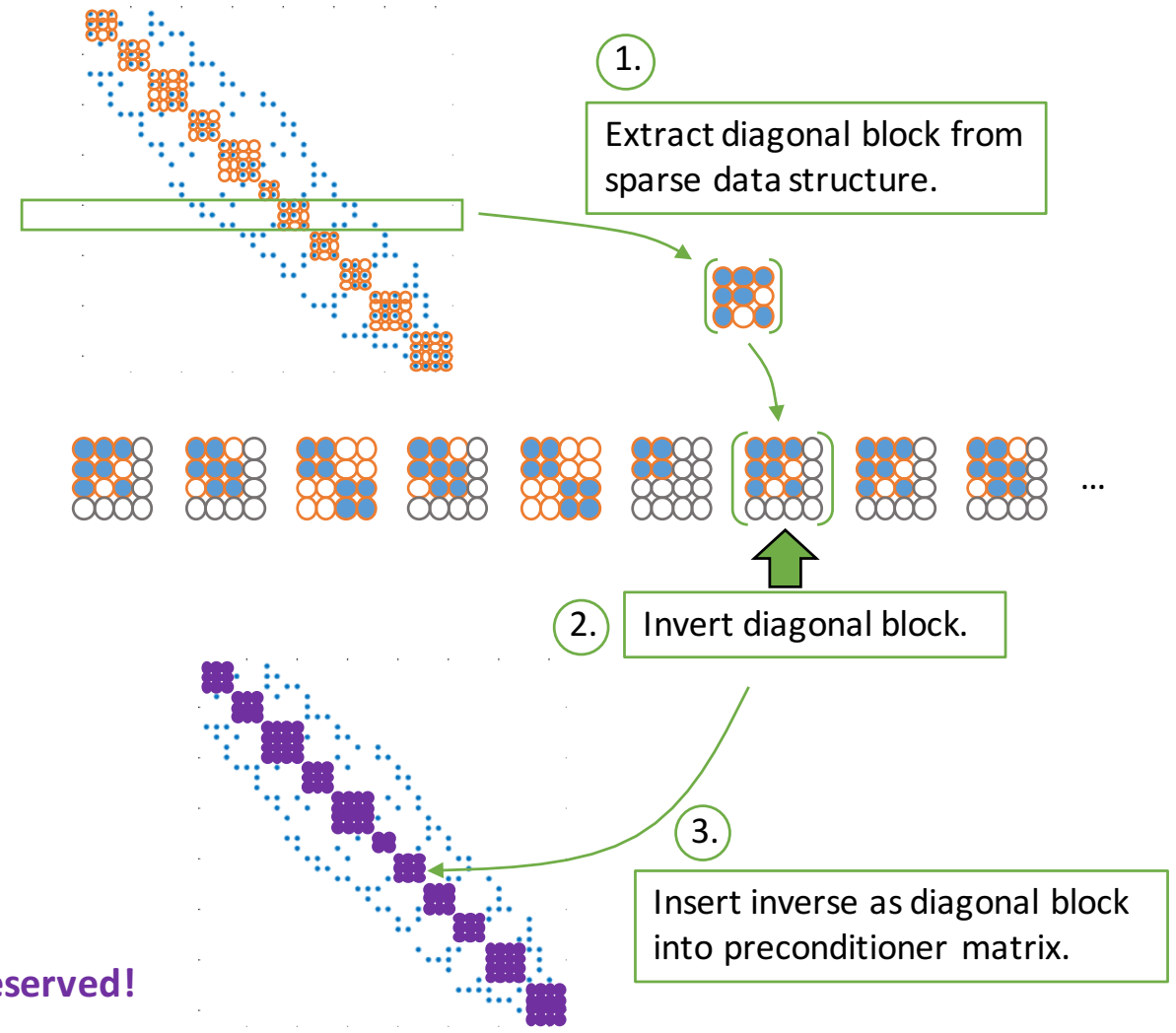
Preconditioner Setup:

- Identify the diagonal blocks $P = diag_B(A)$
- Form the block-Inverse $P^{-1} \approx A^{-1}$

Preconditioner Application:

- Apply the preconditioner in every solver iteration via:

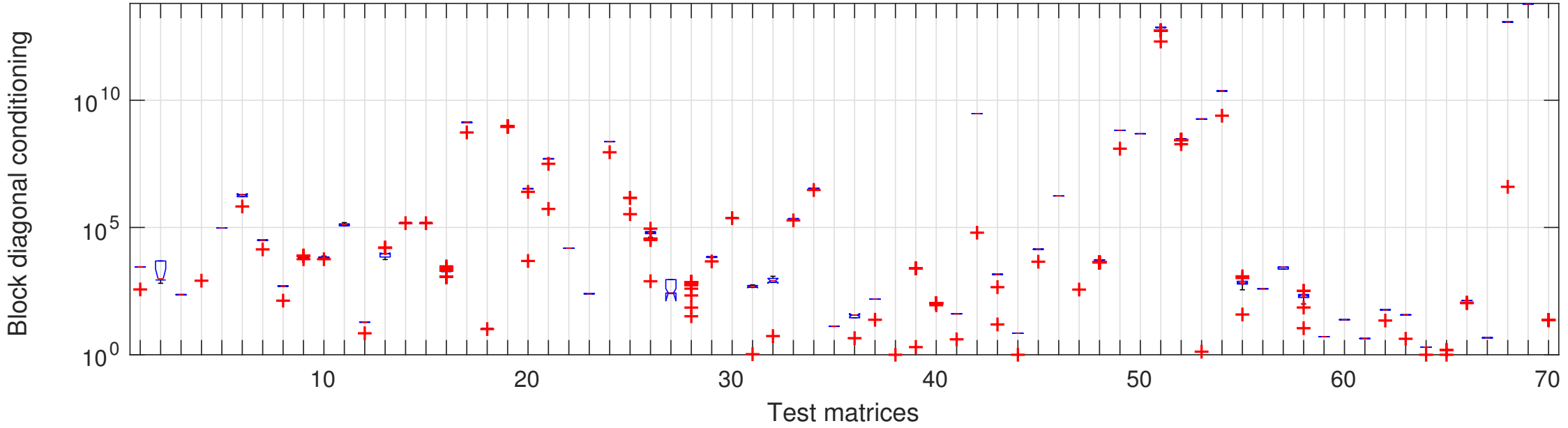
$$y := P^{-1}x$$



We can store diagonal blocks in lower precision, if regularity is preserved!

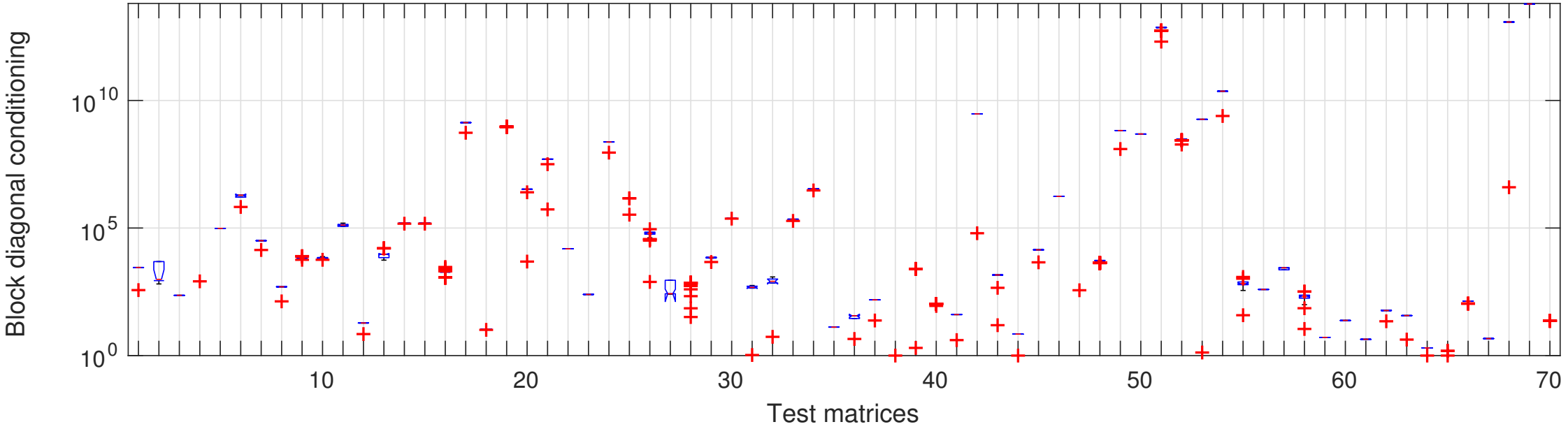
Adaptive Precision Block-Jacobi Preconditioning

- 70 matrices from the SuiteSparse Matrix Collection
- Use block-size 24 with Super-Variable agglomeration (24 is upper bound for size of blocks)
- Report conditioning of all arising diagonal blocks

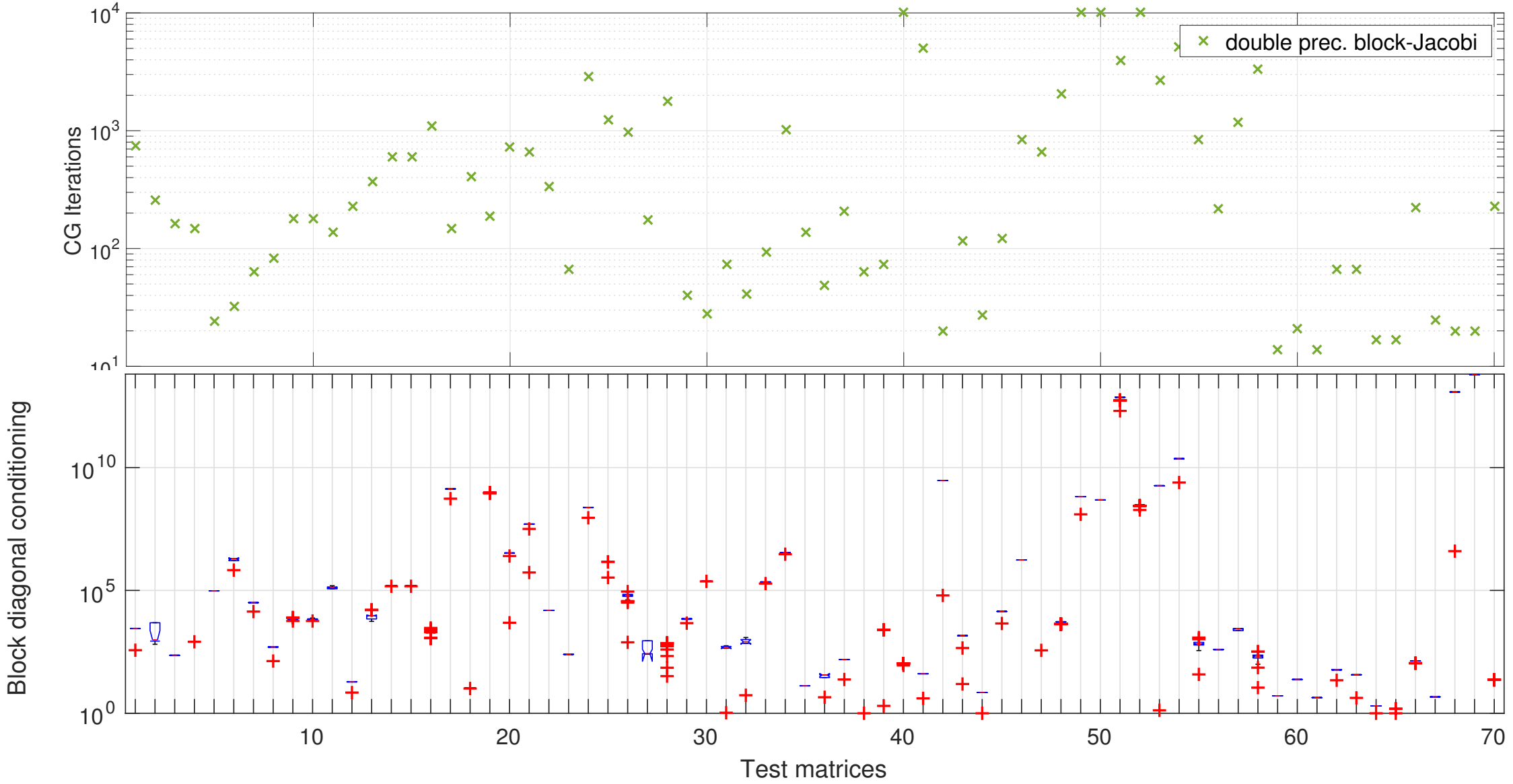


Adaptive Precision Block-Jacobi Preconditioning

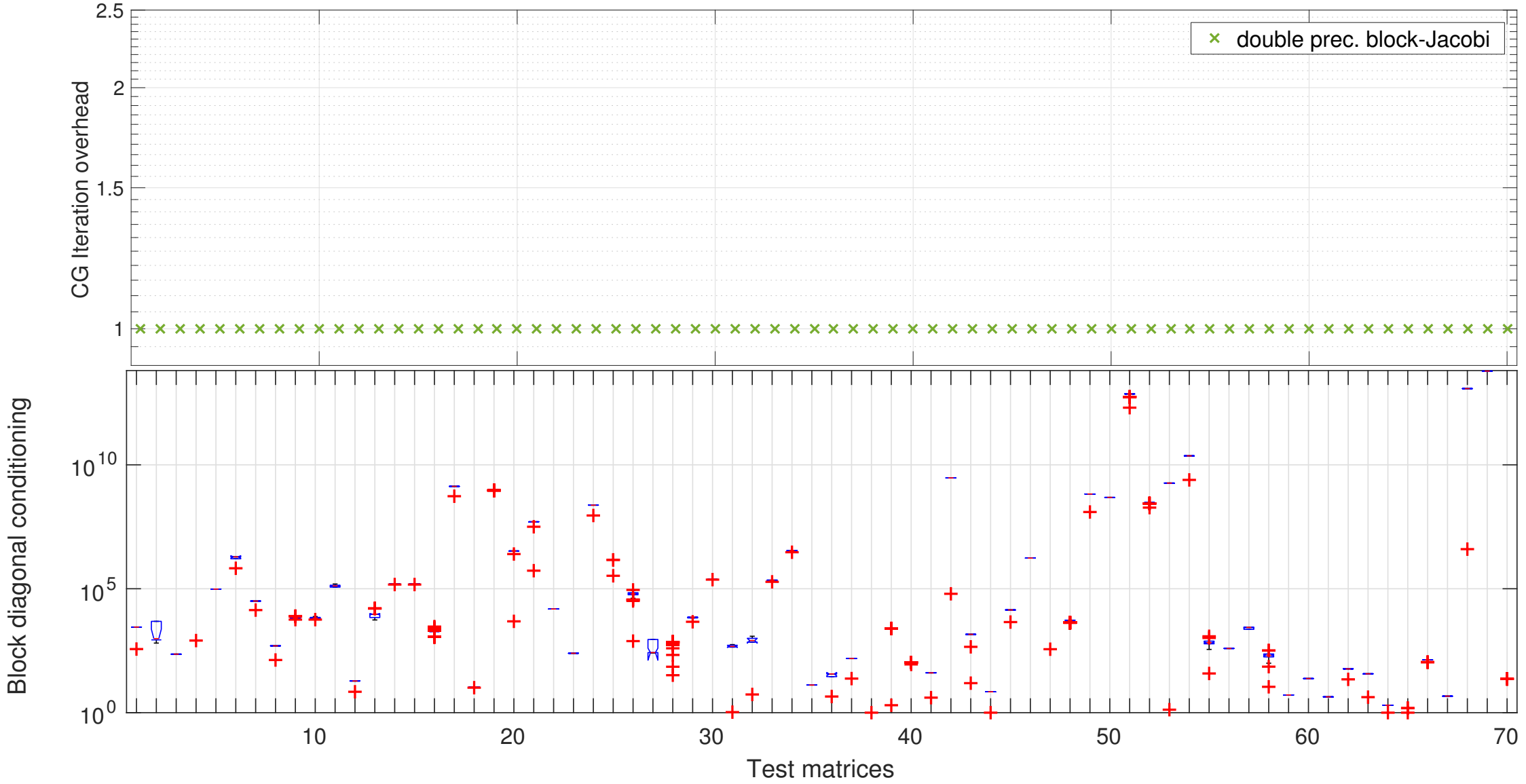
- 70 matrices from the SuiteSparse Matrix Collection
- Use block-size 24 with Super-Variable agglomeration (24 is upper bound for size of blocks)
- Report conditioning of all arising diagonal blocks
- Analyze the impact of storing block-Jacobi in lower precision a top-level Conjugate Gradient solver (CG)



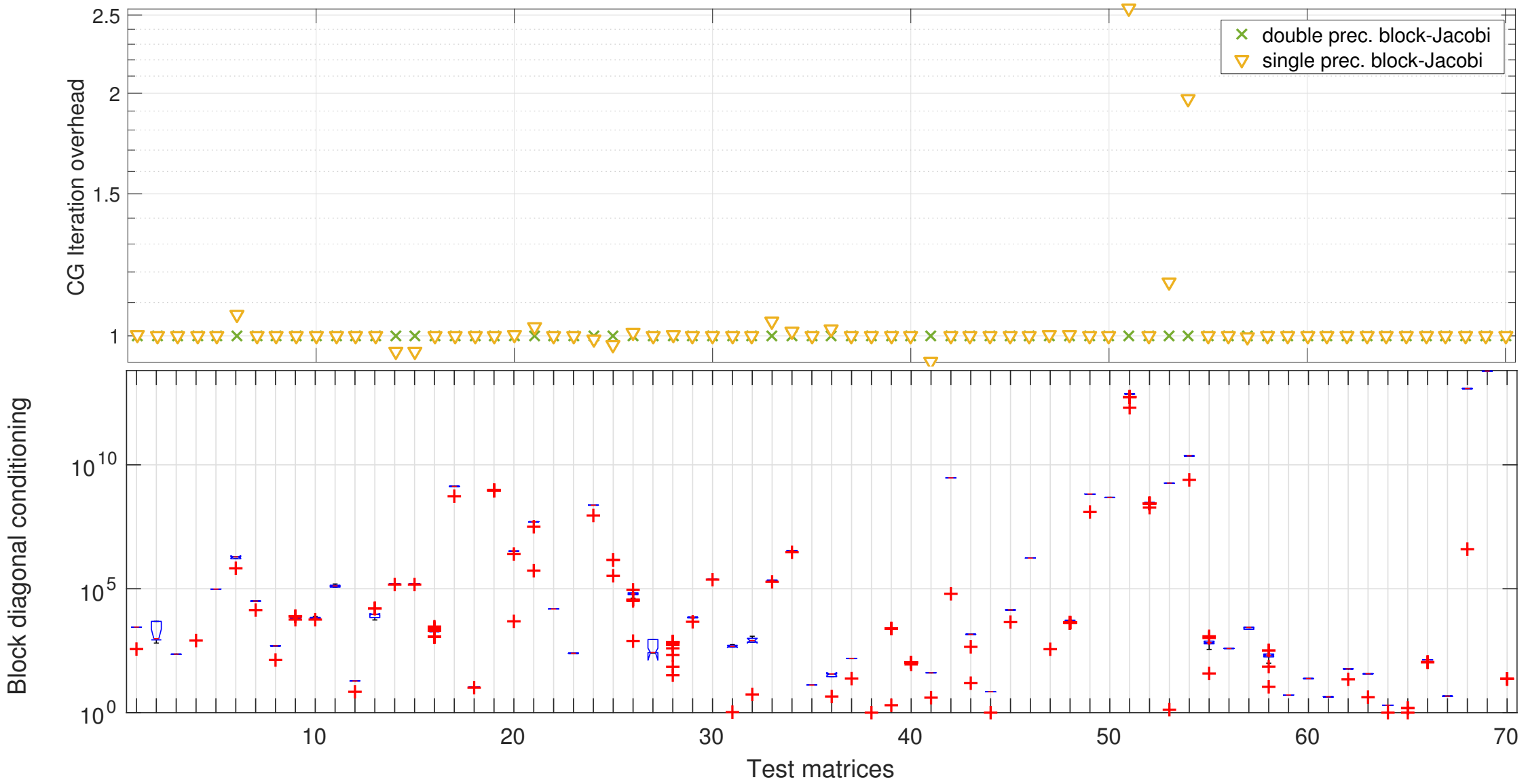
Adaptive Precision Block-Jacobi Preconditioning



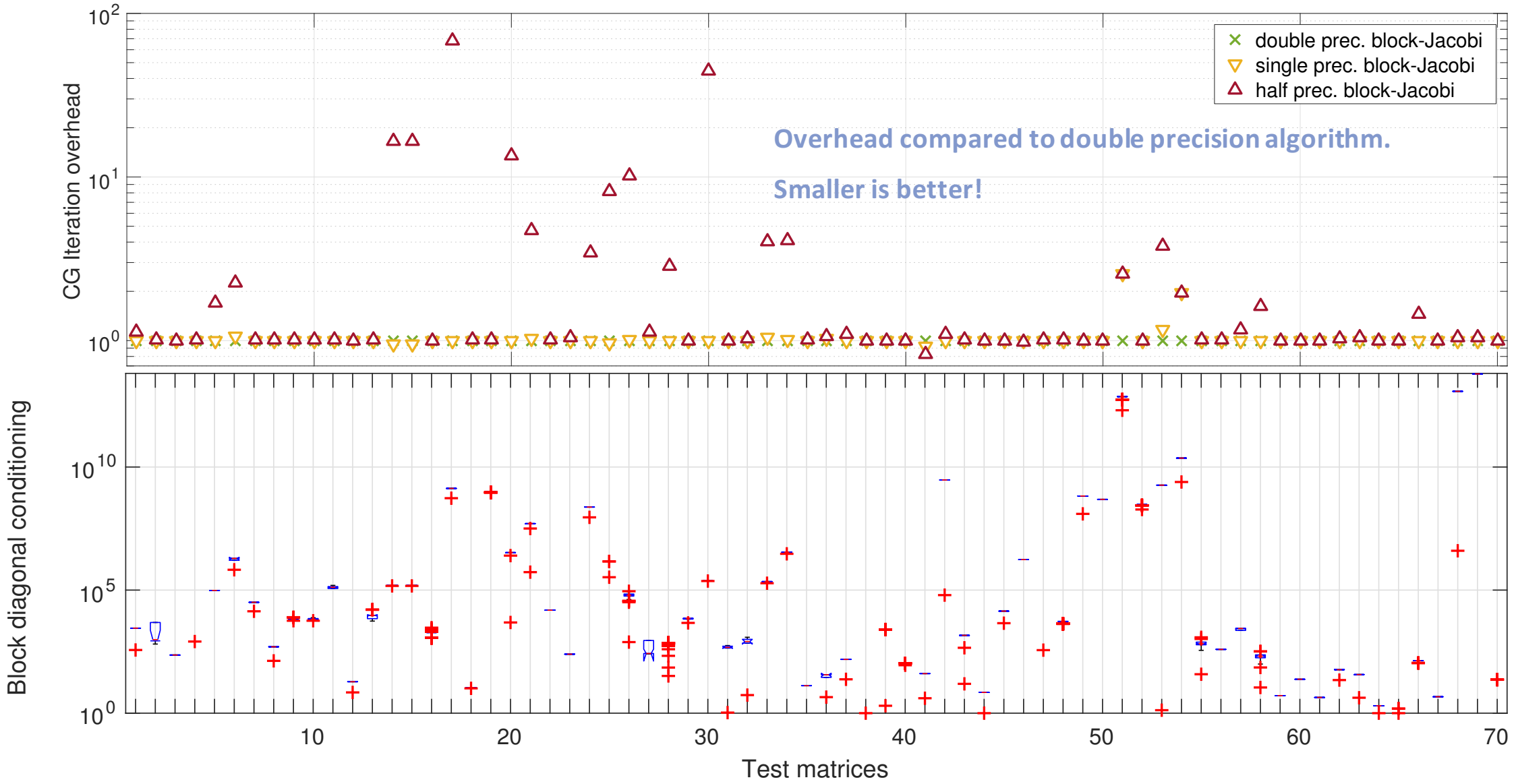
Adaptive Precision Block-Jacobi Preconditioning



Adaptive Precision Block-Jacobi Preconditioning



Adaptive Precision Block-Jacobi Preconditioning

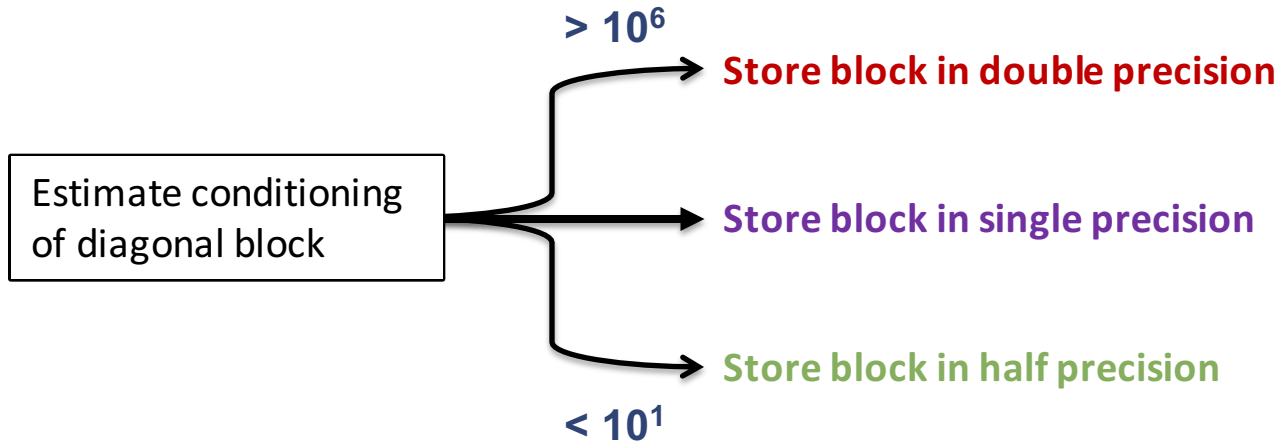


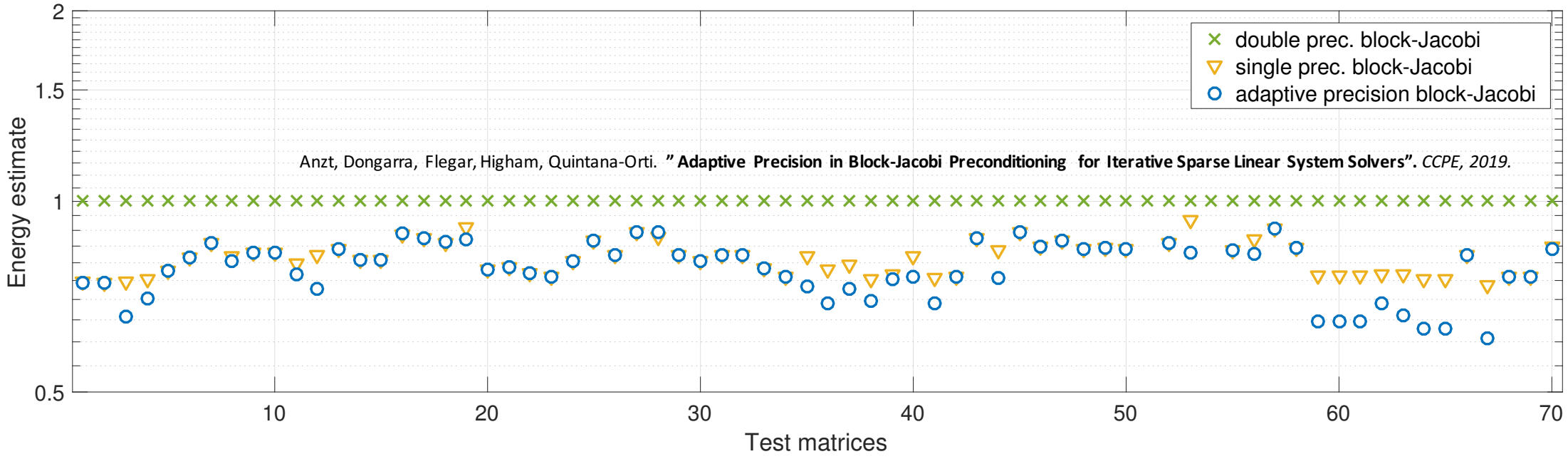
Adaptive Precision Block-Jacobi Preconditioning

Multi-Precision Idea:

- **All computations use double precision!**

- Store distinct blocks in different formats
- Use **single precision as standard** storage format
- Where **necessary:** switch to **double**
- For well-conditioned blocks use **half precision**

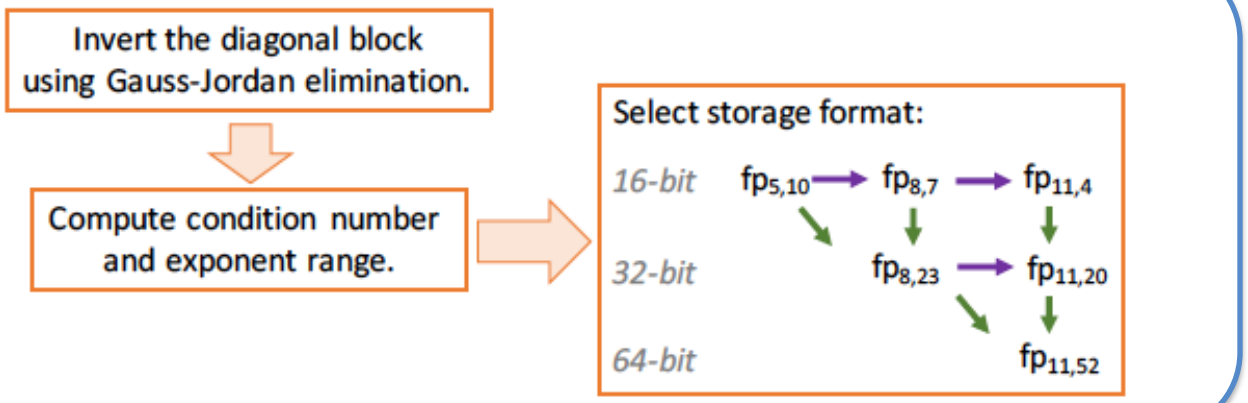




Adaptive Precision Block-Jacobi Preconditioning

Multi-Precision Idea:

- **All computations use double precision!**
- **Depart from the rigid IEEE precision formats!**
- Preserve either 1 or 2 digits accuracy of the inverted diagonal blocks.

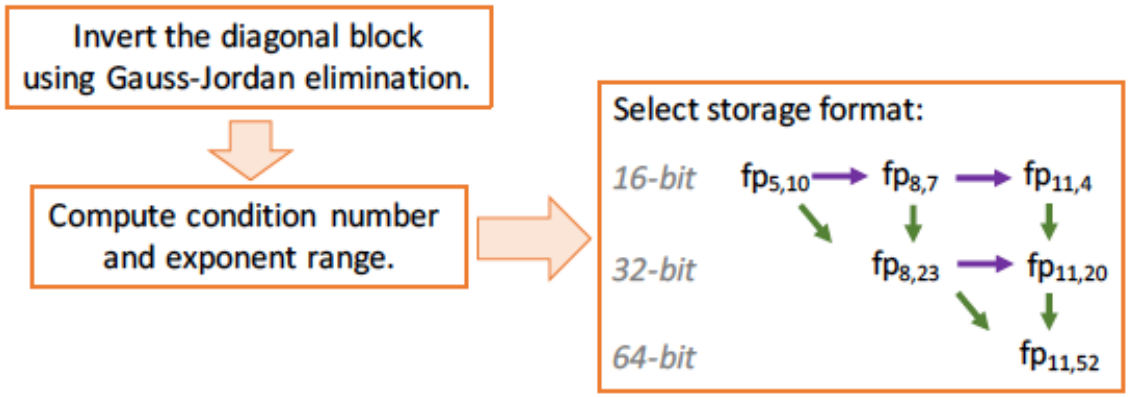


Flegar, Anzt, Quintana-Orti. **"Customized-Precision Block-Jacobi Preconditioning for Krylov Iterative Solvers on Data-Parallel Manycore Processors".** *TOMS, submitted.*

Adaptive Precision Block-Jacobi Preconditioning

Multi-Precision Idea:

- **All computations use double precision!**
- **Depart from the rigid IEEE precision formats!**
- Preserve either **1** or 2 digits accuracy of the inverted diagonal blocks.



- ✓ Regularity preserved;
- ✓ No flexible Krylov solver needed
  (Preconditioner constant operator);
- ✓ Can handle non-spd problems
  (inversion features pivoting);
- ✓ Preconditioner for any iterative preconditionable solver;

- – Speedups / preconditioner quality **problem-dependent**;
- – **Overhead** of the **precision detection**
  (condition number calculation);
- – **Overhead** from storing **precision information**
  (need to additionally store/retrieve flag);

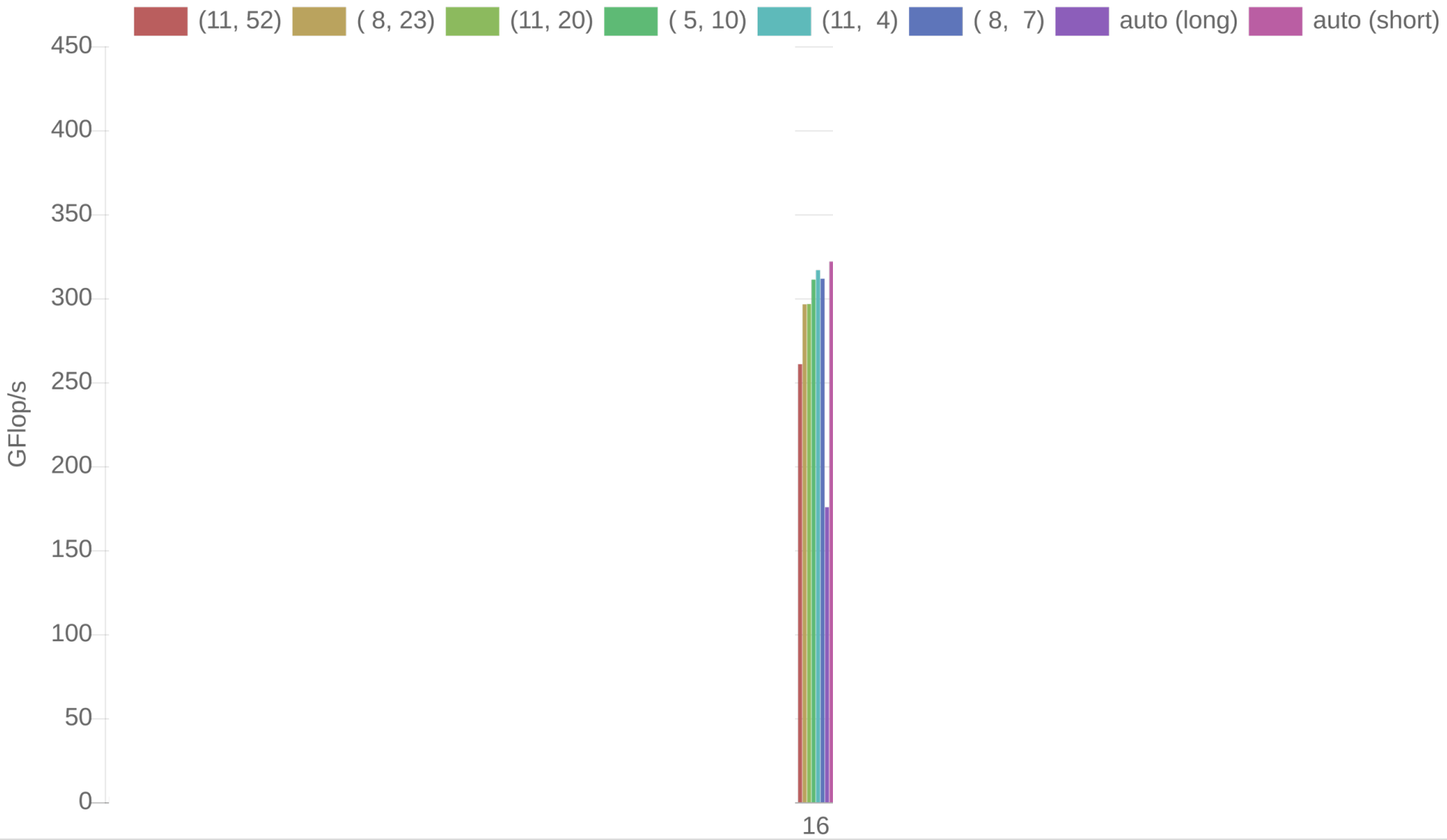
Flegar, Anzt, Quintana-Orti. **"Customized-Precision Block-Jacobi Preconditioning for Krylov Iterative Solvers on Data-Parallel Manycore Processors".** *TOMS, submitted.*

Precision distribution in Adaptive Block-Jacobi

Precision distribution in Adaptive Block-Jacobi

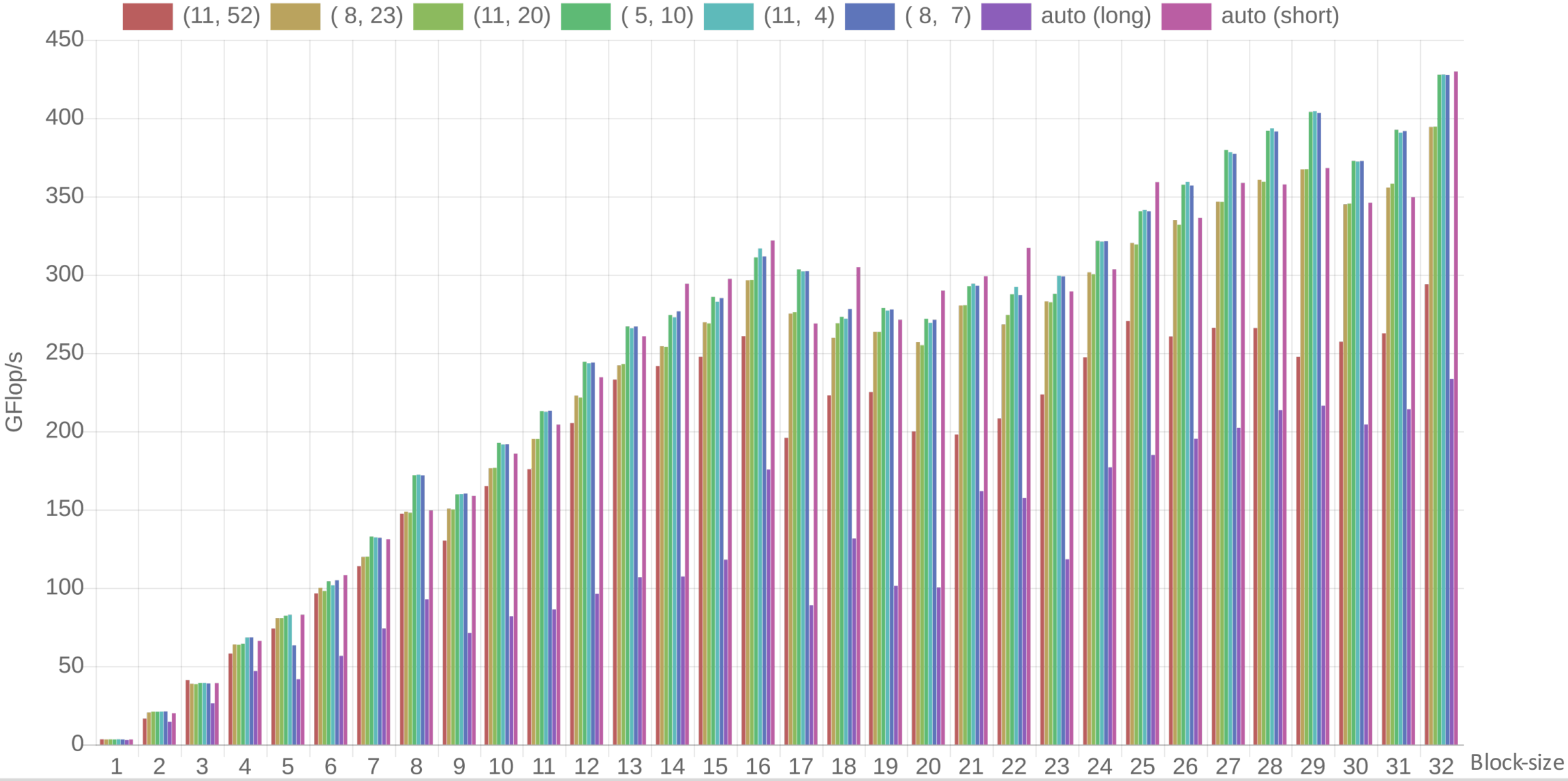
Adaptive Block-Jacobi Generation

NVIDIA V100 GPU



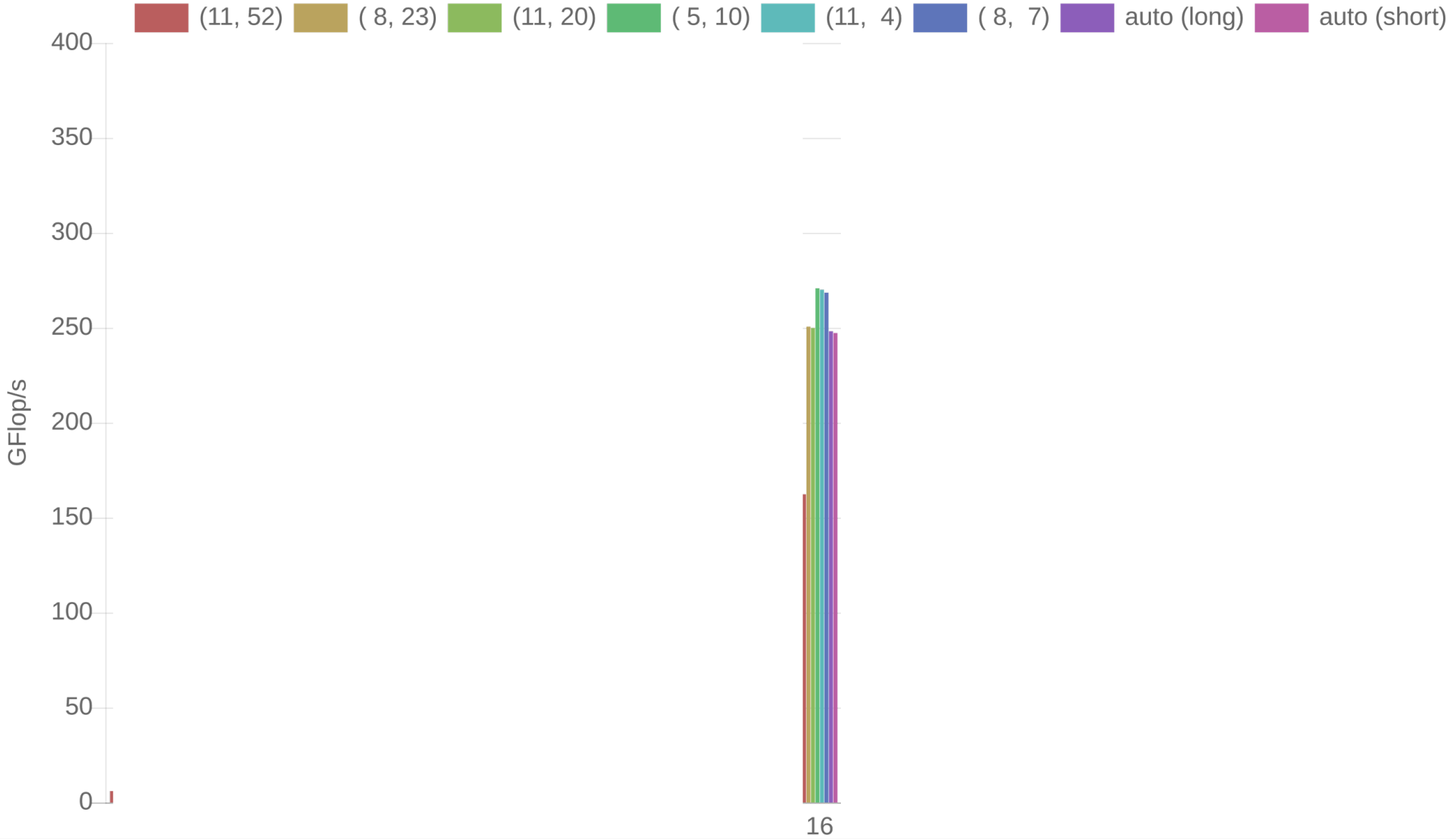
Adaptive Block-Jacobi Generation

NVIDIA V100 GPU



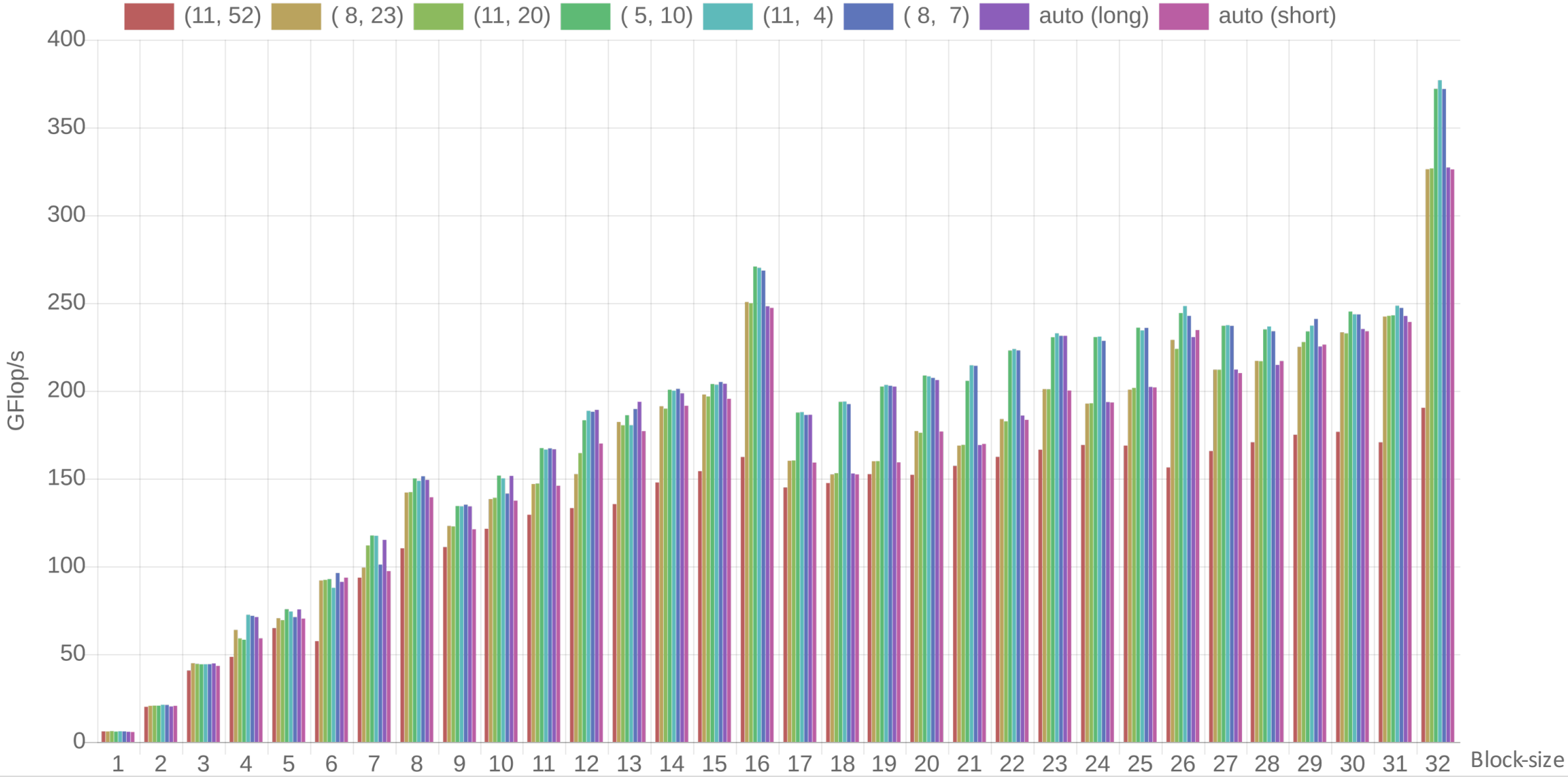
Adaptive Block-Jacobi Application

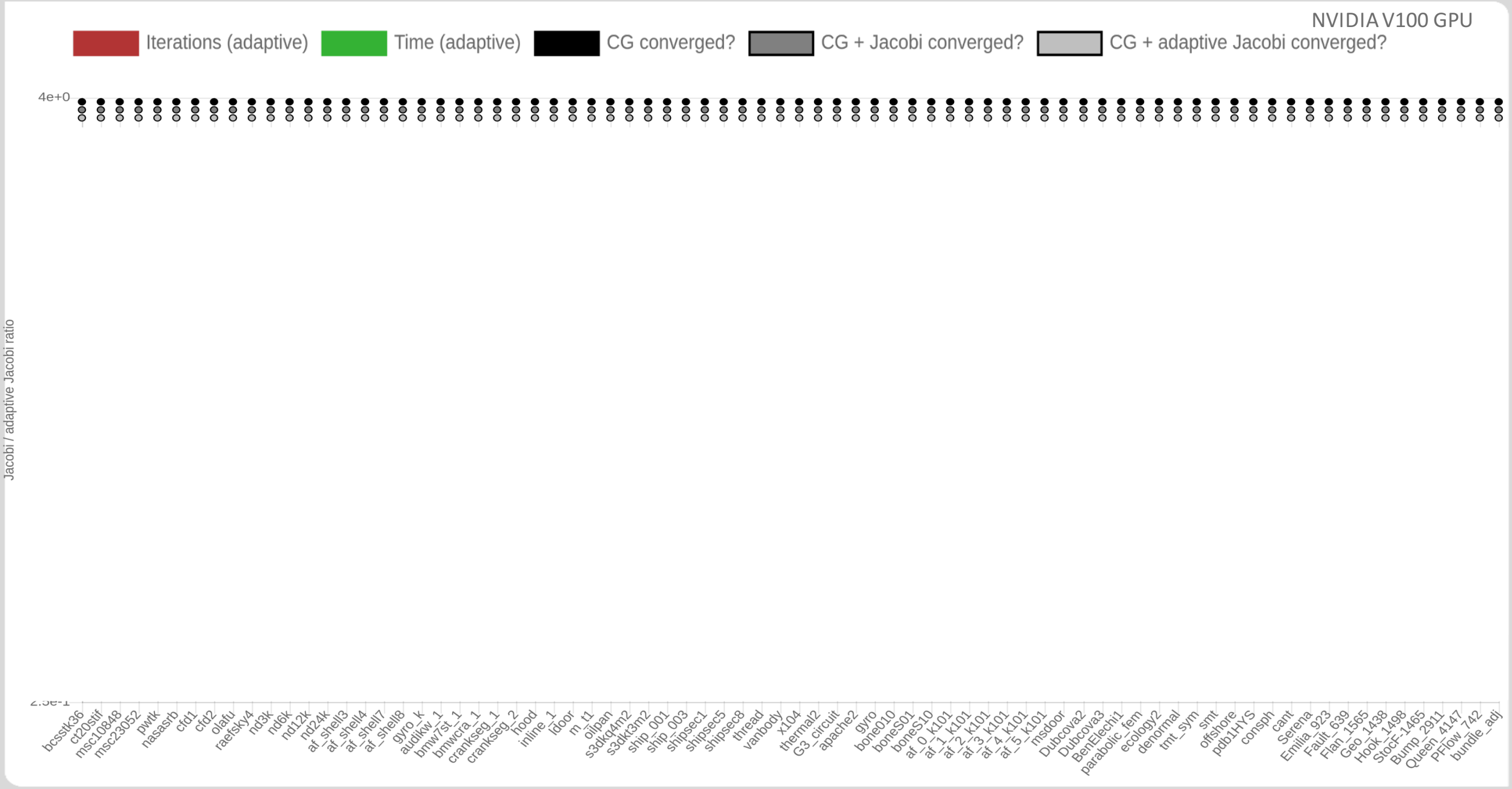
NVIDIA V100 GPU

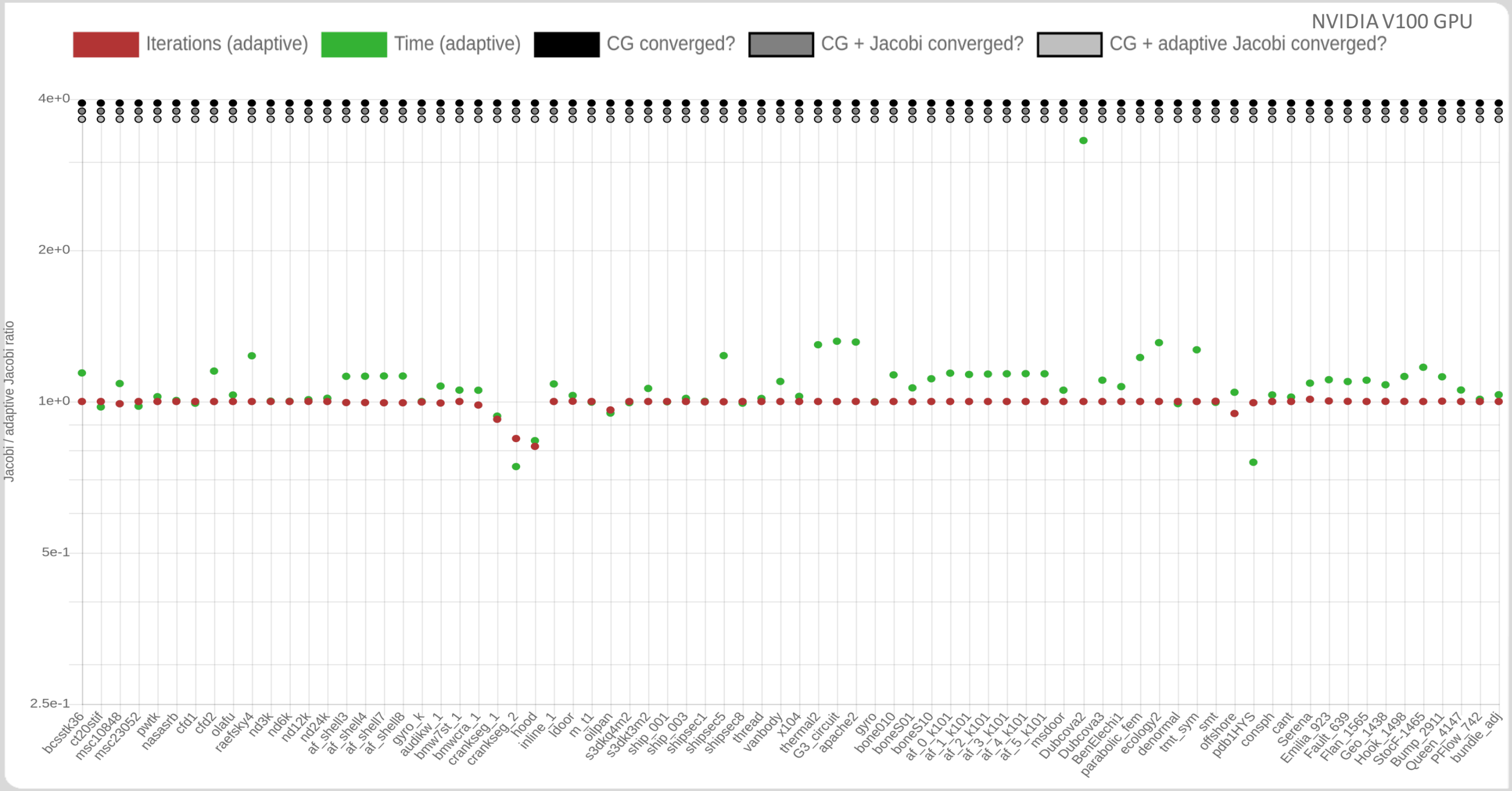


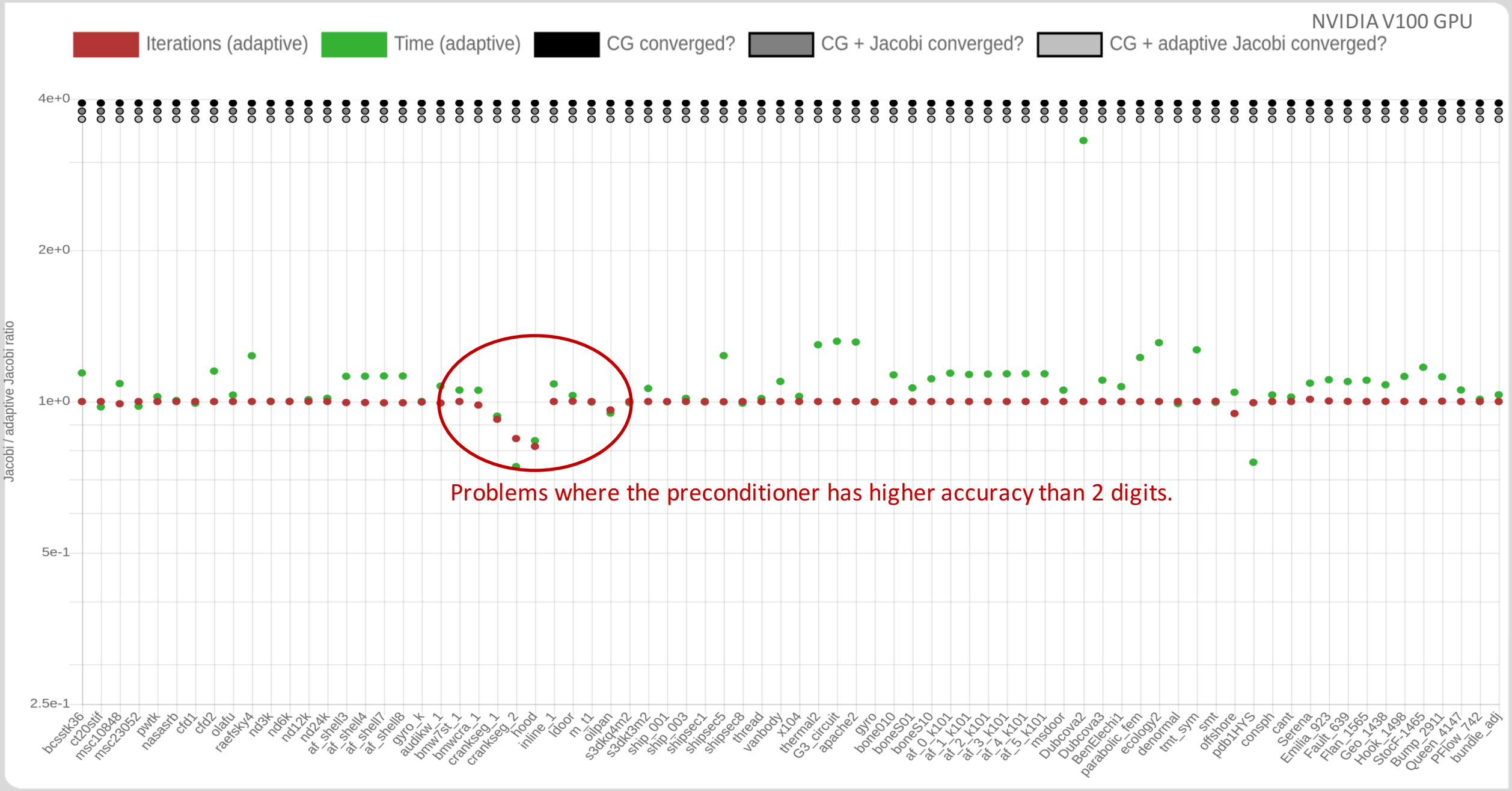
Adaptive Block-Jacobi Application

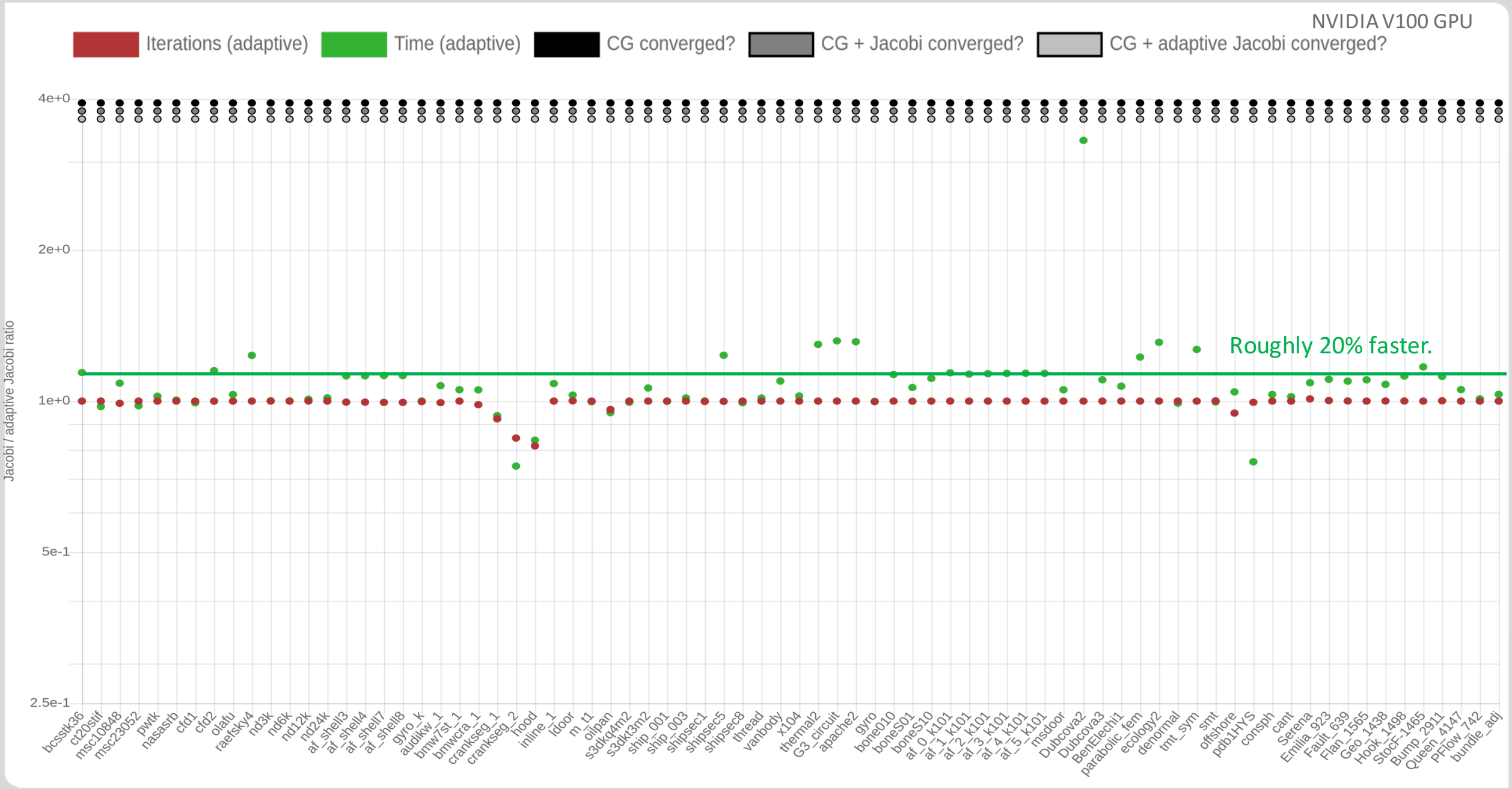
NVIDIA V100 GPU

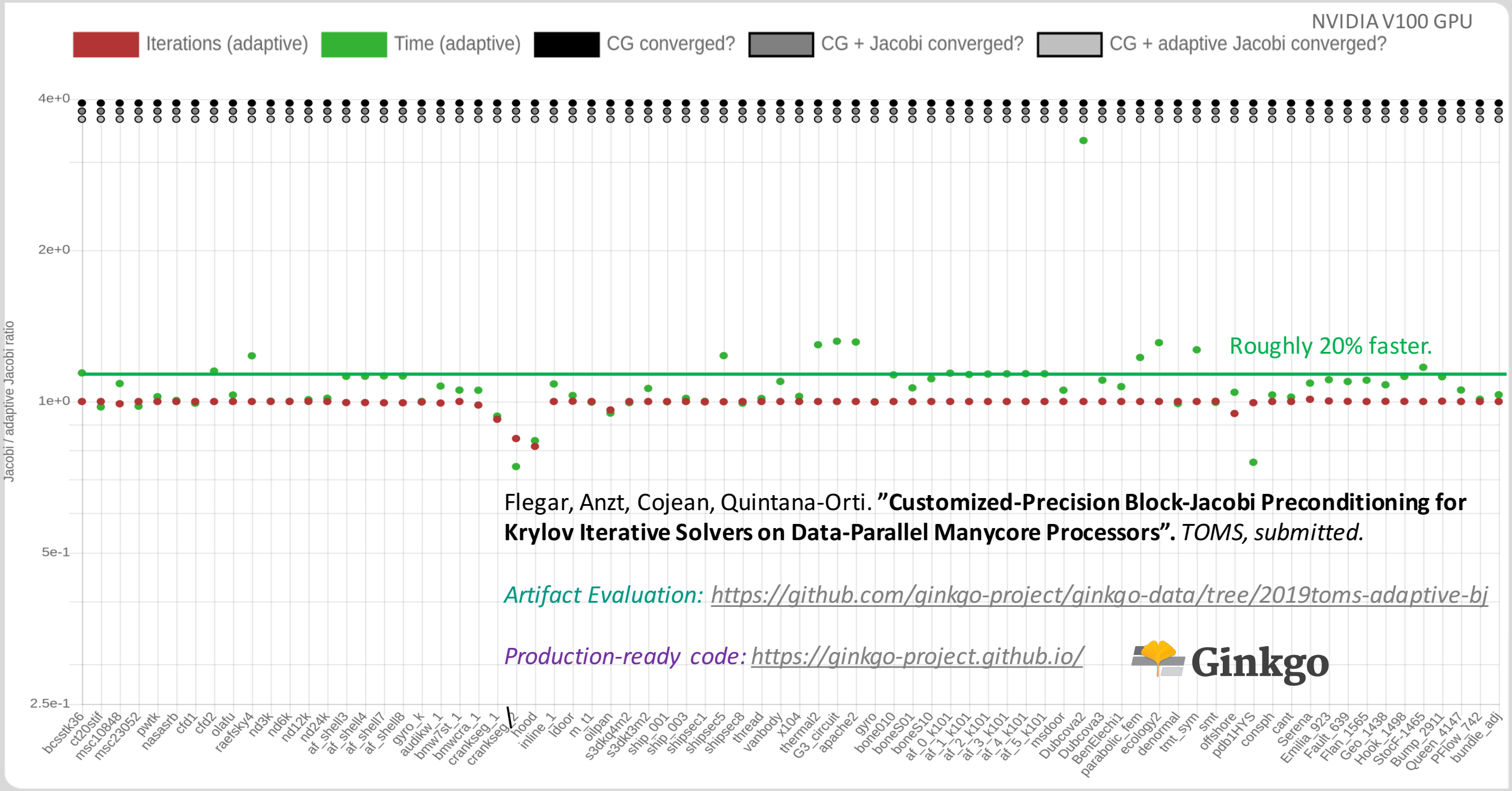



NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
4e+0
Jacobi / adaptive Jacobi ratio
Problem
bcsstk36
ct20stif
msc10848
msc23052
pwtk
nasasrb
cfd1
cfd2
olafu
raefsky4
nd3k
nd6k
nd12k
nd24k
af_shell3
af_shell4
af_shell7
af_shell8
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


NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
Jacobi / adaptive Jacobi ratio
4e+0
2e+0
1e+0
5e-1
2.5e-1
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
Problem


NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
Jacobi / adaptive Jacobi ratio
4e+0
2e+0
1e+0
5e-1
2.5e-1
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
boneS10
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


NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
Jacobi / adaptive Jacobi ratio
4e+0
2e+0
1e+0
5e-1
2.5e-1
Roughly 20% faster.
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
boneS10
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, submitted.*

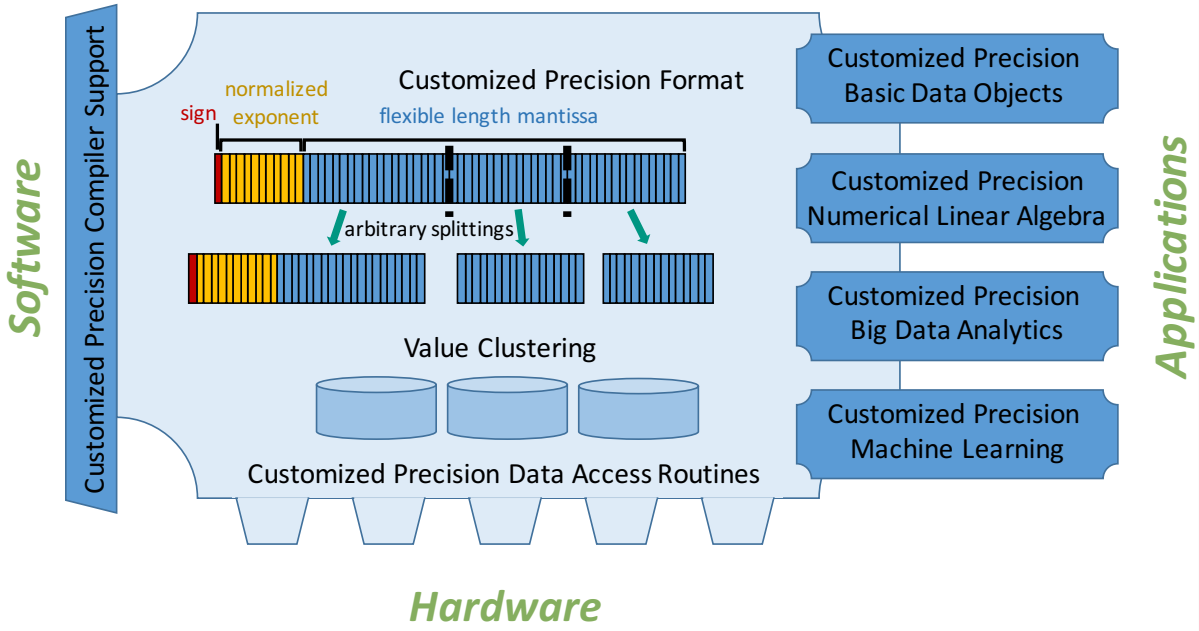
Artifact Evaluation: https://github.com/ginkgo-project/ginkgo-data/tree/2019toms-adaptive-bj

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

Ginkgo

Summary and next steps

- **Decouple arithmetic precision from memory precision.**
- Using **customized precisions** for memory operations.
- Speedup of up to 1.3x for adaptive precision block-Jacobi preconditioning.
- Creating a **Modular Precision Ecosystem** inside Ginkgo.
  https://github.com/ginkgo-project/ginkgo



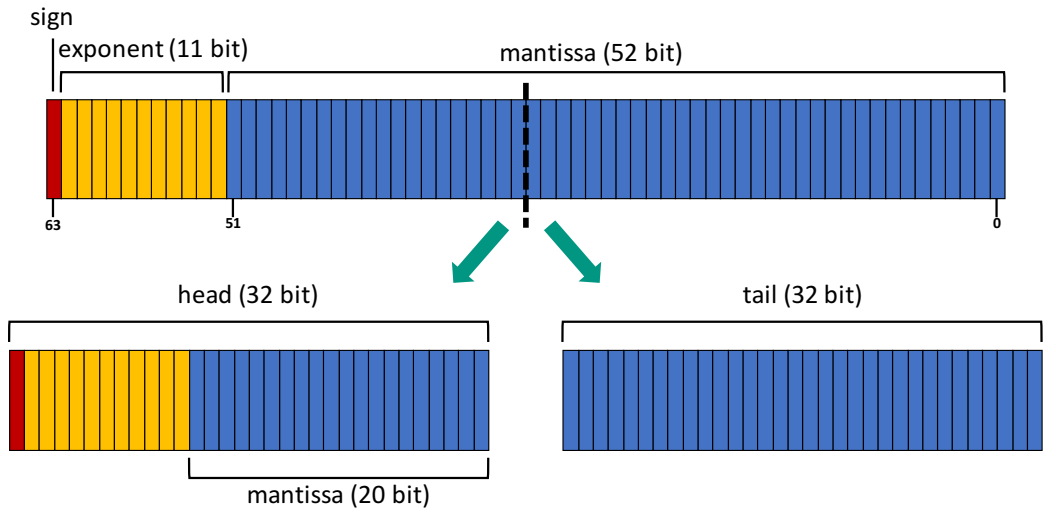
EXASCALE COMPUTING PROJECT

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

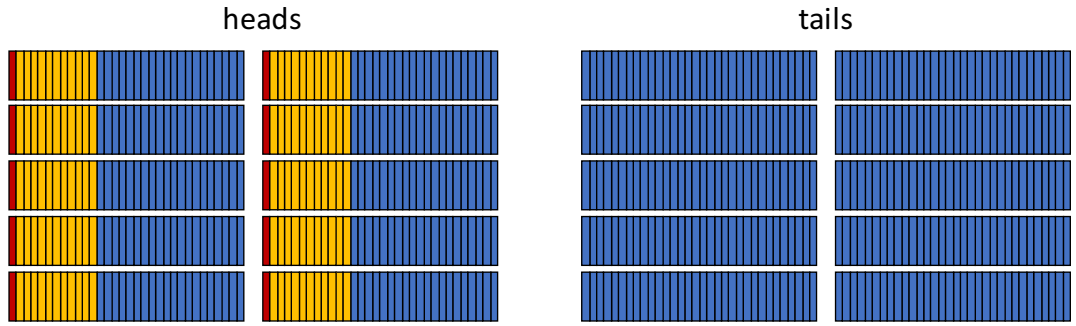
This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration and the Helmholtz Impuls und VernetzungsfondVH-NG-1241.

Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)

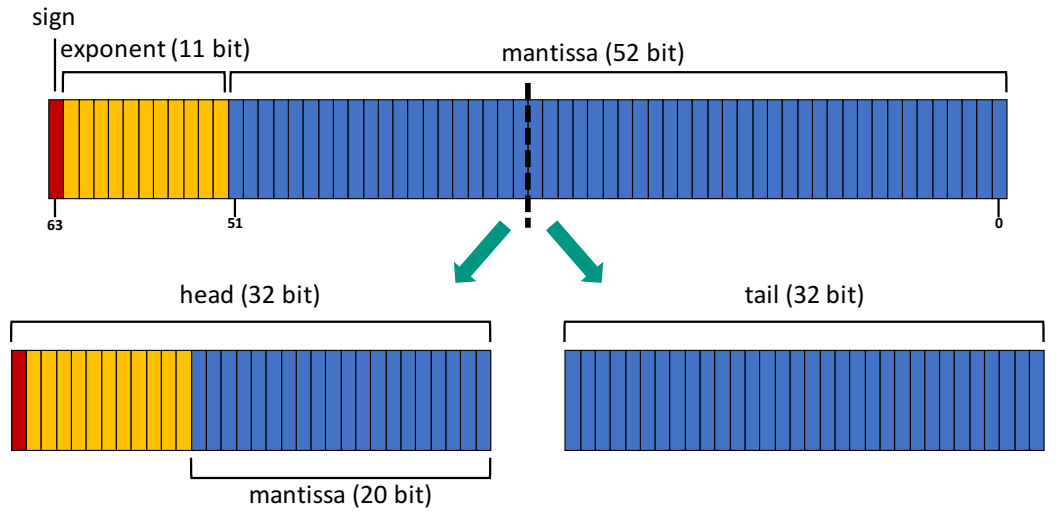


- *Special "conversion" routines to double precision.*
- *Significand much shorter than IEEE single/half precision.*
- *No under- / overflow.*

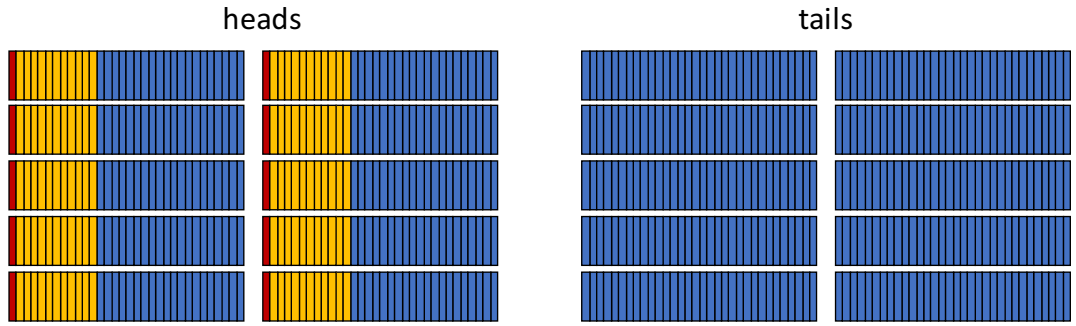


Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.

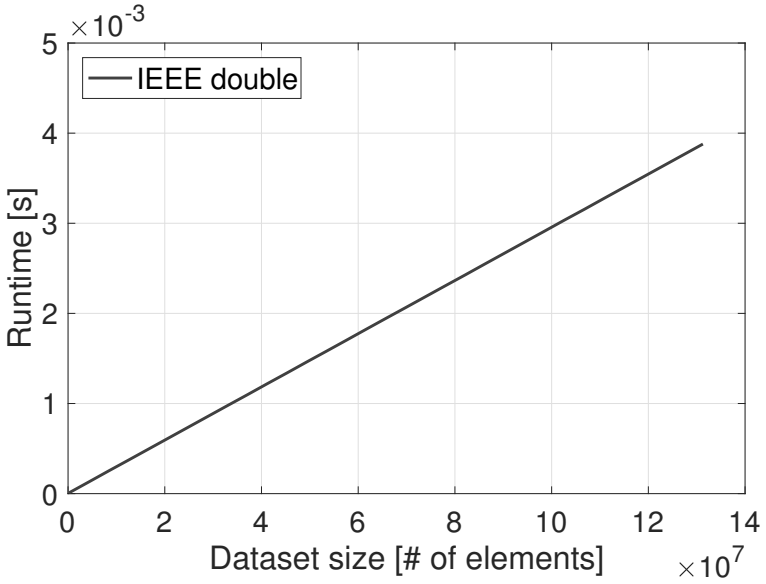


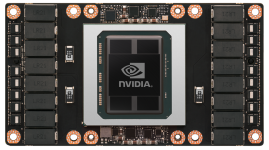
- *Special "conversion" routines to double precision.*
- *Significand much shorter than IEEE single/half precision.*
- *No under- / overflow.*



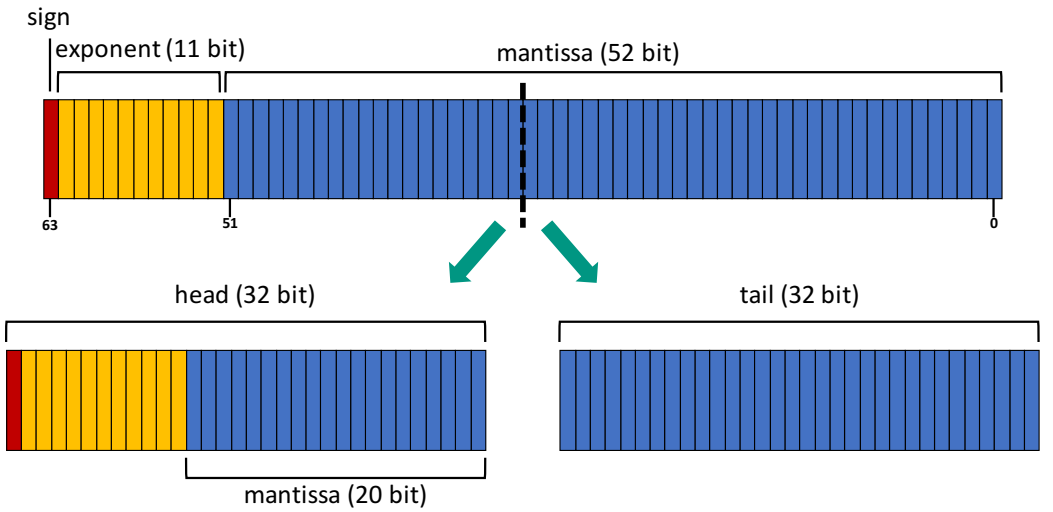
Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*

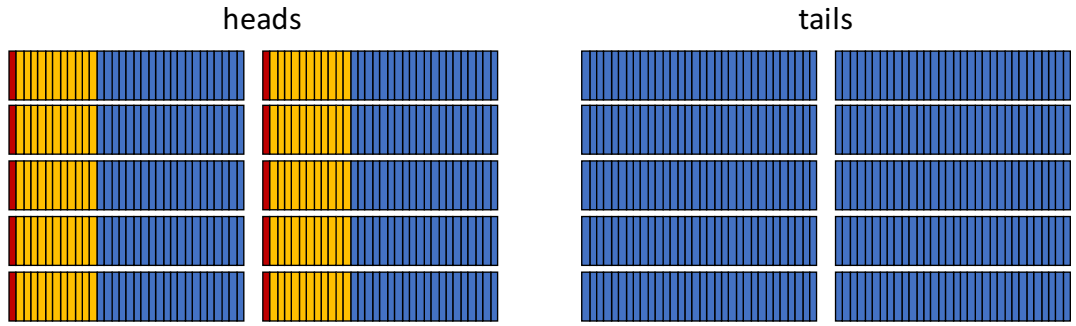




NVIDIA P100 "Pascal"
5.3 TFLOP/s DP
16GB RAM @720 GB/s

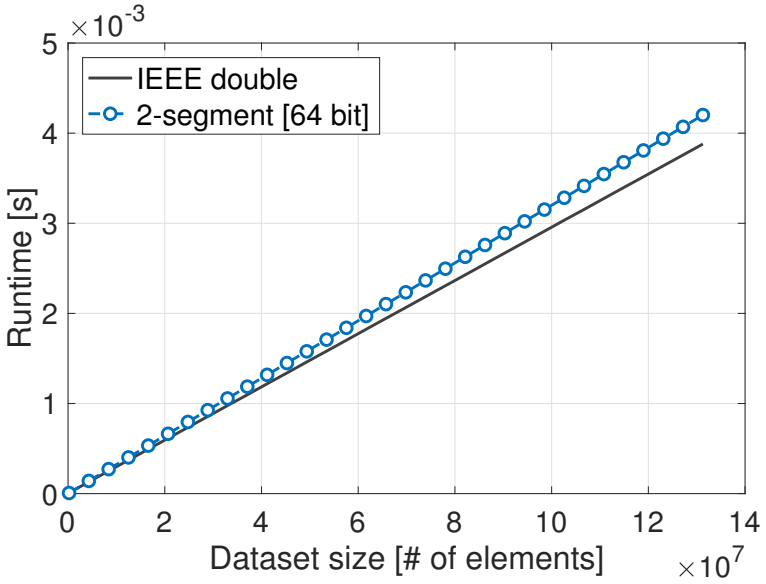


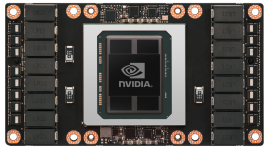
- *Special "conversion" routines to double precision.*
- *Significand much shorter than IEEE single/half precision.*
- *No under- / overflow.*



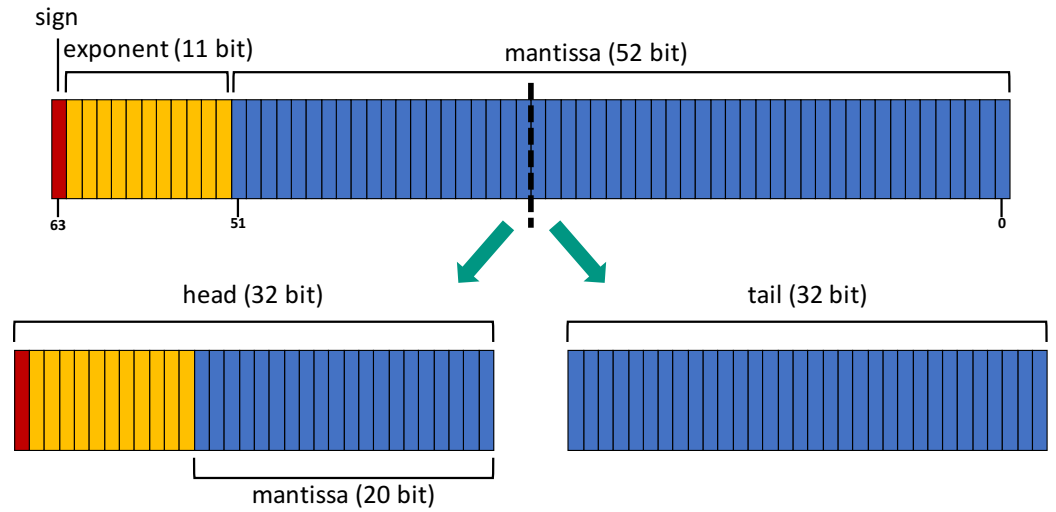
Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*

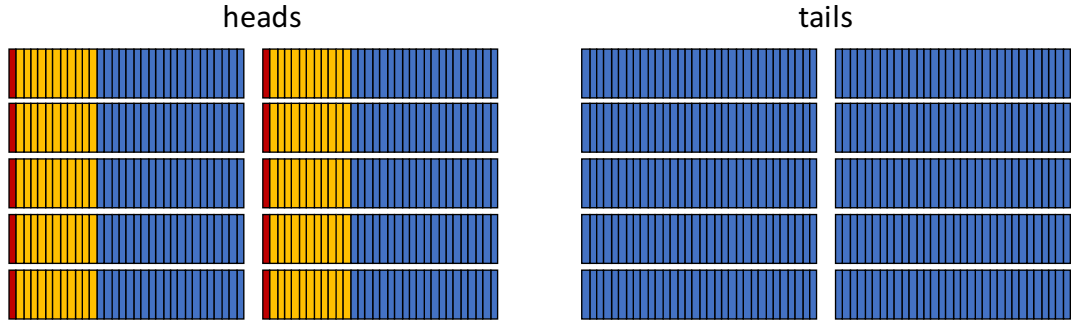




NVIDIA P100 "Pascal"
5.3 TFLOP/s DP
16GB RAM @720 GB/s

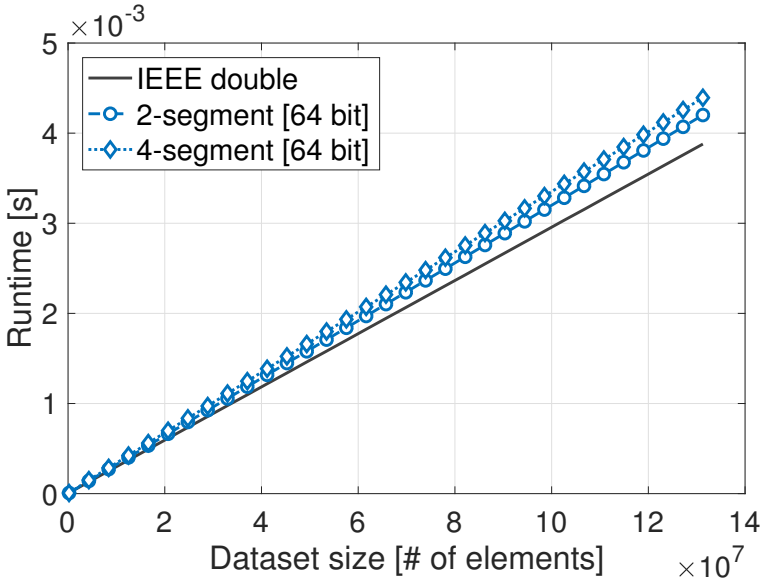


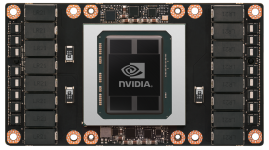
- *Special "conversion" routines to double precision.*
- *Significand much shorter than IEEE single/half precision.*
- *No under- / overflow.*



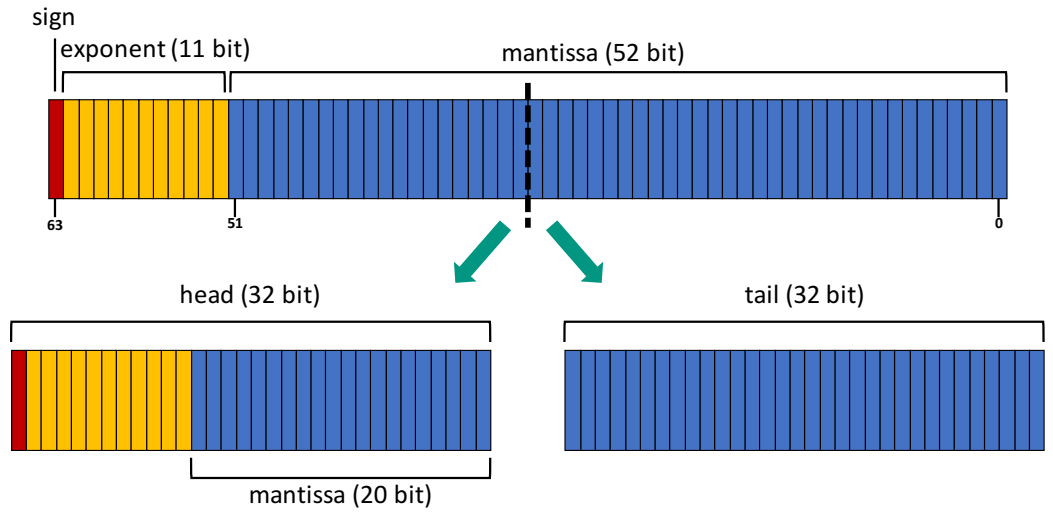
Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*

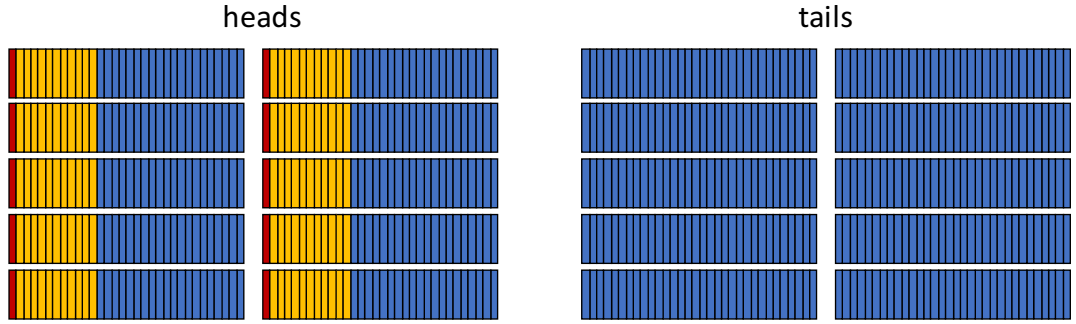




NVIDIA P100 "Pascal"
5.3 TFLOP/s DP
16GB RAM @720 GB/s

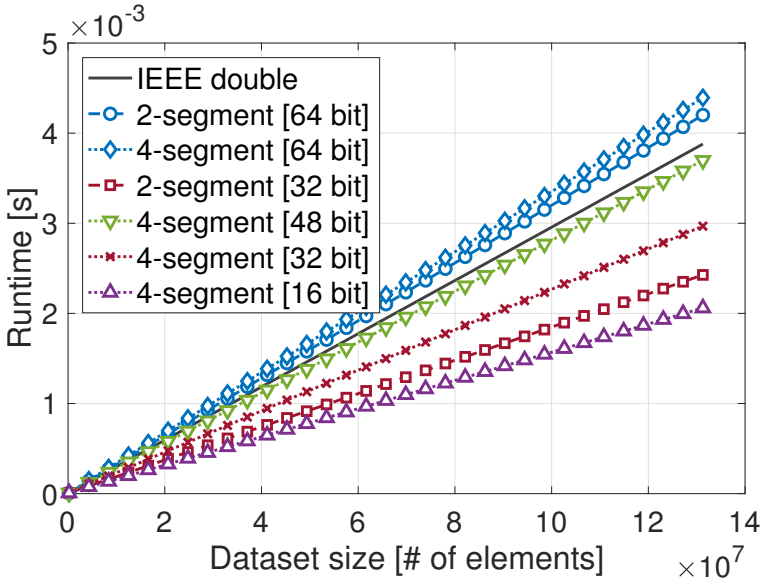


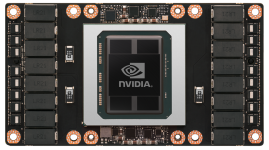
- *Special "conversion" routines to double precision.*
- *Significand much shorter than IEEE single/half precision.*
- *No under- / overflow.*



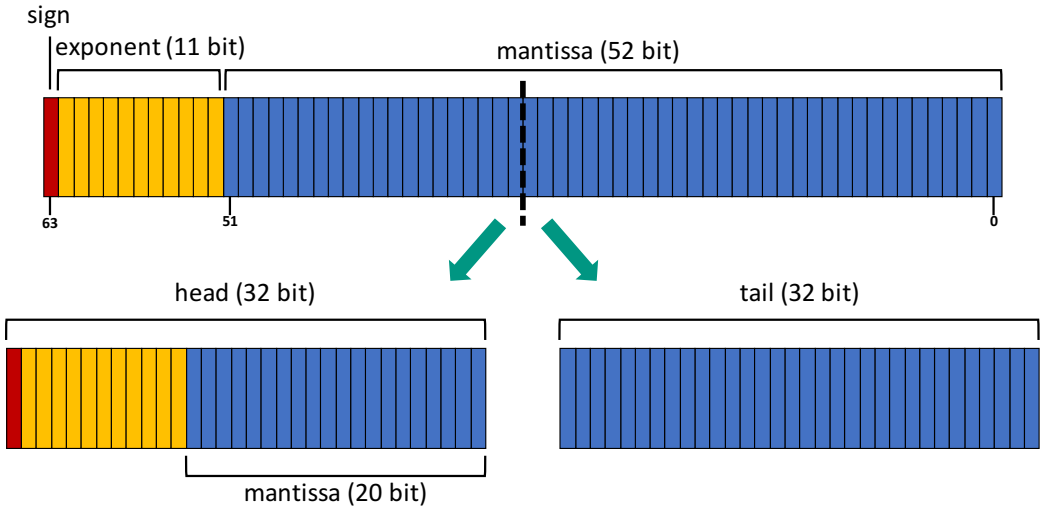
Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*

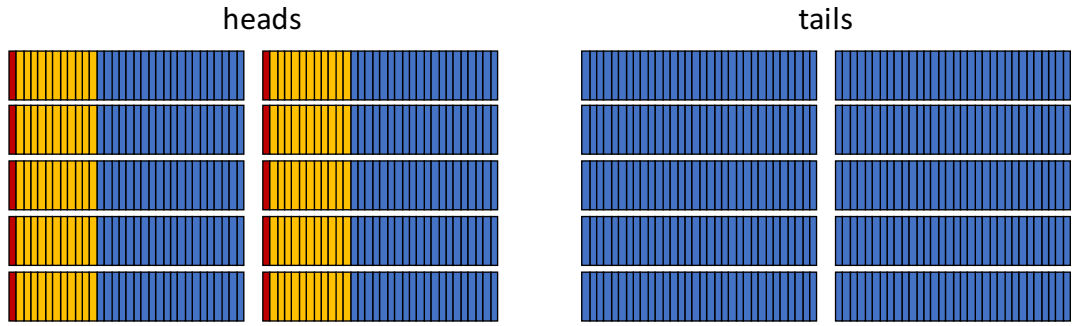




NVIDIA P100 "Pascal"
5.3 TFLOP/s DP
16GB RAM @720 GB/s



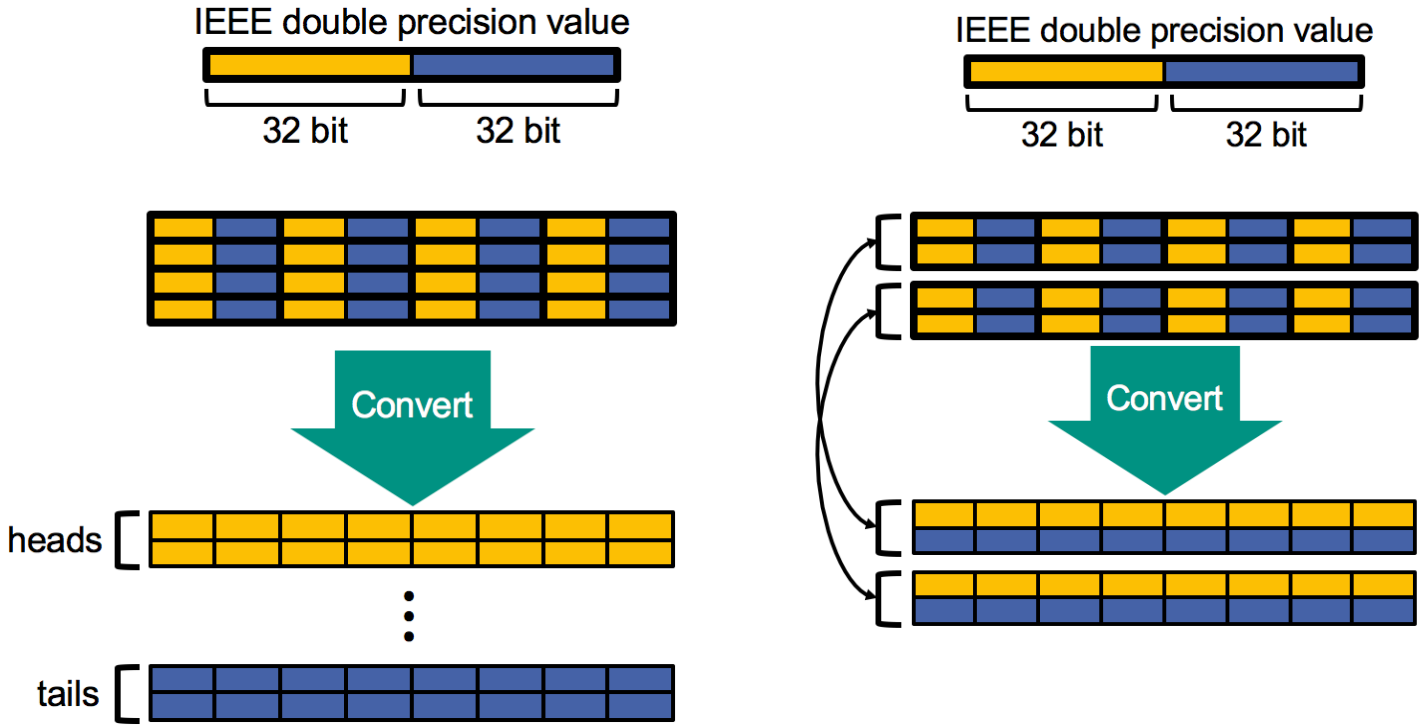
- *Special "conversion" routines to double precision.*
- *Significand much shorter than IEEE single/half precision.*
- *No under- / overflow.*



Significand Segmentation

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*

- *Special "conversion" routines to double precision.*
- *Mantissa much shorter than IEEE single/half precision.*
- *No under- / overflow.*



Customized Precision Adaptive Jacobi

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*
- Need precision-aware algorithms, e.g. Adaptive Precision Jacobi[1].
  - *Use low precision data reads whenever accuracy allows for it.*

Adaptive precision solvers for sparse linear systems
Anzt, Dongarra, Quintana-Orti, Published in SC-Workshops in 2015.

- *Start iterations with reading low precision.*
- *Increase accuracy over iteration process adapted to requirements.*
- *Adapt the accuracy on a component level.*

[1]Anzt, Dongarra, Quintana-Orti. **"Adaptive precision solvers for sparse linear systems".** *In: Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing. (2015).*

Customized Precision Adaptive Jacobi

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*
- Need precision-aware algorithms, e.g. Adaptive Precision Jacobi[1].
  - *Use low precision data reads whenever accuracy allows for it.*

Adaptive precision solvers for sparse linear systems
Anzt, Dongarra, Quintana-Orti, Published in SC-Workshops in 2015.

- *Start iterations with reading low precision.*
- *Increase accuracy over iteration process adapted to requirements.*
- *Adapt the accuracy on a component level.*

Jacobi relaxation

Iterative method for solving $Ax = b$:

$$x^{k+1} = D^{-1}\left(b - (A - D)x^k\right)$$

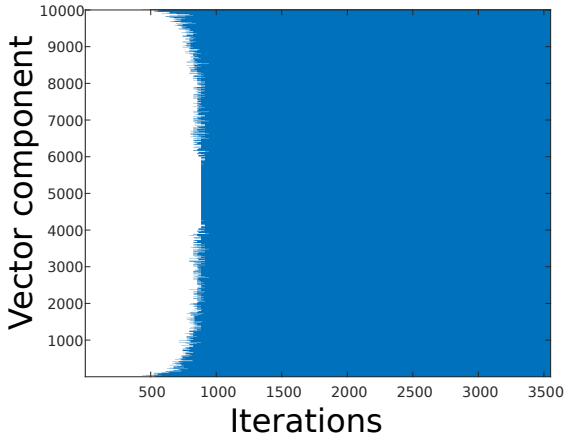
$$x+ = -D^{-1}Ax + c$$

[1]Anzt, Dongarra, Quintana-Orti. **"Adaptive precision solvers for sparse linear systems".** *In: Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing. (2015).*

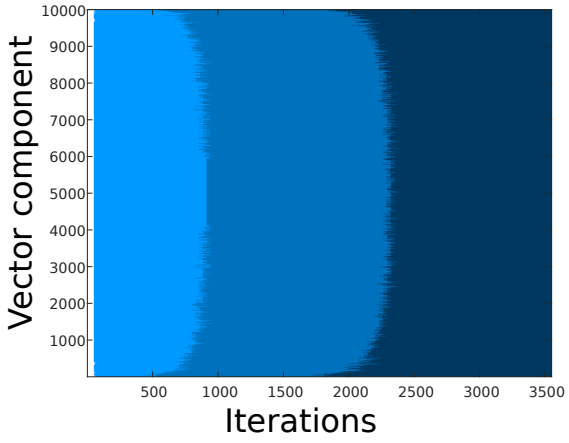
Customized Precision Adaptive Jacobi

- Split the IEEE double precision format into segments.
  (2-segment customized precision, 4-segment customized precision…)
- For coalesced data access, interleave data in memory.
- *Data can be accessed much faster if low precision is acceptable.*
- Need precision-aware algorithms, e.g. Adaptive Precision Jacobi[1].
  - *Use low precision data reads whenever accuracy allows for it.*

2-segment



4-segment

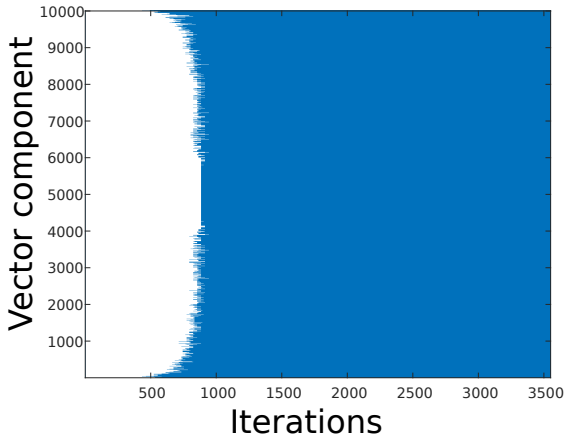


[1]Anzt, Dongarra, Quintana-Orti. **"Adaptive precision solvers for sparse linear systems".** *In: Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing. (2015).*
[2]Grützmacher, Anzt. ***"A Customized Precision Format for decoupling Arithmetic Format and Storage Format".*** *Submitted to HeteroPar workshop 2018.*

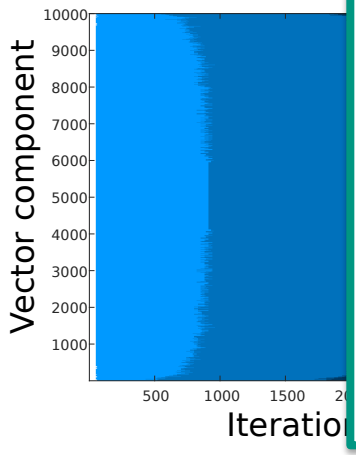
Customized Precision Adaptive Jacobi

- Split the IEEE double precision format into segm
  (2-segment customized precision, 4-segment cus
- For coalesced data access, interleave data in mer
- *Data can be accessed much faster if low precisio*
- Need precision-aware algorithms, e.g. Adaptive
  - *Use low precision data reads whenever a*

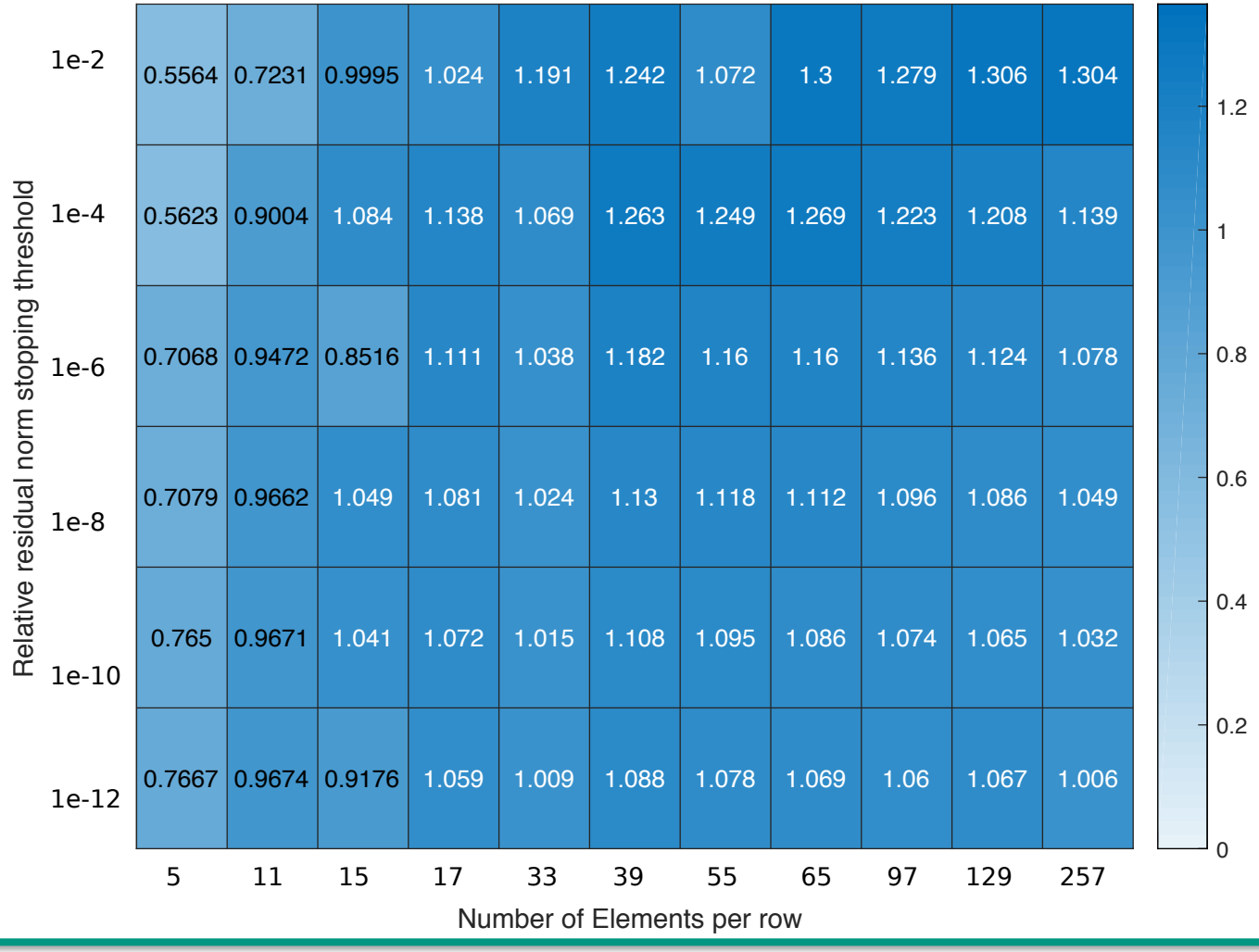
2-segment



4-segm



Speedup of adaptive precision Jacobi in 2-segment Customized Precision Format



[1]Anzt, Dongarra, Quintana-Orti. **"Adaptive precision solvers for sparse linear systems".** *In: Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing. (2015).*
[2]Grützmacher, Anzt. ***"A Customized Precision Format for decoupling Arithmetic Format and Storage Format".*** *Submitted to HeteroPar workshop 2018.*

Customized Precision PageRank

- Algorithm ranking web-pages based on “importance.”
- Underlying algorithm is a eigenvalue solver based on the **Power Iteration.**
- **Fixed-point iteration type.**
- **Sparse matrix vector product** is the central and most expensive building block.

PageRank(A, ε, δ) :

$\mathbb{S}$ = All indices where A has a empty row

$p^{\{0\}} = \frac{1}{n} \cdot e$

$k = 0$

repeat

$k = k + 1$

$s = \sum_{\{i \in \mathbb{S}\}} p_i^{\{k-1\}}$

$p^{\{k\}} = \delta \cdot A^T \cdot p^{\{k-1\}} + \frac{(1-\delta)}{n} \cdot e + \frac{s}{n} \cdot e$

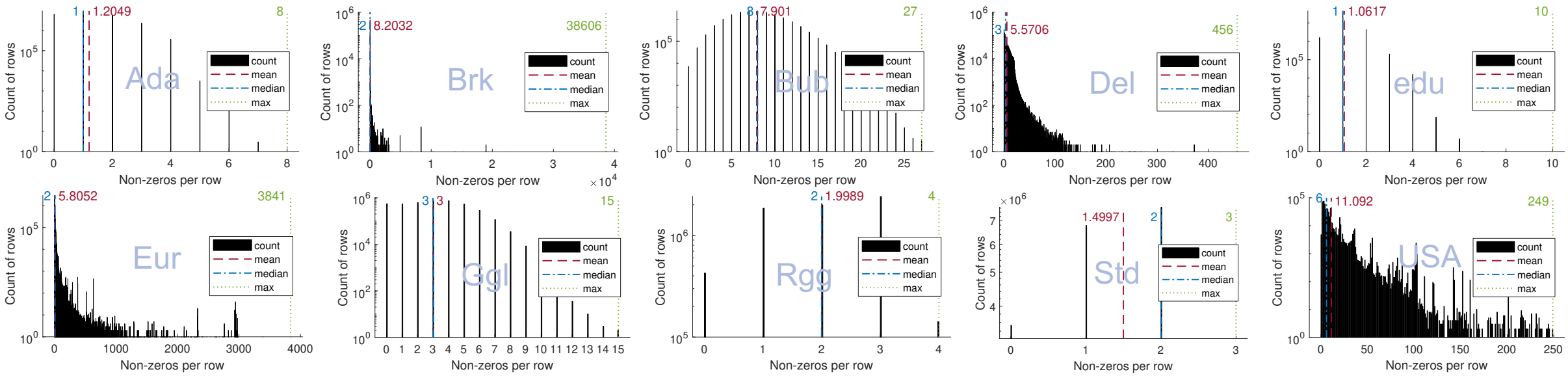
$\gamma = \left\| p^{\{k\}} - p^{\{k-1\}} \right\|_1$

until $(\gamma < \varepsilon)$

return $p^{\{k\}}$

The PageRank citation ranking: Bringing order to the Web L. Page, S. Brin, R. Motwani, and T. Winograd. Proceedings of the 7th International World Wide Web Conference, page 161--172. Brisbane, Australia, (1998)

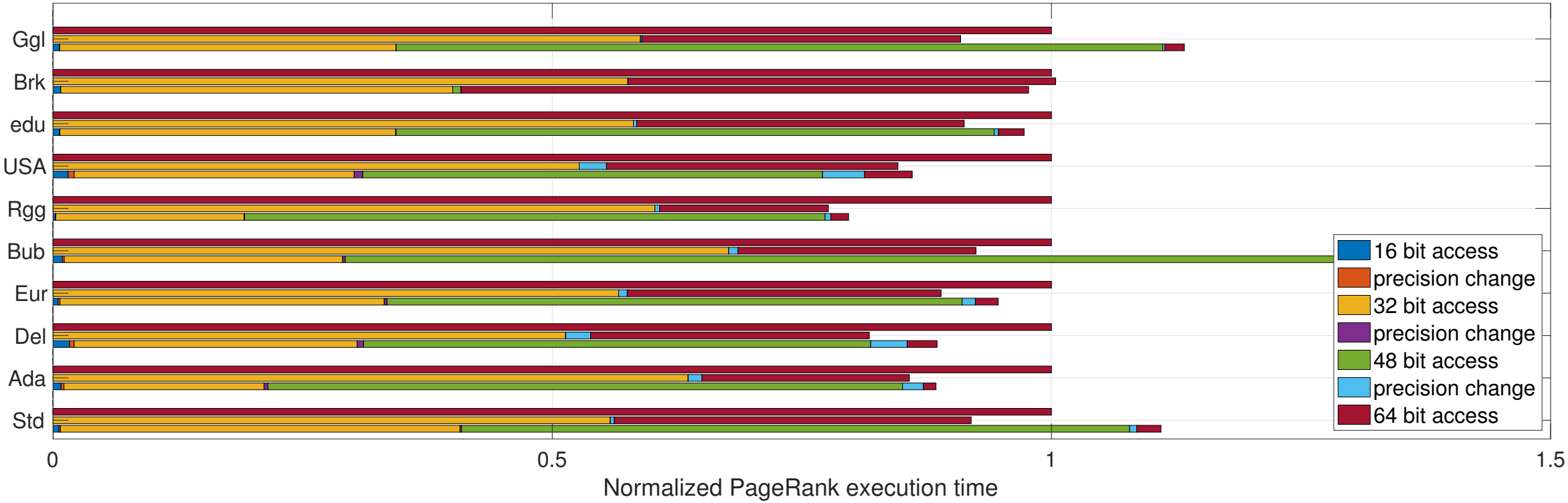
Customized Precision PageRank



10 test problems from Suite Sparse (community networks, etc.)

Customized Precision PageRank

Trace of 2-segment and 4-segment customized precision PageRank (NVIDIA V100 GPU, Relative residual stopping criterion 1e-12).

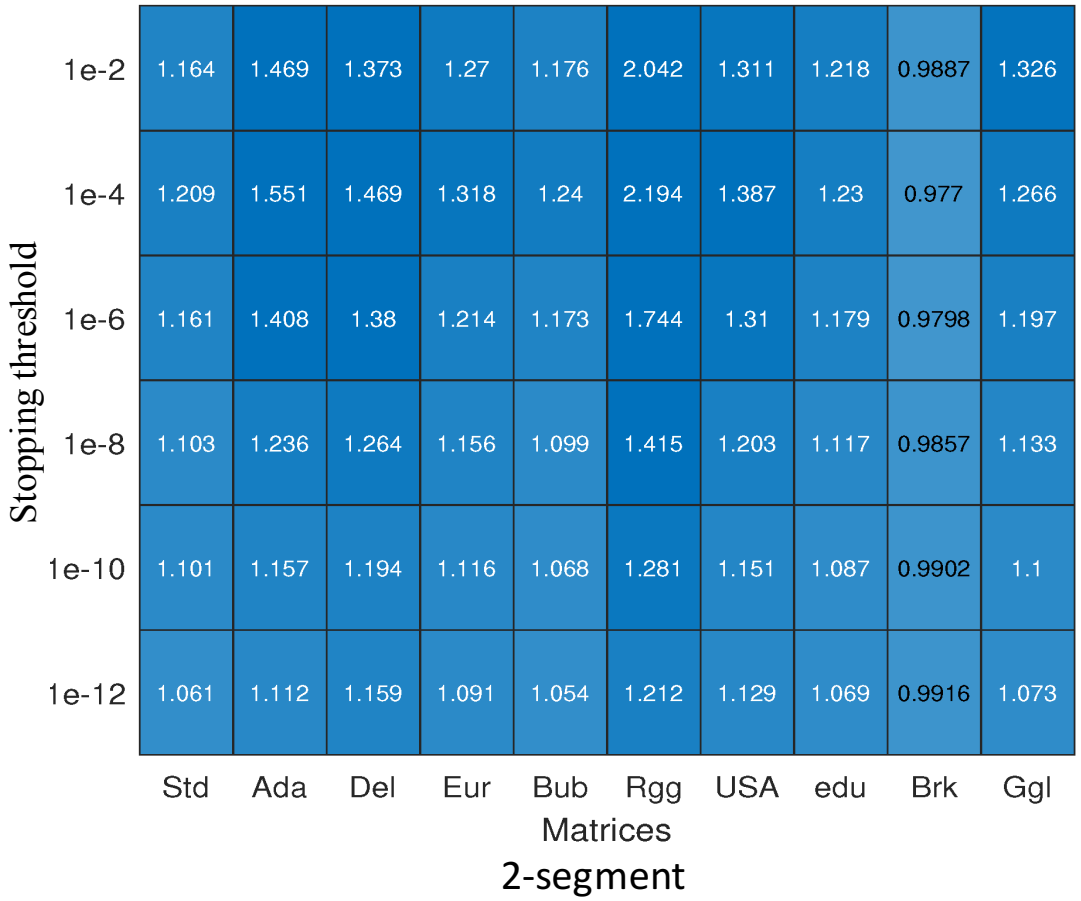


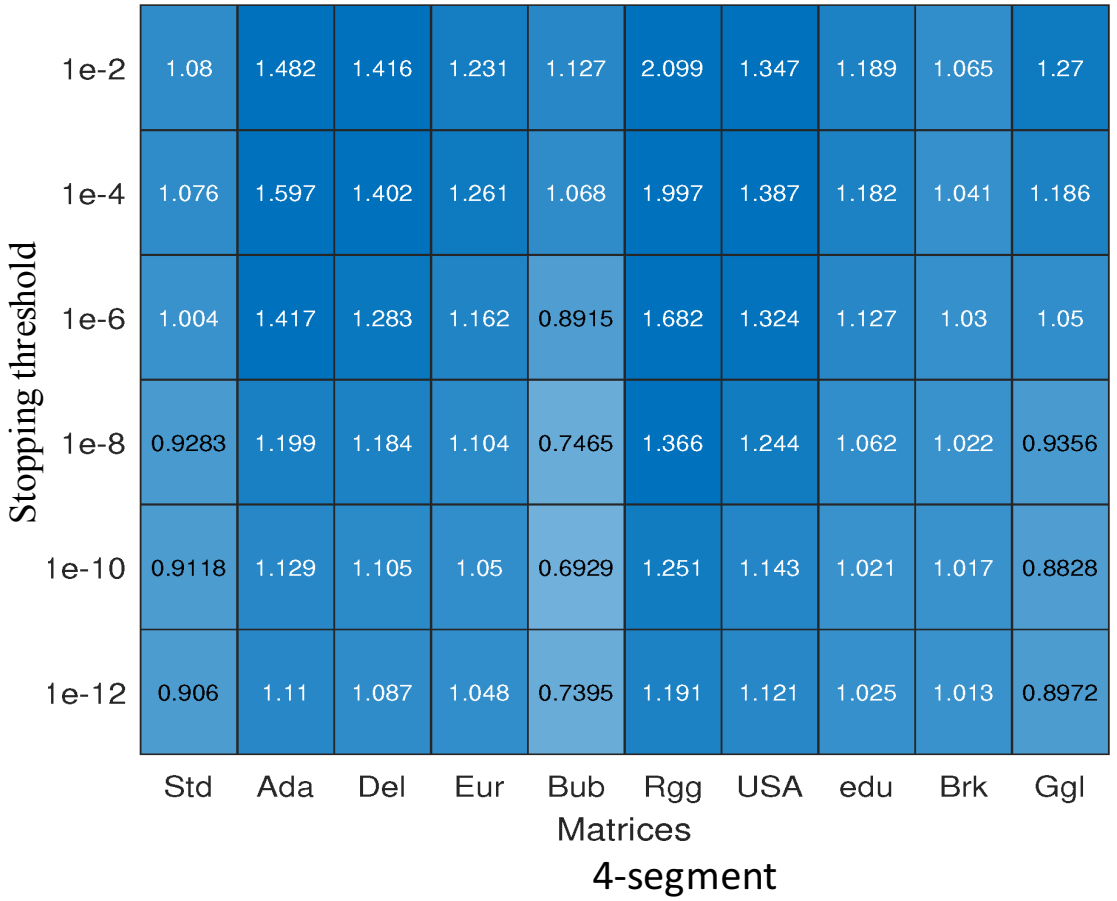
Benchmark results on NVIDIA V100 GPU

Submitting to SC Workshop on Irregular Applications: **"Customized Precision PageRank on GPUs".**

Customized Precision PageRank

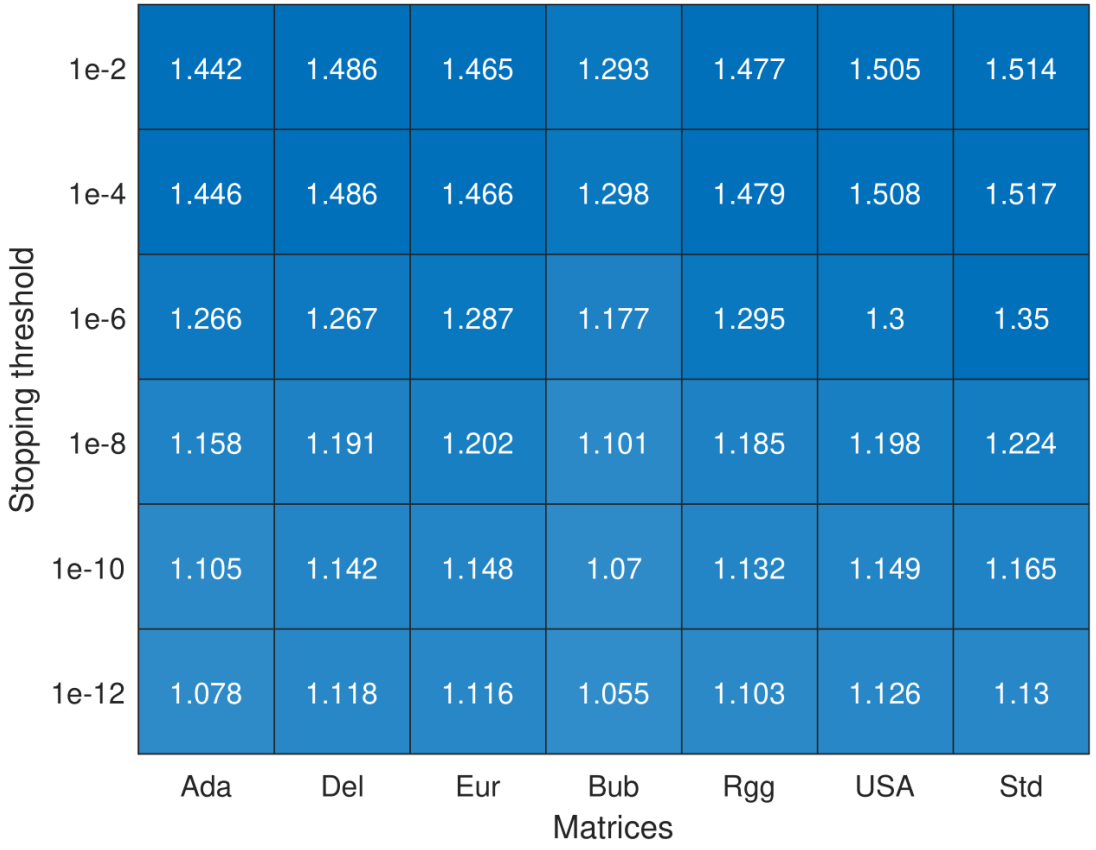
PageRank speedup over reference implementation (NVIDIA V100 GPU). CSR format.





Customized Precision PageRank

PageRank speedup over reference implementation (NVIDIA V100 GPU). ELL format.

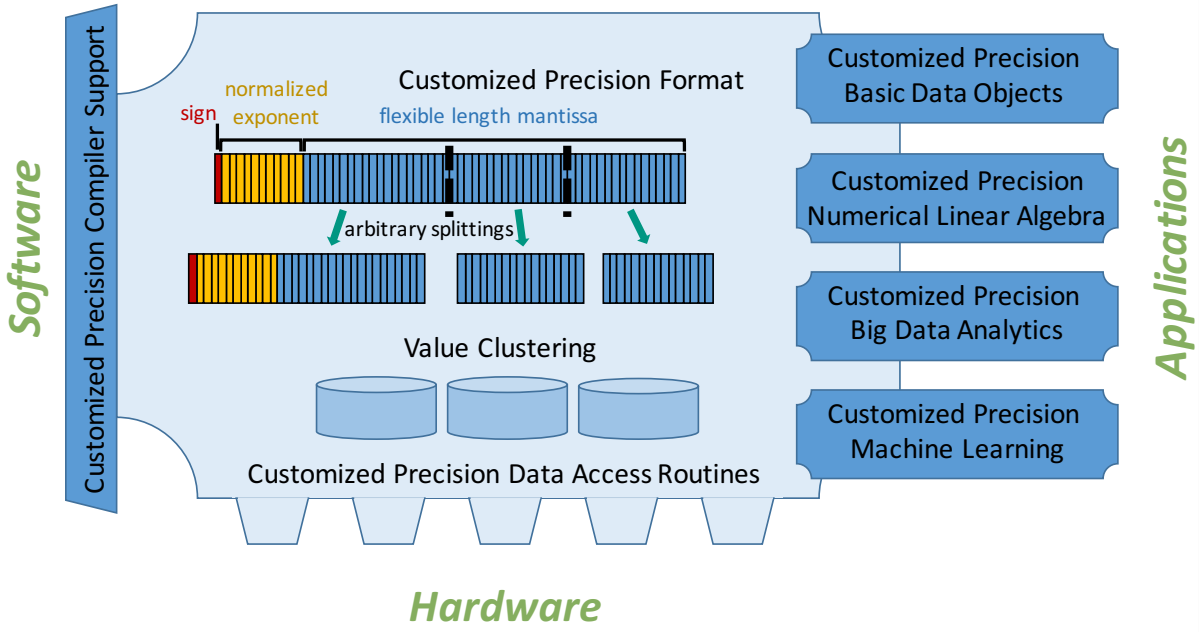


2-segment

4-segment

Summary and next steps

- **Decouple arithmetic precision from memory precision.**
- Using **customized precisions** for memory operations.
- Dynamically **adapt the accuracy** to the **algorithm** properties.
- Speedup of up to 1.3x for Jacobi iterations and PageRank algorithm.
- Speedup of up to 1.3x for adaptive precision block-Jacobi preconditioning.
- Creating a **Modular Precision Ecosystem** inside Ginkgo.



EXASCALE COMPUTING PROJECT

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration and the Helmholtz Impuls und VernetzungsfondVH-NG-1241.

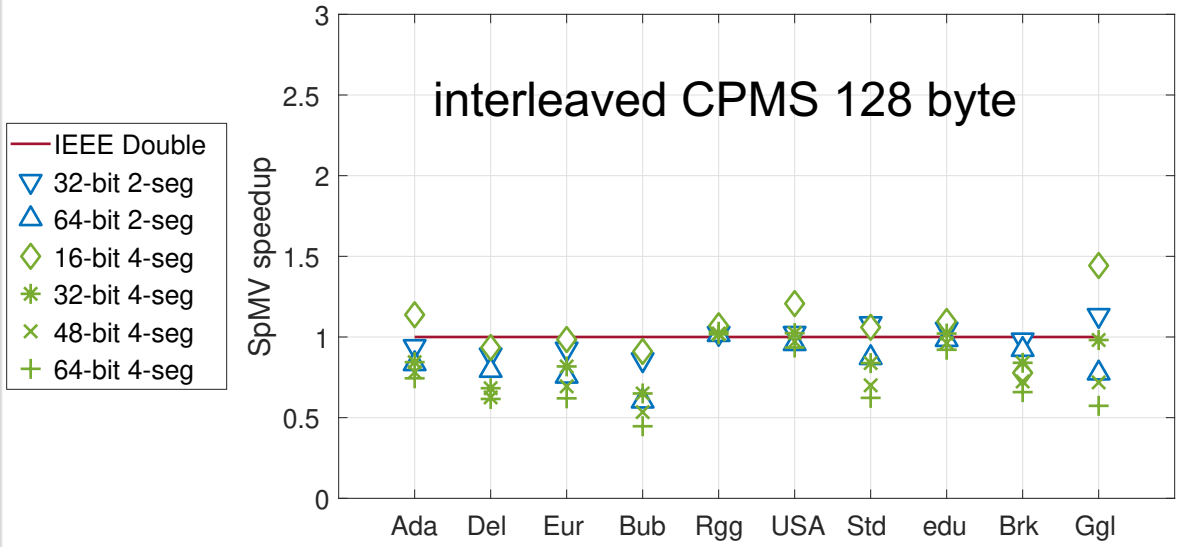
Customized Precision PageRank on CPUs?

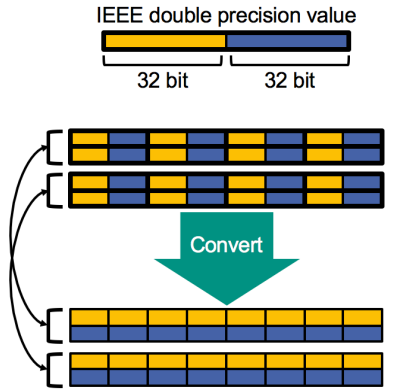
Much harder!

- Caches, prefetching, AVX…
- No easy-to-understand PTX
- Highly optimized compilers

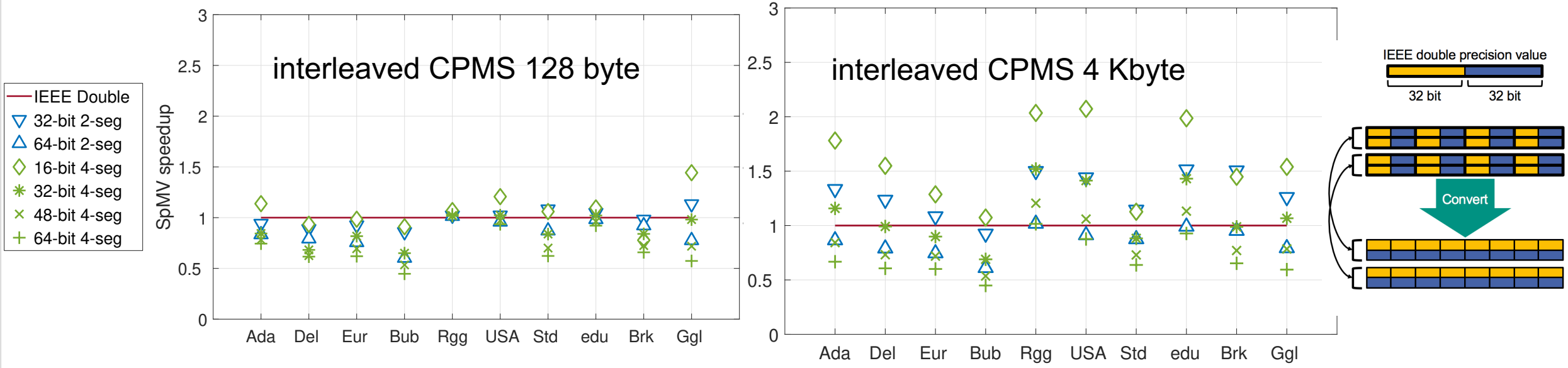
```
#pragma omp parallel for schedule(guided, ompHelper::chunkSize)
for (std::size_t row = 0; row < m; ++row) {
    IndexType i = rowStartIndices[row];
    __m256d currentResPack = _mm256_setr_pd(0.0, 0.0, 0.0, 0.0);
    if (rowStartIndices[row + 1] > (4-1)) {
        for (i = rowStartIndices[row]; i < rowStartIndices[row + 1] - (4-1); i += 4) {
            const __m128i rankIndices32 = _mm_loadu_si128((__m128i *) (colIndices + i));
            __m256d matValPack =
                avx2Helper::readPack<partPointersToRead, NumberSegments, double>(valsSplit, i);
            __m256d rankPack =
                avx2Helper::readPack<partPointersToRead, NumberSegments, double>(oldRankSplit, rankIndices32);
            __m256d multResPack = _mm256_mul_pd(matValPack, rankPack);
            currentResPack = _mm256_add_pd(currentResPack, multResPack);
        }
    }
```

Customized Precision SpMV on CPUs

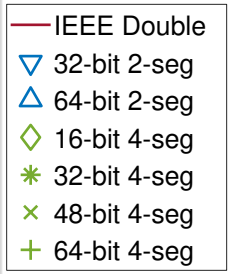


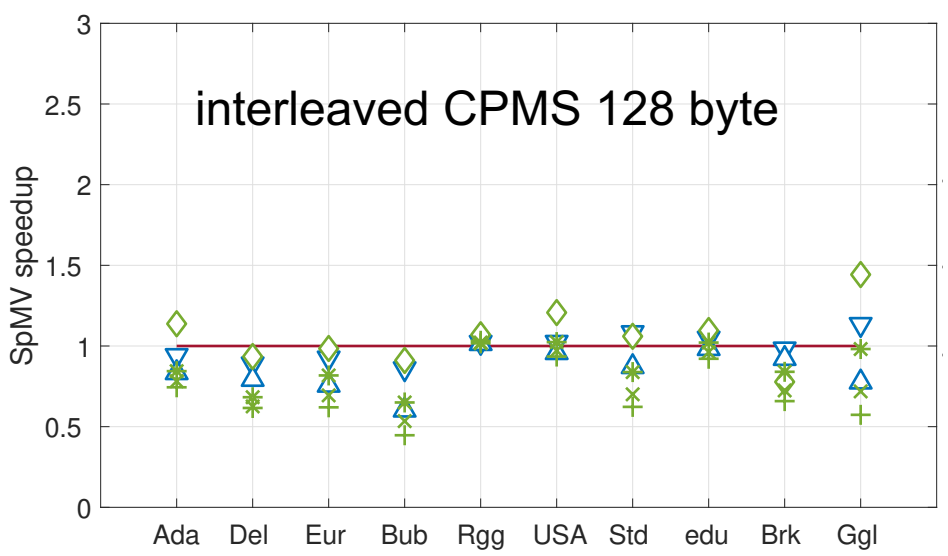


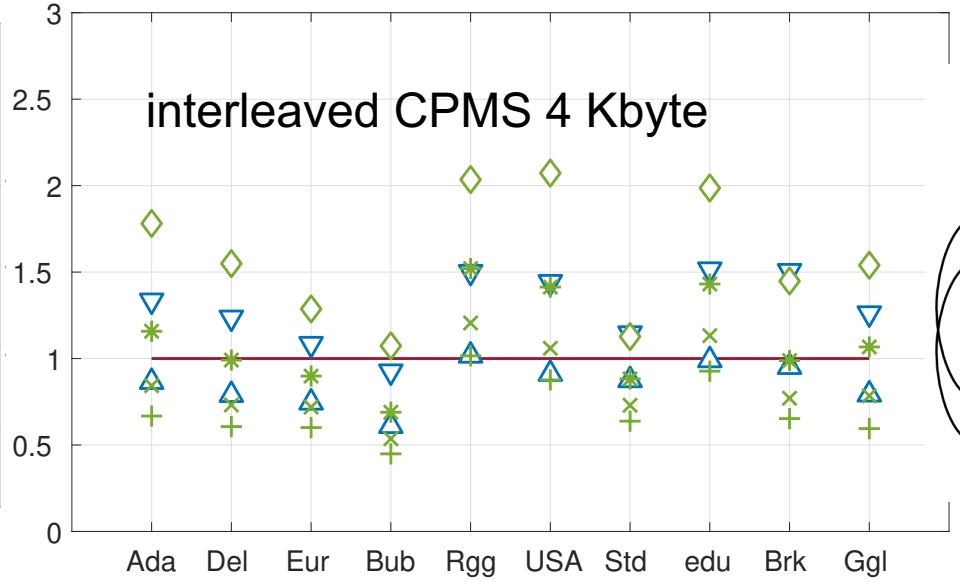
Customized Precision SpMV on CPUs

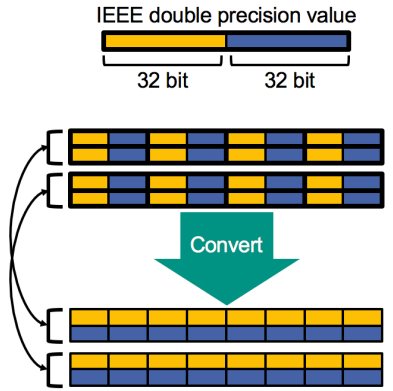


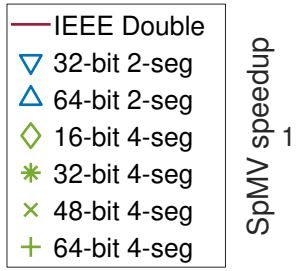
Customized Precision SpMV on CPUs

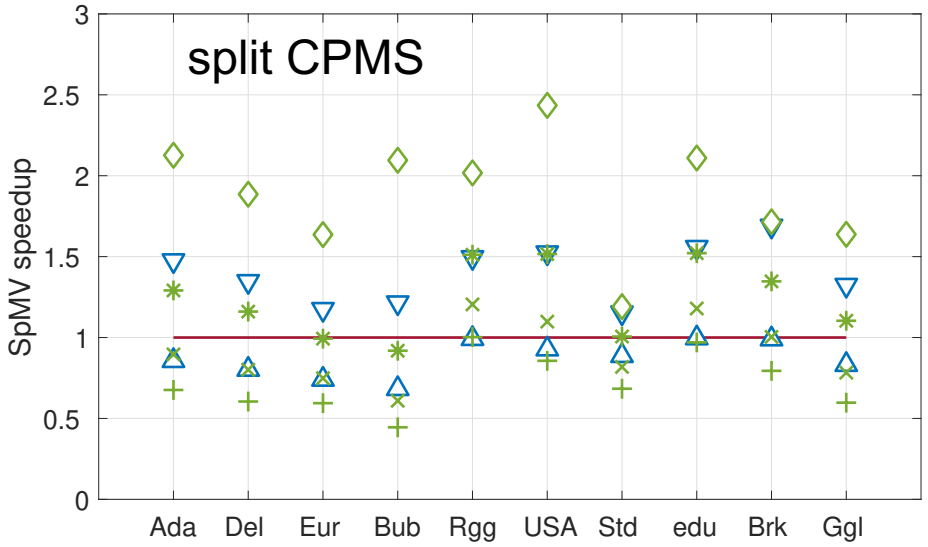


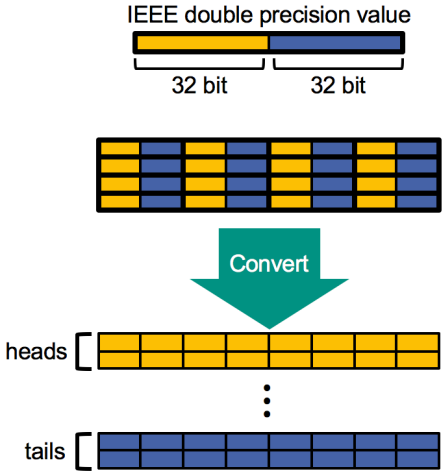






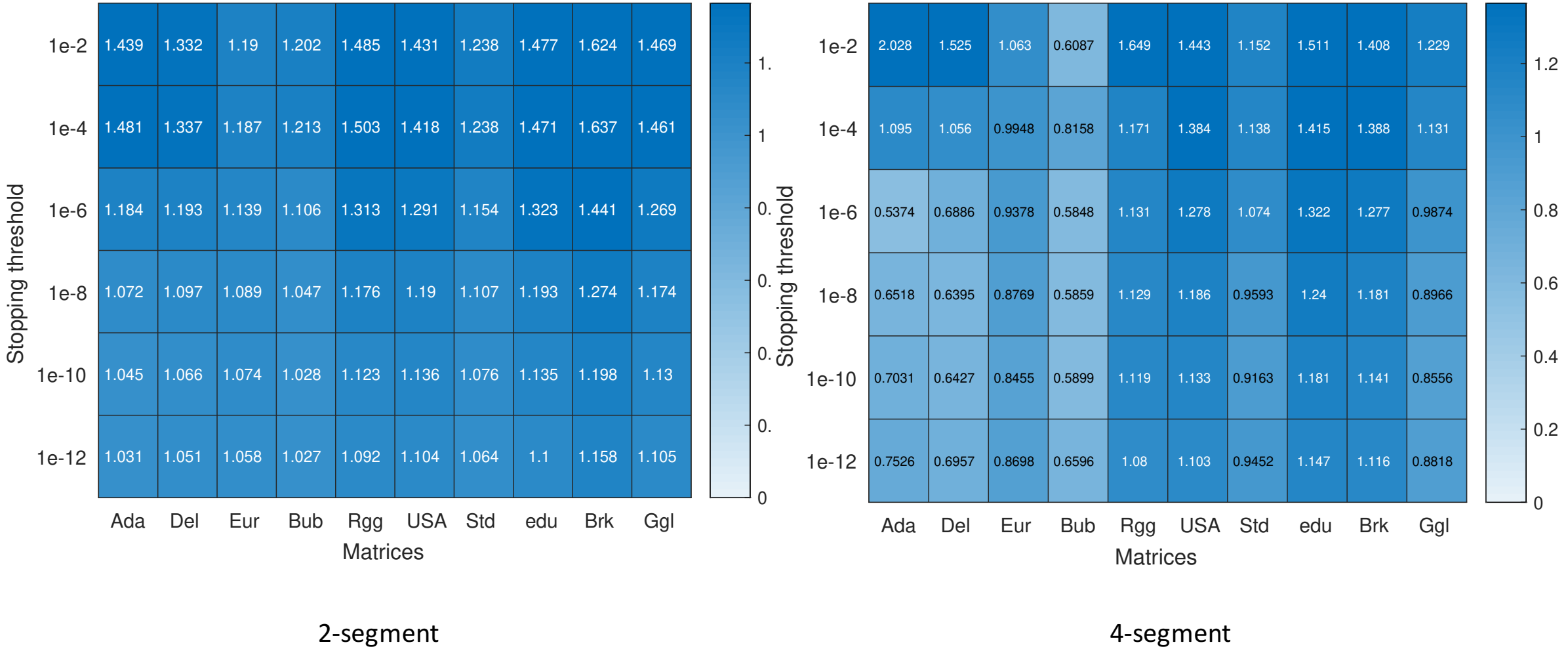






Customized Precision PageRank on CPUs

PageRank speedup over reference implementation (2x Intel(R) Xeon(R) Platinum 8168 "Skylake"). CSR format.

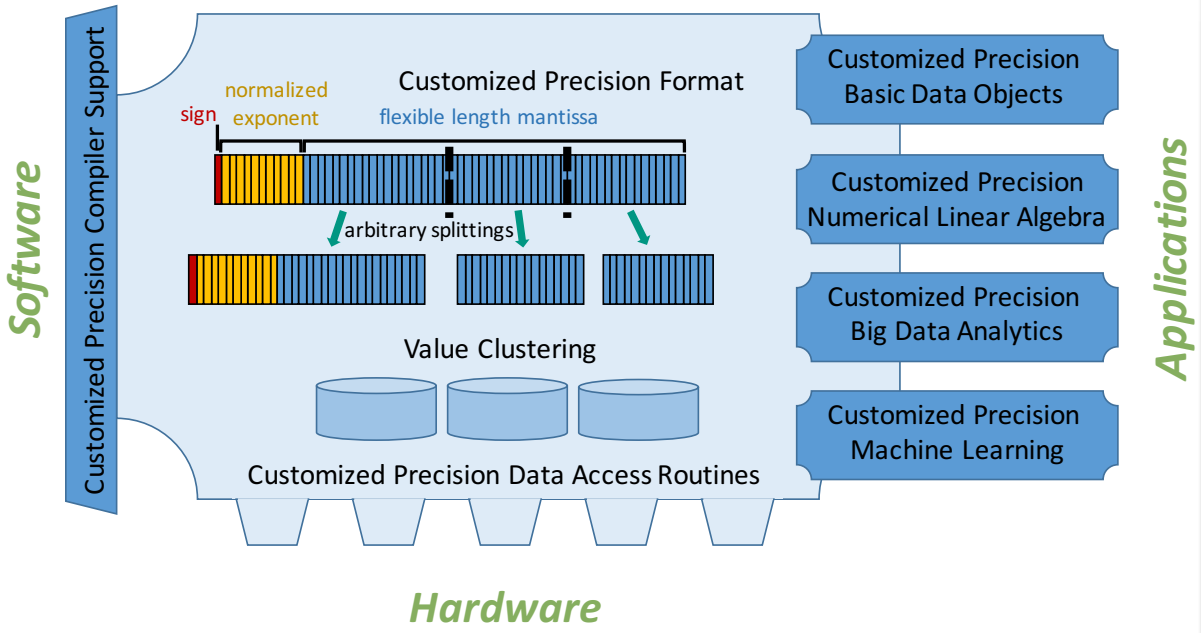


2-segment

4-segment

Summary and next steps

- **Decouple arithmetic precision from memory precision.**
- Using **customized precisions** for memory operations.
- Dynamically **adapt the accuracy** to the **algorithm** properties.
- Speedup of up to 1.3x for Jacobi iterations and PageRank algorithm.
- Speedup of up to 1.3x for adaptive precision block-Jacobi preconditioning.
- Creating a **Modular Precision Ecosystem** inside Ginkgo.



EXASCALE COMPUTING PROJECT

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration and the Helmholtz Impuls und VernetzungsfondVH-NG-1241.