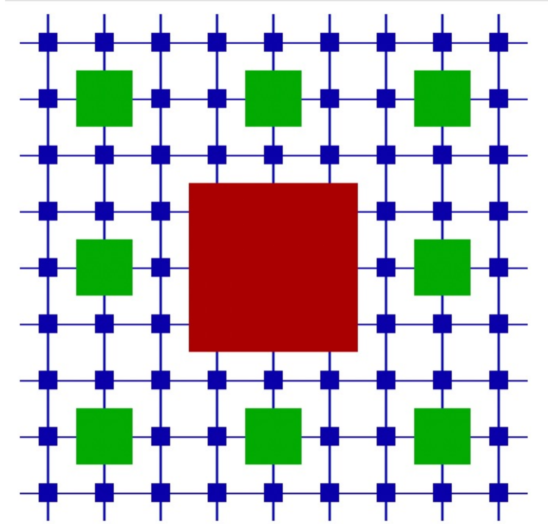


CarpetX: faster, more accurate, safer

Erik Schnetter, Perimeter Institute for Theoretical Physics

2024 European Einstein Toolkit Meeting

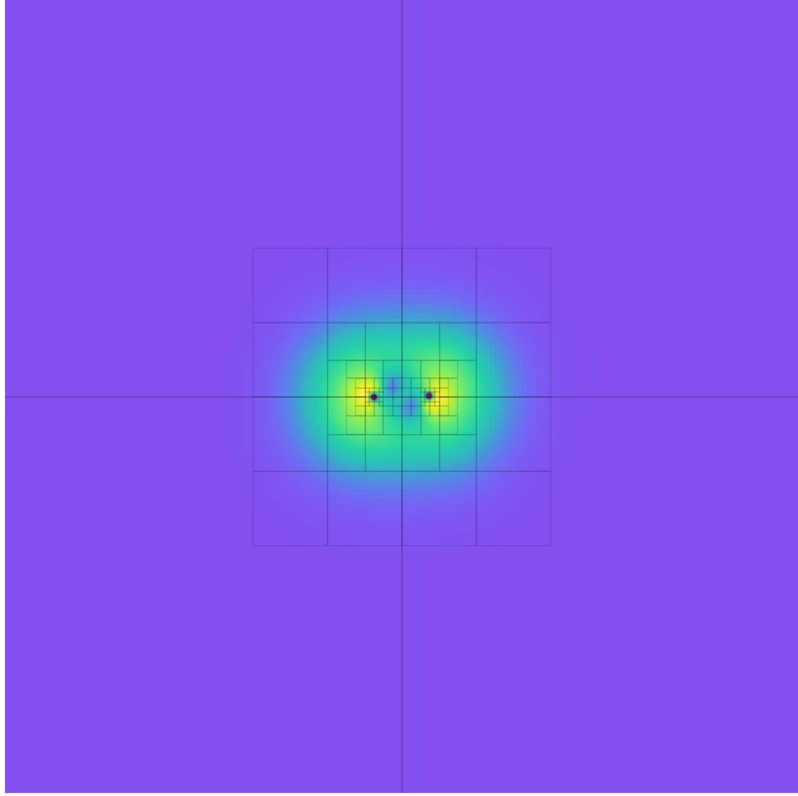
2024-07-09

Perimeter Institute for Theoretical Physics

CarpetX, a Driver for the Einstein Toolkit

1. Exascale computing:
   - Highly efficient and parallel (many nodes, many cores)
   - Supports CPUs, GPUs, other accelerators
2. Modern discretization methods:
   - Adaptive mesh refinement (AMR)
   - Conservative discretizations, constraint-preserving discretizations
   - (multi-patch grids – soon!)
3. Offers safe programming model:
   - Catches undefined values, catches writes to read-only values in grid functions

[Adam Peterson, Don Wilcox,
https://amrex-codes.github.io/amrex/gallery.html]



Exascale computing

Exascale computing

- The Einstein Toolkit runs on many architectures, from a small laptop to the largest supercomputers
- Modern computer have **heterogenous architectures** – computing power is provided not just by a single CPU
- Programming for a single CPU is easy, but the code will run very slowly, compared to the hardware capabilities

Speed

- 1 CPU core: 0.01 TFlop/sec (single core, optimistic assumptions)
  - Straight-forward serial code
- 1 CPU node: 3 TFlop/sec (40 cores, SIMD code, optimistic)
  - Best parallel CPU code (e.g. OpenMP)
- 1 GPU: 10 TFlop/sec (Nvidia A100, theoretically)
  - Best GPU code (e.g. CUDA, ROCm)
- Frontier: 1,200,000 TFlop/sec (38,000 AMD MI250X GPUs)
  - Largest public DOE system

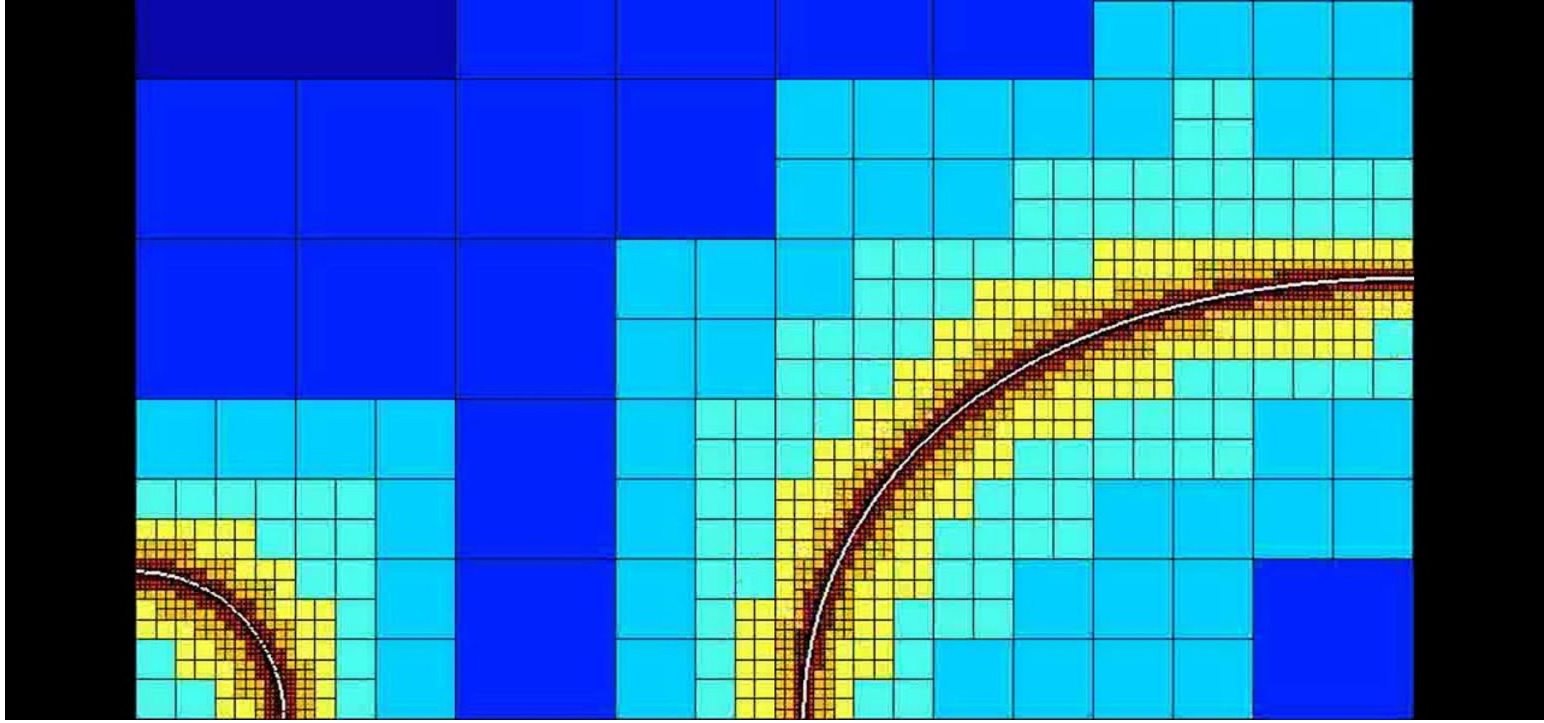
Programming approach:

1. Start with a serial code. Make it correct, keep it simple, test it well.
2. Measure performance, see what it slow and why
3. Add more and more parallelism, until the code is fast enough
4. If necessary, re-design the algorithm
5. If stuck, consult with an expert, show the working-but-slow code

- (Don't start with step 4. Many people do. Don't skip step 2 either.)

- There are many kinds of parallelism: SIMD, multi-threading (OpenMP), GPU programming (CUDA, ROCm, oneAPI), distributed computing (MPI).
- The Einstein Toolkit helps with all of these!

https://www.youtube.com/watch?v=5NvYsI4szwY

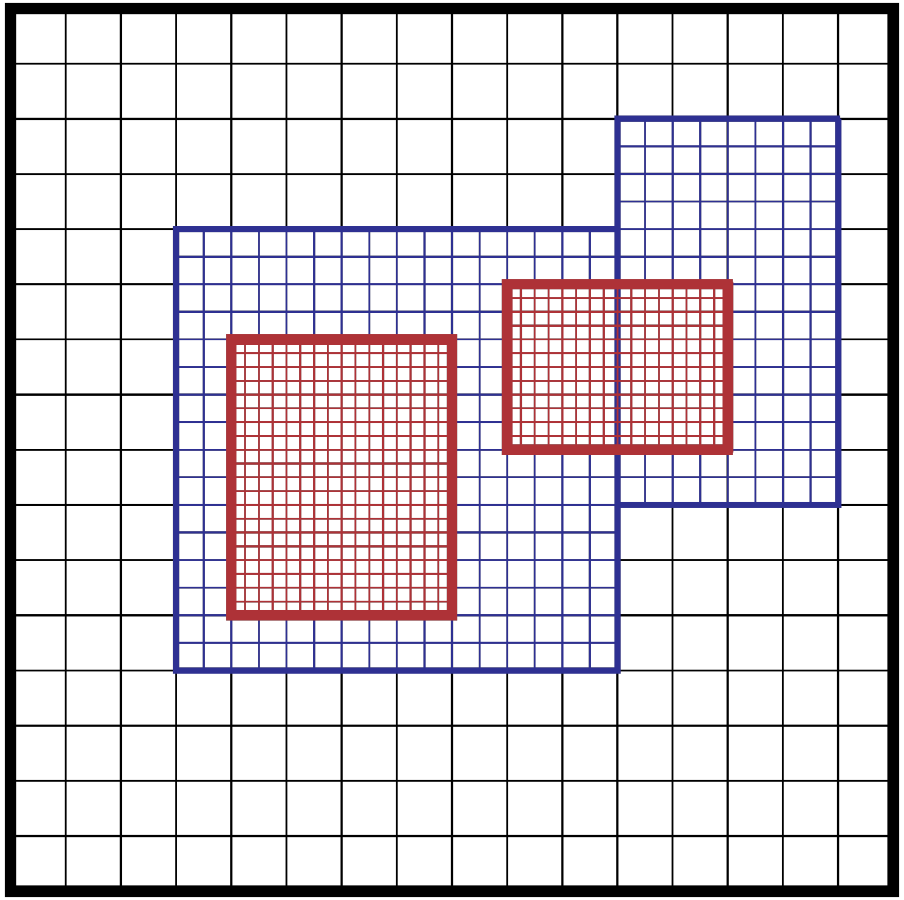


Modern discretization methods

Modern discretization methods supported by CarpetX

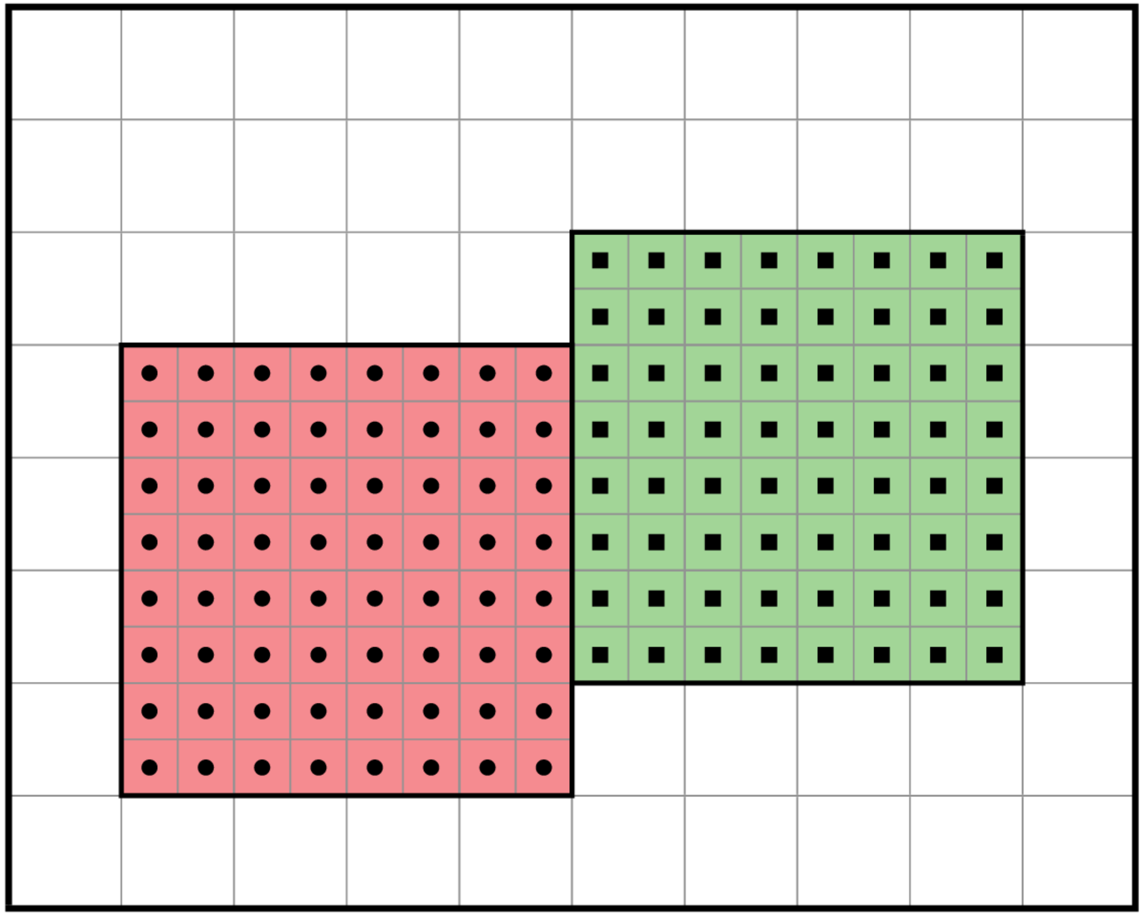
- Adaptive Mesh Refinement (AMR)
- (Multi-patch methods)
- Conservative discretizations, constrained-preserving discretizations, staggered grids
- Higher-order time integration for coupled multi-physics systems (*method of lines*)

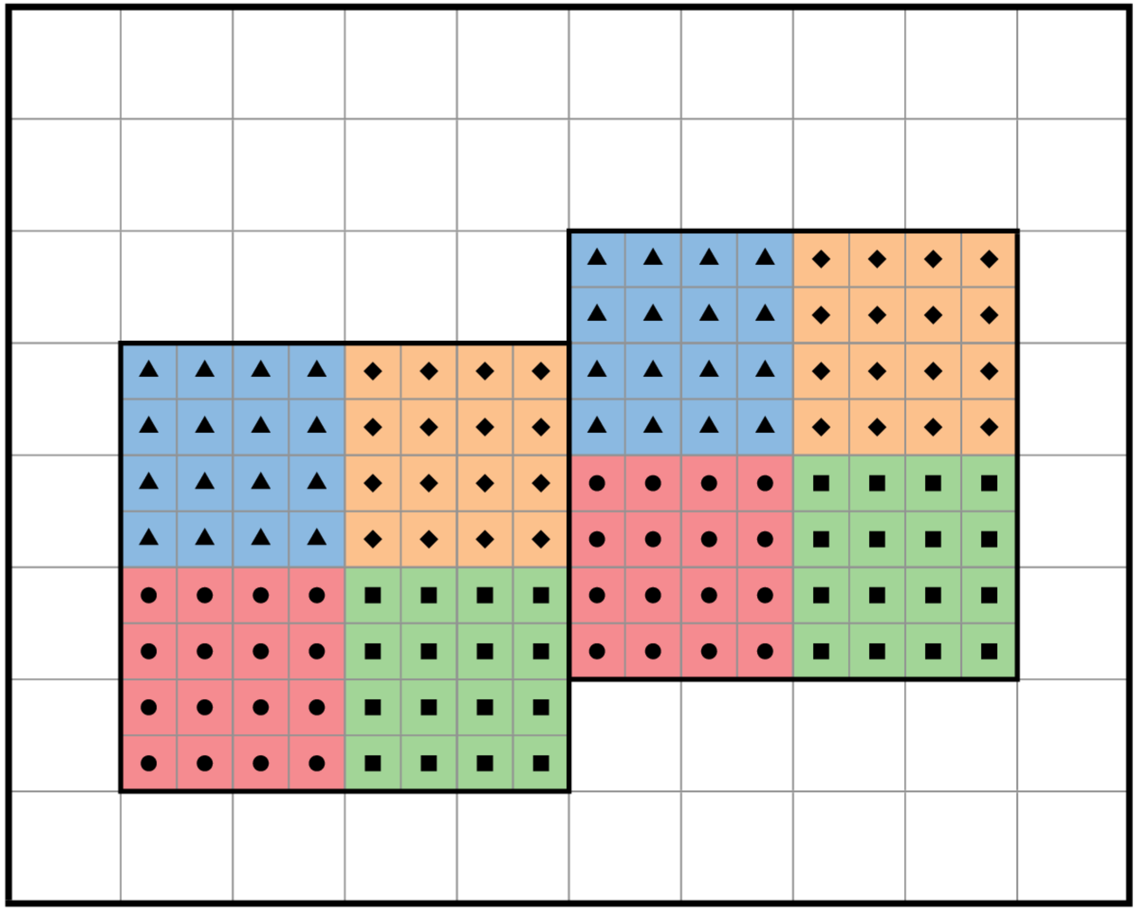
Adaptive Mesh Refinement (AMR)



https://amrex-codes.github.io/amrex/docs_html/

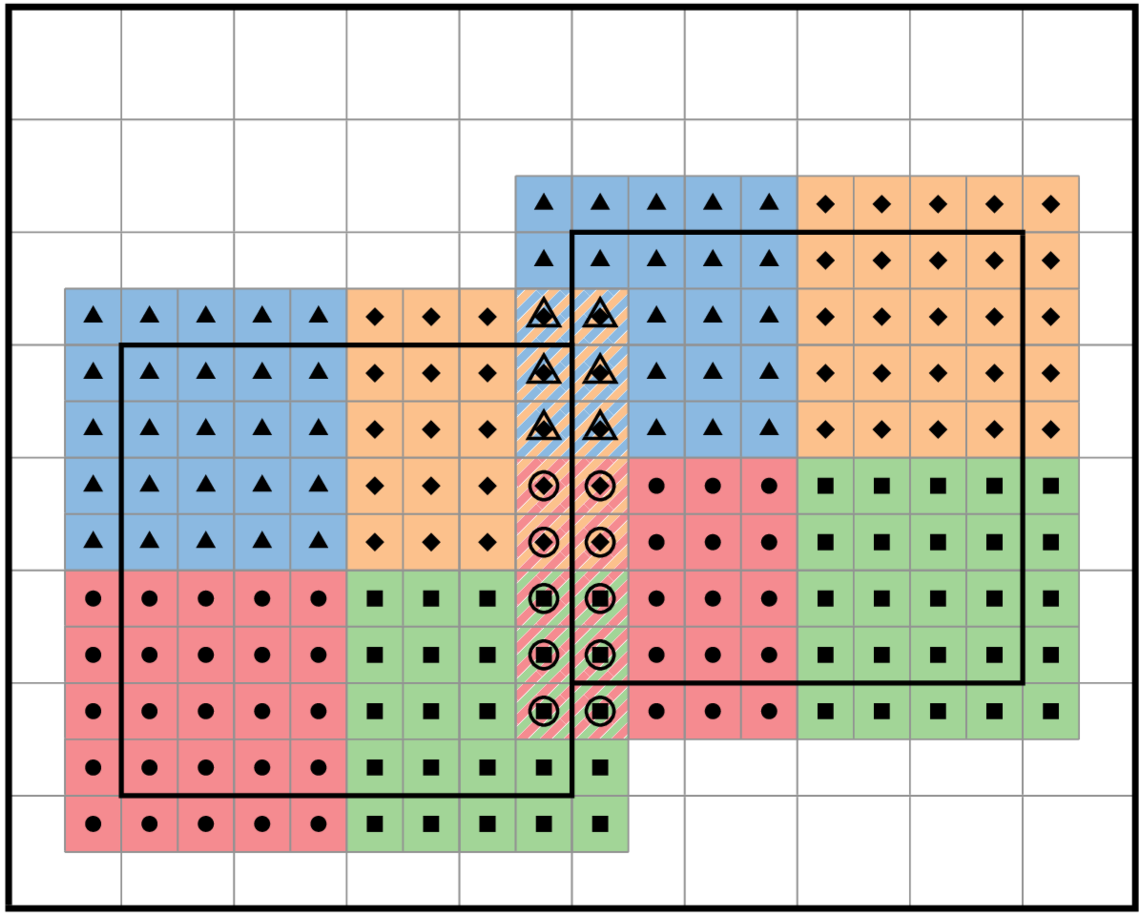
Multi-Threading (grid function tiling)





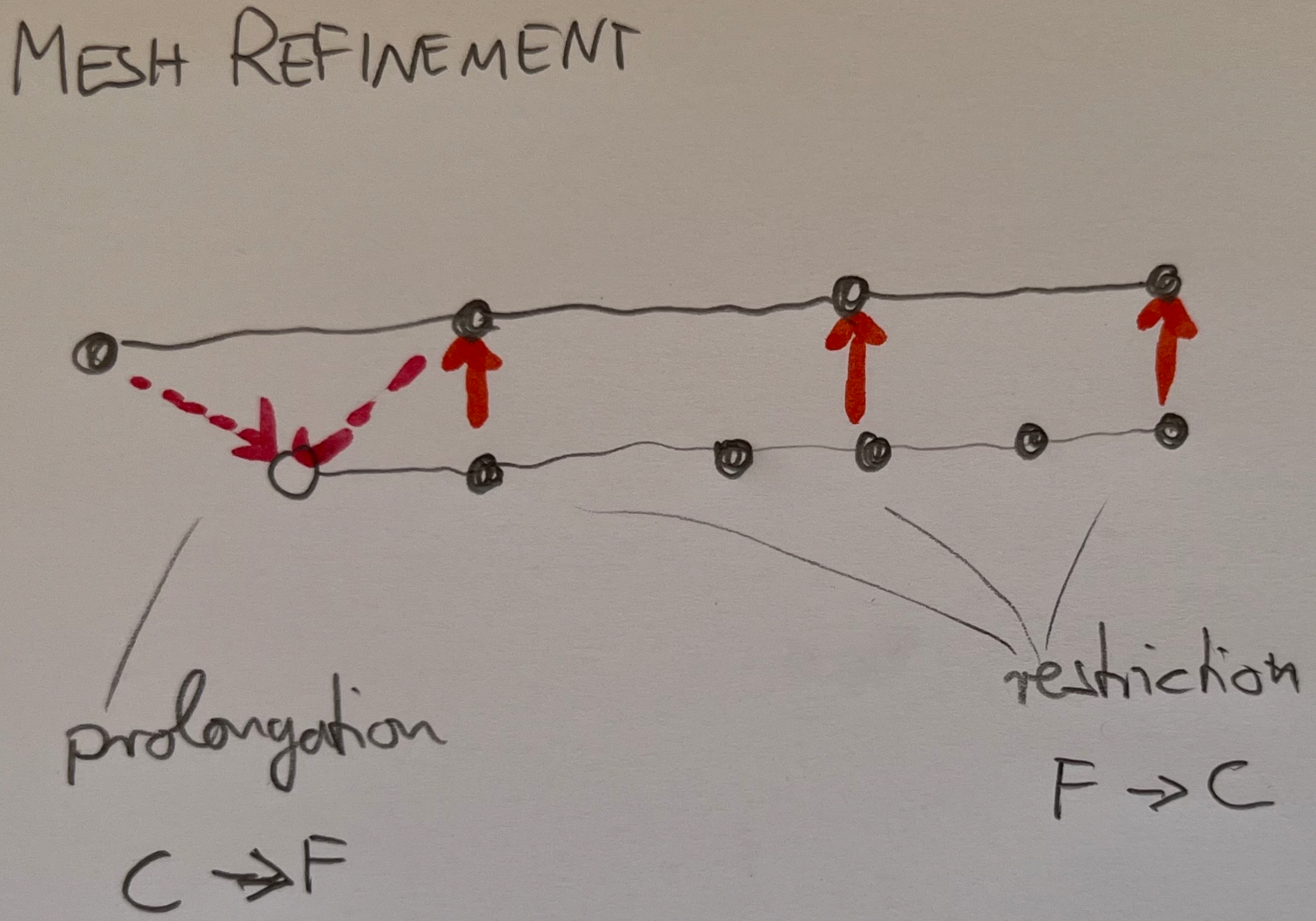
https://amrex-codes.github.io/amrex/docs_html/

Ghost Zones



https://amrex-codes.github.io/amrex/docs_html/

Mesh Refinement



STAGGERED GRIDS

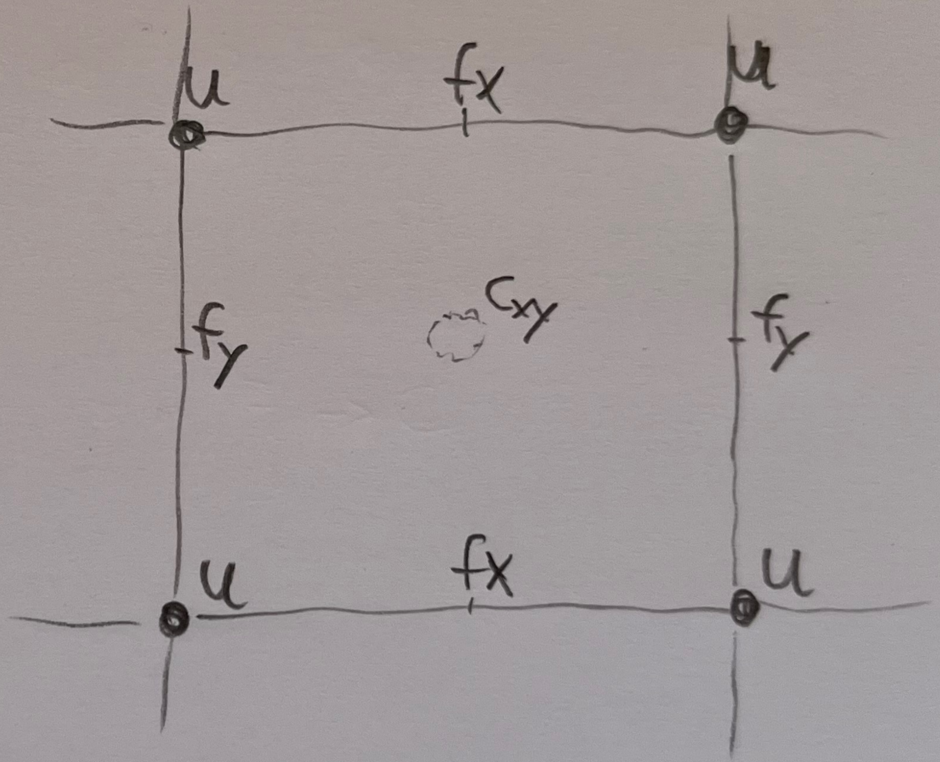
Staggered Grids

u

$f_i := \nabla u$

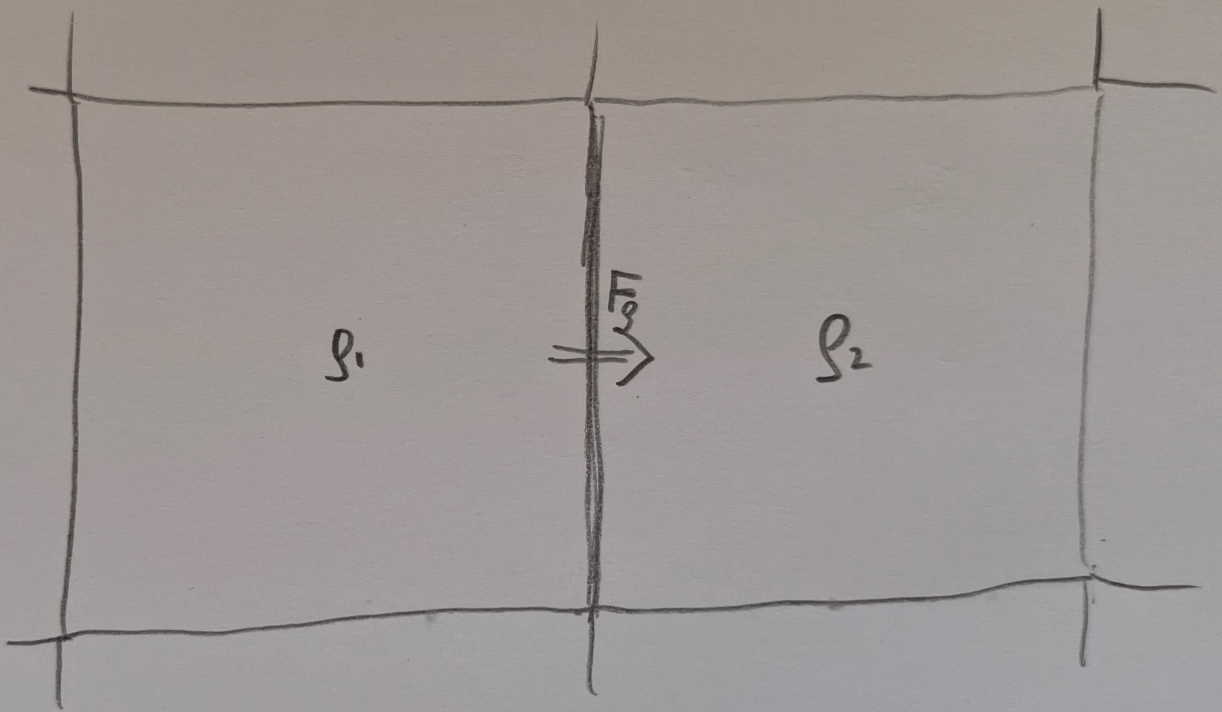
$c_{ij} := \nabla \times f_i = 0$

$\partial_i f_j - \partial_j f_i = 0$



FLUXES

Fluxes and Conservation



$$\dot{\rho} = -\nabla \cdot F_\rho$$

https://www.canadianfreestuff.com/safety-1st-canada/

Safety first

Safety first

- Cactus and CarpetX keep track of which regions of what variables have valid values
  - Each scheduled function specifies which variables it **reads** and which it **writes**, implementation errors are caught
  - Many programming languages (C, C++, Fortran) can do this for scalar values, but not for array elements

- For a beginning users the Einstein Toolkit looks like a black box. This helps catch errors.
- Use *poisoning* to check your code.