geoconvention
Calgary • Canada • June 17-19 2024

Challenges of 5D interpolation on GPUs using the non-uniform Discrete Fourier Transform

Author information – Kai, zhuang, Daniel Trad
Affiliation – University of Calgary CREWES research group

Summary

We implement 5D interpolation with the non-uniform DFT interpolation operator on GPUs to address binning-related errors in far offset approximations without compromising computational efficiency. This approach aims to accelerate the relatively slow naive DFT operator using GPU computation. Our findings indicate that the GPU implementation effectively resolves both issues. Although the computation of the non-uniform DFT is more expensive than the FFT, transitioning to GPU computation significantly reduces runtime compared to the standard CPU implementation of DFT interpolation.

Theory / Method / Workflow

The Fast Fourier Transform (FFT) algorithm, introduced by Cooley and Tukey in 1965 (Cooley, 1965), significantly reduces the time complexity associated with computing the Discrete Fourier Transform (DFT) of regularly sampled signals. This enhancement is particularly beneficial for processing long-period signals, leading to a substantial decrease in computational time. The time complexity of the DFT is well-known to be O(N^2), where N represents the number of data points.

In contrast, the FFT algorithm achieves a time complexity of O(N log N) through recursive decomposition and recombination of the DFT. However, it is crucial to recognize that this increased efficiency comes with the requirement that the data must be regularly sampled.

Various strategies have been developed to address this constraint and enable the FFT to operate on irregularly sampled data. One such approach is the non-uniform FFT (NUFFT), proposed by Dutt and Rokhlin in 1993. Many of these methods primarily focus on upsampling and/or interpolating the data to transform it into an evenly sampled signal before applying the FFT algorithm.

An alternative method for tackling this issue involves using an iterative approach to solve the NUFFT as a forward-adjoint problem, as discussed by Keiner et al. in 2009 Techniques for computing the NUFFT have been implemented on GPUs, showcasing varying performance levels, as demonstrated by Shih et al. in 2021. However, these approaches have drawbacks compared to the straightforward DFT method. For instance, the resampling approach may increase memory usage or introduce approximations (such as sinc, bilinear, etc.) in the interpolation case. In the inversion case, the higher computational time is incurred due to the need for iterative calculations.

For our application in 5D interpolation, we opt to solve the Non-Uniform DFT matrix, referred to as NUDFT (Non-Uniform DFT), instead of the standard FFT. NUDFT accommodates irregular spatial positioning, allowing for more accurate results, especially in wide azimuth data, thereby improving the output image quality. However, as described in (Trad,2016), 5D interpolation via NUDFT requires a quadruple nested loop for matrix calculations, resulting in significant computational times due to increased time complexity over FFT. The NUDFT operator is a candidate for GPU application due to the substantial thread advantage provided by GPU architecture. The complexity of applying NUDFT 5D interpolation on GPUs arises from multiple interpolation steps performed between each application of the Fourier transform. These steps must be optimized for GPU applications to prevent a significant algorithm slowdown caused by substantial memory transfers between GPU and CPU. This issue arises because the algorithm computes one frequency slice at a time, using the previous frequency slice to aid in the convergence of other frequencies, leading to the interleaving of operators.

Processing using NUDFT is costly because, due to unevenly spaced samples, it takes more time to calculate the extra setup needed for the unevenly spaced matrix. This challenge is akin to the computer science problem of sparse matrix-vector multiplication, where sparse matrices must account for the non-uniform way in which data is stored. Implementing 5D interpolation using a NUDFT operator bypasses the need to bin traces for a uniform grid, adding a significant processing requirement. The incorporation of the GPU kernel into the broader 5D interpolation framework introduces an additional layer of complexity to the complete 5D implementation on GPUs.

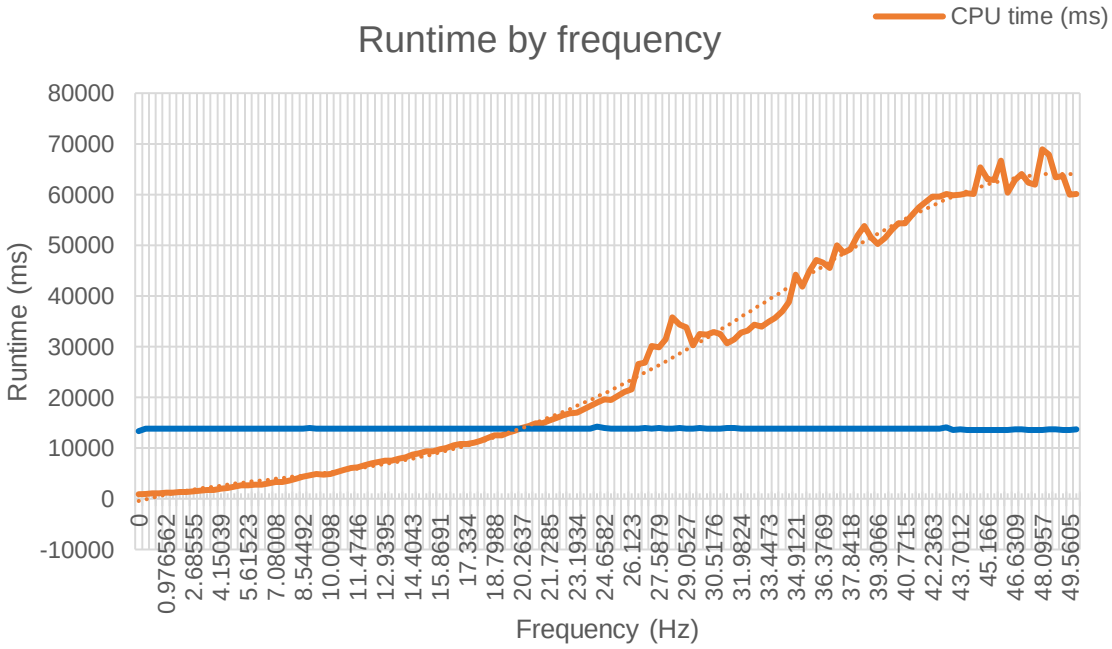
In general, CPU parallelization tends to perform better in coarse-grained applications, and this complexity increases when considering GPU applications. The preference for coarse-grained parallelization on CPUs is primarily influenced by the processor architecture, which is optimized for executing multiple independent tasks simultaneously. In this context, coarser parallelization is advantageous as CPUs do not anticipate heavy resource sharing. This stands in contrast to GPUs, designed for fine-grained applications where a significant number of resources are shared across threads. The conventional approach to parallelization for a 5D problem on a CPU involves adopting the coarsest-grained strategy, with each CPU operating on distinct trace grouped windows. However, this parallel window approach is unsuitable for GPUs due to architectural constraints, particularly the GPU's inefficiency in handling branching statements, such as if-else statements. In the GPU environment, a branching statement necessitates a device-wide wait of a significant portion of threads since only one branch can be executed at a time. This effectively diminishes a parallel process by a factor of the amount of branching statements in the GPU code.

In recent years, GPU memory sizes have experienced significant growth, alleviating the limitations imposed by the previously restrictive memory constraints. However, concerns persist regarding memory transfer bandwidth and latency across the PCI-Express bus when implementing code on the GPU. Additionally, these challenges exacerbate limitations on the applications of fine-grained parallelism on the GPU. Excessive back-and-forth transfers can lead to substantial slowdowns due to the limitations of the PCIe bus.

The incorporation of the GPU kernel into the broader 5D interpolation framework introduces an additional layer of complexity to the complete 5D implementation on GPUs. We observed the relative speedup between the GPU and CPU Discrete Fourier Transform (DFT) operator to be approximately 1:360 in various tests for our implementation at a common trace length. However, the results following the integration into the overarching 5D code reveal a substantial reduction in speedup. Specifically, the speedup observed by switching just the kernel is approximately 1:3. This outcome was deemed unsatisfactory, as this level of improvement could likely be achieved through better optimization of the CPU code.

The notable decline in speedup can be attributed to the issues mentioned earlier, primarily excessive memory transfers between the CPU and GPU during kernel execution. Due to the kernel's efficiency and the small effective size of the windows, each iteration spends minimal time executing the kernel. However, the need to transfer data back and forth for conjugate gradient calculations, for each frequency slice, contributes significantly to the diminished speedup.

Observations



Acknowledgements

We thank the sponsors of CREWES for their continued support. This work was funded by CREWES industrial sponsors and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 543578-19.

References

Cooley, J. W., and Tukey, J. W., 1965, An algorithm for the machine calculation of complex fourier series: Mathematics of Computation, 19, 297–301.
Dutt, A., and Rokhlin, V., 1993, Fast fourier transforms for nonequispaced data: SIAM Journal on Scien#tific Computing, 14, No. 6, 1368–26, copyright - Copyright] © 1993 Society for Industrial and Applied Mathematics; Last updated - 2021-09-11.
Keiner, J., Kunis, S., and Potts, D., 2009, Using nfft 3—a software library for various nonequispaced fast fourier transforms: ACM Trans. Math. Softw., 36, No. 4.
URL https://doi.org/10.1145/1555386.1555388
Shih, Y.-h., Wright, G., Andén, J., Blaschke, J., and Barnett, A. H., 2021, cufinufft: a load-balanced gpu library for general-purpose nonuniform ffts.
URL https://arxiv.org/abs/2102.08463
Trad, D., 2016, Five-dimensional interpolation: exploring different fourier operators: CREWES Annual Research Report, 28.
Trad, D., Ulrych, T., and Sacchi, M., 2003, Latest views of the sparse radon transform: GEOPHYSICS, 68, No. 1, 386–399.