Student ID: F74077120

Name: 蔡孝龍 Marc Fraser Go Magante

Department and Grade: Computer Science and Information Engineering (大三)

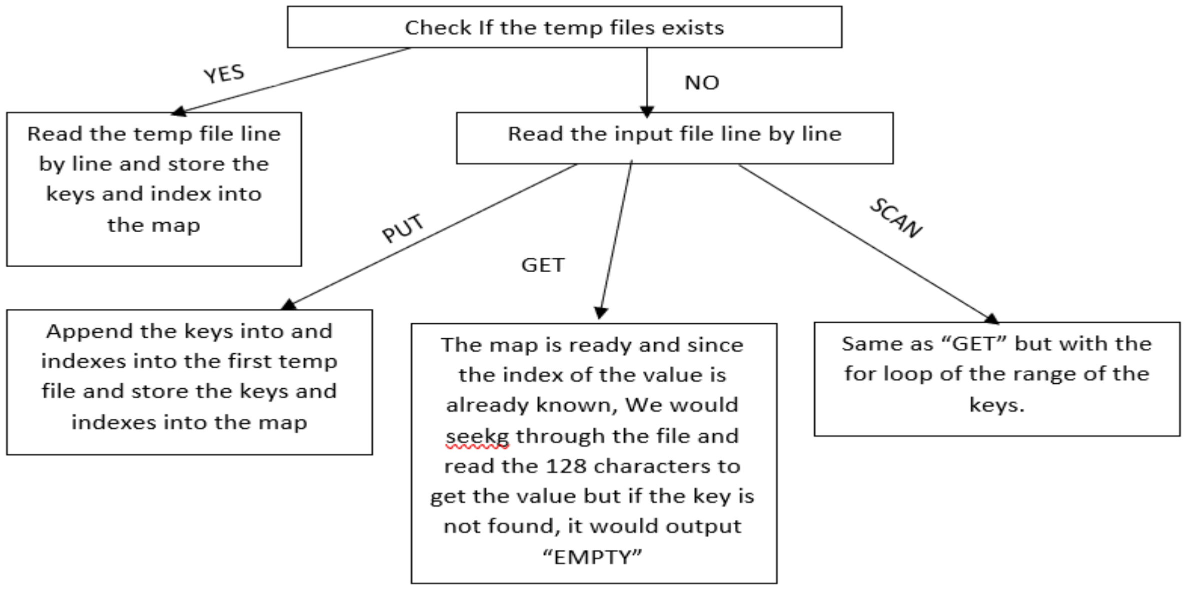
Development environment:

- OS: Ubuntu 18.04.2 LTS (VM VirtualBox)
- CPU: Intel Core i7-8750H CPU @ 2.20 x 6
- Memory: 16GB but in VirtualBox is approx. 4GB
- Programming Language (version): C++ 17

Development and user guide:

1. Development

- In order to develop a program that would support keys and values storage without actually exceeding the memory usage, I have 2 temp files. The folder "Storage" will be created automatically and the 2 temp files will be stored there. The program will first check if the two temp files exist in the folder, if it exists, it reads it line by line and stores the keys and index into the map right away. If not, it would read the input file right away. While the input file is being read, the first temp file which is the storage of keys and its index, for every "PUT", it would append the keys and the indexes into the file and store the keys and index into the map called key_storage. On the other hand, the second temp file stores/appends the values of the keys. When "GET" is detected in the input file, the temp file for value will be opened and since I already have the map ready on hand and know the index of the value, I could get the value right away through seekg and read the 128 characters of the value but if the key is not found, it would output "EMPTY". When "SCAN" is detected in the input file, it's pretty much the same as "GET" but with the for loop for the range of indexes.


Check If the temp files exists
YES
NO
Read the temp file line by line and store the keys and index into the map
Read the input file line by line
PUT
GET
SCAN
Append the keys into and indexes into the first temp file and store the keys and indexes into the map
The map is ready and since the index of the value is already known, We would seekg through the file and read the 128 characters to get the value but if the key is not found, it would output "EMPTY"
Same as "GET" but with the for loop of the range of the keys.

2. User guide:

- # Compile
- $ g++ -std=c++17 F74077120.cpp -o F74077120 -lstdc++fs
- # Run
- $ ./F74077120 ./data path or /data path or data path

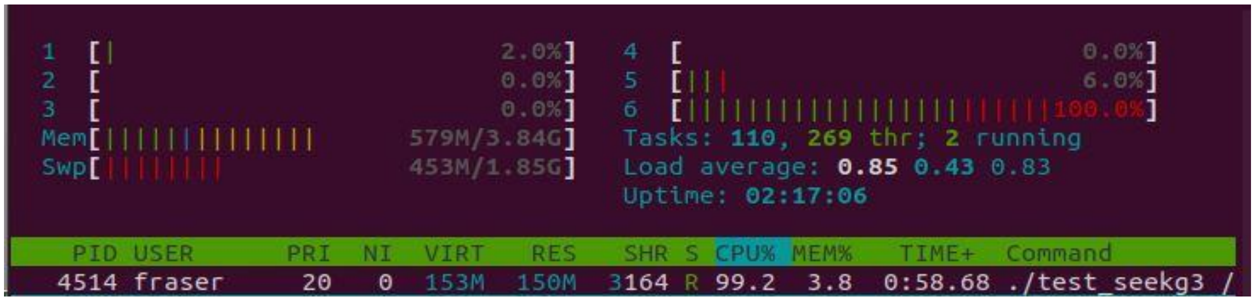
Performance analysis report:

To implement this homework, I mainly used map which is a sorted associative container that contains key-value pairs with unique keys. Keys are sorted by using the comparison function Compare. Search, removal, and insertion operations have logarithmic complexity. The advantage of using map is that it allows storing multiple keys and its values. Another advantage is that when a key's value is being updated, it updates right away and it is already sorted so finding for the desired value spends less time to search which makes it efficient.

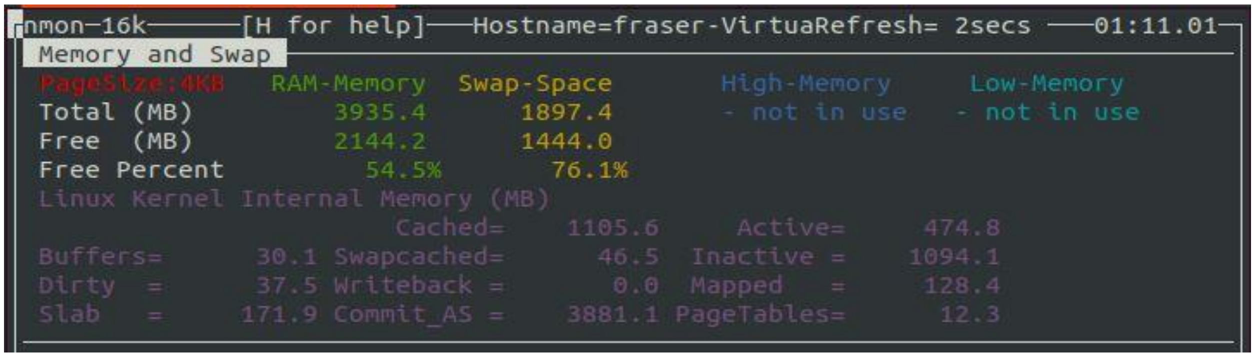
Since we are limited in memory, we would need to store the part of our data in the disk which are my 2 temp files, one for key and index and another for just values. This helps in minimizing the usage of the memory. Without it, when the file is greater than the memory, the memory usage will be over the limit which is bad and in ubuntu it would kill the program. So storing the data into the disk is actually a good thing. But there is actually a drawback, the execution time will increase due to I/O operation of the files. It increases because I/O operations are quite expensive.

For the "PUT" operation, the memory usage wasn't really that bad since I was just reading the input file and outputting the keys, index, and values into their respective temp files while storing the keys and indexes into the map.

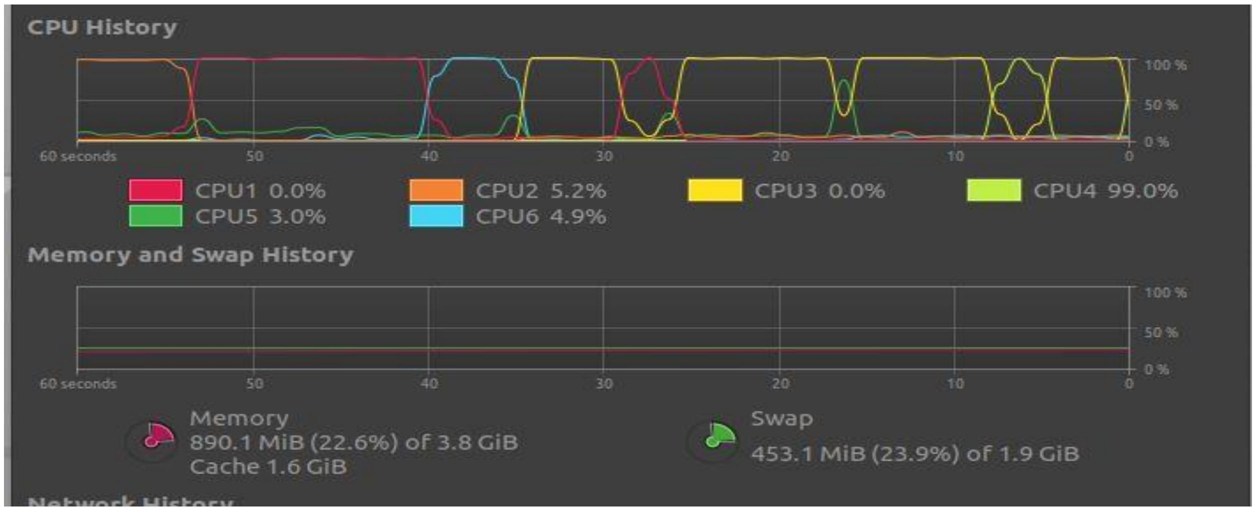
During the execution of my 1.input file that only contains "PUT" instructions, the RAM was used a little but the CPU was used quite heavily.



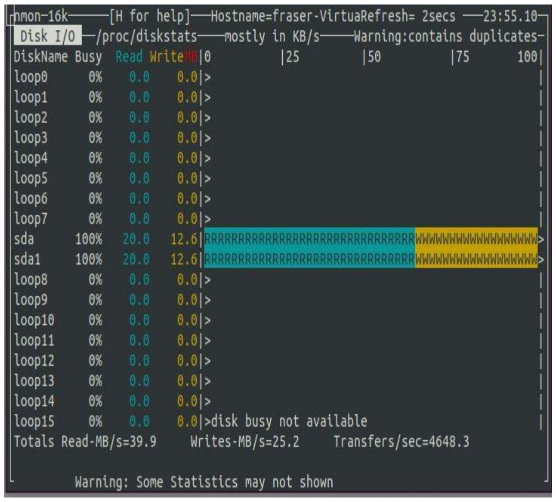
```
  1  [|                         2.0%]   4  [                              0.0%]
  2  [                          0.0%]   5  [|||                           6.0%]
  3  [                          0.0%]   6  [||||||||||||||||||||||||||100.0%]
  Mem[|||||||||||||||    579M/3.84G]   Tasks: 110, 269 thr; 2 running
  Swp[||||||||           453M/1.85G]   Load average: 0.85 0.43 0.83
                                        Uptime: 02:17:06

   PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
  4514 fraser     20   0  153M  150M  3164 R 99.2  3.8  0:58.68 ./test_seekg3 /
```



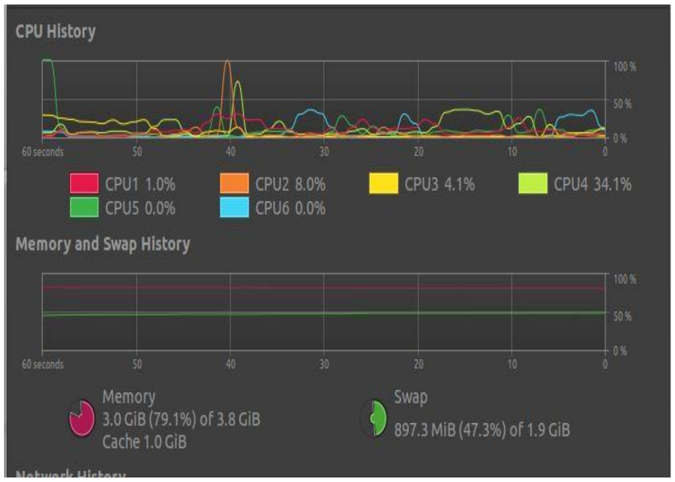
```
nmon—16k———[H for help]——Hostname=fraser-VirtuaRefresh= 2secs ——01:11.01—
 Memory and Swap
 PageSize:4KB  RAM-Memory  Swap-Space       High-Memory     Low-Memory
 Total (MB)        3935.4      1897.4       - not in use    - not in use
 Free  (MB)        2144.2      1444.0
 Free Percent       54.5%       76.1%
 Linux Kernel Internal Memory (MB)
                       Cached=     1105.6     Active=      474.8
 Buffers=       30.1 Swapcached=     46.5  Inactive =     1094.1
 Dirty  =       37.5 Writeback =      0.0  Mapped   =      128.4
 Slab   =      171.9 Commit_AS =   3881.1 PageTables=       12.3
```

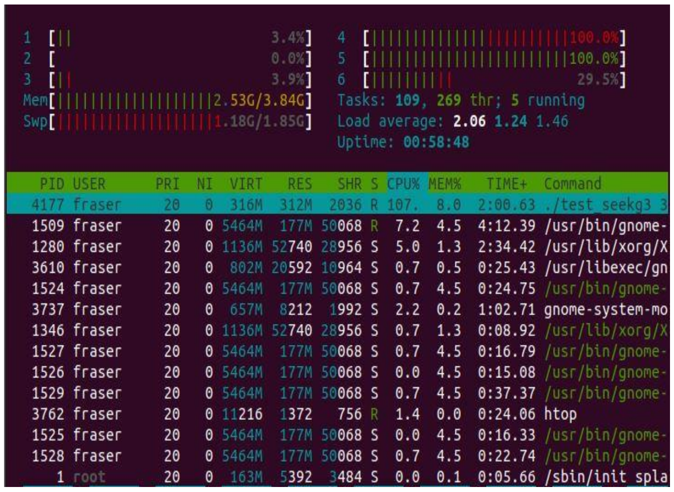


As for “GET” and “SCAN” operations, the RAM was used more but was able to manage kinda well because we have stored the keys, indexes, and values into the temp files which minimizes the toll of using the memory. In Disk I/O, disk perform read/write operations throughout the execution of the entire program. Disk read/write operation level increases and decreases depending on the input file.



```
┌nmon-16k──────[H for help]───Hostname=fraser-VirtuaRefresh= 2secs ───23:55.10─┐
│ Disk I/O ──/proc/diskstats────mostly in KB/s─────Warning:contains duplicates─│
│DiskName Busy  Read WriteMB|0          |25         |50          |75       100|
│loop0       0%    0.0    0.0|>                                                |
│loop1       0%    0.0    0.0|>                                                |
│loop2       0%    0.0    0.0|>                                                |
│loop3       0%    0.0    0.0|>                                                |
│loop4       0%    0.0    0.0|>                                                |
│loop5       0%    0.0    0.0|>                                                |
│loop6       0%    0.0    0.0|>                                                |
│loop7       0%    0.0    0.0|>                                                |
│sda       100%   20.0   12.6|RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRWWWWWWWWWWWWWWWWWW>
│sda1      100%   20.0   12.6|RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRWWWWWWWWWWWWWWWWWW>
│loop8       0%    0.0    0.0|>                                                |
│loop9       0%    0.0    0.0|>                                                |
│loop10      0%    0.0    0.0|>                                                |
│loop11      0%    0.0    0.0|>                                                |
│loop12      0%    0.0    0.0|>                                                |
│loop13      0%    0.0    0.0|>                                                |
│loop14      0%    0.0    0.0|>                                                |
│loop15      0%    0.0    0.0|>disk busy not available                         |
│Totals Read-MB/s=39.9      Writes-MB/s=25.2      Transfers/sec=4648.3          │
│                                                                               │
└         Warning: Some Statistics may not shown                                ┘
```





```
  1  [||                   3.4%]   4  [|||||||||||||||||||||||||100.0%]
  2  [                     0.0%]   5  [|||||||||||||||||||||||||100.0%]
  3  [||                   3.9%]   6  [|||||||||||            29.5%]
  Mem[||||||||||||||||||2.53G/3.84G]   Tasks: 109, 269 thr; 5 running
  Swp[||||||||||||||||||1.18G/1.85G]   Load average: 2.06 1.24 1.46
                                       Uptime: 00:58:48

    PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
   4177 fraser     20   0  316M  312M  2036 R 107.  8.0  2:00.63 ./test_seekg3 3
   1509 fraser     20   0 5464M  177M 50068 R  7.2  4.5  4:12.39 /usr/bin/gnome-
   1280 fraser     20   0 1136M 52740 28956 S  5.0  1.3  2:34.42 /usr/lib/xorg/X
   3610 fraser     20   0  802M 20592 10964 S  0.7  0.5  0:25.43 /usr/libexec/gn
   1524 fraser     20   0 5464M  177M 50068 S  0.7  4.5  0:24.75 /usr/bin/gnome-
   3737 fraser     20   0  657M  8212  1992 S  2.2  0.2  1:02.71 gnome-system-mo
   1346 fraser     20   0 1136M 52740 28956 S  0.7  1.3  0:08.92 /usr/lib/xorg/X
   1527 fraser     20   0 5464M  177M 50068 S  0.7  4.5  0:16.79 /usr/bin/gnome-
   1526 fraser     20   0 5464M  177M 50068 S  0.0  4.5  0:15.08 /usr/bin/gnome-
   1529 fraser     20   0 5464M  177M 50068 S  0.7  4.5  0:37.37 /usr/bin/gnome-
   3762 fraser     20   0 11216  1372   756 R  1.4  0.0  0:24.06 htop
   1525 fraser     20   0 5464M  177M 50068 S  0.0  4.5  0:16.33 /usr/bin/gnome-
   1528 fraser     20   0 5464M  177M 50068 S  0.7  4.5  0:22.74 /usr/bin/gnome-
      1 root       20   0  163M  5392  3484 S  0.0  0.1  0:05.66 /sbin/init spla
```