An Approach for Large-scale Distributed FFT Framework on GPUs

Yichang Hu[1], BinBin Zhou[1] and Lu Lu[2]

[1]School of Compute Science and Engineering, South China University of Technology, Guangzhou, China [2] Advanced Micro Devices Inc

Abstract & Introduction

Multi-nodes heterogeneous systems aspire to meet the increasing demands from parallel computing FFT and we present a highly efficient GPU-based distributed FFT framework by adapting the Cooley-Tukey recursive FFT algorithm in this work.

Typical approaches used by parallel libraries are *pencil* and *slab* decomposition. Our work employs *slab* decomposition (Fig.1) and consists of four steps, batched 2D FFTs, all-to-all communication (Fig.2), local transpose, batched 1D FFTs.

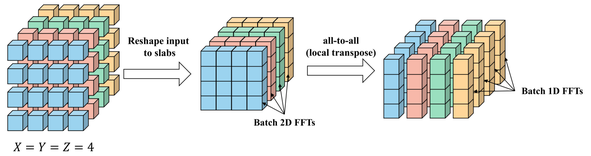

Fig. 1 Slab decompostion

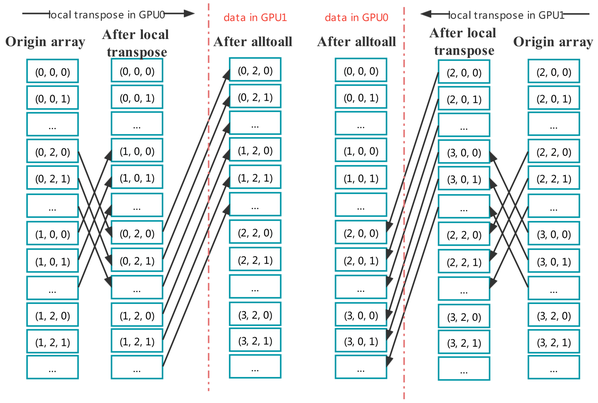

Fig. 2 Illustration of the local and global communication

Design and methodology

Dynamic low-dimensional FFT kernels generation.

- Group small sequence butterflies.
- Non-strided data access mode.
- Runtime compliation by *hipRTC* APIs.

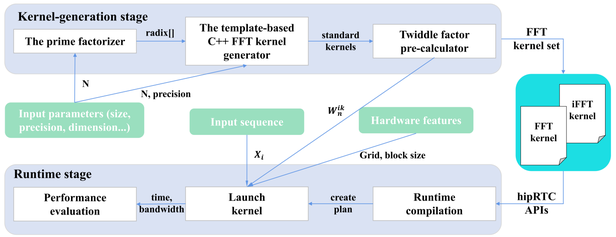

Fig. 3 Concept diagram for FFT kernels generation

Hybrid OpenMP-MPI programming model.

- Multiple threads for intra-node communication to control multiple devices.
- GPU-aware MPI for inter-node communication to avoid the high latency through CPU.
- Efficient asynchronous management of packing and communication (shown in Fig.4, which demonstrates the classical packing process and a novel approach in our design, overlapping packing/unpacking with MPI communication).

Specific high performance transpose kernel.

- A 3D point on (i,j,k) can be mapped into the $(i{\times}Y{\times}Z+j{\times}Z+k)^{th}$ element, and the destination location after transposition is (D is the number of devices):

$$X{\times}\lceil Y/D\rceil{\times}Z{\times}(j/\lceil Y/D\rceil)+i{\times}H{\times}Z+jmod\lceil Y/d\rceil{\times}Z+k$$

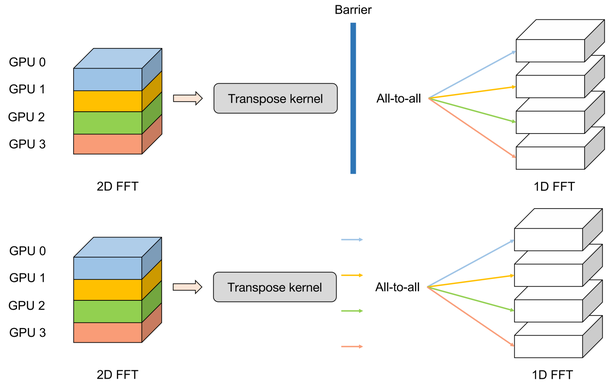

Fig. 4 Comparison of synchronous and asynchronous communication

Evaluation

The performance of our work is compared with heFFTe[1] and FFTW on a heterogeneous system, which are the state-of-art distributed FFT libraries. Fig. 5 is case study on AMD MI100 and MI250 GPUs, and scalability comparison is in Table 2. It is clear that our implementations consistently outperform heffte, while not compromising on communication overhead.

Table 1 Platform Information

Platform	
CPU	64-core AMD EPYC 7763
GPU	AMD Instinct MI100 or MI250
ROCm	ROCm 5.2.0
MPI	OpenMPI 4.1.0 with UCX 1.11.0
Network	Infiniband (4X HDR)

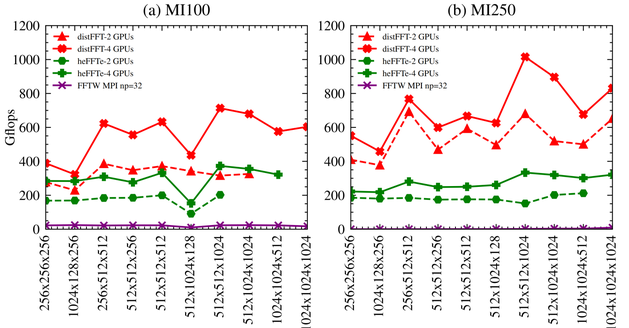

Fig. 5 Case study

Table 2 Scalability on MI250 (GFlops)

GPUs	1	2	4	8	16	32
heFFTe	160	320	591	1200	2010	3694
Ours	498	818	1538	2581	4312	6942

Conclusion

This work proposes an apporach for large-scale distributed FFT framework on GPUs that makes following three primary novel contributions:

- Novel data reshuffling and memory transfer pattern.
- Asynchronous management for packing and communication.
- Unit-stride data layout and specific transpose kernel for high efficient memory operations.

Our work achieves more than 40x speedup over CPU FFT libraries and about 2x speedup over heFFTe.

References

[1] A. Ayala, S. Tomov, A. Haidar, and J. Dongarra, “heffte: Highly efficient fft for exascale,” in Computational Science – ICCS 2020, 2020.