The Parallel Solution of Matrix Equations Resulting from Unstructured Finite-Element Problems

Daniel S. Katz

Cray Research, a Silicon Graphics Company

Tom Cwik

Jet Propulsion Laboratory, California Institute of Technology

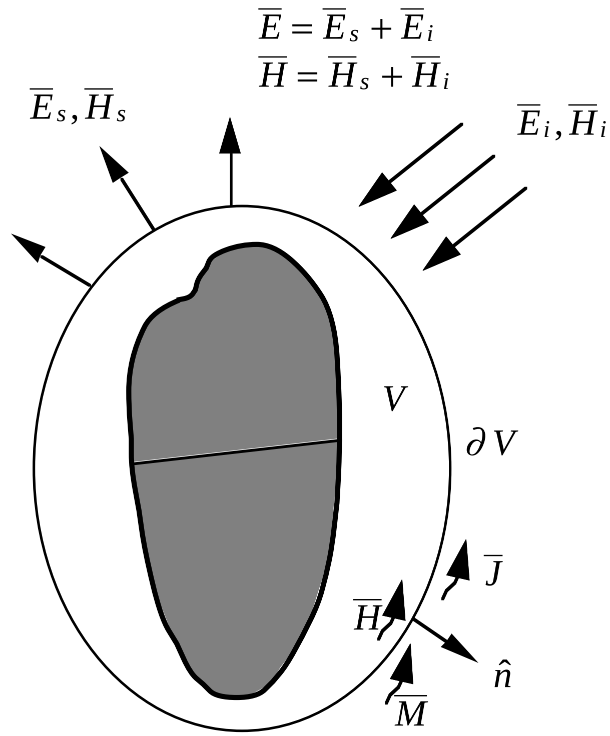
July 25, 1996

PHOEBUS References

- T. Cwik, C. Zuffada, and V. Jamnejad, "Modeling 3-Dimensional Scatterers Using a Coupled Finite-Element - Integral-Equation Technique," *IEEE Trans. Ant. Prop.*, v. 44, pp. 453-459, April 1996.

- T. Cwik, D. S. Katz, C. Zuffada, and V. Jamnejad, "The Application of Distributed Memory Computers to the FInite Element Modeling of Electromagnetic Scattering and Radiation," submitted to *Intl. J. Num. Meth. Eng.*

Coupled Formulation



For three unknowns,

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

The following three equations must be solved:

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

(finite element equation)

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

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

CRAY RESEARCH, INC.

DSK 7/96

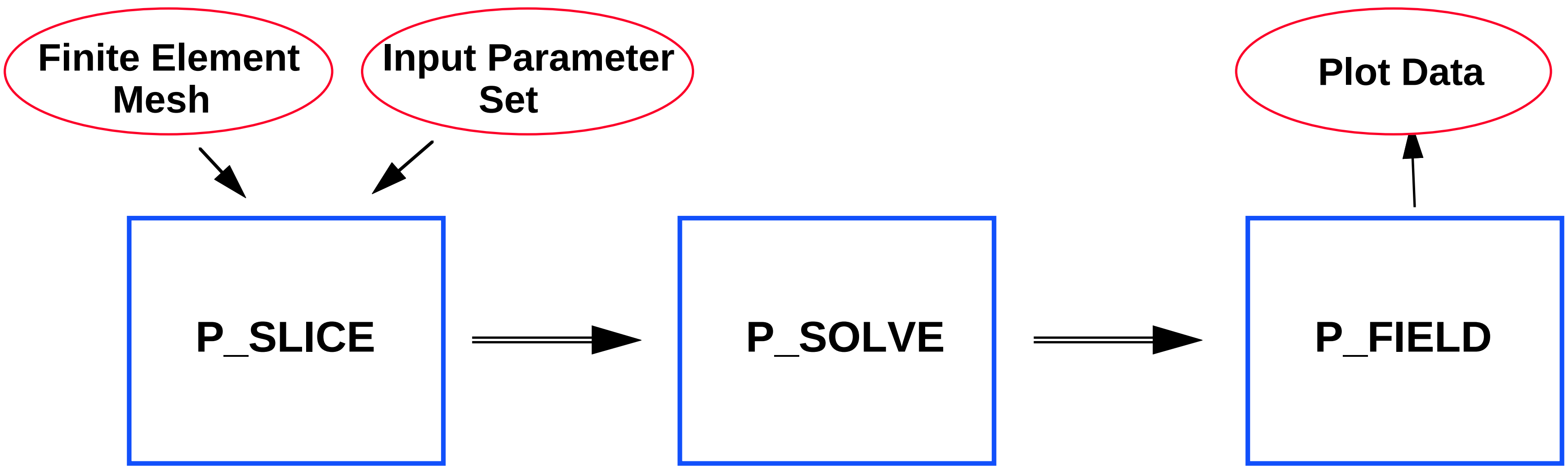
Coupled Equations

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

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 equations**

Parallel PHOEBUS Three Stage Simulation



Read Mesh Data
Generate Finite Element Matrix
Generate Coupling Matrix
Slice Matrix Data
Write Data to Disc

(Uses CRAFT Compiler)

Read Sliced Data
Compute Intermediate Solution
Write Data to Disc

Read Solve Data
Read Mesh Data
Generate Final Solution
Compute RCS
Compute Near and Far Fields

(Flexible for Additional Output)

Two step method

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

$$H = -K^{-1} C M$$

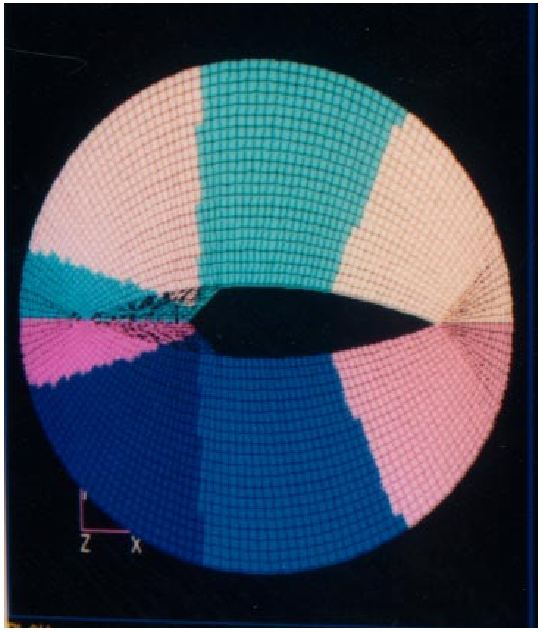
$$\begin{bmatrix} -C^\dagger K^{-1} C & Z_0 \\ Z_M & Z_J \end{bmatrix} \begin{bmatrix} M \\ J \end{bmatrix} = \begin{bmatrix} 0 \\ V \end{bmatrix}$$

- **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 (P_SOLVE)

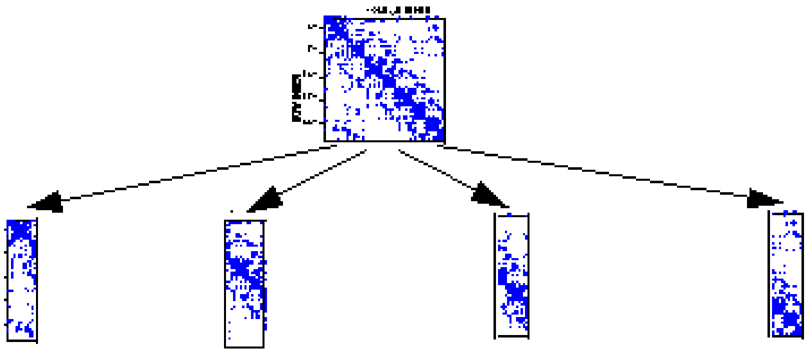
- P_SOLVE performs the first step in solving the matrix equation
- Reads in data from P_SLICE stored in individual files
- Uses Parallel Quasi-Minimum Residual Iterative Algorithm (Freund 1992)
  - Written to be portable to other platforms
  - Test systems from target meshes with up to 579,993 equations solved
- Writes out Z_k, for use in P_FIELD
- Issues in parallel iterative solution
  - Data load balance
  - Communication overhead (equalization over processors and total time)
  - Floating point performance per processor
  - Scalability

Mesh vs. Matrix Decomposition



Graph Decomposition

•colors indicate processors



Matrix partitioning

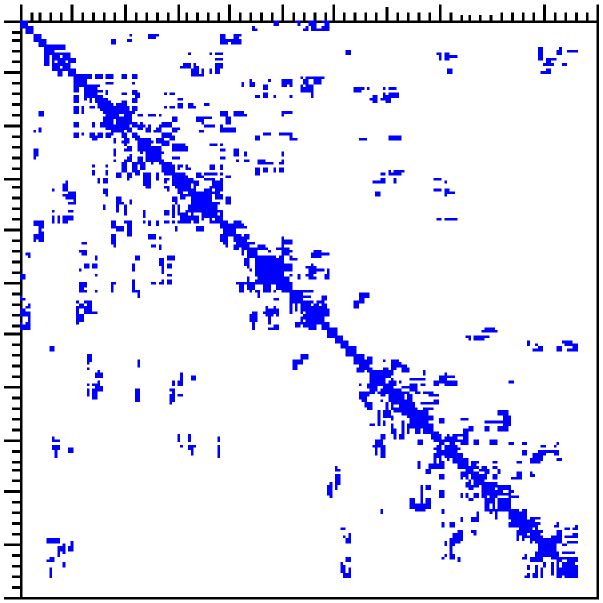
•column (or row) slabs spread over processors

Mesh vs. Matrix Decomposition

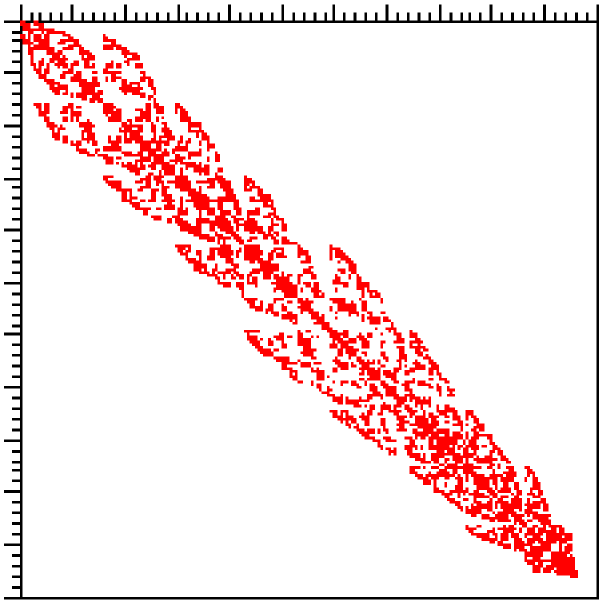
- **Mesh Decomposition**
  - Partition unstructured mesh (or graph of mesh)
    - Spectral partitioning
    - Recursive inertial partitioning
    - Multilevel graph partitioning
  - Each processor receives ‘piece’ of mesh
  - Matrix piece for each mesh piece assembled
  - Solve matrix equation

- **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

Reordering the Sparse System



Original System



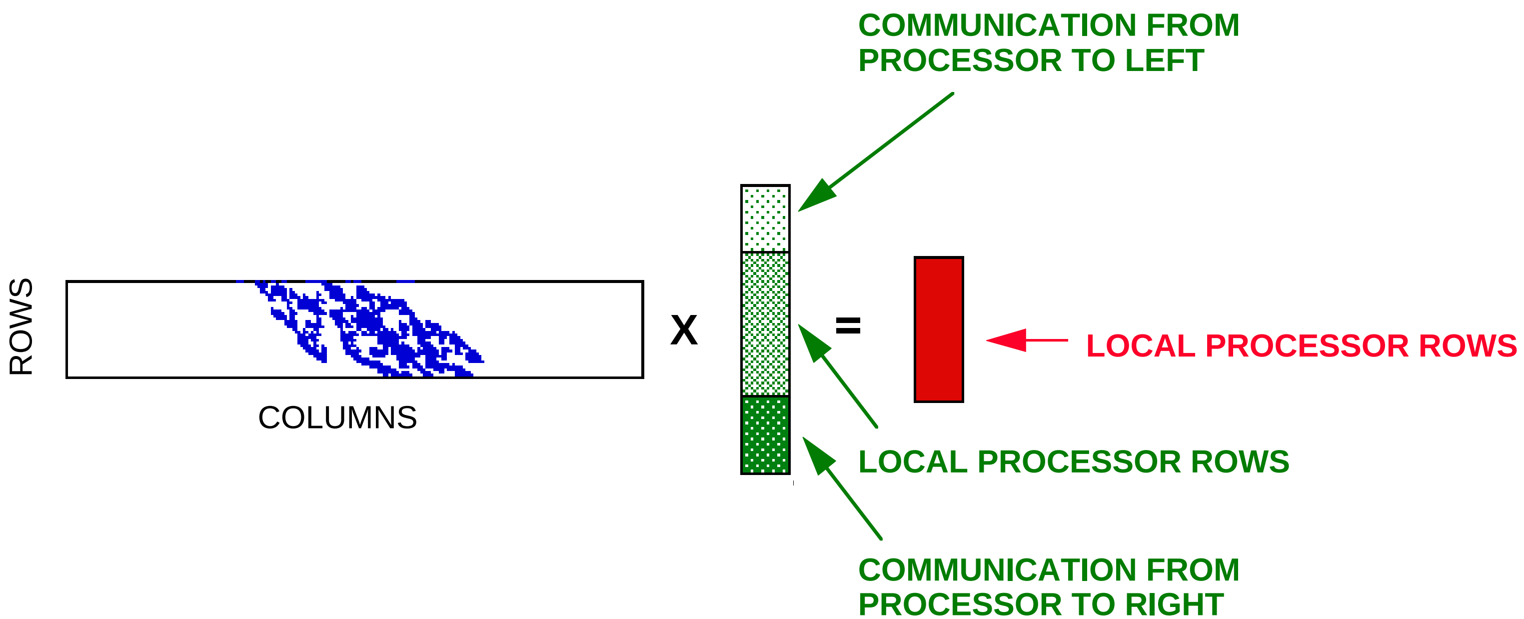
System after Reordering
for Minimum Bandwidth

Using SPARSPAK GENRCM Reordering Routine

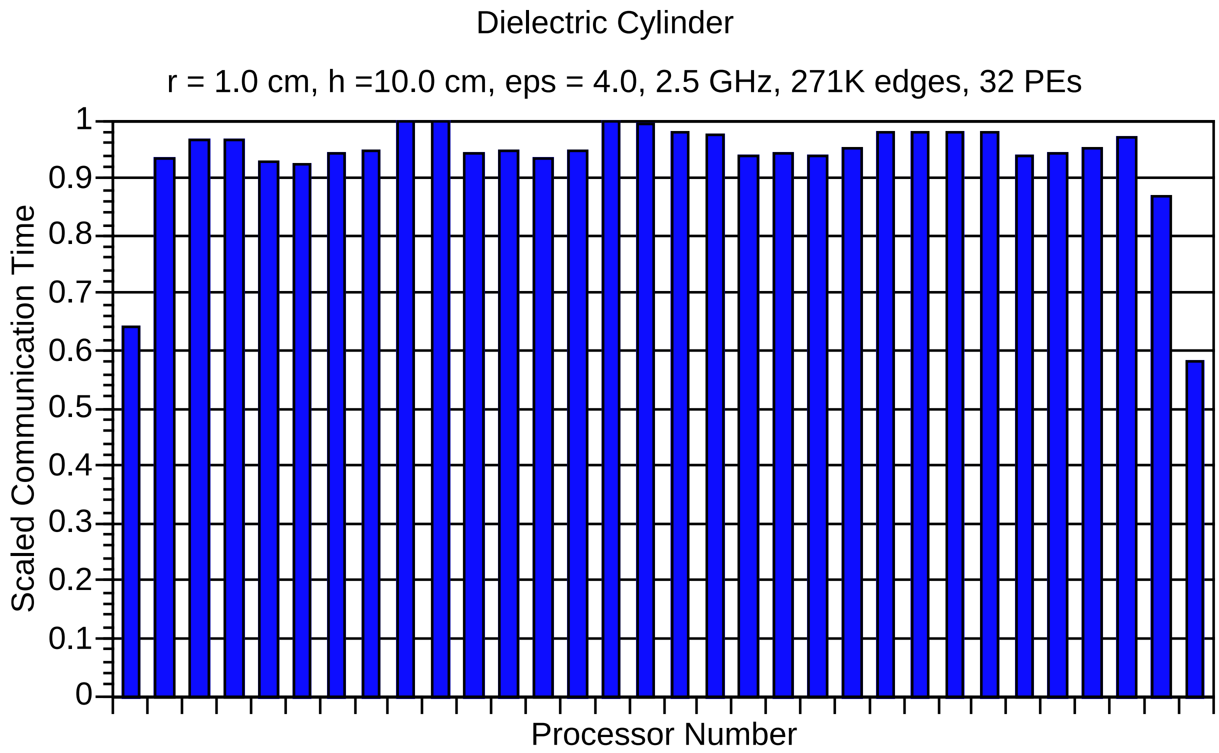
CRAY RESEARCH, INC.

DSK 7/96

Parallel Matrix Vector Multiply



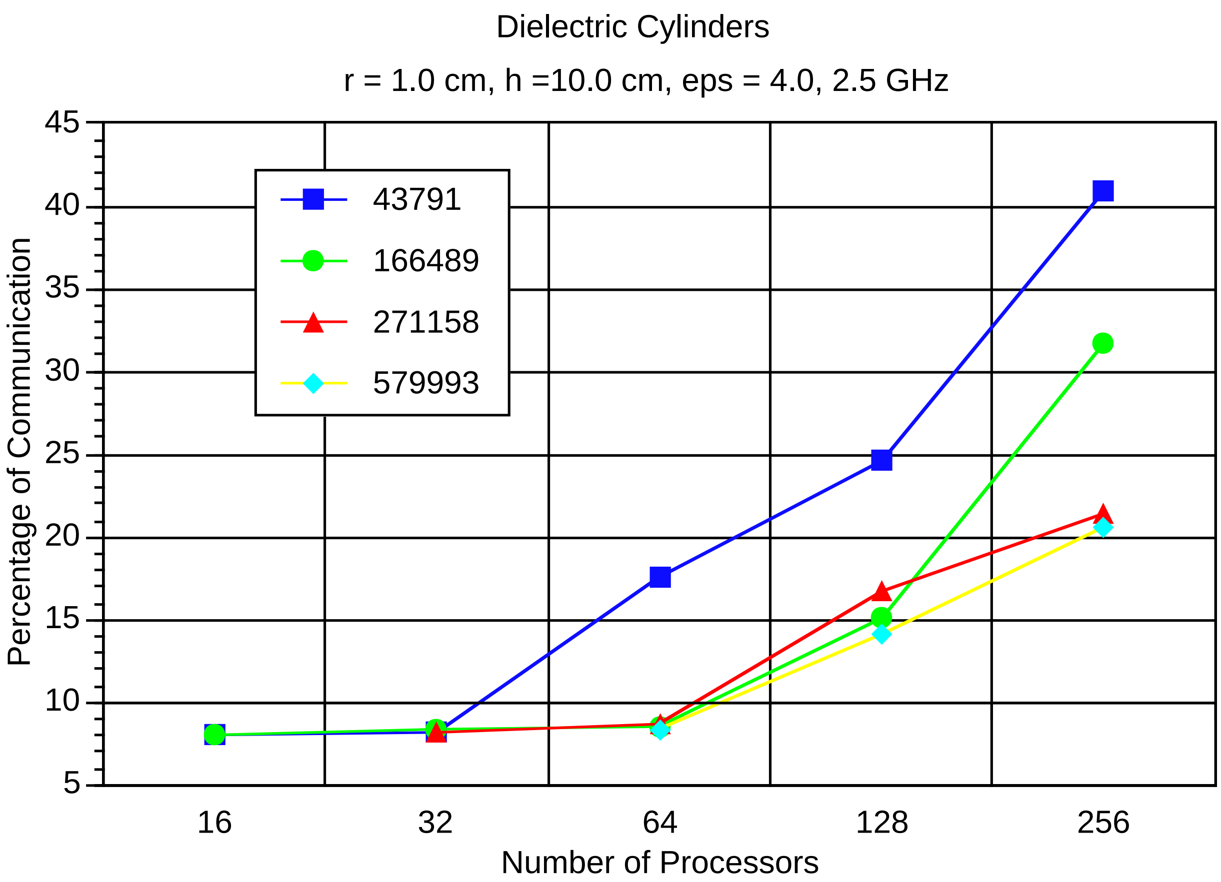
Communication Overhead



CRAY RESEARCH, INC.

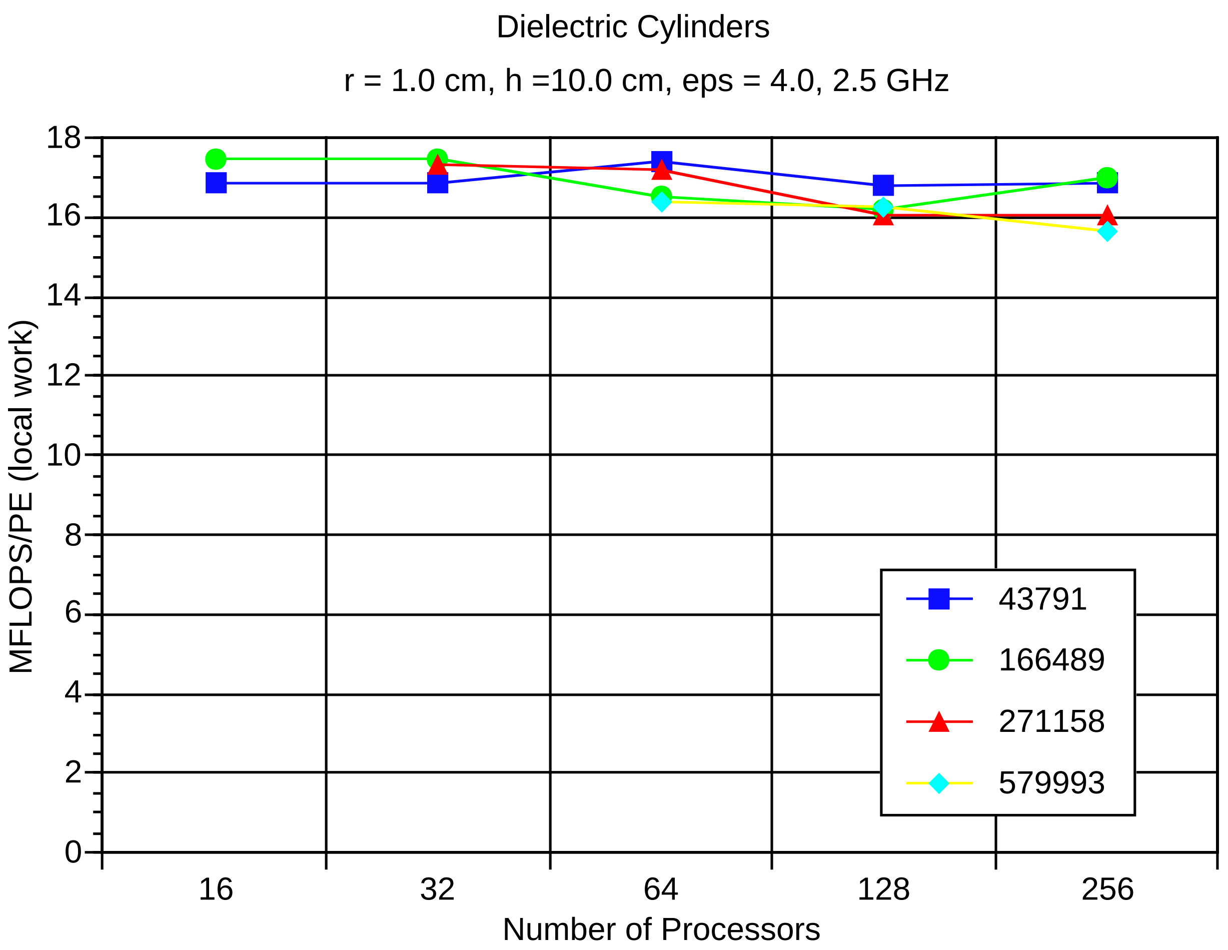
DSK 7/96

Communication Overhead



DSK 7/96

Local Multiply Performance



Comparison with Graph Decomp.

- Matrix Partitioning Compared Against Graph Decomposition
  - Matrix Partitioning (MP)
  - JOSTLE (Walshaw, Cross and Everett, 1995)
  - METIS (Karypis and Kumar, 1995)
- Data Load Balance
  - No difference for three approaches (nearly uniform load balance)
- Communication Load Balance
  - No difference for three approaches except outlier in MP
- Total Amount of Communication
  - MP 1.00
  - JOSTLE 0.26 Possible 6% gain
  - METIS 0.22 for total solver
- Local Matrix-Vector Performance (FLOPS)
  - No difference for three approaches

Mesh vs. Matrix Decomposition Conclusions

- Matrix decomposition code requires little time to run, and took a relatively small effort to program.
- For all but the smallest problems, data load balance is nearly perfect, and communication balance is very good.
- Since the percentage of communication is small, very little would be gained if there was less, or even zero, communication.
- Therefore, matrix decomposition has proven to be a reasonable method to solve this type of problem.

Preconditioning

- Current preconditioning of *K* for the QMR algorithm is done by diagonal scaling.
- Incomplete Cholesky factorization could be used to precondition *K*. Since the Cholesky factorization of a sparse matrix is a dense matrix, the complete factorization is not practical to obtain or work with, even though were it used as a preconditioner, it would cause convergence in 1 iteration.
- Currently being examined is what fraction of the Cholesky factorization of *K* is required to precondition effectively.
- Using a sparse approximate inverse of *K* as a preconditioner will also be studied.

Future Work

- Continue advertising EMLIB
  - http://emlib.jpl.nasa.gov/
- Complete examination of Incomplete Cholesky factorization as a preconditioner
- Examine Approximate Inverse as a preconditioner
- Examine block QMR algorithm
  - Current algorithm is pseudo-block