Imaging subsurface geology with seismic migration on a computing cluster

Sudhakar Yerneni[1,*], Suhas Phadke[2], Dheeraj Bhardwaj[3], Subrata Chakraborty[4] and Richa Rastogi[4]

[1]Cray–Hinditron, 203-A, Hemkunt Tower, 98, Nehru Place, New Delhi 110 019, India
[2]Reliance Industries Limited (Oil & Gas), Dhirubhai Ambani Knowledge City, Navi Mumbai 400 709, India
[3]Department of Computer Science and Engineering, Indian Institute of Technology, Hauz Khas, New Delhi 110 016, India
[4]Seismic Data Processing Team, Centre for Development of Advanced Computing, Pune University Campus, Ganeshkhind, Pune 411 007, India

In petroleum exploration, migration is one of the most compute-intensive steps in the seismic data processing sequence. It accurately images the reflected and diffracted energy and aids in delineating the detailed structural features of the underground geological formations. Migration techniques are highly compute and input/output (I/O) intensive, and therefore require high-performance computers. Parallel computers based upon clustering technology provide a cost-effective solution as most of the seismic migration algorithms are inherently parallel in several domains. Parellelism is achieved by decomposing the seismic data into frequency slices, plane-wave volumes or simply spatial cubes. Design and implementation of parallel algorithms for a cluster of workstations is quite straightforward. Large data volume necessitates the optimization of I/O, to and from mass storage devices. Here we have looked at the computational demands of several seismic migration methods and showed their implementation on a cluster of workstations. Efficiency and speed is achieved by code restructuring, parallel I/O, and overlapping computations with communication. Performance of migration algorithms is demonstrated on PARAM series of parallel computers.

SEISMIC data acquisition involves recording the wavefield at the earth's surface by sensors placed along a line or in an aerial pattern. The recorded wavefield is processed significantly before interpretation, to obtain a meaningful image of the subsurface. Migration[1] is the last and most compute-intensive step in the long chain of seismic data processing sequence. On an unmigrated seismic section, the reflected energy often provides an incorrect picture of subsurface structure, suffering from several distorting effects. Most notable are the diffractions from geological bed truncations and lateral movement of energy between the reflection points on dipping beds and the surface locations. Therefore, it is necessary to migrate seismic data in geologically complex areas.

There are many ways to migrate seismic datasets. The numerical techniques employed can generally be separated into three broad categories – summation or integral methods such as Kirchhoff migration[2]; finite difference methods[3,4]; and transform methods such as f–k migration[5–7]. All these migration methods make use of some approximation to the scalar wave equation. Methods employing full wave equation are also discussed in the literature[8,9].

The choice of a migration method to a particular dataset depends upon the complexity of the velocity model. Migration methods (f–k migration) that are computationally fast can only accommodate velocity variations with depth. Other methods, e.g. Kirchhoff method, finite difference method, and PSPI (Phase Shift Plus Interpolation) method, can also handle lateral velocity variations, but require large computational resources in terms of speed, memory and input/output (I/O). The highly accurate techniques involve the solution of the wave equation and are characterized by the requirement of large computational resources. Figure 1 illustrates the computational requirements of different migration methods. The Stolt method requires the least computational time, whereas reverse time migration requires the maximum computational time.

Migration can be carried out in prestack or poststack domain. Poststack migration methods are applied to stacked (zero-offset) data and are based upon the exploding reflector concept[4]. Prestack migration methods make use of gathers (e.g. common depth point or common mid point or shot or receiver gather). The input data size for a prestack migration technique is several orders of magnitude higher

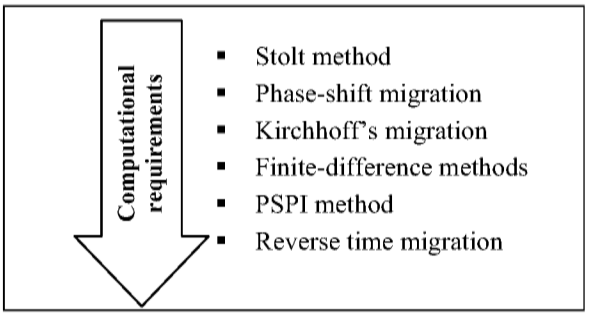

Figure 1. Various migration methods in ascending order of computational requirements.

*For correspondence. (e-mail: sudhakar@hinditron.com)

than that of the poststack migration technique. Therefore prestack migration involves large-scale numerical computation and I/O, and it images targets better even in the presence of geologically complex overburden. Migration can be performed in time or in depth. In the presence of strong lateral velocity variations, time migration followed by time to depth conversion does not image the reflected energy to its true subsurface position. Depth migration is essential in these cases. It compensates for ray bending, lateral velocity pull-ups and structure. A natural advantage of depth migration is that the output image is displayed in depth and therefore can be directly utilized for geological interpretation. All these migration methodologies can be applied on 2D and 3D datasets. For the obvious reason, the resolution is much better in 3D migration at the expense of greater computational cost.

Here we have discussed several imaging techniques, namely Kirchhoff method, frequency–wavenumber domain method (phase shift and PSPI), frequency–space domain method (finite difference migration) and their parallel implementation using message passing environment on PARAM series of high-performance parallel computers.

Parallel computing

The need for increased performance from a computer seems to have no bounds. Ever since conventional computers were invented, their speed has steadily increased to match the needs of emerging applications. Faster and faster processors have always been in demand to gain high performance. But, there is a limit to speed, whereby nothing can move faster than the speed of light. Single-processor computers have achieved speeds of several billion instructions per second, and pushed hardware technology to the physical limits of chip-building. But, soon this trend will come to an end because of physical and architectural bounds, which limit the computational power that can be achieved with a single-processor computer.

A natural way to circumvent this situation is to use parallel computing. To overcome the saturation point of the computation rates achievable by a serial computer, a large number of processors are connected in parallel and are programmed to work simultaneously on a given problem. Thus it can provide much higher raw computation rates than the fastest serial computers as long as this power can be translated into high computation rates for actual applications. Parallelism is the process of performing tasks concurrently, thereby offering the user the ability to execute the program more quickly and run larger applications. It is to be noted that parallelism has to ensure efficacy and should not affect accuracy.

The fact that most geophysical problems display an inherent parallelism, the geophysical programs written in true parallel form can exploit the power of parallel computers. Parallel processing has proven to be a viable solution to improve performance in seismic industry. The traditional single symmetric multiprocessor (SMP)-based high performance computing systems have lost their popularity due to higher costs. The constant trend towards the dramatically improved price/performance and parallel performance, is moving computing inexorably towards a commodity processor model. Clustering of commodity microprocessors has become the technology for all seismic companies. Their compelling price/performance ratios, flexibility to add additional processors, ability to utilize heterogeneous hardware and operating systems, are all significant advantages for seismic imaging.

PARAM

The Centre for Development of Advanced Computing (C-DAC), Pune has advented the openframe architecture for scalable and flexible high-performance computing unifying well-known NOW (Network of Workstations), COW (Cluster of Workstations) and MPP (Massively Parallel Processor) architectures. This architecture has been realized in the PARAM series of parallel supercomputers at C-DAC. The latest member of this series, PARAM Padma, is the next generation high-performance scalable computing cluster, with a peak computing power of a little over one teraflop. The hardware environment is powered by the compute nodes based on the state-of-the-art Power4 RISC processors from IBM, in SMP configurations. Fifty-four four-way SMPs and one 32-way SMP (in totality 248 CPUs with a clock speed of 1 GHz per CPU) are connected through a primary high-performance system area network, PARAMNet-II, designed and developed by C-DAC and a gigabit ethernet as a back-up network. The total system has an aggregate primary memory of 0.5 terabytes. The storage system of PARAM Padma has been designed to provide a primary storage of 5 terabytes scalable to 22 terabytes and a secondary back-up storage subsystem of 10 terabytes scalable to 100 terabytes. The network-centric storage architecture, based on state-of-the-art Storage Area Network (SAN) technologies from SUN Microsystems, ensures high performance, scalable and reliable storage and achieving an I/O performance of up to 2 gigabytes/s. This system is installed at CDAC's Terascale Supercomputing Facility (CTSF) at C-DAC, Bangalore.

One generation before PARAM Padma in the series of PARAM supercomputers is PARAM 10000. This machine is powered by the UltraSparc series of servers/workstations configured as Compute nodes, file servers, graphics nodes and internet server nodes from SUN. These nodes are interconnected through PARAMNet high-bandwidth, low-latency network designed in-house. Further, a choice of other high-performance networks such as myrinet and gigabit/fast ethernet is also available. High-performance secondary storage of terabytes in capacity is based on SUN Enterprise Network Array. This system is housed in National PARAM Supercomputing Facility (NPSF) at C-DAC, Pune.

Parallel programming environments

The most commonly used method of parallel programming is message passing or some variant of message passing. In basic 'message passing', the processors coordinate their activities by explicitly sending and receiving messages[10,11].

In all parallel processing algorithms, data must be exchanged between cooperating tasks. Parallel Virtual Machine (PVM)[12] and Message Passing Interface (MPI)[13] are the most common message-passing libraries used for exchange of data between processors. The main advantages of establishing message-passing interfaces are portability and ease of use. A standard message-passing interface is a key component in building concurrent computing in which applications, software libraries, and tools can be transparently ported between different machines.

PVM/MPI systems use the message-passing model to allow programmers to exploit distributed computing across a wide variety of computer types, including MPPs. PVM/MPI supports heterogeneity in architecture, data format, computational speed, machine load and network load.

Parallel strategies for seismic imaging

There can be two types of approaches for any parallel implementation: function parallelism and data parallelism. In the function parallel approach, portions of the code are distributed among the processors, while in the data parallel approach subsets of the data or model space are distributed among the processors for parallel computation, which runs identical codes on these subsets. For parallellizing most of the seismic migration methods, the data parallel approach is the most suitable one.

The parallel migration codes are analogous to a client–server system, where there is one server with multiple clients. One can also think of it as a master–worker system, where the master works as the manager and assigns tasks to his workers. The job of the master is to provide the required parameters and data to all the workers and distribute workload properly, so that idle time of the workers is minimized. Also, at the end the master should collect the finished work from all the workers, compile it and store it in a proper manner. One of the processors acts as master and the worker tasks are assigned to different processors. In another approach, the master can also be a worker after distributing the data, which is termed as latency hiding in parallel computing literature.

Migration algorithms

There are two basic steps in the migration of seismic data – extrapolation and imaging. Extrapolation involves numerical reconstruction of the wavefield at depth from the wavefield recorded at the earth's surface. Imaging is the process that allows one to obtain the local reflection strength from the extrapolated data in depth and create an image of the subsurface reflectors.

Phase shift and PSPI migration

Given the upcoming seismic wavefield $P(x, y, 0, t)$ that is recorded at the surface, the aim is to determine reflectivity $P(x, y, z, 0)$, which requires extrapolating the surface wavefield to depth z, and then recovering it at $t = 0$ (imaging). The phase-shift method is a straightforward process for extrapolating downward with phase-shift operator $[e(ik_z z)]$ and subsequently evaluating the wavefield at $t = 0$ by summing over all the frequencies, since the reflectors explode at $t = 0$ according to the exploding reflector concept.

The phase-shift method begins with a three-dimensional Fourier transform (or 2D transform in case of 2D data) of the dataset. Then the wavefield is downward continued in depth by applying phase-shift operators. This forms the extrapolation part of migration. The imaging part involves inverse Fourier transformation over all the wavenumbers and the summation of all the frequencies at each depth step.

If the migration velocity has no horizontal variations, the phase-shift method extrapolates the wavefield exactly by rotating the phases of each Fourier component. In the presence of lateral velocity variations, the exact extrapolation equation is no longer valid. The PSPI method circumvents the problem of lateral changes in migration velocity by downward extrapolation of the wavefield with several reference velocities and then interpolating the wavefield for the correct velocity[7].

The phase shift migration algorithm is well suited to the data parallel approach because of its decoupling of the problem in the wavenumber domain (either inline wavenumber, kx or crossline wavenumber, ky) or frequency domain. First the master program reads all data parameters and the velocity model, and broadcasts them to all processors. Next, the master program reads part of the Fourier transformed data that has to go to a specific worker processor and sends it to that worker. The master distributes the data over wavenumber corresponding to crossline direction. Thus, each worker has (ω, k_x) data corresponding to his part of k_y. The master processor collects the downward continued data from all the worker processors, performs imaging by inverse Fourier transformation and writes it onto the disk.

The parallelization of the PSPI method is in terms of frequency (ω). Here the data are first Fourier transformed and then different processors read and migrate their share of frequencies. At each depth step the phase-shift is applied for the reference velocities and then wavefield is interpolated for the actual velocity. One of the processors that acts as the master, collects and images the data by summing over all the frequencies.

Kirchhoff migration

The diffraction summation approach is one of the most popular migration techniques in seismic processing industry. It

was the first computer implementation of a migration algorithm. In this method, the amplitudes are summed along the diffraction hyperbola and the result is placed at its apex[2]. A straightforward summation of amplitudes is carried out along the hyperbolic trajectory whose curvature is governed by the velocity function. Before this the summation data are corrected for obliquity factor, spherical spreading factor and wavelet shaping factor. In time migration, the velocity function used to compute the travel-time trajectory is the RMS velocity at the apex of the hyperbola. The lateral extent of the diffraction hyperbola (aperture) in the summation process is a crucial parameter in the migration process[14].

Parallelization of Kirchhoff migration algorithm is quite straightforward. We use the data parallel approach and parallelize the loop over the output location. The master sends the migration locations to each worker along with the corresponding velocity model. The master also calculates the aperture function and sends it to the workers. With a knowledge of the location of the output CDP and aperture, the master identifies how much input data each worker requires for migrating its portion of output locations.

Finite difference migration

Finite difference migration is also referred as ω–x migration, as it is carried out in the frequency–space domain. The first part of the migration, i.e. extrapolation of the input wavefield recorded at the surface is performed here by finite difference solutions to the scalar wave equation. When the recorded wavefield is downward continued in depth, diffractions in the data collapse to the apexes of their respective diffraction hyperbolas. First the data are Fourier transformed in the time direction. These frequency planes are downward continued by predefined depth step-size. At each depth the migrated image is obtained by summing over all frequencies.

The finite difference migration code is parallelized in terms of frequency planes. Frequency parallelism takes advantage of the fact that each frequency is downward continued independently following the principal of linear superposition; thus no communication is necessary until the final image is generated. Because of this limited communication, high parallel efficiency could be obtained.

Finite difference migration codes are written in client–server or worker–master mode. In poststack migration algorithm, the stacked data are first Fourier transformed with respect to time and stored in frequency sequential format. The frequency bandwidth to be used for migration is determined from spectral analysis of the input traces. Temporally transformed data with velocity depth model form the input to the depth migration code. After reading all the required parameters, the master determines the number of frequencies and frequency bandwidth to be assigned to each worker. Then the master reads and sends the frequency data to the designated worker. Further, the migration algorithm runs through the depth steps. The required velocity data for that depth step are sent to the workers. Finally, the migrated data from all the workers at each depth are collected by the master, imaged and stored on the disk.

Parallel I/O in seismic imaging

There are two stages of I/O in the seismic migration algorithms: reading the input seismic data during initialization and reading the velocity model, and writing the migrated image during migration. Throughout initialization, the time-consuming portion of the workload is the I/O, while preliminary computations are a secondary load. However, during migration these roles are reversed; migration is the primary workload and I/O is secondary.

Most of the clusters use NFS (Network File System) and MPI calls to communicate and synchronize between the processors. One limitation of NFS is that the I/O nodes are driven by standard UNIX read and write calls, which block requests. This is not a problem for applications with a small volume of I/O, but as the volume increases, it is necessary to be able to overlap computations with I/O to maintain efficient operation[15,16]. In the traditional master–slave type of workflow, only the master reads the data from input files and sends them to the workers or receives data from the workers and writes to the output files. In MPI I/O implementation, workers can access input or output files directly to read or write data.

To reduce I/O time, MPI I/O calls have been incorporated into all discussed migration algorithms, wherever it is advantageous. In this implementation, all the workers use MPI I/O to read the required data (both seismic and velocity data) from the respective portions of the data files, simultaneously, if each worker needed different data. But in case the same portion of data is needed for all workers, then it is efficient if the master read the particular data and broadcast them to all workers[17].

In finite difference migration and PSPI seismic data are read using MPI I/O calls, i.e. workers reading the respective portions of seismic data directly from the file. In Kirchhoff migration, both seismic data and velocity data are read in the same way. Even final migrated seismic traces that each worker is generating in Kirchhoff migration, are written to the output file by individual workers themselves using MPI I/O. But the velocity file is read by the master and then broadcast to workers in finite difference migration and PSPI migration. This is because all workers need whole velocity depth slice. The same technique is used while writing migrated seismic depth slice in finite difference migration and PSPI migration algorithms.

Results and discussion

The output of any migration process is a seismic section that shows the structural details of the subsurface in true sense.

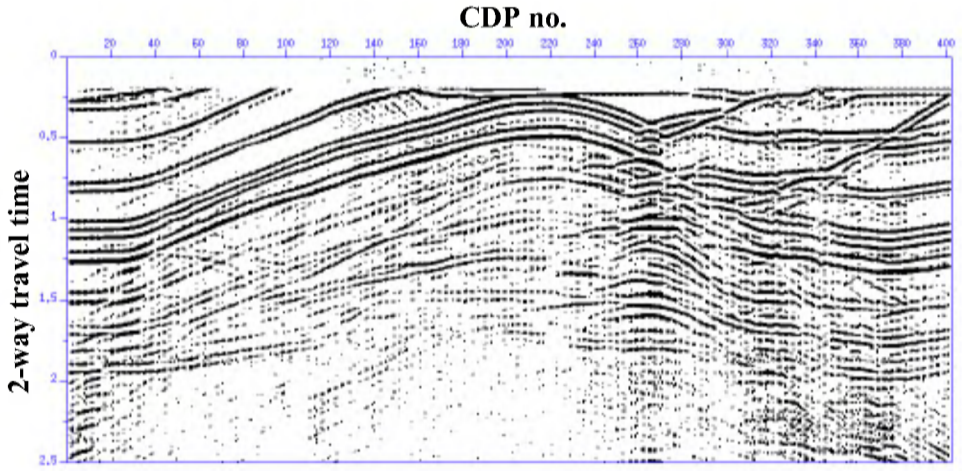

Figure 2. A line from 3D stacked data volume of SEG/EAGE overthrust model.

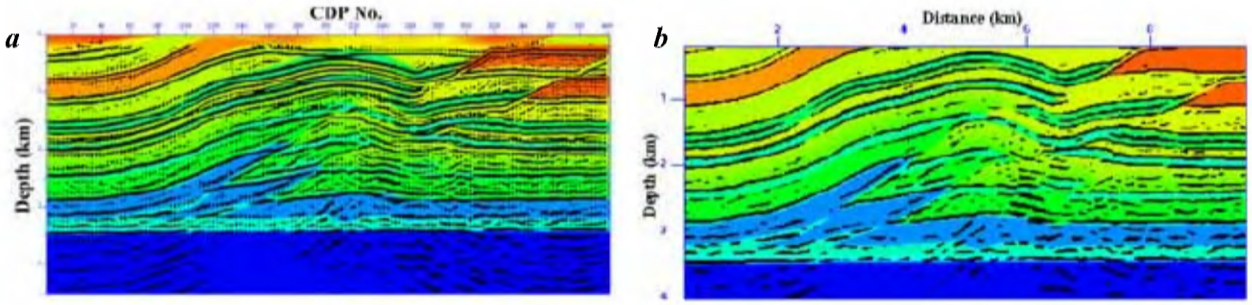

Figure 3. The same line as in Figure 2 from (*a*) 3D finite difference and (*b*) 3D PSPI migrated volume.

The stacked seismic section before migration, though it shows the subsurface structure, is often misleading and sometimes gives rise to spurious prospective subsurface structures. Hence migration is an essential and crucial step of data processing in petroleum exploration.

To show the efficiency and efficacy of our developed parallel migration algorithms, we migrated SEG/EAGE overthrust dataset using all discussed migration algorithms with different number of processors of PARAM Padma and PARAM 10000. The dataset consists of 97 lines, 401 CDPs per line and 350 time samples per trace with 8 ms sampling interval. In depth migration algorithm, the wavefield was downward continued by 187 depth steps with 25 m depth interval. Figure 2 shows a line from the 3D stacked data volume. Figure 3 *a* and *b* shows the same line as in Figure 2 after finite difference and PSPI migration respectively. Figure 4 *a* and *b* depicts the whole 3D migrated volume of the same overthrust data by Kirchhoff and finite difference migration algorithms respectively.

Efficiency of any parallel application depends on its scalability with increasing number of processors. Execution timing and speedup graphs of finite difference migration and PSPI migration algorithms are shown in Figure 5 *a* and *b* respectively, on PARAM Padma. Figure 6 is a bar chart for execution time of Kirchhoff migration on PARAM 10000. The graph shows how the time of execution of a given problem changes with increase in the number of CPUs used to solve the problem. Speedup indicates how fast the job is done on a given number of CPUs compared to the best sequential execution timing. As is clearly seen in the execution-timing graph, PSPI has larger computations than finite difference migration. Thus, for any given number of processors, PSPI has better computations to communication ratio. This explains why PSPI has a better speedup graph than that of finite difference migration, as seen in Figure 5 *b*.

Conclusion

Here we have described several methods for the migration of seismic data. Highly efficient and scalable codes have been written for all the methods and are implemented on PARAM 10000 and PARAM Padma parallel supercomputers at C-DAC. All the migration codes (finite difference, PSPI

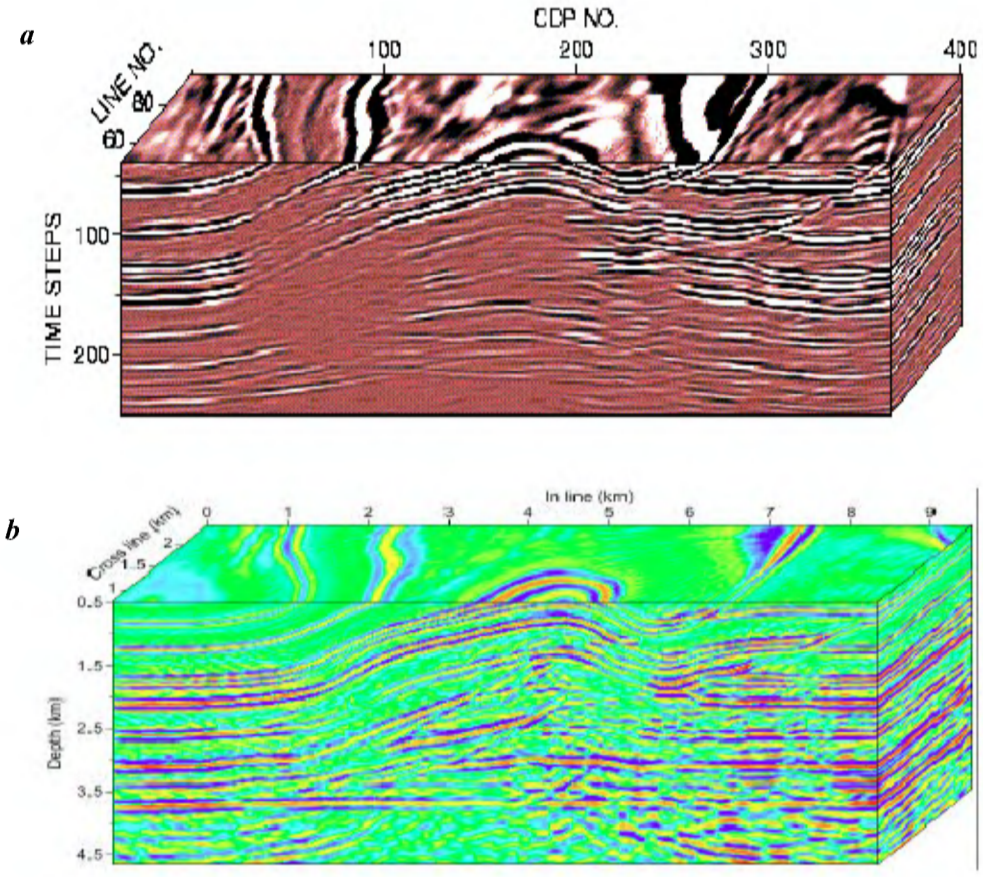

Figure 4. Migrated volume of SEG/EAGE overthrust data by (*a*) Kirchhoff migration and (*b*) finite difference 3D depth migration algorithms.

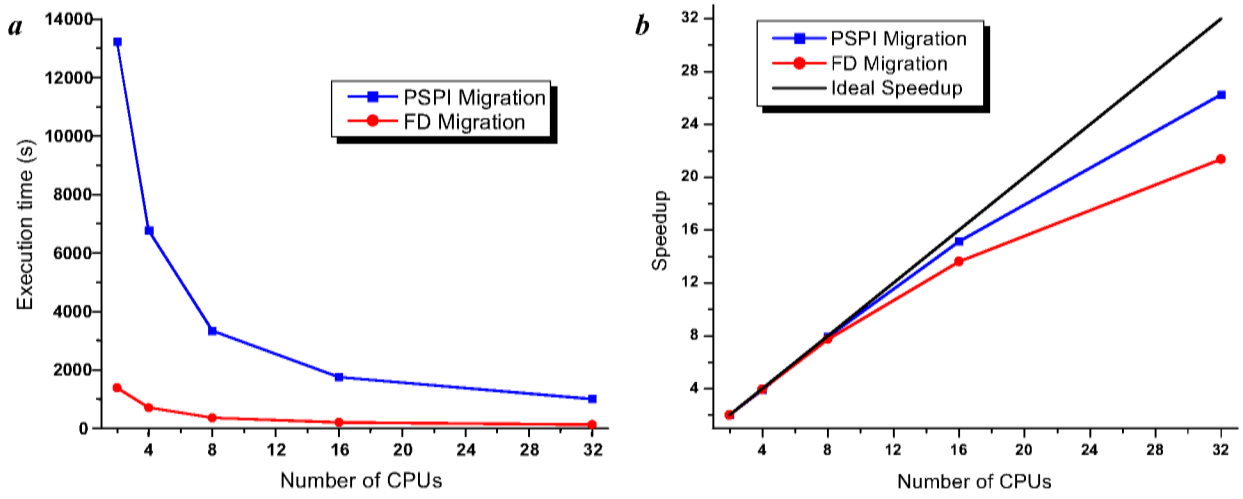

Figure 5. (*a*) Execution timings and (*b*) speedup graphs for 3D finite difference and PSPI migration of SEG/EAGE overthrust data on PARAM Padma.

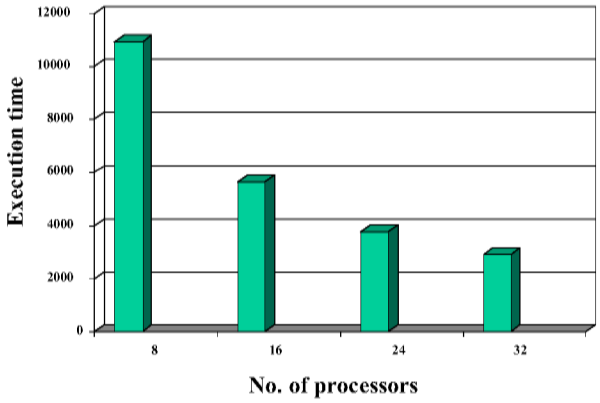

Figure 6. Number of processors versus execution time graph of Kirchhoff migration for 3D dataset of SEG/EAGE overthrust model on PARAM 10000 system.

and Kirchhoff) are inherently parallel in several domains and therefore it is possible to write efficient parallel codes. The performance graphs clearly demonstrate this claim. Parallel computing has made it possible to apply such techniques to large, real datasets on a routine basis.

1. Berkhout, A. J., *Seismic Migration – Imaging of Acoustic Energy by Wavefield Extrapolation*, Elsevier, 1980.
2. Schneider, W. A., Integral formulation for migration in two and three dimensions. *Geophysics*, 1978, **43**, 49–76.
3. Claerbout, J. F., *Fundamentals of Geophysical Data Processing*, McGraw Hill, 1976.
4. Claerbout, J. F., *Imaging the Earth's Interior*, Blackwell, 1985.
5. Stolt, R. H., Migration by Fourier transform. *Geophysics*, 1978, **43**, 23–48.
6. Gazdag, J., Wave equation migration with the phase-shift method. *Geophysics*, 1978, **43**, 1342–1351.
7. Gazdag, J. and Sguazzero, P., Migration of seismic data by phase-shift plus interpolation. *Geophysics*, 1984, **49**, 124–131.
8. Kosloff, D. D. and Baysal, E., Migration with full wave equation. *Geophysics*, 1983, **48**, 677–687.
9. Reshef, M. and Koslofdf, D., Migration of common shot gathers. *Geophysics*, 1986, **51**, 324–331.
10. Geist, Al, Beguelin, Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V., *PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing*, The MIT Press, 1994.
11. Pacheco, P. S., *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc, California, 1997.
12. Phadke, S. and Bhardwaj, D., Parallel implementation of seismic modeling algorithms on PARAM OpenFrame, Neural, *Parallel Sci. Comput.*, 1998, **6**, 469–478.
13. Phadke, S. and Yerneni, S., A PVM implementation of 2D acoustic wave modelling on PARAM 10000. *Int. J. Appl. Sci. Comput.*, 1999, **6**, 120–130.
14. Rastogi, R. and Phadke, S., Optimal aperture width selection and parallel implementation of Kirchhoff migration algorithm. Expanded abstr., SPG-2002, 2002.
15. Oldfield, R. A., Womble, D. E. and Ober, C. C., Efficient parallel I/O in seismic imaging. *Int. J. High Performance Comput. Appl.*, 1998, **12**, 333–344.
16. Poole, J., Scalable I/O initiative, Technical Report CCSF-38, Caltech concurrent supercomputing facilities, California Institute of Technology, Pasadena, 1994.
17. Bhardwaj, D., Yerneni, S. and Phadke, S., Efficient parallel I/O for seismic imaging in a distributed computing environment. In Proceedings of 3rd Conference and Exposition on Petroleum Geophysics (SPG 2000), 2000, pp. 105–108.

ACKNOWLEDGEMENTS. We thank the Deep Crustal Studies Division of DST, New Delhi for financial support. We also thank C-DAC, Pune, for providing PARAM computing facility and for permission to publish this work.

Received 22 May 2003; revised accepted 18 September 2004