Dynamic Adaptively Refined Mesh Simulations on 1M+ Cores

Brian T. N. Gunney

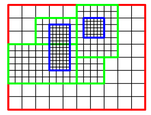
Center for Applied Scientific Computing, Lawrence Livermore National Laboratory

Supercomputing 2015, Austin, TX, 15-20 November 2015

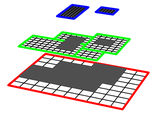
Regridding an adaptive mesh is very communication intensive. This poster presents two new parallel algorithms that allowed adaptive mesh refinement (AMR) simulations to scale to more than 1M cores.

SAMRAI Patch-based AMR background

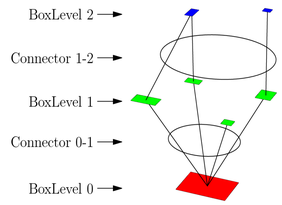
AMR mesh:



As BoxLevels:



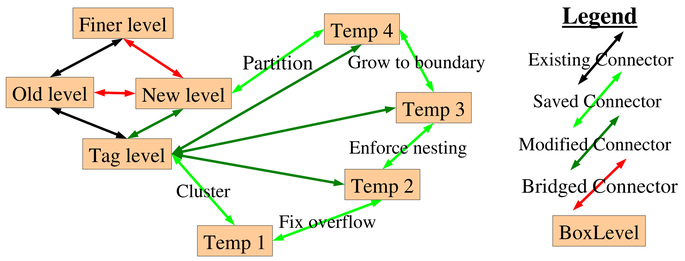
As proximity graph:



What is metadata?

- BoxLevel: Distributed sets of boxes.
- Connectors: Specify neighbor relationships between boxes.
- Important because solvers need proximity data to work.
- **Steps** to update a level: **cluster** tags into boxes, adjust boxes, **partition** and compute proximity.

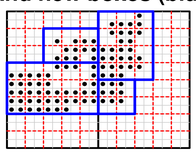
BoxLevels and Connectors are created and evolved together to update an old level. Saved Connectors contain information traditional approaches discard. Other Connectors are the results of bridge and modify operations [1], avoiding global searches. Result: New level complete with proximity data.



Tile Clustering Algorithm

Objective: Generate non-overlapping boxes covering all tags.
Strategy: Overlay tile mesh. Pick tiles that contain any tag. Find proximity using bridge and modify operations [1].

Tags (dots), tiles (red) and new boxes (blue)



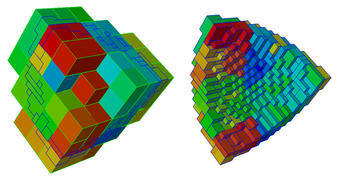
Opposing needs:

- Reduce overhead → big tiles
- Reduce untagged cells → small tiles
- Improve data locality → big tiles

Solution:

- Use small tiles, but coalesce into big clusters.
- Benchmarks averaged up to 38 tiles/box.

Coalesced tiles: (color = owner)



Cascade Partitioning Algorithm

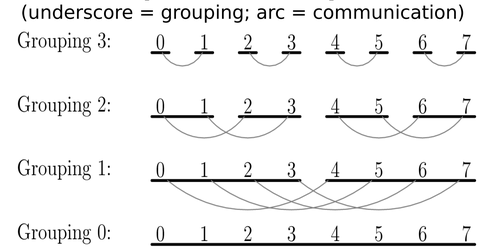
Objective: Compute new mesh distribution, minimizing load imbalance while preserving data locality.
Strategy: Balance weights in each half of machine, then balance halves recursively.

Cascade partitioner for balancing halves

1. Determine weight of each half, using lg(N) messages per process.
2. Compute how much work a process in the heavy half should give up.
3. Determine send/receive pairs for inter-group load transfer.
4. Apportion and send (heavy half) or receive load (light half).

- Processes are grouped by recursive bisection. Groups form a binary tree.
- Pair-wise communications give each process the load of each group it is in and of the sibling groups.
- Load transfer is done by Grouping 1.

8-process example
(underscore = grouping; arc = communication)



Who donates what amount? Processes closer to receiving side have donor priority. Donate as much work as needed up to max locally available.

Preserving data locality: Avoid breaking boxes until required.

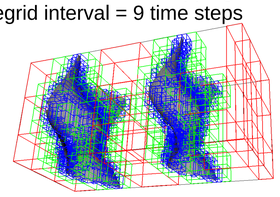
Typical maximum overloads achieved were 10% of average workload.
Color grouping in figures indicates data locality.

Scaling Benchmarks

Designed to be challenging for dynamic adaptivity: low computation; high regrid frequency

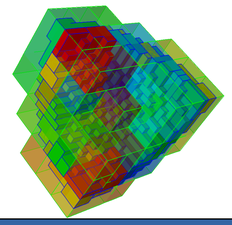
Weak scaling

- Sinusoidal wavy walls in linear advection
- Mesh size increases by domain tiling
  - 22.2x10^3 cell/core for 3 levels
  - 195x10^3 cell/core for 4 levels
  - 1.64x10^6 cell/core for 5 levels
- Refinement ratio = 3. Tile size = 9x9x9.
- 1 unknown/cell
- Forward Euler time stepping
- Regrid interval = 9 time steps



Strong scaling

- Spherical shock wave; Inviscid hydrodynamics.
- Fixed global mesh size
  - 8.41x10^9 global cells at 4-levels
- Refinement ratio = 4. Tile size= 8x8x8.
- 5 unknowns/cell
- Forward Euler time stepping
- Regrid interval = 8 time steps

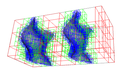


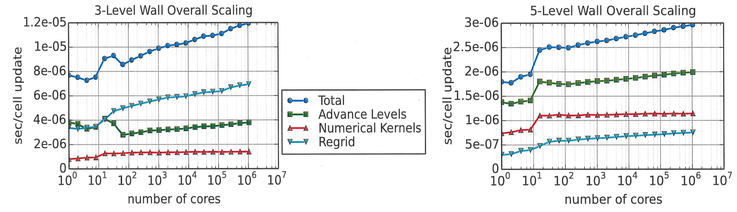
[1] Brian T. N. Gunney. Scalable Mesh Management for Patch-based AMR. In *NECDC 2012 Proceedings*, October 2012.

Overall Scaling Performance

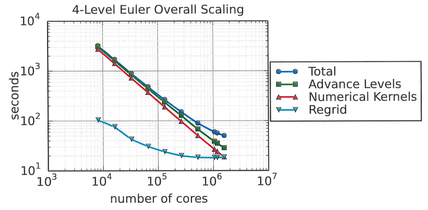
Run on LLN's Sequoia (IBM BG/Q)

Weak scaling
wall benchmark
1M cores
2M tasks



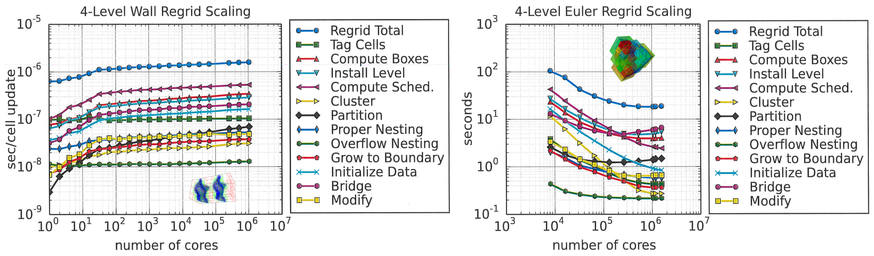


Strong scaling
Euler benchmark
1.5M cores

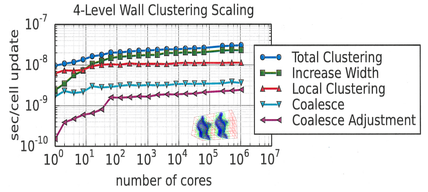


Regrid Scaling Performance

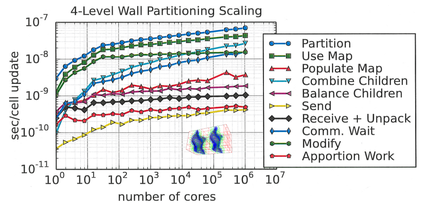
Challenging for dynamic adaptivity: any non-scaling component will eventually affect overall scaling!



Clusterer Scaling Performance



Partitioner Scaling Performance



Summary and Conclusion

- Clustering and partitioning algorithms were the two final components of scalable AMR.
  - Clustering algorithm significantly reduced metadata overhead without incurring excessive refinement.
  - Partitioning algorithm limited imbalance to 10%.
  - Both new algorithms scaled well individually and contributed to overall benchmark scaling.
- Weak and strong scaling benchmarks showed scalability to 1M+ cores.
- Our benchmarks were designed to be challenging. Applications with more computation and/or less regridding would scale even better.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-POST-674467