INNOVATIVE COMPUTING LABORATORY

HEFFTE: Highly Efficient FFT for Exascale

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.*

1. Introduction

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) in the USA, are designed for heterogeneous architectures and, currently, most of them rely on efficient state-of-the-art FFT libraries built as CPU kernels.

In this context, we have just released heFFTe [1,2] (pronounced *hefty*) library for FFT computations on heterogeneous platforms. We based our algorithm design on well-know libraries, FFTMPI [3] (used by EXAALT ECP-project) and SWFFT [4] (used by HACC project).

Using GPUs on multiple nodes, we achieve over 40x speedup on local kernels for 3D FFTs, and over 2x speedup for the whole computation. Fig. 1 shows heFFTe within the ECP software stack, and its dependences (e.g. cuBLAS, MAGMA).

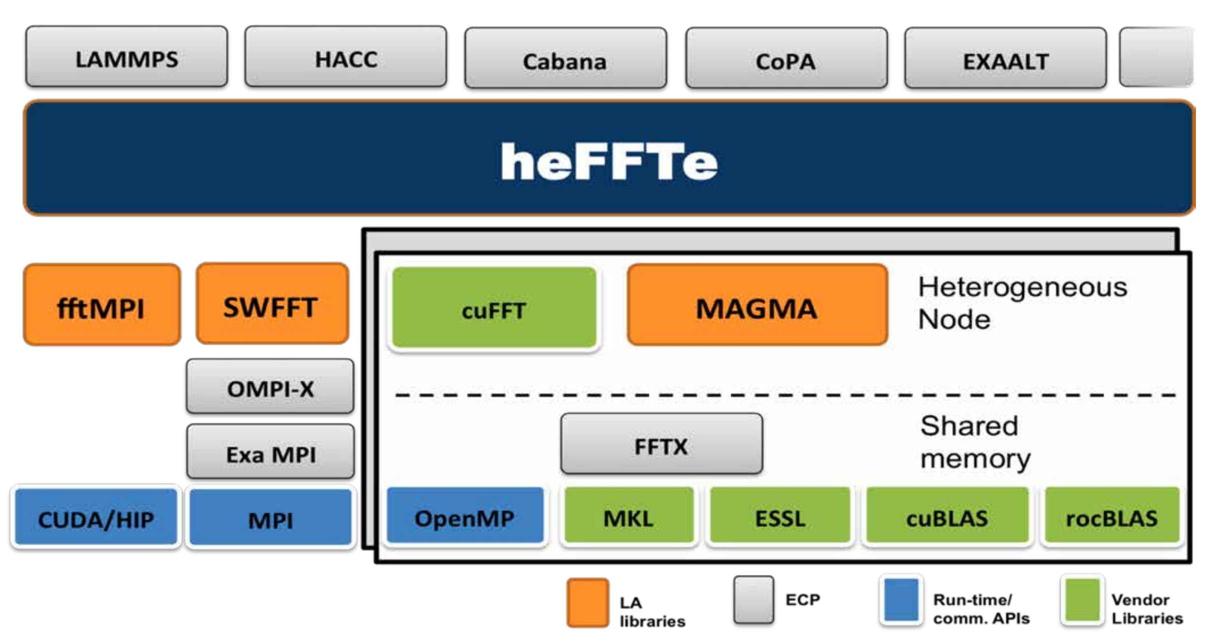

Fig 1. heFFTe in the ECP software stack.

2. Methodology and Algorithmic Design

In Fig. 2, we present the methodology called *pencil-to-pencil*, to compute a 3D FFT, this can be easily generalized for a general number of dimensions. It can be seen as a sequence 2 main tasks:

1. <u>Computation of 1D FFTs:</u> this can be done with standard **1DFFT** libraries such as FFTW3, MKL, CUFFT, CLFFT, etc.
2. <u>Tensor transposition or Reshape:</u> (shown as the arrows) it involves 3 kernels, the first one for **packing** data in contiguous memory, a second one for **inter-process communication**, and the last one is the **unpacking**, performed at the receiving process.

The bottleneck for this algorithm is well-known to be the communication.

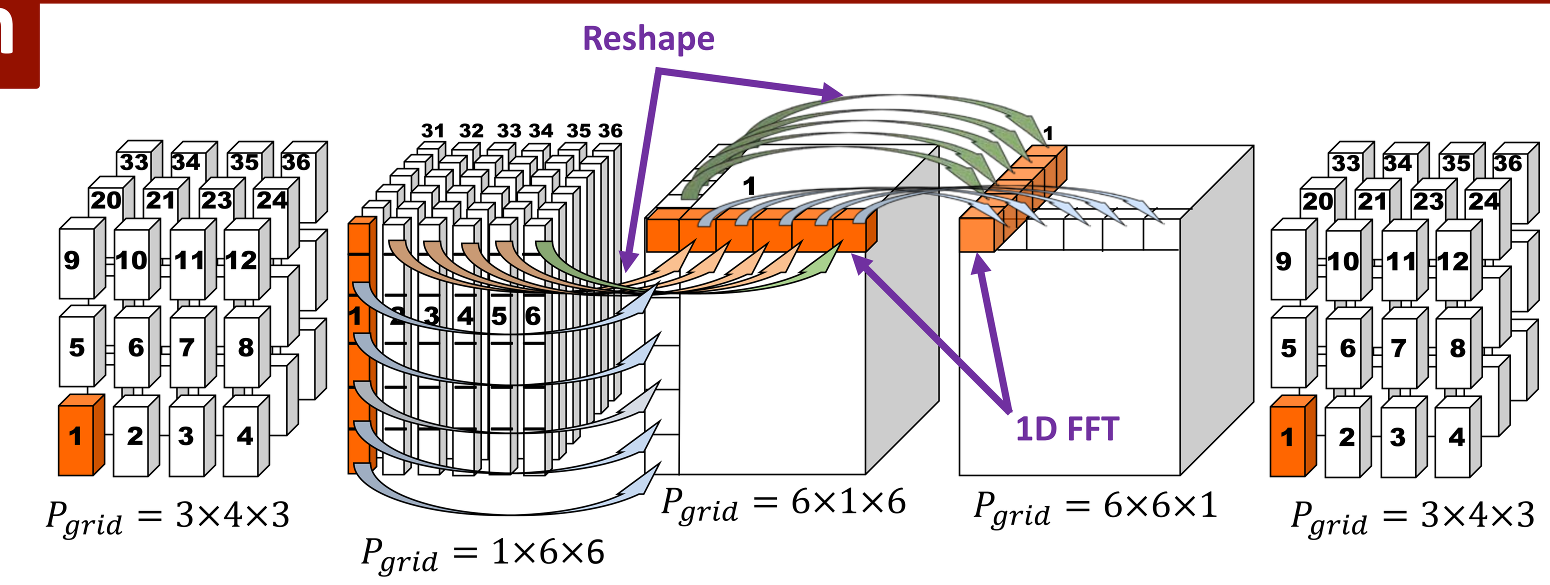

Fig 2. Schematic representation of tasks to perform a 3D FFT.

3. GPU speedup

Local CPU kernels presented on Section 2 are typical on state-of-the-art parallel FFT libraries, heFFTe provide new GPU kernels for these tasks achieving over **40x speedup**.

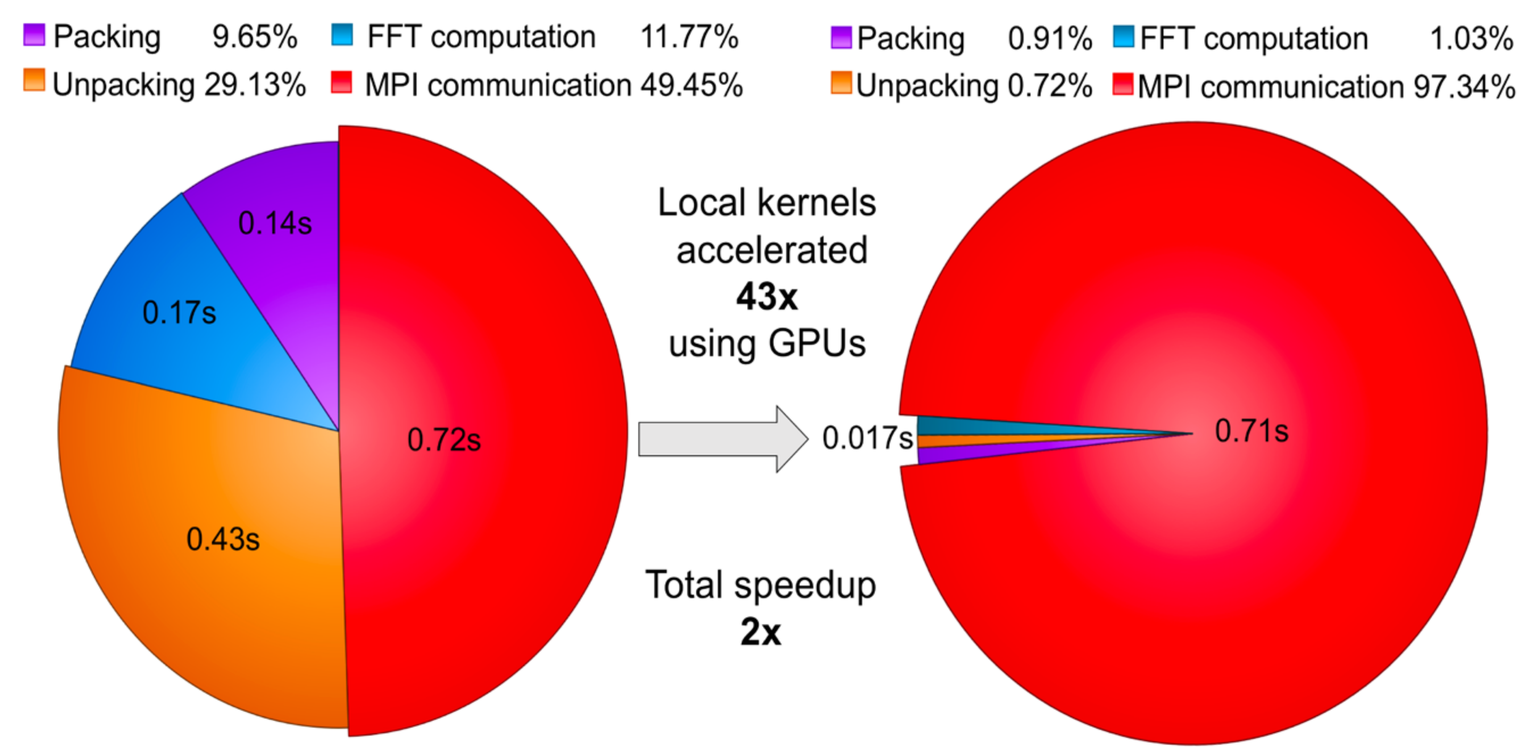

Fig3. 1024^3 FFT on 4 nodes, 32 MPIs per nodes (FFTMPI, Left) vs. 24 MPI processes, 6 MPIs (6 Volta100 GPUS) per node (heFFTe, Right).

4. Scalability

The GPU version of heFFTe has very good weak and strong scalability, and achieves around 20 gigaFlops/s, which is over **2x** the performance reported by state-of-the-art libraries FFTMPI [3] and SWFFT [4] for these sizes. We use 24 MPIs/node on each case, 1MPI/core for heFFTe CPU and 4MPI/GPU-Volta100 for heFFTe GPU.

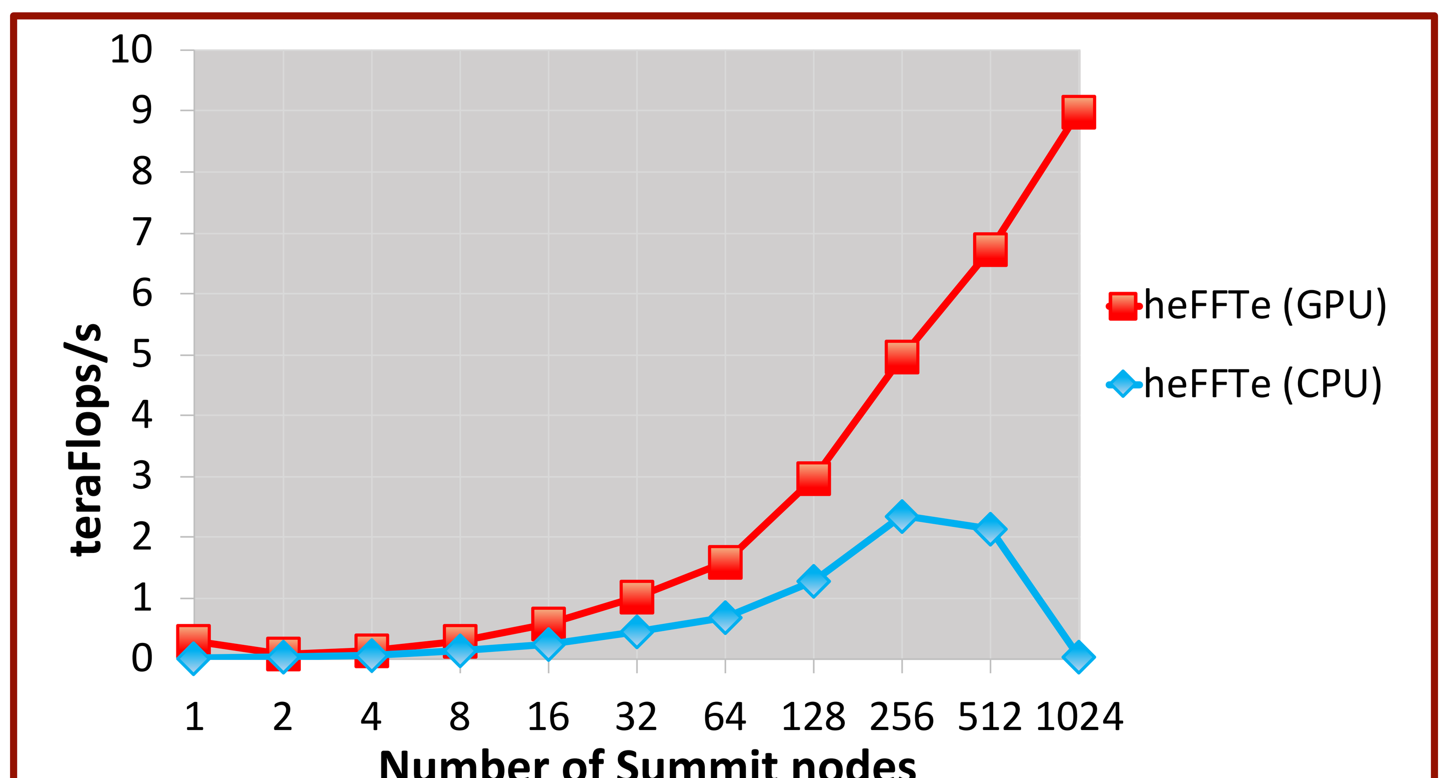

Fig 4. HEFFTE strong scalability

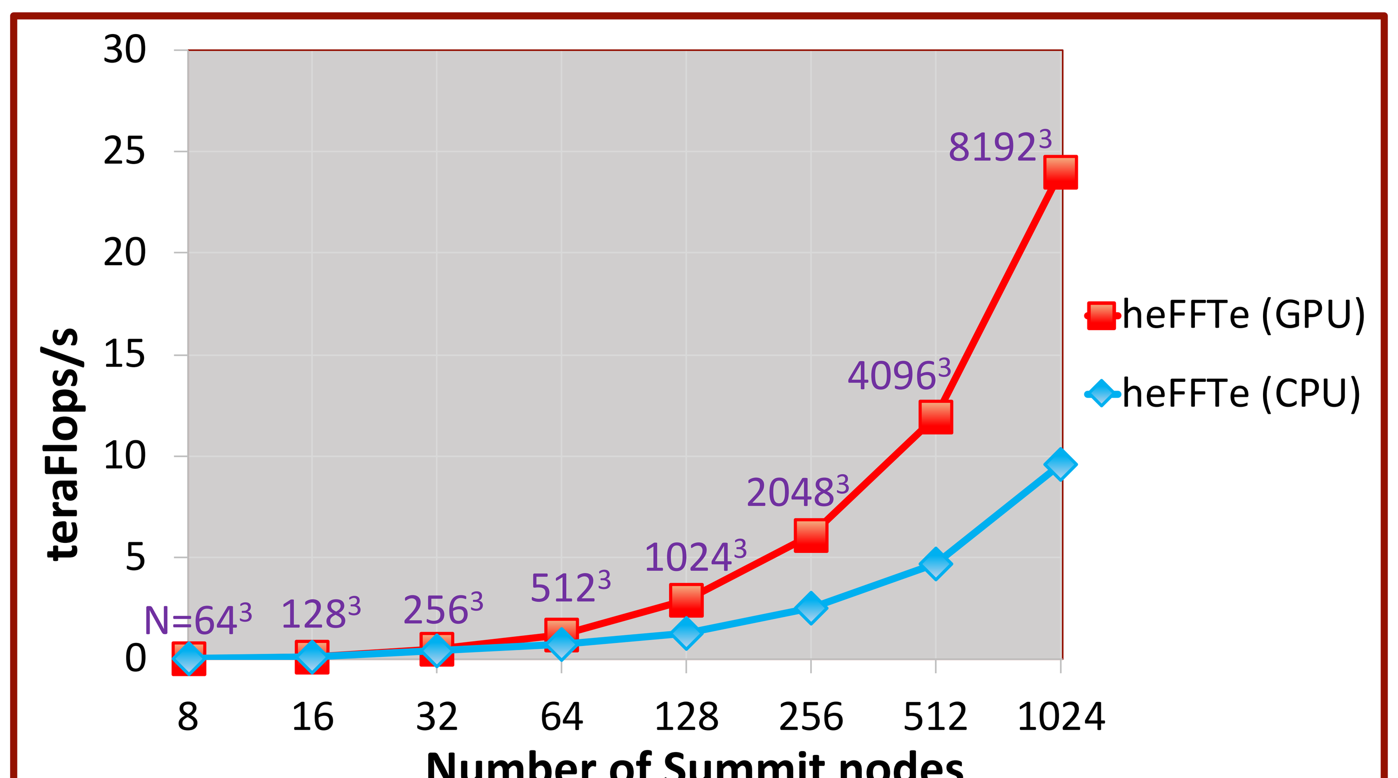

Fig 5. HEFFTE weak scalability

5. Communication Bottleneck

Fig. 3 clearly shows how heFFTe's GPU version is greatly impacted by the communication cost (>95% of runtime). This issue is well-known for parallel FFTs. Typically, MPI all-to-all communication yields better performance than the point-to-point MPI, and this is supported by most libraries. However, there is a lack of hardware aware optimized versions of MPI all-to-all routine, and it is still not available on NVIDIA's NCLL library. On the other hand, most libraries, such as FFTMPI and SWFFT, do not use non-blocking collective MPI routines.

Given these issues, we have developed a routine called ***heffte_alltoall*** which provides several options (that can be tuned) to perform an all-to-all communication, taking advantage of non-blocking MPI routines, as well as of IPC CUDA memory handlers.

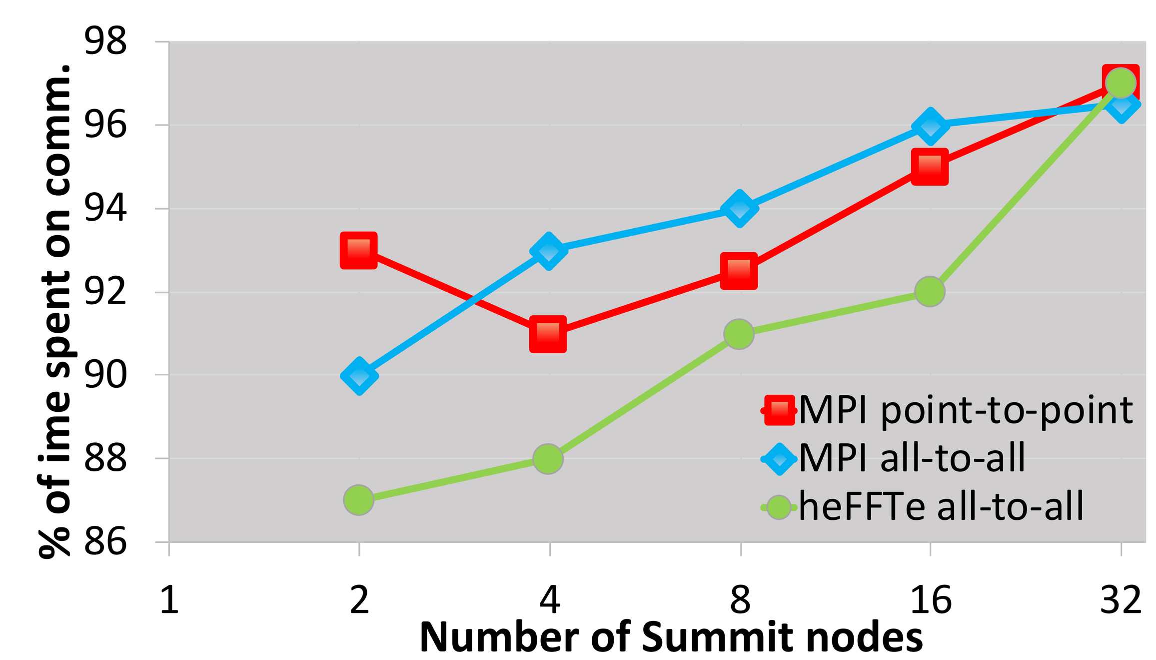

Fig 6. Percentage of time spent in communication for classical MPI routines and heFFTe collective approach.

6. References

[1] https://bitbucket.org/icl/heffte/
[2] Tomov, S., Haidar, A., Ayala, A., Shaiek, H., Dongarra, J.: FFT-ECP Implementation Optimizations and Features Phase. Tech. Rep. ICL-UT-19-12 (2019-10 2019).
[3] S. Plimpton et. al, FFTMPI, a library for performing 2d and 3d FFTs in parallel, Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM, USA (2018).
[4] D. Richards et al., Quantitative performance assessment of proxy apps and parents, Tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA, USA (2018).

SPONSORED BY: U.S. Department of Energy Office of Science, ECP Exascale Computing Project, NNSA National Nuclear Security Administration