Cost-Effective Parallel Computational Electromagnetic Modeling

Daniel S. Katz, Tom Cwik
{Daniel.S.Katz, cwik}@jpl.nasa.gov
Beowulf System at JPL (Hyglac)

- 16 Pentium Pro PCs, each with 2.5 Gbyte disk, 128 Mbyte memory, Fast Ethernet card.
- Connected using 100Base-T network, through a 16-way crossbar switch.
- Theoretical peak: 3.2 GFLOP/s
- Sustained: 1.26 GFLOP/s
Beowulf System at Caltech (Naegling)

- ~120 Pentium Pro PCs, each with 3 Gbyte disk, 128 Mbyte memory, Fast Ethernet card.
- Connected using 100Base-T network, through two 80-way switches, connected by a 4 Gbit/s link.
- Theoretical peak: ~24 GFLOP/s
- Sustained: 10.9 GFLOP/s
Hyglac Cost

- **Hardware cost:** $54,200 (as built, 9/96)
  $22,000 (estimate, 4/98)
  - 16 (CPU, disk, memory, cables)
  - 1 (16-way switch, monitor, keyboard, mouse)
- **Software cost:** $600 ( + maintainance)
  - Absoft Fortran compilers (should be $900)
  - NAG F90 compiler ($600)
  - public domain OS, compilers, tools, libraries
Naegling Cost

- **Hardware cost:** $190,000 (as built, 9/97)
  - $154,000 (estimate, 4/98)
  - 120 (CPU, disk, memory, cables)
  - 1 (switch, front-end CPU, monitor, keyboard, mouse)

- **Software cost:** $0 (+ maintenance)
  - Absoft Fortran compilers (should be $900)
  - Public domain OS, compilers, tools, libraries
## Performance Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Hyglac</th>
<th>Naegling</th>
<th>T3D</th>
<th>T3E600</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Speed (MHz)</strong></td>
<td>200</td>
<td>200</td>
<td>150</td>
<td>300</td>
</tr>
<tr>
<td><strong>Peak Rate (MFLOP/s)</strong></td>
<td>200</td>
<td>200</td>
<td>300</td>
<td>600</td>
</tr>
<tr>
<td><strong>Memory (Mbyte)</strong></td>
<td>128</td>
<td>128</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Latency (μs)</strong></td>
<td>150</td>
<td>322</td>
<td>35</td>
<td>18</td>
</tr>
<tr>
<td><strong>Throughput (Mbit/s)</strong></td>
<td>66</td>
<td>78</td>
<td>225</td>
<td>1200</td>
</tr>
</tbody>
</table>

(Communication results are for MPI code)
Message-Passing Methodology

- Issue (non-blocking) receive calls:
  
  \[
  \text{CALL MPI_Irecv(\ldots)}
  \]

- Issue (synchronous) send calls:
  
  \[
  \text{CALL MPI_Ssend(\ldots)}
  \]

- Issue (blocking) wait calls (wait for receives to complete):
  
  \[
  \text{CALL MPI_Wait(\ldots)}
  \]
Time steps of a gaussian pulse, travelling on a microstrip, showing coupling to a neighboring strip, and crosstalk to a crossing strip. Colors showing currents are relative to the peak current on that strip. Pulse: rise time = 70 ps, freq. ≈ 0 to 30 GHz. Grid dimensions = 282 \times 362 \times 102 cells. Cell size = 1 mm^3.
FDTD Algorithm

- Classic time marching PDE solver
- Parallelized using 2-dimensional domain decomposition method with ghost cells.

Standard Domain Decomposition

Required Ghost Cells

Interior Cells
Ghost Cells
FDTD Algorithm Details

- Uses Yee’s staggered grid
- Time Stepping Loop:
  - Update Electric Fields (three 5-point stencils, on x-y, x-z, y-z planes)
  - Update Magnetic Fields (three 5-point stencils, on x-y, x-z, y-z planes)
  - Communicate Magnetic Fields to ghost cells of neighboring processors (in x and y)
FDTD Results

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>Naegling</th>
<th>T3D</th>
<th>T3E-600</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.44 - 0.0</td>
<td>2.71 - 0.0</td>
<td>0.851 - 0.0</td>
</tr>
<tr>
<td>4</td>
<td>2.46 - 0.097</td>
<td>2.79 - 0.026</td>
<td>0.859 - 0.019</td>
</tr>
<tr>
<td>16</td>
<td>2.46 - 0.21</td>
<td>2.79 - 0.024</td>
<td>0.859 - 0.051</td>
</tr>
<tr>
<td>64</td>
<td>2.46 - 0.32</td>
<td>2.74 - 0.076</td>
<td>0.859 - 0.052</td>
</tr>
</tbody>
</table>

Time (wall clock seconds / time step), scaled problem size (69 × 69 × 76 cells / processor), times are: computation - communication
FDTD Conclusions

- Naegling and Hyglac produce similar results for 1 to 16 processors
- Scaling from 16 to 64 processors is quite reasonable
- On all numbers of processors, Beowulf-class computers perform similarly to T3D, and worse than T3E, as expected.
PHOEBUS

Radiation Pattern from JPL Circular Waveguide
(from C. Zuffada, et. al., IEEE AP-S paper 1/97)

Typical Applications:

Radar Cross Section of a dielectric cylinder
PHOEBUS Coupled Equations

\[
\begin{bmatrix}
K & C & 0 \\
C^\dagger & 0 & Z_0 \\
0 & Z_M & Z_J
\end{bmatrix}
\begin{bmatrix}
H \\
M \\
J
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
V_{inc}
\end{bmatrix}
\]

- This matrix problem is filled and solved by PHOEBUS
  - The K submatrix is a sparse finite element matrix
  - The Z submatrices are integral equation matrices.
  - The C submatrices are coupling matrices between the FE and IE matrices.
PHOEBUS Solution Process

\[
\begin{bmatrix}
K & C & 0 \\
C^\dagger & 0 & Z_0 \\
0 & Z_M & Z_J
\end{bmatrix}
\begin{bmatrix}
H \\
M \\
J
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
V
\end{bmatrix}
\]

\[H = -K^{-1}CM\]

\[
\begin{bmatrix}
-C^\dagger K^{-1}C & Z_0 \\
Z_M & Z_J
\end{bmatrix}
\begin{bmatrix}
M \\
J
\end{bmatrix}
= \begin{bmatrix}
0 \\
V
\end{bmatrix}
\]

- Find \(-C^\dagger K^{-1}C\) using QMR on each row of \(C\), building \(x\) rows of \(K^{-1}C\), and multiplying with \(-C^\dagger\).
- Solve reduced system as a dense matrix.
PHOEBUS Algorithm

- Assemble complete matrix
- Reorder to minimize and equalize row bandwidth of $K$
- Partition matrices in slabs
- Distribute slabs among processors
- Solve sparse matrix equation (step 1)
- Solve dense matrix equation (step 2)
- Calculate observables
PHOEBUS Matrix Reordering

Original System

System after Reordering for Minimum Bandwidth

Non-zero structure of matrices, using SPARSPAK’s GENRCM Reordering Routine
PHOEBUS
Matrix-Vector Multiply

Communication from processor to left

Local processor’s rows

Communication from processor to right

Local processor’s rows
### PHOEBUS Solver Timing

**Model:** dielectric cylinder with 43,791 edges, radius = 1 cm, height = 10 cm, permittivity = 4.0, at 5.0 GHz

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>T3D (shmem)</th>
<th>T3D (MPI)</th>
<th>Naegling (MPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-Vector Multiply Computation</td>
<td>1290</td>
<td>1290</td>
<td>1502</td>
</tr>
<tr>
<td>Matrix-Vector Multiply Communication</td>
<td>114</td>
<td>272</td>
<td>1720</td>
</tr>
<tr>
<td>Other Work</td>
<td>407</td>
<td>415</td>
<td>1211</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1800</strong></td>
<td><strong>1980</strong></td>
<td><strong>4433</strong></td>
</tr>
</tbody>
</table>

Time of Convergence (CPU seconds), solving using 16 processors, pseudo-block QMR algorithm for 116 right hand sides.
PHOEBUS Solver Timing

Model: dielectric cylinder with 100,694 edges, radius = 1 cm, height = 10 cm, permittivity = 4.0, at 5.0 GHz

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>T3D (shmemb)</th>
<th>T3D (MPI)</th>
<th>Naegling (MPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix-Vector Multiply Computation</td>
<td>868</td>
<td>919</td>
<td>1034</td>
</tr>
<tr>
<td>Matrix-Vector Multiply Communication</td>
<td>157</td>
<td>254</td>
<td>2059</td>
</tr>
<tr>
<td>Other Work</td>
<td>323</td>
<td>323</td>
<td>923</td>
</tr>
<tr>
<td>Total</td>
<td>1348</td>
<td>1496</td>
<td>4016</td>
</tr>
</tbody>
</table>

Time of Convergence (CPU seconds), solving using 64 processors, pseudo-block QMR algorithm for 116 right hand sides.
PHOEBUS Conclusions

- Beowulf is 2.4 times slower than T3D on 16 nodes, 3.0 times slower on 64 nodes
- Slowdown will continue to increase for larger numbers of nodes
- T3D is about 3 times slower than T3E
- Cost ratio between Beowulf and other machines determines balance points
General Conclusions

- Beowulf is a good machine for FDTD
- Beowulf may be ok for iterative solutions of sparse matrices, such as those from Finite Element codes, depending on machine size
- Key factor: amount of communication