Computing Lab #8: First MPI program

Due: **Thursday, before class** (the next lab will be on Thursday, and will build on this example)

Programming language: Write, compile, and run a Fortran, C or C++ program. Your choice! This lab assumes the C programming.

Points: 10

In this lab you will write your first message passing interface (MPI) program. You will write a simple optimization algorithm and parallelize it with MPI.

I. TASK #1: MPI HELLO WORLD (3 POINTS)

In this first part, you will write, compile and run a simple MPI program on your laptop and control the number of *processes* (not threads!) to use.

1. Write the following code

```c
#include <stdio.h>
#include <mpi.h> // include mpi library header


int main(int argc, char **argv)
{
  int rank, size;

  MPI_Init (&argc, &argv);      // initializes MPI
  MPI_Comm_rank (MPI_COMM_WORLD, &rank); // get current MPI-process ID. 0, 1, ...
  MPI_Comm_size (MPI_COMM_WORLD, &size); // get the total number of processes
  printf( "Hello world from process %d of %d\n", rank, size );
  MPI_Finalize(); // programs should always perform a "graceful" shutdown
  return 0;
}
```

2. Compile this code using an MPI compiler (e.g. mpicxx).

```
>>> mpicc -Wall -o helloMPI helloMPI.c
```

3. Launch your program using mpirun (please read the documentation, e.g. "man mpirun", to find out more). Try three separate cases: launch 1, 4, and 16 processes, and confirm you get the expected output. To launch 4 processes, you should do

```
>>> mpirun -np 4 helloMPI
```

II. TASK #2: BRUTE FORCE OPTIMIZATION (4 POINTS)

In this second part, you will develop a simple "optimization" algorithm. In its simplest form, numerical optimization is the study of finding a maximum (or minimum) of a function. The applications are endless, since many questions seek to know where the maximum is. Things people like to maximize may include profits, voter turnout, engine efficiency, goals in a soccer game, gravitational waveform "alignments", etc. Of course these are complicated tasks and will certain require a model. Here we will consider optimizing (finding the maximum) for the function

$$\mathcal{L}_1(x_1, x_2) = \exp\left(-\left(1 - x_1\right)^2 - 100\left(x_2 - x_1^2\right)^2 \right) . \quad (1)$$

This is exactly the function which appeared in your last homework. What is the maximum value possible for Eq. (1)? This can be done without any codes.

To start, you can use my serial optimization code:

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>

// will sample any dim-dimensional box of size [a,b]^dim
// the random point is stored in the array x
void sample_rand(const double a, const double b ,const int dim, double *x) {

  for(int i=0;i<dim;++i) {
    double tmp = ((double) rand())/((double) RAND_MAX);
    x[i] = (b-a)*tmp + a;
  }

}

double logL(double x1,double x2) {
  return -(1.-x1)*(1.-x1) -100.*(x2-x1*x1)*(x2-x1*x1);
}

int main(int argc, char **argv)
{

  long N = atol( argv[1] );
  srand(time(NULL)); // each MPI process gets a unique seed

  const int dim = 2;
  double x[dim];  // array of random numbers

  // search for the function's maximum
  double max = -1;
  for(long int i=1; i<=N; ++i) {
    sample_rand(-5.,5.,dim,x);
    double f_i = exp(logL(x[0],x[1]));
    if( f_i > max) {
      max = f_i;
    }
  }

  printf("local max = %1.5e\n",max);
  return 0;
}
```

- (**part a**) Next, you will "merge" your hello-world MPI code (task 1) and the serial code I provided above. To your hello-world MPI code, add
   1. Code to randomly sample a 2-dimensional box centered on the origin with sides of length 10. That is
$$x_1, x_2 \in [-5, 5] \,,$$
   2. Code to evaluate the function Eq. (1) at this point.
   3. Make sure your MPI code allows you to pass input as an integer N and each process of your program loops over N random samples (that is, your for-loop should stop after N iterations. So if you run 2 processes

there are effectively $2 \times N$ random samples).

4. So that each MPI process gets a unique seed, you should replace the srand command in the serial version to be "srand(time(NULL) * rank);"
5. Write code to display the following information (i) the maximum value of the function you find after N random samples, (ii) the number of for loop iterations and (iii) the process ID (as you did in the hello world problem. If you have P processes running, you should have P outputs – one from each process.

- Compile and run your MPI program using 4 processes.

III. TASK #3: RUN YOUR CODE ON STAMPEDE2 (3 POINTS)

1. Log into Stampede2
2. Use git to move the program you wrote in task 2 to Stampede
3. Compile the program.

```
>>> mpicc -Wall -o optimizer code.c
```

4. Create a batch script file "stampedeMPI.job" (to run this program, you will be using 128 cores in total, split across two separate nodes)

```
#!/bin/bash
#----------------------------------------------------
#SBATCH -J myjob              # Job name
#SBATCH -o myjob.o%j          # Name of stdout output file
#SBATCH -e myjob.e%j          # Name of stderr error file
#SBATCH -p development        # Queue (partition) name
#SBATCH -N 2                  # Total # of nodes lets use 2
#SBATCH -n 128                # Total # of mpi tasks (64 per node)
#SBATCH -t 00:05:00           # Run time (hh:mm:ss)
#SBATCH --mail-user=YOUR-XSEDE-NAME@umassd.edu
#SBATCH --mail-type=all       # Send email at begin and end of job

echo "Hello"
ibrun ./optimizer
```

5. Submit the batch script file to the queue with

```
>>> sbatch stampedeMPI.job
```

6. Check the status of your job

```
>>> squeue -u YOURNAME # YOURNAME -> your stampede2 name
```

7. Once your job finishes running, look at the output file and make sure it has the output from 128 different processes.

IV. UPLOAD YOUR WORK TO GIT

Congratulations! You've completed the lab. Now lets upload your work to git.

Follow the same series of steps as you've done for the last few labs. Place all work for this lab (task 1: HelloWorld code and its output, task 2: optimization code and its output, task 3: output and error files from stampede) into a folder called "Lab8" and push it to bitbucket.