Examples of GPU optimisation for wave propagation problems

Dr. Sofya Titarenko, PDRA in LIDA

Summary

- This talk will focus on NVDIA GPUs;
- Not every problem can be solved using GPU;
- Different problems gain different speedup when moving from CPU onto GPUs;
- It may take effort to find the best method of optimisation;
- For many problems it is probably easiest to start with NVIDIA libraries.

Q: When do we choose to use GPU?

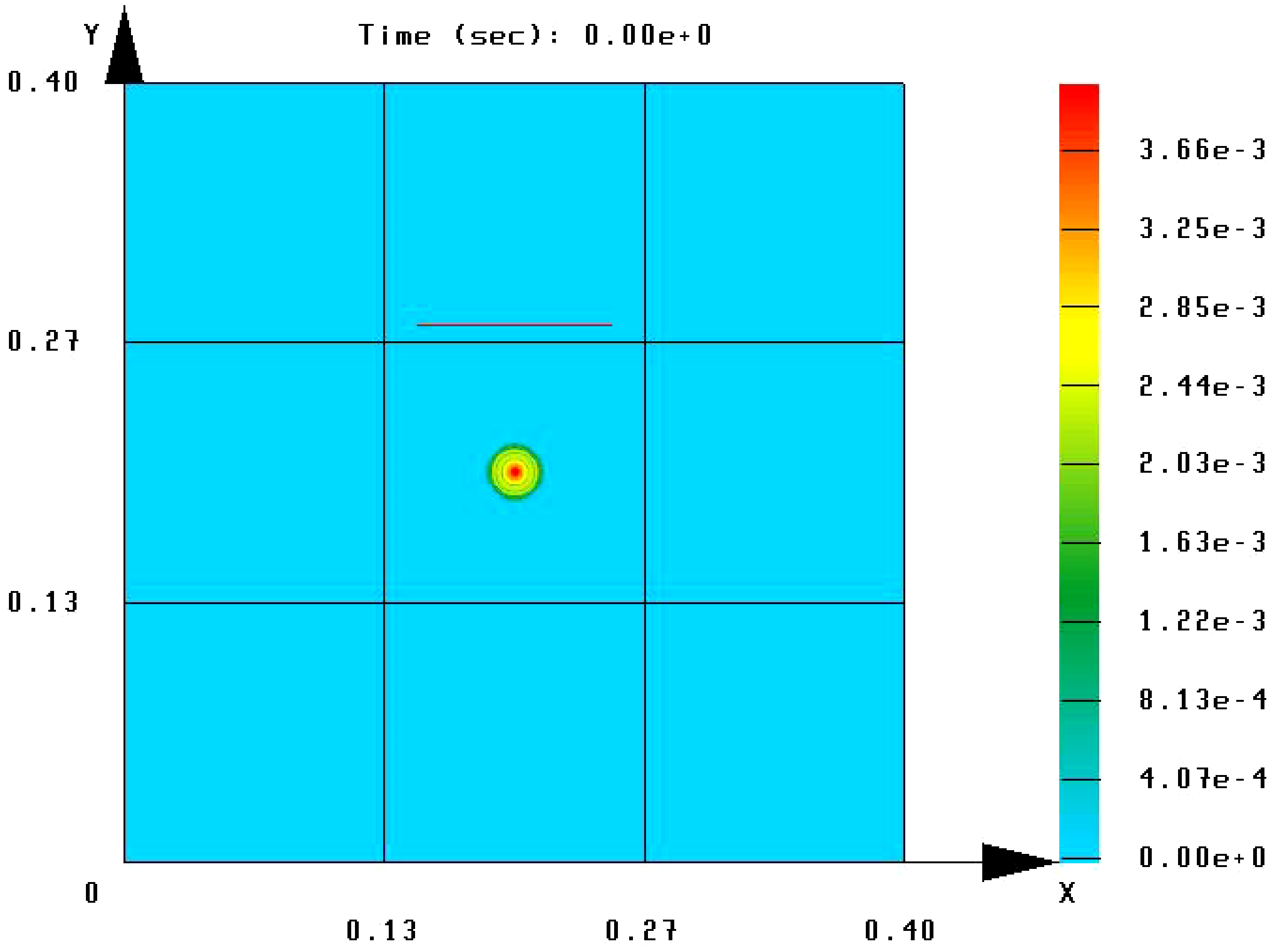
Answer: The problem has to include a routine calculation applied to a big set of data.

Single Instruction Multiple Data (SIMD) principle!

Examples: FFT, Large Matrix-Vector operations, Finite Difference calculations…

Solving wave equation, heat equation etc…

Wave propagation through elastic media with a fracture



Wave propagation through elastic media

$$
\begin{cases}
\rho \dfrac{\partial \dot{u}_1}{\partial t} = \dfrac{\partial \sigma_{11}}{\partial x_1} + \dfrac{\partial \sigma_{12}}{\partial x_2}, \\
\rho \dfrac{\partial \dot{u}_2}{\partial t} = \dfrac{\partial \sigma_{22}}{\partial x_2} + \dfrac{\partial \sigma_{12}}{\partial x_1}, \\
\dfrac{\partial \sigma_{11}}{\partial t} = \left(\lambda + 2\mu\right) \dfrac{\partial \dot{u}_1}{\partial x_1} + \lambda \dfrac{\partial \dot{u}_2}{\partial x_2}, \\
\dfrac{\partial \sigma_{12}}{\partial t} = \mu \dfrac{\partial \dot{u}_1}{\partial x_2} + \mu \dfrac{\partial \dot{u}_2}{\partial x_1}, \\
\dfrac{\partial \sigma_{22}}{\partial t} = \lambda \dfrac{\partial \dot{u}_1}{\partial x_1} + \left(\lambda + 2\mu\right) \dfrac{\partial \dot{u}_2}{\partial x_2},
\end{cases}
$$

Finite Difference approach, leap-frog approach

Applying FD formula for the whole array of $\sigma_{11}, \sigma_{12}, \sigma_{22}, \dot{u}_1, \dot{u}_2$

SIMD principle! **Good for GPU**

How we solve it on GPU?

1. Move data from **host (CPU)** to **device (GPU)**

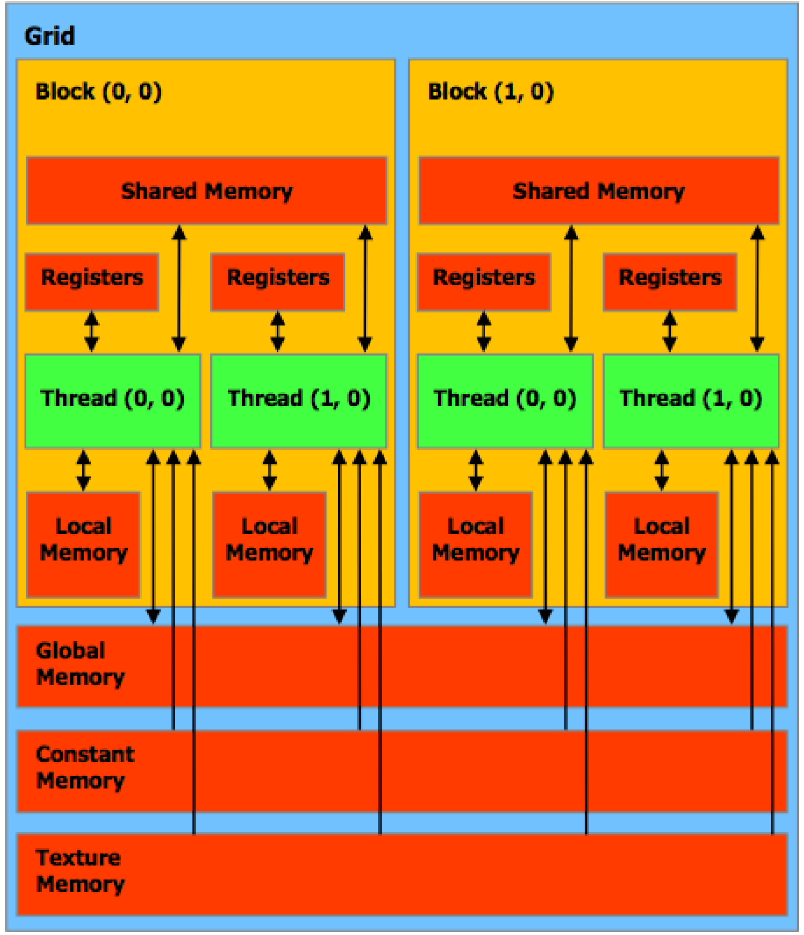
It is relatively slow, should be minimized. Use asynchronous transaction?

Global memory is large (ex. Tesla K80, **12GB**), but **slowest**

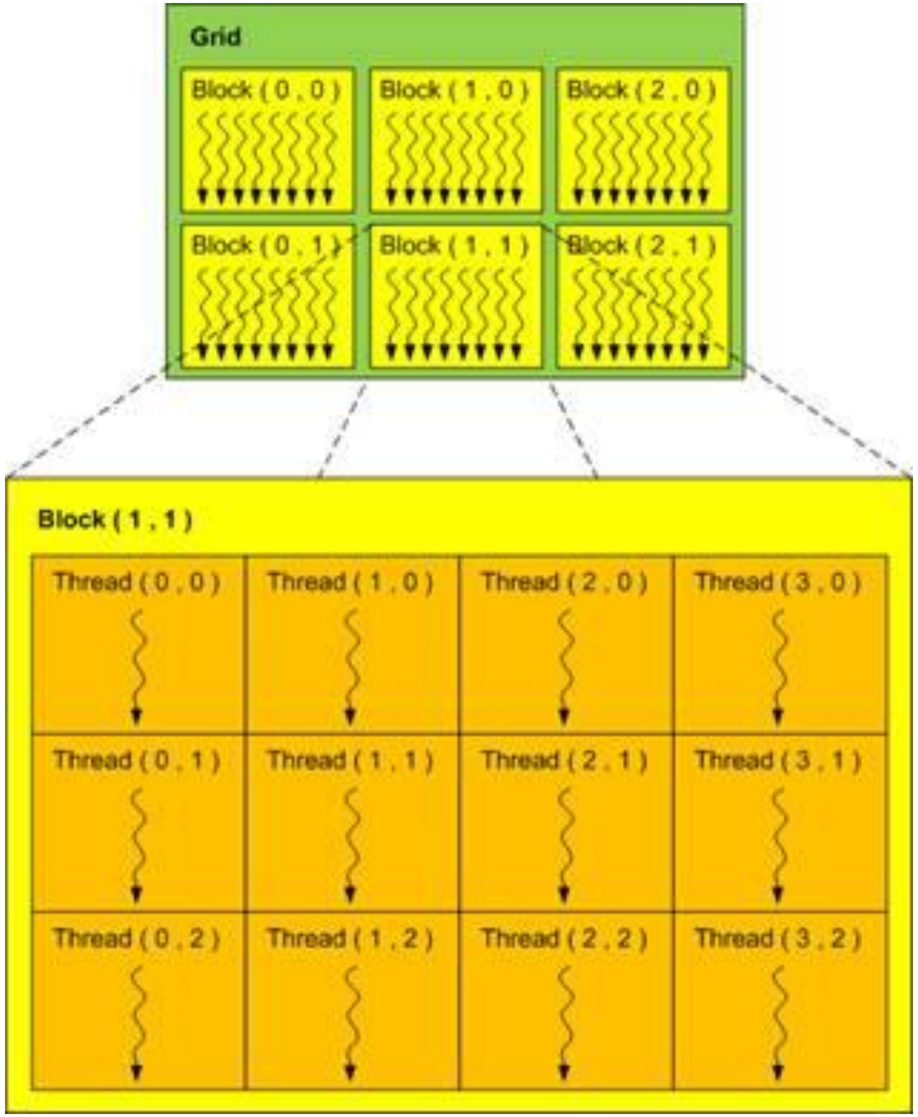
Shared memory is small (ex. Tesla K80, **64KB**), but **fast**

It is more efficient to divide data into blocks so the data you work with can fit in shared memory.

Memory structure on GPU



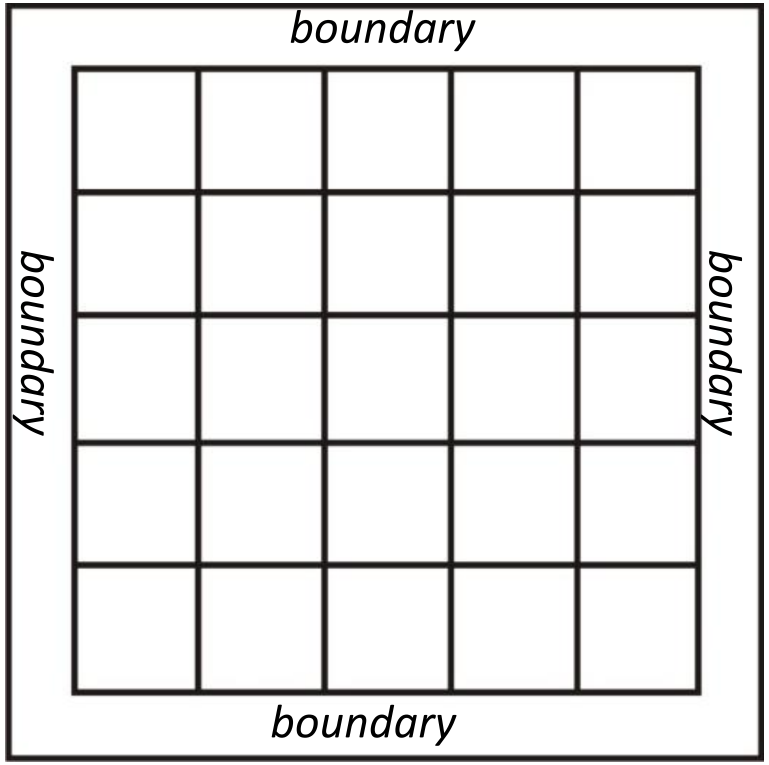
Running on GPU



- All threads on GPU can be represented as grid of blocks;
- Problem is run on GPU threads; parallelism

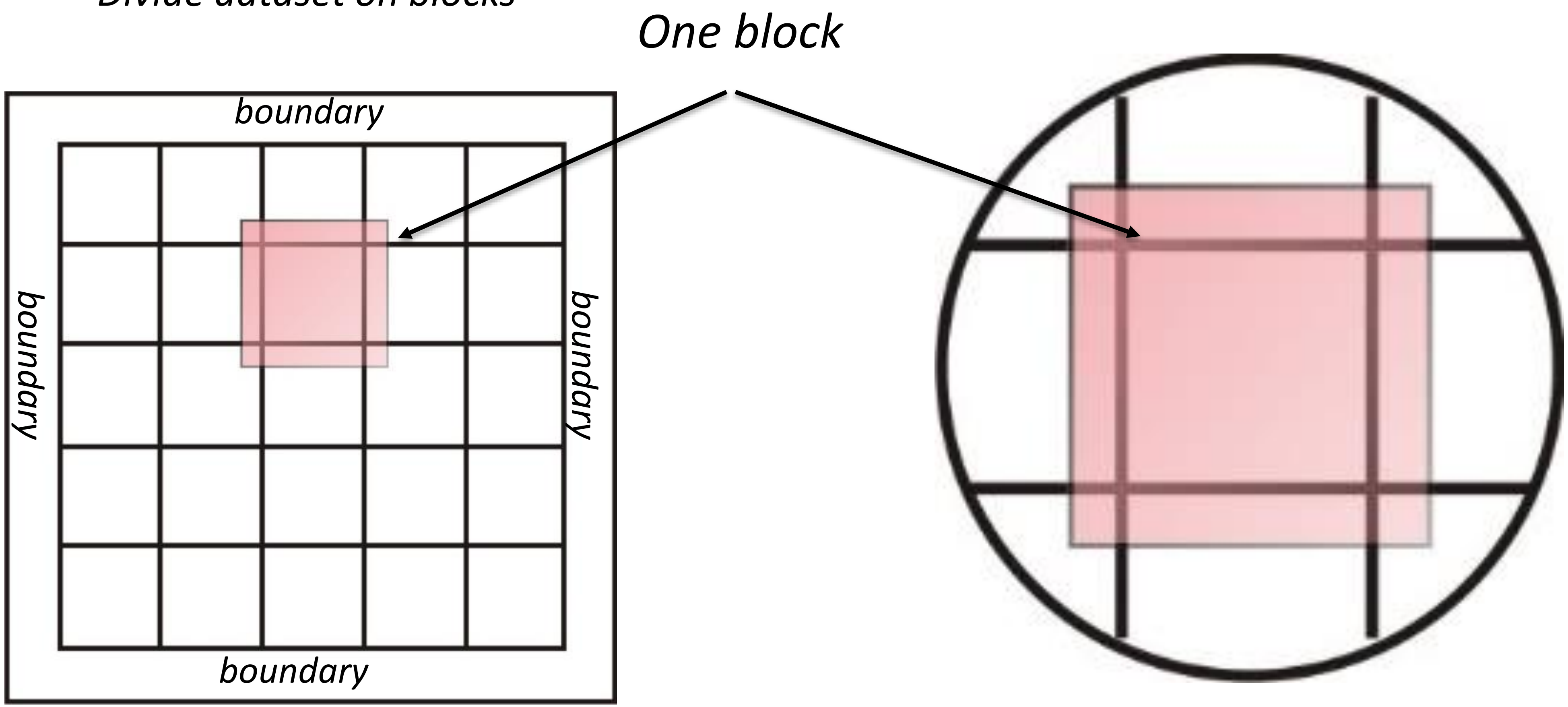
Wave propagation problem on GPU

Divide dataset on blocks



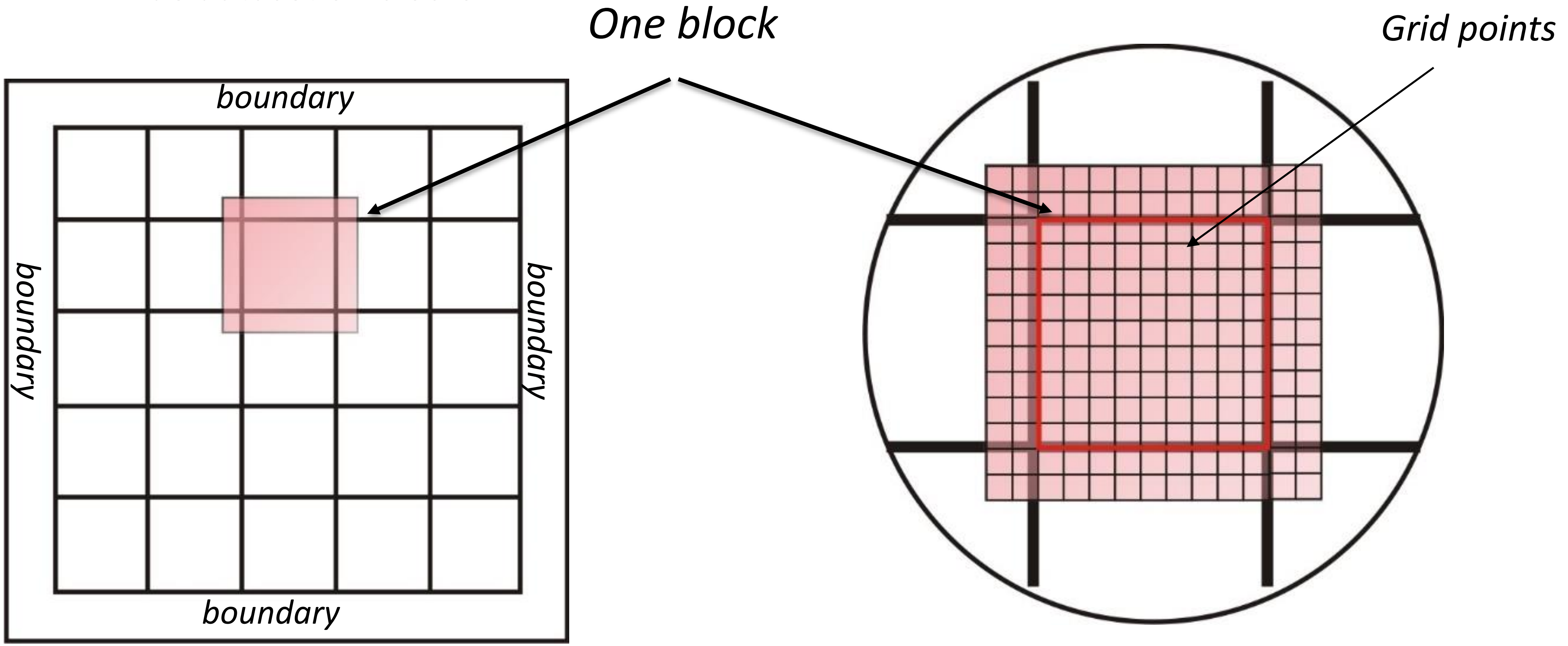
Wave propagation problem on GPU

Divide dataset on blocks

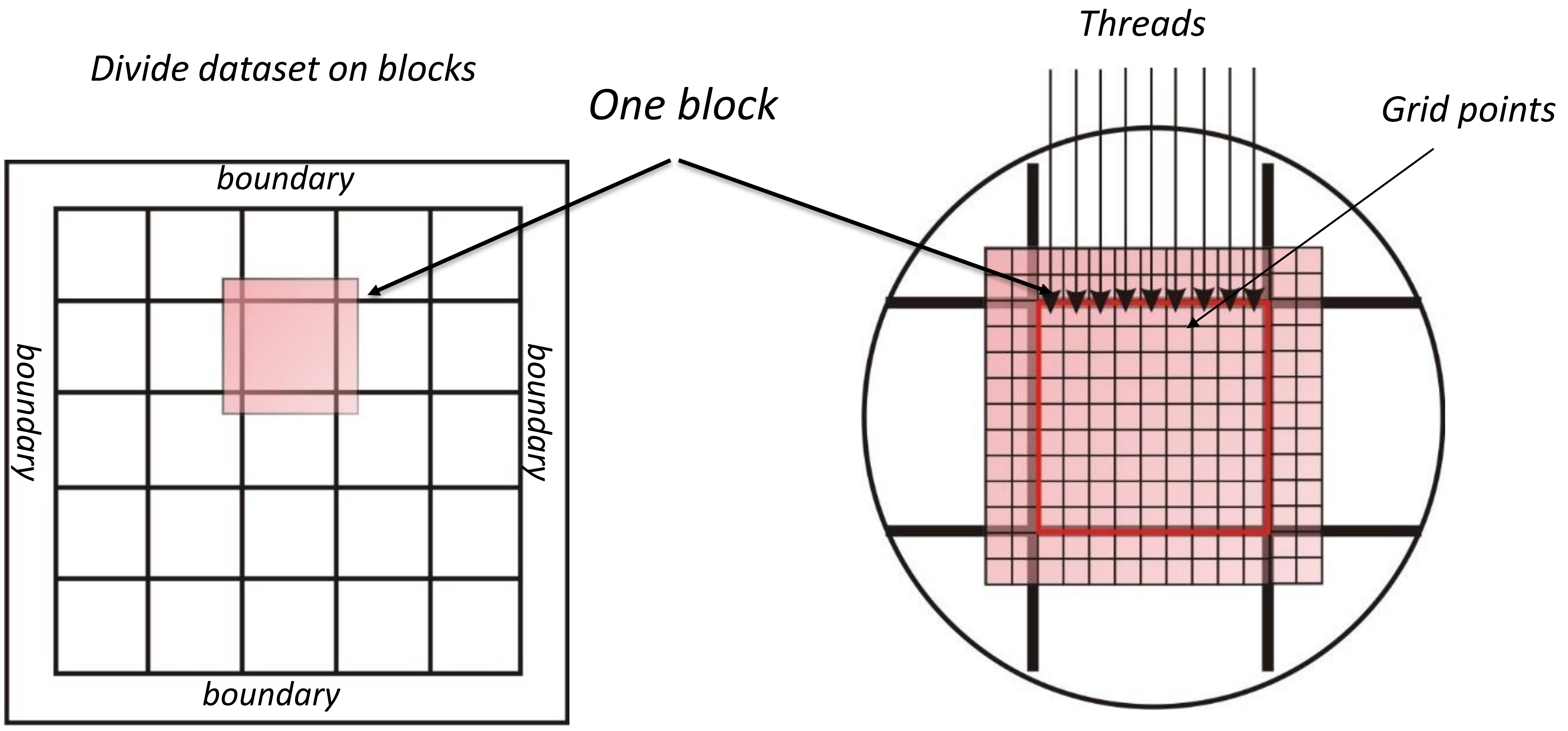


Wave propagation problem on GPU

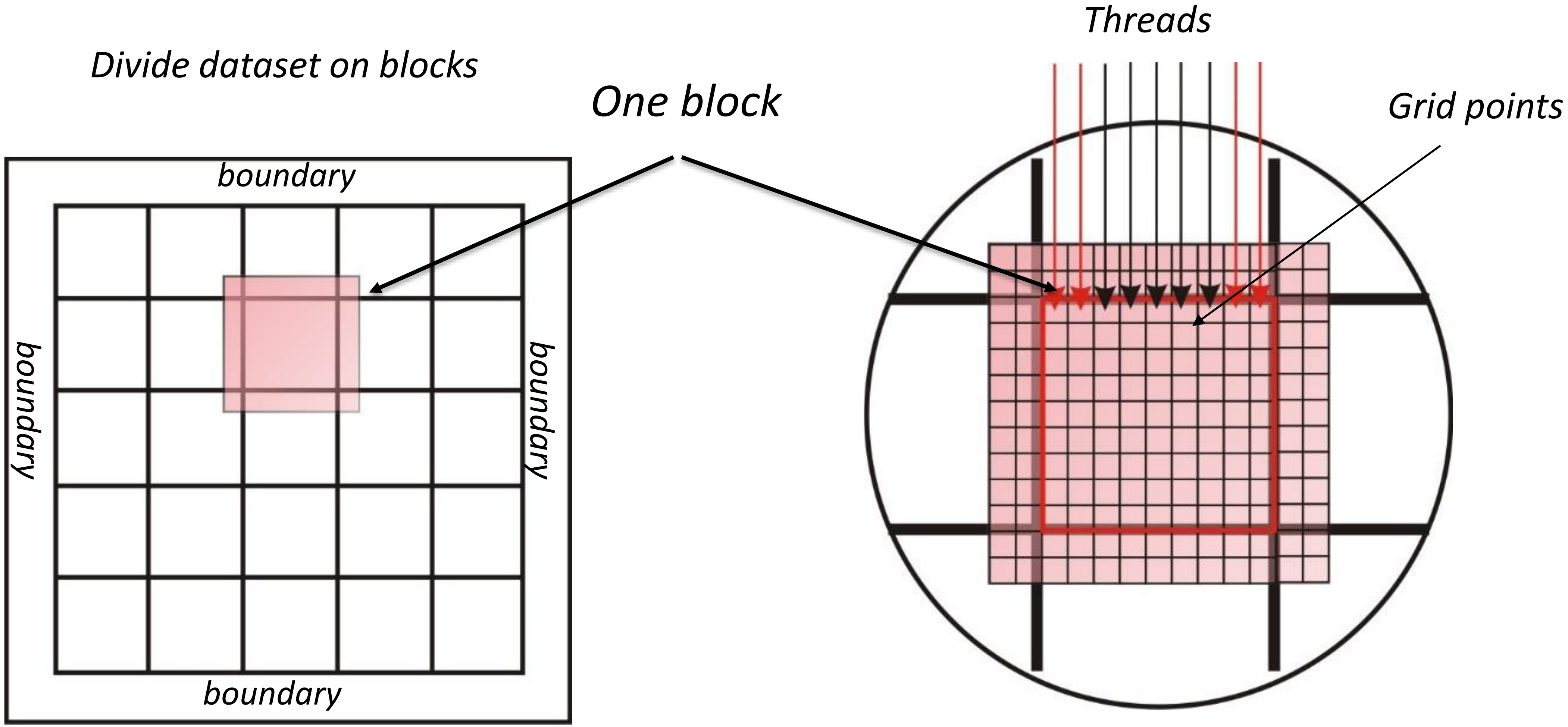
Divide dataset on blocks



Wave propagation problem on GPU



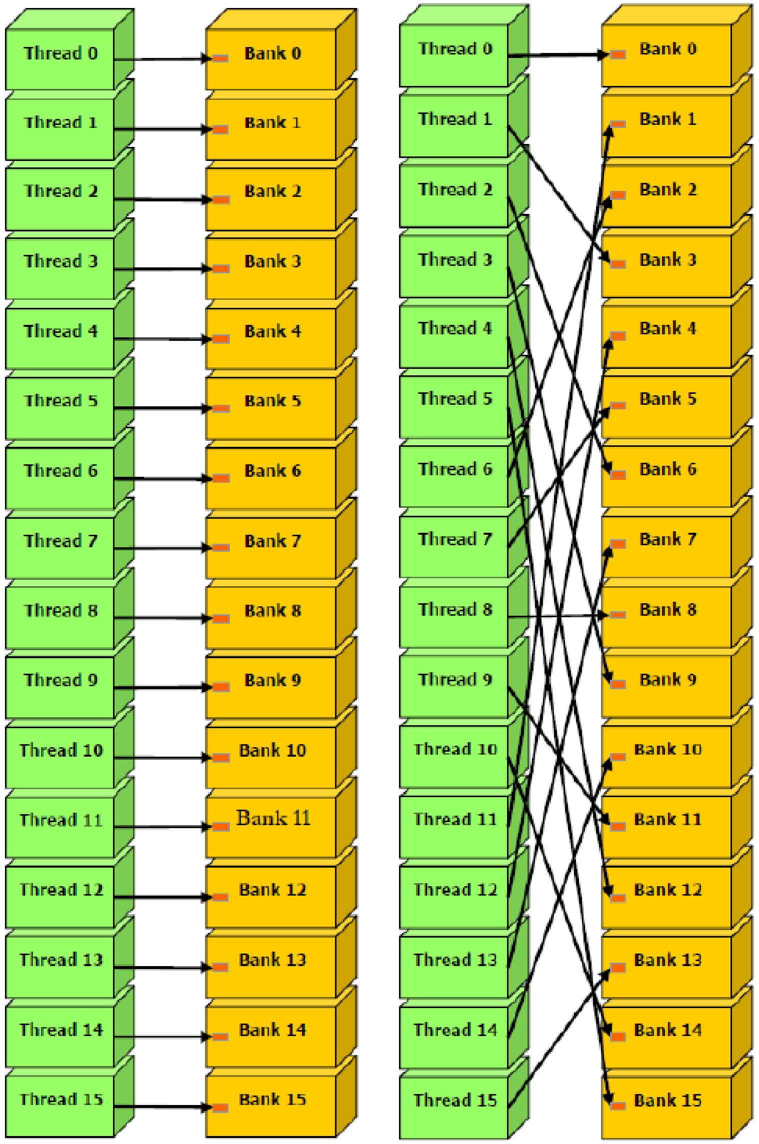
Wave propagation problem on GPU



Tips for acceleration

- Use shared memory;
- Use block sizes so
- they could fit in shared memory; In my example it is [32X32]
- Avoid bank conflicts;

Use NVIDIA profiler to get the idea where bottlenecks are



http://cuda-programming.blogspot.co.uk/2013/02/bank-conflicts-in-shared-memory-in-cuda.html

Speedup on single GPU (GeForce GTX 760, CPU Intel core i7, 6 cores, Sandy Bridge)

Problem: Wave propagation through elastic medium *with cracks*, 1000 steps

N, size=NXN	-O3, no OMP, sec	OMP, -O3, sec	OMP, -O3, improved, sec	GPU (using CUDA), sec	Speed up vs the most optimised
32X32X2	1164.440 (19.41 min)	227.556 (3.79 min)	125.662 (2.09 min)	**10.571**	**11.89**
32X64X2	4426.665	882.383	460.963	**41.640**	**11.07**

Conclusion:

Speedup depends on the problem: it could be higher for the more complicated problems!

steps

	Speed up vs the most optimised
	8.73

Acceleration on multiple GPUs

Problem: Wave propagation through elastic medium *with cracks*, 1000 steps

size	/OMP best, sec	GeForce GTX 760, sec	GeForce GTX 770, sec	2 GeForce cads, sec	speedup
32X32X2	125.662	10.571	7.435	5.925	21.209
32X64X2	460.963	41.640	31.338	23.951	19.24

Estimation: **for two GeForce GTX 770** cards I would expect speedup ~ **30 times!**

Conclusion: speedup depends on the graphics card is used. But it make sense to upgrade a system with extra GPU card, even if it is less powerful

, 500 steps

…rce …ec	speedup
	16.95

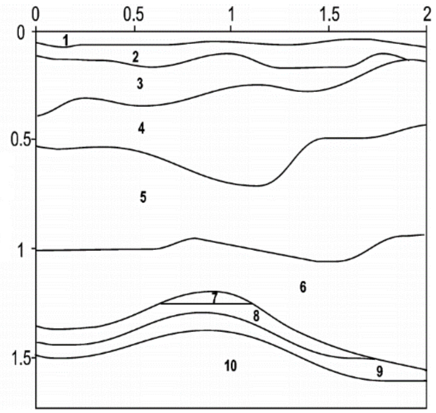
…*s multicore optimization"*, CF'17

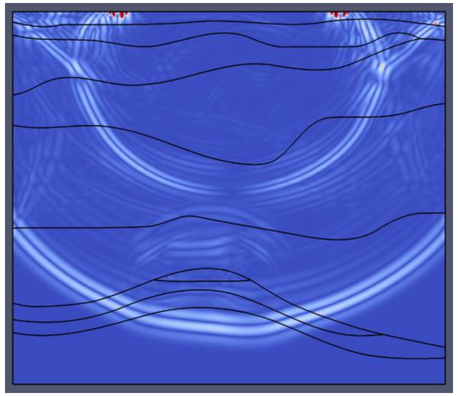
…, CCIS Springer Series

Wave propagation through layered media

"Applying CUDA and OpenCL technology for modelling seismic processes using grid-charactersitic methods", Andrey Ivanov, Nikolay Khokhlov, Moscow Institute of Physics and Technology, Russian Supercomputing Days 2016.

I.B. Petrov and N.I. Khokhlov *"Modeling 3D Sesmic Problems Using High-Perfomance Computing systems",* Mathematical Models and Computer Simulations, 2014





Grid-characteristic monotonic difference method

TVD (Total Variation Diminishing) Finite Difference Scheme

Applying TVD finite difference formula to all 9 arrays of variables $\{\dot{u}_1, \dot{u}_1, \dot{u}_1, \sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{22}, \sigma_{23}, \sigma_{33}\}^T$

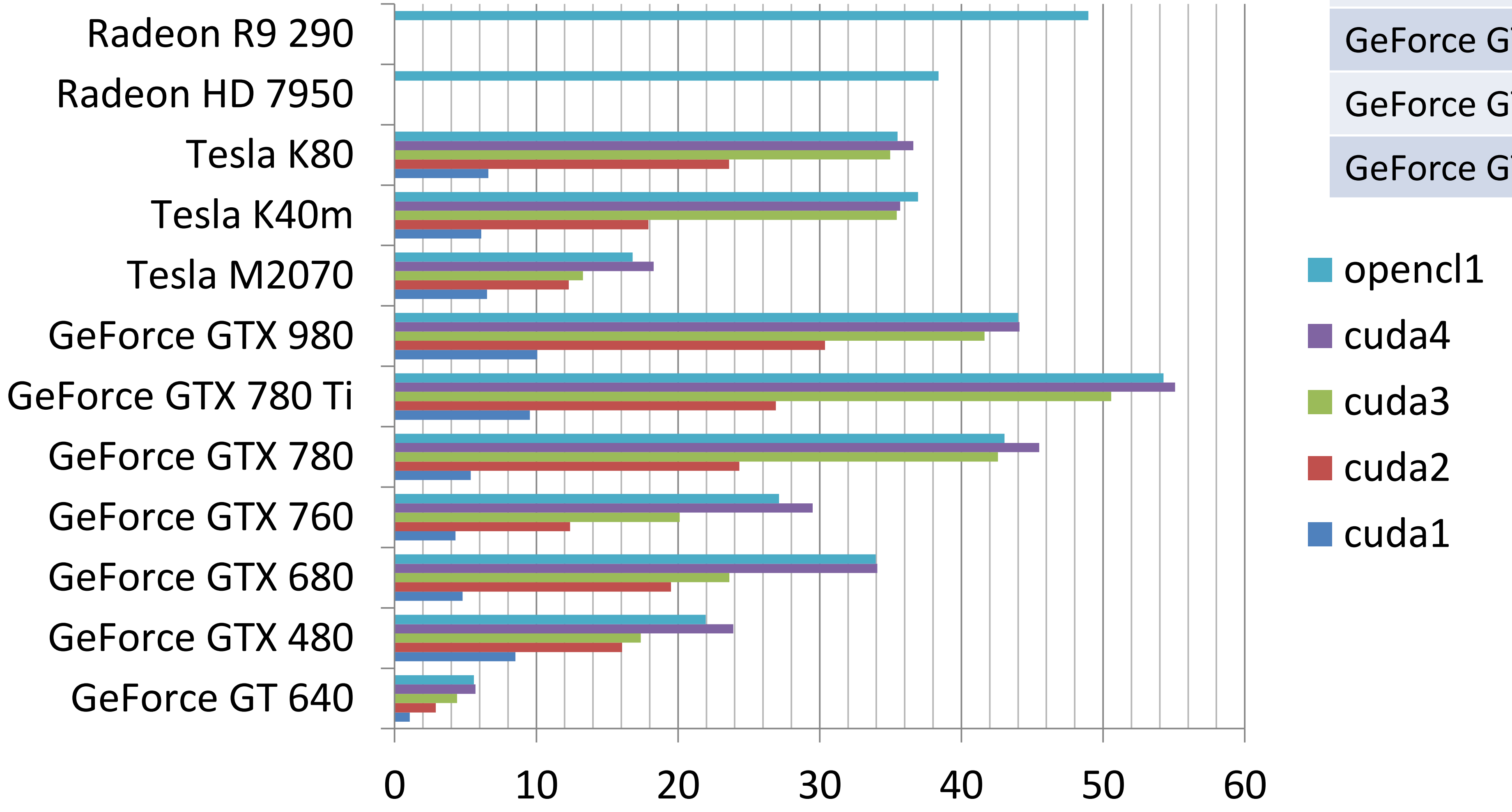
SIMD principle! **Good for GPU**

- CPU
  - Intel Xeon E5-2697 2.7 GHz
  - Compilers: icc
  - Compiler Options:
    - -mavx
    - -fopenmp (auto vectorization)
    - -O2
- GPU
  - Compilers: nvcc, gcc
  - Compiler Options:
    - -O2
    - -use_fast_math

GPU	Cores	Clock rate, MHz	GFLOPS - single precision	SP:DP	GFLOPS - double precision
GeForce GT 640	384	900	691	24	29
GeForce GTX 480	480	1401	1345	8	168
GeForce GTX 680	1536	1006	3090	24	129
GeForce GTX 760	1152	980	2258	24	94
GeForce GTX 780	2304	863	3977	24	166
GeForce GTX 780 Ti	2880	876	5046	24	210
GeForce GTX 980	2048	1126	4612	32	144
Tesla M2070	448	1150	1030	2	515
Tesla K40m	2880	745	4291	3	1430
Tesla K80	2496	562	2806	1.5	1870
Radeon HD 7950	1792	800	2867	4	717
Radeon R9 290	2560	947	4849	8	606

Speedup GPU/CPU, single precision

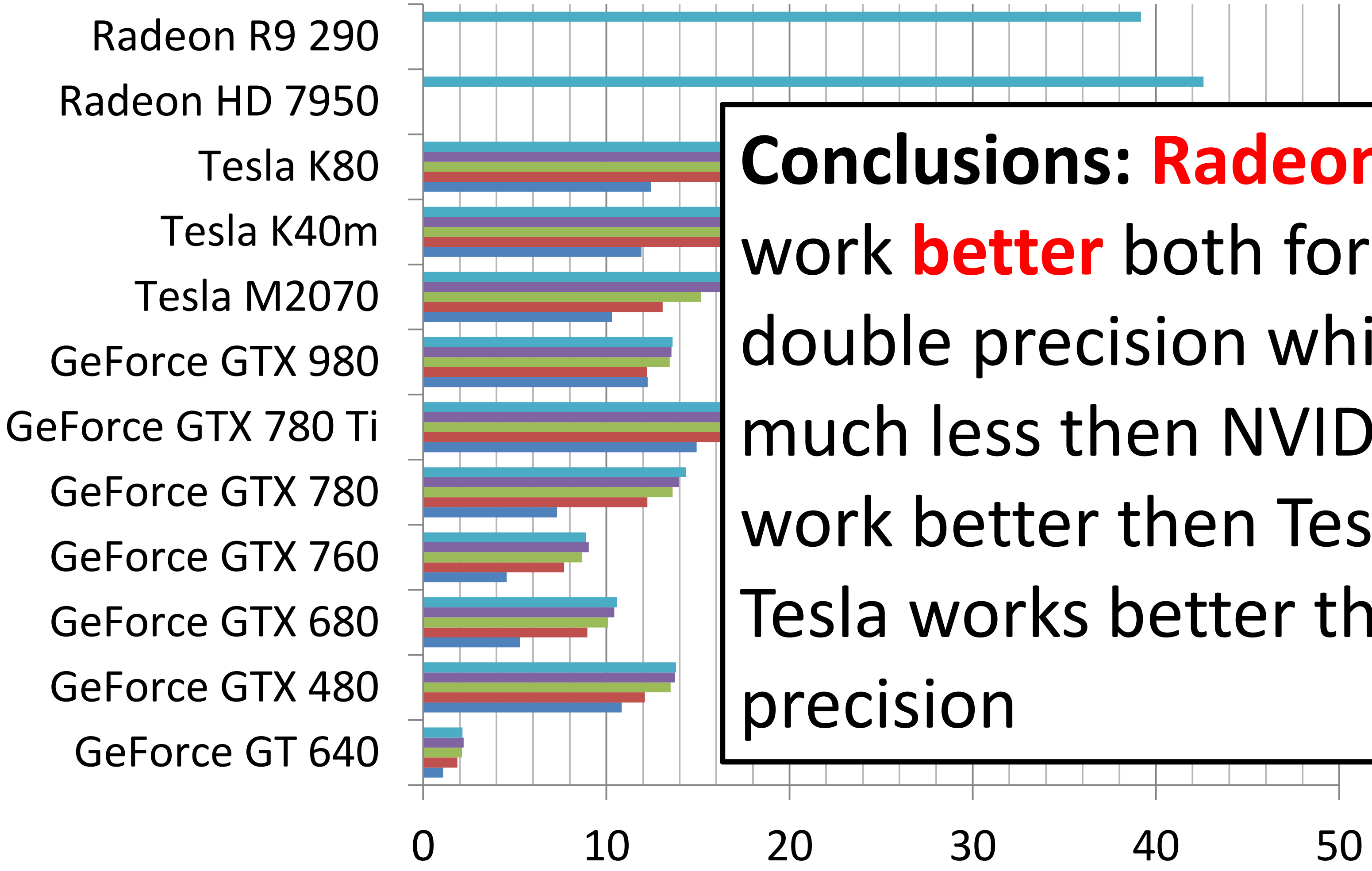
"Applying CUDA and OpenCL technology for modelling seismic processes using grid-charactersitic methods", Andrey Ivanov, Nikolay Khokhlov, Moscow Institute of Physics and Technology, Russian Supercomputing Days 2016.



GPU	cost
Radeon R9 290 (AMD)	£314
Tesla K80 (NVIDIA)	£4,962
GeForce GTX 780	
GeForce GTX 770	£429.98
GeForce GTX 760	£149.99

Speedup GPU/CPU, double precision

"Applying CUDA and OpenCL technology for modelling seismic processes using grid-charactersitic methods",
Andrey Ivanov, Nikolay Khokhlov, Moscow Institute of Physics and Technology, Russian Supercomputing Days 2016.



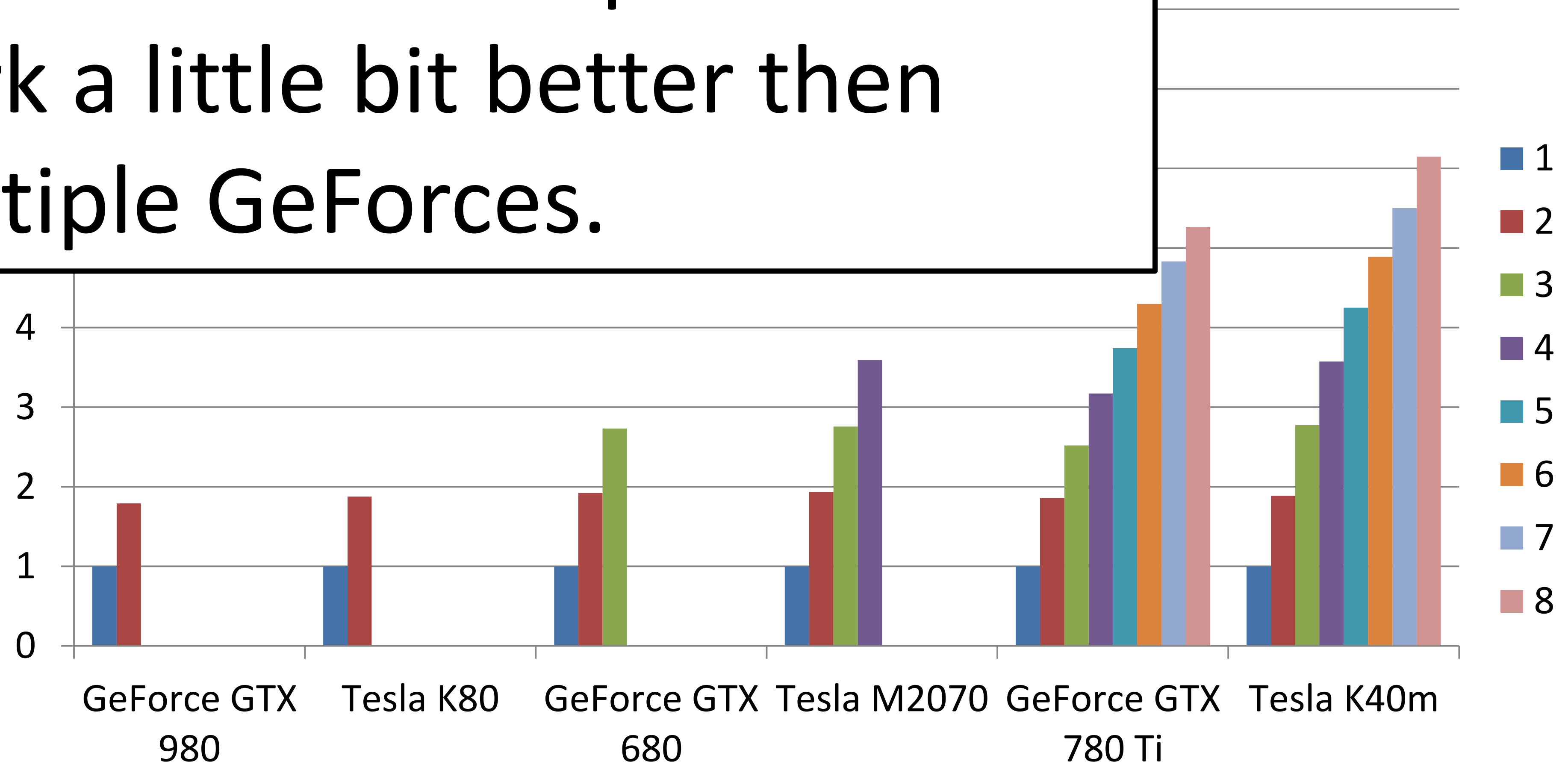
Conclusions: **Radeon** AMD cards are actually work **better** both for single precision and double precision while cost much less then NVIDIA Tesla! GeForce can work better then Tesla for single precision, Tesla works better then GeForce for double precision

Speedup on multiple GPUs, CUDA (for multiple GPU processor setup)

"Applying CUDA and OpenCL technology for modelling seismic processes using grid-charactersitic methods",

Days 2016.

Conclusions: multiple Teslas work a little bit better then multiple GeForces.



Problem: wave propagation through irregularly layered medium

Conclusions

- GPU parallelisation can help sufficiently speed up calculations;
- GPU parallelisation can help sufficiently speed up calculations while keeping hardware and software expenses low;
- Using multiple GPUs where it is possible can help further speed up of calculations.

Some tips

- The formulation of problem we want to calculate does matter;
- Try to formulate problem using SIMD principle;
- Try to minimise exchange between CPU host and GPU;
- Try to use fast shared memory where it is possible; Block size fit in shared memory;
- Try to avoid “if” construction in your loop and use threads efficiently;

Thank you for your attention!

Any questions?