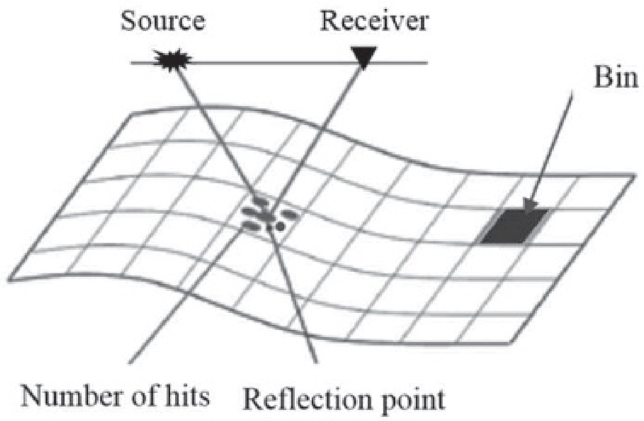
Application of GPU parallelization for non-uniform Fourier interpolation

Kai Zhuang, and Daniel Trad

Dec 2, 2022 CREWES annual sponsor's meeting

NSERC CRSNG
UNIVERSITY OF CALGARY
FACULTY OF SCIENCE
Department of Geoscience

DFT versus FFT for multi-dimension interpolation



DFT Advantages

- Higher accuracy of interpolated data
  - No binning of traces (works better for narrow azimuth and long offsets)

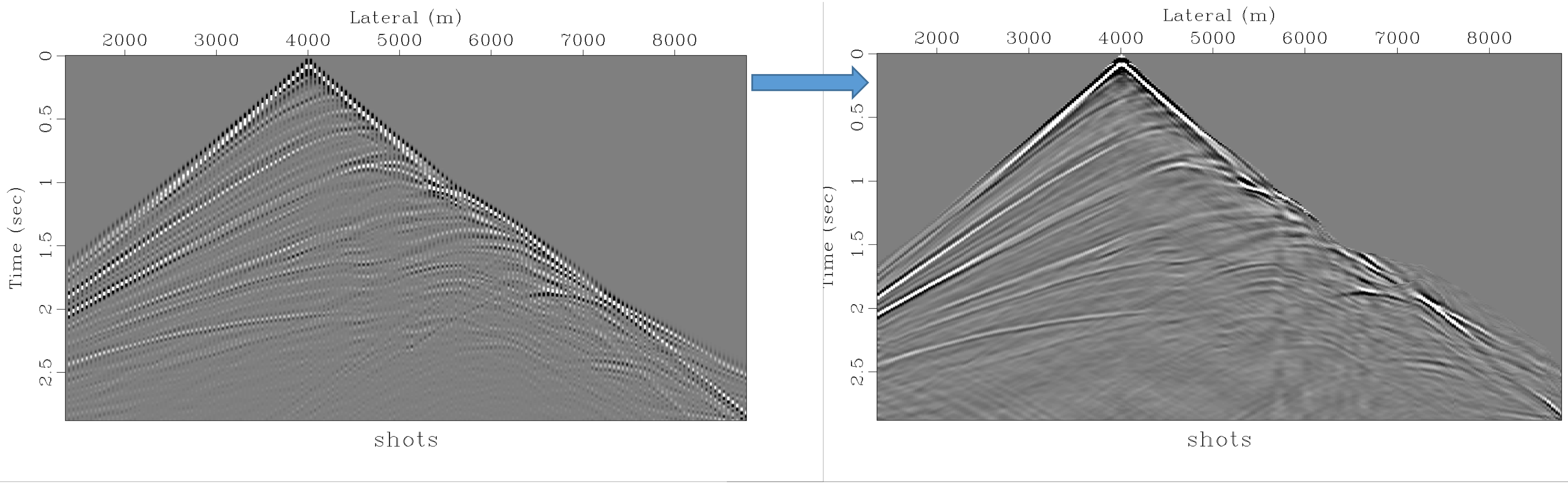
DFT Drawbacks

- Significantly increased memory usage
  - Need to pre-calculate Fourier operator (dependent on data size)

- DFT has generally increased run times vs binned FFT
  - Can be solved using GPUs (FFT scales poorly in parallel)

DFT interpolation

$$\left|\left|d - \underbrace{\boxed{\mathrm{T}}}_{\text{Sampling}}\underbrace{\boxed{F}}_{\text{DFT}}{}^{-1}\underbrace{\boxed{\mathrm{p}}}_{\text{Masking}}m\right|\right|_2^2 + \mu||m||_1^1$$



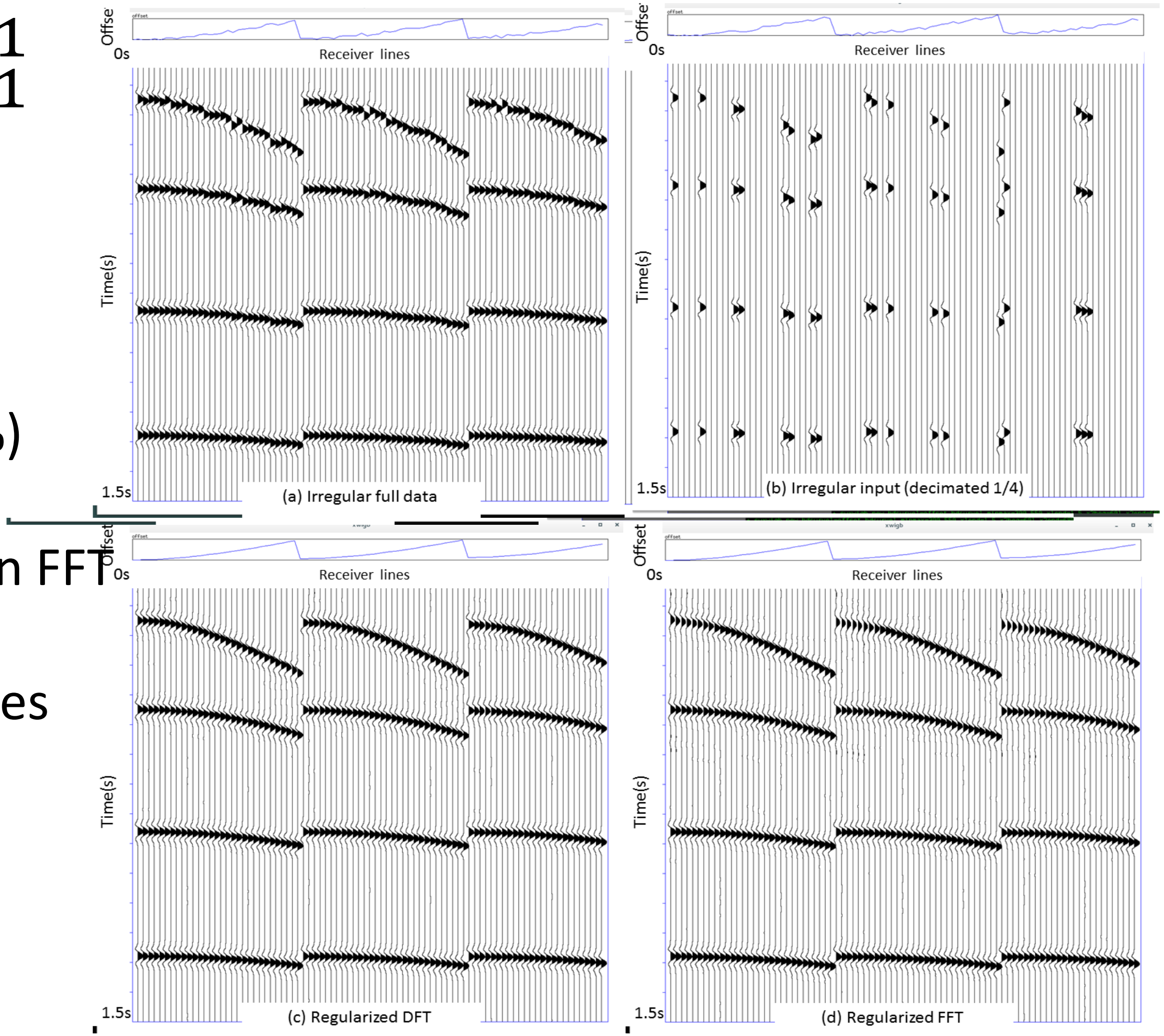
DFT interpolation

$$\left|\left| d - \underset{\text{Sampling}}{\boxed{T}}\,\underset{\text{DFT}}{\boxed{F}}^{-1}\underset{\text{Masking}}{\boxed{p}}\,m \right|\right|_2^2 + \mu ||m||_1^1$$

In highly decimated cases (decimated by 75%)

- DFT has better curvature preservation than FFT

- Jitter is removed from full data in both cases



(a) Irregular full data

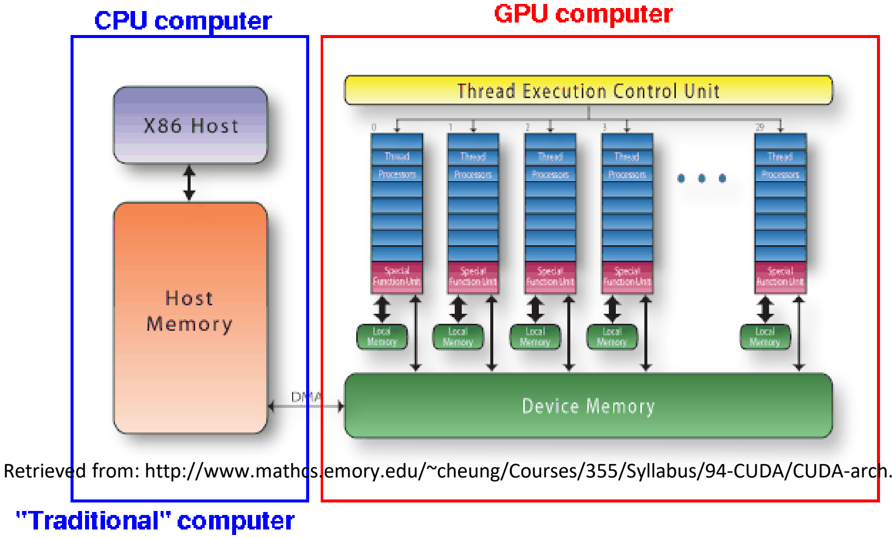
(b) Irregular input (decimated 1/4)

(c) Regularized DFT

(d) Regularized FFT

Why GPUs

GPUs	CPUs
built to handle massive amounts of light weight tasks simultaneously	built to handle complex tasks very quickly
82 SMs * 2048 Threads (RTX 3090)	64 Cores * 2 Threads (Threadripper 3990x)
High throughput	Low Latency
Best for parallel processing	Best for multi-serial processing



Retrieved from: http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/94-CUDA/CUDA-arch.html

CUDA

General architecture of GPUs

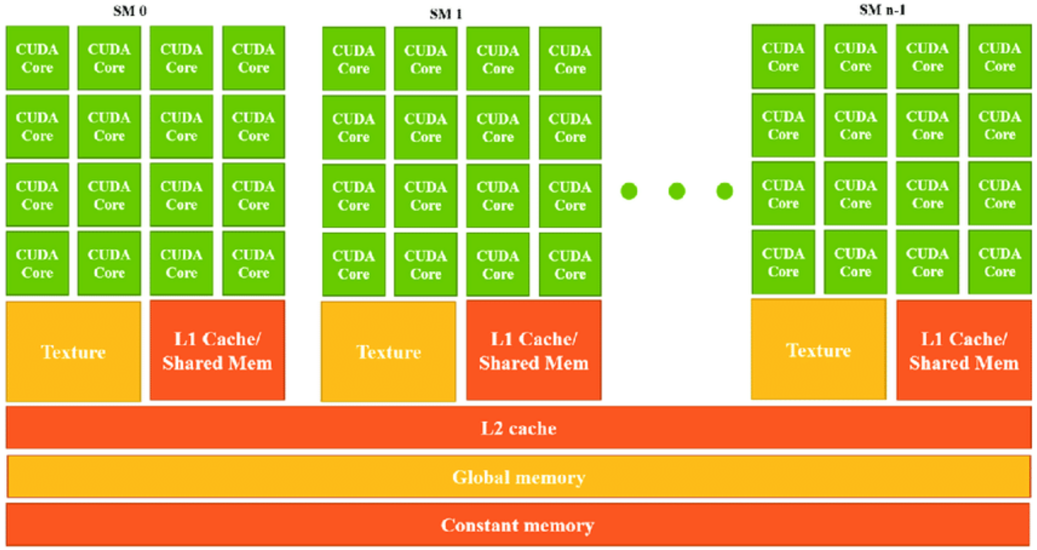
-A GPU is grouped into compute units call SMs (streaming multiprocessors)

-Each SM possesses a number of “threads” (generally 2048 threads)

-Threads are grouped into “warps” of 32 threads each (1 CUDA core = 1 warp)

-Threads in a warp must do the same operation (+,-,*,/)

-Avoid branching of work



GPU Challenges

Implementation

- Multidimensional access will always be non-sequential
  - 5D sums in 4 Spatial dimensions
- Multiple conflicting implementations
  - Shared memory vs SHFL transfers cannot be used at same time (for DFT)
- Ram usage for pre-calculation of DFT operator major limit on volume to interpolate
  - Common GPUs have small memory (~8-24GB)
  - Best GPUs are expensive (~80GB)

GPU Implementations

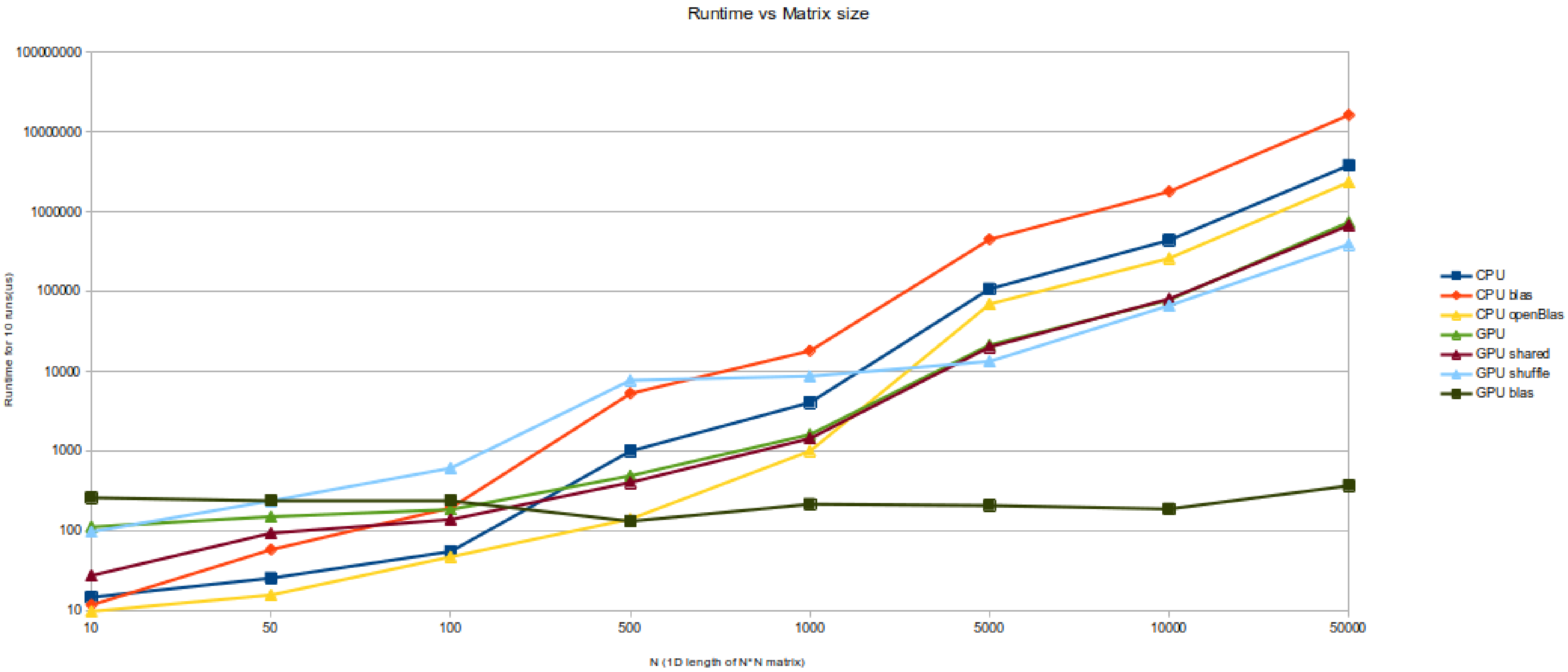
Shared memory

- User managed L3 Cache
  - Can be read from at very high speeds
  - Used to store reusable constant values
  - For DFT used to store current matrix part for calculation

Shuffle transfers

- Allows a thread to directly read from another thread’s register
  - Register is fastest, smallest data storage element
  - For DFT rather than reading from shared or global read directly from other thread

Results



Runtime vs Matrix size
Runtime for 10 runs(us)
100000000
10000000
1000000
100000
10000
1000
100
10
10
50
100
500
1000
5000
10000
50000
N (1D length of N*N matrix)
CPU
CPU blas
CPU openBlas
GPU
GPU shared
GPU shuffle
GPU blas

Conclusion

DFT has quality advantages of FFT implementations but is significantly more expensive in RAM usage and time (for CPUs)

Some implementations are a trade off between certain aspects of a kernel

- Especially dependent on size of operations

- Most performance CUBLAS mvm implementation cannot be used as we cannot modify the core code for our purposes.

Future Work

- Implement full 5D with DFT operator
  - Observe bottlenecks from dataflow implications

acknowledgement

We thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors, and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13.