Adaptive Mesh Refinement Methods and Parallel Computing

Rich Hornung
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
hornung@llnl.gov
www.llnl.gov/CASC/SAMRAI

The Conference on High-Speed Computing
Gleneden Beach, OR
April 19-22, 2004

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.
Presentation Outline

- Overview of adaptive mesh refinement (AMR) approaches
  - General issues
  - Unstructured AMR
  - Cell-by-cell structured AMR
  - Patch-based structured AMR
  - Hybrid AMR methods
- SAMRAI algorithm development to improve parallel scalability
- Concluding remarks
Adaptive Mesh Refinement (AMR) targets “multi-resolution” problems

- Many science & engineering problems exhibit solutions with large gradients separated by relatively large smooth regions

- Numerical solution of PDEs involves a discrete domain (i.e., grid) and algebraic approximation of equations
  - fine grids required to resolve local features
  - coarser grids suffice in regions where solution is smooth

- Grid spacing determines accuracy and cost of computation
  - fine grid everywhere may be inefficient
  - location and resolution of fine grid not always known a priori

- AMR dynamically focuses computational effort
  - adjust grid resolution to resolve local features without incurring cost of globally fine grid
  - computed solution should have globally “uniform” accuracy
AMR is increasingly important for large-scale problems, presents challenges

- AMR has demonstrated its usefulness in many application areas
  - shocks, fluid/material interfaces
  - plasma physics
  - radiation diffusion, transport
  - magnetohydrodynamics, space weather
  - combustion
  - solids, fracture, shear bands
  - astrophysics, cosmology
  - electromagnetics, semiconductors
  - complex geometry in engineering problems
  - many others...

- Adaptive methods use either unstructured or structured meshes
  - All start with coarse base grid and add local refinement as needed
  - Mathematical challenges: numerical approximation, error estimation
  - Computational challenges: optimize dynamic load balance, data (re)distribution, complex communication (overheads cannot be amortized)
  - Details are specific to choice of AMR methodology
Unstructured adaptive mesh refinement

- Typically used with FE, FV methods
  - good for complex geometries
  - mathematically rigorous error est. (FEM)

- AMR mesh & std. unstructured mesh similar
  - AMR mesh typically a “single-level” mesh
  - often relies on mesh “smoothing” to ensure mesh/element quality (elt angle size, elt aspect ratio, etc.)
  - mesh partitioning algorithms (graph-based or geometry-based) used to assign mesh points to processors

- Key parallel AMR issues
  - tradeoff between dynamic mesh/data migration & load balance
    - mesh/data repartitioned entirely or partially during (de)refinement
    - communication hard to factor into load balance
Cell-by-cell (structured) mesh refinement

- Used with FE, FV, or FD methods
  - structured multi-level solvers
  - facilitates local timestepping

- AMR mesh organized as a hierarchy
  - hierarchy of nested levels
  - typically uses small, same-sized blocks
  - typically uses octree (quadtree in 2d) to maintain parent-child cell relationships

- Key parallel AMR issues
  - distribution of tree structure → potentially high storage, bookkeeping overhead
  - tradeoff between load balance & interprocessor data communication → split parents & children across procs?
  - tradeoff between block size and communication complexity/cost
Space filling curves are a useful tool to facilitate data locality & load balance

- Space filling curve is a mapping: \( N^d \rightarrow N^1 \)
- Goal is to optimize locality in a global sense
  - numerical algorithm locality \( \rightarrow \) cell index locality \( \rightarrow \) data locality
- Enables dynamic load balance & reduces interprocessor comm in AMR (Browne & Parashar -- DAGH/GrACE)
  - refined regions “inserted” into curve
  - recursively filled w/ curve of same structure
Patch-based structured mesh refinement

- Used with FD, FV methods
  - structured multi-level solvers
  - facilitates local timestepping, legacy numerical kernels

- AMR mesh organized as a hierarchy
  - hierarchy of nested levels
  - cells clustered into “patches” of varying size → increased comp/comm volume ratio
  - simple model of data locality (no indirection)
  - low overhead mesh description

- Key parallel AMR issues
  - cell “clustering” scalability
  - tradeoff between patch size and communication complexity/cost & load balance
  - small patches more flexible for LB, but communication overhead greater

SAMRAI project (LLNL)
www.llnl.gov/CASC/SAMRAI
AMR helps resolve complex geometry

CART3D, Berger (NYU), Aftosmis et al (NASA) generates AMR meshes to resolve features around embedded boundaries
http://people.nas.nasa.gov/~aftosmis/cart3d/

Griffith, Peskin (NYU) developing electrical-mechanical heart model combining immersed boundaries and AMR (SAMRAI)
ALE-AMR combines ALE integration with AMR

- Advantages of ALE (multiple materials, moving interfaces)
- Advantages of AMR (dynamic addition & removal of mesh points)

Adaptive Mesh and Algorithm Refinement (AMAR) refines mesh and numerical model

- AMR is used to refine continuum calculation
- Algorithm switches to discrete atomistic method to include physics absent in continuum model

Particles resolve molecular-scale dynamics of mixing region in adaptive shear layer calculation

Presentation Outline

● Overview of adaptive mesh refinement (AMR) approaches
  — General issues
  — Unstructured AMR
  — Cell-by-cell structured AMR
  — Patch-based structured AMR
  — Hybrid AMR methods

● SAMRAI algorithm development to improve parallel scalability


● Concluding remarks
Box calculus is fundamental to patch-based structured AMR

- Patches are distributed among processors one level at a time

1) Box regions generated by “clustering” tagged cells

2) Boxes “chopped” to form patch regions

3) Patches mapped to processors

- Serial numerical routines operate on patches after boundary data exchange

```plaintext
exchange patch boundary data
forall patches i in level X
    call do_some_work (X(i))
end forall
```

parallel data communication

“parallel” loop

serial code
Communication schedules create data dependencies from global mesh information

- Cost of creating send/receive sets can be amortized over multiple communication cycles

- Data from various sources packed into single message stream
  - supports arbitrary, variable-length data types
  - two sends per processor pair (low latency)

```
packStream(...);
```
Our experience: data communication efficient; adaptive meshing operations may scale poorly.
Communication schedule construction

- Data dependencies between patches determined by identifying box intersections
- Original implementation compared each destination patch with all potential source patches (not just neighbors)
- Complexity $O(N^2)$, $N =$ number of patches
- $N$ grows proportionally with problem size

Communication schedule construction costs grow dramatically with problem size
Recursive Binary Box Tree (RBBT) efficiently describes spatial relationships

- Build a tree containing spatial relationships between boxes, based on their corner indices
- boxlist is a list of boxes; d is a direction x, y, or z.

**ConstructRBBTNode**(boxlist B, direction d)

1. Compute a bounding box around B
2. Bisect bounding box along d axis - form L,R parts
3. Form 3 new boxlists:
   - \( B_L \) = boxes entirely contained in L part
   - \( B_R \) = boxes entirely contained in R part
   - \( B_{\text{node}} \) = boxes lying on interface
4. Form child nodes by recursing \( B_L, B_R \) with same d. Form “secondary” tree by recursing \( B_{\text{node}} \) with new direction d.
Use RBBT in communication schedule construction

- Fast determination of which boxes intersect (limits box comparisons to neighbors)

- **Example:** Is box B in the neighborhood of box A?
  - “Walk the tree” – is B inside A’s bounding box? If so, recurse to child node.
    - Last node in the recursion provides a small subset of boxes that *may* intersect A
    - Then we apply naïve $O(M^2)$ comparison algorithm, $M$ is number of neighbors of A

- **Complexity analysis:**
  - Setup: $O(N \log(N))$, done once for each level & reused
  - Query (for each box): $O(\log(N))$
  - Runtime complexity: $O(N \log(N))$ – varies with box layout, but valid for nearly all cases encountered
Non-scaled Sedov problem – same problem size on all processor counts

3D Sedov shock - Euler hydrodynamics

- 4 levels
- Refine ratio = 4 between levels
- Total work increases during simulation
- Per-processor workload decreases as number of processors increased

![Graph showing problem size on each level and number of gridcells over simulation time.](image-url)
Scaling of non-scaled Sedov problem

Non-scaled
4 level Sedov Problem
ASCI IBM Blue Pacific

Original

With new algorithms
**Scaled linear advection problem – problem size increased with proc count**

3D sinusoidal front – linear advection
- 3 levels
- Refine ratio = 4 between levels
- Per-processor workload remains roughly constant as number of processors is increased
- Computationally simple, cheap numerical kernels → high relative cost of adaptivity, communication

Number of Patches on Finest Level

![Graph showing the number of patches on the finest level over simulation time for different processor counts.](image)
Scaling of scaled linear advection problem

Scaled
3 level linear advection problem
LLNL Linux MCR Cluster

Original

With new algorithms

Wallclock Time vs. Processors

Ideal
Total
Time Advance
Adaptive Gridding
Other
We are developing new algorithms to further improve scaling

- Most “AMR overhead” currently scales as $O(N)$ for storage and $O(N \log N)$ for operation complexity, $N =$ number of patches
- New approaches necessary for $O(10K-100K)$ processors (Blue Gene/L)
  - Asynchronous point clustering algorithm parallelizes standard “serial” algorithm (Berger-Rigoutsos)
    - Current implementation reduces collective communication, but still serializes operations on independent subdomains
    - New implementation overlaps communication with computation to reduce processor wait time
  - “Distributed box graph” approach reduces AMR mesh description to include only local patch and neighbor patch information
    - Recast all global domain operations in terms of local and nearest neighbor operations
    - Use previously discarded incidental data to replace globally computed data (i.e., recycle graph information at regrid steps)
Presentation Outline

- Overview of adaptive mesh refinement (AMR) approaches
  - General issues
  - Unstructured AMR
  - Cell-by-cell structured AMR
  - Patch-based structured AMR
  - Hybrid AMR methods
- SAMRAI algorithm development to improve parallel scalability
- Concluding remarks
Concluding remarks

- AMR is an important technology for large-scale science & engineering problems that exhibit behavior requiring fine local grids to resolve
- AMR dynamically focuses computational effort to avoid cost of globally fine grid
- There are a variety of AMR approaches aimed at different problems, soln. algs
- Balance of work assigned to processors and cost of complex interprocessor communication are key scalability issues
  - load balance, data (re)distribution, complex data communication patterns
  - tradeoffs among these are dynamic
  - overheads that cannot be amortized
- AMR shows good scaling to O(1K) processors
- New developments show promise to scale to O(10K+) processors → reduce storage and operation overheads
- Combining techniques from different AMR approaches is beneficial
- Multiphysics problems that combine algorithms with different performance characteristics is a major challenge