Compulsory Assignment
The Parallel Quicksort algorithm

Course ' *Parallel and Distributed Programming*'
Division of Scientific Computing, Department of Information Technology
Uppsala University

1 Problem setting

Given a sequence of n real numbers, the task is to implement the Parallel Quicksort algorithm using C and MPI, and evaluate the performance.

2 Outline of the algorithm

Algorithm 2.1 The Parallel Quick-sort algorithm

1 Divide and distribute the data into p equal parts, one per process
2 Sort the data locally for each process
3 Perform global sort
  3.1 Select pivot element within each process set
  3.2 Locally in each process, divide the data into two sets according to the pivot (smaller or larger)
  3.3 Split the processes into two groups and exchange data pairwise between them so that all processes in one group get data less than the pivot and the others get data larger than the pivot.
  3.4 Merge the two sets of numbers in each process into one sorted list
  Repeat 3.1 - 3.4 recursively until each group consists of one single process.
4 Collect data into one sorted array in root processor.

The algorithm converges in $\log_2 p$ steps, see Figure 1.

2.1 Pivot strategies

You are expected to implement the following pivot strategies:

1. Select the median in one processor in each group of processors.
2. Select the median of all medians in each processor group.
3. Select the mean value of all medians in each processor group.

You are free to test other pivot selecting techniques as well if you like.

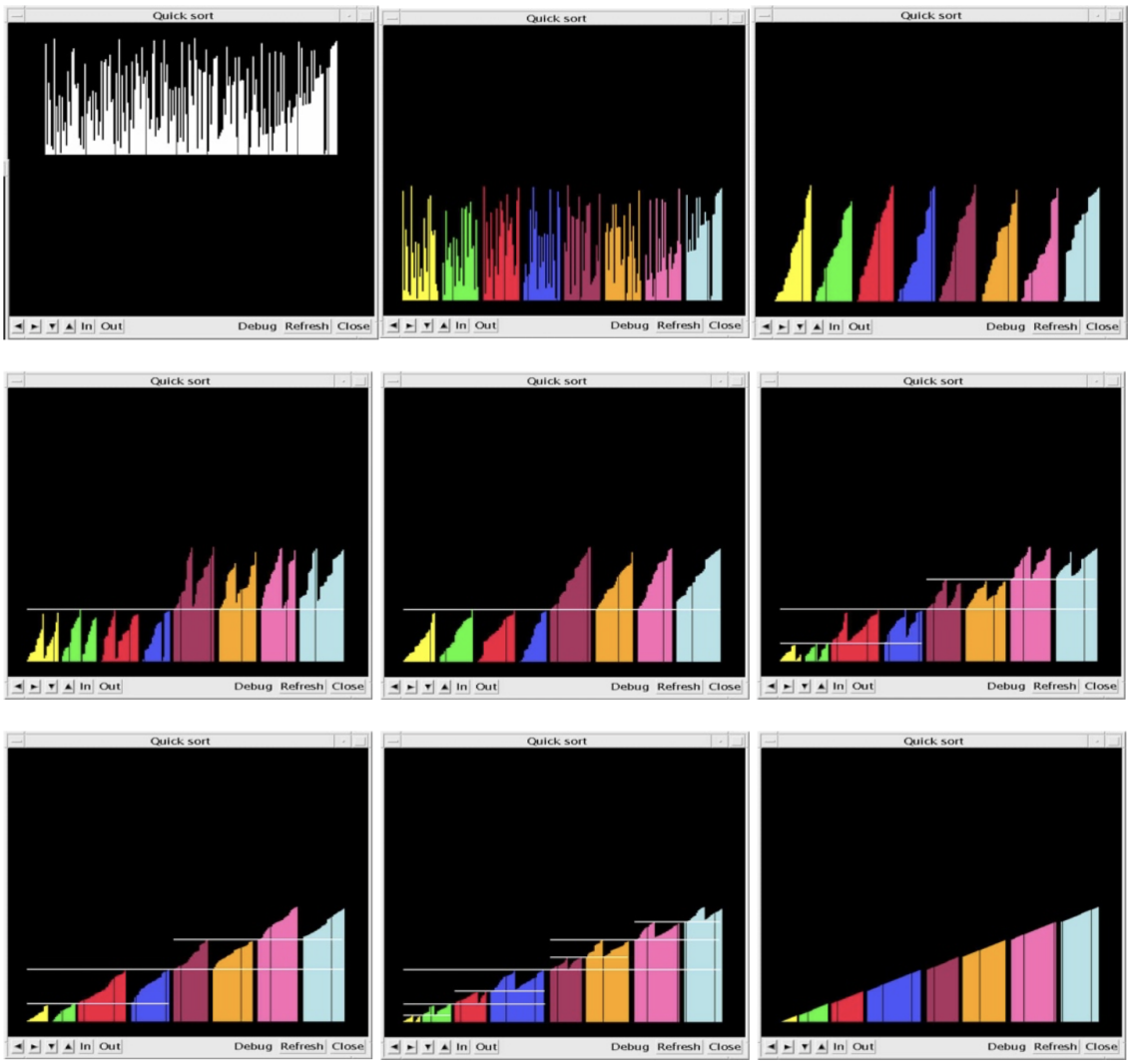

Figure 1: Visualization of Parallel Quick Sort, bars of different heights are sorted on eight processors. Colors represent processor owner ship and the horizontal lines are pivot elements. On the first row bars are distributed to the processors and sorted locally. Then follows a step where the processors are divided into two groups and a pivot is chosen. Data is then exchanged between the processor sets according to the pivot and merged into one sorted sequence in each processor. This is repeated recursively in each processor group until the bars are sorted globally.

3 Implementation

Your program is supposed to take three arguments. The first argument is to specify the random number sequence. The second argument is the length of the sequence. The third argument is a number specifying which pivot strategy to use, according to the list in Section 2.1. Your implementation must be independent of n. Assume that the number of processes is equal to 2^k for some non-negative integer k. You can use the functions provided in `quicksort.c` as a starting point for your implementation. Note that your code should be reasonably well structured, efficient and documented.

4 Numerical experiments

You are required to run the numerical experiments on UPPMAX but you are recommended to develop and test the code first on your computer or the Linux/Unix servers. Vary the number of processing elements and compute the corresponding speedup. Perform a number of experiments to demonstrate strong and weak scalability of your implementation. Run your program on different random number sequences and with different sizes. Also, evaluate the performance for a sequence of numbers sorted in descending order. Compare the different pivot strategies for the different types of sequences.

5 Report

You are supposed to write a report on your results. The report can be written in Swedish or English, and should contain (at least) the following parts:

1. A brief description of the theoretical problem and your implementation.
2. A description of your numerical experiments.
3. The results from the numerical experiments. Execution times for different random number sequences, different problem sizes, different pivot selection strategies and different number of processes should be presented in tables. Speedups values should be presented both in a table and a plot. Also plot the ideal speedup in the same figure. The plots can be done using MATLAB, or any other tool that you are familiar with.
4. A discussion of the results. Do they follow your expectations? Why/why not?

6 Files to be submitted

Upload the following file to Studium:

- A PDF file named A3_Report.pdf containing you report.
- Your code, following the specifications listed in Section 3.
- A Makefile for compiling your code. The code should compile both on the Unix/Linux system and on UPPMAX without warnings.

The assignment should be submitted no later than **May 17:th** and, if necessary, revised no later than **June 10:th**.

7 Precaution

Use the UPPMAX resources with care! We are given 2000 core/hours for the whole course. Any intentional misuse or usage outside the course tasks will result in suspension of your account.

Happy hacking!
Maya & Jarmo

Any comments on the assignment will be highly appreciated and will be considered for further improvements. Thank you!