Applications and Experiences on Hyglac, the JPL Beowulf

Daniel S. Katz

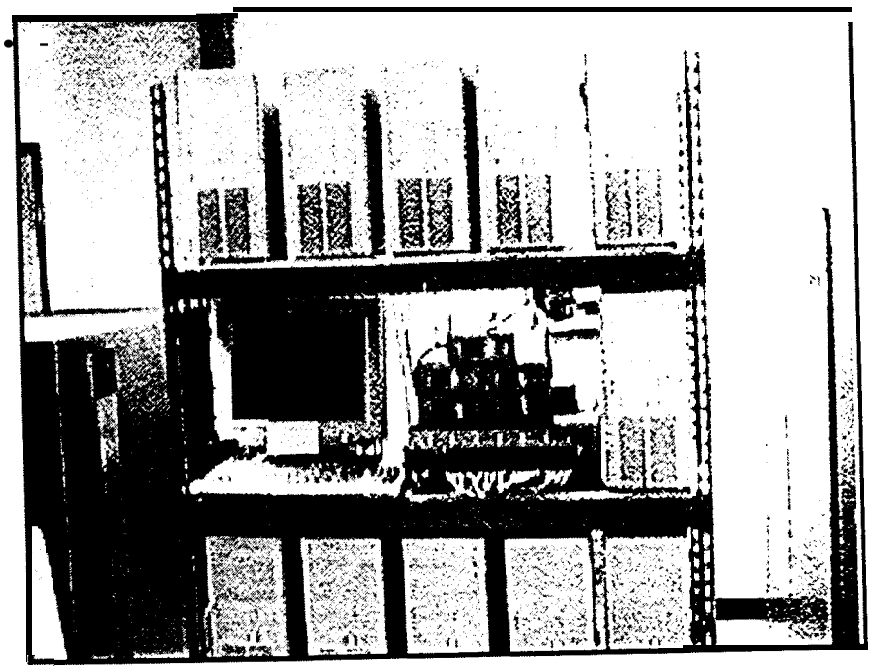
JPL

High Performance Computing Group

Imaging and Spectrometry Systems Technology Section

Beowulf System at JPL (Hyglac)

- 16 PentiumPro 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 performance: 3.2 GFlop/s.
- Achieved sustained performance: 1.26 GFlop/s.



Hyglac Cost

- Hardware cost: $54,200 (9/18/96)
  - 16 (CPU, disk, memory)
  - 1 (16-way crossbar, monitor, keyboard, mouse)
- Software cost: $0 (+ maintenance)
  - Public Domain OS, compilers, tools, libraries.
- 256 PE T3D: $8 million (1/94)
  - including all software and maintenance for 3 years.
- 16 PE T3E: $1 million

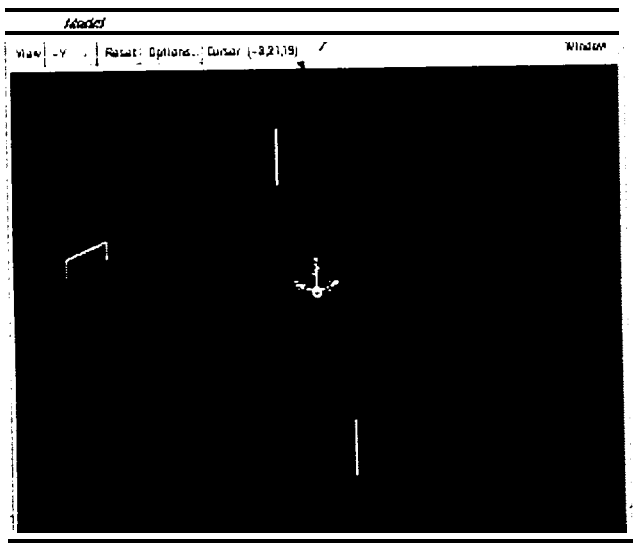
  3 to 4 times more powerful then T3D.

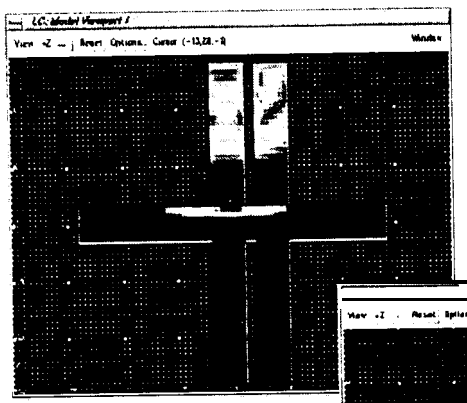
Hyglac Performance (vs. T3D)

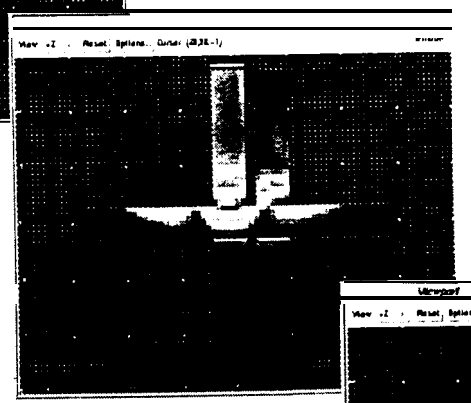
	Hyglac (MPI)	T3D (MPI)	T3D (shmem)
CPU Speed (MHz)	200	150	150
Peak Rate (MFlop/s)	200	150	150
L1, L2 Cache Size (Kbytes)	8i+8d, 256	8i+8d, 0	8i+8d, 0
Memory Bandwidth (Gbit/s)	0.78	1.4	1.4
Communication Latency (μs)	156	55	1.8
Communication Bandwidth (Mbit/s)	80	225	280-970

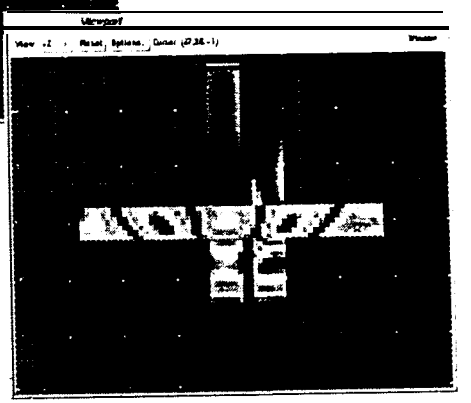
- Next, examine performance of various applications...

Application:







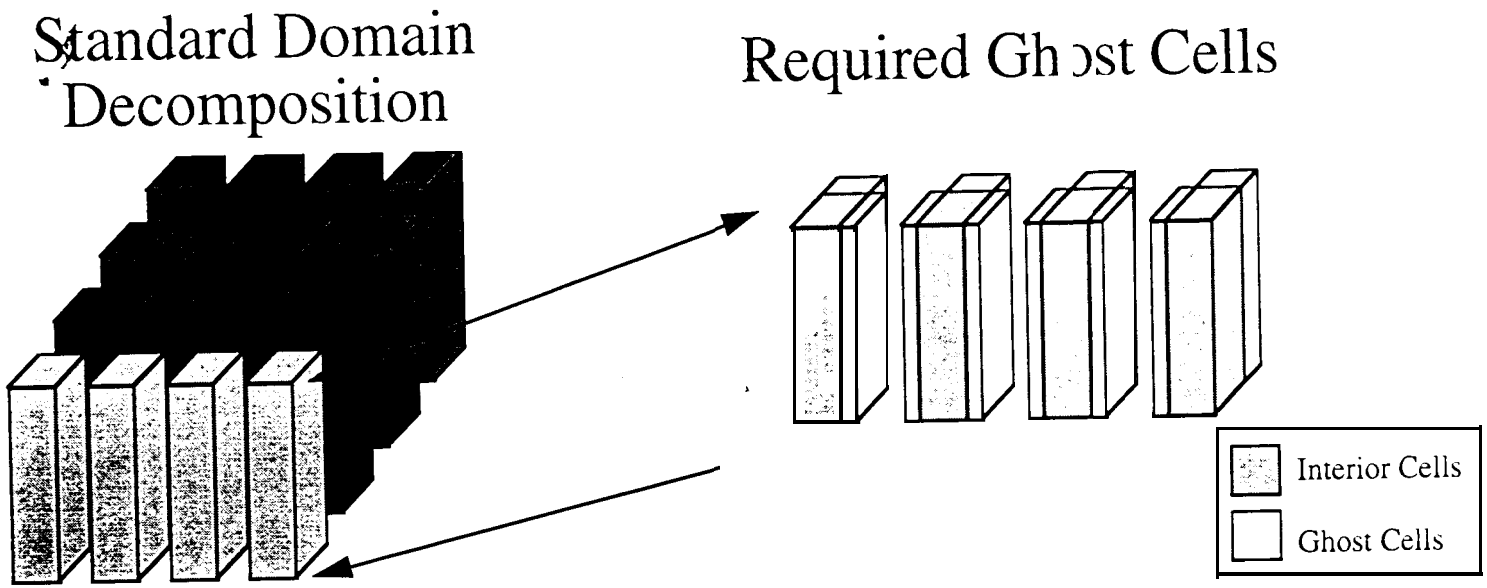


Images produced at U of Colorado's Comp. EM Lab. by Matt Larson using SGI's LC FDTD code

Time steps of a gaussian pulse, travelling on a microstrip, showing coupling to a neighboring strip, and cross talk 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.

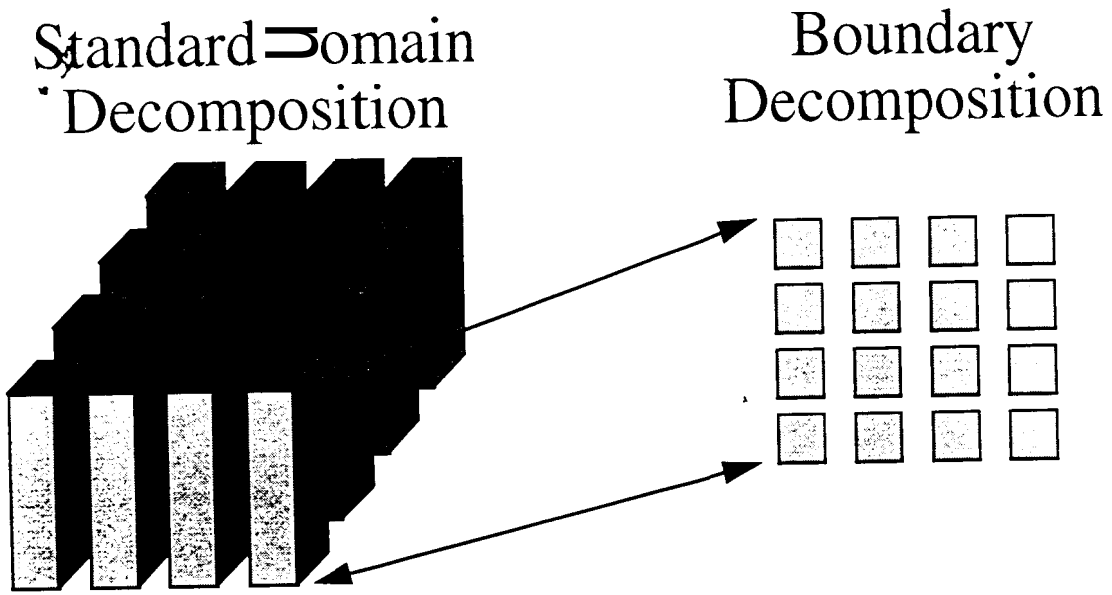
Electromagnetic Finite-Difference Time-Domain code

FDTD Interior Communication



One plane of ghost cells must be communicated to each neighboring processor each time step.

FDTD Boundary Communication



Extra work required along faces (boundary conditions, wave source, far-field data locus).

→

Different decomposition required for good load balance.

Data at 4 faces must be redistributed twice each time step!!

FDTD Timing

	T3D (shmem)	T3D (MPI)	Hyglac (MPI, Good Load Balance)	Hyglac (MPI, Poor Load Balance)
Interior Computation	1.8 (1.3*)	1.8 (1.3*)	1.1	1.1
Interior Communication	0.007	0.08	3.8	3.8
Boundary Computation	0.15	0.15	0.14	0.42
Boundary Communication	0.04	1.5	51	0.0
Total	2.0 (1.5*)	3.5 (3.0*)	55.1	5.5

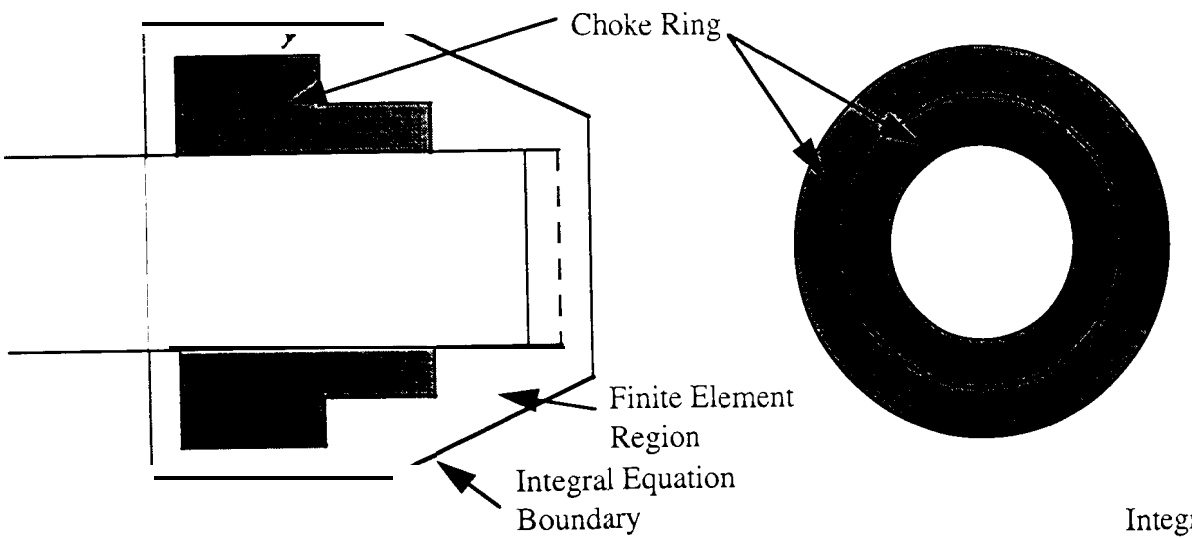
(using assembler kernel)

All timing data is in CPU seconds/simulated time step, for a global grid size of 282 × 362 × 102, distributed on 16 processors.

FDTD Timing, cont.

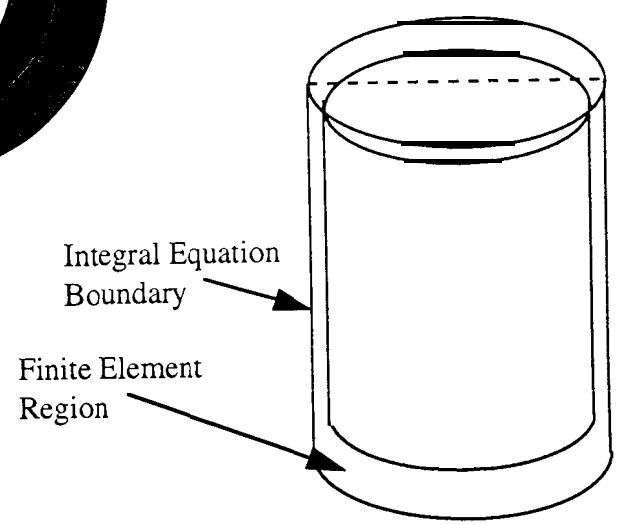
- Computation:
  - Hyglac CPU is 35-65% faster than T3D CPU.
- Communication (including barriers):
  - T3D: MPI is 4 to 9 times slower than shmem.
  - Hyglac MPI is 30-50 times slower than T3D MPI.
- Good (or even acceptable) performance may require rewriting/modifying code.

Application:



Radiation Pattern from JPL Circular Waveguide

(from C. Zuffada, *et. al.*, IEEE AP-S paper 1/97)



Radar Cross Section of a dielectric cylinder

PHOEBUS: Electromagnetic Coupled Finite Element - Integral Equation Code

PHOEBUS Coupled Formulation

For three unknowns,

$\overline{H},\ \overline{J},\ \overline{M}$

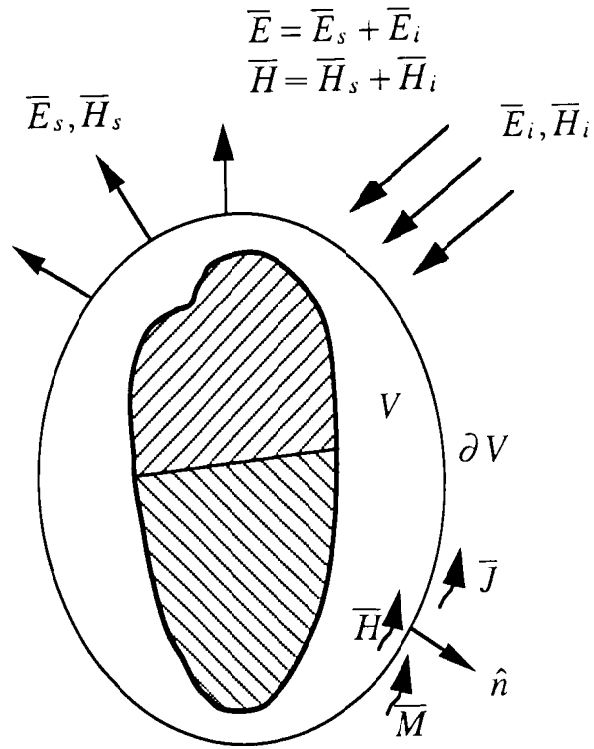
The following three equations must be solved:

$$\frac{1}{-j\omega\varepsilon_0}\int_V\left[(\nabla\times\overline{T})\ \frac{1}{\varepsilon_r}(\nabla\times\overline{H})-k_0^2\overline{T}\ \mu_r\overline{H}\right]dV+\int_{\partial V}\overline{T}\ \overline{M}\,dS=0$$

(finite element equation)

$$\int_{\partial V}\hat{n}\times\overline{U}\cdot\left[\hat{n}\times\overline{H}-\overline{J}\right]dS=0$$ (essential boundary condition)

$$Z_e\left[\overline{M}\right]+Z_h\left[\overline{J}\right]=V_i$$ (combined field integral equation)



PHOEBUS Coupled Equations

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

- 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 Two Step Method

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

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

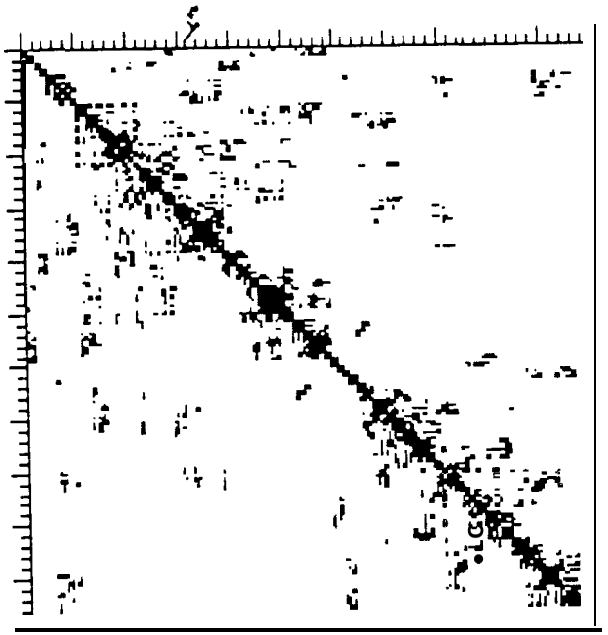
$$\begin{bmatrix} -C^{\dagger}K^{-1}C & Z_0 \\ Z_M & Z_J \end{bmatrix} \begin{bmatrix} M \\ J \end{bmatrix} = \begin{bmatrix} 0 \\ V \end{bmatrix}$$

- 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.
- If required, save $K^{-1}C$ to solve for H.

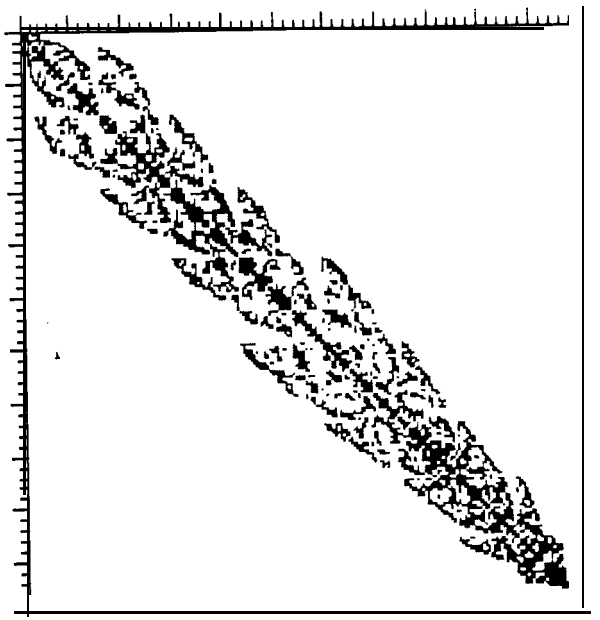
PHOEBUS Decomposition

- Matrix Decomposition
  - Assemble complete matrix.
  - Reorder to equalize row bandwidth.
    - » Gibbs-Poole-Stockmeyer
    - » SPARSPAK's GENRCM
  - Partition matrix in slabs or blocks.
  - Each processor receives slab of matrix elements.
  - Solve matrix equation.

PHOEBUS Matrix Reordering



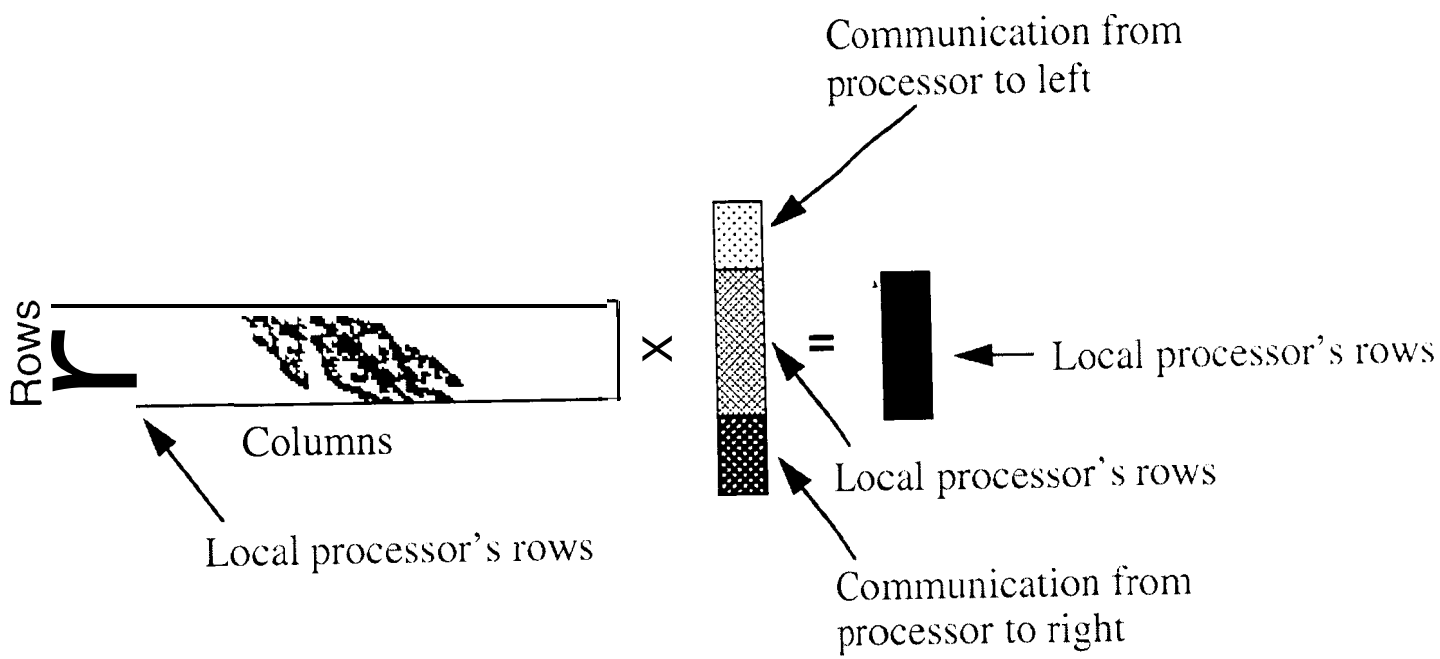
Original System



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 = cm, height = 10 cm, permittivity = 4.0, at 5.0 GHz

	T3D (shmem)	T3D (MPI)	Hyglac (MPI)
Matrix-Vector Multiply Computation	1290	1290	590
Matrix-Vector Multiply Communication	114	272	3260
Other Work	407	415	1360
Total	1800	1980	5220

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

PHOEBUS Solver Timing, cont.

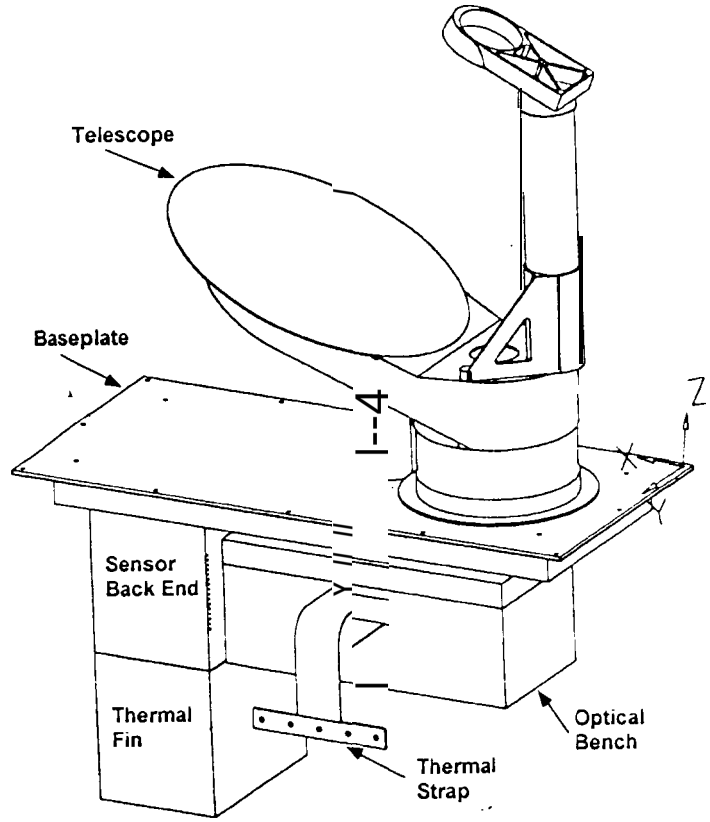
- Computation:
  - Hyglac CPU works 55% faster than T3D CPU.
  - For sparse arithmetic, large secondary cache provides substantial performance gain.
- Communication (including barriers):
  - T3D MPI is 2.5 times slower than T3D shmem.
  - Hyglac MPI is 10 times slower than T3D MPI.
  - The code was rewritten to use manual packing and unpacking, to combine 10 small messages into 1 medium-large message.

PHOEBUS Solver Timing, cont.

- Other work:
  - Mostly BLAS1-type operations (dot, norm, scale, vector sum, copy)
    - » Heavily dependent on memory bandwidth.
  - Also, some global sums
    - » Over all PEs.
    - » Length: number of RHS/block complex words.
  - 3.5 times slower than T3D.

Application

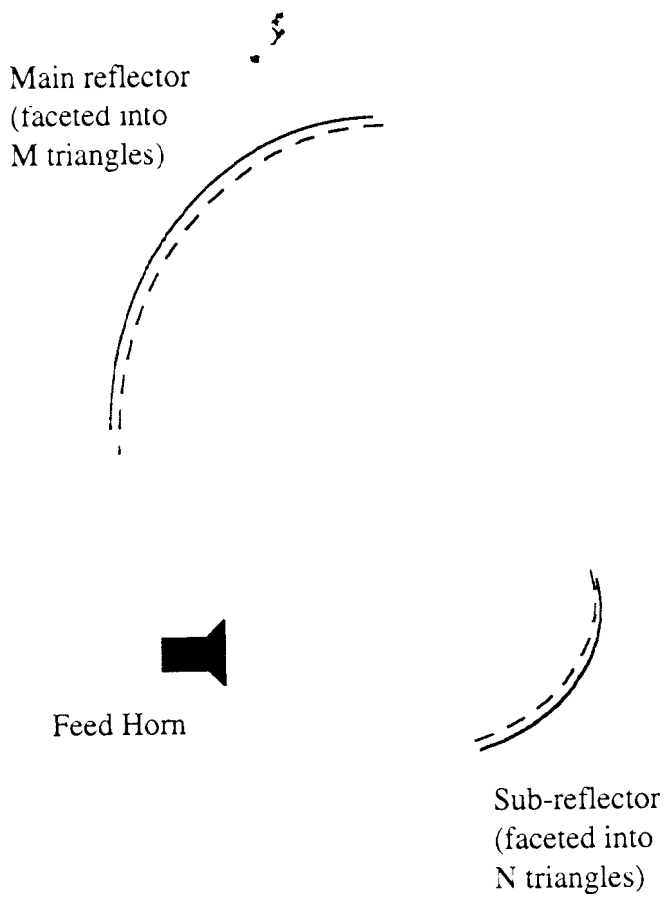
DSN antenna - 34 meter main



MIRO antenna - 30 cm main

Physical Optics (Tom Cwik, Dan Katz)

Physical Optics (Tom Cwik, D. Katz)



- Algorithm:
  - 1. Compute N currents on sub-reflector due to feed horn.
  - 2. Compute M currents on main reflector due to currents on sub-reflector.
  - 3. Compute antenna pattern due to currents on main reflector.
- Parallel implementation:
  - (M) Main reflector currents are distributed over all (P) processors.
  - Step 1 is performed redundantly by all processors.
  - Step 2 is performed for all M/P currents on each processor.
  - Step 3 is performed partially by each processor, followed by a global sum to calculate the final antenna pattern.

Physical Optics (Tom Cwik, D. Katz)

Numbe of Processors	Beowulf	T3D
1	4 0	572
4	106	150
16	29.3	46.9

Timing results (seconds) for PO code for 40,000 triangle main reflector, 400 triangle sub-reflector

Number of Processors	Beowulf	T3D
1	66.0	89.9
4	16.3	22.7
16	4.33	6.05

Timing results (minutes) for PO code for 40,000 triangle main reflector, 4,900 triangle sub-reflector

Physical Optics (Tom Cwik, D. Katz)

- PO code has almost zero communication
  - The Beowulf version is clearly faster.
- The difference in code performance is mostly the difference in CPU speeds.
- Amdahl's law affects scaling...
  - The sequential parts are slower on T3D (I/O)...
    - » T3D's scaling is worse than Beowulf's.
  - To reduce sequential part of code, more communication would be required...
    - » On a larger number of processors, the T3D would improve faster than the Beowulf.

Application:

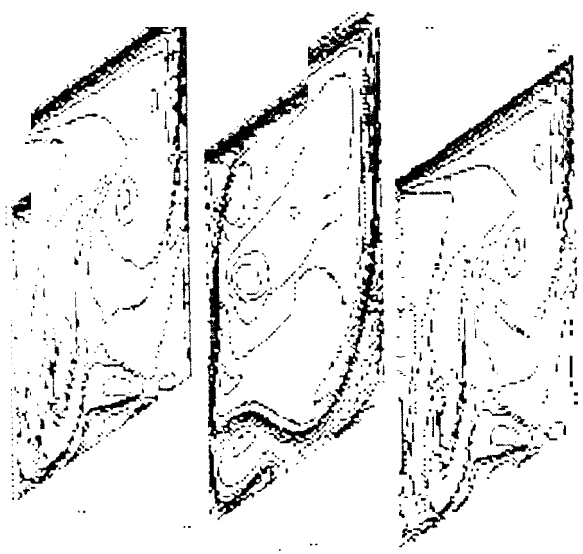


Image: Vorticity projections in streamwise-vertical planes
Flow Problem: 3-D driven cavity flow, Re=2,500
Grid Size: 256 x 256 x 256
Algorithm: Second order projection method with a multigrid full V-cycle kernel
Computer: Cray T3D with 256 processors

Incompressible Fluid Flow Solver (John Lou)

Incompressible Fluid Flow (John Lou)

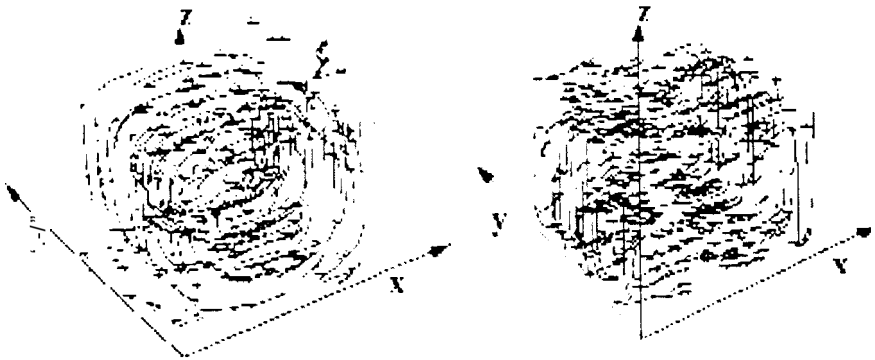
Grid Size	Number of Processors	Beowulf Run Time (seconds): Total - computation - communication	T3D Run Time (seconds): Total - computation - communication
128×128	1	6.4 - 6.4 - 0.0	13.8 - 13.8 - 0.0
256×256	4	22.2 - 7.0 - 15.2	19.1 - 14.7 - 4.4
512×512	16	36.6 - 7.3 - 29.3	22.7 - 15.4 - 7.3

Grid Size	Number of Processors	Beowulf Run Time (seconds)	T3D Run Time (seconds)
64×64	16	12.1	3.6
256×256	16	22.7	9.6
1024×1024	16	67.5	67.2

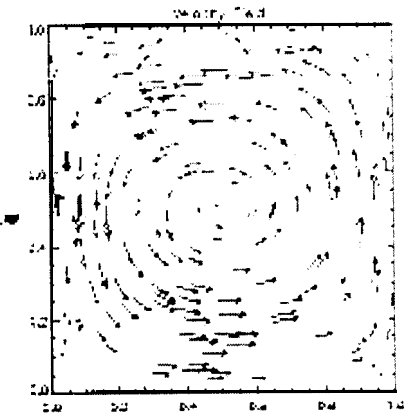
Incompressible Fluid Flow (John Lou)

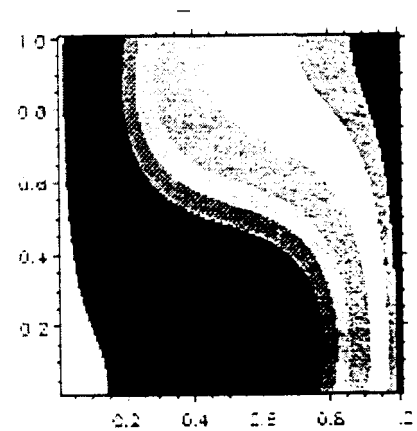
- For large local grid size, where communication/computation ratio is small, Beowulf performs similarly to T3D.
- For small local grid size, where communication/computation ratio is larger, T3D performs better than Beowulf.
- Since the T3D beats the Beowulf in communication and loses in computation, these results are reasonable.

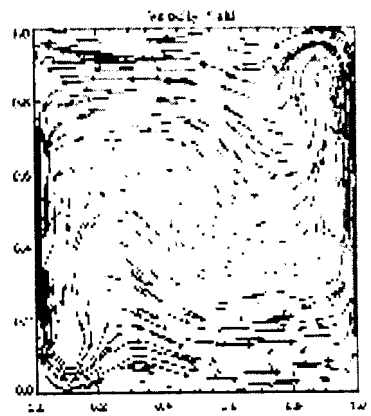
Application:

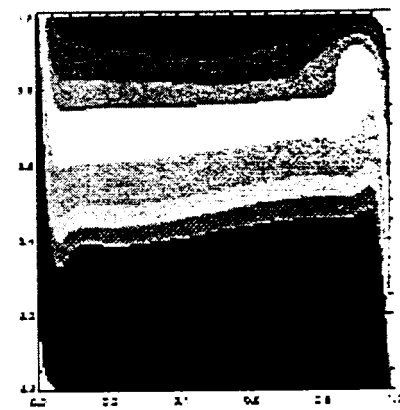


Velocity in 3-D domain for Rayleigh number: 14460 (left), 10^7 (right)









Velocity (left), temperature (right) on y=0.5 for Rayleigh number 14460 (top), 10^7 (bottom)

Thermal Convection Solver (Ping Wang)

Thermal Convection (Ping Wang)

	Paragon	Beowulf	T3D
Computation	110	35	27
Communication	49	60	19
Other Work	22	5	12
Total	181	100	58

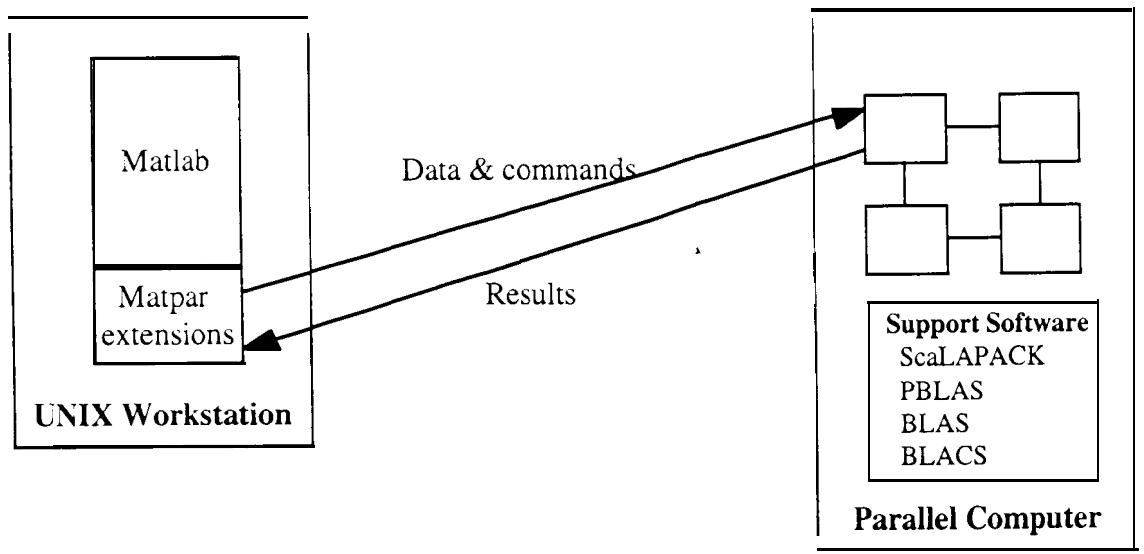
Grid Size/ System	32 x 32 x 32	64 x 64 x 64	128 x 128 x 128	256 x 256 x 256
Paragon	60	181		
Beowulf	41	100	177	
T3D	18	58	117	
SP2	12	38	95	280
T3E	4	14	32	155

Wallclock times (seconds) using 16 processors, Rayleigh number = 10^6

Thermal Convection (Ping Wang)

- As problem sizes grow, the Beowulf speed approaches the T3D speed, and always exceeds the Paragon speed.
- Computation rates for Beowulf and T3D are similar.
- Communication is much faster on T3D.
- Other work includes I/O and grid set-up.

Application:



Matpar
Parallel Extensions for Matlib (Paul Springer)

Matpar (Paul Springer)

- Port in progress. Current issues include:
  - Communication between workstation and CPUs
    - » individual Beowulf CPUs don't have IP addresses visible outside the cluster
    - » may use "IP masquerading" to allow this
    - » ideal solution would be to extend PVM, as MPP vendors do
  - Communication performance:
    - » PVM required due to job spawning
    - » PVM performance is worse than MPI performance
  - Computation performance
    - » Optimized math libraries (BLAS) needed
    - » currently on order from NAG

Scaling to Larger Systems

- FDTD and FEM codes should scale almost perfectly to larger systems. The flow and convection codes should scale fairly well. PO code will probably scale poorly, due to sequential components. Matpar scaling will probably be good, but initial and final communications may cause some problems.

Conclusions

- User experience has been mostly good:
  - Lack of both support and documentation can be frustrating.
  - Easy to use, once you know how use it.
- Most important factor in predicting a code's performance: Communication/computation ratio.
- Second factor is message size.