/\* client

-n <number of data requests per patient>

-b <size of bounded buffer between request and worker threads>

-w <number of worker threads>

\*/

-n 1000 for all test

Case 1: -b 50

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #worker | 1 | 2 | 3 | 4 | 5 |
| Time(s) | 12.790652 | 7.675637 | 6.213753 | 5.207001 | 5.279857 |

Chart, line chart

Description automatically generated

Case 2: -b 100

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #worker | 1 | 2 | 3 | 4 | 5 |
| Time(s) | 12.940675 | 8.558816 | 6.853832 | 6.218420 | 5.294778 |

Chart, line chart

Description automatically generated

Case 3: -b 25

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #worker | 1 | 2 | 3 | 4 | 5 |
| Time(s) | 13.032214 | 6.494931 | 6.385509 | 5.802769 | 5.028879 |

Chart, line chart

Description automatically generated

Conclusion:

Although increasing the number of the worker threads improves speed, there comes a point where there is no further improvement for an additional worker thread, and all cases narrow down to a small range of time spend as the number of worker thread increases.

Although the time spend is roughly the same for 1 worker and more than 5 workers, the increment of number of workers has greater influence for smaller size of the PCBuffer, which means smaller size of the PCBuffer reaches the “limit speed” faster than others.

For application, when we have limited number of threads to share, it is better the share resources to the buffer with smaller size.