Universität Freiburg
Institut für Informatik

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg

Fang Wei-Kleiner

fwei@informatik.uni-freiburg.de

Advanced Databases and Information Systems
Summerterm 2019
Discussion on

6. Sheet: Distributed Processing I

Exercise 1 (The Google File System)
Read the Paper "The Google File System," available online at `https://research.google.com/archive/gfs-sosp2003.pdf`, and answer the following questions:

a) What are the radically different design choices addressed by the Google File System (GFS) compared to previous distributed file systems?

b) Which operations are unusual for file systems, but provided by the Google File System?

c) What are the special caching mechanisms in the Google File System?

Exercise 2 (Setup)
For the MapReduce exercises, you can either install Hadoop and use Java (or Python) to write and evaluate your code, or only write pseudocode. While the latter suffices for the exam, we recommend to use Hadoop in order to test your code.
Hadoop: Install the Hadoop Distribution of Cloudera [1] in Pseudo-Distributed Mode (easiest way is to use Cloudera Manager[2] and choose the free Cloudera Express Edition) or use one of the Virtual Machine Images provided by Cloudera. Alternatively you can also install Hadoop alone.[3]

a) Familiarize yourself with Hadoop, especially with the distributed file system HDFS and the various Web Interfaces (Cloudera Manager, YARN Resource Manager UI, Namenode UI, Hue). The Web Interfaces should be accessible via:
   - Cloudera Manager: <your machine address or ip>:7180
   - YARN Resource Manager UI: <your machine address or ip>:8088
   - Namenode UI: <your machine address or ip>:50070/dfshealth.html#tab-overview
   - Hue: <your machine address or ip>:8888

b) Download the Java project `wordcount.zip` from the ILIAS course page. You can use it as a kick-start for the development of MapReduce applications. We recommend using Maven as it automatically resolves the needed Hadoop dependencies.

   If you use Eclipse on a Linux machine where Hadoop is installed, you can test run your applications from within Eclipse. If you do so, be aware that this runs Hadoop in local mode, reading files from your local file system instead of HDFS using only a single reducer (even if more reducers are specified in your application).

[1] `http://www.cloudera.com/products/apache-hadoop/key-cdh-components.html`
[2] `http://www.cloudera.com/downloads/manager/5-7-0.html`
[3] See `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/`.

The project contains an already pre-compiled binary (see folder 'dist'). To run the WordCount application on your machine in pseudo-distributed mode, use the following command:

```
hadoop jar dist/MapReduce_apps.jar wordcount.WordCountDriver \
<input dir> <output dir> <num of reducers>
```

Run this program for a large text file and review the most frequent words.

The same command will also run your application in fully-distributed mode on a cluster. Note that the input directory must exists in HDFS and the output directory must not exist.

Exercise 3 (Simple MapReduce)
Design and implement a MapReduce program that takes a large file of integers as input and outputs
a) the largest integer
b) the average of all the integers.

Exercise 4 (Basic Operations)
For the following tasks use the file '`sibdataset200.zip`' as input which contains a social network with 200 users. The file is available at `http://dbis.informatik.uni-freiburg.de/content/courses/SS17/Vorlesung/Distributed_Systems/sibdataset200.zip`.

a) Implement a MapReduce program that retrieves the friends of a user, represented by the edge `'foaf:knows'`. This user should be specified dynamically as a parameter while starting your MapReduce program.

   Example: For user `'sibu:u107'`, your program should emit in total 49 other users.

b) Implement a MapReduce program that counts the amount of likes for each user, where each like is represented with the edge `'sib:like'`. This should be done for all users in parallel. You can use the word count example as starting point.

   Example: The user `'sibu:u107'` has 1608 likes.

Exercise 5 (Aggregations)
Compute how many likes users have on average. Likes are represented by the edge `'sib:like'`.

Exercise 6 (Selections)
Determine the most popular users in the social graph. You can assume that popularity is measured by the amount of ingoing `'foaf:knows'` edges only. The more such edges exist, the more popular a user is.

Furthermore, consider a threshold α, which specifies how many ingoing edges a user should have at least, to be considered as popular. This value should be specified dynamically as a parameter while starting your MapReduce program. The order of emitted users is not of importance.