Programming Assignment 4

Instructions

- Due by 11:59pm of April 13, 2021.
- Late penalty: **10%** penalty for **each day late**.
- This is an individual assignment, although you may work in groups to brainstorm on possible solutions; your code implementation, report, and evaluation must be your own work.
- Upload your assignment on the **Blackboard** with the following name: **Section_LastName_FirstName_PA4.zip**
- Please do **NOT** email your **assignment** to the **instructor** or **TA**!
- Use any popular language, if in doubt ask the TA ASAP.

Overview

In this assignment, you are required to change PA2 to adding parallelism in data transfer within a single file. In PA2, a peer node downloads a file from a single node. However, a file could be broken into a set of fixed size chunks while transferring. Therefore, a peer node could download these chunks from multiple peer nodes in parallel and then make the original source file. You will need to check for the integrity of the data for each chunk to ensure the file is the same as the original one.

Detailed Description of Assignment

Here you add parallelism to the data transfer of a file. You will download different chunks of a file from different nodes in parallel.

You will have to change the node in the following ways:

- Peer Node:
    - As a server,
        - It registers all the files it holds with **their file sizes** to the indexing server, when it starts up.
        - It can **send partial file to the client**. **Partial file** means a chunk of the file. The chunk size is fixed and is the same for all the peer nodes (e.g., a chunk is 64KB)
    - As a client, when requesting to download a new file,
        - Send a query request to the indexing server. The server will return two stuffs:
            - A list of peer nodes that holds the request file
            - The size of the requested file
        - Download this file in parallel.
            - Break the requested file into multiple fixed-size chunks.
            - Download these chunks from multiple peer nodes, such as chunk1 from peer node1, chunk2 from peer node2, etc. Please note that

the **download** request should include the start offset and the chunk size.
        - Arrange the chunks order and create the final requested file.
    - When finish downloading the file,
        - You need to check the integrity of the data to make sure that the downloaded file is the same as the original file.
        - The peer node should update the file information with the indexing server as usual.
    - **Peer Node output**: For every file downloaded, you need to track which node each chunk# comes from, with a timestamp,
- The Central Indexing server node:
    - It maintains a list of active nodes and the files they hold
    - It also maintains the information about each file size
    - When receiving a query request from a peer node, it needs to return two stuffs to the client:
        - A list of peer nodes that have the requested file.
        - The size of the requested file.

Note: The core functionalities of the peer node and Indexing server node have not changed and should support all the functionalities from the PA2.

Evaluation

1. All the peers can be setup on the same machine or different machines.
    a. Ensure you can download one file properly
    b. Ensure multiple peer nodes can simultaneously upload and download files
2. Measuring how the transfer speed changes when varying the number of nodes that holds the same requested file.
    a. Deploying 1 indexing server and a bunch of peer nodes
    b. Varying the number of peer nodes (N) that holds 10 copy of the same files (N: 2, 4, 8, 16)
    c. Observe how the transfer speed changes when a peer node joins and requests to download these 10 files. (Each file is at least 1MB)
    d. Graph your results.

Submission Information

When you have finished implementing the complete assignment as described above, you should submit your solution to the blackboard. Each program must work correctly and be well documented. You should hand in:

1. **Source Code (25 points)**: You must hand in all your source code, including with in-line documentation.
2. **Makefile/Ant/requirements (5 points)**: You must use Makefile or Ant to automate your programming assignment compilation or a requirements file to download any

dependencies. If using external libraries (mainly for compiled languages) put them in a folder marked external. **Note:** The external library cannot do the heavy lifting of the assignment tasks for you. If in doubt, please reach out to the TA.

3. **Deployment scripts (10 points)**: You must provide a deployment script for the different nodes.
4. **Readme (10 points)**: A detailed manual describing how the program works and how each evaluation scripts run. The manual should be able to instruct users other than the developer to run the program step by step.
5. **Compiles Correctly (10 points)**: Your code must be compiled in a Linux environment.
6. **Output files & Performance Evaluation (25 points)**: A copy of the output generated on running your programs for each of the evaluations. For (2) in evals, provide the output for each level scaled. Write a report for each evaluation and your findings along with the graphs.
7. **Design Doc (15 points)**: You must write about how your program was designed, what tradeoffs you made, etc. Also describe possible improvements and extensions to your program (and sketch how they might be made). Separate from report.
8. Please structure your assignment root folder as follows:
    a. Code: for your source code and make files and deployment scripts. Your file structure in here is up to you but please make sure it is clean.
    b. Docs: for all written documents, report, readme, design doc etc.
    c. Out: for all output files from your peer nodes and Indexing server node, named indicating which specific evaluation.
    d. Misc: for other files not mentioned specifically.
9. Please put all of the above into **one** .zip file, and upload it to the **blackboard**. The name of .zip should follow this **format**: “Section_LastName_FirstName_PA4.zip”

Submission checklist:

- Your source codes
- Makefile or equivalent specified above
- Deployment scripts
- Readme
- Your performance evaluation report
- The output files of your peer nodes and Indexing server node as specified above
- A Design Document
- **NOTE**: You do not need to add all your test files which you transferred during your run, as it increases the file size and can lead to some issues when uploading to blackboard. You only need to upload the files mentioned above.