Dense matrix algorithms

• We are going to study algorithms involving dense matrices (as opposed to sparse matrices)

• A very important issue is how to map a matrix onto processors
  – the combination of proper mapping and efficient algorithm is performance critical

• Main mapping schemes are:
  – striped partitioning
  – blocked partitioning
  – checkerboard partitioning
Striped partitioning

- Ways of partitioning a $16 \times 16$ matrix on 4 processors

(a) Columnwise block striping

(b) Rowwise cyclic striping
Checkerboard partitioning

- Ways of partitioning a $8 \times 8$ matrix on 16 processors
- Checkerboard partitioning splits both rows and columns

<table>
<thead>
<tr>
<th>$P_0$</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
<th>$P_8$</th>
<th>$P_9$</th>
<th>$P_{10}$</th>
<th>$P_{11}$</th>
<th>$P_{12}$</th>
<th>$P_{13}$</th>
<th>$P_{14}$</th>
<th>$P_{15}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>(0,2)</td>
<td>(0,4)</td>
<td>(0,6)</td>
<td>(1,0)</td>
<td>(1,2)</td>
<td>(1,4)</td>
<td>(1,6)</td>
<td>(2,0)</td>
<td>(2,2)</td>
<td>(2,4)</td>
<td>(2,6)</td>
<td>(3,0)</td>
<td>(3,2)</td>
<td>(3,4)</td>
<td>(3,6)</td>
</tr>
<tr>
<td>(0,1)</td>
<td>(0,3)</td>
<td>(0,5)</td>
<td>(0,7)</td>
<td>(1,1)</td>
<td>(1,3)</td>
<td>(1,5)</td>
<td>(1,7)</td>
<td>(2,1)</td>
<td>(2,3)</td>
<td>(2,5)</td>
<td>(2,7)</td>
<td>(3,1)</td>
<td>(3,3)</td>
<td>(3,5)</td>
<td>(3,7)</td>
</tr>
<tr>
<td>(1,0)</td>
<td>(1,2)</td>
<td>(1,4)</td>
<td>(1,6)</td>
<td>(2,0)</td>
<td>(2,2)</td>
<td>(2,4)</td>
<td>(2,6)</td>
<td>(3,0)</td>
<td>(3,2)</td>
<td>(3,4)</td>
<td>(3,6)</td>
<td>(4,0)</td>
<td>(4,2)</td>
<td>(4,4)</td>
<td>(4,6)</td>
</tr>
<tr>
<td>(1,1)</td>
<td>(1,3)</td>
<td>(1,5)</td>
<td>(1,7)</td>
<td>(2,1)</td>
<td>(2,3)</td>
<td>(2,5)</td>
<td>(2,7)</td>
<td>(3,1)</td>
<td>(3,3)</td>
<td>(3,5)</td>
<td>(3,7)</td>
<td>(4,1)</td>
<td>(4,3)</td>
<td>(4,5)</td>
<td>(4,7)</td>
</tr>
</tbody>
</table>

Copyright (c) 1994 Benjamin/Cummings Publishing Co.

(a) Block-checkerboard partitioning  
(b) Cyclic-checkerboard partitioning
Matrix Transposition: mesh \((n^2=p)\)

- Simple case is \(n^2 = p\) i.e. one element per processor
- Algorithm for checkerboard partitioning

(a) Communication steps

(b) Final configuration
Matrix Transposition: mesh $(n^2 > p)$

- Longest path: $2\sqrt{p}$  - Block size: $n^2/p$
- Total comm. time: $2(t_s + t_w n^2/p) \sqrt{p}$
- Local exchange time: $n^2/2p$
Recursive Transposition Alg. (RTA)

- RTA for a $8 \times 8$ matrix
- Since each recursive step reduces the size of the subcubes by a factor of four, there is a total of $\log_4 p$ or $(\log p)/2$ steps
Matrix Transposition: hypercube

- Block-checkerboard mapping, $8 \times 8$ matrix, 16 proc. Hypercube
- The steps of the RTA involve smaller and smaller subcubes
  - corresponding nodes across subcubes are hypercube itself

---

Division of the matrix into four blocks and exchange of top-right and bottom-left blocks

Division of each block into four subblocks and exchange of top-right and bottom-left subblocks
Transposition: striped partitioning

- Simple case: $n \times n$ matrix on $n$ processor (one row per proc)
  - Element $[i, j]$ moves to position $[j, i]$
- General case ($p < n$): blocks are moved, then internally transposed