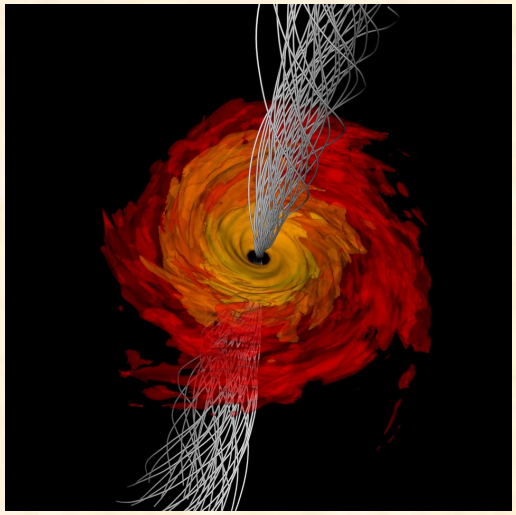
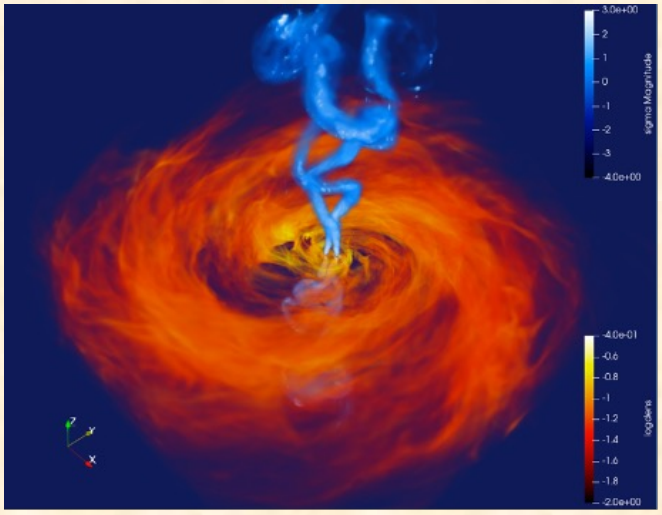
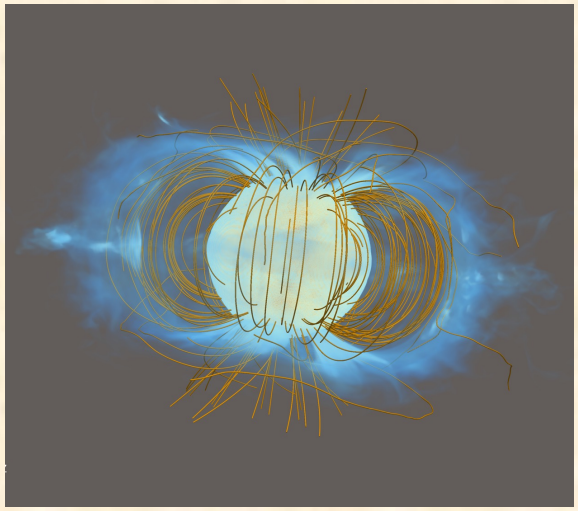
Introduction to AthenaK

Jim Stone

Institute for Advanced Study







Motivation

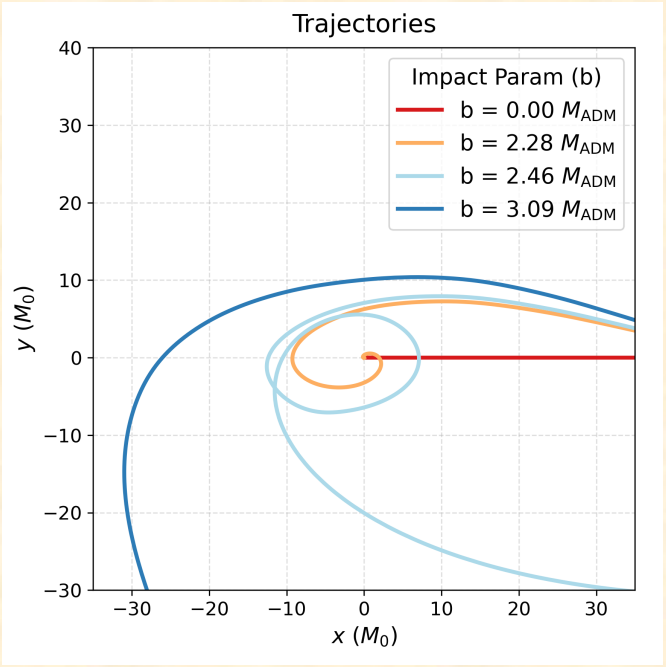
Computation has become an essential tool for understanding complex, nonlinear dynamics in a variety of regimes:

- GW emission from merging BHs
- Merging neutron stars, and GRBs
- Stellar core collapse and Supernovae
- Accretion flows
- Turbulence in the ISM and star formation
- Dynamics of protoplanetary disks and planet formation
- Galaxy formation and the IGM
- Etc., etc.

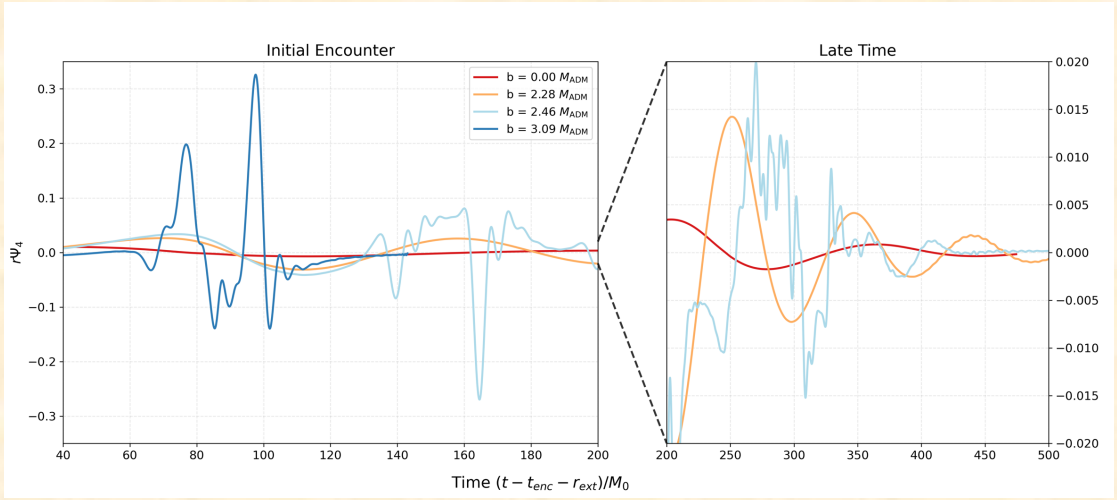
Example: merger of boosted BHs

Work with H. Zhu, F. Pretorius

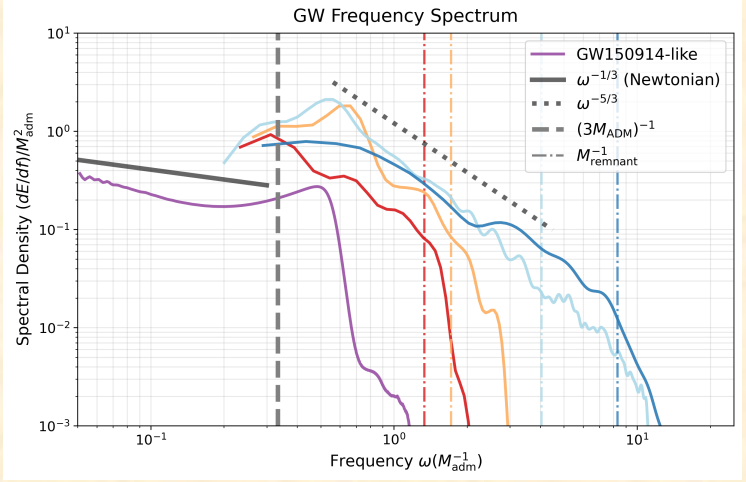
Study GW emitted in collisions between BHs boosted up to Γ=5.1 and various impact parameters, pure vacuum.



Coordinate paths

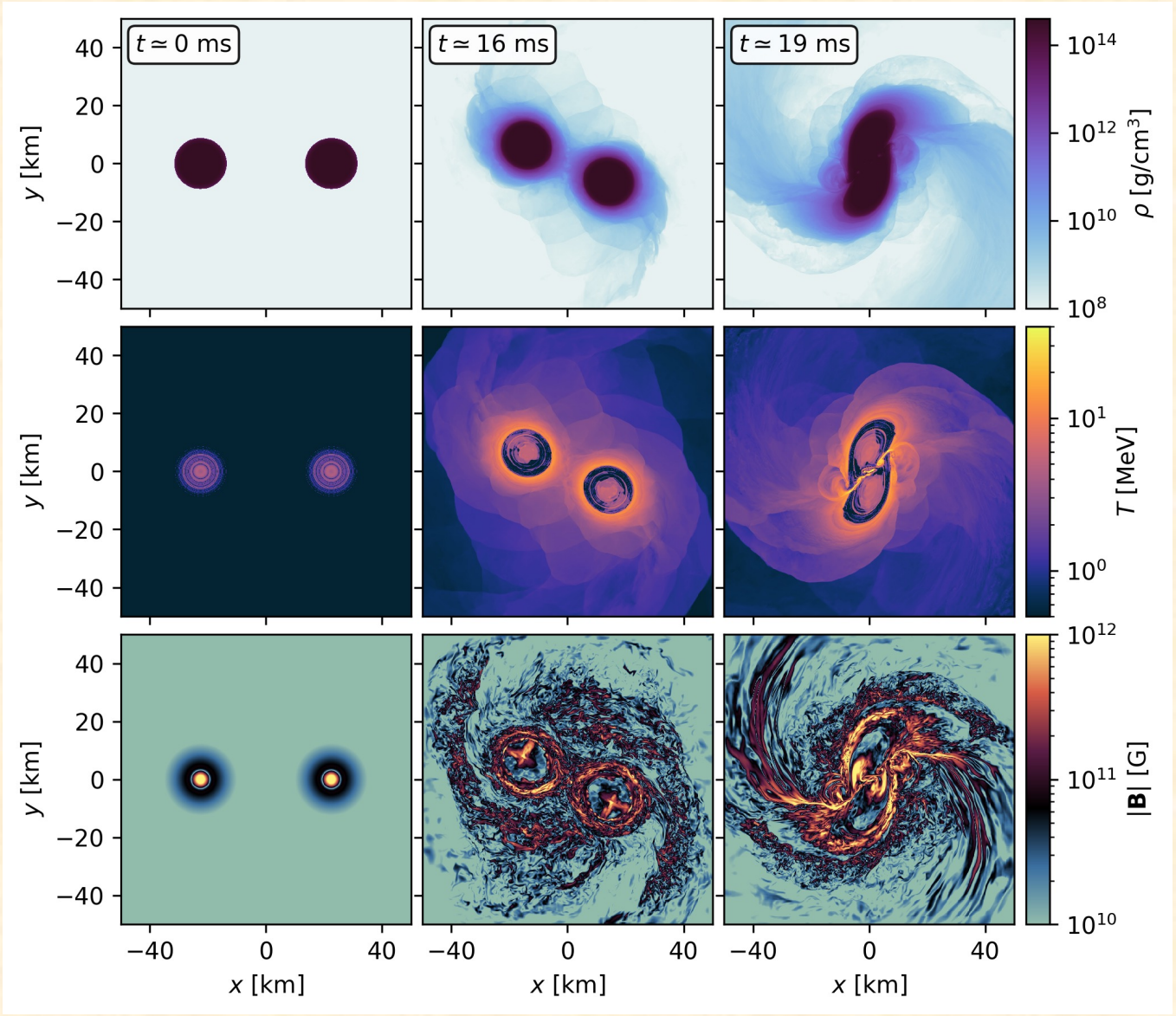


GW form emitted



Spectrum shows strong nonlinear interactions

Example: zoom-in to merging BNS



Work led by E. Gutierrez

Density

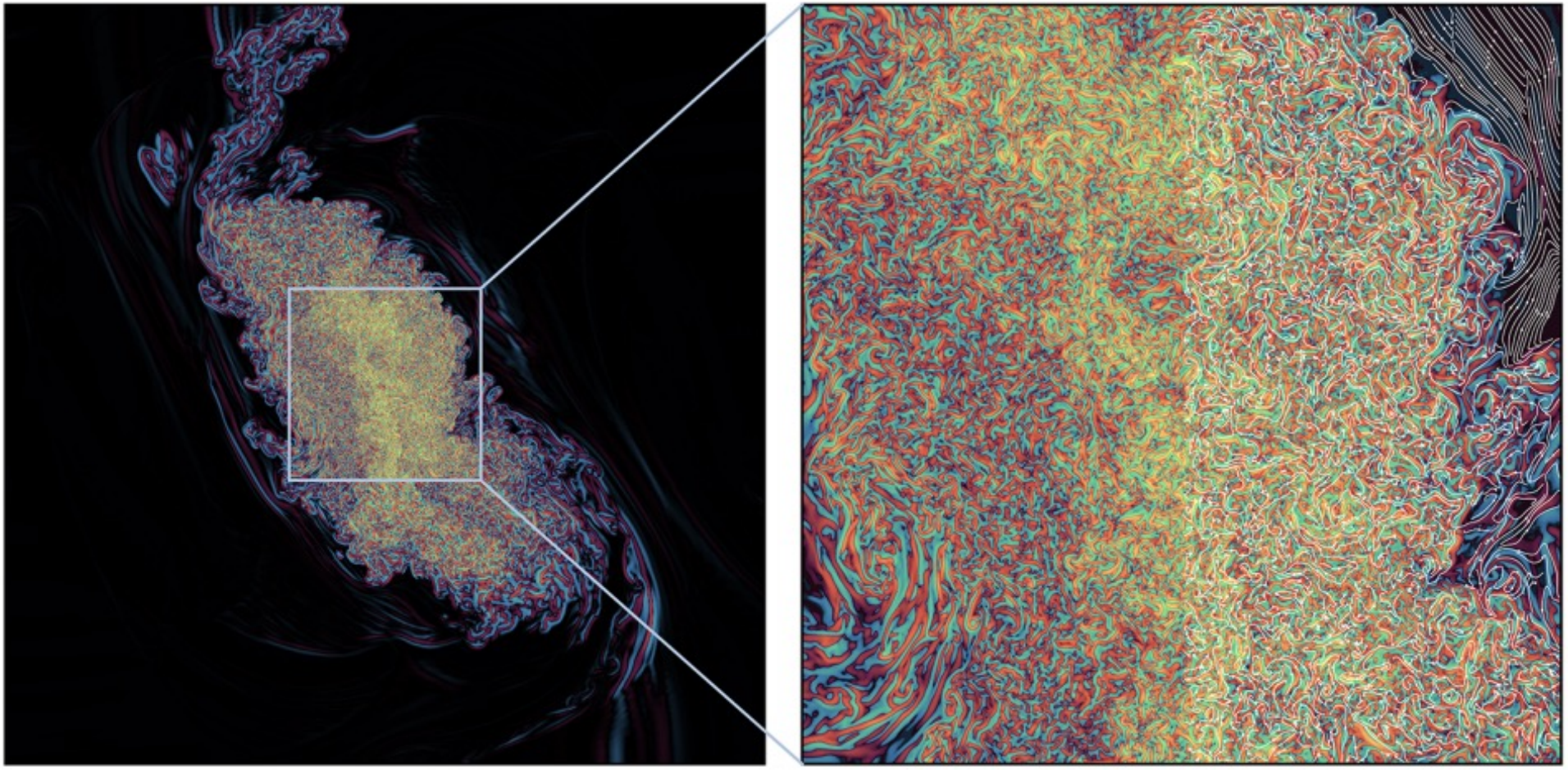
Kelvin-Helmholtz instability (KHI) at contact surface

Temperature

Magnetic field strength

Are stronger fields produced at higher resolution?

Zoom-in with 1.5m resolution at the highest level

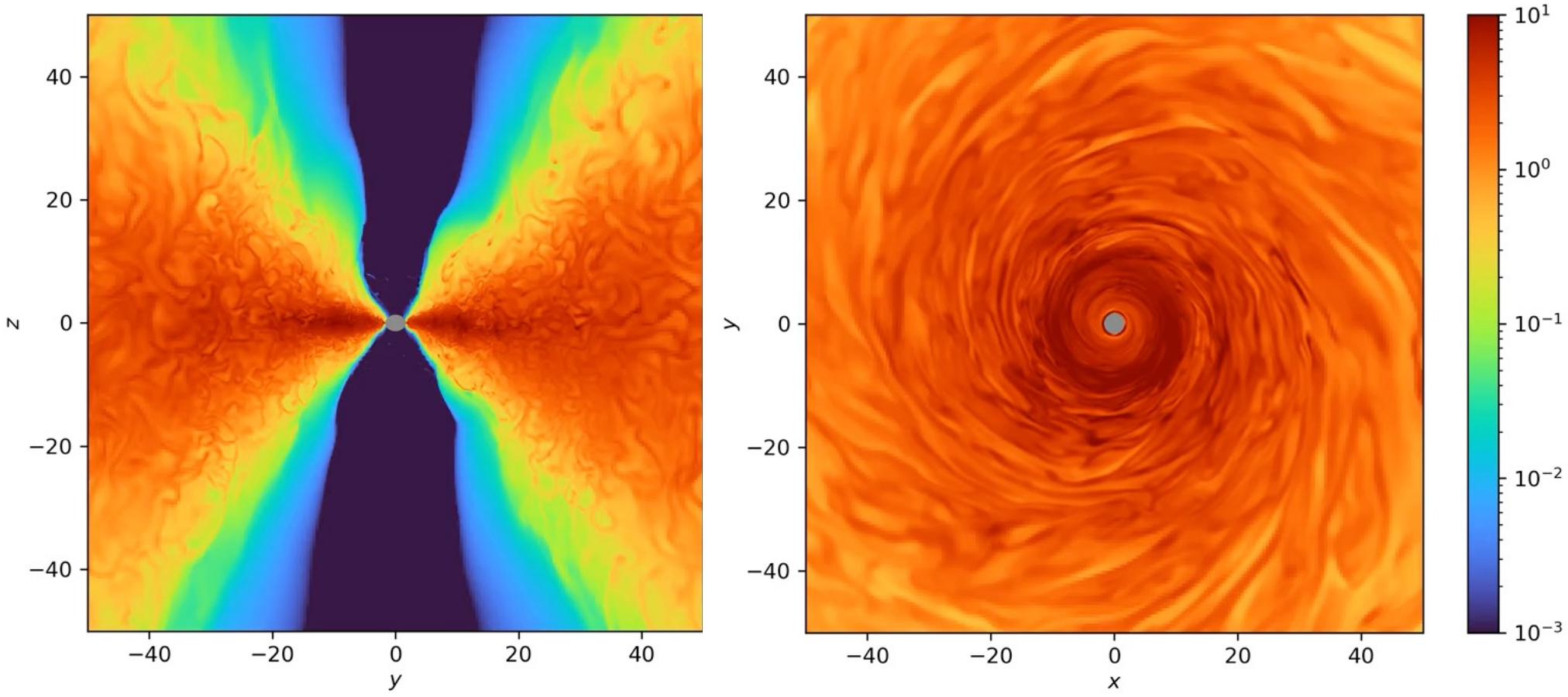


Dynamo driven by Kelvin-Helmholtz instability generates 10^{16} G fields!

Example: Radiation-dominated black hole accretion

Work led by L. Zhang

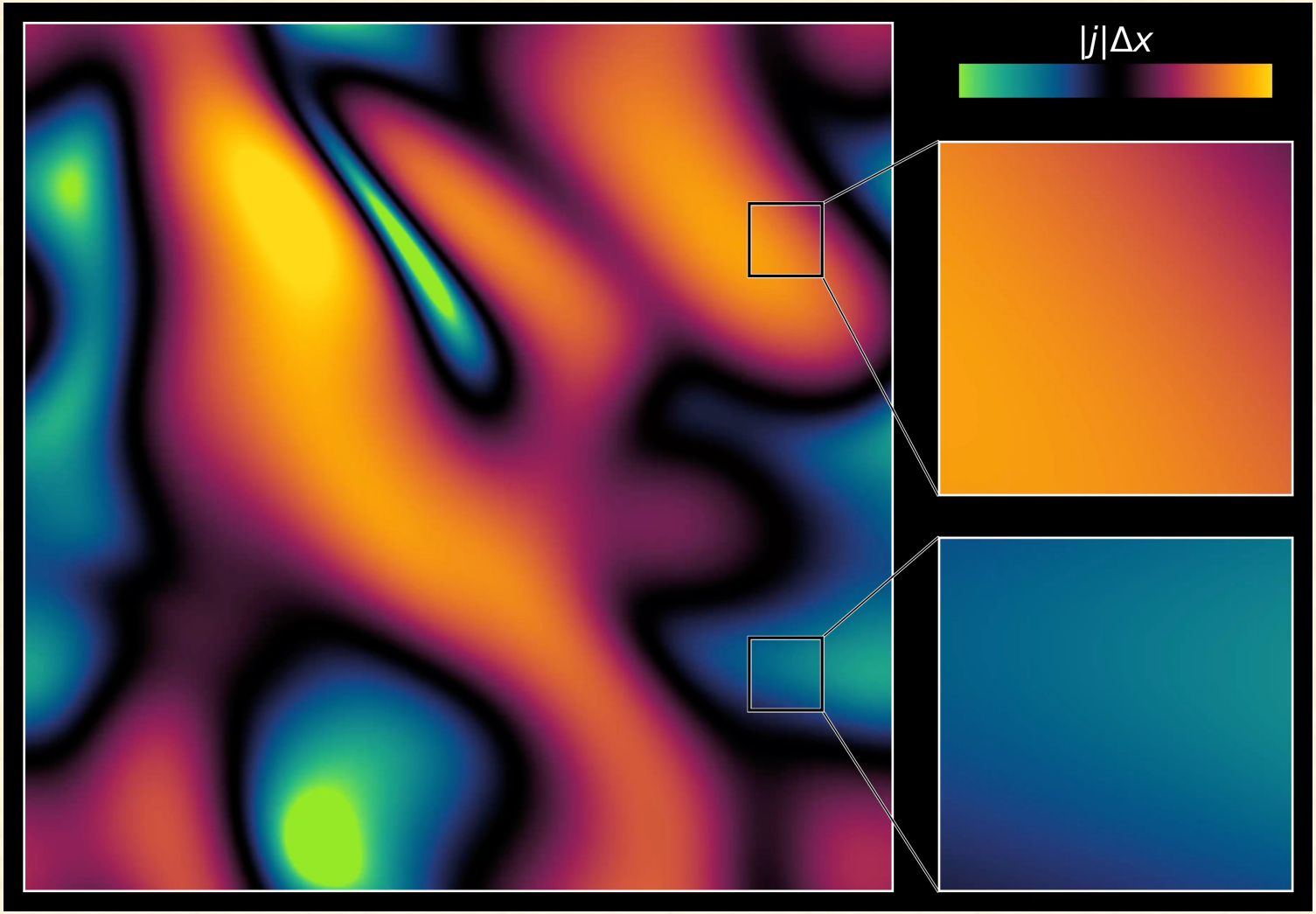
Starting from a torus, turbulent magnetized accretion flow at late times ($t > 10^4$ M) around a rapidly spinning black hole



Full transport solution, not diffusion or M1 approximation!

Example: MHD turbulence in ISM

Exascale enables driven MHD turbulence with a resolutions up to $10{,}000^3$ (Beattie+ 2025, Fielding+ 2025)



Calculations that took years can now be run in days.
Enables study of CR diffusion in MHD turbulence, among other projects

Summer School is about much more than AthenaK

Science applications:

- Proto-planetary disks (Zhu)
- Galaxy formation and IGM (Fielding)
- ISM and star formation (Ostriker)
- Stellar core collapse and SNe (Halevi)
- Planet formation (Simon)
- GW from merging BHs (Campanelli)
- Binary NS mergers (Gutierrez)

Numerical Methods

- Numerical relativity (Campanelli, Zhu)
- Radiation Transport (Bhattacharyya, Zhang)
- Nuclear Reaction Networks (Smith)
- Particle methods (Wong)
- GR MHD (Fields)
- Multigrid and self-gravity (Velasco)

Nevertheless, my lectures are going to focus on AthenaK!

But much of what we cover is applicable to other codes too.

Three lectures on AthenaK

1. Introduction to AthenaK
2. MHD Algorithms
3. Black hole accretion disks

Lecture 1: Introduction to AthenaK

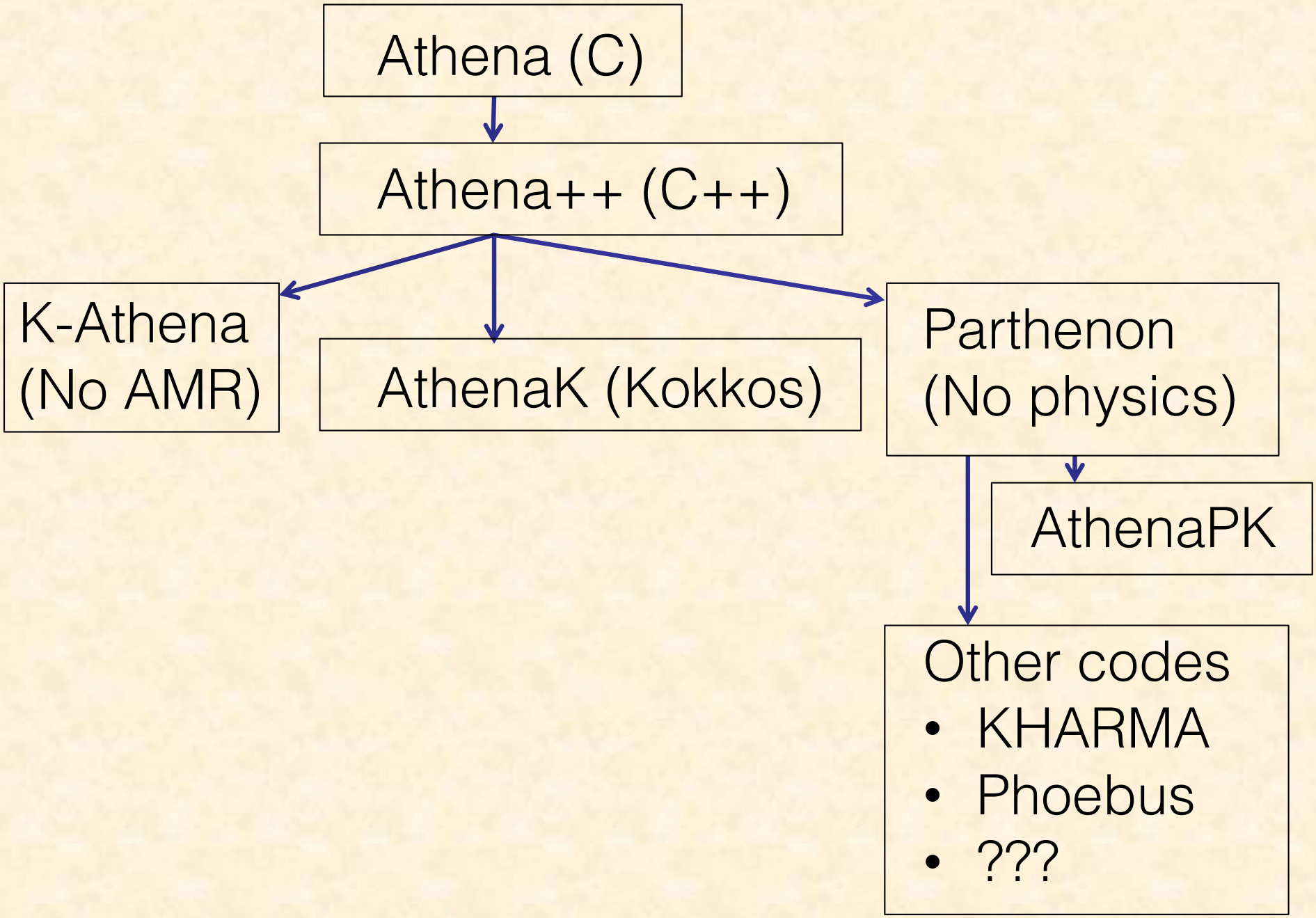
1. What is AthenaK?
2. Introduction to Kokkos.
3. The AMR Framework.
4. Overview of Solvers.

What is AthenaK?

- Block-based AMR framework in C++
- Performance portability using Kokkos (runs on anything)
- Fluid solvers
  - Newtonian/SR/GR hydro and MHD (Stone+ 2024)
  - GRMHD in dynamical spacetimes (Fields+ 2024)
- Numerical Relativity (NR) solver using Z4c (Zhu+ 2024)
- Relativistic Radiation Transport (White+ 2023)
- Particles: CRs, Lagrangian tracers
- Publicly available on GitHub

Plus, many new features being ported from Athena++ using Claude

Some history



Developing AthenaK: team effort

Athena++: contributions from **35 developers**, most number of commits by:

Kyle Felker, Kengo Tomida, Chris White, Matt Coleman, Munan Gong, Chang-Goo Kim, Yanfei Jiang, Patrick Mullen, …

AthenaK: contributions from **30 developers**, most number of commits by:

Jacob Fields, Hengrui Zhu, Patrick Mullen, Minghao Guo, George Wong, David Radice, …

Learning about AthenaK

AMR mesh:

- Athena++ method paper (Stone et al., ApJS 2020)

Algorithms and solvers:

- AthenaK method paper (Stone et al., ApJS 2026)
- AthenaK NR method paper (Zhu et al., ApJS 2025)
- AthenaK GRMHD method paper (Fields et al., ApJS 2025)

How to use the code:

- Documentation is now live!
- https://ias-astrophysics.github.io/athenak-docs/

Kokkos: running on exascale

OLCF “Frontier” system,
began operations early 2023

ALCF “Aurora” system,
began operations Feb 2025

Each is about 1.5 Eflops, about 10^9x faster than Cray X-MP (first open science national supercomputer installed in 1986).

Challenge: each vendor (Nvidia, AMD, Intel) adopts a different programming model (CUDA, HIP, SYCL).

One solution: Kokkos (Trott et al. 2021)

Reasons for adopting Kokkos:

- Well-supported open source code project https://github.com/kokkos/
- It's just C++ (templates and abstractions for data containers and parallel execution strategies).
- Builds into the native programming environment of target architecture: exploits vendor optimizations.
- This results in ***performance portability.*** Ability to develop and test on laptops is crucial.
- Enables exascale for application scientists.

Performance portability

PLM, RK2, HLLE solver

zone-cycles/sec ($\times 10^6$)	AthenaK CPU single core※	AthenaK CPU one node(96 cores)※	AthenaK GPU A100	Athena++ CPU single core	Athena++ CPU one node(96 cores)
Hydro	4.89	184	555	4.94	186
MHD	2.00	80.8	260	2.24	90.5
SR Hydro	1.51	88.4	263	2.30	136
SR MHD	0.883	55.4	142	1.02	65.5
GR Hydro	0.728	44.1	220	1.57	95.3
GR MHD	0.350	22.3	98.7	0.550	35.0

Table 1. Performance of **AthenaK** on various devices

	Apple M1 pro (8 cores)	Intel Xeon Gold 6326 (32 cores)	NVIDIA V100	NVIDIA A100	AMD MI250	NVIDIA H100	NVIDIA Grace Hopper
hydro	34	63	298	614	405	677	1077
MHD	11	33	158	298	190	290	565

Data structures in Kokkos

Most important new data structure is `Kokkos::View`

- Up to 6D managed array of basic types, or structs of POD
- Allocation/de-allocation/reference counting handled automatically
- Array layout automatically optimized (including padding) for machine architecture at compile time
- Memory space and other attributes set by programmer

```
Kokkos::View<double ***,
             Kokkos::LayoutRight,
             Kokkos::DefaultExecutionSpace::memory_space> my_array;
```

AthenaK uses wrapper classes to hide most of the Kokkos magic, e.g.

```
template <typename T>
using DvceArray1D = Kokkos::View<T *, LayoutWrapper, DevMemSpace>;
```

Parallel loops in Kokkos

In essence, GPUs implement *shared memory parallelization*, where parallel regions are generally (nested) loops.
Kokkos provides control structures that allow (nested) loops to be executed in parallel on the GPU.

In C++:

```
for (int i=0; i<N; i++) {
  Do_Work_Here(i);
}
```

In Kokkos:

```
Kokkos::parallel_for("Loop1", N, KOKKOS_LAMBDA (const int i) {
  Do_Work_Here(i);
});
```

Loop body specified either through *functors* or *lambdas*. Loop can either be executed on host or device.

Where to locate data?

Programmer must ensure all operations on GPU access data which is already in GPU memory spaces. Any data not already located in appropriate memory space must be copied manually.

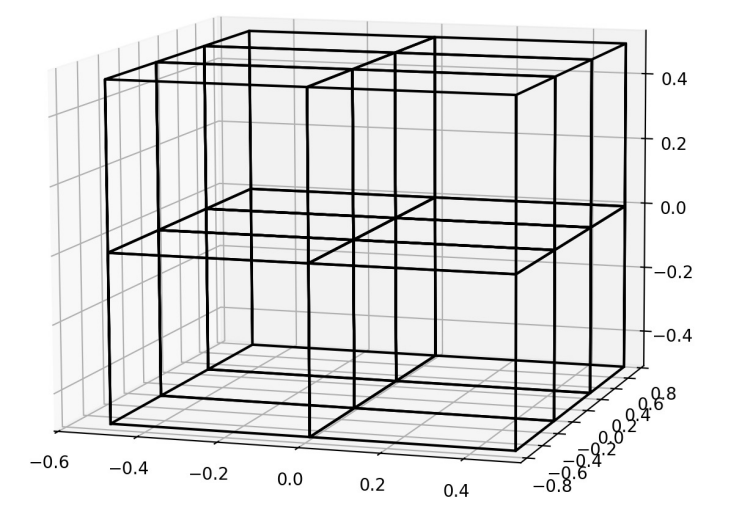
Moving data to GPU from host is slow. Therefore, best to locate data on GPU as much as possible (caveat: GPU memory is very limited).

`Kokkos::deep_copy` provides mechanism to move data between devices.

Can also use: `Kokkos::DualView` which provides arrays defined on both host and device that can be synchronized automatically.

Mesh in AthenaK

AthenaK uses a **uniform Cartesian mesh**
Root grid is decomposed into MeshBlocks



Screenshot from `plot_mesh.py` after running AthenaK with `-m` option

Controlled by `<mesh>` and `<meshblock>` blocks in input file

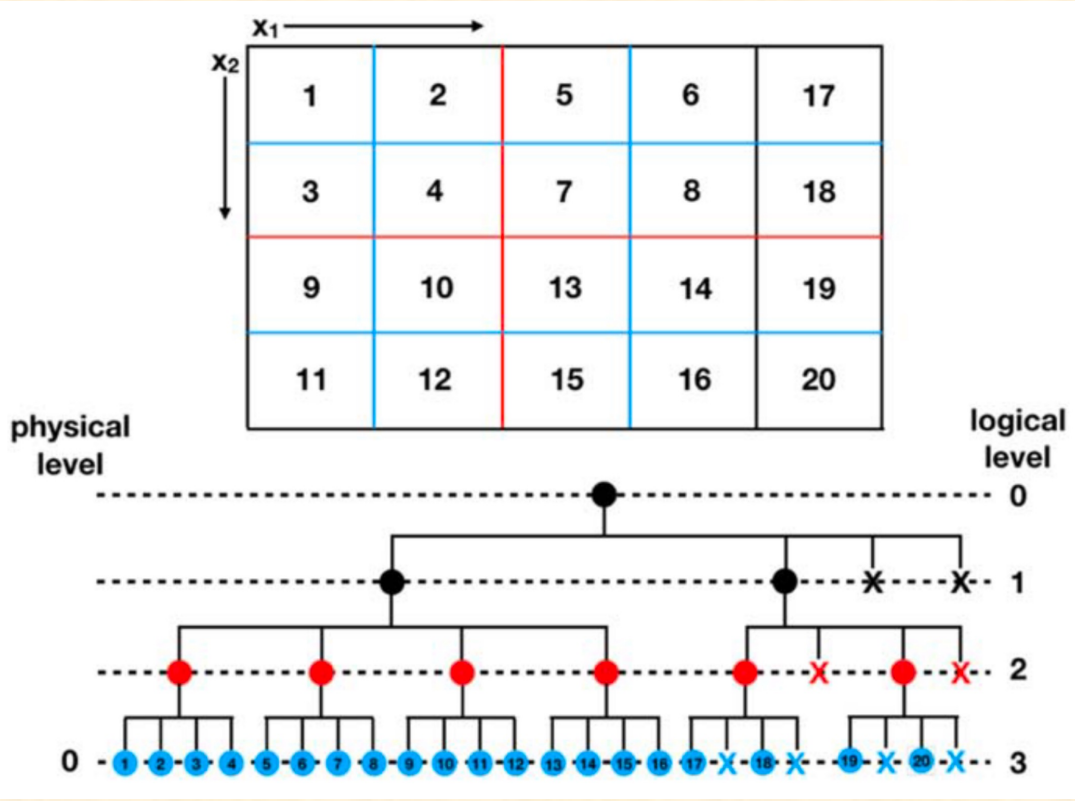
Why no curvilinear coordinates?

1. Avoids coordinate singularities at poles
2. Enables higher-order (4^{th}-order) methods
3. *For converged solutions, coordinates do not matter!*

If you need to run 2D models in axisymmetry, use Athena++ or similar codes.

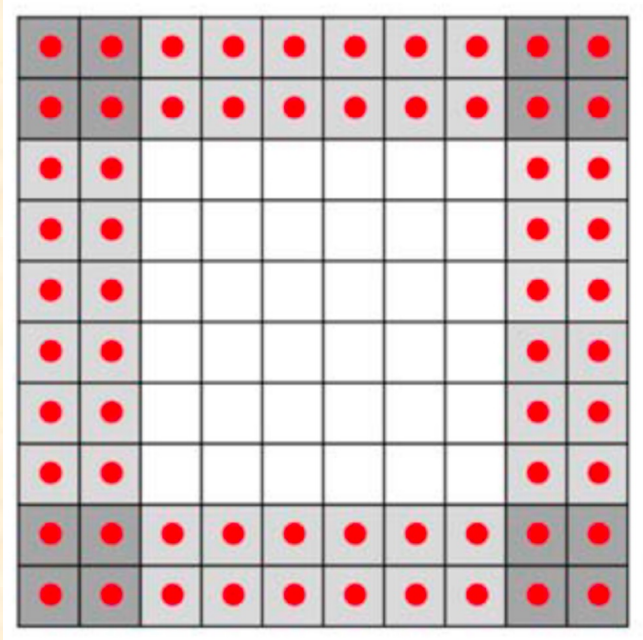
Parallelization

Distributed memory parallelization relies on domain decomposition using MPI. Grid is divideded into equally sized blocks, with each block assigned to different processors.

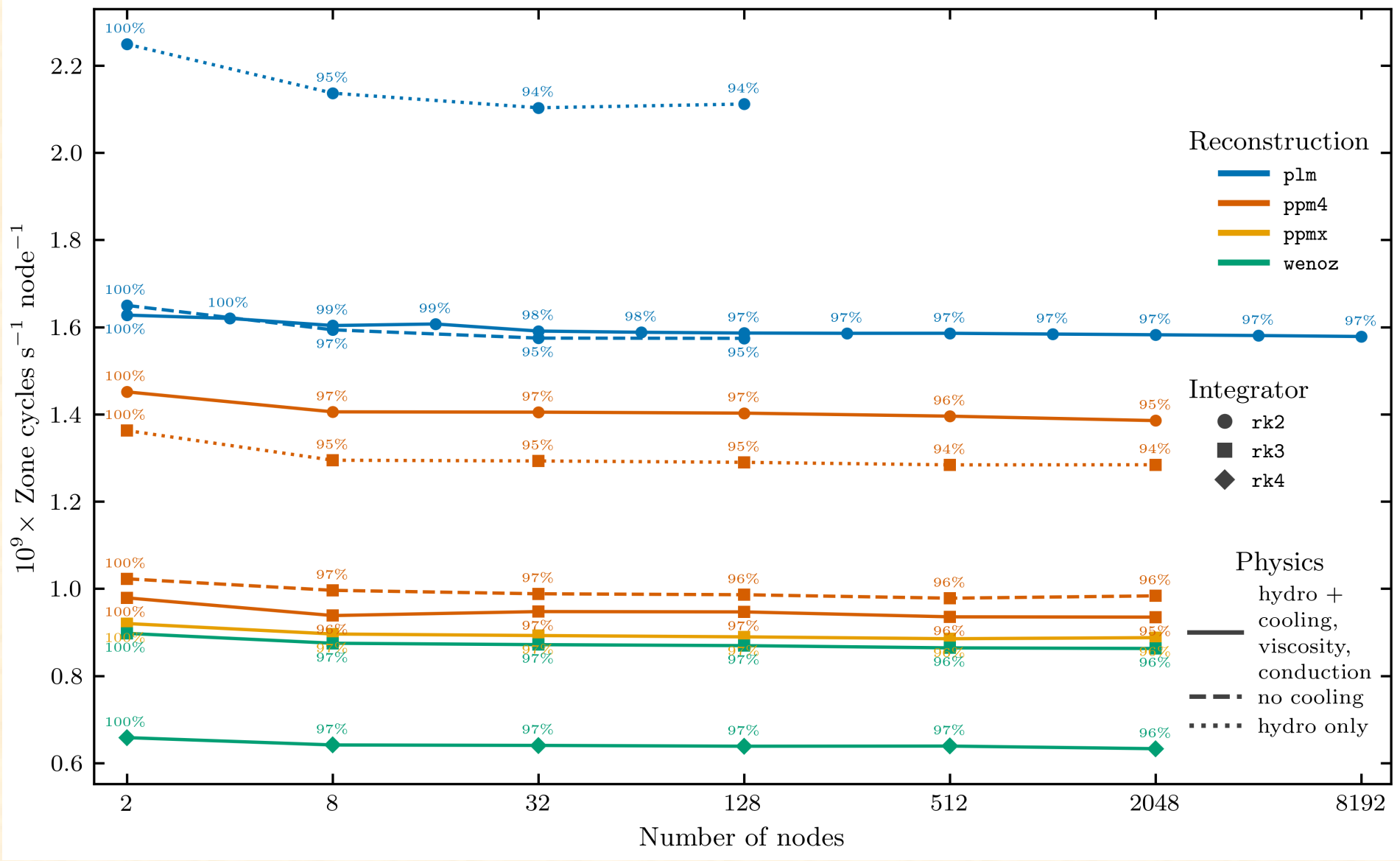


Blocks are organized into a quad(oct)-tree in 2D(3D) using Z-ordering to improve locality.

Ghost cells are communicated between blocks using MPI.

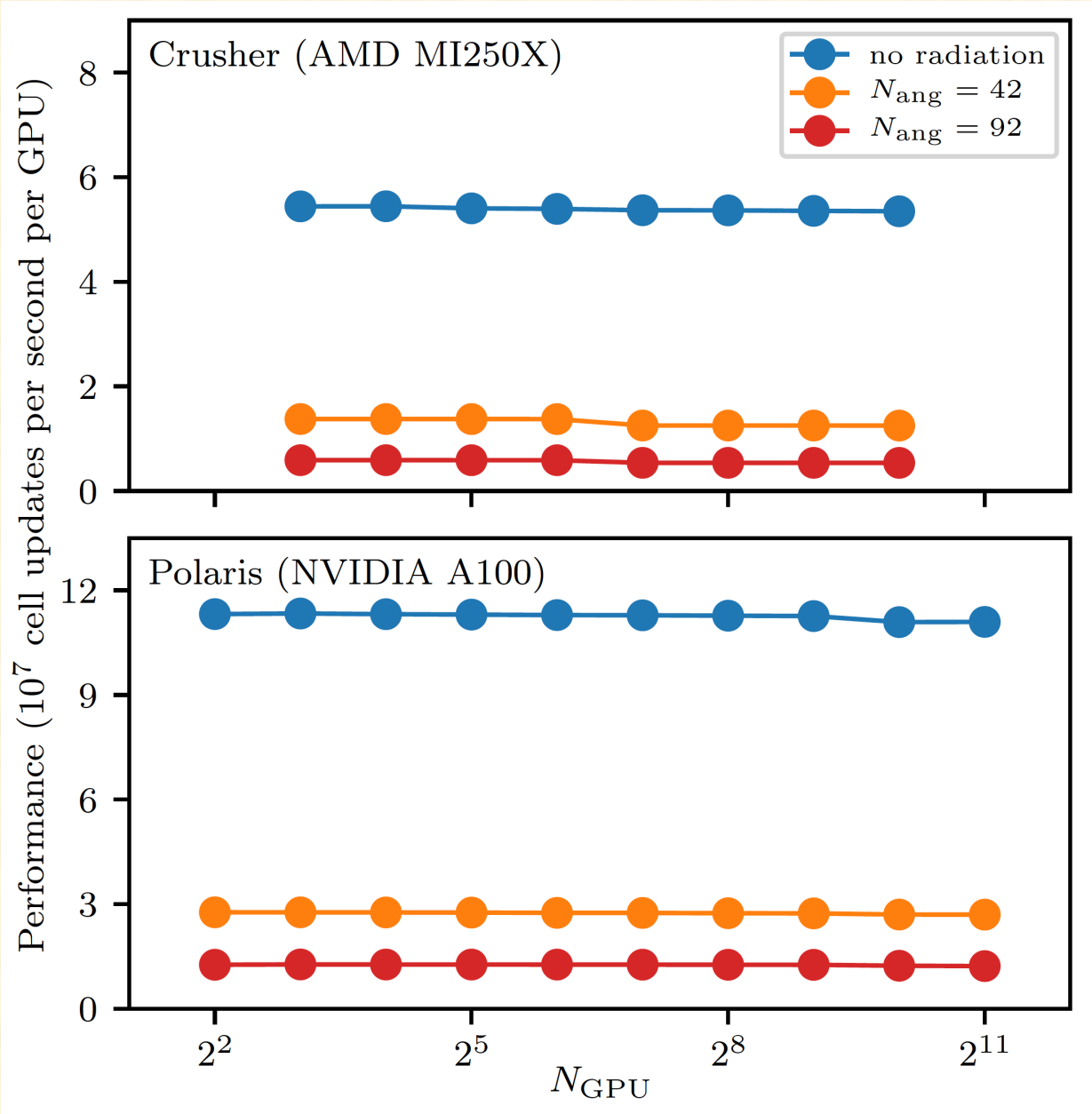


Excellent weak scaling on GPUs:



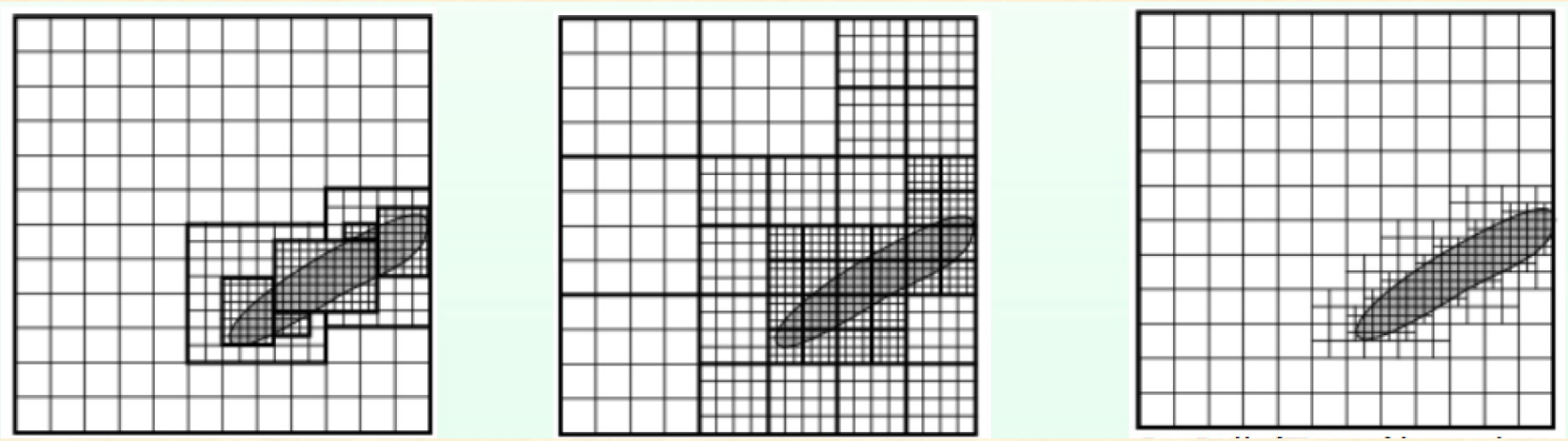
Data from Frontier (AMD MI250X) courtesy D. Fielding

Excellent weak scaling on GPUs:



Adaptive Mesh Refinement

Blocks can be refined to improve resolution where it is needed.



Patch based

AMRex
Chombo
PLUTO

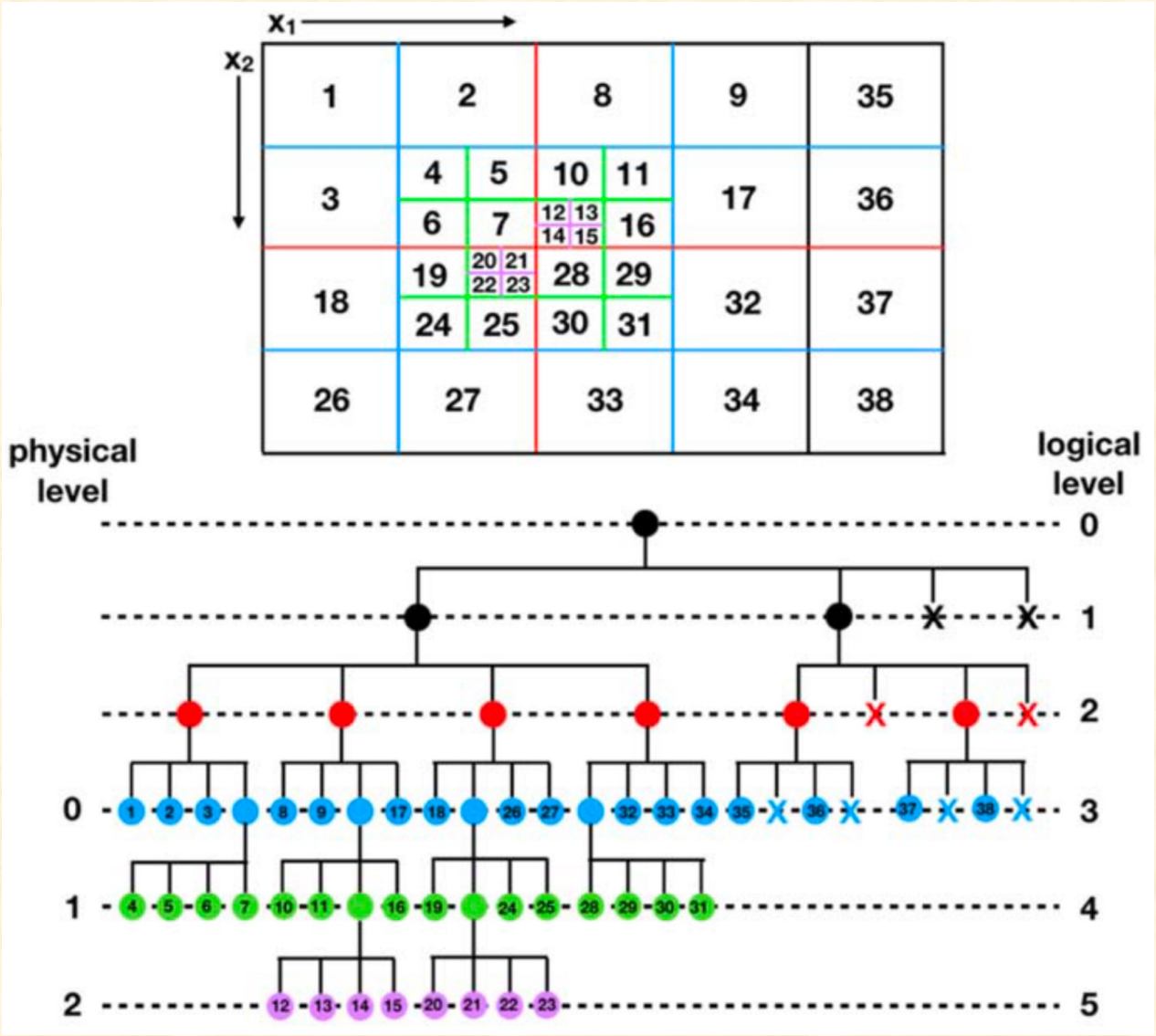
Block based

Athena++/AthenaK
FLASH
ENZO-E

Cell based

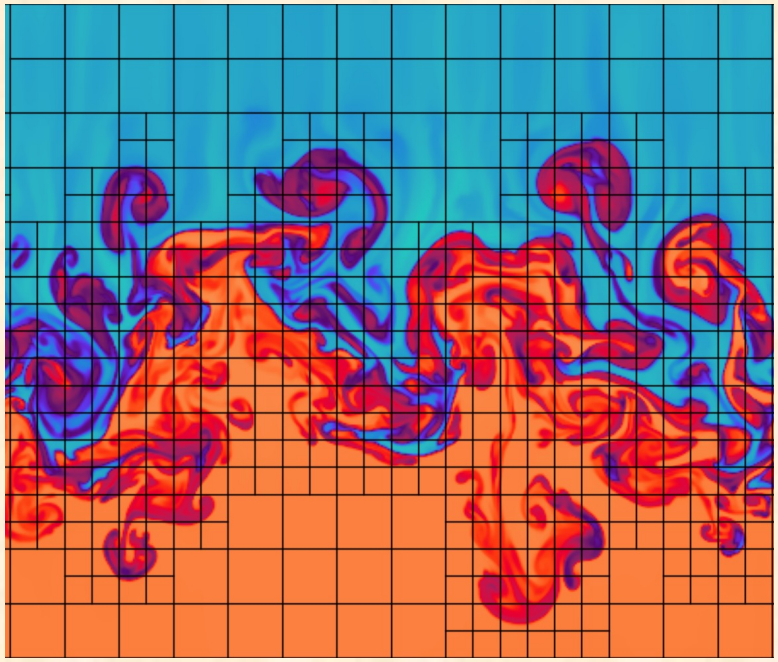
RAMSES
ART

Quad(oct)-tree in 2D(3D) is especially useful for organizing blocks with AMR.



Refinement criteria

Virtually any criterion can be used to flag MeshBlocks for refinement or de-refinement, e.g.

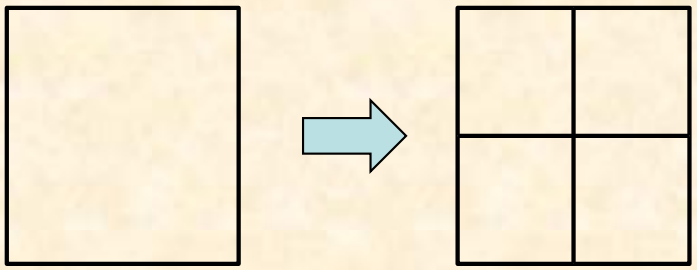


- Thresholds on conserved or primitives
- Thresholds on 1st or 2nd derivatives
- Thresholds on any function of conserved or primitives
- Etc., etc.

Solution can change depending on refinement criteria. Must be very careful results are not affected.

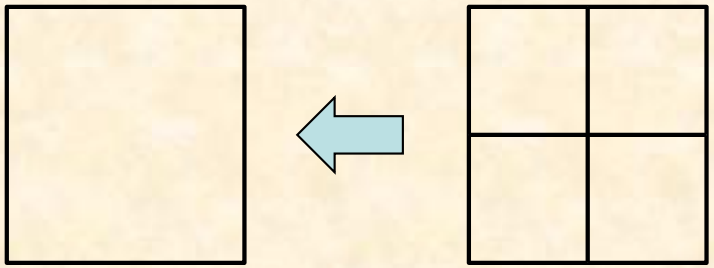
Prolongation and restriction

When a refined region is created, variables must be interpolated to finer mesh: ***prolongation***.



Adopt similar monotonic interpolation methods used for reconstruction (PLM, PPM). Special interpolation needed for magnetic field to enforce dib(B)=0

When a refined region is destroyed, variables must be interpolated to coarser mesh: ***restriction***.

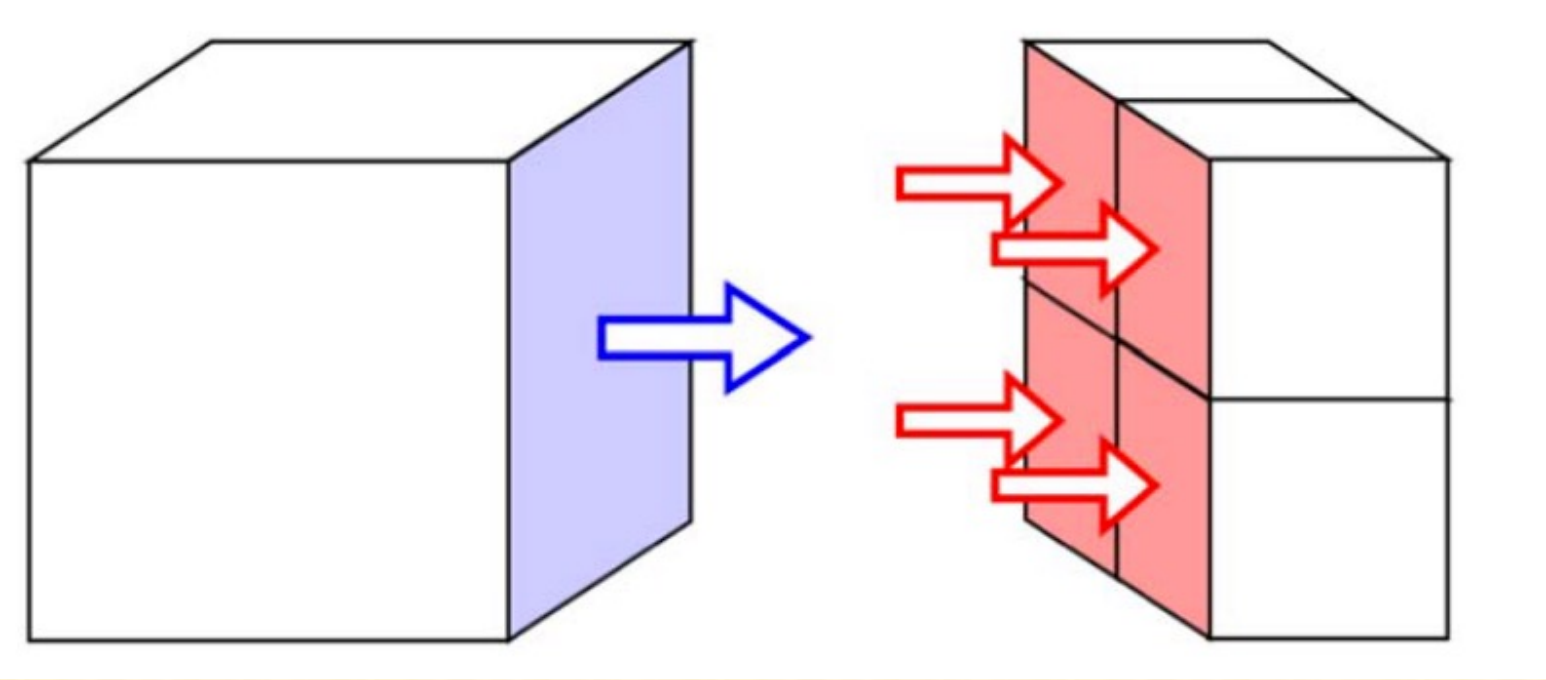


Simple conservative averaging can be used.

Prolongation and restriction also required to load ghost cells at fine/coarse boundaries.

Communicating ghost cells for cell-centered variables with AMR becomes much more complicated.

In addition, flux correction at fine/coarse boundaries is needed with AMR to conserve volume integrals of **U**. For example (cell-centered variables):



Additional corrections needed for electric fields (fluxes of face-centered variables).

AMR in parallel: load balancing

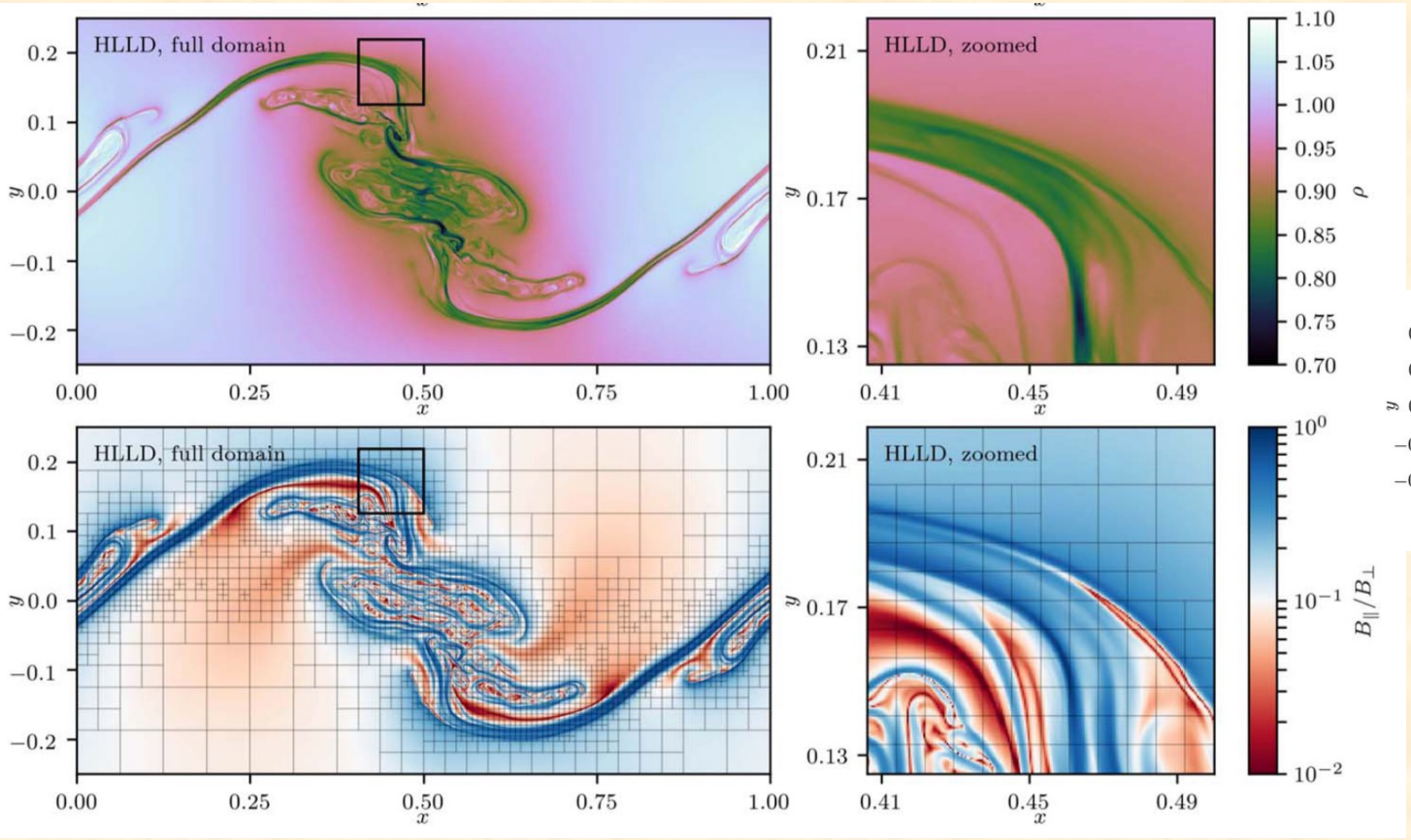
With distributed memory parallel applications, important that all MPI ranks have the same amount of work to do: ***load balancing.***

This is easily achieved by ensuring each rank has the same number of parallel domains (MeshBlocks).

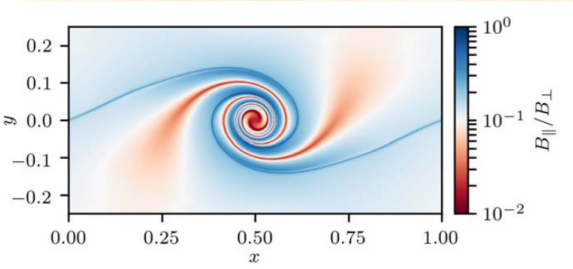
But with AMR, the number of MeshBlocks changes with time. In this case, MeshBlocks must be shuffled between ranks to ensure load balancing.

With Z-ordering, each MPI rank must have a continuous range of MeshBlocks ids.

Should we expect solutions on an AMR mesh to be identical to those on a uniform grid?



MHD KHI with AMR grid solution



Uniform grid solution

Physics Solvers

Wide variety of physics solvers implemented in AthenaK is one of its strengths:

- NR/SR/GR Hydro and MHD
- Diffusion processes: conduction, viscosity, resistivity
- Ambipolar diffusion
- Ion-neutral fluids
- Self-gravity (Velasco)
- Numerical Relativity (Zhu)
- GRMHD in dynamical spacetimes (Fields)
- Relativistic radiation transport (Zhang)
- Particles (Wong)

Plus much more being implemented

Physics Solvers

Wide variety of physics solvers implemented in AthenaK is one of its strengths:

- NR/SR/GR Hydro and MHD
- Diffusion processes: conduction, viscosity, resistivity
- Ambipolar diffusion
- Ion-neutral fluids
- Self-gravity (Velasco)
- Numerical Relativity (Zhu)
- GRMHD in dynamical spacetimes (Fields)
- Relativistic radiation transport (Zhang)
- Particles (Wong)

Plus much more being implemented