Composability, Flexibility, and Ease of use for CliMA's Next Generation Earth System Model

Valeria Barra
and the entire CliMA team...

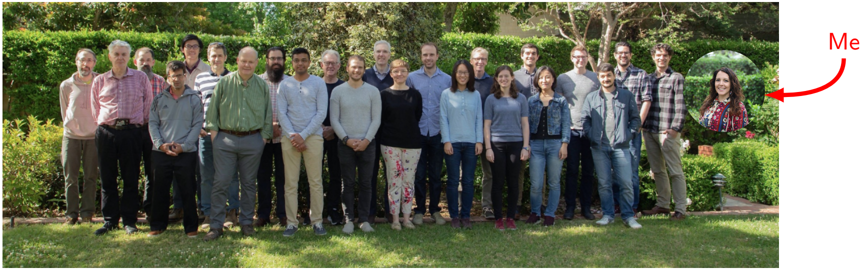
Department of Environmental Science and Engineering,
California Institute of Technology

February 25th, 2022

Outline

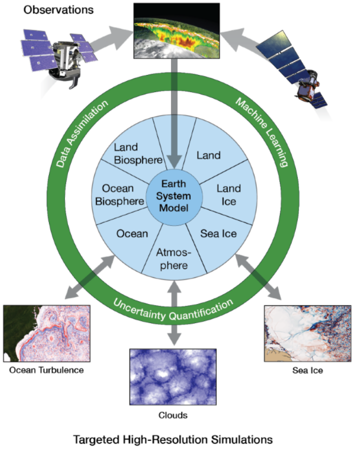
About CliMA

The Climate Modeling Alliance (CliMA) is a coalition of scientists, engineers, and applied mathematicians from Caltech, MIT, the Naval Postgraduate School, and the NASA Jet Propulsion Laboratory. We are building the first Earth System Model (ESM) in the Julia programming language that automatically learns from diverse data sources to produce more accurate climate predictions with quantified uncertainties.



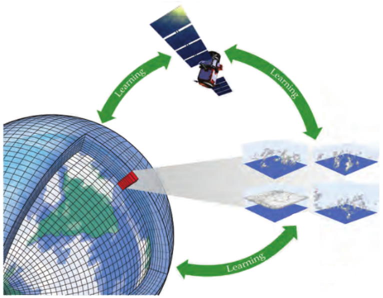
Caltech

Goals



[Source: courtesy of Tapio Schneider (Caltech)]

- The Earth System Model (ESM) will be grounded in physics (using sub-grid scale, cloud-resolving modeling) and designed for automated calibration of parameters using machine learning.
- High-resolution Large-Eddy Simulations (LES) are used to inform parametrizations of the global circulation model (GCM), which in turn, can be used for large-scale forcings to force the LES.



[Source: Physics Today - June 2021, pg. 44-51]

Context/Motivation

Technical aims:

- Support CPUs and GPUs using a common open-source code base written in the high-level, dynamic Julia programming language.
- Support both Large-Eddy Simulation (LES) and General Circulation Model (GCM) configurations (i.e., Cartesian and spherical domains).
- Be accessible and extensible by a mixture of users.
- Various options evaluated:
    - First codebase in Julia to demonstrate feasibility for GPUs: ClimateMachine.jl (https://github.com/CliMA/ClimateMachine.jl).
- Overall decision has worked well: embraced by researchers, overcome initial skepticism by funders.

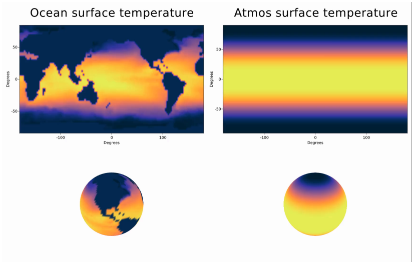
ClimateMachine.jl: a first codebase

- Supports only Discontinuous Galerkin (DG) discretization. Same in each direction (horizontal/vertical) but allows different polynomial order. No staggered grids supported.
- Can prescribe PDEs only in conservation form $\partial_t \boldsymbol{Q} + \nabla \cdot \boldsymbol{F}(\boldsymbol{Q}) = S(\boldsymbol{Q})$.
- Operator volume/face kernels written in KernelAbstractions.jl (a unified programming model, similar to OpenCL/SYCL, which allows for single-source code for CPUs & GPUs, but primarily "*a GPU code which runs on CPUs*").
- Overlaps computation & communication:
  - Distributed via MPI.jl.
  - Exchange boundary faces during volume & internal face integrals.
- Efficient, but somewhat inflexible. For scaling studies, see:

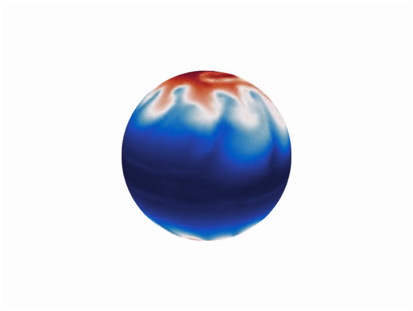
A. Sridhar et al., *Large-eddy simulations with ClimateMachine v0.2.0: a new open-source code for atmospheric simulations on GPUs and CPUs*, in review for Geoscientific Model Development [Preprint]

Caltech

ClimateMachine.jl: a first codebase (cont'ed)



Ocean and atmosphere surface temperature. The atmosphere's dry Held-Suarez test case is excited by a baroclinic instability initial condition. The ocean goes through a few seasons.



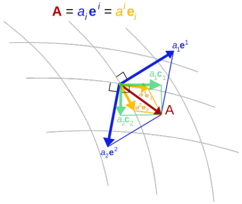
Dry Held-Suarez test case for the atmosphere. A forcing that mimics radiative forcing is applied such that a random initial condition evolves towards a statistically steady state. Simulation done w/ entropy-stable DG method in ClimateMachine.jl.

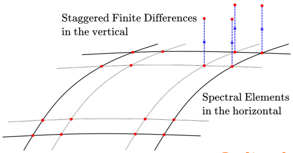
[Source: Courtesy of Andre Souza from the Oceananigans.jl team (MIT). Visualizations done w/ Makie.jl]

ClimaCore.jl

ClimaCore.jl — the new dycore.
A library (suite of tools) for constructing flexible space discretizations.

- Geometry:
  - Support for different geometries (Cartesian & spherical).
  - Supports covariant vector representation for curvilinear, non-orthogonal systems and Cartesian vectors for Euclidean spaces.
- Space Discretizations:
  - Horizontal: Support both Continuous Galerkin (CG) and Discontinuous Galerkin (DG).
  - Vertical: staggered Finite Differences (FD).



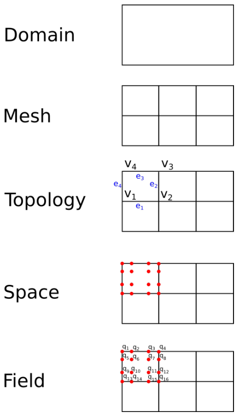

Staggered Finite Differences in the vertical
Spectral Elements in the horizontal

ClimaCore.jl: API

- API objects:
    - Domain, Mesh, Topology, Space, Field.
- `Field` abstraction:
    - Scalar, Vector or Struct-valued.
    - Stores values, geometry, and mesh info.
    - Flexible memory layouts.
    - Useful overloads: `sum` (integral), `norm`, `mean`.
    - Compatible with DifferentialEquations.jl time integrators.

```julia
domain = Domains.SphereDomain(radius)
mesh = Meshes.EquiangularCubedSphere(domain, Ne)
grid_topology = Topologies.Topology2D(mesh)
quad = Spaces.Quadratures.GLL{Nq}()
space = Spaces.SpectralElementSpace2D(grid_topology, quad)
```

```
julia> sum(ones(space))
4.618802153517008
```



ClimaCore.jl: API (cont'ed)

Flexible data layout:

- Support for different memory layouts: Array-of-Structs (AoS), Struct-of-Arrays (SoA), Array-of-Struct-of-Arrays (AoSoA).
- Common interface: `slab` for extracting 2D horizontal field slices; `column` for 1D vertically-aligned nodes.
- Add element node size dimensions to type domain (i.e., specialize on polynomial degree, useful for loop unrolling; important for kernel performance).
- Flexible memory layouts allow for flexible threading models:
  - CPU thread over elements.
  - GPU thread over nodes/node columns (upcoming).

ClimaCore.jl: API (cont'ed)

ClimaCore.jl's composable `Operators` and Julia broadcasting:

- Julia broadcasting:
  - apply a vectorized function point-wise to an array. Scalar values are "broadcast" over arrays; Fusion of multiple operations.
  - User-extensible API: can be specialized for custom functions or argument types (e.g., `CuArray` compiles and applies a custom CUDA kernel).
- `Operators` (`grad`, `div`, `interpolate`) are "pseudo-functions": act like functions when broadcasted over a `Field`, but can't be called on a single value; can be composed and fused w/ function calls. Matrix-free, i.e., no assembly; specify action of operator.

```julia
# apply f to each element of X
f.(X)

# fuse multiple operations
# and assign to existing array
# without intermediate temporaries
Y .= X0 .+ ϵ .* f.(X)

# expression internally calls
materialize!(Y,
  broadcasted(+, X0,
    broadcasted(*, ϵ,
      broadcasted(f, X))))
```

```julia
grad = Operators.Gradient()
wdiv = Operators.WeakDivergence()
diff = @. -wdiv(grad(u))
```

Examples: Shallow-water equations

The shallow water equations (in *vector invariant form*):

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\boldsymbol{u}) = 0 \tag{1a}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla(\Phi + \tfrac{1}{2}\|\boldsymbol{u}\|^2) = (\boldsymbol{u} \times (f + \nabla \times \boldsymbol{u})) \tag{1b}$$

where f is the Coriolis term and $\Phi = g(h + h_s)$.

Written in terms of a curvilinear, non-orthogonal basis:

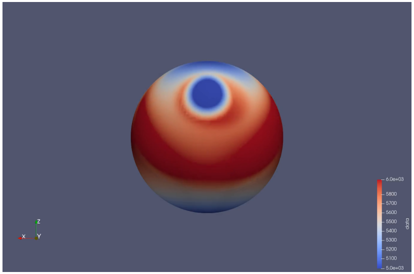
$$\frac{\partial h}{\partial t} + \frac{1}{J}\frac{\partial}{\partial \xi^j}\left(hJu^j\right) = 0 \tag{2a}$$

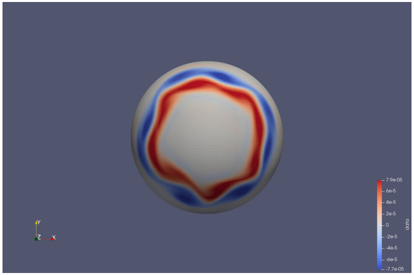
$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial \xi^i}(\Phi + \tfrac{1}{2}\|\boldsymbol{u}\|^2) = E_{ijk}u^j(f^k + \omega^k) \tag{2b}$$

```
@. begin
    dYdt.h += -wdiv(y.h * y.u)
    dYdt.u +=
        -grad(g * (y.h + h_s) + norm(y.u)^2 / 2) + y.u × (f + curl(y.u))
end
```

Shallow-water equation Test Cases

ClimaCore.jl/examples/sphere/shallow_water.jl





Shallow-water equations suite, Test Case 5 [**Williamson et al 1992**]. Zonal flow over an isolated mountain.

Shallow-water equations suite, barotropic instability test case [**Galewsky et al 2004**]. Zonal jet with compact support at mid-latitude. A small height disturbance is then added, which causes the jet to become unstable and collapse into a highly vortical structure.

Examples: Flux limiters for advection (transport) problems

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \boldsymbol{u}, \tag{3a}$$

$$\frac{\partial Q}{\partial t} = -\nabla \cdot Q \boldsymbol{u}, \tag{3b}$$

Transport of a passive tracer, with $Q = \rho q$, where q denotes tracer concentration (i.e., mixing ratio or mass of tracer per mass of dry air, in dry problems, or tracer mass per mass of moist air, in moist problems) per unit mass, and ρ fluid density.

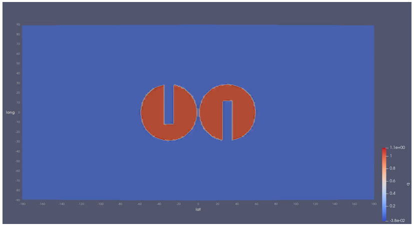
```
@. ystar.ρ = -wdiv(y.ρ * u)          # contintuity equation
@. ystar.ρq += -wdiv(y.ρq * u)       # adevtion of tracers equation
```

Traditional SEM results are quite oscillatory for advection-dominated problems. Hence, we want to apply a local element reconstruction for the mimetic SEM formulation which yields an efficient quasimonotone (i.e., monotone w.r.t. the spectral element nodal values) limiter on q. This involves solving a constrained optimization problem (a weighted least square problem) that is local to each element [**Guba et al 2014**].

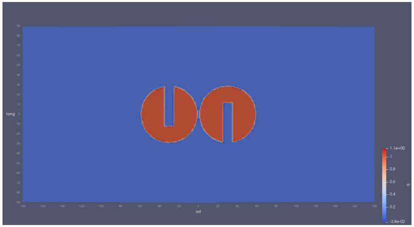
Flux limiter Test Cases

ClimaCore.jl/examples/sphere/opt_limiters_solidbody.jl

$p = 3$, $ne = 40 \times 40 \times 6$ (effective resolution 0.75° at equator.)



No limiter.

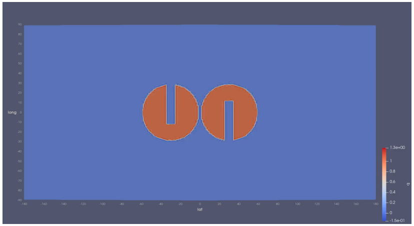


With limiter.

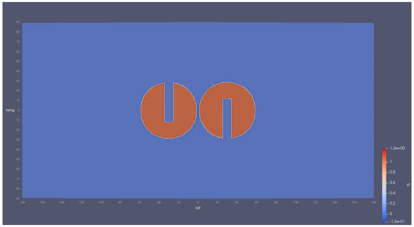
Flux limiter Test Cases (cont'ed)

ClimaCore.jl/examples/sphere/opt_limiters_solidbody.jl

$p = 6$, $ne = 20 \times 20 \times 6$ (effective resolution 0.75° at equator.)



No limiter.



With limiter.

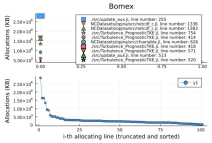
Caltech

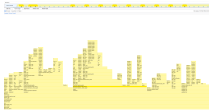
Performance/optimization efforts

Three current streams of work to reduce time-to-solution:

- Mathematical/numerical: Use IMEX time integrators for a less stringent CFL condition, otherwise exacerbated by elements aspect ratio $\sim 1 : 10^4$.

- Julia Optimizations:
  - Disable bounds checking to facilitate vectorized (SIMD) instructions via `@inbounds`.
  - Ensure type stability using tools, e.g., JET.jl that allows to do static type checking.
  - Eliminate dynamic memory allocations.





[Courtesy of C. Kawczynski]

- Parallelization (ongoing):
  - ClimaComms.jl wrapper library that supports generic distributed computing paradigms. Currently using MPI.jl in the backend. May use NVIDIA Collective Communications Library (NCCL) in the future.

Conclusions and Outlook

- Conclusions:
  - ClimaCore.jl is the new open-source dycore for CliMA's proposed new Earth System Model (ESM), entirely written in the Julia dynamic language.
  - We introduced ClimaCore.jl's API for flexible discretizations and composable solvers.
  - We showed examples of applications for atmospheric flows and flux limiters to overcome high-order SEM oscillation challenges in advection-dominated problems.

Caltech

Conclusions and Outlook

- Conclusions:
  - ClimaCore.jl is the new open-source dycore for CliMA's proposed new Earth System Model (ESM), entirely written in the Julia dynamic language.
  - We introduced ClimaCore.jl's API for flexible discretizations and composable solvers.
  - We showed examples of applications for atmospheric flows and flux limiters to overcome high-order SEM oscillation challenges in advection-dominated problems.
- Future Work:
  - Extend flux limiters to 3D dry and moist problems
  - Add support for high-order vertical FD stencils
  - More distributed functionality and scaling

Caltech

Conclusions and Outlook

- Conclusions:
  - ClimaCore.jl is the new open-source dycore for CliMA's proposed new Earth System Model (ESM), entirely written in the Julia dynamic language.
  - We introduced ClimaCore.jl's API for flexible discretizations and composable solvers.
  - We showed examples of applications for atmospheric flows and flux limiters to overcome high-order SEM oscillation challenges in advection-dominated problems.
- Future Work:
  - Extend flux limiters to 3D dry and moist problems
  - Add support for high-order vertical FD stencils
  - More distributed functionality and scaling

🚀

Caltech

Acknowledgements

Thank you to all ClimateMachine.jl's and ClimaCore.jl's users and contributors. In particular, for the latest CliMA dycore efforts a special mention goes to:

- Tapio Schneider[1] (PI), Paul Ullrich[2], Oswald Knoth[3], Simon Byrne[1], Jake Bolewski[1], Charles Kawczynski[1], Sriharsha Kandala[1], Zhaoyi Shen[1], Jia He[1], Kiran Pamnany[1], Ben Mackay[1], Akshay Sridhar[1], Dennis Yatunin[1], Lenka Novak[1], Toby Bischoff[1], Daniel (Zhengyu)Huang[1], Andre Souza[4], Yair Cohen[1]

  1: Caltech, 2: UC Davis, 3: TROPOS, 4: MIT

- Our funders:

SCHMIDT FUTURES NSF DARPA HEISING-SIMONS FOUNDATION Audi Environmental Foundation CHARLES TRIMBLE

Thank you!

Caltech