Addressing the Communication Bottleneck:
Towards a Modular Precision Ecosystem for High Performance Computing

A Structured Approach for Programming Extreme-Scale and Heterogeneous Systems (EPIGRAM-HS project)
Zurich, September 9th, 2019

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

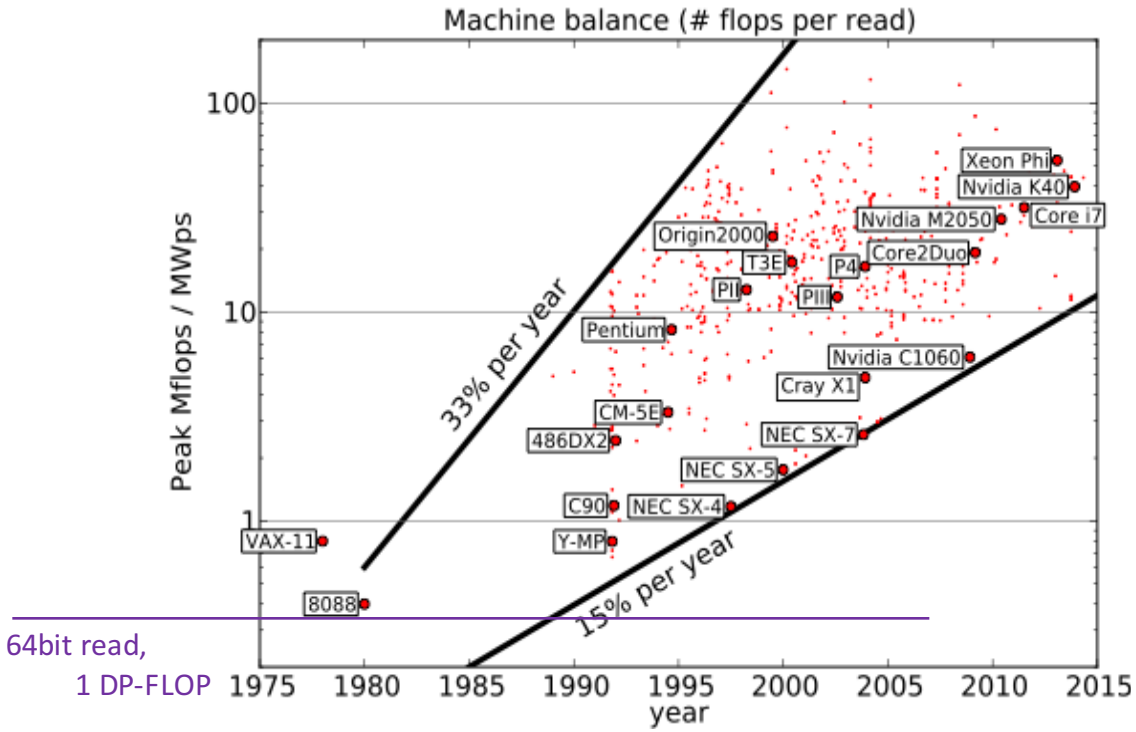
Where do we stand?

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

Copyright@ORNL



John D. McCalpin (TACC)

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

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

Where do we stand?

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

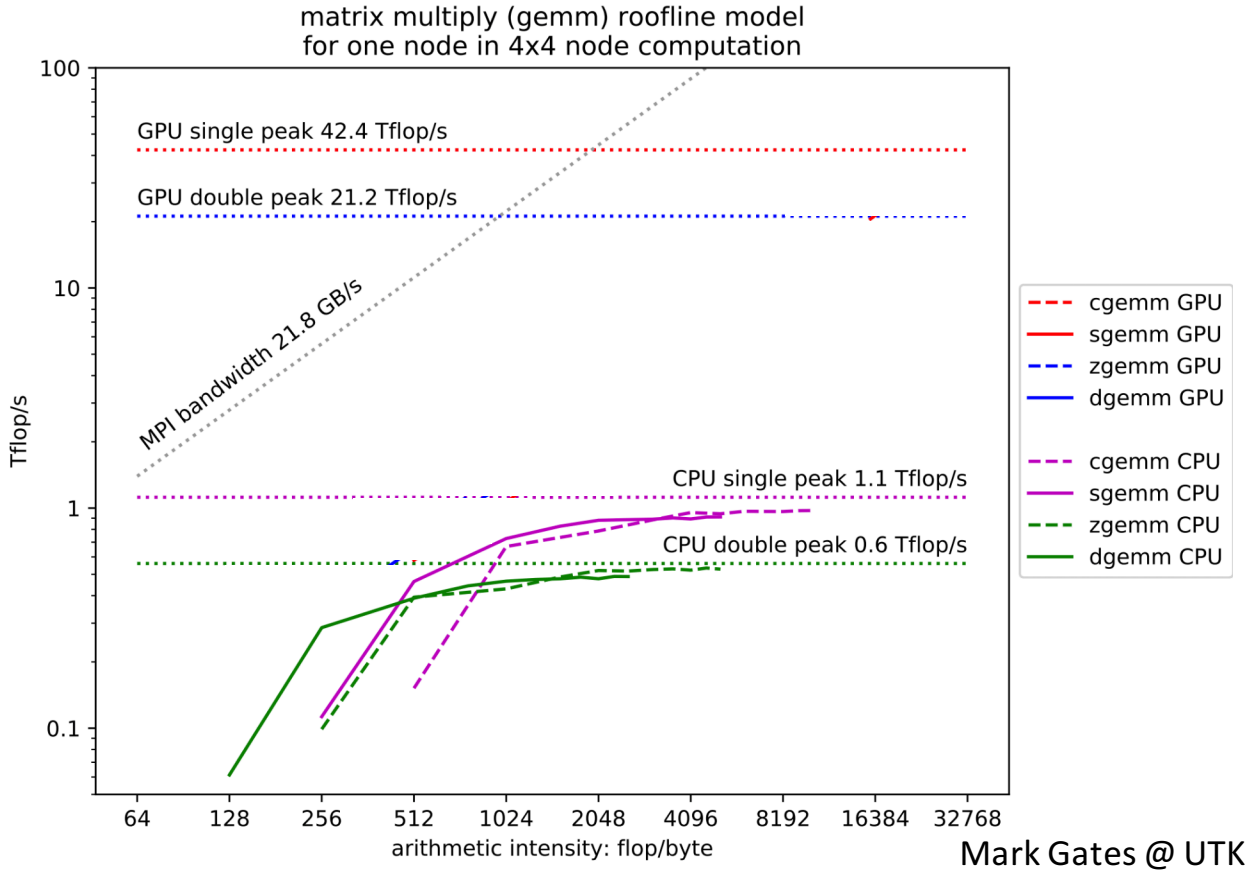
summit

- 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

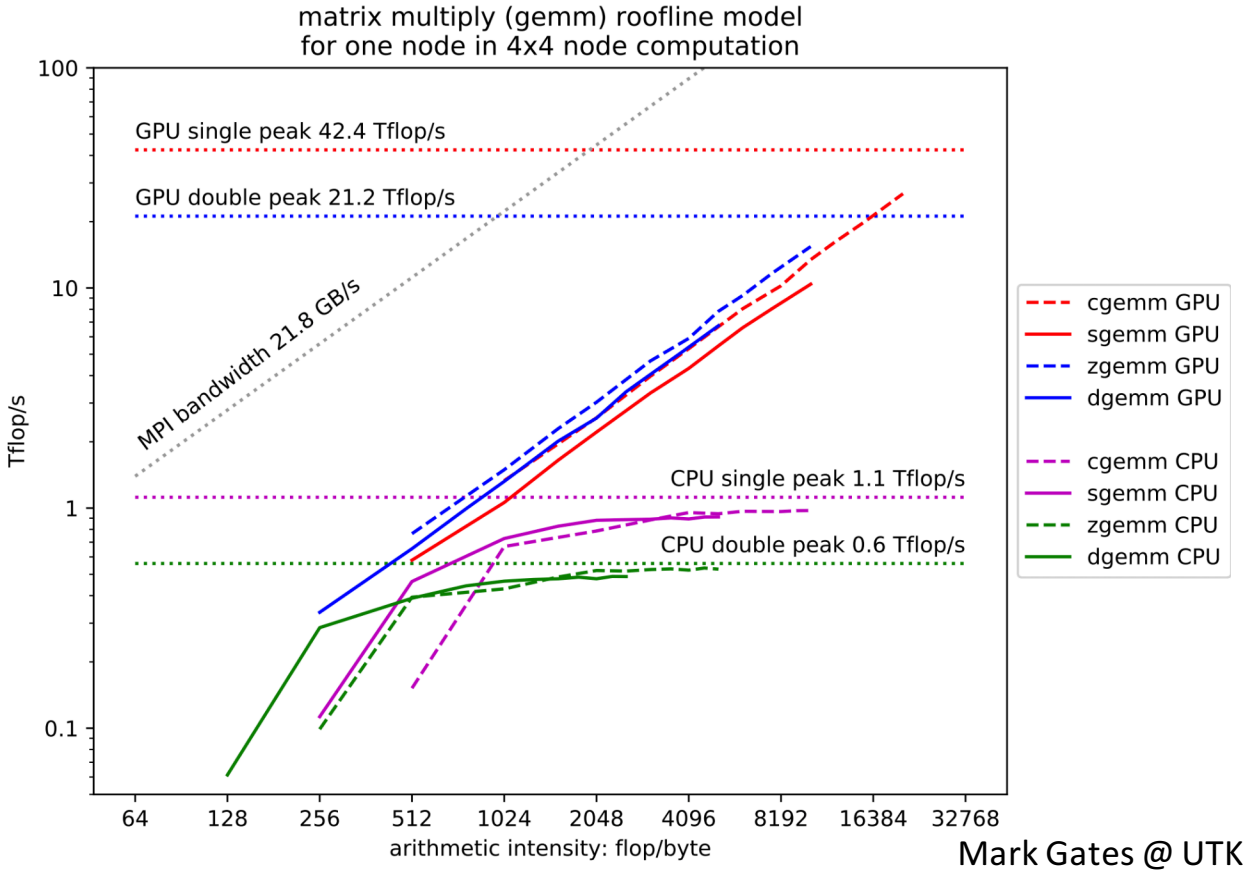
summit

- 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;



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

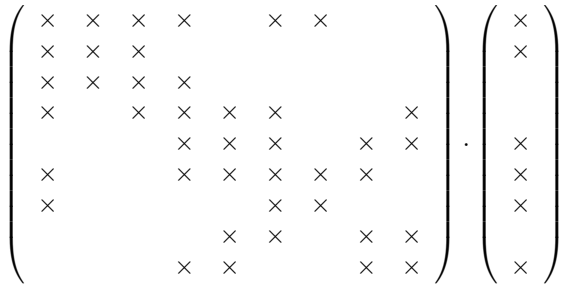
Copyright@ORNL

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

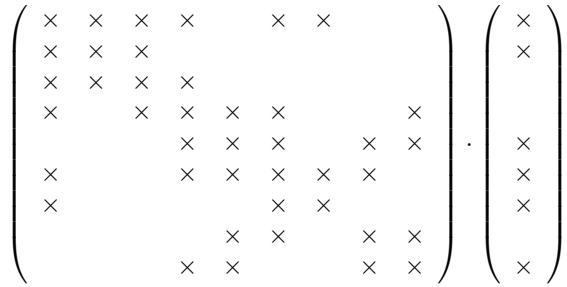
- Node: 2 IBM POWER9 + 6 NVIDIA V100 GPUs
- 4,608 nodes, 9,216 IBM Power9 CPUs
- 27,648 V100 GPUs (8 TFLOPs / GPU)
- Peak performance of **200 Pflop/s** for modeling & simulation
- Peak performance of **3.3 Eflop/s** (10^18) for 16 bit floating point used in data analytics and artificial intelligence

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/

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

@ORNL

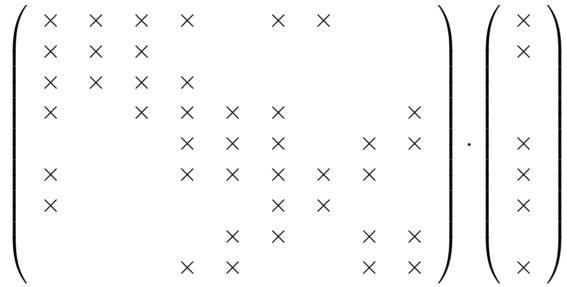
Dense Matrix Operations?

- The inter-node comm[illegible]n is the limiting resource;
- Each node has m[illegible]putational power than what we can leverage;

Sparse / Gra[illegible]blems?

- ***Spa[illegible]ix Vector Product*** (SpMV) is a central building block;
- [illegible]like for the test problems at the SuiteSparse Matrix [illegible]lection[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.



Radically decouple storage format from arithmetic format.

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

Where do we stand?

OAK RIDGE National Laboratory | L C F

- Node: 2 IBM POWER
- 4,608 nodes, 9,216 I
- 27,648 V100 GPUs (
- Peak performance of & simulation
- Peak performance of for 16 bit floating po analytics and artifici

Copyright@ORNL

Radically decouple storage format from arithmetic format.

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

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**

urce;

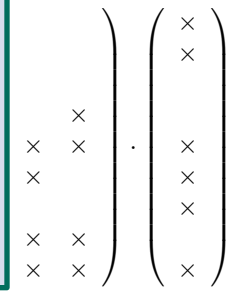
at we can leverage;

uilding block;

e Matrix

ngle-node SpMV;

itude slower



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

Use reduced precision for “approximate Operators”

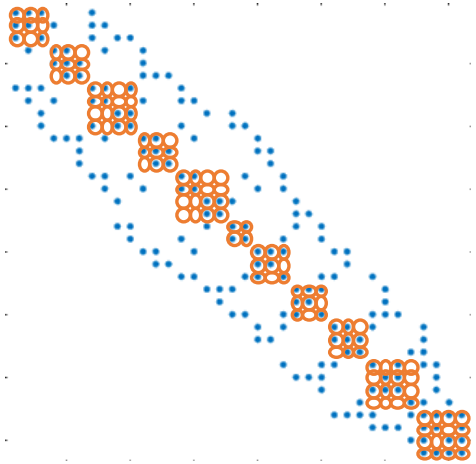
- **Preconditioners for iterative solvers.**
  - Idea: Approximate inverse of system matrix to make the system “easier to solve”: $P^{-1} \approx A^{-1}$

    $\tilde{A} = P^{-1}A\,, \quad \tilde{b} = P^{-1}b\,,$ 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

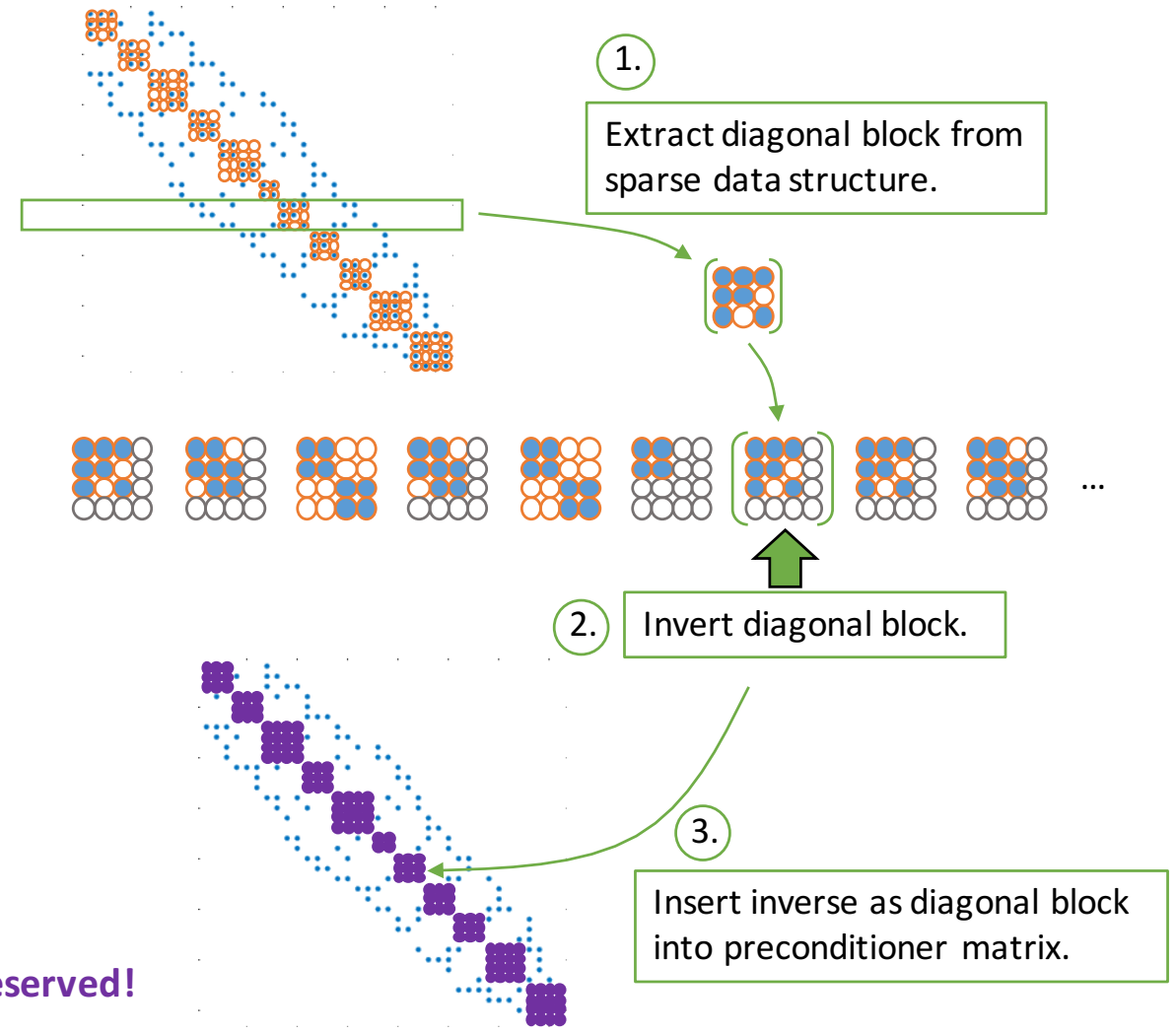
Preconditioner Setup:

- 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:

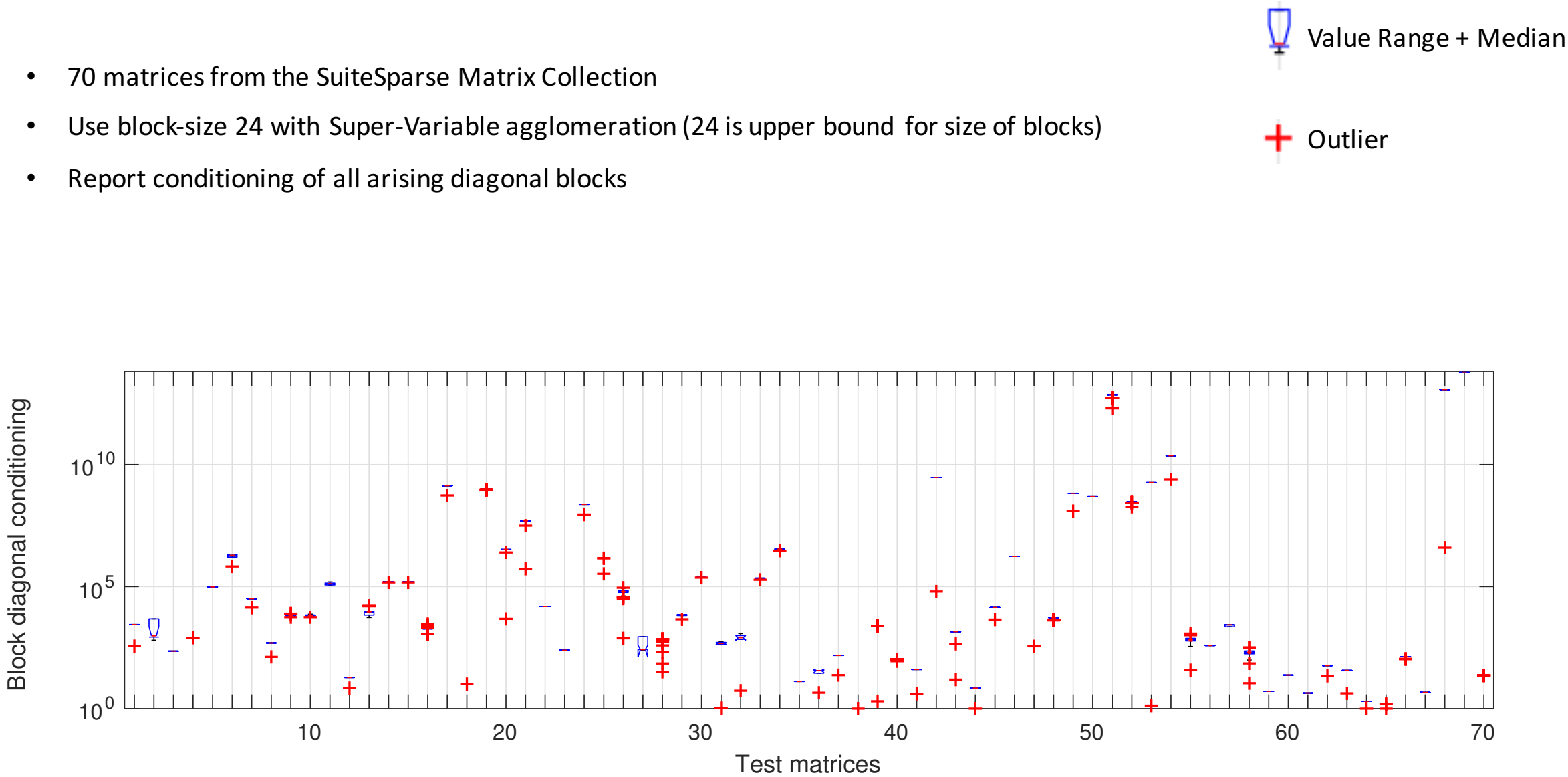
$$y := P^{-1}x$$



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

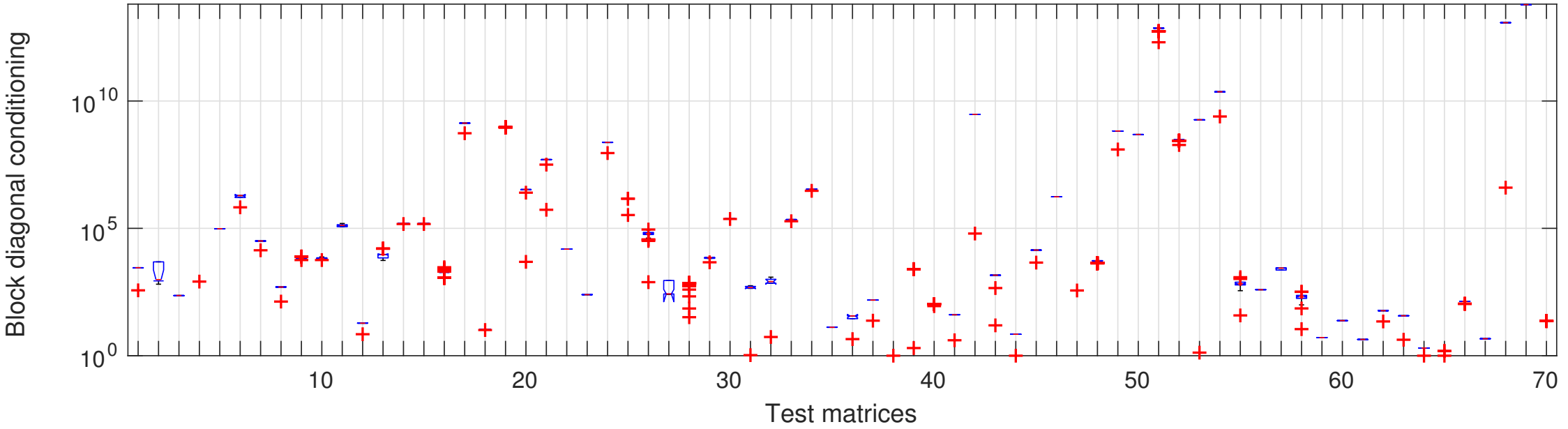
Adaptive Precision Block-Jacobi Preconditioning

- 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

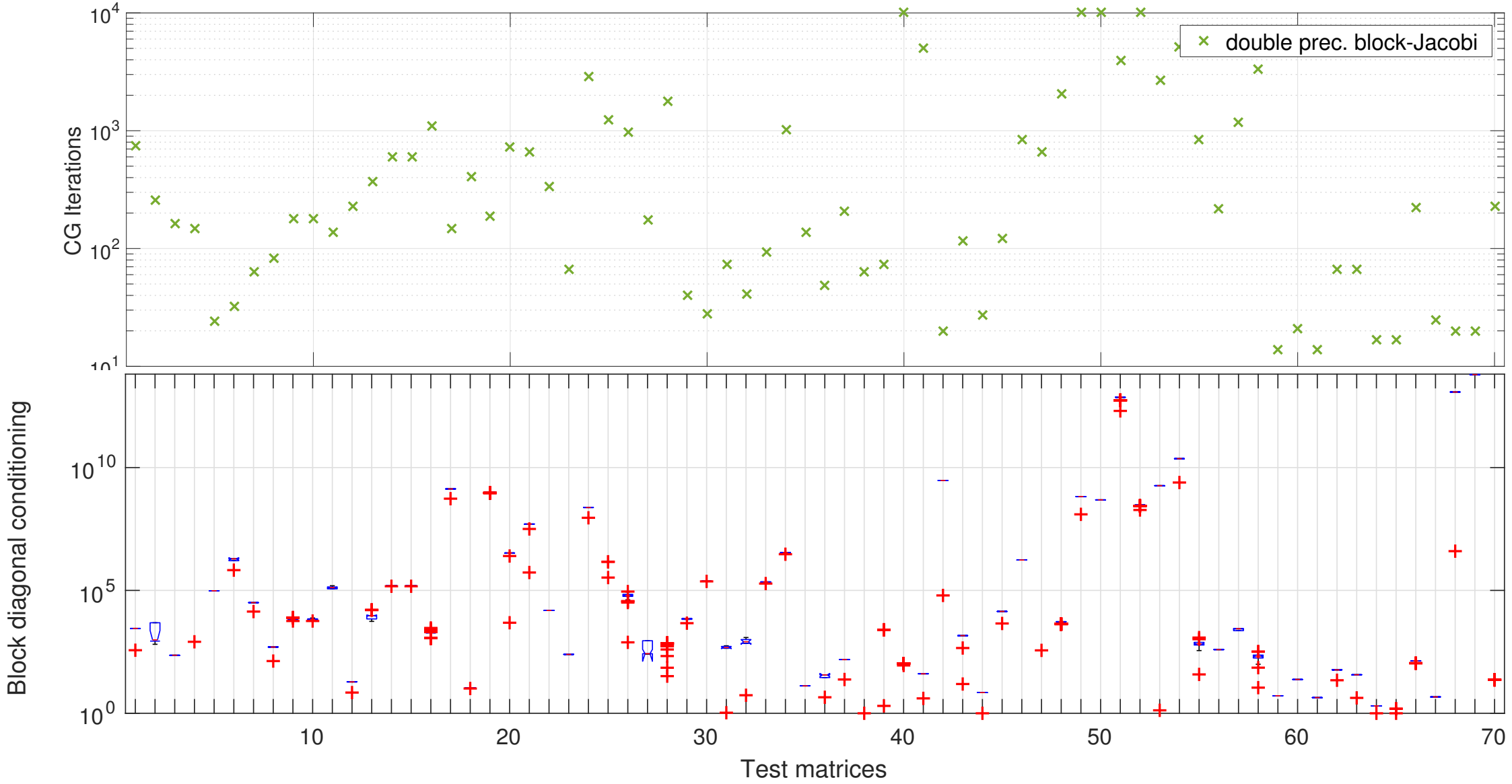


Adaptive Precision Block-Jacobi Preconditioning

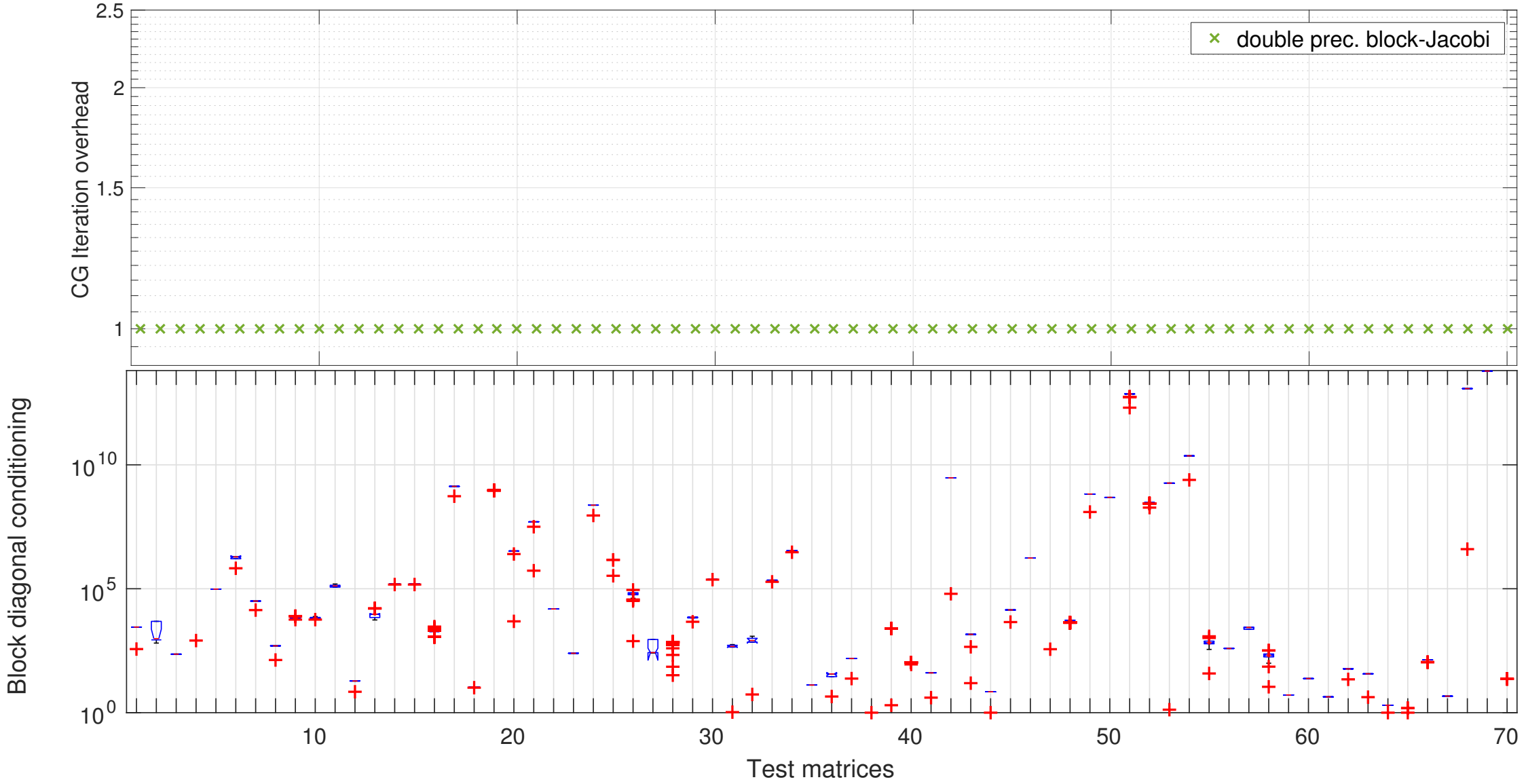
- 70 matrices from the SuiteSparse Matrix Collection
- Use block-size 24 with Super-Variable agglomeration (24 is upper bound for size of blocks)
- Report conditioning of all arising diagonal blocks
- Analyze the impact of storing block-Jacobi in lower precision a top-level Conjugate Gradient solver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



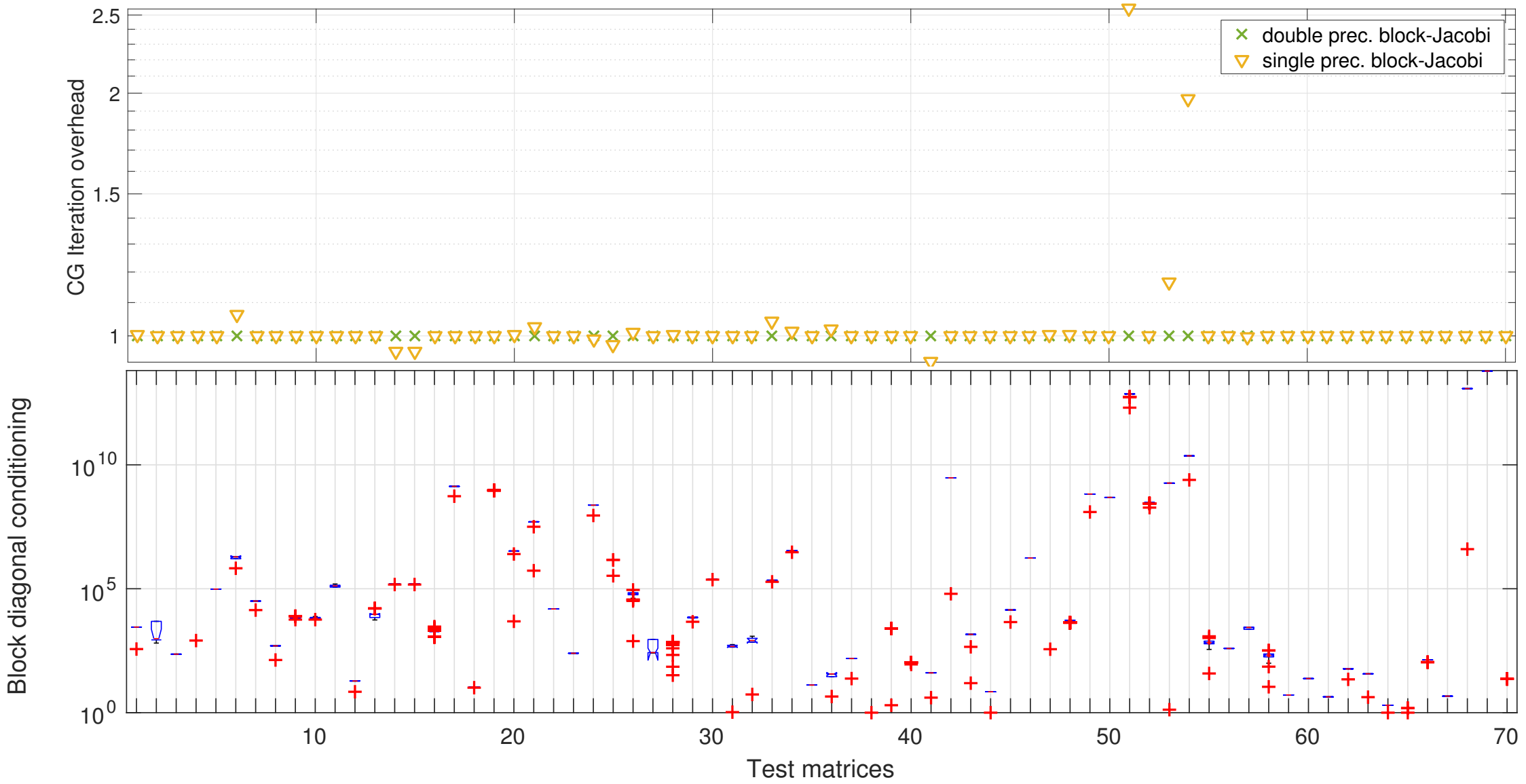
Adaptive Precision Block-Jacobi Preconditioning



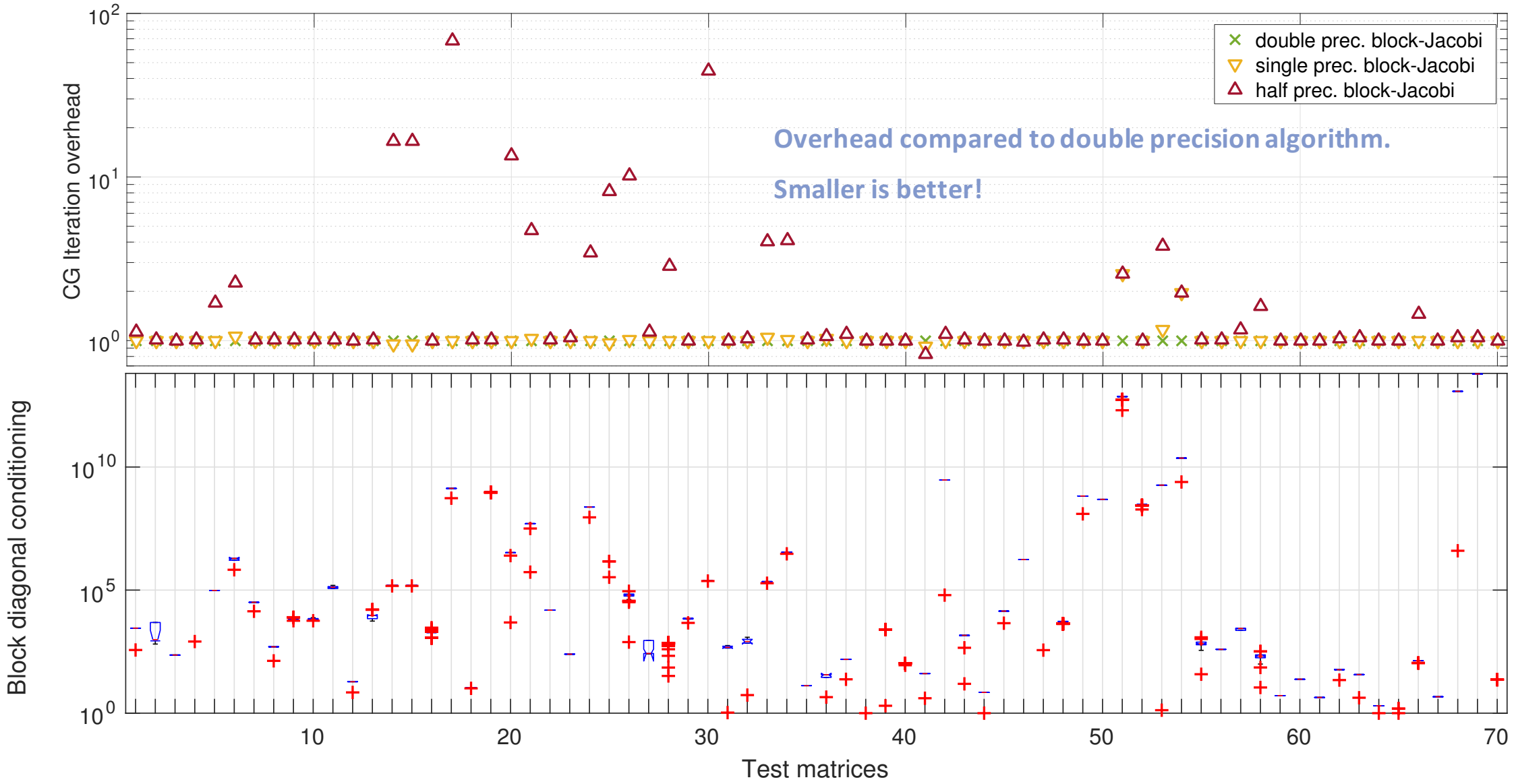
Adaptive Precision Block-Jacobi Preconditioning



Adaptive Precision Block-Jacobi Preconditioning



Adaptive Precision Block-Jacobi Preconditioning

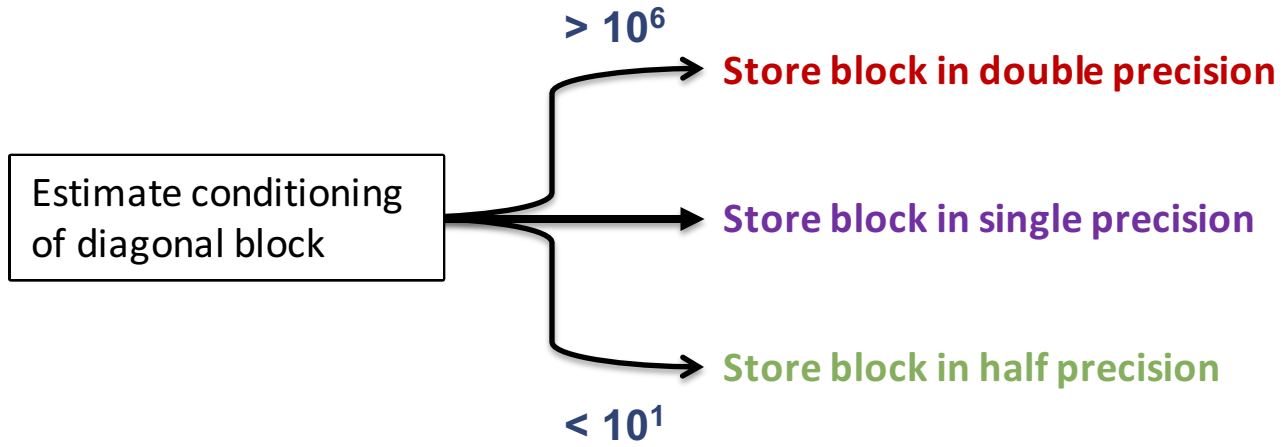


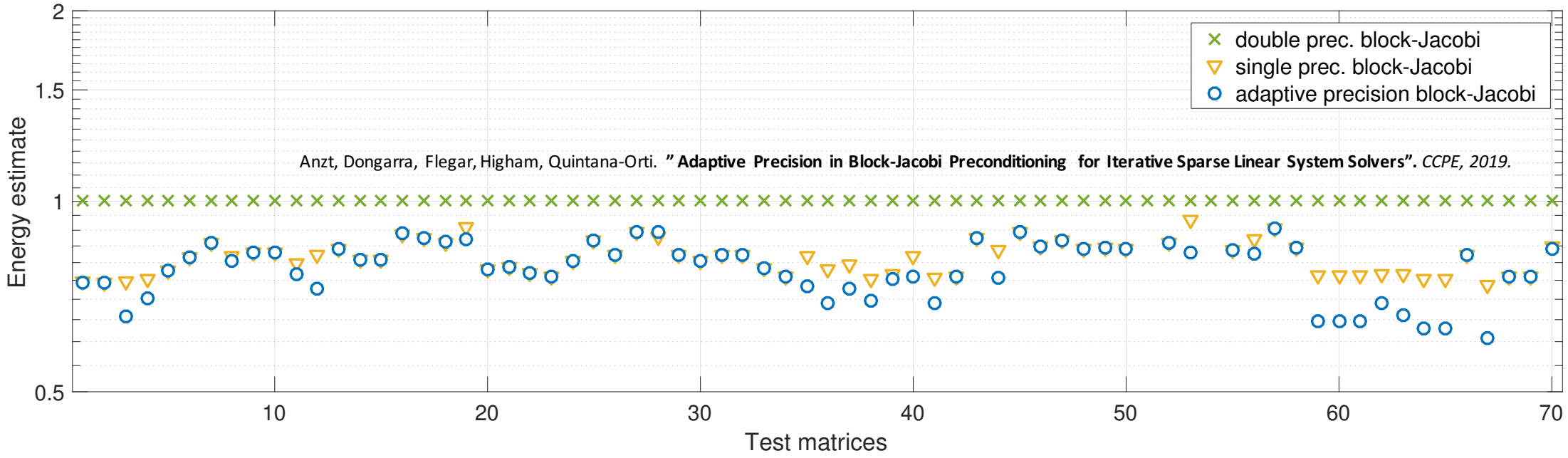
Adaptive Precision Block-Jacobi Preconditioning

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**

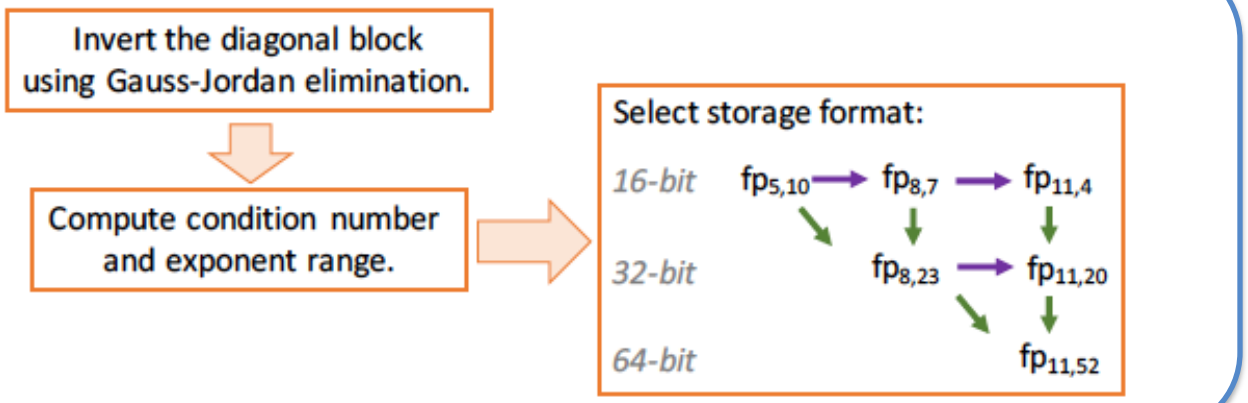




Adaptive Precision Block-Jacobi Preconditioning

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.

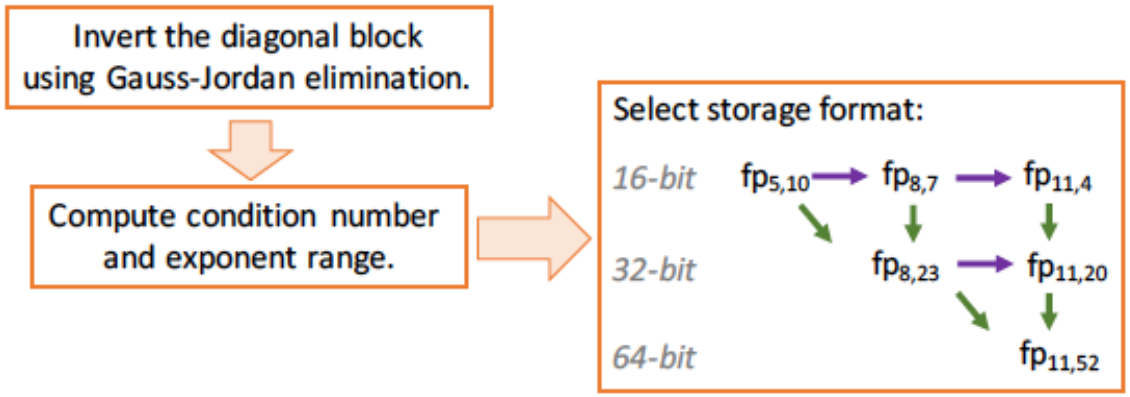


Flegar, Anzt, Quintana-Orti. **"Customized-Precision Block-Jacobi Preconditioning for Krylov Iterative Solvers on Data-Parallel Manycore Processors".** *TOMS, submitted.*

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);

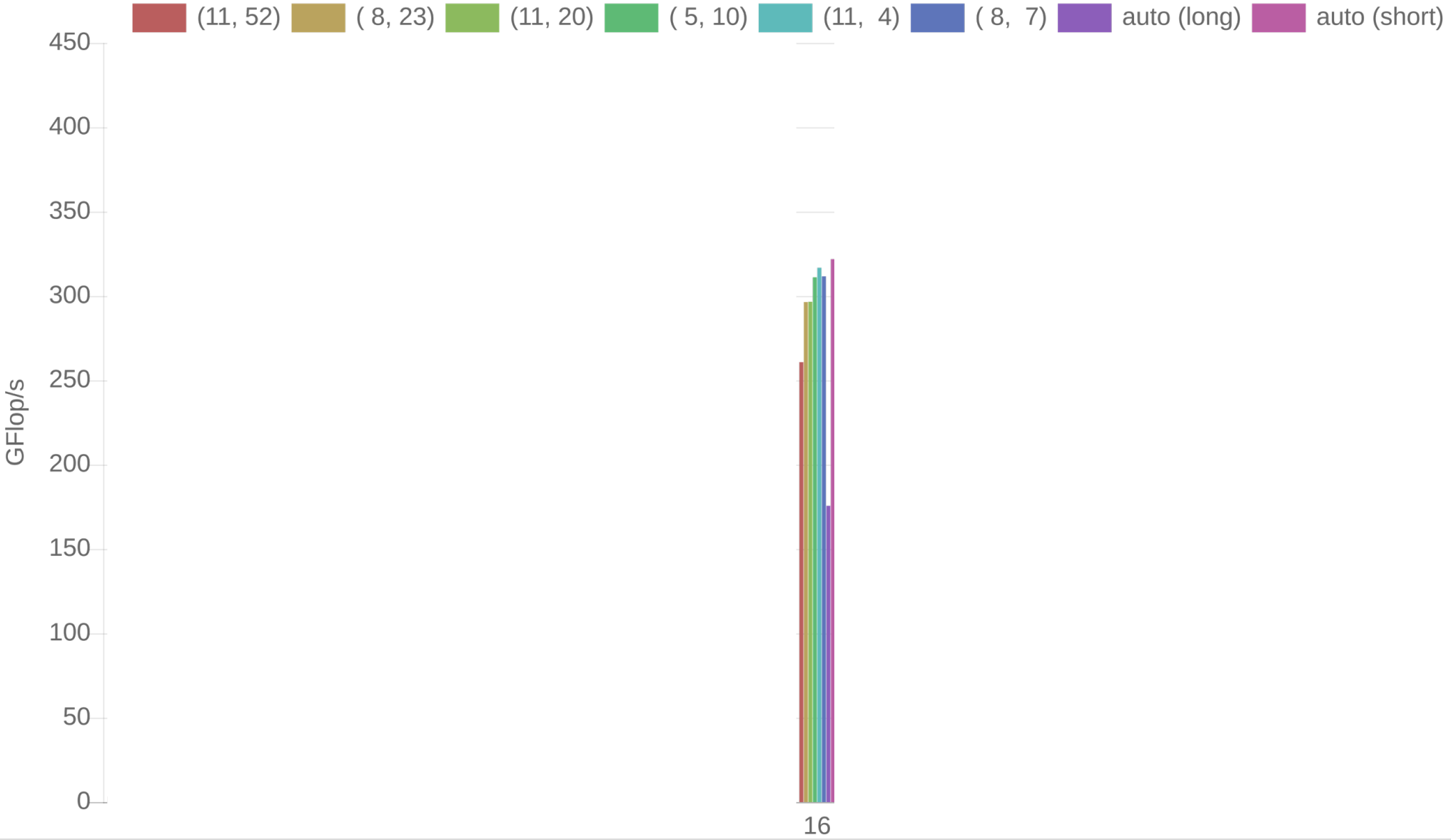
Flegar, Anzt, Quintana-Orti. **"Customized-Precision Block-Jacobi Preconditioning for Krylov Iterative Solvers on Data-Parallel Manycore Processors".** *TOMS, submitted.*

Precision distribution in Adaptive Block-Jacobi

Precision distribution in Adaptive Block-Jacobi

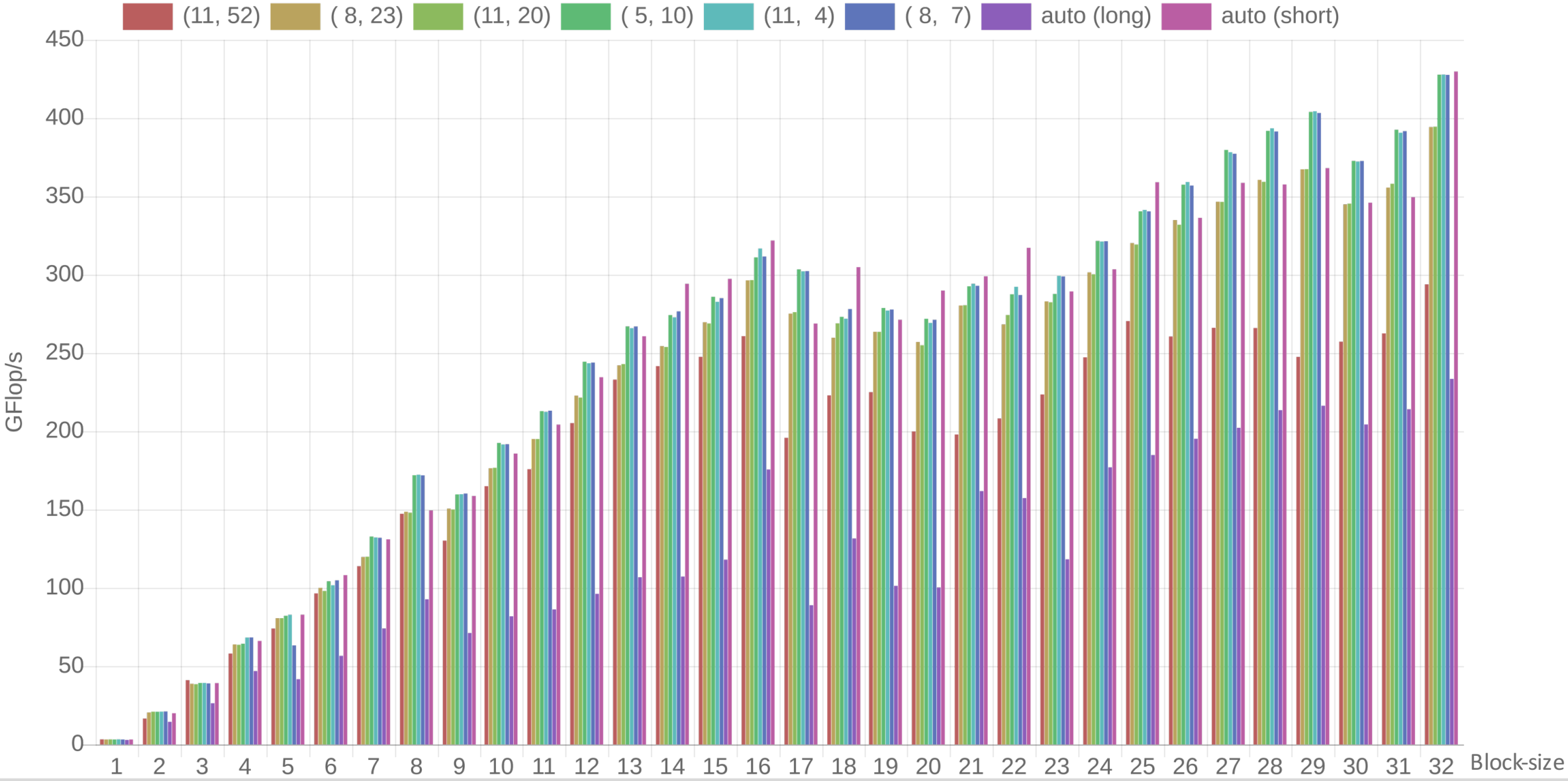
Adaptive Block-Jacobi Generation

NVIDIA V100 GPU



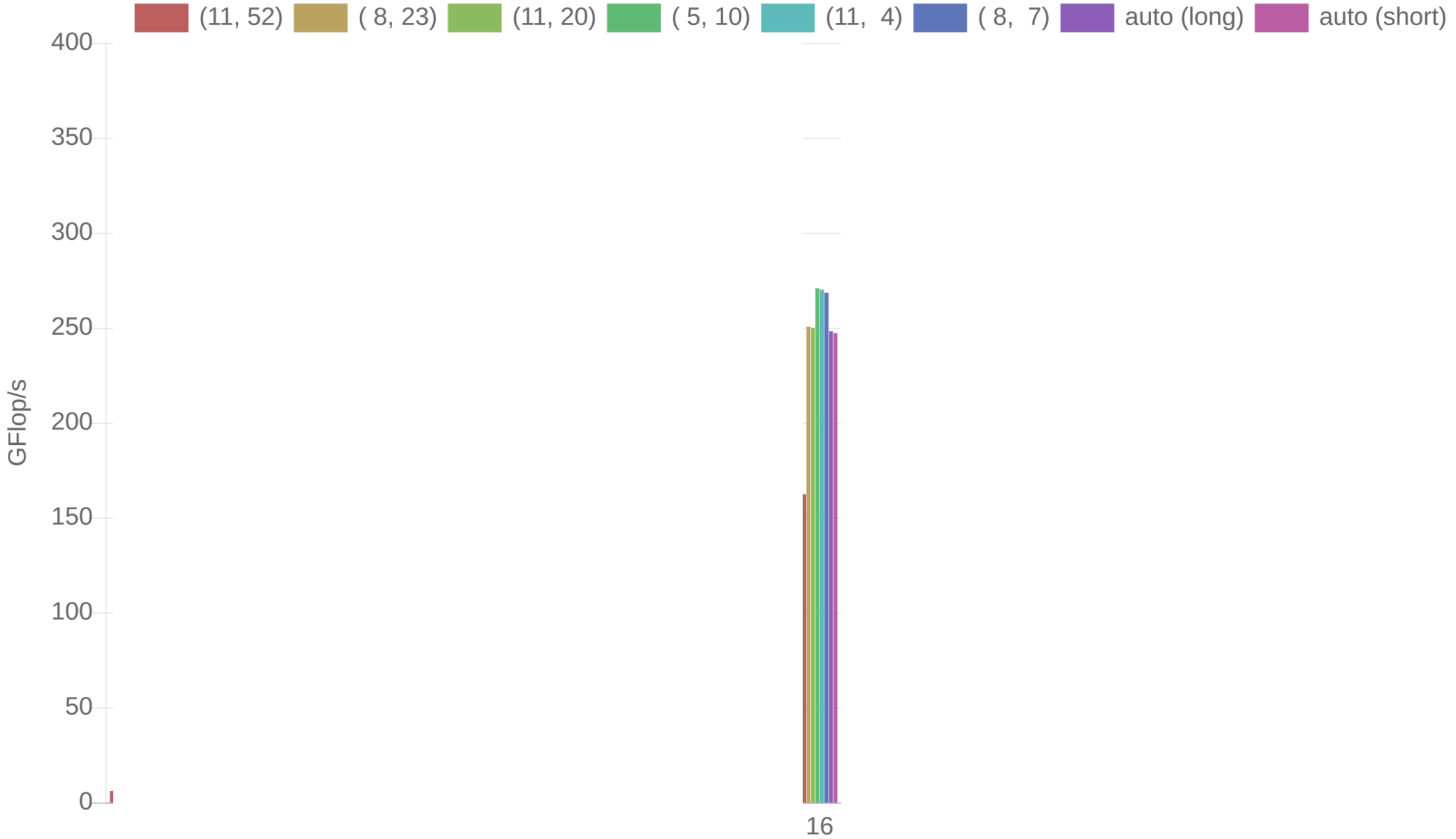
Adaptive Block-Jacobi Generation

NVIDIA V100 GPU



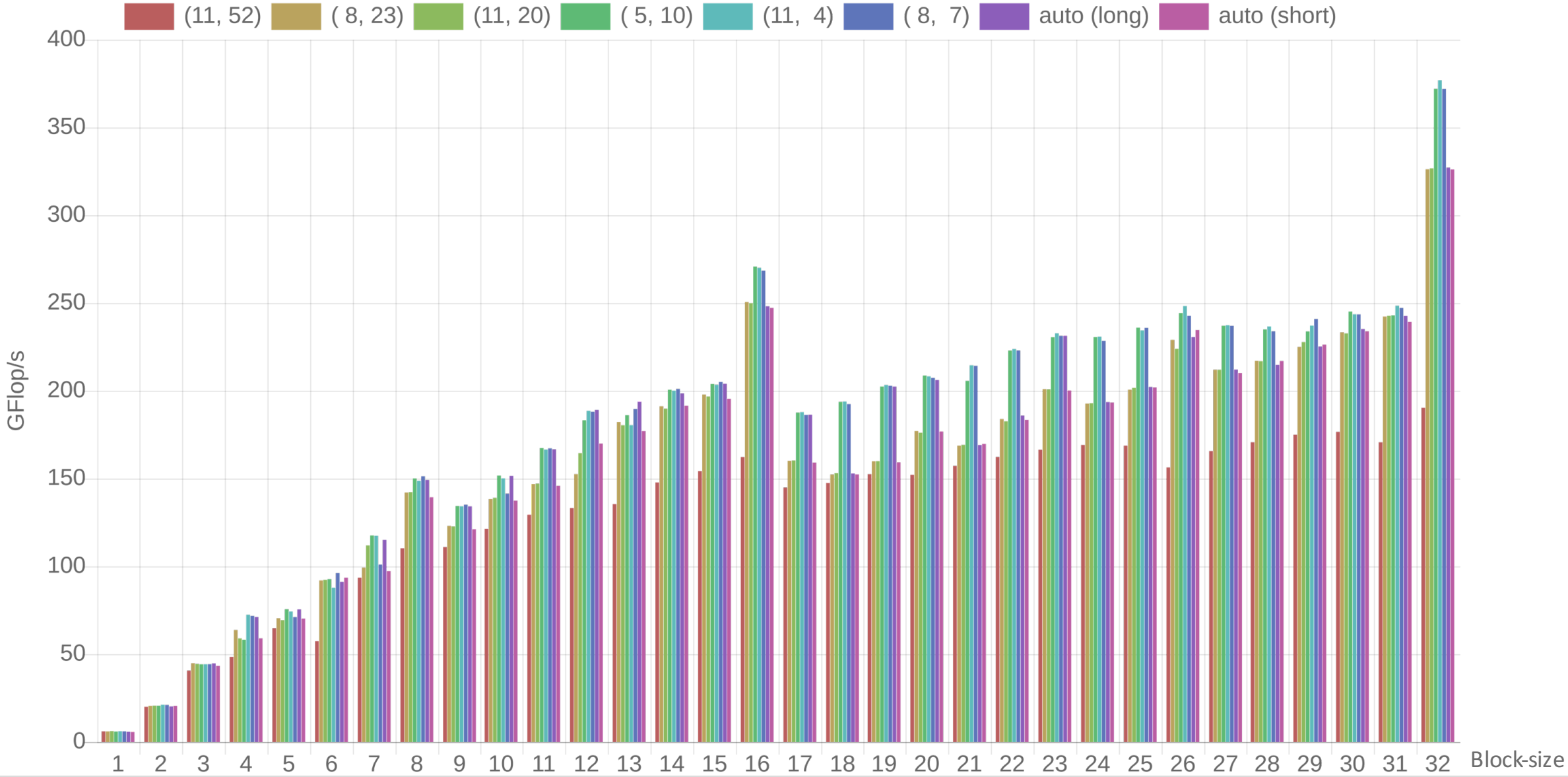
Adaptive Block-Jacobi Application

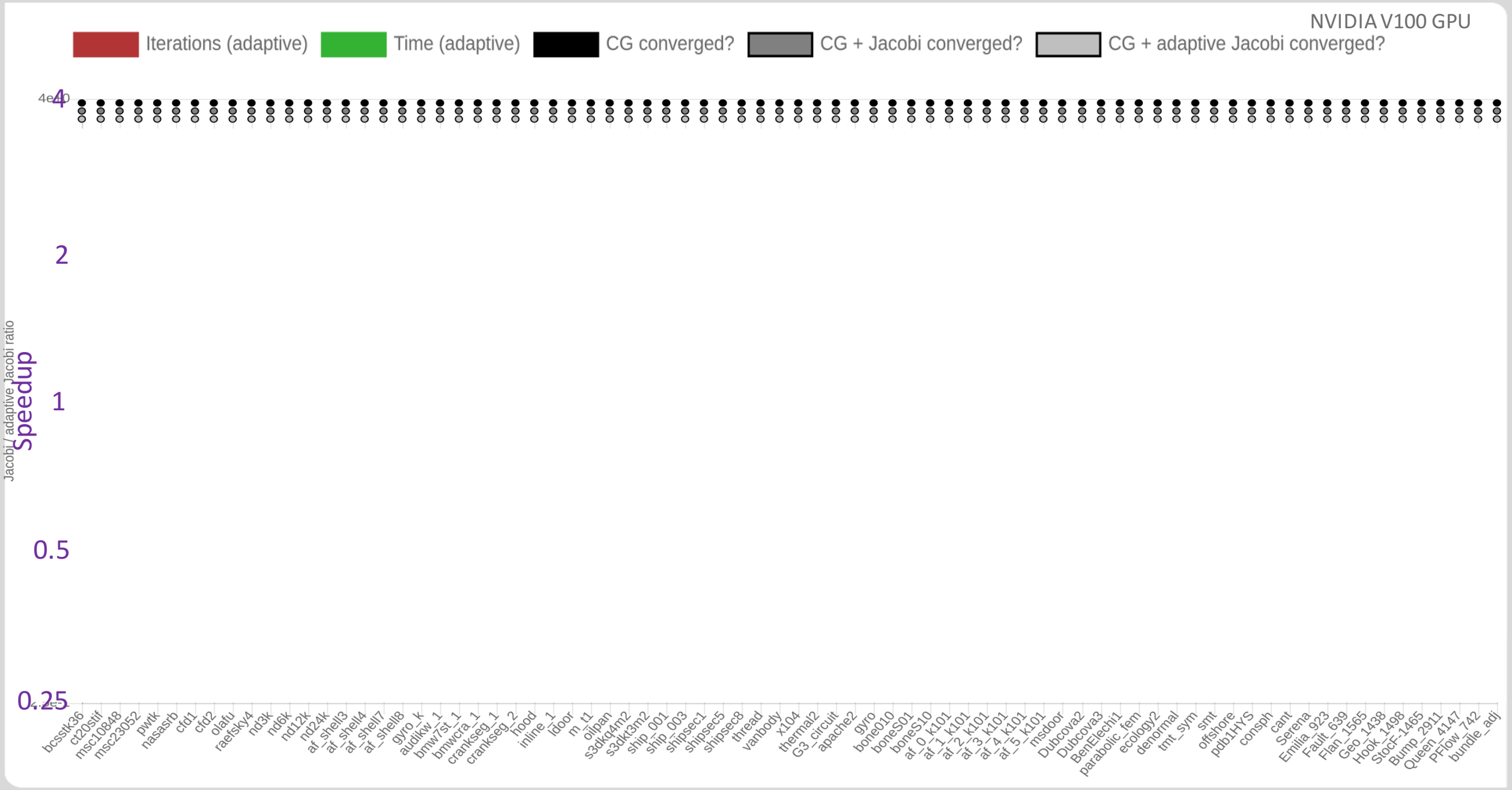
NVIDIA V100 GPU

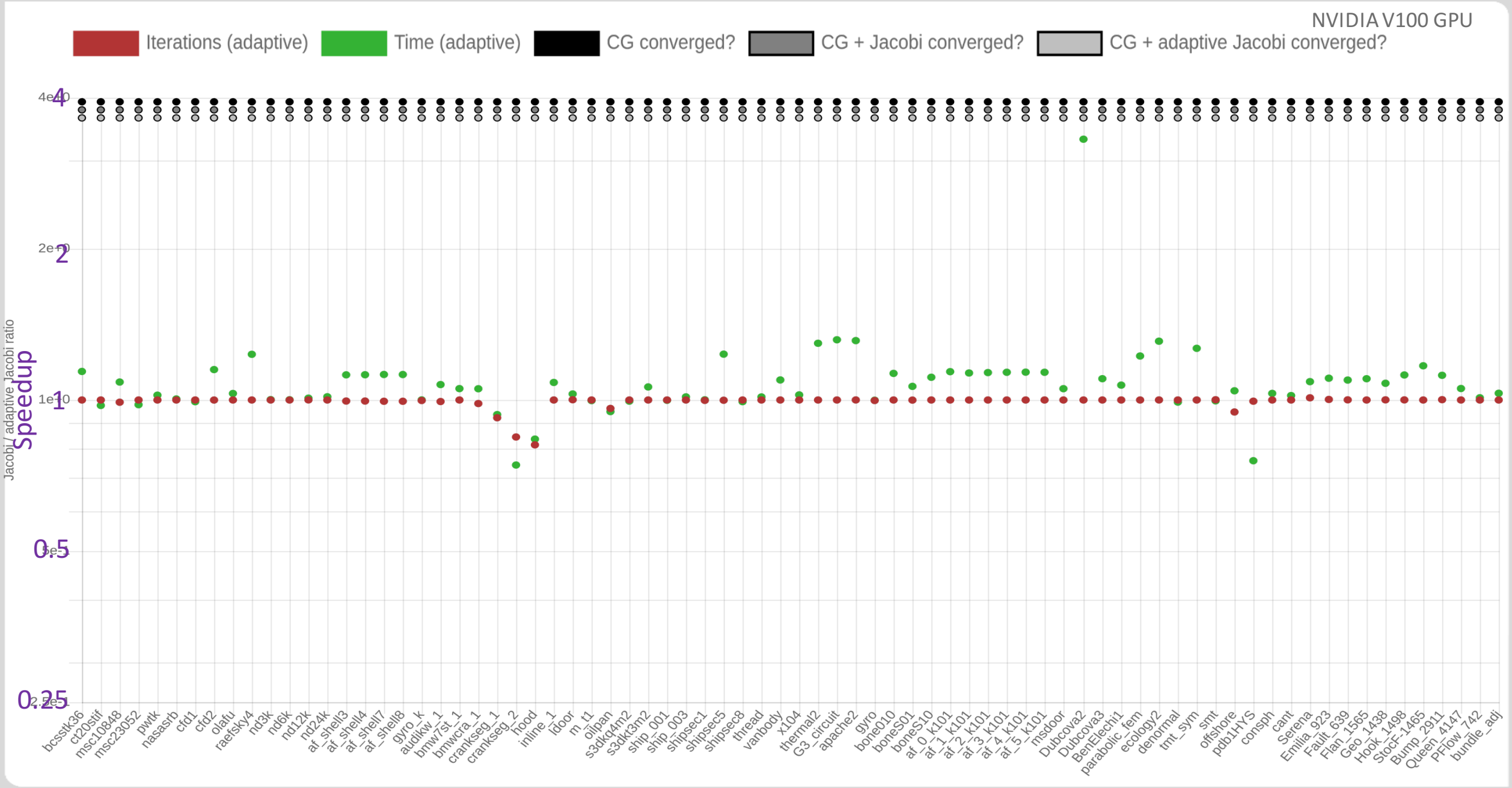


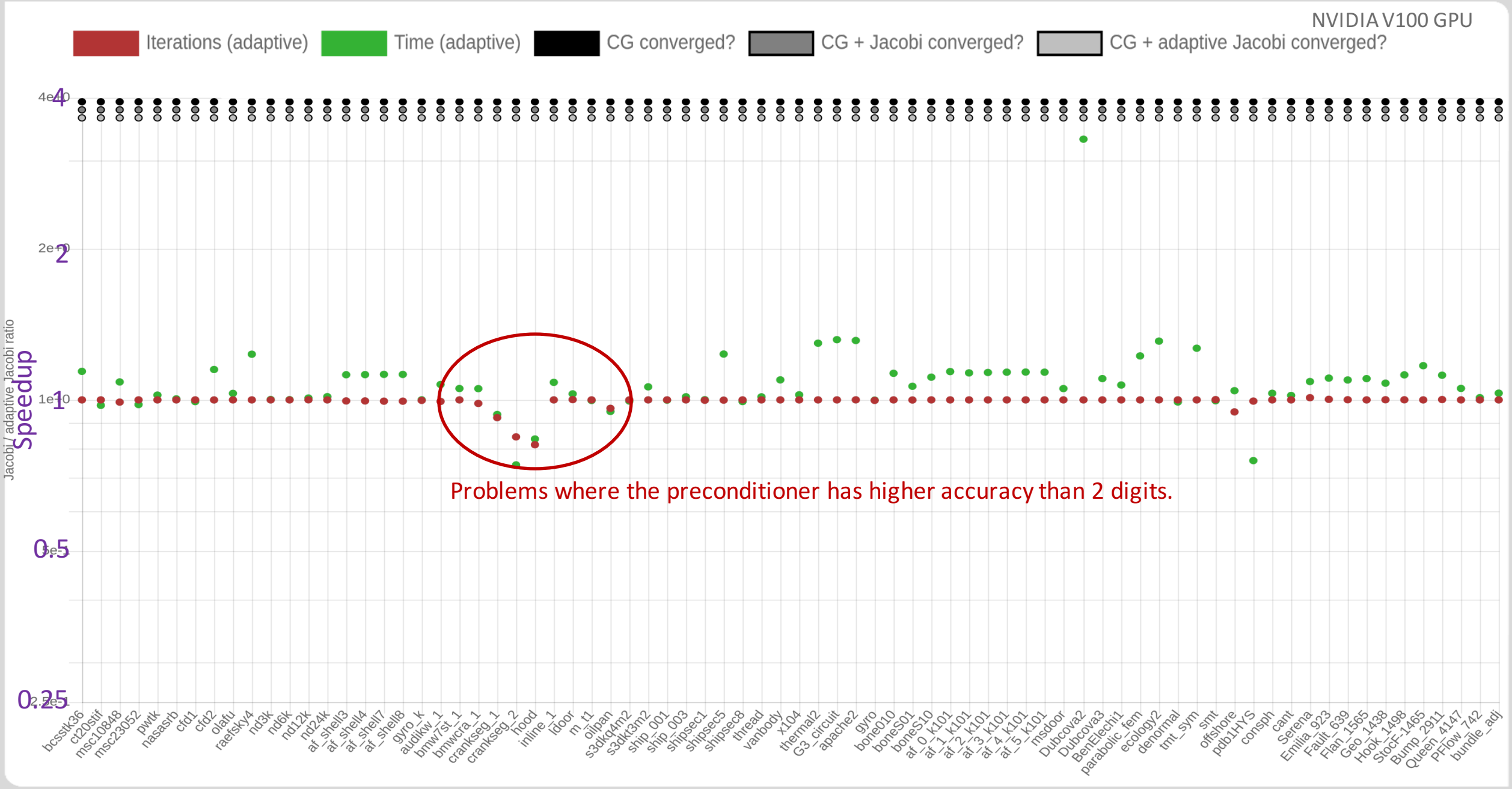
Adaptive Block-Jacobi Application

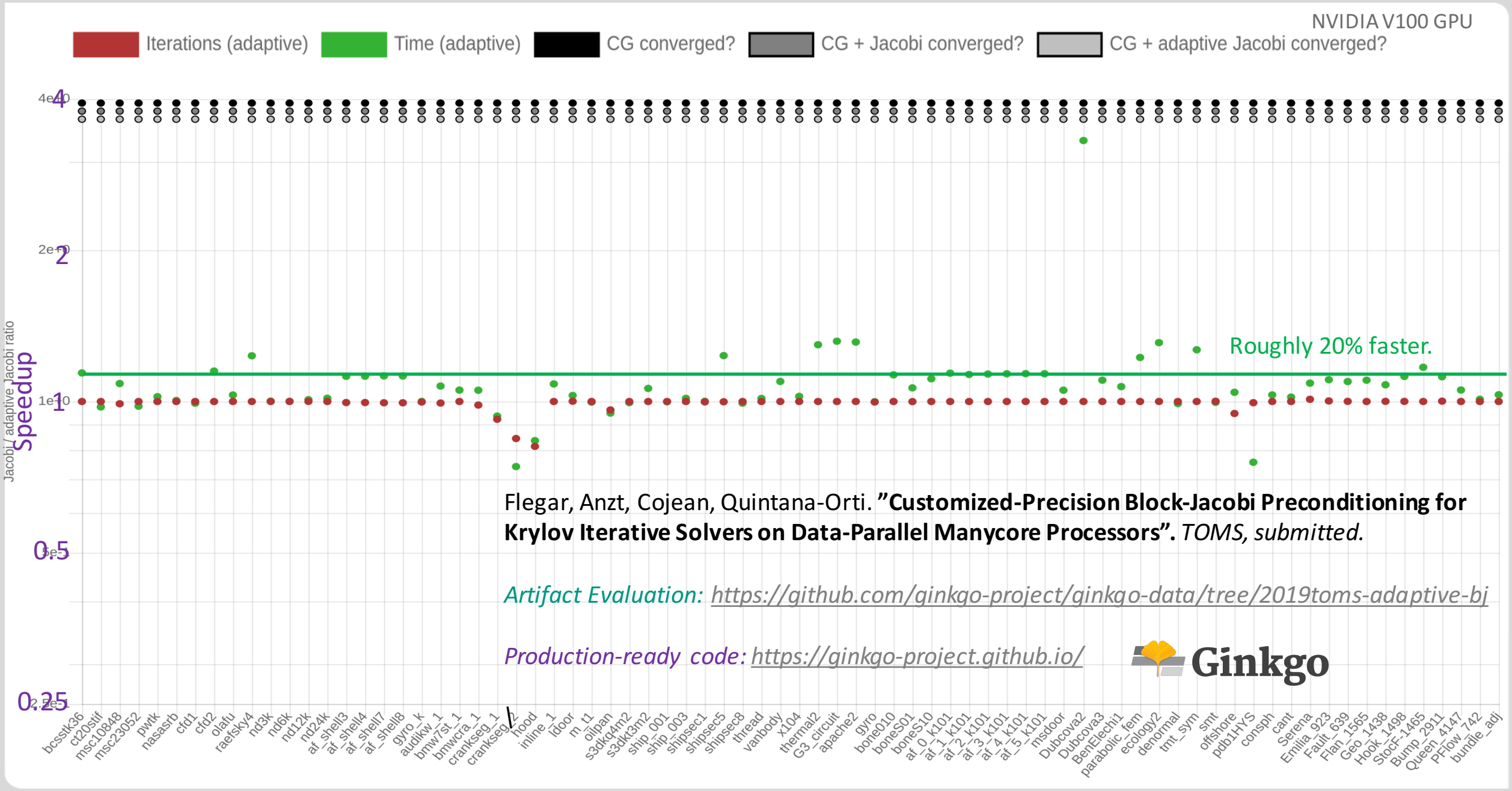
NVIDIA V100 GPU




NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
4e+00
4
2
1
0.5
0.25
Speedup
Jacobi/ adaptive Jacobi ratio
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
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

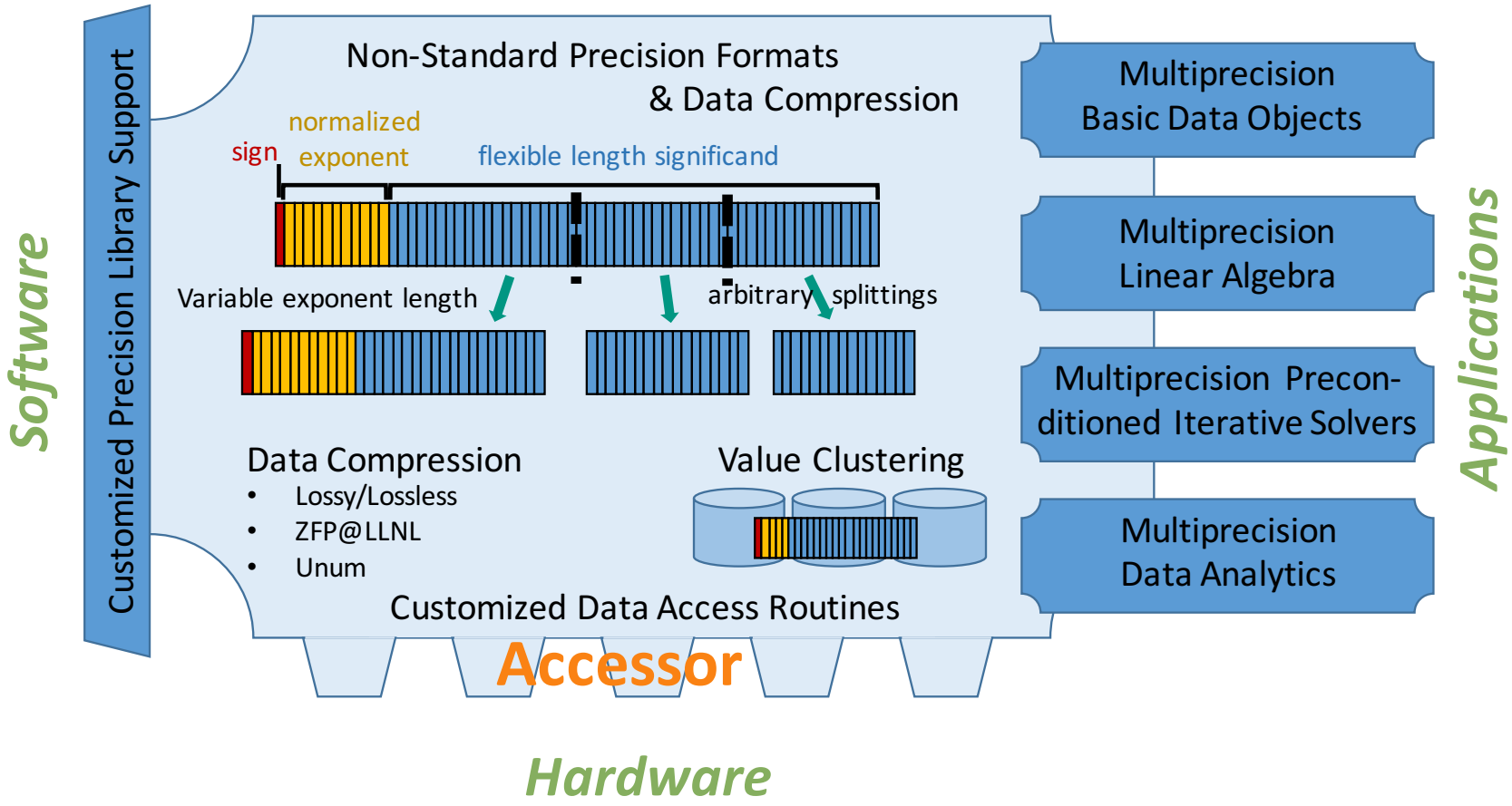

NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
Speedup
Jacobi/ adaptive Jacobi ratio
4
2
1
0.5
0.25
Problem
bcsstk36
ct20stif
msc10848
msc23052
pwtk
nasasrb
cfd1
cfd2
olafu
raefsky4
nd3k
nd6k
nd12k
nd24k
af_shell3
af_shell4
af_shell7
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


NVIDIA V100 GPU
Iterations (adaptive)
Time (adaptive)
CG converged?
CG + Jacobi converged?
CG + adaptive Jacobi converged?
Speedup
Jacobi/ adaptive Jacobi ratio
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
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?
Speedup
Jacobi/ adaptive Jacobi ratio
4
2
1
0.5
0.25
Roughly 20% faster.
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
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
Problem

Towards a Modular Precision Ecosystem

- Creating a **Modular Precision Ecosystem** inside Ginkgo .



Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Core

Library Infrastructure
Algorithm Implementations
- Iterative Solvers
- Preconditioners
- …

Architecture-specific kernels execute the algorithm on target architecture;

Kernels

- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Core

Library Infrastructure
Algorithm Implementations
- Iterative Solvers
- Preconditioners
- …

Architecture-specific kernels execute the algorithm on target architecture;

Kernels

Reference

Reference kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

Reference are sequential kernels to check correctness of algorithm design and optimized kernels;

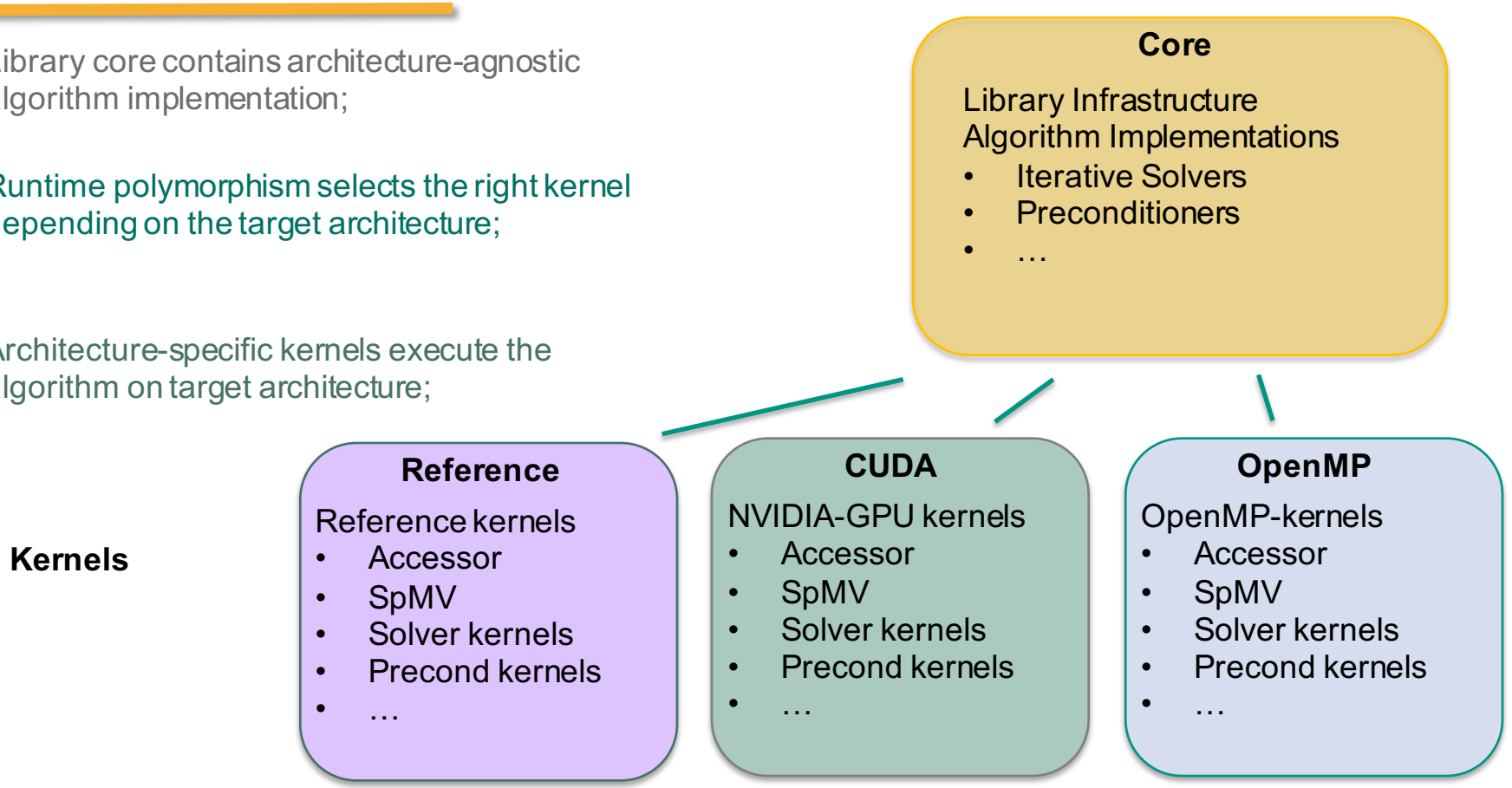
Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Runtime polymorphism selects the right kernel depending on the target architecture;

Architecture-specific kernels execute the algorithm on target architecture;



Reference are sequential kernels to check correctness of algorithm design and optimized kernels;

Optimized architecture-specific kernels;

Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Runtime polymorphism selects the right kernel depending on the target architecture;

Architecture-specific kernels execute the algorithm on target architecture;

Core

Library Infrastructure
Algorithm Implementations
- Iterative Solvers
- Preconditioners
- …

Kernels

Reference

Reference kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

CUDA

NVIDIA-GPU kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

OpenMP

OpenMP-kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

Reference are sequential kernels to check correctness of algorithm design and optimized kernels;

Optimized architecture-specific kernels;

Core Concept: Separate Algorithm from Kernels

Ginkgo

Library core contains architecture-agnostic algorithm implementation;

Runtime polymorphism selects the right kernel depending on the target architecture;

Architecture-specific kernels execute the algorithm on target architecture;

Core

Library Infrastructure
Algorithm Implementations
- Iterative Solvers
- Preconditioners
- …

Kernels

Reference

Reference kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

CUDA

NVIDIA-GPU kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

OpenMP

OpenMP-kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

HIP

AMD-GPU kernels
- Accessor
- SpMV

Multi-GPU

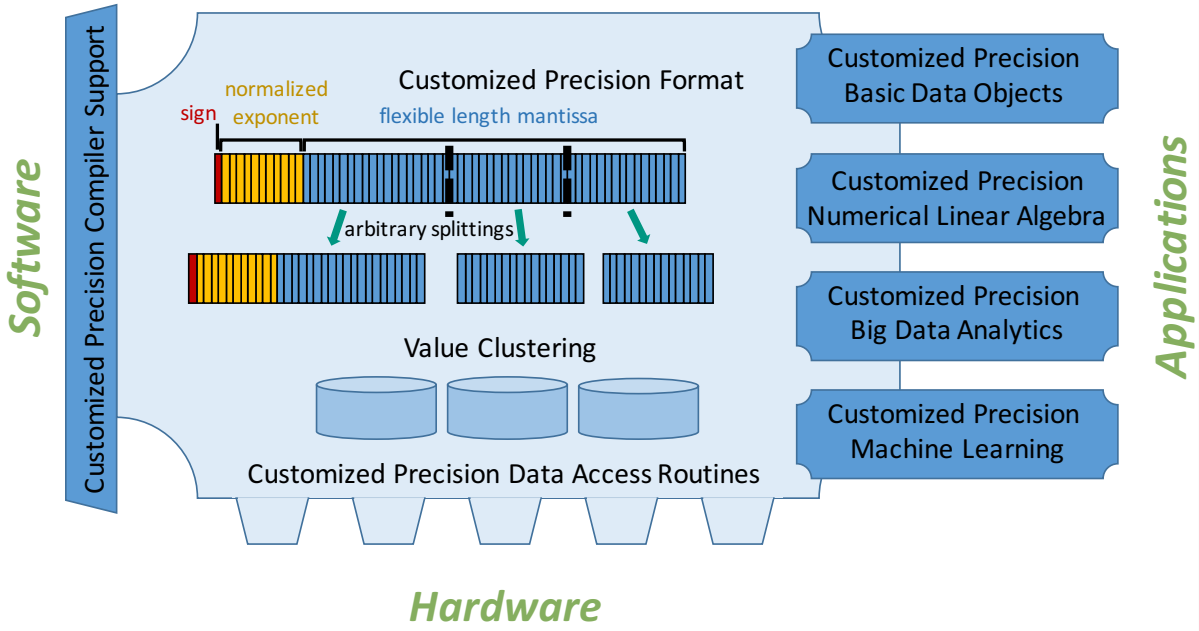
NVIDIA-GPU kernels
- Accessor
- SpMV
- Solver kernels
- Precond kernels
- …

Reference are sequential kernels to check correctness of algorithm design and optimized kernels;

Optimized architecture-specific kernels;

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.