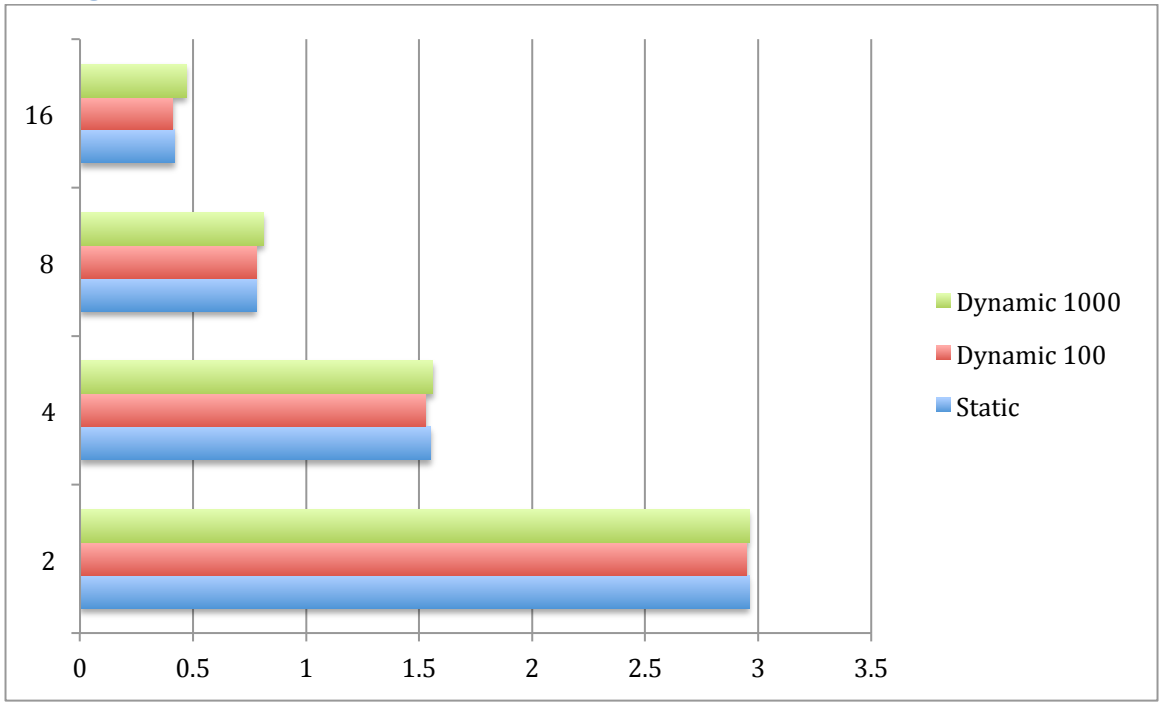
Ian Schweer
22514022

Project 3

Graphs and analysis:

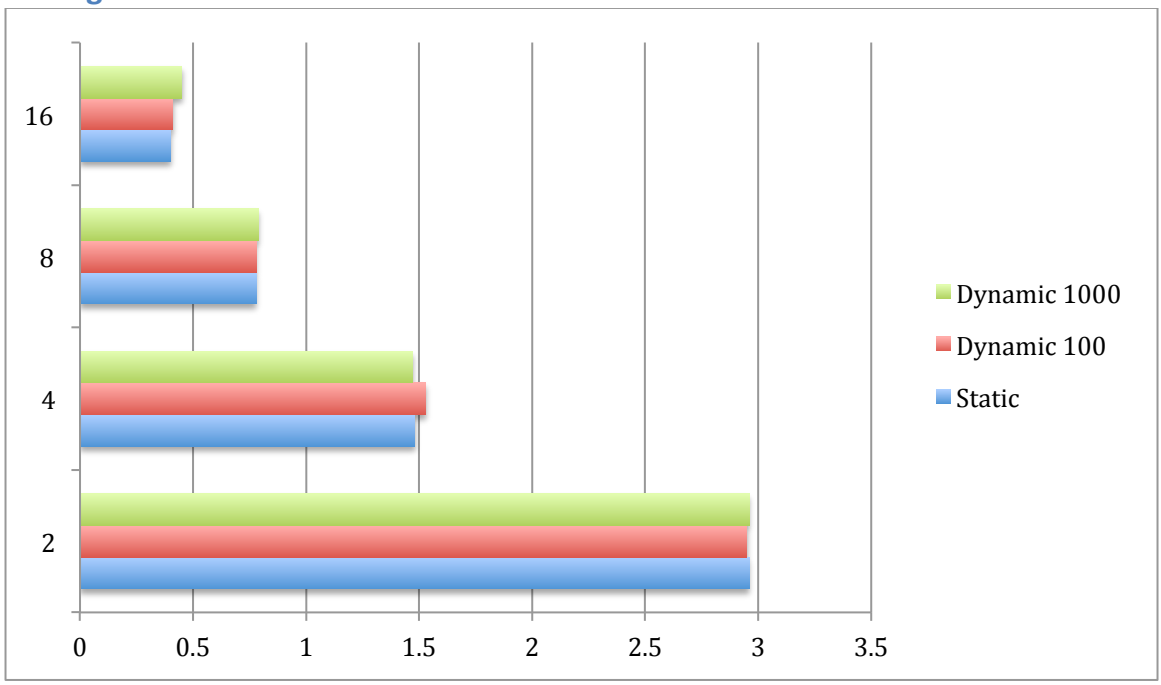
In order to get actual comparable numbers, there was a 500 nanosecond wait inserted into the code to exaggerate the runtimes.

Timing 1



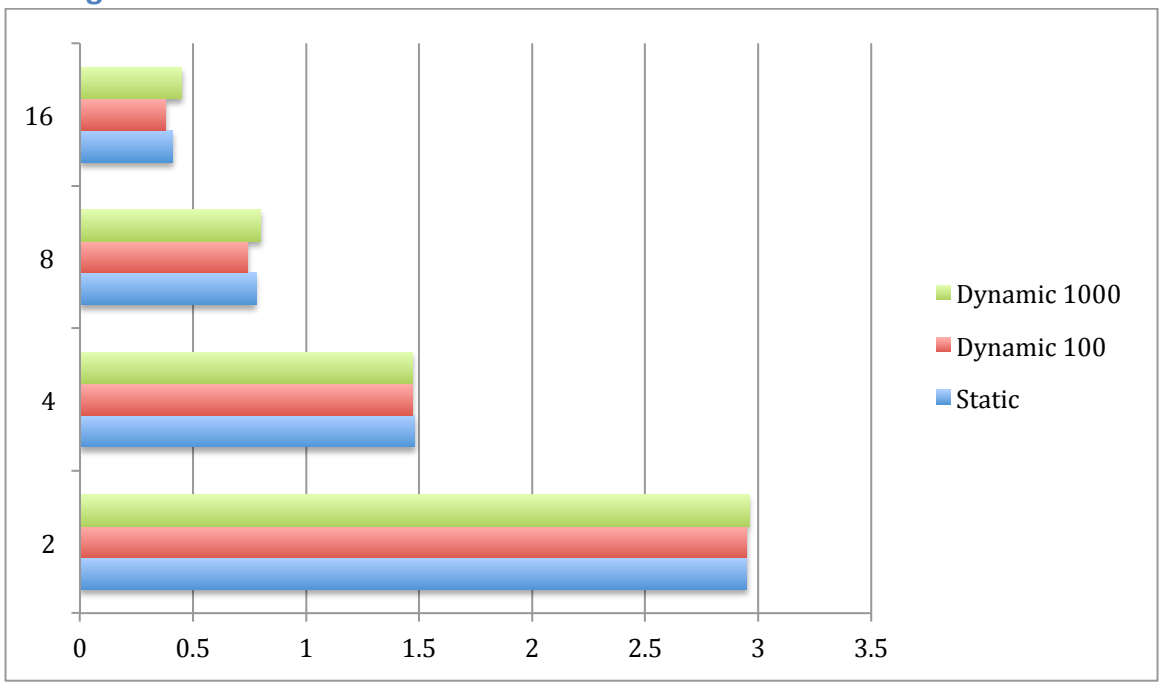
For static scheduling, we see that we increase in runtime as we increase the number of threads. Dynamic scheduling of 100 chunks runs slightly better on this data set as opposed to static scheduling by roughly 100 milliseconds. However dynamic scheduling of 1000 chunks takes slightly longer on average of 75 milliseconds. This is similar to what we saw in lab one where as the thread number increases, we see an increase. The reason why a chunk size of 1000 doesn't have threads "report" as much which makes it slower.

Timing 2



For this timing file, static scheduling proved to be best with this file. Dynamic scheduling was slower, 100 size chunk's being the slowest. That could be because the file contains mostly whitespace, so reporting more often actually takes more time then running the palindrome algorithm. We saw this in the first assignment when the data increased in length as the file got bigger.

Timing 3



For this timing file, we saw a lot of whitespace but lines with actual data appeared more often (more uniformly distributed content). In this case, checking in more often was better, because a thread was highly likely to get one palindrome to look through, as opposed to getting maybe 10 palindromes to search through in the case of 1000 sized chunks and some amount close to (10000 / number of threads) / 100 palindromes. This is similar to when palindromes got smaller as the file grew in palindrome size.