Applying OpenCL technology for seismic modeling using grid-characteristic methods

Andrey Ivanov, MIPT
Nikolay Khokhlov, MIPT

Quasilinear equations, inverse problems and their applications
Moscow Institute of Physics and Technology, Dolgoprudny, 12-15 Sept. 2016

Outline

- Mathematical model and numerical method
- Test conditions
- Description of program
- Optimization
- Test results
  - Single GPU
    - Speedup (compared to GPU)
    - Percentage of peak performance
    - Performance (FLOPS)
  - Multiple GPUs
    - Speedup (compared to single GPU)
    - Speedup with GPUDirect

Mathematical model

Relation between velocity and deformation

$$\begin{cases} \rho\dot{\vec{v}} = \nabla \cdot \mathbf{T}, \\ \dot{\mathbf{T}} = \lambda(\nabla \cdot \vec{v})\mathbf{I} + \mu(\nabla \otimes \vec{v} + \vec{v} \otimes \nabla). \end{cases}$$

Motion equation

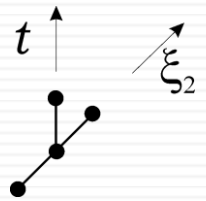
Hooke's law

ρ – density
λ, μ – Lame elastic parameters
v – velocity
T – stress tensor

Numerical method

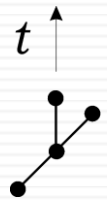
Split directions

$$\vec{u} = (\upsilon_x, \upsilon_y, \sigma_{xx}, \sigma_{xy}, \sigma_{yy})$$



$$\frac{\partial \vec{u}}{\partial t} + \mathbf{A}_1 \frac{\partial \vec{u}}{\partial \xi_1} = 0$$

$$\vec{u}' = \vec{u}^n - \tau \mathbf{A}_1 \Delta_1 \vec{u}^n$$



$$\frac{\partial \vec{u}}{\partial t} + \mathbf{A}_2 \frac{\partial \vec{u}}{\partial \xi_2} = 0$$

$$\vec{u}^{n+1} = \vec{u}' - \tau \mathbf{A}_2 \Delta_2 \vec{u}'$$

$$\vec{u}^{n+1} = \vec{u}^n - \tau(\mathbf{A}_1\Delta_1 + \mathbf{A}_2\Delta_2)\vec{u}^n + O(\tau^2)$$

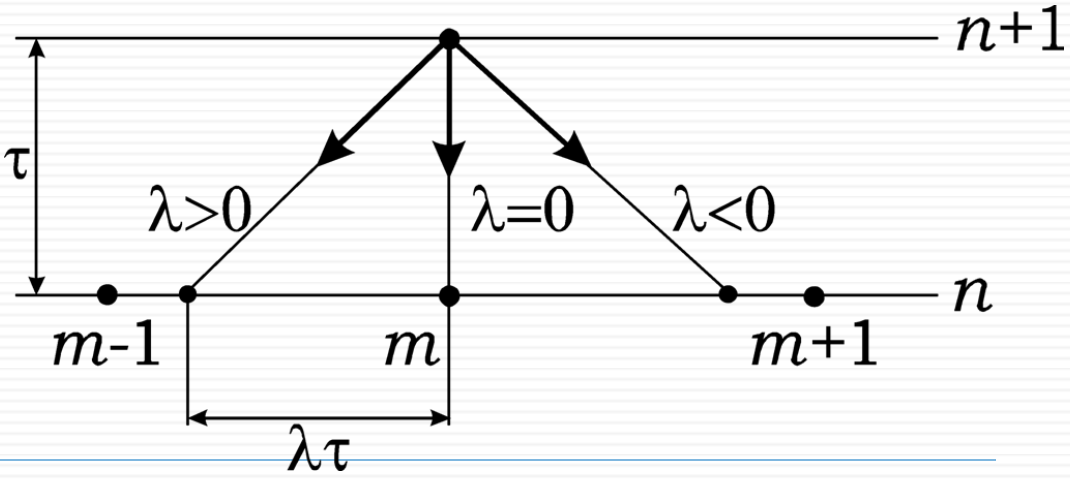
Hyperbolic problem

$$\mathbf{A} = \mathbf{\Omega}^{-1}\mathbf{\Lambda}\mathbf{\Omega}$$

$$\frac{\partial \vec{u}}{\partial t} + \mathbf{\Omega}^{-1}\mathbf{\Lambda}\mathbf{\Omega}\frac{\partial \vec{u}}{\partial \xi} = 0$$

$$\frac{\partial \vec{v}}{\partial t} + \mathbf{\Lambda}\frac{\partial \vec{v}}{\partial \xi} = 0 \quad (\vec{v} \equiv \mathbf{\Omega}\vec{u})$$

$$v^{n+1}(\xi) = v^n(\xi - \lambda\tau)$$



Test conditions

- CPU
  - Compilers: icc
  - Compiler Options :
    - -mavx
    - -fopenmp (auto vectorization)
    - -O2
- GPU
  - Compilers: nvcc, gcc
  - Compiler Options:
    - -O2
    - -use_fast_math

CPU properties: Intel Xeon E5-2697 2.7 GHz

GPU properties:

GPU	CUDA cores (streaming processors)	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

Test program

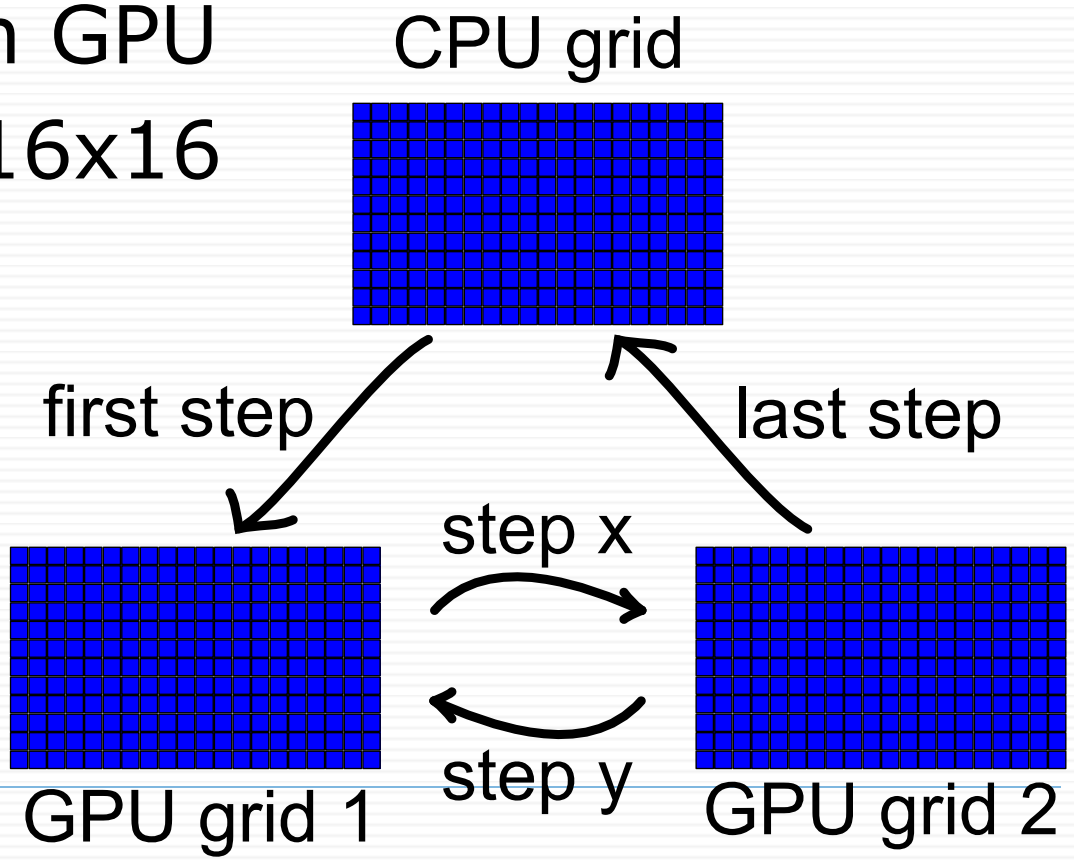
- Grid size: 4096x4096
- Time steps: 6500
- Data type: float, double
- Grid node: 5 float (double)
- Occupied memory:
  - 320 MB (float)
  - 640 MB (double)

CPU version

- Single-precision and double-precision
- 190 FLOPS to recalculate one node in grid
- Program consumes 18.8 TFLOPS
- Single-thread, single CPU core
- AVX instructions – vectorization

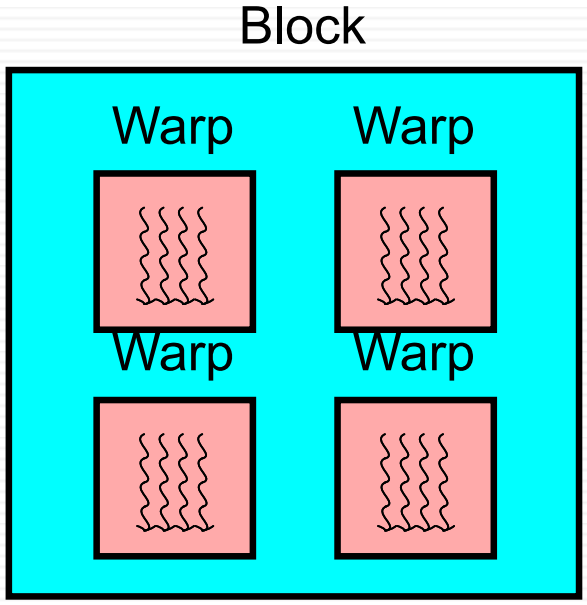
Optimization

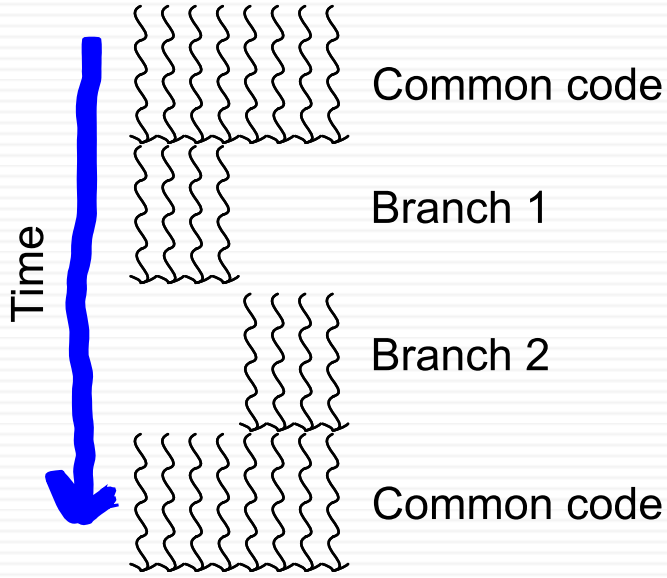
- Array of structures (AOS)
- Two grids on GPU
- Block sizes 16x16



Optimization

- Structure of arrays (AOS -> SOA)
- Coalesced memory access
- Use of GPU shared memory
- Reduce conditional branches

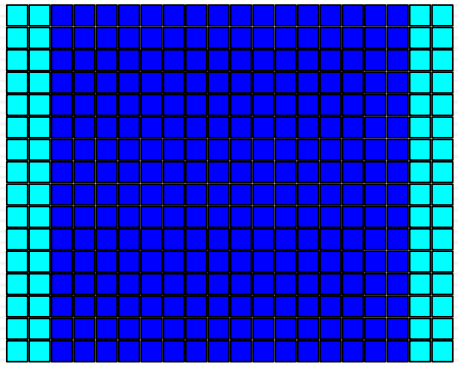




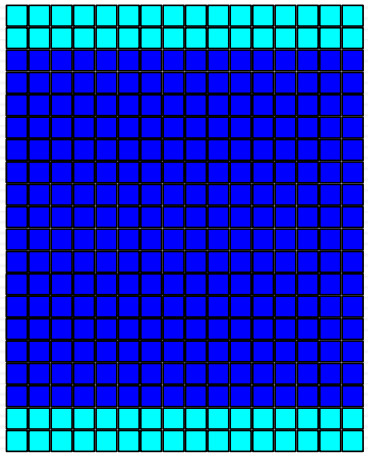
Optimization

- Block size in step X – 256x1
- Block size in step Y – 16x16

step X 16x16

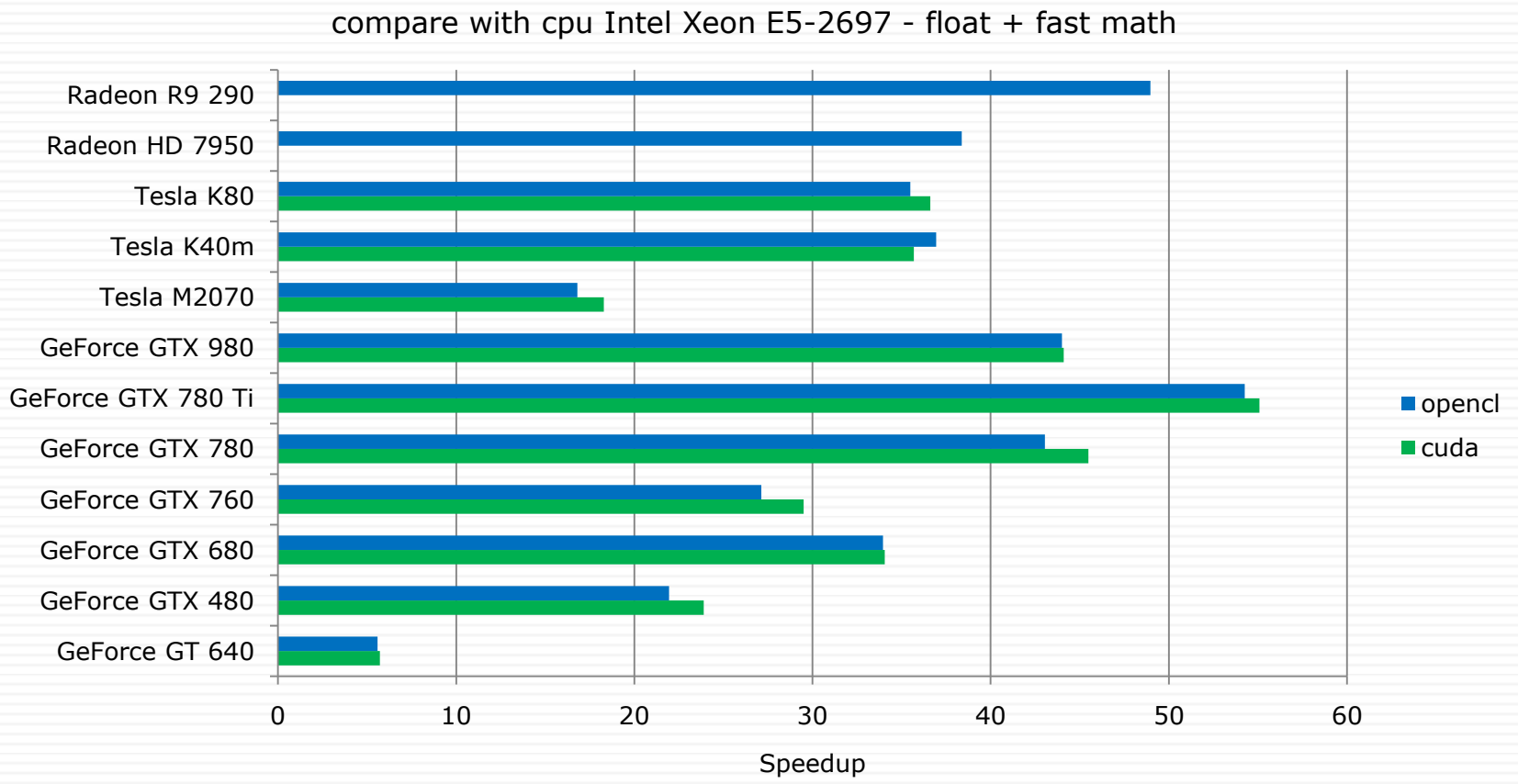


step Y 16x16

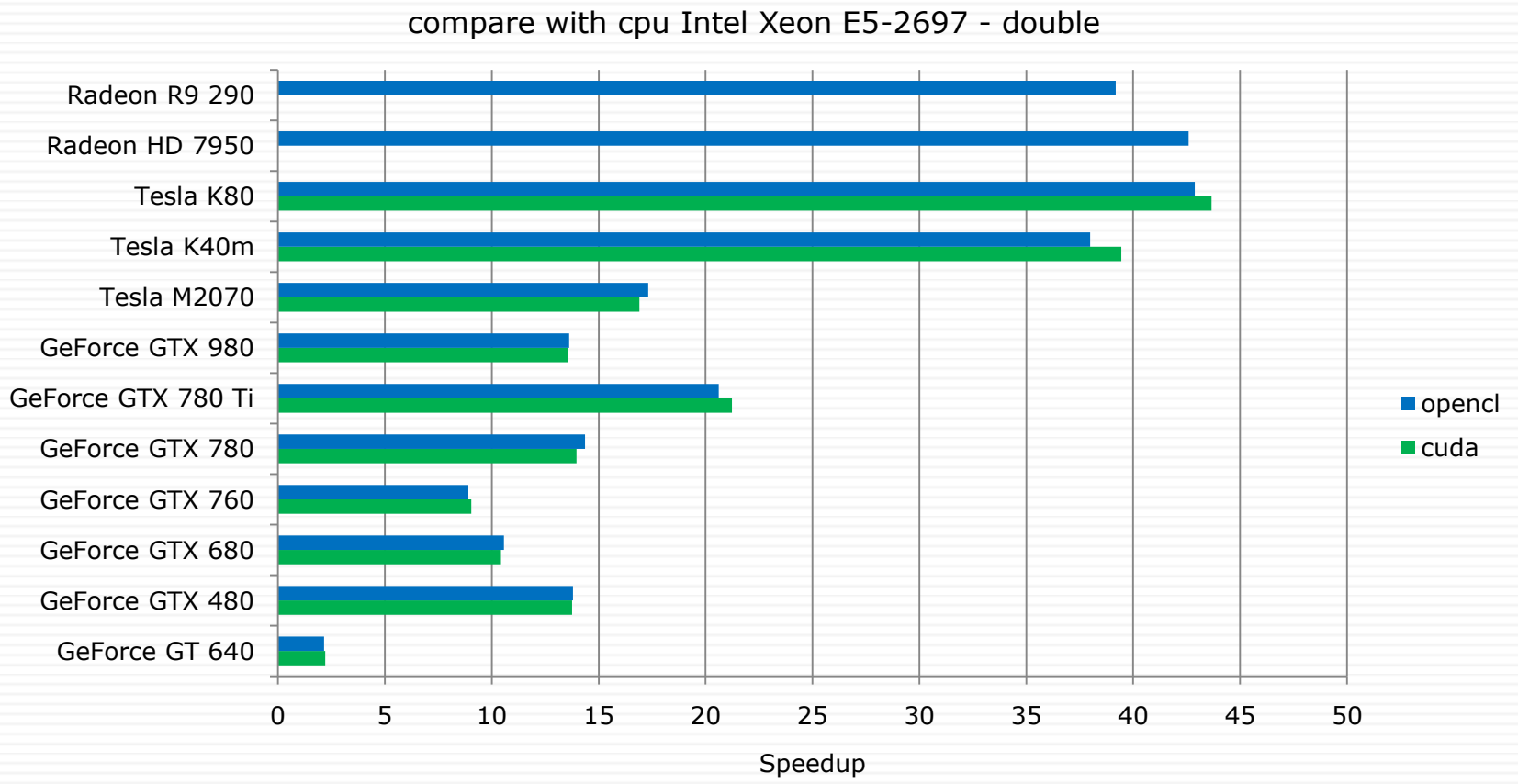


step X 256x1

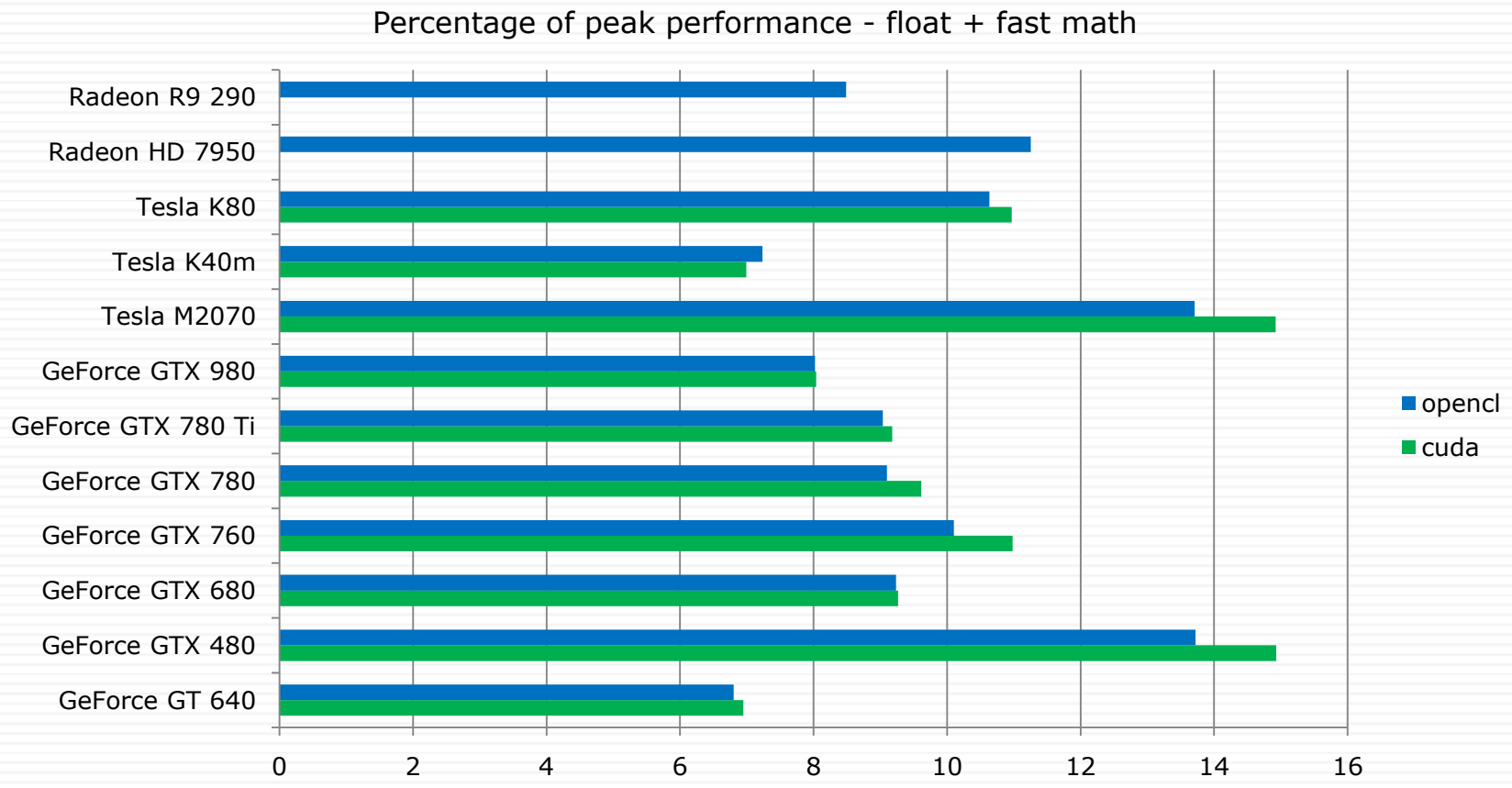
Speedup of GPU implementation compared to CPU



Speedup of GPU implementation compared to CPU

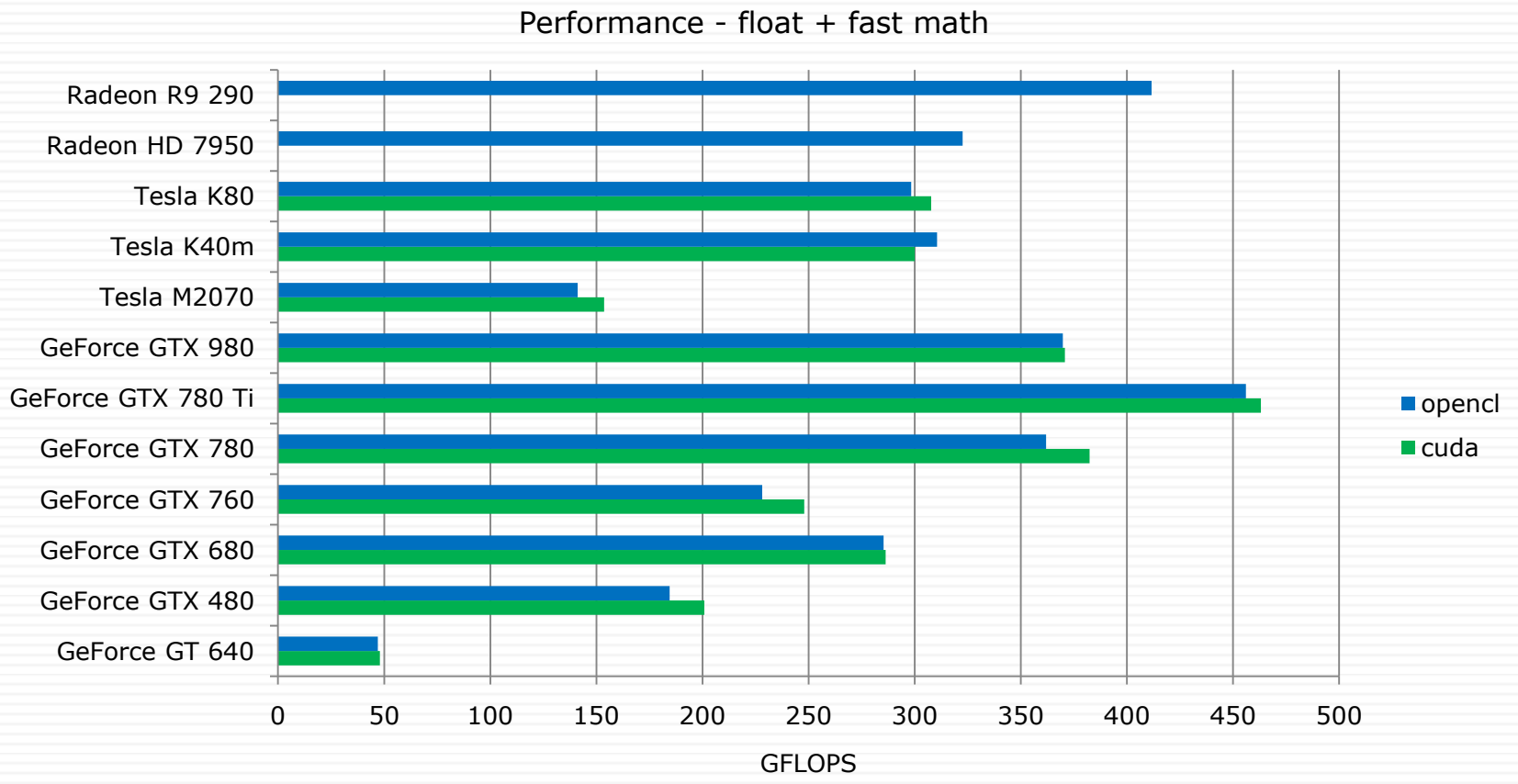


Percentage of peak performance

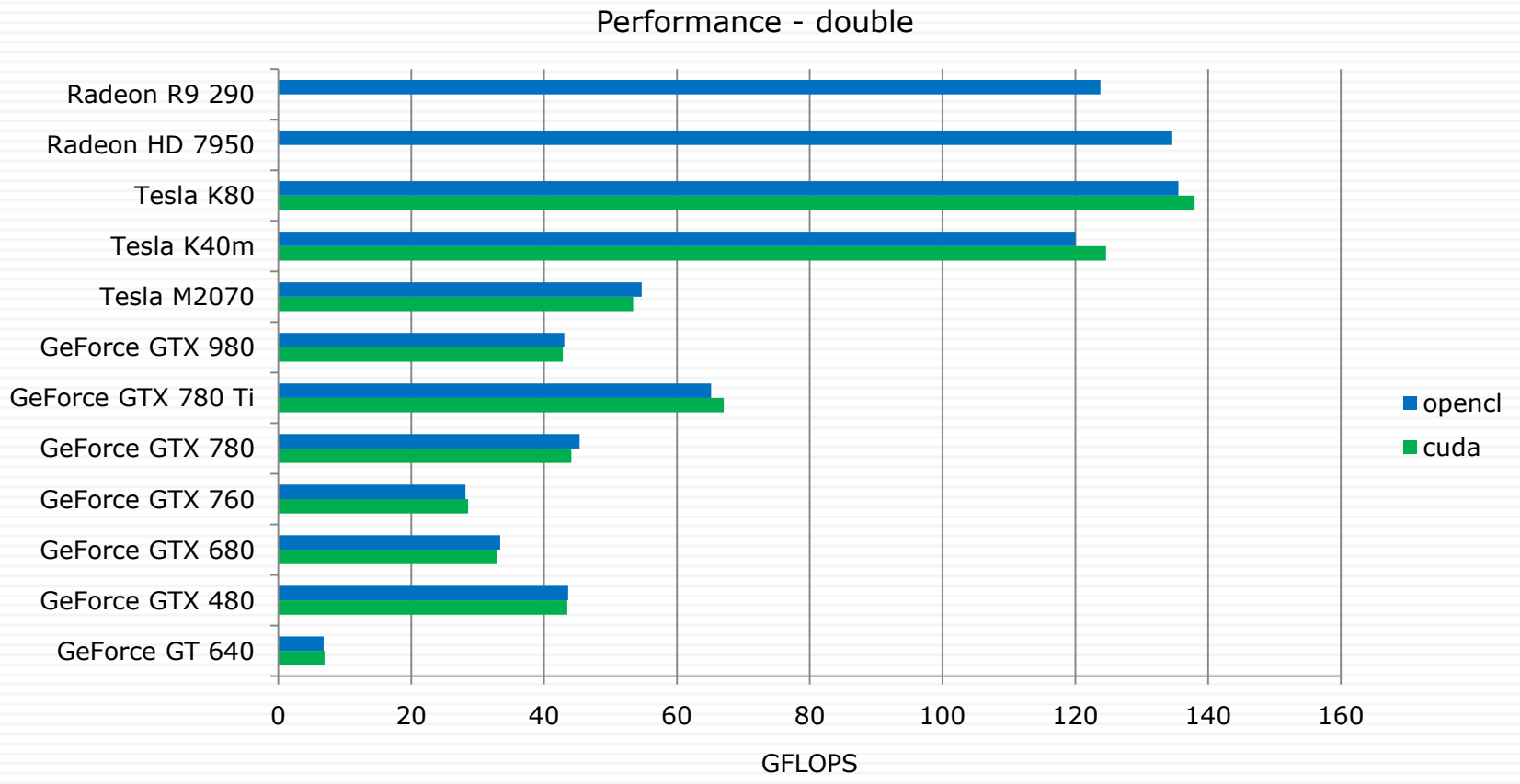


Percentage of peak performance

Performance



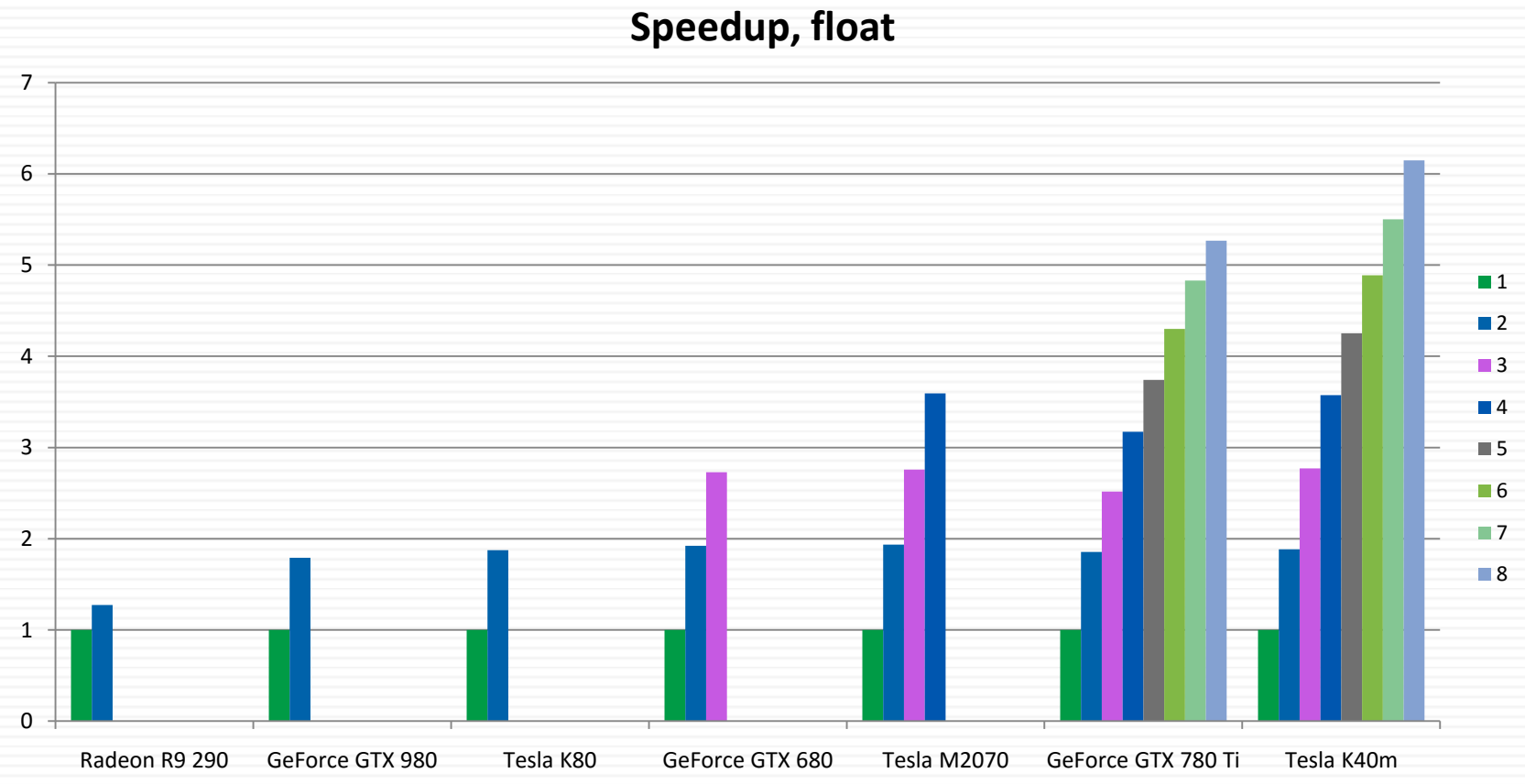
Performance



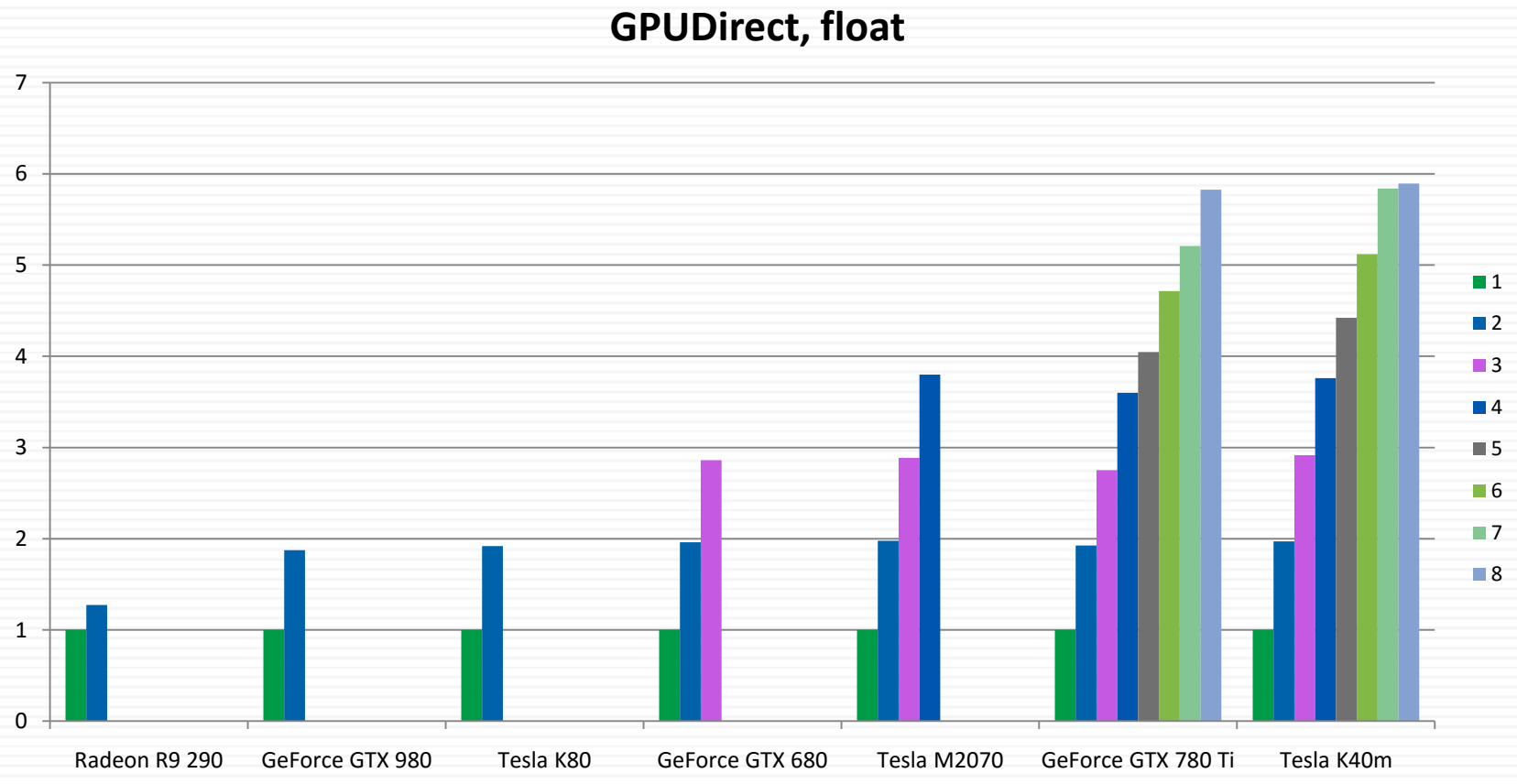
GPU parallelization

- Multiple GPUs
- Divide grid along axis Y
- Data exchanges between GPUs by adjacent grid nodes
- GPUDirect (only in CUDA) – exchange data by PCI Express bypassing CPU

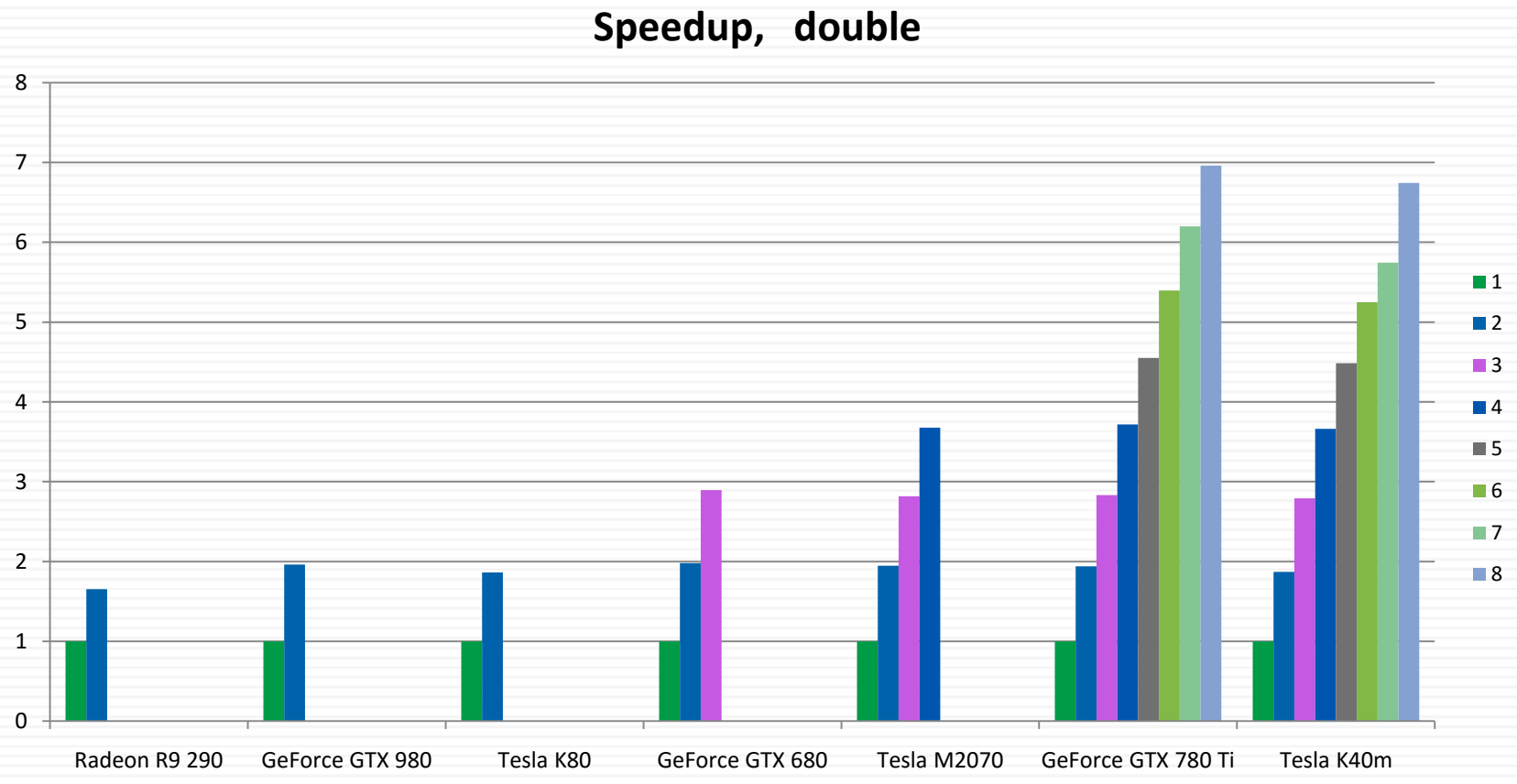
Speedup (number of GPUs)



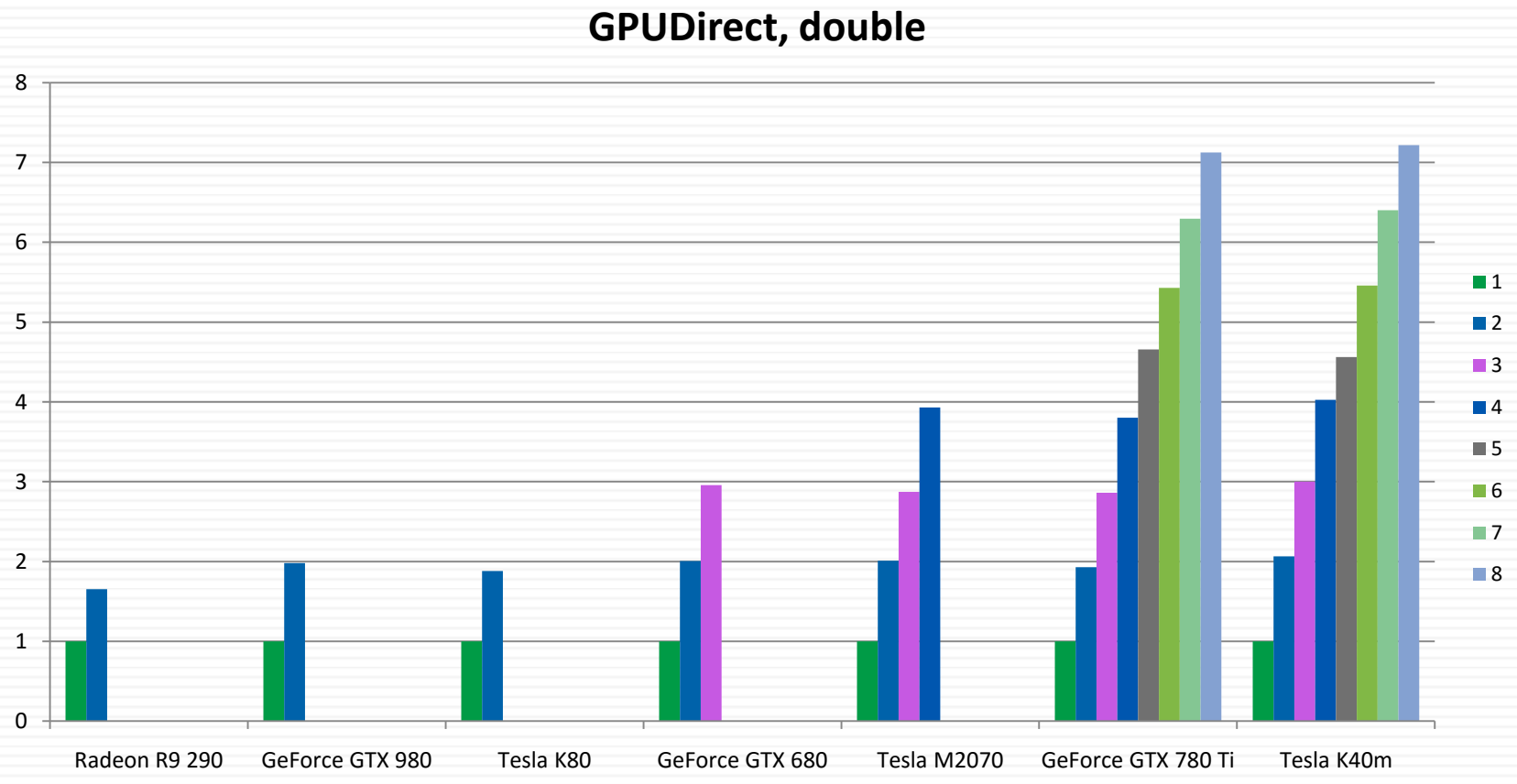
GPUDirect (except Radeon R9 290)



Speedup (number of GPUs)



GPUDirect (except Radeon R9 290)



Conclusion

- Speedup (single GPU compared with CPU):
  - Single-precision – up to **55** times (**GeForce GTX 780 Ti**)
  - Double-precision – up to **44** times (**Tesla K80**)
- Performance (single GPU):
  - Single-precision – up to **460** GFLOPS (**GeForce GTX 780 Ti**)
  - Double-precision - up to **138** GFLOPS (**Tesla K80**)
- Speedup (multiple GPU compared with single GPU):
  - Single-precision – up to **6.1** times (**Tesla K40m**)
  - Double-precision – up to **7.1** times (**GeForce GTX 780 Ti**)
- Increase in speedup with GPUDirect
  - Single-precision - **10**% on 8 **GeForce GTX 780 Ti**
  - Double-precision – **2.4**% on 8 **GeForce GTX 780 Ti**