Solution 10

Solution 10

c

```
#include <stdio.h>
#include <mpi.h>

int main (int argc, char* argv[]) {

    int rank, size;

    MPI_Datatype diag;

    /* MPI Initialization */
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);

    /* Matrix initialization */
    int matrix[size][size];

    for (int i=0; i<size; i++)
        for (int j=0; j<size; j++)
            if (i==j) matrix[i][j] = rank;
            else matrix[i][j] = 0;

    /* Print rank 0 matrix (should be filled with 0s) */
    if (rank == 0) {
    printf("Rank 0 matrix before communication:\n");
       for (int i=0; i<size; i++) {
           for (int j=0; j<size; j++)
               printf("%d ",matrix[i][j]);
           printf("\n");
           }
       printf("\n");
       }
```

```
    /* Diagonal datatype vector creation */
    MPI_Type_vector(size,1,size+1,MPI_INT,&diag);
    MPI_Type_commit(&diag);

    /* Communication: rank 0 gathers all the diagonals from the other ranks and stores them in the row corresponding to the
     * sending rank. Note that 1 "diag" type is sent and size MPI_INT types are received, so that the values can be stored
     * contiguously in the receiving matrix */
    MPI_Gather(matrix,1,diag,matrix,size,MPI_INT,0,MPI_COMM_WORLD);

    /* Print rank 0 matrix after communication (each element should be its row number) */
    if (rank == 0) {
    printf("Rank 0 matrix after communication:\n");
       for (int i=0; i<size; i++) {
           for (int j=0; j<size; j++)
               printf("%d ",matrix[i][j]);
           printf("\n");
           }
       printf("\n");
       }

    /* Remember to free the datatype! */
    MPI_Type_free(&diag);

    MPI_Finalize();
    return 0;
    }
```

FORTRAN

```
program diagonal


use mpi

implicit none

    INTEGER :: rank, size, ierr
    INTEGER :: i,j
    INTEGER, DIMENSION (:,:), ALLOCATABLE :: matrix
    INTEGER :: diag

    !$ MPI Initialization
    CALL MPI_Init(ierr)
    CALL MPI_Comm_size(MPI_COMM_WORLD,size,ierr)
    CALL MPI_Comm_rank(MPI_COMM_WORLD,rank,ierr)

    !$ Matrix initialization
```

```
    ALLOCATE (matrix(size,size))

    DO j=1,size
        DO i=1,size
            IF (i.EQ.j) THEN
            matrix(i,j) = rank
            ELSE
            matrix(i,j) = 0
            ENDIF
        ENDDO
    ENDDO

    !$ Print rank 0 matrix (should be filled with 0s)
    IF (rank.EQ.0) THEN
       WRITE (*,*) 'Rank 0 matrix before communication:'
       DO i=1,size
           PRINT *, matrix(i,:)
       ENDDO
    ENDIF

    !$ Diagonal datatype vector creation
    CALL MPI_Type_vector(size,1,size+1,MPI_INT,diag,ierr)
    CALL MPI_Type_commit(diag,ierr)

    !$ Communication: rank 0 gathers all the diagonals from the other ranks and stores them in the column corresponding to the
    !$ sending rank. Note that 1 "diag" type is sent and size MPI_INT types are received, so that the values can be stored
    !$ contiguously in the receiving matrix
    CALL MPI_Gather(matrix,1,diag,matrix,size,MPI_INT,0,MPI_COMM_WORLD,ierr)

    !$ Print rank 0 matrix after communication (each element should be its column number)
    IF (rank.EQ.0) THEN
       WRITE (*,*) ' '
       WRITE (*,*) 'Rank 0 matrix after communication:'
       DO i=1,size
           PRINT *, matrix(i,:)
       ENDDO
    ENDIF

    !$ Remember to free the datatype!
    CALL MPI_Type_free(diag,ierr)
    DEALLOCATE(matrix)

    CALL MPI_Finalize(ierr)

end program diagonal
```

‹ Exercise 10 up Exercise 11 ›