3D Seismic Imaging on Massively Parallel Computers [1]

David E. Womble, Curtis C. Ober, and Ron Oldfield
Applied and Numerical Mathematics Department
Sandia National Laboratories
Albuquerque, NM 87185-1110, U.S.A.
Voice phone: (505) 845–7471, Fax phone: (505) 845–7442
dewombl@cs.sandia.gov (email)

Summary

The ability to image complex geologies such as salt domes in the Gulf of Mexico and thrusts in mountainous regions is a key to reducing the risk and cost associated with oil and gas exploration. Imaging these structures, however, is computationally expensive. Datasets can be terabytes in size, and the processing time required for the multiple iterations needed to produce a velocity model can take months, even with the massively parallel computers available today. Some algorithms, such as 3D, finite–difference, prestack, depth migration remain beyond the capacity of production seismic processing. Massively parallel processors (MPPs) and algorithms research are the tools that will enable this project to provide new seismic processing capabilities to the oil and gas industry. The goals of this work are to

- develop finite–difference algorithms for 3D, prestack, depth migration;
- develop efficient computational approaches for seismic imaging and for processing terabyte datasets on massively parallel computers;
- develop a modular, portable, seismic imaging code.

In our work, we have investigated the imaging capabilities of a range of finite-difference imaging algorithms; we have investigated the potential parallelism of these algorithms and developed efficient parallel versions of several key computational kernels; and we have designed and tested several approaches to handling

[1] This work was supported by the United States Department of Energy (DOE) under Contract DE–AC04–94AL85000 and by DOE's Mathematics, Information and Computational Sciences Program. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company.

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

the massive amount of I/O.

We have incorporated this algorithmic work into a 3D, finite–difference, prestack, depth migration code called Salvo. Salvo has been validated for a range of problems, including the SEG/EAEG overthrust model, SEG/EAEG salt model and real data from a Gulf of Mexico survey.

Salvo runs efficiently on several MPP platforms, including the Cray T3D and T3E, the Intel Paragon and the IBM SP2. It also runs on several symmetric multiprocessor (SMP) platforms including the SGI Power Challenge and DEC AlphaServer and on a network of workstations. Salvo has a JAVA interface for remote processing and viewing output.

In this paper, we will discuss the design of Salvo, focusing on the communication structure and design of the I/O subsystem. We will also discuss the performance of Salvo for a range of MPP and SMP platforms.

1. Introduction. A key to reducing the risks and costs of associated with oil and gas exploration is the fast, accurate imaging of complex geologies. Prestack depth migration generally yields the most accurate images, and one approach to this is to solve the scalar wave equation using finite differences. As part of an ongoing Advanced Computational Technologies Initative (ACTI) project, a finite–difference, 3-D prestack, depth–migration code for a range of platforms has been developed. The goal of this work is to demonstrate that massively parallel computers (thousands of processors) can be used efficiently for seismic imaging, and that sufficient computing power exists (or soon will exist) to make finite–difference, prestack, depth migration practical for oil and gas exploration.

Several problems have been addressed to obtain an efficient code. These include efficient I/O, efficient parallel tridiagonal solves, and high single-node performance. Furthermore, portability considerations have restricted the code to the use of high-level programming languages and interprocessor communications using MPI.

Efficient I/O is one of the problems that have been addressed. The initial input to our seismic imaging code is a sequence of seismic traces, which are scattered across all the raid systems in the I/O subsystem and may or may not be in any particular order. The traces must be read, Fourier transformed and redistributed to the appropriate processors for computation. In Salvo, the input is performed by a subset of the nodes, while the remaining nodes perform the pre-computations in the background.

A second problem that has been addressed is the efficient use of thousands of processors. There are a couple types of parallelism available in a finite–difference solution of the wave equation for seismic imaging. The first and most obvious is frequency parallelism; however, this limits the available parallelism to hundreds of processors and restricts the size of problem that can be solved in-core. Spatial parallelism addresses both of these problems, but introduces another issue. Specifically, an alternating direction implicit (ADI) method

(or a variant) is typically used for the solution at each depth level, which means that tridiagonal solves must be parallelized. Parallelizing individual tridiagonal solves is difficult, so the problem has been handled by pipelining many tridiagonal solves.

The remainder of this paper describes in more detail the seismic imaging algorithms, the computational problem and implementation used in Salvo and presents some numerical results.

2. The imaging algorithm. The following development is an industry-standard approach and can be found in [Claerbout 1985, Yilmaz 1987, Li 1991]. It is repeated here for reference. The equation used to model the propagation of pressure waves through the earth is the scalar wave equation,

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} + \frac{\partial^2 P}{\partial z^2} = \frac{1}{v^2}\frac{\partial^2 P}{\partial t^2}, \tag{1}$$

where $P(x, y, z, t)$ is pressure, and $v(x, y, z)$ is the acoustic velocity of the media. This equation is transformed to a Helmholtz equation and then to the paraxial wave equation,

$$\frac{\partial P}{\partial z} = \left\{ \pm \frac{i\omega}{v} \left[1 + \frac{v^2}{\omega^2} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \right]^{1/2} \right\} P, \tag{2}$$

where ω is the frequency of the propagating wave. The positive and negative signs correspond to upcoming and downgoing wave fields.

The evaluation of the square-root operator is numerically difficult, hence it is approximated by a series that has its origin in a continued–fraction expansion [Claerbout 1985, p. 84] [Yilmaz 1987, p. 513]. The continued–fraction expansion can be represented by ratios of polynomials [Ma 1981] and the polynomial coefficients can be optimized for propagation angle [Lee and Suh 1985]. With these approximations, the paraxial wave equation can be written as

$$\frac{\partial P}{\partial z} = \pm \frac{i\omega}{v} \left[1 + \sum_{\ell=1}^{m} \frac{\alpha_\ell S}{1 + \beta_\ell S} \right] P, \tag{3}$$

where

$$S = S_x + S_y = \frac{v^2}{\omega^2}\frac{\partial^2}{\partial x^2} + \frac{v^2}{\omega^2}\frac{\partial^2}{\partial y^2}, \tag{4}$$

and α_ℓ and β_ℓ are the expansion coefficients [Lee and Suh 1985]. Eq. (3) is approximated by

$$\frac{\partial P}{\partial z} = \pm \frac{i\omega}{v} \left[1 + \sum_{\ell=1}^{m} \frac{\alpha_\ell S_x}{1 + \beta_\ell S_x} + \frac{\alpha_\ell S_y}{1 + \beta_\ell S_y} \right] P. \tag{5}$$

For most practical work, only the first two terms are retained, and the equation is split into a sequence of three equations by the method of fractional steps. The sequence of equations that must be solved is thus

$$\frac{\partial P_0}{\partial z} = \pm \frac{i\omega}{v} P_0,$$

$$
\begin{aligned}
\frac{\partial P_1}{\partial z} &= \pm\frac{i\omega}{v}\frac{\alpha_1 S_x}{1+\beta_1 S_x}P_1, \\
\frac{\partial P_2}{\partial z} &= \pm\frac{i\omega}{v}\frac{\alpha_1 S_y}{1+\beta_1 S_y}P_2.
\end{aligned} \tag{6}
$$

To obtain an image that is reasonably accurate for wave propagating at angles up to 65°, we use $\alpha_1 = 0.478242$ and $\beta_1 = 0.376370$. Other approximations are obtained by choosing different values for α and β and/or by retaining more terms in (3) [Yilmaz 1987].

There are many sources of error introduced by solving the sequence of equations (6) instead of Eq. (2). Several filters have been introduced to correct the errors (or a subset of the errors). One of these was introduced in [Li 1991], in an attempt to correct all the errors after a transformation into Fourier space. The primary computational operation in the Li filter is the FFT and the correction involves assumptions about boundary conditions and the velocity model. Another filter was introduced in [Graves and Clayton 1990] in an attempt to correct the error introduced by the approximations used to derive Eq. (5). The primary computational operation in this filter is the solution of a tridiagonal linear system.

The boundary conditions must be chosen so that no waves are reflected back into the domain. Several possible absorbing boundary conditions are given in [Clayton and Engquist 1980]. The best of these is the boundary condition

$$
i\frac{\partial P}{\partial z} + c\frac{v}{\omega}\frac{\partial^2 P}{\partial x \partial z} = \mp\left(a\frac{\omega}{v}P - ib\frac{\partial P}{\partial x}\right) \tag{7}
$$

where

$$
\begin{aligned}
a &= 1 \\
b &= \begin{cases} 1, & \text{left boundaries} \\ -1, & \text{right boundaries} \end{cases} \\
c &= \begin{cases} 2-2/\sqrt{3}, & \text{left boundaries} \\ 2/\sqrt{3}-2, & \text{right boundaries} \end{cases}
\end{aligned}
$$

In phase–space, this is a hyperbola fit to the circle of the wave equation. The use of first order differences in the boundary conditions maintains the tridiagonal structure of the resulting linear system. Salvo also includes absorbing boundary conditions based on a Padé approximation [Xu 1996].

3. The computational problem. A seismic image can be computed using the equations above by the following high-level algorithm.

```
Begin migration
    Read source and receiver traces measured on the surface
    Transform traces to frequency domain using an FFT
    For each depth level
        Read velocity
        For each frequency being migrated
            Migrate source and receiver traces one depth level
        End for
        Combine source and receiver data to create an image
        Write image
    End for
End migration
```

The computational "problem" to a large extent is based on the volume of data to be processed, i.e., the size of the problem. A typical marine seismic survey consists of between 10,000 and 100,000 shots. For each shot, there are between 1500 and 3500 receiver traces, and receiver traces typically contain 2500 samples. Thus, the input dataset can contain over 10 megabytes of data for each shot and over 1 terabyte (10^{12} bytes) of data for the whole survey. The specific numbers for land surveys are different, but the survey dataset can still be very large.

A typical processing grid for finite–difference, prestack migration is $500 \times 200 \times 1000$ points, and up to 1000 frequencies are migrated for both shot and receiver fields for a total of 2×10^{11} grid points in the (x, y, z, ω) comutational domain.

It is clear that a lot of data must be read, processed and written. In the following sections, we examine issues related to I/O and processing speed. The discussion below is based on our implementation of finite–difference, prestack, depth migration in a code called Salvo.

4. I/O. I/O is a large part of seismic imaging and some effort must be made to do it efficiently. There are two parts to the I/O problem in finite–difference migration, the initial input in which data is read and transformed to the frequency domain for processing, and the read (velocity)–write (image) pair that occurs at every depth step. (Additional I/O is required if the computations do not fit in the computers main memory, but we do not examine that issue here.) We describe our approach to both parts below, but there are several general principles that should be adhered to in all cases.

- The amount of I/O should be minimized.

- interprocessor communication resulting from I/O should be minimized.
- The I/O bandwidth should be maximized.
- I/O and computation should be overlapped.

The time required to read the initial seismic data, read the velocity models and write the images can be substantial. In Salvo, the effect of the "I/O bottleneck" is mitigated by performing preliminary computations and data redistribution using nodes not directly involved in the I/O.

The trace dataset is distributed across many disks to increase the total disk-to-memory bandwidth. A subset of the available nodes is assigned to handle the I/O. Each of these I/O nodes is assigned to handle I/O from one file system.

The remaining nodes, termed compute nodes, can complete computations and communications necessary before the migration can begin. Each compute node is assigned an I/O node, and performs the pre-computations on the data read by its I/O node. Currently the pre-computation comprises fast Fourier transforms (FFTs), but other computations could also be performed. If we assign a sufficient number of compute nodes to each I/O node, the time to read a block of seismic data will be greater than the time required to compute the FFTs and distribute the frequencies to the correct nodes for later computations. Thus, the computation time will be hidden behind the I/O time.

A model of the I/O and pre-computations and communications can be developed to determine the proper balance between I/O nodes and compute nodes. The I/O node begins by reading a block of data from a disk and distributing this data to a set of compute nodes. The time required for this operation is approximately

$$\Phi b + c\left[\alpha + \beta\left(\frac{b}{c}\right)\right],$$

where Φ is the disk bandwidth, b is the blocksize, α is communication latency, β is the time to communicate one byte, and c is the number of compute nodes.

After completing an FFT, the compute node must distribute each frequency to the processor assigned to perform the seismic migration for that x and y location and frequency. The time to evenly distribute the frequencies of one trace is approximated by

$$p_\omega\left[\alpha + \beta\left(\frac{ng}{p_\omega}\right)\right],$$

where p_ω is the number of nodes at a specific x and y location, that is, the number of nodes in the frequency decomposition, n is the number of words in a frequency trace, g is the size of one word of data ($g = 8$ for single precision, complex numbers). The total time required to FFT the traces and redistribute frequencies

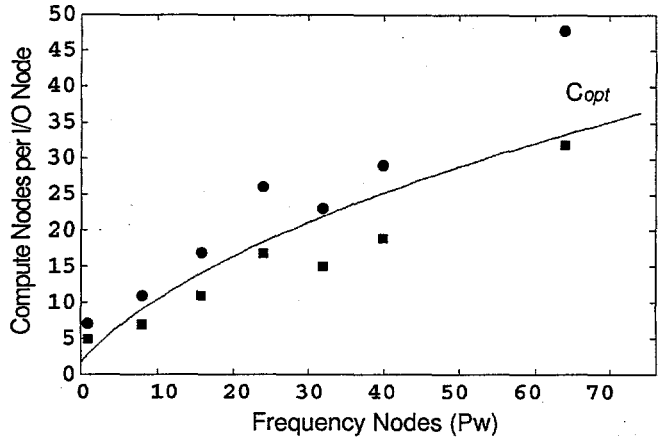


FIG. 1. *The graph shows c_{opt} as a function of p_ω. Circles correspond to actual runs in which I/O nodes had no idle time; squares correspond to actual runs in which I/O node were idle for part of the run.*

for $b/(c\,n\,g)$ traces, (*i.e.*, the number of traces which one compute node processes) is approximately

$$\frac{b}{c\,n\,g}\left\{p_\omega\left[\alpha+\beta\left(\frac{ng}{p_\omega}\right)\right]+\tau\right\},$$

where τ is the time to compute an FFT. τ is machine and library dependent, and because τ can be measured easily on most platforms, it is not further decomposed into computational rates.

To determine the minimum number of compute nodes for each I/O node, c_{opt}, the time required to read and distribute a block of data must be equal to or greater than the time required to FFT the time traces and redistribute frequencies. This yields

$$c_{opt}=\frac{-b\,(\Phi+\beta)+\sqrt{\kappa}}{2\alpha}, \tag{8}$$

where

$$\kappa=(\Phi b+\beta b)^2+4\alpha\left\{p_\omega\left[\alpha+\beta\left(\frac{ng}{p_\omega}\right)\right]+\tau\right\}.$$

All of the variables in the expression for c_{opt}, except p_ω, are either machine constants or defined by the problem size. Figure 1 shows the c_{opt} as a function of p_ω and points indicating several "real" runs. We see that the model does a good job of predicting whether the run time is dominated by disk reads or by computation and communication.

An I/O partition is also maintained during the computational phase to handle reading the velocity data and writing the image data. Here, the computations are expected to be the bottleneck, and I/O is expected to occur in the background; however, the basic principle of the I/O partition remains the same.

The velocity data is needed by every compute processor. The I/O processors coordinate the reading of the velocity data and send the next velocity level to a buffer on each compute processor while the current migration step is in process. Also, since the velocity model is often given on a grid different from the computational grid, the I/O nodes can perform an interpolation/extrapolation operation in the background.

The image write is similar to the velocity read. The image is collected at the end of the current migration step, and written to disk while the next migration step is in process. As in the case of the velocity read, there are computations that can be performed in the background. For example, image points can be stacked into a cummulative image or binned by offset into several images.

5. Computational speed. The biggest improvements in computational speed are achieved by choosing algorithms that reduce the number of floating point operations. However, once the algorithm is chosen, the goal must be to perform these operations as efficiently as possible. Several techniques can be used to improve computational efficiency on each processor, some of which are geared towards today's superscaler architectures.

- Use vendor–optimized libraries when possible. For example, an important operation in seismic processing is the FFT, and most vendors provide several optimized FFT routines.
- Choose array sizes and the order of array dimensions to improve cache efficiency. For example strides whose length are a power of two can seriously degrade cache performance.
- Combine source and receiver migrations to reduce redundent operations. The basic migration operation for source and receiver fields is the same, and coefficients can be calculated once but used twice.
- Arrange loops to increase the number of floating point operations in relation to memory accesses.

We must also consider the parallel efficiency in addition to the processor efficiency. This generally involves reducing the amount of communication between processors and structuring this communications to reduce latencies.

There are two types of parallelism in the seismic imaging algorithm described above, frequency parallelism and spatial parallelism. In frequency parallelism, each processor migrates a subset of the frequencies. There is relatively little communication: the velocity model must be distributed to all processors at the beginning of a migration step, and the image is constructed by summing pressure fields across all processors. The problems are that the number of processors must be smaller than the number of frequencies being processed and that each processor must store the full velocity model. These problems are mitigated by spatial parallelism in which the (x, y) domain is distributed among processors. The problem here is that the algorithm described above involves many tridiagonal solves in the x and y directions that must be parallelized.

We have taken advantage of both types of parallelism, and our solution to the problem of tridiagonal solves is described below.

It is difficult to parallelize the solution of a single tridiagonal system, but this difficulty is offset because there are many such systems. Salvo takes advantage of this by setting up a pipeline. That is, in the first stage of the pipeline, processor one starts a tridiagonal solve. In the second stage of the pipeline, processor two continues the first tridiagonal solve, while processor one starts a second tridiagonal solve. This process continues until all processors are busy.

In the implementation of a pipeline, there are two sources of parallel inefficiency. The first is communication between processors. This communication time is dominated by the message latency since very small amounts of data must be transferred. This can be offset by grouping several tridiagonal solves into each stage of the pipeline.

The second source of parallel inefficiency is processor idle time associated with the pipeline being filled or emptied. This is dominated by the computation time of each pipeline stage. It can be reduced by reducing the computation time, but it is increased by grouping several tridiagonal solves in each stage of the pipeline.

The total parallel overhead can be minimized by choosing the number of tridiagonal solves that are grouped into each stage of the pipeline. The optimal number of tridiagonal solves to group is based on the following model. The communication time is approximated by

$$T_{comm} = N\left(2\frac{\alpha}{b} + 24\beta\right),$$

where N is the total number of tridiagonal solves, b is the number to be grouped into each stage of the pipeline, α is the communication latency, and β is time to communicate one byte. The pipeline idle time is approximated by

$$T_{pipe} = W\,b\,n\,\gamma + p\,(2\alpha + 24b\beta),$$

where W is the total number of floating point operations required at each grid point, n is the number of points in each stage of the pipeline, p is the number of processors in the pipeline, and γ is the computational time required for one floating point operation.

The value of b that minimizes the total overhead, b_{min} is computed by summing T_{comm} and T_{pipe}, and minimizing. This yields

$$b_{min} = \left(\frac{2\,N\,\alpha}{W\,n\,\gamma + 24\,p\,\beta}\right)^{1/2}.$$

We have found this model to be quite accurate, and all results presented later in this paper use this value of b_{min}.

6. Results. To validate Salvo, several tests were performed to ensure accurate imaging of reflecting layers. The problems selected for the test cases include a simple impulse response from a hemispherical reflector, the ARCO–French Model [French 1974], the Marmousi model, the SEG/EAEG 3D overthrust model [Aminzadeh *et al.* 1994], the SEG/EAEG 3D salt model, and real data.

As an example of a Salvo migration, we show the 3D SEG/EAEG salt model. This is a synthetic model with synthetic receiver data available through the SEG internet home page at http://www.seg.com. Figure 2(a) shows a corner–cut view of the model. The grayscale colormap indicates the speed of sound in a region; the lighter a region, the higher the speed. The white region is the salt dome. Figure 2(b) shows the same corner–cut view of the image produced by Salvo. The image is $600 \times 600 \times 210$. It is a stack of 45 shots, each processed on a $200 \times 200 \times 210$ grid with a surface grid fully populated with receivers. For this image $\Delta x = \Delta y = \Delta z = 20$ meters. The receiver traces contain 626 samples with $\Delta t = 0.004$ seconds, and 511 frequencies were migrated. Some additional postprocessing of the images has been performed by ARCO and Oryx Energy Company.

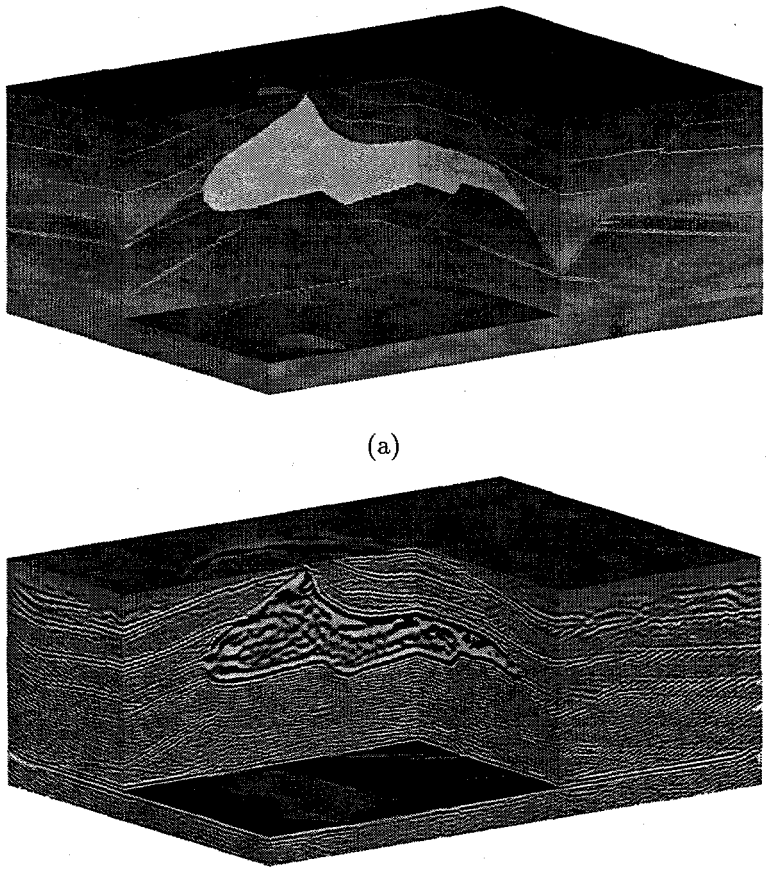
To test the computational performance of Salvo, the sample impulse problem was run on the Intel Paragon. The spatial size of the impulse problem has been adjusted so that each processor has approximately a 101×101 spatial grid. Sixty-four frequencies have been retained for the solution independent of how many frequency processors were used.

Timings for the sample impulse run are shown in Table 1. From these numbers, we can make a few statements about the parallelism of the migration routine. First, the spatial parallelism is very efficient as soon as the pipeline is fully utilized (after $3 \times 3 \times 1$ processor mesh). However there is a penalty for introducing the pipeline in each direction, which is about 10% for each (*i.e.*, $1 \times 1 \times 1$ at 100% to 91% for $2 \times 1 \times 1$, and to 81% for $2 \times 2 \times 1$). The origins of this "overhead" is still under investigation.

Second, the frequency parallelism is very efficient, staying in the upper 90's for most of the problems. This is expected, since frequency parallelism requires little communication during the solve. The primary communications are a broadcast of velocity data at the beginning of each depth step and a summation to produce an image at the end of each depth step.

For an additional timing figure, we ran a $2000 \times 2000 \times 1000$ impulse problem with 140 frequencies on a 1,792–node Intel Paragon. Each node processed a 250×250, x–y subdomain with 5 frequencies. The total run time was 7 hrs., 51 mins, which corresponds to a computational rate of 22 Mflops/second/node.

7. Conclusions. In this paper, an implementation of a wave-equation-based, finite–difference, prestack, depth migration code for MPP computers has been presented. The results of several test runs were presented to show the accuracy of the code. Also, timing results and performance models have been presented to show



(a)

(b)

FIG. 2. *A corner-cut view of the SEG/EAEG 3D salt model and the same corner cut view of the Salvo image produced by processing 45 shots.*

that the code can be tuned to run efficiently on MPP computers.

REFERENCES

[Aminzadeh *et al.* 1994] Aminzadeh, F.; N. Burkhard; L. Nicoletis; F. Rocca; K. Wyatt. 1994. "SEG/EAEG 3-D Modeling Project: 2nd Update." *Leading Edge.* September, 949–952.

[Bunks 1995] Bunks, C. 1995. "Effective Filtering of Artifacts for Implicit Finite-Difference Paraxial Wave Equation Migration." *Geophysical Prospecting*, vol. 43: 203–220.

[Claerbout 1985] Claerbout, J. F. 1985. *Imaging the Earth's Interior.* Blackwell Scientific Publications, Boston.

$p_x \times p_y \times p_\omega$	Runtime (sec.)	Efficiency (%)
Spatial Parallelism		
$1 \times 1 \times 1$	84.1	100.0
$2 \times 1 \times 1$	92.4	91.0
$2 \times 2 \times 1$	103.2	81.5
$3 \times 3 \times 1$	108.7	77.4
$4 \times 4 \times 1$	108.9	77.2
$5 \times 5 \times 1$	112.2	75.0
$6 \times 6 \times 1$	114.8	73.3
$7 \times 7 \times 1$	115.6	72.8
$8 \times 8 \times 1$	116.2	72.4
Frequency Parallelism		
$1 \times 1 \times 1$	84.1	100.0
$1 \times 1 \times 2$	42.21	99.6
$1 \times 1 \times 4$	21.19	99.2
$1 \times 1 \times 8$	10.63	98.9
$1 \times 1 \times 16$	5.35	98.2
$1 \times 1 \times 32$	2.71	97.0
$1 \times 1 \times 64$	1.40	93.8

TABLE 1

Timings for a sample impulse problem for spatial, frequency, and mixed parallelism on the Intel Paragon. Single processor times are estimated. All other times are measured.

[Clayton and Engquist 1980] Clayton, R. and B. Engquist. 1980 "Absorbing Boundary Conditions for Wave-Equation Migration," *Geophysics*, vol. 45, 895–904.

[Fletcher 1988] Fletcher, C. 1988 *Computational Techniques for Fluid Dynamics Vol. I.* Springer-Verlag, Berlin.

[French 1974] French, W. S. 1974. "Two-Dimensional and Three-Dimensional Migration of Model-Experiment Reflection Profiles." *Geophysics*, vol. 39, 265–277.

[Graves and Clayton 1990] Graves, R. and R. Clayton. 1990. "Modeling Acoustic Waves with Paraxial Extrapolators." *Geophysics*, vol. 55, 306–319.

[Lee and Suh 1985] Lee, M. W. and S. Y. Suh. 1985. "Optimization of One-Way Wave Equations." *Geophysics*, vol. 50, 1634–1637.

[Li 1991] Li, Z. 1991. "Compensating Finite-Difference Errors in 3-D Migration and Modeling." *Geophysics*, vol. 56, 1650–1660.

[Ma 1981] Ma, Z. 1981 "Finite-Difference Migration with Higher Order Approximation." In *1981 Joint Mtg. Chinese Geophys. Soc. and Society of Exploration Geophysicists*, Society of Exploration Geophysicists.

[Xu 1996] Xu, L. 1996 *Working Notes on Absorbing Boundary Conditions for Prestack Depth Migration.*

[Yilmaz 1987] Yilmaz, O. 1987. *Seismic Data Processing, Investigations in Geophysics No. 2.* P.O. Box 702740, Tulsa, OK 74170-2740: Society of Exploration Geophysicists.