CSCE 313 PA1
Client/Server process communication

PA timing analysis Deliverables

Abhimanyu Singh
ECEN CSCE 313
TA: Di Xhao

Due February 12th, 2020

Introduction/Purpose

In this lab, we used client-server communication through FIFO pipelines via the system kernel. These 2 processes communicated in order to request files and transfer them.

Data requests:

We requested 1 data point initially, then 1000 using a terminal flag to set the amount of data point requests to be sent to the server, looping through the file to collect both ecg values and ofstreaming the data to a file when the client received it, via a char val, later a buffer during file transfer

File requests:

The file requests used a different msg flag that was interpreted by the server. Once the empty filemessage was declared, the filemessage was then added onto a buffer, then after that the filename itself as a cstring. When the server received this request with the filename, it sent back the size of the file. Now a While loop is in charge of making sure that the filesize is larger than the offset value, once the offset value reaches this all the data has been sent over. The actual data sent in the buffer was either the buffer capacity or the filesize - offset once there were less than bufcap bytes to be sent. To address this I made an if else statement that handles both scenarios and sends a filemessage to the server which has the initial address of the offset and the file length of either the buffer capacity or the amount of bytes left “the space”. This data was then written into a file with name filename under the received directory.

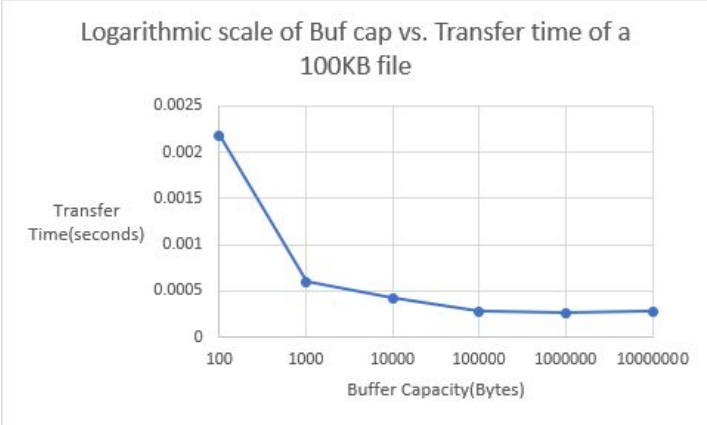
New Channel:

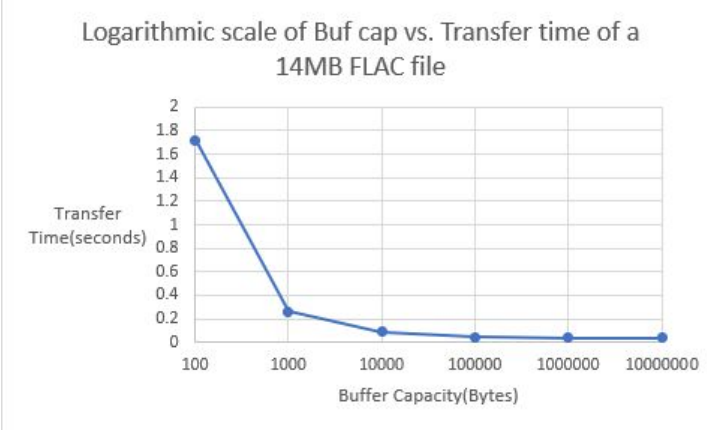
The new channel requests were very similar to the regular channel request, the only difference was that we had to test this new channel whilst the old channel was still running. Once the channel was opened, the tests were run to gather ecg data and compare.

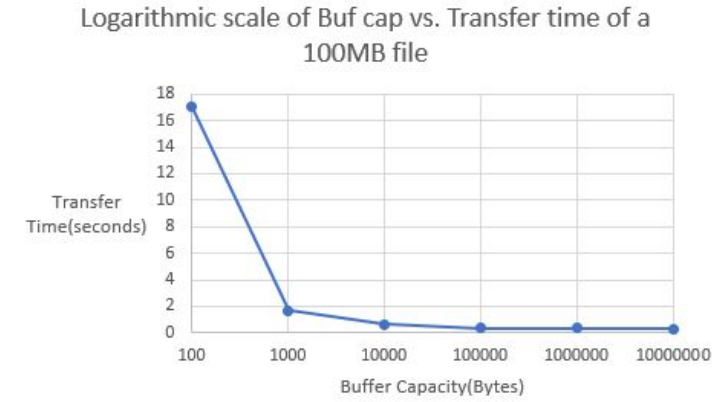
Fork() Parent and Child process:

The instantiation of a int value that was equal to the return of a fork() call, enabled me to branch the current process into a new identical copy while running that one and then returning back into the parent process. When int pid == 0, the child process was started and was given ./server name and the same arguments as the ./client

Graphs:







100 KB file transfer was quick compared to the 100MB file transfer; the relative speedup began to wear down in all 3 file transfers when the buffer capacity was set to 1MB. 14 MB music file also had a significant speedup when using larger buffer size, although past a certain point the throughput was negligible.

The main bottleneck was definitely the Buffer Capacity, but after a certain value, 1MB, the speedup was negligible, at this point the one thing restricting the speed of

the file request and transfer is the speed of my computer's SSD(disk). Upgrading this would lead to faster times but in the grand scheme of things not by much.

DEMO: https://youtu.be/BjYvuI8v-Bg