Electrical Engineering and Computer Science

EECS 358 - INTRODUCTION TO PARALLEL COMPUTING

Lecture 16

Parallel Sparse Solvers

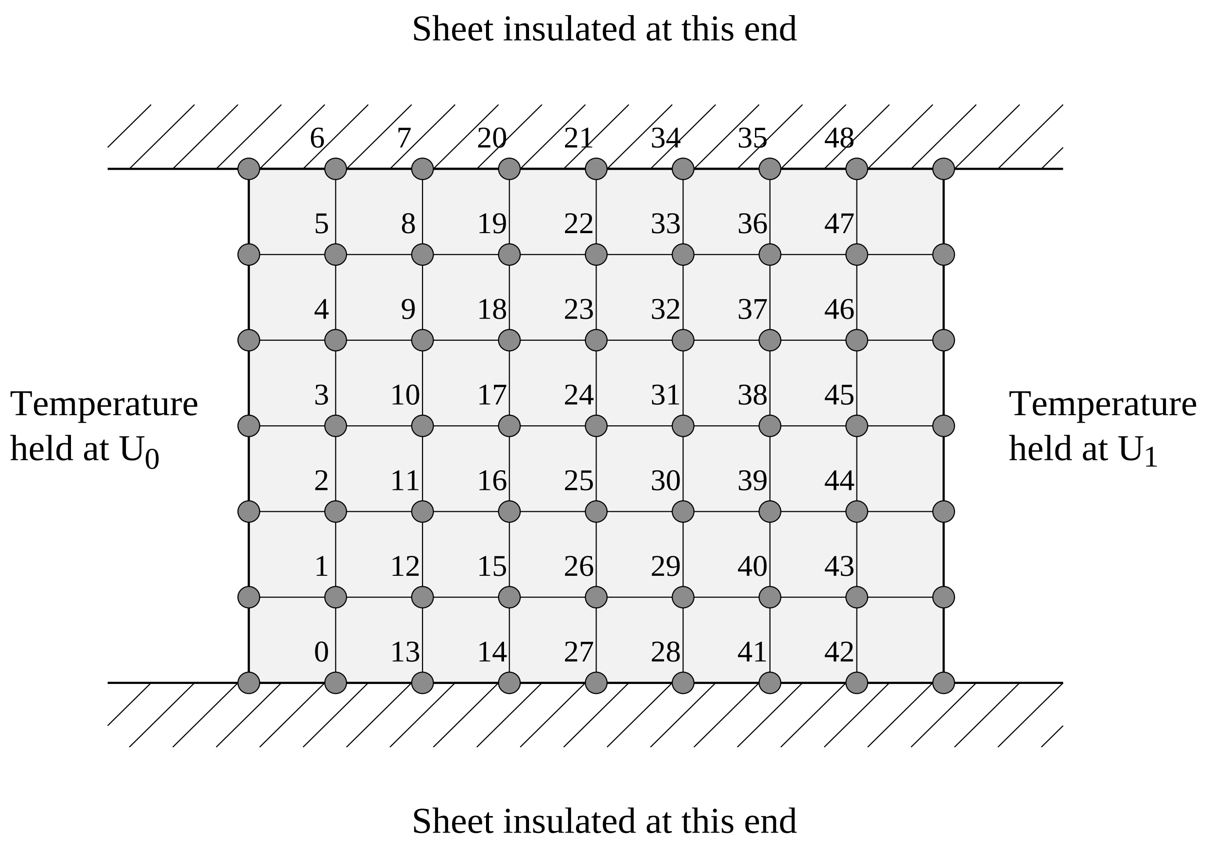
Outline

- Motivation for sparse solvers
- Basic Operations
- Iterative Methods for Sparse Linear Systems

Why Sparse Linear System Solvers

- Solving systems of linear equations is at core of many problems in science and engineering.
- In physical systems, continuous physical domain is discretized by imposing a grid over the domain.
- Solving the mathematical model over discrete domain involves obtaining values of certain physical quantities.

An Example


Sheet insulated at this end
6
7
20
21
34
35
48
5
8
19
22
33
36
47
4
9
18
23
32
37
46
Temperature held at U_0
3
10
17
24
31
38
45
Temperature held at U_1
2
11
16
25
30
39
44
1
12
15
26
29
40
43
0
13
14
27
28
41
42
Sheet insulated at this end

Storage Schemes for Sparse Matrices

- Instead of storing an n X n sparse matrix as an n X n array, store in coordinate form in linear array with q nonzero entries, with two additional arrays storing I and J locations.

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 3 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 8 \\ 9 & 0 & 0 & 10 & 11 & 12 \\ 0 & 13 & 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{pmatrix}$$

(a) A sparse matrix

VAL

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

I

2	2	3	0	5	4	3	0	0	1	4	1	3	2	3

J

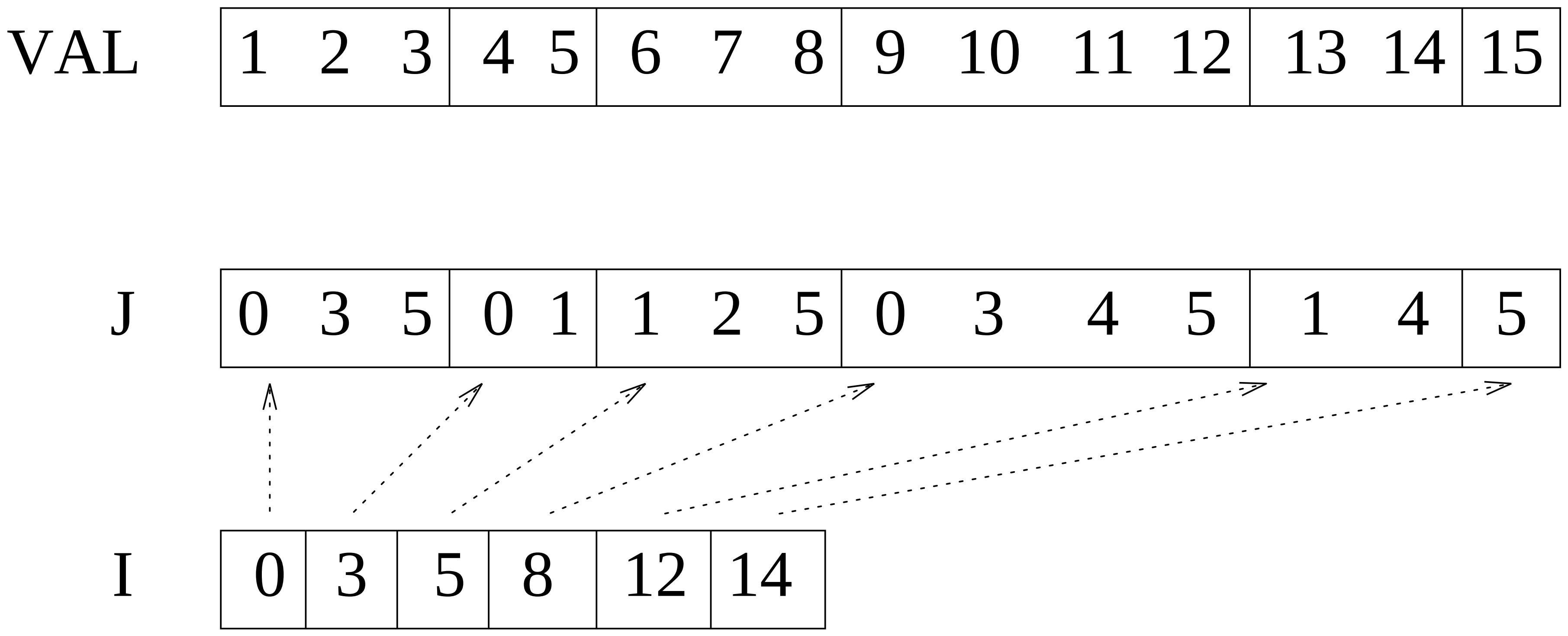
5	1	5	0	5	4	0	3	5	1	1	0	4	2	3

(b) Storage in coordinate format

Compressed Sparse Row (CSR) Format

- A q X 1 array VAL containing nonzero elements stored in order of rows.
- a q X 1 array J that stores the column numbers of each nonzero element.
- An n X 1 array I, the ith entry of which points to first entry of ith row in VAL and J.

Example of CSR



Diagonal Storage Format

- Suitable for sparse matrices where nonzero entries are arranged in few diagonals
- Consider n X n matrix consisting of d diagonals
- Store nonzero diagonals as n X d array VAL
- A d X 1 array sores offest of each diagonal with respect to principal diagonal

Example of Diagonal Format

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 0 \\ 8 & 0 & 9 & 10 & 11 & 0 \\ 0 & 13 & 0 & 0 & 14 & 15 \\ 0 & 0 & 16 & 0 & 17 & 18 \end{pmatrix}$$

(a) A sparse matrix

VAL

-	-	1	2
-	3	4	5
-	6	7	0
8	9	10	11
13	0	14	15
16	17	18	-

OFFSET

-3	-1	0	1

(b) Storage in diagonal format

Ellpack-Itpack Format

- Suitable for general sparse matrices
- Assume maximum number of elements in any row is m
- Store n X n matrix as two n X m matrices VAL and J
- Each row of VAL contains nonzero entries of corresponding row of sparse matrix.
- Array J stores column numbers of corresponding entries in VAL.

Example of Ellpack/Itpack Format

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 6 & 7 & 0 & 0 & 0 \\ 8 & 0 & 9 & 10 & 11 & 0 \\ 0 & 13 & 0 & 0 & 14 & 15 \\ 0 & 0 & 16 & 0 & 17 & 18 \end{pmatrix}$$

(a) A sparse matrix

VAL

1	2	-	-
3	4	5	-
6	7	-	-
8	9	10	11
13	14	15	-
16	17	18	-

J

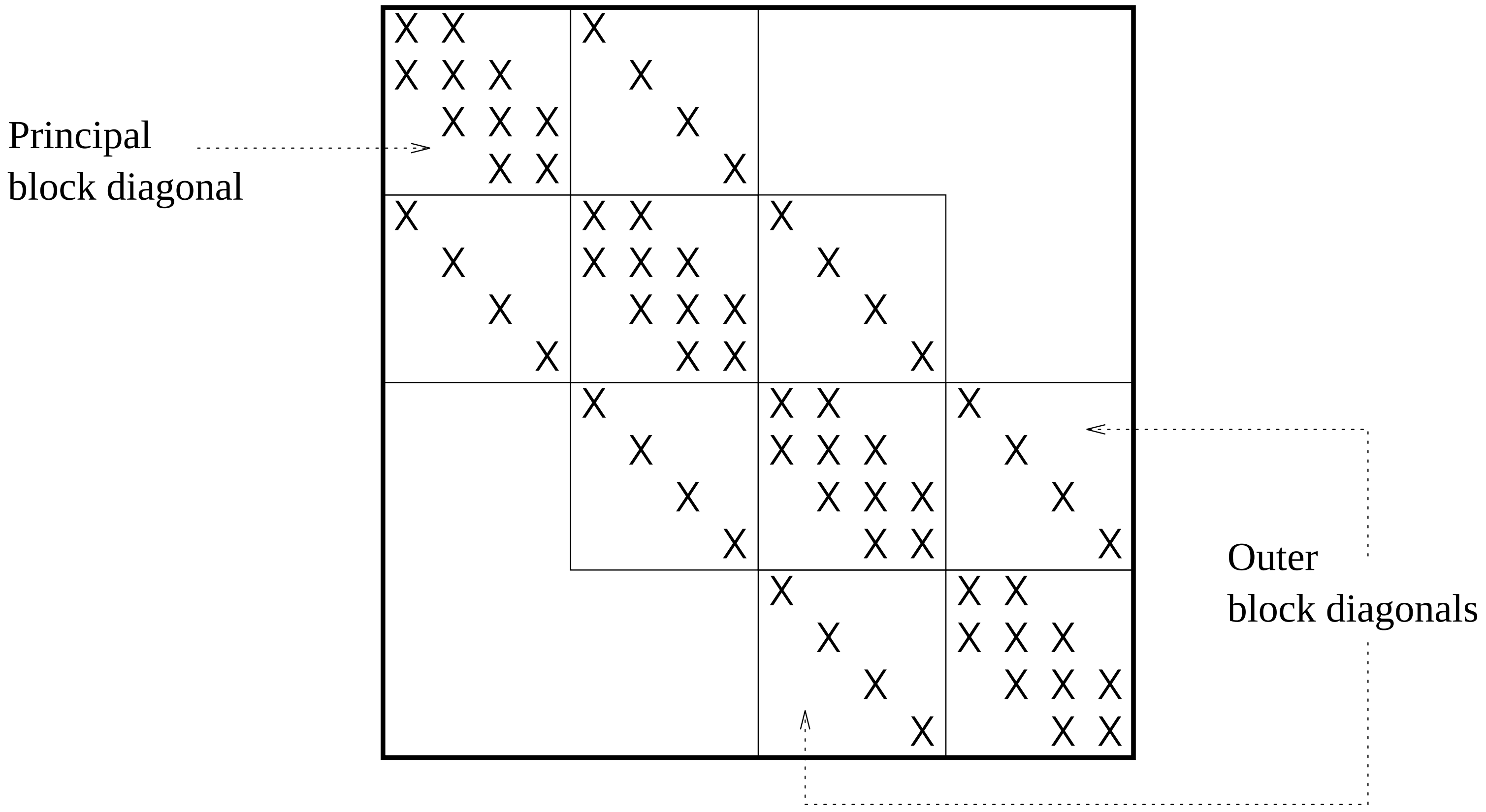
0	1	-1	-
0	1	2	-1
1	2	-1	-
0	2	3	4
1	4	5	-1
2	4	5	-1

(b) Storage in Ellpack-Itpack format

Sparse Matrix-Vector Multiplication

- Multiplication of sparse matrix with dense vector is a key step in solving linear equations using iterative methods.
- Often the multiplication is tuned to the specific characteristics of the sparse matrix
  - Matrices in which all non zeros are arranged in few diagonals (block triagonal matrices)
  - Unstructured Sparse matrices, no structure
  - Banded sparse matrices

Example of Block Tridiagonal Matrix



Example Finite Difference Application

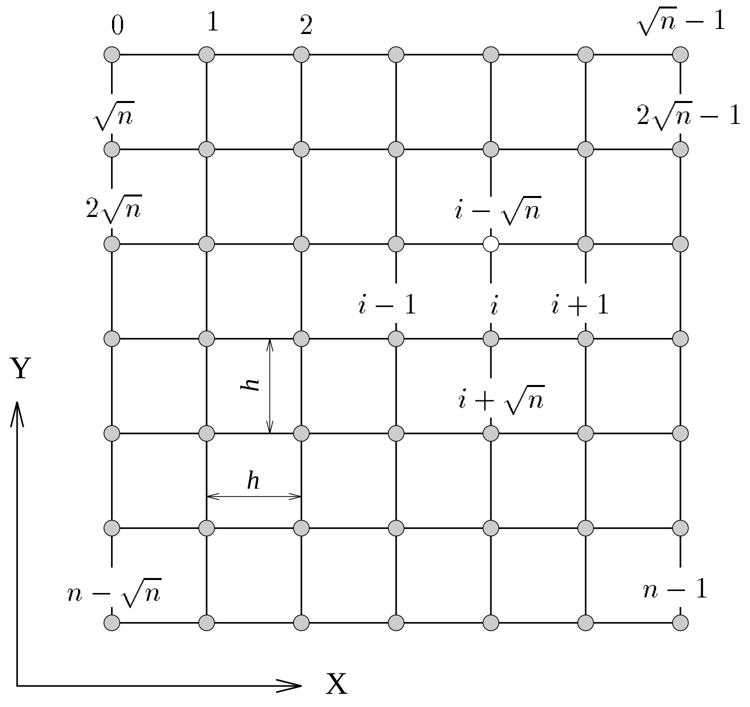
- Assume a partial differential equation

$$\frac{d^2u}{dX^2} + \frac{d^2u}{dY^2} = f$$

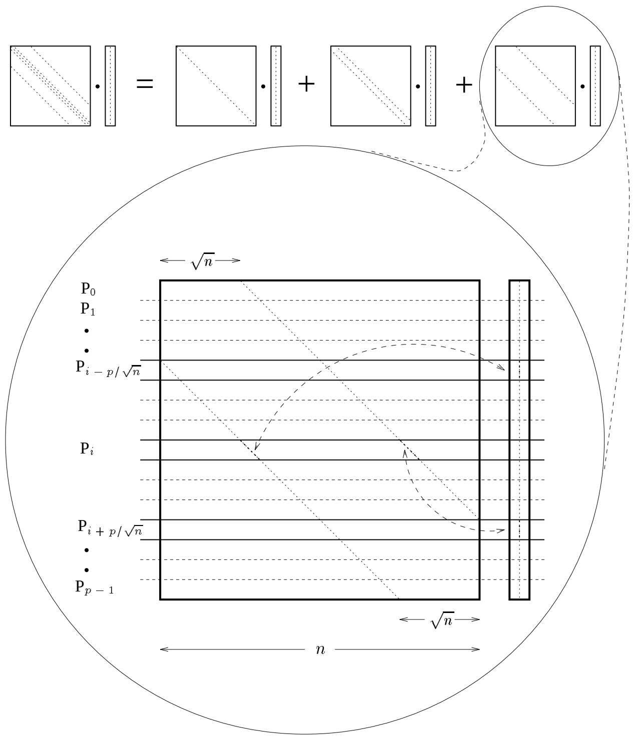
- Finite difference approximation on regular square grid yields

$$x[i - \sqrt{n}] + x[i-1] - 4x[i] + x[i+1] + x[i+\sqrt{n}] = h^2 f$$

Example of Grid



Parallel Block Triagonal Matrix Multiplication



Parallel Block Tridiagonal Matrix Multiplication

- Each processor gets n/p elements of the vector and each diagonal of the matrix.
- Diagonal storage scheme is natural choice
- Array VAL is distributed using block-striped partitioning and array OFFSET is not required since it is implicit.
- Each row requires five vector elements for multiplication
- Vector elements on principal diagonal is available on same processor, vector elements on inner diagonal is also on same processor except on boundaries, and vector elements of offdiagonal are on different processors.

Analysis of Parallel Matrix Multiplication

- Communication takes $2(t_s + t_w)$ time at each processor for inner diagonals
- Communication for offdiagonal elements takes roughly $2(t_s + t_w(n/p))$ if number of elements per processor (n/p) is smaller than $\sqrt{n}$
- Run time of parallel algorithm

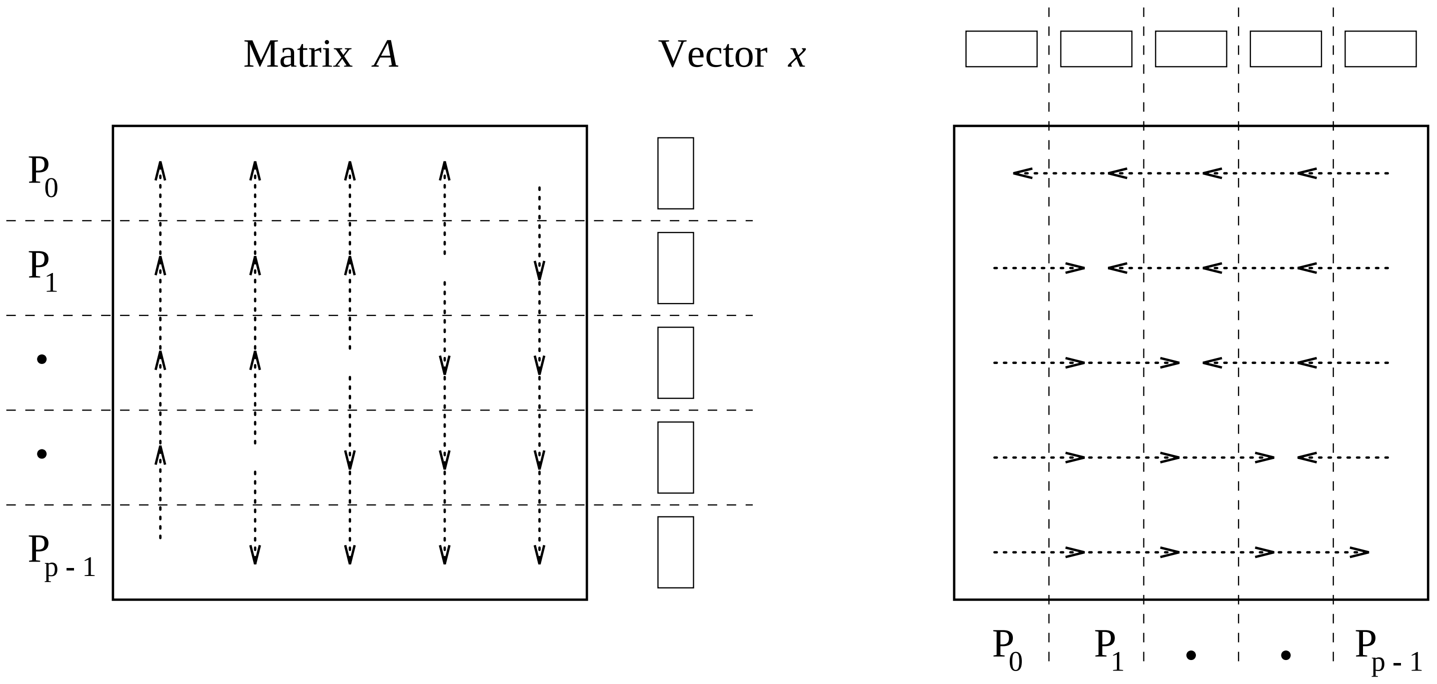
$$T_P = 5t_c n/p + 4t_s + 2t_w(1 + (n/p))$$

Unstructured Sparse Matrix Multiplication

- Use Ellpack-Itpack format which is row oriented
- Partition arrays VAL and J such that each processor receives n/p rows or mn/p nonzero elements
- Vector x is partitioned uniformly so each processor gets n/p elements
- Requires an all-to-all broadcast among processors which takes time $t_s log(p) + t_w n$
- Parallel runtime is

$$T_P = t_c mn/p + t_s log(p) + t_w n$$

Example of Unstructured Matrix Vector Multiplication



(a) All-to-all broadcast with rowwise striping

(b) Multinode accumulation with columnwise striping

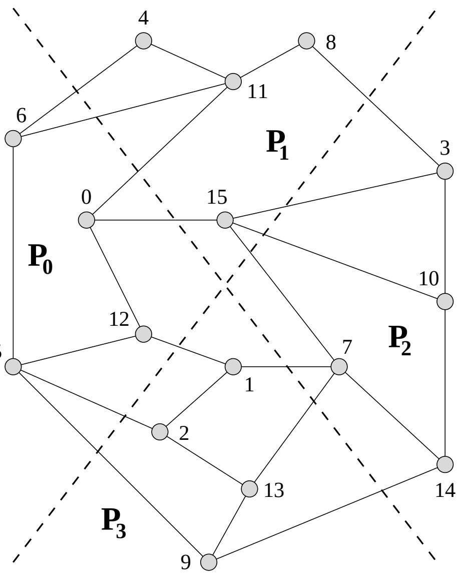
Scalable Parallel Implementation

- If we assume a symmetric sparse matrix for a problem which has a planar graph
- Let G(A) be a graph such that there is edeg between nodes i and j if A[i,j] $\neq$ 0, and A[j,i] $\neq$ 0.
- Partition nodes of graph among processors to minimize edges cut (correspond to communication)
- Perform matrix vector multiplication by partitioning vector elements so that ith element resides on same processor as ith row of matrix.
- Processors perform communication only for those rows of A that correspond to nodes of G(A) lying at processor's partition

Example of Scalable Parallel Implementation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	X											X	X			X	P_0
1		X	X					X					X				P_3
2		X	X			X								X			P_3
3				X					X		X					X	P_2
4					X		X					X					P_1
5			X			X	X			X			X				P_0
6					X	X	X					X					P_0
7		X						X						X	X	X	P_2
8				X					X			X					P_1
9						X				X				X	X		P_3
10				X							X				X	X	P_2
11	X				X		X		X			X					P_1
12	X	X				X							X				P_0
13			X					X		X				X			P_3
14								X		X	X				X		P_2
15	X			X				X			X					X	P_1

(a) A 16 x 16 symmetric random sparse matrix



(b) The associated graph and its four partitions

Iterative Methods for Sparse Linear Systems

- Iterative methods are techniques to solve systems of equations of form $A.x = b$ that generate a sequence of approximations to the solution vector x.

- In each iteration, the matrix A is used to perform a matrix vector multiplication.

- Number of iterations required to solve system is data dependent.

- Iterative methods do not guarantee solution for all systems.

Jacobi Iterative Method

- The ith equation of system can be written as

$$\sum_{j=0}^{n-1} A[i,j]x[j] = b[i]$$

- If all the diagonal elements of matrix A are non-zero, we can rewrite above as

$$x[i] = \frac{1}{A[i,i]}(b[i] - \sum_{j \neq i}^{n-1} A[i,j].x[j])$$

- Start with a initial guess x_0 for the solution vector.

- This is used for the next approximation x_1.

Jacobi Iterative Method (Contd)

- A typical iteration in kth step is:

$$x_k[i] = \frac{1}{A[i,i]}(b[i] - \sum_{j \neq i}^{n-1} A[i,j].x_{k-1}[j])$$

- Process is converged when the residual $(b - A.x_k)$ becomes small.
- A Jacobi iteration can be written as

$$x_k[i] = \frac{r_{k-1}[i]}{A[i,i] + x_{k-1}[i]}$$

Code for Jacobi Method

```
procedure JACOBI_METHOD ($A$,$b$,$x$,$\epsilon$)
begin
    $k := 0$;
    Select initial solution vector $x_0$;
    $r_0 := b - Ax_0$;
    while ($\|r_k\|_2 > \epsilon \|r_0\|_2$) do
    begin
        $k := k + 1$;
        for $i := 0$ to $n - 1$ do
            $x_k[i] := r_{k-1}[i]/A[i,i] + x_{k-1}[i]$;   /* Equation 11.17 */
        $r_k := b - Ax_k$;
    endwhile;
    $x := x_k$;
end JACOBI_METHOD
```

Program 11.2 The serial Jacobi iterative method for solving a system of linear equations.

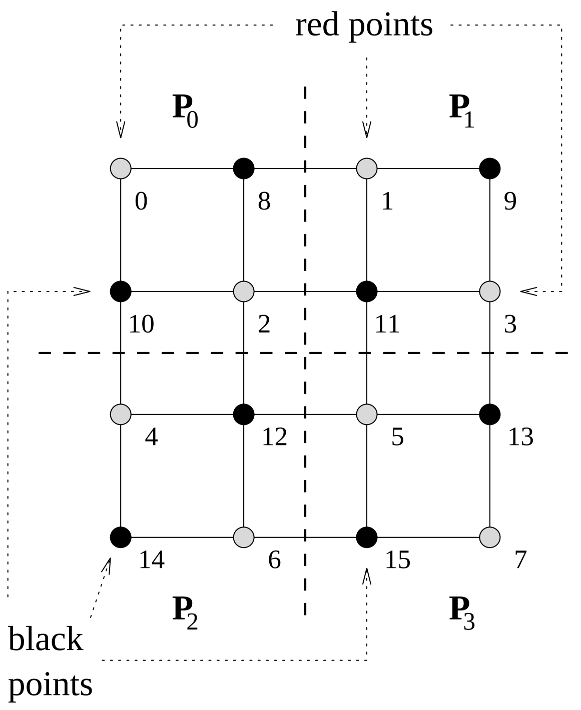
Parallel Jacobi Implementation

- Each iteration of Jacobi method performs three main computations: inner product of line 6, loop of lines 9 and 10, and matrix vector multiplication of line 11.
- Assign $A[i,i], r_k[i], x_k[i]$ to the same processor.
- Loop in line 9,10 can be executed in parallel without any communication.
- Other two steps require communication.
- Vector inner product takes $O(log\ (p))$ communication time on hypercube and $O(\sqrt{p})$ time on mesh.
- Matrix vector multiplication has been discussed earlier.

Red-Black Ordering

- The order in which the grid points in the discretized domain are numbered determines the order of the rows and columns in the coefficient matrix, and hence the location of non-zero elements.
- The degree of parallelism and rate of convergence depends on this ordering.
- Red-black ordering is a numbering scheme where every alternate grid point in each row or column is colored red, and the remaining points are colored black.
- Red points have black neighbors, and vice versa.
- In coefficient matrix, first n/2 rows (red points) have offdiagonal nonzero elements only in last n/2 columns (black points).

Red-Black Ordering



(a) A 4 X 4 grid with red-black ordering

$$
\begin{array}{c|cccccccccccccccc|c}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \\
\hline
0 & X & & & & & & & & X & & X & & & & & & P_0 \\
1 & & X & & & & & & & X & X & & X & & & & & P_1 \\
2 & & & X & & & & & & X & & X & X & X & & & & P_0 \\
3 & & & & X & & & & & & X & & X & & X & & & P_1 \\
4 & & & & & X & & & & & & X & & X & & X & & P_2 \\
5 & & & & & & X & & & & & & X & X & X & & X & P_3 \\
6 & & & & & & & X & & & & & & X & & X & X & P_2 \\
7 & & & & & & & & X & & & & & & X & & X & P_3 \\
8 & X & X & X & & & & & & X & & & & & & & & P_0 \\
9 & & X & & X & & & & & & X & & & & & & & P_1 \\
10 & X & & X & & X & & & & & & X & & & & & & P_0 \\
11 & & X & X & X & & X & & & & & & X & & & & & P_1 \\
12 & & & X & & X & X & X & & & & & & X & & & & P_2 \\
13 & & & & X & & X & & X & & & & & & X & & & P_3 \\
14 & & & & & X & & X & & & & & & & & X & & P_2 \\
15 & & & & & & X & X & X & & & & & & & & X & P_3 \\
\hline
\end{array}
$$

(b) Corresponding coefficient matrix

Summary

- Motivation for sparse solvers
- Basic Operations
- Iterative Methods for Sparse Linear Systems