Tomas Woldemichael
June 6th, 2021
Program 4

Sockets Program 4

The purpose of this program was to build a server that could interact with client using read and writes to some sort of buffer through a socket. For this program, I gathered metrics in regards to speed of the various types of write test and different number of buffers and buff sizes. I plan on talking more about the metrics and there meaning later in this report. Overall, this program allows the sever to connect to a port and the client can connect to socket using a the server name and port to send bits of data back and forth.

I first started this program working on the server portion of the program. A lot of the code was similar to the already provided template for socket development. The serve.cpp first checks to see that the user has put in the correct amount of arguments (executable file and port number). Once the program verifies that he amount of inputs is correct it begins to build the socket in which various clients could connect to. Once it has built the receiving socking it will then open the socket and bind to the port that was passed in by the user. Clients will be required to input the same port if they would like to communicate with this particular server. Once the server is configured in the appropriate manner it wait and listen at that port till a particular clients connects to the port.

As the server waits and listens at a certain port, a client must come and connect for it to continue the magic. In the client.cpp, I first begin by checking if all the correct arguments are passed in by the user. Once all the correct arguments are passed in such as [exec][serverName][port][repetitions][numBuffs][buffSize][type], the client will run and configure its self to the specified configuration. Once the client configures it self, it will try to connect to the server that is listen at that port. If the client successfully connects, the client will continue with the next part of its program. The next part of the program for the client is to begin writing to a buffer through the socket.

There are three different types of writing test, Type 1 multiple writes, type 2 writev , type 3 single write. All of the following types of writes behave differently with the amount of data that is being transferred over the socket at a different rate of speed. Each of the types of test are performed many time over many repetitions in order to gather appropriate metrics. As the writes are being completed the server is working parallel to read from the socket buffer.

The server.cpp works in Parallel after the client has connected to the port. The server creates helper thread to read data from a buffer as the client is transferring data over the socket. The server helper thread will then record the amount of reads that was required based on the type of writes that was written over the socket. Once the amount of repetitions is complete the helper thread will send the amount of reads back to the client in order for the client to display the metrics of the performed test. The helper thread will be exited and the main thread will go back to listening at the port till a new client connects and the process will be performed once again till the number of connections is complete.

Tomas Woldemichael
June 6th, 2021
Program 4

Performance Evaluation:

numBuff/buffSize/Type	Sending Data avg	Round Trip Avg	Gbps Avg	NReads Avg
15/100/1	295529 usec	311452 usec	.8125528 gbps	39305.8
15/100/2	45190.1 usec	45464.4 usec	5.29378 gbps	20000
15/100/3	22448.9 usec	22504.8 usec	10.691 gbps	20000
30/50/1	730167 usec	730272 usec	0.328692 gbps	104136
30/50/2	63438.3 usec	63460.7 usec	3.7832 gbps	20000
30/50/3	22760.4 usec	22853.2 usec	10.5446 gbps	20000
60/25/1	1.31483e+06 usec	1.31492e+06 usec	0.182533 gbps	151339.4
60/25/2	104600 usec	144030 usec	2.29445 gbps	20000
60/25/3	24465.7 usec	24586.4 usec	9.80965 gbps	20000
100/15/1	2.14657e+06 usec	2.1466e+06 usec	0.111806 gbps	235103
100/15/2	158869 usec	158998 usec	1.51068 gbps	20000
100/15/3	23186.1 usec	23279.6 usec	10.351 gbps	20000

Discussion:

Comparing your actual throughputs to the underlying bandwidth
Depending on the type of test the through put would change significantly. As the amount of time it took to send and receive data across the socket increased the through put would decrease. This makes complete sense because as time is the denomintar o therefore as the amount of bits stays consistent but takes longer to send the gbps will decrease.

Comparisons of the performance of multi-writes, writev, and single-write performance
Overall, the test that always seemed to have the lowest amount of gbps was type 1,which was multiple writes to the buffer. Then type 2 came in second place with the amount of gbps that was able to send across the socket and type 3 was able to send the most amount of data in the least amount of time. This particularly has to do with the write being done at once or multiple times, and since going to the disk to write data is extremely expensive it is reflecting In time.

Comparison of the different buffer size / number buffers combinations.
The way the data was also passed in has a significant effect on the amount of bits sent over a period of time. The reason being that if there is more buffers than the memory has to load and unload many different buffers but if the buffer is larger with less amount of individual buffers then its doesn't have to spend time load and can write and read directly you. From the data you can see as the amount of buffers increase so does the time and that directly correlates to a lower amount of gbps as time increase.