Computers and Geosciences 121 (2018) 39–52

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

Static load balancing using non-uniform mesh partitioning based on ray density prediction for the parallel wavefront construction method

Abdullah Fahad Alyabes[*,1], Richard L. Gibson Jr.[2]

Texas A&M University, Department of Geology and Geophysics, 3115 TAMU, College Station, TX 77843, USA

ARTICLE INFO

Keywords:
Wavefront construction method
Ray tracing
Load balancing
High performance computing
Standard template adaptive parallel library

ABSTRACT

The Wavefront Construction (WFC) method, which was developed based on ray theory, is one of the most efficient seismic modeling tools. WFC propagates a wavefront represented by rays in a computational mesh that is refined whenever an accuracy criterion is violated. Since WFC interpolates new rays during wave propagation, the wavefront mesh is considered highly adaptive. Recently, a parallel WFC was developed using the Standard Template Adaptive Parallel Library. However, due to wavefront density adaptivity, the parallel implementation exhibits inefficient performance owing to load imbalances between processors. In this paper we apply a static load balancing approach based on the prediction of future load for a synthetic salt dome model, to improve performance. This approach utilizes a preliminary conventional ray simulation to estimate the cost (future load) of each cell in the WFC's initial wavefront mesh. Then it applies a non-uniform mesh decomposition that results in a more efficient parallel WFC. Compared to the original implementation, our implementation shows better and more stable scalability in most WFC simulations conducted on the salt model. This paper contributes to understanding the behavior of wavefront mesh adaptability and predicting earth model complexities, and serves as a guide for achieving the ultimate goal, a fully load-balanced parallel WFC.

1. Introduction

Several geophysical applications in seismology, including seismic acquisition, imaging, and interpretation, utilize ray methods to represent subsurface seismic wave propagation (Červený, 2005; Gjøystdal et al., 2007, 2002; Alaei, 2012). The Wavefront Construction (WFC) method, an extension of conventional ray tracing, is one of the most efficient tools for seismic modeling (Carcione et al., 2002; Fehler and Huang, 2002). WFC methods have been implemented to simulate seismic wave propagation in stratified and gridded models for both isotropic and anisotropic media (Vinje et al., 1993, 1996; 1999; Gibson Jr, 2000; Lee, 2005; Gibson et al., 2005; Kaschwich, 2006; Chambers and Kendall, 2008; Chen, 2011). The main idea behind the WFC method is to propagate wavefronts represented by rays arranged in a triangular or quadrilateral computational mesh. The accuracy of the mesh is controlled by interpolating new rays whenever an accuracy criterion is violated, allowing WFC to efficiently track amplitudes and multiple arrival times.

The simulation of seismic waves in complex 3D structures has made high performance computing an essential tool for seismic modeling. In the past several years seismic modeling has utilized parallelization techniques involving both CPUs and GPUs (Bohlen, 2002; Grunberg et al., 2004; Komatitsch et al., 2010; Szostek and Leniak, 2012; Mohammadzaheri et al., 2013). Recently, the WFC method was implemented as a parallel application using the Standard Template Adaptive Parallel Library (STAPL) (Jain, 2011). STAPL is a C++ development framework that allows users to implement parallel applications with a high level of abstraction by hiding specific details of the parallel programming (Buss et al., 2010).

Parallel WFC (pWFC) performance varies based on the input parameters and the desired output precision and is therefore affected by several factors, including source locations, the complexity of the earth model, and the desired simulation accuracy. pWFC has been shown to lose a considerable degree of performance due to load imbalances (Jain, 2011). Load imbalances occur when there is a major difference in the work load distribution between CPUs, leading to longer turnaround times. pWFC load imbalancing arises mainly from the adaptivity of the wavefront mesh density during propagation.

pWFC was recently implemented by applying uniform domain decomposition among CPUs during the initialization phase. As a

* Corresponding author.
E-mail address: abdullah.alyabes@aramco.com (A.F. Alyabes).
[1] Abdullah Alyabes is the author who conducted the research project and wrote the paper. Present address: P.O. Box 13950, Dhahran 31311, Saudi Arabia
[2] Dr. Richard Gibson is the author who supervised the research project and reviewed the paper.

https://doi.org/10.1016/j.cageo.2018.09.002
Received 29 December 2017; Received in revised form 27 July 2018; Accepted 13 September 2018
Available online 19 September 2018
0098-3004/

consequence, the rays on the wavefront mesh are almost evenly distributed between processes during the first simulation steps. Typically, rays are interpolated as the wavefront travels away from the source. Since ray tracing is the most expensive part of pWFC, it is better to have a load imbalance during the initial steps rather than later. In this research, we applied static load balancing using an initial non-uniform distribution of rays based on predictions of wavefront mesh density. We first develop a method to estimate the future cost (load) of each cell in the wavefront mesh using a preliminary ray simulation, and then distribute ray tracing work to CPUs based on these load predictions. A synthetic Gulf of Mexico salt dome model was used to investigate the impact of load imbalances on pWFC performance. This research improved our understanding of the behavior of pWFC load balancing and guided us toward the best strategy to achieve a load-balanced pWFC application.

This paper describes our static load balancing approach based on mesh density predictions. First, we review the WFC and pWFC algorithms and their performance. Then we propose three different approaches for predicting costs. We thoroughly evaluate each approach by applying pWFC to the salt dome model. Finally, we compare the performance of our proposed non-uniform wavefront mesh decomposition method to the original pWFC implementation.

2. Background theory and method

2.1. Wavefront construction method

The WFC method, which was developed by applying a high-frequency approximation to the elastic wave equation (Červený, 2005), has been applied to isotropic and anisotropic media to simulate seismic wave propagation (Vinje et al., 1993, 1996; 1999; Gibson Jr, 2000; Lee, 2005; Gibson et al., 2005; Chambers and Kendall, 2008; Chen, 2011). The main idea behind the WFC method is to trace ray fields rather than individual rays. A computational mesh is used to represent a wavefront that connects adjacent rays at the same travel time. WFC begins as an initial sparse set of rays and interpolates new rays whenever an accuracy criterion is violated. By tracking the propagation of the wavefront WFC addresses the two-points problem, which is to find a direct connection between the source and the receiver. WFC can therefore efficiently track amplitudes and multiple arrival times (Vinje et al., 1996; Gibson et al., 2005). The results can be utilized in different geophysical applications, such as seismic imaging (Gajewski et al., 2002; Kaschwich, 2006). WFC can also be useful for educational purposes to visualize seismic waves in complex media (Gjøystdal et al., 2002).

The WFC algorithm used in this research consists of the following steps: (1) initialize ray directions (for the initial mesh), (2) trace individual rays, (3) construct a computational mesh that represents a wavefront, and (4) interpolate or coarsen the wavefront mesh as rays propagate through the earth model (Gibson et al., 2005; Lee and Gibson, 2007; Jain, 2011). The first step initializes a sparse set of rays from the source, for which there are two methods: the take-off angle method and the cubed sphere method. The latter method is more accurate and efficient (Lee, 2005; Lee and Gibson, 2007). This method initializes the direction of rays based on a focal cube surface surrounding the source. It then traces each ray using a predefined time step (the wavefront time step) based on asymptotic ray theory (Červený, 2005). Next, it constructs a quadrilateral mesh that connects adjacent rays at the same travel time, to represent the wavefront. Each intersection (corner) of adjacent mesh cells represents a point along the ray at a constant time, and each cell represents a ray tube in the next propagation step. Rays typically diverge as the wavefront travels away from the source, which leads to inaccurate estimations of travel time and amplitude for points arbitrarily located within the cells (receivers) (Lee and Gibson, 2007). Therefore, WFC interpolates new rays as needed to maintain travel time accuracy (Fig. 1). New rays are added using a specific threshold that defines the desired accuracy of the results

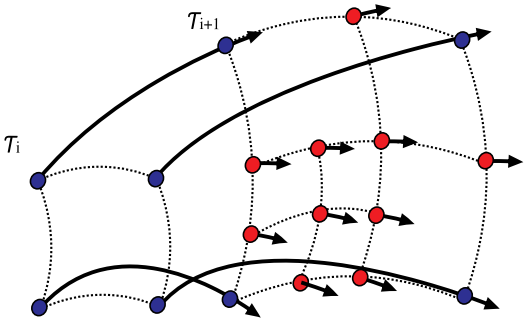


Fig. 1. Schematic illustration of ray interpolations for one cell in the wavefront mesh from an arbitrary time τ_i to the next time step τ_{i+1}. Arrows represent rays, dotted lines represent the mesh geometry, blue circles indicate old rays, and red circles indicate new interpolated rays. The example shows more interpolated rays in the lower left part of the mesh at τ_{i+1} that have been added due to a lower degree of accuracy. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

based on paraxial time prediction (Gibson et al., 2005). Since ray tracing is the most expensive part of the algorithm, WFC removes rays whenever the ray field reaches a sufficient density. Therefore, the distribution of the rays in each wavefront is uneven, with local density based on the ray field's local behavior. During wave propagation, when a ray tube passes through a receiver on the surface, its travel time and amplitude are recorded. Numerical implementation details for WFC are provided in Appendix A.

2.2. Parallel wavefront construction method

Geophysical applications have taken great advantage of high performance computing. Researchers have implemented different seismic modeling methods using a variety of parallel libraries and frameworks, such as the finite difference method using MPI (Bohlen, 2002), seismic ray tracing in adaptive mesh models using MPI (Grunberg et al., 2004), the high-order finite element technique using CUDA (Komatitsch et al., 2010), the WFC method based on Lomax's waveray approximation using C# threading (Szostek and Leniak, 2012), and distributed ray tracing using a map-reduce technique (Mohammadzaheri et al., 2013). Even though WFC is considered one of the fastest methods in seismic modeling, it is still computationally expensive to capture sufficient information about seismic waves in complex 3D media. Therefore, the parallel WFC method (pWFC) was developed using the Standard Template Adaptive Parallel Library (STAPL) (Jain, 2011).

STAPL is a C++ development framework that allows users to implement parallel applications with a high level of abstraction by hiding specific parallel programming details (An et al., 2003; Buss et al., 2010). STAPL was developed to address parallel programming difficulties and to support portable parallel performance on both shared and distributed memories. STAPL was designed to play a role in parallel development similar to the role of the C++ Standard Template Library (STL) for serial program development. STL is a collection of basic algorithms (e.g., find, merge, copy, sort), generic data structures called containers (e.g., lists, sets, vectors, maps), and iterators to facilitate access to container data (Musser et al., 2001). Similar to STL, STAPL provides distributed data structures (pContainers), parallel algorithms (pAlorithms), task-dependent graphs (pRange), and pViews to facilitate data access in pContainers. More advanced users can build their own STAPL algorithms and containers and have access to runtime system components such as STAPL thread scheduling, memory management, and synchronization.

The pWFC utilizes two pContainers: pMap for ray collection and pGraph for wavefronts, two pViews: pMap_pView and pGraph_pView, and two pAlorithms: map and map reduce (Jain, 2011). The pWFC implementation (Algorithm 1) parallelizes the most expensive

operations. It begins by partitioning the initial wavefront mesh into equal partitions for multiple processors, where rays located on the mesh partition boundaries are shared between processors (initialization phase; Fig. 2a). Then, for every wavefront time step, each processor traces the rays belonging to its own partial mesh, and a partial wavefront is constructed for interpolating or coarsening rays (propagation phase). After the simulation, pWFC records multiple arrivals for each receiver on the surface (surface mapping phase).

Algorithm 1 Parallel WFC Algorithm

Input: Earth model description, wavefront mesh description, source locations and receiver locations

```
for each CPU do                                          ▷ Initialization phase
    Initialize ray directions                            ▷ partial initial mesh (uniform)
end for
while true do                                            ▷ Propagation phase
    for each CPU do
        Trace individual rays by one wavefront time step
    end for
    for each CPU do
        Construct a wavefront quadrilateral mesh
        Interpolate/remove rays in/from the mesh         ▷ for next simulation step
    end for
    if no ray tubes remain then
        break
    end if
end while
for each CPU do                                          ▷ Surface mapping phase
    if a ray tube passed through a receiver on surface then
        record travel time and amplitude
    end if
end for
```

Output: Ray path for each ray, wavefront, arrivals (travel time and amplitude) for each receiver, and surface map (wavefront mesh cells on the surface)

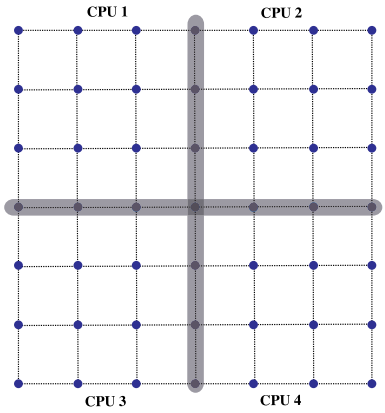
2.3. Parallel wavefront construction performance

The pWFC algorithm consists of three main phases: the initialization phase, the propagation phase, and the surface mapping phase. The main factor affecting initialization performance is the wavefront mesh description (initial number of rays and their directions). The initialization phase shows strong scalable performance using large initial meshes (large numbers of rays) (Jain, 2011). One main idea of the WFC is to begin with a sparse set of rays. The time taken for initialization is negligible compared to the propagation phase, which is the most expensive part of the algorithm. Several user specified input factors can affect pWFC's propagation phase performance, such as the earth model description (isotropic, anisotropic, number of interfaces, and the choice of stratified or gridded regions), the desired ray behavior (transmitted rays, reflected rays, or both), the number of sources and their locations, and the desired wave field accuracy (determined by the ray tracing step size, wavefront step size, interpolation threshold, and coarsening threshold). The propagation phase shows to have good scalable performance, however, load imbalances between CPUs cause a considerable loss in performance (Jain, 2011). Finally, two main factors affect the performance of the surface mapping phase: the number of receivers on the surface and the number of multiple arrivals.

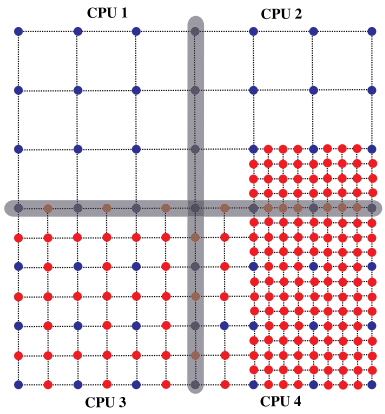
Load imbalances arise whenever there are large differences in the work distribution, leading to overloaded and underloaded processors. Load imbalances are one of the most investigated issues in parallel computing, including seismic modeling (Grunberg et al., 2004). Load balancing minimizes process idle time through approximate uniform work distribution among the processors. There are two main types of load balancing algorithms: static load balancing and dynamic load balancing. Static load balancing distributes work based on fixed and preconfigured rules and is usually done before execution. In contrast, dynamic load balancing algorithms monitor and redistribute the work load during execution (Hanxleden and Scott, 1991). Recently, more scalable performance of STAPL-based motion planning applications has been achieved using dynamic load balancing (Fidel et al., 2014).

The original pWFC implementation uses static load balancing based on uniform domain decomposition. In the initialization phase, the wavefront mesh is uniformly partitioned between processors (Fig. 2a). During the first wavefront steps, rays are almost evenly distributed between processes. However, as the simulation progresses we typically witness a non-uniform distribution of the rays among processes. Two main factors affect ray distribution in a wavefront mesh: ray terminations and ray interpolations. Rays can be terminated for several reasons, such as when they reach the earth model boundary or hit a surface with post-critical incidence, or when the mesh is coarsened when the algorithm detects that rays are oversampling the wavefront locally. Conversely, rays can be interpolated (added) to parts of the mesh based on the local ray field behavior (Fig. 1). This adaptive nature of wavefront mesh density is the main reason for pWFC load imbalances (Fig. 2b).

Our ultimate goal is to create a load-balanced pWFC application, so



(a)



(b)

Fig. 2. Schematic illustration of uniform partitioning of an initial wavefront mesh with 49 rays (36 ray tubes). Blue circles represent old rays, and red circles represent new interpolated rays. Shaded area indicates rays that are shared between different CPUs. (a) Initial wavefront mesh at τ_0 that shows identical loads on all four CPUs. (b) Wavefront mesh at an arbitrary time τ_i that shows load imbalances among the CPUs. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

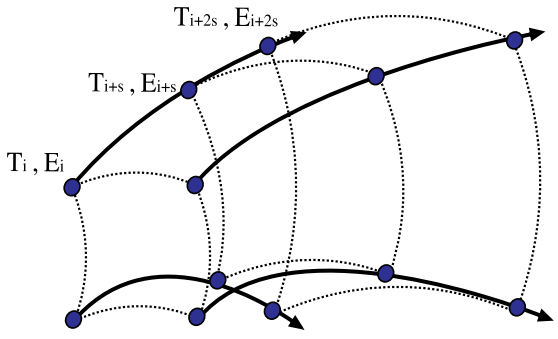


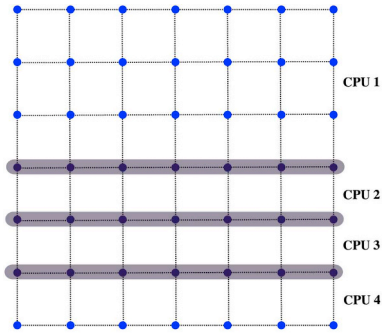
Fig. 3. Schematic illustration of preliminary ray tracing (without interpolation) used for cost estimation. Arrows represent rays, dotted lines represent the mesh geometry, and the blue circles are rays at constant time steps. T_i and E_i are the ray tube's travel time and error at arbitrary time i. The time T and error E are recorded at each predefined time step (wavefront time step) s until the ray tube termination. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

existing load balancing algorithms and options were investigated. Dynamic load balancing algorithms have been suggested in the literature (Jain, 2011). However, in this research, we develop and investigate a new static load balancing approach that predicts the future wavefront mesh density and then non-uniformly partitions the initial wavefront mesh based on this prediction, so that the eventual partitioning will be more balanced.

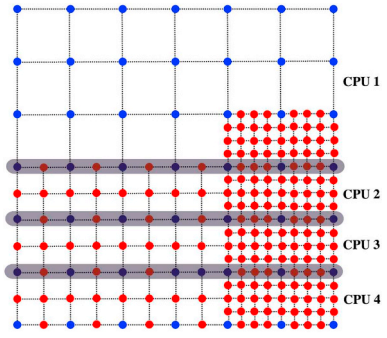
2.3.1. *Wavefront mesh density prediction*

As the wavefront propagates through the model, new rays are interpolated into or deleted from the mesh. This behavior causes variations in the wavefront mesh density that will typically lead to non-uniform distributions of rays or ray tubes among the processors (i.e., load imbalances). Therefore, predicting the future mesh density should help in choosing the correct number of processors and their load distribution to minimize pWF turnaround time. Since selecting an adequate sparse set of initial rays is very beneficial for WFC (Coman and Gajewski, 2001), this prediction can be used to guide the choice for the initial ray set density and the ray directions.

We estimate the computational cost of each cell (ray tube) in the initial wavefront mesh using a preliminary ray simulation in the targeted earth model. Prior to the WFC propagation phase, we trace each ray tube in the initial wavefront mesh without interpolation or coarsening. During this preliminarily simulation, we capture each ray tube's travel time and error in consecutive predefined time steps (wavefront



(a)



(b)

Fig. 4. Schematic illustration of non-uniform partitioning of an initial wavefront mesh with 49 rays (36 ray tubes). Blue circles represent old rays and red circles represent new interpolated rays. Shaded area indicates rays that are shared between different CPUs. (a) Initial wavefront mesh at τ_0 showing the different loads on the CPUs. (b) Wavefront mesh at an arbitrary time τ_i showing a load balance among the CPUs. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

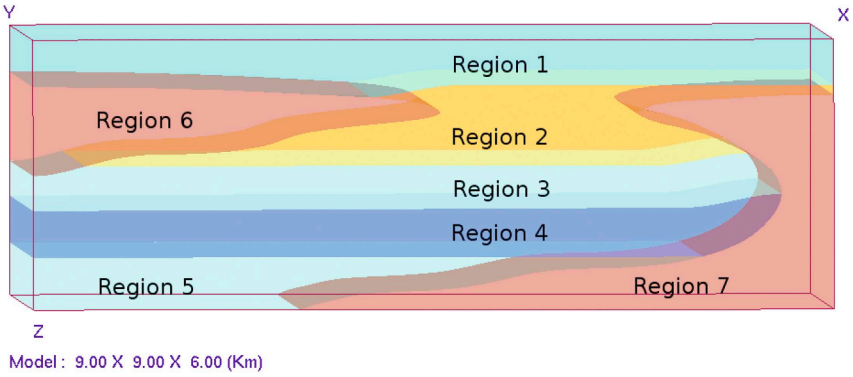


Fig. 5. A synthetic Gulf of Mexico salt dome model. Regions 1, 4, and 5 are isotropic regions. Regions 2 and 3 are transverse isotropic (VTI) regions. Regions 6 and 7 are salt regions (Jain, 2011).

time step; Fig. 3). We then calculate a weight for each ray tube. We investigated three approaches to calculating the weight in this study, (1) maximum error (Max_E), (2) maximum travel time (Max_T), and (3) maximum error and travel time (Max_ET). Finally, based on the chosen approach, we predict the computational cost of each cell.

Wavefront curvature can change as waves propagate away from the source. These changes cause WFC to interpolate new rays based on an accuracy criterion. Thus, it is reasonable to use this accuracy criterion to estimate how many rays might be interpolated in the future WFC simulation. We have therefore defined ray tube error in our method as the difference between the ray tube's travel time for WFC and the predicted paraxial travel time (Appendix A.2), which is the criterion used for ray interpolation in WFC.

In our first approach to weight prediction (Max_E), we calculate the weight of a ray tube by capturing its maximum error as follows:

$$W_n = \max_{0 \le i \le n_{wavefronts}} E_i, \tag{1}$$

where W_n is the weight of ray tube n, i is the time step index for the ray tube, $n_{wavefronts}$ is the number of wavefronts (time steps) of ray tube n, and E_i is the error of ray tube n at index i.

Since some ray tubes can terminate sooner than other ray tubes, we introduced travel times in our second approach to weight prediction (Max_T). In this approach, we calculate the weight by capturing the maximum travel time of each ray tube as follows:

$$W_n = \max_{0 \le i \le n_{wavefronts}} T_i, \tag{2}$$

where T_i is ray tube n's travel time at time step i.

The two previous weighting approaches consider the maximum time or the maximum error, which for the most ray tubes occur at the last time step. However, our third approach (Max_ET) captures both maximum travel time and maximum error at each time step. We then estimate weights by multiplying these values as follows:

$$W_n = \left(\max_{0 \le i \le n_{wavefronts}} E_i \right) \times \left(\max_{0 \le i \le n_{wavefronts}} T_i \right). \tag{3}$$

Finally, we calculate the total computational cost of each cell in the initial wavefront mesh using one of these three weighting approaches, as follows:

$$C_n = \frac{W_n}{\sum_{n=0}^{n=n_{tubes}} W_n} \times 100 \tag{4}$$

$$\sum_{n=0}^{n=n_{tubes}} C_n = 100\%, \tag{5}$$

where C_n is the final cost of ray tube n and n_{tubes} is the number of ray tubes (cells) in the initial mesh.

2.3.2. Non-uniform wavefront mesh partitioning

In this research we used the cubed sphere ray initialization method, which is based on a focal cube surface surrounding the source (Lee, 2005). The wavefront mesh is virtually divided into six faces: +X, -X, +Y, -Y, +Z, and -Z. These faces help identify the direction of rays; however, during mesh decomposition the entire wavefront mesh is considered as a single entity.

As stated earlier, load imbalances occurred in the uniform domain (initial mesh) decomposition implementation. However, since the number of mesh rays is much smaller during the first simulation steps, we believe it is more efficient to have a load imbalance at this stage rather than during later simulation steps. Therefore, we implemented a simple approach to non-uniformly partition the initial mesh. In our approach we distribute the domain based on partition computational costs rather than partition sizes, by decomposing the mesh into 2D rectangular blocks. Using this approach proved that our cost prediction is effective. Theoretical assumptions of even and uneven cell distributions are illustrated in Figs. 2 and 4.

3. Load balancing evaluation

3.1. Earth model

In order to study load imbalance, it was necessary to select a relatively complex earth model. Therefore, we built our 3D model based on a synthetic Gulf of Mexico salt dome model (Willis et al., 2006; Lu et al., 2009; Jain, 2011) (Fig. 5). This model consists of three isotropic regions, two vertical transverse isotropic (VTI) regions, and a salt dome region with a nearby salt canopy region (Table 1). The size of this 3D model is (9.00,9.00,6.00) km.

3.2. Input parameterization

Three salt dome model (Fig. 5) test cases were evaluated in this research. Each test case used the same input parameters, as shown in Table 2. The only difference between the three test cases was the source location. However, all three sources were located in isotropic Region 4. Each test case was executed using the original pWFC with uniform partitioning and combinations of 1, 2, 4, 8, 16, 32, and 64 CPUs.[3] Fig. 6 shows the preliminary ray tracing (without interpolation) used for the

[3] All research evaluations were conducted on a Cray XE6m supercomputer at the Texas A&M University Parasol Lab.

Table 1
Regions' physical properties in the salt dome model. V_p is the p-wave velocity, V_s is the s-wave velocity, ρ is the density, and c_{ijkl} is the stiffness tensor.

Region number	Physical properties
1	V_p = 3 km/s, V_s = 1.73 km/s, ρ = 2.5 g/cm^3
2	$c_{ijkl} = \begin{pmatrix} 20.28 & 13.104 & 15.028 & 0.0 & 0.0 & 0.0 \\ 13.104 & 20.28 & 15.028 & 0.0 & 0.0 & 0.0 \\ 15.028 & 15.028 & 22.542 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 4.498 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 4.498 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 3.588 \end{pmatrix}$ ρ = 2.4 g/cm^3
3	$c_{ijkl} = \begin{pmatrix} 25.9 & 6.825 & 7.075 & 0.0 & 0.0 & 0.0 \\ 6.825 & 25.9 & 7.075 & 0.0 & 0.0 & 0.0 \\ 7.075 & 7.075 & 23.775 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 7.325 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 7.325 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 9.525 \end{pmatrix}$ ρ = 2.5 g/cm^3
4	V_p = 3.4 km/s, V_s = 1.83 km/s, ρ = 2.67 g/cm^3
5	V_p = 3.8 km/s, V_s = 1.9 km/s, ρ = 2.7 g/cm^3
6	V_p = 4.78 km/s, V_s = 2.7 km/s, ρ = 2.2 g/cm^3
7	V_p = 4.78 km/s, V_s = 2.7 km/s, ρ = 2.2 g/cm^3

Table 2
WFC input parameters for all test cases.

Parameter	Value
Source location	Test case 1 = (1.0,4.5,4.2) km Test case 2 = (4.0,4.5,4.2) km Test case 3 = (7.0,4.5,4.2) km
Ray initialization method	Cubed sphere
Number of initial rays	17 × 17 on each face of the focal cube Total of 1724 rays
Number of initial ray tubes	16 × 16 on each face of the focal cube Total of 1536 ray tubes
Wave type	P-wave
Ray selection	Only transmitted rays
Ray tracing step size	0.01 s
Wavefront step size	0.04 s
Interpolation threshold	0.001 s
Coarsening threshold	0.0002 s

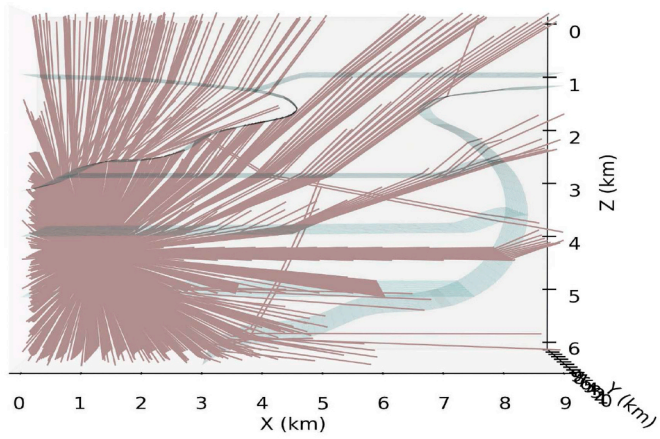
cost (load) estimation. Supplementary media for this paper show the WFC method wave simulations and preliminary ray tracing for each test case (Appendix B).

3.3. Weight prediction assessment

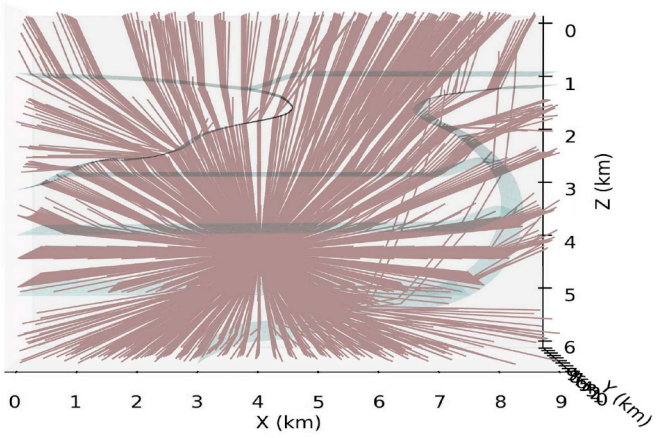
Our first test was designed to verify that all three test cases exhibited load imbalance, and to evaluate our approaches to weight prediction. Using preliminary ray tracing we calculated the cost of each cell in the initial wavefront mesh for the three test cases based on the three suggested approaches (Max_E, Max_T and Max_ET). Then we captured the actual load (number of ray tubes) for each CPU using the original pWFC algorithm with uniform partitioning. Following that, and using the original mesh decomposition, we conducted a comparison of the actual and predicted loads for each test case with 2, 4, 8, and 16 CPUs.

Fig. 7 shows two examples of the accuracy of our three prediction approaches, i.e, the differences between the predicted loads and the actual loads. These plots clearly show the load imbalances between the CPUs. They also show that all three prediction approaches estimate load fairly well.

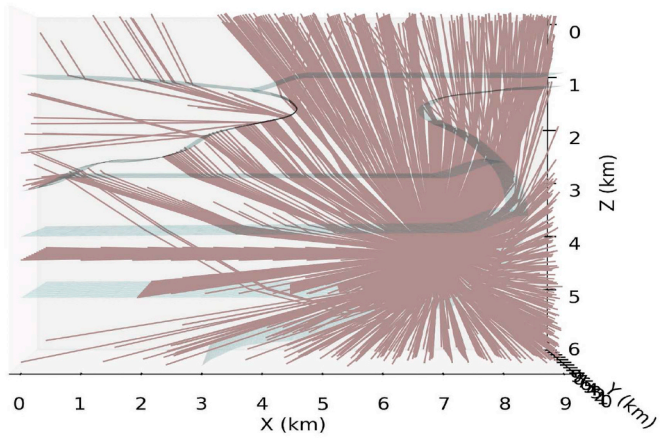
To determine the best approach to weight prediction we calculated



(a)

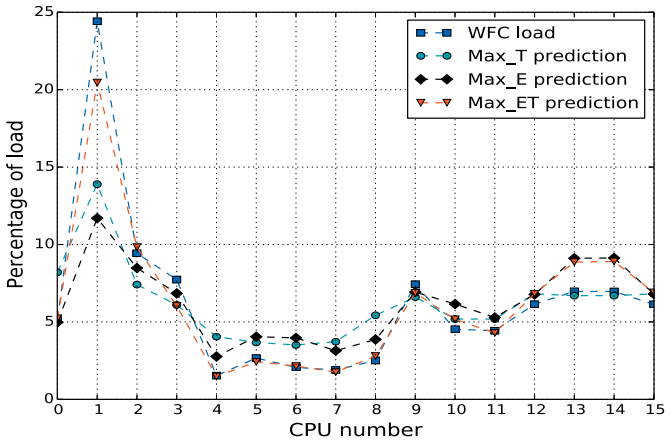


(b)

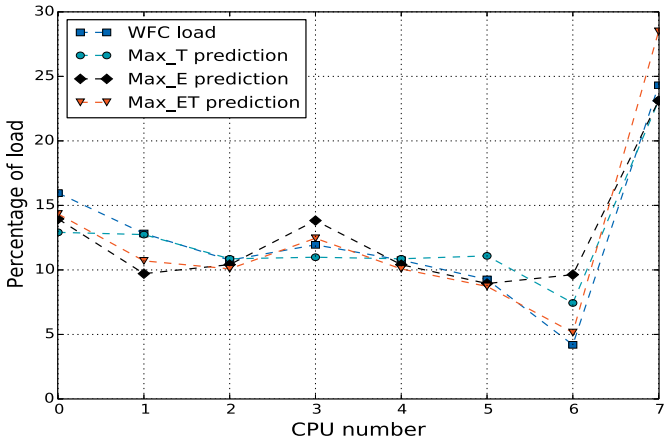


(c)

Fig. 6. Preliminary 3D ray tracing (without interpolation) in salt dome model that was used for cost estimation.(a) Test case 1 with a source located at (1.0,4.5,4.2) km. (b) Test case 2 with a source located at (4.0,4.5,4.2) km. (c) Test case 3 with a source located at (7.0,4.5,4.2) km.



(a)



(b)

Fig. 7. Two examples comparing the actual pWFC CPU load to the predicted loads based on the preliminary ray tracing (used for density prediction) for (a) test case 1 with a source located at (1.0,4.5,4.2) km using 16 CPUs, and (b) test case 2 with a source located at (4.0,4.5,4.2) km using 8 CPUs.

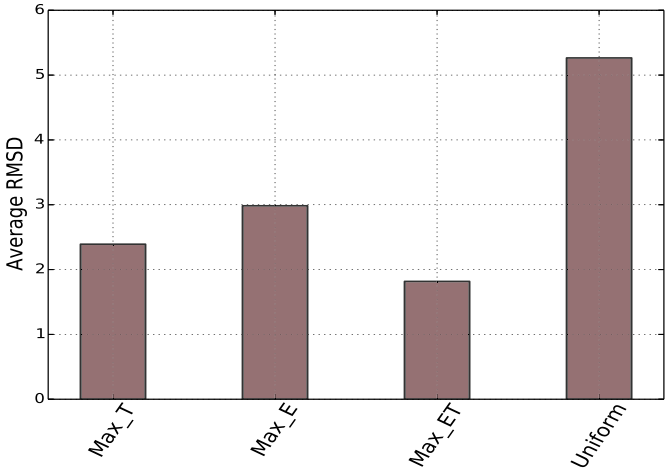


Fig. 8. The average root-mean-square deviation *RMSD* between the pWFC (actual) and predicted load for each weight prediction approach, using a combination of 2,4,8, and 16 CPUs. Max_T is the maximum travel time weight prediction approach, Max_E is the maximum error weight prediction approach, Max_ET is the maximum travel time and error weight prediction approach, and Uniform is the original (uniform) load distribution.

the root-mean-square deviation *RMSD* between the pWFC (actual) and predicted loads for each approach for all test cases with combinations of 2, 4, 8, and 16 CPUs, as follows:

$$RMSD = \sqrt{\frac{\sum_{c=0}^{c=n_{cpus}} (P_c - A_c)^2}{n_{cpus}}}, \tag{6}$$

where n_{cpus} is the number of CPUs (2, 4, 8, or 16), A_c is the pWFC (actual) percentage of the load for CPU c and P_c is the predicted percentage of the load for CPU c. Then we calculated the average *RMSD* for the three approaches to weight prediction for all test cases (Fig. 8). The original (uniform) load distribution, which assumes that each CPU will have the same load, has the highest *RMSD* compared to our three approaches. The introduction of ray tube travel time in Max_T and Max_ET results in a better load prediction than Max_E. Overall, these results show that Max_ET has the smallest *RMSD* from the actual load.

3.4. Non-uniform partitioning evaluation

After choosing Max_ET as the best prediction approach, we executed the modified pWFC (non-uniform partitioning), using combinations of 1, 2, 4, 8, 16, 32, and 64 CPUs for each test case. During wavefront propagation, using both uniform and non-uniform decomposition, we recorded the CPU load at each simulation time step. Next, we calculated the improvement of load distribution for each simulation run by measuring the percentage of diference between *RMSD* of total load percentage using original pWFC implementation and optimal load percentage (perfectly balanced), and *RMSD* of total load percentage using our pWFC implementation and optimal load percentage as follows:

$$improvement = \frac{RMSD_u - RMSD_n}{RMSD_n} \times 100, \tag{7}$$

where $RMSD_n$ is the root-mean square deviation between the uniform decomposition's total CPUs load percentages and optimal load percentages, and $RMSD_u$ is the root-mean square deviation between the non-uniform decomposition's total CPUs load percentages and optimal load percentages. In the following, we discuss two examples comparing uniform and non-uniform decomposition by cost prediction.

Before discussing the efficiency of the non-uniform decomposition, we show how the cost percentage is distributed among the wavefront mesh cells. Fig. 9 shows the cost estimation for each cell in the initial wavefront mesh. Using the cubed sphere initialization method, the initial wavefront consists of six faces (+X, +Y, -X, -Y, +Z, and -Z) that represent a focal cube for ray direction initialization. For example, Fig. 9 shows a realistic high cost percentage in the +Z face for all three test cases. This high cost arises because of ray tube divergence (a change in the wavefront curvature) in that direction, which increases the difference between the ray tube travel time and the paraxial travel time prediction. Also, rays in this direction travel longer than rays in the -Z direction, as can clearly be seen in Fig. 6. Another observation is the symmetry between the +Y and the -Y faces for all test cases. This symmetry exists because most rays traced in the +Y and -Y directions were in the same region, and the sources were located in the middle of Y axis. Therefore, in our new pWFC implementation, the initial wavefront mesh is decomposed into non-equal partitions so that all parts will have roughly equal shares of the cost. This might lead to a load imbalance in the first wavefront propagation steps. However, in later wavefront steps, which usually have a larger numbers of rays to process, the ray tubes load should be distributed more evenly between CPUs compared to the uniform decomposition distribution, thereby decreasing the turnaround time for pWFC execution.

The first example comparing uniform and non-uniform decomposition involves test case 2 with a source located at (4.0,4.5,4.2) km. First, we ran this test case on 4 CPUs using the original pWFC (with uniform partitioning). Fig. 10a shows the initial uniform decomposition distributing the rays/ray tubes among the CPUs. Fig. 11a shows the

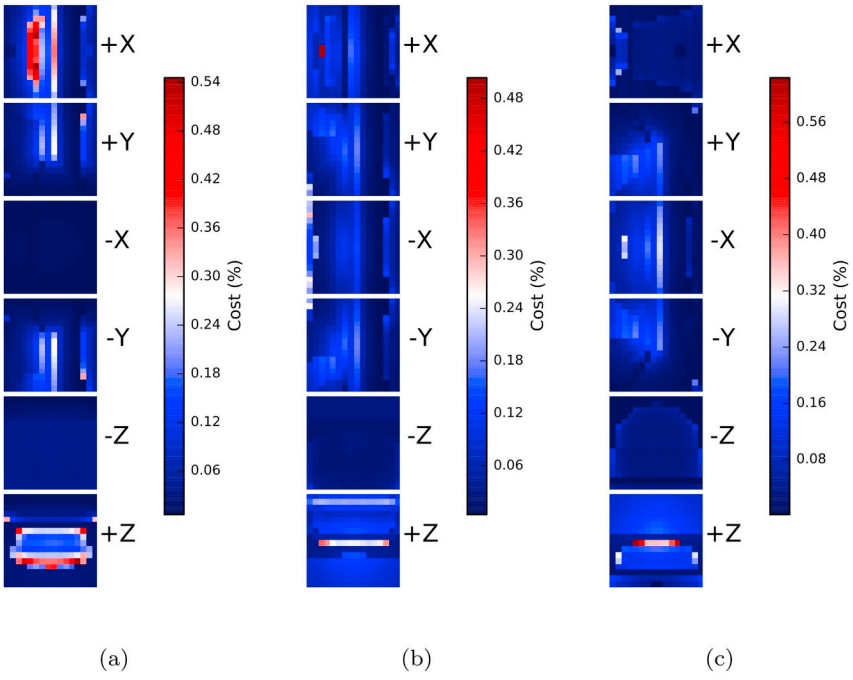


Fig. 9. Initial wavefront mesh cost distribution between cells using the Max_ET approach to prediction. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. Red indicates a high potential future load and blue indicates a low potential future load. The total cost for each wavefront mesh is 100%. (a) Test case 1 with a source located at (1.0,4.5,4.2) km, (b) test case 2 with a source located at (4.0,4.5,4.2) km, and (c) test case 3 with a source located at (7.0,4.5,4.2) km. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

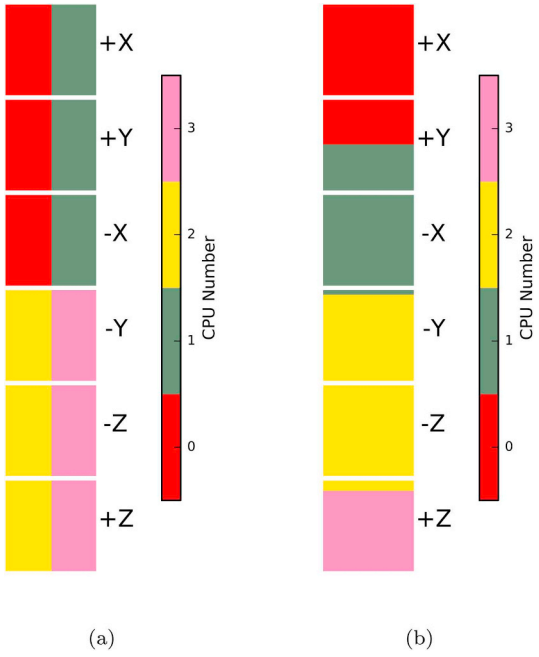
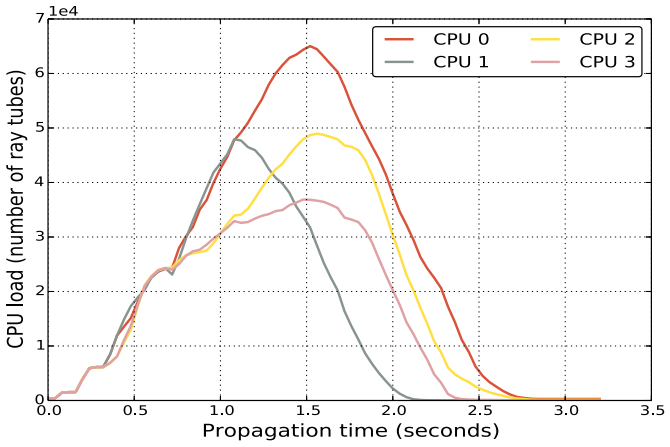


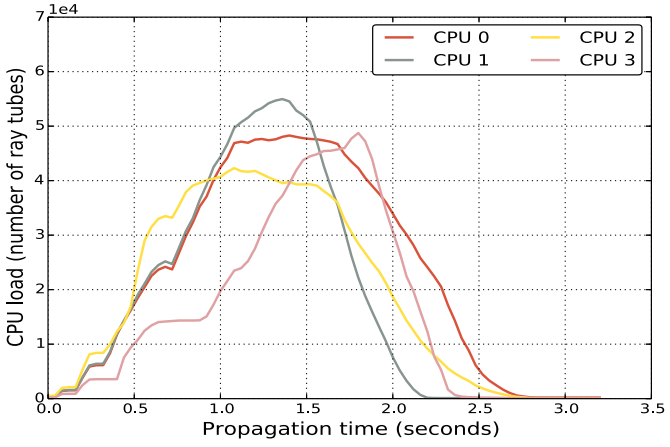
Fig. 10. Initial wavefront mesh partitioning for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

amount of the ray tube load that each CPU has during the wavefront propagation. It is obvious that CPU 0 (red color) has a larger load than the other CPUs, and this is clearly reflected in the high predicted cost for that part of the initial mesh (Figs. 9b and 11a). Next we ran our modified pWFC (with non-uniform partitioning) using the same number of CPUs. The non-uniform decomposition of the initial wavefront mesh is based on the ray density predictions (Fig. 9b) using preliminary ray tracing (Fig. 6b). Thus, Fig. 10b shows different partition sizes for the CPUs, and Fig. 11b shows the amount of the ray tube load that each CPU has during the wavefront propagation. This execution achieves a reduction of more than 1000 ray tubes for the average CPUs load difference (Fig. 12), and improves the total CPUs load distribution by more than 45% (Fig. 13). pWFC wavefront simulation results are exactly the same for both versions.

A second example comparing uniform and non-uniform decomposition involves test case 1 with the source located at (1.0,4.5,4.2) km. First, we ran this test case on 4 CPUs using the original pWFC (with uniform partitioning). Fig. 14a shows the initial uniform decomposition distributing the rays/ray tubes among the CPUs. Fig. 15a shows the amount of the ray tube load that each CPU has during the wavefront propagation. Next we ran our modified pWFC (with non-uniform partitioning) using the same number of CPUs. The non-uniform decomposition of the initial wavefront mesh is based on the ray density predictions (Fig. 9a) using preliminary ray tracing (Fig. 6a). Thus, Fig. 14b shows different partition sizes for the CPUs, and Fig. 15b shows the amount of the ray tube load that each CPU has during the wavefront propagation. In contrast to the first example, here the non-uniform partitioning results in higher load differences compared to uniform partitioning (Fig. 16). For instance, CPU 2 (yellow color) has a high load compared to other CPUs (Fig. 15b), which exists only in the first simulation steps. This behavior appears because CPU 2 is responsible for tracing a large portion of the wavefront mesh with rays that

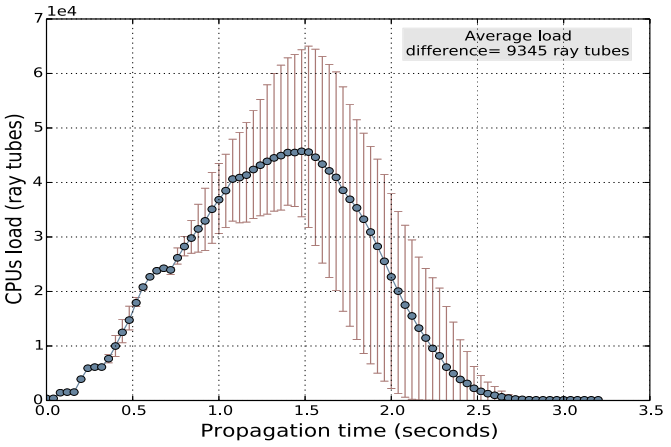


(a)

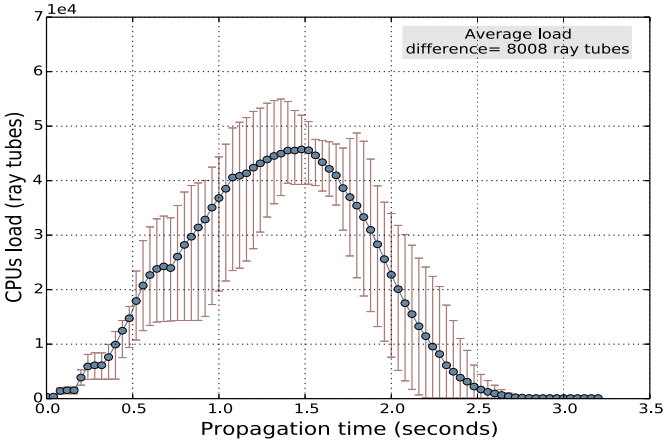


(b)

Fig. 11. CPU load profiles during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. The colors and CPU numbers can be cross-referenced with the wavefront mesh partitioning shown in Fig. 10. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)



(a)



(b)

Fig. 12. Difference between CPU loads (ray tubes) during wavefront propagation for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. Green circles indicates average load, and red bars show maximum and minimum load. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

terminated sooner than others (-X and -Z faces) (Figs. 14b and 6a). Therefore, even though the total load distribution is improved by more than 55% (Fig. 17), in this case the average difference in the number of ray tubes between CPUs increases (Fig. 16). This should also result in a slight decrease in performance compared to the original pWFC implementation.

3.5. Final performance results

As discussed, both versions of pWFC were executed with combinations of 1, 2, 4, 8, 16, 32, and 64 CPUs. The execution time of preliminary ray tracing is included in the total execution time of the modified pWFC (non-uniform decomposition). However, it is a very small fraction of the overall computation time and does not impact the final results. In order to accurately measure the performance, each simulation was run five times to minimize the impact of any external influences. The differences in execution times between the five runs was negligible compared to the absolute total execution time. For most of the cases, the non-uniform initial decomposition showed a faster turnaround time (Fig. 18). For

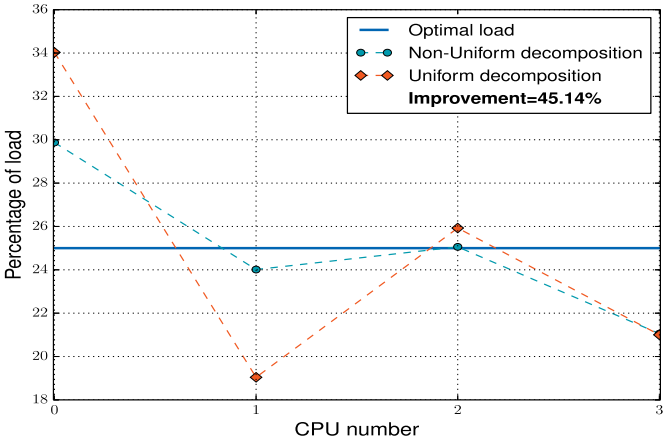


Fig. 13. Total CPUs' percentage of load distribution of uniform and non-uniform decomposition for test case 2 with a source located at (4.0,4.5,4.2) km using 4 CPUs. The percentage of improvement is calculated using equation (7).

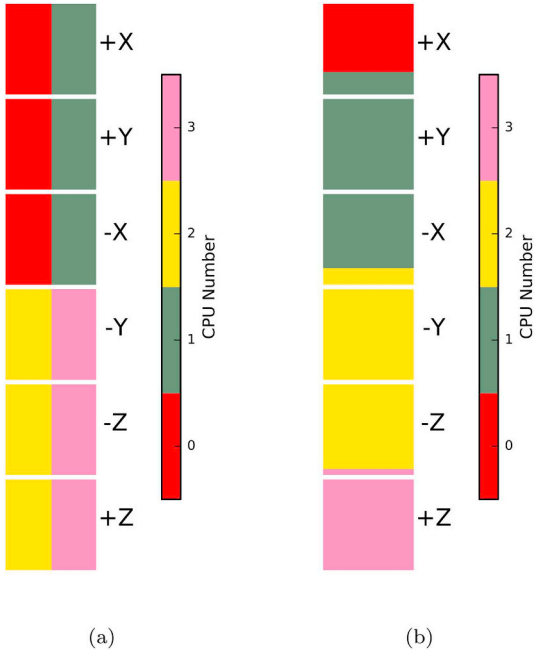


Fig. 14. Initial wavefront mesh partitioning for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. The mesh face layout is represented here as it is implemented in the software. Each focal cube face represents the direction of the ray tube. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning.

instance, our modified pWFC shows an average improvement of 23% for test case 1. Therefore, we can see in Fig. 19 that our modified pWFC achieves better scalability. The scalability of a parallel application describes how efficiently the work load can be distributed across multiple CPUs. Ideal scalability is linear, i.e. doubling the number of CPUs would result in half the execution time. It is interesting that the original pWFC implementation has unstable (unsystematic) scalability compared to our implementation. This instability appears because uniform decomposition may, in some circumstances, lead to better load balancing. However, in our implementation, where initial wavefront mesh is partitioned based on a systematic criterion (future ray density), scalability appears more stable.

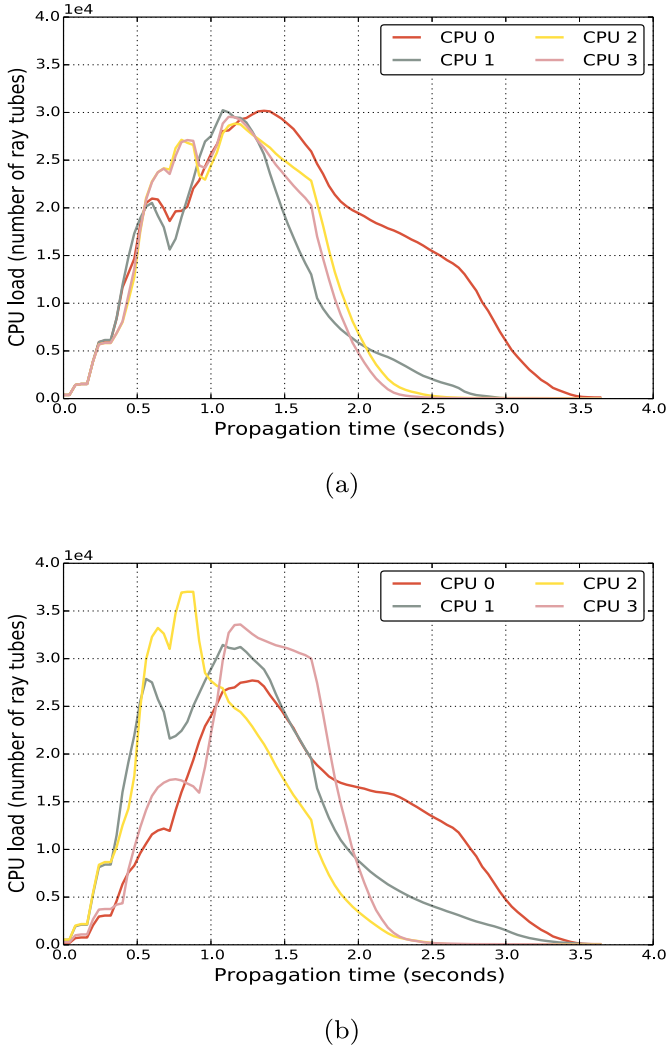


Fig. 15. CPU load profiles during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. The colors and CPU numbers can be cross-referenced with the wavefront mesh partitioning shown in Fig. 10. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

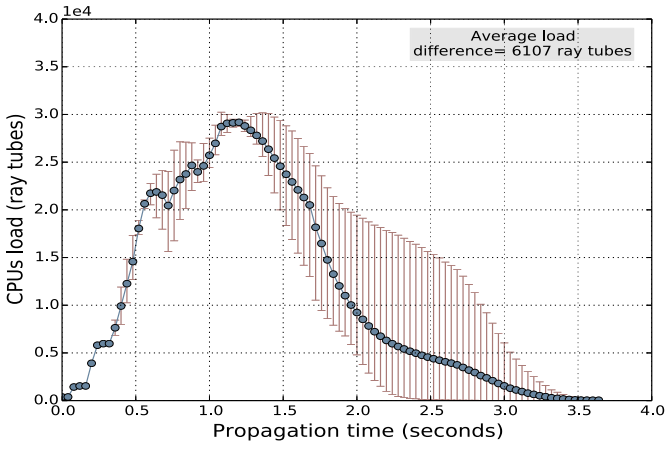
4. Discussion

We built three separate test cases (based on a synthetic salt dome model) to evaluate our proposed approaches to weight estimation (Max_E, Max_T and Max_ET) and to analyze the behavior of our non-uniform mesh partitioning scheme.

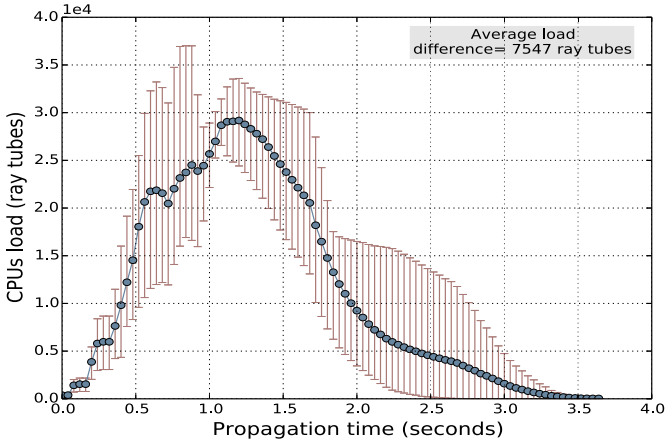
The weight prediction approaches were used to first prove the existence of a load imbalance and then to assess the accuracy of each approach's estimation of the imbalance. All three test cases showed a clear load imbalance when using a range of 4–64 CPUs. Two main variables contributed to the weight predictions in all three approaches: the ray tube travel time and error. The ray tube error, which is based on paraxial travel time prediction, is suggested due to its importance in indicating the probability of future ray interpolation. However, the use of both variables, travel time and error, together proved to provide more accurate load estimation than either one separately. This establishes the importance of using travel time in the load predictions, because some ray tubes may be terminated sooner than others and this leads to load imbalances.

We compared uniform and non-uniform mesh decomposition to show the efficiency of our cost (load) prediction method. We proved its effectiveness using a simple 2D block (rectangular) decomposition technique that decomposes the initial wavefront mesh into parts that have roughly equal costs regardless of their sizes. In the future, it would be interesting to investigate the behavior of non-rectangular decomposition. However, CPU communication costs should also be considered in this situation.

Generally, seismic modeling applications use multiple sources that are distributed over the model. However, it is expensive to perform the



(a)



(b)

Fig. 16. Difference between CPU loads (ray tubes) during wavefront propagation for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. Green circles indicates average load and red bars show maximum and minimum load. (a) Original pWFC implementation using uniform partitioning. (b) Our modified pWFC implementation using non-uniform partitioning. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

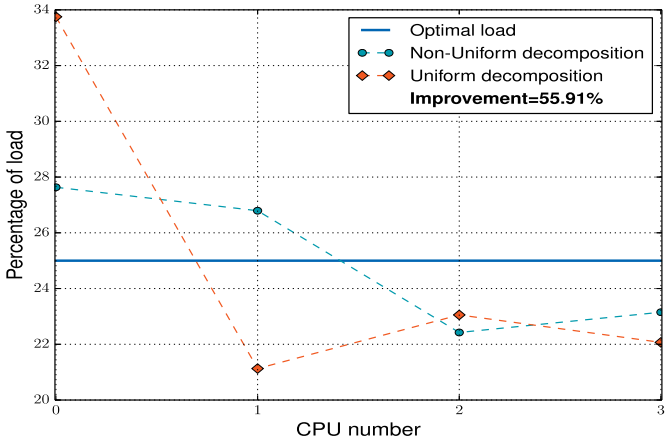
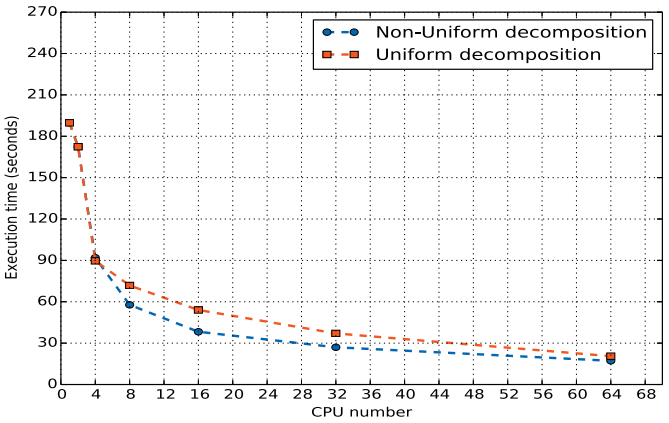
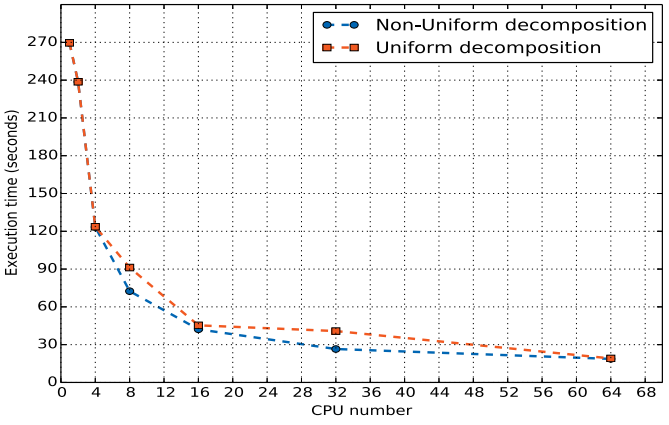


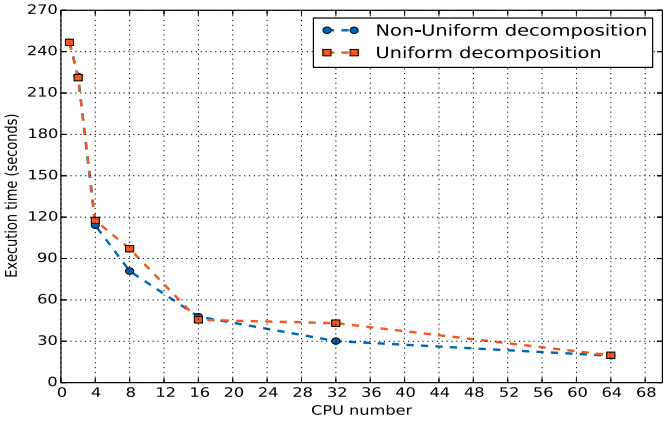
Fig. 17. Total CPUs' percentage of load distribution of uniform and non-uniform decomposition for test case 1 with a source located at (1.0,4.5,4.2) km using 4 CPUs. The percentage of improvement is calculated using equation (7).



(a)

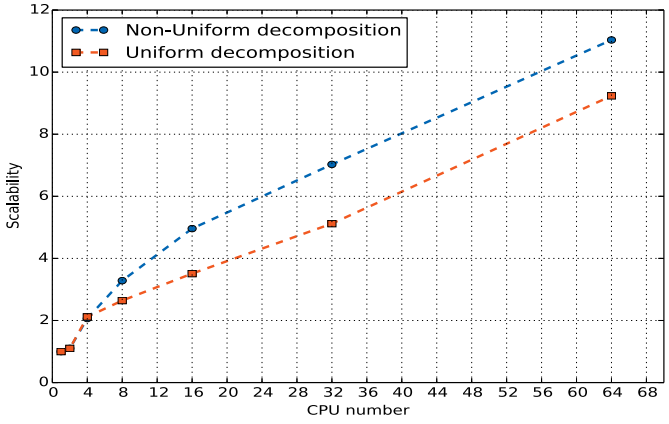


(b)

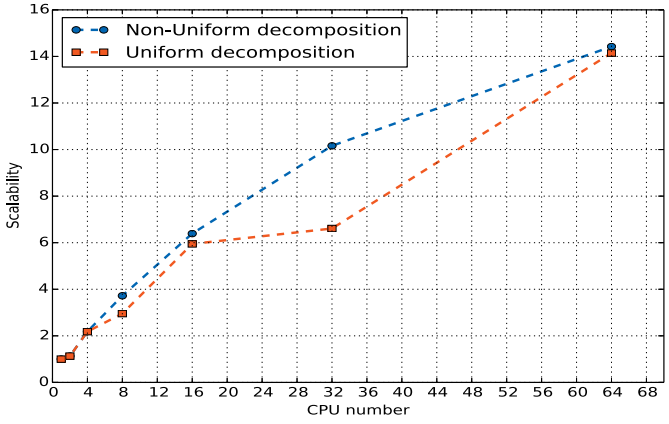


(c)

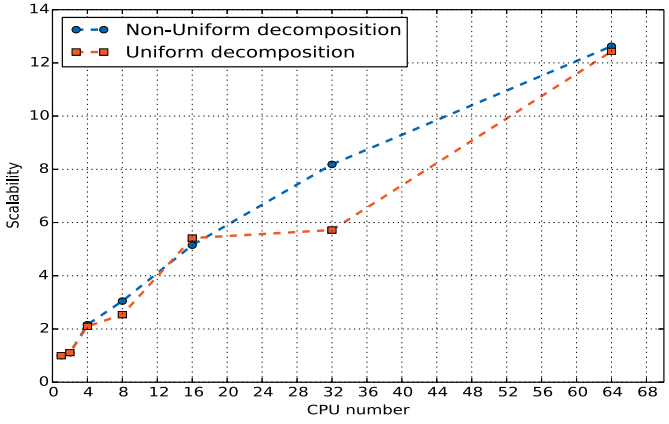
Fig. 18. Comparison of the total execution times for the original pWFC (uniform partitioning) and our modified pWFC (non-uniform partitioning) using 1, 2, 4, 8, 16, 32, and 64 CPUs for (a) test case 1 with a source located at (1.0,4.5,4.2) km, (b) test case 2 with a source located at (4.0,4.5,4.2) km, and (c) test case 3 with a source located at (7.0,4.5,4.2) km. Preliminary ray tracing run time, used for cost estimation, is included in the total execution time of our modified pWFC.



(a)



(b)



(c)

Fig. 19. Comparison of the scalability of the original pWFC (uniform partitioning) and our modified pWFC (non-uniform partitioning) using 1, 2, 4, 8, 16, 32, and 64 CPUs for (a) test case 1 with a source located at (1.0,4.5,4.2) km, (b) test case 2 with a source located at (4.0,4.5,4.2) km, and (c) test case 3 with a source located at (7.0,4.5,4.2) km.

predictive ray tracing at every source location, therefore it would be preferable to use a subset of source locations. A computational cost for the initial meshes (for each source) can then be estimated, and this prediction can be interpolated and applied through subvolumes. Therefore, this process will help in predicting the complexity of the earth model and efficiently estimating the future load for multiple sources.

A few simulation runs demonstrated a slight decrease in the performance of our pWFC implementation compared to the original uniform implementation. This decrease appeared because the uniform decomposition can sometimes fortuitously lead to a more balanced ray tube distribution. However, considering the final performance results over multiple source locations and multiple numbers of CPUs, our pWFC implementation provided superior load balancing, on the average, compared to the uniform distribution approach.

All simulation runs showed our implementation resulted in a large improvement of CPU load distribution. However, results also showed this improvement does not guarantee even load balancing between CPUs, because there may still be a considerable variation in load in some specific wavefront simulation steps as the wavefront propagates through the model. These results are not surprising because we use both the maximum ray tube travel time and error in our cost estimation. However, since we know the ray tube travel time and error at each wavefront time step, we could apply the criterion during wave propagation to readjust the load balance (distribution of ray tubes) for future steps. This technique could potentially serve as the basis of a dynamic load balancing approach.

The present study showed, even with static load balancing, how better performance can be achieved is seismic modeling computations. This result can be used to accomplish our ultimate goal: a fully load-balanced pWFC. True dynamic load balancing will require a huge amount of CPU communication and work redistribution. Our proposed cost estimate can be used to predict future load imbalance and therefore can also be used to determine whether dynamic load balancing will be worthwhile. Additionally, for some model types and source locations a mixture of the two methods of load balancing, dynamic and static, may result in better performance.

5. Conclusion

This study has implemented a new static load balancing algorithm for pWFC. The method consists of two major parts: density load estimation and non-uniform decomposition. We first evaluated different approaches for predicting future loads. Next, we investigated a non-uniform partitioning of the initial wavefront mesh and tested our new algorithm on three test cases based on a synthetic salt dome model. Our results show that our new algorithm achieves better and stable scalability in most of the cases. Overall, this research studied the behavior of pWFC load imbalances, and the implemented technique should help in determining the best strategy for load-balancing pWFC applications.

Acknowledgment

We would like to thank Saudi Aramco for sponsoring Abdullah Alyabes graduate studies and helping publish this research. We also extend our thanks to Parasol Lab students at Texas A&M University for their excellent support.

Appendix A. Key results for WFC

The WFC method is based on applying a high-frequency approximation to the elastic wave equation. It simulates seismic wave propagation by tracing ray fields rather than individual rays. WFC begins with an initial sparse set of rays and interpolates new rays whenever an accuracy criterion is violated.

Appendix A.1. Seismic ray tracing

Ray paths, travel times, and amplitudes for general anisotropic media are calculated by solving a set of ordinary differential equations (ODEs) of the following form (Gibson et al., 2005):

$$\frac{dx_i}{d\tau} = a_{ijkl} p_l g_j g_k \tag{A.1}$$

$$\frac{dp_i}{d\tau} = -\frac{1}{2}\frac{da_{ijkl}}{dx_i} p_n p_l g_j g_k \tag{A.2}$$

$$a_{ijkl} = \frac{c_{ijkl}}{\rho}, \tag{A.3}$$

where τ is the travel time, x_i is a spatial coordinate component, a_{ijkl} is a density-normalized elastic modulus (stiffness tensor), p_i is a component of the slowness vector, and g_i is a component of the particle-motion vector. In the present research, these ODEs (A.1 and A.2) are solved for each ray using fifth-order Runge Kutta methods.

Appendix A.2. Predicted paraxial travel time

In this research, the accuracy criterion used for interpolating new rays is based on the predicted paraxial travel time. This is a second-order Taylor series expansion of travel time from a known location on ray x to a nearby location y in the following form (Lee and Gibson, 2007):

$$\tau(\mathbf{x}) \approx \tau(\mathbf{y}) + p_i(x_i - y_i) + \frac{1}{2}\frac{\partial^2\tau}{\partial x_i \partial x_j}(x_i - y_i)(x_j - y_j), \tag{A.4}$$

$$\frac{\partial^2\tau}{\partial x_i \partial x_j} = \frac{p_i}{\gamma_k}\left(\frac{\partial x_j}{\partial \gamma_k}\right)^{-1}, \tag{A.5}$$

where the slowness vector components p_i are the first derivatives of the travel time field. γ_k are ray coordinates, which are the travel time τ and azimuthal and inclination takeoff angles of the rays. This method is based on predicting the travel time for a point located diagonally to the reference ray. If the difference between the predicted and actual travel times exceeds a predefined threshold, new rays are interpolated in the mesh cell. The predicted paraxial travel time is also used to calculate the travel time for new rays and the travel time for receivers.

Appendix B. Supplementary data

Supplementary media for this research show the WFC method wave simulations and 3D preliminary ray tracing (without interpolation) for each executed test case in the targeted earth model (salt dome). Supplementary data to this article can be found online at https://doi.org/10.1016/j.cageo.2018.09.002.

References

Alaei, B., 2012. Seismic modeling of complex geological structures. In: Kanao, M. (Ed.), Seismic Waves - Research and Analysis. InTech, Rijeka, Croatia, pp. 213–233. Ch. 11. https://doi.org/10.5772/29423.

An, P., Jula, A., Rus, S., Saunders, S., Smith, T., Tanase, G., Thomas, N., Amato, N., Rauchwerger, L., 2003. STAPL: an adaptive, generic parallel C++ library. In: Dietz, H. (Ed.), Languages and Compilers for Parallel Computing. Springer, Berlin, Heidelberg, New York, pp. 193–208. Vol. 2624 of Lecture Notes in Computer Science. https://doi.org/10.1007/3-540-35767-X_13.

Bohlen, T., 2002. Parallel 3-D viscoelastic finite difference seismic modelling. Comput. Geosci. 28 (8), 887–899. https://doi.org/10.1016/S0098-3004(02)00006-7.

Buss, A., Harshvardhan, Papadopoulos, I., Pearce, O., Smith, T., Tanase, G., Thomas, N., Xu, X., Bianco, M., Amato, N.M., Rauchwerger, L., 2010. STAPL: Standard template adaptive parallel library. In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference. SYSTOR '10. ACM, New York, NY 14:1–14:10. https://doi.org/10.1145/1815695.1815713.

Carcione, J., Herman, G., ten Kroode, A., 2002. Seismic modeling. Geophysics 67 (4), 1304–1325. URL. https://doi.org/10.1190/1.1500393.

Červený, V., 2005. Seismic Ray Theory. Cambridge University Press, Cambridge, U.K 724pp. http://bit.ly/1k7qoOY.

Chambers, K., Kendall, J.-M., 2008. A practical implementation of wavefront construction for 3-D isotropic media. Geophys. J. Int. 173 (3), 1030–1038. https://doi.org/10.1111/j.1365-246X.2008.03790.x.

Chen, B., 2011. Efficient Smoothing and Interpolation of Velocity Models for Seismic Wavefront Construction Algorithms. M.Sc. Thesis. Texas A&M University, College Station, TX 46 pp. http://hdl.handle.net/1969.1/ETD-TAMU-2011-08-9761.

Coman, R., Gajewski, D., 2001. Estimation of multivalued arrivals in 3D models using wavefront ray tracing. In: SEG Technical Program Expanded Abstracts 2001. Society of Exploration Geophysicists, Tulsa, OK, pp. 1265–1268. https://doi.org/10.1190/1.1816324.

Fehler, M.C., Huang, L., 2002. Modern imaging using seismic reflection data. Annu. Rev. Earth Planet Sci. 30 (1), 259–284. https://doi.org/10.1146/annurev.earth.30.091201.140909.

Fidel, A., Jacobs, S.A., Sharma, S., Amato, N.M., Rauchwerger, L., 2014. Using load balancing to scalably parallelize sampling-based motion planning algorithms. In: Proceedings of IEEE 28th International Parallel and Distributed Processing Symposium. IEEE Computer Society, Los Alamitos, CA, pp. 573–582. https://parasol.tamu.edu/publications/download.php?file_id=865.

Gajewski, D., Coman, R., Vanelle, C., 2002. Amplitude preserving Kirchhoff migration: a traveltime based strategy. Studia Geophys. Geod. 46 (2), 193–211. https://doi.org/10.1023/A%3A1019849919186.

Gibson Jr., R., 2000. Ray tracing by wavefront construction for anisotropic media. In: SEG Technical Program Expanded Abstracts 2000. Society of Exploration Geophysicists, Tulsa, OK, pp. 2305–2308. https://doi.org/10.1190/1.1815919.

Gibson, R., Durussel, V., Lee, K., 2005. Modeling and velocity analysis with a wavefront-construction algorithm for anisotropic media. Geophysics 70 (4), T63–T74. https://doi.org/10.1190/1.1988188.

Gjøystdal, H., Iversen, E., Laurain, R., Lecomte, I., Vinje, V., Åstebøl, K., 2002. Review of ray theory applications in modelling and imaging of seismic data. Studia Geophys. Geod. 46 (2), 113–164. https://doi.org/10.1023/A:1019893701439.

Gjøystdal, H., Iversen, E., Lecomte, I., Kaschwich, T., Drottning, Å., Mispel, J., 2007. Improved applicability of ray tracing in seismic acquisition, imaging, and interpretation. Geophysics 72 (5), SM261–SM271. https://doi.org/10.1190/1.2736515.

Grunberg, M., Genaud, S., Mongenet, C., 2004. Seismic ray-tracing and earth mesh

modeling on various parallel architectures. J. Supercomput. 29 (1), 27–44. https://doi.org/10.1023/B%3ASUPE.0000022571.28175.e9.
Hanxleden, R.V., Scott, L., 1991. Load balancing on message passing architectures. J. Parallel Distr. Comput. 13 (3), 312–324. https://doi.org/10.1016/0743-7315(91)90078-N.
Jain, T.K., 2011. Parallel Seismic Ray Tracing. Unpublished M.Sc. Thesis. Texas A&M University, College Station, TX 75 pp.
Kaschwich, T., 2006. Traveltime Computation and Migration in Anisotropic Media. Ph.D. Dissertation. Universität Hamburg, Hamburg, Germany 142 pp. http://ediss.sub.uni-hamburg.de/volltexte/2007/3192.
Komatitsch, D., Gddeke, D., Erlebacher, G., Micha, D., 2010. Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs. Comput. Sci. Res. Dev. 25 (1–2), 75–82. https://doi.org/10.1007/s00450-010-0109-1.
Lee, K.J., 2005. Efficient Ray Tracing Algorithms Based on Wavefront Construction and Model Based Interpolation Method. Ph.D. Dissertation. Texas A&M University, College Station, TX 125 pp. http://hdl.handle.net/1969.1/3771.
Lee, K., Gibson, R., 2007. An improved mesh generation scheme for the wavefront construction method. Geophysics 72 (1), T1–T8. https://doi.org/10.1190/1.2399366.
Lu, R., Willis, M.E., Campman, X.H., Toksz, M.N., 2009. Evaluation of elastodynamic interferometric redatuming: a synthetic study on salt dome flank imaging. Geophys. J. Int. 176 (3), 889–896. https://doi.org/10.1111/j.1365-246X.2008.04019.x.
Mohammadzaheri, A., Sadeghi, H., Hosseini, S.K., Navazandeh, M., 2013. DISRAY: a distributed ray tracing by map-reduce. Comput. Geosci. 52 (0), 453–458. https://doi.org/10.1016/j.cageo.2012.10.009.
Musser, D.R., Derge, G.J., Saini, A., 2001. STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library, second ed. Addison-Wesley Professional, Indianapolis, IN 560pp.
Szostek, K., Leniak, A., 2012. Parallelization of the seismic ray trace algorithm. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waniewski, J. (Eds.), Parallel Processing and Applied Mathematics. Springer, Berlin, Heidelberg, New York, pp. 411–418. Vol. 7204 of Lecture Notes in Computer Science. https://doi.org/10.1007/978-3-642-31500-8_42.
Vinje, V., Iversen, E., Gjøystdal, H., 1993. Traveltime and amplitude estimation using wavefront construction. Geophysics 58 (8), 1157–1166. https://doi.org/10.1190/1.1443499.
Vinje, V., Iversen, E., Åstebøl, K., Gjøystdal, H., 1996. Estimation of multivalued arrivals in 3D models using wavefront construction—Part I1. Geophys. Prospect. 44 (5), 819–842. https://doi.org/10.1111/j.1365-2478.1996.tb00175.x.
Vinje, V., Åstebøl, K., Iversen, E., Gjøystdal, H., 1999. 3-D ray modeling by wavefront construction in open models. Geophysics 64 (6), 1912–1919. https://doi.org/10.1190/1.1444697.
Willis, M., Lu, R., Campman, X., Nafi Toksz, M., Zhang, Y., Hoop, M., 2006. A novel application of time-reversed acoustics: salt-dome flank imaging using walkaway vsp surveys. Geophysics 71 (2), A7–A11. https://doi.org/10.1190/1.2187711.