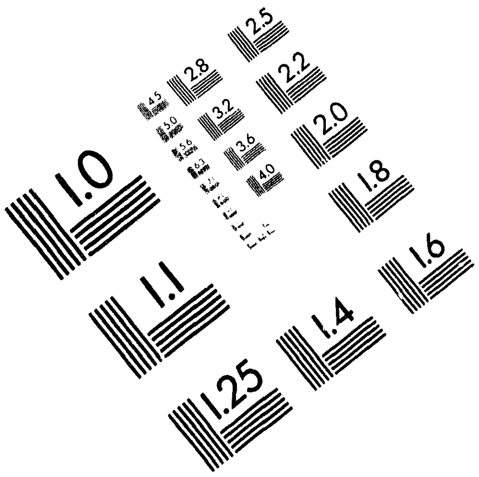
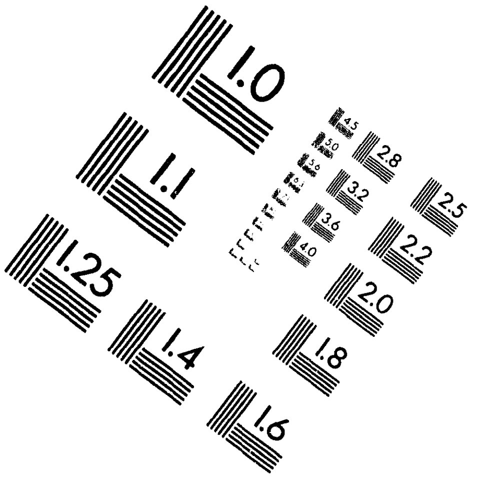
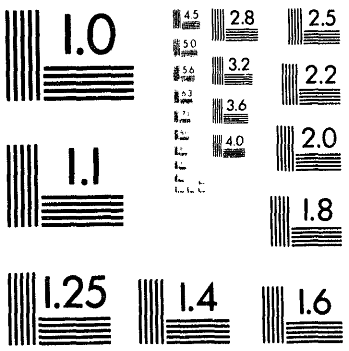
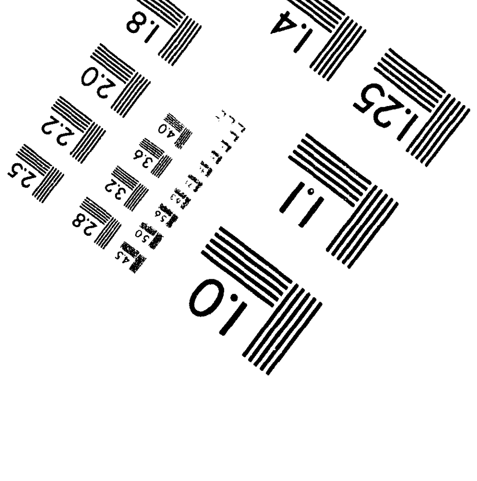
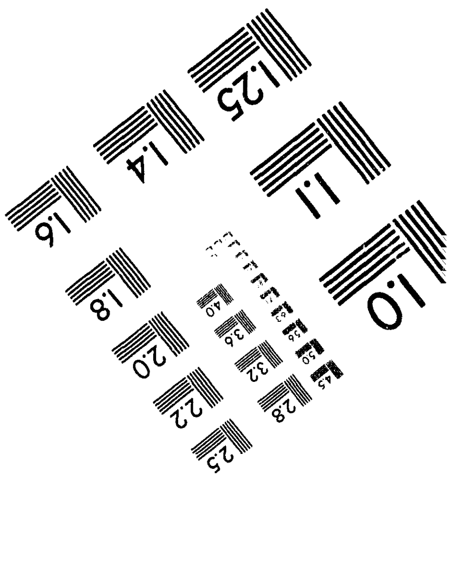



Centimeter
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
mm
1
2
3
4
5
Inches


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

A NEW COMMUNICATION SCHEME FOR THE NEUTRON DIFFUSION NODAL METHOD IN A DISTRIBUTED COMPUTING ENVIRONMENT*

Conf-940407--23

Bernadette L. Kirk
Yousry Azmy

Oak Ridge National Laboratory
P.O. Box 2008, Oak Ridge, TN 37831-6362
615-574-6176

KEYWORDS: Neutron diffusion; distributed computing; nodal method

ABSTRACT

A modified scheme is developed for solving the two-dimensional nodal diffusion equations on distributed memory computers. The scheme is aimed at minimizing the volume of communication among processors while maximizing the tasks in parallel. Results show a significant improvement in parallel efficiency on the Intel iPSC/860 hypercube compared to previous algorithms.

1. INTRODUCTION

The rapid progress in processor speed has accelerated the computation of CPU (central processing unit) intensive algorithms as seen in discretized elliptic partial differential equations like the neutron diffusion equation. Furthermore, the integration of several of these fast processors into a distributed environment has enabled the computational scientist to solve complex problems in parallel. The neutron diffusion equation is a prime example. Kirk and Azmy[1] discussed the two-dimensional nodal method equations as solved on shared memory parallel computers (Sequent Balance 8000) and distributed parallel computers (Intel iPSC/2). Y. H. Kim and N. Z. Chao,[2] in a recent study, performed the parallel solution of the neutron diffusion equation, on a series of transputers. The results by Kirk and Azmy showed that the shared memory architecture produced higher efficiency compared to the distributed memory architecture. In their algorithm, each processor is assigned an equal number of rows/columns to solve for the surface fluxes. The nodal averages are updated at each stage of the iteration using the

*Research sponsored by the Office of Science and Technology, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

newly computed surface fluxes. However, in the hypercube application (i.e., the iPSC/2), the processors have to globally sum the full array of nodal averages to each other in order to obtain the most recent iterate. Also, the convergence test is performed over the entire mesh simultaneously by all utilized processors. These two components of the execution time on the iPSC/2 comprise a large fraction of the total execution time, especially when a large number of processors is used, resulting in the previously observed[1] rapid deterioration in parallel efficiency.

In this paper, we develop a new algorithm for solving the nodal diffusion equations on the Intel iPSC/860 hypercube. The iPSC/860 differs from the iPSC/2 in processor power and communication speed, particularly its much lower message-passing latency.[3] In the hypercube topology, communication time grows linearly with the size of the message which in our application constitutes the node-averaged flux array. It is this portion of the communication time component that we address in our new algorithm.

The Intel iPSC/860 at Oak Ridge National Laboratory (ORNL) is a 128-processor hypercube. Each processor has 8 megabytes of memory. The hypercube has a peak rating of 5 gigaflops.

2. THE NODAL DIFFUSION METHOD EQUATIONS

The two-dimensional nodal method equations for neutron diffusion theory are derived from the general equation:

$$D\left[\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right] - \sigma\phi = -S \ , \tag{1}$$

where
- D = diffusion coefficient
- σ = macroscopic removal cross section
- ϕ = neutron scalar flux
- S = volumetric external source of neutrons.

The detailed derivation of the nodal method equations from Eq. (1) has been presented in Kirk and Azmy[1] and results in three systems of equations: nodal neutron balance equations,

$$\frac{-D_m}{2}\left[\left(\frac{\gamma_m^2 P_m^x}{1 - P_m^x}\right)(\bar{\phi}^y_{+m} - 2\bar{\bar{\Phi}}_m + \bar{\phi}^y_{-m}) + \left(\frac{\gamma_m^2 P_m^y}{1 - P_m^y}\right) \times (\bar{\phi}^x_{+m} - 2\bar{\bar{\Phi}}_m + \bar{\phi}^x_{-m})\right] + \sigma_m\bar{\bar{\Phi}}_m = S_m \ , \qquad m=1,\ldots,I \tag{2}$$

x-current continuity conditions across x = constant edges of the computational cells,

$$\bar{\phi}^y_{+m}\left[D_m\left[\frac{1+\omega^x_m}{2a_mP^x_m}\right]+D_l\left[\frac{1+\omega^x_l}{2a_lP^x_l}\right]\right]$$

$$-\bar{\phi}^y_{-m}\,D_m\left[\frac{1-\omega^x_m}{2a_mP^x_m}\right]-\bar{\phi}^y_{+l}\,D_l\left[\frac{1-\omega^x_l}{2a_lP^x_l}\right] \tag{3}$$

$$-\bar{\bar{\phi}}_m\left[\frac{D_m\omega^x_m}{a_mP^x_m}\right]-\bar{\bar{\phi}}_l\left[\frac{D_l\omega^x_l}{a_lP^x_l}\right]=0\;,$$

and y-current continuity conditions across y = constant edges of the computational cells,

$$\bar{\phi}^x_{+m}\left[D_m\left[\frac{1+\omega^y_m}{2b_mP^y_m}\right]+D_n\left[\frac{1+\omega^y_n}{2b_nP^y_n}\right]\right]$$

$$-\bar{\phi}^x_{-m}\,D_m\left[\frac{1-\omega^y_m}{2b_mP^y_m}\right]-\bar{\phi}^x_{+n}\,D_n\left[\frac{1-\omega^y_n}{2b_nP^y_n}\right] \tag{4}$$

$$-\bar{\bar{\phi}}_m\left[\frac{D_m\omega^y_m}{b_mP^y_m}\right]-\bar{\bar{\phi}}_n\left[\frac{D_n\omega^y_n}{b_nP^y_n}\right]=0\;,$$

where we have used the definitions,

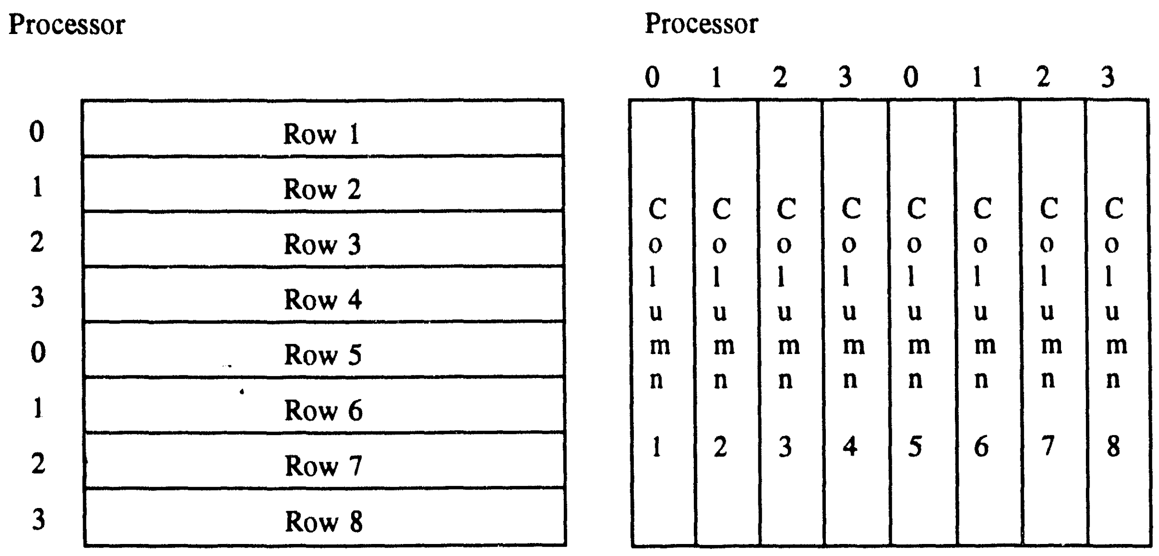
γ^2_m	$\equiv$	σ_m/D_m
σ_m	$\equiv$	average value of σ over cell m
D_m	$\equiv$	average value of D over cell m
P^x_m	$\equiv$	$\tanh(\gamma_m a_m)/\gamma_m a_m$
ω^x_m	$\equiv$	$(\gamma_m a_m P^x_m)^2/(1 - P^x_m)$

and the l'th computational cell is adjacent to the m'th computational cell in the positive x direction such that the two surfaces $x = a_m$ and $x = a_l$ coincide (analogously for the y-current continuity equation).

3. THE ITERATIVE SOLUTION ON PARALLEL COMPUTERS — THE SPANNING TREE MODEL

The iterative solution of the above equations is based on setting initial estimates for the nodal averages $\bar{\bar{\phi}}_m$ $m=1,\ldots, I^2$ in the current continuity equations (for purposes of this study, we will consider square meshes, $I \times I$, only). Equations (3) and (4) then reduce to tridiagonal systems with unknowns $\bar{\phi}^y_{\pm m}$ and $\bar{\phi}^x_{\pm m}$.

Our previous algorithm (Refs. 1 and 4) proceeded along the following scenario — processors alternately solve the tridiagonal systems corresponding to rows and columns as seen in Fig. 1. The integers printed outside the outer box in Fig. 1 denote the processor identification assigned to the corresponding row or column in the 8×8 mesh and 4-processor case. At the end of each iteration, processor 0 will have values of $\bar{\phi}^x$ corresponding to rows 1 and 5, and values for $\bar{\phi}^y$ corresponding to columns 1 and 5; and so on. Thus, each processor will be solving four tridiagonal systems of equation in the case shown in Fig. 1.



Row assignment to processors on an 8×8 mesh.

Column assignment to processors on an 8×8 mesh.

Figure 1: Assignment of 4 processors for an 8×8 mesh according to the spanning tree scheme.

After the tridiagonal matrix equations are solved, each processor computes its contribution to the node-averaged flux, $\bar{\bar{\phi}}$, via the balance equation, Eq. (1). A global summation of these contributions across processors is accomplished by the Intel system routine gdsum, which implements a spanning tree routing for the global summation phase and the broadcast phase (see Fig. 2). The scheme described has these characteristics: a) the processors are in a minimally configured distance mode, and b) even in the best case as discussed in Ref. 1, there is overhead in mapping a two-dimensional array to a single array for the message exchange. The second property is evident from the model for communication in Azmy and Kirk.[4]

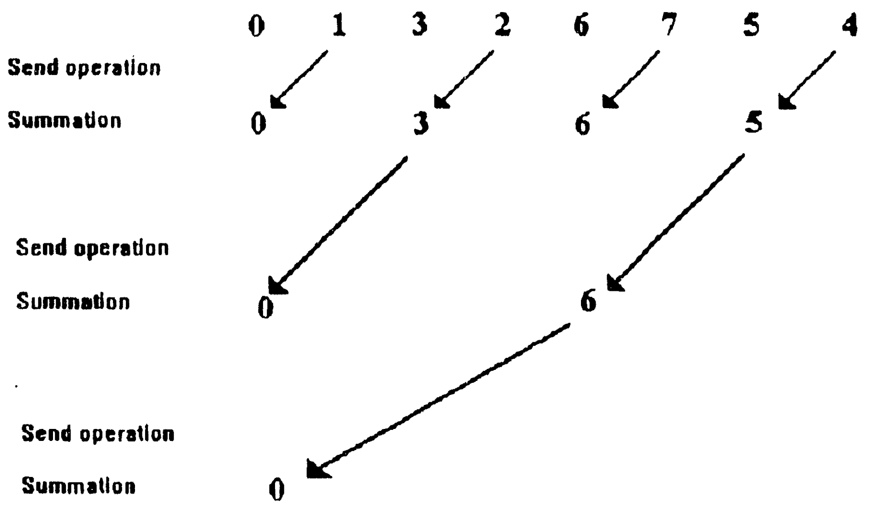


Figure 2: Spanning tree for 8 processor GDSUM model.

4. THE ITERATIVE SOLUTION ON PARALLEL PROCESSORS - THE PARQUET SCHEME

An alternative method for handling processor communication to reduce contention among processors is based on global combine operations which offer the least communication penalty.[5] The idea is to reduce the volume of data traffic across the network.

We start with a new row and column assignment for the participating processors. Suppose there are P processors, then the first $P/2$ processors are assigned to solve contiguous columns, and the remaining set of $P/2$ processors is assigned to solve contiguous rows. This scheme has the following properties (assuming $P/2$ and $2I/P$ are integers):

1. All messages consisting of contributions to the nodal averaged flux are the same length $(2I/P)^2$ and are stored in the vector $\bar{\bar{\phi}}_m$, $m=1, \ldots, I^2$.

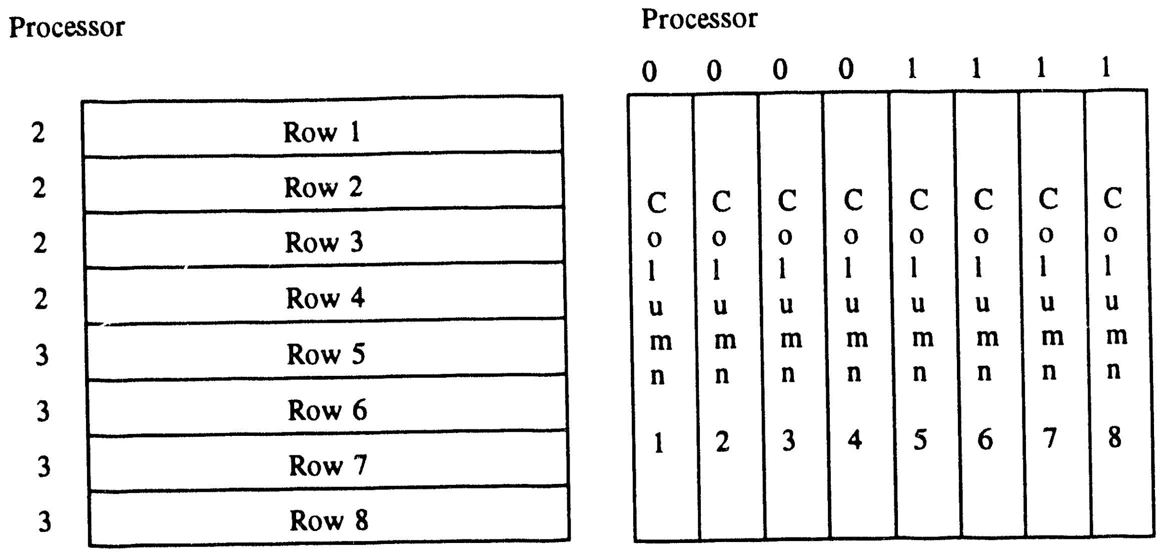
2. Processors solving rows do not need to communicate their nodal averaged contributions to one another; similarly for processors solving columns.

3. At each step, there will be one SEND followed by one RECEIVE per processor.

4. The number of processors must be divisible by 4.

Figure 3 shows the new designation for a 4-processor case or an 8×8 mesh.

Upon conclusion of the tridiagonal systems solutions, the participating processors communicate the contribution they computed to the new iterate of $\bar{\bar{\phi}}$ following the pattern shown in Fig. 3. The four internal boxes depicted in Fig. 4 represent the four quadrants of the full mesh shown in Fig. 3; thus each such box contains 4×4 computational cells. The entries within the internal boxes denote the sending and receiving processors of the $\bar{\bar{\phi}}$ contributions for the computational cells within each quadrant box. In



Row assignment to processors on an 8 × 8 mesh.

Column assignment to processors on an 8 × 8 mesh.

Figure 3: Assignment of 4 processors for an 8 × 8 mesh according to the Parquet scheme.

each instance the receiving processor sums the received message to its own contribution to $\overline{\overline{\phi}}$ along rows or columns as depicted in Fig. 4, thus generating the new iterate of $\overline{\overline{\phi}}$ for that particular quadrant, and tests its convergence compared to a pointwise relative convergence criterion. In the case of 4 processors considered here, the entire communication stage is concluded in one step. This process is easily extendible to $P > 4$, encompassing $P/4$ steps analogous to the one described above. This extension is illustrated via an example of an 8 × 8 mesh on 8 processors, shown in Fig. 5.

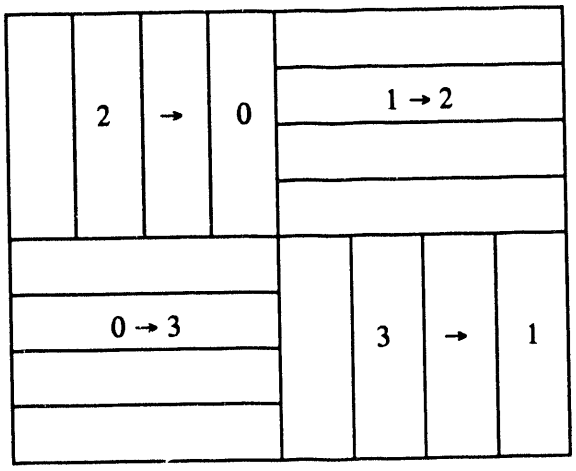
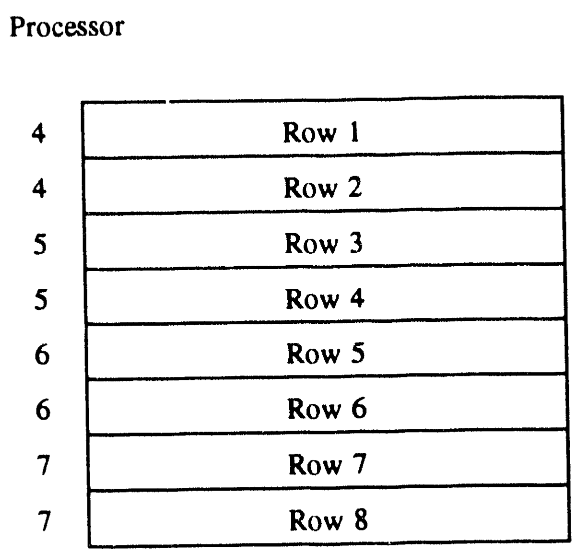
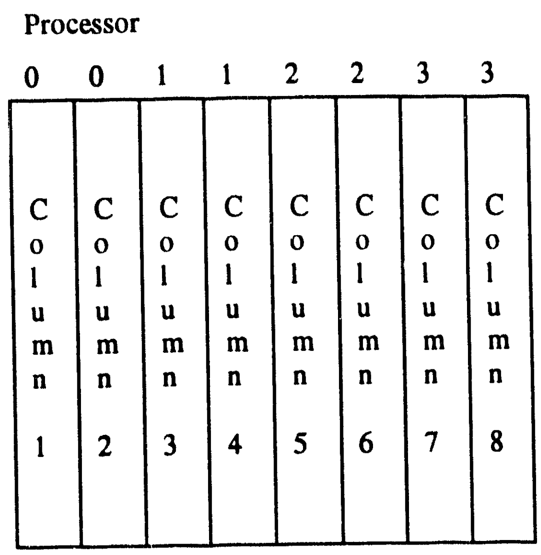


Figure 4: Communication pattern for the 4-processors case, 8 × 8 mesh.

For the 8-processor case, there are $P/4 = 2$ steps involved in the communication, as seen in Fig. 6. Each processor will solve $(2I/P)=2$ rows or columns. Each message is of length $(2I/P)^2=4$.



Row assignment to processors on an 8×8 mesh.



Column assignment to processors on an 8×8 mesh.

Figure 5: Assignment of 8 processors for an 8×8 mesh according to the Parquet scheme.

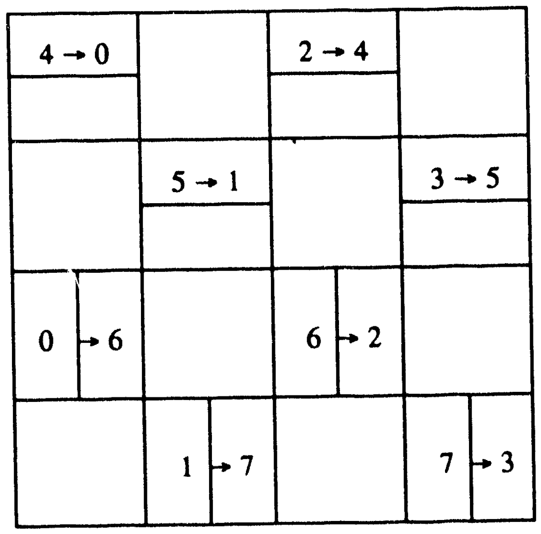


Figure 6a: Communication pattern for the 8 processors — step 1.

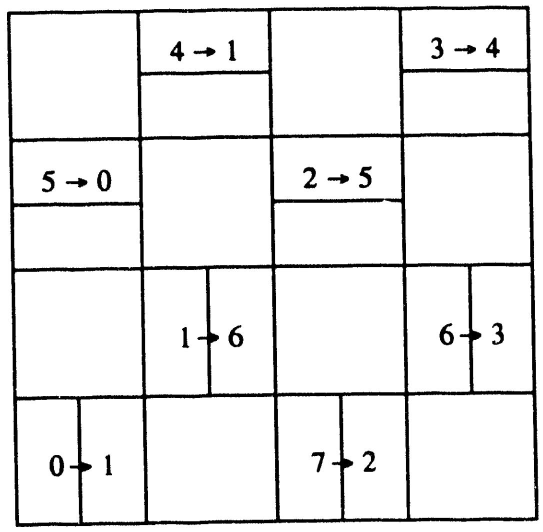


Figure 6b: Communication pattern for the 8 processors — step 2.

At the end of the message passing stage, the processor will have the new iterate of $\overline{\overline{\Phi}}$ on the 2×4 or 4×2 cells as shown in Fig. 7. As before this is followed by the convergence test and so on. The resemblance of the pattern in Figs. 4 and 7 to that of a Parquet tile explains the name of our new scheme.

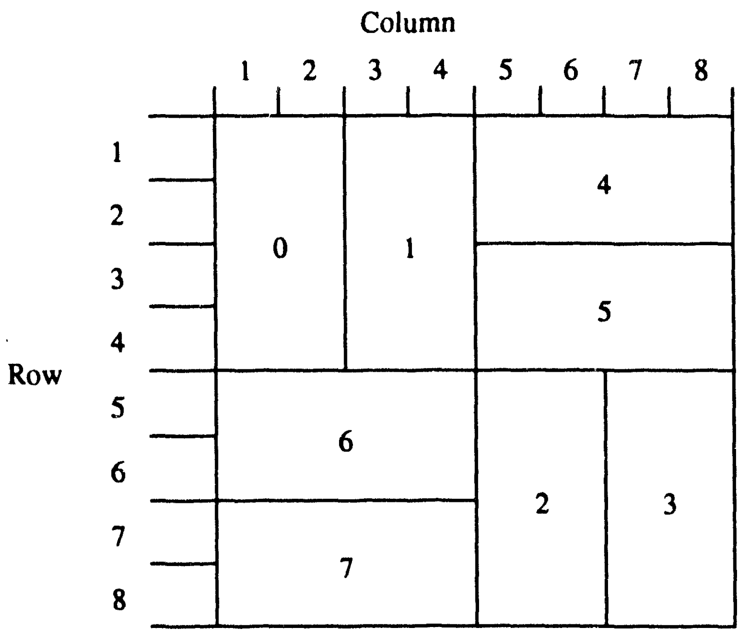


Figure 7: Distribution of new iterate vector among participating processors whose id numbers appear in the corresponding vector.

5. RESULTS

In order to test the performance of the Parquet scheme, we implemented it in our parallel nodal diffusion code,[1] and used it to solve a simple test problem on a sequence of 32 $\times$ 32, 64 $\times$ 64, and 96 $\times$ 96 meshes. For each mesh, we measure the execution time on various number of processors for the Parquet and the spanning tree schemes. The measured parallel efficiency, $E = \frac{100 \times T_1}{PT_p}$, where T_1 and T_p are the measured execution times for a given mesh using 1, and P processors, respectively, are presented in Tables I, II, and III.

Table I: Efficiency (%) for 32 $\times$ 32 mesh.

Number of processors	Spanning tree	PARQUET
4	75.3	89.5
8	51.3	81.0
16	28.6	57.6
32	13.8	40.4
64	7.3	18.3

Table II: Efficiency (%) for 64 $\times$ 64 mesh.

Number of processors	Spanning tree	PARQUET
4	76.6	91.8
8	52.9	87.7
16	30.0	78.5
32	14.8	56.5
64	6.8	43.0

Table III: Efficiency (%) for 96 × 96 mesh.

Number of processors	Spanning tree	PARQUET
4	76.9	91.3
8	52.8	87.6
16	30.2	81.8
32	14.9	68.8
64	6.8	56.3

The results in the preceding tables clearly show the PARQUET method to be superior to the spanning tree. More importantly, the Parquet scheme scales up with mesh size, providing a greater potential for higher efficiency on larger meshes, and three dimensional applications. In the 64-processor case, for example, the spanning tree saturated at about 7% efficiency for each of the mesh sizes. The Parquet scheme, on the other hand, continued to rise in efficiency as the mesh is refined.

Finally, we wish to note that our present implementation, hence the above results also, of the Parquet scheme is still in its preliminary stages, and we anticipate further performance improvement as it matures.

REFERENCES

1. Bernadette L. Kirk and Yousry Y. Azmy, "An Iterative Algorithm for Solving the Multidimensional Neutron Diffusion Nodal Method Equations on Parallel Computers," *Nucl. Sci. Eng.*, 111, 57-65 (1992).

2. Yong Hee Kim and Nam Zin Chao, "Parallel Solution of the Neutron Diffusion Equation with the Domain Decomposition Method on a Transputer Network," *Nucl. Sci. Eng.*, 114, 252-270 (1993).

3. Thomas H. Dunigan, *Communication Performance of the Intel Touchstone Delta Mesh*, Oak Ridge National Laboratory report ORNL/TM-11983, January 1992.

4. Y. Y. Azmy and B. L. Kirk, "Performance Modeling of Parallel Algorithms for Solving Neutron Diffusion Problems," submitted for publication in *Concurrency: Practice and Experience*, 1993.

5. Robert A. van de Gejin, *LAPACK Working Note 29 on Global Combine Operations*, University of Tennessee, CS-91-129, April 1991.

DATE
FILMED
9/29/94

END