heFFTe: Highly Efficient FFTs for Exascale Systems

Ahmad Abdelfattah, Natalie Beams
Innovative Computing Laboratory
University of Tennessee, Knoxville

MUG'25, Columbus, OH
August 18-20, 2025

Credit to the heFFTe team

- Natalie Beams (UTK)
- Miroslav Stoyanov (ORNL)
- Alan Ayala (UTK → AMD)
- Stan Tomov (UTK → NVIDIA)
- Azzam Haidar (UTK → NVIDIA)
- Jack Dongarra (UTK)
- Sebastien Cayrols (UTK → NVIDIA)
- Jiali Li (UTK)
- George Bosilca (UTK → NVIDIA)
- Veronica Montanaro (ETH)
- Sonali Mayani (ETH)
- Andreas Adelmann (ETH)
- students and outside collaborators

Credit to the heFFTe team

- Stan Tomov (UTK → NVIDIA)
  (1971 – 2024)

STATE-OF-THE-ART GPU NUMERICAL COMPUTING
HONORING STAN TOMOV
APRIL 2025

Outline

1) High-level introduction to heFFTe as an efficient distributed implementation of multi-dimensional FFT
2) The status of heFFTe
3) heFFTe as a benchmark for MPI implementations
4) Performance results using MVAPICH
5) Future Directions

Fast Fourier Transform (FFT)

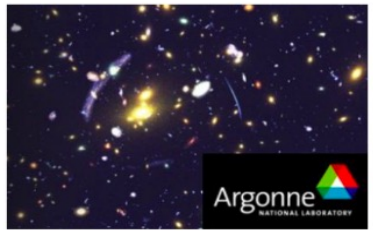
- FFT computes the Discrete Fourier Transform (DFT) of a series:
  Let $x = x_0, \ldots, x_{N-1}$ are complex numbers. The DFT of x is the sequence $X = X_0, X_1, \ldots, X_{n-1}$,such that:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N} \qquad k = 0, \ldots, N-1.$$

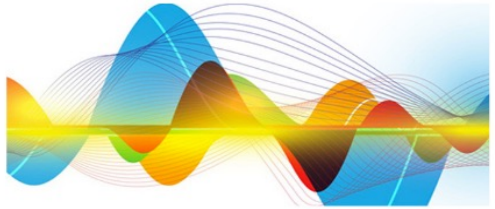
- DFT can be computed as a matrix-vector multiplication (GEMV) in $\mathcal{O}(N^2)$ FLOPs (memory-bound)
- FFT reduces the complexity to $\mathcal{O}(N \log_2 N)$ (even more memory-bound)
- The Inverse Discrete Fourier Transform (IDFT) is similarly defined except that the *'e'* exponents are taken as $(i\, 2\pi\, k\, n\, /\, N)$, and elements divided by N

FFT in the DOE's Exascale Computing Project (ECP)

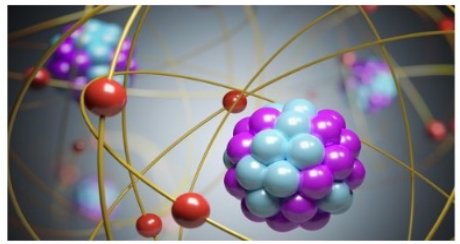
Cosmology
***ECP* ExaSky - HACC**



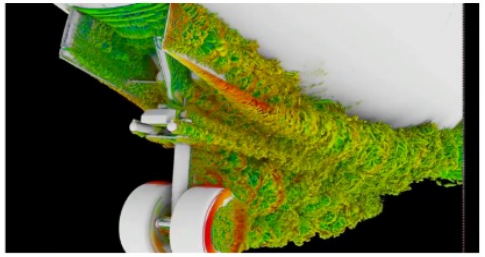
Signal processing,
***ECP* WARPX**



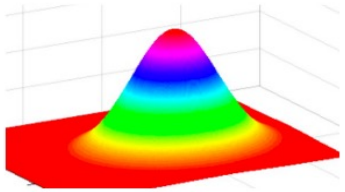
Deep Learning



Molecular Dynamics
***ECP* EXAALT**



Particle Simulations
***ECP* CoPa / Cabana**



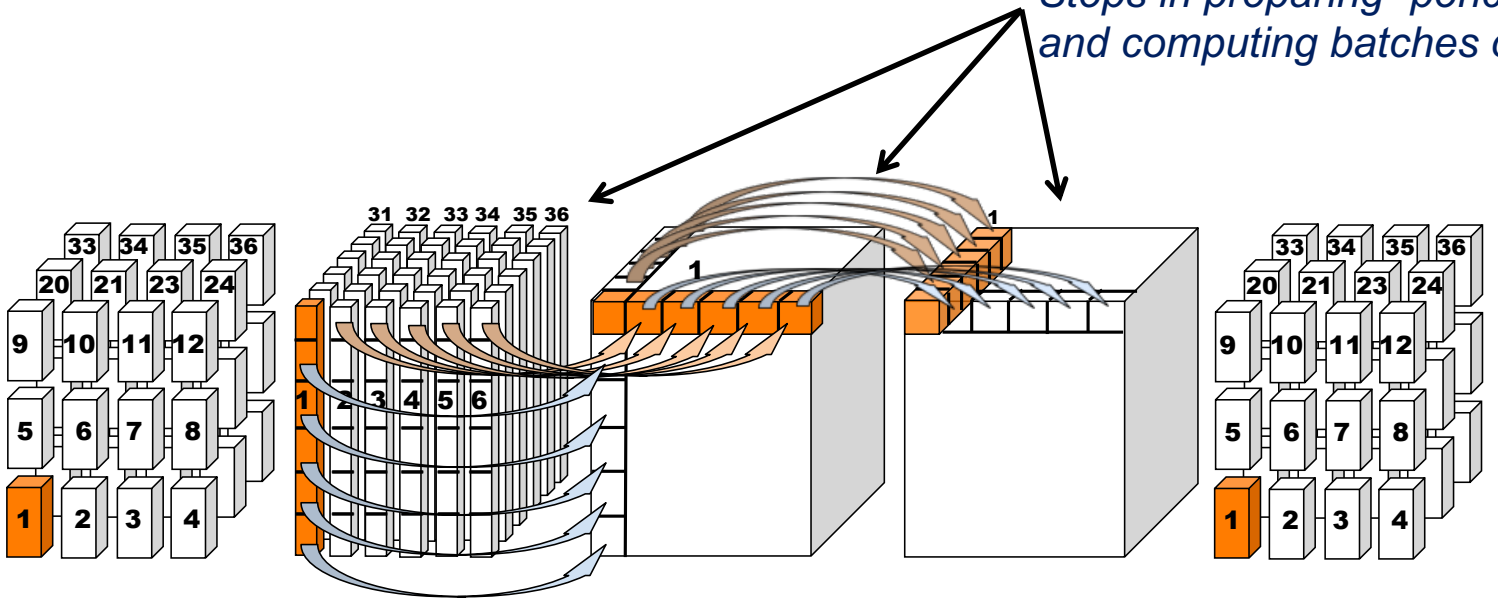
PDE solutions, **MASSIF**

- Some of these applications require multi-dimensional FFT at large scale
- Must run on DOE's ECP systems (i.e. use GPUs)

heFFTe: Highly Efficient Exascale FFT Library for Heterogeneous Architectures

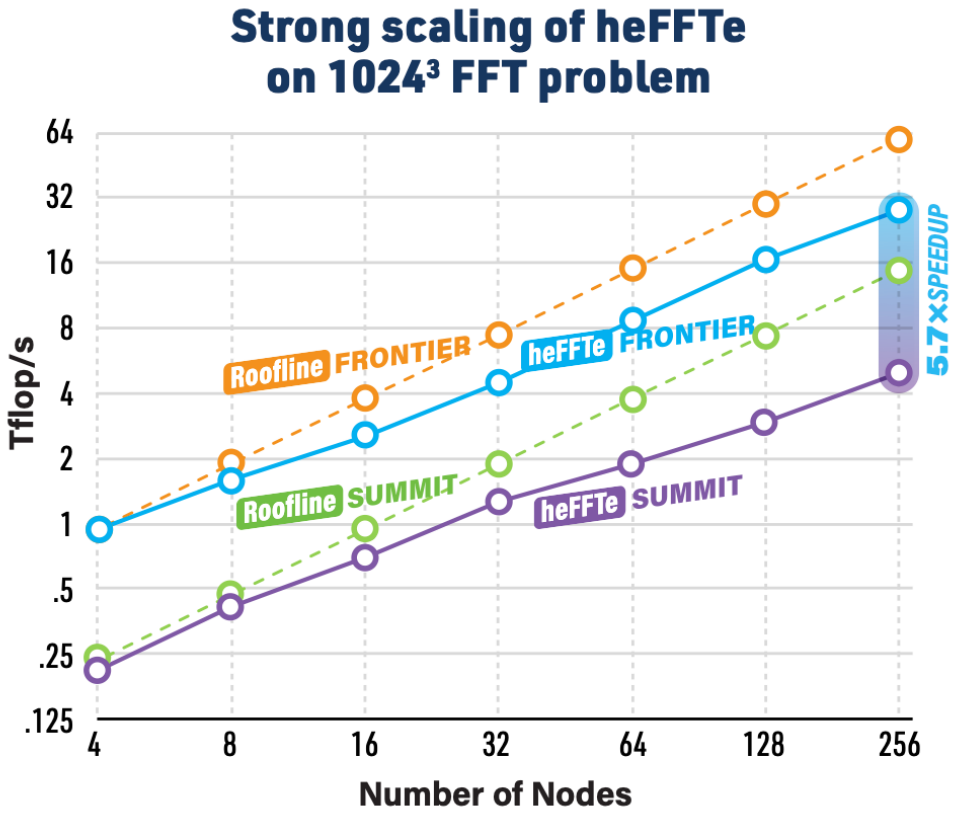
- 1D, 2D, and 3D distributed FFT library
  - GPU-enabled
  - Relies on single-node FFT libraries (FFTW, cuFFT, rocFFT, & oneMKL)
  - Simple data transposition/reshape kernels
  - MPI Communication

Steps in preparing "pencils" in the different dimensions and computing batches of 1D FFTs locally on each node/GPU



Status of heFFTe

- heFFTe-2.4.1 with support for CPUs, Nvidia, AMD, and Intel GPUs
- Capabilities:
  - Multidimensional FFTs
  - C2C, R2C, C2R
  - DCS, DST, and convolution
  - Batched FFTs
  - Multiple data layouts & communications patterns
- Open-source Software
  - Spack installation and integration in xSDK
  - Homepage: http://icl.utk.edu/fft/
    Repository: https://github.com/icl-utk-edu/heffte
- So far, no major developments after ECP closeout



heFFTe among Other Developments

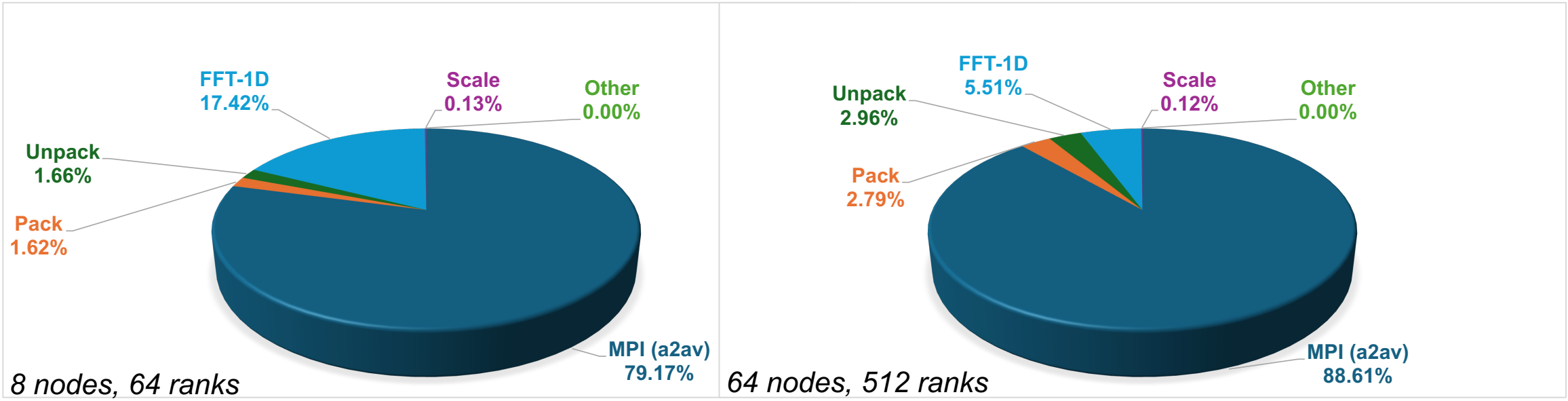
- Amongst the very few parallel FFT libraries that support GPUs, heFFTe provides unique functionalities that cover a large number of features from the state-of-the-art, making it ubiquitous for a wide range of applications

	Library	Pencil Decomp	Brick Decomp	Slab Decomp	Transpose Reshape	Stride Reshape	R2C Transform	Single precision	Mixed precision	Multiple backends	Nonblocking All-to-All
CPU	FFTW3	✓				✓	✓	✓			
CPU	FFTMPI	✓	✓		✓			✓		✓	
CPU	2DECOMP	✓				✓	✓				
CPU	SWFFT		✓		✓						
CPU	PFFT	✓			✓		✓				
CPU	P3DFFT	✓		✓	✓		✓	✓			✓
GPU	AccFFT	✓			✓	✓	✓	✓		✓	
GPU	FFTE	✓		✓	✓		✓	✓			
GPU	heFFTe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

heFFTe is heavily communication-bound

- Using heFFTe's own minimal tracing tool
- MPI calls dominate the execution time, especially on the GPU
- Any improvement in communication leads to huge performance gains

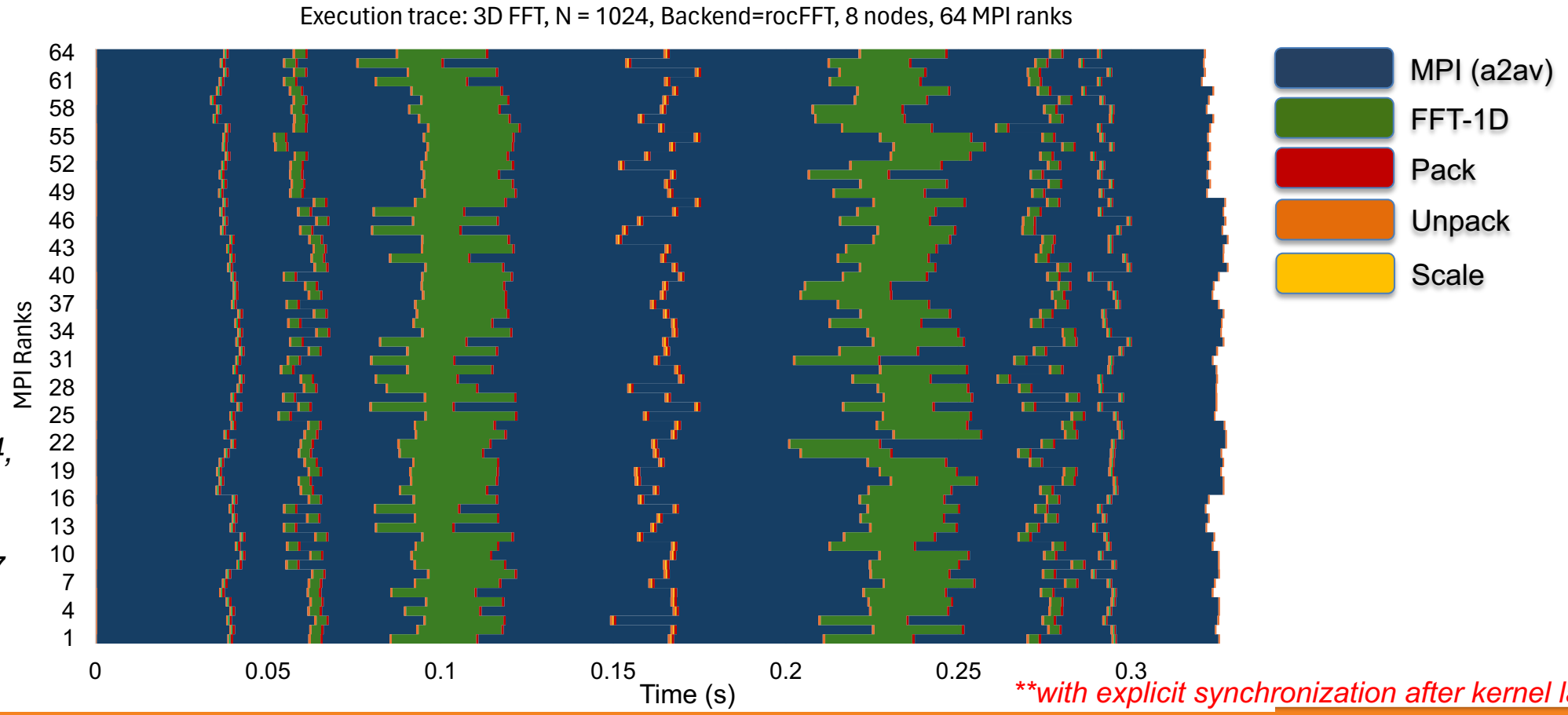
Time Breakdown of heFFTe on Frontier using rocFFT
3D FFT, N = 1024, FP64 (C2C), ROCm-5.6.0, cray-mpich/8.1.27



***with explicit synchronization after kernel launches*

heFFTe is heavily communication-bound

- Using heFFTe's own minimal tracing tool
- MPI calls dominate the execution time, especially on the GPU
- Any improvement in communication leads to huge performance gains

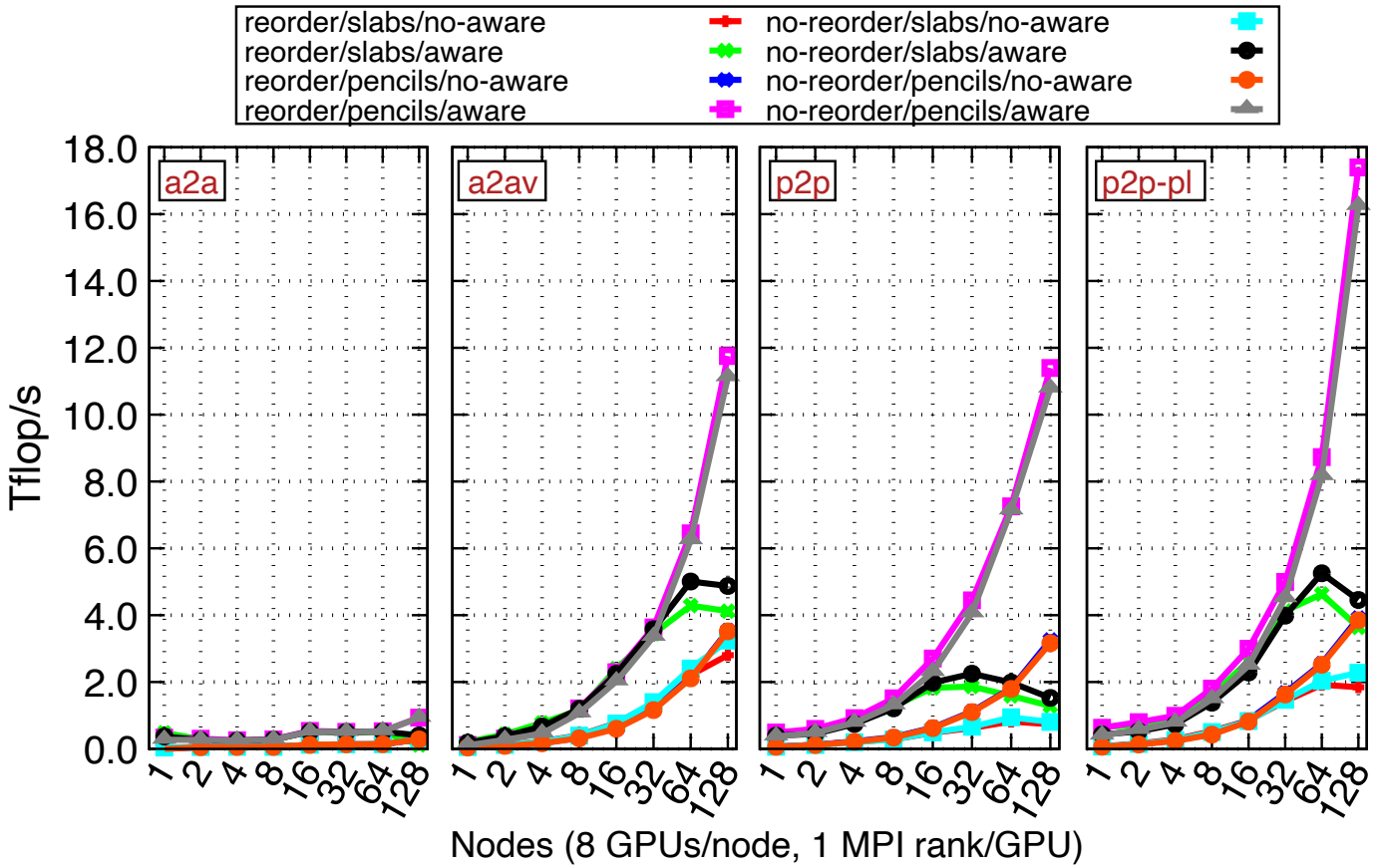


3D FFT, N = 1024,
FP64,
ROCm-5.6.0,
cray-mpich/8.1.27
8 nodes
64 ranks

***with explicit synchronization after kernel launches*

heFFTe Scalability on high-end GPU systems

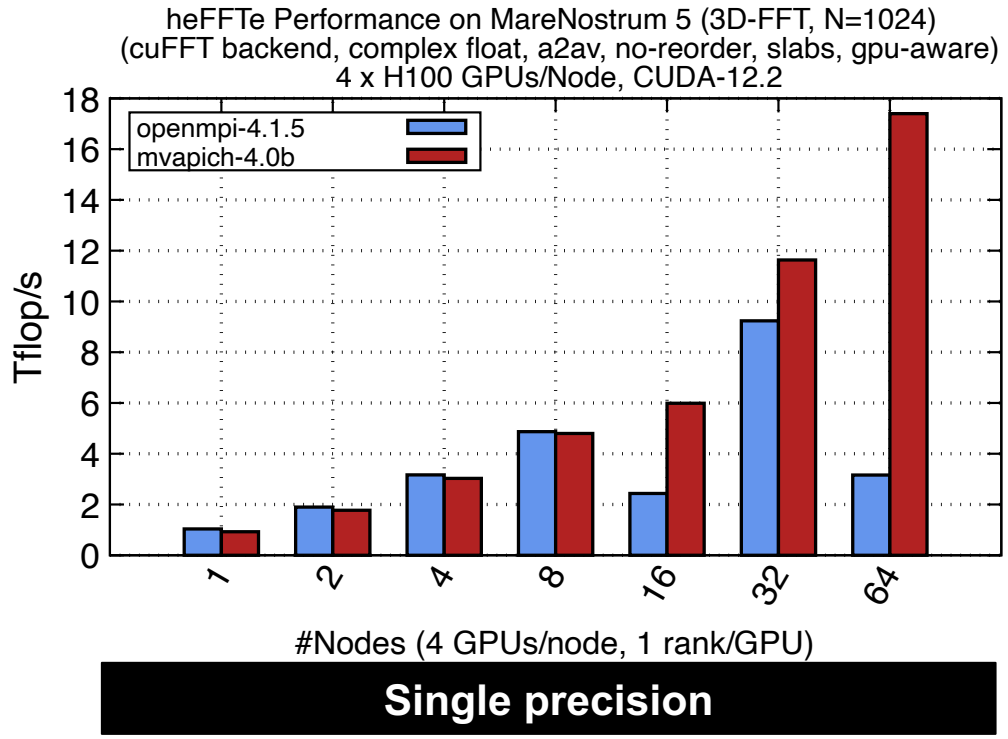
- 3D FFT, N = 1024
- rocFFT backend (ROCm-5.3.0)
- Cray-mpich-8.1.23
- 32 different runs
  - 4 communication patterns (a2a, a2av, p2p, p2p_pl)
  - 2 decompositions (pencils vs. slabs)
  - 2 FFT-1D modes (contiguous vs. strided)
  - 2 modes for MPI (std vs. GPU-aware)
- Could test/expose several aspects of an MPI implementation



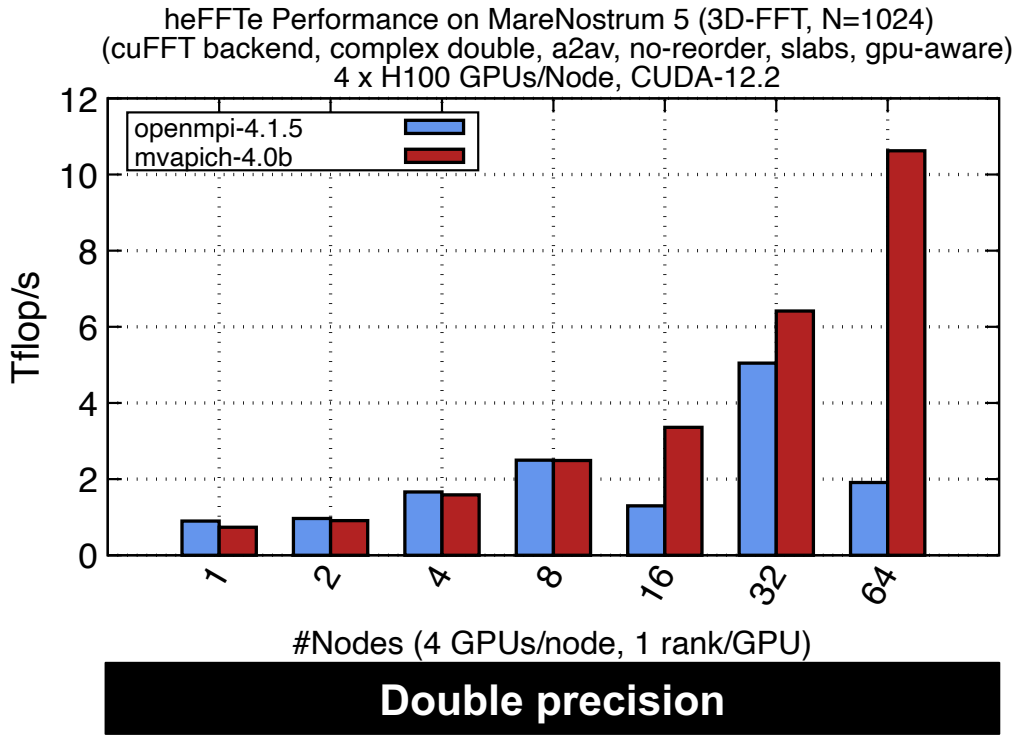
- *p2p: uses MPI_Send and MPI_Irecv, receive is pipelined with packing and sending*
- *p2p_pl: uses MPI_Isend and MPI_Irecv, all sending receiving packing and unpacking are pipelined*

heFFTe using MVAPICH Collectives on MareNostrum 5

Credits: MVP Team, especially Chen-Chun Chen



Single precision

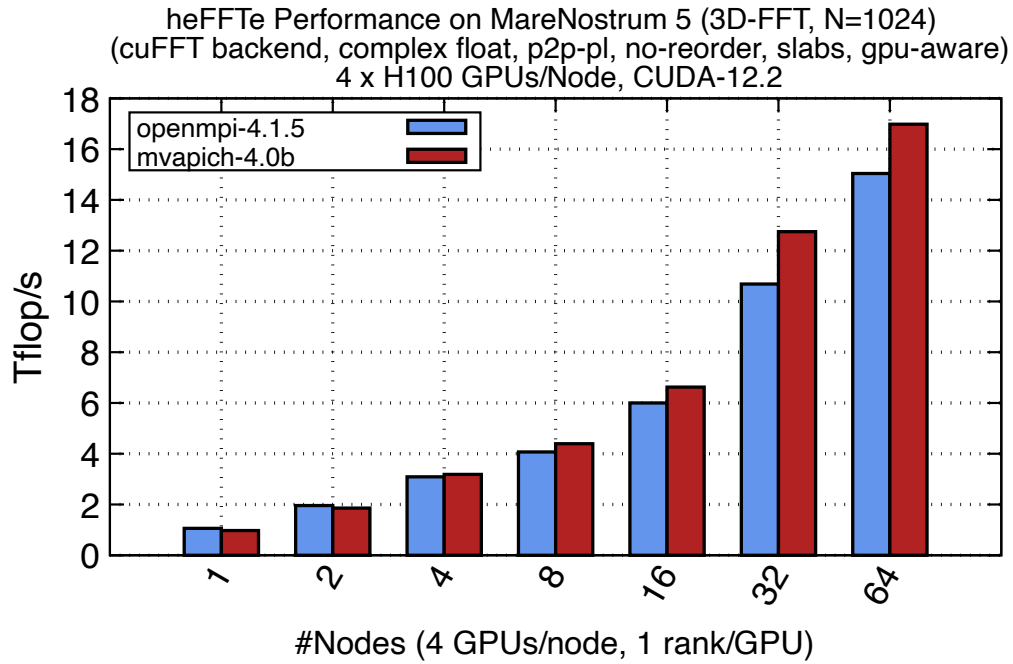


Double precision

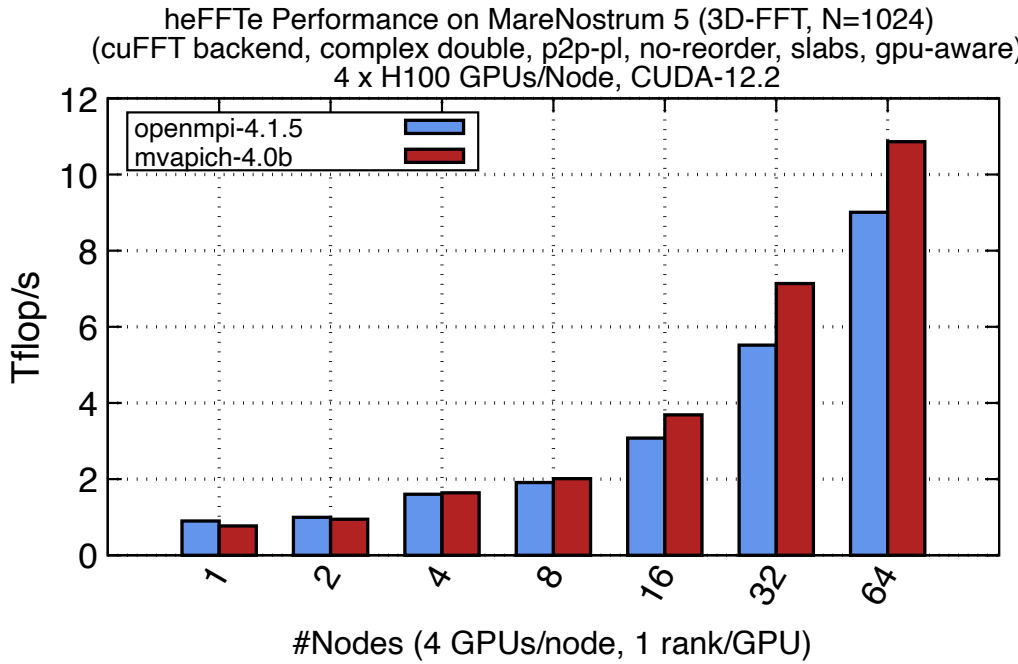
- A2AV is the default communication pattern in heFFTe
- Best performance is observed for (slabs decomposition, no-reorder)
- Impressive performance gains at scale, up to 5.6x against OpenMPI for FP32/FP64

heFFTe using MVAPICH P2P on MareNostrum 5

Credits: MVP Team, especially Chen-Chun Chen



Single precision



Double precision

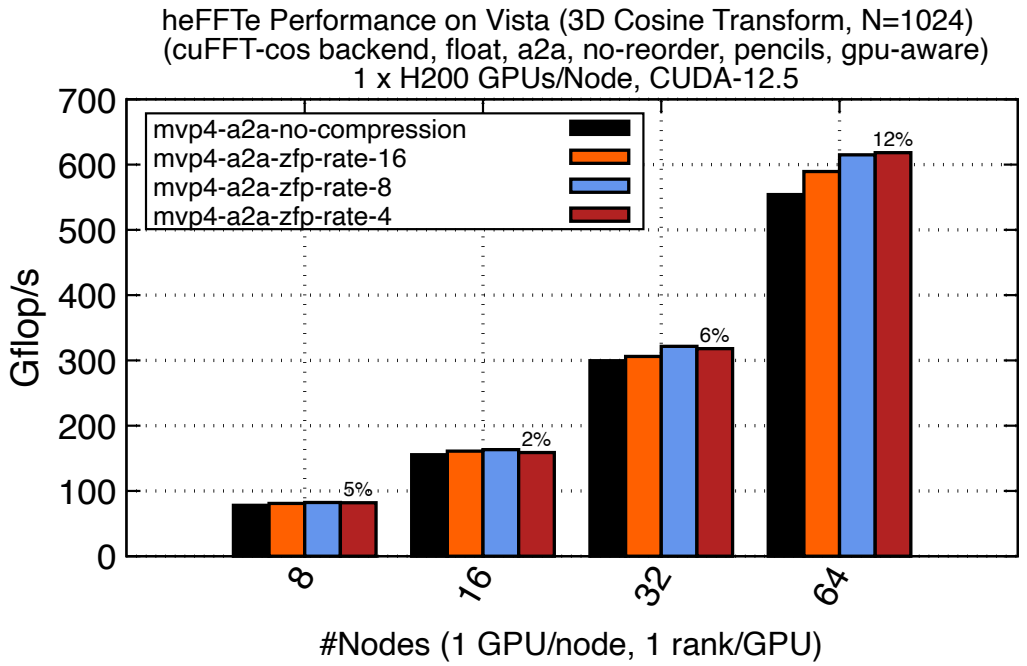
- Best performance is observed for (slabs decomposition, no-reorder)
- Impressive performance gains against OpenMPI at scale, up to 19% (FP32) and 29% (FP64)

heFFTe using MVAPICH Data Compression

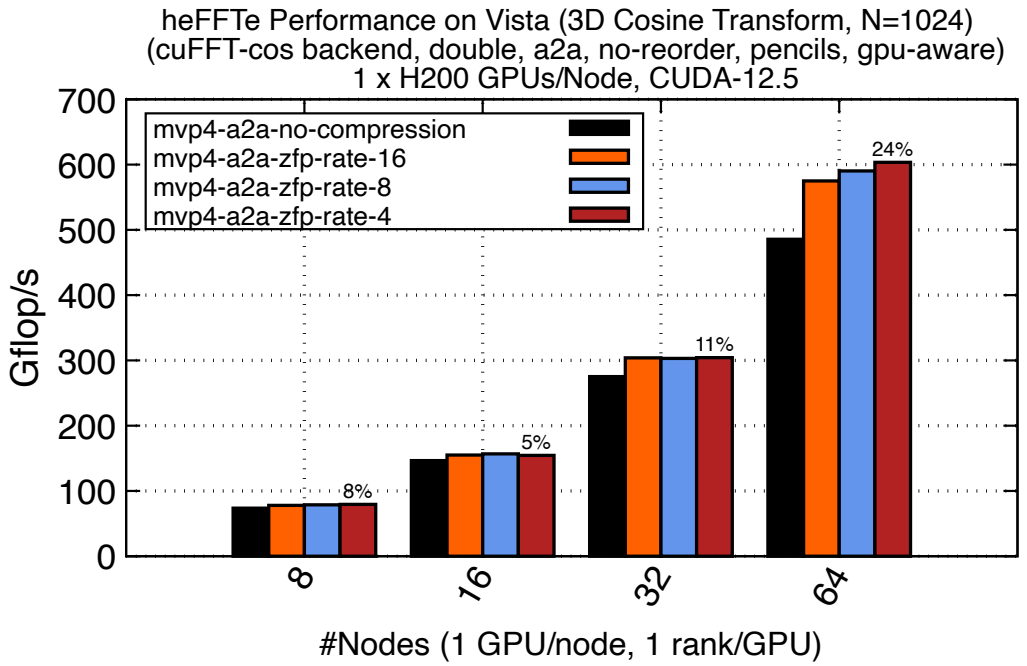
- MVP4 provides on-the-fly data compression
- Interfaces with libzfp underneath
  - Used for compressing floating point arrays
  - Mostly lossy compression, relies on “smoothness” of the data
  - Supports lossless compression (not parallelized, not available for GPUs)
- Challenges for heFFTe
  - No direct support for complex numbers
  - Out-of-the-box use shows very large errors
  - However, real sine/cosine transforms can benefit from MVP4 data compression using ZFP

heFFTe using MVAPICH Data Compression

Credits: MVP Team, especially Nick Contini and Chen-Chun Chen



Single precision



Double precision

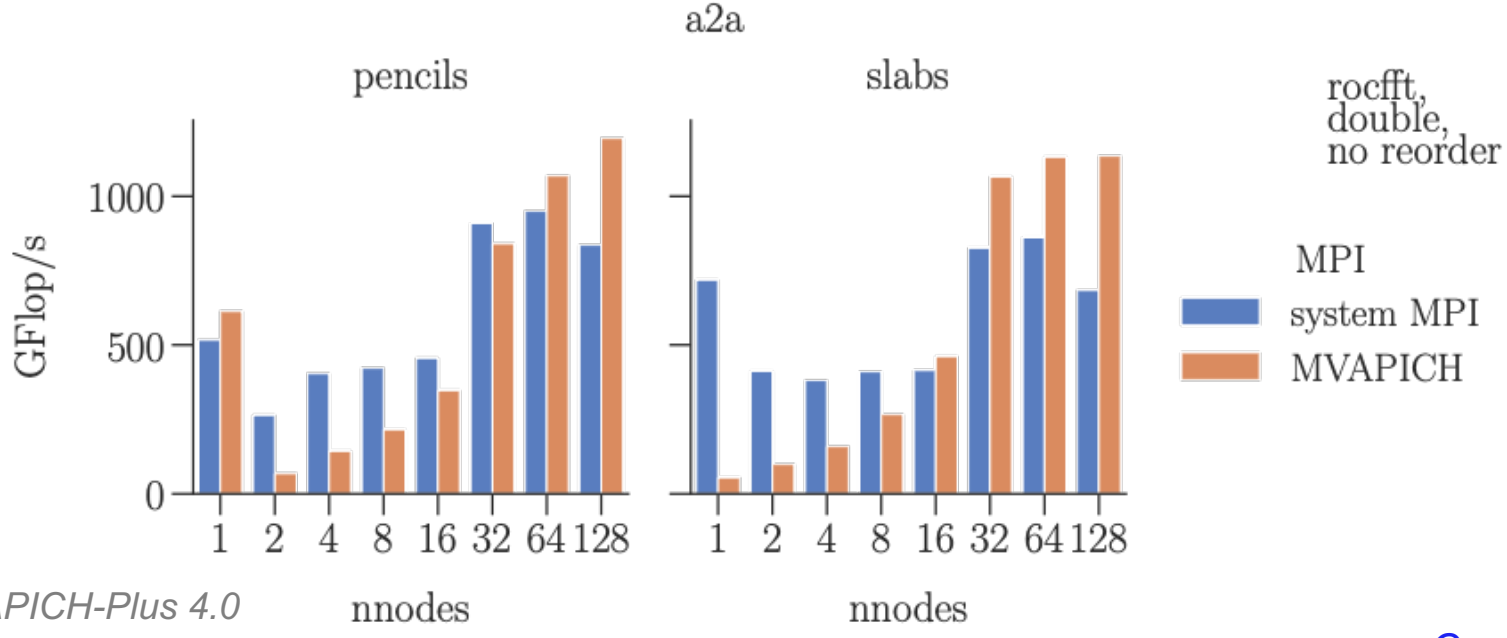
- Results for the Cosine Transform on Vista (GH200 GPU)
- Up to 12% and 24% on single/double precisions
- However, varying ZFP rate does not affect the performance
- Also the performance gains are expected to be higher, need to profile network traffic

Performance on Tuolumne (Tuo)

- Tuo is the little sister of El Capitan (an MI300A machine)
- Just cleared by LLNL to share these results ☺
- Highlights:
  - Performance portability: 84% improvement on 512 GPUs
    - up to **~16 Tflop/s on Tuo** vs **~8.7 Tflop/s on Frontier**
  - heFFTe + MVAPICH (A2A) outperforms system MPI at scale

Performance on Tuolumne (Tuo)

- Tuo is the little sister of El Capitan (an MI300A machine)
- Just cleared by LLNL to share these results ☺
- Highlights:
  - Performance portability: 84% improvement on 512 GPUs
    - up to **~16 Tflop/s on Tuo** vs **~8.7 Tflop/s on Frontier**
  - heFFTe + MVAPICH (A2A) outperforms system MPI at scale

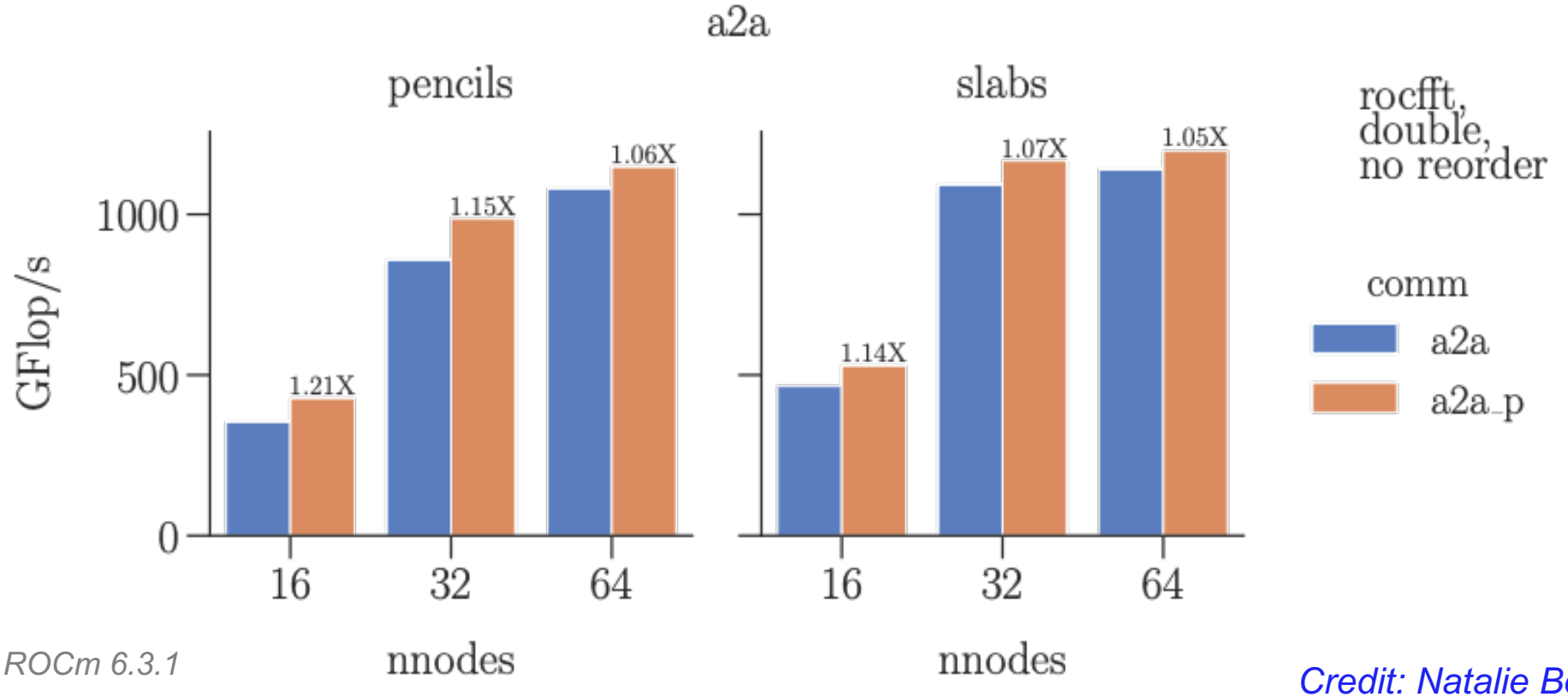


Cray MPICH 8.1.31 / MVAPICH-Plus 4.0
ROCm 6.3.1

Credit: Natalie Beams (UTK)

Performance on Tuolumne (Tuo)

- Persistent collectives show some promise
  - Up to 21% gain in performance
  - Still need more investigation to understand the performance behavior



MVAPICH-Plus 4.0 / ROCm 6.3.1

Credit: Natalie Beams (UTK)

Conclusion

- heFFTe is a robust library for multi-dimensional FFT computations
  - Mainly targeting DOE's Exascale system
  - GPU-enabled for NVIDIA, AMD, and Intel GPUs
  - Highly configurable
- heFFTe can serve as a good benchmark for MPI implementations
  - ~90% of execution time is spent in MPI calls
  - Uses different communication patterns (a2a, a2av, p2p, and pipelined p2p)
- heFFTe benefits from innovations in MVAPICH and TAU
  - Better performance for collectives
  - Real-time data compression (real transforms only at this point)
  - Need deeper integration with MVP and TAU to profile data compression

Future Directions

- heFFTe using MVP4 and TAU
  - Support data compression for complex transform
  - Use TAU to profile the size of compressed data under different compression rates
  - Real-time data compression on AMD systems (Frontier, Tuolumne, El Capitan)
- Improve performance through persistent collectives
  - Work by Natalie Beams
- Auto-tuning framework for heFFTe
  - Message size is key to performance (w/o compression)
  - How to choose communication pattern (collectives vs. p2p) for a given problem size

Thank You


ICL
INNOVATIVE
COMPUTING LABORATORY


THE UNIVERSITY OF
TENNESSEE
KNOXVILLE