The Impact of Thread Count on Hash Computation

This experiment involves comparing the execution times of three different files being hashed using incremented thread counts. This report analyzes the trend of time reduction relative to thread counts and their process speed-up.

Setup

The system the experiment is being run has some important details that should be noted. The CPU model is Intel(R) Xeon(R) CPU E5-2695 v2 running at 2.40GHz on the x86_64 architecture. This CPU includes 48 cores and 2 threads per core, 32k L1 cache, 256k L2 cache, and 30720k L3 cache. The code is written and executed on a UNIX-based Operating System. There are 3 files, each subsequent file has more data that is being hashed than the last. The objective of this setup is to show how the CPU specifications will be reflected in the data.

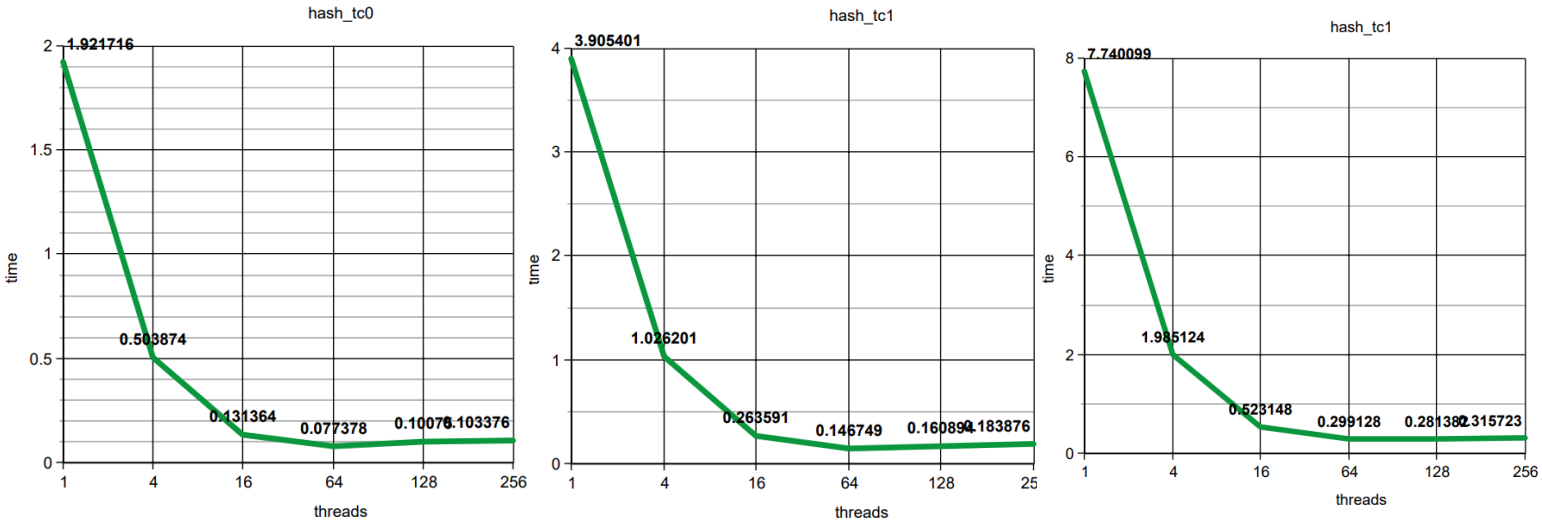
Trend in Computation Time

This experiment involves comparing the execution times of three different files at different numbers of threads. As depicted below in the graphs, generally, the computation time decreases as the number of threads increases. There are significant reductions in execution times when more threads are introduced due to parallel computing. The times increase in each subsequent

file most likely because ‘hash_tc0’ has the least amount of data while ‘hash_tc2’ has the most. However, this is not a linear trend, the graphs indicate that there are diminishing returns in the high end of thread counts. This is likely due to a couple of factors.

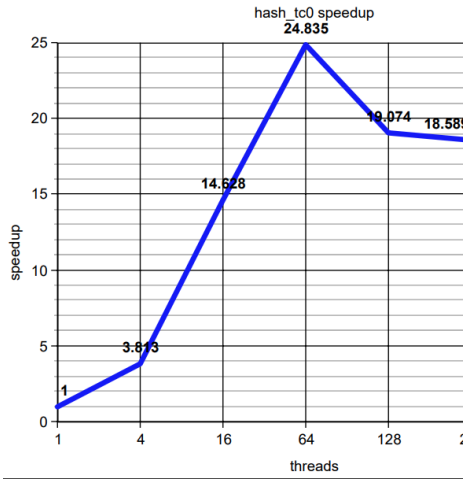
- More threads require more management by the computer. This increases tasks such as synchronization and communication between threads.
- The limitations of the hardware executing the processes. The amount of CPUs the hardware might not be able to take advantage of thread counts higher than 64. The hardware this is being run has 48 CPUs and 2 threads per core for a total of 96 threads on the hardware. The data supports these specifications because there are diminishing returns at 128 threads.

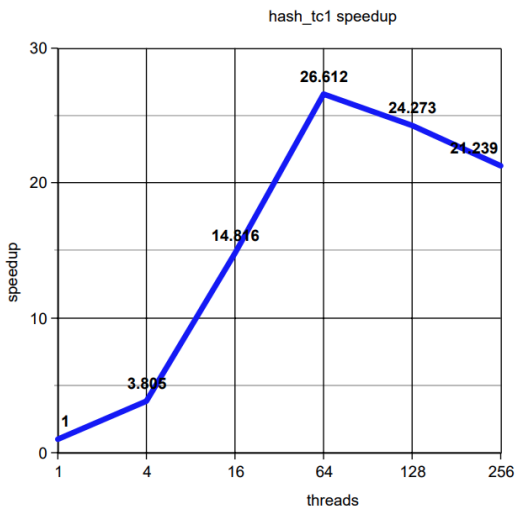
(Graphs are time in seconds vs the number of threads)

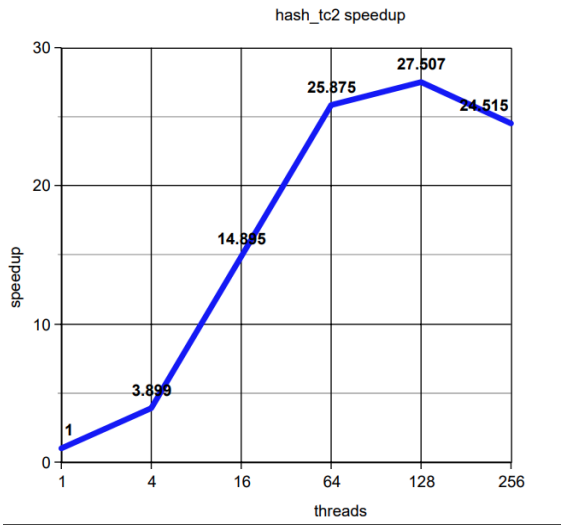


Speedup

Speed-up is a measure that shows the performance improvement using computation times. It is the ratio of time taken for a single thread to the time taken for n threads. All 3 files follow similar trends.







Each file indicates that the most aggressive speed-up is between threads 1 and 64. After thread 64, each file levels out (relative to before thread 64). It is most linear from thread 1 to thread 64. Because of this, speed-up is not proportional. The period in which it appears most proportional is the beginning where the speed-up is near-linear. Once again the decrease in speed-up is prevalent in threads after 96. If the hardware had more threads, this indicates the upward trend in speed-up would continue. Speed-up increases post thread 64 in subsequent files. The speed-up for thread 128 in ‘hash_tc2’ than in ‘hash_tc0’. This indicates that speed-up improves when more data is present since ‘hash_tc2’ includes more data for hashing. This is likely due to increased efficiency when working in parallel and better utilization of Cache Memory at higher volumes.

Conclusion

Increasing the number of threads in hash value computation typically results in reduced computation time up to a certain point, after which the benefits decrease or become negative due to overhead and hardware limitations. The speed-up achieved by adding more threads is not

directly proportional to the number of threads, emphasizing the need for a balanced and hardware-supported approach in parallel computing setups. This analysis underscores the importance of tailoring thread usage to the specific requirements and limits of the computing environment to optimize performance.