Parallelization Strategy for Wavefield Simulation with the Elastic Iterative Solver

Mikhail Belonosov, ***Vladimir Cheverda,*** Victor Kostin, Dmitry Neklyudov

Institute of Petroleum Geology and Geophysics SB RAS &

Aramco Research Center - Delft, Aramco Overseas Company B.V.

Novosibirsk, Russia & Delft, the Netherlands

Contents

- Motivation
- Equations and parameters
- Parallelization
- Examples
  - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Contents

- Motivation
- Equations and parameters
- Parallelization
- Examples
  - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Full Waveform Inversion algorithm

Misfit functional

$$E(m) = \frac{1}{2}\left\| d^{obs} - d^{cal}(m) \right\|_D^2$$

d^{obs}	-	the receivers data;
m	-	the current model parameters;
$d^{cal}(m)$	-	wavefield computed in receivers for the current model; in this talk we deal with frequency time domain 3D isotropic elastic media.

*Virieux, J. and Operto, S. An overview of full-waveform inversion in exploration geophysics, Geophysics, 74 (6), WCC1-WCC26.

In Frequency Domain, the misfit functional gradient (the case of scalar wave equation):

$$\nabla E(m) = -2\,\mathrm{Re}\int_{\omega_1}^{\omega_2} \omega^2 U(x,y,z;\omega)\overline{W}(x,y,z;\omega)\,d\omega$$

$U(x,y,z;\omega)$ - wavefield computed for the current model for a specific source position

$W(x,y,z;\omega)$ - wavefield for the current model for the registered wavefield taken as sources

In a case of several sources the gradients to sum up.

Contents

- Motivation
- Equations and parameters
- Parallelization
- Examples
  - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Equations and parameters

- Needed a method for effective computing wavefields
- Input: model parameters, source/receivers positions, a set of frequencies
- Output: the wavefield in the target domain

Equations and parameters

We use elastic wave propagation equation for isotropic 3D media

$$\mathrm{L} \equiv \left[i\omega \begin{pmatrix} \rho \boldsymbol{I}_{3\times3} & 0 \\ 0 & \boldsymbol{S}_{6\times6} \end{pmatrix} - \begin{pmatrix} 0 & \hat{P} \\ \hat{P}^T & 0 \end{pmatrix} \frac{\partial}{\partial x} - \begin{pmatrix} 0 & \hat{Q} \\ \hat{Q}^T & 0 \end{pmatrix} \frac{\partial}{\partial y} - \gamma(z) \begin{pmatrix} 0 & \hat{R} \\ \hat{R}^T & 0 \end{pmatrix} \frac{\partial}{\partial z} \right] \boldsymbol{v} = \boldsymbol{f}.$$

where vector of unknowns $\boldsymbol{v}$ comprises nine components. These components include the displacement velocities (v_x, v_y, v_z) and components of the stress tensor $(\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{yz}, \sigma_{xz}, \sigma_{xy})$. ω is the real time frequency, $\rho(x, y, z)$ is the density, $\boldsymbol{I}_{3\times3}$ is 3 by 3 identity matrix, $\boldsymbol{S}_{6\times6}(x, y, z) = \begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix}$ is 6 by 6 compliance matrix and

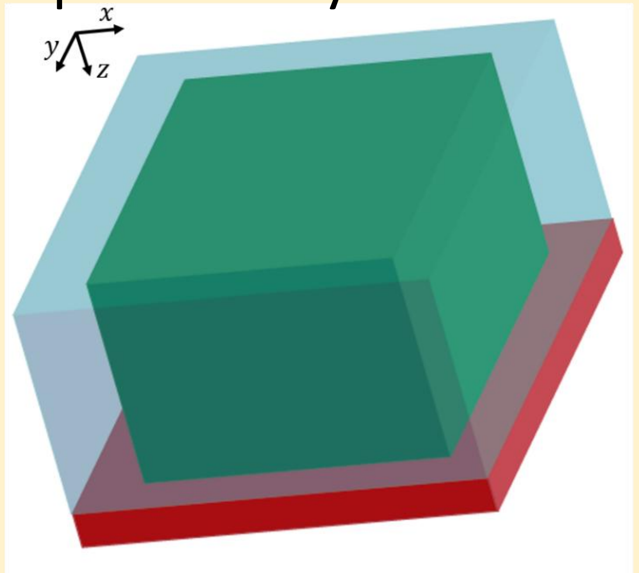
f,

$$A = \begin{pmatrix} a & -b & -b \\ -b & a & -b \\ -b & -b & a \end{pmatrix}, C = \begin{pmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{pmatrix}, \hat{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \hat{Q} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \hat{R} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Equations and parameters

Coefficients $a(x,y,z)$, $b(x,y,z)$ and $c(x,y,z)$ are related to the Lame parameters λ and μ as follows $a = \frac{\lambda+\mu}{\mu(2\mu+3\lambda)}$, $b = \frac{\lambda}{2\mu(2\mu+3\lambda)}$, $c = \mu^{-1}$. $\boldsymbol{f}$ is the right-hand side representing the seismic source. In our experiments, we consider either a volumetric or vertical point-force source. $\gamma(z)$ may be either unity or the damping along z representing the Perfectly Matched Layer (PML)

Computational domain is a cuboid of Nx x Ny x Nz points. This domain includes sponge layers [5] on the horizontal and PML on the vertical boundaries (top and bottom) imitating an elastic radiation condition at infinity. The top boundary can be also the free surface.



Computational domain (green), sponge layers (blue), PML (red)

Equations and parameters: preconditioning

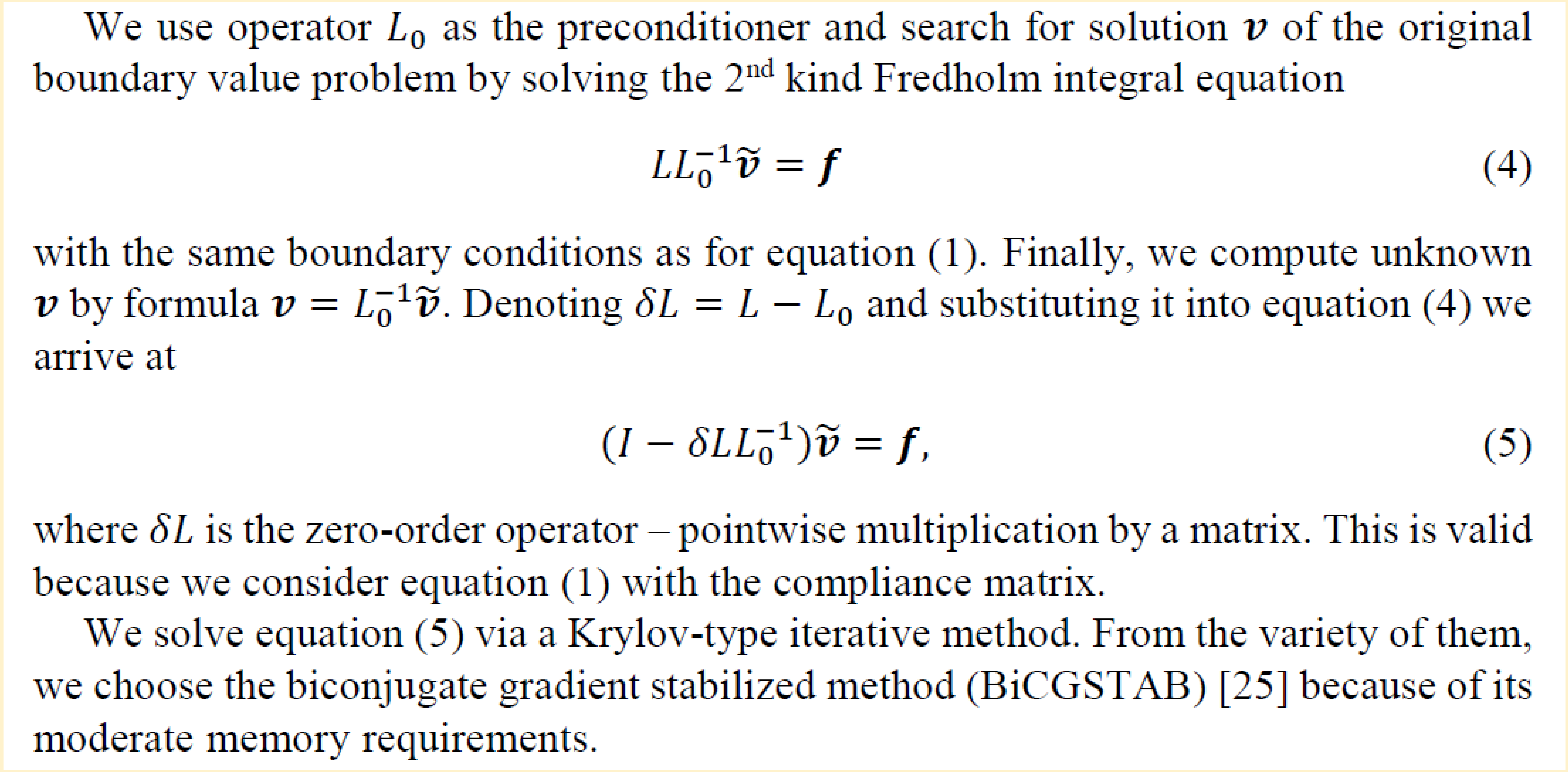
Let L_0 be the same operator as L , but with

$$\rho(x,y,z) = \rho_0(z), \boldsymbol{S}_{\boldsymbol{6\times 6}}(x,y,z) = (1+i\beta)\cdot \boldsymbol{S}_{\boldsymbol{0}}(z)$$

where

$$\boldsymbol{S}_{\boldsymbol{0}}(z) = \begin{pmatrix} A_0 & 0 \\ 0 & C_0 \end{pmatrix}, A_0 = \begin{pmatrix} a_0 & -b_0 & -b_0 \\ -b_0 & a_0 & -b_0 \\ -b_0 & -b_0 & a_0 \end{pmatrix}, C_0 = \begin{pmatrix} c_0 & 0 & 0 \\ 0 & c_0 & 0 \\ 0 & 0 & c_0 \end{pmatrix}$$

Equations and parameters: preconditioning



We use operator L_0 as the preconditioner and search for solution $\boldsymbol{v}$ of the original boundary value problem by solving the 2nd kind Fredholm integral equation

$$L L_0^{-1} \tilde{\boldsymbol{v}} = \boldsymbol{f} \tag{4}$$

with the same boundary conditions as for equation (1). Finally, we compute unknown $\boldsymbol{v}$ by formula $\boldsymbol{v} = L_0^{-1}\tilde{\boldsymbol{v}}$. Denoting $\delta L = L - L_0$ and substituting it into equation (4) we arrive at

$$(I - \delta L L_0^{-1})\tilde{\boldsymbol{v}} = \boldsymbol{f}, \tag{5}$$

where δL is the zero-order operator – pointwise multiplication by a matrix. This is valid because we consider equation (1) with the compliance matrix.

We solve equation (5) via a Krylov-type iterative method. From the variety of them, we choose the biconjugate gradient stabilized method (BiCGSTAB) [25] because of its moderate memory requirements.

Equations and parameters: preconditioning

This assumes computing several times per iteration (depending on a method) the product of the left-hand side operator of equation (5) by a particular vector $\boldsymbol{w}$, i.e. computing $[w-\delta L L_0^{-1} w]$. This process breaks down into three computational steps

1. first, computing $q_1 = L_0^{-1} w$ by solving boundary value problem $L_0 \mathrm{q}_1 = w$;
2. then, computing $q_2 = \delta L q_1$, that in the discrete case is a pointwise multiplication of a tridiagonal matrix by a vector;
3. finally, subtracting the two vectors $[w - q_2]$.

Equations and parameters: preconditioning

To solve $L_0 q_1 = w$ we assume that function $\boldsymbol{w}(x,y,z)$ is expanded into a Fourier series with respect to the horizontal coordinates with coefficients $\hat{\boldsymbol{w}}(k_x,k_y,z)$, where k_x and k_y are the respective spatial frequencies. These coefficients are solutions to the boundary value problems for ordinary differential equations (ODEs)

$$\left[i\omega\begin{pmatrix}\rho_0\boldsymbol{I}_{3\times3} & 0\\ 0 & \boldsymbol{S}_0\end{pmatrix} - ik_x\begin{pmatrix}0 & \hat{P}\\ \hat{P}^T & 0\end{pmatrix} - ik_y\begin{pmatrix}0 & \hat{Q}\\ \hat{Q}^T & 0\end{pmatrix} - \gamma(z)\begin{pmatrix}0 & \hat{R}\\ \hat{R}^T & 0\end{pmatrix}\frac{\partial}{\partial z}\right]\hat{\boldsymbol{v}} = \hat{\boldsymbol{w}}, \quad (6)$$

We solve it numerically, applying a finite-difference approximation, that results in a system of linear algebraic equations (SLAEs) with a banded matrix, whose bandwidth depends on the order of the finite-difference scheme. In this case, computation of $\hat{\boldsymbol{w}}(k_x,k_y,z)$ can be performed via the 2D Fast Fourier Transform (FFT) and after $\hat{\boldsymbol{v}}(k_x,k_y,z)$ are found, $L_0^{-1}\boldsymbol{w}$ can be computed via the inverse 2D FFT.

Contents

- Motivation
- Equations and parameters
- Parallelization
- Examples
    - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Parallelization

Parallelization has three levels:

Highest level.

The FWI for macro velocity reconstruction involves simulations for different seis-mic sources at different low frequencies. This means, that in fact, many boundary value problems for equation (1) are solved at the same time, each having its own right-hand side f. Since they are solved independently of each other, we solve each one with a separate MPI process, assigned to a single node or a group of cluster nodes. This is the highest level of our parallelization strategy. There are no communications between these MPI processes. Assuming that all computational nodes have similar performance, this parallel process scales very well. This is why we do not mention this level of par-allelization in subsequent tests and consider the case of one seismic source and one frequency only.

Parallelization

Second level:

Four computational processes including Krylov iteration method, the 2D forward and inverse FFTs and solving the boundary value problem for equation (6), mainly drive our solver. We decompose the computational domain along one of the horizontal coordinates and parallelize these processes via MPI. The main exchanges between the MPI processes are while performing FFTs. For computing them, we use the Intel MKL library [10] supporting the decomposition along one direction only. In principle, the decomposition along the second horizontal dimension may be also applied with minor corrections of the code using a 2D FFT realization, supporting this functionality. Decomposition along the z-direction is not that obvious, since this involves solving each boundary value problems for equation (6) in parallel.

Parallelization

Third level:

Following this strategy, each MPI process would independently solve its own set of $Nx \cdot Ny/N$ (N – the number of MPI processes) problems. We solve them in a loop, parallelized via OpenMP.

Schematically, our parallelization strategy is presented in Figure:

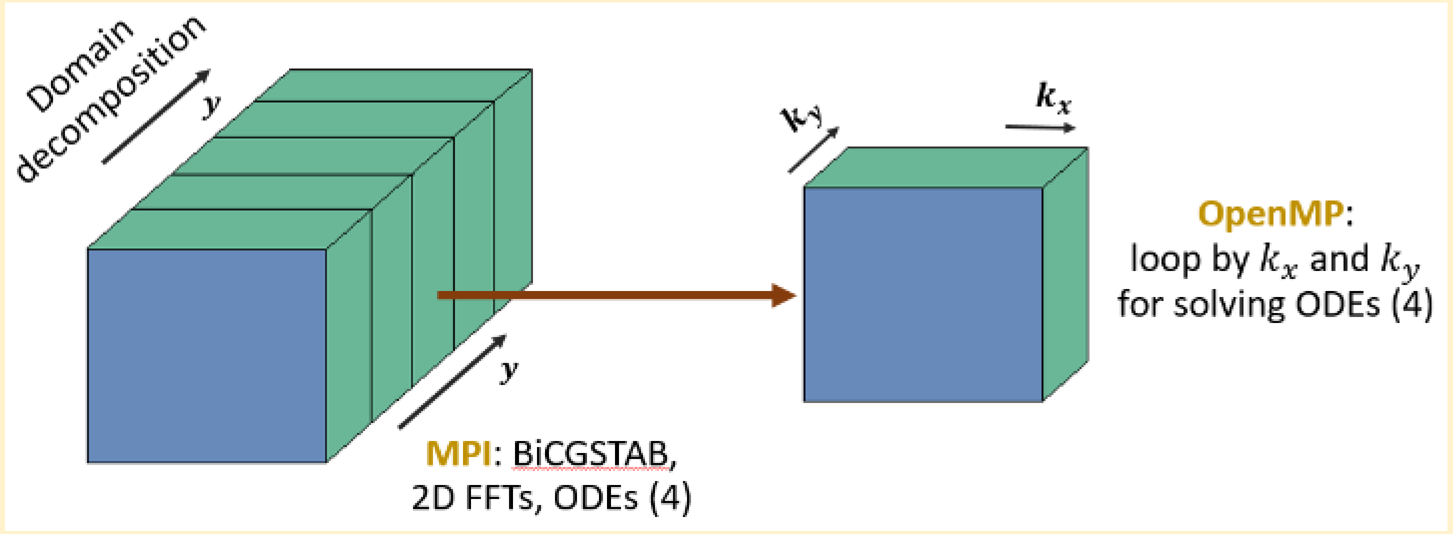
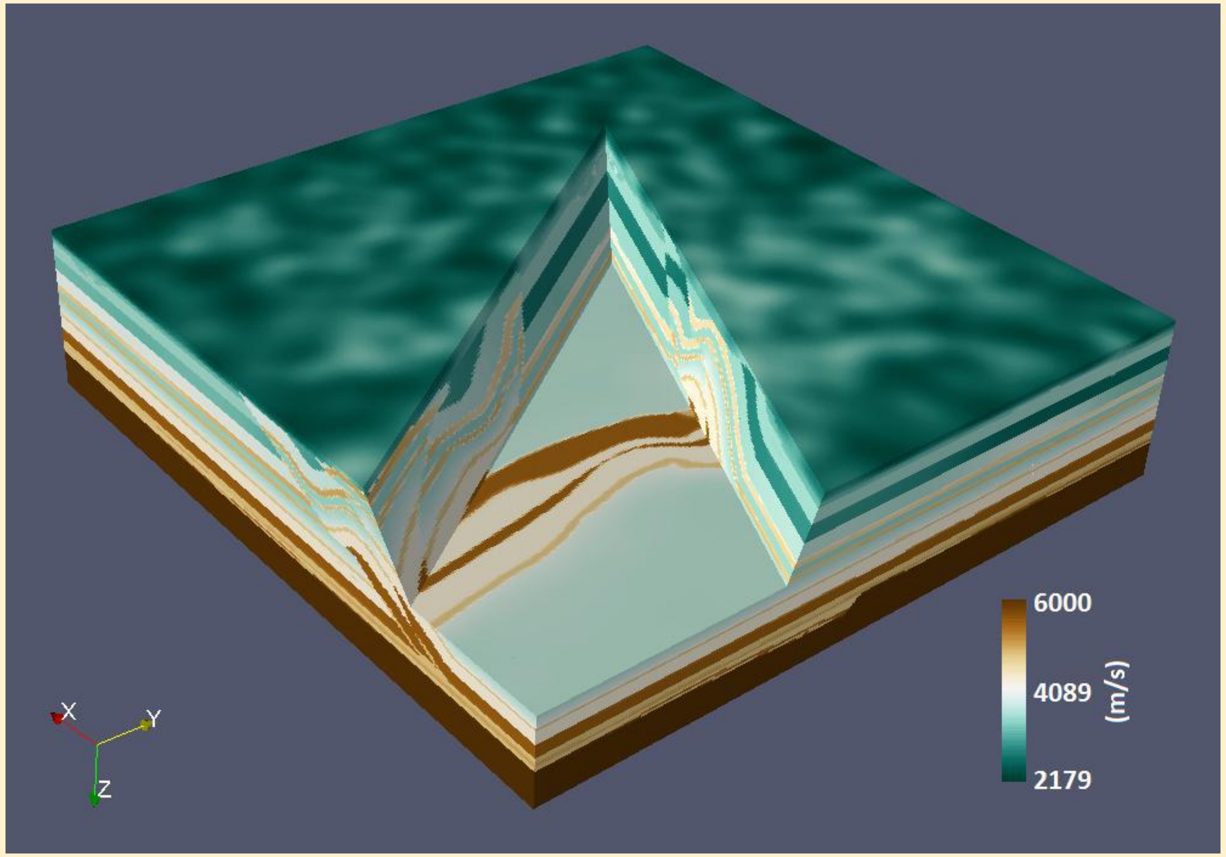

Domain decomposition
y
y
MPI: BiCGSTAB, 2D FFTs, ODEs (4)
k_y
k_x
OpenMP: loop by k_x and k_y for solving ODEs (4)

Figure. Parallelization strategy

Parallelization: Strong and Weak Scaling

Now I present the results of scaling analysis for both MPI and OpenMP. All results presented here have been computed on a HPC cluster comprising nodes with two Intel® Xeon® E5-2680v4 @ 2400 MHz CPUs and interconnected with 56 Gb FDR InfiniBand HCA. Double precision floating point format has been used in the computations.
This is necessary, when dealing with vectors of huge dimensions, for instance, for computing their dot product. As a stopping criterion for the BiCGSTAB, we used a 10^{-3} threshold for the relative residual of the L_2-norm providing enough accuracy for FWI applications.

Parallelization: Strong and Weak Scaling



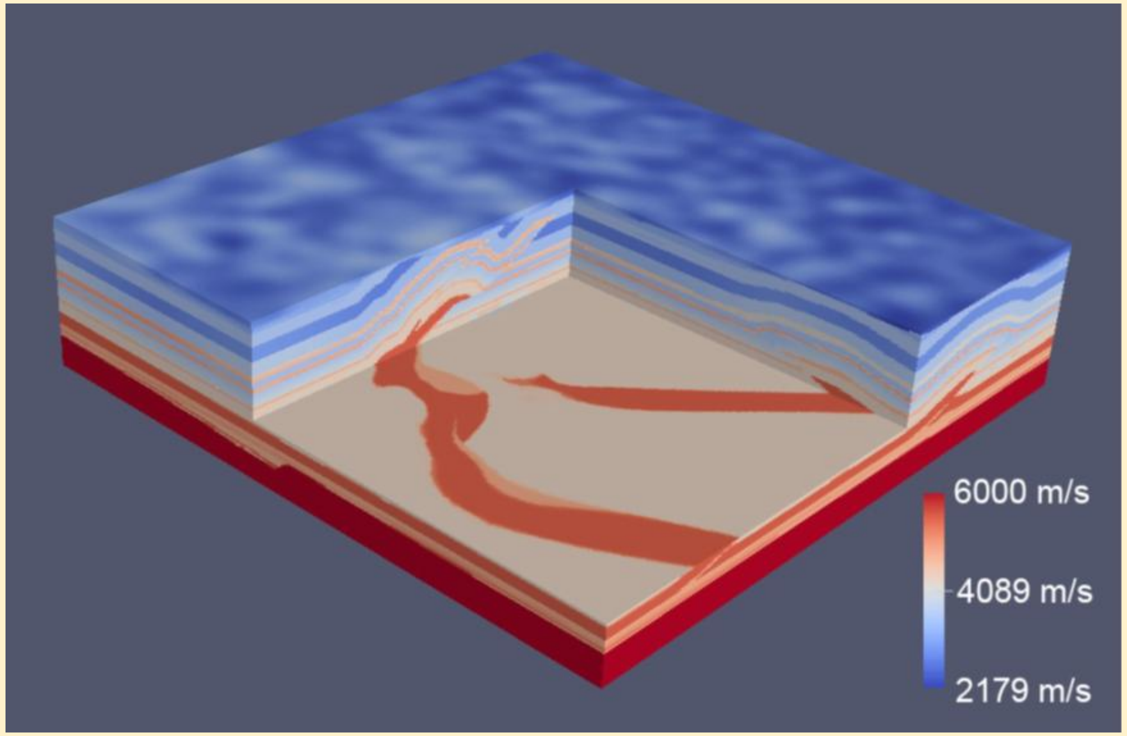
The model used for numerical experimets: 3D SEG/EAGE overthrust model (19.8 x 19.8 x 4.65 km).

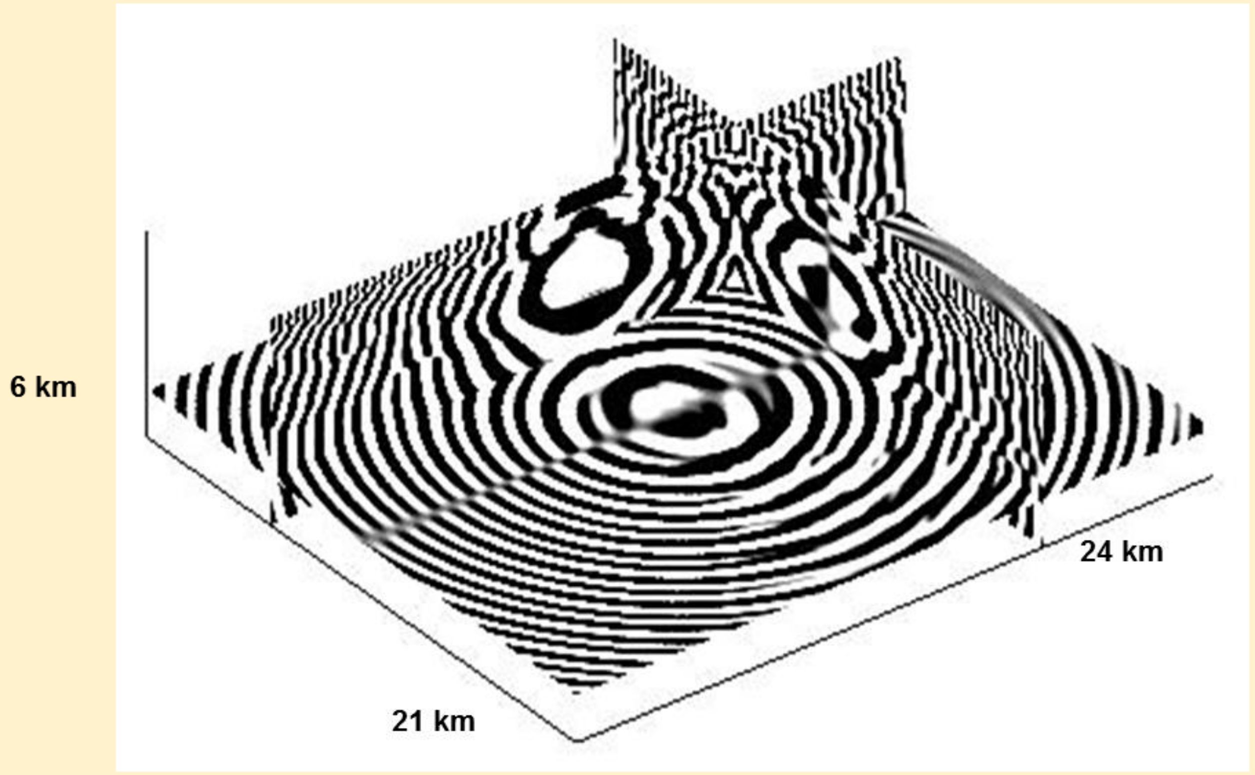
Parallelization: Strong and Weak Scaling

The model was discretized with a uniform grid of 957 x 169 x 651 points with a lateral cell size of 25 m and a vertical cell size of 10 m. The source was placed in the middle of the area at 10 m depth. In the next slide we present a 3D view of the vertical velocity at 5 Hz and 10 Hz computed with the iterative solver. To obtain these results we used 18 computational nodes with 4 MPI processes per node and 7 OpenMP threads per MPI process. The total computational times for the 5 and 10 Hz solutions are 32 and 108 minutes, respectively.

Parallelization: Strong and Weak Scaling

Russian Supercomputing Days



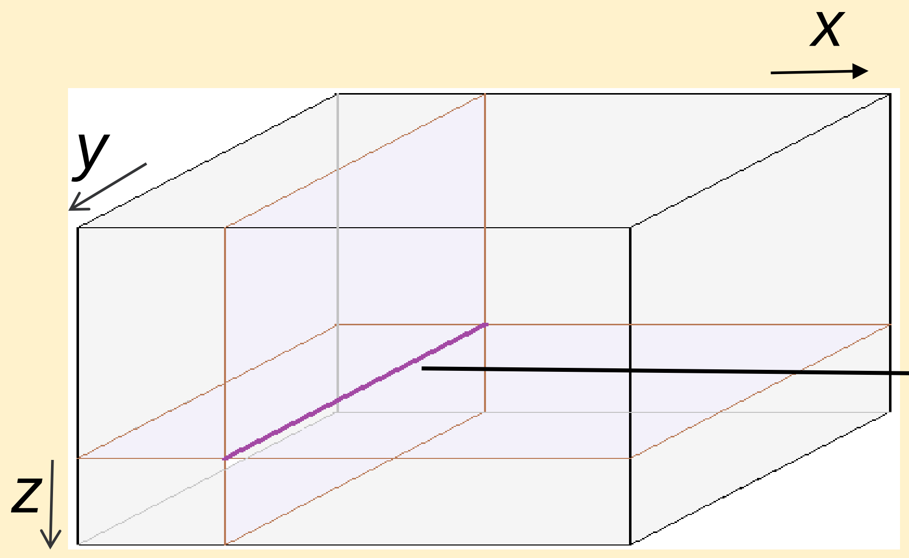


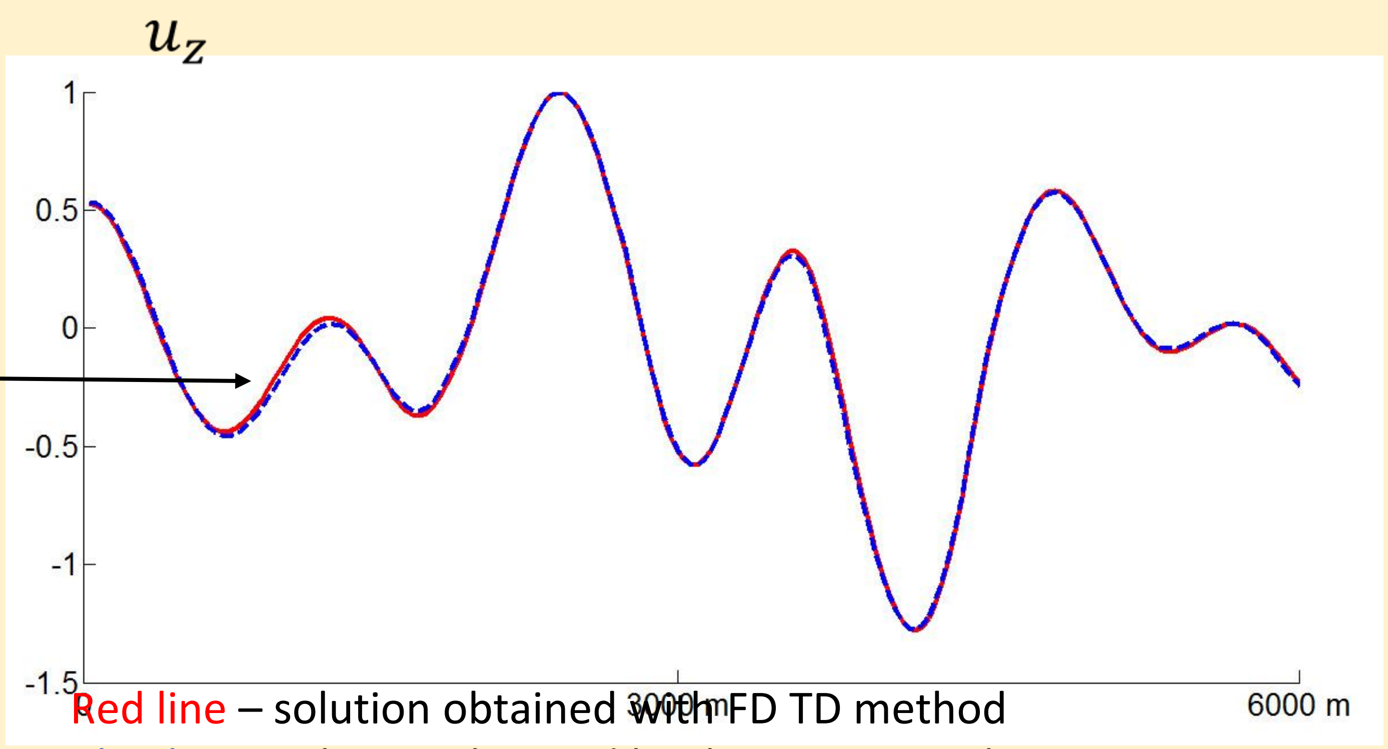
RAM needed: 420 GB.
# of MPI processes: 75.
Time per one source: 84 min

66 iterations to converge

Parallelization: Strong and Weak Scaling

- SEG/EAGE overthrust submodel (5 Hz)

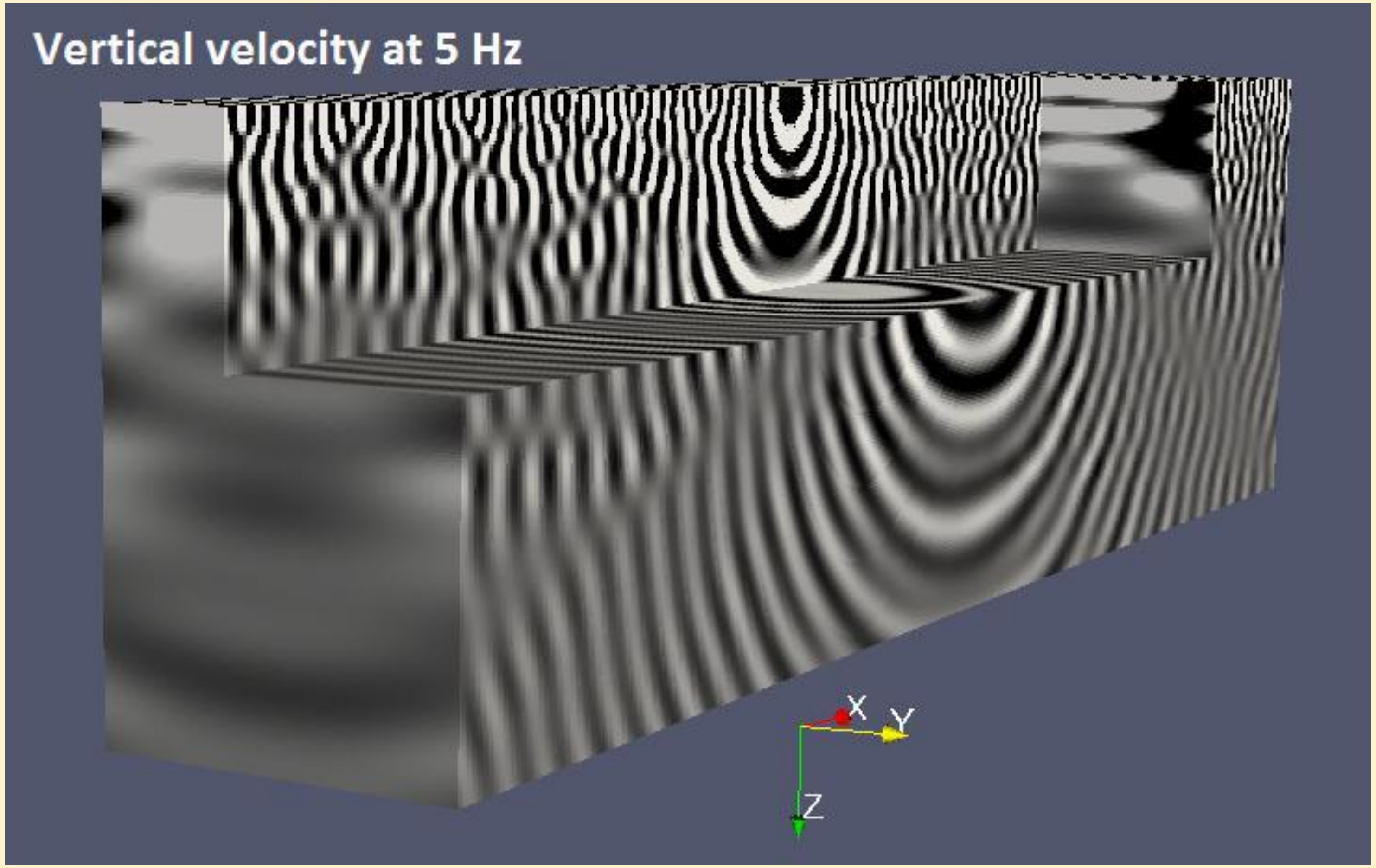




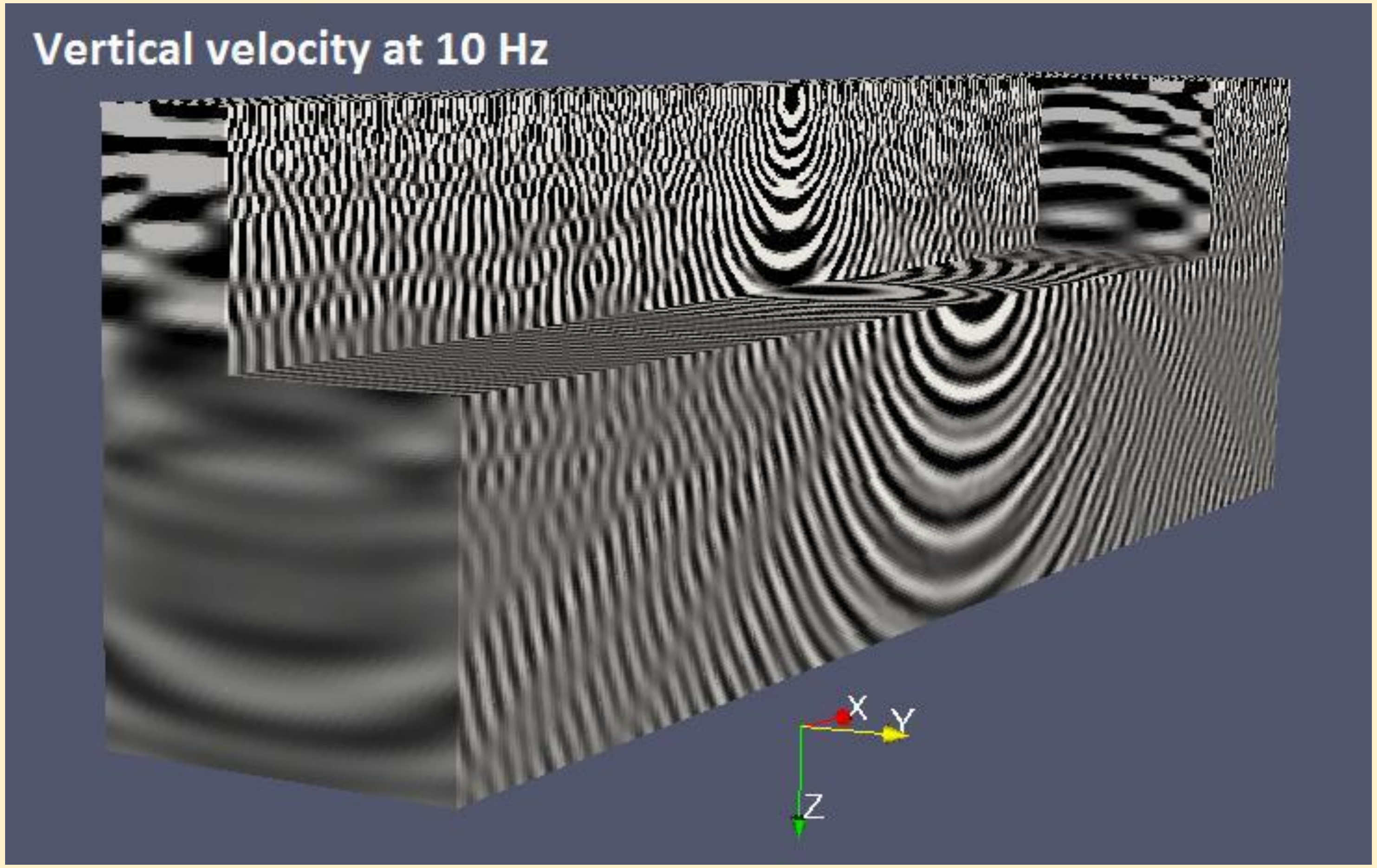
Red line – solution obtained with FD TD method
Blue line - solution obtained by the iterative solver in frequency domain

80 iterations to converge

Parallelization: Strong and Weak Scaling



Parallelization: Strong and Weak Scaling



Parallelization: Strong and Weak Scaling

MPI strong scalability of the solver is defined as ratio t_M/t_N, where t_M and t_N are elapsed run times to solve the problem with N and $M>N$ MPI processes each corresponding to a different CPU. Using MPI, we parallelize two types of processes. First, those scaling ideally (solving problems (6)), for which the computational time with N processes is T/N. Second, the FFT, that scales as $T_{FFT}/\alpha(N)$, with coefficient $1<\alpha(N)<N$. The total computational time becomes $T/N+T_{FFT}/\alpha(N)$ (here we simplify, assuming no need of synchronization) with scaling coefficient $(T+T_{FFT})/(T/N+T_{FFT}/\alpha(N))$, that is greater than $\alpha(N)$.
This is why, we expect very good scalability of the algorithm, somewhere between ***the scalability of the FFT and the ideal scalability***.

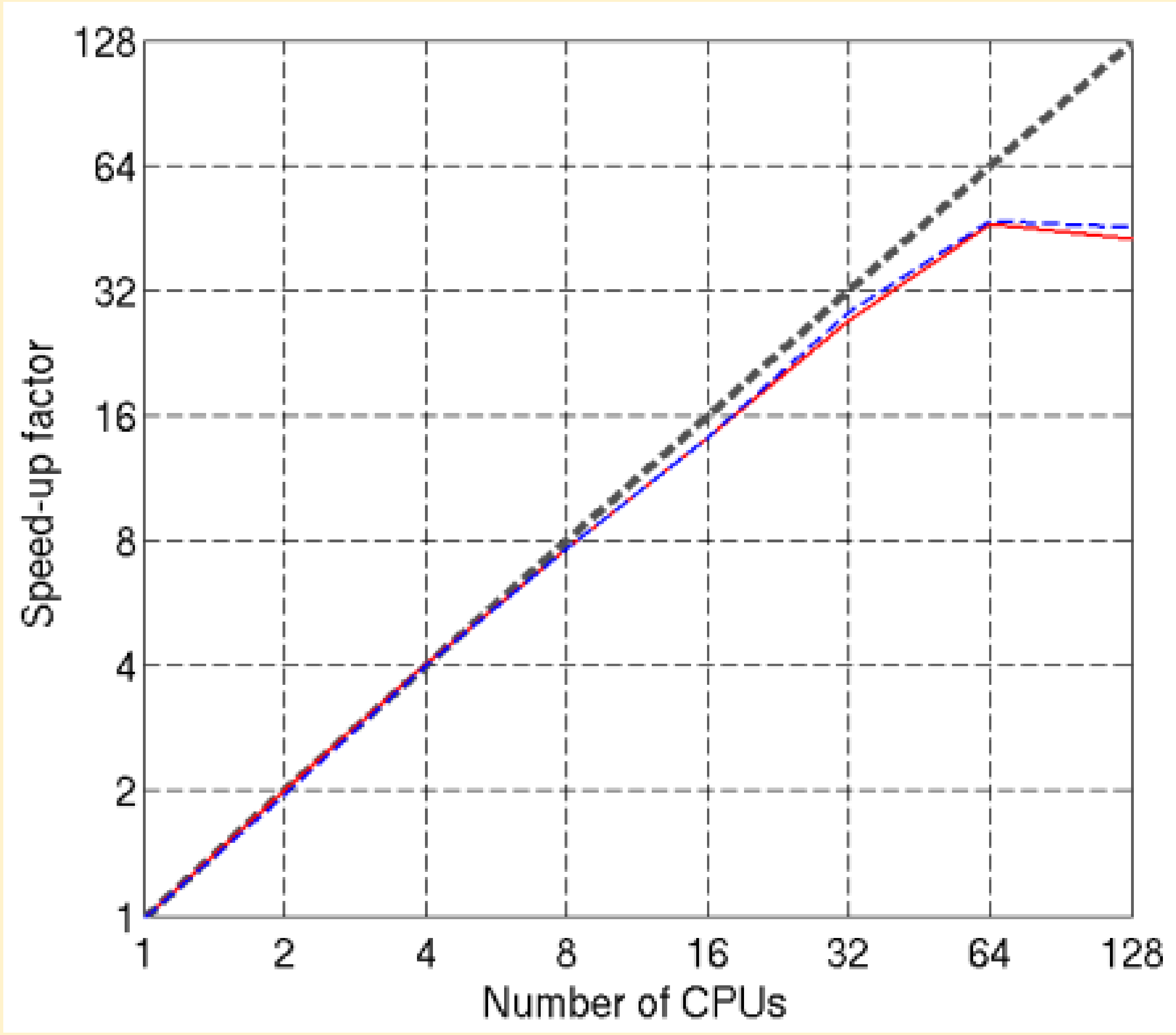
Parallelization: Strong and Weak Scaling

Strong MPI scaling of the solver developed:

•The blue dashed line is the result for the Marmousi model,

•The red line - for the SEG/EAGE overthrust model.

•The dashed grey line is the ideal scalability.



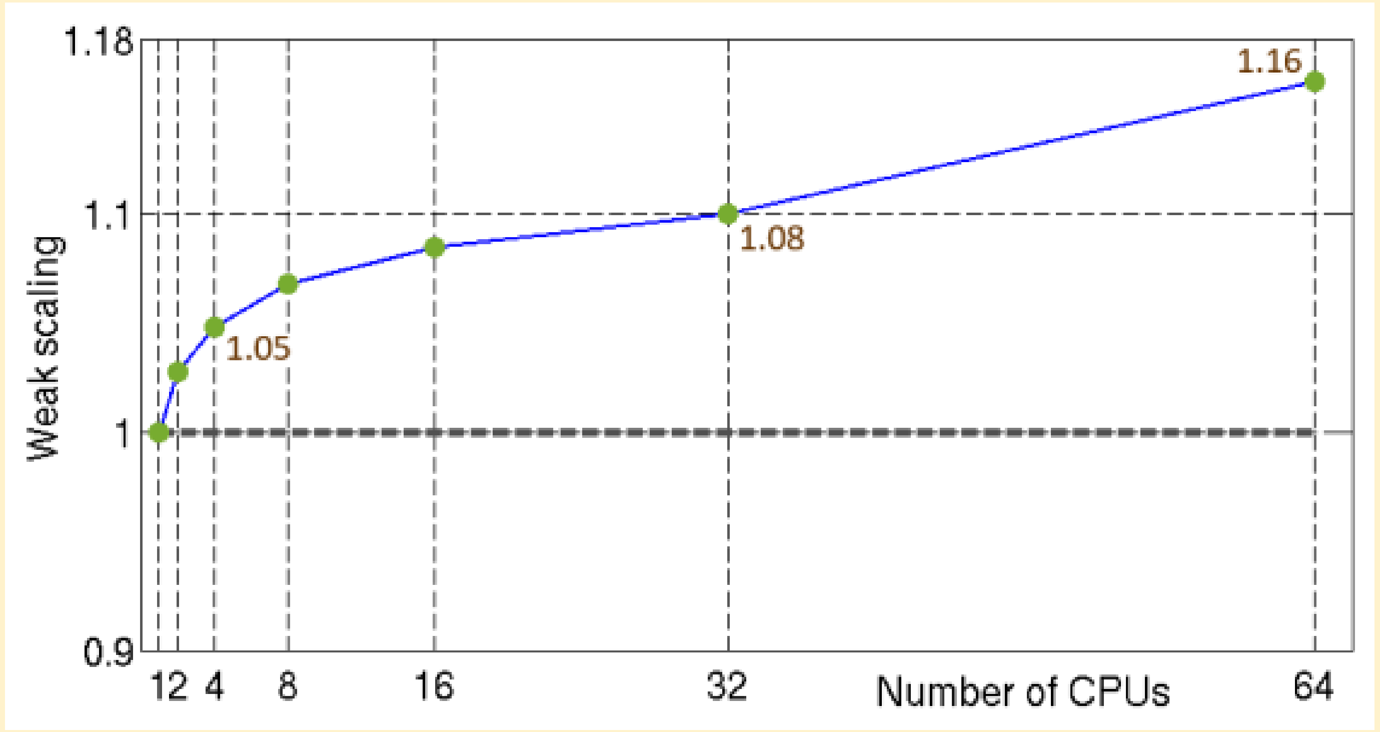
Parallelization: Strong and Weak Scaling

MPI Weak Scaling Analysis

For weak scaling estimation, we assign the computational domain to one MPI pro-cess and then extend the size of the computational domain along the y-direction, while increasing the number of MPI processes. Here, we use one MPI process per CPU. The load per CPU is fixed. For the weak scaling, we use function $fweak(N)$=$T(N)/T(1)$, where $T(N)$ is the average computational runtime per iteration with N MPI processes. The ideal weak scalability corresponds to $fweak(N)$=1.

Parallelization: Strong and Weak Scaling

MPI Weak Scaling Analysis



Weak scaling measurements: the blue line is the result of the iterative solver and the dashed grey line is the ideal weak scaling.

Parallelization: Strong and Weak Scaling

Russian Supercomputing Days

OpenMp Scaling Analysis

As already explained above, with OpenMP we parallelize the loop over spatial fre-quencies for solving the boundary value problems (6). To estimate the scalability of this part of our solver, we performed simulations in a small part of the SEG/EAGE overthrust model comprising 660×50×155 points on a single CPU having 14 cores with hyper-threading switched off and without using MPI. Fig. 5 shows that our solver scales well for all threads involved in this example.

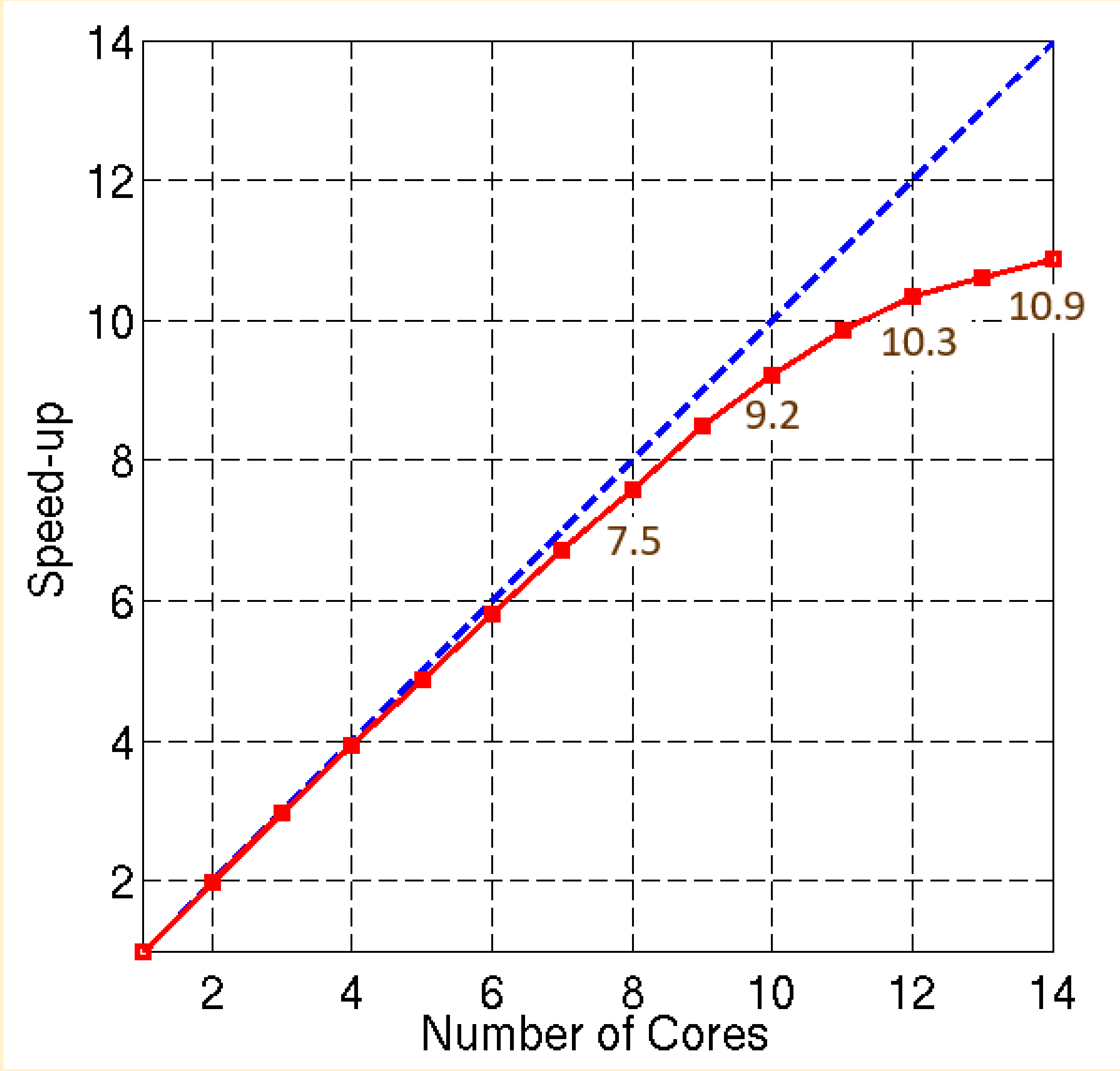
It is worth mentioning, that we use OpenMP as an extra option applied when further increasing of the number of MPI processes doesn't improve performance any more, but the computational system is not fully loaded, i.e., there are free cores.

Parallelization: Strong and Weak Scaling

Russian Supercomputing Days

OpenMp Scaling Analysis

Strong scalability analysis on one CPU of the part parallelized via OpenMP: the dashed blue line is the ideal scalability and the red line is the iterative solver scalability.



Conclusions

We present a parallel iterative solver capable of modeling wavefields in 3D elastic land models of big size at low frequencies. The solver includes both MPI and OpenMP to reduce the computation time and shows good scalability. Further improvement of MPI scaling may be achieved by incorporating domain decomposition along the two horizontal directions into the current MPI parallelization scheme.

Acknowledgements

We are grateful to Vincent Etienne and Michael Jervis for reviewing of our manu-script. Special thanks to Maxim Dmitriev for useful discussions and advice on this topic. Two of the authors (Victor Kostin and Vladimir Tcheverda) have been sponsored by the Russian Science Foundation grant 17-17-01128.

THANK YOU!

Questions?