# Parallel Statistic Analyzer (PaStA)

Dominic J. Catalano

February 22, 2016

## 1 Project Description

In order to clearly understand data in a dataset, it is necessary to see the distribution of the data. By creating a histogram of the data allows users to see the distribution of the dataset while also highlighting any anomalies that may exist. The Parallel Statistic Analyzer (PaStA) scans over a given dataset and produces such a histogram based on the frequency of entries that exist within a certain standard deviation range while also displaying the mean and standard deviation of the data. Such a histogram can help a data administrator note trends while also identifying possible outliers agnostic of the dataset provided.

The program is designed to work with any set of numbers. However for the purposes of testing, a random number generator library (rngs and rvgs) with a set seed will be used to generate a geometric distribution of high variance. This generator was chosen because of its high variance, resulting in larger, more meaning histograms. It was designed by Steve Park and Dave Geyer in conjunction with a paper on random number generation. It will also provide similar histogram results regardless of set size.

## 2 Code Design

Code for the calculation of the standard deviation along with the construction of the histogram was designed specifically for this project using a two pass algorithm. The two pass algorithm should further the performance gains via parallelism while also simplifying some mathematics involved. The first pass is required to find the mean of the data as $\bar{x} = \frac{\sum x_i}{n}$. The second pass is then used to calculate the deviation as $\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$. Once these values are known, a histogram can be constructed. Another pass over the data is then necessary to count the numbers in each standard deviation bin. The histogram is then iterated over, normalized, and printed for the benefit of the user along with the set statistics.

This project will demonstrate that parallelization of these processes will increase the performance by markedly decreasing the runtime of each test when compared to the serial version. Different forms of parallelization will be tested

| Set Size | Serial($s$) |
|---|---|
| $10^7$ | 0.26 |
| $10^8$ | 2.58 |
| $10^9$ | 25.77 |
| $10^{10}$ | 36.16 |

Table 1: Serial Runtimes

to determine the best implementation and coding paradigm to achieve optimal performance.

# 3 Execution Instruction

Because the dataset is generated by the program, the only parameter passed will be the set size (i.e. PaStA 1000000) for an interactive session. Each version of the program will be accompanied by its bash script that is meant to produce a plethora of results (i.e. serial.bash). For the serial version, sets of $10^7$, $10^8$, $10^9$, and $10^10$ are passed into the program. The output for each test will be generated in an output file prefixed by the execution time (i.e. serial_output_file).

# 4 Initial Findings

The initial results come from serial.bash whereby SerialPaStA was run with tests sets of $10^7$, $10^8$, $10^9$, and $10^10$ as an input. The timing was done by using the time.h library to time the execution from after the generation of the dataset to the completion of the program. The histograms each provided results that were comparable but not exact. This is to be expected because, while the seed is set, the sets are all different sizes. For this concision the result sets are not reproduced here. A table of the run times compared to set size and execution method can be seen in Table 1.

# 5 Git Repository

The git repository can be found at *https://github.com/dom-catalano12/PaStA*.

# References

[1] S.K. Park & K.W. Miller *Random number generators: good ones are hard to find* Communications of the ACM, Volume 31 Issue 10, Oct. 1988