Parallel Sn for Unstructured Grids

with Cycle Detection

Steve Plimpton
Bruce Hendrickson
Shawn Burns
Kent Budge
Will McLendon, TAMU

TAMU Labfest -- May 2001
Parallel Sweeping Kernel

Unstructured 3d mesh -> directed graph.

Topological sort of graph -> lower-triangular sweep soln.
Parallel Sweep Algorithm

Work queue of computable tasks, task = cell/ordinate pair

count = # of upwind dependencies for each cell
put cells with count = 0 into queue (boundary cells)

while (undone work)
  while (cells in queue)
    remove cell from queue and solve it
    for each downwind dependency:
      if (I own cell)
        decrement count for downwind cell
        if (count = 0) add to queue
      else (another proc owns cell)
        SEND info to other proc

RECV all waiting messages
  decrement counts for my downwind cells
  if (count = 0) add to queue
Features of Parallel Sweep Algorithm

• Solves identical sweep equations as in serial:
  
  not block-Jacobi

• Multiple ordinates simultaneously:
  
  avoids some waiting by downwind procs

• Works for any decomposition of mesh:
  
  3-d spatial
  2-d columnar
  re-entrant processor boundaries

• “Tunable” parameters:
  
  # of ordinates, energies
  decomposition (an input)
  ordering of tasks within a proc’s sub-domain
  sizes of messages
Effects of Latency and Waiting for Work

- 3-d hex mesh with spatial decomposition
- 1 million unknowns (6360 elms, 80 ords, 2 energies)
- Fixed-size scalability
  -> Intel Tflops: Pentiums, fast message passing
  -> DEC CPlant: Alphas, slower message passing

- Ideal idea due to Shawn Pautz (LANL).
- Synchronous algorithm, assume “instant” messages.

- Tflops is OK, but what can improve the ideal curve?
Prioritization Scheme

• Priority = which cell a processor chooses to compute:

  Simple stack -> Priority queue.

  each cell/ordinate pair is assigned a priority value
  always compute highest-priority task available

• Priority = projection of cell onto ordinate direction

  work on my upwind cells first
  finish one ordinate before starting another
  propagates info to neighboring procs quicker
KBA-like Decomposition for Unstructured Grids

- Decomposition = assignment of grid cells -> procs

- Project grid cells to 2-d plane, do inertial decomposition

- Prioritize tasks along KBA columns
Effect of Two Improvements

- Priority queues
- KBA-like decomposition
- 4 million unknowns

- Boosts the ideal efficiency curve:
Parallel Scalability

- 3-d rectangular domain, irregular hexahedral grid:
  (80 ordinates, 2 energy groups)
Parallel Scalability

- Spherical domain, irregular hexahedral grid:
  (80 ordinates, 2 energy groups)
Grind Times

- 3-d rectangular domain, irregular hexahedral grid
- 6360 elements, 80 ordinates, vary energy groups
- Run on Tflops - 256 procs - Pentium II (333 MHz)

Grind time = CPU / elements / ordinates / energy-groups

<table>
<thead>
<tr>
<th>Energy Groups</th>
<th>Grind Time (usec)</th>
<th>Parallel Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>398</td>
<td>61.2%</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>59.8%</td>
</tr>
<tr>
<td>4</td>
<td>102</td>
<td>60.7%</td>
</tr>
<tr>
<td>8</td>
<td>52.8</td>
<td>59.0%</td>
</tr>
<tr>
<td>16</td>
<td>28.3</td>
<td>54.8%</td>
</tr>
<tr>
<td>32</td>
<td>16.3</td>
<td>49.6%</td>
</tr>
<tr>
<td>64</td>
<td>10.8</td>
<td>44.4%</td>
</tr>
<tr>
<td>128</td>
<td>8.37</td>
<td>40.6%</td>
</tr>
<tr>
<td>256</td>
<td>7.09</td>
<td>35.4%</td>
</tr>
</tbody>
</table>

- Parallel efficiency drop due to bandwidth cost of sending bigger messages (not just latency anymore).
Varying Number of Ordinates

- 3-d spherical domain, irregular hexahedral grid
- 32,000 elements, 2 energy groups, vary ordinates
- Run on Intel Tflops - 256 procs

- Ideal efficiencies approach 90%.
- Actual efficiencies approach 80%, but 3-D scheme becomes better than KBA-decomposition.
Cycles

- Graph is no longer a DAG:
  
  not a lower-triangular matrix
  no direct solve sweep
  parallel algorithm will hang!

- Detect cycles:
  
  non-trivial because graph (grid) is distributed

- Break cycles:
  
  delete edge (face) with minimal flux
Will McLendon (TAMU) Effort

• Asynchronous MPI code - quite complex!
  termination detection
  recursive
  multiple graphs simultaneously

• Masters thesis.
  paper presentation at SIAM - March 2001

• Gained another convert to MPI-style parallelism!

• Integrating Will’s module into sweep code.
  call whenever grid geometry changes
  cost of detection < single numerical sweep

• Sandia is very happy with this TAMU collaboration.