On the Comparison of MUMPS and STRUMPACK for 3D Frequency-Domain Elastic Wave Modeling

Y. Li[1], R. Brossier[1], L. Métivier[2]

[1] Univ. Grenoble Alpes, ISTerre; [2] CNRS, Univ. Grenoble Alpes, LJK

Summary

We investigate the performance of parallel direct solvers for performing the 3D frequency-domain elastic wave modelling using the spectral element method. The widely used MUMPS and STRUMPACK solvers are chosen. Numerical experiments show that MUMPS can not keep a satisfactory scalability when using more than about one hundred MPI processes. Resorting to a hybrid implementation of MPI and OpenMP circumvents such a limitation, whereas the number of threads has a limit depending on the architecture of CPU. As a comparison, STRUMPACK has a better MPI scalability. Although STRUMPACK benefits less from multi-threading for a given number of cores, a hybrid implementation is still feasible. Thus STRUMPACK might be a better option for larger-scale problems because of its better scalability. Current bottleneck for both of these two solvers is the matrix reordering method. A parallel, scalable and efficient reordering method is necessary for large-scale modelling. Computing the matrix reordering on a large-memory node and using such a reordering for matrix factorization on normal nodes may also avoid the bottleneck. FWI applications can adopt such a workflow, because for each frequency, the impedance matrices of each iteration share the same structure and the analysis needs to be performed only once.

Introduction

Elastic, visco-elastic or anisotropic effects have a great effect on onshore seismic imaging techniques, for example, the full waveform inversion (FWI). It is thus necessary to perform the anisotropic visco-elastic wave modeling to incorporate all these effects. Current 3D seismic imaging applications mainly rely on the time-domain forward modeling due to its memory efficiency and satisfactory scalability. Study on 3D frequency-domain modeling in the framework of onshore seismic imaging is relatively rare. A successful application in offshore environment (Operto and Miniussi, 2018) prompts us to investigate the 3D frequency-domain anisotropic elastic wave modeling to prepare for further imaging. In terms of discretization method, the spectral element method (SEM, Komatitsch et al., 2000; Trinh et al., 2019) has been investigated particularly in seismic imaging and seismology. The specific character of SEM consists in using a mesh of hexahedra in 3D and choosing the Gauss-Lobatto-Legendre (GLL) points for the integration and Lagrange interpolation. Using high-order Lagrange polynomials and Gauss quadrature rules on GLL points enables spectral convergence when solving smooth problems. SEM fits well the applications for seismic exploration where we can presume the continuity of the solution and media. Note that anisotropy can be considered without making any extra effort and the viscosity can also be easily incorporated by using complex-valued elastic modulus in the frequency domain.

We investigate the performance of direct solvers for solving the linear system related to the frequency-domain wave modeling. The widely used MUMPS and STRUMPACK solvers are compared (MUMPS-team, 2019; Ghysels et al., 2018). Previous numerical experiments show a loss of scalability of MUMPS when more than about a hundred MPI processes are used (Amestoy et al., 2001; Y. Li, 2019). Resorting to a hybrid MPI/OpenMP implementation can improve the scalability performances of MUMPS, however, the number of threads which can be used simultaneously is limited by the CPU architecture. As a comparison, we would like to investigate the performance of STRUMPACK, which is shown to be better suited for larger-scale modelings (Mary, 2017). We use the full-rank version of MUMPS and STRUMPACK and compare their performance with pure MPI implementation and hybrid implementation of MPI and OpenMP.

Discretization

A fifth-order SEM is used to discretize the 3D frequency-domain elastic wave equation,

$$\rho\omega^2 u_j + \frac{\partial}{\partial x_i}\left(c_{ijkl}\frac{\partial u_k}{\partial x_l}\right) + f_j(\omega,\mathbf{r}_s) = 0, \quad i,j,k,l = 1,2,3, \tag{1}$$

where ρ is the density, ω is the angular frequency, u_j is the displacement vector, c_{ijkl} is the elastic modulus tensor and $f_j(\omega,\mathbf{r}_s)$ is the point source force vector located at $\mathbf{r}_s$. Perfectly matched layers are used to absorb the outgoing waves. The discretized linear system reads

$$A\mathbf{u} = \mathbf{f}, \tag{2}$$

where $A = \omega^2 M + K$ is the impedance matrix, M is the mass matrix and K is the stiffness matrix. Vector $\mathbf{u}$ is the discretized displacement vector and $\mathbf{f}$ represents the discretized source vector.

Numerical experiments

We use METIS (Karypis, 2013) for the matrix reordering and to reduce the fill-in of the factorization. Experiments are conducted with an isotropic homogeneous model of $20 \times 20 \times 20$ elements and with $V_P = 5000\,\text{m/s}$, $V_S = 2500\,\text{m/s}$ and $\rho = 1\,\text{g/cm}^3$. Other physical parameters are summarized in Table 1. The source is located at the center of the model. Free surface boundary condition is considered and two elements in the PML is set on lateral and bottom sides of the model. Thus the model grid is $111 \times 121 \times 121$. Taking into account the three components u_x, u_y and u_z, the size of the linear system and number of nonzeros in the matrices are given in Table 1. We propagate 20 S-wavelengths in each dimension. The real part of u_x, u_y and u_z are shown in Figure 1. The distorted concentric circles indicate the influence of the free-surface boundary condition.

Table 1 *Parameter settings for wave modeling.*

N_e/dim	N_{PML}	$\lvert e\rvert$ (m)	V_P (m/s)	V_S (m/s)	ρ (g/cm^3)	f (Hz)	DOFs	N_{NNZ}
20	2	100	5000	2500	1	25	4,875,453	562,733,859

Pure MPI implementation

The number of MPI processes increases from 16 to 512 with the number of OpenMP threads kept as one. The statistics of MUMPS and STRUMPACK, including the time for analysis (T_A), factorization (T_F), solving (T_S) and total time (T_T) are summarized in Tables 2 and 3. Figure 2 shows the factorization and solving time of MUMPS and STRUMPACK with respect to different number of MPI processes. The dashed lines represent the ideal scaling and the solid curves are from MUMPS and STRUMPACK. STRUMPACK has a better scalability in such a pure MPI implementation. MUMPS scalability starts to deteriorate when the number of MPI processes becomes larger than 128. For the solving step, both MUMPS and STRUMPACK are efficient (a few seconds) but STRUMPACK is relatively faster (Mary, 2017). Moreover, the solving step of STRUMPACK also scales well as the number of MPI processes increases, while it is not the case for MUMPS. For 3D frequency-domain FWI applications, where thousands of solving might be required, time saved by STRUMPACK may be larger and this feature might be crucial for FWI efficiency.

Table 2 *MUMPS performance details.*

#MPIs	T_A(s)	T_F(s)	T_S(s)	T_{Total}(s)
16	321.9	2234.4	3.89	2622.5
32	334.9	1425.5	3.11	1836.4
64	345.6	879.1	2.45	1296.6
128	392.9	605.0	2.44	1092.6
256	394.5	581.2	3.19	1070.9
512	396.4	553.5	2.61	1106.9

Table 3 *STRUMPACK performance details.*

#MPIs	T_A(s)	T_F(s)	T_S(s)	T_{Total}(s)
16	361	1997	2.03	2430
32	357	1136	1.51	1570
64	363	678	0.93	1089
128	411	539	0.75	1033
256	408	270	0.38	761
512	406	157	0.27	584

Hybird implementation of MPI and OpenMP

The numerical settings are the same as in Table 1. We increase the total number of computing cores from 128 to 512. The number of OpenMP threads varies from 1 to 16 to fit the hardware settings (2 Intel Xeon SKL Gold 6130 processors per node, 16 cores per processor) and the number of MPI processes changes accordingly. Figure 3 presents the corresponding factorization time of MUMPS and STRUMPACK with different number of OpenMP threads. The dashed lines are MUMPS results and the solid curves are for STRUMPACK. Similar as mentioned previously, STRUMPACK has a better scalability. However, MUMPS benefits more from the increase of OpenMP threads. Given a fixed number of processors, the performance of STRUMPACK degrades if more OpenMP threads are used. To summarize, MUMPS prefers a hybrid implementation of MPI and OpenMP. STRUMPACK performs well with pure MPI implementation.

Benefits from OpenMP threads

As shown in previous section, for a fixed number of processors, STRUMPACK favors a pure MPI implementation than a hybrid implementation. To check more clearly the benefits from OpenMP threads, we investigate the performance of both MUMPS and STRUMPACK with a fixed number of MPI processes and with different number of OpenMP threads. The same model is used. The number of MPI processes

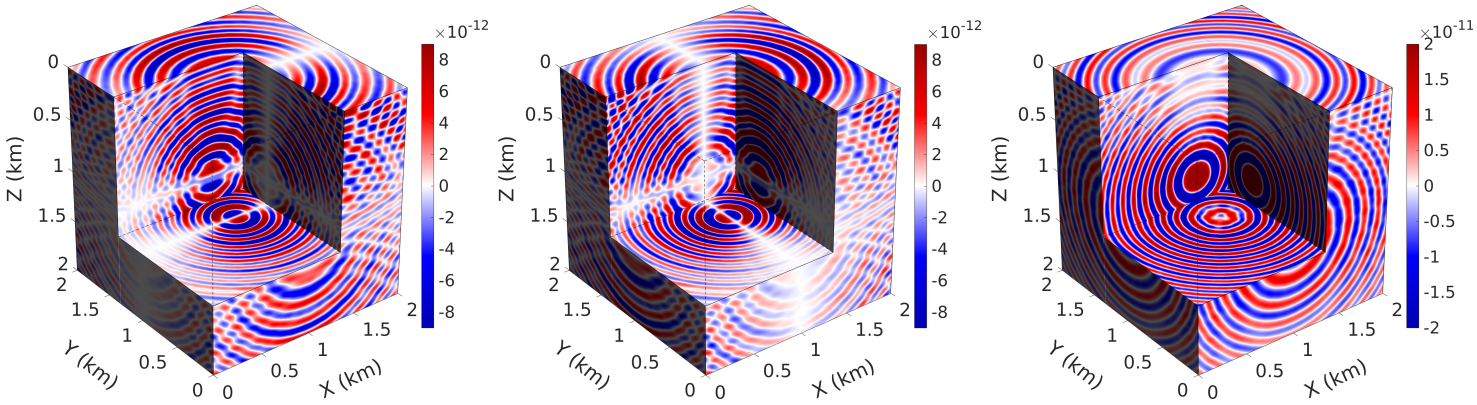

Figure 1 *Real part of wavefields u_x (left), u_y (middle) and u_z (right).*

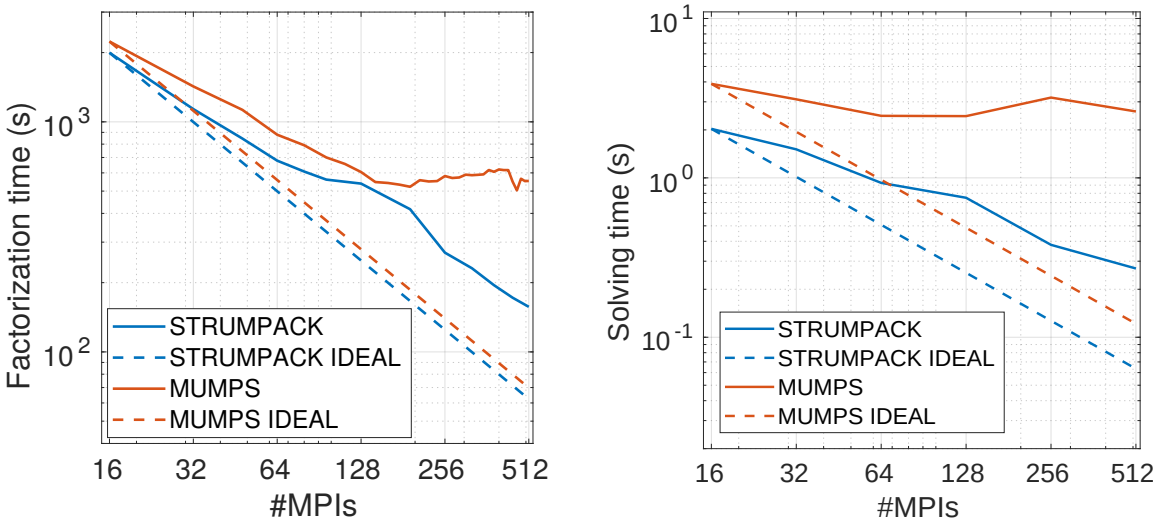

Figure 2 *Factorization (left) and solving (right) time scalability of MUMPS and STRUMPACK with a pure MPI implementation. The dashed lines represent the ideal scaling.*

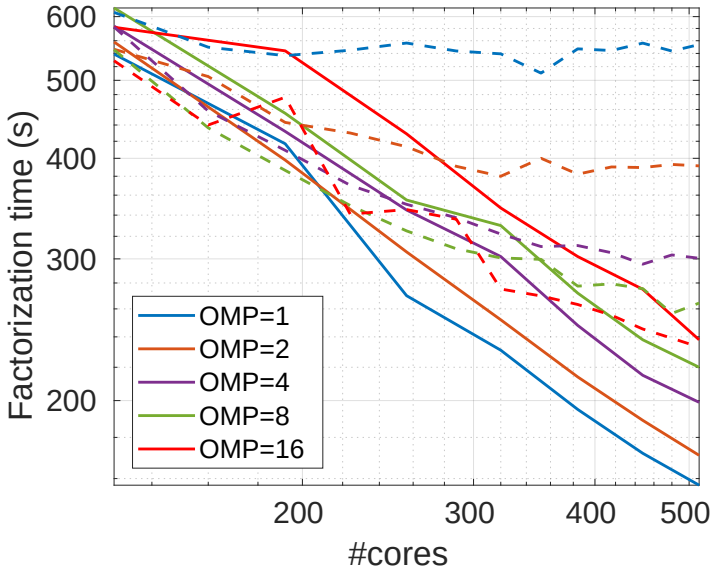

Figure 3 *Factorization time of MUMPS and STRUMPACK with different number of OpenMP threads. The dashed lines are for MUMPS and the solid lines for STRUMPACK. The number of MPI processes equals the total number of cores divided by the number of OpenMP threads.*

is 8 and the number of OpenMP threads increases from 1 to 16. Figure 4 presents the corresponding factorization time of MUMPS and STRUMPACK with respect to the number of OpenMP threads. Both of MUMPS and STRUMPACK factorization time decrease as the number of OpenMP threads increases. The factorization time is close to each other.

Discussion and Conclusions

STRUMPACK has a better MPI scalability than MUMPS. MUMPS can compensate its unsatisfactory MPI scalability with hybird OpenMP implementation but this is not ideal. STRUMPACK prefers MPI parallelism but can also benefit from OpenMP if MPI limit is reached. In absolute on this test case, STRUMPACK is more efficient than MUMPS for a given number of computational resources. In addition, STRUMPACK solve step scales with the number of MPI processes which is not the case for MUMPS, and this is important for FWI where the large number of RHS can make the solve step as costly as the factorization step (Operto and Miniussi, 2018; Amestoy et al., 2016). The current bottleneck for both of the two solvers is the matrix reordering method. METIS is the best reordering method among the methods that we have tested, in reducing the flops and memory cost of the LU factorization. The problem is that METIS is sequential and the memory cost of METIS reordering has already used up the memory of one node. Parallel reordering methods, such as ParMETIS and PTSCOTCH, are potential alternatives. However, the quality of matrix reordering is not as good, i.e., the memory cost of the LU factorization is more than doubled compared with that using METIS. In addition, the scalability of ParMETIS is not satisfactory enough. Thus a scalable and efficient reordering method becomes neces-

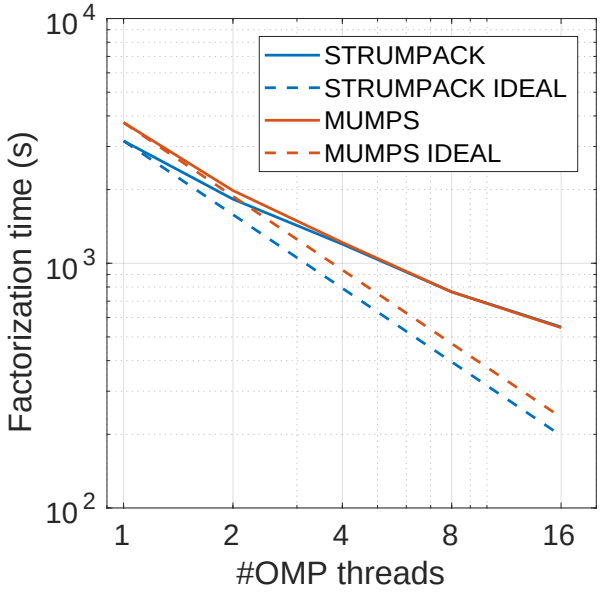

Figure 4 *Factorization time of MUMPS and STRUMPACK with different number of OpenMP threads for 8 MPI processes. The dashed lines represent the ideal scaling and the solid curves are from MUMPS and STRUMPACK.*

sary. Another way to circumvent this bottleneck is to perform the analysis step with METIS on a node with larger memory and to write the matrix permutation vector in a file. Thus such a reordering can be used for factorization which is performed on normal computing nodes. FWI applications can adopt such a workflow, because for each frequency, the impedance matrices of each iteration share the same structure and the analysis needs to be performed only once. Future investigation may also consist in using the HSS preconditioning of STRUMPACK for an iterative solver (GMRES) because HSS favors being used in a preconditioner mode for very large-scale problem.

Acknowledgements

This study was partially funded by the SEISCOPE consortium (*http://seiscope2.osug.fr*), sponsored by AKERBP, CGG, CHEVRON, EQUINOR, EXXON-MOBIL, JGI, PETROBRAS, SCHLUMBERGER, SHELL, SINOPEC, SISPROBE and TOTAL. This study was granted access to the HPC resources of CIMENT infrastructure (https://ciment.ujf-grenoble.fr) and CINES/IDRIS/TGCC under the allocation 046091 made by GENCI.

References

Amestoy, P., Brossier, R., Buttari, A., L'Excellent, J.Y., Mary, T., Métivier, L., Miniussi, A. and Operto, S. [2016] Fast 3D frequency-domain full waveform inversion with a parallel Block Low-Rank multifrontal direct solver: application to OBC data from the North Sea. *Geophysics*, **81**(6), R363 – R383.

Amestoy, P.R., Duff, I.S., Koster, J. and L'Excellent, J.Y. [2001] A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal of Matrix Analysis and Applications*, **23**(1), 15–41.

Ghysels, P., Li, X.S., Chávez, G. and Liu, Y. [2018] *STRUMPACK Users' Guide - Version 3.1.1*. Lawrence Berkeley National Laboratory.

Karypis, G. [2013] *METIS - A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices - Version 5.1.0*. University of Minnesota.

Komatitsch, D., Barnes, C. and Tromp, J. [2000] Simulation of anisotropic wave propagation based upon a spectral element method. *Geophysics*, **65**(4), 1251–1260.

Mary, T. [2017] *Block Low-Rank multifrontal solvers: complexity, performance and scalability*. Ph.D. thesis, PhD thesis, Université de Toulouse.

MUMPS-team [2019] *MUltifrontal Massively Parallel Solver (MUMPS 5.2.1) Users' guide (June, 2019)*. ENSEEIHT-ENS Lyon, http://www.enseeiht.fr/apo/MUMPS/ or http://graal.ens-lyon.fr/MUMPS.

Operto, S. and Miniussi, A. [2018] On the role of density and attenuation in 3D multi-parameter visco-acoustic VTI frequency-domain FWI: an OBC case study from the North Sea. *Geophysical Journal International*, **213**, 2037–2059.

Trinh, P.T., Brossier, R., Métivier, L., Tavard, L. and Virieux, J. [2019] Efficient 3D time-domain elastic and viscoelastic Full Waveform Inversion using a spectral-element method on flexible Cartesian-based mesh. *Geophysics*, **84**(1), R75–R97.

Y. Li, R. Brossier, L.M. [2019] 3D frequency-domain elastic wave modeling with spectral-element method using a massively parallel direct solver. *Geophysics*, **accepted**.