Student ID: F74077120

Name: 蔡孝龍 Marc Fraser Go Magante

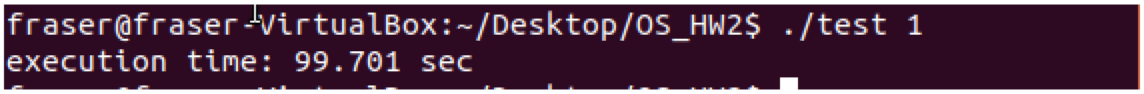
Department and Grade: Computer Science and Information Engineering (大三)

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 2.9 GB so approx. 3GB
- Programming Language (version): C++ 11

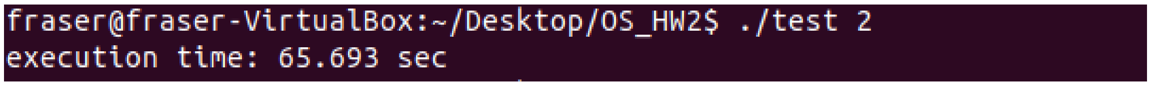
Execution Time:

- 99.701 seconds (1GB file and 1 thread)



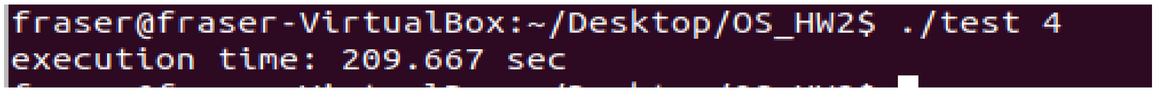
```
fraser@fraser-VirtualBox:~/Desktop/OS_HW2$ ./test 1
execution time: 99.701 sec
```

- 65.693 seconds (1GB file and 2 threads)



```
fraser@fraser-VirtualBox:~/Desktop/OS_HW2$ ./test 2
execution time: 65.693 sec
```

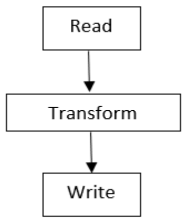
- 209.667 seconds (1GB file and 4 threads)



```
fraser@fraser-VirtualBox:~/Desktop/OS_HW2$ ./test 4
execution time: 209.667 sec
```

Development and user guide:

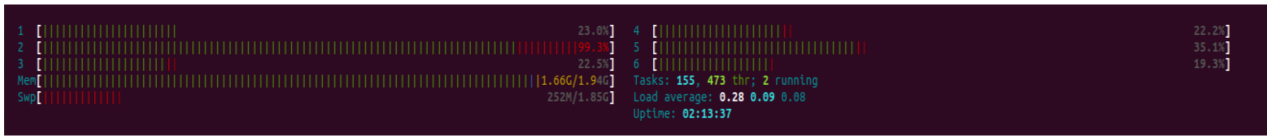
- I developed my program by firstly reading the file line by line, and in that process I have a mutex read which would let the thread identify if it's their turn or not. If it's not their turn then the thread would have to wait. Then it will go to the transforming part in which I made another getline that reads the column of the line through the help of the istringstream iss. When it detects the '|' it pushes back the number string into the string vector and depending on the for loop there is a specific format on how to output the file. So when the number is pushed back the ostringstream output would save the string. Afterwards, I would pop_back all of those string numbers so that I could use it for the next line in the file. Then I would output the ostringstream output's strings into the output file in the end and in that process I have a mutex write which would know if it's the thread's turn to write the output file or not, if not then they'll have to wait. Otherwise, they can output the file. With that, my code runs in parallel and in order/concurrently.
- Logic and flow of the program



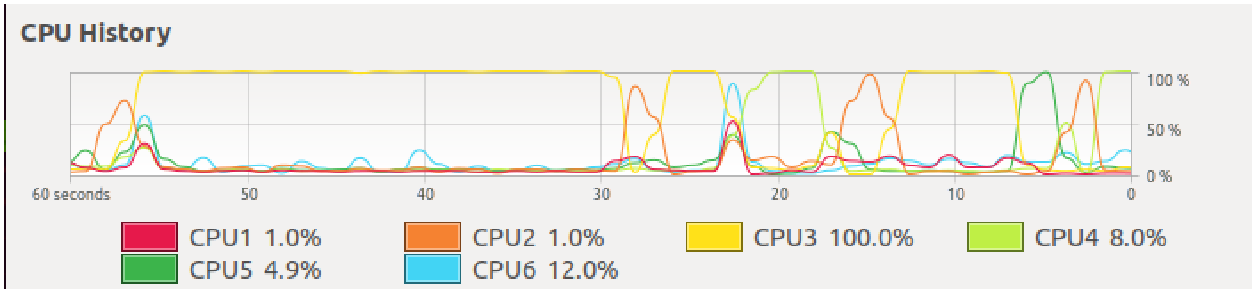
- User guide:
  ```
  # Compile
  $ g++ -pthread -o test test.cpp
  # Run
  $ ./test [threads]
  ```

Performance Analysis Report:

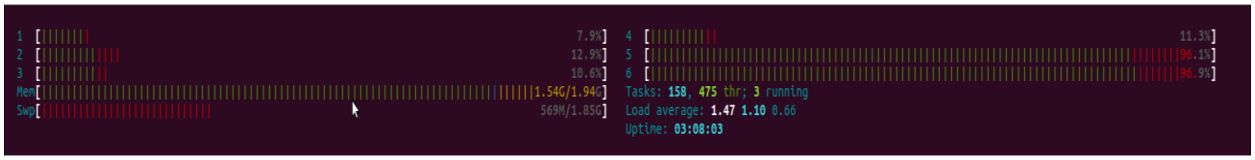
<u>1 thread</u>



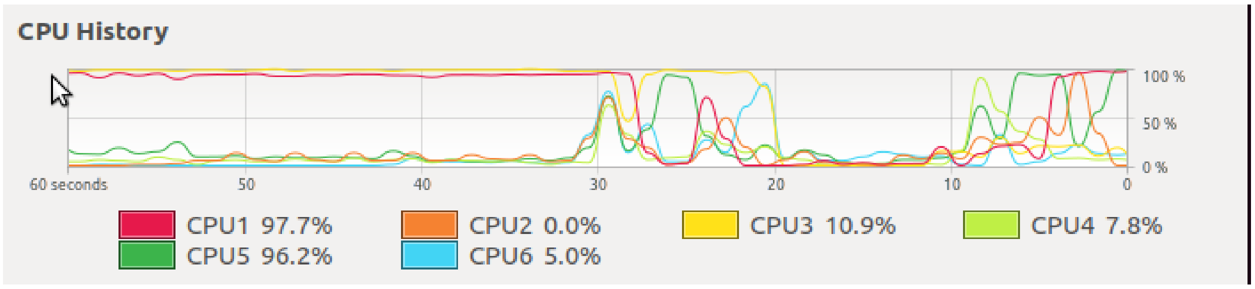
```
1  [||||||||||||||||||||||                                                        23.0%]   4  [||||||||||||||||||||||                          22.2%]
2  [|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||99.3%]   5  [||||||||||||||||||||||||||||||||||              35.1%]
3  [||||||||||||||||||||||                                                        22.5%]   6  [||||||||||||||||||||                            19.3%]
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||1.66G/1.94G]   Tasks: 155, 473 thr; 2 running
Swp[|||||||||||||                                                             252M/1.85G]   Load average: 0.28 0.09 0.08
                                                                                            Uptime: 02:13:37
```



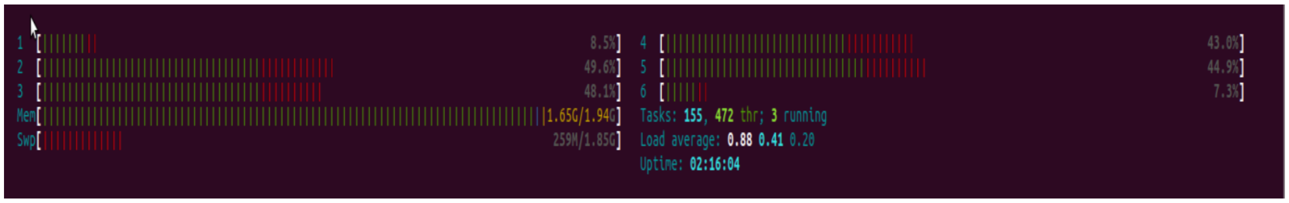
<u>2 threads</u>



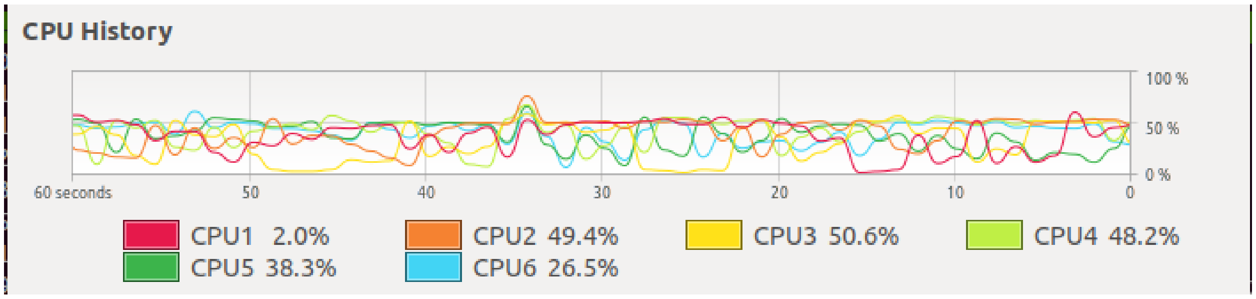
```
1  [|||||||||                                                                      7.9%]   4  [||||||||||||                                    11.3%]
2  [||||||||||||||                                                                12.9%]   5  [||||||||||||||||||||||||||||||||||||||||||||||||96.1%]
3  [||||||||||||                                                                  10.6%]   6  [||||||||||||||||||||||||||||||||||||||||||||||||96.9%]
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||1.54G/1.94G]   Tasks: 158, 475 thr; 3 running
Swp[||||||||||||||||||||||||||||                                              569M/1.85G]   Load average: 1.47 1.10 0.66
                                                                                            Uptime: 03:08:03
```



4 threads



```
1  [|||||||||                         8.5%]  4 [|||||||||||||||||||||||||||||||||||||||||||         43.0%]
2  [|||||||||||||||||||||||||||||||||||||||||||||||||||  49.6%]  5 [||||||||||||||||||||||||||||||||||||||||||||||  44.9%]
3  [|||||||||||||||||||||||||||||||||||||||||||||||||    48.1%]  6 [|||||||                              7.3%]
Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||1.65G/1.94G]  Tasks: 155, 472 thr; 3 running
Swp[||||||||||||                      259M/1.85G]  Load average: 0.88 0.41 0.20
                                                    Uptime: 02:16:04
```



Observation:

Based from the pictures, I observed that in most cases the more threads that are being used in a program, the slower the execution time will be. I think that having more threads will cause threads to fight over the CPU resource and that having more threads generally means extra work. Another reason why I think more threads makes the execution time slower is context switching. It is the process of saving the state of the current thread to start executing another thread. If a number of threads are given the same priority they need to be switched around till they finish execution. Therefore, when using multiple threads overhead occurs because of context switching which is the main reason of running the program slowly. As for the 2 threads, I think the reason why it has the fastest execution time is because one thread is working on the CPU while the other is waiting for disk I/O. Another reason is by the use of Amdahl's law whereby in any program there is always a percentage that cannot be run in parallel and there is another percentage that can be run in parallel. To optimize further optimize, '\n' should be used instead of endl for another line and also I think its better to have a queue where in for a thread they would be doing multiple lines at a time instead of just one line at a time.