Comparing Hybrid CPU-GPU and Native GPU-only Acceleration for Linear Algebra

Mark Gates, Stan Tomov, Azzam Haidar

SIAM LA — Oct 29, 2015

INNOVATIVE COMPUTING LABORATORY

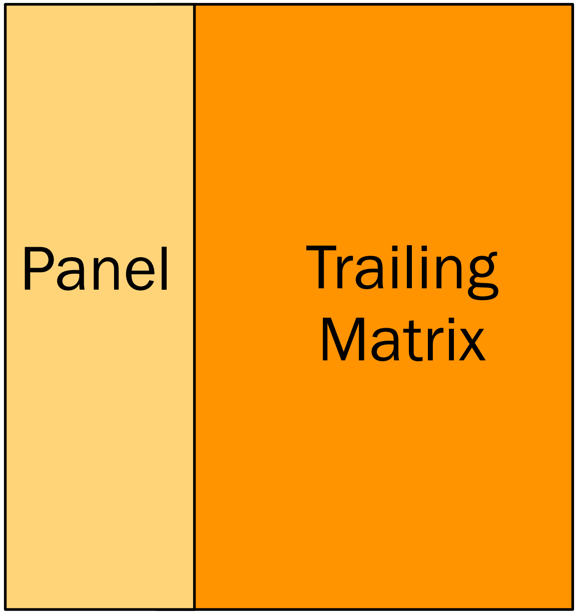
THE UNIVERSITY OF TENNESSEE KNOXVILLE

Overview

- Dense linear algebra algorithms
- Hybrid CPU–GPU implementation
- GPU–only implementation
- Case studies:
  - QR factorization
  - QR with column pivoting
  - Hessenberg reduction

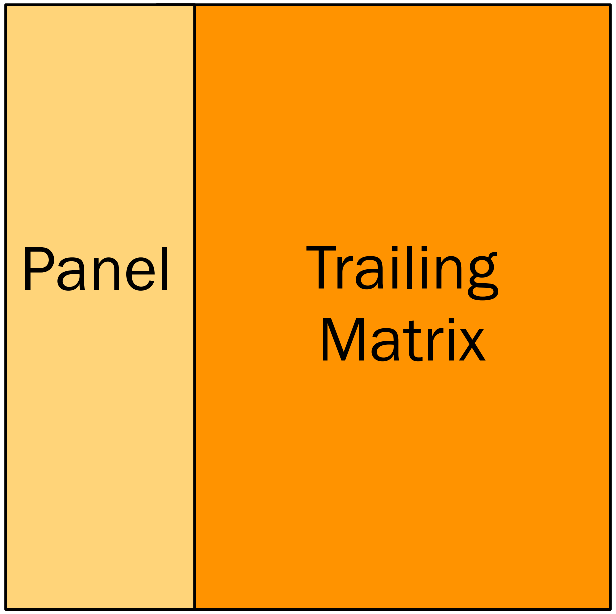
Linear algebra routines

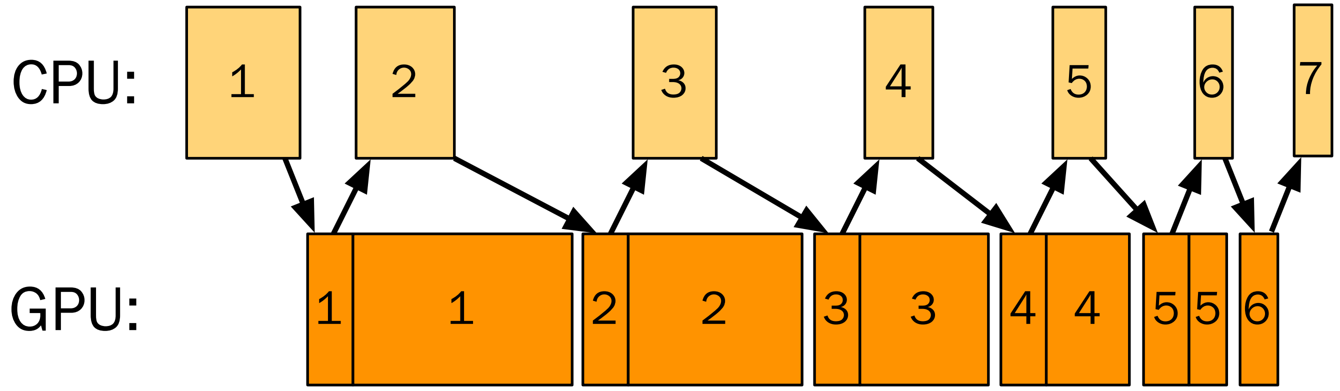
- Iterate two steps:
  - Panel factorization
    - Level 1–2 BLAS
    - Control flow
    - Data dependent (pivoting, etc.)
  - Trailing matrix update
    - Level 3 BLAS



Hybrid CPU–GPU algorithms

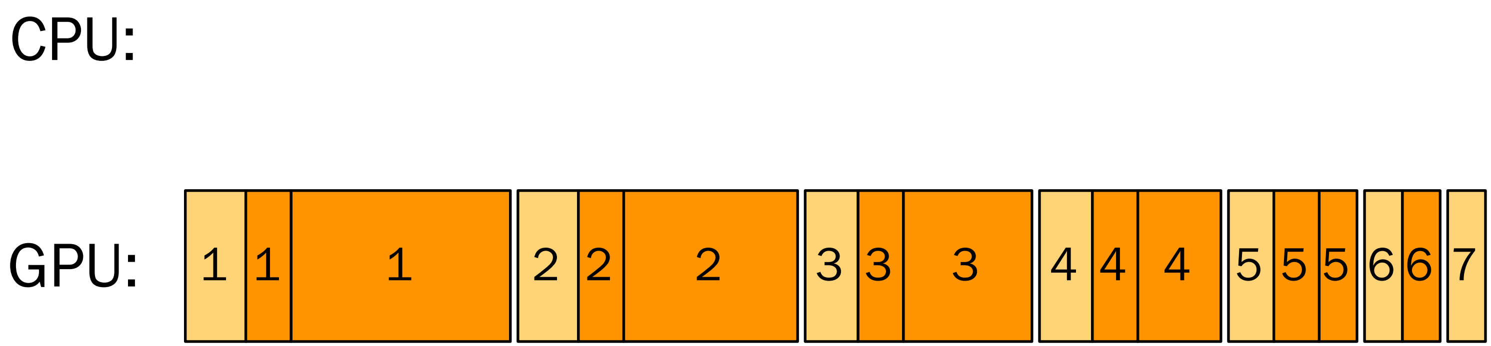
- Assign panel to CPU
- Assign trailing matrix to GPU
- Communicate panel from CPU <=> GPU
- Overlap next panel during trailing matrix update





GPU–only algorithms

- Assign both panel and trailing matrix to GPU
- No CPU <=> GPU communication
- CPU available for other tasks
- No overlap
  - Some algorithms don't allow overlap anyhow

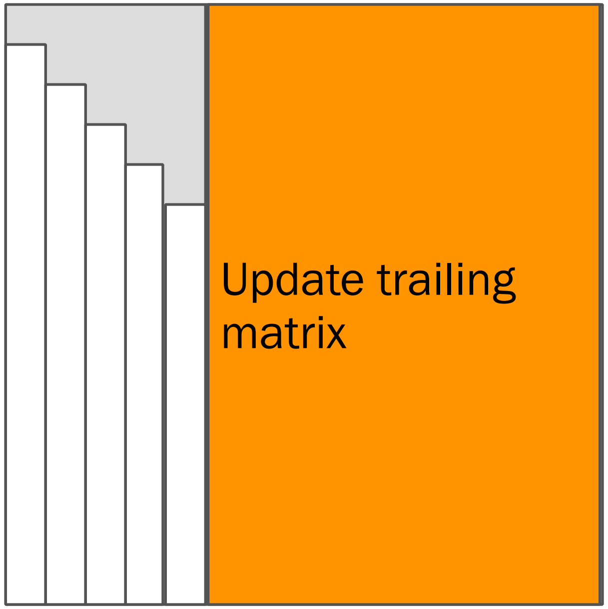


Householder–based algorithms

- QR factorization (geqrf)
  - $A = QR$
  - Least squares, etc.
- QR with column pivoting (geqp3)
  - $AP = QR$
  - More stable, esp. for rank-deficient matrices
- Hessenberg reduction (gehrd)
  - $Q^H A Q = H$
  - Non-symmetric eigenvalues

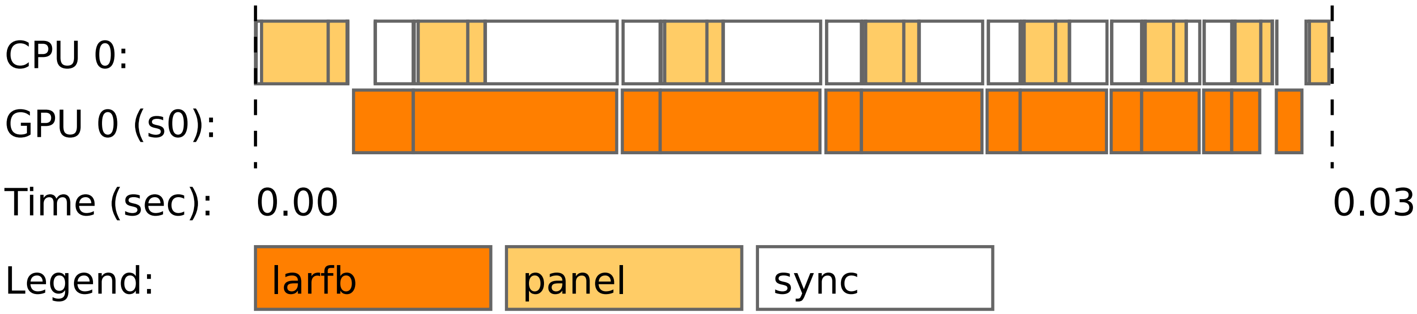
QR factorization

- Panel (nb columns)
  - for each column
    - apply previous reflectors
    - annihilate entries below diagonal
- Trailing matrix
  - update next panel (look-ahead)
  - update rest of A
- Overlap next panel & trailing matrix update

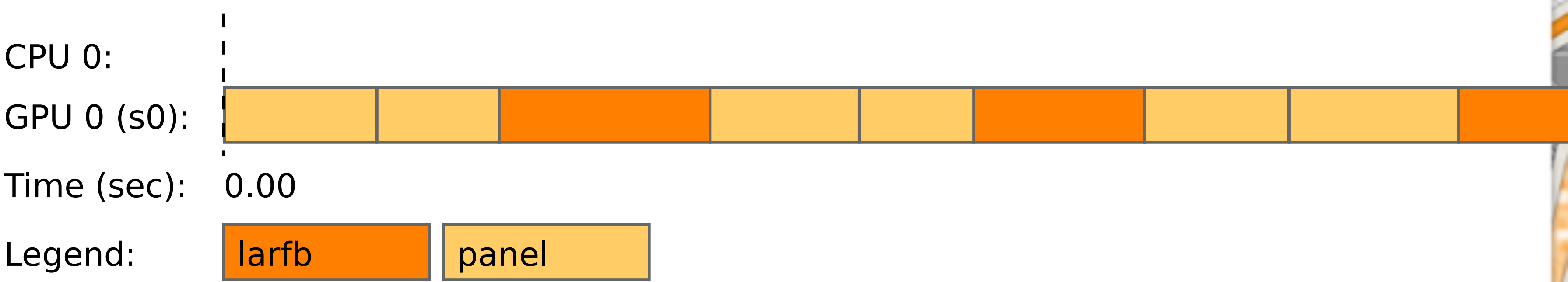


Execution trace

- Hybrid CPU–GPU

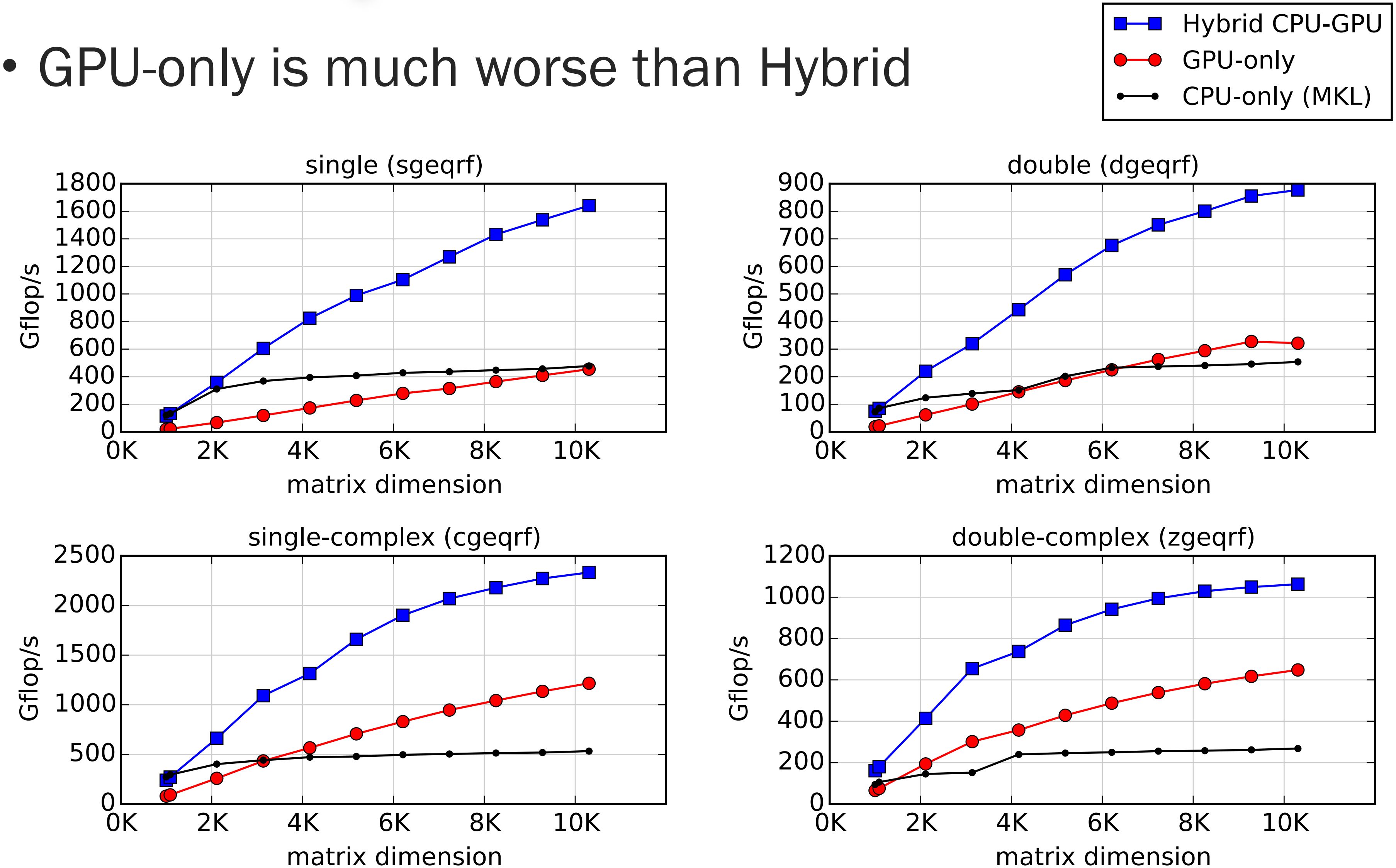


- GPU–only



Results: QR

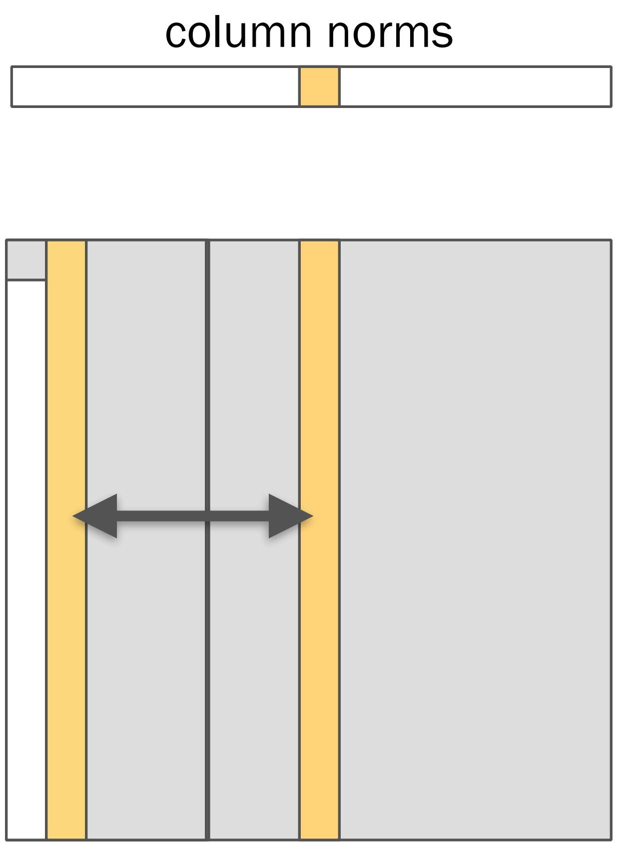
- GPU-only is much worse than Hybrid



2 x 8 core Intel Sandy Bridge E5-2670, NVIDIA K40c

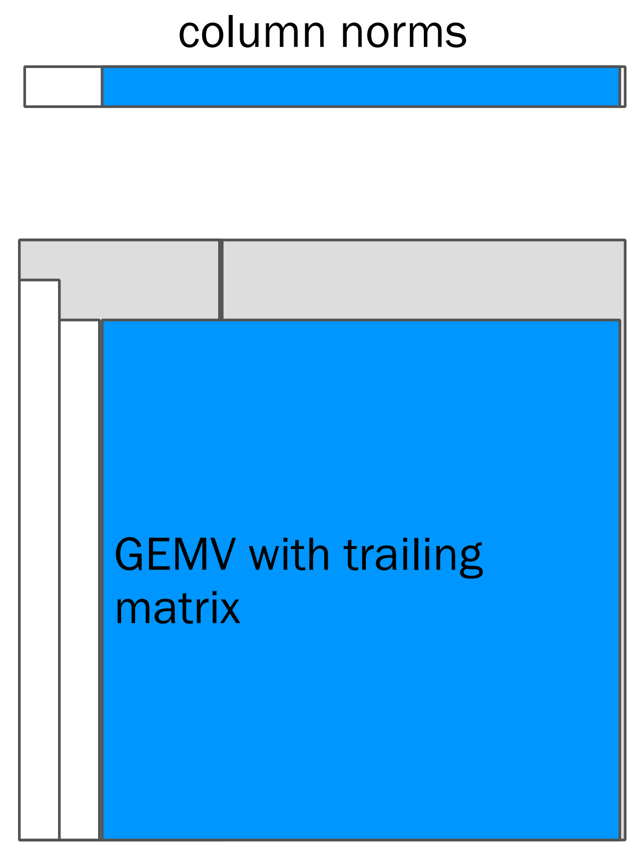
QR with column pivoting

- Compute column norms
- Panel (nb columns)
  - for each column
    - swap with column of max norm
    - apply previous reflectors
    - annihilate entries below diagonal
    - GEMV with trailing matrix on GPU
    - update column norms
- Trailing matrix
  - update rest of A
- Dependencies prevent overlap



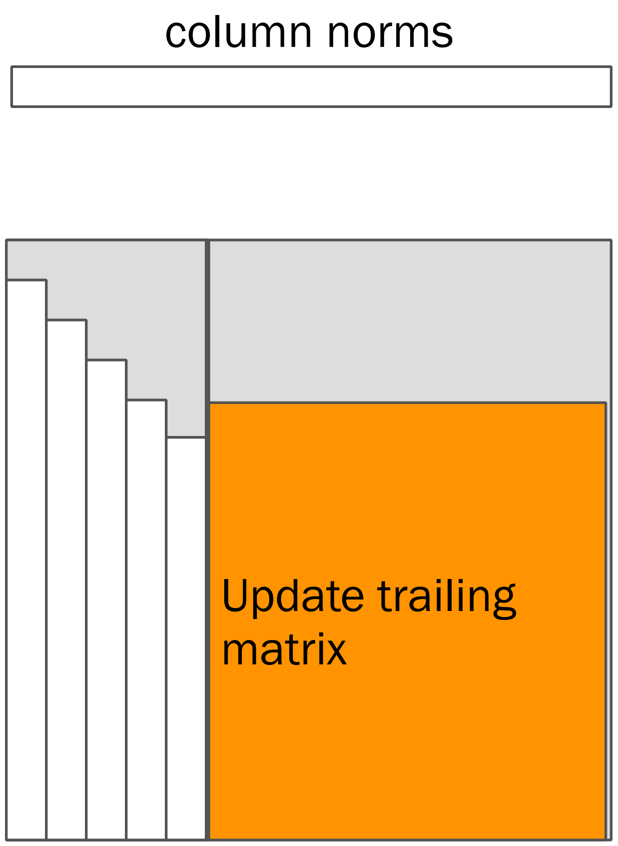
QR with column pivoting

- Compute column norms
- Panel (nb columns)
  - for each column
    - swap with column of max norm
    - apply previous reflectors
    - annihilate entries below diagonal
    - GEMV with trailing matrix on GPU
    - update column norms
- Trailing matrix
  - update rest of A
- Dependencies prevent overlap



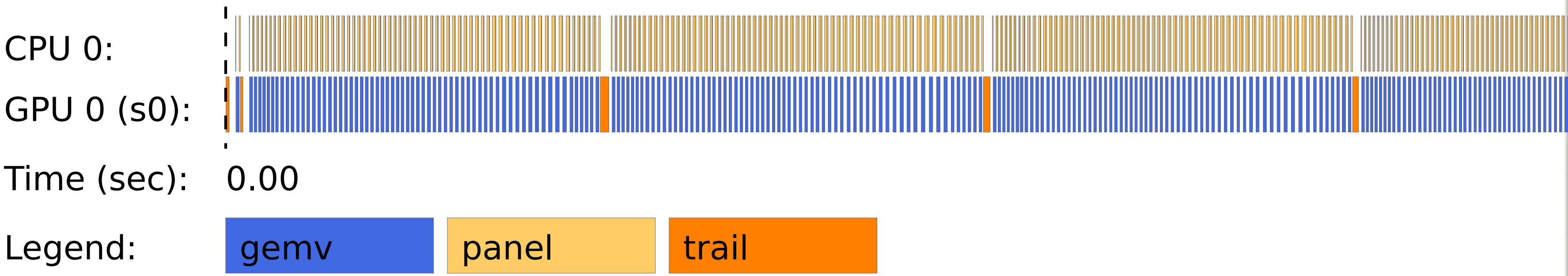
QR with column pivoting

- Compute column norms
- Panel (nb columns)
  - for each column
    - swap with column of max norm
    - apply previous reflectors
    - annihilate entries below diagonal
    - GEMV with trailing matrix on GPU
    - update column norms
- Trailing matrix
  - update rest of A
- Dependencies prevent overlap

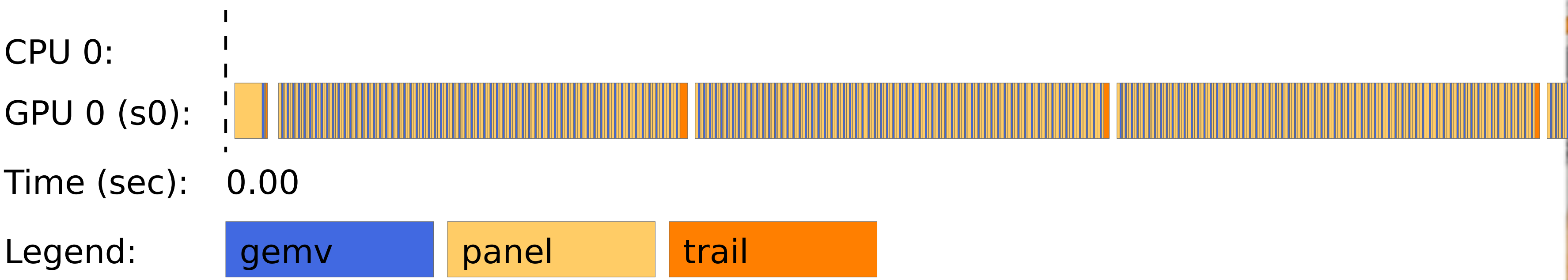


Execution trace

- Hybrid CPU–GPU

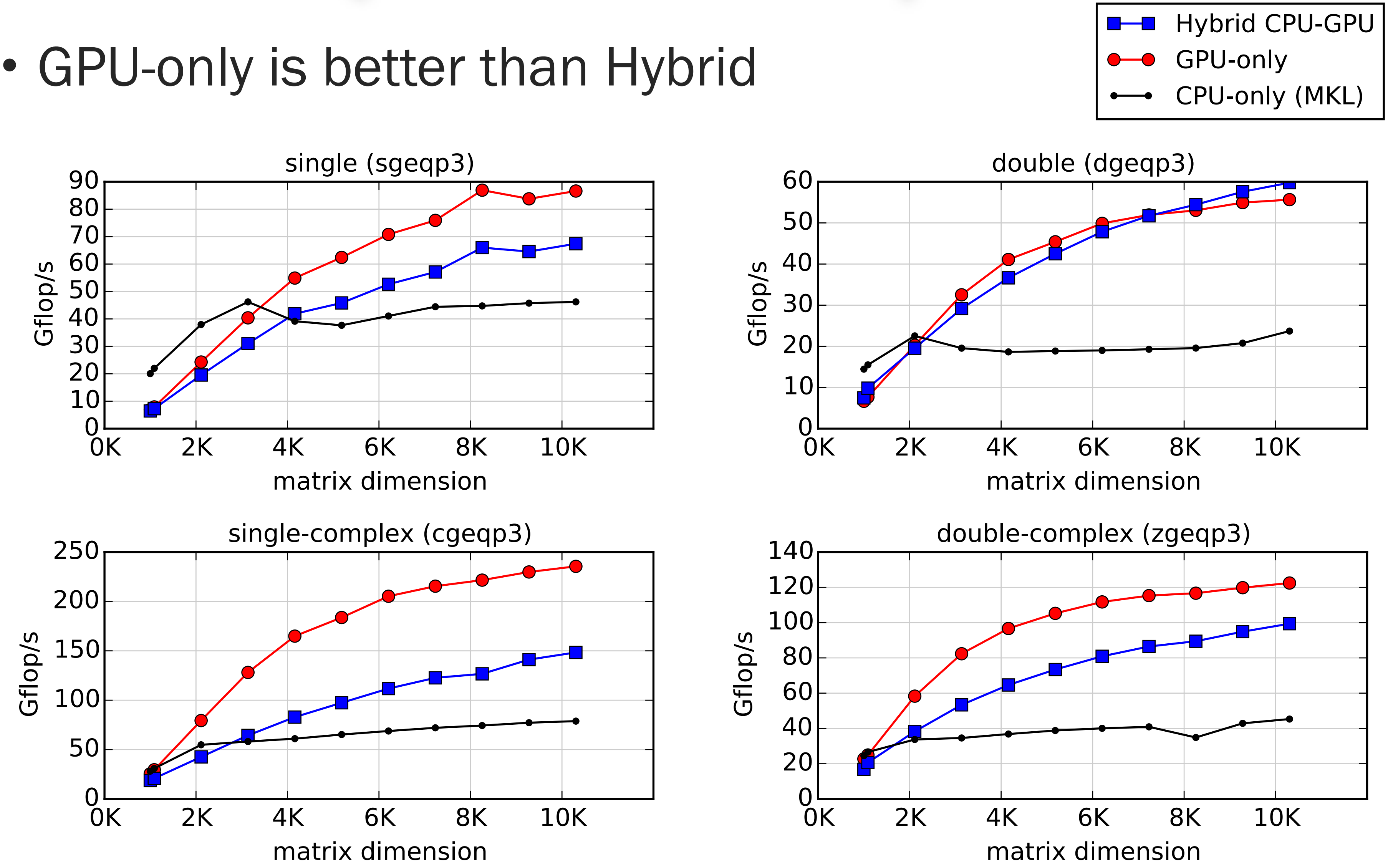


- GPU–only



Results: QR with column pivoting

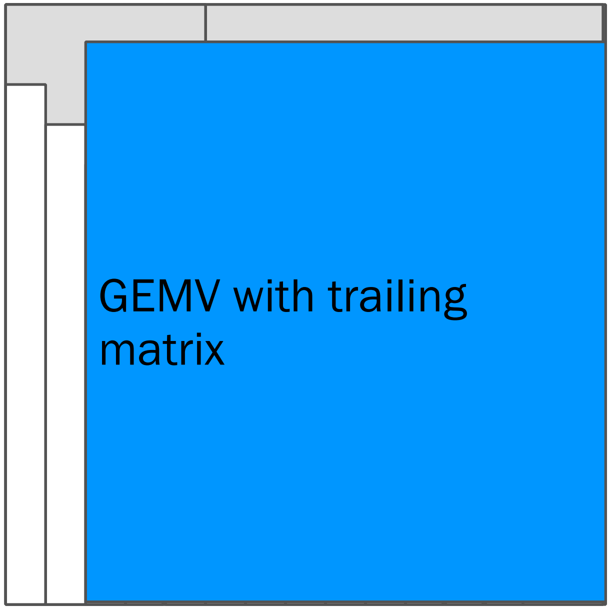
- GPU-only is better than Hybrid



2 x 8 core Intel Sandy Bridge E5-2670, NVIDIA K40c

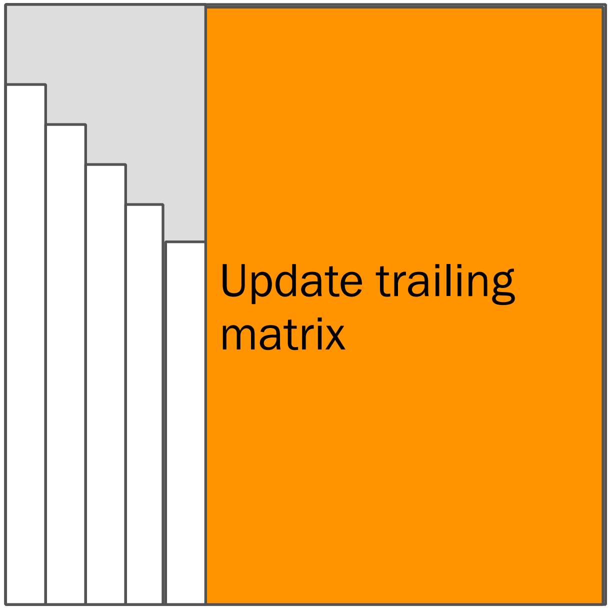
Hessenberg reduction

- Panel (nb columns)
  - for each column
    - apply previous reflectors (from right and left)
    - annihilate entries below sub-diagonal
    - GEMV with trailing matrix on GPU
- Trailing matrix
  - update rest of A from right and left
- Dependencies prevent overlap



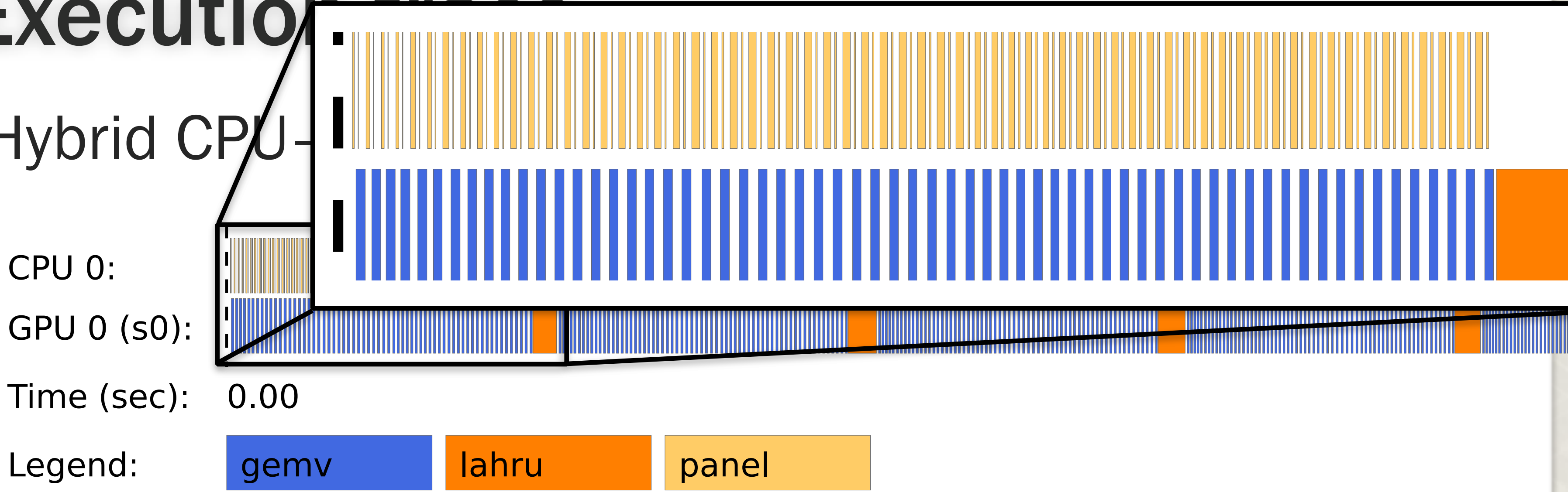
Hessenberg reduction

- Panel (nb columns)
  - for each column
    - apply previous reflectors (from right and left)
    - annihilate entries below sub-diagonal
    - GEMV with trailing matrix on GPU
- Trailing matrix
  - update rest of A from right and left
- Dependencies prevent overlap

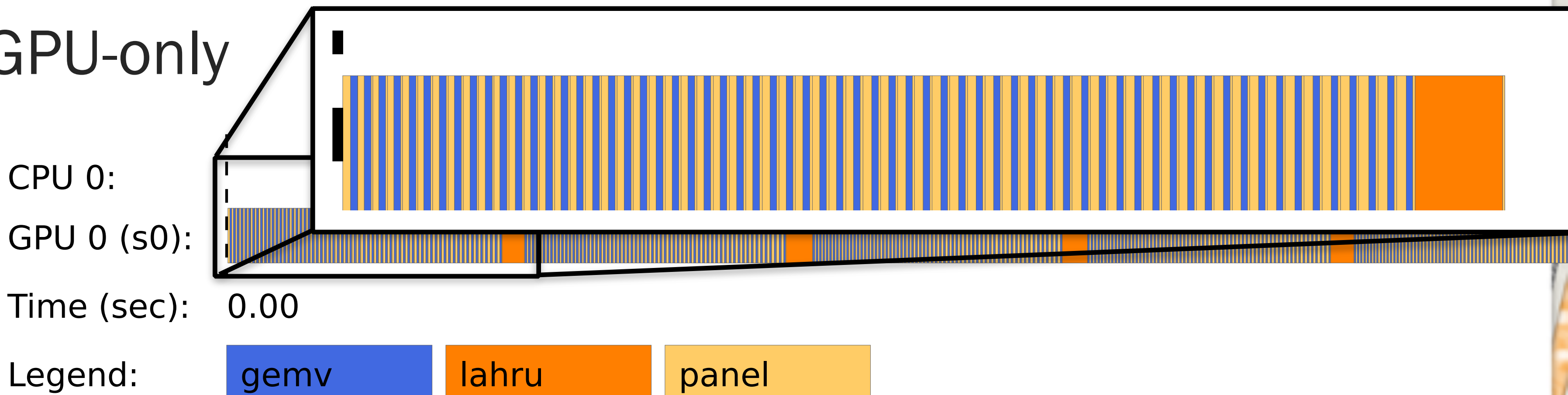


Execution trace

- Hybrid CPU-

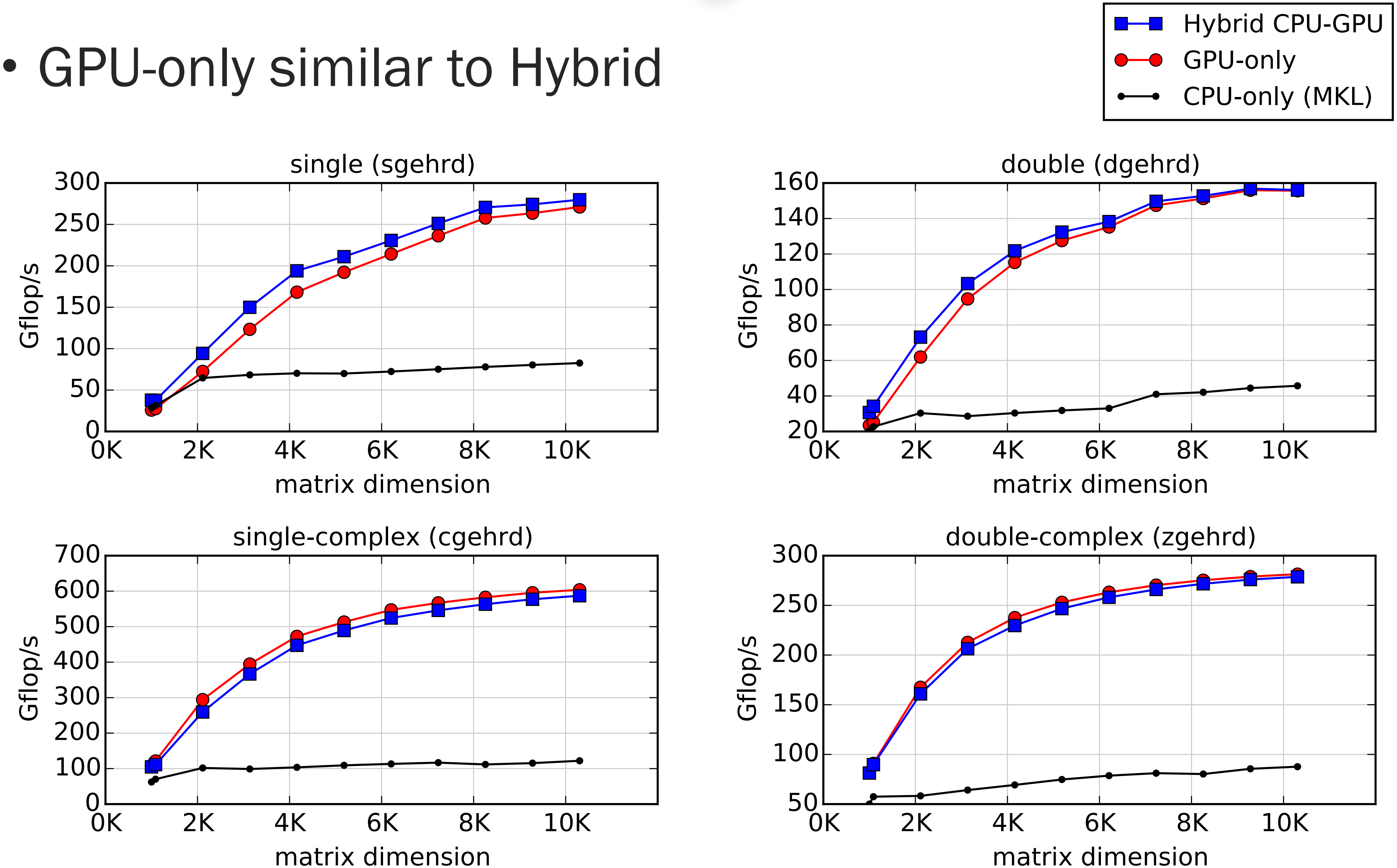


- GPU-only



Results: Hessenberg

- GPU-only similar to Hybrid



2 x 8 core Intel Sandy Bridge E5-2670, NVIDIA K40c

ICL THE UNIVERSITY OF TENNESSEE KNOXVILLE

GPU–only kernels & optimizations

- Householder reflectors
  - Generate — vector norm and scaling (larfg)
    - save extra copies of tau in T, etc.
  - Apply — dot product and axpy (larf)
- Custom norm update for QR with pivoting
- Optimized gemv
  - Tall matrix transposed * vector: $V^T a_j$
- Use gemv, faster than trmv
  - Store V and T with explicit 0's and 1's
  - Merge trmv+gemv into one gemv

Lessons Learned

- Panels
  - Lack parallelism
  - Significant control flow
  - Many separate function calls
    - Perform poorly on GPUs
  - Requires programming custom GPU kernels
  - Merge kernels together to reduce overheads

- GPU-only reduces communication
  - Modest win for QR with pivoting
  - No improvement for Hessenberg

Thank you

MAGMA

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

NVIDIA CUDA CENTER OF EXCELLENCE

The MathWorks

U.S. DEPARTMENT OF ENERGY

NSF