Solution 8

Solution 8

C

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

#define N 50

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

int i, my_id, num_procs, num_elem;
int array[N],array_final[N];
int *array_recv;

MPI_Init(&argc, &argv); /* starts MPI */
MPI_Comm_rank (MPI_COMM_WORLD, &my_id); /* get current process id */
MPI_Comm_size (MPI_COMM_WORLD, &num_procs); /* get number of processes */

/* proc 0 initializes the principal array */
if(my_id == 0) {
for(i=0;i<N;i++) array[i]=i ;
}

num_elem = N/num_procs;
if(my_id < (N%num_procs)) num_elem = num_elem +1;

array_recv = (int*) malloc(num_elem*sizeof(int));

/* in case that N is a multiple of the number of MPI tasks, the same number of
* elements is sent to (and received from) root process to others, so
* mpi_scatter and mpi_gather are called */

printf("num_elem = %d my_id = %d\n",num_elem,my_id);

if((N%num_procs) == 0 ){
MPI_Scatter(array,num_elem, MPI_INT,array_recv,num_elem, MPI_INT, 0, MPI_COMM_WORLD);
```

```
for(i=0;i<num_elem;i++){
printf("my_id = %d , Received elements : %d\n", my_id,array_recv[i]);
array_recv[i]= array_recv[i]+my_id;
}

MPI_Gather(array_recv,num_elem,MPI_INT,array_final,num_elem,MPI_INT,0,MPI_COMM_WORLD);

if(my_id == 0){
printf("N is multiple of num_procs, mod(N,num_procs)= %d\n", N%num_procs);
for(i=0;i<N;i++){
printf("i = %d ; Array_final : %d\n", i,array_final[i]);
}
}
}

else{

/* in case that N is not a multiple of the number of MPI tasks,
* mpi_scatterv and mpi_gatherv have to be used
*/

int *sendcount, *displs;
sendcount = (int*) malloc(num_procs*sizeof(int));
displs = (int*) malloc(num_procs*sizeof(int));

displs[0] = 0;
for(i=0;i<num_procs;i++) sendcount[i] = N/num_procs;
if(0<(N%num_procs)) sendcount[0] = N/num_procs +1;

for(i=1;i<num_procs;i++){
if(i<(N%num_procs)) sendcount[i] = N/num_procs +1;
displs[i] = displs[i-1] + sendcount[i-1];
}

printf("\n");
if(my_id == 0){
printf("sendcount:");
for(i=0;i<num_procs;i++) printf("%d \t", sendcount[i]);
printf("\n");
printf("displs:");
for(i=0;i<num_procs;i++) printf("%d \t", displs[i]);
}
printf("\n");

MPI_Scatterv(array,sendcount,displs,MPI_INT, array_recv,num_elem, MPI_INT, 0, MPI_COMM_WORLD);

for(i=0;i<num_elem;i++){
printf("my_id = %d, Received elements: %d\n ", my_id,array_recv[i]);
array_recv[i]= array_recv[i]+my_id;
}
```

```
printf("\n");

MPI_Gatherv(array_recv,num_elem,MPI_INT,array_final,sendcount,displs,MPI_INT,0,MPI_COMM_WORLD);

if(my_id == 0){
printf("N is multiple of num_procs, mod(N,num_procs)= %d\n", N%num_procs);
printf("Array_final : ");
for(i=0;i<N;i++){
printf("%d \t",array_final[i]);
}
}
free(sendcount);
free(displs);
}
free(array_recv);
MPI_Finalize();
return 0;
}
```

FORTRAN

```
PROGRAM main
 IMPLICIT NONE
 include 'mpif.h'

 INTEGER, parameter::N=50
 INTEGER ::my_id, num_procs, ierr
 INTEGER ::i, num_elem
 INTEGER, dimension(N)::array, array_final
 INTEGER, allocatable, dimension(:)::array_recv
 INTEGER, allocatable:: sendcount(:), displs(:)

 CALL MPI_INIT( ierr )
 CALL MPI_COMM_RANK( MPI_COMM_WORLD, my_id, ierr )
 CALL MPI_Comm_size ( MPI_COMM_WORLD, num_procs, ierr )

 ! proc 0 initializes the principal array
 IF( my_id .eq. 0) THEN
   DO i=1,N
     array(i)= i
   END DO
 END IF

 num_elem= N/num_procs
 IF (my_id < MOD(N,num_procs)) THEN
    num_elem = num_elem +1
 ENDIF

 ALLOCATE(array_recv(num_elem))
```

```
 ! in case that N is a multiple of the number of MPI tasks, the same number of
 ! elements is send to (and received from) by root process to others, so
 ! mpi_scatter and mpi_gather are called
IF ( MOD(N,num_procs) .eq. 0 ) THEN
    CALL MPI_SCATTER(array, num_elem, MPI_INTEGER, array_recv, num_elem, &
MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

    WRITE(*,*) "my_id", my_id, "elementi ricevuti:", array_recv(1:num_elem)

    DO i=1,num_elem
      array_recv(i)= array_recv(i)+my_id
    END DO

    CALL MPI_GATHER(array_recv, num_elem, MPI_INTEGER, array_final, &
num_elem, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

    IF( my_id .eq. 0) THEN
       WRITE(*,*) "N is multiple of num_procs, mod(N,num_procs)= ", &
mod(N,num_procs)
       WRITE(*,*) " array finale: ", array_final(:)
    END IF

ELSE

!in case that N is not a multiple of the number of MPI tasks,
!mpi_scatterv and mpi_gatherv have to be used

    ALLOCATE(sendcount(num_procs),displs(num_procs))

    displs(1) = 0
    sendcount=N/num_procs
    if(0<mod(N,num_procs)) sendcount(1)=N/num_procs+1

    DO i=2,num_procs
      if( (i-1) < mod(N,num_procs) ) sendcount(i) = N/num_procs +1
      displs(i) = SUM(sendcount(1:i-1))
    END DO

IF (my_id .eq. 0 ) THEN
       WRITE(*,*) "sendcount: ", sendcount
       WRITE(*,*) "displs: ", displs
    END IF

    CALL MPI_SCATTERV(array, sendcount, displs, MPI_INTEGER, &
array_recv, num_elem, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

    WRITE(*,*) "my_id", my_id, "elementi ricevuti:", array_recv(1:num_elem)

    DO i=1,num_elem
      array_recv(i)= array_recv(i)+my_id
```

```
    END DO

    CALL MPI_GATHERV(array_recv, num_elem, MPI_INTEGER, array_final, &
sendcount, displs, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

    IF( my_id .eq. 0) THEN
       WRITE(*,*) "N is not multiple of num_procs, mod(N,num_procs)= ", &
mod(N,num_procs)
       WRITE(*,*) " array finale: ", array_final(:)
    END IF
ENDIF

DEALLOCATE(array_recv)
CALL MPI_FINALIZE(ierr)
END PROGRAM main
```

‹ Exercise 8 up Exercise 9 ›