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.

METHODOLOGY AND ALGORITHMIC DESIGN
The parallel implementation of FFT requires 2 main tasks:
1. 1D FFT computation: which can be done with FFTW3, MKL, UFFFT, 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 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 heFFTe library.

<table>
<thead>
<tr>
<th>Point-to-point routines</th>
<th>Collective routines</th>
<th>Process Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocking</td>
<td>Non-blocking</td>
<td>Blocking</td>
</tr>
<tr>
<td>MPI_Send</td>
<td>MPI_Recv</td>
<td>MPI_Altoall</td>
</tr>
<tr>
<td>MPI_Recv</td>
<td>MPI_Altoall</td>
<td>MPI_Commm_create</td>
</tr>
<tr>
<td>MPI_Altoall</td>
<td>MPI_Altoall</td>
<td>MPI_Group</td>
</tr>
<tr>
<td>MPI_BARRIER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, heFFTe provides new GPU kernels for these tasks achieving over 40x speedup.

Fig 3. FFT of size 1024x1024x1024 on 4 cores, 32 MPIs per node (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 to 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 heFFTe CPU and 4MPI/GPU/Volta100 for heFFTe GPU.

C. Using heFFTe in ECP Applications
Applications such as LAMMPS (EXALI1/EC) 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/icsrheffte/