Introduction

Full-waveform inversion (FWI) is one important tool in seismic exploration (the search for hydrocarbons):

- Vibrational energy is sent into the earth.
- Observations of waves reflected and refracted back are made at the surface and in boreholes.
- The wave equation, parameterized with the subsurface quantity of interest (e.g., slowness squared, $1/v^2$), is solved.
- The result is compared with the data, and based on this the parameter model is updated.

The process is repeated, refining the subsurface parameter model with each iteration. FWI is important because it puts geophysical inversion on a rigorous footing, eliminating the manual tuning of traditional algorithms (such as migration). Simulation of seismic wave propagation is the biggest computational burden of FWI.

The Wave Equation: $Hu = q$

- $H$: Wave physics operator; $N \times N$ Helmholtz matrix; large, sparse and has diagonal band structure.
- $m$: $N$-dimensional; defines the model parameter throughout the $x \times y \times z = N$ physical domain.
- $u$: Complex Fourier transform of the pressure with respect to time.
- $q$: Amplitude of the source at angular frequency $\omega$.
- Constant density isotropic acoustic wave equation formulated in the time harmonic (frequency) domain.
- Contains perfectly matched layer functions that eliminate numerical reflection artifacts from the boundaries.

Iterative Linear Solvers: Kaczmarz, CARP and CARP-CG

The Kaczmarz algorithm [1] solves a general linear system $Ax = b$ by projecting the current iterate $x_0$ onto the hyperplane orthogonal to a row $a_j$ of the matrix, and setting the next iterate equal to the result ($w$ is a relaxation parameter we set to 1.5, as in [2]):

$$x_{i+1} = x_i + w(b - a_j^T x_i) / ||a_j||^2$$

- CARP (Component Averaged Row Projections) [3] is a parallelization of the Kaczmarz algorithm that works on sets of rows simultaneously, averaging elements that sets have in common.
- Using CARP to cyclically project onto each row form first to last and back (a double sweep) gives an equivalent symmetric positive semi-definite system (which $H$ is not), which can be solved with the method of conjugate gradients (CARP-CG) [4].

Implementation Details

- Diagonal storage format used for $H$, with rows contiguous in memory.
- Single-threaded Kaczmarz (CARP) sweeps.
- Programmed in C, reading matrix and vector $(H, u_0$ and $q$) inputs from disk.
- Complex numbers stored as arrays of structures.
- Compiled with $-O2$ and $-xavx$ switches to take advantage of vector CPU instructions.
- Simulations bound to one hardware thread on one core of an Intel Xeon E5-2670.
- Medium $m$ was taken to be the SEG/EAGE overthrust velocity model, presented by [6], undersampled by various factors.
- Timing results are the average of 10 runs.

Hypothesis

Since the double CARP sweep is the most time-consuming part of CARP-CG it was the object of optimization. We noticed that the forward Kaczmarz sweeps were consistently faster than the backward sweeps; our hypothesis was that this was due to different patterns of memory access:

- Forward sweeps access matrix $H$ elements in the order in which they are stored: both row and diagonal indices increasing.
- Backward sweeps jump from the end of one row to the start of the previous row: row index decreasing, diagonal index increasing.

Results

The four memory access patterns tested and their execution times.

- Note that the curves for implementations 1 and 4 overlap.
- For the last set of points the speed-up between implementations 4 and 2 is 7%.
- For comparison, the speed-up we measured between sweeps programmed in MATLAB (not shown) and any of the C-sweeps is approximately 80-fold.
- For the $94 \times 401 \times 401 \approx 15 \times 10^6$ model, efficiency was measured with processor pipeline slots, which are the basic unit of instruction level parallelism. They can be in one of four states [7]:

Conclusions

- The execution times presented above are a demonstration of the well known rule that memory is best addressed in order.
- Improvements in efficiency metrics such as the percentage of time the pipeline is waiting for memory access (back-end bound pipeline slots), do not translate directly into improvements in execution time.

Acknowledgements

This work was partly financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Collaborative Research and Development Grant DND/ESS-II (ST560-00). This research was carried out as part of the UBCs ONR D program with support from the following organizations: EG Group, BDP, BCG, Chetco, Cancomp/Phillips, Ion CXT, Perentie, FOS, Towd SA, WesternGeco, and Woodside.

References