universität koblenz
weiter:denken

Big Data
Session 4: Distributed Data Processing

Frank Hopfgartner
Institute for Web Science and Technologies

Last week

- Part 1:
  - Column stores (Introduction)
  - Big Table
  - HBase and Hive
- Part 2:
  - Job Scheduling
  - Coordination

Motivation


We FINALLY GOT THE RESULTS
OF YOUR DATA QUERY.
SORRY IT TOOK SO LONG.
Solve your biggest data challenge, quickly.
IBM
marketoonist
#WatsonAnalytics

Intended Learning Outcomes

universität koblenz
weiter:denken

At the end of this lecture, you will be able to:

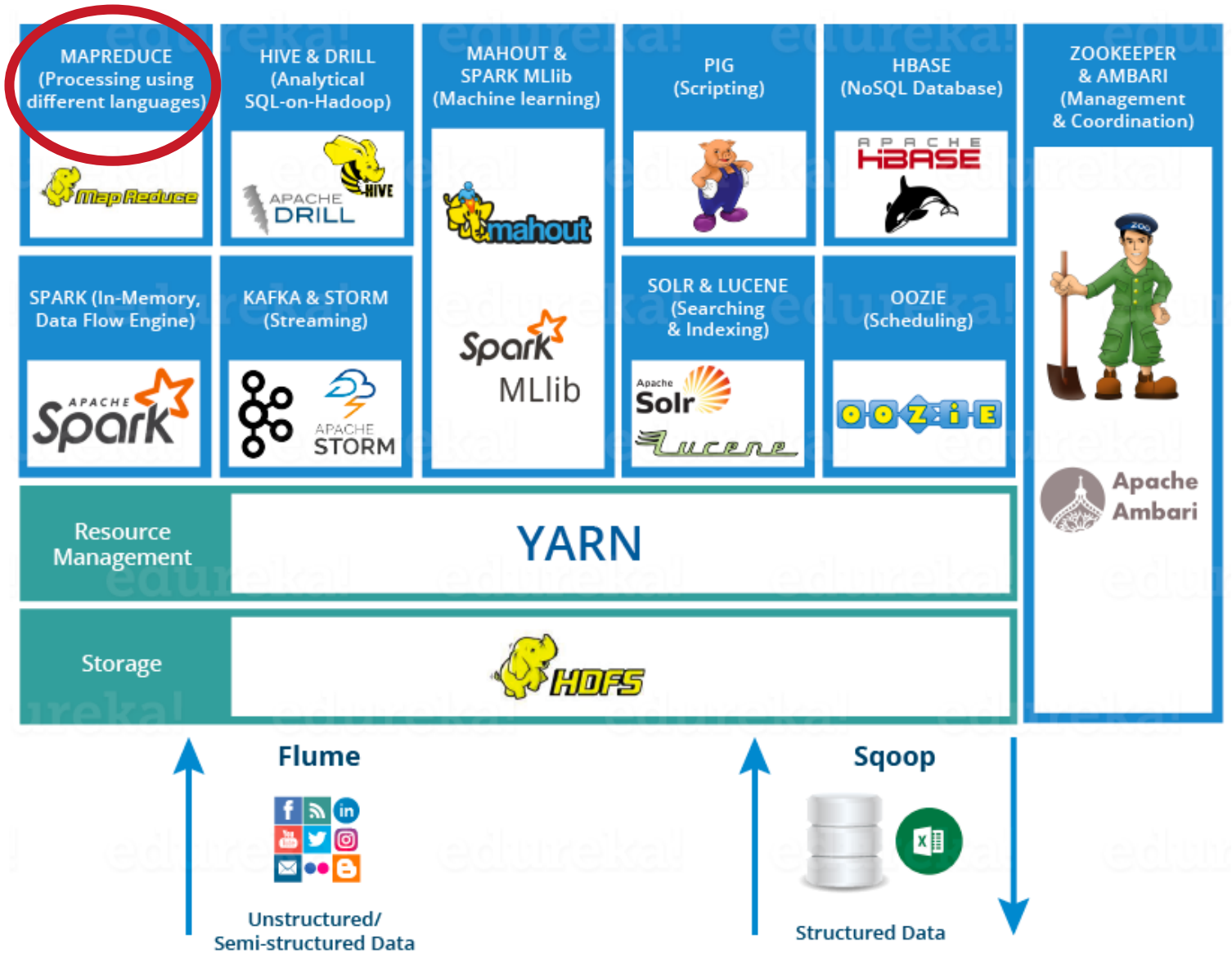
- Illustrate how to scale processing tasks in a Hadoop cluster using MapReduce
- Describe popular querying approaches
- Outline how to employ machine learning on big data

Outline

universität koblenz
weiter:denken

- Map/Reduce framework
- Querying
  - Spark Core API
  - Pig, Pig Latin
- Machine Learning at Scale
  - Spark MLLib
  - Mahout

Hadoop Ecosystem

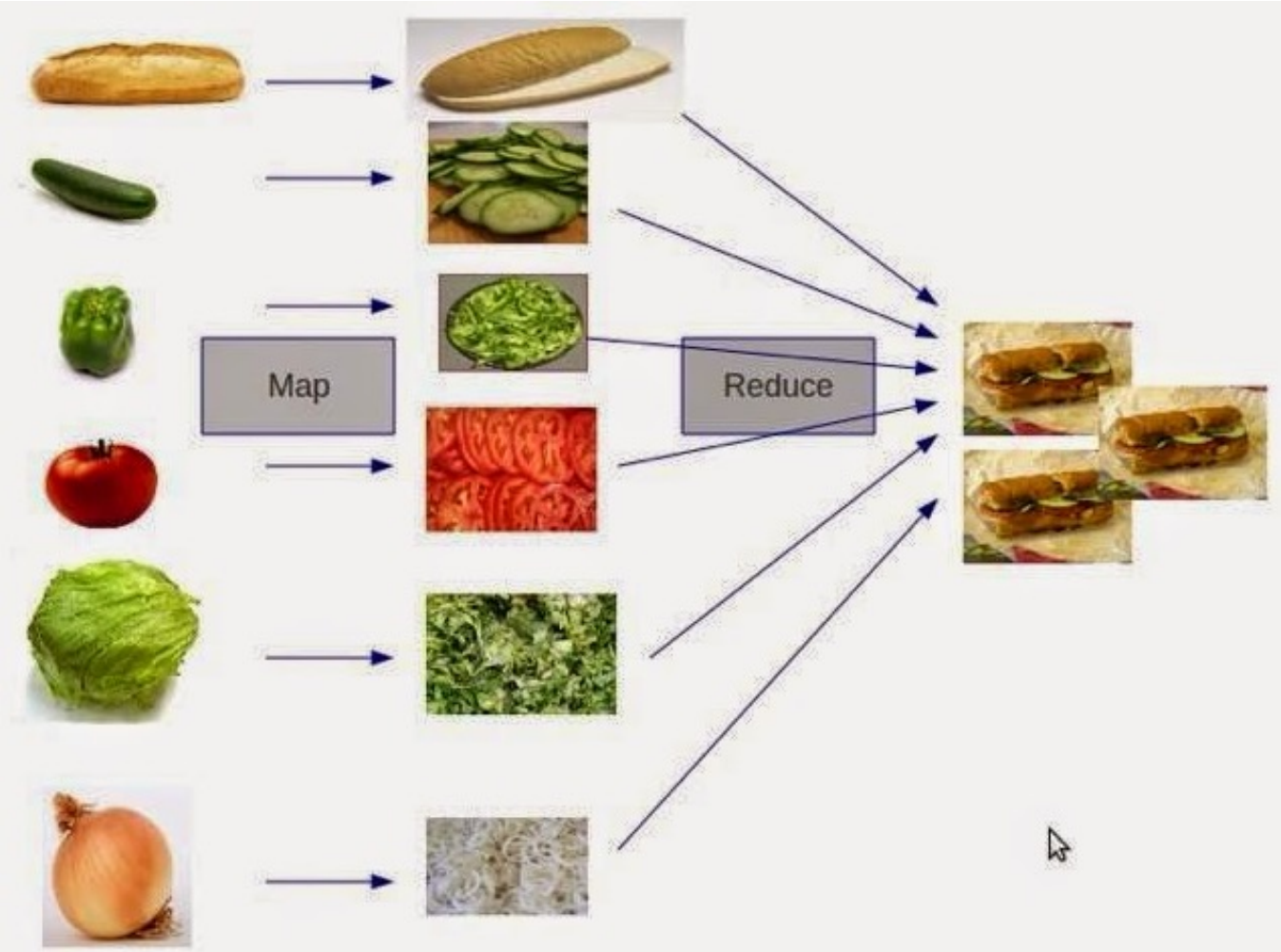


MapReduce - Introduction

universität koblenz
weiter:denken

Source: https://www.youtube.com/watch?v=lgWy7BwIKKQ

MapReduce



Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (January 2008), 107–113. https://doi.org/10.1145/1327452.1327492

MapReduce – What?

universität koblenz
weiter:denken

- MapReduce is a programming model for efficient distributed computing
- It works like a Unix pipeline
  - cat input | grep | sort | uniq -c | cat > output
  - **Input | Map |** Shuffle & Sort **| Reduce | Output**
- Efficiency from
  - Streaming through data, reducing seeks
  - Pipelining
- A good fit for a lot of applications
  - Log processing
  - Web index building

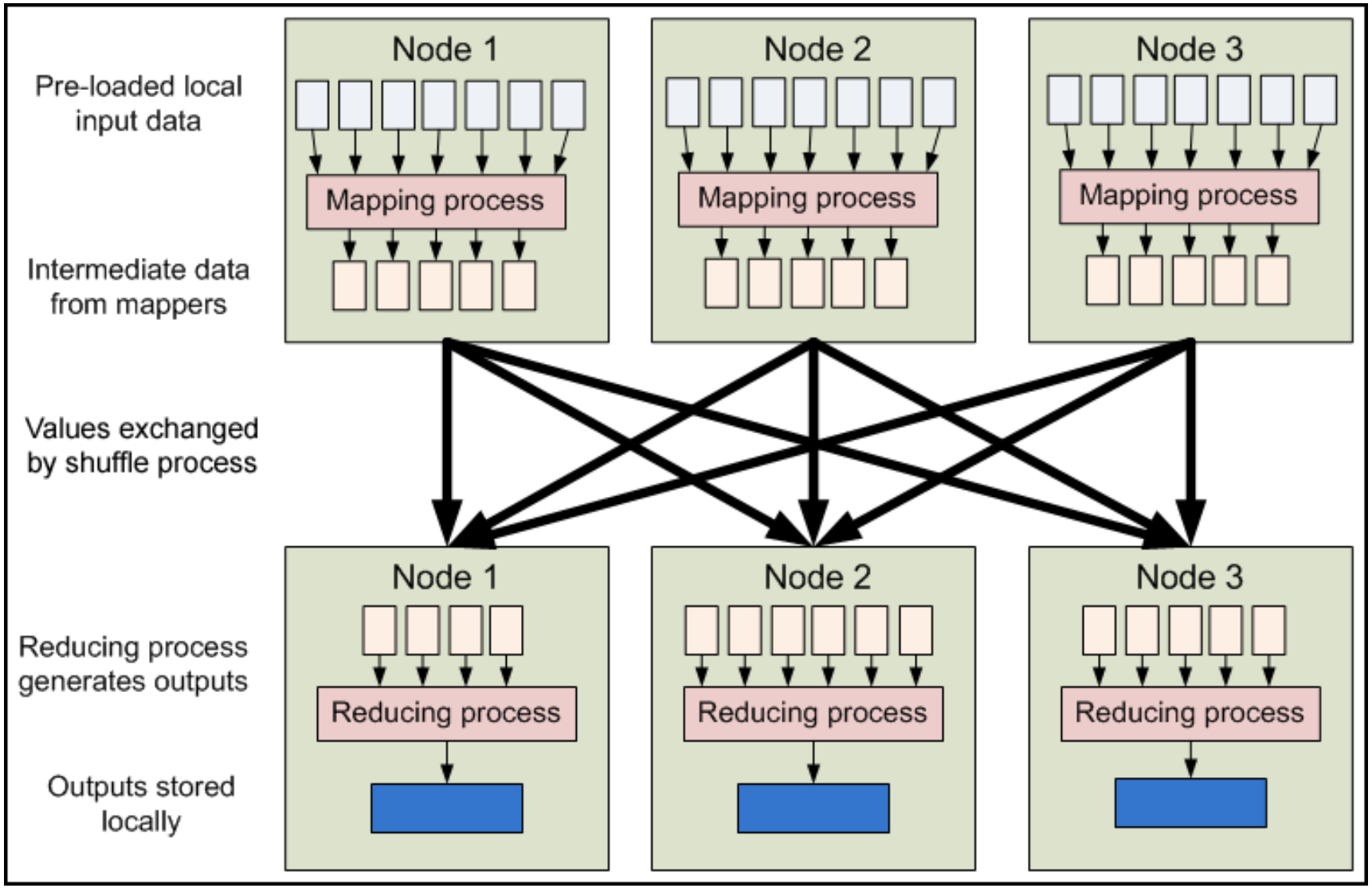
MapReduce

universität koblenz
weiter:denken

- Commodity hardware (files on HDFS)
- Fault-tolerant
- **Divide & conquer**: partition a large problem into smaller sub-problems
  - **Independent sub-problems** can be executed in parallel by workers
  - Intermediate results from each worker are **combined** to get the final result

MapReduce - Dataflow

universität koblenz
weiter:denken



MapReduce - Features

universität koblenz
weiter:denken

- Fine grained Map and Reduce tasks
  - Improved load balancing
  - Faster recovery from failed tasks
- Automatic re-execution on failure
  - In a large cluster, some nodes are always slow or flaky
  - Framework re-executes failed tasks
- Locality optimizations
  - With large data, bandwidth to data is a problem
  - Map-Reduce + HDFS is a very effective solution
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled close to the inputs when possible

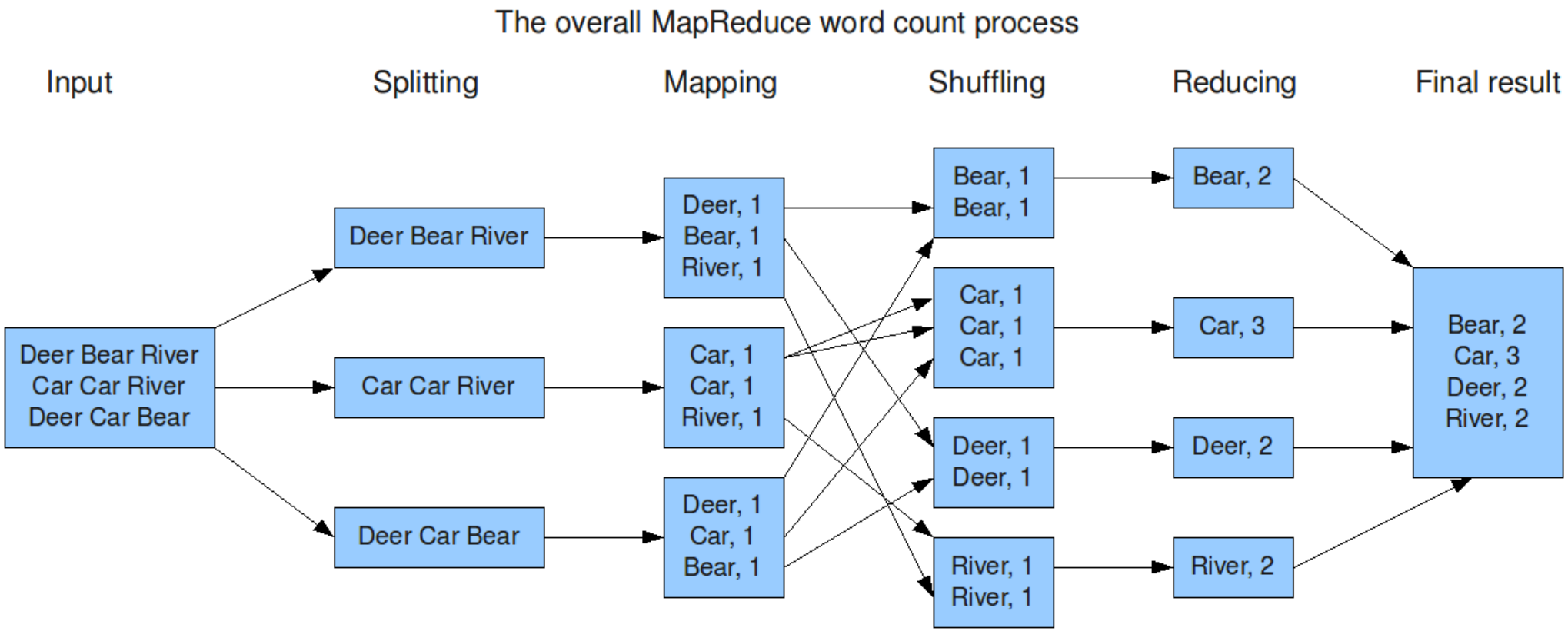
WordCount Example

universität koblenz
weiter:denken

- Mapper
  - Input: value: lines of text of input
  - Output: key: word, value: 1
- Reducer
  - Input: key: word, value: set of counts
  - Output: key: word, value: sum
- Launching program
  - Defines this job
  - Submits job to cluster

WordCount Dataflow

universität koblenz
weiter:denken



Example: MR word length count

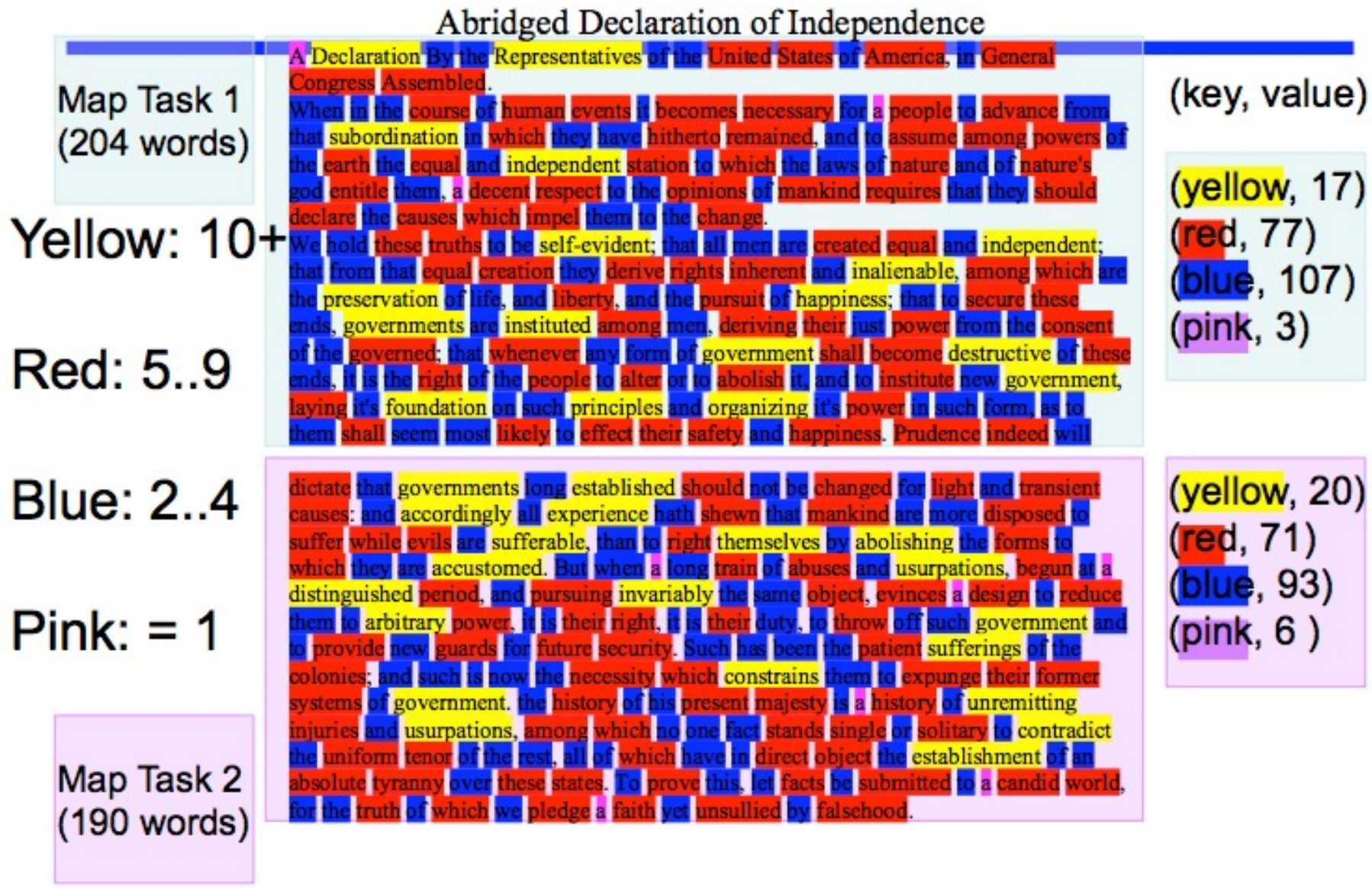
universität koblenz
weiter:denken

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.
When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

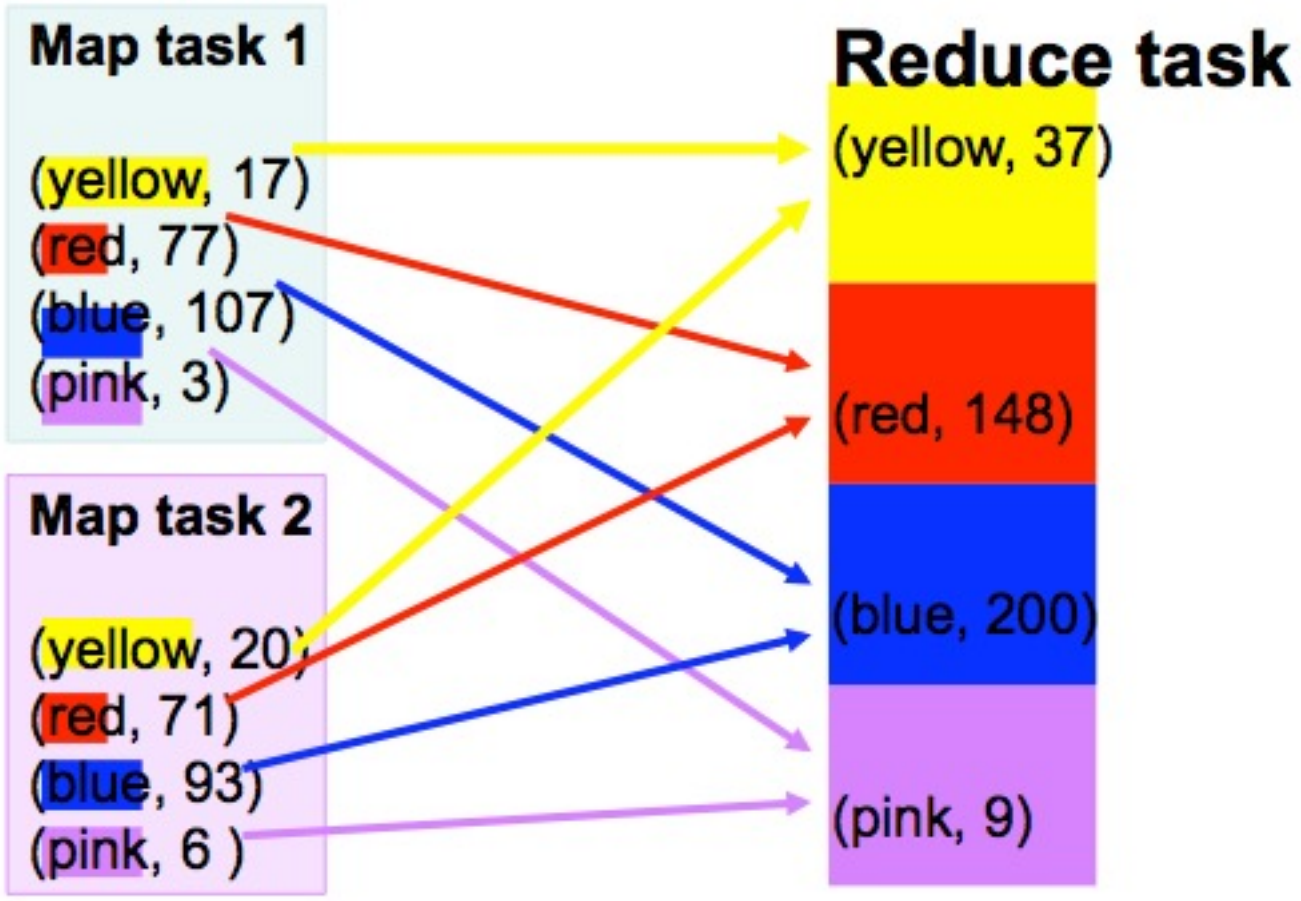
Example: MR word length count

universität koblenz
weiter:denken

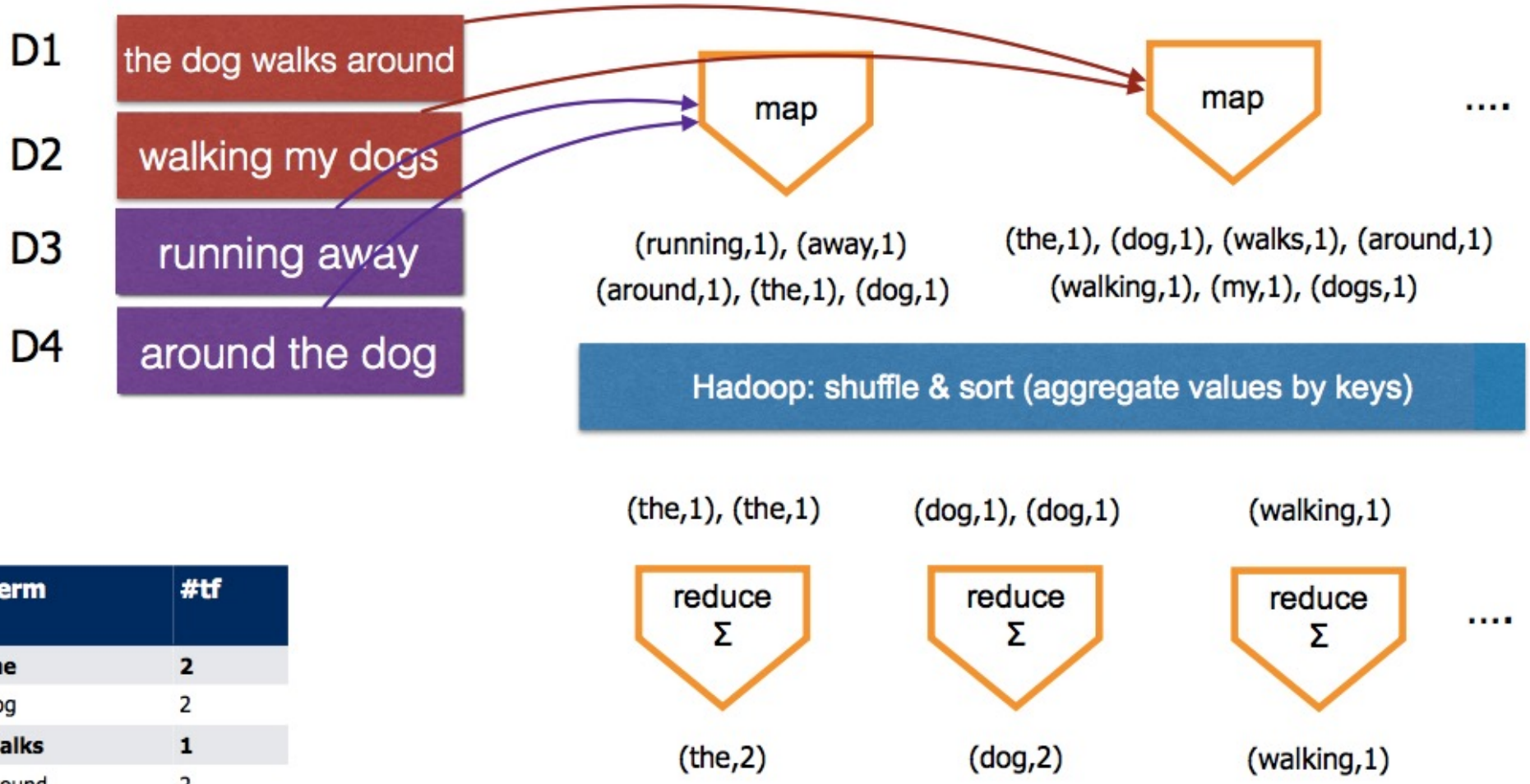


Example: MR word length count

universität koblenz
weiter:denken



MR: Word count example



Term	#tf
the	**2**
dog	2
walks	**1**
around	2
walking	**1**
my	1
....	...

Word Count Mapper

universität koblenz
weiter:denken

```
public static class Map extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable> {
   private static final IntWritable one = new IntWritable(1);
   private Text word  = new Text();

   public static void map(LongWritable key, Text value, OutputCollector<Text,IntWritable>
output, Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer = new StringTokenizer(line);
      while(tokenizer.hasNext()) {
         word.set(tokenizer.nextToken());
         output.collect(word,one);
         }
      }
 }
```

Word Count Reducer

universität koblenz
weiter:denken

```
public static class Reduce extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
public static void map(Text key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while(values.hasNext()) {
           sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
 }
```

Putting it all together

universität koblenz
weiter:denken

- Create a launching program for your application
- The launching program configures:
  - The *Mapper* and *Reducer* to use
  - The output key and value types (input types are inferred from the *InputFormat*)
  - The locations for your input and output
- The launching program then submits the job and typically waits for it to complete

Input and Output Formats

universität koblenz
weiter:denken

- A Map/Reduce may specify how its input is to be read by specifying an *InputFormat* to be used
- A Map/Reduce may specify how its output is to be written by specifying an *OutputFormat* to be used
- These default to *TextInputFormat* and *TextOutputFormat*, which process line-based text data
- Another common choice is *SequenceFileInputFormat* and *SequenceFileOutputFormat* for binary data
- These are file-based, but they are not required to be

Word Count Example

universität koblenz
weiter:denken

- Jobs are controlled by configuring *JobConfs*
- JobConfs are maps from attribute names to string values
- The framework defines attributes to control how the job is executed
  - conf.set(“mapred.job.name”, “MyApp”);
- Applications can add arbitrary values to the JobConf
  - conf.set(“my.string”, “foo”);
  - conf.set(“my.integer”, 12);
- JobConf is available to all tasks

Putting it all together

universität koblenz
weiter:denken

```
JobConf conf = new JobConf(WordCount.class);
conf.setJobName("wordcount");

conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);

conf.setMapperClass(Map.class);
conf.setCombinerClass(Reduce.class);
conf.setReducer(Reduce.class);

conf.setInputFormat(TextInputFormat.class);
Conf.setOutputFormat(TextOutputFormat.class);

FileInputFormat.setInputPaths(conf, new Path(args[0]));
FileOutputFormat.setOutputPath(conf, new Path(args[1]));

JobClient.runJob(conf);
```

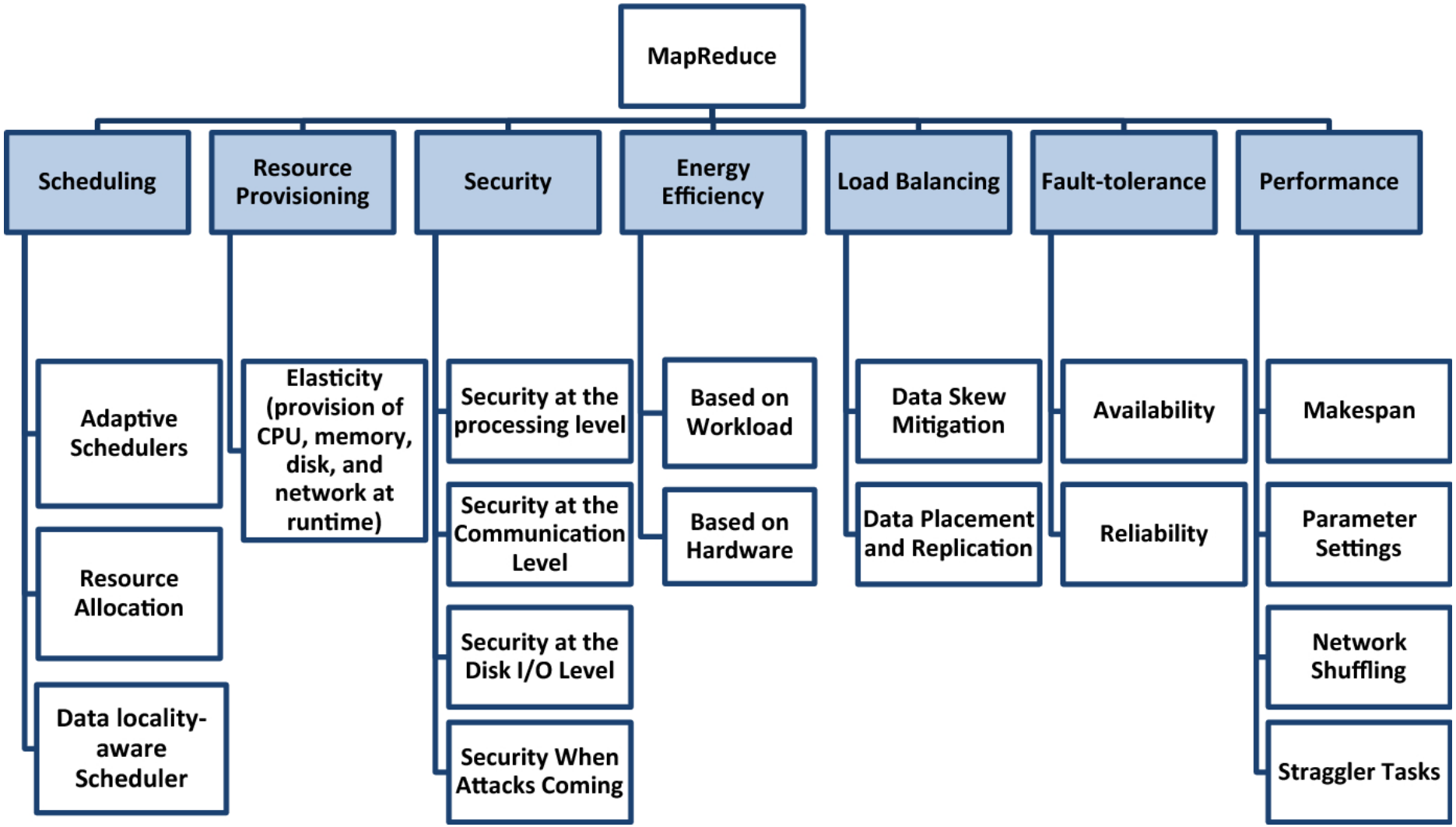
How many maps and reduces

universität koblenz
weiter:denken

- **Maps**
  - Usually as many as the number of HDFS blocks being processed, this is the default
  - Else the number of maps can be specified as a hint
  - The number of maps can also be controlled by specifying the *minimum split size*
  - The actual sizes of the map inputs are computed by:
    - *max(min(block_size,data/#maps), min_split_size*
- **Reduces**
  - Unless the amount of data being processed is small
    - *0.95*num_nodes*mapred.tasktracker.tasks.maximum*

Research involving the MapReduce framework

universität koblenz
weiter:denken



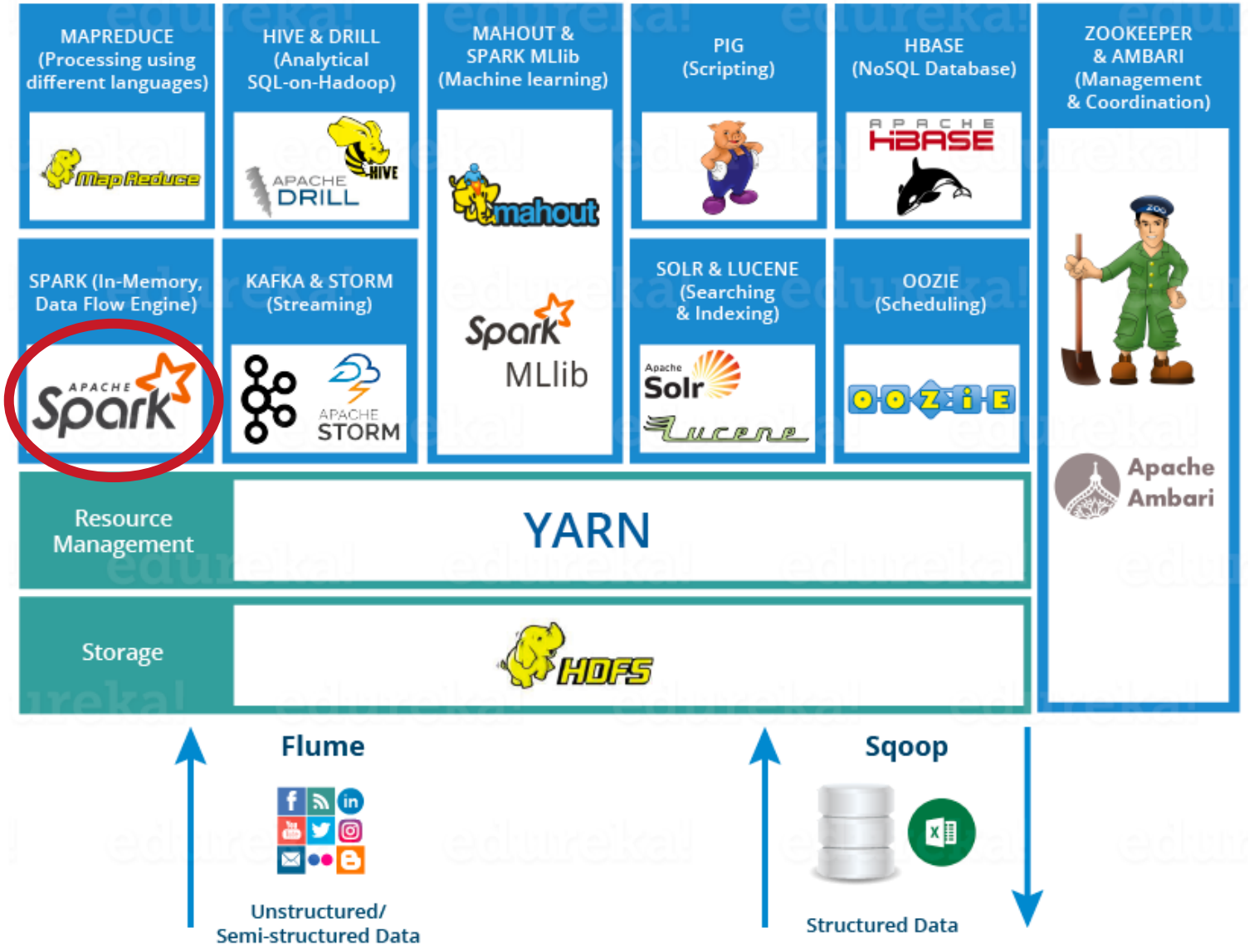
Maleki, N., Rahmani, A.M. & Conti, M. MapReduce: an infrastructure review and research insights. *J Supercomput* **75**, 6934–7002 (2019).

Outline

universität koblenz
weiter:denken

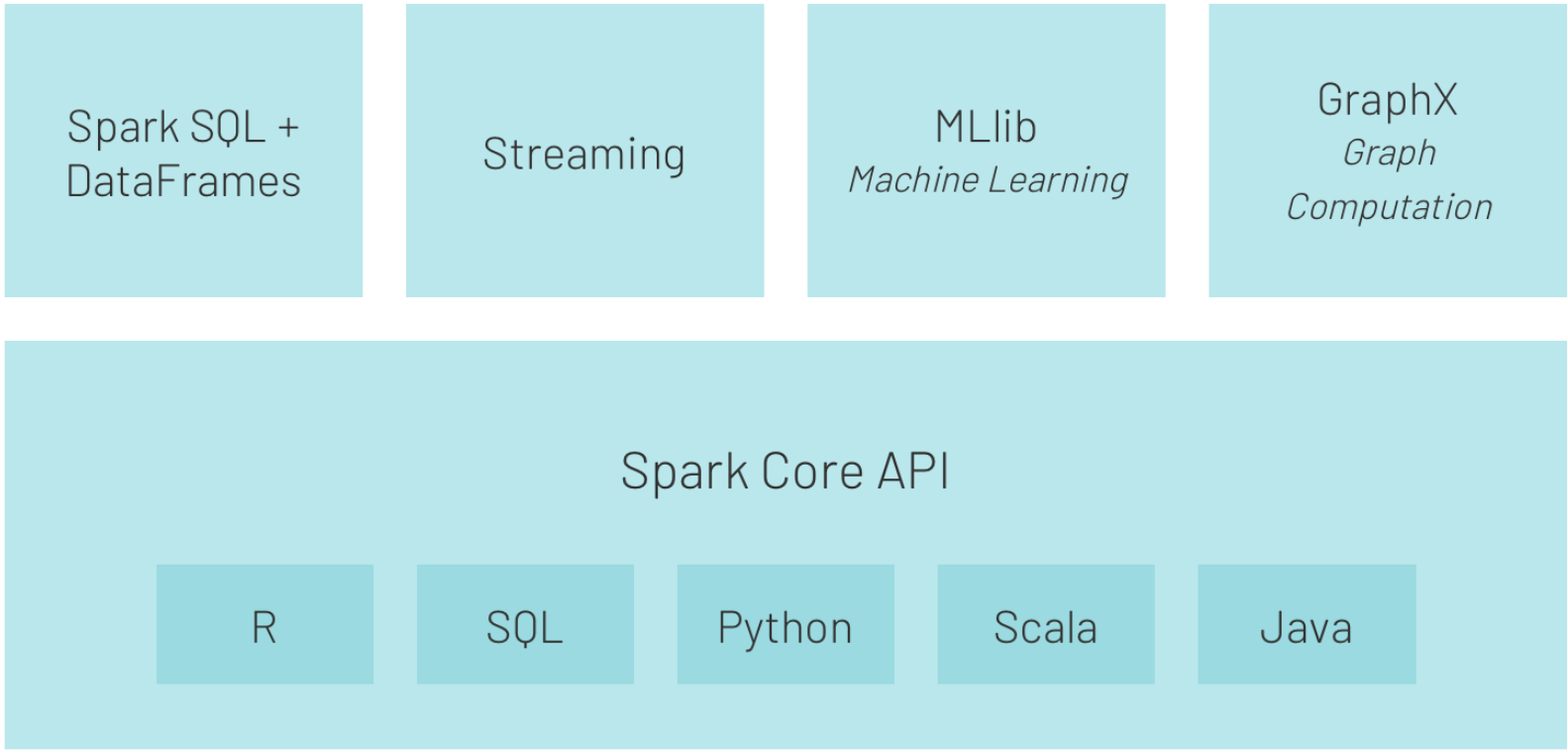
- Map/Reduce framework
- Querying
  - **Spark Core API**
  - Pig, Pig Latin
- Machine Learning at Scale
  - Spark MLLib
  - Mahout

Hadoop ecosystem



Spark Core APIs

universität koblenz
weiter:denken



SparkSQL

universität koblenz
weiter:denken

- Query structured data as a distributed dataset (RDD) in Spark
- Load and query data from a variety of sources
  - Hive table
  - JSON files
- Hive queries
- Spark SQL - DataFrames

SparkSQL – Example commands

universität koblenz
weiter:denken

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
val df = sqlContext.read.json("employee.json")

df.show()
df.select("name").show()
df.groupBy("age").count().show()
```

https://www.tutorialspoint.com/spark_sql/spark_sql_quick_guide.htm

SparkR

universität koblenz
weiter:denken

- Connect your R program to a Spark cluster from Rstudio
- SparkR package

```
library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib")))
sparkR.session(master = "local[*]", sparkConfig = list(spark.driver.memory = "2g"))
```

- convert a local R data frame into a SparkDataFrame (using “as.DataFrame()”)

https://spark.apache.org/docs/latest/sparkr.html

M/R in Python (PySpark package)

universität koblenz
weiter:denken

```
from pyspark import SparkConf, SparkContext
conf = (SparkConf()
         .setMaster("local")
         .setAppName("My app")
         .set("spark.executor.memory", "1g"))
sc = SparkContext(conf = conf)
```

https://spark.apache.org/docs/0.9.1/python-programming-guide.html

Word Count M/R in Python

universität koblenz
weiter:denken

```
36  lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
37  counts = lines.flatMap(lambda x: x.split(' ')) \
38                .map(lambda x: (x, 1)) \
39                .reduceByKey(add)
40  output = counts.collect()
```

Take first element of the input

Input file

Split lines using empty spaces to get a list of words

Count each word as appearing 1 time

Store results.
Because of lazy evaluation it only computes the data now

Add as reduce function: sum of all word values

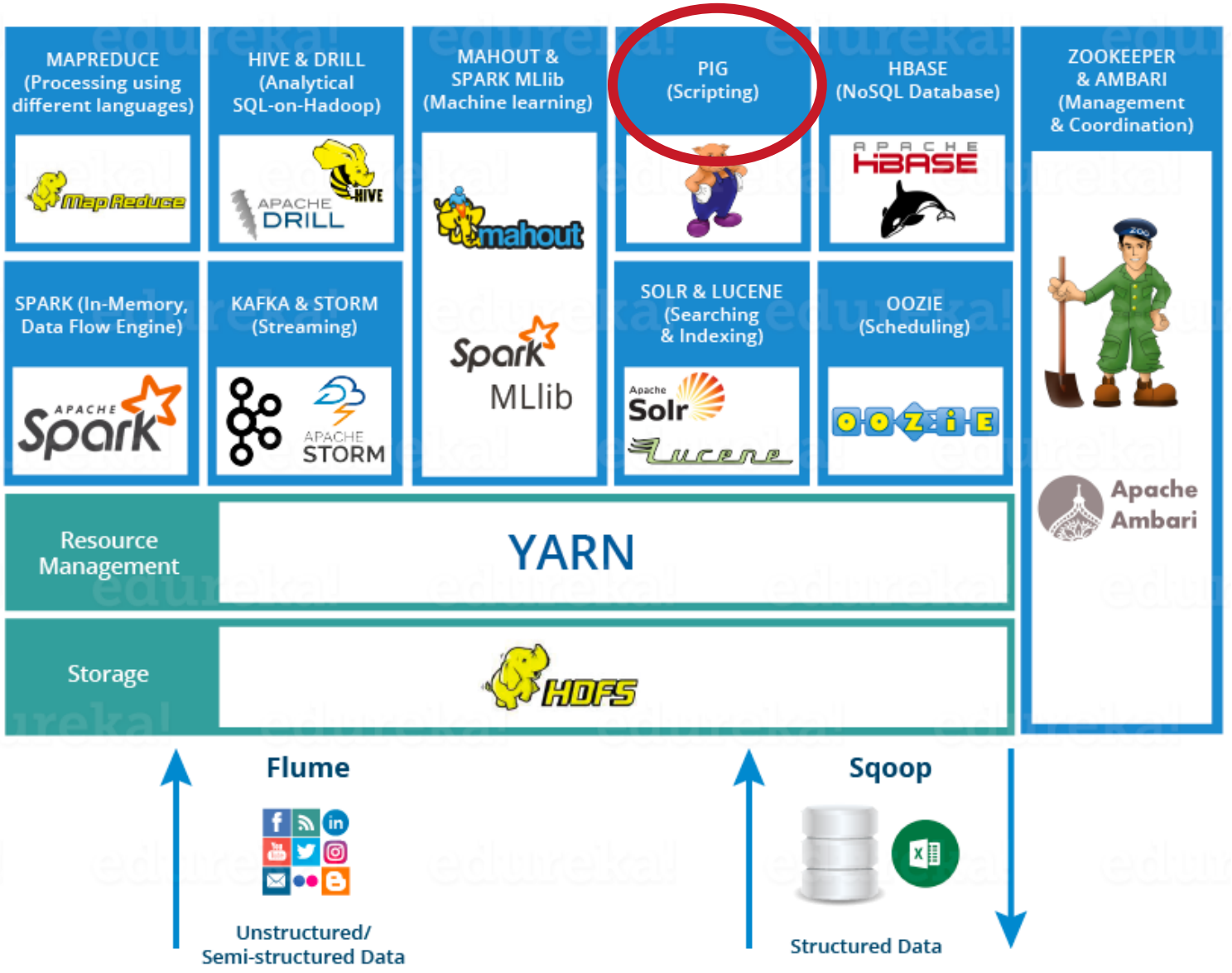
https://github.com/apache/spark/blob/master/examples/src/main/python/wordcount.py

Outline

universität koblenz
weiter:denken

- Map/Reduce framework
- Querying
  - Spark Core API
  - **Pig, Pig Latin**
- Machine Learning at Scale
  - Spark MLLib
  - Mahout

Hadoop ecosystem



Pig

universität koblenz
weiter:denken

- Started at Yahoo! Research
- Features
  - Expresses sequences of MapReduce jobs
  - Data model: nested “bags” of items
  - Provides relational (SQL) operators
  - (JOIN, GROUP BY, etc.)
  - Easy to plug in Java functions

Pig - Features

universität koblenz
weiter:denken

- Pig **handles erroneous**/corrupt **data** entries gracefully
  - Schema can be **inconsistent** or missing
  - (cleaning step can be skipped)
  - Exploratory analysis can be performed **quickly**

Pig - Features

universität koblenz
weiter:denken

- **Pigs eat anything**
  - Pig operates on any data (schema or not, files or not, nested or not)
- **Pigs live anywhere**
  - Parallel data processing language; implemented on Hadoop but not tied to it
- **Pigs are domestic animals**
  - Easily controlled and modified
- **Pigs fly**
  - Fast processing

Pig on Hadoop

universität koblenz
weiter:denken

- Makes use of **HDFS** and the **MapReduce core** of Hadoop
  - By default, reads input from & writes output to HDFS
- Pig Latin scripts are **compiled** into **one or more** Hadoop jobs which are executed in order
- Pig Latin users need **not** to be aware of the algorithmic details in the map/reduce phases

Pig Latin

universität koblenz
weiter:denken

- A parallel **dataflow language**: users describe **how** data is read, processed and stored
- Dataflows can be simple (e.g. “counting words”) or complex (multiple inputs are joined, data is split up into streams and processed separately)

Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. 2008. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 1099–1110. https://doi.org/10.1145/1376616.1376726

Pig vs. Pig Latin

universität koblenz
weiter:denken

- Pig: an **engine** for executing **data flows** in parallel on Hadoop
- **Pig Latin**: the high-level (SQL-like) language for expressing data flows
- Pig Latin contains common data processing operators (**join**, **sort**, **filter**, …)
- **User defined functions** (UDFs): developers can write their own functions to read/process/store the data

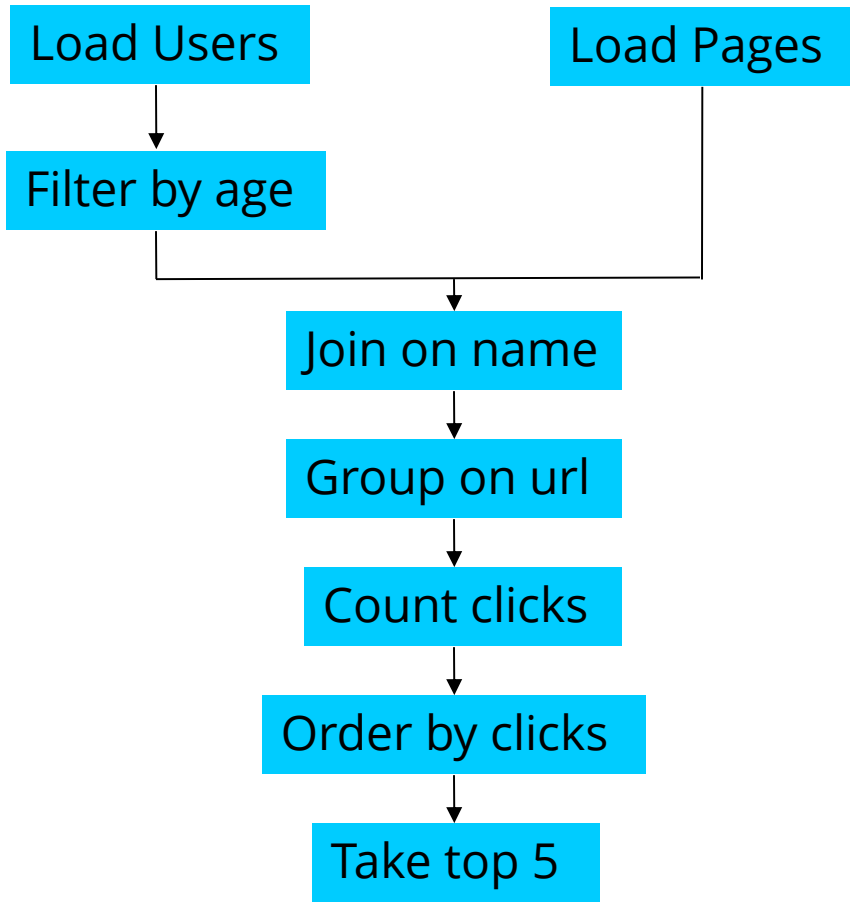
Pig vs. SQL

universität koblenz
weiter:denken

Pig	SQL
Procedural: script describes **how** to process the data	Descriptive: query describes **what** the output should be
Workflows can contain many data processing operations	One query answers one question (*subqueries)
Schemas may be unknown or inconsistent	RDBMSs have defined schemas
Reads files from HDFS (and other sources)	Data is read from database tables

An example problem

Suppose you have user data in a file, website data in another, and you need to find the top 5 most visited pages by users aged 18-25



In Java

universität koblenz
weiter:denken

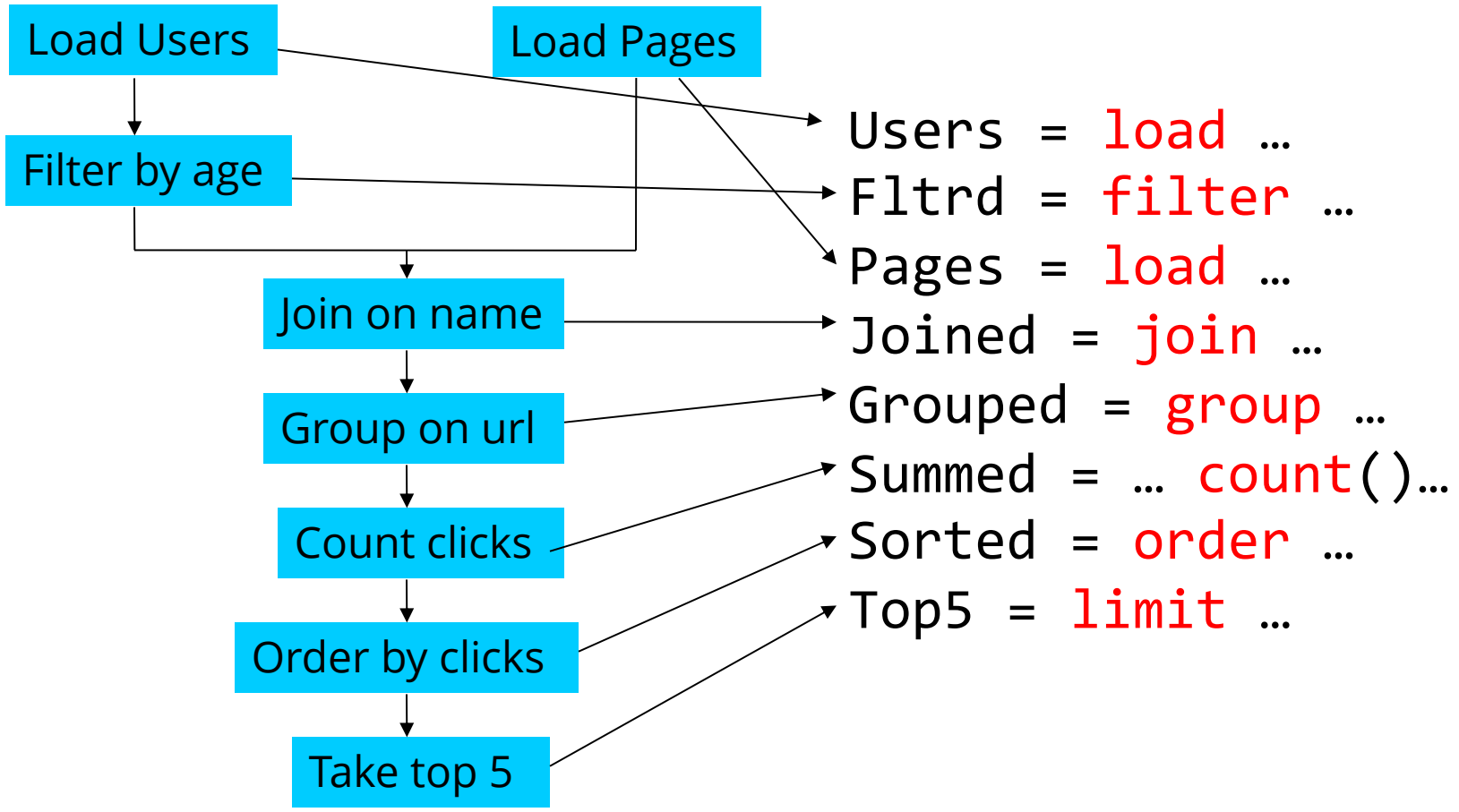
```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobControl;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring(firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.toString();
                if (value.charAt(0) == '1')
first.add(value.substring(1));
                else second.add(value.substring(1));

                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don't need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
                new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.addInputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
                new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf(MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMapper.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRExample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFileOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputFormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

In Pig Latin

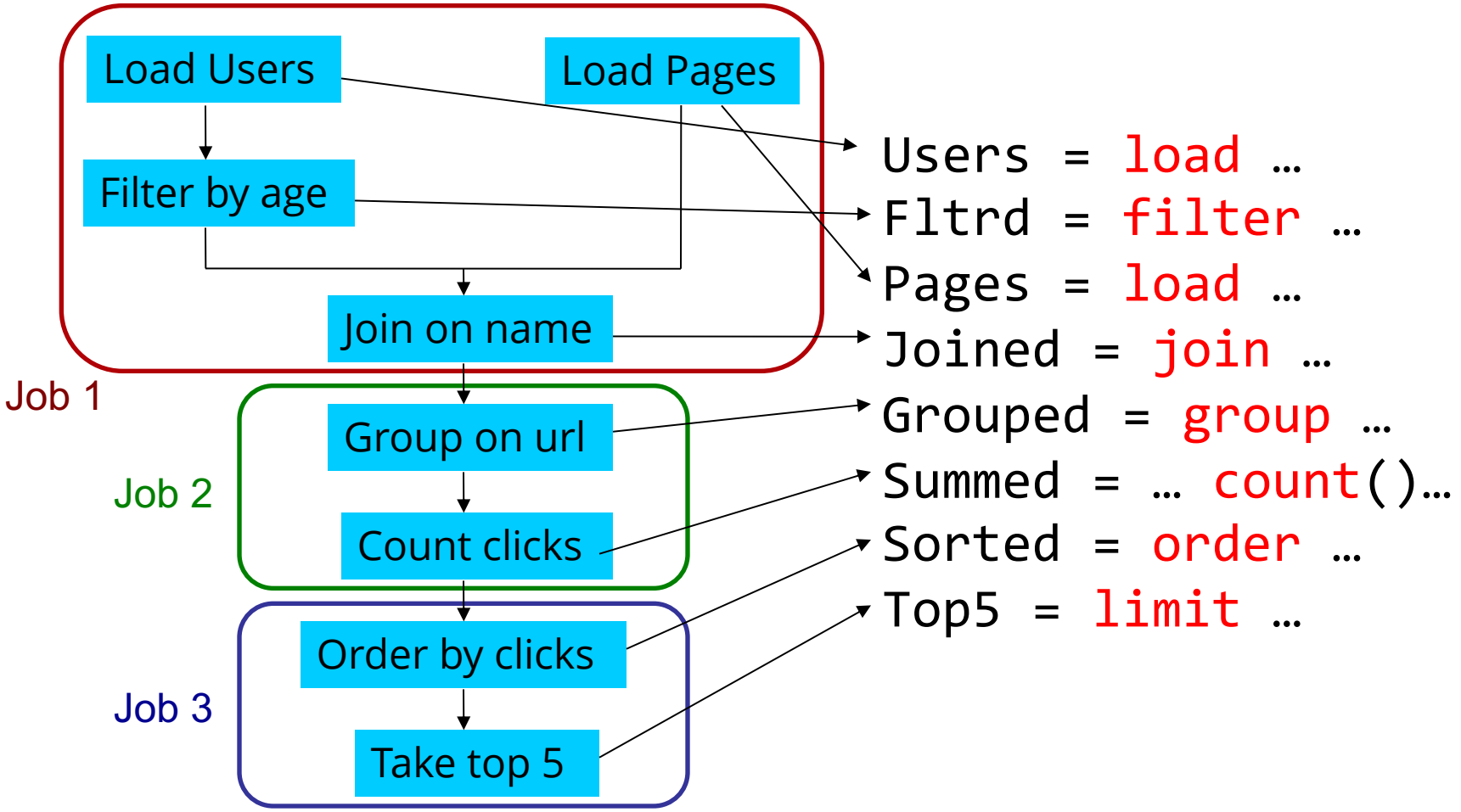
universität koblenz
weiter:denken

```
Users = load ‘users’ as (name, age);
Filtered = filter Users by age >= 18 and age <=
25;
Pages = load ‘pages’ as (user, url);
Joined = join Filtered by name, Pages by user;
Grouped = group Joined by url;
Summed = foreach Grouped generate group,
                count(Joined) as clicks;
Sorted = order Summed by clicks desc;
Top5 = limit Sorted 5;
store Top5 into ‘top5sites’;
```

Ease of Translation



Ease of Translation



Pig – Word Count Example

universität koblenz
weiter:denken

```
-- read the file pg46.txt line by line, call each record line
cur = load 'pg46.txt' as (line);

-- tokenize each line, each term is now a record called word
words = foreach cur generate flatten(TOKENIZE(line)) as word;

-- group all words together by word
grpd  = group words by word;

-- count the words
cntd  = foreach grpd generate group, COUNT(words);

/*
 * start the Hadoop job and print results
 */
dump cntd;
```

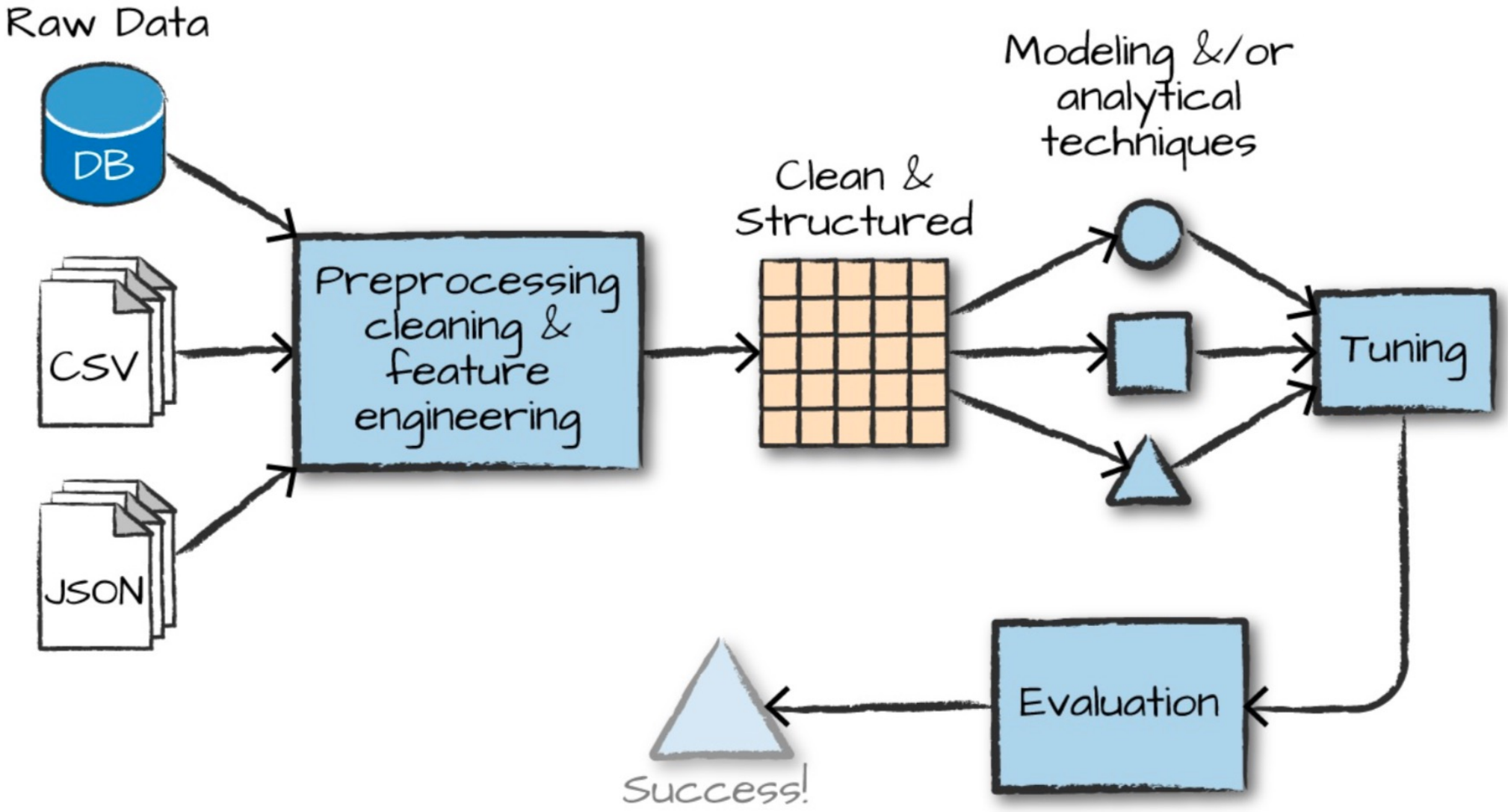
Outline

universität koblenz
weiter:denken

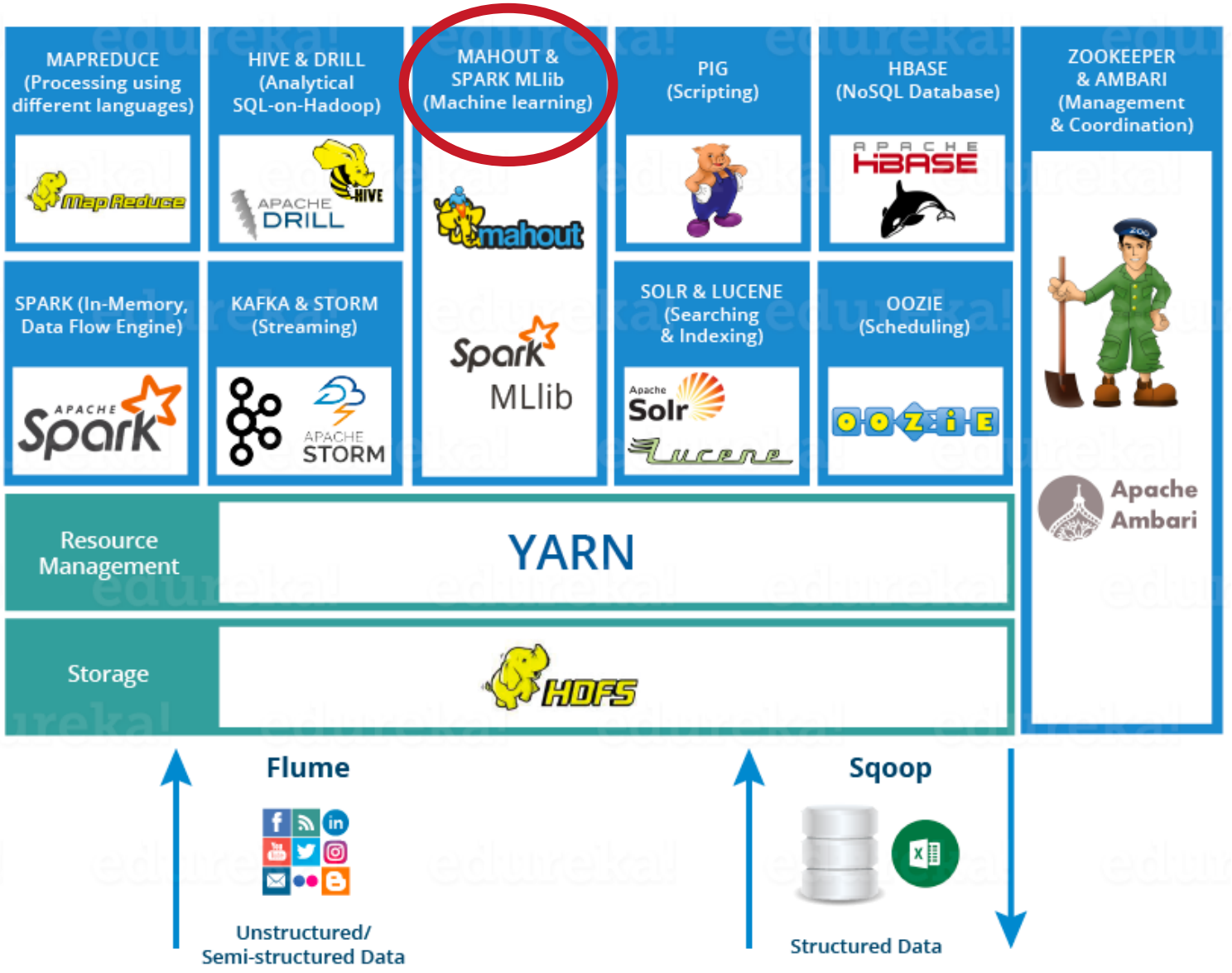
- Map/Reduce framework
- Querying
  - Spark Core API
  - Pig, Pig Latin
- **Machine Learning at Scale**
  - Spark MLLib
  - Mahout

Going beyond M/R

universität koblenz
weiter:denken



Hadoop ecosystem



Spark's Advanced Analytics Tools

universität koblenz
weiter:denken

- Spark includes several core packages and many external packages for performing advanced analytics

- **MLlib:** Primary package
  - Provides an interface for building ML pipelines:
    - Gathering and cleaning data
    - Feature engineering and selection
    - Training and tuning large-scale supervised and unsupervised ML models
    - Using models in production

MLLib vs. other ML packages

universität koblenz
weiter:denken

- ML packages like scikit-learn, TensorFlow:
- Designed to perform ML on a single machine
- Limited either in terms of the size of data you can train on or the processing time
- Are *complementary* to MLlib
- When you hit scalability issues, take advantage of Spark's abilities!

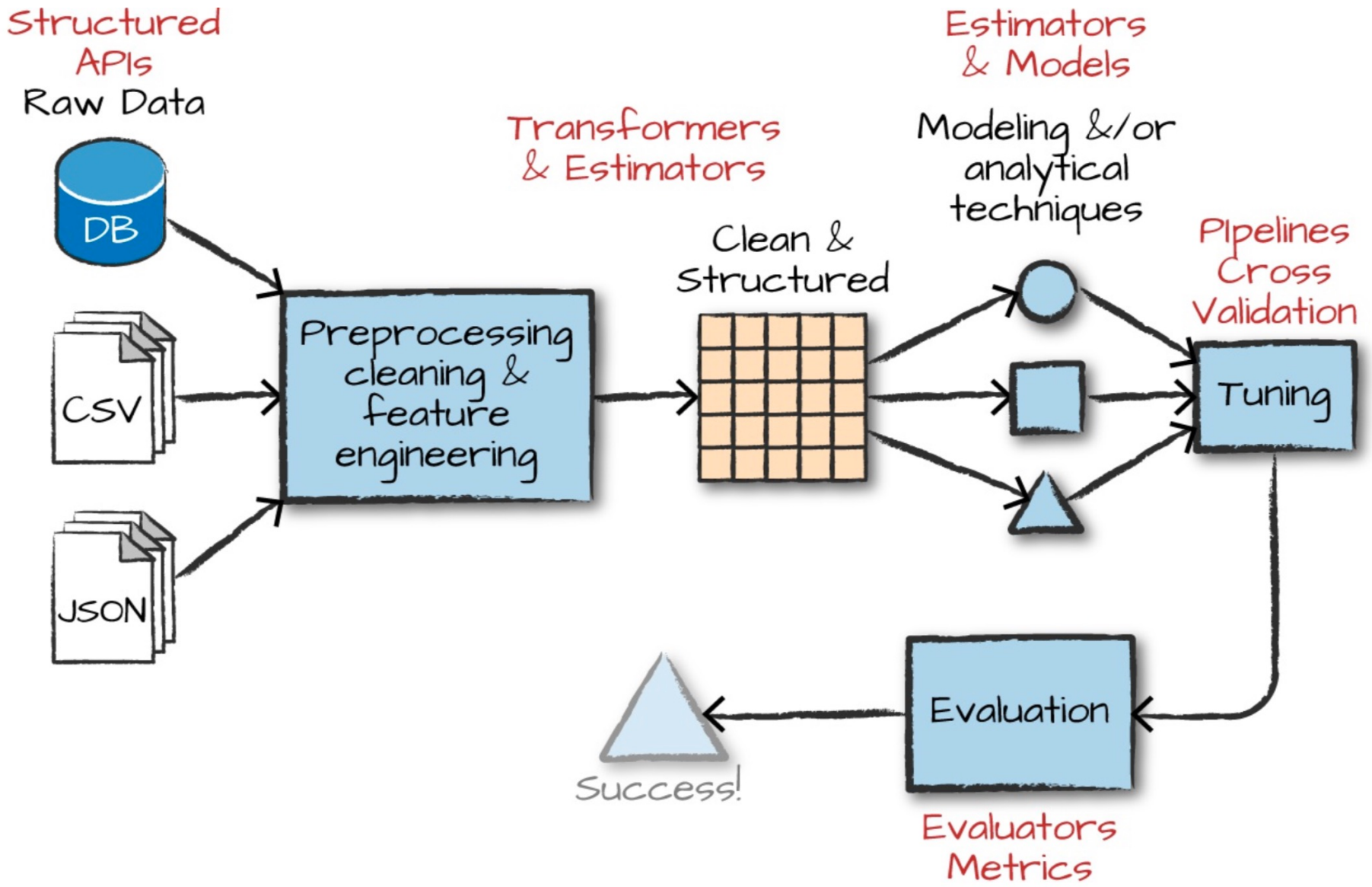
Scalable ML using Spark MLLib

universität koblenz
weiter:denken

- **Preprocessing and feature generation:**
  - To reduce the time to produce training and test data from a large amount of data (use Spark's feature preprocessing and generation)
- **Input data or model size cannot be handled by one machine:**
  - Use Spark's ability for distributed ML

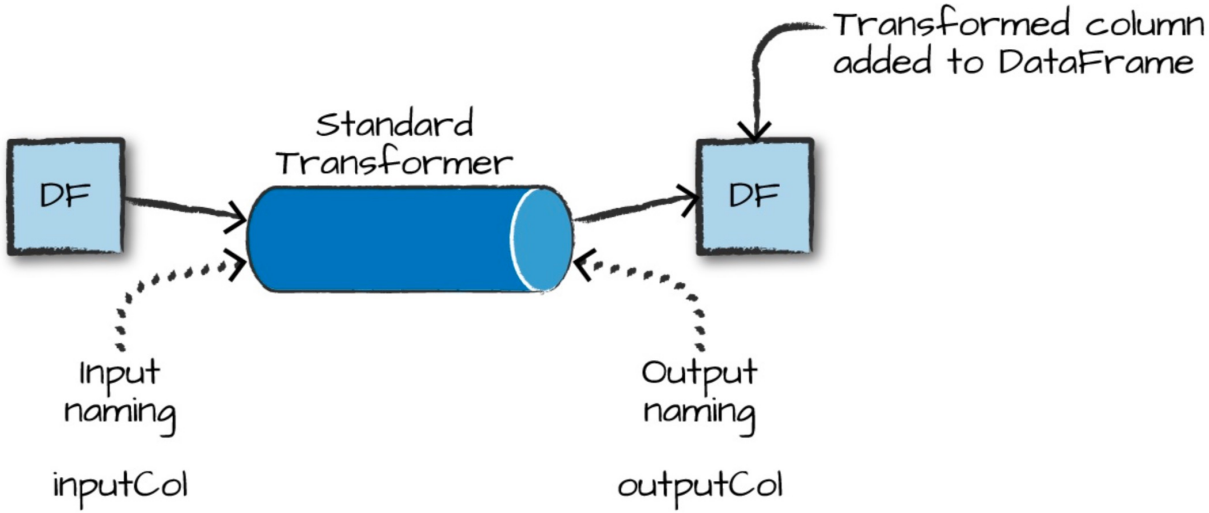
M. Assefi, E. Behravesh, G. Liu and A. P. Tafti, "Big data machine learning using apache spark MLlib," *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 2017, pp. 3492-3498.

High-level MLLib concepts



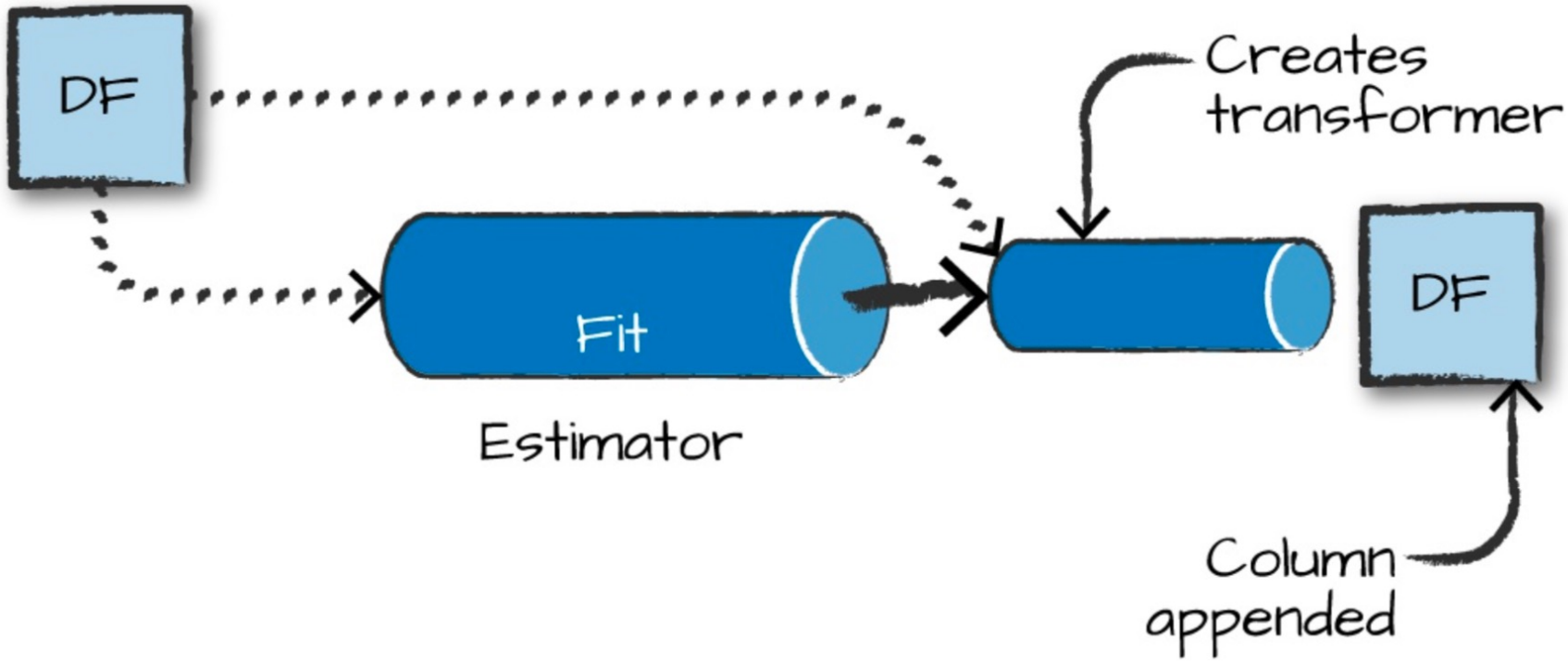
Transformers - Functions that convert raw data in some way

- **Examples**:
  - Create new interaction variable (from two other variables)
  - Change an Integer into a Double type
  - Convert string categorical variables into numerical values



Estimators

- Transformer that is initialized with data
  - e.g., StandardScaler scales an input column to have a zero mean and a variance of 1 in each dimension



Access via Spark Core APIs (e.g., using SparkR)

universität koblenz
weiter:denken

SparkR supports the following machine learning algorithms currently:

- spark.glm or glm: Generalized Linear Model
- spark.survreg: Accelerated Failure Time (AFT) Survival Regression Model
- spark.naiveBayes: Naive Bayes Model
- spark.kmeans: K-Means Model
- spark.logit: Logistic Regression Model
- spark.isoreg: Isotonic Regression Model
- spark.gaussianMixture: Gaussian Mixture Model
- spark.lda: Latent Dirichlet Allocation (LDA) Model
- spark.mlp: Multilayer Perceptron Classification Model
- spark.gbt: Gradient Boosted Tree Model for Regression and Classification
- spark.randomForest: Random Forest Model for Regression and Classification
- spark.als: Alternating Least Squares (ALS) matrix factorization Model
- spark.kstest: Kolmogorov-Smirnov Test

https://spark.apache.org/docs/latest/sparkr.html

Apache Mahout

universität koblenz
weiter:denken

- Open source implementation of distributed machine learning algorithms with focus on linear algebra
- Initially implemented on top of Hadoop M/R framework, now increasingly focused on Spark

Robin Anil, Gokhan Capan, Isabel Drost-Fromm, Ted Dunning, Ellen Friedman, Trevor Grant, Shannon Quinn, Paritosh Ranjan, Sebastian Schelter, and Özgür Yilmazel. 2020. Apache Mahout: machine learning on distributed dataflow systems. *J. Mach. Learn. Res.* 21, 1, Article 127, 2020.

https://mahout.apache.org/

MAHOUT

Algorithms implemented

universität koblenz
weiter:denken

- Distributed linear algebra
  - Distributed QR Decomposition
  - Distributed Stochastic Principal Component Analysis
  - Distributed Stochastic Singular Value Decomposition
- Regression
  - Ordinary Least Square
  - Ridge Regression
  - Cochrane Orcutt Procedure
  - Durbin Watson Test
- Clustering
  - Canopy Clustering
  - Distance metrics

Summary

universität koblenz
weiter:denken

- Map/Reduce framework
- Querying
  - Spark Core API
  - Pig, Pig Latin
- Machine Learning at Scale
  - Spark MLLib
  - Mahout

What's next – Data Streams

universität koblenz
weiter:denken

- Record-at-a-time streaming
- Declarative, functional streaming
- Declarative, relational streaming