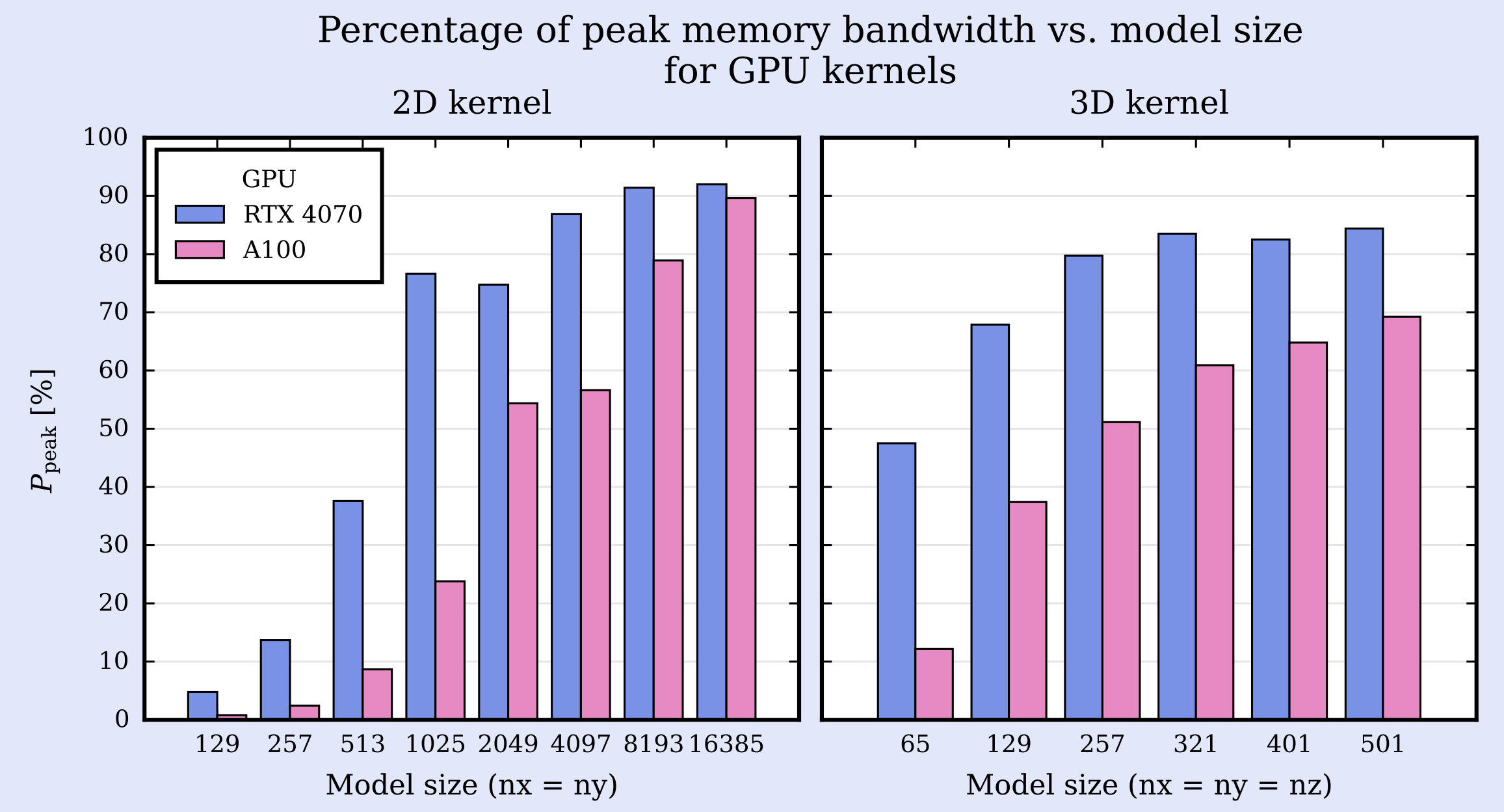
SeismicWaves.jl: an efficient yet user-friendly Julia package for Full-Waveform Inversion on multi-xPUs

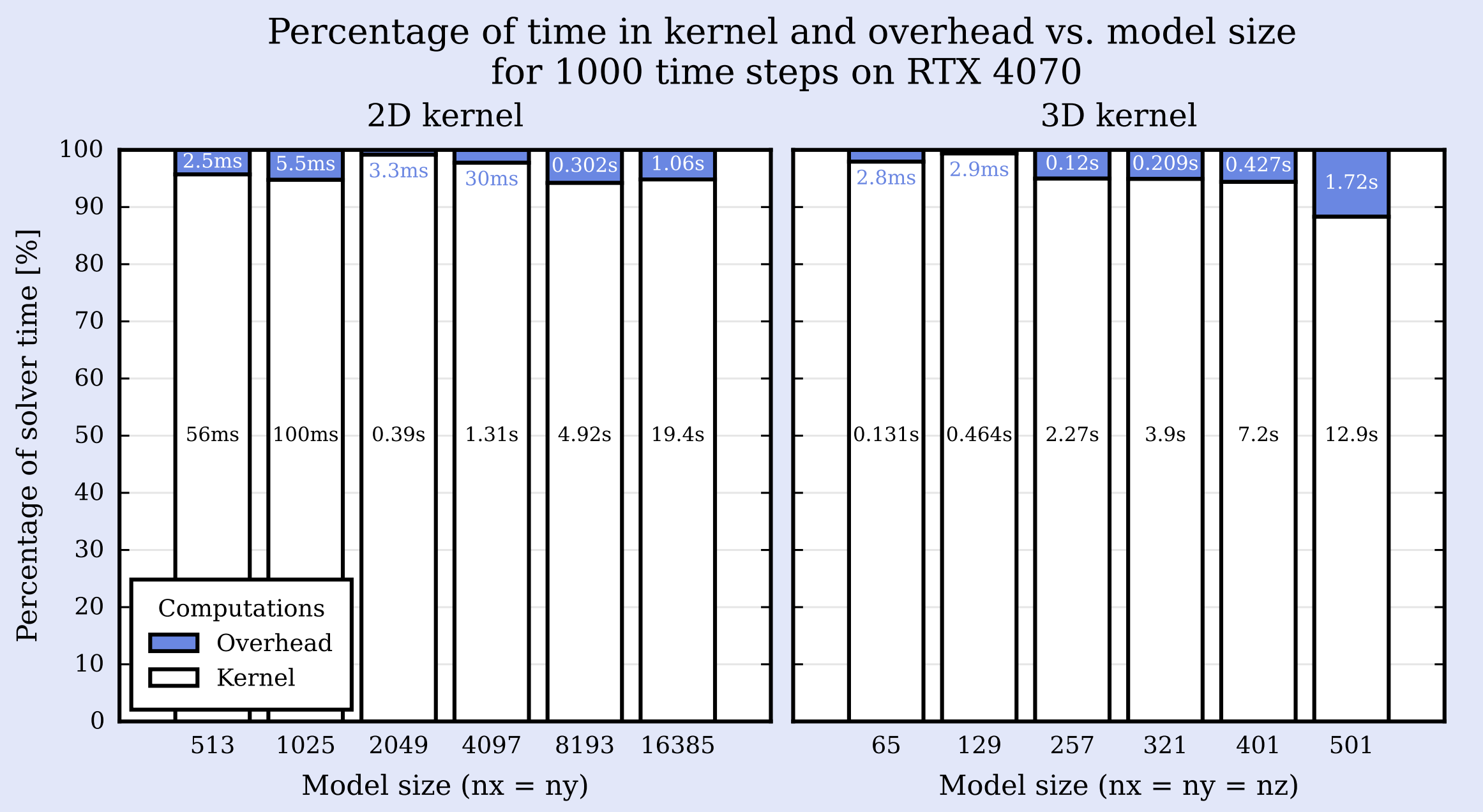
Giacomo Aloisi, Andrea Zunino and Andreas Fichtner
Institute of Geophysics, ETH Zürich, Switzerland <giacomo.aloisi@erdw.ethz.ch>

Built for efficiency

Achieving close to peak performance



Most of computation time spent in kernel



Open-source GitLab repository

2

Abstract
SeismicWaves.jl is a package for **finite-differences-based** wave simulations in the context of Full-Waveform Inversion for **seismic tomography** problems. It is written 100% in the **Julia** programming language and by leveraging its **parallel** capabilities and **easiness of use**, it can provide scientists with **rapid prototyping** in FWI scenarios where performance and efficiency are critical factors. The developed code can then be run on multi-node clusters, automatically adapting to the specific hardware and enabling true ***multi-xPUs*** computing.

Easy to use and adapt to new features

Acoustic 2D variable density forward code snippet

```
using SeismicWaves, CUDA, HDF5

# Load velocity and density models, sources and receivers positions from HDF5 file
vp, rho, possrcs, posrecs = h5open("setup.h5", "r") do ... end
# Numerics
nx, ny, nt = size(vp), 25000                    # number of grid points and iterations
Δh, Δt = 0.01666666, 0.002                      # grid step and time step sizes [m, s]
# Ricker source time function with central frequency f0 [Hz] and activation time t0 [s]
stf = reshape(rickerstf.([(i-1)*Δt for i in 1:nt], t0=1.0, f0=2.0), nt, 1)
# Single shot setup
shot = Shot(ScalarSources(possrcs, stf, f0), ScalarReceivers(posrecs, nt))
# Setup parameters
bdcs_params = CPMLBoundaryConditionParameters(halo=20, rcoef=0.001, freeboundtop=true)
params = InputParametersAcousticVariableDensity(nt, Δt, (nx, ny), (Δh, Δh), bdc_params)
# Build wavesim
wavesim = build_wavesim(params; parall=:GPU)
# Run forward simulation
swforward!(wavesim, VpRhoAcousticVDMaterialProperty(vp, rho), [shot])
```

import SeismicWaves into your code

create shot(s) with sources and receivers configuration

setup wave simulation parameters

build wave simulation and backend selection

create material properties

run the simulation

Only 10 lines of code to run a forward simulation!

Gradients with respect to material properties require just a few more lines

Complete control over choice of misfit and regularization with possibility to implement your own!

3

Open-source GitLab repository

Sharing is encouraged

Functionalities
- **Forward** and **adjoint** 4th order finite-difference for **acoustic** and ***elastic*** waves
- **Reflective** and **CPML** (absorbing) boundary conditions
- **1D**, **2D**, and **3D** rectangular domains on a regular uniform grid
- Adjoint-based **gradients** with respect to material properties
- Linear **checkpointing** to store intermediate timesteps for adjoint computations

Poster abstract

1

SeismicWaves.jl

4

Leveraging state-of-the-art HPC packages and techniques
ParallelStencil.jl and ***ImplicitGlobalGrid.jl*** enable device-agnostic distributed computing (***multi-xPUs***) making SeismicWaves.jl suitable for all needs: from quick prototyping on your laptop to large-scale simulations on supercomputing clusters. Performance benchmarks show close to peak performance utilization (up to 90%) on modern GPUs and ideal weak scaling efficiency on distributed systems.

This presentation participates in OSPP

Outstanding Student & PhD candidate Presentation contest

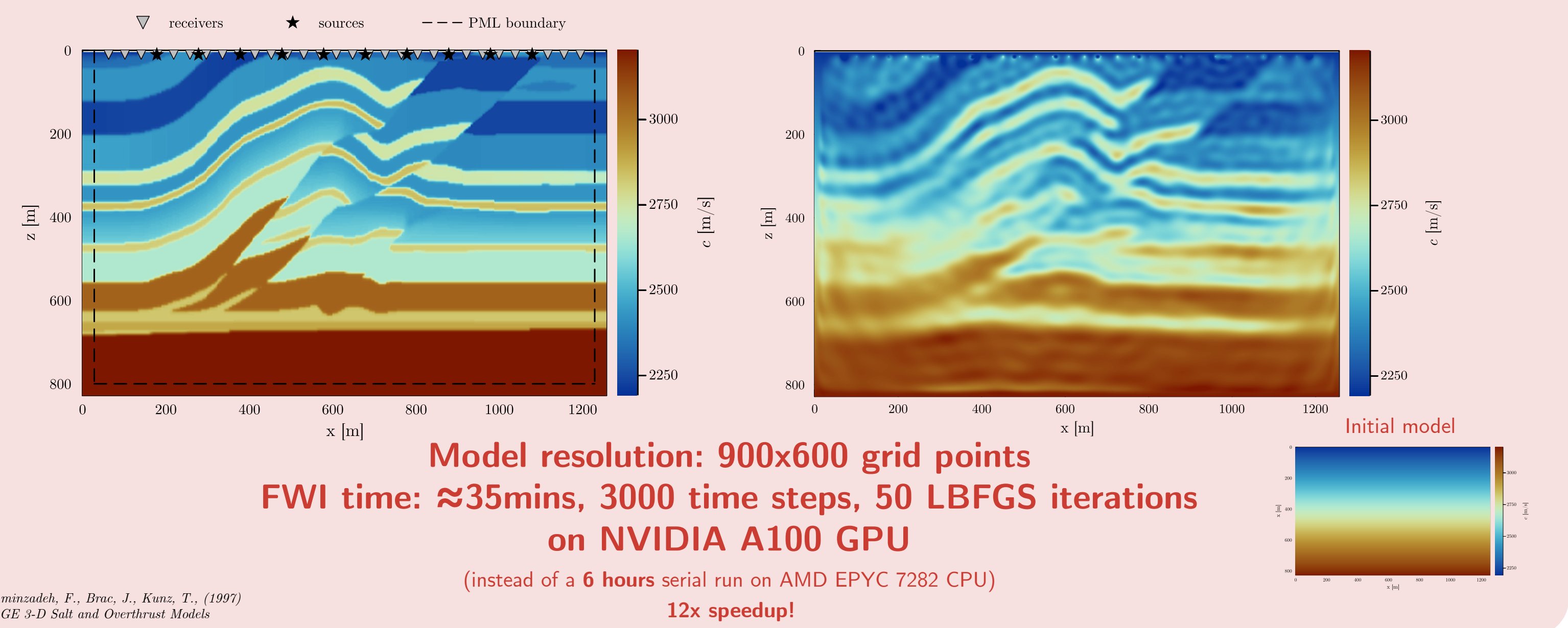
Poster abstract

Designed for Full-Waveform Inversion

Overthrust 2D acoustic FWI w/correlated noise

Target model + source/receivers setup

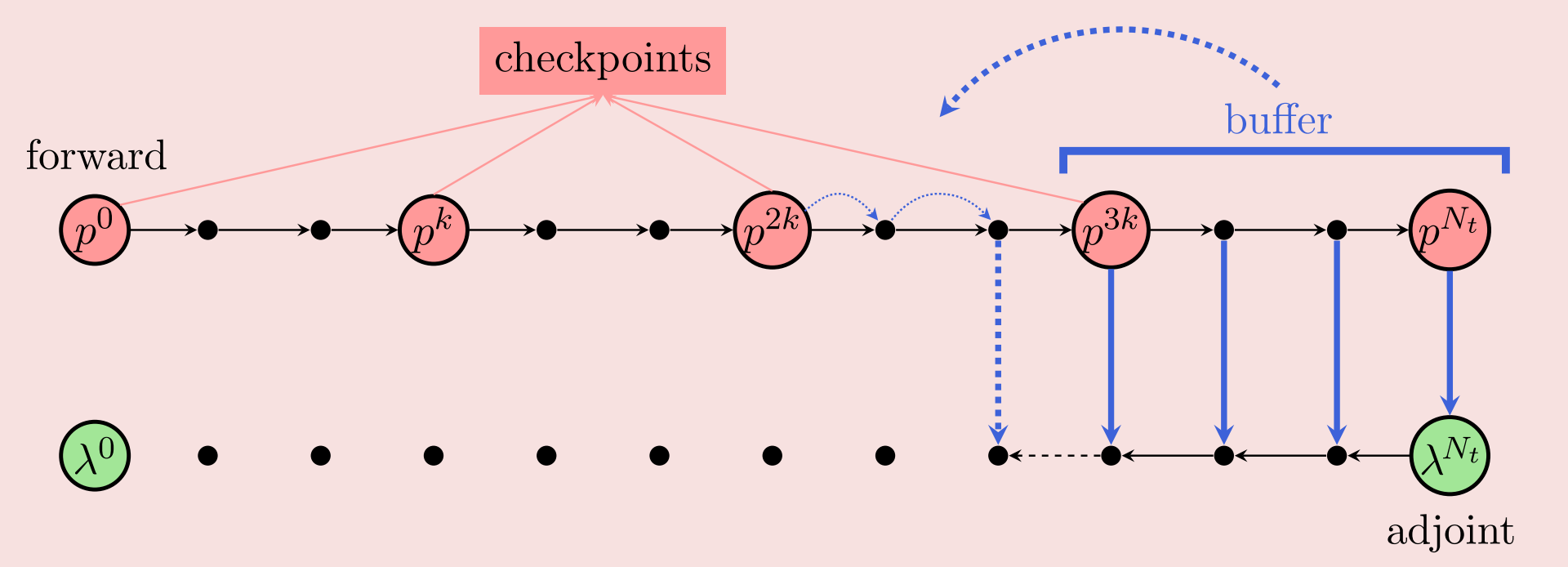
Reconstructed model



Model resolution: 900x600 grid points
FWI time: ≈35mins, 3000 time steps, 50 LBFGS iterations on NVIDIA A100 GPU
(instead of a **6 hours** serial run on AMD EPYC 7282 CPU)
12x speedup!

models from Aminzadeh, F., Brac, J., Kunz, T., (1997) SEG/EAGE 3-D Salt and Overthrust Models

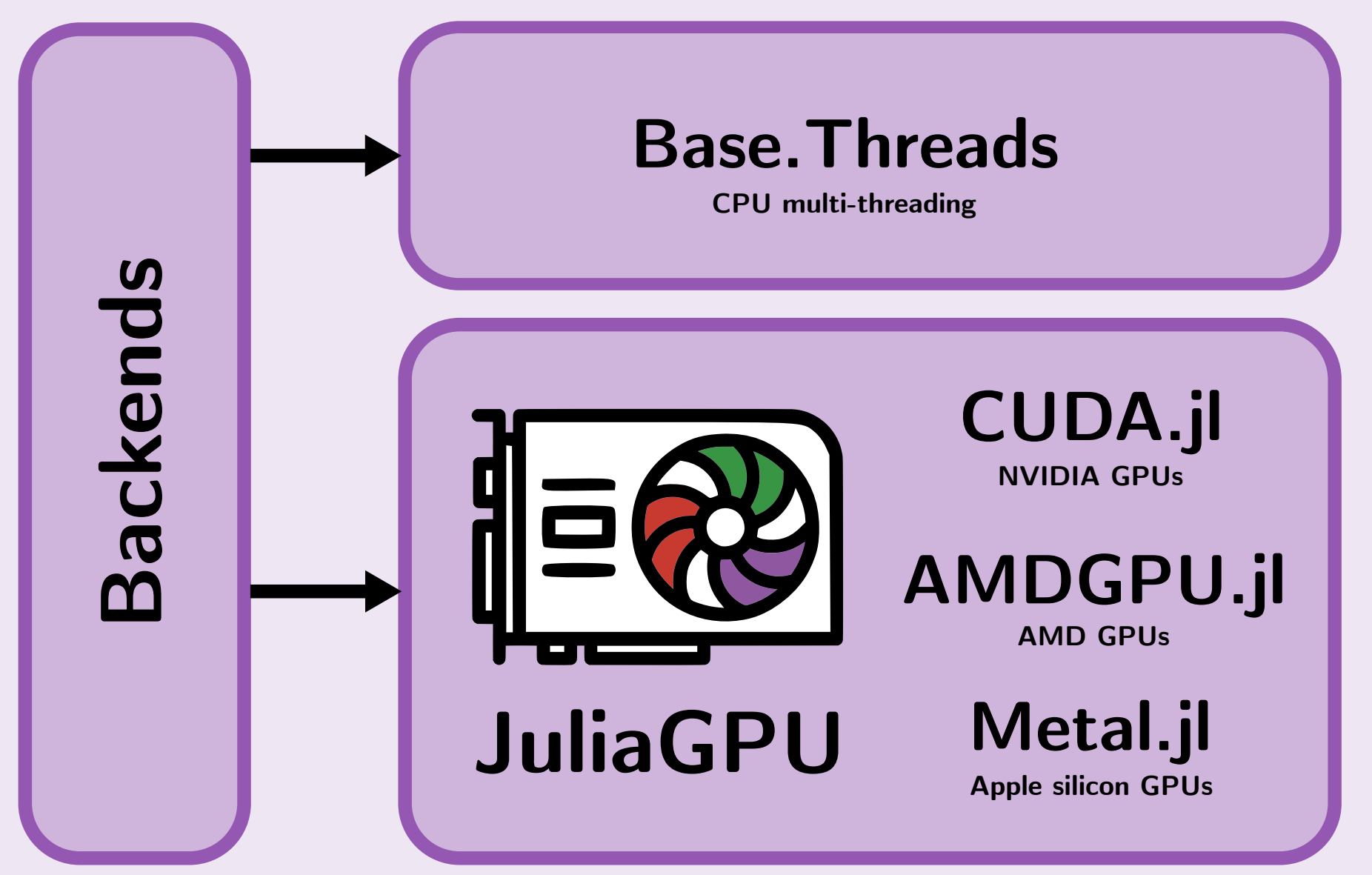
Checkpointing for storing timesteps



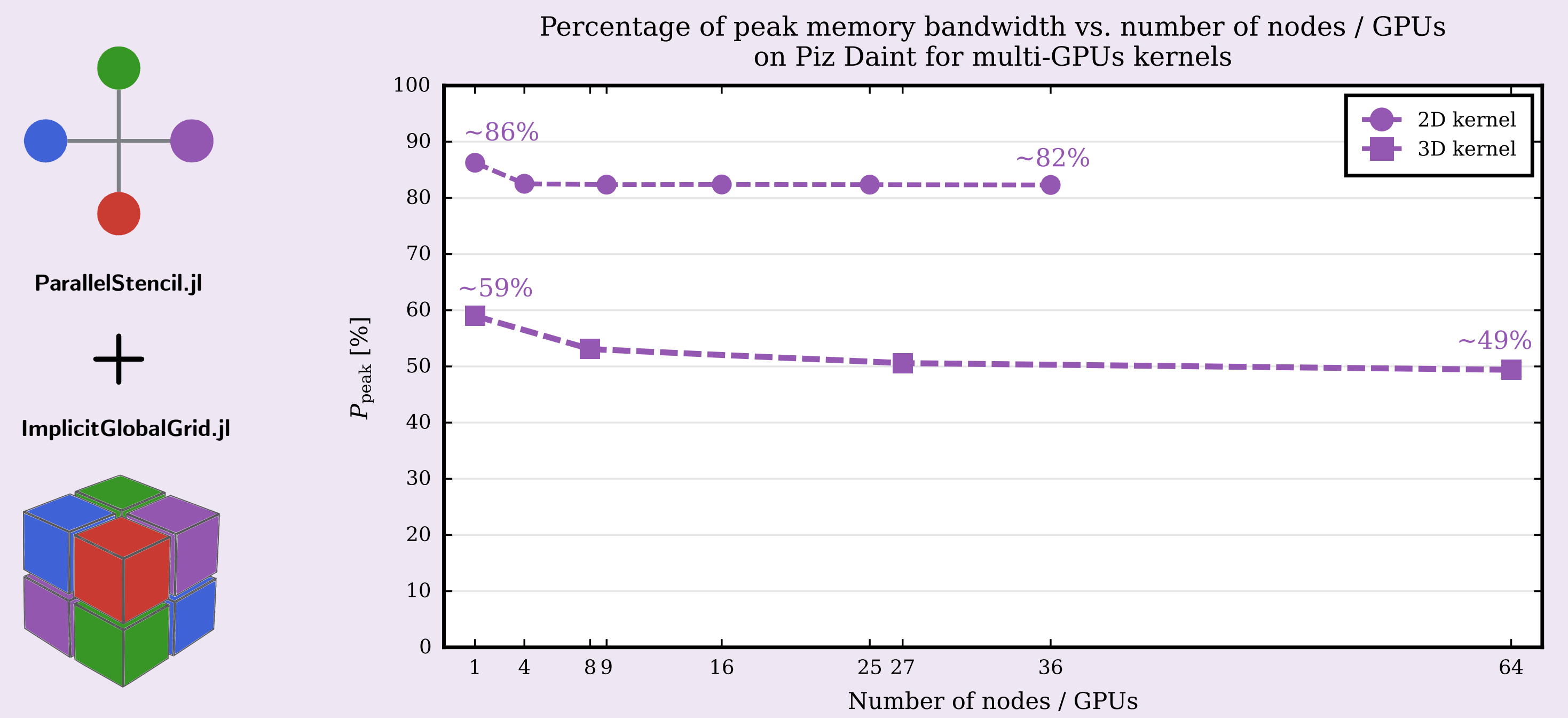
Linear checkpointing saves up storage requirements:
only need to store $\mathbf{2k = 2\sqrt{N_t}}$ instead of $\mathbf{N_t}$ timesteps
(at the cost of one more forward simulation)

Device-agnostic and scalable

ParallelStencil.jl backend selection



Weak scaling benchmark shows ideal scaling efficiency



EGU General Assembly 2024

References
- Aloisi G., Zunino A., Fichtner A., (2023) **Full Waveform Inversion for Medical Ultrasound Tomography in Julia on multi-xPUs**, MSc Thesis, ETH Zürich
- Zunino A, Gebraad L., Ghirotto A. and Fichtner A. (2023), **HMCLab: a framework for solving diverse geophysical inverse problems using the Hamiltonian Monte Carlo method**
- Omlin S., Räss L., (2022), **High-performance xPU Stencil Computations in Julia**

Features written in oblique are work in progress!
SeismicWaves.jl v0.6 pre-release available
Features planned on v1.0 final release
- fully-fledged multi-xPUs implementation using PS.jl + IGG.jl
- P-SV elastic implementation on a staggered grid (4th order in space)
- more misfits and regularizations available (only L2 misfit in pre-release)
- framework for Full-Waveform Ambient Noise Inversions (FWANI)

powered by
julia
in collaboration with
HMCLab
ETH zürich
ETH zürich SWP SEISMOLOGY & WAVE PHYSICS