B4M36DS2, BE4M36DS2: **Database Systems 2**

https://cw.fel.cvut.cz/b231/courses/b4m36ds2/

Practical Class 3

MapReduce

Yuliia Prokop
prokoyul@fel.cvut.cz

9. 10. 2023

Author: Martin Svoboda
(martin.svoboda@matfyz.cuni.cz)

Czech Technical University in Prague, Faculty of Electrical Engineering

MapReduce Model

Map function

- *Input*: **an input key-value pair** (input record)
- *Output*: **a set of intermediate key-value pairs**
  - Usually from a different domain
  - Keys do not have to be unique
- $(key, value) \rightarrow$ list of $(key, value)$

Reduce function

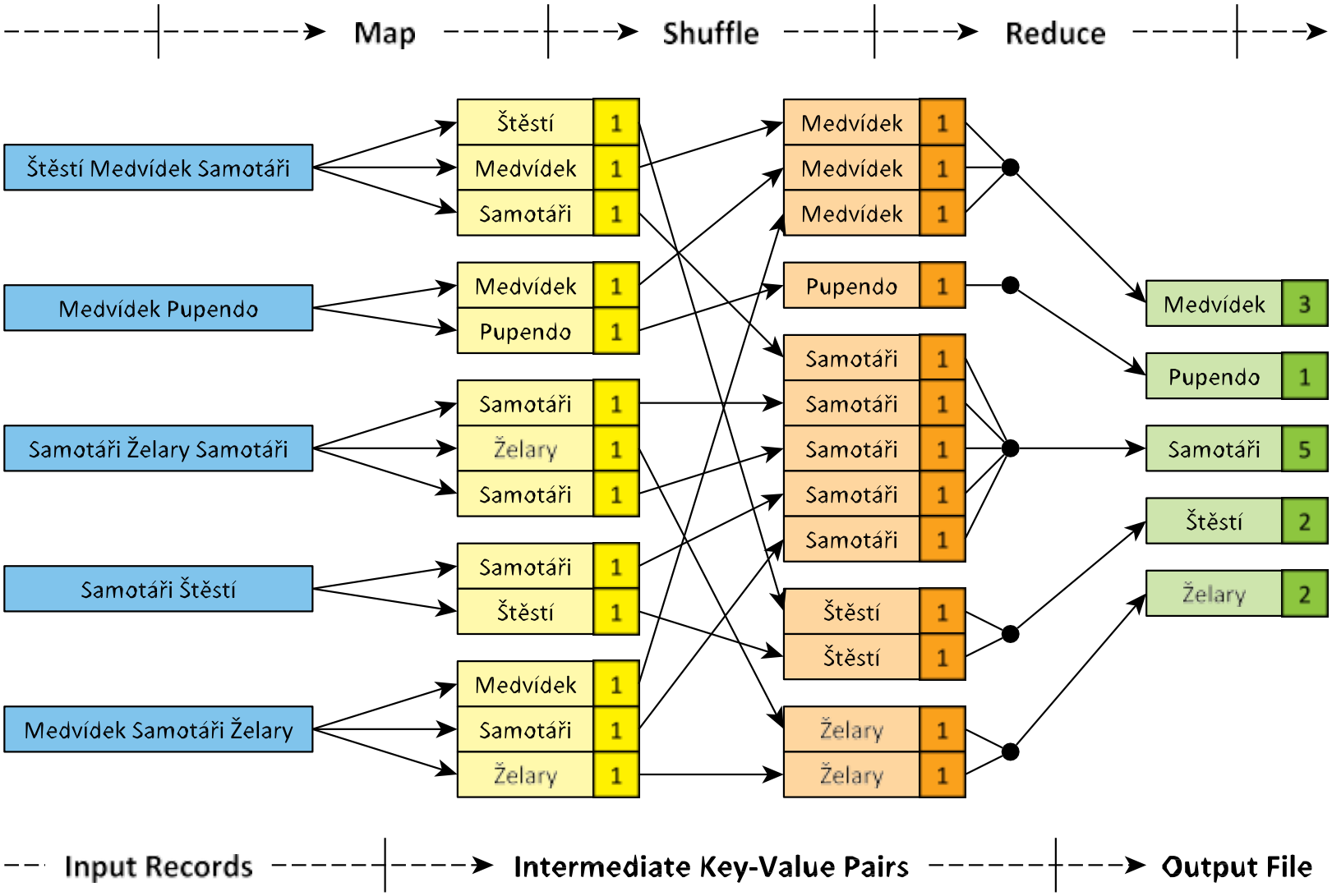
- *Input*: **an intermediate key + a set of (all) values** for this key
- *Output*: **a possibly smaller set of values** for this key
  - From the same domain
- $(key,$ list of $values) \rightarrow (key,$ list of $values)$

Example: Word Frequency

```
/**
 *  Map function
 *  @param key     Document identifier
 *  @param value Document contents
 */
map(String key,  String  value)  {  foreach
   word w in value: emit(w, 1);
}
```

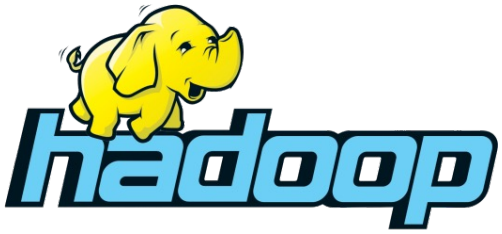
```
/**
 *  Reduce function
 *  @param key       Particular word
 *  @param values List of count values generated for this word
 */
reduce(String key, Iterator values) {  int
   result = 0;
   foreach v in values: result += v;
   emit(key, result);
}
```

Example: Word Frequency



Apache Hadoop

Open-source framework



- Hadoop Common
- Hadoop **Distributed File System** (HDFS)
- Hadoop Yet Another Resource Negotiator (YARN)
- Hadoop **MapReduce**

Server Access

Connect to our NoSQL server

- `ssh` and `sftp` on Linux
- **PuTTY** and **WinSCP** on Windows
- nosql.felk.cvut.cz

Login and password sent by e-mail

Change your initial password (if not yet changed)

- `passwd`

First Steps

Get familiar with basic Hadoop commands

- **hadoop**
  - Basic help for Hadoop commands
- **hadoop fs**
  - Distributed file system commands
- **hadoop jar**
  - Execution of MapReduce jobs

Browse the HDFS namespace

- hadoop fs -ls /
- hadoop fs -ls /user/
- hadoop fs -ls /user/login/

```
Found 2 items
drwxrwxrwt   - hadoop supergroup          0 2022-08-21 10:20 /tmp
drwxr-xr-x   - hadoop supergroup          0 2022-09-18 12:49 /user
```

Word Count Job

Create your working directory

- cd ~
- mkdir -p mapreduce/WordCount
- cd mapreduce/WordCount

Make a copy of the sample java source file

- cp /home/DS2/mapreduce/WordCount.java .

```
f201_student@nosql:~$ mkdir -p mapreduce/WordCount
f201_student@nosql:~$ cd mapreduce/WordCount
f201_student@nosql:~/mapreduce/WordCount$ cp /home/DS2/mapreduce/WordCount.java .
f201_student@nosql:~/mapreduce/WordCount$ ls -la
total 12
drwxrwxr-x 2 f201_student f201_student 4096 Nov  2 14:51 .
drwxrwxr-x 3 f201_student f201_student 4096 Nov  2 14:50 ..
-rw-rw-r-- 1 f201_student f201_student 2480 Nov  2 14:51 WordCount.java
f201_student@nosql:~/mapreduce/WordCount$
```

Word Count Job

Compile our Word Count implementation

- mkdir classes
- **javac** -classpath \
  /home/DS2/mapreduce/**hadoop-common-3.1.1.jar**:\
  /home/DS2/mapreduce/\
  hadoop-mapreduce-client-core-3.1.1.jar \
  -d classes/ WordCount.java
- **jar** -cvf WordCount.jar -C classes/ .

```
f201_student@nosql:~/mapreduce/WordCount$ mkdir classes
f201_student@nosql:~/mapreduce/WordCount$ javac -classpath /home/DS2/mapreduce/hadoop-common-
3.1.1.jar:/home/DS2/mapreduce/hadoop-mapreduce-client-core-3.1.1.jar -d classes/ WordCount.ja
va
f201_student@nosql:~/mapreduce/WordCount$ rm -r classes/
```

Word Count Job

Compile our Word Count implementation

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

```
f201_student@nosql:~/mapreduce/WordCount$ jar -cvf WordCount.jar -C classes/ .
added manifest
adding: WordCount$MyMapper.class(in = 1523) (out= 667)(deflated 56%)
adding: WordCount$MyReducer.class(in = 1633) (out= 686)(deflated 57%)
adding: WordCount.class(in = 1456) (out= 798)(deflated 45%)
f201_student@nosql:~/mapreduce/WordCount$ ls -la
total 20
drwxrwxr-x 3 f201_student f201_student 4096 Nov  2 14:58 .
drwxrwxr-x 3 f201_student f201_student 4096 Nov  2 14:50 ..
drwxrwxr-x 2 f201_student f201_student 4096 Nov  2 14:56 classes
-rw-rw-r-- 1 f201_student f201_student 2895 Nov  2 14:58 WordCount.jar
-rw-rw-r-- 1 f201_student f201_student 2480 Nov  2 14:51 WordCount.java
f201_student@nosql:~/mapreduce/WordCount$
```

Word Count Job

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/DS2/mapreduce/input1/movies.txt \
  /user/login/WordCount/input1

```
f201_student@nosql:~/mapreduce/WordCount$ hadoop fs -mkdir /user/f201_student/WordCount
f201_student@nosql:~/mapreduce/WordCount$ hadoop fs -mkdir /user/f201_student/WordCount/input
1
```

```
f201_student@nosql:~/mapreduce/WordCount$ hadoop fs -ls /user/f201_student/WordCount/
Found 1 items
drwxr-xr-x   - f201_student f201_student          0 2020-11-02 15:00 /user/f201_student/WordC
ount/input1
```

Word Count Job

…

Prepare the sample input data

- **hadoop fs -copyFromLocal** \
  /home/DS2/mapreduce/input1/movies.txt \
  /user/login/WordCount/input1

```
f201_student@nosql:~/mapreduce/WordCount$ hadoop fs -copyFromLocal /home/DS2/mapreduce/input1
/movies.txt /user/f201_student/WordCount/input1
f201_student@nosql:~/mapreduce/WordCount$
f201_student@nosql:~/mapreduce/WordCount$ hadoop fs -ls /user/f201_student/WordCount/input1/
Found 1 items
-rw-r--r--   1 f201_student f201_student        108 2020-11-02 15:03 /user/f201_student/WordC
ount/input1/movies.txt
```

Word Count Job

Run the prepared MapReduce job

- **hadoop jar** WordCount.jar WordCount \
  /user/login/WordCount/input1 \
  /user/login/WordCount/output1

```
f201_student@nosql:~/mapreduce/WordCount$ hadoop jar WordCount.jar WordCount /user/f201_stude
nt/WordCount/input1 /user/f201_student/WordCount/output1
2020-11-02 15:08:07,194 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2020-11-02 15:08:07,388 INFO client.AHSProxy: Connecting to Application History server at /0.
0.0.0:10200
2020-11-02 15:08:07,673 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsin
g not performed. Implement the Tool interface and execute your application with ToolRunner to
 remedy this.
2020-11-02 15:08:07,687 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path
: /tmp/hadoop-yarn/staging/f201_student/.staging/job_1603183763126_0157
2020-11-02 15:08:07,889 INFO input.FileInputFormat: Total input files to process : 1
2020-11-02 15:08:07,956 INFO mapreduce.JobSubmitter: number of splits:1
2020-11-02 15:08:07,991 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-p
ublisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
2020-11-02 15:08:08,089 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_160318376
3126_0157
2020-11-02 15:08:08,090 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-11-02 15:08:08,286 INFO conf.Configuration: resource-types.xml not found
2020-11-02 15:08:08,286 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2020-11-02 15:08:08,350 INFO impl.YarnClientImpl: Submitted application application_160318376
3126_0157
```

Word Count Job

```
i.cz:8088/proxy/application_1603183763126_0157/
2020-11-02 15:08:08,388 INFO mapreduce.Job: Running job: job_1603183763126_0157
2020-11-02 15:10:05,795 INFO mapreduce.Job: Job job_1603183763126_0157 running in uber mode :
 false
2020-11-02 15:10:05,796 INFO mapreduce.Job:  map 0% reduce 0%
2020-11-02 15:10:09,843 INFO mapreduce.Job:  map 100% reduce 0%
2020-11-02 15:10:15,882 INFO mapreduce.Job:  map 100% reduce 100%
2020-11-02 15:10:15,887 INFO mapreduce.Job: Job job_1603183763126_0157 completed successfully
2020-11-02 15:10:15,994 INFO mapreduce.Job: Counters: 53
        File System Counters
                FILE: Number of bytes read=76
                FILE: Number of bytes written=429821
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=240
                HDFS: Number of bytes written=50
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
```

Word Count Job

Retrieve and explore the job result

- **hadoop fs -copyToLocal** \
  /user/login/WordCount/output1/part-r-00000 \
  result.txt
- **cat** result.txt

Clean the output HDFS directory

- **hadoop fs -rm -r** /user/login/WordCount/output1/

```
f201_student@nosql:~/mapreduce/WordCount$ hadoop fs -copyToLocal /user/f201_student/WordCount
/output1/part-r-00000 result1.txt
f201_student@nosql:~/mapreduce/WordCount$ cat result1.txt
Medvidek        3
Pupendo 1
Samotari        5
Stesti  2
Zelary  2
```

Bigger Word Count Job

Run our MapReduce job on a bigger input file

- Create your `input2` HDFS directory
- Deploy a copy of the following input file `/home/DS2/mapreduce/input2/RomeoAndJuliet.txt`
- Run the MapReduce job
- Retrieve and browse the result
- Clean the output HDFS directory

Useful Commands

Additional MapReduce commands that might be helpful

- `mapred job –list all`
  - Lists identifiers of all the MapReduce jobs
- `mapred job -status job-id`
  - Prints status counters for a given MapReduce job
- `mapred job -kill job-id`
  - Kills a particular MapReduce job

MapReduce Project

Download the following Hadoop libraries

- /home/DS2/mapreduce/
  hadoop-common-3.1.1.jar
- /home/DS2/mapreduce/
  hadoop-mapreduce-client-core-3.1.1.jar

Choose your preferred Java IDE

- NetBeans
- IntelliJ IDEA

NetBeans Project

Launch NetBeans IDE and create a new project

- Select *Java application* as a project type
  - Let the main class to be created automatically
  - Do not use explicit packages
- Add both the Hadoop libraries into the project
  - Use *Add JAR/Folder* in the project context menu
- Replace the contents of the main class with our pattern
  - `/home/DS2/mapreduce/WordCount.java`
  - Change the name of the class appropriately

Build the project to create a *jar* distribution

IntelliJ IDEA Project

Launch IntelliJ IDEA and create a new project

- Select *Java* as a project type
  - Do not include any additional libraries or frameworks
- Add both the Hadoop libraries into the project
  - Open *Project Structure* in the *File* menu
  - Select *Modules* at the left panel and then *Dependencies* tab
  - Click *+* on the right and select *JARs or directories*
- Add a new Java class into the project
  - Replace its contents with the sample pattern
  - `/home/DS2/mapreduce/WordCount.java`
  - Change the name of the class appropriately

Build the project to create a *jar* distribution

Java Interface

Mapper class

- Implementation of the **map function**
- Template parameters
  - `KEYIN`, `VALUEIN` – types of input key-value pairs
  - `KEYOUT`, `VALUEOUT` – types of intermediate key-value pairs
- Intermediate pairs are emitted via `context.write(k, v)`

```
class MyMapper extends Mapper<KEYIN, VALUEIN, KEYOUT,
   VALUEOUT> {  @Override
   public void map(KEYIN key, VALUEIN value, Context context)
      throws  IOException, InterruptedException
   {
      // Implementation
   }
}
```

Java Interface

Reducer class

- Implementation of the **reduce function**
- Template parameters
  - `KEYIN`, `VALUEIN` – types of intermediate key-value pairs
  - `KEYOUT`, `VALUEOUT` – types of output key-value pairs
- Output pairs are emitted via `context.write(k, v)`

```
class MyReducer extends Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT>{
  @Override
  public void reduce(KEYIN key, Iterable<VALUEIN> values, Context context)
    throws IOException, InterruptedException
  {
    // Implementation
  }
}
```

Inverted Index

Implement an *inverted index* using MapReduce

- Use input files in /home/DS2/mapreduce/input3/
- Produce a list of *file:occurrences* pairs for each word
  - E.g.: `Samotari file1:1 file3:2 file4:1 file5:1`
- Use `((FileSplit)context.getInputSplit()).getPath().getName();` to access input file names
- Use `Map<String, Integer> map = new HashMap<>();` to process intermediate key-value pairs
- Use `map.entrySet()` to iterate over map entries

Compile, deploy and run the job…

References

HDFS: File System Shell commands

- https://hadoop.apache.org/docs/r3.1.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

MapReduce: tutorial

- https://hadoop.apache.org/docs/r3.1.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

MapReduce: shell commands

- https://hadoop.apache.org/docs/r3.1.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapredCommands.html

MapReduce: JavaDoc

- https://hadoop.apache.org/docs/r3.1.1/api/