Uppsala University
Department of Information Technology
Scientific Computing

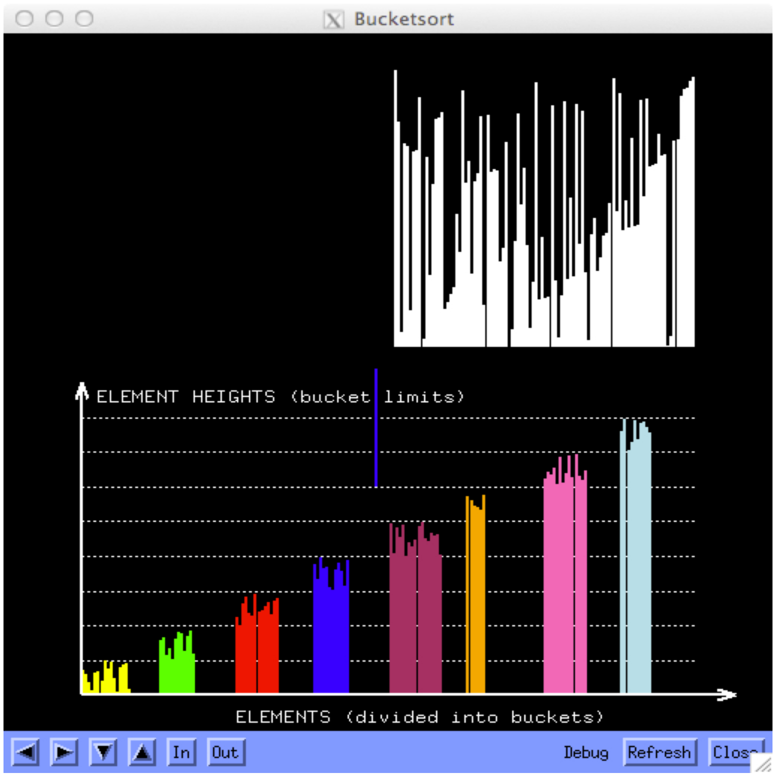
Programming of Parallel Computers
2015-01-13

Compulsory Assignment 3

Shared memory parallelization using OpenMP

1 The Bucket-sort Algorithm

In this task you will implement the Bucket-sort algorithm, here the elements are first filtered into buckets (i.e. elements within range $[a_i\ b_i]$ belong to bucket i). Then the buckets can be sorted concurrently and independently in different threads, e.g., using the quicksort algorithm. See Figure 1 below.


Bucketsort
ELEMENT HEIGHTS (bucket limits)
ELEMENTS (divided into buckets)
In
Out
Debug
Refresh
Close

Figur 1: The Bucket-sort Algorithm

The algorithm is straightforward to parallelize as the sorting of the buckets is trivially parallel. The problem becomes to get a good load balance among the threads. Here, we will investigate different strategies to load balance the computations in OpenMP as described below.

2 Data generation

We will investigate two different random number sequences, one uniform distribution and one normal distribution. For the uniform distribution you can use the drand48() function in C and for the generation of a normal distribution use the Box-Muller method described in Wikipedia
`http://en.wikipedia.org/wiki/Box_Muller_transform`

3 Load balancing strategies

Create uniform buckets in the range min to max (with exception for normal distribution where the last and the first bucket can be infinite to cover in all outliers, i.e, where the first bucket covers all elements within $[-\infty \text{ min}]$ and the last bucket $[\text{max } \infty]$). For the load balancing of the buckets to the threads we will investigate the following strategies, in the strategies 1-3 you create more buckets than threads and match the number of threads to the available cores:

1. **Pragma schedule:** Assign the buckets to the threads in a for-loop. Investigate different options for load balancing by using the schedule directive.

2. **Pragma task:** Put the buckets into a task queue by using the task directive and let OpenMP handle the load balance of assigning tasks to threads.

3. **Bin-pack:** Load balance the computations explicitly by using the Bin-pack algorithm (see lecture notes), i.e., use the Bin-packing algorithm to explicitly assign buckets to threads such that the work load in each thread will be as equal as possible.

4. **Run time system:** Create as many threads as buckets and assign one bucket per thread but use more threads than available cores. Let the run time system handle the work load on the computer system by time sharing of threads.

5. **Nested parallelism (challenge, optional):** Create only a small number of buckets and assign one bucket per thread, then within a thread/bucket create new threads proportional to the relative bucket size and parallelize the sorting algorithm of the elements in the bucket.

4 Experiments

We have an open research like question of how to use load balancing in OpenMP in an efficient way and how the different strategies compare to each other. Therefore it is up to you to design your experiments in a way such that the results and conclusions will be convincing. You can vary the data sequences, number of threads, number of buckets, computer system and algorithms.

5 Coaching session

You are welcome anytime to ask about the assignment but we will also like to meet all groups in coaching sessions. The intention is to see that you are on the right track and to correct any misunderstandings in an early stage. Please, contact us for booking a time for the coaching session. Before the coaching session you should have started the implementation of the assignment but there is no requirement that the program can run (or even compile correctly) but the more you have done the more help we can give.

6 Writing a report on the results

Write a report as you would submit this as paper to a conference in parallel computing or to a scientific journal within the subject, i.e., the report should contain an abstract, introduction, theory, experiments, conclusions and references. Limit the paper to 4-8 pages. The quality of the paper should be good enough, a minimum requirement is that you would not be a shamed of it if it would come to print for a larger audience. Include your source files in an appendix in the report (in addition to the 4-8 pages). Submit also the source codes as C-files so that we can compile and run them if necessary.

The last day to hand in the report is **March 6, 2015** but try to do it as soon as possible.