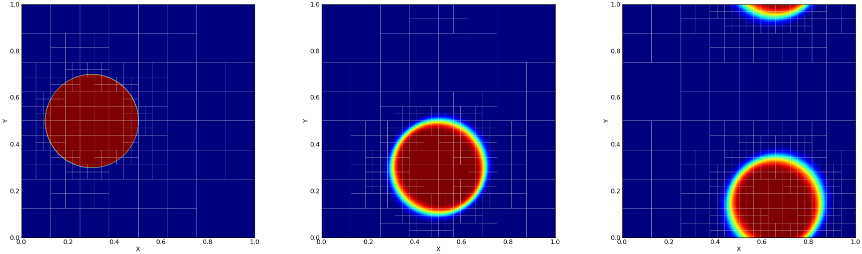
Scalable Algorithms for Distributed-Memory Adaptive Mesh Refinement

Akhil Langer*, Jonathan Lifflander*, **Phil Miller***, Kuo-Chuan Pan‡, Laxmikant V. Kale*, Paul Ricker‡

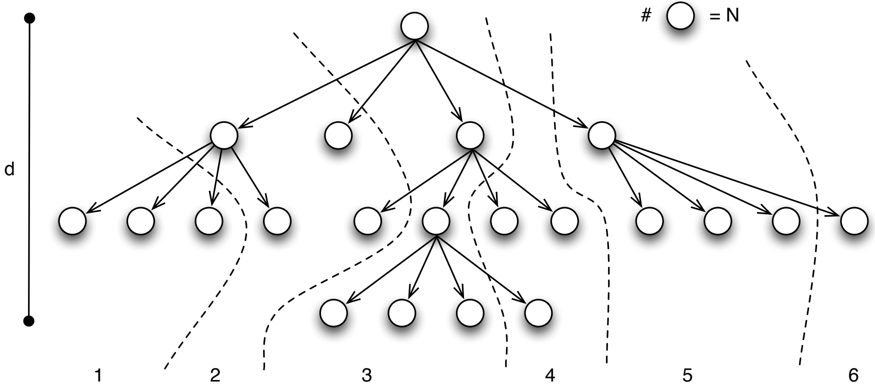
*Parallel Programming Laboratory, ‡ Astronomy
University of Illinois Urbana-Champaign

Thursday, 25 October 2012

Background on AMR



Background on AMR



Approach

Promote individual blocks to first-class entities, instead of processes

- Unit of algorithm expression
- Endpoint of communication

Naming

Give blocks *invariant, structure-determined names*

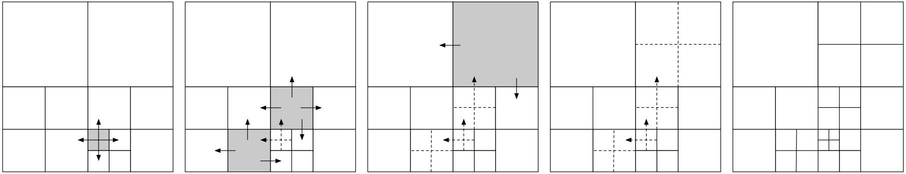
- Bitvector describing path from root to block's node
- One bit per dimension at each level
- Easy to compute parent, children, siblings

Finding Blocks

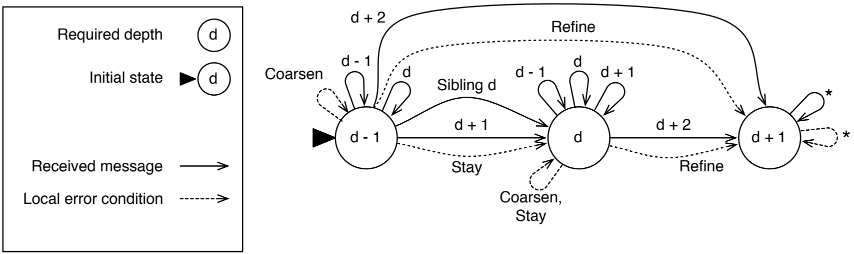
Each block has a unique *home PE* responsible for its location

- Locally computable, deterministic function of name (e.g. hash)
- Others ask home PE for current location
- Cache answers locally
- Responsibility roughly load-balanced
- Persistent $\mathcal{O}(P)$ distribution records obviated

Mesh Adaptation Algorithm

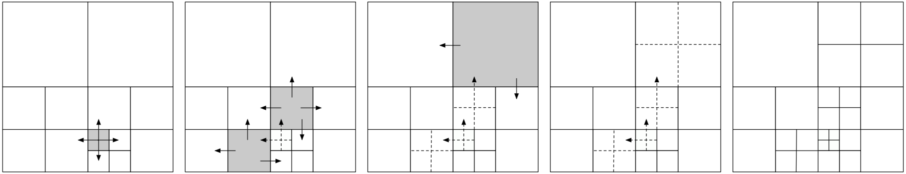


Mesh Adaptation Algorithm

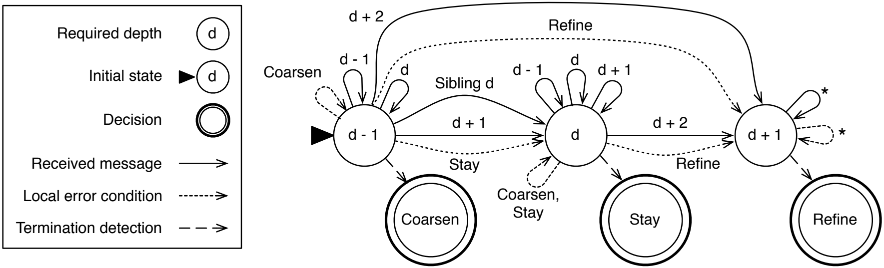


Mesh Adaptation Algorithm

When to stop?



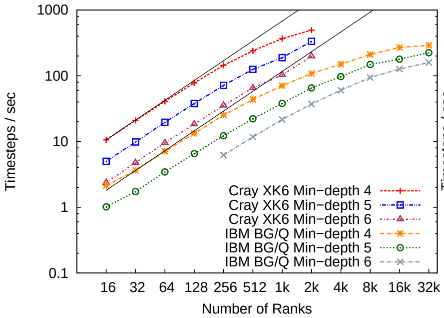
Mesh Adaptation Algorithm

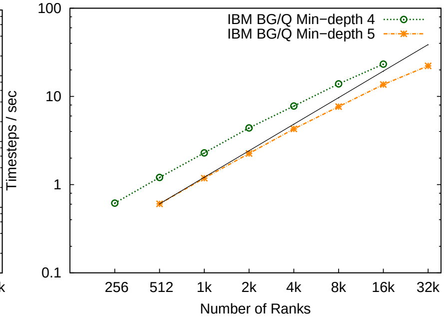


Termination Detection

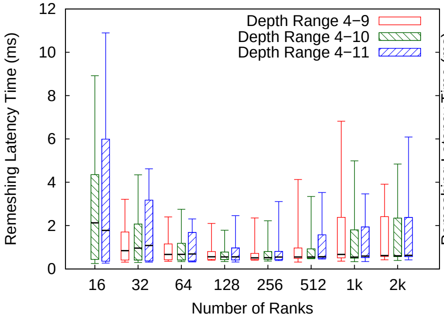
- Various classes of algorithms (wave, parental, credit)
- Theoretical bounds on each
- Practical cost is low
- Cost is *independent of dynamic range in refinement depth*

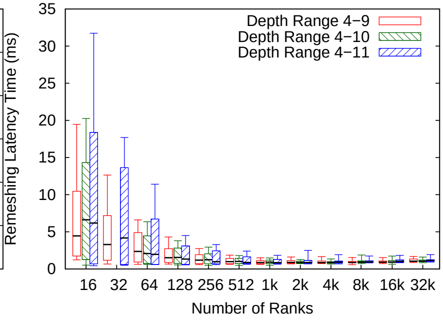
Overall Performance



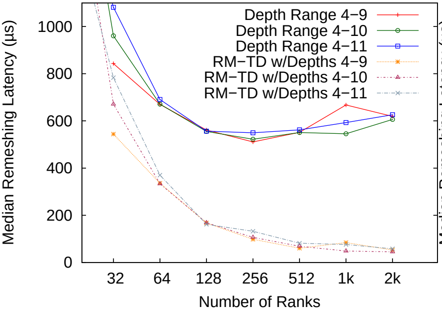


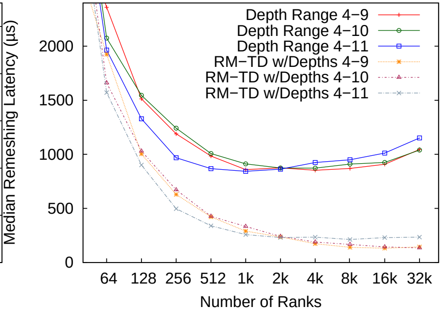
Remeshing Performance



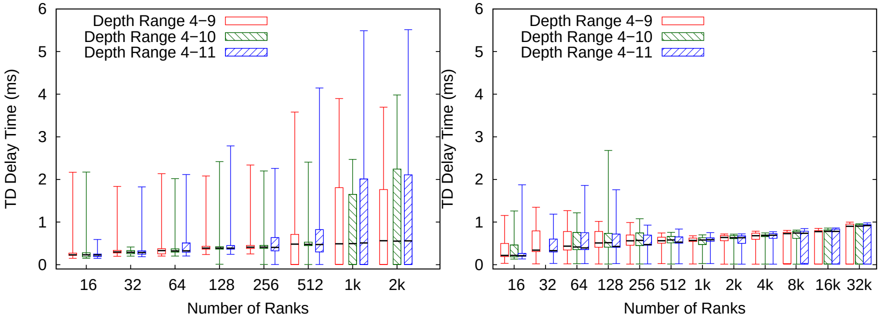


Remeshing Performance





Remeshing Performance



Conclusion

- Elevate blocks to first-class entities
- $\rightarrow$ No more $\mathcal{O}(P)$ data structures
- Adapt mesh using near-neighbor point-to-point messages & termination detection
- $\rightarrow$ No more memory-hungry collectives taking $\mathcal{O}(d \log P)$ time per cycle