Karlsruher Institut für Technologie

EXASCALE COMPUTING PROJECT

IMAGINE TOMORROW

ISC High Performance

MAY 21 – 25, 2023 | HAMBURG, GERMANY

A Mixed Precision Randomized Preconditioner for the LSQR Solver on GPUs

Vasileios Georgiou, KIT
vasileios.georgiou@kit.edu

Christos Boutsikas, Purdue
cboutsik@purdue.edu

Petros Drineas, Purdue
pdrineas@purdue.edu

Hartwig Anzt, UTK
hanzt@icl.utk.edu

Introduction

Reduced precision formats

- fp32, tf32, fp16, bfloat16

Mixed precision NLA algorithms

- solvers, preconditioners, eigenvalues, multigrid and more ...

Software

- Ginkgo, MAGMA, hypre, heFFTe, PETSc, SLATE, SuperLu

Randomized NLA & mixed precision computations

Why randNLA?

- Introduces noise to input data anyway
- Works well with applications that don’t require accuracy **[Paschou, et. al.]**
- Necessary for large-scale data analysis **[Bose et. al.]**
- Emphasis on data interpretation **[Mahoney, Drineas]**
- Probabilistic rounding*

Goals

1. Implement a preconditioner for regression solvers
    - right-preconditioner
    - LSQR iterative solver **[Paige, Saunders]**

2. Introduce mixed precision **only on the preconditioner-generation step**
    - Is the convergence of LSQR affected?
    - Speedup?
    - What matrix properties affect performance?

LSQR

$$x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \left\| b - A x \right\|_2$$

Based on the solution of the augmented system:

$$\begin{bmatrix} I & A \\ A^\top & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

Preconditioned LSQR

$$z^* = \underset{z \in \mathbb{R}^n}{\operatorname{argmin}} \left\| b - A M^{-1} z \right\|_2$$

$$M x = z$$

$$\begin{bmatrix} I & (AM^{-1}) \\ (AM^{-1})^\top & 0 \end{bmatrix}$$

symmetric → lanczos procedure

i-th iteration of LSQR

update basis vectors:

1. $\beta_{i+1} u_{i+1} = A(M \backslash v_i) - \alpha u_i$
2. $\alpha_{i+1} v_{i+1} = M^\top \backslash (A^\top u_{i+1}) - \beta v_i$

solve 2 x 3 system with givens qr:

3. $c_i = \rho_i / \rho_i$
4. $s_i = \beta_{i+1} / \rho_i$
5. $\theta_{i+1} = s_i \alpha_{i+1}$
6. $\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$
7. $\phi_i = c_i \phi$
8. $\bar{\phi}_{i+1} = c_i \bar{\phi}$

Update solution:

9. $x_i = x_{i-1} + M \backslash ((\phi_i / \rho_i) w_i)$
10. $w_i = v_{i+1} + (\theta_{i+1} / \rho_i) w_i$

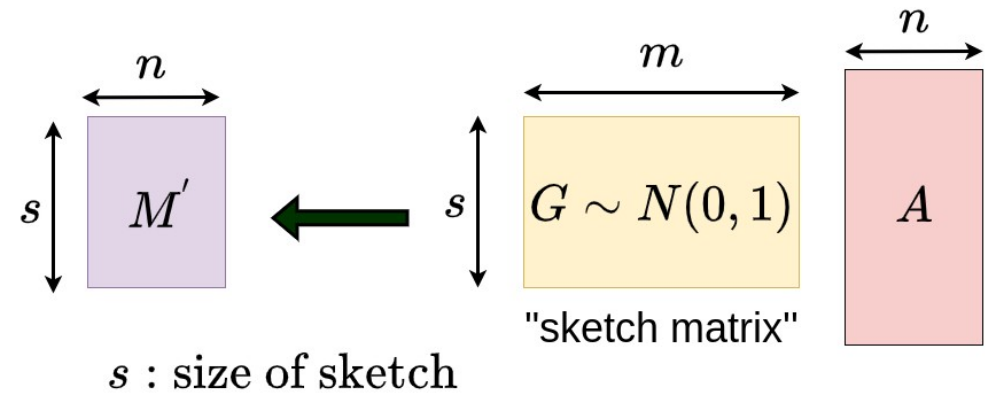
Randomized preconditioning

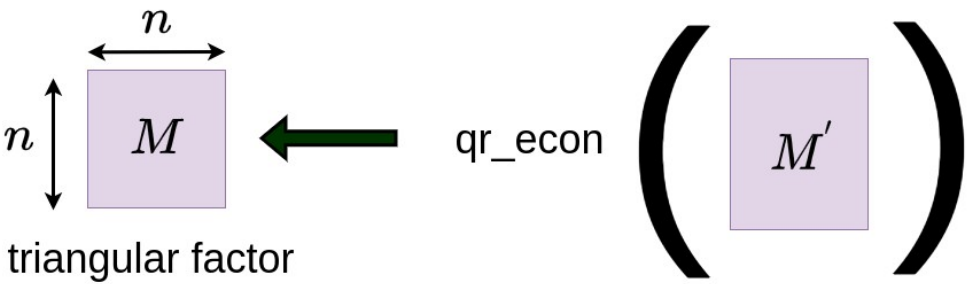
Based on a random ‘’**sketch operation”** on the matrix to reduce its dimensions.

- Early work **[Rokhlin, Tygert]**
- BLENDENPICK outperforms dense qr codes **[Avron, Maymounkov, Toledo]**
- LSRN **[Meng, Saunders, Mahoney]**

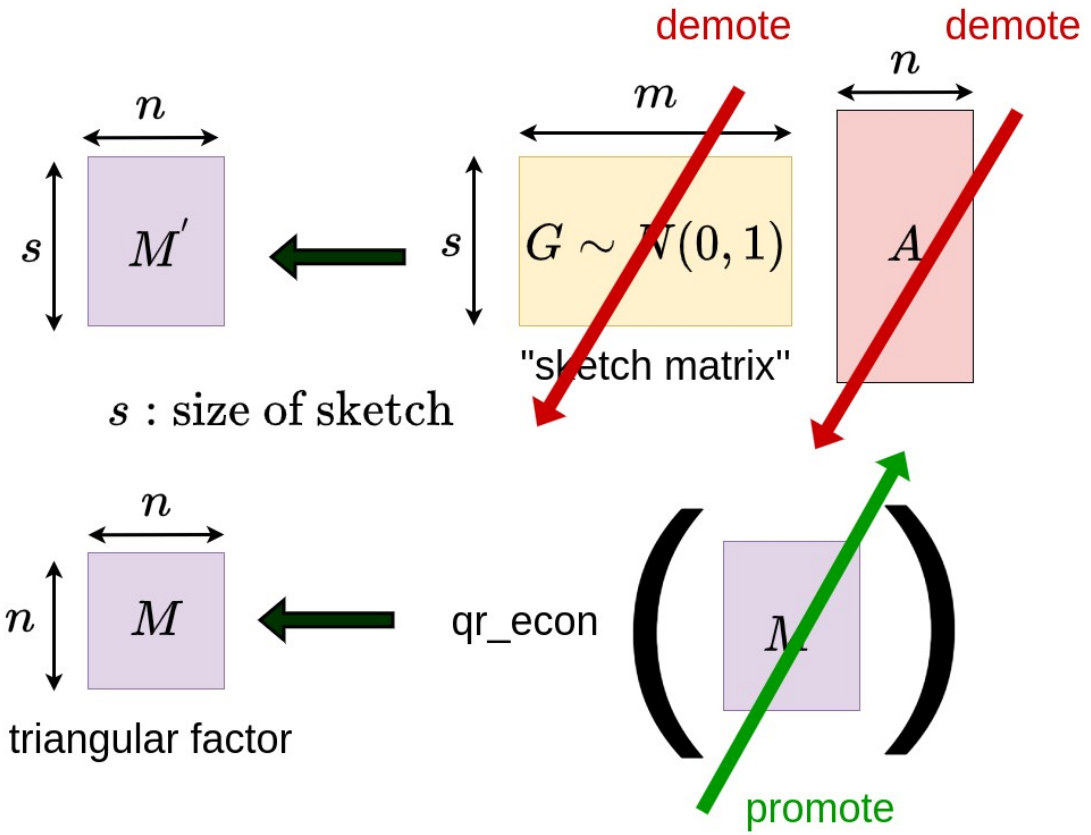
Ideally sketching preserves singular values of the input matrix.

Randomized preconditioning





Randomized preconditioning



Implementation

C++ implementation of LSQR and the randomized preconditioner.

MAGMA as baseline for linear algebra operations: https://icl.utk.edu/magma/

	high precision	reduced precision
preconditioner	**fp64** [fp32]	**fp64, fp32, tf32, fp16**
solver	**fp64** [fp32]	**fp64**, [fp32, tf32, fp16]

Supported precisions. Precisions used for our experiments are in bold.

Generating random samples via cuRand: https://docs.nvidia.com/cuda/curand/index.html.

Conversions via custom CUDA kernels

Experiment setup

1. For every precision combination report:
   (a) convergence
   (b) preconditioner generation runtimes
   (c) runtime profile
   (d) total runtime savings: $\frac{t_fp64 - t_best}{t_best}$

2. Compare for number of $sketch_size = sampling_coefficient \times mtx_cols$

   sampling_coefficient = 1.5, 2.0, …, 4.0

3. Generate rhs that belongs in the column space: $x_true = randn(n, 1);$ $b = A * x_true;$

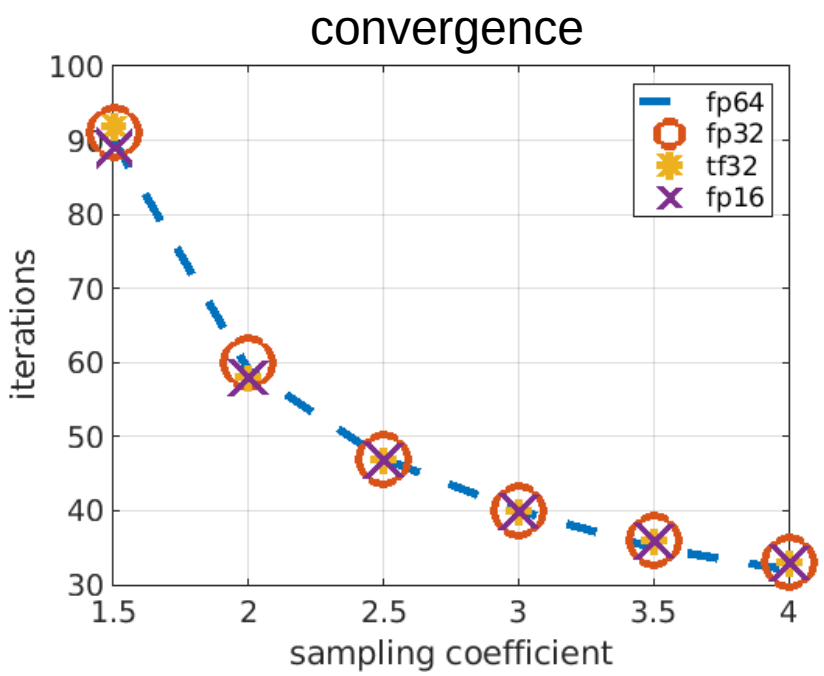
4. Experiments run on MD EPYC 7742 64-Core Processor cluster CPUs and a single A100 80GB SXM NVIDIA GPU

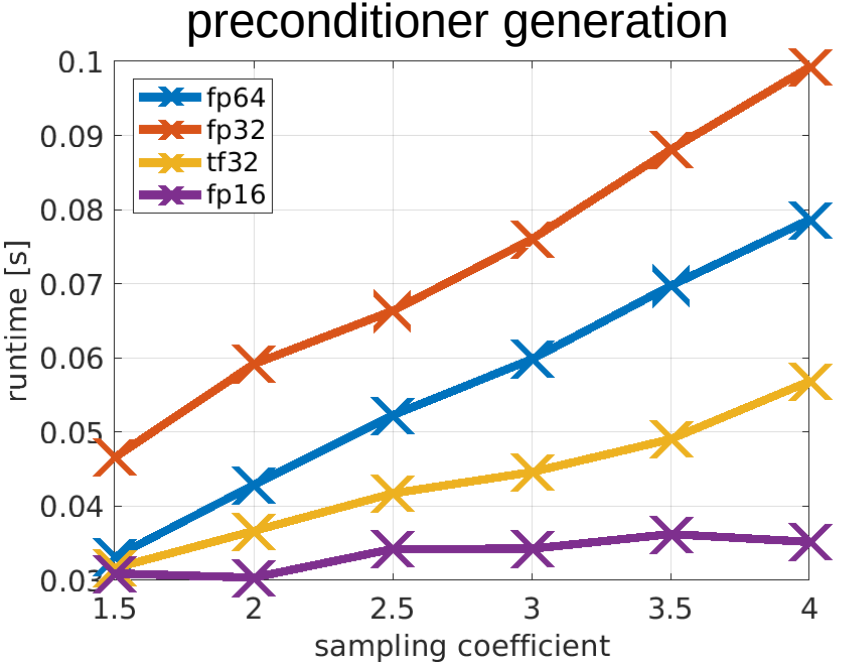
Datasets

dataset	rows	cols	cond	aspect ratio
HGDP_1	643,862	425	~1e3	~1.5e3
HGDP_2	60,000	1000	~1e6	6.0e1
CIFAR_1	20,000	1,000	~1e3	2.0e1
CIFAR_2	20,000	3,072	1e4	~6.5e0

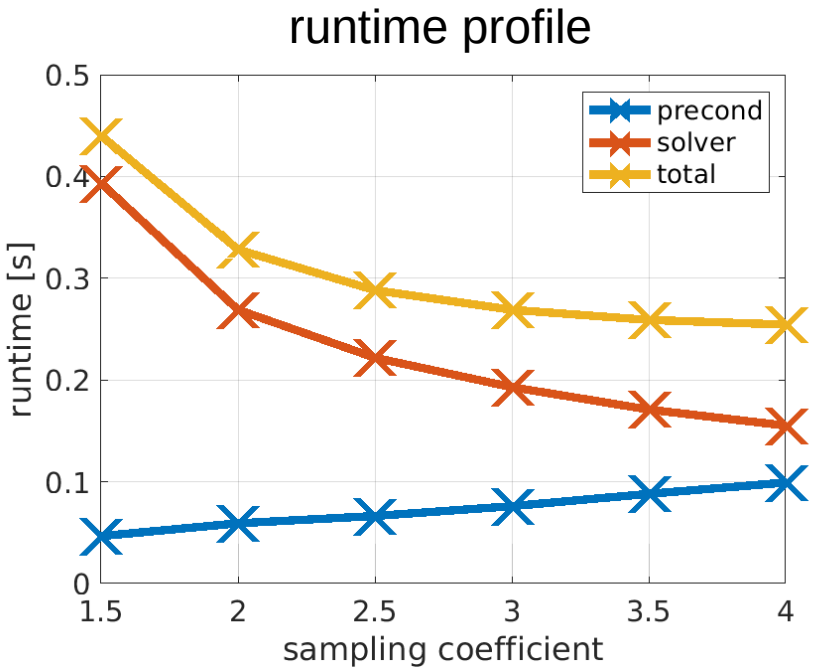
HGDP: population genetics (**[Paschou, et. al.]**)
CIFAR: images, https://www.cs.toronto.edu/~kriz/cifar.html

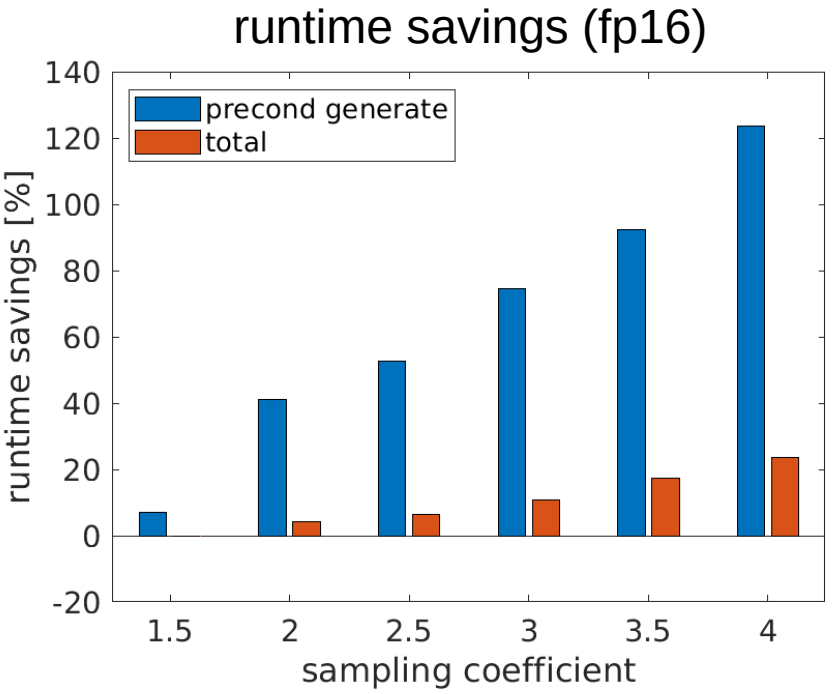
HGDP_1, cond = 1e3, rows: 643,862, cols: 425, aspect ratio = 1.5e3, tol = 1e-10



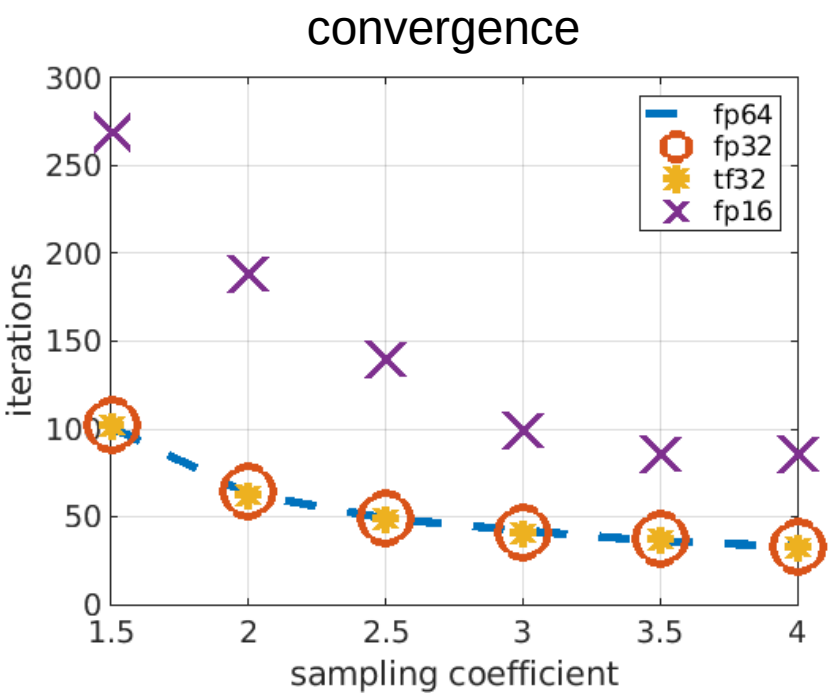


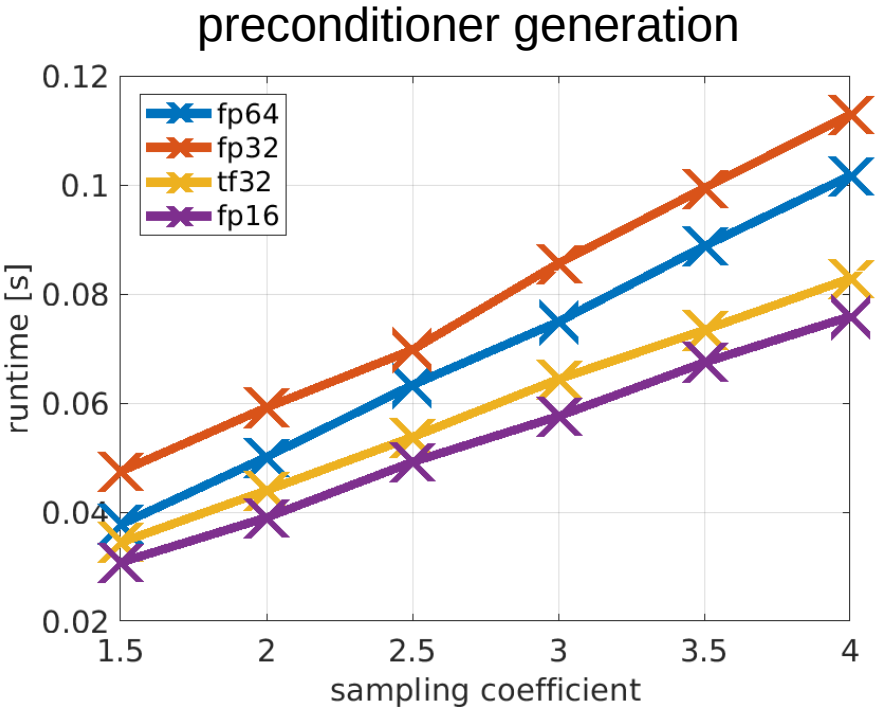
HGDP_1, cond = 1e3, rows: 643,862, cols: 425, aspect ratio = 1.5e3, tol = 1e-10



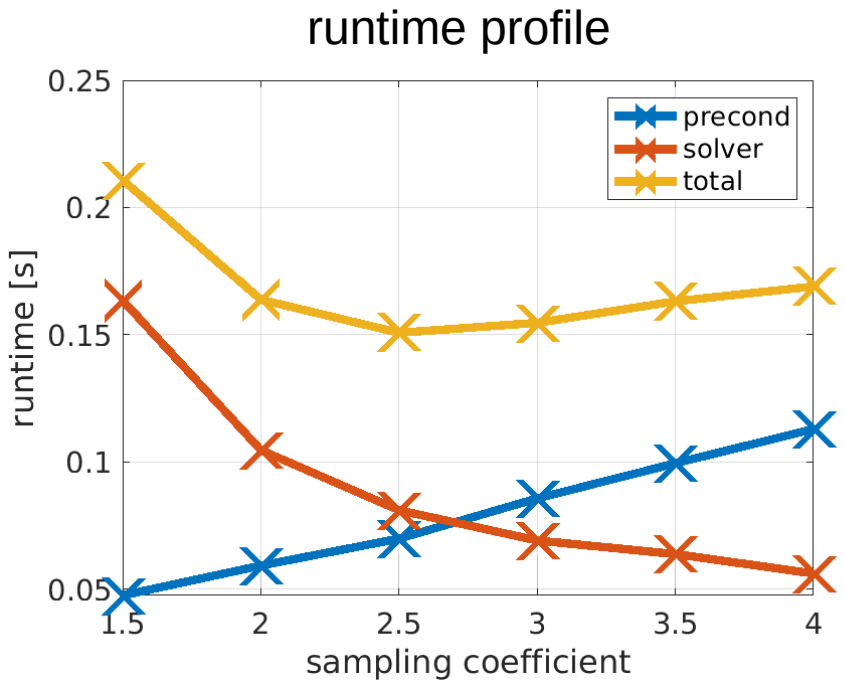


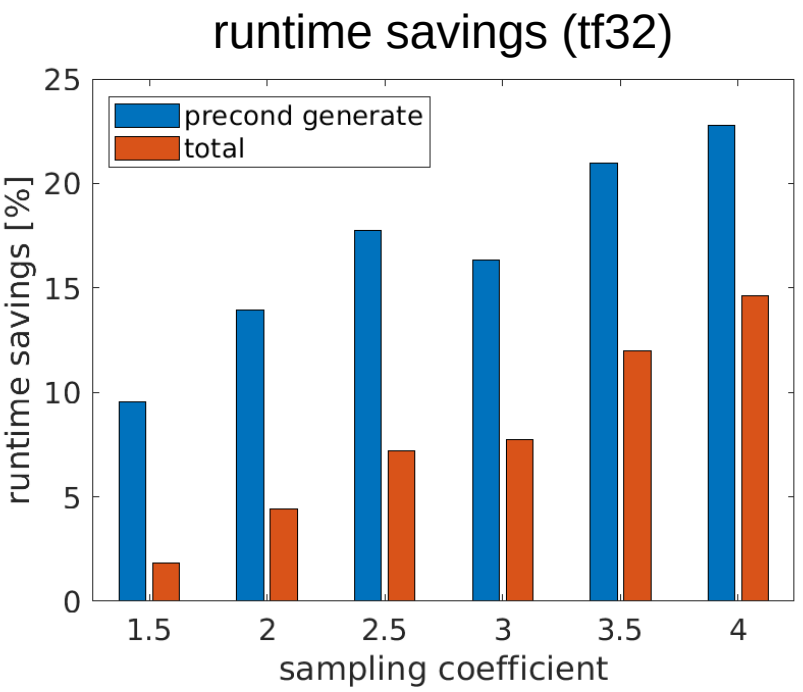
HGDP_2, cond = 1e6, rows: 60,000, cols: 1000, aspect ratio = 6.0e1, tol = 1e-10



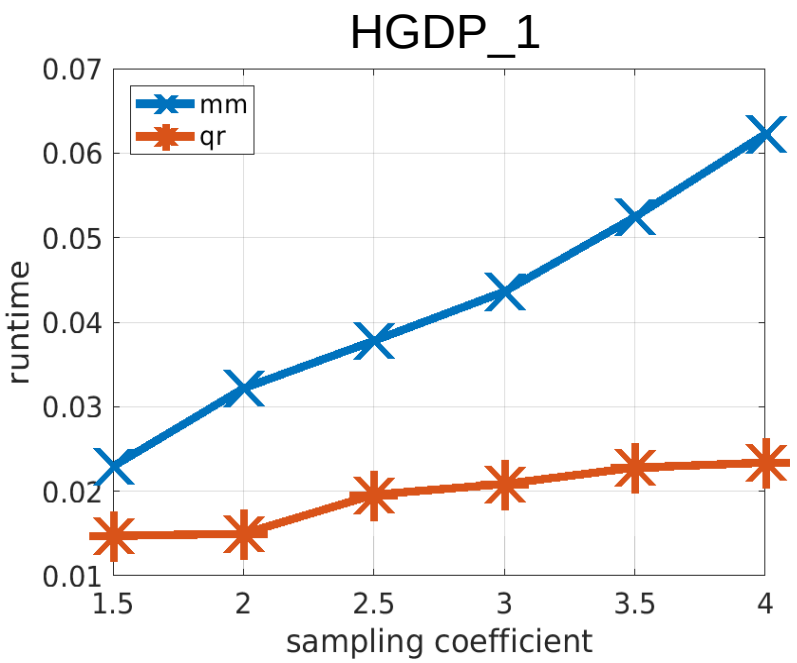


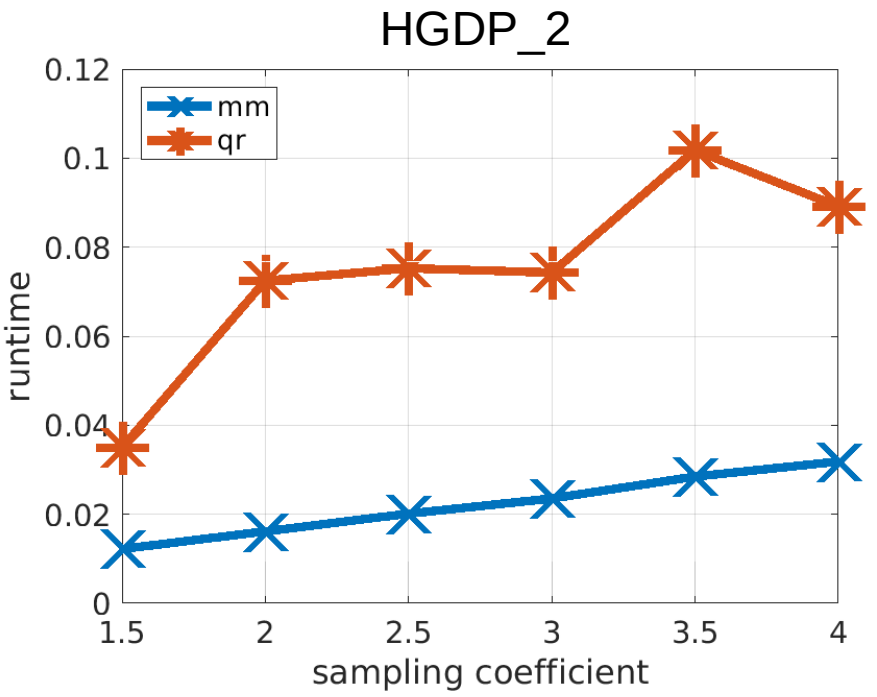
HGDP_2, cond = 1e6, rows: 60,000, cols: 1000, aspect ratio = 6.0e1, tol = 1e-10





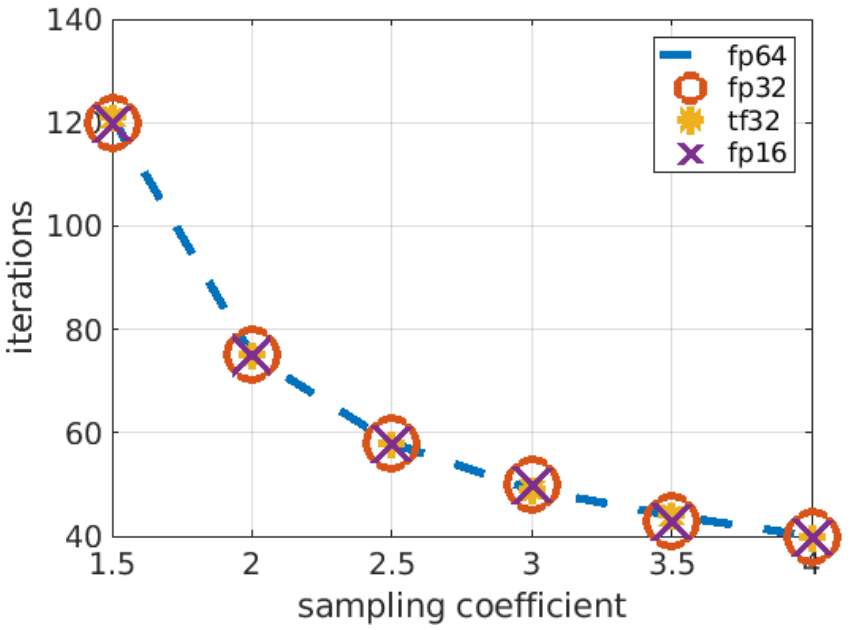
Preconditioner components (HGDP)



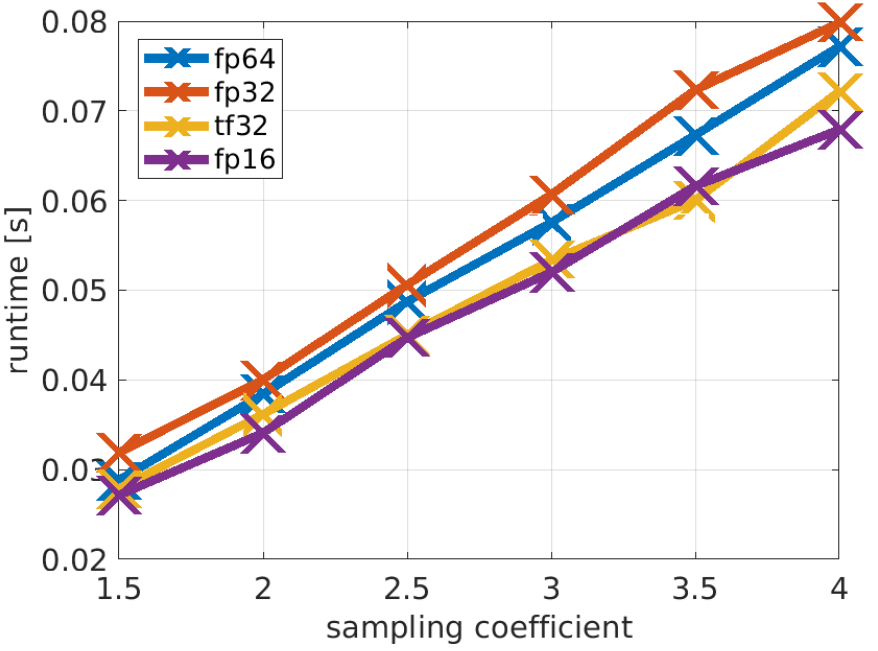


CIFAR_1, cond = 1e3, rows: 20,000, cols: 1,000, aspect ratio = 2.0e1, tol = 1e-12

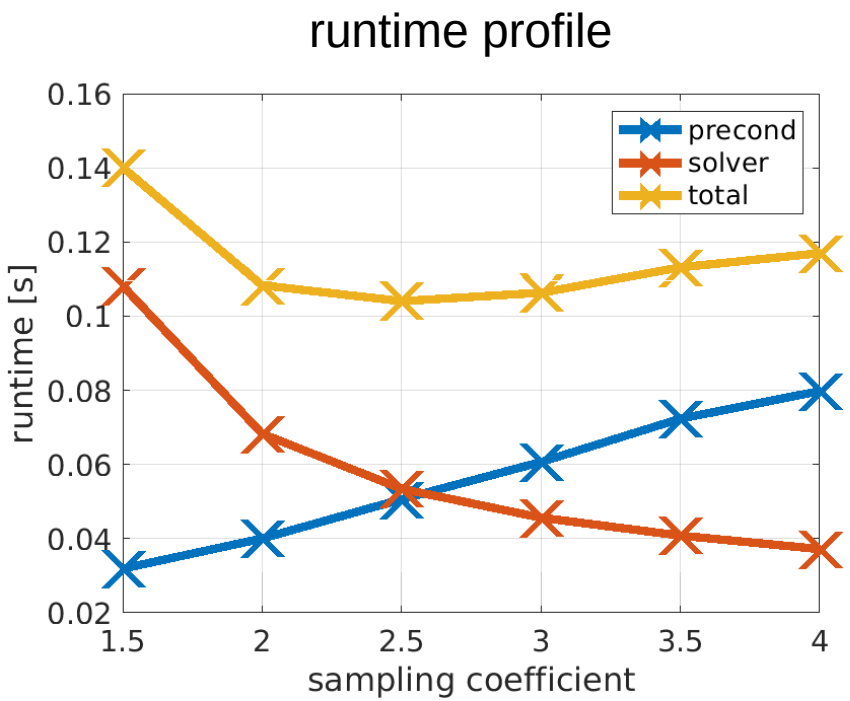
convergence

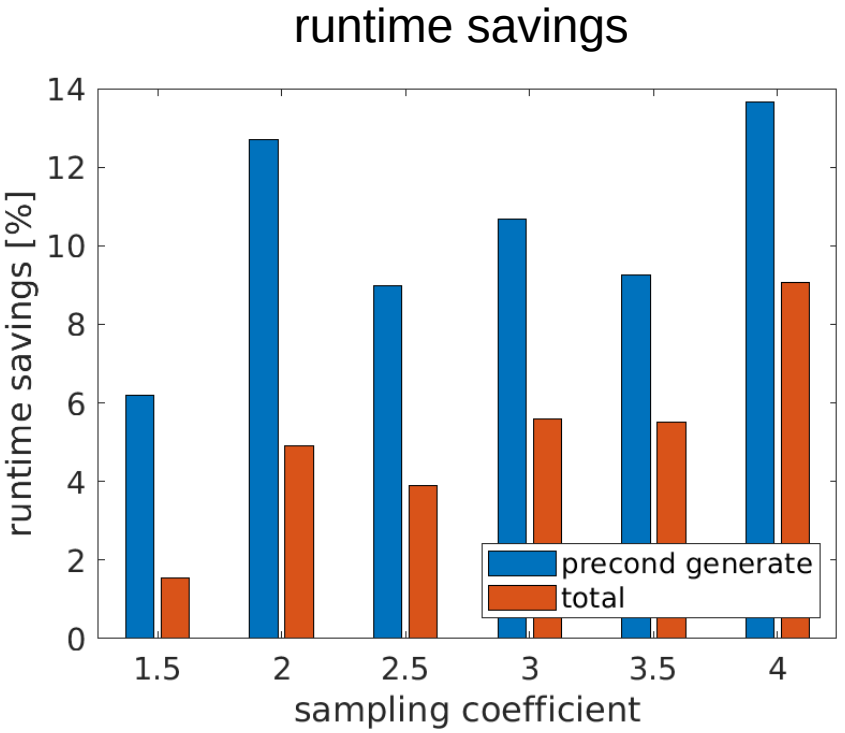


preconditioner generation



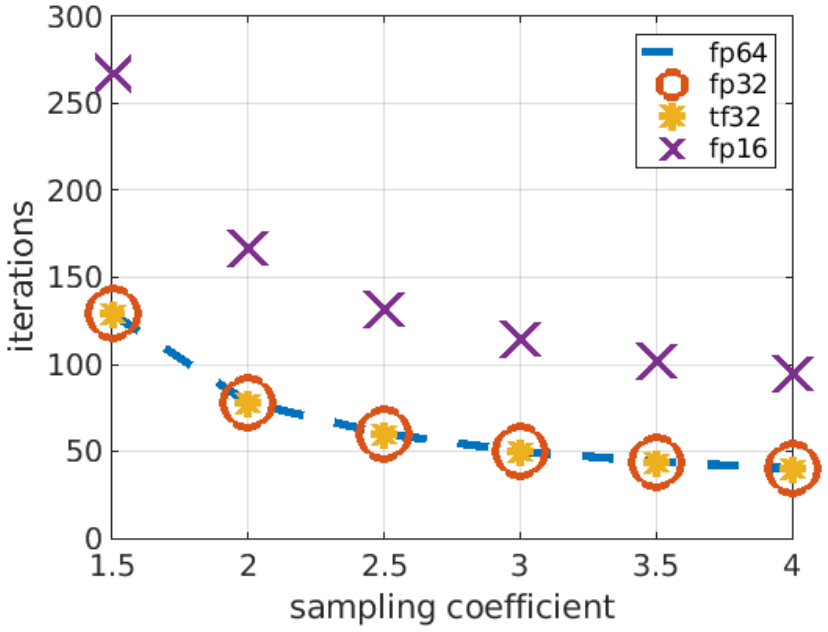
CIFAR_1, cond = 1e3, rows: 20,000, cols: 1,000, aspect ratio = 2.0e1, tol = 1e-12



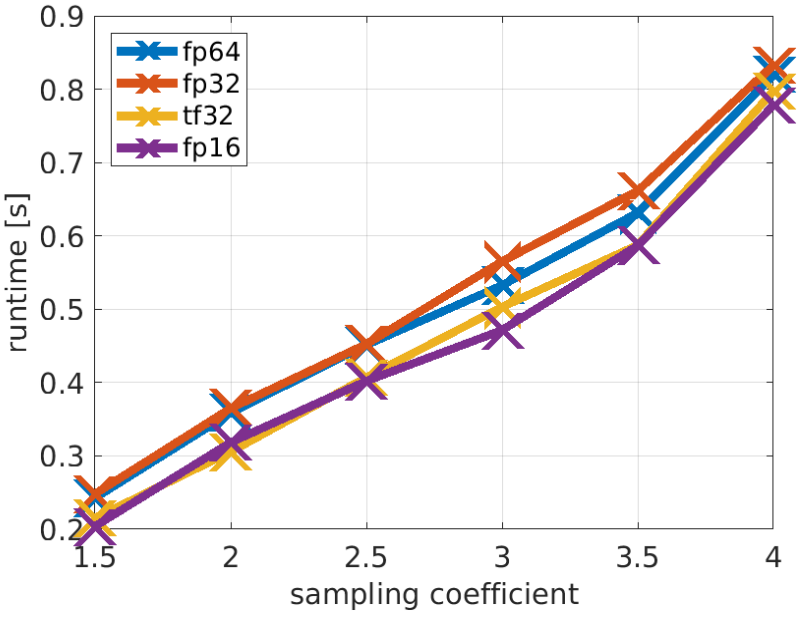


CIFAR_2, cond = 1e3, rows: 20,000, cols: 1,000, aspect ratio = 6e0, tol = 1e-12

convergence

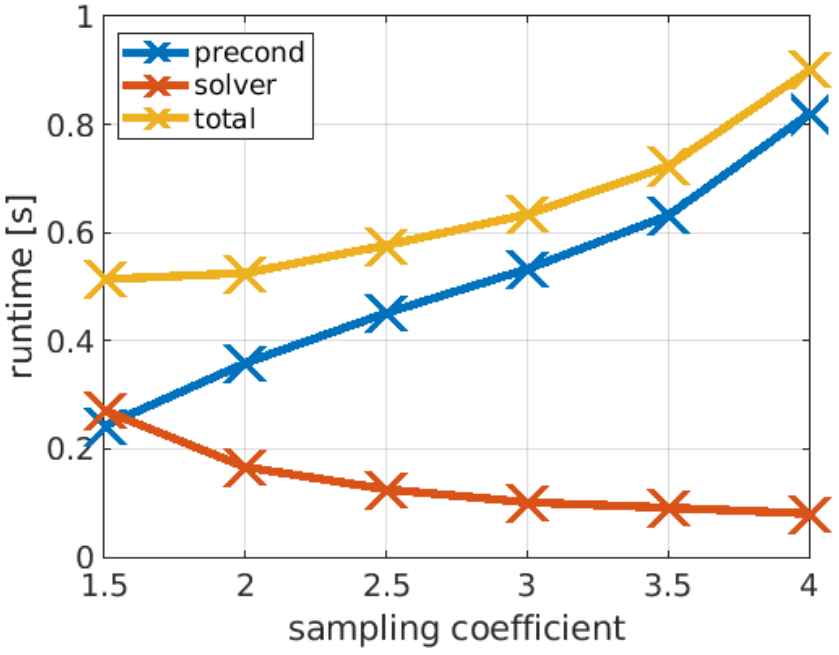


preconditioner generation

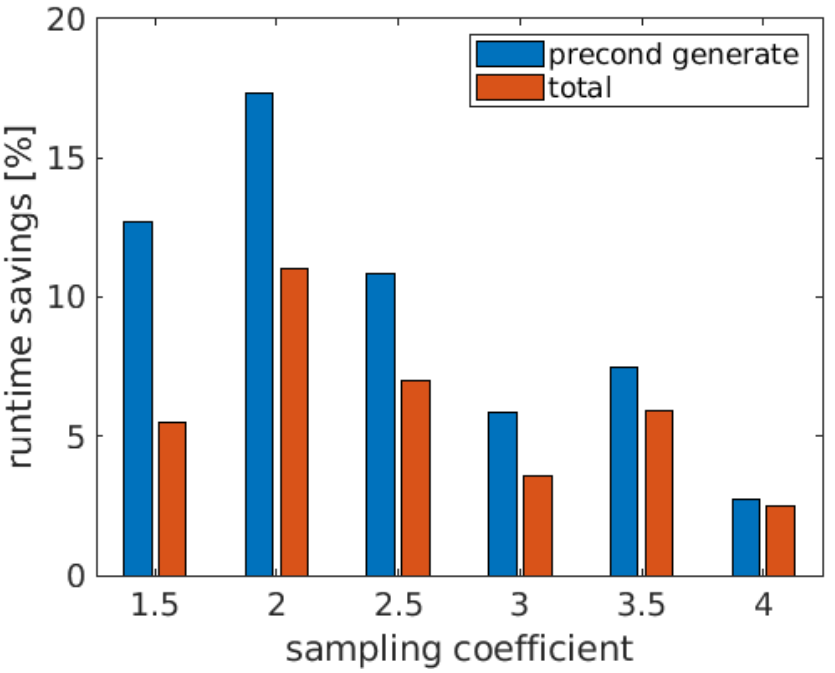


CIFAR_2, cond = 1e3, rows: 20,000, cols: 1,000, aspect ratio = 2.0e1, tol = 1e-12

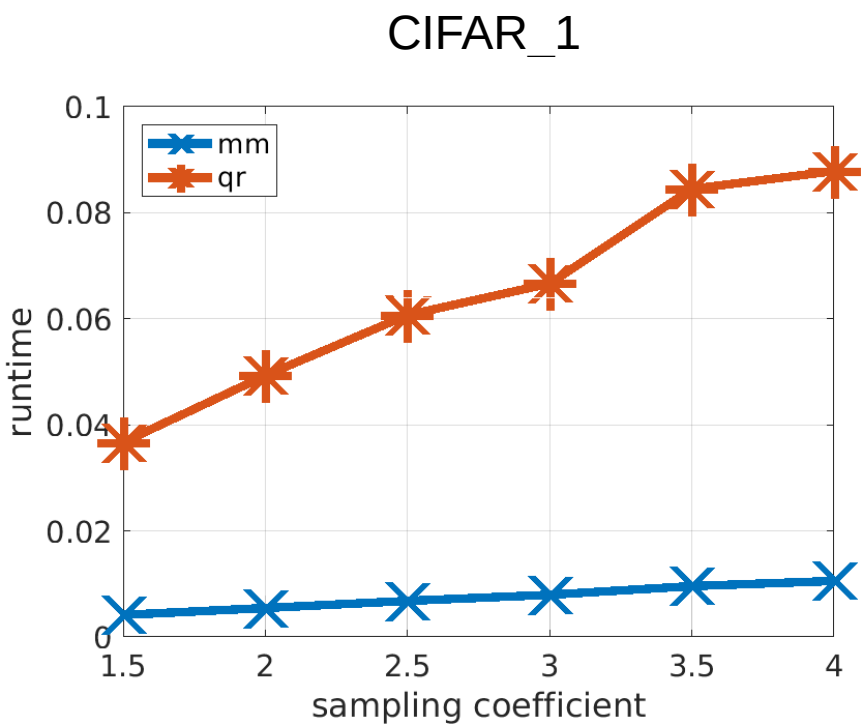
runtime profile

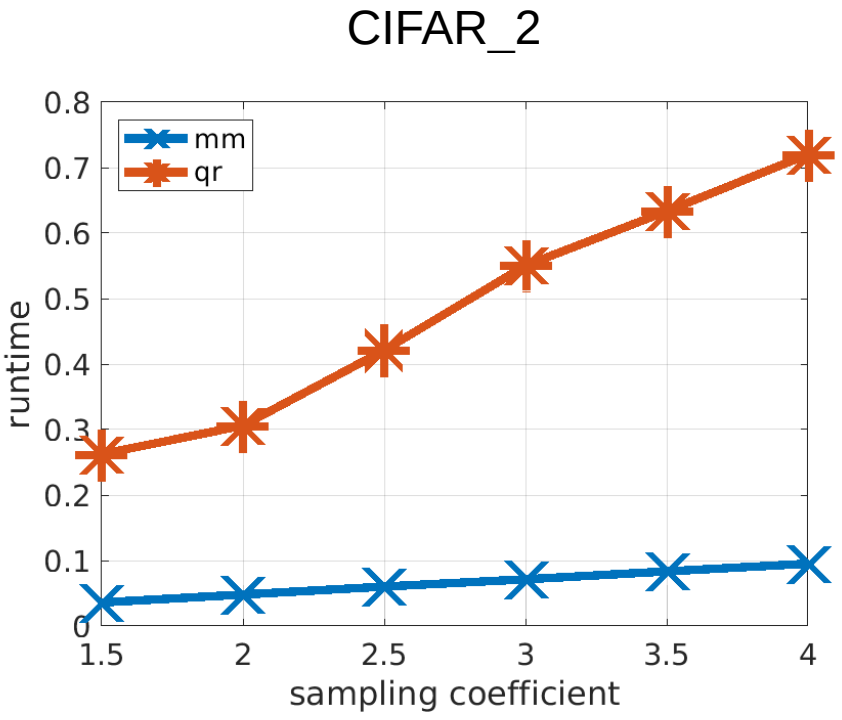


runtime savings



Preconditioner components (CIFAR)





Conclusions

4 test matrices with distinct runtime profile and aspect ratio

- Convergence sometimes delayed for fp16 (HGDP_2, CIFAR_2)
- Balanced preconditioner generation & solver runtime required for speedup achieved at the generation stage to translate to overall speedup
- QR computation dominates preconditioner runtime as aspect ratio increases, diminishing the effectiveness of the reduced precision computation

Open problems & future work

- Combine with mixed precision LSQR
- qr in reduced precision
- Can column pivoting qr fix convergence slowdown? IR?
- Sparse problems
- Improve problem selection. Benchmark?

Acknowledgments

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

Many thanks to the Innovative Computing Lab at University of Tennessee, for providing access to their compute cluster.

References

[Avron, Maymounkov, Toledo]: Blendenpik: supercharging LAPACK's least-squares solver. SIAM J. Sci. Comput. 32(3), 1217–1236 (2010). https://doi.org/10.1137/090767911

[Meng, Saunders, Mahoney]: LSRN: a parallel iterative solver for strongly over- or underdetermined systems. SIAM J. Sci. Comput. 36(2), C95–C118 (2014). https://doi.org/10.1137/120866580

[Rokhlin, Tygert]: A fast randomized algorithm for overdetermined linear least-squares regression. Proc. Natl. Acad. Sci. 105(36), 13212–13217 (2008). https://doi.org/10.1073/pnas.0804869105. https://www.pnas.org/doi/abs/10.1073/pnas.0804869105

[Paschou, Lewis, Javed, Drineas]: Ancestry informative markers for fine-scale individual assignment to worldwide populations. J. Med. Genet. 47(12) (2010). https://doi.org/10.1136/jmg.2010.078212

[Paige, Saunders]: LSQR: an algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Softw. (TOMS) 8(1), 43–71 (1982)

[Tomov, Dongarra, Baboulin]: Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Comput. 36(5–6), 232–240 (2010). https://doi.org/10.1016/j.parco.2009.12.005

[Anzt, Cojean, Flegar, Göbel, Grützmacher,Nayak, Ribizel, Tsai, and Quintana-Ortí.] 2022. Ginkgo: A Modern Linear Operator Algebra Framework for High Performance Computing. ACM Trans. Math. Softw. 48, 1, Article 2 (March 2022), 33 pages. https://doi.org/10.1145/3480935

[Mahoney, Drineas] CUR matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences 106, 697-702 (2009)

[Bose, Kalantzis, Kontopoulou, Elkady, Paschou, Drineas] TeraPCA: a fast and scalable software package to study genetic variation in tera-scale genotypes. Bioinformatics (October 2019), Issue 35, Volume 19, Pages 3679–3683