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

Phil Miller (UIUC)
Background on AMR

![Diagram of AMR structure]

Phil Miller (UIUC) Scalable AMR Thursday, 25 October 2012

# \( = N \)
Approach

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

- Unit of algorithm expression
- Endpoint of communication
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 $O(P)$ distribution records obviated
Mesh Adaptation Algorithm
Mesh Adaptation Algorithm

Initial state

Required depth

Received message

Local error condition

d

Coarsen

d - 1

Stay

Sibling d

Coarsen, Stay

Refine

d + 1

Refine

d + 2

*
Mesh Adaptation Algorithm

When to stop?

![Diagram showing mesh adaptation stages](image)
Mesh Adaptation Algorithm

<table>
<thead>
<tr>
<th>Required depth</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial state</td>
<td>d</td>
</tr>
<tr>
<td>Decision</td>
<td></td>
</tr>
</tbody>
</table>

- Received message
- Local error condition
- Termination detection

**Diagram:**

- **d - 1:**
  - Coarsen
  - Sibling d
  - Stay

- **d:**
  - Coarsen, Stay
  - Refine

- **d + 1:**
  - Refine

**Decision Process:**

1. **Initial state:** d
2. **Required depth:** d
3. **Decision:**
   - Coarsen, Stay, Refine
4. **Termination detection:**
   - *
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

![Graph showing overall performance for Cray XK6 and IBM BG/Q with different depth configurations.](image)

- **Cray XK6**
  - Min-depth 4
  - Min-depth 5
  - Min-depth 6

- **IBM BG/Q**
  - Min-depth 4
  - Min-depth 5

The graphs illustrate the relationship between the number of ranks and the time steps per second, with different lines representing different depth configurations.
Remeshing Performance

Remeshing Latency Time (ms)
Number of Ranks
Depth Range 4–9
Depth Range 4–10
Depth Range 4–11

Remeshing Latency Time (ms)
Number of Ranks
Depth Range 4–9
Depth Range 4–10
Depth Range 4–11
Remeshing Performance

- Depth Range 4–9
- Depth Range 4–10
- Depth Range 4–11
- RM–TD w/Depths 4–9
- RM–TD w/Depths 4–10
- RM–TD w/Depths 4–11

Median Remeshing Latency (µs)
Number of Ranks
Depth Range 4–9
Depth Range 4–10
Depth Range 4–11
RM–TD w/Depths 4–9
RM–TD w/Depths 4–10
RM–TD w/Depths 4–11
Remeshing Performance

![Graph showing TD Delay Time vs Number of Ranks for different depth ranges.]

- Depth Range 4–9
- Depth Range 4–10
- Depth Range 4–11

**TD Delay Time (ms)**

**Number of Ranks**

**Depth Range 4–9**

**Depth Range 4–10**

**Depth Range 4–11**
Conclusion

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