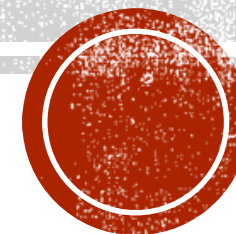


Doc. RNDr. Irena Holubová, Ph.D. & PROFINIT

DATA SCIENCE

NDBI048

Practicals 11. MapReduce



<https://www.ksi.mff.cuni.cz/~holubova/NDBI048/>

MAPREDUCE FRAMEWORK

RECAPITULATION FROM THE LECTURE

- A programming model and associated implementation
- Supports distributed, parallel computing on large data
- Divide-and-conquer paradigm
 - **Map** processes a key/value pair to generate a set of intermediate key/value pairs
 - $(k_1, v_1) \rightarrow \text{list}(k_2, v_2)$
 - **Reduce** merges all intermediate values associated with the same intermediate key
 - $(k_2, \text{list}(v_2)) \rightarrow (k_2, \text{possibly smaller list}(v_2))$
- Programmer focuses on map/reduce code
- Framework cares about data partitioning, scheduling execution across machines, handling machine failures, managing inter-machine communication, ...



MAPREDUCE

EXAMPLE: WORD FREQUENCY

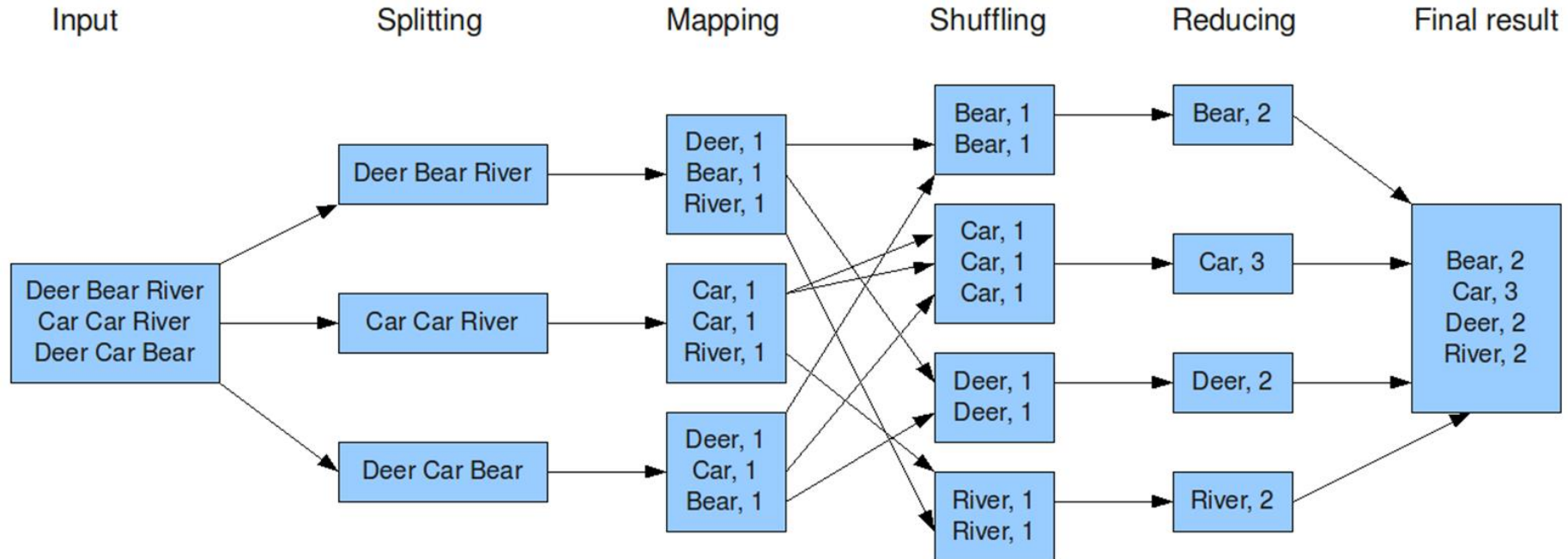
```
map(String key, String value):  
    // key: document name  
    // value: document contents  
    for each word w in value:  
        EmitIntermediate(w, "1");
```

```
reduce(String key, Iterator values):  
    // key: a word  
    // values: a list of counts  
    int result = 0;  
    for each v in values:  
        result += ParseInt(v);  
    Emit(key, AsString(result));
```



EXAMPLE — MAPREDUCE

EXAMPLE: WORD FREQUENCY



APACHE HADOOP



- Open-source software framework
- Running of applications on large clusters of commodity hardware
 - Multi-terabyte data-sets
 - Thousands of nodes
- Implements MapReduce
- Derived from Google's MapReduce and Google File System (GFS)
 - Not open-source

<http://hadoop.apache.org/>



ACCESS

- Use SSH
 - host: nosql.ms.mff.cuni.cz
 - port: 42222
 - login: ds_<Surname>
 - Or surname+1st letter of First name
 - E.g. ds_Novak or ds_NovakP
 - password: Specified in the lab (or send me an e-mail)
- Please, **do not upload too large data**
 - We have a limited disk space and too many users
 - It is not the aim of the practicals, the principles work also for small data
- Sample data shown during practicals can be found in
 - /home/NDBI040/mapreduce
 - /home/NDBI040/mapreduce2



FIRST STEPS

hadoop

- **Help for Hadoop commands**

hadoop fs

- **Help for HDFS commands**

hadoop jar

- **Execution of MapReduce jobs**

hadoop fs -ls /

hadoop fs -ls /user/

hadoop fs -ls /user/<login>

Note: Copy the commands from commands.txt



Testing example in Java

COMPILE TESTING EXAMPLE WORDCOUNT.JAVA

- **Create your own directory**

```
cd ~
```

```
mkdir -p mapreduce/WordCount
```

```
cd mapreduce/WordCount
```

- **Make a copy of the sample java source file**

```
cp /home/NDBI048/mapreduce/WordCount.java .
```



COMPILE TESTING EXAMPLE WORDCOUNT.JAVA

- **Compile sample Word Count implementation**

```
mkdir classes
```

```
javac -classpath  
/home/NDBI048/mapreduce/hadoop-common-  
3.1.1.jar:/home/NDBI048/mapreduce/hadoop-  
mapreduce-client-core-3.1.1.jar -d classes/  
WordCount.java
```

```
jar -cvf WordCount.jar -C classes/ .
```



STORE TESTING DATA TO HDFS

- **Create your HDFS working directories**

```
hadoop fs -mkdir /user/<login>/WordCount
```

```
hadoop fs -mkdir  
/user/<login>/WordCount/input1
```

- **Prepare the sample input data**

```
hadoop fs -copyFromLocal  
/home/NDBI048/mapreduce/input1/movies.txt  
/user/<login>/WordCount/input1
```



RUN TESTING EXAMPLE

A SMALL FILE

```
hadoop jar WordCount.jar WordCount /user/<login>/WordCount/input1  
/user/<login>/WordCount/output
```

- **Run the example (it first deletes the output directory /user/<login>/output)**

```
hadoop fs -copyToLocal /user/<login>/WordCount/output1/part-r-  
00000 result.txt
```

```
cat result.txt
```

- **Check the output**

```
hadoop fs -rm -r /user/<login>/WordCount/output1/
```

- **Delete the output directory, if required**



STATUS: COMMAND LINE

```
holubova@bde-bc5eda-hadooper-client-0-mapred:~  
rm: `output': No such file or directory  
[holubova@bde-bc5eda-hadooper-client-0-mapred ~]$  
[holubova@bde-bc5eda-hadooper-client-0-mapred ~]$ hadoop jar wc.jar WordCount /u  
ser/holubova/input1 /user/holubova/output  
17/03/31 20:56:14 INFO client.RMProxy: Connecting to ResourceManager at bde-bc5e  
da-hadooper-computemaster-0-mapred.virt.local/10.20.1.61:8032  
17/03/31 20:56:14 WARN mapreduce.JobSubmitter: Hadoop command-line option parsin  
g not performed. Implement the Tool interface and execute your application with  
ToolRunner to remedy this.  
17/03/31 20:56:15 INFO input.FileInputFormat: Total input paths to process : 2  
17/03/31 20:56:15 INFO mapreduce.JobSubmitter: number of splits:2  
17/03/31 20:56:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_14  
88293277689_0216  
17/03/31 20:56:16 INFO impl.YarnClientImpl: Submitted application application_14  
88293277689_0216  
17/03/31 20:56:16 INFO mapreduce.Job: The url to track the job: http://bde-bc5ed  
a-hadooper-computemaster-0-mapred.virt.local:8088/proxy/application_148829327768  
9_0216/  
17/03/31 20:56:16 INFO mapreduce.Job: Running job: job_1488293277689_0216  
17/03/31 20:56:21 INFO mapreduce.Job: Job job_1488293277689_0216 running in uber  
mode : false  
17/03/31 20:56:21 INFO mapreduce.Job:  map 0% reduce 0%  
17/03/31 20:56:28 INFO mapreduce.Job:  map 100% reduce 0%
```



RUN TESTING EXAMPLE

BIGGER EXAMPLE

- Create your input2 HDFS directory
- Deploy a copy of the following input:
`/home/NDBI048/mapreduce/input2/RomeoAndJuliet.txt`
- Run the MapReduce job
- Retrieve and browse the result
- Clean the output HDFS directory



USEFUL COMMANDS

mapred

mapred job -list all

- **List of all jobs**

mapred job -kill <job-id>

- **Kill a job**

mapred job -status <job-id>

- **Print status (counters)**



Testing example in Python

HADOOP MAPREDUCE PROGRAM

MAPPER.PY, REDUCER.PY

- **Create your own directory**

```
cd ~
```

```
mkdir -p mapreduce2
```

```
cd mapreduce2
```

- **Make a copy of the sample source files**

```
cp /home/NDBI040/mapreduce2/mapper.py .
```

```
cp /home/NDBI040/mapreduce2/reducer.py .
```



HADOOP MAPREDUCE PROGRAM

MAPPER.PY, REDUCER.PY

```
hadoop fs -rm -r /user/<login>/WordCount/output1/
```

- **Delete the output directory, if required**

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-  
*streaming*.jar
```

```
-files mapper.py, reducer.py
```

```
-mapper mapper.py
```

```
-reducer reducer.py
```

```
-input /user/<login>/WordCount/*
```

```
-output /user/<login>/WordCount/output1
```

- **Run the Python example using Hadoop Streaming**



HADOOP MAPREDUCE PROGRAM

MAPPER.PY, REDUCER.PY

```
hadoop fs -ls /user/<login>/WordCount/output1
```

```
rm result
```

```
hadoop fs -copyToLocal /user/<login>/WordCount/output1/part-00000 result
```

```
cat result
```

- **Check and print the result**
- **For Python experts, there are files `mapper2.py` and `reducer2.py` with a more optimized version**



REFERENCES

- MapReduce and Python: <https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>
- Writing An Hadoop MapReduce Program In Python: <https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>
- Hadoop MapReduce Tutorial: <https://hadoop.apache.org/docs/r2.6.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>
- HDFS commands: <https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-common/FileSystemShell.html>

