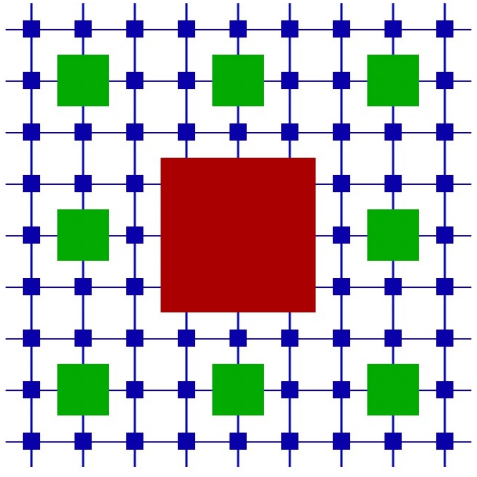
Using CarpetX: A Guide for Early Adopters

Erik Schnetter, Perimeter Institute

Einstein Toolkit Seminar, 2021-03-18



A new driver for the Einstein Toolkit

- Based on AMReX framework (github.com/AMReX-Codes/amrex)
  - Many users, independently funded
  - Cell/face/edge/vertex-centred grids, refluxing (for AMR with hydro)
  - AMR based on local criteria
  - Support for accelerators (e.g. CUDA)
  - Elliptic solvers
  - Scalable (exascale)
- Other goals:
  - Simplify scheduling, prevent errors
  - Improve I/O performance

A new driver for the Einstein Toolkit

- Based on AMReX framework (github.com/AMReX-Codes/amrex)
  - Many users, independently funded
  - Cell/face/edge/vertex-centred grids, refluxing (for AMR with hydro)
  - AMR based on local criteria
  - Support for accelerators (e.g. CUDA)
  - Elliptic solvers
  - Scalable (exascale)
- Other goals:
  - Simplify scheduling, prevent errors
  - Improve I/O performance

[done] [in progress] [unknown]

What else is working today:

- I/O, both HDF5 and ASCII, reading/writing HDF5 (using Silo format)
  - no checkpointing/recovery yet
- Reductions, Interpolations
  - no hyperslabbing yet
- Periodic and reflecting symmetries
  - no rotating symmetries yet
- Adaptive mesh refinement with high order prolongation/restriction
- WaveToy (basic scalar wave)
- HydroToy (basic non-relativistic Euler equations)
- Z4c (full Einstein equations)
  - No SIMD parallelization yet

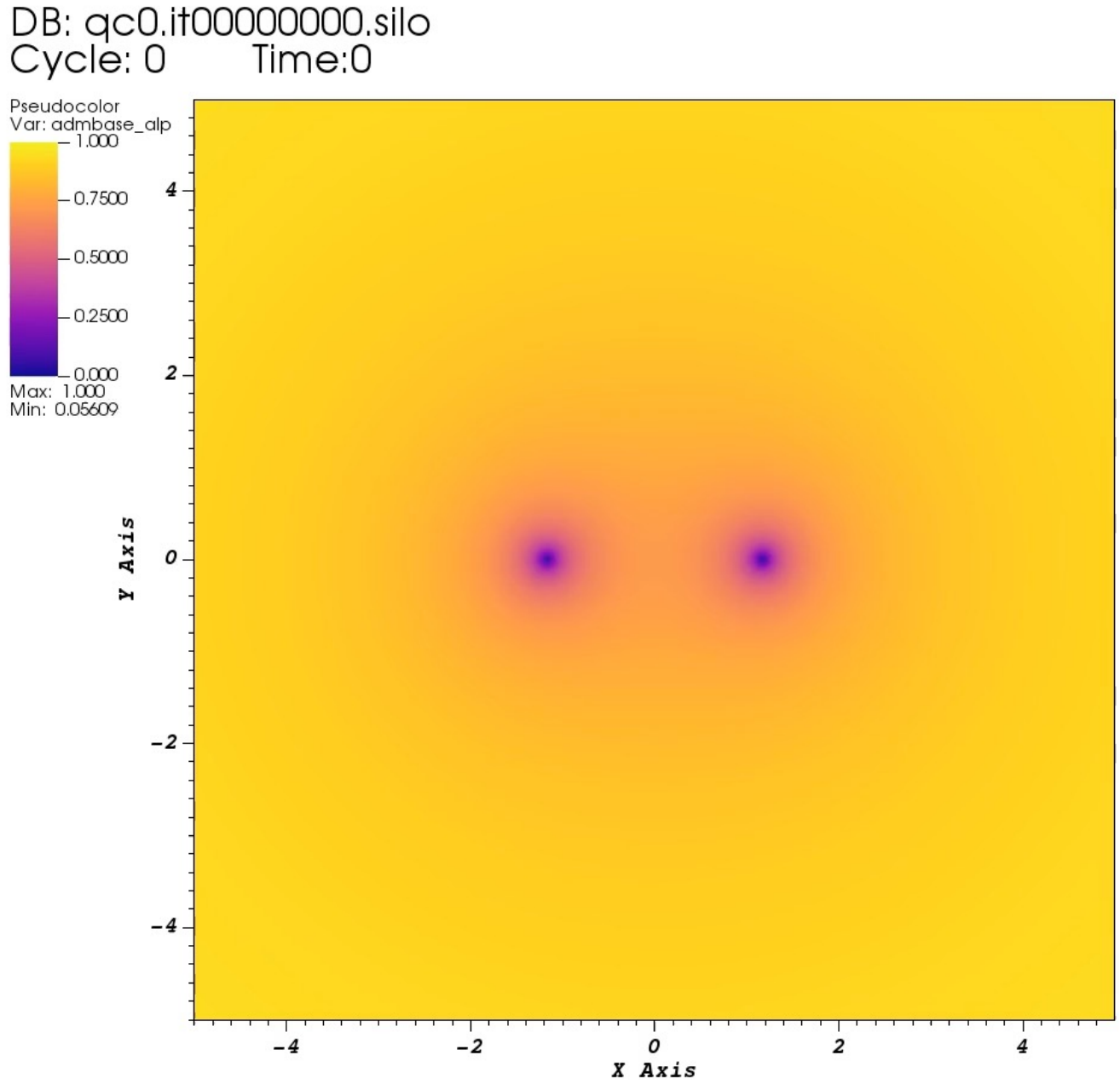
QC-0
(black hole binary)

Large box (+/- 128)

dx[coarse] = 1
dx[fine] = 1/24

4th order everything

~3/4 orbit before merging



Adaptive Mesh Refinement

- CarpetX does not yet support subcycling in time
  - All refinement levels advance simultaneously (efficient, simple)
  - Prolongation and restriction happen when synchronizing
- Regridding:
  - User code defines *local error* for every grid cell, chooses error threshold
  - AMReX refines any block (e.g. 8x8x8) where any cell has an error too high
- No multi-block systems yet

[Show WaveToyCPU, interface.ccl]

- Discuss:
  - Timelevels
  - “index” tag
  - “rhs” tag
  - “checkpoint” tag

Scheduling

- Initial conditions:
  1. CCTK_BASEGRID
  2. Loop:
     1. Initialisation done?
     2. Regrid
        - CCTK_BASEGRID
        - CCTK_POSTREGRID
     3. CCTK_INITIAL
     4. CCTK_ POSTINITIAL
  3. CCTK_POSTSTEP
  4. CCTK_ANALYSIS
  5. OutputGH

- Evolution:
  1. Evolution done?
  2. Regrid
     - CCTK_BASEGRID
     - CCTK_POSTREGRID
  3. CCTK_PRESTEP
  4. CCTK_EVOL
  5. CCTK_POSTSTEP
  6. CCTK_ANALYSIS
  7. OutputGH

Scheduling

- BASEGRID:
  - Set up constant data (e.g. coordinates)
- INITIAL, POSTINITIAL:
  - Initialise state vector (including boundaries)
  - Define local error for regridding
- EVOL:
  - Evolve state vector (and set boundaries)
- POSTSTEP, ANALYSIS:
  - Calculate ephemeral values (e.g. constraints)
  - Define local error for regridding
- POSTREGRID:
  - Re-apply boundary conditions to state vector

[Show WaveToyCPU, schedule.ccl]

- Discuss:
  - Missing STORAGE statements
  - Using schedule groups
  - ODESolvers_RHS, ODESolvers_PostStep
  - READS, WRITES
  - Sync *after* boundary conditions (prolongation stencils)

Declaring Dependencies in the Schedule

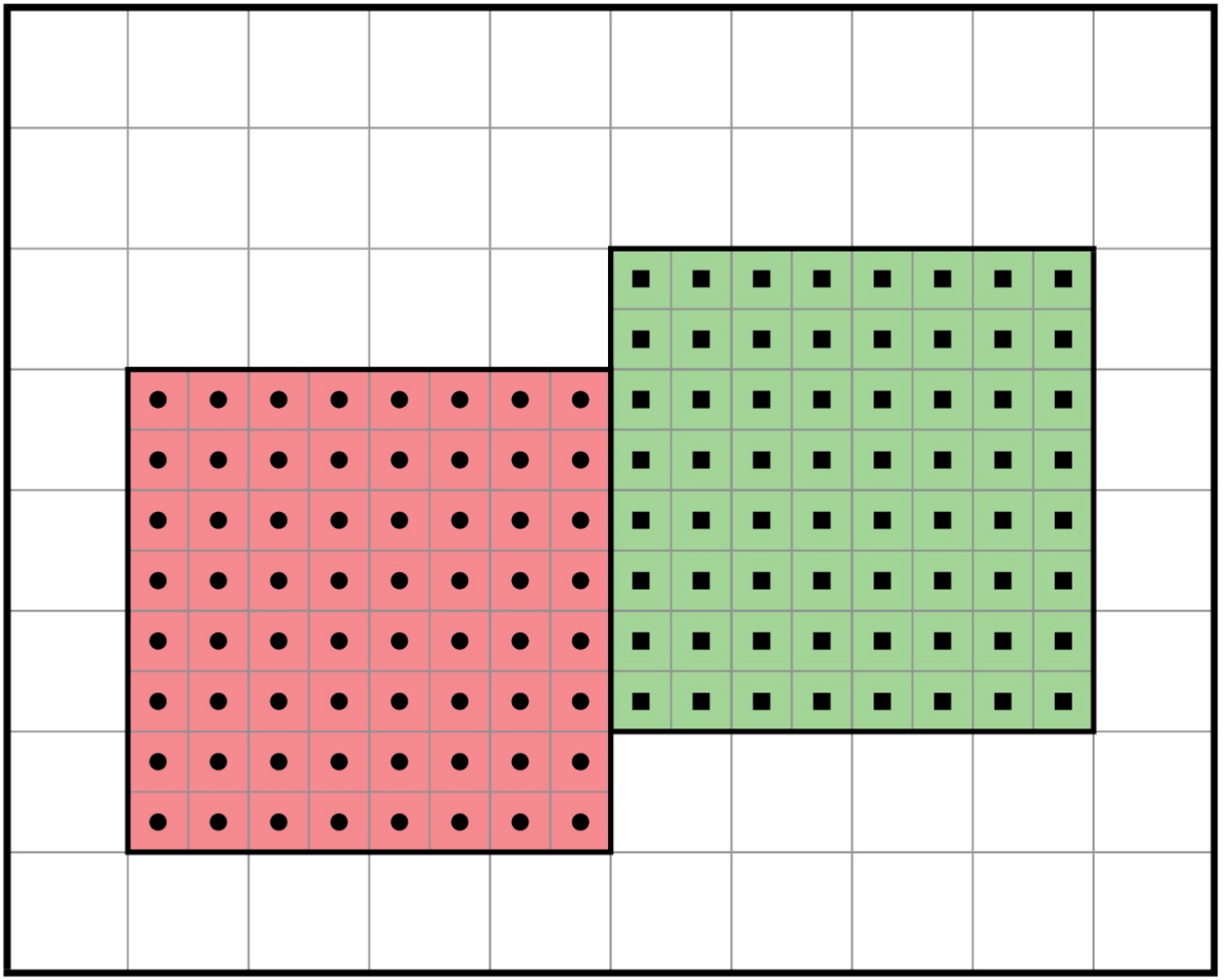
- Each scheduled function needs to declare which parts of which variables it reads or writes
- Regridding, synchronization, prolongation etc. do this implicitly as well
- The driver checks consistency, and flags errors
- The driver cross-checks these declarations via:
  - Undefined variables
  - Const variables
  - Variables set to nan, and checked for nan
  - Checksums of parts of variables
- Also, there are “informative” error messages

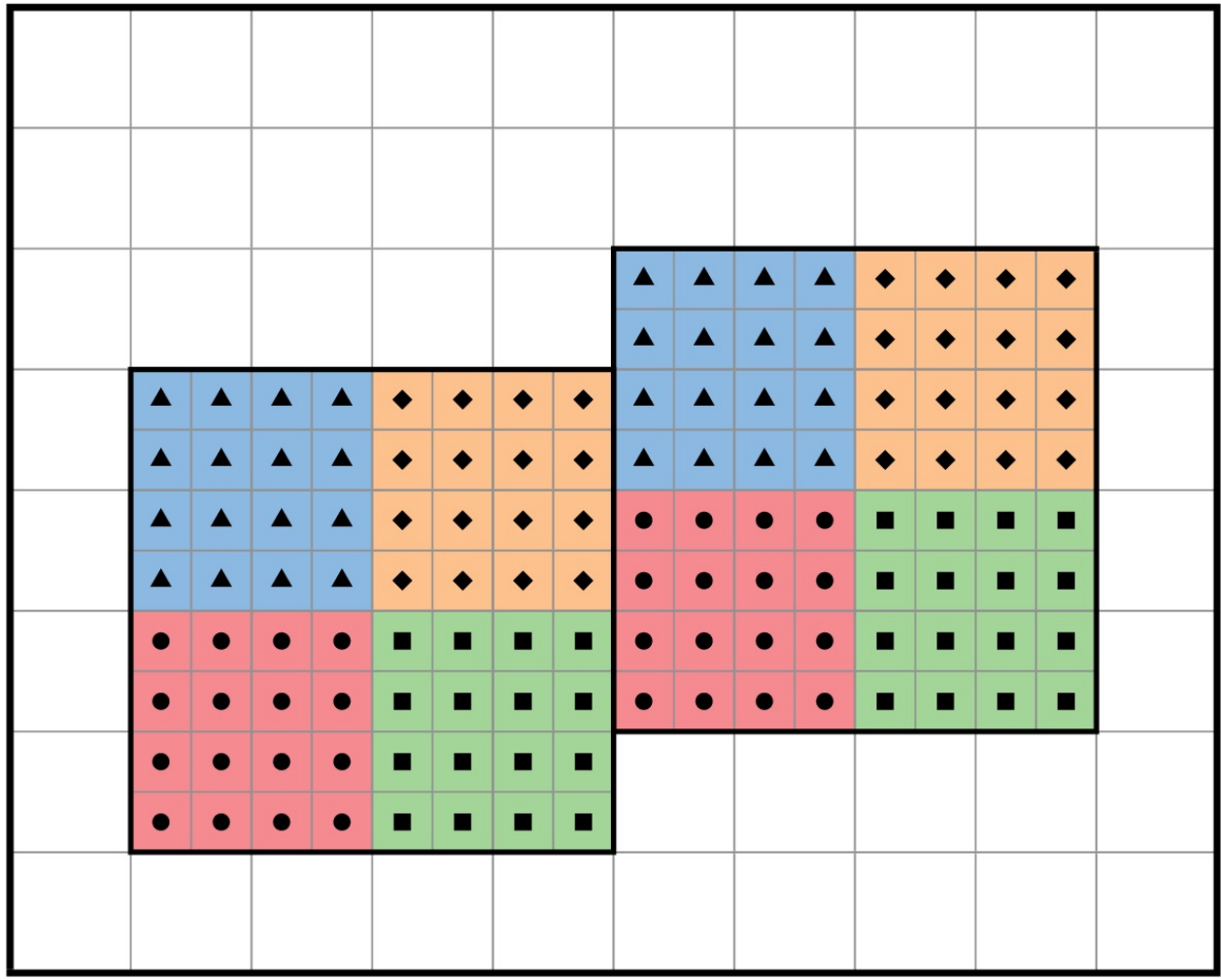
[Show WaveToyCPU, planewave.par; Z4c, qc0.rpar]

- Discuss:
  - Thorn list
  - Cactus::presync_mode = "mixed-error”
  - CarpetX::xmin, ncells
  - CarpetX::periodic, reflection
  - CarpetX::blocking_factor
  - CarpetX::max_grid_size, max_tile_size
  - Regridding
  - ODESolvers
  - I/O

New OpenMP Parallelisation via Tiling

[AMReX documentation, section “MFIter with Tiling”]





Choosing Good Run-Time Parameters

system	time	iter	size	nodes	cores_node	threads	threads_prc	blocking_fa	max_grid_s	max_tile_si	max_tile_si	max_tile_si
symmetry	57.2429	10	256	1	40	40	10	8	128	1024000	4	2
symmetry	57.7137	10	256	1	40	40	5	8	128	1024000	2	4
symmetry	57.9681	10	256	1	40	40	5	8	128	1024000	4	2
symmetry	58.3404	10	256	1	40	40	10	8	128	1024000	2	4
symmetry	59.1469	10	256	1	40	40	10	8	128	1024000	2	2
symmetry	59.453	10	256	1	40	40	5	8	64	1024000	8	2
symmetry	59.6888	10	256	1	40	40	5	8	64	1024000	4	2
symmetry	59.7976	10	256	1	40	40	5	8	128	1024000	2	2
symmetry	60.0603	10	256	1	40	40	5	8	128	1024000	4	4
symmetry	60.1971	10	256	1	40	40	10	8	128	1024000	8	2
symmetry	60.5352	10	256	1	40	40	5	8	64	1024000	2	4
symmetry	61.0143	10	256	1	40	40	20	8	128	1024000	4	2
symmetry	61.0216	10	256	1	40	40	10	8	128	1024000	4	4
symmetry	61.1173	10	256	1	40	40	5	8	64	1024000	4	4
symmetry	61.2639	10	256	1	40	40	10	8	128	1024000	2	8
symmetry	61.65	10	256	1	40	40	5	8	128	1024000	8	2
symmetry	62.3565	10	256	1	40	40	10	8	64	1024000	4	2
symmetry	62.3756	10	256	1	40	40	5	8	64	1024000	2	8
symmetry	62.6931	10	256	1	40	40	5	8	128	1024000	2	8
symmetry	62.7316	10	256	1	40	40	20	8	128	1024000	2	4

Choosing Good Run-Time Parameters

system	time	iter	size	nodes	cores_node	threads	threads_pre	blocking_fa	max_grid_s	max_tile_si	max_tile_si	max_tile_si
symmetry	57.2429	10	256	1	40	4	10	8	128	1024000	4	2
symmetry	57.7137	10	256	1	40	40	5	8	128	1024000	2	4
symmetry	57.9681	10	256	1	40	40	5	8	128	1024000	4	2
symmetry	58.3404	10	256	1	40	40	10	8	128	1024000	2	4
symmetry	59.1469	10	256	1	40	40	10	8	128	1024000	2	2
symmetry	59.453	10	256	1	40	40	5	8	64	1024000	8	2
symmetry	59.6888	10	256	1	40	40	5	8	64	1024000	4	2
symmetry	59.7976	10	256	1	40	40	5	8	128	1024000	2	2
symmetry	60.0603	10	256	1	40	40	5	8	128	1024000	4	4
symmetry	60.1971	10	256	1	40	40	10	8	128	1024000	8	2
symmetry	60.5352	10	256	1	40	40	5	8	64	1024000	2	4
symmetry	61.0143	10	256	1	40	40	20	8	128	1024000	4	2
symmetry	61.0216	10	256	1	40	40	10	8	128	1024000	4	4
symmetry	61.1173	10	256	1	40	40	5	8	64	1024000	4	4
symmetry	61.2639	10	256	1	40	40	10	8	128	1024000	2	8
symmetry	61.65	10	256	1	40	40	5	8	128	1024000	8	2
symmetry	62.3565	10	256	1	40	40	10	8	64	1024000	4	2
symmetry	62.3756	10	256	1	40	40	5	8	64	1024000	2	8
symmetry	62.6931	10	256	1	40	40	5	8	128	1024000	2	8
symmetry	62.7316	10	256	1	40	40	20	8	128	1024000	2	4

Z4c Single-Node Performance

- Total cost for RHS for one grid point: 5462 flop (I counted)
- CPU performance: 2.4 GHz, 32 flop/cycle
- Benchmark: 57.2 s for 10 iterations on 40 cores
  - 1 + 5 * 10 = 51 RHS calls
- Theoretical Peak Performance:
  - **0.071 µs/RHS point** (per core)
- Measured:
  - **2.7 µs/RHS point** (per core), **2.7% of TPP**

I/O

- Each time step goes into its own file. Files are never re-opened.
- Norms (reductions): all in one file
- TSV (Tab-Separated Values, ASCII): one file per group
  - Now easily readable by Excel, gnuplot, Julia, Mathematica, Python
- Silo (HDF5): Standard VisIt format
  - Can be read and written
  - Could add compatibility shim for old Carpet format
  - Checkpointing/Recovery not yet supported

[Show WaveToyCPU, wavetoy.cxx]

- Discuss:
  - #include <cctk_Arguments_Checked.h>, DECLARE_CCTK_ARGUMENTS_WaveToyCPU_Initialize
  - Loop::loop_int<1, 1, 1>(cctkGH, [&](const Loop::PointDesc &p) {
  - PointDesc p

Manual Loop Tiling
(You don’t need to do that.)

- Tiling is a parallelization that does not need ghost zones

- New cGH entries:
  - cctkGH->cctk_tile_min[3];
    cctkGH->cctk_tile_max[3];
- Scheduling functions are called for a single tile only, not for the whole grid
- Don’t loop over all of cctk_lsh;
  only write to the current tile (but can read outside tile)

Immediate Next Steps

- Assist early adopters to get started
  - Big changes: schedule clauses, no subcycling, loop tiling
- Implement checkpointing/restart
- Test performance on many (10k … 100k) cores
- Do some physics
  - Probably needs relativistic hydro thorn