Accelerating 2D FFT: Exploit GPU Tensor Cores through Mixed-Precision

Xiaohe Cheng, Anumeena Sorna, Eduardo D'Azevedo (Advisor), Kwai Wong (Advisor), Stanimire Tomov (Advisor)

Hong Kong University of Science and Technology, National Institute of Technology, Oak Ridge National Laboratory, University of Tennessee

Overview

- 2D FFT in HPC applications
  - Frequency domain analysis
  - Quantum cluster simulations
- Large volume and high parallelism
  - Exploit modern parallel architectures
  - Graphics Processing Units (GPUs)
  - Nvidia CUDA
- cuFFT library: current state of the art, but can **NOT** benefit from the FP16 arithmetic on recent hardware due to accuracy limitations

Operation	Acceleration
GEMM	320%
FFT FP16	17.02%
FFT FP32	12.33%

- cuFFT does not achieve the same level of acceleration as cuBLAS GEMM

- Results: *Tensor Core* accelerated FFT & improved accuracy
  - Straightforward CUDA implementation costs **~2.5x** time of cuFFT32
  - Error within 10^{-4}, **1000x** better than cuFFT16

Motivation

- Mixed-precision methods benefit both computation and memory
- Tensor cores on new GPU architecture
  - Matrix-multiply-and-accumulate units with throughput up to 125 TFLOPS
  - Multiply Inputs: FP16 (half type) only

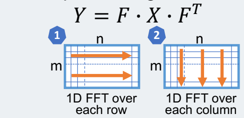
- FFT properties: linearity, numerical stability, intensive matrix multiplications

Our novel implementation that exploits tensor cores by dynamically splitting a FP32 input into two FP16 operands

Our Proposed Approach

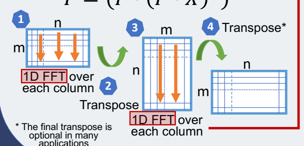
- Implementing 2D FFT

$$Y = F \cdot X \cdot F^T$$

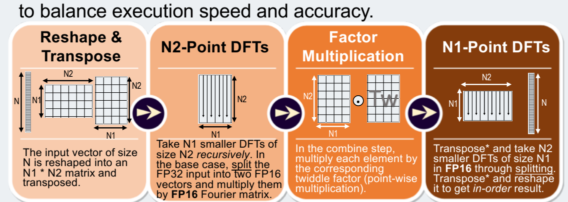


- To utilize column major 1D FFT routine

$$Y = (F \cdot (F \cdot X)^T)^T$$

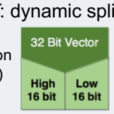


- 1D FFT: Apply Cooley–Tukey algorithm, choose N1 = 4 (radix-4) to balance execution speed and accuracy.



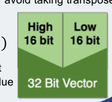
* In implementation we modify the DFT kernel to avoid taking transpose.

- Mixed precision DFT: dynamic splitting
  - Linearity of FFT allows the separate computation of FFT(X_{hi}) and FFT(X_{lo}) in half precision



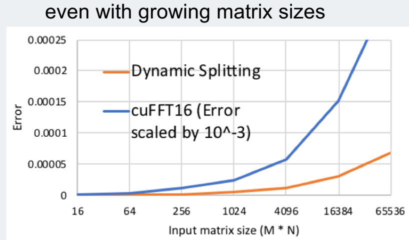
$$X(:) = \alpha X_{hi}(:) + \beta X_{lo}(:)$$

α – infinity norm of input
β – infinity norm of residue



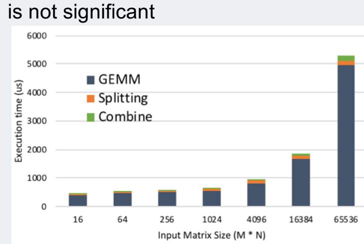
Experimental Results

- The method preserves high accuracy, even with growing matrix sizes



*The relative error of 2D FFT at different input sizes (horizontal dimension * vertical dimension), using our implementation and half precision cuFFT.*

- The cost of dynamic splitting and combine is not significant



The execution time breakdown at different input sizes. About 90% of total time is spent on matrix multiplication.

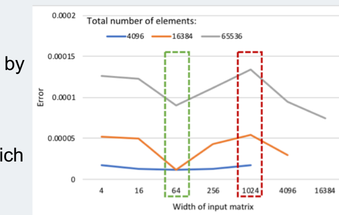
- The implementation can handle a wide range of inputs and produce accurate results

Data range	$\pm 10^{-7}$	$\pm 10^{-2}$	± 1.0	$\pm 10^{2}$	$\pm 10^{4}$	$\pm 10^{9}$
cuFFT16	59.088	5.818%	5.602%	5.783%	N/A	N/A
Splitting	0.002%	0.001%	0.001%	0.001%	0.001%	0.001%

Relative error at different data ranges.

Additional Observations

- For fixed number of input elements, the accuracy is affected by the shape of matrix. Particular matrix dimensions lead to higher accuracy, which can be exploited by FFT applications.



Conclusions & Future Work

- Our dynamic splitting method computes 2D fast Fourier transform efficiently by utilizing the hardware advancement in half-precision floating-point arithmetic
- The implementation effectively emulates single precision calculation, and produces highly accurate results from a variety of inputs
- The speed of current cuBLAS-based implementation is inferior to cuFFT library, but optimizations are available:
  - Tiled matrix transpose via GPU shared memory
  - Pre-computation of twiddle factors
  - Combination of real and imaginary operations
- Input-aware auto-tuning splitting algorithm is to be designed to support ill-conditioned inputs. It may further improve execution speed and accuracy.

Acknowledgements & References

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at University of Tennessee Knoxville. This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation. In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team. This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.

[1] Kumar, Vipin, et al. Introduction to parallel computing: design and analysis of algorithms, Vol. 400. Ch. 10. Redwood City: Benjamin/Cummings, 1994.
[2] Staar, Peter, et al. "Taking a quantum leap in time to solution for simulations of high-Tc superconductors." High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for. IEEE, 2013.
[3] Göddeke, Dominik, Robert Strzodka, and Stefan Turek. "Performance and accuracy of hardware-oriented native-, emulated-and mixed-precision solvers in FEM simulations." International Journal of Parallel, Emergent and Distributed Systems 22.4, 2007, pp. 221-256.
[4] Appleyard and Yokim, Programming Tensor Cores in CUDA 9, NVIDIA Developer Blog, Oct 2017.