STDISCM

P3 - Networked Producer and Consumer

Jack Elinzano, John Joseph Giron, Ronn Patrick Ramos, and Francis Matthew Recomono

https://github.com/jjg-git/p3-network

GITHUB LINK

Key Implementation Steps

1. Implementing the gRPC protocol that handles video transferring.
    a. It has two messages: 1. VideoInfo, and 2. VideoData
        i. VideoInfo - contains the filename and the file size.
        ii. VideoData - contains the byte string from the raw data in the video file
2. The gRPC classes and methods are generated by protobuf-java from the protobuf files
3. The Producer now has to implement the gRPC services. On the Consumer side, the stubs must be implemented.
4. Consumer service connects to a Producer gRPC server to retrieve the information about the videos and the video data.

Queueing details

gRPC has a built-in streaming algorithm that lines up the requests from the client. It also uses threading to optimize the performance. However, gRPC's default behavior does not put limits on how many threads to use.

To simulate the waiting in the queue, the thread pool is implemented to be fixed by a number of threads.

Producer and Consumer Concepts Applied

- gRPC Streaming
- Protobuf format
- GUI

Synchronization Mechanisms

Multithreading was used for the implementation of the “getFiles()” method in Consumer.

```
public void getFiles(ConsumerConfig setting, Iterator<VideoInfo> videos) {  1 usage   Rikariooo +1
    // Use a fixed thread pool with, say, 4 threads (tweak as needed)
    ExecutorService executor = Executors.newFixedThreadPool( nThreads: 4);
    List<Future<?>> futures = new ArrayList<>();

    while (videos.hasNext()) {
        VideoInfo info = videos.next();

        // Submit each video download as a separate task
        Future<?> future = executor.submit(() -> {
            System.out.println("Reading " + info.getFilename() +
                    " (" + info.getFilesize() + ")...");

            String filename = setting.output() + info.getFilename();
            try (FileOutputStream os = new FileOutputStream(filename)) {
                Iterator<VideoData> streamedData = blockingStub.sendVideo(info);
                while (streamedData.hasNext()) {
                    VideoData data = streamedData.next();
                    os.write(data.getData().toByteArray());
                }
            } catch (Exception e) {
                System.err.println("Error downloading " + info.getFilename() + ": " + e.getMessage());
            }
        });
```

```
    for (Future<?> future : futures) {
        try {
            future.get();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    executor.shutdown();
}
```

End of slide