University of Basel

Prof. Dr. Heiko Schuldt
Ralph Gasser, M. Sc.
Alexander Stiemer, M. Sc.
Marco Vogt, M. Sc.

Distributed Information Systems — Spring 2018

Exercise 2 — Hand-in: 08.04.2018 (11:59pm)

Advisor: Marco Vogt (marco.vogt@unibas.ch)

Modalities of solving the exercises: The exercises can be solved in small groups of a maximum of two people.

Modalities of the exercise: This exercise has to be uploaded to the courses website (`https://courses.cs.unibas.ch/`) before the deadline and presented to the advisor. For this, everyone has to schedule an appointment with an advisor by reserving a slot on `https://doodle.com/poll/58mifc9sy8d6h9tc`. Every slot has a capacity of 2 and for groups every group member has to reserve a slot. Please note that reservations are made on a first-come-first-serve basis. In this meeting, the work and the understanding of the technical background is evaluated. Therefore, each group member has to attend this meeting and must be able to fully explain the solution. If this is not the case, the whole exercise will be graded with zero points.

Introduction

Data has been growing rapidly on the web over the past few decades and by the year 2020 it is estimated to grow up to 44 ZB (4.4 ZB in 2015) [1]. Organizations are interested in capturing and analyzing this data for trends and in mining other useful information. The amount of data requires systems that possess very high processing power. The need for such processing power pushes the boundaries of traditional data warehouse and data management techniques, which is where Big Data technologies come into the picture.

Big Data usually includes datasets with sizes that are beyond the ability of commonly used software tools to capture, curate, manage and process it within a tolerable time frame. A general architecture of a Big Data system would be an implementation of a distributed file system like the Google File System or Hadoop File System over a cluster of computers or commodity servers. This file system combines the processing power of the underlying machines to create a system with the capability to handle very large datasets. Data stored on these file systems are distributed and replicated, which allows for parallel processing and thereby increasing the speed of operations. In addition to structured information stored neatly in tabular form (rows and columns), Big Data comes in complex, unstructured formats, including everything from websites, social media and email, etc. Thus, in addition to a need for technologies that could work with massive volumes, we also need these technologies to be able to process this data without enforcing any schema on it. One such iterative programming model is called MapReduce. In the following subsections, the MapReduce Programming Model and Apache Hadoop (a framework that implements MapReduce) are described.

MapReduce Programming Model

MapReduce[1] is a programming model that allows for parallel and distributed processing of data sets. It is particularly well suited to process data on a cluster of nodes. MapReduce operates on key-value pairs and has three major phases: First, during the *map* phase, the input is read, processed and zero to n key-value pairs are emmitted. Subsequently, during the *shuffle* phase, the key-value pairs are sorted and re-distributed among the worker nodes based on their key. Finally, the key-value pairs are re-combined (aggregated) during a *reduce* phase and the result is usually written to disk.

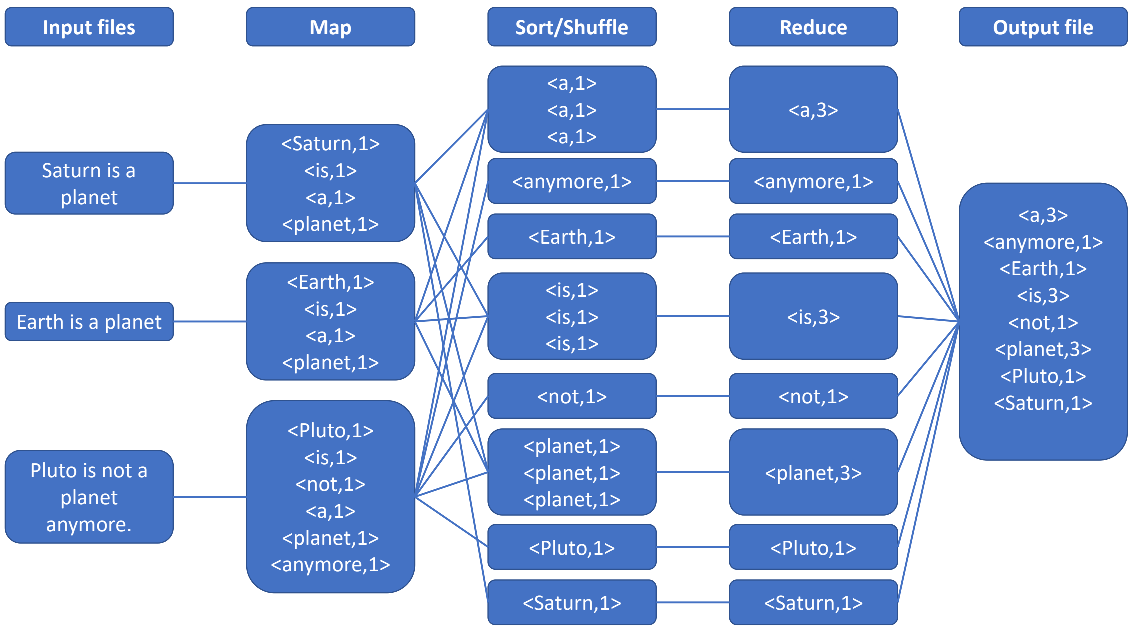
Figure 1 gives an example of a word-count application implemented as MapReduce. When an input text-file is submitted to the *map* phase, the content of the file (the text) is divided into a set of key-value pairs `<word,1>`. The intermediate results are then sorted by key (the word) and re-distributed among the nodes, where during the *reduce* phase the values of all pairs belonging to the same key are added up, which results in a list of key-value pairs `<word,occurence>`.

Apache Hadoop

Apache Hadoop [2] is a Java-based open source framework that supports distributed storage and processing of large datasets over a network cluster. A few features that make Apache Hadoop popular are:

1. Cost Effectiveness: The network cluster on which this Java framework is installed can be comprised of simple commodity servers or computers. Thus, the infrastructure cost is highly affordable.
2. Highly Scalable: New nodes can easily be added to the network cluster to increase storage capacity and processing performance without much change in configuration of the existing system.
3. Fault Tolerant: By default, Hadoop replicates data stored on a given node to two other locations, thus, even if one or two nodes are down the data is still available.

[1]The name was inspired by the primitives *map* and *reduce* from Lisp.



Input files
Map
Sort/Shuffle
Reduce
Output file
Saturn is a planet
Earth is a planet
Pluto is not a planet anymore.
<Saturn,1> <is,1> <a,1> <planet,1>
<Earth,1> <is,1> <a,1> <planet,1>
<Pluto,1> <is,1> <not,1> <a,1> <planet,1> <anymore,1>
<a,1> <a,1> <a,1>
<anymore,1>
<Earth,1>
<is,1> <is,1> <is,1>
<not,1>
<planet,1> <planet,1> <planet,1>
<Pluto,1>
<Saturn,1>
<a,3>
<anymore,1>
<Earth,1>
<is,3>
<not,1>
<planet,3>
<Pluto,1>
<Saturn,1>
<a,3> <anymore,1> <Earth,1> <is,3> <not,1> <planet,3> <Pluto,1> <Saturn,1>

Figure 1: The MapReduce steps of a word-count program

4. Flexibility in Data Processing: It addresses a common problem that most organizations had with processing of unstructured data. Whether the data is structured or unstructured, coded or uncoded, Hadoop has the capability to process it.

5. Fast Processing Time: With respect to complexity and the sheer volume of data to be processed, Hadoop has very good performance characteristics. Many different companies use Hadoop to analyze their vast volumes of data for their product needs on a daily basis.

The architecture of Apache Hadoop is composed of four core components as depicted in Figure 2:

1. **Node Cluster**
   An Apache Hadoop Node cluster is a set of nodes connected to each other in a network on which the Apache Hadoop Framework is installed. Apache Hadoop is designed to work on both: on a single-node setup as well as on a multi-node cluster (Figure 3). There are two types of servers, which are *master* and *slave* servers (called NameNode and DataNode respectively), installed on the nodes. In a single node cluster setup both these servers are installed on the same node. In a multi-node cluster setup, one node could function as a master server and the others as slaves. There is also an option in which the node that has the master server installed can also have a slave server.

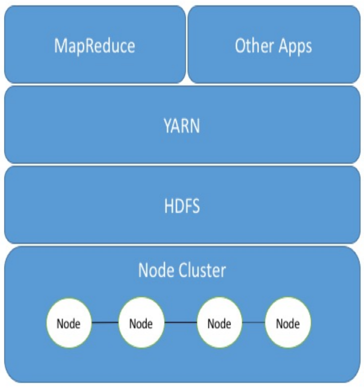


Figure 2: The Architecture of Hadoop

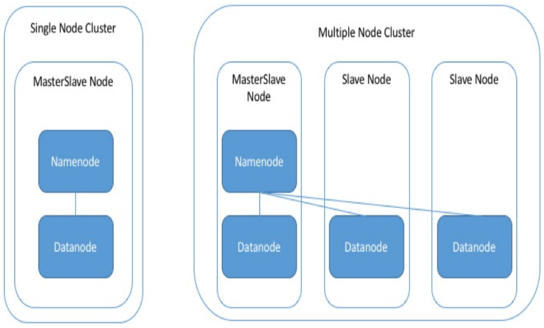


Figure 3: Hadoop Single-Node and Multi-Node Setup

2. **Hadoop Distributed File System (HDFS)**
   This is a file system that provides reliable data storage and access to all nodes in the Hadoop cluster. The physical file systems of all nodes in the cluster are linked together by the HDFS to form a single file system across all nodes. The HDFS has a master/slave architecture with a single NameNode as master and multiple DataNodes as slaves (ideally one per node in the cluster). The function of the NameNode is to maintain the file system on the whole cluster. Thus, it primarily performs meta-data operations. The DataNodes manage the storage of data on the nodes that they run on. Files stored on HDFS are generally replicated and split into smaller parts and stored on these DataNodes. DataNodes perform file system operations like read, write, etc. and replication upon instruction from the NameNode.

3. **Yet Another Resource Negotiator (YARN)**
   YARN is a cluster management technology. It assigns processing resources like CPU, memory and storage to applications running on the Hadoop cluster.

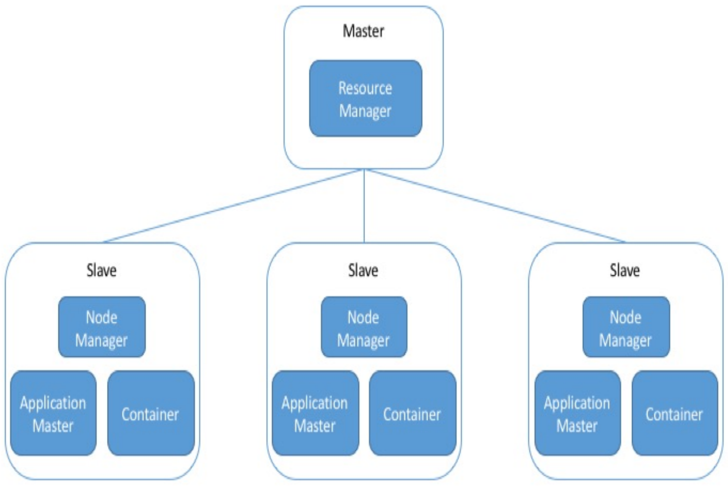


Figure 4: YARN

The components of YARN are depicted in Figure 4. A *Container* is an abstract representation of compute resources available on a given node. Every node is managed by the *NodeManager*, which lets the central *ResourceManager* know, what resources are currently available. The *ApplicationMaster* represents an application running on Hadoop and it negotiates with both the *NodeManager* and the *ResourceManager* to get the desired resources for executing its program.

4. **MapReduce**
   The current versions of Hadoop can configure MapReduce to run over YARN. When the MapReduce Framework runs over YARN, it has the same components as outlined in Figure 4, i.e. a Resource Manager, Node Manager, MapReduce Application Master (specific for MapReduce Applications) and Containers. Minimally, a MapReduce application specifies input/output locations of data on HDFS and a set of *Map* and *Reduce* functions to be applied on this data as a part of a job configuration that needs to be executed.

Motivation and Application

In this exercise, you will implement a basic Big Data application using Apache Hadoops MapReduce [3] framework. The exercise is designed with a given set of questions that build on each other. The solutions to each of these questions shall be the stepping stones to develop the required application. You first begin with an installation of Apache Hadoop's MapReduce framework on a multiple node cluster. Then, you will develop a basic MapReduce application that implements a program that counts the word occurrences on a small set of files. Further, you develop a more complex MapReduce application that works with a large dataset. This dataset is not as big as the datasets that exist in the industry. However, they suit the educational purpose of this exercise, as in, they are large enough to provide exposure to Big Data concepts.

One of the main reasons organizations store data is so that they can analyze it and make informed decision based on their analysis. For example, every company wants to know how its customers feel about it and its products. Online reviews are usually analyzed by manufacturers to decide what products they would like to produce in the future. Thus, for this exercise you will work with an online movie review dataset from Amazon and implement solutions to find the most popular movies by analyzing the review text (the review rating score has been purposefully ignored).

The solution approach does have an interdisciplinary nature wherein a few concepts of Machine Learning could be used. However, it is important to note that the focus of this exercise is the Data Management aspect of Big Data and NOT Data Analysis. Therefore, in order to keep the analysis techniques simple, the suggested solution uses a naive word count approach. However, if you have good working knowledge of a few machine learning techniques, then you are free to implement them along with MapReduce but remember that the machine learning technique is NOT the motive of this exercise.

Information on the Dataset

In the following exercise, you shall develop a MapReduce application that evaluates the popularity of movies by analyzing its reviews. This shall be accomplished by analyzing two datasets: (1) a training dataset from the Association for Computational Liguistics (ACL) project [5] and (2) an Amazon movie review dataset [6]. At a high level, this application learns from the training dataset (ACL) and grades the reviews from the Amazon dataset as positive and negative reviews. These grades are then used to find the most popular movies or find the list of movies a given user likes, etc.

The ACL dataset is attached with this exercise. There are 2 folders in this data set – `pos` and `neg`. Each of these folders contains a file with 12500 reviews that have already been classified as either a positive or a negative review.

The `amazonmoviereview` folder contains files with data from the Amazon Movie Review dataset. This dataset has 5 types of files: `link.txt`, `productId.txt`, `reviewInfo.txt`, `reviewTxt.txt` and `userData.txt`. The attribute-wise distribution of data is as follows:

1. `link.txt` - reviewId, productId, userId
2. `productId.txt` - productId, actor (multivalued attributed), director, language, studio, title
3. `reviewInfo.txt` - reviewId, helpfulness, score, time
4. `reviewTxt.txt` - reviewId, summary, text
5. `userData.txt` - userId, profileName, age, country, gender

Note: the time field in the `reviewInfo.txt` is in Unix time format.

Getting the Dataset

You can download the dataset here:
`http://download-dbis.dmi.unibas.ch/DIS/15729-DIS18-ex2-Dataset.zip`

Expected Deliverables

1. Source code with appropriate comments on the steps taken.
2. Jar files of the MapReduce application.
3. Command-line instruction to run the jar file (including program arguments).
4. Output files of your solutions.
5. A descriptive design document that visualizes the map and reduce steps in the achieved solutions to questions 3 and 4.
6. Your Hadoop configuration files (*core-site.xml*, *hdfs-site.xml*, *mapred-site.xml*, *yarn-site.xml*)

Question 1: Installation of Apache Hadoop (0 points)

In this task you shall setup a three-node (multi-node) hadoop cluster on the AWS platform and install Apache Hadoop on it. Therefore you should use three EC2 nodes (e. g. t2.large). The learning objective of this step is to gain familiarity with the components of Apache Hadoop, mainly the HDFS and the MapReduce framework.

Question 2: Develop a basic Word-Count Program (2 points)

After the installation of Apache Hadoops MapReduce framework, you shall develop a basic MapReduce word-count application. The input files for this application shall be taken from the Gutenberg dataset that come with this exercise. The files in the `gutenberg` folder include 3 txt-files that are text copies of 3 books from the Gutenberg Project [4]. The program should count the occurrence of words in the given books.

This program needs to be executed on the node cluster setup from Question 1. It is required that the dataset is loaded from the local file system on to the HDFS (you may use the `hdfs dfs -put` command) and the program needs to be executed from the cluster itself. We suggest to develop the solution on your local machine and then transfer the jar file to the node cluster for execution (use `scp` or `rsync` command to transfer files to the multi-node cluster).

A successful execution of this program verifies the installation of the Apache Hadoop MapReduce framework and helps you in familiarizing yourself with the MapReduce paradigm.

Question 3: Assign Scores to a Review (8 points)

a) Develop a learning list of scores
Use the word-count approach to create a list of words and use the number of occurrence of a given word across reviews as it's score. You have to use the ACL project dataset for this task (`acl` folder). Thus, a positive review score for a given word is the number of times it occurs in the set of positive reviews (`pos.txt`) and it's negative review score is the number of times the same word occurs in negative reviews (`neg.txt`). At the end of this step, the output is a list of words with their positive review scores and negative review scores.

(4 points)

b) Score each movie review based on the learning list
Use the same word count approach on each single review. In this step we only use the review text field from the review dataset. For each review, read the list of words in the review text and sum up the 2 scores (positive and negative review scores available from the previous step) for every word in the review. Thus, the output of this step is the list of both positive and negative review scores against a review id.

(4 points)

Output format suggestion as key value pair:

```
<review_id>:<positive_score>p/<negative_score>n
```

Optional: The two steps for finding the positive and negative review scores for a given review use a basic word-count approach. Students with better machine learning knowledge who would like to implement a more refined prior with training dataset and build scores with a Naive Bayes Classifier or another method should feel free to do so. However, it is best to keep the final output format consistent with the one suggested above so to remain compatible with the next steps.

Question 4: Analyze the Scores (10 points)

Analyze the movies based on the review scores you calculated in the previous task and store the results in a separate files.

a) List the top 5 most popular movies (i.e. movies with the highest average score) in the year 2008.

(4 points)

b) List the movies that a given user rates as good movies, i.e., with an average score greater than zero.

(4 points)

c) Provide one additional unique and interesting list. Please note, that "unique" means that it should not be similar to the lists from a) or b).

(2 points)

The solutions to the tasks above must be designed along the lines of the following definitions:

1. Average Score: The average score of a movie is given by $\frac{p-n}{p+n}$, where p denotes the positive score and n denotes the negative score.
2. Classification: A review is classified as positive review if the number of positive ratings exceed the number of negative ratings and vice versa, which is equivalent to say that the average score is greater than zero.

Interview (10 points)

During the presentation of your solutions, you will be asked questions related to your specific solution of the exercise and to the topics discussed in the lecture. You can receive the points by correctly answering all questions.

References

[1] Berbard Marr. *Big Data: 20 Mind-Boggling Facts Everyone Must Read*, 2015. `http://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/#263aa27d6c1d`.

[2] Apache Software Foundation. *Apache Hadoop 2.7.0*, 2015. `https://hadoop.apache.org/docs/r2.7.0/index.html`.

[3] Apache Software Foundation. *MapReduce Tutorial*, 2015. `https://hadoop.apache.org/docs/r2.7.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html`.

[4] Project Gutenberg. *Free ebooks*. `https://www.gutenberg.org/`.

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[6] SNAP. *Web Data: Amazon Movie Reviews*, 2015. `https://snap.stanford.edu/data/web-Movies.html`.