

# GPU4GEO

## Frontier GPU multi-physics solvers



<sup>1,2</sup> Ludovic Räss

<sup>1,2</sup> Ivan Utkin

<sup>1</sup> Albert De Montserrat

<sup>3</sup> Boris Kaus

<sup>4</sup> Samuel Omlin

<sup>5</sup> Thibault Duretz

1 | ETH Zurich

2 | Swiss Federal Institute for Forest, Snow and Landscape Research (WSL)

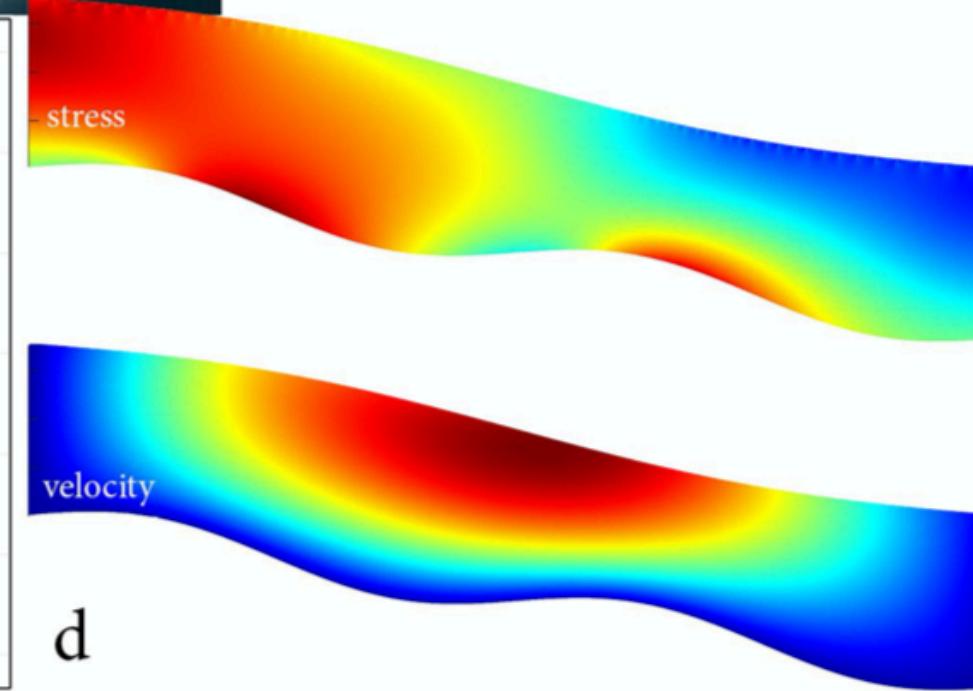
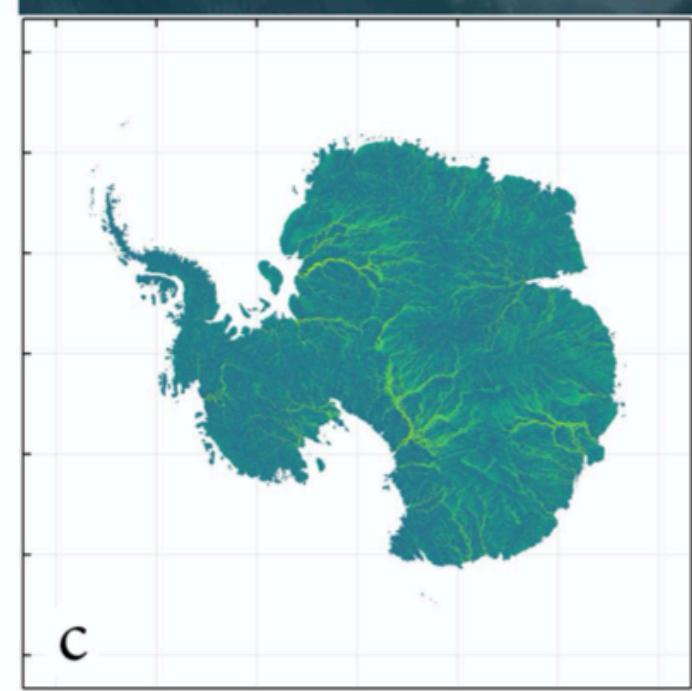
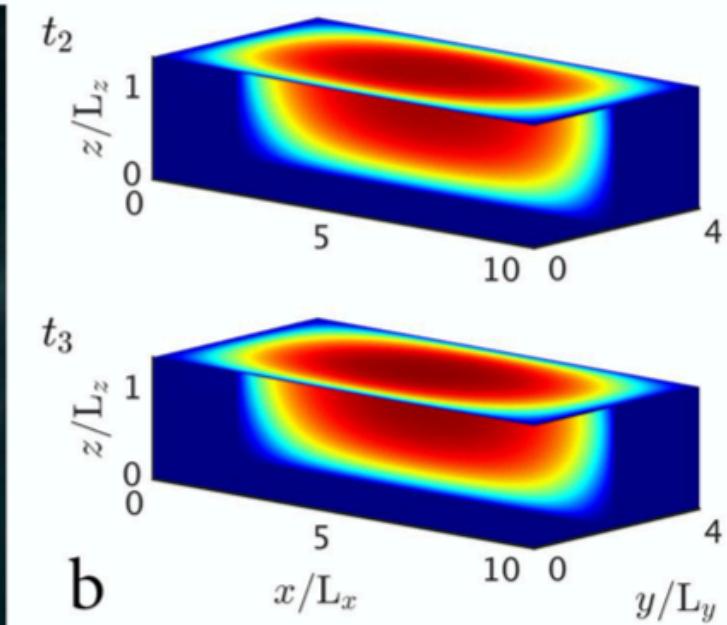
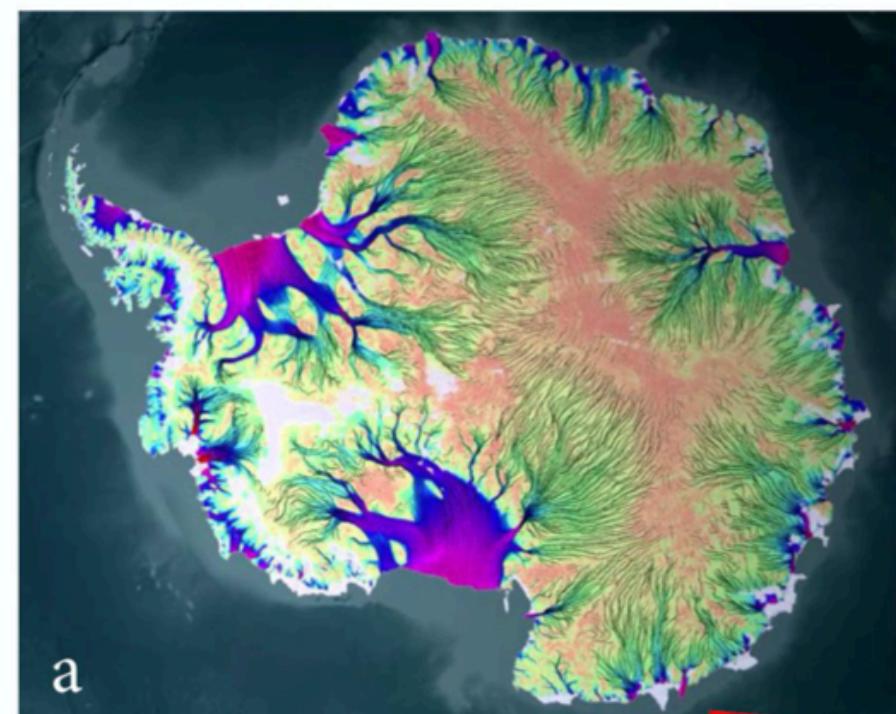
3 | Johannes Gutenberg University Mainz

4 | Swiss National Supercomputing Centre, CSCS

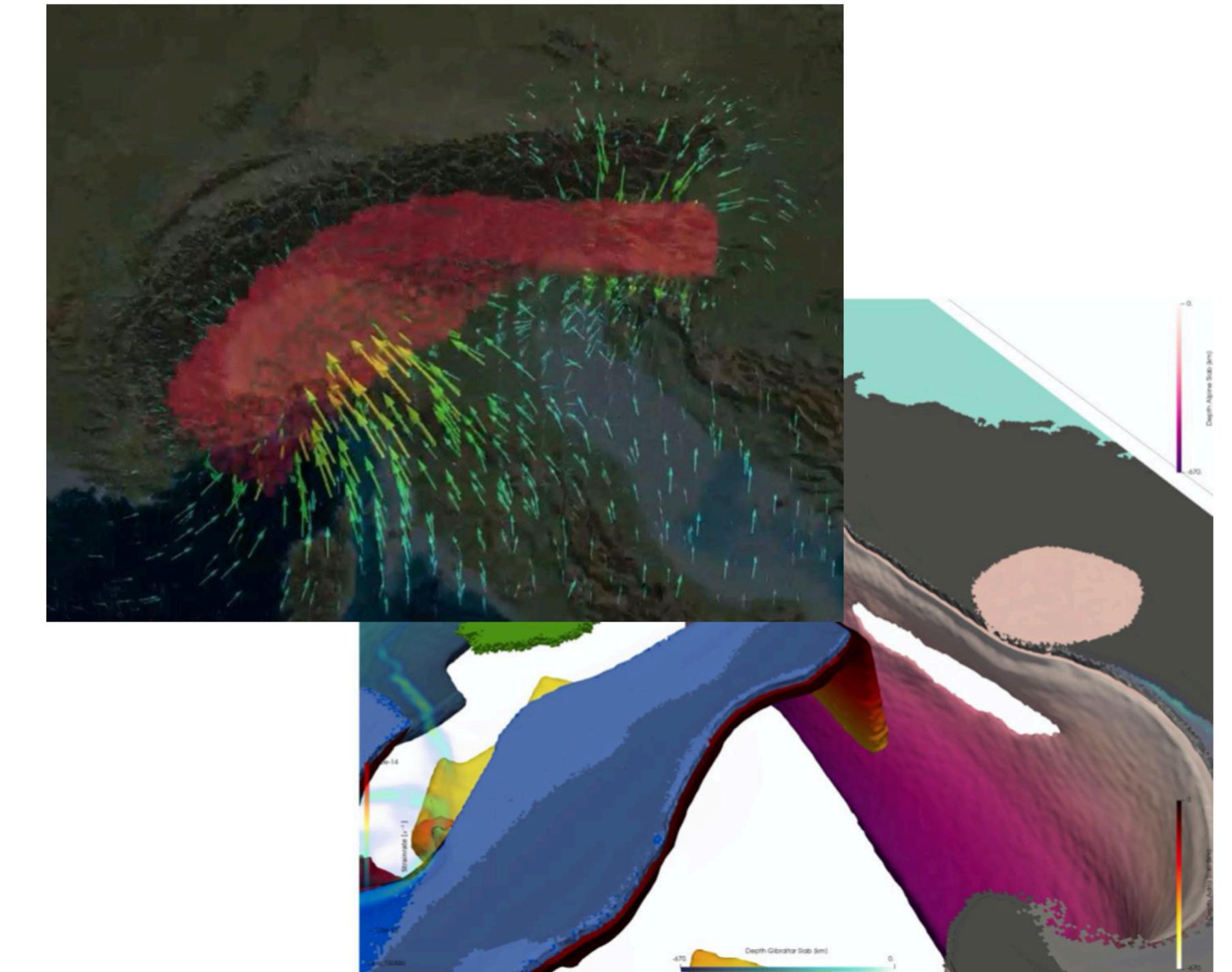
5 | Institut für Geowissenschaften, Goethe-Universität Frankfurt, Frankfurt

# Application to ice flow dynamics and geodynamics

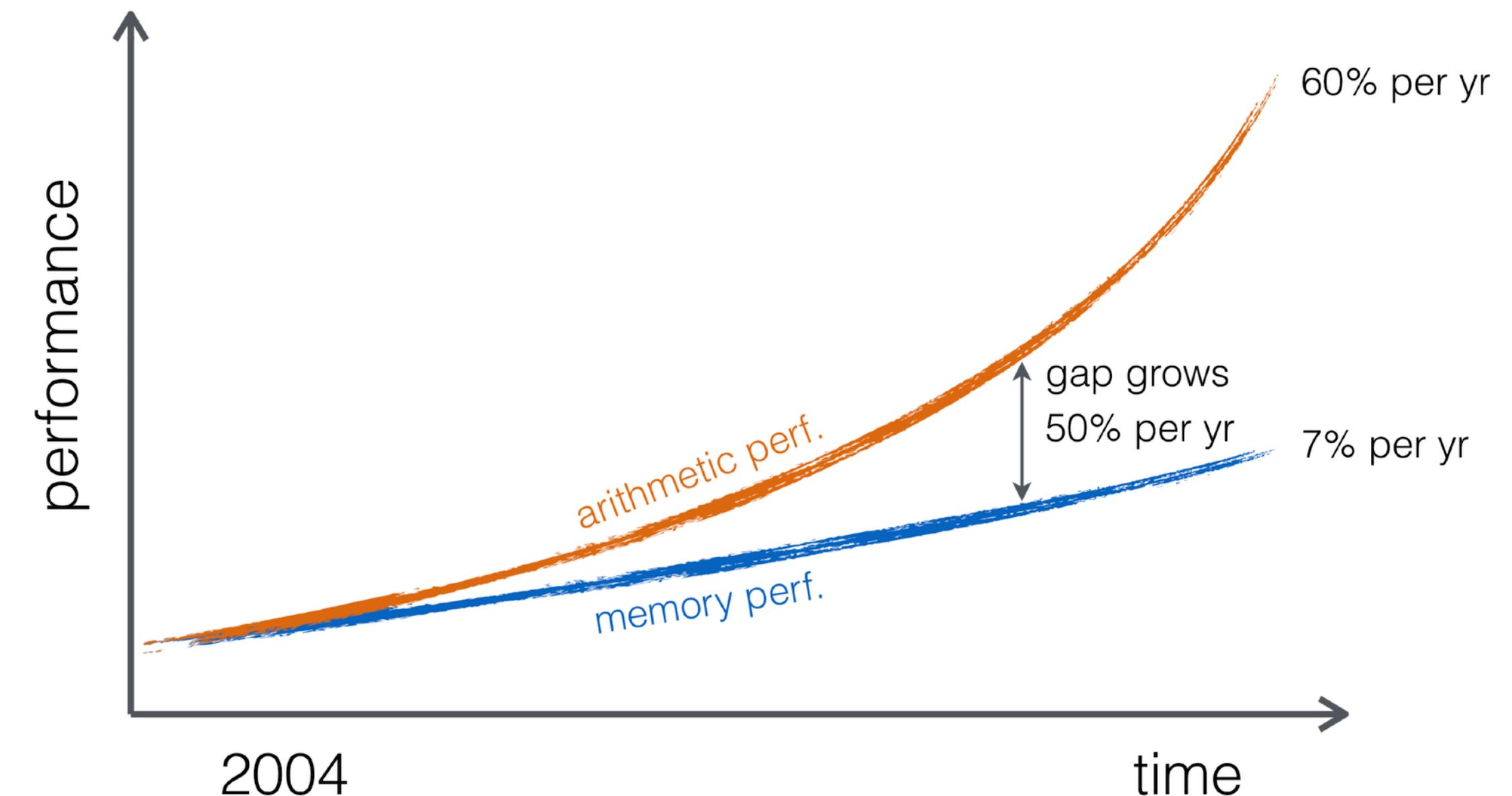
$$\begin{aligned}\nabla \cdot \left( \eta \left[ \nabla u + (\nabla u)^T \right] \right) - \nabla p &= f \\ \nabla \cdot u &= 0\end{aligned}$$



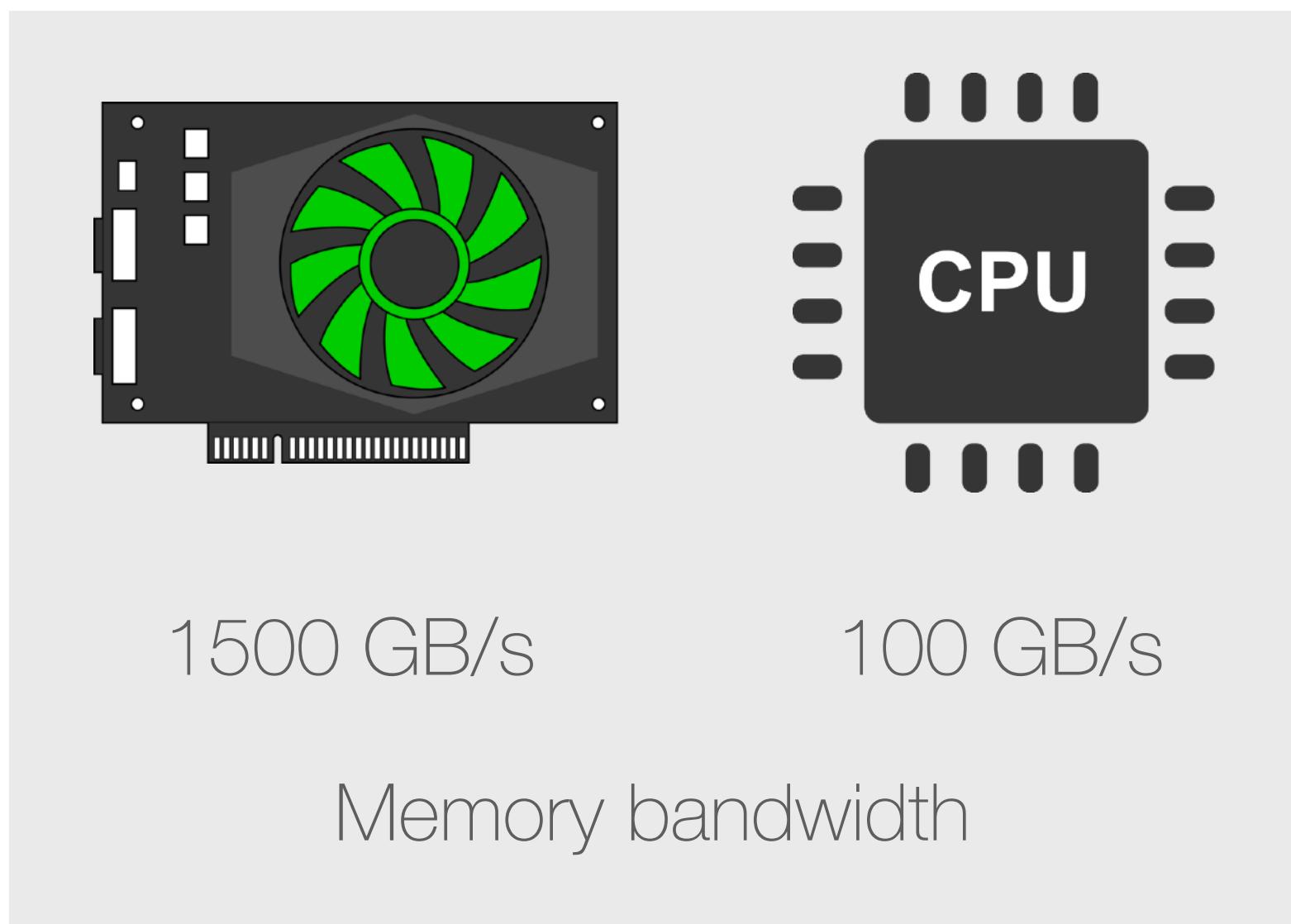
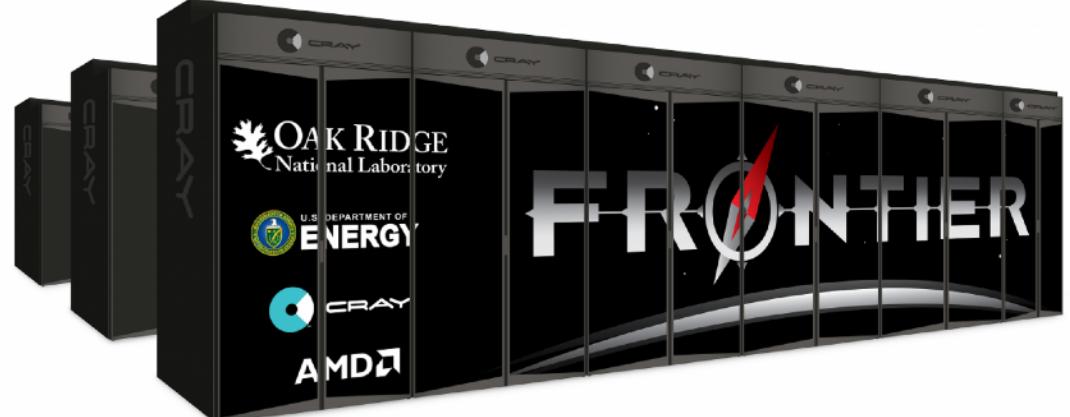
ice flow dynamics



*“The goal of the GPU4GEO project is to propose software tools which provide a way forward in these two domains by exploiting two powerful emerging paradigms in HPC: massively parallel iterative solvers, and HPC with Julia on graphical processing units (GPUs).”*



*“The goal of the GPU4GEO project is to propose software tools which provide a way forward in these two domains by exploiting two powerful emerging paradigms in HPC: massively parallel iterative solvers, and HPC with Julia on graphical processing units (GPUs).”*



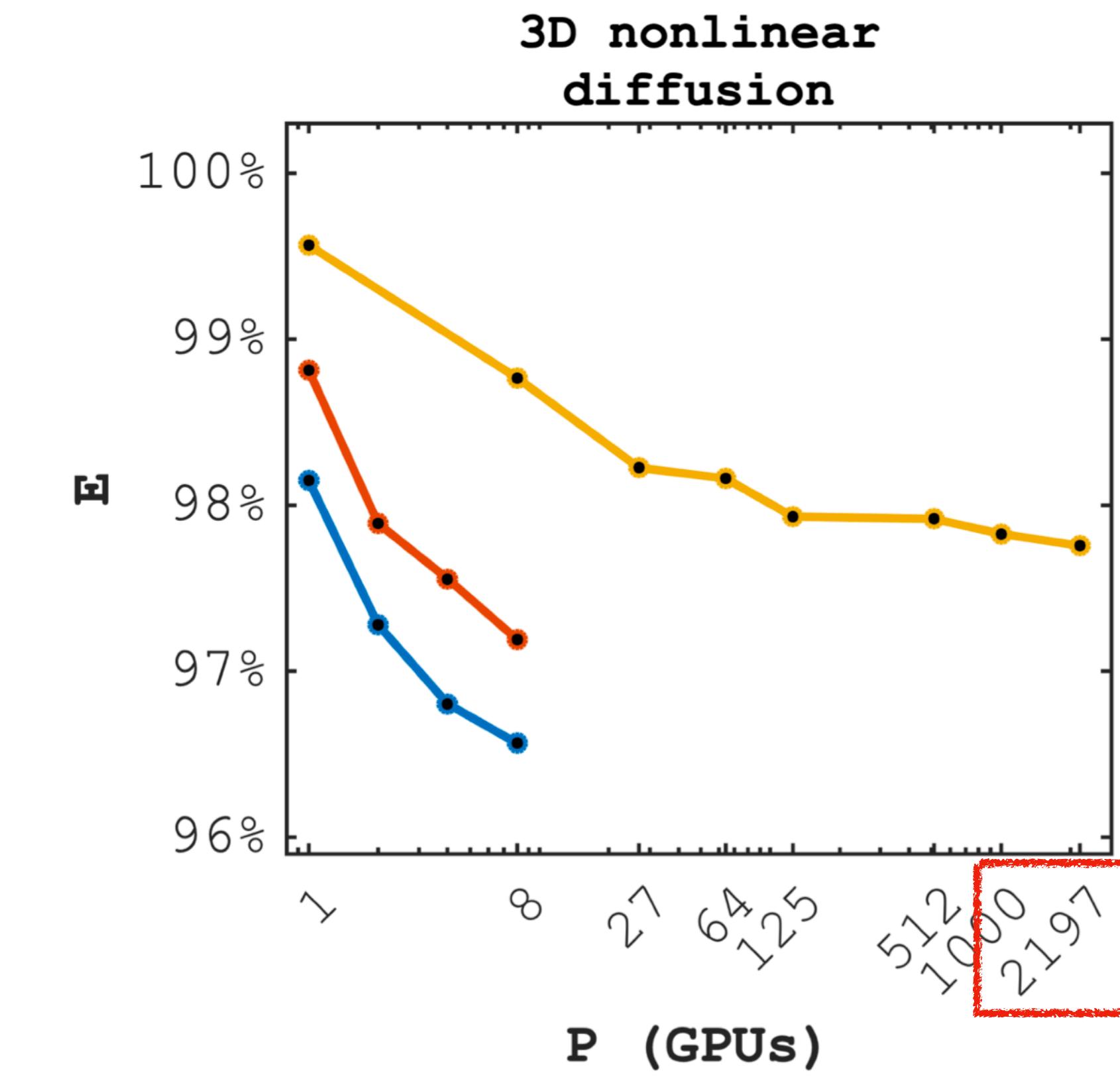
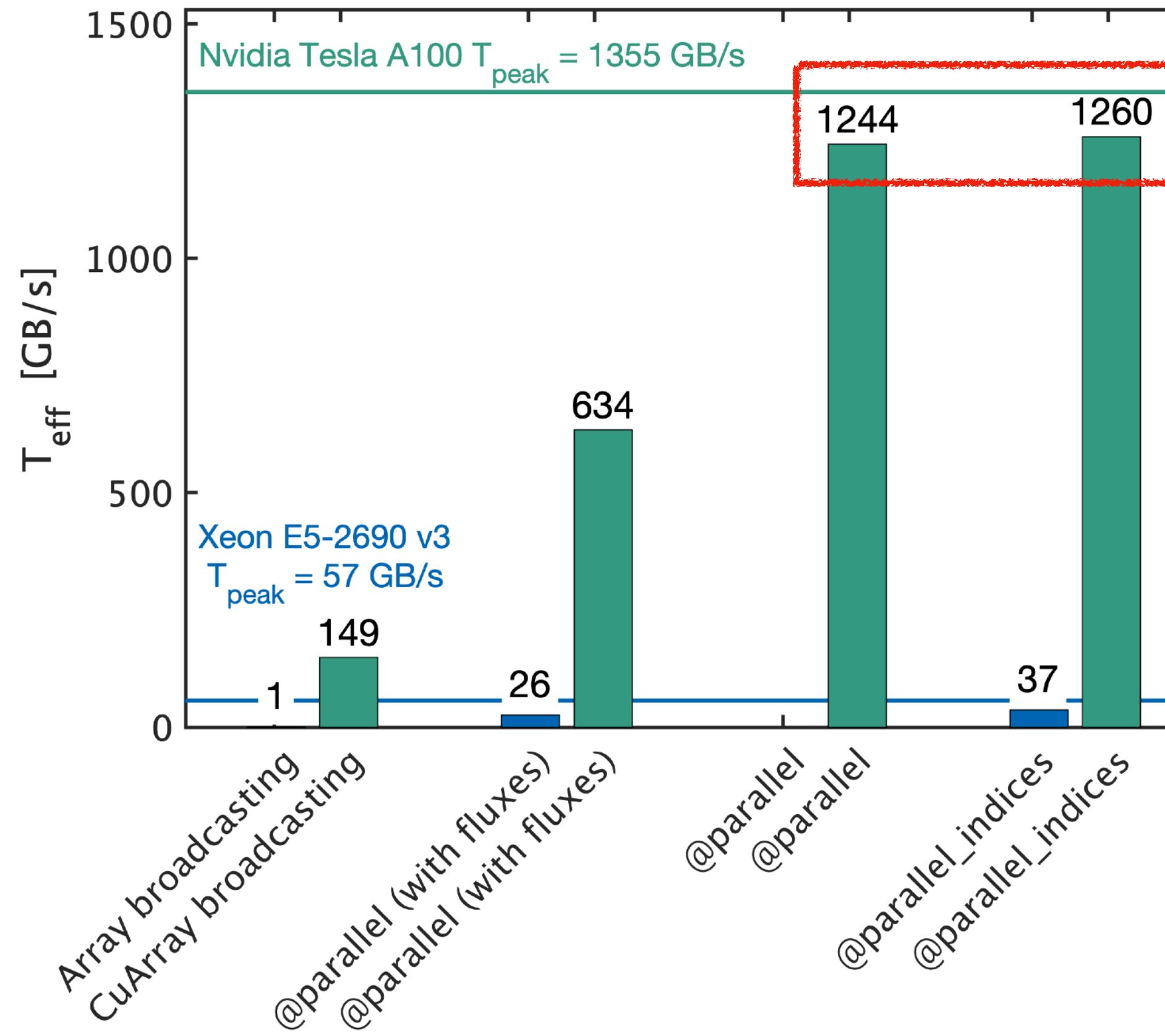
But why **julia** + ?

# Unleash HPC on (multi-)GPUs

 **ParallelStencil.jl** enables parallel stencil computations on GPUs (Nvidia & AMD) and CPUs.



**ImplicitGlobalGrid.jl** enables stencil-based distributed parallelisation of GPU and CPU.

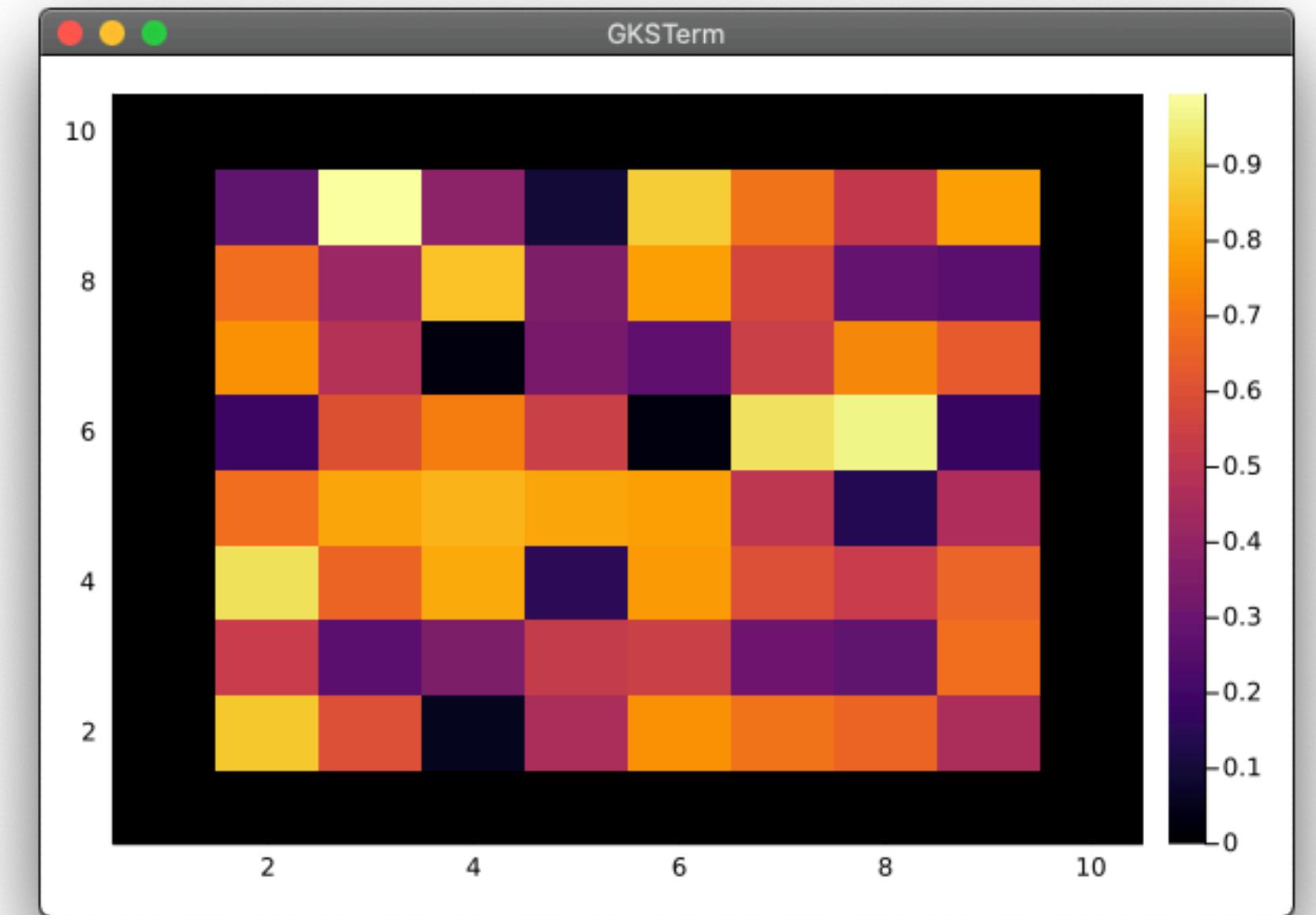


Räss et al. (2022)

# Scalable solvers

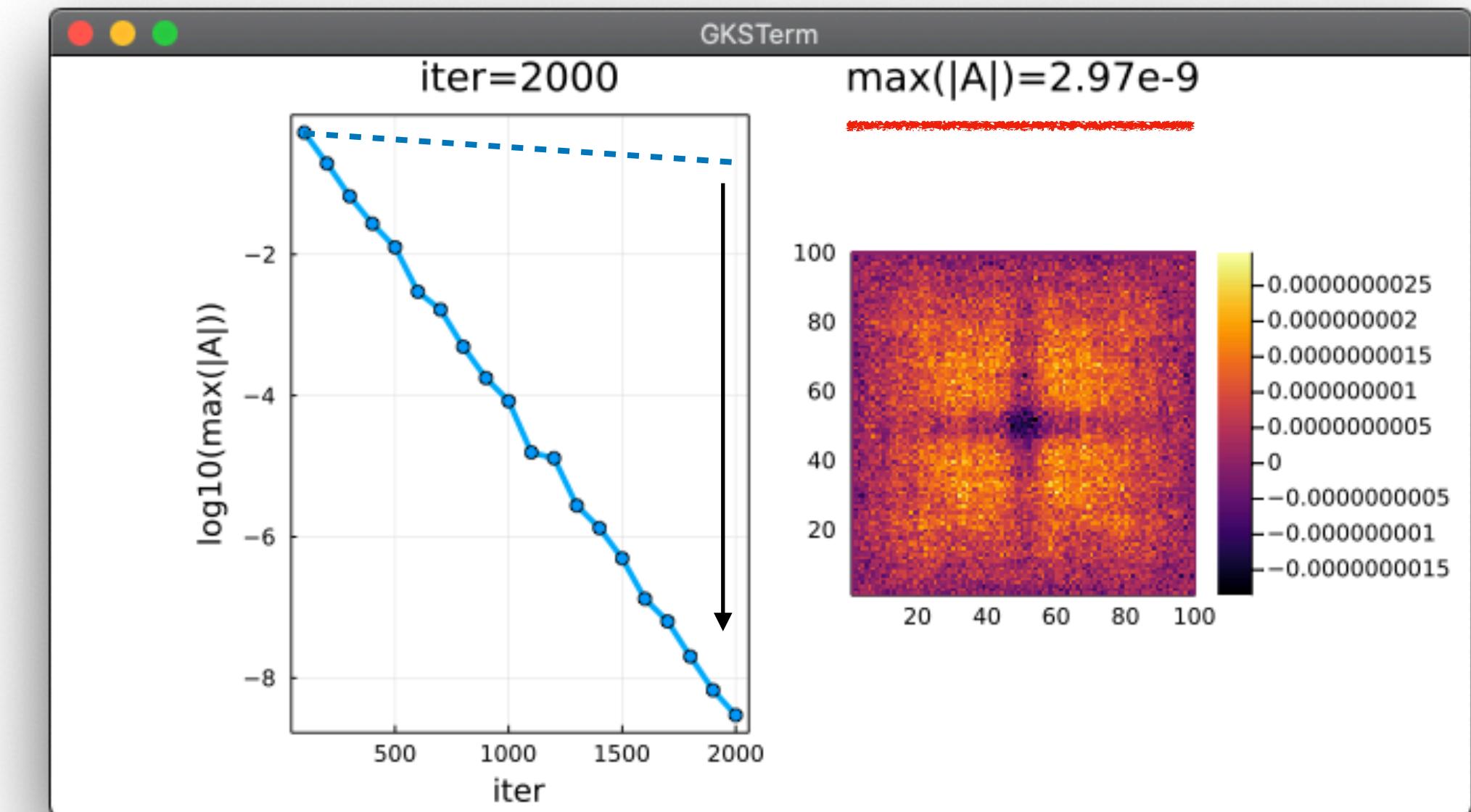
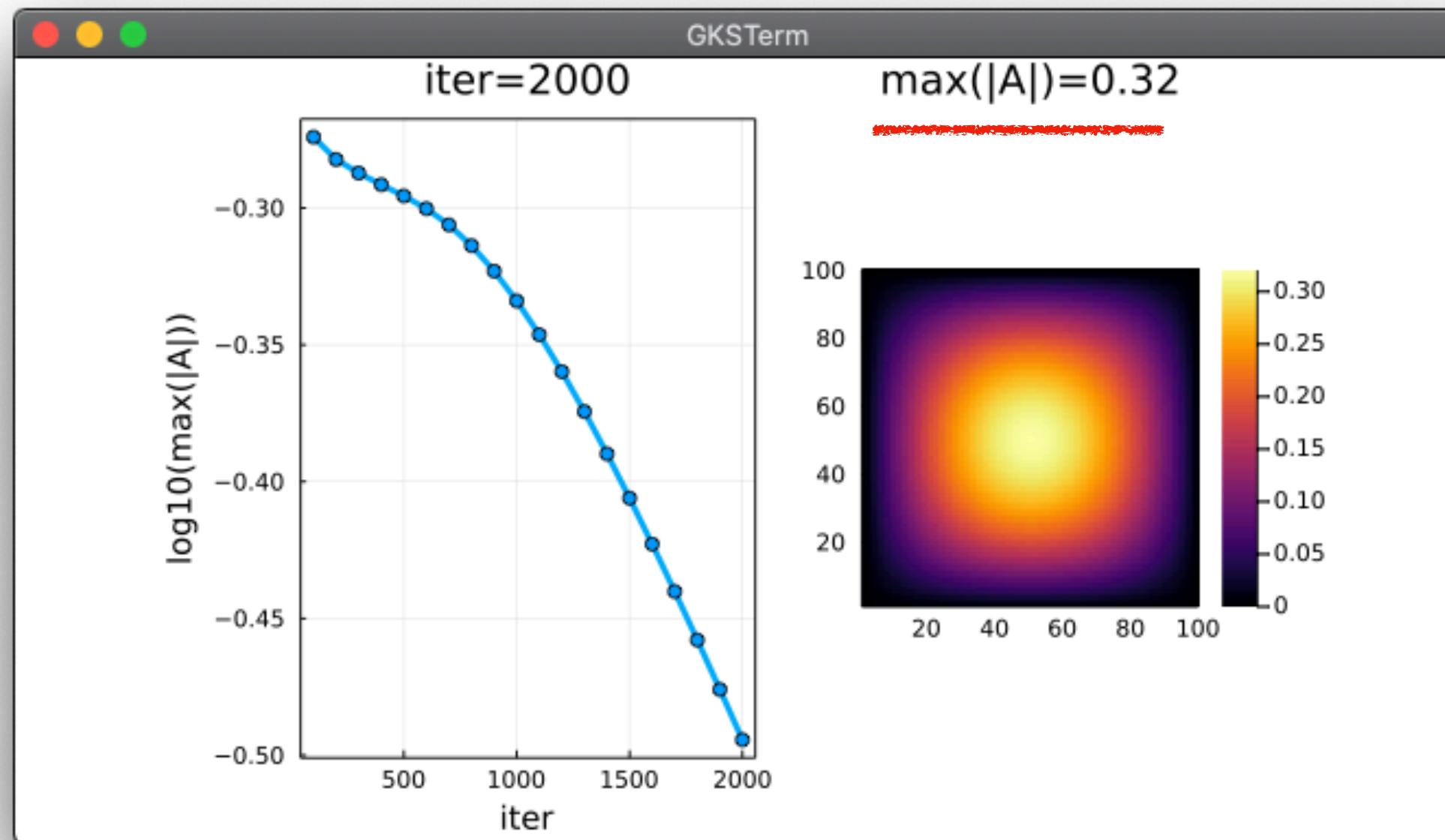
The accelerated pseudo-transient method - using wave-like pseudo-physics to accelerate convergence.

```
@views function laplacian2D()
    order      = 2
    fact       = 2
    # Physics
    lx, ly     = 10, 10
    D          = 1
    # Numerics
    nx, ny    = fact*50, fact*50
    dx, dy    = lx/nx, ly/ny
    niter     = 20*nx
    alpha     = 2.0/nx
    dt        = dx/sqrt(D)/2.1
    # Initial conditions
    A         = zeros(Float64, nx,ny)
    dAdt     = zeros(Float64, nx,ny)
    A[2:end-1,2:end-1] .= rand(nx-2,ny-2)
    # iteration loop
    for it = 1:niter
        dAdt[2:end-1,2:end-1] .= dAdt[2:end-1,2:end-1]*(1-alpha)*(order-1) +
            dt*( diff(diff(A[:,2:end-1],dims=1),dims=1)/dx^2 +
                  diff(diff(A[2:end-1,:]),dims=2),dims=2)/dy^2 )
        A
        .= A + dt*dAdt
    end
    return
end
```



# Scalable solvers

Acceleration is cool - it keeps the iterations count low and scales linearly with  $\text{dx}$



Application	GPU	$n_x$ local	$n_x$ global	$n_{\text{dof}}^{\text{tot}}$	$t/\text{iter}$ (s)	$n_{\text{iter}}^{\text{tot}}$	wall time
Nonlinear diffusion 3D	Tesla P100	512	6632	$0.292 \times 10^{12}$	0.0348	12'932	7.5 min
Visco-elastic Stokes 3D	Tesla P100	383	4955	$1.23 \times 10^{12}$	0.0644	408'788	7.31 h

# Scalable solvers

Acceleration is cool - it keeps the iterations count low and scales linearly with  $\text{dx}$

Geosci. Model Dev., 15, 5757–5786, 2022  
<https://doi.org/10.5194/gmd-15-5757-2022>  
© Author(s) 2022. This work is distributed under the Creative Commons Attribution 4.0 License.

**Geoscientific Model Development**

Open Access

**Development and technical paper**

**Assessing the robustness and scalability of the accelerated pseudo-transient method**

Ludovic Räss<sup>1,2,★</sup>, Ivan Utkin<sup>1,2,3,★</sup>, Thibault Duretz<sup>4,5</sup>, Samuel Omlin<sup>6</sup>, and Yuri Y. Podladchikov<sup>3,7,8</sup>

<sup>1</sup>Laboratory of Hydraulics, Hydrology and Glaciology (VAW), ETH Zurich, Zurich, Switzerland

<sup>2</sup>Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), Birmensdorf, Switzerland

<sup>3</sup>Faculty of Mechanics and Mathematics, Lomonosov Moscow State University, Moscow, Russia

<sup>4</sup>Institut für Geowissenschaften, Goethe-Universität Frankfurt, Frankfurt, Germany

<sup>5</sup>Univ. Rennes, CNRS, Géosciences Rennes UMR 6118, 35000 Rennes, France

<sup>6</sup>Swiss National Supercomputing Centre (CSCS), ETH Zurich, Lugano, Switzerland

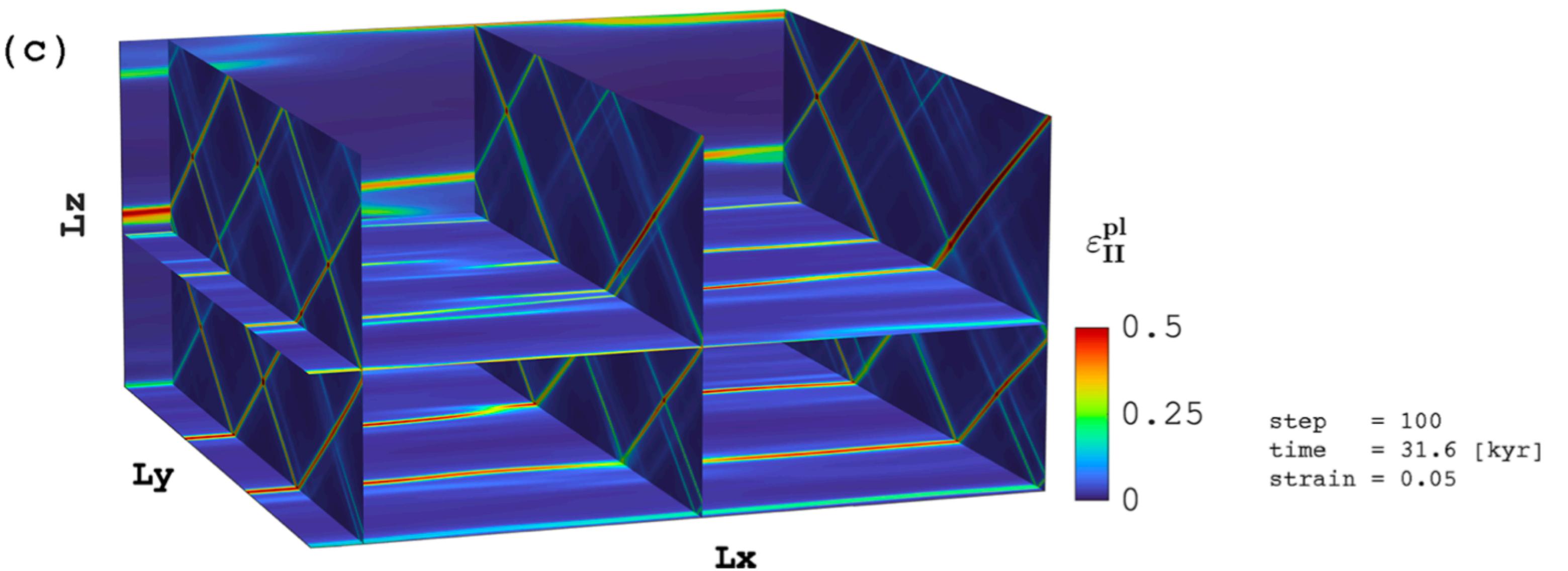
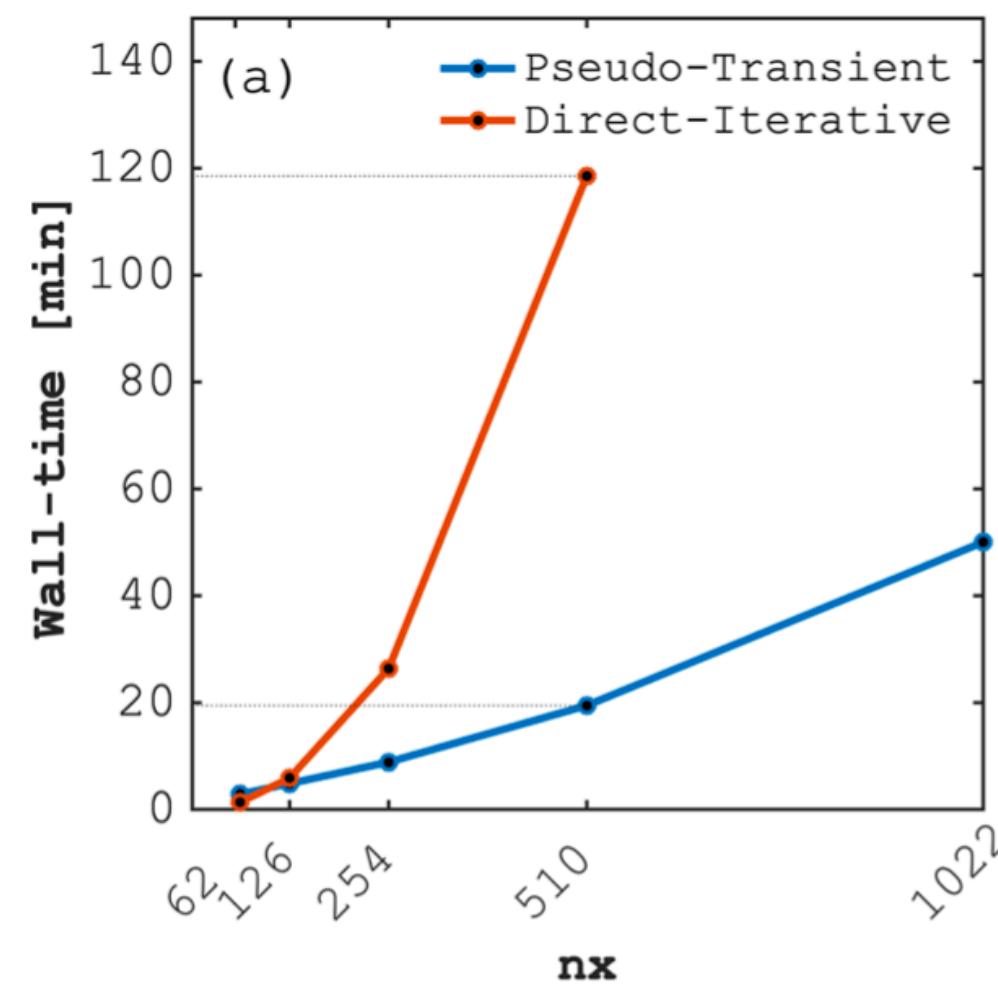
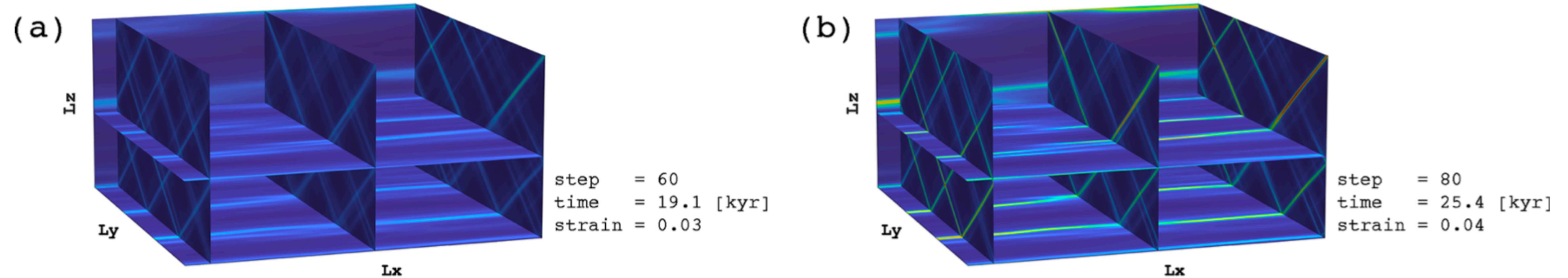
<sup>7</sup>Institute of Earth Sciences, University of Lausanne, Lausanne, Switzerland

<sup>8</sup>Swiss Geocomputing Centre, University of Lausanne, Lausanne, Switzerland

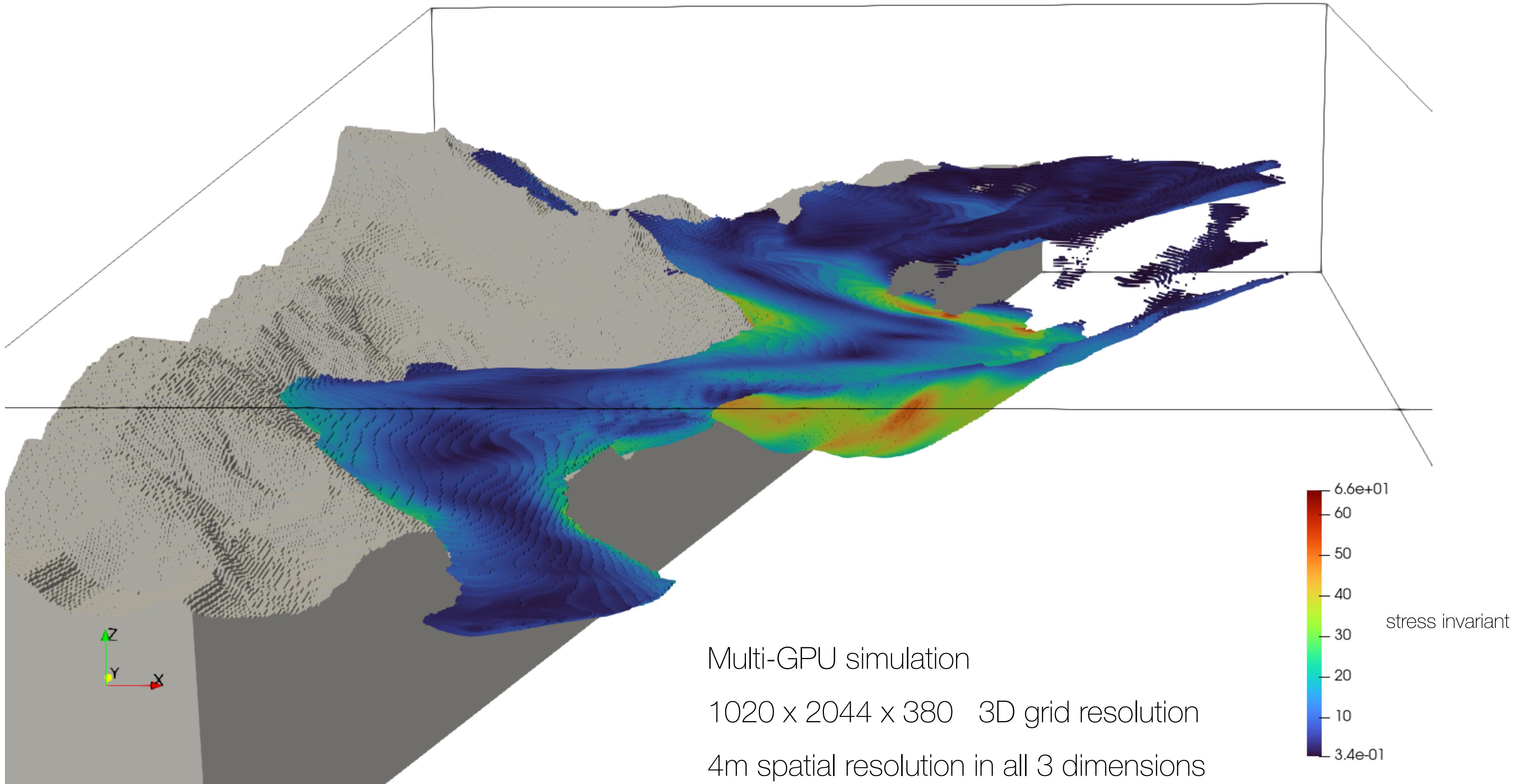
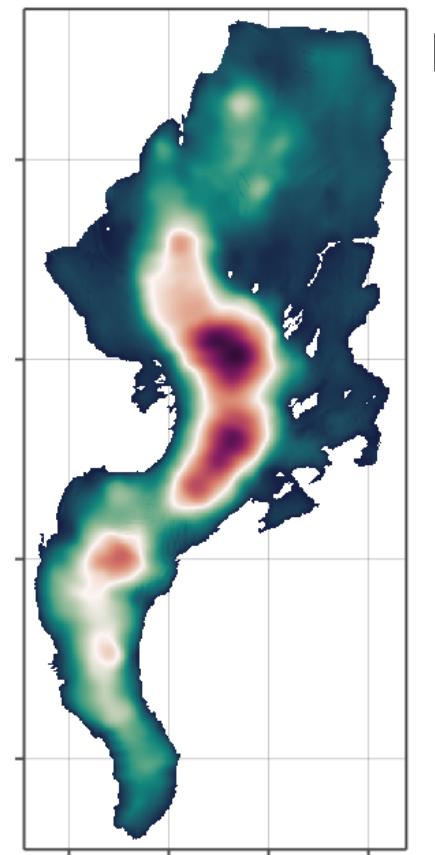
★These authors contributed equally to this work.

Correspondence: Ludovic Räss (luraess@ethz.ch)

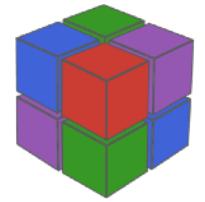
# Nonlinear mechanical elasto-viscoplastic solvers



# Thermomechanical ice flow solvers - Fastice

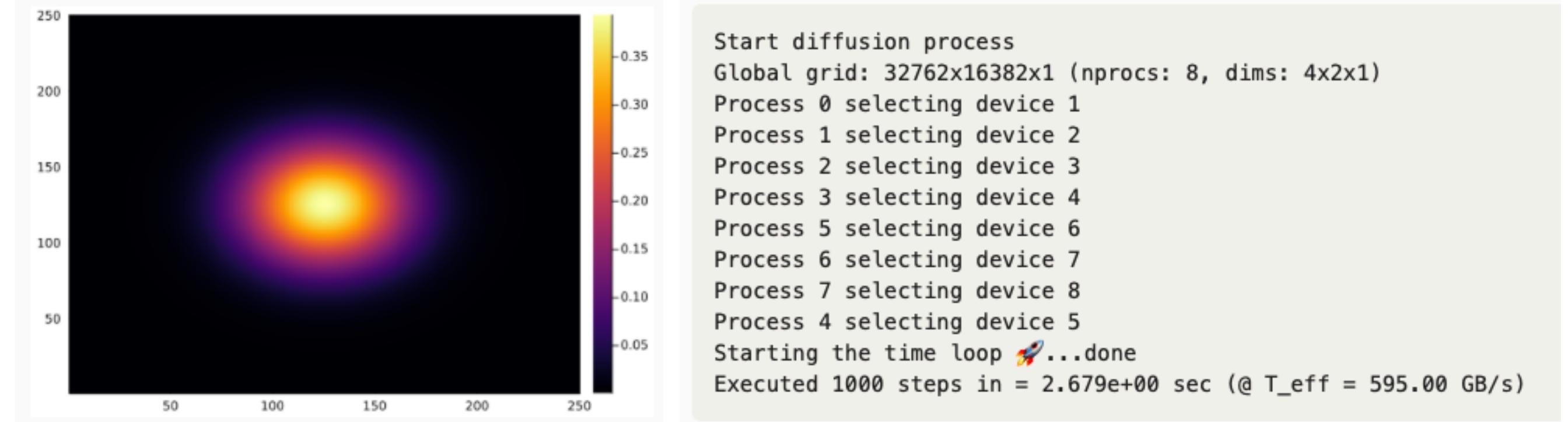


# AMD GPUs for exascale



ImplicitGlobalGrid.jl

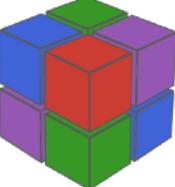
2D diffusion solver running on 8 AMD MI250x GPUs using ROCm-aware MPI on OLCF's Crusher test system, a small replica of Frontier, the first exascale supercomputer.



- Recently added AMDGPU.jl support to MPI.jl and ImplicitGlobalGrid.jl
- AMDGPU.jl support soon available in ParallelStencil.jl
- Enables ROCm-aware MPI: check out <https://github.com/luraess/ROCM-MPI>

# HPC packages we develop

 **ParallelStencil.jl** enables parallel stencil computations on GPUs (Nvidia & AMD) and CPUs.

 **ImplicitGlobalGrid.jl** renders stencil-based distributed parallelisation of GPU and CPU applications trivial.

**CellArrays.jl** arrays containing logical cells of small arrays or structs.

**GeoParams.jl** provides computational routines to define material parameters and perform non-dimensionalisation.

... and more, see:

<https://ptsolvers.github.io/GPU4GEO/software/>



**GPU4GEO**  
Frontier GPU multi-physics solvers

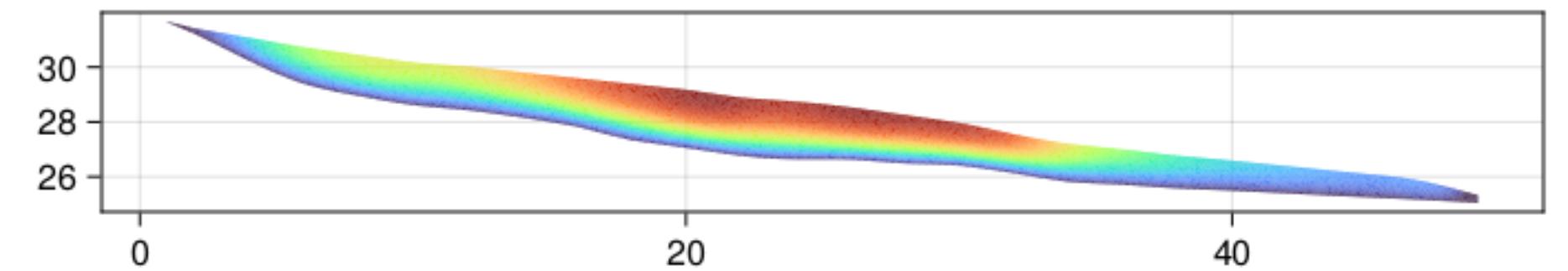
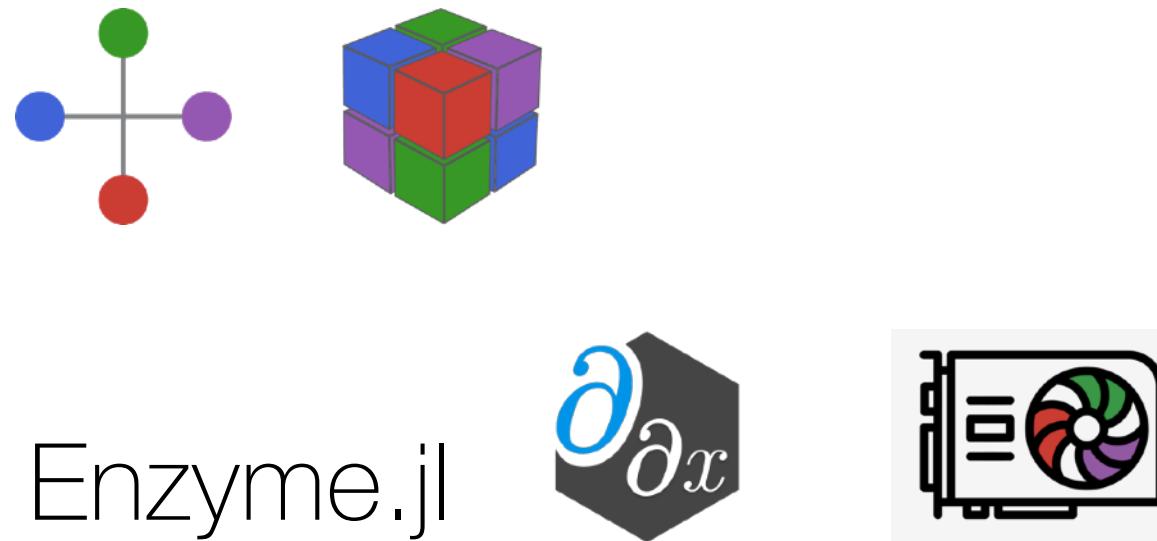


© 2022 GPU4GEO · Powered by [Franklin.jl](#) & [Coder](#).

# Outlook

GPU4GEO project runs from 2021 to 2024 and we'll further work towards flagship ice dynamics and geodynamic applications, and enhance **julia** support for AMD GPUs, ARM CPUs, and more:

- Further develop ParallelStencil and ImplicitGlobalGrid
- Work on scalable GPU HPC optimisation using AD with Enzyme.jl
- Explore (hybrid) FD - FEM matrix-free iterative GPU solvers



Stay tuned with related contributions at JuliaCon22:

<https://ptsolvers.github.io/GPU4GEO/posts/julia-juliacon22/>

contact luraess@ethz.ch