GPU applications for modelling, migration, FWI and machine learning

CREWES 2021

Daniel Trad

December 2, 2021

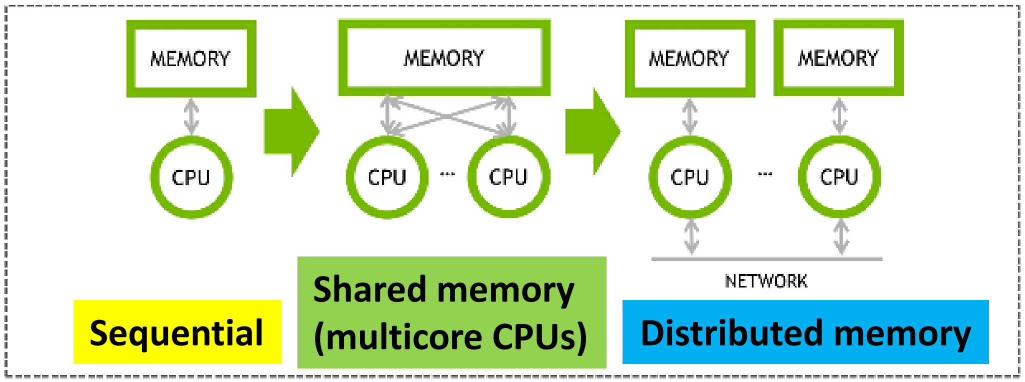
NSERC CRSNG
UNIVERSITY OF CALGARY
FACULTY OF SCIENCE
Department of Geoscience

Outline

- HPC models and different philosophies
- CUDA implementations for finite differences
- Examples: Modelling and RTM
- Examples for multigrid FWI
- Applications to Machine Learning
- The future

HPC evolution

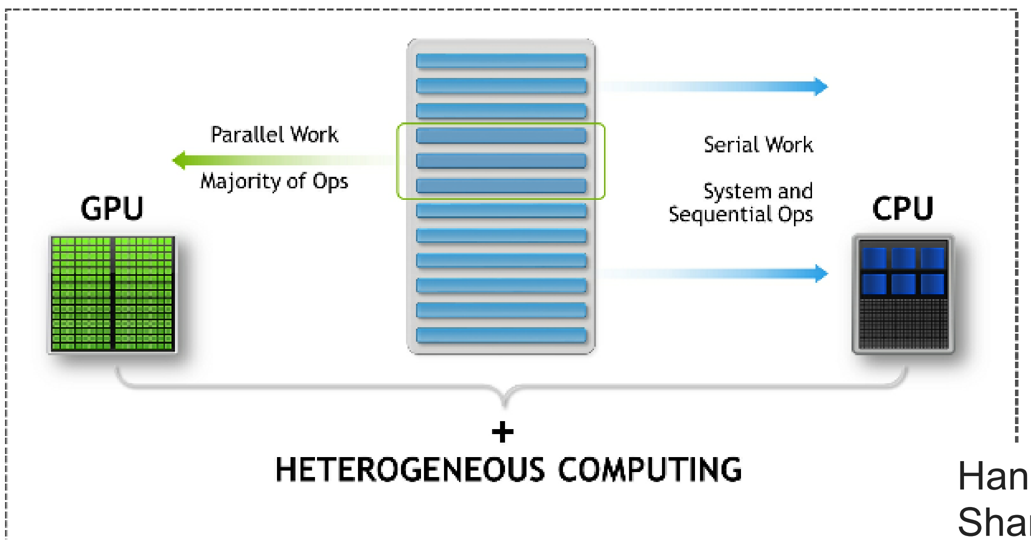
CPU based



HPC SYSTEM EVOLUTION

Latency reducing (mostly by cache)

CPU+GPU heterogeneous

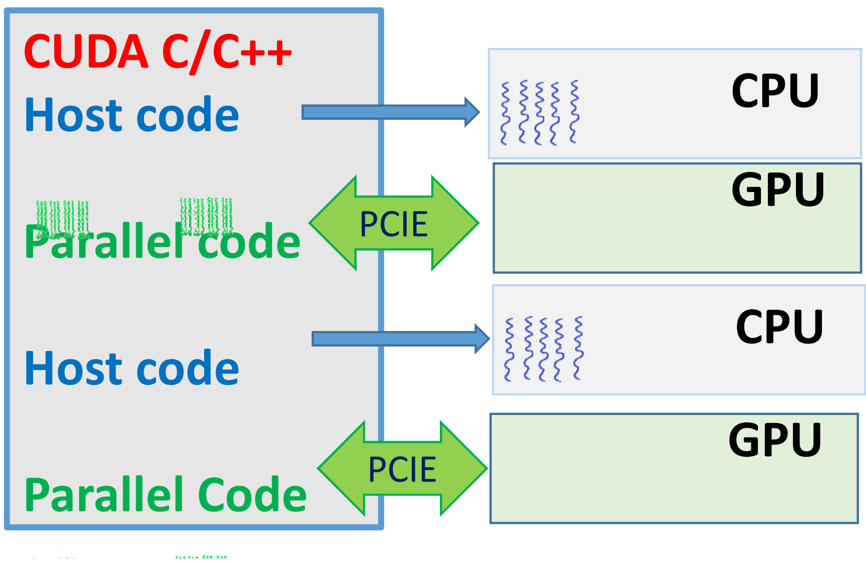


Latency hiding (multithreading)

Han, Jaegeun, and Bharatkumar Sharma, 2019

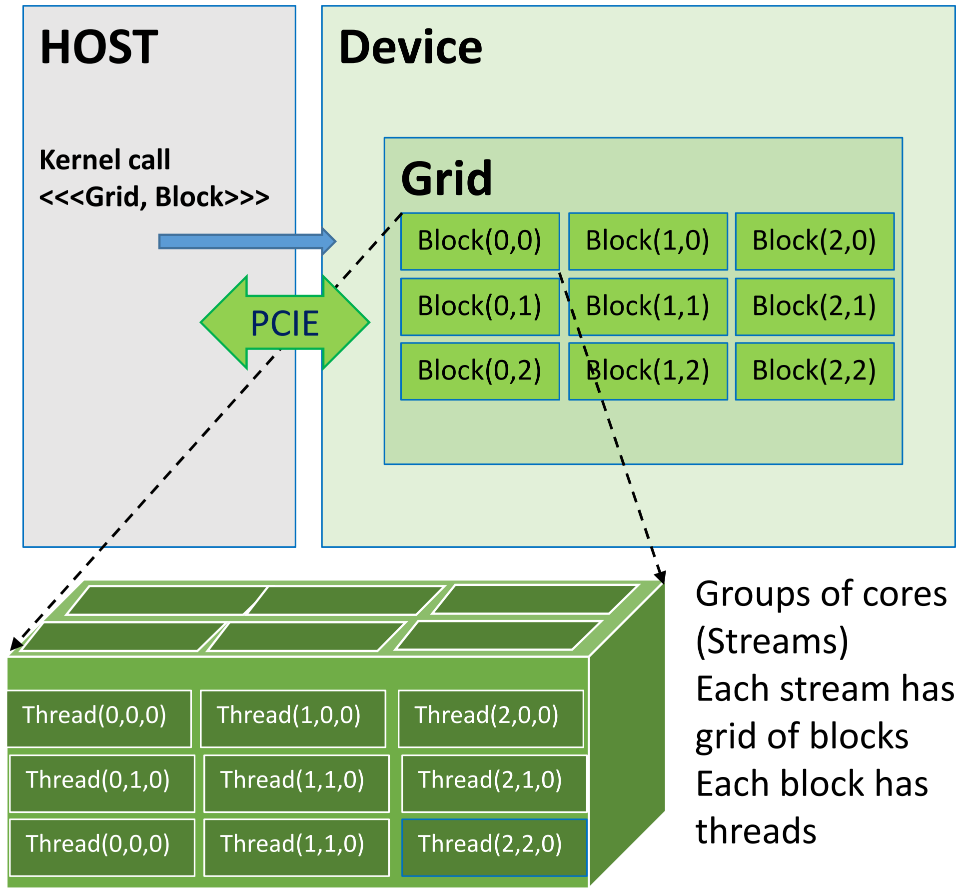
Programming for GPUs: Host-Device model

Host/Device Heterogenous Dataflow



Distinction between host (CPU) and device (GPU). They have different code and different memory

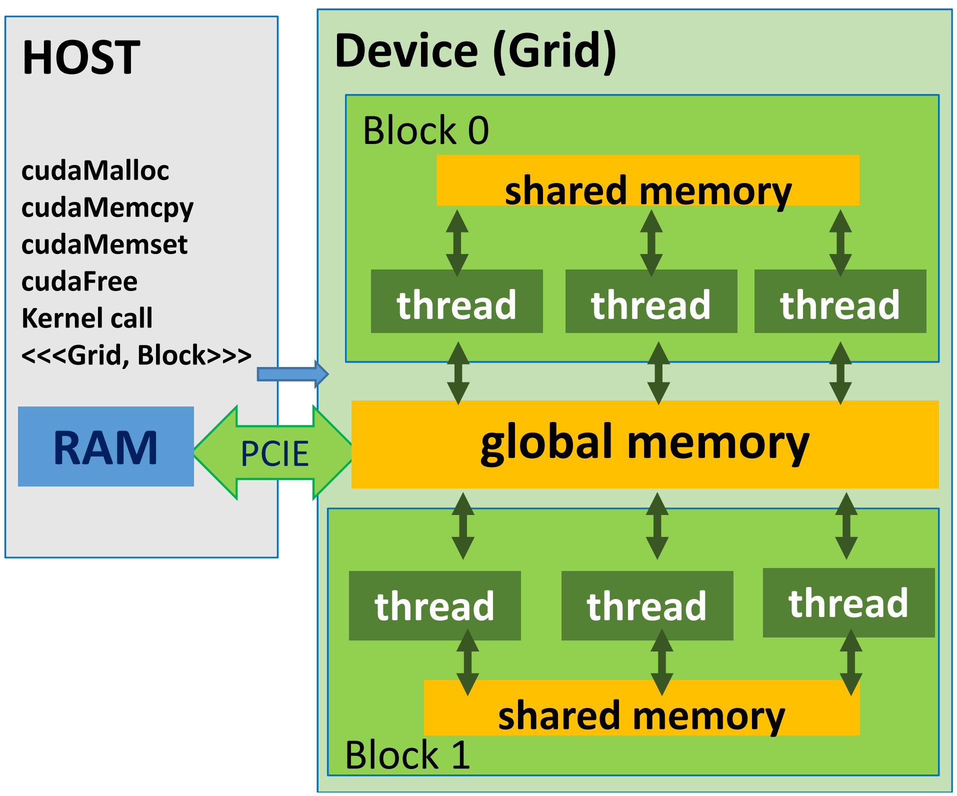
Device hierarchy Grid/Block/Thread



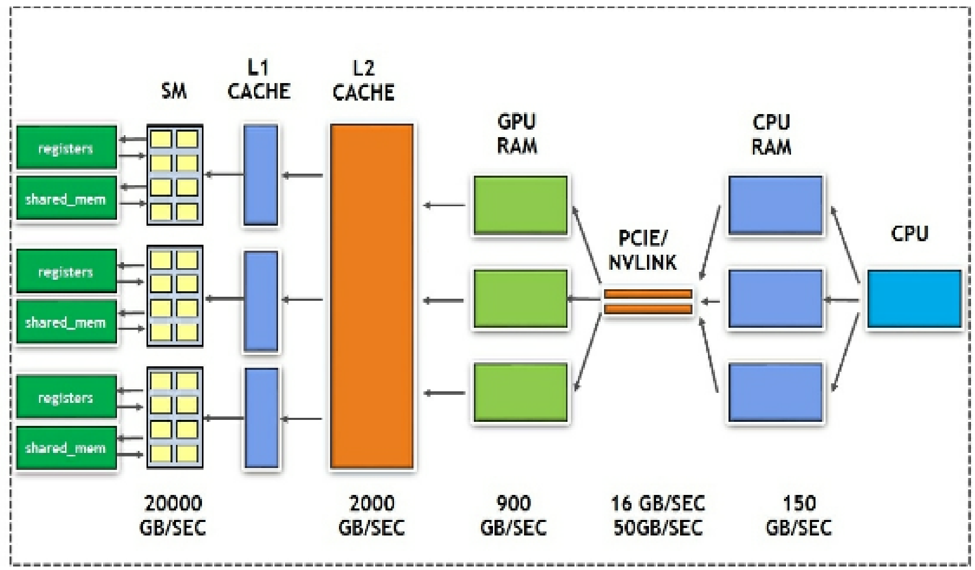
Groups of cores (Streams)
Each stream has a grid of blocks
Each block has threads

Programming for GPUs: Memory hierarchies

Device hierarchy Grid/Block/Thread

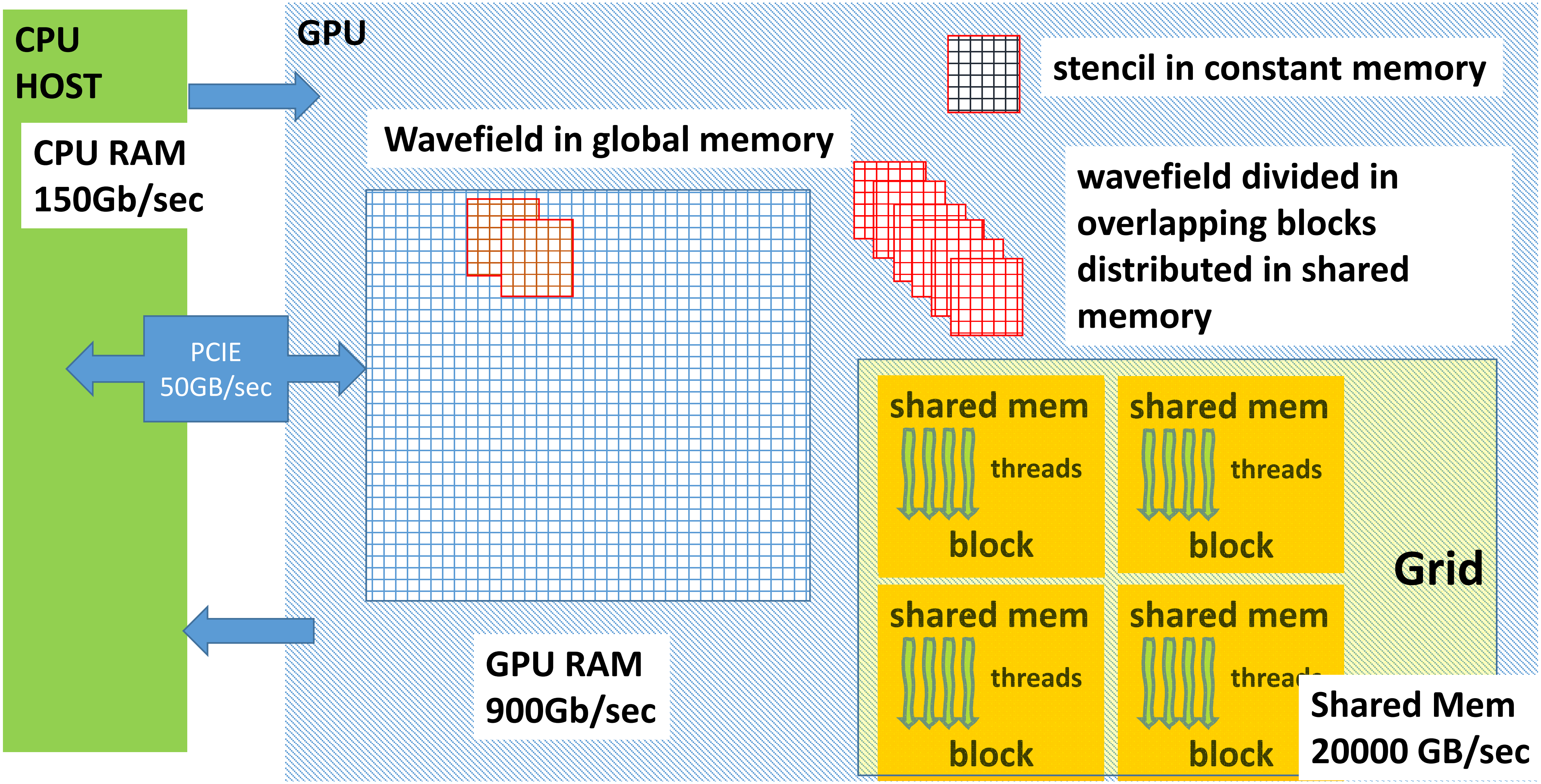


Threads can communicate very fast inside the same block, but much slower across blocks.

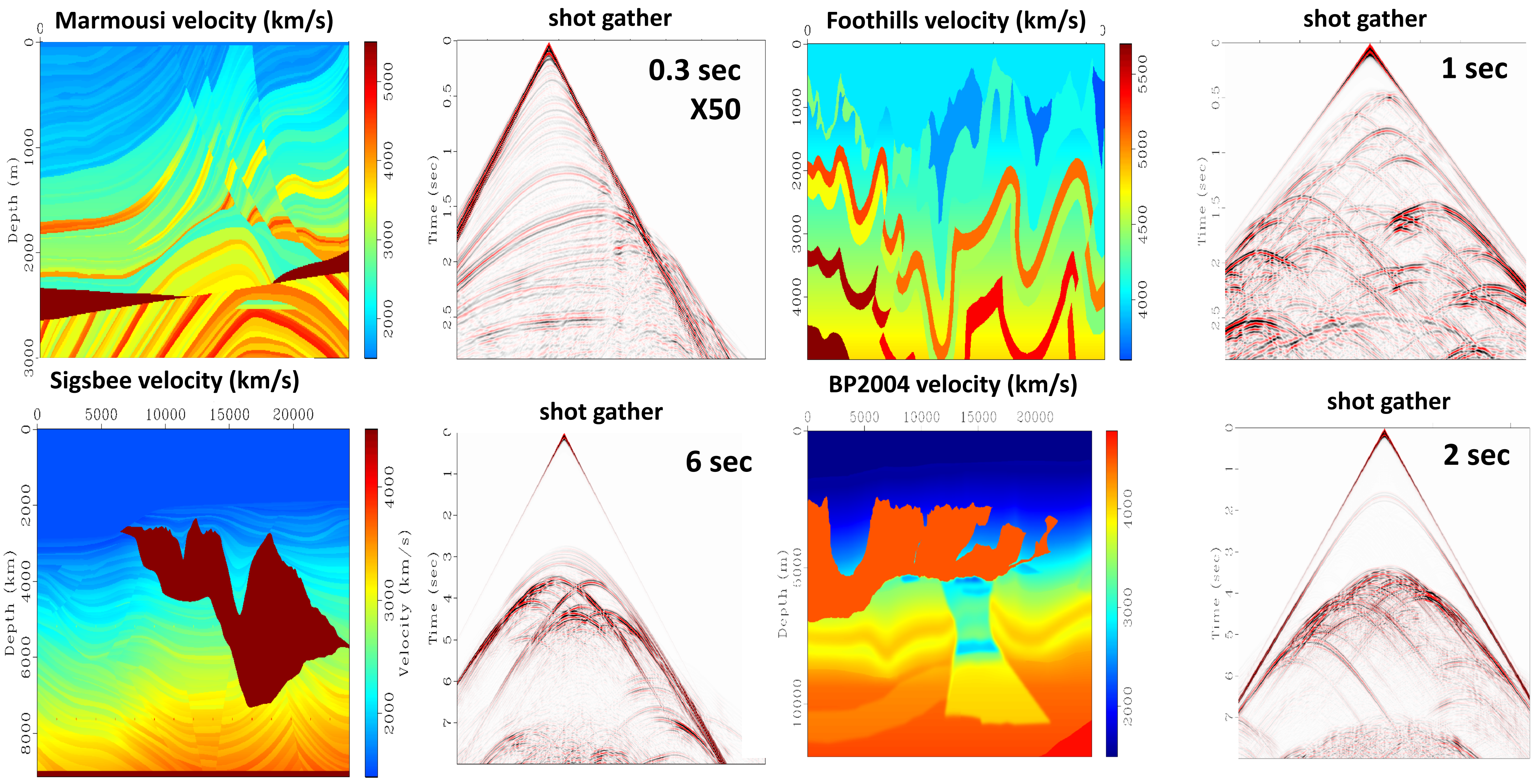


Different memory bandwidth for each type of memory (notice it differs by orders of magnitude)

Programming for GPUs: Convolutional Pattern



Finite difference (8^{th} order, 25Hz) running times per shot (RTX2070)



Computing times for Finite Differences

Modeling times with OPENMP, OPENMPI and CUDA (same desktop)

Second Order:

OPENMP, 12 threads, 10 shots → 18 sec

CUDA, RTX2070, 10 shots → **3** sec

Fourth Order:

OPENMP, 12 threads, 10 shots → 220 sec

OPENMPI, 12 threads (1node=1thread), 10 shots → **60** sec

CUDA, RTX2070, 10 shots → **3** sec

Eight Order:

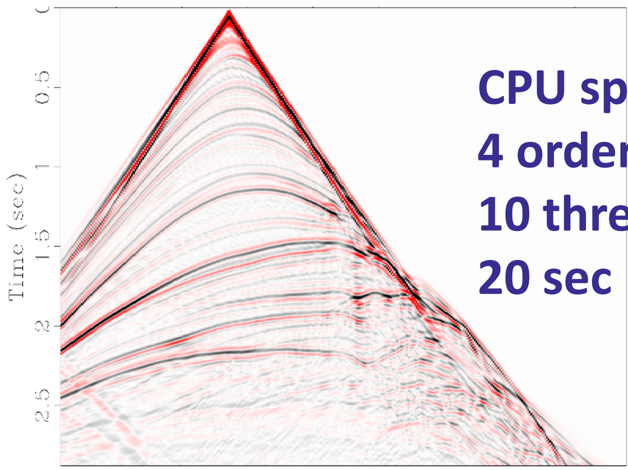
OPENMP, 12 threads, 10 shots -> 800sec

OPENMPI, 12 threads (1node=1thread), 10 shots → **300** sec

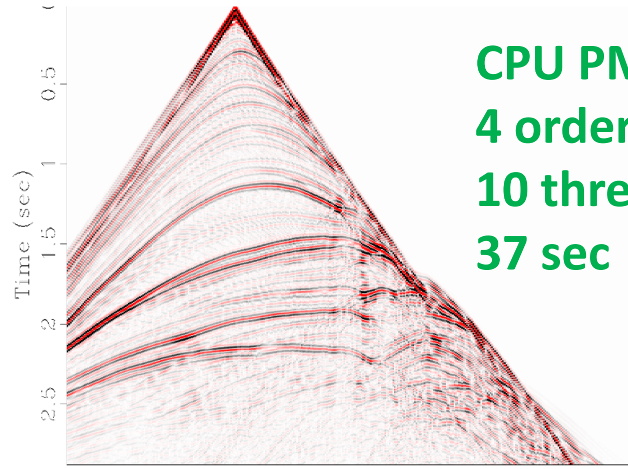
CUDA, RTX2070, 10 shots → **3** sec

All times measured on the same desktop computer:
CPU hexacore i7, 2016
GPU RTX2070

Comparison CPU/GPU standard/PML (Marmousi)

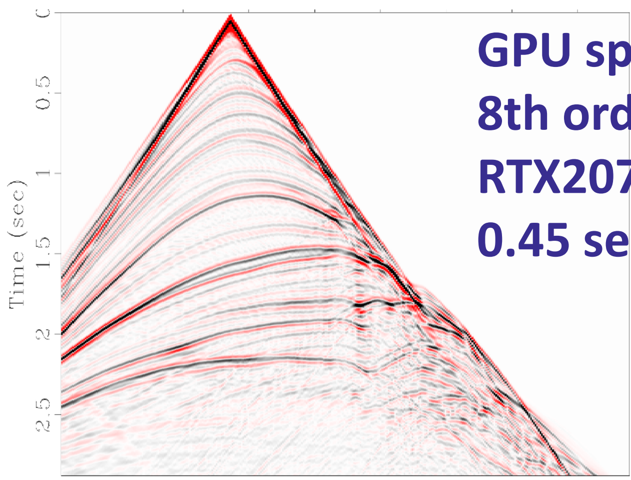


CPU sponge
4 order
10 threads
20 sec

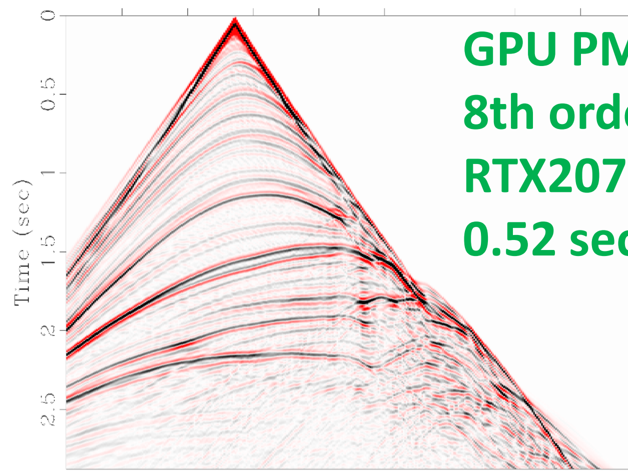


CPU PML
4 order
10 threads
37 sec

Perfectly Matched Layer
Propagates 4 waves
Px, Py, Vx, Vz
4 times more work
same time with GPU
double time with CPU



GPU sponge
8th order
RTX2070
0.45 sec

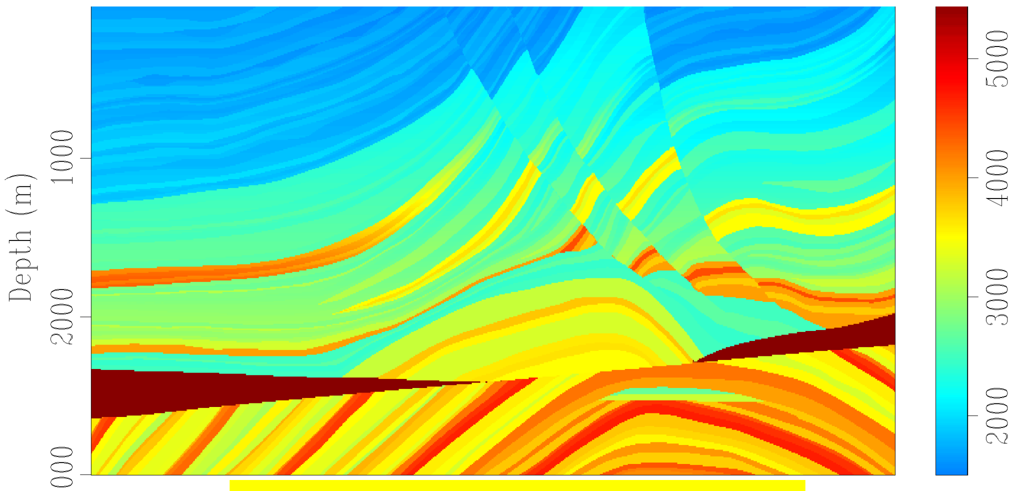


GPU PML
8th order
RTX2070
0.52 sec

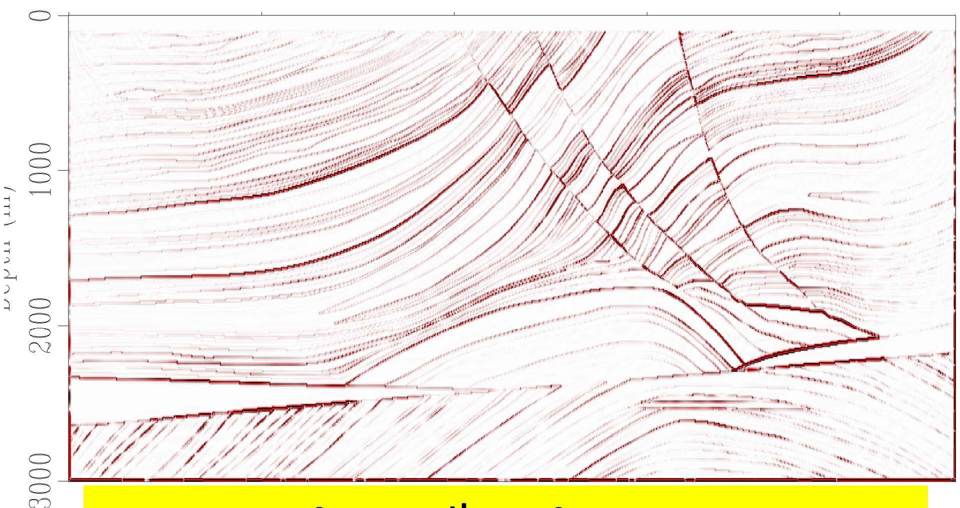
Superlinear scaling ?
GPU resources are not fully used.

RTM Marmousi

Comparison same machine
OPENMP 10 threads ~ 10 min.
MPI 10 nodes (1 thread each) ~ 4 minutes
GPU ~ 11 seconds (30X)



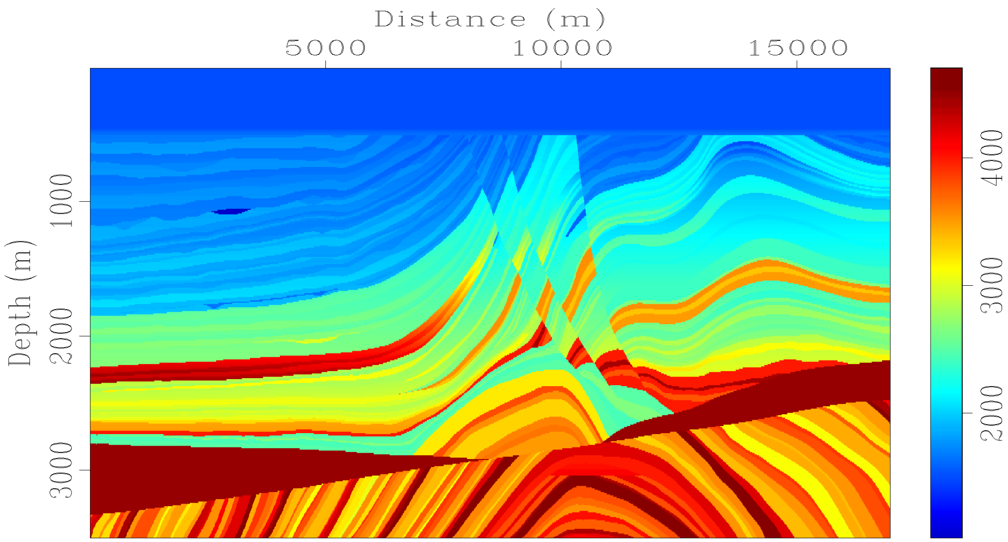
Velocity (376x1151 cells)



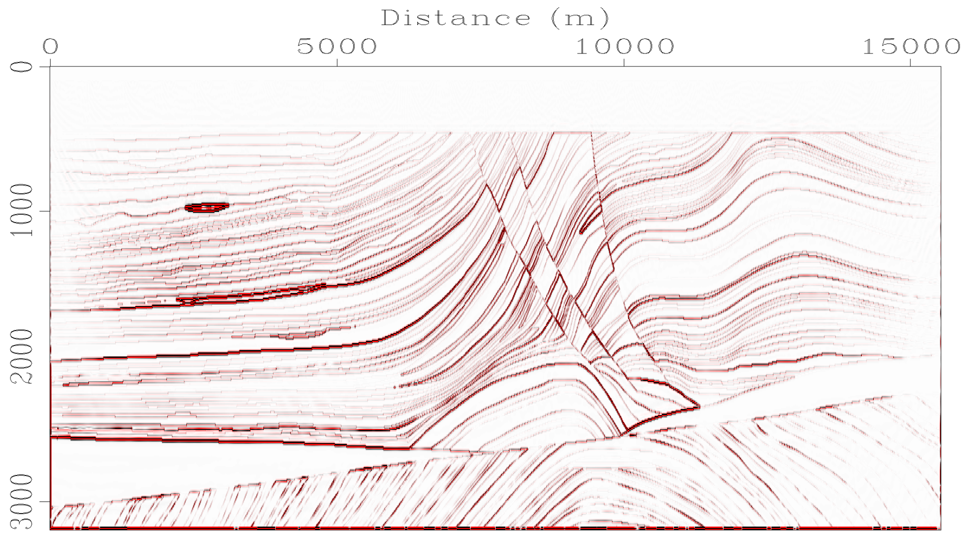
RTM, 12 shots 4th order space, 20 Hz

RTM Marmousi II

Computing time: 25 seconds



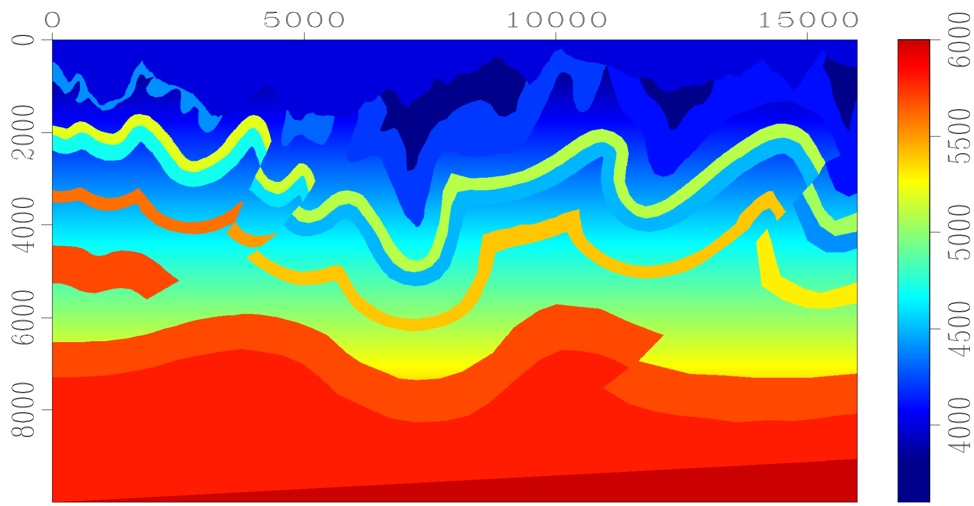
Velocity (400x1942 cells)



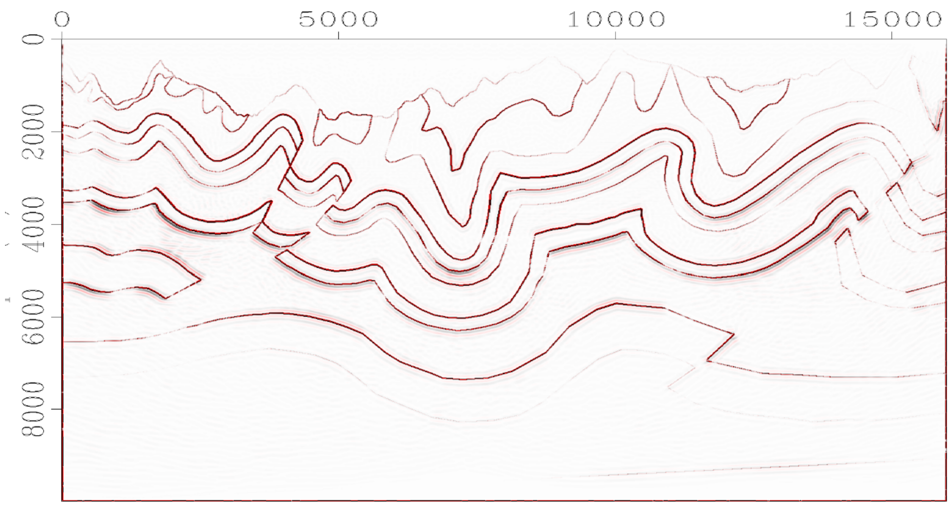
RTM, 25 shots 4th order space, 20 Hz

RTM Foothills, 50 shots, ~ 3 minutes

Computation time ~ 3 minutes

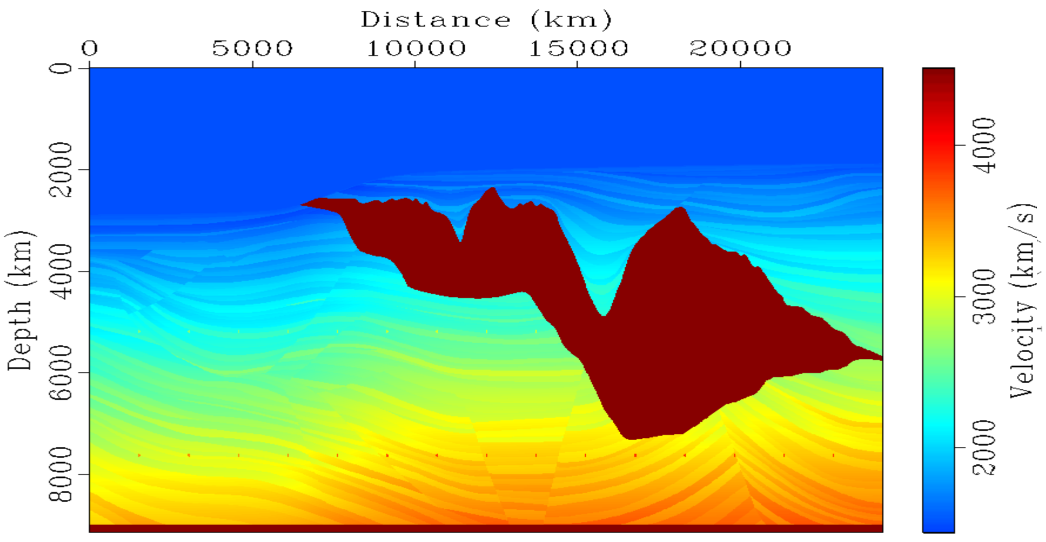


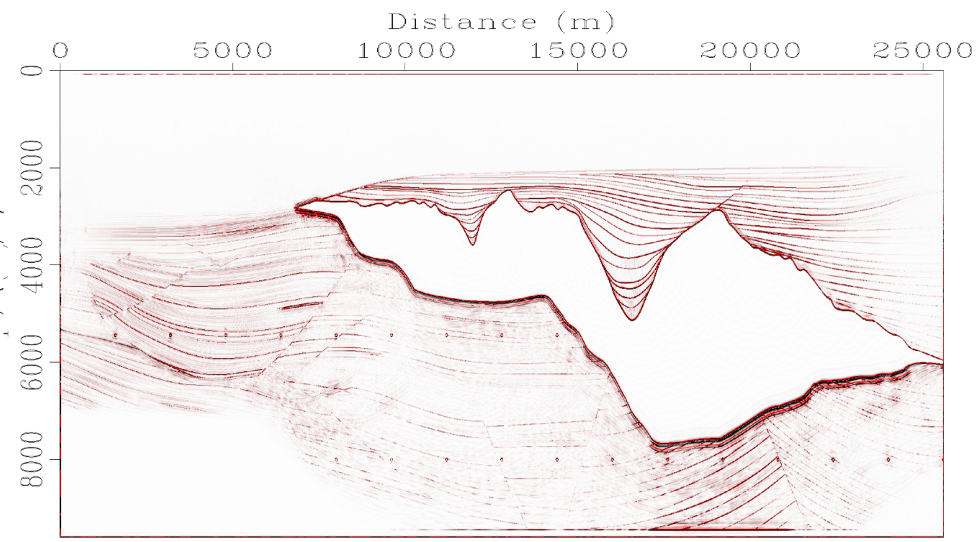
Velocity (1000x1600 cells)



RTM, 50 shots 4th order space, 20 Hz

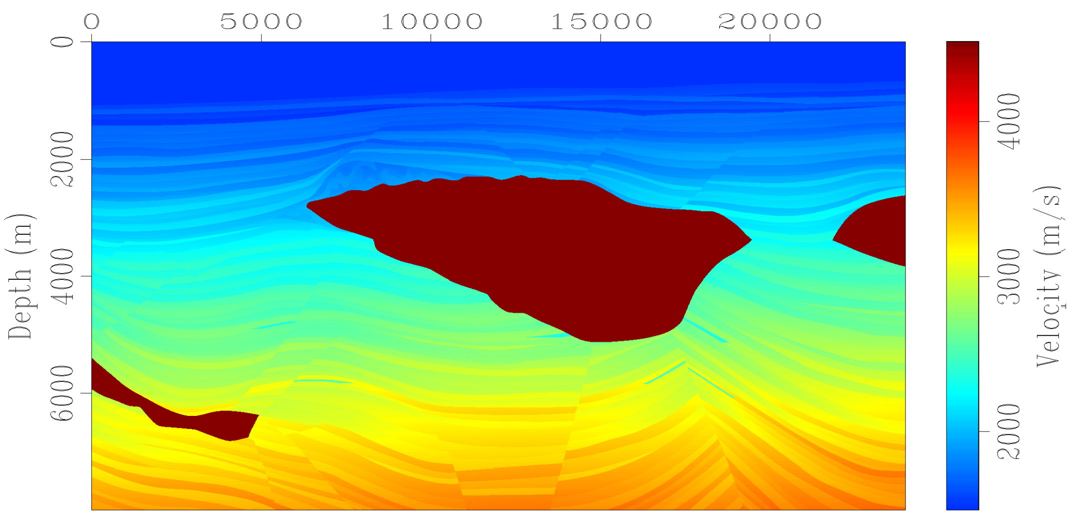
Sigsbee 50 shots, ~ 22 minutes

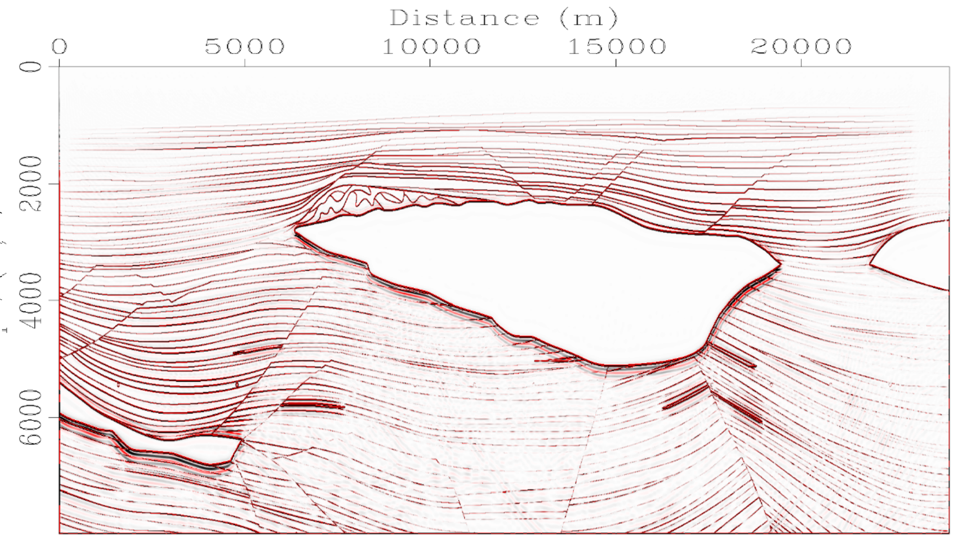




model size 1201x3201, 8th order in space 20Hz.

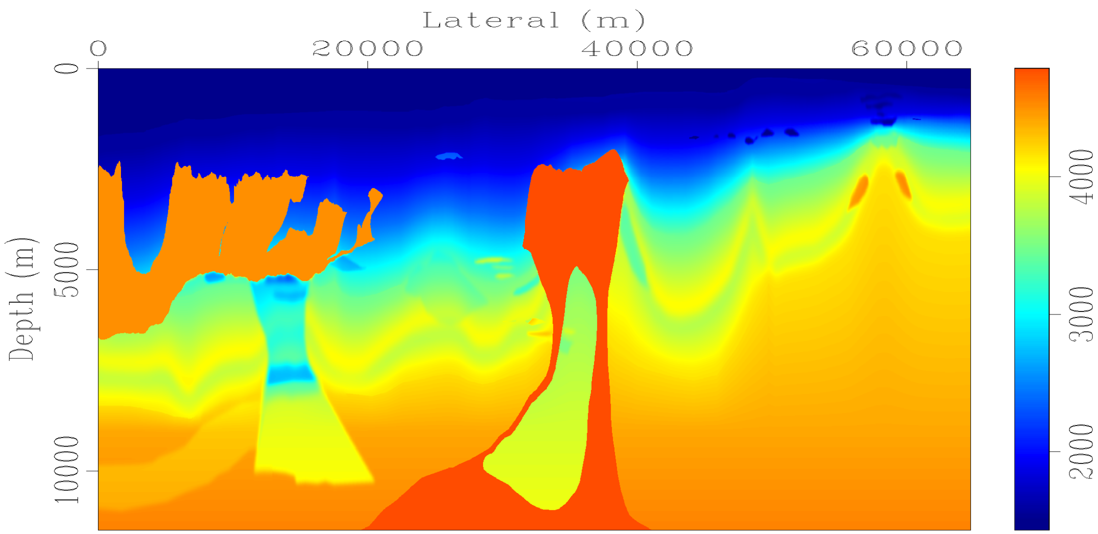
RTM Pluto, 50 shots, ~ 16 minutes

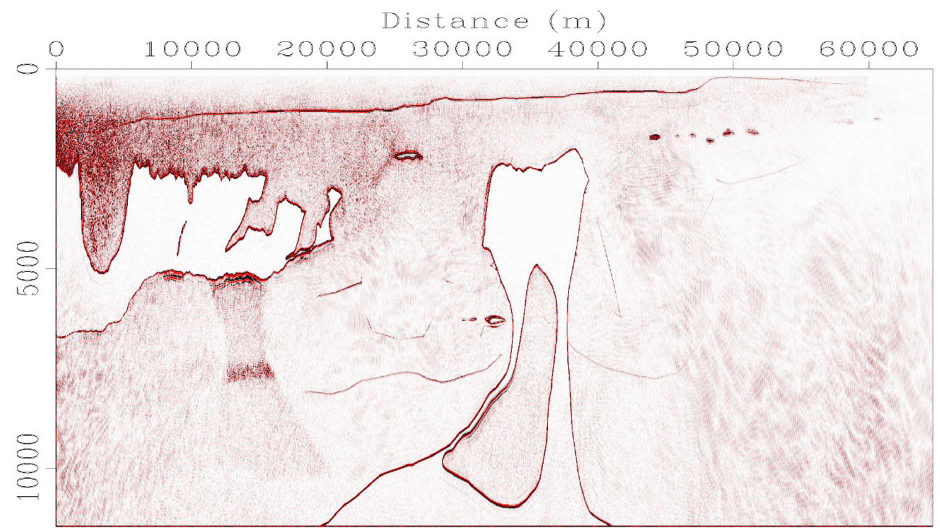




model 1000 x 3000, 4th order space, 20 Hz

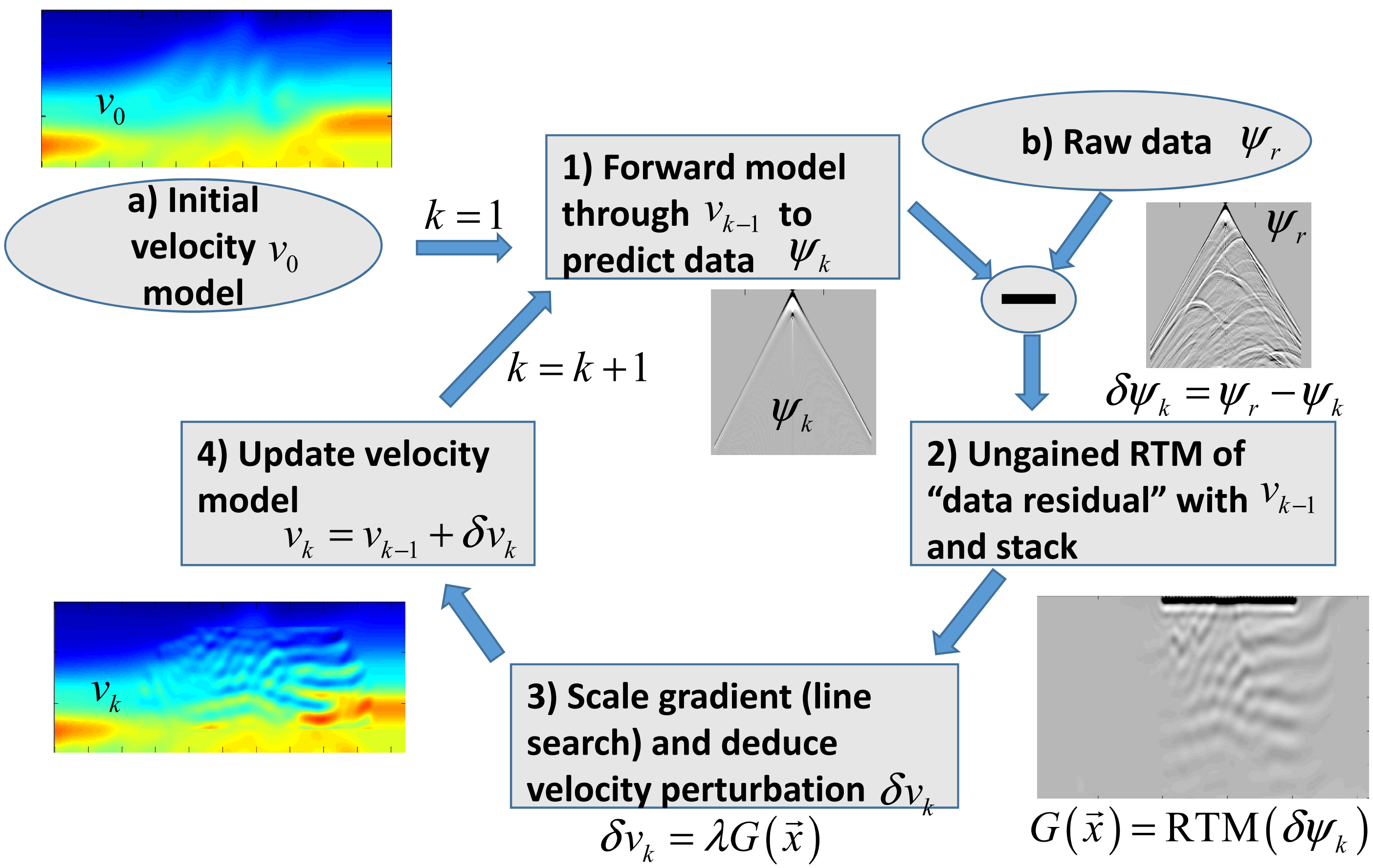
BP2004 100 shots, ~ 1hour 40 minutes





model size 956x5395, 4th order in space 20Hz.

Full Waveform Inversion Basics (Inversion)



$$J = \left\| d_{predicted} - d_{acquired} \right\|^2$$

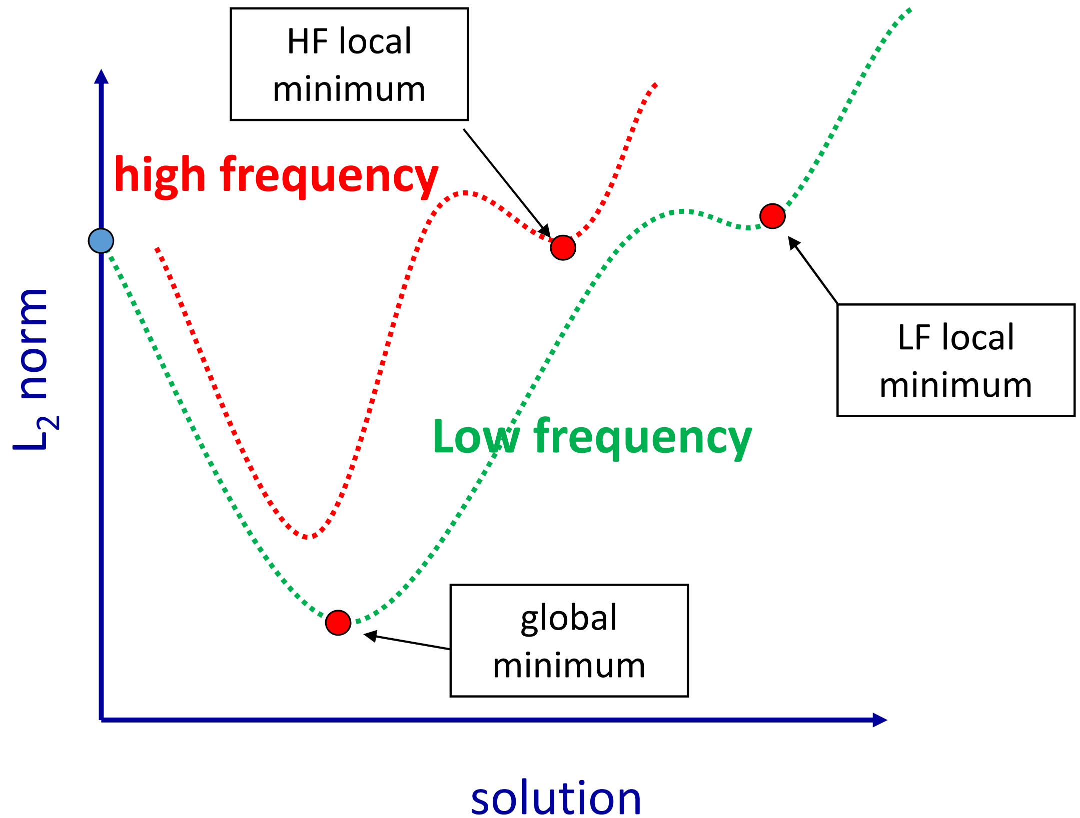
$$\mathbf{d}_{\text{predicted}} = \mathbf{L}(\mathbf{v}_{\text{iter}})$$

$$\mathbf{v}_{\text{iter}} = \mathbf{v}_{\text{iter}-1} + \alpha\mathbf{\Delta v}$$

$$\Delta v = RTM(Residuals)$$

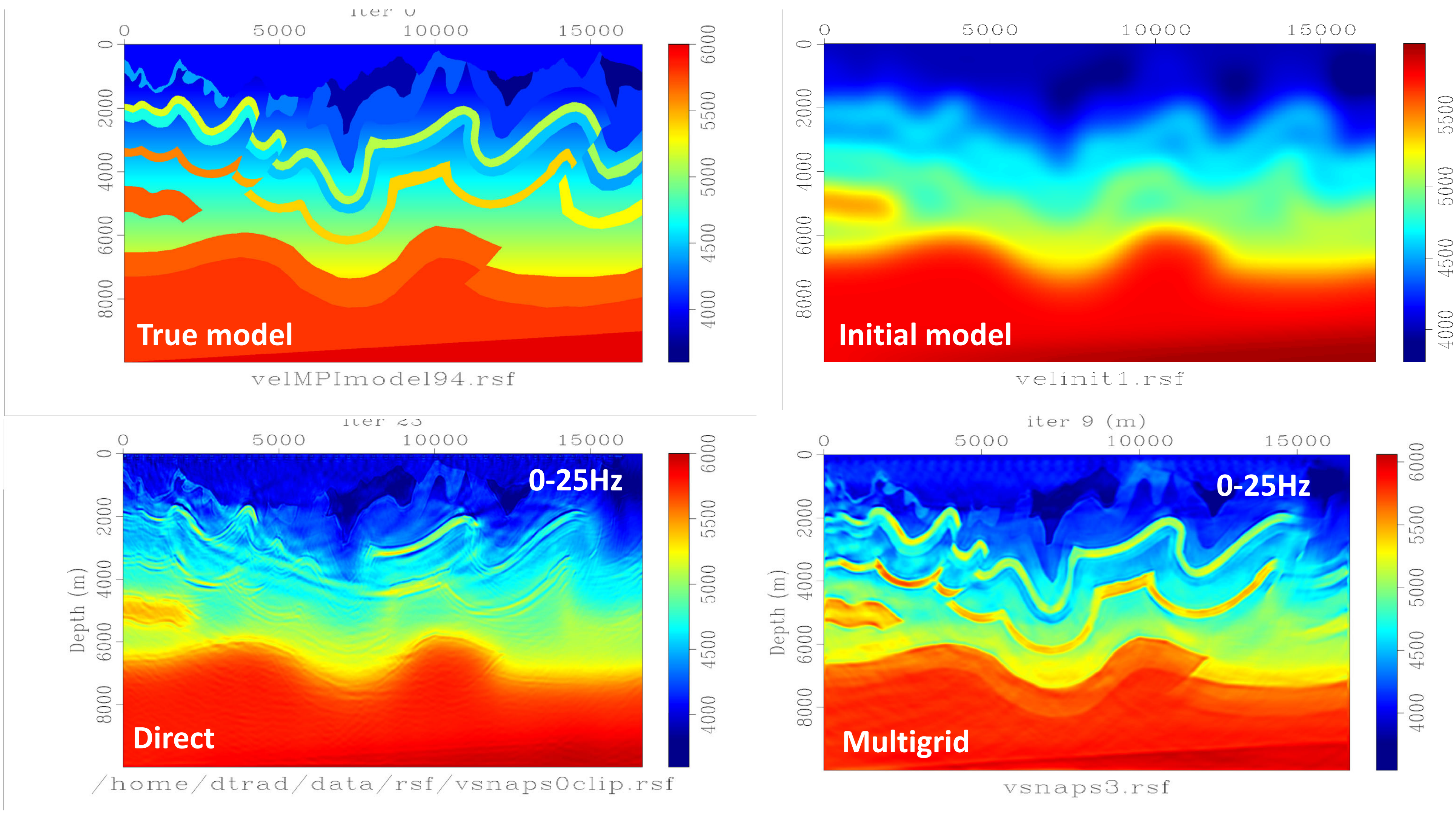
The cycle skipping problem

cost functions for different frequencies

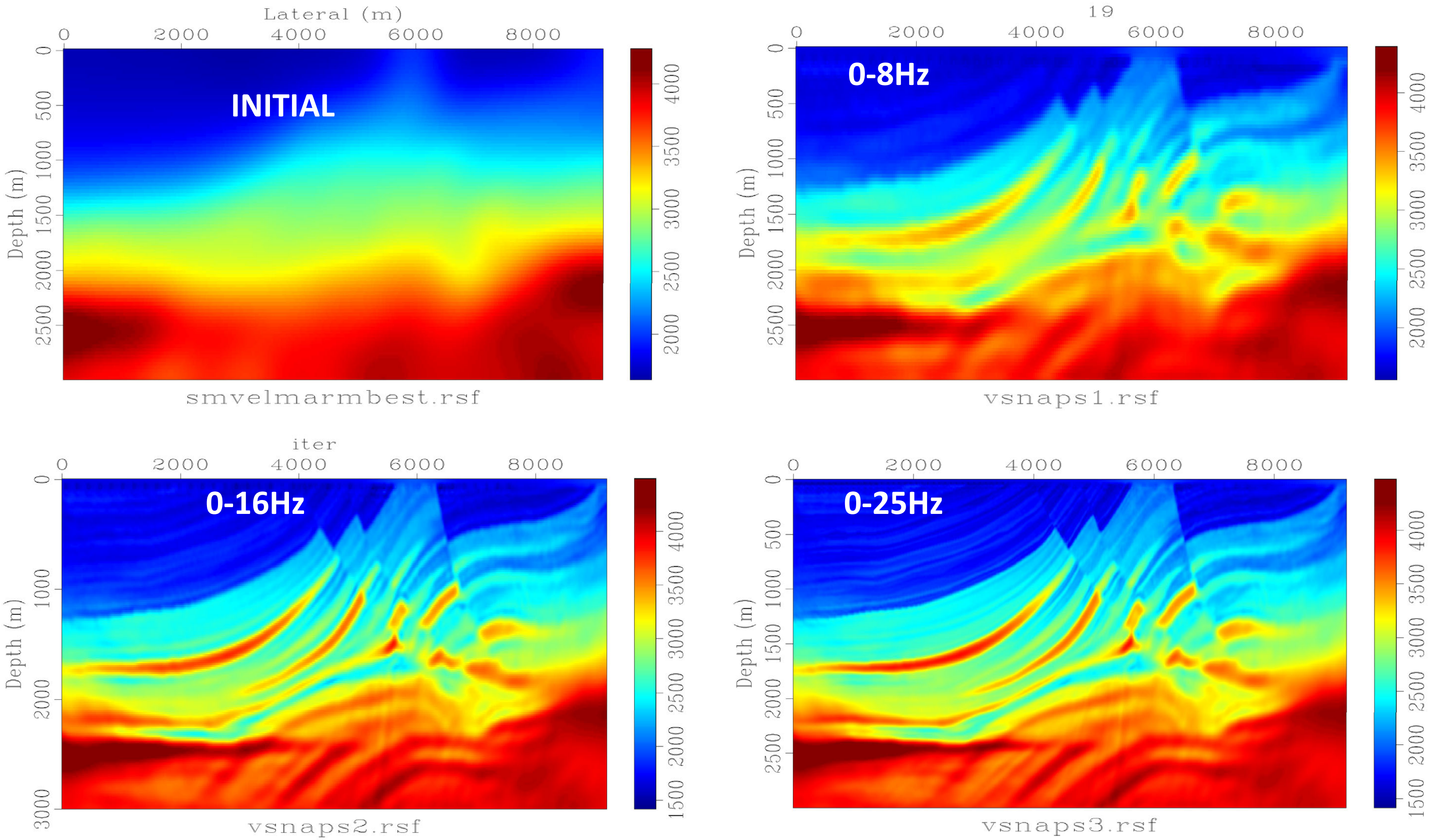


High frequencies require better initial model

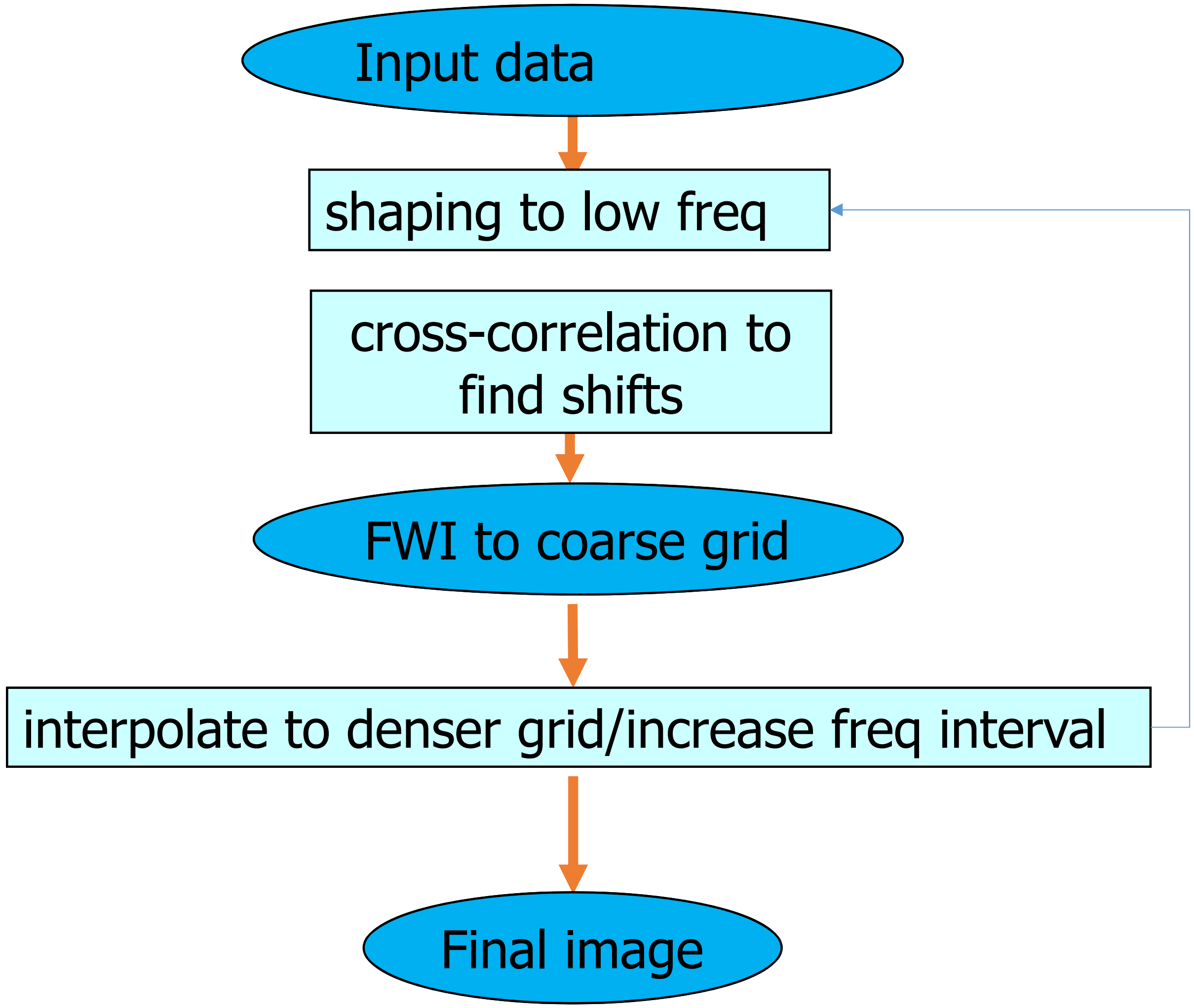
Direct vs Multigrid for foothills model



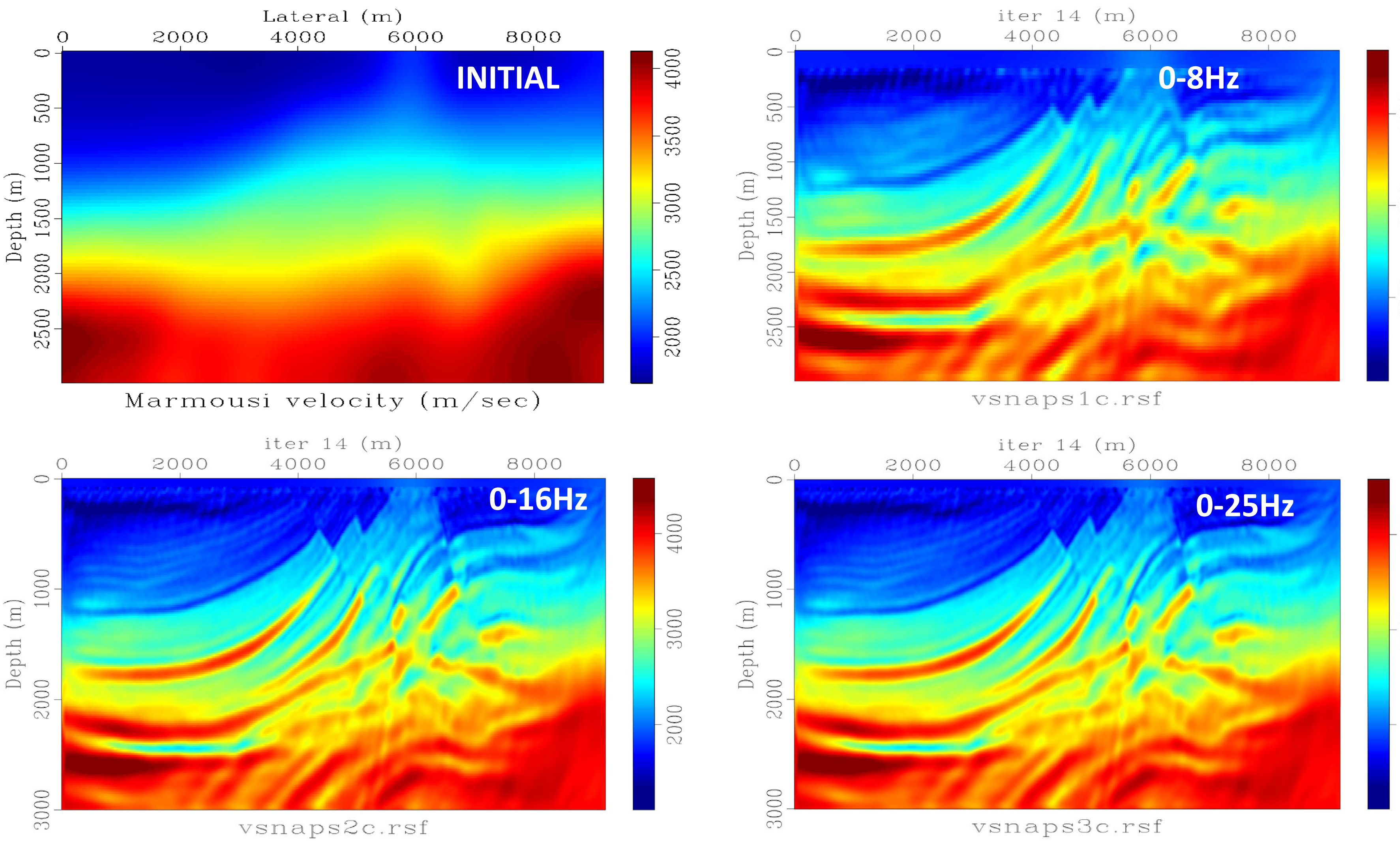
Multigrid stages for Marmousi Model (inverse crime)



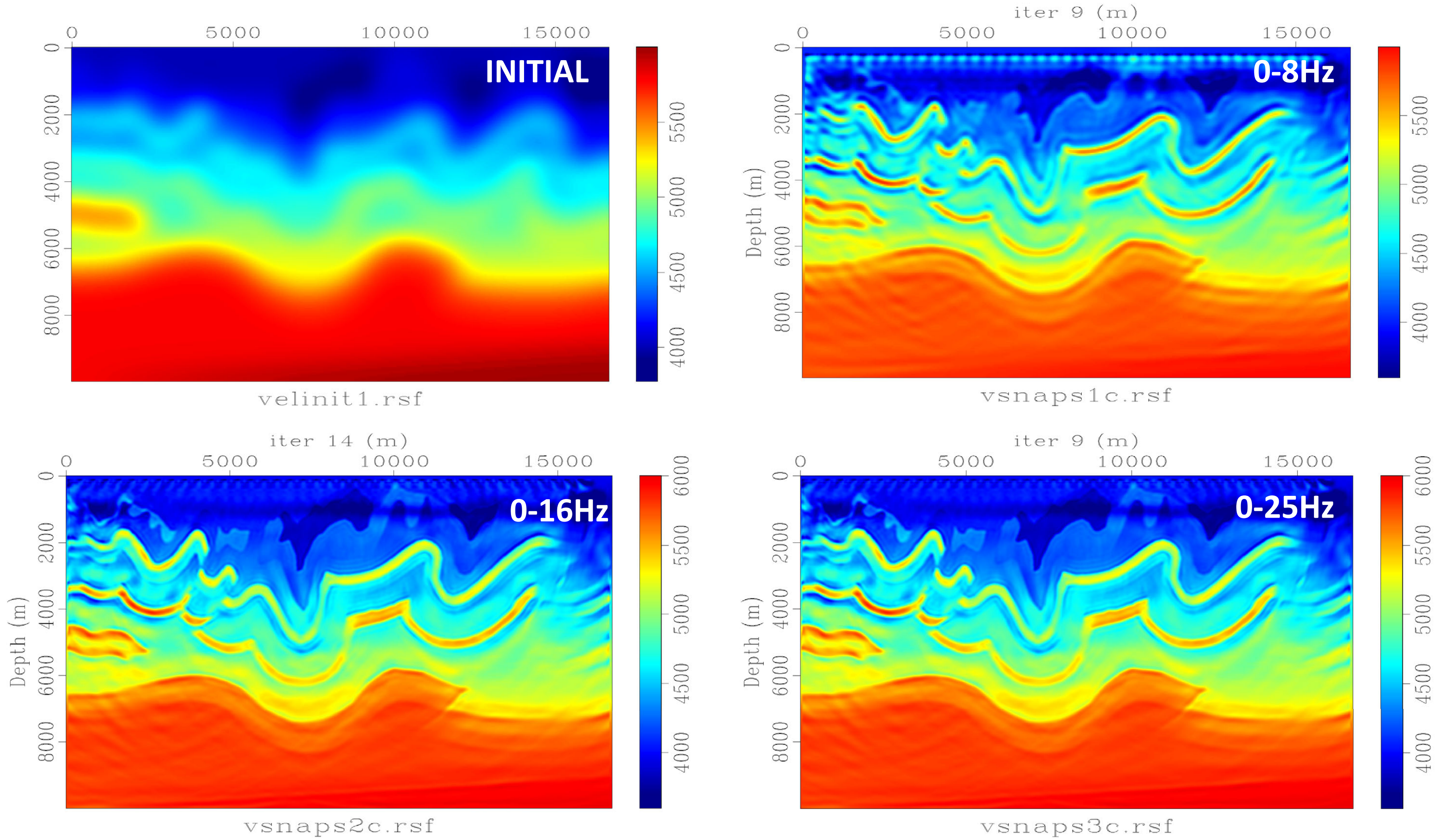
Dataflow III – Multigrid with shaping filter and cross-correlation shifts.



Multigrid for Marmousi model: CUDA and multigrid (no inverse crime)



Multigrid stages for foothills model: CUDA (no inverse crime)

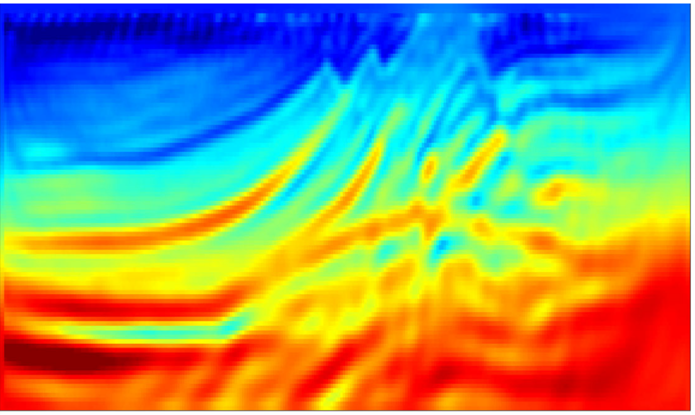


GPU Computing times multigrid FWI

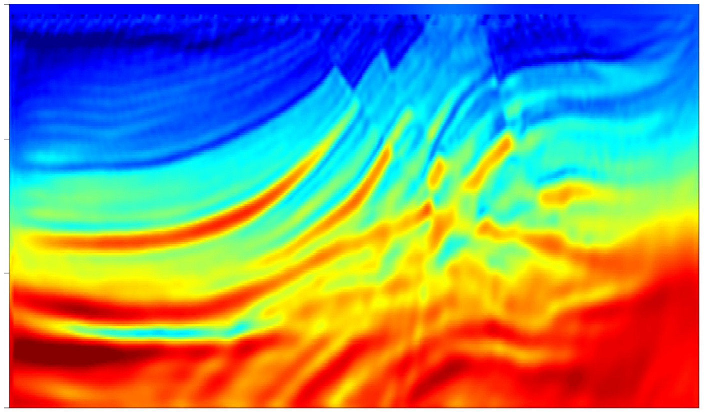
Table 1. GPU Computation times in a regular desktop (for the example in Figure 20)

model size	cell size	time steps	nshots	iterations	time
96 x 288	32, 32	4600	40	15	263 secs
188 x 576	16, 16	4600	40	15	410 secs
376 x 1151	8, 8	4600	40	5	366 secs

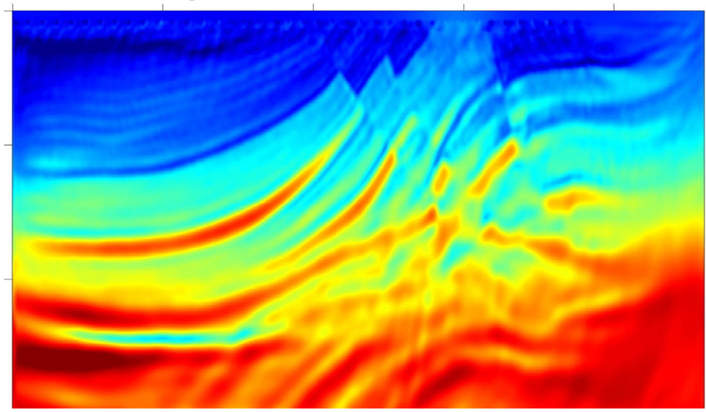
Stage 1 - 0-8Hz 32x32



Stage 2 - 0-16Hz 16x16



Stage 3 - 0-25Hz 8x8



Computing times multigrid FWI CPU vs GPU

Table 2. CPU Computation times (MPI-Multithread) in a 10 nodes cluster

model size	**cell size**	**time steps**	**nshots**	**iterations**	**time**
96 x 288	32, 32	2800	40	20	196 secs
188 x 576	16, 16	2800	40	20	576 secs
376 x 1151	8, 8	4600	40	20	4481 secs

Table 3. GPU Computation times in a regular desktop with a RTX2070

model size	**cell size**	**time steps**	**nshots**	**iterations**	**time**
96 x 288	32, 32	2800	40	20	206 secs
188 x 576	16, 16	2800	40	20	327 secs
376 x 1151	8, 8	4600	40	20	1466 secs

Superlinear scaling in multigrid FWI

CPU scaling, expected (from stage 1) and real

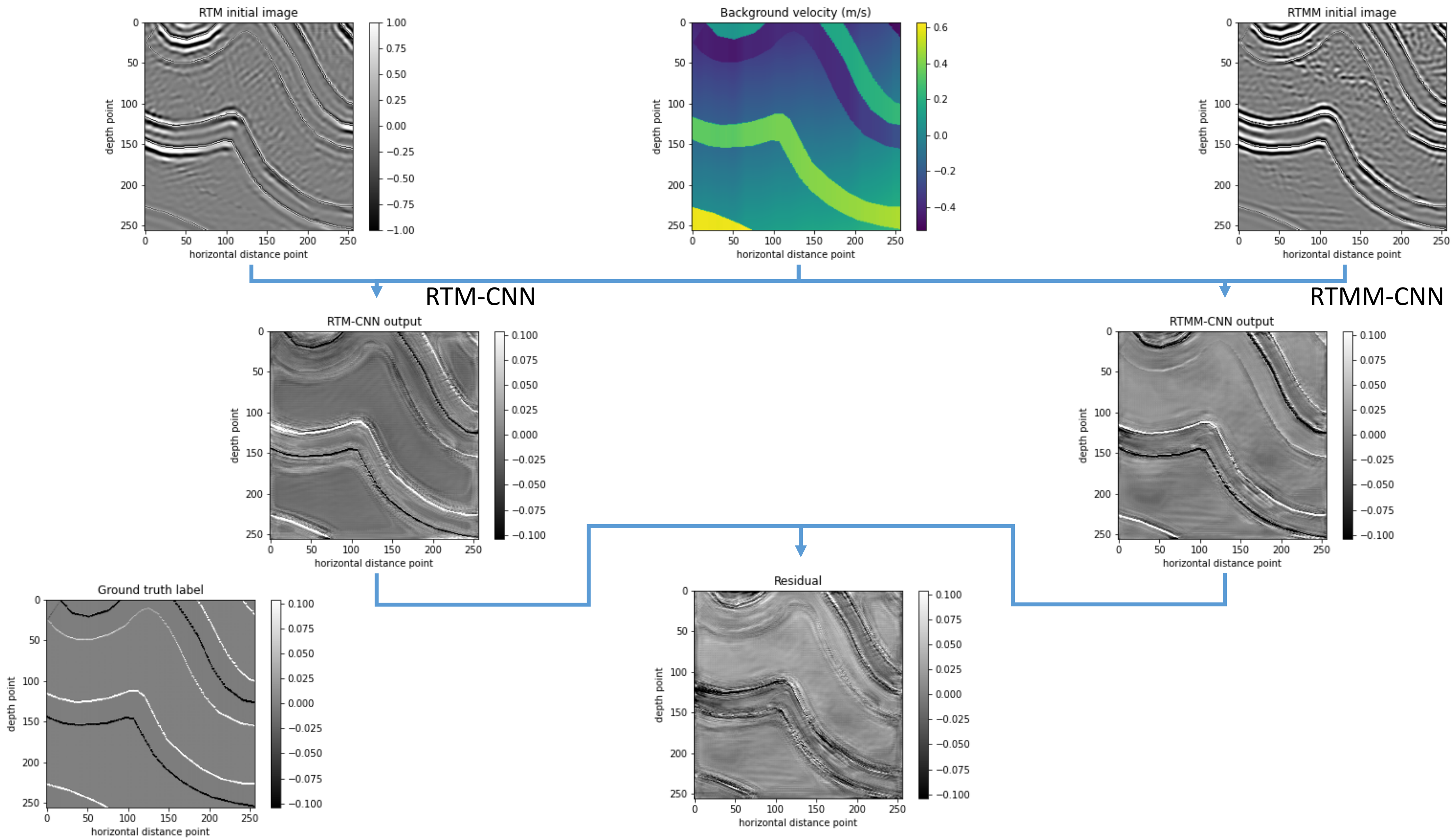
all times in seconds

stage	time expected	time real	ratio expected	ratio real
1	–	196	–	–
2	784	576	4	2.9
3	5100	4600	26	23.5

GPU scaling, expected (from stage 1) and real

stage	time expected	time real	ratio expected	ratio real
1	–	206	–	–
2	824	327	4	1.6
3	5100	1466	26	3.6

Applications to Machine Learning



The Future

Immediate future (next report), expanding to:

- 3D
- anisotropy
- Near surface and topography
- Elastic. Next year meeting.

Game changer technologies:

- Fourier Neural Operators: both Nvidia and Microsoft claim enormous speedups for modelling. We are trying for wave equation. Still it needs training. Probably 10X faster than this work for fine grids (after training).

SUMMARY

GPU-Finite difference, implemented correctly, 100x speedups.

Under-used resources lead to superlinear speedup

Permit better approximations without computational penalty.

Applications for modeling, RTM, FWI and training neural networks.

Parallelization essential component of research.

Ackowledgments

Support:

- CREWES sponsors
- Canadian SEG (Chair in Exploration Geophysics)
- Natural Sciences and Engineering Research Council of Canada

special thanks to Penliang Yang, Sam Gray and Torre Zuk