S3774430 Myeonghoon Sun https://github.com/hooniez/osp_assessment1

Assignment 1

The Source Data

I have downloaded all the 12 files, and combined them into *combined.txt* in *original_data* so that it can be used as input data for Task 1 (the input data does have duplicate words).

Task 1

- the filtering rule: select words that are made solely of English letters without special symbols such as the apostrophe, comma, and hyphen.

- the coreutils tools used: awk (to use regex to apply the filtering rule) and sort (to remove duplicates)

- the number of words of length 3 to 15 letters in the dataset: 1,367,007 words

- the performance on my machine (Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz):

  - time ./Task1.sh => 5.319s (an average time of 5 executions)
  - time ./Task1Filter ../original_data/combined.txt ../input.txt => 5.751s (an average time of 5 executions)

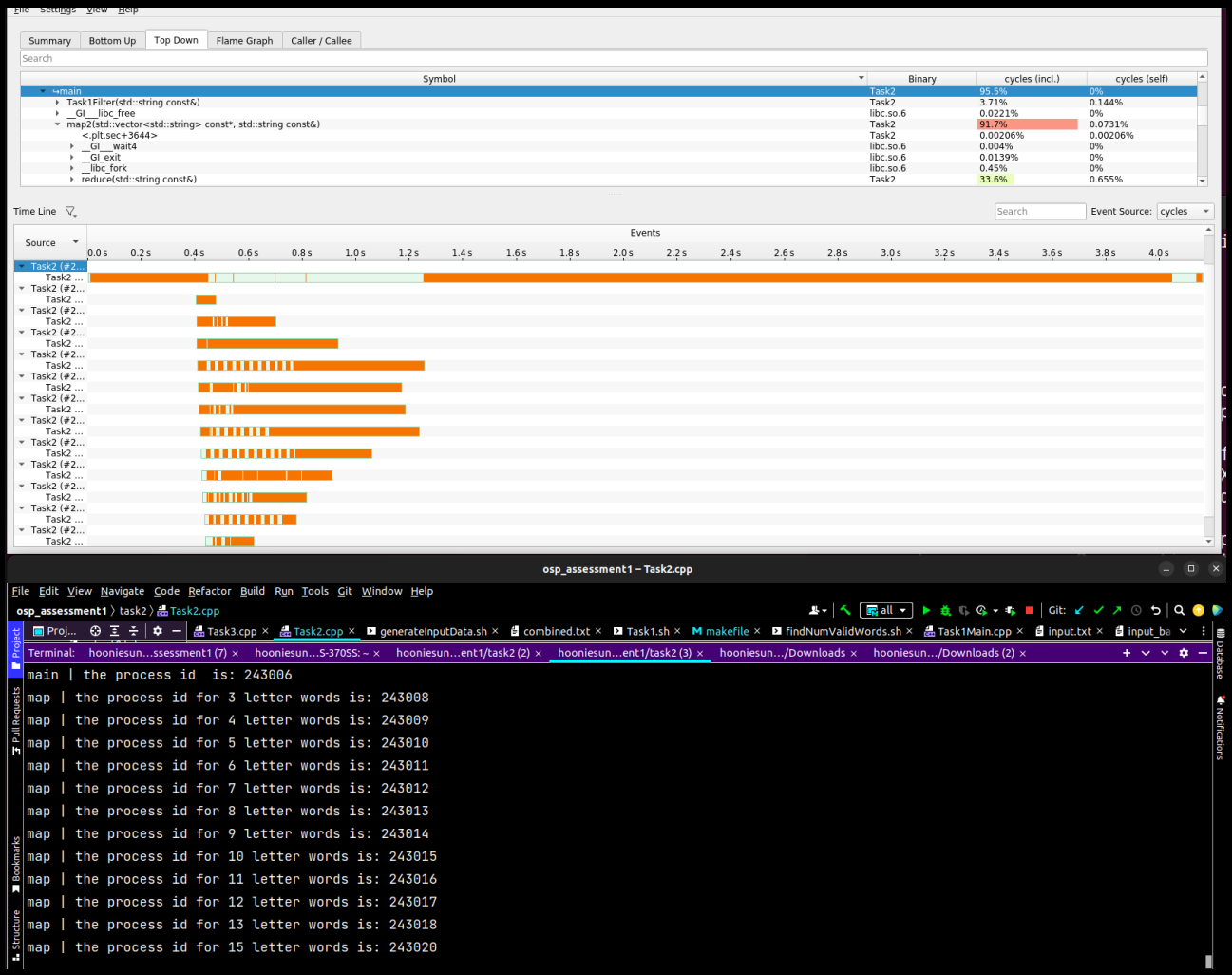
Many hours were spent to come up with a way to shuffle both of the filtered datasets such that they are identical. Because of the limitation not to use the same coreutils tools used in the bash script in the corresponding C program, I couldn’t just use the system() call in my C program to shuffle identically. I finally devised a simple yet effective way to shuffle b y creating a 2d array to store the filtered data and then transposing it both in the bash script and the C program only to be told by the course coordinator that shuffling is in fact unnecessary at odds with the specification. I believe this implementation, which is now commented out, be recognised as a deed exceeding the assignment requirements.

Task 2

- the performance (I have chosen to use the perf tool because it allows for profiling on per-thread and per-process basis, which is necessary to optimise thread performance as required in Task 4; *perf record* collects samples for event *cycles*, which then can be analysed to find out the workload of each thread or process):

- Run Time: 3.962s

As seen below, it takes twice as much of the total clock cycle for the map2() to sort and write outputs as for the reduce2() to perform 13 to 1 merge operations; a 91.7% of the total clock cycle can be attributed to operations within map2() in which reduce2() is also called with its share being 33.6% of the total clock cycle.

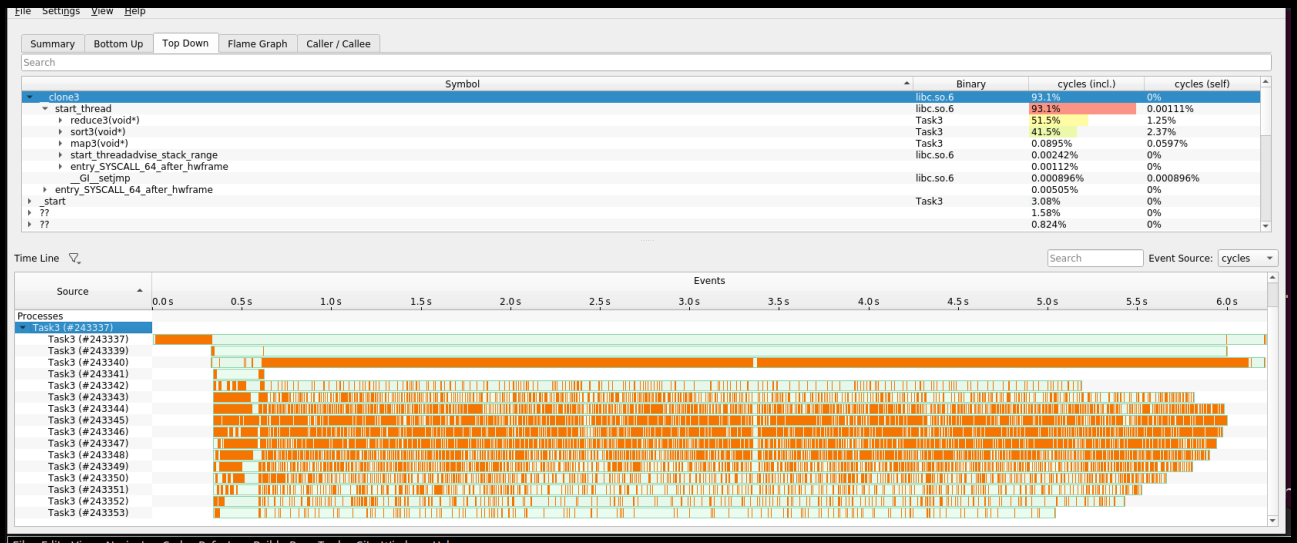


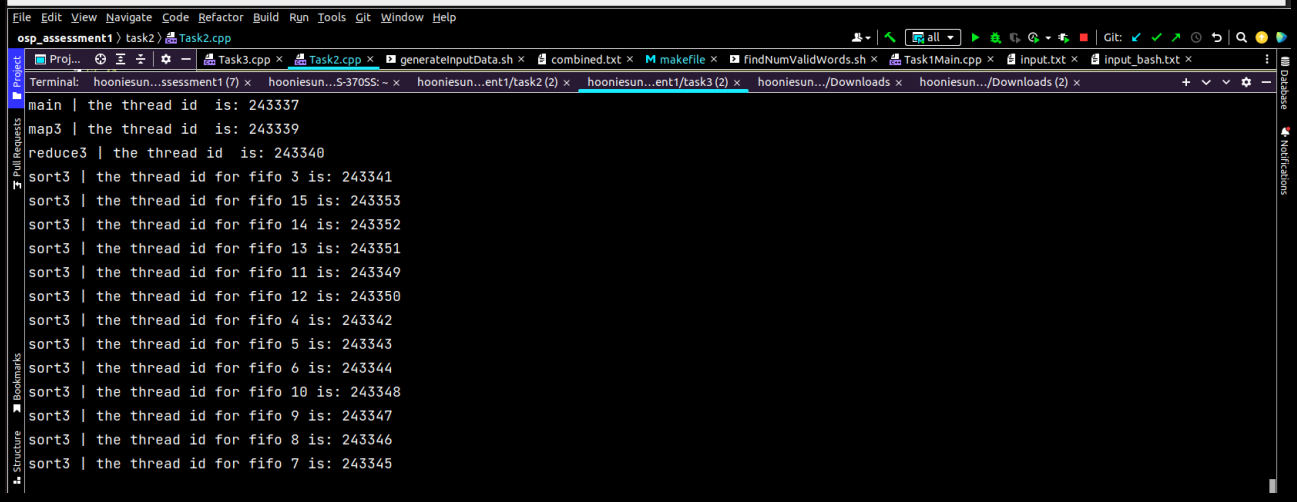
Task 3

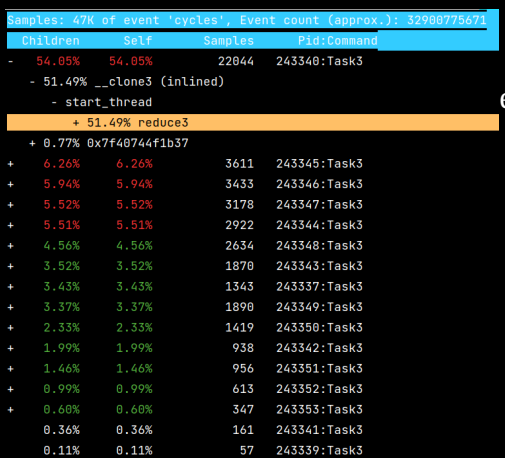
- the performance:

- Run time: 6.225s

As seen below, the reduce3() is responsible for 51.5% of the total clock cycle, which is a significantly greater proportion than the reduce2()'s share of its own clock cycle.







The left screenshot shows in more detail the workload of each thread including the one executing reduce3().