DEVITO: A DSL AND COMPILER FOR AUTOMATED GENERATION OF PRODUCTION-GRADE WAVE PROPAGATORS

Fabio Luporini, Rhodri Nelson, Mathias Louboutin,
George Bisbas, Edward Caunt,
Gerard Gorman

From Data Analysis to High-Performance Computing
Domain-Specific Languages in High-Performance Computing
October 2020

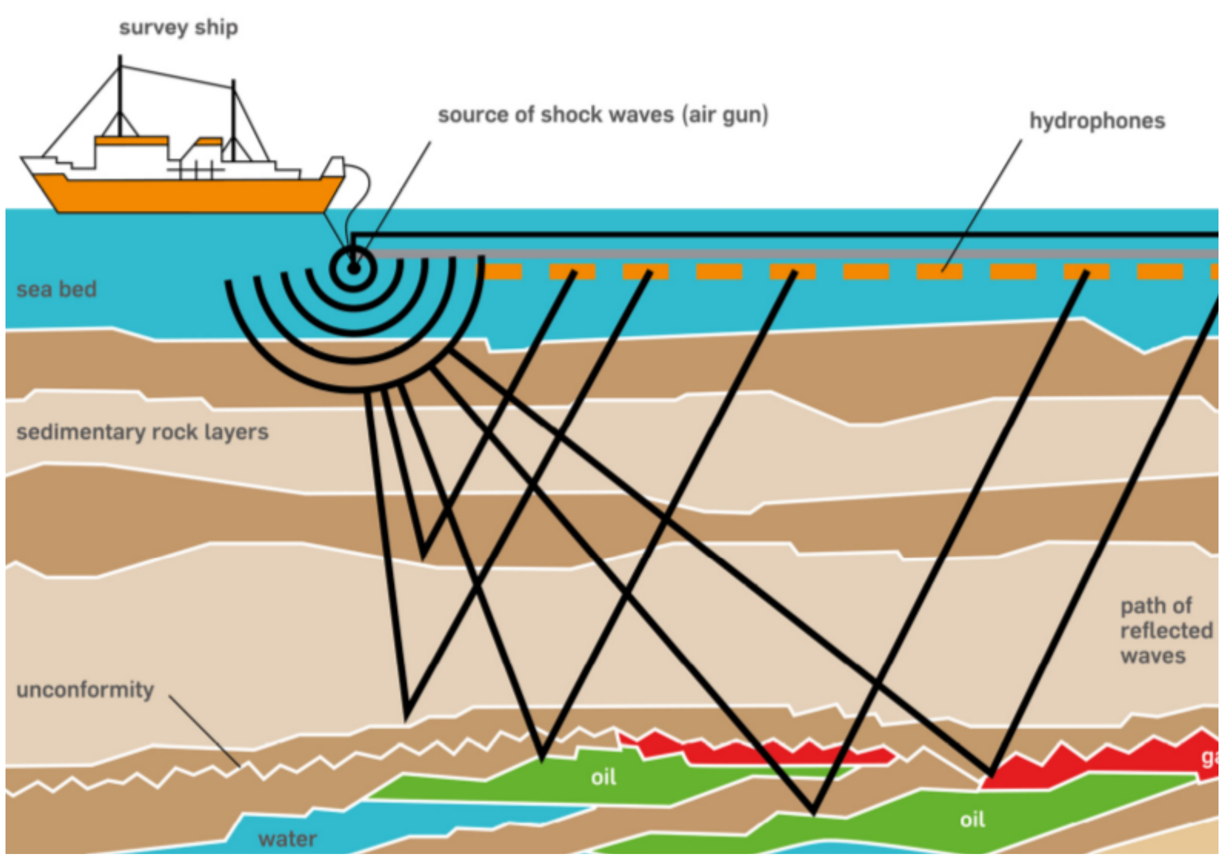
Motivation

- 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.

- Now maturating strong interest in medical imaging and, more generally, ML

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

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



http://www.open.edu/openlearn/science-–maths-–technology/science/environmental-–science/earths-–physical-–resources-–petroleum/content-–section-–3.2.1

The code really needs to fly

Realistic full-waveform inversion (FWI) scenario

- **$O(10^3)$** FLOPs per loop iteration or high memory pressure
- 3D grids with **$>10^9$** grid points
- Often more than **3000** time steps
- Two operators: forward + adjoint, to be executed **~15** times
- Usually **30000** shots

≈ **O(billions) TFLOPs**

Which means days, or weeks, or months on supercomputers

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

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

Devito

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

```
void kernel(…) { … }
```

Why raising the level of abstraction?

- Many formulations of wave equations (R&D still super active)
- Many space and time discretizations
- Many types of boundary conditions in finite differences (too many)
- Unstructured computation (e.g., interpolation for sparse data)
- Proliferation of computer architectures (functional and performance portability)
- …

So, lots of variations in physics, mathematics, platforms, …

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, OpenACC, MPI
  - 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.

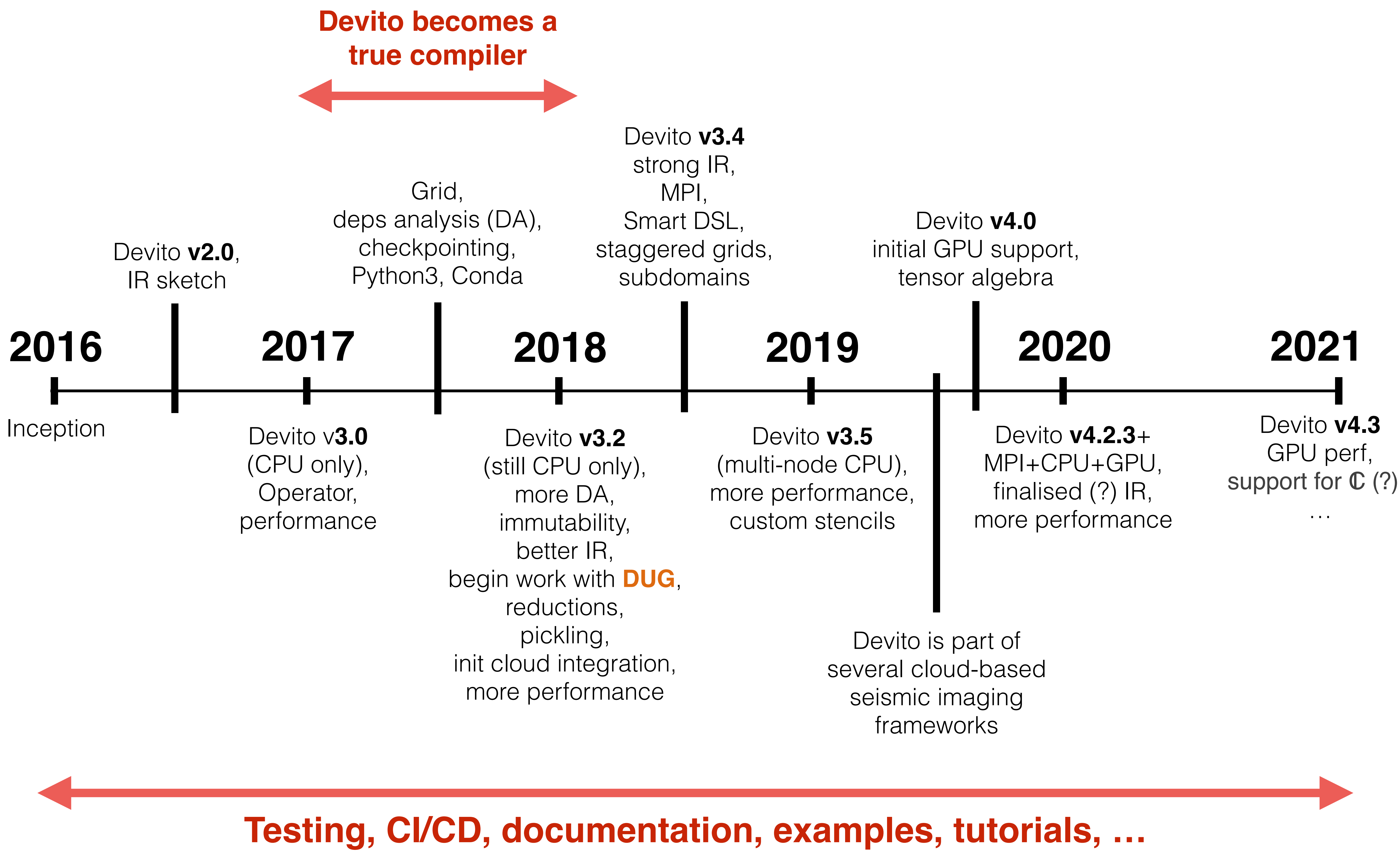
- **Best practises software engineering:** extensive software testing, code verification, CI/CD, documentation, tutorials and PR code review.

- **Cloud ready**

Growing open source and commercial community

- Started in 2016 … just released **Devito v4.2.3**:
  - Core compiler is ~20k lines of code, 8k lines of comments for developers
  - ~12k 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: DUG, BP, Microsoft, Shell.
  - Worked with **DUG** to bring Devito from research to production grade.
  - Open source collaboration with Chevron and SENAI Cimatec.
  - Several academic groups with expertise in physics and geophysics
  - 370+ people on our open Slack workspace from 100+ different companies and research institutions.

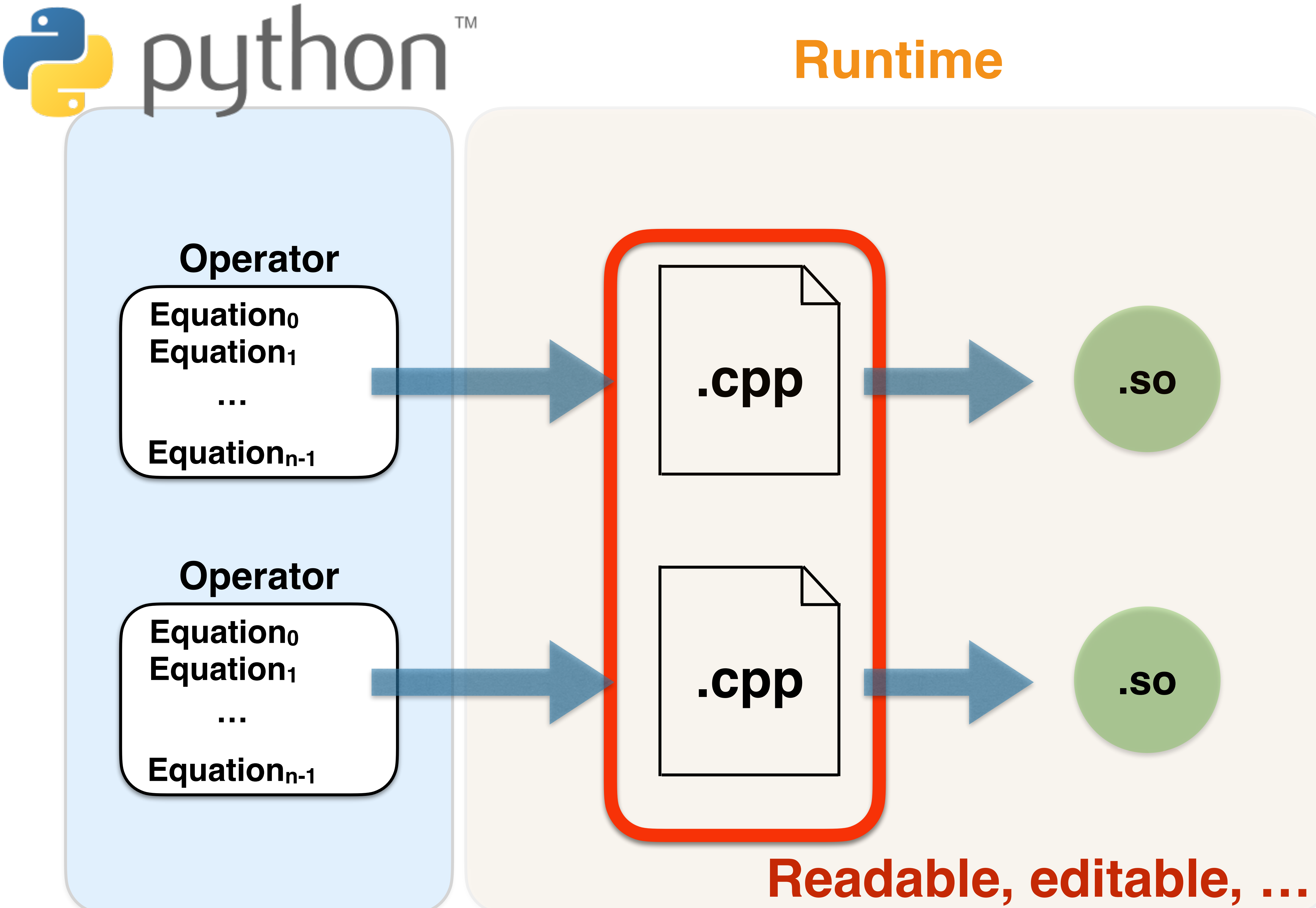
Timeline


Devito becomes a true compiler
Devito **v2.0**, IR sketch
Grid, deps analysis (DA), checkpointing, Python3, Conda
Devito **v3.4** strong IR, MPI, Smart DSL, staggered grids, subdomains
Devito **v4.0** initial GPU support, tensor algebra
2016
2017
2018
2019
2020
2021
Inception
Devito v**3.0** (CPU only), Operator, performance
Devito **v3.2** (still CPU only), more DA, immutability, better IR, begin work with **DUG**, reductions, pickling, init cloud integration, more performance
Devito **v3.5** (multi-node CPU), more performance, custom stencils
Devito is part of several cloud-based seismic imaging frameworks
Devito **v4.2.3**+ MPI+CPU+GPU, finalised (?) IR, more performance
Devito **v4.3** GPU perf, support for ℂ (?), …
Testing, CI/CD, documentation, examples, tutorials, …

So… who or what is Devito?

notebook

High level view of a Devito program



A glance at the compilation pipeline

figure

Some performance optimizations

NOTE: the implementation of a single optimization may actually consist of multiple, small compilation passes

Loop transformations

- Loop blocking
  - classic
  - hierarchical
  - overlapped (see next slides)
- Loop fusion
- Loop fission
- Loop-invariant code motion
- SIMD-ization (through OpenMP pragmas)
- OpenMP
  - classic
  - nested parallelism
  - scheduling heuristics depending on loop body

Expression transformations

- CSE — common sub-expressions elimination
- CIRE — cross-iteration redundancies elimination (see next slides)
- Factorization
- Constant folding
- Optimization of powers

MPI optimizations

- Computation/communication overlap
- Fusion of halo exchanges
- Threaded packing/unpacking
- Asynchronous poking on the progress engine

CIRE — Cross-Iteration Redundancies Elimination

```
a = sin(phi[i,j]) + sin(phi[i-1,j-1]) + sin(phi[i+2,j+2])
```

Observations:

- Same operators (`sin`), same operands (`phi`), same indices (`i, j`)
- Linearly dependent index vectors (`[i, j], [i-1, j-1], [i+2, j+2]`)
- Taking derivatives creates this sort of expressions

```
B[i,j] = sin(phi[i,j])

a = B[i,j] + B[i-1,j-1] + B[i+2,j+2]
```

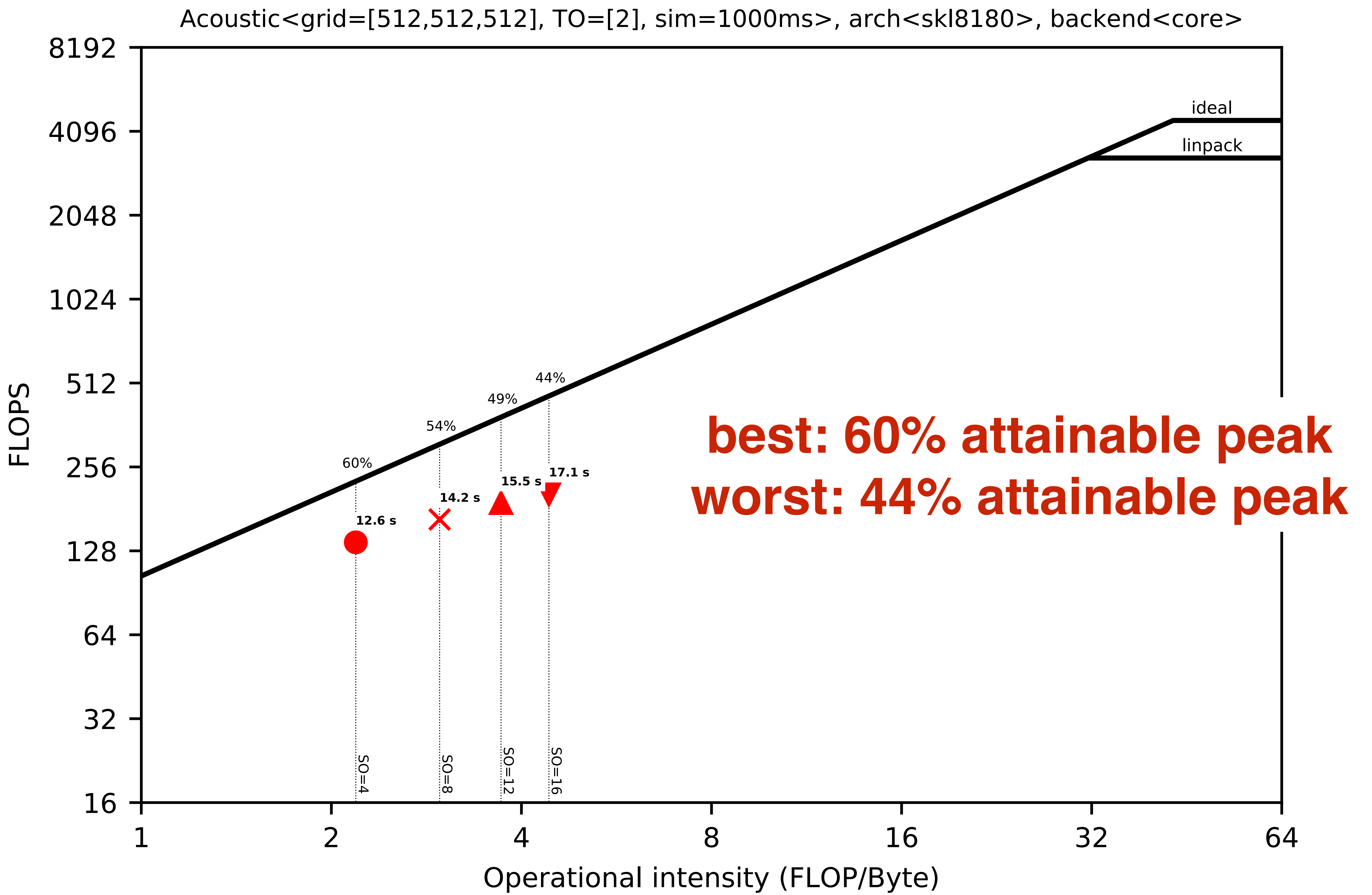

Trade-off FLOPs/storage

Example: CIRE + Overlapped tiling + SIMD + …

```
for (int x0_blk0 = x_m; x0_blk0 <= x_M; x0_blk0 += x0_blk0_size)
{
  for (int y0_blk0 = y_m; y0_blk0 <= y_M; y0_blk0 += y0_blk0_size)
  {
    for (int x = x0_blk0 - 1, xs = 0; x <= x0_blk0 + x0_blk0_size - 1; x += 1, xs += 1)
    {
      for (int y = y0_blk0 - 1, ys = 0; y <= y0_blk0 + y0_blk0_size - 1; y += 1, ys += 1)
      {
        #pragma omp simd aligned(u,v:32)
        for (int z = z_m - 1; z <= z_M; z += 1)
        {
          float r55 = -u[t1][x + 4][y + 4][z + 4];
          float r54 = -v[t1][x + 4][y + 4][z + 4];
          r47[xs][ys][z + 1] = 1.0e-1F*(-(r54 + v[t1][x + 4][y + 4][z + 5])*r18[x + 1][y + 1][z +
+ 5][y + 4][z + 4])*r20[x + 1][y + 1][z + 1]*r21[x + 1][y + 1][z + 1]);
          r52[xs][ys][z + 1] = 1.0e-1F*(-(r55 + u[t1][x + 4][y + 4][z + 5])*r18[x + 1][y + 1][z +
+ 5][y + 4][z + 4])*r20[x + 1][y + 1][z + 1]*r21[x + 1][y + 1][z + 1]);
        }
      }
    }
    for (int x = x0_blk0, xs = 0; x <= x0_blk0 + x0_blk0_size - 1; x += 1, xs += 1)
    {
      for (int y = y0_blk0, ys = 0; y <= y0_blk0 + y0_blk0_size - 1; y += 1, ys += 1)
      {
        #pragma omp simd aligned(damp,epsilon,u,v,vp:32)
        for (int z = z_m; z <= z_M; z += 1)
        {
          float r61 = 1.0/dt;
          float r60 = 1.0/(dt*dt);
          float r59 = 1.0e-1F*(r18[x + 1][y + 1][z]*r47[xs + 1][ys + 1][z] - r18[x + 1][y + 1][z
1][y + 1][z + 1]*r20[x + 1][y + 1][z + 1]*r47[xs + 1][ys + 1][z + 1] + r20[x][y + 1][z + 1]*r21[x]
```

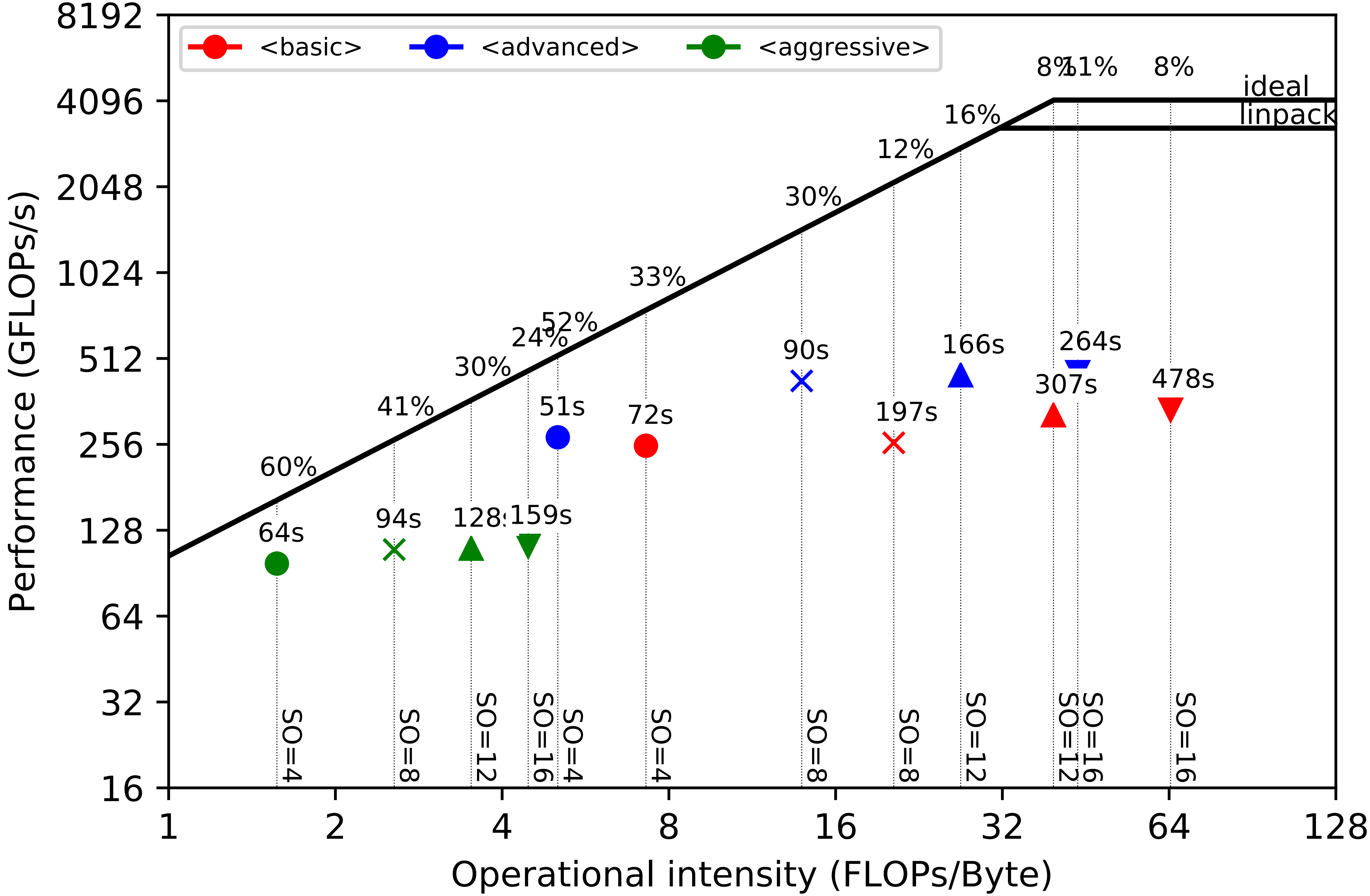
Statistics and performance numbers

Single-socket — Isotropic acoustic on Skylake 8180

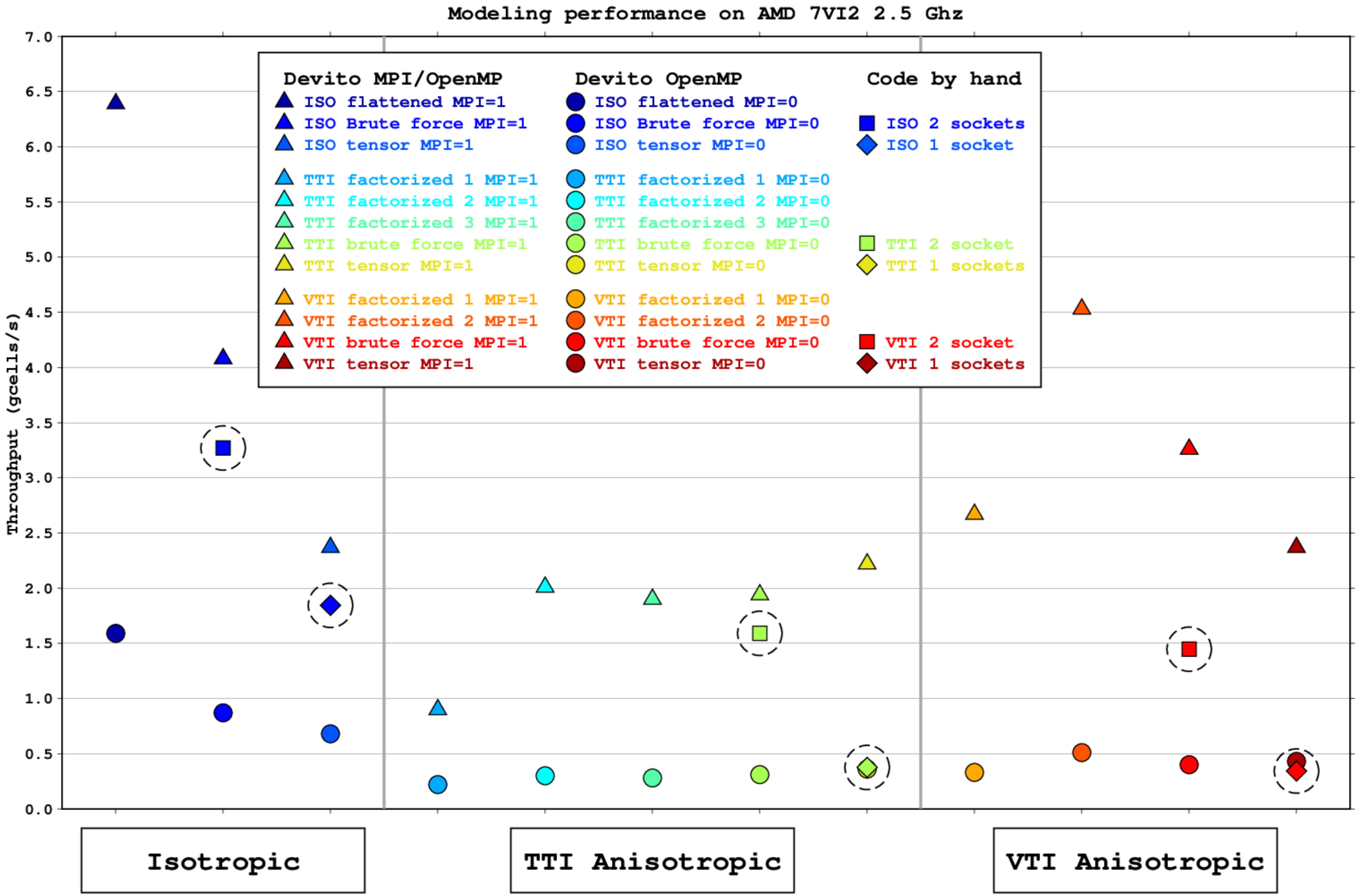


best: 60% attainable peak
worst: 44% attainable peak

Single-socket — TTI on Skylake 8180



Multi-socket — on going open source work with Chevron



Devito in Dugwave, DUG's seismic inversion software

- Dugwave is written in Python
- It uses Devito to implement, among other things, the wave propagators
- Some interesting numbers:
  - over the last 90 days, on average ~1300 KNL nodes were running Dugwave, and therefore Devito, at any given time.
  - this is equivalent to **4PF DP peak**
  - Dugwave's TTI runs on average at 700 Mpts/s (on each KNL, on any given problem instance) after careful tuning and optimization, with peaks of 800 Mpts/s
  - Early TTI MPI results show ~80% parallel efficiency on 4 nodes at a large spatial order (i.e. thick halos), without spending a huge amount of time on tuning yet

MPI support

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

Virtually no changes to user code required!

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

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

Website: http://www.devitoproject.org
GitHub: https://github.com/opesci/devito
Slack: https://join.slack.com/t/devitocodes/shared_invite/zt-gtd2yxj9-Y31YKk_7lr9AwfXeL2iMFg

Appendix

Experimentation details

- Architectures
  - Intel® Xeon® Platinum 8180 Processor ("Skylake", 28 cores)
  - Intel® XeonPhi® 7250 (68 cores)
    - Quadrant mode (still no support for NUMA)
    - Tried 1, 2, 4 threads per core. Shown best.

- Compiler
  - ICC 18 -xHost -O3
  - -xMIC-AVX512 on Xeon Phi
  - -qopt-zmm-usage=high on Skylake

- Runs
  - Single socket
  - Pinning via Numactl
  - On the XeonPhi®, data fits in MCDRAM

- Roofline calculations:
  - Memory bandwidth: STREAM
  - CPU peak: pen & paper
  - Operational intensity: source-level analysis (automated through Devito)