N87-26560

Reverse Time Migration - A Seismic Processing Application on the Connection Machine

Rolf-Dieter Fiebrich
Thinking Machines Corporation
Cambridge, MA 02142

Abstract

This article describes the implementation of a reverse time migration algorithm on the Connection Machine, a massively parallel computer. Essential architectural features of this machine as well as programming concepts are presented. The data structures and parallel operations for the implementation of the reverse time migration algorithm are described. The algorithm matches the Connection Machine architecutre closely and executes almost at the peek performance of this machine.

Introduction

This paper describes the implementation of a reverse time migration algorithm on a massively parallel computer. These computers, with thousands of processing elements are entering the marketplace and offer better cost performance than conventional mainframes, and more importantly, promise to reach significantly higher absolute performance levels in the coming years than those which can be realized by conventional architecures. This paper discusses how this technology can be utilized efficiently for extremely computation-intensive algoritms in seismic processing.

This paper focuses on the implementation of a reverse time migration algorithm for 2D seismic processing. The close match of this application with the Connection Machine architecture results in substantial speedups compared to conventional mainframes and strongly suggests that this machine puts 3D seismic processing in reach.

The remainder of this paper summarizes the architecture and programming issues of massively parallel computers, followed by a discussion of the reverse time migration algorithm and its implementation. A brief performance summary for the program follows. The paper concludes with comments on what massively parallel computers can do today for seismic processing problems and the promise of this technology for the future.

Architecture and programming of a massively parallel machine

A massively parallel computer can be viewed as a machine which can operate on thousands of data objects at once, whereas a conventional computer operates on one data object at a time. If an application permits for instance to operate on all elements of a large vector or matrix or on all nodes or edges of a graph in parallel, then substantial execution speed improvements can be obtained if a large number of processors is available, ideally one processor for each data object.

A program for such a computer looks very much like a program for a conventional computer, except that certain program variables are declared to be parallel variables which means that operations on those variables can take place in parallel. Instead of using a loop statement to process all the elements of a vector or a matrix or a graph, one uses a select statement for all the elements and all statements in the body of the select statement are performed in parallel on the selected set, like the statements in the body of a loop statement are performed on each element visited.

The above sketches an idealized programming model. Physical parallel machines usually have some limitations for implementing this model efficiently. For instance, processors have to be assigned to data objects at compilation time and cannot be reassigned during execution time. Also the computation which a processor performs an a data object that is assigned to it in general involves access to data ojbects assigned to other processors. The execution time of "local access" vs. "nonlocal" access is of course different. Programmers need to take this into account for writing efficient programs, but this is not much different from optimization considerations on conventional computers.

A physical computer which implements this model is the Connection Machine [Hillis, 1985]. Most of the experience reported in this paper was gained on this machine. The Connection Machine uses a conventional host which provides all the infrastructure for program development and communicating with other computers. This host has, however, several important enhancements. It has a significantly enlarged memory which is partitioned in equal chunks and each chunk has a processor associated with it. These processors cannot just access data in their part of the memory, but can also access the entire memory of the machine. All processors can perform these accesses in parallel. The processors of the Connection Machine can be viewed as an extension of the execution unit of the host. Figure 1 illustrates the architecture of the Connection Machine.

The Connection Machine model which was used for most of the work described in this paper has a maximum of 32 MBytes of memory and 64 K processors. The number of processors which a programmer sees is typically significantly larger than the number of physical processors. The system supports a virtual processor concept. The host for the Connection Machine can be either a VAX or a Lisp Machine. Parallel programming concepts as described above are implemented as straigtforward extensions of C and Lisp.

Reverse time migration

Finite-difference scheme

The reverse time migration process is well known and well documented in the literature [McMechan,1983]. Conceptually, reverse time migration, as with all depth migrations, involves the transfer of data from the

$(y, z = 0, t)$ time plane to the $(y, z, t = 0)$ depth plane. This concept is illustrated graphically in Figure 2. For the acoustic case, which is discussed here, wave propagation through the earth is governed by the acoustic wave equation. Attention is further restricted to the two dimensional case. Hence, the wave equation has the form

$$U_{yy} + U_{zz} = \frac{1}{v^2(y,z)} U_{tt} \tag{1}$$

where U is the acoustic wave field. Reverse time migration is based on an exploding reflector concept wherein the interface between rock strata explode with sound at time t_0. From that moment on waves propagate according to the above wave equation at velocities one-half their true velocity in accordance with the exploding reflector model. If acoustic measurements are made at various places along the earth's surface for all subsequent time we have the equivalent of a zero offset stacked section. Migration is implemented by reversing the process and exciting mesh points at $z = 0$ with the time reversed recorded signals. Since the wave equation is ambivalent to the direction of time this is no problem. The recorded signals act as boundary values in the numerical solution of the wave equation.

Discretization of the acoustic wave equation in time and space follows traditional numerical methods [Dablain,1986]. Using these methods equation (1) may be approximated by a fourth order spatial and second order temporal operator. The notation

$$U_{i,j}^k = U(y_i, z_j, \hat{t}_k)$$

is used in writing the difference operator (where $\hat{t}_k$ refers to reverse time) as

$$\begin{aligned} U_{i,j}^k &= 2U_{i,j}^{k-1} - U_{i,j}^{k-2} \\ &\quad + \frac{A^2}{12} [\, 16(U_{i+1,j}^{k-1} + U_{i-1,j}^{k-1} + U_{i,j+1}^{k-1} + U_{i,j-1}^{k-1}) \\ &\quad - 60U_{i,j}^{k-1} - U_{i+2,j}^{k-1} \\ &\quad - U_{i-2,j}^{k-1} - U_{i,j+2}^{k-1} - U_{i,j-2}^{k-1} \,] \end{aligned} \tag{2}$$

where

$$A = v(y,z)\frac{\delta t}{\delta y}.$$

For purposes of simplifying the explanation of the implementation it will be useful to have the simpler second order spatial operator as well. The difference equation is [McMechan,1983]

$$\begin{aligned} U_{i,j}^k &= 2(1 - 2A^2)U_{i,j}^{k-1} - U_{i,j}^{k-2} \\ &\quad + A^2 [\, U_{i+1,j}^{k-1} + U_{i-1,j}^{k-1} \\ &\quad + U_{i,j+1}^{k-1} + U_{i,j-1}^{k-1} \,]. \end{aligned} \tag{3}$$

Consider equation (3). There are three time steps involved and three spatial points in each direction. This equation is illustrated graphically in Figure 3 where the three time steps are represented as three depth planes in reverse time, $\hat{t}_i$. Each illustrated plane is a small part of a larger mesh on which the finite-difference scheme is carried out. The data values required to compute the current grid point value, $U_{i,j}^k$, are identified as the grid point's own previous and second previous value, and its immediate neighbors' previous value. Again, refer to Figure 3 for a graphical representation of the process. Initially, the previous and second previous depth planes are zero. This corresponds to the assumption that all signals are recorded until they are identically zero. Conceptually the reverse time migration proceeds as follows; 1) load the boundary value corresponding to time step $\hat{t}_0$, 2) compute all the grid points in the current depth plane, 3) push the stack of depth planes so that the current plane becomes the previous plane and the previous plane becomes the second previous plane, 4) repeat until time $\hat{t}_{max}$ (or t_0) is reached. The final solution will be an acoustic wave field reconstruction of the exploding reflectors imaged at time t_0 for all depths.

Because of reflections from the boundary of the computational grid it is desirable to implement absorbing boundary conditions along the two edges and the bottom [Clayton,1977]. When the wave field in the depth plane is computed in step (2) above, an absorbing boundary difference scheme must be used on the edges.

Parallel Implementation

Finally, we are ready to discuss the paralle algorithm for reverse time migration. We assign a processor to each grid point in the finite-difference mesh Figure 3. To compute the current value in a processor requires that the processor reference its own local memory for the previous and second previous value. It also requires that the processor get the previous value from each of its neighbors. This is exactly what is done in mapping the algorithm onto the Connection Machine. The time axis in Figure 3 corresponds to the memory axis of each processor. The time section is usually larger, in terms of the number of samples per trace, than the depth section. As a result, the time section is incrementally fed into the Connection Machine.

In generating a new data value in each processor (at each grid point) two of the memory accesses are local and the rest are non-local. The non-local accesses require utilization of the general communication system. Four such accesses are needed for the second order finite-difference operator. In addition, to load the boundary value at the beginning of each time step requires another non-local memory access. To implement the absorbing boundary conditions the processors on the edge of the computational grid are selected and use an absorbing boundary finite-difference operator to compute a new value. Consider a typical unmigrated seismic section. There might be 1024 traces and 2500 time samples. If 512 depth steps are desired, there must be 512K processors using the purposed parallel approach. Since there are only 64K processors, virtual processors must be used for almost all practical cases. Using a virtual processor ratio of 8:1 will provide the required 512K processors.

Timing

The total execution time for a data set of the size 1425 x 625 x 2500 is 441 seconds. This computation takes several hours on a large mainframe.

The whole issue of timing is obviously machine specific and is instantly out of date due to hardware improvements. The point is that parallel computers can compete with the fastest serial supercomputers. In addition, the very fact that the Connection Machine has floating point instruction times measured in hundreds of microseconds instead of tens of nanoseconds points provides significant technological improvement.

Conclusions

Results from the reverse time migration implementation and from the many other non-seismic applictions that have been programmed indicate that massively parallel architectures are viable and can perform at supercomputer levels.

In the specific case of a reverse time migration algorithm, performance improved by two orders of magnitude relative to a VAX 785. This improvement is achieved despite relatively simple individual processors in the fine-grain computer. The fact that there are 64K such processors far outweighs the fact that each processor is slow. Overall, vast speed improvements are possible both for reverse time migraion in particular and for seismic processing in general. One exciting possibility is that the dream of interactive interpretation/processing might be realized. Imagine a work station where an interpreter can repeatedly migrate a section, trying different velocity models each time. In so doing, the iterative process of converging on a satisfactory depth model might take a few hours instead of many days. In addition, because the interpreter would be intimately involved in the processing, the final result would be better than with batch processing. This is just one computational problem that parallel computers can address.

Acknowledgements

The author wishes to thank Rob Fricke and Monica Wong for the implementation of the reverse time migration algorithm on the Connection Machine. Many thanks go also to Olivia Fiske for her quick help in preparing this document.

References

[Clayton,1977] Clayton, R. W., and Engquist, B., 1977, Absorbing boundary conditions for acoustic and elastic wave equations: Bulletin fo the Seismological Society of America, **67**, 1529-1540.

[Dablain,1986] Dablain, M. A., 1986, The application of high-order differencing to the scalar wave equation: Geophysics, **51**, 54-66.

[Hillis,1985] Hillis, W. D., 1985, The Connection Machine: M.I.T. Press.

[McMechan,1983] McMechan, G. A., 1983, Migration by extrapolation of time-dependent boundary values: Geophysical Prospecting **31**, 413-420.

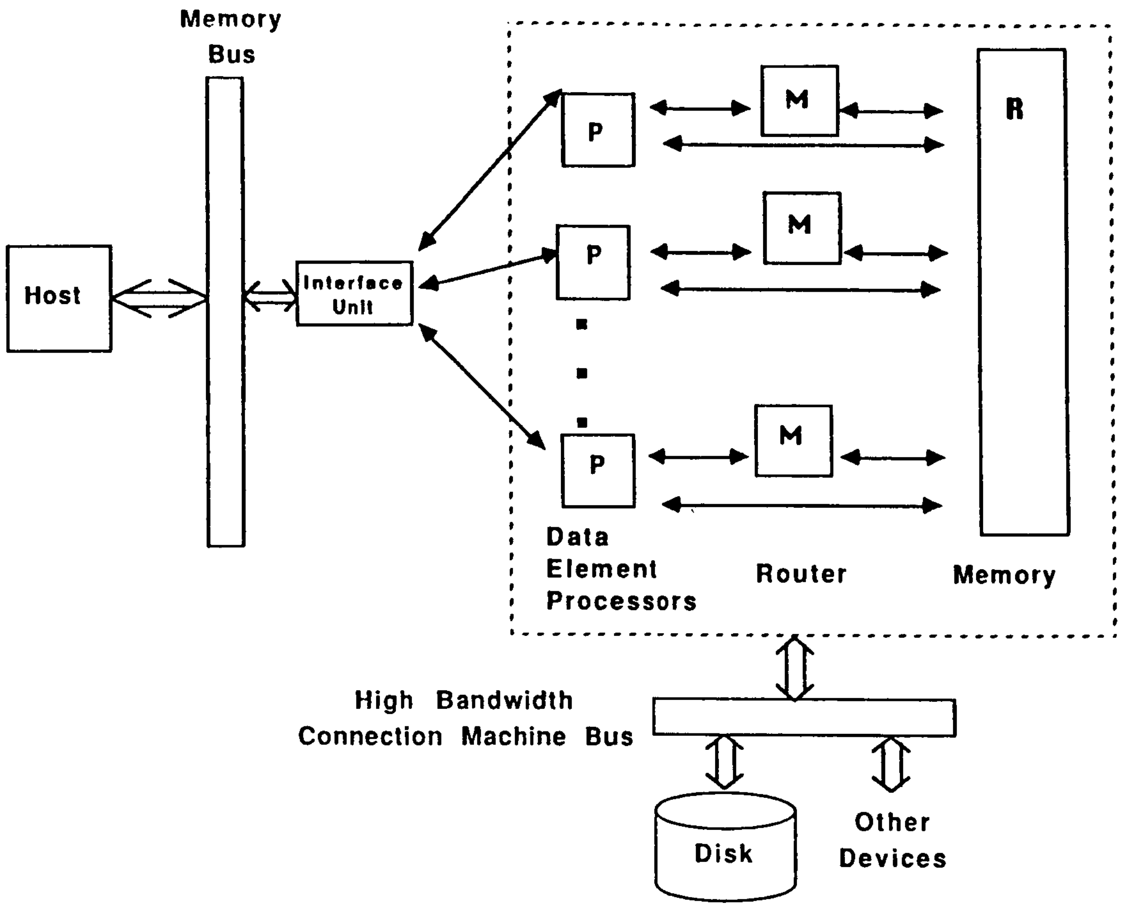

Figure 1: Architecture of the Connection Machine System

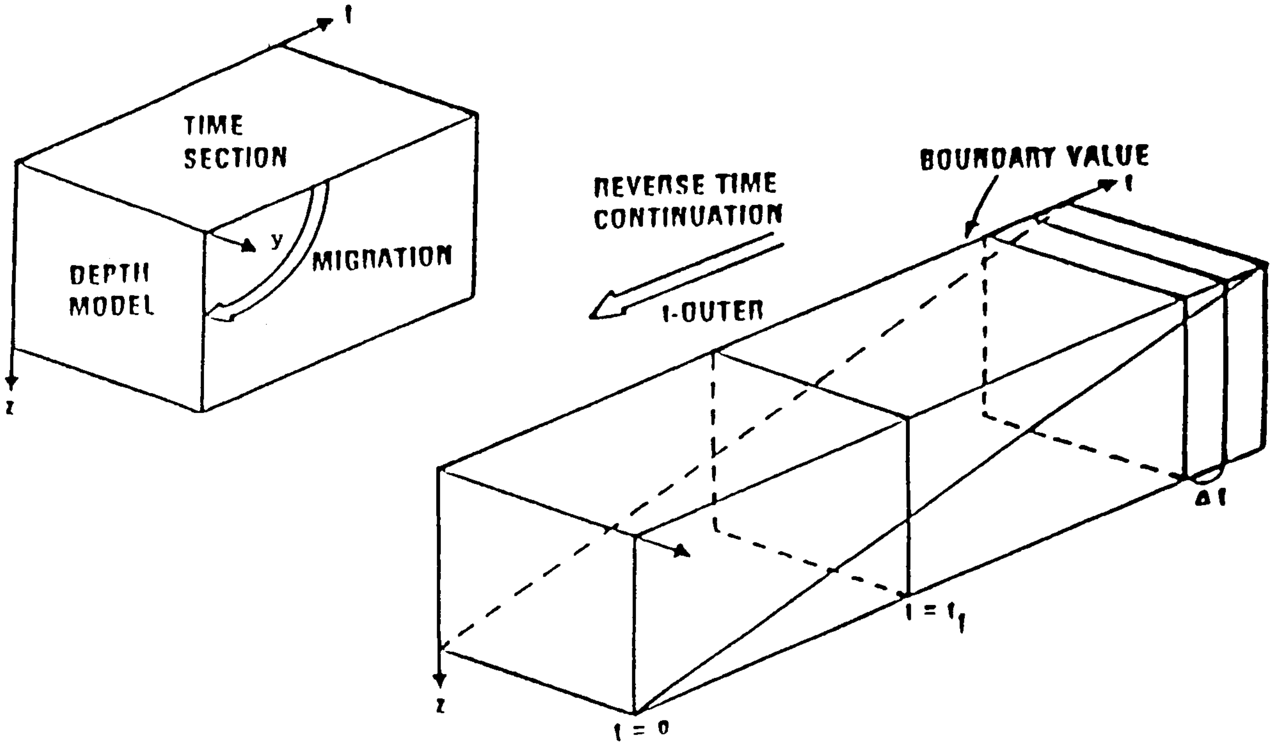

Figure 2: Reverse Time Migration

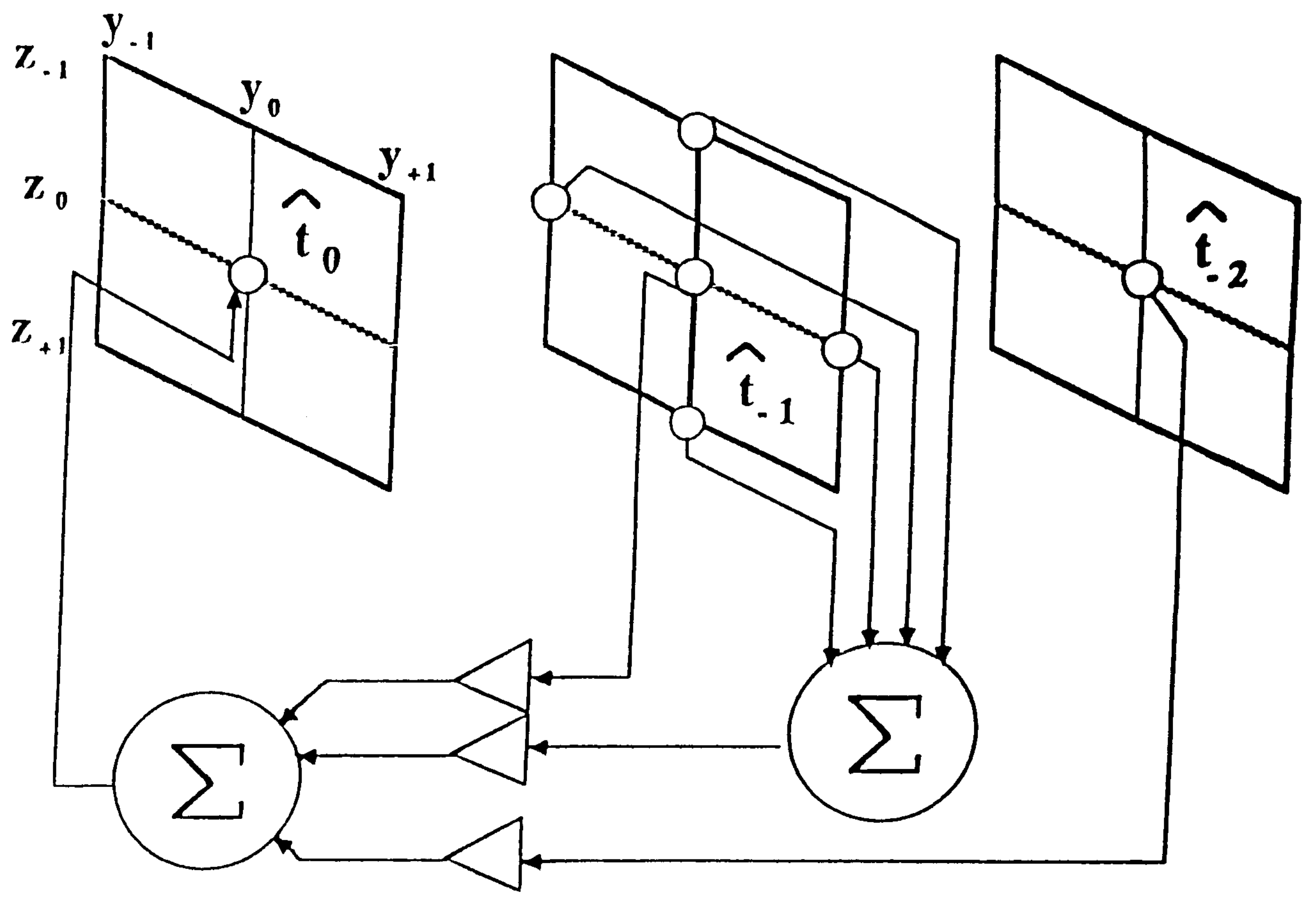

Figure 3: Computation of Second Order Difference Operator