QR-Ω: Communication Avoiding 2.5D Householder QR

Felipe Tomé
Federal University of São Carlos (UFSCar)
São Carlos, São Paulo, Brazil
Massachusetts Institute of Technology (MIT)
Cambridge, MA, USA

Rabab Alomairy (advisor)
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia
Massachusetts Institute of Technology (MIT)
Cambridge, MA, USA

Alan Edelman (advisor)
Massachusetts Institute of Technology (MIT)
Cambridge, MA, USA

Hermes Senger (advisor)
Federal University of São Carlos (UFSCar)
São Carlos, São Paulo, Brazil
Universidade de São Paulo (USP)
São Paulo, São Paulo, Brazil

1 Introduction

Householder QR is a building block of least-squares solvers, Krylov methods, tensor networks, and orthogonality-constrained learning. Most of its arithmetic lies in trailing updates that are matrix products, yet panel dependencies and data movement hold GPU throughput below what those updates alone could sustain. Communication-avoiding algorithms reduce this cost [1]; hierarchical QR (HQR) organizes eliminations into local and global reduction trees [2]. On a GPU, however, an elimination tree does not specify how the parallel work within each update should use registers, shared memory, HBM, or inter-GPU links.

We present QR-Ω, a hierarchical 2.5D Householder QR formulation that carries those choices down through the GPU memory hierarchy. It keeps HQR's elimination list and adds four execution operations to each elimination: *Split*, *Replicate*, *Combine*, and *Pipeline*.

2 Method

For a compact-WY transformation $Q = I - VTV^{\mathsf{T}}$ [3], the update of trailing columns X is

$$W = V^{\mathsf{T}}X, \quad Z = T^{\mathsf{T}}W, \quad D = VZ, \quad X \leftarrow X - D.$$

Each product $C = AB$ has a carrier (p_i, p_j, c), which partitions its row, column, and contraction indices. Peer (x, y, z) computes $C^{(z)}_{xy} = A_{I_x,K_z}B_{K_z,J_y}$, and the owner obtains $C_{I_x,J_y} = \sum_z C^{(z)}_{xy}$. *Split* assigns these index ranges to concurrent peers; *Replicate* copies operand blocks required by multiple peers; *Combine* sums disjoint contraction contributions in a fixed order; and *Pipeline* overlaps column strips through the product chain, prioritizing the strip that releases the next panel.

HQR determines which tiles are eliminated and their dependencies. QR-Ω also specifies the decomposition of each product, the location of operand replicas and partial sums, and pipelining within an elimination. The additive reduction of matrix-product partials is distinct from the Householder transformations that merge triangular factors. Different products can therefore use different carriers without changing the QR elimination order.

Why 2.5D?. Replication trades memory for communication [4]. A fully three-dimensional decomposition fixes the replication dimension according to the processor grid. QR-Ω instead chooses c within the capacity of each memory domain. Smaller replication factors preserve most of that reuse at a fraction of the storage of full 3D replication, which matters most in registers and shared memory, where capacity is tightest.

3 Evaluation

Figures 1 and 2 summarize single-GPU performance and accuracy on NVIDIA H200 GPUs with CUDA 13.0.2. The single-GPU study covers $n = 256$–$131{,}072$ and compares tuned cuSOLVER, MAGMA, and SLATE configurations in FP64 and FP32. TF32 and 3×TF32 [5] use the references' FP32 results. Timings are medians after warm-up, excluding allocation and validation; throughput counts $4n^3/3$ operations. Numerical checks use the same configurations as timing.

At $n = 131{,}072$, QR-Ω attains 47.0, 47.5, 247.4, and 91.9 TFLOP/s in FP64, FP32, TF32, and 3×TF32, respectively. The tensor modes are 5.3× and 2.0× faster than the fastest FP32 reference. FP64 at this size and FP32 at $n = 80{,}000$ are ties within 1%. At $n = 80{,}000$, FP64 and FP32 reach 81% and 90% of measured same-size GEMM throughput.

4 Conclusion

QR-Ω extends hierarchical Householder QR by fixing how each individual update runs at every level of the memory hierarchy. On the H200, controlled replication and pipelining bring QR closer to matrix-product throughput while keeping reconstruction and orthogonality errors small in FP64 and FP32. The tensor modes trade accuracy for speed. These claims cover only the tested H200 setup; application-level speedups and distributed multi-node scaling remain untested.

5 Future Work

The single-node multi-GPU version of this work is implemented and tested but is no within the scope of this poster. Next we will measure QR-Ω inside tensor-network, pre-training, and simulation codes where QR sits on the critical path, and then validate distributed multi-node execution and scaling. Source code, measurements, and reproduction instructions are available at https://github.com/NextLinearAlgebra/qr-omega.

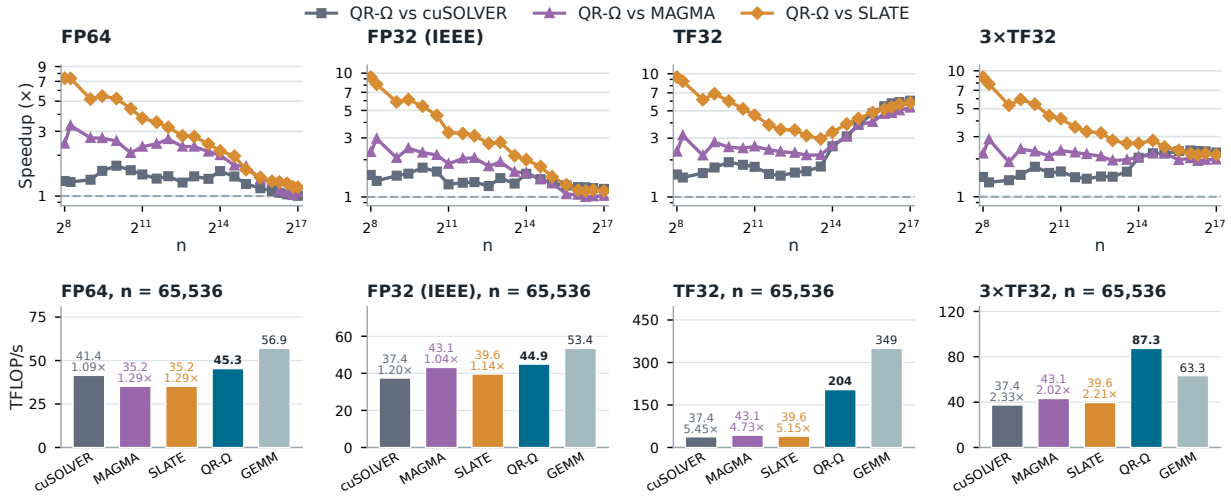


Figure 1: One H200 GPU. Top: QR-Ω speedup relative to each tuned reference across measured sizes. Bottom: throughput at $n = 65{,}536$, with speedups below the reference values and a measured same-size GEMM comparison. TF32 and 3×TF32 use the references' FP32 results. GEMM is an empirical comparator, not a hardware roofline.

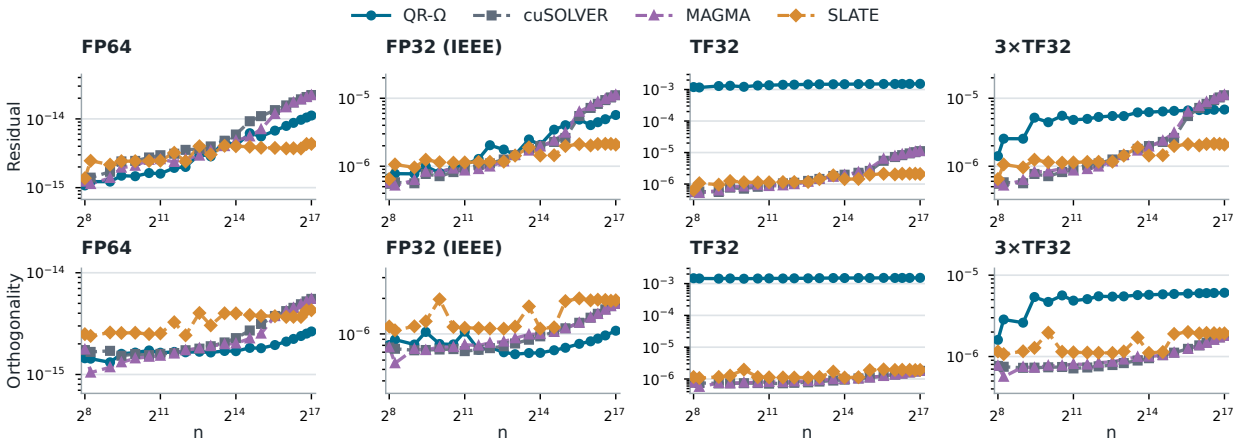


Figure 2: One H200 GPU: maximum relative reconstruction residual (top) and orthogonality error (bottom) for the timed configurations. Reference FP32 curves are repeated in the tensor-mode panels. Numerical error and performance must be considered together.

References

[1] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. 2012. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM J. Sci. Comput.* 34, 1, A206–A239. https://doi.org/10.1137/080731992

[2] Jack Dongarra, Mathieu Faverge, Thomas Hérault, Mathias Jacquelin, Julien Langou, and Yves Robert. 2013. Hierarchical QR factorization algorithms for multi-core clusters. *Parallel Comput.* 39, 4–5, 212–232. https://doi.org/10.1016/j.parco.2013.01.003

[3] Robert Schreiber and Charles Van Loan. 1989. A storage-efficient WY representation for products of Householder transformations. *SIAM J. Sci. Stat. Comput.* 10, 1, 53–57. https://doi.org/10.1137/0910005

[4] Edgar Solomonik and James Demmel. 2011. Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms. In *Euro-Par 2011*, 90–109. https://doi.org/10.1007/978-3-642-23397-5_10

[5] Hiroyuki Ootomo and Rio Yokota. 2022. Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance. *Int. J. High Perform. Comput. Appl.* 36, 4, 475–491. https://doi.org/10.1177/10943420221090256