**CSS432 Program 1: Socket**

Eunmin Lee

This program evaluates round trip times and the number of reads when the server and client is networking using socket (TCP).

**Documentation**

This is the general flow of the program

|  |  |
| --- | --- |
| Client | Server |
| 1) Get server name, server port, number of iteration, number of buffers, buffer size and test type from the user command line. | 1) Get port number from the user command line, and bind the socket to the address. |
| 2) Connect to server | 2) Start serving pthread with the connected client |
| 3) Send the number of iteration within the buffer by write(...) | 3) Read iteration time by read(...)   * The return without reading the entire data if the network is slow. Multiple read() will be done. |
| 4) Send the buffer by write(...) for the time of iteration | 4) Read the buffer by read(...) for the time of iteration, and add up the number of read() calls. |
| 5) Read the number of read(...) calls. | 5) Send the number of read(...) calls within buffer as acknowledgement. |

This is elements required for the program

|  |  |
| --- | --- |
| Client | Server |
| /\*main function\*/  **int main(int argc, char\* argv[]):**  //store servername to connect  char\* serverName;  //store serverPort to connect  char\* serverPort;  //store the number of iteration  int repetition;  //store the number of buffer  //use for databuf[nbufs][bufsize];  int nbufs;  //store the size of buffer  //use for databuf[nbufs][bufsize];  int bufsize;    //store data type  int type;  //date buffer  char databuf[nbufs][bufsize];  //Store IP, TCP, flag, other protocol  struct addrinfo hints;  //Store boolean whether conversion of //domain names, hostnames and IP //addresses to binary success or not  int rc;  //Store conversion of domain names, //hostnames, and IP addresses to binary  struct addrinfo\* result, \* rp;  //store socket  int clientSD;  //store starting time  chrono::steady\_clock::time\_point t1;  //store finishing time  chrono::steady\_clock::time\_point t2;  //store elasped time (t2-t1)  int etime;  //store the number of read(...) calls passed from //server  int count; | /\* Global integer store buffer size\*/  **const int BUFFSIZE;**  /\*Global integer store number of client to put in the connection request queue\*/  **const int NUM\_CONNECTIONS;**  /\* Service thread function\*/  **void\* servicingThread(void\* args):**  //store referrence to the socket  int& newSD;  //date buffer  databuf[BUFFSIZE];  //store number of iteration get from the client  int iteration  //store count of read(...) call  int count;  /\* main function\*/  **int main(int argc, char \*argv[]):**  //store port to bind socket  int port  //store sever socket address  sockaddr\_in acceptSocketAddress;  //store boolean whether binding socket to //address success or not  int rc  //store server socket  int serverSD;  //store client socket address  sockaddr\_in newSockAddr;  //store socket length  socklen\_t newSockAddrSize  //store attribute option for pthread  pthread\_attr\_t attr;  //store group of pthread  pthread\_t threads; |

**Performance Evaluation**

nodes : csslab4 (client) and csslab10 (server)

Repition set: 20000

Averaged from 5 results:

|  |  |  |  |
| --- | --- | --- | --- |
| nbufs \* bufsize | Type1: multi-writes | Type2: writev | Type3: single-writes |
| 15 \* 100 | 853Mbps/22918reads | 4417 Mbps/20154reads | 4823 Mbps/20151reads |
| 30 \* 50 | 430Mbps/24757reads | 3159 Mbps/20832reads | 4729 Mbps/20141reads |
| 60 \* 25 | 240Mbps/28966reads | 2144Mbps/22563reads | 4972Mbps/20116reads |
| 100 \* 15 | 150Mbps/34037reads | 1438Mbps/23378reads | 4952Mbps/20155reads |

**Discussion**

(1) comparing your actual throughputs to the underlying bandwidth

*The bandwidth of the Linux lab is 10Gbps, while the actual throughputs result shows 150Mbps-4982Mbps. The actual throughputs can be up to ~66 times slower.*

(2) comparisons of the performance of multi-writes, writev, and single-write performance

*single -write showed the best performance (highest throughput and lowest number of reads) while multi-writes showed the worst performance. Multi-writes showed 5937 more average read calls comparing to writev, and 7530 more average read calls comparing to single-writes. Type 1 throughput is 5- 9 times slower than type 2 and 5-33 times slower than single-writes.*

(3) comparison of the different buffer size/number buffers combinations.

*As the number of buffers gets bigger and buffer size gets smaller, the performance of type 1 and type2 test got poorer. The performance of multi-writes drops 6 times when nbufs: bufsize = 15:100 change to nbufs: bufsize = 100:15. The performance of writev drops 3 times when nbufs: bufsize = 15:100 change to nbufs: bufsize = 100:15. However, type3 test result didn’t show lots of difference when nbufs and bufsize changes.*