

Storage and performance issues in reverse time migration

Suhas Phadke

Computational Research Laboratories Ltd.
TACO House, Damle Path, Pune 411004, India
suhas.phadke@crlindia.com

Santosh Dhubia

Computational Research Laboratories Ltd
TACO House, Damle Path, Pune 411004, India
santosh.dhubia@crlindia.com

SUMMARY

Prestack Reverse Time Migration (RTM) algorithm for depth imaging has now become feasible due to advances in high performance computing and clever programming techniques. RTM makes use of full acoustic two-way wave propagation algorithm for both forward and reverse time extrapolation. It is a computationally expensive process and the program run times are large in terms of CPU cycles and disk storage. The computational challenge is overcome by clever parallel programming techniques. As the forward propagated wavefield has to be stored at all time steps and read back for imaging, RTM needs a large amount of storage. This problem is solved by either using a large central storage or computing the forward wavefield twice. In this paper we propose the use of local disk space available on each node to write the forward wavefield and compare it with the other two methods. The use of local disk space reduces the communication time to the central storage and hence provides an improvement in performance. The performance is shown for 2D RTM algorithm.

Key words: Reverse Time Migration (RTM), parallel programming, acoustic wave propagation, depth imaging.

INTRODUCTION

Prestack Reverse Time Migration (RTM) is a state of the art depth migration technique for imaging subsurface geological structures from the recorded seismic data (Baysal et al., 1983; McMechan, 1983; Yoon et al., 2003; Farmer et al., 2006; Jones et al.). The strength of RTM is based upon the fact that it uses a two-way acoustic wave equation for both forward and reverse time extrapolation, thus improving imaging in areas where complex geology violates the assumptions made in Kirchhoff or one-way wave equation based migration. The affordable availability of computer power and advances in programming techniques have made it feasible to apply prestack RTM algorithm to field data sets.

Reverse Time Migration (RTM) is based upon the reconstruction of the wavefield using two-way acoustic wave equation. It reconstructs the seismic images in the complex subsurface with minimal approximations. RTM's finite difference implementation is highly compute and storage intensive, as the computations are directly proportional to the number of processed shot gathers and the storage is proportional to the number of grid points in the model multiplied by the number of time steps. The computational

cost, data storage and run time also increase with the frequencies of interest and dimensionality of the considered subsurface model. 3D RTM is an order of magnitude more compute and storage intensive as compared to 2D RTM.

RTM models two wavefields: a forward wavefield produced by the seismic shot; and a reverse wavefield modeled by backward propagation of the data recorded by each receiver. The image is obtained by cross correlation of the forward and reverse wavefields. One of the wavefields has to be stored on the disk and reused during imaging. The requirement poses the data storage problem in RTM implementation. For small problems the scheme with storage of all the forward wave fields, and its reuse during reverse extrapolation and imaging can serve the purpose. But as the problem size increases, the disk I/O and data storage become the time consuming operation. To overcome this problem, Symes (2007) introduced the optimal checkpointing scheme in RTM, which reduces the need of disk I/O at the cost of increased computational time. Typically, it is a trade-off between the computational time and the computational resources.

In this paper we look at three different ways of implementing data storage in RTM algorithm. We shall demonstrate this for only 2D RTM algorithm, which can be easily extended to 3D.

COMPUTING AND STORAGE FOR RTM

Modern day linux clusters offer large computational power. Algorithms have to be rewritten in such a manner as to harness the available computing power. RTM based upon the finite difference solution to acoustic wave equation can take advantage of a large number of compute nodes by distributing shot gathers and solving the wave propagation problem by domain decomposition.

Depth migration of a single shot gather is carried out on several nodes depending upon the problem size. The problem domain is decomposed into several subdomains and distributed on available nodes using MPI parallel programming paradigm. Each node has a number of cores (4 or 8), which can work in parallel. Wave propagation is carried out by a time marching process on each node/core. Along with the computational power, linux clusters have a ravenous appetite for high performance storage and data access.

RTM makes use of the solution of two-way acoustic wave equation in the following manner. The algorithm is implemented in shot gather domain. First the forward extrapolation of the source wavefield is carried out for each shot location through the gridded velocity model using finite difference method. In order to make the forward wavefield accessible in reverse order during reverse extrapolation of the

recorded data, it must be stored at each time step. This requires a large amount of storage, specially for 3D RTM. Next the recorded wavefield (shot gathers) is backward propagated in time and is correlated with the forward propagated wavefield to obtain the image. Storing the forward wavefield and then reading it back during reverse extrapolation poses storage and I/O problem for large models. Various methodologies have been put forward to deal with this issue. Symes (2007) has suggested an optimal checkpointing approach, where the forward wavefield is calculated twice, thereby eliminating the need for the storage of the snapshots. Here we propose a different approach that makes use of the local storage available on each node.

Figure 1 illustrates the application of RTM to a part of the Marmousi2 model (Martin et al., 2006). One can observe that RTM is able to accurately image the complex geological subsurface. Now let us look at the storage and performance issues in RTM.

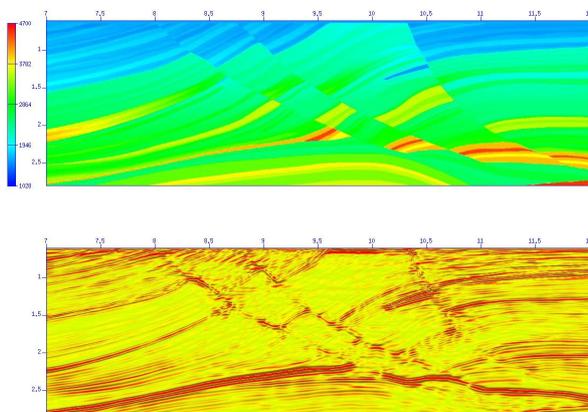


Figure 1: (a) A portion of the Marmousi2 model. (b) Reverse Time Migrated section of the synthetic data for the model shown in Figure 6a. The input data for RTM was generated using an acoustic wave propagation algorithm.

STORAGE AND PERFORMANCE ISSUES

Reverse Time Migration (RTM) involves forward modeling, reverse time extrapolation and imaging, so it is more compute intensive than seismic modeling. Parallel implementation of both forward modeling and reverse extrapolation are done using a domain decomposition scheme, where the model domain is broken into a number of subdomains of equal size, and each subdomain is assigned to a different CPU core. After each time step the wavefield at grid points lying on the subdomain boundaries have to be communicated to the neighboring subdomains for the next time step. Forward modeling and reverse time extrapolation requires almost equal amount of computational time as floating point operations involved in both are almost same. Imaging requires cross correlation of wavefield amplitudes at time step 't' of forward modeling versus wavefield amplitudes at time step 'N-t' of reverse time extrapolation, where 'N' is the number of total time steps. The implementation of the RTM algorithm for the clusters with several Tera Bytes of central storage and several Giga Bytes of local storage, can be done in the following three ways:

Scheme 1: In this scheme the forward wave propagation is completed first, and the wavefield at each time step is stored

on the central disk. During the reverse time extrapolation of the recorded data, the forward propagated wavefield is read from the central disk and correlated with the backward propagated wavefield. In the RTM implementation of master / worker model, where one core acts as the master and others act as workers, the snapshots from all the subdomains have to be written on the central disk using the interconnect.

Scheme 2: In this scheme each subdomain stores its snapshots of forward wave propagation on the local storage attached to each processor. This does not require communicating to the master for storage. During reverse extrapolation and imaging these snapshots are read back from the local storage and used for crosscorrelation. Therefore each processor creates the image of its subdomain. In the end these images are collected by the master to make the final migrated image.

Scheme 3: If there is not enough storage on the central disk or on the local disk then both the schemes given above may fail. In this case the snapshots of forward propagated wavefield are not stored at each time step, but instead very few snapshots are stored at some predefined time intervals. In the present work, we have chosen to store the wavefields data at every 1000 time steps. As the snapshots of forward propagated wavefield are required at each time step for imaging during reverse extrapolation, we recomputed the forward wavefield using the snapshots stored at predefined time intervals, in our case 1000 time steps. This increases the computational requirement as the forward wave propagation is carried out twice, but reduces the storage requirements. Thus, in this scheme the disk storage problem has been countered by increase in computation.

HARDWARE SPECIFICATIONS

For this work we made use of CRL's high performance compute infrastructure called "EKA". The system, comprises of 1800 computing nodes, where each node is an Intel Xeon clovertown dual quad-core processor, with 3GHz processor speed, 16GB RAM and 72GB local hard disk space. There is an 80 TB of external storage with a parallel file system attached to this High Performance computer. A schematic of the node is shown in Figure 2. All the 8 cores of the node share the same memory and local disk via a common I/O bus. It will be shown later that the speedup is dependent upon the number of cores used per node. The nodes are connected by a 20 GBPS infiniband interconnect.

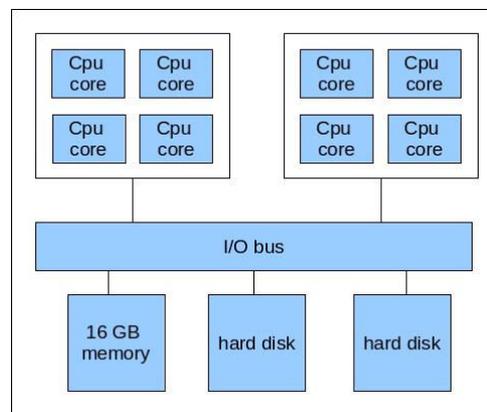


Figure 2: A schematic of the nodes of the high performance computing system.

COMPUTATIONAL AND STORAGE REQUIREMENT OF EACH SCHEME

In this section we look at the computational and storage requirement for each scheme for a single shot gather. Floating point operations, communication between nodes and storage contribute towards the overall performance.

Scheme 1: The forward modeling and reverse time extrapolation both use a finite difference scheme. The model is discretized in terms of number of grid points $N_x \times N_z$, where N_x are the number of grid points in x direction and N_z are the number of grid points in z direction. Hence floating point operations are of the order of total number of grid points. Storage is a function of the number of snapshots and the number of grid points. Snapshots are stored on the central disk for this scheme. Since the wave propagation problem is solved by domain decomposition, all the subdomains have to write the wavefield for each snapshot on the central storage, using the network. Thus, the total floating point operations are of the order of $O(N_x \times N_z \times N)$ and disk space requirement is $N_x \times N_z \times N \times \text{sizeof(float)}$, where N is the number of snapshots.

Scheme 2: In this scheme the floating point operations for forward and reverse extrapolation are same as scheme 1. But the storage of the snapshots of each subdomain takes place on the local disk attached to the CPU core, thereby eliminating the need for communicating to the master. To store N snapshots for a model size of $N_x \times N_z$, distributed evenly on M nodes the local disk required is $(N_x \times N_z \times N \times \text{sizeof(float)}) / M$. Local disk is common to all the cores used that node.

Scheme 3: In this scheme the wave field information from the forward wave propagation is stored only at selected time intervals, so the hard disk space requirement will be $N_x \times N_z \times \text{sizeof(float)} \times \text{Number of Time Intervals}$. During the reverse time extrapolation, the required forward wavefield is recomputed from the forward wavefield stored at selected time intervals, therefore the required computational time will be 1.5 times than that of the above two schemes.

RESULTS AND DISCUSSION

We performed the benchmark tests of the parallel 2D RTM of a single shot gather, for all the above three schemes, for variable problem sizes on CRL's high performance computing system.

Figure 3 shows the comparative plot of the RTM 2D code for all schemes, for variable grid sizes. Schemes 1 and 2 have the same compute requirement but the scheme 2 takes less time on account of local storage. Scheme 3 which is more compute intensive as compared to schemes 1 and 2, obviously takes more time to complete.

Figure 4 shows the speedup for 2D RTM of a single shot gather on different number of cores. It is important to note that the speedup is dependent upon the number of cores used on the same node. If we use only single core of each node, then the speedup is almost linear. However if we use more cores from the same node (maximum of 4 cores per node) then the speedup is not linear. This is due to the fact that the cores residing on the same node use the common I/O bus to access

memory and local disk. However for production runs we have to use multiple cores from the same node as we are interested in the overall reduction in the compute time.

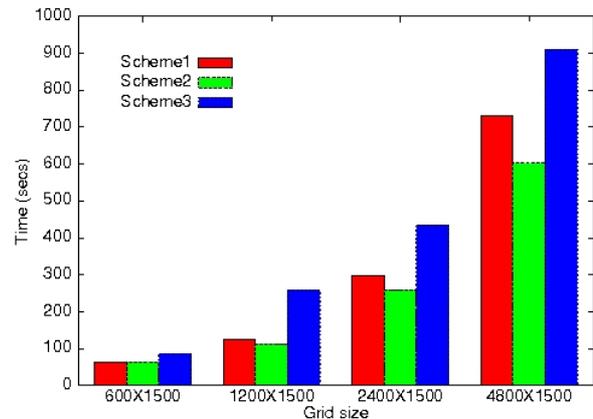


Figure 3: Runtime of 2D RTM for variable grid sizes on 16 cores for all schemes for 2000 time steps.

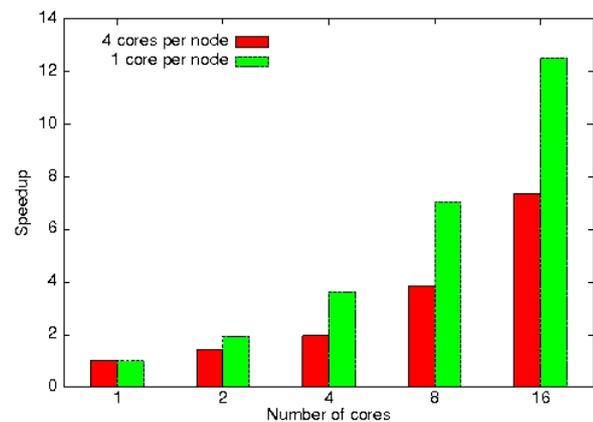


Figure 4: The speedup graph of 2D RTM of a single shot gather. The speedup is a function of the number of cores used on each node. See text for more details.

Run time for a shot gather for all three schemes for 2000 time steps are shown in Figure 5a for a model size of 4800X1500 grid points. As expected the scheme 2 gives the best run times. Figure 5b illustrates the speedup for the three schemes. The speedup is almost same for all the schemes for smaller number of cores, however for large number of cores scheme 2 performs better.

To compare the difference between runtimes of scheme 1 and scheme 2, we perform the 2D RTM of 5 seconds data with the grid size of 4800X1500 for a single shot gather using 16 cores (4cores per node). Run time for scheme 1 was about 2220 seconds (37 minutes) for 10000 time steps using 16 cores on 4 nodes, while scheme 2 finished the same job in 1585 seconds (27 minutes), a reduction of about 10 minutes. If we assume that the number of shots in a survey is 1000, then we shall save about 166 hours to carry out 2D RTM of all shot gathers. The main purpose is to reduce the total runtime of the project, and scheme 2 will achieve this objective.

Scheme 1 uses central storage while scheme 2 uses local storage, and generally the central storage is a lot more as compared to the local storage attached to each node. Disk space requirement is also proportional to the problem size.

Since larger problems give better speedup, it can be distributed on a large number of nodes in such a manner that the local disk space can accommodate the snapshots of the subdomains. This way we can solve larger problems with scheme 2. Disk space requirement for scheme3 is much lesser compared to other two schemes, but the computation time is 1.5 times than that of scheme 2. For thousands of shot gathers, scheme 3 will be more time consuming as compared to other two schemes.

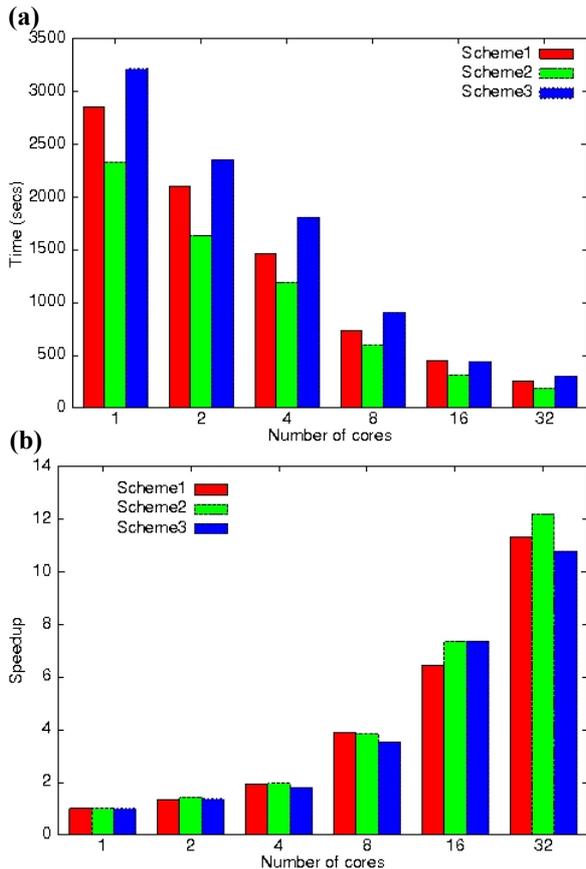


Figure 5: (a) Absolute time for 2000 time steps as a function of number of cores. (b) Speedup for all schemes as a function of number of cores.

CONCLUSIONS

Today, RTM is the best available imaging algorithm in the seismic processing industry. However the runtimes for its

application are large. Therefore even today this technique is sparingly used. In this paper we have looked at storage and performance issues of 2D RTM. A new scheme has been proposed which stores the snapshots of the forward wavefield on the local disk of the nodes. It has been demonstrated that this scheme reduces the overall runtime for 2D RTM, thereby reducing the overall cost of the project. Efficiency and speedup for the proposed scheme is also good from production point of view. This methodology is also being implemented for 3D RTM.

ACKNOWLEDGMENTS

The authors would like to thank Computational research Laboratories Ltd., Pune, India, for the usage of the supercomputing system 'EKA' and the permission to publish this work.

REFERENCES

- Baysal, E., Kosloff, D. D., and Sherwood, J. W. C., 1983, Reverse Time Migration, *Geophysics*, 48, 1514-1524.
- Farmer, P. A., Jones, I. F., Zhou, H., Bloor, R. I., and Goodwin, M. C., 2006, Application of reverse time migration to complex imaging problems, *First Break*, 24, 65-73.
- Jones, I. F., Sugrue, M., King, D., Goodwin, M., Berranger, I., Zhou, H., Farmer, P., 2006, Application of reverse time migration to complex North sea imaging, *PETEX Biennial Meeting*.
- Martin, G. S., Wiley, R., and Marfurt, K. J., 2006, Marmousi2: An elastic upgrade for Marmousi, *The Leading edge*, 25, 156-166.
- McMechan, G. A., 1983, Migration by extrapolation of time dependent boundary values, *Geophysical Prospecting*, 31, 413-420.
- Symes, W. W., 2007, Reverse time migration with optimal checkpointing, *Geophysics*, 72, SM213-SM221.
- Yoon, K., Shin, C., Suh, S., Lines, L. R., and Hong, S., 2003, 3D reverse time migration using the acoustic wave equation: An experience with the SEG/EAGE data set, *The Leading Edge*, 22, 1, 38-41.