MPI example for Abacus1

gavin@epcc.ed.ac.uk

Example Fortran/MPI code

```
        PROGRAM hello
! Warning: this code contains intentional bugs: write your own
        INCLUDE "mpi.h“
        INTEGER IERROR, RANK, NUM_TASKS
        CALL MPI_INIT(ERROR)
        CALL MPI_COMM_RANK(MPI_COMM_WORLD, RANK, ERROR)
        CALL MPI_COMM_SIZE(MPI_COMM_WORLD, SIZE, ERROR)
        PRINT *,'HELLO FROM CORE ',RANKID,' OUT OF ‘,NCORES,’ TASKS’
        CALL MPI_FINALISE(ERROR)
        END
```

Compilation

```
# loads compilers and MPI for both Intel and GNU
      module load intel/tools-16.0.109
# to use Intel (fortran)
      mpiifort hello.f –o hello
# to use GNU (fortran)
      mpif90 hello.f –o hello
```

SLURM Batch Script for Intel

- #!/bin/bash
- #SBATCH --partition=curso
- #SBATCH --job-name=mpi_hello_world
- #SBATCH -o slurm.%N.%j.out
- #SBATCH -e slurm.%N.%j.err
- #SBATCH --nodes=1
- #SBATCH --ntasks=28
- #SBATCH --ntasks-per-node=28
- #SBATCH --ntasks-per-core=1
- #SBATCH --exclusive
- #SBATCH --time=00:05 # time format (HH:MM)
- export OMP_NUM_THREADS=1
- cd $SLURM_SUBMIT_DIR
- module load intel/tools-16.0.109
- mpiexec.hydra -n 28 -ppn 28 ./hello

SLURM Batch Script for GNU

- #!/bin/bash
- #SBATCH --partition=curso
- #SBATCH --job-name=mpi_hello_world
- #SBATCH -o slurm.%N.%j.out
- #SBATCH -e slurm.%N.%j.err
- #SBATCH --nodes=1
- #SBATCH --ntasks=28
- #SBATCH --ntasks-per-node=28
- #SBATCH --ntasks-per-core=1
- #SBATCH --exclusive
- #SBATCH --time=00:05 # time format (HH:MM)
- export OMP_NUM_THREADS=1
- cd $SLURM_SUBMIT_DIR
- module load intel/tools-16.0.109
- mpirun -n 28 -ppn 28 ./hello

Job submission

```
sbatch mpi_hello_world.sh
```

MPI exercises

gavin@epcc.ed.ac.uk

Exercise 1: Hello World

- Write an MPI code which prints to the screen:
  - Hello world, I'm rank 0 out of a total of 4 tasks
  - Hello world, I'm rank 1 out of a total of 4 tasks
  - Hello world, I'm rank 2 out of a total of 4 tasks
  - Hello world, I'm rank 3 out of a total of 4 tasks
- Why is your initial attempt random?!

Exercise 2: Ping Pong

- Write an MPI code for 2 tasks
- The first task sends a message to the second task
  - The message can be a single integer
- The second task receives this message
- The second task changes the message somehow
  - Add 99 to the original message, for instance
- The second task sends the message back to the first task
- The first task receives the message
- The first task prints this new message to the screen

Exercise 4: Ring Code

- Write an MPI code which sends a message around a ring
- Each MPI task receives from the left and passes that message on to the right
- Every MPI task sends a message
  - Its own rank ID
- How many steps are required for the message to return ‘home’.

- **Update the code so that each MPI Tasks takes the incoming message (an integer), and adds it to a running total**
  - **What should the answer be on MPI Task 0**
  - **Is the answer the same on all tasks?**

Exercise 5: Collective Communications

- Use a Collective Communications routine to produce the same result as your Ring Code so that only Task 0 ‘knows’ the answer.
- Now change the Collective Communications call so all tasks ‘know’ the answer.
- Can you think of another way to implement the collective communications routine than the Ring Code?
- Compare the execution times for both the Ring Code and the associated Collective Communication routine.
  - Which is faster?
  - Why do the times change so much when you run it again and again?
  - What happens when the size of the message goes from 1 MPI_INT to 10000 MPI_INTs?!

Exercise 6: Virtual topology

- Take the ‘Ring Code’ and determine neighbours using the virtual topology method
  - Construct the loop
  - Determine the Rank ID of who sends to you
  - Determine the Rank ID of who you send the message to