Next Generation AMR:

An Application Developer's Perspective

Ann Almgren

April 7, 2016

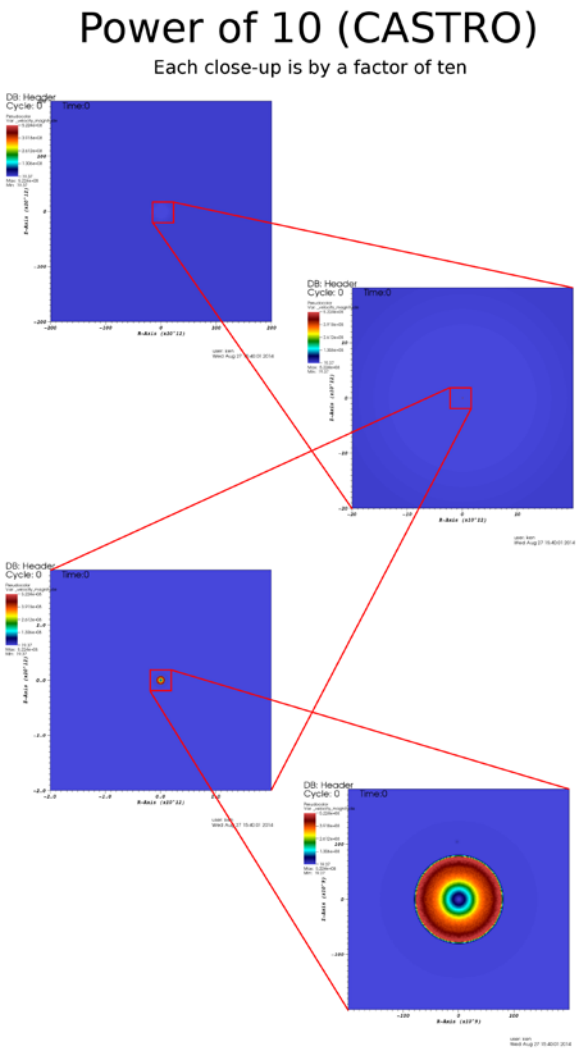
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

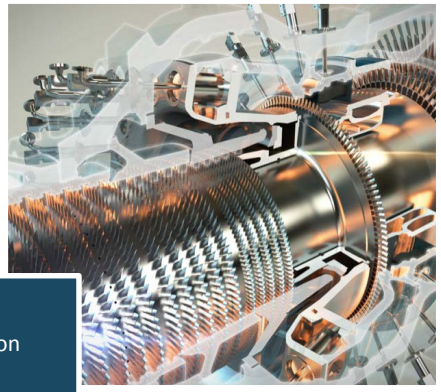
AMR is a Team Sport

- BoxLib Contributors and Collaborators:

  Vince Beckner, John Bell, Cy Chan, Marc Day, Brian Friesen, Max Katz, (Mike Lijewski), Andy Nonaka, Sam Williams, (Mike Welcome), Weiqun Zhang, Yili Zheng, Mike Zingale and more

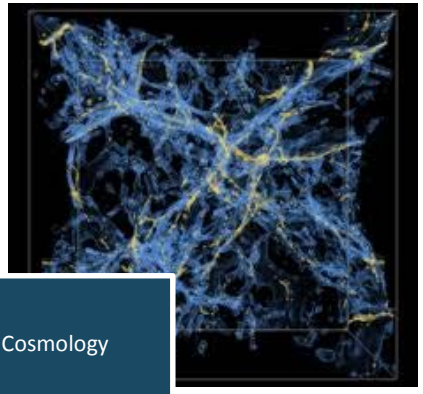


AMR is Used in a Number of LBL Applications

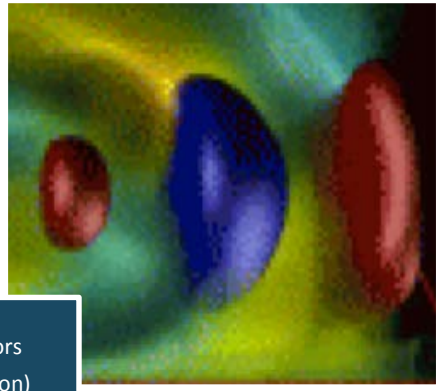


Combustion

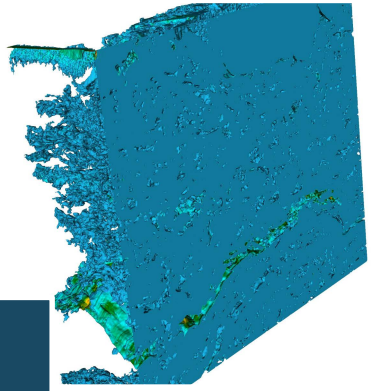
Astrophysics



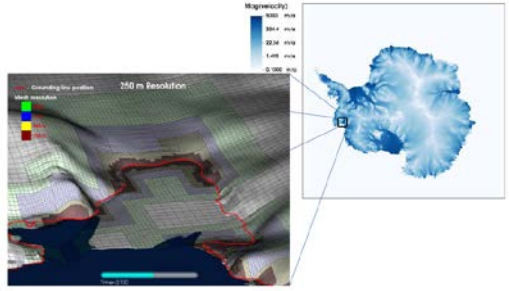
Cosmology



Accelerators
(coming soon)



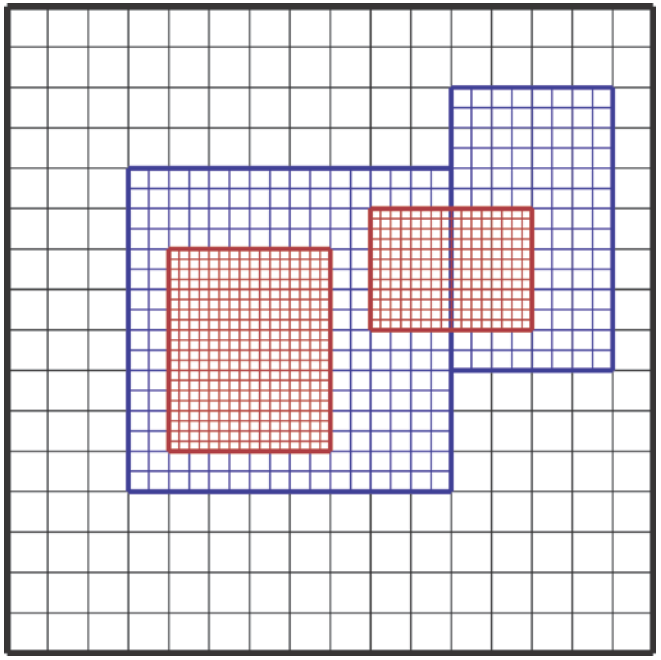
Subsurface



Climate

Block-Structured AMR for Time-Dependent PDEs

- Data in the form of
  - mesh data (on centers, faces and corners of cells) and
  - particles
- Data is (in the eyes of the application) organized into “large” grid patches at different levels
  - Patches may not be fixed size
  - Patches change dynamically



Key Features

- Meshing is dynamic
  - Can't statically optimize
  - Can't always amortize set-up time / caching of important information over many operations
- Single-level operations and multi-level operations
- Communication between grids at a single level, and between grids at coarser/finer levels
- Mesh and particle-mesh operations
- Explicit and implicit solves
- No one approach to spatial or temporal discretizations
- We already have many codes that work really well, and the design space of adaptive algorithms keeps increasing

Software Principles

- Software "solutions" must not get in the way of solving the problems we want to solve. Show-stoppers include

- The software must not preclude the design of new algorithms and the inclusion of new physics.
- The application code must work once your research project has ended

Key Issues from “Our” Perspective

1. On-node performance
2. Programming Models – will MPI+X be enough?
3. Load Balancing
4. Synchronicity

1. On-Node Performance

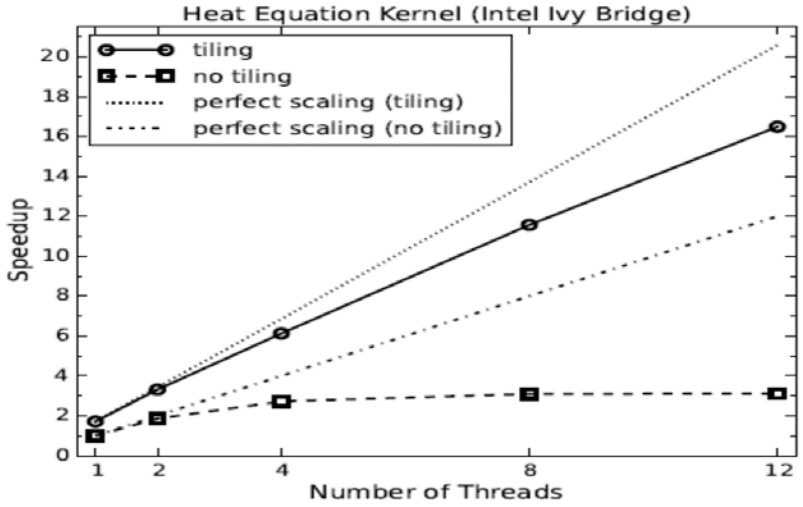
1. Use all the cores effectively when Ngrids << Ncores
   - Tiling (unit of work = 1 tile not 1 grid) is one way to break up the work
     - Can be hidden in the iterator and invisible to the application

2. Optimize the performance of each core (autotuning, vectorization, code transformation, communication-avoiding algorithms)
   - This tends to be more application-specific
   - Can write specific optimized code for some common routines, but would be nice to have “easy” ways to optimize others.

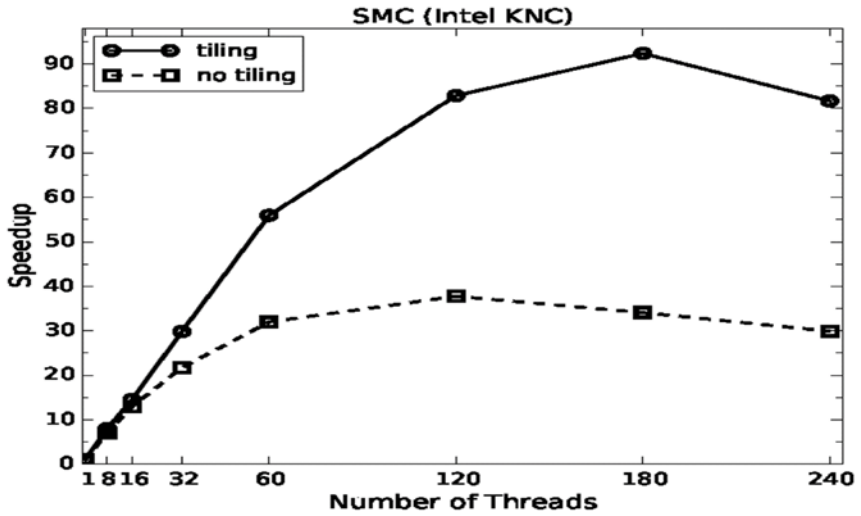
Performance Benefits from Tiling

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	–



1 node of Edison (12 cores)



1 node of Babbage (60 cores)

Courtesy of Weiqun Zhang

2. Programming Models: Is MPI+X Enough?

Current paradigm has been 1-4 MPI processes per node, with OpenMP to thread over tiles.

- Flat MPI is not the answer
  - Too many grids (bad for metadata)
  - Grids too small (too many ghost cells)
  - Too many MPI processes → performance hit
- MPI+OpenMP communication has not been great in our applications

2. Programming Models: Is MPI+X Enough?

- Talk by Yili Zheng yesterday gave examples of BoxLib + X where X was
  - Flat vs Hierarchical
  - Combinations of MPI, OpenMP, UPC++
- UPC++ shows performance benefits even now (thanks to Weiqun Zhang and Yili Zheng)
- Our strategy is to keep the options available in BoxLib as a run-time option
- Keys to Success:
  - "Collaboration and integration are key!" (quote from Yili)
  - Incremental approach – never lost functionality of current codes
  - By keeping options open, we are guaranteed no loss of performance relative to committing to a single model
  - Choice of model is invisible to the application developer

3. Coarse-Grained Load Balancing

Predictive

- Use a model to determine
  - optimal grid size, tile size, and distribution of grids to processes
  - cost of data movement vs computational imbalance, etc
    →heuristic for when it is worth making changes
- This allows us to assess both current and future architectures

Run-time

- Use real-time measurements in combination with heuristics to determine when action is required and what action to take – moving data more often might make more sense when network less crowded, etc
- Experiments suggest that on Edison/Cori data movement of LMC simulation state data roughly 1% of simulation time

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

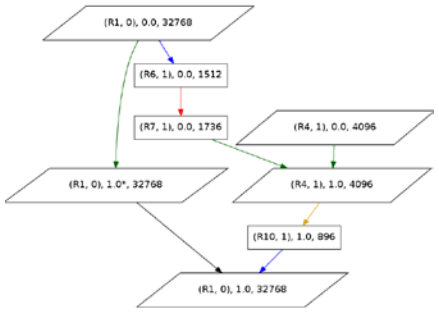
XML

```
<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



Task Dependency Graph

Credit to: Cy Chan, John Bachan, Vince Beckner, John Shalf, Joseph Kenny, Jeremiah Wilke

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

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!

Summary

We want to take advantage of what CS experts have to offer -- languages/runtimes/programming models can certainly make our lives easier.

Caveats:

- Don’t take away our ability to design new algorithms and support multiple applications
- Make the hard things easy; don’t make the easy things hard.