

Enhancing the Performance of Geomechanical Reservoir Simulations through Multiscale Preconditioning Techniques

Figueiredo M.¹, Rodrigues J.¹, Gasparini L.¹, Carvalho L.M.², Augusto D.³, Goldfeld P.⁴

¹Petrobras Research Center - CENPES Horácio Macedo Av., 950, 21941-915, Rio de Janeiro - RJ, Brazil
mateus.figueiredo@petrobras.com.br, jrprodrigues@petrobras.com.br, lsgasparini@petrobras.com.br

²Rio de Janeiro State University - UERJ Rua São Francisco Xavier, 524, 20559-900, Rio de Janeiro - RJ, Brazil
luzmc@ime.uerj.br

³Oswaldo Cruz Foundation Av. Brasil, 4365, 21040-900, Rio de Janeiro - RJ, Brazil
daa@fiocruz.br

⁴Rio de Janeiro Federal University - UFRJ Av. Athos da Silveira Ramos, 149, 21941-909, Rio de Janeiro - RJ, Brazil
goldfeld@matematica.ufrj.br

Abstract. The simulation of reservoir geomechanical models presents significant challenges. The need to include under, over, and side burden, besides the reservoir itself, results in meshes that can reach several hundred million elements. In this work, we explore a multiscale methodology designed to enhance the performance of iterative solvers in reservoir geomechanical simulations, leveraging our previous findings presented in Figueiredo Et. Al., “A parallel viscoplastic multiscale reservoir geomechanics simulator” (Upstream Oil and Gas Technology, 2022). Our approach employs multiscale techniques as a preconditioning strategy to accelerate the convergence of the conjugate gradient method, which is essential given the complexity and size of the problems we address. In this study, we focus on optimizing the parameters of our iterative method to maximize efficiency and convergence speed when solving large scale geomechanical models. We will systematically investigate the impact of various configurations for iteratively solving the coarse scale problem, including single versus double precision, ILU(0) versus ILU(1) preconditioning, and different convergence tolerances. Our goal is to identify the optimal settings that yield the best overall performance for practical geomechanical simulations. Through a series of real field geomechanical test cases, we will demonstrate the impact the configuration of the iterative linear solver for the coarse problem has on the effectiveness of our proposed methodology. These test cases will be executed in parallel on a high-performance computing environment utilizing both distributed and shared memory architectures. We will utilize the linear solver framework outlined in Gasparini et al., “Hybrid parallel iterative sparse linear solver framework for reservoir geomechanical and flow simulation” (Journal of Computational Science, 2021), to ensure robust and efficient computations. Particular emphasis will be placed on the advantages of employing single precision computations, which have shown promising results in our initial tests. This choice aligns with current trends in computational practices, highlighting the potential for significant performance improvements that mixed precision techniques exhibit on modern hardware.

Keywords: High Performance Computing, Linear Solvers, Geomechanics.

1 Introduction

A significant portion of the runtime in various types of simulations is spent solving linear systems, and geomechanical simulations are no exception. Investing in a robust and fast linear solver is extremely important to achieve quick results. However, this involves using advanced computational linear algebra methods, which may require adjusting numerous parameters. In this work, for instance, we use Krylov solvers (PCG and GMRES) combined with ILU preconditioners for the original system, along with the implementation of a multiscale method. The objective is to evaluate the best configurations for solving systems in geomechanical simulations.

2 Geomechanics Simulator

The geomechanics simulator used here is described in [1]. It is a one-way geomechanics simulator that initializes the stress in the domain and calculates its modifications during field exploitation. The pore pressure and temperatures come from the flow reservoir simulator. Its viscoplastic methods require one assembly of the rigid matrix, which is a huge advantage over multiscale methods because they do not need to recalculate it when the matrix changes. The simulator runs in a distributed memory paradigm with MPI and can use several nodes of a computer cluster. It is capable of running models of hundreds of millions of elements.

It uses by default a Preconditioned Conjugate Gradient (PCG) with a multiscale preconditioner together with an ILU fine preconditioner. The multiscale preconditioner is capable of reducing the low-frequency errors and the ILU is used to reduce the high-frequency errors. These two preconditioners are combined additively:

$$\mathbf{M}^{-1} = \mathbf{M}_{\text{ilu}}^{-1} + \mathbf{M}_{\text{ms}}^{-1} \quad (1)$$

Where $\mathbf{M}_{\text{ilu}}^{-1}$ is a fine ILU preconditioner and $\mathbf{M}_{\text{ms}}^{-1}$ is the multiscale preconditioner. Combining the two preconditioners this way is important to keep it symmetric, due to PCG requirements. Using these two combined with a multiplicative strategy would need to change the Krylov algorithm that supports nonsymmetric matrices like GMRES or BICGSTABSL. Equation 2 illustrates the method for combining two preconditioners using a multiplicative strategy.

$$\mathbf{M}^{-1} = \mathbf{M}_{\text{ms}}^{-1} + \mathbf{M}_{\text{ilu}}^{-1}(\mathbf{I} - \mathbf{K}^h \mathbf{M}_{\text{ms}}^{-1}) \quad (2)$$

Here, \mathbf{K}^h represents the matrix of the linear system. Note that the multiplicative preconditioner is more expensive than the additive because it needs to calculate a matrix-vector multiplication.

The matrix assembly, linear solver algorithms, matrix multiplication, and other calculations are implemented in the linear algebra framework described in [2]. It has support for shared and distributed memory environments and has specific functionalities for the geomechanics problem such as: Blocked Compressed Row Sparse matrix (BCSR), triple matrix product, Krylov Linear Solvers Algorithms (GMRES and PCG), and Sparse Symmetric matrices.

3 Multiscale Preconditioner

In this paper, the linear solver has its convergence accelerated by using a fine preconditioner and a coarse preconditioner. The fine preconditioner is applied to the original linear system generated by the geomechanical simulator, and all the cases were executed with ILU with fill level 1. The coarse preconditioner uses a multiscale method described in [3]. The multiscale method generates a coarser version of the original grid which results in a matrix with fewer equations. This kind of method is efficient in reducing the low-frequency errors in the linear solver convergence. Original grid matrices and vectors will be followed by superscript h and coarse grid by superscript H. Figure 1 shows a diagram of how the multiscale method works.

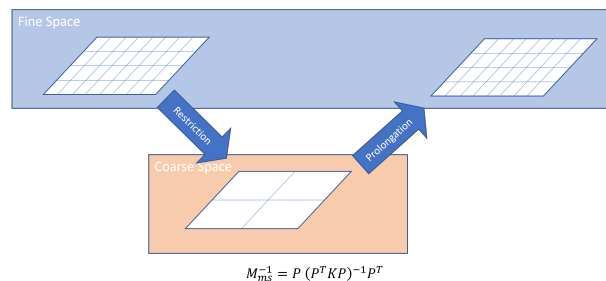


Figure 1. Schematics of the application of the Multiscale Preconditioner.

To construct the coarser version, it is necessary to build two operators to project and return vectors between the two spaces (fine and coarse) that are: Prolongator (\mathbf{P}) and Restriction (\mathbf{R}). To build the prolongator, local

problems are solved to calculate basis functions in the coarse grid. In the method used $\mathbf{R} = \mathbf{P}$. After the calculation of these operators, the coarse system matrix is calculated with the Galerkin method ($\mathbf{K}^H = \mathbf{P}^T \mathbf{K}^h \mathbf{P}$). The triple matrix product necessary for this application is implemented in the linear algebra library presented in [2]. The application of the multiscale preconditioner on a residue \mathbf{r}^h consists of three steps:

1. Restrict \mathbf{r}^h to the coarse grid ($\mathbf{r}^H = \mathbf{P}^T \mathbf{r}^h$)
2. Solve the coarse linear system ($\mathbf{K}^H \mathbf{x} = \mathbf{r}^H$)
3. Prolongate \mathbf{x} to the fine grid ($\mathbf{r}^h = \mathbf{P} \mathbf{x}$)

The second step consists of solving a linear system; for this it was also used the Preconditioned Conjugate Gradient (PCG) with ILU. Unlike the fine system ILU, the coarse ILU was set to fill level 0 or 1.

4 Results

4.1 Models Description

The results presented in this paper were made using three variations of the same model with different discretizations, as well as a real model of an oil field. Table 1 shows the size of each model executed. The UNISIM-I model is based on the Namorado Oil Field, a brownfield located at Campus Basin in Brazil. Three different discretizations were used to compare models on a scale from 7 million elements to 60 million elements. The PRESALT model represents a real oil field of the Brazilian Presalt located in the Santos Basin.

Table 1. Number of elements of each model.

Model	Size (Million of Elements)
UNISIM-I 7MM	7
UNISIM-I 35MM	35
UNISIM-I 60MM	60
PRESALT	93

4.2 Hardware Configuration and Parameters

The results were executed on a cluster with the following configuration:

- Two Intel Xeon Gold 6148 - 2.4GHz - 20 cores per processor,
- Memory of 384 GB,
- OS: Red Hat Enterprise Linux (RHEL) 8
- InfiniBand EDR Network.

The executions compare the solver configurations, changing the following parameters:

- Coarse Solver Tolerance: The tolerance of the conjugate gradient convergence of the coarse linear solver. The tolerance used is compared with the relative preconditioned norm
- Coarse Factor: To build a coarse grid, the multiscale method merges fine elements to build a coarse element. The coarse factor is how many elements will be merged in each direction to construct the new element.
- Preconditioner float point precision: The Fine and Coarse ILU preconditioner executed with single precision (32 bits) and double precision (64 bits) operations
- Combinative Preconditioner Strategy: The fine ILU was combined with the multiscale preconditioner with additive and multiplicative strategies.

4.3 Coarse Solver Tolerance Comparison

First, we compare how the coarse solver tolerance can improve the time of the total solver. For this we used the values $\{10^{-1}, 10^{-2}, 10^{-4}, 10^{-6}\}$. Figure 2 shows the comparison between these configurations on UNISIM-I 7MM. The other parameters are constants with values: Coarse factor of $4 \times 4 \times 4$, fine ILU with double precision preconditioner and additive strategy to combine the preconditioners.

Figure 2 shows the increase in the total linear iterations of the solver and the decrease in the wall clock time

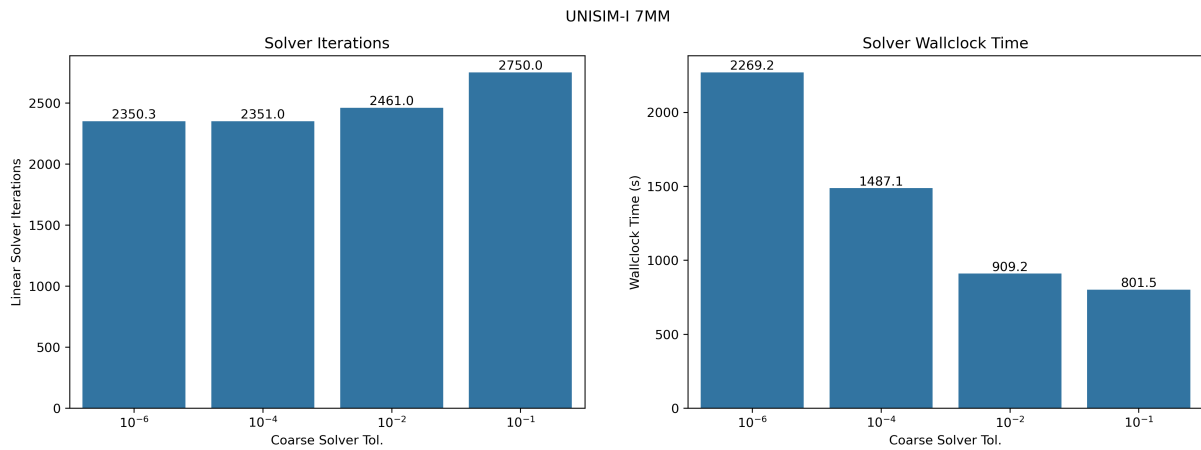


Figure 2. Influence of coarse tolerance on the number of linear iterations and the total solver time at UNISIM-I 7MM.

with the increase in the coarse solver tolerance for UNISIM-I 7MM. This is explained because using a larger tolerance makes the coarse solution less accurate, which means a weaker preconditioner, but also fewer coarse conjugate gradient iterations. Because the increase in total iterations is less pronounced than the reduction in time of application of the multiscale preconditioner, the result is a faster solution of the linear system.

Figure 3 shows the same result for the other models. The executions were performed using four nodes in the cluster with 160 MPI processes. For these models, 10^{-1} and 10^{-2} are better than 10^{-4} when comparing the wallclock time (around 31–33% for PRESALT, 17–25% for UNISIM-I 35MM, and 25–30% for UNISIM-I 60MM). The 10^{-2} tolerance is better than 10^{-1} in the UNISIM-I models and worse on PRESALT, but their performances are close.

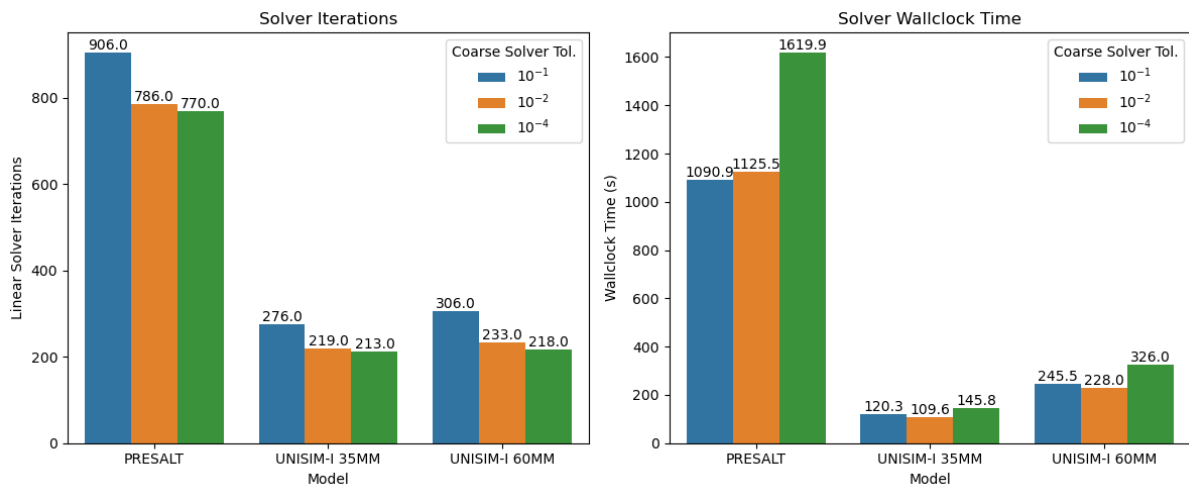


Figure 3. Influence of coarse tolerance in the UNISIM-I 35MM, UNISIM-I 60MM, and PRESALT using four nodes of the cluster.

4.4 Coarse Factor Comparison

When using the multiscale method, one important parameter is how the fine elements will be merged to build the coarse element. The strategy used on the Geomechanics Simulator is simple; the user inputs how many elements in each direction (x, y, z) the simulator should merge. Tables 2 and 3 show the results of simulations for

different coarse factors. The same coarse factor was used in all three directions.

Table 2. Different Coarse Factors in UNISIM-I 35MM using 2 nodes, coarse tol. 10^{-2} and single precision ILU(1)

Coarse Factor	Total Simulation (s)	Linear Solver (s)	Linear Solver Iterations	M_{ms}^{-1} apply (s)	M_{ms}^{-1} apply per iteration (s)
2	648.1	386.6	191	270.0	1.413
4	412.9	164.9	216	68.6	0.318
8	422.9	157.4	267	47.0	0.176
16	1240.1	249.2	447	77.9	0.174
20	1644.0	380.4	447	147.9	0.331

Table 3. Different Coarse Factors in PRESALT using 4 nodes, coarse tol. 10^{-2} and single precision ILU(1)

Coarse Factor	Total Simulation (s)	Linear Solver (s)	Linear Solver Iterations	M_{ms}^{-1} apply (s)	M_{ms}^{-1} apply per iteration (s)
2	1942.3	1567.0	561	1296.9	2.312
4	787.3	423.2	828	189.5	0.229
8	838.6	447.3	1220	125.8	0.103
16	3182.0	852.5	2282	268.3	0.118
20	3167.8	846.7	2282	261.9	0.115

As expected, the number of iterations increases with the increase in the coarse factor because the multiscale solution is less accurate when the coarsening is higher. However, the application of the multiscale preconditioner (M_{ms}^{-1}) becomes cheaper because increasing the coarse factor makes K^H smaller. These two effects explain why greater coarse factors start reducing the total time of the linear solution and then start to increase. For UNISIM-I 35MM and PRESALT models the best results are reached with a coarse factor of $4 \times 4 \times 4$ when looking for the total simulation time. But the performance of $8 \times 8 \times 8$ coarse factor is quite similar. All other simulations in this paper were executed with the coarse factor of $4 \times 4 \times 4$.

4.5 Comparison of Preconditioner Precision

The Linear Algebra Library used has support for ILU preconditioners in single precision. The main advantage of this is that many available hardware perform single-precision calculations twice as fast as double precision calculations. Figures 4, 5, and 6 show the total solver iterations of the linear solver, the total walltime solution, and the respective time of the ILU application.

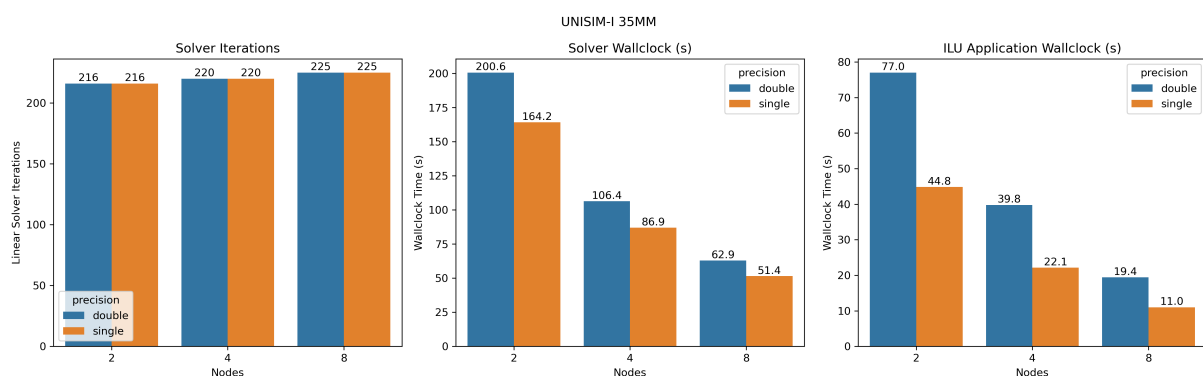


Figure 4. Comparison of using Fine ILU application with single and double precision on UNISIM-I 35MM

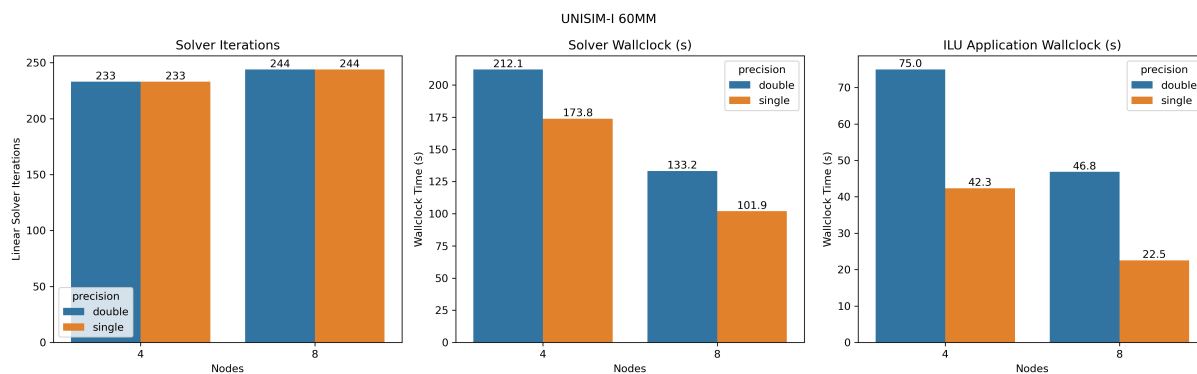


Figure 5. Comparison of using Fine ILU application with single and double precision on UNISIM-I 60MM

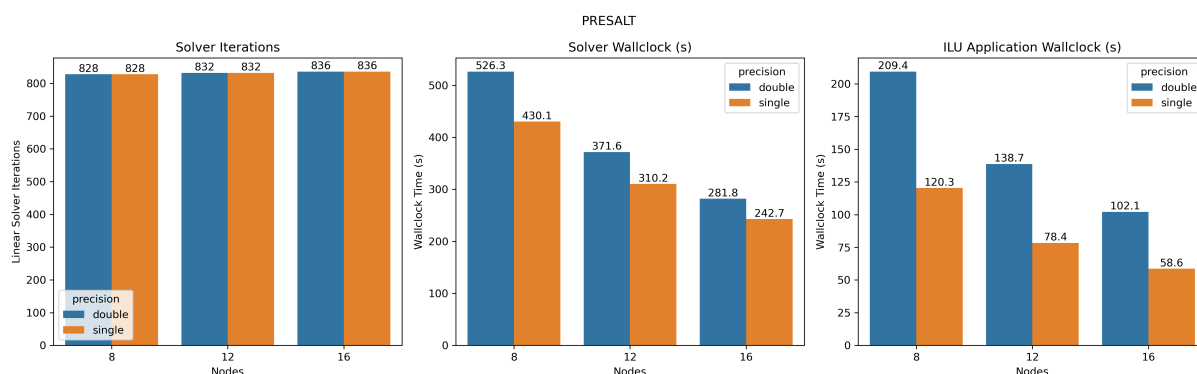


Figure 6. Comparison of using Fine ILU application with single and double precision on PRESALT

It is important to note that the number of iterations does not increase using single precision, which is very good because the time to apply ILU is reduced using the single precision. Looking at the ILU application, the single precision version is 1.7x faster, making the total solver time execute with a speed-up between 1.16x and 1.31x.

4.6 Comparison of Additive Against Multiplicative Preconditioners

Figure 7 shows the comparison using the combination of preconditioners with multiplicative and additive strategies, one using ILU (0) and the other ILU (1) in the PCG of the coarse linear system. The multiplicative strategy uses the right preconditioning. Using GMRES, the solver only converged with a coarse tolerance equal to 10^{-6} using the additive strategy and with 10^{-4} and 10^{-6} using the multiplicative strategy, which is already a disadvantage compared to PCG. The number of iterations is lower when using the multiplicative strategy, and it reduces the number of iterations around 8%, also reducing the total solver time. Using ILU(0) is a better option than ILU(1) in the coarse linear system.

When comparing the same configuration but using the conjugate gradient with the additive preconditioner strategy, the results below show that the performances are quite similar. But, conjugate gradient has the advantage of using a lower coarse tolerance, and this accelerates the convergence. Figure 8 shows this comparison. The conclusion is that PCG with a coarse tolerance of 10^{-6} is faster to use and has a similar performance to GMRES with a multiplicative strategy with the same tolerance, but it is 2.26 times faster when the coarse tolerance is 10^{-2} .

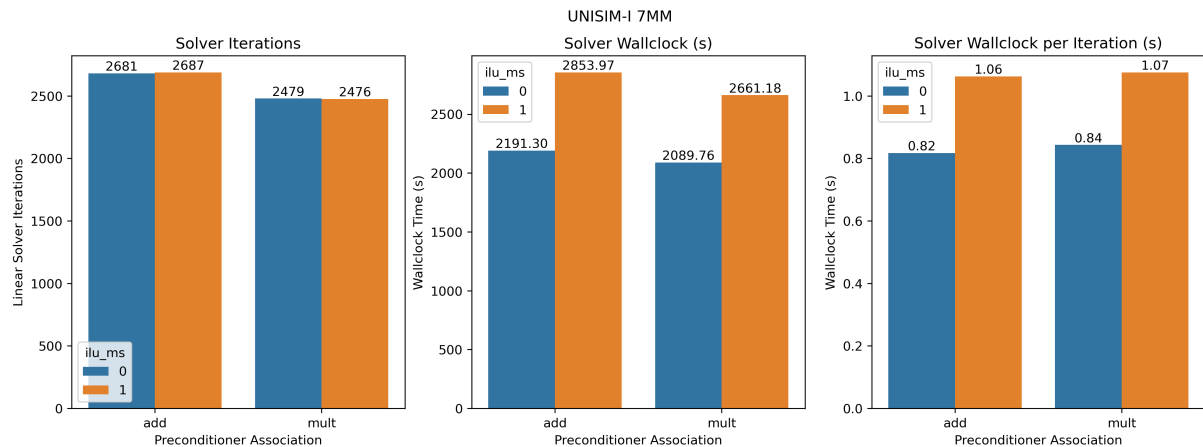


Figure 7. Comparison of Additive against Multiplicative Preconditioner. The ILU fill of the coarse system is also tested with values 0 and 1. GMRES with right preconditioning was used on both cases.

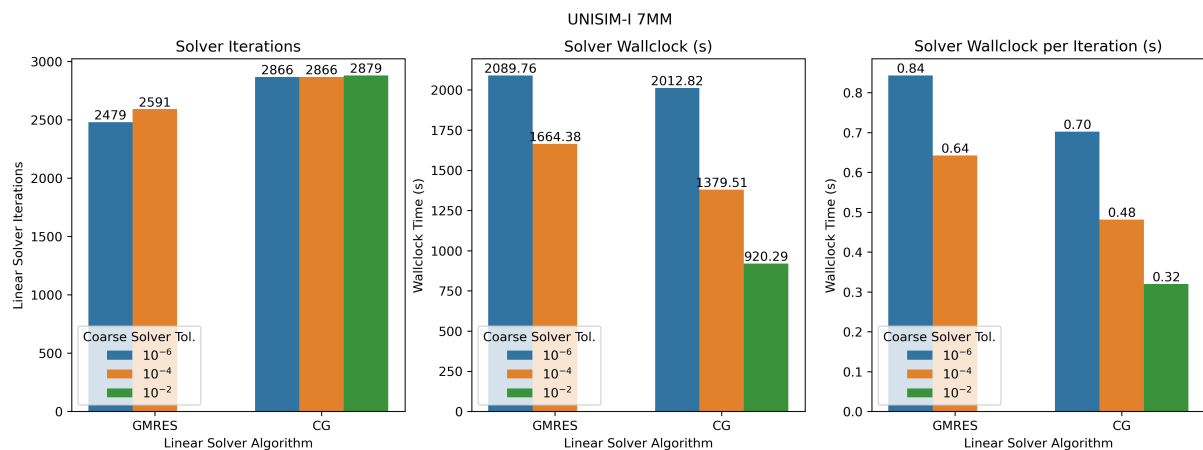


Figure 8. Comparison of CG with additive preconditioner and GMRES with multiplicative preconditioner

4.7 Solver Scalability

Figure 9 shows the speedup of the linear solver using optimized parameters for each of the three bigger models. Each simulation starts from the minimum nodes needed to execute them due to RAM memory needs and ends with 16 nodes.

One major effect that explains why the scalability of the solver cannot be ideal is that the number of iterations increases due to the increase of domains when using the Block Jacobi preconditioner. But the result shows speed-up of 1.77x (ideal 2x) in PRESALT, 5.23x (ideal 8x) in UNISIM-I 35MM and 3.18x in UNISIM-I 60MM.

5 Conclusions

The parameter setting of the linear solver is essential to improve the performance of the solution time. Tuning the coarse rate of the multiscale preconditioner can reduce the simulation time, and on our test, the best coarsening ratios were $4 \times 4 \times 4$ and $8 \times 8 \times 8$. Selecting the correct coarse solver tolerance in the multiscale preconditioner can accelerate the results up to 2.83 times faster (on UNISIM-I 7MM). The fine ILU(1) singles precision application on the geomechanics problem did not increase the number of PCG iterations and reduced the time of the solver with a speedup of 1.7x. Using the additive preconditioner is possible to accelerate the performance of the solver because PCG was able to converge with a coarse tolerance lower than that of GMRES.

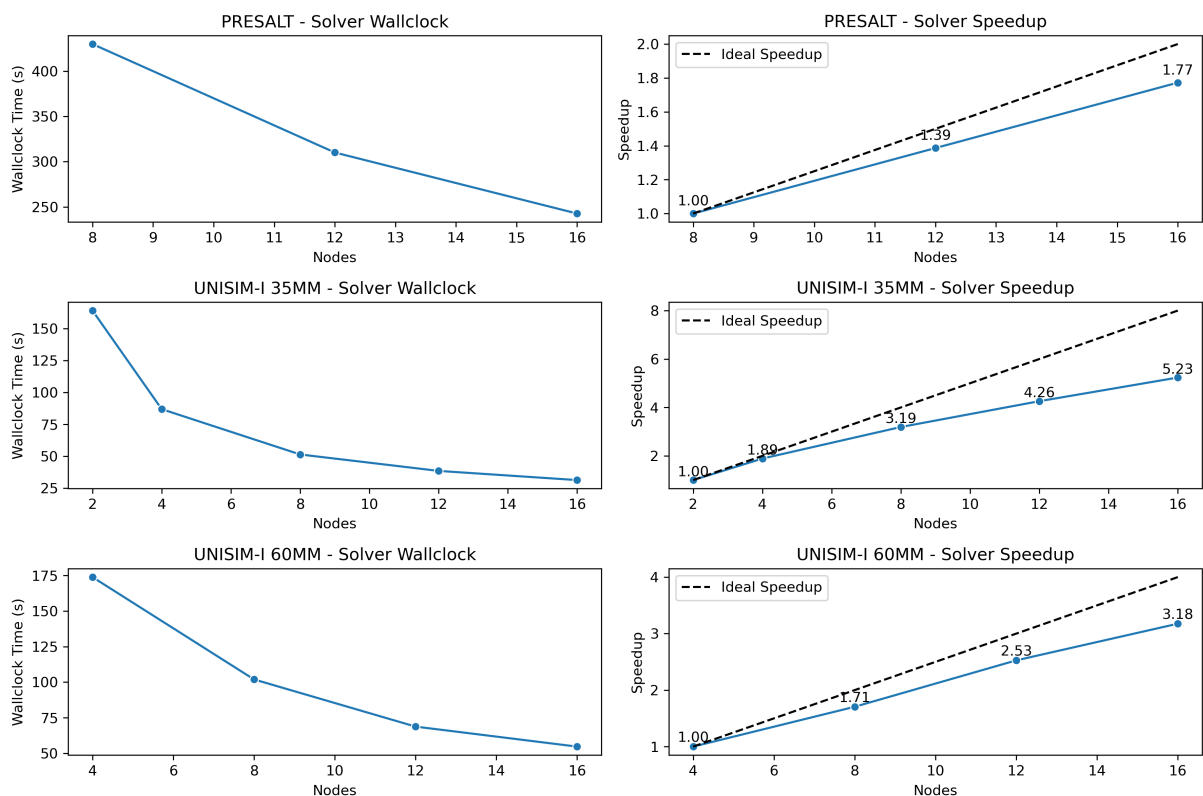


Figure 9. Scalability of the three bigger models executed with the best parameters found (Coarse Tol. 10^{-2} , Coarse Factor $4 \times 4 \times 4$ and single precision fine ILU(1))

Acknowledgements. Thanks to Petrobras for sponsoring and for the permission to publish this work.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] M. O. de Figueiredo, L. C. de Sousa Junior, J. R. Rodrigues, L. B. dos Santos, L. S. Gasparini, R. F. do Amaral, and R. J. de Moraes. A parallel viscoplastic multiscale reservoir geomechanics simulator. *Upstream Oil and Gas Technology*, vol. 11, pp. 100095, 2023.

[2] L. Gasparini, J. R. Rodrigues, D. A. Augusto, L. M. Carvalho, C. Conopoima, P. Goldfeld, J. Panetta, J. P. Ramirez, M. Souza, M. O. Figueiredo, and V. M. Leite. Hybrid parallel iterative sparse linear solver framework for reservoir geomechanical and flow simulation. *Journal of Computational Science*, vol. 51, pp. 101330, 2021.

[3] N. Castelletto, H. Hajibeygi, and H. A. Tchelepi. Multiscale finite-element method for linear elastic geomechanics. *Journal of Computational Physics*, vol. 331, pp. 337–356, 2017.