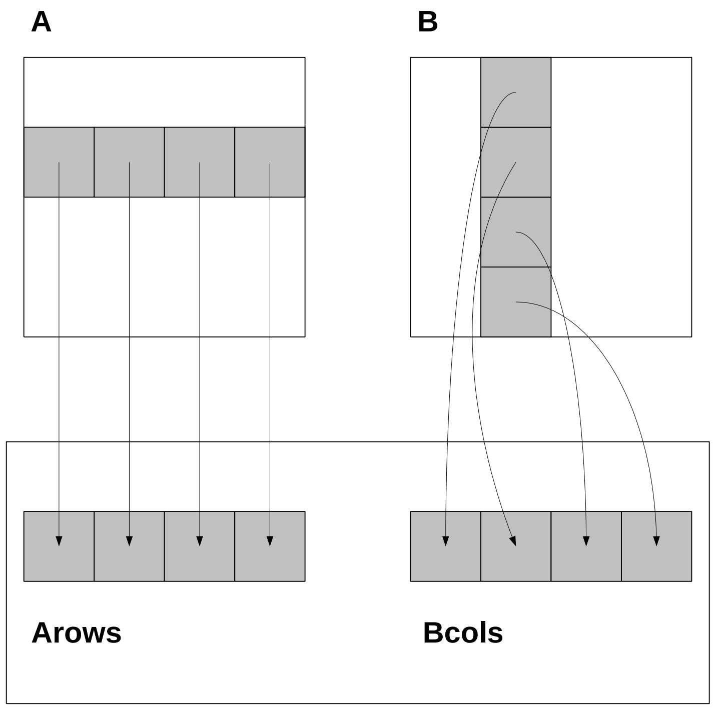
```
 PROGRAM SIMPLE
 REAL, DIMENSION(250,250):: A, B, C
 REAL, DIMENSION(250,1000):: Arows, Bcols
 INTEGER colcom, rowcom, col, row
 INTEGER rank, colrank, rowrank
 INTEGER err
 CALL MPI_INIT(ierr)
 CALL MPI_COMM_RANK(MPI_COMM_WORLD, rank);
 row = rank/4
 col = rank-row*4
 DO J=1,250
       DO I=1,250
             A(I,J)=1.0
             B(I,J)=2.0
       END DO
 END DO
 CALL MPI_COMM_SPLIT(MPI_COMM_WORLD, row, rank, rowcom, err)
 CALL MPI_COMM_SPLIT(MPI_COMM_WORLD, col, rank, colcom, err)
 CALL MPI_ALLGATHER(A, 62500, MPI_REAL, Arows, 62500,
&MPI_REAL, rowcom, err)
 CALL MPI_ALLGATHER(B, 62500, MPI_REAL, Bcols, 62500,
&MPI_REAL, colcom, err)
 DO J=1,250
       DO I=1,250
             C(I,J)=0.0
             ind1=1
             ind2=J
             DO K=1,1000
                   C(I,J)=C(I,J)+Arows(I,K)*Bcols(ind1,ind2)
                   IF(ind1.LT.250) THEN
                         ind1=ind1+1
                   ELSE
                         ind1=1
                         ind2=ind2+250
                   END IF
             END DO
       END DO
 END DO
 CALL MPI_COMM_FREE(rowcom, err)
 CALL MPI_COMM_FREE(colcom, err)
 CALL MPI_FINALIZE(err)
 END
```

This code is in Fortran 77 with calls to MPI routines. It is supposed to be executed by all 16 processes making up the parallel program. Each process locally contains one 250×250 block of global arrays A, B, and C of the source HPF program. A logical 4×4 process grid is formed from the 16 participating processes, and each process gets its coordinates row and col in the grid. In order to compute its block of the resulting matrix *C*, the process needs blocks of matrix *A* from its horizontal neighbours in the 4×4 process grid, and blocks of matrix *B* from its vertical neighbours (see Figure 4.3). The necessary communication is achieved as follows.

Firstly, communicators are created for horizontal communication, one for each row of the 4×4 process grid. The MPI_COMM_SPLIT routine performs this operation returning the new communicators in `rowcom`. Then, a call to `MPI_COMM_SPLIT` creates communicators for columns of the process grid and returns them in `colcom`.

After that, all processes call the `MPI_ALLGATHER` routine supplying `rowcom` as a communicator argument. This call performs in parallel 4 collective gather-to-all communication operations, each on its row of the process grid. As a result, each process has in its local array `Arows` a copy of the corresponding row of blocks of matrix *A*.

The next call to `MPI_ALLGATHER` with `colcom` as a communicator argument also performs in parallel 4 collective gather-to-all communication operations, but on columns of the process grid. During the execution of this call, each process gathers in local array `Bcols` copies of blocks of the corresponding column of blocks of matrix *B*. Note that array `Bcols` stores the blocks horizontally, as a row of blocks, not in the form of column of blocks as they are normally stored (see Figure below). Therefore, the inner `DO` loop, which computes an element of the resulting matrix *C*, must recalculate indices of `Bcols` elements to make successive iterations of the loop refer successive elements of the corresponding column of matrix *B*.


A
B
Arows
Bcols