Parallel Implementation of MEMOSA-FVM

B. Gazi Yildirim and Sanjay R. Mathur
Purdue University

Objective: Parallelization of MEMOSA-FVM for multiprocessor and multicore platforms

- Unstructured finite volume algorithm
- Cell centered storage
- Implicit solution of multiple PDEs

Code Architecture

- Object Oriented C++ implementation
- Spatial decomposition into multiple "Mesh" objects
- "Field" objects store quantities over multiple Meshes
- Discretization operators designed to handle multiple Meshes and Fields

Approach

- Distribute Mesh objects over number of processors
- Partition single Mesh into multiple Mesh objects
- Implement synchronization of ghost cell values in Field
- Explicitly invoked during the solution process as needed
- Communicate data using MPI send/receive
- Rest of the solver code needs no changes
- Discretization in parallel with very good scaling
- Main challenge is linear solver performance
- Each processor has its own part of the global matrix and can do local Gauss-Seidel relaxation
- Need to sync solution after each relaxation sweep
- Creation of coarse levels for AMG
- Poor computation to communication ratio at coarse levels

Code Architecture

- PartMesh class partition (by PartMetis)
- prepare scatter and gather mappings
- make MEMOSA-FVM meshes
- support tri, quad, tet and hex meshes

Timing for 15.4M Tetrahedral mesh partitioning by PartMesh

- Timing results from Coates cluster AMD 2.5 GHz Quad Core 8 cores per node 1-10 GigE interconnect

Partitioning

- Coefficient based grouping of cells to create coarse matrices
- Initial coarsening only with cells on the same processor
- Need to communicate coarse indices to setup coarse ghost cells
- Coarse linear systems use the same data structures and thus can use same parallel execution mechanisms
- Below a certain threshold all coarse matrices moved to a single processor and merged
- Provides better coarsening
- Avoid large communication to computation costs

Algorithmic Multigrid (AMG) Parallelization

- 4M hexahedral mesh
- Good efficiency for fixed iteration count.
- Primary issue is increase of iteration count.

Results

<table>
<thead>
<tr>
<th>#Procs</th>
<th>Time (sec)</th>
<th>Speedup</th>
<th>Efficiency %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>151.3</td>
<td>1.0</td>
<td>100.0</td>
</tr>
<tr>
<td>4</td>
<td>76.1</td>
<td>2.0</td>
<td>100.0</td>
</tr>
<tr>
<td>8</td>
<td>37.0</td>
<td>4.1</td>
<td>102.0</td>
</tr>
<tr>
<td>16</td>
<td>16.8</td>
<td>9.1</td>
<td>113.8</td>
</tr>
<tr>
<td>32</td>
<td>8.66</td>
<td>17.3</td>
<td>108.1</td>
</tr>
<tr>
<td>64</td>
<td>5.35</td>
<td>28.3</td>
<td>88.4</td>
</tr>
<tr>
<td>128</td>
<td>2.87</td>
<td>56.7</td>
<td>88.8</td>
</tr>
<tr>
<td>256</td>
<td>2.05</td>
<td>73.8</td>
<td>87.6</td>
</tr>
</tbody>
</table>

MultiField and MultiFieldMatrix Structure

- Designed to handle multiple Meshes and/or Fields
- Individual matrices of base Matrix class
- Derived Matrix classes to handle variety of storage formats
- Compressed row format for unstructured meshes
- Templated to handle arbitrary arithmetic and coefficient types

BCGSTAB Scaling with AMG

- Agglomeration algorithm to keep number of iterations constant in AMG
- Multilevel parallel programming (hybrid MPI/OpenMP)
- Parallel IO
- Integration of SPIKE and TRILINOS solvers

Future Work

- 4M hexahedral mesh
- Good efficiency for fixed iteration count.
- Primary issue is increase of iteration count.