Beowulf Applications and User Experiences

Daniel S. Katz

Daniel.S.Katz@jpl.nasa.gov

JPL

High Performance Computing Group

Imaging and Spectrometry Systems Technology Section

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

Six applications analyzed in paper by Katz, et. al, Advances in Engineering Software, v.26, August 1998

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

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

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 (+ maintainance)
  - » Absoft Fortran compilers (should be $900)
  - » public domain OS, compilers, tools, libraries

Performance Comparisons

	Hyglac	Naegling	T3D	T3E600
CPU Speed (MHz)	200	200	150	300
Peak Rate (MFLOP/s)	200	200	300	600
Memory (Mbyte)	128	128	64	128
Communication Latency (µs)	150	322	35	18
Communication Throughput (Mbit/s)	66	78	225	1200

(Communication results are for MPI code)

Message-Passing Methodology

- Issue (non-blocking) receive calls:

```
CALL MPI_IRECV(...)
```

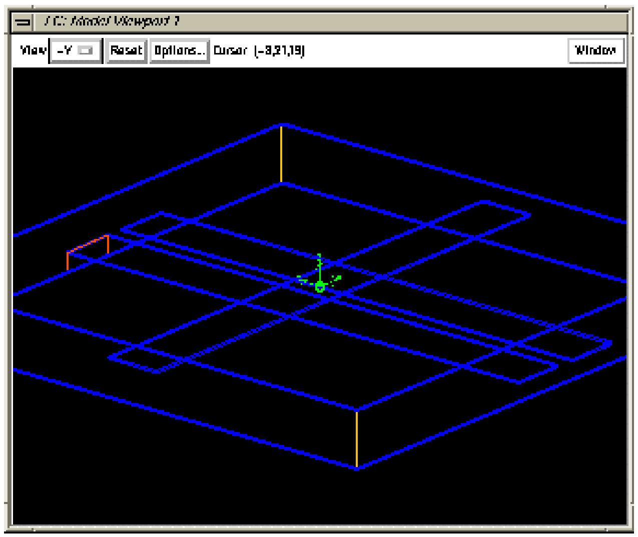
- Issue (synchronous) send calls:

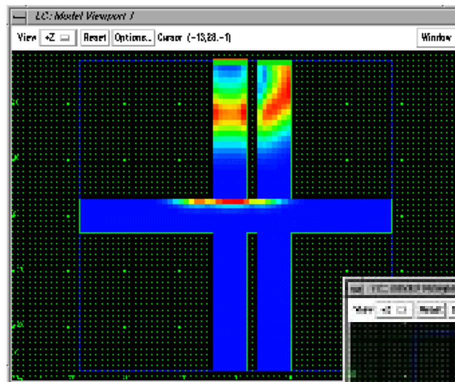
```
CALL MPI_SSEND(...)
```

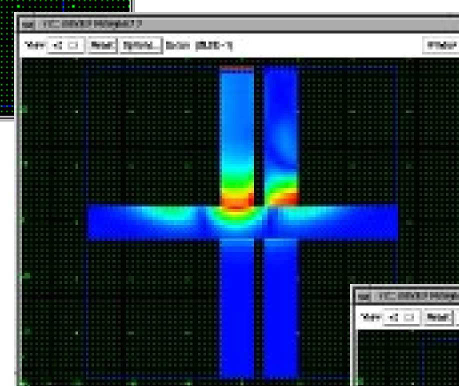
- Issue (blocking) wait calls (wait for receives to complete):

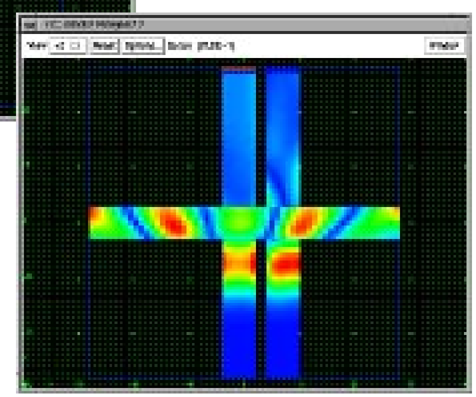
```
CALL MPI_WAIT(...)
```

Finite-Difference Time-Domain Application







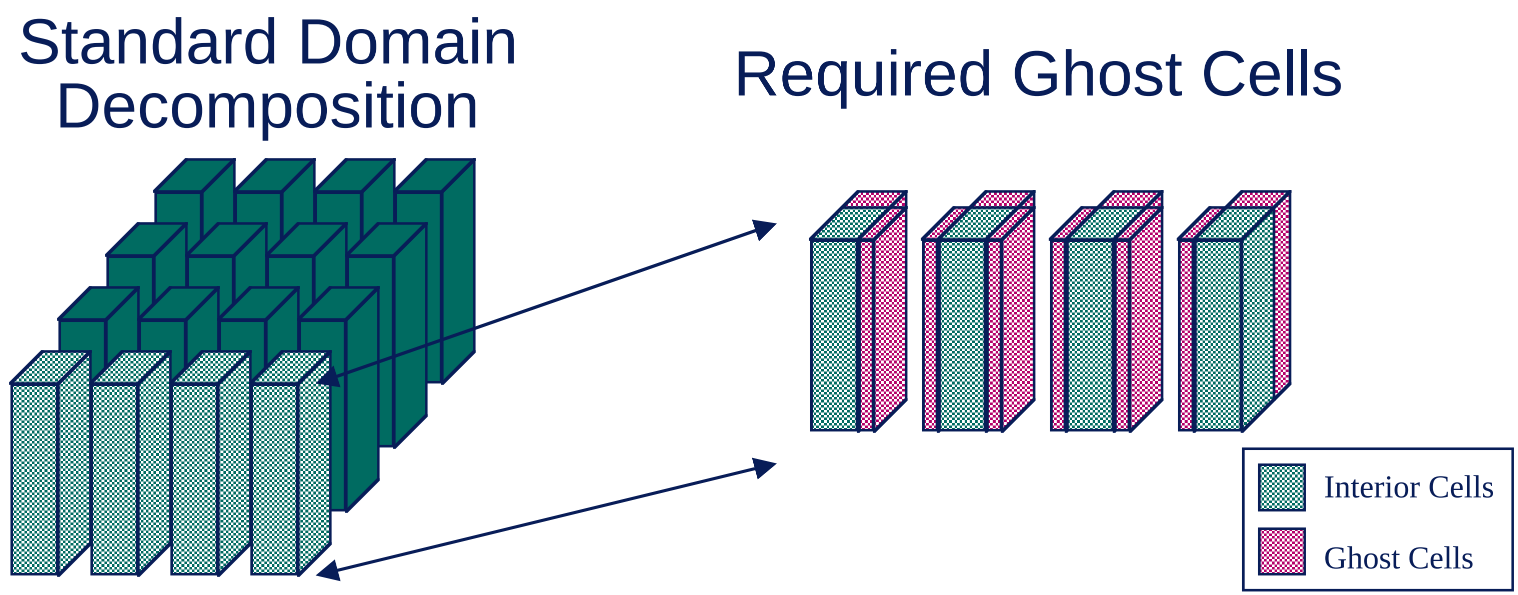


Images produced at U of Colorado's Comp. EM Lab. by Matt Larson using SGI's **LC** FDTD code

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 × 362 × 102 cells. Cell size = 1 mm^3.

FDTD Algorithm

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



FDTD Results

Number of Processors	Naegling	T3D	T3E-600
1	2.44 - 0.0	2.71 - 0.0	0.851 - 0.0
4	2.46 - 0.097	2.79 - 0.026	0.859 - 0.019
16	2.46 - 0.21	2.79 - 0.024	0.859 - 0.051
64	2.46 - 0.32	2.74 - 0.076	0.859 - 0.052

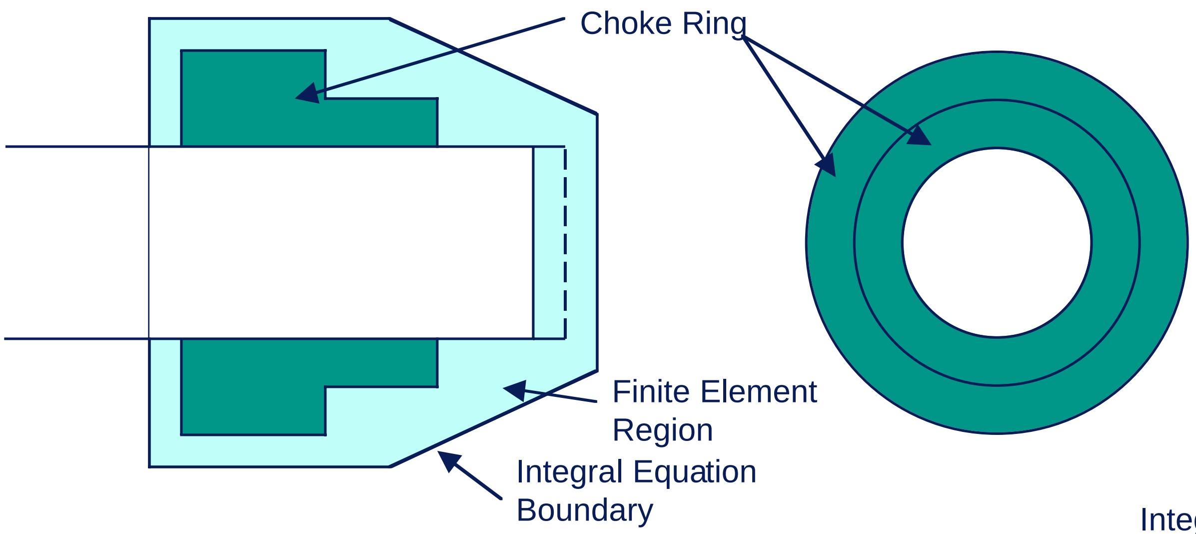
Time (wall clock seconds / time step),
scaled problem size (69 × 69 × 76 cells / processor),
times are: computation - communication

- Initial tests indicate 20% computational speed-up with 300 MHz Pentium II

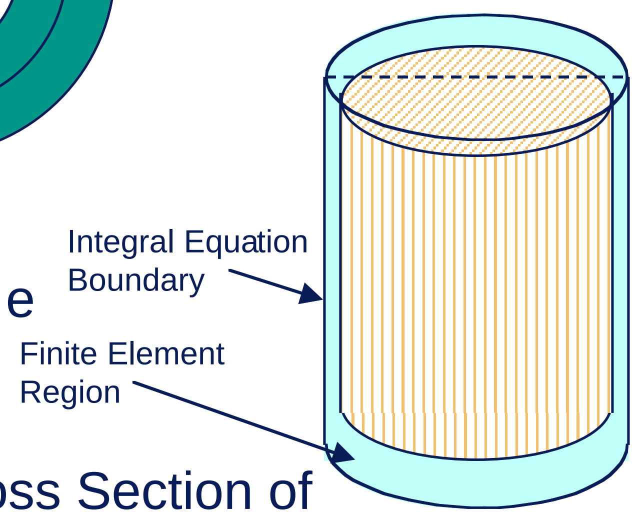
FDTD Conclusions

- On all numbers of processors, Beowulf-class computers perform similarly to T3D, and worse than T3E, as expected.
- A few large messages each time step take significantly longer on the Beowulf, but do not make an overall difference.

PHOEBUS Application (D. Katz, T. Cwik)



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

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

$$H = -K^{-1}CM$$

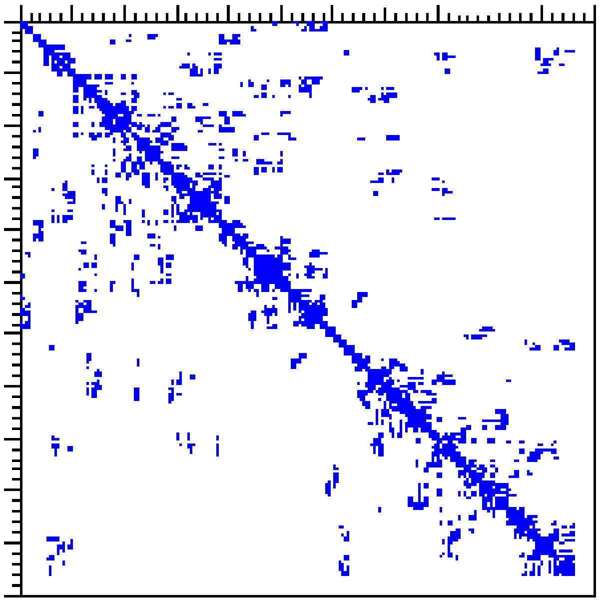
$$\begin{bmatrix} -C^\dagger K^{-1}C & Z_0 \\ Z_M & Z_J \end{bmatrix} \begin{bmatrix} M \\ J \end{bmatrix} = \begin{bmatrix} 0 \\ V \end{bmatrix}$$

- 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.

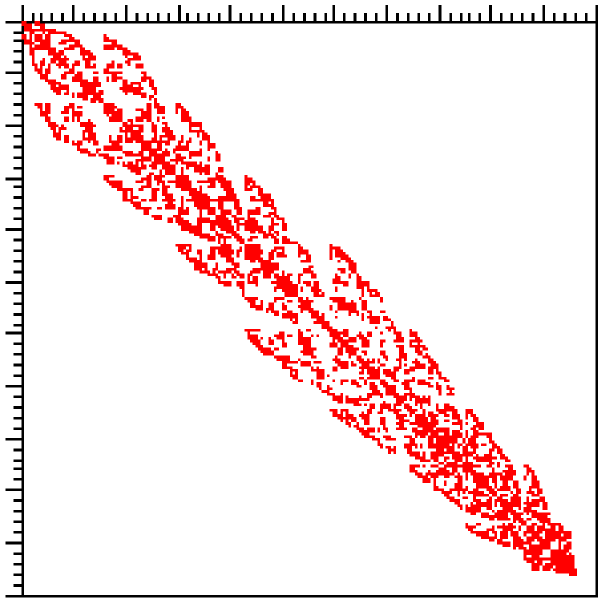
PHOEBUS Algorithm

- 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



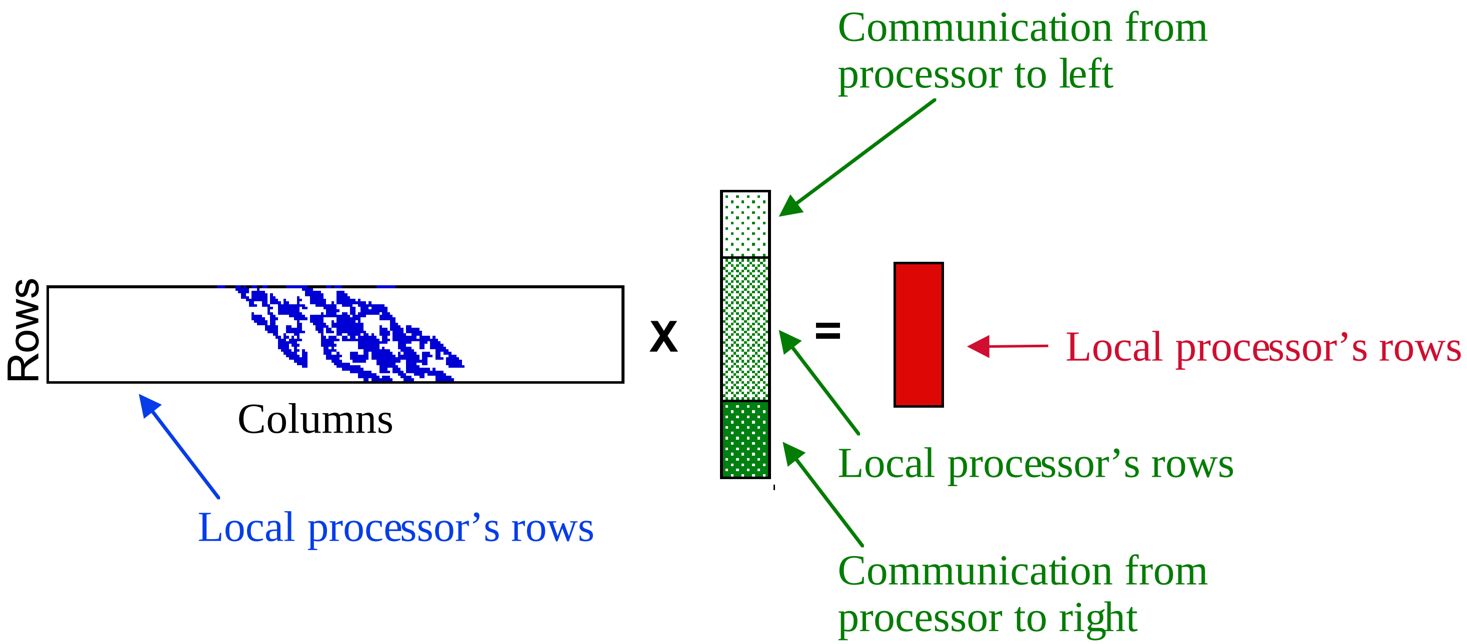
Original System



System after Reordering
for Minimum Bandwidth

Non-zero structure of matrices,
using SPARSPAK's GENRCM Reordering Routine

PHOEBUS
Matrix-Vector Multiply



PHOEBUS Solver Timing

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

Number of Processors	T3D (shmem)	T3D (MPI)	Naegling (MPI)
Matrix-Vector Multiply Computation	1290	1290	1502
Matrix-Vector Multiply Communication	114	272	1720
Other Work	407	415	1211
Total	1800	1980	4433

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

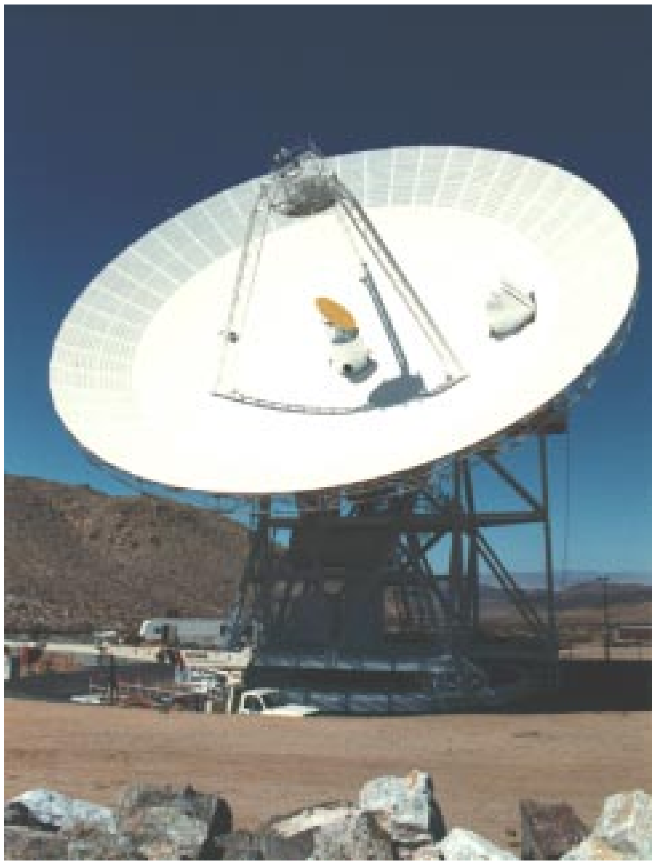
Number of Processors	T3D (shmem)	T3D (MPI)	Naegling (MPI)
Matrix-Vector Multiply Computation	868	919	1034
Matrix-Vector Multiply Communication	157	254	2059
Other Work	323	323	923
Total	1348	1496	4016

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

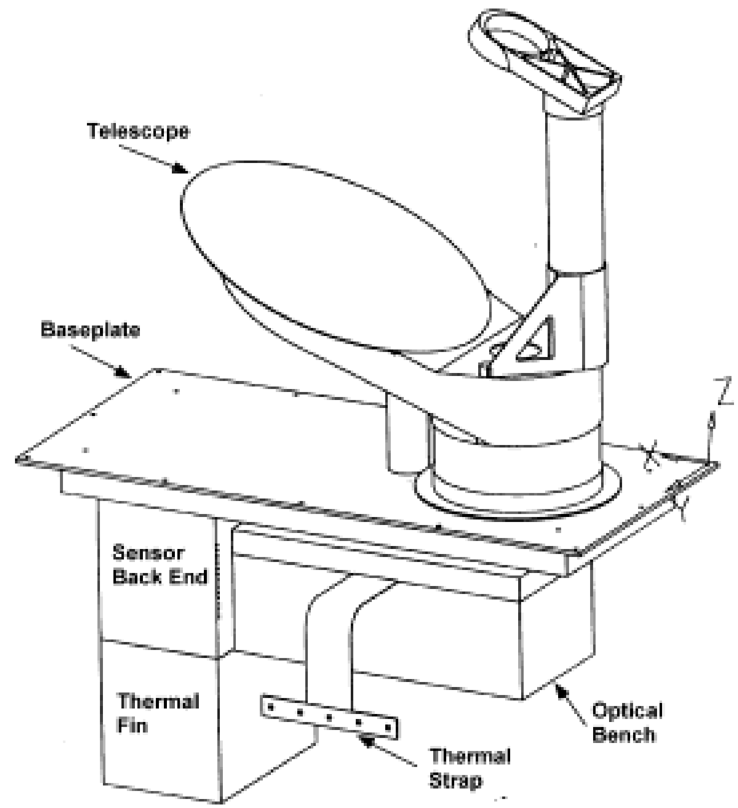
PHOEBUS Conclusions

- 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

Physical Optics Application (D. Katz, T. Cwik)

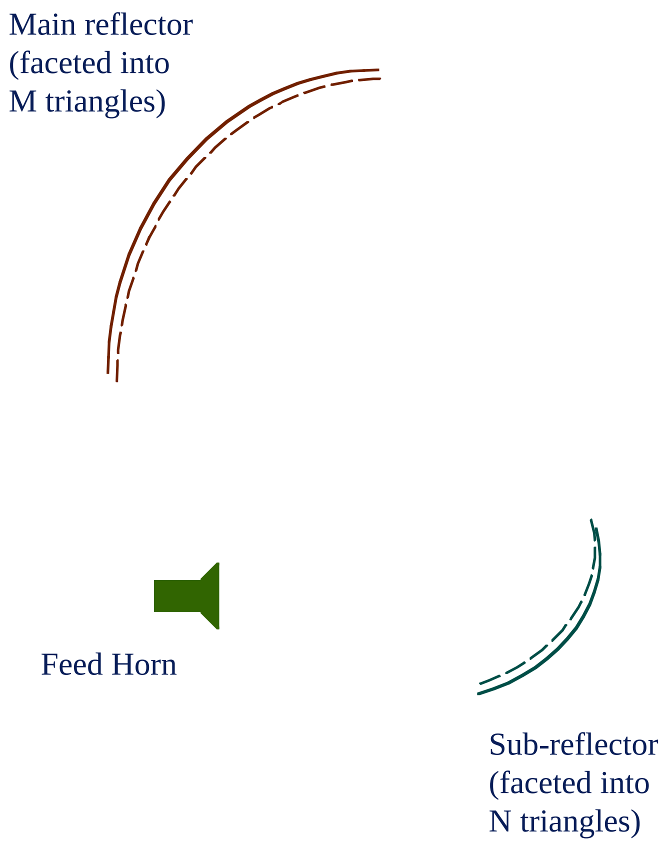


DSN antenna - 34 meter main



MIRO antenna - 30 cm main

Physical Optics Algorithm



1. Create mesh with N triangles on sub-reflector.
2. Compute N currents on sub-reflector due to feed horn (or read currents from file)
3. Create mesh with M triangles on main reflector
4. Compute M currents on main reflector due to currents on sub-reflector
5. Compute antenna pattern due to currents on main reflector (or write currents to file)

Parallelization of PO Algorithm

- Distribute (M) main reflector currents over all (P) processors
- Store all (N) sub-reflector currents redundantly on all (P) processors
- Creation of triangles is sequential, but computation of geometry information on triangles is parallel, so 1 and 3 are partially parallel
- Computation of currents (2, 4, and 5) is parallel, though communication is required in 2 (MPI_Allgatherv) and 5 (MPI_Reduce).
- Timing:
  - » Part I: Read input files, perform step 3
  - » Part II: Perform steps 1, 2, and 4
  - » Part III: Perform step 5 and write output files
- Algorithm:
  1. Create mesh with N triangles on sub-reflector.
  2. Compute N currents on sub-reflector due to feed horn (or read currents from file)
  3. Create mesh with M triangles on main reflector
  4. Compute M currents on main reflector due to currents on sub-reflector
  5. Compute antenna pattern due to currents on main reflector (or write currents to file)

Physical Optics Results (Two Beowulf Compilers)

Number of Processors	Part I	Part II	Part III	Total
1	0.0850	64.3	1.64	66.0
4	0.0515	16.2	0.431	16.7
16	0.0437	4.18	0.110	4.33

Time (minutes) on Hyglac, using gnu (`g77 -O2 -fno-automatic`)

Number of Processors	Part I	Part II	Part III	Total
1	0.0482	46.4	0.932	47.4
4	0.0303	11.6	0.237	11.9
16	0.0308	2.93	0.0652	3.03

Time (minutes) on Hyglac, using Absoft (`f77 -O -s`)

M = 40,000 N = 4,900

Physical Optics Results

Number of Processors	Naegling	T3D	T3E-600
4	95.5	102	35.1
16	24.8	26.4	8.84
64	7.02	7.57	2.30

Time (minutes), N=160,000, M=10,000

- Initial tests indicate 40% computational speed-up with 300 MHz Pentium II

PO Conclusions

- Performance of codes with very small amounts of communication is determined by CPU speed.
- Naegling results are between T3D and T3E.
- This is close to the best that can be attained with Beowulf-class computers.

Incompressible Fluid Flow Solver (John Lou)

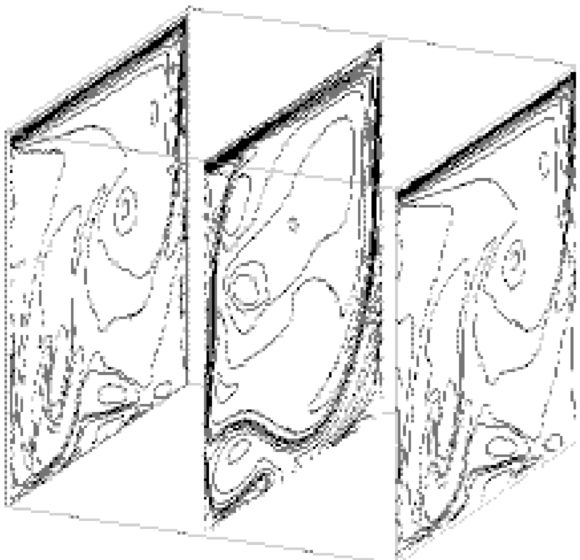


Image: Vorticity projections in
 streamwise-vertical planes
Flow Problem: 3-D driven cavity flow, Re=2,500
Grid Size: 256 x 256 x 256
Algorithm: Second order projection method with
 a multigrid full V-cycle kernel
Computer: Cray T3D with 256 processors

Incompressible Fluid Flow Solver (John Lou)

Grid Size	Number of Processors	Beowulf Time	T3D Time	T3E Time
128×128	1	6.4 - 6.4 - 0.0	13.8 - 13.8 - 0.0	5.8 - 5.8 - 0.0
256×256	4	22.2 - 7.0 - 15.2	19.1 - 14.7 - 4.4	7.8 – 5.9 – 1.9
512×512	16	36.6 - 7.3 - 29.3	22.7 - 15.4 - 7.3	9.6 – 6.0 – 3.6

Times are run times in seconds (total - computation - communication)

Grid Size	Number of Processors	Beowulf Time	T3D Time	T3E Time
128×128	64	21.2	5.0	2.1
512×512	64	52.7	11.5	5.2
2048×2048	64	230	75.0	31.0

Times are total run times in seconds

- Initial tests indicate 40% computational speed-up with 300 MHz Pentium II

Incompressible Fluid Flow Solver (John Lou)

- As the number of processors increases, Beowulf performance drops, compared with T3D and T3E.
- For a fixed number of processors, Beowulf performance increases with local problem size
- Beowulf memory would need to grow as the number of processors increases to get scalable performance, relative to T3D/E.

General Conclusions

- Key factor in predicting code performance: amount of communication
- Beowulf has a place at JPL/Caltech
  - » Each machine should have:
    - – Small numbers of processors
    - – Limited number of codes/users
- Not a replacement for institutional supercomputers