CS 460 Assignment #4 Threading Basics

Date Assigned: March 17, 2014
Date Due: April 7, 2014
Points: 50

Goal:
Learn about basic threading without synchronization issues, use concepts from Data Structures, see what kind of speedup can be obtained using multiple threads.

Description:

Consider the following bubble sort:

```
void bubbleSort (int *pArray, int count)
{
  int bExchange = TRUE;
  int indx;
  int temp;
  while (bExchange)
  {
    bExchange = FALSE;

    for (indx = 0; indx < count - 1; indx++)
    {
      if (*(pArray + indx) > *(pArray + indx + 1))
      {
        temp = *(pArray + indx);
        *(pArray + indx) = *(pArray + indx + 1);
        *(pArray + indx + 1) = temp;
        bExchange = TRUE;
      }
    }
  }
 }
```

We know from Data Structures that this program has a time complexity of O(n^2) and a space complexity of DSPACE(bubbleSort) = O(n), i.e. for the deterministic bubbleSort computation, the space is the size of the array, n. While the bubble sort is easy to understand and code, it suffers speedwise in that faster sorting algorithms execute in nlogn time.

So how can we increase the speed of execution of a problem that uses bubble sort. Yup!!! Threads.

Your Program:

You are to write a multithreaded program that sorts up to 100,000 integer values in a global array gArrayValues. The max number of values is to be set in a variable NUM_VALUES that is #defined. Your program is to be executed from the command line as follows:

`./CS460_SimpleThreads 100 2 1` where the first argument is the executable, followed by the number of random values to sort (100 in this case), a seed value (2 in this case) and 1 is the number of threads used to sort the array of values. Fill the array with the specified number of random values before sorting the array of values. We will all be sorting the same exact values!!!!

As output, you are to print one of the following screens based on the number of sorting threads:

Screen #1

```
*** Sorting With Threads ***

Number Of Sorting Threads: 1
Values Sorted: 10
Seed Value: 2

Sorted Values

 142559277   190686788   260874575   747983061   906156498
1261608745  1380759627  1502820864  1505335290  1738766719

Clock Ticks: XXXXXX

SORTING COMPLETE.
```

Screen #2

```
*** Sorting With Threads ***

Number Of Sorting Threads: 2
Values Sorted: 100000
Seed Value: 2

Elapsed Time (Seconds): 1
Elapsed Time (Seconds): 2
…

Sorted Values

     28866       36416       61026       79486      108359
    111543      111856      116746      129286      156142
         …

Clock Ticks: XXXXXX

SORTING COMPLETE.
```

Notes:

1. The difference between Screen #1 and #2 is that not only are there 2 sorting threads, there is a third thread that is printing the elapsed time to the screen in 1 second intervals. It is important when writing multithreaded applications that the main UI screen is always responsive to the user. The time counter will simulate this.
2. You must use the bubbleSort routine for sorting and no other sorting routine.
3. You cannot add any additional array space. This includes not using any recursive code that uses runtime stack space to create additional space.
4. You will need to modify bubbleSort so that it can be properly passed to pthread_create. Do not modify the way bubbleSort works or you will lose significant points. If you have questions, ask. Better to be safe than sorry.
5. Create your own make file and make sure to run valgrind on your executable.

Executable Location

You must build the executable (CS460_MultithreadedSorting) at the root of your Eclipse Project.

Submitting

On the day your assignment is due, you are to submit a zipped up tar file on zeus called CS460_MultithreadedSorting_PUNetID.tar.gz

```
zeus$ submit cs460s14 CS460_MultithreadedSorting_PUNetID.tar.gz
```

Program Design

Don’t just start writing code as your end result will be spagetti code. Design before coding writing well defined functions to do one major task. If your design is poor, you will lose significant point

Extra Credit

If you figure out a way to use 4 or more threads (remember each thread needs to do the same amount of work) for sorting, you will receive 5 points of extra credit. If you go to 4 or more threads, you must use an array of threads. If you have the fastest sorting solution, you will receive 5 points of extra credit. Please get the sorting working with 2 sorting threads before trying the 4 or more threads version. Use subversion often. I will test your solution on zeus and if you go for the extra credit, I will test your code on my machine which is 8 cores. Ummmm. Eight cores … that's an interesting number!!! ☺