# A2 - MapReduce and Hadoop

## Introduction:

The objective of this assignment is to gain familiarity with the MapReduce programming model using the Hadoop framework.

To pass the assignment (and thus gain 2 points) you need to complete all tasks in part 1. Students who wish to try for 3 or 4 points should also hand in part 2.

You will be required to hand in a **report** for A2 that should contain answers to the questions in this document as well as plots displaying your results and a discussion of your results.

The code that you are instructed to hand in (specified at each task) should be attached **as separate documents** and handed in together with the report in Studium.

Be advised that you will have to consult lecture notes and external resources to complete the assignment. Some useful links are provided, but you will also have to search for additional information on your own.

# Part 1

## Background

For this task you will boot a new virtual machine instance based on a snapshot that we provide you with. The snapshot is a customized image containing a complete Hadoop 3.3.1 framework, see:
https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

Hadoop is installed in “/usr/local/hadoop-3.3.1”

Java path can be found by “echo $JAVA_HOME”

The executable binaries are in the directory “/usr/local/hadoop-3.3.1/bin”. If you like, you can add this directory to the environment variable PATH, but it is not necessary.

**Important:** For Hadoop to work in all examples below, you need to set the hostname of your instance in /etc/hosts. Open the file (you need to be sudo to edit) and modify the first line:

```
127.0.0.1      localhost      <your hostname>
```

# Task 1. Introduction to the Hadoop Framework

## Task 1.0: Setup

- Follow the instructions below to start a new VM, and review A1 if needed.
    - Under Compute-Instances, click Launch Instance
    - Use the source instance snapshot “hadoop-lab2-snapshot”.
    - Use Volume Size = 40GB.
    - Select “Yes” to “Delete Volume on Instance Delete”.
    - Use the flavor ssc.medium.
    - Use the security group “default”.
    - Your instance name should contain your own full name so the owner can be identified.

- After the instance has been launched: go to the “Volumes” menu in the OpenStack dashboard and locate the volume attached to your instance, and rename it with your full name.
- Change the file /etc/hosts on the VM to contain your hostname, like the instruction in red above says.

## Task 1.1: Word count example in local (standalone) mode

This task requires successful execution of a basic Hadoop program. The Hadoop framework executes programs using MapReduce. The code to be run is available as a test example shipped with the Hadoop libraries. As a reference to Hadoop MapReduce, consult:

https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Use this tutorial (and other provided links) to understand the executed commands for this lab and to answer the questions.

1. Create a folder “input” in the directory “wordcount” located in the home directory of your instance.
2. Then download the following data file and place it in that folder:

   http://www.gutenberg.org/ebooks/20417.txt.utf-8

**Hint:** On Ubuntu linux you can use the ‘wget’ command to download data over http.

3. Then execute the following command to run the canonical MapReduce example:

```
/usr/local/hadoop-3.3.1/bin/hadoop jar
/usr/local/hadoop-3.3.1/share/hadoop/mapreduce/hadoop*examples*.j
ar wordcount /home/ubuntu/wordcount/input
/home/ubuntu/wordcount/output
```

**Hint:** Copying commands from a pdf file often does not work. If you get errors, try to type in the command instead. It should all be on one line.

**Note:** the folder names “input” and “output” are arbitrary - you can use any names you like for the input and output directories. But the “input” folder should contain the downloaded file. Note that the output folder is created by Hadoop, so you should not create it manually before running the command above.

4. Answer the following questions:
    a. Look at the contents of the folder “output” - what are the files placed in there? What do they mean?
    b. How many times did the word ‘Discovery’ (case-sensitive) appear in the text you analyzed?
    c. In this example we used Hadoop in “Local (Standalone) Mode”. What is the difference between this mode and the Pseudo-distributed mode?

## Task 1.2: Setup pseudo-distributed mode

1. Follow the instructions in the link below to set up Hadoop in pseudo-distributed operation. Do the instructions for **Configuration**, **Setup passphraseless ssh** and steps **1,2,4** of **Execution.**

   **Hint:** In the instructions, the hadoop folder paths given are relative to the installation directory. In our case the installation directory is ‘/usr/local/hadoop-3.3.1’. **You will have to modify the given commands accordingly.**

   **Warning:** Do **NOT** format namenode more than once, otherwise there will be too many clusterID, and you will face problems with starting namenode**.**

   https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation

2. To verify that services are running, you can use the command ‘jps’

```
jps
```

You should expect to see an output like this (the numbers are the process IDs (PIDs), these will be different in your output.):

```
ubuntu@test-hadoop-from-snapshot:~$ jps
17922 NameNode
18266 SecondaryNameNode
18047 DataNode
18383 Jps
```

3. Answer the following questions:
   a. What are the roles of the files core-site.xml and hdfs-site.xml ?
   b. Describe briefly the roles of the different services listed when executing 'jps'.

## Task 1.3: Word count in pseudo-distributed mode

Now we will use Hadoop in pseudo-distributed mode to build and run our own version of the MapReduce example. The directory wordcount contains the java source file for the word count example we ran in Task1 (WordCount.java).

1. To compile the wordcount example and make a jar-file:

```
cd /home/ubuntu/wordcount

javac -cp `/usr/local/hadoop-3.3.1/bin/hadoop classpath` WordCount.java

jar -cvf wordcount.jar  *.class
```

***Obs:*** *Don't just copy and paste this command. The backquotes in the second command are supposed to be the grave accent diacritical mark, it encapsulates the inner command so that the inner command in backquotes will be executed first.*

You should now have a file "wordcount.jar" in your folder.

2. Next, load the file whose words we are counting into hdfs. The file is in the folder /home/ubuntu/wordcount/input. You can stage the input folder in HDFS by:

```
/usr/local/hadoop-3.3.1/bin/hdfs dfs -put /home/ubuntu/wordcount/input
```

3. You should now have added the text file from the file system of your VM to the hadoop File System (HDFS). Use the command below to verify that the file is indeed in HDFS:

   ```
   /usr/local/hadoop-3.3.1/bin/hdfs dfs -ls input
   ```

   You should see an output that looks something like this:

   ```
   Found 1 items
   -rw-r--r--   1 ubuntu supergroup 67 2020-03-30 10:12 input/20417.txt.utf-8
   ```

4. Run the following Hadoop command. Name your own `<output_dir>`.

   ```
   /usr/local/hadoop-3.3.1/bin/hadoop jar wordcount.jar WordCount
    input <output_dir>
   ```

   If the hadoop run completes normally, verify that the output looks as expected. First check the content of the output directory in hdfs,

   ```
   /usr/local/hadoop-3.3.1/bin/hdfs dfs -ls <output_dir>
   ```

5. Then check the content of the output file using the '-cat' argument to 'hdfs dfs'. Verify that it contains the word counts you saw before when running the word count program locally.

   ```
   /usr/local/hadoop-3.3.1/bin/hdfs dfs -cat <output_dir>/part-r-00000
   ```

6. Answer the following questions:
   a. Explain the roles of the different classes in the file WordCount.java.
   b. What is HDFS, and how is it different from the local filesystem on your virtual machine?

## Task 1.4: Modified word count example

1. Modify the above example code in WordCount.java so that it, based on the same input files, counts the occurrences of words starting with the same first letter (non case-sensitive). The output of the job should thus be a file containing lines like:

   a   number_of_words_starting_with_a_or_A

2. Make a plot that shows the counts for each letter and include that in your report. *Do not include the entire output file.*

**Tips:**

1. Again, use the tutorial link provided in task 1.2 and see how they "get" data from hdfs to the home folder on the instance.
2. For the plotting you will need to transfer data between hosts (your local computer and your instance) through ssh, for this there is a command called "scp" (Secure Copy). Don't forget that you need your key ( -i option)
3. There is a useful class called "Character" in Java that might come in handy.

Hand in the following code:

- A file FirstLetterCount.java with your modified codes and sufficient comments of codes.

## Task 1.5: NoSQL and MongoDB

### Background

NoSQL stands for "Not only SQL" and consists of database solutions that don't use the more traditional relational model of data in the form of tables. NoSQL databases are frequently used within big data analytics and are especially suited for semi-structured and unstructured data. They are distributed solutions, scale well horizontally and are open-source. One example of NoSQL is the document-store MongoDB which stores data in JSON-like documents (https://www.mongodb.com/).

Answer the following questions:

1. One example of JSON formatted data is Twitter tweets: https://dev.twitter.com/overview/api/tweets. Based on the twitter documentation, how would you classify the JSON-formatted tweets - structured, semi-structured or unstructured data?
2. Elaborate on pros and cons for SQL and NoSQL solutions, respectively. Give some examples of particular data sets/scenarios that might be suitable for these types of databases. (expected answer length: 0.5 A4 pages)

## Part 2

(This part of the assignment is only required if you want to collect points toward higher marks)

## Task 2.1: Analyzing twitter data using Hadoop streaming and Python (awards up to 1 point)

### Background

While Hadoop/MapReduce is based on Java, it is not necessary to use Java to write your mapper and reducers. The Hadoop framework provides the “Streaming API”, which lets you use any command line executable that reads from *standard input* and writes to *standard output* as the mapper or reducer. The following tutorial, although a bit old, provides an excellent introductory example to using Python and Hadoop streaming:

http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

**Important notes:**

1. The tutorial above uses Python 2 (which is no longer maintained past 2020). **You should use Python 3.**
2. The tutorial above is **not** using Hadoop 3. Although you are expected to follow these steps, please note that there could be some subcommands/options in the tutorial deprecated in Hadoop 3. **You should manage to fix them by reading error/warning messages.**
3. The location of the jar file for the Hadoop streaming API is in a different location than in the tutorial. On your distribution it is located in **/usr/local/hadoop-3.3.1/share/hadoop/tools/lib/** (and NOT in /usr/local/hadoop/contrib/streaming/)

### Data

In this part of the assignment, we will analyze a dataset of ~5.000.000 Twitter tweets collected using Twitters datastream API. The total size of the dataset is still a modest ~9GB. The data is available as a tarball on the instance in the home folder.

Each tweet is a JSON document http://en.wikipedia.org/wiki/JSON. JSON is one of the standard Markup formats used on the Web. For the specific case of Twitter tweets, you can read about the possible fields in the documents here:
https://dev.twitter.com/overview/api/tweets

This particular Twitter dataset was collected by filtering the stream of tweets to store those containing the Swedish pronouns “han”, “hon”, “den”, “det”, “denna”, “denne”, and the gender neutral pronoun “hen”.

## Do the following:

1. Use Python and the Hadoop streaming API to count the **number of tweets** mentioning each of these pronouns.
    - Use the tutorial linked above (noting the Important Notes)
    - In this analysis, only unique tweets should be taken into account, i.e. ‘retweets’ should be disregarded.
    - The count should be case-insensitive, i.e. “Han” and “HAN” should count as mentions of the pronoun “han”.

2. Plot a bar chart visualizing the counts of each pronoun.
    - The counts that you plot should be **normalized** by the total number of **unique** tweets (e.g $\frac{number\ of\ unique\ tweets\ containing\ 'hen'}{total\ number\ of\ unique\ tweets}$).

**Hint 1:** Use the Python library ‘json’ to parse the tweets.
**Hint 2:** It can also be good to look up regular expressions for this task. There is a library in python called ‘re’.

Hand in the following code:
- mapper.py and reducer.py files with your map code and reduce code.

Include descriptions about your implementation and the results in your report.

## Task 2.2: NoSQL and MongoDB (awards up to 1 point)

### Do the following:

1. Redo the analysis from Task 2.1, now using MongoDB.
    - Use the exact same data as in Task 2.1
    - Some suggested options include using the Mongo Shell or PyMongo
    - This is an open problem, it is up to you to experiment with MongoDB and develop a solution to the problem that makes efficient use of MongoDBs capabilities. Add complexity to the problem if you desire. Note that there is no requirement to use a specific language or interface - the implementation is up to you.
    - Present plots corresponding to those you obtained for Task 2.1

2. Answer the following question:
   Motivate your chosen implementation and how you did it. What are some pros and cons of the MongoDB solution compared to the implementation you did in Task 2.1?

Hand in the following code:

- All code for your MongoDB implementation, as well as all commands/queries used to obtain the results (e.g. Mongo shell queries).