High Performance Computing

Homework #3: Part B

Due: Saturday February 22 2016 by 11:59 PM (Midnight)

Email-based help Cutoff: 5:00 PM on Mon, Feb 21 2016

Maximum Points For This Part: 10

Objective
The objective of this part of the homework is to develop a micro-benchmark in C++ to compare the performance of using methods vs. lambdas in algorithms.

Background:

In C++ many standard algorithms such as: `std::sort`, `std::count_if` accept predicates to which you may pass methods or specify lambdas. Recollect that in computer science, a "predicate" is a mathematical function that takes zero or more parameters and returns a Boolean value (i.e., true or false). Predicates that accept one parameter are called unary predicates while predicates that accept two parameters are called binary predicates.

Scientific question:

Is there a performance difference between using an inline function (such as those used in C language) versus supplying a C++ lambda for a predicate?

Hypothesis:

Using a lambda would be faster as lambdas can be effectively inlined (thereby avoiding overhead of a function call and control hazards) and the compiler can better optimize lambda.

Instructions:

This homework requires you to design an experimental benchmark to test the aforementioned hypothesis and conduct experiments to accept/reject the hypothesis. Complete the report (in this document), save it as a PDF file, and upload it to Canvas.

Here are a few tips to help design your benchmark:

1. Working with your `std::sort` algorithm would be effective (because of its time complexity arising from the number of times the predicate is used for comparisons). See slides off Canvas for example on using `std::sort`.
2. Ensure your benchmark runs for about 10 seconds with optimizations (`-O3`).
3. Don't overcomplicate the benchmark. A few lines of code can be more useful than 100s of lines of code.

Name: Henry Ni

Apparatus Used (experimental platform)

The experiments documented in this report were conducted on the following platform:

Component	*Details*
CPU Model	Intel Core i5-4258U
CPU/Core Speed	2.4 GHz
Main Memory (RAM) size	8388608 KB (8 GB)
Operating system used	OS X 10.11.3
Interconnect type & speed (if applicable)	n/a
Was machine dedicated to task (`yes/no`)	Yes (but with minor load from general purpose software and system processes)
Name and version of C++ compiler (if used)	g++ -std=c++11
Name and version of Java compiler (if used)	n/a
Name and version of other non-standard software tools & components (if used)	`callgrind` profiler and `kcachegrind` (GUI viewer)

Benchmark Design

The benchmark generates a vector of size 100 and fills it with random integers. This is only done once to reduce noise in the program execution. Command line arguments determine if the vector will be sorted with an inline function or a lambda (18 million times, to be exact). Calling std::sort on a sorted vector can't be reduced in time anymore so that is not a concern.

Experiments and Observations

Briefly describe how the experiments were conducted (indicate commands used and details on the jobs that were used to conduct the experiments)

Observations:

Record the timing information collated from your experiments conducted using your micro-benchmark in the tables below:

Using inline function	Elapsed Time (sec) measured using `/usr/bin/time` command		
	Opt: `-O0`	**Opt: `-O2`**	**Opt: `-O3`**
Observation #1	23.16	11.47	10.32
Observation #2	22.61	10.66	10.64
Observation #3	23.41	10.67	10.60
Averages (of 3 runs)	**23.06**	**10.93**	**10.52**

Using lambda	Elapsed Time (sec) measured using `/usr/bin/time` command		
	Opt: `-O0`	**Opt: `-O2`**	**Opt: `-O3`**
Observation #1	22.31	4.27	3.4
Observation #2	23.12	4.71	3.38
Observation #3	22.82	4.51	3.38
Averages (of 3 runs)	**23.01**	**4.50**	**3.39**

Results and Conclusions

Using the above data develop a report (about 10 sentences) discussing the performance aspects and conclude if you accept or reject the hypothesis. You may also mention data that you may not have recorded (as in peak memory usage) as needed. Indicate what suggestion you would provide to a programmer based on your experiment.

Note: In this part of the homework the inferences drawn have 50% of points. Consequently, ensure that this discussion is comprehensive. The discussion will be critically graded and only the comprehensive / complete discussions will be assigned full score. See solution for Exercise #4 for example discussion.

Appendix

Copy-paste your style-checked benchmark program source code into the space below:

```
// Copyright 2016 Henry Ni

#include <algorithm>
#include <iostream>
#include <cstdlib>
#include <functional>
#include <vector>
#include <iterator>
#include <climits>
#include <string>

inline bool comp(int& x, int& y) {
  return x < y;
}

int main(int argc, char* argv[]) {
  string cmd = argv[1];
  const int VECTOR_SIZE = 100;
  std::vector<int> v(VECTOR_SIZE);
  std::generate(v.begin(), v.end(), std::rand);
```

```
        if (argc > 1 && cmd == "inline") {
          for (int i = 0; i < 18000000; i++) {
            std::sort(v.begin(), v.end(), comp);
          }
        } else {
          for (int i = 0; i < 18000000; i++) {
            std::sort(v.begin(), v.end(), [](int x, int y){return x < y;});
          }
        }
      return 0;
}
```

Submission

Once you have completed your report upload the following onto Canvas:

1. This report document (duly filled) and saved as a PDF file with the naming convention `MUID_Homework3_PartB.pdf`.
2. The benchmark program that you have developed named with the convention `MUID_Homework3_PartB.cpp`.