**Christopher Lang**

**STA9794 Spring 2017, Assignment/Project A**

The normal program was somewhat simpler in terms of work. It heavily borrowed/stole code from the scrub program

A key difference was that normal program was significantly faster in general. Its likely because it heavily leans on numeric computation vs. string manipulation. Normal’s use of numpy greatly enhanced performance

I had settled on Jarque-Bera hypothesis test for normality. The test statistic requires the series’ kurtosis and skewness, which in turn requires standard deviation and mean. Ultimately, we needed to calculate the mean, which turned out to be fast across multiple process

The mean is then use to calculate the standard deviation, kurtosis, and skewness. The approach for parallelization is based on the formula itself:

Above is the standard deviation formula. We can parallelize easily because:

1. The numerator is a summation. Hence each process can work on a part of the data, sum it up, and send the results to the root process. The root process then sums each process’ result
2. Similarly, the denominator is effectively the sum of the number of points. Each process keeps track of this number, which is then sent to the root process

Kurtosis and Skewness have similar formulas, though it also depends on the standard deviation. Hence the workflow is:

1. Root determines each process’ start line number for the file
2. Root scatters this to have all process help compute the mean (sum numerator, and denominator, and return the results). Then root gathers the partial numerators and denominators, sums them and divides to get the sample mean
3. The sample mean is send to all process to begin computing the skewness, kurtosis, and standard deviation. Each process does all three, returning the numerator and denominator to the root process
4. The root process computes the final standard deviation, skewness, and kurtosis. With only 3-6 numbers, the root can then compute the Jarque-bera statistic and compare it to the chi-square

This program proved fast, with its emphasis on numpy to crunch numbers quickly

**Update for Nathaniel**

I was not able to get MPI running for Nathaniel’s requests on grading this Assignment. There were errors in relation to orte\_init.c, seemingly a C file related to open MPI. This error continued to appear when running in “interactive” mode as suggested by CUNY HPCC. As an alternative, I ran the scrub program in serial, without the use of MPI. The program ran successfully prior to the original project submission deadline

There are two log files, “results” and “execution”. The results file, located under the result directory, contains topline statistics about the program. This includes system information, execution times, speed, etc. The execution log file, located under the log directory, contains a raw step-by-step program execution information