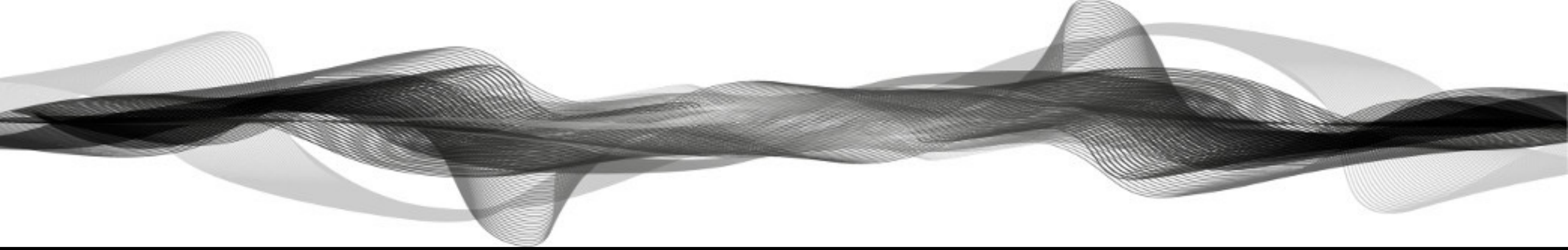


Devito

Imperial College London

Performance-portable high-performance software seismic imaging
Leveraging the power of DSLs, symbolic computation and Python

Dr Gerard Gorman
September 2021

Problem – chapter 1

Hydrocarbon exploration is heavily dependent upon the state-of-the-art 3D seismic imaging techniques to make well informed decisions.

- The cost of drilling a new well at sea is in the region of $100m. Frequently these wells are discovered to be dry.
- Need accurate imaging through the lifecycle of the reservoir to maximize efficiency.
- Imaging will also play a critical role in the long-term monitoring the carbon capture, utilization and storage.

Problem – chapter 2

RTM and FWI (and their variants) are the best methods for creating images of the subsurface data.

- However, they are computationally expensive – a single 3D image can take weeks to compute on a modern supercomputer/Cloud; involves processing TBytes's to PBytes's of data for a single survey. Largest application of HPC in O&G industry.
- The core of these methods all involve solving a wave equation and its adjoint using explicit finite difference methods.
- These methods are under constant innovation to improve accuracy - ie regular software updates.

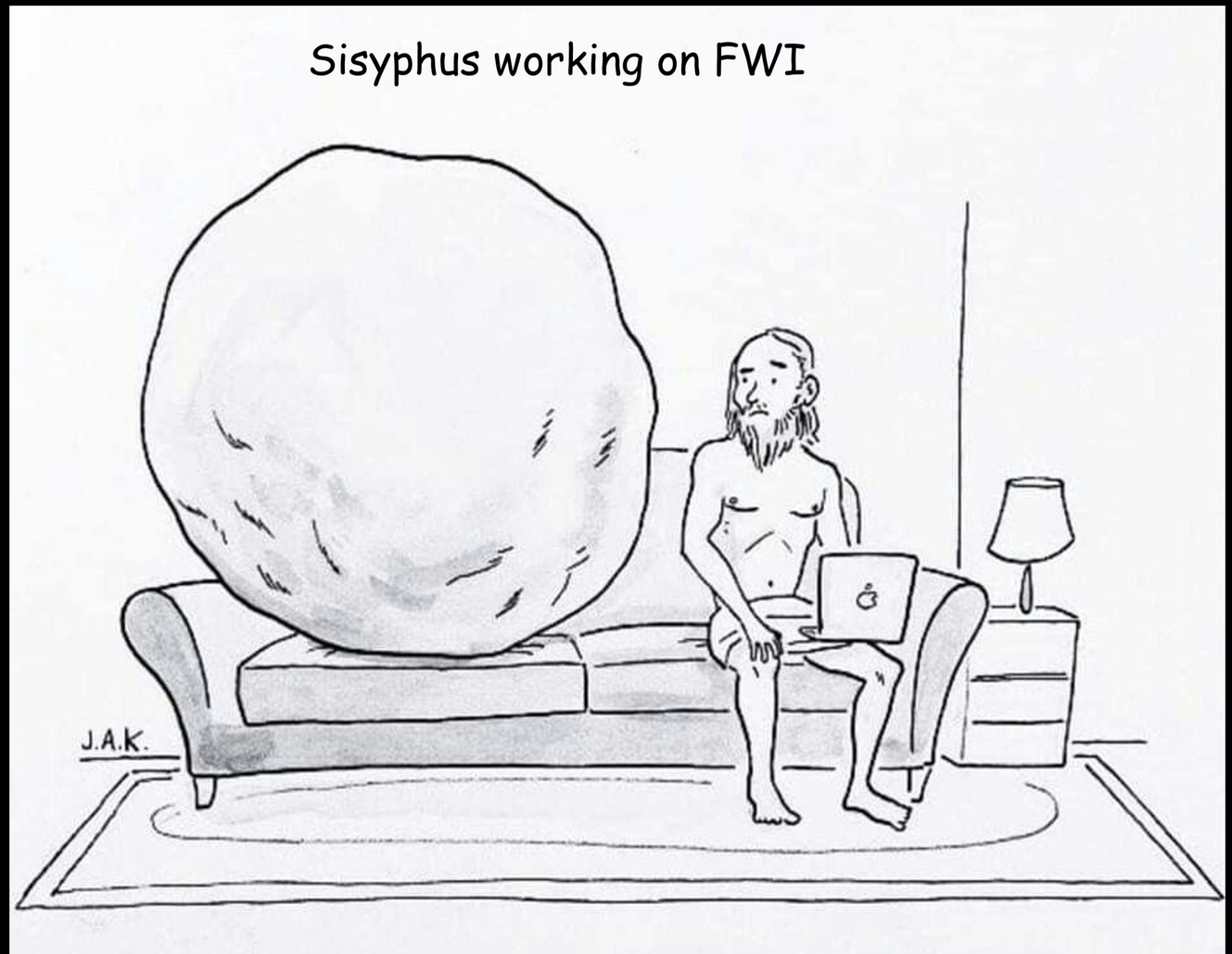
Problem – chapter 3

Developing, testing and optimizing seismic imaging software is like painting the Golden Gate Bridge for HPC developers – they only finish when they have to start all over again.

- Because of the scale of the computation even a 10% increase in performance is valuable.
- High cost of porting and optimizing software from one architecture to another - particularly when the programming model also changes.
- There is a constant stream of prototype codes coming from researchers implementing algorithmic improvements that need to be productionized.
- There are very few coding ninjas.

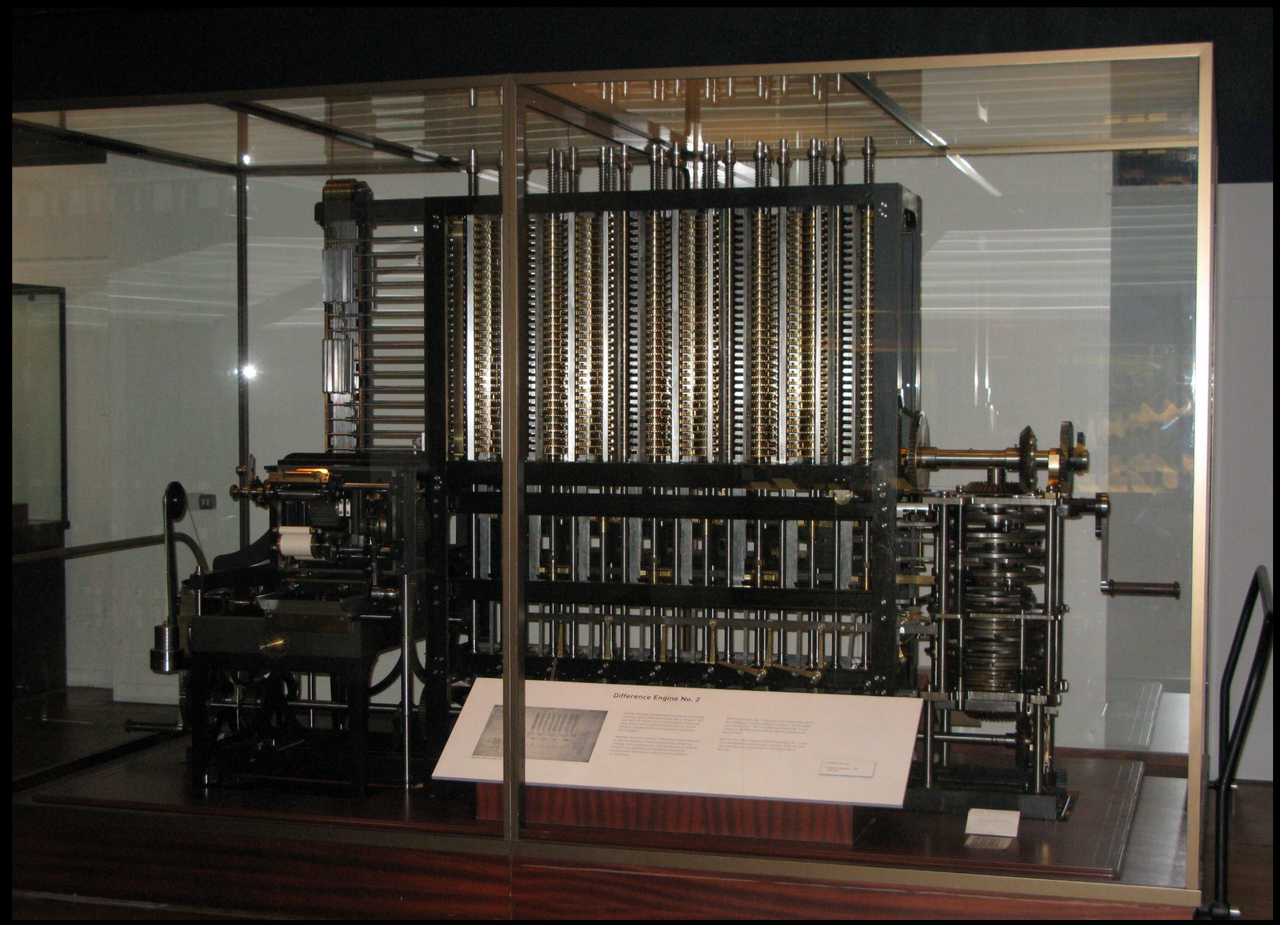
Problem – chapter 4

And everyone is trying to migrate their stack onto the Cloud…



Solution – automation in the form of domain-specific languages (DSLs) and compiler technology

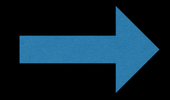
We developed **Devito** makes it easy for scientists and engineers to write high-performance and performance-portable finite-difference software for CPUs and GPUs – and for whatever architecture is around the corner.



Babbage Difference Engine, Science Museum, London – first attempt at automating the calculation of finite differences

Traditional HPC software development

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$



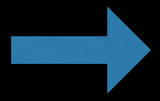
```
void kernel(…) {
```

- Inpenetrable code, often with architecture specific programming models and optimizations (data layout, intrinsics, …). Code regularly mothballed when developer or architecture leaves the scene.
- Staggering cost of redevelopment and maintenance of proprietary software based on 300 old, well known, mathematics.
- Opportunity losses due to the divide between the domain experts who can use insight to innovate novel solutions and the HPC experts who have the skills to optimise mathematical software on different architectures.

```
}
```

Domain Specific Language (DSL) approach

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$



Devito DSL

Domain Specific Language

- Embedded in Python
- Subclasses SymPy – symbolic math

```
eqn = m * u.dt2 - u.laplace
```

Devito Compiler

```
void kernel(…) {
  …
```

- Automatically generated, optimized code
- Just-in-time compilation
- JIT-backdoor for guru's working on new features/optimizations

```
}
```

Devito is a domain-specific language (DSL) for finite-differences

The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise

Edsger W. Dijkstra

Features

- Language semantics embedded in Python - enable developers to write finite-difference solvers at the same level of abstraction as the mathematics.
- Compiler uses symbolic computation to lower problem specification into an intermediate representation decorated with all data dependencies, followed by a range of compiler optimisation passes before being lowered to the target specific source code.

Scientists write **performance portable software** in symbolic maths not loops

- Popular with researchers coding in Jupyter notebooks, quick prototyping etc.
- Quickly switch from research to production – one commercial user is currently running at a yearly average of multiple petaflops 24/7, Devito ~x10 faster than previous code.

Devito features

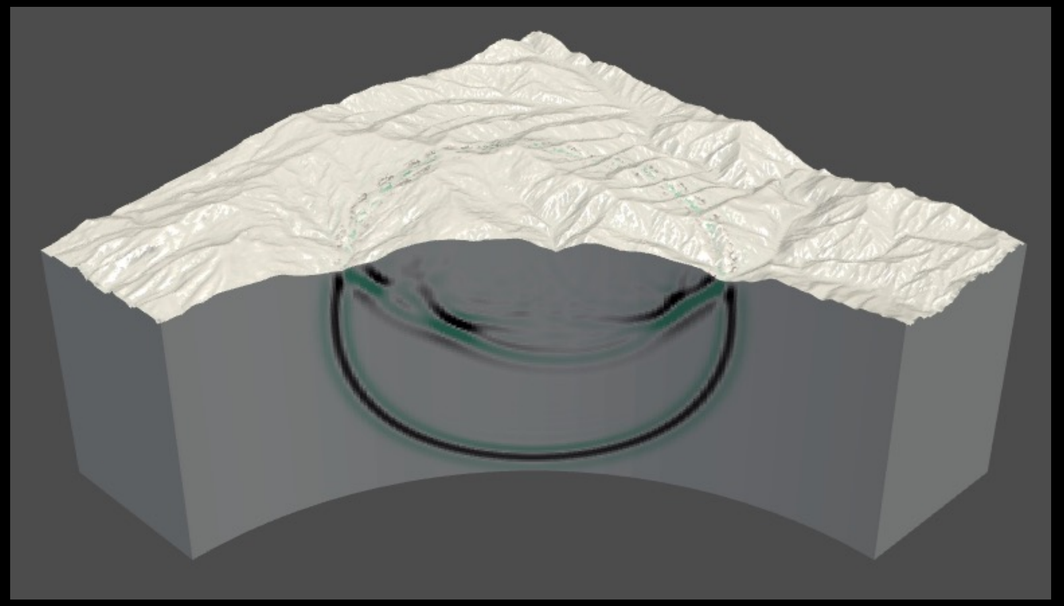
Open source platform – MIT license:
- Open-source/maintained supported by industry consortium.
- Spin-out company Devito Codes Ltd using an open core business model.

Python package:
- Easy to learn.
- Interoperable with Python ecosystem, including ML.
- Interoperable with other languages (Julia, C).

Supported backends:
- {C, SIMD, OpenMP, OpenACC} + MPI
- CPUs: Intel, AMD, ARM, GPUs: NVIDIA, AMD

Best practices software engineering:
- CI/CD (87% code coverage)
- Rigorous code verification
- Code review on all PR's

- Explicit finite-difference schemes
- Sparse functions (sources and receivers)
- Subdomains (eg boundary conditions)
- Staggered grids
- Immersed boundary
- …



Credit to Ed Caunt et al.

Devito JIT-backdoor

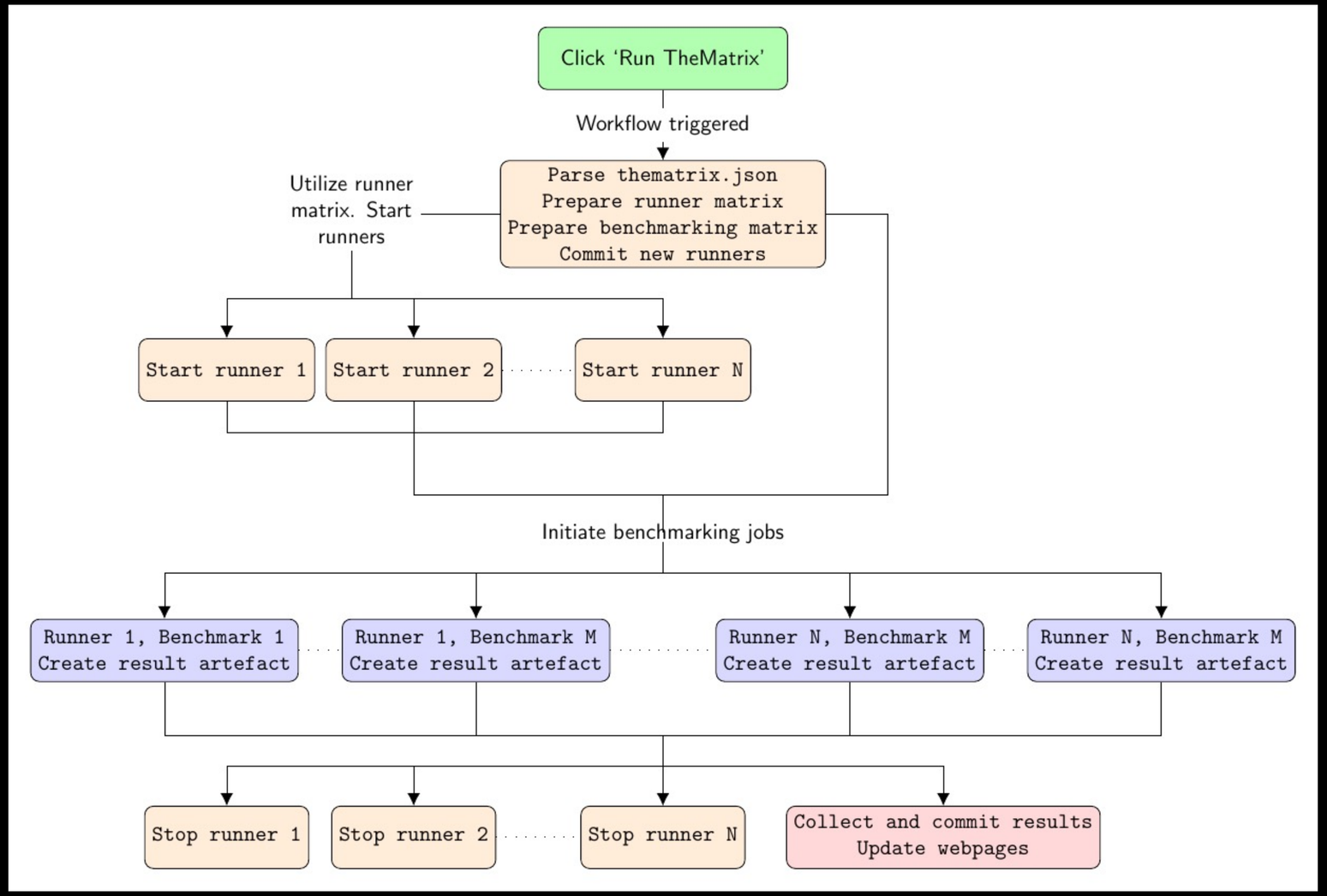
Problem

- There will always be an need for new features, innovation, customisation.
- Blackbox evil for devel platform; it takes away opportunity to learn and innovate.
- Most DSL's failed because:
  - They are not designed by a multidisciplinary team (throw over the wall mentality).
  - Abstractions are too rigid and difficult for anyone outside core compilers developers to contribute.

Solution (i.e. how to pop the bonnet)

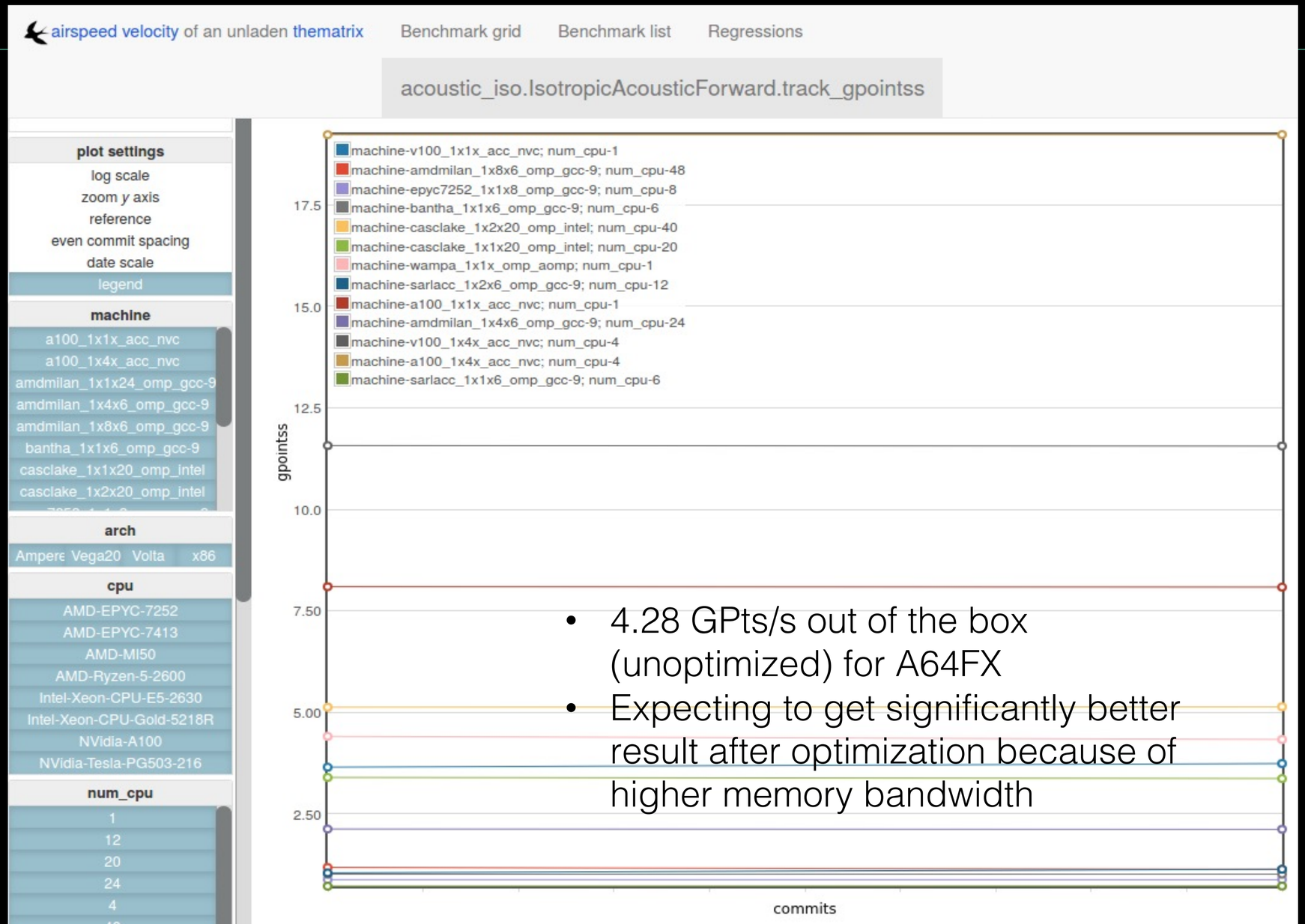
- Devito JIT-backdoor refers to an option to keep the generated source code for experimentation.
- Generate quality human readable code.
- Scientists use to prototype new features.
- Vendors use to perform in-depth performance analysis and performance optimisation.
- "Hacked" code is analysed by Devito developer and encoded into Devito compiler.

TheMatrix – programmatically benchmark every seismic operator across all available computer architectures.



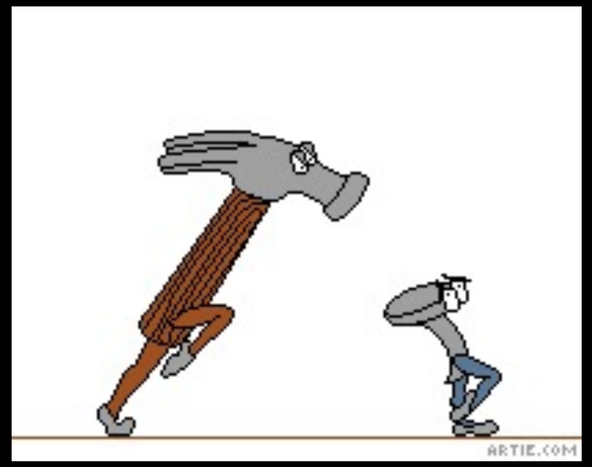
- Facilitates apples-to-apples performance comparisons.
- Tracks performance history so we ensure code monotonically improves in performance

Sample of results



- 4.28 GPts/s out of the box (unoptimized) for A64FX
- Expecting to get significantly better result after optimization because of higher memory bandwidth

Outlook



- **Devito is a finite-difference development platform (akin to FEnICS and Firedrake for finite-element method).**
  - **Community, collaborators, commercial partners use Devito as a platform for end user applications.**
- **Seismic imaging**
  - Acoustic, TTI, elastic, visco-elastic, pore-elastic propagators, various boundary conditions etc… and their adjoints
- **Lluis. Guasch et al. on medical imaging - see E32059 GTC talk on GPU-accelerated neuroimaging using FWI-Devito where we achieved a x20 speedup out-of-the-box.**
- Developing Devito-core for pyTorch – targeting large (distributed memory) **neural networks**
- **CFD problems** in renewable energy and civil nuclear engineering.
- Black-Scholes in **finance.**
- Virtually any partial differential equations on structured grids;
  more generally, any stencil-like code.
- Continue to add support for more architectures.

Acknowledgements

Big thanks to the current Devito **open-source** partners for their support:

- **AMD**
- **BP**
- **DUG**
- **Microsoft**
- **NVIDIA**
- **Petrobras**
- **Shell**

Many thanks to our community of users and contributors!

Releases and full list of contributors:
https://github.com/devitocodes/devito/releases

Imperial College London