1. Introduction

Many large-scale application-software require Fast Fourier Transforms (FFT), e.g., within the Exascale Computing Project (ECP) of the United States.

Hybrid CPU-GPU systems are widely used and are expected for the upcoming exascale machines. FFT libraries targeting such architectures have been accelerated via tuning and asynchronous kernel evaluation on GPUs [2], obtaining up to 2x speedup compared to fully CPU libraries.

We present techniques to further accelerate FFT computation by overcoming the communication bottleneck, we provide architecture-aware selection of FFT algorithm, a novel All-to-All routine (which can considerably speedup default MPI standard routines), and a mixed-precision implementation.

2. Even faster FFTs

Multidimensional FFT are computed by a sequence of 1D or 2D FFTs, with intermediate data reshapes. The latter is the most expensive task (>90% of runtime [2]), moving data among processes (4), typically in an All-to-All fashion [2,3].

We developed a mixed-precision version for heFFTe, which exploits GPU power to compress data (using casting/ZFP) to save in All-to-All cost (which usually takes over 90% of runtime [2,3]). We used a ring version of our OSC_A2A routine.

3. Mixed-Precision FFT

Fig. 3, shows that heFFTe linearly scales. We use a 3D complex-data grid, and compute both: single (FP32) and double (FP64) precision FFTs.

We developed a mixed-precision version for heFFTe, which exploits GPU power to compress data (using casting/ZFP) to save in All-to-All cost (which usually takes over 90% of runtime [2,3]). We used a ring version of our OSC_A2A routine.

CR2 means a compression ratio of 2 times. CR2 is up to 2.6x faster than FP64 and up to 1.4x faster than FP32. CR4 is up to 5x faster than FP64 and up to 2.6x faster than FP32. CR2 validation error is $O(10^{-6})$ while for FP32 it is $O(10^{-3})$; i.e., we move the same data volume faster, while getting a better quality FFT output.

4. Communication model

We introduce a novel communication model for hybrid-distributed FFTs which can adapt to any architecture [2], and gives a theoretical estimation of the reshaping cost. This model assumes that fast communication is available within a node, e.g., Summit at ORNL, which has NVLINK connections.

5. Applications

Following figures show some applications heFFTe targets, to some of which it has already been integrated, to accelerate FFT calculus while ensuring scalability.

Cosmology: HACC

Molecular Dynamics: EXAALT

Particle Simulations: CPAs

Tomography – Image processing

Deep-learning

Signal processing – Randomized FFT – Matrix Compression

6. References

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