Introduction to Parallel Processing

Lecture 9 : Sparse Matrices

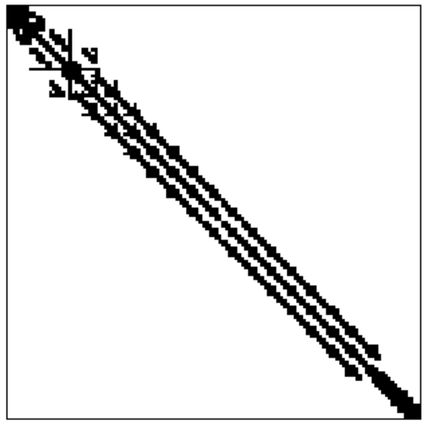
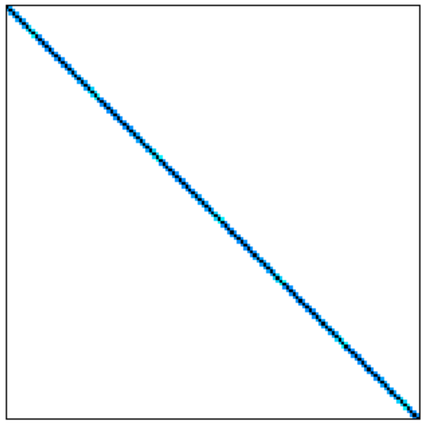
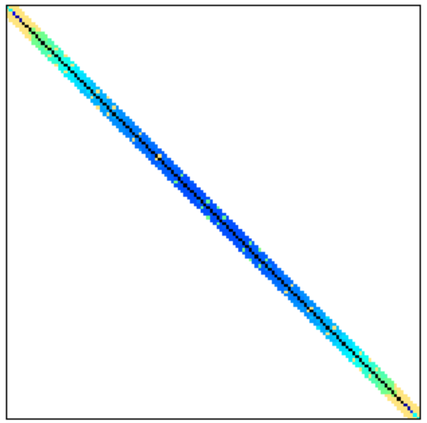
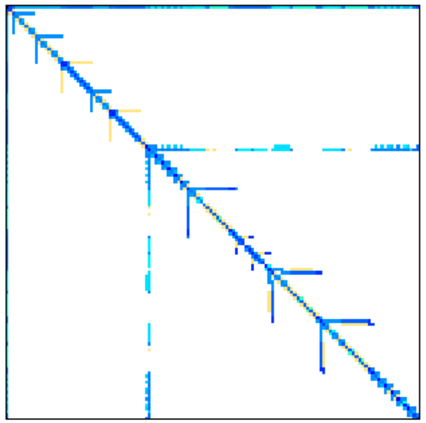
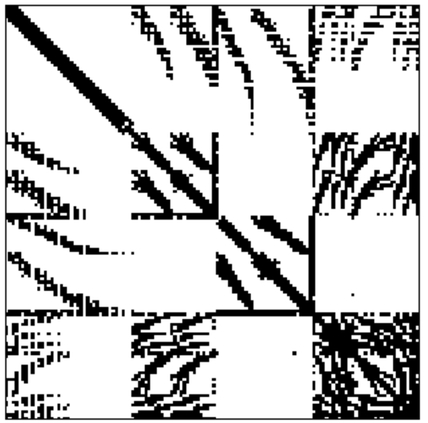
Professor Amanda Bienz

Sparse Matrices

- A **sparse** matrix is a matrix where most entries are zero
- Only need to store and operate on non-zero entries
- Not stored in 2D array… extra overhead in storage and computation
- "Useful" sparsity if $\mathcal{O}(n)$ non-zero entries
- Today we are going to talk about some special types of sparse matrices

Sparsity Pattern

- Dots are non-zero entries, white space is full of zeros

Coordinate (COO) Format

- Three arrays:
  - Rows : row of each non-zero index
  - Cols : column of each non-zero index
  - Data : values of each non-zero index
- Let’s step through an example…

Compressed Sparse Row (CSR) Format

- Three arrays:
  - Rowptr : points to index in next two arrays of beginning of each row
  - Cols : column of each non-zero index
  - Data : values of each non-zero index
- Let’s step through an example…

Compressed Sparse Column (CSC) Format

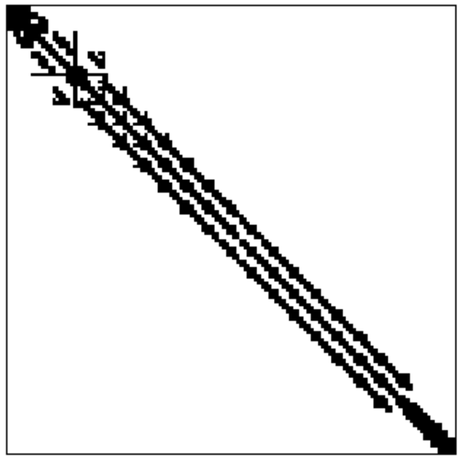
- Three arrays:
  - Colptr : points to index in next two arrays of beginning of each column
  - Rows : row of each non-zero index
  - Data : values of each non-zero index
- Let’s step through an example…

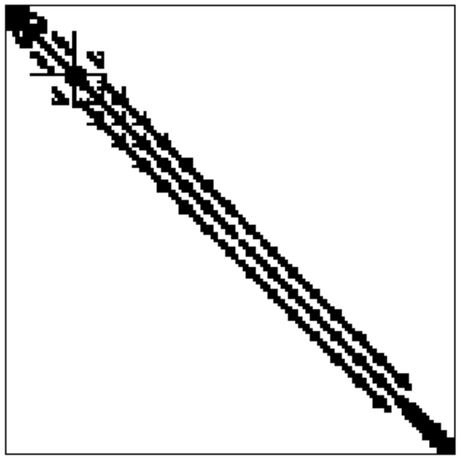
Sparse Matrix Datatypes

- Most commonly, sparse matrix operations use CSR format
  - What are the storage requirements for a matrix with n rows and nnz non-zero elements?
  - How many reads are needed when stepping through a sparse matrix?
  - How many reads are needed if this matrix was stored in dense format instead?
  - How sparse does a matrix need to be for it to be cheaper to use CSR format than storing as dense?

Serial SpGEMM Algorithm

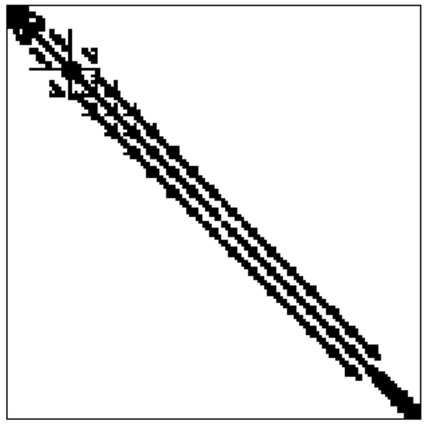
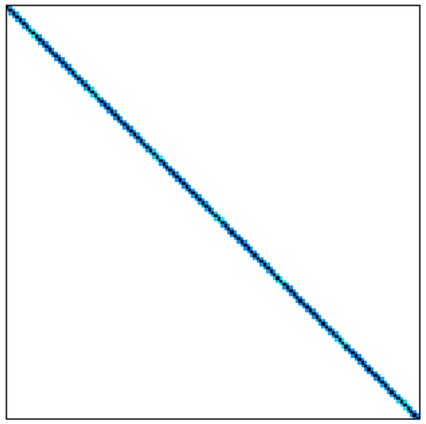
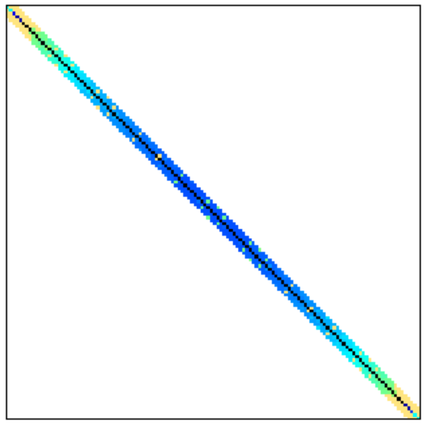
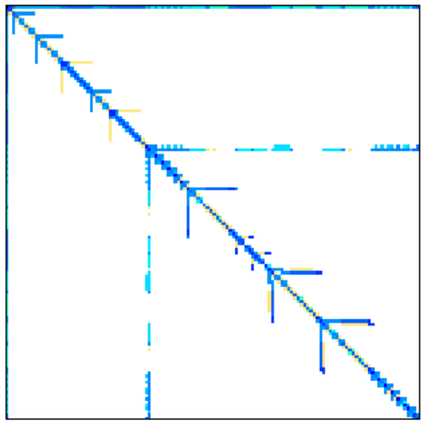
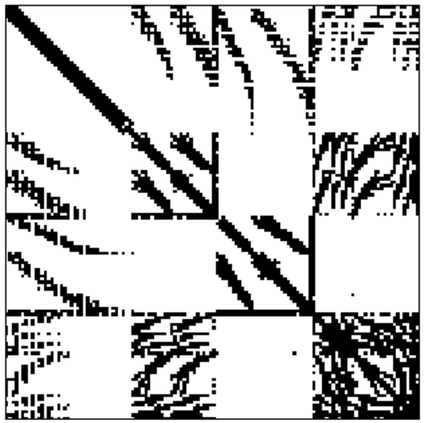
- How do we multiply two sparse matrices?
- What if we use same approach as a dense matvec?





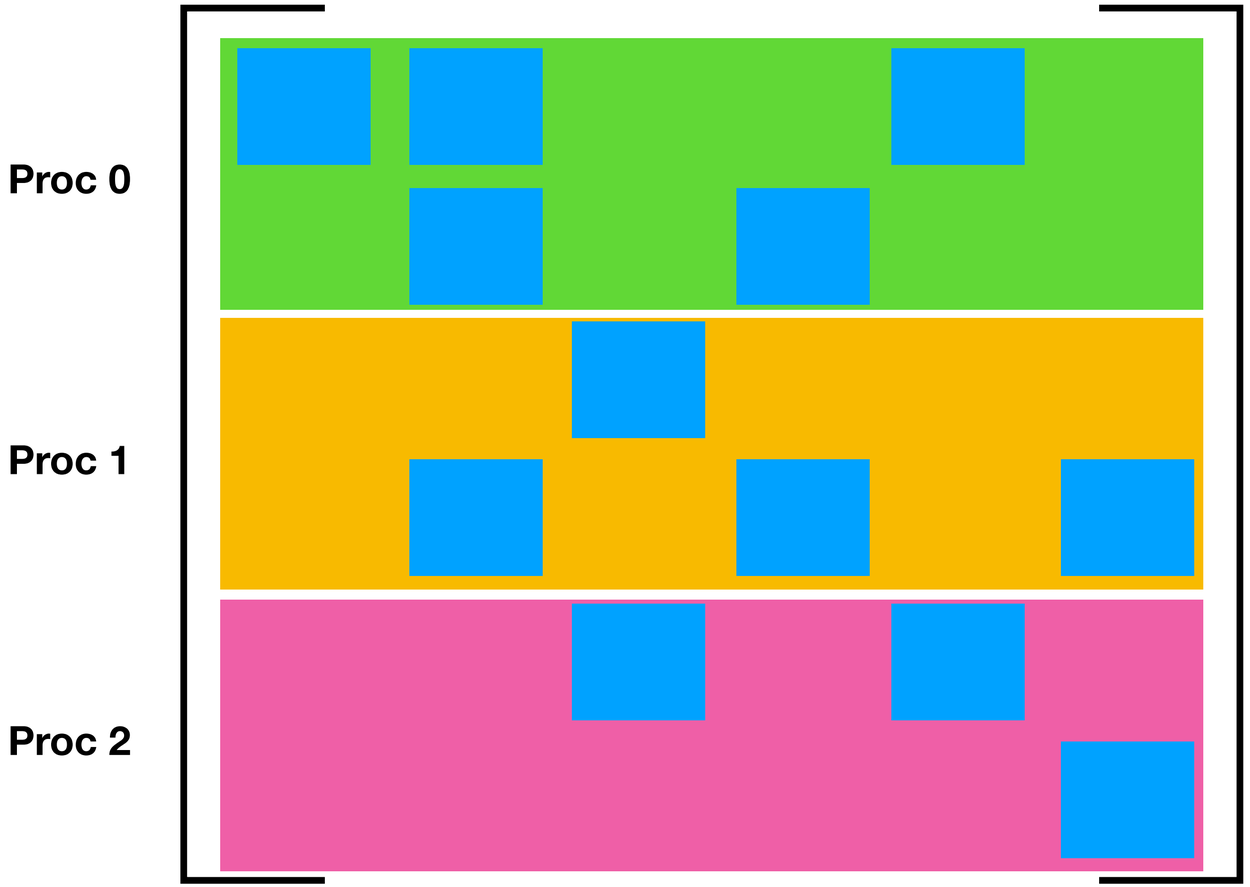
Parallel Partitioning

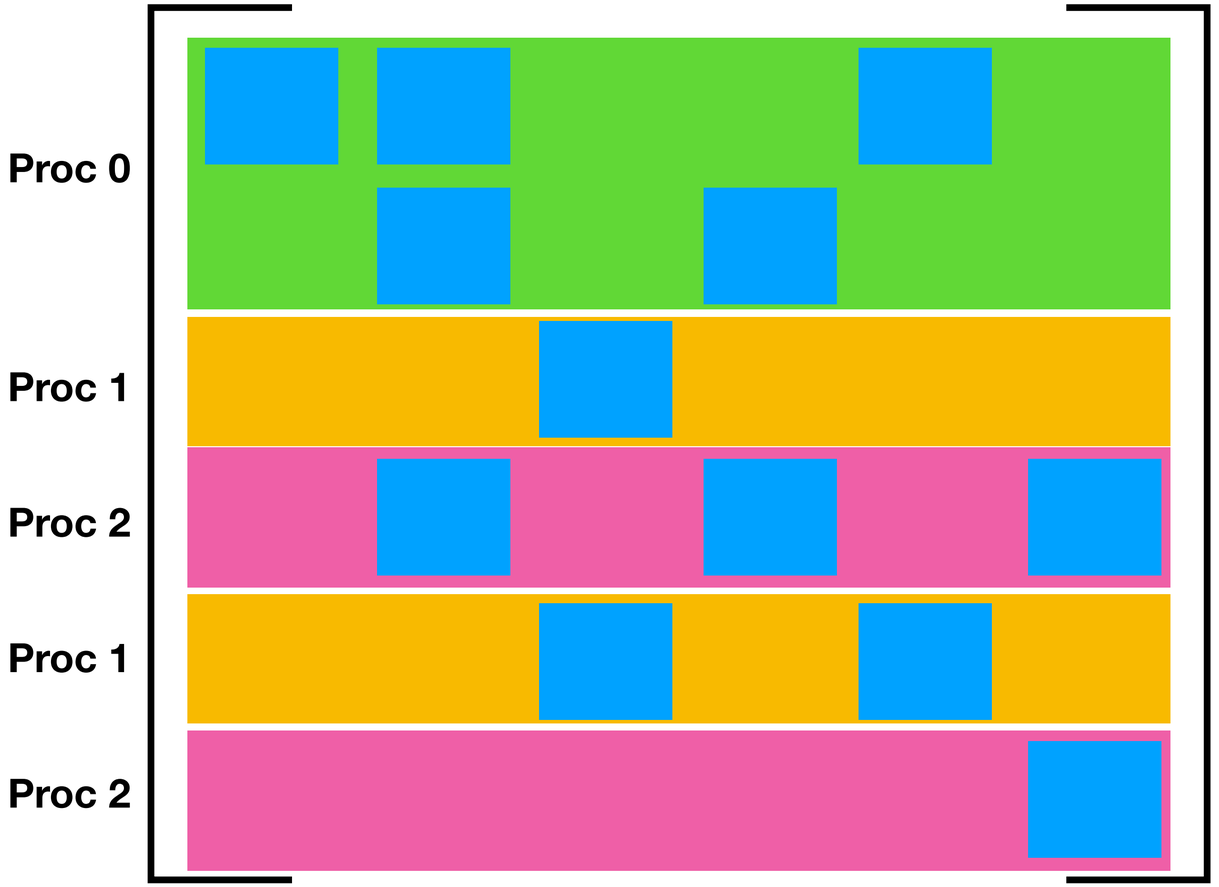
- For dense matrices, we wanted 2D partitions… what about when sparse?

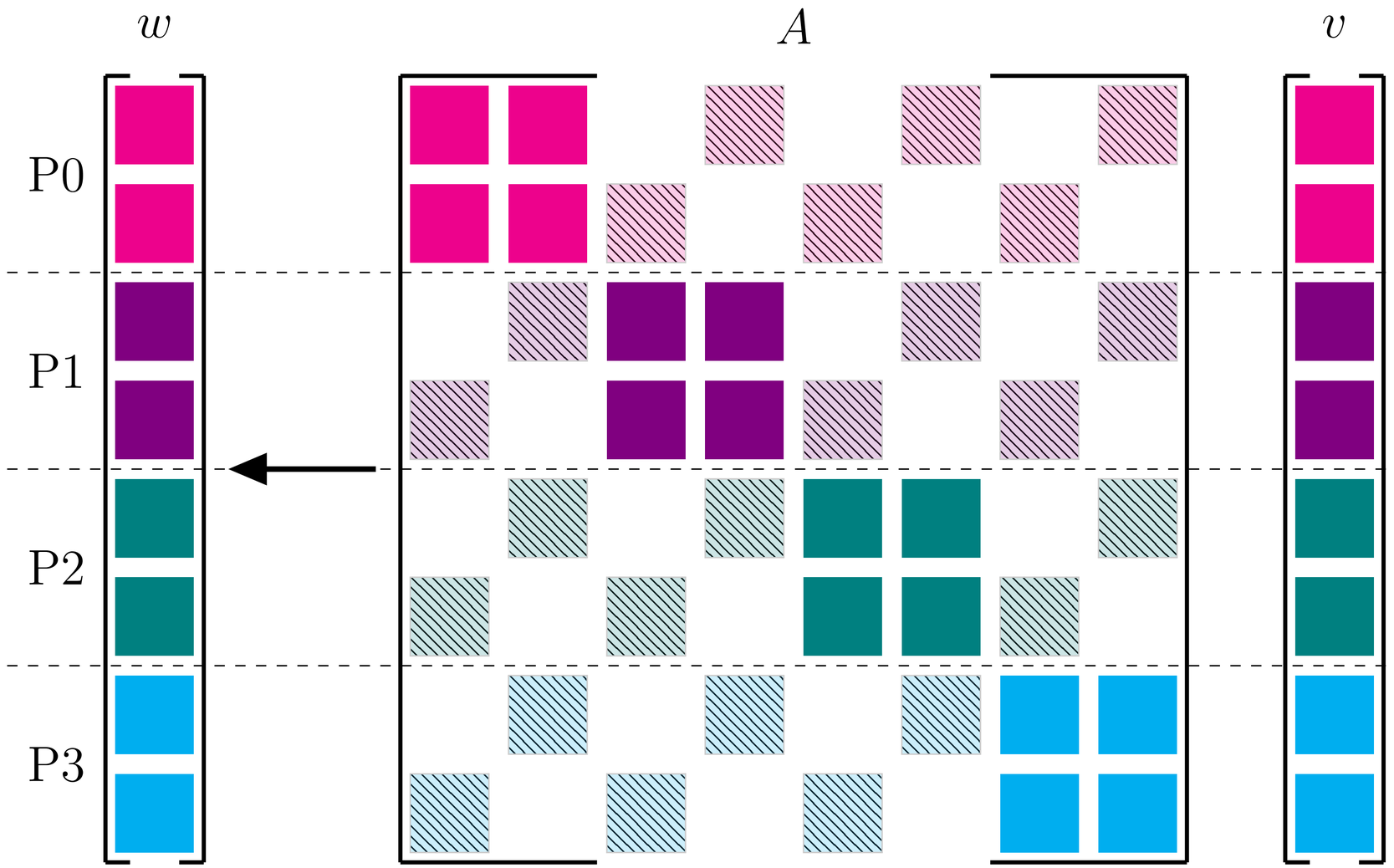
Parallel Partitioning

- Ideal partition varies with matrix : **hard problem**



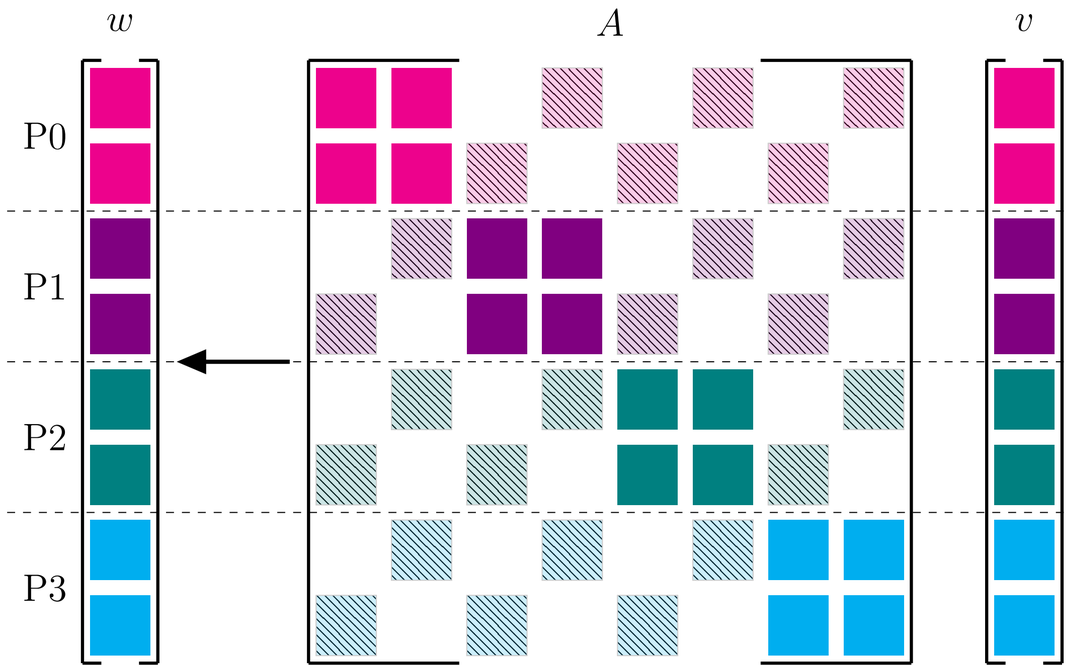


Parallel CSR



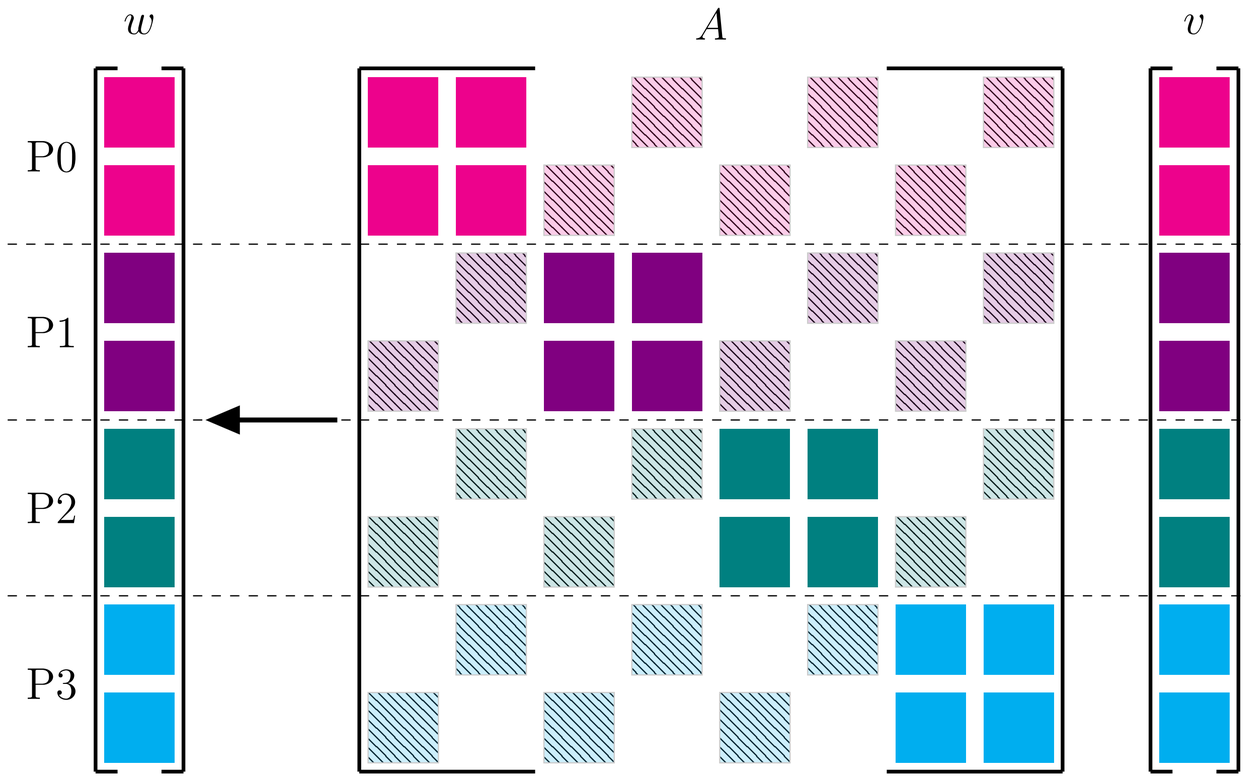
Sparse Matrix Operations - Communication Pattern

- SpMV: Process p needs to receive all values of v corresponding to local columns that hold non-zeros
- Easy to figure out processes from which to recv and what to recv from each
- Sending side : more difficult to figure out, requires probe. Figure out one time and reuse (communication package)



Parallel SpMV

- Initialize MPI_Isend and MPI_Irecv operations
- Perform local SpMV
- Wait for non-blocking communication to finish
- Perform non-local SpMV



Parallel SpMV Drawbacks

- No structured pattern of communication with sparse matrices
- Each process only talks to a small subset of the other processes (but this could mean 1000 of the 100,000 processes)
- Typical approach : send all 1000 messages, wait to receive the 1000 messages that I need
  - **Large communication costs!**
- For dense matrices, could have all processes hold entire vector… not the case for sparse matrices (that could be too large of a memory requirement)

Parallel SpGEMM

- Initialize MPI_Isend and MPI_Irecv operations
- Perform local SpGEMM
  - C_on = A_on * B_on
    C_off = A_on * B_off
- Wait for non-blocking communication to finish
- Perform non-local SpGEMM
  - C_on += A_off * B_recv_on
    C_off += A_off * B_recv_off