ICL Friday Lunch Talk
Sept. 1st, 2017

Adaptive Precision in Block-Jacobi Preconditioning for Iterative Sparse Linear System Solvers

Hartwig Anzt, Jack Dongarra, Goran Flegar, Enrique S. Quintana-Orti

http://www.icl.utk.edu/~hanzt/talks/paco2017_anzt.pdf
Concurrency & Computation, Practice & Experience (CCPE), 2017

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Karlsruhe Institute of Technology

Motivation: We have to focus on data movement and memory!

Operation	approximate energy cost
DP floating point multiply-add	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

*John Shalf (LBNL)
VECPAR 2010

Motivation: We have to focus on data movement and memory!

Operation	approximate energy cost
DP floating point multiply-add	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

48x

*John Shalf (LBNL)
VECPAR 2010

Motivation: We have to focus on data movement and memory!

Which are the communication-intensive algorithms?

- Sorting
- Graph Algorithms
- Sparse Linear Algebra
  - Sparse Direct Solvers *(1D problems, 2D problems)**
  - Sparse Iterative Solvers *(2D problems on manycore, 3D problems)**
    - Multigrid , Fast Multipole Method (FMM), Krylov Methods …
    - Preconditioning: Incomplete LU, (block-) Jacobi, Gauss-Seidel...

*Mike Heroux (SNL)

Motivation: We have to focus on data movement and memory!

Which are the communication-intensive algorithms?

- Sorting
- Graph Algorithms
- Sparse Linear Algebra
  - Sparse Direct Solvers *(1D problems, 2D problems)**
  - Sparse Iterative Solvers *(2D problems on manycore, 3D problems)**
    - Multigrid , Fast Multipole Method (FMM), Krylov Methods …
    - Preconditioning: Incomplete LU, (block-) Jacobi, Gauss-Seidel...

*Mike Heroux (SNL)

Block-Jacobi Preconditioning

- **Jacobi method** based on **diagonal scaling:** $P = diag(A)$
  - Can be used as iterative solver:

$$x^{(k+1)} = x^{(k)} + P^{-1}b - P^{-1}Ax^{(k)}$$

  - Can be used as preconditioner: $\tilde{A} = P^{-1}A$, $\tilde{b} = P^{-1}b$

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

Block-Jacobi Preconditioning

- **Jacobi method** based on **diagonal scaling:** $P = diag(A)$
  - Can be used as iterative solver:

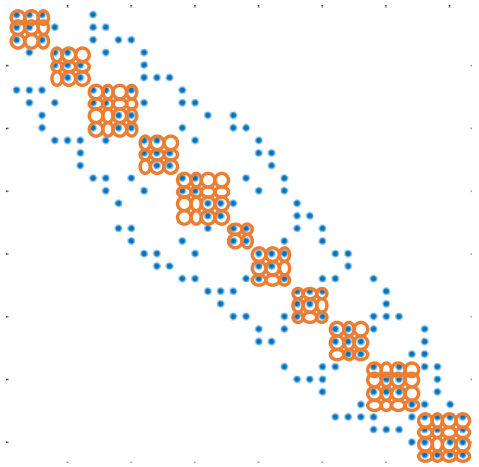
$$x^{(k+1)} = x^{(k)} + P^{-1}b - P^{-1}Ax^{(k)}$$

  - Can be used as preconditioner: $\tilde{A} = P^{-1}A$, $\tilde{b} = P^{-1}b$

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

- **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.

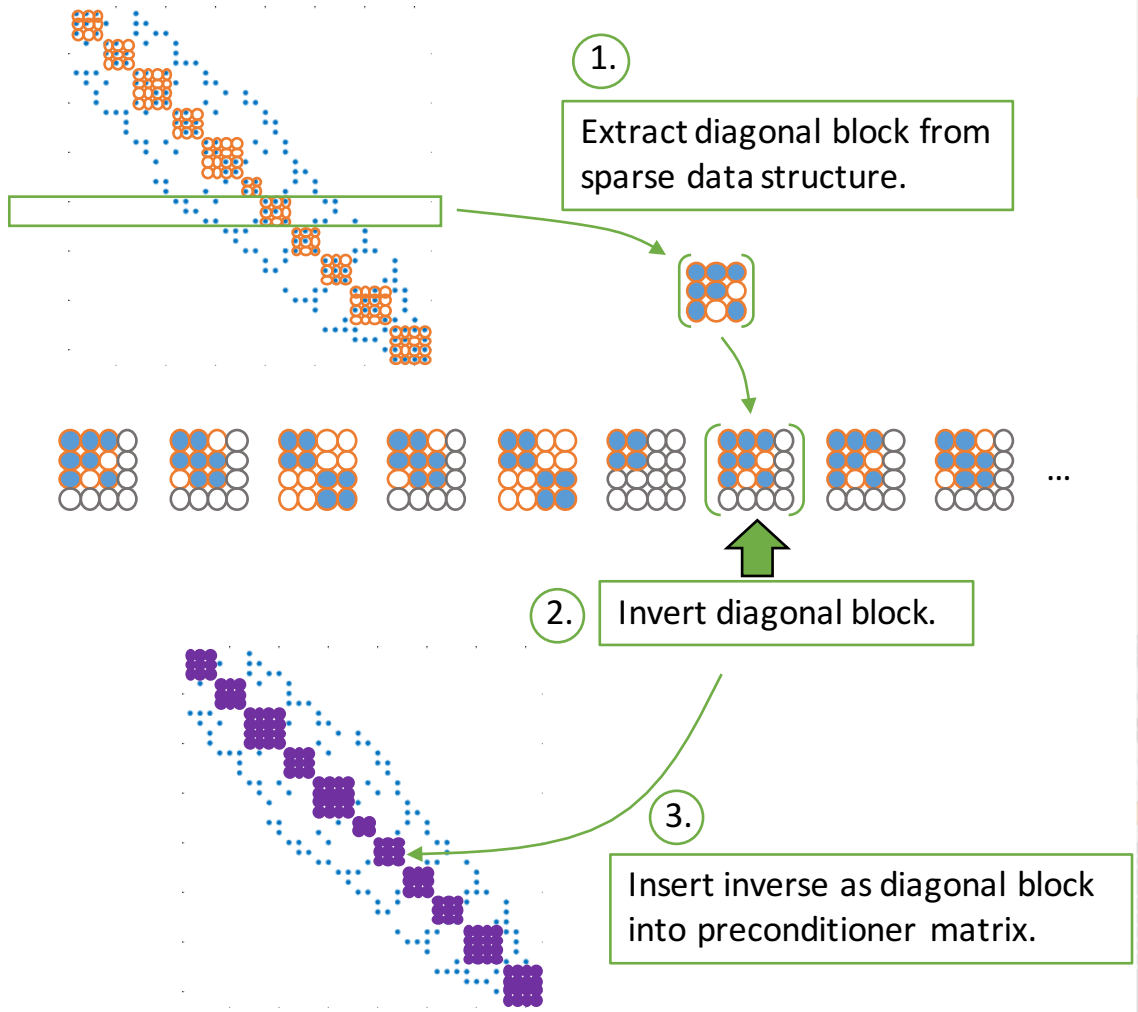


Inversion-based Block-Jacobi Preconditioning

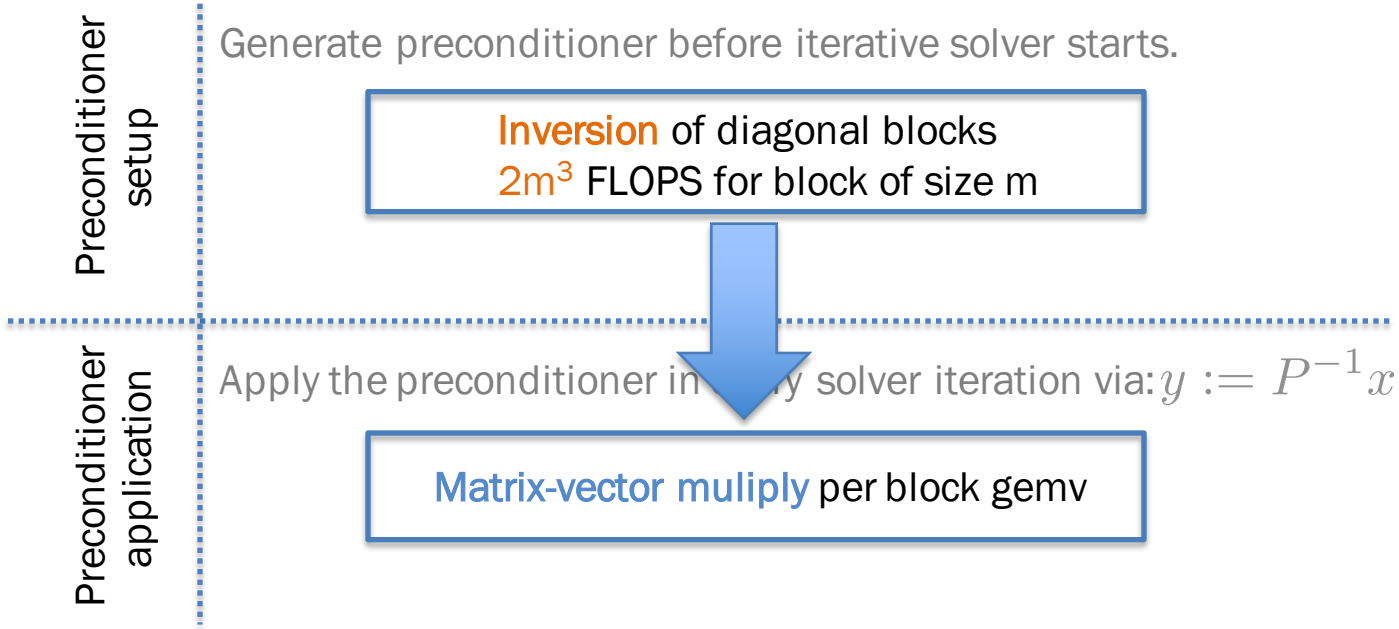
Preconditioner setup

Generate preconditioner before iterative solver starts.

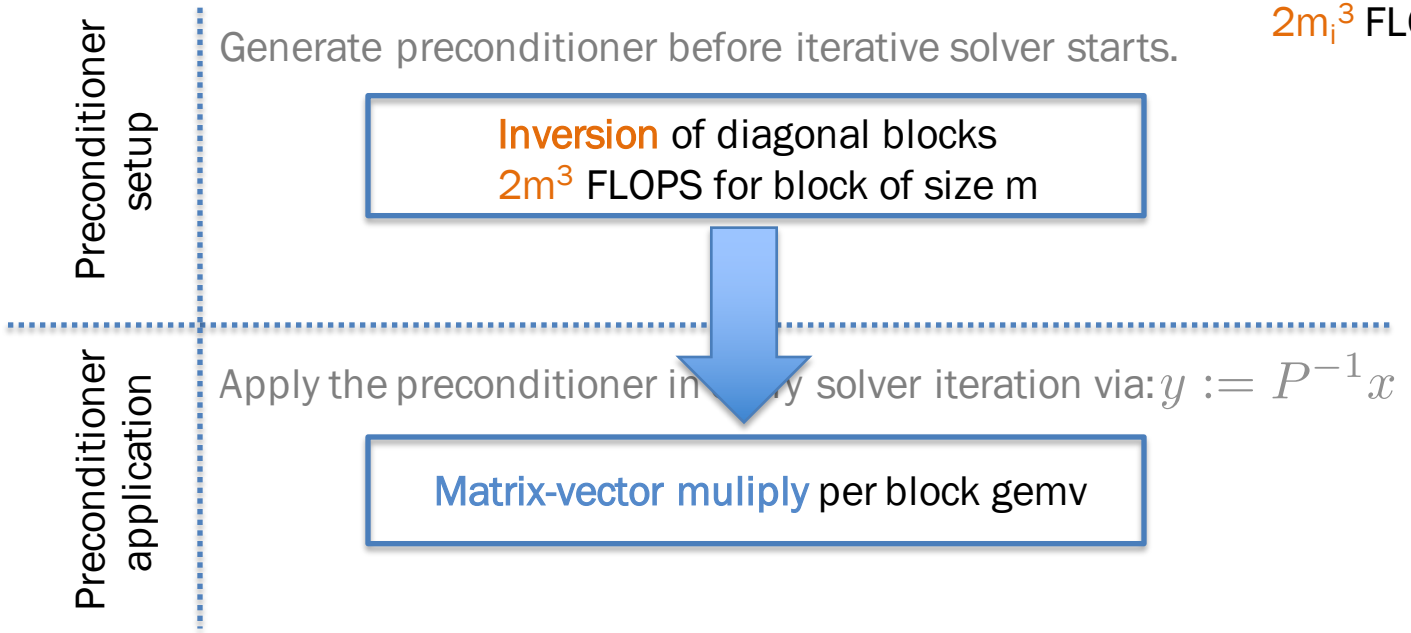
Inversion of diagonal blocks
$2m^3$ FLOPS for block of size m



Inversion-based Block-Jacobi Preconditioning



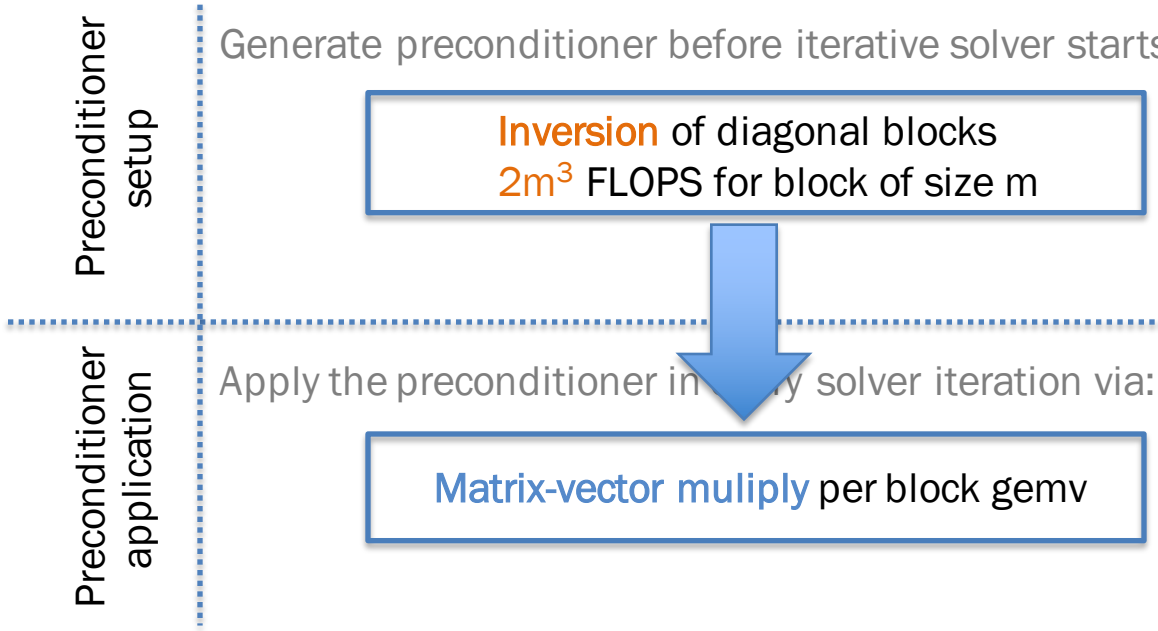
Inversion-based Block-Jacobi Preconditioning



- Cost of Inversion:
  $2m_i^3$ FLOPS for block of size m_i.

$$\sum_{blocks} 2m_i^3$$

Inversion-based Block-Jacobi Preconditioning



- Cost of **Inversion:**
  $2m_i^3$ FLOPS for block of size m_i. $\sum_{blocks} 2m_i^3$
- Cost of Preconditioner application:
  m_i^2 FLOPS for block of size m_i : $\sum_{blocks} 2m_i^2$
- Total memory consumption: $\sum_{blocks} m_i^2$

Inversion-based Block-Jacobi Preconditioning

Preconditioner setup

Generate preconditioner before iterative solver starts.

Inversion of diagonal blocks
$2m^3$ FLOPS for block of size m

Preconditioner application

Apply the preconditioner in every solver iteration via: $y := P^{-1}x$

Matrix-vector muliply per block gemv

- Cost of **Inversion**:
  $2m_i^3$ FLOPS for block of size m_i.
  $$\sum_{blocks} 2m_i^3$$
- Cost of Preconditioner application:
  m_i^2 FLOPS for block of size m_i :
  $$\sum_{blocks} 2m_i^2$$
- Total memory consumption:
  $$\sum_{blocks} m_i^2$$

- **Energy balance** for one **preconditioner application** (DP)*:

$$\sum_{blocks} m_i^2 \cdot \underbrace{(200}_{\text{computation}} + \underbrace{4800)}_{\text{data read}}$$

*John Shalf (LBNL)

Mixed Precision Block-Jacobi Preconditioning

Mixed Precision Idea:

- Do all calculations in working precision
- Preconditioner is only an approximation!
- Store the block-Jacobi matrix in reduced precision
  - Benefit from faster data access
  - Benefit from reduced data read cost

Implications:

- Reduced preconditioner quality
  - Need for a flexible variant (FCG)
  - Need for more Krylov solver iterations
  - Potential loss of regularity (breakdown)

- Cost of Inversion:
  $2m_i^3$ FLOPS for block of size m_i.
  $$\sum_{blocks} 2m_i^3$$
- Cost of Preconditioner application:
  m_i^2 FLOPS for block of size m_i :
  $$\sum_{blocks} 2m_i^2$$
- Total memory consumption:
  $$\sum_{blocks} m_i^2$$

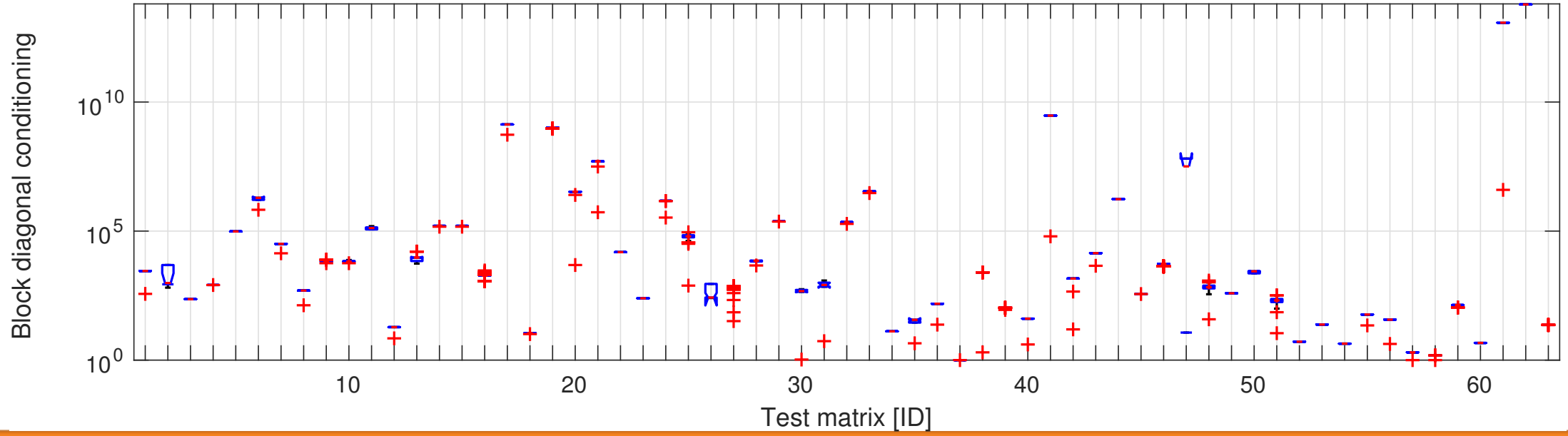
- Energy balance for one preconditioner application (DP)*:

$$\sum_{blocks} m_i^2 \cdot \underbrace{(200}_{\text{computation}} + \underbrace{4800)}_{\text{data read}}$$

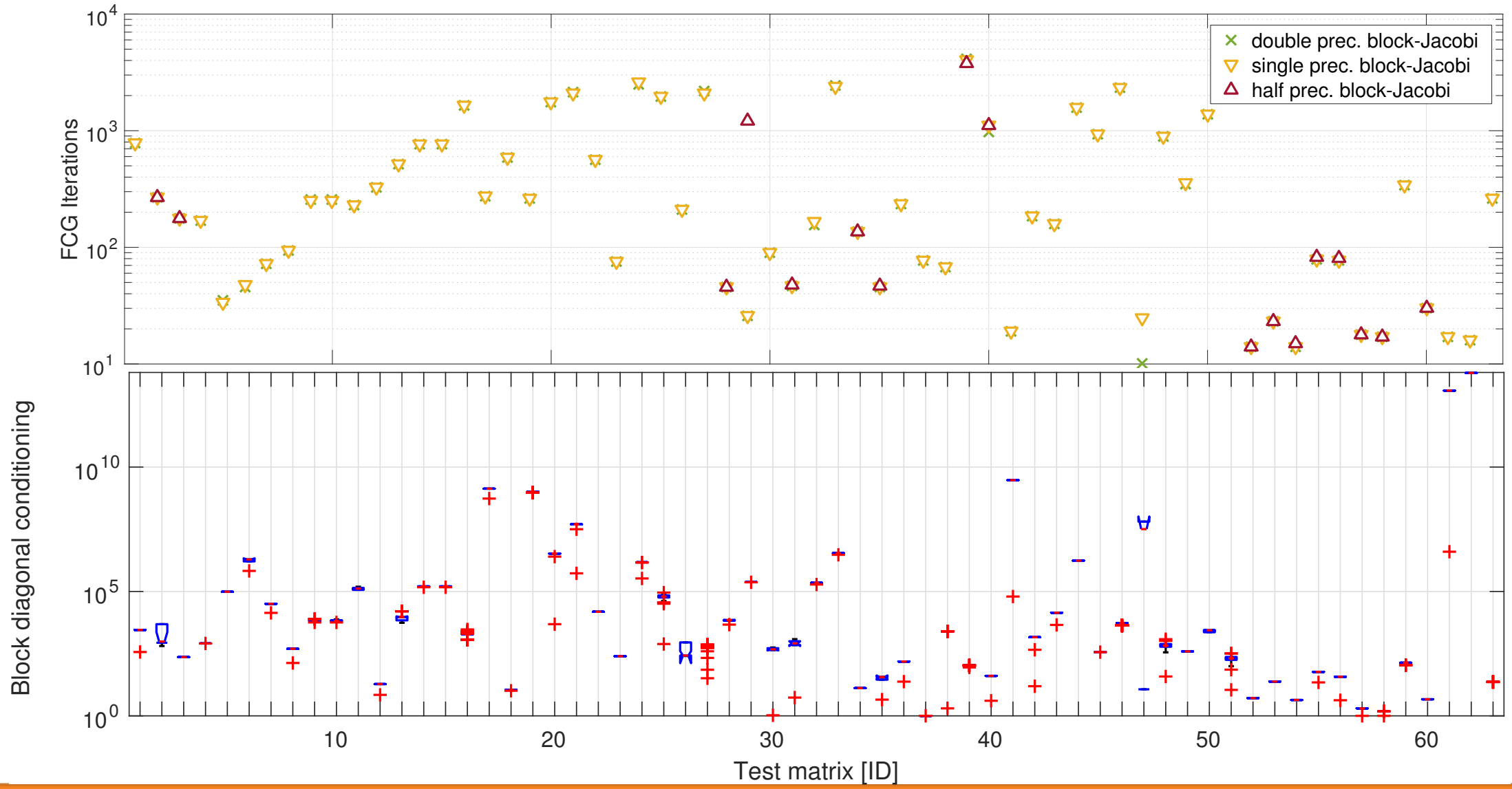
*John Shalf (LBNL)

Mixed Precision Block-Jacobi Preconditioning

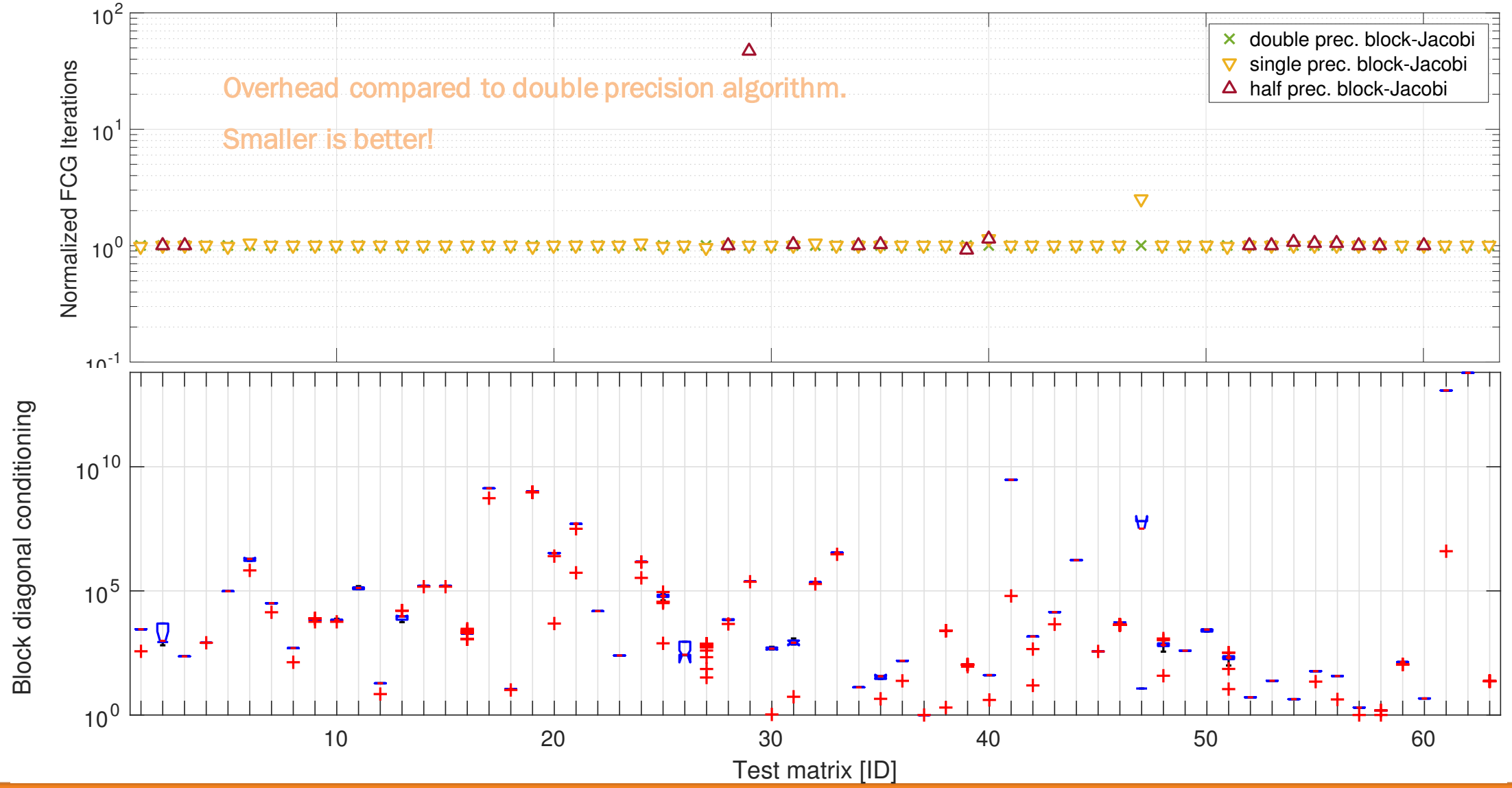
- 63 matrices from the SuiteSparse Matrix Collection
- Use block-size 24 with Super-Variable agglomeration (24 is upper bound for size of blocks)
- Report conditioning of all arising diagonal blocks
- Analyze the impact on a top-level flexible Conjugate Gradient solver (CG)



Mixed Precision Block-Jacobi Preconditioning



Mixed Precision Block-Jacobi Preconditioning

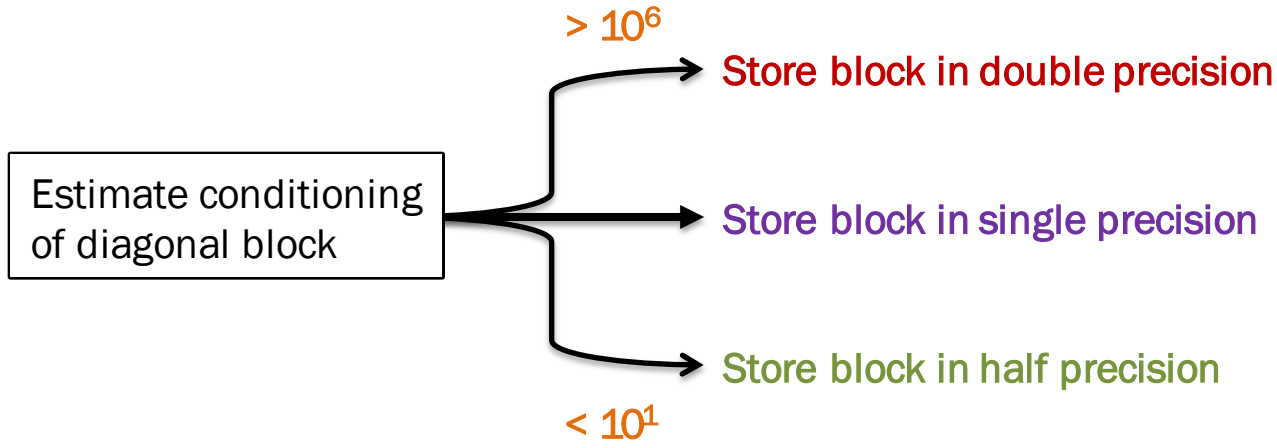


Adaptive Precision Block-Jacobi Preconditioning

Adaptive Precision Idea:

- All computations use double precision!

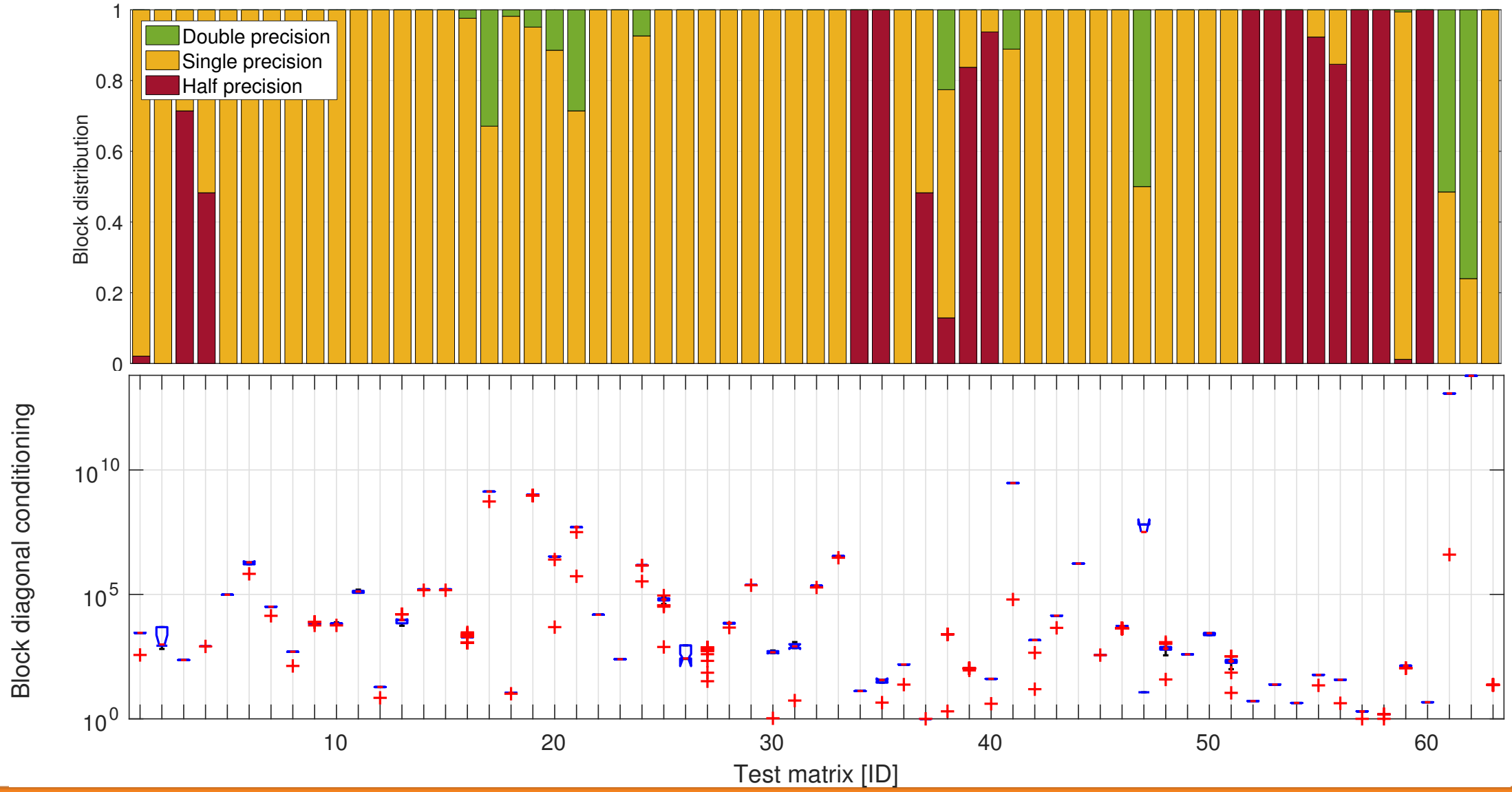
- Store distinct blocks in different formats
- Use single precision as standard storage format
- Where necessary: switch to double
- For well-conditioned blocks use half precision
- Care about under/overflow



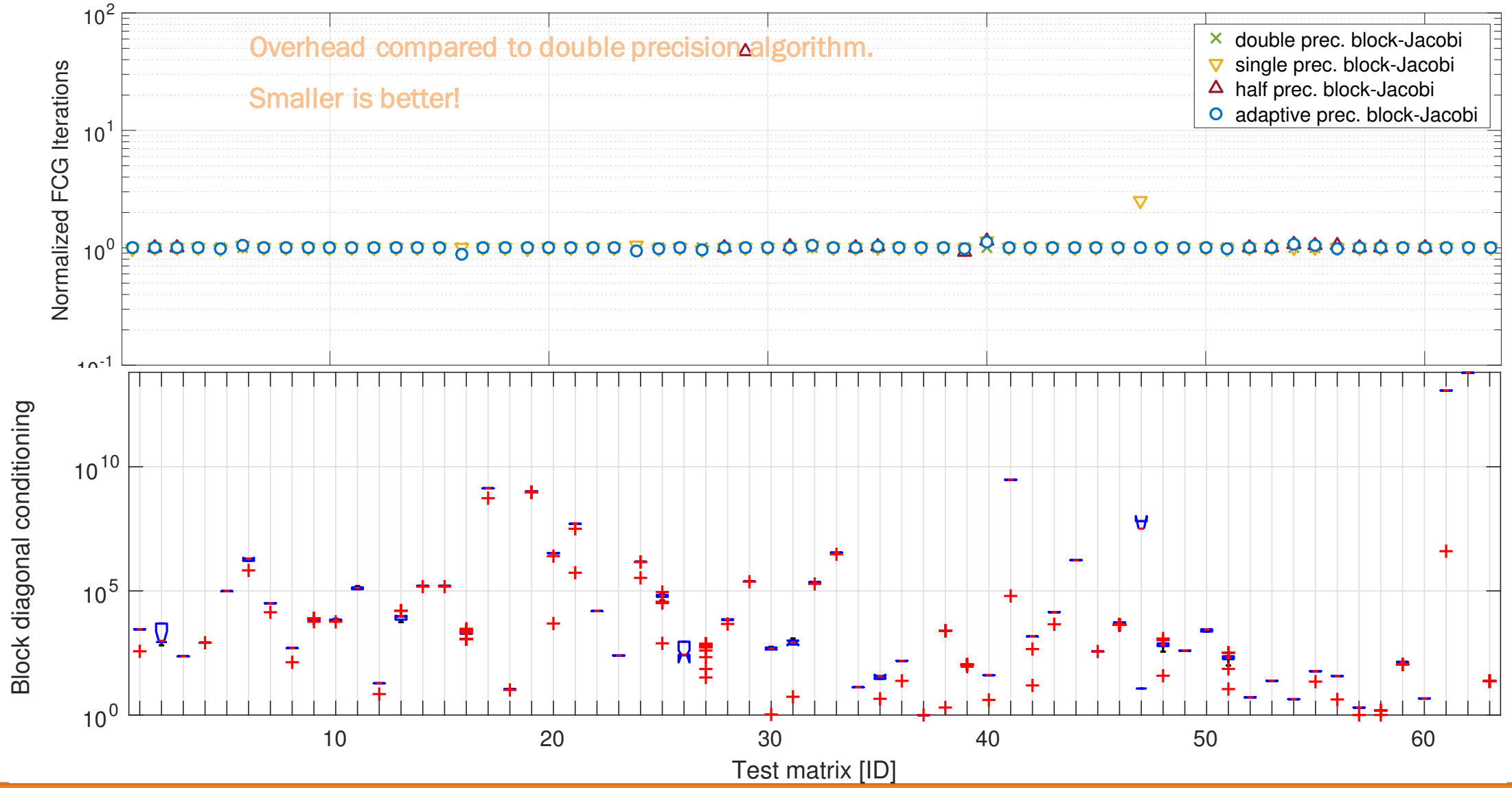
Block-Jacobi CG

- Ax=b with x:=0 and b:=A*1
- Relative residual stopping crit. 1e-9

Mixed Precision Block-Jacobi Preconditioning



Mixed Precision Block-Jacobi Preconditioning

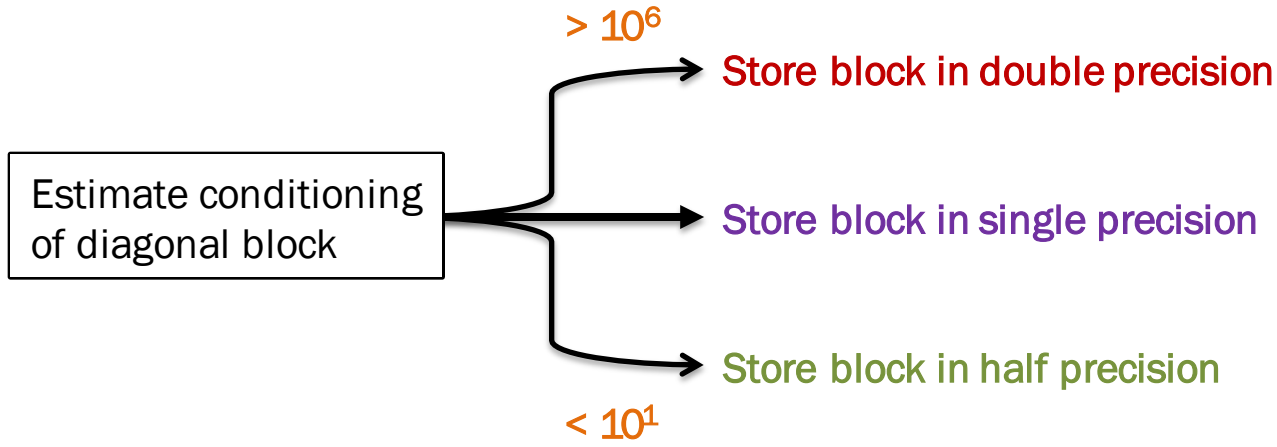


Adaptive Precision Block-Jacobi Preconditioning

Adaptive 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
- Care about under/overflow



Operation	approximate energy cost
DP floating point multiply-add	100 pJ
DP DRAM read-to-register	4800 pJ
DP word transmit-to-neighbor	7500 pJ
DP word transmit-across-system	9000 pJ

*John Shalf (LBNL)

Energy model:

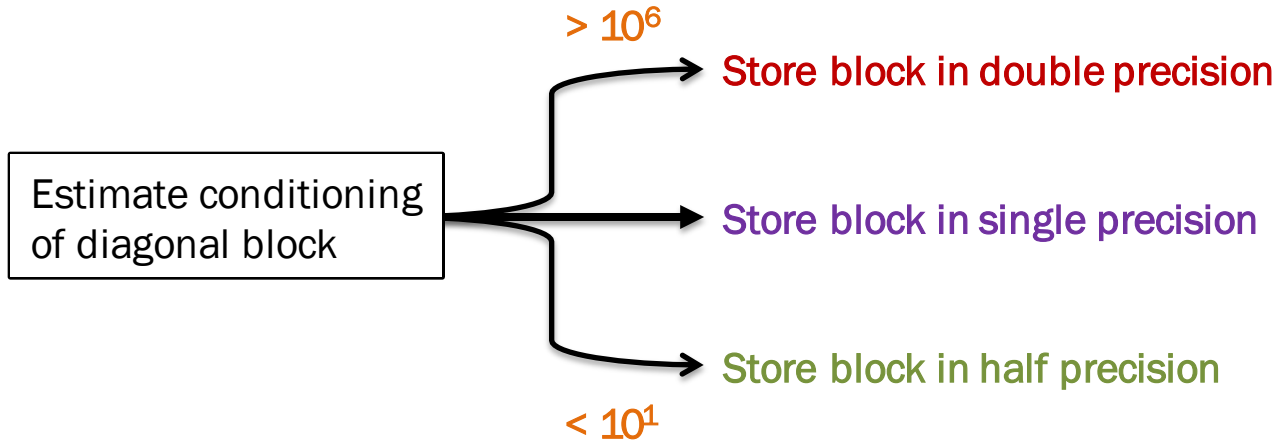
- 4800 pJ for double precision (64 bit)
- 2400 pJ for single precision / integers (32 bit)
- 1200 pJ for half precision (16 bit)

Adaptive Precision Block-Jacobi Preconditioning

Adaptive 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
- Care about under/overflow



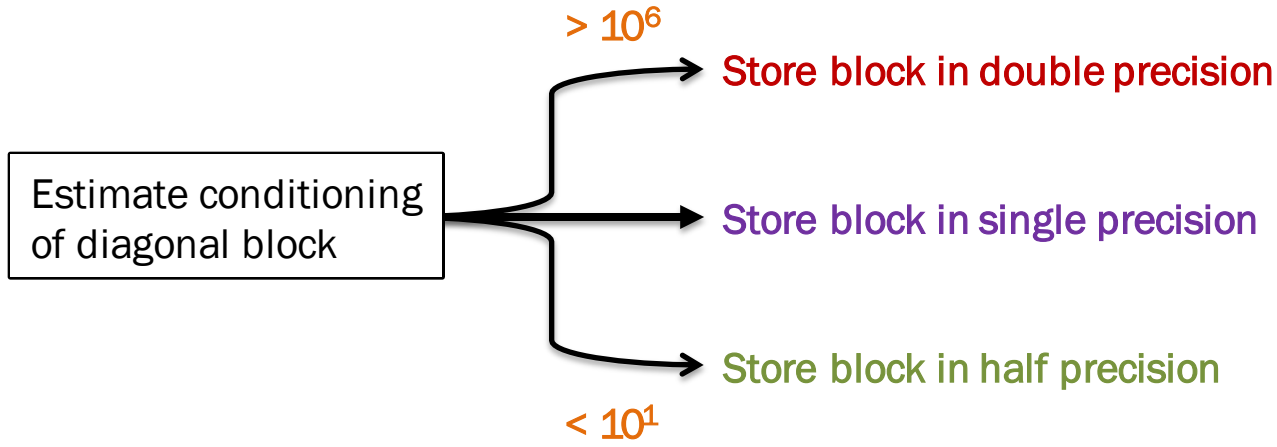
How much data we need to read/write in a Conjugate Gradient (CG) loop:

Operation	Memory volume	# per FCG loop	Energy est.
CSR-SpMV	nz double + nz int + n int + 2n double	1	(nz + 2n) * 4800 pJ + (nz + n) * 2400 pJ
axpy	3n double	5	9n * 4800 pJ
Dot / nrm	2n double / n double	4 + 1	9n * 4800 pJ
preconditioner	$\sum_{blocks} m_i^2$ [used format]	1	$\sum_{blocks} m_i^2$ * ?

Adaptive Precision Block-Jacobi Preconditioning

Adaptive 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
- Care about under/overflow



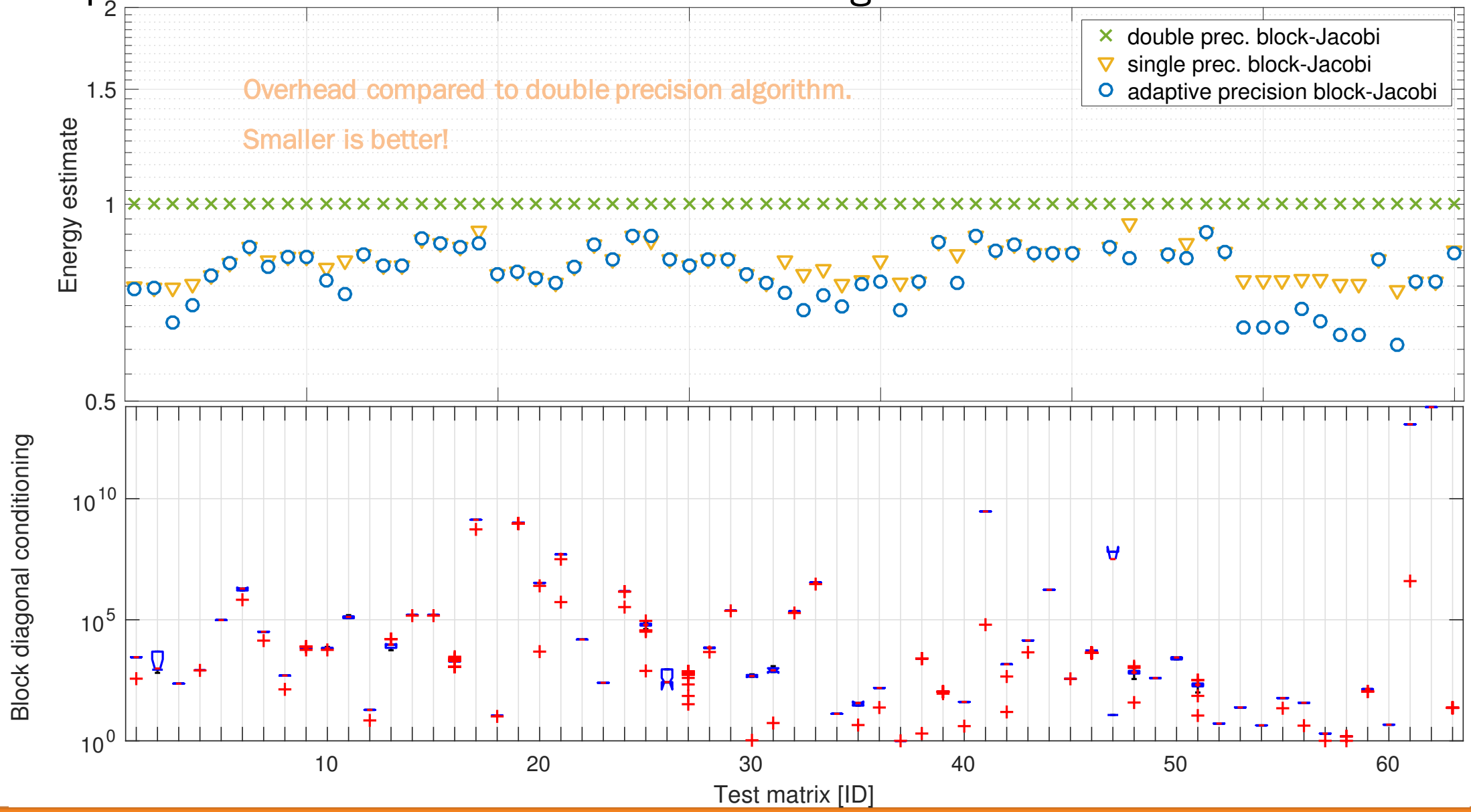
Energy model for block-Jacobi-CG

- We ignore computational cost, only memory access
- No data (matrix / vector) is cached, only DRAM reads
- FCG outer solver (in DP)
- Adaptive precision for the block-Jacobi preconditioner

Block-Jacobi CG

- Ax=b with x:=0 and b:=A*1
- Relative residual stopping crit. 1e-9

Adaptive Precision Block-Jacobi Preconditioning



Interested in working on numerical linear algebra…?

Research Position in Numerical Linear Algebra

Full Time - The Innovative Computing Lab (ICL, http://icl.cs.utk.edu/) of the University of Tennessee is looking for a bright, motivated person(s) to join as a staff researcher. The primary duties of this position are: to assist in the design, development and maintenance of numerical software libraries for solving linear algebra problems on large distributed memory machines with multicore processors and hardware accelerators; write research papers documenting research findings; present material at conferences; direct students and research team in their research endeavors as related to ongoing and future projects. There will be opportunities for publication, travel, and high profile professional networking in academia, labs, and industry.

Education

PhD in Computer Science or related field with demonstrable background in applied mathematics and computer science, in particular distributed computing, multicore computing, GPU computing; or MS in Computer Science or related field + 3-5 years relevant research or work experience.

Experience

Background in applied mathematics, technical experience in parallel computing, distributed computing, multithreading; familiarity with numerical software libraries; experience with performance diagnostics and optimization techniques, tracing, profiling. Experience with developing mathematical software desired. Track record of contributing to open source projects a plus. Experience with GPU computing highly desired.

Job Skills

Required: background in applied mathematics, familiarity with numerical software; extensive knowledge of programming techniques, highly proficient in C/C++, basic understanding of Fortran language; proficiency with MPI and OpenMP, familiarity with CUDA or OpenCL; technical writing and presentation skills, excellent communication skills and a strong publication record.

Salary starting at $60,000/year, depending on experience and qualifications. For consideration, send CV and contact information for three references to Tracy Rafferty at rafferty@icl.utk.edu.

rafferty@icl.utk.edu

Karlsruhe Institute of Technology

hartwig.anzt@kit.edu

http://www.icl.utk.edu/~hanzt/talks/paco2017_anzt.pdf