You should run your program with
`mpirun -n` N (executable name)
where the number of processors N is a square, say $N = p^2$.

For an $n \times n$ matrix M, we assume M to be logically divided into p^2 square submatrises, each of size $(n/p) \times (n/p)$. We denote the (i,j)-th processor by $P_{i,j}$ and (i,j)-th submatrix of M by $M_{i,j}$ where $0 \leq i, j < p$. Initially $P_{i,j}$ should generate $M_{i,j}$, so that at the begining of the given algorithm outline we assume that $M_{i,j}$ is already available to $P_{i,j}$

Cannon's Matrix Multiplication (with Non-Blocking Communication Primitives

Notation: For an integer x, $\bar{x}$ is the integer s.t. $0 \leq \bar{x} < p$ and $x \equiv \bar{x} \mod p$

Algorithm outline

Goal: Given $n \times n$ matrices A and B compute $C = AB$.

Initially $P_{i,j}$ generates $A_{i,j}$ and $B_{i,j}$

Perform cyclic shift on i-th row of A by i places so that $P_{i,j}$ contains $A_{i,\overline{i+j}}$
Perform cyclic shift on j-th column of B by j places so that $P_{i,j}$ contains $B_{\overline{i+j},j}$

(Notice that at the begining of k-th iteration, $P_{i,j}$ contains $A_{i,\overline{i+j+k}}$ and $B_{\overline{i+j+k},j}$)
For $k = 0, \ldots, p-1$ do the following in $P_{i,j}$
  If $k < p-1$
    Initiate cyclic shift on all rows of A by 1 place
    Initiate cyclic shift on all columns of B by 1 place
  If $k = 0$, compute $C_{i,j} = A_{i,\overline{i+j}} B_{\overline{i+j},j}$
  If $k > 0$, compute $C_{i,j} = C_{i,j} + A_{i,\overline{i+j+k}} B_{\overline{i+j+k},j}$
  Wait for cyclic shifts to complete so that, $P_{i,j}$ contains $A_{i,\overline{i+j+k+1}}$ and $B_{\overline{i+j+k+1},j}$)
(Notice that at the end of $p-1$-th iteration, $P_{i,j}$ contains $C_{i,j}$ is indeed the submatrix corresponding to $C = AB$)

Note

Perform the cyclic shift inside the loop using non-blocking primitives. Also, in round k, $P_{i,j}$ performs computation with $A_{i,\overline{i+j+k}}$ and $B_{\overline{i+j+k},j}$ while receiving $A_{i,\overline{i+j+k+1}}$ and $B_{\overline{i+j+k+1},j}$ from $P_{i,\overline{j+1}}$ and $P_{\overline{i+1},j}$ respectively. Hence you need to use **different buffers** to hold the data that you are computing with and the data that you are sending/receiving.