3–D SEISMIC IMAGING OF COMPLEX GEOLOGIES

David E. Womble, Sudip S. Dosanjh, John P. VanDyke, Ron A. Oldfield, David S. Greenberg
Sandia National Laboratories
Albuquerque, New Mexico 87185-1110

ABSTRACT

We present three codes for the Intel Paragon that address the problem of three-dimensional seismic imaging of complex geologies. The first code models acoustic wave propagation and can be used to generate data sets to calibrate and validate seismic imaging codes. This code reported the fastest timings for acoustic wave propagation codes at a recent SEG (Society of Exploration Geophysicists) meeting. The second code implements a Kirchhoff method for pre–stack depth migration. Development of this code is almost complete, and preliminary results are presented. The third code implements a wave equation approach to seismic migration and is a Paragon implementation of a code from the ARCO Seismic Benchmark Suite.

INTRODUCTION

The high cost of drilling wells has led the Oil and Gas Industry to invest a lot of resources into computationally imaging the geology of potential sites in the hope of optimizing drill placement. For complex geologies, such as thrusts in mountainous areas and sub–salt structures in and around the Gulf of Mexico, these computational images are difficult to construct and to verify. It is generally accepted that, although much success has been achieved with 2–D data, the use of 3–D seismic data will be required to identify the smaller and deeper features of these geologies (Matthews 1995). However, processing these 3–D data sets requires both tremendous computational resources and the ability to handle large (terabyte) datasets. In this paper we will discuss how MP computers (such as the Intel Paragon) can be used to attack these large problems.

The general approach to imaging geological structures is to start with a guess of the subsurface velocity field, compute the pressure field of an acoustic wave corresponding to this velocity field, and refine the guess of the velocity field. There are two primary computational approaches to computing the pressure field, Kirchhoff methods (ray tracing) and solving the acoustic wave equation. Both of these approaches can be applied to pre–stack or post–stack migration algorithms. Stacking reduces the total amount of computation but inevitably leads to inaccuracies.

One reason that Kirchhoff methods are currently popular is that they can be readily applied to sub–domains of the full 3–D domain, e.g., to a 2–D slice of the domain. By looking at only a small region the amount of computation can be greatly reduced. However, even on sub-domains, it is still desirable to take advantage of the computational power of large scale MP supercomputers since one may wish to process many regions and/or apply iterative refinements to the velocity model.

The acoustic wave equation approach (Stolt and Benson 1986) attempts to gain an increase in accuracy as compared to Kirchhoff methods in return for increased computational time. Typical approaches to solving the wave equations start with a finite difference grid and it is common to transform the equation from the time domain to the frequency domain and make a paraxial approximation. While these computational approximations allow some reduction in computational cost it is still prohibitively time consuming to process data for large surveys on workstation or common vector supercomputers. Further algorithmic refinements, such as amplitude preservation and AVO/inversion promise even greater accuracy, require additional computational power, and increase the incentive to use MP computers.

One problem in developing large seismic processing codes is calibration and validation, especially for codes designed to image complex geologies. Sandia National Labs has participated in a GO–NII project to produce a repository of synthetic data which can be used for this purpose. The result of this project has been to produce a code (running on the Intel Paragon) which simulates the propagation of seismic waves through known geologies thereby creating surfaces traces which would be observed by typical geological surveys.

In the remainder of this paper we discuss the application of parallel computing to these three areas of seismic imaging: synthetic data production, Kirchhoff imaging, and wave equation imaging. Each of the three projects is in a different stage of development. The synthetic data generator has already been used to process

several test geologies and reported the fastest timings for acoustic wave propagation codes at a recent SEG (Society of Exploration Geophysicists) meeting. The Kirchhoff method code for pre–stack depth migration on the Intel Paragon is almost complete, and we present preliminary results here. Our work on acoustic wave equation is in its initial stages. We present here an implementation on the Paragon for post–stack depth migration based on a code from the ARCO Seismic Benchmark Suite (Mosher 1993). We further present an analysis of parallelism techniques to scale pre–stack and post–stack to larg data sets.

ACOUSTIC WAVE PROPAGATION

Validating geological imaging codes by drilling wells is extraordinarily expensive. Although some data from existing wells exists, it is essential to have synthetic data for a variety of geological structures with which to test imaging codes. We describe here a code which takes as input a description of a test geology and produces as output the traces which would have been observed by a surface survey of the site. This is the logical inverse of the typical imaging code which attempts to deduce the subterranean geological formations based on the reflection of acoustic waves from a source to an array of receivers placed on or near the surface. Thus industrial users of our synthetic data can calibrate their codes by comparing their predicted geology to the known geology.

The size of the data (tens of gigabytes) and the number of experiments needed to fully describe the geology make the problem very computationally intensive. Our parallelization of the code involved careful load balancing, and optimizing communication, I/O, and computational performance.

The basis of our parallelism was a domain decomposition approach, i.e. each processor computes its own spatial region. However, the outer boundary of the compute region required an additional "sponge" calculation that serves as an absorbing boundary condition. Consequently, the assignment of equal sized spatial regions to each processor resulted in a large computational load imbalance. By modifying the domain sizes to assign smaller regions to processors whose regions included boundary regions we were able to achieve a 15% improvement in performance.

In order to achieve the desired accuracy we used a finite difference discretization that is tenth order in space and second order in time. These high order equations necessitated the communication of five outer layers of data between neighboring processors. Thus, this code requires more communication than the average finite difference code. Communication became a critical performance factor especially for 3–D codes. We expect that our concurrent research on techniques for overlapping computation with communication will further speed this code in the future.

As with many seismic codes the data files sufficiently large to present challenges to I/O use. In particular, two areas of I/O were problematic: reading and distributing the velocity field, and formatting and storing the output. A typcial full velocity file is 400 to 500 Mbytes and contains velocity information about every grid point on the entire computational model. Each simulation, however, only uses a section of the velocity file. Our algorithm takes each slice along the z axis of this section and distributes pieces of the slice to the appropriate nodes. This process takes four to five minutes to complete, which is roughly 10% of the execution time required for one shot. We avoid rereading data by interchanging loops to calculate data for multiple shots.

The output format was an often debated issue among the industrial organizations involved in the project. Our code has the ability to write data into one of two formats. A record in the first format, SEP (Stanford Exploration Format), contains the pressure recorded by each receiver at any one time step. A record in the second format, enhanced SEP, contains twenty-five header values followed by the pressure values recorded at every time step for that receiver. Neither format includes data compression.

Inevitably, parallel performance is limitted by sequential performance on the individual nodes. We modified the compiler–produced assembly code for the main kernel to take advantage of the dual instruction mode and the linked multiply–add units available in the i860XP. This resulted in close to a 50% increase in performance for that section of the code, and a 25% decrease in the overall execution time for the simulation.

Table 1 shows the performance of our acoustic wave propatation code, and Figure 1 shows the speedup. The benchmark used to measure the performance of our code is a single shot, 2,450 time step simulation of the overthrust model on a $400 \times 400 \times 186$ grid. On a 500–node Intel Paragon running the SUNMOS operating system (Wheat 1994), it executes at 14.7 Mflops/sec/node and completes in 25.1 minutes. Our reported time of 31 minutes at an SEG meeting in October was the fastest time for the benchmark.

3–D PRE-STACK KIRCHHOFF DEPTH MIGRATION

As mentioned in the introduction pre-stack Kirchhoff depth migration is a popular technique to process the data from a field survey to yield a seismic image of the earth. The essence of the Kirchhoff imaging is to distribute energy recorded for a trace at a given time over all points of potential reflection that correspond to that travel time. The imaging results because the contributions from various traces interfere constructively at points of actual reflection and tend to cancel out at other points.

Table 1: Run times and estimated speedup for an overthrust model with a $400 \times 400 \times 186$ grid and 50 time steps. (Speedup and efficiency are computed based on an estimated single processor timing.)

p	Time (sec)	Speedup (est.)	Efficiency (est.)
1	7910	1	1.0
75	122.78	64.4	.85
108	95.4	82.9	.76
147	79.1	100.0	.68
196	69.1	114.5	.58
224	57.4	137.8	.62
256	49.7	159.15	.62
288	48.8	162.1	.56
320	44.8	176.6	.55
405	37.9	208.7	.52
450	40.1	197.3	.44
500	33.2	238.3	.47

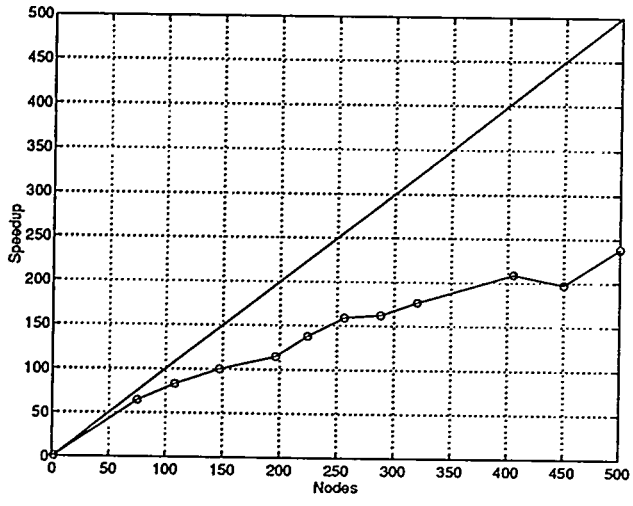

Figure 1: Speedup for an overthrust model with a $400 \times 400 \times 186$ grid and 50 time steps.

The code described here was developed in conjunction with the ORYX Energy Company and the University of Texas at Dallas starting from an Intel iPSC-860 code (Epili 1994). The algorithm constructs an image in two distinct steps. First, for every point in a grid defined on the image plane, the travel time is computed by ray tracing to source and receiver locations. The result is a time map for each unique surface source/receiver location (SID) telling the propagation time to each possible internal reflection point. Each of these maps is the same size as the image grid and must be saved for the second half of the calculation.

The computation time to generate the time maps grows as the number of grid points being calculated. However, the time for computing the path for a single ray is also dependent on many other factors that are not easily predicted. The time required for ray tracing, which is done using adaptations of the iterative technique of Um and Thurber (Um 1987), depends very strongly on the complexity of the velocity model and on the accuracy demanded. This variation in time leads to a potential load balancing problem. On the Paragon, the parallel efficiency of this step was greatly enhanced by using a server/worker model in which one node (the server) distributes SIDs to all the other nodes (the workers) as they are ready rather than statically distributing an SIDs to each node.

The second half of the calculation reads the seismic data and constructs the image. For each trace and grid point the sum of the time in the source map and the time in the receiver map for point is the travel time from source to receiver if reflection occurred at that point on the image grid. The image is then accumulated by looping over the grid and using the value from the time map as an index to locate the value from the trace to be used at that point in the image. For a real data set consisting of over 6 million traces (46.5 Gigabytes), distributed over 48 files on the Paragon PFS file system on 48 RAID disks units, the time to read the data set once is between 800 and 900 seconds, when there is relatively little disk activity from other Paragon jobs. In cases of interest examined to date, time for ray tracing significantly exceeds that. Unfortunately, these two processing steps, one compute bound and one I/O bound, occur in series, rather than overlapped. We intend to continue work on techniques to allow overlap of computation and I/O.

The intermediate data set of time maps for a single full image can be larger than the seismic data set. Due to the necessity of repeated random access to the time maps it is highly desirable to hold them in node memory rather than on disk. The 1824 node Paragon has a total of about 37 Gigabytes of memory and the use of the SUNMOS node operating system allowed a significant fraction of the memory to be used for time map storage. An improved method of using the disks to supplement the internal memory will be needed for processing of

even larger problems.

One technique we applied to reduce the I/O bottleneck was to assign a node as data manager and I/O buffer for each of the 48 PFS units. These nodes then distribute the traces to the compute nodes as the compute nodes are ready. Although many vendors have proposed I/O systems which should supply these functions we found that only by explicitly managing the I/O ourselves could we avoid I/O snarls.

Our present code demonstrates that it is possible to do 3-D pre-stack depth migration on real seismic data in a reasonable amount of time on a parallel machine. However, it also reveals several deffeciencies in the current infrastructure for MP machines. I/O is still cumbersome and it is very difficult to overlap I/O and/or communication with computation.

ANALYSIS OF FINITE DIFFERENCE METHODS AND PRELIMINARY RESULTS

A commonly used form of the 3–D acoustic wave equation for seismic imaging is

$$\frac{\partial P}{\partial z} = \frac{i\omega}{v}\sqrt{1+\frac{v^2}{\omega^2}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)}P, \qquad (1)$$

where P is the pressure field and $v = v(x, y, z)$ is the velocity field. This equation is in the frequency–space domain, neglects spatial derivatives of the velocity and models upcoming waves (Li 1991). The solution to this equation is obtained by advancing each frequency independently down through consecutive z levels and solving a 2–D elliptic PDE at each level.

The two aspects of parallelism with which we will be concerned in this analysis are distribution of storage and distribution of computations. We will assume that there are n_ω frequencies and n_x, n_y and n_z points in the x, y and z directions respectively. We will also assume that there are p_ω, p_x and p_y processors assigned to the ω, x and y directions respectively. Thus, the total number of processors is $p = p_\omega p_x p_y$. We note that algorithms exist for solving on multiple z levels in parallel (Womble 1990), but these will not be considered here.

Using this notation, the total storage required for the solution and the velocity profile on each processor is

$$B_p = 16\frac{n_x}{p_x}\frac{n_y}{p_y}\frac{n_\omega}{p_\omega} + 4\frac{n_x}{p_x}\frac{n_y}{p_y}.$$

Note that the storage does not depend on the n_z as only the solution on the current level and the previous level must be stored. Note also that the storage of the velocity field (the second term above) cannot be spread over processors in the ω direction. Thus, the total storage,

$$B = 16n_x n_y n_\omega + 4n_x n_y p_\omega,$$

grows linearly with p_ω, and even though extracting parallelism in the frequency direction is trivial, we do not have enough memory to set $n_x = n_y = 1$ and $p_\omega = p$ for a large problem on an MP machine.

It has just been observed that "frequency parallelism" is trivial, which is a result of the fact that the solution to Eq. 1 can be computed independently for each frequencies. Because memory requirements make this impractical on an MP machine, we consider the spatial decomposition of computations.

For each value of z, an we must solve an elliptic PDE, and we see from Eq. (1) that this equation is nonlinear. We can reduce this to a sequence of linear PDEs by a partial fraction expansion of which one or two terms are typically used (Li 1991). There are many algorithms available for the solution of the resulting linear, elliptic PDE. The ARCO Seismic Benchmark Suite incorporates an alternating direction implicit (ADI) method consisting of a tridiagonal solve along each line in the x direction followed by a tridiagonal solve along each line in the y direction. There is limited parallelism available in a tridiagonal solve. Hence, we analyze the ARCO Seismic Benchmark Suite as a worst case.

Based on the flops computed in the ARCO Seismic Benchmark Suite, the computational time required by each processor for one step of the downward continuation is

$$T_{comp} = \frac{240 n_x n_y n_\omega}{p_x p_y p_\omega}\gamma,$$

where γ is the time required for one floating point operation.

There are many tridiagonal solves: for each frequency we have one solve for each line in the x and y directions, or

$$N_{solves} = \left(\frac{n_x}{p_x}+\frac{n_y}{p_y}\right)\frac{n_\omega}{p_\omega}$$

To reduce the number of "small" messages, we group b tridiagonal solves into a block. The resulting communication time for each processor (sends and receives) is

$$T_{comm} = \frac{n_\omega}{p_\omega}\left(\frac{n_y}{p_y}+\frac{n_x}{p_x}\right)\left(\frac{4}{b}\alpha + 48\beta\right),$$

where α is the message latency, and β is the time required to send one byte.

The tridiagonal solves proceed left to right (or right to left) along each line, and these blocks can be "pipelined" through the processors that are participating in the solve. In addition to the communications time on each processor, there is an idle time while the pipeline is being filled and emptied. This idle time is

$$T_{pipe} = 120b\left(n_x + n_y\right)\gamma + \left(p_x + p_y\right)\left(2\alpha + 32b\beta\right).$$

The parallel overhead is

$$T_{over} = T_{comm} + T_{idle}.$$

If we have an equal number of points in the x and y directions ($n_x = n_y$), the overhead is minimized for $p_x = p_y$,

$$T_{over} = \frac{1}{b}\frac{n_x n_\omega}{p_x p_\omega}(8\alpha + 96b\beta) + 4p_x(\alpha + 16b\beta) + 140 n_x b\gamma.$$

Now differentiating with respect to b, we find the the overhead is minimized for

$$b_{min} = \sqrt{\frac{\frac{n_x n_\omega}{p_x p_\omega}\alpha}{30 n_x \gamma + 2p_x\beta}}.$$

Finally, speedup is given by

$$S = \frac{pT_{comp}}{T_{comp} + T_{over}}.$$

On an Intel Paragon, the parameters α, β and γ are given by

$$\begin{aligned} \alpha &= 40 \times 10^{-6}, \\ \beta &= 2 \times 10^{-8}, \\ \gamma &= 3 \times 10^{-8}. \end{aligned}$$

With these parameter, we find that for a large problem with $n_x = n_y = 2000$, $n_\omega = 1000$ and $p_x = p_y = 32$ and $p_\omega = 1$ ($p = 1,024$), we have $b_{min} = 45$ and $S = 968$ and a parallel efficiency of 95%. We conclude that there is tremendous parallel potential for large seismic processing codes on MP machines.

The ARCO Seismic Benchmark Suite currently takes advantage of "frequency parallelism" only ($p_x = p_y = 1$ and $p_\omega = p$). As pointed out earlier in this section, there can be substantial memory overhead in this approach caused by repeated storage of the velocity field. Table 2 shows the run times and speedup for a "small," 2–D problem on the Intel Paragon under the SUNMOS operating system. The speedup shown is based on the 16 processor timing.

Table 2: Run time and speedup for a problem from the ARCO Seismic Benchark Suite. Speedup is based on the 16-processor timing, and $n_x = 600$, $n_z = 400$, $n_\omega = 206$.

p	Time	Speedup
16	103.6	1.00
32	59.1	1.75
64	34.0	3.04
128	21.5	4.81

SUMMARY

Many sub-problems in the imaging of geological sites for drill placement will require very large computational and data storage resources. It is, however, possible to apply MP techniques to these problems. Although it is often straightforward to decompose these problems into parallel tasks one must be careful to properly load balance the computation since the individual tasks are often complex and highly variable in their compute requirements. In addition the very large size of the data sets involved and the often high-order equations used leads to high volumes of inter-processor communication and I/O. Careful consideration of task layout and data movement can allow initial implementations to proceed but better techniques for overlapping communication and I/O with computation and for sharing of geology data amoung processors will be needed before MP implementations will be ready for production use.

ACKNOWLEDGEMENTS

The authors would like to thank Ted Barragy of Intel Scalable Systems Division and Chuck Mosher of ARCO for their help in using the ARCO Seismic Benchmark Suite.

This work was supported by the Applied Mathematical Sciences program, U.S. Department of Energy, and was performed at Sandia National Laboratories, operated by the U.S. Department of Energy under contract DE–AC04–94AL85000.

REFERENCES

Claerbout, J.F., 1985. *Imaging the Earth's Interior.* Blackwell Scientific publications.

Epili, D. and G. McMechan, 1994 "Target Oriented 3–D Prestack Kirchhoff Depth Migration on the Intel iPSC860: User's Manual." Center for Lithospheric Studies, University of Texas at Dallas, technical report.

Li, Z., 1991. "Compensating Finite–difference Errors in 3–D Migration and Modeling." Geophysics 56, no. 10 (October): 1650–1660.

Matthews, W., 1995. "Independents Step Up Use of Onshore 3D Seismic Surveys." Oil & Gas Journal (Jan 2): 16-20.

Mosher, C. and S. Hassanzadeh, 1993. "ARCO Seismic Processing Performance Evaluation Suite: Sies 1.0 Users' Guide." ARCO technical report.

Stolt, R. H., and A. K. Benson, 1986. *Seismic Migration: Theory and Practice.* Geophysical Press, London.

Um, J. and C. Thurber, 1987. "A Fast Algorithm for Two–point seismic Ray Tracing." Bulletin of Seismological Society of America 77: 972–986.

Wheat, S. R., A. B. Maccabe, R. E. Riesen, D. W. van Dresser, and T. M. Stallcup, 1994. "PUMA: An Operating System for Massively Parallel Systems." Proceedings of the Twenty–Seventh Annual Hawaii International Conference on System Sciences, IEEE Computer Society Press: 56–65.

Womble, D. E., 1990. "A Time–stepping Algorithm for Parallel Computers." SIAM Journal on Scientific and Statistical Computing 11, no. 5 (September): 824–837.

Yilmaz, O., 1987. *Seismic Data Processing.* Society of Exploration Geophysicists, Tulsa, OK.