Matrix Algs: M x = y

Matrix Vector Multiplication

Thursday, September 27, 2018 11:44 AM

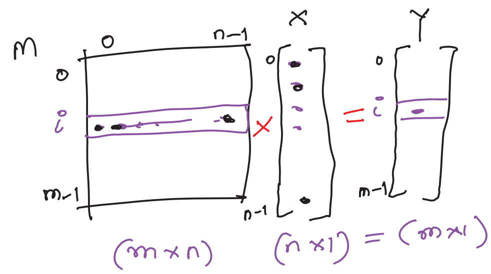
Input:

- $\mathbf{M}[m, n]$: an $m \times n$ matrix
- $\mathbf{x}[n, 1]$: a column vector with n values

Output:

- $\mathbf{y}[m, 1]$: a column vector with m values, such that: $\mathbf{M} \times \mathbf{x} = \mathbf{y}$

Serial Algorithm:



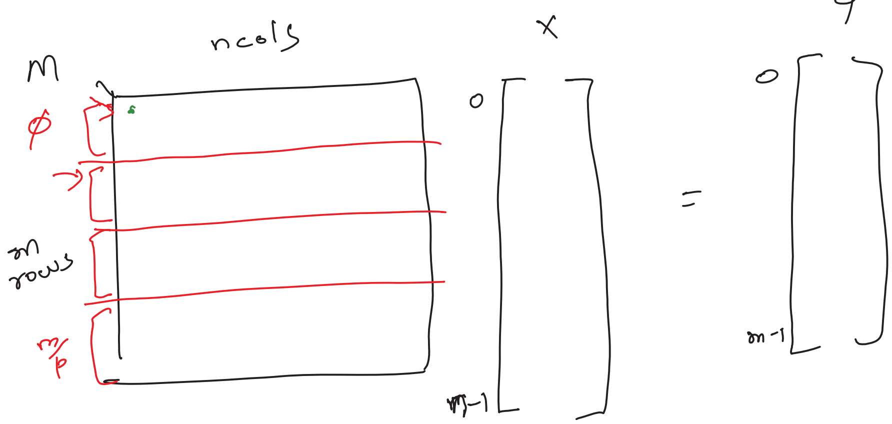
Data: $\mathbf{M}[m, n]$, $\mathbf{x}[n, 1]$
Result: $\mathbf{y}[m, 1]$
for i: 0 *to* $m - 1$ **do**
  $\mathbf{y}[i] = 0;$
  for j: 0 *to* $n - 1$ **do**
    $\mathbf{y}[i] += \mathbf{M}[i][j] \times \mathbf{x}[j];$
  end
end

O(mn) time

Q: How to distribute the input?

p procs.
$m \geq p$
$m \% p == 0$

$q \times p = m$
$q = \lceil \frac{m}{p} \rceil$



> Each proc gets $\left(\frac{m}{p} \times n\right)$ cells of M (Block decomp)

> Each proc stores entire x ⇒ n more cells (duplication)

Proc i starts reading from byte offset:

$$i \times \underbrace{\frac{mn}{p} \times \text{sizeof(int)}}_{\leq \text{ \# bytes to read per proc (or) EOF}}$$

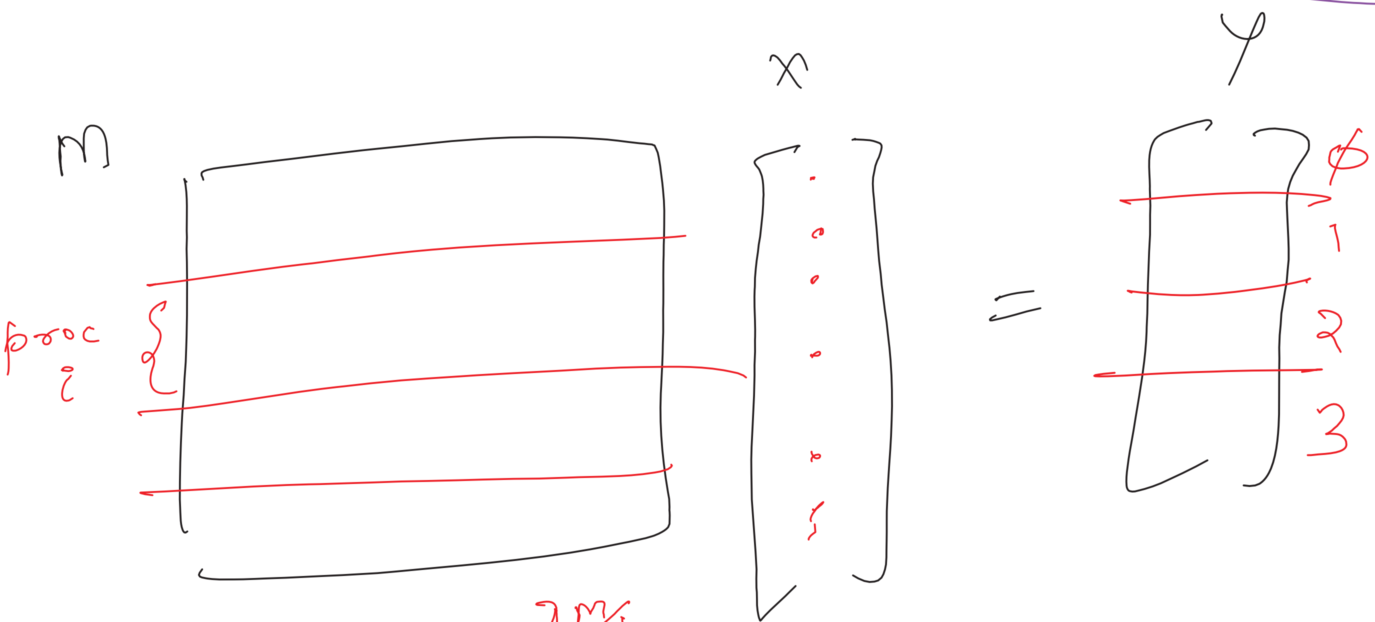
my rank

Approach A:

Matrix Algs: Mx=y Parallelization

Thursday, September 27, 2018 11:44 AM

Case: $m \geq n$ & $m \geq p$



$\frac{m}{p}$

Data: $\mathbf{M}[m, n]$, $\mathbf{x}[n, 1]$
Result: $\mathbf{y}[m, 1]$ — $y[\frac{m}{p}, 1]$
for i: 0 *to* $m - 1$ **do**
  $\mathbf{y}[i] = 0;$
  for j: 0 *to* $n - 1$ **do**
    $\mathbf{y}[i] += \mathbf{M}[i][j] \times \mathbf{x}[j];$
  end
end

$\Theta\left(\frac{mn}{p}\right)$ time

$y[i]_{local}$

$y_{local} = y[i \times \frac{m}{p} : \ldots : (i+1)\frac{m}{p} - 1]$

Runtime:	Steps	time
Assuming $m \geq p$	Input loading	
	Multiplication	$\Theta\left(\frac{mn}{p}\right)$ time
		$\Theta\left(\frac{mn}{p} + n\right)$

Q) what if $m < p$?

Space complexity: $\Theta\left(\frac{mn}{p} + n + \frac{m}{p}\right)$ — I/O time

↑ vs. ↑

Approach A:

Matrix Algs: Mx=y Parallelization

Thursday, September 27, 2018 11:44 AM

Parallel pseudocode for block decomposition of row (approach A)

Data: $\mathbf{M}[m,n]$, $\mathbf{x}[n,1]$; p processors
Result: $\mathbf{y}[m,1]$
Pre-condition:
$\mathbf{x}$ resides on each processor;
The rows of $\mathbf{M}$ are distributed evenly among p processors (i.e., $O(\frac{m}{p})$ rows per proc.). This is same as row-wise block partitioning. We will refer to the local copy of the matrix as $\mathbf{M}_{\text{local}}$.
Post-condition:
Rank i outputs $\mathbf{y}[i \times \frac{m}{p}, (i+1) \times \frac{m}{p} - 1]$. We will refer to the local copy of the output vector as $\mathbf{y}_{\text{local}}$.
Parallel algorithm:

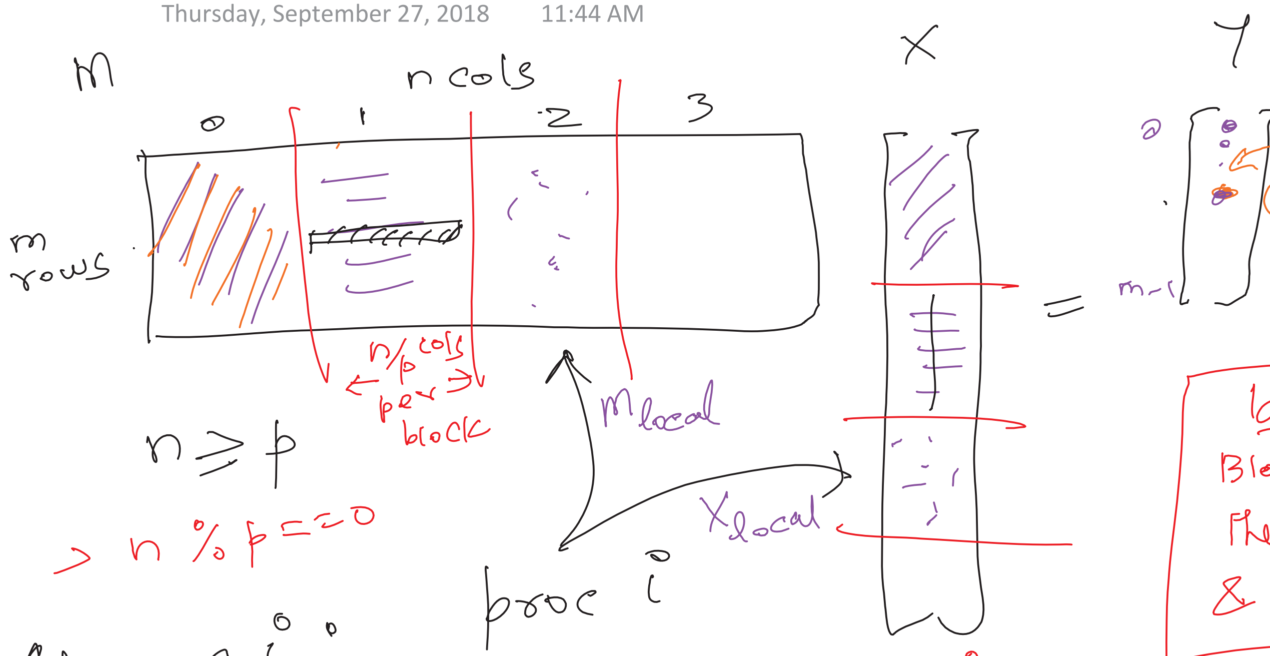
for i: 0 *to* $\frac{m}{p} - 1$ **do**
  $\mathbf{y}_{\text{local}}[i] = 0$;
  for j: 0 *to* $n - 1$ **do**
    $\mathbf{y}_{\text{local}}[i] += \mathbf{M}_{\text{local}}[i][j] \times \mathbf{x}[j]$;
  end
end
Output $\mathbf{y}_{\text{local}}$.

Approach B:

Matrix Algs: Mx=y Parallelization

Thursday, September 27, 2018 11:44 AM

Case: $m < n$ & $m \le p$



$n \geq p$

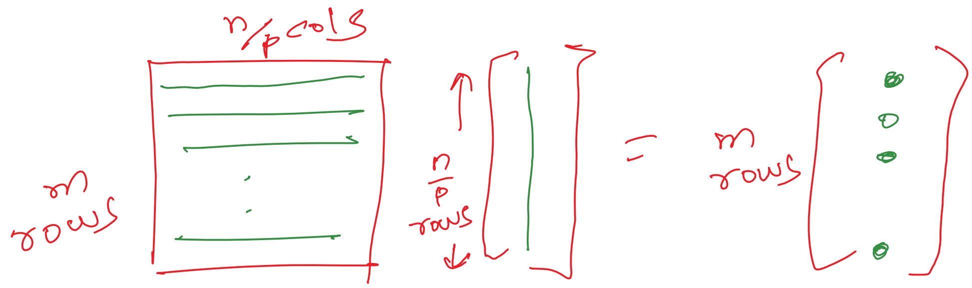
→ $n \% p == 0$

Idea: Block decompose the columns of M & X

At proc i:

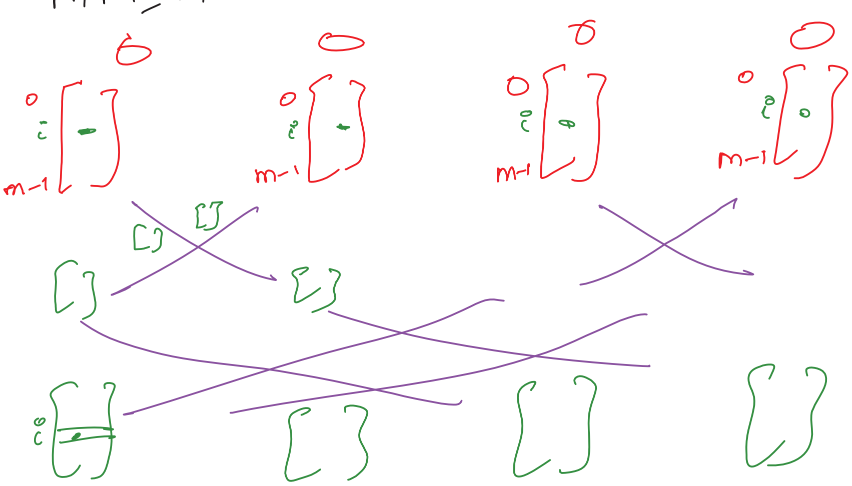
$$M^i_{local} \times X^i_{local} = Y^i_{local}$$

$$Y[i] = \sum_{k=0}^{p-1} Y^k_{local}[i]$$



$\frac{m}{p} \times n$?

MPI_AllReduce (using vectors):



Comm. cost:

$(\lg p)$ timesteps

$$(\tau + \mu m) \lg p$$

Total time

size of vector