Towards Half-Precision Computation for Complex Matrices: A Case Study for Mixed-Precision Solvers on GPUs

Ahmad Abdelfattah, Stan Tomov, and Jack Dongarra

Innovative Computing Laboratory
University of Tennessee, Knoxville

2019 ScalA Workshop, Denver, CO

Monday, November 18th, 2019

INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Outline

Outline

1. Introduction
2. Matrix Multiplication with Half-complex Precision
3. Mixed-precision Factorization and Solve
4. Final Performance
5. Conclusion

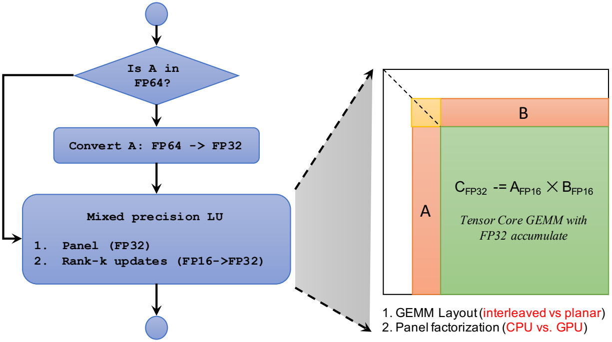
What are trying to solve?

- Solve a linear system of equation ($Ax = b$)
- A is a general square matrix in single-complex precision
- Use half-precision to accelerate the solution (mixed-precision solver)
  - Accuracy is recovered using Iterative Refinement (IR)
- Similar algorithm to
  - *Carson and Higam: Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions, SIAM SISC, 2018*
  - *Haidar et al.: Harnessing GPU Tensor Cores for Fast FP16 Arithmetic to Speed Up Mixed-precision Iterative Refinement Solvers, SC'18*
- No native support for half-complex computation

This work is part of the MAGMA library

What are the steps?

1. **Mixed-precision LU factorization with partial pivoting**
2. Iterative refinement using preconditioned GMRES

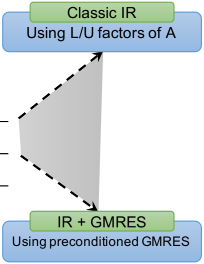


1. GEMM Layout (interleaved vs planar)
2. Panel factorization (CPU vs. GPU)

What are the steps?

1. Mixed-precision LU factorization with partial pivoting
2. **Iterative refinement using preconditioned GMRES**

(1) Compute residual	$r = b - Ax$
(2) Solve for c	$Ac = r$
(3) Update solution	$x_{i+1} = x_i + c$
(4) Repeat 1-3 until converged	



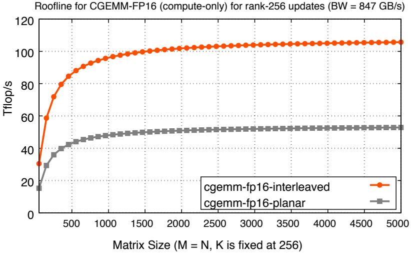
Outline

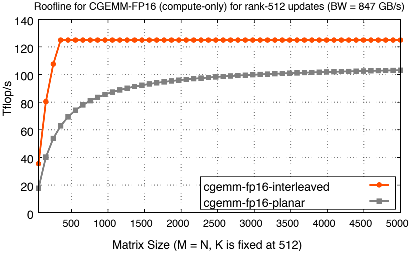
Mixed-precision Matrix Multiplication

- $C_{FP32} = C_{FP32} - A_{FP16} \times B_{FP16}$
- A and B are originally in FP32, but are converted to half-complex
- No native support for half-complex kernels
  - cuBLAS provides only real arithmetic FP16 kernels
  - cuBLASLt and CUTLASS suggest using split-complex kernels using planar layout
- But interleaved layout is a better alternative
  - Used by almost all linear algebra algorithms
  - Better operational intensity (i.e. flops/bytes ratio)

Mixed-precision Matrix Multiplication

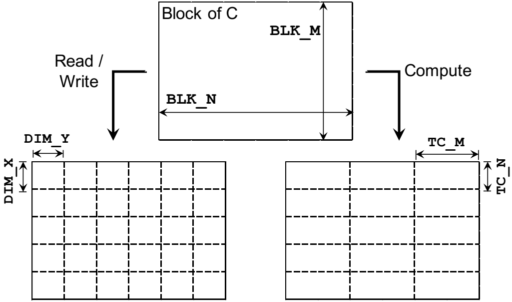
- Goal: `CGEMM` kernel with FP16 acceleration
- Using cuBLAS (planar-layout)
  - Four calls to `cublasGemmEx`: arith. intensity = $\frac{mnk}{4mn+k(m+n)}$
  - Overhead of splitting and merging real/imaginary components
- Using a new MAGMA kernel (interleaved-layout)
  - One kernel call: arith. intensity = $\frac{2mnk}{4mn+k(m+n)}$
  - No overheads





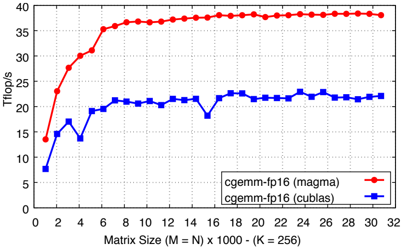
MAGMA's New Kernel (CGEMM-FP16-Interleaved)

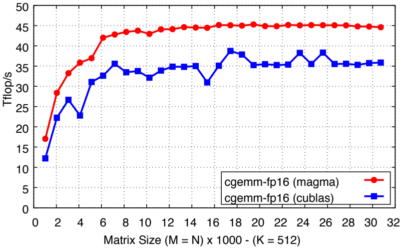
- Input: *A* and *B* in half-complex precision (`half2`)
- Input/Output: *C* in single-complex precision
- An abstraction layer over the Tensor Cores (TCs)
  - Using `WMMA` device routines to manage TCs
  - Split and merge in shared memory
- Double-sided recursive blocking + auto-tuning



Performance of CGEMM-FP16

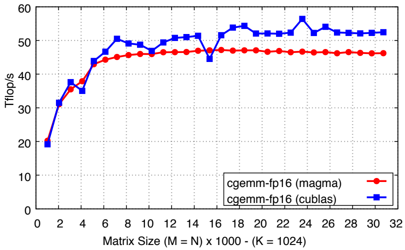
- Tested on Tesla V100-PCIe GPU, CUDA 10.1
- 2 advantages for MAGMA: arithmetic intensity and splitting/merging overhead
- 70% better than cuBLAS ($k = 256$), 24% if $k = 512$





Performance of CGEMM-FP16 “cont.”

- Tested on Tesla V100-PCIe GPU, CUDA 10.1
- MAGMA loses the advantage for $k \geq 800$
- Summary of results
  - Blocking size ≤ 800, use MAGMA
  - Otherwise, use cuBLAS

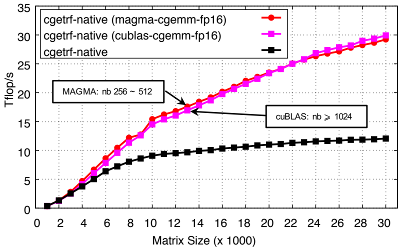


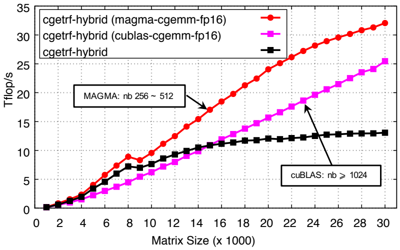
Let’s test both!

Outline

Performance of the mixed-precision LU factorization

- Panel factorization: CPU (hybrid) vs. GPU (native)?
- Larger *nb* means more time spent in single-complex precision (i.e. the panel)
- Use hybrid-LU only if the matrix is larger than 22*k* (with MAGMA CGEMM-GP16)

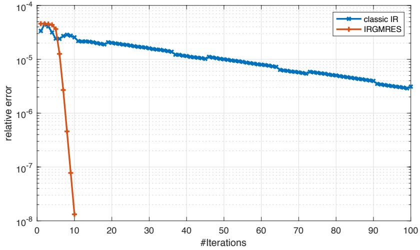




Performance of the mixed-precision LU factorization using native (left) and hybrid (right) executions. Results are shown on a 20-core Haswell CPU and a Tesla V100 GPU. MAGMA is built using CUDA 10.1 and MKL 2018.0.1

Classic IR vs. IR + GMRES

- GMRES is more stable for solving the correction equation ($Ac = r$)
- GMRES is preconditioned by the low precision factors of A
- Much faster conversion than classic IR
- Based on FGMRES implementation by Yousef Saad (ZITSOL)

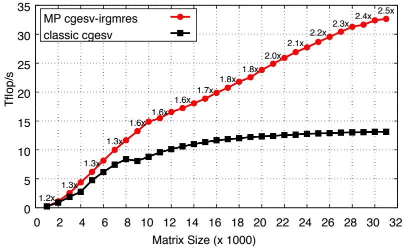


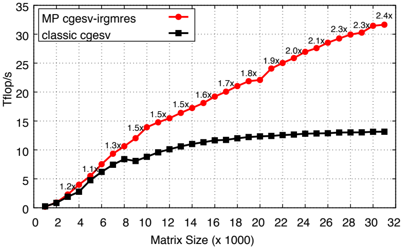
Convergence history of both `IR` *and* `IRGMRES` *on a matrix of size 20k.* $k_\infty(A) = 10^5$. *Clustered distribution of singular values* ($\sigma_i = 1, 1, \cdots, \frac{1}{k_\infty(A)}$).

Outline

Final Performance: Mixed-precision CGESV-IRGMRES

- Performance is up to 2.5×, but depends on the matrix properties





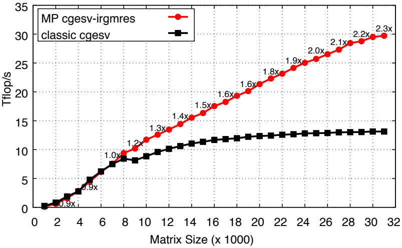
System: 20-core Intel Haswell CPU, Tesla V100 GPU - using MKL 2018.0.1 and CUDA 10.1

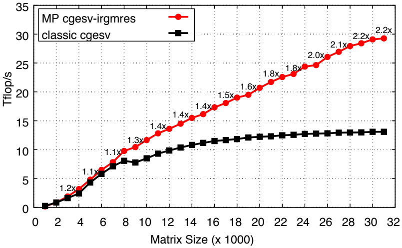
Left: Diagonally dominant matrices. $k_\infty(A) \leq 10^2$.

Right: Matrices with positive eigenvalues and arithmetic distribution of singular values $(\sigma_i = 1 - (\frac{i-1}{n-1})(1 - \frac{1}{k_\infty(A)}))$, $k_\infty(A) \approx 4.3e{+}5$

Final Performance: Mixed-precision CGESV-IRGMRES

- Performance is up to 2.5×, but depends on the matrix properties





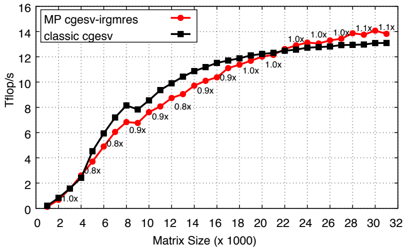
System: 20-core Intel Haswell CPU, Tesla V100 GPU - using MKL 2018.0.1 and CUDA 10.1

Left: Positive eigenvalues and logarithmic uniform distribution of singular values ($\log(\sigma_i)$ uniform between $\log(\frac{1}{k_\infty(A)})$ and $\log(1)$), $k_\infty(A) \approx 1e{+}5$

Right: Clustered singular values ($\sigma_i = 1, 1, \cdots, \frac{1}{k_\infty(A)}$), $k_\infty(A) \approx 4.3e{+}4$

Sometimes it does not pay off

- The refinement steps can consume all the performance advantage of the factorization



System: 20-core Intel Haswell CPU, Tesla V100 GPU - using MKL 2018.0.1 and CUDA 10.1

Arithmetic distribution of singular values $(\sigma_i = 1 - (\frac{i-1}{n-1})(1 - \frac{1}{k_\infty(A)}))$, $k_\infty(A) \approx 4.3e{+}4$.

Outline

Summary

1. FP16 arithmetic is not only for machine learning
2. New family of mixed-precision linear solvers
3. Half-complex precision accelerates single-complex systems by factors up to 2.5$\times$
4. Next: Double-complex systems, matrix scaling, other half-complex kernels

- https://icl.utk.edu/magma/
- Release expected by Spring 2020

Thank You!