Workload scheduling comparison in a full waveform inversion distributed memory implementation

Carla Santana (UFRN), Tiago Barros (UFRN), Idalmis Milian (UFRN), Calebe Bianchini (Mackenzie Presbyterian University) and Samuel Xavier-de-Souza (UFRN)

Abstract

Full waveform inversion has gained attention in the geophysics community as an efficient technique for the determination of seismic subsurface velocity models. However, the full waveform inversion technique is computationally intensive, both in terms of execution time and memory usage. In order to obtain shorter execution times, usually the full waveform inversion algorithm is implemented using the concept of parallel programming. For the implementation in distributed memory environments, it is important to define the tasks (processes) distribution among the computational nodes, which is known as workload scheduling. The workload scheduling approach directly impacts on computationally intensive algorithms, such as the seismic wave propagation, highly employed in full waveform inversion. For a large number of nodes and tasks, the full waveform inversion might suffer of scalability issues. In this work we compare two workload scheduling approaches for a parallel 2D acoustic full waveform inversion implementation running in a distributed memory system: the centralized dynamic and the decentralized static. The comparison is made by presenting speedup and efficiency plots for different model sizes and number of nodes. The scalability and performance analysis showed that the decentralized static algorithm is highly scalable and efficient in distributed memory systems.

Introduction

Full waveform inversion (FWI) has been highlighted in the geophysics community as an efficient technique for mapping the Earth's subsurface, in the search for oil and gas reservoirs. The FWI technique is used to obtain quantitative information from seismograms (Virieux and Operto, 2009), for instance, the seismic wavefront velocity model in the subsurface. It is based on numerical optimization methods, which employ the misfit between the observed seismograms and modeled data and use it to iteratively estimate this velocity model. The modeled data in FWI is mostly generated with the use of finite difference methods (FDM) (Claerbout and Doherty, 1972), which are computationally intensive, both in terms of execution time and memory usage.

In order to obtain shorter execution times, usually the FWI algorithm is implemented using the concept of parallel programming. In this case, a parallel algorithm might be implemented to be executed in shared and/or distributed memory environments. Usually the employed parallelization technique consists of executing the seismic shots on different nodes of a computational cluster for distributed memory parallelization; and propagating the seismic wave by dividing the propagation domain among the threads of a single node, employing the concept of shared memory parallelization.

For distributed memory environments, it is important to define the tasks (processes) distribution among the nodes, which is known as workload scheduling (Tchiboukdjian et al., 2013). One of the most used scheduling techniques is the master-slave algorithm, in which the tasks are distributed by one process (namely *the master*) to be executed by the remaining ones (namely *the slaves*) (Sahni and Vairaktarakis, 1996). This approach is a centralized scheduling type, with the tasks being dynamically distributed to the nodes. In this scheduling technique, the master might be overloaded, since all decisions are centralized on it. Depending on the application type, a possible alternative to the master-slave scheduling is to decentralize the workload distribution by initially sharing the tasks among the processes. In this later scheduling type, the workload is statically distributed to each process, which evaluates the received tasks and shares its results with all processes.

The workload scheduling approach directly impacts on computationally intensive algorithms, such as the seismic wave propagation (Tesser et al., 2014), highly employed in FWI. For a large number of nodes and tasks, FWI might suffer of scalability issues. In this work we investigate the workload scheduling in FWI. We compare the master-slave centralized dynamic scheduling (CD) approach to a decentralized and static (DS) one, for the 2D acoustic FWI, using distributed memory parallelization and implemented with the message passing interface (MPI) library. For this comparison, we employ a synthetic velocity model and present speedup and efficiency plots, generated with different model sizes and number of nodes.

FWI workload scheduling

In a simple manner, FWI consists into finding the model vector $\mathbf{m}^*$ which minimizes the following equation:

$$\mathbf{m}^* = \arg\min_{\mathbf{m}} ||g(\mathbf{m}) - \mathbf{d}||_2^2, \quad (1)$$

where $\mathbf{d}$ are the observed seismic data and $g(\mathbf{m})$ is the operator representing the process of artificially modeling the data, usually performed by propagating seismic waves with numerical methods (Virieux et al., 2009). The function being minimized is known as misfit and can be represented as $f = ||g(\mathbf{m}) - \mathbf{d}||_2^2$. This minimization problem might be solved iteratively, with the use of the initial condition $\mathbf{m}_0$, by an optimization method. In FWI, one of the most used optimization methods is the gradient approach (Tarantola, 2005), which gives:

$$\mathbf{m}_{j+1} = \mathbf{m}_j - \alpha_j \mathbf{g}_j, \tag{2}$$

where the variable $j \geq 0$ defines the iteration number, $\mathbf{m}_j$ is the model vector at the j-th iteration and α_j is a scalar that defines the step size in the direction given by the gradient $\mathbf{g}_j$.

In FWI, the gradient optimization approach is employed to estimate the velocity model of the seismic wavefront in the subsurface. Usually, the relationship between the model in (1) and the velocity model $\mathbf{v}$ is given by $m_n = 1/v_n^2$. Note that these vectors are linearized matrices, with n representing the space coordinates x and z, in two-dimensions. At each algorithm iteration, the gradient ($\mathbf{g}_j$) is computed with the adjoint-state method (Plessix, 2006), which uses the misfit (f_j) between the recorded and modeled data to update the velocity model employed in the following algorithm iterations. The seismic data d is composed by several seismograms, generated in different seismic shots. The gradient and misfit are obtained by summing the gradient and misfit evaluated in each different seismic shot. Due to several wave equation calculations, FWI presents high computational costs, which can be reduced by paralleling the synthetic seismogram evaluation. As previously discussed in the introduction, the shots are allocated to the nodes by a workload scheduling technique, which might be centralized or decentralized and static or dynamic. At the end of each FWI algorithm iteration, the f_j and $\mathbf{g}_j$ quantities, computed on the different nodes, are added and used to update the velocity model.

Centralized dynamic scheduling

In this work, for the implementation of the master-slave FWI parallel algorithm, with the CD scheduling of seismic shots, a process (master) is chosen to distribute the shots among the other processes (slaves). The N_p processes are divided into one master process, P_master, and $N_\text{p} - 1$ slave processes, P^k_slave, with k going from 1 to $N_\text{p} - 1$. The master assigns to each slave one shot index (i) at a time, from a total of N_s shots, i.e., $i = 0, \ldots, N_\text{s} - 1$. On its turn, the slave computes the quantities f_j^i and $\mathbf{g}_j^i$ for the assigned shot and returns the results back to the master. After receiving these results, the master assigns a different shot to the slave. This procedure is repeated for all the slaves, until there are no remaining shots. Finally, the master is also in charge of summing the misfit (f_j^i) and gradient ($\mathbf{g}_j^i$) from the different shots, in order to update the f_j and $\mathbf{g}_j$ quantities, in the j-th FWI iteration. These summations are given by

$$f_j = \sum_{i=0}^{N_\text{s}-1} f_j^i \tag{3}$$

and

$$\mathbf{g}_j = \sum_{i=0}^{N_\text{s}-1} \mathbf{g}_j^i. \tag{4}$$

As a drawback, centralizing the decisions in only one node might pose a bottleneck to the master, since the communication between the nodes increases with the growth of slave nodes. In Figure 1 we illustrate the master-slave communication in the CD workload scheduling.

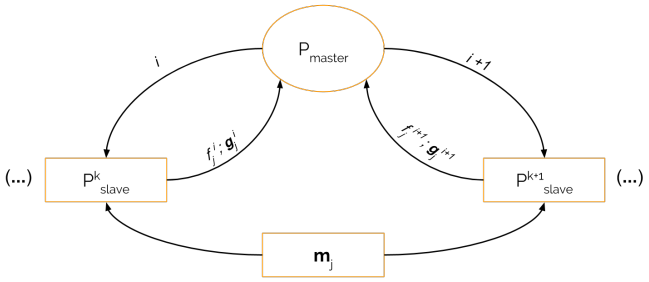

Figure 1: CD scheduling structure.

Decentralized static scheduling

In our implementation of the 2D acoustic FWI with DS scheduling, the misfit and gradient evaluation are decentralized and equally distributed between the N_p processes P_k, with k going from 0 to $N_\text{p} - 1$. For the j-th iteration of the FWI optimization algorithm, each process parallel evaluates part of the quantities f_j and $\mathbf{g}_j$, for one or more shots, considering the model $\mathbf{m}_j$. The shots assignment is performed according to the process number. The quantities evaluated by each process are given by:

$$f_j^k = \sum_{i=N_\text{s}^{k-1}}^{N_\text{s}^k - 1} f_j^i \tag{5}$$

and

$$\mathbf{g}_j^k = \sum_{i=N_\text{s}^{k-1}}^{N_\text{s}^k - 1} \mathbf{g}_j^i, \tag{6}$$

where N_s^k is the number of the last shot assigned to the k-th process and f_j^i and $\mathbf{g}_j^i$ are the misfit and gradient from the i-th shot. Note that the process P_0 computes the shots in the indexes range of $k = 0, \ldots, N_\text{s}^0 - 1$, i.e., $N_\text{s}^{-1} = 0$.

At the end of the j-th iteration, the quantities f_j and $\mathbf{g}_j$ are evaluated by

$$f_j = \sum_{k=0}^{N_\text{p}-1} f_j^k \tag{7}$$

and

$$\mathbf{g}_j = \sum_{k=0}^{N_\text{p}-1} \mathbf{g}_j^k \tag{8}$$

and broadcasted to all the processes. Finally, each process computes its own model $\mathbf{m}_{j+1}$, for the $j+1$-th iteration. However, since the workload is statically distributed at the beginning of each FWI iteration, the DS scheduling might present workload unbalance due to several factors, such as heterogeneous systems, uneven shots distribution, etc. The DS scheduling procedure is illustrated in Figure 2. We implemented the DS scheduling with the MPI library and the summation and broadcasting operations were performed with the function *MPI_Allreduce*.

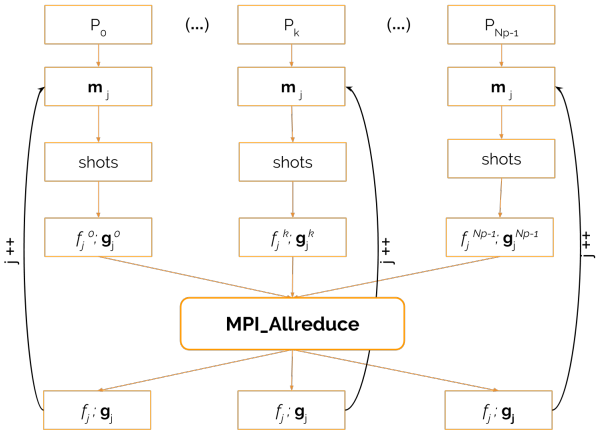

Figure 2: DS scheduling structure.

Metrics for scalability analysis

The main objective of writing parallel programs is, generally, to increase the code execution performance. In order to achieve that, some metrics might be used to compare different parallelization strategies. The most used ones are the speedup and efficiency, which are defined as follows.

Speedup

The speedup is the ratio between the gain of a parallel algorithm and its sequential equivalent and is defined by

$$S_p \simeq \frac{T_s}{T_p}, \tag{9}$$

where S_p is the speedup, T_s is the execution time for the sequential algorithm and T_p is the execution time for the parallel algorithm. Considering p the total number of processors, ideally, it is expected that $S_p = p$, which means that the parallel program runs p times faster than the sequential.

Efficiency

The efficiency is the ratio between the speedup and the quantity of processors and is defined by

$$E = \frac{S_p}{p}. \tag{10}$$

This metric can be viewed as the speedup normalized by the number of processors and indicates how much each processor is used. The ideal speedup is p, resulting in an ideal efficiency of 1.

Numerical experiments

In order to analyze the performance of the FWI CD and DS scheduling algorithms, for different velocity model sizes and number of nodes, we created a synthetic layered velocity model with no lateral velocity variation. We generated a base model, named $\mathbf{v}_1$, with dimensions 525×200, in x and z coordinates, respectively, and a sample spacing of 10 m in both coordinates. As there is no lateral velocity variation, we generated bigger versions of $\mathbf{v}_1$ (named $\mathbf{v}_2$, $\mathbf{v}_3$ and $\mathbf{v}_4$) by increasing the number of points in the x coordinate. We show the characteristics of these velocity models in Table 1 and, in Figure 3, we illustrate $\mathbf{v}_1$. In Figure 4 we show one common-shot gather artificially computed with $\mathbf{v}_1$.

Table 1: Wave propagation parameters for different velocity models

Velocity model	Mesh size	Model size	Memory allocated in propagation
$\mathbf{v}_1$	525×200	0,42MB	840MB
$\mathbf{v}_2$	1050×200	0,82MB	1680MB
$\mathbf{v}_3$	2100×200	1,7MB	3360MB
$\mathbf{v}_4$	4200×200	3,4MB	6720MB

Regarding the FWI algorithm, we used the sample time interval of 1 ms and we executed the wave propagation for the total time of 1 s, resulting in seismic seismograms with 1001 time samples. The FWI was executed with the total of 2048 shots and a shot increment of 100 m. For the parallel implementation, we used the OpenMPI version 1.8.5 and GCC version 4.9.3. The code was executed in the computational cluster Yemoja, which has a total of 856 nodes, 132TB of RAM memory and is located in Salvador-BA, Brazil, in SENAI-CIMATEC.

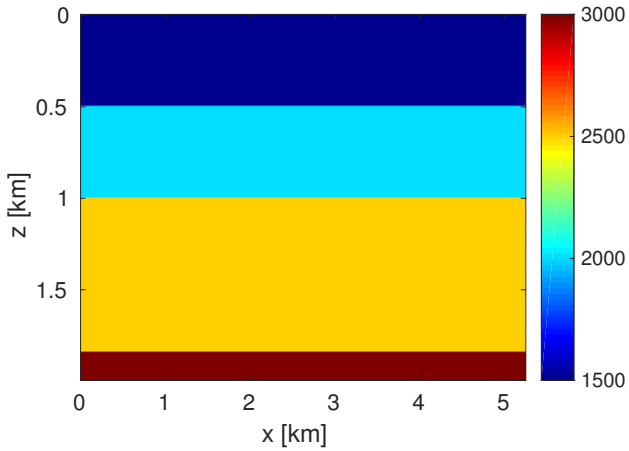

Figure 3: Illustration of one of the employed velocity model ($\mathbf{v}_1$).

We analyzed the code performance considering the speedup and the efficiency. The speedup and efficiency graphics were generated for 2, 4, 8, 16, 32, 64, 128 and 256 nodes. In order to perform a more detailed scalability analysis, the FWI was executed using four different model sizes, from $\mathbf{v}_1$ to $\mathbf{v}_4$. For each different model, FWI was executed five times and we used the median to plot the speedup and efficiency.

In Figure 5, we show the CD and DS algorithms speedup. The CD speedup for smaller problem sizes is closer to the ideal one, when compared to the speedup for bigger problem sizes. This is mostly because the master presents more communication with the slaves in bigger problems, which results in bigger overheads. This does not happen for the DS scheduling, since the communication in DS occurs only at the end of each FWI iteration.

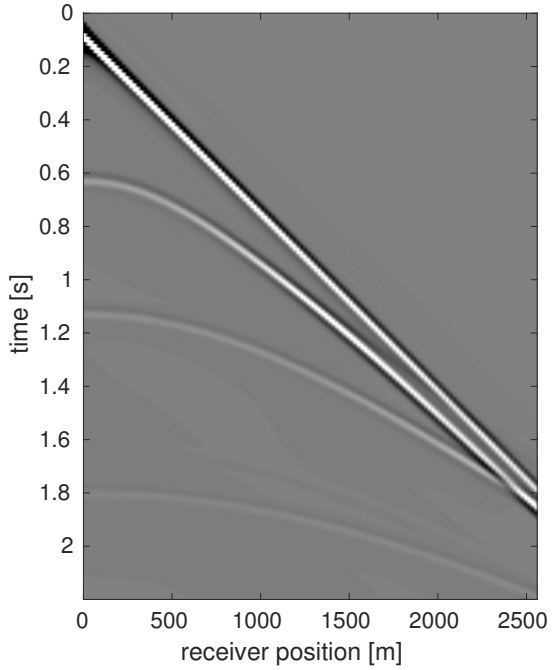

Figure 4: Illustration of one common-shot gather from the seismic data, computed with $\mathbf{v}_1$.

The CD efficiency depends on the communication overhead that is related with the model size and the quantity of nodes: for a large number of nodes in a big model, the efficiency becomes worst, as the one for 128 and 256 nodes in the models $\mathbf{v}_3$ and $\mathbf{v}_4$. The DS efficiency is more well-behaved, as shown in Figure 6. This is due to the fact that DS scheduling presents less communication than the CD one.

Conclusions

FWI technique is computationally intensive and requires efficient scheduling methods. In this work we compared two different workload scheduling approaches for a 2D acoustic FWI implementation: the centralized dynamic (CD) and decentralized static (DS). Our results, for a synthetic velocity model, showed better scalability for the DS scheduling. The DS also presented efficiency values nearly constant, whereas the CD presented better efficiency for fewer nodes and smaller models. For more realistic problems, such as the 3D FWI, we expect a similar behaviour. In other words, the larger the number of nodes the more advantageous would the DS scheme be. Nevertheless, producing the same results for a 3D code would require either much more computing time or a much larger machine.

Acknowledgements

The authors gratefully acknowledge support from Shell Brazil through the project “Novos Métodos de Exploração Sísmica por Inversão Completa das Formas de Onda” at the Federal University of Rio Grande do Norte (UFRN) and the strategic importance of the support given by ANP through the R&D levy regulation. This research was supported by the High-Performance Computing Center at UFRN (NPAD/UFRN).

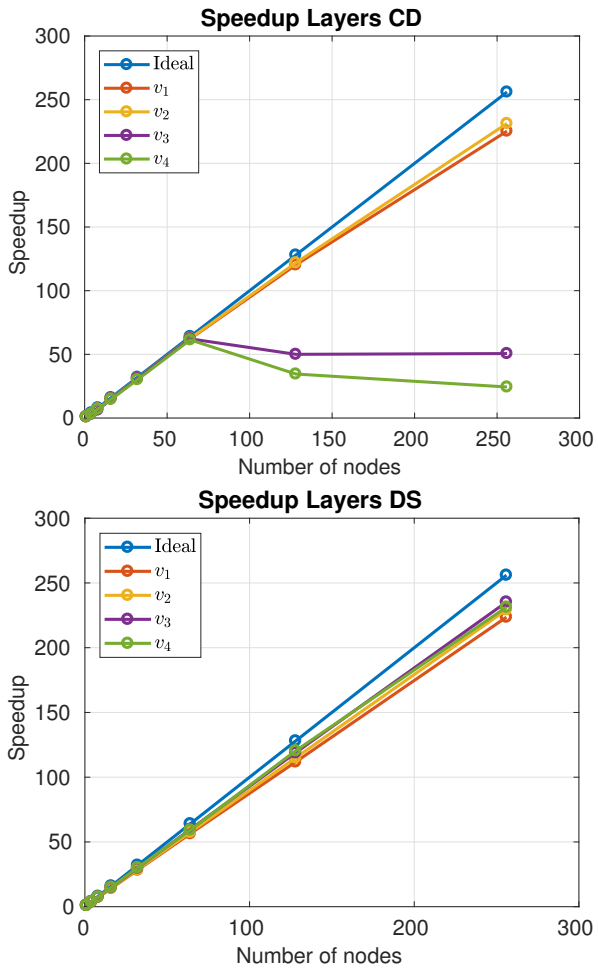

Figure 5: Speedup for the layered velocity model.

References

Claerbout, J. F., and S. M. Doherty, 1972, Downward continuation of moveout-corrected seismograms: Geophysics, **37**, 741–768.

Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: Geophysical Journal International, **167**, 495–503.

Sahni, S., and G. Vairaktarakis, 1996, The master-slave paradigm in parallel computer and industrial settings: Journal of Global Optimization, **9**, 357–377.

Tarantola, A., 2005, Inverse problem theory and methods for model parameter estimation: siam, **89**.

Tchiboukdjian, M., N. Gast, and D. Trystram, 2013, Decentralized list scheduling: Annals of Operations Research, **207**, 237–259.

Tesser, R. K., L. L. Pilla, F. Dupros, P. O. A. Navaux, J.-F. Méhaut, and C. Mendes, 2014, Improving the performance of seismic wave simulations with dynamic load balancing: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on, IEEE, 196–203.

Virieux, J., and S. Operto, 2009, An overview of

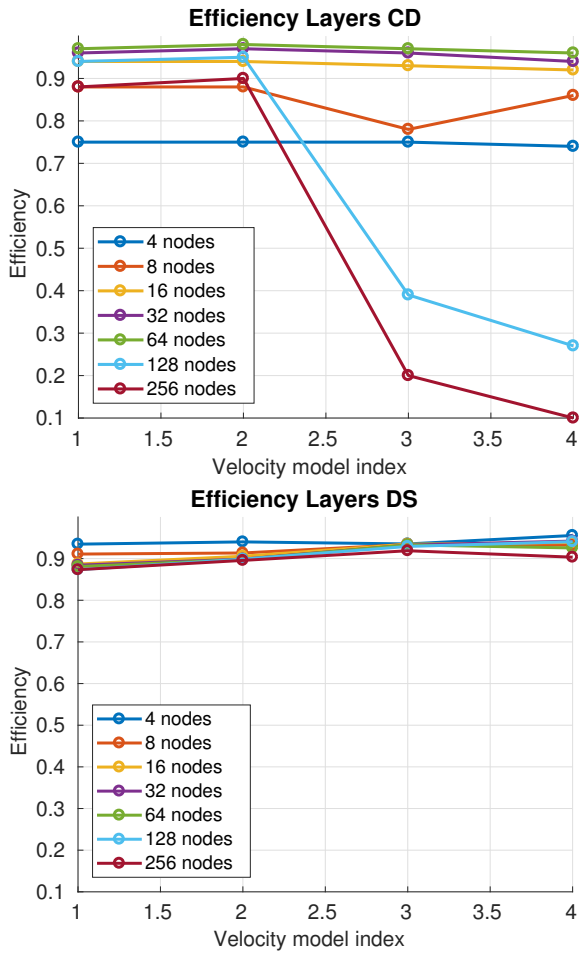

Figure 6: Efficiency for the layered velocity model.

full-waveform inversion in exploration geophysics: Geophysics, **74**, WCC1–WCC26.

Virieux, J., S. Operto, H. Ben-Hadj-Ali, R. Brossier, V. Etienne, F. Sourbier, L. Giraud, and A. Haidar, 2009, Seismic wave modeling for seismic imaging: The Leading Edge, **28**, 538–544.