A. Ayala, S. Tomov, J. Dongarra
Innovative Computing Laboratory
University of Tennessee at Knoxville
A. Haidar*
NVIDIA Corporation
** Contribution done while author was at UTK.*

heFFTe

HIGHLY EFFICIENT FFTs FOR EXASCALE

OVERVIEW

Considered one of the top 10 algorithms of the 20th century, the Fast Fourier Transform (FFT) is widely used by applications in science and engineering. Large scale parallel applications targeting exascale, such as those part of the Exascale Computing Project (ECP), are designed for heterogeneous architectures and, currently, some of them rely on efficient state-of-the-art FFT libraries built as CPU kernels. We have developed and released heFFTe [1], a hybrid highly-scalable and robust library for multidimensional FFT computations targeting exascale.

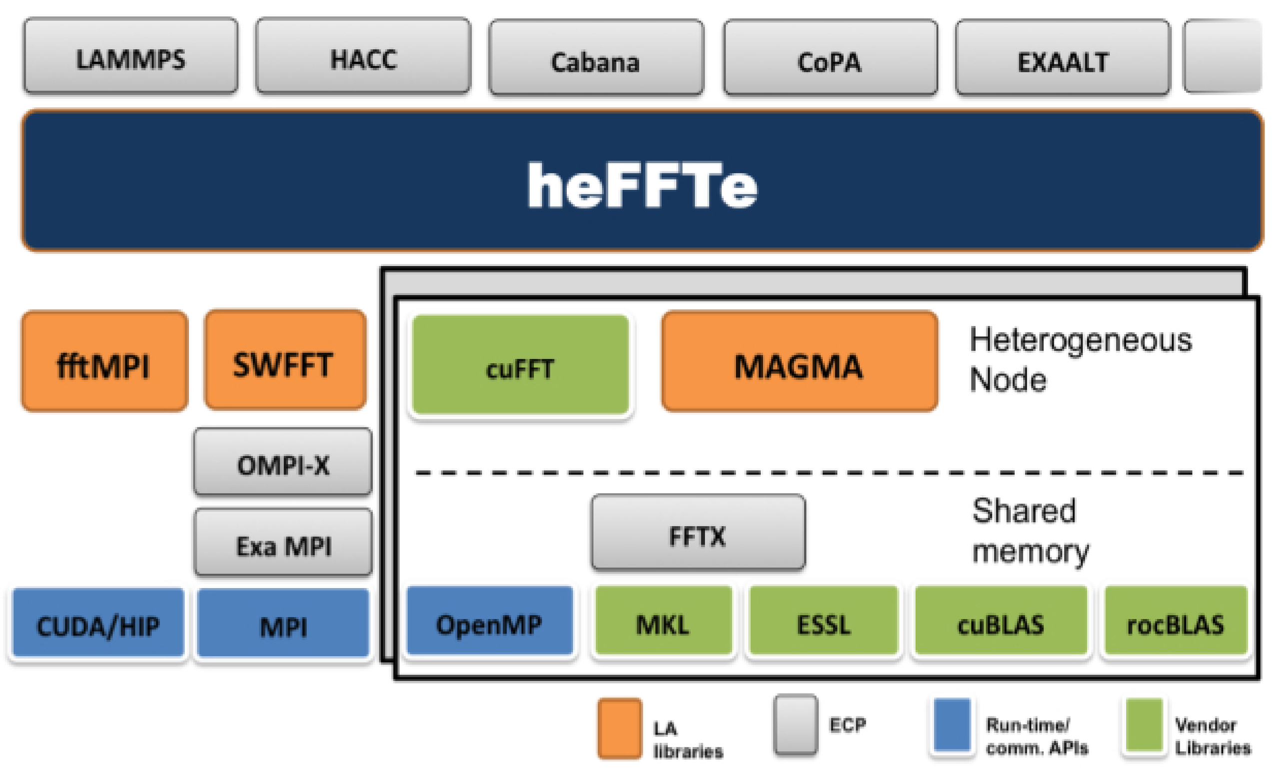

Fig 1. *he*FFTe in the ECP software stack.

Currently, *he*FFTe supports 2D and 3D FFTs in single and double precision. Pencil and slab decompositions are available, and the input can be real or complex. User has control over the processors grid at each stage. Moreover, tuning is available.

METHODOLOGY AND ALGRORITHMIC DESIGN

The parallel implementation of FFT require 2 main tasks:

1. 1D FFT computation: which can be done with FFTW3, MKL, CUFFT, CLFFT, etc.
2. Reshape: involves 3 kernels: **packing** data in contiguous memory, **inter-process communication**, and **unpacking**, performed at the receiving process.

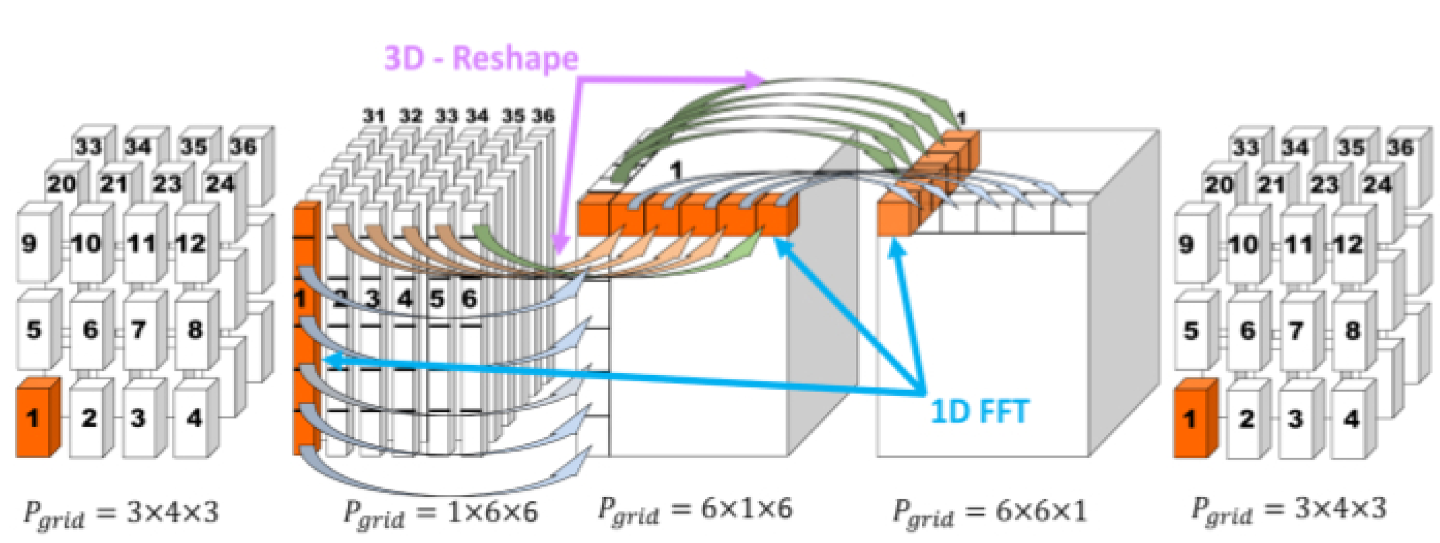

Fig 2. Schematic representation of tasks to perform a 3D FFT.

COMMUNICATION BOTTLENECK

It is well-known that parallel FFT computation is communication bounded, and performance is greatly impacted by hardware limitations [3]. In an effort to optimize MPI communications, we developed ***heffte_alltoall*** which can be tuned to perform all-to-all communication, using non-blocking CUDA-Aware MPI and IPC CUDA memory handlers.

Table 1. MPI routines required by the *he*FFTe library.

Point-to-point routines		Collective routines		Process Topology
Blocking	*Non-blocking*	*Blocking*	*Non-blocking*	
MPI_Send MPI_Recv	MPI_Isend MPI_Irecv	MPI_Alltoallv MPI_Allreduce MPI_Barrier	**heffte_Alltoallv**	MPI_Comm_create MPI_Group

PERFORMANCE AND RESULTS

A. Speedup for Local Kernels Computation

Typical local kernels for packing, unpacking and computation are available on CPU based state-of-the-art libraries, *he*FFTe provides new GPU kernels for these tasks achieving over **40x speedup**.

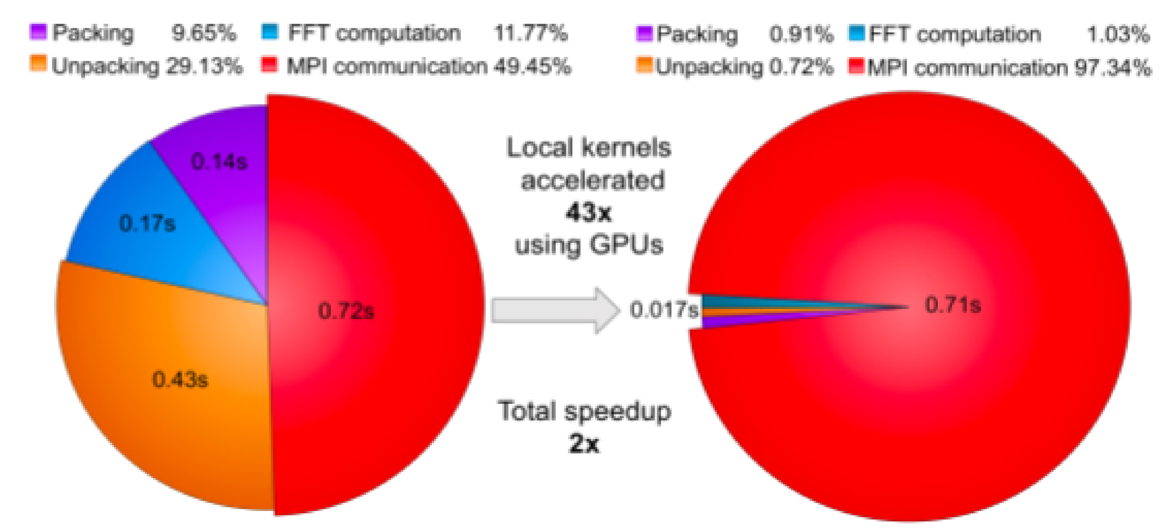

Fig 3. FFT of size 1024x1024x1024 on 4 nodes, 32 MPIs per nodes (CPU kernels, left) vs. 24 MPI processes, 6 MPIs (6 Volta100 GPUs) per node (GPU kernels, right).

B. Scalable Performance

The GPU version of heFFTe has very good weak and strong scalability, and achieves close tp 90% of the roof-line theoretical peak performance.

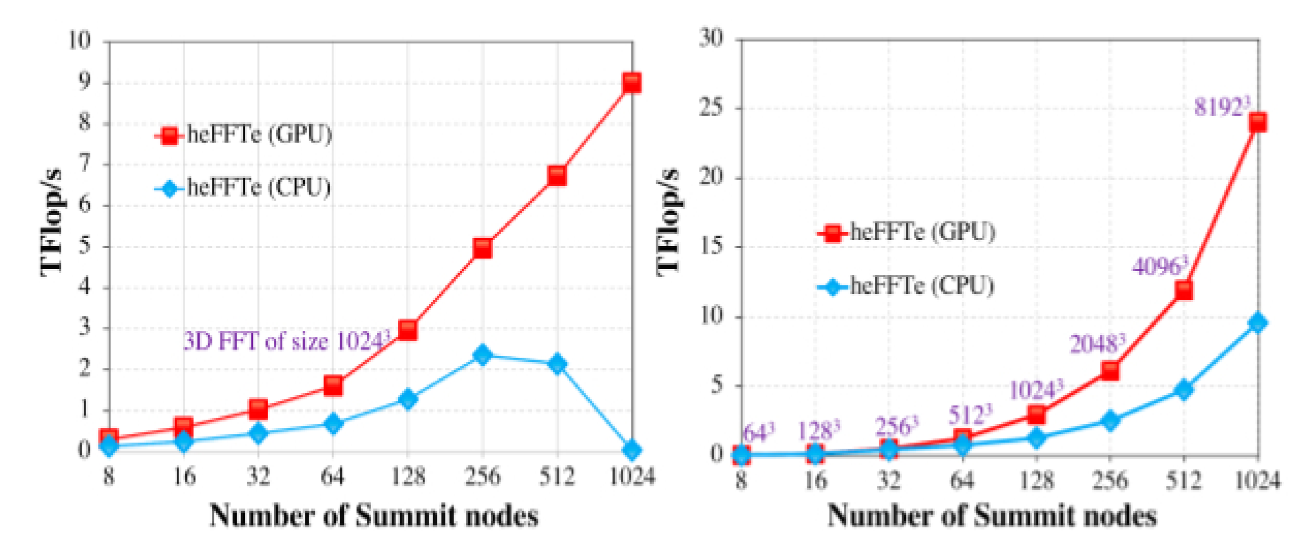

Fig 4. Strong (left) and weak (right) scalability. We use 24 MPIs/node on each case, 1MPI/core for *he*FFTe CPU and 4MPI/GPU-Volta100 for *he*FFTe GPU.

C. Using heFFTe in ECP Applications

Applications such as LAMMPS (EXAALT-ECP) rely on their own FFT library (FFTMPI for this case). We provide wrappers to directly call heFFTe and observe the performance gain, while maintaining good scalability [2].

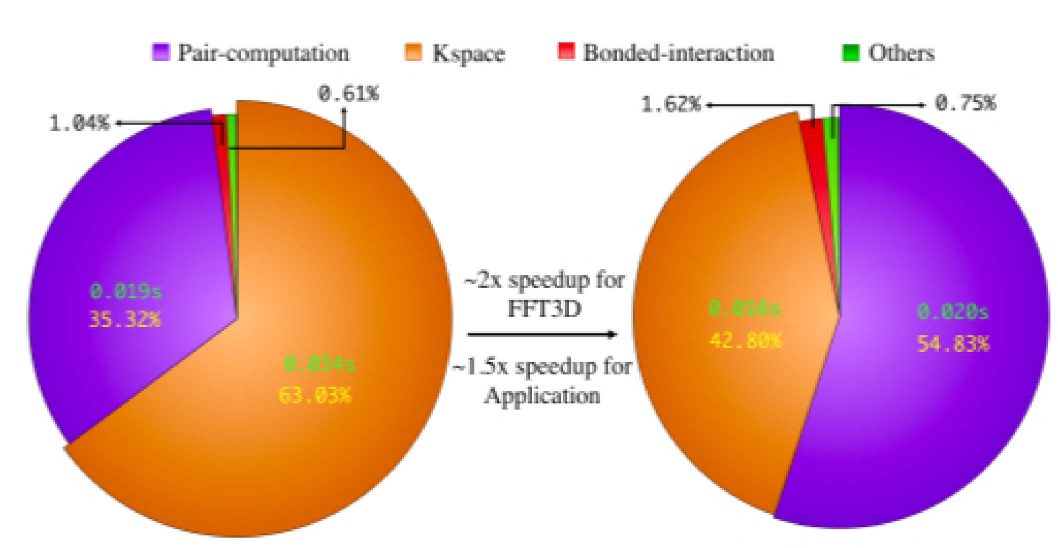

Fig 5. Rhodopsin protein benchmark for LAMMPS, on 2 nodes and 4 MPI ranks per node, on a 128x128x128 FFT grid.

REFERENCES

[1] https://bitbucket.org/icl/heffte/

[2] S. Tomov, A. Haidar, A. Ayala, H. Shaiek, J. Dongarra: *"FFT-ECP Implementation Optimizations and Features Phase."* Tech. Rep. ICL-UT-19-12 (October, 2019)

[3] A. Ayala, S. Tomov, X. Luo, H. Shaiek, A. Haidar, G. Bosilca, J. Dongarra: *"Impacts of Multi-GPU MPI Collective Communications on Large FFT computation"* SC19, Proc. of Workshop on Exascale MPI, Denver, CO (November, 2019)

[4] S. Tomov, A. Ayala, A. Haidar, J. Dongarra: *"FFT-ECP API and High-Performance Library Prototype for 2-D And 3-D FFTs on Large-Scale Heterogeneous Systems with GPUs"*, ICL FFT-ECP STML13-27, January, 2020.

SPONSORED BY

FIND OUT MORE AT
http://icl.utk.edu/fft/