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 integers, the task is to implement the Parallel Quicksort algorithm using C and MPI, and evaluate the performance. You are encouraged to work in pairs. However, all topics in the assignment should be covered by each student.

2 Outline of the algorithm

Algorithm 2.1 The Parallel Quick-sort algorithm

1 Divide 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
4 Repeat 3.1 - 3.4 recursively for each half until each group consists of one single process.

The algorithm converges in $\log_2 p$ steps.

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.

3 Implementation

Your program is supposed to take three arguments. The first argument is the name of a file containing data to be sorted (the input file). The second argument is the name of the file to which the program will write the sequence of sorted numbers (the output file, in ascii format, as the input file). The third argument is a number in the range $1 - 3$ specifying which pivot strategy to use, according to the list in Section 2.1. If you choose to implement other pivot strategies, you may of course extend the range.
The input file will contain $n + 1$ numbers. The first number is n (that is, the number of elements to sort) and the n subsequent numbers are the actual number sequence. The output file shall contain these numbers, sorted, and nothing else. The numbers should be separated by white space. This holds also for the input file.
Your implementation must be independent of n. Assume that the number of processes is equal to 2^k for some non-negative integer k.
For Step 2 in Algorithm 2.1 you may use the C function `qsort`. You are supposed to sort the numbers in ascending order.
Measure the time for sorting in seconds, not including file reading and writing. The number of seconds (and nothing else) shall be written to stdout when the sorting is done.
Note that your code should be reasonably well structured and documented.

4 Numerical experiments

It is recommended to run the numerical experiments on UPPMAX. 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 sequences of n random numbers with different values of n. Also, evaluate the performance for a sequence of numbers sorted in descending order. Compare the different pivot strategies for both types of sequences.
When planning: You may use the files stored in the directory `/proj/uppmax2023-2-13/nobackup/qsort_in` at UPPMAX as input data. Larger files are intended for numerical experiments, while smaller files can be used for testing of your program.
Files whose name starts with 'input' contain sequences of random numbers and files whose name start with 'backwards' contain elements sorted in *descending* order. The number in the file name is the size of (i.e. the number of elements in) the sequence contained in the file. All files follow the format described in Section 3.
Note that you have a limited disc quota at UPPMAX! Therefore, it is recommended that you don't copy the larger input files to your home directory, but read them from the project directory (where they are stored now). Moreover, it is a good idea to remove large output files as soon as possible. You may also skip the printing to files during the numerical experiments. However, note that the code that you hand in shall follow the specifications above.

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 problem sizes (n) and different number of CPU:s/cores should be presented in tables. Speedup values should be presented both in a table and in 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 Self testing

You will be provided some scripts to self-test the correctness of your code. The details are included in a separate instruction file, named `Self_test_A3.pdf`.

7 File to be submitted

The file that you upload in Studentportalen must be a compressed tar file named A3.tar.gz. It shall contain a directory named A3, with the following files inside:

- A PDF file named A3_Report.pdf containing you report.
- Your code, following the specifications listed in Section 3.
- A Makefile that does the following.
    1. Builds a binary named `quicksort` from your code when the command `make` is invoked. This must work on the Linux system as well as on UPPMAX!
    2. Removes all binary files and object files when the command `make clean` is invoked.

Since your code will be tested automatically, it is very important that you follow the instructions regarding the implementation and the structure of the tar file! Failure to do so will result in immediate rejection of the assignment.

8 Deadline

The assignment should be submitted and, if necessary, revised according to the deadlines given at the student portal. Assignments submitted after these deadlines will not be considered.

9 Precaution

Use the UPPMAX resources with care! We are given 2000 core/hours for the whole course.

Happy hacking!

Maya, Marina and Tobias

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