3D Seismic Modeling in a Message Passing Environment

Suhas Phadke, Dheeraj Bhardwaj and Sudhakar Yerneni*
Centre for Development of Advanced Computing, Pune University Campus, GaneshKhind, Pune 411007, India

SUMMARY

In this paper we describe the MPI (Message Passing Interface) implementation of an algorithm for solving 3D acoustic wave equation. The derivatives are approximated by central differences and the solution is obtained in a distributed computing environment. Numerical solution uses an explicit finite difference scheme, which is second order accurate in both space and time. Various domain decomposition schemes are used for distributing the workload and the tasks communicate via MPI message passing calls. The performance analysis shows that checkerboard type of scheme gives the best performance. The extension of the method to higher order finite difference schemes is straightforward.

INTRODUCTION

Seismic modeling is an integral part of the seismic processing, as it provides us the seismic response for a given earth model. The synthetic seismograms or the time slices generated by seismic modeling are used in processing as well as in interpretation. In the recent past a great deal of attention has been focused on the use of wave equation for modeling and imaging the Earth's interior (SEG/EAGE 3-D Salt and Overthrust Models. 1997). Seismic wave modeling algorithms used for calculating the seismic response for a given earth model, require large computational resources in terms of speed and memory (Phadke et al., 1998). The finite difference methods for modeling wave propagation in the earth offer a most direct solution to the problem expressed in terms of the basic equations and the initial and boundary conditions (Kelly et al. 1976, Villarreal and Scales 1997). Their usage for calculating synthetic seismograms on a regular basis was not possible until the advent of high performance computers.

In this paper we describe the implementation of an explicit finite difference approximation to 3D acoustic wave equation, in a message passing environment. For explicit schemes, the stability condition restricts the size of the time step, which is normally very small. Therefore, the computational time required for calculations become very large. Supercomputers based on vector or parallel architecture are generally used for this purpose.

MATHEMATICAL FORMULATION

The acoustic wave equation in a 3D heterogeneous medium is given by

$$\frac{1}{K}\frac{\partial^2 p}{\partial t^2} = \frac{\partial}{\partial x}\left[\frac{1}{\rho}\frac{\partial p}{\partial x}\right] + \frac{\partial}{\partial y}\left[\frac{1}{\rho}\frac{\partial p}{\partial y}\right] + \frac{\partial}{\partial z}\left[\frac{1}{\rho}\frac{\partial p}{\partial z}\right] \tag{1}$$

where, p is the negative pressure wavefield, ρ is the density and K is the incompressibility.

We divide the 3D geological model into a grid of I X J X K points. In order to obtain finite difference approximation to equations (1), let us introduce a set of indices i, j, k and n such that

$$x = i\ \Delta x \quad i = 0,1,2,......\, I, \qquad y = j\ \Delta y \quad j = 0,1,2,......\, J$$
$$z = k\ \Delta z \quad k = 0,1,2,......\, K, \qquad t = n\ \Delta t \quad n = 0,1,2,......\, N$$

where Δx, Δy and Δz are the grid spacing and I ,J and K are the number of grid points in x- y- and z- directions respectively, Δt is the time step and N is the total number of time steps. Physical parameters, density ρ(i,j,k) and incompressibility K(i,j,k) are specified at each grid point.

Substituting central difference approximations of the derivatives in equation (1), an expression is obtained for calculating the wavefield $p^{n+1}_{i,j,k}$ from the knowledge of the wavefield at previous time levels i.e. $p^{n}_{i,j,k}$ and $p^{n-1}_{i,j,k}$ as

$$\begin{aligned} p^{n+1}_{i.j,k} = {} & 2p^{n}_{i,j,k} - p^{n-1}_{i,j,k} + A_{i,j,k}\,p^{n}_{i-1,j,k} + B_{i,j,k}\,p^{n}_{i+1,j,k} \\ & + D_{i,j,k}\,p^{n}_{i,j-1,k} + E_{i,j,k}\,p^{n}_{i,j+1,k} + F_{i,j,k}\,p^{n}_{i,j,k-1} \\ & + H_{i,j,k}\,p^{n}_{i,j,k+1} - \ (A_{i,j,k} + B_{i,j,k} + D_{i,j,k} \\ & + E_{i,j,k} + F_{i,j,k} + H_{i,j,k})\,p^{n}_{i,j,k} \end{aligned} \tag{2}$$

where *A, B, D, E, F*, and *G* are the functions of physical parameters K and ρ.

Equation (2) is programmed to calculate the wave propagation in heterogeneous media. This approximation is second order accurate both in space and in time. Grid dispersion is minimised by keeping the grid spacing smaller than one tenth of the shortest wavelength. The finite difference approximation (2) is stable if

$$\Delta t \le \frac{min(\Delta x, \Delta y, \Delta z)}{\sqrt{2}\,V_{max}} \tag{3}$$

where $(V = K/\rho)$ and V_{max} is the maximum wave velocity in the medium.

Since a digital computer has finite memory capabilities, we have to restrict the model size to a fixed number of grid points. This introduces artificial boundaries at the edges of the model. In reality the earth is infinite and therefore all the energy impinging on these boundaries must be absorbed. For the finite difference scheme presented here a sponge boundary condition as described by Sochacki et.al. 1987, is used for attenuating the energy impinging on the left, right and bottom edges of the model. To implement sponge boundary condition extra grid points are added to gradually attenuate the energy. The free-surface condition is applied to the top boundary.

IMPLEMENTATION OF THE ALGORITHM

Sequential Algorithm

The algorithm for sequential implementation of 3D wave propagation is as follows:

```
BEGIN
        Setup data structure, variable and constants
        Read velocity model
        Check stability condition
        FOR every time step DO
           FOR every grid point DO
              Evaluate wavefield
           END
        END
END
```

Parallel Algorithm

The most important part of parallel programming is to map out a particular problem on a multiprocessor environment. The problem must be broken down into a set of tasks that can be solved concurrently. The choice of an approach to the problem decomposition depends on the computational scheme. For the second order central difference scheme used here, one can observe that the calculation of the wavefield at a grid point at an advanced time level involves the knowledge of the wavefield at seven grid points of the current time level. Therefore, it is a seven point differencing star. Therefore, if we use a domain decomposition scheme for solving this problem only first order neighbors will be involved in communication for central difference scheme.

Domain Decomposition

The parallel implementation of the algorithm is based on domain decomposition. Domain decomposition involves assigning subdomains of the computational domain to different processors and solving the equations for each subdomain concurrently. The problem domain is a cuboid as shown in the figure 1,

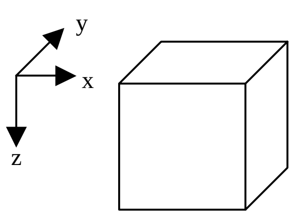


Figure 1: Problem domain (grid point size: I x J x K)

This domain can be partitioned in three ways viz., stripe, hybrid stripe and checkerboard.

Striped Partitioning

In the striped partitioning of the 3D domain, the domain is divided into horizontal or vertical planes, and each processor is assigned one such plane. Striped partition can be done in three ways as shown in figure 2.

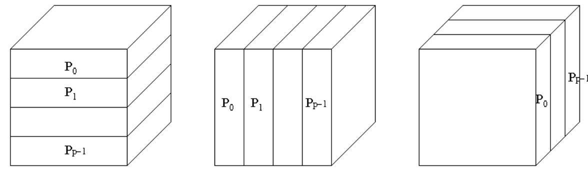


Figure 2: (a) Partition in z- direction, (b) partition in x-direction and (c) partition in y-direction

If we chose partitioning in z – direction then x-y planes have to be distributed among processors, if we chose partition in x-direction, then y-z planes have to be distributed among processors and if we chose partition in y direction x-z planes have to be distributed among processors. For load balancing we divide the domain in equal size of the pizza boxes, depending upon the number of available processors.

Hybrid Stripe Partitioning

In hybrid stripe partitioning, partition is done using combination of two of the striped partitioning as shown in Figure 3.

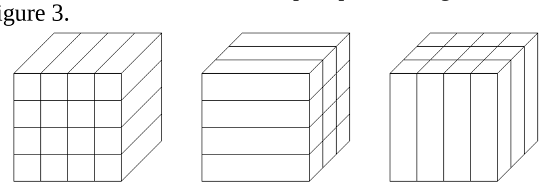


Figure 3: (a) Partitioning in z- & x- directions (b) partitioning in z- and y- directions (c) partitioning in x- and y- directions.

Checkerboard Partitioning

In checkerboard partitioning, domain is divided in all three directions creating smaller subdomains. In uniform checkerboard partitioning, all sudomains are of the same size. These subdomains have to be distributed among processors and no processor gets the complete plane.

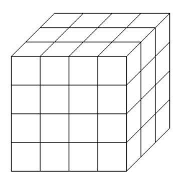
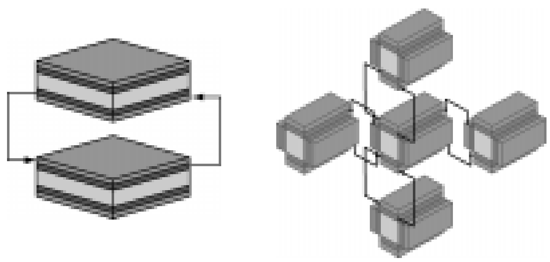


Figure 4: Domain decomposition using checkerboard partitioning.

Interprocessor Communication

In the stripe partitioning as shown in figure 2, each individual processor calculates the wavefield at each grid point in the corresponding subdomain at time k+1, using wavefield values at previous time steps at the same grid point and its adjacent neighbors (depends on finite difference scheme used). The grid points can be updated using finite difference formula, simultaneously in all the subdomains except the grid points on the boundary which require information from the neighboring



(a) (b)

Figure 5: Communication between two adjacent tasks in (a) stripe (b) Hybrid stripe partitioning

processors. In order to calculate the wavefield at the grid points on the subdomain, at each time step, the required boundary grid points should be interchanged between the processors. Figure 5(a), shows the communication pattern in the stripe partitioning. For interchange of grid point, we attach an extra buffer layer (depth depends on the finite difference scheme) with the subdomains. The grid point(s) in the darker zone in the subdomain goes to the lighter zone (grey zone) of the other processor. Thus, in this case the two-way communication is in one direction only. This communication is known as ghost point communication.

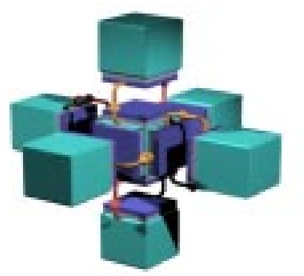


Figure 6: Communication between adjacent tasks in checkerboard partitioning.

In the case of hybrid stripe partitioning, ghost point communication is in two directions i.e. each processor should exchange boundary grid points with its two neighboring processors as shown in the Figure 5(b). While in the case of checkerboard partitioning the ghost point communication is in all three directions, i.e. each processor should exchange the boundary grid points with its six neighbors as shown in Figure 6.

Algorithm

In summary, the parallel algorithm for implementation looks as follows:

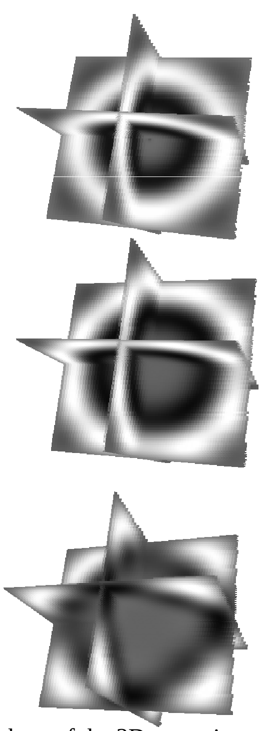
```
BEGIN
    Setup data structures, variables and constants
    Read velocity model
    Setup the domain decomposition
    Send the decomposed subdomain with corresponding
            velocity to different processors
FORALL processors simultaneously DO
    FOR every time step DO
        FOR every grid point DO
            - Evaluated wavefield
        END
        -   Interchange border grid points with each adjacent
            layer(s).
    END
 - Gather wavefield from every processor
END
```

Implementation

Since the algorithm is implemented on distributed computing environment, it is designed using message-passing paradigm. MPI (Message Passing Interface) has been used to communicate between the processors. The present implementation is analogous to a Master-Worker system, where master works as the manager and assigns tasks to his workers. The main job of master is to provide the required data to all the workers and distribute workload properly, so that the idle time of the workers is minimized. Also, at the end, the master collects the completed work from all the workers, compiles it and writes it on the disk in a proper manner.

Numerical Examples

Finite-difference computation of the snapshots can help in our understanding of wave propagation in the medium. We have used a constant velocity model as a numerical example for generating snapshots of 3D acoustic wave propagation. Source is placed at the center of the cubic model. For simplicity sake there is no density variation within the model. However, the algorithm can handle density variations. The source wavelet used for calculation of snapshots is the second derivative of a Gaussian function with a dominant frequency of 30Hz.



t = 0.07 sec

t = 0.075 sec

t = 0.08 sec

Figure 9: Snapshots of the 3D acoustic wave propagation through the constant velocity model.

Performance Analysis

We performed the benchmark test of the parallel algorithm for problem sizes 400 X 400 x 400 and a smaller problem size of 200 X 200 X 200. The grid spacing in all three directions was 2m. A time step of 0.0001sec was used and the wave propagation was carried out for 0.1sec. Since the model size 400 X 400 X 400 is too large to fit into the processor memory of PARAM 10000. The test was performed using minimum of 8 processors for the bigger model.

We have used three types of partitioning for the domain decomposition and have experimented with all the three types. For implementation point of view all three types of partitioning play an important role on the basis of memory access pattern. Theoretically, checkerboard partitioning has the best memory access pattern as the partitioned data can reside in the first level of the cache available. In the case of stripe and hybrid stripe partitioning for access of data from memory may require swapping between first and second levels of cache which is an expensive operation. Hybrid stripe partitioning has better access patter as compared to stripe partitioning. Bar charts for execution time verses number of processors for 3D acoustic wave modeling, shown in figures 10 for two different problem sizes, support this statement.

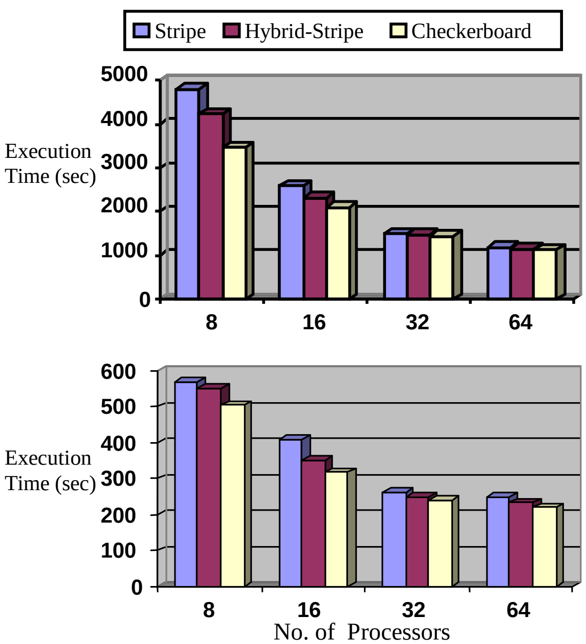


Figure 10: Comparison of execution time for Stripe, Hybrid-Stripe and Checkerboard partitioning for 3-D acoustic wave modeling for two model sizes viz., (a) 400X400X400, (b) 200X200X200

A speedup analysis for the two model sizes shows a sub-linear speedup as we increase the number of processors. For a fixed model size the compute to communication ratio decreases with the increase in the number of processors. Therefore if we increase the size of the problem, better speedup can be achieved for large number of processes.

Conclusion

In this paper we have described the parallel implementation of a finite difference based 3D acoustic wave propagation algorithm. Acoustic wave propagation is being used as forward modeling tool in Seismic Data Processing for Oil Exploration. The codes for these parallel implementations have been written using MPI message passing library. Performance analysis shows that for the domain decomposition implementation, checkerboard partitioning gives the best performance as it has suitable memory access pattern for such problems. Performance analysis also shows that we can achieve a good price performance ratio for smaller size problems on less number of processors and for large size of the problems we have to use large number of processors. We have also found that for the large size of the problems that can not fit into the memory of serial computers, parallel computing is the only solution.

Acknowledgements

Authors wish to thank Department of Science and Technology (Govt. of India) for funding the project and C-DAC, Pune for providing computational facility on PARAM 10000 and permission to publish this work.

References

1. Kelly, K. R., Ward, R. W., Treitel, S., and Alford, R. M., 1976, Synthetic Seismograms – A finite difference approach, Geophysics, 41, 2-27.
2. Phadke, S. and Bhardwaj, D., 1998, Parallel implementation of seismic modelling algorithms on PARAM OpenFrame, Neural, Parallel and Scientific Computation, 6(4), 469-478.
3. Mitchell, A. R. and Griffiths, D. F., 1981, The finite difference method in partial differential equations: John Wiley & Sons Inc.
4. SEG/EAGE 3-D Modeling series No. 1, 1997, 3-D Salt and Over thrust models. SEG publications.
5. Sochacki, J., Kubichek, R., George, J., Fletcher, W. R. and Smithson, S., 1987, Absorbing boundary conditions and surface waves: Geophysics, 52, 60-71
6. Pacheco, Peter S., 1997, Parallel Programming with MPI, Morgan Kaufmann Publishers, Inc, California.
7. Vafidis, A., 1988, Supercomputer finite difference methods for seismic wave propagation: Ph.D. Thesis, University of Alberta, Edmonton, Canada.
8. Villarreal, A., and Scales, J. A., 1997, Distributed three-dimensional modeling of wave propagation in acoustic media, Computers in Physics, 11, 388-399.