Matrix Vector Multiplication With Block-Checkerboard Partitioning

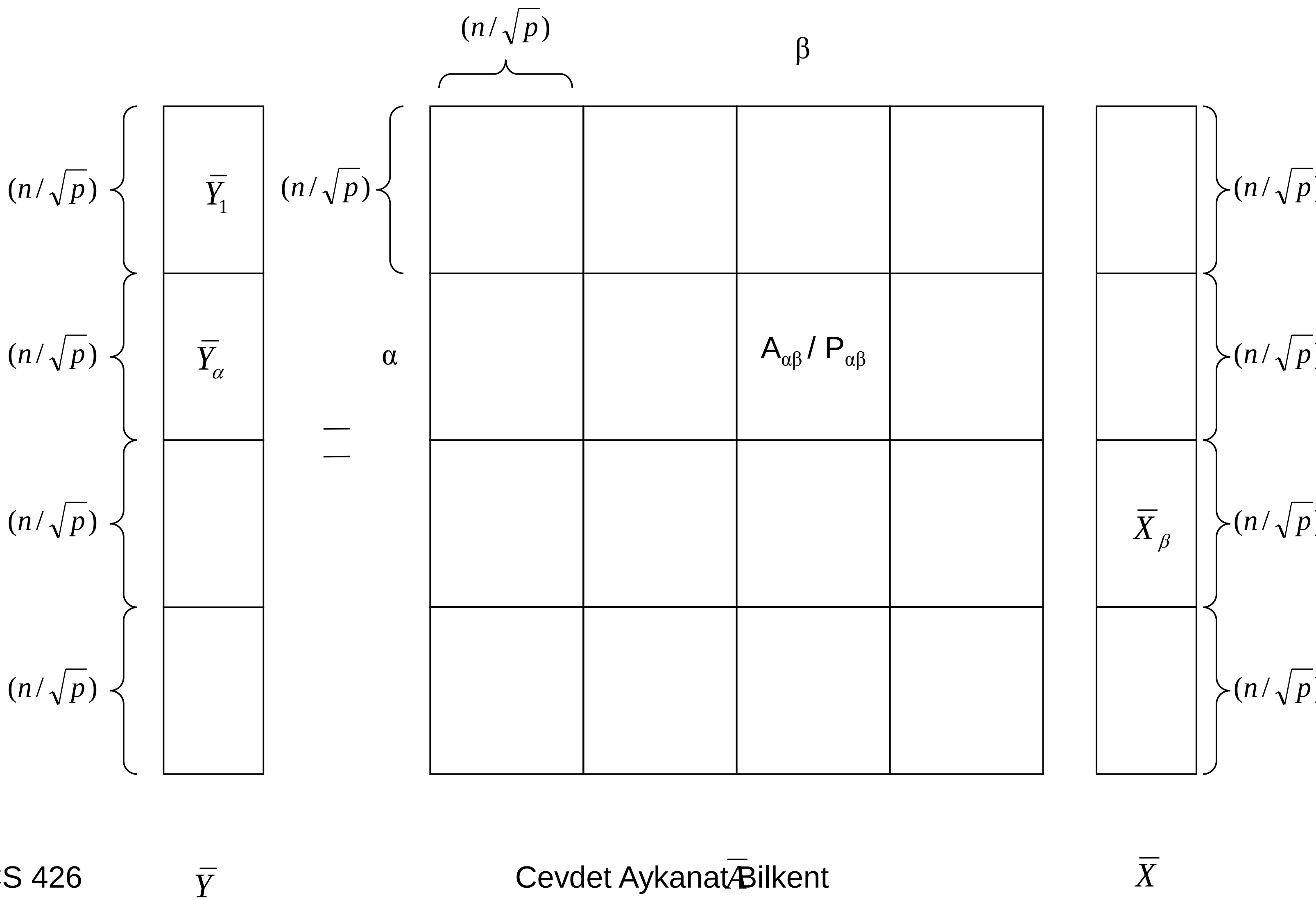
Coarse Grain Algorithm

- n x n matrix A mapped onto a $\sqrt{p}$ x $\sqrt{p}$ mesh of p< n^2 processors
- matrix A decomposed onto square blocks of size $(n/\sqrt{p})$ x $(n/\sqrt{p})$
- each block assigned to an individual processor
  - α = 0, 1, … $\sqrt{p}-1$ index rows
  - β = 0, 1, … $\sqrt{p}-1$ index columns
- $A_{\alpha\beta}$ = (α, β) is a subblock of A of size $(n/\sqrt{p})$x $(n/\sqrt{p})$

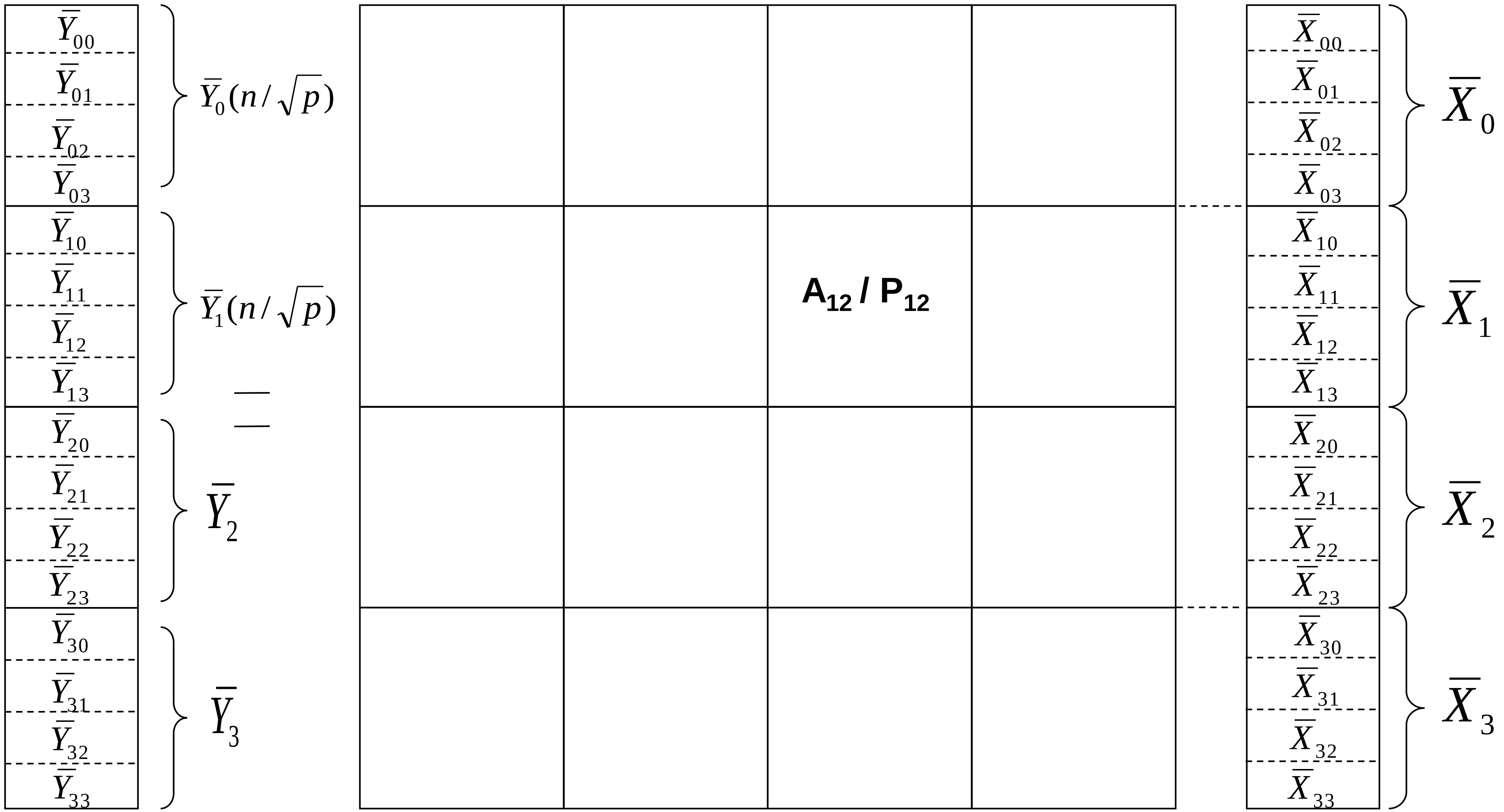
C Algorithm

- X and Y are also conceptually divided into pieces, each of size;
  - X_β : β^{th} slice of the X vector
  - Y_α : α^{th} slice of the Y vector

Coarse Grain Algorithm



Coarse Grain Algorithm



Coarse Grain Algorithm

- $\bar{Y}_\alpha$ and $\bar{X}_\beta$ subvectors are also divided into pieces, each of size n / p
  - $\bar{Y}_{\alpha\beta}$: β[th] subvector of the $\bar{Y}_\alpha$ – subvector for $\beta = 0,1,\ldots\sqrt{p}-1$
  - $\bar{X}_{\beta\alpha}$: α[th] subvector of the $\bar{X}_\beta$ – subvector for $\alpha = 0,1,\ldots\sqrt{p}-1$

- processor $P_{\alpha\beta}$ initially stores $A_{\alpha\beta}$ and $\bar{X}_{\beta\alpha}$

- processor $P_{\alpha\beta}$ must know $\bar{Y}_{\beta\alpha}$ at the end of the operation

Communication Primitives

<u>FOLD</u> operation (Multinode Accumulation)

- each processor q has a local vector $\overline{X}_q$ of size n
- addition $\overline{S} = \sum_{q=0}^{p-1} \overline{X}_q$ of these P vectors so that:
  - processor P_i gets only the q-th slice of $\overline{S}_q$ of the resultant vector S
  - $\overline{S}_i = \left[S_{(n/p)i}, S_{(n/p)i+1}, \ldots, S_{(n/p)(i-1)-1}\right]^+$

Communication Primitives

- processor $P_{\alpha\beta}$ must know $\overline{X}_\beta$ to compute its contribution to $\overline{Y}_\alpha$
  - i.e., $\overline{Y}_\alpha \leftarrow \sum_{\beta=0}^{\sqrt{p}-1} \overline{Y}_\alpha^{\ \beta}$
  - $\overline{Y}_\alpha^{\ \beta} = A_{\alpha\beta}\overline{X}_\beta$ where $\overline{Y}_\alpha^{\ \beta}$ is a vector of size $n/\sqrt{p}$ thus, $\overline{Y}_\alpha = \sum_{\beta=0}^{\sqrt{p}-1} \overline{Y}_\alpha^{\ \beta}$

  vector summation over all processors sharing row α

4-Phase Parallel Algorithm

1. Perform $\sqrt{p}$ concurrent **AABC** operations among $\sqrt{p}$ processors sharing each column

- each processor $P_{\alpha\beta}$ on column β for α = 0,1,…, $\sqrt{p}-1$

$$\bar{X}_{\beta} \leftarrow AABC(\bar{X}_{\beta\alpha})$$

2. Each processor $P_{\alpha\beta}$ computes: $\bar{Y}_{\alpha}^{\beta} \leftarrow A_{\alpha\beta}\bar{X}_{\beta}$

4-Phase Parallel Algorithm

3. Perform $\sqrt{p}$ concurrent FOLD operations among $\sqrt{p}$ processors sharing each row

   - fold on row α : $\bar{Y}_{\alpha} \leftarrow \sum_{\beta=0}^{\sqrt{p}-1} \bar{Y}_{\alpha}^{\;\beta}$ is computed such that
     - each processor $P_{\alpha\beta}$ of row α gets $\bar{Y}_{\alpha\beta}$ at the end for β = 0,1,…, $\sqrt{p}-1$

4. transpose $\bar{Y}_{\alpha\beta}$'s with $\bar{Y}_{\beta\alpha}$'s
   - processor $P_{\alpha\beta}$: SENDS $\bar{Y}_{\alpha\beta}$ to processor $P_{\beta\alpha}$
     RECEIVES $\bar{Y}_{\beta\alpha}$ from processor $P_{\beta\alpha}$