ICCS 2017, Zürich, Switzerland

Batched Generation of Block-Jacobi Preconditioners for Iterative Sparse Linear System Solvers on GPUs

Hartwig Anzt, Jack Dongarra, Goran Flegar, Enrique S. Quintana-Ortí, Andrés E. Tomás

flegar@uji.es

Problem setting

- Solve sparse linear system using an iterative Krylov method

$$Ax = b, \;\; A \in \mathbb{R}^{n \times n}$$

Problem setting

- Solve sparse linear system using an iterative Krylov method
- Convergence typically benefits from using a preconditioner

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

$$M^{-1}Ax = M^{-1}b$$

Problem setting

- Solve sparse linear system using an iterative Krylov method
- Convergence typically benefits from using a preconditioner
- Need high degree of parallelism to use a GPU effectively
  - 56 SMs x 64 cores = 3584 cores!
  - Oversubscribe to hide memory latency
- Use a preconditioner with high *parallelization potential*

$$Ax = b, \; A \in \mathbb{R}^{n \times n}$$

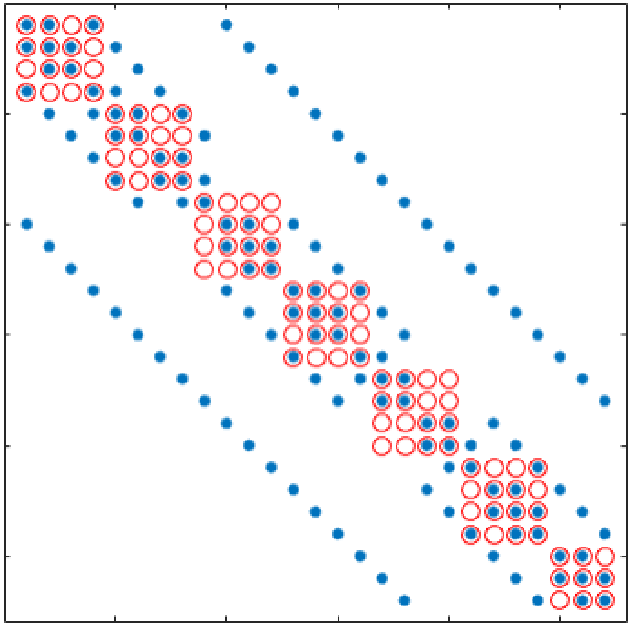
$$M^{-1}Ax = M^{-1}b$$

NVIDIA GP100

source: devblogs.nvidia.com/parallelforall/

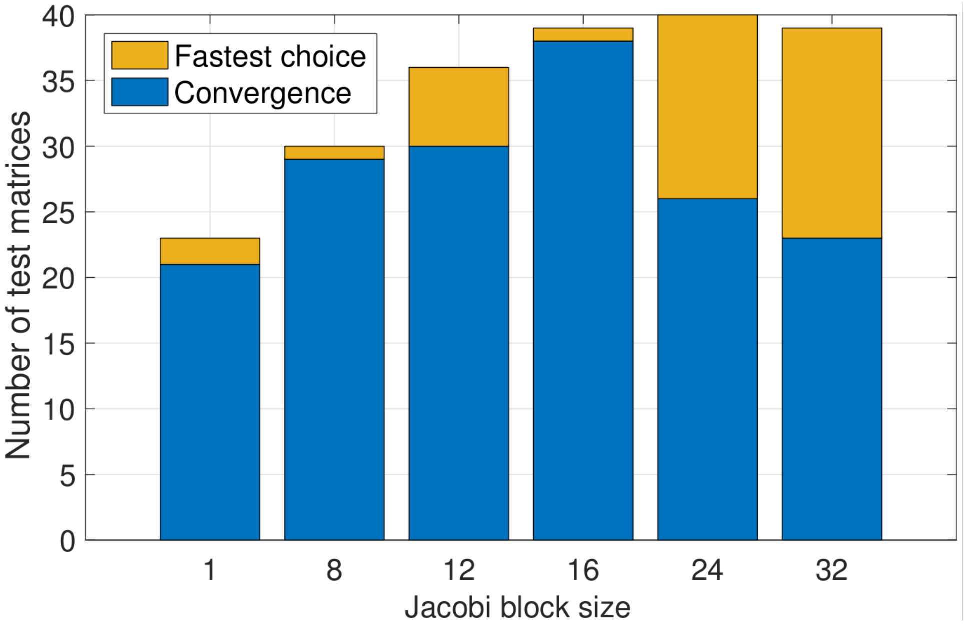
Block-Jacobi preconditioning

- Scalar Jacobi
  - Scale with inverse of main diagonal
- Block-Jacobi
  - Scale with inverses of diagonal blocks (possibly of different sizes!)
  - Can reflect the block structure of the problem
  - Often superior to scalar Jacobi
- Can process each block independently!



Benefits of block-Jacobi

- 40 matrices from SuiteSparse
- MAGMA-sparse open source library
  - IDR solver
  - Jacobi preconditioner
  - Supervariable blocking
- Block-Jacobi improves the robustness of the solver
  - More problems converge
- Decreases time-to-solution



General Ideas

- Restrict block size to 32x32
  - Large block sizes require more memory to store the preconditioner matrix

General Ideas

- Restrict block size to 32x32
  - Large block sizes require more memory to store the preconditioner matrix
- Use a single warp to process the whole block (one thread per column)
  - No need for explicit synchronization
- Use the large register file to store the entire block
  - Read/write from mem. once
  - Comm. via warp shuffles
  - Avoids load/store instructions

General Ideas

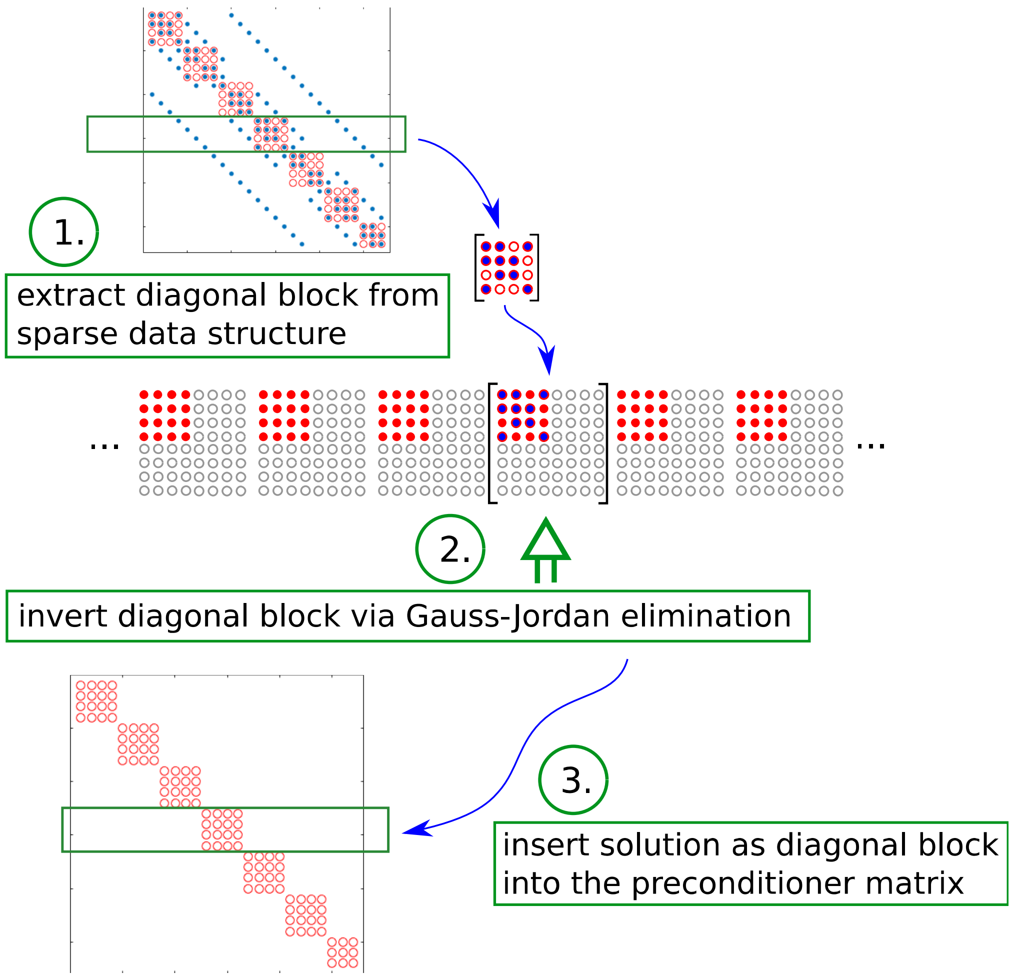
- Restrict block size to 32x32
  - Large block sizes require more memory to store the preconditioner matrix
- Use a single warp to process the whole block (one thread per column)
  - No need for explicit synchronization

General Ideas

- Restrict block size to 32x32
  - Large block sizes require more memory to store the preconditioner matrix
- Use a single warp to process the whole block (one thread per column)
  - No need for explicit synchronization
- Use the large register file to store the entire block
  - Read/write from mem. once
  - Comm. via warp shuffles
  - Avoids load/store instructions

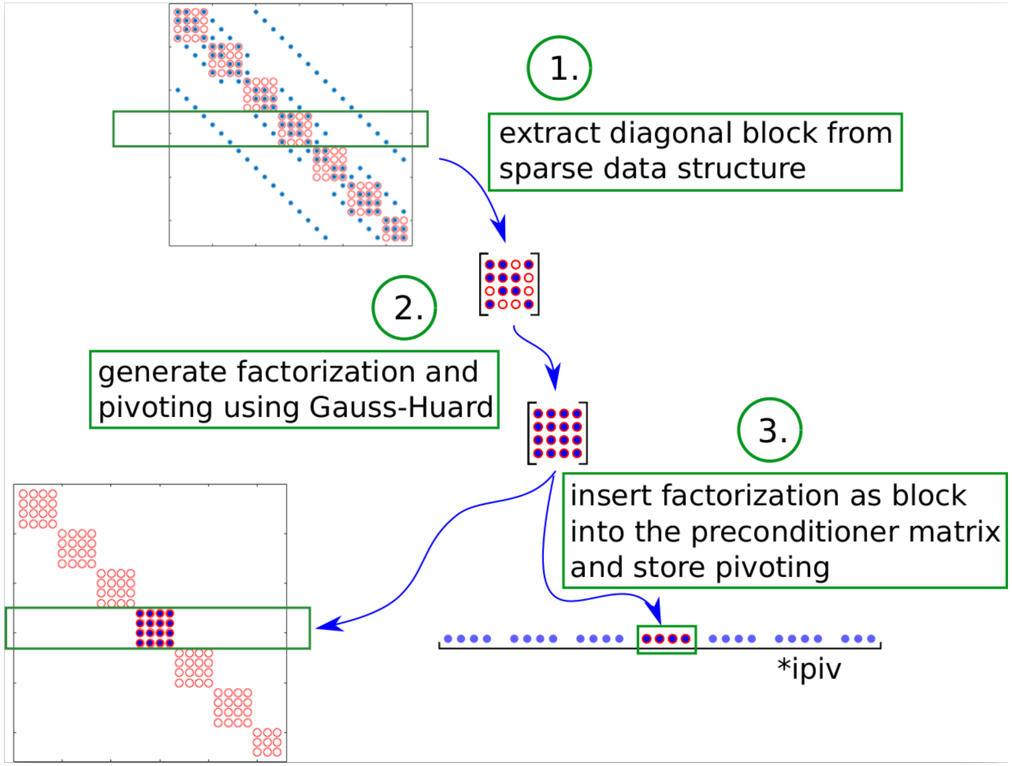
Implementation options

- Inversion in preconditioner setup + matrix-vector product in application
  - (FLOPS: $2n^3$ setup, $2n^2$ app.)
  - Batched Gauss-Jordan elimination (BGJE)
    - Each step consists of column scaling and a rank-1 update of the whole matrix
    - Easily achievable load balancing
  - H. Anzt et al., “Batched Gauss-Jordan Elimination for Block-Jacobi Preconditioner Generation on GPUs”, PMAM’17

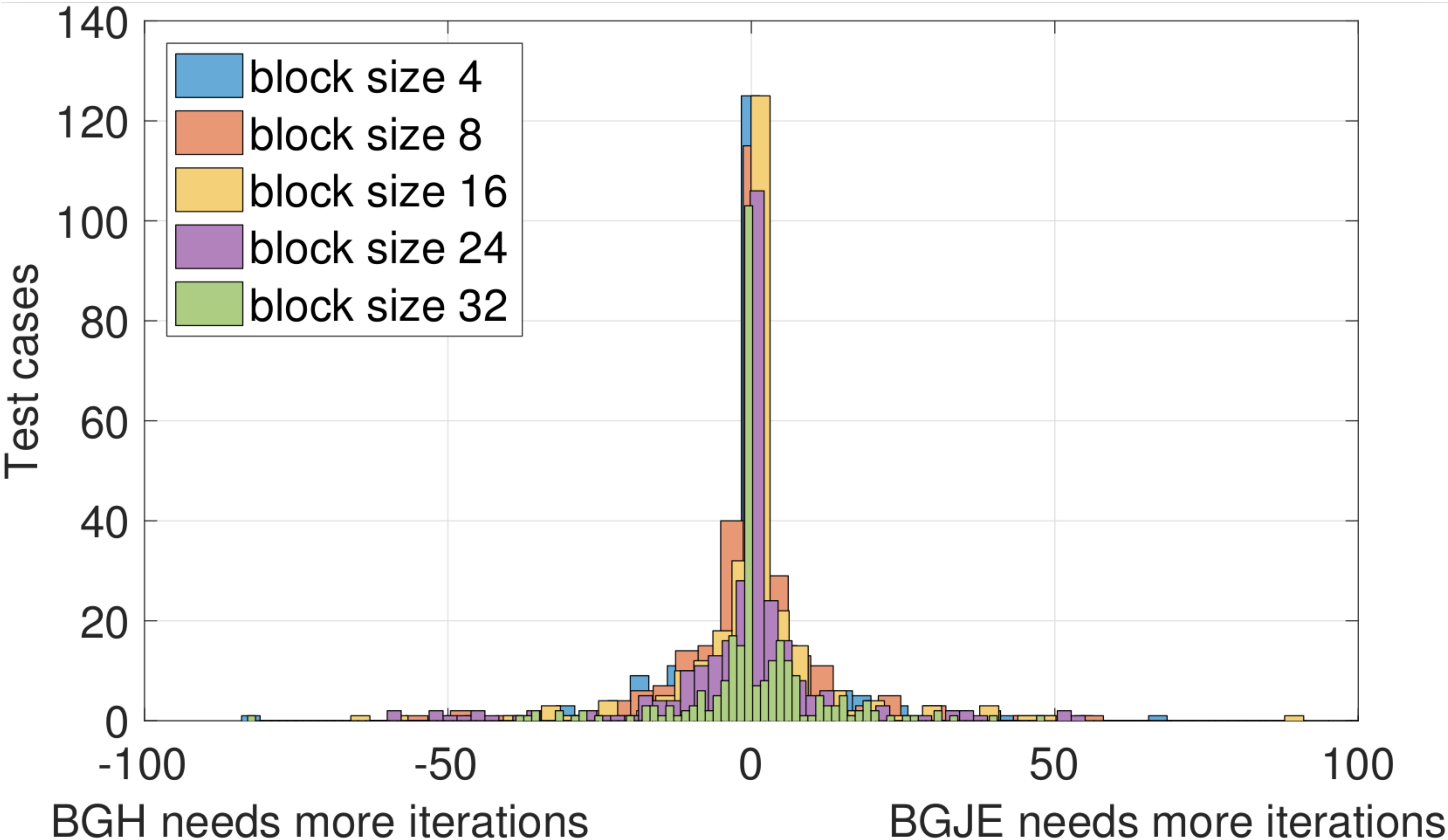


Implementation options

- Matrix decomposition in setup + solve in application
  - (FLOPS: $2/3n^3$ setup, $2n^2$ app.)
  - Gauss-Huard decomposition



Inversion?!



block size 4
block size 8
block size 16
block size 24
block size 32
Test cases
140
120
100
80
60
40
20
0
-100
-50
0
50
100
BGH needs more iterations
BGJE needs more iterations

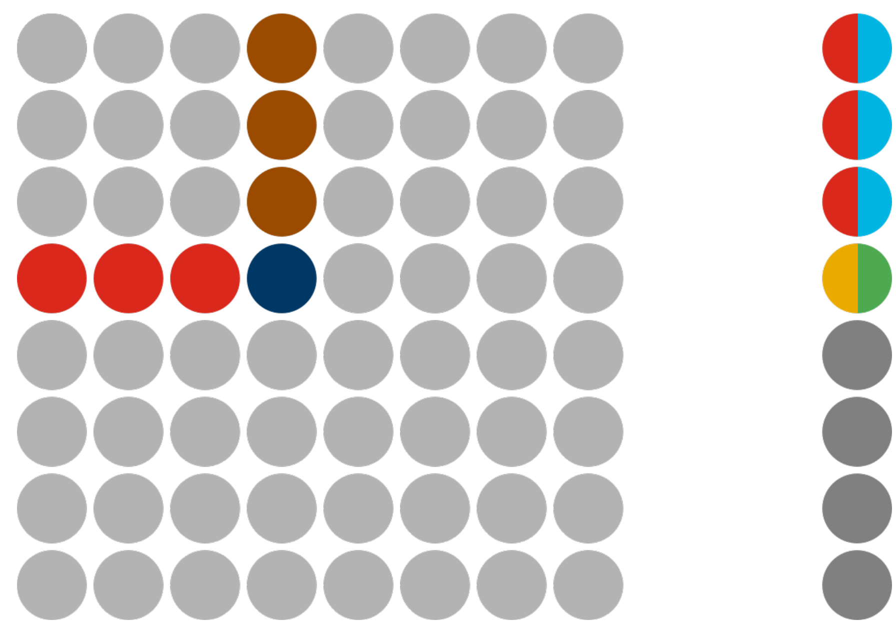
Gauss-Huard decomposition

- Decomposition
  - GEMV (G = G - RR)
  - SCAL (O = O / B)
  - GER (L = L - BO)
  - Column pivoting
    - Do not swap the columns, just remember which thread holds which column of the result

Gauss-Huard decomposition

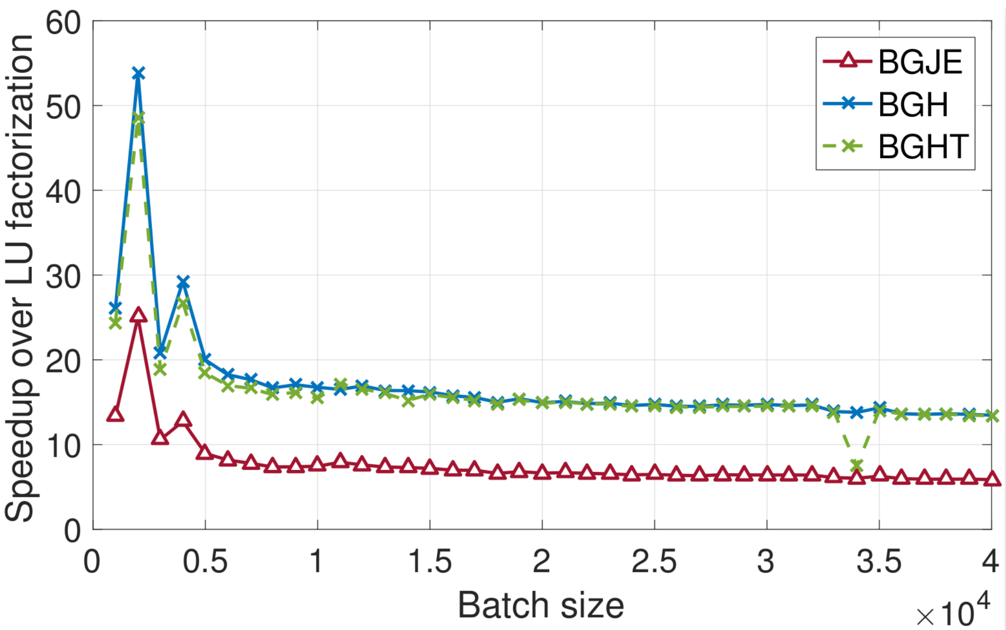
- Decomposition
  - GEMV (G = G - RR)
  - SCAL (O = O / B)
  - GER (L = L - BO)
  - Column pivoting
    - Do not swap the columns, just remember which thread holds which column of the result

- Solve
  - Load only the solution vector into registers
  - DOT (G = G – RR)
  - SCAL (O = O / B)
  - AXPY (L = L - BO)
  - Write lower part transposed wrp. to anti-diagonal for coalesced mem. access (*GHT*)

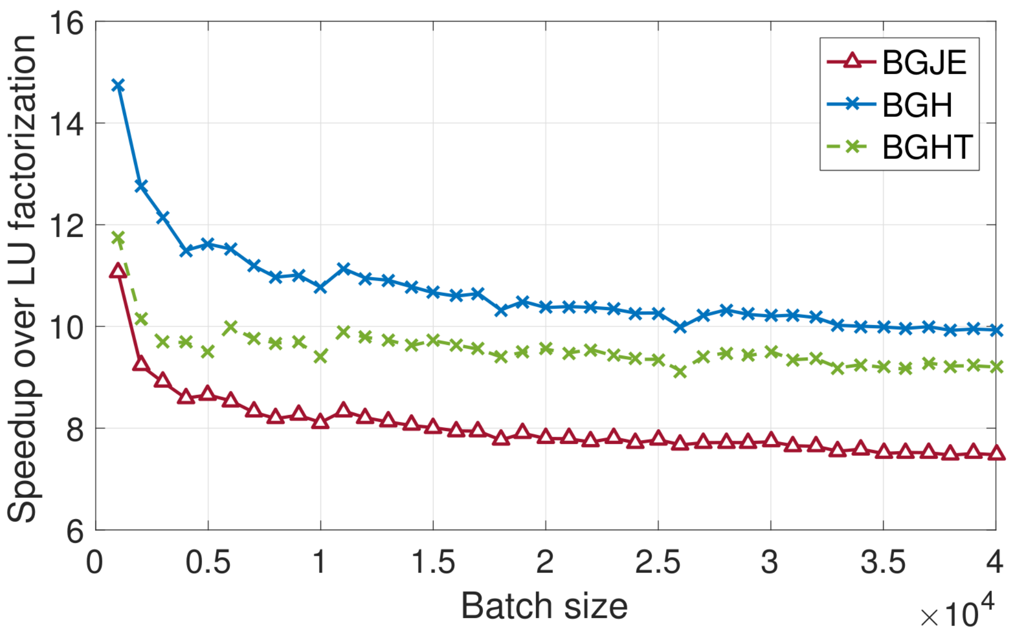


Decomposition comparison to batched LU (MAGMA)

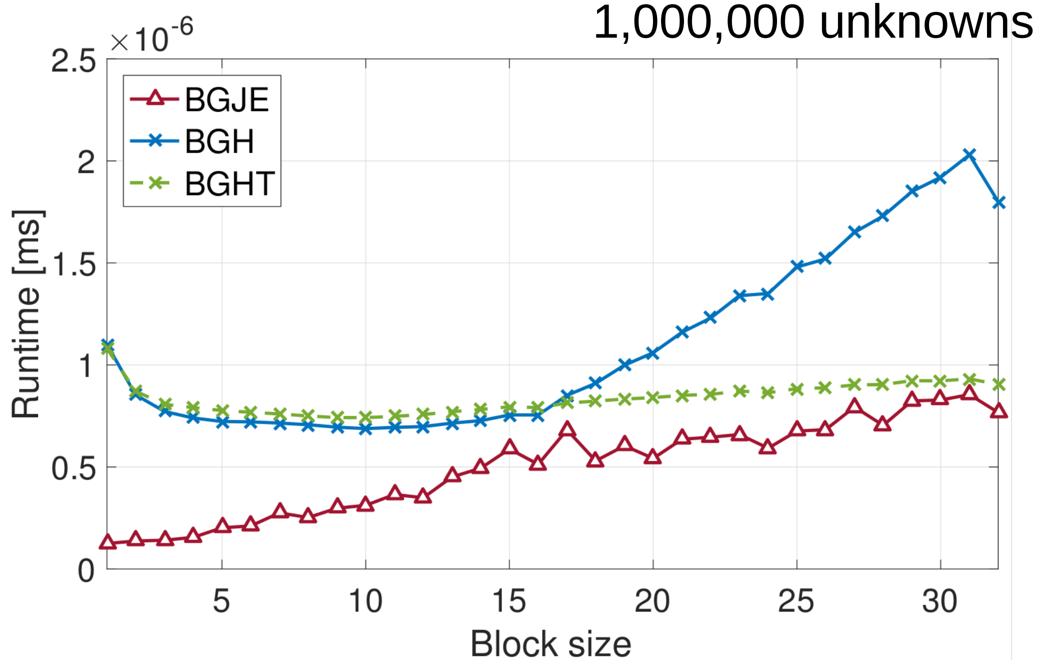
Block size 16



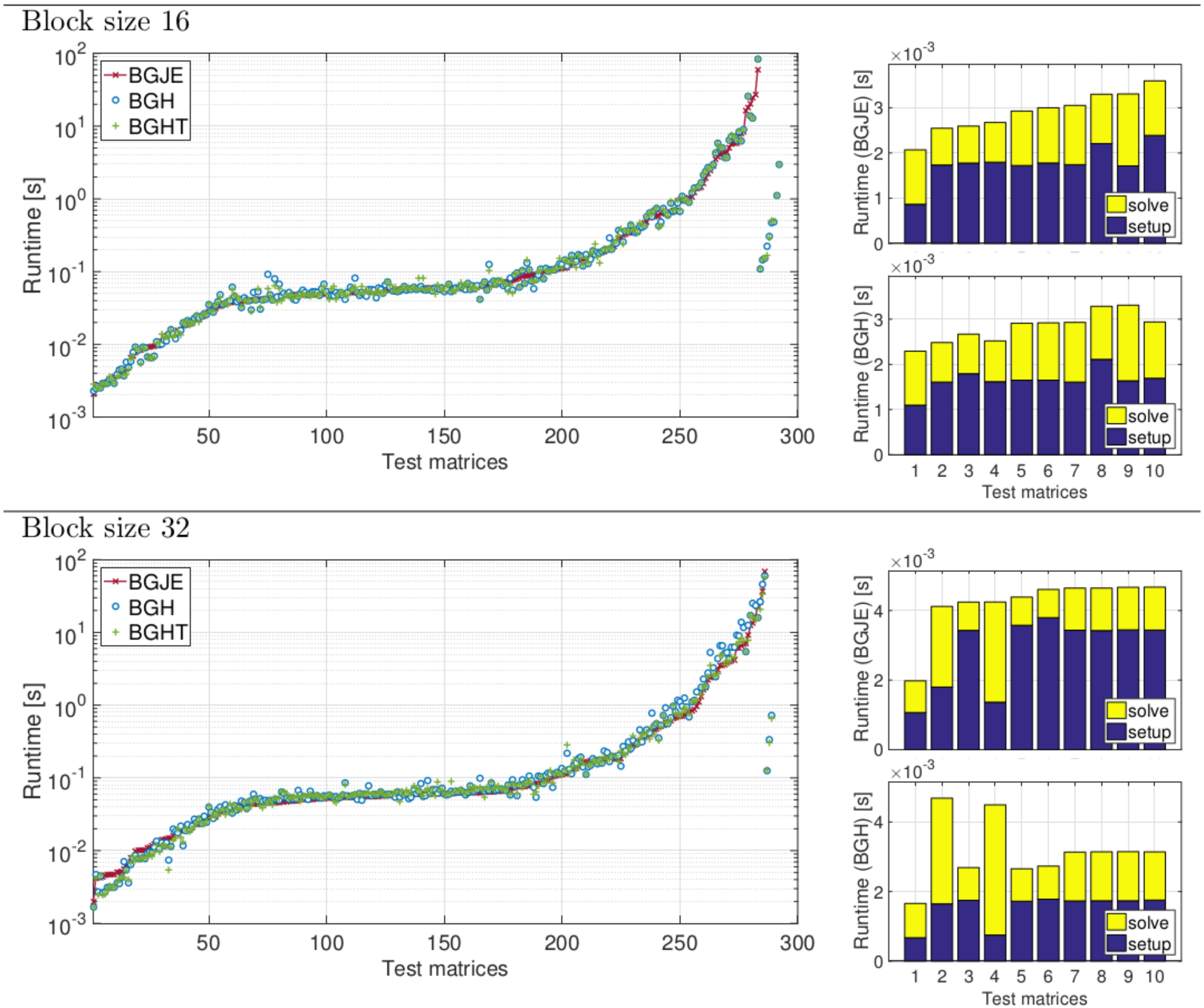
Block size 32



Application time



Total runtime of block-Jacobi preconditioned BiCGSTAB



Thank you! Questions?

All functionalities are part of the MAGMA-sparse project.

MAGMA SPARSE

ROUTINES	BiCG, BiCGSTAB, Block-Asynchronous Jacobi, CG, CGS, GMRES, IDR, Iterative refinement, LOBPCG, LSQR, QMR, TFQMR
PRECONDITIONERS	ILU / IC, Jacobi, ParILU, ParILUT, Block Jacobi, ISAI
KERNELS	SpMV, SpMM
DATA FORMATS	CSR, ELL, SELL-P, CSR5, HYB

http://icl.cs.utk.edu/magma/

This research is based on a cooperation between Hartwig Anzt, Jack Dongarra (University of Tennessee), Goran Flegar, Enrique S. Quintana-Ortí and Adrés E. Tomás (Universidad Jaume I).

THE UNIVERSITY OF TENNESSEE KNOXVILLE

UNIVERSITAT JAUME•I