Software acceleration of CARP, an iterative linear solver and preconditioner

Art Petrenko, Tristan van Leeuwen and Felix J Herrmann

Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia
2207 Main Mall, Vancouver BC, V6T 1Z4, Canada

E-mail: apetrenko@eos.ubc.ca

Abstract. We present the results of software optimization of a row-wise preconditioner (Component Averaged Row Projections) for the method of conjugate gradients, which is used to solve the diagonally banded Helmholtz system representing frequency domain, isotropic acoustic seismic wave simulation. We demonstrate that in our application, a preconditioner bound to one processor core and accessing memory contiguously reduces execution time by 7% for matrices having on the order of 10^8 non-zeros. For reference we note that our C implementation is over 80 times faster than the corresponding code written for a high-level numerical analysis language.

Simulation of seismic wave propagation is the biggest computational burden of full-waveform inversion (FWI), an important method of calculating earth properties based on measured seismographic data. In the frequency domain, and specializing to the constant density isotropic acoustic (pressure waves) case for concreteness, this simulation corresponds to solving the wave equation, a large linear system which can be succinctly represented in matrix notation as $H(\omega, \mathbf{m})\mathbf{u} = \mathbf{q}$, where H is the $N \times N$, Helmholtz operator which depends on angular frequency ω of the wavefield being simulated, the vector of medium properties $\mathbf{m}$, and includes a perfectly matched layer at the boundary of the domain to eliminate reflection artifacts. $\mathbf{u}$ is the (complex) Fourier transform of pressure with respect to time and $\mathbf{q}$ is the amplitude of the source at ω. The matrix H is sparse and has diagonal band structure, and the vectors $\mathbf{m}$, $\mathbf{q}$ and $\mathbf{u}$ have one element per physical domain grid point. We solve the linear system by an iterative method: the method of conjugate gradients with the CARP preconditioner known as CARP-CG, developed by [1, 2] and in turn based on the Kaczmarz algorithm for solving linear systems [3], originally published in 1937.

The Kaczmarz algorithm solves a general linear system $A\mathbf{x} = \mathbf{b}$ by projecting the current iterate $\mathbf{x}_i$ onto the hyperplane orthogonal to a row of the matrix, and setting the next iterate equal to the result:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + w(b[k] - \langle \mathbf{a}_k, \mathbf{x}_i \rangle) \frac{\mathbf{a}_k^*}{\|\mathbf{a}_k\|^2}. \tag{1}$$

Here w is a relaxation parameter, $\mathbf{a}_k$ is the k^{th} row of A, $b[k]$ is the corresponding element of the right-hand side, and * denotes Hermitian transpose. A Kaczmarz "sweep" consists of i and k jointly running from the first row to the last (forward), from the last to the first (backward), or from first to last and back again (double). CARP builds on the Kaczmarz algorithm by performing double sweeps on sets of rows in parallel, averaging those elements of $\mathbf{x}_i$ that are updated by sweeps on more than one set. If CARP (or Kaczmarz) is run repeatedly, $\mathbf{x}_i$ will

converge to the solution of $A\mathbf{x} = \mathbf{b}$, but instead we use the algorithm as a preconditioner for CG.

We represent the double CARP sweep operator in matrix notation and apply it to the Helmholtz system noting that since $\mathbf{u}$ is the solution, it is an eigenvector of the double sweep. This leads to an equivalent formulation of the problem:

$$\begin{aligned} \mathrm{DCSWP}(H, \mathbf{u}, \mathbf{q}, w) = \mathbf{u} &= Q\mathbf{u} + R\mathbf{q} \\ (I - Q)\mathbf{u} &= R\mathbf{q}. \end{aligned} \tag{2}$$

A solution $\mathbf{u}$ to (2) will also solve the original wave equation. Furthermore, because the matrix Q is a product of symmetric matrices Q_i each corresponding to the effect of a single CARP iteration performed in the symmetric order of a double sweep, $I - Q$ itself is symmetric and positive semi-definite (unlike H). Thus the equivalent system is amenable to solution with CG, as shown by [2] and further demonstrated by [4]. The double CARP sweep is used in the CG method to multiply vectors with the matrix $I - Q$ without having to form it explicitly. This matrix-vector operation is the most time-consuming part of that algorithm, and we devoted efforts to optimize its implementation. Related work has been done by [5], who have examined the improvements that Kaczmarz and other algorithms experience on Graphical Processing Units (GPUs).

We constructed the wave equation operator H using a finite-difference stencil designed to minimize numerical dispersion, as in [6], and normalized its rows. Since the main operation in CARP is taking dot products with rows of H, the matrix diagonals were stored with the rows contiguous in memory, allowing for efficient access. Code for the Kaczmarz sweeps was written in C, reading the matrix H (in the format described above), and the vectors $\mathbf{u}_0$ and $\mathbf{q}$ from files stored on disk. To ensure a predictable profiling environment, the simulations were bound to one hardware thread on one core of an Intel Xeon E5-2670 machine. The medium $\mathbf{m}$ was taken to be the SEG/EAGE overthrust velocity model, presented by [7], undersampled by various factors. The initial guess and the source $\mathbf{q}$ were i.i.d. Gaussian, with the source multiplied element-wise by the same vector used to normalize the rows of H.

Our main result is removal of an inefficiency we observed that had to do with different patterns of memory access: while the forward sweeps accessed matrix elements in the order in which they were stored, backward sweeps jumped from the end of one row to the start of the previous row. To test the hypothesis, we ran the four possible implementations: 1) forward sweep accessing memory contiguously, 2) backward sweep jumping from end of row to start of previous row, 3) forward sweep jumping from start of row to end of next row and 4) backward sweep accessing memory contiguously. The execution time (wall time) is shown in Figure 1.

As expected, those combinations that correspond to accessing memory sequentially ran faster that those that had to make jumps. However while there is a clear correlation between execution time and the fraction of the pipeline containing retired (completed) operations per cycle, the correlation with back-end bound operations (the ones that would be waiting on memory access) is more muddled. As can be seen from Figure 2, the front-end bound fraction (which is high when there are not enough instructions to process) also changes across implementations. Thus we cannot yet definitively conclude that our hypothesis of contiguous memory access is the correct explanation for the improvement in timing seen from implementation 3 to 1 and from 2 to 4.

We are currently undertaking further work with the CARP preconditioner. This includes measuring the effect of replacing direct `for` loops in the body of the sweeps with calls to CBLAS, an optimized linear algebra library, and expanding the scope of the implementation to include the whole CG method (parallelized via multi-threading). We are also porting the algorithm onto FPGAs, and look forward to being able to report on these developments in the future.

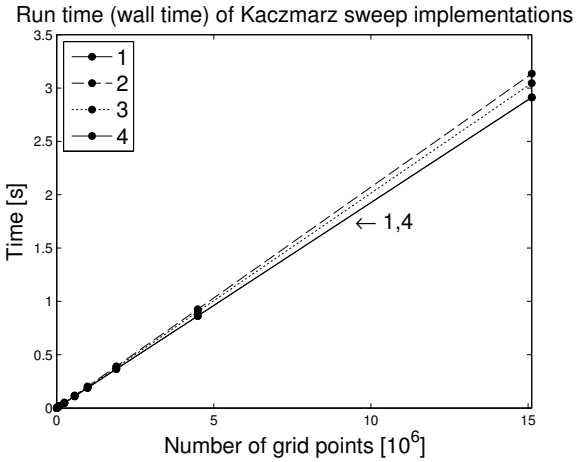

Figure 1. Average time over 10 sweeps; note that the curves for implementations 1 and 4 overlap. For the last set of points, which correspond to a $94 \times 401 \times 401$ physical domain, the speed-up between implementations 4 and 2 is 7%. For comparison, the speed-up we measured between sweeps programmed in MATLAB (not shown) and any of the C-sweeps is approximately 80-fold.

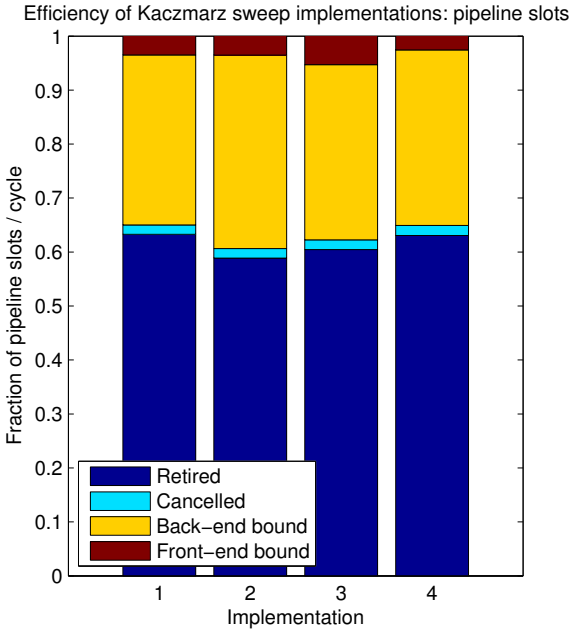

Figure 2. Average performance over 10 runs using a $94 \times 401 \times 401 \approx 15 \cdot 10^6$grid points subsampling of the SEG/EAGE overthrust model. A pipeline "slot" is one pipeline unit in the processor.

Acknowledgments

We would like to thank Henryk Modzelewski for his support of the infrastructure necessary to carry out this research. This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BGP, BP, Chevron, ConocoPhillips, Petrobras, PGS, Total SA, and WesternGeco.

References

[1] Gordon D and Gordon R 2005 *SIAM Journal on Scientific Computing* **27** 1092–1117 (*Preprint* `http://epubs.siam.org/doi/pdf/10.1137/040609458`) URL `http://epubs.siam.org/doi/abs/10.1137/040609458`

[2] Gordon D and Gordon R 2010 *Parallel Computing* **36** 495–515 ISSN 0167-8191 URL `http://www.sciencedirect.com/science/article/pii/S0167819110000827`

[3] Kaczmarz S 1993 *International Journal of Control* **57** 1269–1271

[4] van Leeuwen T, Gordon D, Gordon R and Herrmann F J 2012 Preconditioning the helmholtz equation via row-projections *EAGE technical program* (EAGE) URL `https://www.slim.eos.ubc.ca/Publications/Public/Conferences/EAGE/2012/vanleeuwen2012EAGEcarpcg/vanleeuwen2012EAGEcarpcg.pdf`

[5] Elble J M, Sahinidis N V and Vouzis P 2010 *Parallel Computing* **36** 215–231 ISSN 0167-8191 URL `http://dx.doi.org/10.1016/j.parco.2009.12.003`

[6] Operto S, Virieux J, Amestoy P, L'Excellent J Y, Giraud L and Ali H B H 2007 *Geophysics* **72** SM195–SM211 (*Preprint* `http://geophysics.geoscienceworld.org/content/72/5/SM195.full.pdf+html`) URL `http://geophysics.geoscienceworld.org/content/72/5/SM195.abstract`

[7] Aminzadeh F, Jean B and Kunz T 1997 *3-D salt and overthrust models* (Society of Exploration Geophysicists)