# Communicators and Topologies: Matrix Multiplication Example

Ned Nedialkov

McMaster University
Canada

CS/SE 4F03
March 2016

# Outline

# Fox's algorithm

$A$ and $B$ are $n \times n$ matrices
Compute $C = AB$ in parallel
Let $q = \sqrt{p}$ be an integer such that it divides $n$, where $p$ is the number of processes
Create a Cartesian topology with processes $(i, j)$, $i, j = 0, \ldots, q - 1$
Denote $m = n/q$
Distribute $A$ and $B$ by blocks on $p$ processes such that $A_{ij}$ and $B_{ij}$ are $m \times m$ blocks stored on process $(i, j)$

On process $(i, j)$, we want to compute

$$C_{i,j} = \sum_{k=0}^{q-1} A_{i,k} B_{k,j} = A_{i,0} B_{0,j} + A_{i,1} B_{1,j} + \cdots A_{i,i-1} B_{i-1,j}$$

$$+ A_{i,i} B_{i,j} + A_{i,i+1} B_{i+1,j} + \cdots A_{i,q-1} B_{q-1,j}$$

Rewrite this as

| stage | compute |
|---|---|
| 0 | $C_{i,j} = A_{i,i} B_{i,j}$ |
| 1 | $C_{i,j} = C_{i,j} + A_{i,i+1} B_{i+1,j}$ |
| $\vdots$ | $\vdots$ |
| | $C_{i,j} = C_{i,j} + A_{i,q-1} B_{q-1,j}$ |
| | $C_{i,j} = C_{i,j} + A_{i,0} B_{0,j}$ |
| | $C_{i,j} = C_{i,j} + A_{i,1} B_{1,j}$ |
| $\vdots$ | $\vdots$ |
| $q-1$ | $C_{i,j} = C_{i,j} + A_{i,i-1} B_{i-1,j}$ |

Each process computes in stages

**stage 0**

- process $(i,j)$ has $A_{i,j}$, $B_{i,j}$ but needs $A_{i,i}$
- process $(i,i)$ broadcasts $A_{i,i}$ across processor row $i$
- process $(i,j)$ computes $C_{i,j} = A_{i,i}B_{i,j}$

**stage 1**

- process $(i,j)$ has $A_{i,j}$, $B_{i,j}$, but needs $A_{i,i+1}$, $B_{i+1,j}$
  - shift the $j$th block column of $B$ by one block up (block 0 goes to block $q-1$)
  - process $(i,i+1)$ broadcasts $A_{i,i+1}$ across processor row $i$
- process $(i,j)$ computes $C_{i,j} = C_{i,j} + A_{i,i+1}B_{i+1,j}$

Similarly on next stages

# Example

Consider multiplying two $3 \times 3$ block matrices:

$$\begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} & B_{02} \\ B_{10} & B_{11} & B_{12} \\ B_{20} & B_{21} & B_{22} \end{bmatrix}$$

Process $(i, j)$ stores $A_{ij}$, $B_{ij}$ and computes $C_{ij}$

## Stage 0

$$
\begin{array}{cc}
\text{process} & \text{broadcasts} \\
\text{(i,i mod 3)} & \text{along row } i \\
\hline
(0,0) & A_{00} \\
(1,1) & A_{11} \\
(2,2) & A_{22}
\end{array}
\rightarrow
\begin{array}{ccc}
A_{00}, B_{00} & A_{00}, B_{01} & A_{00}, B_{02} \\
A_{11}, B_{10} & A_{11}, B_{11} & A_{11}, B_{12} \\
A_{22}, B_{20} & A_{22}, B_{21} & A_{22}, B_{22}
\end{array}
$$

Process $(i, j)$ computes

$$
\begin{array}{lll}
C_{00} = A_{00}B_{00} & C_{01} = A_{00}B_{01} & C_{02} = A_{00}B_{02} \\
C_{10} = A_{11}B_{10} & C_{11} = A_{11}B_{11} & C_{12} = A_{11}B_{12} \\
C_{20} = A_{22}B_{20} & C_{22} = A_{22}B_{21} & C_{12} = A_{22}B_{22}
\end{array}
$$

Shift-rotate on the columns of $B$:

$$
\begin{array}{lll}
A_{00}, B_{10} & A_{00}, B_{11} & A_{00}, B_{12} \\
A_{11}, B_{20} & A_{11}, B_{21} & A_{11}, B_{22} \\
A_{22}, B_{00} & A_{22}, B_{01} & A_{22}, B_{02}
\end{array}
$$

## Stage 1

| process (i,(i+1)mod 3) | broadcasts along row $i$ |
|---|---|
| $(0,1)$ | $A_{01}$ |
| $(1,2)$ | $A_{12}$ |
| $(2,0)$ | $A_{20}$ |

$$\rightarrow \begin{array}{lll} A_{01}, B_{10} & A_{01}, B_{11} & A_{01}, B_{12} \\ A_{12}, B_{20} & A_{12}, B_{21} & A_{12}, B_{22} \\ A_{20}, B_{00} & A_{20}, B_{01} & A_{20}, B_{02} \end{array}$$

Process $(i,j)$ computes

$$\begin{array}{lll} C_{00} += A_{01}B_{10} & C_{01} += A_{01}B_{11} & C_{02} += A_{01}B_{12} \\ C_{10} += A_{12}B_{20} & C_{11} += A_{12}B_{21} & C_{12} += A_{12}B_{22} \\ C_{20} += A_{20}B_{00} & C_{21} += A_{20}B_{01} & C_{22} += A_{20}B_{02} \end{array}$$

Shit-rotate on columns of $B$:

$$\begin{array}{lll} A_{01}, B_{20} & A_{01}, B_{21} & A_{01}, B_{22} \\ A_{10}, B_{00} & A_{10}, B_{01} & A_{10}, B_{02} \\ A_{21}, B_{10} & A_{21}, B_{11} & A_{21}, B_{12} \end{array}$$

## Stage 2

| process (i,(i+2)mod 3) | broadcasts along row $i$ |
|---|---|
| $(0,2)$ | $A_{02}$ |
| $(1,0)$ | $A_{10}$ |
| $(2,1)$ | $A_{21}$ |

$$\rightarrow \begin{array}{ccc} A_{02}, B_{20} & A_{02}, B_{21} & A_{02}, B_{22} \\ A_{10}, B_{00} & A_{10}, B_{01} & A_{10}, B_{02} \\ A_{21}, B_{10} & A_{21}, B_{11} & A_{21}, B_{12} \end{array}$$

Process $(i,j)$ computes

$$\begin{array}{lll} C_{00}+=A_{02}B_{20} & C_{01}+=A_{02}B_{21} & C_{02}+=A_{02}B_{22} \\ C_{10}+=A_{10}B_{00} & C_{11}+=A_{10}B_{01} & C_{12}+=A_{10}B_{02} \\ C_{20}+=A_{21}B_{10} & C_{21}+=A_{21}B_{11} & C_{22}+=A_{21}B_{12} \end{array}$$

# Implementation

```
void Setup_grid(GRID_INFO_T*  grid)
{
  int old_rank;
  int dimensions[2];
  int wrap_around[2];
  int coordinates[2];
  int free_coords[2];
  /* Set up Global Grid Information */
  MPI_Comm_size(MPI_COMM_WORLD, &(grid->p));
  MPI_Comm_rank(MPI_COMM_WORLD, &old_rank);
```

```
/* We assume p is a perfect square */
grid->q = (int) sqrt((double) grid->p);
dimensions[0] = dimensions[1] = grid->q;
/* We want a circular shift in second dimension. */
/* Don't care about first                         */
wrap_around[0] = wrap_around[1] = 1;
MPI_Cart_create(MPI_COMM_WORLD, 2, dimensions,
                wrap_around, 1, &(grid->comm));
MPI_Comm_rank(grid->comm, &(grid->my_rank));
MPI_Cart_coords(grid->comm, grid->my_rank, 2,
                coordinates);
grid->my_row = coordinates[0];
grid->my_col = coordinates[1];
```

```
  /* Set up row communicators */
  free_coords[0] = 0;
  free_coords[1] = 1;
  MPI_Cart_sub(grid->comm, free_coords,
               &(grid->row_comm));
  /* Set up column communicators */
  free_coords[0] = 1;
  free_coords[1] = 0;
  MPI_Cart_sub(grid->comm, free_coords,
               &(grid->col_comm));
}
```

```
void Fox(int n,GRID_INFO_T* grid,
         LOCAL_MATRIX_T* local_A,
         LOCAL_MATRIX_T* local_B,
         LOCAL_MATRIX_T* local_C)
{
  LOCAL_MATRIX_T*  temp_A;
  int              stage;
  int              bcast_root;
  int              n_bar;  /* n/sqrt(p) */
  int              source;
  int              dest;
  MPI_Status       status;
  n_bar = n/grid->q;
  Set_to_zero(local_C);
```

```
/* Calculate addresses for circular shift of B */
source = (grid->my_row + 1) % grid->q;
dest = (grid->my_row + grid->q - 1) % grid->q;
/* Set aside storage for the broadcast block of A */
temp_A = Local_matrix_allocate(n_bar);
```

```
  for (stage = 0; stage < grid->q; stage++) {
      bcast_root = (grid->my_row + stage) % grid->q;
      if (bcast_root == grid->my_col) {
          MPI_Bcast(local_A, 1, local_matrix_mpi_t,
                    bcast_root, grid->row_comm);
          Local_matrix_multiply(local_A, local_B,local_C
              );
        }
      else {
          MPI_Bcast(temp_A, 1, local_matrix_mpi_t,
                    bcast_root, grid->row_comm);
          Local_matrix_multiply(temp_A, local_B,local_C)
              ;
        }
      MPI_Sendrecv_replace(local_B, 1,
          local_matrix_mpi_t,
                              dest, 0, source, 0,
                              grid->col_comm, &status);
    }
}
```