Structured Mesh Technologies

Presented to
ATPESC 2017 Participants

Ann Almgren
CCSE Group Lead & Senior Scientist
LBNL

Q Center, St. Charles, IL (USA)
August 7, 2017

U.S. DEPARTMENT OF ENERGY | Office of Science

ATPESC Numerical Software Track

EXASCALE COMPUTING PROJECT

Argonne NATIONAL LABORATORY | BERKELEY LAB | Sandia National Laboratories | Rensselaer | SMU

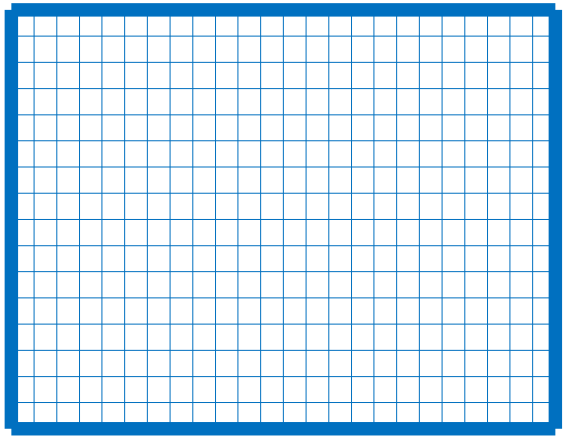
Outline

- A look at solving time-dependent PDE's
  - Choosing your spatial discretization
  - What are possible mesh refinement strategies?
  - What are pros and cons of structured vs unstructured?
- AMR for Structured Mesh
- What is AMReX?
- What does AMReX do for you?
- What doesn't AMReX do for you?
- How do you use AMReX?
- Hands-on exercises using AMReX

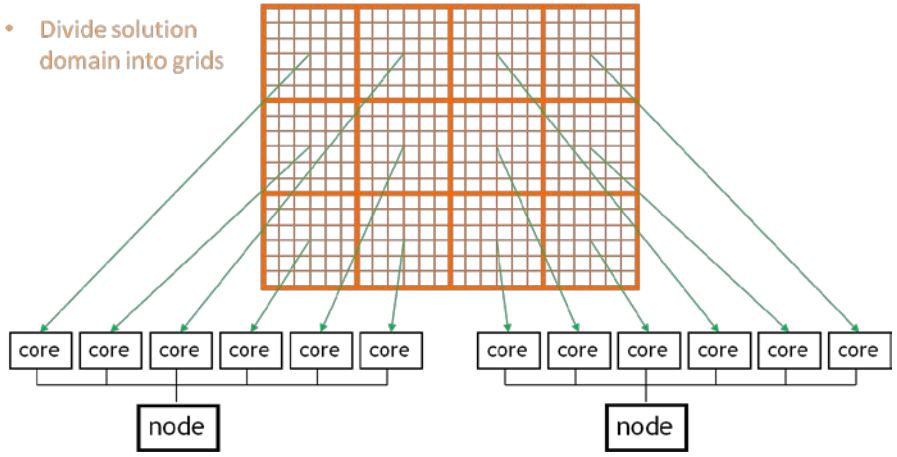
Solution of Time-Dependent PDE's

Suppose you want to solve a single system of time-dependent PDEs in a regular domain.

With unlimited resources – both computational time and memory – you could discretize the PDE on a uniform structured grid and use a fixed-in-time-and-space time step to advance the solution.



Unew[i][j] = Uold[i][j] + speed * dt

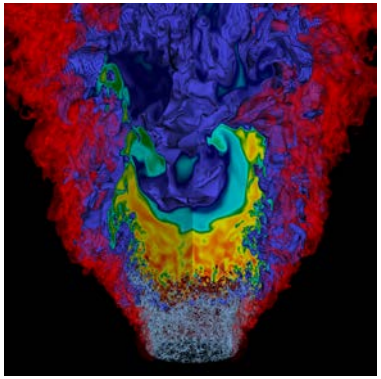


You could use domain decomposition to break the domain into smaller grids to distribute to MPI processes.

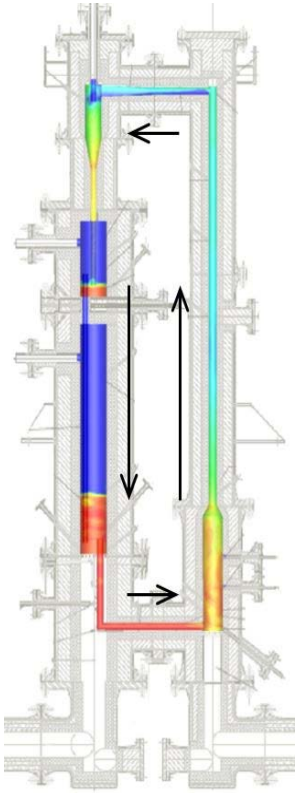
End of story.

Solution of Time-Dependent PDE's (p2)

However, most problems you want to solve aren't that simple:



- Multiphysics – more than one physical process
- Multirate – different time scales
- Multiscale – different spatial scales



You may not want to solve it in a regular domain:

- Complex domain boundaries
- Complex material boundaries

And, unfortunately, you don't have unlimited resources.

- You will probably need some type of adaptivity – changes in the mesh due to the geometry or the solution or both

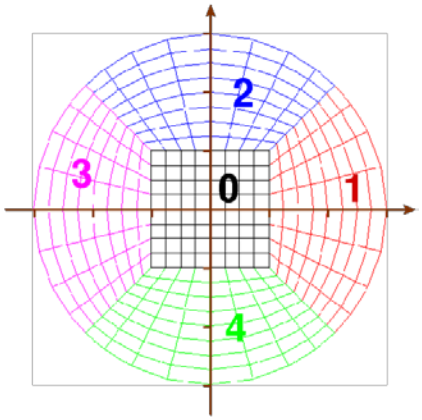
Types of (Structured) Mesh Refinement

Unstructured meshes obviously have the ability to adapt to the problem geometry and the solution (see the next talk).

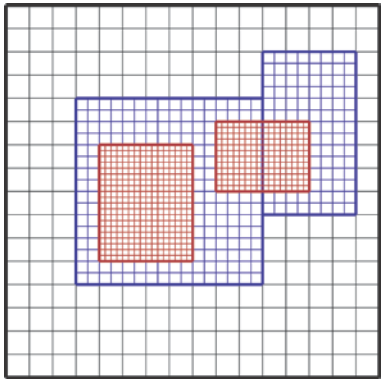
What about structured meshes?

Ways to increase resolution in certain areas include:

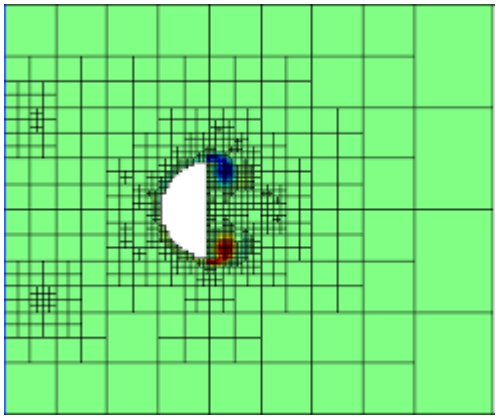
- mapped meshes
- point-wise refinement (becoming less common)
- block-structured refinement – patch-based or octtree



Mapped multiblock (Chombo)



Block-structured refinement – patch-based (AMReX)



Block-structured refinement – octree-based (Gerris)

Pros and Cons of Structured Mesh

Pros:

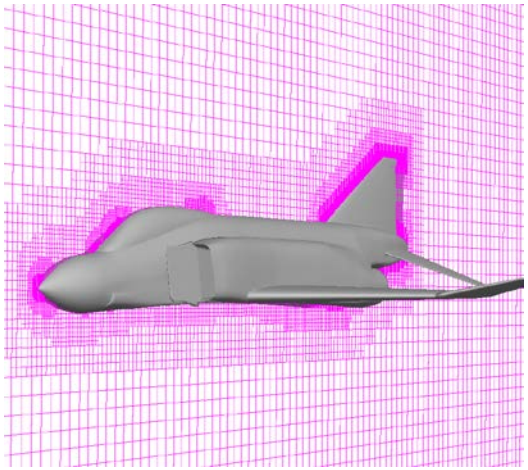
- Ease of discretization –
  - metric terms (if any) are more regular
  - connectivity is easy (global index space even with AMR)
  - can gain order of accuracy by cancellation of terms

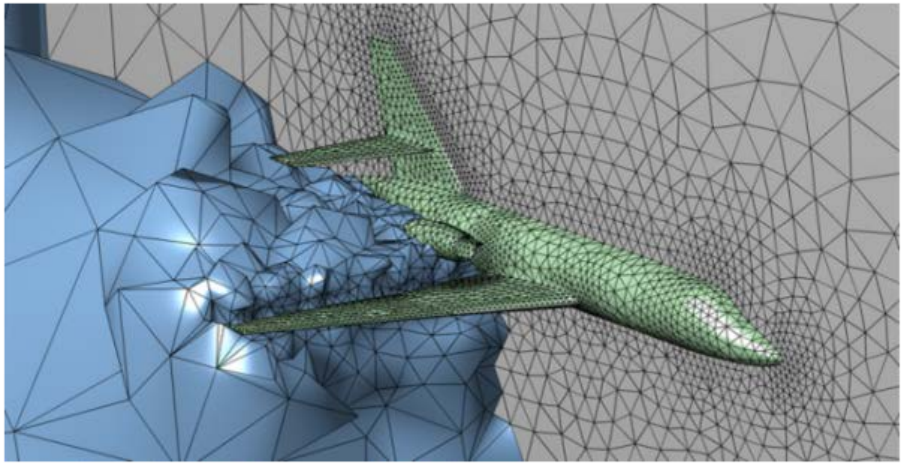
- Ease of vectorization –looping over large-enough contiguous blocks of data to exploit vectorization

- Ease of hybrid parallelism – distribute grids to MPI processes, for example, while threading over cells within a grid

Pros and Cons of Structured Mesh (p2)

- Geometry does introduce irregular meshing but …
  - irregularity can be localized near geometric features (doesn't shape the whole mesh)
  - mesh generation is (relative to unstructured meshes) "fast and easy" – especially important when geometry changing in time



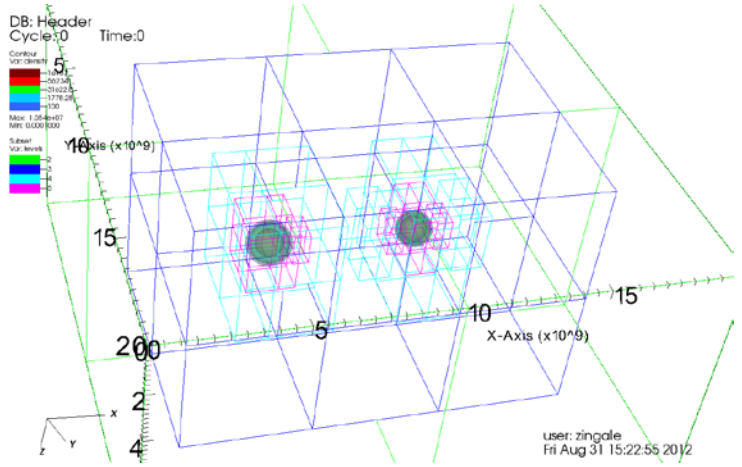


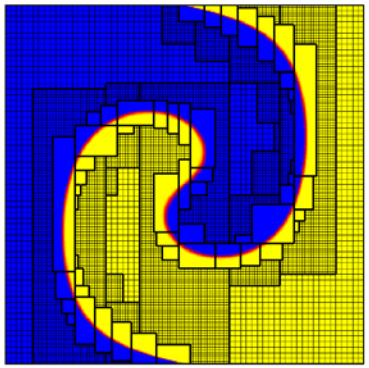
Cons:

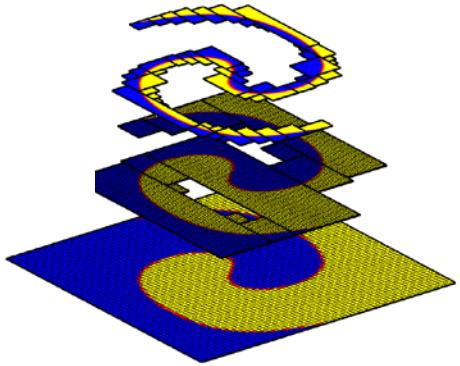
- At physical boundaries or material boundaries
  - can't just align cell faces to boundaries
  - potential loss of accuracy near boundaries
- Can modify resolution without introducing coarse-fine boundaries

What Exactly is Block-Structured AMR?

In block-structured AMR, the solution is defined on a hierarchy of levels of resolution, each of which is composed of a union of logically rectangular grids/patches







- Patches can change dynamically
- Oct-tree refinement with fixed size grids is special case
- More generally, patches may not be fixed size and may not have unique parent

Adaptive Mesh Refinement Isn't New

Block-structured AMR has a rich history … from the 1980's to today

- Different frameworks, different languages
- Different styles of refinement – block-structured vs oct-tree vs point-wise …
- Different application targets ... from hyperbolic conservation laws to elliptic solves to complex multiphysics applications

See, e.g., Donna Calhoun's website, for available codes today:

http://math.boisestate.edu/~calhoun/www_personal/research/amr_software

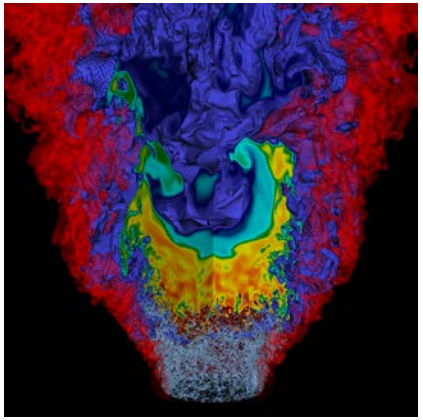
What is AMReX?

- AMReX is an ECP-funded software framework to support the development of block-structured AMR applications for current and next-generation architectures
  - Starting point was: BoxLib & Chombo (existing AMR frameworks)

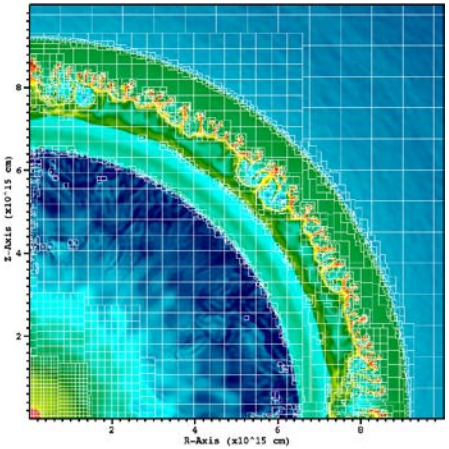
- AMReX repo is freely available at https://www.github.com/AMReX-Codes/amrex

- Originally designed for solution of time-dependent PDEs but is not constrained to PDEs
  - Doesn't dictate anything about the physics, the discretization or the numerics other than fundamentally uses block-structured mesh

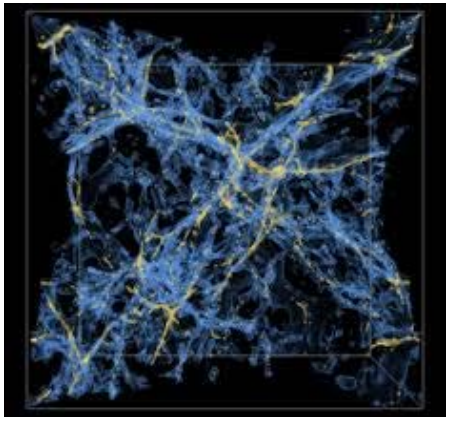
Current Applications Using AMReX



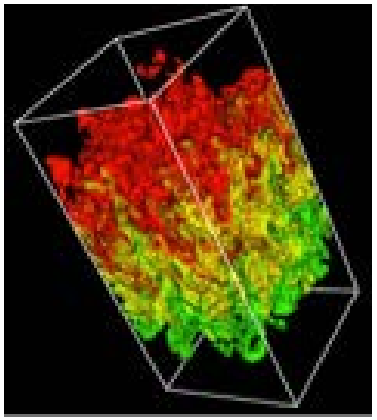
Combustion



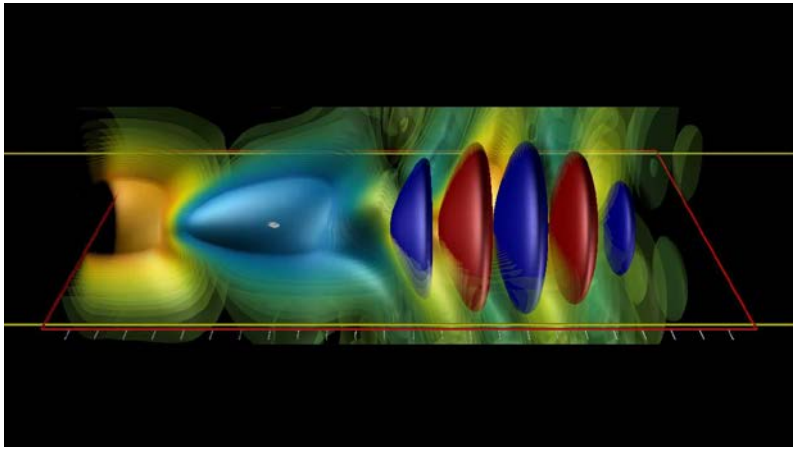
Astrophysics



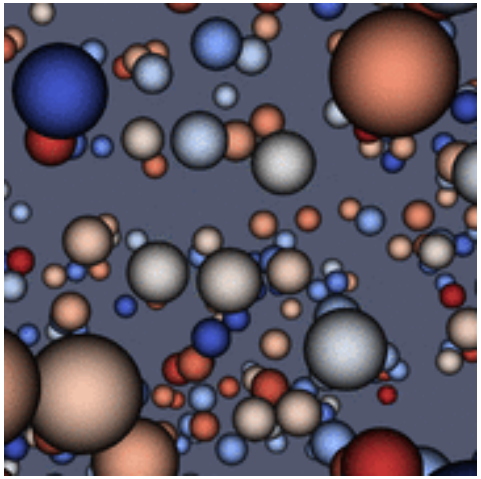
Cosmology



FLASH



Accelerators

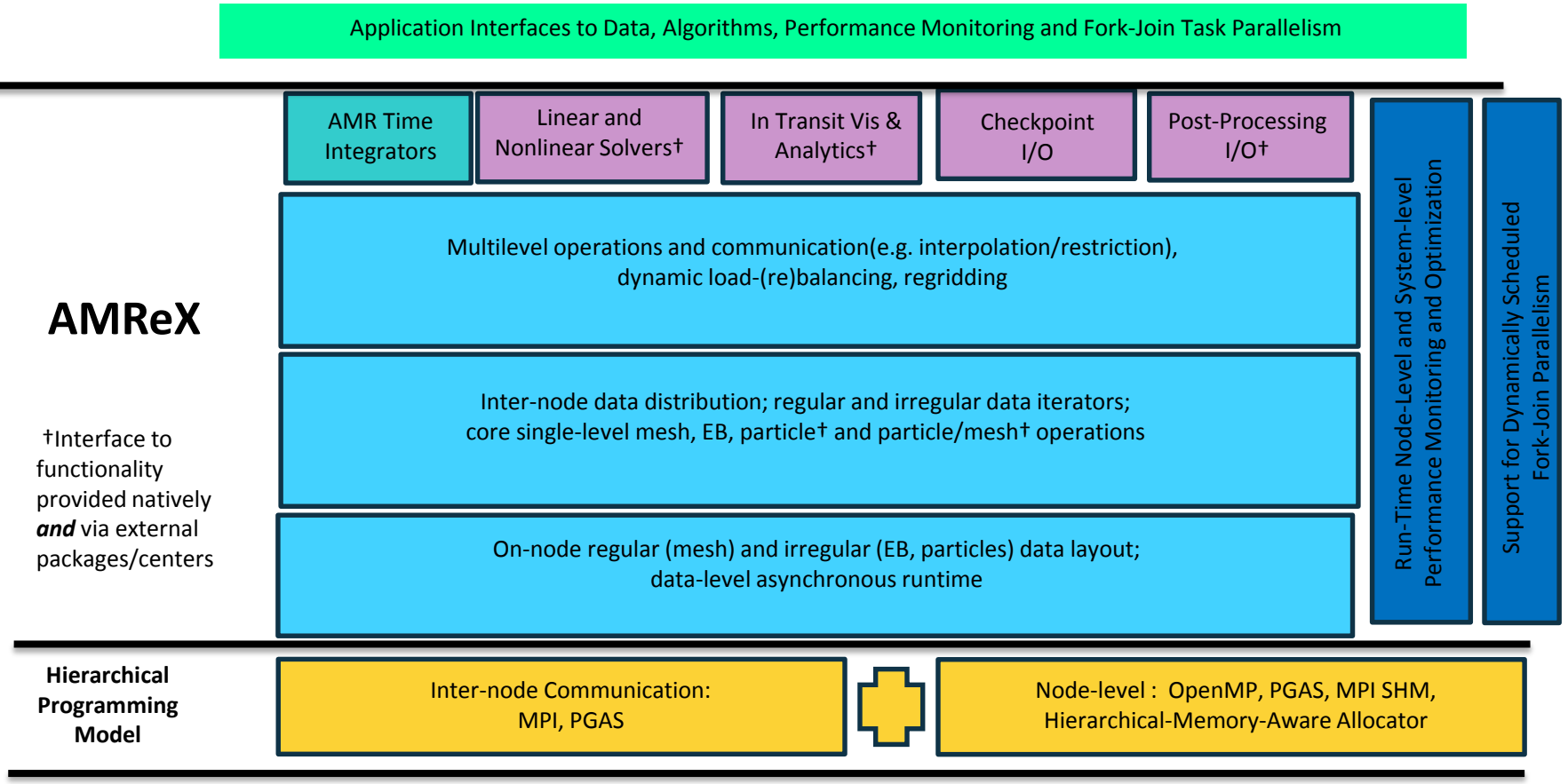


Multiphase Flow

What does AMReX Do For You?

- Provides support for
  - explicit & implicit mesh operations
  - multilevel synchronization operations
  - particle and particle/mesh algorithms
  - Solution of parabolic and elliptic systems using geometric multigrid solvers
- Hierarchical parallelism: hybrid MPI + OpenMP with logical tiling to work efficiently on new multicore architectures
- Core functionality written in C++ with frequent use of Fortran kernels; Fortran-interface to C++ also available
- Highly efficient parallel I/O for both restart and plotfiles
- Visualization format supported by Visit, Paraview, yt, etc

What does AMReX do for you? (p2)



What Doesn't AMReX Do For You?

AMReX is designed to help the application developer access other software packages to provide additional functionality

For example, AMReX provides interfaces to other solvers, eg

- hypre and PETSc for algebraic multigrid solvers
- SUNDIALS for solving ODEs

How Do You Use AMReX?

There are three primary ways to build an application with AMReX:

1. Use the core **single-level** data structures and operations only (amrex/Src/Base)

2. Use the AmrCore **multi-level** functionality in addition
   - amrex/Src/AmrCore

3. Use the full **AMR time-stepping** functionality
   - amrex/Src/Amr

Take-Away

- AMReX is a framework that you can use to build a massively parallel multiphysics application on block-structured grids

- AMReX allows you to build your application using your discretizations and solvers of choice but provides the parallelism “for free” (e.g. you will never type “MPI_Send”)

- A large amount of what you would normally have to write yourself is supported within AMReX:
  - Domain decomposition with distribution of grids to MPI processes
  - Logical tiling of grids with distribution of tiles to OpenMP threads
  - Ghost cell filling
  - Support for multilevel mesh operations (coarsening / interpolation between different levels – recall multiple grids at each level)
  - Support for particles and particle-mesh operations

- If you can define it, you can build it.