Cluster-based parallelization of simulations on dynamically adaptive grids on the sphere

Martin Schreiber, Hans-Joachim Bungartz

Left: Simulation of the Tohoku Tsunami, right: earth-scale simulation
(developed in collaboration with Alexander Breuer, based on Augmented Riemann solvers [1])

Cluster-based parallelization

Clustering:
- **Multiple partitions** in a program's context
- **Replicated** shared hyperfaces
- Our clusters are generated by **subtrees** of the bisective Sierpinski **SFC-induced space tree** [2]

Intra-cluster communication:
- Based on **location-independent** stack communication system [3] (see e.g. Peano framework)

Inter-cluster communication:
- Use SFC supporting **RLE connectivity information**
- **Implicitly update** RLE connectivity information

Clustered parallelization (efficient) handling of multiple partitions in a single program context

[6] Run-length encoded (RLE) connectivity info. for edges and nodes

[7] Run-length encoded (RLE) connectivity info. for edges and nodes

openMP/TBB + MPI parallelization on cluster level [7]

Cluster-based optimizations [8]

Cluster-based data migration for load balancing [9]

Radial breaking dam on the sphere

Cluster skipping enabled

Hyperthreaded cores


Link to video:

https://www.youtube.com/watch?v=WllGOALQerI
Cluster-based parallelization of simulations on dynamically adaptive grids on the sphere

Martin Schreiber
schreibm@in.tum.de

Hans-Joachim Bungartz
bungartz@in.tum.de

Sierpinski Framework - http://www5.in.tum.de/sierpinski

Cluster-based optimizations:

- Skippable (conformity) adaptivity traversals (see above) [8]
- Local residual corrections (skip smoother)
- Compensate load imbalances with unpredictable flux computations, e.g. [4]

Applications

Tsunami simulation: Tohoku

En enabled in collaboration with Alexander Breuer, based on [4]

- (5 augmented Riemann solvers)

- Performance improvement of 6.6
- Dynamical adaptivity saves more than 95% of the cells
- (90070 cells in average per time step with reduced error)

Error and number of cells relative to the maximum value

<table>
<thead>
<tr>
<th>d</th>
<th>r</th>
<th>Error</th>
<th>Num of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.6</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>0.5</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>0.4</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1</td>
<td>0.1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Clustering allows skipping and reordering of the execution of operations:

- skipping of conformity adaptivity traversals (see above) [8]
- local residual corrections (skip smoother)
- compensate load imbalances with unpredictable flux computations, e.g. [4]

Cluster-based data migration [9]:

1. En bloc transfer of “raw” cluster data
2. Efficiently update RLE connectivity information

Example: Simulation of radial dam break on the sphere

Parallelization

Connectivity information (Sparse connect. graph)

- edges/nodes (solid line) [6]
- edges/nodes (dashed line) [7]

Dynamically changing grids [6]:

- Implicity update connectivity information by transferring adaptivity markers

Cluster generation [6]:

- During split and join, update connectivity data by information inferred from stacks

Applications

Shallow water, Tsunami, Euler, multi layer...

Data access patterns:

- Cell data (DG/FV simulation data, ...)
- Edge data (flux computations, limiter, adaptivity markers, ...)
- Node data (visualization, limiter, ...)