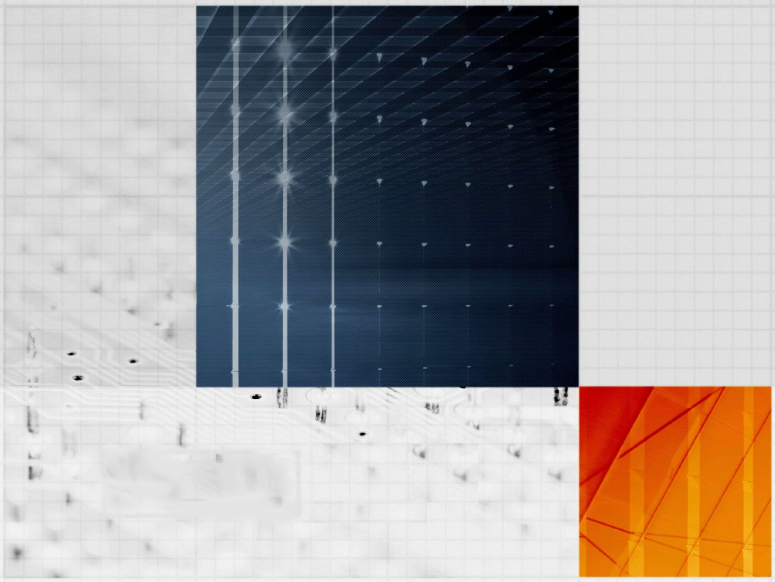

Analysis of the Communication and Computation Cost of FFT Libraries towards Exascale

Alan Ayala
Stanimire Tomov
Piotr Luszczek
Sébastien Cayrols
Gerald Ragghianti
Jack Dongarra

ICL Technical Report ICL-UT-22-07

Knoxville, July 1, 2022

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Abstract

The Exascale Computing Project (ECP) of the United States, supports the design of software and hardware towards computation at exaflops rate. Many applications of this kind, rely on efficient and scalable Fast Fourier Transform (FFT) computation. In previous work, we presented interim benchmark reports comparing almost a dozen state-of-the-art FFT libraries, analyzing their scalability and main features. Furthermore, we explored the impact of GPU accelerators from different vendors to further speedup FFT computation; as well as the communication bottleneck.

In this report, we present a detailed study of the computation and communication costs of parallel multidimensional FFT libraries. We focus on hardware with GPU accelerators, as those envisaged for modern exascale supercomputers. We present results obtained using an FFT Benchmark harness that we developed to easily benchmark and compare numerous FFT libraries on DOE exascale systems.

Contents

List of Figures

List of Tables

Acknowledgment

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early tested platforms, in support of the nation's exascale computing imperative.

Chapter 1

Introduction

In this report, we use our benchmark harness [1] to analyze the computation and communication cost of modern FFT libraries using some of world's most powerful computing architectures. We focus on 3-D FFT computation with either the 2-D decomposition approach (*pencils*) or the 1-D decomposition (*slabs*) as shows in Figure 1.1.

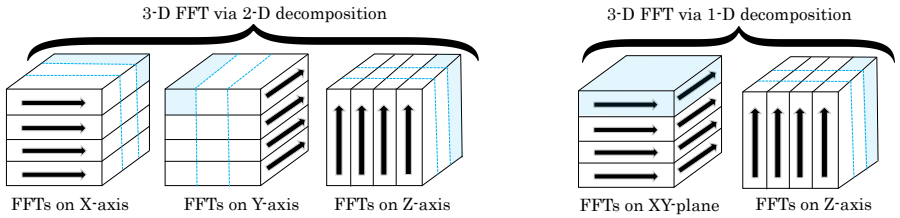

Figure 1.1: Computation of 3-D FFT via 2-D pencils decompostition (left), and 1-D slabs decomposition (right).

We base our analysis on the computation sequence described by Algorithm 1, which represents the steps that the FFT libraries used in this report follow for computing the FFT 3-D transform. Note that most state-the-art libraries only support 2-D and 3-D FFTs, while some, e.g. AccFFT [2], extend their support to 4 dimensions.

The *transfer* phase of Algorithm 1 (also known as remap, reshape or transposition) is performed using a Message Passing Interface (MPI) distribution. In [3], a novel approach, which we refer to as FFTADVMPI in Table 1.1, was introduced to perform m-dimensional FFTs with general backends using few lines of code, and MPI_Alltoallw to handle the exchanges. This novel approach avoids data packing and unpacking.

Algorithm 1 Parallel 3-D FFT computation

Input: 3-D data array, grid of processors $P \times Q$.
Define processor grids (MPI groups) for each direction
for $r \leftarrow 1, \cdots, n_{\text{batches}}$ **do**
    for $r \leftarrow 1, \cdots, n_{\text{exchanges}}$ **do**
        Compute local 1-D or 2-D FFTs on the GPUs
        Pack data in contiguous memory
        for *Process* on my MPI group **do**
            Transfer computed data to neighbor processes
        end for
        Unpack data in contiguous memory
    end for
end for

1.0.1 State-of-the-art Libraries

In this report, we consider nine state-of-the-art software libraries listed in Table 1.1.

Table 1.1: Multidimensional FFT Libraries in the FFT benchmark tests. All libraries were tested using their latest release version.

Library Name	Version	CPU support	GPU support	Slabs Decomp.	Pencils Decomp.
AccFFT	2.0	yes	yes	no	yes
2Decomp&FFT	1.5.847	yes	no	no	yes
FFTE	7.0	yes	yes	yes	yes
FFTW	3.3.8	yes	no	yes	no
FFTMPI	1.0	yes	no	no	yes
heFFTe	2.2	yes	yes	yes	yes
SWFFT	1.0	yes	no	no	yes
P3DFFT	3	yes	no	yes	yes
FFTADVMPI	N/A	yes	no	yes	yes

Amongst the most recent libraries from Table 1.1, heFFTe [4] and FFTW [5] libraries are used for error validation and their results served as the basis for comparisons with the other FFT libraries. This is, for a given input array $X \in \mathbb{R}^{m \times n \times q}$ and a parallel FFT library, we calculate the computation error as shown in Eqs. (1.1) and (1.2).

$$E_1 = \|X - \mathbf{IFFT}_{\text{heFFTe}}(\mathbf{FFT}_{\text{Library}}(X))\|_{\text{max}}, \tag{1.1}$$

$$E_2 = \|X - \mathbf{IFFT}_{\text{FFTW}}(\mathbf{FFT}_{\text{Library}}(X))\|_{\text{max}}, \tag{1.2}$$

This is, we compute an inverse transform (with heFFTe and FFTW) of the forward trans-

form obtained by the given library. For the experimental part in Section 4, we verify that the errors are less than 10^{-14}, which is roughly in the order of $\mu\|X\|_2$, being the machine precision $\mu = O(10^{-16})$.

The harness software that we prepared for testing the libraries from Table 1.1 uses hardware GPU acceleration whenever possible and allows mixed-interface to accommodate C, C++, and Fortran codes by wrapping and properly linking object files together with the required dependent runtime, system, and language. Some tests still had to be performed outside the software harness due to the cross-language and cross-platform portability issues.

Chapter 2

Experimental Setup

2.1 Description of Hardware Resources

The performance experiments in this report are obtained using two supercomputers located at the Oak Ridge National Laboratory, namely Summit and Spock. The former is an IBM AC922 supercomputer featuring IBM POWER 9 CPUs and NVIDIA Volta V100 GPUs and is currently ranked number two on the TOP500 list of the largest supercomputing installations. The latter machine is Spock, which is a precursor of the upcoming Frontier supercomputer at ORNL. Frontier is expected to overcome the 1 Exa-flop/s barrier using double-precision data. Spock is composed of nodes with AMD CPUs and GPUs. Below we list their technical specifications relevant for this report:

- Summit has a total of 4, 608 nodes. Each node consists of two sockets, each with a 22-core IBM POWER 9 CPU and 3 NVIDIA Volta V100 GPUs. The total of 6 GPU accelerators provide a theoretical double-precision peak performance of approximately 47 Tera-flop/s. Within the same socket, the computing units are connected with the NVIDIA NVLink node interconnect with a theoretical uni-directional bandwidth of 50 GB/s (or 100 GB/s bi-directional). Inter-node injection bandwidth is limited to two Mellanox Infiniband NICs totaling in uni-directional bandwidth of 25 GB/s (or 50 GB/s bi-directional).
- Spock has hardware and software in preparation for the upcoming much larger Frontier system. We use it as an early-access testbed available as part of the Exascale Computing Project (ECP) early access platforms. In total, it has a total of 36 compute nodes, each consisting of one 64-core AMD EPYC 7662 CPU equipped with to 256 GB of DDR4 memory and connected to 4 AMD MI100 GPUs. The CPU is connected to all GPUs via PCIe Gen4 with uni-directional bandwidth of 32 GB/s (or 64

GB/s of bi-directional bandwidth). The GPUs are connected in an all-to-all arrangement via the Infinity Fabric (xGMI) with uni-directional bandwidth of 46 GB/s (or 92 GB/s bi-directional bandwidth). The Spock nodes are connected with Slingshot-10 dragonfly network providing a node injection uni-directional bandwidth of 12.5 GB/s (25 GB/s bi-directional) from a single NIC card.

2.2 Description of Software Resources

For this report we use the libraries showed in Table 1.1; for their compilation and execution, we use the software listed below in Tables 2.1 and 2.2.

Table 2.1: Software versions used on Summit.

Software Module	Version Used in Tests
CUDA	11.4.2
FFTW	3.3.9
GNU compilers	9.3.0
Spectrum MPI	10.4.0.3-20210112
CMake	3.21.3
PGI	20.4
Scalasca	2.5
ScoreP	7.0
Vampir	10.0

Table 2.2: Software versions used on Spock.

Software Module	Version Used in Tests
ROCM	4.5.0
FFTW	3.3.9
GNU compilers	9.3.0
Cray-MPICH	8.1.12
CMake	3.21.3

2.3 Data Inputs and Outputs

We performed a variety of experiments to benchmark the nine state-of-the-art parallel FFT libraries mentioned earlier with only three of them having support for GPU accelerators to offload computation. Specifically, we used the following setup:

- We consider that input data is in pencil or slab decomposition, ready to start computing the batches of 1-D FFTs, c.f. Figure 1.1. Output is arranged along the slow dimension.
- For our analysis of computation cost, we employ a 3-D FFT transforms of size ranging from 128^3 to 1024^3.
- In order to maintain statistical rigor, we report the average time of ten consecutive runs of single forward 3-D FFT transforms and use that value on the graphs directly or convert them into Gflop/s rate using the well-known FFT complexity: $5N\log_2(N)$, where N is the FFT size.
- For a fair comparison, we employed our benchmark harness [1], which ensures that all libraries used the same input and output data as well as MPI process grids or to perform the same type of communication exchange: either slabs or pencils. Also for fairness purposes, we disabled the tuning phases in case it was enabled by default in any given library: this was done to report out-of-the-box experimental results as some of the libraries do not include tuning as one of the phases included in their functional scope.

Chapter 3

Analysis of FFT Communication cost

In our previous work [6, Appendix], we profiled the tensor transposition (lines 6 to 10 from Algorithm 1) for different study cases. We observed how the dominance of MPI cost in runtime, around 90%, slows down the computation for both collective and binary exchanges. In this section we analyze the communication cost from a theoretical point-of-view and then with experiments on different architectures.

3.1 Communication models for FFTs

The Cooley-Tukey algorithm for FFT computation gives us the asymptotic complexity of $O(N \cdot \log(N))$ for an m-dimensional FFT of size N on a single-device. In parallel systems, however, it is the cost of MPI communication that dominates the runtime [2, 7]. There is a vast literature on performance models for FFTs when using Algorithm 1 with Π processes on n computing nodes. And since such models are architecture dependent, there is no a single one that can accurately hold for all supercomputers. Amongst the most relevant models:

- In [2], authors propose to use $O\left(\frac{N}{\sigma(P)}\right)$, where $\sigma(P)$ is the bisection bandwidth of the network.
- In [8], authors use regression to find γ such that the communication cost is $O(n^{-\gamma})$. This was developed for a Cray XC40 system (Shaheen II).
- In [9], authors propose a theoretical lower-bound for the communication cost on 3-D FFT computations towards exascale computing systems, assuming a 3-D torus topology (which is found in supercomputers targeting exascale such as Fugaku at Riken-

Japan). The communication time is given as $\Omega\left(\frac{N}{P^{5/6} \cdot B}\right)$, where B is the network bandwidth.

3.2 Bandwidth Analysis

In Figure 3.1, we compare the practical inter-node peak bandwidth we obtained on either Summit or Spock. We used the same configuration of the Network Interface Cards (NIC) on both systems. Note that the inter-node bandwidth achieved on both systems gets close to the peak. Summit has two NICs per node and thus gets twice the overall bandwidth of Spock that was only equipped with a single network card.

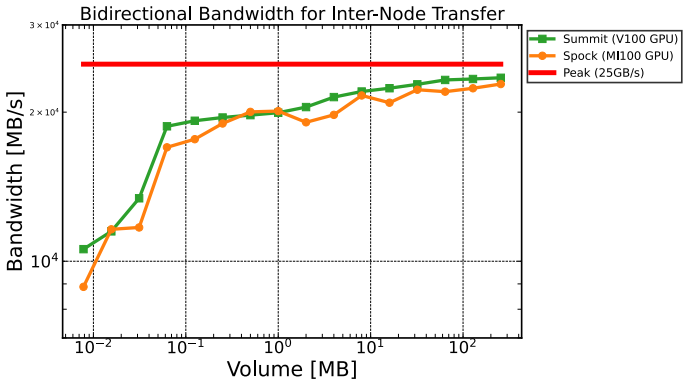

Figure 3.1: Evolution of bandwidth achieved on Summit and Spock of inter-node communication when increasing the communication volume and using one NIC. Summit has a theoretical peak of 50 GB/s (when using 2 NICs per node) while Spock's peak bandwidth is 25GB/s (using one NIC).

[3] Refer to [10] for a similar for 3-D real-to-complex model designed for Intel Xeon Phi Clusters.

Chapter 4

FFT Benchmark Results

4.1 Close to the peak performance

On systems that support GPU accelerators, it is typically the vendor implementations that offer the fastest computation, such is the case of cuFFT on NVIDIA GPU devices or rocFFT for AMD devices. When using multi-GPU devices, e.g. a DGX system from NVIDIA, it is also the vendor implementation, in this case cuFFTMp [11], which claims to achieve excelent scalability up to PetaFlop scale, and more than 70% of the peak machine. In a hybrid CPU-GPU system, in general there is no a single library that can claim the title of the fastest, instead, it highly depends on how they handle the MPI communication and their portability (e.g., cuFFTMp is dependent on NVSHMEM, which restricts its usage to specific systems).

4.2 Comparison of Strong Scalability Results

4.2.1 CPU-based libraries

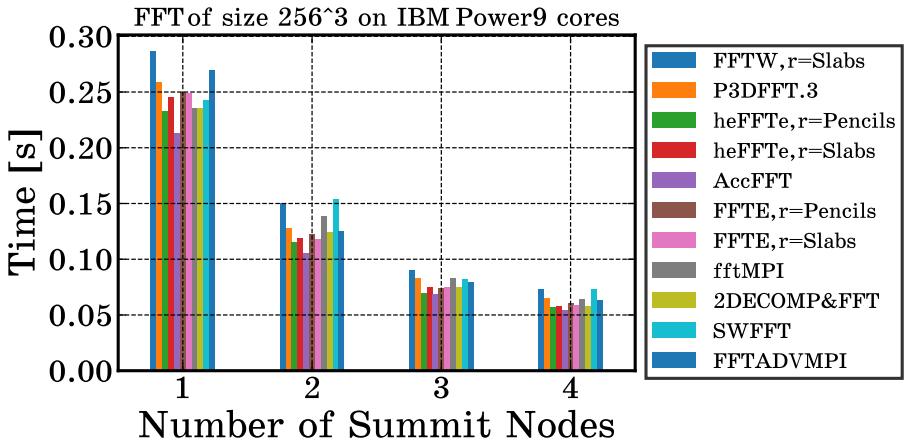

Figure 4.1: Comparison of parallel FFT libraries on up to 4 Summit nodes, using 4 MPIs per node and 1 MPI per single core of IBM POWER 9.

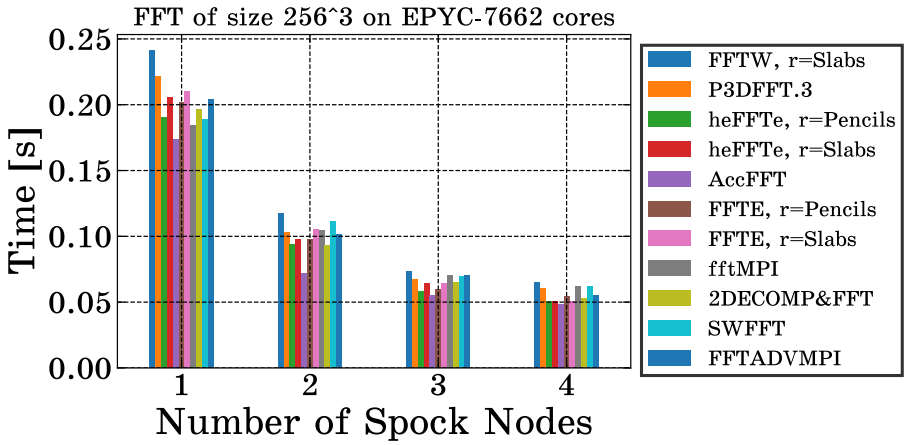

Figure 4.2: Comparison of parallel FFT libraries on up to 4 Spock nodes, using 4 MPIs per node and 1 MPI per EPYC-7662 core.

Figures 4.1 and 4.2 show the strong scalability timing of 256^3 FFTs on the multicore CPUs of Summit and Spock, respectively. These results feature all seven software libraries considered

for this report and hence these figures offer the most comprehensive comparison between them.

4.2.2 GPU-based libraries

Figure 4.3 shows the strong scaling timings of 256^3 FFTs on the GPUs of Summit spread among up to 4 nodes. Note that heFFTe, AccFFT, and FFTE are the only libraries compared in the figure as they are the only ones that provide support for distributed-memory systems with NVIDIA GPUs. Finally, heFFTe is the only library that provides support for distributed-memory systems with AMD GPUs. Therefore, we show in Section 5.1 performance results on Spock exclusively featuring the heFFTe library.

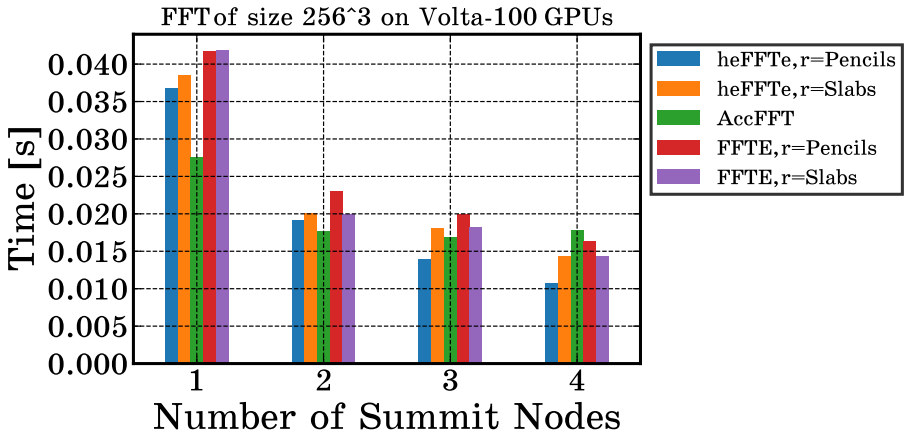

Figure 4.3: Comparison of parallel FFT libraries with GPU support on up to 4 Summit nodes, using 4 MPIs per node and 1 GPU per MPI.

Chapter 5

Discussion

In this chapter, we analyze different scenarios that can play a fundamental role in tuning FFT software implementations to adapt them to run efficiently on exascale hardware components.

5.1 Impact of Vendors' GPUs and Software

At the time of this writing, heFFTe is the only library that provides multidimensional FFTs for distributed-memory systems with AMD and Intel GPUs; therefore, it is not currently possible to compare hybrid FFT performance on Spock-like systems.

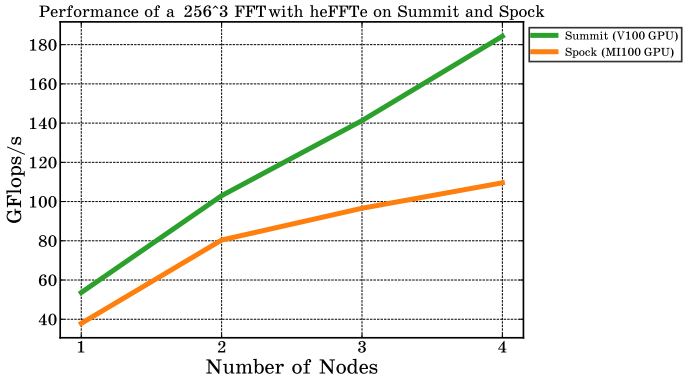

Figure 5.1: Performance comparison of a 256^3 FFT using heFFTe with vendor 1-D FFT libraries (NVIDIA and AMD) on Summit and Spock. The number of GPUs used per node is four.

Figure 5.1 shows the strong scalability timing of 256^3 FFTs of heFFTe on the GPUs of Summit in comparison with Spock. The performance shown is for heFFTe's cuFFT backend on Summit and heFFTe's rocFFT backend on Spock. This figure sheds light on the optimizations required for systems based on AMD GPUS with a single NIC configuration:

- The scaling deteriorates past 2 nodes indicating excessive increase of demand for communication bandwidth capacity as the all-to-all exchanges dominate quickly the completion time.
- The scaling dip may most likely be alleviated by additional network resources such as extra NIC cards, which are present on the Summit system for a higher injection bandwidth of the off-node traffic.
- Vendors such as AMD, Intel and NVIDIA need to further accelerate the inter-node GPU transfers. Efforts such as NCLL [12], would help with this task and must get a considerable attention towards exascale.

5.2 Impact of the libraries setup time

Most applications, such as LAMMPS [13], require several FFTs to compute calculations such as the energy of a system. When the same data structure is going to be used many times, the time for setting up the transform may not be relevant; however, when single or few transformations are needed then it may be relevant to analyze how much time a given library takes for planning the FFT. In Figure 5.2, we show how different setup timing varies amongst libraries, AccFFT being the slowest one almost 3× worse than the fastest one (heFFTe).

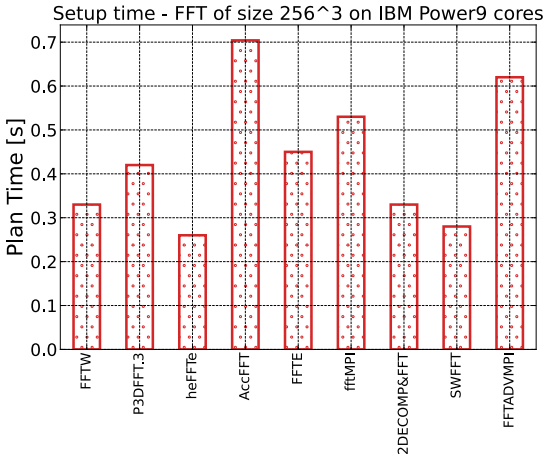

Figure 5.2: Comparison of the average setup (FFT planning) time for all libraries from Figure 4.1 at 4 nodes.

5.3 The FFT communication bottleneck

In the Fig. 5.3, we show a breakdown of the kernel components within a 3-D FFT running on multi-core CPUs for five of the libraries shown in Fig. 4.1. We can clearly observe a similar behavior for all the implementations, and note that optimizations of MPI_Alltoallw could potentially yield to getting much faster FFTs via the approach of the FFT library with advance MPI routines [3], which moves the packing and unpacking kernels into the network. This, however, is at an early stage, since MPI_Alltoallw is far from being optimized in comparison to MPI_Alltoall or MPI_Alltoallv, and in distributions such as SpectrumMPI, it is not even GPU-aware [14].

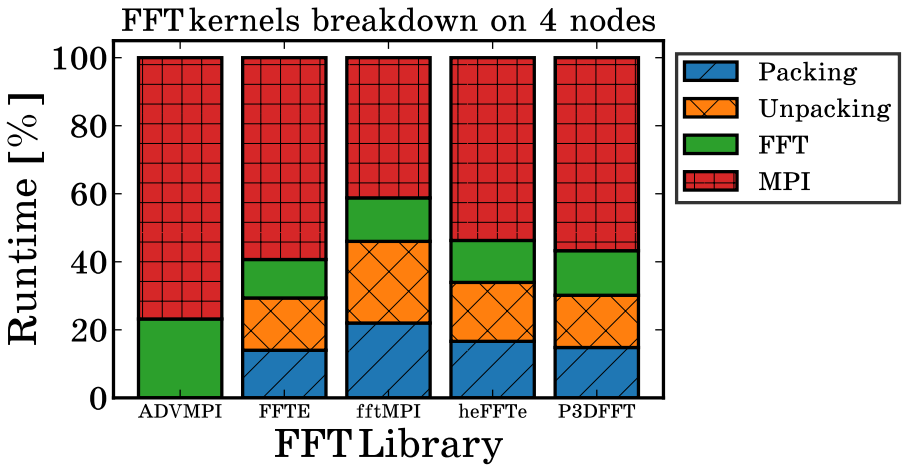

Figure 5.3: Comparison of setup (planning) time for different libraries. Note that FFTADVMPI is only composed of two kernels, since the packing and unpacking is embedded into MPI_Alltoallw.

Chapter 6

Conclusions

This report introduced an FFT Benchmark harness that was developed to easily benchmark and compare numerous FFT libraries on DOE exascale systems. The harness is open source [1] and facilitates the addition of other FFT libraries and FFT benchmarking problems from FFT as well as application developers and users.

The FFT benchmark harness was used to benchmark and compare the FFT libraries from [15], by adding the Spock ECP system at ORNL that features AMD MI100 GPUs. Spock is an early-access system and precursor to Frontier, so only 4 nodes were available for now to run.

Results show that all libraries can be easily ported and run on the multicore CPU parts of these two systems, while support for GPUs is provided only in the heFFTe, AccFFT, and FFTE libraries for Nvidia GPUs, and only heFFTe supports AMD GPUs.

Single GPU 3-D FFTs on AMD GPUs need further optimizations. Currently, rocFFT on MI100 is up to 6× slower than cuFFT on V100 GPUs. For 512^3 FFTs cuFFT reaches 12.3% of the V100 GPU compute peak vs. rocFFT reaches 2.1% of the MI100 GPU compute peak. Regardless of that, the 3-D FFTs are communication-bound and therefore the local (single GPU) performance is less important in the distributed computing setting. Indeed, Summit has about 2× faster inter-node communication bandwidth, and our benchmark results quantify that this translates to about 2× faster 3-D FFTs. Frontier is expected to have the same inter-node communication bandwidth as Summit.

Bibliography

[1] "HPFFT: A Parallel FFT Benchmark Harness." 2022. [Online]. Available: https://github.com/icl-utk-edu/fiber

[2] A. Gholami, J. Hill, D. Malhotra, and G. Biros, "Accfft: A library for distributed-memory FFT on CPU and GPU architectures," *CoRR*, vol. abs/1506.07933, 2015.

[3] L. Dalcin, M. Mortensen, and D. E. Keyes, "Fast parallel multidimensional FFT using advanced MPI," *Journal of Parallel and Distributed Computing*, vol. 128, pp. 137–150, 2019.

[4] A. Ayala, S. Tomov, A. Haidar, and J. Dongarra, "heFFTe: Highly Efficient FFT for Exascale," in *ICCS 2020. Lecture Notes in Computer Science*, 2020.

[5] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

[6] A. Ayala, S. Tomov, P. Luszczek, S. Cayrols, G. Ragghianti, and J. Dongarra, "Fft benchmark performance experiments on systems targeting exascale," Tech. Rep. ICL-UT-22-02, 2022-03 2022.

[7] "heFFTe library," 2021, available at https://bitbucket.org/icl/heffte.

[8] A. G. Chatterjee, M. K. Verma, A. Kumar, R. Samtaney, B. Hadri, and R. Khurram, "Scaling of a Fast Fourier Transform and a pseudo-spectral fluid solver up to 196608 cores," *J. Parallel Distributed Comput.*, vol. 113, pp. 77–91, 2018.

[9] K. Czechowski, C. McClanahan, C. Battaglino, K. Iyer, P.-K. Yeung, and R. Vuduc, "On the communication complexity of 3D FFTs and its implications for exascale," 06 2012.

[10] D. Takahashi, "Implementation of Parallel 3-D Real FFT with 2-D decomposition on Intel Xeon Phi Clusters,," in *13th International conference on parallel processing and applied mathematics.*, 2019.

[11] "cuFFTMp library," 2022. [Online]. Available: https://developer.nvidia.com/blog/multinode-multi-gpu-using-nvidia-cufftmp-ffts-at-scale/

[12] NVIDIA, "NCCL library," 2019. [Online]. Available: https://github.com/NVIDIA/nccl

[13] "Large-scale atomic/molecular massively parallel simulator," 2018, available at https://lammps.sandia.gov/.

[14] "Release notes on IBM SpectrumMPI 10.4," 2021, Available at https://www.ibm.com/docs/en/smpi/10.4?topic=release-notes.

[15] A. Ayala, S. Tomov, P. Luszczek, S. Cayrols, G. Ragghianti, and J. Dongarra, "Interim report on benchmarking FFT libraries on high performance systems," Innovative Computing Laboratory, University of Tennessee, Tech. Rep. ICL-UT-21-03, Jun. 2021. [Online]. Available: https://www.icl.utk.edu/publications/interim-report-benchmarking-fft-libraries-high-performance-systems

[16] D. Takahashi, "FFTE 7.0: A fast Fourier transform package," *http://www.ffte.jp/*, 2021.

[17] "parallel 2d and 3d complex ffts," 2018, available at http://www.cs.sandia.gov/~sjplimp/download.html.

[18] A. Ayala, S. Tomov, M. Stoyanov, A. Haidar, and J. Dongarra, "Accelerating Multi-Process Communication for Parallel 3-D FFT," in *2021 Workshop on Exascale MPI (ExaMPI)*, 2021, pp. 46–53.

[19] A. Ayala, S. Tomov, M. Stoyanov, and J. Dongarra, "Scalability Issues in FFT Computation," in *Parallel Computing Technologies*. Springer International Publishing, 2021.

[20] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, S. Kumar, E. Lusk, R. Thakur, and J. L. Träff, "MPI on a Million Processors," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, M. Ropo, J. Westerholm, and J. Dongarra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 20–30.

Chapter 7

Appendix: Tensor transposition cost for parallel FFTs

In this appendix section, we further analyze the communication schemes available in most state-of-the art FFT libraries. The effect that the parallel tensor transposition bottleneck has on scalability at large-scale has been extensively studied, see e.g., [15, 9].

Given the upcoming exascale ecosystem of software and hardware, we consider relevant to analyze the different MPI routines in which state-of-the-art libraries with GPU support, c.f., Table 7.1, rely to perform tensor transpositions. For the experiments below, we use heFFTe library (due to their wide range of communication options) and Vampir for visualization, c.f., Table 2.1.

Table 7.1: Available MPI routines in FFT libraries

Library	**Communication Type**[1]	
	AlltoAll	***Point-to-Point***
AccFFT [2]	MPI_Alltoall	MPI_Isend / MPI_Irecv MPI Sendrecv
FFTE [16]	MPI_Alltoall MPI_Alltoallv	-
fftMPI [17]	MPI_Alltoallv	MPI_Send / MPI_Irecv
heFFTe [4]	MPI_Alltoall MPI_Alltoallv	MPI_Send / MPI_Isend MPI_Irecv

[1] Refer to [18] for a survey of different MPI routines used in modern FFT libraries

.

7.1 Multi-core CPU based systems

Systems like Fugaku, an exascale supercomputer at the Riken Center for Computational Science in Kobe, Japan, rely on multi-core chips with high speed interconnections. In [19], authors studied FFT scalability issues in such systems. In Figure 7.1, we observe that for a small count of cores (168) the communication cost (tensor transpose) is ≈ 85% even when overlapping the packing and unpacking with the MPI exchange.

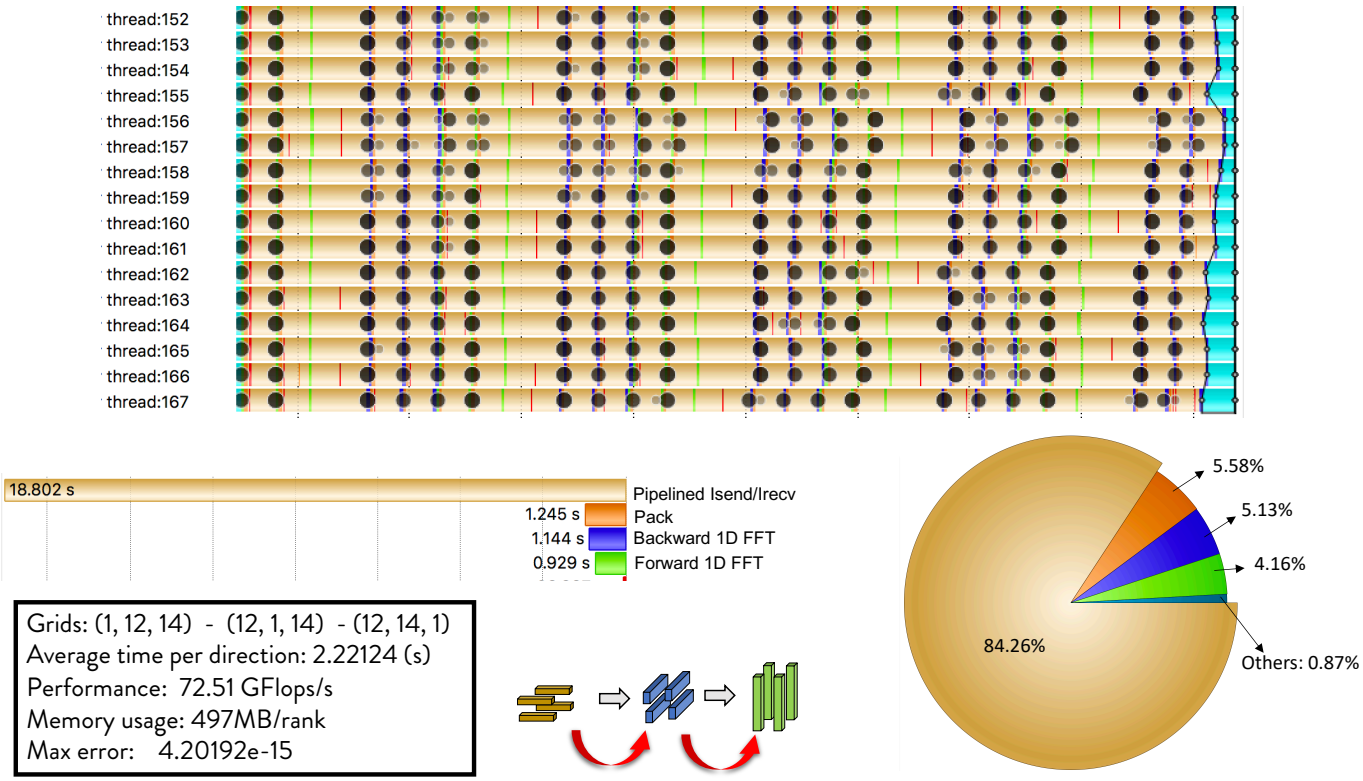

Figure 7.1: Vampir trace of back-to-back 3-D FFTs of size 1024^3 (5 forward + 5 backward), using 4 Summit nodes with 168 IBM Power9 cores, 42 MPIs per node. We use heFFTe with FFTW backend and pipelined MPI_Isend and MPI_Irecv communication.

Next, we present an experiment using MPI_Alltoall for the tensor transposition on 168 CPU cores. Libraries that support this type of communication require to have a padding step before calling the MPI routine. In Fig. 7.4, we can observe that the pack and unpack kernels take a considerable amount of time (around 40%). In Fig. 7.4 we will see that when moving those kernels into the GPUs could reduce their cost to under 7%. In Section 5.3 we also commented on how moving these kernels into the network could help accelerate the FFT computation.

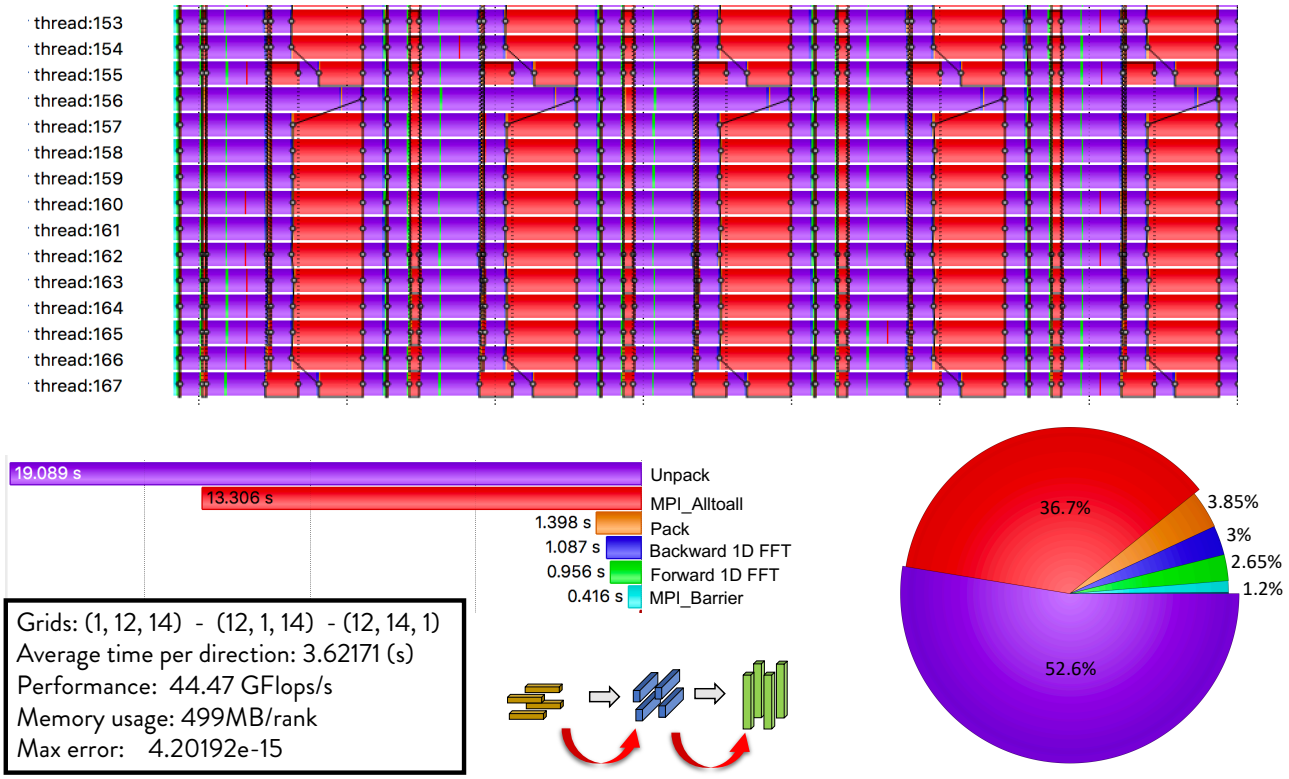

Figure 7.2: Vampir trace of back-to-back 3-D FFTs of size 1024^3 (5 forward + 5 backward), using 4 Summit nodes with 168 IBM Power9 cores, 42 MPIs per node. We use heFFTe with FFTW backend and MPI_Alltoall communication.

7.2 GPU based systems

Systems like Summit, and the upcoming Frontier, rely on hybrid CPU-GPU systems. The accelerators have proven to considerable speedup the computation of local kernels, such as the batched 1-D FFTs, packing and unpacking, c.f., [4, 2, 16]. In Figures 7.3and 7.4 we observe that the non-blocking point-to-point exchange yields to faster computation than the all-to-all approach; however, in both cases MPI covers around 90% of the runtime, and this was achieved only with 16 GPUs, which is just 0.07% of the total number of GPUs available in Summit.

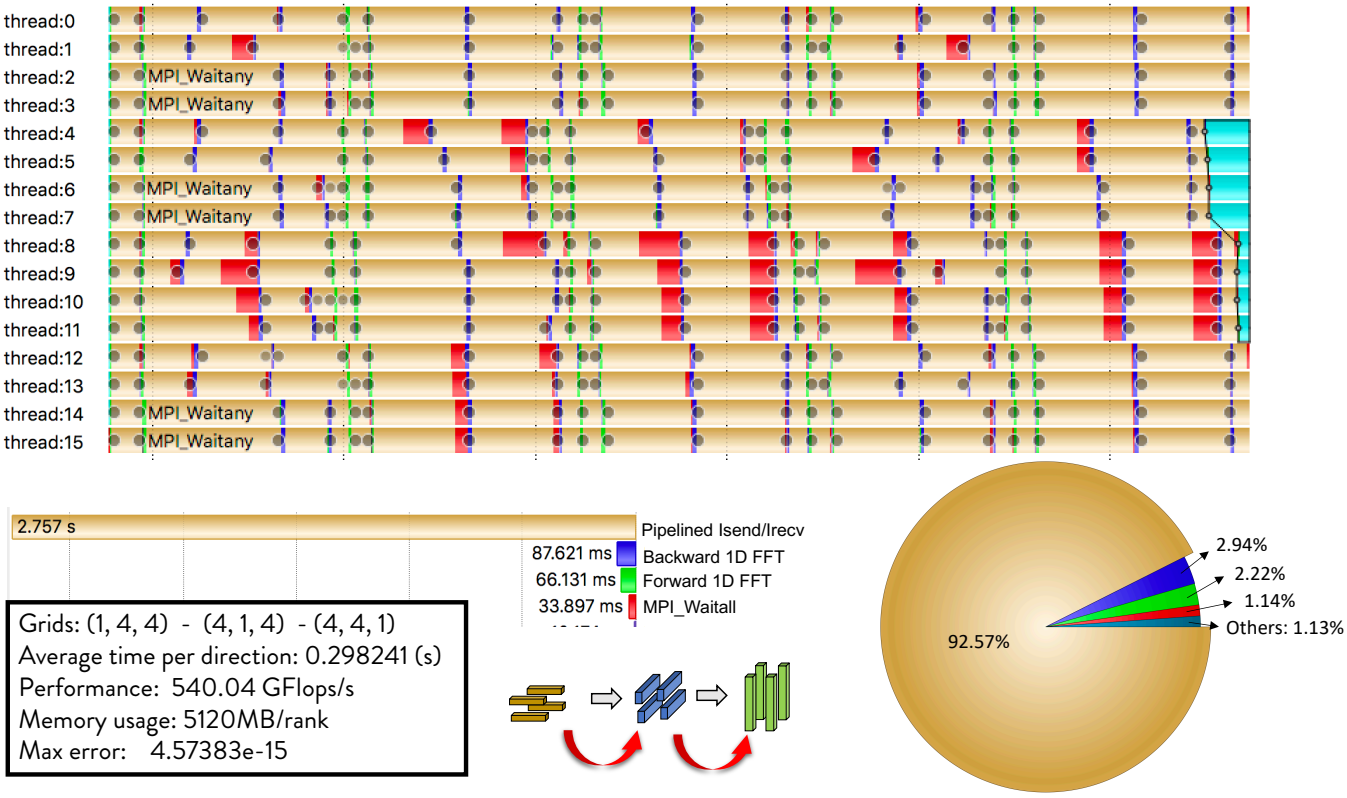

Figure 7.3: Vampir trace of back-to-back 3-D FFTs of size 1024^3 (5 forward + 5 backward), using 4 Summit nodes with 16 NVIDIA GPUs, 4 MPIs per node. We use heFFTe with CUFFT backend and pipelined MPI_Isend and MPI_Irecv communication.

Figure 7.4 also shows how the sub-communicators were created for the all-to-all transfers (see the black polygons). All libraries from Table 1.1, set the MPI_groups and communicators during the FFT planning, trying to ensure good load-balancing. In [15], it was shown how tuning grids can ensure linear scaling to over forty thousand processes, and how a bad choice of grids could lead to scalability failure. And even though tuning FFT parameters helps to achieve better performance on several thousand of processes; our experiments also suggest that at very large processes count (e.g., millions), it is the associated latency of MPI_Alltoall(v) what will produce scaling failures, refer to [20] for an external analysis. Therefore, efforts to further optimize all-to-all MPI routines, mainly for small data-volume exchanges, are critical and will have a great impact on the performance of FFT libraries at the exascale level.

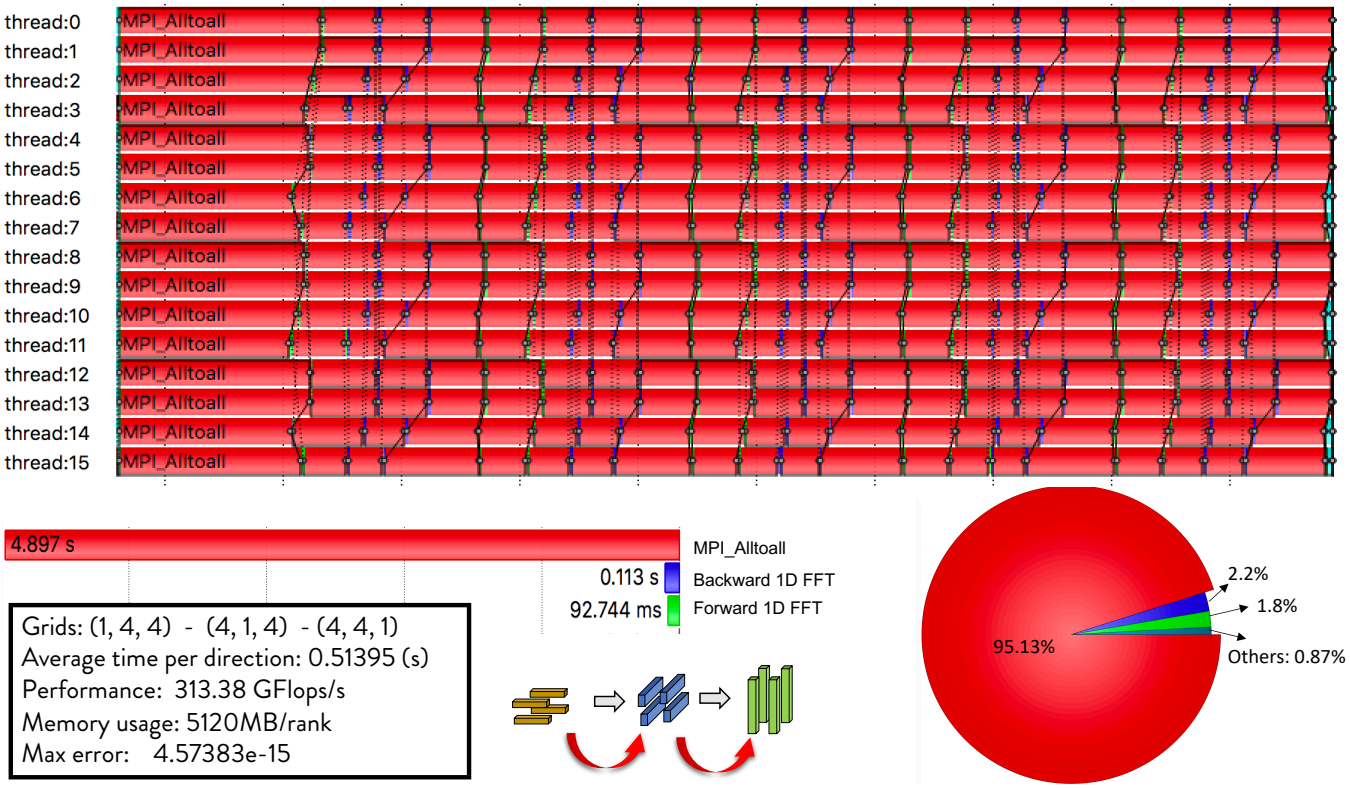

Figure 7.4: Vampir trace of back-to-back 3-D FFTs of size 1024^3 (5 forward + 5 backward), using 4 Summit nodes with 16 NVIDIA GPUs, 4 MPIs per node. We use heFFTe with CUFFT backend and MPI_Alltoall communication.