COSC 3P95- Software Analysis & Testing

Assignment 2

Due date: Sunday, Nov 19th, 2023, at **23:59** (11:59 pm)

Attention:

- This is a **group** assignment designed for a maximum of two students. You have the option to complete the assignment either individually or as part of a two-person group.
- Combine all textual and visual elements into a single PDF file. This includes explaining code snippets, screenshots, and your report. Make sure this PDF is well-organized and easy to navigate.
- You may also include a zip file containing the complete source code, Open-Telemetry configurations, and any other relevant files.
    - Alternatively, you can provide a link to a GitHub repository that contains all these elements. If you choose this method, ensure you have met the criteria outlined in the "Extra Credit Assignment."
- Each group should submit only one set of files. Make sure to clearly indicate the names and student IDs of all group members in your submission.
- In the introduction of your assignment, provide a detailed explanation outlining the contributions made by each group member. Specify what parts of the assignments each person worked on and the extent of their involvement.
    - Failure to include this section may result in a 15-point reduction for each group member.
- This assignment has 100 + 10 (bonus) points and is worth 10% of the course grade. Please also check the Late Assignment Policy.

Name1: _____ **Student ID 1:** _____

Name2: _____ **Student ID 2:** _____

Questions:

1- Using your favorite programming language, develop a client-server program that reads from a local folder and sends file contents to a server, implementing Open-Telemetry for tracing and performance monitoring. **(50 pts)**

Detailed requirements:

- The local folder should contain at least 20 files with varying sizes, from 5KB to 100MB. The goal is to have a mix of small and large files. You can generate these files randomly or download them from the internet.
- Implement both the client and server applications. The server should be multi-threaded using the fork/join parallel pattern. Write received files into a local folder on the server side.
- Use both Open-Telemetry **auto** and **manual** instrumentation.
- Implement and analyze at least two different sampling strategies: AlwaysOn and Probability sampling (below a 40% rate).
- Select and implement at least **three** of the advanced features below.
    a) Encryption & Security: Add a layer of encryption to secure the content transfer.
    b) Compression: Use a data compression and decompression algorithm to enhance data transfer efficiency.
    c) Chunking & Streaming: Enable content chunking and streaming to better handle large files.
    d) Rate Limiting: Implement rate limiting to control the volume of data transfer requests.
    e) Error Handling & Retry Logic: Add error-handling and retry mechanisms.
    f) Concurrency Models: Implement an alternative concurrency model to the fork/join pattern. Or in the fork/join model, limit the number of worker threads (the maximum number of threads the server can fork to handle the requests).
    g) Data Integrity: Use checksums to ensure the integrity of transferred data.
- Use Open-Telemetry to collect relevant data or metrics with and without the advanced features.
- Use an Open-Telemetry compatible external tool like Jaeger, Prometheus or Grafana to visualize the spans and telemetry data.
- Evaluate how each selected feature or the combination of them impacts the system's performance in terms of latency (time), throughput (speed), error rates or other factors.
- Discuss your observations and findings in your assignment.

Deliverables:

- Source code for both the client and server applications.
    - Provide comments in the code.
- Open-Telemetry and visualization tool configuration/data files.
- Screenshots or screen recordings of the visualization dashboard.
- A report explaining implementation details, performance analysis, findings and any challenges faced.

2- Extend the client-server application developed in Question 1 by introducing a deliberate bug or misconfiguration. Use Statistical Debugging (SD) techniques to identify and resolve the issue, leveraging Open-Telemetry for collecting/visualizing data. **(50 pts + 5 bonus)**

Detailed requirements:

- Insert a bug or misconfiguration into either the client or server application from Question 1. This could be within the core functionalities or one of the advanced features. Such as:
    - Encryption algorithm not initializing properly.
    - Rate-limiting feature incorrectly blocking or allowing requests.
    - Data compression algorithm causing data loss or corruption.
    - Fork/join pattern misconfiguration leading to thread leaks or slowness in the system.
    - A deliberate delay in part of your client or server which happens only in some cases. For example:

```
if (randomNumber(0, 1) < 0.3) {  // 30% chance to trigger the delay
        deliberateDelay();
}
```

    - Or any other code-level bugs.
- Use Statistical Debugging (SD) technique explained in the classroom to define the appropriate predicates and collect enough data (samples) for your executions. Then use the metrics (Failure, Increase, etc.) to analyse your data and identify the root-cause of the bug.
- **Optional** (bonus) mark (**5 pts**):
    - Once identified, resolve the issue, and validate that it has been effectively removed.
    - Use Open-Telemetry to compare, analyse and explain the data/metrics before and after the resolution.

Deliverables:

- Modified source code for the client-server application with the introduced (bonus: and then resolved) bug or misconfiguration.
- Updated Open-Telemetry configuration files and visualizations if any changes were needed.
- A report, detailing the introduced bug, SD techniques applied for debugging, analysis (predicates, metrics, etc. helped in identifying the bug) and other findings.
    - Bonus: include steps taken to fix the issue and validate the fix.

3- **Extra Credit Assignment**: Create a GitHub repository to host all the elements of this assignment. This includes source codes, test data, and any screenshots or logs you have generated. Submit the GitHub link along with your main submission through Brightspace. **Initially,** set the repository to 'Private'. Then, make the repository 'Public' the day after the assignment deadline, or keep the repository 'Private' and add the TAs and me to the repository. Use the university email addresses for granting access. **(5 bonus pts)**

Marking Scheme:

Marks will be awarded for completeness and demonstration of understanding of the material. It is important that you fully show your knowledge when providing solutions in a concise manner. Quality and conciseness of solutions are considered when awarding marks. Lack of clarity may lead you to lose marks, so keep it simple and clear.

Submission:

*The submission is expected to contain a sole word-processed document. The document can be in either **DOC or PDF** format; it should be a single column, at least single-spaced, and at least in font 11. It is strongly recommended to use the assignment questions to facilitate marking: answer the questions just below them for easier future reference.*

Late Assignment Policy:

A one-time penalty of 25% will be applied on late assignments. Late assignments are accepted until the Late Assignment Date, four days after the Assignment Due Date. No excuses are accepted for missing deadlines. However, deadline extensions may be granted under extenuating circumstances, such as medical or physical conditions; please note that granting the extension is under the instructor's discretion. However, deadline extensions may be granted under extenuating circumstances, such as medical or physical conditions; please note that granting the extension is under the instructor's discretion.

Plagiarism:

*Students are expected to respect academic integrity and deliver evaluation materials that are only produced by themselves. Any copy of content, text or code, from other students, books, web, or any other source is not tolerated. If there is any indication that an activity contains any part copied from any source, a case will be open and brought to a plagiarism committee's attention. In case plagiarism is determined, the activity will be canceled, and the author(s) will be subject to university regulations. For further information on this sensitive subject, please refer to the document below: **https://brocku.ca/node/10909***