19. Lab Distributed Algorithms

Distributed Computing involves the breaking down a computational problem into several parallel tasks to be completed by two or more computers in a network which form a distributed system. This combines the computational power of several computers to solve large problems which involve the processing of large data or require a huge number of iterations.
In this lab, we will experiment with a classical problem, sorting. Sorting can be easily parallelized using Mergesort[1].

Speedup in Distributed Computing

Amdahls law[2] provides a way to estimate the speedup of running a program on several machines. It requires a single parameter p, which describes the fraction of the code that can be parallelized. Given n machines, the speedup S compared with running the program on a single machine is calculated as:

$$S = \frac{1}{1-p+\frac{p}{n}}$$

Example: Consider a program of which 90% can be parallelized (p=0.9). Then the speedup of running the program on 3 machines is $\frac{1}{1-0.9+\frac{0.9}{3}} = 2.5$.

Distributed Mergesort

Download the distributed Mergesort code from the course website. The servers are in this case machines that provide sort functionality. The client is the one that uses the provided sort services, by giving each provider a subsegment of the whole array to be sorted.

[1] https://en.wikipedia.org/wiki/Merge_sort
[2] https://en.wikipedia.org/wiki/Amdahl's_law

Tasks

1. Estimate the speedup of the distributed Mergesort code when distributing the sorting of 100, 1000, 10000, 50000, 100000, 500000 numbers using 2 Raspberries. ***Hint****: To determine the fraction of the Mergesort that can be parallelized, remember that sorting has complexity O(n*log n), and that Mergesort contains a centralized merge step for the individual results, which runs in O(n) time.*

2. Measure the runtime of the local Mergesort and the distributed Mergesort using the problem sizes described above.

3. Plot the runtimes on the blackboard.

4. Try to explain any differences to your theoretical calculation from (1).

5. Team up with another group in order to run Mergesort distributed over 3 and 4 machines. Again, take note of the runtimes and compare them with the theoretically possible speedup.

Note

To successfully run the code on the RPIs, remember to

- Run rmiregistry from the same directory as the code
- Create a static mapping for *raspberrypi* to its own IP in */etc/hosts*

Further Reading

- A Comparison of Parallel Sorting Algorithms on Different Architectures (http://dl.acm.org/citation.cfm?id=892860)