streamLES

- Multigrid: one of the key component of CDP.
- Used to accelerate the solution of the Poisson equation.
- Also:
  - implicit time-stepping
  - MG can be used to efficiently address stiffness in the physical coupling of equations (e.g. combustion source terms and their effect on momentum).
- So far: Poisson is accelerated using Multigrid.
  - has lead to a demonstrated reduction in computational expense.
- Our first approach to porting CDP and testing its performance on Merrimac.
## Status of MG in CDP

<table>
<thead>
<tr>
<th>Poisson Solver</th>
<th>Cost</th>
<th>Scalable</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCG</td>
<td>~2000 N</td>
<td>No</td>
<td>Old solver</td>
</tr>
<tr>
<td>Algebraic MG</td>
<td>~200 N</td>
<td>Yes</td>
<td>LLNL AMG already in production code</td>
</tr>
<tr>
<td>Geometric MG</td>
<td>~20-50 N*</td>
<td>Yes</td>
<td>Under development</td>
</tr>
</tbody>
</table>
Multigrid works!
Comparison of PCG and AMG

Full Pratt & Whitney Combustor Simulation,
35M CV on 480 proc (73,000 CV/proc)

~8x faster for Poisson solve,
Overall 4x speedup
GMG estimates

Speedup estimates based on 2-level GMG implemented in CDP
estimated speedup: ~20x for Poisson solve
Multigrid components

• Multigrid accelerates the solution by solving the problem at different grids: smooth components of the solution are efficiently solved on coarse grids, while fine grids handle high-frequency components.

• Challenging for Merrimac because:
  – Unstructured mesh, hence irregular data structure.
  – Complicated connectivity.
  – Varying number of neighbors.
  – Size of problem vary from one grid to the next: lots of parallelism on fine grids, little parallelism on coarse grid.
Algebraic and Geometric MG

- **Geometric:**
  - Based on the underlying mesh.
  - Works by coarsening the mesh geometrically.
  - Optimal efficiency.
  - Connectivity between cells remains low.
  - Difficulty to construct an optimal coarsening strategy.

- **Algebraic:**
  - Currently implemented in CDP. Uses HYPRE library of LLNL.
  - Do not use the underlying mesh. Works directly with the sparse matrix.
  - Completely general. “Black box.”
  - Often less efficient than geometric MG.
  - Connectivity increases on coarser grids. Stencils can become arbitrarily large.
  - Large memory requirement.
We want to solve:
\[ Au = f \]

3 essential components:

1. **Pre-smoothing**: local procedure which is able to rapidly “smooth” the error. Error can then be solved for on a coarser grid.

\[ d = f \quad A v \]
\[ e = u \quad v \]
\[ A e = d \]
2. **Restriction**: residual is restricted to the coarse grid. Solution is found approximately using Multigrid (*i.e.* we are defining the algorithm recursively).

3. **Interpolation**: correction is interpolated back to the fine grid.

4. **Post-smoothing.**
Geometric MG

Two-grid method:

Three-grid methods:

\[ \gamma = 1 \] \[ \gamma = 2 \] \[ \gamma = 3 \]

Four-grid methods:

\[ \gamma = 1 \] \[ \gamma = 2 \]

A five-grid method:

\[ \gamma = 2 \]
Parallel Implementation of GMG

• **Restriction** and **interpolation** can be naturally executed in parallel.

• **Smoothing** can be executed in parallel if appropriate smoother is chosen:
  – Jacobi smoother. Less efficient than others but very parallel.
  – Gauss-Seidel: lexicographic is very sequential.
Smoothing (cont’d)

• Multi-colored Gauss-Seidel:
  – Excellent smoothing.
  – Very parallel.
  – Requires multi-coloring: can this be implemented in parallel?
• Often a combination of those is used.
• For example in CDP for GMG:
  – Processor based Gauss-Seidel.
  – Degradation of performance but less communication.
Coarsening strategy

- Performance degrades if mesh is highly anisotropic.
- This is the case for Navier-Stokes at high Reynolds number: formation of a boundary layer.
- Semi-coarsening is used to regain good convergence: if cell is stretched in the $x$ direction, we coarsen along $y$.
- Procedure can be generalized to un-structured meshes: directional agglomeration technique of Frank Ham.
- Essential for good performance.
Programming issues

- Smoothing requires sparse matrix vector operations. Compressed Sparse Row can be used to operate on vectors.
  - Corresponds to a Gather
  - Reduction is local in this storage ($\neq$ CSC).
  - Stream can be organized such that number of neighbors is known in advance.

| 4 neighbors | 5 neighbors | 6 neighbors | 7 neighbors | 8 neighbors |
• Restriction:
  - Either: stream over fine nodes & distribute + reduce coarse nodes.
  - Or: group fine nodes. “group” needs to handle varying number of arguments.

• Interpolation:
  - Either: stream over coarse nodes & distribute + reduce fine nodes.
  - Or: group coarse nodes.

• Note difference with smoothing: size of output vector is different from input vector.

• How was this implemented in streamSPAS? [Jung Ho – Tim]
AMG: same as GMG but probably more challenging

Operator Matrix Information:

<table>
<thead>
<tr>
<th>lev</th>
<th>rows</th>
<th>nonzero</th>
<th>entries</th>
<th>sparse</th>
<th>min</th>
<th>max</th>
<th>entries per row</th>
<th>min</th>
<th>max</th>
<th>row sums</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>35148896 197671320</td>
<td>0.000</td>
<td>3</td>
<td>7</td>
<td>5.6</td>
<td>-3.553e-15</td>
<td>3.553e-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21851518 240568884</td>
<td>0.000</td>
<td>3</td>
<td>33</td>
<td>11.0</td>
<td>-5.454e-15</td>
<td>5.732e-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>11905829 334670059</td>
<td>0.000</td>
<td>3</td>
<td>140</td>
<td>28.1</td>
<td>-6.922e-15</td>
<td>6.923e-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6609555 422452067</td>
<td>0.000</td>
<td>3</td>
<td>453</td>
<td>63.9</td>
<td>-7.704e-15</td>
<td>8.305e-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3614134 461756434</td>
<td>0.000</td>
<td>4</td>
<td>1138</td>
<td>127.8</td>
<td>-9.851e-15</td>
<td>7.884e-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1901592 434218990</td>
<td>0.000</td>
<td>5</td>
<td>2057</td>
<td>228.3</td>
<td>-1.096e-14</td>
<td>1.058e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>961982 365133782</td>
<td>0.000</td>
<td>11</td>
<td>3276</td>
<td>379.6</td>
<td>-1.124e-14</td>
<td>9.388e-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>470676 282066006</td>
<td>0.001</td>
<td>21</td>
<td>4570</td>
<td>599.3</td>
<td>-1.521e-14</td>
<td>1.307e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>222927 201111273</td>
<td>0.004</td>
<td>17</td>
<td>6049</td>
<td>902.1</td>
<td>-1.703e-14</td>
<td>2.757e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>101651 128929059</td>
<td>0.012</td>
<td>19</td>
<td>6287</td>
<td>1268.4</td>
<td>-1.604e-14</td>
<td>2.843e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>44539 68913785</td>
<td>0.035</td>
<td>37</td>
<td>5307</td>
<td>1547.3</td>
<td>-4.171e-14</td>
<td>3.548e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>18425 27325507</td>
<td>0.080</td>
<td>15</td>
<td>4792</td>
<td>1483.1</td>
<td>-1.344e-13</td>
<td>8.503e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>6863 7549315</td>
<td>0.160</td>
<td>79</td>
<td>3128</td>
<td>1100.0</td>
<td>-1.434e-13</td>
<td>9.524e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2126 1215780</td>
<td>0.269</td>
<td>111</td>
<td>1620</td>
<td>571.9</td>
<td>-1.302e-13</td>
<td>7.497e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>533 138559</td>
<td>0.460</td>
<td>90</td>
<td>532</td>
<td>245.0</td>
<td>-1.236e-13</td>
<td>7.029e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>115 10731</td>
<td>0.811</td>
<td>58</td>
<td>115</td>
<td>93.3</td>
<td>-1.598e-13</td>
<td>6.598e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>28 782</td>
<td>0.997</td>
<td>27</td>
<td>28</td>
<td>27.9</td>
<td>-1.597e-13</td>
<td>6.576e-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>7 49</td>
<td>1.000</td>
<td>7</td>
<td>7</td>
<td>7.0</td>
<td>-1.212e-13</td>
<td>1.182e-13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Difficulties with AMG

• Stencil can become arbitrarily large, \textit{i.e.} matrix is not so sparse.
  – Anywhere between 4 to a thousand.
  – For GMG: typical connectivity is 4 to 16.

• Large memory requirement.
Coarse grids

Difficulty with very coarse grids:
• Number of nodes decreases.
• For multi-node configuration, we won’t have enough computational cells to keep all the nodes busy.
• Several remedies have been proposed in the literature:
  – Domain decomposition.
  – Additive Multigrid.
  – Parallel super convergent multigrid.