# Next Generation AMR

**Ann Almgren**

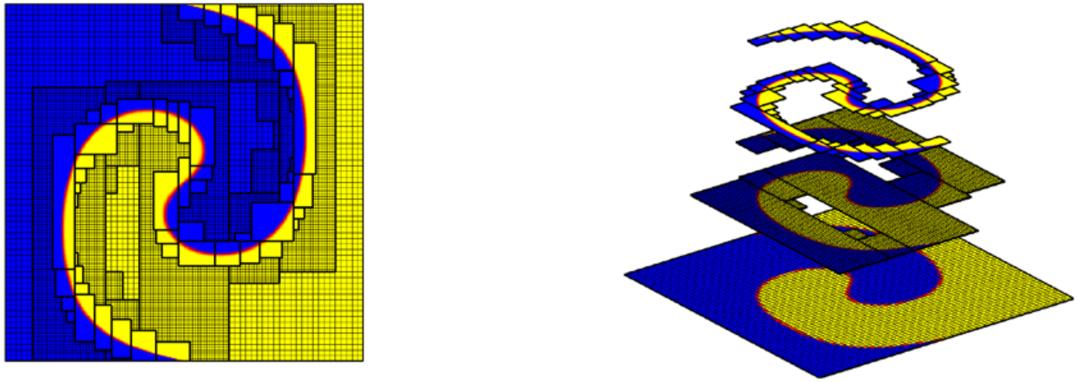
**Lawrence Berkeley National Laboratory**

**March 1, 2017**

# AMR is Everywhere!

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

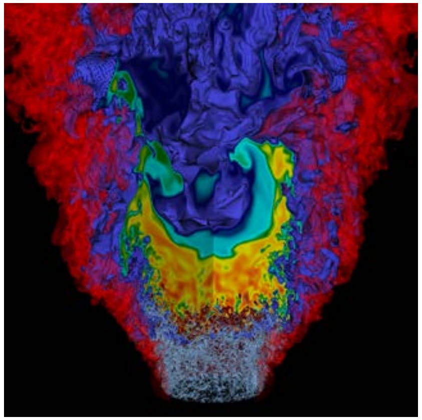
- Different frameworks, different languages
- More general PDE’s … from hyperbolic conservation laws to elliptic solves to complex multiphysics applications



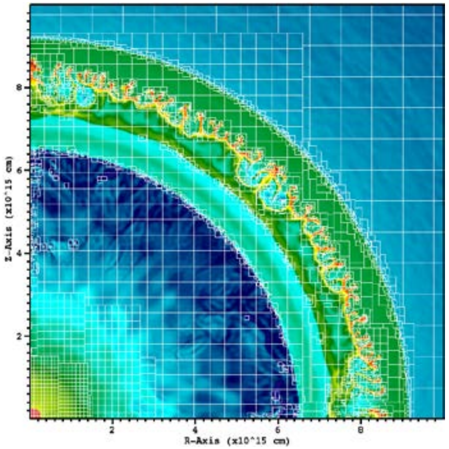
See, e.g., Donna Calhoun’s website, for available codes today:

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

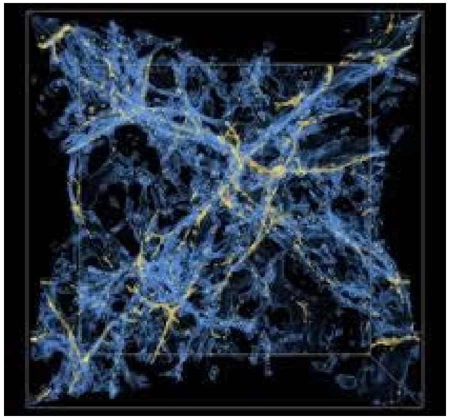
# AMR applications span many fields



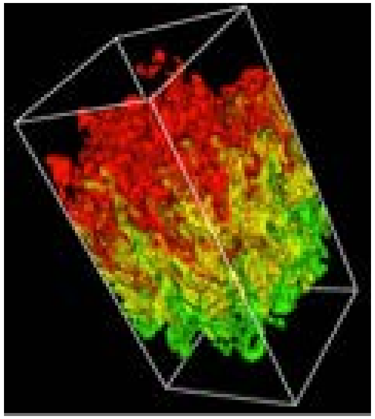
Combustion



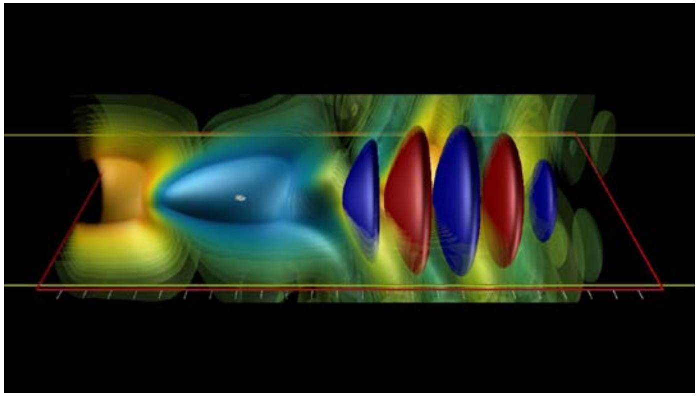
Astrophysics



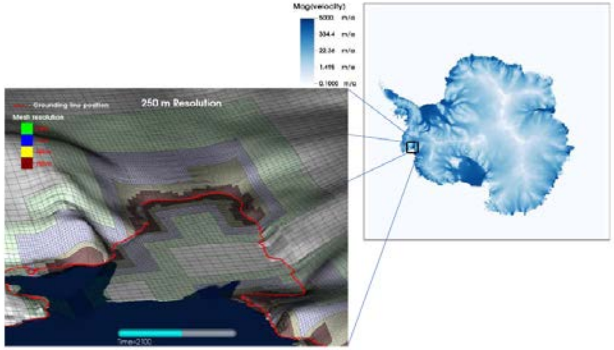
Cosmology



Fluid Instabilities

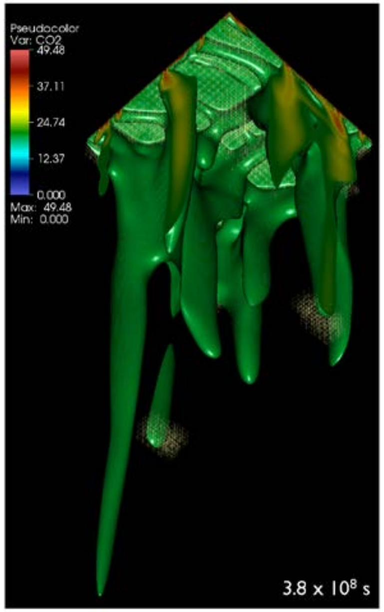


Accelerators



Ice sheets

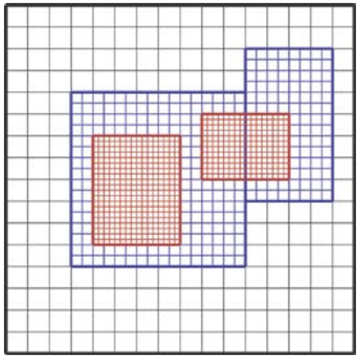
Subsurface Flow

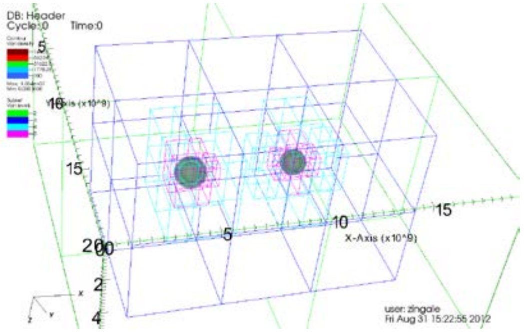


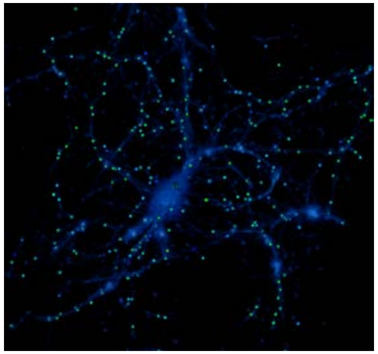
U.S. DEPARTMENT OF ENERGY | Office of Science

# Block-Structured AMR Defines the Data Layout

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 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

Data is in the form of

- mesh data
- Particles

# AMR provides natural opportunities for parallelism

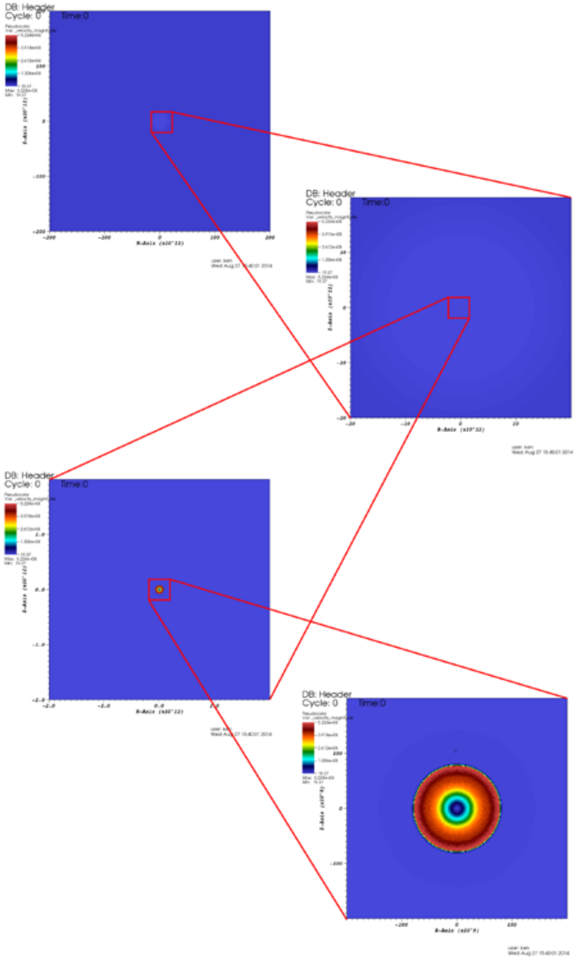
- AMR provides a natural framework for reducing the memory footprint and computational cost of a structured grid simulation

- The infrastructure to support block-structured AMR naturally supports hierarchical parallelism:
  - Coarse-grained dynamic load balancing due to decomposition into multiple grids at multiple levels
  - Fine-grained optimization opportunities due to regular patches of data

# AMR Does Not Define the Discretizations

- Block-structured AMR does not define the algorithm or the spatial or temporal discretizations

- Time-stepping options including
  - Advancing all levels with a single time step
  - Subcycling in time (finer levels take multiple time steps for each coarser time step)
  - Optimal subcycling (subcycle between some but not all levels as determined by the time step constraints)
  - Multilevel iterative approaches such as MLSDC (multilevel spectral deferred corrections)



Power of 10 (CASTRO)

Each close-up is by a factor of ten

# Key Issues for Next Generation AMR

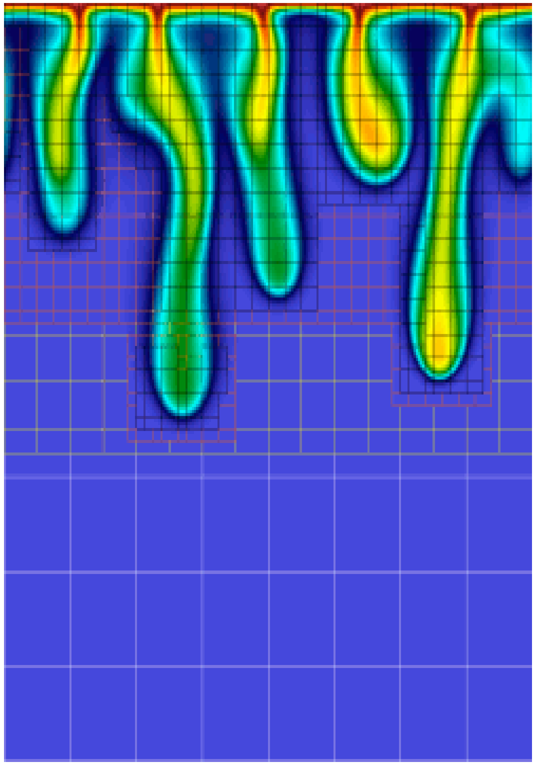
1. Single-core and single-node performance
2. Programming Models – is MPI+X the answer?
3. Load Balancing
4. Synchronicity
5. New Equations / Algorithms?
6. In Situ / In Transit Analytics & Visualization

# Key Issues for Next Generation AMR

1. Single-core and single-node performance
2. Programming Models – is MPI+OpenMP the answer?
3. Load Balancing
4. Synchronicity
5. New Equations and Algorithms
6. In Situ / In Transit Analytics & Visualization

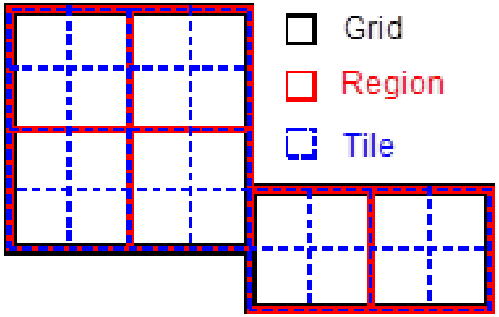
# Single-Core Performance Still Matters!

- Per-core performance still matters
- Memory access cost increasingly important
- Block-structured refinement provides natural framework for regular memory access
  - tiling
  - vectorization
  - autotuning
  - communication-avoiding algorithms

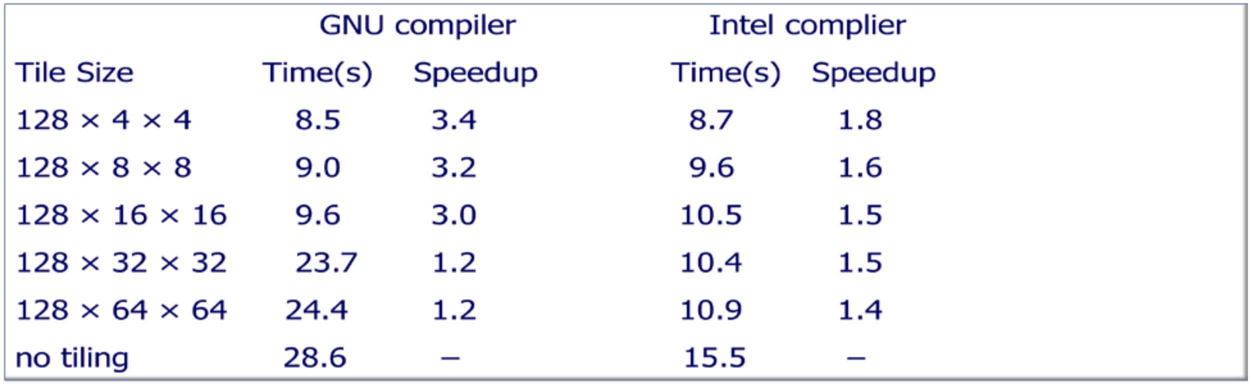


# Logical Tiling Can Reduce Cost on a Single Core

- With logical tiling, the data layout is unchanged but the unit of work is a tile rather than a grid
- Can hide tiling in the iterator so is invisible to the application
- Leads to more efficient memory access



**1 core of Edison**
**128^3 domain**

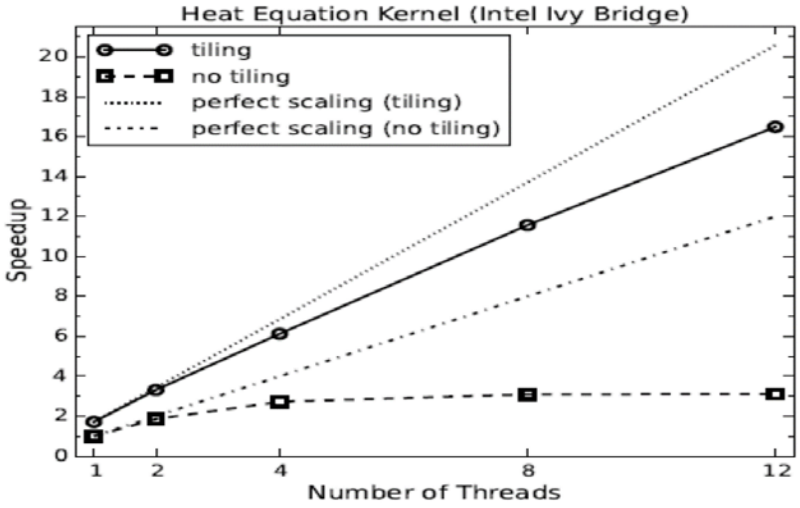


| | GNU compiler | | Intel complier | |
|---|---|---|---|---|
| Tile Size | Time(s) | Speedup | Time(s) | Speedup |
| 128 × 4 × 4 | 8.5 | 3.4 | 8.7 | 1.8 |
| 128 × 8 × 8 | 9.0 | 3.2 | 9.6 | 1.6 |
| 128 × 16 × 16 | 9.6 | 3.0 | 10.5 | 1.5 |
| 128 × 32 × 32 | 23.7 | 1.2 | 10.4 | 1.5 |
| 128 × 64 × 64 | 24.4 | 1.2 | 10.9 | 1.4 |
| no tiling | 28.6 | – | 15.5 | – |

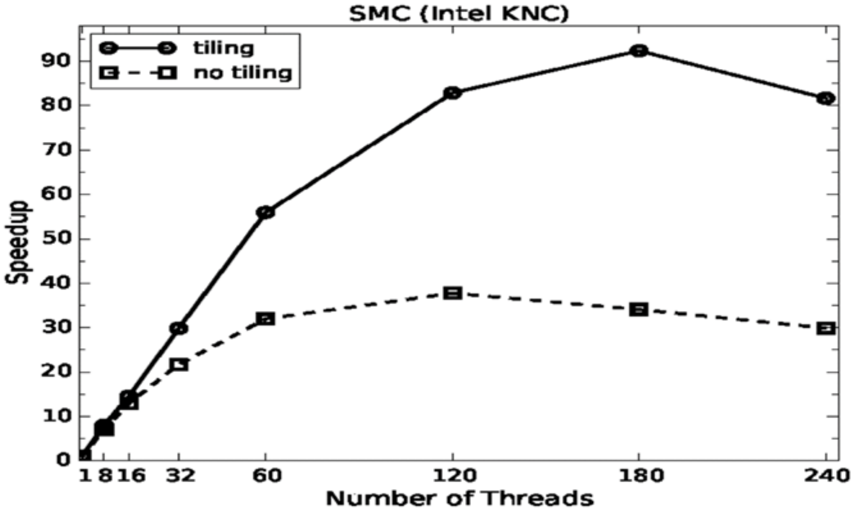
Courtesy of Weiqun Zhang, Didem Unat and Tan Nguyen

# Logical Tiling Can Reduce Cost Across Cores

- Logical tiling makes smaller units of work, so we can distribute work more effectively over all the cores when Ngrids << Ncores



**1 node of Edison (12 cores)**



**1 node of Babbage (60 cores)**

Courtesy of Weiqun Zhang, Didem Unat and Tan Nguyen

# Key Issues for Next Generation AMR

1. Single-core and single-node performance
2. Programming Models – is MPI+OpenMP the answer?
3. Load Balancing
4. Synchronicity
5. New Equations and Algorithms
6. In Situ / In Transit Analytics & Visualization

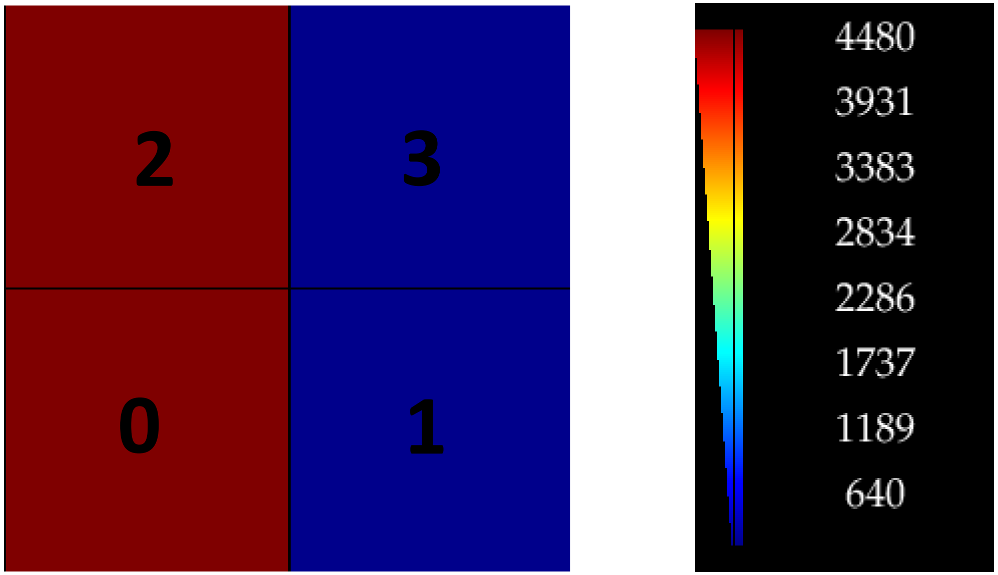
# Single-Level Single-Physics Load Balancing

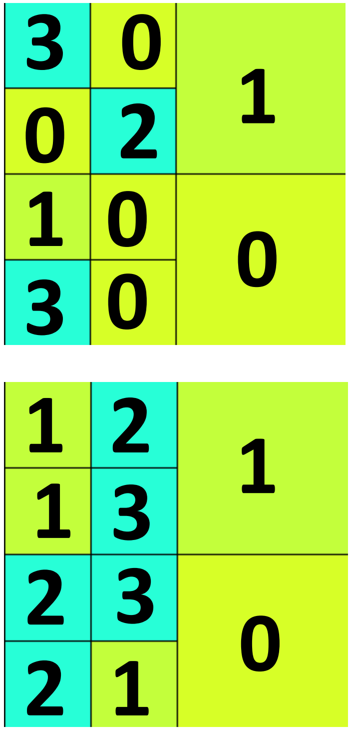
Load balancing based on number of cells

- 4484 particles on brown grids,
- 644 particles on blue grids

Load balancing based on number of particles

- 2244 -- 2916 particles per process
- Maximum particle work / process reduced by 33%





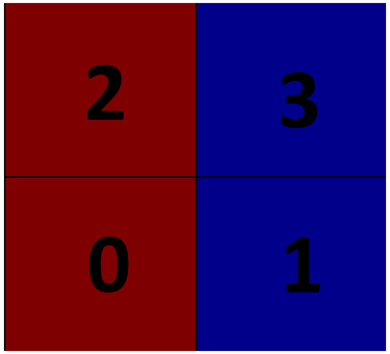
Simple Test Case –

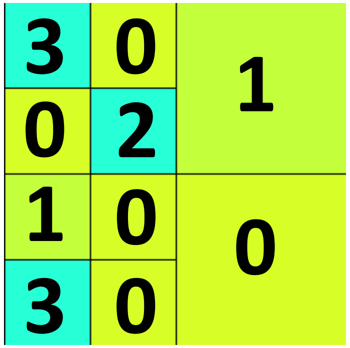
- 4 MPI processes
- Particles mostly on left side of domain
- Knapsack algorithm for load balancing

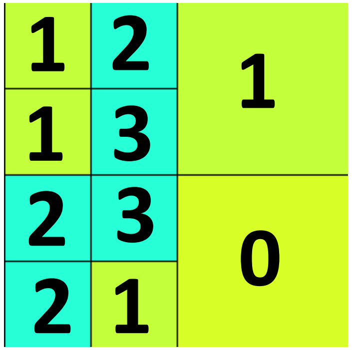
# Single-Level Multi-Physics Load Balancing

With dual grid approach, particle work
(e.g. particle-particle operations and deposition) can be done on right mesh while operations on only mesh data are done on the left mesh

- In PIC algorithm we would typically use the left mesh for the elliptic solve -- so we will need to copy data between meshes – whether this is a win depends on cost of particle work vs grid-grid communication
- Obviously cost of communication depends on locality of grids between decompositions as well as topology – and cost is changing dynamically







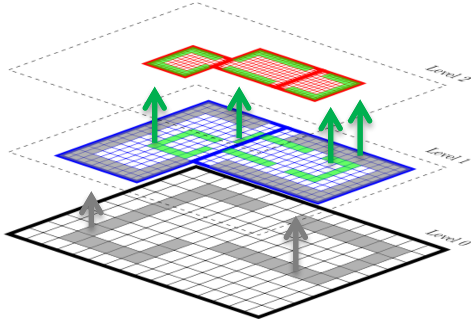
# Single-Level Multi-Physics Load Balancing

Approaches:

- Single decomposition with uniform tiling approach
- Dual grid decomposition
- Single decomposition with physics-specific tiling / "work stealing"

Multi-level multiphysics load balancing is a much harder problem – there is not an exact solution!

We need good heuristics



# A Good Model Can Help Us Understand and Predict the Costs of Different Data Distributions

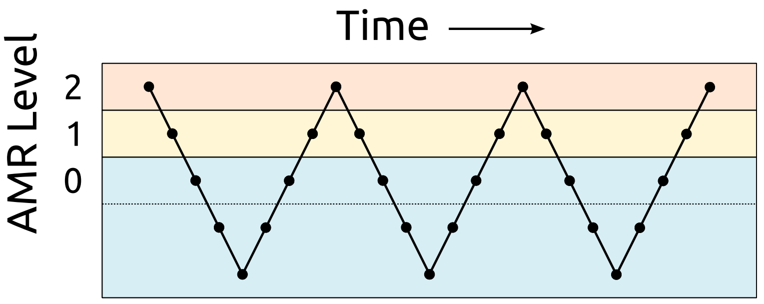
Can use a model – of the network, the data dependencies, and the computational tasks

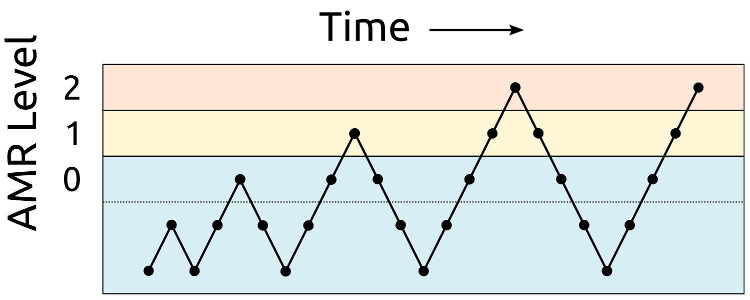
- to determine the optimal distribution of grids to processes
- to assess the cost of data movement vs computational imbalance, etc →heuristic for when it is worth making changes
- to assess impact of different (current and future) hardware architectures on overall performance

Courtesy of Cy Chan

# Predictive Load Balancing Can Help Us Make Algorithmic Choices

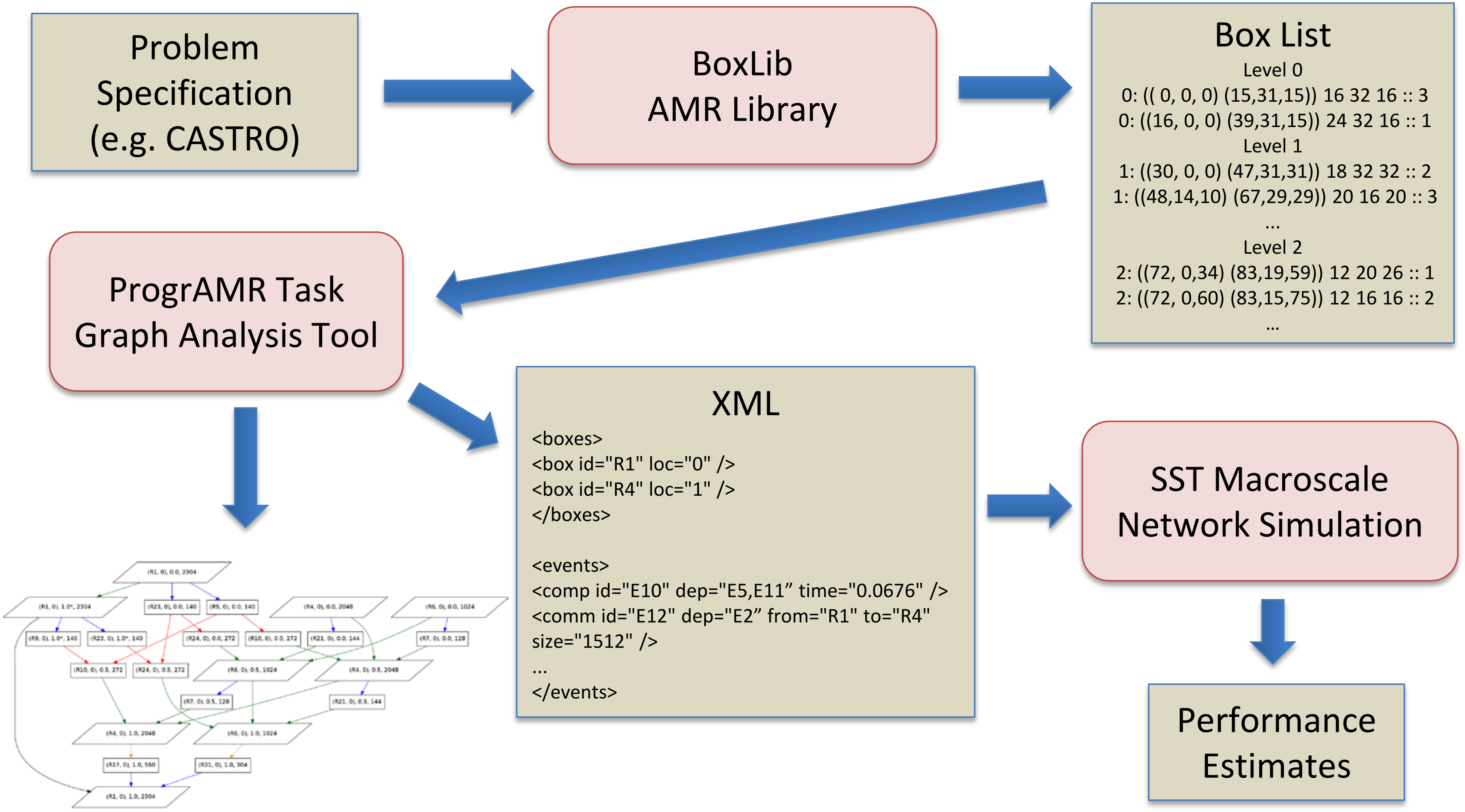
We can also use the model to choose between algorithmic variants – such as whether to use V-cycles vs F-cycles in geometric multigrid





Courtesy of Cy Chan

# BoxLib/ProgrAMR/SST Analysis Workflow



Problem Specification (e.g. CASTRO)

BoxLib AMR Library

**Box List**

Level 0
0: (( 0, 0, 0) (15,31,15)) 16 32 16 :: 3
0: ((16, 0, 0) (39,31,15)) 24 32 16 :: 1
Level 1
1: ((30, 0, 0) (47,31,31)) 18 32 32 :: 2
1: ((48,14,10) (67,29,29)) 20 16 20 :: 3
...
Level 2
2: ((72, 0,34) (83,19,59)) 12 20 26 :: 1
2: ((72, 0,60) (83,15,75)) 12 16 16 :: 2
...

ProgrAMR Task Graph Analysis Tool

**XML**

```
<boxes>
<box id="R1" loc="0" />
<box id="R4" loc="1" />
</boxes>

<events>
<comp id="E10" dep="E5,E11" time="0.0676" />
<comm id="E12" dep="E2" from="R1" to="R4"
size="1512" />
...
</events>
```

SST Macroscale Network Simulation

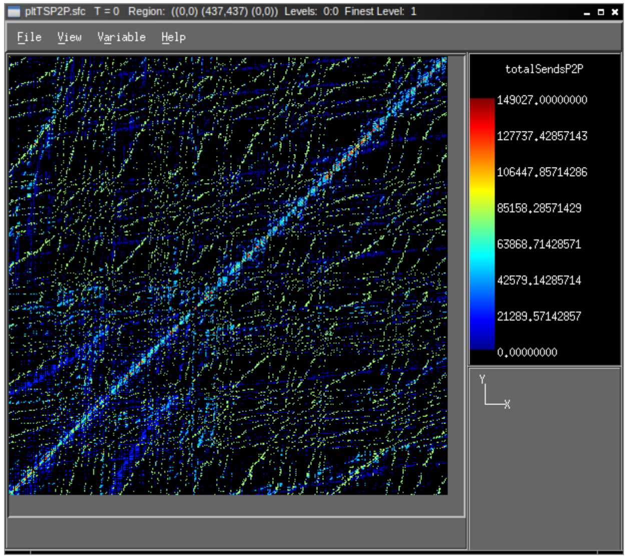
Performance Estimates

Courtesy of Cy Chan, John Bachan, Vince Beckner, John Shalf, Joseph Kenny, Jeremiah Wilke

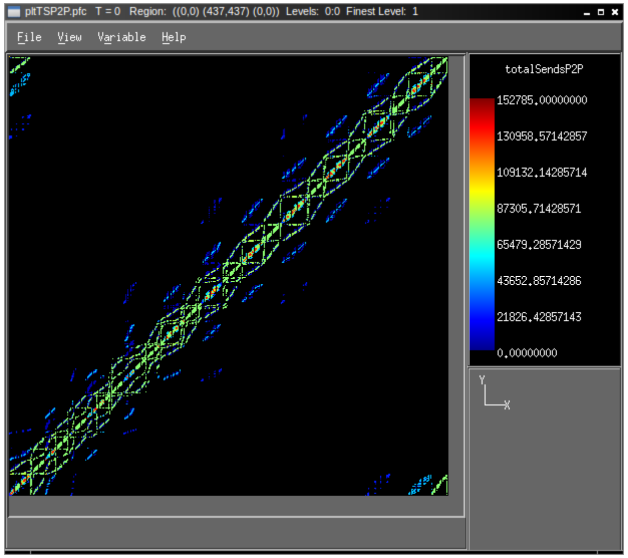
# Must Calibrate the Performance Model with Real-Time Performance Measurement

We use a real-time communication and computation profiling capability to generate a database that we can query for specific features and visualize as an adaptive data set.

This shows number of sends from node x to node y



Space-filling curve



Proximity-filling curve

Credit to Vince Beckner

# Key Issues for Next Generation AMR

1. Single-core and single-node performance
2. Programming Models – is MPI+OpenMP the answer?
3. Load Balancing
4. Synchronicity
5. New Equations and Algorithms
6. In Situ / In Transit Analytics & Visualization

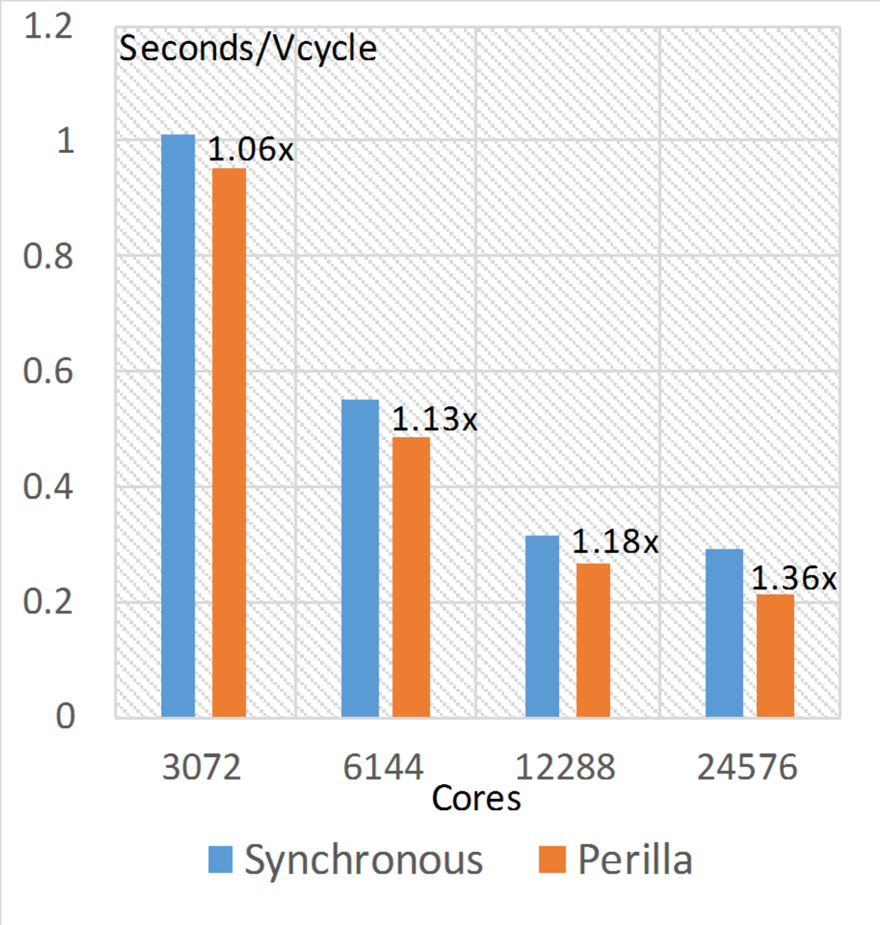
# 4. Synchronicity

- Synchronicity means different things to different people
- Not clear that we really know what we need
- Possible needs:
  - Low-level asynchrony: imagine operating on “interior” tiles while filling ghost cells of tiles touching boundaries –
    - invisible to the application
  - Medium-level asynchrony: imagine performing 4 multigrid solves (on different solution variables) at the same time in order to e.g., overlap computation of one with communication of the other –
    - visible to application but ok
  - High-level asynchrony – change ordering of high-level tasks
    - for an algorithm with many implicit operations this may be less effective – can’t have any one grid get too far ahead …
    - Potential memory bloat if can’t update solution in place
    - Needs to know a lot more about the algorithm!

# AMR Metadata Can Facilitate Use of Asynchronous Runtime

- At the lowest level, we can use an asynchronous runtime
  - Leverages the metadata already created to simplify the process of constructing a task graph
  - Hides communication overhead with asynchronous messages
  - NUMA-aware: communication within a compute node is fast

- Results show up to 1.36x speedup for 2K^3 geometric multigrid solver on 24K cores on Edison



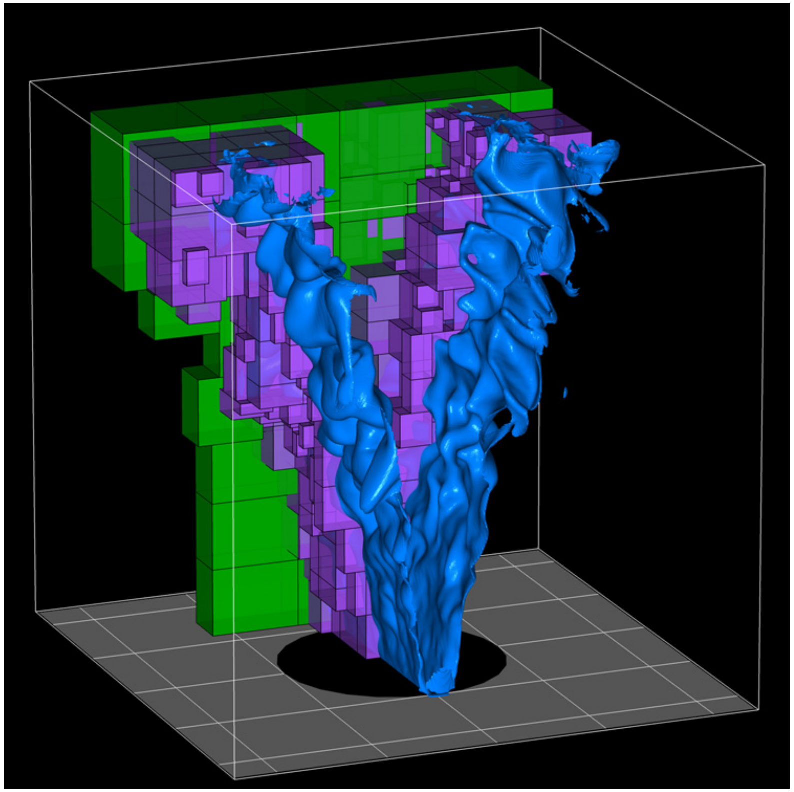
Courtesy of Tan Nguyen, Didem Unat, Weiqun Zhang

# Key Issues for Next Generation AMR

1. Single-core and single-node performance
2. Programming Models – is MPI+OpenMP the answer?
3. Load Balancing
4. Synchronicity
5. New Algorithms and Equations
6. In Situ / In Transit Analytics & Visualization

# The Software Should Not Dictate the Algorithms

- When we use AMR to solve the equations for complex multiscale processes, the physics -- not the software -- should dictate the algorithm



# Algorithms Can Allow New Opportunities

- Can the algorithm be implemented differently to remove synchronization points?
    - e.g., remove norm calculations

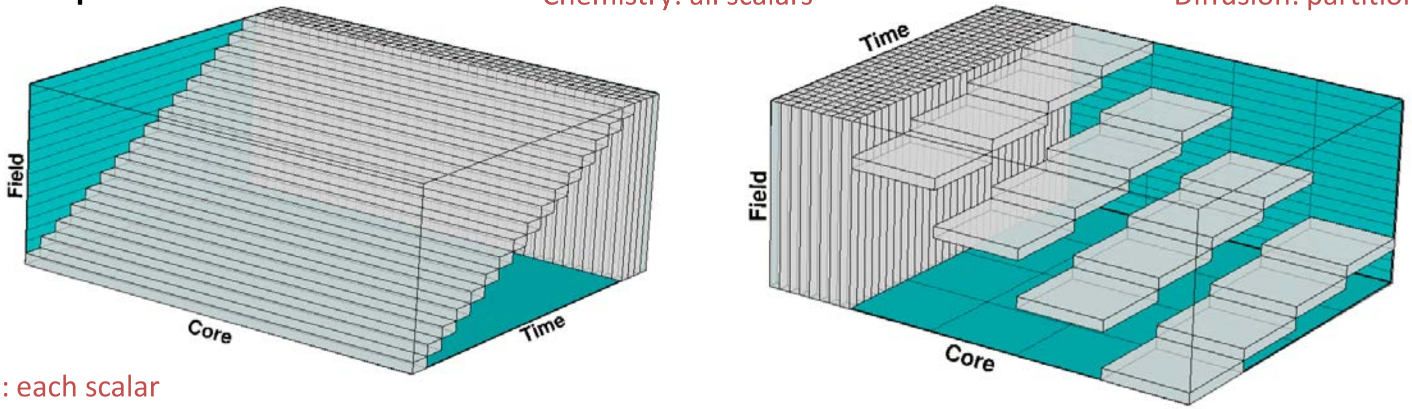
- Can the algorithm itself can be modified to allow asynchronous execution?
    - e.g., multiple diffusion solves at one time

- Iterative approach allows asynchronous evaluation of component processes
    - e.g, overlap diffusion and reaction processes

# Asynchronous Spectral Deferred Corrections

Iterative time step couples multiple (stiff) processes with different communication/computation requirements (e.g., diffusion vs. chemistry)

Time step:



Chemistry: all scalars

Diffusion: partitioned cores

Diffusion: each scalar over entire machine

*Synchronous (SDC)* *vs.* *Asynchronous (ASDC)*

Requires:

1. Time-stepping strategy that preserves high-order capability, and is independent of evaluation order of processes
2. Significant communication – effectively a “transpose” over (field, space)

Courtesy of Marc Day

# New Equations Can Create New Opportunities

- Computational efficiency can arise not just from carefully implementing the existing equations, but also from solving different equations

- Low Mach number approximations are an example of this – each time step is more expensive due to the linear solve for the pressure update, but the time step is much larger

- AMR opens up the possibility of different descriptions of the physics at different levels
  - Particle description of fluid in fine patch embedded within continuum model
  - Low Mach number model in fine patch embedded within compressible fluid formulation

# We Can Take Advantage Of All These Opportunities By …

- Leveraging the software stack to achieve peak performance: taking advantage of advances in compilers, programming models, analysis & vis tools, …
- Working together: computer scientists / mathematicians / application scientists must work together – this requires a team effort
- Keeping our options open: new machines enable us to solve new problems which leads to new challenges – it is essential that we retain the flexibility to address the new challenges as they arise