P. Koumoutsakos, W. Petersen, P. Hadjidoukas
ETH Zentrum, CLT E 13
CH-8092 Zürich

Fall semester 2017

Set 3 - Amdahl's Law, OpenMP

Issued: October 13, 2017
Hand in (optional): October 20, 2017 8:00am

Question 1: Amdahl's Law

a) Assume you work on Euler and you have one node with 24 cores that you can use to solve a problem in parallel for which 91% of your code is parallelizable. Can you get a speedup of 8? If so, how many cores are needed at least?

b) Profiling a serial code for Molecular Dynamics you find that 90% of the time is spent in a large loop with independent iterations (perfectly parallelizable with N threads), another 5% is spent in a region that can be parallelized with at most 2 threads and the remaining part is purely serial.

Given Amdahl's law, what is the strong scaling for $N \to \infty$?

For what value of N is the speedup equivalent to 90% of the asymptotic maximum?

Question 2: Parallel Monte Carlo using OpenMP

Monte Carlo integration is a method to estimate the value of an integral as the mean over a set of random variables. For instance,

$$\int_a^b f(x)\,dx \;\approx\; \frac{b-a}{N}\sum_{i=1}^{N} f(X_i),$$

where X_i are samples from a uniform distribution on the interval $[a, b]$.

Consider integration of $f(x) = 4\sqrt{1-x^2}$ over the interval $[0, 1]$ which evaluates to

$$\int_0^1 4\sqrt{1-x^2}\,dx \;=\; \pi.$$

a) Given a serial implementation of the algorithm provided in the skeleton code, write a parallel version using OpenMP by splitting the sampling work among multiple threads. Make sure you do not introduce race conditions. If possible, try various directives such as

- `omp parallel` with separate `omp for`;
- `omp parallel for`;

- `omp parallel reduction`.

Note that the random number generators used in the code are not thread-safe. Therefore, each thread should have its own copy of the generator initialized with a different seed (e.g. depending on thread id).

b) Make strong scaling plots up to 24 cores. Control the number of threads by setting the environment variable `OMP_NUM_THREADS` or with function `omp_set_num_threads()`.

c) Answer the following questions:

- Is the amount of computational work equal for all threads?
- Are the numerical results identical between runs? Check cases of (a) serial program, (b) OpenMP with one thread, (c) OpenMP with multiple threads.