PA1: A Client Process Speaking to a Server Process

Brandon DeAlmeida
CSCE 313-508
Instructor: Tanzir Ahmed
TA: Feras Khemakhem

Design:

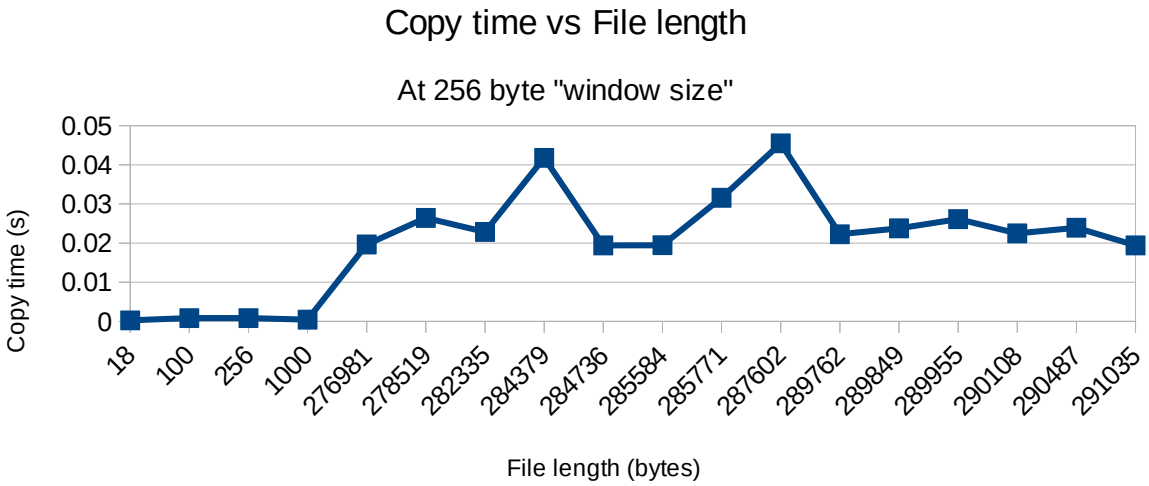
The client process has been designed to be controlled by whatever arguments are passed to the executable. A file copy is always favored such that if multiple arguments are provided, including **-f**, the rest are effectively ignored such that only a file copy from the server occurs. The arguments **-e**, **-p**, and **-t** allow the user to get a single data point from the server, while omitting -t yields the first thousand entries for person **-p** (if **-e** is given only that column will be passed) and stores the data within x1.csv within the current directory. Finally, running the client without any flags performs a simple test of creating new channels. Only one terminal is required to run both the client and server as the client starts the server itself utilizing **fork()** and **execvp()**. This is further expanded upon with the **-m** argument that sets the maximum buffer size, max_buffer, (defaults to **MAX_MESSAGE**) for both the client and server as the parent client passes it along to its child.

A file copy is performed by first determining the length of the file in question (assuming it exists) such that the number of *n* requests required to copy the file can be computed with $n = \lceil file\_length/max\_buffer \rceil$. Then, $n - 1$ requests for the maximum buffer size are performed and the last request is only the remaining bit of the file that's less than the max_buffer. When dealing with the file messages required for this process with the server a custom packet must be create that includes both the file message (the offset to start the copy and the amount to copy) and the file name itself: this allows the server to know what file is being requested.

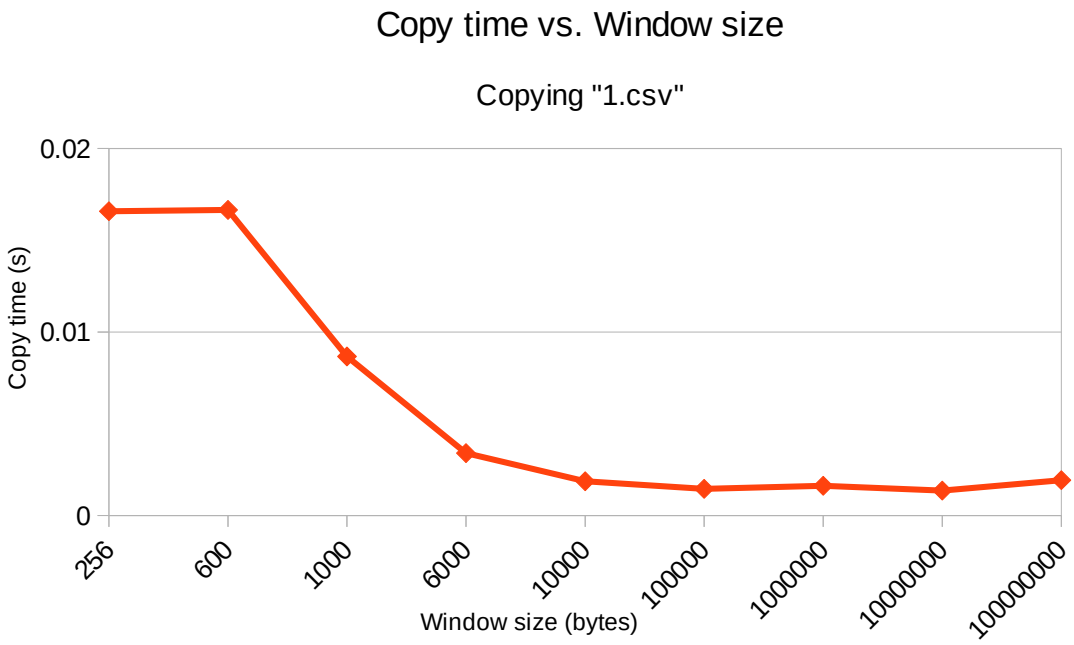
A data point request is a singular message passed along to the server to request the point specified by the user. This simple process is repeated for multiple points such that **many** requests are made (for example, getting both e=1 and e=2 values for the first thousand entries of a file results in two thousand requests), this makes for a much slower copy of data than the file copy.

Data:

While copying both e=1 and e=2 for just a thousand data points I saw a bit of change within the time it took to perform this operation: over ten run times I found an average time length of 5.222592s with very little variance.



It was observed that while larger file sizes do generally take longer to copy than smaller ones there are a few other factors that influence how long a file copy takes. This can be seen by the large fluctuations in the copy times found above where there are some spikes. I found a much bigger factor in the time to copy an entire file to be the max_buffer size used (called the window size when taking about file copies): a larger window meant that less exchanges happened between the server and client such that copy times were reduced exponentially with diminishing returns, shown below.


Copy time vs. Window size
Copying "1.csv"
0.02
0.01
0
Copy time (s)
256
600
1000
6000
10000
100000
1000000
10000000
Window size (bytes)