Lab2 Analysis

October 5, 2023

Linear reduction: from 1 thread to 8 threads, the processing time is significantly reduced. This means that multi-threading is efficient for parallelizing this task. This reduction is more noticeable at the beginning, as the initial increase in threads (e.g. from 1 to 2 or 4) brings a huge parallelism advantage to the task.

Performance platform: From 8 threads to 16 threads, the reduction in processing time is not obvious. Although the number of threads has increased, the processing time has barely changed. This means that further increasing the number of threads does not bring more parallel benefits to the task.

Potential reason: Why is there no noticeable performance improvement from 8 to 16 threads? One possible reason is that my computer may only support 10 parallel threads. If a computer's hardware only supports 10 true concurrent threads, then any number above that will not bring a significant performance improvement because they cannot truly run in parallel. Threads will be scheduled by the operating system to run on a limited number of cores, potentially causing contention and context switches between threads, negating any performance benefits that adding more threads might bring.

Other potential factors: In addition to hardware thread limitations, other factors may limit performance gains. For example, the code may have locks in certain parts, preventing threads from fully executing in parallel. Alternatively, the task may be limited by I/O, memory bandwidth, or other system resources, which may at a certain point make increasing the number of threads no longer effective.

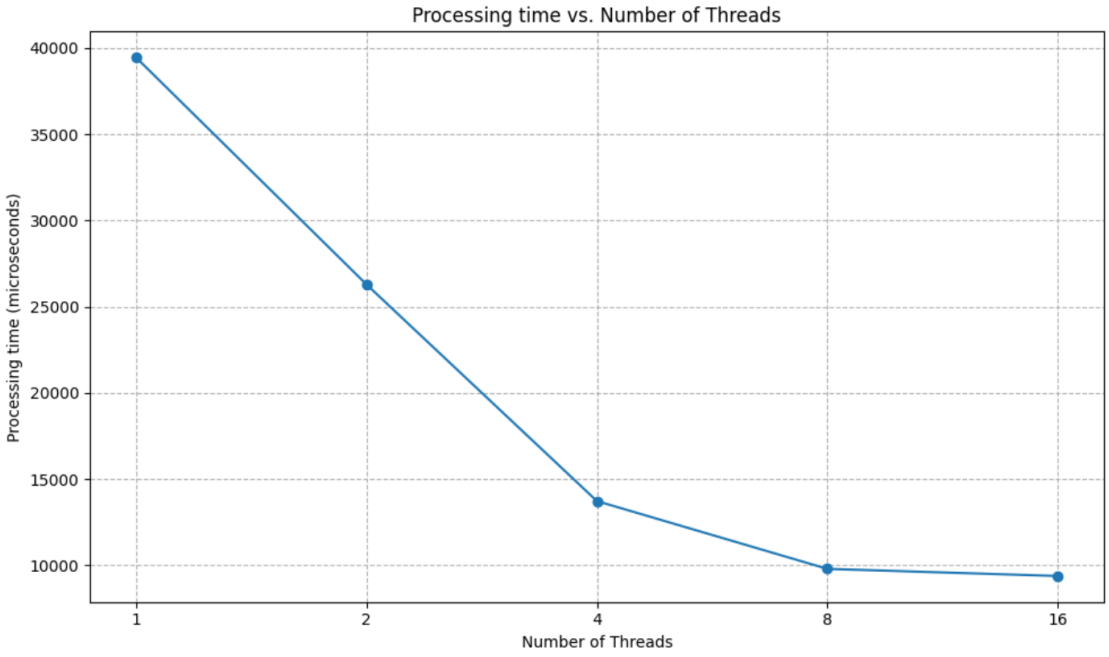

Figure 1: Performance Analysis