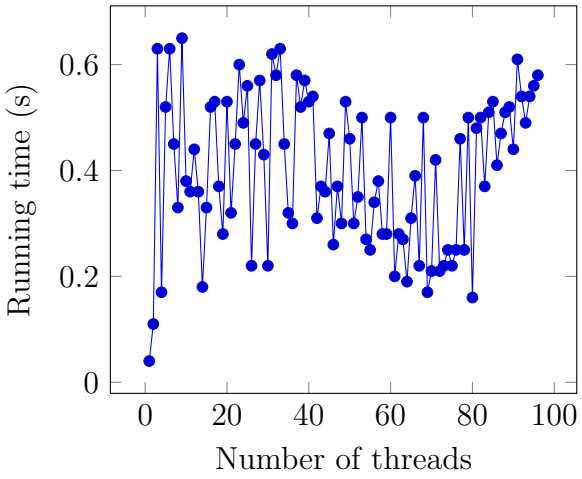
Systems Programming (2024 Fall)
Programming Assignment 4

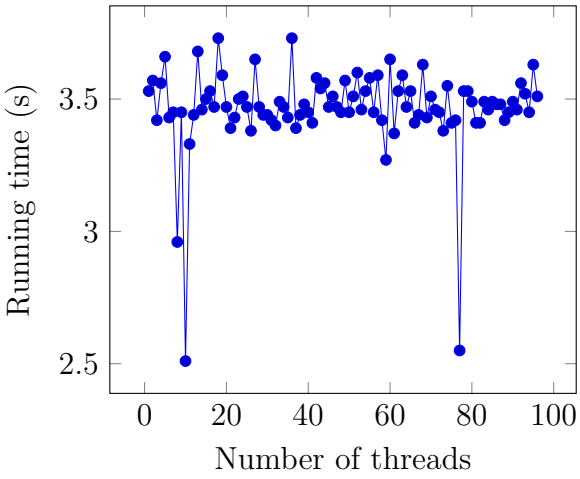
B12902110 呂承諺

1. The following are 3 cases with different n's and t's. For every request, `num_works` is set to be equal to n, that is, every row is processed by a thread.

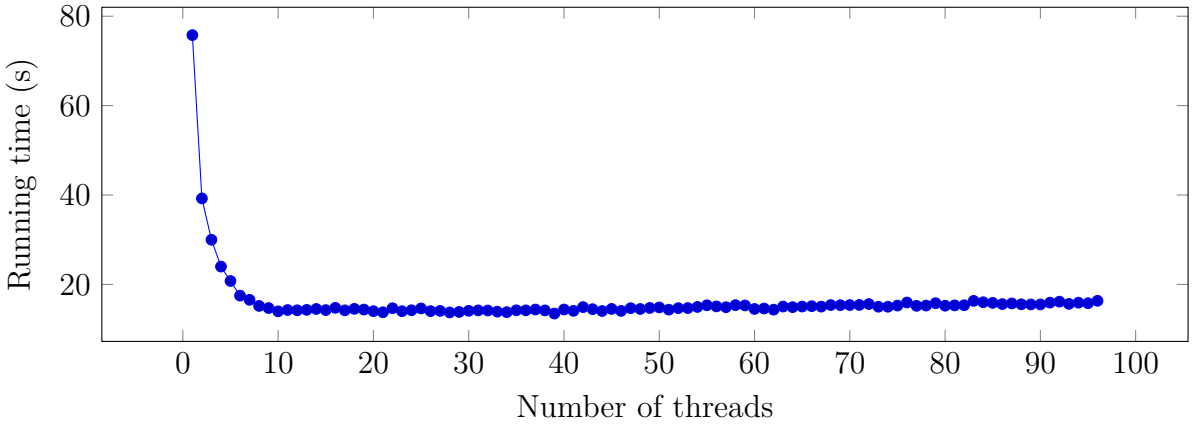
$n = 2, t = 10000$



$n = 100, t = 1000$



$n = 2000, t = 10$



The first two cases doesn't show a significant trend. We suspect the frontend thread is the bottleneck. There are too many requests for he frontend thread to handle, but there is only one frontend thread.

From the last case, we can see that running time decreases significantly when the number of threads increases from 1 to around 10, but remains roughly the same when the number of threads ranges from 10 to 100. The speedup disappears after 10 threads because there is bottleneck introduced by context switching.

2. Synchronization is handled by checking the following three conditions:

   - The frontend queue is empty.
   - The worker queue is empty.
   - There are no threads currently performing a job.

   If all three conditions are fulfilled, then all threads are synchronized.

   The last condition is maintained by variable `running_count`, representing the number of threads that has popped a request or work from the queue but is still processing it. This variable is protected by a mutex (`running_mutex`).

   When a thread has finished its job, it signals the condition variable `done`, which is waited by `tpool_synchronize()` in a while loop the check for the three conditions above. This way, `tpool_synchronize()` will not busy wait.