Mathematics Department, University of Massachusetts Dartmouth

High Performance Scientific Computing
MTHEAS 520 – Section 01 – Spring 2015
Project 2
Running with MPI
Due April 14, 2015

There are two major tasks for this project:

a. Run a simple job on the cluster using MPI (simulation of the heat equation)
b. Take some serial code and make it run in an embarrassingly parallel fashion using MPI

We will be doing all of these on the cluster, so you may want to familiarize yourself with logging into the cluster and running jobs. (See the `job-submission` repository on the class git server.)

Please understand that the cluster is a shared resource – do not hog several nodes for yourself for an extended period of time.

1 Heat flow simulation

The *heat equation* is the colloquial term for the partial differential equation

$$u_t = u_{xx} + u_{yy}, \qquad u(x, y, 0) = f(x, y),$$

where $u(x, y, t)$ is a function of space $x \in [0, L]$ and time $t > 0$ (e.g., u is temperature in a region), subscripts denote partial differentiation with respect to the subscripted variable, and $f(x, y)$ is a known function representing the initial value of u.
An example of the solution $u(t, x, y)$ is shown in Figure 1 for $t = 0$ and $t = 1$. In this problem, you will run a simulation of the heat equation using MPI.

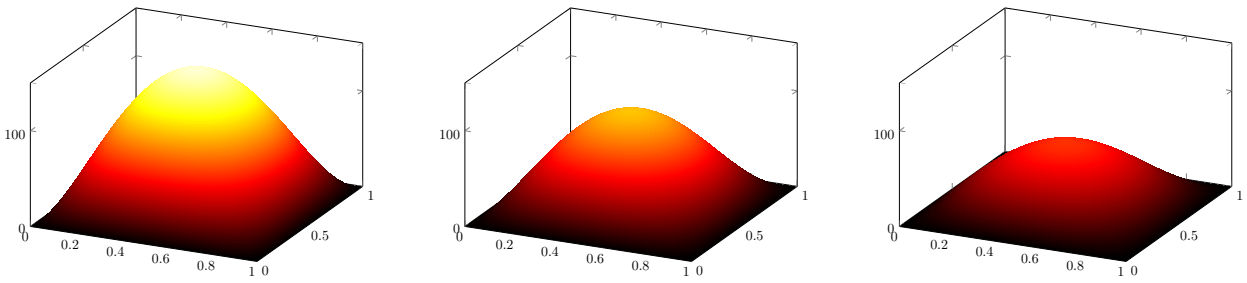

Figure 1: Snapshots of the solution to the heat equation with initial data $f(x, y) = 2500xy(1 - x)(1 - y)$ on the domain $(x, y) \in [0, 1]^2$. Left: $u(x, y, 0)$. Middle: $u(x, y, 5.0)$. Right: $u(x, y, 10.0)$.

Navigate to the URL `https://computing.llnl.gov/tutorials/mpi/exercise.html#Exercise1` and find the files in the table under the description "2D Heat Equation". Download all the 2D

heat equation source files for either C or Fortran (depending on which language you intend to use). Note that these files use a `draw_heat.c` file – you will be ignoring this file.[1] Perform the following operations:

- **a.** Disable the graphical plotting (the lines calling the `draw_heat` subroutines) inside both the `ser_heat2D` and `mpi_heat2D` files.
- **b.** Rewrite portions of the code so that the data output files (`.dat`) are written to your own subfolders on one of the cluster's data partitions (mounted on `/vulcan`, `/aphelion`, `/umbra`, `/barycenter`, or `/quasar`). You may first need to create your own subdirectories in these folders.
- **c.** Write pbs scripts to run *both* the `ser_heat2D` and `mpi_heat2D` files on the cluster.
- **d.** Run the serial and MPI codes (using `mpirun` for the MPI code) on the cluster, and plot the output for one of the codes, similar to Figure 1. (Feel free to play around with various parameters in the code; I am not asking you to reproduce an exact result.)
- **e.** Your submission should contain (1) the code and scripts you used on the cluster, (2) a picture of the output (for your report).
- **f.** Your report should contain (1) a description of the heat equation problem, and (2) how you amended the code, and (3) what you did on to run the jobs on the cluster. You may optionally include investigations about run timings, e.g., comparisons between the serial and parallel code.
- **g.** For an optional challenge: make a video of the evolution of the solution u. Do not submit your video in the git repository. Instead, create a folder for your data on one of the data partitions of the cluster. (Again, they are mounted on `/vulcan`, `/barycenter`, etc.) Put your video in a folder of your own on one of these data partitions and include the location in your report.

2 Chaos in the Lorentz attractor

In three dimensional space $(x, y, z) \in \mathbb{R}^3$, let $x(t)$, $y(t)$, and $z(t)$ denote the position of a particle as a function of time. Consider the evolution of these coordinates as described by the system of ordinary differential equations called the *Lorentz system*:

$$\begin{aligned} x'(t) &= \sigma(y - x) \\ y'(t) &= x(\rho - z) - y \\ z'(t) &= x\,y - \beta z, \end{aligned}$$

Above, some initial data $(x(0), y(0), z(0)) = (x_0, y_0, z_0)$ is prescribed and σ, ρ, and β are constant parameters. This system of equations is a simplified model of particulate motion in the atmosphere.

This system of differential equations is interesting because, for certain choices of the parameters σ, ρ, and β, it is known to exhibit *chaotic* behavior. That is, the trajectory of the solution is extremely sensitive to miniscule perturbations of the initial data (x_0, y_0, z_0). We will investigate this behavior. Your task involves simulating the state of the Lorentz system and showing how it behaves with respect to tiny perturbations of the initial data.

[1] The purpose of this file is to generate an interactive graphics window to plot the solution, which we won't be doing on the cluster.

Precisely, let $\mathbf{Z}$ be a three-dimensional standard normal random variable (i.e. each component of $\mathbf{Z}$ is a mean-0 and variance-1 random variable with Gaussian distribution.) We will consider the solution to the Lorentz system with the initial data $(x(0), y(0), z(0)) = (1, 0, 0) + \frac{1}{100}\mathbf{Z}$. For each realization of $\mathbf{Z}$, we can obtain the solution to the system as a point $(x(T), y(T), z(T)) \in \mathbb{R}^3$. By considering a Monte Carlo ensemble of realizations of $\mathbf{Z}$, this can be represented as a particle cloud: See Figure 2, which shows these particle clouds for the Lorentz system with the parameter choices $\sigma = 10$, $\beta = \frac{8}{3}$, and $\rho = 28$, which are values known to induce chaotic behavior.

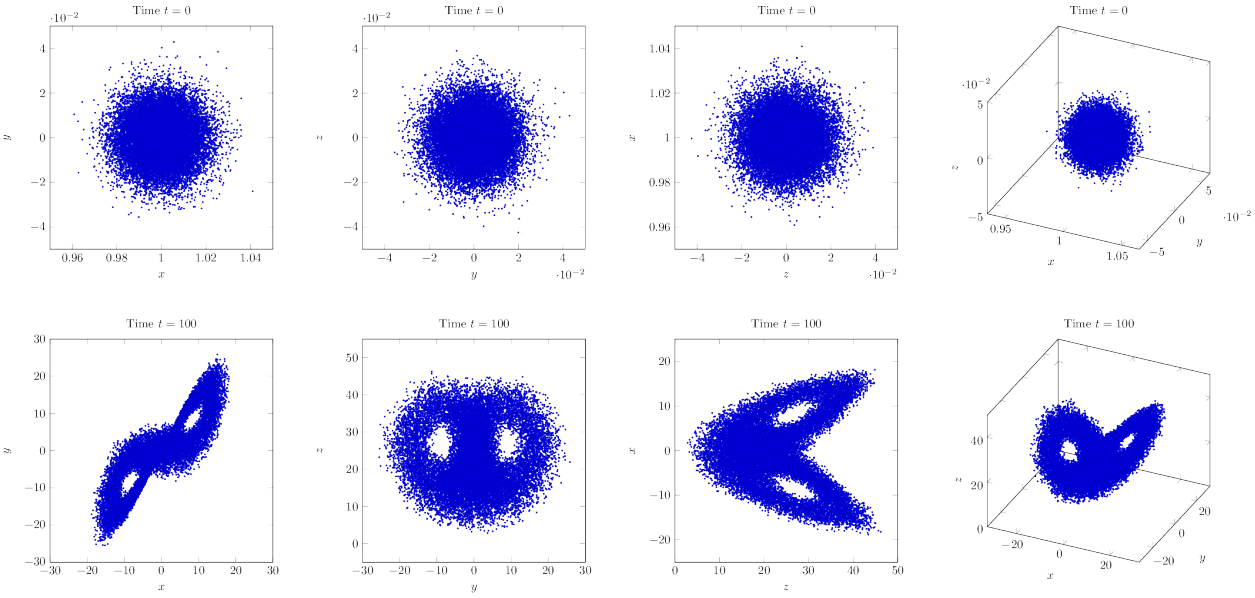

Figure 2: A size-10^4 ensemble of solutions to the Lorentz system at fixed values of time. We consider cloud particle representations of the initial data ensemble, taken as realizations of $(1, 0, 0) + \frac{1}{100}\mathbf{Z}$, where $\mathbf{Z}$ is a 3-dimensional standard normal random variable. Top: particle cloud at time $t = 0$. Bottom: the ensemble at time $t = 100$. From left to right: different viewing angles. Note in particular the scale: the $t = 0$ particle cloud is very compact compared to the spread of the $t = 100$ cloud.

Your task is to reproduce, extend, and/or augment the scatterplots in Figure 2.

- From the class git server, clone the repository `lorentz`, which contains basic serial C and Fortran code for simulating the Lorentz system.
- You are free to modify the code in the `lorentz` repository as much or as little as you like. But your submitted code must employ MPI to foster computational parallelism.
- You may want to use directives such at `MPI_GATHER` to collect point locations from different tasks.
- To generate standard normal random variables and cannot generate them natively (e.g., in C), then you are free to use any external library on the Internet that you wish. However, it may be simpler to use the Box-Muller transform:
  see `http://en.wikipedia.org/wiki/Normal_distribution#Generating_values_from_normal_distribution`,
  `http://en.wikipedia.org/wiki/Box-Muller_transform`.

- The parameter choices $\sigma = 10$, $\beta = \frac{8}{3}$, and $\rho = 28$ are known to produce chaos, but many others do, too. You may change the parameter values to anything you wish.
- Your submission should contain your C or Fortran code with an appropriate makefile, and bash/pbs scripts.
- Your report should describe what problem you are trying to solve and what your code does. It should contain whatever scatterplots you were able to generate, and should give a high-level description of your approach for parallelism.
- For an optional challenge: make a video of the particle cloud evolution. Submit your video as directed in the heat equation problem. (I.e., upload it onto a data partition of the cluster, do not submit it via git.)

3 Submit

You must submit (1) your source code (C and/or Fortran), and (2) your LaTeX source code (*not* the dvi/ps/pdf output) via a Git repository. You are submitting to the `john-smith-project-2` repository on the class git server, where you should replace `john-smith` with your first and last name.