MPI Tutorials

Credits

The material in these tutorials was adapted from the followinfg source:
Link: http://einspem.upm.edu.my/INSPEM/MPI/knowing%20mpi/exercise.html

Objective

The aim of these case studies is to demonstrate the utility of parallel programming by rewriting a serial code into a parallel, MPI code that performs the same task.

Source code

The source code for the case studies can be downloaded from the course web site. For each example, versions in both C and FORTRAN are given.

Method

For each study you will be given the serial version of the code and a partial version of the MPI implementation. You should perform the following procedure:

1. Compile the serial version, run it and make a note on how much time is required. Keep a copy of the output.
2. Edit as necessary the MPI version. Compile and run it.
3. Check the parallel version by comparing the output to that obtained from the serial program.
4. Measure the performance either by inserting MPI_Wtime functions or by shell commands.
5. Re-run the MPI program with different numbers processors to test scalability.

Tutorials

1. Matrix multiplication
2. Prime number generation

Tutorial 1. Matrix Multiplication

Files

Serial program: ser_mm.f90, ser_mm.c
Parallel: mpi_mm.f90, mpi_mm.c

Description

This program performs a basic matrix multiplication. To change the dimensions of the matrices, and hence the execution time, you can change the nra,nrb and nrc parameters.

Proposed Parallelisation Strategy

Use a master/worker strategy, where the master sends the A matrix and a portion of the B matrix to each worker. After the multiplication, each worker sends back its portion of the result matrix back to master.

Thus, in detail the master should:

1. Initialise the A and B matrices.
2. Divide the columns of B amongst the workers and issue four MPI_Sends to each worker i as follows:
    a. send the next column position ("offset") in B to send to the worker i
    b. send the number of columns
    c. send the whole A matrix
    d. send the columns from matrix B
3. The master should then issue three MPI_Recvs from each worker as follows:
    a. the offset originally passed
    b. the number of columns
    c. the calculated results in the C matrix (using the passed offset)
4. Print out the results

Each worker should:

1. Issue 4 MPI_Recvs to receive the corresponding data from the Master
2. Matrix multiply A and the columns of B
3. Send back the corresponding data to the Master.

For the C version of the program we should send **rows** instead of columns.

Notes

In practice, on an HPC system an optimised library routine (from e.g. ESSL, MKL, LAPACK, Scalapack, etc) should be used so this example is only for demonstration purposes. Passing the whole A matrix in MPI_send, in particular is unlikely to be efficient for large matrices.

Tutorial 2. Prime number generation

Files

Serial: ser_prime.f90, ser_prime.c

Parallel: mpi_prime.f90, mpi_prime.c

Description

The aim here is to find all the prime numbers upto a certain limit and print out the maximum. Here we use a “Brute force” approach to prime generation. Since each prime can be computed independently, should be easy to parallelise by simply dividing the main loop over the n tasks. In fact the only MPI communication commands required are MPI_Reduce collectives to collect the number of prime numbers and max primes found from each task.

Proposed parallelisation strategy

Recall that we only need to check odd numbers. So we loop over odd numbers upto the limit. Try a structure such as:

```
limit=2500000 ! max number to check up to
start=rank*2 +1
do n=start, limit, ntasks
...if  isprime(n) nprimes=nprimes+1
endo
! find sum of primes from all tasks
mpi_reduce(nprimes, sum_nprimes, …,MPI_SUM,..)
! find biggest prime from all tasks
mpi_reduce(n,maxprime,....,MPI_MAX..)
```

Notes

The template solution uses two loops, one for rank 0 and one for the other ranks, but this is not essential.

The loop construct

```
do start=myrank, end=some limit, stride=no. of ranks
```

is very useful for data parallelisations such as this.