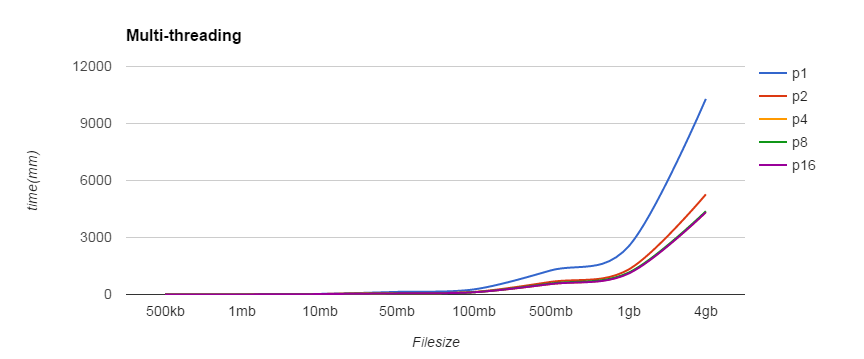
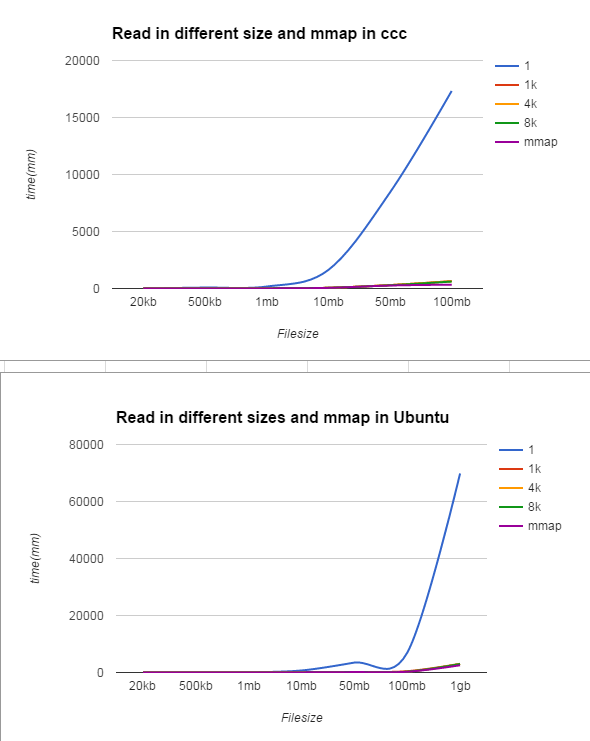
**ReadMe**:<-This is **important**!

I used recursion to search for string. So when the file size is too large while the occurrence of start char of the string is too high , mmap (single or multithread) function sometimes ran out of the stack memory because the recursion goes too deep and overflows the stack.

But if the occurrence of the start char of the string is low, regardless of the file size. Or the occurrence is high but file size is small. Mmap (single or multithread) should work fine since the recursion will not go deep.

In performance analysis, I tested large files with little occurrence of the string. Mmaping(single thread and multi threads), reading in default size and reading in customized size performed the same search result if the stack was not overflow

**Graph and Analysis are in next page:**

Performance Analysis: 

**If additional threads provide better performance even though the machines have a smaller number of cores?**

Yes, it speed up the searching process up to a point where the file size gets really large that even though multithreading cannot speed up a lot.

**Other general ideas about the performance:**

I tested various reading size as well as single thread mmap in both the ccc server and my Ubuntu. Reading the file in a large chunk of size definitely speeds up the search a lot. However, once the read size passed a point, it speeded up only a little. Read() by 1 byte was so slow and mmap worked the best. Mmap() worked mostly the same as Read() in a large size(e.g. 8k), which make sense because reading in a size that close to the file size or larger than the file size is similar to mmap because they both bring large size of contents into the process at once.

I tested multithreading only on my Ubuntu since there are so many processes running on the ccc server at the same time. Multithreading helped the searching in a range. If the file was too small, multithreading did not help (or helped a tiny bit). Also, once the file got too large, more threads would help, but only a little. Multi-threading worked the best in file’s size range of 10MB to 1GB. There was a remarkable difference between dual threading and single threading. However, the differences among multi-threading and dual threading were not large.