Performance and Scalability of 3D FFT using heFFTe and MVAPICH

Natalie Beams and Ahmad Abdelfattah

University of Tennessee

14th Annual MUG

Aug. 19, 2026

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Credit to the heFFTe team

- 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 → AMD)
- George Bosilca (UTK → NVIDIA)
- Veronica Montanaro (ETH)
- Sonali Mayani (ETH)
- Andreas Adelmann (ETH)
- students and outside collaborators

Fast Fourier Transform (FFT)

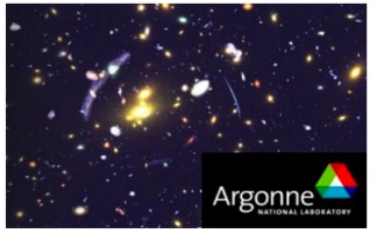
- FFT computes the Discrete Fourier Transform (DFT) of a series:
  Let $x = x_0, \ldots, x_{N-1}$ are complex numbers. The DFT of x is the sequence
  $X = X_0, X_1, \ldots, X_{n-1}$,

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

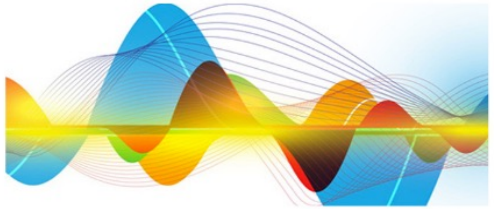
- DFT can be computed as a matrix-vector multiplication (GEMV) in $\mathcal{O}(N^2)$ FLOPs (memory-bound)
- FFT reduces the complexity to $\mathcal{O}(N \log_2 N)$ (even more memory-bound)
- The Inverse Discrete Fourier Transform (IDFT) is similarly defined except that the 'e' exponents are taken as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

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

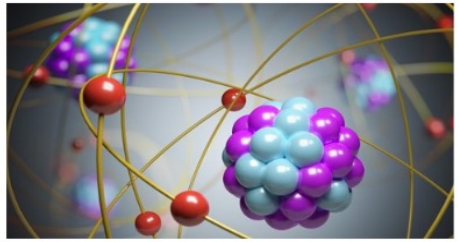
Cosmology
***ECP* ExaSky - HACC**



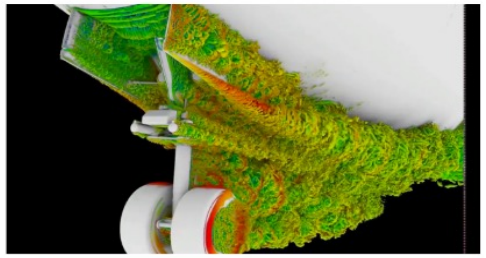
Signal processing,
***ECP* WARPX**



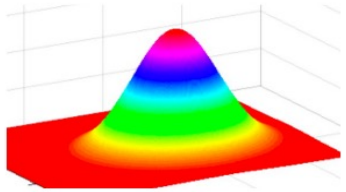
Deep Learning



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)

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

heFFTe among other FFT projects

- 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

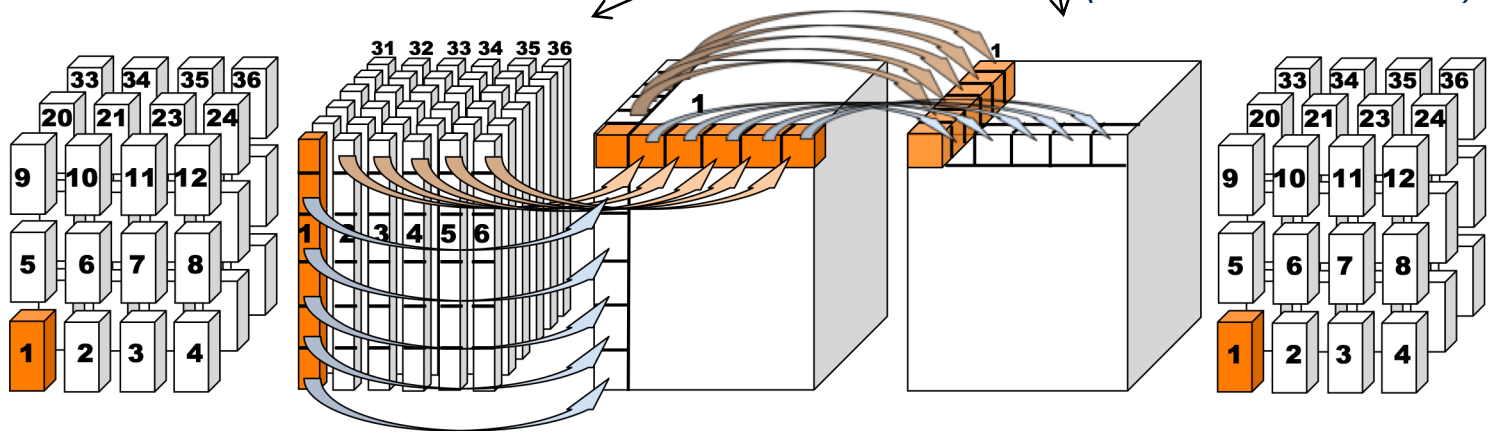
	Library	Pencil Decomp	Brick Decomp	Slab Decomp	Transpose Reshape	Stride Reshape	R2C Transform	Single precision	Mixed precision	Multiple backends	Nonblocking All-to-All
CPU	FFTW3	✓				✓	✓	✓			
CPU	FFTMPI	✓	✓		✓			✓		✓	
CPU	2DECOMP	✓				✓	✓				
CPU	SWFFT		✓		✓						
CPU	PFFT	✓			✓		✓				
CPU	P3DFFT	✓	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

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY OF TENNESSEE KNOXVILLE

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) – different backends
  - 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 (Can also do "slabs")



ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

heFFTe options & related MPI needs

- Both “pencils” and “slabs”
  - Pencils: more alltoall calls with fewer ranks in each alltoall
  - Slabs: fewer calls with more ranks (num ranks: $P_{pencils} = \sqrt{P_{slabs}}$)
- The all-to-all communication can be done with collectives or point-to-point communication:

heFFTe comm option	MPI routines
a2a	Alltoall
a2av	Alltoallv
p2p (point to point)	Send, IRecv
p2p_pl (point to point, pipelined)	ISend, IRecv

heFFTe options & related MPI needs

- Both “pencils” and “slabs”
  - Pencils: more alltoall calls with fewer ranks in each alltoall
  - Slabs: fewer calls with more ranks (num ranks: $P_{pencils} = \sqrt{P_{slabs}}$)
- The all-to-all communication can be done with collectives or point-to-point communication:

	heFFTe comm option	MPI routines
	a2a	Alltoall
★	a2av	Alltoallv
	p2p (point to point)	Send, IRecv
★	p2p_pl (point to point, pipelined)	ISend, IRecv

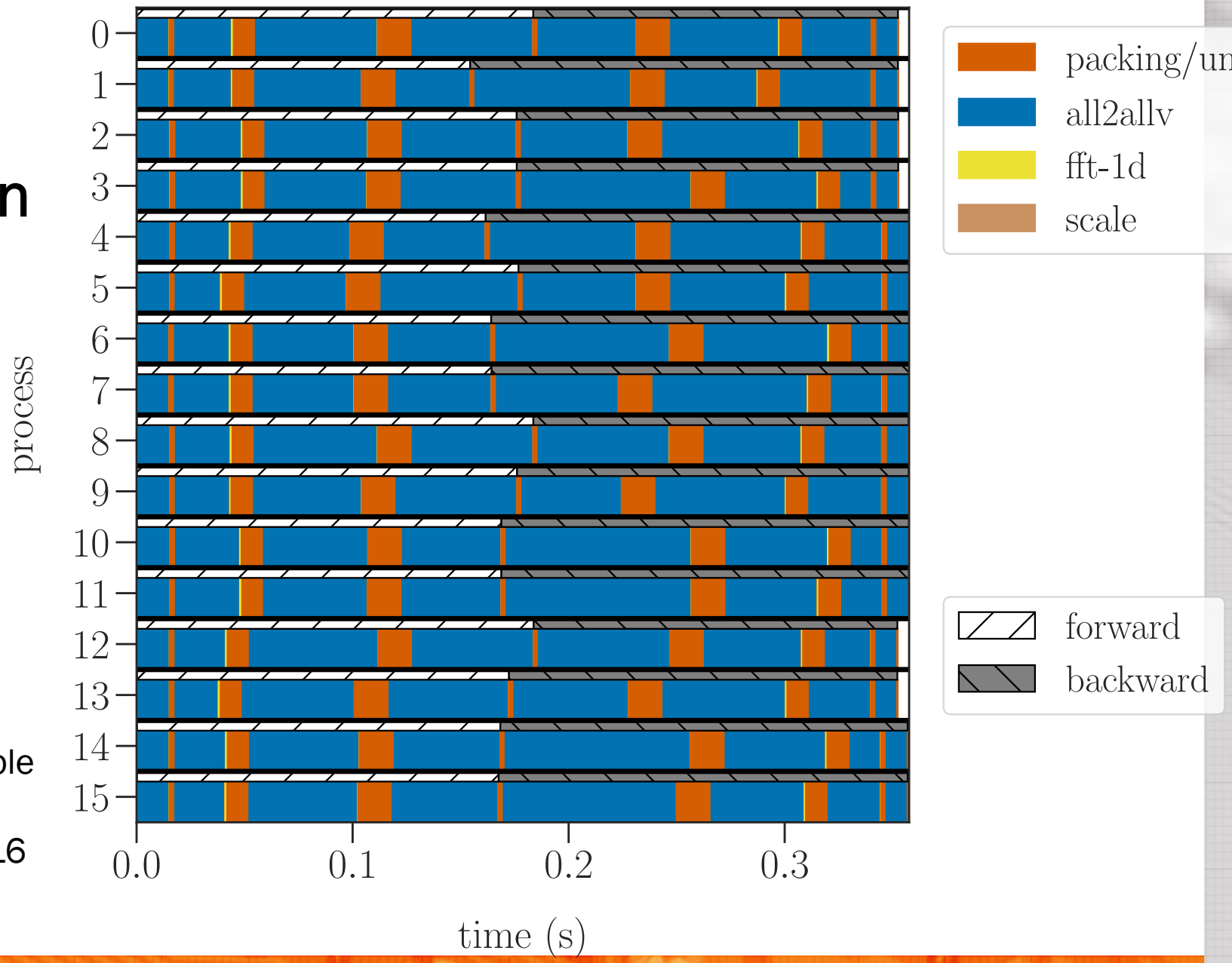
heFFTe options & related MPI needs

- Both “pencils” and “slabs”
  - Pencils: more alltoall calls with fewer ranks in each alltoall
  - Slabs: fewer calls with more ranks (num ranks: $P_{pencils} = \sqrt{P_{slabs}}$)
- The all-to-all communication can be done with collectives or point-to-point communication:

	heFFTe comm option	MPI routines
	a2a	Alltoall
★	a2av	Alltoallv
	p2p (point to point)	Send, IRecv
★	p2p_pl (point to point, pipelined)	ISend, IRecv

(There is also a “reordering” option to avoid using strided 1D FFTs)

heFFTe is heavily communication bound...

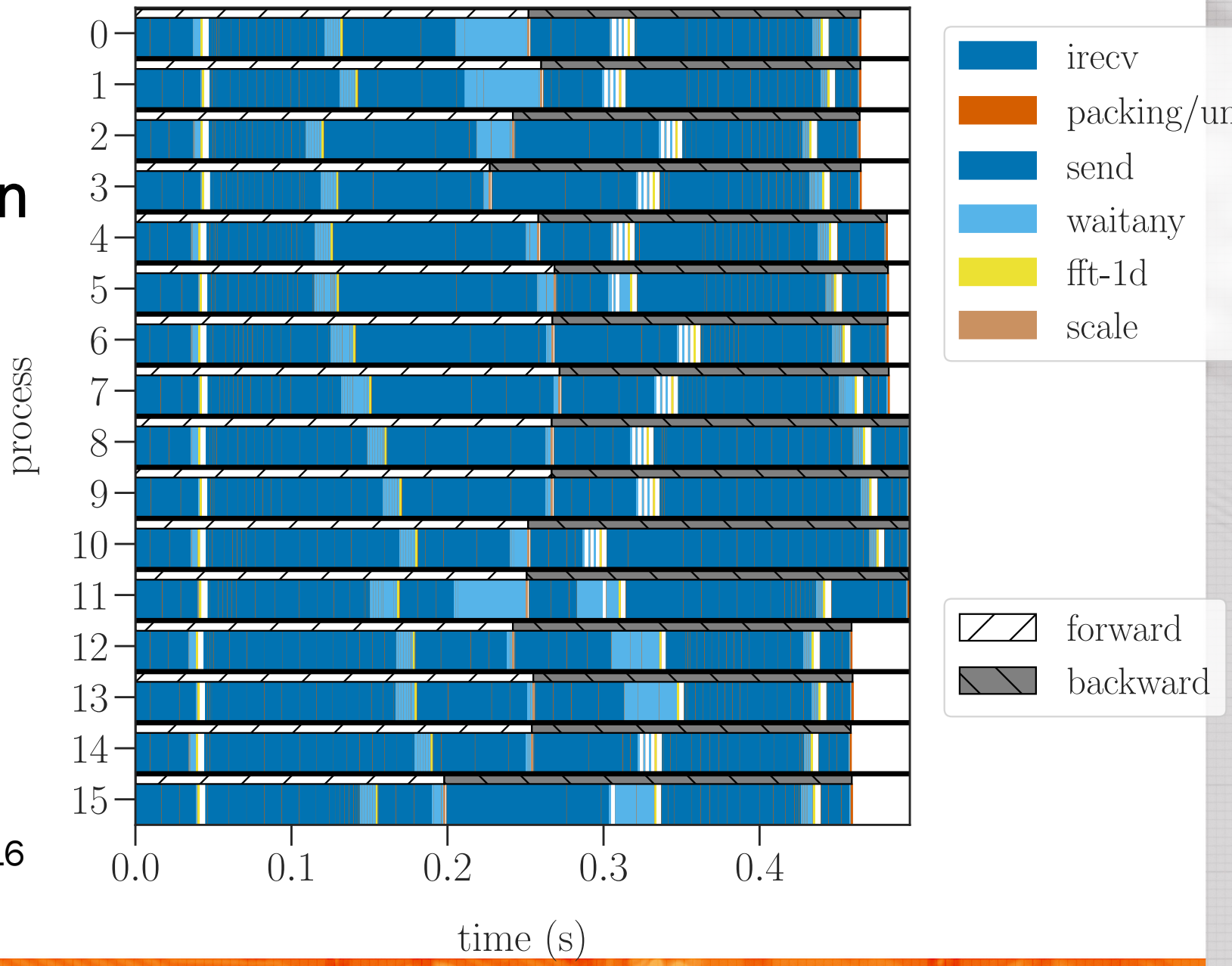


No reorder, pencils, double precision FFT
4 nodes of Tuolumne – 16 ranks, 1 rank per MI300

heFFTe is heavily communication bound...

in all configurations

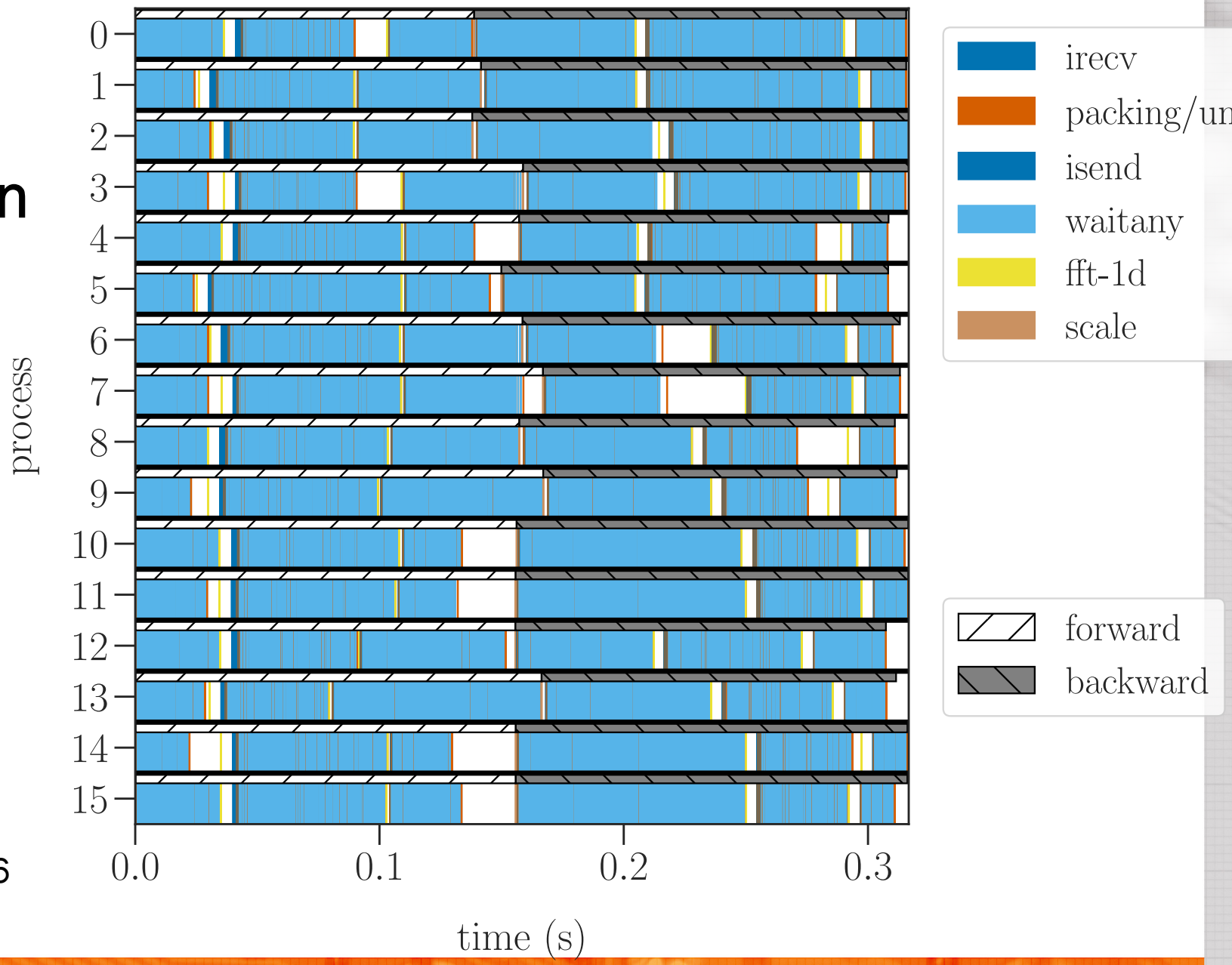
Reorder, slabs, double precision FFT
4 nodes of Tuolumne – 16 ranks, 1 rank per MI300



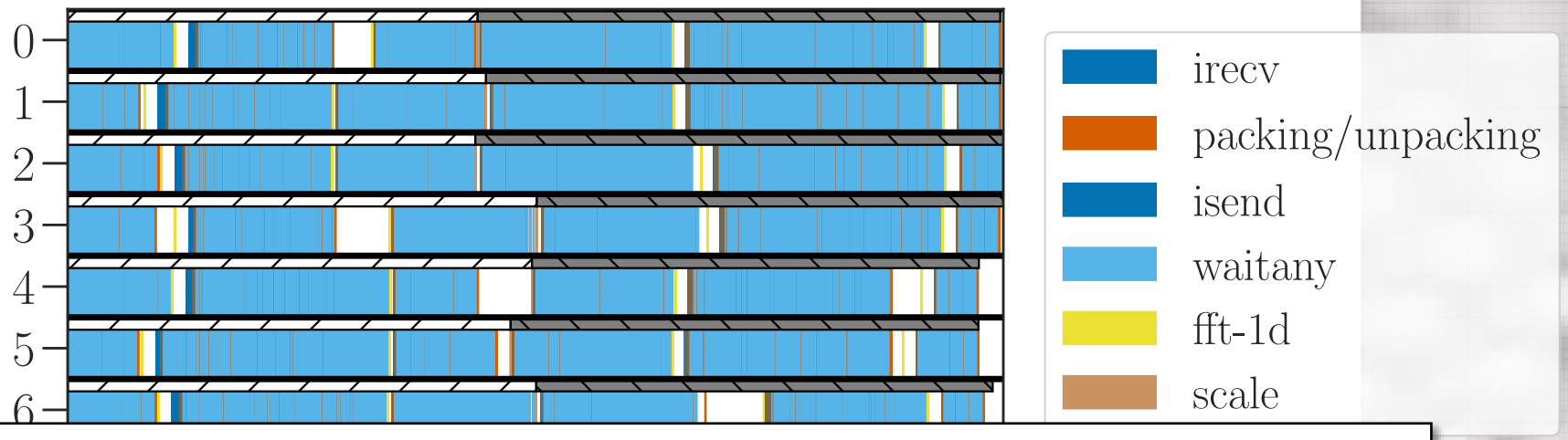
heFFTe is heavily communication bound...

in all configurations

Reorder, slabs, double precision FFT – p2p pipelined
4 nodes of Tuolumne – 16 ranks, 1 rank per MI300



heFFTe is heavily communication bound...



in all configura

Reorder, slabs,
precision FFT –
pipelined
4 nodes of Tuol
ranks, 1 rank per MI300

Good benchmarking vehicle for MPI implementations:
Improvement in communication performance will have a big impact on heFFTe's performance

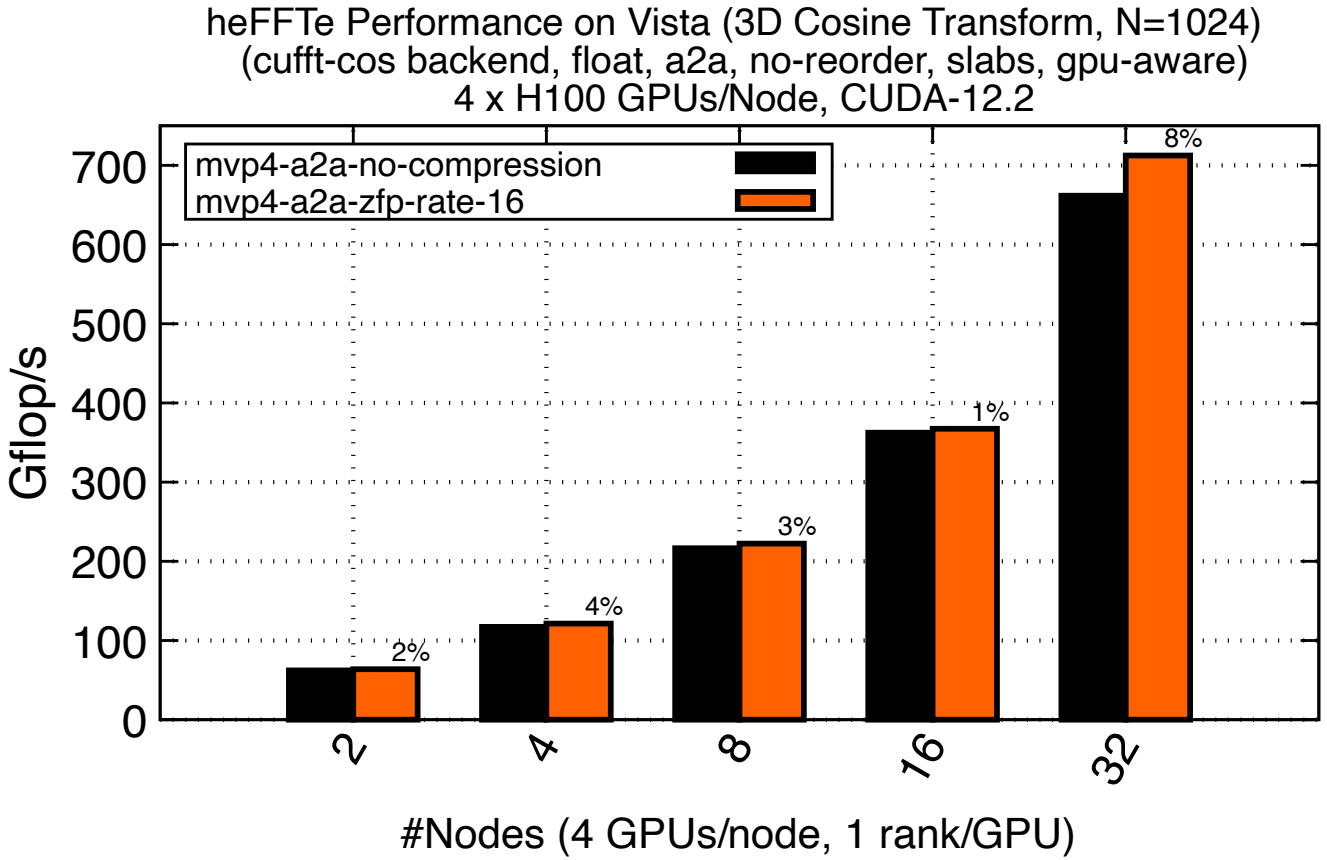
Current investigations for communication in heFFTe

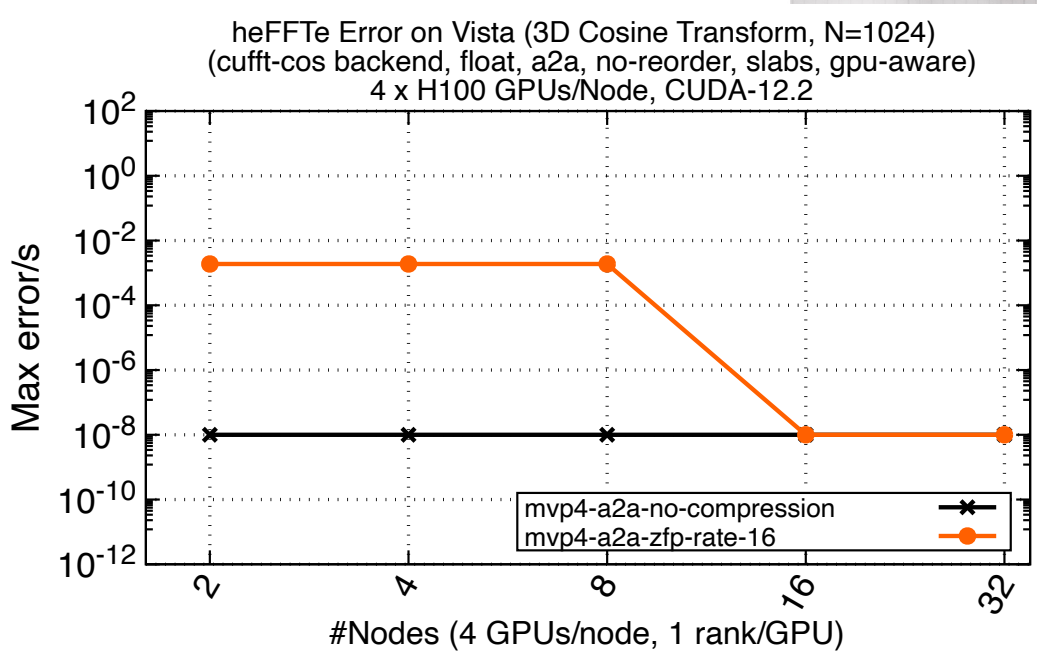
- **Lossy compression** of messages – can we gain speed while maintaining acceptable accuracy?
- **Persistent communication** – could there be a benefit when an application does many FFTs with the same communication patterns?
- Plus: some new results on Tuolumne (AMD MI300A)

Compression with ZFP in MVAPICH

- Benchmarks on BSC MareNostrum 5 (4 NVIDIA H100 per node)
- Basic heFFTe benchmarks use random input data; not good for ZFP compression
  - Consider constant input
- Use ***discrete cosine transform (DCT)*** instead of FFT since we also don't expect MVAPICH's ZFP compression to work well with interleaved complex data
- Test ranging from 2 to 64 nodes, with 4 ranks/node (1 per GPU)

Compression with ZFP in MVAPICH



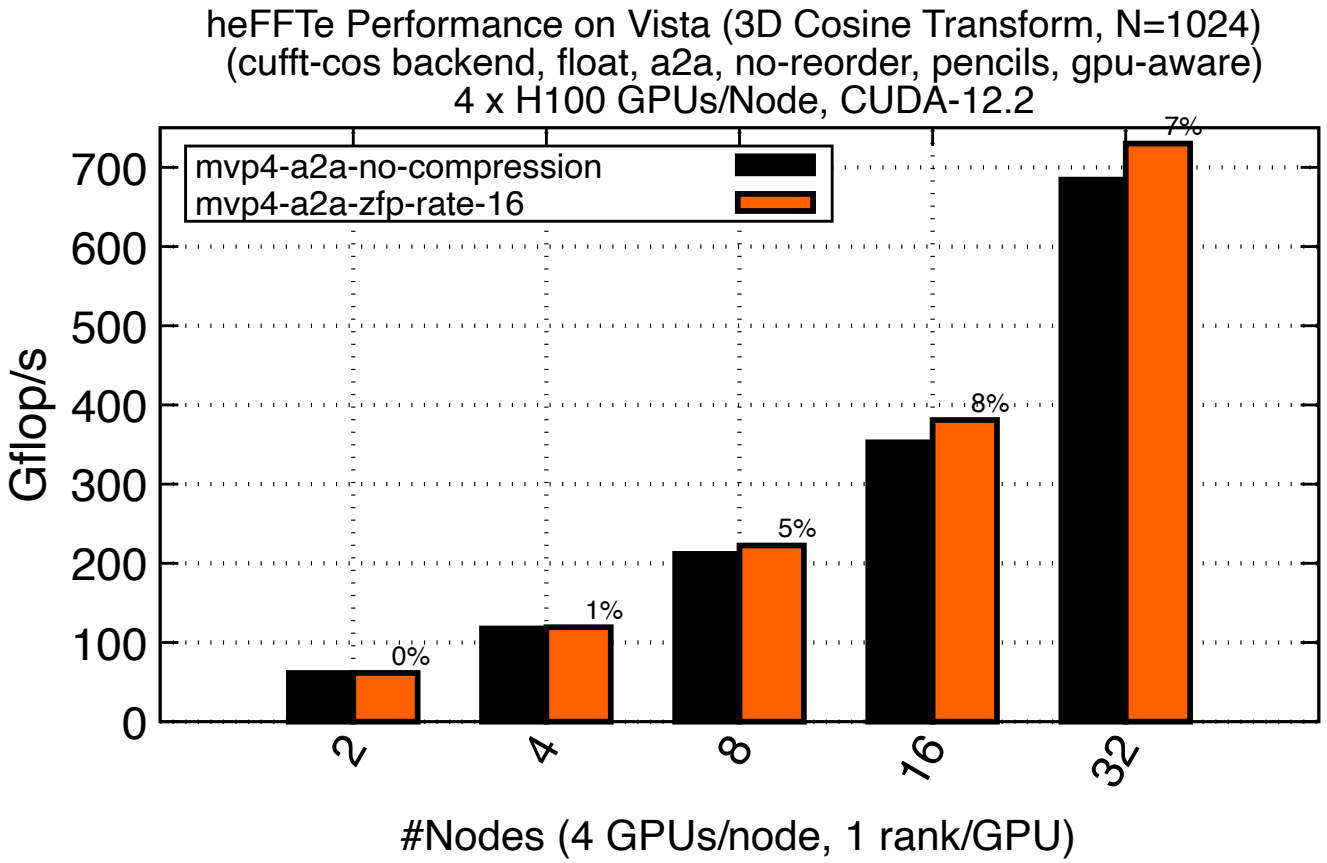


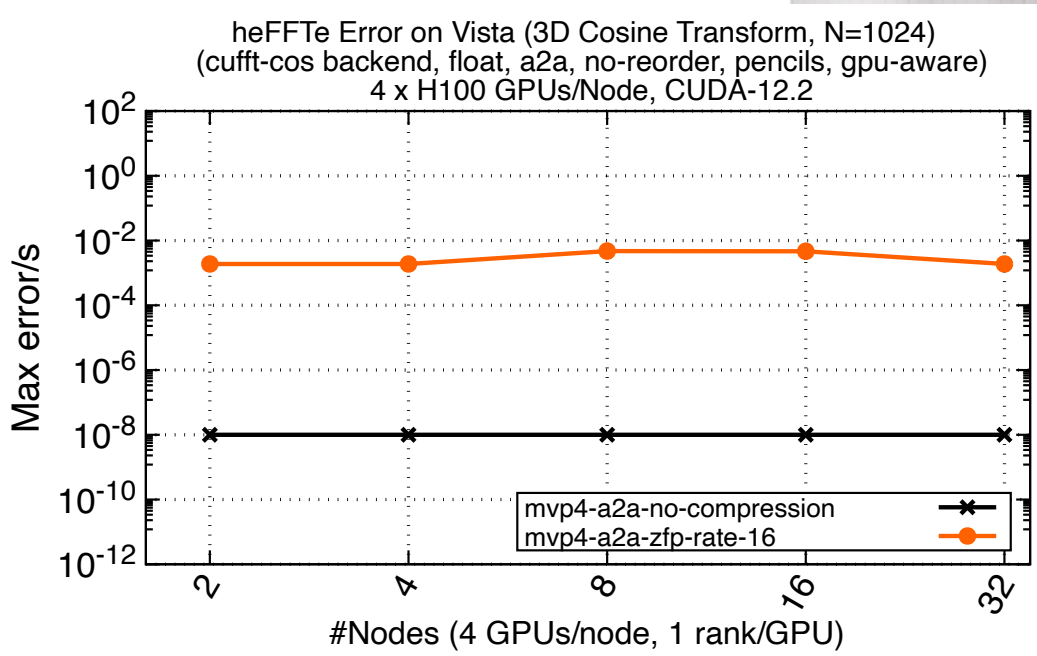
Up to 8% speedup for 32 nodes/128 ranks in all-to-all, several digits of accuracy lost for smaller number of nodes/bigger messages

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

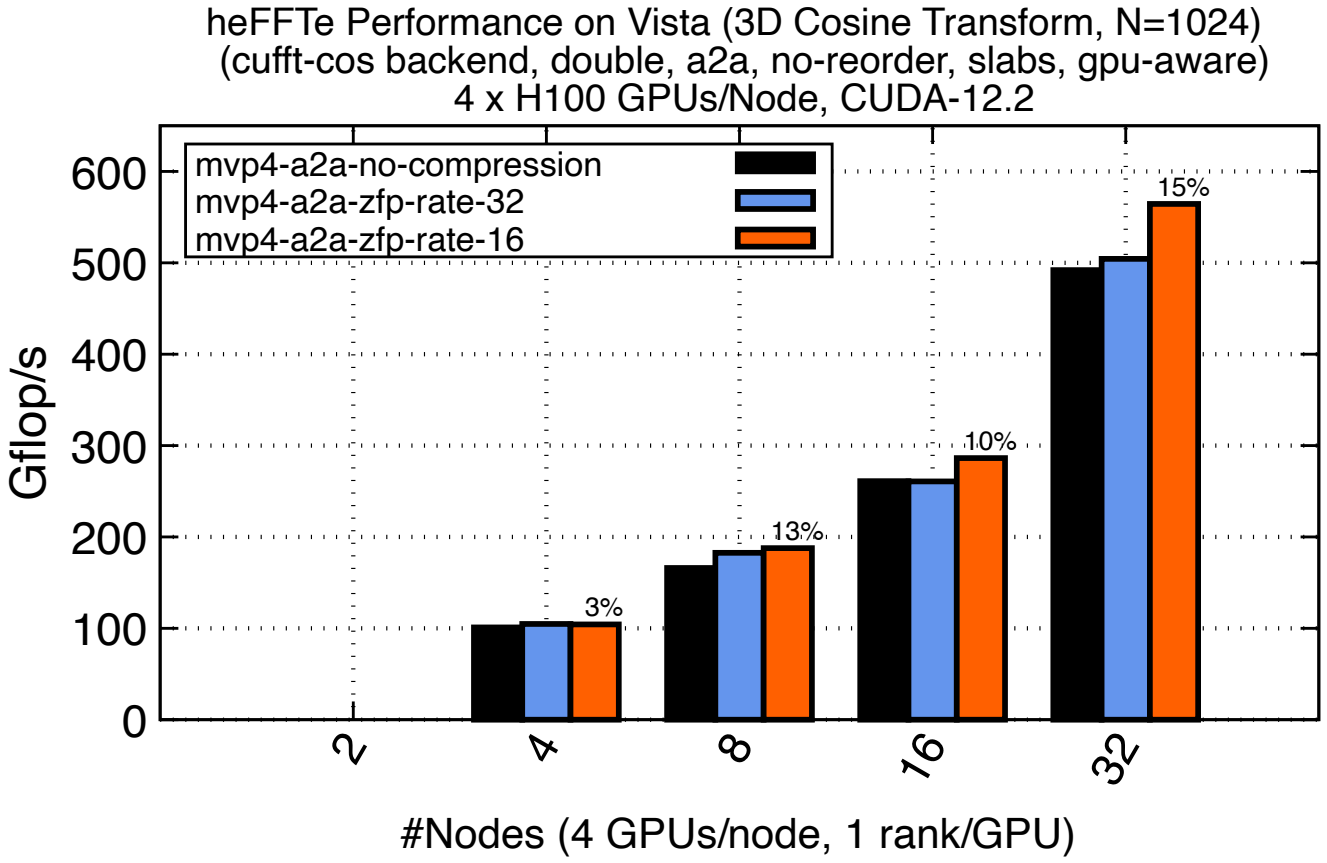
Compression with ZFP in MVAPICH

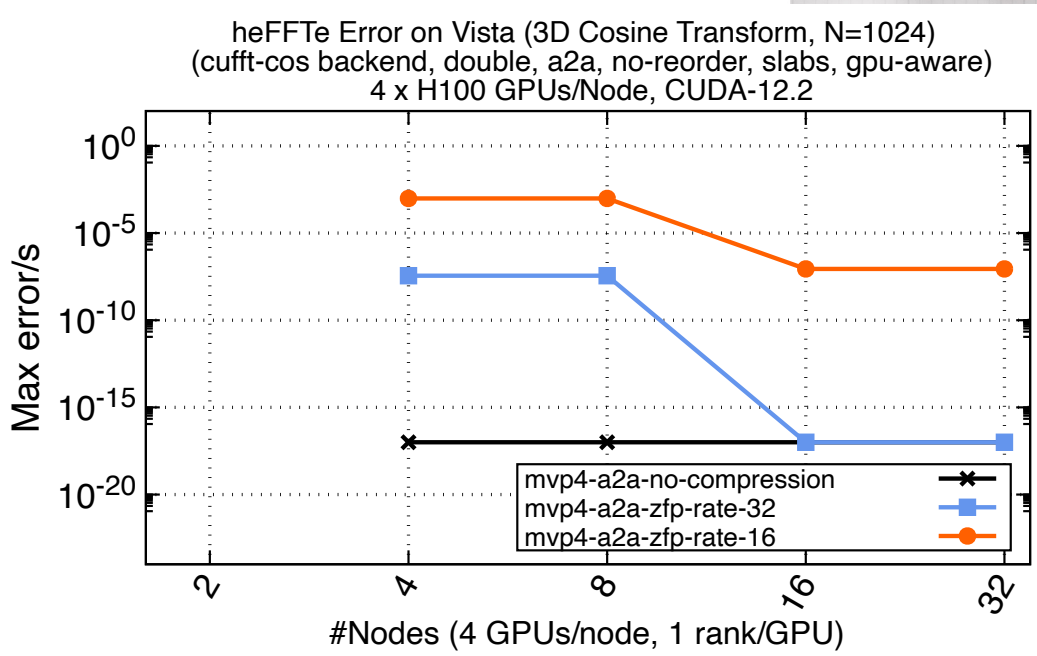




Similar performance gains for pencils decomposition

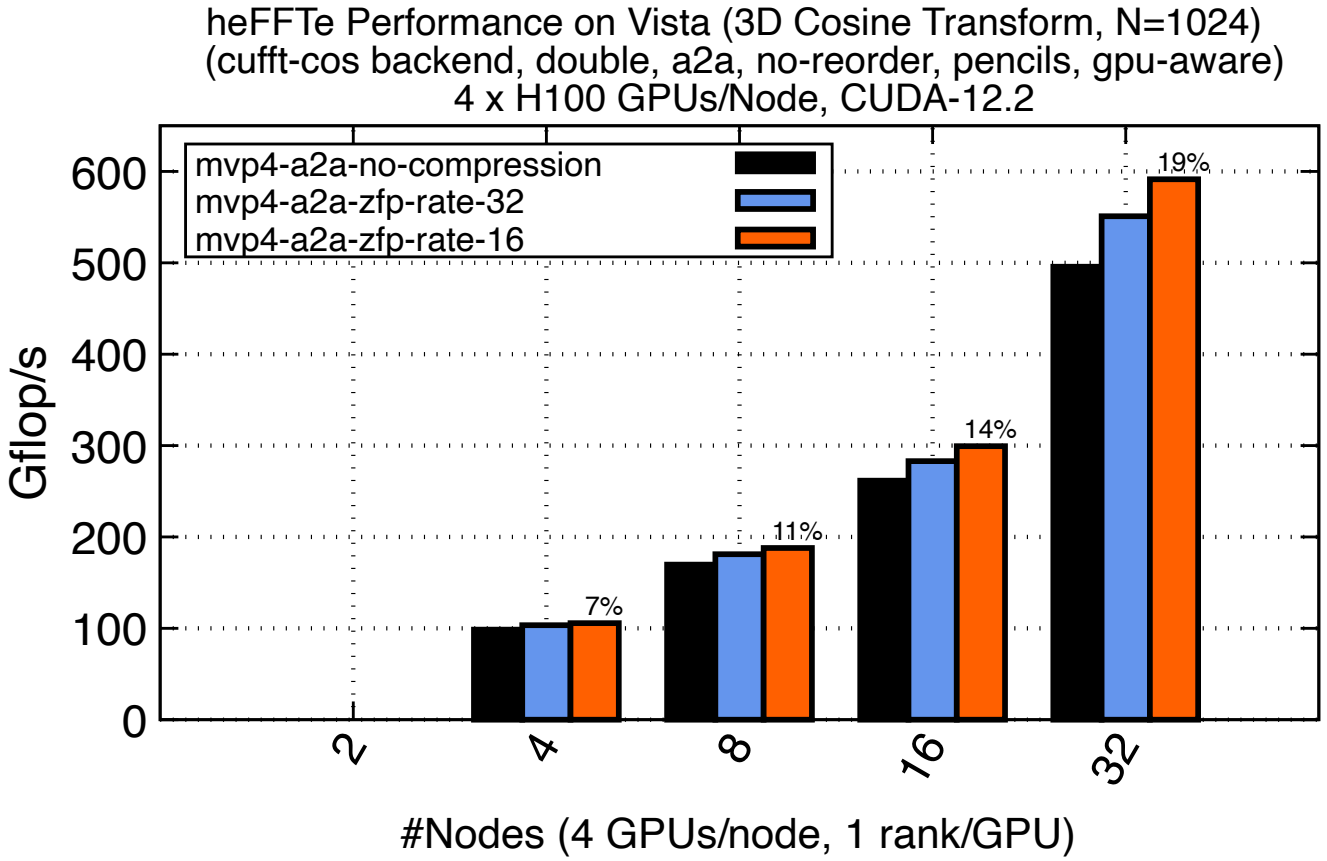
Compression with ZFP in MVAPICH

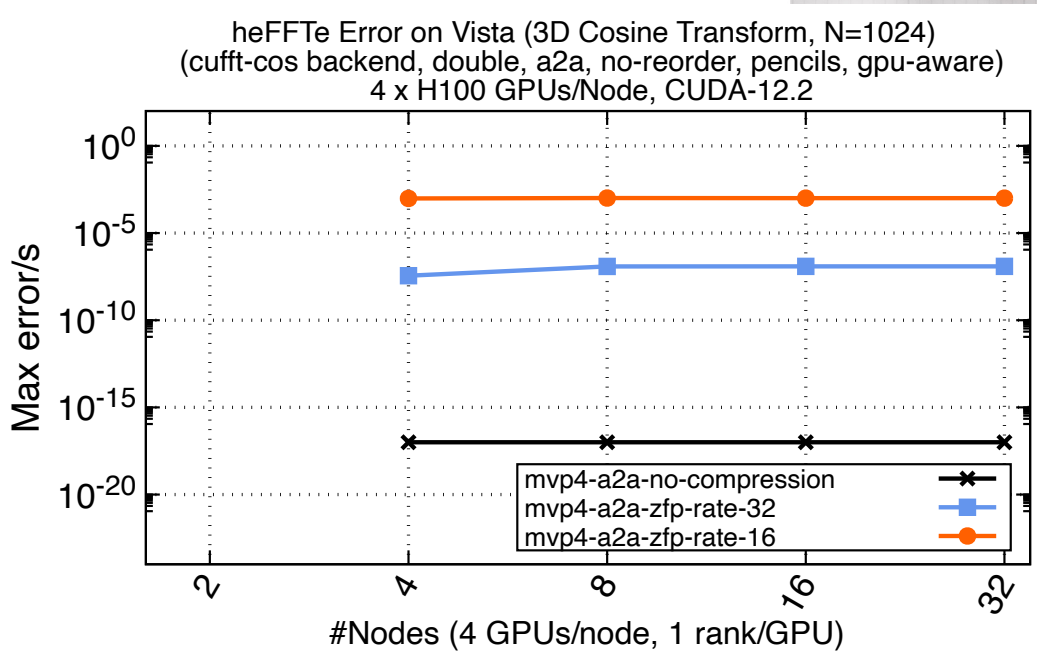




With double precision, more options for compression; up to 15% speedup

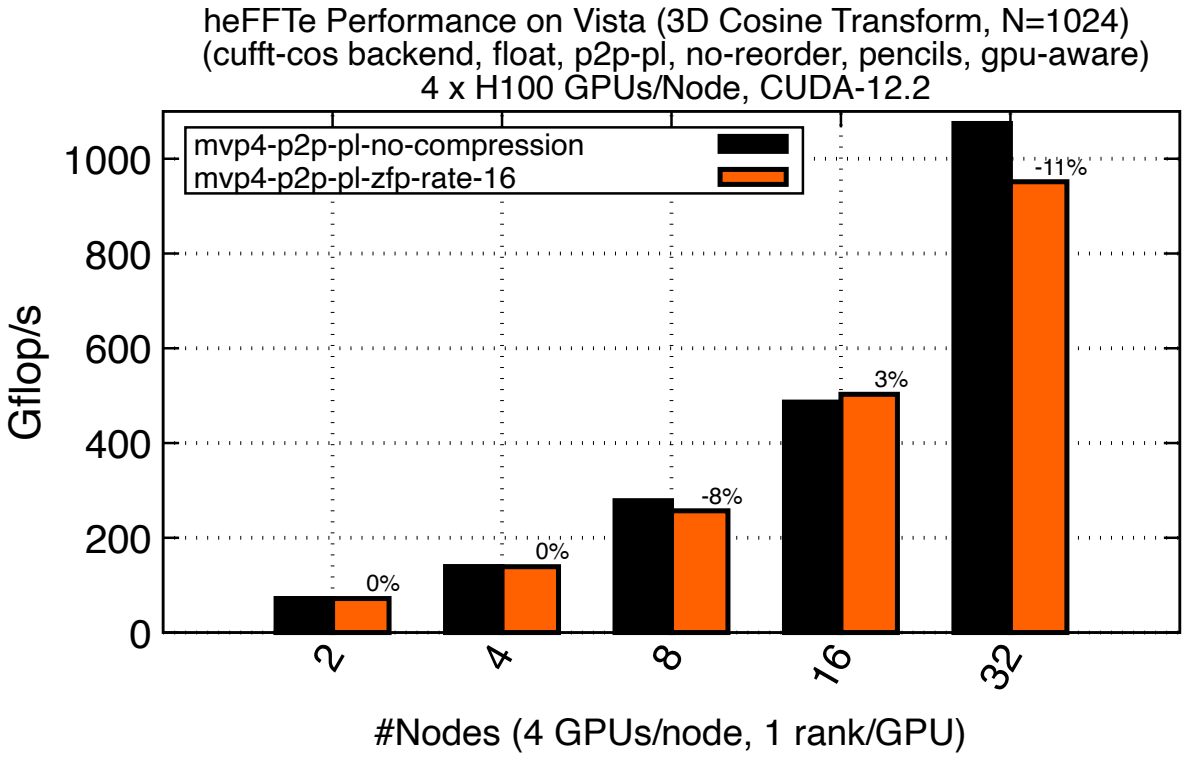
Compression with ZFP in MVAPICH

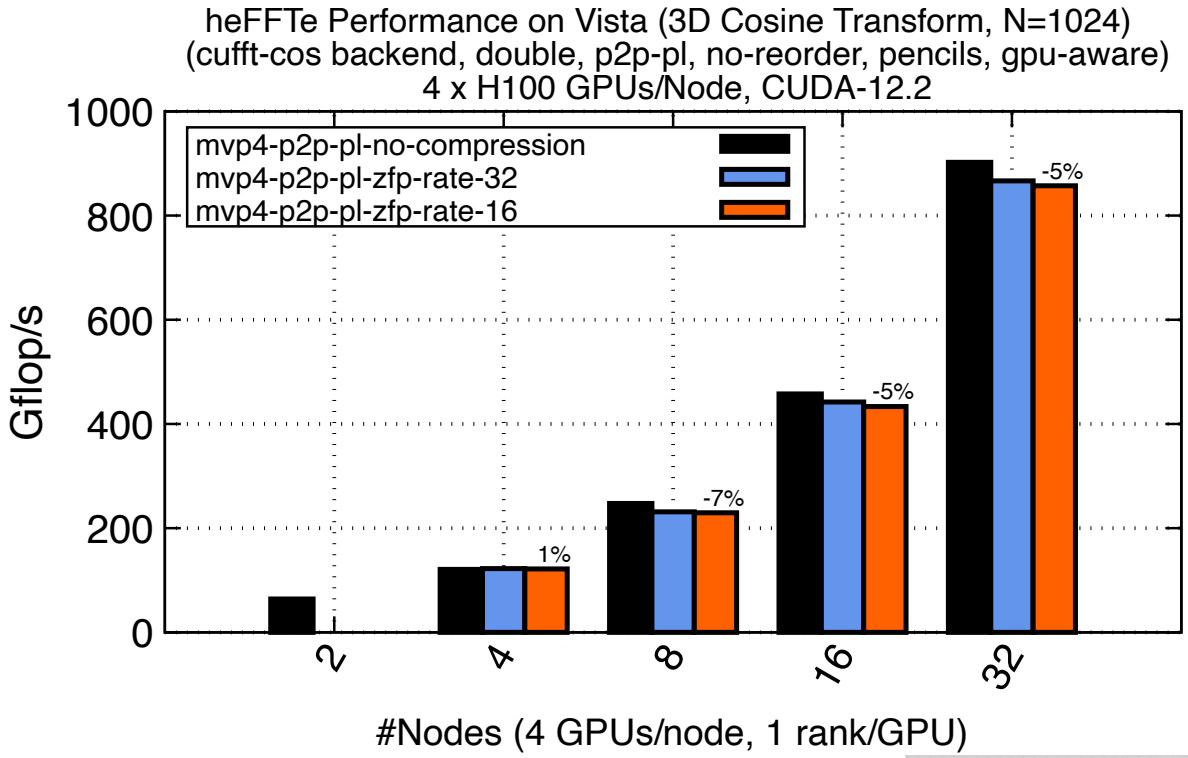




With pencils, more performance gains, but higher errors at larger numbers of nodes

Compression with ZFP in MVAPICH





For point-to-point communication, no advantage to compression

Compression summary

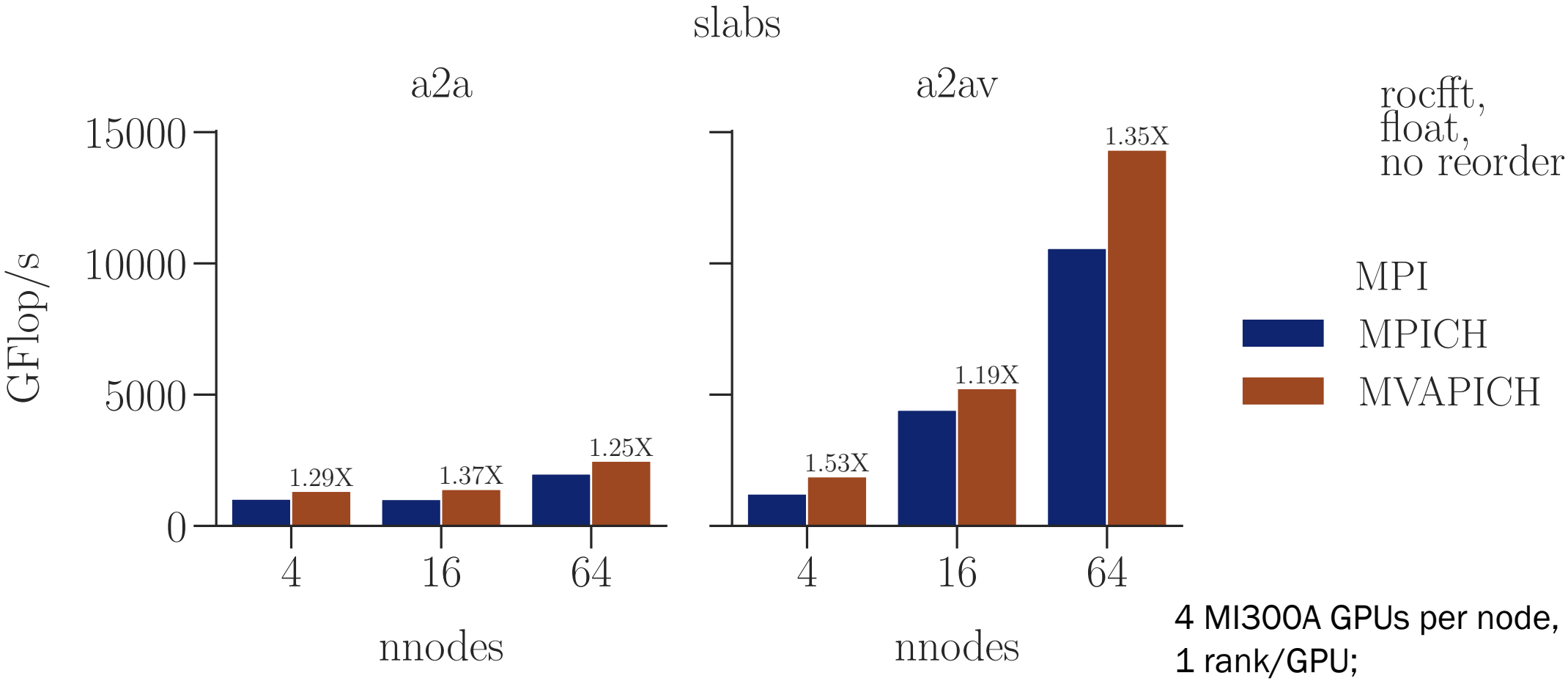
- Potential for speedup in applications which can tolerate some loss of accuracy
  - Can achieve significant speedup if using all-to-all communication option, not with point-to-point

- Limiting factor: inability to handle complex data stored in interleaved format
  - Cannot necessarily expect smoothness *between* real/imaginary components, even if we can expect smoothing *within* each component
  - Biggest users of heFFTe need Fourier transform, not cosine/sine transform

Tuolumne: MVAPICH 4.1 vs MPICH 9.0.1

- 4 MI300A GPUs per node
  - 1 rank/GPU
- ROCm 6.4.2
- GPU-aware MPI

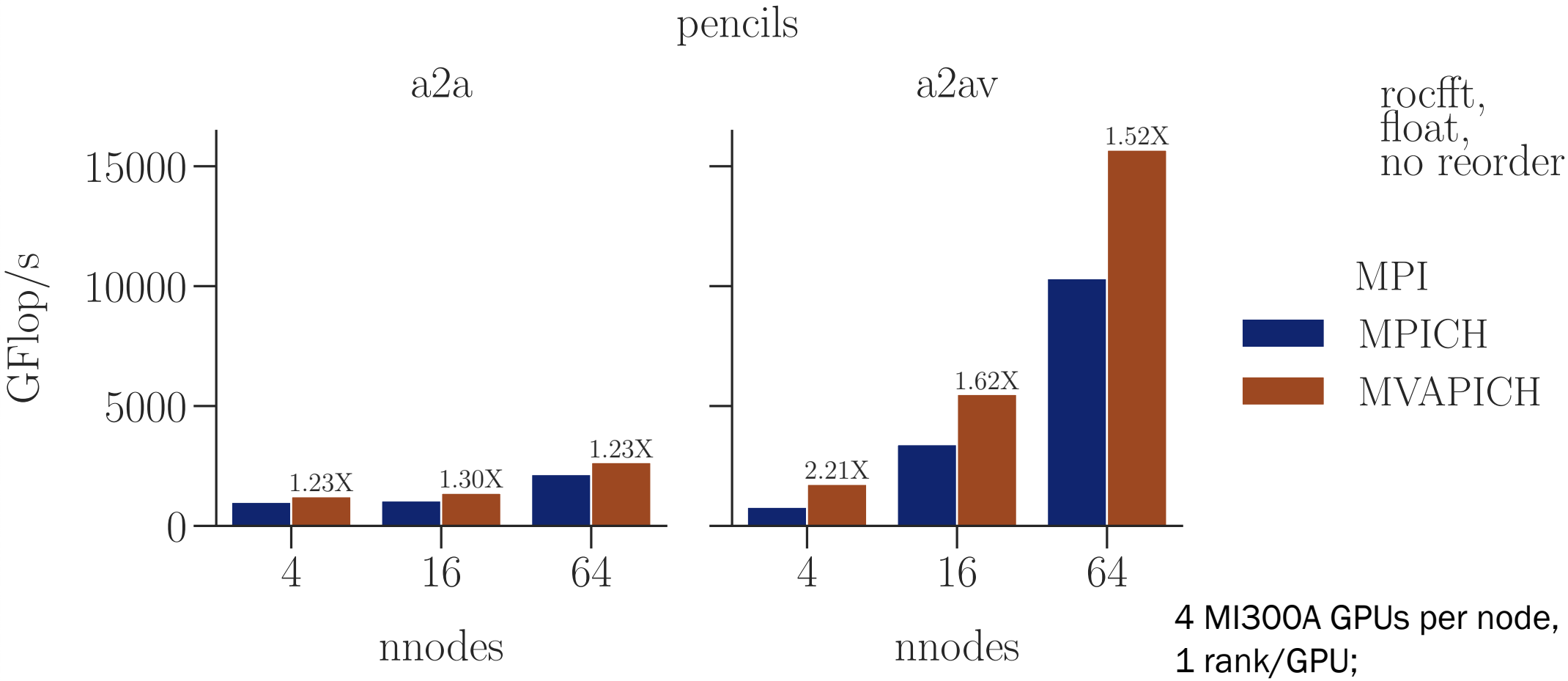
Tuolumne: MVAPICH 4.1 vs MPICH 9.0.1



4 MI300A GPUs per node,
1 rank/GPU;
with ROCm 6.4.2

MVAPICH with good speedup over MPICH for alltoall operations!

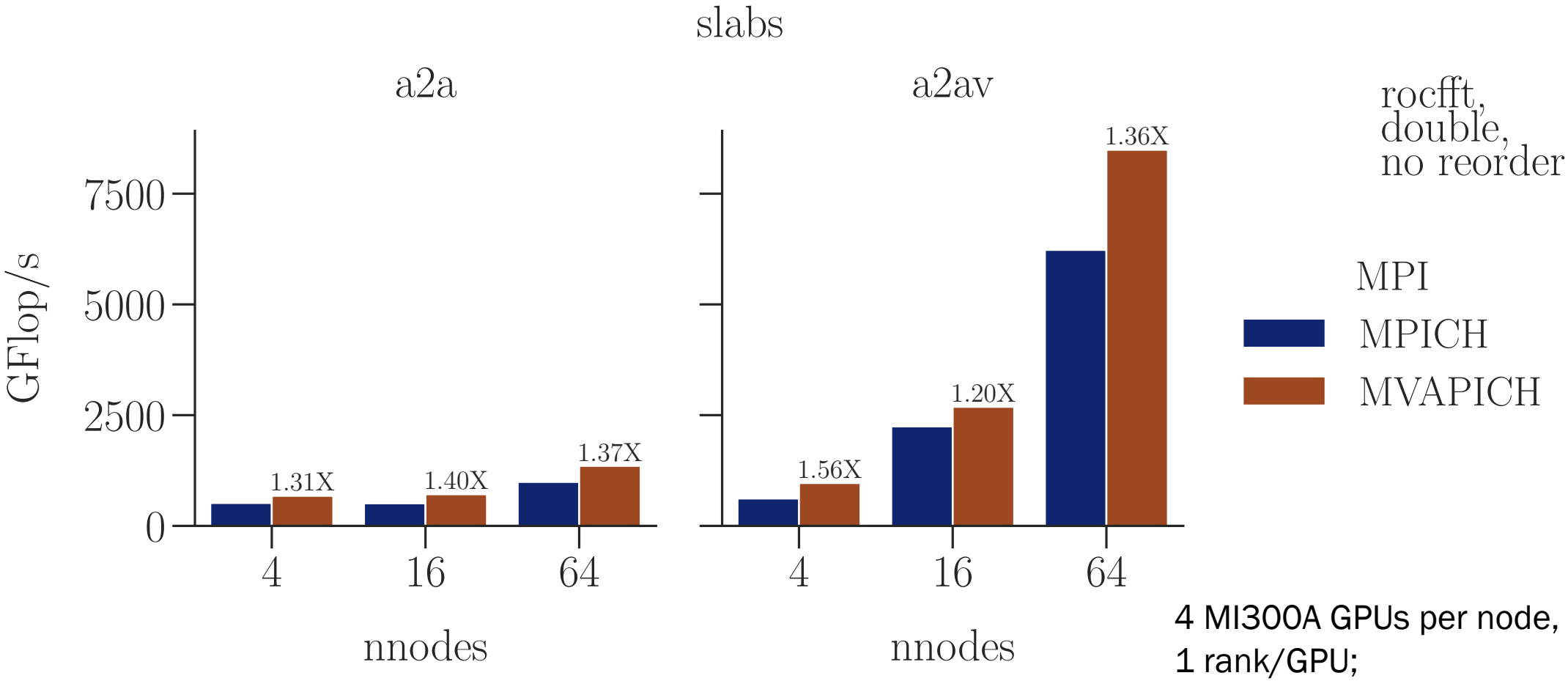
Tuolumne: MVAPICH 4.1 vs MPICH 9.0.1



4 MI300A GPUs per node,
1 rank/GPU;
with ROCm 6.4.2

MVAPICH with good speedup over MPICH for alltoall operations, also with “pencils” decomposition

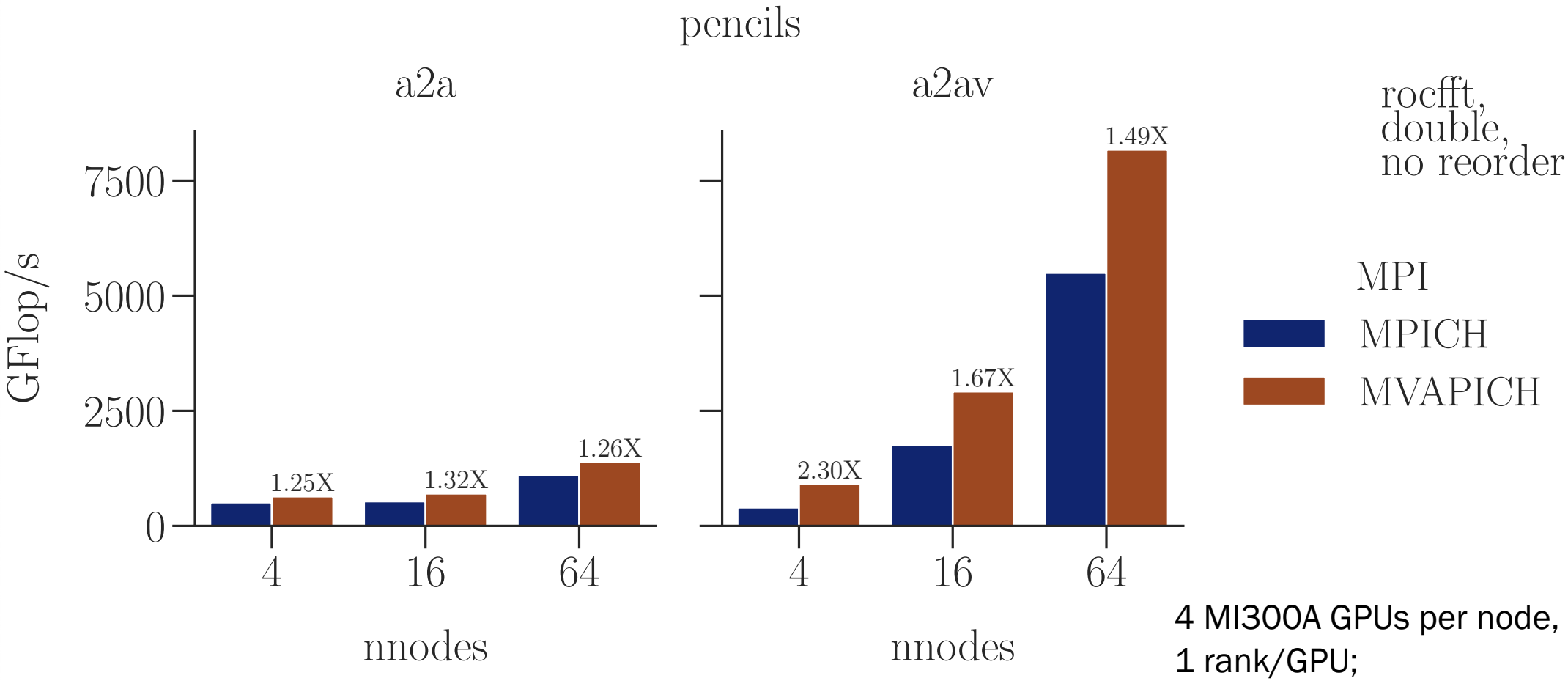
Tuolumne: MVAPICH 4.1 vs MPICH 9.0.1



4 MI300A GPUs per node,
1 rank/GPU;
with ROCm 6.4.2

Double precision – same story as single precision

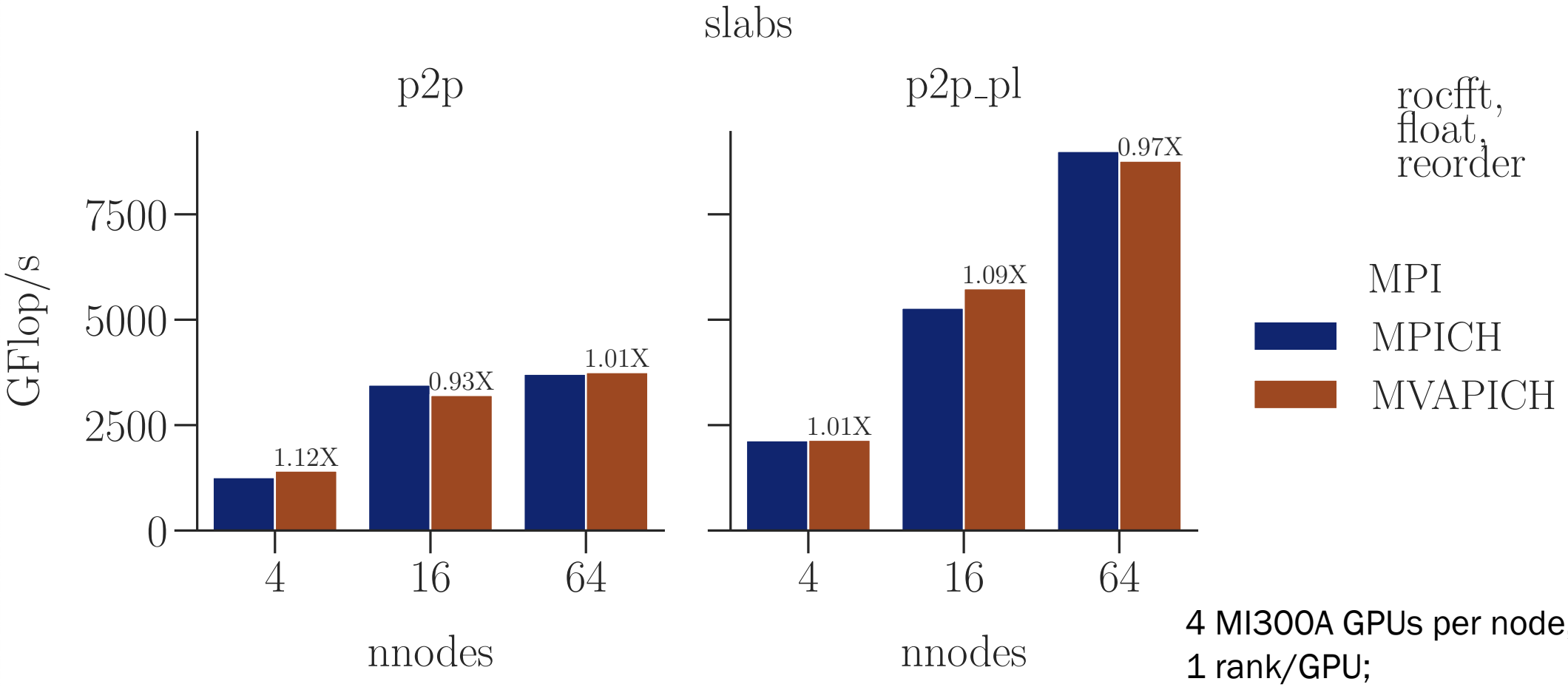
Tuolumne: MVAPICH 4.1 vs MPICH 9.0.1



4 MI300A GPUs per node,
1 rank/GPU;
with ROCm 6.4.2

Double precision – same story as single precision

Tuolumne: MVAPICH 4.1 vs MPICH 9.0.1



4 MI300A GPUs per node,
1 rank/GPU;
with ROCm 6.4.2

MVAPICH on par with MPICH for point-to-point communication options (other configurations similar)

Persistent communication on Tuolumne

- Persistent communication potentially useful for applications: using same FFT setup repeatedly with updated data through timestepping, e.g.
- We have implemented experimental persistent communication options for a2a and p2p in heFFTe

Persistent communication on Tuolumne

- Persistent communication potentially useful for applications: using same FFT setup repeatedly with updated data through timestepping, e.g.
- We have implemented experimental persistent communication options for a2a and p2p in heFFTe

	MPICH 9.0.1	MVAPICH 4.1
5 runs	(No notable speedups)	64 nodes, **float, with reordering: 24% speedup p2p slabs, 6% speedup p2p pencils**
20 runs	18%-33% speedup, 64 nodes, **p2p: slabs**, float (with/without reordering); double (with reordering)	(No notable speedups)

Summary and future possibilities

- heFFTe is a natural benchmark for MPI performance on different machines
- Have seen MVAPICH outperform other MPI implementations on several machines (including ones from this talk: MareNostrum 5 and Tuolumne)
- Potential to make use of MVAPICH's compression capability
  - Handling of complex numbers?
- Possibility for persistent communication?


ICL
INNOVATIVE
COMPUTING LABORATORY
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE