Sergi Aliaga

Northeastern University

Homework 1

Question 1

In this problem, you are to select a set of 3 single-threaded benchmark programs. Try to provide some diversity in the set of workloads you have chosen (e.g., floating point, integer, memory intensive, sparse). Then complete the following experiments using these benchmarks. Make sure to provide as many details as possible about the systems you are using, and where you obtained the source code for these benchmarks.

a. *Compile and run these 3 benchmarks on a Linux-based system of your choosing (you can also use either the COE systems or the Discovery systems). Provide detailed information about the platform you chose, including the model and frequency of the CPU, number of cores, the memory size, and the operating system version. You should record the execution time, averaged over 10 runs of the program. Is any run of the program faster than another? If so, comment on any difference you observe in your write-up, providing some justification for the differences.*

The source code for the chosen benchmarks is included in the same folder of this document and are described below:

"1CPUBenchmark.c": This benchmark evaluate the performance of the CPU in terms of floating-point operations per second (FLOPS). It runs a set of 30 operations (2-element additions and subtractions) per loop iteration and thread. The test has been performed with 1e8 loops and 1 thread. Source code can be found here: https://github.com/arihant15/Performance-Evaluation-Benchmark/tree/master/CPU

"2BenchmarkFLOPS64.c": This benchmark measures performance in terms of 64bit FLOPS. It performs a total of 51*30 operations per loop and thread and computes the number of operations carried out in 1 minute. In this case, operations concern multiplication, addition, and subtraction. Source code can be found here: https://github.com/shaswata56/BenchUtil/blob/master/BenchmarkFLOPS64.c

"3CPUBenchmark.c": This benchmark measures performance in terms of FLOPS by computing a 20 operations addition per loop and thread. The test is run with a total of 1,5e8 loops, to match the same number of operations as the first benchmark. Source code can be found here: https://github.com/harrism/mini-nbody/blob/master/nbody.c

The benchmarks has been run on my PC: Apple M1 Max, 10 cores (8 performance at 3.22GHz and 2 efficiency at 2.064GHz), 64Gb of unified memory and running macOS Monterey version 12.1. Side comment, running these benchmarks has been the first time I heard the fans turned on in this machine.

The results are reported below:

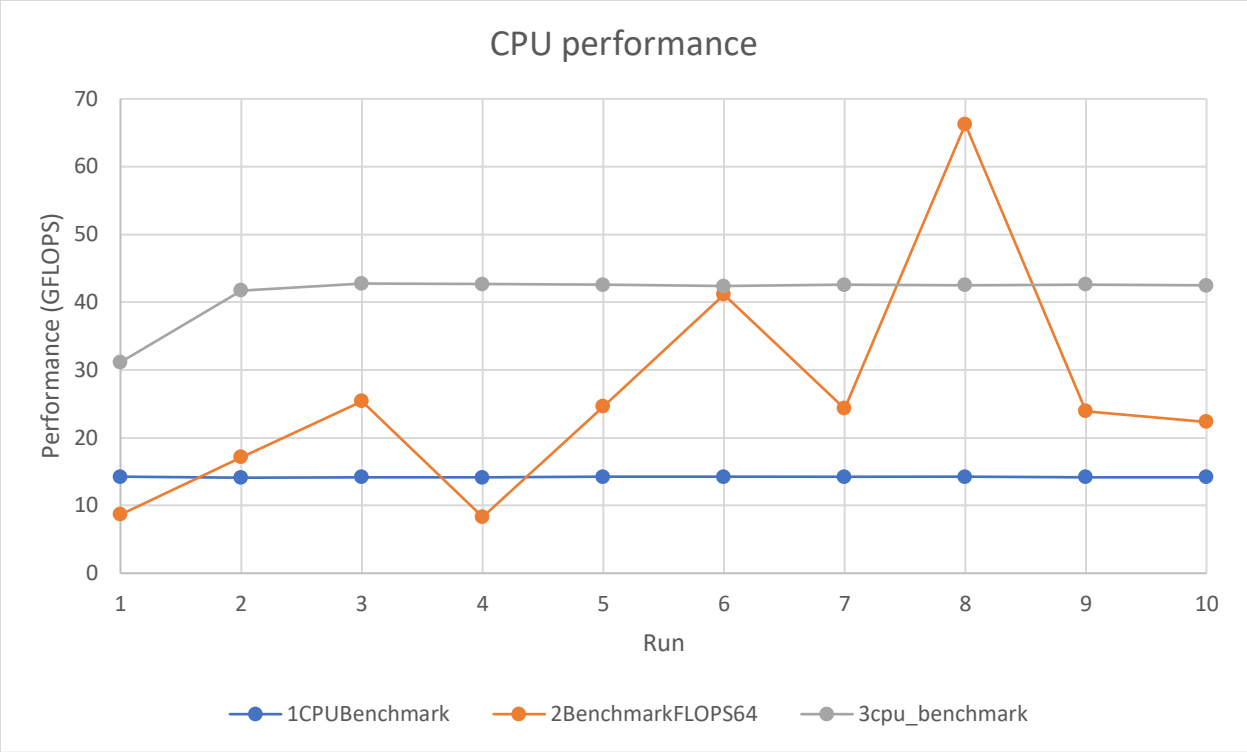

Figure 1: CPU performance according to the 3 performed benchmarks

Benchmark	1	2	3
Avg. performance (GFLOPS)	14.18	26.2	41.33

Table 1: Averaged results

From the results, we outline the following observations

- CPU performances between each test is very different. This might be due to the nature of each test, since, as described before, the operations performed by each one are not exactly de same. This result highlights the importance of running the exact same benchmark in two different machines to get a fair comparison between them.
- Benchmark 1 and 3 show very consistent results compared to benchmark 2. This is because of the approach of benchmarks2. In benchmarks 1 and 3 a finite set of operations are carried out and the time used for them is measured, whereas in benchmark 2, the number of operations is not limited, only the time to make them. Therefore, the inconsistency in the results of benchmark 2 might be due to the use of different types of cores at each run (performance or efficiency), resulting in very different results across realizations.

b. *Next, explore the compiler optimizations available with the compiler on your system (e.g., gcc or g++), and report on the performance improvements found for the 3 workloads. Describe the optimization you applied, and provide insight why each of your benchmarks benefitted from the specific compiler optimization applied.*

For this section of the assignment, section 1 was repeated but using the gcc compiler optimization levels 2 and 3. The optimizations flags implemented by each level can be found in https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html. Results are reported below (benchmark 2 was compiled but for some reason the results where of 0.0 GFLOPS every time, reason why they have been omitted in the report):

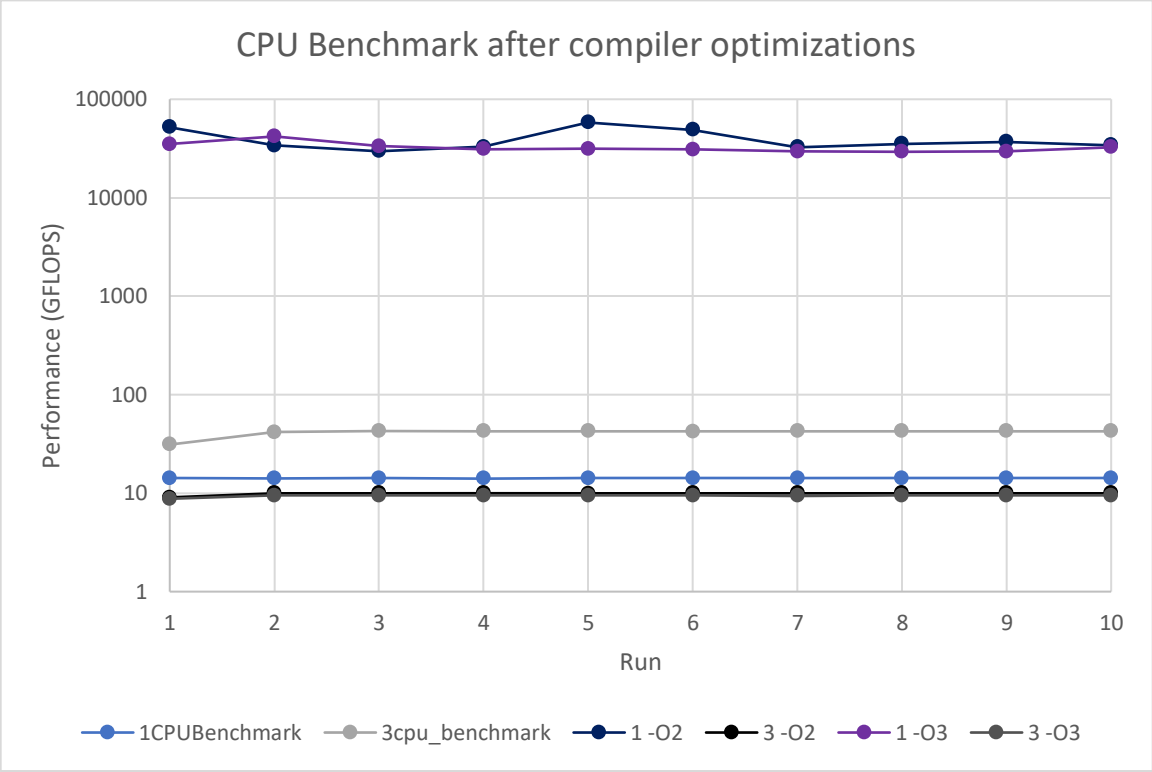

Figure 2: Benchmark results after applying complier optimization levels 2 and 3.

From the above results we report the following:

- Performance of benchmark 1 is greatly improved by using the optimization of the compiler. However, not much improvement is noticed between the 2nd and 3rd optimization levels
- For benchmark 3, performance is decreased when applying the compiler optimizations. Moreover, the performance at the 2nd and 3rd optimization levels is very similar again.

c. *Assume that you were going to rewrite these applications using pthreads. Describe how you w ould use pthreads to obtain additional speedup by running your benchmark on multiple cores.*

In all of the three presented benchmarks there is the possibility of extending the test to parallel performance with the use of pthreads. The benchmark basically asks the user how many threads he wants to use and assign a set of operations to each of the threads spawned. Once the benchmark is run, the results of each thread are added together and the performance of the total CPU is reported, as well as the maximum single-CPU performance (maximum across threads).