Comparing various Client Designs

T Dinesh Ram Kumar

2014A3A70302P

MANDAR KHAPEKAR

2014A3TS0271P

Comparing Various Client Designs (Using Single Connection)

Echo Server

Simple TCP Echo server is designed which creates a new process for each client request. It uses blocking IO operations. It reads input string and capitalizes it and writes back the string. It repeats this till no more data is available to read. Then it closes the connection. Now various client designs have been designed. For measuring performance of each client design, each client sends 10GB of request to client in request sizes of 8kB, average response time, total time and overall throughput is computed. Before that we will explain design features of the clients.

1. Blocking IO with select

Client connects to the echo server. Select is used to check if socket is readable or writable or both. Then data is sent or received from server respectively using blocking system calls. Shutdown is used to indicate no more data to server.

2. Non-Blocking IO with select

Client connects to the echo server. Select is used to check if socket is readable or writable or both. Then data is sent or received from server respectively using non-blocking system calls using MSG_DONTWAIT flag. Buffers and pointers are used to maintain states across calls. Once an entire request is sent, new request is sent and similarly for response. Shutdown is used to indicate no more data to server.

3. Two Processes

Client connects to the echo server. Then forks another process. One process is used to send request to server, while other is used to process responses from server. Shared Memory is used for IPC between two processes. All IO operations use blocking calls. Shutdown is used to indicate no more data to server.

4. Two Threads

Client connects to the echo server. Then creates another thread. One thread is used to send requests to server, while other reads and process the server responses. All IO operations use blocking system calls.

Response time, Throughput and total time is shown below (Taken as an average of 5 iterations) .

Client Design	Response Time (s)	Throughput (bytes/s)	Total Time (s)
Blocking IO	0.000042	196099207.64	51.010

Non Blocking IO	0.000042	196272195.53	50.953
Two Processes	0.000046	177023049.46	56.450
Two Threads	0.000042	192083192.64	52.071

File Clients

File Server

File server handles two types of queries.
1. Information Query
Client request information about a file to server. And server responds with size of file to client if it exists else indicates an error.
2. Fragment Fragment Query
Client request file fragment from server identified by offset (in file) and size of fragment. Server sends the file fragment as a response else it closes connection.
File server forks for each new connection. Reads a request and responds to it and closes the connection. (Only single request per connection like non-persistant HTTP) It uses sendfile system call to send the file fragment.

All file clients first query the server for information about desired file to obtain its size. Once its size is known, based on number of connections desired, it decides on fragment sizes and offsets to request from the server. It then outputs the fragments to the desired output file (single file). It uses lseek64 to position to offset desired. File clients are different in how they make connections and IO operations.

5. Non-Blocking I/O with non-Blocking Connect

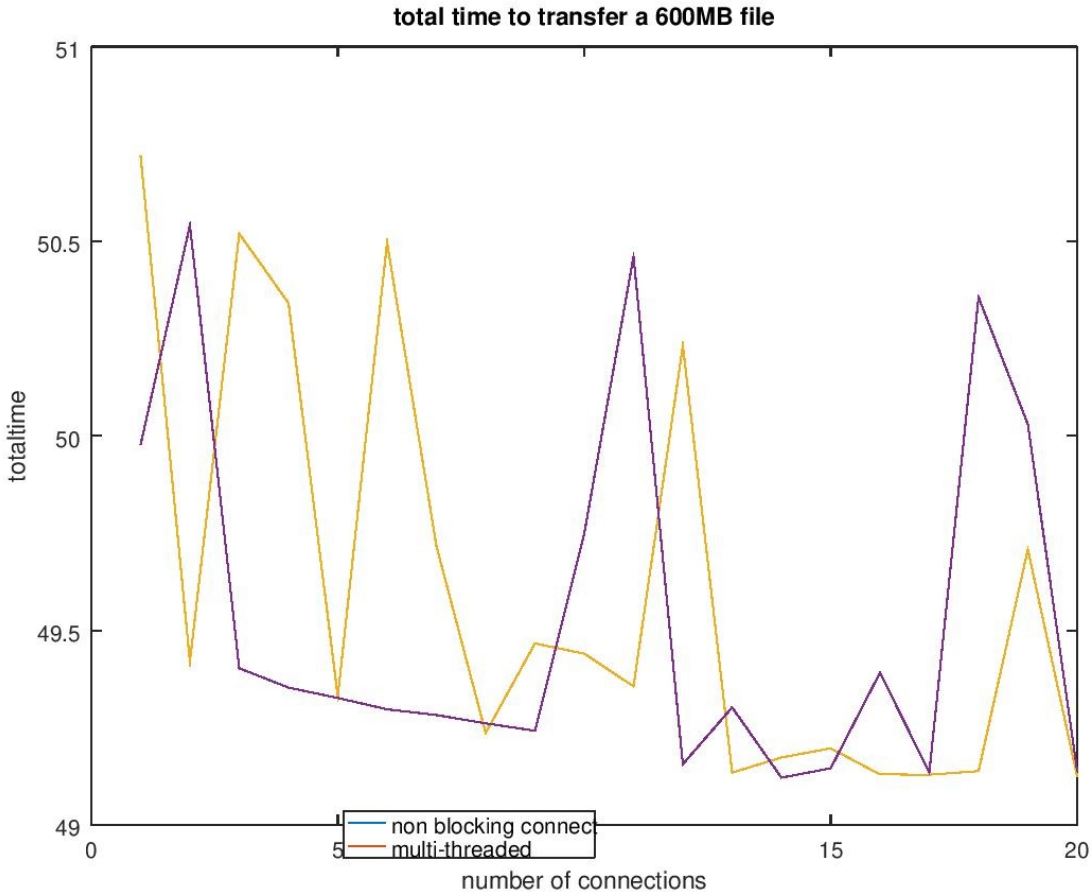
This client uses a single thread of execution. It uses poll system call to watch over many file descriptors. Connects are non-blocking. It is checked for writability. If no error in connection, request for file fragment is sent to server using block call. Then socket is watched for readability. Output file is opened number of connection times and offset accordingly set. Files are checked for writability. Buffers and pointers are used to maintain states in the non-blocking operation separately for each connection. To maximize the utilization of buffer (to effectively use both the ends), readv and writev system calls are used.

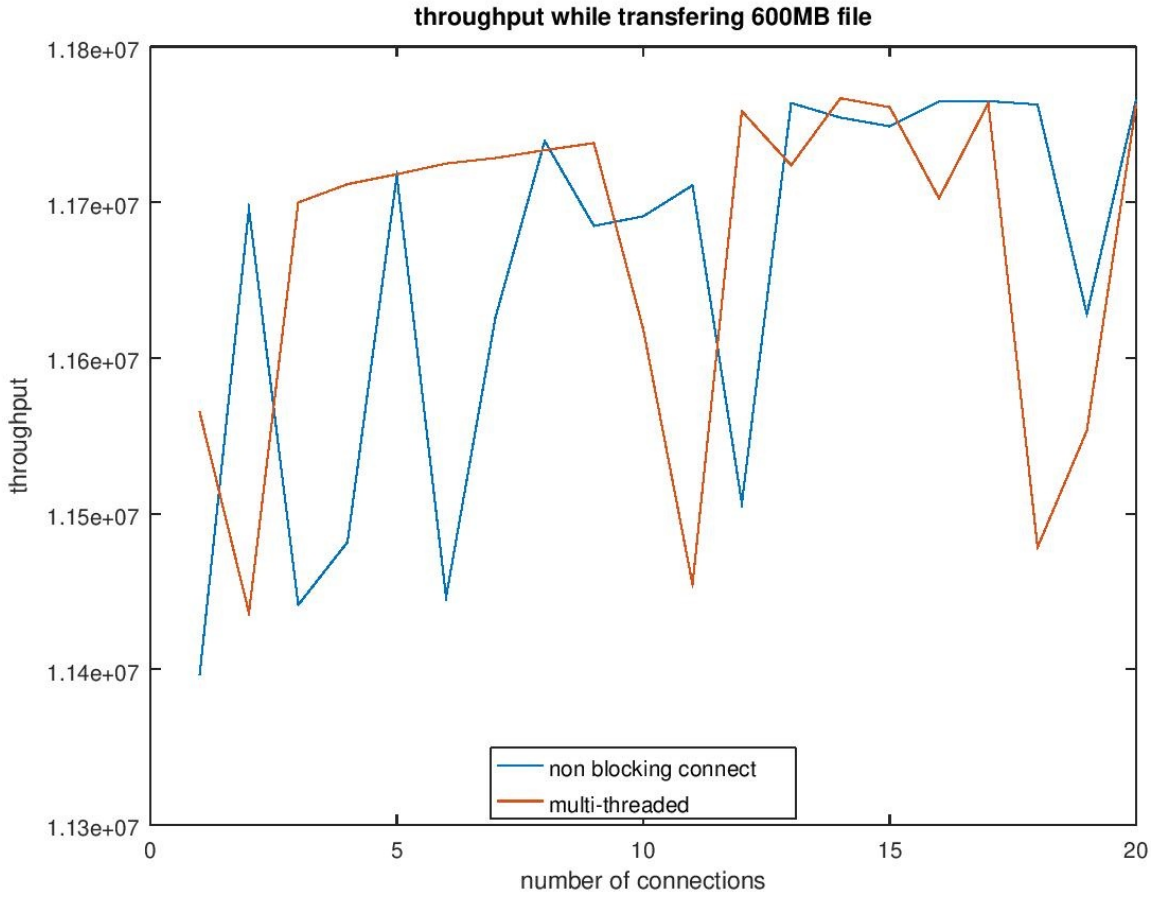
6. Blocking I/O with threads

For each connection, client creates a new thread. Now threads open the output file position the offset and requests for file fragments and repeatedly reads the response and outputs to the file.

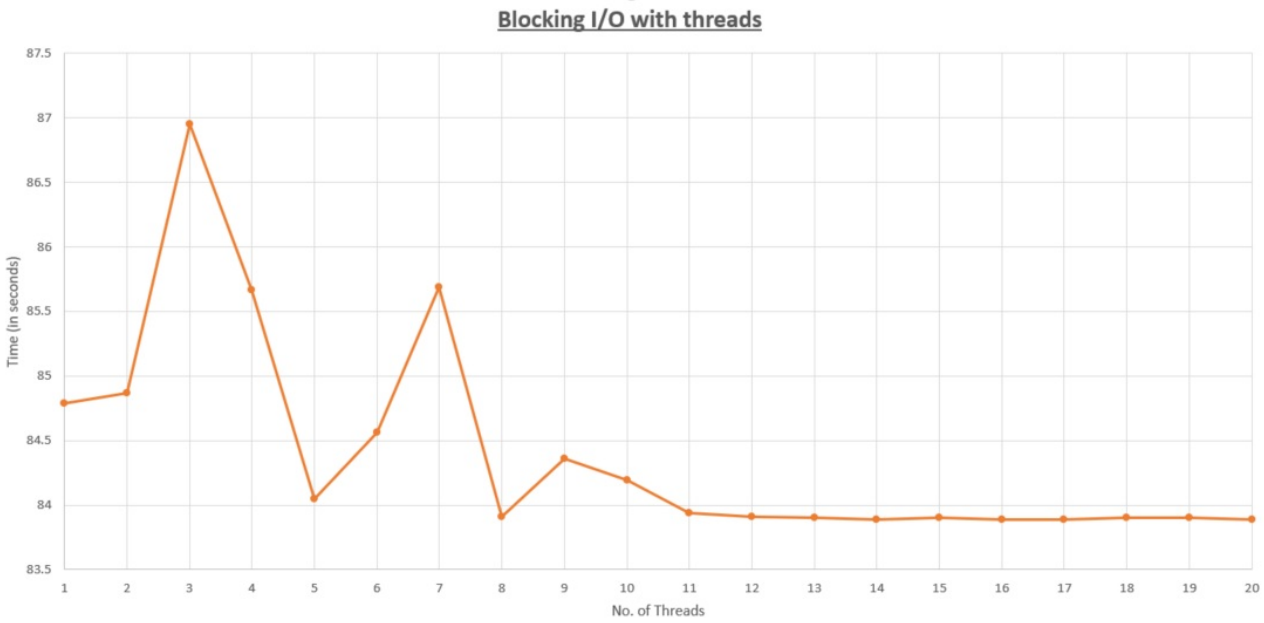
Plots are shown for number of connections varying from 1 to 20.

For a 600MB file

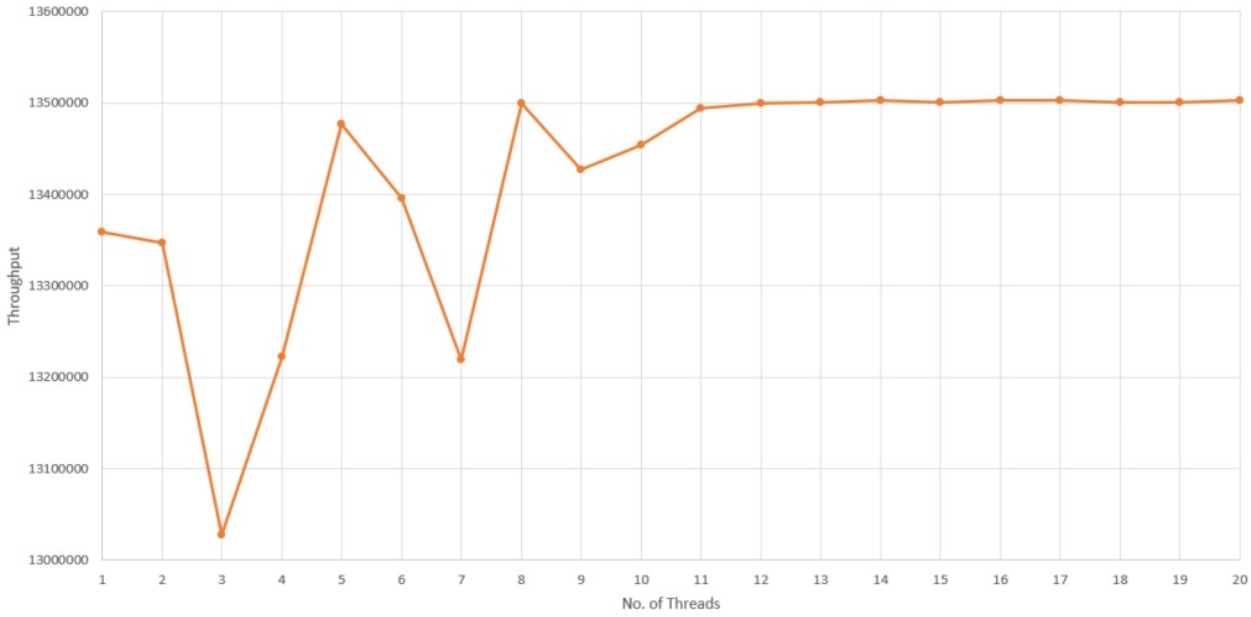

total time to transfer a 600MB file
51
50.5
50
49.5
49
totaltime
0
5
15
20
number of connections
non blocking connect
multi-threaded



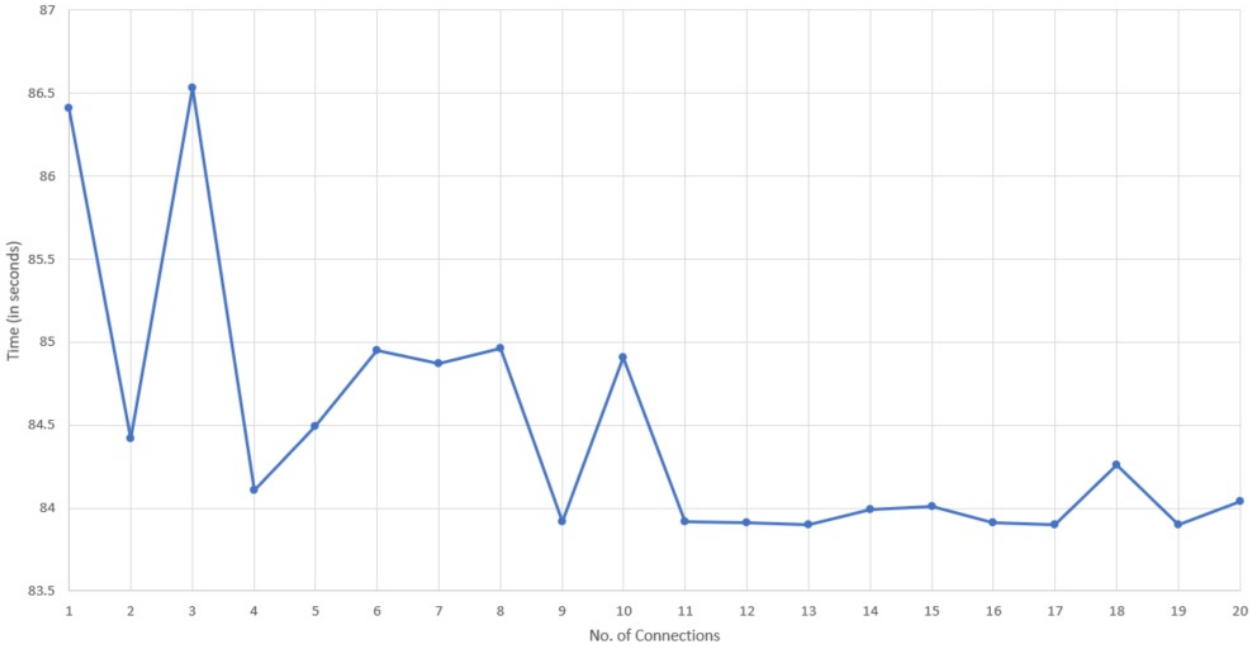
For a 1GB file

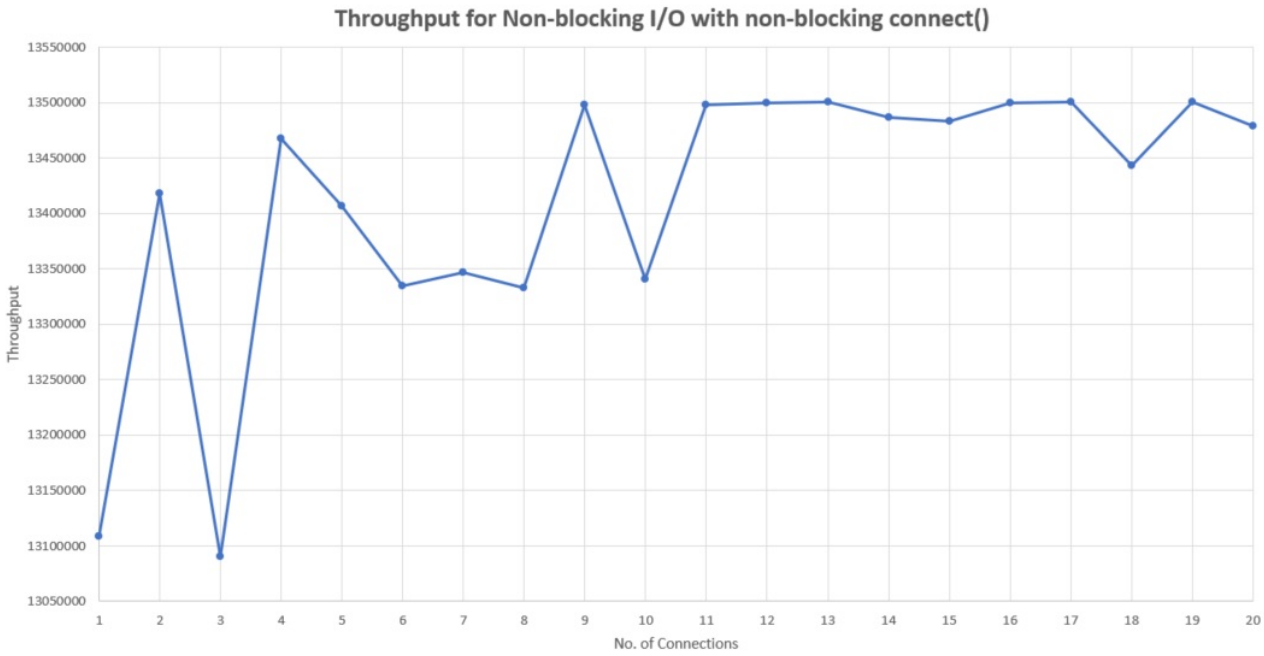


Throughput for Blocking I/O with threads



Non-blocking I/O with non-blocking connect()




Throughput for Non-blocking I/O with non-blocking connect()
Throughput
13550000
13500000
13450000
13400000
13350000
13300000
13250000
13200000
13150000
13100000
13050000
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
No. of Connections