**Student-id: s3654028**

**Name: Nghia Huu Le**

**Task1:**

awk '$1 ~ /^[a-z]+$/' $1 | awk '{ if (length($0) < 16 && length($0) > 2) { print } }' | uniq -u

- filter to keep only z-a characters

- filter to keep only the words with length from 3→15

- filter all duplicate words

This is performance of task1

*sudo perf stat -e cs -e cpu-clock ./Task1filter shuf\_wlist\_match1.txt ttt.txt*

*Performance counter stats for './Task1filter shuf\_wlist\_match1.txt ttt.txt':*

*17 cs # 0.003 K/sec*

*6,198.54 msec cpu-clock # 1.000 CPUs utilized*

*6.199168974 seconds time elapsed*

*0.680306000 seconds user*

*5.518484000 seconds sys*

Since it’s hard to compare the outputs of script and task1 bin, so I shuffle the input firstly then task1 and script take the same shuffled input. I think it’s no impact to our purpose.

**Task2:**

I sorted in reduce function the third character also since I don’t see the requirement to sort words in which position.

Also, I output the words to files stored in disk.

This is the performance:

*sudo perf stat -e cs -e cpu-clock ./task2 ttt.txt | wc -l*

*Performance counter stats for './task2 ttt.txt':*

*364 cs # 0.027 K/sec*

*13,733.44 msec cpu-clock # 1.043 CPUs utilized*

*13.164187583 seconds time elapsed*

*3.089765000 seconds user*

*10.644659000 seconds sys*

*1367007*

**Task3:**

This is task3 performance

*sudo perf stat -e cs -e cpu-clock ./task3 ttt.txt | wc -l*

*Performance counter stats for './task3 ttt.txt':*

*3,479 cs # 0.464 K/sec*

*7,492.94 msec cpu-clock # 1.331 CPUs utilized*

*5.631251732 seconds time elapsed*

*1.735851000 seconds user*

*5.768919000 seconds sys*

*1367007*

It seems quickly than the task2, I think the fifo speed is better than read data from file in disk but using multi-thread to send make write process better, in other words: map thread is quicker.

I try to measure each child thread in this task by modifying the code to run 1,2,3...15 child-map3 threads

here is the result,

Run only 3-word-length thread (1 thread):

*Performance counter stats for './task3 ttt.txt':*

*34 cs # 0.010 K/sec*

*3,555.78 msec cpu-clock # 1.002 CPUs utilized*

*3.548367529 seconds time elapsed*

*0.556022000 seconds user*

*3.000122000 seconds sys*

*14476*

Run only 3-4-word-length thread (2 threads):

*Performance counter stats for './task3 ttt.txt':*

*82 cs # 0.022 K/sec*

*3,688.85 msec cpu-clock # 1.020 CPUs utilized*

*3.616404072 seconds time elapsed*

*0.535178000 seconds user*

*3.154736000 seconds sys*

*192698*

Run only 3-4-5-word-length thread (3 threads):

*Performance counter stats for './task3 ttt.txt':*

*237 cs # 0.059 K/sec*

*4,042.90 msec cpu-clock # 1.056 CPUs utilized*

*3.828774516 seconds time elapsed*

*0.695614000 seconds user*

*3.349404000 seconds sys*

Run only 3-4-5-6-word-length thread (4 threads):

*Performance counter stats for './task3 ttt.txt':*

*548 cs # 0.122 K/sec*

*4,480.52 msec cpu-clock # 1.104 CPUs utilized*

*4.058914866 seconds time elapsed*

*0.710595000 seconds user*

*3.773785000 seconds sys*

*374909*

Run only 3-4-5-6-7-word-length thread (5 threads):

*Performance counter stats for './task3 ttt.txt':*

*905 cs # 0.176 K/sec*

*5,135.58 msec cpu-clock # 1.161 CPUs utilized*

*4.424585026 seconds time elapsed*

*0.904795000 seconds user*

*4.234441000 seconds sys*

*585919*

Run only 3-4-5-6-7-8-9-10-11-12-word-length thread (10 threads):

*Performance counter stats for './task3 ttt.txt':*

*2,618 cs # 0.366 K/sec*

*7,160.30 msec cpu-clock # 1.317 CPUs utilized*

*5.436954559 seconds time elapsed*

*1.484099000 seconds user*

*5.688381000 seconds sys*

*1275969*

**Task 4:**

The running time of threads in map depends on their word number.

This is the word number in my dedicated test case:

14476 ./hw1/wordfile3.txt

63447 ./hw1/wordfile4.txt

114775 ./hw1/wordfile5.txt

182211 ./hw1/wordfile6.txt

211010 ./hw1/wordfile7.txt

205862 ./hw1/wordfile8.txt

177662 ./hw1/wordfile9.txt

139146 ./hw1/wordfile10.txt

99187 ./hw1/wordfile11.txt

68193 ./hw1/wordfile12.txt

44451 ./hw1/wordfile13.txt

28720 ./hw1/wordfile14.txt

17867 ./hw1/wordfile15.txt

So, base on this, I set priority for this dedicated test case

Performance of task4:

*sudo perf stat -e cs -e cpu-clock ./task4 ttt.txt | wc -l*

*Performance counter stats for './task4 ttt.txt':*

*684 cs # 0.392 K/sec*

*1,747.09 msec cpu-clock # 1.240 CPUs utilized*

*1.409103488 seconds time elapsed*

*0.345150000 seconds user*

*1.404407000 seconds sys*

*307436*

how the priority levels are related to speed?

→ From my perspective , thread in user space map to kernel space, and priority in user space affect to thread in kernel space. That means if thread priority is higher, then thread in kernel has more chance to run frequently (consume CPU load), so the time of high priority thread is speed up.

The run time of this task is O(n) with n is the word number

The thread performance is better if the thread number is more, it also depends on the word number that thread need to process.