Scalable and Asynchronous Algorithms for Block Structured Adaptive Mesh Refinement

Akhil Langer and Laxmikant Kale

Parallel Programming Laboratory, Department of Computer Science
University of Illinois at Urbana-Champaign

Introduction to Adaptive Mesh Refinement (AMR)

- Solving Partial Differential Equations (PDEs)
  - PDEs solved using discrete domain
  - Algebraic equations estimate values of unknowns at mesh points
  - Resolution of mesh points determines error
- Applications
  - CFD
  - Astrophysics
  - Climate Modeling
  - Turbulence
  - Mars/Moon Convection Modeling
  - Combustion
  - Bio/Physics
  - and many more

Uniform meshes
- High resolution required for handling difficult regions (discontinuities, steep gradients, shocks, etc.)
- Computationally extremely costly
AMR makes it feasible to solve problems that are intractable on uniform grid

Adaptively Refined Meshes
- Data with a coarse grid
- Identify regions that need finer resolution
- Superimpose finer subgrid only on those regions

Typical Traditional Approach

- Refinement levels of neighboring blocks differ by ±1
- Refinement structure can be represented using a quad tree (2D) or oct tree (3D)

Uniform Meshes
- Tree meta-data replicated on each process
  - Large memory per process
- Level by level restructuring
  - Ripple propagation
  - O(N) reductions
- Does not allow coarsening of sibling blocks residing on different processors

Adaptively Refined Meshes
- Block acts as a virtual process
  - Overlap of computation with communication of other blocks
  - Run time handles communication between arbitrary blocks
- Dynamic placement of blocks on physical processes
  - Facilitates dynamic load balancing
- Block is a unit of algorithm expression
  - Simplifies implementation complexity

Scalable Approach – Basic Design

- Basic Design: Promote individual block as first-class entities, instead of a process
- Block acts as a virtual process
- Block Naming
  - Behavior describing path from root to block’s node
  - One bit per dimension at each level
  - Easy to compute parent, children, siblings
    - Using bit manipulation
- Block is a unit of algorithm expression

Algorithmic Benefits

<table>
<thead>
<tr>
<th>Typical Traditional Approach</th>
<th>Charm++ Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>O(Ah/blk) per process</td>
</tr>
<tr>
<td>Mesh Restructuring</td>
<td>O(h/blk) + O(h/blk)/(P) time</td>
</tr>
<tr>
<td>Neighbor Lookup</td>
<td>O(P) data structure + O(1) time</td>
</tr>
<tr>
<td>Implementation</td>
<td>Complex, size: 1300 for 2D, 1600 for 3D, Advection</td>
</tr>
</tbody>
</table>

Highly Asynchronous Scalable Approach

- The Highly Asynchronous Mesh Restructuring Algorithm
  - Based on local error estimate, blocks make
    - refine and stay decisions communicated to neighbors
  - Decisions updated based on DFT, changes in decision are communicated
  - refine, stay propagate along the mesh irrespective of blocks refinement levels
  - When to stop?
    - System quiescence indicates global consensus on refinement decisions
  - Blocks proceed to next iterations when quiescence detected, no need to wait for blocks to be created
  - Messages directed to not yet created blocks are buffered

Performance Results