DATA ANALYTICS WITH APACHE SPARK

A short introduction

René Jäkel

www.scads.de

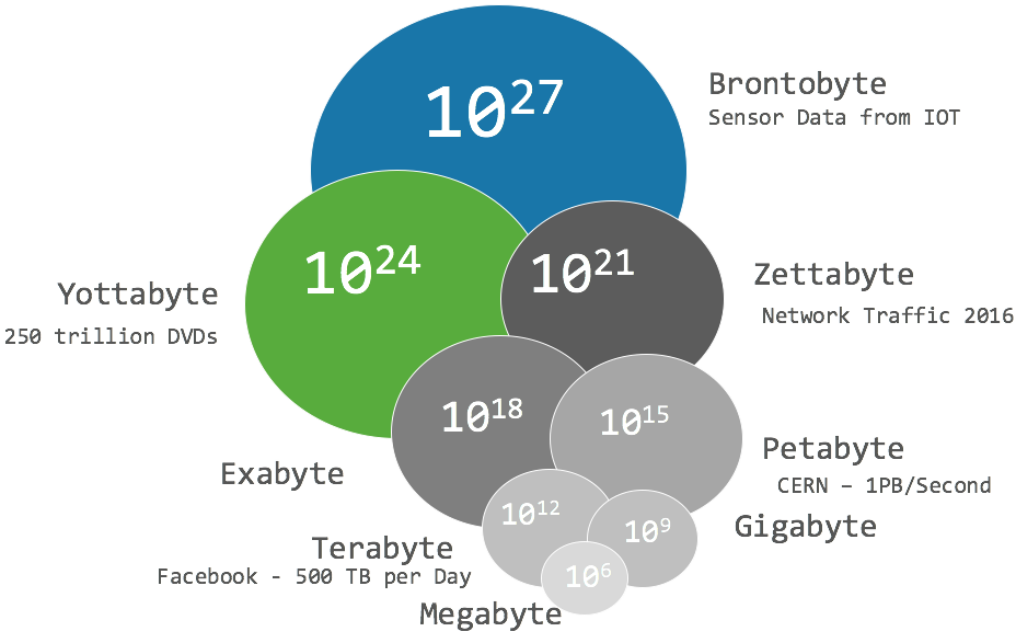
ScaDS DRESDEN LEIPZIG

AGENDA

Time	Topic
11:00-13:00	Brief introduction in Big Data Context and historical remarks Hands-On Part 1 (preparation and basics)
13:00-14:00	Lunch Break
14:00-16:00	Hands-On Part 2 (data manipulation) Further analytics concepts Big Data trends

BIG DATA - 4VS

Big Data „too big for traditional methods“



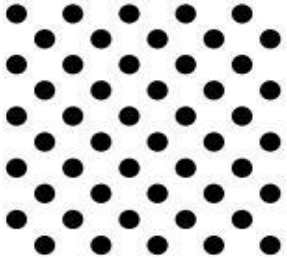
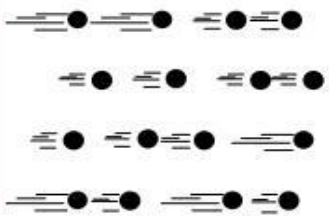
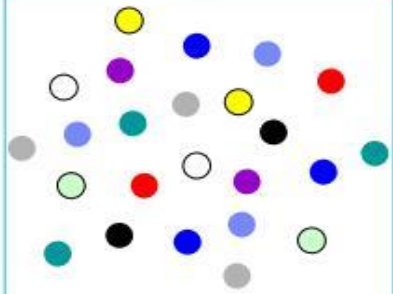
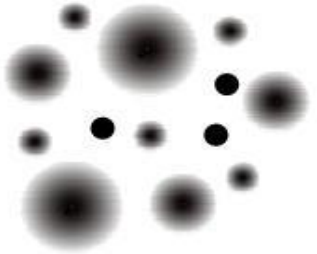
Volume
Machinegenerated + Humangenerated + Businessdata

Variety
Text,
Audio,
Images,
Video,
Clickstreams,
Logfiles

Velocity
Realtime
Seconds
Minutes
Hours
Days

Veracity
Weatherdata, Sensordata, Tradedata, ...

BIG DATA? – SOME V'S

Volume	Velocity	Variety	Veracity*
			
Data at Rest	**Data in Motion**	**Data in Many Forms**	**Data in Doubt**
Terabytes to exabytes of existing data to process	Streaming data, milliseconds to seconds to respond	Structured, unstructured, text, multimedia	Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

more important:

extract new content from database

ScaDS DRESDEN LEIPZIG

EVEN MORE ASPECTS ?!

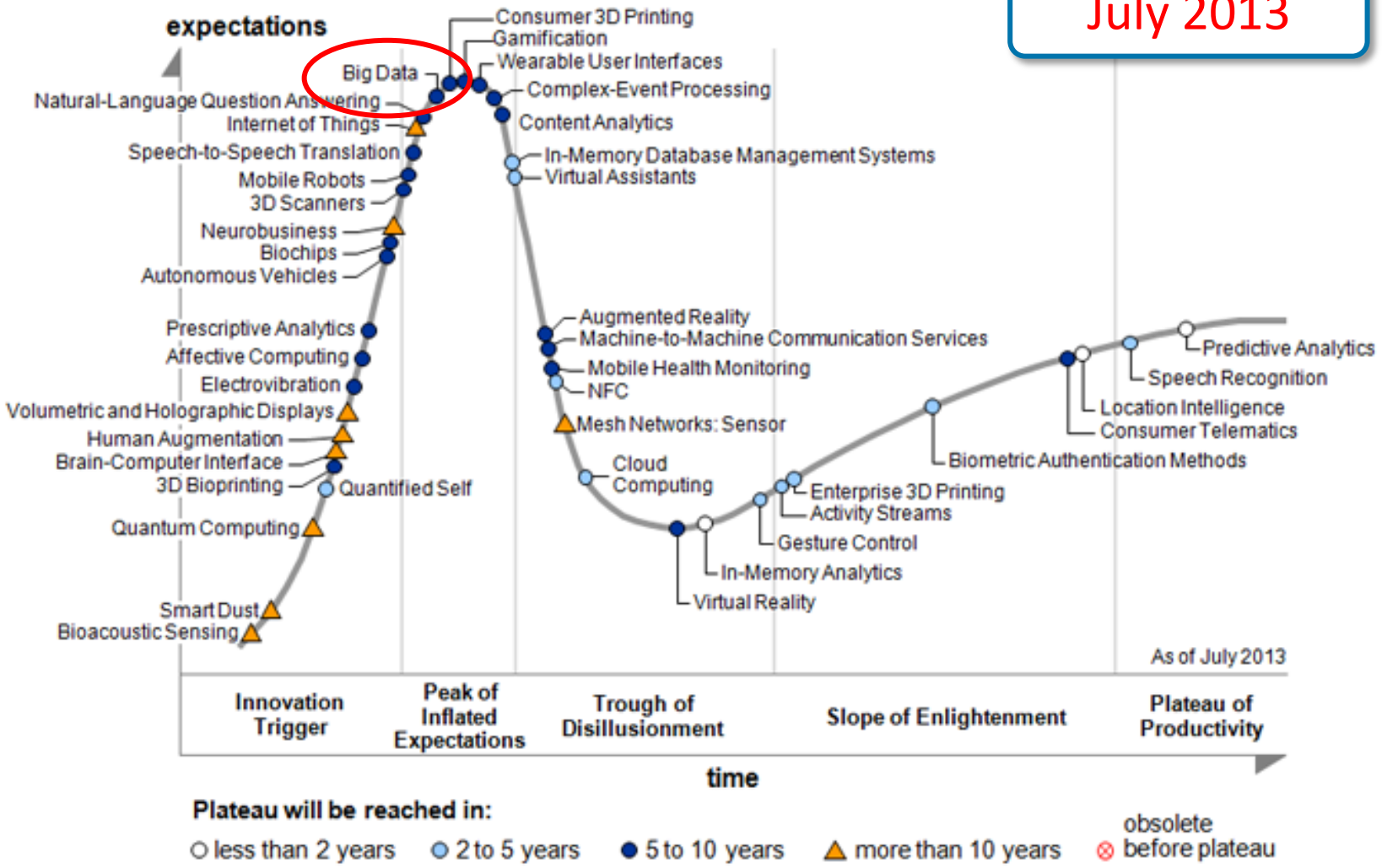
- **Venue**: distributed, heterogeneous data from multiple platforms, from different owners' systems, with different access and formatting requirements, private vs. public cloud.
- **Vocabulary**: schema, data models, semantics, ontologies, taxonomies, and other content- and context-based metadata that describe the data's structure, syntax, content, and provenance.
- **Vagueness**: "confusion" over the meaning of big data (Is it Hadoop? Is it something that we've always had? What's new about it? What are the tools? Which tools should I use? etc.)

Source: MapR blog by Kirk Borne:
"Top 10 Big Data Challenges – A Serious Look at 10 Big Data V's" [1]

ScaDS DRESDEN LEIPZIG

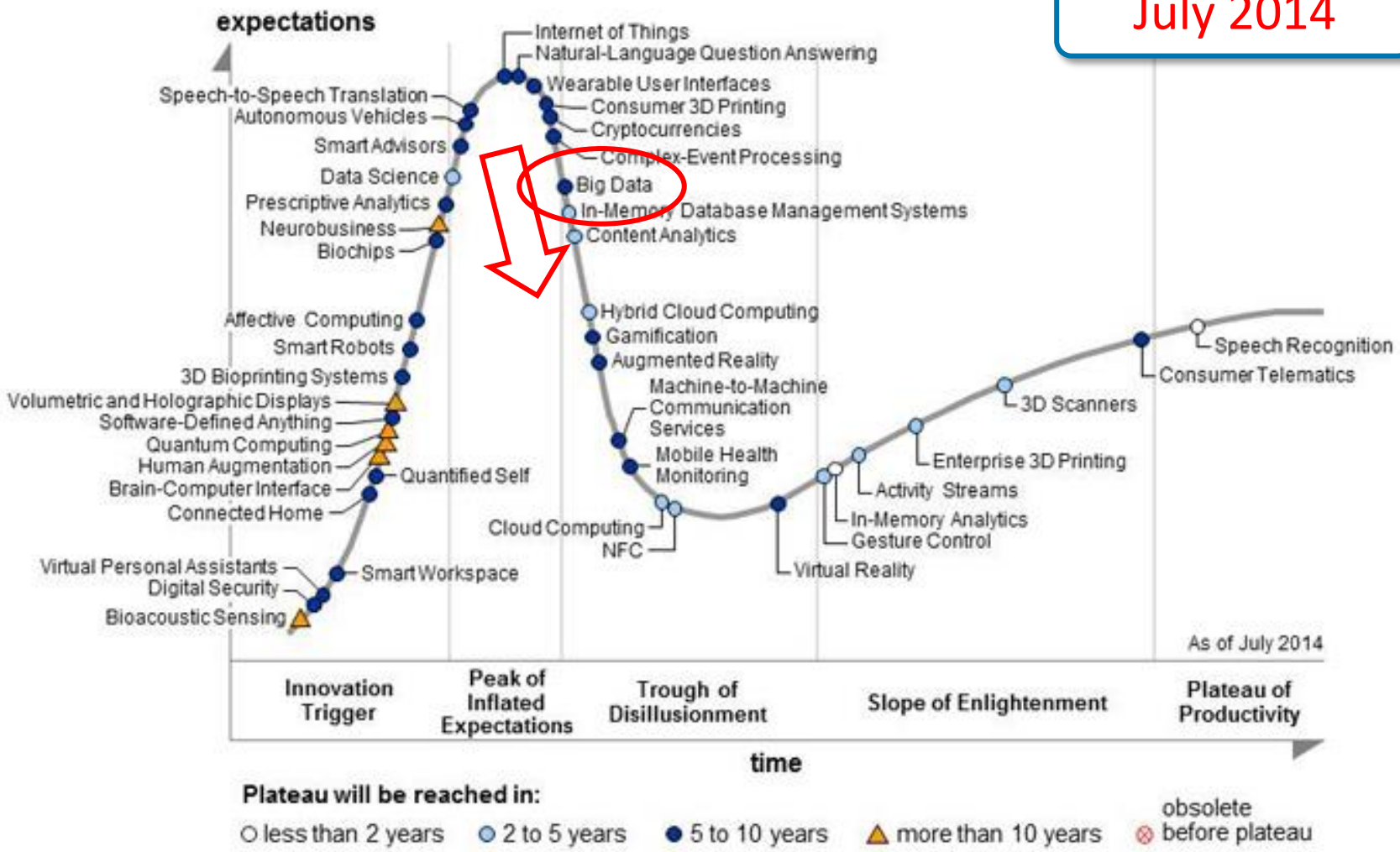
HYPE VS. REALITY?

July 2013



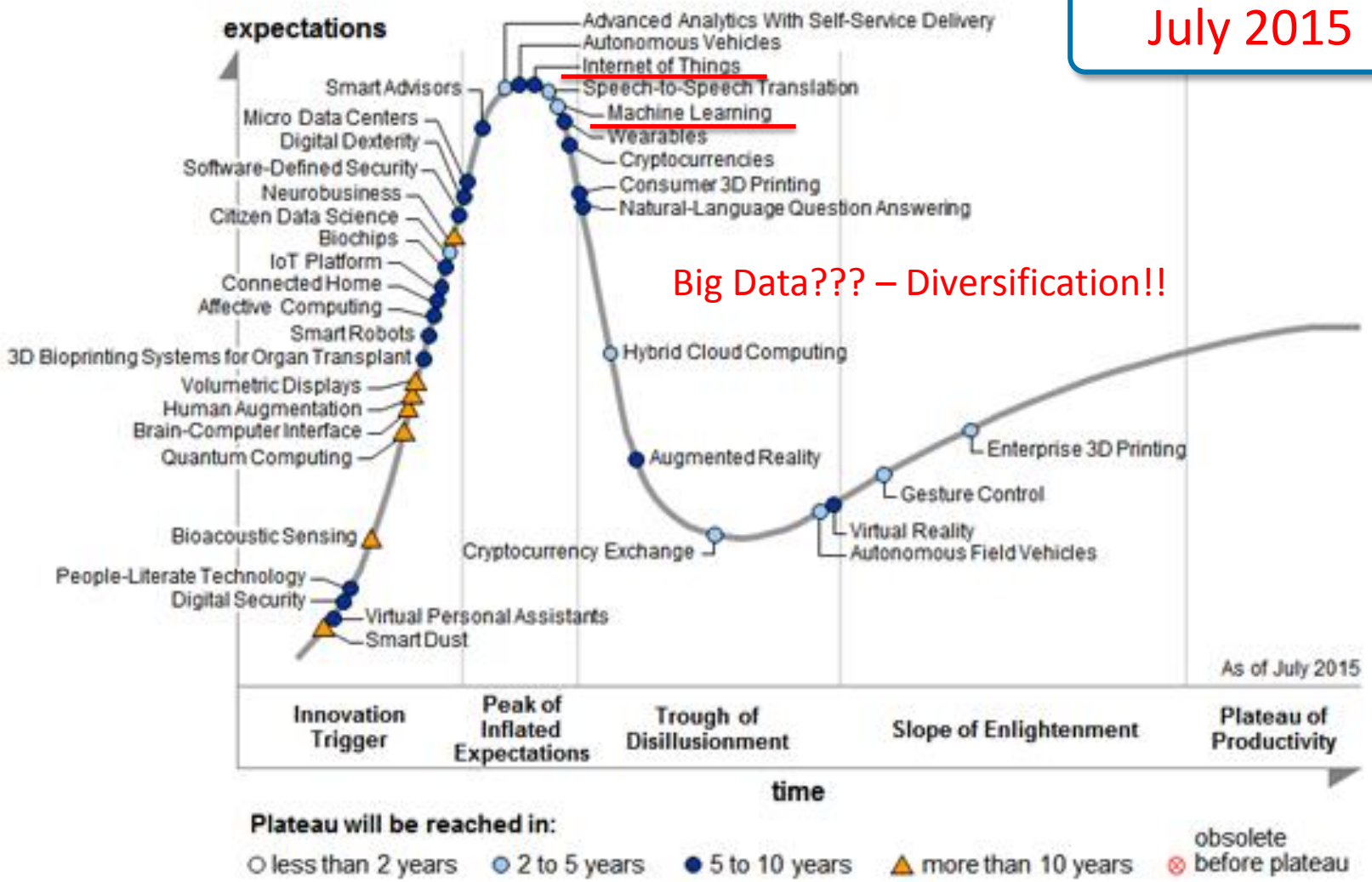
HYPE VS. REALITY?

July 2014



HYPE VS. REALITY?

July 2015

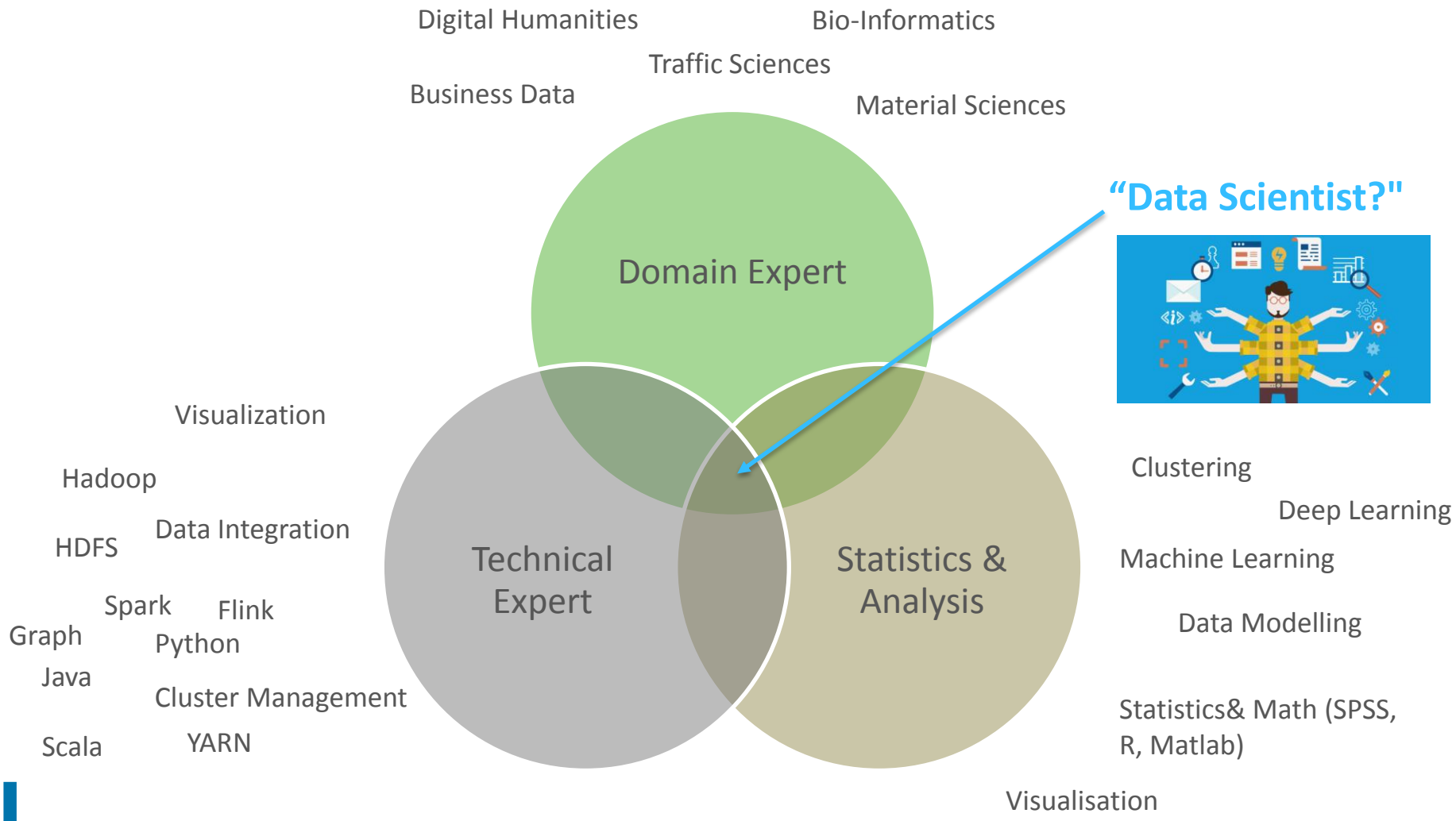


ScaDS
DRESDEN LEIPZIG

SHORT SCADS INTRODUCTION

ScaDS DRESDEN LEIPZIG

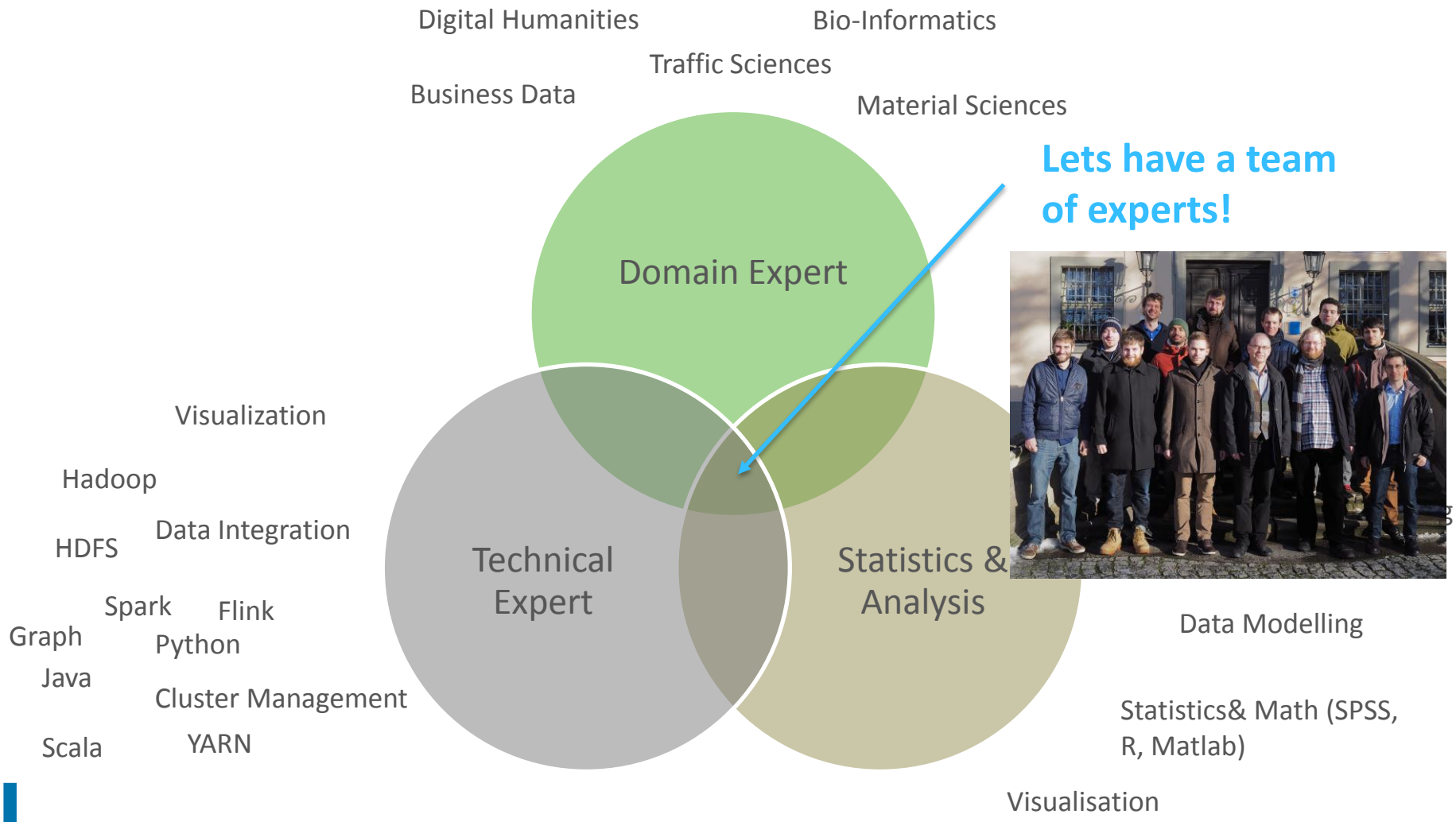
IS THERE A JACK OF ALL TRADES?



https://www.semrush.com/blog/community-manager-a-jack-of-all-trades/

ScaDS DRESDEN LEIPZIG

IS THERE A JACK OF ALL TRADES?



https://www.semrush.com/blog/community-manager-a-jack-of-all-trades/

ScaDS DRESDEN LEIPZIG

NATIONAL BIG DATA COMPETENCE CENTER

- Project period: 4 years (10/2014 – 09/2018), option for +3 more years after evaluation
- Many involved research groups + many associated partners
- Focal point for new research activities
- Specialists from computer and domain sciences

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITÄT LEIPZIG

Leibniz-Institut für ökologische Raumentwicklung

Max Planck Institute of Molecular Cell Biology and Genetics

ScaDS DRESDEN LEIPZIG

STRUCTURE OF THE COMPETENCE CENTER

E.Rahm

Life-Sciences

Material Sciences

Environmental and Traffic Sciences

Digital Humanities

Business Data

Service Center

W.E. Nagel

Big Data Life Cycle Management und Workflows

W. Lehner

Data Quality/ Data Integration

Knowledge Extraction

Visual Analysis

S. Gumhold

Efficient Big Data Architecture

K.-P. Fähnrich

M. Bogdan

C. Rother

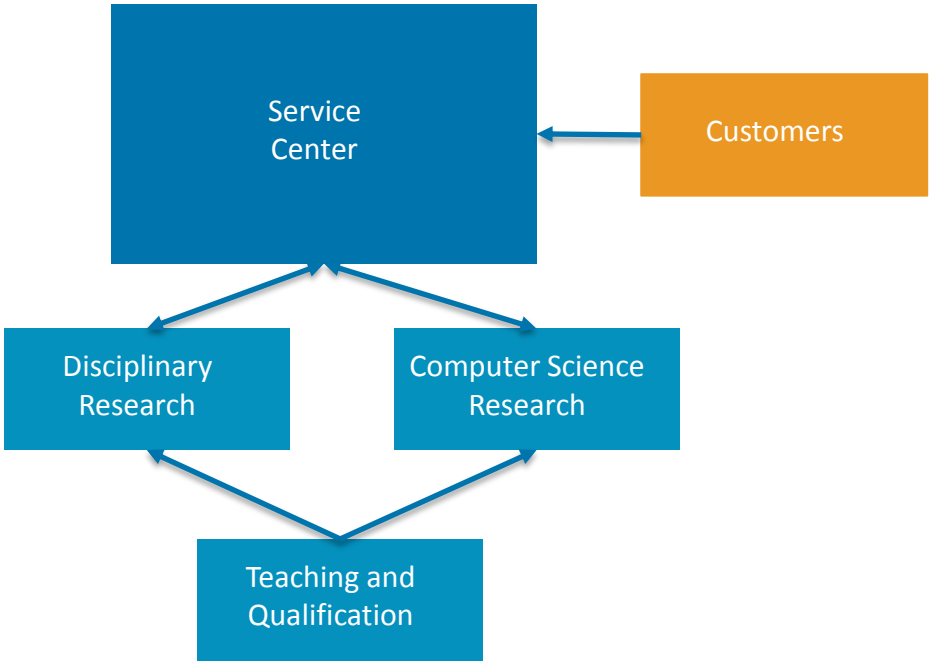
P. Stadler

G. Heyer

G. Scheuermann

SERVICE CENTER AS FOCAL POINT

- Trainings & Education
- Big Data Consulting & Collaborative Research
- Technical Infrastructure & Resources



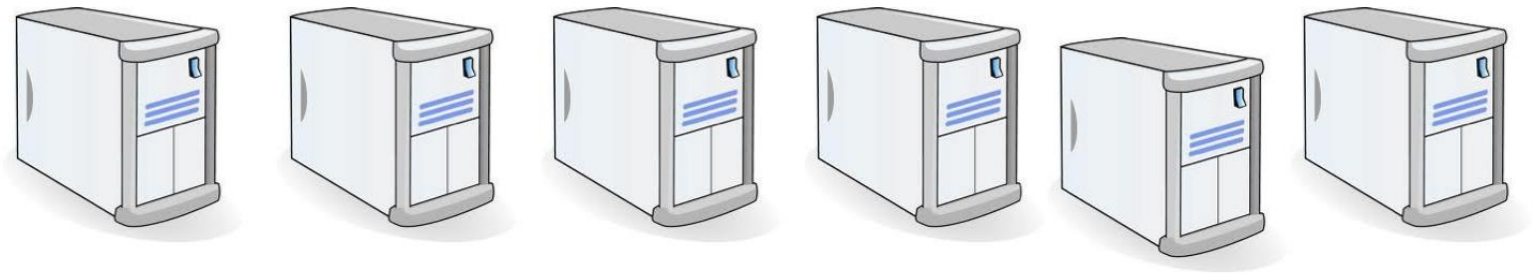
ScaDS
DRESDEN LEIPZIG

SOME HISTORIC REMARKS

ScaDS DRESDEN LEIPZIG

SCALE-OUT

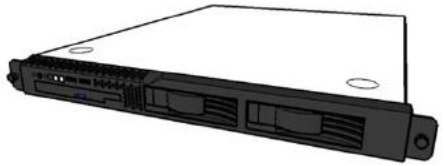
- Many machines (hundreds, thousands)



- As opposed to scale-up, where one very powerful (single) server is used

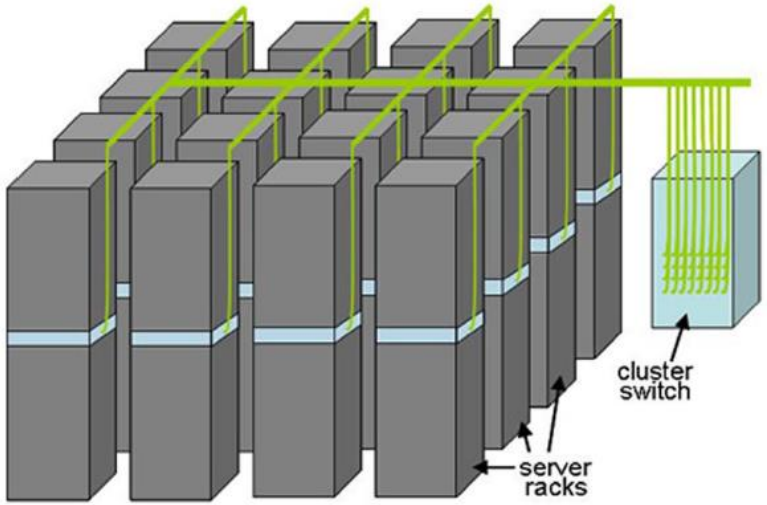
ScaDS DRESDEN LEIPZIG

INFRASTRUCTURE FOR BIG DATA



Server
- CPUs
- DRAM
- Disks

Rack
- 40-80 Server
- Ethernet Switch



Cluster

ScaDS DRESDEN LEIPZIG

EXAMPLE: GOOGLE DATACENTER

Source: http://www.google.com/about/datacenters/gallery/#/all/10

ScaDS
DRESDEN LEIPZIG

First approaches: HADOOP

HOW TO PROCESS AND STORE DATA ON THIS INFRASTRUCTURE?

ScaDS DRESDEN LEIPZIG

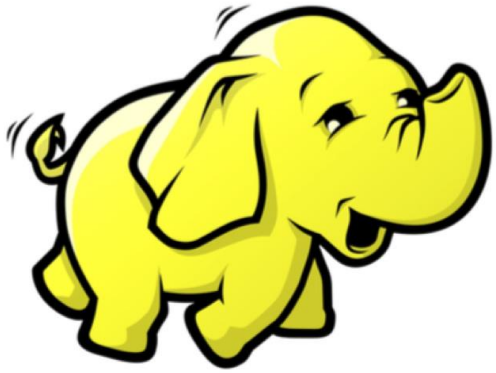
HISTORY

- The Google File System (Sanjay Ghemawat, H. Gobioff, S. Leung. SOSP 2003)
- MapReduce: Simplified Data Processing on Large Clusters (Jeff Dean, S. Ghemawat. OSDI 2004)

ScaDS DRESDEN LEIPZIG

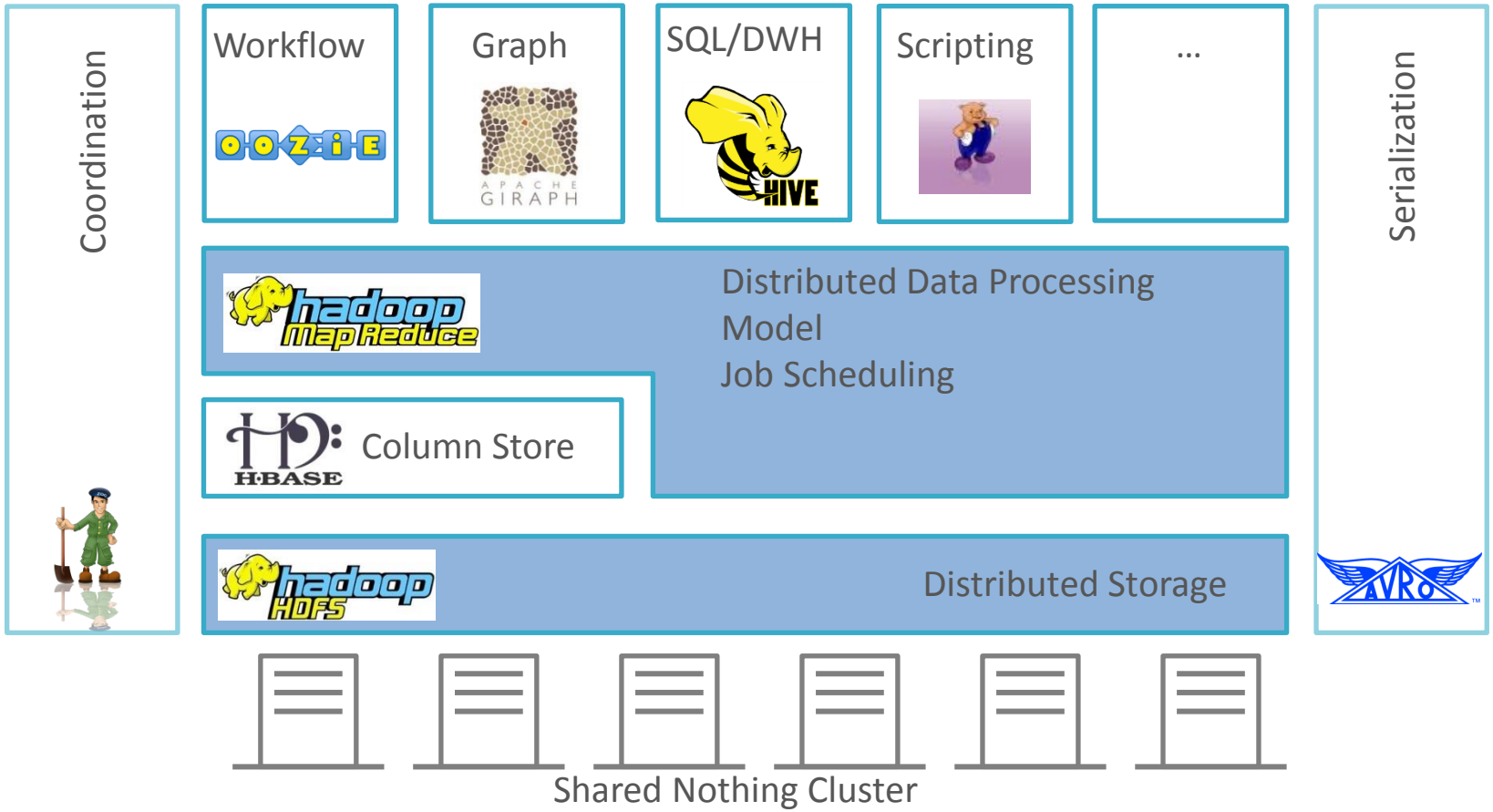
APACHE HADOOP

- Hadoop is open-source software framework
- supports data-intensive distributed applications and clones the Google‘s MapReduce
- designed to process verly large amounts of unstructured and complex data
- designed to run on a large number of machines



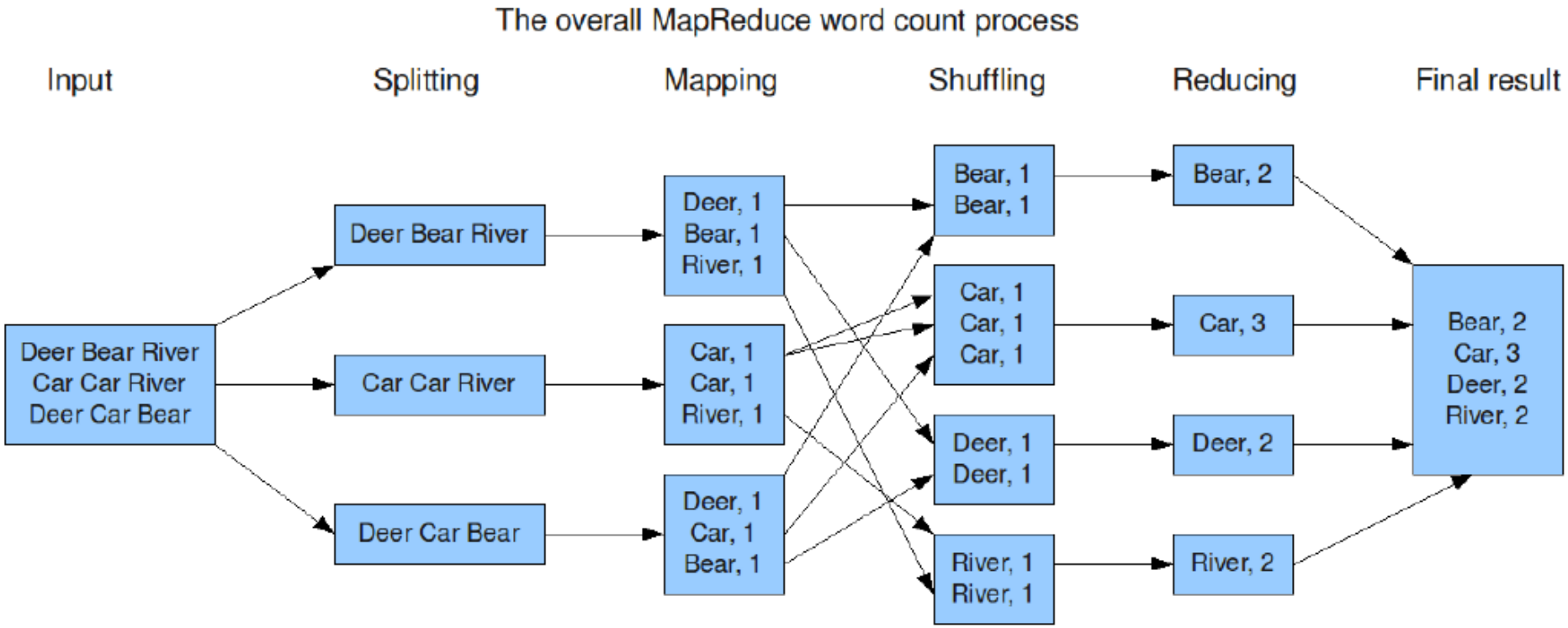
ScaDS DRESDEN LEIPZIG

HADOOP 1.0 ECOSYSTEM



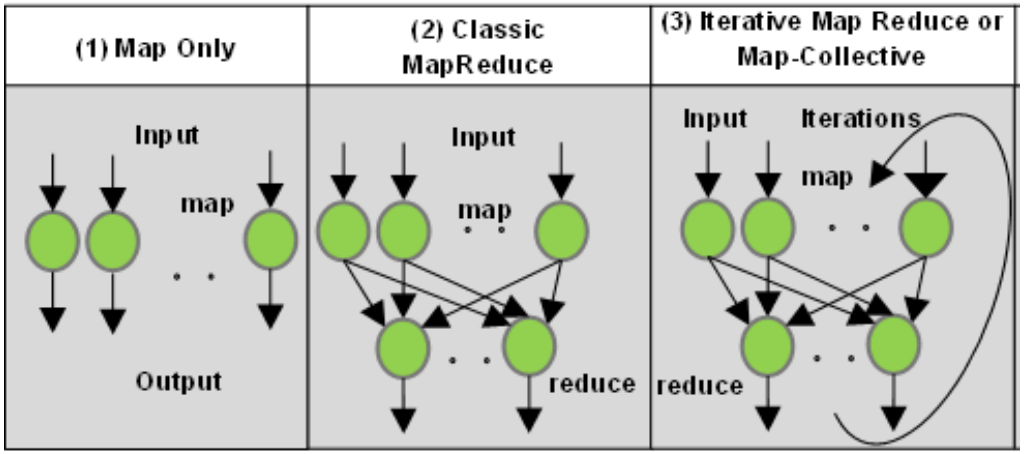
ScaDS DRESDEN LEIPZIG

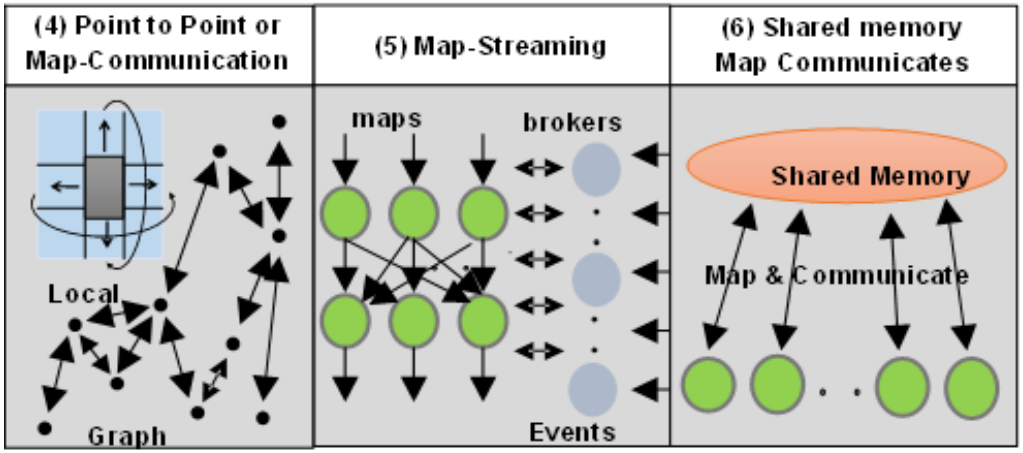
MAP REDUCE EXAMPLE - COUNTING



ScaDS DRESDEN LEIPZIG

EXTENSIONS NEEDED





ScaDS DRESDEN LEIPZIG

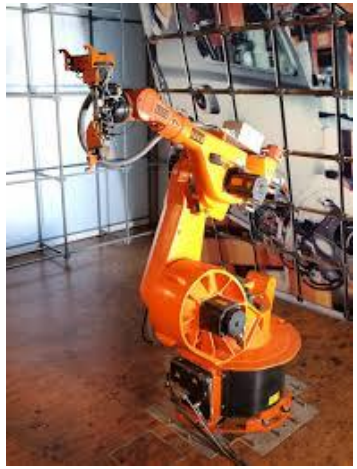
BIG DATA 2.0 PROCESSING PLAFROMS

General Purpose	Big SQL	Big Graph	Big Stream
Spark	cloudera IMPALA	APACHE GIRAPH	STORM
Flink	presto	GraphLab	S4 distributed stream computing platform
	Spark SQL	GraphX	samza
			Spark
			Flink

ScaDS DRESDEN LEIPZIG

DYNAMIC DATA

Availability of sensors & Cyberphysical Systems



USE CASE: TRAFFIC MONITORING

- Toll collection, Speed control, traffic-jam detection, route planning
- Toll stations, smart phones..

ScaDS DRESDEN LEIPZIG

GESTURE DETECTION

- stream of sensor data (30hz)

INTRODUCTION TO SPARK

Overview + Hands On

René Jäkel

www.scads.de

ScaDS
DRESDEN LEIPZIG

TUTORIAL REQUIREMENTS

SPARK DEPLOYMENT ON LINUX

For this tutorial, this will be the base:

- Preparation:
  - Go to spark website and download latest base release: http://spark.apache.org/downloads.html
  - Unpack on your local machine (in user space; hint: use a unique location for this tutorial in your workstation)
  - ***Testing***: run a small example - Pi-estimation

```
$ ./bin/run-example SparkPi 10
```

- Download code skeleton and example data from here:

  https://wwwpub.zih.tu-dresden.de/~jaekel/

  tutorial.zip (source code examples)

  yellow_tripdata_2016-01_10k.csv (…100k – taxi data example)

SPARK WINDOWS DEPLOYMENT

- Install Java JRE:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
  (check environment variables in '**Path**' for '**Java_Home**' on your system
- Install Scala: http://www.scala-lang.org/download/
- Install SBT: http://www.scala-sbt.org/download.html
- Install python 2.7.13 from here: https://www.python.org/downloads/windows/
  and add location to '**PATH**' environment variable
- Add tool "winutils.exe" to path
  - Download from http://public-repo-1.hortonworks.com/hdp-win-alpha/winutils.exe
  - Create a directory on your system, e.g. 'C:\Hadoop' and copy "winutils.exe" into subdirectory 'bin'
  - Set environment variable on your system to point to the actual location (HADOOP_HOME=c:\Hadoop)

SPARK WINDOWS DEPLOYMENT (II)

- Download Spark (evtl. need for zipper, e.g. 7zip, to unpack release)
  - Go to spark website and download latest base release: http://spark.apache.org/downloads.html
  - Create tutorial directory somewhere and unpack release
- Open command line and run example from the release home dir
  - ***Testing***: run a small example - Pi-estimation

```
$ .\bin\run-example SparkPi 10
```

INTRODUCTION TO SPARK

Overview

René Jäkel

www.scads.de

ScaDS DRESDEN LEIPZIG

OVERVIEW

- Motivation/Overview
  - Some historic and general remarks
  - Alternatives
  - Use and scope
- Spark and its core functionalities – examples for data manipulation
  - Running Spark
  - Handling and transforming data
  - RDDs – basic structures (the heart of Spark)
  - Additions to core – SQL example
- Where to go from here?

ScaDS
DRESDEN LEIPZIG

SPARK – A SHORT INTRODUCTION

HISTORY / EVOLUTION OF BIG DATA

2000

- Hardware
  - Disk space cheap (primary storage solutions)
  - Network was costly
  - RAM was very expensive
  - Single core machines were dominant
- Software
  - Object orientation and optimization for single core
  - SQL as primary analysis language; some specific frameworks (Mathlab)

Now (~20a)

- Hardware
  - RAM/Flash got cheaper and faster – tend to become primary storage, disk as fall-back
  - Network is faster / Virtualization
  - Multi core machines are dominating, different architectures
- Software
  - More functional programming and frameworks
  - Multicore-programming and distribution
  - NoSQL alternatives

HISTORY / EVOLUTION OF BIG DATA

First generation of data processing frameworks (Big Data)

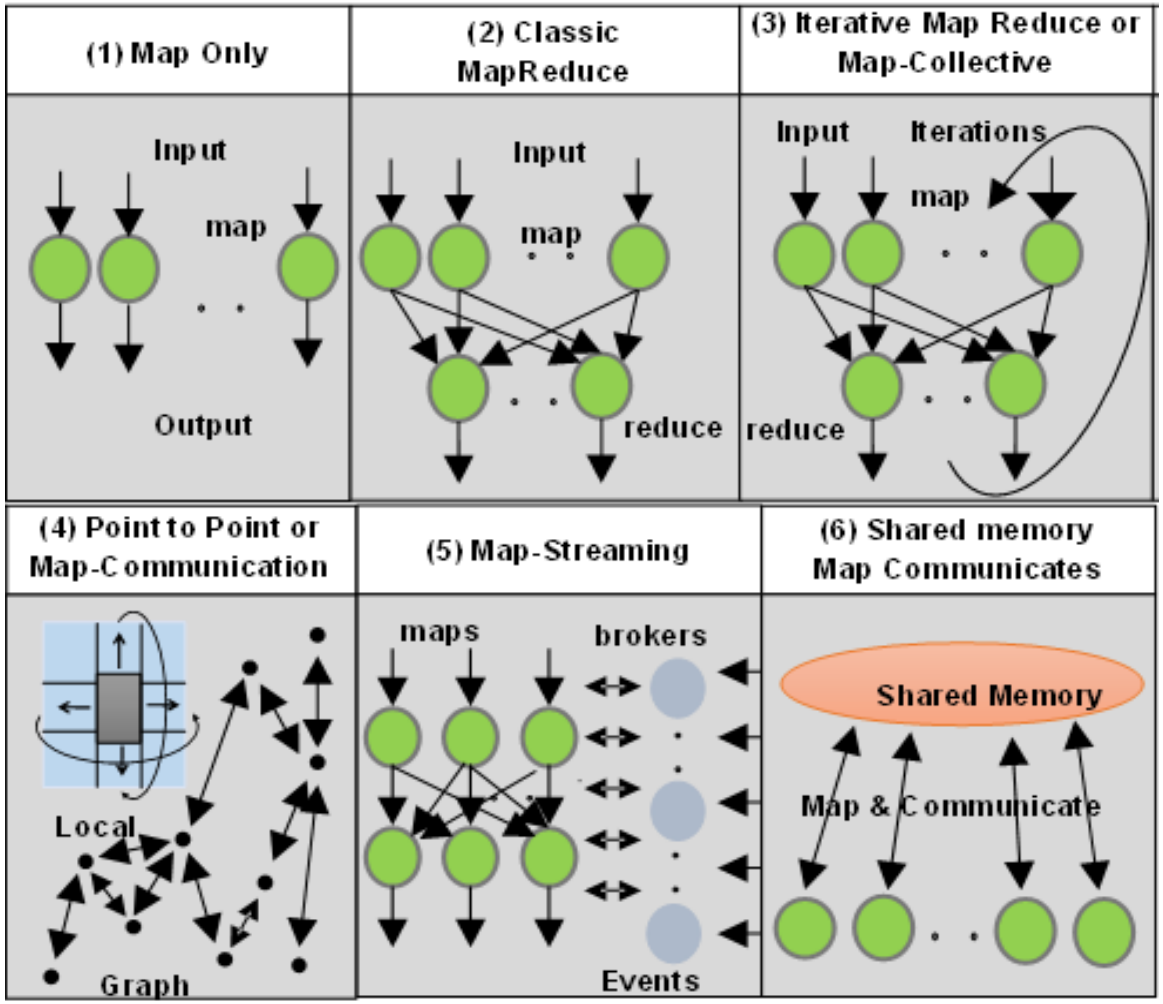
- Purpose/Background
  - ONLY few companies had real big data need
  - Batch processing was still dominant way to distribute workloads
  - Primarily volume was biggest concern (not 5 V's)
  - Mostly used for Search/basic behavior analysis (logging data)
- Hadoop-implementation offered simple programming approach
  - Batch orientated
  - Underlying HDFS for data distribution

HISTORY / EVOLUTION OF BIG DATA

Second generation

- Purpose/Background
  - Most companies have need to use Big Data technologies
  - Velocity now dominating, also “Value”
  - Diverse use cases
    - Real time processing of data
    - Learning approaches, iterations, interactions
- Hadoop-approach not flexible enough any more
  - Need for in-memory processing (handle iterations efficiently)
  - Very flexible hardware options (Shared-Nothing vs. HPC?)

LIMITATIONS OF AND EXTENSIONS TO MAP/REDUCE



Geoffrey Fox:- Lecture: Building a Library at the Nexus of High Performance Computing and Big Data, Indiana University

IS BIG DATA KILLING HPC?

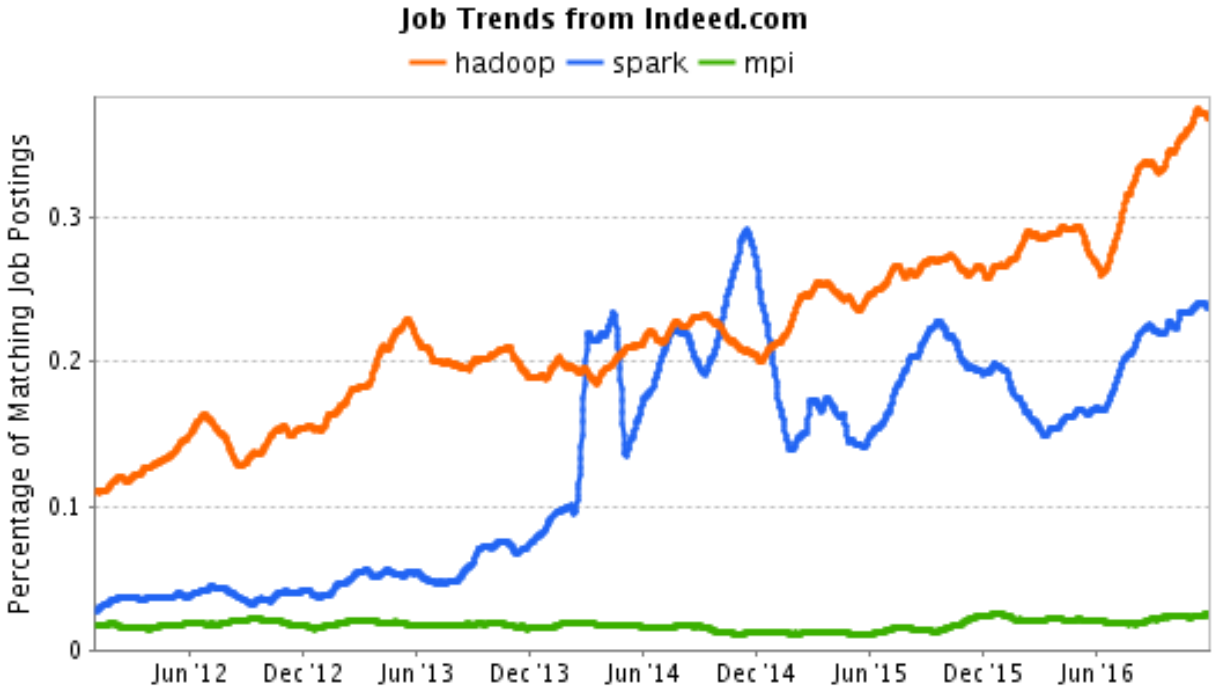
Pictured: The HPC community bravely holds off the incoming tide of new technologies and applications.

- Blog post “HPC is dying, and MPI is killing it” by Jonathan Dursi (from April 2015, see http://www.dursi.ca/hpc-is-dying-and-mpi-is-killing-it/)

Slide courtesy: A. Knüpfer (ZIH), ScaDS Big Data Fall Schlool 2015

IS BIG DATA KILLING HPC?

Data based on Google trends results for MPI, Hadoop, and Spark



- Blog post “HPC is dying, and MPI is killing it” by Jonathan Dursi (from April 2015, see http://www.dursi.ca/hpc-is-dying-and-mpi-is-killing-it/)

Slide courtesy: A. Knüpfer (ZIH), ScaDS Big Data Fall Schlool 2015

ScaDS DRESDEN LEIPZIG

WHERE HPC IS WRONG TODAY

- Jonathan Dursi addresses HPC in general
  - “This should be a golden age for High Performance Computing.” But it is not. Instead new technologies are developed by other communities.
  - Analysis of Internet data and DNA sequencing brought huge amounts of data in new areas. Why wasn’t HPC the logical solution?
  - Prevailing “Not invented here” or “this is not real HPC” attitudes.
  - HPC stayed with traditional concepts largely, both in hardware and software.
  - Other communities developed their own solutions, re-inventing several wheels, producing many successful new technologies and software.

Slide courtesy: A. Knüpfer (ZIH), ScaDS Big Data Fall Schlool 2015

WHERE HPC IS WRONG TODAY

- Jonathan Dursi addresses HPC in general
  - Used to be the “killer app” in HPC, particularly because it was a firm standard for 27 years
  - Very high quality implementations, highest speed, constantly adapting to newest hardware.
  - But also bloated and inflexible
  - Just left no room for alternatives inside the HPC community:
    - Chapel, X10, UPC, CoArray Fortran, Java, Scala, Python
    - GASPI and OpenSHMEM
  - Very slow and difficult standardization process today.
    - Quite backward oriented
    - stuck with 32bit variables
    - fault tolerance solution on the horizon

Slide courtasy: A. Knüpfer (ZIH), ScaDS Big Data Fall Schlool 2015

WHERE HPC IS WRONG TODAY

- Compare implementation effort for 1D diffusion simulation in lines of code
- Very simple example in MPI, Spark, and Chapel by Jonathan Dursi

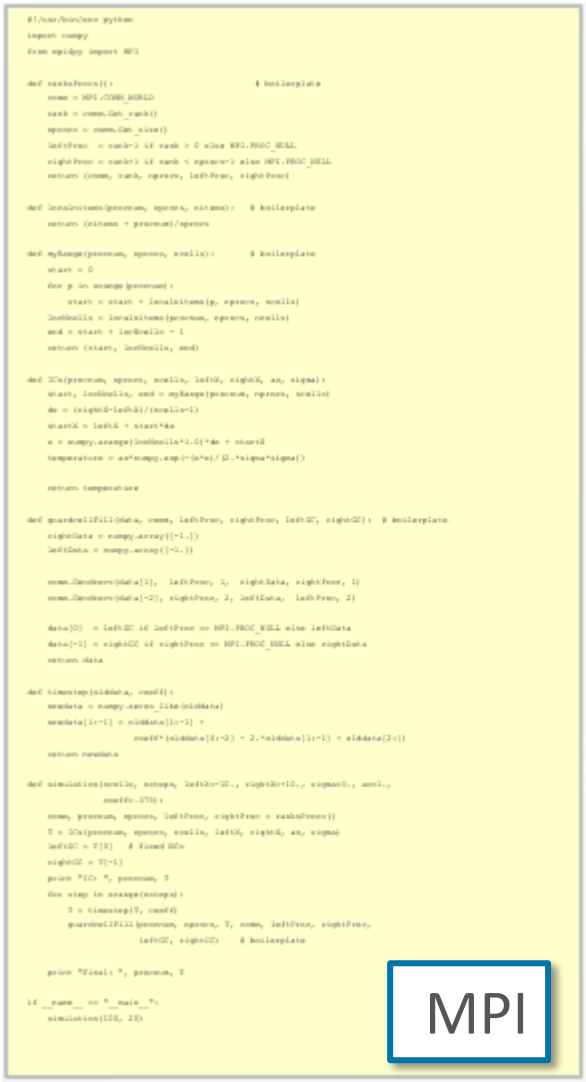
Framework	Lines	Lines of Boilerplate
MPI+Python	52	20+
Spark+Python	28	2
Chapel	20	1

- (The Spark version is also fault-tolerant)
- (The Chapel version includes command line parameter parsing)
- Change of data distribution requires
  - Complete rewrite in MPI
  - Change hash function in Spark
  - Change declaration in Chapel

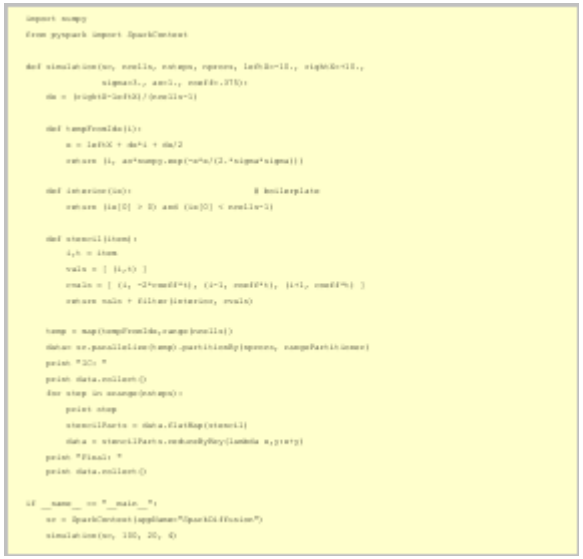
Slide courtesy: A. Knüpfer (ZIH), ScaDS Big Data Fall Schlool 2015

ScaDS DRESDEN LEIPZIG

WHERE HPC IS WRONG TODAY



MPI



Spark

Chapel

Slide courtesy: A. Knüpfer (ZIH), ScaDS Big Data Fall Schlool 2015

ScaDS
DRESDEN LEIPZIG

SPARK OVERVIEW

HISTORY / EVOLUTION OF BIG DATA

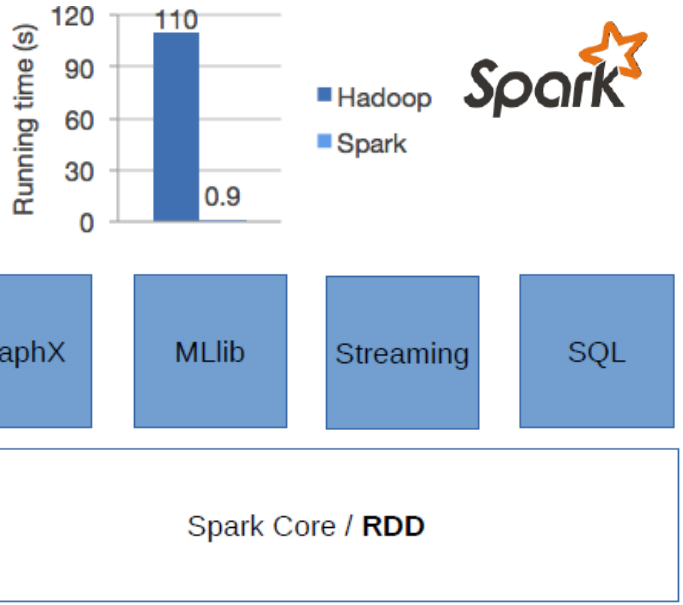
Third generation – multi purpose frameworks and large diversity of analytics environments, e.g.:

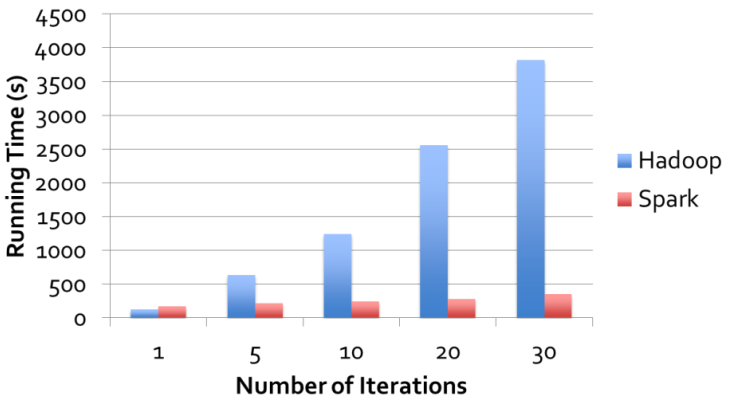
- Apache Spark
  - A fast and general engine for large scale data processing
  - Created by AMPLab (Berkeley/2009) now Databricks
  - Written in Scala and licensed under Apache Foundation
  - Strong international developer community and organized as open source

Blog post about history of Spark by Madhukara Phatak:
http://blog.madhukaraphatak.com/history-of-spark/

SPARK OVERVIEW

- Apache Spark offers Directed Acyclic Graph (DAG) execution engine that supports cyclic data flow and in-memory computing
- Offers over 80 high-level operators to build parallel applications; use it interactively from the Scala, Python and R shells
- Offers stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming; combine libraries seamlessly in the same application
- Run Spark standalone or in cluster mode, on EC2, on Hadoop YARN, or on Apache Mesos; access data in HDFS, Cassandra, HBase, Hive, Tachyon, and any Hadoop data source



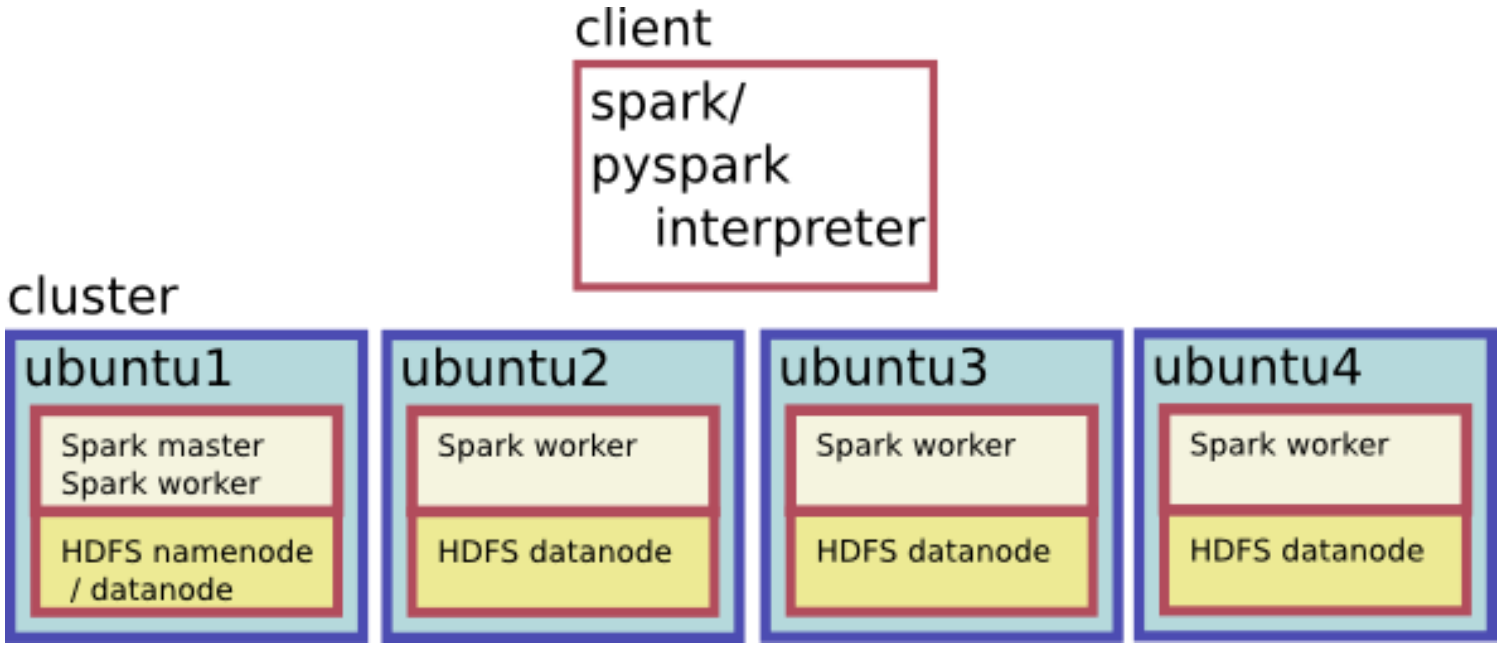


SPARK OVERVIEW

Multi language API

- Now completely written in Scala but versatile API offered
  - Scala
  - Java
  - Python
  - R-interface
- SparkSQL provides tool to analyze data in SQL-like manner
- Supports batch-, stream-, and in-memory-processing
- Fully integrate with Hadoop and HDFS

SPARK ON A HADOOP-LIKE CLUSTER



- Deployment
  - Local / Standalone
  - On Hadoop-Cluster, Yarn
  - Mesos, EC2

SPARK IN LOCAL MODE

- For this tutorial, this will be the base:
- Preparation:
  - Go to spark website and download latest base release: http://spark.apache.org/downloads.html
  - Install on your local machine (in user space, platform independent)
  - ***Exercise***: run a small example - Pi-estimation

```
$ ./bin/run-example SparkPi 10
```

- Have a look into example subdirectory
- ***Exercise:*** run a different example of your own choice from the base package
- Have a look in web frontend after starting Spark http://localhost:4040

SPARK CONFIGURATION

- You might want to change general behaviour of Spark settings in configurations
- E.g. alter log-level for output
- Go to ./config‘- sub-directory

```
cp conf/log4j.properties.template conf/log4j.properties
```

- Change log level:

  log4j.rootCategory=INFO, console

  to:

  log4j.rootCategory=WARN, console
- Re-run small example and watch output

INTERACTING WITH SPARK

- Quick intro into python programming interface
- Running the Python-Spark shell on 2 cores

```
$./bin/pyspark –master=local[2]
```

- To quit, use ***quit()*** function call
- Simple and efficient ways to try things in quick-mode and interactively
- **Exercise**: write very simple python example and execute via spark
  - Make a local working directory e.g. tutorial
  - Write a minimal python “Hello World” example:'
  - Run your first example using ‘***spark-submit***’ with your python input file as argument (exchange pyspark-executable with ‘spark-submit’)

INTERACTING WITH SPARK

- Quick intro into python programming interface
- Running the Python-Spark shell on 2 cores

```
$./bin/pyspark –master=local[2]
```

- Make a local working directory e.g. tutorial
- Write a minimal python “Hello World” example:
- Run your first example (watch output)

```
$./bin/spark-submit –master=local[2] tutorial/hello.py
```

```
Hello, world!
```

TEXT HANDLING EXAMPLE

- First analysis example to find interesting tags in text-based documents
- Find some text file on your system
- Running the Python-Spark shell with 2 cores in interactive mode:

```
$./bin/pyspark –master=local[2]
```

- Load text file via spark context

```
localInputFile = "myLocalFile.txt"
logData = sc.textFile(localInputFile)
```

- Apply filter-transformation on input (filter for some string pattern)

```
letterAs = logData.filter(lambda s: 'a' in s).count()
```

What is your observation?

transformation

action

TEXT HANDLING EXAMPLE

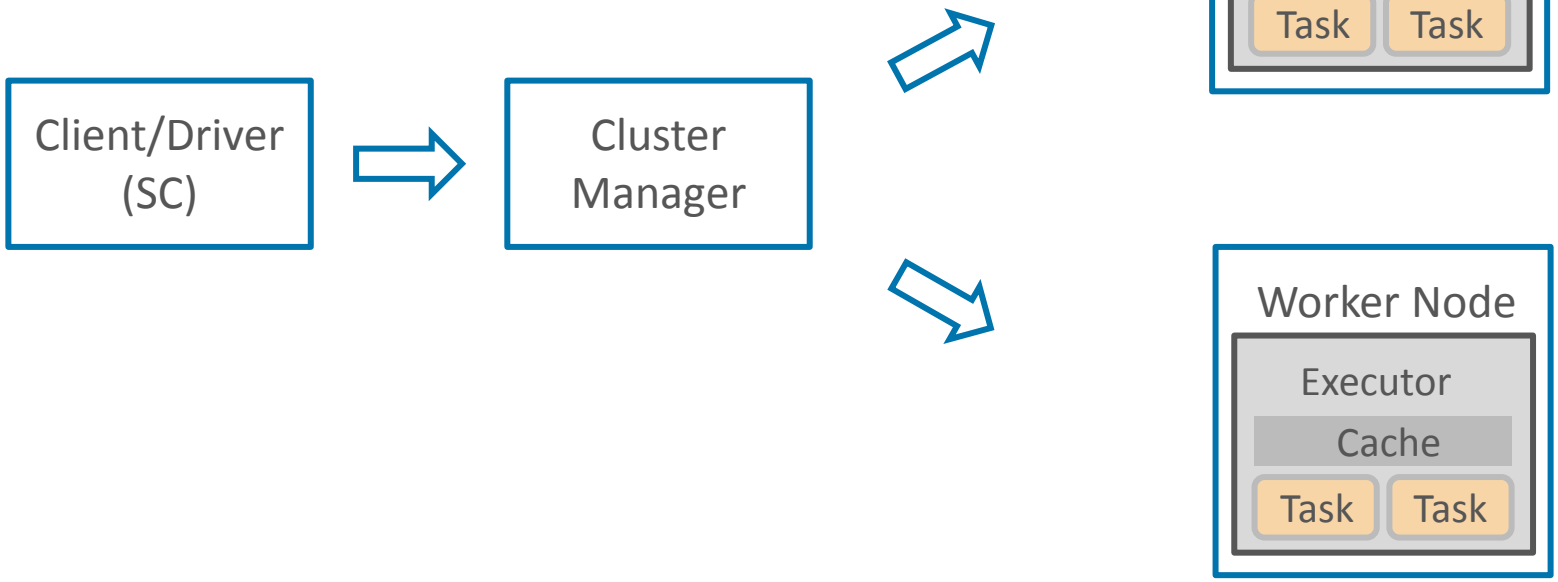
- **Exercise**: transform test code to external python program
  - Need to include ‘SparkContext’
    - Add explicit definition of context object

```
from pyspark import SparkContext

localInputFile = "./myLocalFile"  # some file on your system

sc = SparkContext("local", "Simple App")
logData = sc.textFile(localInputFile)

#…some more code here…

sc.stop()
```

SPARK - CONTEXT

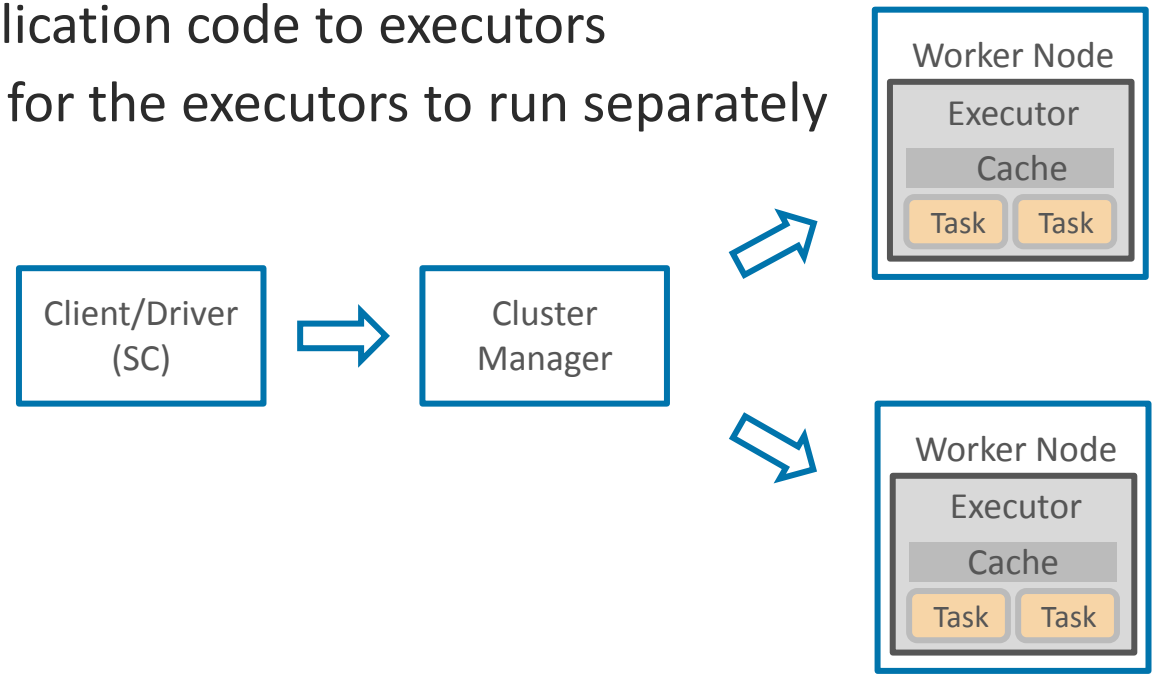
- Connection of the client to the available cluster
  - Master defines cluster connection
  - Local / Spark standalone / Mesos / EC2

```
./bin/spark-shell -master=mesos://HOST:PORT myAPP
```



SPARK - CONTEXT

- Master workflow
  - Connects to a cluster manager which allocates resources across applications
  - Acquires executors on worker nodes to processes computations on stored data
  - Sends driver application code to executors
  - Distributes tasks for the executors to run separately



TEXT HANDLING EXAMPLE

Solution

- Simple text example

```
./bin/spark-submit --master=local[2] ../02_SimpleApp.py
```

```
"""SimpleAppV1.py"""
from pyspark import SparkContext

localInputFile = "myLocalFile.txt"  # Should be some file on your
system
sc = SparkContext("local", "Simple App")
sc.setLogLevel("WARN")
logData = sc.textFile(localInputFile)

numAs = logData.filter(lambda s: 'a' in s).count()
numBs = logData.filter(lambda s: 'b' in s).count()

print("Lines with 'a': %i, lines with 'b': %i" % (numAs, numBs))
```

READING IN COLLECTION OF FILES

- **Exercises**: update program to load collection of files
  - E.g. from logging directory `'/var/log/'`
    - 1) all files with .log extension
    - 2) all files in given directory
  - Add elements to data distribution

```
addInputFile = "/var/log/kern.log"
logDataAll = sc.textFile(addInputFile)
```

  - Or by simply using python notation

```
sc.textFile("/my/dir1,
            /my/paths/part-00[0-5]*,
            /another/dir,
            /a/specific/file")
```

READING IN COLLECTION OF FILES

Solution

- **Exercises**: update program to load collection of files
  - E.g. from logging directory `'/var/log/'`
    - 1) all files with .log extension
    - 2) all files in given directory

```
localInputFile = "/var/log/syslog"  # some file on your system
sc = SparkContext("local", "Simple App")
logDataAll = sc.textFile(localInputFile)

addInputFile = "/var/log/kern.log"
logDataAll = sc.textFile(addInputFile)
#logDataAll = sc.textFile("/var/log/*.log")

numKs = logDataAll.filter(lambda s: 'kernel' in     s).count()
numWs = logDataAll.filter(lambda s: 'warning' in    s).count()

print("Lines with kernel: %i, lines with warnings: %i" % (numKs, numWs))
sc.stop()
```

ScaDS
DRESDEN LEIPZIG

SPARK – RDD

ScaDS DRESDEN LEIPZIG

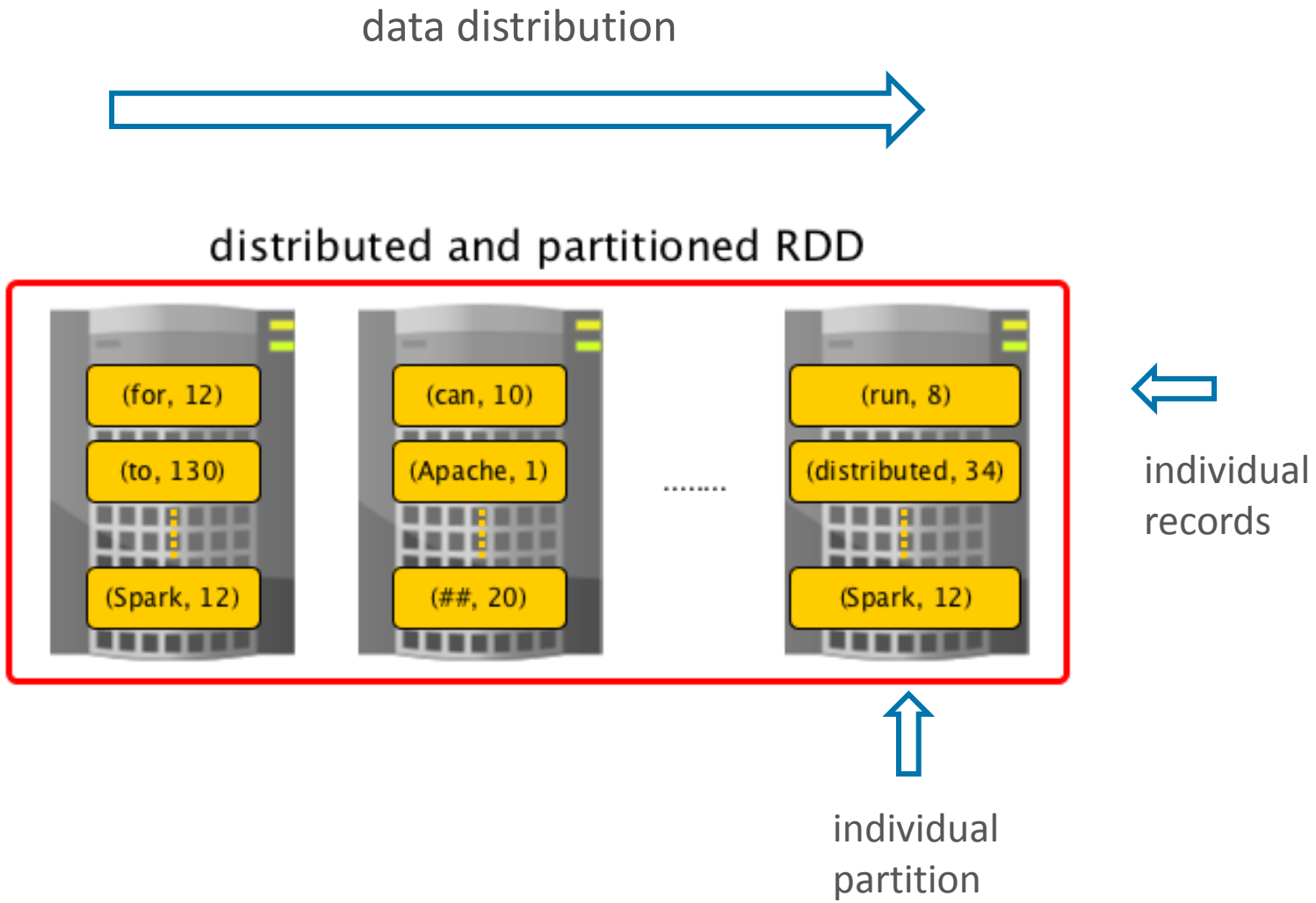
RDD – RESILIENT DISTRIBUTED DATASET

- RDD is a resilient and distributed collection of records
  - **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph and so able to recompute missing or damaged partitions due to node failures
  - **Distributed** with data residing on multiple nodes in a cluster.
  - **Dataset** is a collection of partitioned data with primitive values or values of values, e.g. tuples or other objects
- Represents core data manipulation methods of Spark
  - **immutable, partitioned** collection of elements that can be operated on in parallel
  - **distributed memory abstraction** that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner
- Motivation: previous computing frameworks handle specific patterns inefficiently
  - **Iterative** algorithms in machine learning and graph computations
  - **Interactive** data mining tools as ad-hoc queries on the same dataset (repetitions)

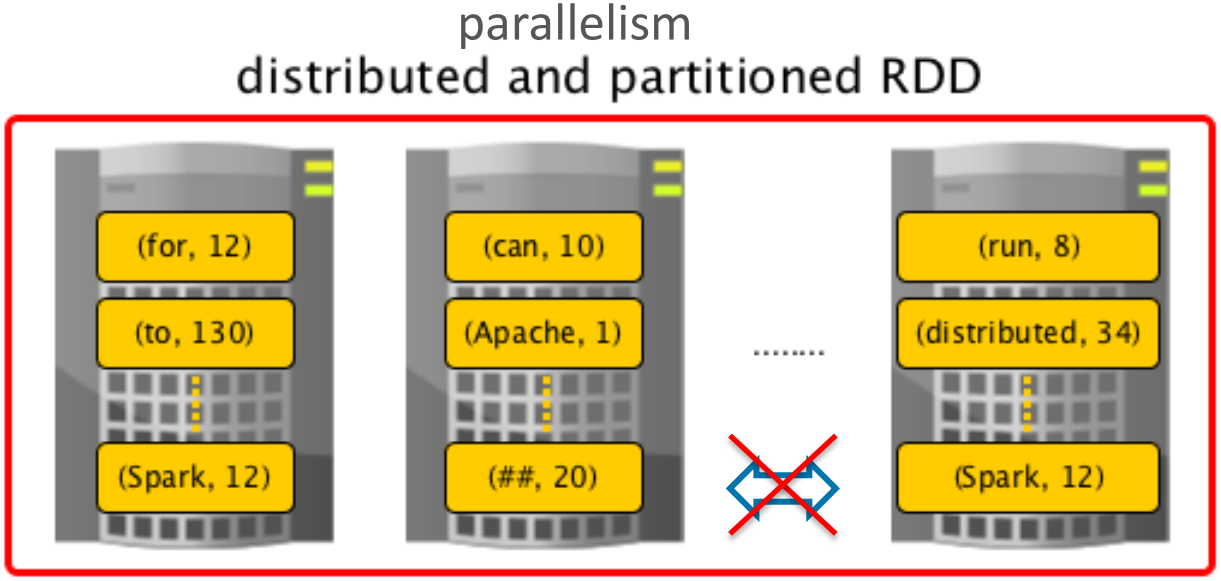
RDD – RESILIENT DISTRIBUTED DATASET

- RDD and its basic features:
  - **In-Memory -** RDD-data is stored in memory if possible (size/time)
  - **Immutable** or **Read-Only**, i.e. it does not change once created and can only be transformed using transformations to new RDDs.
  - **Lazy evaluated**, i.e. the data inside RDD is not available or transformed until an action is executed that triggers the execution.
  - **Cacheable**, i.e. you can hold all the data in a persistent "storage" like memory (default and the most preferred) or disk (the least preferred due to access speed).
  - **Parallel**, i.e. process data in parallel.
  - **Typed**, i.e. values in a RDD have types, e.g. RDD[Long] or RDD[(Int, String)].
  - **Partitioned**, i.e. the data inside a RDD is partitioned (split into partitions) and then distributed across nodes in a cluster (one partition per JVM that may or may not correspond to a single node).

RDD – RESILIENT DISTRIBUTED DATASET



RDD – RESILIENT DISTRIBUTED DATASET



- Parallelism controlled by RDD using ‘repartition’ method
- Spark tries to distribute computing to already partitioned data, avoiding data exchange
- RDDs support two kinds of operations:
  - transformations - lazy operations that return another RDD
  - actions - operations that trigger computation and return values

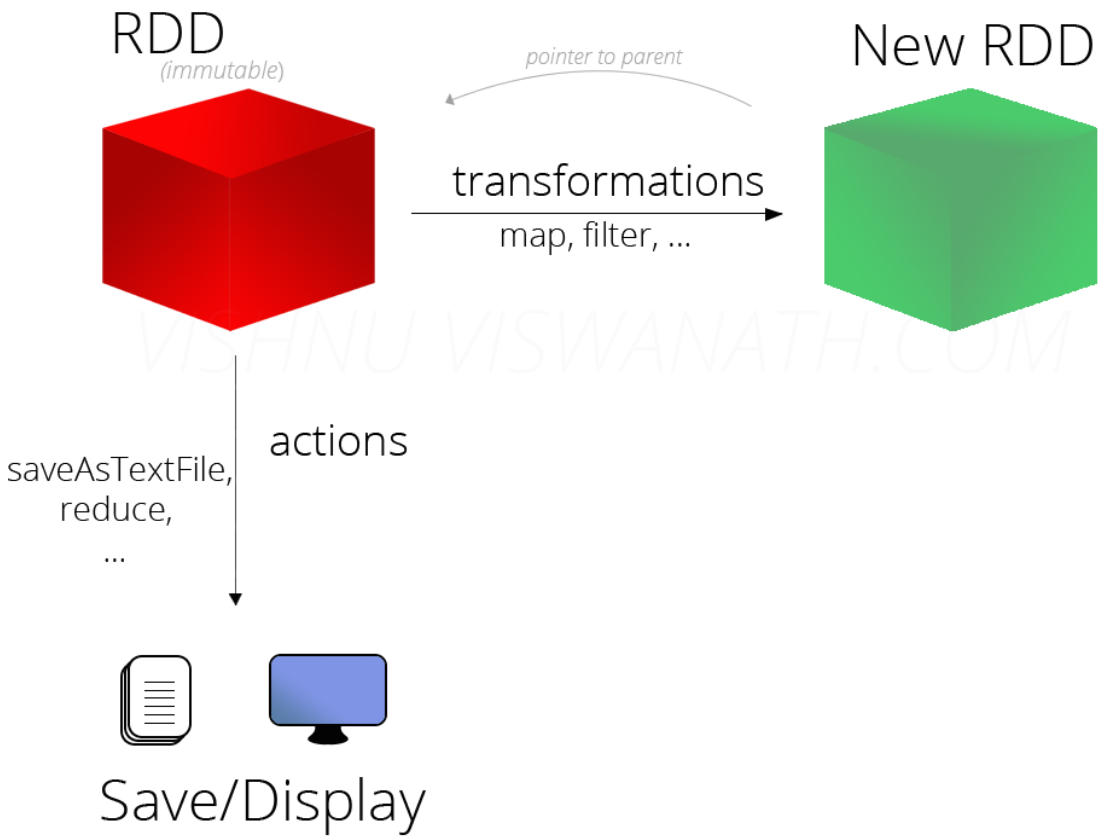
RDD – RESILIENT DISTRIBUTED DATASET

- Start Sparks python shell and create a RDD

```
>>> data = [1, 2, 3, 4, 5]
>>> data
[1, 2, 3, 4, 5]

>>> distData = sc.parallelize(data)
>>> distData
ParallelCollectionRDD[0] at parallelize at
PythonRDD.scala:475
```

RDD – RESILIENT DISTRIBUTED DATASET

- "Actions" and "Transformations"



RDD – TRANSFORMATIONS AND ACTIONS

- Transformations: http://spark.apache.org/docs/latest/programming-guide.html#transformations
- Actions http://spark.apache.org/docs/latest/programming-guide.html#actions

RDD – RESILIENT DISTRIBUTED DATASET

- **Exercises**: map input to some function
- In pyspark-shell:

```
distFile = sc.textFile("test.txt") #creates RDD of lines of text
distFile.map(lambda x: x.split(' ')).collect()
distFile.flatMap(lambda x: x.split(' ')).collect()
```

  - Redo with LogLevel set to "INFO" and watch output (`sc.setLogLevel("INFO")`)
- What is the `collect()` – action doing – run with and without this action
- What is the difference between `map()` and `flatMap()` ?

RDD – RESILIENT DISTRIBUTED DATASET

- Simple live-mapping example:
  - In pyspark-shell:

```
distFile = sc.textFile("README.md")
d = distFile.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1))

from operator import add
d.reduceByKey(add).collect()
# or via lambda definition
# d.reduceByKey(lambda x,y: x+y)
```

  - **Exercise**: find most common word and sort output in descending order
    - Go to Spark API – Documentation and find definition of ***sortBy()*** method of RDD
    - Hint: https://spark.apache.org/docs/2.1.0/api

RDD – RESILIENT DISTRIBUTED DATASET

- Simple live-mapping example :
  - In pyspark-shell:

Solution

```
distFile = sc.textFile("README.md")
d = distFile.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1))

from operator import add
d.reduceByKey(add).collect()
# or via lambda definition
# d.reduceByKey(lambda x,y: x+y)
```

  - **Exercise**: find most common word and sort output in descending order
    - Go to Spark API – Documentation and find definition of ***sortBy()*** method of RDD

```
rdd.reduceByKey( lambda x, y: x+y).sortBy(lambda x: x[1],
                                ascending=False).collect()
```

ScaDS
DRESDEN LEIPZIG

RDD ON NUMERIC DATA

RDD – RESILIENT DISTRIBUTED DATASET

- Download parts of the ’KDD data set’
  - From source:

```
wget
http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz
```

  - Or direct from within spark-shell:

```
import urllib
f = urllib.urlretrieve("http://kdd.ics.uci.edu/databases/
kddcup99/kddcup.data_10_percent.gz", "kddcup.data_10_percent.gz")
```

- Create a RDD from input

```
data_file = "./kddcup.data_10_percent.gz"
raw_data = sc.textFile(data_file)
```

- Count all elements using the count() action and take first 5 elements

```
raw_data.count()
raw_data.take(5)
```

RDD – RESILIENT DISTRIBUTED DATASET

- **Exercise**: Write a program and apply a filter-transformation on the label ‘normal’ and count elements of the new RDD

```
normal_raw_data = raw_data.filter(lambda x: 'normal.' in x)
```

- Use skeleton script for additions (*04_rdd_basics_skeleton.py*)
- Compare number of elements in both datasets
- What’s your observation?

RDD – RESILIENT DISTRIBUTED DATASET

- Sampling-Transformation:
  - “simple” statistical method for approximate results, e.g. statistical ML
  - Three parameters: with/without replacement; sample size as fraction; random seed

```
raw_data_sample = raw_data.sample(False, 0.1, 1234)
```

- **Exercise**: Get an approximation (e.g. 10%) of the amount of ‘normal’-datasets in distribution and apply sampling method on dataset
- Count elements in sample and measure time for this action
- Use time function to measure difference in counting elements of different datasets (sampled, not-sampled)
- Use skeleton: ***05_rdd_sampling_skeleton.py***
- What is your observation?

RDD – RESILIENT DISTRIBUTED DATASET

- Hints:

```
rawData = sc.textFile(data_file)
rawData_sample = rawData.sample(False, 0.1, 1234)
sample_size = rawData_sample.count()
total_size = rawData.count()

print("Sample size is {} of {}").format(sample_size, total_size)
# transformations to be applied
sample_normal_tags = rawData_sample.map(lambda x:
x.split(",")).filter(lambda x: "normal." in x)

from time import time
t0 = time()
sample_normal_tags_count = sample_normal_tags.count()
tt = time() - t0

sample_normal_ratio = sample_normal_tags_count / float(sample_size)
print "The ratio of 'sample' interactions is
{}".format(round(sample_normal_ratio,3))
print "Count done in {} seconds".format(round(tt,3))
```

ScaDS
DRESDEN LEIPZIG

SPARK SQL

SPARK SQL

- Spark SQL is a Spark module for structured data processing
- Provide framework with more information about the structure of both the data and the computation being performed
  - Used internally for optimization
- Primary usage: execute SQL queries
- Basis of Spark SQL are ‘data frames’
  - Distributed collection of data
  - new interface since Spark 1.6 (couples benefits of RDDs (strong typing, ability to use powerful lambda functions) with optimized SQL-execution engine
  - API is available in Scala and Java, modules for R and Python

SPARK SQL

- Documentation for Python
- http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html

- New data set: NYC taxi transportation data – aka TLC Trip Record Data
  - Go to webseite: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

SPARK SQL – EXAMPLE

- Download smaller test data sample from online location
- Have a look into code example ***'sql_basics.py'***
- Run the code and watch output:
- Create data frame via:

```
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

SPARK SQL – EXAMPLE MANIPULATIONS

- Print the schema of the data frame and show elements
  (have a look into documentation for function description)

```
taxiDF.groupBy().avg().collect()
taxiDF.dtypes
taxiDF.schema
taxiDF.groupBy("ID").count().show()
```

- **Exercise**: correct and extend the input RDD to get also the average amount of tip per trip
- Some basic SQL-statements

```
taxiDF.registerTempTable("taxi")
sqlContext.sql("SELECT ID, COUNT(*) FROM taxi GROUP BY
ID").show()
```

ScaDS
DRESDEN LEIPZIG

WHERE TO GO FROM HERE?

ScaDS DRESDEN LEIPZIG

GARTNER: BIG DATA TRENDS FOR BI

No. 1: By 2020, information will be used to reinvent, digitalize or eliminate 80% of business processes and products from a decade earlier.

No. 2: By 2017, more than 30% of enterprise access to broadly based big data will be via intermediary data broker services, serving context to business decisions.

No. 3 By 2017, more than 20% of customer-facing analytic deployments will provide product tracking information leveraging the IoT.

ScaDS DRESDEN LEIPZIG

WHAT DO THE BLOGGERS SAY?

„8 New Big Data Projects To Watch“

„Sneak peek – latest trends in Big Data Analytics“

„5 Big Data Technology Predictions for 20XX“

„Five Big Data Trends for 20XX“

A Shift Towards Data-Driven Cultures

ScaDS DRESDEN LEIPZIG

UPCOMING TRENDS

- Big Data Platform Consolidation
- A Shift Towards Data-Driven Cultures
- Owning Up to Your Own Identity – Claiming Your Personal Data
- Data Agility Emerges as a Top Focus
- Organizations Move from Data Lakes to Processing Data Platforms
- Self-Service Big Data Goes Mainstream
- Hadoop Vendor Consolidation: New Business Models Evolve
- Enterprise Architects Separate the Big Hype from Big Data (High availability, mission creatical needs)
- Big Data Security Analytics Gaining Traction
- Time to Experiment with Data Lakes
- Explosion of Demand for Big Data Talent
- Predictive analtytics
- Machine learning / Deep Learning
- Analytics in NLP

ScaDS DRESDEN LEIPZIG

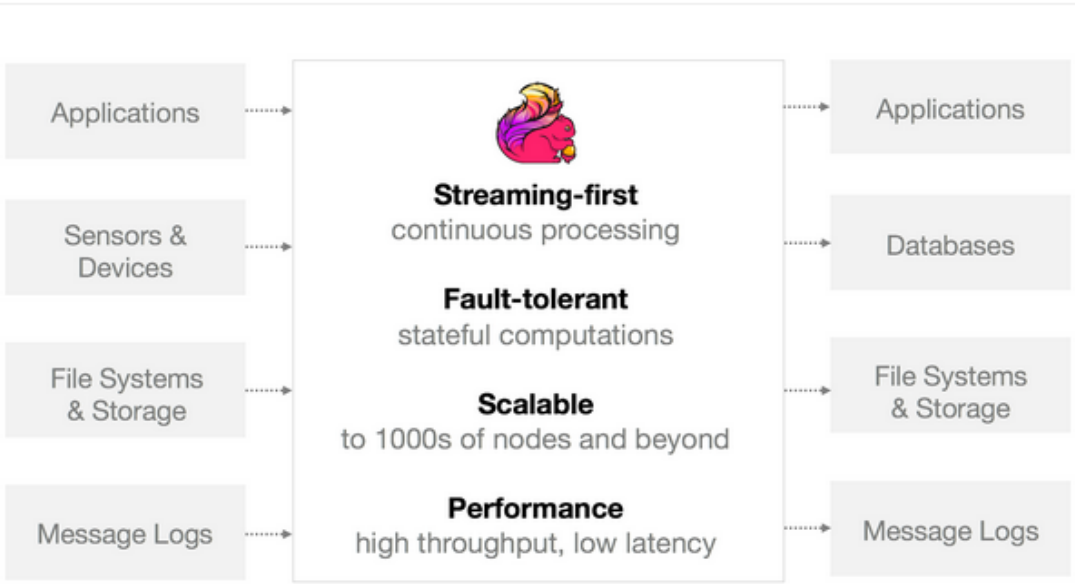
ROAD AHEAD TECHNOLOGY PROLIFERATION!

TOOLS AND EXPERTISE

- Lots of resources and projects available for data analytics
  - Not just the Hadoop-style, also additional modern frameworks
- Apache.org lists 37 projects with tag ‘big data’

Apache Flink® is an open-source stream processing framework for **distributed, high-performing, always-available,** and **accurate** data streaming applications.

Introduction to Flink

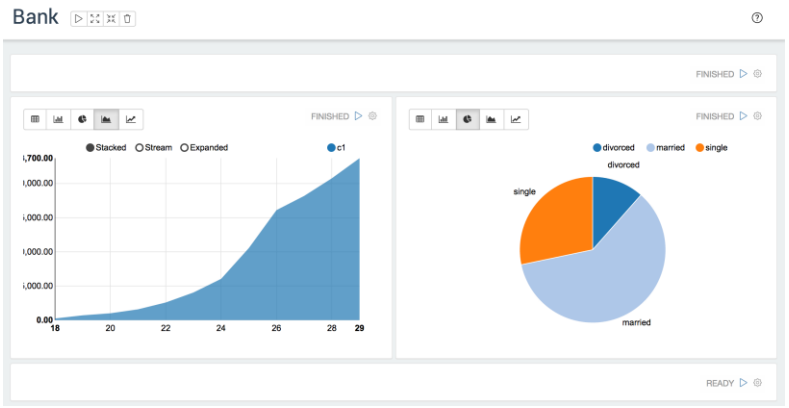


ScaDS DRESDEN LEIPZIG

ROAD AHEAD: SIMPLIFICATION

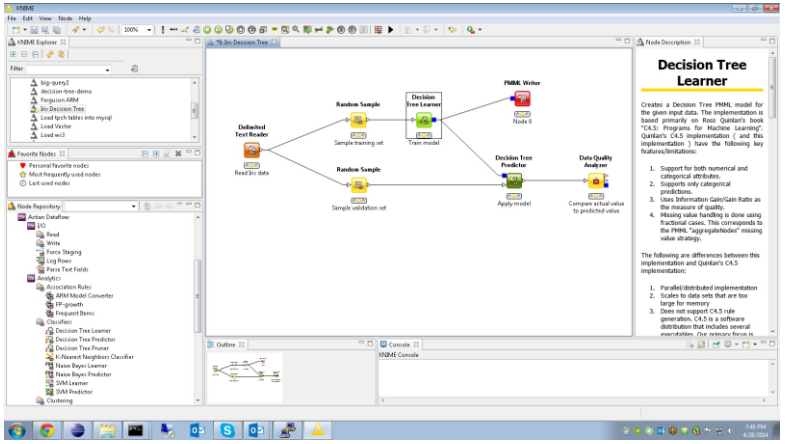
Apache Zeppelin

- Data visualization
- spark integration
- Collaboration



KNIME

- Visual programming of Big Data Workflows



ROAD AHEAD: SIMPLIFICATION(2)

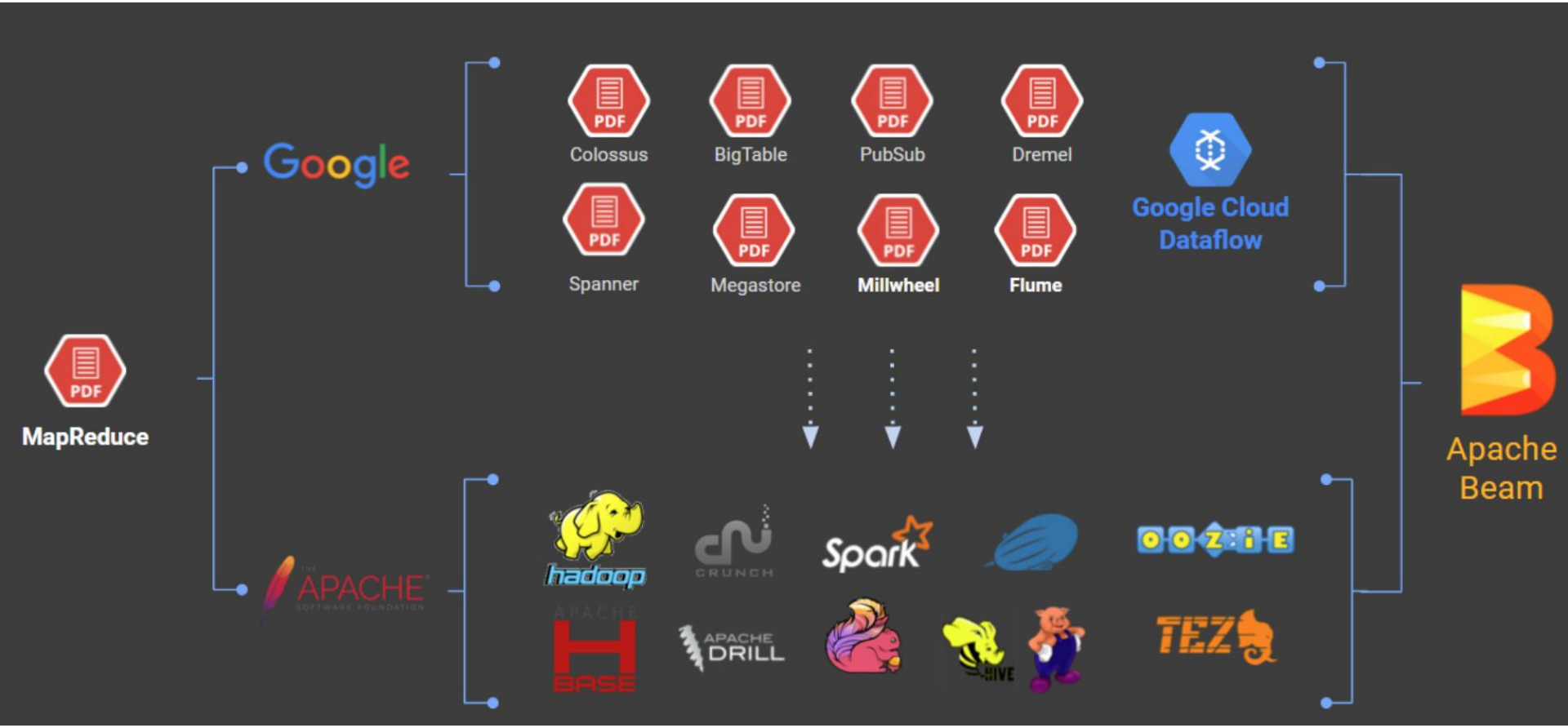
Docker for Cluster/Task Management

Docker is an open-source project started 2013, combining existing technologies for a **lightweight virtualization** with **modularization** and **portability** in mind.

“If it can run on the host, it can run in the container”
(Source: Docker Introduction Slides November 2013)

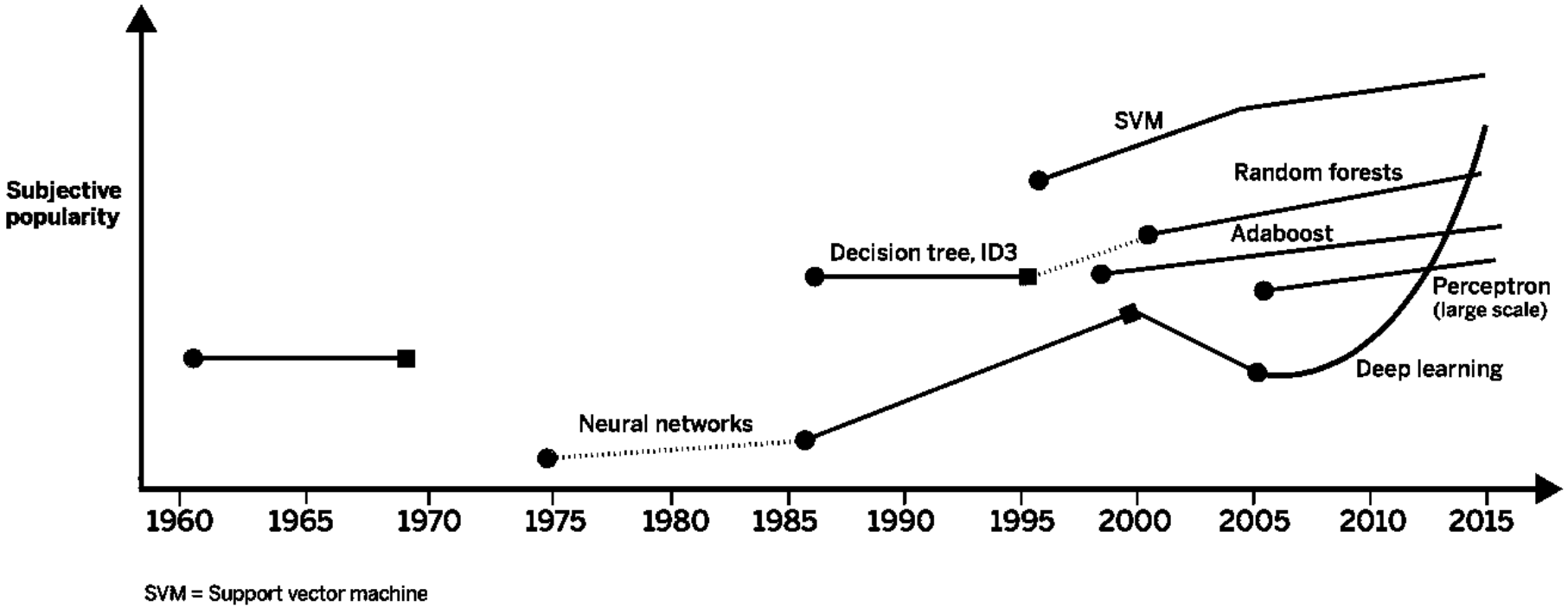
ScaDS DRESDEN LEIPZIG

ROAD AHEAD: UNIFICATION



ScaDS DRESDEN LEIPZIG

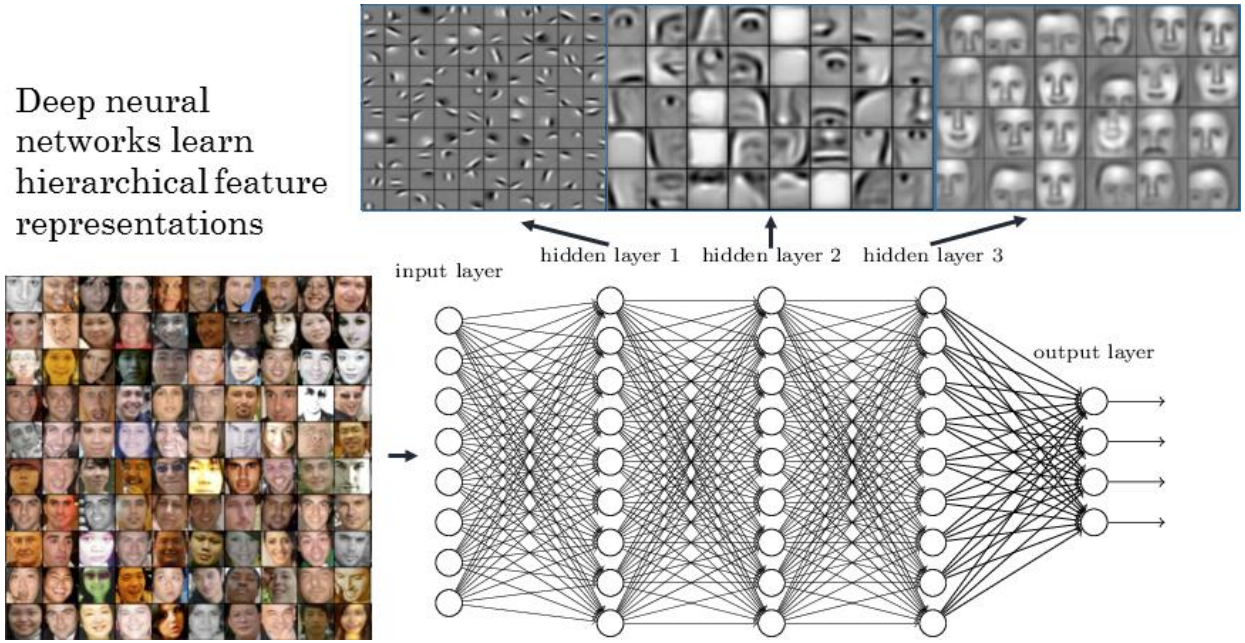
ROAD AHEAD: DEEP LEARNING



SVM = Support vector machine

ScaDS DRESDEN LEIPZIG

ROAD AHEAD: DEEP LEARNING (2)



- Deep Learning wins all competitions: Regognition, Segmentation Handwriting
- Exactly same neuronal networks as before, just BIGGER
  - Combination of three factors:
    - Big Data, Better algorithms, Parallel computing (GPU)

TOOLS AND EXPERTISE

- Emerging frameworks for DL – ‘the new big data’
- Good blog post for further references - https://deeplearning4j.org/compare-dl4j-torch7-pylearn
- Frameworks tend to interconnect
  - Spark+Hadoop, Spark+Cassandra, DL4J+Spark, Keras+Theano/TensorFlow, …

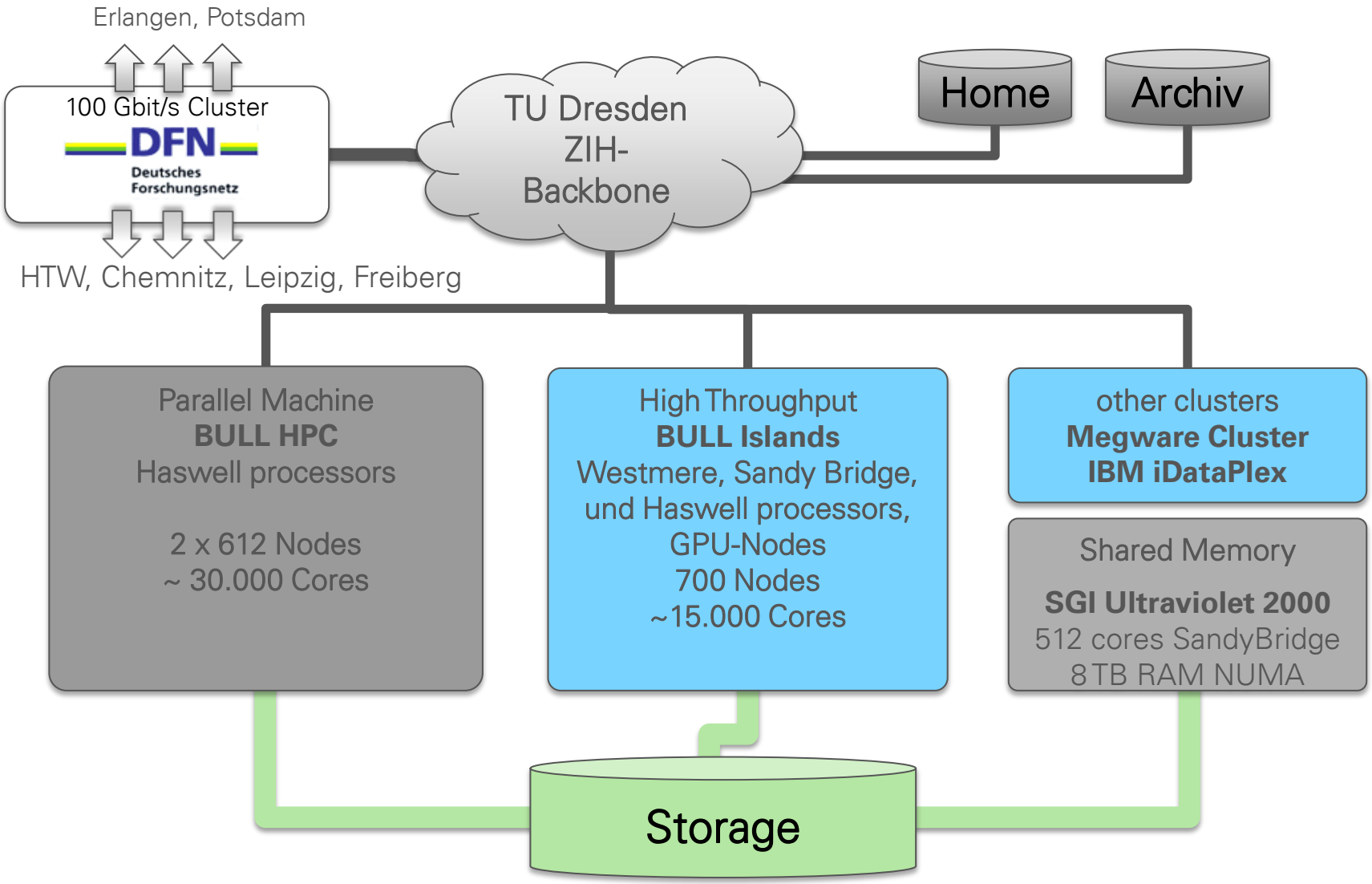
ScaDS DRESDEN LEIPZIG

RUN BIG SCALE ANALYSIS

Have you access to own cluster? Of what size?

ScaDS DRESDEN LEIPZIG

ZIH HPC AND BIG DATA INFRASTRUCTURE

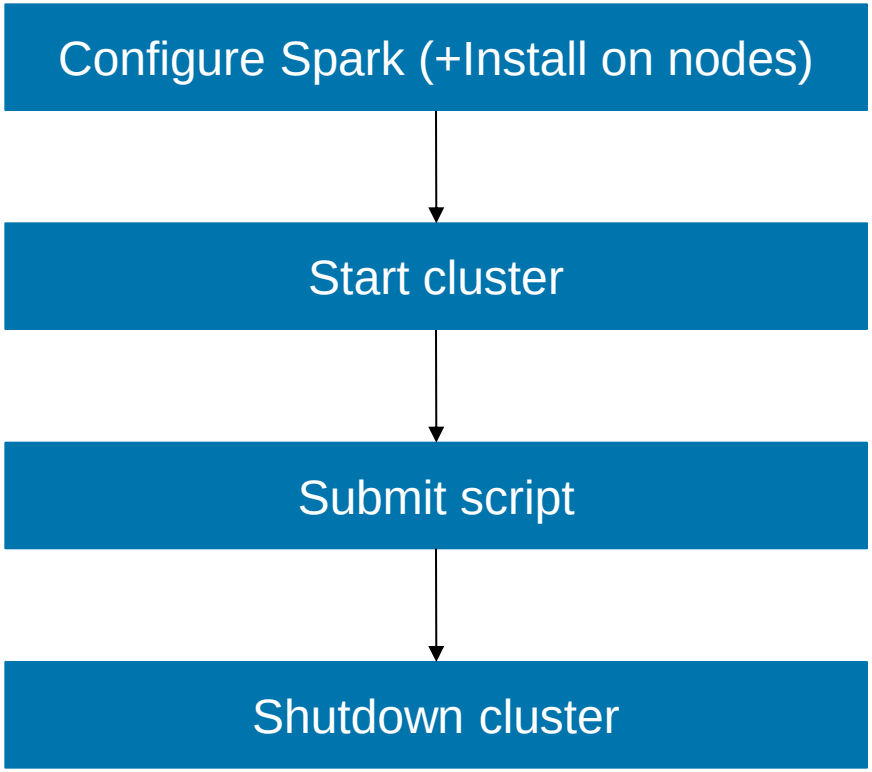


Comparison of Spark in normal and HPC environment

	Spark	Spark on HPC
Cluster started	Once	For each HPC job
Nodes selected by	User	Job scheduling system (SLURM)
Configuration requires changes	no	For each HPC job
Distributed filesystem	HDFS required	Lustre available

Problem, because nodes must be specified in configuration!

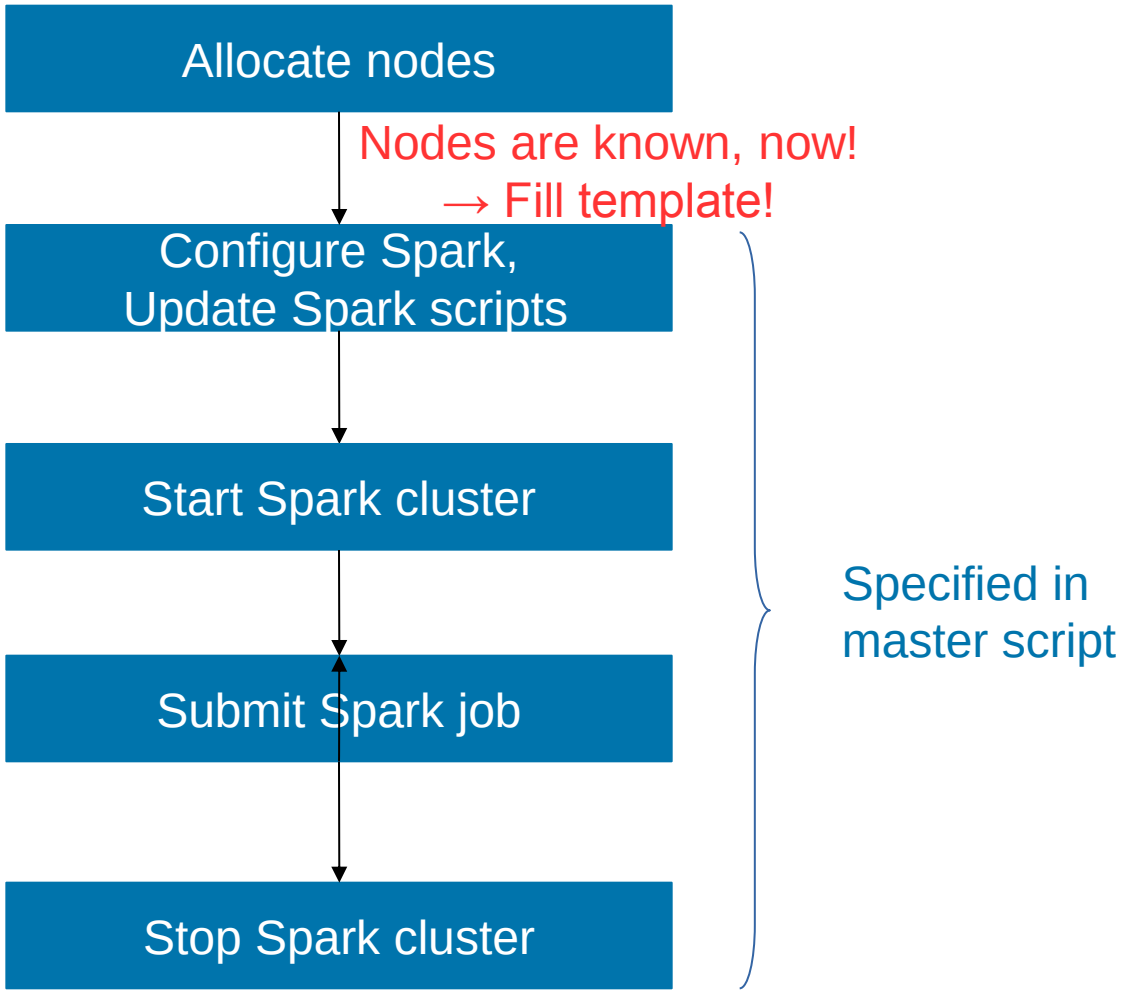
Problem, because Master‘s address needs to be known for job submission!


Configure Spark (+Install on nodes)
Start cluster
Submit script
Shutdown cluster

The master script – Example: JavaWordCount

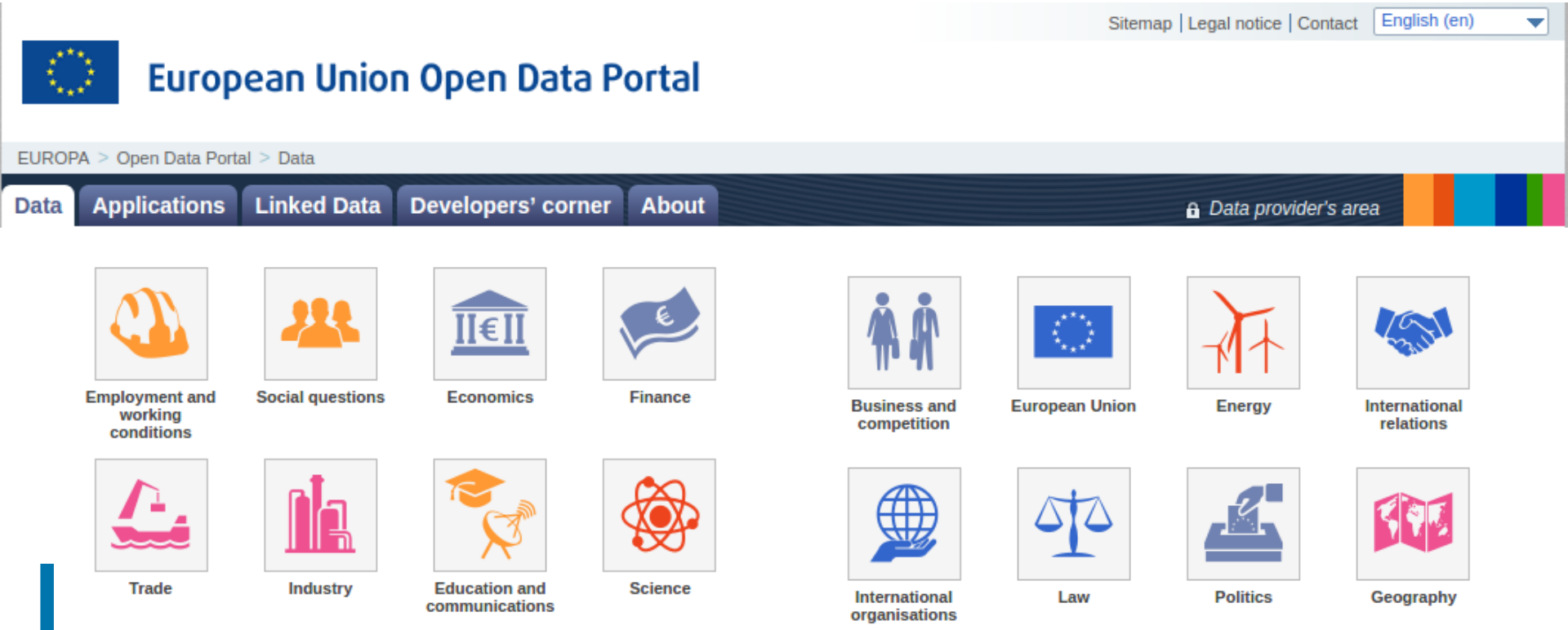
```
#!/bin/bash
MY_SCRIPT_PATH="$HOME/spark"
BD_FRAMEWORK_DIR=/projects/p_scads/bigdataframeworks

#configuration
$BD_FRAMEWORK_DIR/configure-scripts/framework-configure.sh spark $MY_SCRIPT_PATH/ ↵
spark-conf-framework

export SPARK_CONF_DIR=$HOME/cluster-conf-$SLURM_JOBID/spark

#start clusters
start-all.sh

#run spark job:
spark-submit --class org.apache.spark.examples.JavaWordCount $BD_FRAMEWORK_DIR/ ↵
spark/versions/spark-2.0.2/examples/jars/spark-examples_2.11-2.0.2.jar ↵
$BD_FRAMEWORK_DIR/examples/pg2701.txt

stop-all.sh
```

- Combine Spark workflow with HPC workflow
  - Job in a job: HPC job starts Spark and submits a job to it
- Avoid problems by using a configuration template
  - User can define what is needed
  - Place holders for nodes, directories
  - Place holder can be replaced with values once they are known
  - Modifies also scripts, e. g. spark-submit


Allocate nodes
Nodes are known, now!
→ Fill template!
Configure Spark,
Update Spark scripts
Start Spark cluster
Submit Spark job
Stop Spark cluster
Specified in
master script

OPEN DATA REPOSITORIES

- Lots of data out there to be analyzed
- Authorities tend to promote or even demand access to open data
- EU open data initiative: https://data.europa.eu



OPEN DATA REPOSITORIES

- Lots of data out there to be analyzed
- Authorities tend to promote or even demand access to open data
- U.S. Government's open data: http://data.gov

ScaDS
DRESDEN LEIPZIG

THANK YOU!!!