CMOR 421/521:

Linear Algebra applications with MPI

M	W	F	
			1
			2
			3
			4
			5
			6
			7
			8
			9
			10
			11
			12
			13
			14
			15

Topics

- What MPI routines do parallel implementations of different linear algebra operations use?
- What is the data layout?
- How do we analyze the cost of different implementations?

Low rank mat-vec y = a * b’ * x

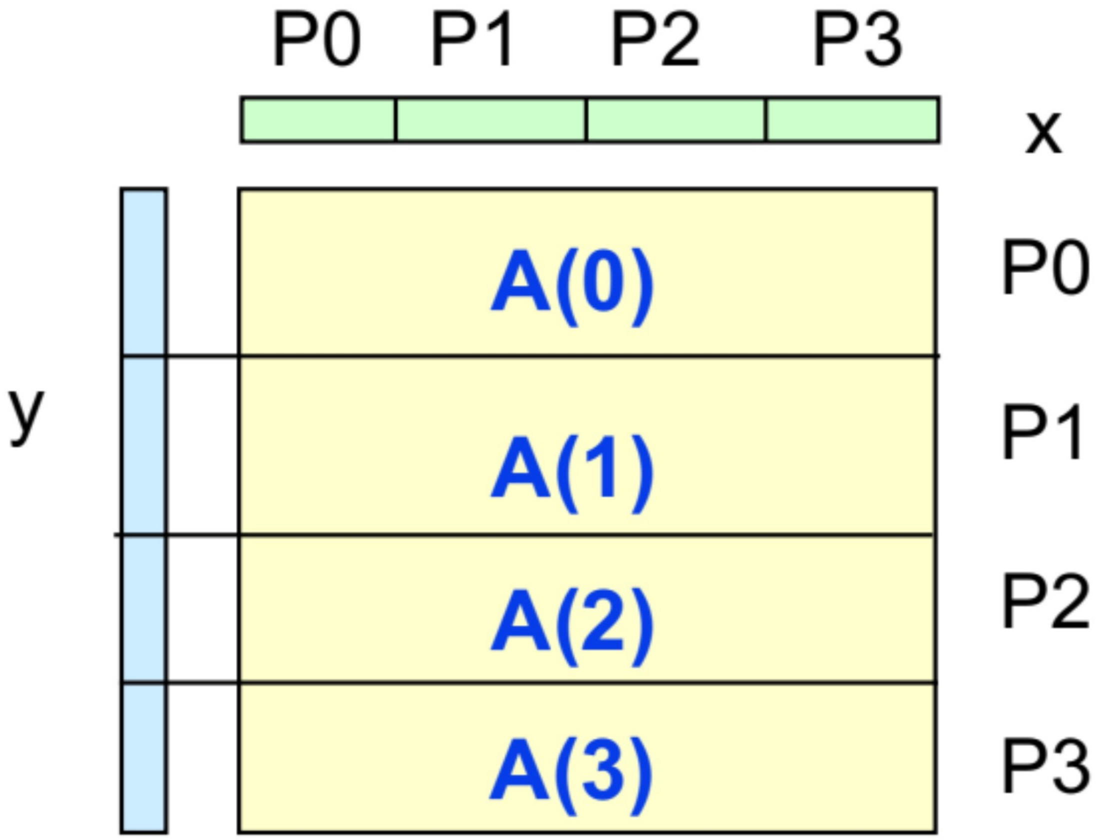
- Assume each processor owns a block of b = n / size entries of x, y, a, b.
- What MPI commands do we need?

Low rank mat-vec y = a * b’ * x

- Assume each processor owns a block of b = n / size entries of x, y, a, b.
- What MPI commands do we need?
  - Compute b_dot_x_reduced = b’ * x
    - Sum local dot products: MPI_Reduce
    - MPI_Bcast reduction to all processes
    - Combine Reduce/Bcast via MPI_Allreduce
  - Locally compute y_i = a_i * b_dot_x_reduced;

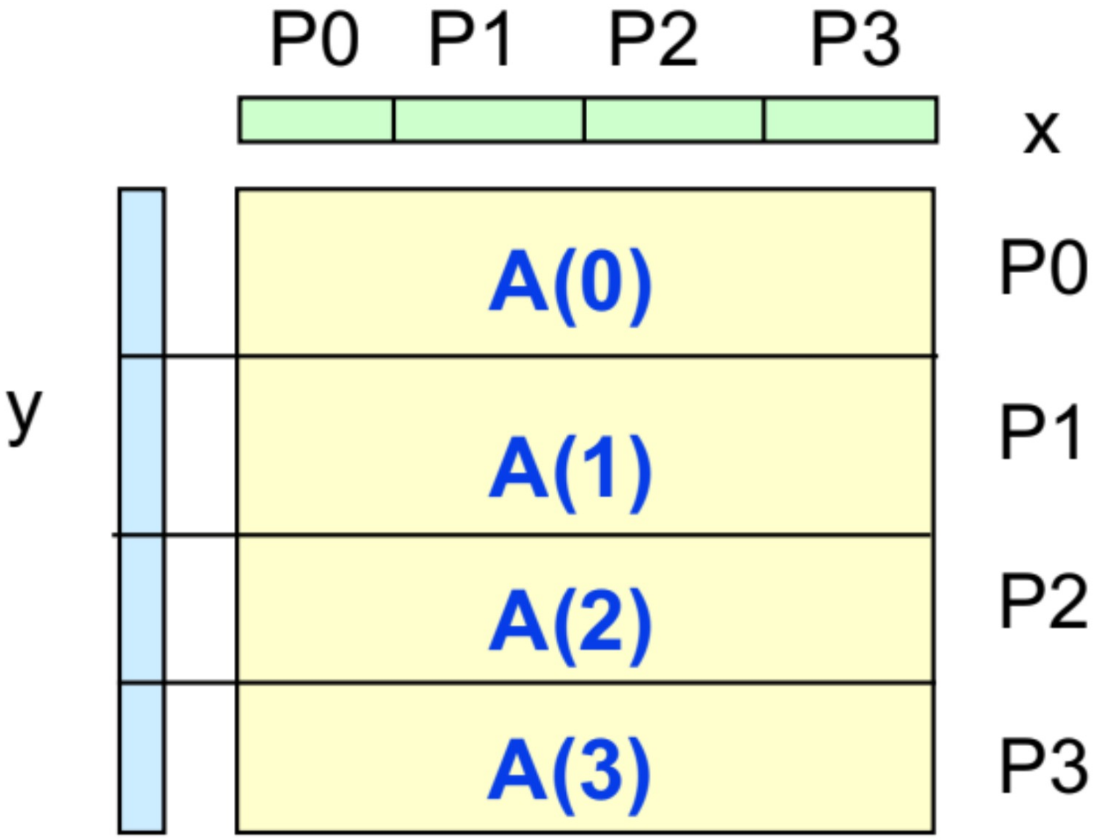
y = A * x (row major storage)

- Assume each processor owns a block of x, y, and a row block A_i
- What MPI commands do we need?



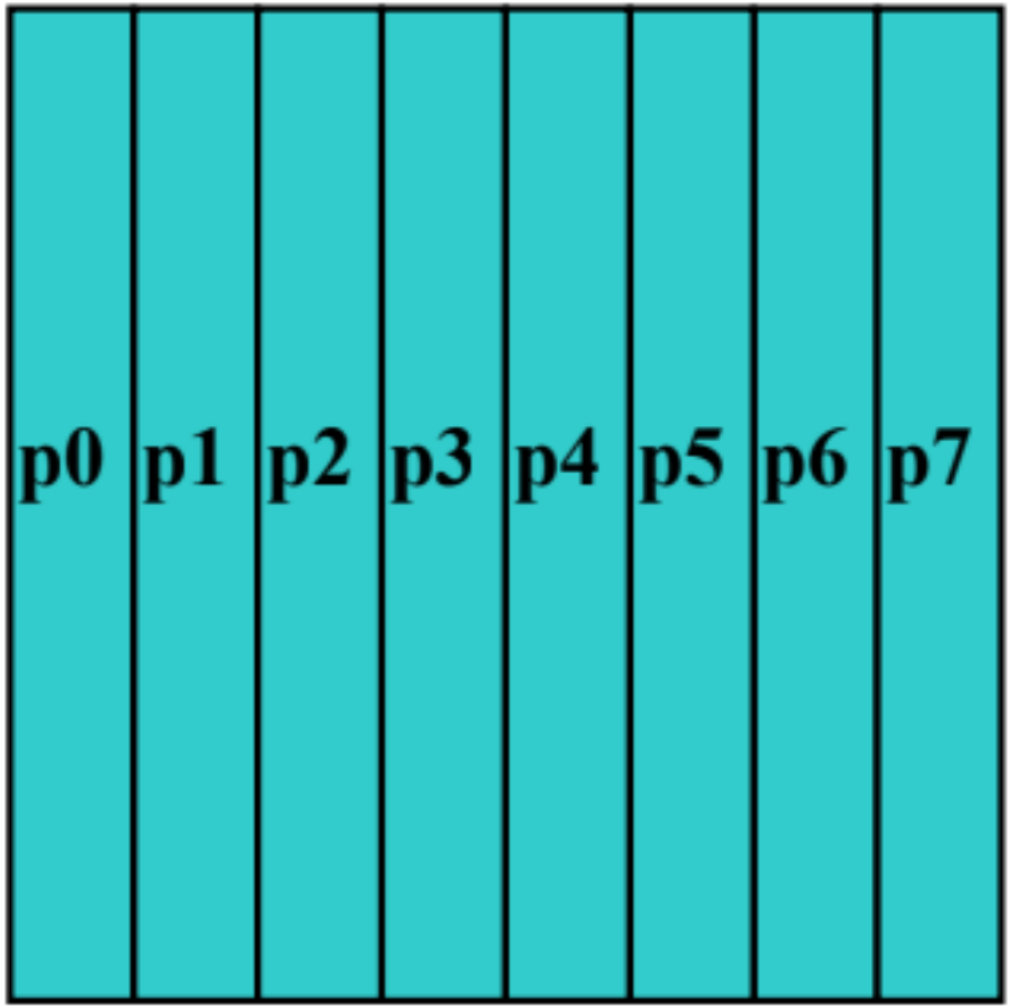
y = A*x (row major storage)

- Assume each processor owns a block of x, y, and a row block A_i
- What MPI commands?
  - Each processor needs all pieces of x
  - MPI_Allgather to collect pieces of x and reassemble
  - Redundant storage of x



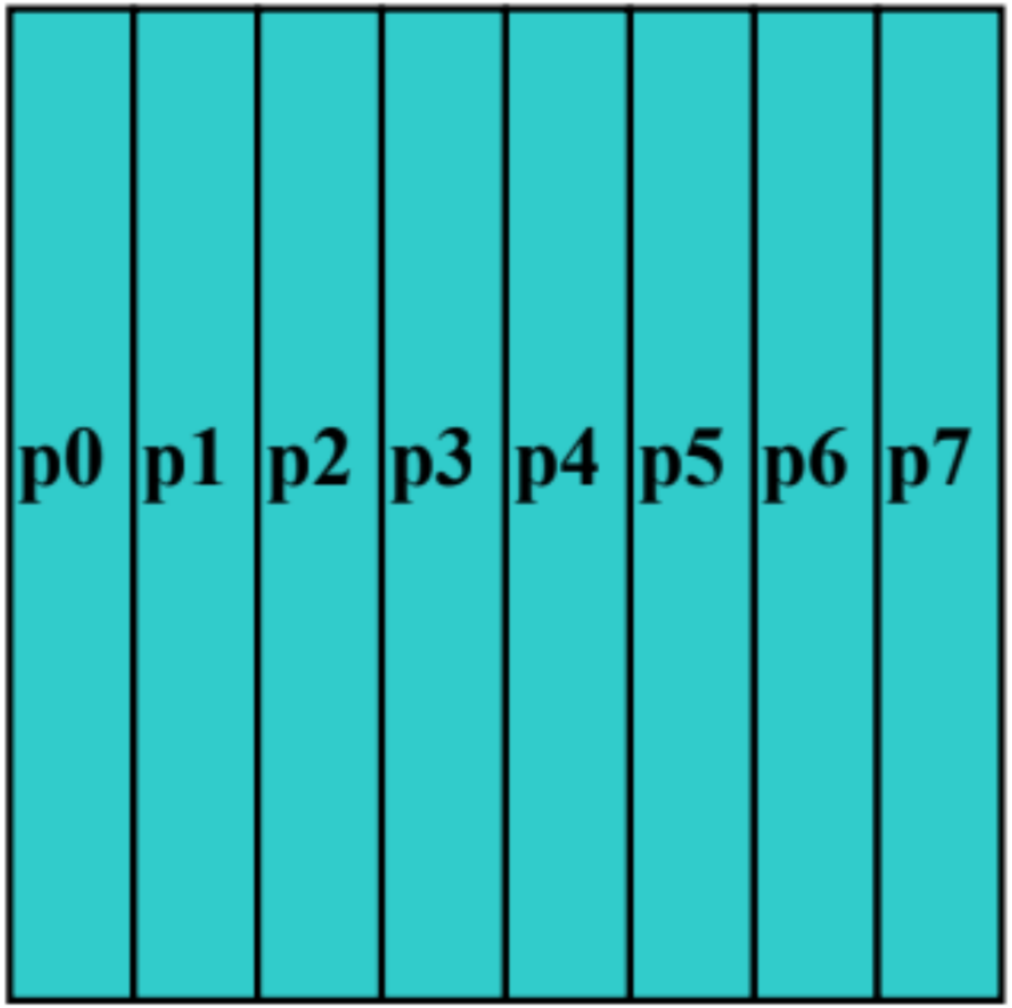
Mat-vec (column major storage)

- Assume each processor owns a block x_i, y_i, and a column block A_i
- What MPI commands?
  - Each processor computes matvec A_i * x_i locally
  - Need to sum matvec over all processors, then break up result into local blocks



Mat-vec (column major storage)

- Assume each processor owns a block x_i, y_i, and a column block A_i
- What MPI commands?
  - Each processor computes matvec A_i * x_i locally
  - Need to sum matvec over all processors, then break up result into local blocks
  - MPI_Alltoall, MPI_Reduce + MPI_Scatter?



Mat-vec (column major storage)

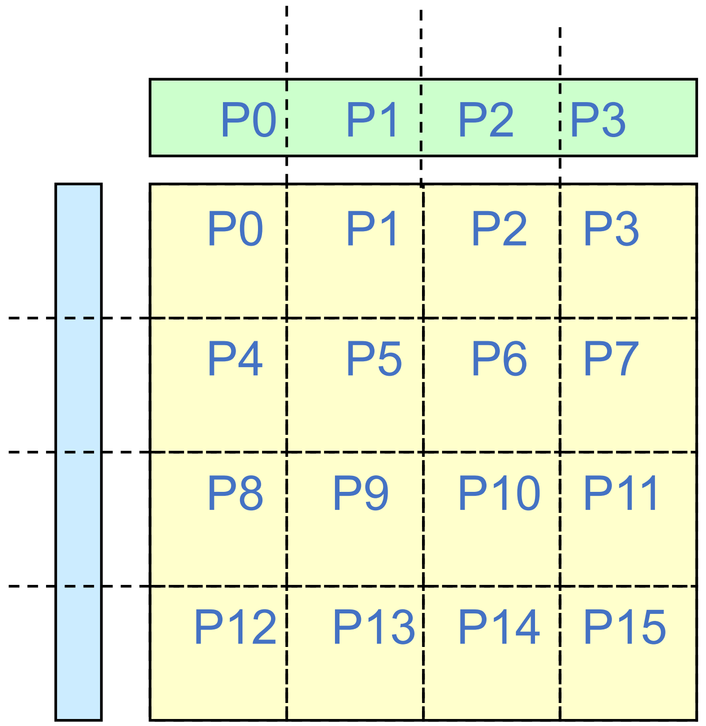
Figure from Mohammad Hammoud

MPI Collectives

- Point-to-point (one-to-one)
  - Send, Recv (and the non-blocking versions)
- All-to-one, one-to-all
  - Bcast, reduce, gather, scatter
- All-to-all
  - AlltoAll, Allreduce, Allgather, ReduceScatter

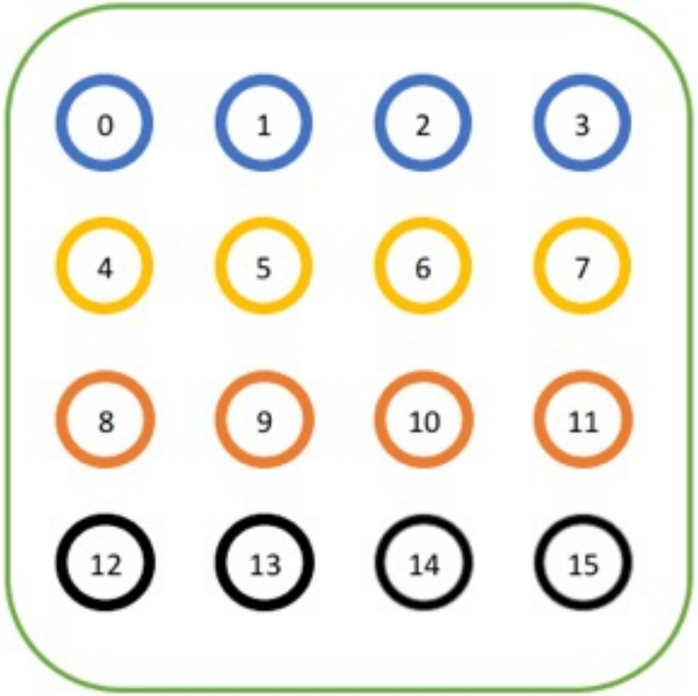
Mat-vec (row major block storage)

- Assume p processors in a sqrt(p) x sqrt(p) grid
- Who owns what now?
  - Suppose P0, P1, P2, P3 hold x[i] and y[i]
- It would be nice if we could restrict collective operations to only a subgroup of workers…



Why collectives vs point-2-point?

- Less explicit management of communication
- Collectives + MPI communicator groups: bulk of work is moved to data layout and setup



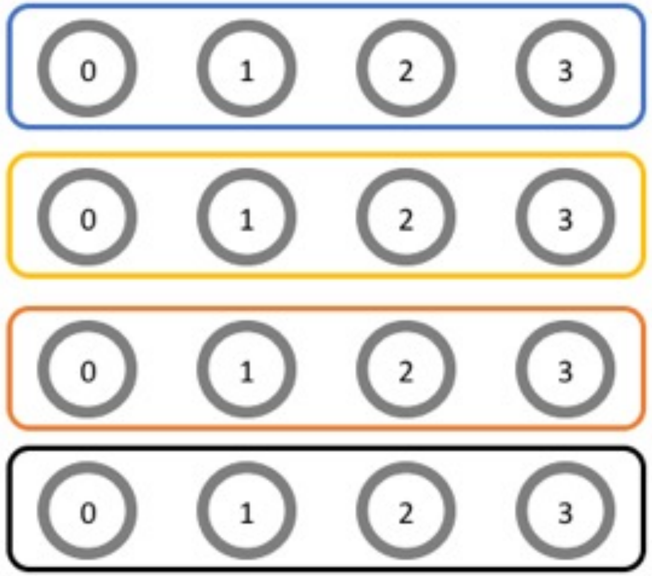


Image from Wes Kendall's MPI Tutorial

MPI communicators

- Useful for defining groups to perform collective operations over among specific groups of processes
- Custom MPI communicators can be useful for this

- MPI_Comm_split splits processes in an existing communicator into a new communicator
- MPI_Comm_create, etc, for more general groups
  - Can also communicate in-between communicator groups…

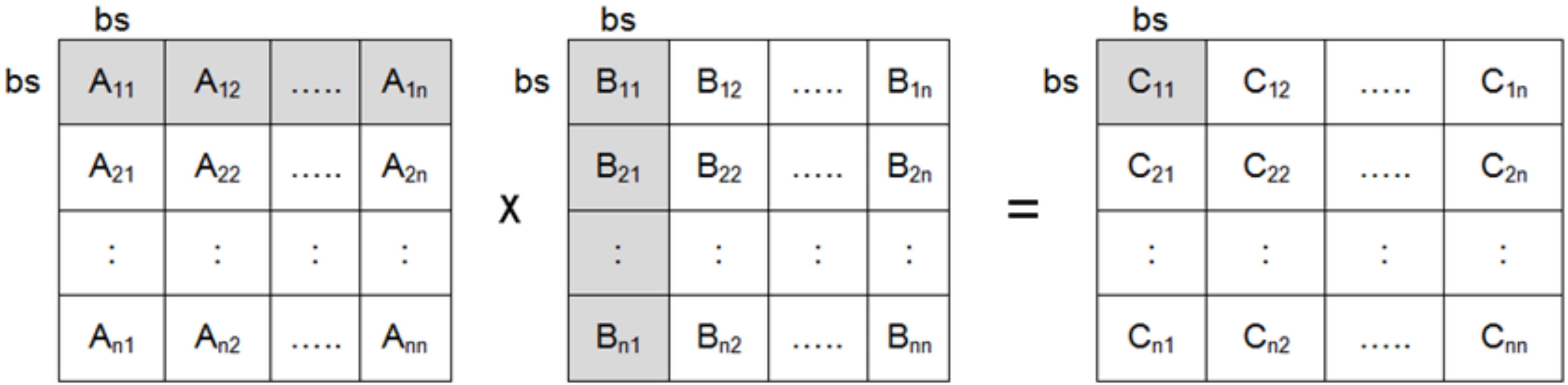
MPI_Comm_split

```
int MPI_Comm_split(
    MPI_Comm comm,
    int color,
    int key,
    MPI_Comm newcomm,
);

// Note: no tag anymore
```

Argument	Description
comm	The communicator to split
	For example, MPI_COMM_WORLD
color	Processes with the same “color” are in the same new communicator
key	Int
	The order (not value) determines the new local ordering
newcomm	The new communicator

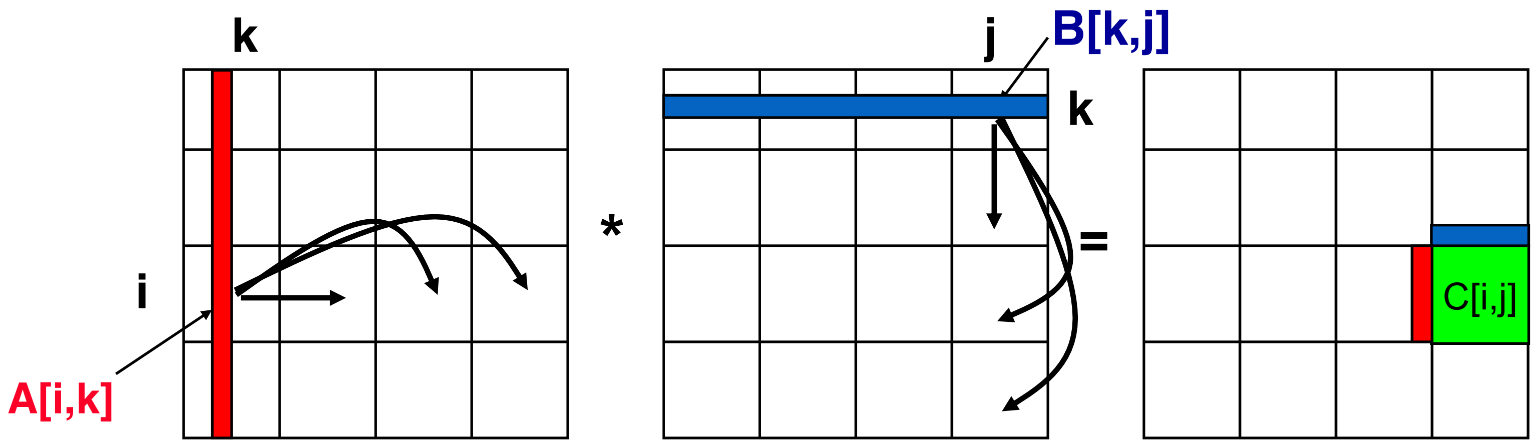
Blocked matrix multiplication



- Each processor holds a block of C, A, B
- Block ordering is the same across each matrix
- What communication/MPI commands?

Image from http://dx.doi.org/10.11591/ijeecs.v11.i3.pp954-961

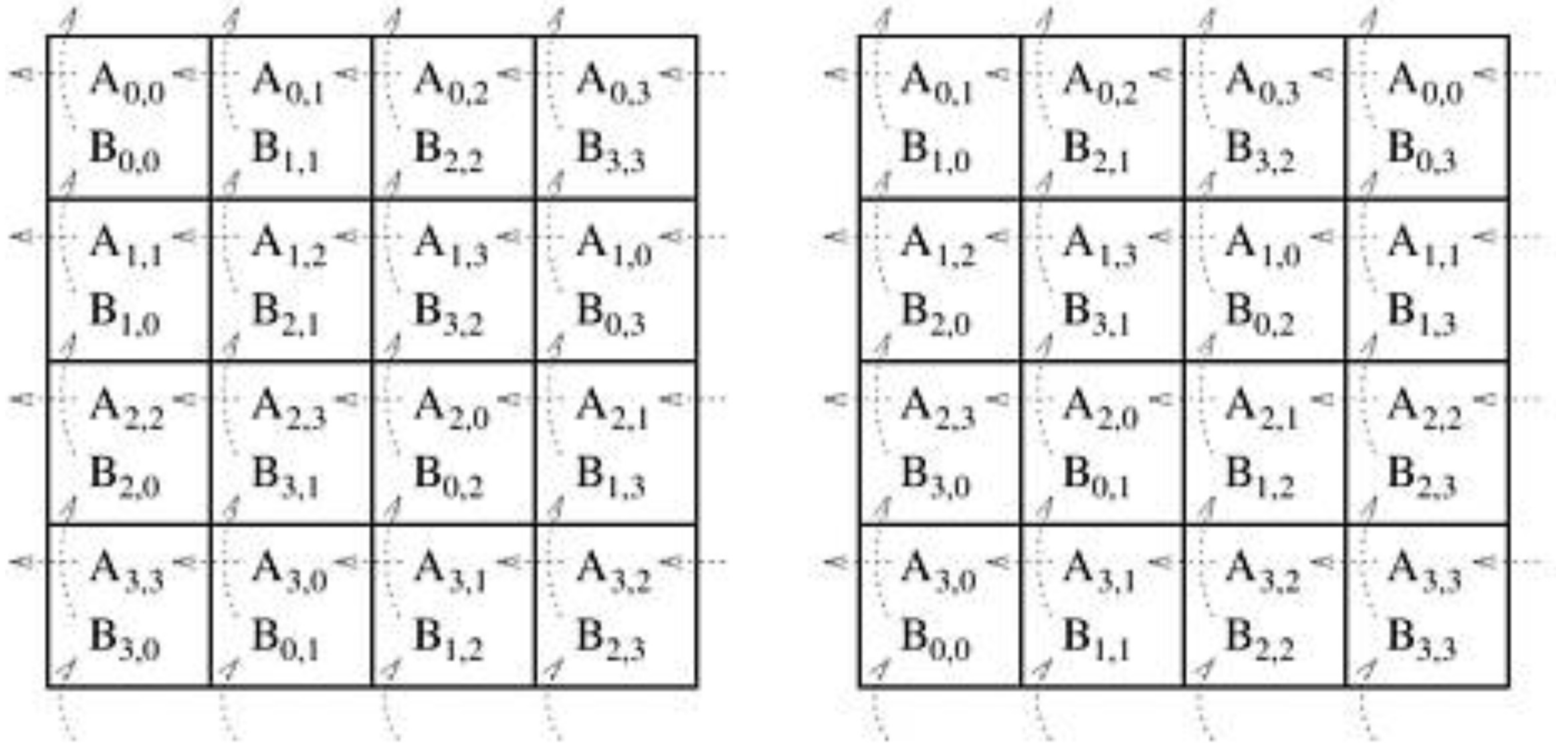
SUMMA: Scalable Universal Matrix Multiplication Algorithm



- Modification to reduce memory, computes outer products of b x k blocks, tune k for memory usage.
- Uses the fact that MPI_Bcast is pretty fast - log(p)

Van de Geijn, Watts (1997)

Cannon’s algorithm



- Slightly more efficient, less flexible than SUMMA
- Each processor holds a block of C, A, B
- Performs n / p “shift” communication steps

MPI Cartesian topologies

- MPI_Cart_create constructs a communicator with a Cartesian grid *topology*
- Query functions like MPI_Cart_coords, MPI_Cart_shift
- New communicator functions like MPI_Cart_sub (create a communicator over a *subgrid)*

How do people usually expose parallelism?

Some examples

- Dense matrix-matrix multiplication
- Large systems of ODEs
  - Explicit and implicit time-stepping
- Grid or stencil-based methods (finite difference, finite element methods, etc)

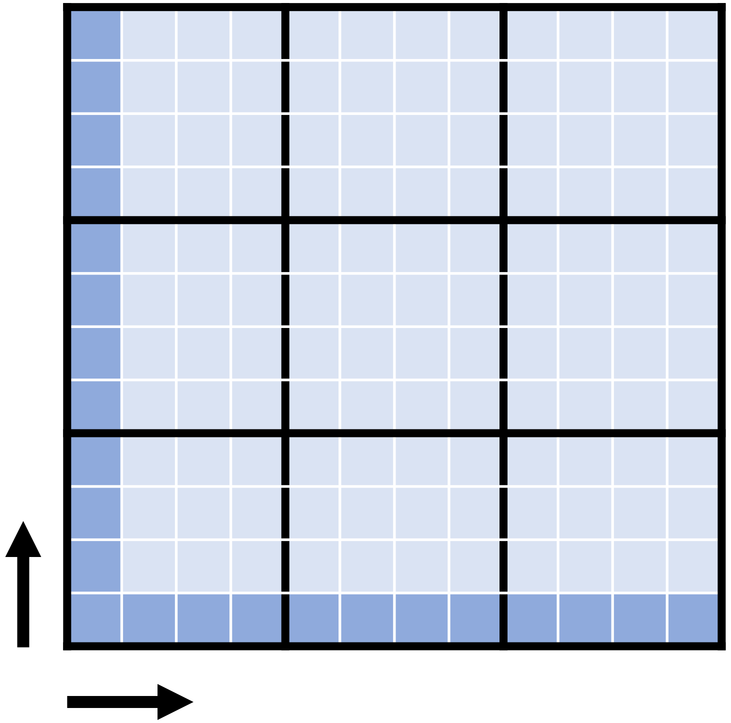
Old Friends: The FD BVP

- How might this problem look different now?

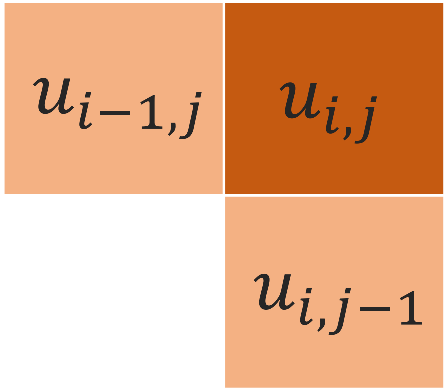
Domain

Known values (boundaries)

Unknown values (interior)



FD stencil



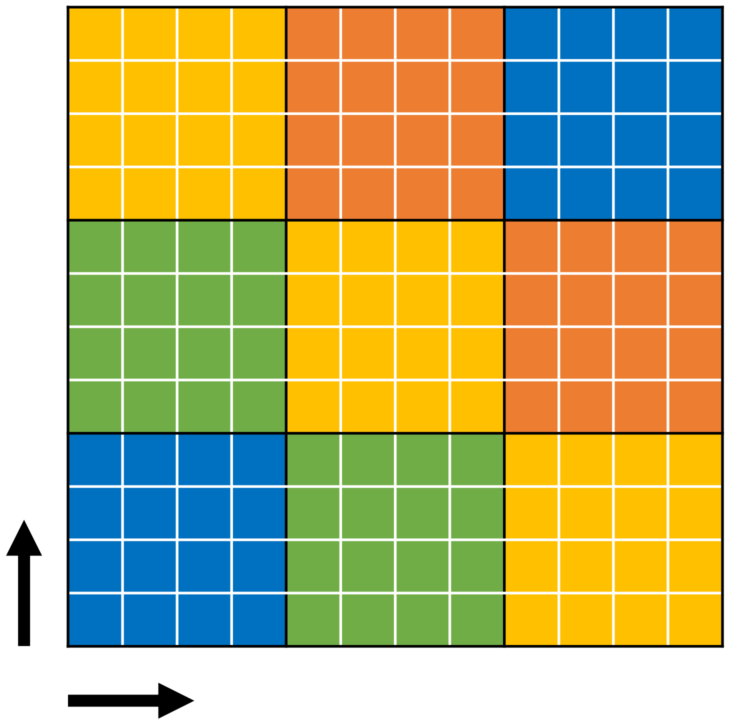
The FD BVP with Communication

The order dependency is exactly the same

- We have to parallelize using a wavefront scheme

But no, there is no shared memory…

- The blocks tell us what values we assign to, not the values we need (i.e. the values we read from)
- We're assuming there is no shared memory; how do blocks get this data?



The FD BVP with Communication

Blocks now have to send and receive information between each parallel step

- There is a "halo" of values around the block that are needed for computation
  - These are also called ghost values
- Those values are computed by different blocks (or maybe not at all)
- The blue block has no halo/ghost values since it is on the boundary
- **We have more work to do now…**