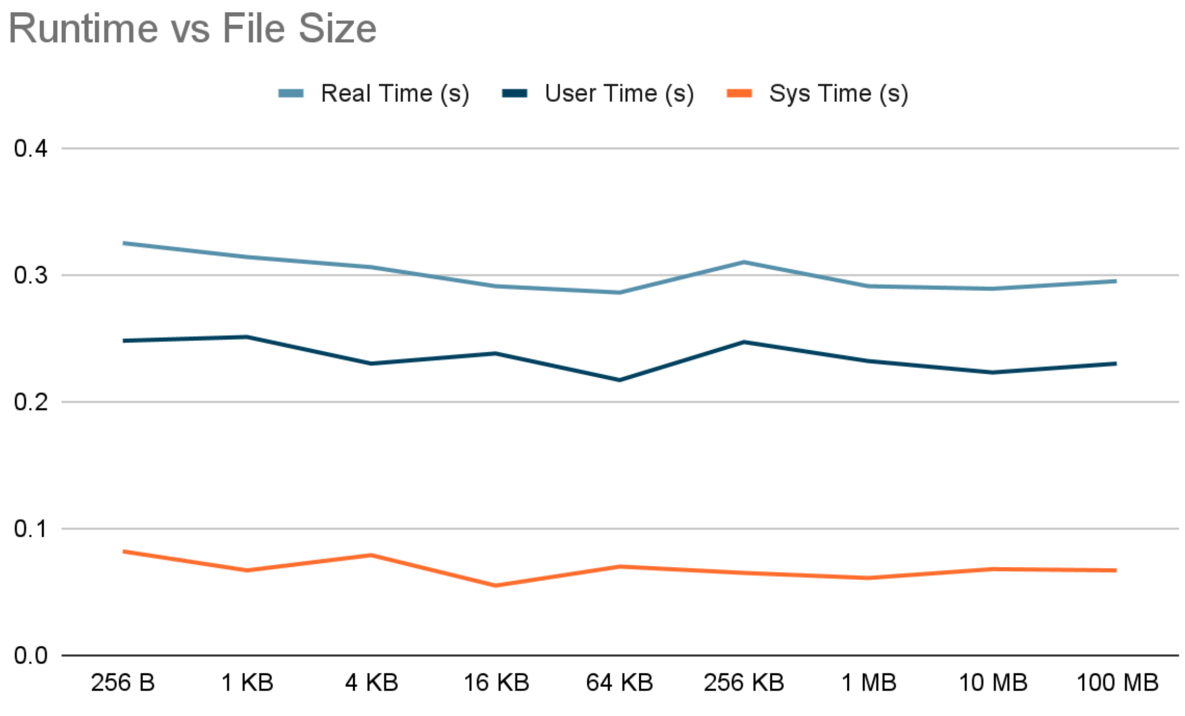
Execution time vs file size: I ran the program many times with files of different sizes and compared the input size to the output time. Here is the data, and a linear regression of it

File Size	Real Time (s)	User Time (s)	Sys Time (s)
256 B	0.325	0.248	0.082
1 KB	0.314	0.251	0.067
4 KB	0.306	0.230	0.079
16 KB	0.291	0.238	0.055
64 KB	0.286	0.217	0.070
256 KB	0.310	0.247	0.065
1 MB	0.291	0.232	0.061
10 MB	0.289	0.223	0.068
100 MB	0.295	0.230	0.067



It seems like there is a constant relationship between the runtime and the size of the files.

Bottleneck: The main bottleneck in this program's output (and this is a common bottleneck in for programs like this) is the IO for both files. Every time my client wants to get an array from the server, it requests the information, the cpu has to wait for the scheduler to make the server run, and then the server just sits there waiting for the scheduler to make the client take in the data, and then use it. Most of the CPU's time is spent waiting. Additionally, because I wanted to be able to create chunks at runtime, and I have to new and delete them constantly, that also produces a lot of overhead (although that is just associated with the IO anyway. In optimized code, I would create a bunch of channels at once (or larger channels), be able to request/send all the data at once in parallel, or I could reuse buffers.