CS 610 Semester 2024–2025-I: Assignment 2

26th August 2024

Due Your assignment is due by Sep 6, 2024, 11:59 PM IST.

General Policies

- You should do this assignment ALONE.
- Do not copy or turn in solutions from other sources. You will be PENALIZED if caught.

Submission

- Submission will be through Canvas.
- Submit a compressed file called "⟨`roll`⟩`-assign2.tar.gz`". The compressed file should have the following structure.

```
-- roll
-- -- roll-assign2.pdf
-- -- <problem1-dir>
-- -- -- <source-files>
-- -- <problem2-dir>
-- -- -- <source-files>
```

  The PDF file should contain descriptions for the first two problems, and your solution for the last problem.

- We encourage you to use the LaTeX typesetting system for generating the PDF file. You can use tools like Tikz, Inkscape, or Draw.io for drawing figures if required. You can alternatively upload a scanned copy of a handwritten solution, but MAKE SURE the submission is legible.
- You will get up to TWO LATE days to submit your assignment, with a 25% penalty for each day.

Evaluation

- Write your programs such that the EXACT output format (if any) is respected.
- We will evaluate your implementations on a Unix-like system, for example, recent Debian-based distributions installed on KD first floor labs.
- We will evaluate the implementations with our OWN inputs and test cases, so remember to test thoroughly.

Problem 1 **[40 marks]**

You are given a multithreaded program with N threads. The program reads N files, and should report the total number of words and lines processed by all the threads.

We provide a driver source code for the problem. Your task is to analyze the source code, and identify and report the performance bugs present in the source code, if any. If a performance bug is identified, provide a manually fixed version. Describe your modifications to the source code and report the performance gain.

Use the following commands to compile the attached driver and run the sample test case.

```
make
./problem1.out 4 ./test1/input
```

You can use the `perf c2c` tool to identify some forms of performance bugs. You can use the following links to learn more about using `perf`.

- C2C - False Sharing Detection in Linux Perf
- perf c2c man page

Input The input to your program will be a path to a file, say `input`.

The file `input` lists the full paths of N source files that are to be processed by N threads. Read the contents of the file `input` into a shared data structure X. Each thread will then pick *one* file from the shared data structure to analyze. A thread reads its file one line at a time and divides the line into tokens. The thread updates a counter to track the words encountered by the threads and updates the shared variable to track the total number of places the line.

Example

```
File 1:
ABC, EFG HIJK.
LMNOP QRST.

File 2:
ABC EF HI LMNOPQ
RST UV

Expected Output:
Thread 0 counter: 6
Thread 1 counter: 5
Total words processed: 11
Total lines processed: 4
```

Problem 2 **[50 marks]**

Write a C++ program that takes five arguments from the command line: a string that represents the path to an input file to be read (say R), an integer representing the number of producer threads (say T), an integer representing the number of lines each thread should read from the file (say L), an integer representing the size of a shared buffer (say M), and a string that represents the path to an output file (say W).

Assume the file R to be read contains N lines. The program will launch the required number of threads T. AFTER all the threads are created, the threads will CONTEND for access to R to repeatedly read L consecutive lines. Then, each thread will write its share of L consecutive lines ATOMICALLY to a FIFO SHARED buffer. A dedicated consumer thread (not included in T) KEEPS READING from the shared buffer and writes its contents to the destination file W.

- N may not be a multiple of L. In such cases, the last reader thread will read whatever lines are left.
- Blank lines are also counted.
- The input file can have more than T*L lines.
- The problem requires synchronization (barriers, mutexes, and condvars) at multiple places.
- All threads should acquire locks only for the minimum required duration (e.g., reader thread reading L lines).
- Use a conditional variable instead of busy waiting for synchronizing accesses to the shared buffer across producer and consumer threads.
- You are not allowed to use concurrent data structures, but you can use other STL data structures like `std::vector` and `std::atomic`.
- The application should terminate properly after all threads are done.

The goal of this problem is to achieve correctness. So, test your code thoroughly with different possible inputs.

Problem 3 **[10 marks]**

Consider the following loop nest.

```
for i = 1, N-2
  for j = i+1, N
    A(i, j-i) = A(i, j-i-1) - A(i+1, j-i) + A(i-1, i+j-1)
```

List all flow, anti, and output dependences, if any, using the Delta test. Show your computation. Assume all array subscript references of array A are valid.