# Tutorial for Assignment 2.0

Florian Klien & Christian Körner

# IMPORTANT

- The presented information has been tested on the following operating systems
  - Mac OS X 10.6
  - Ubuntu Linux
- The installation on Windows machines will not be supported by us in the newsgroup

# Today's Agenda

- Motivation
- Quick introduction into Map/Reduce and Hadoop
- The assignment
- Pitfalls during the setup

# What you should have learned so far

- Network analysis and operations
  - such as degree distribution
  - Clustering Coefficient
  - Google's PageRank
  - Network Evolution
- Computed for very small networks

# Motivation

- So far these analyzes do NOT scale - What about networks which contain millions of nodes and edges or GB/TB of data?
- Computation would take quite a long time
- How can we process large amounts of data?

# Apache Hadoop - One solution of the scaling problem

- Uses the Map/Reduce paradigm
- Written in Java
  - But also other programming languages are possible
- Is used by Yahoo, Amazon etc.

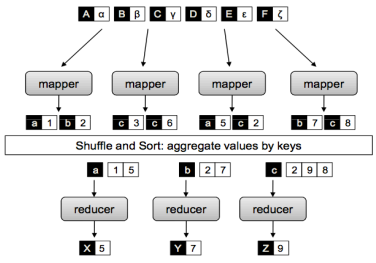
# What is Map/Reduce? /1

- Framework to support distributed computing of large data sets on clusters
- Used for data-intensive information processing
- Large Files/Lots of computation

# What is Map/Reduce? / 2

Abstract view:

- Master splits problem in smaller parts
- Mappers solve sub-problem
- Reducer combines results from Mappers
- Examples:
  - WordCount
  - Inverted Index



## Distributed File System (DFS)

- Hadoop comes with a distributed file system (HDFS)
- Highly fault tolerant
- Splits data in blocks of 64mb (default configuration)

## Example of a Map/Reduce Application /1

- Word Count - counting occurrences of words in lots of documents
- To keep things simple we will use the example from [1] which uses Python, reads from StdIn and writes to StdOut

# Example of a Map/Reduce Application / 2

```
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

# Example of a Map/Reduce Application / 3

- Example Code - Reducer

```
#!/usr/bin/env python

from operator import itemgetter
import sys

# maps words to their counts
word2count = {}

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
        word2count[word] = word2count.get(word, 0) + count
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        pass

# sort the words lexigraphically;
#
# this step is NOT required, we just do it so that our
# final output will look more like the official Hadoop
# word count examples
sorted_word2count = sorted(word2count.items(), key=itemgetter(0))

# write the results to STDOUT (standard output)
for word, count in sorted_word2count:
    print '%s\t%s'% (word, count)
```

## Example of a Map/Reduce Application / 4

- Testing the code you have written on a small subset is always recommended!
  - Example:
    - cat subset.txt | python mapper.py | python reducer.py
- Run the code on the cluster by issuing:
  - bin/hadoop jar contrib/streaming/hadoop-0.20.0-streaming.jar -file /home/hadoop/mapper.py -mapper / home/hadoop/mapper.py -file /home/hadoop/reducer.py -reducer /home/hadoop/reducer.py -input $input -output $output

# The Assignment

- Team up in groups of 5 students
- Create Subversion repository
- Implement TunkRank and compute it on the provided data (one iteration is sufficient)
- Hand in your used source code and the top 10.000 twitter users in descending order
- See assignment document on submission details!

# Provided Data

- You are given a subset of a large twitter data set which was gathered for a scientific paper [2]
  - compressed 530mb
- Tab separated:
  - First column: User
  - Second column: Follower (user who follows user from first column)

# TunkRank

- Tool to measure the influence on Twitter
- The higher the TunkRank score is the more influential a Twitter user is
- Twitterers with high TunkRank:
  - Barack Obama
  - Kevin Rose
  - Steven Colbert
- see http://www.tunkrank.com or [3] for details

$$Influence(X) = \sum_{Y \in Followers(X)} (1 + p * Influence(Y))/||Following(Y)||$$

# Hand In / 1

- Create a Subversion repository on the TUG server
  - name: WSWT10_<GROUPNAME>
  - Group members as members
  - Teaching assistents as readers

# Hand In / 2

Structure of the repository

- report.pdf (short! - approx. 1 page)
- bash scripts (optional)
- python/
  - mapper_1.py
  - mapper_2.py
  - ...
  - readme.txt
- results/
  - tunkrank_run_1.txt (top 10000 twitterers in descending order + their Tunkrank score)

# Important Dates

- NOW: Team up in groups of 5
- Assignment is due: Friday, JUNE 18th
  - 12:00 (noon) - soft deadline
  - 24:00 - hard deadline
- “Abgabegespräche” will be on JUNE 22nd
  - Every team member has to participate

# Hadoop Setup / 1

- create new user “hadoop” on your system
- use functioning DNS or /etc/hosts file for client/master lookup
- Download current Hadoop distribution from http://hadoop.apache.org/
- unpack distribution in a directory (e.g. /usr/local/hadoop/)
- create temp directory (e.g. /usr/local/hadoop-datastore)

# Hadoop Setup / 2

- conf/hadoop-env.sh - holds environment variables and java installation
- conf/core-site.xml - names the host the default file system & temp data
- conf/mapred-site.xml - specifies the job tracker
- conf/masters - names the masters
- conf/slaves (only on master nescessary) - names the slaves
- conf/hdfs-site.xml - specifies replication value

# Hadoop Setup / 3

- Format DFS
  - `bin/hadoop namenode -format`

# Starting the Hadoop Cluster

- bin/start-dfs.sh starts HDFS daemons
- bin/start-mapred.sh - starts Map/ Reduce daemons
- alternative: start-all.sh

- stopper scripts also available

# Pitfalls for the Setup of Hadoop

- Use machines of approximately the same speed / setup
- Use the same directory structure for all installations of your machines
- Ensure that password-less ssh login is possible for all machines
- Avoid the term localhost and the ip 127.0.0.1 at all cost --> use fixed IPs or functioning DNS for your experiments
- Read the Log files of the Hadoop installation
- Use the web interface of your cluster
- If there are problems --> use the newsgroup

# Thanks for your attention!

- Are there any questions?

# References


- [1] Michael G. Noll's Hadoop Tutorial:
  - Single Node Cluster
    - http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_%28Single-Node_Cluster%29
  - Multi Node Cluster
    - http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_%28Multi-Node_Cluster%29
  - Writing Map/Reduce Program in Python
    - http://www.michael-noll.com/wiki/Writing_An_Hadoop_MapReduce_Program_In_Python
- [2] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In WWW '10: Proceedings of the 19th international conference on World wide web, pages 591–600, New York, NY, USA, 2010. ACM.
- [3] http://thenoisychannel.com/2009/01/13/a-twitter-analog-to-pagerank/