Uppsala University
Department of Information Technology
Scientific Computing

Programming of Parallel Computers
2015-01-13

Compulsory Assignment 2

Shared memory parallelization using Pthreads

1 Quick-Sort

The serial Quick-sort algorithm is described numerously in the literature and briefly the algorithm follows as:

1. Select a pivot element.
2. Divide the data into two sets according to the pivot (smaller and larger).
3. Sort each list with Quick-sort recursively.

Start by implementing a serial version of this algorithm following the notes, e.g., in Wikipedia [**http://en.wikipedia.org/wiki/Quicksort**] using the in-place version. The algorithm can then be parallelized in many ways but we will only study and implement two (or three optionally) inherently different ways to do this, a *divide-and-conquer algorithm* and a *peer algorithm*.

1.1 Divide and conquer parallelization

A straightforward parallelization of Quick-Sort is to acquire a new thread for each recursion step and sort the two lists in parallel on the two threads. One problem is then to not acquire to many threads (or too few threads). After a number of recursion levels one can proceed without requiring new threads using the serial Quick-sort algorithm on the respective threads. How does the number of threads affect the performance and in what way? What are the performance obstacles?

1.2 Peer parallelization

Create p threads as peers, divide the data equally assigning n/p elements to each thread, and sort the elements internally within each thread using the serial Quick-Sort algorithm. Then, in p phases alternating with your left and your right neighbor, merge data and keep either the right or the left part of the data. Compare *parallel odd-even-sort* algorithm 9.4 in the textbook or in the lecture notes. How does the number of threads affect the performance and in what way? What are the performance obstacles?

1.3 Challenge (optional): Parallel Quick-Sort

In the lectures we have talked about a distributed memory parallelization of Quick-Sort that is suitable for a MPI-implementation. The algorithm follows as:

1. Divide the data into p equal parts
2. Sort the data locally in each processor
3. Perform global sort
   3.1 Select pivot in each processor set
   3.2 In each processor, divide the data
   into two sets (smaller or larger)
   3.3 Split the processors into two groups
   and exchange data pair-wise
   3.4 Merge data into a sorted list in each processor
4. Repeat 3.1-3.4 recursively for each processor group

Implementing this algorithm in Pthreads will in some sense be simpler as we have a global address space but in another sense harder as we do not have the support of communicators grouping the threads together. As a challenge, this part is optional and not required, implement the algorithm in Pthreads.

2 Experiments

Sort double precision random number sequences up to 100 000 000 elements and measure the speedup using your parallel implementation. To get a good random number sequence use the function drand48(). Measure speedup in your implementations, compare the algorithms, and analyze/explain your results discussing the advantages and disadvantages of the respective algorithm.

3 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.

4 Writing a report on the results

No formal report is required but present your experiments with relevant figures, tables, reflections and explaining text, i.e., submit a short informal report including your source codes in **one PDF file**. 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 **February 24, 2015** but try to do it as soon as possible.