Cannon's Algorithm and pseudo code

Given 2 $n_x n$ matrices and p processes you will divide each matrix into a total number of p submatrices with block sizes of $\frac{n}{\sqrt{p}}$

Note: All Matrices are zero indexed meaning $A_{i=0\rightarrow n, j=o\rightarrow n}$ and $B_{i=0\rightarrow n, j=o\rightarrow n}$
Note: Assume you are always given perfect square matrices.

- Allocate and read in matrices $A_{i,j}$ and $B_{i,j}$ such that each row and column has a total of $\sqrt{p}$ submatrices where p is the total number of ***processes***.
- Allocate and initialize a solution matrix $C_{i,j}$
- Create new communicator(s)
- Create your Cartesian topology setup
- Align block elements $A_{i,j}$ and $B_{i,j}$ such that $A_{i,j+i}$ and $B_{i+j,j}$ are on block/process $p_{i,j}$
  Given matrix A with blocks A1=>A16 sequentially

$$\begin{bmatrix} A1 & A2 & A3 & A4 \\ A5 & A6 & A7 & A8 \\ A9 & A10 & A11 & A12 \\ A13 & A14 & A15 & A16 \end{bmatrix} \xrightarrow{Shift\ rows\ left, columns\ up} \begin{bmatrix} A1 & A2 & A3 & A4 \\ A6 & A7 & A8 & A5 \\ A11 & A12 & A9 & A10 \\ A16 & A13 & A14 & A15 \end{bmatrix}$$

In MPI think MPI_Cart_shift(…,.,., &rightrank, &leftrank) and MPI_Cart_shift(…,.,., &downrank, &uprank

- Each $p_{i,j}$ locally calculates $C_{i,j} = A_{i,j+i} * B_{i+j,j}$
- For (k = 0; n-1; k++)
  - Shift A left one column
  - Shift B up one row
  - Each $p_{i,j}$ locally calculates $C_{i,j} += A_{i,j+i+k} * B_{i+j+k,j}$
- Next
- Restore original alignment of $A_{i,j}$ and $B_{i,j}$
- Free communicator(s)

TIPS AND HINTS for both success and maximum points:

- Don’t worry about HDF5 I/O at first. Create dummy matrices with known values and known results. A 4x4 matrix with a 1x1 submatrix is ideal for testing!

- Creating and using a simple matrix printout fxn is ***HIGHLY*** recommended!!! Not only for initial input matrices and final output matrix but using for debugging your shifting. ***HINT***: If you can’t prove your code works with outputted data (***in your report***) how does one know the code works? It’s not called a Results section for nothing!!

- When all else fails, pencil and paper algorithm tracing is time well spent!

- Better yet a few minutes of Matlab prototyping can also save lots of time and sanity!

- Both blocking and non-blocking methods are possible, be careful to choose one and stay with it.

- There’s a cool MPI fxn called MPI_Sendrecv_replace() that can be your friend!

- “Your code must run in serial” means running with np=1 you do not need separate serial code.

- Modular Code is your friend!!!