Exercise 9

Exercise 9

Each process initializes a one-dimensional array by giving to all the elements the value of its rank+1. Then the root process (task 0) performs two reduce operations (sum and then product) to the arrays of all the processes. Finally, each process generates a random number and root process finds (and prints) the maximum value among these random values.

Modify the code to perform a simple scalability test using MPI_Wtime. Notice what happens when you go up with the number of processes involved.

HINTS:

	C
MPI_REDUCE	`int MPI_Reduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)`
MPI_INIT	`int MPI_Init(int *argc, char ***argv)`
MPI_COMM_SIZE	`int MPI_Comm_size(MPI_Comm comm, int *size)`
MPI_COMM_RANK	`int MPI_Comm_rank(MPI_Comm comm, int *rank)`
MPI_FINALIZE	`int MPI_Finalize(void)`
MPI_WTIME	`double MPI_Wtime(void)`

	FORTRAN

MPI_REDUCE	`MPI_REDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM, IERROR)`
	`<type> SENDBUF(*), RECVBUF(*)`
	`INTEGER COUNT, DATATYPE, OP, ROOT, COMM, IERROR`

MPI_INIT	`MPI_INIT(IERROR)` `INTEGER IERROR`
MPI_COMM_SIZE	`MPI_COMM_SIZE(COMM, SIZE, IERROR)` `INTEGER COMM, SIZE, IERRO`
MPI_COMM_RANK	`MPI_COMM_SIZE(COMM, SIZE, IERROR)` `INTEGER COMM, SIZE, IERROR`
MPI_FINALIZE	`MPI_FINALIZE(IERROR)` `INTEGER IERROR`
MPI_WTIME	`MPI_WTIME()`

‹ Solution 8 up Solution 9 ›