AUTOMATIC CODE GENERATION FOR GPUs USING DEVITO

Fabio Luporini, Gerard Gorman

RICE O&G HPC 2020
Talk outline

• Motivation - why do we care?
• Who or what is Devito?
• GPU support - without the excruciating pain
• Roadmap
• Closing remarks
• Acknowledgements
Motivation

• Seismic imaging:
  • FWI, RTM, LS-RTM, TTI, elastic, visco-elastic propagators, etc.
  • Some of the most computational expensive and algorithmically complex workloads found in industry.

• Reducing the cost of modernizing software for exascale and Cloud.

• Skills/knowledge gap between geophysicists, data scientists and HPC developers.

• **Do researchers/developers have the tools that they need to develop next-generation AI/ML technologies?**
Who or what is Devito?
Traditional approach

\[ m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0 \]

void kernel(...) {
    ...
    <impenetrable code with aggressive performance optimizations written by rockstars, gurus, ninjas, unicorns and celestial beings>
    ...
}
Raising the level of abstraction

\[ m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0 \]

void kernel(...) {
    ...
    <impenetrable code with aggressive performance optimizations written by rockstars, gurus, ninjas, unicorns and celestial beings>
    ...
}
Raising the level of abstraction

\[ m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0 \]

void kernel(…) {
    ...
    ...
    <impenetrable code with aggressive performance optimizations written by rockstars, gurus, ninjas, unicorns and celestial beings>
    ...
}
Raising the level of abstraction

\[ m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0 \]
Raising the level of abstraction

\[ m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0 \]

eqn = m * u.dt2 + eta * u.dt - u.laplace
Raising the level of abstraction

\[
m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0
\]

eqn = m * u.dt2 + eta * u.dt - u.laplace

void kernel(...) { ... }
Raising the level of abstraction

\[ m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0 \]

Devito

eqn = m * u.dt2 + eta * u.dt - u.laplace

void kernel(...) { ... }
Devito: a DSL and compiler for explicit finite differences

• **Python** package — easy to learn (and no, this does not mean it runs slow)

• **Devito is a compiler** that generates optimized parallel code:
  • C, SIMD, OpenMP, OpenMP 5 offloading, MPI (soon OpenACC)
  • x86 (including Xeon Phi series), **GPUs**, ARM64, Power8/9

• **Composability: integrate with existing codes and AI/ML**
  • Integrate with existing codes in other languages
  • Works out-of-the-box with other popular packages from the Python ecosystem (e.g. PyTorch, NumPy, Dask, TensorFlow)

• **Open source platform** – MIT license.

• **Best practises software engineering:** extensive software testing, code verification, CI/CD, documentation, tutorials and PR code review.

• **Cloud ready** - Wednesdays hands-on workshop+hackathon running on Azure.
Growing open source and commercial community

• Started in 2016 … just released **Devito v4.1**:
  
  • Core compiler is 17k lines of code, 8k lines of comments for developers
  
  • 9k lines of unit and regression tests used in CI/CD (ie automated testing)
  
  • ~40 Jupyter tutorials and examples - included in CI/CD
  
  • 32 contributors to the code base, 7 people in the core team.

• Users:
  
  • Several companies financially support the open source Devito consortium. Announced: BP, DUG, Microsoft, Shell (more are welcome!)
  
  • Worked with DUG to bring Devito from research to production grade.
  
  • 272 people on our open Slack workspace from 90+ different companies and research institutions.
GPU support - without the excruciating pain
grid = Grid(shape=...)  
u = TimeFunction(name='u', grid=grid)  
m = Function(name='m', grid=grid)  
src = SparseFunction(name='src', grid=grid, npoints=...)  
eqn0 = m * u.dt2 - u.laplace  
eqn1 = ...  
injection = src.inject(field=u.forward, expr=src*s*s**2/m)  

op = Operator(eqn0, eqn1, injection, ...)  

Analysis  

Equations lowering  
Lower symbolic derivatives to stencil expressions  
Constant folding  
Index shifting (to account for halo and padding)  
Lower SubDimensions and SubDomains
Clustering

Group equations into “Clusters”, based on data dependencies
Derive iteration and data spaces
Detect computational properties (e.g., parallelism)

Clusters Optimization

Symbolic (flop-reducing) transformations:
Common sub-expressions elimination
Aliases detection and precomputation
Factorization
Code motion
...

Optimizations for data locality and parallelism:
Fusion
Fission
Blocking

Tree-fication

Turn an ordered list of Clusters into an Abstract Syntax Tree (AST)
AST specialization

Optimized distributed-memory parallelism via MPI
Optimized shared-memory parallelism via OpenMP
SIMD vectorization via OpenMP
Misc optimizations (e.g., denormals)

AST finalization

Loop nest optimization, such as
Symbol declarations and definitions
Data movement (host-device)
Instrumentation for profiling
Header files, globals, macros, ...

JIT-compilation

Synthesis (AST -> file on disk)
Invocation of backend compiler to create a library (".so")
Generated code

https://github.com/devitocodes/devito/blob/master/examples/gpu/01_diffusion_with_openmp_offloading_ipynb

```c
#pragma omp target enter data map(to: u[0:u_vec->size[0]][0:u_vec->size[1]][0:u_vec->size[2]])
for (int time = time_m; time <= time_M; time += 1)
{
    #pragma omp target teams distribute parallel for collapse(2)
    for (int x = x_m; x <= x_M; x += 1)
    {
        for (int y = y_m; y <= y_M; y += 1)
        {
            <stencil update for the 2D diffusion equation>
        }
    }
}
#pragma omp target update from(u[0:u_vec->size[0]][0:u_vec->size[1]][0:u_vec->size[2]])
#pragma omp target exit data map(release: u[0:u_vec->size[0]][0:u_vec->size[1]][0:u_vec->size[2]])
```

Through sophisticated data dependence analysis, the Devito compiler knows:

- where to insert the OpenMP pragmas for host-device data movement
- what the parallel and reduction loops are, so it knows where to insert the OpenMP pragmas for parallelism and synchronization
Current performance on GPUs

- Devito v4.1 - [https://github.com/devitocodes/devito/releases/tag/v4.1](https://github.com/devitocodes/devito/releases/tag/v4.1)
- GPU offloading via OpenMP 5
- NVidia V100
- Propagator performance (includes BCs/sources/receivers/…)
- **No performance optimizations yet (join us on Wednesday!)**

<table>
<thead>
<tr>
<th></th>
<th>OI (Flops/Bytes)</th>
<th>GFlops/s</th>
<th>attainable peak</th>
<th>FD-GPoints/s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>iso-acoustic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(12th order, 512³ points)</td>
<td>3.74</td>
<td>600</td>
<td>18%</td>
<td>8.80</td>
</tr>
<tr>
<td><strong>TTI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(12th order, 350³ points)</td>
<td>3.64 (thanks to symbolic optimizations)</td>
<td>387</td>
<td>11%</td>
<td>1.15</td>
</tr>
</tbody>
</table>
MPI support — so far, only for CPUs

```bash
mpirun <mpi args> python app.py
```

No changes to user code required!
GPU support roadmap

- JIT-backdoor to engage HPC/GPU developers directly in Devito development
- MPI support for domain decomposition across multiple devices
  - UCX proposed as an alternative
- Strategies for checkpointing (optimal strategies, lossy compression)
- OpenACC backend (started last week; PR at Wednesdays hackathon?)
- Performance optimization (shared memory?)
- Other backends (OneAPI, CUDA, …)?
Conclusions

• Devito is an open-source high-productivity and high-performance Python framework for finite-differences.

• Driven by commercial & research seismic imaging demands:
  • Industrial advisory board == Devito consortium.

• Based on actual compiler technology (not a source-to-source translator!)

• Interdisciplinary, interinstitutional, international open source effort.

• Growing open source community and commercial users

• Gentle request: Many(!) silent/semi-anonymous industry users - open source is still a novel idea in this industry despite clear evidence from tech industry that it is a critical business strategy. Please engage.

Website: http://www.devitoproject.org
GitHub: https://github.com/opesci/devito
Slack: https://opesci-slackin.now.sh
Acknowledgements

• Thanks for our sponsors who are supporting and collaborating on the continued open source development of Devito for the wider community

• Thanks to our many collaborators and contributors, in particular (those in bold are at OGHPC running the workshop on Wednesday)

  • George Bisbas (see poster session)
  • Edward Caunt (see poster session)
  • Navjot Kukreja (ask about AD and compression)
  • Fabio Luporini (lead Devito developer, and GPU support)
  • Vitor Mickus (see poster session)
  • Rhodri Nelson (ask about PDE’s/solvers)
  • SLIM Group: Felix Herrmann, Mathias Louboutin, Philipp Witte (talk)

For a full list of contributors for each release please see https://github.com/devitocodes/devito/releases