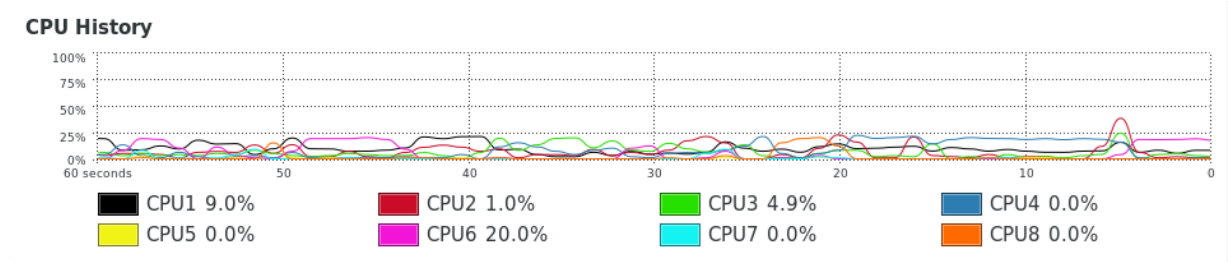
Matt Dumford
mdumfo2@uic.edu
CS385 homework 3

The main purpose of my experiments was to determine how affects how the program runs on a computer. I chose to implement the program by simply splitting the file into equal parts and letting each thread work on a single segment of the file. This method was easier to implement than having the threads draw from a pool of chunks of data and it also assured that all threads would run for the same duration and perform the same amount of calculations.
I implemented each of the equations provided in the pdf instructions.

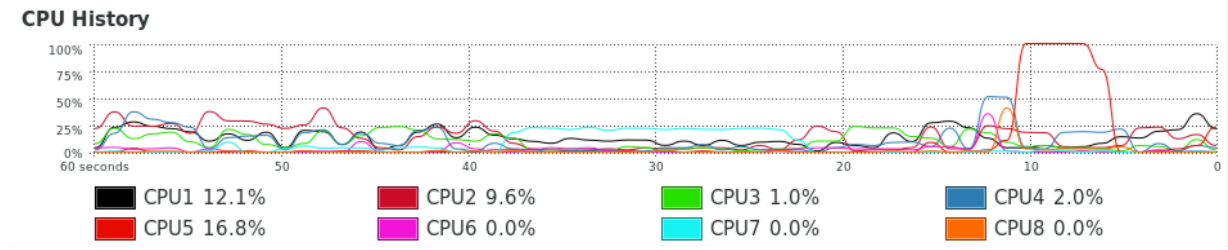
I also found that the files provided were too small to see any significant results using my system monitoring tool, so I created a 2M file of random data using the following command:

```
dd if=/dev/urandom of=rand.out bs=1M count=200
```

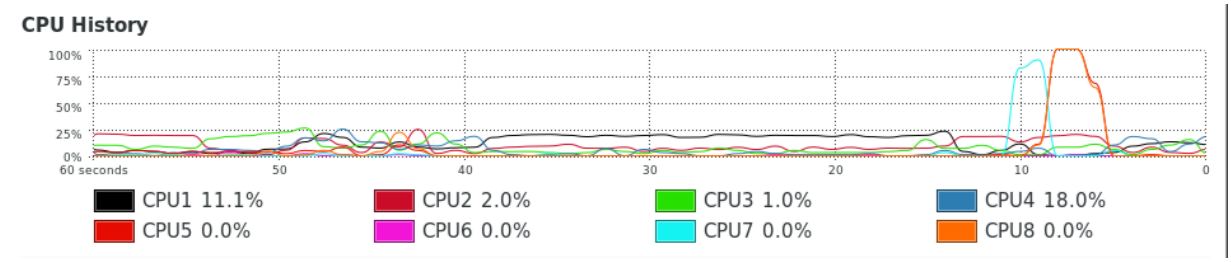
This allowed the program to run much longer (still only 5-6 seconds though) and do more calculations, giving me better results. I then ran my program on the resulting file with a varying number of threads and saw the following results:



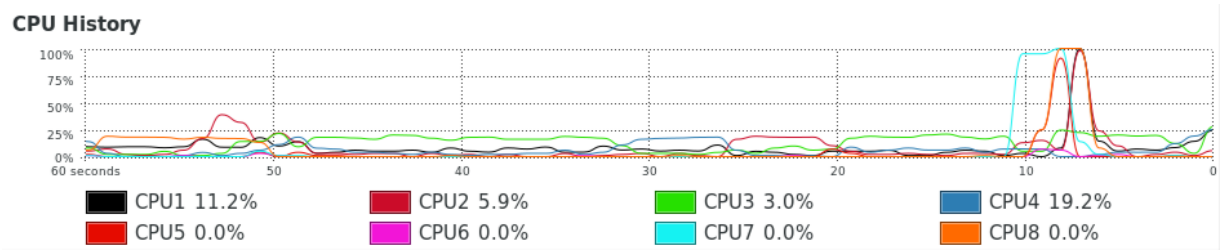
Not Running



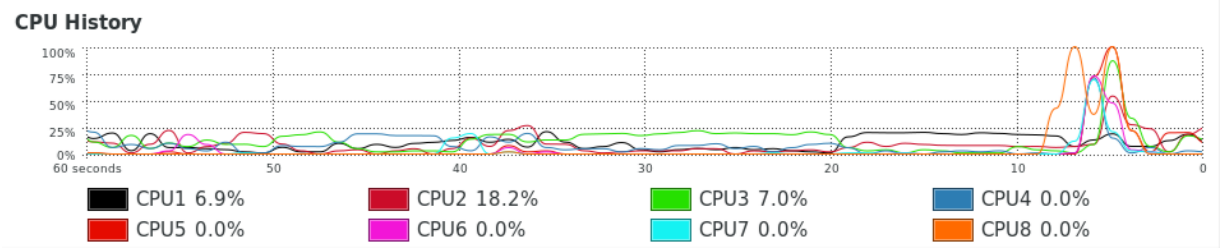
1 Thread



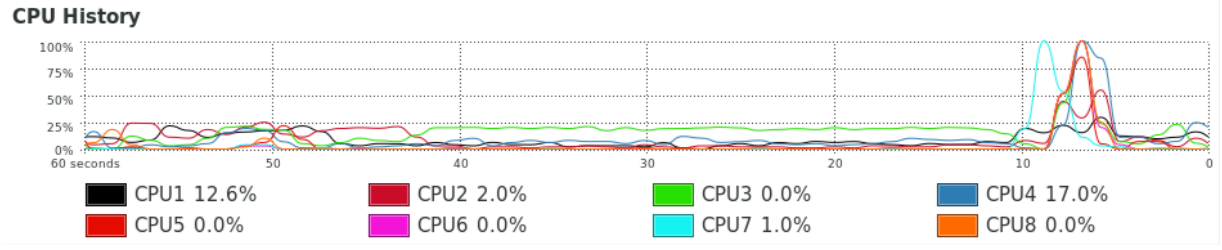
2 Threads



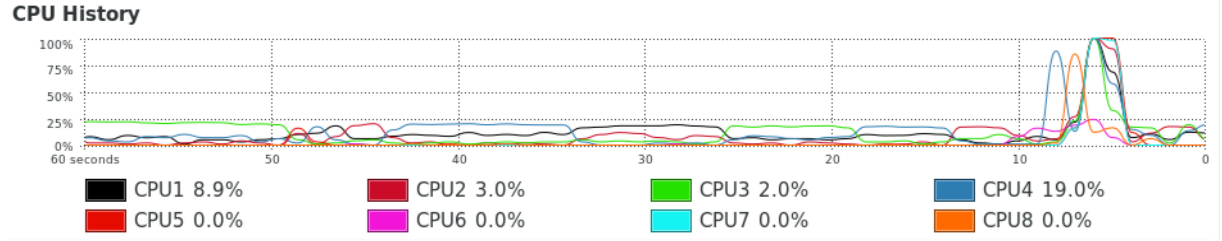
3 Threads



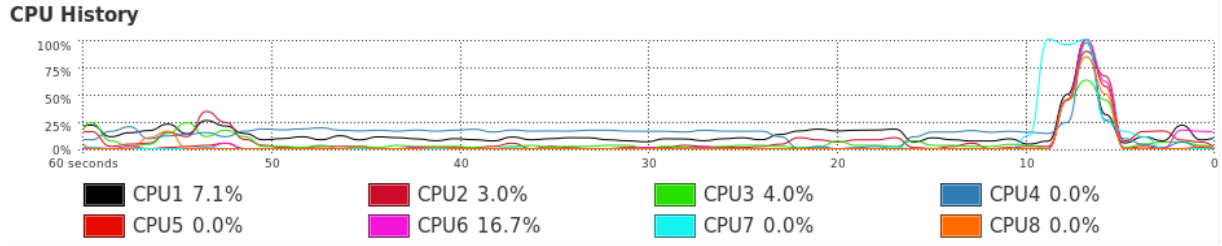
4 Threads



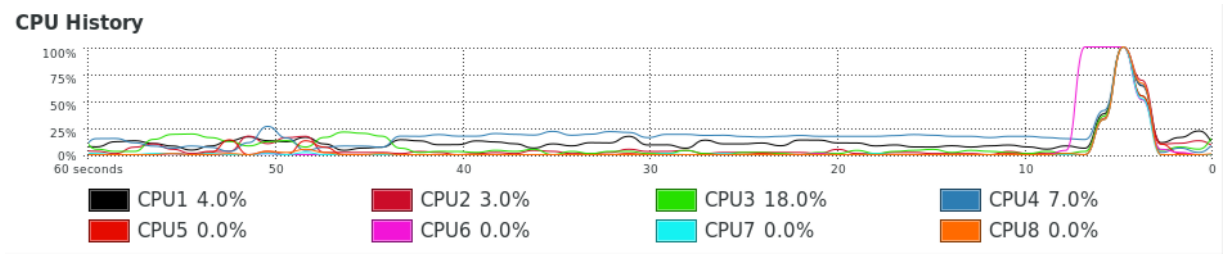
5 Threads



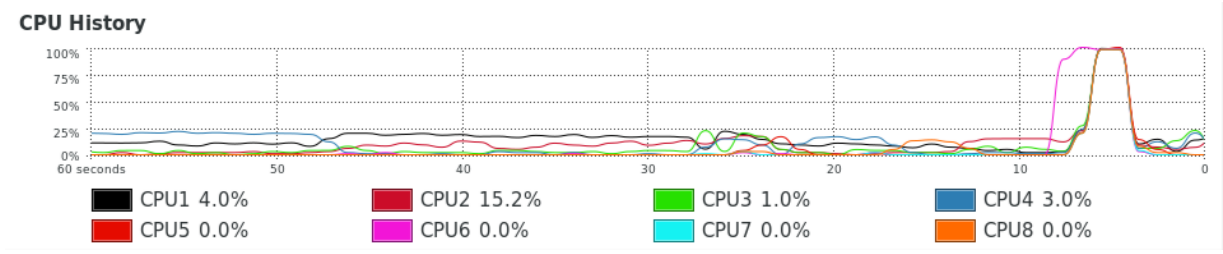
6 Threads



7 Threads



8 Threads



5000 Threads

These tests were done on a 4 core machine with hyper threading, resulting in 8 virtual cores. The first spike at the beginning of each run is most likely when the file is being read in, since that is done with a single thread. After that is the threads actually doing their computations. For the most part, the number of CPUs that became busy was the same as the number of threads I chose. It seems like extra CPUs started doing more work too though, perhaps to take care of other processes while the other CPUs were busy running my program. After I got past 8 threads there were no major changes except that it would take slightly longer when run with an extreme amount of threads. I think that this is because switching all of the threads on and off the CPUs is taking up some extra time. I also only saw improvements when run with up to 3 or 4 threads though. After that, any improvements were negligible.