OPESCI: Open Peformance portablE Seismic Imaging

Marcos de Aguiar [2] Gerard Gorman [1] David Ham [1] Christian Jacobs [1] Paul Kelly [1] Michael Lange [1] Fabio Luporini [1] Renato Miceli [2]
Tianjiao Sun [1] Felippe Vieira Zacarias [2]

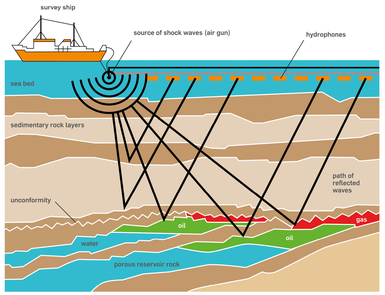
[1]Imperial College, London, United Kingdom

[2]SENAI CIMATEC, Salvador, Brazil

Motivation

Seismic imaging, used in energy exploration, is arguably the most compute and data intensive application in the private sector. The bulk of the computational cost relates to simulating the propagation of waves through the subsurface. Most seismic imaging codes today use a simplified physics model based on the acoustic equation. Using the elastic wave equation would provide a more accurate physics model and ultimately lead to more accurate subsurfaces images. However, this would dramatically increases the cost of an already expensive computation.

Computer architectures are rapidly evolving and diversifying to continue delivering performance increases within tight energy envelopes. These changes offer opportunities — but also demand disruptive software changes to harness the full hardware potential. The question is how to achieve performance portability across diverse, rapidly evolving architectures, in spite of the sharp trade-off between easy-to-maintain, portable software written in high-level languages such as Python; and highly optimized, parallel codes for modern many-core architectures. The solution proposed is to leverage domain-specific languages (DSL) and code generation. At the highest level of abstraction application developers write algorithms in clear, concise manner akin to how algorithms are written mathematically, while at the lower levels source-to-source compilers explore rich implementation spaces to transform DSL codes into highly optimized native codes that can be compiled for target platforms for near-to-peak performance. The framework provides layers, decoupling domain experts from code tuning specialists, where different optimized code generator back ends can be replaced.



Full Waveform Inversion

- Given a sound source, an array of receivers, and a physics model for how seismic energy propagates, use a wiggle for wiggle comparison infer the structure of the subsurface.
- Today a seismic survey has in the order of a petabyte of data. Expectation that this could be 10's of petabytes in 12-18 months...
- Subsurface image is built up in iterative process by running the physics model forwards and backwards, all the while minimising the difference between the measured data and the simulated data.
- Dominant computational expense is running the wave model (aka forward model, and propagator) for each shot of data.
- Current state of the art is to use a modified acoustic equation.
- A linear elastic model will result is greater accuracy, but this greatly increases the computational cost.

Opportunity & challenge
1. Elastic wave equation

- Provides a significantly better representation of the wave field than the standard acoustic models.
  - Models both p-waves and s-waves.
- Also much more expensive to compute:
  - S-waves travel at about half the speed of p-waves, and therefore half the wavelength:
    - Grid resolution needs to be doubled (factor of 8 increase in memory for 3D).
    - Time step needs to be halved therefore must execute twice the number of time steps.

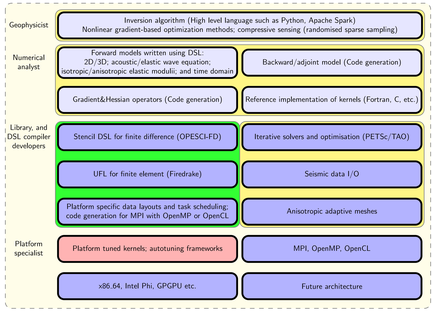
2. Advanced numerical methods

- Regular grids with finite difference is the modus operandi of O&G industry.
  - Higher order methods can achieve greater accuracy at lower cost:
    - Coarser resolution (fewer grid points).
    - Larger time step.
    - Great data locality, opportunities for vectorisation etc.
- Implementation complexity!
  - High order methods are many times more involved.

3. Architecture and code modernisation

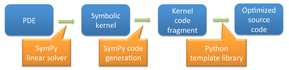
- Hardware continues to track Moore's Law — but working harder on software.
- Must exploit parallelism at every level to achieve good performance, e.g.:
  - Various parallel programming models (MPI, threads, etc.)
  - Deep memory hierarchy, data locality.
  - SIMD vectorisation.
  - Heterogeneous computing Intel® Xeon®, Intel® Xeon Phi™.
- Increasingly difficult for domain specialists to implement HPC software.
- Traditional numerical algorithms may need to be discarded in favor of methods better suited to computer architectures.
- Domain Specific Languages (DSL's) offer a route to bridge the divide between domain specialists (often the application developers) and parallel programming specialists (in this case compiler writers).

Architectural layout

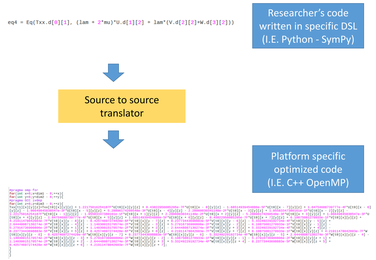


Finite difference using symbolic mathematics and code generation

- SymPy is symbolic maths library in Python.
  - Systematically derive computational kernel from governing equations.
  - Extensible code generation classes that can be used to generate bespoke code.
- Finite difference is highly structured.
  - Make use of template python libraries (Mako, Jinja) to insert kernel into pre-prepared code templates.
  - Parameters (e.g. domain size, grid spacing) can be passed to templates as additional keywords.
  - One template file can be used for variety of problems.



- Only changing parameters at symbolic level without directly modifying source code.
  - Parameter/scheme autotuning.
- Separate mathematics from implementation.
  - Generate multiple implementations in different stencil languages from same math specification.
  - Only need to extend code generation class and template file to support a new stencil language.



Initial tests using elastic wave equation

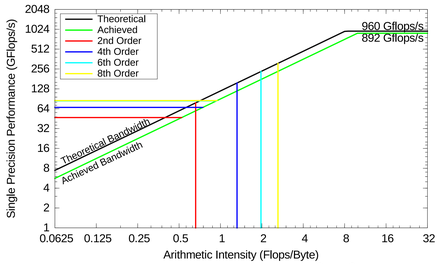


Figure: Roofline Model for Wave Equation on Intel® Xeon®.
Under investigation why the 8th order performance didn't increase proportionally.

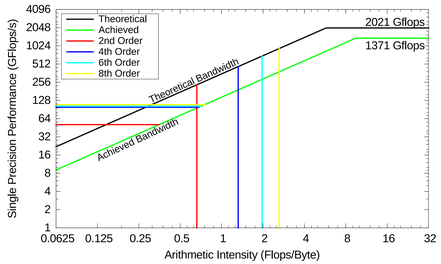


Figure: Roofline Model for Wave Equation on Intel® Xeon Phi™.
Under investigation why the 8th order performance didn't increase proportionally.

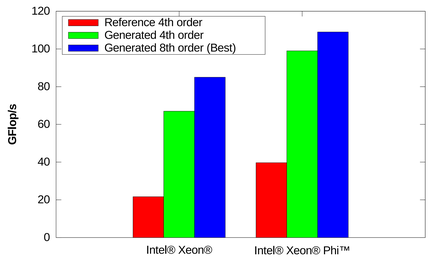


Figure: Elastic Wave Stencil: Manual vs. Generated

Results

- Operation intensity of the reference code (OpenMP) is relatively low.
- Generated code (OpenMP) was able to increase the intensity, fixed alignment and enabled SIMD vectorization, thus was able to achieve much better performance.
- There is still some room for performance gain in the generated code, according to the roofline estimation and initial profiling, but the code was able to come close to the theoretical peak.
- Generated code is outperforming the reference implementation in the Intel® Xeon®and Intel® Xeon Phi™platforms.

Future work

- Profile code in both Intel® Xeon®and Intel® Xeon Phi™.
- Implement different code generators for each platform if necessary.
- Optimize to try to achieve the roofline estimated value for this algorithm.
- Test polyhedral approaches for better cache usage.
- Test other equations.
- Optimising the backend compiler for Intel® Xeon Phi™.
- Build a FWI framework on top of this high level abstraction.

Links

- OPESCI – http://opesci.github.io (Intel PCC project)
- AMCG – http://amcg.ese.ic.ac.uk

References

[1] Williams, S., Waterman, A., Patterson, D., "Roofline: an insightful visual performance model for multicore architectures." Communications of the ACM. **52 (4)**, 2009.

[2] Borges, L. "3D Finite Differences on Multi-core Processors.", available on-line at http://software.intel.com/en-us/articles/3d-finite-differences-on-multi-core-processors , 2011.

[3] Reinders, J., Jeffers, J. "High Performance Parallelism Pearls" (Morgan Kaufmann) (2014).

[4] Joyner D., Čertík, O., Muerer, A., Granger B. E. "Open Source Computer Algebra Systems: Sympy." ACM Communications in Computer Algebra Volume **45** (2011).

Acknowledgements

Open Performance portablE SeismiC Imaging (OPESCI) is funded under the Intel Parallel Computing Center program — a unique collaboration between industrial partners and researchers at Imperial College London and SENAI CIMATEC, focused on developing disruptive open-source software on HPC architectures. The authors would also like to acknowledgement support from EPSRC EP/L000407/1.

EPSRC Pioneering research and skills

FINEP

intel

BG GROUP