Parallel Gradient Descent Algorithm with *MPI* on *Distributed Architecture*

Dr. Süha Tuna

FSMVU – Department of Computer Engineering

Content

1. Problem Definition
2. Gradient Descent Algorithm
3. Thinking Parallel
4. Implementation Details

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Problem Definition

- Solving an underdetermined linear equation system
- $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$
- $\boldsymbol{A}$ and $\boldsymbol{b}$ is given, $\boldsymbol{x}$ is unknown
- A is size of $n_1 \times n_2$ where $n_2 > n_1$
  - # of unknowns > # of equations
- No unique solution!
- Conventional iterative solvers may not work!
  - Rectangularity

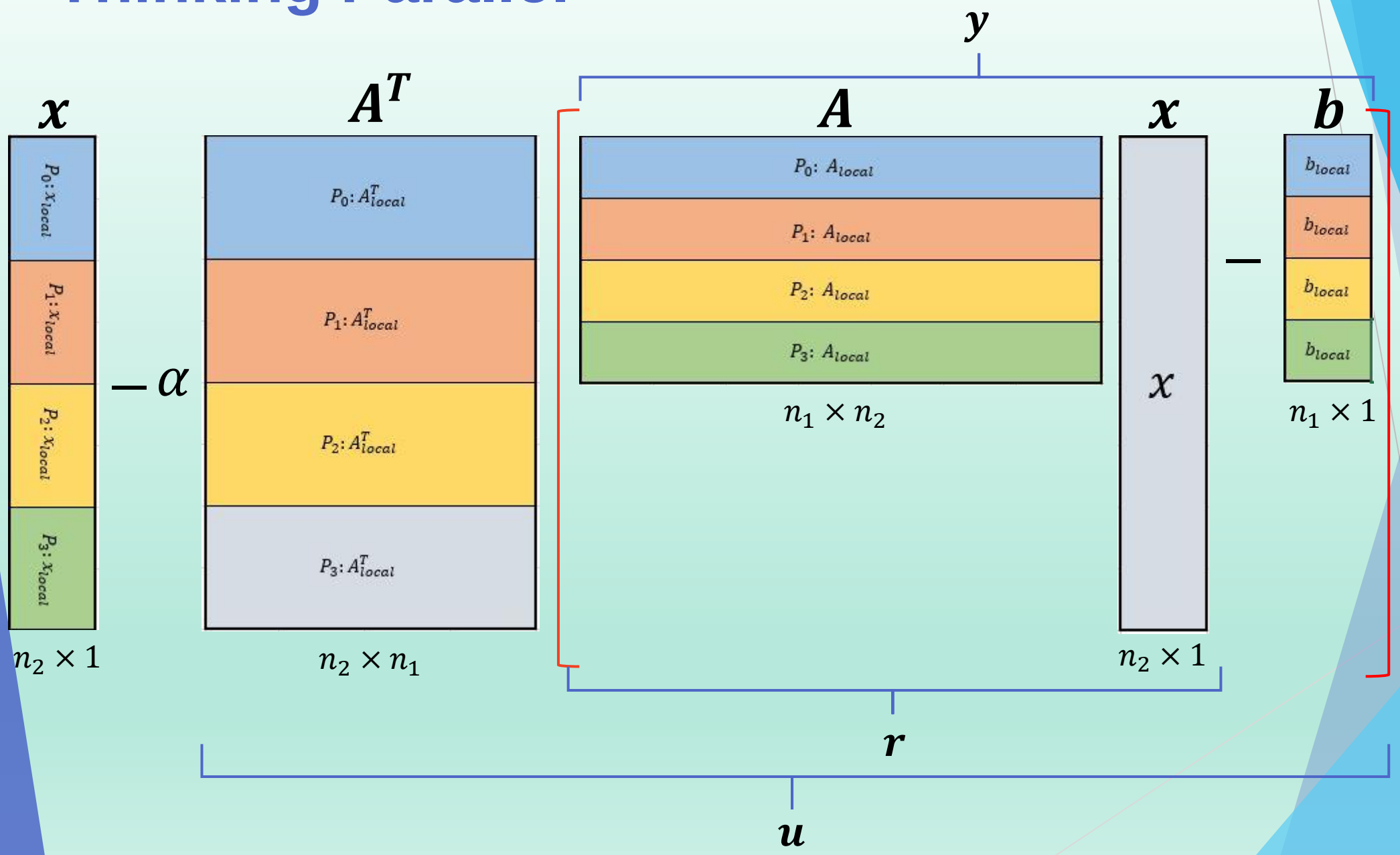
FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Gradient Descent Algorithm

- Minimize $E(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b}||_2^2$
  - Convex cost function
- $\nabla_{\boldsymbol{x}} E(\boldsymbol{x}) = \boldsymbol{A}^T(\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b})$
- Initial guess: $\boldsymbol{x}_0$
- Learning rate: α
- GD iterations: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla_{\boldsymbol{x}} E(\boldsymbol{x}_k), \quad k = 0, 1, 2, \ldots$
- Converges to global minimum, if
  - Cost is convex
- α should be examined and chosen carefully!

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Thinking Parallel



# of procs: p

$$chunk_1 = \frac{n_1}{p}$$

$$chunk_2 = \frac{n_2}{p}$$

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

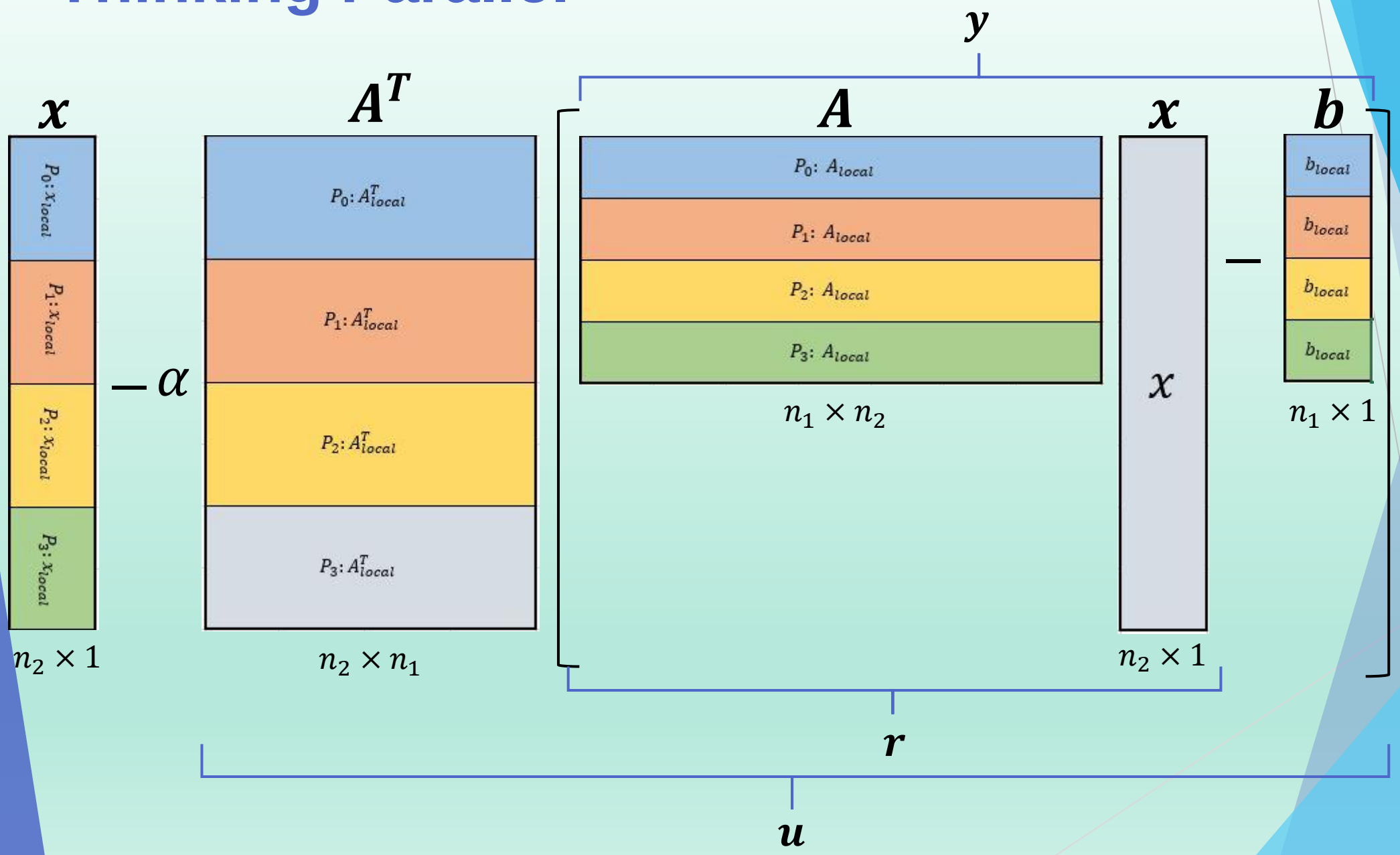
Parallel Implementation - I

- Main iteration:
  - $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \boldsymbol{A}^T (\boldsymbol{A}\,\boldsymbol{x}_k - \boldsymbol{b})$

1. Matrix vector multiplication: $\boldsymbol{A}\,\boldsymbol{x} = \mathbf{r}$
   1. Start with an initial guess: $\boldsymbol{x}_0$
   2. `MPI_Scatter` rows of $\boldsymbol{A}$ matrix to each process' $\boldsymbol{A}_{local}$
      1. Can use derived data type: `MPI_Type_contiguous`
   3. `MPI_Broadcast` $\boldsymbol{x}$ to each process
   4. Perform $\boldsymbol{A}_{local} \times \boldsymbol{x} = \boldsymbol{r}_{\boldsymbol{local}}$
   5. No need to `MPI_Gather` for $\boldsymbol{r}_{local}$

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Thinking Parallel



# of procs: p

$$chunk_1 = \frac{n_1}{p}$$

$$chunk_2 = \frac{n_2}{p}$$

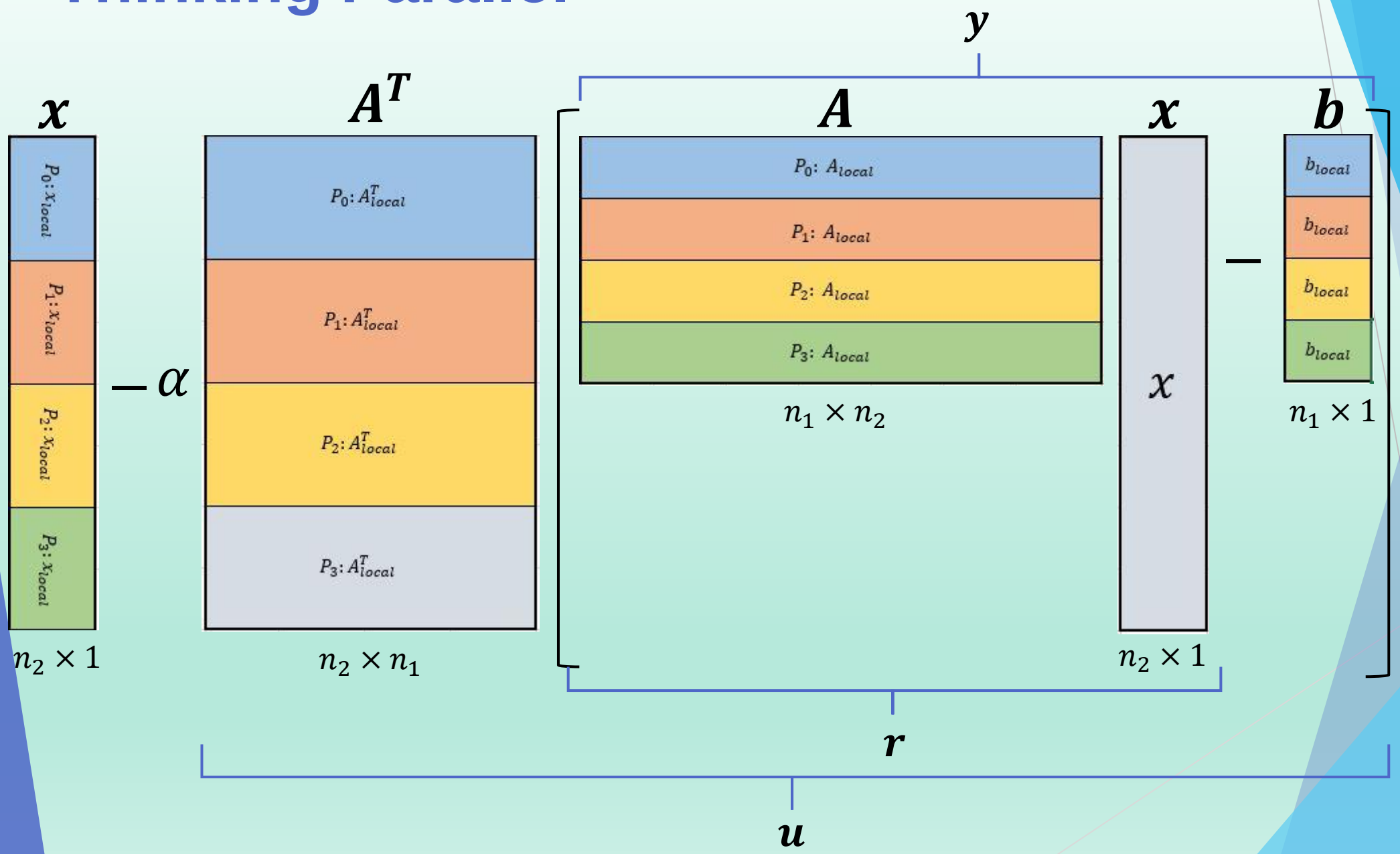
FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Parallel Implementation - II

1. Vector subtraction: $\boldsymbol{r} - \boldsymbol{b} = \mathbf{y}$
    1. $\boldsymbol{r}_{local}$ is already calculated
    2. MPI_Scatter $\boldsymbol{b}$ vector to each process' $\boldsymbol{b}_{local}$
    3. Perform local vector subtraction $\boldsymbol{r}_{local} - \boldsymbol{b}_{local} = \boldsymbol{y}_{local}$

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Thinking Parallel



# of procs: p

$$chunk_1 = \frac{n_1}{p}$$

$$chunk_2 = \frac{n_2}{p}$$

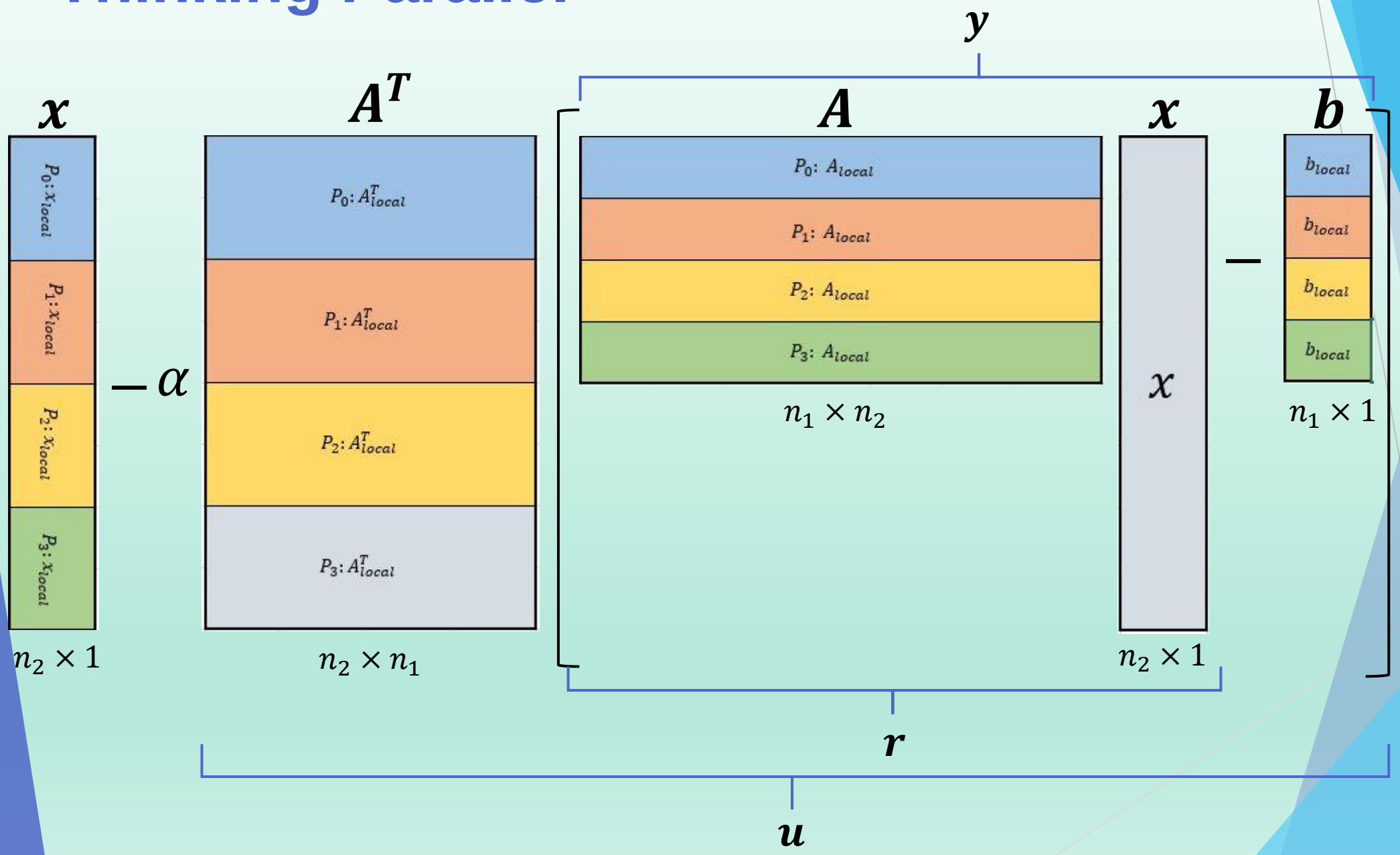
FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Parallel Implementation - III

1. Matrix vector multiplication: $\boldsymbol{A}^T \boldsymbol{y} = \boldsymbol{u}$
    1. `MPI_Scatter` columns of $\boldsymbol{A}$ matrix to each process' $\boldsymbol{A}^T_{local}$
        1. Can use derived data type: `MPI_Type_vector`
        2. Do not forget to use `MPI_Type_create_resized`
    2. `MPI_Allgather` $\boldsymbol{y}_{local}$'s from each process to $\boldsymbol{y}$ vector
    3. Perform $\boldsymbol{A}^T_{local} \times \boldsymbol{y} = \boldsymbol{u}_{local}$
    4. No need to `MPI_Gather` for $\boldsymbol{u}_{local}$

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Thinking Parallel



# of procs: p

$$chunk_1 = \frac{n_1}{p}$$

$$chunk_2 = \frac{n_2}{p}$$

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

Parallel Implementation - IV

1. Scalar multiplication: $\alpha\, u_{local}$
2. `MPI_Scatter` $\boldsymbol{x}$ to each process' $\boldsymbol{x}_{local}$
3. Vector subtraction: $\boldsymbol{x} \coloneqq \boldsymbol{x} - \alpha\, \boldsymbol{u}$
    1. Perform local subtraction: $\boldsymbol{x}_{local} \coloneqq \boldsymbol{x}_{local} - \alpha\, \boldsymbol{u}_{local}$
4. Repeat steps 1 to 4 till convergence or the last iteration

Implement the algorithm using MPI in C.

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010