



Distributed File Systems, MapReduce

Seminar 1 of *NoSQL Databases* (PA195)

David Novak & Vlastislav Dohnal
Faculty of Informatics, Masaryk University, Brno

<http://disa.fi.muni.cz/vlastislav-dohnal/teaching/nosql-databases-fall-2019/>

Agenda



- MetaCentrum Hadoop **Cluster**
 - Login, **basic** tools, monitoring
- Hadoop Distributed File System
 - Basics, working with **files**, monitoring, **advanced** settings
- Hadoop MapReduce
 - Writing **own** MapReduce program: WordCount
 - **Running** on small data, monitoring
 - Running on **large** data
 - **Advanced** MapReduce task: Average Max Temperature
- Simple example in **Spark**

Basic Information



- We will be using **Hadoop** version 2.6.0
 - Hadoop **main** page: <http://hadoop.apache.org/>
 - **documentation** (v2.6.4):
<http://hadoop.apache.org/docs/r2.6.4/>
- MetaCentrum Hadoop cluster
 - MetaCentrum **account**:
<http://metavo.metacentrum.cz/en/application/index.html>
 - Hadoop cluster **access**:
<https://www.metacentrum.cz/en/hadoop/>
 - MetaCentrum Hadoop cluster **documentation**:
https://wiki.metacentrum.cz/wiki/Hadoop_documentation

Access from you home



- Due to having the seminars online
- Connect to the faculty's
 - <https://www.fi.muni.cz/tech/unix/vpn.html.en>
- In case of failure, use the university's VPN
 - <https://it.muni.cz/en/services/vpn>
 - Unfortunately not all computers @FI are accessible then
 - E.g., nymfe* are behind another firewall, so Putty/SSH port forwarding must be used.
 - https://www.akadia.com/services/ssh_putty.html



COVID19
Slide

Remote access to Nymfe*



- Nymfe* are local FI's computers located in a computer room
- Open an VNC connection (mainly for Win users)
 - <https://www.fi.muni.cz/tech/unix/nymfe-remote.html.en>
- or forward X11 connection when on Mac/Linux
 - `ssh -X nymfe0$((RANDOM % 2 + 1)).fi.muni.cz`
 - chooses randomly nymfe01-nymfe03

COVID19
Slide

MetaCentrum Hadoop Access



localhost

nymfe

- edit file on your local machine `~/ .ssh/config`
 - or for OpenSSH on Windows `C:\Users\<<login>\.ssh`

```
## MetaCentrum #####
```

```
Host hador
```

```
HostName hador.ics.muni.cz
```

```
User <login_in_MetaCentrum>
```

```
Port 22
```

- **log in** to Hadoop Cluster **frontend**

```
$ ssh hador
```

COVID19
Slide

MetaCentrum Hadoop Access



nymfe

- **Web interface:**

- Firstly, gain **Kerberos** ticket on your **local machine**:

```
scp login@hador.ics.muni.cz:/etc/krb5.conf .  
export KRB5_CONFIG=krb5.conf  
kinit <login>@META
```

- Chrome on the local machine (in the same terminal):

```
$ /opt/google/chrome/chrome --auth-server-whitelist="hador*.ics.muni.cz" &
```

- open <https://hador-c1.ics.muni.cz:9871/>

- or use **Firefox** and **configure** it using this manual:

https://wiki.metacentrum.cz/wiki/Hadoop_documentation#Web_accessibility

HDFS DFS (1)



hadoop

- HDFS system monitoring & basic commands

```
$ hdfs dfs -help
```

- [Documentation](#) of HDFS DFS file system commands

- get some data (complete Shakespeare's plays)

```
$ wget https://goo.gl/KyDfc7 -O shake.txt
```

```
$ hdfs dfs -put shake.txt
```

- or, alternatively

```
$ hdfs dfs -put shake.txt /user/<user>/shake.txt
```

```
$ hdfs dfs -ls
```


HDFS DFS (2)



hador

```
$ hdfs dfs -ls
```

```
$ hdfs dfs -setrep -w 2 shake.txt
```

```
$ hdfs dfs -rm shake.txt
```

```
$ hdfs dfs -D dfs.block.size=1048576 -put shake.txt
```

```
$ hdfs fsck /user/<user>/shake.txt -files -locations -  
blocks
```

```
$ hdfs dfs -mkdir input
```

Check HDFS files in browser

nymfe

<https://hador-c1.ics.muni.cz:9871/explorer.html#/user/<user>/>



Java Development

- Download the project from IS [seminar 1](https://is.muni.cz/auth/el/fi/podzim2020/PA195/um/seminar-1/)
<https://is.muni.cz/auth/el/fi/podzim2020/PA195/um/seminar-1/>

- `$ unzip pa195-hadoop-scaffolding.zip`

- Development with IntelliJ IDEA

```
$ module add jdk-1.8.0
```

```
$ module add idea-loc
```

```
$ idea.sh &
```

- Project is in Maven with dependencies:

```
org.apache.hadoop.hadoop-common
```

```
org.apache.hadoop.hadoop-mapreduce-client-core
```

- Compilation by Maven

```
$ mvn install
```

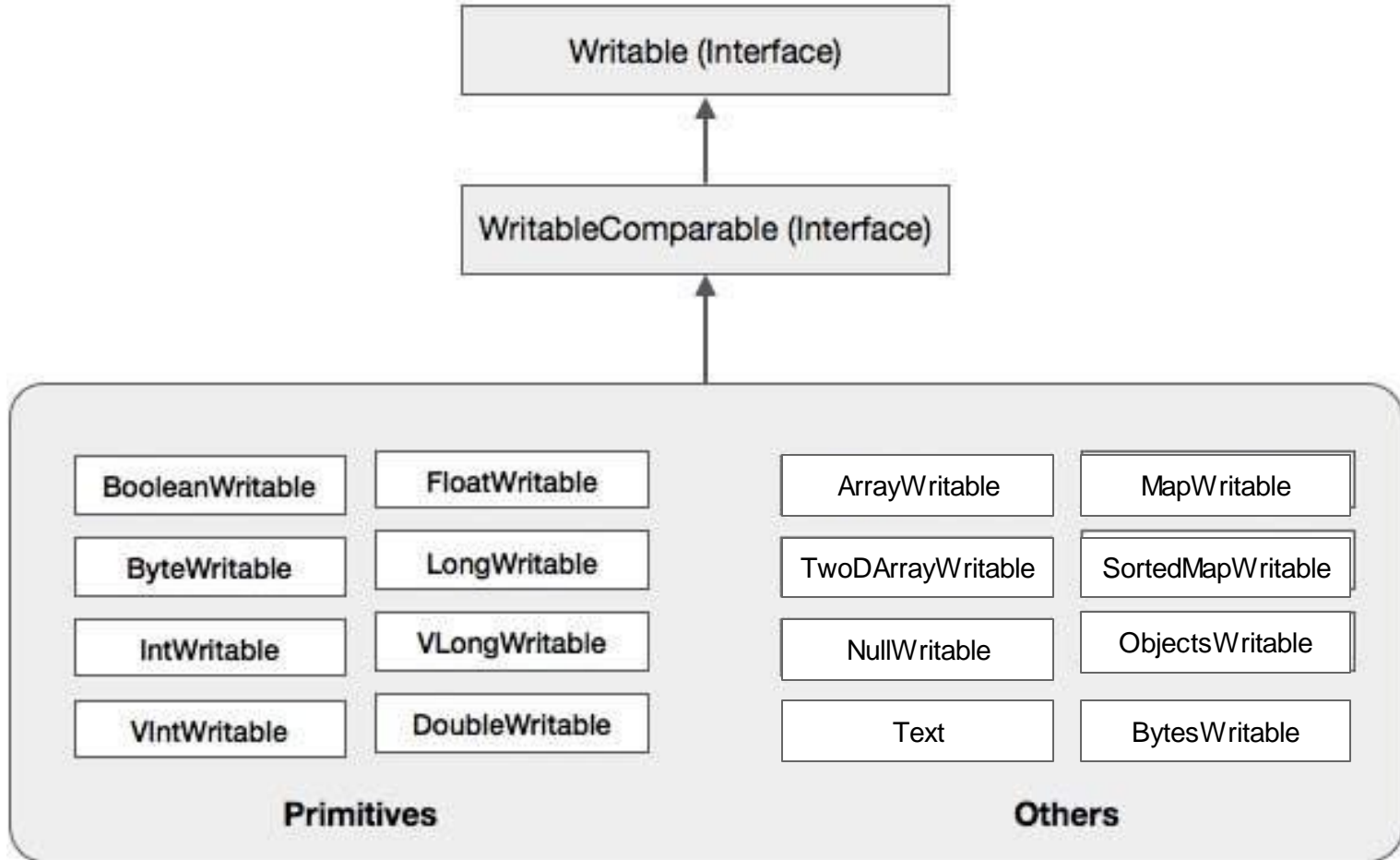
MapReduce: WordCount (1)



Task 1: Calculate **word frequency** in a document.

Sub-task 1.1: Use the **Hadoop Java** interface v2.6.0 to implement the WordCount as introduced in the lecture.

Hadoop Writable Classes



MapReduce: WordCount (2)



localhost

```
local$ module add maven
```

```
local$ mvn install
```

```
local$ scp target/pa195-hadoop-1.0.jar hador:
```

hador

```
hador$ hdfs dfs -mkdir input
```

```
$ hdfs dfs -mv shake.txt input
```

```
$ hadoop jar pa195-hadoop-1.0.jar  
pa195.hadoop.WordCount input/ output/
```

```
$ hdfs dfs -get output .
```

```
$ sort -k 2 -g -r part-r-00000 > sorted.txt
```

<https://hador-c2.ics.muni.cz:19890>

MapReduce: WordCount (3)



- **Sub-task 1.2:**

- try it with a **Combiner** and observe the **difference** in MapReduce log (output of the hadoop process)

- **Sub-task 1.3:**

- **clean** the input: remove characters `,.());:;!?-` and numbers

MapReduce: WordCount (3)



- **Sub-task 1.4:** do not lowercase the characters but ignore case when counting the words
- **Sub-task 1.5:** sort the results by the word frequency (descending)
 - use a second MapReduce job to do this

MapReduce: Large-scale Test



- **Task 2:** Run the WordCount (count & sort) on a **multi-GB collection** of documents
 - observe the **performance**
 - the actual output is not important
 - downloaded **Wikipedia** in

```
hadoop
```

```
$ DIR=/storage/brno2/home/dohnal/pa195/wikipedia  
$ hdfs dfs -mkdir wiki-input  
$ for F in $DIR/*.xml; do hdfs dfs -put $F wiki-  
input; done
```

- increase # of **reduce jobs**

Proof of practice



- Report the time it took to sort the Wiki data
 - copy & paste the line from [JobHistory](#) to 'wiki-sort.txt':
 - <https://hador-c2.ics.muni.cz:19890/jobhistory/app>
 - e.g.

2020.10.09 10:45:30 CEST	2020...	2020.10...	job_159...	word count	dohnal
root.dohnal SUCCEEDED	86	86	1	1	00hrs,
02mins, 30sec					
 - copy to the instructor's HDFS:

```
$ hdfs dfs -put wiki-sort.txt /user/dohnal/pa195nosql-seminar1/<login>-wiki-sort.txt
```

- ZIP your project in IntelliJ IDEA (src dir)
 - Upload to the IS's [vault](#):
 - https://is.muni.cz/auth/el/fi/podzim2020/PA195/ode/105893552/?pre_dmet=1324117



MapReduce: Weather Data



- **Task 3:** Find out the **average** maximum **temperature** for each month.

Data: historic temperatures in Milano (CSV format)

```
date,day-min,day-max
```

```
01012000,-4.0,5.0
```

```
02012000,-5.0,5.1
```

```
03012000,-5.0,7.7
```

```
04012000,-3.0,9.7
```

```
...
```

```
hadoop
```

```
$ /storage/brno2/home/dohnal/pa195/weather.csv
```

```
public static class MeanMapper extends Mapper<Object, Text, Text, DoubleWritable> {
```

```
    private final static int DATE = 0;  
    private final static int MAX = 2;  
    private final Map<Text, List<Double>> maxMap = new HashMap<>();
```

```
    @Override
```

```
    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {  
        // gets the fields of the CSV line  
        String[] values = value.toString().split(",");  
  
        // gets date and max temperature  
        Text month = new Text(values[DATE].substring(2));  
        Double max = Double.parseDouble(values[MAX]);  
  
        // if not present, put this month into the map  
        if (!maxMap.containsKey(month)) {  
            maxMap.put(month, new ArrayList<Double>());  
        }  
        // adds the max temperature for this day to the list of temperatures  
        maxMap.get(month).add(max);  
    }  
}
```

Is this correct?

```
    @Override
```

```
    protected void cleanup(Context context) throws IOException, InterruptedException {  
        // loops over the months collected in the map() method  
        for (Text month: maxMap.keySet()) {  
            List<Double> temperatures = maxMap.get(month);  
  
            // computes the sum of the max temperatures for this month  
            double sum = 0d;  
            for (Double max: temperatures) {  
                sum += max;  
            }  
            // emits the month and partial average  
            context.write(month, new DoubleWritable(sum / temperatures.size()));  
        }  
    }  
}
```

Weather: Partial Avg Example

source data:

```
01012000, -4.0, 10.0
02012000, -5.0, 20.0
03012000, -5.0, 2.0
04012000, -3.0, 4.0
05012000, -3.0, 3.0
```

Mapper #1: lines 1,2

Mapper #2: lines 3,4,5

Mapper #1 avg: $(10 + 20) / 2 = 15$

Mapper #2 avg: $(2 + 4 + 3) / 3 = 3$

Reducer avg: $(15 + 3) / 2 = 9$

Not correct!

Correct avg: $(10+20+2+4+3)/5 = 7.8$

This is correct

```
@Override
protected void cleanup(Context context) throws IOException, Interrupter

// loops over the months collected in the map() method
for (Text month: maxMap.keySet()) {

    List<Double> temperatures = maxMap.get(month);

    // computes the sum of the max temperatures for this month
    Double sum = 0d;
    for (Double max: temperatures) {
        sum += max;
    }

    // emits the month as the key and a SumCount as the value
    context.write(month, new SumCount(sum, temperatures.size()));
}
}
```

```
private final Map<Text, SumCount> sumCountMap = new HashMap<>();
```

```
@Override
public void reduce(Text key, Iterable<SumCount> values, Context context) throws IOException,
```

```
SumCount totalSumCount = new SumCount();
// loops over all the SumCount objects received for this month (the "key" param)
for (SumCount sumCount : values) {
    // sums all of them
    totalSumCount.addSumCount(sumCount);
}
// puts the resulting SumCount into a map
sumCountMap.put(new Text(key), totalSumCount);
}
```

```
@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
```

```
// loops over the months collected in the reduce() method
for (Text month: sumCountMap.keySet()) {
    double sum = sumCountMap.get(month).getSum().get();
    int count = sumCountMap.get(month).getCount().get();
    // emits the month and the mean of the max temperatures for the month
    context.write(month, new DoubleWritable(sum/count));
}
}
```

Spark: Simple Example



hador

- The MetaCentrum cluster has Spark installed: [doc](#)
- A simple example to count words in Shakespeare:

```
$ spark-shell --master yarn
```

```
scala> :help
```

```
scala> val file = sc.textFile("hdfs://hador-  
cluster/user/<login>/shake.txt")
```

```
scala> val counts = file.flatMap(line => line.split("  
")).map(word => (word, 1)).reduceByKey(_ + _)
```

```
scala> counts.saveAsTextFile("spark-output")
```

```
scala> :quit
```

```
$ hdfs dfs -get spark-output/
```

Lessons Learned & Cleanup



What **lessons** did we take from the following?

- Basic work with the **HDFS** distributed file system
- Hadoop **MapReduce** in Java
 - simple **word count** and its modifications
 - **large-scale** distributed job
 - distributed **average**

hadoop

- Clean the **large** files from both **HDFS** and the **your home dir** on the Hadoop Cluster, please

```
$ hdfs dfs -rm -R wiki-input/
```

```
$ hdfs dfs -rm -R output
```

Cleanup on Nymfe



- Log out of the Gnome session
 - it may take a while (cca 20 secs) to get *Log Out prompt*, so wait.
 - if it fails, exit x11vnc in the terminal window (by Ctrl-C) and run: `$ gnome-session-quit --force -logout`
- Check and kill remaining processes
 - `$ ps ux`
 - `$ kill <each_PID>`

COVID19
Slide