COP290 Lab 2

Dhruv Gupta 2021CS50125
Niranjan Sarode 2021CS50612
Dhruv Verma 2021CS51086
Team Name : DND

Abstract

This report answers the questions asked in the lab 2 assignment. Find the attached files alongside this report for complete information.

Part 2

Question 1 : Does the output vary?

Yes the output varies upon changing between Part 2 and Part2 with *yield_context*. Here is the table depicting a few examples of changes in output upon these given inputs.

Text Files				CheckPoint 2	CheckPoint2 with *yield_context*
hello world test	hello world	hello world	hello world	key hello , count 4 key test , count 1 key world , count 4	key hello , count 2 key test , count 1 key world , count 2
hello world test	hello world	world	hell testing	key testing, count 1 key test, count 1 key hello, count 2 key world, count 3 key hell, count 1	key testing, count 1 key test, count 1 key hello, count 2 key world, count 2 key hell, count 1
test test test	test test test	test test test	test test test	key test, count 12	key test, count 4

Table 1: Test Case Examples

Question 2 : Why do you think the output varies?

It occurs due to Race Conditions.

Examplewise exlpanation given below:

In First Test case: Words are parsed in this order:

START → hello (file 1) → world (file 1) → test (file 1) → hello (file 2) (yield) → hello (file 3) (yield) → hello (file 4) (yield) → hello (file 2) → world (file 2) (yield)→ world (file 3) (yield) → world (file 4) → END

Here because of yield when the hello (file 3) (yield) → hello (file 4) (yield) is called so the c* value is compared before being updated .To avoid it we switch it again till it reached the required file which can be done using locks. Without lock, context-switching allows even the other context to access the shared resource which leads to variation in output.

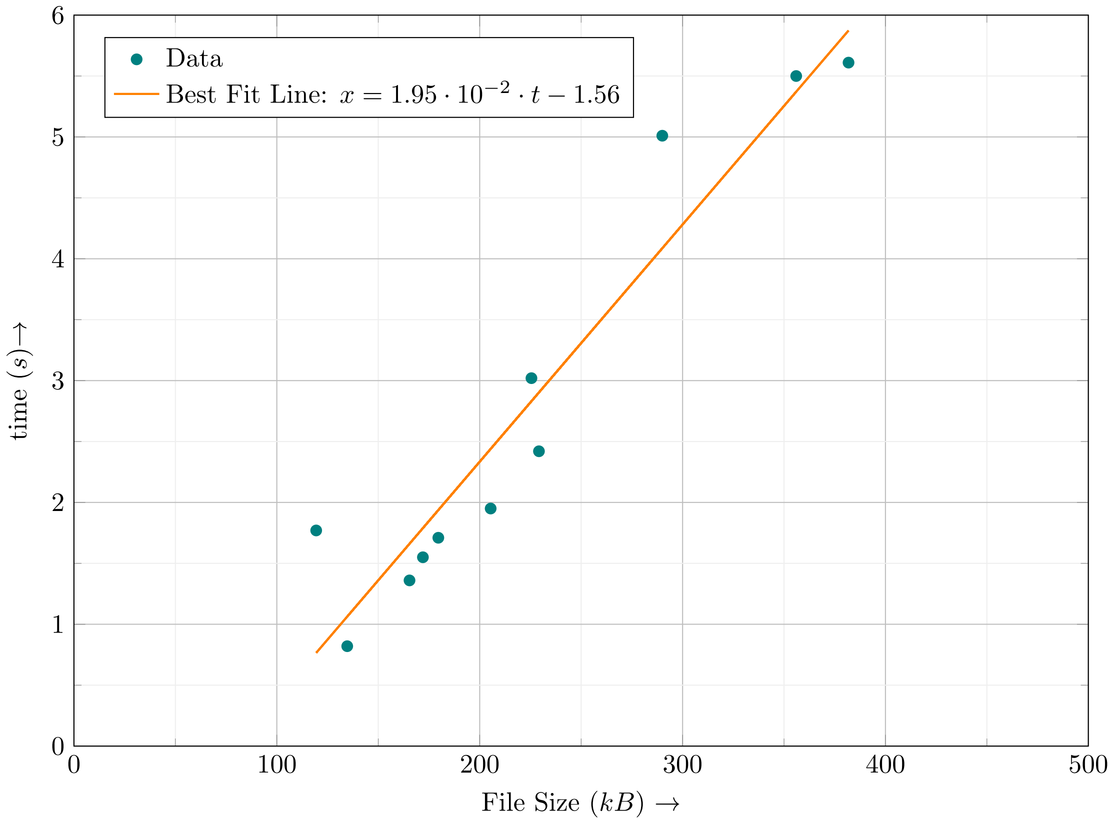
Similarly in next examples whenever a repreated word is encountered the context used to access that is switched before updating which lead to entering hashmap once instead of twice in some cases which leads to less count of words in output in some cases.

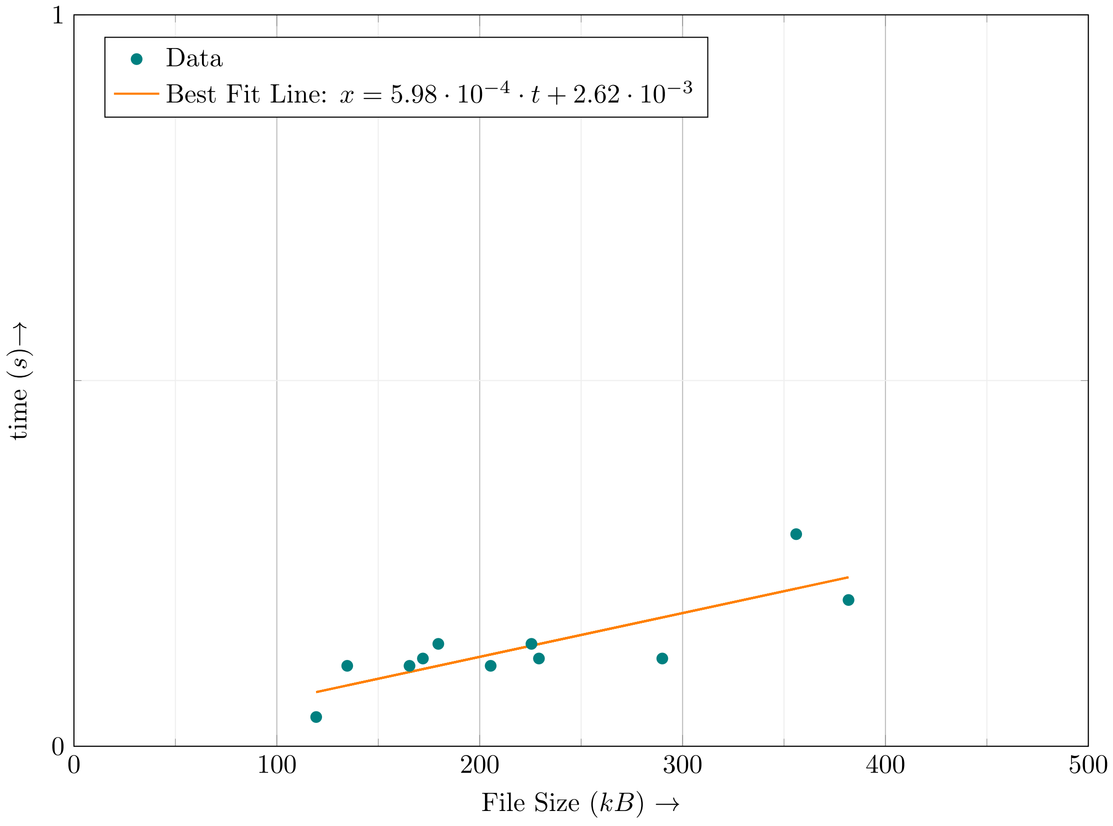
Race conditions occurs when two or more contexts attempt to access and modify the same global variable or shared resource. Here when yielding is done the context is switched to another context and the other context is allowed to access the shared resource. This leads to the output varying. To prevent it we can use synchronization mechanisms such as locks to ensure that only one context can access a shared resource at a time, preventing conflicts and ensuring that the resource remains in a consistent state.

Comparative analysis of runtimes of Part 2 and Part 3

Plotting tables and graphs for comparative analysis between runtimes of part 2 and part 3. The first graph represents the runtime of part 2 (with yield and locks) versus filesize . The first graph represents the runtime of part 3 versus filesize .

Large text files to benchmark taken from https://www.gutenberg.org/browse/scores/top.





From the data we can see that multithreaded program works way faster than single threaded

S.No.	File Size (kB)	Part 2	Part 3	Link
1	179.6	1.71	0.14	https://www.gutenberg.org/files/2701/2701-0.txt
2	134.7	0.82	0.11	https://www.gutenberg.org/files/2160/2160-0.txt
3	165.4	1.36	0.11	https://www.gutenberg.org/cache/epub/84/pg84.txt
4	381.8	5.61	0.2	https://www.gutenberg.org/cache/epub/70054/pg70054.txt
5	172.0	1.55	0.12	https://www.gutenberg.org/files/74/74-0.txt
6	356	5.5	0.29	https://www.gutenberg.org/cache/epub/4363/pg4363.txt
7	119.4	1.77	0.04	https://www.gutenberg.org/cache/epub/15399/pg15399.txt
8	229.2	2.42	0.12	https://www.gutenberg.org/files/2680/2680-0.txt
9	290.0	5.01	0.12	https://www.gutenberg.org/files/140/140-0.txt
10	225.5	3.02	0.14	https://www.gutenberg.org/files/1998/1998-0.txt
11	205.4	1.95	0.11	https://www.gutenberg.org/cache/epub/37106/pg37106.txt

Table 2: Comparative Analysis of runtimes of Part 2 and Part 3

program with multiple concurrent stacks. Data in the table represents runtime in seconds.

Variation on basis of number of input files (Part 3)

To Test this we use the files

- File 1 : https://www.gutenberg.org/files/74/74-0.txt (172.0 kB)
- File 2 : https://www.gutenberg.org/cache/epub/15399/pg15399.txt(119.4 kB)
- File 3 : https://www.gutenberg.org/cache/epub/37106/pg37106.txt(205.4 kB)
- File 4 : https://www.gutenberg.org/files/1998/1998-0.txt(225.5 kB)

S.No.	Files Tested	Runtime Part 2 (in s)	Runtime Part 3 (in s)
1	File 1, File 2, File 3 , File 4	29.28	0.39
2	File 1, File 2, File 3	14.32	0.24
3	File 1, File 2	7.5	0.18
4	File 1	1.46	0.11

Table 3: Testing Multiple Files

On increasing the number of input files the runtime increases but the increment in much higher in part 2 than in part 3. This is because we are using single thread with yield and locks in part 2 and multiple threads in part 3.

Variation on basis of size of input files (Part 3)

Refer to table 2 for data. and the second graph for variation .From this we can see that as the size of input files increases the runtime increases.

Variation on basis of repeated words in file (Part 3)

For this we took a 300 word text and copied and pasted over and over to obtain larger size .

S.No.	File Size	Runtime in Part 2 (s)	Runtime in Part 3 (s)
1	179.8kB	2.6	0.01
2	359.5kB	8.25	0.03
3	719.0kB	23.49	0.05
4	1126.3kB	47	0.10

Table 4: Variation in runtime for repeated word file

As the repetitions increase the Part 3 takes way less time than in Part 2.

Token distribution

Thw work was done by all 3 of us together and equally (not distributed partwise) so we distribute 10 tokens each of us.

S.No.	Token Receiver	Total Tokens Received
1	Dhruv Gupta	10
2	Dhruv Verma	10
3	Niranjan Sarode	10

Design Decisions and Assumptions

- In Part 3 main.c doesnt prints key and value correctly. We replaced
  ”$printf(”key\%s, count\%d\backslash n”, e- > key, *count);$”
  by ”$printf(”key\%s\backslash n”, e- > key); printf(”count\%d\backslash n”, count);$”
- Input files must end in newline character.
- Input files has no non-ascii characters.
- Paths to header accessed in test files are corrected to relative paths.