GRChombo

Adaptive Mesh Refinement for Numerical Relativity

Saran Tunyasuvunakool

DAMTP, University of Cambridge

In collaboration with:
Pau Figueras, **Markus Kunesch** (DAMTP)
Hal Finkel (Argonne National Laboratory)
Katy Clough, **Eugene Lim** (King's College London)

Resolution vs Memory

- Need high resolution in the "interesting" region
- But also need large domain to reach the gravitational wave zone (and even larger still to isolate spurious boundary effects)
- Generally cannot afford to pay the price of high resolution throughout the entire domain
- Separation in physical lengthscales should be reflected in the computational setup
- Different resolutions for different scales interest!

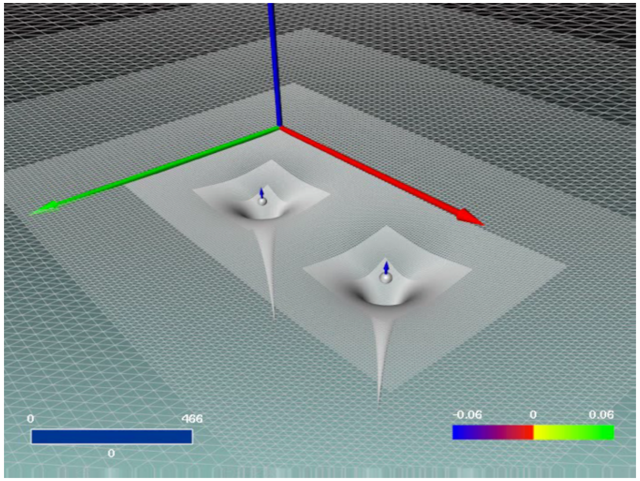
Moving-box mesh refinement

- User specify the size and initial position of a hierarchy of nested boxes with progressively higher resolutions

- Boxes may move around, either by manually specifying a trajectory or tracking some features, but cannot be generally created or change in size or shape

- Very successful in e.g. binary black hole simulations

- Can be more efficient computationally compared to fully-flexible AMR



[Image credit: J. Seiler]

Higher dimensions

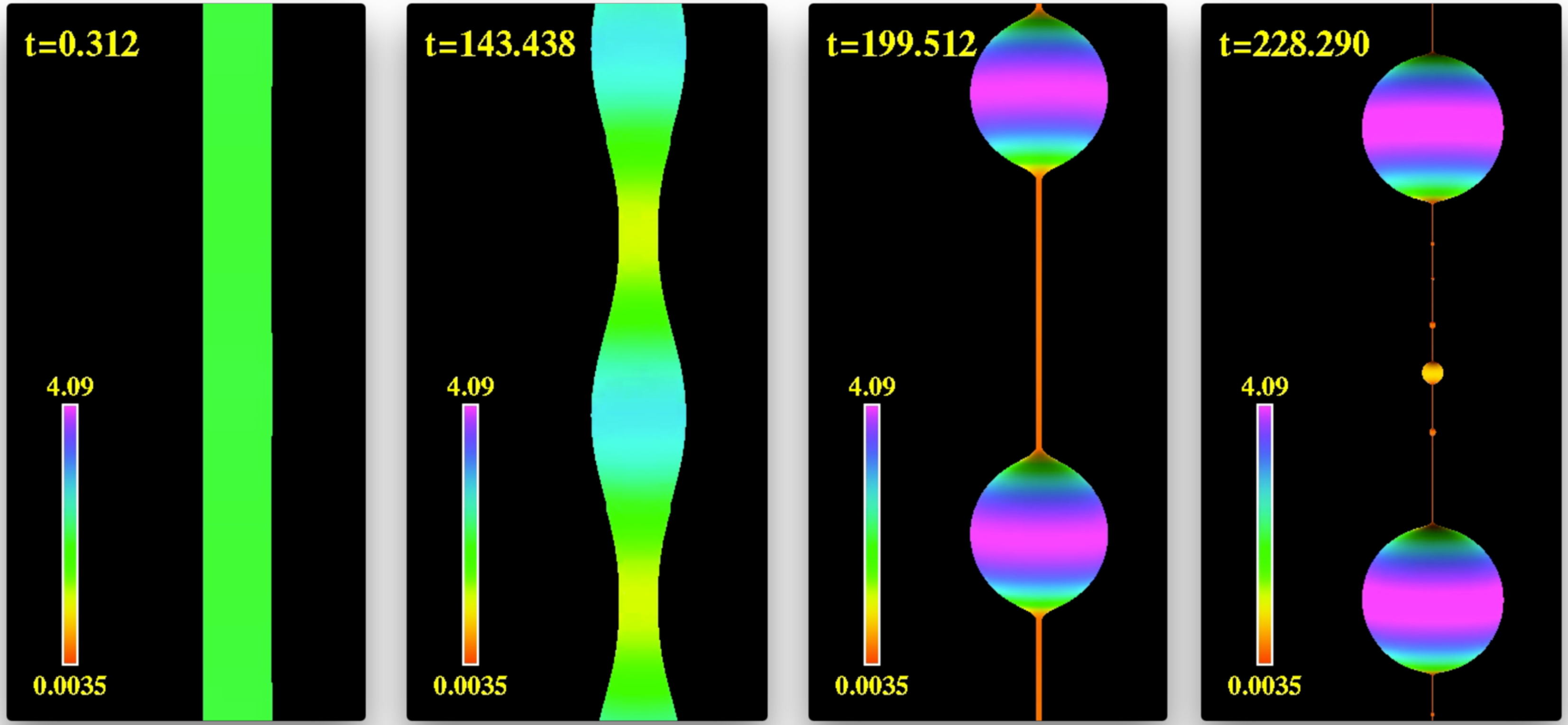
- GR is richer when number of dimensions is > 4 and/or boundary conditions other than asymptotic flatness
- In D ≥ 5, black holes need not have spherical topology: black rings [Emparan and Reall]
- In D ≥ 6, fast-rotating black holes no longer becomes extremal: ultraspinning regime exhibits new instabilities [Myers, Emparan, Shibata, Yoshino]
- Naked singularity formation in asymptotically-KK spacetimes [Gregory and LaFlamme, Lehner and Pretorius]

Adaptive mesh refinement

- More complicated GR setups can have dynamically emerging lengthscales at hard-to-predict times and locations
- Manually pausing and inserting boxes to resolve all of these new features is infeasible
- Nested-box structure inadequate for more complicated topologies, e.g. a torus or a shell
  - Technically, one can still implement FMR with nontrivial topology, but any simplification in programming is quickly offset by the complexity in the manual setup!

PAMR/AMRD [Pretorius]

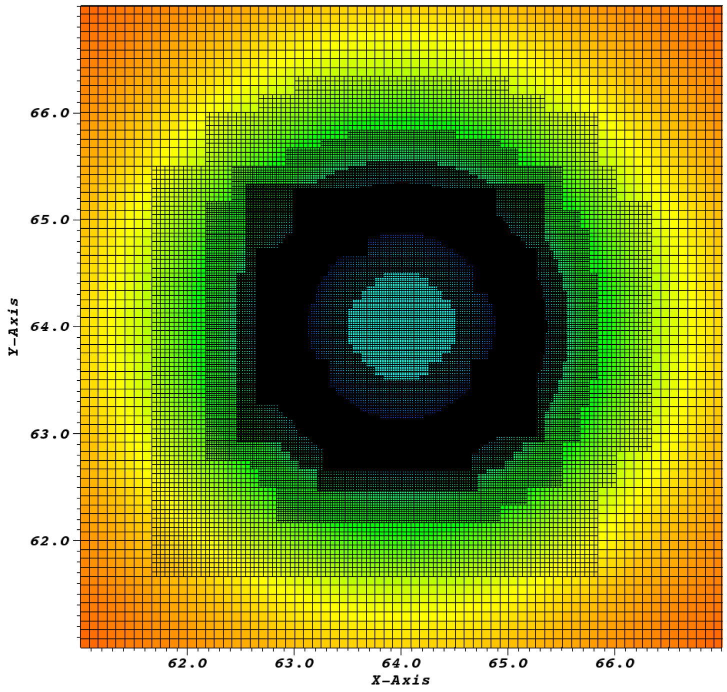
Endpoint of the Gregory-LaFlamme instability [Lehner and Pretorius]

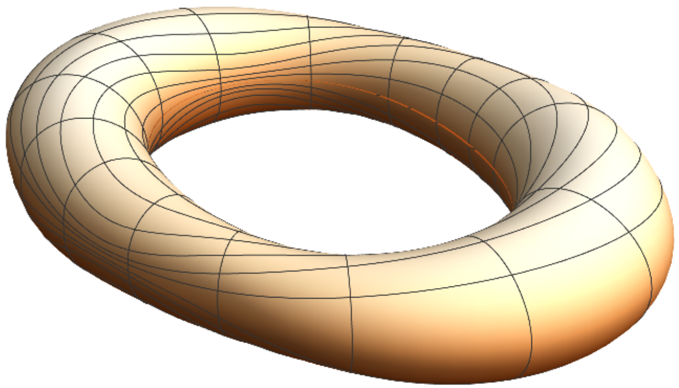


GRChombo

Dynamical evolution of black ring instability in D=5

[Figueras, Kunesch, ST (in progress)]

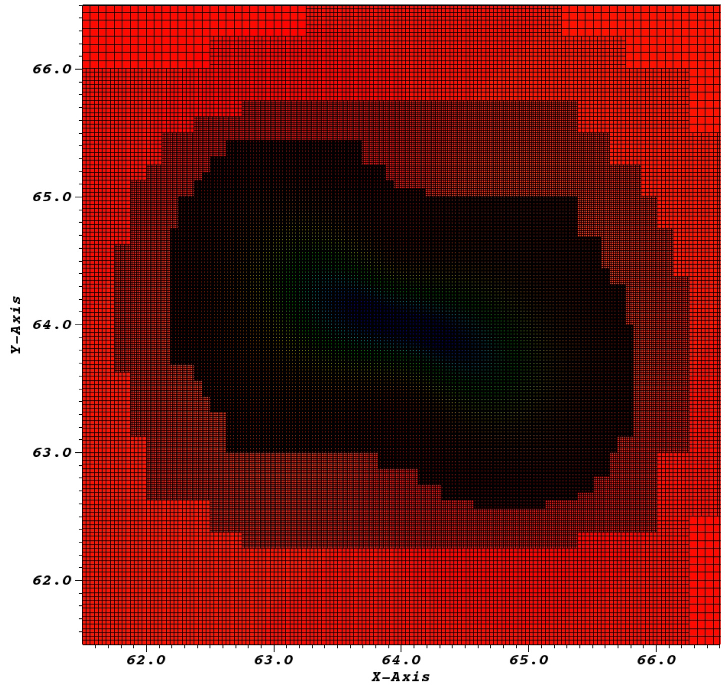


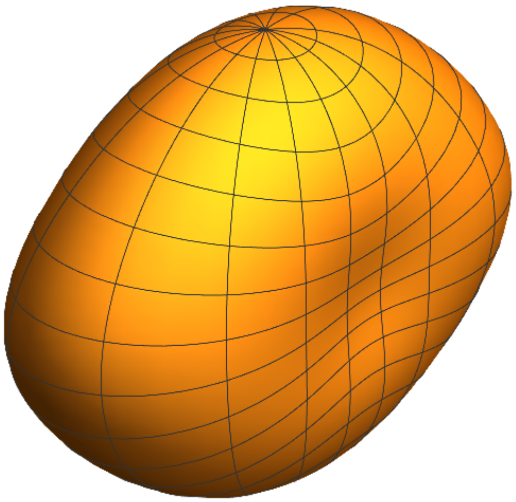


GRChombo

Bar-mode instability of ultraspinning 6D Myers-Perry black hole

[Shibata and Yoshino, Figueras, Kunesch, ST (in progress)]

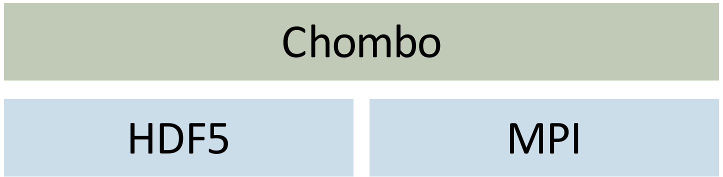




Chombo

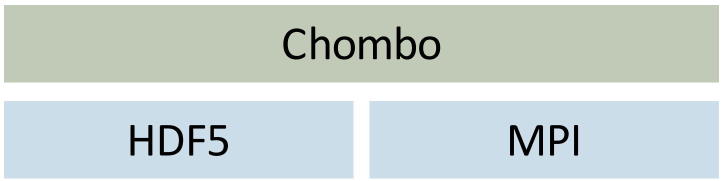
General purpose AMR library from Lawrence Berkeley (LBL)

- Provides building blocks for implementing AMR for structured-grid PDE problems
- Written in (partially templated) C++ and utilises MPI for parallelism
- Produces HDF5 output with built-in data on the mesh structure, which is understood by Paraview and VisIt
- Very easy to build!



Chombo

- Provided classes can be categorised into roughly three layers
- **Layer 1**: distributed data structures (set ops, distribution, synchronisation, load balancing)
- **Layer 2**: interpolation, ghost cell filling, flux matching
- **Layer 3**: Berger-Oliger time subcycling, elliptic solver
- Physics implemented by subclassing "AMRLevel" and overloading functions to calculate RHS, tag cells for refinement, exchange data between levels, etc.



AMRLevel

Public Member Functions

	AMRLevel ()
virtual	~AMRLevel ()
virtual void	define (AMRLevel *a_coarser_level_ptr, const Box &a_problem_domain, int a_level, int a_ref_ratio)
virtual void	define (AMRLevel *a_coarser_level_ptr, const ProblemDomain &a_problem_domain, int a_level, int a_ref_ratio)
virtual Real	advance ()=0
virtual bool	convergedToSteadyState ()
virtual void	postTimeStep ()=0
virtual void	tagCells (IntVectSet &a_tags)=0
virtual void	tagCellsInit (IntVectSet &a_tags)=0
virtual void	preRegrid (int a_base_level, const Vector< Vector< Box > > &a_new_grids)
virtual void	regrid (const Vector< Box > &a_new_grids)=0
virtual void	postRegrid (int a_base_level)
virtual void	initialGrid (const Vector< Box > &a_new_grids)=0
virtual void	postInitialGrid (const bool a_restart)
virtual void	initialData ()=0
virtual void	postInitialize ()=0
virtual void	conclude (int a_step) const

I/O functions

virtual void	writeCheckpointHeader (HDF5Handle &a_handle) const =0
virtual void	writeCheckpointLevel (HDF5Handle &a_handle) const =0
virtual void	readCheckpointHeader (HDF5Handle &a_handle)=0
virtual void	readCheckpointLevel (HDF5Handle &a_handle)=0
virtual void	writePlotHeader (HDF5Handle &a_handle) const =0
virtual void	writePlotLevel (HDF5Handle &a_handle) const =0
virtual void	writeCustomPlotFile (const std::string &a_prefix, int a_step) const

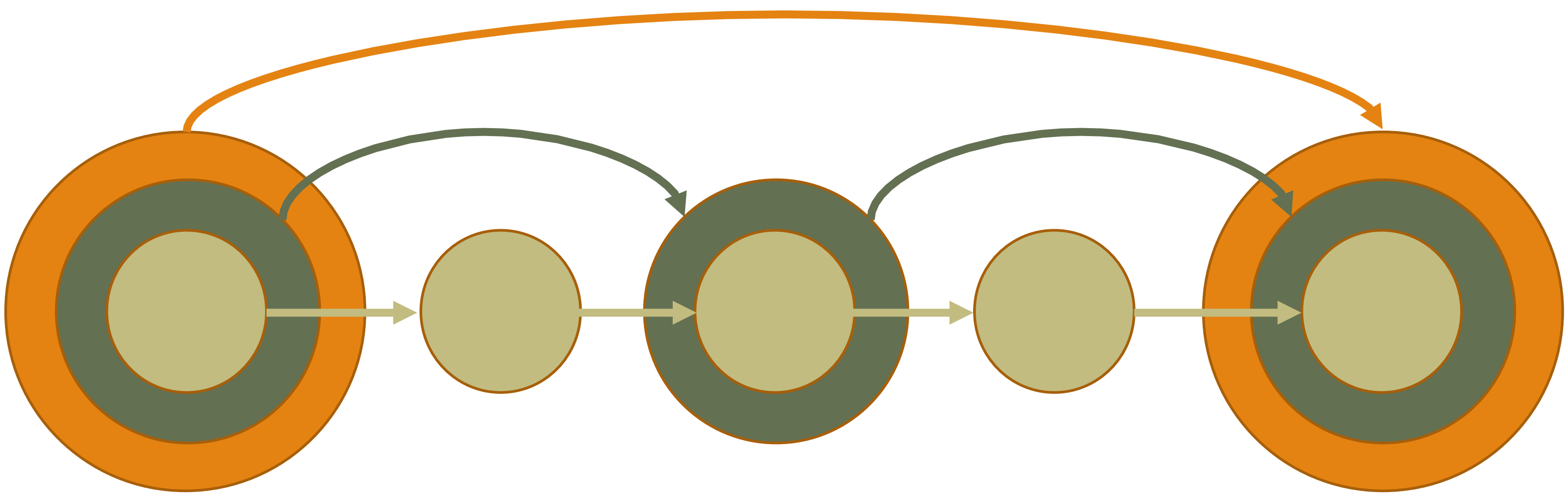
Berger-Oliger AMR

Box-structured local refinement [Berger and Oliger, 1984]

- Estimate local error at each cell on the mesh
- Cells whose error exceeds some threshold are tagged
- Arrange tagged cells into a collection of boxes
- Populate the boxes with a new mesh at a higher resolution
- Do the same thing on the new mesh until no more cells are tagged, or maximum number of levels is reached

Subcycling in time

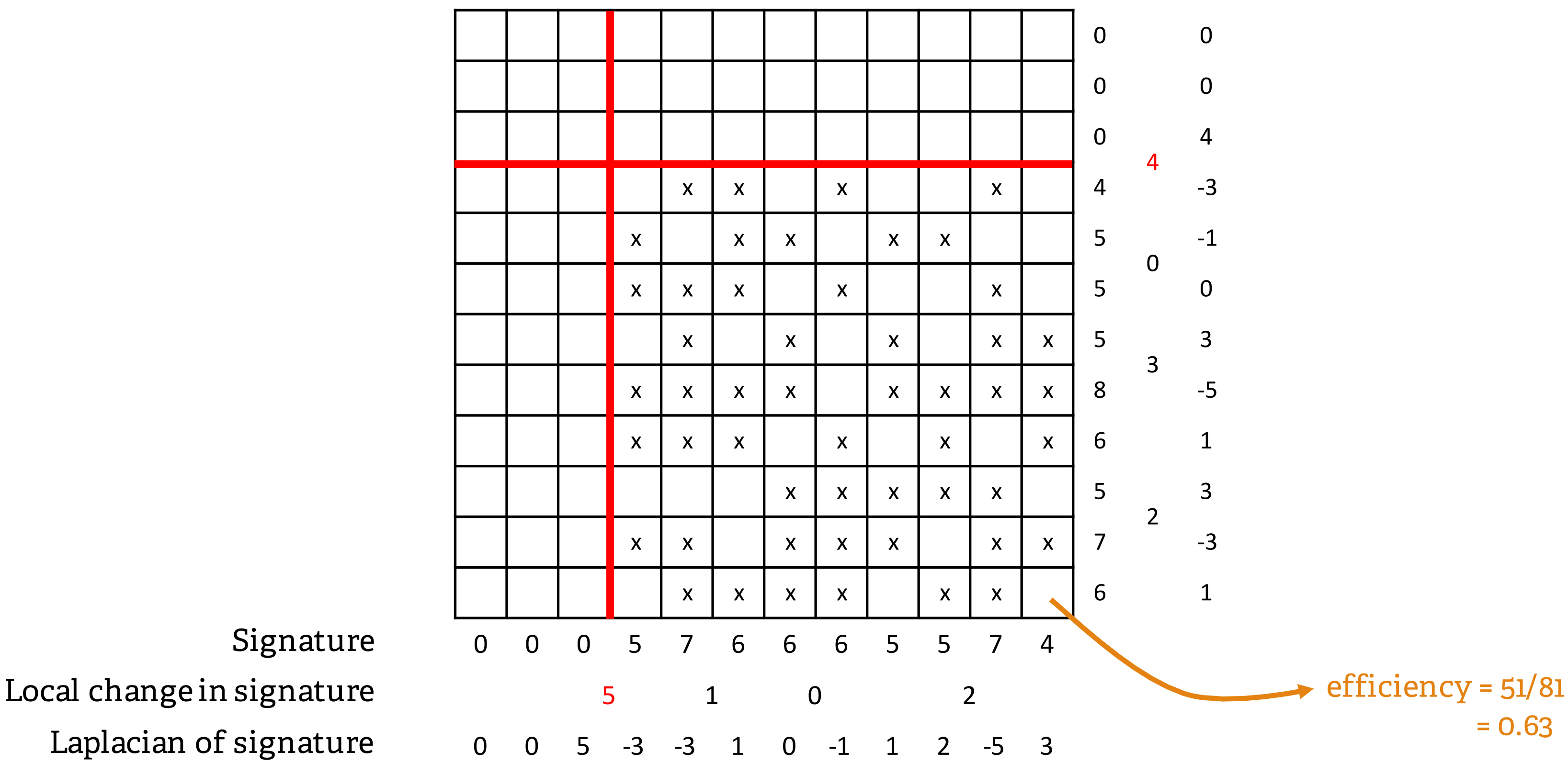
In order to maintain the CFL condition, we must also "refine in time" each time we refine in space. *(Applicable to both FMR and AMR)*



Level 2	$t = 0$	$t = 0.25$	$t = 0.5$	$t = 0.75$	$t = 1$
Level 1	$t = 0$		$t = 0.5$		$t = 1$
Level 0	$t = 0$				$t = 1$

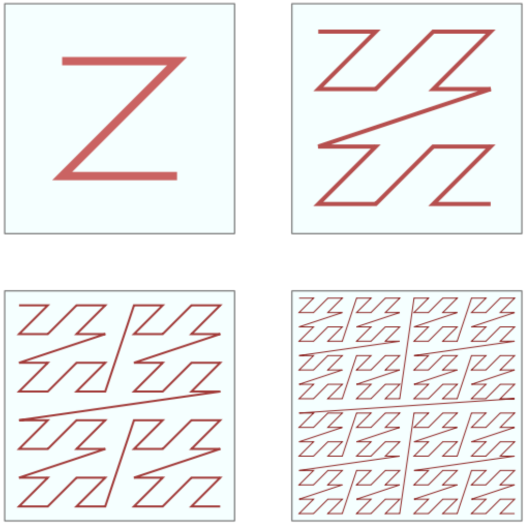
Grid generation

Optimum clustering of points into boxes [Berger and Rigoutsos, 1991]



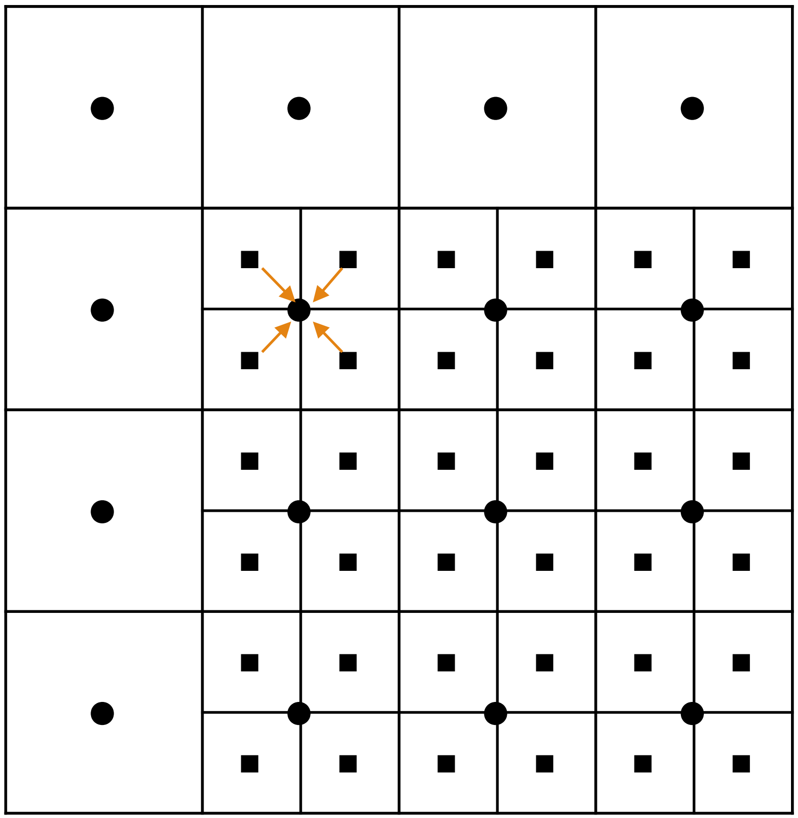
Load balancing

- Morton-order the boxes

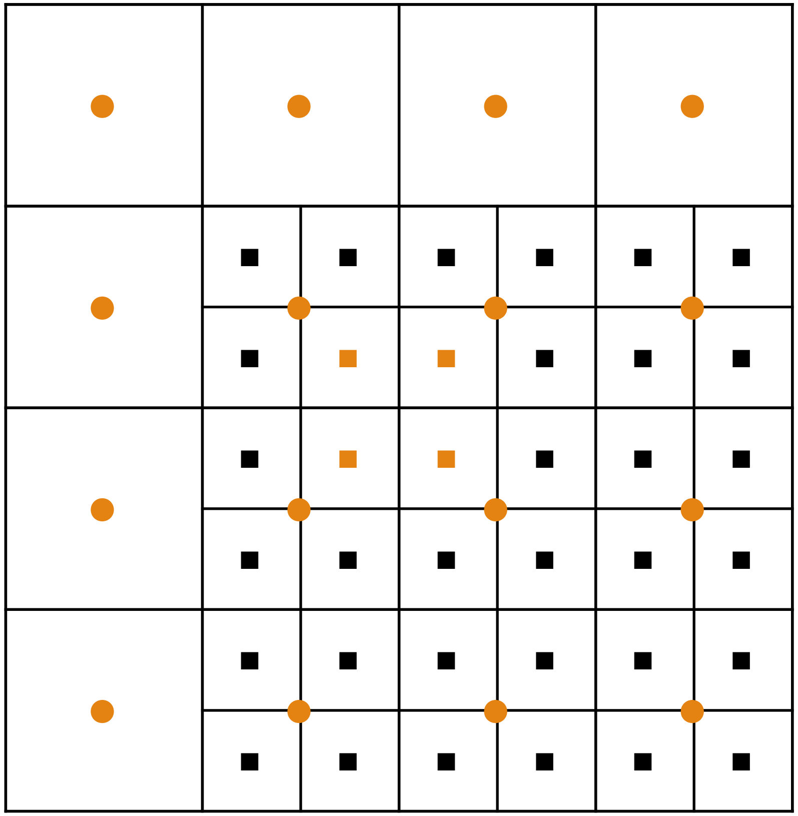


- Apply the knapsack algorithm

Coarse averaging



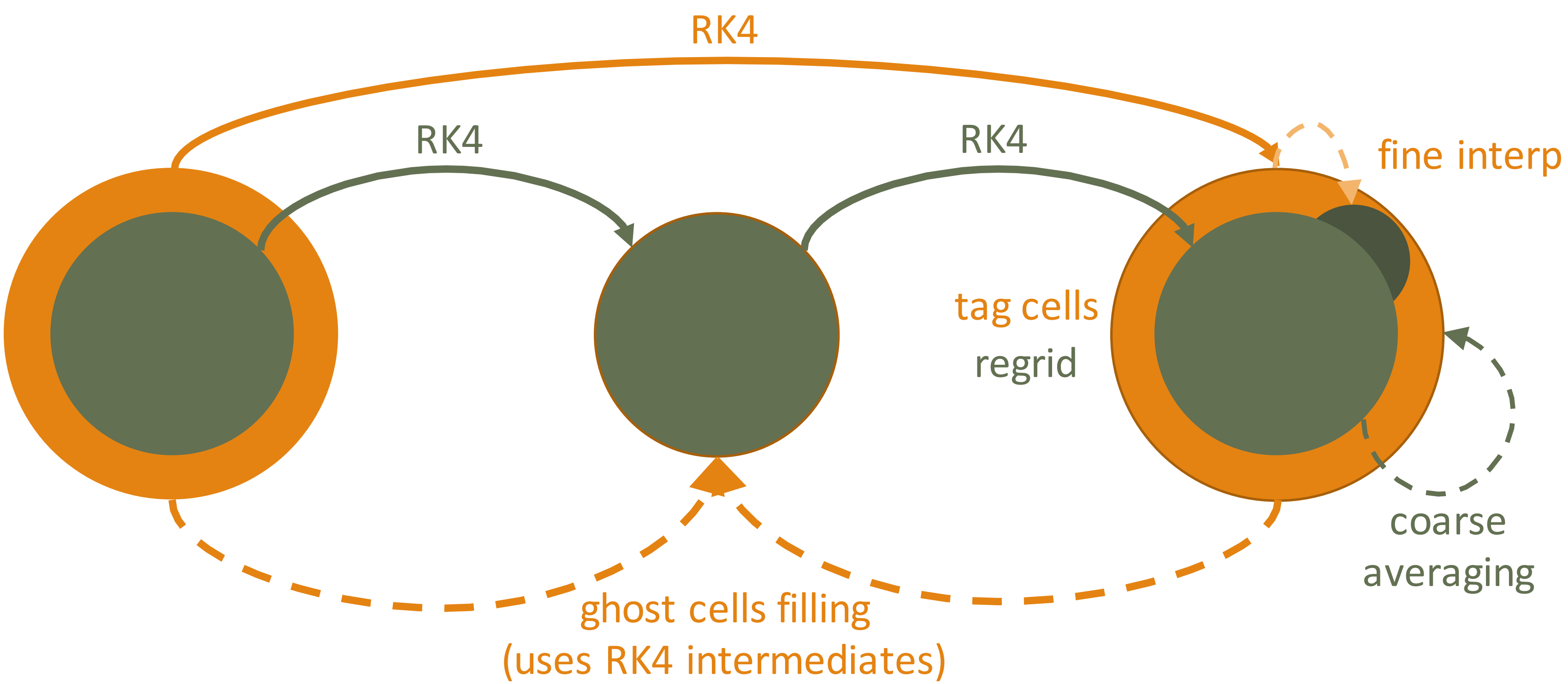
Fine interpolation



Ghost cell filling

- Chombo provides “patcher” classes to fill ghost cells between each subcycle in time
- Runge-Kutta already uses time interpolation internally in each step to achieve higher-order convergence in time
- Chombo provides an RK4 stepper which stores these intermediate values, then use them to provide the correct ghost value at the subcycled times for the next level
- Also simultaneously performs fine-interpolation in space

A typical timestep



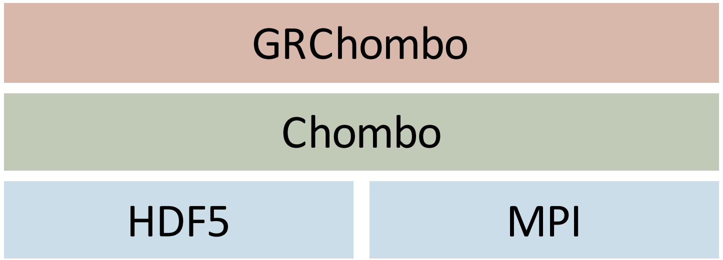
GRChombo

- Implements BSSN/CCZ4 equations on top of Chombo
  - + “modified cartoon” terms D>4 simulations
  - + scalar field

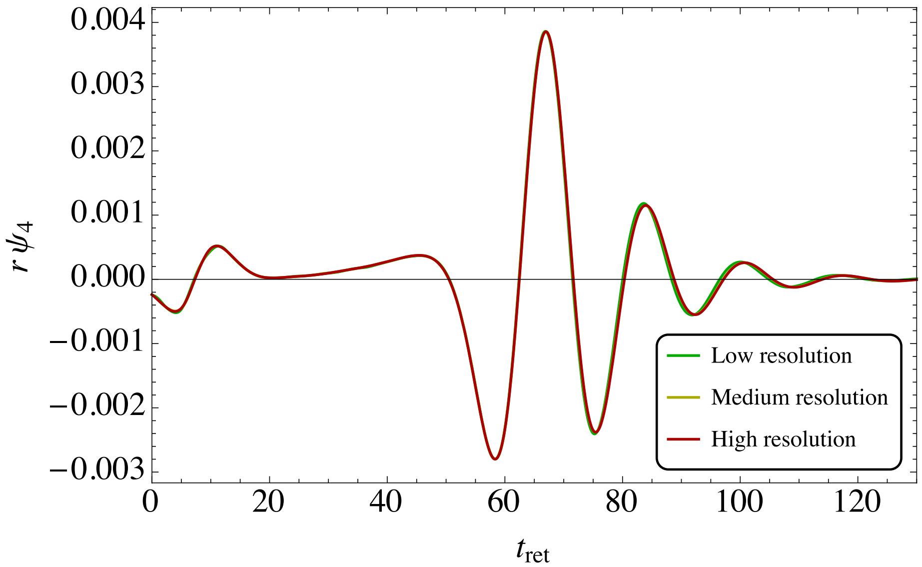
- Fourth-order finite differences in space (centered stencils for most terms, upwinded stencils for shift-advected terms)

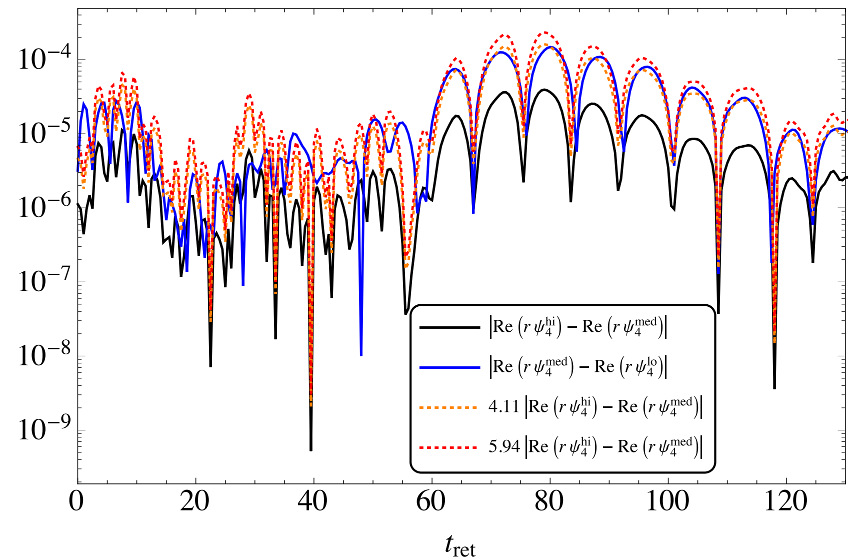
- Explicit fixed-step RK4 in time

- No proper initial data solver yet, but Chombo does come with “AMRElliptic” capabilities

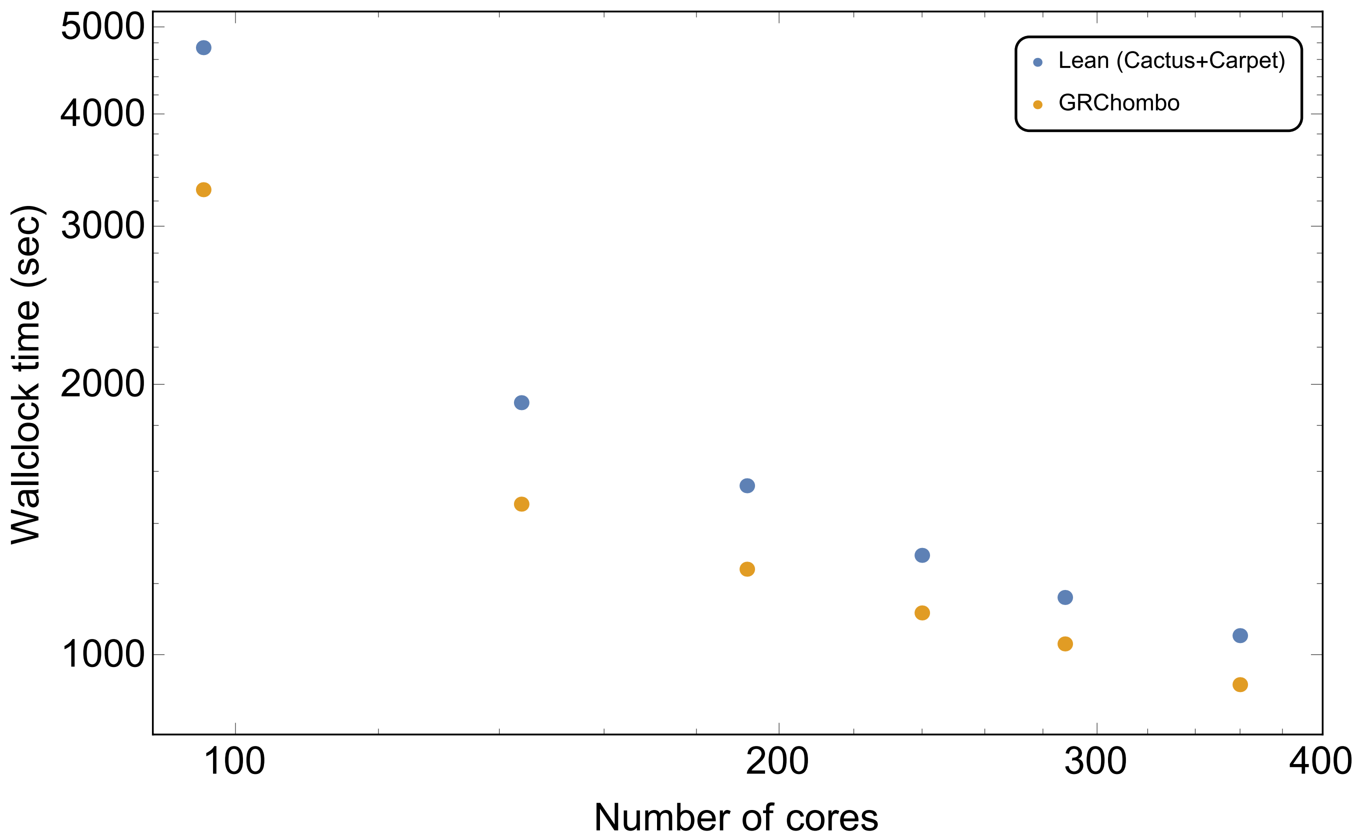


Convergence test

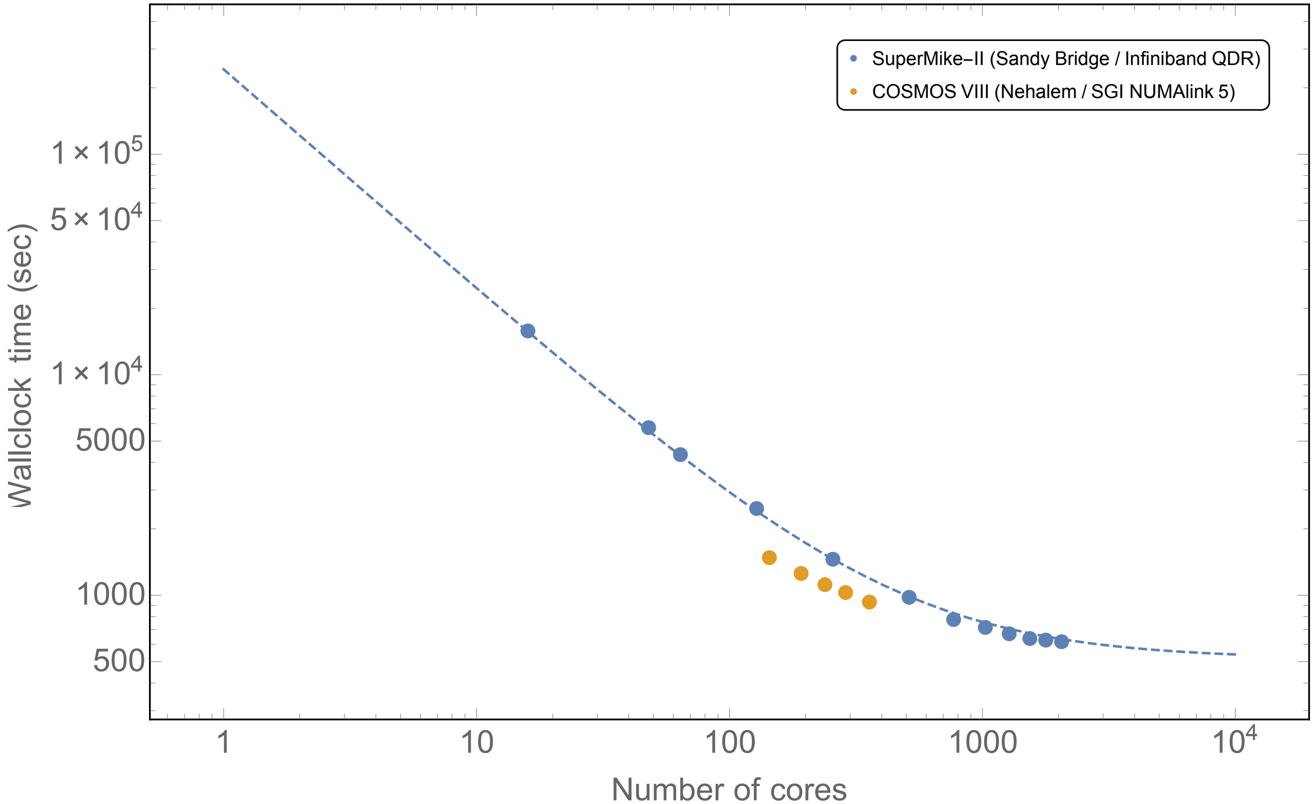




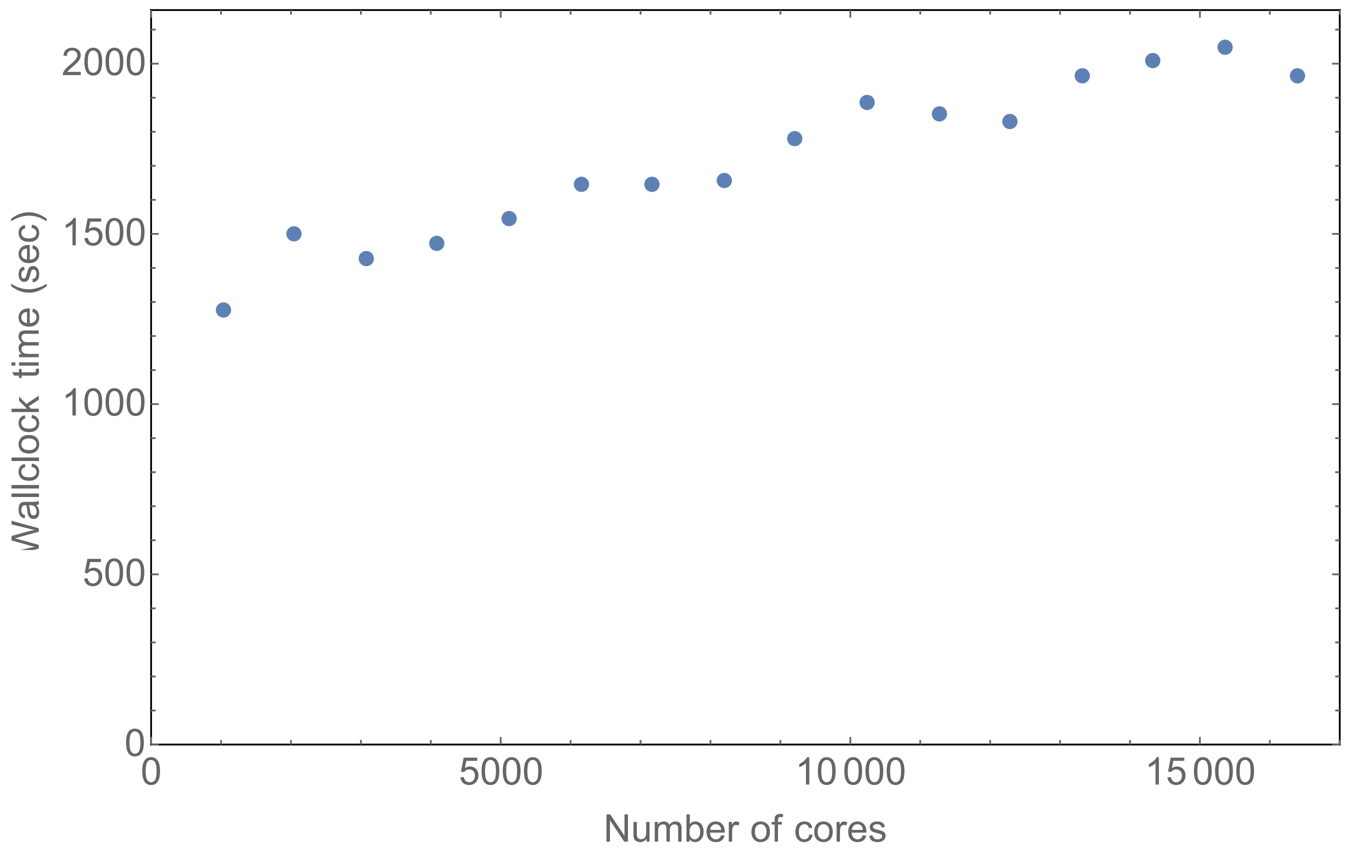
Strong scaling



Strong scaling

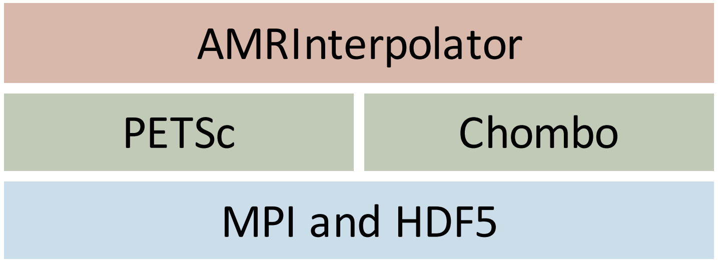


Weak scaling



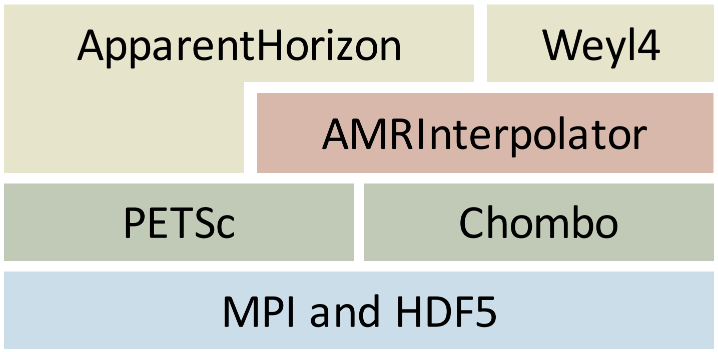
Analysis tools

- We have developed some essential analysis tools for numerical GR problems
- Most analysis tasks involve interpolating data onto a secondary grid; AMR is not usually necessary on this secondary grid
- **AMRInterpolator** uses PETSc to set up a distributed uniform grid and has custom MPI code to perform all-to-all query of interpolated data using Lagrange polynomials (arbitrary order)
- Take data from the finest available level for at any given point



Analysis tools

- AMRInterpolator is modular: specific analysis tasks can be easily built on top of it
- Calculation of ψ_4 on a sphere of any given radius
- Apparent horizon finder via Newton's method; supports both spherical or toroidal horizon topologies
- New AH coordinates can be implemented very easily (code is templated on the transformation functions)



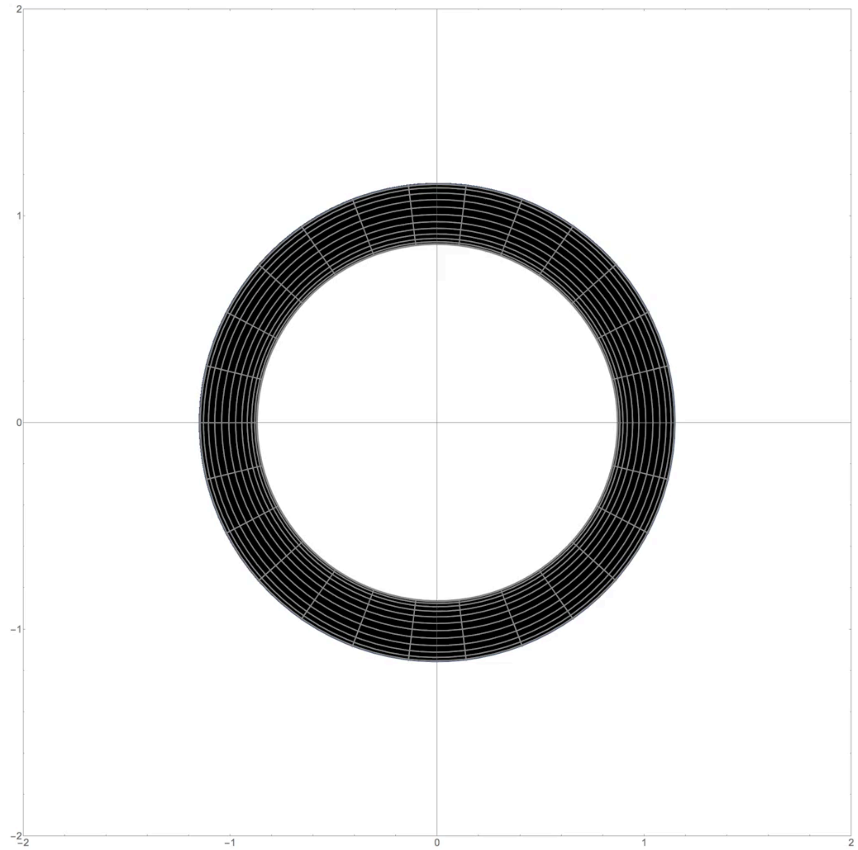
Upcoming development

- Code refactoring: now that we know it can work well, let's make it more maintainable and extensible also
- Profiling and optimisation: can we benefit from e.g. Intel Xeon Phi? More and more clusters are relying on these accelerators to provide the bulk of their FLOPS
- Dynamical excision? In principle can piggyback on the AMR tagging workflow, but interlevel and ghost filling is problematic
- Public code release planned

```
      subroutine GETBSSNCRHSF(
c     {{{1
c    subroutine input {{{2
     &      CHF_FRA1[dchidt],
     &      CHF_FRA[dhdt],
     &      CHF_FRA1[dhwwdt],
     &      CHF_FRA1[dKdt],
     &      CHF_FRA[dAdt],
     &      CHF_FRA1[dAwwdt],
     &      CHF_FRA1[dThetadt],
     &      CHF_FRA[dGamma1dt],
     &      CHF_FRA1[dlapsedt],
     &      CHF_FRA[dshiftdt],
     &      CHF_FRA[dBdt],
     &      CHF_FRA1[dGphidt],
     &      CHF_FRA1[dGpidt],
     &      CHF_CONST_FRA1[chi],
     &      CHF_CONST_FRA[h],
     &      CHF_CONST_FRA1[hww],
     &      CHF_CONST_FRA1[K],
     &      CHF_CONST_FRA[A],
     &      CHF_CONST_FRA1[Aww],
     &      CHF_CONST_FRA1[Theta],
     &      CHF_CONST_FRA[Gamma1],
     &      CHF_CONST_FRA1[lapse],
     &      CHF_CONST_FRA[shift],
     &      CHF_CONST_FRA[B],
     &      CHF_CONST_FRA1[Gphi],
     &      CHF_CONST_FRA1[Gpi],
     &      CHF_CONST_REAL[dx],
     &      CHF_CONST_REAL[chi_gamma],
     &      CHF_CONST_REAL[LapseCoeff],
     &      CHF_CONST_INT[ShiftBCoeff],
     &      CHF_CONST_REAL[LapseAdvectionCoeff],
     &      CHF_CONST_REAL[ShiftAdvectionCoeff],
     &      CHF_CONST_REAL[ShiftGammaCoeff],
     &      CHF_CONST_REAL[BetaDriver],
     &      CHF_CONST_REAL[SpatialBetaDriverRadius],
     &      CHF_CONST_REAL[dampk1],
     &      CHF_CONST_REAL[dampk2],
     &      CHF_CONST_REAL[GammaShift],
     &      CHF_CONST_INT[covariantZ4],
     &      CHF_CONST_REAL[sigma],
     &      CHF_CONST_REAL[epsXtraDiss],
     &      CHF_CONST_REAL[rDiss],
     &      CHF_CONST_REAL[z0Diss],
     &      CHF_CONST_REAL[nu],
     &      CHF_CONST_REAL[Rring],
     &      CHF_CONST_REAL[fracInner],
     &      CHF_CONST_REAL[fracOuter],
     &      CHF_CONST_REAL[dissCutInner],
     &      CHF_CONST_REAL[dissCutOuter],
     &      CHF_CONST_REAL[centerx],
     &      CHF_CONST_REAL[centery],
     &      CHF_CONST_REAL[centerz],
     &      CHF_CONST_INT[dims],
     &      CHF_BOX[box])
c    }}}
```

```
void calculate_bssn(
    const struct bssn& in_fab,
    struct bssn& out_fab,
    const struct box& params)
```

Physics work in progress



Physics work in progress