



CONFERENCE

—NORTH AMERICA 2026—

Linear algebra kernels for the age of research

Mark Saroufim

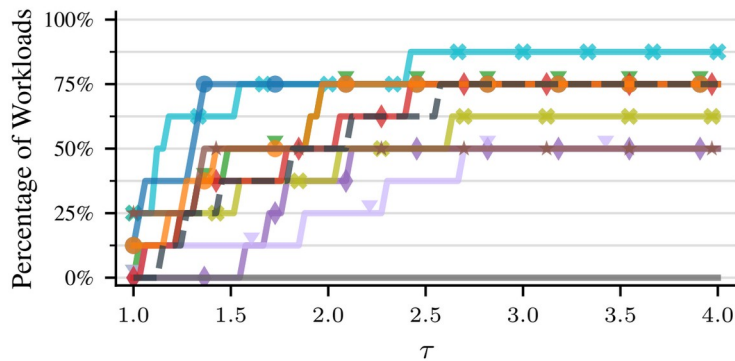
Core Auto / GPU MODE

Better optimizers can train faster

Shampoo: ~28% faster training on average

Submission	Line	Score
PYTORCH DISTRIBUTED		0.7784
SHAMPOO		0.7077
SCHEDULE FREE ADAMW		0.6383
GENERALIZED ADAM		0.6301
CYCLIC LR		0.5909
NADAMP		0.5909
BASELINE		0.5707
AMOS		0.4918
CASPR ADAPTIVE		0.4722
LAWA QUEUE		0.3699
LAWA EMA		0.3384
SCHEDULE FREE PRODIGY		0

(a) External tuning leaderboard



(b) External tuning performance profiles

AlgoPerf, Kasimbeg et al. (ICLR 2025), Figure 1

A better optimizer can still be slower

Optimizer A $100 \text{ steps} \times 1.0 \text{ s} = 100 \text{ s}$

Optimizer B $80 \text{ steps} \times 1.3 \text{ s} = 104 \text{ s}$

Illustrative numbers

The three kernels

QR

$$A = QR$$

Q: orthonormal columns
R: upper triangular

SOAP eigenbasis updates

eigh

$$A = U\Lambda U^T$$

U: eigenvectors
 Λ : eigenvalues

Shampoo inverse roots

Cholesky

$$A = LL^T$$

L: lower triangular
A: positive definite

Solving SPD systems

QR has more than one right answer

$$A = QR = (QD)(DR)$$

$$D = \text{diag}(1, -1, 1)$$

Flip a column of Q and the matching row of R .

Reconstruction

$$QR \approx A$$

Orthogonality

$$Q^T Q \approx I$$

Tolerances scale with matrix size and machine precision.

GPU MODE's linear algebra competition

75,306

submissions

394

people

QR

10billiontokens
gau.nernst
dhu.randhar

eigh

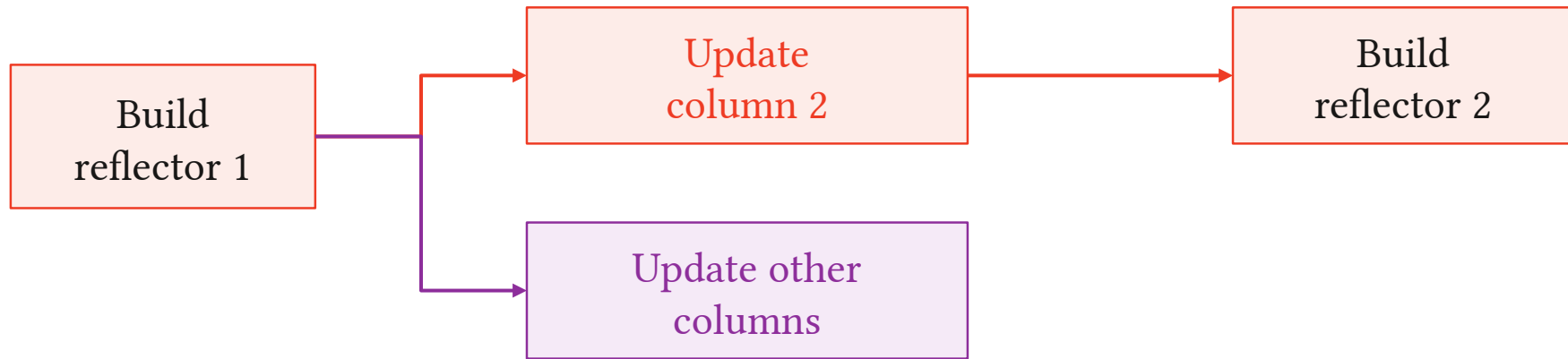
teelaitila
nikhilbarhate99
elianaive

Cholesky

zhongmingee
Frosty40
Olek

Sequential dependencies make GPUs sad

QR: reflector 2 needs the updated column 2.



The column updates can run in parallel.

QR creates zeros below the diagonal

A Householder reflector preserves length.

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} -5 \\ 0 \end{bmatrix}$$

Both vectors have length 5. The lower entry is now zero.

Apply to the remaining columns, then repeat.

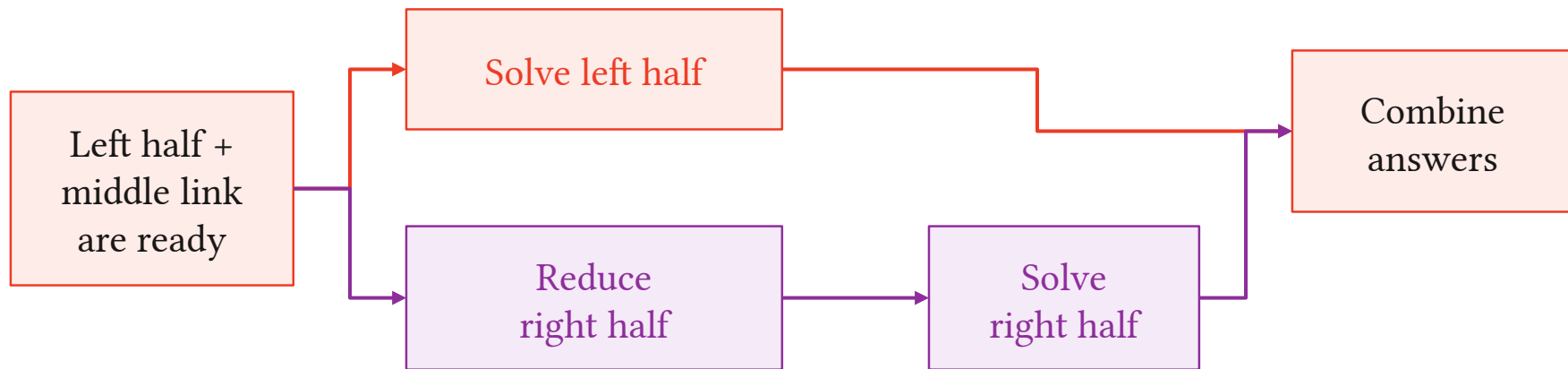
Input shape changes the QR schedule

Input and batch	Blocks per matrix	Scheduling strategy
512×512 640 matrices	1	Run matrices independently
1024×1024 60 matrices	2	Share reflectors across two SMs
4096×4096 2 matrices	Many	Split panel columns across blocks

Fewer matrices? More parallel work inside each matrix.

eigh: overlap independent GPU work

Teelaitila's 2048×2048 solver



The left solver reads finished entries. Reduction writes elsewhere.

Cholesky: combine the updates

$$A = LL^T$$

Frosty40

Separate subtractions

Subtract column 0's
outer product

Subtract column 1's
outer product



One matrix multiply

Subtract both
in one matmul

2 read/write passes

1 read/write pass

Scheduling the overlap

TMA moves data

Tensor cores compute

We schedule the overlap

CUDA Graphs, PDL, megakernels

Megakernels vs CUDA Graphs ♦ PDL

Implementation	Source lines*	H100 $\mu\text{s}/\text{step}$
CUDA Graphs + PDL	429	979.51
Hazy megakernel	3,498	972.78

88% fewer source lines

0.7% slower

Llama-3.2-1B, batch 1. Graphs + PDL passes numerical checks.

* Includes comments/blanks. Excludes harness and dependencies.

How to write a fast GPU program



1. Understand the memory hierarchy.
2. Write a fast kernel.

The kernel writeups

gpumode.com

Submissions: Hugging Face / GPUMODE / kernelbot-data

Core Auto is hiring

hello@coreauto.com