Software Systems - Concurrency Assignment

Jiacong Li (5656354, jiacongli@tudelft.nl) Tongyun Yang (5651794, tongyunyang@tudelft.nl)

1 Description

The aim of this assignment is to implement a concurrent version of grep as mentioned in [1]. As described in [2] grep is a command-line utility for searching plain-text data sets for lines that match a regular expression. The program must be implemented based on a certain given structure, and no external libraries besides the given ones are allowed.

A checklist for all the baseline and extra requirements is made. The full checklist is delivered in the git repository on GitLab[1] in the README.md file on the "1-concurrency" branch. The checklist is a direct display of what is being delivered in the assignment. Some requirements are discussed later in section 3, and the same discussion could also be seen in the README.md file. Note that the explanation of what concurrency primitives are used and why they are used is also delivered in the same section 3 later.

2 Performance Analysis

After implementing the program, the performance analysis is done. Two testing platforms are used, they are displayed respectively below:

- Apple MacBook Air 2020
  - 8 Cores 8 Threads Apple M1
  - 8 GB RAM
  - macOS 13
- Dell G15 5511
  - 8 Cores 16 Threads Intel i7
  - 16 GB RAM
  - Windows 10

All benchmarks below run the same program with "**time cargo run –release torvalds .**". Benchmarks are done ten times (result for the first run is expelled) and averaged to get a clearer result. The baselines below run the same command - "**time grep -r torvalds .**". Baselines are done ten times (result for the first run is expelled) and averaged to get a rather fair result. Note that for a better comparison and display, the performance is calculated as $\frac{1}{ExecutionTime}$.

2.1 Dell

Figure 1 shows clearly that the performance of the program on Dell rises as the number of threads increases. The performance is better than the baseline (the grep command installed on windows) when the number of threads is beyond 4. This pattern continues until the number of threads reaches 128. An interesting fact is noticed, the processor on Dell only has 16 threads, but the performance reaches its peak when 128 threads are launched in the program. The answer to this issue is not figured out.

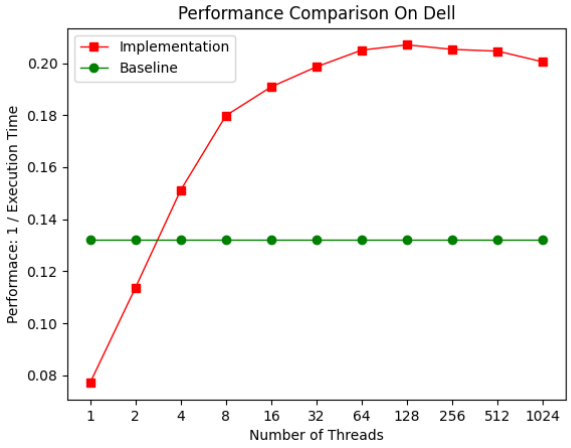

Figure 1: Dell performance graph

2.2 MacBook

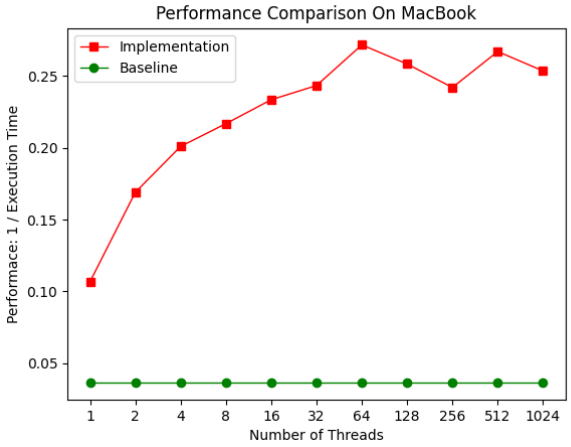

Figure 2: MacBook performance graph

Figure 2 displays the comparison of performance of the baseline (the grep function on macOS) and the program on macOS. The performance of the baseline on macOS is insanely slow. An immature guess to this issue is that it has not been optimized much for the newly released Apple Silicon. While running the base-

[1] A DevOps software package that combines the ability to develop, secure, and operate software in a single application.

line and monitoring the htop[2] interface at the same time, it is also found that the load on each core of the processor is very low.

Following a similar pattern as seen in the performance of Dell in section 2.1, the performance of MacBook also reaches its peak when 64 threads are launched in the program. The answer to a similar question - "Why a processor with 8 threads reaches its peak performance when the program launches 64 threads?" remains unknown.

2.3 Dell VS MacBook

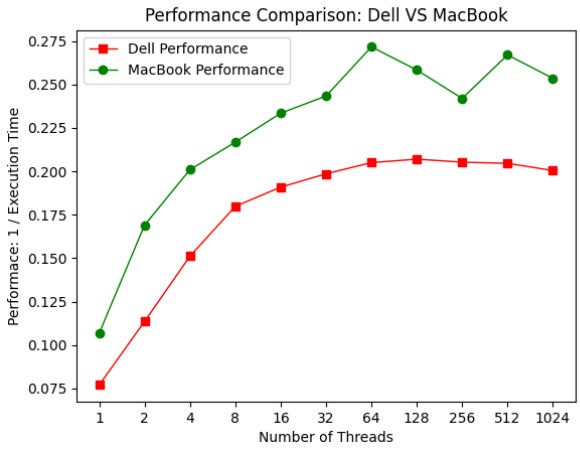

Figure 3: Dell and MacBook performance comparison

Figure 3 shows a comparison of the performance of the same program running on Dell and MacBook. Overall, the performance on MacBook is better.

3 Discussion

In section 2, the analysis is done with a wide range of threads launched in the program. However, by default, the number of threads launched running the program differs among machines. It is equal to the maximum number of threads the machine could handle. In this case, 8 threads for MacBook and 16 threads for Dell.

Besides the baseline, most of the extra requirements are met in our program. However the "Good error handling" is not certain whether it is fully realized, due to lack of verification. For most functions implemented, error handling with user friendly messages are included. Except the situation when the program encounters files that it does not have permission to read. It also remains unsure if every single potential error is dealt with properly.

Another issue regarding the performance of the program in respect to the number of threads launched in the program requires further analysis. The peak performance of the program reaches its peak performance on both machines with different operating systems when 128 and 64 threads are launched respectively. However, according to the CPU hardware parameters, the CPU in two machines should only be able to handle 16 threads and 8 threads at the same time. The answer to this issue remains unknown and requires further analysis.

The only two concurrency primitives used in the program are a counter of type Mutex which counts the numbers of files that match the regex, and a channel sending "GrepResult" between two threads. The counter of type Mutex is necessary because it guarantees the safety of the value in counter. Due to the fact that Mutex could only be accessed when it is locked, and block threads waiting for the lock to become available. Hence, only one thread is adding to the counter each time. The second concurrency primitive is a channel which sends "GrepResult" found in every file to another thread for printing out the result. This is due to the requirement of the assignment - "Use at least one non-trivial channel.". In our case, two different type of threads are created. One only consists of a thread which loops and prints out the result in a provided format once it receives "GrepResult" via the channel. The other one consists of several threads which reads a file respectively and sends the "GrepResult" via the channel once the single file is done reading and matches are found.

4 Conclusion

Overall, the program is faster than the baseline throughout all our tests (when number of threads launched are the maximum that the machine could handle). The speedup is 136.03% on Dell (with 16 threads launched in the program) and 642.96% on MacBook (with 8 threads launched in the program). Most of the requirements are fully realized. Though answers to certain issues remain unknown, they are discovered and reasonable guesses for the cause are given.

References

[1] "Software systems," Technische Universiteit Delft, (Date last accessed 11-18-2022). [Online]. Available: https://software-fundamentals.pages.ewi.tudelft.nl/software-systems/website/index.html

[2] "grep," From Wikipedia, the free encyclopedia, (Date last accessed 11-18-2022). [Online]. Available: https://en.wikipedia.org/wiki/Grep

[2]htop is an interactive system-monitor process-viewer and process-manager. It is designed as an alternative to the Unix program top.