GPUDirect MPI Communications and Optimizations to Accelerate FFTs on Exascale Systems

Hejer Shaiek
hejer@icl.utk.edu
Innovative Computing Laboratory,
The University of Tennessee
Knoxville, Tennessee, USA

Stanimire Tomov
tomov@icl.utk.edu
Innovative Computing Laboratory,
The University of Tennessee
Knoxville, Tennessee, USA

Alan Ayala
aayala@icl.utk.edu
Innovative Computing Laboratory,
The University of Tennessee
Knoxville, Tennessee, USA

Azzam Haidar
azzamhaidar@nvidia.com
Nvidia Corporation
Santa Clara, California, USA

Jack Dongarra
dongarra@icl.utk.edu
Innovative Computing Laboratory,
The University of Tennessee
Knoxville, Tennessee, USA

ABSTRACT

Fast Fourier transforms (FFTs) are used in applications ranging from molecular dynamics and spectrum estimation to machine learning, fast convolution and correlation, signal modulation, wireless multimedia applications, and others. However, FFTs are memory bound, and therefore, to accelerate them, it is crucial to avoid and optimize the FFTs' communications. To this end, we present a 3-D FFT design for distributed graphics processing unit (GPU) systems that: (1) efficiently uses GPUs' high bandwidth, (2) reduces global communications algorithmically, when possible, and (3) employs GPUDirect technologies as well as MPI optimizations in the development of high-performance FFTs for large-scale GPU-accelerated systems. We show that these developments and optimizations lead to very good strong scalability and a performance that is close to 90% of the theoretical peak.

KEYWORDS

GPU, GPUDirect, CUDA-Aware MPI, FFT, FFT-ECP, ECP

ACM Reference Format:
Hejer Shaiek, Stanimire Tomov, Alan Ayala, Azzam Haidar, and Jack Dongarra. 2019. GPUDirect MPI Communications and Optimizations to Accelerate FFTs on Exascale Systems. In *Proceedings of EuroMPI '19 Posters.* , 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Fast Fourier transforms (FFTs) are among the most important kernels used in numerous domain applications, including molecular dynamics, cosmology, signal modulation, spectrum estimation, machine learning, and others. The Exascale Computing Project (ECP), funded by the US Department of Energy (DOE), is focused on accelerating the delivery of an exascale-capable computing ecosystem. This includes FFTs, as FFTs are present in the software stack for almost all ECP applications. Indeed, FFT performance can affect the application scalability on larger machines [1, 3, 4], and FFTs must therefore be highly accelerated. Of interest are multi-dimensional FFTs. A 3-D distributed FFT, for example, consists of three stages of computation and Message Passing Interface (MPI) communications between each of them. The computation itself is memory bound, but the MPI communications are the biggest challenge.

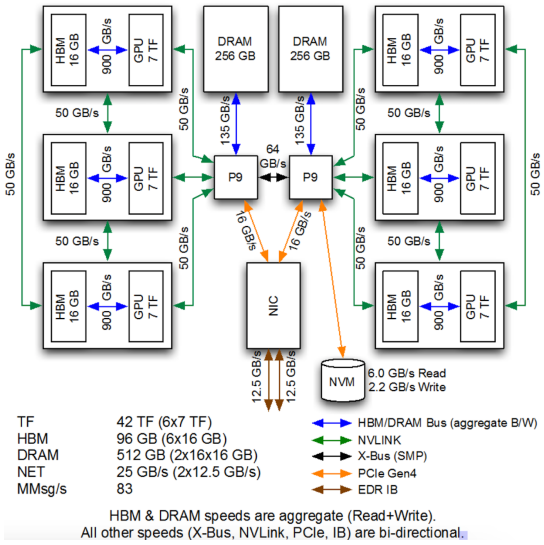

Figure 1: Summit node architecture and connectivity.

The FFT developed under ECP is FFT-ECP [7]. FFT-ECP aspires to be a new and sustainable high-performance FFT library for exascale platforms that leverages the large investments in FFT software by the broader HPC community. Indeed, there are many FFT libraries. Analysis and performance comparisons of major FFTs on current architectures are available in [8]. The results [8] motivated basing the FFT-ECP design on FFTMPI [3], a CPU FFT library developed by Sandia National Laboratory (SNL). In this work we describe the latest efforts and experiences in porting the FFT-ECP computations to graphics processing units (GPUs), as well as accelerating the MPI communications using GPUDirect technologies. Figure 1 illustrates a target architecture—the Summit supercomputer at Oak Ridge

EuroMPI '19 Posters, September 11-13, 2019, Zurich, Switzerland

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

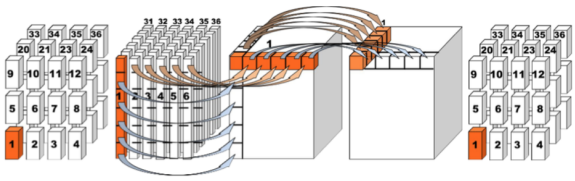

Figure 2: FFT-ECP framework design with flexible API

National Laboratory (ORNL). Shown are the node architecture and connectivity. The goal is to design FFTs that reduce, as well as optimize, communication on all connectivity and memory hierarchy levels: (1) leverage the GPU's 900 GB/s bandwidth, (2) reduce global inter-node communications, and (3) employ GPUDirect technologies as well as MPI optimizations to efficiently communicate both intra-node (through the 50 GB/s NVLinks) and inter-node (through the 2×12.5 GB/s InfiniBand).

2 FFT DESIGN FOR GPUS

FFT-ECP is based on FFTMPI and follows the same algorithmic patterns (see Figure 2). Data transpositions create contiguous vectors (*pencils*) in the x, y, and z directions, and call an external FFT library for the 1-D FFT calculations on the corresponding pencils. One of the most important features of FFTMPI is the flexibility of input and output data layout. That flexibility is kept in FFT-ECP. The algorithm works as follows: given a 3-D tensor $A = \{a_{i,j,k}\}$ distributed on P processors, the first step is to make the data belonging to the first dimension available on the same processor (i.e., $\forall j_0, k_0, \exists$ process P_l s.t. $a_{i,j_0,k_0} \in P_l$ for $\forall i$) so that the computation of the first dimension can start on P_l. After it is finished, data is transposed again and 1-D FFTs are computed along the second dimension, and the same goes for the last one. Then, if necessary, a final step of communication is performed to build the output layout.

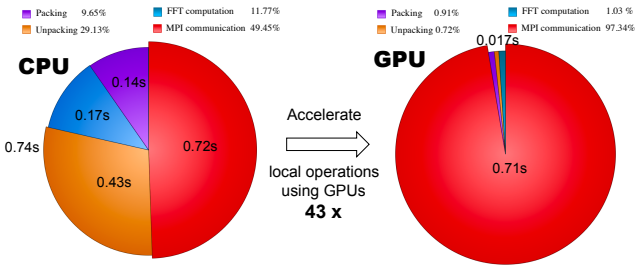

Figure 3: Profile of a 3-D FFT of size 1024^3 on 4 CPU nodes—using 128 MPI processes, i.e., 32 MPIs per node, 16 MPIs per socket (Left) vs. 4 GPU nodes—using 24 MPI processes, i.e., 6 MPIs per node, 3 MPI per socket, 1 GPUs per MPI (Right)

A typical profile of running 3-D FFTs on multi-core CPUs (e.g., Summit) is given on Figure 3, Left. The times reported (in seconds) are for double complex arithmetic performing Forward FFT, starting from brick distribution and ending with the same brick distribution over the processes. The local FFT operations (within an MPI process) are memory bound and take about 50% of the time, in this case. Included are local packing, unpacking, and 1-D FFTs (using any

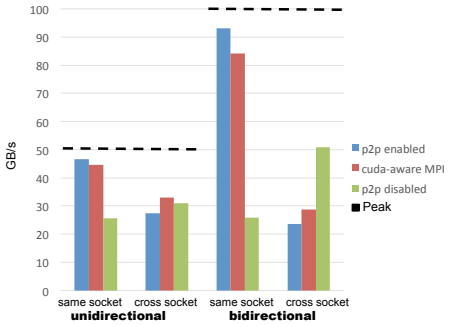

Figure 4: Bandwidth Benchmark for different types of p2p with message size = 40MB

CPU FFT library that gives best performance, e.g., Intel MKL FFT, FFTW, or any other). The other 50% are in MPI communications.

Note that the local computations are memory bound and can therefore benefit from GPUs' high bandwidth (900 GB/s). To leverage this, we ported all operations to the GPU: packing and unpacking using the fast matrix transpositions available in MAGMA [6], and the 1-D FFTs using cuFFT [2] from NVIDIA. This accelerated the local operations 43×, in this case. MPI communications remained about the same, using CUDA-Aware MPI from IBM (Spectrum MPI) doing GPUDirect communication.

Other versions were also developed to support different user interfaces (e.g., data starting from the CPU memory). In general, any version that involves movement of data to the CPU memory was significantly slower due to the costly data movements (through the 50 GB/s NVLinks vs. the 900 GB/s GPU bandwidth).

3 COMMUNICATION REDUCTION IN FFTS

The current bottleneck in FFTs are MPI communications. As illustrated on Figure 3, Right, MPI communications take more than 97% of the total time.

To optimize communications, we developed benchmarks and tested different mechanisms for transferring data. Figure 4 summarizes the comparison for GPUDirect Peer to Peer (P2P) communications. The comparisons are when a GPU communicates with GPUs on the same socket, cross socket GPUs, unidirectional and bidirectional, and using Nvidia GPUDirect technologies in single process versus CUDA-aware MPI communications from different MPI processes. This helped identify drawbacks, summarized in the conclusions, as some of these communications are far away from the theoretical peaks. Similar benchmarks and studies were performed on All2All communications. The best performing versions were selected to achieve the results in Figure 3, Right, and in general for the tuning of the FFT-ECP library.

We achieved the best performance with a combination of P2P Spectrum MPI communications for FFTs on up to four Summit nodes, and Spectrum MPI All2All for more than four Summit nodes. This is illustrated on Figure 5. Note that the code also has very good, strong scalability. The computations are in double complex arithmetic and the gigaFLOP/s rate reported assumes $5N^3 log_2 N^3$ floating-point operations (FLOPs), where N is the size for the N^3 3-D FFT performed. This computation also starts from bricks and

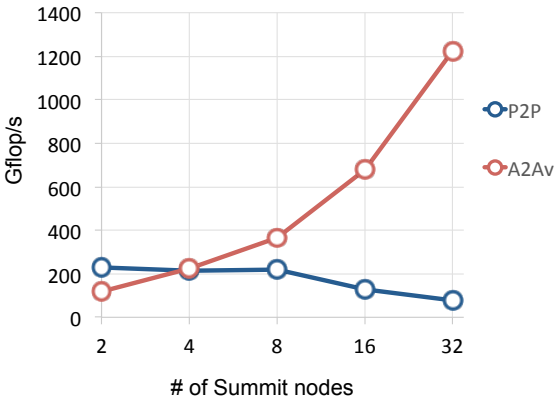

Figure 5: Strong scalability on 3D FFTs of size 1024^3, comparing the use of All2All vs. P2P MPI communications using CUDA 9.1 and Spectrum MPI 10.2

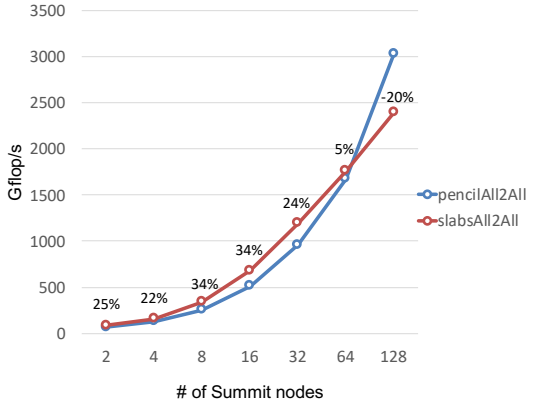

Figure 6: Strong scalability on 3D FFTs of size 1024^3, comparing the use of slab vs. pencil decompositions using CUDA 10.1 and Spectrum MPI 10.3

ends up with the same brick distribution over the MPI processes (on the GPUs' memories).

To further reduce communications, we explored algorithmic 3-D FFT variations that can reduce communications. In particular, we looked into ways to reduce the stages of communication. For example, this can be done with so called *slab* decompositions, where a process will get the data needed for two directions of 1-D FFTs, thus avoiding the step of communications between the two. This can save between 25% and 50% of the total time.

This can be further extended, for example, by testing whether all the data fits in the memory of a single GPU, and if that is the case, we can send all the data to a single GPU, do the 3-D transform without any communication and send it back to build the output data layout requested by the the user. Other extensions are to do this agglomeration to a single node, or a subset of the compute resources.

Figure 6 illustrates the effect of using slab decompositions to reduce communications, and hence to increase performance. The starting splitting is based on bricks, so the pencil decomposition results in four communication stages: transpose in x, transpose in y, transpose in z, and move back to the original brick decomposition. The slab decomposition on the other hand results in only three stages: move to x-y slabs, transpose in z, and move back to the original brick decomposition. This is a theoretical 25% reduction in communications that results in a corresponding 25% increase in performance. For 8 and 16 nodes, speedup is 35% and 32%, respectively, and goes down as the number of nodes used grows. The effect becomes negative at 128 nodes due to lack of parallelism (at 128 the number of MPIs/GPUs used is 768).

Note the results in Figures 5 and 6 differ by up to 20% performance degradation when CUDA and MPI were updated on Summit. With the new MPI we also do not see the positive effect of selecting proper GPU network affinities as we used to when using Spectrum MPI 10.2 [7]. GPU affinity was an important tuning parameter.

4 CONCLUSIONS AND FUTURE DIRECTIONS

FFT-ECP has achieved significant acceleration of 3-D FFTs using GPUs. Results show very good scalability, including strong scalability, due to highly optimized and reduced MPI communications. Locally, on a GPU, we have accelerated the FFT operations about 40× compared to the multi-core CPU counterpart (using V100 GPUs vs. Power9 CPUs on the Summit supercomputer at ORNL). This acceleration uses the high bandwidth that GPUs provide (up to 900 GB/s). At this stage, the FFT computation is dominated (97% and above) by MPI communications, so any further improvement in the overall speed will come from the development of CUDA-aware MPI optimizations.

Figure 4 shows that there are areas that need improvement, especially in the cross-socket GPU Direct communications for both uni- and bi-directional communications. Furthermore, there is performance degradation between MPI Spectrum versions 10.2 and 10.3. Performance loss of 20%, as illustrated in Figures 5 and 6, is significant and needs to be addressed.

Current performance can be best described by the bandwidth achieved through a node. For example, performance on a 1024^3 3-D FFT is about 400 gigaFLOP/s on 8 nodes. One can compute that data is sent from the node at a 21.8 GB/s rate: the formula is Bytes sent ($= 16 \times 1024^3/8$) over the time to send ($= 0.98 \times (5 \times 1.024^3 \times log_2(1024^3)/400)/4$, where 4 represents the four stages used). Note that the node also receives about the same amount of data at the same time, so the bi-directional bandwidth achieved is 43.6 GB/s out of 50 GB/s, which is 87.2% of the theoretical peak.

Future work will concentrate on MPI optimizations for strong scaling on many nodes, optimizations for a single node for cross-socket communications, and algorithmic optimizations based on slab partitions, or other reductions of the computational resources used that can lead to reduced communications. More versions and support for different FFT features are being added in FFT-ECP. Application-specific optimizations and use of mixed-precision calculations [5] that can result in additional acceleration due to reduced communications are also of interest.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations (the Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem.

REFERENCES

[1] JD Emberson, N. Frontiere, S. Habib, K. Heitmann, A. Pope, and E. Rangel. 2018. *Arrival of First Summit Nodes: HACC Testing on Phase I System.* Technical Report MS ECP-ADSE01-40/ExaSky. Exascale Computing Project (ECP).

[2] CUDA Nvidia. 2018. CUFFT library.

[3] Steven Plimpton, Axel Kohlmeyer, Paul Coffman, and Phil Blood. 2018. *fftMPI, a library for performing 2d and 3d FFTs in parallel.* Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[4] Steven J. Plimpton. [n.d.]. FFTs for (mostly) Particle Codes within the DOE Exascale Computing Project, 2017. ([n. d.]).

[5] Anumeena Sorna, Xiaohe Cheng, Eduardo F. D'Azevedo, Kwai Wong, and Stanimire Tomov. 2018. Optimizing the Fast Fourier Transform Using Mixed Precision on Tensor Core Hardware. *2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW)* (2018), 3–7.

[6] S. Tomov, J. Dongarra, and M. Baboulin. 2010. Towards Dense Linear Algebra for Hybrid GPU Accelerated Manycore Systems. *Parellel Comput. Syst. Appl.* 36, 5-6 (2010), 232–240. DOI: 10.1016/j.parco.2009.12.005.

[7] Stanimire Tomov, Azzam Haidar, Alan Ayala, Daniel Schultz, and Jack Dongarra. 2019. *Design and Implementation for FFT-ECP on Distributed Accelerated Systems.* ECP WBS 2.3.3.09 Milestone Report FFT-ECP ST-MS-10-1410. Innovative Computing Laboratory, University of Tennessee. revision 04-2019.

[8] Stanimire Tomov, Azzam Haidar, Daniel Schultz, and Jack Dongarra. 2018. *Evaluation and Design of FFT for Distributed Accelerated Systems.* ECP WBS 2.3.3.09 Milestone Report FFT-ECP ST-MS-10-1216. Innovative Computing Laboratory, University of Tennessee. revision 10-2018.