Forest-of-octrees AMR: algorithms and interfaces

Carsten Burstedde

joint work with

Omar Ghattas, Tobin Isaac, Georg Stadler, Lucas C. Wilcox

Institut für Numerische Simulation (INS)
Rheinische Friedrich-Wilhelms-Universität Bonn, Germany

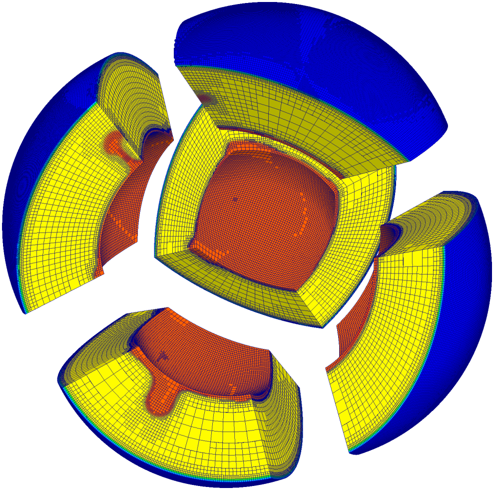
Institute for Computational Engineering and Sciences (ICES)
The University of Texas at Austin, USA

Feb 05, 2012

Second [HPC]^3 Workshop
KAUST, Saudi Arabia

Key points about AMR

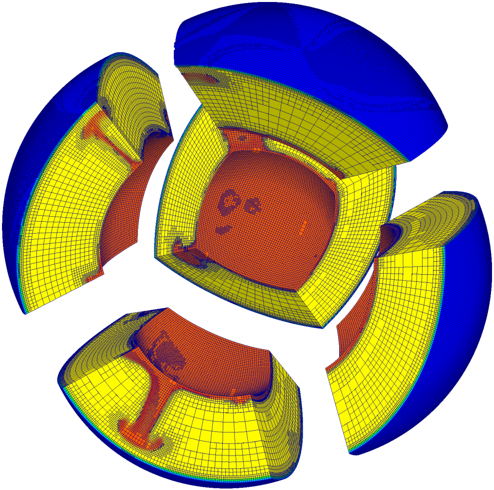
AMR—Adaptive Mesh Refinement



- local refinement
- local coarsening
- dynamic
- parallel
- (element-based)
- (general geometry)

Key points about AMR

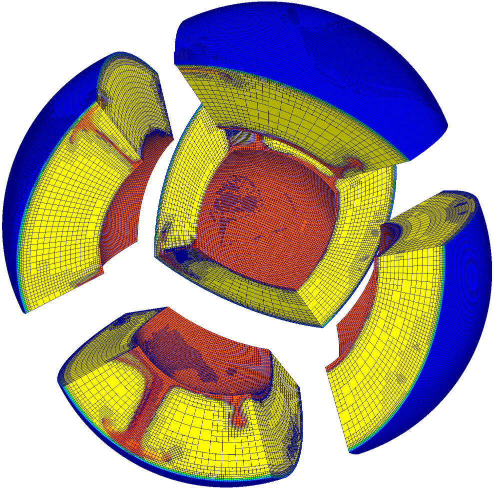
AMR—Adaptive Mesh Refinement



- local refinement
- local coarsening
- dynamic
- parallel
- (element-based)
- (general geometry)

Key points about AMR

AMR—Adaptive Mesh Refinement



- local refinement
- local coarsening
- dynamic
- parallel
- (element-based)
- (general geometry)

Why (not) use AMR?

AMR—Adaptive Mesh Refinement

Benefits (problem-dependent)

- Reduction in problem size
- Reduction in run time
- Gain in accuracy per degree of freedom
- Gain in modeling flexibility

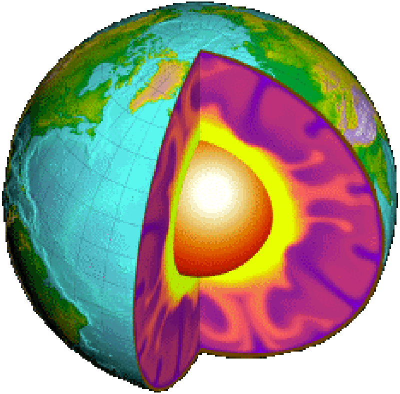
Challenges (fundamental)

- Storage: Irregular mesh structure
- Computational: Tree traversals and searches
- Networking: Irregular communication patterns
- Numerical: Horizontal/vertical projections

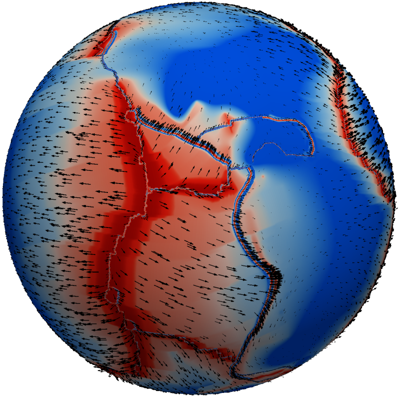
Geoscience simulations enabled by AMR

AMR—Adaptive Mesh Refinement

Mantle convection: High resolution for faults and plate boundaries



Artist rendering
Image by US Geological Survey

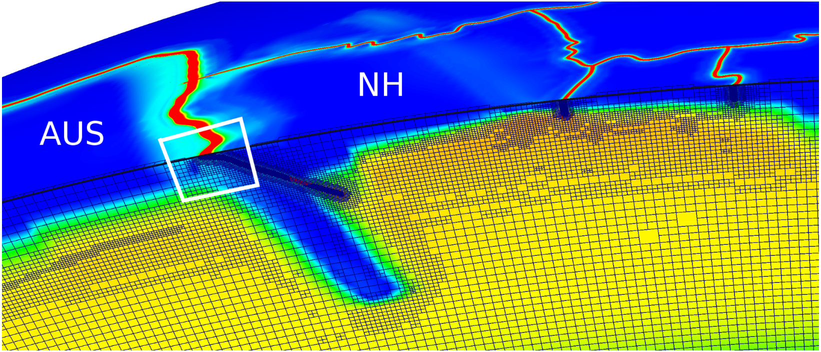


Simul. (w. M. Gurnis, L. Alisic, CalTech)
Surface viscosity (colors), velocity (arrows)

Geoscience simulations enabled by AMR

AMR—Adaptive Mesh Refinement

Mantle convection: High resolution for faults and plate boundaries

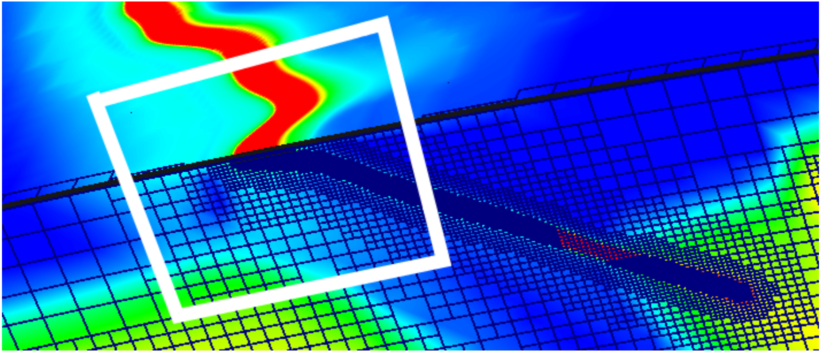


Zoom into the boundary between the Australia/New Hebrides plates

Geoscience simulations enabled by AMR

AMR—Adaptive Mesh Refinement

Mantle convection: High resolution for faults and plate boundaries

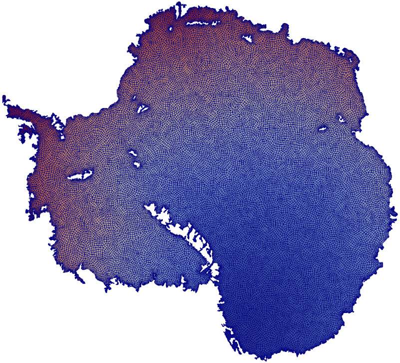


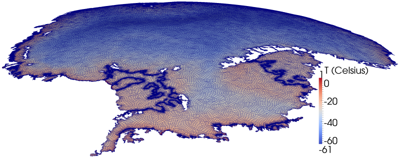
Zoom into the boundary between the Australia/New Hebrides plates

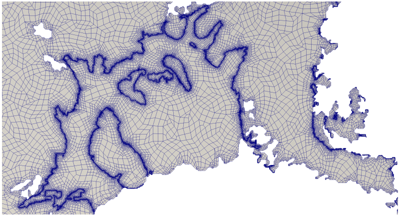
Geoscience simulations enabled by AMR

AMR—Adaptive Mesh Refinement

Ice sheet dynamics: Complex geometry and boundaries







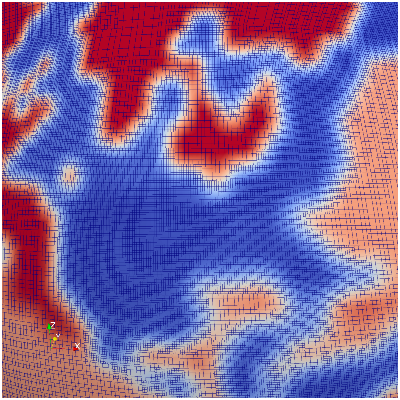
Antarctica meshes (w. C. Jackson, UTIG)
Adapt to geometry from SeaRISE data

Geoscience simulations enabled by AMR

AMR—Adaptive Mesh Refinement

Seismic wave propagation: Adapt to local wave length

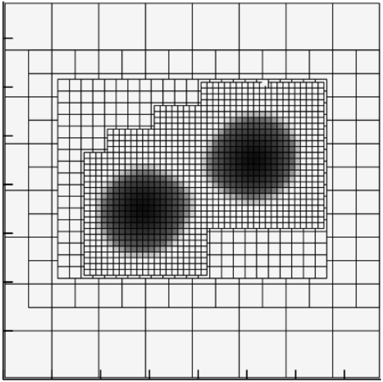
Varying local wave speeds



Adapt to local wave length

Types of AMR

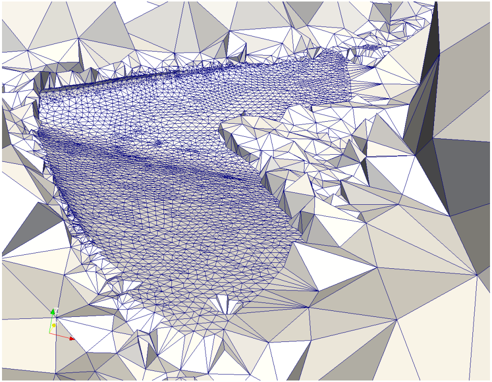
- Block-structured (patch-based) AMR



www.cactuscode.org

Types of AMR

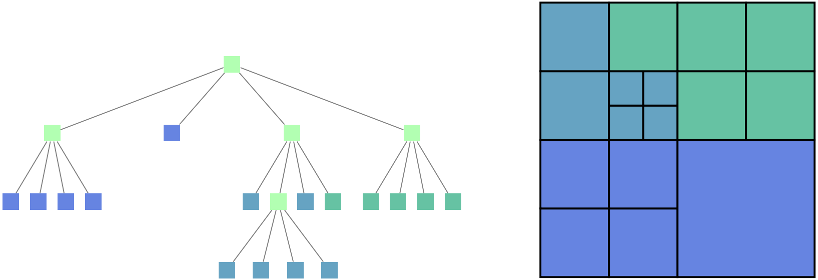
- Conforming tetrahedral (unstructured) AMR



mesh data courtesy David Lazzara, MIT

Types of AMR

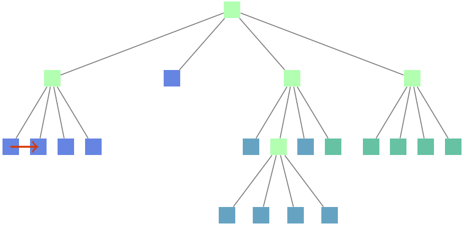
- Octree-based AMR

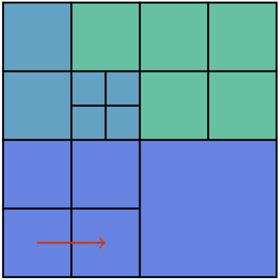


- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

- Octree-based AMR

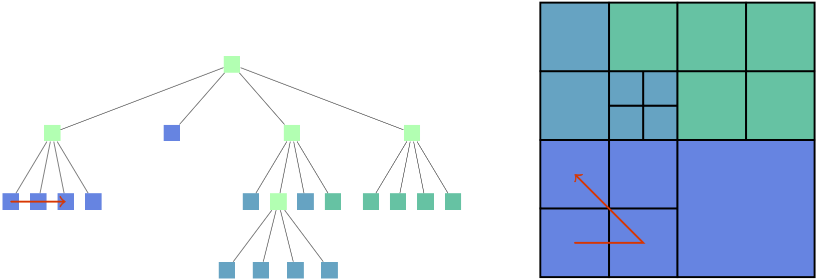




- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

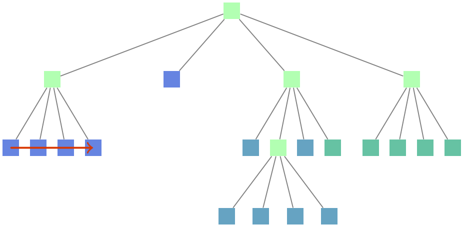
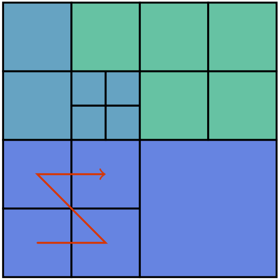
- Octree-based AMR



- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

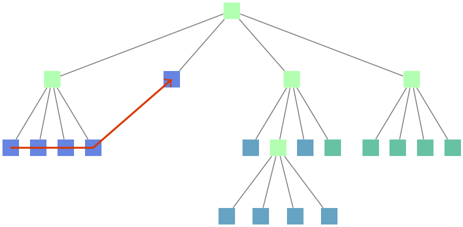
- Octree-based AMR

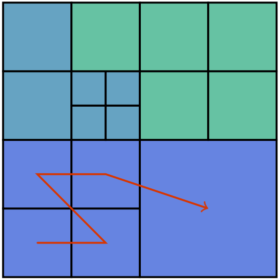



- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

- Octree-based AMR

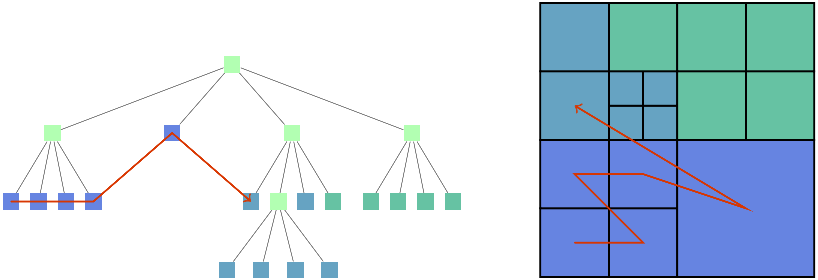




- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

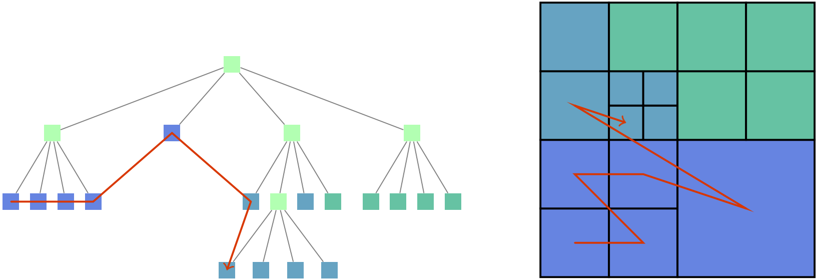
- Octree-based AMR



- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

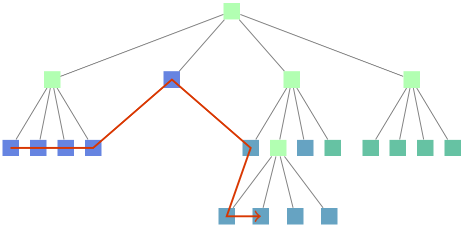
- Octree-based AMR

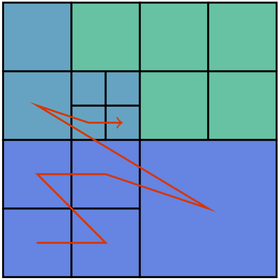


- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

- Octree-based AMR

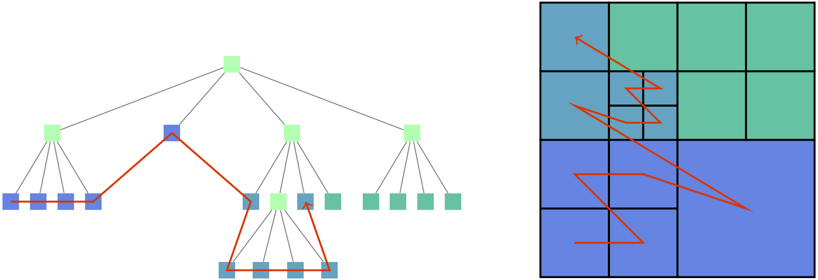




- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

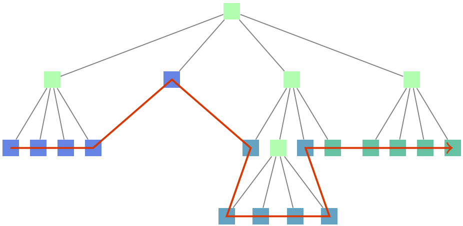
- Octree-based AMR

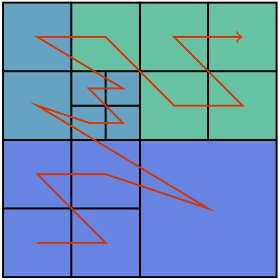


- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

- Octree-based AMR

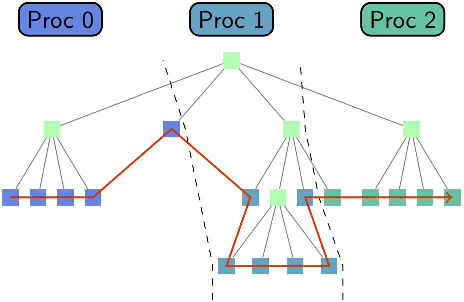


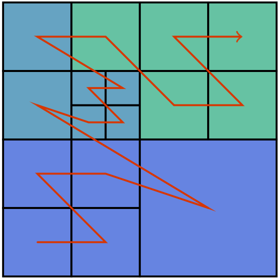


- Octree maps to cube-like geometry
- 1:1 relation between octree leaves and mesh elements

Types of AMR

- Octree-based AMR

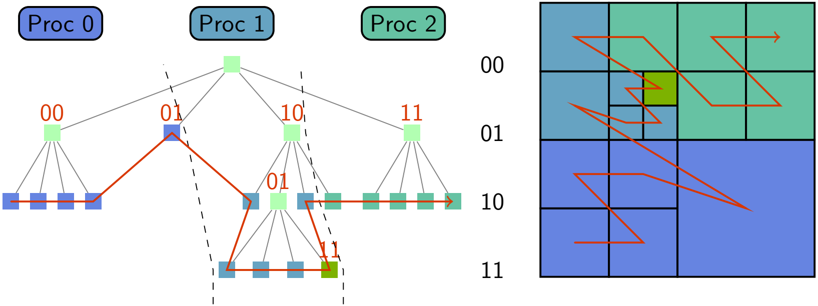




- Space-filling curve (SFC): Fast parallel partitioning
- Fast parallel tree algorithms for sorting and searching

Octree-based AMR

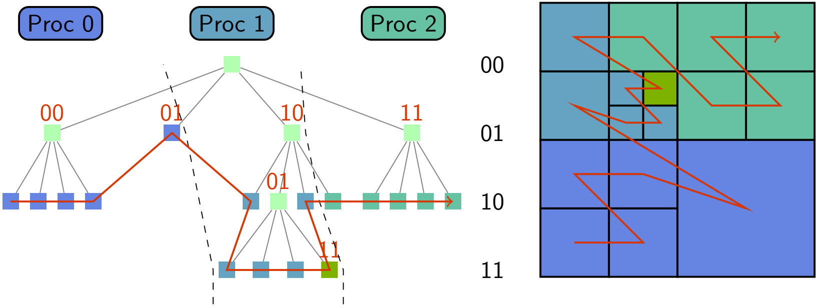
Efficient encoding and total ordering



- 1:1 relation between leaves and elements → efficient encoding
- path from root to node 10 01 11

Octree-based AMR

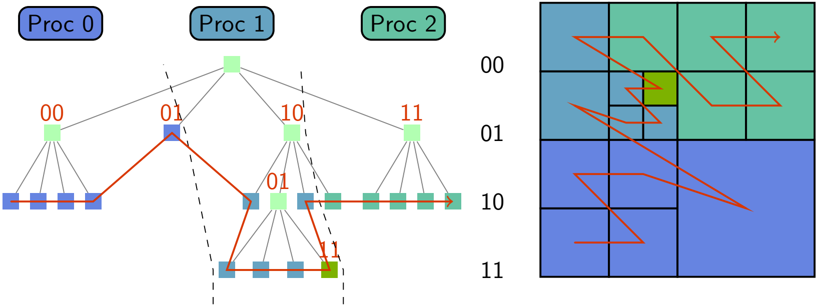
Efficient encoding and total ordering



- 1:1 relation between leaves and elements → efficient encoding
- path from root to node, append level `10 01 11 11` → `key`

Octree-based AMR

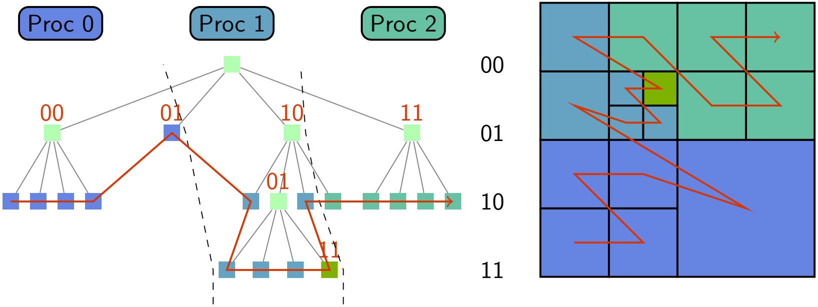
Efficient encoding and total ordering



- 1:1 relation between leaves and elements $\rightarrow$ efficient encoding
- path from root to node, append level `10 01 11 11` $\rightarrow$ `key`
- derive element x-coordinate `_0 _1 _1` $\rightarrow x = 3$

Octree-based AMR

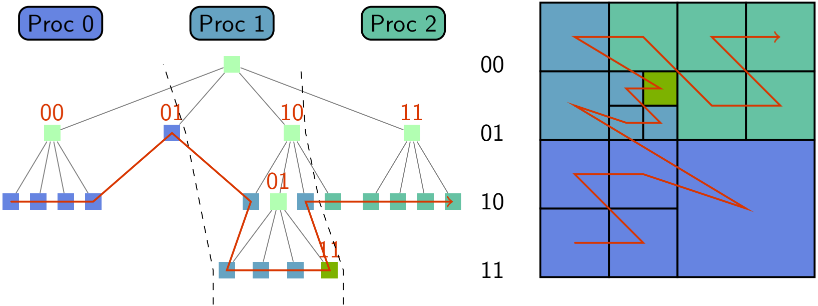
Efficient encoding and total ordering



- 1:1 relation between leaves and elements → efficient encoding
- path from root to node, append level 10 01 11 11 → key
- derive element x-coordinate _0 _1 _1 → $x = 3$
- derive element y-coordinate 1_ 0_ 1_ → $y = 5$

Octree-based AMR

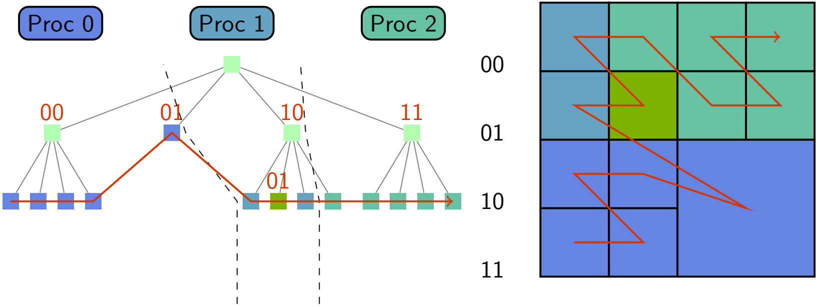
Fast elementary operations



- Construct parent or children $\rightarrow$ vertical tree step $\mathcal{O}(1)$
- path from root to node, append level `10 01 11 11` $\rightarrow$ `key`

Octree-based AMR

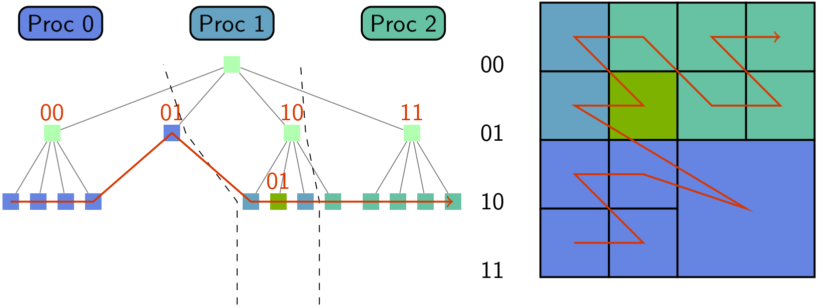
Fast elementary operations



- Construct parent or children → vertical tree step $\mathcal{O}(1)$
- path from root to node, append level `10 01 11 11`
- zero level coordinates, decrease level `10 01 00 10` → `key`

Octree-based AMR

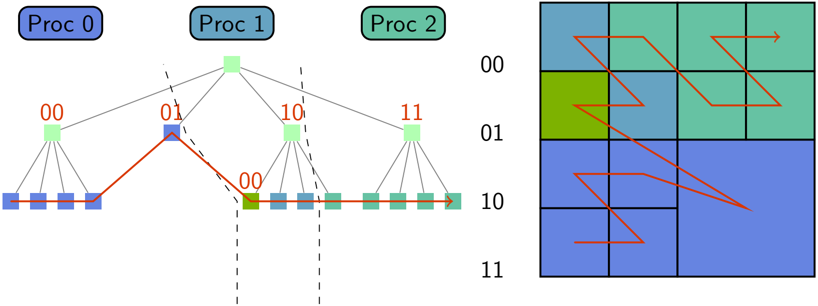
Fast elementary operations



- Construct neighbors → horizontal tree step/jump $\mathcal{O}(1)$
- path from root to node, append level `10 01 00 10` → `key`

Octree-based AMR

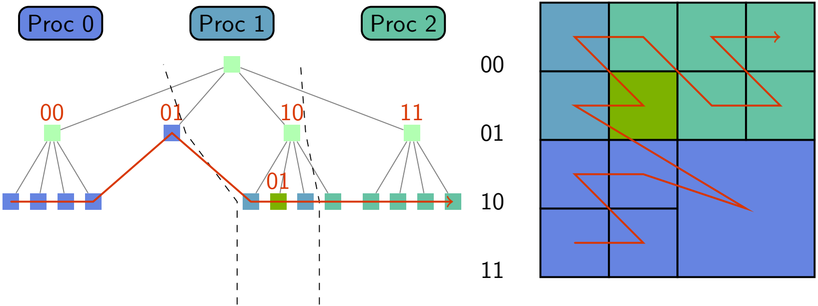
Fast elementary operations



- Construct neighbors → horizontal tree step/jump $\mathcal{O}(1)$
- path from root to node, append level 10 01 00 10
- Substract x-coordinate increment 10 00 00 10 → `key`
- Search on-processor element → tree search $\mathcal{O}(\log \frac{N}{P})$

Octree-based AMR

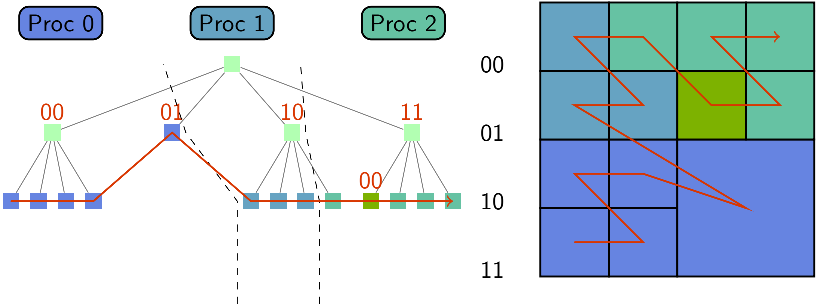
Fast elementary operations



- Construct neighbors → horizontal tree step/jump $\mathcal{O}(1)$
- path from root to node, append level `10 01 00 10` → `key`

Octree-based AMR

Fast elementary operations

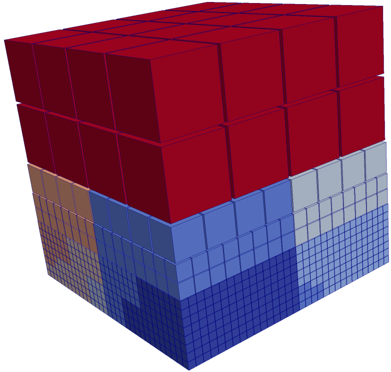


- Construct neighbors → horizontal tree step/jump $\mathcal{O}(1)$
- path from root to node, append level `10 01 00 10`
- Add x-coordinate increment `11 00 00 10` → `key`
- Search off-processor element-owner → search SFC $\mathcal{O}(\log P)$

Synthesis: Forest of octrees

From tree...

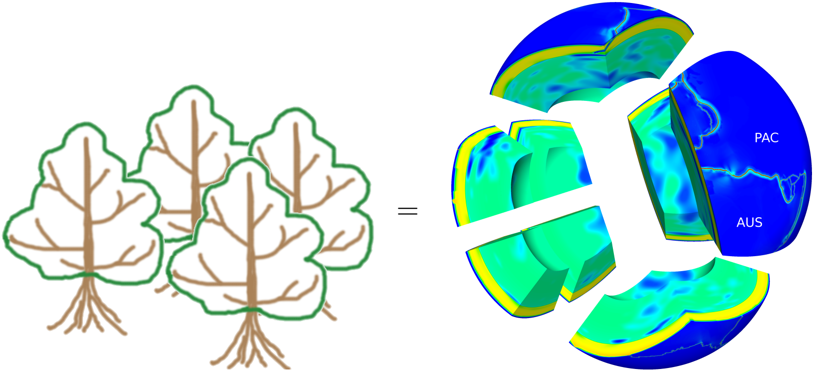
=



- Limitation: Cube-like geometric shapes

Synthesis: Forest of octrees

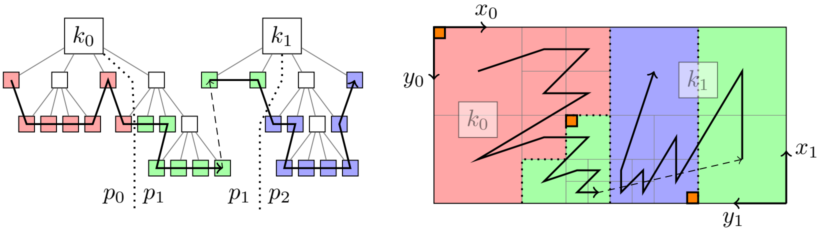
...to forest



- Advantage: Geometric flexibility
- Challenge: Non-matching coordinate systems between octrees

“p4est”—forest-of-octrees algorithms

Connect SFC through all octrees [1]



Minimal global shared storage (metadata)

- Shared list of octant counts per core $(N)_p$ $4 \times P$ bytes
- Shared list of partition markers $(k; x, y, z)_p$ $16 \times P$ bytes
- 2D example above $(h = 8)$: markers $(0; 0, 0), (0; 6, 4), (1; 0, 4)$

[1] C. Burstedde, L. C. Wilcox, O. Ghattas (SISC, 2011)

"p4est"—forest-of-octrees algorithms

p4est is a pure AMR module

- Rationale: Support diverse numerical approaches
- Internal state: Element ordering and parallel partition
- Provide minimal API for mesh modification

Connect to numerical discretizations / solvers ("App")

- p4est API calls are like MPI collectives (atomic to App)
- p4est API hides parallel algorithms and communication
- App $\rightarrow$ p4est: API invokes per-element callbacks
- App $\leftarrow$ p4est: Access internal state read-only

“p4est”—forest-of-octrees algorithms

p4est core API (for “write access”)

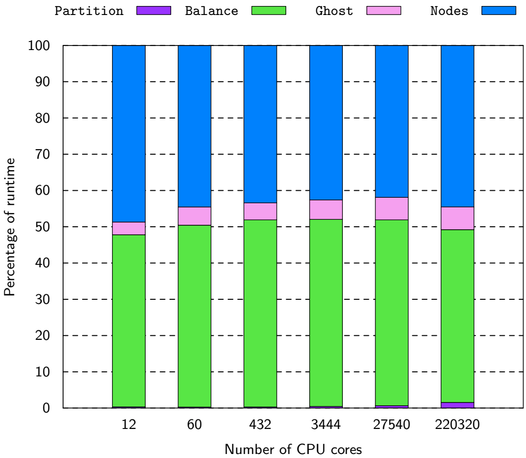
- p4est_new: Create a uniformly refined, partitioned forest
- p4est_refine: Refine per-element acc. to $0/1$ callbacks
- p4est_coarsen: Coarsen 2^d elements acc. to $0/1$ callbacks
- p4est_balance: Establish 2:1 neighbor sizes by add. refines
- p4est_partition: Parallel redistribution acc. to weights
- p4est_ghost: Gather one layer of off-processor elements

p4est “random read access” not formalized

- Loop through p4est data structures as needed

"p4est"—forest-of-octrees algorithms

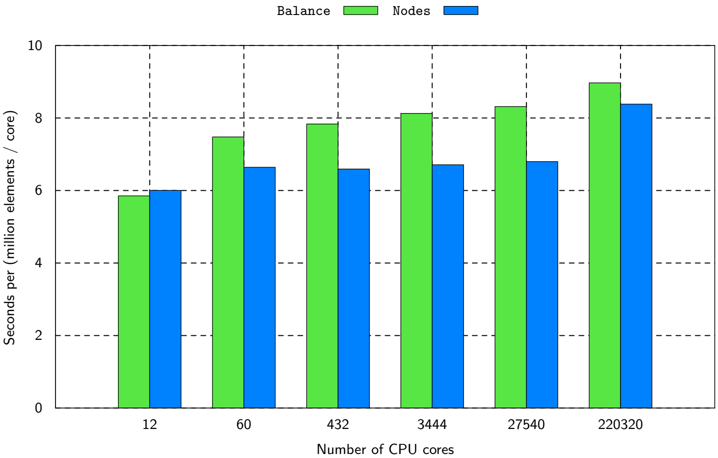
Weak scalability on ORNL's "Jaguar" supercomputer



- Cost of `New`, `Refine`, `Coarsen`, `Partition` negligible
- 5.13×10^{11} octants; < 10 seconds per million octants per core

"p4est"—forest-of-octrees algorithms

Weak scalability on ORNL's "Jaguar" supercomputer



- Dominant operations: `Balance` and `Nodes` scale over 18,360x
- 5.13×10^{11} octants; < 10 seconds per million octants per core

“p4est”—forest-of-octrees algorithms

What is a p4est element? Anything!

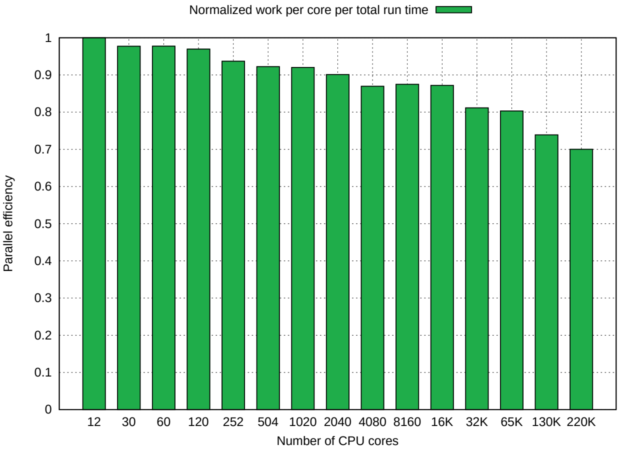
- The App defines how it will interprete an element

Examples

- Continuous bi-/trilinear elements
- High-order continuous spectral elements
- High-order DG elements with Gauss quadrature, LGL, ...
- An ijk subgrid optimized for GPU computation
- An M^d patch from PyClaw
- ...

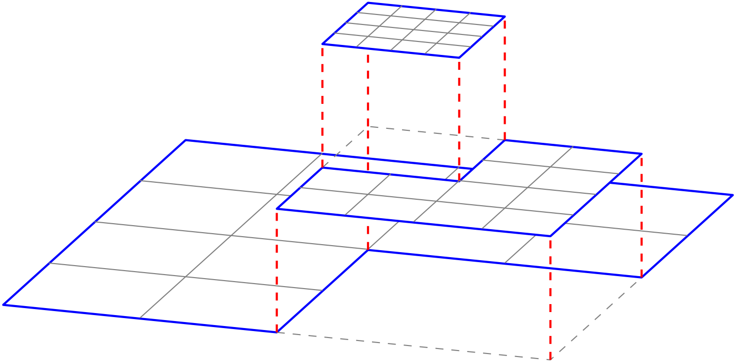
App: Dynamic-mesh DG (3D advection)

Weak scalability on ORNL's “Jaguar” supercomputer

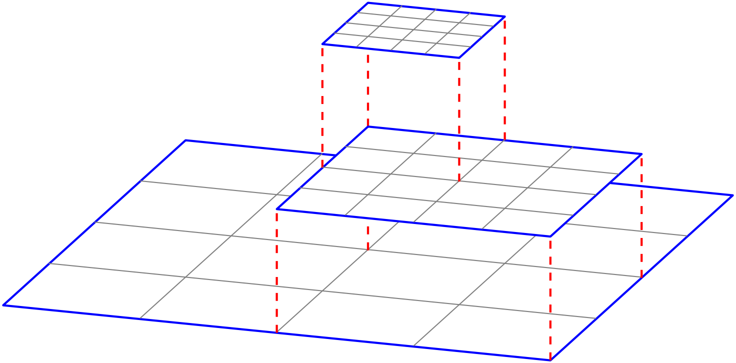


- 3,200 high-order elements per core from 12 to 220,320 cores
- Overall parallel efficiency is 70% over a 18,360x scale

Concepts related to patch-AMR



Concepts related to patch-AMR



Concepts related to patch-AMR

Differences

- SFC logical structure vs. unrestricted patch location
- Non-overlapping FE/DG allows arbitrary polynomial order
- Non-overlapping elements favor parallel efficiency
- Overlapping elements favor sharp CFL time step size

Best of both worlds?

- One leaf $\equiv$ One PyClaw patch: Reuse efficient math code
- Allow overlap $\equiv$ Allow data at non-leaf octree nodes
- No overlap: "Standard" FV or DG method
- Is local time stepping a requirement?
- Should we use implicit time stepping?

Acknowledgements

Publications

- Homepage: `http://burstedde.ins.uni-bonn.de/`

Funding

- NSF DMS, OCI PetaApps, OPP CDI
- DOE SciDAC TOPS, SC
- AFOSR

HPC Resources

- Texas Advanced Computing Center (TACC)
- National Center for Computational Science (NCCS)