High Performance Computing: Homework 3

Dmitriy (Tim) Kunisky [dk3105]

- Processor: Intel(R) Core(TM) i5-5257U CPU @ 2.70GHz
  - 2 cores; 4 threads with hyperthreading
  - Maximum flop rate: 19.76 Gflop/s, or 4.94 Gflop/s per core
  - Maximum main memory bandwidth: 25.6 GB/s
- Compiler: Apple LLVM version 9.1.0 (clang-902.0.39.2)

Problem 1: Approximating Special Functions

I implemented the higher-order Taylor approximation for the `sin4_vec()` function. The resulting function is roughly four times faster than the reference implementation and two times faster than the serial implementation of the Taylor series of the same order.

Problem 2: Parallel Scan

To parallelize the scan routine over p threads, I divide the input array (of length N) into $p - 1$ parts of length $\lceil \frac{N}{p} \rceil$, and one part containing the remaining entries (possibly having slightly smaller length if N is not divisible by p). Then, as suggested in the problem, I perform the prefix sum calculation in parallel for each of these segments, and then correct with one final serial scan. The timing results are presented below for the serial implementation and $p \in \{1, 2, 3, 4\}$ when $N = 10^8$.

Threads	Runtime (s)
Serial	0.52
1	0.53
2	0.39
3	0.34
4	0.30

The efficiency gradually improves with the number of threads, reaching an improvement of approximately 40% over the serial or single-threaded version when four threads are used. This is reasonable to expect, since we always retain the overhead of the final serial correction which is not parallelized.