forum acusticum 2026
8-12 September · Graz, Austria

MULTI-GPU K-WAVE FOR LARGE-SCALE ULTRASOUND SIMULATION: A COMPARISON OF GLOBAL AND LOCAL FFT APPROACHES

Oliver Kunik[1], Jiri Jaros[1]

[1] *Faculty of Information Technology, Brno University of Technology, Czech Republic*

This paper is a revised version of the authors' submitted manuscript that was peer-reviewed by at least two anonymous reviewers.

Abstract

k-Wave is widely used for time-domain ultrasound simulation, but large three-dimensional problems often exceed the memory and performance limits of a single GPU. Multi-GPU implementations are therefore required to enable larger computational domains while reducing simulation time.

This work compares two multi-GPU implementations of the Fourier collocation scheme employed in k-Wave. The first preserves the original pseudospectral formulation using distributed Global FFTs over the entire computational domain, while the second replaces them with Local FFTs evaluated on overlapping subdomains connected by halo exchange. Both implementations were evaluated on a single node equipped with eight NVIDIA A100 GPUs interconnected by NVSwitch.

The results show that both approaches enable substantially larger simulations than a single GPU while providing good strong scaling. By replacing global all-to-all communication with nearest-neighbor halo exchange, the Local FFT implementation reduces inter-GPU communication by approximately one order of magnitude and achieves lower runtime for most tested domain sizes. The resulting approximation maintains relative errors on the order of 10^{-3} for appropriately chosen halo widths, providing a practical trade-off between numerical accuracy and computational performance.

Keywords: *k-Wave, FFT, multi-GPU, domain decomposition, CUDA*

1. Introduction

The increasing complexity of ultrasound simulations has created a growing demand for numerical models capable of resolving large three-dimensional computational domains while maintaining high spatial accuracy. Applications such as ultrasound imaging, therapy planning, and transcranial focused ultrasound often require computational grids containing hundreds of millions of grid points, making efficient multi-GPU implementations essential for reducing simulation time and overcoming the memory limitations of a single accelerator.

Among the available numerical approaches for acoustic wave propagation, the pseudospectral k-Wave toolbox has become one of the most widely used simulation frameworks. A recent benchmark study comparing compressional-wave models for transcranial ultrasound identified k-Wave as one of the reference solvers owing to its combination of numerical accuracy and computational efficiency [1]. Nevertheless, its pseudospectral formulation requires repeated multidimensional Fast Fourier Transforms (FFTs), which dominate the computational cost and become the primary scalability bottleneck on multi-GPU systems.

k-Wave solves a system of three coupled first-order acoustic equations using a Fourier collocation method for spatial derivatives and a k-space corrected finite-difference scheme for temporal integration [2], [3]. In three-dimensional simulations, each time step requires fourteen FFTs, making efficient parallel FFT execution critical for overall performance. While distributed FFT libraries enable scaling across multiple GPUs, they require repeated global data redistributions whose communication cost grows rapidly with increasing problem size.

One approach to reducing this communication overhead is Local FFT domain decomposition, where the computational domain is partitioned into overlapping subdomains and spatial derivatives are evaluated independently on each GPU [4], [5]. Instead of global all-to-all communication, only neighboring subdomains exchange halo regions, substantially reducing communication volume at the cost of introducing a controlled approximation of the original pseudospectral operator.

This paper compares two multi-GPU implementations of the Fourier collocation method used in k-Wave.

***Corresponding author**: ikunik@fit.vut.cz.*

The first preserves the original global pseudospectral formulation using distributed FFTs, while the second employs Local FFTs with halo exchange. Both approaches are evaluated on an eight-GPU NVIDIA A100 system in terms of numerical accuracy, communication overhead, and execution time. To isolate differences between the numerical approaches, all benchmarks are performed for a homogeneous linear acoustic medium, providing a controlled reference for comparing the two implementations.

2. k-Wave Numerical Model

The k-Wave toolbox solves acoustic wave propagation using a first-order formulation consisting of the conservation of momentum, conservation of mass, and a pressure–density relation [2], [3], [6]. Compared with the classical second-order wave equation, this formulation naturally supports heterogeneous media, nonlinear propagation, and frequency-dependent acoustic absorption.

The governing equations are

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho_0}\nabla p, \tag{1}$$

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho_0 \mathbf{u}), \tag{2}$$

together with the pressure–density relation

$$p = c_0^2 \left(\rho + \mathbf{d} \cdot \nabla \rho_0 + \frac{B/A}{2\rho_0}\rho^2 - L\rho \right), \tag{3}$$

where $\mathbf{u}$ is the particle velocity, p the acoustic pressure, ρ the acoustic density perturbation, ρ_0 the ambient density, and c_0 the small-signal sound speed. The remaining terms account for medium heterogeneity, cumulative nonlinearity, and power-law acoustic absorption [3]. The attenuation follows the empirical power-law model

$$\alpha(f) = \alpha_0 f^y, \tag{4}$$

where α_0 is the absorption coefficient and y is the power-law exponent.

Spatial derivatives are computed using a Fourier collocation method,

$$\frac{\partial q}{\partial x} = \mathcal{F}^{-1}\left\{ ik_x \mathcal{F}\{q\} \right\}, \tag{5}$$

which provides spectral spatial accuracy while reducing numerical dispersion compared with conventional finite-difference schemes [2]. Time integration employs a k-space corrected finite-difference method,

$$\kappa = \operatorname{sinc}\left(\frac{c_0 k \Delta t}{2} \right), \tag{6}$$

which is exact for homogeneous lossless media and significantly improves accuracy for heterogeneous simulations [3].

Since the Fourier collocation method assumes periodic boundary conditions, artificial wave reflections would occur when waves leave the computational domain. To suppress these nonphysical reflections, k-Wave employs an anisotropic perfectly matched layer (PML), which gradually attenuates outgoing waves before they reach the domain boundaries while introducing minimal numerical reflection [2], [3]. In the Local FFT implementation described in Section 4, an additional PML is applied independently to each extended subdomain because every local Fourier transform also assumes periodicity within its own computational domain.

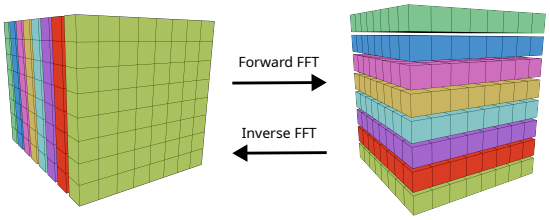

Figure 1: Data layout and redistribution in the Global FFT implementation. The initial slab decomposition used in the spatial domain is transformed into a different decomposition during distributed FFT evaluation. Each colour represents data stored on one GPU.

3. Global FFT Implementation

The **Global FFT** implementation serves as the reference solution because it preserves the original k-Wave pseudospectral algorithm while extending it to multiple GPUs. Spatial derivatives are evaluated using the global Fourier representation of the entire computational domain, ensuring numerical equivalence with the original single-GPU implementation.

The computational domain is evenly partitioned along the last spatial dimension, i.e., the z-axis in three dimensions and the y-axis in two dimensions. The primary acoustic field variables are distributed across GPUs, whereas small auxiliary arrays, such as one-dimensional PML coefficient vectors, are replicated on every device.

As illustrated in Fig. 1, the slab decomposition is well suited for storing the simulation variables but not for every stage of the multidimensional FFT. Consequently, the data must be redistributed between GPUs during the transform. In three-dimensional simulations, this corresponds to a transpose from the initial z-decomposition to a y-decomposition. These redistributions are performed transparently by the NVIDIA `cuFFTXt` library and require global all-to-all communication.

A single host thread sequentially launches CUDA kernels on all GPUs, while `cuFFTXt` performs the distributed FFTs and manages the associated inter-GPU communication.

Each three-dimensional time step requires fourteen FFTs, comprising six forward and eight inverse real-to-complex transforms. Between the transforms, multiplication by the spectral wave-number operator is

performed locally on each GPU. After the inverse transforms, the remaining field updates and PML corrections are evaluated in the spatial domain.

The principal advantage of the Global FFT implementation is that it preserves the original global pseudospectral formulation without introducing additional numerical approximations. Its primary limitation is scalability. Every distributed FFT requires global data redistribution between GPUs, and because fourteen FFTs are executed during each time step, the cumulative communication overhead increases rapidly with problem size and GPU count. Although FFT execution remains largely limited by memory bandwidth, the repeated all-to-all communication becomes an increasingly important performance bottleneck, motivating the communication-avoiding Local FFT approach described in the following section.

4. Local FFT Implementation

The **Local FFT** implementation follows the spectral domain decomposition method proposed for large-scale k-Wave simulations [4], [5]. Instead of evaluating spatial derivatives over the entire computational domain, the domain is partitioned into independent overlapping subdomains, each processed by one GPU. This replaces the global pseudospectral operator with local spectral operators, substantially reducing communication at the cost of introducing a controlled numerical approximation.

Unlike the Global FFT implementation, the Local FFT approach supports one-, two-, and three-dimensional domain decomposition. The decomposition can therefore be adapted to the number of available GPUs. In the experiments, the optimal configurations were 2×2 for four GPUs and $2 \times 2 \times 2$ for eight GPUs.

Each GPU stores its local subdomain together with surrounding halo regions, as illustrated in Fig. 2. Fourier transforms are evaluated over this extended region. Since every local FFT assumes periodic boundary conditions, each subdomain employs an independent perfectly matched layer (PML) to suppress non-physical reflections introduced by the local periodicity. The PML width was fixed to 10 grid points in all experiments. Previous work has shown that halo widths exceeding the local PML thickness significantly improve numerical accuracy [5].

The halo width also affects performance because it determines the size of the extended local FFT. Therefore, halo widths were selected to balance numerical accuracy and FFT efficiency.

As in the Global FFT implementation, each simulation step requires fourteen FFTs. However, all transforms are performed independently on each GPU using only local data. After transformation, multiplication by the spectral wave-number operator is entirely local and requires no communication.

Communication is limited to nearest-neighbour halo exchange. Instead of the repeated global redistributions required by distributed FFTs, the Local FFT

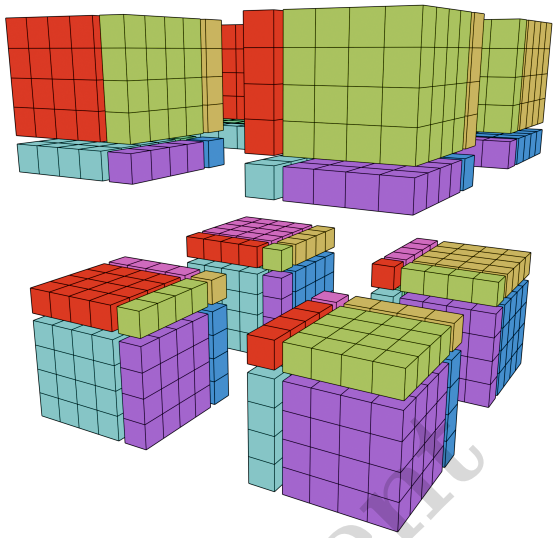

Figure 2: Domain decomposition in the Local FFT implementation. The computational domain is partitioned into overlapping subdomains assigned to individual GPUs. Each subdomain is extended by halo regions used for local FFT evaluation and data exchange.

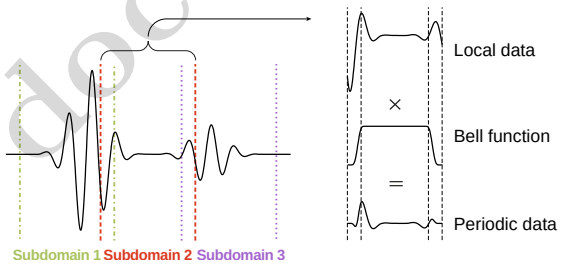

Figure 3: Bell-shaped weighting function applied before halo exchange. The weighting smoothly attenuates values towards the halo boundary, reducing discontinuities between neighbouring subdomains.

implementation performs six halo exchanges per time step: three for the particle velocity components and three for the density-related quantities. During each exchange, halo regions in all spatial directions are transferred simultaneously.

The halo exchanges are implemented using nonblocking CUDA-aware MPI, allowing data to be transferred directly between GPU memories without staging through host memory. Communication is overlapped with computation by initiating halo exchange immediately after an inverse FFT while subsequent FFT operations continue in parallel. Consequently, the implementation is suitable for both single-node and distributed multi-node systems.

Before communication, halo values are multiplied by a bell-shaped weighting function that smoothly attenuates them towards the halo boundary, as illustrated in Fig. 3 [4], [5]. The weighted values are then inserted directly into neighbouring extended subdomains, reducing discontinuities at subdomain interfaces.

The Local FFT implementation approximates the original global pseudospectral formulation because spatial derivatives are evaluated independently on local subdomains rather than over the complete computational domain. The resulting numerical error reflects the combined effects of local spectral approximation, independent PML treatment, and weighted halo exchange. While larger halo regions reduce this error, they also increase the computational cost of the local FFT. Consequently, the Local FFT approach represents a trade-off between numerical accuracy and communication efficiency, which is evaluated quantitatively in the following sections.

5. Experimental Setup

All experiments were performed on a single accelerated node of the IT4Innovations *Karolina* supercomputer equipped with eight NVIDIA A100 GPUs (40 GB each) interconnected through NVSwitch. Restricting all benchmarks to a single node isolates the impact of the communication strategy from inter-node network effects.

The Global FFT implementation uses a single host thread that controls all GPUs. In contrast, the Local FFT implementation employs one MPI process per GPU, with each process pinned to the NUMA domain closest to its assigned GPU.

The implementations were compiled using the NVIDIA HPC SDK 24.1 with CUDA 12.4 and OpenMPI 4.1.6. Distributed FFTs were performed using the NVIDIA cuFFTXt library, while local FFTs used cuFFT. CUDA-aware MPI was employed for halo exchange in the Local FFT implementation.

All simulations were performed in single precision (FP32). Both implementations used identical numerical parameters, including the computational domain, time step, PML width, arithmetic precision, and number of time steps. The reported metric is the average execution time per time step measured over 100 iterations. Initialization overheads, including memory allocation and FFT plan generation, were excluded.

All benchmarks used a homogeneous, linear, non-absorbing medium to isolate the numerical differences between the two implementations.

Only cubic computational domains were considered, although both implementations support general cuboid domains. Domain sizes were selected to favour efficient FFT execution. Since cuFFT performs best for transform sizes with small prime factors [7], the global domains were chosen accordingly. For the Local FFT implementation, the halo width was set to a minimum of eight grid points and increased when necessary so that the extended local subdomains also resulted in FFT-efficient transform sizes.

The Local FFT implementation was evaluated using 2, 4, and 8 GPUs. The computational domain was partitioned along the z-direction for two GPUs, along the y- and z-directions for four GPUs, and along all three spatial directions for eight GPUs.

Table 1: Accuracy comparison between the Global FFT and Local FFT implementations.

Domain size	Halo size	L_∞
800^3	20	9.43×10^{-4}
1008^3	8	1.22×10^{-2}
1050^3	15	1.81×10^{-3}
1200^3	25	1.23×10^{-3}
1260^3	10	6.13×10^{-3}

6. Simulation Accuracy

The numerical accuracy of the **Local FFT** implementation was evaluated by comparison with the **Global FFT** implementation, which serves as the reference solution.

An initial spherical pressure source was placed at the centre of the computational domain and propagated until the wavefront approached the outer PML boundary. The error was quantified using the relative L_∞ norm

$$L_\infty = \frac{\max |P_{\text{glob}} - P_{\text{loc}}|}{\max |P_{\text{glob}}|}, \tag{7}$$

where P_{glob} and P_{loc} denote the pressure fields computed by the Global FFT and Local FFT implementations, respectively.

Table 1 shows that the numerical error depends strongly on the halo width. Larger halos provide better agreement with the Global FFT reference solution, whereas halo widths comparable to or smaller than the local PML thickness lead to noticeably larger errors. For sufficiently large halo regions, the relative L_∞ error is consistently of the order of 10^{-3} (approximately 0.1%), which is in agreement with the behaviour reported in previous studies of Local FFT-based k-Wave simulations [5]. These results indicate that the communication savings of the Local FFT implementation can be achieved while maintaining high numerical accuracy when an appropriate halo width is selected.

7. Performance Evaluation

This section compares the Global FFT and Local FFT implementations in terms of strong scaling, execution time, and communication volume. The execution-time results are reported as the average time per simulation step in milliseconds.

7.1. Strong Scaling

Figures 4 and 5 present the strong-scaling behaviour of both implementations. As expected, larger computational domains achieve better scalability because computation increasingly dominates communication. For example, the Global FFT runtime for a 640^3 domain decreases from 129 ms on one GPU to 52 ms on four GPUs, whereas a 1024^3 domain decreases from 248 ms on three GPUs to 117 ms on eight GPUs.

The scalability of the Global FFT implementation is ultimately limited by the repeated global redistributions required by distributed FFTs. In contrast, the Lo-

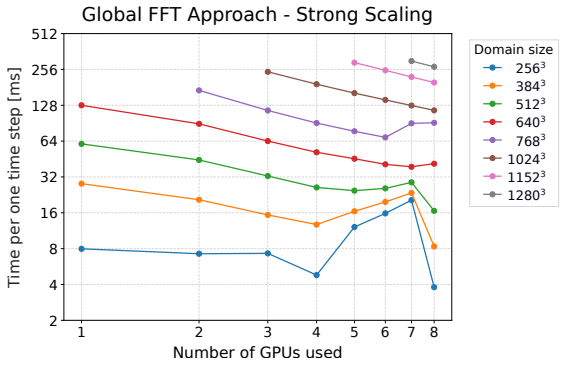

Figure 4: Strong scaling of the Global FFT implementation. The y-axis shows the average execution time per step for cubic domains from 256^3 to 1280^3.

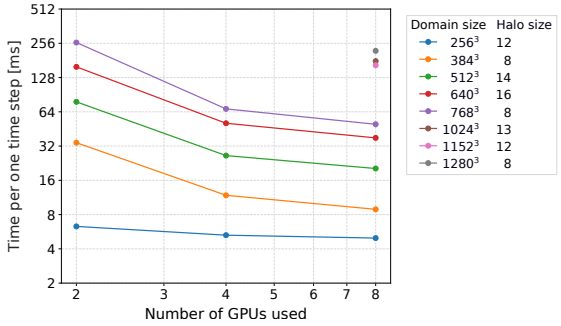

Figure 5: Strong scaling of the Local FFT implementation. The y-axis shows the average execution time per step for cubic domains from 256^3 to 1280^3.

cal FFT implementation replaces these all-to-all communications with nearest-neighbour halo exchange, resulting in better scalability for larger problems.

An exception occurs for the Local FFT implementation at the 1152^3 domain size, where the runtime is lower than for the smaller 1024^3 problem. This behaviour is caused by the efficiency of the corresponding padded local FFT sizes rather than by the global domain size itself, illustrating the importance of selecting FFT-friendly local dimensions.

7.2. Runtime Comparison on 8 GPUs

Figure 6 compares both implementations for 21 cubic domains between 800^3 and 1280^3 using eight GPUs. The Local FFT bars additionally indicate the halo width selected for each simulation.

The Local FFT implementation achieves lower execution times for 18 of the 21 tested domain sizes and nearly identical performance for two additional cases. The largest improvement is observed for the 1008^3 domain, where the average runtime decreases from 140 ms to 84 ms per step. The only notable exception is the 1024^3 case, where the distributed Global FFT benefits from the highly favourable transform size 2^{10}, while the padded local transforms are less efficient. Overall, these results demonstrate that reducing communication generally outweighs the additional computational cost associated with local FFTs.

7.3. Communication Volume

Figure 7 compares the total amount of data exchanged between GPUs during one simulation step for domain sizes 256^3, 512^3, and 1024^3.

The communication volume is substantially lower for the Local FFT implementation because global FFT redistributions are replaced by six nearest-neighbour halo exchanges. Across all tested configurations, this reduces the communicated data volume by approximately one order of magnitude, with reductions ranging from about 10× to 25×. For example, for the 1024^3 domain, the Global FFT implementation exchanges approximately 12.25 GB of data per time step, whereas the Local FFT implementation with a halo width of 13 grid points exchanges only about 500 MB.

This significant reduction in communication directly explains the runtime improvements observed in Fig. 6. The communication volumes for the Global FFT implementation were obtained from profiler measurements for representative configurations and extrapolated to the remaining domain sizes.

8. Conclusion

This paper presented and compared two multi-GPU implementations of the Fourier-collocation method used in the k-Wave toolbox: a **Global FFT** implementation based on distributed FFTs over the entire computational domain and a **Local FFT** implementation based on domain decomposition with halo exchange.

The Global FFT implementation preserves the original pseudospectral formulation and therefore provides the most faithful extension of the single-GPU k-Wave algorithm. In contrast, the Local FFT implementation replaces repeated global redistributions with local FFTs and nearest-neighbour communication, substantially reducing communication overhead. The benchmark results show that this approach achieves lower execution times for most tested problem sizes while maintaining relative errors on the order of 10^{-3} when an appropriate halo width is selected.

The presented results demonstrate that reducing communication can provide greater performance benefits than preserving the exact global spectral operator, particularly for large multi-GPU simulations. Consequently, the Global FFT implementation is preferable when maximum numerical fidelity is required, whereas the Local FFT implementation offers a more efficient alternative for large-scale simulations where a small, controllable approximation is acceptable.

Future work will focus on extending both implementations to distributed multi-node systems, where the communication advantages of the Local FFT implementation are expected to become even more significant. Additional work will investigate automatic selection of FFT-efficient local domain sizes and adaptive optimisation of halo widths to further improve the balance between numerical accuracy and computational performance.

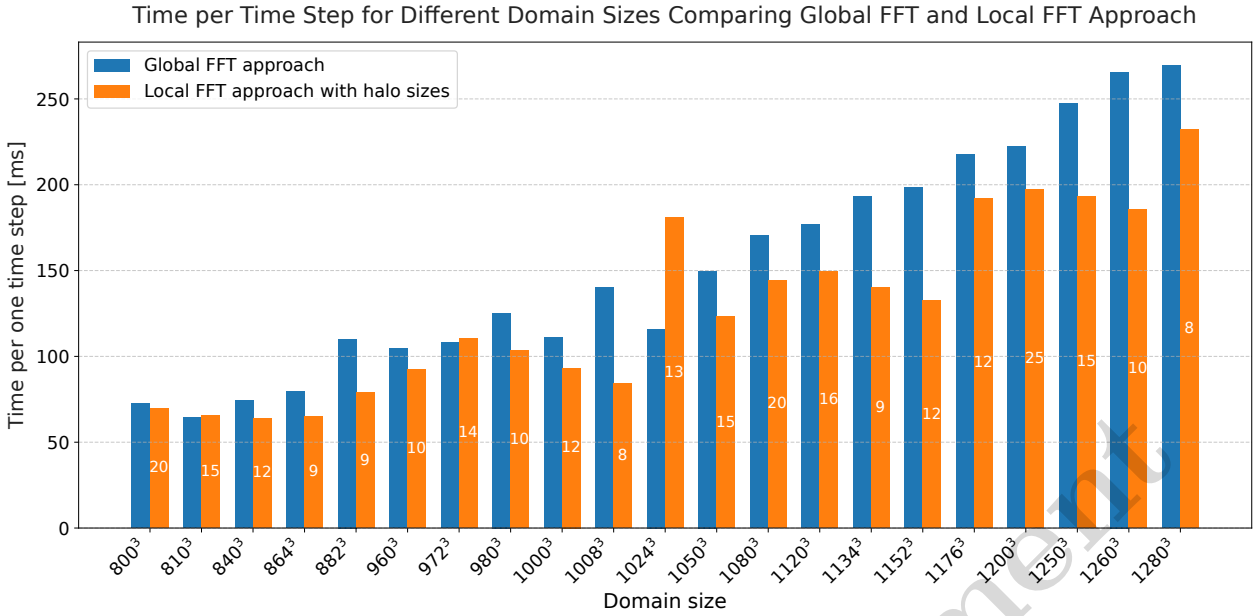

Figure 6: Runtime comparison of the Global FFT and Local FFT implementations on eight GPUs for 21 cubic domain sizes between 800^3 and 1280^3. Halo widths are shown for the Local FFT bars.

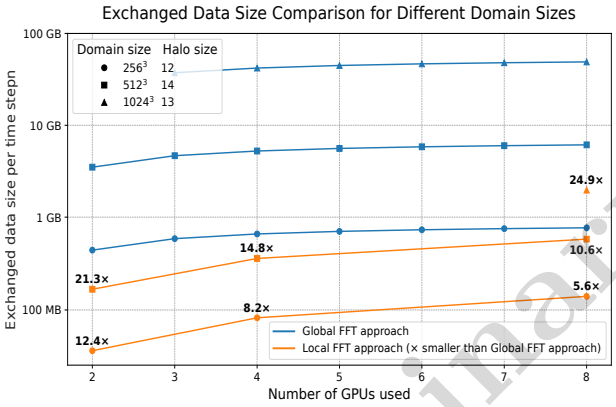

Figure 7: Total data exchanged between GPUs during one simulation step for the Global FFT and Local FFT implementations. Labels above the Local FFT points indicate the communication reduction factor.

9. Acknowledgments

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254). This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101071008. This work was supported by Brno University of Technology under project number FIT-S-23-8141.

References

[1] J.-F. Aubry, B. E. Treeby, et al., "Benchmark problems for transcranial ultrasound simulation: Intercomparison of compressional wave models," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 2022. DOI: 10.1109/TUFFC.2022.3158934.

[2] B. E. Treeby and B. T. Cox, "K-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave-fields," *Journal of Biomedical Optics*, vol. 15, no. 2, p. 021 314, 2010. DOI: 10.1117/1.3360308.

[3] B. E. Treeby, J. Jaros, A. P. Rendell, and B. T. Cox, "Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method," *The Journal of the Acoustical Society of America*, vol. 131, no. 6, pp. 4324–4336, 2012. DOI: 10.1121/1.4712021.

[4] J. Jaroš, F. Vaverka, and B. E. Treeby, "Spectral domain decomposition using local fourier basis: Application to ultrasound simulation on a cluster of gpus," *International Journal of Supercomputing Frontiers and Innovations*, vol. 3, no. 3, pp. 40–55, 2016.

[5] B. E. Treeby, F. Vaverka, and J. Jaroš, "Performance and accuracy analysis of nonlinear k-wave simulations using local domain decomposition with an 8-gpu server," *Proceedings of Meetings on Acoustics*, vol. 34, no. 1, p. 022 002, 2018. DOI: 10.1121/2.0000883.

[6] B. E. Treeby, J. Jaros, A. P. Rendell, and B. T. Cox, *K-wave user manual*, Version 1.1, k-Wave, 2016. [Online]. Available: https://www.k-wave.org/manual/k-wave_user_manual_1.1.pdf.

[7] *Cufft library user's guide*, NVIDIA, 2025. [Online]. Available: https://docs.nvidia.com/cuda/cufft/.