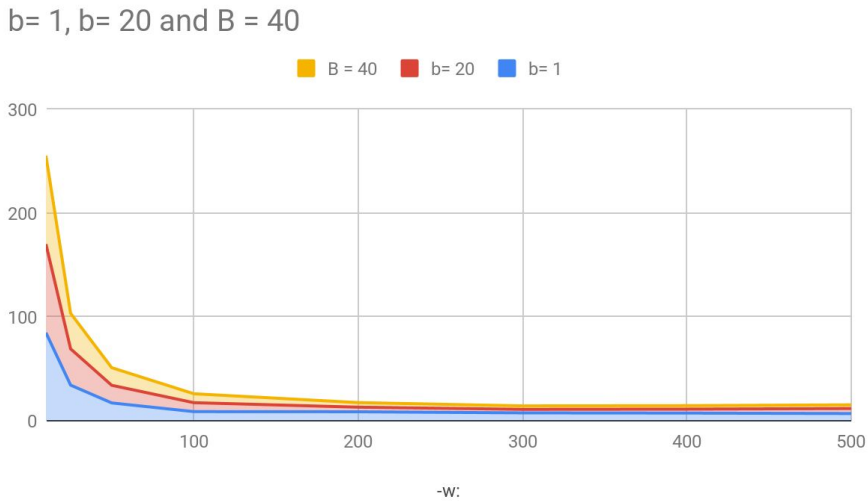
Kevin Tang
CSCE 313

PA4 Threading and Synchronization Report

I will discuss the results and performance of the client program in two types of requests: data, and file. We will see how performance changes with increase in number of worker threads and varying size request buffer. All tests were done in a linux vm on my pc with processor specs: 3.7 Ghz, 8 Cores (16 Logical Processors). The VM was given 8 Gb of RAM, and 8 Logical Processors.

Table for Data Requests:

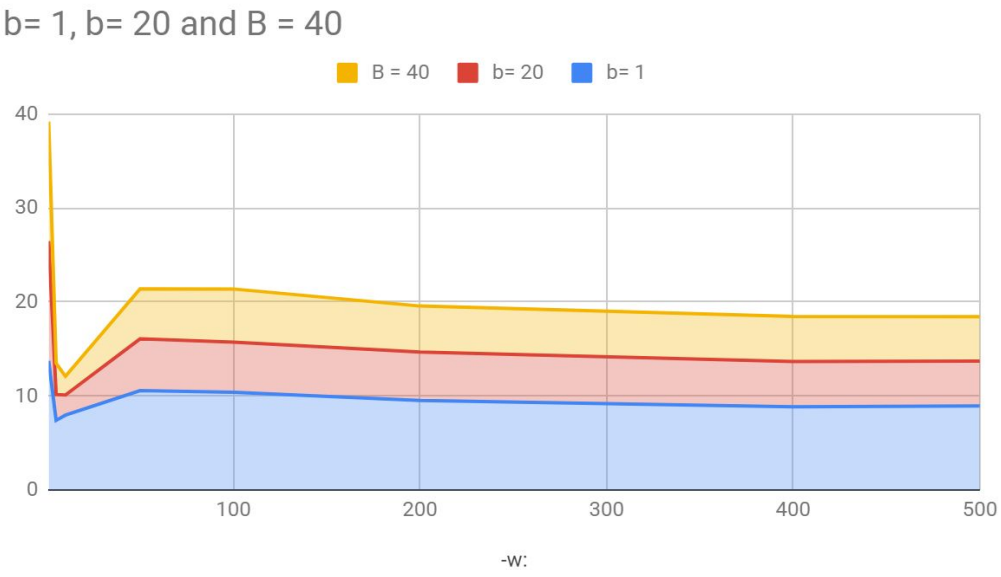
Data Request with varying worker threads and buffer sizes Constant: n = 15K p = 10			
-w:	b= 1	b= 20	B = 40
10	84.735	85.283	85.167
25	34.273	34.905	34.273
50	17.128	16.986	16.952
100	8.751	8.610	8.596
200	8.621	4.402	4.424
300	7.534	3.413	3.335
400	7.338	3.816	3.337
500	6.905	4.837	3.576



Analysis:

- As the number of threads w increases we see an exponential decay in the time in seconds. This indicates that the performance increases non linearly and logarithmically as the number of threads increase.
- As for buffer size, it does not seem to have a large effect on performance; however, it seems to affect increased thread counts. This make sense because with more threads, there will be more waiting on buffer to open up. This would become a larger bottleneck if more threads are requesting for that resource.

File Request with varying worker threads and buffer sizes with a binary file of 67MB size Constant: f = mybin m = 256			
-w:	b= 1	b= 20	B = 40
1	13.732	12.739	12.716
5	7.396	2.755	3.312
10	7.953	2.138	1.993
50	10.567	5.501	5.320
100	10.389	5.328	5.659
200	9.519	5.148	4.895
400	8.837	4.819	4.800
500	8.936	4.774	4.725



Analysis:

- For threading, it has similar graph to data requests threading. It is logarithmic in terms of performance, and at a certain point more threads cause larger overhead and actually decrease performance. This is likely due to CPU limitations not actually supporting that amount of threads.
- For buffer size we see a larger effect on performance than in data requests. Increased buffer size seems to increase performance. After a certain point there is no effect on buffer size and performance. A possible bottleneck in this implementation is that the thread pushing to the buffer is not “as fast” as the the many threads popping from the stack.