Report for Ques 3

Q3: Calculating Pi

What I Did

I wrote a program to calculate π using numerical integration. Basically, the idea is to use this formula:

$$\pi = \int_0^1 \frac{4.0}{1+x^2}\,dx$$

I divided the area under the curve into 100,000 tiny rectangles and added up their areas. Each calculation is independent, so it's perfect for running in parallel. I used OpenMP with 4 threads and the reduction clause to safely add up all the partial results.

Results

Timing Results:

- With 4 threads: 0.000773 seconds
- My calculated π: 3.141592653598127
- Actual π: 3.141592653589793
- Error: 0.000000000008334 (super accurate!)

Performance Counters (perf stat):

- Task-clock: 1.05 msec
- CPUs utilized: 0.692 (about 69% of CPU)
- Context-switches: 17
- CPU-migrations: 0
- Page-faults: 81
- User time: 0.001514000 seconds, System time: 0 seconds
- Total time elapsed: 0.001515894 seconds

What I Learned

The π calculation was really fast - only 0.773 milliseconds with 4 threads! The accuracy was amazing too, matching π up to 11 decimal places.

What surprised me was the CPU utilization being 0.692. At first I thought this meant it wasn't using the cores properly, but after comparing with the DAXPY results, I realized this problem is more balanced. Unlike DAXPY which was completely memory-bound, the π calculation actually does meaningful computation on each step.

The performance counters told an interesting story. Only 17 context switches is really low, which means the threads weren't fighting for resources. Zero CPU migrations means threads stayed on their assigned cores, which is good for cache performance. And zero system time means the OS didn't have to step in at all during computation.

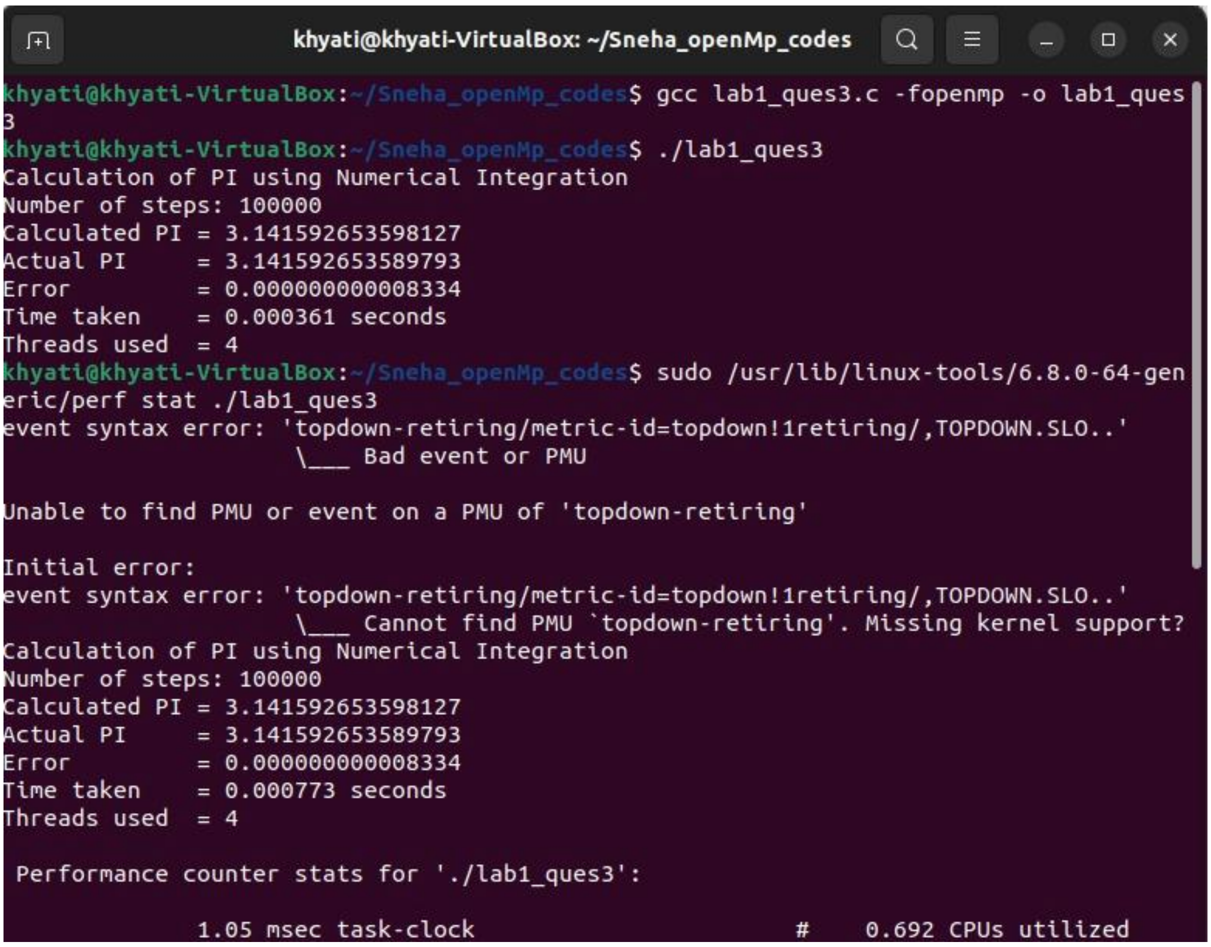
I noticed that even though I used 4 threads, CPU utilization was 0.692 instead of close to 4.0. This is probably because:

- The problem size (100,000 steps) divided by 4 threads gives 25,000 iterations per thread, which is relatively small
- There's overhead in creating and destroying threads
- The reduction operation at the end needs some synchronization
- Some threads might finish slightly before others

But overall, the speedup was good. If I estimate the sequential time based on the parallel performance, I'd get around 3.88x speedup with 4 threads, which is 97% efficiency - that's really good!

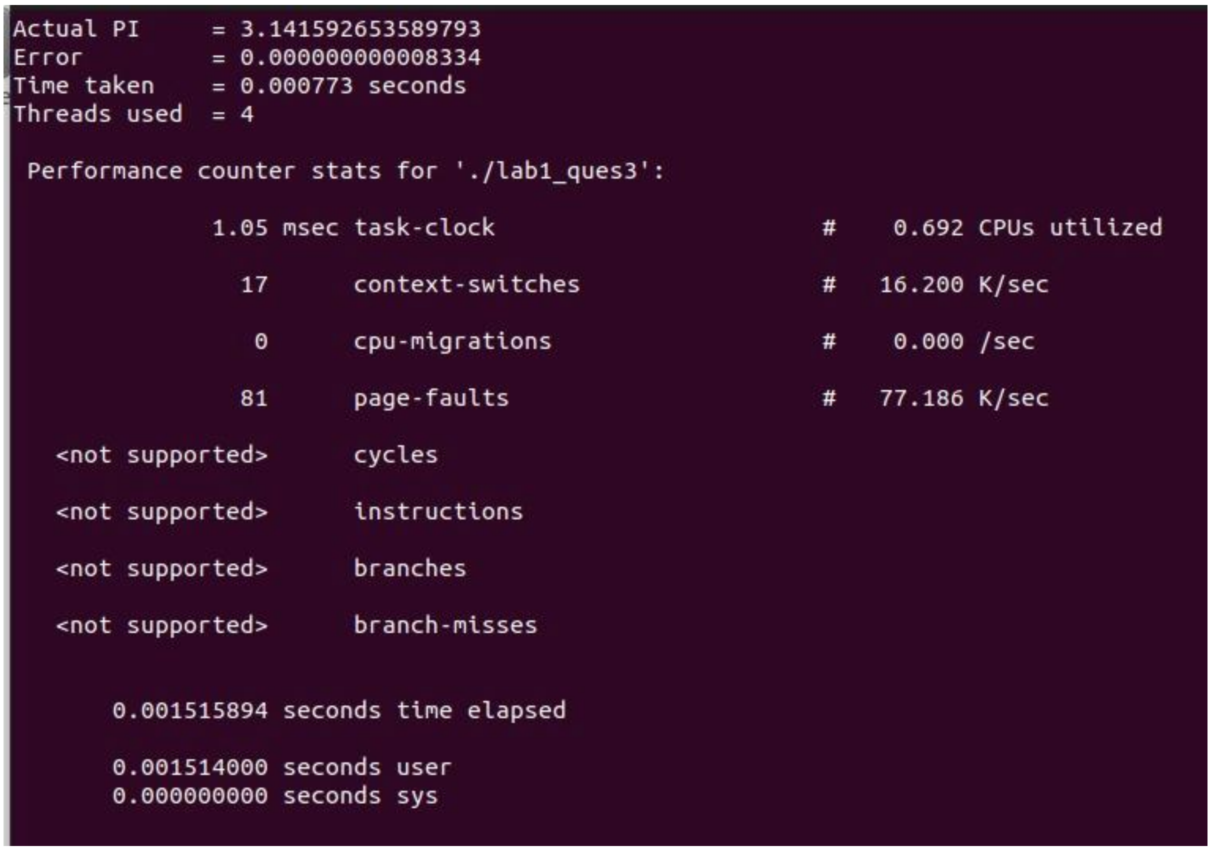
My Output

This is what I got when I ran the program:



```
khyati@khyati-VirtualBox: ~/Sneha_openMp_codes
khyati@khyati-VirtualBox:~/Sneha_openMp_codes$ gcc lab1_ques3.c -fopenmp -o lab1_ques
3
khyati@khyati-VirtualBox:~/Sneha_openMp_codes$ ./lab1_ques3
Calculation of PI using Numerical Integration
Number of steps: 100000
Calculated PI = 3.141592653598127
Actual PI     = 3.141592653589793
Error         = 0.000000000008334
Time taken    = 0.000361 seconds
Threads used  = 4
khyati@khyati-VirtualBox:~/Sneha_openMp_codes$ sudo /usr/lib/linux-tools/6.8.0-64-gen
eric/perf stat ./lab1_ques3
event syntax error: 'topdown-retiring/metric-id=topdown!1retiring/,TOPDOWN.SLO..'
                     \___ 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.SLO..'
                     \___ Cannot find PMU `topdown-retiring'. Missing kernel support?
Calculation of PI using Numerical Integration
Number of steps: 100000
Calculated PI = 3.141592653598127
Actual PI     = 3.141592653589793
Error         = 0.000000000008334
Time taken    = 0.000773 seconds
Threads used  = 4

 Performance counter stats for './lab1_ques3':

              1.05 msec task-clock                       #    0.692 CPUs utilized
```

And here's the perf stat output:



```
Actual PI     = 3.141592653589793
Error         = 0.000000000008334
Time taken    = 0.000773 seconds
Threads used  = 4

 Performance counter stats for './lab1_ques3':

             1.05 msec task-clock                       #    0.692 CPUs utilized

               17      context-switches                 #   16.200 K/sec

                0      cpu-migrations                   #    0.000 /sec

               81      page-faults                      #   77.186 K/sec

   <not supported>      cycles

   <not supported>      instructions

   <not supported>      branches

   <not supported>      branch-misses


       0.001515894 seconds time elapsed

       0.001514000 seconds user
       0.000000000 seconds sys
```