Parallel Programming
in C with MPI and OpenMP

Michael J. Quinn

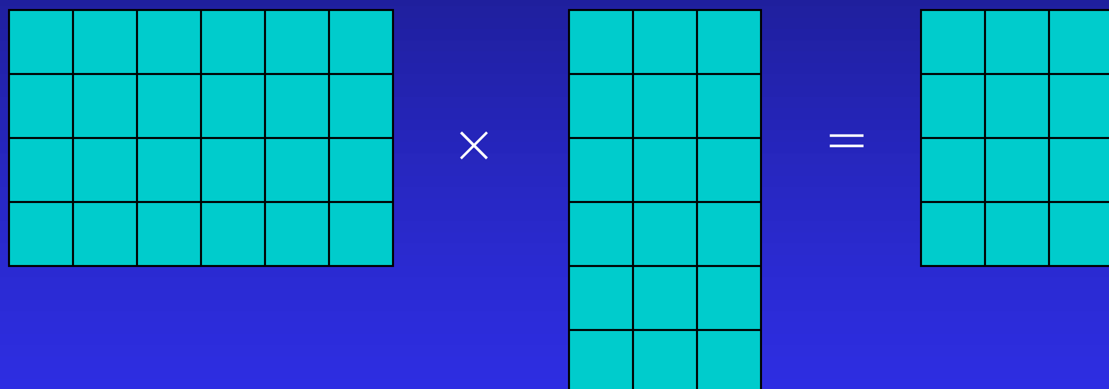
Mc Graw Hill

Chapter 11

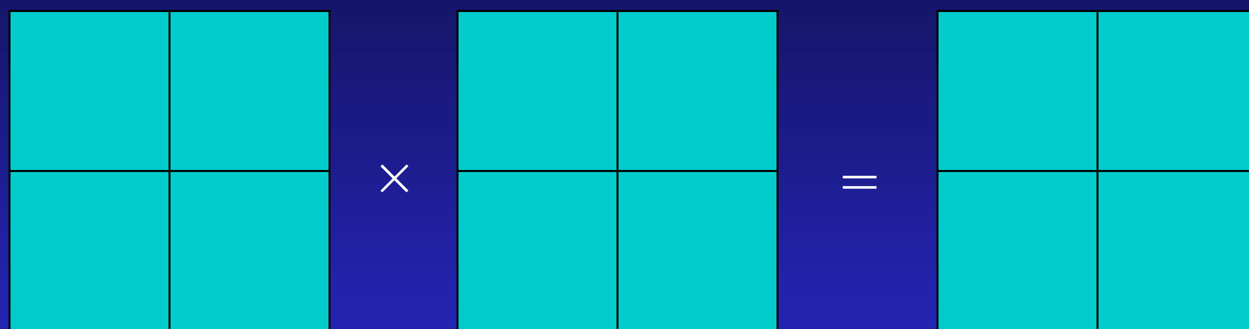
Matrix Multiplication

Iterative, Row-oriented Algorithm

Series of inner product (dot product) operations



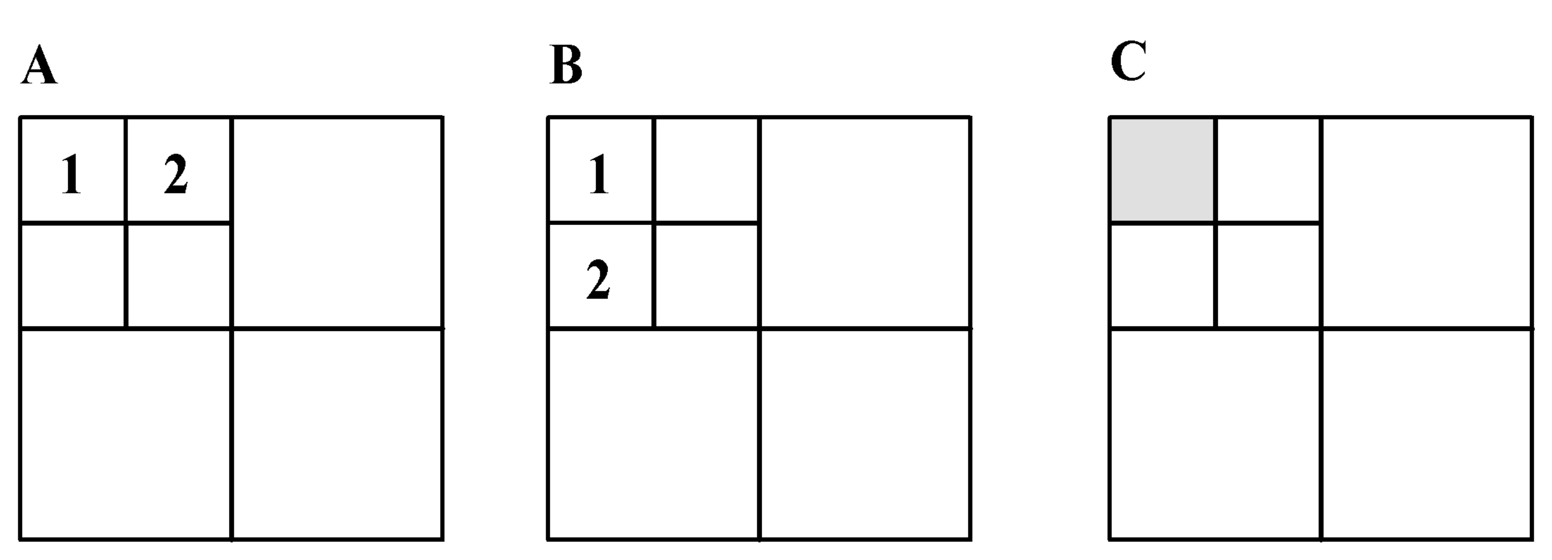
Block Matrix Multiplication



Replace scalar multiplication
with matrix multiplication

Replace scalar addition with matrix addition

Recurse Until B Small Enough



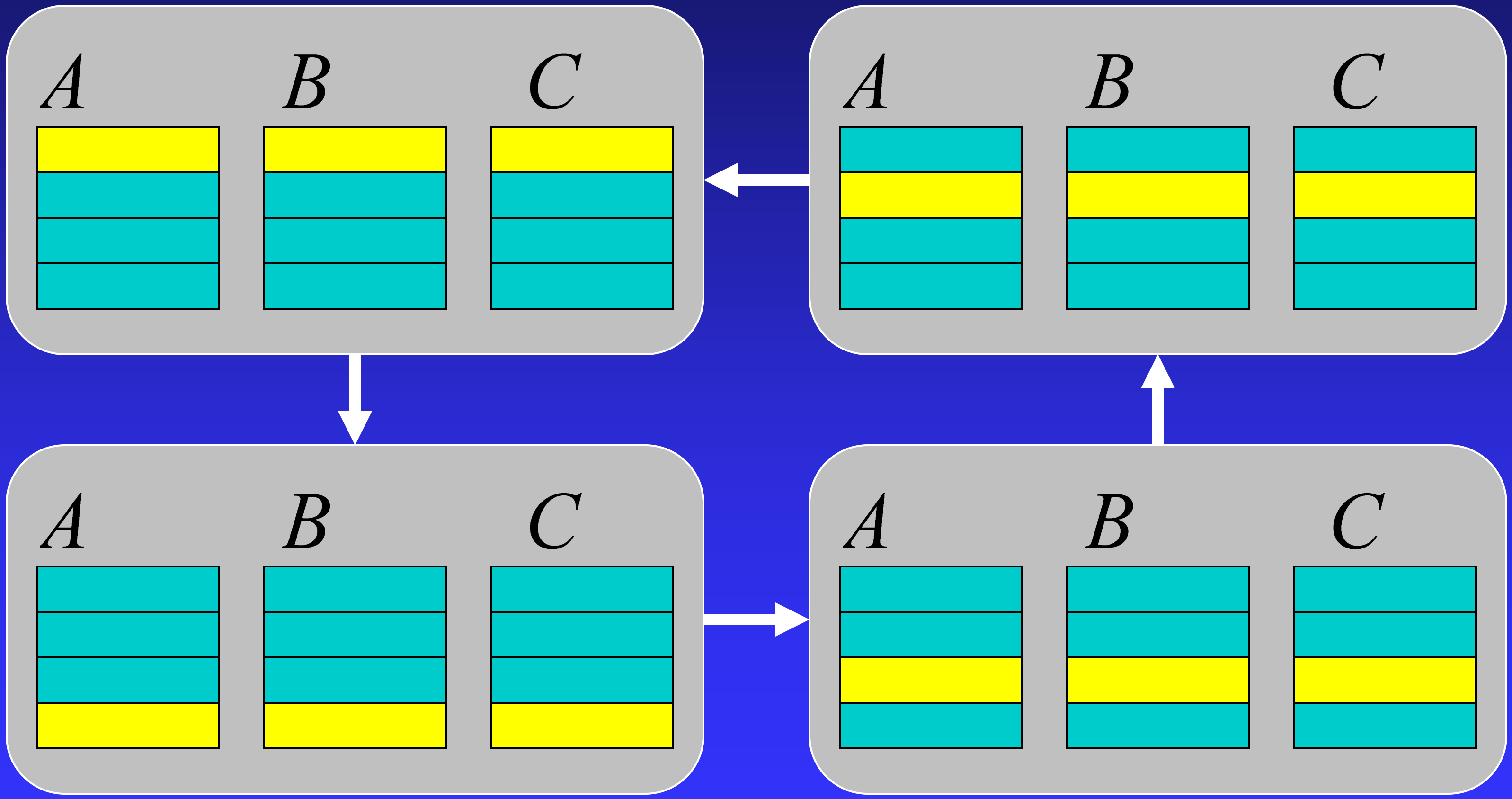
First Parallel Algorithm

- Partitioning
  - Divide matrices into rows
  - Each primitive task has corresponding rows of three matrices
- Communication
  - Each task must eventually see every row of B
  - Organize tasks into a ring

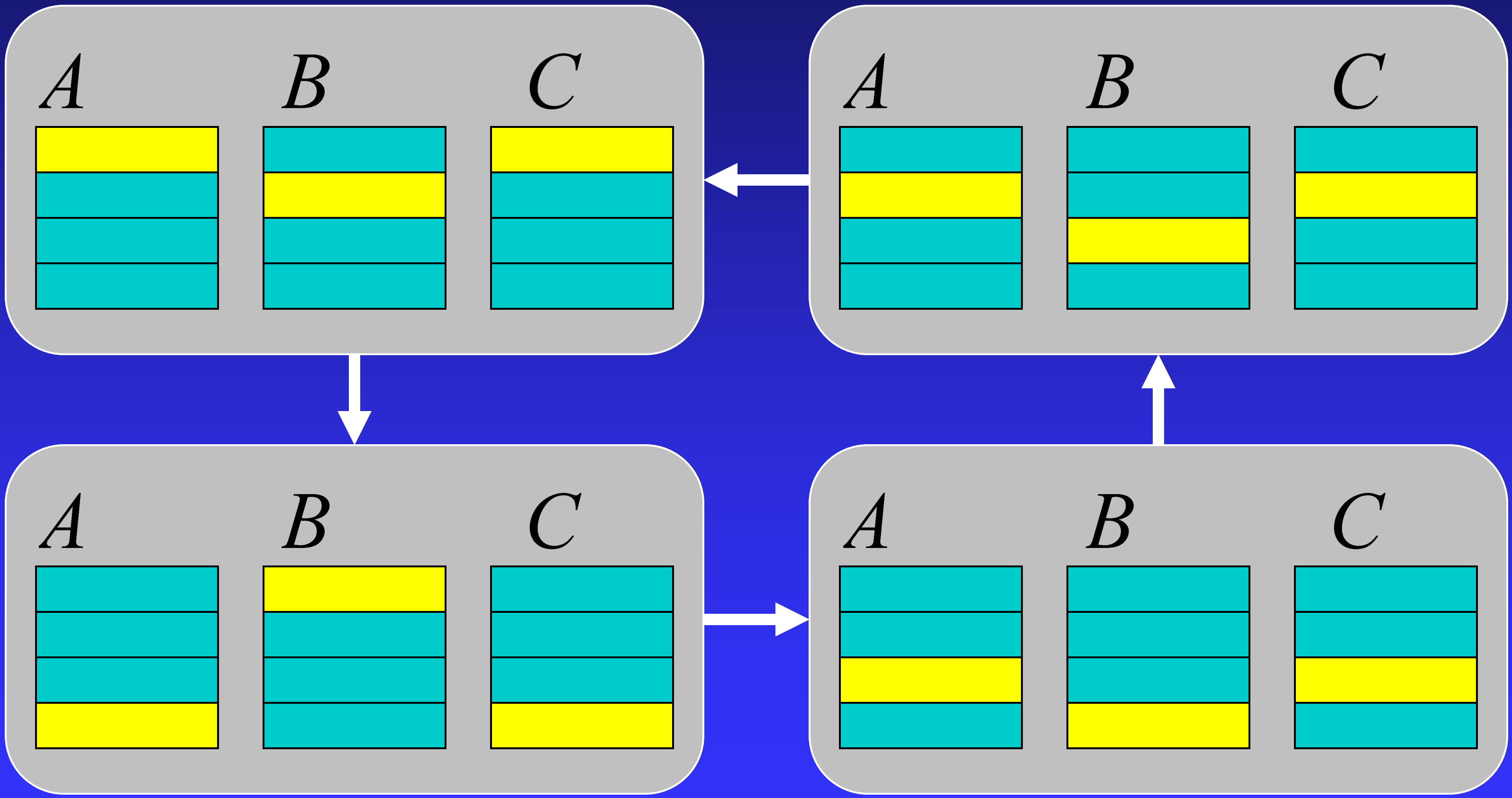
First Parallel Algorithm (cont.)

- Agglomeration and mapping
  - Fixed number of tasks, each requiring same amount of computation
  - Regular communication among tasks
  - Strategy: Assign each process a contiguous group of rows

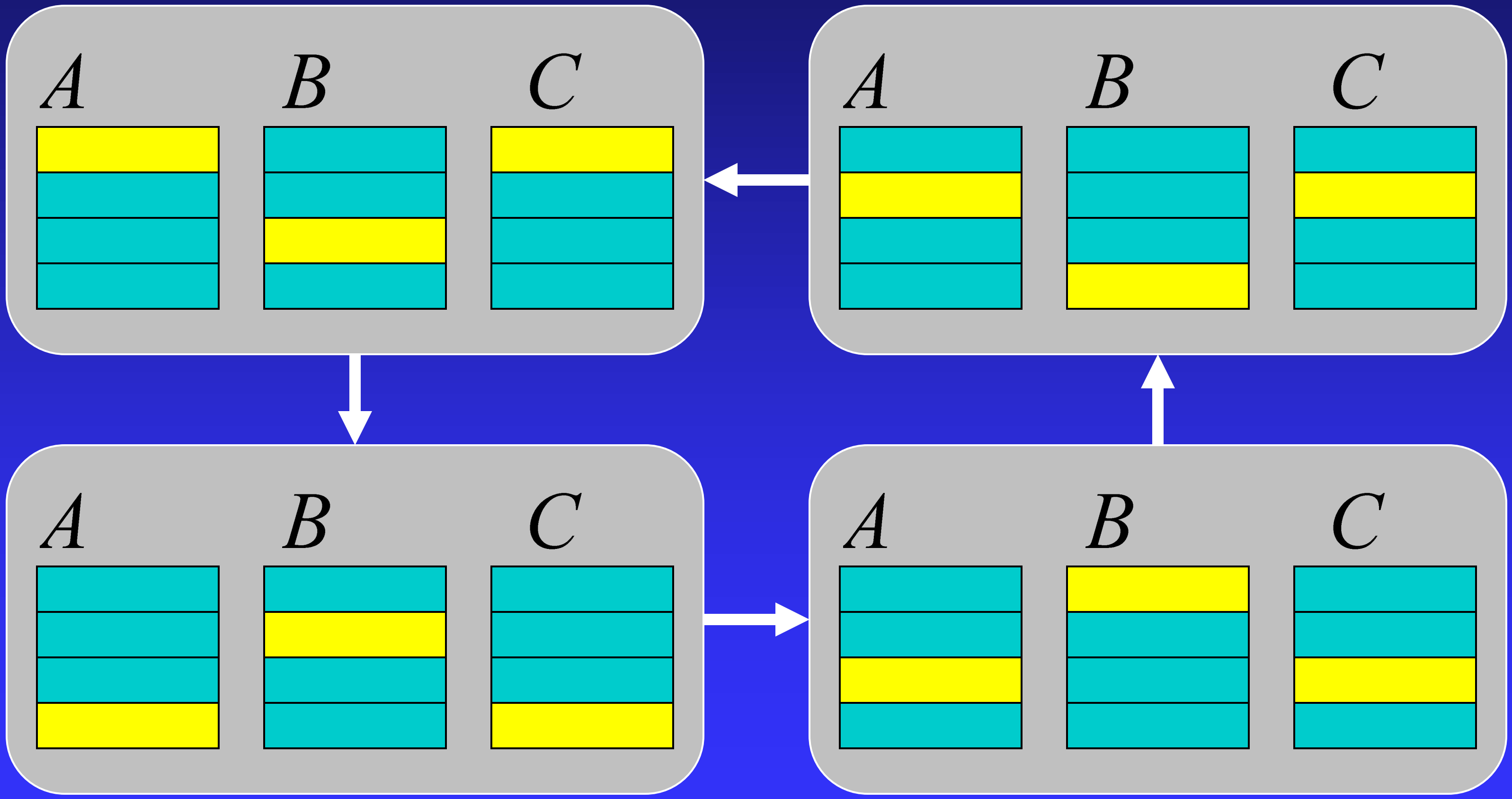
Communication of B



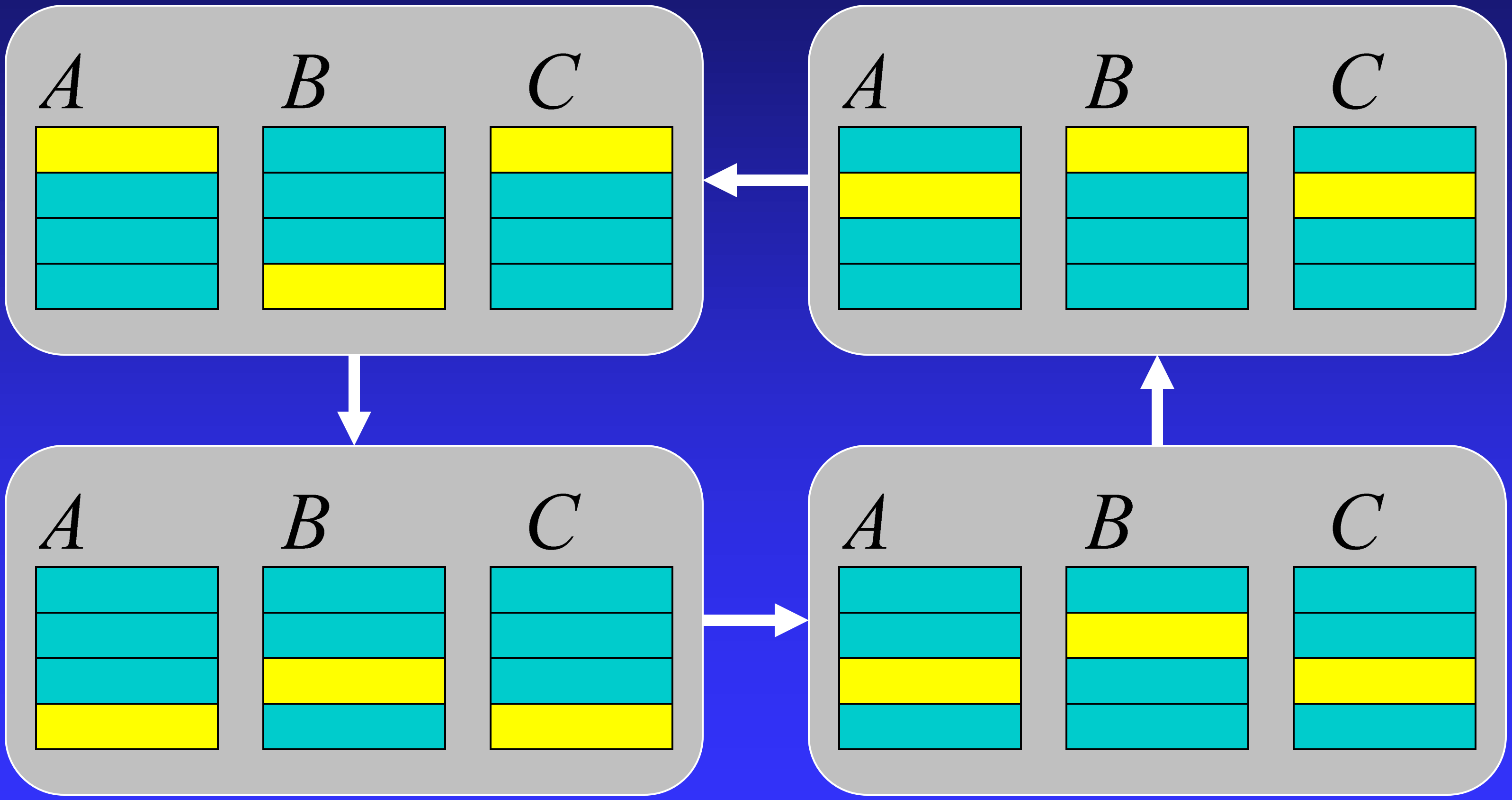
Communication of B



Communication of B



Communication of B



Complexity Analysis

- Algorithm has p iterations
- During each iteration a process multiplies $(n / p) \times (n / p)$ block of A by $(n / p) \times n$ block of B: $\Theta(n^3 / p^2)$
- Total computation time: $\Theta(n^3 / p)$
- Each process ends up passing $(p-1)n^2/p = \Theta(n^2)$ elements of B

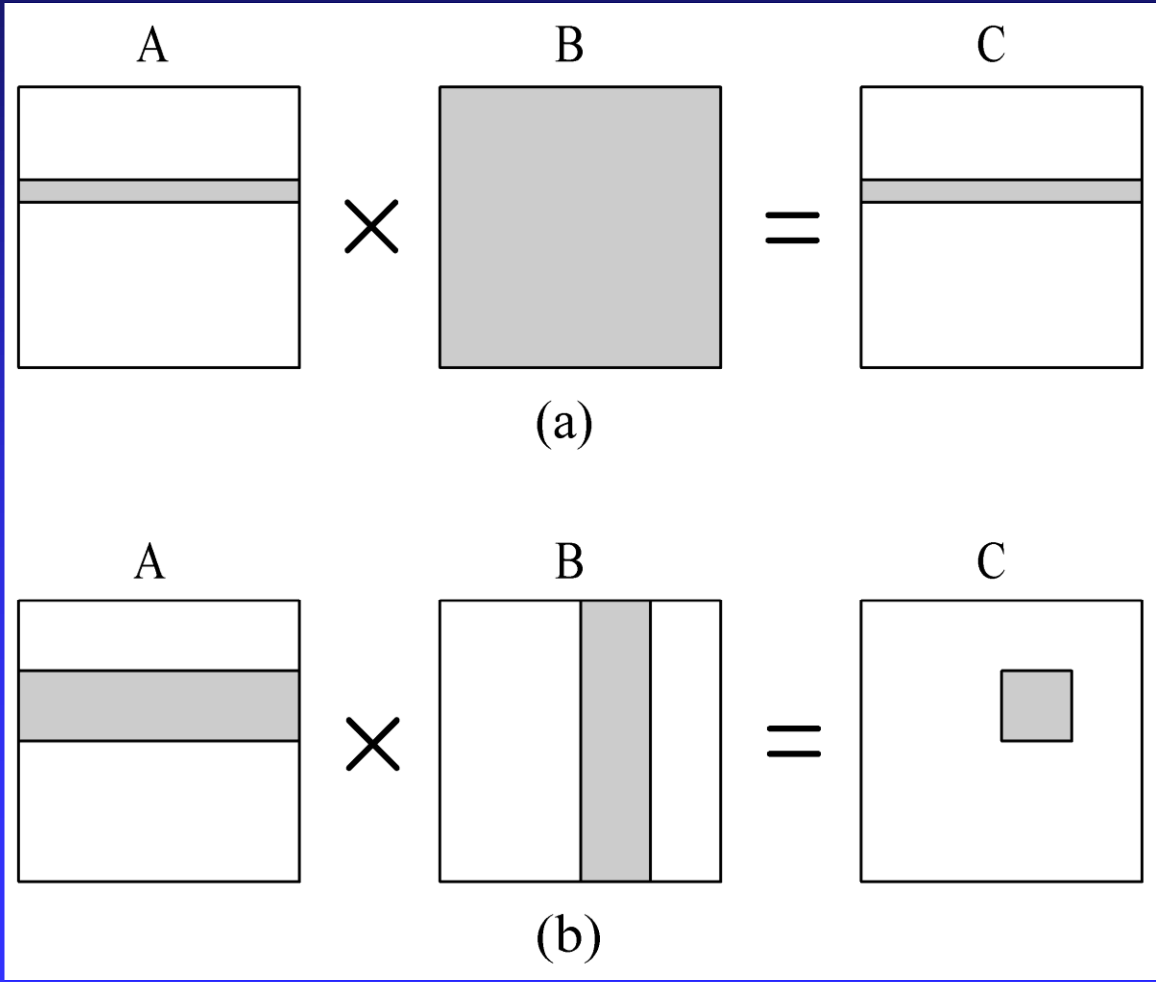
Weakness of Algorithm 1

- Blocks of B being manipulated have *p* times more columns than rows
- Each process must access every element of matrix B
- Ratio of computations per communication is poor: only $2n / p$

Parallel Algorithm 2 (Cannon's Algorithm)

- Associate a primitive task with each matrix element
- Agglomerate tasks responsible for a square (or nearly square) block of C
- Computation-to-communication ratio rises to $n / \sqrt{p}$

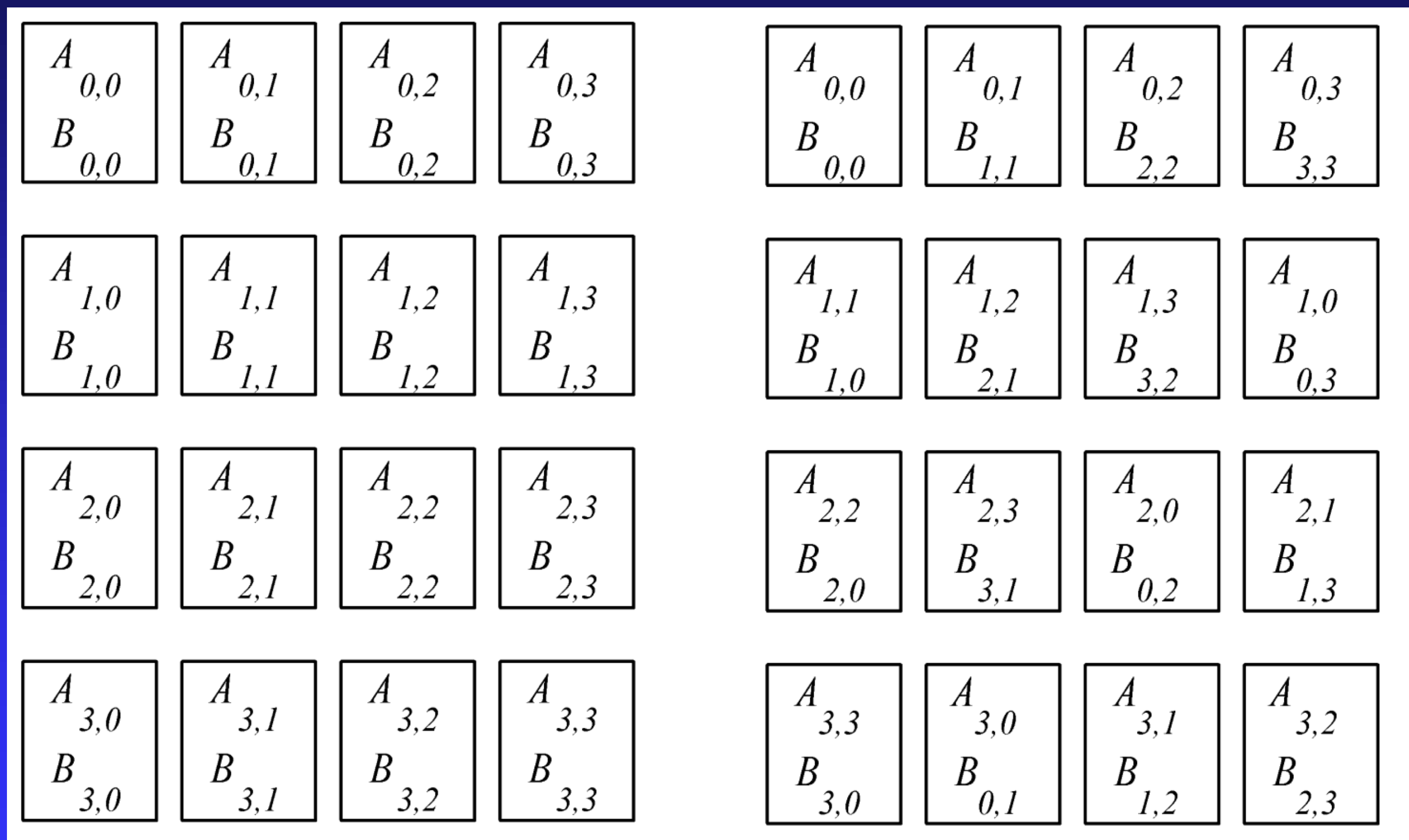
Elements of A and B Needed to Compute a Process's Portion of C



Algorithm 1

Cannon's Algorithm

Blocks Must Be Aligned

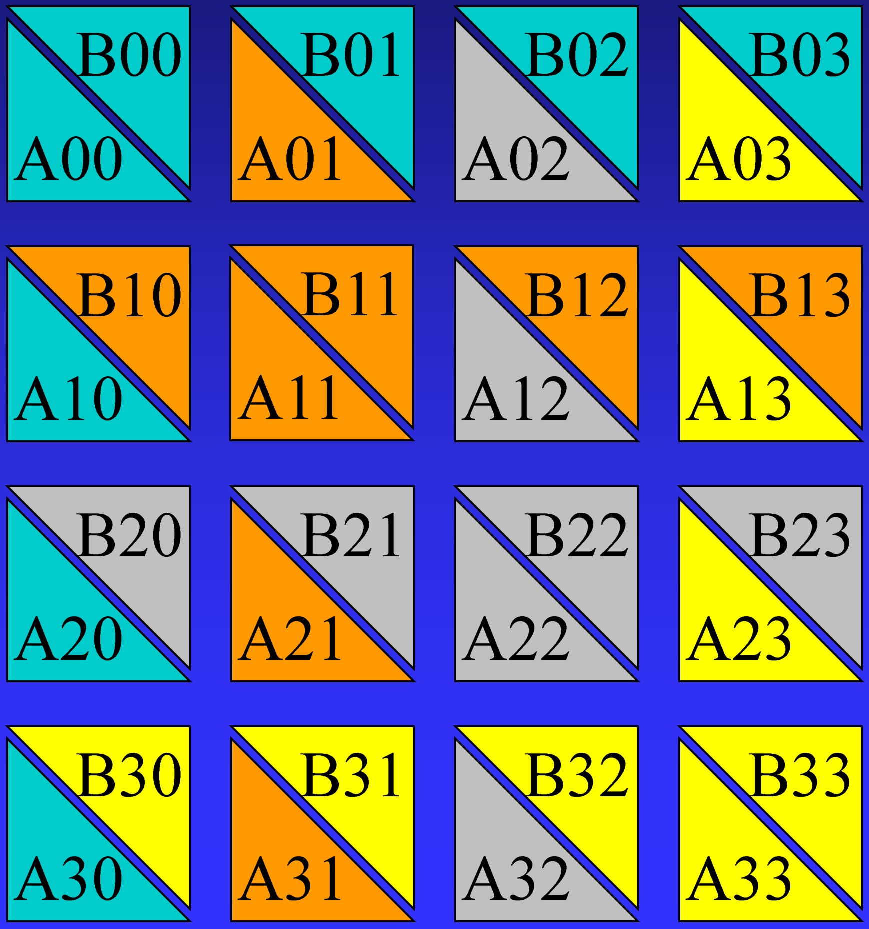


Before After

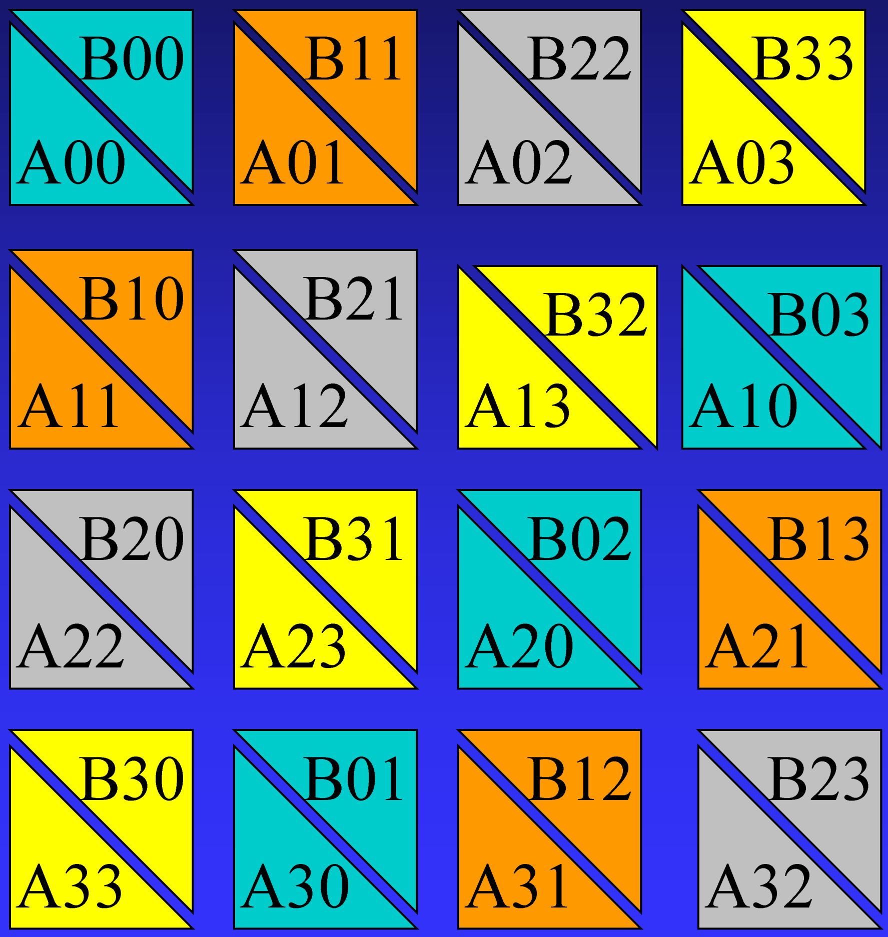
Blocks Need to Be Aligned

Each triangle represents a matrix block

Only same-color triangles should be multiplied



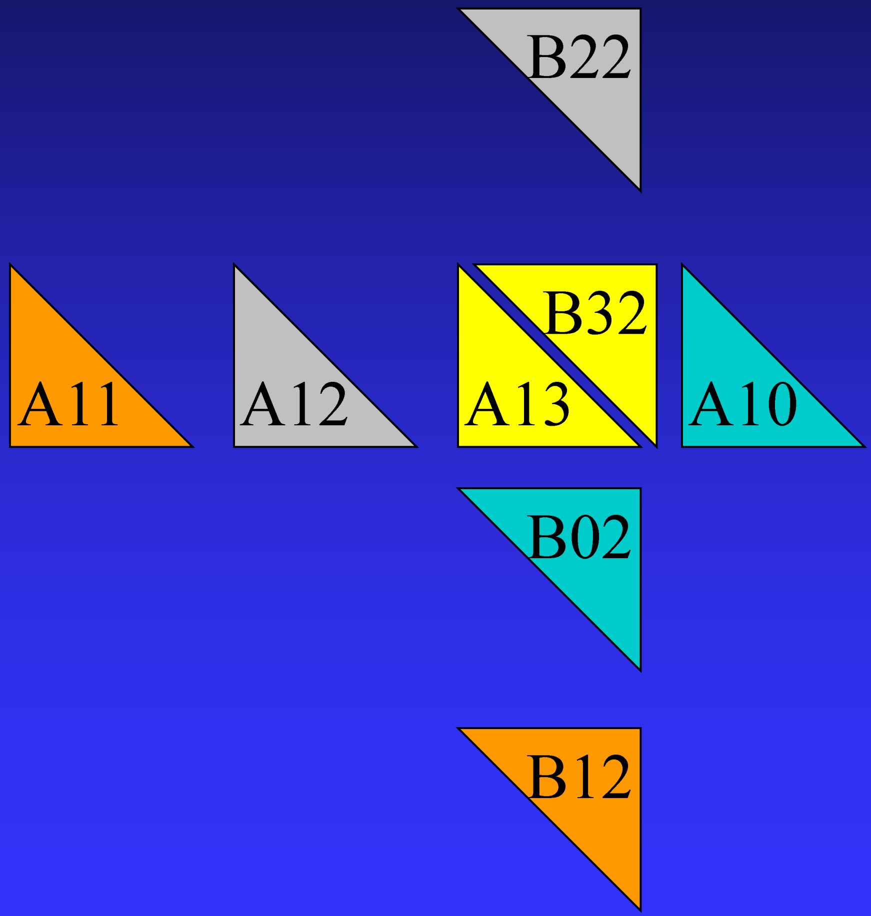
Rearrange Blocks



Block Aij cycles left i positions

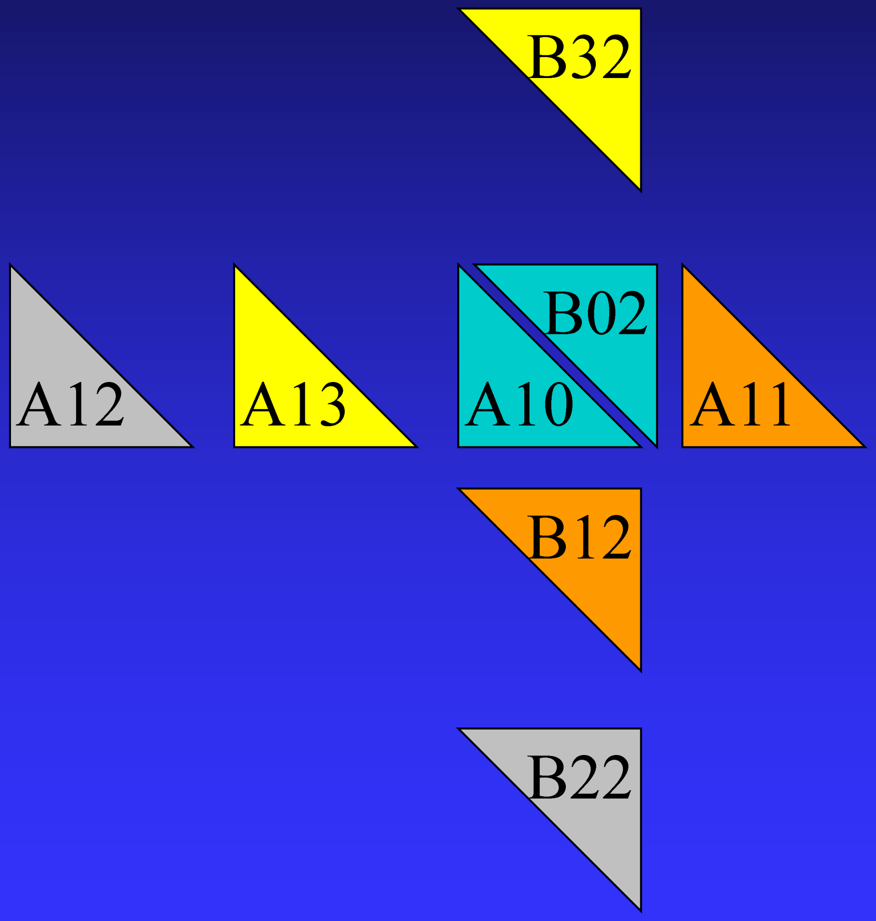
Block Bij cycles up j positions

Consider Process $P_{1,2}$



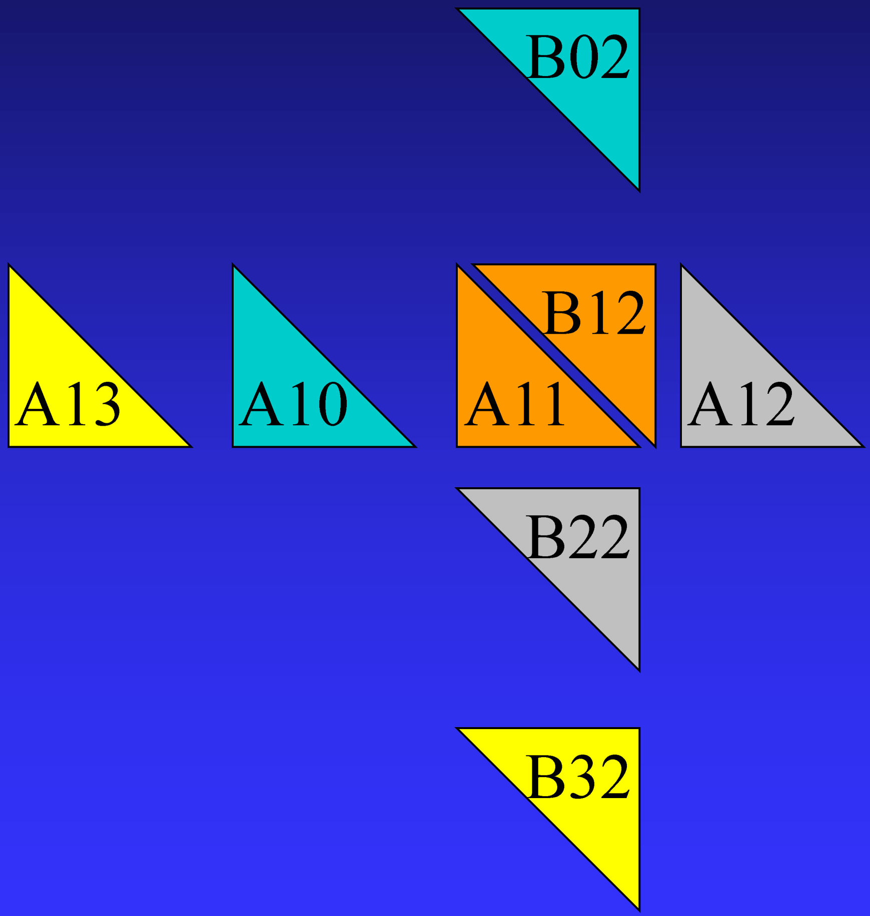
Step 1

Consider Process $P_{1,2}$



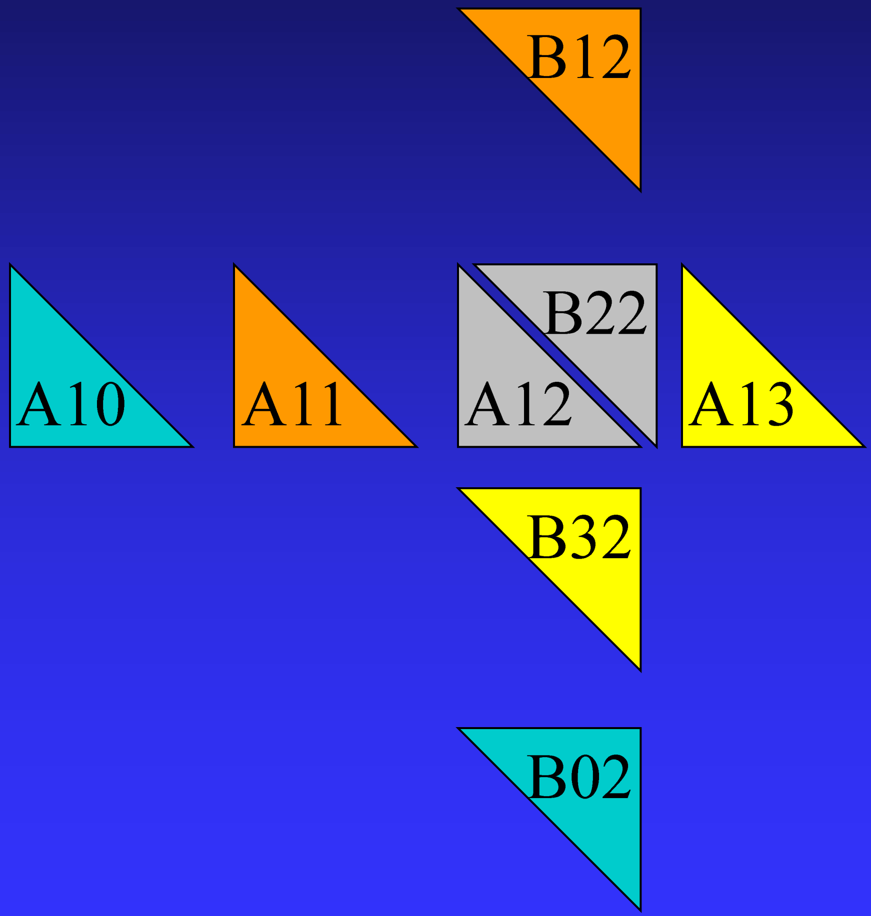
Step 2

Consider Process $P_{1,2}$



Step 3

Consider Process $P_{1,2}$



Step 4

Complexity Analysis

- Algorithm has $\sqrt{p}$ iterations
- During each iteration process multiplies two $(n / \sqrt{p}) \times (n / \sqrt{p})$ matrices: $\Theta(n^3 / p^{3/2})$
- Computational complexity: $\Theta(n^3 / p)$
- During each iteration process sends and receives two blocks of size $(n / \sqrt{p}) \times (n / \sqrt{p})$
- Communication complexity: $\Theta(n^2 / \sqrt{p})$

This system is highly scalable!

- Sequential algorithm: $\Theta(n^3)$
- Parallel overhead: $\Theta(\sqrt{p}n^2)$