WebCMS2@CSE@UNS

COMP677117s2
Advanced C++
Programming

• **Notices**
• **Calendar**
• **Course Outline**
+ **Administration**
|– Classes
|– Consultations
+ **CourseWork**
|– Lectures
|– Notes
|– Tutorials
|– Assignments
+ **Staff**
|– Jingling Xue
|– Marzieh Jalal Abadi
|– Peter Kydd
|– Joshua Pratt
|– Matthew Stark
|– Donny Yang
|– Henry Zhang
• **MessageBoard**
+ **Resources**
|– C++ Books
|– Extra Readings
|– Videos
|– FAQs
• **WebCMS Info**
|- Change Password?
|- Forgot Password?

Status:
Not Logged In

Username:
z5094935
Password:
••••••••
Login

CRICOS Provider No. 00098G

CS6771 Assignment Five 2017

Parallel Bucket Sort

Due Date: 11:59pm Sunday 29 October 2017

Worth: 10%

Aims:

- To learn how to write multithreaded programs in C++
- To learn how to use mutex locks to avoid race conditions in multiple threads
- To learn about potential performance bottlenecks and performance evaluation of parallel programs

Your task is to take a slow single threaded sort algorithm and redesign it into a parallel radix sort algorithm that sorts a vector of numbers as quickly as possible.

The sorting algorithm takes a vector of unsigned ints and sorts them by using a MSD (Most Significant Digit) radix sort, which uses lexicographic order.

For example, given a vector [32, 53, 3, 134, 643, 3, 5, 12, 52, 501, 98], the sorted output would be [12, 134, 3, 3, 32, 5, 501, 52, 53, 643, 98] (as if a shorter number were conceptually left-

A MSD radix sort can be done easily through a parallel bucket sort. A bucket sort takes the most significant digit of each number and groups the list of numbers with the same digit into one b

```
Bucket 1: [ 134, 12 ]
Bucket 3: [ 32, 3, 3 ]
Bucket 5: [ 53, 5, 52, 501]
Bucket 6: [ 643 ]
Bucket 9: [ 98 ]
```

Afterwards, each bucket is sorted recursively, starting with the next most significant digit:

```
Bucket 1: [ 12, 134 ]
Bucket 3: [ 3, 3, 32 ]
Bucket 5: [ 5, 501, 52, 53]
Bucket 6: [ 643 ]
Bucket 9: [ 98 ]
```

Finally, all the buckets are concatenated together in order: [12, 134, 3, 3, 32, 5, 501, 52, 53, 643, 98]

Requirements:

In this assignment, you will need to create two files: `BucketSort.cpp` and `BucketSort.h.` These files can contain as many classes, functions and structs as you require. However, bec

`BucketSort.h` must contain the following struct (which can be added to):

```
#ifndef BUCKET_SORT_H
#define BUCKET_SORT_H

#include <vector>

struct BucketSort {

        // vector of numbers
        std::vector<unsigned int> numbersToSort;

        void sort(unsigned int numCores);
};


#endif /* BUCKET_SORT_H */
```

The member field `numbersToSort` will be modified by user code to add numbers to be sorted to a BucketSort object. The same member field will then be read from after the sort method l

The sort method will be passed an unsigned int containing the number of CPU cores that are available to be used. You do not need to adhere to this number when creating threads, however, i *system and you will score 0 for that test case.*

Single Threaded Starter Code

The following definition of the sort member function is the reference solution for the correct sort output. Your multithreaded implementation should produce the same output as this function,

```
#include "BucketSort.h"

#include <algorithm>
#include <cmath>

bool aLessB(const unsigned int& x, const unsigned int& y, unsigned int pow) {

        if (x == y) return false; // if the two numbers are the same then one is not less than the other

        unsigned int a = x;
        unsigned int b = y;

        // work out the digit we are currently comparing on.
        if (pow == 0) {
                while (a / 10 > 0) {
                        a = a / 10;
                }
                while (b / 10 > 0) {
                        b = b / 10;
                }
        } else {
                while (a / 10 >= (unsigned int) std::round(std::pow(10,pow))) {
                        a = a / 10;
                }
                while (b / 10 >= (unsigned int) std::round(std::pow(10,pow))) {
                        b = b / 10;
                }
        }

        if (a == b)
                return aLessB(x,y,pow + 1);  // recurse if this digit is the same
        else
                return a < b;
}

// TODO: replace this with a parallel version.
void BucketSort::sort(unsigned int numCores) {
        std::sort(numbersToSort.begin(),numbersToSort.end(), [](const unsigned int& x, const unsigned int& y){
                return aLessB(x,y,0);
        } );
}
```

Sample User Code

Test cases for this assignment will involve user code creating large vectors of random numbers and calling the sort method. Your code will be limited by time and memory with tests increasi

The following sample test case shows how this is done. For the given single threaded code, this sort takes around about 23 seconds to run on the CSE server williams (with 8 cores). In compa

[illegible]

```
#include <iostream>
#include <random>
#include <thread>

#include "BucketSort.h"

int main() {

        unsigned int totalNumbers =     500000;
        unsigned int printIndex =       259000;

        // use totalNumbers required as the seed for the random
        // number generator.
        std::mt19937 mt(totalNumbers);
        std::uniform_int_distribution<unsigned int> dist(1, std::numeric_limits<unsigned int>::max());

        // create a sort object
        BucketSort pbs;

        // insert random numbers into the sort object
        for (unsigned int i=0; i < totalNumbers; ++i) {
                pbs.numbersToSort.push_back(dist(mt));
        }

        // call sort giving the number of cores available.
        const unsigned int numCores = std::thread::hardware_concurrency();
        pbs.sort(numCores);
```