CME 213 - Parallel Computing
homework 2

Leon H. Kloker
Institute for Computational and Mathematical Engineering
CME 213, Stanford University
leonkl@stanford.edu

Problem 1

Output from main_q1:

```
 --------
Running main() from ./googletest-main/googletest/src/gtest_main.cc
[==========] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from testQ1
[ RUN ] testQ1.test
Parallel
Sum Even:  757361650
Sum Odd:  742539102
Time:  0.0156226
Serial
Sum Even:  757361650
Sum Odd:  742539102
Time:  0.153143
[ OK ] testQ1.test (381 ms)
[-----] 1 test from testQ1 (381 ms total)


[-----] Global test environment tear-down
[==========] 1 test from 1 test suite ran.  (382 ms total)
[ PASSED ] 1 test.
```

Problem 2: Question 1 - 5

Output from main_q2:

```
--------
Running main() from ./googletest-main/googletest/src/gtest_main.cc
[==========] Running 7 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 7 tests from testQ2
[ RUN ] testQ2.test1
[ OK ] testQ2.test1 (3 ms)
[ RUN ] testQ2.test2
[ OK ] testQ2.test2 (3 ms)
[ RUN ] testQ2.test3
[ OK ] testQ2.test3 (0 ms)
[ RUN ] testQ2.test4
```

```
[ OK ] testQ2.test4 (3 ms)
[ RUN ] testQ2.test5
[ OK ] testQ2.test5 (4 ms)
[ RUN ] testQ2.serialSortTest
Serial Radix Sort:  PASS
stl:  0.3641
serial radix:  0.0894177
[ OK ] testQ2.serialSortTest (646 ms)
[ RUN ] testQ2.parallelSortTest
Parallel Radix Sort:  PASS
stl:  0.363946
parallel radix:  0.0416456
[ OK ] testQ2.parallelSortTest (598 ms)
[-----] 7 tests from testQ2 (1259 ms total)


[-----] Global test environment tear-down
[==========] 7 tests from 1 test suite ran.  (1259 ms total)
[ PASSED ] 7 tests.
```

Problem 2: Question 6

Output from main_q2_part6:

```
--------
Running main() from ./googletest-main/googletest/src/gtest_main.cc
[==========] Running 8 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 8 tests from testQ2
[ RUN ] testQ2.test1
[ OK ] testQ2.test1 (2 ms)
[ RUN ] testQ2.test2
[ OK ] testQ2.test2 (3 ms)
[ RUN ] testQ2.test3
[ OK ] testQ2.test3 (0 ms)
[ RUN ] testQ2.test4
[ OK ] testQ2.test4 (3 ms)
[ RUN ] testQ2.test5
[ OK ] testQ2.test5 (4 ms)
[ RUN ] testQ2.serialSortTest
Serial Radix Sort:  PASS
stl:  0.363891
serial radix:  0.0868982
[ OK ] testQ2.serialSortTest (644 ms)
[ RUN ] testQ2.parallelSortTest
Parallel Radix Sort:  PASS
stl:  0.328215
parallel radix:  0.0266581
[ OK ] testQ2.parallelSortTest (547 ms)
[ RUN ] testQ2.radixSortTest
Threads Blocks / Timing
1     2     4     8     12     16     24     32     40     48
1 0.052 0.052 0.053 0.055 0.057 0.058 0.062 0.065 0.069 0.073
2 0.055 0.042 0.032 0.033 0.038 0.036 0.040 0.041 0.044 0.051
4 0.061 0.031 0.020 0.019 0.024 0.021 0.026 0.027 0.030 0.034
8 0.056 0.031 0.018 0.017 0.019 0.020 0.025 0.026 0.028 0.034
12 0.055 0.033 0.031 0.020 0.022 0.022 0.025 0.027 0.030 0.033
16 0.062 0.032 0.019 0.019 0.017 0.024 0.023 0.025 0.028 0.031
24 0.062 0.041 0.037 0.038 0.034 0.029 0.030 0.033 0.037 0.040
```

```
32 0.077 0.037 0.029 0.020 0.016 0.019 0.020 0.022 0.025 0.028
40 0.058 0.050 0.041 0.023 0.024 0.028 0.028 0.034 0.034 0.040
48 0.077 0.075 0.051 0.040 0.034 0.035 0.037 0.039 0.039 0.039
[ OK ] testQ2.radixSortTest (4615 ms)
[-----] 8 tests from testQ2 (5821 ms total)


[-----] Global test environment tear-down
[==========] 8 tests from 1 test suite ran.  (5821 ms total)
[ PASSED ] 8 tests.
```

The observed runtimes of Radixsort for a given number of threads and blocks on the icme-gpu cluster is listed again in table 1.

We can see that only increasing the number of blocks while running only with 1 thread just increases the overhead slightly and the running time increases. Similarly, increasing solely the amount of threads while having only one block does not really affect the running time as the code with one block is not really parallelizable. The running time is the smallest when both the amount of blocks and the amount of threads are between 8 and 16. As we only have 16 cores available, using more than 16 threads leads to concurrently but not really parallel running threads. Moreover, having more blocks than threads generally speaking only increases the overhead and does not really result in a computation time reduction as visible in table 1.

Threads / Blocks	1	2	4	8	12	16	24	32	40	48
1	0.052	0.052	0.053	0.055	0.057	0.058	0.062	0.065	0.069	0.073
2	0.055	0.042	0.032	0.033	0.038	0.036	0.040	0.041	0.044	0.051
4	0.061	0.031	0.020	0.019	0.024	0.021	0.026	0.027	0.030	0.034
8	0.056	0.031	0.018	0.017	0.019	0.020	0.025	0.026	0.028	0.034
12	0.055	0.033	0.031	0.020	0.022	0.022	0.025	0.027	0.030	0.033
16	0.062	0.032	0.019	0.019	0.017	0.024	0.023	0.025	0.028	0.031
24	0.062	0.041	0.037	0.038	0.034	0.029	0.030	0.033	0.037	0.040
32	0.077	0.037	0.029	0.020	0.016	0.019	0.020	0.022	0.025	0.028
40	0.058	0.050	0.041	0.023	0.024	0.028	0.028	0.034	0.034	0.040
48	0.077	0.075	0.051	0.040	0.034	0.035	0.037	0.039	0.039	0.039

Table 1: Running times