High-Performance Computational Electromagnetic Modeling Using Low-Cost Parallel Computers

July 14, 1997

Daniel S. Katz

(Daniel.S.Katz@jpl.nasa.gov)

Jet Propulsion Laboratory

California Institute of Technology

Pasadena, CA 91109

Beowulf System (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

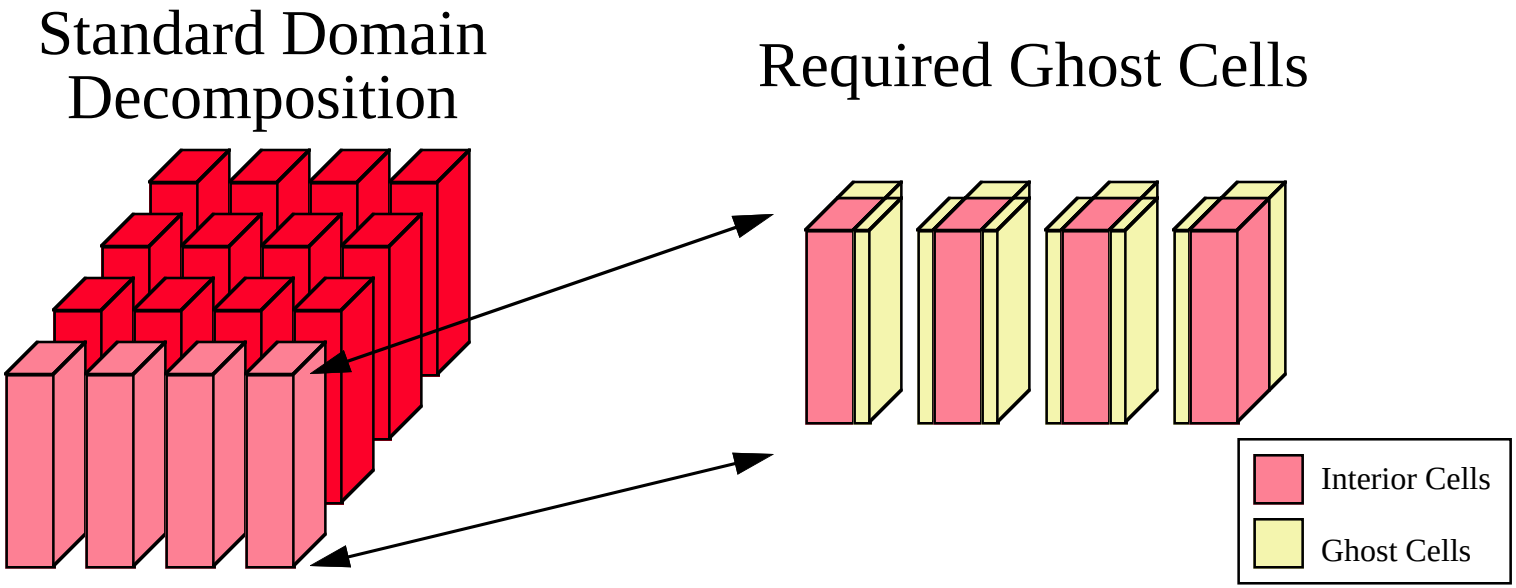
- Hardware cost: $54,200 (9/18/96)
  - 16 (CPU, disk, memory)
  - 1 (16-way crossbar, monitor, keyboard, mouse)
- Software cost: $0 (+ maintenance)
  - Public Domain OS, compilers, tools, libraries.
- 256 PE T3D: $8 million (1/94)
  - including all software and maintenance for 3 years.
- 16 PE T3E: $1 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)	150	35	1.8
Communication Bandwidth (Mbit/s)	66	225	280-970

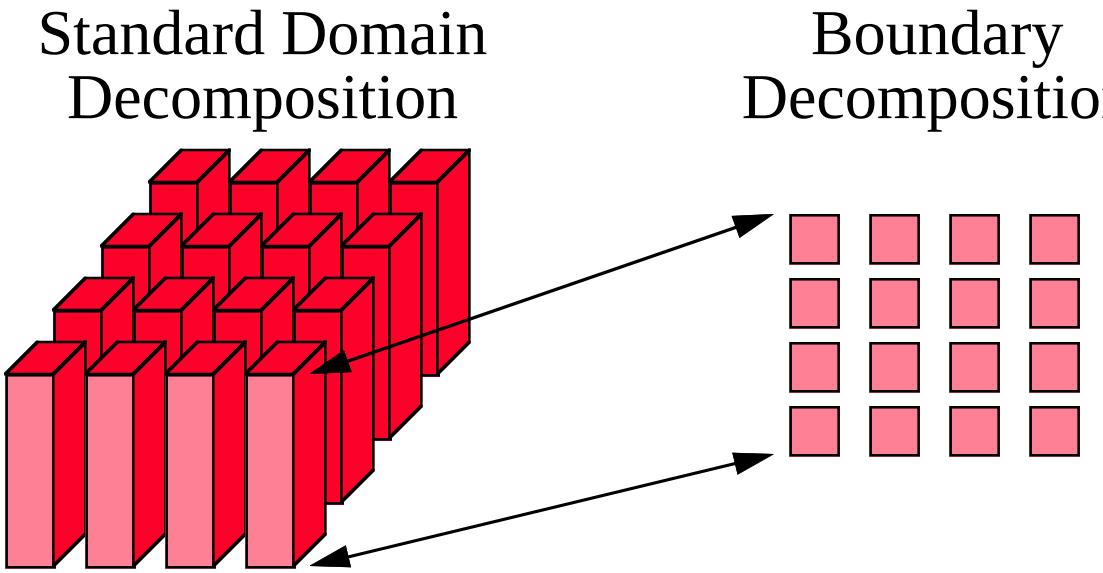
- Next, examine performance of EMCC FDTD and PHOEBUS (FE) codes...

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	0.19	0.19	0.14	0.42
Boundary Communication	0.04	1.5	50.1	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 $282 \times 362 \times 102$, distributed on 16 processors.

FDTD Timing, cont.

- Computation:
  - Hyglac CPU is 35-65% faster than T3D CPU.
- Communication:
  - 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.

PHOEBUS Coupled Formulation

For three unknowns,

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

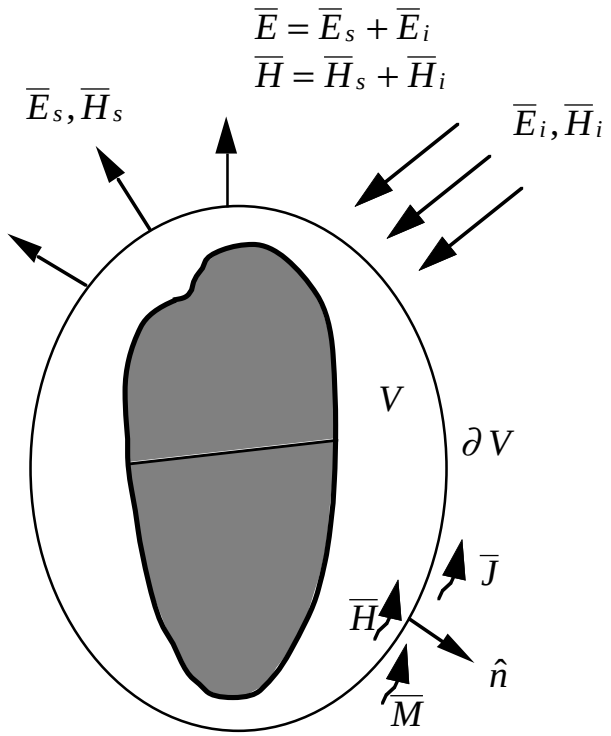
The following three equations must be solved :

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

(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

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

- 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 Two Step Method

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

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

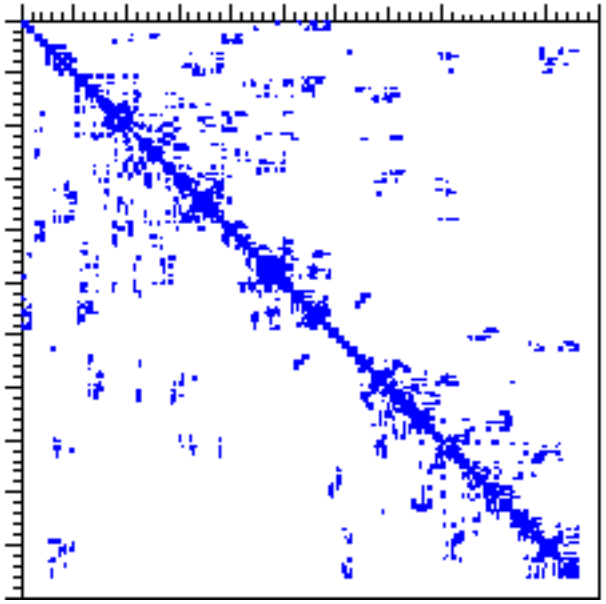
$$\begin{bmatrix} -C^{\dagger}K^{-1}C & Z_0 \\ Z_M & Z_J \end{bmatrix} \begin{bmatrix} M \\ J \end{bmatrix} = \begin{bmatrix} 0 \\ V \end{bmatrix}$$

- 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.

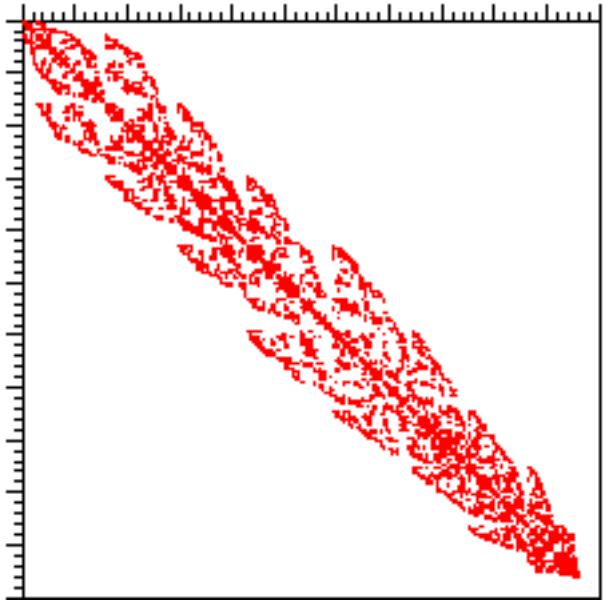
PHOEBUS Decomposition

- 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



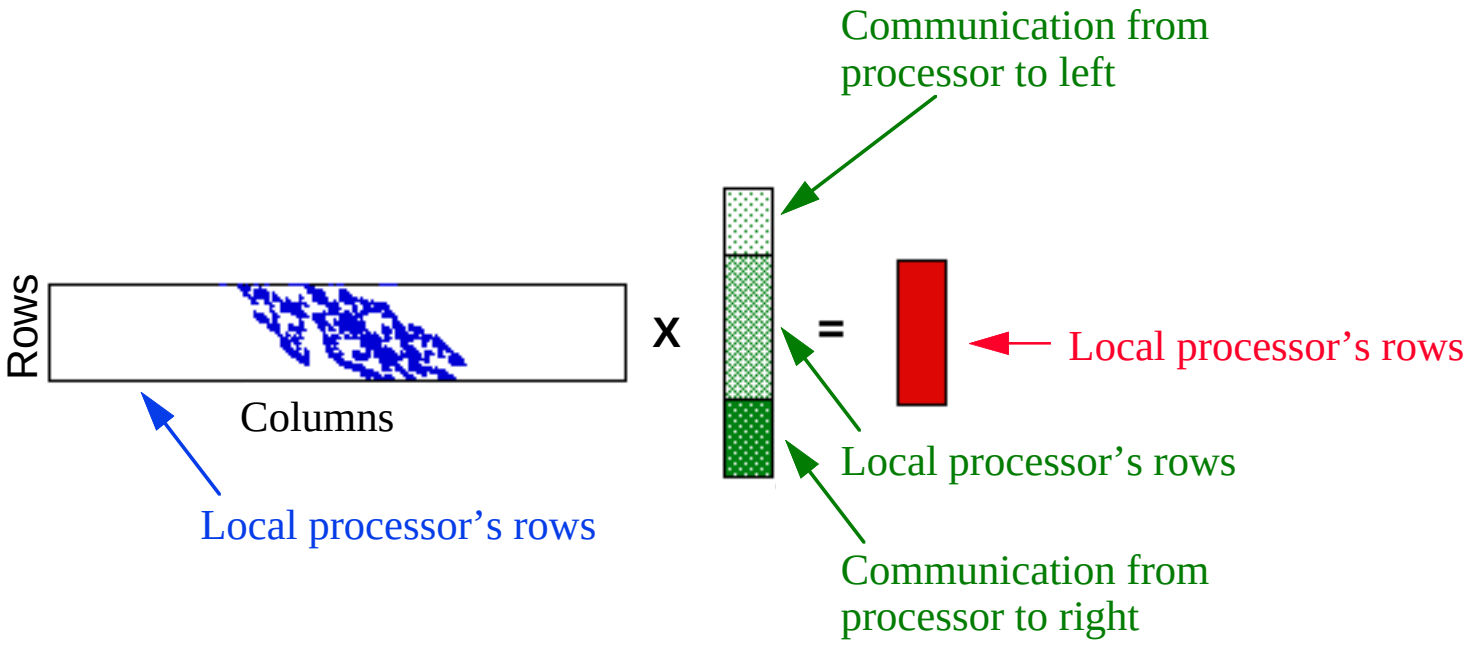
Original System



System after Reordering
for Minimum Bandwidth

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

	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:
  - 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.

Conclusions

- If message sizes are not too small, adequate communication performance is possible.
- Computation rates are good, provided there is adequate data reuse.
- Low-cost parallel computers can provide better performance/$ than traditional parallel supercomputers, for limited problem sizes, and with some code rewriting.

Conclusions, cont.

- Both EMCC FDTD code and PHOEBUS QMR solver should scale to larger numbers of processors easily and well, since most communication is to neighbors.
- User experience has been mostly good:
  - Lack of both support and documentation can be frustrating.
  - Easy to use, once you know how use it.