Large-scale FFT on GPUs

Survey and Scalability Analysis

Alan Ayala
alan.ayala@amd.com
2023-08-22

AMD
together we advance_

Contents

The Fast Fourier Transform (FFT)

- The FFT is an algorithm developed by Cooley-Tukey in 1965.
- Considered one of the top 10 algorithms of the 20^{th} century.

Definition (Discrete Fourier Transform (DFT))

Let x be an m-dimensional array of size $N := N_1 \times N_2 \times \cdots \times N_m$. Its DFT is defined by $y := DFT(x)$, obtained as:

$$y(k_1, k_2, \ldots, k_m) := \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cdots \sum_{n_m=0}^{N_m-1} x(n_1, n_2, \ldots, n_m) \cdot e^{-2\pi i \left(\frac{k_1 n_1}{N_1} + \frac{k_2 n_2}{N_2} \cdots + \frac{k_m n_m}{N_m} \right)}.$$

- A naive DFT costs $\mathcal{O}(N^2)$.
- Using the FFT, the cost can be reduced to $\mathcal{O}(N \log_2 N)$.
- Alternative implementations widely used in the literature, such as the Bluestein Algorithm (1970).

FFTs on Large-scale Applications

- Several K-space methods in ultrasound and molecular dynamics applications use large 3D FFTs (sizes $\gtrsim 1024^3$ or greater), and generally constitute over half of the total computation time [1].
- Cosmological simulation codes, such as HACC, rely on large scale FFTs that can be greater than $20,000^3$. HACC kernels based on FFTs have been shown to scale up to over 1.5 million MPI ranks [2].
- Reverse Time Migration (RTM) Applications rely on operators that heavily use batches of FFTs.

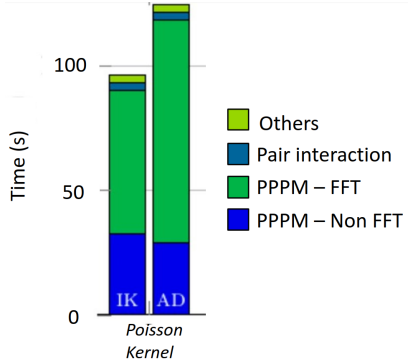


Figure: LAMMPS Water benchmark with a cutoff of 3 angstroms. The FFT part of the PPPM solver can take over 50% of runtime [3].

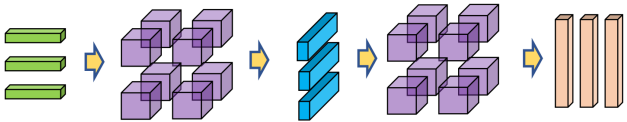
State-of-the-art

- 2DECOMP&FFT Library is written in Fortran 2008 standard and has recently re-started development. It supports NVIDIA GPUs using cuFFT, CUDA-aware MPI, and the NVIDIA collective communication library (NCCL) [4].
- AccFFT Library was built on the idea of overlapping computation and blocking collective communication by reducing the PCIe overhead. It supports NVIDIA GPUs and it is one of the few supporting four dimensional transforms and FFT-based tools such as Laplace and divergence useful for diverse scientific applications [5].
- cuFFTMp Library is a vendor implementation offered by NVIDIA. The communication is performed either by a provided MPI library or by nvSHMEM. They also provide a reshape API with functions such as cufftXtSetDistribution and cufftMpReshape to compute data transpositions without performing FFTs [6].
- DiGPUFFT Library is one of the first parallel libraries with GPU support. Their implementation was experimental and achieved by replacing default FFTW backend of P3DFFT with cuFFT. From their experiments, the authors envisaged that the network cost would account for around 40% of runtime [7].

State-of-the-art

- FFTE Library includes a wide range of optimizations on NVIDIA GPUs, based on 1-D and 2-D decompositions. However it does not support arbitrary grid sizes and numbers of MPI processors [8].
- fftMPI Library is widely used for applications in molecular dynamics. As part of its KOKKOS package, LAMMPS extended fftMPI support to enable computation on AMD and NVIDIA GPUs [9].
- heFFTe Library extends portability to AMD and Intel GPUs. It also provides features to support batched parallel FFTs, convolutions, and sine/cosine transforms [10].
- MFFT Library uses a "high-precision computation, low precision communication" strategy, and a shared-exponent floating-point number compression technique [11].
- SpFFT Library, supports AMD and NVIDIA GPUs. Their parallel implementation is based on OpenMP and MPI frameworks that use slab decomposition for the spatial domain and pencil decomposition for the frequency domain. [12]

Multidimensional FFT Algorithm



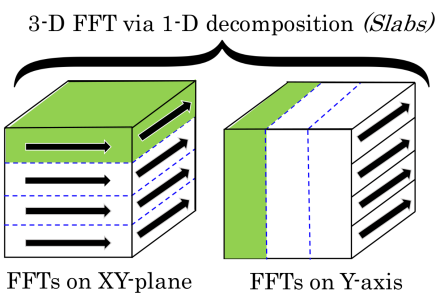
1: **Input:** Distributed m-dimensional input, processors grids at input and output: P_{in}, P_{out}
2: Transfer data from P_{in} to a pencil or slab FFT grid.
3: Get n_{reshapes}: number of reshapes needed.
4: Define processor grids and communication groups for each reshape step
5: **for** $i \leftarrow 1, \cdots, n_{\text{reshapes}}$ **do**
6: *// inter_bpb is true for pencil ↔ brick reshape (no FFTs are computed)*
7: **if not** inter_bpb **then**
8: Compute local FFTs *// 1-D or 2-D FFTs, typically done on GPUs*
9: **end if**
10: *// Begin reshape phase*
11: Pack data in contiguous memory
12: **for** P on my Communication Group **do**
13: *Transfer* computed data to neighbor processors *// Using binary or collective MPI*
14: **end for**
15: Unpack data in contiguous memory
16: *// End reshape phase*
17: **end for**
18: Transfer data from pencil or slab grid to P_{out} arrangement.

FFT Parallel Implementation: Slabs

For the slabs-decomposition, a single data transfer is required among processes distributed in the y axis. Since every process holds N/Π data and communicates a $1/\Pi$ fraction to each of its $(\Pi - 1)$ neighbors, the communication cost is:

$$T_{\text{slabs}} = (\Pi - 1)\left(L + \frac{S_p(N)}{W \cdot \Pi^2}\right) \approx \Pi L + \frac{S_p(N)}{W \cdot \Pi},$$

while the number of messages sent by each process is $M_{\text{slabs}} = \Pi - 1$.

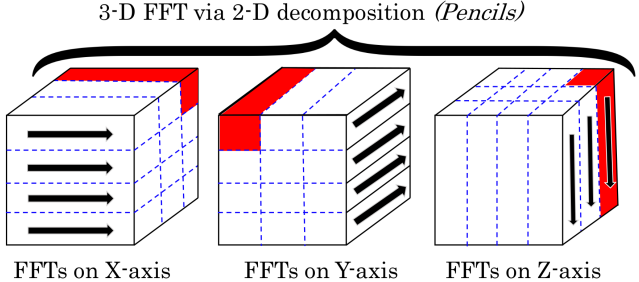


FFT Parallel Implementation: Pencils

For the pencils-decomposition, Π is split over a 2-D grid of processes, $\Pi := P \times Q$, with P processes along the x-axis and Q processes along the y axis. Then, simply by considering each pencil as a slab, we can apply the idea above and find that the communication cost is:

$$T_{\text{pencils}} = (P-1)\left(L + \frac{S_p(N)}{W \cdot P \cdot \Pi}\right) + (Q-1)\left(L + \frac{S_p(N)}{W \cdot Q \cdot \Pi}\right) \approx (P+Q)L + \frac{2S_p(N)}{W \cdot \Pi},$$

while the number of messages sent by each process is $M_{\text{pencils}} = P + Q - 2$.

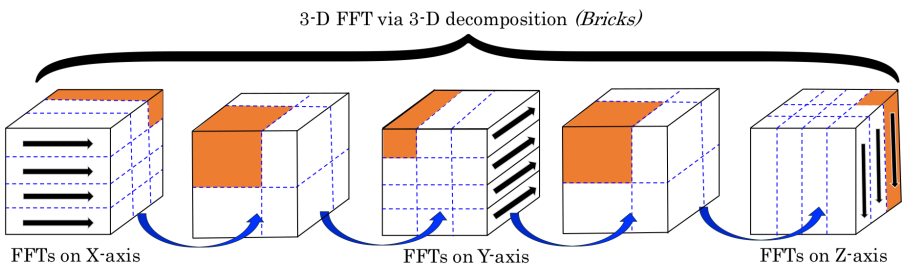


FFT Parallel Implementation: Bricks

For the bricks-decomposition, we need two grids, one to handle the pencil-shapes, $\Pi = P \times P = \rho \times \rho \times \rho$. Four Exchanges are needed:
pencils $\rightarrow$ bricks $\rightarrow$ pencils $\rightarrow$ bricks $\rightarrow$ pencils.
Each brick needs data from other $\rho - 1$ pencil-shaped processors, and the amount of data that each of these processors sends is $N/\Pi/\rho$.

$$T_{\text{bricks}} = 4\left[(\rho-1)\left(L + \frac{S_p(N)}{W \cdot \rho \cdot \Pi}\right)\right] \approx 4\rho L + \frac{4S_p(N)}{W \cdot \Pi},$$

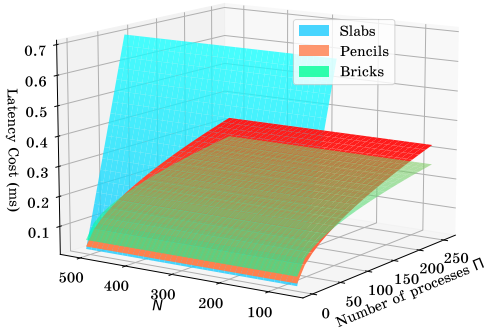
while the number of messages sent by each process is $M_{\text{bricks}} = 4\rho$.



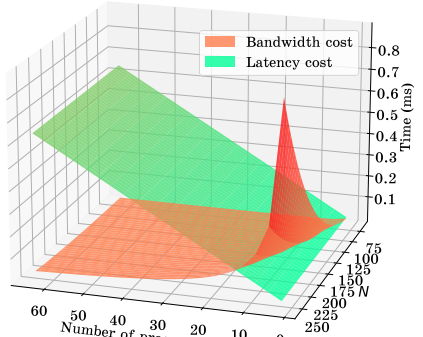
Asymptotic Communication Cost for 3-D Array Reshape (Transposition)

Decomposition	Communicating Processors	Communication cost (s)	
		Due to latency	Due to bandwidth
Slabs (1-D)	Π	ΠL	$\frac{S_p(N)}{W\Pi}$
Pencils (2-D)	$\sqrt{\Pi}$	$2\sqrt{\Pi}L$	$\frac{2S_p(N)}{W\Pi}$
Bricks (3-D)	$\sqrt[3]{\Pi}$	$4\sqrt[3]{\Pi}L$	$\frac{4S_p(N)}{W\Pi}$

Latency model cost for different FFT decompositions



Communication cost comparison for Slabs decomposition



Communication Time for a Parallel FFT of size N on n nodes, using Inter-Node Bandwidth Based on Network Topology

Bisection bandwidth: minimum volume of communication allowed between any two halves of the network, cutting a minimum number of links.

Network Topology	Example System	Bisection Bandwidth	Communication Time		
			Slabs	Pencils	Bricks
Fat-Tree	Oakforest-PACS Selene Summit	$O(n)$	$O\left(Ln + \frac{N}{Wn}\right)$	$O\left(Ln^{1/2} + \frac{N}{Wn^{1/2}}\right)$	$O\left(Ln^{1/3} + \frac{N}{Wn^{1/3}}\right)$
Dragonfly	Cray XC40 Frontier Piz Daint				
Tapered Fat-Tree	Sunway TaihuLight				
Torus 3-D	Blue Gene/P	$O\left(n^{2/3}\right)$	$O\left(Ln + \frac{N}{Wn^{2/3}}\right)$	$O\left(Ln^{1/2} + \frac{N}{Wn^{1/3}}\right)$	$O\left(Ln^{1/3} + \frac{N}{Wn^{2/9}}\right)$
Torus 5-D	Blue Gene/Q	$O\left(n^{4/5}\right)$	$O\left(Ln + \frac{N}{Wn^{4/5}}\right)$	$O\left(Ln^{1/2} + \frac{N}{Wn^{2/3}}\right)$	$O\left(Ln^{1/3} + \frac{N}{Wn^{4/15}}\right)$
Torus 6-D	Fugaku	$O\left(n^{5/6}\right)$	$O\left(Ln + \frac{N}{Wn^{5/6}}\right)$	$O\left(Ln^{1/2} + \frac{N}{Wn^{5/12}}\right)$	$O\left(Ln^{1/3} + \frac{N}{Wn^{5/18}}\right)$

Effective Bandwidth for Different MPI Implementations

- Frontier has Cray-MPICH by default.
- Overall, it performs better than OpenMPI.

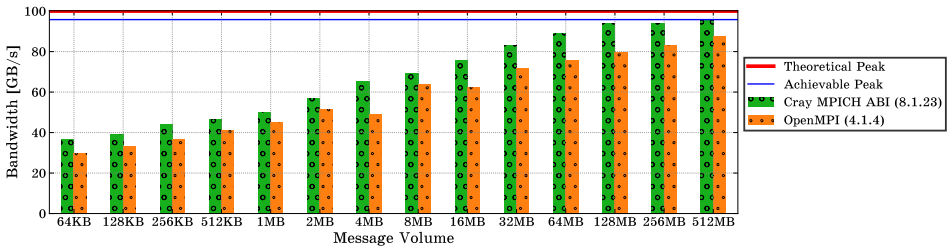


Figure: Comparison of achievable bandwidth using different MPI distributions on Frontier.

Effective Bandwidth for Different MPI Implementations

- Summit has Spectrum-MPI by default.
- For small sizes, alternative distributions can perform better. This was more notorious with Spectrum-MPI 10.3.

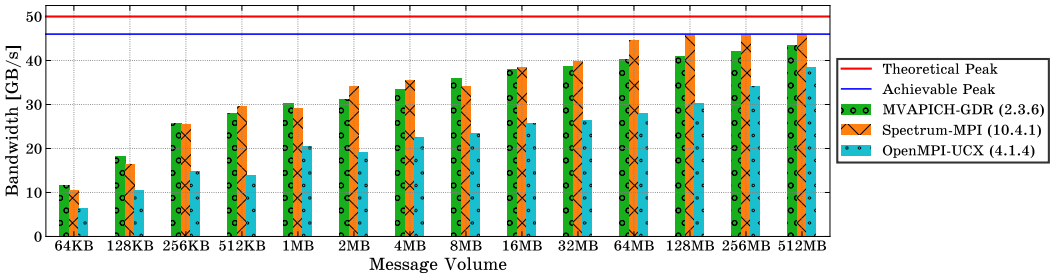


Figure: Comparison of achievable bandwidth using different MPI distributions on Summit.

Impact of Collective Operation

In Fig. 4, we compare three MPI all-to-all implementations on up to 128 nodes on Summit, using a 3-D complex-to-complex FFT of size 512^3 with pencils decomposition starting and ending on brick shape (4 reshapes).

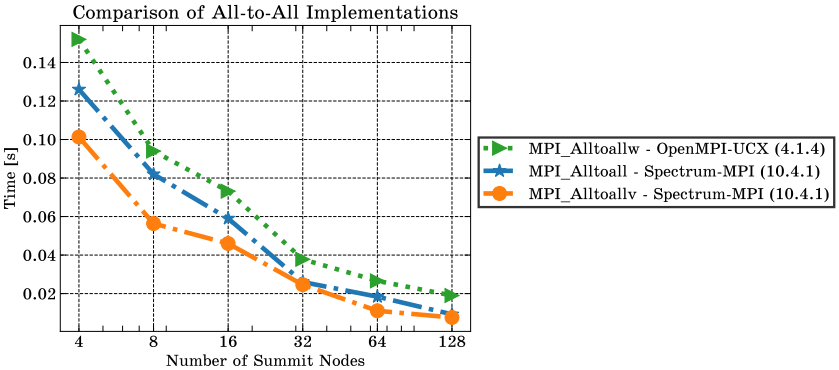


Figure: Comparison of communication time for different All-to-All MPI implementations. Using 6 GPUs per node on Summit.

The impact of GPU-awareness of MPI Implementation

Around a decade ago, Nandapalan et al. performed experiments using GTX580 GPUs (200 GB/s links) and PCIe 2.0 (8GB/s links). Compared to global data exchange via shared memory, the CUDA-based UVA approach reduced the execution time of a case study 3D FFT by up to 49% [13]. This was one of the first signs that direct D2D exchanges are a key factor in obtaining high performance on multi-GPU systems.

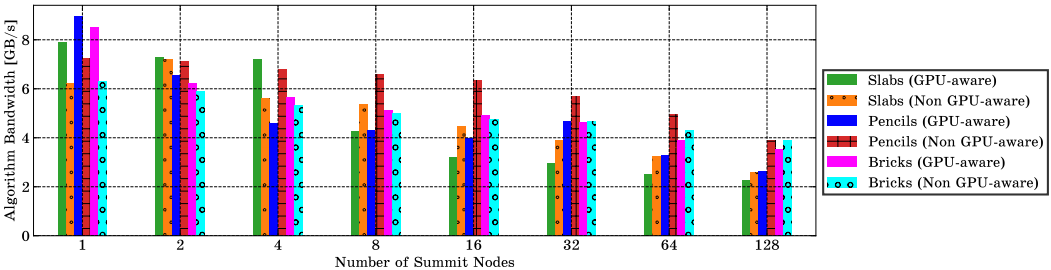


Figure: Average algorithm bandwidth for a complex-to-complex 3-D FFT of size 1024^3 using heFFTe with 6 V100 GPUs per Summit node and switching the GPU-awareness of Spectrum-MPI.

Scalability Analysis

Communication cost becomes an issue when adding more MPI resources. This causes linear scaling to stop. Therefore, for an efficient and scalable FFT, it is important to leverage each decomposition approach, MPI distribution, and the type of communication.

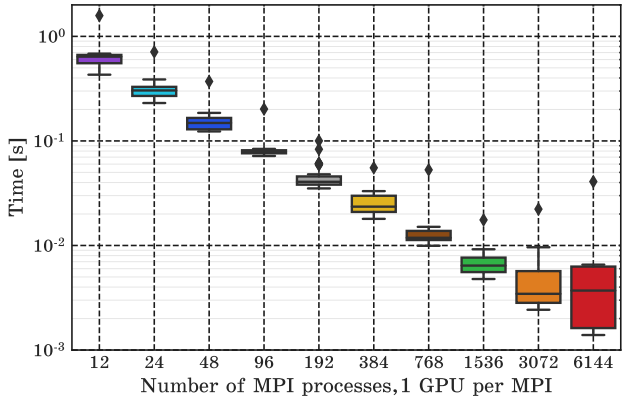


Figure: Scalability of a single reshape using MPI_Alltoallv, 1024^3 FFT on Summit with 6 GPUs/node.

Scalability Analysis

The following figure shows how the selection of the intermediate grids to be used for the global transposition (slides 4-7) is critical to ensure scalability of the parallel FFT [14].

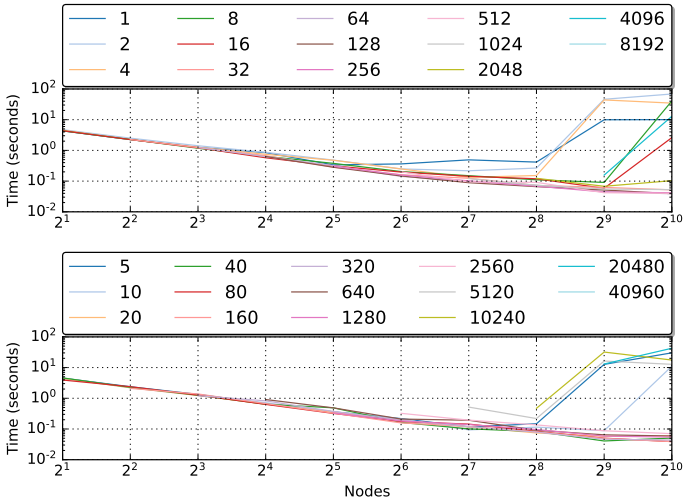


Figure: Scalability of a 3D FFT of size 1024^3 using P3DFFT library on different number of Summit nodes with different shapes of two-dimensional processors grids: processor rows as power of 2: 2^ℓ of row MPI ranks (top) and power of 2 multiple of 5: $5 \times 2^\ell$ of row MPI ranks (bottom).

Conclusions

- At large-scale, latency effects considerably impact FFT scalability, and algorithmic tuning is necessary to keep scaling (slide 17).
- Our mathematical models can help to define an optimal communication framework for different hardware and network topologies.
- To ensure parallel FFT scalability, it is critical to choose:
  - The right decomposition approach (slabs, pencils, or bricks).
  - The type of MPI communication (slide 15).
  - The processors grids used for the global transpositions (slide 18).
- In the literature, collective MPI operations are preferred over point-to-point since they show better performance at large-scale. From slide 15, we conclude that choosing the right collective routine can yield significant speedups.
- GPU awareness in MPI distributions must be tuned for each implementation type and architecture to get the best performance of modern interconnections (slide 16).
- Parallel FFT performance vary for each MPI distribution. At large scale, it is important to select an MPI that handles well a large number of small volume messages.

Bibliography I

[1] Bradley Treeby, Jiri Jaros, Alistair Rendell, et al. "Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method". In: *The Journal of the Acoustical Society of America* 131 (June 2012), pp. 4324–36. DOI: 10.1121/1.4712021.

[2] Susan M Mniszewski, James Belak, Jean-Luc Fattebert, et al. "Enabling particle applications for exascale computing platforms". In: *The International Journal of High Performance Computing Applications* 35.6 (2021), pp. 572–597. DOI: 10.1177/10943420211022829.

[3] William McDoniel, Markus Höhnerbach, Rodrigo Canales, et al. "LAMMPS' PPPM Long-Range Solver for the Second Generation Xeon Phi". In: *High Performance Computing*. Cham: Springer International Publishing, 2017, pp. 61–78.

[4] N. Li et al. *2DECOMP&FFT library*. Available at https://github.com/2decomp-fft/2decomp-fft. 2023.

[5] Amir Gholami, Judith Hill, Dhairya Malhotra, et al. "AccFFT: A library for distributed-memory FFT on CPU and GPU architectures". In: *CoRR* abs/1506.07933 (2015). arXiv: 1506.07933.

Bibliography II

[6] NVIDIA. *Multinode Multi-GPU: Using NVIDIA cuFFTMp FFTs at Scale*. 2023. URL: https://developer.nvidia.com/blog/multinode-multi-gpu-using-nvidia-cufftmp-ffts-at-scale.

[7] Kenneth Czechowski, Chris McClanahan, Casey Battaglino, et al. "On the communication complexity of 3D FFTs and its implications for Exascale". In: June 2012. DOI: 10.1145/2304576.2304604.

[8] Daisuke Takahashi. *FFTE 7.0: A fast Fourier transform package*. 2021. URL: http://www.ffte.jp/.

[9] Aidan P. Thompson, H. Metin Aktulga, Richard Berger, et al. "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales". In: *Computer Physics Communications* 271 (2022), p. 108171. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2021.108171. URL: https://www.sciencedirect.com/science/article/pii/S0010465521002836.

[10] Alan Ayala et al. *heFFTe library*. Available at https://bitbucket.org/icl/heffte. 2023.

Bibliography III

[11] Yuwen Zhao, Fangfang Liu, Wenjing Ma, et al. “MFFT: A GPU Accelerated Highly Efficient Mixed-Precision Large-Scale FFT Framework”. In: *ACM Trans. Archit. Code Optim.* (June 2023). Just Accepted.

[12] Simon Frasch et al. *spFFT library*. Available at `https://github.com/eth-cscs/SpFFT`. 2023.

[13] Nimalan Nandapalan, Jiri Jaros, Alistair P. Rendell, et al. “Implementation of 3D FFTs Across Multiple GPUs in Shared Memory Environments”. In: *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*. 2012, pp. 167–172.

[14] Alan Ayala, Stan Tomov, Piotr Luszczek, et al. *Interim Report on Benchmarking FFT Libraries on High Performance Systems*. ICL Tech Report ICL-UT-21-03. University of Tennessee, 2021-07 2021.