Andrew Allen - 124008219
PA7 - CSCE 313 - Sec 507

README: Please see below to see specific custom functionality

The client and dataserver executables are called as follows upon calling make within the directory. **-i is useless as we were only called to implement the network request channel.**

./client -n <#reqs> -w <# workers> -b <bb size> -i <f|q|m|n> -h <host name> -p <port no> -z <if 1 then don't fork and call ./dataserver> -k <if 0 then don't kill server instance>

./dataserver -p <port no>.

If calling only the ./client on local host, the program assumes it should spin the server up for you meaning that it will call ./dataserver through execl unless you specify the -**z flag with 1** (I know you asked for it separate, but this makes it easier if running on localhost). Also, if running on an address other than localhost, the client assumes that the dataserver is already running and runs as expected. The client also assumes you want to close the server so it sends a command upon finishing that kills it unless you specify not to with **-k 0.**

Make instructions:

Simply navigate to the working directory and run:

make clean
make

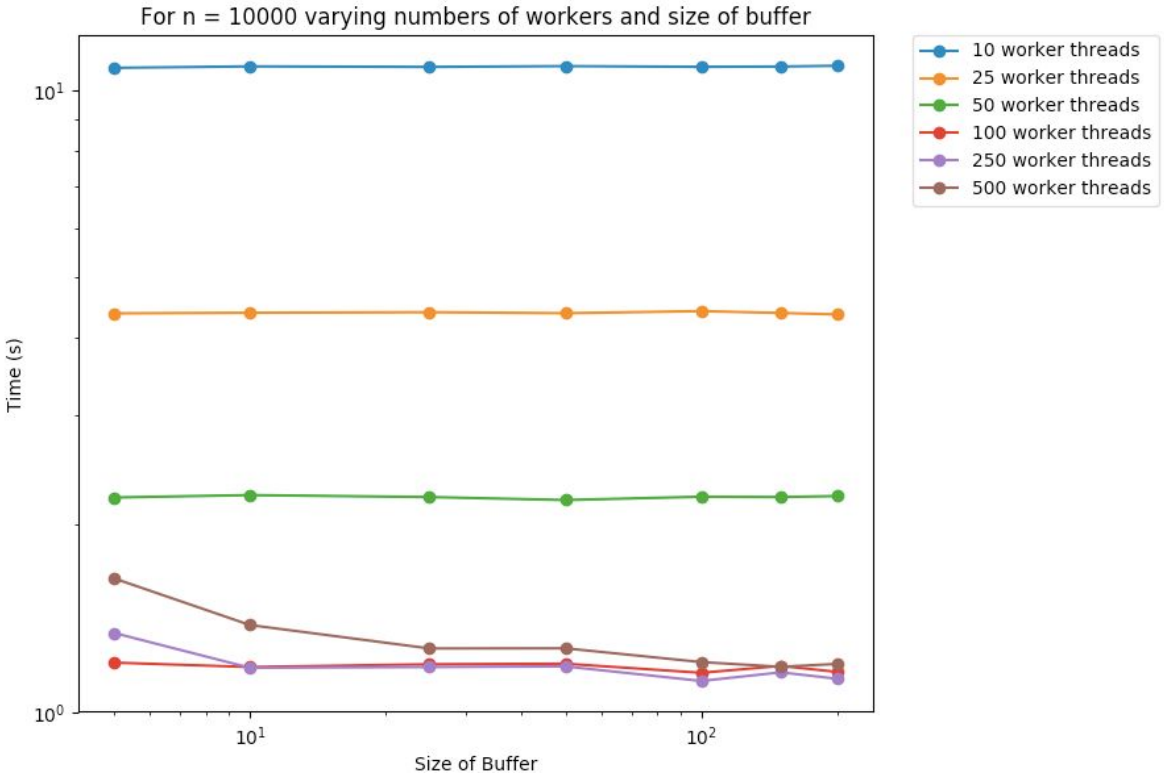
Then run the above commands as you wish

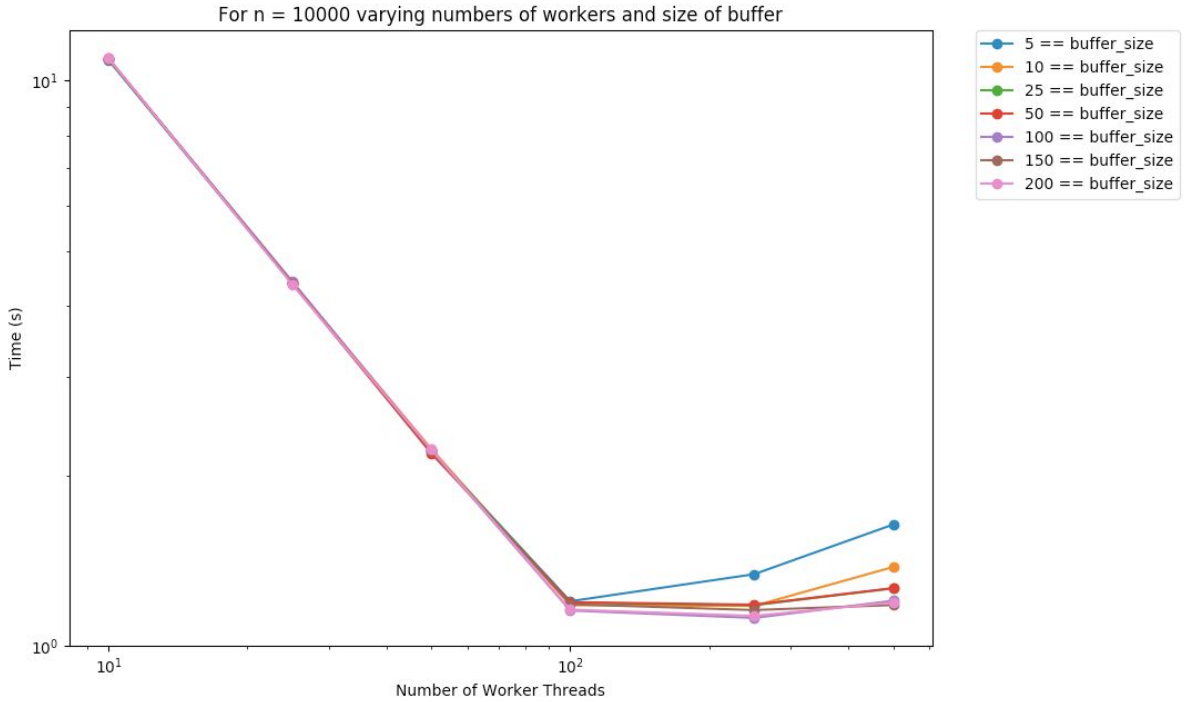
1. How does the performance compare to your implementation in the previous PAs?

The difference was minimal from PA5 or an of the other IPC methods used when running on localhost. I quite preferred this approach to some of the IPC approaches attempted previously as it seems less error prone and it has the ability to communicate with nodes outsite of the local network which is pretty cool.

2. What was so different about implementing this request channel?

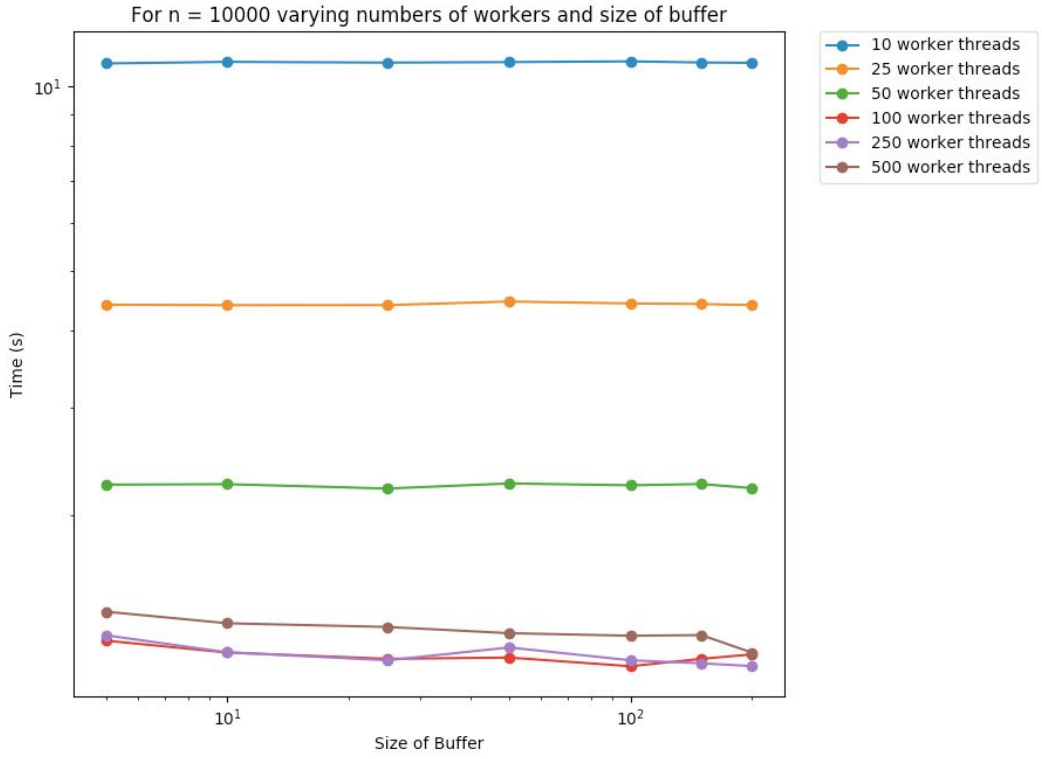
The huge difference between this and the other methods used is that a separate thread is made for each client request channel to communicate with. This means that for each request channel on the client side there is a thread listening to it on the server side. This makes the code very thread safe as all functionality is separate, but one does have to deal with the overhead of creating threads. The other advantages relate the fact that it is a network interface and that you don't have to deal with creating unique files and names to keep communication separate, the threads take care of that.

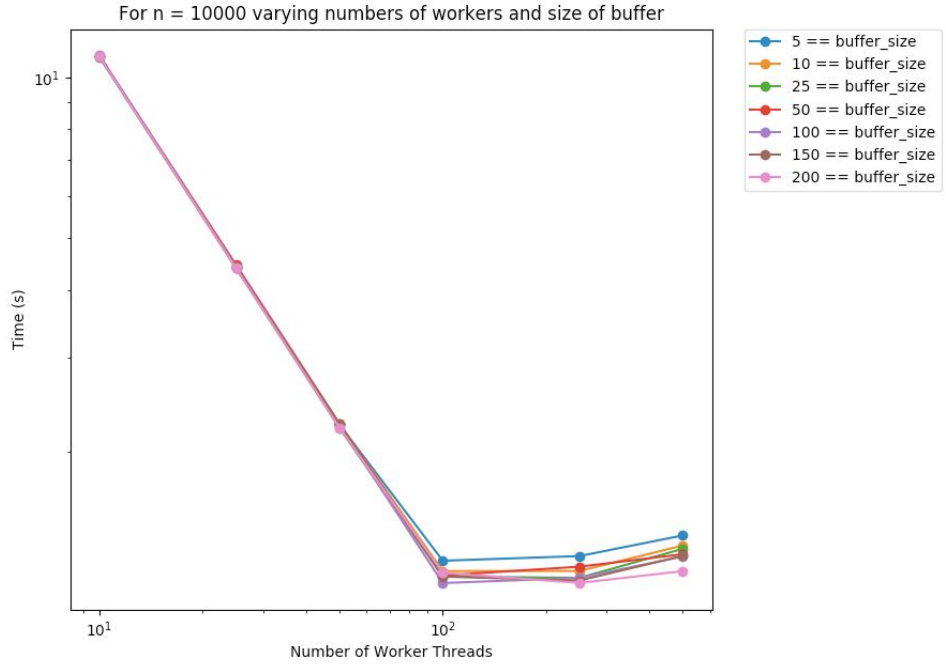




^1st run results of the test script on an Ubuntu server that was connected to localhost

-**Side note**: the connection seemed a lot slower (>10x) when I was connecting to a non-local ip address so I tested this many times using local host





^2nd run results of the test script, things were pretty consistent over time and just like the previous PAs, buffer size doesn't matter much at all and the number of worker threads is the only significant variable