INF3203 - Assignment 1

You will in this assignment experiment with the Map-Reduce programming model, and evaluate the scalability and properties of a map reduce framework implementation.

The foundations of the Map-Reduce programming model were defined in the paper *MapReduce: Simplified Data Processing on Large Clusters* from 2004. Here, the bulk parallel processing of large datasets is expressed as two functions following map and reduce semantics: *map* takes one input data point, potentially transforms it, and assigns it a key; *reduce* finds all data points with the same key and performs a calculation on the entire set, reducing the vector of values to a single entry, or a smaller vector.

The assignment pre-code contains an implementation of a distributed word count application. Your task is to extend this to implement the simplified PageRank algorithm in the same framework, and perform experiments to evaluate it.

This is the first of two assignments in INF-3203. Both assignments must be completed in order to qualify for the oral exam. This comes in addition to the mandatory presentation of a scientific paper.

PageRank

PageRank is a ranking algorithm developed by Google to rank the importance of a website, given the set of websites that link to it.

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

Given a site *u*, and the set *B* containing websites that link to it (*v*), the PageRank of a site *u* is equal to the sum of the PageRank of sites that link to it (*v*) divided by the number of total links that site has (*L(v)*). Initially, all sites has a PageRank of 1.

This is the simplified PageRank - the full version includes a dampening factor to reduce the impact of chain-linking from sites (simulating a human, they imagined that there is a given chance of someone stopping their surfing after a given amount of clicks), in addition to a number of all sites in the dataset, which all PageRanks are divided by, so that every PageRank is a number between 0.0 and 1.0. For this assignment the simple PageRank as described above will suffice (but you are free to extend it further if you wish).

Your task is to implement simplified PageRank using Map and Reduce semantics, and evaluate how this scales with different MapReduce configurations.

Starter code

You will be provided with a simple MapReduce framework in Python that uses `ssh` and NFS for communication across the cluster. You do not need to modify this code, but you may do so if you want to. The main programming task in this assignment is implementing the `map` and `reduce` functions. You also need to modify your `config` file when running the system.

The starter code includes an example implementation of word count and a test script to verify that the map/reduce output is correct. Investigate this implementation in order to understand the semantics that you need to use to express the PageRank problem using the same MapReduce framework. It is recommended that you test the word count implementation and examine how it works. You can use the `run-sanity-check.py` checker to test the correctness of the word count implementation.

Datasets

You will be provided with a dataset consisting of a graph of Wikipedia articles. It contains data about which articles link to each other. Using this graph, you can calculate the Wikipedia-internal PageRank of each article.

Here is an example entry (text formatting used for illustration)

> **Achilles**, *Aegis, Olympias, Apollo, Aeschylus, Patroclus, 1956, Danube, Chiron, Lycophron, Centaur, Chariot, Mysia, 1962, Statius, Toploader, Ethiopia, Sophocles, Plato, Calchas, 1996, Chryses, Odysseus, Scholiast, Trilogy, Euripides, Mummy,...*

This entry describes that the Wikipedia-article for *Achilles* has hyperlinks to articles for *Aegis*, *Olypmpias*, etc.

Calculating the simplified PageRank for this entry can result in something like this:

> ["Achilles", 25.974207490571107]

We are mainly concerned with executing one iteration of PageRank in this assignment, but multiple operations can be chained together to calculate the true PageRank of a site - the calculated PageRank should spread to the sites it links to until the value converges. In order to do this, the output must keep the graph structure of the input intact in the output. This is not required in this assignment (but reflect on how this would influence your implementation).

Report

The report for this assignment need not be lengthy (less than 1000 words), but it has the following requirements:

1. Explain what problem you have solved with your implementation.
2. Show results from your experimental evaluation with graphs.
3. Answer the following questions:
   - How does the time complexity of PageRank scale with different numbers of mappers and reducers?
   - How is this impacted by different datasets sizes?
   - How does this differ from the word count problem?
   - How does the use of multiple workers within the same compute nodes impact this?
4. Explain consisely how your experiments can be used to answer these questions. Provide arguments for your selection of experiments and how you have interpreted their results.

As such, an important aspect of this assignment is the selecting and designing the experiments you need to answer the questions above. All four questions must be considered and answered with relevant results.

Practical information

This assignment can be completed in groups of up to three people. Solving it alone is allowed, but groups of two to three is recommended. Groups of more than three are not allowed. You are not restricted to sticking with the same groups for all assignments or presentations - group management is up to you. When working in groups, it is important that all group members contribute adequately to the to the final hand-in. Taking credit for other people's work, for example by not contributing to the group work, can in the worst case be considered plagiarism or cheating.

Please remember to provide a list of group members somewhere in your hand-in, either through a dedicated text-file in your submission, or as hand-in comments on Canvas.

The assignment should be handed in contained in a zipped folder (.zip), named after your UiT/Feide username (e.g. aov014). This root folder should contain two folders: "doc", containing your report (and supplementary material if any), and "src", containing your implementation. All group members are responsible for handing in their own work.

Find the deadline on the submission site on Canvas.

Cluster

The assignment must be solved on the ificluster (ificluster.ifi.uit.no). The pre-code is dependent on the specific ificluster infrastructure. See files in Canvas for more information about using the cluster. If you have not received access to the cluster (through email), contact the course staff.

If you took the course INF-3200 last semester, you most likely still have access with the same account.