**MP4 – CSCE 313**

Ryan Walters

Garrett Haynes

**Introduction:**

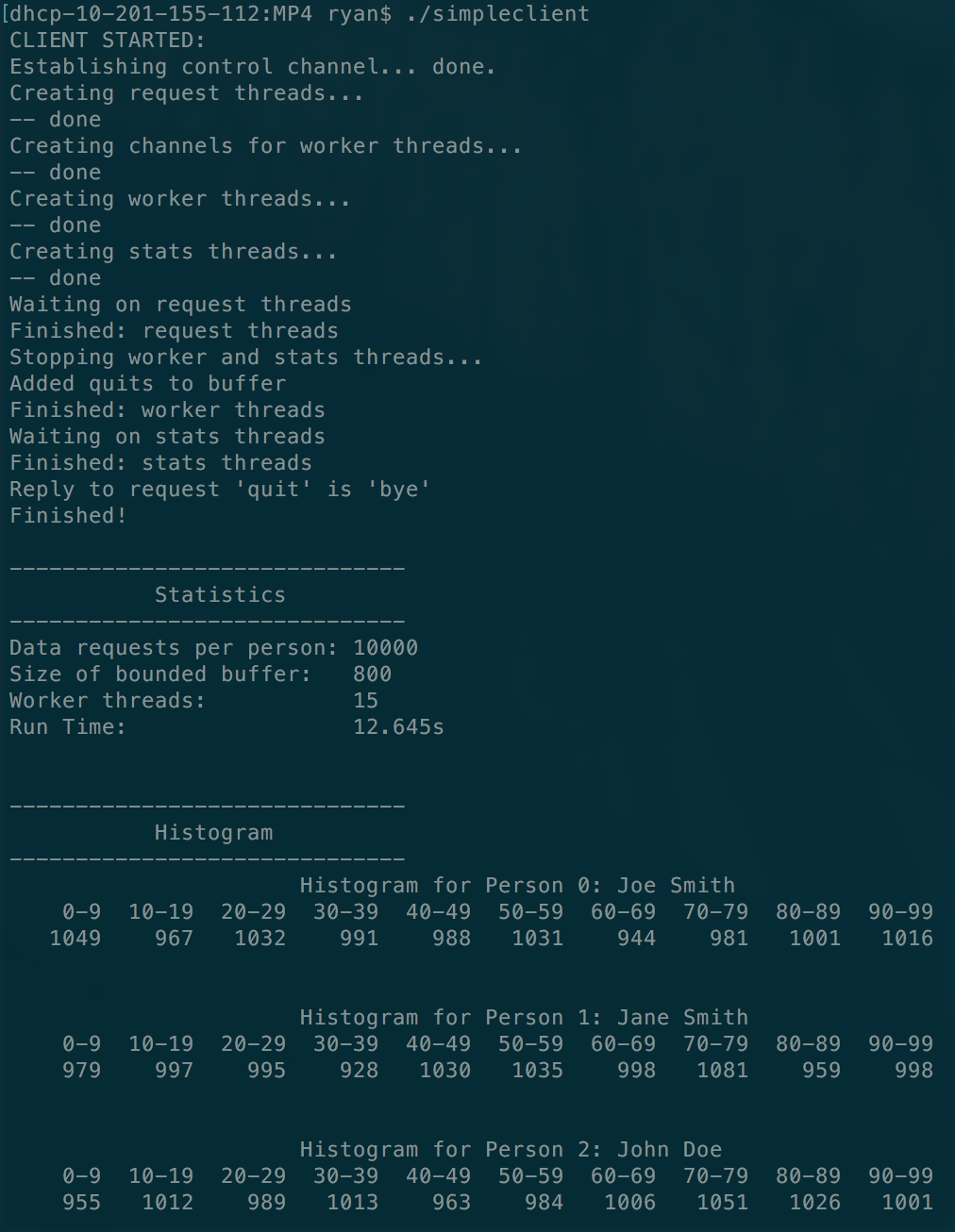
In this machine problem we edited out data server to make it seem more realistic, by having the data server perform some operation before a reply is sent. In this case return a number between 0 and 99. Also we edited the server to handle multiple requests at a time using a buffer of request threads. Also we improved our previous client to load the server with multiple concurrent requests. Also the client created and maintains a histogram of the returned values from the server.

**Procedure:**

First we edited our previous client to have a multiple number of request threads (i.e. one for each person). Then we have these threads generate requests and store them into a bounded buffer. We then spawn off worker threads which access the data in the bounded buffer and make the requests. A semaphore was implemented to protect the bounded buffer shared resource which multiple threads were accessing it. The responses received by the worker threads were stored in a buffer for each request thread(person). Statistics threads read the responses for each request thread and incremented a counter based on the number in the response. We then grouped these numbers into groups of 10 and displayed the results in the histogram.

**Result:**

The result of running the program can be seen in the following screenshot:



**Conclusion:**

**Does increasing the number of worker threads improve the performance and if so by how much?**

Using 10000 requests and a buffer size of 300 we got the following results for runtime at different amounts of worker threads:

|  |  |
| --- | --- |
| Number of Worker Threads | Runtime (s) |
| 15 | 12.6 |
| 16 | 12.0 |
| 17 | 11.4 |
| 18 | 11.0 |
| 19 | 10.7 |
| 20 | 10.5 |
| 30 | 10.6 |
| 60 | 10.2 |

As you can see, to a certain extent, increasing the number of worker threads does improve runtime, however this effect seems to bottle off at around 20 worker threads for our amount of requests and buffer size. Adding 5 worker threads to the default 15 improves the runtime by 2.1s.

**Is there a point at which increasing the number of worker threads does not further improve performance?**

As explained above, after roughly 20 worker threads, increasing the amount of worker threads does not give as much of a desired runtime improvement.