Overcoming computational cost problems of reverse-time migration

Zaiming Jiang*
University of Calgary, Calgary, Alberta
jianz@ucalgary.ca

and

Kayla Bonham, John C. Bancroft, Laurence R. Lines
University of Calgary, Calgary, Alberta

Summary

Prestack reverse-time migration is computationally expensive. Program run times are long, in terms of the total number of CPU cycles, and it requires large amounts of hard disk free space. To accelerate computing, we do parallel processing using Intel Threading Building Blocks (TBB) and multi-core computers, for both the forward-time modelling and reverse-time migration phases of the computation. To solve the problem of limited free disk space, we use a technique that may seem counter-intuitive: the forward modelling phase is done twice instead of once.

Introduction

Elastic wave modelling based on finite-difference methods is time consuming. For example, it took Martin (2004) a total of 70,000 hours, or approximately 8 CPU years to do elastic wave modelling using the Marmousi2 model.

Prestack reverse-time migration needs even longer computational time. Gavrilov, Lines, Bland, and Kocurko (2000) have already practiced parallel computing to accelerate reverse-time migration. They used the message-passing interface (MPI) to develop the "distributed parallel implementation" and carried out the computing on a cluster computer with many processors.

This report describes another method of parallel computing, developing software using Intel TBB, a C++ template library introduced in 2006 for writing software programs that take advantage of multi-core processors, and carrying out computation on multi-core processors, which have been widely used both in cluster computers and personal computers ever since Intel developed its first dual-core processor in 2005. As pointed out by Lines, Castagna, and Treitel (2001), the "advances in computer hardware have had a big impact on how we operate." Multi-core processor parallel computing will become more and more popular.

In addition to CPU time, disk space is another important computational resource when we do reverse-time migration. To solve the problem of limited free disk space, we use a technique that may seem counter-intuitive: the forward modelling phase is done twice instead of once.

Computational cost problems and some solutions

Computational time

Prestack reverse-time migration is more computationally intensive than seismic modelling since it involves not only forward modelling, but also reverse-time extrapolation and imaging. If we do reverse-time migration on the Marmousi2 model, for which the number of numerical nodes in the direction is 13601 and the number in the direction is 2801, the total time will be at least doubled, i.e., up to 16 CPU years.

Our solution is to use parallel processing to accelerate the computation. The implementation of parallelization was first done on a dual-core PC, and then the parallelized code was ported onto an 8-CPU shared memory computer, available to us as a single node of Gilgamesh, CREWES' new cluster computer (Figure 1).

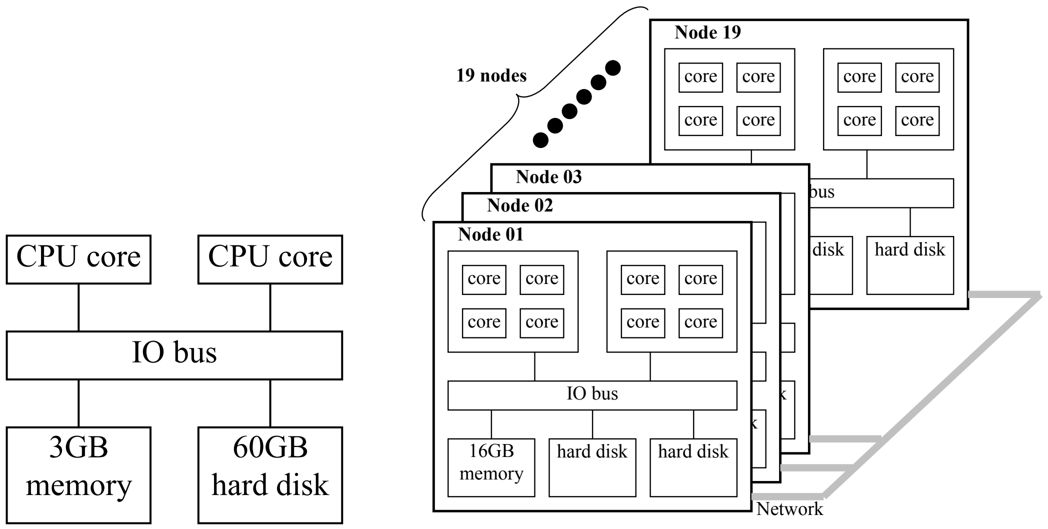

Figure 1: The logical architecture of two multi-core computers: a dual-core PC (left), and Gilgamesh (right).

Intel® Threading Building Blocks (TBB) is used to parallelize the modelling and reverse-time migration application. Intel TBB is a C++ template library for writing software programs that take advantage of multi-core processors. There are two builds of it: a commercial build and an open source build. According to Intel's website, "these are built from the same source code, the only real difference is the license and support offering". What we use is the open source build. The most recent version is 2.2; what we use is version 2.1, which was released in June 2008.

Once the Intel TBB package is installed, the modelling and migration programs, which are written in C++, are parallelized using the header files and the library provided by the TBB package.

The simplest form of parallelization is a loop of C++ template function tbb::parallel_for. It looks like a "for" loop in C/C++, but it involves the definition of an iteration space. The tbb::parallel_for compares the iteration space size and the available CPU cores, and then breaks down the iteration space into chunks. Then the function tbb::parallel_for runs each chunk on a separate thread. For the prestack reverse-time migration, the intuitive choice of the iteration space is by shot numbers.

The efficiency of parallelization was first tested by running the modelling and reverse-time migration program on the dual-core PC using a simplified model which contains 872 nodes in the x direction and 366 nodes in the z direction. For 10 shots, the parallelized program employs the dual-core CPU and the total computation time is reduced by 44.7% (Figure 2).

We then tested a modelling program on a Gilgamesh node with eight CPU cores. The subsurface model contains 3000 nodes in the x direction and 800 nodes in the z direction. For 16 shots, the parallelized program employs the eight CPU cores and the total computation time is reduced by 75.3% (Figure 2).

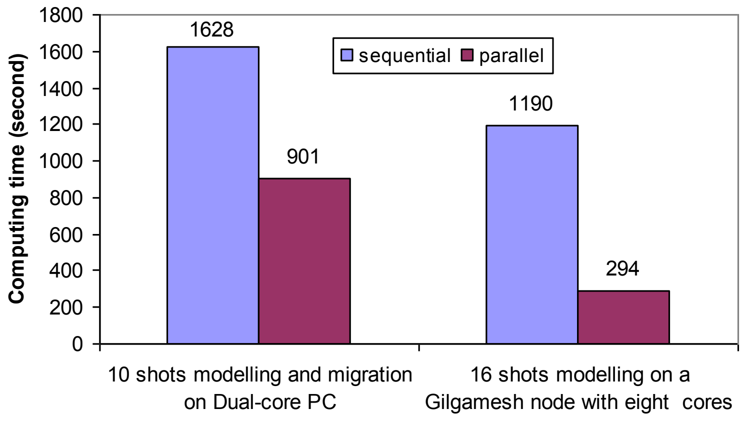

Figure 2: Computational costs of sequential and parallel programs on a Dual-core PC, and a Gilgamesh node with eight CPU cores.

Hard disk free space

The problem of limited hard disk free space arises when we try to calculate source-normalized crosscorrelation imaging condition. The imaging condition is

$$image(x,z)=\frac{\sum_{time}S(x,z,t)R(x,z,t)}{\sum_{time}S^2(x,z,t)}, \qquad (1)$$

where $S(x,z,t)$ and $R(x,z,t)$ are, respectively, the source wavefield produced by modelling and the receiver wavefield produced by reverse-time extrapolation. Imagine that we have decided that we should compute the forward modelling by time steps t = [1, 2, 3, …, T]. When we reach the last time T, we begin the reverse-time migration phase by time steps t = [T, T-1, T-2, …, 1]. At each step in the reverse-time calculation, the imaging condition requires crosscorrelation with the corresponding in the forward time calculation. This means stepping backward through the snapshots of the wavefield representation, which had been previously computed in the forward direction. Unfortunately, the disk space required to store every step in the forward calculation, would be prohibitive.

The solution to this problem can be to use CPU time, doing modelling twice instead of once, to keep the disk space requirements within available limits. During the first forward modelling phase, instead of saving all the wavefield snapshots (subsurface particle horizontal and vertical velocities) for each time t = [0, 1, 2, 3, …, 9599], we save the wavefield state (subsurface particle velocities and stresses) for only every 1000th one, i.e., for t = [1000, 2000, 3000, …, 9000]. When we work backwards in the reverse-time migration for t = [9599, 9598, 9597, …, 1, 0], we can re-model each block of 1000 from the stored wavefield state at the time it is needed for the crosscorrelation. For example, we would re-compute snapshots for time t = [3001, 3002, 3003, …, 3999] from the stored wavefield state at time t = 3000. Thus, without storing all the model snapshots at every time t onto disk, the imaging condition can be implemented, although the modelling has to be done twice (Figure 3).

Conclusions

Modelling and reverse-time migration based on finite-difference methods are compute-intensive. The challenges are the long computational time and the need of large hard disk free space. To accelerate computation, parallel computing is implemented using Intel TBB and multi-core computers; to overcome disk space limitations, the modelling part of the reverse-time migration is done twice, instead of once, in “chunks” whose size is optimized to avoid exceeding the available disk space.

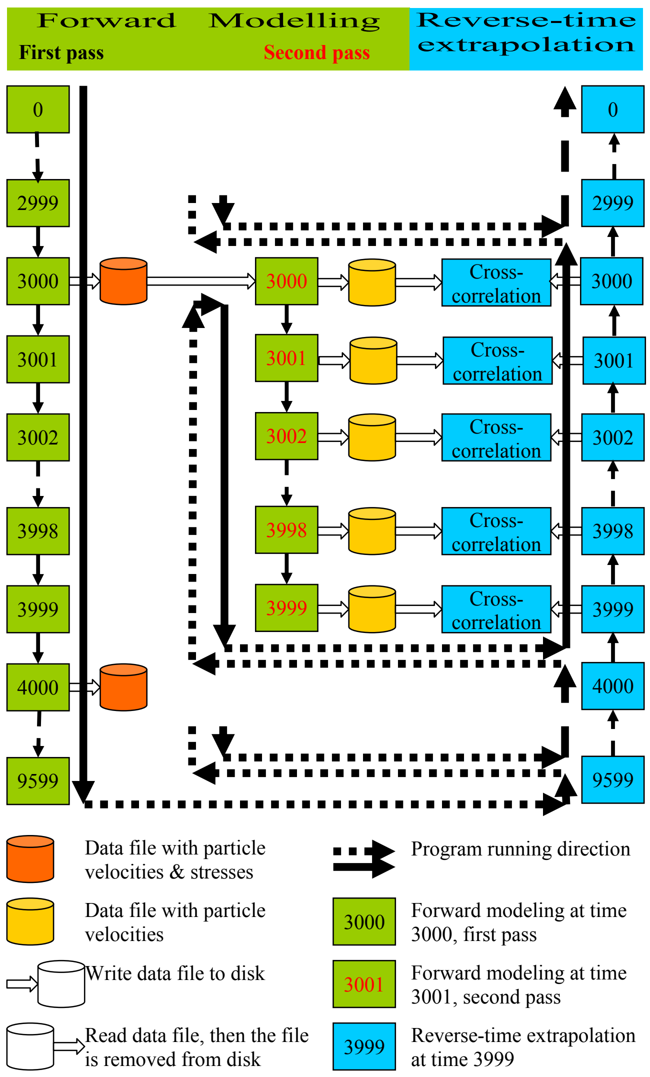
Forward Modelling
First pass
Second pass
Reverse-time extrapolation
0
2999
3000
3001
3002
3998
3999
4000
9599
Cross-correlation
Data file with particle velocities & stresses
Data file with particle velocities
Write data file to disk
Read data file, then the file is removed from disk
Program running direction
Forward modeling at time 3000, first pass
Forward modeling at time 3001, second pass
Reverse-time extrapolation at time 3999

Figure 3: Do modelling twice instead of once, to keep the disk space requirements within available limits.

Acknowledgements

The authors gratefully acknowledge the support of CREWES sponsor companies and various CREWES staff and students.

References

Bonham, K., Hall, K.W., and Ferguson R.J., 2008, The epic of Gilgamesh: CREWES' new cluster computer, CREWES research report, 20, 70.1-70.12.

Gavrilov, D., Lines, L., Bland, H., Kocurko, A., 2000, 3-D depth migration: parallel processing and migration movies, The Leading Edge, 19, 1282-1284.

Jiang, Z., Bonham, K., Bancroft, J.C., and Lines, L.R., 2009, Overcoming computational cost problems of reverse-time migration, 21th Annual CREWES Sponsors Meeting.

Lines, L.R., Castagna, J.P., and Treitel, S., 2001, Geophysics in the new millennium, Geophysics, 66, 14.

Martin, G.S., 2004, The Marmousi2 model, elastic synthetic data, and an analysis of imaging and AVO in a structurally complex environment, MSc thesis, University of Houston.