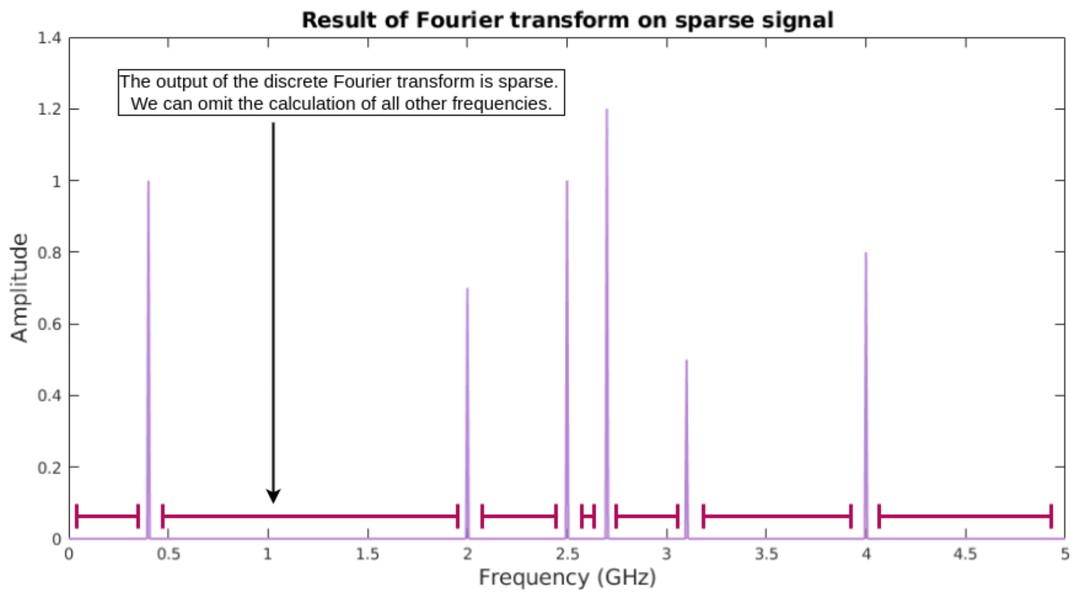
Comparison of SpFFT and FFTW3 library

Ondřej Olšák

Faculty of Information Technology, Brno University of Technology, CZ

Overview

The most popular approach for computing the discrete Fourier transform (DFT) is the Fast Fourier transform algorithm. This algorithm works well on various types of signals. However this algorithm is not fast enough to handle large signals to satisfy requirements on computation time. Most of those large signals have same common feature. They usually consists of small amount of frequencies. This bring us to the Sparse Fourier transform. This algorithm tries to separate each frequency in the sparse signal and estimate it's value.



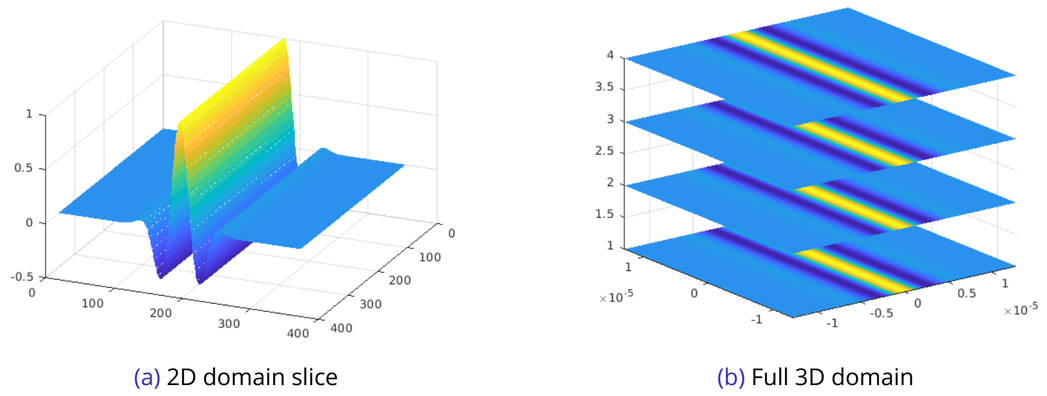
Library description

FFTW3[1] is a widely used library for the Fast Fourier transform computation. FFTW3 is able to perform one-dimensional or multidimensional transforms.

SpFFT[2] is a C++ library providing the Fast Fourier transform of sparse frequency domains of both real and complex data. This library works over 3D input data. There is support for both local computation and distributed computing by using OpenMPI and OpenMP. The library also supports computation on GPU using CUDA and ROCm. SpFFT supports computation with double or single (must be specified during library compilation) precision. In addition to sparse Fourier transform, this library has feature called *Multi-Transform*. This feature allows execute multiple independent forward transforms at once by using internal pipelining.

Benchmark data

The comparison of the SpFFT and FFTW3 libraries was performed on a set of several artificial signals. Each domain consists of *N* 2D signals [1a], where *N* is equal to the height of the domain (z-axis) [1b]. Shape of each input domain is cube. Number of elements in each signal domain are $d = 1024^3, 1088^3, 1152^3, 1216^3, 1280^3$. Figure 1 shows an example of Gaussian pulse displayed as single 3D domain slice and few layers of full 3D domain.



(a) 2D domain slice (b) Full 3D domain

Figure 1: Visualisation of signal domain

We create three types of signals:

1. 16 harmonics frequencies combined with a white Gaussian noise
2. Square wave with a period of 2π
3. Dirichlet or periodic sinc function of degree 15

Results

The figures below shows strong scaling of the FFTW3 and the SpFFT library in double and single precision on signal with 16 harmonics frequencies with the Gaussian noise. The measurement was performed on Karolina cluster. Number of MPI ranks depends on domains size and number of used computation nodes.

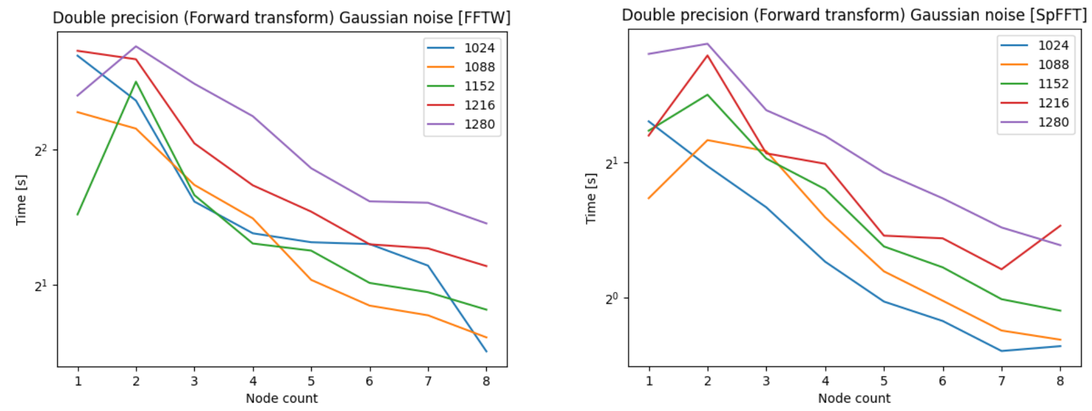


Figure 2: Strong scaling of SpFFT and FFTW3 on forward transform over signal in double precision

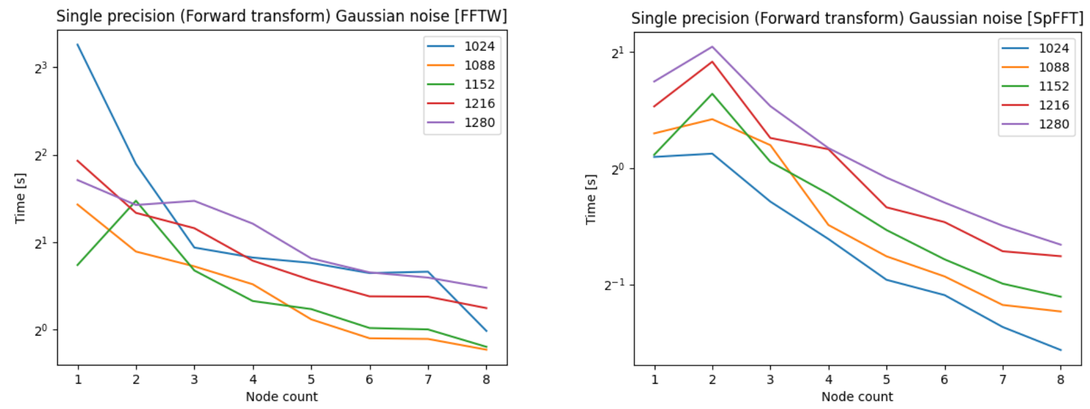


Figure 3: Strong scaling of SpFFT and FFTW3 on forward transform over signal in single precision

Next two figures shows in one plot all computation times for all mentioned input signals on domain with 1280^3 numbers for single and double precision. It is obvious, that SpFFT is on given signals faster than FFTW3.

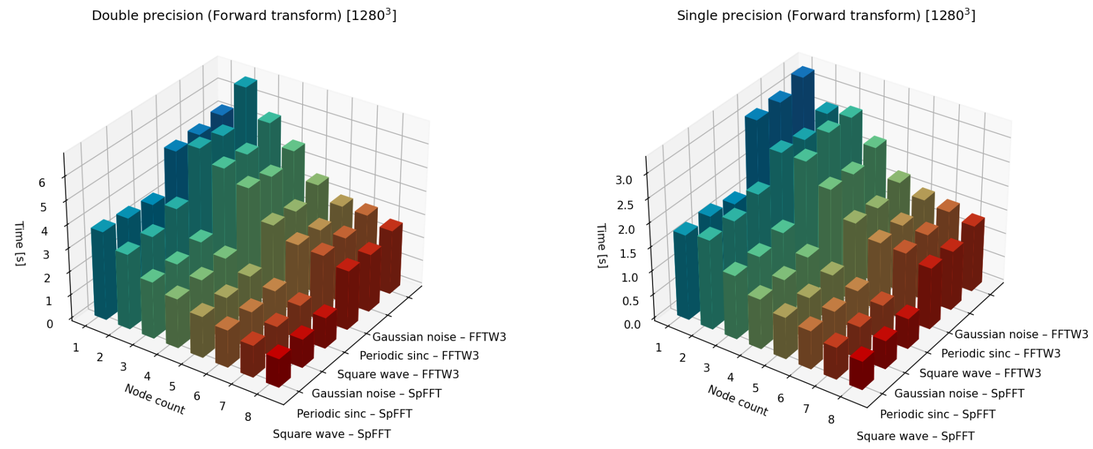


Figure 4: Computation time of SpFFT and FFTW3 on all mentioned signals.

Following table shows speedup on domain size equal to 1280^3 on each mentioned signal for given number of computation nodes.

Node count	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
MPI ranks/node	**128**	**128**	**107**	**107**	**128**	**107**	**92**	**80**
Square wave	1.72	1.37	1.92	2.06	1.89	1.78	2.10	2.16
Harmonic with Gauss	1.94	1.29	1.91	2.05	1.85	1.93	2.12	2.19
Dirichlet	1.72	1.32	2.10	2.02	1.82	1.78	2.07	2.10

Conclusion

We compared FFTW3 library for the Fast Fourier transform widely used in many application with SpFFT library on 3D domains containing different types of signals. The SpFFT library computation time was (in our case) better than computation time of the FFTW3. The reason why SpFFT was much better is probably caused by fact, that each layer of 3D domain contains same signal (except white Gaussian noise used in one input domain). For a the next measurement may be interesting to use a signal with large number of frequencies which are not similar across the domain layers.

[1] FFTW homepage: https://www.fftw.org/
[2] SpFFT documentation: https://spfft.readthedocs.io/en/latest/

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140). This work was supported by Brno University of Technology under project number FIT-S-20-6309 Design, Optimization and Evaluation of Application Specific Computer Systems.