Threading the Needle

Testing the effect of threads and parallel programming

**Damion Monroe**

2.23.2018

CSCI 440

# INTRODUCTION

The goal of the experiment was to write a program that found the frequency of each word in a file and then use parallel mechanisms to see the difference in speeds. I chose to use python for this project in order to enhance my knowledge of the language and for its ease of use.

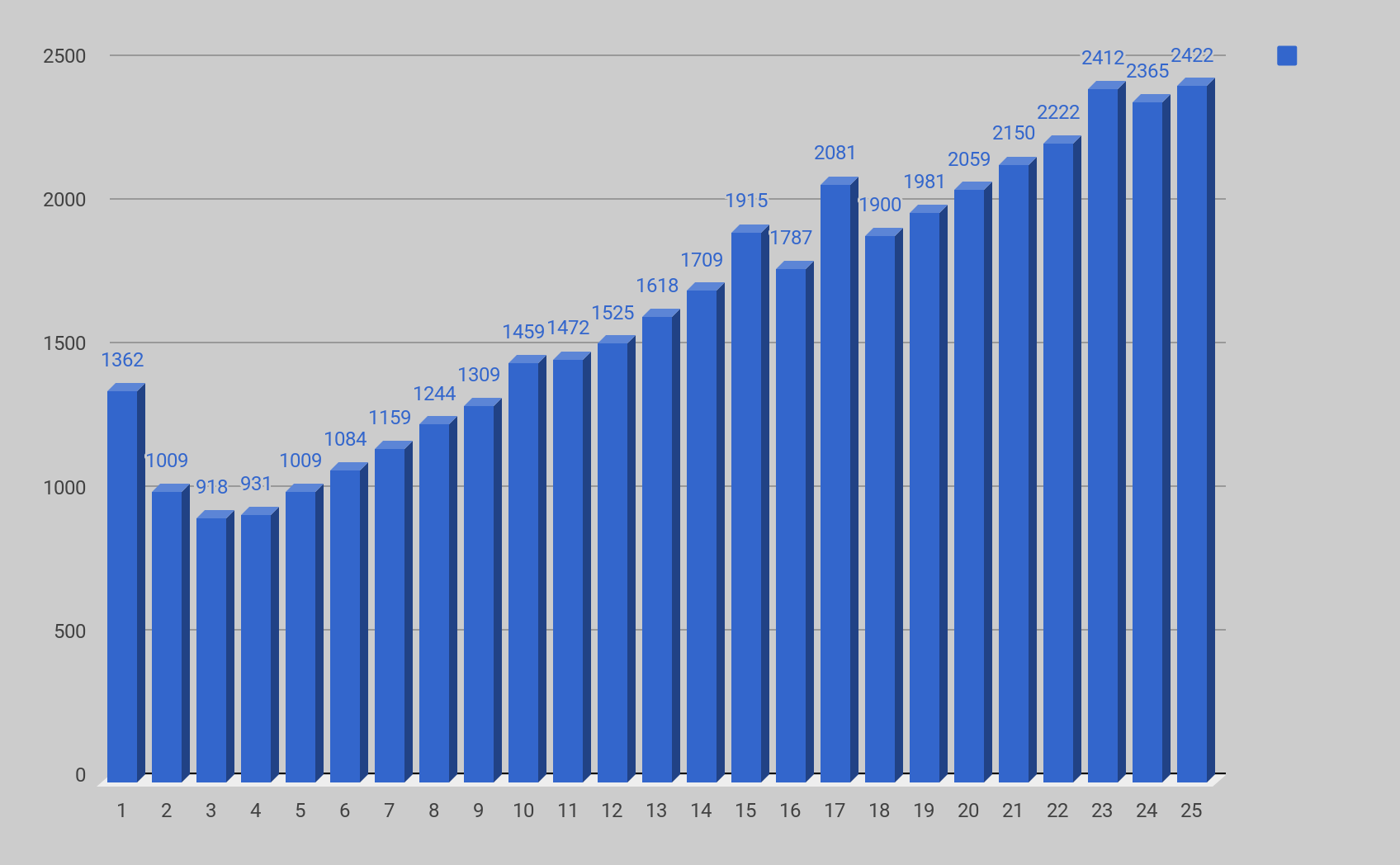
# HYPOTHESIS

I hypothesized that the program would get faster as the number of threads increases.

# PROCESS

As I was testing some early versions of the program I encountered a weird phenomena in which the time was increasing as I added more threads. After some research I found that Python cannot use threads in parallel this is due to the Global Interpreter Lock (GIL). The GIL forces threads to be used only as a concurrency tool and not a parallel tool. So in order to make a Python program parallel you must use multiple processes which require more overhead than threads due to the fact that processes do not share memory like threads do. My current program uses processes from the multiprocess library that Python provides as it's tool to parallel programming. Another phenomena that is still currently unsolved is that a current style of file takes an absurd amount of time to compute compared to other much larger files. I found that when the file is just a few words repeated many times the processes took around 1.5-3x the time of a file that was around 10x the size.

# DATA



# RESULTS

Currently the data shows that for my machine 3 separate processes provides the best performance for my test file which is a 100mb text file. The reasons that the times start to increase as you add more than 3 processes is because the overhead of controlling the processes overshadows the splitting of the data into smaller pieces.