High Performance Computing for Science and Engineering

Exercise 3: Monte Carlo, OpenMP, False Sharing

Common mistakes

Implementation

- no unique seed for random engine
  - use `g.seed(tid + 1)` or pass to constructor
  - `seed(0)` and `seed(1)` are equivalent for `std::linear_congruential_engine`
- random engine initialized inside for-loop <- no random sequence
- wrong padding for array of random engines
  - array size `nthreads * 64` <- wasted memory, `std::mt19937` is 2500 bytes
  - array size `nthreads * (64 / sizeof(std::default_random_engine))` <- may be 0
  - use `struct { std::default_random_engine g; char p[64]; }`

Implementation

- `#pragma omp critical` in loop body <- serialization
- array size `omp_get_num_threads()`
  returns 1 outside parallel region, use `omp_max_num_threads()`
- cases `C2()` and `C3()` allowed only `#pragma omp parallel for`
- no need in `firstprivate` for variables with default initialization, use just `private`
- std::uniform_real_distribution is lightweight, can be inside loop body; but not thread-safe

Analysis

- use at least 24 cores on Euler
- false sharing depends on optimization (`-O2`, `-O3`, `-Ofast`)
- `exactly the same numerical results` means value of integral not timings,
  may differ due to reordering of floating point operations