Lab D: Map-reduce in Erlang

DAT280 - Parallel Functional Programming (Group 17)

Theodor Åstrand – theast@student.chalmers.se – 931109-9114
Rafael Mohlin – mrafael@student.chalmers.se – 931106-0017

Chalmers University of Technology

May 16, 2016

All the performance measurements were performed on a Macbook Pro Late 2011 with three different nodes. One of the nodes were acting as a controller while the rest was workers. In total we were searching 238mb of data.

Distributed method

In Table 1 we can see the different times we got from running the parallel version versus the distributed version of map_reduce.

Method	**Mean (s)**
Parallel	244.8361
Distributed	234,4436

Table 1: Times from running different versions of map_reduce.

Distributed load-balancing method

In Table 2 we can see the different times that we got from running different versions of map_reduce. As we can see in Table 2 there is a small difference in time between the two different methods. As you can see in Table 1 and Table 2 the difference in times between the two different distributed methods is negligible.

Method	**Mean (s)**
Parallel	244.8361
Distributed load-balanced	235.9554

Table 2: Times from running different versions of map_reduce.

From this exercise we can conclude that Erlang is really good at distributing work between different nodes. We also tried the "Distributed load-balancing method" with nodes on two different computers. In this configuration the first computer was a Dell Inspiron 13 7000-series with an Intel i7-6500U @ 2.50Ghz, and the second computer was a "server" with an underclocked AMD A10-5800K @ 1.90Ghz. Half of the time this configuration was faster and the other half it was slower, due to the difference in computing power between the two computers; the total time is dependant on which computer got the hardest job. This is the problem with a naive load-balancer that was discussed in one of the lectures.