VRE for regional Interdisciplinary communities in Southeast Europe and the Eastern Mediterranean

VI-SEEM data analysis service

Petar Jovanovic
Institute of Physics Belgrade

The VI-SEEM project initiative is co-funded by the European Commission under the H2020 Research Infrastructures contract no. **675121**

Agenda

- Apache Hadoop
- Hadoop basic component
- HDFS
- MapReduce
- Word count example with mvn archetype from VI-SEEM code repo
- Hadoop streaming
- More examples

Apache Hadoop

- Open-source programming framework that supports processing and storing large data sets in a distributed computing environment.
- Based on Google's MapReduce and Google File System papers
  - https://research.google.com/archive/mapreduce.html
  - https://research.google.com/archive/gfs.html
- Evolved from an effort to build an open source search engine (Nutch)
- Hadoop ecosystem

Hadoop: basic components

HDFS

- A distributed file system
- Cuts files into chunks
- Stores chunks across multiple machines with multiple copies
  - helps with fault tolerance and access speeds

MapReduce

- A basic approach to programming data analysis workflows in Hadoop.
- A batch system of execution

Two types of nodes:

- Name node
- Data node

HDFS file system commands

(https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html)

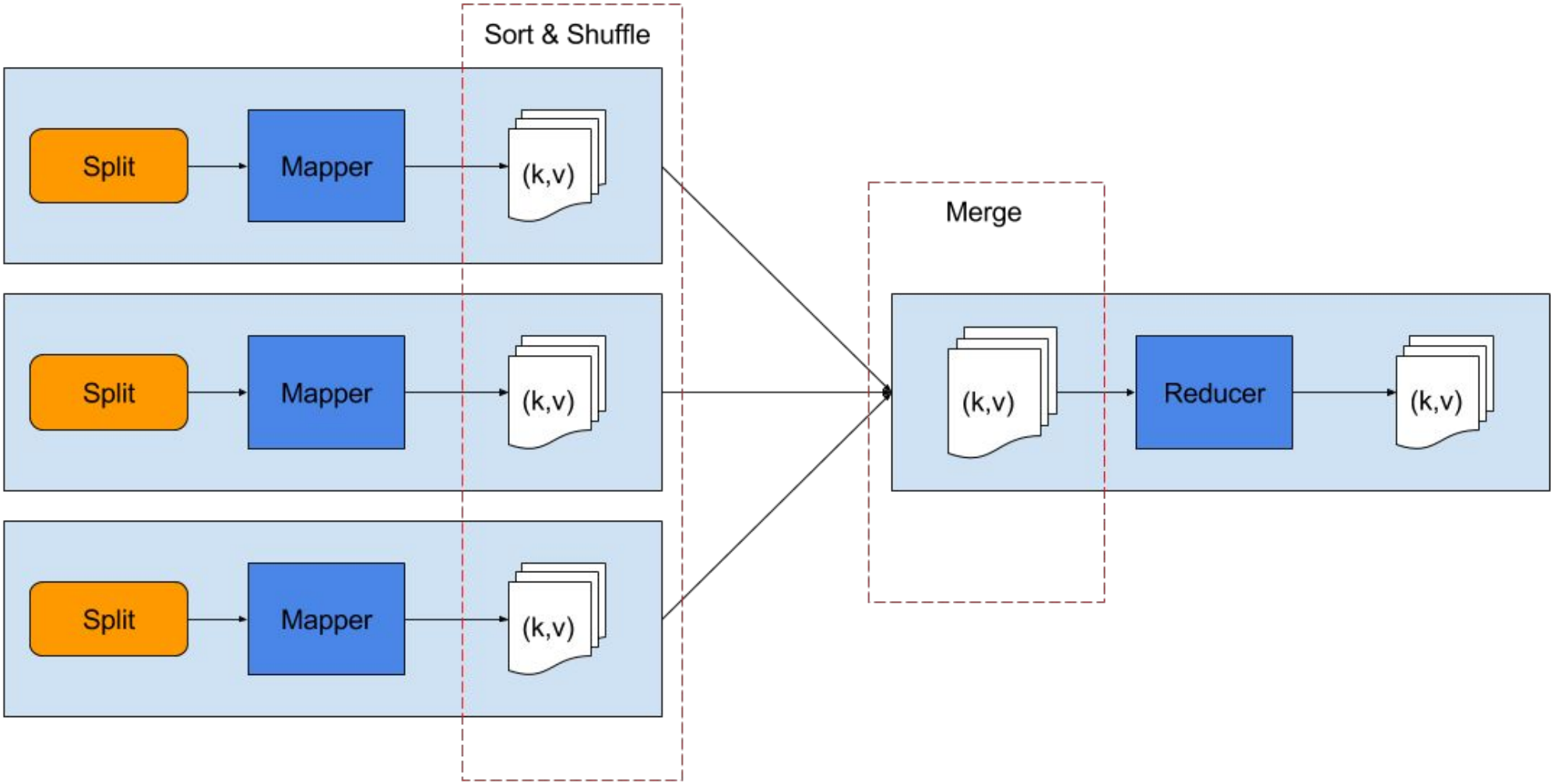
- list files in the current directory:
  ```
  hdfs dfs -ls
  ```
- put a file into HDFS:
  ```
  hdfs dfs -put data_file.dat
  ```
- get a file from HDFS:
  ```
  hdfs dfs -get data_file.dat
  ```

MapReduce

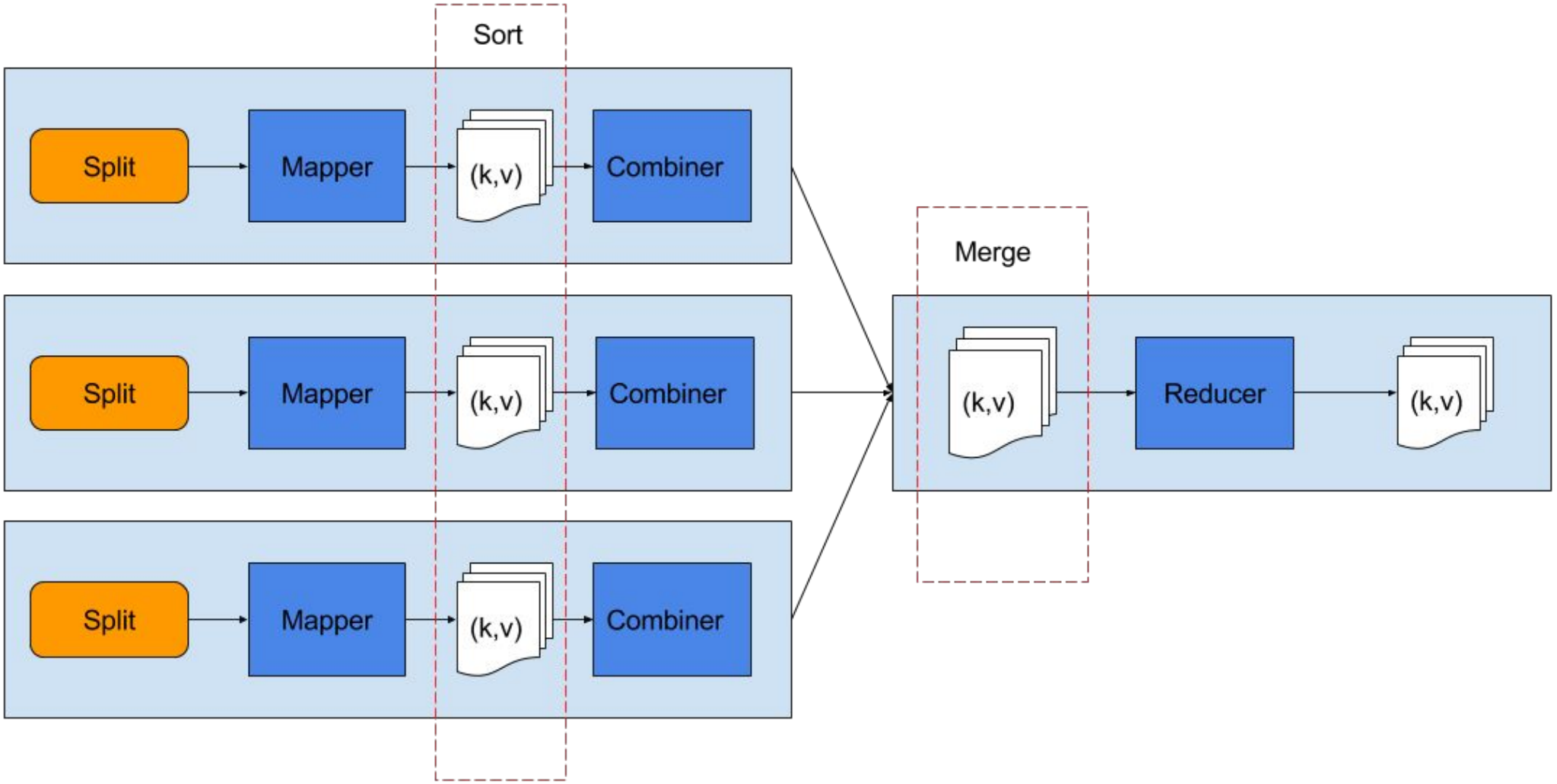
- Works on data as key-value pairs
- 2 step approach:
  - Map is a function applied to every record in a dataset, which translates it into a key-value pair
  - Reduce is a function that combines key-value pairs in some meaningful way and calculates results on them (e.g. word frequencies in a language corpus)
- Between map and reduce step, key-value pairs are sorted by key.
- MapReduce in Unix commands:

```
$ cat input.dat | mapper | sort | reducer > output.dat
```

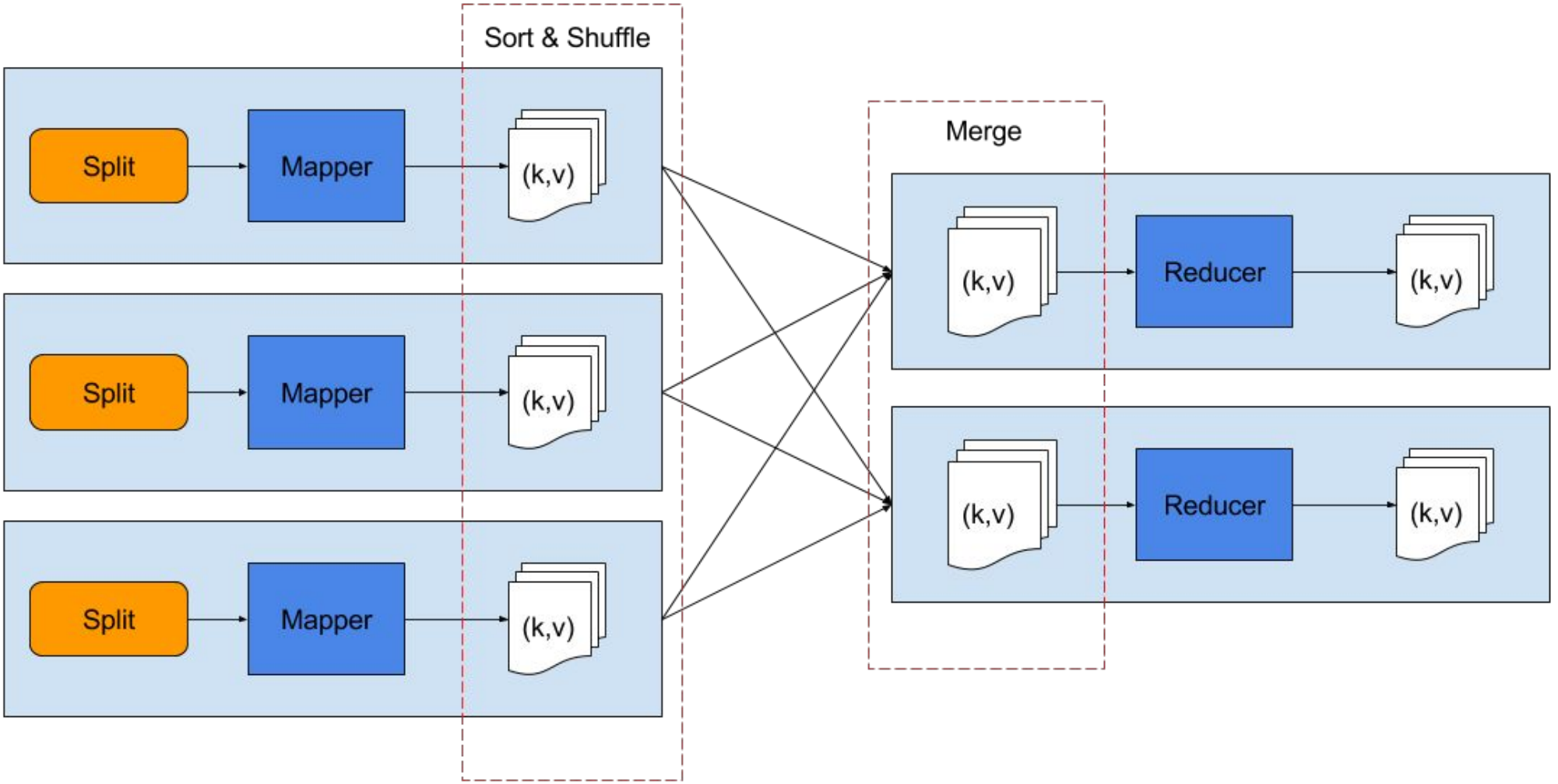
MapReduce - basic



MapReduce - with combiner



MapReduce - multiple reducers



Word count example

- Mapper
  - split input lines into words
  - add 1 as initial count of each word
- Reducer
  - for each word (key), sum its counts (value)

- Combiner
  - same as reducer, but runs locally on mapper outputs
  - commutative and associative operation
  - useful when it can reduce the mapper output

- Maven project archetype in VI-SEEM code repository: https://code.vi-seem.eu/petarj/hadoop-archetype

Hadoop streaming (1)

- MapReduce in Unix commands:
  ```
  $ cat input.dat | mapper | sort | reducer > output.dat
  ```
- Hadoop streaming pipes input through standard input to the mapper, whose output is sorted and then piped to reducer.
- Inputs are processed line by line, and each line is a string
  - the mapper and the reducer are responsible for parsing the input lines
  - slight difference to the Java api where reducer gets a list of values associated with a key

Hadoop streaming (2)

- Command to execute:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar \
      -files mapper_executable,reducer_executable[,combiner_executable] \
      -mapper mapper_executable \
      [-combiner combiner_executable] \
      -reducer reducer_executable \
      -input input_data \
      -output output_data
```

- Combiner is optional
- To specify more than one reducer add parameter:

```
-D mapreduce.job.reduces=42
```

Word count with straming

- Getting the code

```
$ git clone https://code.vi-seem.eu/petarj/hadoop-training.git
```

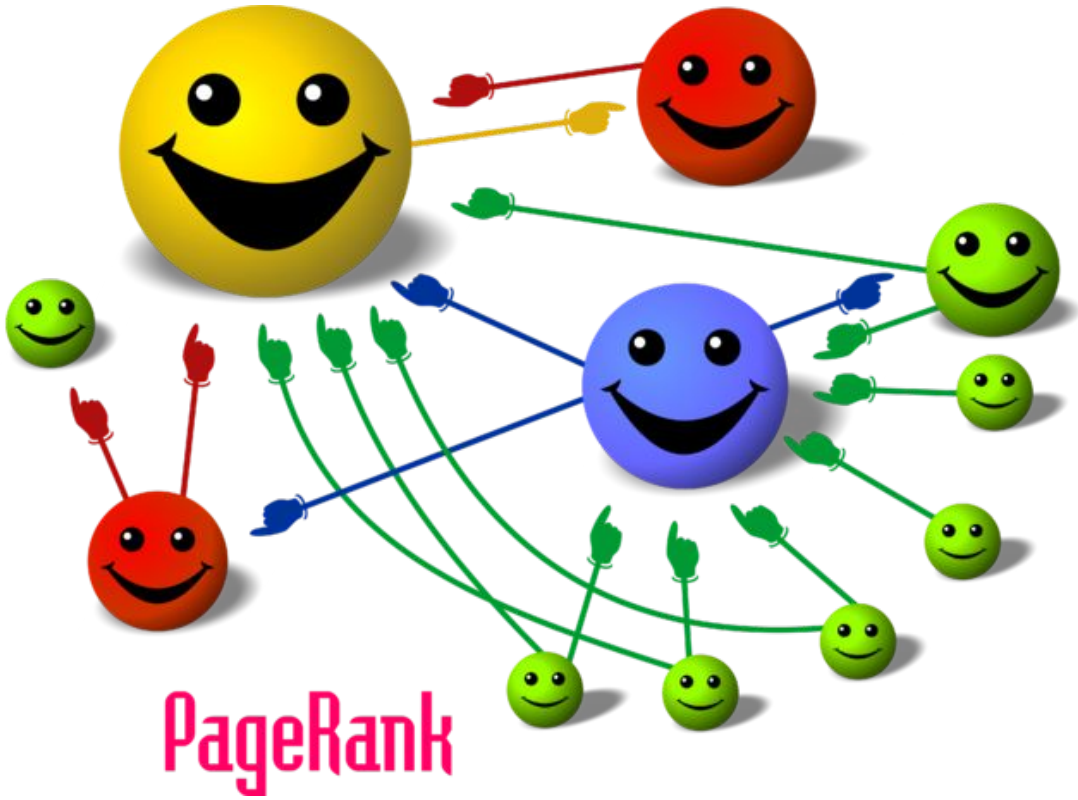
- Executing

```
$ cd hadoop-training/wordcount
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar \
          -files=mapper.py,reducer.py \
          -mapper mapper.py \
          -combiner reducer.py \
          -reducer reducer.py \
          -input /user/petarj/train_v2.txt
          -output wcout
```

Twitter user ranking (1)

Goal: find influential Twitter users

- Simple PageRank algorithm:



PR(u) - page rank of u

L(v) - number of out edges of v

Bu - set of nodes adjacent to u

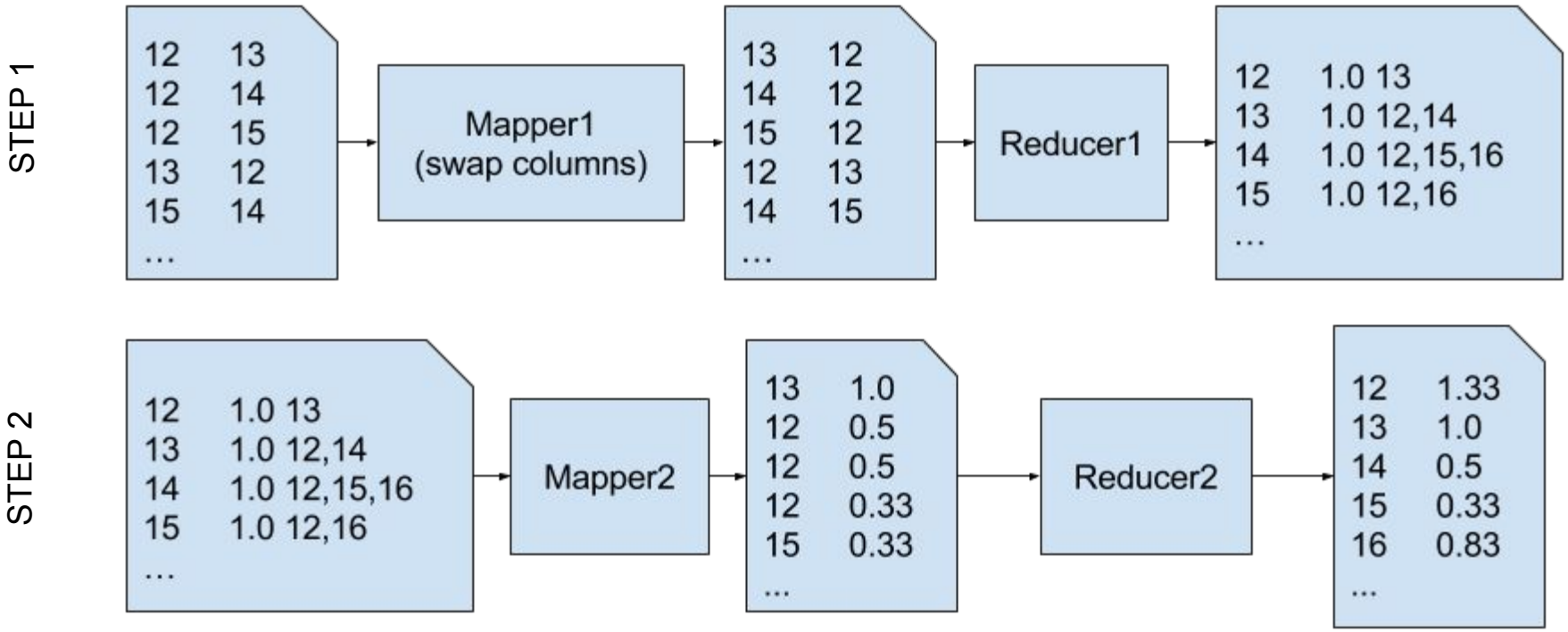
u & v - any two distinct nodes

In our case: *L(v)* is the number of people *v* is following, *Bu* are all the people *u* is following.

Twitter user ranking (2)

- Input data:
  - Text file: */user/petarj/twitter_rv.net* (26 GB)
  - Each line contains 2 numbers separated by tabs
  - First column is id of a user, second column is an id of his follower
- Two map-reduce steps needed to perform the analysis:
  - Convert data from edge list to adjacency list and set initial PageRank
  - Compute the PageRank approximation
- Running multiple reducers

Twitter user ranking (3)



Twitter demo

Getting the code

```
$ git clone https://code.vi-seem.eu/petarj/hadoop-training.git
```

Executing

```
$ cd hadoop-training/twitterrank
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar \
          -D mapreduce.job.reduces=20
          -files=mapper1.py,reducer1.py \
          -mapper mapper1.py \
          -reducer reducer1.py \
          -input /user/petarj/twitter_rv.net
          -output twrank_step1
$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar \
          -D mapreduce.job.reduces=6
          -files=mapper2.py,reducer2.py \
          -mapper mapper2.py \
          -reducer reducer2.py \
          -input twrank_step1
          -output twrank_step2
```

Questions

Thank you for your attention.