Report for Ques 1

Question 1: DAXPY Vector Operations

What the Program Does

DAXPY stands for "Double precision A times X Plus Y" which is a basic vector operation used in scientific computing. The program takes two large vectors (about 1 million elements each) and performs the calculation X[i] = 2.5 × X[i] + Y[i] for every element. This is a very simple operation - just one multiplication and one addition per element.

Performance Results

When I ran this program, I got unexpected results. The sequential version took only 0.000167 seconds (167 microseconds). When I added parallel threads, instead of getting faster, the program actually got slower. With 2 threads it took 0.000225 seconds, with 4 threads it took 0.000218 seconds, and with 16 threads it took a massive 0.002402 seconds - over 14 times slower than sequential!

The speedup numbers tell the story clearly. Sequential has a speedup of 1.00× (baseline). With 2 threads the speedup dropped to 0.74×, meaning it was only 74% as fast as the original. With 16 threads it was catastrophically slow at 0.07× speedup. The efficiency also plummeted from 37% with 2 threads down to just 0.43% with 16 threads.

Hardware Analysis

The perf stat output revealed why this happened. The task-clock showed 8.07 milliseconds of CPU time, but the CPU utilization was only 0.539 cores, meaning only about half of one CPU core was being used on average. This is extremely low and indicates the CPUs are mostly sitting idle waiting for something.

There were 345 context switches in just 8 milliseconds, which is a lot of switching between threads. Page faults numbered 373, showing memory allocation activity. The elapsed time was 0.015 seconds but actual user time was only 0.006 seconds, with 0.002 seconds spent in system overhead.

Why This Happened

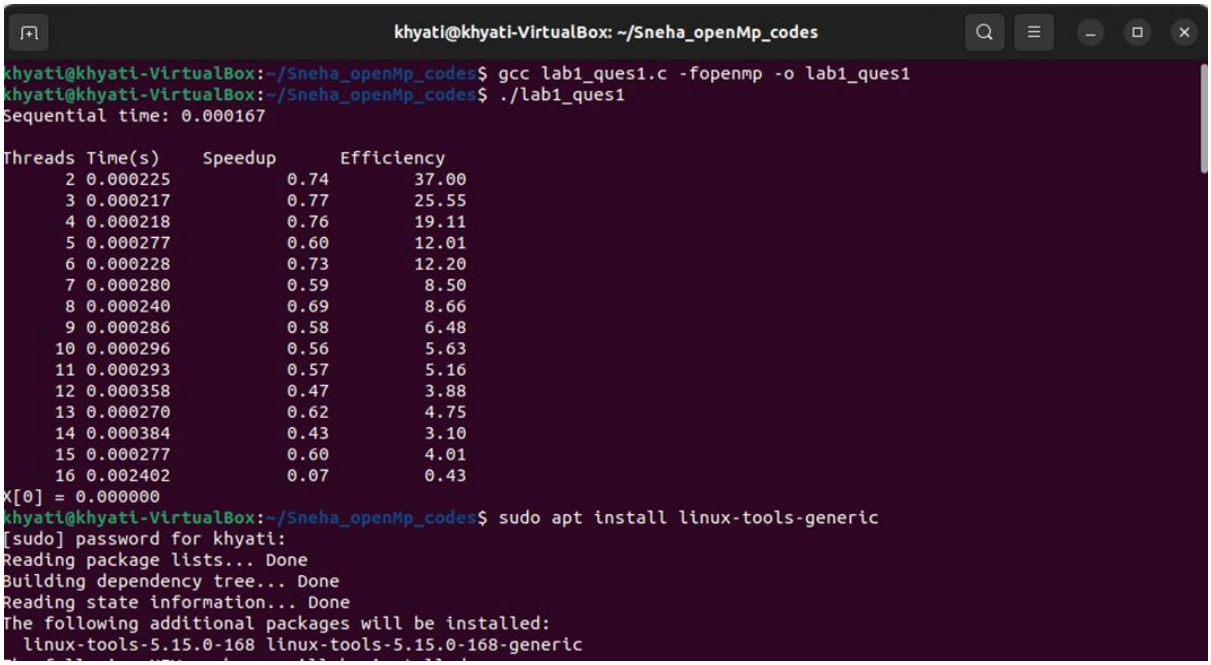
The problem is simple: this operation is too fast and too simple. Each element only needs one multiply and one add - that takes nanoseconds. But creating threads, coordinating

them, and combining their results takes microseconds. The overhead of parallelization is much larger than the actual work being done.

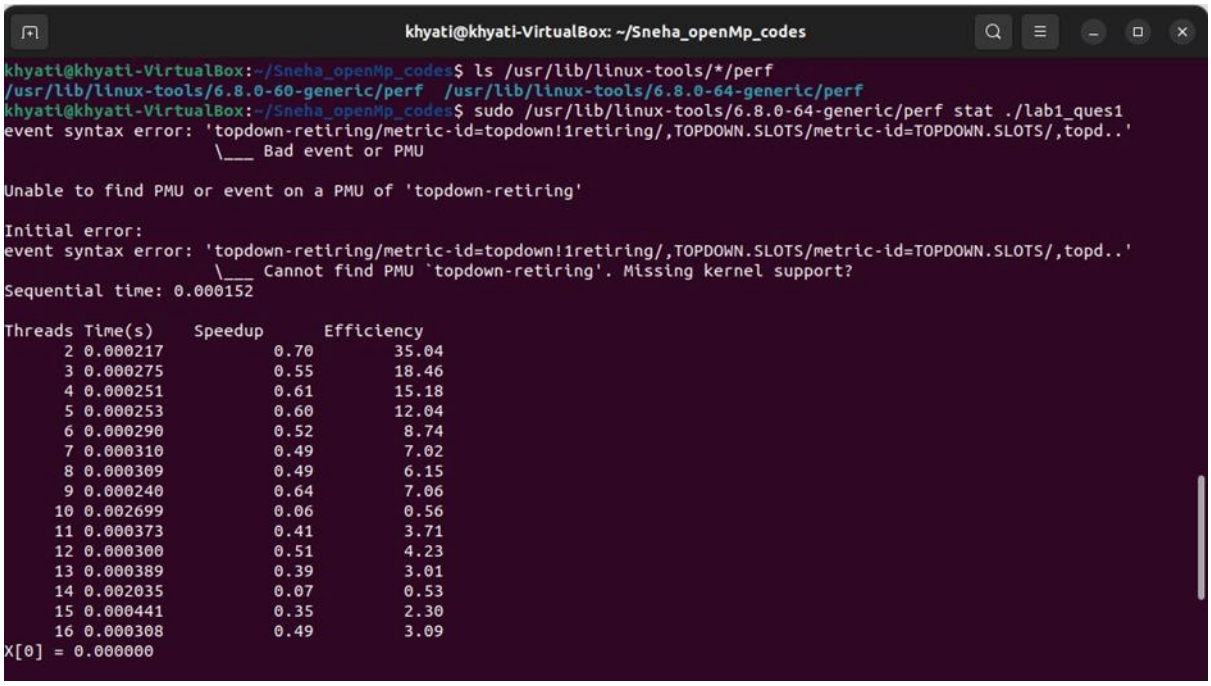
Additionally, this is a memory-bound operation. The CPU has to read data from RAM, do a tiny calculation, and write back to RAM. Reading and writing memory is much slower than computing, so the CPU spends most of its time waiting for memory, not computing. Adding more threads just creates more threads waiting for memory, which makes things worse. The low CPU utilization of 53.9% confirms this - the CPU cores are idle because they're waiting on memory.

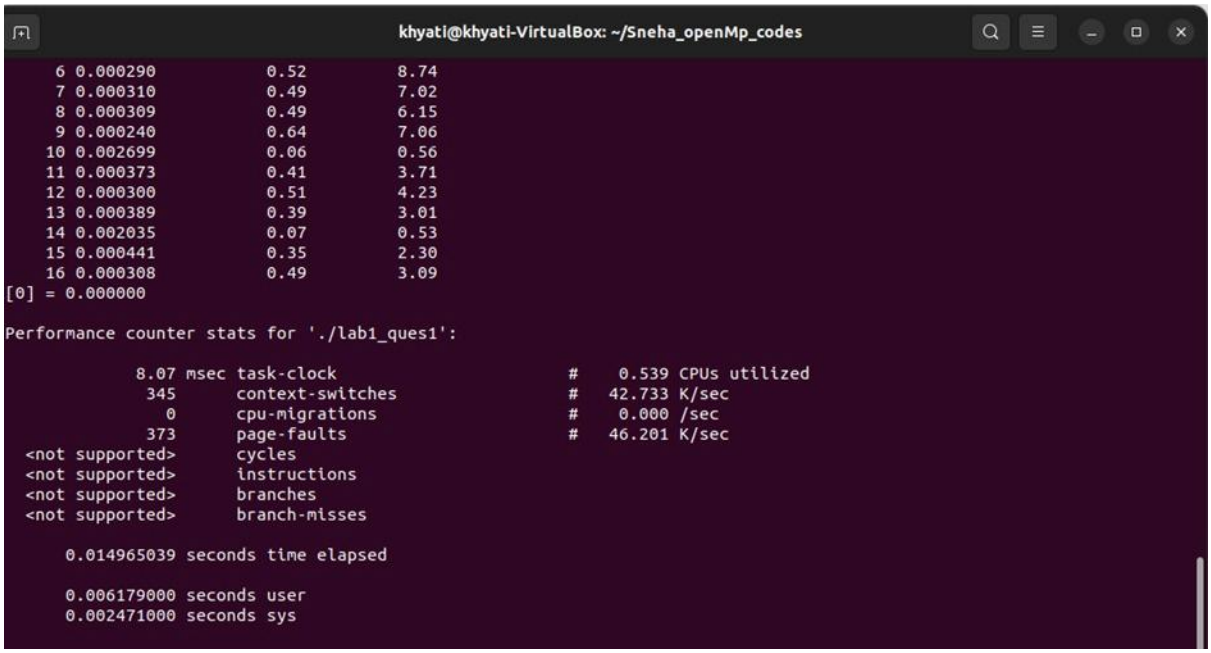
My Output

Terminal output showing timing for different thread counts:



Performance counter stats from perf:


khyati@khyati-VirtualBox: ~/Sneha_openMp_codes
khyati@khyati-VirtualBox:~/Sneha_openMp_codes$ ls /usr/lib/linux-tools/*/perf
/usr/lib/linux-tools/6.8.0-60-generic/perf /usr/lib/linux-tools/6.8.0-64-generic/perf
khyati@khyati-VirtualBox:~/Sneha_openMp_codes$ sudo /usr/lib/linux-tools/6.8.0-64-generic/perf stat ./lab1_ques1
event syntax error: 'topdown-retiring/metric-id=topdown!1retiring/,TOPDOWN.SLOTS/metric-id=TOPDOWN.SLOTS/,topd..'
___ Bad event or PMU
Unable to find PMU or event on a PMU of 'topdown-retiring'
Initial error:
event syntax error: 'topdown-retiring/metric-id=topdown!1retiring/,TOPDOWN.SLOTS/metric-id=TOPDOWN.SLOTS/,topd..'
___ Cannot find PMU `topdown-retiring'. Missing kernel support?
Sequential time: 0.000152
Threads Time(s) Speedup Efficiency
2 0.000217 0.70 35.04
3 0.000275 0.55 18.46
4 0.000251 0.61 15.18
5 0.000253 0.60 12.04
6 0.000290 0.52 8.74
7 0.000310 0.49 7.02
8 0.000309 0.49 6.15
9 0.000240 0.64 7.06
10 0.002699 0.06 0.56
11 0.000373 0.41 3.71
12 0.000300 0.51 4.23
13 0.000389 0.39 3.01
14 0.002035 0.07 0.53
15 0.000441 0.35 2.30
16 0.000308 0.49 3.09
X[0] = 0.000000


khyati@khyati-VirtualBox: ~/Sneha_openMp_codes
6 0.000290 0.52 8.74
7 0.000310 0.49 7.02
8 0.000309 0.49 6.15
9 0.000240 0.64 7.06
10 0.002699 0.06 0.56
11 0.000373 0.41 3.71
12 0.000300 0.51 4.23
13 0.000389 0.39 3.01
14 0.002035 0.07 0.53
15 0.000441 0.35 2.30
16 0.000308 0.49 3.09
[0] = 0.000000
Performance counter stats for './lab1_ques1':
8.07 msec task-clock # 0.539 CPUs utilized
345 context-switches # 42.733 K/sec
0 cpu-migrations # 0.000 /sec
373 page-faults # 46.201 K/sec
<not supported> cycles
<not supported> instructions
<not supported> branches
<not supported> branch-misses
0.014965039 seconds time elapsed
0.006179000 seconds user
0.002471000 seconds sys