Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel

Exercise Sheet 10
Big Data Analytics
WiSe 2017/2018
Due Date: 2018-01-19, 390 P

Please document your code, without sufficent documentation you won't recive any points.

1 Data-Flow Programming (Python) (180 P)

In this exercise, you will implement a simplified data-flow programming model in Python.
The language should support the following operators:

- $read(file)$, provides the content of a (trivial) CSV file as a list of tuples
- $write(file)$, writes the output into the file
- $filter(function)$, keeps the list of tuples for which $function(t)$ returns true
- $map(function)$, transforms one tuple for each input tuple
- $flatmap(function)$, transforms each input tuple into an arbitrary number of tuples
- $group(function)$, groups into a hashmap (key is the group name) containing the list of tuples for each group
- $reduce(function)$, applies the reduction function to the group returning a single tuple (function is called for each group)
- $join(y, function)$, joins the data with the data from y by calling $function(x, y)$

These operators should be chainable with the functional programming paradigm. An example program using your class could be:

```
d = dataflow.read("file.csv")
bd = d.map(lambda x : (x[0], x[1]) ).filter(lambda x: x[0] == "big data")
bd.write("out.csv")

z = d.group(lambda x: x[0])
r = z.reduce(lambda x : sum([y[0] for y in x]))
print(r)
```

1.1 Hints

Implement a class keeping the list of tuples internally, each operator should return a new class instance while updating the changed data. Note that there are several implementations possible (particularly for join) which result in more or less complex code to write to the user of this library.
Implement the `__str__()` method for the class to allow convenient printing of the class.

Submission:

`1-data-flow.(py|pdf)` Your (commented) data-flow implementation with a few examples or a lab notebook using Jupyter.

2 Pig Basic & User Defined Functions (120 P)

In this task, basics of Pig are illustrated on simple examples. Write individual scripts to perform the following tasks:

- Read the file `/user/bigdata/10/numbers` and compute the mean of the contained numbers.
- Read `/user/bigdata/10/students` and `/user/bigdata/10/semesters`. For each student, compute the numbers of semesters and store it as a separate file. The header is part of the file, filter it. Visualize the data flow of your script using a pipe diagram (including examples).
- Read the file `/user/bigdata/10/1998_FIFA_World_Cup.txt` and replace the occurrence of numbers from 1 to 100 by a textual representation, e.g., 5 becomes five. Provide a user defined function in Python to perform the replacement. Save the article under a new file in your directory on HDFS.
- Read an arbitrary file and perform the wordcount task.

2.1 Hints

The input file for the FIFA World Cup may need preprocessing before it can be processed effectively.
User defined functions are documented here: `http://pig.apache.org/docs/r0.15.0/udf.html#python-udfs`.

Submission:

`2-compute-avg.pig`	The Pig script to compute the average.
`2-count-semester.pig`	The Pig script to compute the student semesters.
`2-pipe-diagram.pdf`	The pipe diagram visualizing the `count-semester.pig` script.
`2-spell-out-numbers.pig`	The script for replacing the numbers with text.
`2-spell-out-udf.py`	The UDF for replacing numbers with text (for the script spell-out-numbers.pig).
`2-word-count.py`	The script for word count.

3 Performance Analysis (Theory) (90 P)

Assessing performance of big data applications is important to understand limitations. In this task, you estimate best case performance for analyzing the full Wikipedia data.
Consider the task to perform word count on an article basis and summarize the total word count. How long would you estimate for the phases of I/O, compute and communication? How well could these phases be overlapped?

3.1 Hints

The Abu nodes utilize Gigabit Ethernet and are equipped with one HDD per node providing roughly 100 MiB/s. A node hosts 4 sockets each equipped with a 12 core AMD CPU @2.6 GHz.

Submission:

`3-analysis.pdf`	Your performance analysis.