UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Intra-node and Inter-node load balancing and other scalable approaches for high-performance seismic processing

Ítalo Augusto Souza de Assis

Thesis submitted in partial fulfillment of the requirements for the degree of **Doctor of Science** in the Graduate Program in Electrical and Computer Engineering of UFRN (area of concentration: Computer Engineering)

PPgEEC order number: D259
Natal, RN, Brazil. October 2019

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Assis, Italo Augusto Souza de.
Intra-node and Inter-node load balancing and other scalable approaches for high-performance seismic processing / Italo Augusto Souza de Assis. - 2019.
109 f.: il.

Tese (doutorado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Natal, RN, 2019.
Orientador: Prof. Dr. Samuel Xavier de Souza.

1. Multi-scale waveform inversion - Tese. 2. Scalability - Tese. 3. Load balancing - Tese. 4. Work-stealing - Tese. 5. Auto-tuning - Tese. 6. Efficiency - Tese. I. Souza, Samuel Xavier de. II. Título.

RN/UF/BCZM CDU 550.344

Elaborado por Ana Cristina Cavalcanti Tinôco - CRB-15/262

Abstract

Seismic modeling, reverse time migration (RTM), and multi-scale waveform inversion (MFWI) are three of the most important techniques in seismic surveying. Seismic modeling simulates the wave propagation, RTM generates an image of the subsurface, and MFWI produces a wave propagation velocity model. These methods demand intensive computational cost due to a large amount of data they process and the complexity of their algorithms. Because of that, they are only implemented for parallel systems in practical. Although there are efficient parallel implementations of modeling, RTM, and MFWI in the literature, further improvement can be achieved by better exploring the parallelism in these methods and the characteristics of the current parallel systems. This research proposes coupled multi-scale waveform inversion (CMFWI), an alternative method to MFWI, which improves parallel scalability by reducing the parallel dependency between the processing of different frequency content of the data. An implementation of CMFWI using the coupled local minimizers method (CLM) is presented. L2-norm results showed that CMFWI had an inferior performance when compared to MFWI. These experiments indicate that further research is necessary to implement CMFWI as it compares data with different frequency contents. This work also introduces an auto-tuning strategy for properly choosing the optimal chunk size that reduces the runtime of a 3D RTM algorithm in shared memory systems. A coupled simulated annealing method (CSA) is employed to adjust the chunk size of work that parallel loops assign dynamically to worker threads. Experiments show that the proposed method is consistently better than two default OpenMP loop schedulers being up to 44% faster. This thesis also introduces the cyclic token-based work-stealing (CTWS) for distributed memory systems. The novel cyclic token approach

reduces the number of failed steals, avoids communication overhead, and simplifies the victim selection and the termination strategy. Results obtained by applying the proposed technique to balance the workload of a 3D RTM present a factor of 14.1% speedup and reductions of the load imbalance of 78.4% when compared to the conventional static distribution. Finally, an implementation of a 2D visco-acoustic modeling is presented.

Keywords: Multi-scale Waveform Inversion (MFWI), Coupled Local Minimizers (CLM), Efficiency, Scalability, Auto-tuning, Coupled Simulated Annealing (CSA), Reverse time migration (RTM), Load Balancing, Cyclic Token-Based Work-Stealing (CTWS), One-Sided Communication, Distributed Memory, Shared Memory, Visco-Acoustic Modeling.

Resumo

A modelagem sísmica, a migração reversa no tempo (RTM) e inversão de forma de onda multiescala (MFWI) são três das técnicas mais importantes no levantamento sísmico. A modelagem sísmica simula a propagação de ondas, a RTM gera uma imagem da subsuperfície e a MFWI produz um modelo de velocidades de propagação de ondas. Esses métodos possuem um alto custo computacional devido à grande quantidade de dados que eles processam e à complexidade de seus algoritmos. Por isso, na prática, eles são implementados apenas para sistemas paralelos. Embora existam implementações paralelas eficientes de modelagem, RTM e MFWI na literatura, melhorias podem ser feitas para melhor explorar o paralelismo nesses métodos e as características dos sistemas paralelos atuais. Esta pesquisa propõe a inversão de forma de onda multiescala acoplada (CMFWI), um método alternativo à MFWI, que melhora a escalabilidade paralela ao reduzir a dependência paralela entre o processamento de diferentes conteúdos de frequência dos dados. É apresentada uma implementação do CMFWI usando o método de minimizadores locais acoplados (CLM). Os resultados utilizando a norma L2 mostraram que a CMFWI teve desempenho inferior quando comparado ao MFWI. Esses testes indicam que mais pesquisa é necessária para implementar a CMFWI, pois ela compara dados com diferentes conteúdos de frequência. Este trabalho também apresenta uma estratégia de ajuste automático para escolher adequadamente o tamanho ideal dos blocos de carga de trabalho que reduz o tempo de execução de uma RTM 3D em sistemas de memória compartilhada. O método *Coupled Simulated Annealing* (CSA) é empregado para ajustar o tamanho dos blocos de carga de trabalho que os laços paralelos atribuem dinamicamente às *threads*. Testes mostram que o método proposto é consistentemente melhor do que dois

agendamentos de laços padrão do OpenMP, sendo até 44% mais rápido. Esta tese também introduz o roubo de trabalho cíclico baseado em *token* (CTWS) para sistemas de memória distribuída. A nova abordagem de *token* cíclico reduz o número de falhas de roubo, reduz o *overhead* de comunicação e simplifica a seleção de vítimas e a estratégia de finalização. Os resultados obtidos com a aplicação da técnica proposta para equilibrar a carga de trabalho de uma RTM 3D apresentam um fator de 14,1% de aceleração e redução do desequilíbrio de carga de 78,4% quando comparadas à distribuição estática convencional. Por fim, é apresentada uma implementação de uma modelagem visco-acústica 2D.

Palavras-chave: Inversão da Forma de Onda Multiescala (MFWI), Minimizadores Locais Acoplados (CLM), Eficiência, Escalabilidade, Ajuste Automático, *Coupled Simulated Annealing* (CSA), Migração Reversa no Tempo (RTM), Balanceamento de Carga, Roubo de Trabalho Cíclico Baseado em *Token* (CTWS), Comunicação Unilateral, Memória Distribuída, Memória Compartilhada, Modelagem Visco-Acústica.

Summary

List of Figures

List of Tables

List of Acronyms and Symbols

CIMATEC manufacturing and technology integrated campus

CLM coupled local minimizers

CMFWI coupled multi-scale waveform inversion

CS common-shot

CSA coupled simulated annealing

CTWS cyclic token-based work-stealing

FDM nite difference method

FWI full waveform inversion

HPC high performance computing

MFWI multi-scale waveform inversion

MPI message passing interface

MWMB min-worst-min block

NPAD Núcleo de Processamento de Alto Desempenho

OpenMP open multi-processing

PGAS partitioned global address space

PML perfectly matched layers

RMA remote memory access

RTM reverse time migration

SA simulated annealing

SENAI national service of industrial training

SLS standard linear solid

SNR signal-to-noise ratio

UFRN Universidade Federal do Rio Grande do Norte

WS work-stealing

Chapter 1

Introduction

The seismic survey is one of the leading geophysical techniques. It is used to identify subsurface characteristics. This method generally consists of three stages, namely the acquisition, processing, and interpretation of seismic data. Its ultimate goal is to provide an image of a region of interest in the subsurface.

Seismic data acquisition begins with the explosion of a seismic source, as known as seismic shot, on or near the surface. Part of the seismic signal re ects when passing through regions of different acoustic impedances. Seismic detectors, positioned on or near the surface, record those signals. This process is repeated for different shots and receivers' positions. Typically, a seismic survey comprises from hundreds to thousands of shots.

Some commonly used seismic sources are the detonation of an explosive charge, the dropping of weight, the explosion of a gas chamber, an airgun, or trucks containing a large vibrating mass. The most used seismic detectors are geophones, for onshore surveys, and hydrophones for offshore surveys. Figure 1.1 illustrates the acquisition of seismic data.

The signal from a shot recorded by a detector is called a seismic trace, and the clustering of seismic traces is a seismogram. A common way to cluster seismic traces is the common shot gather where the traces are grouped by the shot which originated them. Figure 1.2 displays traces of a common shot gather for the acquisition of Figure 1.1. These seismic data can be subjected to several processing steps. In general, the purpose of those

Figure 1.1: Example of seismic acquisition where the star and the circles represent the seismic shot and the receivers respectively.

steps is to increase the signal-to-noise ratio (SNR) and improve the vertical resolution of individual seismic traces.

Seismic modeling is an essential step in many seismic data processing methods. It is responsible for simulating wave propagation through the subsurface. Many aspects can be modeled, such as density, acoustic pressure, and particle velocities. The more aspects are modeled, the more accurate is the wave propagation.

Migration is a critical step in the seismic data processing. This procedure is responsible for correctly positioning the re ecting events in the subsurface. Among several migration methods, reverse time migration (RTM) (Baysal et al. 1983, Whitmore 1983) has been highlighted for its robustness in the presence of strong velocity variations. As well as many other geophysical methods, RTM uses information about the propagation velocities of the seismic waves in the region of interest. RTM is detailed at Chapters 3 and 4.

The process of obtaining a velocity model from the observed seismic data is called velocity analysis. In this context, Full Waveform Inversion (FWI) (Tarantola 1984) is one

Figure 1.2: Example of a common shot gather. Source: Kearey et al. (2002)

of the most prominent methods today. FWI iteratively adjusts the velocity model so that the difference between the observed data and the modeled data decreases. As FWI uses a local optimization strategy, it tends to converge to a local minimum in the neighborhood of the initial velocity model.

As an alternative to FWI, the multi-scale waveform inversion (MFWI) (Bunks et al. 1995) initially process the lowest frequencies of the observed seismic data, thus obtaining the largest structures of the model. Smaller structures are gradually inserted in the model as higher frequencies are included. This way, MFWI can scape from local minima. FWI and MFWI are detailed in Chapter 2.

Each cycle of a seismic survey ends with the interpretation phase. In two-dimensional studies, the results are presented to the interpreter as non-migrated and migrated seismic sections. Based on these sections, the interpreter extracts geological information through analyzes of the re ection event pattern.

On the other hand, in interpreting data from a three-dimensional survey, the interpreter has direct access, on a workstation, to all re ection data contained in a seismic data volume. The interpreter can select various types of data for a color presentation, for example, vertical or horizontal sections of a data volume.

Most of the geophysical methods are computationally intensive because of either the signi cant amount of data or the complexity of the algorithms involved. For this reason, for industry-scale data, they can only be performed in a reasonable time if taking advantage of parallel systems. Although there are ef cient parallel implementations of modeling, RTM, and FWI in the literature, further improvement can be achieved by better exploring the parallelism in these methods and the characteristics of the current parallel systems.

1.1 Objectives

The following topics summarize the objectives of this thesis:

1. To introduce a method, called cyclic token-based work-stealing (CTWS), to balance the load of the RTM in distributed memory systems. By employing a work-stealing strategy, CTWS aims to ensure that no processing unit will become idle before all tasks in the system are being processed. Moreover, its purpose is to reduce synchronization among processes and overlap communication and computation.
2. To propose an auto-tuning technique to balance the load of the RTM in shared memory systems. By using an optimization method, the coupled simulated annealing (CSA), this technique aims to automatically determine the size of blocks of iterations of parallel loops to be dynamically distributed among the threads. This strategy intends to reduce RTM runtime by increasing cache reuse.
3. To design a strategy, the coupled multi-scale waveform inversion (CMFWI), to reduce dependency on processing multiple frequency scales of the MFWI. This way, all frequency scales can be processed simultaneously. Besides that, the inversion of the frequency scales can exchange information in order to converge to a global optimum. By doing so, CMFWI aims to increase MFWI parallel scalability as well as providing more accurate resulting velocity models.

1.2 Thesis outline

This doctoral research is divided into three main parts; each of them proposes a method to improve the parallel ef ciency of a seismic application. Chapter 2 proposes, CMFWI, an alternative to MFWI's formulation in order to make it more scalable. Chapters 3 and 4, introduce techniques to reduce the imbalance of the workload distribution of the RTM. Chapter 3 describes an auto-tuning method based on CSA to improve the

cache reuse of RTM in shared memory systems. This chapter has been under review for re-submission for publication. Chapter 4 introduces CTWS, a work-stealing method that aims to reduce processing units idle time in distributed memory systems. Also, in this chapter, CTWS is applied to a 3D RTM. Chapter 4 has been published at (Assis et al. 2019). Appendix A reports the implementation of a visco-acoustic modeling method and is not related to the rest of this thesis. Chapter 5 summarizes this doctoral research. Chapters 2, 3, 4 and A are self contained.

Chapter 2

Coupled multi-scale waveform inversion

Full waveform inversion (FWI) is one of the most known methods for seismic velocity analysis. This method uses an initial velocity model generated by another velocity analysis technique. At each FWI iteration, an initial velocity model is used to simulate the acquisition of seismic data computationally. A local optimization method is then used to obtain a new model that minimizes the difference between observed and calculated seismic data.

By using local optimization methods, FWI tends to converge to a local optimum model in the vicinity of the initial model. Using a global optimization method in FWI is not feasible since a large number of objective function evaluations performed by these methods would imply a high computational cost. Because of that, it is common to use techniques that use local optimizers but increase the likelihood of convergence to the global optimum.

The most commonly used technique for this purpose is the multi-scale waveform inversion (MFWI) (Bunks et al. 1995). Its strategy is to initially use the lowest frequencies of the observed data, thus obtaining the largest structures of the model. Higher frequencies are added gradually to insert smaller structures into the model.

Despite achieving robust results, MFWI has an even higher computational cost than FWI, especially for three-dimensional surveys. The computational effort demanded grows, the higher the number of scales. Besides, the scales cannot be processed simultaneously since the model resulting from each scale inversion is used as the initial model for the

next scale inversion. This restriction limits the scalability of this method in parallel computational systems.

According to Li et al. (2015), in addition to this parallel ef ciency issue, it is common that low-frequency information is not available or unreliable in real seismic data. This fact hinders the convergence of traditional MFWI to the global minimum. Li et al. (2015) proposed a method with two levels of parallelism for MFWI in the frequency domain.

This chapter introduces the coupled multi-scale waveform inversion (CMFWI). Its strategy is to perform an MFWI in which the initial models used for the inversion of the frequency scales are independent of each other. Also, the inversions of the frequency scales are coupled to each other, i.e., the inversion of a scale uses information about the other scales.

Like MFWI, CMFWI's strategy is to increase the chance of convergence to the global optimum. However, CMFWI reduces the dependency between the inversions of the frequency scales. In this way, it becomes possible to perform the inversion of the scales simultaneously, making better use of the commonly used parallel architectures.

CMFWI's proposal is suitable for the use of restricted minimization methods, such as penalty methods. However, in this work, it is proposed to use coupled local minimizers (CLM) (Suykens et al. 2001) as the CMFWI optimization method.

The rest of this chapter is organized as follows. Section 2.1 and 2.2 describe FWI and MFWI respectively. Section 2.3 introduces CMFWI. A formulation of CMFWI based on CLM is shown at Section 2.4. Section 2.5 presents results of CMFWI in comparison to FWI and MFWI. Ultimately, Section 2.6 concludes this chapter.

2.1 Full waveform inversion

Seismic velocity analysis estimates the seismic wave propagation velocities from the observed data, i.e., the resulting data from the acquisition step. Figure 2.1 presents an

Figure 2.1: 2D slice of a 3D velocity model.

example of a 2D slice of a 3D velocity model. The velocity model is used in another stage of seismic processing, called migration, in which recorded reflection events are positioned in their correct locations. The relationship between the seismic velocities and the observed seismic data can be expressed by

$$G(m) = d; \tag{2.1}$$

where G is the seismic modeling operator.

The seismic modeling operator simulates the propagation of the seismic shots using the velocity model and records the seismic waves in the same positions where the observed data were recorded. Thus, to obtain the velocity model from the observed data, we can rewrite Equation 2.1 as (dos Santos 2013)

$$m = G^{-1}(d); \tag{2.2}$$

where G^{-1} is the inverse seismic modeling operator.

The exact calculation of G and, especially, G^{-1} are, in many cases, not feasible due to the complexity of these operators. Therefore, approximations are usually used.

The velocity analysis is an inverse problem since it seeks to infer the velocity model m from the observed data d. Different formulations for the operators G and G^{-1} give rise to several methods of velocity analysis. These include transit time tomography, stereotomography, residual migration, and residual curvature tomography. In this context, one of the most used methods is FWI. It has been notable for obtaining models with higher resolution than those achieved by methods traditionally used in the analysis of seismic velocities.

Given an initial velocity model, m_0, FWI uses an approximation of the seismic modelling operator, G, to simulate the acquisition of seismic data. FWI's strategy is to iteratively minimize the difference between the calculated seismic data, $G(m)$, and the observed seismic data, d. Mathematically, FWI can be described as a least squares optimization problem whose objective function is (dos Santos 2013)

$$\min_{m \in R^n} J(m) = \frac{1}{2} \| d - G(m) \|^2; \tag{2.3}$$

where $\| \cdot \|$ is the Euclidean norm.

Let m_{k+1} be the model resulting from the minimization of $J(m)$ in the kth iteration of FWI. m_{k+1} is defined as the sum of a perturbation Δm_k to an initial model m_k. In general terms, the methods used to perform the minimization of the FWI's objective function update the model according to the formula (dos Santos 2013)

$$m_{k+1} = m_k + \Delta m_k = m_k + \alpha_k p_k; \tag{2.4}$$

where p_k is a vector of the search direction and α_k is the step length to be taken in the direction p_k.

Using Newton's method, the iterative scheme of FWI is defined as (dos Santos 2013)

$$m_{k+1} = m_k + \Delta m_k = m_k - H_k^{-1} g_k; \tag{2.5}$$

where g_k and H_k^{-1} are the gradient and the inverse of the Hessian of $J(m)$ applied to the model m_k, respectively. In this case $\alpha_k = 1$ e $p_k = -H_k^{-1} g_k$.

However, calculating the Hessian of $J(m)$, in general, has a high computational cost (Virieux & Operto 2009). Thus, it is common to use less costly methods such as gradient-based and quasi-Newton methods. In these minimization methods, it is common for the search direction to be given by (Nocedal & Wright 2006)

$$p_k = -B_k^{-1} g_k; \tag{2.6}$$

where B_k is an approximation of the Hessian. In the steepest-descent method, for example, B_k^{-1} is the identity matrix.

FWI terminates when, for example, it reaches a predetermined number of iterations or through a threshold (e.g., $|m_{k+1} - m_k| < \epsilon$).

The algorithm of FWI can be defined as:

1. $k = 0$;
2. Compute the residual data $\delta d = d - G(m_k)$;
3. Calculate the search direction p_k;
4. Calculate the step length α_k;
5. Set a new model m_{k+1} following Equation 2.4.

   If a stop condition is satisfied, m_{k+1} is the resulting model;

   Otherwise, set $k = k + 1$ and return to step 2;

The length of the step α_k can be a predefined constant or determined by line search. The objective function of this search can be given by

$$f(\alpha) = J(m_k + \alpha p_k). \tag{2.7}$$

The direction p_k, in general, is obtained from the gradient g_k. One of the ways to

calculate this gradient is through the adjoint-state method (Plessix 2006) which de nes that

$$g(m) = \frac{\P J(m)}{\P m} = \frac{2}{v(x)^3} \int_0^T \lambda \frac{\P^2 u(x;t)}{\P t^2} dt; \tag{2.8}$$

where t is the time dimension, v is the propagation velocity of the wave which is represented by the model m, T is the duration of the seismic survey and u is the wave eld of the source λ is the wave eld resulting from the propagation of the residual data $r(m)$ in a reverse order in time and is obtained through the relation

$$\frac{1}{v(x)^2} \frac{\P^2 \lambda(x;t)}{\P t^2} = \tilde{N}^2 \lambda(x;t) + r(m): \tag{2.9}$$

Equation 2.8 de nes a cross-correlation between the wave eld of the source and the wave eld of the backpropagation of the residual data.

2.2 Multi-scale waveform inversion

The goal of FWI is to iteratively optimize the model that represents the characteristics of the subsurface. This iterative update of the model is usually done through local optimization methods such as gradient and quasi-newton methods. As a consequence, FWI tends to converge to a local optimum model.

An alternative is using meta-heuristics for global optimization such as genetic algorithm and simulated annealing. However, these methods perform several evaluations of the objective function, which could make the FWI implementation unfeasible.

A more appropriate option is to use strategies that, with an acceptable computational cost, increase the likelihood of convergence to the global minimum. One such strategy is the multi-scale waveform inversion (Bunks et al. 1995), here denoted by MFWI. While FWI processes all the frequency content of the observed data, MFWI processes rst the lowest frequencies of the observed data and adds the higher frequencies gradually. Figure

2.2 presents the heuristic behind MFWI.

MFWI's strategy also aims to overcome problems of cycle-skipping, a possible source of convergence for local minima. It occurs that, due to errors in the initial velocity model, a signal of the calculated data may be displaced in time by more than half of its period. In this way, this signal will be in a different cycle from its corresponding cycle in the observed data. Using only low frequencies makes the signal period longer. Therefore, it would reduce the number of errors in the model capable of generating cycle-skipping.

The inversion of the observed data from the initial model using low frequencies results in a new model. This model is used as input for the inversion of the observed data using a broader frequency band, as exempli ed by Figure 2.3.

In the example of Figure 2.3, a data having a frequency band of 0 to 40 Hz can be inverted through an MFWI using the frequency band of 0 to 10 Hz initially. The model resulting from this rst inversion can then be used as the initial model to invert the observed data using the 0 to 20 Hz frequency band. This procedure is repeated by progressively increasing the frequency band used up to the full band, i.e., from 0 Hz to 40 Hz.

2.3 Coupled multi-scale waveform inversion

This chapter proposes the coupled multi-scale waveform inversion (CMFWI). Like MFWI, in this approach, frequency scales are chosen so that the rst scale contains only the lowest frequencies, and the last scale contains the entire frequency band. However, the inversion of each scale does not use the nal velocity model obtained by the inversion of the previous scale. Instead, the inversion of each of the scales uses a possibly different initial model, i.e., the qth scale takes an initial model m_0^q independent of the initial models used in the other scales.

Although the initial model used in the inversion of each scale is independent of the other models, updating the model at each step of the inversion of a scale uses informa-

(a) Lower frequency content of the objective function.

(b) Intermediary frequency content of the objective function.

(c) All frequency content of the objective function.

Figure 2.2: Multi-scale heuristic. Starting from the low frequency content may prevent the local optimization method from converging to a local minima. The optimizations using lower frequencies content provide accurate initial solutions to the optimizations using higher frequencies content.

Figure 2.3: Multi-scale FWI scheme. K is the number of FWI iterations at each frequency scale. Q is the number of frequency scales.

Figure 2.4: CMFWI schema. K is the number of FWI iterations at each frequency scale. Q is the number of frequency scales.

tion from the inversion of the other scales. The purpose of this coupling is to make the inversion of a scale escape local minima towards the global minimum. By using different initial models, CMFWI can explore a broader region of the domain. Figure 2.4 shows the CMFWI scheme.

Let Q be the number of frequency scales, and m^q be the initial model to be used for the inversion of the qth scale ($0 \quad j < Q$). The seismic data d^q used in the inversion of the qth scale is the result of applying a bandpass filter to the observed data d. The purpose of this filter is to remove frequencies that are not contained in the qth scale. Given that, the objective function for the FWI of each scale is given by

$$J(d^q; m^q) = \frac{1}{2} \| d^q \quad G(m^q) \|^2; \quad q = 1; 2; :::; Q: \tag{2.10}$$

Where G is the modeling operator. If this operator simulates a seismic source, the signal representing that source pulse must also be filtered in order to remove frequencies that do not belong to the qth scale.

Besides the set of objective functions defined at Equation 2.10, CMFWI systems must have coupling constraints, which provokes that all the local optimizers to exchange search direction information in order to converge to the same final model. The way these

constraints are de ned depends on which minimization method is applied. Section 2.4 presents an example where CLM is used.

From the parallelism perspective, the algorithms used for seismic data processing in the time domain usually have up to two levels of parallelism. The rst level concerns the parallel processing of seismic data from different shots. The other is domain decomposition.

CMFWI, in addition to these two levels of parallelism, parallelize the inversion of the scales. The independence between the initial models used in CMFWI makes it possible to calculate the gradient for each scale simultaneously. The addition of this new level of parallelism makes the algorithm more scalable as more work can be distributed simultaneously among the various processing units available.

From the perspective of the quality of the obtained model, CMFWI may obtain better results than MFWI. It is common for real seismic data to contain little low-frequency information. Thus, the inversion of the initial scales may provide a model far from the global minimum as the initial estimate for the upper scales. In these cases, CMFWI may be able to escape from local minima as the scales that obtain better-evaluated models in the objective function may in uence the search direction of the other scales.

2.4 Coupled multi-scale waveform inversion via coupled local minimizers

CLM (Coupled Local Minimizers) is a technique that proposes improvements to local multi-start optimization so that the search points converge to the global minimum. For this, during CLM iterations, each local optimizer uses information from the other local optimizers.

Let $f(x)$ be a function, with $x \in R^n$, from which you want to nd the global minimum. CLM uses a population of Q points, $X = \{x^1; :::; x^Q\}$, and seeks to optimize the function

average of these points de ned by

$$\langle f \rangle = \frac{1}{Q} \sum_{q=1}^{Q} f(x^q). \tag{2.11}$$

In the context of CMFWI, each optimizer aims to minimize Equation 2.10 for a frequency scale q. Applying CLM to CMFWI's set of optimizers leads to the following objective function:

$$\langle J \rangle = \frac{1}{Q} \sum_{q=1}^{Q} J(d^q; m^q). \tag{2.12}$$

In order for the population to converge to a single point, CLM adds equal parity restrictions between population points. In this way, the minimization problem can be reformulated as

$$\begin{aligned} \min_{m^q \in R^n} \langle J \rangle \ \text{s.t.}\ & m^q - m^{q+1} = 0 \\ & q = 1, 2, \ldots, Q \\ & m^{Q+1} = m^1. \end{aligned} \tag{2.13}$$

The problem represented by the equation 2.13 is then solved through the augmented lagrangian method (Nocedal & Wright 2006) whose objective function is

$$L(M, L) = \frac{h}{Q} \sum_{q=1}^{Q} J(d^q; m^q) + \sum_{q=1}^{Q} [l^q]^T [m^q - m^{q+1}] + \frac{g}{2} \sum_{q=1}^{Q} \| m^q - m^{q+1} \|^2, \tag{2.14}$$

where $M = [m^1, \ldots, m^Q]$, $L = [l^1, \ldots, l^Q]$; $l^q \in R^n$; are the Lagrange multipliers, h is a weighting factor of the average function, g is the penalty parameter and $\|\cdot\|$ is the Euclidean norm.

Local optimization such as gradient, newton or quasi-newton methods can be used to minimize the Equation 2.14. The gradient and hessian of $L(M, L)$ are determined by

(Teughels et al. 2004)

$$\tilde{N}_{m^q} L(M;L) = \frac{h}{t}\tilde{N}_{m^q} J(d^q;m^q) \quad l^{q-1} + l^q \quad g\, m^{q-1} \quad m^q + g\, m^q \quad m^{q+1}; \tag{2.15}$$

$$\tilde{N}^2_{m^q} L(M;L) = \frac{h}{t}\tilde{N}^2_{m^q} J(d^q;m^q) + 2g\, I; \tag{2.16}$$

$$\tilde{N}^2_{m^q m^{q-1}} L(M;L) = \quad g\, I; \tag{2.17}$$

$$\tilde{N}^2_{m^q m^{q+1}} L(M;L) = \quad g\, I: \tag{2.18}$$

where I is the identity matrix. The boundary conditions are $m^0 = m^Q$ and $m^{Q+1} = m^1$. From Equations 2.15 to 2.18, it is inferred that gradient information may be exchange by neighbors optimizers.

Lagrange multipliers are updated at each iteration k according to Equation 2.19.

$$l^q_{k+1} = l^q_k + g(m^q_{k+1} \quad m^{q+1}_{k+1}): \tag{2.19}$$

Teughels et al. (2004) present a discussion on the influence of the values of the h and g parameters in the search process performed by CLM.

Ideally, at the end of the inversion, $L(m;L) = 0$ and $m^1 = m^1 = \quad = m^q$. However, the Q resulting models may be different since each of them is evaluated by a different $J\ d^j;m^j$ function. CMFWI can also achieve this condition since the automatic adjustment of the Lagrange multipliers may eventually relax the equality constraints. Parameters h and g can also be chosen in order to balance the priority of the average objective function and constraints.

Although the CLM has been chosen in this work, the CMFWI problem, described in Equation 2.13, can be solved by another minimization method for constrained functions.

(a) True model (b) Initial model

Figure 2.5: (a) True geological model used to generate the observed data. (b) Initial model for FWI, MFWI and CMFWI.

2.5 Results and discussion

The experiments were performed at OPTIMUM HPC Cluster at the University of British Columbia. OPTIMUM is composed of 56 compute nodes, 2 CPUs Intel's Ivy Bridge 2.8 GHz E5-2680v2 per node, and ten cores per CPU. Fifty-two nodes are equipped with 128 GB RAM and four nodes with 256 GB RAM. It also has a 182 TB Lustre distributed parallel file-system.

For the following experiments, the spatial dimensions were $(N_{x_1}, N_{x_2}) = (125, 200)$, the spatial resolution was 10 m and the source used in the modelling operator was a Ricker wavelet with peak frequency of 10 Hz. The geological model used to generate the observed is presented in Figure 2.5a. The initial model for FWI, MFWI, and each scale of CMFWI is shown in 2.5b. The frequency bandwidth for the three methods was 3–15 Hz. For both multi-scale methods, the frequency batches were $\{4, 5, \ldots, 9, 10\}$ Hz. Finally, the optimization method used to update the model in the three methods was the L-BFGS (Nocedal & Wright 2006).

A prototype of the proposed method was implemented in 2D using the framework introduced by Da Silva & Herrmann (2016). Perfectly matched layers (PML) (Chen et al.

Figure 2.6: L2-norm for the resulting model of CMFWI, MFWI and FWI compared to the true model. Different sets of CLM parameters (h and g) were tested on CMFWI. FWI performed 20 iterations. MFWI and CMFWI performed 20 FWI iterations per scale.

2013) implemented by Da Silva & Herrmann (2016) with a thickness of 60 points were used to absorb the energy reaching the limits of the models during seismic modeling.

Figure 2.6 presents a comparison of the results obtained via FWI, MFWI, and CMFWI. Different values for the CMFWI's parameters h and g were tested. The resulting models for each method are compared to the true geological model (see Figure 2.5a) through L2-norm.

The measurements displayed in Figure 2.6 show that CMFWI had roughly the same results of FWI. When compared to MFWI, CMFWI presented considerably higher L2-norm. These results indicate that CMFWI using a single initial model to all frequency scales is dominated by the inversion of the larger frequency scale. As all initial models are the same, the equality constraints may prevent CMFWI's models from exploring different paths.

Figure 2.7 shows graphical results of an inversion of a model with dimensions $N_x =$

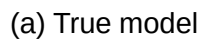
$(N_{x_1}; N_{x_2}) = (261; 351)$ obtained via FWI, MFWI and CMWFI. Figure 2.7 also presents the true geological model and the initial model used in this experiment. As it shows, the resolution of the CMFWI output model is lower than FWI and MFWI results in this experiment. This characteristic is especially clear at the bottom of the model.

2.6 Conclusions

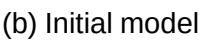
This chapter proposed the CMFWI, an alternative method to MFWI. By using possibly different initial models for the inversion of each frequency scale, it allows different frequency scales to be processed in parallel. Nevertheless, constraints must be added to ensure that the inversions of different frequency scales can exchange information.

A formulation of CMFWI using CLM was presented and tested against FWI and MFWI. For the experiments shown in this chapter, CMFWI used the same initial model for all frequency scales inversions. L2-norm results showed that, under these circumstances, CMFWI had an inferior performance when compared to FWI and, especially, to MFWI.

These results are inconclusive as the tests were performed using the same initial model for all frequency scale inversions. This fact may have prevented the optimization system by exploring different results. Further research should investigate the generation of proper initial models for each frequency inversion as well as a method to compared models with different frequency content.



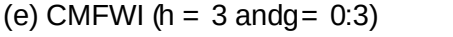
(a) True model



(b) Initial model

(c) FWI

(d) MFWI



(e) CMFWI ($\eta = 3$ and $\varepsilon = 0.3$)

Figure 2.7: (a) True model with $N = (N_{x_1}, N_{x_2}) = (261, 351)$. (b) Initial model with $N = (N_{x_1}, N_{x_2}) = (261, 351)$. (c) Resulting model of FWI after 20 iterations. (d) Resulting model of MFWI after 20 iterations per scale. (e) Resulting model of CMFWI after 20 iterations per scale. The frequency batches used for MFWI and CMFWI were $\{3.4, 3.5, \ldots, 3.10\}$ Hz

Chapter 3

Auto-tuning for 3D reverse time migration in shared memory systems

Seismic re ection surveying is the best known and used geophysical method for subsurface imaging. Oil and gas exploration is likely its primary application. Its main objective is to generate an image of a region of the subsurface to identify structures of interest.

Seismic data can go through several processing steps in order to improve the signal-to-noise ratio (SNR) and the resolution of the seismic image. One of the most important of these steps is migration, which is responsible for positioning seismic re ection events in their correct place when imaging the subsurface. In this context, reverse time migration (RTM) (Baysal et al. 1983, Kosloff & Baysal 1983) has been widely used as a migration technique to more accurately take into account the wave propagation effects resulting in subsurface images with higher de nition.

Simulating wave propagation comprises the majority of an RTM and is computationally intensive, especially for the three-dimensional case. Therefore, the computational cost is the main factor that limits the application of RTM, as well as for several other geophysical algorithms (Zhang et al. 2009, Araya-Polo et al. 2009). For this reason, parallel computing techniques have been widely applied to these methods (e.g., Nunes-do Rosario et al. (2015)).

Load balancing is one of the main aspects to be considered in parallel applications.

It can be de ned as the distribution of the computational load among the available processing resources (e.g., cores, computing nodes). A way to perform load balancing is by dividing the workload in chunks of computation to be distributed among the computational resources either statically or dynamically.

Load balancing parallel seismic methods, such as RTM, is especially challenging with the rise of heterogeneous machines. Nevertheless, as we will see, even for homogeneous architectures, a static load balancing may not be optimal.

Auto-tuning techniques have been used as an approach to nd near-optimal load balancing. Tchiboukdjian et al. (2011) introduced a scheduler for applications with linear access to shared memory. They aim to improve locality by guaranteeing that all data in the cache are used before being replaced.

Furthermore, load balancing can also be used along with processors frequency control tools to improve the energy ef ciency of imbalanced parallel applications in multi-core systems, as shown by Padoin et al. (2014) and Padoin et al. (2017). Both these works use processors frequency scaling techniques to slow down less loaded cores in order to save energy.

In the context of seismic applications, some authors have developed auto-tuning techniques for the nite difference method (FDM), which is often used in geophysical applications. Katagiri et al. (2014) and Katagiri et al. (2015) introduced ppOpen-AT, a framework for code optimization guided by directives. Barros et al. (2018) provided experiments to show that the optimal load balancing for shared memory environments depends on the hardware and software employed. They also point to the coupled simulated annealing (CSA) (Xavier-de Souza et al. 2010) algorithm as a promising method to obtain a near-optimal load balance for a 3D FDM.

RTM has also been the target of auto-tuning techniques. Sena et al. (2011) presented a method to determine a near-optimal block size by testing a set of possible values in a few time steps of RTM and choosing the one with the shortest execution time. Andreolli

et al. (2014) and Andreolli et al. (2015) introduce an approach to tune seismic applications automatically by compiling and running each set of parameters chosen by a genetic algorithm, including chunk size and compilation flags. According to Andreolli et al. (2015), after all code optimization is done, auto-tuning will become a necessary step to extract the best performance from computing systems.

We present an execution time CSA-based auto-tuning strategy for correctly choosing the optimal chunk size that reduces the execution time of a 3D RTM algorithm. Our strategy aims to reduce the execution time of a 3D RTM by automatically finding an ideally optimal chunk size for OpenMP (Dagum & Menon 1998) parallel loops.

Following, we present the basics of our target application: the RTM (Section 3.1), the parallelization strategies made available by OpenMP (Section 3.2) and the optimization method that comprises the proposed auto-tuning, the CSA (Section 3.3). Then, we provide a detailed description of our RTM implementation (Section 3.4) as well as of the proposed auto-tuning approach (Section 3.5). Section 3.6 displays the results of the proposed method in comparison with the standard OpenMP schedules. Section 3.7 concludes this paper.

3.1 Reverse time migration formulation

The seismic reflection method consists of three main steps: acquisition, processing, and interpretation of seismic data. In the acquisition, seismic shots, reflected by subsurface interfaces, are recorded at surface level by receivers. The signal recorded by each detector, from each seismic shot, is called a seismic trace. A set of seismic traces is called a seismogram. Seismograms can be converted to depth estimates of interfaces between different subsurface materials during processing.

After the acquisition step, several techniques can be used to process seismic data. In general, the purpose of processing reflection data is to increase the SNR and improve the

vertical resolution of resulting seismic images. Migration is one of the main steps in seismic data processing. It aims to 1) properly position seismic re ections at the coordinates of the re ector in the subsurface; 2) reduce diffraction effects in the images; 3) improve the spatial resolution.

Modern migration approaches use the seismic wave equation, a partial differential equation describing wave motion, generated by a source in a medium. The scalar equation for 3D acoustic waves is de ned as

$$\frac{\partial^2 u(x)}{\partial x_1^2} + \frac{\partial^2 u(x)}{\partial x_2^2} + \frac{\partial^2 u(x)}{\partial x_3^2} = \frac{1}{c(x)^2}\frac{\partial^2 u(x)}{\partial t^2} + s(t); \tag{3.1}$$

where $x = (x_1; x_2; x_3)$ are the spatial dimensions, $u(x)$ is the acoustic pressure, $c(x)$ is the propagation velocity and $s(t)$ is the source function at time t.

Spatial and time restrictions should be observed when solving nite differences by a numerical approach (Carcione et al. 2002). These restrictions are de ned as:

$$\max(\Delta x_1; \Delta x_2; \Delta x_3) \quad \frac{c_{min}}{W f_{max}} \tag{3.2}$$

and

$$\Delta t \quad \frac{2\min(\Delta x_1; \Delta x_2; \Delta x_3)}{p c_{max}\sqrt{3}}; \tag{3.3}$$

where Δx_1, Δx_2 and Δx_3 are the spatial sampling of dimensions x_1, x_2 and x_3, Δt is the time sampling; f_{max} is the maximum frequency of $s(t)$; c_{min} and c_{max} are the minimum and the maximum values of $c(x)$; and W is the number of grid points per minimum wavelength. According to Carcione et al. (2002), W must be equal or greater than 4 for high order nite differences schemes. Non-compliance with (3.2) and (3.3) would result in numerical dispersion and instability.

Another important aspect is that the geological model encoded in $c(x)$ must be restricted to a nite number of points on a mesh, even though the Earth is heterogeneous

and continuous. In order to represent real boundaries, it is common to apply artificial edges to the model limits, to absorb the energy reaching the borders (Cerjan et al. 1985).

There are several approaches to migrate seismic data. We use migration by finite differences (or wave equation migration), in which the wave equation is approximated by a finite-difference equation, suitable to be solved by a computer as explained above. One of the main migration methods by finite differences is RTM (Baysal et al. 1983, Kosloff & Baysal 1983). In RTM, source and receiver wavefields are propagated forward and backward in time, respectively. RTM imaging relies on the physical property that those pressure waves must correlate at the reflective interfaces.

The core of an RTM can be divided into three stages. The first stage is the simulation of the propagation of a wavefield resulting from the excitation of a seismic source. The second stage is the backpropagation of wavefields registered in a seismogram. Finally, the third stage is the imaging condition, which is a correlation between the forward and backward propagated wavefields and produces an image of the subsurface. This process is repeated for all the shots of seismic data available.

The propagation and backpropagation steps use the same velocity model shown in (3.1) as $c(x)$. This model specifies the wave velocity for each mesh point and represents the different properties of the materials and boundaries, in the volume being imaged.

In the imaging condition stage, the wavefields generated by the propagation of the source and the backpropagation of the observed data are correlated pointwise, at each time interval, to generate an image. Mathematically, it is defined as (Claerbout & Doherty 1972)

$$I(x) = \sum_{t=0}^{T} u_i(x;t)\ u_r(x;t)\,dt; \tag{3.4}$$

where $I(x)$ is the resulting image, $u_i(x;t)$ is the wavefield propagated with the source excitation, $u_r(x;t)$ is the wavefield of backpropagated data and T is the total simulation time. The migration of each seismic shot generates an image. These images are stacked

to build the total migrated volume.

Each cycle of a seismic survey ends with the interpretation phase. Since both coverage and resolution are better with 3D data, these surveys lead to improvement of interpretation, compared with 2D surveys, and are standard today (Yilmaz 2001).

3.2 Parallelization strategy

In this work, the RTM algorithm was implemented with two degrees of parallelization. The first is the migration of different common-shot (CS) gathers, i.e., seismic data with the same shot coordinates, which is implemented with the message passing interface (MPI) (Clarke et al. 1994), for distributed memory environments. The second is the migration of a single CS gather and is performed in shared memory environments, with OpenMP (Dagum & Menon 1998).

Our work is applied in the second degree of parallelization, where different loops of the RTM operation of each CS gather are parallelized among the cores of a multi-core system, with OpenMP. This parallelization is performed by dividing each loop into loops of smaller sizes, which are computed in the different cores of the multi-core system. The size of these smaller loops is usually referred to as the *chunk size*. The main goal of our work is to balance the computation of the smaller loops by the different cores, by choosing the proper chunk size, which is known as workload balancing. The proposed load balancing approach is discussed in more detail in Section 3.3.

For the parallelization with OpenMP, the *parallel for* construction was employed. This construction automatically distributes the workload (N_{loop}) among all threads ($N_{threads}$), in the loop where it is applied. The workload distribution within the threads can be changed by using the OpenMP clause *schedule* and variable *chunk size*. The different OpenMP workload distributions used in this work are explained next.

Static: is the standard distribution in OpenMP, where the load is distributed for each

thread in fixed data blocks of roughly $N_{loop}/N_{threads}$. It is possible to choose the size of these blocks by changing the *chunk size* variable.

Dynamic: similar to the *static* one, with the main difference that, when a thread finishes to compute the work allocated to it and becomes idle, the system automatically allocates more work to this thread, until all the work finishes.

Auto: is the automatic distribution provided by OpenMP. It delegates all the scheduling decisions to the compiler.

3.3 Coupled simulated annealing

Coupled Simulated Annealing (CSA) (Xavier-de Souza et al. 2010) is a global optimization algorithm, based on the well-known simulated annealing (SA) algorithm (Kirkpatrick et al. 1983). The SA algorithm, also a global optimization method, is inspired by the thermodynamic annealing process, which consists of a heat treatment that alters the physical or chemical properties of a given material. The SA algorithm is employed in minimization (or maximization) problems, where the goal is to obtain the minimum (or maximum) of a specific cost function, namely the *energy* of the annealing process. Our work is focused solely on minimization problems.

Briefly, the SA algorithm is divided into the generation of new solutions and the acceptance of these solutions. Algorithm parameters, known as generation and acceptance temperatures, control both these stages. The proper tuning of these temperatures is a considerable challenge when using SA-based algorithms. New possible solutions, also known as probe solutions, are generated by a function of the generation temperature. If a probe solution yields a smaller value of the cost function, this solution is accepted as the new one with probability one; otherwise, this solution is only accepted as the new one with the probability given by a function of the acceptance temperature.

In its turn, the CSA algorithm consists of a set of parallel SA algorithms, known as

SA optimizers. Each SA optimizer generates and evaluates a probing solution, updating its current state. The generation and acceptance temperatures are equal for all the different SA instances. The main differences between CSA and SA are that 1) for accepting solutions with higher cost function values, the CSA considers all current solutions; and 2) the acceptance criterion is based on the current solutions and also on a coupling term between these solutions. The coupling approach has shown to be capable of reducing the sensitivity of the algorithm to initialization parameters and providing information that might steer the overall optimization process toward the global optimum.

The CSA algorithm employed in this work is the one described in (Gonçalves-e Silva et al. 2018, Gonçalves-e Silva & Xavier-de Souza 2018), which is implemented as follows. Let $a_i \in Q$ and $b_i \in W$ be, respectively, the current and probe solutions of the i-th SA optimizer; with Q and W being the set of current and probe solutions, respectively, and $i = 1, \ldots, m$, where m is the number of elements in both Q and W. At the k-th iteration of the CSA algorithm, the probe solutions are given by (Xavier-de Souza et al. 2010)

$$b_i = a_i + e_i T_k^{gen}, \tag{3.5}$$

where T_k^{gen} is the generation temperature and e_i is a random variable sampled from the Cauchy distribution (Szu & Hartley 1987)

$$g(e, T) = \frac{T}{(e^2 + T^2)^{(D+1)/2}}, \tag{3.6}$$

where $T = T_k^{gen}$ and D is the dimension of the problem. The rule for updating T_k^{gen} is also a free choice of the specific CSA implementation. We followed the guidelines from (Gonçalves-e Silva et al. 2018) and used as update $T_{k+1}^{gen} = 0.99999\, T_k^{gen}$, with T_k^{gen} being updated to 99.999% of its previous value.

Each solution a_i and b_i, has an associated energy (or cost) value $E(a_i)$ and $E(b_i)$.

The acceptance probability function is de ned as:

$$A_Q = \frac{\exp^{\frac{E(a_i) \quad \max(E(a_i))_{a_i 2 Q}}{T_k^{ac}}}}{g}; \tag{3.7}$$

where T_k^{ac} is the acceptance temperature and g is the coupling term, given by:

$$g = \sum_{8a2Q} \exp^{\frac{E(a) \quad \max(E(a_i))_{a_i 2 Q}}{T_k^{ac}}} : \tag{3.8}$$

If $E(b_i) > E(a_i)$, a_i assumes the value of b_i only if $A_Q < r$, where r is a random variable sampled from an uniform distribution in the interval $[0; 1]$. Otherwise, if $E(b_i) < E(a_i)$, a_i assumes the value of b_i with probability one.

As shown in Xavier-de Souza et al. (2010), the CSA performance is improved if the variance of A_Q is kept close to its maximum value. This variance might be written as

$$s^2 = \frac{1}{m} \sum_{8a2Q} A_Q^2 \quad \frac{1}{m^2} \tag{3.9}$$

and lays in the interval

$$0 \quad s^2 \quad \frac{m \quad 1}{m^2}: \tag{3.10}$$

The controlling of this variance value can be accomplished by using the following rule to update the acceptance temperature:

$$T_{k+1}^{ac} = \begin{cases} T_k^{ac}(1 \quad a); & \text{if } s^2 < s_D^2 \\ T_k^{ac}(1+ a); & \text{if } s^2 \quad s_D^2 \end{cases}; \tag{3.11}$$

where s_D^2 is the desired variance, which should be kept as close as possible to $\frac{m \quad 1}{m^2}$, and a is the acceptance temperature modi cation rate, usually a value within the interval $(0; 0{:}1]$.

The CSA algorithm is parameterized by setting the initial temperature values T_0^{gen}

and T_0^{ac}. In this work, we determined these initial temperatures mostly by trial and error. However, we noticed in our experiments that the CSA algorithm is very robust to the initial values of the acceptance temperature, T_0^{ac}. In this work, the adopted stopping criterion for the CSA implementation was the total number of algorithm iterations, which was left to be set by the user and is represented by the variable NT. There are several comparisons between CSA based optimization methods and other algorithms (Xavier-de Souza et al. 2010, Gonçalves-e Silva & Xavier-de Souza 2018), where one may see the main advantages in using CSA, mostly related to its robust initialization and functional capacity of finding values close to the global optimum. Due to these characteristics, the CSA seemed the right choice for the proposed auto-tuning algorithm.

In the proposed auto-tuning approach, the CSA is employed to minimize the execution time of different loops in the RTM algorithm, parallelized with OpenMP, by properly choosing the optimal workload chunk size of each OpenMP thread. Therefore, the cost function $E(a_i)$ is related to the execution time of an OpenMP parallel for construction and the variable a_i is related to the chunk size, in the dynamic OpenMP distribution.

3.4 Implementation aspects of RTM

The RTM program developed to test the proposed auto-tuning is implemented in C, using a hybrid parallel approach. MPI distributes shots among the nodes of a distributed memory system while OpenMP schedules chunks of the 3D mesh representing the spatial domain to cores of a shared memory system.

The wave propagator of our RTM implementation solves the wave equation by FDM, using an eighth-order in space and second-order in time stencil. We used a non-reflecting boundary condition to absorb the energy at the boundaries, as described in (Cerjan et al. 1985).

In order to avoid the use of secondary storage and memory, our RTM code implements

the optimal checkpointing strategy described in Symes (2007) and Griewank & Walther (2000). Details of our RTM implementation are shown in Algorithm 1, which is further discussed in Section 3.5.

3.5 CSA-based auto-tuning

Katagiri et al. (2003) de nes three types of auto-tuning: i) install-time: when the estimation procedure is affected by machine environments, ii) before execution-invocation: when the estimation procedure is affected by user's knowledge, input parameters or number of processors, for example, and iii) runtime: when the estimation is affected by other parameters generated in runtime. In this paper, we propose a runtime auto-tuning for adequately determining the size of parallel loops subsets to be dynamically distributed among OpenMP threads.

Since the relation between the chunk size of the parallel loops and the total execution time of a program is unknown, the use of a stochastic optimization method is mandatory. This relation is particularly challenging for the FDM because of its stencil. Using a multidimensional stencil means that the access to memory is non-linear at each wave propagation time step, making it more complex to avoid cache misses. For this reason, the proposed auto-tuning employs CSA to nd the chunk size that minimizes the execution time.

The CSA parametrization is slightly more straightforward than the SA, but still a challenging task. For all the tests performed in this work, we adopted the following parametrization rules. For the proposed auto-tuning algorithm, we determined, by trial and error, the values of $T_0^{gen} = 0{:}1$ and $T_0^{ac} = 0{:}9$, for the initial generation and acceptance temperatures, respectively. The number of iterations, N, and the number of SA optimizers, m, in uence the convergence and exploration of the solution variables space. Large values of N and m might result in chunk size estimates closer to the global optimum, with the

Table 3.1: CSA parameters used in the numerical experiments.

T_0^{gen}	T_0^{ac}	N	m
0.1	0.9	30	5

drawback of greater execution times. We found a good compromise with the values $N = 30$ and $m = 5$. The parameters σ_D^2 and a do not need to be configured, they can be kept fixed, in all simulations, according to Xavier-de Souza et al. (2010), with $\sigma_D^2 = 0.99\,\frac{m-1}{m^2}$ and $a = 0.005$. For all the tests that we performed, with different data sets, problem sizes, and machines, the CSA algorithm has shown to be quite robust to the initialization of the parameters. We were able to achieve consistent results by using the same parameter values in all the tests. The CSA configuration parameters used in the experiments are summarized in Table 3.1.

Four main parallel loops of the RTM are able to have their chunk sizes defined, namely, i) the forward propagation of the source (Line 14 of Algorithm 1), ii) the backward propagation of the observed data (Line 24 of Algorithm 1), iii) the insertion of the receivers data (Line 28 of Algorithm 1) and iv) the image condition (Line 33 of Algorithm 1). The optimal checkpointing strategy (Line 31 of Algorithm 1) recomputes some of the forward propagation time steps and can also have its chunk size defined by an auto-tuning method.

Since the three propagation loops are essentially the same, we only apply the proposed auto-tuning for the forward propagation. The chunk size obtained is then applied to the forward propagation, the backward propagation, and the checkpointing. On the other hand, the receivers' insertion and image condition loops are not auto-tuned. These loops have a significantly smaller dimension in comparison with the propagation loops. In our tests, these loops together spent less than 2% of the total execution time. Furthermore, they mostly perform linear access to memory, which is ideal for a static distribution. Its overhead may overcome the benefit of auto-tuning the receivers' data insertion and image condition loops. As shown in Algorithm 1 (Line 9), the proposed auto-tuning is performed

Algoritmo 1: Reverse Time Migration with auto-tuning. ns is the number of time steps. t_i is the i-th time step in the RTM algorithm.

```
1: distribute shots among nodes using MPI
2: read RTM parameters
3: initialize checkpointing variables
4: compute absorbing boundaries coefficients
5: initialize auto-tuning parameters
6: #OpenMP parallel section begin
7: for all shots location do
8:    read shot seismogram
9:    if it is the first shot then
10:      autotuning() (See Algorithm 2)
11:   end if
12:   for (t_i = 0 to ns) do
13:      #OpenMP parallel loop using the auto-tuned chunk size in a dynamic
         distribution
14:      for all grid points do
15:         compute the wave field
16:      end for
17:      add the source wavelet
18:      if (t_i is a checkpoint) then
19:         save Checkpoint
20:      end if
21:   end for
22:   for (t_i = ns - 1 to 0) do
23:      #OpenMP parallel loop using the auto-tuned chunk size in a dynamic
         distribution
24:      for all grid points do
25:         compute the wave field
26:      end for
27:      #OpenMP parallel loop using static distribution
28:      for all receivers location do
29:         inject observed data samples at time t_i
30:      end for
31:      get forward wave field at t_i from the checkpoints using the auto-tuned chunk
         size in a dynamic distribution
32:      #OpenMP parallel loop using static distribution
33:      for all main grid points do
34:         perform image condition
35:      end for
36:   end for
37: end for
38: #OpenMP parallel section end
39: reduce all nodes migrated sections
```

only for the first shot. All the following shots use the same chunk size computed for the first shot.

Algorithm 2 details the implementation of the proposed auto-tuning. The initial set of solutions (chunk sizes) is randomly chosen in the interval $[5, N_{loop}/N_{threads}]$. We disregarded small chunk sizes because of the high overhead to dynamically schedule them. Chunk sizes greater than the chunk size of the static distribution ($N_{loop}/N_{threads}$) are also not taken into consideration because they would lead to the number of blocks less or equal than the number of threads. Thus, the distribution would be forced to be static.

For each CSA iteration, each optimizer only measures the execution time of the first time step in the forward propagation, using its current chunk size (Lines 6 and 13). As shown in (Barros et al. 2018), the runtime of the first time step can accurately represent the total propagation execution time. This first-time step is performed twice (Line 4) and only the elapsed time of the second repetition is registered (Lines 5 and 12) in order to avoid cache population effects. The CSA then uses those time measures as the cost function values and generates the next set of solutions (Line 17).

3.6 Numerical experiments

Our experiments were conducted on 3 different computational environments, namely:

Leuven: Single compute node hosting four sixteen-core AMD Opteron(TM) Processor 6376 at 2.3 GHz and 256 GB RAM. This equipment is located at the Universidade Federal do Rio Grande do Norte (UFRN).

NPAD: 68 compute nodes. Each compute node hosts two CPUs Intel Xeon Sixteen-Core E5-2698v3 at 2.3 GHz and 128 GB RAM DDR4 2133. It is equipped with a 60 TB Lustre parallel distributed file system. This equipment is located at the High-Performance Computing Center at UFRN (NPAD/UFRN).

Yemoja: 860 compute nodes. Each compute node hosts two 10-core Intel Xeon

Algoritmo 2: Proposed auto-tuning method, function autotuning() of Algorithm 1. t_i is the i-th time step in the RTM algorithm.

```
1:  t_i = 0
2:  for all (auto-tuning iterations) do
3:      for all optimizers do
4:          for (i = 1 to 2) do
5:              if (i == 2) then
6:                  time measure begin
7:              end if
8:              #OpenMP parallel loop using the current chunk size in a dynamic
                distribution
9:              for all grid points do
10:                 compute the wave field
11:             end for
12:             if (i == 2) then
13:                 time measure end
14:             end if
15:         end for
16:     end for
17:     CSA generates a new solution for each optimizer from the time measures
18: end for
19: return the solution with the lowest cost function
```

Figure 3.1: Seismic traces (blue) from the analytical solution and (red) from our wave propagator. The source is a Ricker wavelet with peak frequency of 20 Hz. The distance between source and receiver is 200 m. The medium has a constant velocity of 2000 m/s.

E5-2690 Ivy Bridge v2 at 3 GHz. 200 nodes with 256 GB of RAM and 656 nodes with 128 GB RAM. It is equipped with an 850 TB Lustre parallel distributed le system. This equipment is located at the Manufacturing and Technology Integrated Campus of the National Service of Industrial Training (SENAI CIMATEC).

In order to validate the 3D acoustic wave propagator used in our RTM program, we compared a seismic trace computed by our program with the 3D acoustic analytical solution, computed based on (De Hoop 1960), in a homogeneous velocity model. Figure 3.1 shows that our wave propagator provides an accurate approximation to the analytical solution of the 3D acoustic wave equation. The waveform of both curves is essentially the same, as well as the amplitudes.

For all the following tests, f_{peak} = 20 Hz, the time sampling is 1 ms, the number of

Table 3.2: Number of buffers and checkpoints used in the experiments as function of the input size. The input size does not include the absorbing border. n_1, n_2 and n_3 are the number of samples for the spatial dimensions x_1, x_2 and x_3, being the latter the vertical dimension.

Input size (n_1 n_2 n_3)	n_b	n_c
201 401 401	170	3330
401 401 401	100	3400
801 401 401	56	1848

time steps is 3501, the spatial resolutions $\Delta x_1 = \Delta x_2 = \Delta x_3 = 10$ m and the absorbing border thickness is 50 points in all directions of the 3D mesh. We obtain $c(x)$ by using a two layers model with a flat interface positioned at the center of the vertical dimension, where the top and bottom layers have velocities of 1400 m/s and 2000 m/s, respectively. The number of buffers (n_b) and checkpoints (n_c) depends on the size of the input, as shown in Table 3.2. The numbers of buffers were chosen in order to use up to 128 GB of RAM. The numbers of checkpoints are optimal, according to (Griewank & Walther 2000).

The first set of experiments compares the performance of the proposed auto-tuning method and the OpenMP *auto* and *static* schedules, performing RTM of a single seismic shot, in different computational resources.

As seen in Figure 3.2, the proposed auto-tuning strategy outperforms the *auto* and *static* schedules in this set of experiments. The proposed method speedups were 2% and 5.8:1%, compared with the *auto* and *static* schedules, respectively, on Yemoja. Regarding the tests performed in Leuven, the proposed method speedups were 8% and 135%, compared with the *auto* and *static* schedules, respectively.

Since Leuven has more core units per node, its maximum chunk size ($N_{iloop} / N_{threads}$) is smaller than it is for NPAD and Yemoja. This fact leads to a significantly smaller search domain for the CSA and might be the reason behind the improved auto-tuning performance in this platform.

Figure 3.3 shows detailed results obtained on NPAD. It shows that, in this particular case, the slowest execution time of the proposed auto-tuning method is faster than the

Figure 3.2: Single shot RTM runtime for the proposed auto-tuning, compared with the auto and the static scheduling types, in 3 different machines, namely NPAD, Yemoja and Leuven. These tests were performed with an input size of $(n_1 \; n_2 \; n_3) = 401 \; 401 \; 401$. Each point is a median of at least 5 executions.

median execution time of both the auto and the static schedules.

On NPAD, taking the median of 10 executions, the proposed auto-tuning method speedup was 10% when compared with the auto scheduling, for the RTM of a single seismic shot. When comparing with the static scheduling, its speedup was 1%, also for the RTM of a single seismic shot.

The second set of experiments compares the performance of the proposed auto-tuning method and the OpenMP auto and static schedules performing a single shot RTM, for different input sizes.

Figure 3.4 shows that the proposed auto-tuning strategy outperforms the auto and static scheduling types, in this set of experiments. The proposed method speedups were 6.14% and 7.14%, compared with the auto and static scheduling types, respectively, for an input size of $201 \; 401 \; 401$. When tests were performed for an input size of $401 \; 401 \; 401$, the proposed method speedups were 6.10% and 10.9%, compared with the auto and

Figure 3.3: Single shot RTM runtime for the proposed auto-tuning, compared with the auto and the static scheduling types. These tests were performed at NPAD for an input size of(n_1 n_2 n_3) = 401 401 401. Each box plot represents a set of 10 executions.

Figure 3.4: Single shot RTM runtime for the proposed auto-tuning, compared with the auto and the static scheduling types, for 3 different input sizes, namely 201 401, 401 401 401 and 801 401 401. These tests were performed at NPAD. Each point is a median of at least 5 executions.

static scheduling types, respectively. Finally, for an input size of 801 × 401 × 401, the proposed method speedups were 44% and 3%, compared with the auto and static scheduling types, respectively.

The larger the input size, the bigger the chunks of the static distribution. For the input sizes of 201 × 401 × 401, 401 × 401 × 401, and 801 × 401 × 401, the chunk size of the static distribution are 9, 15, and 27 millions of loop iterations. By working with larger chunks, the data locality of the static distribution reduces, which explains its performance decrease for larger input sizes.

On the other hand, for the same input sizes, the median of the chunk sizes chosen by the proposed auto-tuning were 71, 264, and 130 thousands of loop iterations. By processing smaller chunks, the set of data being processed by all cores at a time are physically closer. This data locality increases the reuse of the data in cache memory.

The third set of experiments compares the performance of the proposed auto-tuning method and the OpenMP auto and static scheduling types, performing RTM in multiple seismic shots. For this test, sets of 1, 2, 4, and 8 shots were migrated in a single node. In the case of using multiple nodes, the same strategy would be replicated at each node.

The proposed auto-tuning method outperforms the auto and the static schedules for all the tested number of shots. As shown in Figure 3.5 for the RTM of multiple shots, since the proposed auto-tuning is performed only a single time and the resulting chunk size is used in the parallelization of all the shots, its relative overhead reduces as the number of shots increases. This fact causes the highest speedups of the proposed auto-tuning to be achieved for the highest number of shots tested. In this scenario, when 8 shots were migrated, the proposed auto-tuning speedups were 4% and 28%, compared with the auto and static schedules.

Figure 3.5: Multiple shots RTM runtime for the proposed auto-tuning, compared with the auto and the static scheduling types. For each amount of shots, all shots were migrated in a single node. These tests were performed at NPAD for an input size of $(n_x, n_y, n_z) = 401 \quad 401 \quad 401$. Each bar represents a set of at least 5 executions.

3.7 Conclusions

We have proposed a CSA-based auto-tuning strategy for properly choosing the optimal chunk size that reduces the execution time of a 3D reverse time migration algorithm, implemented in parallel with OpenMP. The proposed approach is designed to work wherever the code is executed, being robust for changes in computational environment parameters, such as the number of threads, processors, RAM, and compiler.

Experiments of auto-tuning the RTM of a single seismic shot have shown that the proposed auto-tuning outperforms OpenMP auto and static scheduling types when using different input sizes and computational resources. In this case, the proposed auto-tuning reached speedups up to 44%. The proposed method presents better results for larger inputs, which improves the parallel scalability of 3D RTM.

We have also extended the proposed method to auto-tune the RTM of multiple seismic

shots, by just reusing the chunk size obtained in the first shot of each node to the following shots migration. In this case, the proposed auto-tuning reached speedups up to 28% for all the number of shots used. the proposed auto-tuning method outperformed the standard static OpenMP schedules.

Different strategies for auto-tuning the 3D RTM of multiple shots, as well as the influence of the CSA parameters on the proposed auto-tuning technique are matters of further research.

Chapter 4

Distributed-Memory Load Balancing with Cyclic Token-based Work-Stealing Applied to Reverse Time Migration

The migration of seismic data is the process that attempts to build an image of the Earth's interior from recorded eld data. Migration places these data into their actual geological position in the subsurface using numerical approximations of either wave-theoretical or ray-theoretical approaches to simulate the propagation of seismic waves (Yilmaz 2001b).

The wave-theoretical approach to the propagation of seismic waves employs the nite difference method (FDM) (Alterman & Karal 1968, Kelly et al. 1976) to numerically solve the equation describing the movement of the waves (Yilmaz 2001a, Claerbout & Doherty 1972). This approach is prevalent among the geophysical community due to its capacity for dealing with substantial velocity variations in complex geology (e.g., pre-salt).

Reverse time migration (RTM) (HEMON 1978, Baysal et al. 1983, McMechan 1983, Whitmore 1983, Kosloff & Baysal 1983) implements this approach. It is one of the most known FDM-based migration methods. RTM is computationally intensive in terms of data storage and handling, and its use of high-complexity algorithms. Therefore, exploit-

ing parallelism is mandatory for RTM implementations in 3D Earth models (3D RTM) (Araya-Polo et al. 2009).

Parallel architectures can be classi ed as shared memory, when there is a single memory address space available to all processing units (e.g., nodes or cores), or distributed memory otherwise (Diaz et al. 2012). Many scienti c and industrial computational resources are distributed memory systems composed of multi-processor nodes with shared memory systems. A hybrid parallel application works at these two levels of parallelism. It can distribute the total workload among the nodes of a distributed memory system. Each node, then, distributes its subset of the workload among the processing units of its shared memory system. Parallel machines can also be described as heterogeneous when they have processing units built from different types of hardware, or homogeneous otherwise (Pacheco 2011).

One of the main concerns in parallel computing is the ef cient use of the available computational resources. Some applications, such as RTM, may suffer from load imbalance. A way of dealing with this issue is to employ load balancing techniques, which usually refer to the distribution of the workload among the available computational resources (e.g., nodes, processors, cores). The objective is to minimize the idling of the computational resources while there are still tasks remaining to be processed.

Ensuring the load balancing for an RTM is especially challenging in distributed memory systems. Distributing the workload in equal amounts of tasks for each computational node may not be optimal. Even for homogeneous computational systems with an evenly distributed workload, several factors may be a source of load imbalance. It can be intrinsic to the application itself or caused by program-external factors such as runtime environment routines (e.g., system calls) and resource availability. The competition for shared resources, such as the parallel le system or the network, can cause idling due to resource contention as the availability of the resources may differ across the nodes and along time.

Work-stealing (WS) is one of the main load balancing strategies. The fundamental

idea of WS methods is that idle processing units steal tasks from the others (Blumofe & Leiserson 1999) in an attempt to avoid the performance overhead of a centralized entity being responsible for the task scheduling. Processes stealing tasks are the thief processes, whereas the processes with stolen tasks are the victim processes. Nevertheless, in the context of distributed systems, some WS implementations present problems with too many failed steal attempts, with communication overhead, with the victim selection, and with the termination strategy.

This paper proposes a cyclic token-based work-stealing (CTWS) algorithm for distributed memory systems applied to RTM. The novel cyclic token approach reduces the number of failed steals, avoids communication overhead, and simpli es the victim selection and the termination process. The proposed work-stealing method was implemented in C using the message passing interface (MPI) (Clarke et al. 1994) standard. The communication was implemented by remote memory access (RMA) using MPI one-sided communication (Gropp et al. 1999). This communication model allows the thief processes to perform the work-stealing without directly involving (or interrupting) the victim processes, thus further reducing communication overhead. Our 3D RTM code was implemented in C using MPI for distributed-memory parallelism across nodes, and OpenMP (Dagum & Menon 1998) for thread-level parallelism within nodes.

The contribution of this paper to the elds of distributed load balancing and RTM are:

1. the proposition of a novel approach to implementing load balancing with WS in distributed systems based on a cyclic token;
2. the mitigation of important WS implementation problems in distributed systems by the proposed cyclic token approach as it avoids failed steals and simpli es the victim selection and the termination strategy;
3. the reduction of the communication overhead of the WS distributed implementation by the use of MPI one-sided communication to implement the cyclic token approach;

4. a detailed evaluation of the conventional load balancing technique for 3D RTM showing that load imbalance is signi cant due to resource contention;
5. improvements in the execution time of 3D RTM of about 14% and reductions of the load imbalance of about 78%.

The rest of this paper is organized as follows. Section 4.1 shows the basics of RTM and describes our RTM implementation. Section 4.2 introduces the work-stealing method proposed in this work. Section 4.3 details the application of the proposed technique to the RTM. Section 4.4 discusses the performance of the RTM with and without the proposed approach. Section 4.5 presents a literature review of related works contrasting them with the proposed approach. Finally, Section 4.6 summarizes this work and proposes future research.

4.1 RTM and Static Load Balancing

In a seismic re ection survey, an acoustic source at a given location (a “seismic shot”) generates a wave that propagates into the subsurface. Each time the wave travels through an interface between two layers with different impedance, part of its energy is re ected and is eventually registered at a set of receivers. This procedure is repeated for different shot locations in order to cover the whole area of interest. The data recorded by a single receiver for a single seismic shot is called a seismic trace, and a set of traces is called a seismogram. The seismograms can pass through many processing steps to nally provide an image of the subsurface.

Migration is one of the most critical steps in processing seismic data. It aims to position the re ection interfaces properly in the subsurface. A migrated section is an image representing the geological structures in the region of interest. This section can be used for interpretation purposes, often to locate and characterize oil and gas reservoirs.

Reverse time migration (RTM) (Baysal et al. 1983, Kosloff & Baysal 1983) is one

of the most known migration methods. The main steps of an RTM are presented in Algorithm 3. The first step, the forward propagation, simulates the incident wave field by propagating a source wavelet through the region of interest. The backward propagation generates the reflected wave field by propagating the seismogram comprised of the seismic traces from a shot, a common shot gather, in reverse time order.

Algoritmo 3: Main steps of a reverse time migration

```
1: for all (shots locations) do
2:     forward propagation
3:     backward propagation of the common shot gather
4:     image condition
5: end for
```

Both forward and backward propagation can be performed by iteratively solving, over a discrete grid, the acoustic wave equation, described as:

$$\frac{\partial^2 u(x)}{\partial x_1^2} + \frac{\partial^2 u(x)}{\partial x_2^2} + \frac{\partial^2 u(x)}{\partial x_3^2} = \frac{1}{c(x)^2}\frac{\partial^2 u(x)}{\partial t^2} + s(t). \tag{4.1}$$

In (4.1), $x = (x_1; x_2; x_3)$ are the spatial dimensions, $u(x)$ is the pressure wave field, $c(x)$ is a velocity model, t is the time dimension and $s(t)$ is the source, i.e., a wavelet representing the seismic shot.

The finite difference method (FDM) is often used to numerically solve (4.1) by approximating its PDEs (partial differential equations). Approximations of higher orders provide more accurate results, with smaller numerical errors. Spatial and time restrictions should be observed when solving finite differences by a numerical approach (Carcione et al. 2002).

The content of the velocity model, $c(x)$, plays an important role from the geophysical perspective. Its complexity is the reason why RTM is used. It also influences the computational cost of the RTM as it determines the spatial and time resolutions. In other words, the maximum and minimum values of the velocity model, c_{min} and c_{max}, directly

in uence on the total number of operations performed by an RTM. However, for a xed f_{max}, two models having the same c_{min} and c_{max} demand the same time and spatial resolutions, and generally incurs in the same computational cost, no matter they have different geological structures.

The wave propagation via FDM is performed over a limited grid representing the region of interest. Nevertheless, the region where the seismic survey takes place is not restricted to that region of interest. For this reason, it is common practice to add extra points to the limits of the grid, in order to absorb the energy reaching the borders of the model (Cerjan et al. 1985).

RTM relies on the principle that the incident and the re ected wave elds, $u_i(x;t)$ and $u_r(x;t)$, correlate at the re ection interfaces. An image condition with the following mathematical description performs this correlation (Claerbout & Doherty 1972).

$$I(x) = \sum_{t=0}^{T} u_i(x;t) \; u_r(x;t)\,dt, \tag{4.2}$$

where T is the total time of the the simulation.

The three RTM steps (as Algorithm 3 shows) are repeated for each shot location generating one migrated section per shot. The nal migrated section is achieved by summing up all shot migrations.

The RTM method used in the experiments described in this paper is an extension of the RTM introduced by Nunes-do Rosario et al. (2015). It is implemented in C with a hybrid parallel approach. MPI is used to distribute the workload of different shots among computational nodes of a distributed system, and OpenMP is employed to parallelize internal loops of each shot processing. The implemented parallel RTM code is described in Algorithm 4.

Absorbing boundaries are implemented in our RTM code as reduction coef cients (Line 3 of Algorithm 4) that taper the wave eld amplitudes in a layer of grid points sur-

Algoritmo 4: Reverse time migration with work-stealing load balancing. ns is the number of time steps, s_{shot} is the number of the shot being processed.

```
1: statically distribute shots among nodes using MPI
2: read RTM parameters
3: compute absorbing boundaries coefficients
4: #OpenMP parallel section begin
5: for all (shots locations of the processes) do
6:    read shot seismogram
7:    for (t_i = 0 to ns) do
8:       #OpenMP for
9:       for (all grid points) do
10:         compute the wave field
11:      end for
12:      add the source wavelet
13:      write wave field to disk
14:   end for
15:   for (t_i = ns - 1 to 0) do
16:      #OpenMP for
17:      for all (grid points) do
18:         compute the wave field
19:      end for
20:      #OpenMP for
21:      for all (receivers location) do
22:         inject observed data samples at time t_i
23:      end for
24:      read forward wave field at t_i from disk
25:      #OpenMP for
26:      for all (main grid points) do
27:         perform image condition
28:      end for
29:   end for
30: end for
31: #OpenMP parallel section end
32: reduce all nodes migrated sections
```

rounding the mesh as proposed in (Cerjan et al. 1985). The acoustic wave equation (4.1) is solved for each propagation by the FDM with a second-order approximation in time and eighth order approximation for each spatial dimension (Lines 10 and 18 of Algorithm 4).

The incident wave eld is stored on disk (Line 13 of Algorithm 4) at each forward wave propagation time step. At each backward wave propagation time step, the incident wave eld is read from disk (Line 24 of Algorithm 4) in order to perform the image condition (Line 27 of Algorithm 4).

The load balancing of the RTM described by Algorithm 4 is static. An equal amount of shots, or nearly equal, is allocated to each node at the beginning of the algorithm (Line 1). From this point on, no more load balancing decisions are taken. If a process nishes processing all its shots, i.e., leaves the shots loop (Lines from 5 to 30), it has to wait for the slowest process in order to collectively summing up all shot migrations, i.e., performing the reduction operation, through the command MPI_Reduce, in Line 32. Since MPI does not provide task schedulers, the static schedule is often used because of its ease of implementation.

4.2 Cyclic Token-based Work-Stealing

Cyclic token-based work-stealing (CTWS) is the load balancing method for distributed memory systems introduced in this chapter. It is a library implemented in C using MPI. In order to reduce communication overhead, CTWS is implemented using MPI one-sided communication.

Since MPI-2 (Gropp et al. 1999), MPI speci cation includes the concept of one-sided communication. This MPI feature implements RMA, which allows processes to make a portion of their local memory available for access by other processes. In one-sided communications, the process that accesses the memory is called the origin process while

the process whose memory is accessed is called the target process (Park & Chung 2009). All processes involved in one-sided communication must collectively create windows. A window is a structure with information on the memory regions which the processes make available for RMA.

Many operations are available on MPI one-sided communication. Our work mainly uses the operations MPI_Put and MPI_Get, which are used to write to and read from remote memory, respectively. These operations are passive, i.e., the target process is not involved in the operation. Therefore, the target process keeps computing its tasks while the RMA operation is performed.

A token and a list of remaining tasks per process are the two main elements of the proposed work-stealing technique. Both are implemented as MPI one-sided communication windows. The token was implemented as an integer number. It is initialized as 0, meaning that, according to the list of remaining tasks, there are tasks to be stolen. The first process to figure out that no more tasks can be stolen sets the token to 1, i.e., sets the token to finish.

The token gets passed around through an MPI_Put operation in a round-robin fashion. Only the process owning the token can update the list of tasks per process and steal tasks. This strategy avoids deadlocks that would be caused by two processes trying to steal from each other at the same time. In such a case, both of them would have to grant access to both of their lists of remaining tasks. If both of them granted access to one of these lists, a deadlock would occur.

In the initialization (Line 1 of Algorithm 5), the token must be allocated to a single process. The shots to be processed are equally distributed among the processes. Each process has its copy of this list of tasks per process. This list is implemented as an array of integers where the i-th element is the number of remaining tasks of the i-th process. The functions `getTask()` and `updateList()` are responsible for managing the token and the list of tasks.

Algoritmo 5: Cyclic Token-based Work-Stealing. t_{id} is the task identification number.

```
1: Initialize CTWS variables
2: t_id = getTask()
3: while (t_id != -1) do
4:     for all (iterations of task t_id) do
5:         updateList()
6:         Compute an iteration of t_id
7:     end for
8:     t_id = getTask()
9: end while
```

The proposed strategy is designed for applications with iterative tasks. At each task iteration, the function `updateList()` is called by each process. It first verify whether it possesses the token (Line 5 of Algorithm 5). Should it have the token and it is not set to finish, the process updates its number of remaining tasks in its list and copies its list to the next process through an MPI_Put operation in a ring fashion. When the process does not have the token, it simply continues working on its tasks. By doing so, any process has a close approximation of the current amount of remaining tasks of each process. This information is then used to lead the stealing stage.

The core of the proposed work-stealing strategy is the function `getTask()` (Lines 2 and 8 of Algorithm 5), which is detailed by the flow chart of Figure 4.1. When a process has shots to be processed, `getTask()` returns the first of them. Otherwise, the process attempts to steal tasks from other processes.

Only the process possessing the token can try to steal tasks. For this reason, the first step of the proposed strategy is to ensure that the process has the token. If it does not, it will perform a busy-wait by repeatedly verifying whether it posses the token. Once the token arrives, the thief process checks whether the token is set to finish. If so, no work-stealing is needed, and the process continues to the reduction operation.

However, when the token is not set to finish, the thief process tries to steal from the process with more remaining tasks, according to the thief's list of tasks. Since the list of

Figure 4.1: Detailed flow chart of the function getTask() that is responsible for determining which is the next task to be processed by each process. In this work, the task unit to be processed is the RTM of each shot gather.

remaining tasks per process is an approximation, the real number of remaining tasks may have changed by the stealing time. For this reason, the thief process verifies the actual number of remaining tasks of the victim process through an MPI_Get operation. If the victim process does not have tasks to be stolen, the thief process updates its list of tasks and restarts the procedure by finding a new victim process in its updated list of tasks. Should there be no more tasks left to be stolen, then the thief process sets the token to finish, forwards it to the next process, and continues to the reduction operation.

When the thief process finds a victim process with tasks to be stolen, it uses MPI one-sided communication (MPI_Put and MPI_Get) to steal a subset of the remaining tasks of the victim process. For the tests in this work, half of the remaining tasks are stolen. As discussed by Dinan et al. (2009), stealing half of the tasks of the victim increases the number of possible victims for the next steals. This strategy aims to improve scalability by reducing the time to locate and steal tasks. Finding an optimal number of tasks to be stolen is not of the scope of this work.

This stealing procedure is seamless to the victim process, i.e., the victim process will not stop processing its current task to communicate with the thief process. At this point, the thief process starts to work on the first of its stolen tasks. Should the victim process have a single remaining task, then the thief process will try to steal it. In case the victim process also tries to start processing its only left task at the same time, the race condition is avoided by the one-sided communication operations `MPI_Win_lock` and `MPI_Win_unlock` set to the type `MPI_LOCK_EXCLUSIVE`. These commands ensure mutual exclusion allowing a single process to access the window at a time.

4.3 CTWS Applied to RTM

In this work, the task unit to be processed is the RTM of each shot gather. In other words, the iterations of the shots loop (Lines from 5 to 30 of Algorithm 4) are distributed

to the nodes of a distributed system using CTWS. For this reason, the commands controlling the shots loop of the RTM must be replaced by the commands controlling the tasks loop of CTWS. Line 5 of Algorithm 4 is replaced by Lines 2 and 3 of Algorithm 5, and Line 30 of Algorithm 4 is replaced by Lines 8 and 9 of Algorithm 5. In this context, the task identification number t_{id}, represents the number of the shot gather. The function updateList() is called inside of both the forward propagation loop (Lines from 7 to 14 of Algorithm 4) and the backward propagation loop (Lines from 15 to 29 of Algorithm 4).

The larger the number of processes, the shorter the time that each process will have the token. Because of that, the overhead caused by updateList() in the RTM is proportionally smaller for larger numbers of processes and larger input sizes. On the other hand, by running updateList() and having the token in each process fewer times, the list of remaining tasks per process is more prone to be out of date. This way, the number of unsuccessful steals attempts performed by getTask() may increase, and so its overhead.

4.4 Results and Discussion

The experiments were performed on Yemoja, an 856 node supercomputer. Each computational node hosts two processors 10-core Intel Xeon E5-2690 Ivy Bridge v2 at 3:00 GHz. 200 nodes are equipped with 256 GB RAM and 656 nodes with 128 GB RAM. This supercomputer employs an 850 TB Lustre parallel distributed file system. Yemoja is located at the Manufacturing and Technology Integrated Campus of the National Service of Industrial Training (SENAI-CIMATEC). Both the 128 GB RAM and the 256 GB RAM were used in the following experiments. Since the total amount of RAM required by our RTM implementation is significantly inferior to 128 GB, this fact does not influence the algorithm performance.

In order to validate the 3D wave propagator, which underlies the 3D RTM algorithm used in the experiments, we compared a seismic trace generated by our propagator with

the analytical solution, computed according to (De Hoop 1960), in a homogeneous velocity model. The source was a Ricker wavelet with a peak frequency of 20 Hz. The distance between source and receiver is 200 m. The medium has a constant velocity of 2000 m/s. In this experiment, our wave propagator provided a very accurate approximation to the 3D waveform analytical solution with a mean squared error of 10^{-14}.

For the following experiments, the size of the input grid is $401 \times 401 \times 401$, the peak frequency of the source wavelet is 20 Hz, the time sampling is 1 ms, the spatial sampling is 10 m, and the number of time steps is 3501. $c(1,x)$ is a two layers model with a horizontal interface positioned at the center of the vertical dimension. The velocity is 1400 m/s for the top layer and 2000 m/s for the bottom layer.

The programs were compiled with the gcc compiler using the optimization flag -O3 and OpenMPI 1.5.2 for all experiments. A single MPI process was created at each computational node. We used HPCToolkit performance tools (Adhianto et al. 2010) to measure the execution times and the overhead of our strategy. For all the following experiments using CTWS, the load balancing overhead was inferior to 4%. A single experiment was performed at a time in order to avoid multiple tests competing for the shared resources of the cluster.

Firstly, we measured the load imbalance of the 3D RTM without applying a dynamic load balancing technique. For that we ran the RTM of 40, 80, 160, 320 and 640 shots with 4, 8, 16, 32 and 64 nodes, respectively. As shown in Figures 4.2 and 4.3, for the experiment with 4 nodes, the average idle time per node is 7.2% of the total time of 18.4 h. As the number of nodes increases up to 64, the average idle time per node increases to 23.4% of the total time of 39.2 h. Although the number of shots per node is the same for each experiment, the competition for shared resources of the cluster (e.g., network and parallel file system) increases the runtime as the number of nodes increase.

Figure 4.4 details the execution of the 3D RTM ran over 64 nodes without applying a dynamic load balancing technique. Although the workload is distributed evenly among

Figure 4.2: 3D RTM maximum process idle time and average process idle time with 4, 8, 16, 32 and 64 nodes. Both RTM implementations with and without the proposed work-stealing method (CTWS) process 10 shots per node.

Figure 4.3: 3D RTM total runtime with 4, 8, 16, 32 and 64 nodes. Both RTM implementations with and without the proposed work-stealing method (CTWS) process 10 shots per node.

Figure 4.4: Example of 3D RTM runtime per process and shot ran over 64 nodes. The shots are numbered in the order they are processed in the node they were assigned to. The processes are sorted by their idle time.

the homogeneous nodes, the runtime of a single shot RTM ranges from 5 to 9.13 h. The fastest node stays idle for 4.17 h while the other nodes finish their tasks, i.e., 45% of the total runtime. Factors as a race condition for the network and the parallel storage system can cause such load imbalance.

Figure 4.3 also shows results generated by the proposed work-stealing technique employed in the same set of experiments. The proposed technique presented a maximum average idle time of 1.9%, showing its effectiveness in balancing the load. For the experiment with 4 nodes, the average idle time per node is 3.2% of the total time of 18.2 h. As the number of nodes increases up to 64, the average idle time per node slightly increases to 9.9% of the total time of 3.4 h.

The total execution times displayed in Figure 4.3 show that the proposed technique outperforms the 3D RTM with the conventional static load balancing when using a more substantial number of nodes. The total runtime was reduced by 14.49%, 14.41%, and 10.38% when ran over 16, 32, and 64 nodes, respectively. For the fewest number of nodes, how-

Figure 4.5: Example of 3D RTM runtime per process and shot ran over 64 nodes using the proposed work-stealing. The shots numbers refer to the order they were processed within its node. The processes are sorted by their idle time.

ever, the proposed technique performance was similar to the static load balancing. The total runtime increased by 2.3% when ran over 8 nodes and decreased by 9.0% when ran over 4 nodes. Industry-scale RTM, however, is usually performed over a large number of computational nodes. Regarding the load imbalance, as shown in Figure 4.2, the proposed method was able to reduce the average idle time for all the performed tests. In the best scenario, when ran over 16 nodes, the idle time was reduced from 21.5% to 4.5%, representing a 78% improvement in the effective use of the resources.

Figure 4.5 details the 3D RTM execution over 64 nodes, using the proposed work-stealing technique. Although there are differences between the processing times of a single shot, nodes with better resource availability steal shots from others that are slower, thus improving load balancing. The least busy node processed only 6 shots while the busiest node processed 14 shots.

Table 4.1 presents all steal attempts of the example of Figure 4.5. It shows that 37 out of 38 steal attempts were successful. In five cases (13%) of the steals, a task was stolen

for the second time:

1. in steal 18, process 20 stole the task 387 from process 21. Before that, in stealing 5, process 21 stole the same task from process 38;
2. in steal 23, process 28 stole the task 178 from process 29. Before that, in stealing 8, process 29 stole the same task from process 17;
3. in steal 24, process 34 stole the task 478 from process 9. Before that, in stealing 10, process 9 stole the same task from process 47;
4. in steal 28, process 21 stole the task 307 from process 16. Before that, in stealing 3, process 16 stole the same task from process 30;
5. in steal 29, process 41 stole the task 318 from process 18. Before that, in stealing 9, process 18 stole the same task from process 31;

In our test case, stealing the same task multiple times does not represent an additional overhead since there is no signi cant extra cost to move a task between processes. This fact occurs because all the data is available to all processes through the Yemoja's parallel le system. A method that considers the cost of moving tasks is left to future work.

Table 4.2 shows the total number of steal attempts and failed steals varying the number of processes. In this test set, 1 out of 55 steal attempts was unsuccessful; 2%, or 98 the steal attempts were successful.

In terms of weak scalability, both the RTM with and without CTWS are not scalable as the total runtime increases when the number of shots and nodes are doubled. This fact means that RTM with CTWS may also be affected by the concurrency for shared resources of the distributed system as the number of nodes increases. However, by moving tasks to nodes with better resource availability, RTM with CTWS was able to deliver up to 1.14 speedup when compared to using a static load distribution.

Table 4.1: Steal attempts of the example of Figure 4.5.

stealing attempt	thief process	victim process	stolen tasks
1	63	20	205,206,207
2	19	24	245,246,247
3	16	30	305,306,307
4	49	34	346,347
5	21	38	386,387
6	10	55	556,557
7	22	7	77,78
8	29	17	177,178
9	18	31	317,318
10	9	47	477,478
11	15	8	88
12	50	51	518
13	30	24	248
14	62	25	258
15	32	55	558
16	63	57	578
17	17	59	598
18	20	21	387
19	2	1	19
20	23	4	49
21	22	5	59
22	42	8	89
23	28	29	178
24	34	9	478
25	35	12	-
26	35	13	139
27	30	14	149
28	21	16	307
29	41	18	318
30	49	37	379
31	13	38	389
32	15	39	399
33	50	54	549
34	18	57	579
35	5	58	589
36	0	59	599
37	3	60	609
38	38	61	619

Table 4.2: Steal attempts varying the number of processes.

number of processes	4	8	16	32	64
total steal attempts	0	2	3	12	38
failed steal attempts	0	0	0	0	1

4.5 Related Works

Several authors have proposed strategies to address the load imbalance for shared memory systems. Barros et al. (2018) introduced a runtime method based on coupled simulated annealing (CSA) (Xavier-de Souza et al. 2010) to auto-tune the workload distribution of 3D acoustic wave propagation implemented with the FDM method. Andreolli et al. (2014) and Andreolli et al. (2015) proposed a compilation-time auto-tuning based on genetic algorithms to nd the best set of parameters (e.g., workload distribution, compilation ags) for seismic applications. Sena et al. (2011) used cache blocking for the 3D RTM and proposed a procedure called Min-Worst-Min Block (MWMB) to nd an ef - cient block size. Hofmeyr et al. (2011) introduced a dynamic load balancing for multicore systems using runtime tools. Tchiboukdjian et al. (2011) proposed a method that ensures all the data in the cache memory is used before being replaced. This method was designed for applications with linear access to memory. Imam & Sarkar (2015) presented a work-stealing scheduler based on task priority queues. Balancing the computational load at the shared memory level can lead to a signi cant reduction in the execution time. Our work aims to achieve further improvement by balancing the workload at the distributed memory level.

Other authors provide methods to deal with the load imbalance of distributed memory systems. Khaitan et al. (2013) proposed a master-slave based load balancing approach. Tesser et al. (2014), Keller Tesser et al. (2017) and Tesser et al. (2018) proposed a simulation-based strategy to evaluate the performance and tune the dynamic load balancing of iterative MPI applications and applied it to a 3D wave propagation. Padoin et al. (2014) and Padoin et al. (2017) proposed combining a load balancing with techniques

of processor frequency control in order to reduce energy consumption along with execution time. These approaches differ from this work for being centralized, i.e., a single or a few computational processes take the load balancing decisions. This behavior may lead to overload at the central element and signi cantly degrade performance (Khaitan et al. 2013).

To avoid losses of performance caused by a centralized load balancing element, some authors proposed decentralized load balancing strategies. Sharma & Kanungo (2014) presented a technique to balance the computational load in heterogeneous multicore clusters, where no prior knowledge about the computational resources is required. Zheng et al. (2011) introduced a periodic load balancing strategy, where the balancing decisions are taken hierarchically in a tree fashion. Different from this work, in these methods, the processes involved in the load balancing decisions have to synchronize to exchange information. This communication synchronization overhead may reduce parallel performance.

Work-stealing algorithms (Blumofe & Leiserson 1999) have been used to provide decentralized load balancing methods for distributed systems. Martinez et al. (2016) used StarPU, a task-based runtime system, to distribute the load balance of the 3D isotropic elastic wave propagation among processors and graphics processing units (GPUs) simultaneously. They compared centralized load-balancing and decentralized work-stealing algorithms from StarPU. Khaitan & Mccalley (2014) applied dynamic load balancing with work-stealing to a contingency analysis application while Mor & Maillard (2011) proposed an MPI library for load balancing branch and bound applications. These approaches use asynchronous communication to reduce the communication overhead as the processes which originate the communication may keep working while waiting for replies from their messages. Different from our work, these papers employ two-sided communication, i.e., both the origin and the destination processes are involved in the communication. This way, the destination processes have to interrupt their computation eventually to reply to the messages they have received. Also, this kind of non-blocking communica-

tion may imply in the origin process having to wait for its reply even when overlapping communication with computation.

One-sided communication is an alternative to reduce communication overhead. This model of communication allows a process to read and write data from a remote memory region without the target process being involved. Some authors have used it in the recent literature. Li et al. (2013) used pro ling information to estimate the task grain size and guide the asynchronous work-stealing. Kumar et al. (2016) introduced a load-aware work-stealing based on a policy to choose a victim that completely avoids the failed steals. Dinan et al. (2009) discussed the design and scalability aspects of work-stealing for distributed memory systems. They also proposed a runtime system for supporting work-stealing, which implements several techniques to achieve scalability in distributed memory systems. These methods employ one-sided communication through partitioned global address space (PGAS), a programming model that provides a globally shared address space for distributed memory. On the contrary, our work employs relies on MPI one-sided communication, which has no global address space. According to Fu et al. (2018), employing PGAS often demands an important development effort to exploit these programming models thoroughly. Moreover, a vast majority of scienti c codes use MPI either directly or via third-party software.

Other authors have recently used MPI implementations of one-sided communication in areas such as large-scale multimedia content analysis (Essa & Hède 2017), graph processing (Fu et al. 2018), and matrix operations (Ghosh et al. 2016, Lazzaro et al. 2017). According to Diaz et al. (2012), MPI has been the de factostandard in HPC for the last decades. In the context of load balancing, Vishnu & Agarwal (2015) introduced a work-stealing method using MPI one-sided communication for machine learning and data mining algorithms. Different from our proposal, their approach for victim selection is either random, which may increase the number of network requests or prone to network contention because of having multiple thief processes trying to steal the same victim. More-

over, Vishnu & Agarwal (2015) employ a termination strategy that does not look at the entire victim set, potentially causing some processes to finish while there are remaining tasks to perform. Differently, we employ a termination strategy in which the processes only finish when there are no more victims to be stolen, and the token is set to finish. This is only achieved because, in our proposed work-stealing algorithm, we share and update global load information without the need for synchronization. Sharing and updating the global load information has the main advantages of helping the thief processes to make better decisions and preventing failed stealing attempts.

Regarding load balancing strategies to RTM, little effort has been employed to schedule RTM tasks among nodes of distributed systems. Several authors implement parallel RTM for distributed systems using static scheduling (Abdelkhalek et al. 2012, Qawasmeh et al. 2017, Akanksha & Kumar 2017, Chu & Stoffa 2008, Perrone et al. 2012, Chu et al. 2009, Lu & Magerlein 2013, Paul et al. 2016). As shown in Section 4.4, the use of static distribution may lead to inefficient use of the distributed computational resources as faster nodes may wait idly for the slower ones to finish their tasks. On the other hand, our work-stealing strategy allows moving tasks among nodes in order to keep all resources busy as much as possible. In this work, we compare our proposed load balancing approach to static scheduling as it is arguably the conventional strategy to distribute RTM shots among the computing nodes in distributed systems.

In summary, this work distinguishes itself from the others as it proposes a decentralized work-stealing method to balance the load of the RTM in distributed systems. It employs MPI one-sided communication to reduce its overhead by communicating asynchronously and without the victim's involvement. By keeping global load information, our method lowers the cost of victim selection and process termination. Consequently, it reduces the number of failed stealing attempts.

Table 4.3 presents a summary of the works related to load balancing mentioned above, highlighting their main characteristics in comparison to the method proposed in this work.

Table 4.3: Literature review on load balancing methods. The proposed work-stealing method bene ts from MPI one-sided communication to further reduce communication overhead and is applied to RTM in distributed memory systems.

	distributed memory	decen-tralized	work-stealing	asynchronous commu-nication	one-sided commu-nication	PGAS	MPI RMA	WS with global load information	RTM
Barros et al. (2018)									
Andreolli et al. (2014)									
Andreolli et al. (2015)									
Sena et al. (2011)									x
Hofmeyr et al. (2011)		x							
Tchiboukdjian et al. (2011)			x						
Imam & Sarkar (2015)		x	x					x	
Khaitan et al. (2013)	x								
Tesser et al. (2014)	x								
Keller Tesser et al. (2017)	x								
Tesser et al. (2018)	x								
Padoin et al. (2014)	x								
Padoin et al. (2017)	x								
Sharma & Kanungo (2014)	x	x						x	
Zheng et al. (2011)	x	x				x			
Martinez et al. (2016)	x	x	x	x					
Khaitan & Mccalley (2014)	x	x	x	x					
Mor & Maillard (2011)	x	x	x	x					
Li et al. (2013)	x	x	x	x	x	x			
Kumar et al. (2016)	x	x	x	x	x	x			
Dinan et al. (2009)	x	x	x	x	x	x			
Vishnu & Agarwal (2015)	x	x	x	x	x		x		
Our proposal	x	x	x	x	x		x	x	x

4.6 Conclusions

We have presented a decentralized work-stealing strategy with asynchronous communication to balance the load of a 3D reverse time migration for distributed computing systems. Each process communicates in a round-robin fashion to maintain a close approximation of the remaining tasks list. This list is used to lead the stealing when processes are idle. This strategy decentralizes the dynamic load balancing and avoids the overhead of centralized decisions. A token avoids deadlocks by ensuring that two processes cannot steal each other at the same time. The MPI one-sided communication prevents race conditions by serializing access to a memory space by multiple processes. By using MPI one-sided communication, the stealing is seamless to the victim processes since they do not stop processing their tasks during the stealing, avoiding unnecessary communication.

In the presented experiments, the 3D RTM had up to 42% of average idle time when ran over 64 nodes. This imbalance might be significantly reduced should the proposed work-stealing be applied. For the set of experiments performed in this paper, the proposed method has reduced the total execution time of the 3D RTM in up to 14% and its load imbalance in the order of 78% when compared to the conventional static distribution.

Further investigation is necessary to assess whether additional improvement can be achieved by adjusting the frequency of checking the token, the number of shots to be stolen, the method used to update the list of remaining tasks, and the technique to avoid deadlocks. Also, future work should focus on different aspects of a distributed system that may influence the load imbalance, such as the use of fault tolerance protocols (e.g., Shang (2018b) and Shang (2018a)) and heterogeneous computational systems. A comparison of our method against other load balancing methods is left to future work.

Chapter 5

Conclusions

This doctoral thesis introduced two methods to balance the workload of the reverse time migration algorithm. The cyclic token-based work-stealing (CTWS) method is designed for distributed memory level. By using the cyclic token approach, it keeps global information about the distribution of the tasks. This information simpli es the victim selection for work-stealing and the termination strategy. Moreover, by using MPI one-sided communication, it reduces de communication overhead. Experiments show that CTWS speeds up a 3D reverse time migration (RTM) for a number of nodes higher than 16. CTWS also signi cantly reduces the nodes' idle time.

The other load-balancing method proposed in this work is an auto-tuning for parallel loops of a 3D RTM in shared memory systems. The method is based on the coupled simulated annealing (CSA), which automatically nds the size of the blocks of work to be dynamically scheduled among the threads. In our experiments, the proposed auto-tuning method obtained reduced execution times when compared to two standard OpenMP schedulers for different input sizes and in different machines.

Besides these two methods, this thesis introduced the coupled multi-scale waveform inversion (CMFWI). This method employs possibly different initial models to the inversion of each frequency scale of a multi-scale waveform inversion (MFWI). By doing so, the frequency scales can be processed in parallel, thus increasing its scalability. An implementation o CMFWI using the coupled local minimizers (CLM) method was presented.

Tests applied to a 3D RTM were performed using a single model as the initial model for the inversion of each frequency scale. L2-norm results showed that, under these circumstances, CMFWI provides less accurate models when compared to MFWI. These results encourage future research on comparing models with different frequency content and using optimizations methods other than CLM.

Bibliography

Abdelkhalek, Rached, Henri Calandra, Olivier Coulaud, Guillaume Latu & Jean Roman (2012), Fast seismic modeling and reverse time migration on a graphics processing unit cluster,em`Concurrency Computation Practice and Experience'.

Adhianto, L., S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey & N. R. Tallent (2010), `HPCTOOLKIT: Tools for performance analysis of optimized parallel programs',Concurrency Computation Practice and Experience.

Akanksha, S. Kansara & Gardas Naresh Kumar (2017), Parallelization of reverse time migration using MPI + OpenMP,em`Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016'.

Alterman, Z & FC Karal (1968), `Propagation of elastic waves in layered media by finite difference methods',Bulletin of the Seismological Society of America58, 367–398.
URL: http://www.bssaonline.org/content/58/1/367.short

Andreolli, C., P. Thierry, L. Borges, C. Yount & G. Skinner (2014), Genetic Algorithm Based Auto-Tuning of Seismic Applications on Multi and Manycore Computers,em`EAGE Workshop on High Performance Computing for Upstream'.

Andreolli, Cedric, Philippe Thierry, Leonardo Borges, Gregg Skinner & Chuck Yount (2015), Characterization and Optimization Methodology Applied to Stencil Computations,emJ.Jeffers & J.Reinders, eds., `High Performance Parallelism Pearls', Elsevier, Boston, capítulo 23, pp. 377–396.
URL: http://www.sciencedirect.com/science/article/pii/B9780128021187000236

Araya-Polo, Mauricio, Félix Rubio, Raúl De La Cruz, Mauricio Hanzich, José María Cela & Daniele Paolo Scarpazza (2009), `3D seismic imaging through reverse-time migration on homogeneous and heterogeneous multi-core processors', Scientific Programming.

Assis, Italo A S, Antonio D S Oliveira, Tiago Barros, Idalmis M Sardina, Calebe P Bianchini & Samuel Xavier De-Souza (2019), `Distributed-Memory Load Balancing With Cyclic Token-Based Work-Stealing Applied to Reverse Time Migration', IEEE Access 7, 128419–128430.
URL: https://ieeexplore.ieee.org/document/8822671/

Barros, T., J. B. Fernandes, I. A. Souza-de Assis & S. Xavier-deSouza (2018), Auto-Tuning of 3D Acoustic Wave Propagation in Shared Memory Environments, `First EAGE Workshop on High Performance Computing for Upstream in Latin America', EarthDoc, Santander.
URL: http://www.earthdoc.org/publication/publicationdetails/?publication=94579

Baysal, Edip, Dan Kosloff & John W. C. Sherwood (1983), `Reverse-time migration', Geophysics 48, 1514–1524.

Blanch, Joakim O, Johan O A Robertsson & William W Symes (1995), `Modeling of a constant Q: Methodology and algorithm for an ef cient and optimally inexpensive viscoelastic technique', GEOPHYSICS 60(1), 176–184.
URL: https://doi.org/10.1190/1.1443744

Blumofe, Robert D. & Charles E. Leiserson (1999), `Scheduling multithreaded computations by work stealing', J. ACM 46(5), 720–748.
URL: http://doi.acm.org/10.1145/324133.324234

Bunks, Carey, Fatimetou M. Saleck, S. Zaleski & G. Chavent (1995), `Multiscale seismic waveform inversion', Geophysics 60(5), 1457–1473.
URL: http://library.seg.org/doi/abs/10.1190/1.1443880

Carcione, José M., Gérard C. Herman & A. P. E. ten Kroode (2002), `Seismic modeling', Geophysics.

Cerjan, Charles, Dan Kosloff, Ronnie Kosloff & Moshe Reshef (1985), `A nonre ecting boundary condition for discrete acoustic and elastic wave equations'.

Chen, Zhongying, Dongsheng Cheng, Wei Feng & Tingting Wu (2013), `An optimal 9-point nite difference scheme for the Helmholtz equation with PML', International Journal of Numerical Analysis and Modeling.

Chu, Chunlei & Paul L. Stoffa (2008), A pseudospectral– nite difference hybrid approach for large–scale seismic modeling and RTM on parallel computers, in `SEG Technical Program Expanded Abstracts 2008'.

Chu, Chunlei, Paul L. Stoffa & Roustam Seif (2009), 3D seismic modeling and reverse–time migration with the parallel Fourier method using non–blocking collective communications, in `SEG Technical Program Expanded Abstracts 2009'.

Claerbout, Jon F & Stephen M Doherty (1972), `Downward continuation of moveout-corrected seismograms'.

Clarke, Lyndon, Ian Glendinning & Rolf Hempel (1994), The MPI message passing interface standard, in K. M.Decker & R. M.Rehmann, eds., `Programming Environments for Massively Parallel Distributed Systems', Birkhäuser Basel, Basel, pp. 213–218.

Da Silva, Curt & Felix Herrmann (2016), A uni ed 2D/3D software environment for large scale time-harmonic full waveform inversion, in `SEG Technical Program Expanded Abstracts', SEG.

Dagum, L. & R. Menon (1998), `OpenMP: an industry standard API for shared-memory programming',IEEE Computational Science and Engineering 5(1), 46–55.

De Hoop, AT (1960), `A modi cation of Cagniard's method for solving seismic pulse problems',Applied Scienti c Research, Section B 8(1), 349–356.

Diaz, Javier, Camelia Muñoz-Caro & Alfonso Niño (2012), `A survey of parallel programming models and tools in the multi and many-core era',IEEE Transactions on Parallel and Distributed Systems

Dinan, James, D. Brian Larkins, P. Sadayappan, Sriram Krishnamoorthy & Jarek Nieplocha (2009), Scalable work stealing, in `Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis', SC '09, ACM, New York, NY, USA, pp. 53:1–53:11.
URL: http://doi.acm.org/10.1145/1654059.1654113

dos Santos, Adriano Wagner Gomes (2013), Inversão de Forma de Onda Aplicada à Análise de Velocidades Sísmicas Utilizando um Abordagem Multiescala, Dissertação de mestrado, Universidade Federal da Bahia.
URL: http://www.pggeo sica.ufba.br/publicacoes/detalhe/284

Essa , Hassane & Patrick Hède (2017), Framstim: Framework for large scale multimedia content feature extraction based on MPI one-sided communication, in `Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing', ICC '17, ACM, New York, NY, USA, pp. 41:1–41:6.
URL: http://doi.acm.org/10.1145/3018896.3018936

Fu, H., M. Gorentla Venkata, S. Salman, N. Imam & W. Yu (2018), SHMEMGraph: Ef cient and balanced graph processing using one-sided communication, in `2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)', pp. 513–522.

Ghosh, S., J. R. Hammond, A. J. Peña, P. Balaji, A. H. Gebremedhin & B. Chapman (2016), One-sided interface for matrix operations using MPI-3 RMA: A case study with elemental, in `2016 45th International Conference on Parallel Processing (ICPP)', pp. 185–194.

Gonçalves-e Silva, Kayo, Daniel Aloise & Samuel Xavier-de Souza (2018), `Parallel synchronous and asynchronous coupled simulated annealing', The Journal of Supercomputing 74(6), 2841–2869.

Gonçalves-e Silva, Kayo & Samuel Xavier-de Souza (2018), `Perpetual orbit coupled simulated annealing for continuous optimization', arXiv preprint arXiv:1803.01059 .

Griewank, Andreas & Andrea Walther (2000), `Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation', ACM Transactions on Mathematical Software .

Gropp, W., E. Lusk & A. Skjellum (1999), Using MPI, 2nd Edition: Portable Parallel Programming with the Message Passing Interface, MIT Press, Cambridge, MA, USA.

HEMON, CH. (1978), `Equations d'onde et modeles ', Geophysical Prospecting 26(4), 790–821.
URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2478.1978.tb01634.x

Hofmeyr, Steven, Juan A. Colmenares, Costin Iancu & John Kubiatowicz (2011), Juggle: Proactive Load Balancing on Multicore Computers, em `Proceedings of the 20th international symposium on High performance distributed computing - HPDC '11'.

Imam, Shams & Vivek Sarkar (2015), Load balancing prioritized tasks via work-stealing, em J. L.Träff, S.Hunold & F.Versaci, eds., `Euro-Par 2015: Parallel Processing', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 222–234.

Katagiri, Takahiro, Kenji Kise, Hiroaki Honda & Toshitsugu Yuba (2003), Fiber: A generalized framework for auto-tuning software, em A.Veidenbaum, K.Joe, H.Amano & H.Aiso, eds., `High Performance Computing', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 146–159.

Katagiri, Takahiro, Satoshi Ohshima & Masaharu Matsumoto (2014), Auto-tuning of computation kernels from an FDM code with ppOpen-AT, em `Proceedings - 2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs, MC-SoC 2014'.

Katagiri, Takahiro, Satoshi Ohshima & Masaharu Matsumoto (2015), Directive-Based Auto-Tuning for the Finite Difference Method on the Xeon Phi, em `Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2015'.

Kearey, Philip, Michal Brooks & Ian Hill (2002), An introduction to geophysical exploration, 3rd ed.ª edição, Malden, MA : Blackwell Science.

Keller Tesser, Rafael, Lucas Mello Schnorr, Arnaud Legrand, Fabrice Dupros & Philippe Olivier Alexandre Navaux (2017), Using Simulation to Evaluate and Tune the Performance of Dynamic Load Balancing of an Over-Decomposed Geophysics Application, em F. F.Rivera, T. F.Pena & J. C.Cabaleiro, eds., `Euro-Par 2017: Parallel Processing', Springer International Publishing, Cham, pp. 192–205.

Kelly, K R, R W Ward, Treitel & R M Alford (1976), `Synthetic Seismograms: A Finite-Difference Aproach', Geophysics 41, 2–27.

Khaitan, Siddhartha Kumar, James D. McCalley & Arun Somani (2013), `Proactive task scheduling and stealing in master-slave based load balancing for parallel contingency analysis', Electric Power Systems Research

Khaitan, S.K. & James Mccalley (2014), `Scale: A hybrid MPI and multithreading based work stealing approach for massive contingency analysis in power systems', Electric Power Systems Research 114, 118–125.

Kirkpatrick, Scott, C Daniel Gelatt & Mario P Vecchi (1983), `Optimization by simulated annealing', Science 220(4598), 671–680.

Kosloff, Dan D & Edip Baysal (1983), `Migration with the full acoustic wave equation', Geophysics 48(6), 677–687.

Kumar, V., K. Murthy, V. Sarkar & Y. Zheng (2016), Optimized distributed work-stealing, em `2016 6th Workshop on Irregular Applications: Architecture and Algorithms (IA3)', pp. 74–77.

Lange, M, N Kukreja, M Louboutin, F Luporini, F Vieira, V Pandolfo, P Velesko, P Kazakas & G Gorman (2016), Devito: Towards a Generic Finite Difference DSL Using Symbolic Python, em `2016 6th Workshop on Python for High-Performance and Scienti c Computing (PyHPC)', pp. 67–75.

Lazzaro, Al o, Joost VandeVondele, Jürg Hutter & Ole Schütt (2017), Increasing the ef ciency of sparse matrix-matrix multiplication with a 2.5D algorithm and one-sided MPI, em `Proceedings of the Platform for Advanced Scienti c Computing Conference', PASC '17, ACM, New York, NY, USA, pp. 3:1–3:9.
URL: http://doi.acm.org/10.1145/3093172.3093228

Li, Shigang, Jingyuan Hu, Xin Cheng & Chongchong Zhao (2013), Asynchronous work stealing on distributed memory systems, in `Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013'.

Li, Yuan-Yuan, Zhen-Chun Li, Kai Zhang & Xuan Zhang (2015), `Frequency-domain elastic full-waveform multiscale inversion method based on dual-level parallelism', Applied Geophysics 12(4), 545–554.
URL: https://doi.org/10.1007/s11770-015-0519-8

Lu, Ligang & Karen Magerlein (2013), Multi-level Parallel Computing of Reverse Time Migration for Seismic Imaging on Blue Gene/Q, in `Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming', PPoPP '13, ACM, New York, NY, USA, pp. 291–292.
URL: http://doi.acm.org/10.1145/2442516.2442550

Martinez, Victor, David Michea, Fabrice Dupros, Olivier Aumage, Samuel Thibault, Hideo Aochi & Philippe O.A. Navaux (2016), Towards seismic wave modeling on heterogeneous many-core architectures using task-based runtime system, in `Proceedings - Symposium on Computer Architecture and High Performance Computing'.

McMechan, G. A. (1983), `Migration by extrapolation of time-dependent boundary values', Geophysical Prospecting 31(3), 413–420.
URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2478.1983.tb01060.x

Mor, Stefano & Nicolas Maillard (2011), `Dynamic workload balancing deques for branch and bound algorithms in the message passing interface', International Journal of High Performance Systems Architecture

Nocedal, Jorge & Stephen J. Wright (2006), *Numerical Optimization*, second edição, Springer, New York, NY, USA.

Nunes-do Rosario, Desnes A., Samuel Xavier-de Souza, Rosangela C. Maciel & Jesse C. Costa (2015), `Parallel Scalability of a Fine-Grain Prestack Reverse Time Migration Algorithm', IEEE Geoscience and Remote Sensing Letters **12**(11), 2433–2437. URL: http://ieeexplore.ieee.org/document/7307125/

Pacheco, Peter (2011), *An Introduction to Parallel Programming*, Morgan Kaufmann, Burlington.

Padoin, E. L., M. Castro, L. L. Pilla, P. O. A. Navaux & J. Méhaut (2014), Saving energy by exploiting residual imbalances on iterative applications, em `2014 21st International Conference on High Performance Computing (HiPC)', pp. 1–10.

Padoin, Edson L., Laércio L. Pilla, Márcio Castro, Philippe O. A. Navaux & Jean-François Méhaut (2017), Exploration of load balancing thresholds to save energy on iterative applications, em C. J. Barrios Hernández, I. Gitler & J. Klapp, eds., `High Performance Computing', Springer International Publishing, Cham, pp. 76–88.

Park, Mi Young & Sang Hwa Chung (2009), Detecting race conditions in one-sided communication of MPI programs, em `Proceedings of the 2009 8th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2009'.

Paul, Sri Raj, Mauricio Araya-Polo, John Mellor-Crummey & Detlef Hohl (2016), Performance analysis and optimization of a hybrid seismic imaging application, em `Procedia Computer Science'.

Perrone, Michael, Lurng Kuo Liu, Ligang Lu, Karen Magerlein, Changhoan Kim, Irina Fedulova & Artyom Semenikhin (2012), Reducing data movement costs: Scalable seismic imaging on blue gene, em `Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium, IPDPS 2012'.

Plessix, Rene Edouard (2006), `A review of the adjoint-state method for computing the gradient of a functional with geophysical applications', Geophysical Journal International 167(2), 495–503.

Qawasmeh, Ahmad, Maxime R. Hugues, Henri Calandra & Barbara M. Chapman (2017), `Performance portability in reverse time migration and seismic modelling via OpenACC', International Journal of High Performance Computing Applications

Robertsson, Johan O A, Joakim O Blanch & William W Symes (1994), `Viscoelastic finite–difference modeling', GEOPHYSICS 59(9), 1444–1456.
URL: https://doi.org/10.1190/1.1443701

Sena, A. C., A. P. Nascimento, C. Boeres, V. Rebello & A. Bulcao (2011), An approach to optimise the execution of RTM algorithm in multicore machines, in `2011 IEEE Seventh International Conference on eScience', pp. 403–410.

Shang, Yilun (2018), `Resilient consensus of switched multi-agent systems', Systems and Control Letters

Shang, Yilun (2018), `Resilient Multiscale Coordination Control against Adversarial Nodes', Energies.

Sharma, R. & P. Kanungo (2014), Dynamic load balancing algorithm for heterogeneous multi-core processors clusters, in `2014 Fourth International Conference on Communication Systems and Network Technologies', pp. 288–292.

Suykens, Johan A. K., Joos Vandewalle & Bart De Moor (2001), `Intelligence and Cooperative Search by Coupled Local Minimizers', International Journal of Bifurcation and Chaos 11(08), 2133–2144.
URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-0035537546&partnerID=tZOtx3y1

Symes, W. (2007), `Reverse time migration with optimal checkpointing', Geophysics

Szu, Harold & Ralph Hartley (1987), `Fast simulated annealing', Physics Letters A

Tarantola, Albert (1984), `Inversion of seismic re ection data in the acoustic approximation', GEOPHYSICS 49(8), 1259–1266.
URL: http://library.seg.org/doi/abs/10.1190/1.1441754

Tchiboukdjian, Marc, Vincent Danjean, Thierry Gautier, Fabien Le Mentec & Bruno Raffin (2011), A work stealing scheduler for parallel loops on shared cache multicores, em`Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)'.

Tesser, Rafael Keller, Laércio Lima Pilla, Fabrice Dupros, Philippe Olivier Alexandre Navaux, Jean-François Méhaut & Celso Mendes (2014), Improving the performance of seismic wave simulations with dynamic load balancing, in`Proceedings of the 2014 22Nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing', PDP '14, IEEE Computer Society, Washington, DC, USA, pp. 196–203.
URL: https://doi.org/10.1109/PDP.2014.37

Tesser, Rafael Keller, Lucas Mello Schnorr, Arnaud Legrand, Franz Christian Heinrich, Fabrice Dupros & Philippe O.A. Navaux (2018), `Performance modeling of a geophysics application to accelerate over-decomposition parameter tuning through simulation'.

Teughels, A, G De Roeck & J A K Suykens (2004), Coupled Local Minimizers: a new global optimization method, in L.Maes, MA and Huyse, ed., `Realiability and Optimization of Structural Systems', IFIP, WG7 5, A A Balkema Publishers, pp. 205–212.

Thorbecke, J.˜W. & D Draganov (2011), 'Finite-difference modeling experiments for seismic interferometry', *Geophysics* **76**, H1.

Virieux, J. & S. Operto (2009), 'An overview of full-waveform inversion in exploration geophysics', *GEOPHYSICS* **74**(6), WCC1–WCC26.
URL: *http://library.seg.org/doi/10.1190/1.3238367*

Vishnu, Abhinav & Khushbu Agarwal (2015), Large scale frequent pattern mining using MPI one-sided model, *em* 'Proceedings - IEEE International Conference on Cluster Computing, ICCC'.

Whitmore, N.D. (1983), 'Iterative depth migration by backward time propagation', *SEG Technical Program Expanded Abstracts* pp. 382–385.

Xavier-de Souza, S, J A K Suykens, J Vandewalle & D Bolle (2010), 'Coupled Simulated Annealing', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **40**(2), 320–335.

Yilmaz, Öz (2001*a*), *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*, número v. 2 *em* 'Investigations in geophysics', Society of Exploration Geophysicists.

Yilmaz, Öz (2001*b*), *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*, número v. 1 *em* 'Investigations in geophysics', Society of Exploration Geophysicists.

Zhang, Jin-Hai, Shu-Qin Wang & Zhen-Xing Yao (2009), 'Accelerating 3D Fourier migration with graphics processing units', *GEOPHYSICS* .

Zheng, Gengbin, Abhinav Bhatelé, Esteban Meneses & Laxmikant V. Kalé (2011), 'Periodic hierarchical load balancing for large supercomputers', *International Journal of High Performance Computing Applications* .

Appendix A

2D visco-acoustic modelling using Devito

Although acoustic modeling has been broadly used for seismic processing, it cannot represent the natural dispersion and attenuation of the materials present in the subsurface. Ignoring those effects can lead to erroneous outcomes. A visco-acoustic approach models attenuation and dispersion by adding the well-known memory variables (Robertsson et al. 1994) to simulate a series of standard linear solids (SLS's). Visco-acoustic modeling also can implement absorbing boundaries efficiently by merely changing the attenuation parameters near the boundaries.

While visco-acoustic modeling has been used for decades, only a few implementations are publicly available. Thorbecke & Draganov (2011) has published one of the most known of them. Since it is implemented in the C programming language, it can be difficult for the broad audience to change its specifications as, for instance, the wave equations and the order and scheme of the derivatives.

We present a 2D visco-acoustic modeling software implemented in Devito (Lange et al. 2016). Devito is based on Domain-Specific Languages that allow the user to focus on geophysics aspects of the code by generating a finite differences low-level code automatically from the specified equations. By using Devito, we hope to provide a tool suitable for beginners in geophysics.

A.1 2D Visco-acoustic modelling

A Standard Linear Solid (SLS) is a combination of springs and a dashpot commonly used to model attenuation and dispersion properties. The attenuation in a material is represented by the quality factor Q. The smaller the Q, the more the wave is attenuated. An array of SLS's can be used in order to represent a given Q. The bigger the number of SLSs, the higher the modeling accuracy and the computational cost.

Each SLS is characterized by two parameters, namely, the stress and strain relaxation times. One of the most popular ways to determine the stress and strain relaxation times from Q is the τ-method (Blanch et al. 1995).

Based on Robertsson et al. (1994), a 2D visco-acoustic model can be mathematically defined as

$$\frac{\partial V_x}{\partial t} = \frac{1}{\rho}\frac{\partial P}{\partial x}, \tag{A.1}$$

$$\frac{\partial V_z}{\partial t} = \frac{1}{\rho}\frac{\partial P}{\partial z}, \tag{A.2}$$

$$\frac{\partial P}{\partial t} = \rho c^2 \left(1 + \sum_{l=1}^{L} \left(1 + \frac{\tau_{el}}{\tau_{sl}}\right)\right)\left(\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\right) + \sum_{l=1}^{L} r_{pl} \tag{A.3}$$

and

$$\frac{\partial r_{pl}}{\partial t} = \frac{1}{\tau_{sl}} r_{pl} + \frac{\tau_{el}}{\tau_{sl}}\left(1 + \frac{1}{k}\right)\left(\frac{\partial V_x}{\partial x} + \frac{\partial V_z}{\partial z}\right) \quad (1 \le l \le L) \tag{A.4}$$

where V_x and V_z are the particle velocity components in x and z directions, t is the temporal dimension, ρ is the medium density, P is the acoustic pressure, c is the velocity model, τ_e is the array of strain relaxation times, τ_s is the array of stress relaxation times, r_p are the memory variables, Q is the quality factor, and L is the number of SLS's.

We implemented Equations (A.1) to (A.4) in Devito using its symbolic representation. This function can be called by

$$p; rec = viscAcousMod(L; \mathbf{m}; \mathbf{ts}; \mathbf{te}; f_0; \mathbf{tr}; \mathbf{src})$$

where the input parameters are the number of SLS's L, the velocity model $\mathbf{m} \in \mathbb{R}^{n_x \; n_z}$ with n_x and n_z being the number of points in x and z dimensions respectively, the array of stress relaxation times $\mathbf{ts} \in \mathbb{R}^{n_x \; n_z \; L}$, the array of strain relaxation times $\mathbf{te} \in \mathbb{R}^{n_x \; n_z \; L}$, the peak frequency f_0 in kHz, the time samples $\mathbf{tr} \in \mathbb{R}^{n_s}$ with n_s being the number of time samples and the source $\mathbf{src} \in \mathbb{R}^{n_s}$. The outputs p and rec are the wave-field and a seismogram respectively. Algorithm 6 presents a portion of the implemented program.

Algoritmo 6: Portion of *viscAcousMod* function. It implements visco-acoustic modeling equations using Devito.

```
def viscAcousMod(L,m,ts,te,f0,tr,src):
    [...]
    rcp2 = m.rho*(1/m.m)
    pde_vx = m.rho*vx.dt + dpxdx
    pde_vz = m.rho*vz.dt + dpzdz
    pde_p = p.dt + rcp2*(1-np.sum(1-te/ts))*(dvxdx+dvzdz)
           + np.sum(rp)
    for l in range(0,L):
        pde_rp[l] = rp[l].dt + (1/ts[l])*rp[l]
            - rcp2*(1/ts[l])*(1-te[l]/ts[l])*(dvxdx+dvzdz)
    u_vx = Eq(vx.forward,
            (1-m.damp)*solve(pde_vx, vx.forward))
    u_vz = Eq(vz.forward,
            (1-m.damp)*solve(pde_vz, vz.forward))
    u_p = Eq(p.forward, (1-m.damp)*solve(pde_p, p.forward))
    for l in range(0,L):
        u_rp[l] = Eq(rp[l].forward,
                (1-m.damp)*solve(pde_rp[l], rp[l].forward))
    [...]
```

According to Lange et al. (2016), Devito automatically generates parallel code. Among the optimizations implemented by the Devito are shared-memory parallelism via OpenMP, vectorization, and loop blocking.

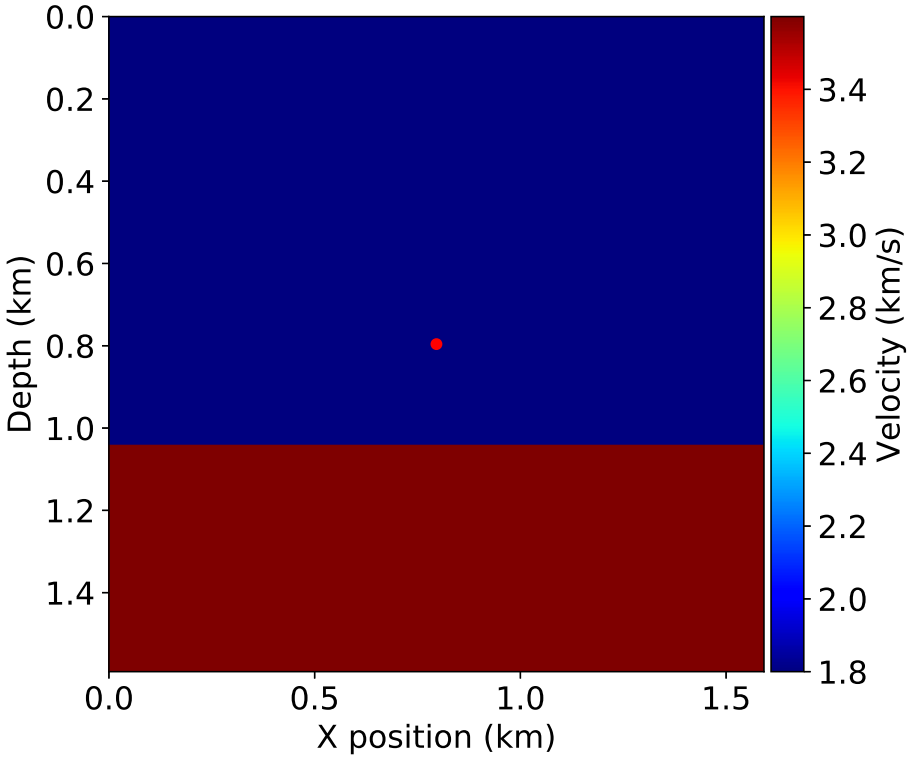


Figure A.1: Input model for the first experiment. The red dot represents the source position.

A.2 Results and Discussion

In this section, we show the effects of using visco-acoustic modeling in comparison to acoustic modeling. The model is shown in Figure A.1 is the input model for our first experiment. It is represented by a 399 399 grid with a spatial sampling of 4 m in both directions. Its velocity is v = 1800 m=s in the top layer and v = 3600 m=s in the bottom layer. A Ricker source with a peak frequency of 50 Hz is located at the center of the model.

Figure A.2 presents wave-field snapshots at 230 ms for the acoustic and visco-acoustic modelling using two different Q models. Both visco-acoustic wavefields present attenuated amplitudes and velocity dispersion compared to the acoustic result. However, amplitude loss and dispersion are less intense in the bottom layer of Figure A.2c.

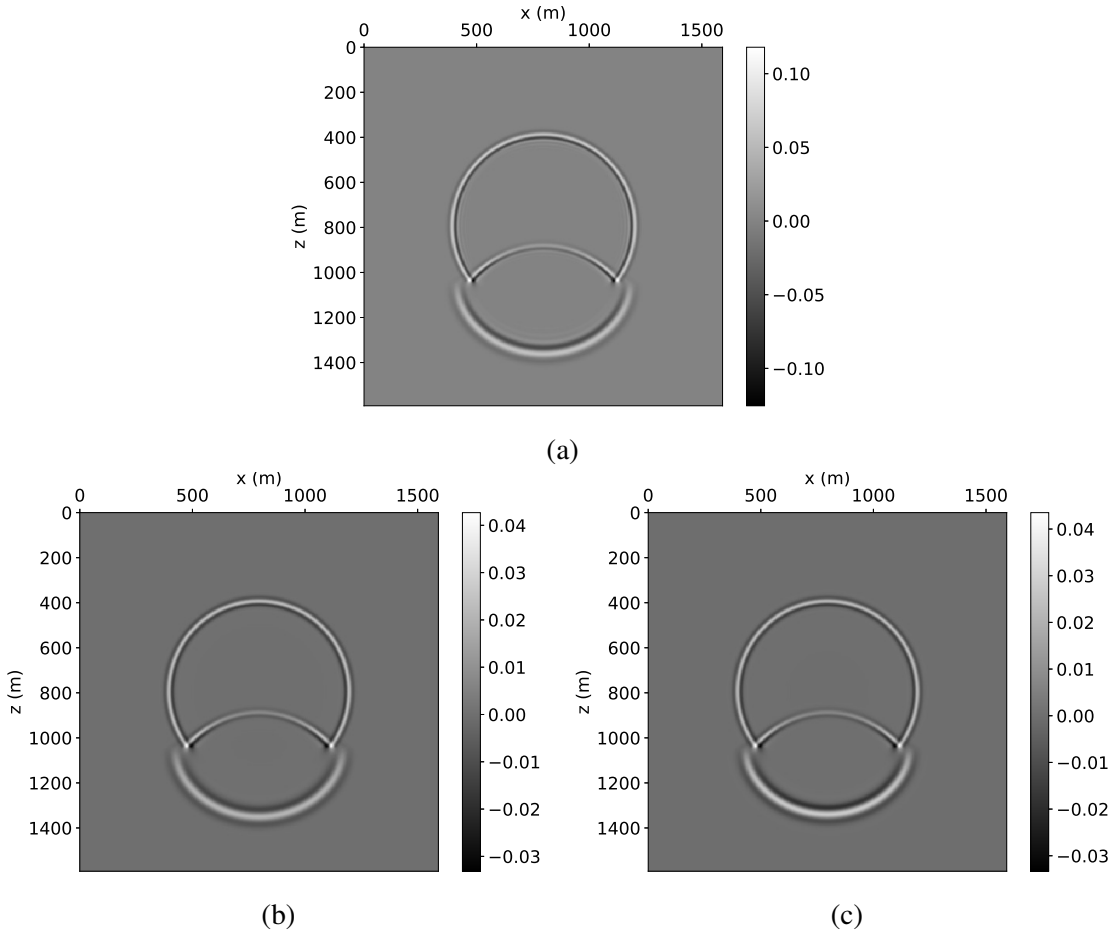


Figure A.2: Wavefields snapshots at 230 ms modelled using A.2a an acoustic approach and A.2b A.2c a visco-acoustic approach with 3 relaxation mechanisms. Q is constant equal to 30 in A.2b. For A.2c, $Q = 30$ for the top layer and 100 to the bottom layer.

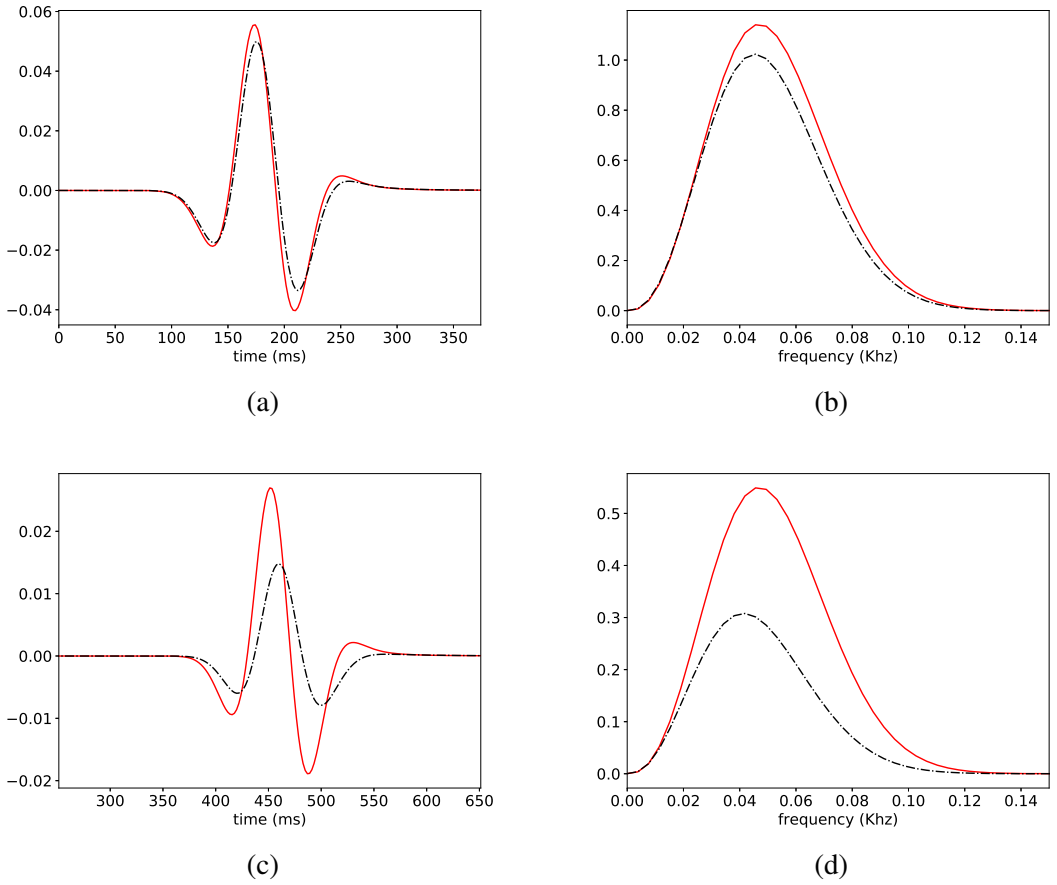


Figure A.3: Traces for receivers at (a-b) (500,200) m and (c-d) (500,800) m in (left) time and (right) frequency domains. Solid lines represent traces obtained by acoustic modelling and dashed lines display traces registered in a visco-acoustic modelling using 3 relaxation mechanisms and $Q = 20$.

In order to illustrate how the use of visco-acoustic modeling changes the frequency content, we plot in Figure A.3 the frequency spectra for two receivers: one placed near the top of the model and the other one near the bottom. The input model is a 1000 1000 m grid with a spatial sampling of 10 m, the velocity is 2000 m/s, and the source is a Ricker wavelet with a peak frequency of 10 Hz placed at $(x; y) = (500; 20)\ m$.

Figure A.3 shows that attenuation causes amplitude loss both in time and frequency contents, and the farther the receiver, the higher the loss. The presence of attenuation also results in the displacement of the peak frequency.

A.3 Conclusions

This chapter presented a 2D visco-acoustic modeling implemented in Devito. Comparisons of seismic traces and wavefields show that, by disregarding attenuation effects, one may miss important information. Amplitudes and frequency content in acoustic modeling diverge from the ones desired.

Taking into consideration that visco-acoustic modeling produces a more accurate seismic data, its use can lead to a better seismic analysis.

As future work, the method proposed here can be extended to 3D, have its results compared to the analytical solution and be used to more sophisticated methods as, for instance, reverse time migration and full waveform inversion.