Operating Systems CSC 415 HW #8

December, 16, 2013

Kenneth Robertson

Files:

-OSHW8linux.c

-OSHW8win32.c

Description:

This program uses an insertion sort function that sorts only part of an array. The program then creates many threads that call this function to sort different sections of the array. The result is that each partition is sorted. Once the threads are done executing the main thread can then run a function that uses these partitions as a list of potential minimum values which it scans to find the smallest.

This program takes at least two command line arguments (I added an, optional, third so that the printing of the array could be turned off for large arrays). The first argument P is an integer used to determine the size of an array of random integers (the array will be of size 2^P). The second argument N is used to determine both the number of threads and the number of partitions that he array of random integers is split into (this value will be 2^N). These arrays are allocated, and the random integer array is given random values. Next an array if integers representing the first index into each of the partitions is created of size 2^N this will represent the minimum index after each partition is sorted, so its called minIndices) An array of thread handles is also created, also of size 2^N. I created a struct called ThreadInput containing startIndex, endIndex, and threadNumber; this ThreadInput is used to pass information to the thread function. It was necessary to create an array of 2^N of these, one for each thread. I found it necessary to create an array bools of size 2^N that indicates if a partition has been completely exhausted of all of its values so that it will be ignored later on in the program.

Next, the initial contents of the array are printed (I added a command line option to turn this off for large arrays, so that the user doesn't have to wait for a huge array to print). A time-stamp is taken and stored, and then the loop to generate the threads is started. Inside the loop before the threads are created the correct ThreadInfo

data for that thread is stored. The main thread then waits for the threads to complete. Once they have completed another time-stamp is taken and sorted. The result of the difference of the initial and final time stamp is printed so the user can see how long it took for the threads to complete there sorting. The thread function is essentially just a insertion sort with some slight alteration to make sure that it doesn't alter any values before or after its partition.

Next a loop is created that cycles through the length of random integer array each time printing out the position and the return value for get_next() (I added a command line option to turn this off for large arrays, so that the user doesn't have to wait for a huge array to print). get_next() Works by initially finding the first value in the min_indexes that does not exceed the bounds of its partition. That value becomes the initial minimum. A for loop cycles through the remaining partitions looking for dataArray[minIndices[i]] that is less than the current minimum and that the partition for minIndices[i] has not been exhausted (this is checked by testing validRanges[i]). If this is true then dataArray[minIndices[i]] becomes the new presumptive minimum. Finally the minIndices[i] is incremented, if minIndices[i] after incrementation is past its partition size then validRanges[i] is set to 0; Then minimum value is then returned.

The last thing the program does it is time how long it takes for the main thread alone to complete the insertion sort and then print out that time.

System Information:

Results were produced on a Intel i7-3820 @ 3.60 GHz; This has 4 actual cores each with 2 threads.

OS and Compilation Information:

The windows version was compiled using Microsoft Visual Studio 2010. The linux version was compiled using gcc on ubuntu 12.04.3 i386 VirtualBox using console command: cc OSHW8.c -o OSHW8 -pthread -lrt

Analysis:

The advantage of multiple threads didn't become evident until the size the the array got to around 2^{16}. The Windows version outperform the the linux version by far amount, however I wonder if that might not have something to do with the fact the the linux OS was running in virtual box not as the native OS.

OS	65536 ints, 8 threads	65536 ints, 4 threads	Relative Speed
Windows	114241 microS	316061. microS	Win32 4.5times
ubuntu 12.04.3 i386 VirtualBox	504934. microS	1089226. microS	Win32 3.5times

Win32 Output:

```
C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 4 1

Initial Array

dataArray[0] = 48
dataArray[1] = 7196
dataArray[2] = 9294
dataArray[3] = 9091
dataArray[4] = 7031
dataArray[5] = 23577
dataArray[6] = 17702
dataArray[7] = 23503
dataArray[8] = 27217
dataArray[9] = 12168
dataArray[10] = 5409
dataArray[11] = 28233
dataArray[12] = 2023
dataArray[13] = 17152
dataArray[14] = 21578
dataArray[15] = 2399


SORTED IN PARALLEL

Position 0 = 48
Position 1 = 2023
Position 2 = 2399
Position 3 = 5409
Position 4 = 7031
Position 5 = 7196
Position 6 = 9091
Position 7 = 9294
Position 8 = 12168
```

```
Position 9 = 17152
Position 10 = 17702
Position 11 = 21578
Position 12 = 23503
Position 13 = 23577
Position 14 = 27217
Position 15 = 28233

It took 292.965611 microseconds for 2 threads to be created and complete there searches.

It took 0.853298 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 5 2
Initial Array

dataArray[0] = 48
dataArray[1] = 7196
dataArray[2] = 9294
dataArray[3] = 9091
dataArray[4] = 7031
dataArray[5] = 23577
dataArray[6] = 17702
dataArray[7] = 23503
dataArray[8] = 27217
dataArray[9] = 12168
dataArray[10] = 5409
dataArray[11] = 28233
dataArray[12] = 2023
dataArray[13] = 17152
dataArray[14] = 21578
dataArray[15] = 2399
dataArray[16] = 23863
dataArray[17] = 16025
dataArray[18] = 8489
dataArray[19] = 19718
dataArray[20] = 22454
dataArray[21] = 12798
dataArray[22] = 1164
dataArray[23] = 14182
dataArray[24] = 29498
dataArray[25] = 1731
dataArray[26] = 27271
dataArray[27] = 18899
dataArray[28] = 6936
dataArray[29] = 27897
dataArray[30] = 11449
dataArray[31] = 31232


SORTED IN PARALLEL

Position 0 = 48
Position 1 = 1164
Position 2 = 1731
Position 3 = 2023
Position 4 = 2399
Position 5 = 5409
Position 6 = 6936
Position 7 = 7031
Position 8 = 7196
```

```
Position 9 = 8489
Position 10 = 9091
Position 11 = 9294
Position 12 = 11449
Position 13 = 12168
Position 14 = 12798
Position 15 = 14182
Position 16 = 16025
Position 17 = 17152
Position 18 = 17702
Position 19 = 18899
Position 20 = 19718
Position 21 = 21578
Position 22 = 22454
Position 23 = 23503
Position 24 = 23577
Position 25 = 23863
Position 26 = 27217
Position 27 = 27271
Position 28 = 27897
Position 29 = 28233
Position 30 = 29498
Position 31 = 31232

It took 308.324973 microseconds for 4 threads to be created and complete there searches.

It took 2.559894 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 8 1 -

It took 197.111814 microseconds for 2 threads to be created and complete there searches.

It took 69.685995 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 8 2 -

It took 229.252702 microseconds for 4 threads to be created and complete there searches.

It took 70.254860 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 8 3 -

It took 374.597777 microseconds for 8 threads to be created and complete there searches.

It took 71.677023 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 16 1 -

It took 1006571.531536 microseconds for 2 threads to be created and complete there searches.

It took 3983588.237118 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 16 2 -

It took 316061.256549 microseconds for 4 threads to be created and complete there searches.

It took 3992256.321586 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 16 3 -
```

```
It took 114240.944589 microseconds for 8 threads to be created and complete there searches.

It took 4115843.438040 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>OSHW8.exe 20 3 -

It took 28737256.778101 microseconds for 8 threads to be created and complete there searches.

It took 1021305312.547376 microseconds to perform the insertion sort using the main thread.

C:\Users\Kenneth\Documents\Visual Studio 2010\Projects\OSHW8\Debug>
```

Posix Output:

```
adjuser@adminuser-VirtualBox:~$ ./OSHW8 4 1

Initial Array

dataArray[0] = 1205554746
dataArray[1] = 483147985
dataArray[2] = 844158168
dataArray[3] = 953350440
dataArray[4] = 612121425
dataArray[5] = 310914940
dataArray[6] = 1210224072
dataArray[7] = 1856883376
dataArray[8] = 1922860801
dataArray[9] = 495649264
dataArray[10] = 8614858
dataArray[11] = 989089924
dataArray[12] = 378651393
dataArray[13] = 1344681739
dataArray[14] = 2029100602
dataArray[15] = 1816952841


SORTED IN PARALLEL

Position 0 = 8614858
Position 1 = 310914940
Position 2 = 378651393
Position 3 = 483147985
Position 4 = 495649264
Position 5 = 612121425
Position 6 = 844158168
Position 7 = 953350440
Position 8 = 989089924
Position 9 = 1205554746
Position 10 = 1210224072
Position 11 = 1344681739
Position 12 = 1816952841
Position 13 = 1856883376
Position 14 = 1922860801
Position 15 = 2029100602

It took 0 s, 173884 ns for 2 threads to be created and complete their searches.
```

```
It took 0 s, 921 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 5 2
Initial Array

dataArray[0] = 1205554746
dataArray[1] = 483147985
dataArray[2] = 844158168
dataArray[3] = 953350440
dataArray[4] = 612121425
dataArray[5] = 310914940
dataArray[6] = 1210224072
dataArray[7] = 1856883376
dataArray[8] = 1922860801
dataArray[9] = 495649264
dataArray[10] = 8614858
dataArray[11] = 989089924
dataArray[12] = 378651393
dataArray[13] = 1344681739
dataArray[14] = 2029100602
dataArray[15] = 1816952841
dataArray[16] = 21468264
dataArray[17] = 552076975
dataArray[18] = 87517201
dataArray[19] = 953369895
dataArray[20] = 374612515
dataArray[21] = 787097142
dataArray[22] = 126313438
dataArray[23] = 1207815258
dataArray[24] = 287632273
dataArray[25] = 1886964647
dataArray[26] = 1220723885
dataArray[27] = 1119448937
dataArray[28] = 444268468
dataArray[29] = 1865680798
dataArray[30] = 1654563454
dataArray[31] = 1649823214


SORTED IN PARALLEL

Position 0 = 8614858
Position 1 = 21468264
Position 2 = 87517201
Position 3 = 126313438
Position 4 = 287632273
Position 5 = 310914940
Position 6 = 374612515
Position 7 = 378651393
Position 8 = 444268468
Position 9 = 483147985
Position 10 = 495649264
Position 11 = 552076975
Position 12 = 612121425
Position 13 = 787097142
Position 14 = 844158168
Position 15 = 953350440
Position 16 = 953369895
Position 17 = 989089924
Position 18 = 1119448937
Position 19 = 1205554746
```

```
Position 20 = 1207815258
Position 21 = 1210224072
Position 22 = 1220723885
Position 23 = 1344681739
Position 24 = 1649823214
Position 25 = 1654563454
Position 26 = 1816952841
Position 27 = 1856883376
Position 28 = 1865680798
Position 29 = 1886964647
Position 30 = 1922860801
Position 31 = 2029100602

It took 0 s, 235003 ns for 4 threads to be created and complete their searches.

It took 0 s, 1702 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 8 1 -

It took 0 s, 262249 ns for 2 threads to be created and complete their searches.

It took 0 s, 84769 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 8 2 -

It took 0 s, 299739 ns for 4 threads to be created and complete their searches.

It took 0 s, 92517 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 8 3 -

It took 0 s, 363619 ns for 8 threads to be created and complete their searches.

It took 0 s, 83620 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 16 1 -

It took 2 s, 122910975 ns for 2 threads to be created and complete their searches.

It took 7 s, 479273883 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 16 2 -

It took 1 s, 89226404 ns for 4 threads to be created and complete their searches.

It took 7 s, 471482994 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$ ./OSHW8 16 3 -

It took 0 s, 504934273 ns for 8 threads to be created and complete their searches.

It took 7 s, 606902426 ns to perform the insertion sort using the main thread.
adminuser@adminuser-VirtualBox:~$
```