Industrial-Scale Reverse Time Migration on GPU Hardware

Darren Foltinek, Daniel Eaton, Jeff Mahovsky, Peyman Moghaddam, Ray McGarry, Acceleware Corp.*

Summary

We give an overview of an implementation of Reverse Time Migration (RTM) on heterogeneous multi-core hardware based on Graphics Processing Units (GPUs). We demonstrate the clear advantages of GPU-based hardware for RTM, not only in terms of performance for small to mid-scale problem sizes, but also for imaging 3D volumes on a scale appropriate to full Wide-Azimuth (WAZ) or Rich-Azimuth (RAZ) surveys in areas such as the Gulf of Mexico. We also point to advantages of GPU-based hardware in terms of energy efficiency.

Introduction

RTM is a state-of-the-art technique for imaging subsurface geological structures that fully handles the two-way wave equation to improve imaging in areas of complex geology. The superior imaging quality of RTM comes at the cost of extremely intensive computational requirements which have traditionally limited its widespread adoption.

RTM was first introduced in the 1980s (Baysal et al., 1983; Whitmore, 1983). The theory is well-known, so we do not dwell upon it here, other than to outline (in the next section) the two major barriers to implementing a practical RTM system for real large-scale imaging projects of interest today within the seismic industry.

Despite the enormous computational requirements of RTM, it has recently been receiving renewed attention from the seismic imaging community, as evidenced by its increasing prominence in the relevant literature. This renewed interest has been fuelled both by a need for better imaging techniques and by advances in computer hardware, particularly with regard to processor speed and memory bandwidth.

Otigosa et al. (2008) performed an evaluation of several high performance computing (HPC) platforms to assess their suitability for RTM. They considered both traditional CPU-based (homogeneous) hardware and "heterogeneous" hardware based on the IBM Cell/BE processor (see also Araya-Polo et al., 2009). These authors concluded that heterogeneous hardware does offer clear advantages for RTM over the more traditional CPU-only systems.

In this paper we are also concerned with implementation of RTM on heterogeneous hardware, but our approach uses the Graphics Processing Unit (GPU) as a co-processor. Historically, GPUs were dedicated graphics rendering devices for a personal computer, workstation, or game console. The highly parallel nature of modern programmable GPUs, coupled with their exceptionally high memory bandwidth, makes them extremely effective general-purpose processors for a range of scientific problems.

Each GPU contains up to 30 multiprocessors, each decomposed into a number of stream processors with generous on-board memory for user-managed cache. For example, recent NVIDIA GPUs such as the Tesla C870 and C1060 have totals of 128 and 240 stream processors respectively. Each stream processor is fully capable of executing integer and floating point arithmetic. GPUs achieve high performance when thousands of threads execute concurrently, giving them a significant advantage over CPU-only hardware. The C1060 has 4 GB of on-board memory and a global memory bandwidth of 102 GB/s.

In recent years GPUs have been successfully used in a number of scientific application areas such as airflow modeling (Fan, 2004) and electromagnetic simulation (Sypek and Mrozowski, 2008; Schneider et al., 2007; Krakiwsky et al., 2004). In the area of electromagnetic simulation in particular, GPUs are rapidly becoming an accepted technology.

Here we present a novel software system which leverages GPU technology for RTM. Our aim is to show that GPUs have the potential to deliver the performance required for RTM to become a practical tool for seismic imaging of large volumes, without the need to resort to low-frequency or other compromising approximations.

RTM Methodology

At the core of RTM is wave propagation according to the two-way wave equation. Scalar acoustic wave propagation of 2N-order in space and second-order in time can be discretized as

$$U_{n+1} = 2U_n - U_{n-1} + \Delta t^2 c^2 \nabla^2 U_n \qquad (1)$$

with

$$\nabla^2 U_n = \sum_r \sum_{i=1}^{N} a_0 + a_i U_n(r \pm i\Delta r) \qquad (2)$$

where $r \in \{x, y, z\}$, U_n is the wavefield at timestep n and c is the wave speed. Efficient implementation of the update equations (1) and (2) poses the first major hurdle to the design of practical RTM for large-scale or high frequency imaging. Key aspects of our implementation are described in the next section.

Industrial-Scale Reverse Time Migration on GPU Hardware

In addition to the main wave propagation kernel, RTM requires that the source wavefield, which is calculated by a forward recursion in time, be made accessible in reverse-time order for correlation with the receiver wave. This poses the second major hurdle for practical RTM implementations. Various methodologies have been put forward for addressing this issue, for example the optimal checkpointing approach (Symes, 2007). We have developed a novel approach to this problem, which removes I/O bottlenecks without impacting image quality.

GPU Implementation

Implementation of general purpose scientific software on NVIDIA GPU architecture was made significantly easier by using the CUDA programming language (NVIDIA, 2008), which is an extension to standard C. CUDA removes the need to program in graphics-specific languages such as OpenGL. The GPU-specific components of the software presented here were programmed in CUDA Version 2.1.

While it is possible to take advantage of the multicore nature of GPUs using naive programming techniques implemented within CUDA, harnessing of the true potential of the GPU requires that very careful thought be given to optimizing memory access. This is particularly important for the current application since finite difference wave propagation (as formulated in equations (1) and (2)) is a problem of very low computational complexity, meaning that the speed of calculations is less important than the speed with which data can be delivered to those calculations.

To achieve maximum performance, memory access redundancy needs to be minimized. In other words, for a given number of finite difference cells for which the wavefield is updated, the total number of wavefield (and velocity) values read should be kept as small as possible.

We have implemented several update schemes for the basic wave propagation kernel, each with different characteristics, strengths and weaknesses. In generating the performance results presented here we used the update scheme described by Micikevicius (2009). The essential idea behind this scheme for 3D wave propagation is to assign threads to compute new wavefield values for a specified column along the slowest varying dimension, which we shall assume is the z-dimension. Relevant wavefield values at neighboring locations in the x- and y-dimensions are stored as 2D tiles in shared memory, which is accessed significantly faster than global memory. Relevant data at neighboring locations in the z-dimension are stored in local variables, which are simply shifted as the calculation proceeds in the z-dimension.

In addition to the main wave propagation kernel, all other components of the RTM system (such as absorbing boundary conditions, imaging condition, I/O, etc.) need to be optimized for the GPU. This optimization process typically proceeds according to the following steps:

- Recasting the logic/arithmetic in a parallelizable form;
- Minimizing memory access redundancy;
- Maximizing shared and local memory usage;
- Optimizing parallel execution parameters;
- Optimizing for different runtime parameters;
- Reusing computation for repeated steps.

We have implemented the full RTM system on both CPU (Intel/AMD) and NVIDIA hardware comprising (i) a single GPU, (ii) multiple (up to 4) GPUs in the same computing node and (iii) multiple (>4) GPUs split across different nodes. The single-GPU implementation simply follows the approach outlined above. For multi-GPU implementation within the same node we use standard domain decomposition techniques. For the multi-node implementation we use MPI over an Infiniband network for communication between nodes.

2D Example and Performance Analysis

To illustrate the performance of our RTM implementation for 2D applications we imaged the BP benchmark model as shown in Figure 1. The model is 67.5 km wide and 12 km deep. There are a total of 1348 shots with 1201 receivers per shot. The acquisition geometry is marine with the receivers towed behind the shot.

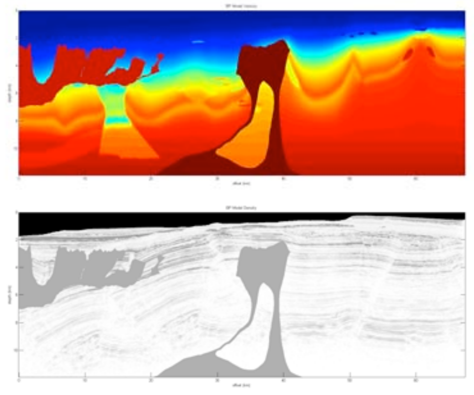

Figure 1 - Velocity (top) and density (bottom) for the BP benchmark model. Courtesy BP.

Industrial-Scale Reverse Time Migration on GPU Hardware

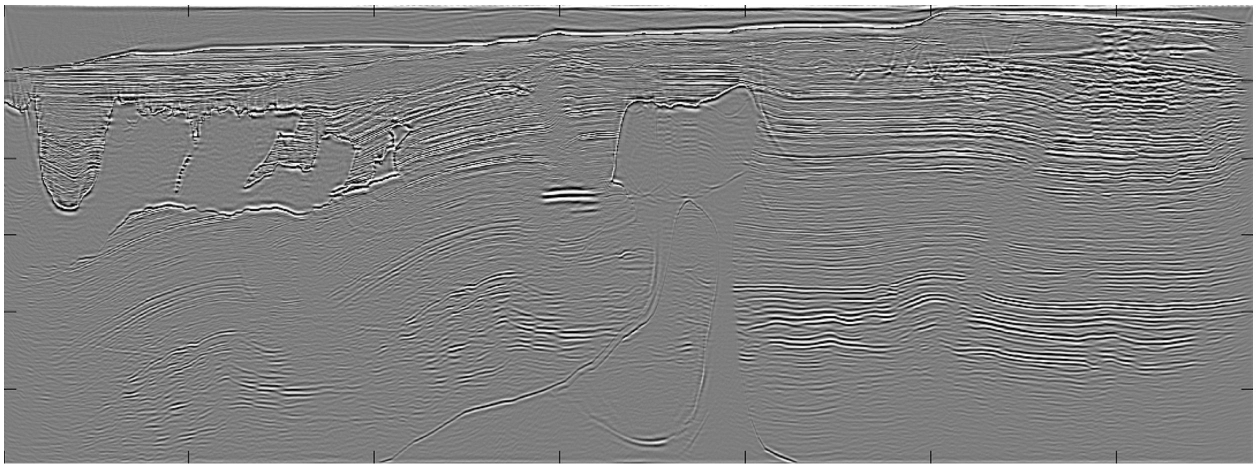
Figure 2 - RTM image for BP benchmark model

The model is discretized to 12.5m grid spacing along the offset and 6.25m spacing in depth, resulting in 5400x1910 cells. A sponge boundary (Dablain, 1986) is used on all faces. The simulation time step is 0.5ms, giving a total of 24000 timesteps. A 2nd-order in time, 8th-order in space finite difference scheme is used for all wave propagation calculations. Figure 2 shows the migrated image.

A single-shot migration for this model on a single NVIDIA C1060 GPU took 8.7 minutes. The same migration took 60.1 minutes on an 8-core CPU-only machine with a 2.0GHz Xeon E5405 processor.

Domain decomposition

3D migration volumes typically require more than the 4GB of memory available on a single GPU, and need to be distributed among multiple CPUs or GPUs. Performance is increased as the added computational hardware works in parallel on the wave propagation and other computations.

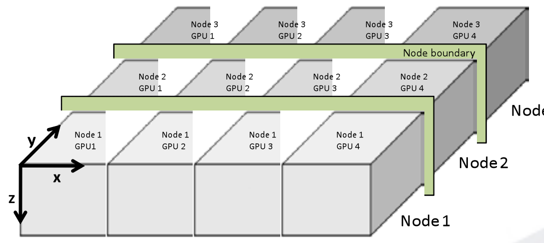

Figure 3 - Partitioning over 12 GPUs on three nodes

Our current implementation of RTM allows the migration volume to be partitioned along two axes. One axis is divided among the computational hardware (either CPU cores or GPU devices) on a single compute node. The other axis is divided among several compute nodes.

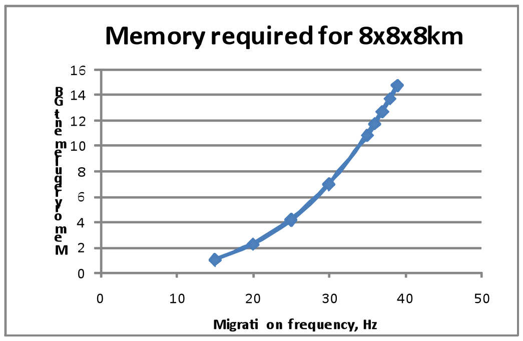

Figure 4 - RTM memory requirement as a function of migration frequency

The second, more important, benefit of volume partitioning is that the maximum size of the migration volume is increased beyond the limits of a single node's CPU or GPU RAM. This is very important because the memory requirements for a given Earth volume scale with the cube of the migration frequency, as shown in Figure 4.

3D Performance Analysis

To analyze 3D RTM performance and scalability across multiple GPUs and CPUs, constant-velocity models were constructed to represent imaging a given geologic volume at certain frequencies. A single shot migration was

Industrial-Scale Reverse Time Migration on GPU Hardware

performed on each model. Multiple shot performance was not measured, since it is relatively trivial using standard computer-cluster management tools to efficiently parallelize seismic migration in the shot domain.

Performance of the RTM is quoted in Imaged Megacells per second (IMC/s). This unit represents the number of output migration volume cells computed per second.

$$\mathrm{IMC/s} = \frac{(output\ image\ volume)\bullet(timesteps)}{(computational\ runtime)} \quad (3)$$

The runtime includes source and receiver wave propagations, with absorbing boundaries, calculation of illumination and imaging conditions, as well as storage and recovery of the source wavefield.

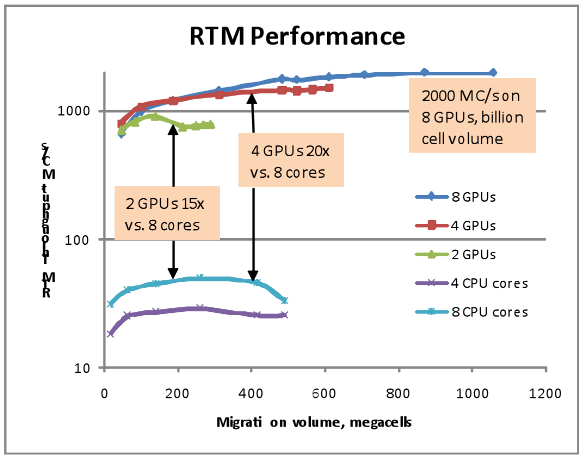

Figure 5 - RTM throughput vs. migration volume size

There is computational overhead in moving data to and from the GPU device. When the migration problem is partitioned among multiple GPU devices there is additional overhead in communicating the partition boundary data between devices, which takes place after every wavefield propagation timestep.

Applying the imaging condition between the back-propagating receiver and reverse-time source wavefields can be a major I/O challenge of RTM. Our solution to this problem maintains excellent computational throughput even as the number of timesteps, and therefore the wavefield storage requirements, increase.

Power Consumption and Footprint

The power consumption of a large data center is a major consideration in the operating costs and environmental impact of the operation. Because the RTM runtime scales with the 4th power of migration frequency, the high performance per watt of GPU technology becomes very important with a realistic-sized migration project.

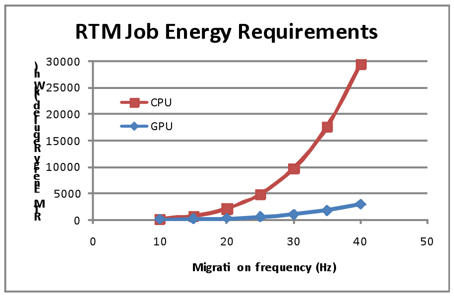

Figure 6 - Energy required for 8x8x8km RTM job

The assumptions made in Figure 6 were that an 8-core CPU node draws 400W and has a throughput of 50MC/s, while an 8-core, 4-GPU node draws 1200W and has a throughput of 1500MC/s. Figure 6 represents total energy for RTM imaging of an 8x8x8km volume with 1000 shots and 10s trace length.

Another benefit is that the total number of compute nodes required to solve a certain RTM problem in a given timeframe is significantly reduced, which has advantages in cost, space and complexity of datacenter management.

Conclusions

NVIDIA GPU hardware has been shown to be an excellent high-performance platform for RTM, with significant advantages over purely CPU hardware in both runtime and power consumption.

The use of GPU hardware, efficient access to the source wavefield in reverse time, and efficient domain decomposition combine to enable our Reverse Time Migration implementation to be a practical and cost-effective solution to industrial-sized RTM problems of over one billion cells.

Further developments include improving multi-GPU scaling efficiency, Vertically Transverse Isotropic (VTI) RTM, efficient implementation of Perfectly Matched Layer (PML) boundary conditions for both isotropic and anisotropic RTM, Pseudo-Spectral RTM and Tilted Transverse Isotropic (TTI) RTM.

Acknowledgements

The authors are thankful to BP, especially Frederic Billette and Joe Dellinger, for providing access to their benchmark dataset.