MPI Workshop Notes

MPI is a standard and portable message passing parallel library used for distributed memory computing. As individual processes have their own private memory stack, processes have to explicitly send messages to each other to communicate. MPI has a huge catalogue of functions/subroutines for the C/C++ and Fortran programming languages.

Below are compilation (and link) commands for C/C++ and Fortran programs:

C language: `mpicc program.c -o program`
C++ language: `mpiCC program.cc -o program`
Fortran 90: `mpif90 program.f90 -o program`
Fortran 77: `mpif77 program.f -o program`

The MPI compilation commands are simply wrappers which execute the underlying compiler command, e.g. `icc`. The Grace cluster uses Platform MPI version 8.1 which is a highly optimised and tuned library for parallel computing clusters. The module files for MPI are:

GNU: `mpi/platform/gcc/8.2.1`
PGI: `mpi/platform/pgi/8.2.1` `pgi/12.5`
Intel: **`mpi/platform/intel/8.2.1`** **`icc/intel/12.1`**

It is recommended to use the Intel version as it provides very good performance. To load the modules, type:

```
module load mpi/platform/intel/8.2.1 icc/intel/12.1
```

Log on to the Grace login node `grace.uea.ac.uk` and download the MPI programs:

```
wget http://grace-head00.uea.ac.uk/grace-docs/mpi_examples.tar
tar -xvf mpi_examples.tar
```

All parallel jobs must be submitted to the LSF job submission system. There is an example script `mpi_job.bsub` for this workshop, and the command to submit is:

```
bsub < mpi_job.bsub
```

Please remember to adjust the job submission script for your program, e.g. changing the program name. There are four parallel queues available for MPI jobs which are listed in Table 1-1.

Queue name	Slots available	Wall time	Slots per job	Priority
`debug-ib`	200	20 minutes	200	20
`short-ib`	404	5 hours	404	15
`medium-ib`	470	24 hours	336	10
`long-ib`	404	5 days	268	5

Table 1-1: Infiniband parallel queues

The postfix `ib` is to denote Infiniband queues, where Infiniband is a type of high performance switch used for parallel programs.

Practical 1 - Hello World

The first practical will simply involve compiling and submitting a job to the cluster. Follow the instructions given below:

1. Compile either `hello_world.c` or `hello_world.f90` using the command:

   ```
   mpicc hello_world.c -o hello_world
   mpif90 hello_world.f90 -o hello_world
   ```

2. Type the command `ldd hello_world` which should show you that it is dynamically linked against the MPI libraries;
3. Submit the job using the command:

   ```
   bsub < mpi_job.bsub
   ```

   Examine the output in the file `parallel.out` - the print statements should show that the processes execute in a completely arbitrary manner.

Practical 2 - Sending and Receiving

This practical will demonstrate using the MPI send and receive functions. If you are using Fortran, the subroutine prototypes are given below:

```
MPI_SEND( BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERR )
      <type> BUF(*)
      INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERR

MPI_RECV( BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS, IERR )
      <type> BUF(*)
      INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE),
      IERROR
```

Edit either `send_recv.c` or `send_recv.f90` and follow the instructions below:

1. Add the `MPI_Send` function call in the first if block with the correct parameters to send to rank zero;
2. In the loop in the else block, add the `MPI_Recv` call with the correct parameters to receive the message;
3. Compile with the the `mpicc` or `mpif90` commands;
4. Edit the `mpi_job.bsub` file and include the name of the MPI executable after the `mpirun` command;
5. Submit the job using:

   ```
   bsub < mpi_job.bsub
   ```

6. What does this program do?

Practical 3 - Numerical Integration

This exercise will involve solving the bounded integral numerically:

$$\int_a^b f(x)dx$$

If you are using Fortran, the subroutine prototype is given below:

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

Edit either the file `integration.c` or `integration.f90` and follow the instructions below:

1. Add the `MPI_Reduce` call after the calculations;
2. Compile the program;
3. Include the name of the executable in the job submission script and submit it to LSF;
4. Validate the result for:

$$\int_2^8 x^2 dx = \left[\frac{x^3}{3}\right]_2^8 = \frac{7000}{3}$$

Practical 4 - Parallel Write

This exercise will involve writing data to a file in parallel using the MPI-IO library. Edit either the file `mpi_write.c` or `mpi_write.f90` and follow the instructions below:

1. Add the `MPI_File_set_view` function/subroutine call using the `disp` variable just after this variable has been assigned. Use the `MPI_INFO_NULL` argument for the `MPI_Info` parameter;
2. Add the `MPI_File_write` call with the `MPI_STATUS_IGNORE` argument for the `MPI_Status` parameter;
3. Compile the program ;
4. Submit the job, ensure the output file `parallel.out` does not contain any errors and check the file `run01.dat` has been created.

Practical 5 - Parallel Read

This exercise will involved reading data from a file in parallel using the MPI-IO library. Edit either the file `mpi_read.c` or `mpi_read.f90` and follow the instructions below:

1. Add the `MPI_File_open` call with `MPI_MODE_RDONLY` for the opening mode;
2. Add the `MPI_File_set_view` call using the `disp` variable;
3. Add the `MPI_File_read` call;
4. Compile the program;
5. Submit the job, ensure the output file `parallel.out` does not contain any errors. Examine the output file and check that the program works. This should read the numbers 1 to 40 using four processes.

Full MPI Specification

The full MPI specification version 2.2 can be downloaded from:

http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf

Feedback

When you have completed the workshop, please remember to complete the course questionnaire at:

http://rscs.uea.ac.uk/events/feedback

All feedback is greatly appreciated!