Nick DePalma
njd200001
CS 3377.505

Project 2 Report

CPU Specs:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 62
Model name:            Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz
Stepping:              4
CPU MHz:               1276.171
CPU max MHz:           3200.0000
CPU min MHz:           1200.0000
BogoMIPS:              4800.06
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ssbd rsb_ctxsw ibrs ibpb stibp tpr_shadow vnmi
flexpriority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts md_clear spec_ctrl intel_stibp flush_l1d
```

Program:

- The program computes the hash value of a file of varying sizes using a varying number of threads. The time to complete computing the hash value of the file compared to the number of threads used to compute the hash value are shown in the following graphs. In total, 5 test cases were used to form a stronger basis of result.

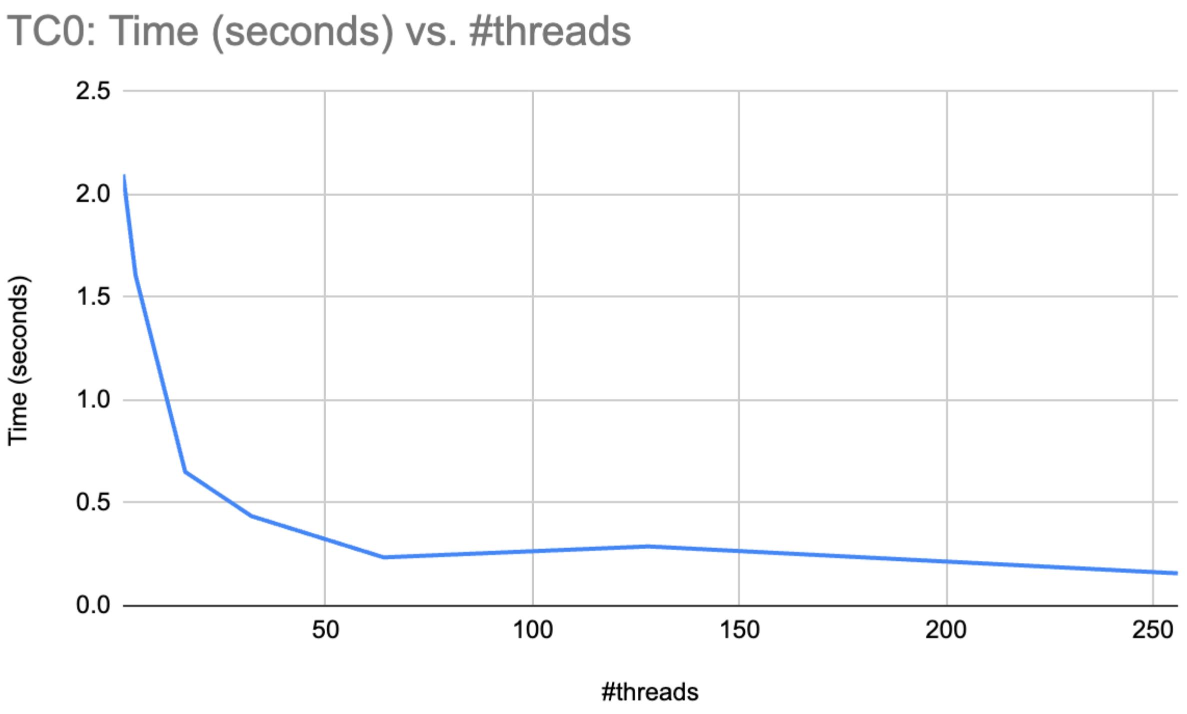
Constants:

- Block size = 4096 bytes

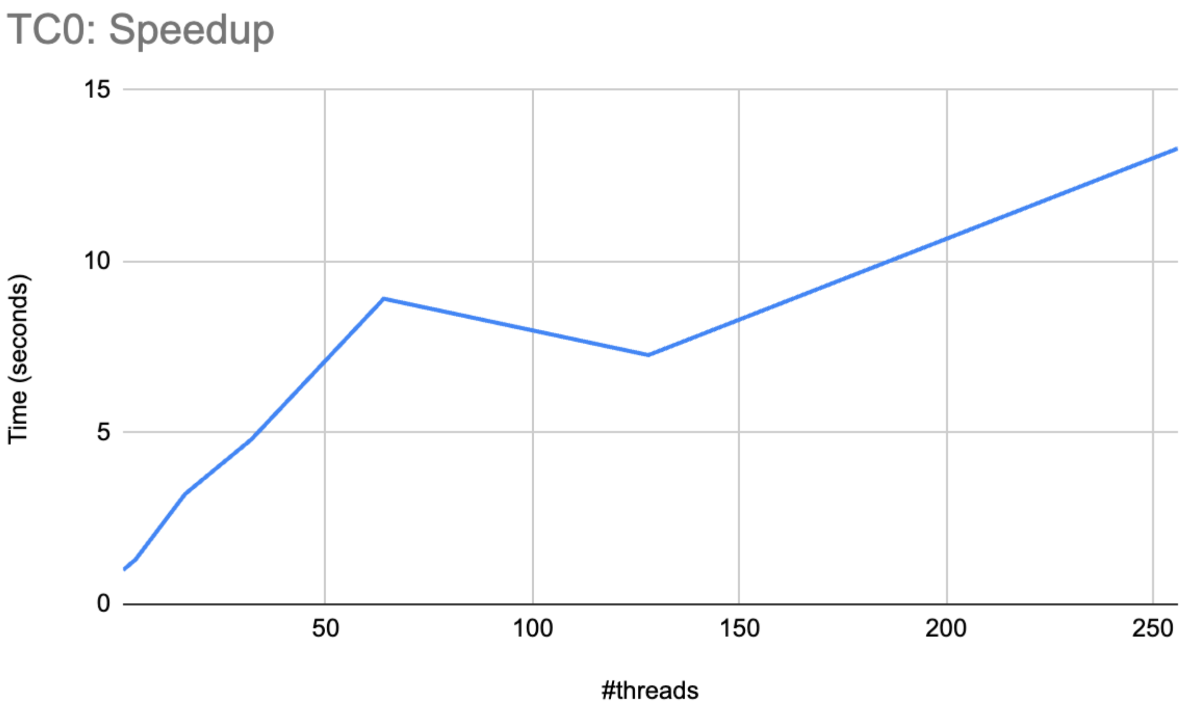
Nick DePalma
njd200001
CS 3377.505

Test case 0:

- Blocks: 65536
- Time vs. #threads:

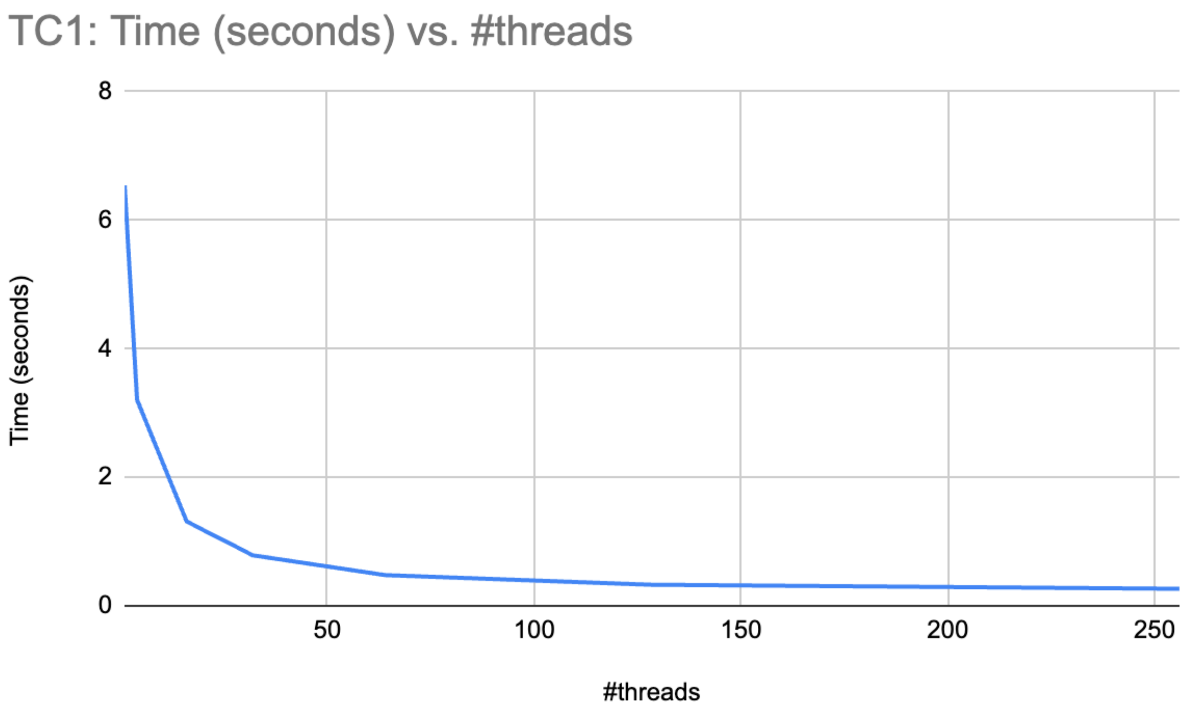


- Speedup(Time for 1 thread vs time for n threads):

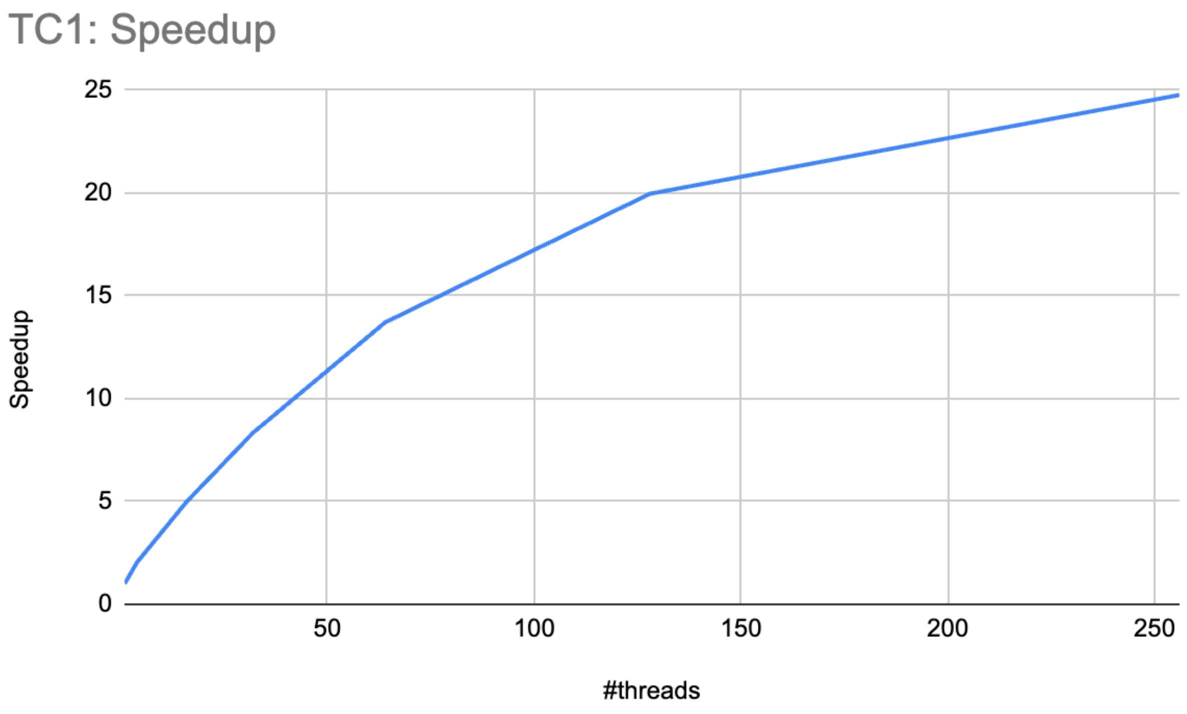


Nick DePalma
njd200001
CS 3377.505

- Test case 1:
    - Blocks: 131072
    - Time vs. #threads:

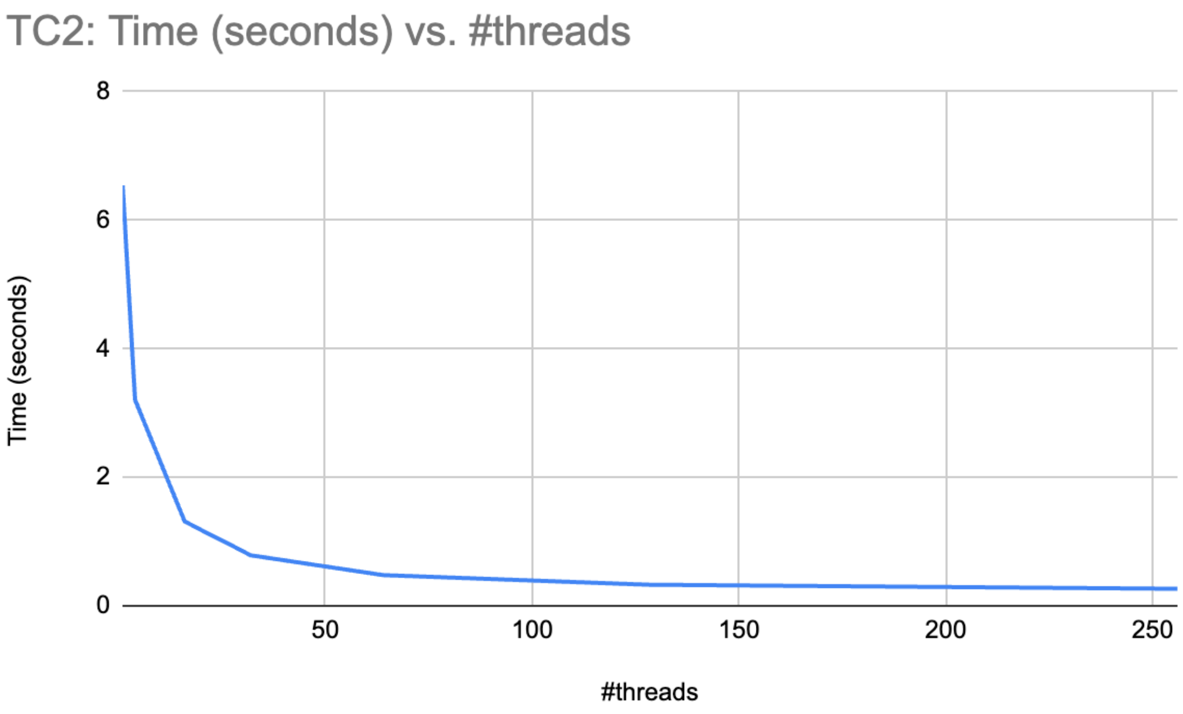


    - Speedup(Time for 1 thread vs time for n threads):

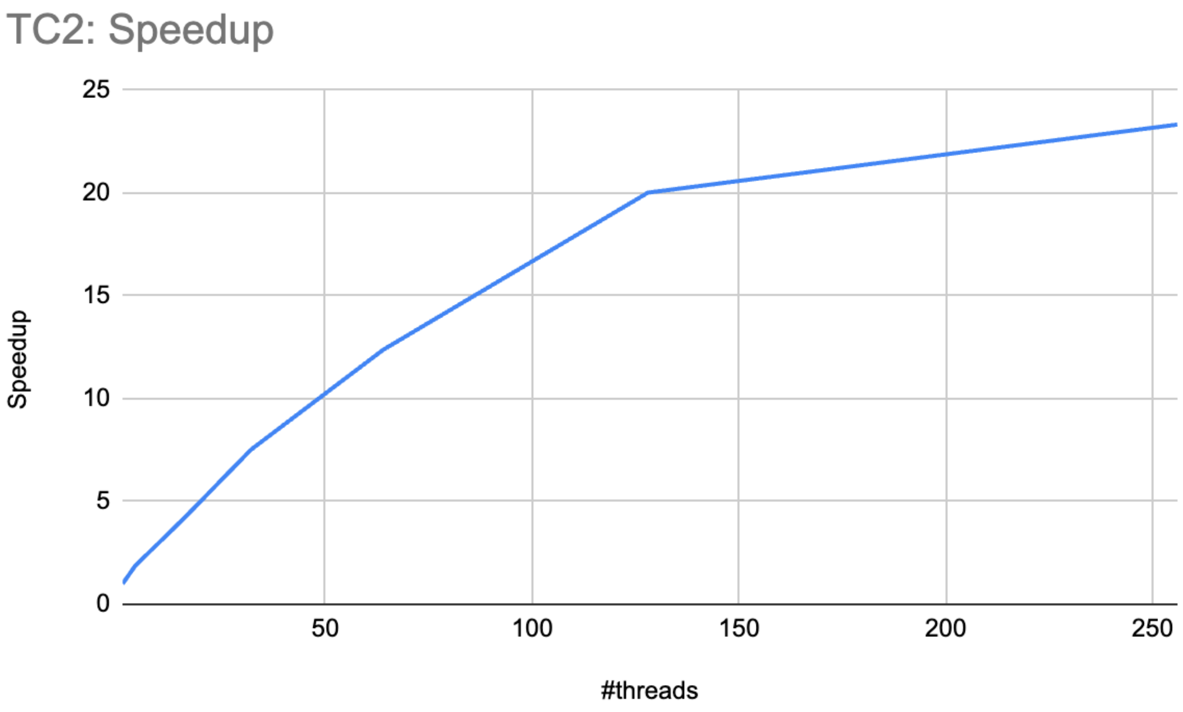


Nick DePalma
njd200001
CS 3377.505

- Test case 2:
    - Blocks: 131072
    - Time vs. #threads:

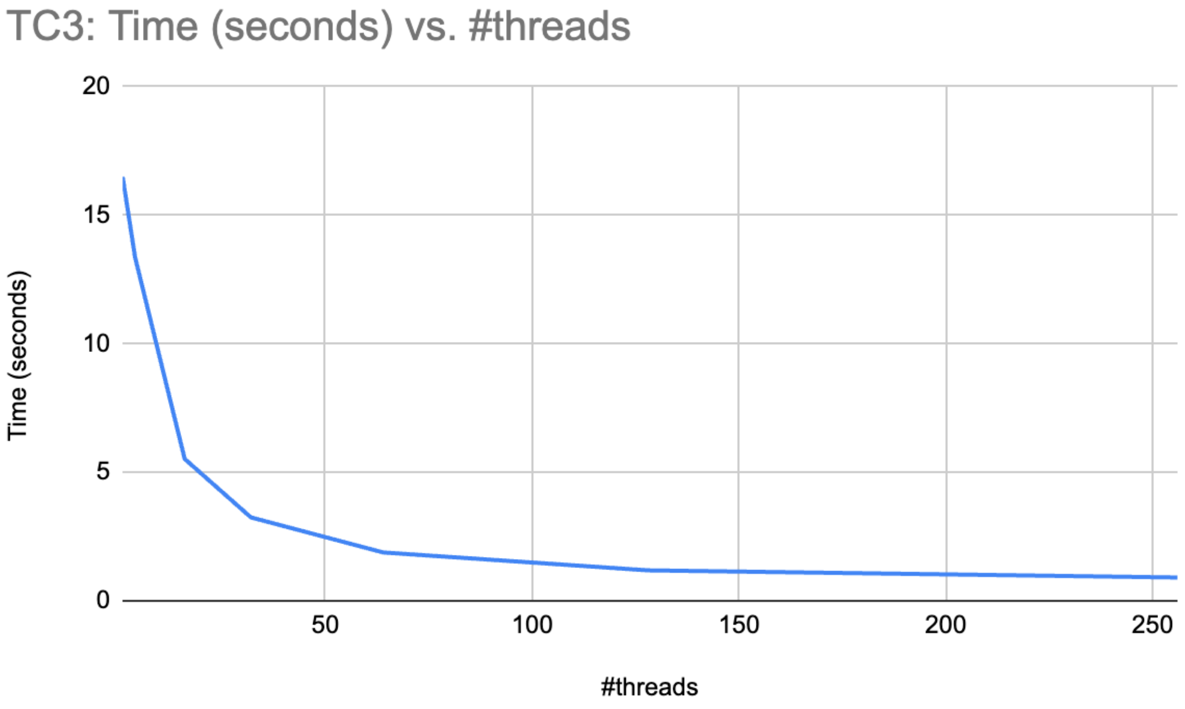


- Speedup(Time for 1 thread vs time for n threads):

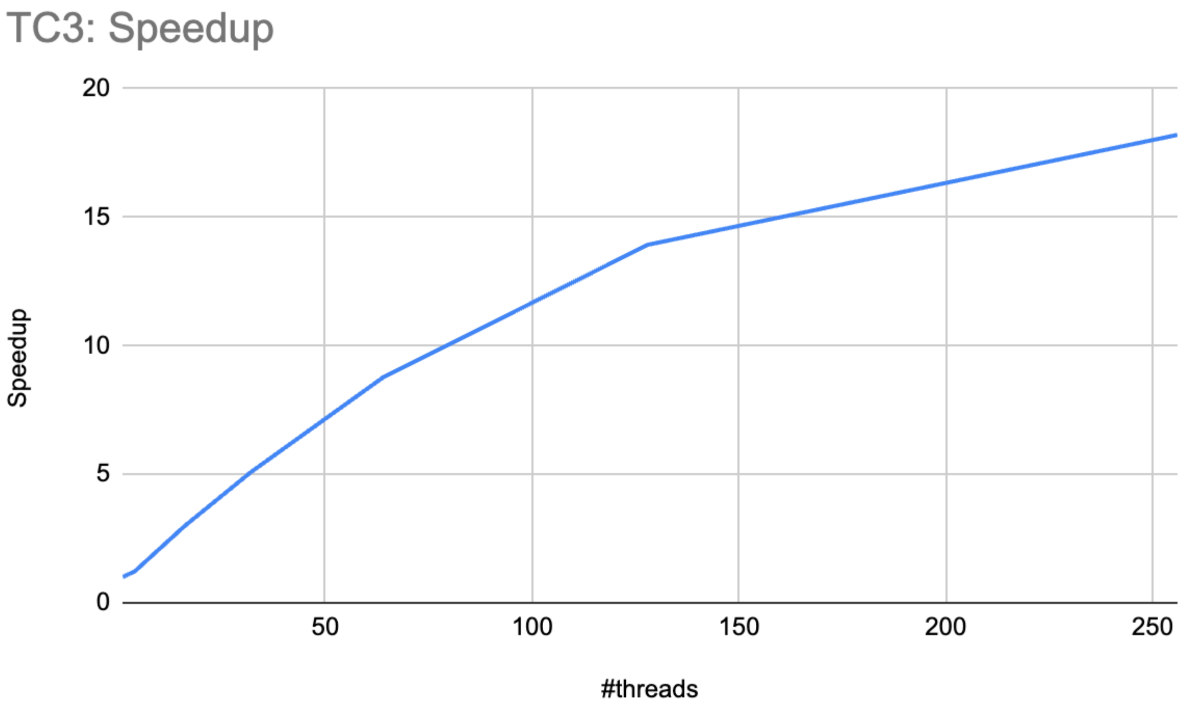


Nick DePalma
njd200001
CS 3377.505

- Test case 3:
    - Blocks: 524288
    - Time vs. #threads:

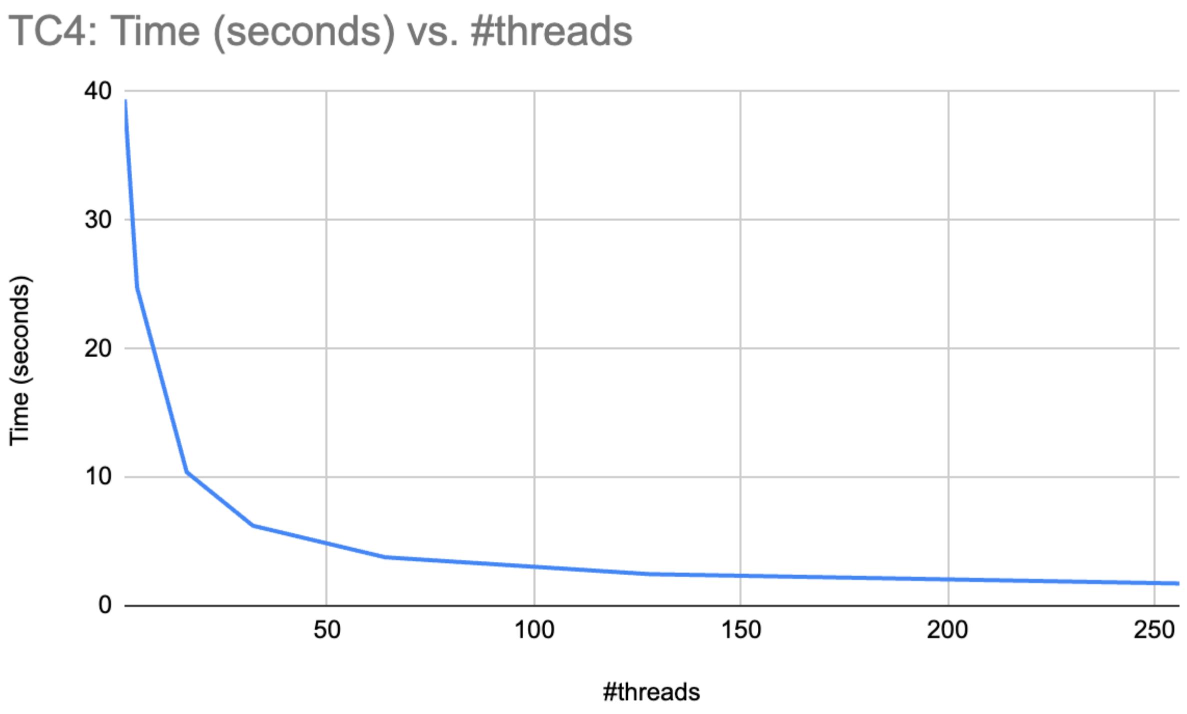


    - Speedup(Time for 1 thread vs time for n threads):

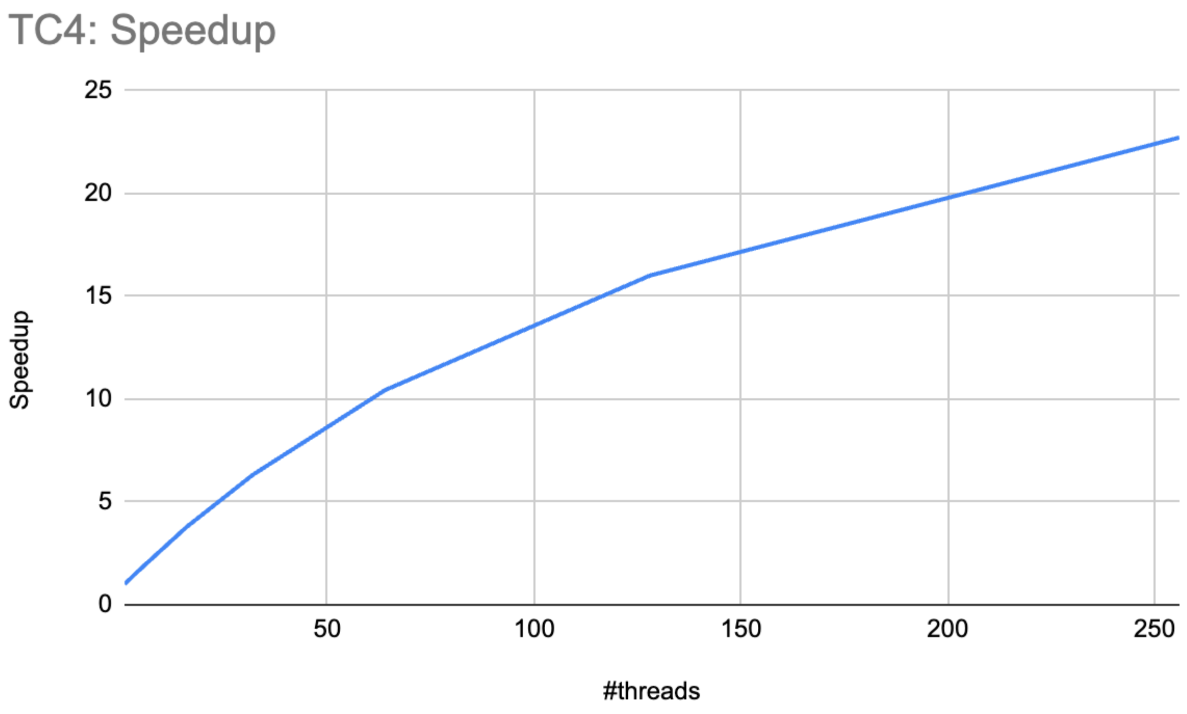


Nick DePalma
njd200001
CS 3377.505

- Test case 4:
    - Blocks: 1048576
    - Time vs. #threads:



    - Speedup(Time for 1 thread vs time for n threads):



Nick DePalma
njd200001
CS 3377.505

Conclusion:

- It is evident that the number of threads used to execute the calculation of the hash value has a direct, inverse correlation with the time taken to calculate the aforementioned value. In almost every case the time taken to compute the hash value decreased as the number of threads increased, with the exception of a few tests (as seen in the graphs). With consistent proportions across the 5 test cases of varying size, my expectations were met for the speedup and time taken for the varying number of threads.