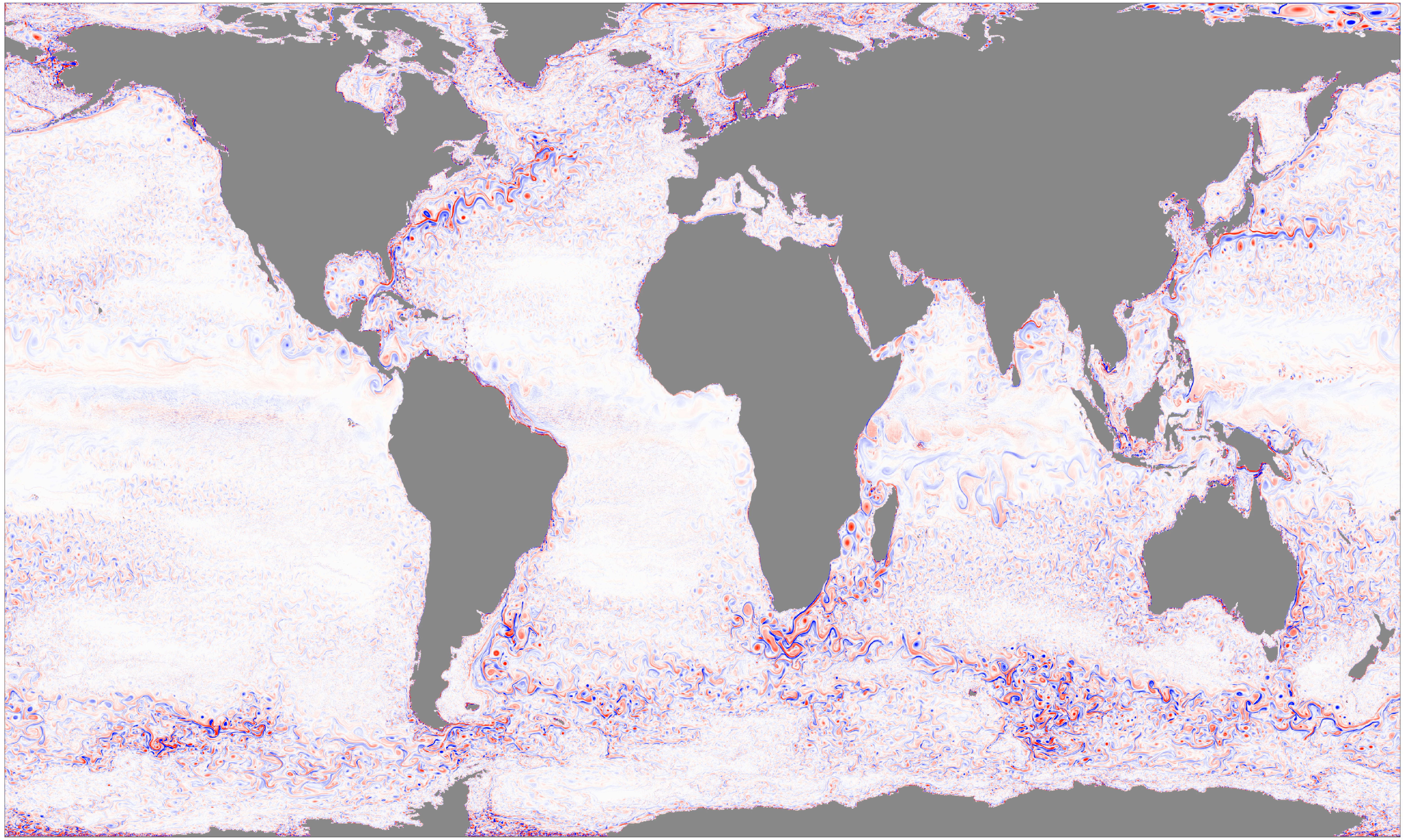
Oceananigans.jl
an ocean-flavoured fluid dynamics library

Navid Constantinou (@navidcy) & the CliMA Ocean Dev Team



near-global (75°S−75°N) ocean simulation at 1/12° horizontal resolution, 48 vertical levels
@ 68 Nvidia A100 achieving 10 simulated years per day

JuliaCon2024

requirements for a climate/ocean model

Computational efficiency + **Flexibility and ease of use**

- Necessary for global calibration
- Possibility of high-resolution

- Simulate physics from meters to global-scale
- Support rapid prototyping of parameterizations

"A fast model can be a good model,
but a good model must be a fast model!
Computational efficiency is crucial…"

[https://www.gfdl.noaa.gov/fv3/]

Oceananigans: Fast and Efficient 🚀

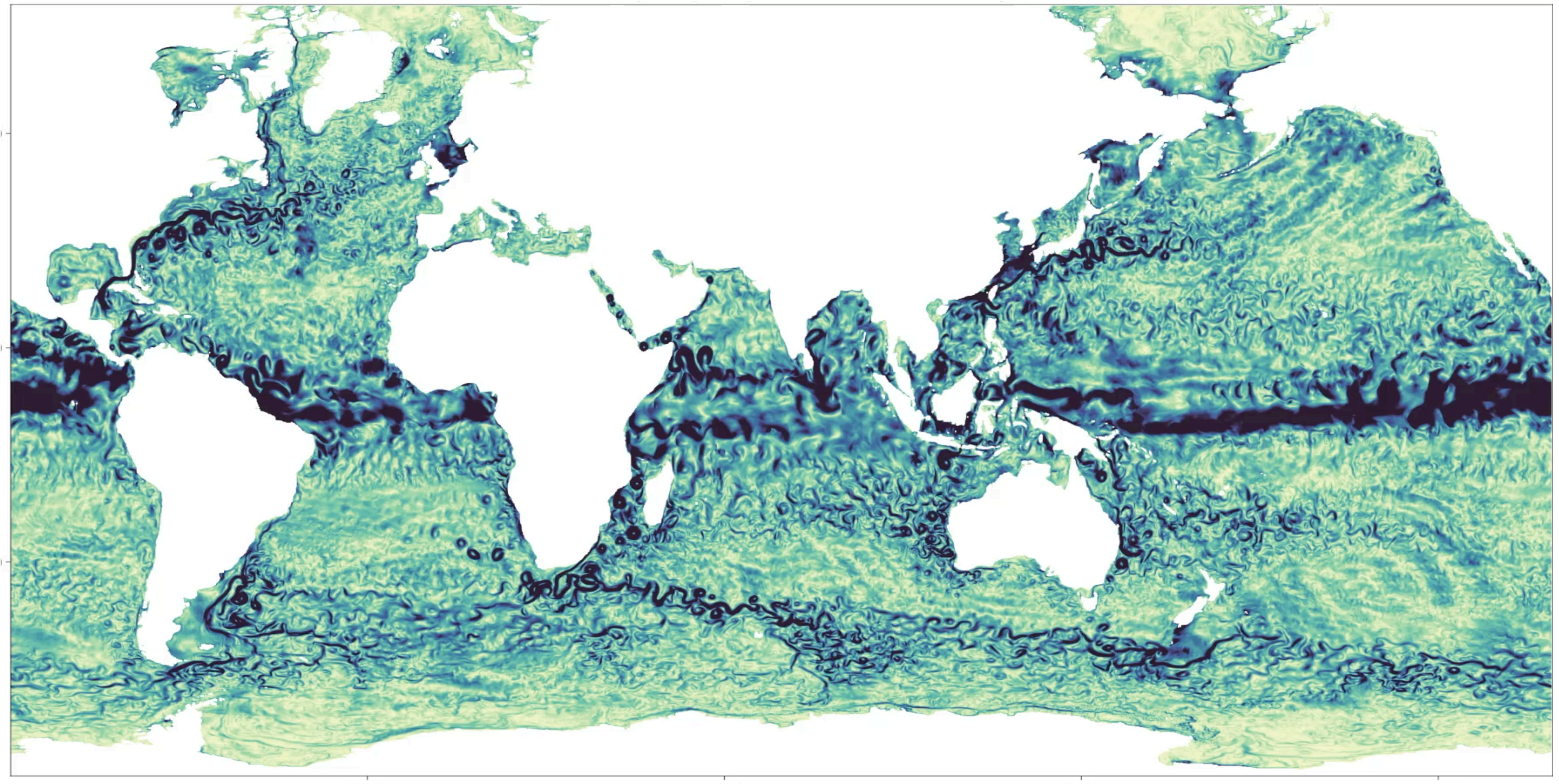
fast compute
written from scratch for GPUs

memory leanness
minimize temporary array creations

scalability
overlap communication & computation

10 Simulated years per day (SYPD):
threshold for climate projections

global ocean simulation forced with atmospheric JRA55 reanalysis



16 km horizontal resolution: **10** SYPD on **8** GPUs

8 km horizontal resolution: **10** SYPD on **64** GPUs

2 km horizontal resolution: **>1** SYPD on **512** GPUs

Oceananigans: Easy to use and Accessible

all written in julia

Faster than interpreted languages (Python, Matlab)

More flexible than compiled languages (C, Fortran)

Easy portability to virtually any architecture/systems

```julia
using Oceananigans
using GLMakie

grid = RectilinearGrid(CPU(),
                       size = (64, 64),
                       x = (-5, 5),
                       y = (-5, 5),
                       topology = (Bounded, Bounded, Flat))

model = NonhydrostaticModel(grid=grid, tracers=:c, advection=WENO())

gaussian(x, y) = exp(-(x^2 + y^2))
set!(model, c=gaussian)

c = model.tracers.c

∇c² = ∂x(c)^2 + ∂y(c)^2
∇c² = Field(∇c²)
compute!(∇c²)
```

try changing `CPU()` *to* `GPU()`

initial conditions

diagnostics

Starred 923

used in *more* than 20 scientific papers

55+ contributors to the codebase

> *"...I have never experienced getting a useful calculation done as easily as I was able to do with Oceananigans. It not only has a sophisticated interface, but it is remarkably fast...".*
>
> *Linux magazine*

User interface:

- Designed so code "reads like a paper"
- Should not require comments

Ramadhan et al, JOSS, 2020

Oceananigans: Flexible

Discrete calculus on a staggered C-grid

Local operations

Global operations

Finite volume engine

- Grids
- Fields
- Operators (low and high order)
- Solvers (FFT, matrix solvers…)
- Parallelization

Utilities for numerical experiments

- OutputWriters
- Diagnostics → e.g., `∂x(u)^2`
- Callbacks

User-facing Utilities

Oceananigans: Flexible

Discrete calculus
on a staggered
C-grid

Local operations

Global operations

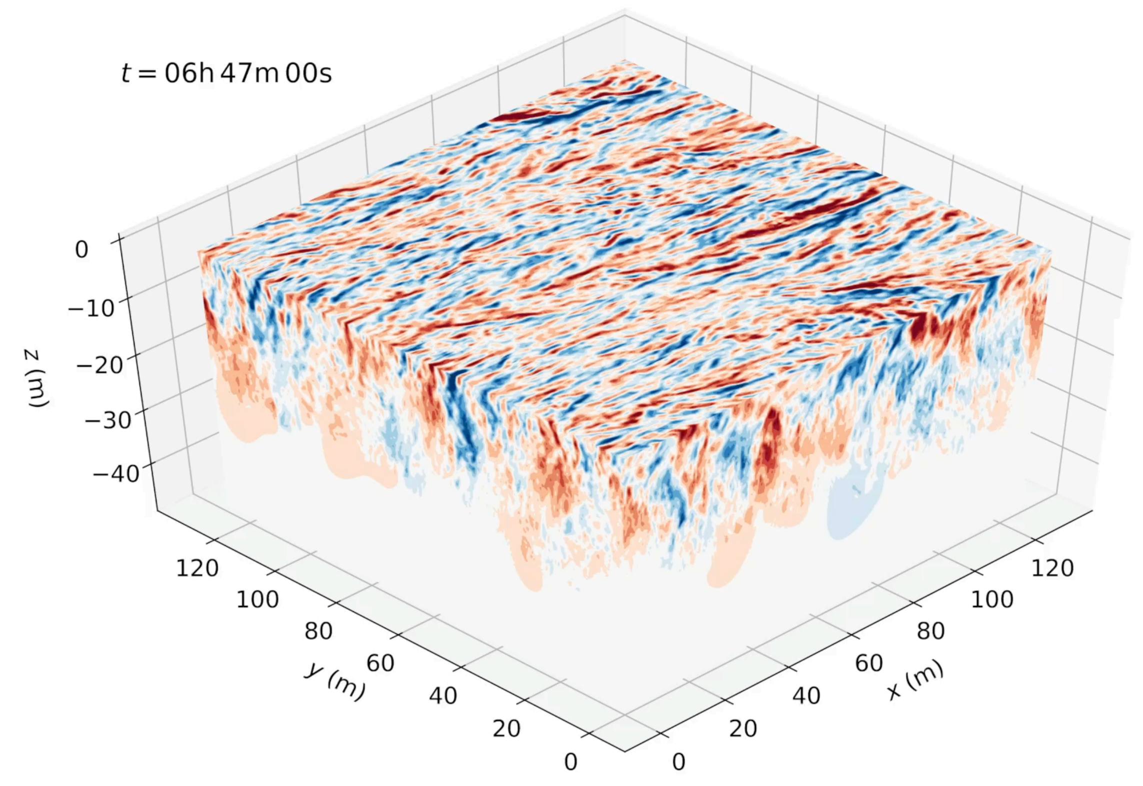
Finite volume engine

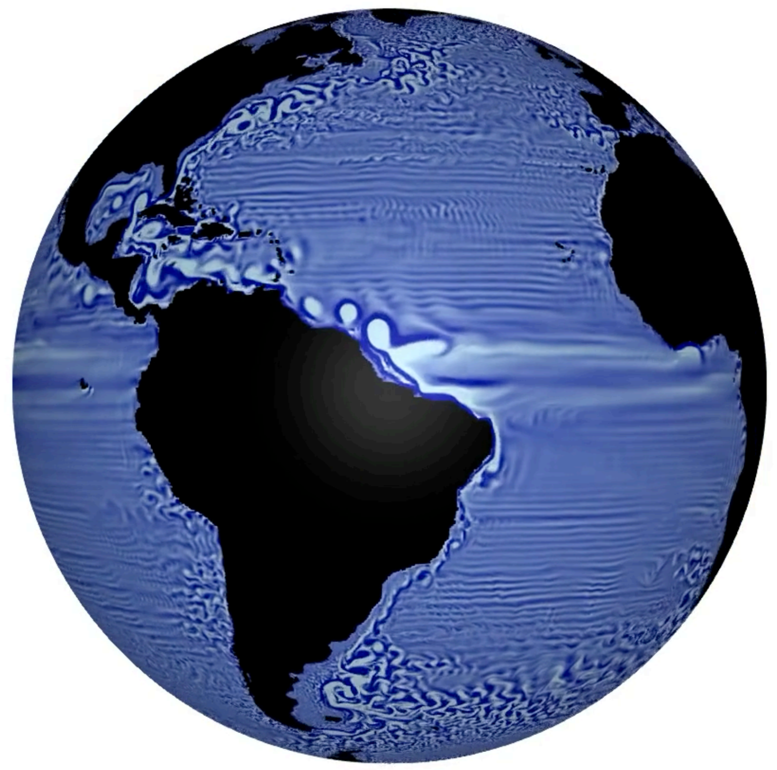
- Grids
- Fields
- Operators (low and high order)
- Solvers (FFT, matrix solvers…)
- Parallelization

Utilities for numerical experiments

- OutputWriters
- Diagnostics ⟶ e.g., ∂x(u)^2
- Callbacks

User-facing
Utilities





Domain-Specific numerics and physics

- ***NonhydrostaticModel,***
- ***HydrostaticFreeSurfaceModel,***
- ***ShallowWaterModel***
- Coriolis, Equation of State, Parameterizations…
- Pressure / free surface solvers…
- Time stepping schemes

Physics Modules
Implemented in
Oceananigans

Injecting code in a simulation: forcing with a neural net

```julia
using Oceananigans,
      Oceananigans.Units
using Lux

grid = LatitudeLongitudeGrid(GPU(); kw...)

u_sgs = Field(grid)

model = HydrostaticFreeSurfaceModel(; grid, forcing = (; u = u_sgs), kw...)

simulation = Simulation(model; Δt = 10minutes, stop_time = 10days)

NN = Chain(args...) |> gpu    # A neural network that computes u -> u_sgs

function neural_network_inference(simulation)
    u_sgs = simulation.model.forcing.u
        u = simulation.model.velocities.u

    u_sgs .= NN(u)
end

simulation.callbacks[:apply_nn] = Callback(neural_network_inference,
                                           IterationInterval(1))

run!(simulation)
```

Simple and effective way to add a NN in Oceananigans thanks to:

GPU

- Inject the function `neural_network_inference` in the time-stepping loop
- A callback has access to all the variables of the simulation
- Each iteration `u_sgs` is used as forcing and then recalculated

http://clima.caltech.edu

thanks

github.com/CliMA/Oceananigans.jl