Master/Slave
by
William Gropp and Ewing Lusk

1. Exercise: Master/slave programs in MPI

Message passing is well-suited to handling computations where a task is divided up into subtasks, with most of the processes used to comput the subtasks and a few processes (often just one process) managing the tasks. The manager is called the "master" and the others the "workers" or the "slaves".

In this example, you will begin to build an Input/Output master/slave system. This will allow you to relatively easily arrange for different kinds of input and output from your program, including

* Ordered output (process 2 after process 1)
* Duplicate removal (a single instance of "Hello world" instead of one from each process)
* Input to all processes from a terminal

This will be accomplised by dividing the processes in MPI_COMM_WORLD into two sets - the master (who will do all of the I/O) and the slaves (who will do all of their I/O by contacting the master). The slaves will also do any other computation that they might desire; for example, they might implement the Jacobi iteration in the first exersise set.

For the first exercise, divide the processors into two communicators, with one processor the master and the others the slave. The master should accept messages from the slaves (of type MPI_CHAR) and print them in rank order (that is, first from slave 0, then from slave 1, etc.). The slaves should each send 2 messages to the master. For simplicity, Have the slaves send the messages

Hello from slave 3
Goodbye from slave 3

(with appropriate values for each slave). You may assume a maximum length message of 256 characters.

For this first exercise, keep the code simple. Do not use intercommunicators. Also, you'll find that you use the new communicator for the slaves only to get the rank of the slave in its communicator. Note that if the slaves were also computing, they would use that new communicator instead of MPI_COMM_WORLD.

You may want to use these MPI routines in your solution:
MPI_Comm_split MPI_Send MPI_Recv

2. Exercise: A simple output server

Modify the output master to accept three types of messages from the slaves. These types are

* Ordered output (just like the previous exercise)
* Unordered output (as if each slave printed directly)
* Exit notification (see below)

The master continues to receive messages until it has received an exit message from each slave. For simplicity in programming, have each slave send the messages

Hello from slave 3
Goodbye from slave 3

with the ordered output mode, and

I'm exiting (3)

with the unordered output mode.

You may want to use these MPI routines in your solution:
MPI_Comm_split MPI_Send MPI_Recv