Parallel histogram

Parallel computing
Assignment 1
Multithreading in C++

Task

- data generator
  - returns integers 0..N
- histogram bucket size 1
  - N+1 buckets
  - number of occurrences of each value counted separately
- specified sample size
  - number of generator results to be entered into the histogram
- N threads
  - `std::thread::hardware_concurrency()`
  - each has copy of the generator
  - all access the same histogram
  - each processes a part of the sample size

Example

- two threads, two generators, N=10, sample_size=8
  - thread1, generator1: 0, 5, 0, 4
  - thread2, generator2: 0, 2, 3, 4
- histogram:
  - 0: 3
  - 1: 0
  - 2: 1
  - 3: 1
  - 4: 2
  - 5: 1
  - 6: 0
  - 7: 0
  - 8: 0
  - 9: 0
  - 10: 0
  - total: 8

Implementation variants

1. histogram buckets as atomics
   - no other synchronization, apart from thread start and join
2. single mutex for the histogram
3. mutex for each bucket
4. mutex per M buckets
   - N/M mutexes
   - N%M!=0
     - one more mutex for the last N%M buckets
     - just go with N/M+1 mutexes
       - if the last is unused, the price is not that high

Time measurement

- make sure to compile at least with -O2

- initialize histogram
- `auto t1 = chrono::high_resolution_clock::now();`
- start threads
- join threads
- `auto t2 = chrono::high_resolution_clock::now();`
- display results
- `cout<<chrono::duration<double>(t2-t1).count()<<endl;`

Work distribution

- static
  - each thread gets to do equal fraction of the sample size
    - plus/minus 1, if not evenly divisible
  - up to 8 points max for the assignment
- dynamic
  - at least one form of dynamic work distribution
  - don’t expect improved performance
  - up to 10 points max for the assignment

Experiments

- run and write a quick report
- submit PDF, raw data, and source code
- run experiments on cora or your machine
  - if hardware parallelism is at least 4 (2 cores with hyperthreading)

- set large-enough sample size
  - so that runtime is ~1s
- compare performance for different variants and different N
- compare variants 3 and 4 for different values on N and M
  - 2D/3D plot of who is faster

Template

- main.cpp
  - in Moodle
  - template for your solution
  - does all the necessary things (with static work distribution) apart from synchronization of histogram
    - the result is wrong
  - for static work distribution, you should only modify the histogram class
    - however, you can also modify main to add test automation
  - for dynamic work distribution, you also need to modify the main and worker
  - to compile on cora:

```
g++-7.2.0 main.cpp -O2 -lpthread -ohistogram
```

Machine

- The cora cluster
- http://www.par.univie.ac.at/teach/doc/cora.html
- account infomration:
  - login: aMatrikelnummer
    - e.g., a01234567
  - password: ss2020 + first letter (capital) of surname + first letter (capital) of (first) forename
    - e.g., John Smith will have ss2020SJ
    - e.g., José Antonio Gómez Iglesias will have ss2020GJ, as the university represents the name as “José Antonio” and “Gómez Iglesias”
  - if you already have an account on cora, use that – no second account will be created
    - for this semester, old accounts are removed