Frontiers + Innovation
2009 CSPG CSEG CWLS CONVENTION • CALGARY, ALBERTA, CANADA

Parallel Implementation of Pre-stack Least-squares Migration for Distributed Systems

Sam T. Kaplan*
SAIG, Department of Physics, University of Alberta, Edmonton, Alberta, Canada
skaplan@ualberta.ca

Summary

Least-squares wave-equation pre-stack depth migration is a computationally expensive algorithm, requiring repeated application of wave-field modeling and migration algorithms. We describe a parallel implementation of least-squares migration for distributed parallel computing systems using MPI. The implementation distributes both the storage and processing requirements. In storage, the migrated data is distributed across the memory spaces of the MPI processes using the depth dimension. Likewise, the data (before migration) is distributed across the memory spaces of the MPI processes using the frequency dimension. In processing, the computational work is shared across MPI processes by partitioning the frequency dimension. We apply the algorithm to the Marmousi finite difference data-set, and find that the computation time of the algorithm scales with the number of MPI processes.

Introduction

In this paper we consider the parallel implementation of split-step wave-field modeling and its adjoint (migration), as well as least-squares migration. The algorithmic complexity (expense) of these algorithms encourages their parallel implementation. Within the context of least-squares migration, previous efforts have been made to parallelize migration operators (split-step and PSPI) using shared memory computers and OpenMP (e.g. Kuehl, 2002; Wang, 2005). In this abstract we consider a parallel implementation for distributed computing systems using MPI. The advantage being that distributed systems tend to be larger than shared memory systems in both their number of CPU's and their amount of memory. The disadvantage being in the complexity of the resulting algorithm in needing to cope with a distributed (rather than shared) memory space. Regardless, we thought it would be a useful exercise to write an MPI implementation of least-squares migration with split-step wave-equation operators.

We begin with the equations of interest. Then, we present our parallelization strategy and provide scaling results, showing the behavior of the implementation for up to, and including, thirty-two MPI processes. Our tests are run on a Linux Opteron cluster with InfiniBand interconnect, and hosted by the Western Canada Research Grid (www.westgrid.ca).

Operators

Pre-stack least squares migration requires a data modeling (forward) operator and its adjoint. These are often referred to as, respectively, demigration and migration (e.g. Santos et al., 2000). Here, we use DSR operators similar to Popovici (1996), with the split-step approximation (Stoffa et al., 1990) to allow for

lateral velocity variation. Our derivation stems from the Born approximation, and is similar (but not the same) as Huang et al. (1999), and results in the forward operator, mapping migrated image gathers α to data gathers ψ_s,

$$\psi_s = \psi_{s(1)} + \psi_{s(2)} + \cdots + \psi_{s(n)}, \tag{1}$$

where,

$$\psi_{s(k)}(\mathbf{x}_g, z_g | \mathbf{x}_s, z_s; \omega) = f(\omega)(\mathbf{F}^{-1}u_{p(1)}\mathbf{F}u_{s(1)})\cdots(\mathbf{F}^{-1}u_{p(k-1)}\mathbf{F}u_{s(k-1)})\int_{z_{k-1}}^{z_k}(\mathbf{F}^{-1}u_{p(k)}\mathbf{F}u_{s(k)})\frac{\omega^2}{c_{1(k)}^2}\alpha(\mathbf{x}_g, \mathbf{x}_s, z')dz', \tag{2}$$

and,

$$u_{p(k)}(\mathbf{k}_{gx}, \mathbf{k}_{sx}, \omega) = -\frac{e^{i(k_{gz(k)}+k_{sz(k)})(z_k - z_{k-1})}}{16 k_{gz(k)} k_{sz(k)}} \qquad u_{s(k)}(\mathbf{x}_g, \mathbf{x}_s, \omega) = e^{i\omega\left(c_{0(k)}^{-1}(\mathbf{x}_g)+c_{0(k)}^{-1}(\mathbf{x}_s)-2c_{1(k)}^{-1}\right)}. \tag{3}$$

In equations (1)-(3), the subscript (k) denotes the k^{th} depth in the discrete velocity model, the source location is $(\mathbf{x}_s, z_s)$, and the receiver location is $(\mathbf{x}_g, z_g)$, k_{sz} and k_{gz} denote the vertical wave-number at source and receiver, respectively, and $\mathbf{F}$ denotes the four-dimensional Fourier transform over $\mathbf{x}_g$ and $\mathbf{x}_s$. The operators $u_{p(k)}$ and $u_{s(k)}$ are, respectively, the phase-shift operator and split-step correction operator at the k^{th} depth. Finally, $f(\omega)$ is the frequency distribution of the source, and c_0 and c_1 are the migration velocity model. The latter being, roughly, the lateral average of the former.

More succinctly, we say that $\psi_s = \mathbf{L}\alpha$, and write the adjoint, $\alpha' = \mathbf{L}^H\psi_s$. For the sake of brevity we omit the details of the adjoint operator. In the context of least-squares migration, we call ψ_s *data*, and α the *model*.

Model and data space partioning for MPI processes

The previous section reviewed the operators used in least-squares migration. In our parallel implementation of these equations each MPI process evaluates the operators for some subset of frequencies, dividing the processing work amongst the MPI processes. In addition, the storage of α and ψ_s is shared amongst the distinct memory spaces of the MPI processes. In particular, α is partitioned along its depth axis, and ψ_s is partitioned along its frequency axis. This allows us to take advantage of the full storage capacity of a distributed system.

We consider the case of n_r MPI processes, n_m common midpoint gathers, n_ω realizations of frequency, and n_z realizations of depth. Further, we define the sets, $\mathcal{W}_k$ as the set of frequency indexes on the k^{th} MPI process, and $\mathcal{Z}_k$ as the set of depth indexes on the k^{th} MPI process. In particular, we write for the k^{th} MPI process,

$$j \in \mathcal{Z}_k = \{j \in \mathbb{Z} \,|\, (k-1)n_z/n_r \le j < kn_z/n_r\}, \tag{4}$$

and,

$$j \in \mathcal{W}_k = \{j \in \mathbb{Z}, i = 1, 2 \ldots n_w/n_r \,|\, j = k + (i-1)n_r\}. \tag{5}$$

The partitioning of depths (equation (4)) is straight-forward. The partitioning of frequencies in equation (5) takes into account the evanescent portion of the wave-field to maintain a load-balanced algorithm. In particular, we use a round-robin distribution for the frequencies. To illustrate, we consider a simple example with $n_\omega = 12$, and $n_r = 3$, and show the resulting distribution of frequencies in Table 1, where r_k denotes the k^{th} MPI process.

Forward operator for MPI processes

$r_1 \bigcap r_2 \bigcap r_3$	1	2	3	4	5	6	7	8	9	10	11	12
r_1	1			4			7			10		
r_2		2			5			8			11	
r_3			3			6			9			12

Table 1: We show the distribution of 12 frequencies across 3 MPI processes and the round-robin strategy (equation (5)).

The parallel organization of the forward and adjoint operators are similar. Here, we consider some aspects of the forward operator. We assume that α is already stored in the memory spaces of the MPI processes according to $\mathcal{Z}_k$ in equation (4). An analysis of equations (1) and (2) show that all depths are required to compute ψ_s for a single frequency. Therefore, a strategy for computing the forward operator must include communication of α between the memory spaces of the MPI processes. Fortunately, a recursion evident in equations (1) and (2) makes this communication feasible. We should emphasize that this recursion is not new to this abstract, but is the accepted method of computing the forward (demigration) operator (e.g., Kühl and Sacchi, 2003), regardless of whether the implementation is parallel or serial.

We find the recursion by a re-organization of equations (1) and (2), giving,

$$\psi_s(\mathbf{x}_g, z_g | \mathbf{x}_s, z_s; \omega) = f(\omega) u_{(1)} \left[\frac{\omega^2}{c^2_{1(1)}} \alpha_1 + u_{(2)} \left[\frac{\omega^2}{c^2_{1(2)}} \alpha_2 + u_{(3)} \left[\frac{\omega^2}{c^2_{1(3)}} \alpha_3 + \cdots + u_n \frac{\omega^2}{c^2_{1(n)}} \alpha_n \right] \cdots \right] \right], \tag{6}$$

where $\alpha_j = \alpha\left(\mathbf{x}_g, \mathbf{x}_s, z_j\right)$ and,

$$u_{(j)} = \mathbf{F}^{-1} u_{p(j)} \mathbf{F} u_{s(j)}. \tag{7}$$

This can be evaluated using the following recursion,

$$\psi_s = g_1 \tag{8}$$

$$g_{j-1} = u_{(j-1)} \left[\frac{\omega^2}{c^2_{1(j-1)}} \alpha_{j-1} + g_j \right], \tag{9}$$

where $j = n+1, n, \ldots 2$, and $g_{n+1} = 0$. Due to the recursion in equations (8) and (9), we use the following steps in our parallel implementation: (1) we communicate α from the memory space of the $k = n_r$ MPI process to all memory spaces. (2) For all $j \in \mathcal{Z}_{n_r}$, we compute the recursion in equations (8) and (9) for all frequencies ω and lateral space $\left(\mathbf{x}_g, \mathbf{x}_s\right)$. In doing so, we remember that the memory space of the l^{th} MPI process stores frequencies in the set $\mathcal{W}_l$. (3) We repeat steps (1) and (2) for α in the memory space of the $k = n_r - p$ MPI process for $p = 1 \ldots (n_r - 1)$.

Conjugate gradient algorithm for MPI processes

We consider the use of the forward and adjoint operators in the normal equations of a least-squares system,

$$\left(\mathbf{L}^H \mathbf{L} + \mu \mathbf{I}\right) \mathbf{m} = \mathbf{L}^H \mathbf{d}, \tag{10}$$

where $\mathbf{L}$ and $\mathbf{L}^H$ were defined previously. We call the vector $\mathbf{m}$ the model, made from realizations of α, and the vector $\mathbf{d}$ the data, made from realizations of ψ_s. We solve equation (10) for $\mathbf{m}$ using a parallel implementation of conjugate gradients (e.g. Gupta et al., 1995). The parallel implementation is fairly

simple, only requiring communication across the memory spaces of the MPI processes for the evaluation of norms required by the conjugate gradient algorithm; namely, distances computed in the Hilbert and Krylov spaces.

Results

We present scaling results of the proposed algorithm, showing an almost ideal speed-up as a function of the number of MPI processes. All tests were performed using the Marmousi data-set (e.g., Versteeg, 1994). The results are presented in Table 2. The code was run on a Linux Opteron cluster with InfiniBand interconnect hosted by the Western Canada Research Grid (www.westgrid.ca). If the algorithm scaled perfectly as a function of the number of MPI processes, then we would expect the number of MPI processes to match the Speed-up. For lack of space, we do not show, here, the migrated and forward modeled Marmousi data. Rather, choosing to show the scaling properties of the algorithm. For results of Least-squares migration applied to the Marmousi data (e.g. Versteeg, 1994), we refer the interested reader to, for example, Kuehl (2002).

MPI Processes	Forward operator		Adjoint operator		Least-squares inverse	
	Run-time	Speed-up	Run-time	Speed-up	Run-time	Speed-up
1	59.6	1.0	71.6	1.0	154.6	1.0
2	29.7	2.0	36.0	2.0	78.2	2.0
4	15.3	3.9	18.6	3.8	40.1	3.9
8	8.2	7.3	9.5	7.5	20.2	7.7
16	4.1	14.7	4.6	14.8	11.3	13.6
32	2.3	25.9	2.8	25.7	6.0	25.8

Table 2: Run-time and speed-up values for evaluating the forward, adjoint and least-squares inverse (using the conjugate gradient method). All run-times are given in minutes.

Discussion

We implement pre-stack split-step wave-field modeling and migration operators for parallel distributed computing systems. In addition, we implemented a distributed parallel version of the conjugate gradient algorithm. The primary purpose of this work is to accelerate our pace of research, reducing the turn-around time when running regularized migration tests. The next step of this work will be a parallel implementation of the iterative re-weighted least-squares method, allowing for quick tests of regularized migration with non-linear model priors.

References

Gupta, A., Kumar, V., and Sameh, A., 1995, Performance and scalability of preconditioned conjugate gradient methods on parallel computers: IEEE Transactions on Parallel and Distributed Systems, 6, 455-469.

Huang, L., Fehler, M., and Wu, R., 1999, Extended local Born Fourier migration method: Geophysics, 64, 1524-1534.

Kuehl, H., 2002, Least-squares wave-equation migration/inversion: PhD Thesis, University of Alberta.

Kühl, H., and Sacchi, M., 2003, Least-squares wave-equation migration for AVP/AVA inversion: Geophysics, 68, 262-273.

Popovici, A., 1996, Prestack migration by split-step DSR: Geophysics, 61, 1412-1416.

Santos, L., Schleicher, J., and Tygel, M., 2000, Modeling, migration, and demigration: The Leading Edge, 19, 712-715.

Stoffa, P., Fokkema, J., de Luna Freire, R., and Kessinger, W, 1990, Split-step Fourier migration: Geophysics, 55, 410-421.

Versteeg, R., 1994, The Marmousi experience: velocity model determination on a synthetic complex data set: The Leading Edge, 18, 86-91.

Wang J., 2005, 3-D least-squares wave-equation AVP/AVA Migration of common azimuth data: PhD Thesis, University of Alberta.