|  |  |  |  |
| --- | --- | --- | --- |
|  | Nbufs 15, bufsize 100 | Nbufs 30, bufsize 50 | Nbufs 60, bufsize 25 |
| Type 1 | 8844.4 usec | 9878.2 usec | 13821.8 usec |
| Type 2 | 9763.6 usec | 8696.2 usec | 8318.2 usec |
| Type 3 | 8626.8 usec | 8731.4 usec | 8716 usec |

Each instance ran 5 times and got the average.

Multiple writes is almost significantly higher than the writev and single write. In writev, the vector iovec and the assigning of the iov\_base and iov\_len gives writev more time complexity making it reach higher than multi writes in 15 and 100 in size. This makes sense because of the need to access the data buf. As expected, single write is almost always lower than multi writes and writev since it is only needed to write one time.

On a slower network, as much as we’d see similar results, the change would be significantly different since the network speed is slowed down too much that the values will be later than the previous ones every time, for single write, this isn’t a huge issue since it is only one write, but for multi writes, and writev, the speed affects the values significantly by delaying each time a read/write is executed. We also have to keep in mind the creating the thread process which takes an ample amount of time as well.

In terms of using threads, using threads actually saves time since you are not overloading the server but using another way to speed up the process. Instead of waiting for a command to finish, you can start an earlier one that is not dependent on the previous one which saves time for the program. Dividing things up in methods and threads in general, saves time for the program making the process faster and more efficient.