Reverse Time Migration: A Case Study of GPU / CPU Hybrid Computing

Darren S. Foltinek, Scott D. Quiring, Michael Durocher, Yoichi Yoshida, Acceleware Ltd.

Reverse Time Migration (RTM) is now the dominant imaging technique used in deep marine seismic exploration. This commercial case study highlights the need for the implementation to be continuously re-balanced as hardware performance and software features evolve over time in order to avoid performance bottlenecks and maintain optimum throughput. The basic methodology of RTM and the computational challenges it raises will be shown, and a solution that takes full advantage of the resources available on a GPU / CPU hybrid compute node will be presented.

There are two major computational challenges in RTM: efficient propagation of a 3D seismic wavefield and accessing the wavefield in reverse-time order.

The key imaging benefit of RTM is the use of a two-way solution to the acoustic wave equation, which is commonly solved using the finite-difference, time-domain (FDTD) method. The FDTD approach requires the computation of 3D spatial derivatives at each of the millions of points in a 3D volume for thousands of timesteps. The suitability of GPU architecture to solve this problem is explained.

The RTM imaging condition requires access to both source and receiver wavefields in reverse-time order, which requires large local storage requirements, extra computation, or a combination of both. The extra arithmetic throughput and memory bandwidth of the GPU enables an efficient solution by trading off computation for storage.

The basic imaging condition is relatively inexpensive, however, more advanced imaging conditions are more computationally and memory intensive. These expensive imaging conditions can be performed concurrently, with minimal slowdown of the GPU propagation, on the relatively underutilized CPU cores. This has the added benefit of freeing up scarce GPU memory.

As GPU performance continues to improve, the wavefield propagation speed increases and bandwidth demands on local storage increase proportionally. By applying data compression techniques, the I/O bandwidth can be reduced, and the data compression can be performed on either GPUs or CPU cores.

Rebalancing an algorithm can involve a codebase refactor, which requires a flexible software architecture and very good test coverage. Additionally, a layered software design allows new hardware technologies to be supported as they become available without changing the software at the higher levels. This architecture will be shared.

The fundamental lesson from this case study is that to achieve optimal throughput, the algorithm needs to be continuously re-balanced as hardware performance and software features evolve over time.