Parallel, adaptive framework for mapped, multi-block domains

Donna Calhoun (Boise State University)

Carsten Burstedde (University of Bonn)

Randall J. LeVeque (Univ. of WA), Marsha Berger (NYU), David George (USGS)

PDEs on the Sphere
April 7-11, 2014
Boulder, CO
Applications of AMR

- Astrophysics, combustion
- Shock capturing for aerodynamic applications
- Storm surges, debris flow, porous media flow
- Ice sheet modeling, tsunami modeling

Tsunami modeling (R. LeVeque, D. George, M. Berger)

Rod stabilized V-flame (J. B. Bell, Lawrence Berkeley Lab)
GeoClaw (R. J. LeVeque, D. George, M. Berger) used to model recent landslide in Washington State
Almost exactly 30 years, Marsha Berger introduced block-structured AMR

Block-structured AMR

- General purpose (freely available) block-structured codes
  - **SAMRAI** (Lawrence Livermore National Lab)
  - **BoxLib** (Lawrence Berkeley Lab)
  - **Chombo** (Lawrence Berkeley Lab)
  - **AMRClaw** (University of Washington/NYU)
  - **AMROC** (Ralf Deiterding, DLR, Germany)
  - **PARAMESH** (NASA/Goddard) (not technically “block-structured”, but rather quadtree-based.)

- Most are large frameworks, with many developers
- Mostly C++ and Fortran libraries (no GUIs) that started life as research codes.

See my website for a list of many more application specific codes.
Goal of patch-based AMR codes

- Make full use of existing solvers for Cartesian grids
- Operate locally on patches whenever possible
- Have the same order of accuracy as the single grid algorithm.
- Maintain conservation where appropriate
- Use local time stepping to maintain a constant CFL number across refinement levels,
- Fully couple solution between grids,
- Operate efficiently on latest hardware.

Goal is to do this without significant overhead associated with managing the grid hierarchy.
Why are AMR codes difficult to write?

- Heterogeneous data structures for storing hierarchy of grids,
- Dynamically creating and destroying grids,
- Need a “factory” paradigm to create user defined auxiliary data arrays (material properties, metric terms, bathymetry, etc) needed for each new grid,
- Communication between patches,
- Parallel load balancing and IO,
- Efficient implementation of multi-rate time stepping schemes,
- User interface for mixed type equations and solvers,
- Error estimation, tuning for efficient use of grids,
...and hard to use

- Time stepping methods beyond one-step, single stage methods, including multi-stage Runge-Kutta, IMEX, SSP, parallel-in-time, exponential integrators, HEVI, spectral deferred correction, …

- Accuracy of multi-rate schemes for PDEs with mixed elliptic/parabolic/hyperbolic terms.

- Elliptic and parabolic solvers (iterative? direct? Explicit? Implicit?)

- Refinement criteria?

- Higher order accuracy

- Complex physics

- Visualization

- Debugging and post-processing
AMR Skeptics?

- Coarse/fine boundaries with abrupt resolution changes are regarded with suspicion,
- Lack of good refinement criteria dampens enthusiasm for trying out AMR,
- Not obvious how to extend sophisticated numerical algorithms and applications to the adaptive setting,
- Physics parameterizations

*When multi-resolution grids are used …*

- Multi-rate time stepping is not often used (it seems)
- The goals are often modest: “Do no harm!”
- One way coupling of regional, static grids
Tracer transport

Tracer transport of pollutants, volcanic ash

AMR for the computational mathematician

• Support for grid management that is separate from the numerics, that is intuitive, with easily manageable data structures,
• Support for multi-rate time stepping with flexibility to include new time stepping schemes (MOL solvers, for example),
• Easy to add diagnostics for convergence studies,
• Natural code for iterating over arrays (in Fortran?),
• Flat data structures - little reliance on templates, and exotic object oriented data structures,
• Parallelism should happen automatically.
• Simple build system

Building your own code is reasonable if you start with…
p4est - dynamic grid management

p4est (Carsten Burstedde, Univ. of Bonn) is a highly scalable library for dynamically managing an octree of grids.

Add solvers

• Wave propagation algorithm - Clawpack (R. J. LeVeque) -
  second order finite volume scheme for hyperbolic
  conservation laws.
    - assumes logically Cartesian smooth or piecewise
      smooth meshes,
    - suitable mapped Euclidean and non-Euclidean grids
    - Available in well tested Clawpack ([www.clawpack.org](http://www.clawpack.org))

• Runge-Kutta Chebyshev solvers for explicit diffusion
  problems (reaction diffusion)

• Flexibility in user choosing their own solver.
AMR using ForestClaw

Local grid refinement based on subdividing quadrants of the computational domain.

*ForestClaw* (D. Calhoun, C. Burstedde)
AMR using ForestClaw

$q(2)$ at time 1.0000

PDEs on the Sphere, 2014
AMR using ForestClaw

Fine level grids

Coarse level grids

Quadtree based refinement

Each grid has a layer of ghost cells (not shown) to facilitate communication
• Ghost cell exchanges
• Multi-rate time stepping
• Parallel communication
How are ghost cells filled?

- Copy
- Average
- Interpolate
Multi-rate time stepping

Coarsest level

Finest level

fill in ghost cells

Levels

L = 3

L = 4

L = 5

L = 6

Time

t

t + \Delta t

PDEs on the Sphere, 2014
Multi-rate time stepping

- Coarsest level
- Finest level

Levels:
- $L=3$
- $L=4$
- $L=5$
- $L=6$

$t + \Delta t$
Mixing diagnostic

Preservation of functional relationship between tracers.

Mixing diagnostics (256 x 256)

<table>
<thead>
<tr>
<th>Diagnostic</th>
<th>Fraction</th>
<th>Diagnostic</th>
<th>Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real mixing (r)</td>
<td>5.45E-04</td>
<td>1.68E-04</td>
<td>0.7709</td>
</tr>
<tr>
<td>Range preserving mixing (g)</td>
<td>1.49E-04</td>
<td>4.2E-05</td>
<td>0.1925</td>
</tr>
<tr>
<td>Under and over shoots (b)</td>
<td>2.63E-05</td>
<td>7.98E-06</td>
<td>0.0366</td>
</tr>
</tbody>
</table>

No AMR vs. with AMR
Mixing diagnostic (1024 x 512)

<table>
<thead>
<tr>
<th>Diagnostic</th>
<th>Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real mixing (r)</td>
<td>1.5586E-05</td>
</tr>
<tr>
<td>Range preserving unmixing (g)</td>
<td>6.9279E-06</td>
</tr>
<tr>
<td>Under and over shoots (b)</td>
<td>1.3079E-06</td>
</tr>
</tbody>
</table>
### Parallel performance - swirl example

#### Figure 4.
Advection, strong scaling for mx = my = 32, uniform grid

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Remesh</th>
<th>Partition</th>
<th>Time Step</th>
<th>Wall Clock Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>None</td>
<td>by count</td>
<td>global</td>
<td>3961.</td>
</tr>
<tr>
<td>AMR every step</td>
<td>by count</td>
<td>252.</td>
<td>54.6</td>
<td></td>
</tr>
<tr>
<td>AMR every 4</td>
<td>by count</td>
<td>178.</td>
<td>39.7</td>
<td></td>
</tr>
<tr>
<td>AMR every step</td>
<td>subcycle</td>
<td>99.9</td>
<td>17.3</td>
<td></td>
</tr>
<tr>
<td>AMR every 4</td>
<td>subcycle</td>
<td>87.2</td>
<td>14.0</td>
<td></td>
</tr>
<tr>
<td>AMR every step</td>
<td>by weight</td>
<td>95.7</td>
<td>18.2</td>
<td></td>
</tr>
<tr>
<td>AMR every 4</td>
<td>by weight</td>
<td>84.4</td>
<td>14.2</td>
<td></td>
</tr>
</tbody>
</table>

**Factor of ~40 speedup for 16 processors, and close to 20 on 256 processors**

Goals and motivation

- Develop framework for general multi-rate time stepping schemes (work with D. Ketcheson)
- Improve parallel efficiency
- Handle general mapped multi-block case

Work towards full 3d simulations, with a stop along the way at 2.5d (refinement in horizontal only) for modeling ash cloud transport.

see [http://www.forestclaw.org](http://www.forestclaw.org)
Ash cloud modeling

- Split horizontal, vertical time stepping
- Fully conservative,
- Eulerian, finite volume
- Algorithms based on wave propagation