CS475: Linear Equations
Gaussian Elimination
LU Decomposition

Wim Bohm

Colorado State University

Linear Equations

$$a_{00}\, x_0 + a_{0\,1}\, x_1 + \ldots + a_{0\,n-1}\, x_{n-1} = b_0$$

$$a_{10}\, x_0 + a_{1\,1}\, x_1 + \ldots + a_{1\,n-1}\, x_{n-1} = b_1$$

$$a_{20}\, x_0 + a_{2\,1}\, x_1 + \ldots + a_{2\,n-1}\, x_{n-1} = b_2$$

. . . . .

$$a_{n-1\,0}\, x_0 + a_{n-1\,1}\, x_1 + \ldots a_{n-1\;\;n-1}\, x_{n-1} = b_{n-1}$$

In matrix notation: Ax=b

matrix A, column vectors x and b

Solving Linear Equations: Gaussian Elimination

- **Reduce Ax=b into Ux=y**
  - U is an upper triangular
  - Diagonal elements Uii = 1

$$
\begin{aligned}
x_0 + u_{0\,1}\, x_1 + \ldots + u_{0\,n-1}\, x_{n-1} &= y_0 \\
x_1 + \ldots + u_{1\,n-1}\, x_{n-1} &= y_1 \\
\ldots\ldots\ldots \\
x_{n-1} &= y_{n-1}
\end{aligned}
$$

- **Back substitute**

Example

$$\begin{array}{ccccc} 2 & 4 & 2 & x_0 & = & 16 \\ 3 & 7 & 4 & x_1 & = & 29 \\ 2 & 5 & 4 & x_2 & = & 24 \end{array}$$

$$\begin{array}{ccccc} 1 & 2 & 1 & x_0 & = & 8 \\ 3 & 7 & 4 & x_1 & = & 29 \\ 2 & 5 & 4 & x_2 & = & 24 \end{array}$$

$$\begin{array}{ccccc} 1 & 2 & 1 & x_0 & = & 8 \\ 0 & 1 & 1 & x_1 & = & 5 \\ 0 & 1 & 2 & x_2 & = & 8 \end{array}$$

$$\begin{array}{ccccc} 1 & 2 & 1 & x_0 & = & 8 \\ 0 & 1 & 1 & x_1 & = & 5 \\ 0 & 0 & 1 & x_2 & = & 3 \end{array}$$

$$x_0 + 2\,x_1 + x_2 = 8$$
$$x_1 + x_2 = 5$$

$$x_0 = 1$$
$$x_1 = 2$$
$$x_2 = 3$$

Upper Triangularization - Sequential

- Two phases repeated n times
- Consider the k-th iteration ($0 \le k < n$)
- Phase 1: Normalize k-th row of A

```
for j = k+1 to n-1
    A_kj /= A_kk
y_k  =  b_k/A_kk
A_kk = 1
```

- Phase 2: Eliminate by * - row k

Using k-th row, make k-th column of A zero for row# > k

```
for i = k+1 to n-1
    for j = k+1 to n-1
        A_ij -= A_ik*A_kj
    b_i  -= a_ik * y_k
    Aik = 0
```

$O(n^2)$ divides, $O(n^3)$ subtracts and multiplies

Upper Triangularization – (Pipelined) Parallel

- p = n, row-striped partition

```
for all P_i  (i = 0 .. n-1) do in parallel
    for k = 0 to n-1
        if (i == k)
            perform k-th phase 1: normalize
            send normalized row k down
        if  (i > k)
            receive row k, send it down
            perform k-th phase 2: eliminate with row k
```

Example

$$\begin{matrix} 2 & 4 & 2 & & 16 \\ 3 & 7 & 4 & & 29 \\ 2 & 5 & 4 & & 24 \end{matrix}$$

$$\begin{matrix} 1 & 2 & 1 & & 8 \\ 3 & 7 & 4 & & 29 \\ 2 & 5 & 4 & & 24 \end{matrix} \quad \begin{matrix} - 3(1 & 2 & 1 & & 8) \\ - 2(1 & 2 & 1 & & 8) \end{matrix} \quad \begin{matrix} = 0 & 1 & 1 & & 5 \\ = 0 & 1 & 2 & & 8 \end{matrix}$$

$$\begin{matrix} 1 & 2 & 1 & & 8 \\ 0 & 1 & 1 & & 5 \\ 0 & 1 & 2 & & 8 \end{matrix} \quad - 1(0 \; 1 \; 1 \quad 5) = 0 \; 0 \; 1 \quad 3$$

$$\begin{matrix} x_0 = 1 \\ x_1 = 2 \\ x_2 = 3 \end{matrix}$$

Pivoting in Gaussian elimination

- What if $A_{kk} \sim 0$?
  - We get a big error, because we divide by A_{kk}
- Find a lower row e with largest A_{ek} and exchange rows
- What if all $A_{ek} \sim 0$?
  - Find a column with largest Ake and exchange columns
- This complicates the parallel algorithm
  - Need to keep track of the permutations

Back-substitute – Pipelined row striped

$$x_0 + u_{0\,1}\, x_1 + \ldots + u_{0\,n-1}\, x_{n-1} = y_0$$

$$x_1 + \ldots + u_{1\,n-1}\, x_{n-1} = y_1$$

$$\ldots\ldots\ldots$$

$$x_{n-1} = y_{n-1}$$

for k = n-1 down to 0

P_k: $x_k = y_k$; send x_k up

$P_{i\,(i<k)}$: send x_k up, $y_i = y_i - x_k * u_{ik}$

LU Decomposition

- In Gaussian elimination we transform

  $Ax=b$ into $Ux=y$

  i.e., we transform **both** A & b.
- So if we get a new system $Ax=c$, we need to do the whole transformation again.
- LU decomposition treats A independently:

  $$A = LU$$

  L **unit** lower triangular matrix

  U upper triangular matrix

Forward / backward substitution

- $Ax=b \quad \rightarrow \quad LUx = b$
- Define $Ux=y$, first solve $Ly=b$ then $Ux=y$
- Solving $Ly=b$ forward substitution:

  fwd substiture y_1:

  $$\begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{matrix} \begin{matrix} y_1 \\ y_2 \\ y_3 \end{matrix} = \begin{matrix} b_1 \\ b_2 \\ b_3 \end{matrix} \qquad \begin{matrix} 1 & 0 \\ 4 & 1 \end{matrix} \begin{matrix} y_2 \\ y_3 \end{matrix} = \begin{matrix} b_2 - 2b_1 \\ b_3 - 3b_1 \end{matrix}$$

  now fwd substitute y_2 and then y_3
- Solving $Ux=y$ backward substitution
  - same mechanism, studied for Gaussian elimination
- Both forward and backward easily pipelined

LU decomposition

- **Recursive approach**
  - n=1: done! $L=I_1$, $U = A$
  - n>1: $A = \left[\begin{array}{c|c} a_{11} & a_{12} \ldots a_{1n} \\ \hline a_{21} & a_{22} \ldots a_{2n} \\ . & \ldots \\ a_{n1} & a_{n2} \ldots a_{nn} \end{array}\right] = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix}$

    v and w: (n-1)-sized vectors
  - $A = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & w^T \\ 0 & A'-vw^T/a_{11} \end{bmatrix}$

You can check this by matrix multiplication

Do it by hand for n=2 and n=3

We have created the first step of the LU decomposition.

n=2

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a_{21}/a_{11} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ 0 & p \end{bmatrix}$$

$p = ?$

n=2

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a_{21}/a_{11} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ 0 & p \end{bmatrix}$$

$p = ?$

$a_{22} = (a_{21} * a_{12})/a_{11} + p$

n=2

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a_{21}/a_{11} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ 0 & p \end{bmatrix}$$

$p = ?$

$a_{22} = (a_{21} * a_{12})/a_{11} + p$

$p = a_{22} - (a_{21} * a_{12})/a_{11}$

LU decomposition: step 1

- **Recursive approach**
  - n=1: done! $L=I_1$, $U = A$
  - n>1: $A = \left[\begin{array}{c|ccc} a_{11} & a_{12} & \dots & a_{1n} \\ \hline a_{21} & a_{22} & \dots & a_{2n} \\ . & & \dots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{array}\right] = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix}$

    v and w: (n-1)-sized vectors
  - $A = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & w^T \\ 0 & A'-vw^T/a_{11} \end{bmatrix}$

You can check this by matrix multiplication

We have created the first step of the LU decomposition.

Inductive Leap

- Solve the rest recursively, ie $A' - vw^T / a_{11} = L' U'$

  Element wise this means: $A_{ij} -= A_{ik} * A_{kj}$

  Very similar to Gauss ($A_{kj} /= A_{kk}$ $A_{ij} -= A_{ik} * A_{kj}$)

  then $A = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11} & w^T \\ 0 & L' U' \end{bmatrix}$

  $= \begin{bmatrix} 1 & 0 \\ v/a_{11} & L' \end{bmatrix} \begin{bmatrix} a_{11} & w^T \\ 0 & U' \end{bmatrix}$

- In place: L and U stored and computed in A
  - 0-s and 1-s of L and U are implicit (not stored)

The code

```
for k = 1 to n  {
    for i = k+1 to n
        Aik /= Akk
    for i = k+1 to n
        for j = k+1 to n
            Aij -= Aik*Akj
}
```

1	4	5
2	10	16
3	20	42

k=1:
divide down by 1
then * -

1	4	5
2	2	6
3	8	27

k=2:
divide down by 2
then * -

1	4	5
2	2	6
3	4	3

CHECK!!!

how?

$$\begin{matrix} 1 & 4 & 5 \\ 2 & 2 & 6 \\ 3 & 4 & 3 \end{matrix} \longrightarrow \begin{matrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{matrix} \times \begin{matrix} 1 & 4 & 5 \\ 0 & 2 & 6 \\ 0 & 0 & 3 \end{matrix} = \begin{matrix} 1 & 4 & 5 \\ 2 & 10 & 16 \\ 3 & 20 & 42 \end{matrix}$$

Pivoting

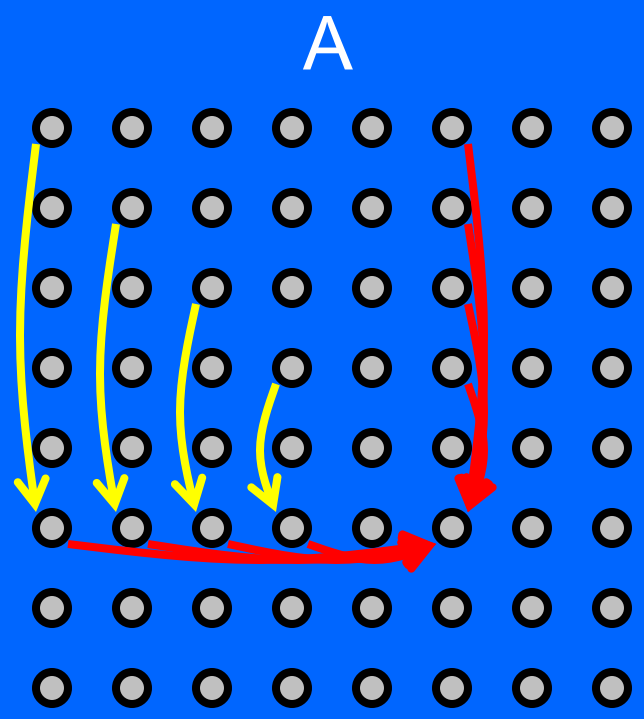
- Why?

  A_{kk} can be close to 0 causing big error in $1/A_{kk}$, so we should find the absolute largest A_{ik} and swap rows i an k

- How?
  - We can register this in a permutation matrix or better an n sized array $\pi[\, i \,]$: position of row i.
- No pivoting:
  - LU code is a straightforward translation of our recurrence.

LUD: data dependence

```
for k = 1 to n  {
  for i = k+1 to n
    Aik /= Akk
  for i = k+1 to n
    for j = k+1 to n
      Aij -= Aik*Akj
}
```



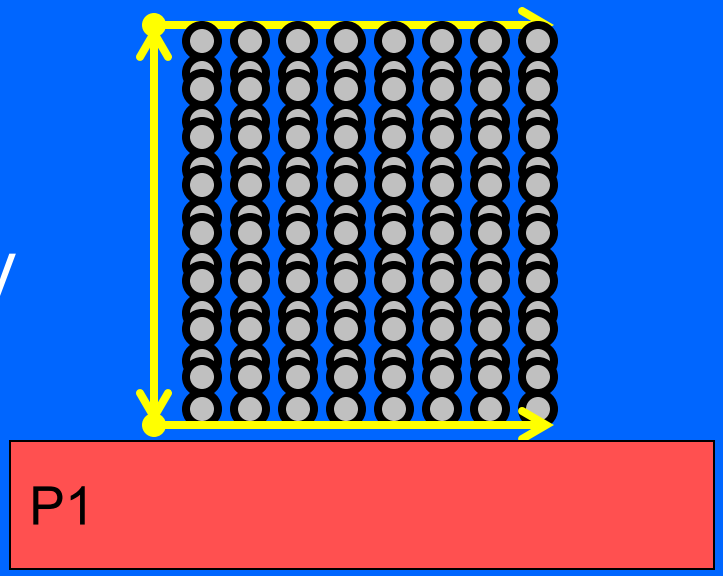
Pipelining

- Iteration k+1 can start after iteration k has finished row k+1
  - neither row k nor column k are read or written anymore
- This naturally leads to a a streaming algorithm
  - every iteration becomes a process
  - results from iteration/process k flow to process k+1
  - process k uses row k to update sub matrix A[k+1:n, k+1:n]
- Actually, only a fixed number P of processes run in parallel
  - after that, data is re-circulated

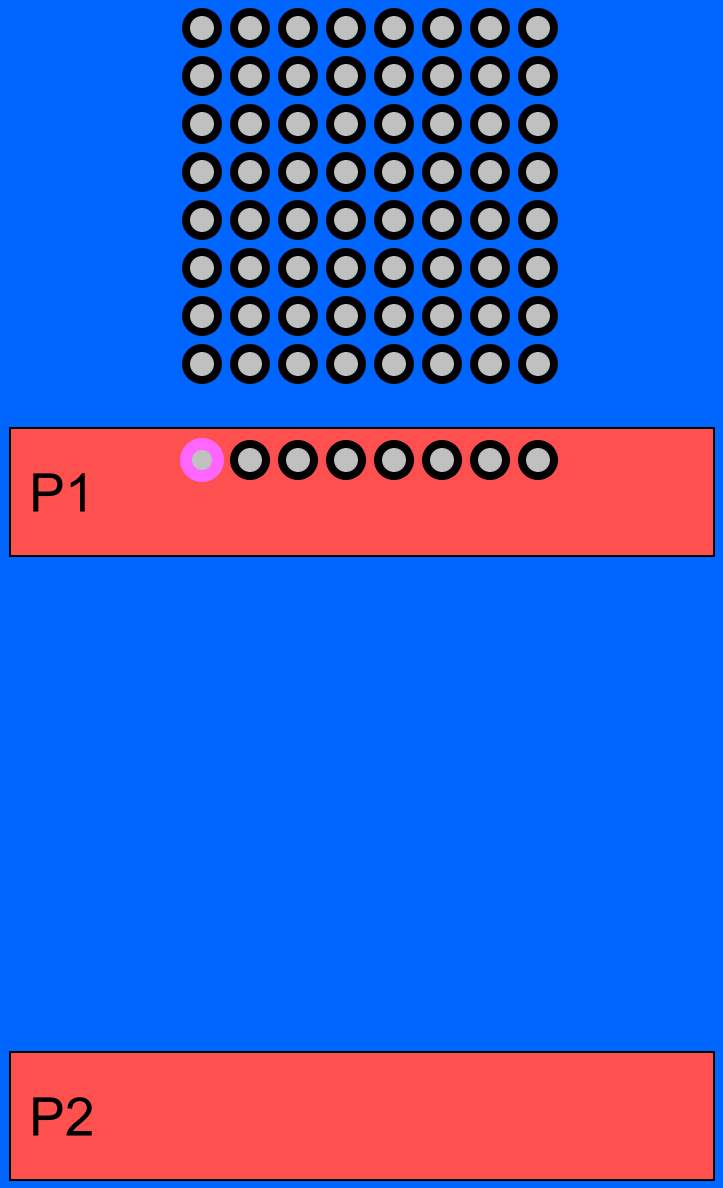
Pipelined LU

- Pipeline of P processes
  - parallel sections in OpenMP lingo
  - algorithm runs in ceil(N/P) sweeps
- In one sweep P outer k iterations run in parallel
- After each sweep P rows and columns are done
  - Problem size n is reduced to $(n-P)^2$
  - Rest of the (updated) array is re-circulated
- Each process owns a row
  - Process p owns row p + (i-1)P in sweep i, i=1 to ceil (N/P)
  - Array elements are updated while streaming through the pipe of processes
- Total work $O(n^3)$, speedup ~P
- Forward / backward substitution are simple loops

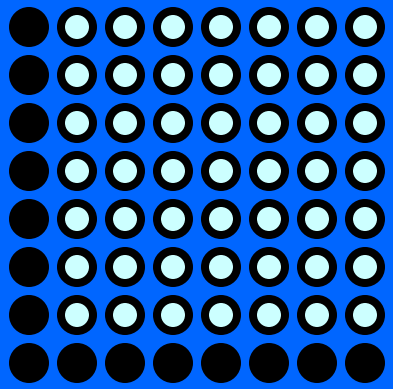
First, Flip the matrix vertically



Feed matrix to Processor 1

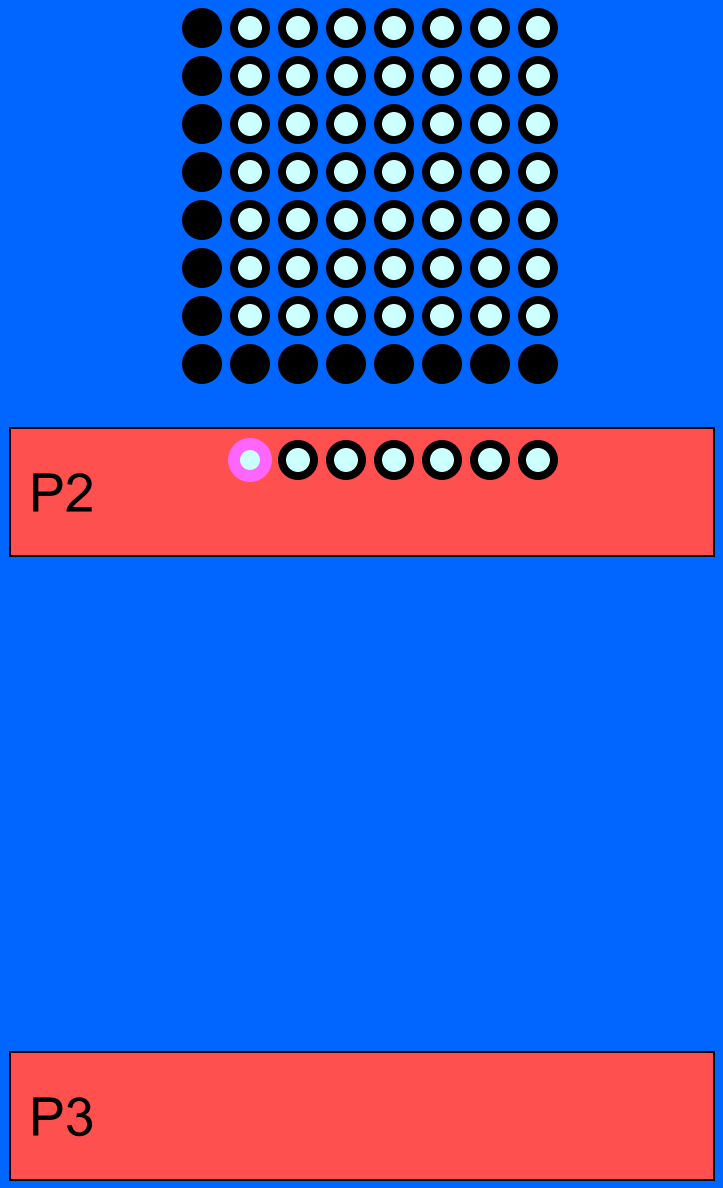


P1

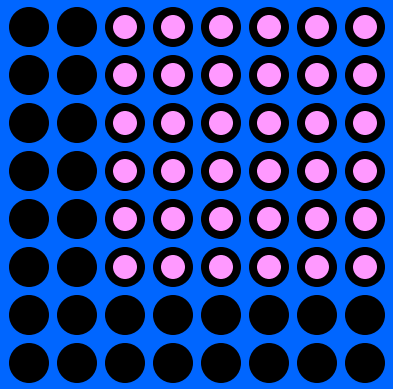


P2

Feed matrix to Processor 2

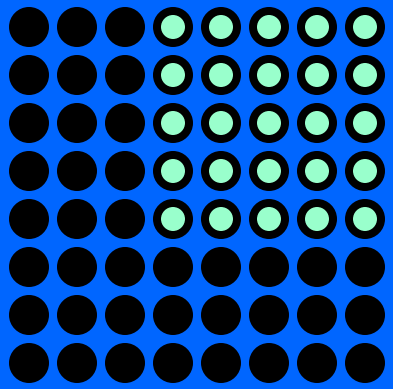


P2



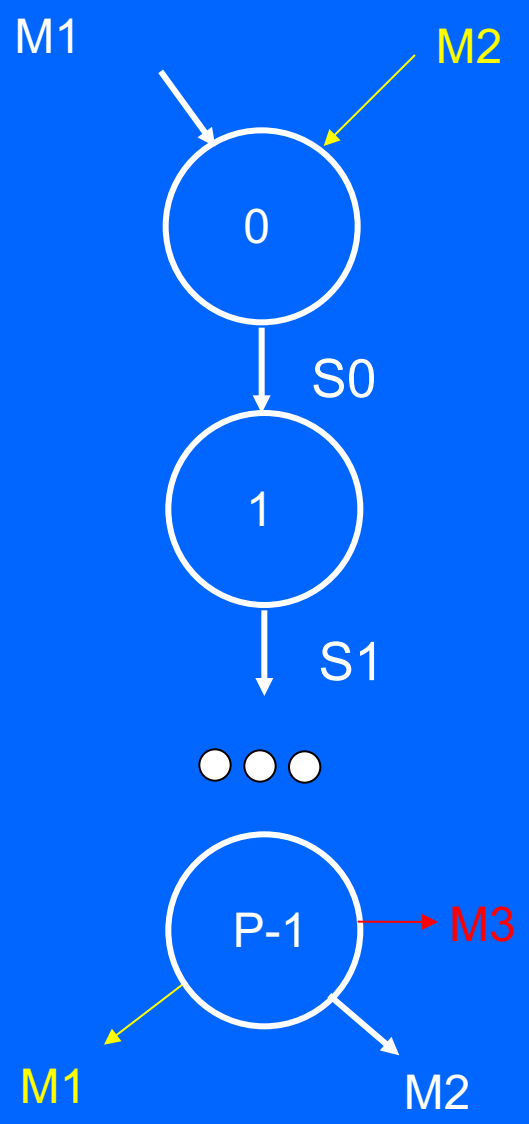
P3

P3



P4

Process Pipeline



Memories:

M1 -> M2 on even sweeps

M2 -> M1 on odd sweeps

M3 : siphoning off finished results

Processes are grouped in OpenMP style parallel sections with intermediate data in streams

Process 0 reads either from M1 or M2
writes to stream S0

Process i reads from stream Si-1
writes to Si

Process P-1
reads from stream Sp-1
writes finished (black) data to M3
writes rest to M1 or M2