Weekly Report

LIU Honghao

March 2021

1 Experiment

1.1 Hardware info

CPU information:
Name: Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
Thread Number: 64
Memory size: 512G
GPU info:
Name: Tesla V100-SXM2-32GB
Global memory: 32GB,**Shared memory per block:** 49KB

1.2 Results

Compare nonuniform Fourier transform with different methods. All methods are run in 10 times and getting the average values.

1.2.1 Notations

N - number of uniformly distributed points
M - number of non-uniform points (not really non-uniformly distributed)
Methods:

1. cufinufft, GPU version using global memory, using cufft for fast Fourier transform.
2. cufinufft, GPU version using shared memory, using cufft for fast Fourier transform.
3. finufft, CPU version with 64 threads (max threads) parallelized by openMP, using FFTW for fast Fourier transform.
4. finufft, CPU version with 32 threads (one thread per core) parallelized by openMP, using FFTW for fast Fourier transform.

Tol - tolerance, the maximum l^2 error for one point (defined as equ. 1) can bear. C is the exactly correct value. c_r is the result calculated by NUFFT. Both C and c_r are complex numbers.

$$err = \frac{|C - c_r|}{\max\{|c_i|\}} \tag{1}$$

MemTrans - time for transferring data between host and to device.

Plan - plan time for some preparation operations like correction factor calculation, memory allocation for plan related information.
Exce - execution time for spreading, FFT and correction.
For direction, there are two types, inverse - from NUPTS to UPTS, forward - UPTS to NUPTS.

1.2.2 Statistics

For GPU version, when the input size is relative small, the plan step accounts for the major consumed time. Memory transferring between host and device costs most time as size is large. And method2 with shared memory is the fastest in all cases and methods. The speed up of GPU is more obvious - around 6 to 10 times faster - when the input size is over 10^8. See table 1 3.
However, the CPU version - finufft - can handle larger input size of M up to 10^9 which is 10 times larger than that of GPU, due to the large memory - 512GB. See table 2.

N	**M** (NUPTS)	**Method**	**MemTrans**	**(Plan+)Exec**	**Total time**
512*512	512*512	1	0.0012	**0.0168** + 0.001	0.0195
		2	0.001	0.0169 + 0.0007	**0.0192**
		3			0.0464
		4			0.0243
1024*1024	1024*1024	1	0.005	**0.017** + 0.0037	0.0267
		2	0.004	**0.017** + 0.0026	**0.025**
		3			0.116
		4			0.063
1024*1024	10000 * 10000	1	0.277	0.0174 + 0.21	0.521
		2	0.277	0.0173 + 0.159	**0.47**
		3			1.69
		4			2.07
10000*10000	10000 * 10000	1	0.45	0.0344 + 0.41	0.964
		2	0.45	0.0332 + 0.29	**0.844**
		3			5.11
		4			4.9
10000 * 10000	10000 * 20000	1	0.753	0.646	1.57
		2	0.838	0.619	**1.34**
		3			7.11
		4			7.43
10000 * 10000	10000 * 40000	1	1.33	1.37	3.06
		2	1.28	0.781	**2.41**
		3			11
		4			12

Table 1: Running time with different methods and input size, precision: double, tolerance: 10^{-6}, direction: inverse.

When the number of non-uniform points up to $5 * 10^8$, the memory is insufficient for spreading in GPU version (may solved by Multi-GPUs with message passing if need to scale), $5 * 10^9$ for CPU. Table 2 shows running time of larger input size executing in CPU.

N	**M** (NUPTS)	**Total time**
10000*10000	$5 * 10^8$	13.1
	$7 * 10^8$	16.8
	10^9	22.8
	$5 * 10^9$	error

Table 2: Running time with different input size, tolerance is 10^{-6}.

N	**M** (NUPTS)	**Method**	**MemTrans**	**(Plan+)Exec**	**Total time**
512*512	512*512	1	0.0012	**0.0179** + 0.0005	0.0201
		2	0.001	**0.0166** + 0.0004	**0.0186**
		3			0.0367
		4			0.0239
1024*1024	1024*1024	1	0.005	**0.0167** + 0.002	0.0246
		2	0.0047	**0.0168** + 0.0016	**0.0239**
		3			0.101
		4			0.0637
1024*1024	10000 * 10000	1	**0.13**	0.017 + 0.0903	0.254
		2	**0.13**	0.0182 + 0.0841	**0.249**
		3			1.87
		4			2.42
10000*10000	10000 * 10000	1	**0.45**	0.033 + 0.255	0.808
		2	**0.45**	0.033 + 0.195	**0.749**
		3			4.78
		4			4.56
10000 * 10000	10000 * 20000	1	**0.487**	0.307	0.96
		2	**0.487**	0.279	**0.931**
		3			6.91
		4			7.41
10000 * 10000	10000 * 40000	1	**0.645**	0.498	1.43
		2	**0.645**	0.449	**1.38**
		3			11.2
		4			12.6

Table 3: Running time with different methods and input size, the precision is double, tolerance is 10^{-6}, direction is forward.

Regarding tolerance, both methods can achieve error just 10^{-12} for each point in 2D cases. When the input sizes of N and M are 10^8 and precision is double, speed of GPU is around 6 to 10 times faster than that of CPU in different tolerance. See table 4.

Tol	Method	Direction	Total time
10^{-1}	1	inverse	0.755
	2		**0.75**
	3		4.66
	4		4.15
	1	forward	0.777
	2		**0.743**
	3		4.24
	4		3.84
10^{-3}	1	inverse	0.799
	2		**0.766**
	3		4.7
	4		4.32
	1	forward	0.786
	2		**0.744**
	3		4.29
	4		3.9
10^{-6}	1	inverse	0.963
	2		**0.843**
	3		5
	4		4.8

Tol	Method	Direction	Total time
10^{-6}	1	forward	0.809
	2		**0.749**
	3		4.72
	4		4.69
10^{-9}	1	inverse	1.21
	2		**1**
	3		5.63
	4		5.93
	1	forward	0.843
	2		**0.792**
	3		5.55
	4		5.94
10^{-12}	1	inverse	1.54
	2		**1.34**
	3		14.1
	4		10.9
	1	forward	0.918
	2		**0.901**
	3		13.7
	4		10.6

Table 4: Running time with different tolerance and methods, the input sizes for N and M are 10^8, precision is double.

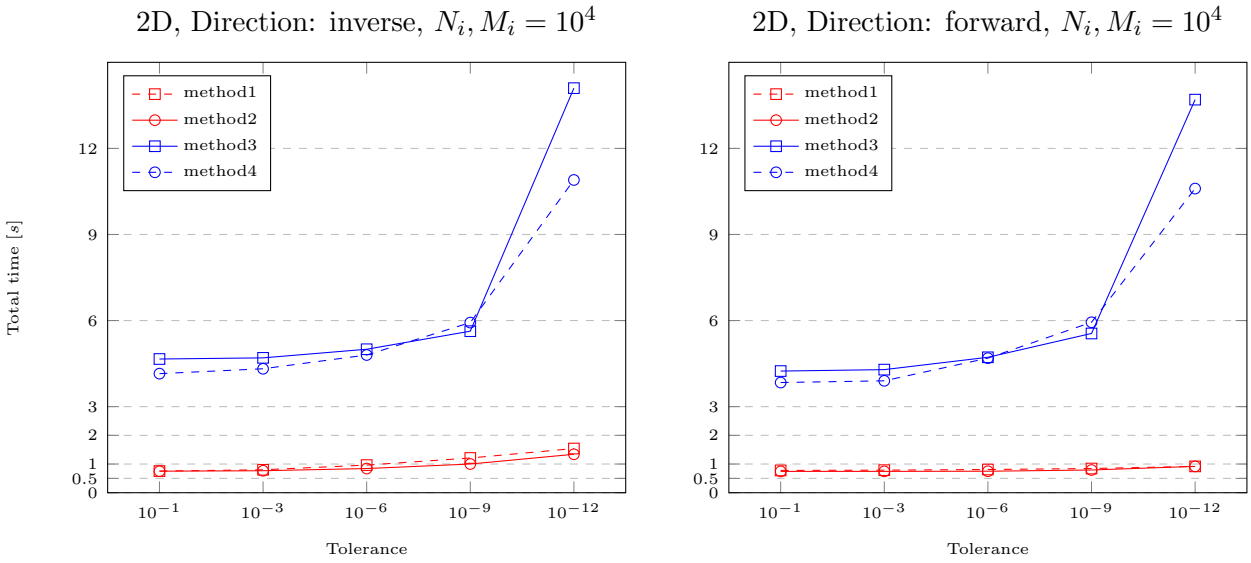

Figure 1: Running time with different tolerance.