MapReduce-System (37)

Fabian Kleinrad (07), 5BHIF

March 2022

Contents

1 Introduction

In this project the technology MapReduce is being simulated. Thereby a simple system has been developed to imitate a the functionality of an MapReduce application. All of the functionality in this project is written with C++17 and compiled with the help of the meson build system[1]. The communication is based on the TCP protocol and realized using the C++ library asio[2].
Furthermore to increase performance and usability protocol buffers[3] are utilized. Protocol Buffers enable the serialization of data structures in an efficient manner, which simplifies working with messages sent between parties in the MapReduce system.
To make use of the advantages of using a MapReduce architecture, a simple use-case consisting of counting the number of character occurrences in a plain text document. This kind of application was chosen due to its simplicity, which enables the focus of this project to stay on MapReduce rather than a test application.

2 MapReduce

MapReduce is a programming model developed to decrease computation time of large data sets. It was invented by Google, the reason being the need to compute various kinds of derived data. Examples would be inverted indices or representations of the graph structure of web documents. These applications all have simplicity in common, there are no complex operations needed to accomplish said tasks. Furthermore are these kinds of processes characterized by accepting large amounts of input data and reducing it to fraction of itself. MapReduce presents a solution to parallelization, fault-tolerance, data distribution and load balancing. Thereby it is based on the principle of map and reduce, which are eponymous for the technology.[4]

[1] *The Meson Build system.* URL: `https://mesonbuild.com` (visited on 03/30/2022).
[2] *asio C++ Library.* URL: `https://think-async.com/Asio/` (visited on 03/30/2022).
[3] *Protocol Buffers.* URL: `https://developers.google.com/protocol-buffers` (visited on 03/30/2022).
[4] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters.* URL: `https://static.googleusercontent.com/media/research.google.com/de//archive/mapreduce-osdi04.pdf` (visited on 04/03/2022).

2.1 Input Data

MapReduce processes unstructured or semi-structured data. The system is designed to accept large amounts of data in bulk. In the first step of the MapReduce process, this data is being split into subsets, to allow for data distribution. The way this data is being divided depends on the implementation and the type of input data present. The result of splitting the raw data is a set of key/value pairs.[5]

2.2 Map

The map function accepts a set of key/value pairs, with the implementation being provided by the user. Result of this phase is once more a set of key/value pairs. These result pairs represent the significant information contained in the input data. These significant data pairs directly influence the result and all unneeded information is being discarded. The name map stems form assigning a quantity attribute which represents the value of the resulting pairs, to an quality attribute.[6][7]

2.3 Reduce

Sorted key/value pairs get passed to the reduce function, which groups and thereby reduces the set of data points. The logic of this grouping functionality depends on the application MapReduce is used for. For that reason the reduce function is also implemented by the user.[8]

[5] Thomas König Thomas Findling. *MapReduce - Konzept.* URL: `https://dbs.uni-leipzig.de/file/seminar_0910_findling_K%C3%B6nig.pdf` (visited on 04/04/2022).

[6] 6.

[7] 2.

[8] 6.

2.4 Additional Phases

2.4.1 Shuffle

In most implementations of a MapReduce model an shuffle phase is carried out, between the map and reduce phase. The shuffle phase is used to sort the resulting key/value pairs from the mapping phase. This is done in an effort to group similar keys into clusters which can than be reduced by a single worker.[9]

2.4.2 Combine

To reduce the network traffic an additional combine phase can be used after mapping. Thereby the large amount of key/value pairs resulting from the mapping phase get reduced before they get transferred over the network. However, because of this local aggregation of data it is possible to slow down the instead through shuffling optimized process of reducing data.[10]

3 Class-diagram

3.1 Classes

4 Implementation

5 Usage

5.1 Command Line Arguments

5.1.1 Configuration

[9] 6.

[10] *MapReduce Tutorial*. URL: `https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.pdf` (visited on 04/04/2022).

6 Project Structure

References

[1] *asio C++ Library*. URL: https://think-async.com/Asio/ (visited on 03/30/2022).

[2] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. URL: https://static.googleusercontent.com/media/research.google.com/de//archive/mapreduce-osdi04.pdf (visited on 04/03/2022).

[3] *MapReduce Tutorial*. URL: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.pdf (visited on 04/04/2022).

[4] *Protocol Buffers*. URL: https://developers.google.com/protocol-buffers (visited on 03/30/2022).

[5] *The Meson Build system*. URL: https://mesonbuild.com (visited on 03/30/2022).

[6] Thomas König Thomas Findling. *MapReduce - Konzept*. URL: https://dbs.uni-leipzig.de/file/seminar_0910_findling_K%C3%B6nig.pdf (visited on 04/04/2022).