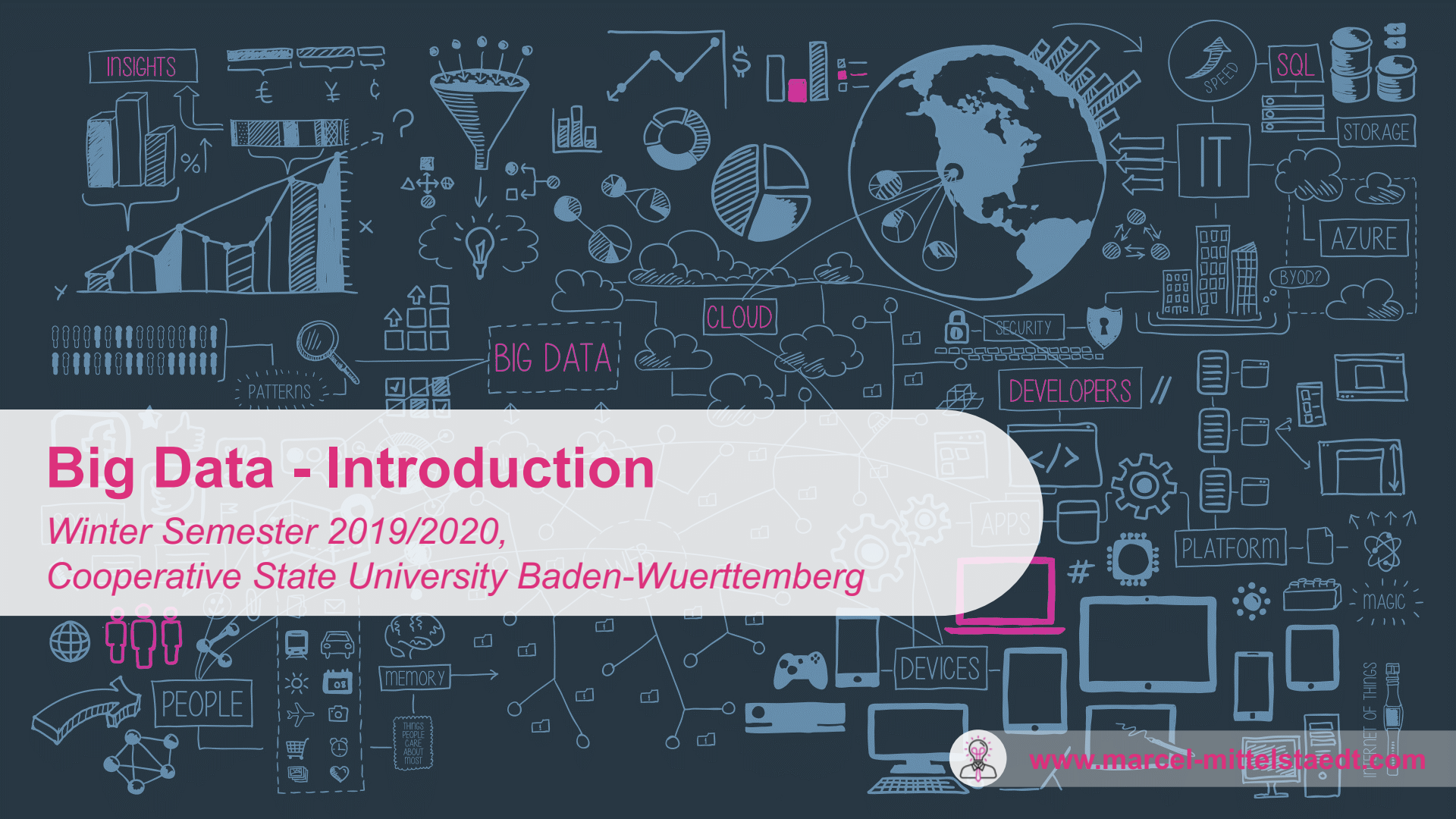


Big Data - Introduction

Winter Semester 2019/2020,
Cooperative State University Baden-Wuerttemberg

www.marcel-mittelstaedt.com

Agenda – 14.10.2019

01

About This Lecture

Audience, Schedule, Materials (Slides, Script, Exercises, Books), Prerequisites, Exam, Feedback, Questions

02

Introduction To Big Data

Distributed Computing, Big Data V's, Big Data and Consistency, Big Data in Business, Definition of Big Data, Big Data Science

03

HandsOn - Hadoop

Quick Introduction To The Hadoop Ecosystem, HDFS, Yarn and MapReduce

04

HandsOn – Hive and Hive(-QL)

Quick Introduction To Hive, HiveQL and external Tables

About This Lecture

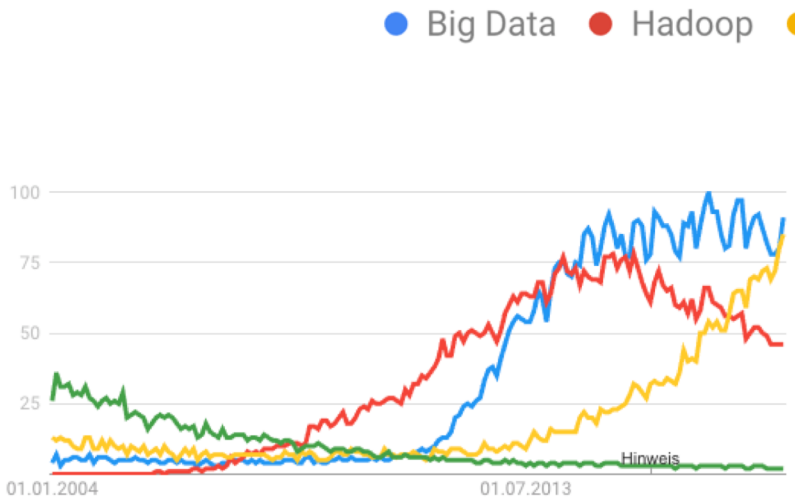
Scope, Schedule, Materials, Prerequisites, Exam, Feedback, Questions

About Big Data

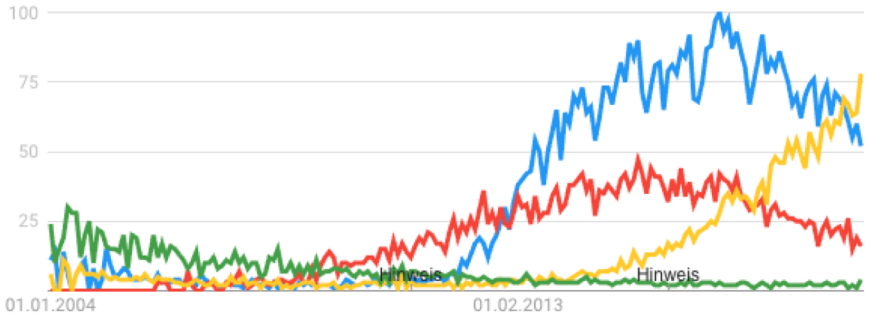
“Big Data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so every one claims they are doing it.”

— Dan Ariely, Professor of Psychology and Behavioral Economics, Duke University

About Big Data



Google Trends, 26.09.2018



Google Trends, 11.10.2019

About This Lecture

- Sorting the **buzzwords** (Big Data, NoSQL, Scalability, Replication, Sharding…)
- Understand the **business value**
- Develop a basic understanding of **Big Data** and **Distributed Systems**:
  - How and why do they work?
  - Fundamental approaches (for e.g. Scaling, Distributed Storage, Distributed Processing, …)
  - Data Models and Query Languages
  - Consistency guarantees of Distributed Big Data data-systems
  - Advantages and disadvantages of different software and approaches
  - Combination of different technologies

About This Lecture

- **HandsOn Exercises** on several software and frameworks (e.g. Hadoop, Map-Reduce, Hive, Spark etc.)
- **How does a productive environment look like?** (e.g. Partitioning, Replication, ETL Workflow-Management etc.)
- Quick **introduction** to **data science**

- **Not in this Lecture**:
  - Database fundamentals (*previous lectures*)
  - …

About Me

2018
- today

Co-Founder
DataWhizz, www.datawhizz.io

2018
- today

University Lecturer
Cooperative State University Baden-Wuerttemberg,
www.dhbw-stuttgart.de

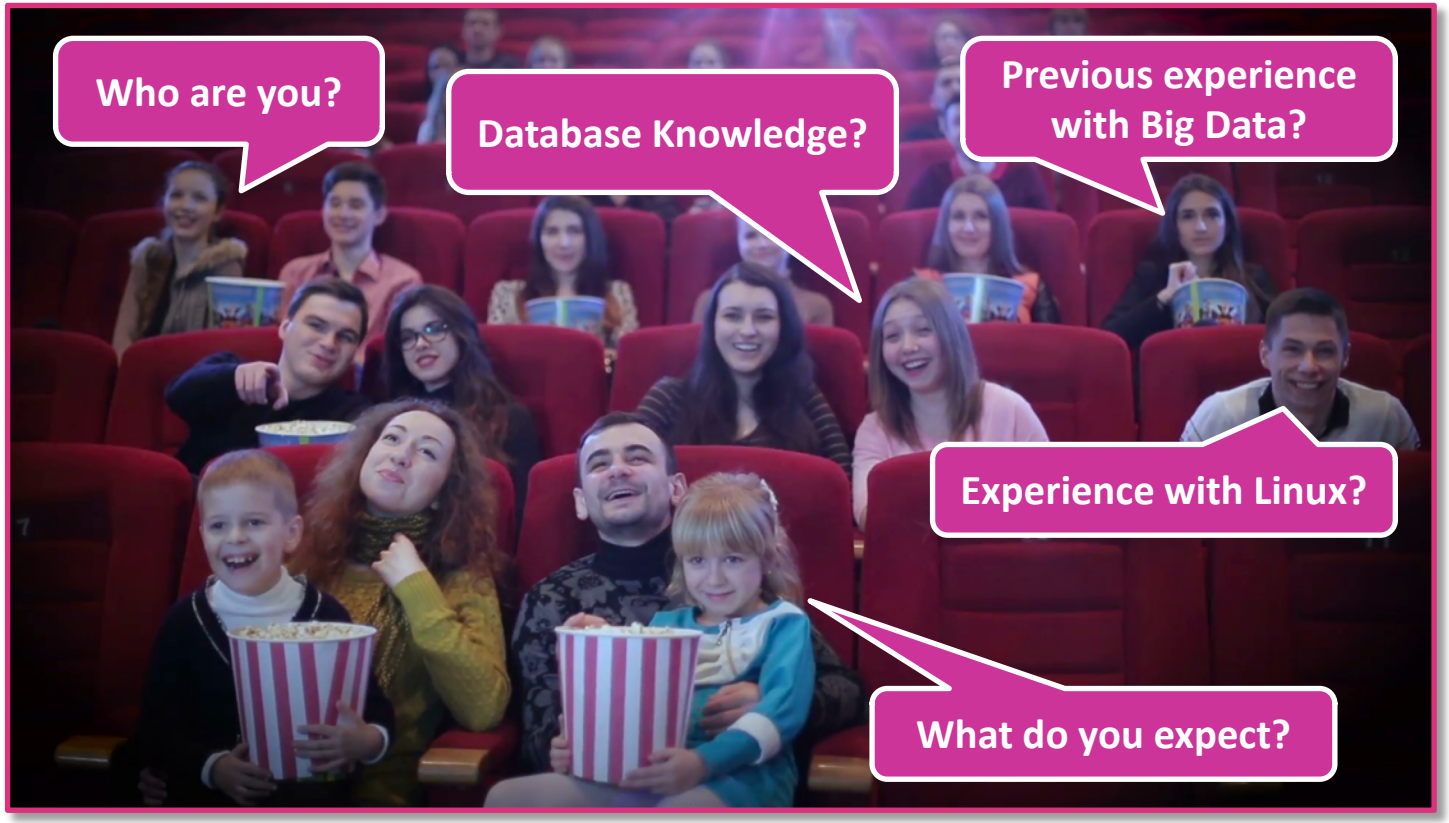
2014 - 2018
jan - nov

Head of Data Architecture and Development
ProSiebenSat.1 Media SE,
www.prosiebensat1.com

2009 - 2013
oct - dec

IT Architect
IBM Global Business Services,
ww.ibm.com

About You


Who are you?
Database Knowledge?
Previous experience with Big Data?
Experience with Linux?
What do you expect?

Schedule

	Lecture Topic	*HandsOn*
14.10.2019 13:15-18:00 Ro. 1.18	**About This Lecture, Introduction to Big Data, Setup Cloud Environment (Google Cloud)**	HandsOn Hadoop, Hive and Hive(-QL)
21.10.2019 13:15-18:00 Ro. 1.18	(Non-)Functional Requirements Of Distributed Data-Systems, Data Models and Access	Partitioning with HDFS/Hive, HiveServer2
28.10.2019 13:15-18:00 Ro. 1.18	Challenges Of Distributed Data Systems: Replication and Partitioning	Spark, Scala and PySpark/Jupyter
04.11.2019 13:15-18:00 Ro. 1.18	ETL Workflow and Automation, Batch Processing	Pentaho Data Integration & Airflow
11.11.2019 13:15-18:00 Ro. 1.18	Practical Exam	Work On Practical Exam
18.11.2019 13:15-18:00 Ro. 1.18	**Practical Exam Presentation**	

Schedule – Lecture Composition

14.10.2019	13:15-18:00	Ro. 1.18
21.10.2019	13:15-18:00	Ro. 1.18
28.10.2019	13:15-18:00	Ro. 1.18
04.11.2019	13:15-18:00	Ro. 1.18
11.11.2019	13:15-18:00	Ro. 1.18
18.11.2019	13:15-18:00	Ro. 1.18

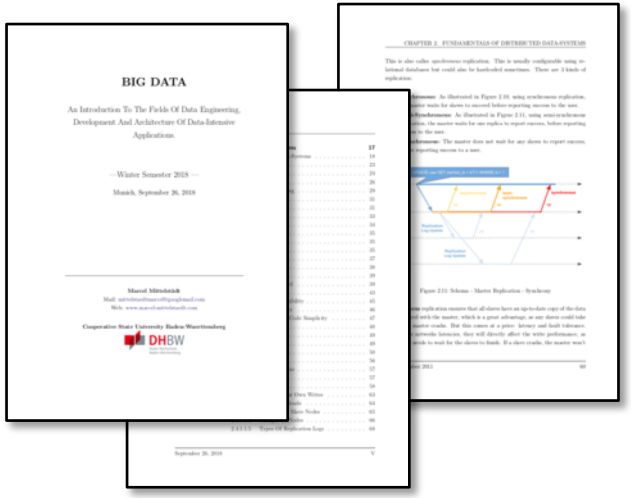
1. **Presentation, Review** and **Discussion** of **exercises** from previous lecture
2. **Lecture** of current topic
3. **HandsOn** to Software&Frameworks
4. **Exercise** execution

Materials

Script

https://github.com/marcelmittelstaedt/BigData/tree/master/script

- citations and references are only in the script



Slides

https://github.com/marcelmittelstaedt/BigData/tree/master/slides

- no citations and references (see script)

Materials

Exercises

https://github.com/marcelmittelstaedt/BigData/tree/master/exercises

Solutions

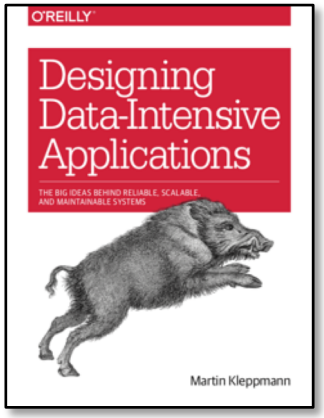
https://github.com/marcelmittelstaedt/BigData/tree/master/solutions

Materials

Docker Images

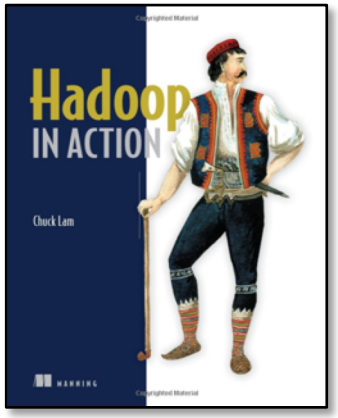
https://github.com/marcelmittelstaedt/BigData/tree/master/docker

Literature / Course Books



Designing Data-Intensive Applications
Martin Kleppmann

Amazon Link



Hadoop In Action
Chuck Lam

Amazon Link

Big Data Architect's Handbook
Syed Muhammad Fahad Akhtar

Amazon Link

PacktPub Link

Tech Ebooks from developer for developer

www.packtpub.com

Prerequisites

To efficiently participate:

- Background on and interest in **databases**
- **Basic knowledge** about and HandsOn **experience** with **Linux** (ideally Ubuntu)

For exam:

- **Attendance** on **lectures**
- **Doing** and **completion** of **exercises**

Feedback

Questions: anytime

This is a new and my second lecture ;)

Feedback: after lecture or via mail

If you find any mistakes or misspellings in the script or slides:

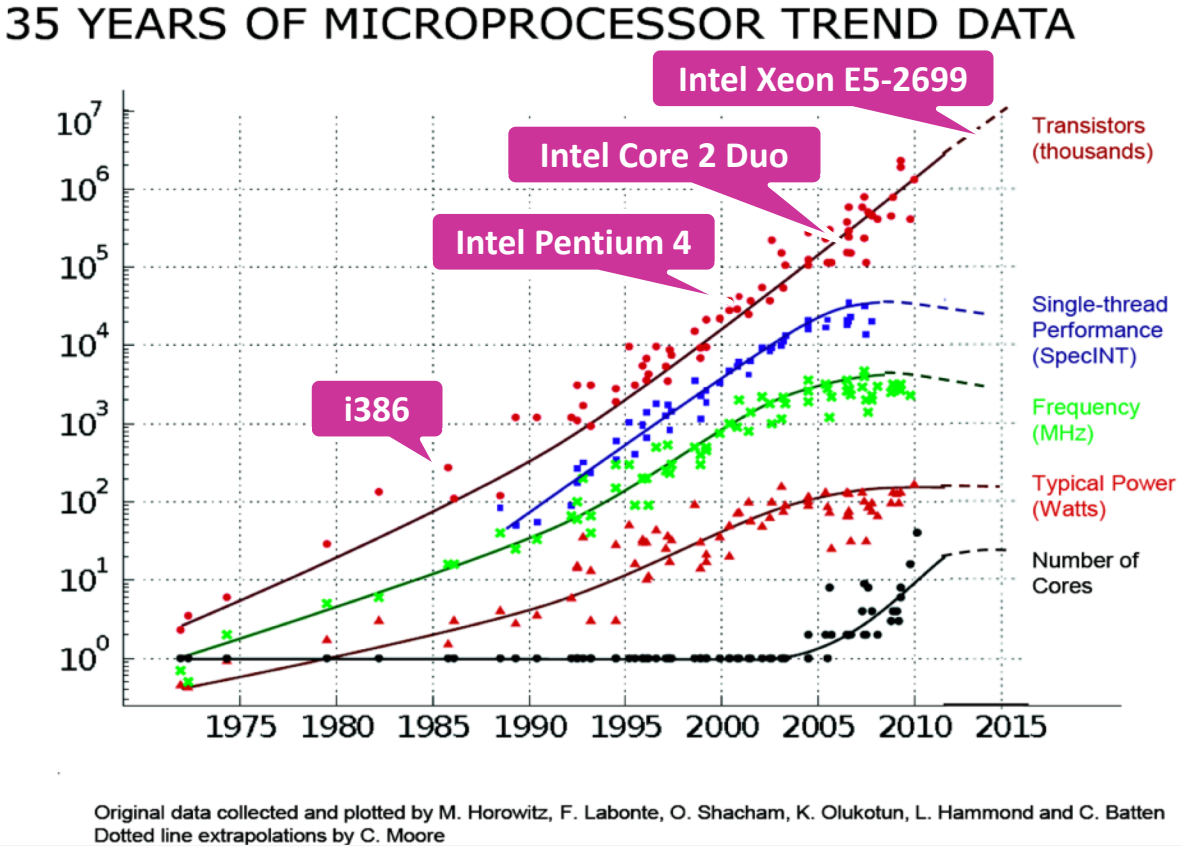
- Mail: contact@marcel-mittelstaedt.com
- commit a push request (git)

Questions

Introduction To Big Data

Distributed Computing, Big Data V's, Big Data and Consistency, Big Data in Business, Definition of Big Data, Big Data Science

Motivation - Paradigm Shift In Computing



- Frequency does not increase significantly anymore
- Watt does not increase anymore → ***power wall*** (temperature and power consumption)
- Transistor count still increases (***Moore's Law***)

Paradigm Shift:

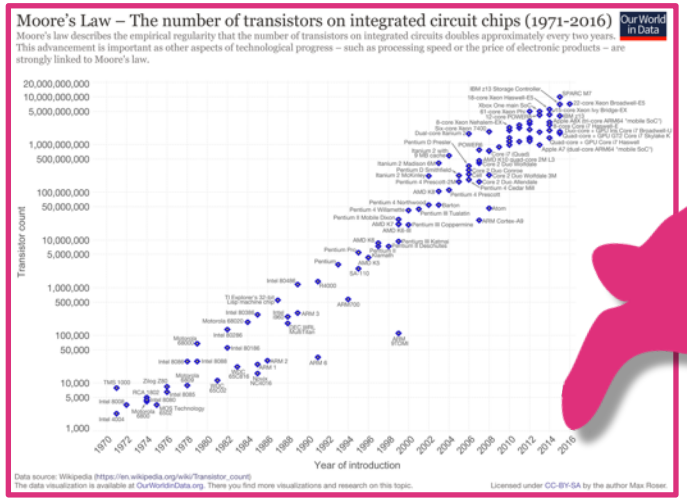
- **Back-In-Time**: → *scale vertical* – optimize code for a **single thread**
- **Today**: *scale horizontal* – run code **in parallel**

Motivation – **Definition Distributed Computing**

"***Distributed computing*** *is a field of computer science that studies distributed systems.*
A ***distributed system*** *is a system whose components are located on different networked computers, which then communicate and coordinate their tasks by passing messages to one other. The components interact with one other in order to achieve a common goal/task.*"

— https://en.wikipedia.org/wiki/Distributed_computing

Distributed Computing - **Bypassing Moore's Law**

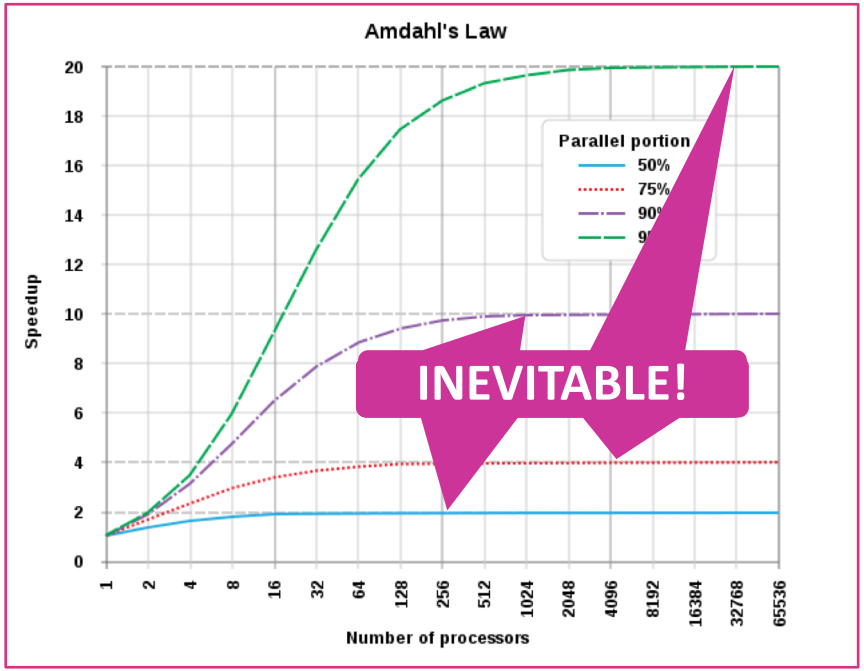


Moore's Law: *is the observation that the number of transistors in a dense integrated circuit doubles about every two years.*

Distributed systems: allow us to build data-systems with **any count of transistors**, just by adding further nodes to a cluster.

Distributed Computing – Amdahl's Law

Amdahl's Law: *is a formula to predict the theoretical speed-up when using multiple processors for distributed/parallel computing. The speedup is limited by the sequential part of the program.*



Formula:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

s *= #Cores/Degree of parallelization*

$$\lim_{s \to \infty} S_{\text{latency}}(s) = \frac{1}{1-p}$$

p *= percentage of part of the code, which benefits from distributed/parallel execution*

Example Program:

→ takes **20 hours** on a **single core**
→ **95%** of the code can be **executed in parallel**
→ **8 cores** available

p = 0,95 *s = 8*

$S_{latency}(s) = 1 / ((1 - 0,95) + (0,95 / 8)) = \mathbf{5,9}$

$\lim S_{latency}(s) = 1 / (1 - 0,95) = \mathbf{20}$

→ *execution time can **never** be **less than one hour***

Distributed Technologies – **Small Scale**

Size: 0 - 10 Nodes
Hardware: “Commodity Hardware”

Space: < ~500 TB
Cores: < ~100 CPUs
Costs: cheap

Low-Cost
e.g. Raspberry Pi

Costs Per Node:
~30€

Desktop PC’s
e.g. anyone

Costs Per Node:
~1.000-3.000€

Distributed Technologies – **Medium Scale**

Size: 10 - 100 Nodes
Hardware: “Commodity Hardware” in terms of racks, usually with tuned parts (e.g. SAS instead of SATA drives, 8GBit Rack Uplink, 64-512GB RAM instead of 16-32GB etc.)

Space: < ~7,2 PB
Cores: < ~1.600-3.200 CPUs
Costs: inexpensive (compared to ***scale-up***)

Racks
e.g. HP ProLiant DL 380 Gen 10, 2RU

Costs Per Node:
~8.000-12.000€

Desktop PC’s
e.g. anyone

Costs Per Node:
~1.000-3.000€

Distributed Technologies – **Large Scale**

Size: 100 - X Nodes (e.g. 4.000 Nodes within Yahoo Hadoop Cluster in 2008)

Hardware: "Commodity Hardware" in terms of racks, usually with tuned parts (e.g. SAS instead of SATA drives, 8GBit Rack Uplink, 64-512GB RAM instead of 16-32GB etc.)

Space: > ~10 PB
Cores: > ~3.200 CPUs
Costs: still rather cheap than pricey (compared to ***scale-up***)

Yahooo Hadoop Cluster, 4.000 nodes, 2008

Racks
40 Nodes / Rack
Rack Uplink 8Gbit

Costs Per Node:
~8.000-15.000€

Distributed Technologies – **Supercomputer**

Name	Standort	TeraFLOPS	Konfiguration	Energiebedarf	Zweck
Summit	Oak Ridge National Laboratory (Tennessee, USA)	122.300,00	9.216 POWER9 CPUs (22 Kerne, 3,1 GHz), 27,648 Nvidia Tesla V100 GPUs	15.000 kW	Physikalische Berechnungen
Sunway TaihuLight	National Supercomputing Center, Wuxi, Jiangsu	93.014,60	40.960 Sunway SW26010 (260 Kerne, 1,45 GHz), 1,31 PB RAM, 40 Serverschränke mit jeweils 4 × 256 Nodes, insgesamt 10.649.600 Kerne	15.370 kW	Wissenschaftliche und kommerzielle Anwendungen
Sierra[6]	Lawrence Livermore National Laboratory (Kalifornien, USA)	71.600,00	IBM Power9 (22 Kerne, 3,1 GHz) 1,5 PB RAM	~12.000 kW	physikalische Berechnungen (z. B. Simulation von Kernwaffentests)
Tianhe-2[7]	National University for Defense Technology, Changsha, China finaler Standort: National Supercomputer Center (Guangzhou, China)	33.862,70 aufgerüstet auf 61.400,00	32.000 Intel Xeon E5-2692 CPUs (Ivy Bridge, 12 Kerne, 2,2 GHz) + 48.000 Intel Xeon Phi 31S1P Co-Prozessoren (57 Kerne, 1,1 GHz), 1,4 PB RAM	17.808 kW	Chemische und physikalische Berechnungen (z. B. Untersuchungen von Erdöl und Flugzeugentwicklung)
Piz Daint	Swiss National Supercomputing Centre (CSCS) (Schweiz)	19.590,00	Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc. (361.760 Kerne)	2.272 kW	wissenschaftliche und kommerzielle Anwendungen
Titan	Oak Ridge National Laboratory (Tennessee, USA)	17.590,00	Cray XK7, 18.688 AMD Opteron 6274 CPUs (16 Kerne, 2,20 GHz) + 18.688 Nvidia Tesla K20 GPGPUs, 693,5 TB RAM	8.209 kW	Physikalische Berechnungen
Sequoia[8]	Lawrence Livermore National Laboratory (Kalifornien, USA)	17.173,20	IBM BlueGene/Q, 98.304 Power BQC-Prozessoren (16 Kerne, 1,60 GHz), 1,6 PB RAM	7.890 kW	Simulation von Kernwaffentests
	Advanced Institute for		88.128 SPARC64-VIII 8-		

https://de.wikipedia.org/wiki/Supercomputer

"Summit",
Oak Ridge National Laboratory, Tennessee, USA

Cores: 202.752
TeraFLOPS: 122.300,00
Energy: 15.000 kW

Summit Supercomputer, Oak Ridge National Laboratory

Distributed Technologies – **Large Cloud Cluster**

Size: 200 Nodes

Cores: 12.800 vCPUs / 6.400 CPUs

** **DISCLAIMER:** Completely ignoring storage, network, virtualization and sharing of CPUs etc.*

AWS:

Instance: *m4.16xlarge*
vCPUs *per Instance:* 64
RAM *per Instance:* 256
Costs per hour *per Instance:* *3,20 USD*

Costs for ***Large Cluster*** *per hour:*
*200 * 3,20 USD =* ***640,00 USD***

https://aws.amazon.com/de/ec2/pricing/on-demand/

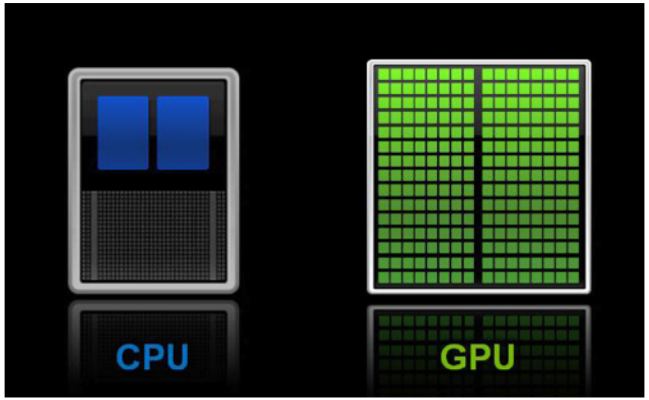
Google Cloud:

Instance: *n1.standard-64*
vCPUs *per Instance:* 64
RAM *per Instance:* 240
Costs per hour *per Instance:* *3,04 USD*

Costs for ***Large Cluster*** *per hour:*
*200 * 3,04 USD =* ***608,00 USD***

https://cloud.google.com/compute/pricing

Distributed Technologies – **GPU Computing**

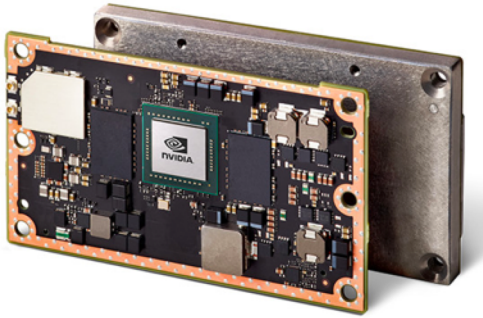


- advanced GPUs **initially developed for** (3D) **gaming**

→ **now also used for** (distributable) **computational workloads** (e.g. Big Data, Machine Learning)

Cores:	**2-18** *(e.g. 12 Cores for Intel Xeon E5-2690)*	**1.000-4.000** *(e.g. 3.584 CUDA for GTX 1080 Ti*
Freq.:	**2-4 Ghz** *(e.g. 2,6-3,5 Ghz for Intel Xeon E5-2690)*	**1.000-2.000 MHz** *(e.g. 1.582MHz for GTX 1080 Ti)*

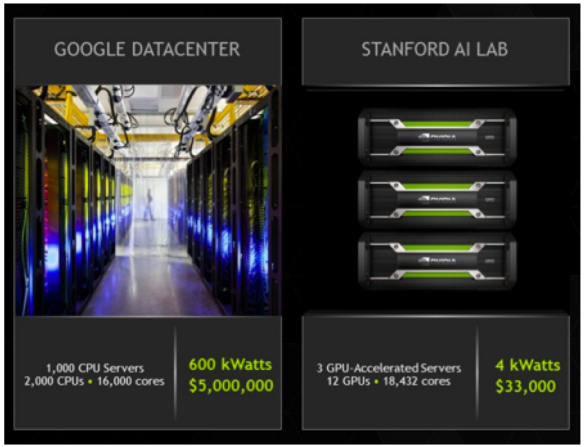
Nvidia Jetson TX2
Cores: *256 (CUDA)*
RAM *per Instance:* *8*



https://www.nvidia.com/de-de/autonomous-machines/embedded-systems-dev-kits-modules/

Distributed Technologies – GPU Cluster

On-Premise:



https://cs.stanford.edu/csdcf/sail-compute-cluster

https://www.slideshare.net/papisdotio/introduction-to-multi-gpu-deep-learning-with-digits-2-mike-wang

Cloud:

AWS:

Amazon EC2 Elastic GPUs – Preise

Mit Amazon EC2 Elastic GPUs zahlen Sie ausschließlich für die tatsächliche Nutzung. Die Preise für Elastic GPUs sind unten aufgeführt.

USA Ost (Ohio) und USA Ost (Nord Virginia)

Größe der Elastic GPU	GPU-Arbeitsspeicher	Preis
eg1.medium	1 GiB	0,050 USD/Stunde
eg1.large	2 GiB	0,100 USD/Stunde
eg1.xlarge	4 GiB	0,200 USD/Stunde
eg1.2xlarge	8 GiB	0,400 USD/Stunde

https://aws.amazon.com/de/ec2/elastic-gpus/pricing/

Google Cloud:

GPU-Modelle für Compute Engine

GPU-Modell	GPUs	GPU-Speicher	Verfügbare vCPUs	Verfügbarer Speicher
NVIDIA® Tesla® V100	1 GPU	16 GB HBM2	1–12 vCPUs	1–78 GB
	8 GPUs	128 GB HBM2	1–96 vCPUs	1–624 GB
NVIDIA® Tesla® P100	1 GPU	16 GB HBM2	1–16 vCPUs	1–104 GB
	2 GPUs	32 GB HBM2	1–32 vCPUs	1–208 GB
	4 GPUs	64 GB HBM2	1–64 vCPUs (us-east1-c, europe-west1-d, europe-west1-b)	1–208 GB (us-east1-c, europe-west1-d, europe-west1-b)
			1–96 vCPUs	1–624 GB

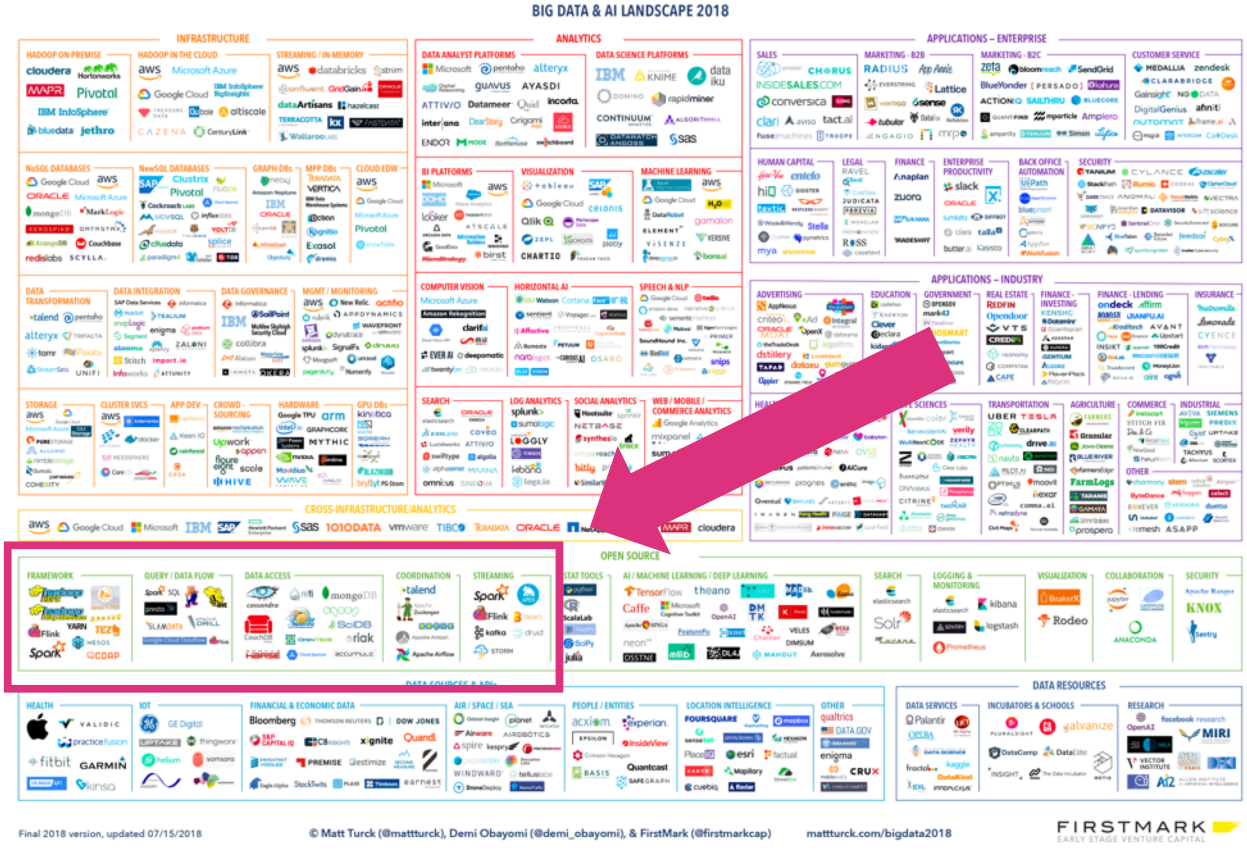
https://cloud.google.com/compute/docs/gpus/

Distributed Technologies – **Big Data**

Distributed Processing (Computing)

Distributed Data (Storaging)

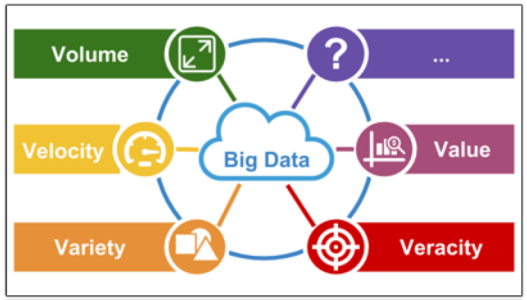
Distributed Technologies – Big Data Landscape



Big Data V's

- Introduced by **Gartner** in **2011**
- **Key-Characteristics** of Big Data
- **More V's** have established over time

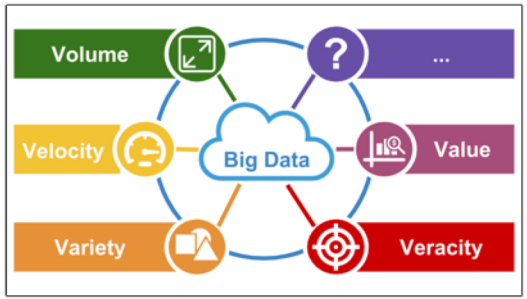
Big Data V's - **Volume**



- **Wikipedia:** in 2014 the english part of Wikipedia and Wikipedia Commons had a size of around **23TB**
- **Spotify:** in 2013 data generated by users of spotify was about **1,5TB/day** → **534TB/year**
- **Google:** search index size currently about **100PB**

- **Rule Of Thumb:** data that can easily be handled (e.g. processed, saved or queried) within a traditional database (e.g. PostgreSQL, Informix, DB2, Oracle etc.) and belonging datawarehouse, like a few gigabytes per week or month, is not Big Data.

Big Data V's - **Velocity**

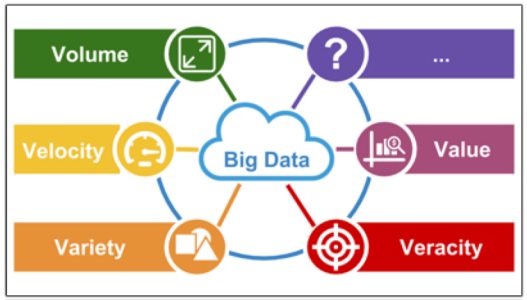


- **Facebook:** in 2012 **every 60 seconds:**
  510.000 comments,
  293.000 status updates and
  136.000 photos
  had been posted on average.

- **Twitter:** in 2013 usually **500 million tweets** were posted each day on Twitter, which means **5.700 Tweets per second** on average.

- **Google:** According to Internet Live Stats, Google is handling around **6 billion search requests** per day right now, which means **70.000 per second**.

Big Data V's - **Variety**



Structured Data:

- **plain** data structures with **fixed** attributes/columns
- **specific data types** and **defined domain**
- related but different kinds of data records can easily be linked (primary/foreign keys)
- can easily (almost directly) be stored into a traditio nal relational database

Semi-Structured Data:

- **do not** (easily) **fit** into the **constraints** of the **relational data model**
- possible (with additional effort) to transfer into relational world, but violating relational data model constraints
- no need to worry about *object-relational impedance mismatch*
- vulnerable to *garbage in – garbage out*

Big Data V's - **Variety**

Semi-Structured Data:

- **highly-nested** and **hierarchical** data structures
- Examples:
  - **JSON** documents (e.g. provided by the Facebook Graph API or the Twitter API)
  - **XML** documents (e.g. provided by the Google Maps Geocoding API or the OpenWeatherMap Weather API)
  - **HTML** documents, for instance if your developing a web crawler (e.g using Scrapy for data gathering or Selenium for testing purposes)

Big Data V's - **Variety**

Unstructured Data:

- everything that is **neither structured** or **semi-structured**
- Examples:
  - **Natural Language**:
    - Social Media Posts (e.g. Facebook and Twitter),
    - Ratings (e.g. on Amazon, IMDB or other platforms) or
    - plain text (e.g. Wikipedia articles, product information on Amazon or any e-commerce shop)

    → you may want to analyze by *natural language processing* with the purpose of calculating a **sentiment** (*What do customers think about your product?*) or the meaning and truthfulness of a rating (*Is the revi ew positive or negative*? Is it fake?).

Big Data V's - **Variety**

Unstructured Data:

- Examples:
  - **Geographical Data**:
    - GPS,
    - Radar,
    - sonar data or images

    → with the purpose of analyzing meteorological, vecicular or seismic behaviour, e.g. for the purpose of predicting the future.

  - **Photos and Videos**:

    → for instance of traffic and surveillance cameras to analyze and optimize traffic flow or for the sake of security.

Big Data V's - **Variety**

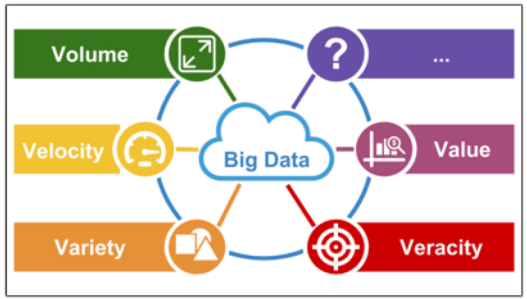
Unstructured Data:

- Examples:
  - **Scientific Data**:
    - physic measuring and sensor data,
    - seismic imagery or
    - athmospheric data

    → to analyze and optimize a physics experiment.

    For instance CERN makes use of Hadoop for unstructured sensor data, to unlock insights from machine data.

Big Data V's - **Veracity**



- ***veracity*** is about the **trustworthiness** and **accuracy** of a certain dataset, as data usually ivolves some uncertainty and ambiguities
- not just about the quality of the data itself, but also tasks of **quality assurance** and data cleansing, like removing **bias**, **superfluous**, **redundant** or just **duplicate records** or attributes.

- **trustworthiness** is highly vulnerable to the **volatility of the data** (which you usually cannot control), as a lot of data sources and related records frequently change in their lifetime

→ an account balance quickly changes over time (due to ongoing transactions)

→ people which have liked a topic, post or site on a social media platform may dislike it the next day

Big Data V's - **Veracity**

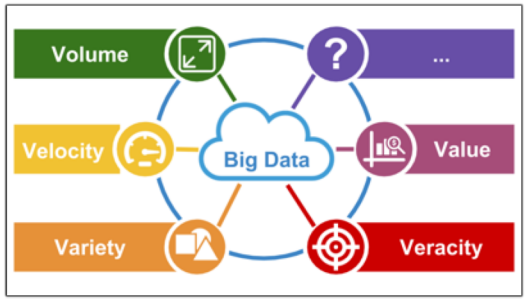
- even if you have a fully-fletched environment which **ensures data quality in its own limits**, you cannot be sure the data you are processing is 100% accurate, as their could already be a **mistake in the source system**

- Big Data data-system should always try to **achieve** the **best data quality possible**, even it is seen a lot: *Garbage In - Garbage Out* is not acceptable.

→ Garbage Data + Perfect ETL or Data Science Model = **Garbage Results**

→ Perfect Data + Messy ETL or Data Science Model = **Garbage Results**

Big Data V's - **Value**



- *volume*, *velocity*, *variety* and *veracity* - all of them are meaningless if you dont derive **business value** from your data, but they are also significant **enabler** and **success factors** for the ***value*** **of your data**
- **value** can probably be found in any kind of data, but the challenge is to **pick the right data** (business case) and **use it the right way**

- Volume and **Value**: The more data you collect, the more insights you can probably gather and the more informed your insights and decisions may be.

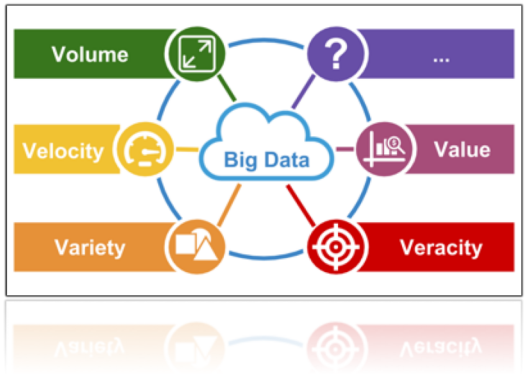
- Velocity and **Value:** The faster data is collected and available for analysis processes, the faster it can be used for decision-making processes.

Big Data V's - **Value**

- Variety and **Value**: The more data sources you connect to your data-system, the more insights you can potentially create.
  You will no longer rely an a single data source for a certain business object (like a client), which overall increases data quality and depth. Enabling you, for instance to build customer journeys, calculate a CLV, and in this way improve engagement and retention. At the end the value of your data increases.

- Veracity and **Value:** The more trustworthy and accurate your data is, more business critical desicisions and applications will rely on it and in this way the revenue of your company will probably increase as well as the value of your data.

Big Data V's – Other V's



- **Volatility:** How long data is valid and how long it should be stored. Retention requirements? Data Protection requirements?

- **Variability:** Format, schema or semantic changes.

- **Validity:** Like veracity, is the data correct and accurate for the intended use? Clearly valid data is key to making the right decisions.

- **Venue:** Different location require different access and work methods.

- **…**

Big Data Challenges – **ACID**

- Gives following **4 guarantees**:

- **Atomic:** all operations in a transaction succeed or every operation is rolled back

- **Consistent:** Before and after a transaction, the data-system is structurally sound.

- **Isolated:** Transactions do not contend with one another. Contentious access to data is moderated by the database so that transactions appear to run sequentially.

- **Durable:** The results of applying a transaction are permanent, even in the presence of failures.

→ **Ensures safe and reliable data storage and processing**
→ **Requires a lot of additonal workload like locks, moderated data access, failover protection**

Big Data Challenges – **BASE**

- Not (fully) ACID/ soft version of ACID

Basic Avaiability: the data-system should work an be available most of the time.

→ availability less than 100%

Soft-State: data stores do not have to be write-consistent, nor do different replicas have to be mutually consistent all the time.

→ Stored data may be inconsistent, but data store can derive consistent states.

Eventual Consistency: Stores exhibit consistency at some point later in time. For instance at at read time.

→ usually consistent within seconds/milliseconds

→ eventual consistency does not mean no-consistency

Big Data Challenges – **ACID** vs **BASE**

ACID	BASE
Strong consistency	Weak consistency
Isolation	Availability first/Last write wins
Focus on „commit“	„Best effort“
Trnsactions	Programmer managed
Nested transaction	Approximate answer
Conservative (pessimistic)	Aggressive (optimistic)
Robust database	Simple database
Simple application code	Complex application code

Big Data Challenges – **CAP** (Eric Brewer)

- **Availability**:
  Every request receives a (*non-error*) **response** - without guarantee that it contains the most recent write

  → e.g. **high load**,
  → server **failures** or
  → query **congestion**
  may **deny service availability**

Big Data Challenges - CAP

- **Consistency**:
  Every read/all clients receive the **most recent write** or an error.

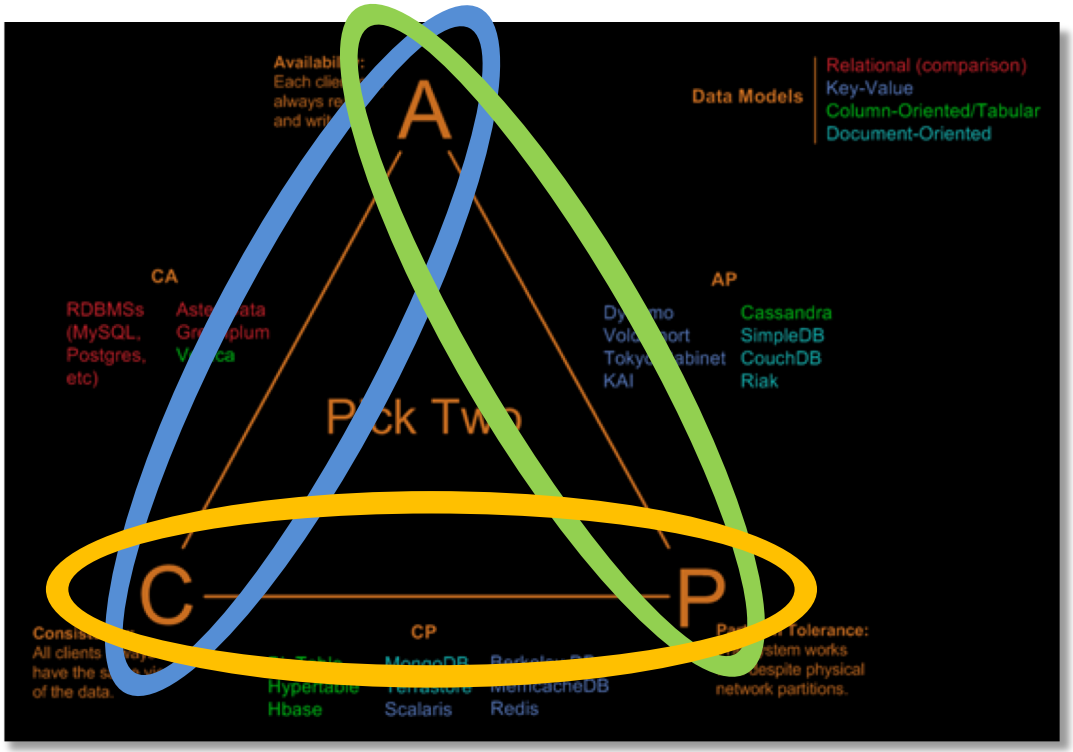
- e.g. ***ACID consistency*** (consistent transactions)
- but on a **cluster-wide** **not node-wi de** consistency

Big Data Challenges - CAP

- **Partition Tolerance**:
  The **system continues to operate** despite an **arbitrary number of messages being dropped** (or delayed) by the network in between nodes.

→ only total network failure might cause the data-system to respond incorrectly.

Big Data Challenges – **CAP Examples**



- Usually achieved by **not sharding**.
- If **server** or **network failures** occure, return **whatever is possible**.
- If **server** or **network failures** occure, try to recover and **deny availability** until state is consistent again.
- **Traditional databases** usually achieve all 3 as they are not distributed.

Big Data Roles - **Data Engineer / Architect**

Study Direction: Computer Science (databases, software engineering, distributed systems and processing, real-time processing)

(Programming) Languages: Java, Scala, Python, Bash, if necessary ETL Tools (e.g. Pentaho Data Integration, Informatica, DataStage, Talend)

Key-Skills:

- data modeling
- ETL and dataflow architecture and design
- data pipeline/workflow design and management
- stream- and batch-processing
- understand the concept, performance factors and limitations of distributed data-systems

Big Data Roles - **Data Engineer / Architect**

Key-Skills:

– alignment with larger organizations goals and strategy as well as corporate guidelines (e.g. data protection, information security, lice nse and toolset policies)

Tasks:

– develop ETL jobs, data- and workflows
– develop and define architecture of a data-system,
– integration of data-sources
– alignment with larger organizations goals and strategy as well as corporate guidelines (e.g. data protection, information security, license and toolset policies)

Big Data Roles - **Data Ops**

Study Direction: Computer Science (databases, software engineering, distributed systems and processing, real-time processing)

(Programming) Languages: Python, Bash

Key-Skills:

– strong communication and collaboration skills
– broad knowledge of Ops Tools, best practices and automation
– frequently refreshs his knowledge as Ops tools crop up continuosly
– Continuos Integration and Deployment Automation (e.g. Jenkins, Gitlab CI),

Big Data Roles - **Data Ops**

Key-Skills:

- Infrastructure Automation (e.g. Foreman, Puppet, Ansible),
- Containerization (e.g. Docker, LXD etc.),
- Orchestration (e.g. Kubernetes, Mesos, Swarm),
- Cloud (e.g. AWS, Google Cloud Platform, Azure, OpenStack)
- Source Control (e.g. Git, Bitbucket, SVN)

Tasks:

- installs, configures, upgrades, scales and operates infra-structure and applications
- monitoring and management of infrastructure and applications
- provides and manages tools for continues delivery
- incident and problem management
- interface to developers (requirements for infrastructure, support developers needs and productive application operation)

Big Data Roles - **Data Scientist**

Study Direction: Statistics, Data Science, Mathematics, Computer Science (Java, Python, R, SQL, Machine Learning)

(Programming) Languages: Python, R, Matlab, Bash, Java, Scala

Key-Skills:

- strong communication and collaboration skills
- business domain expertise
- deep understanding of the data he is working with
- broad experience in using programming languages and tools for statistical purposes (Python, R, SQL, MapReduce, Spark, SparkML, Pandas, dataframes)

Big Data Roles - **Data Scientist**

Key-Skills:

– knowledge of a variety of machine learning techniques (e.g. clustering, regression, decision trees, neural networks, properties of distr ibution)
– strong skills in data visualization (e.g. seaborn, Matplotlib, ggplot)

Tasks:

– identify and evaluate data-analytics problems that offer the greatest opportunities and/or business value
– determining the right data sets and variables
– cleansing and validation of data to ensure completeness, accuracy and uniformity
– developing and applying models and algorithms on data
– analyze data to identify patterns, risks and trends
– commmunication with and visualization for (business) stakeholder

Big Data Roles - **Salaries**

Big Data Ops

€€€ ?

Big Data Engineer

€€€ ?

Data Scientist

€€€ ?

Big Data Roles - **Salaries**

Big Data Ops

→ **45.000€ - 70.000€**

Big Data Engineer

→ **50.000€ - 80.000€**

Data Scientist

→ **50.000€ - 100.000€**

https://www.glassdoor.de/Gehälter

Big Data Definition

*"**Big Data** is a term for **datasets** that are **too large**, **too fast** and/or **too complex** for traditional databases and ETL tools to handle.*

*The term **Big Data** also refers to all related **algorithms**, **software**, **infrastructure** and **tools** used to be able to cope with such datasets."*

Break

TIME FOR A BREAK

Google Cloud Setup

Setup Google Cloud SDK, Setup SSH Access,
Setup Firewall, Spawn Instances

Setup gcloud SDK

1. Download and Install SDK:

Mac OSX:

```
# Download
wget https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-245.0.0-darwin-x86_64.tar.gz

# Untar
tar -xvzf google-cloud-sdk-245.0.0-darwin-x86_64.tar.gz -C ~/

# Install
~/google-cloud-sdk/install.sh
```

Windows:

```
https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe
```


Setup gcloud SDK

1. Download and Install SDK:

Linux:

```
# Add the Cloud SDK distribution URI as a package source
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] http://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list

# Import the Google Cloud Platform public key
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add -

# Update the package list and install the Cloud SDK
sudo apt-get update && sudo apt-get install google-cloud-sdk
```


Setup gcloud SDK

2. Initialize SDK:

```
gcloud init
```

- Authenticate using your Google Cloud Account
- When asked for which cloud project to use: choose the one associated with the lecture Google Cloud Credits

Setup SSH Access

1. If you do not already have SSH keys, create them:

```
ssh-keygen -t rsa -b 4096 -C "hans.wurst@lehre.dhbw-stuttgart.de"
```

2. Create **`gcloud_keys.txt`** containing your public key (id_rsa.pub) in this format:

```
hans.wurst:ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAl5EcXPTmrT2efonOKmr1yrNZtyEKmnWIOe4H9xbJp02Kus
NRQTtdcWTpzDNKKCVxIYAfhaQ01dJaW8CZNJw8JaRnK8eLEHFg/+T86u2ep6BYLNDMjicnN8k3qZoCk4eAomCtqNo9O1/
4TgFAo7a2Nk1LmoATPPUORoagbVCwZpU4e3emnS1yUkwkcjbMw24Fzwv4mkiebSGe58AsberhYtCaoGKRpjCtEMtCG0zb
OxfC2Em4wYh9FQnLvKkdbEU1raPZ3aKIOpeDKjYLmp9r8D2akBvA3lSgNRS/0uh7YhLw4kmiCscTKlSEtMG/TNZEP2smW
fasG03LvMpC3vZ7w== hans.wurst@lehre.dhbw-stuttgart.de
```

3. Upload Key to Gcloud Metadata:

```
gcloud compute project-info add-metadata --metadata-from-file ssh-keys=gcloud_keys.txt
```


Setup gcloud Firewall

1. Update Firewall Settings to be able to access services within gcloud:

```
gcloud compute firewall-rules create remoteaccess \
            --action=ALLOW \
            --rules tcp:80,tcp:8088,tcp:9870,tcp:10000,tcp:4040,tcp:8042,tcp:8888,tcp:9864 \
            --description remote_access \
            --direction INGRESS \
            --network default \
            --source-ranges XXX.XXX.XXX.XXX
```

- source_ranges needs to be your public IP
- this limits accessibility of your future VM instances to you
- to get your current public IP:

```
curl ifconfig.me
80.124.10.234
```

or: https://www.wieistmeineip.de/

Spawn VM instance

1. Spawn VM Instance with 15GB RAM and 4 Cores:

```
gcloud compute --project=[your-project-id] instances create big-data \
          --zone=europe-west3-c \
          --machine-type=n1-standard-4 \
          --subnet=default --network-tier=PREMIUM \
          --maintenance-policy=MIGRATE \
          --image=ubuntu-1804-bionic-v20190918 \
          --image-project=ubuntu-os-cloud \
          --boot-disk-size=30GB \
          --boot-disk-type=pd-standard \
          --boot-disk-device-name=big-data
```

2. List all instances to get public IP of gcloud VM instance:

```
gcloud compute instances list
NAME      ZONE            MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP   STATUS
big-data  europe-west3-c  n1-standard-4               10.156.1.3   33.89.216.84  RUNNING
```

Spawn VM instance

3. Connect To VM instance using SSH

```
ssh hans.wurst@33.89.216.84
```


Manage VM instances

1. Stop instance

```
gcloud compute instances stop big-data
```

2. Start instance

```
gcloud compute instances start big-data
```

!!! ALWAYS REMEMBER TO **STOP** YOUR **INSTANCE** when **not used**,
to do not waste gcloud credits!

Check if instance was successfully shut down:

```
gcloud compute instances list
NAME      ZONE            MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP  STATUS
big-data  europe-west3-c  n1-standard-4               10.156.1.3                TERMINATED
```

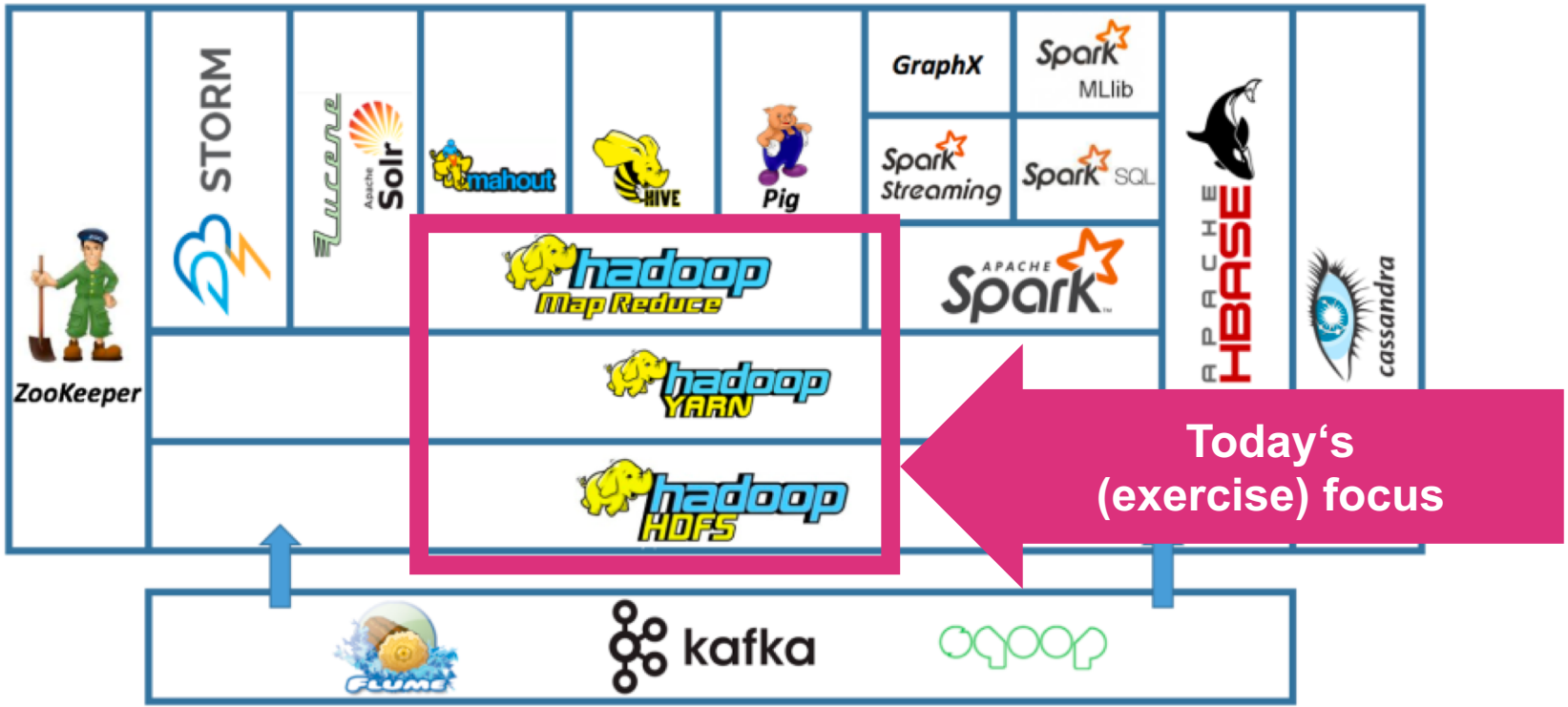

Break

TIME FOR A BREAK

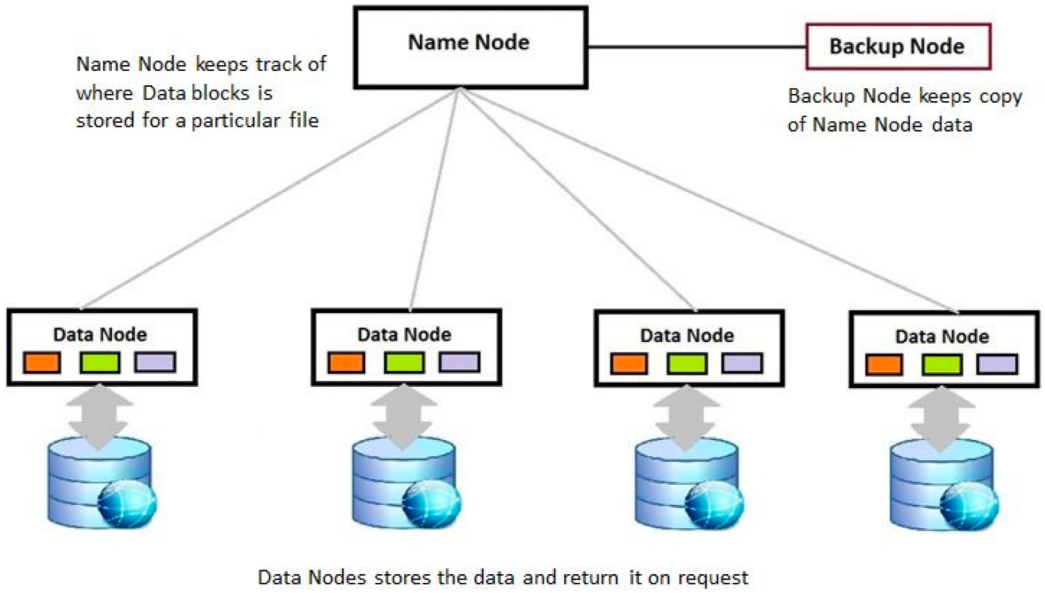
HandsOn - Hadoop

Quick Introduction To The Hadoop Ecosystem,
HDFS, Yarn and MapReduce

The Hadoop Ecosystem



The HDFS

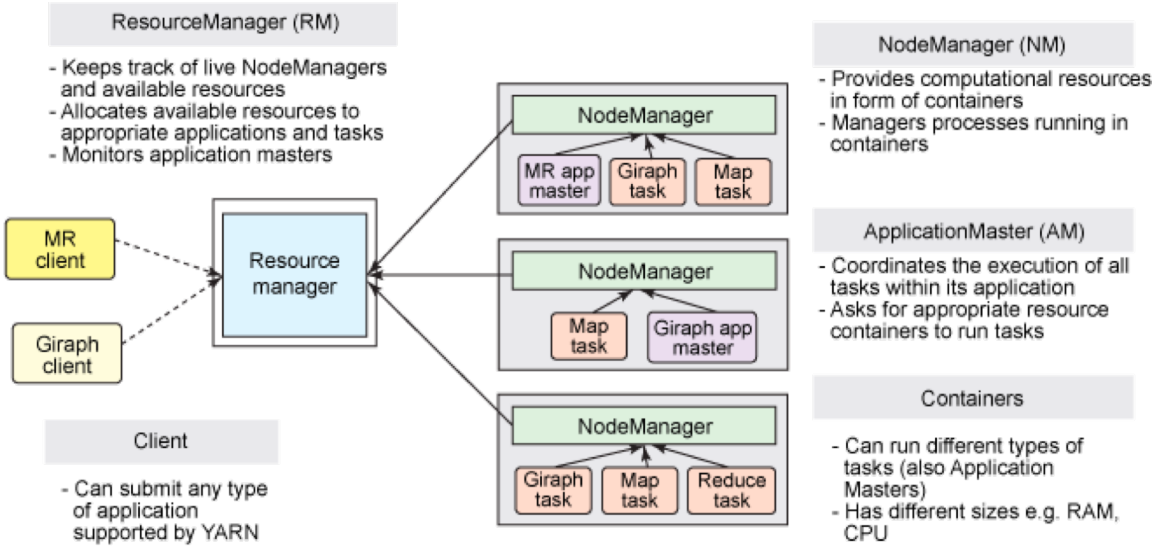


- ***H**adoop **D**istributed **F**iles**s**ystem*
- Highly available and distributed filesystem
- Can easily be scaled to a cluster of hundreds or even thousands of nodes and in this way **Petabytes of data**

YARN

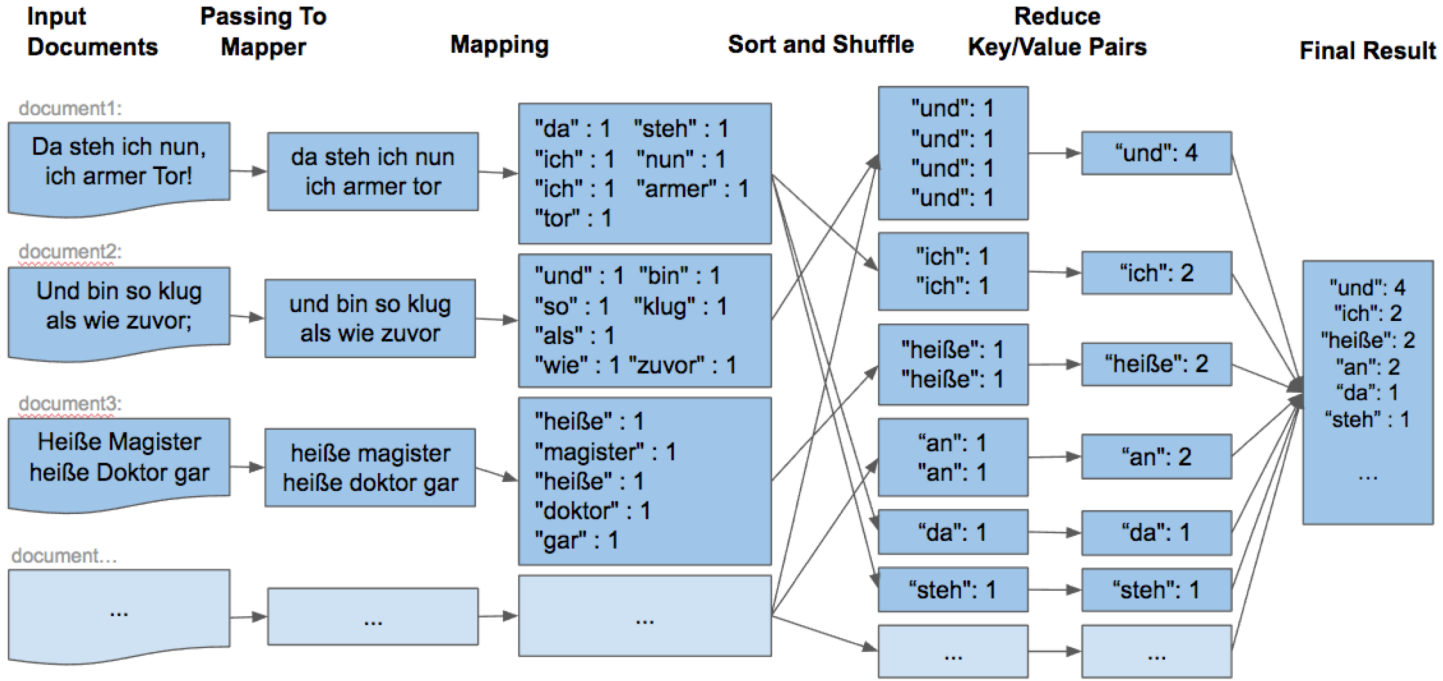
- ***Y**et **a**nother **R**essource **N**egotiator*

- Hadoop Ressource Manager (e.g. CPU, Memory, Job Scheduling, Preparation, Execution and Priorization)



MapReduce

- Programming paradigm for processing large datasets in parallel on a distributed cluster



Exercises Preparation

Setup Hadoop, HDFS, Yarn

Install and Setup Java

1. Install OpenJDK (JDK 8):

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

2. Verify installation:

```
java -version
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-1ubuntu1~18.04.1-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

2. SET *JAVA_HOME* and *JRE_HOME*:

```
sudo vi /etc/environment
```

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
JRE_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre"
```


Setup Hadoop User

1. Create User:

```
sudo adduser hadoop
sudo passwd hadoop
```

2. Switch To User:

```
sudo su hadoop
```

3. Switch Back To Root user:

```
exit
```


Setup SSH (needed by Hadoop components)

1. Install SSH and PDSH:

```
sudo apt-get install ssh pdsh
```

2. Create Private/Public Keypair for hadoop user (*without passphrase*):

```
sudo su hadoop
cd
ssh-keygen -t rsa -N "" -f /home/hadoop/.ssh/id_rsa
```

3. Add Public Key To Authorized Keys file (to enable passwordless ssh login)

```
cat /home/hadoop/.ssh/id_rsa.pub >> /home/hadoop/.ssh/authorized_keys
chmod 0600 /home/hadoop/.ssh/authorized_keys
```


Setup SSH (needed by Hadoop components)

4. Check If SSH Is Working

```
hadoop@big-data:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:YEUFliBVczkz2rvKWnYU9hB2ix2jnhBqLlbsJQfuBpE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1044-gcp x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
  System information as of Sat Oct 12 15:01:56 UTC 2019
  System load:  0.0               Processes:           117
  Usage of /:   5.8% of 28.90GB   Users logged in:     1
  Memory usage: 2%                IP address for ens4: 10.156.0.6
  Swap usage:   0%
30 packages can be updated.
17 updates are security updates.
Last login: Sat Oct 12 14:49:27 2019 from 80.144.211.195

hadoop@big-data:~$ exit
logout
Connection to localhost closed.

hadoop@big-data:~$
```

Install Hadoop

1. Download Hadoop (v3.1.1):

```
wget https://www-eu.apache.org/dist/hadoop/common/hadoop-3.1.2/hadoop-3.1.2.tar.gz
```

2. Extract Binaries:

```
tar -xvzf hadoop-3.1.2.tar.gz
```

3. Move Binaries:

```
mv hadoop-3.1.2 hadoop
```


Configure Hadoop

1. Set Up **UNIX** Environment Variables

```
vi .bashrc
```

```
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

```
source .bashrc
```

Configure Hadoop

2. Add **Hadoop** Environment Variables *(hadoop-env.sh)*

```
vi /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Configure Hadoop

3. Set Up **CORE** Variables *(core-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/core-site.xml
```

```
<configuration>
     <property>
          <name>fs.default.name</name>
          <value>hdfs://localhost:9000</value>
     </property>
</configuration>
```

Configure Hadoop

4. Set Up **HDFS** Variables *(hdfs-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/hdfs-site.xml
```

```
<configuration>
      <property>
            <name>dfs.replication</name>
            <value>1</value>
      </property>

      <property>
            <name>dfs.name.dir</name>
            <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
      </property>

      <property>
           <name>dfs.data.dir</name>
           <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
      </property>
</configuration>
```


Configure Hadoop

5. Set Up **MapReduce** Variables *(mapred-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/mapred-site.xml
```

```
<configuration>
      <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
      </property>
      <property>
            <name>yarn.app.mapreduce.am.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
      <property>
            <name>mapreduce.map.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
      <property>
            <name>mapreduce.reduce.env</name>
            <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
      </property>
</configuration>
```


Configure Hadoop

6. Set Up **YARN** Variables *(yarn-site.xml)*

```
vi /home/hadoop/hadoop/etc/hadoop/yarn-site.xml
```

```
<configuration>
          <property>
                    <name>yarn.nodemanager.aux-services</name>
                    <value>mapreduce_shuffle</value>
          </property>
</configuration>
```

Configure Hadoop

7. Clear HDFS

```
hdfs namenode -format
```

8. Start HDFS:

```
start-dfs.sh
```

9. Start YARN:

```
start-yarn.sh
```


Check Hadoop/HDFS

10. Run Admin Status Report

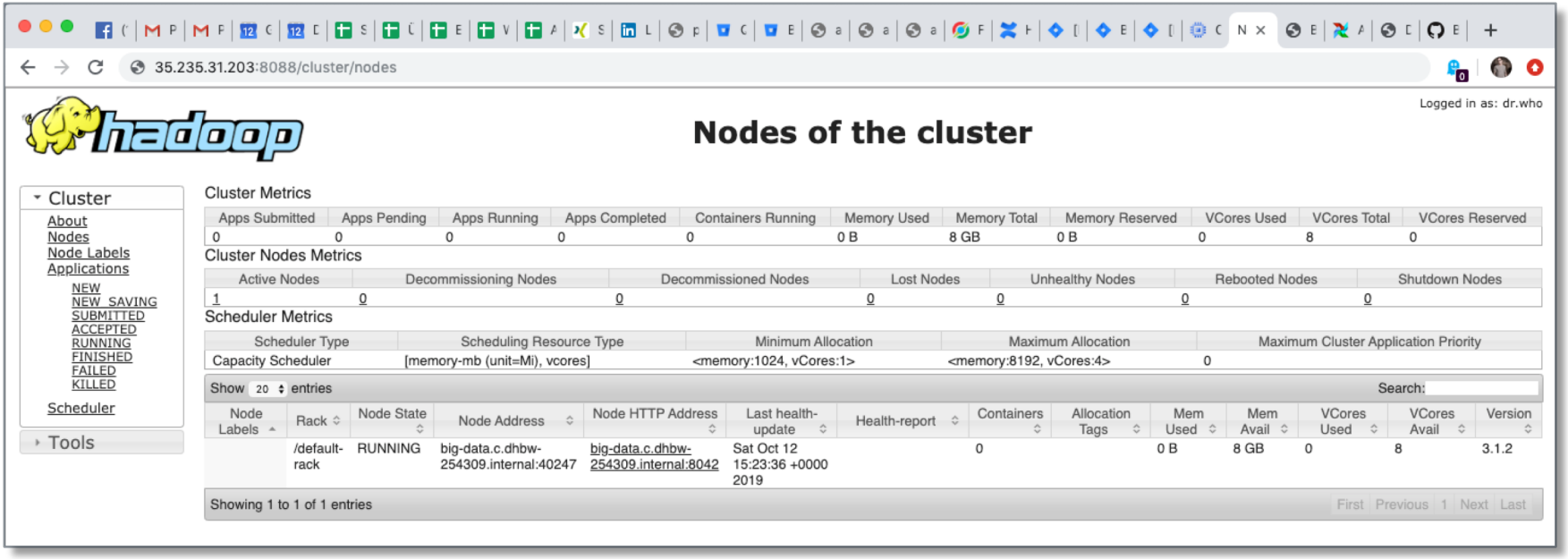
```
hdfs dfsadmin -report
```

```
Configured Capacity: 31035637760 (28.90 GB)
Present Capacity: 28187471872 (26.25 GB)
DFS Remaining: 28187447296 (26.25 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0.00%
Replicated Blocks:
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
Missing blocks (with replication factor 1): 0
Low redundancy blocks with highest priority to recover: 0
Pending deletion blocks: 0
Erasure Coded Block Groups:
Low redundancy block groups: 0
Block groups with corrupt internal blocks: 0
Missing block groups: 0
Low redundancy blocks with highest priority to recover: 0
Pending deletion blocks: 0

-------------------------------------------------
Live datanodes (1):
Name: 127.0.0.1:9866 (localhost)
Hostname: big-data.c.dhbw-253679.internal
Decommission Status : Normal
Configured Capacity: 31035637760 (28.90 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 2831388672 (2.64 GB)
DFS Remaining: 28187447296 (26.25 GB)
DFS Used%: 0.00%
DFS Remaining%: 90.82%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 1
Last contact: Sat Oct 12 15:19:44 UTC 2019
Last Block Report: Sat Oct 12 15:18:29 UTC 2019
Num of Blocks: 0
```

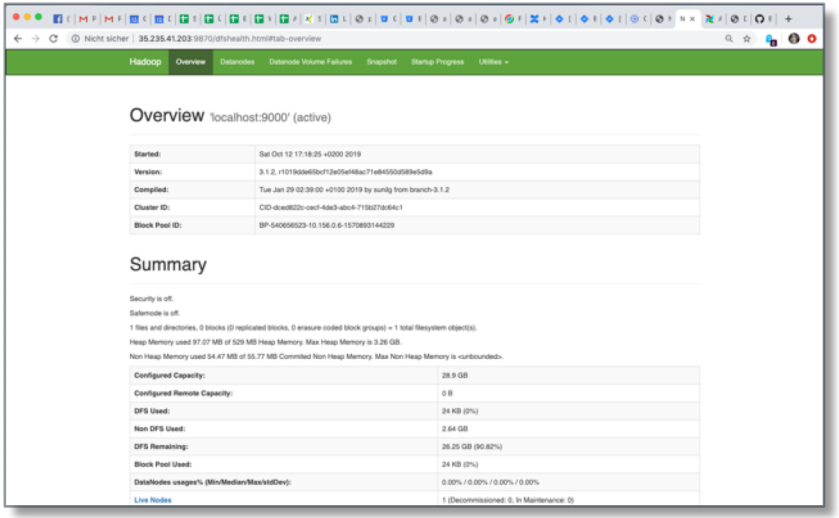
Check Hadoop/HDFS

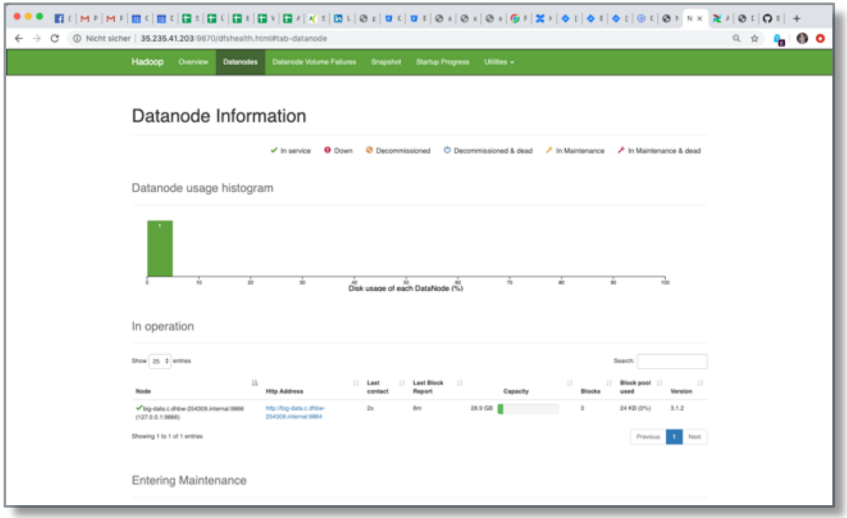
11. Check Ressource Manager Landing Page (http://XXX.XXX.XXX.XXX:8088/cluster):



Check Hadoop/HDFS

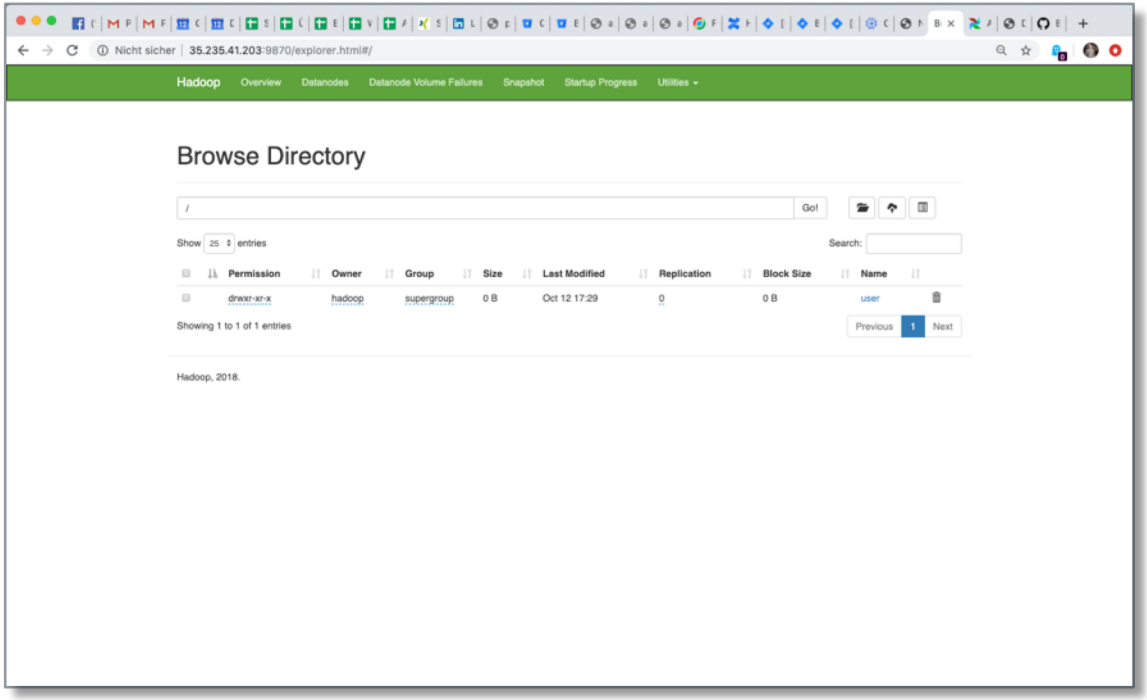
12. Check NameNode Landing and Status Page (http://XXX.XXX.XXX.XXX:9870):





Check Hadoop/HDFS

13. Check HDFS File Browser (http://XXX.XXX.XXX.XXX:9870/explorer.html#/)



Working with HDFS

1. Create User Directory (*on HDFS*):

```
hadoop fs -mkdir /user
hadoop fs -mkdir /user/hadoop
```

2. List Directories (*on HDFS*):

```
hadoop@big-data:~$ hadoop fs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2019-10-12 15:29 /user
hadoop@big-data:~$
```


Working with HDFS

3. Copy File (just a *random log file*) from local directory to HDFS:

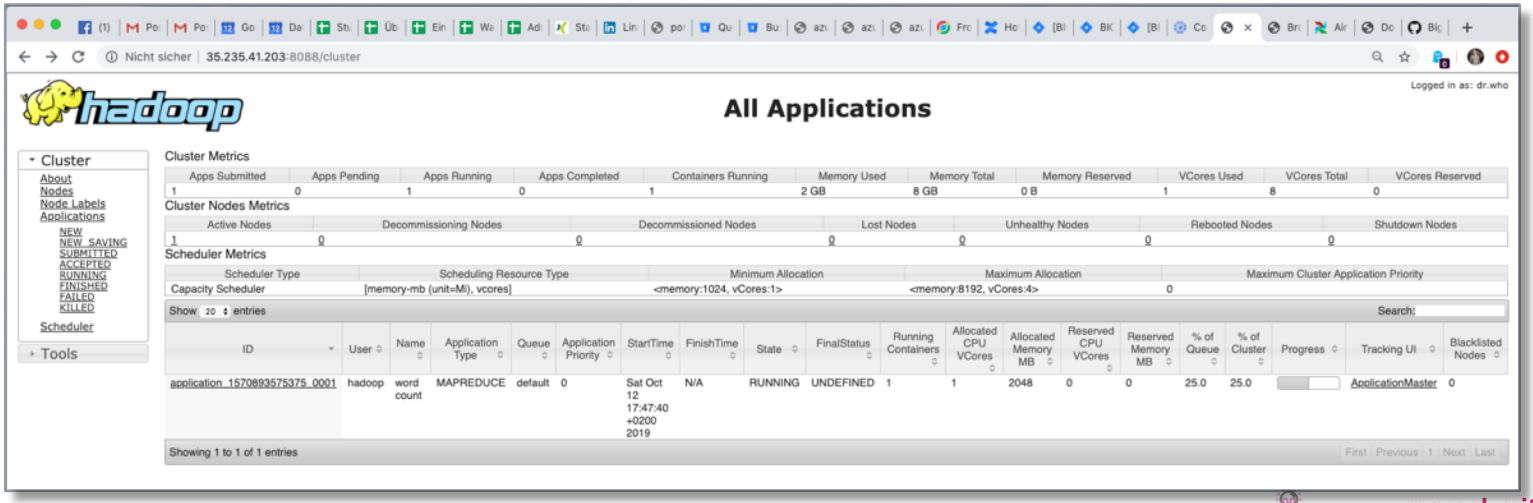
```
hadoop fs -put /var/log/dpkg.log /user/hadoop/dpkg.log
```


Run Example MapReduce Job

1. Using MapReduce WordCount Jar provided by Hadoop to count words within file *dpkg.log*

```
hadoop jar hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar
wordcount /user/hadoop/dpkg.log /user/hadoop/test_output
```

2. View Running MapReduce Job:



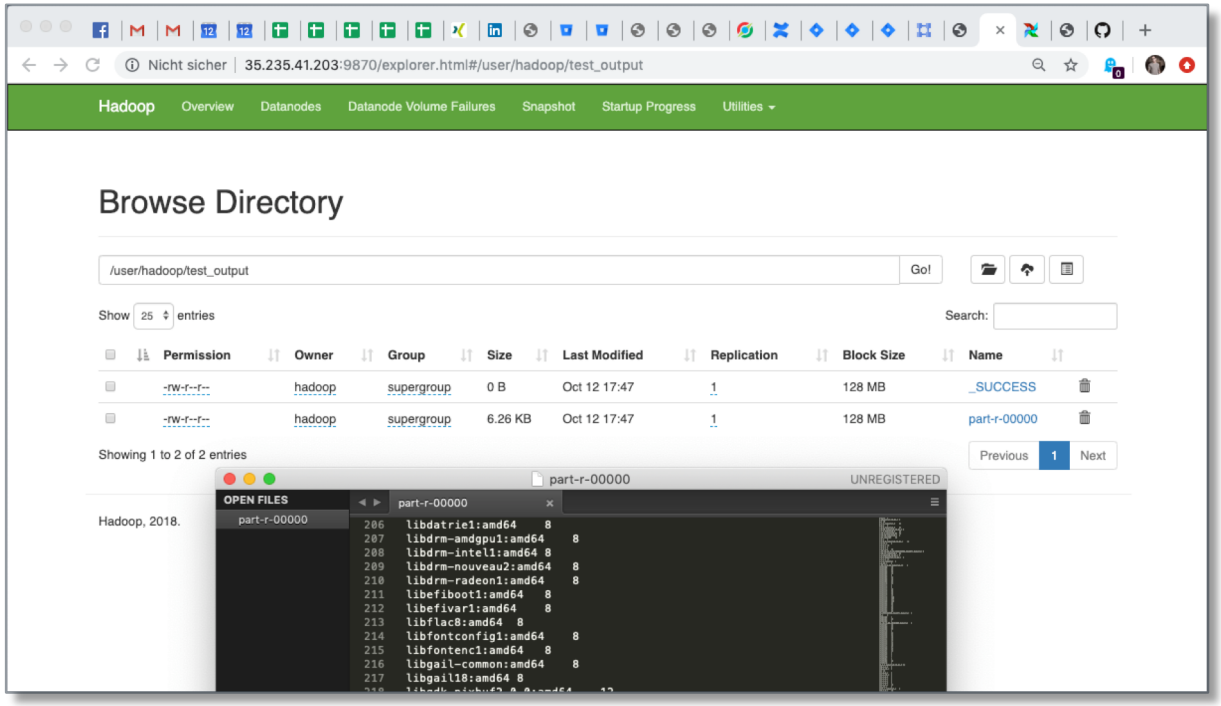
Run Example MapReduce Job

3. Take A Look At The Output/Result (*via Bash*):

```
hadoop@big-data:~$ hadoop fs -cat /user/hadoop/test_output/part-r-00000
…
libglx0:amd64 8
libgraphite2-3:amd64 8
libgtk2.0-0:amd64 8
libgtk2.0-bin:amd64 8
libgtk2.0-common:all 9
libharfbuzz0b:amd64 8
libice-dev:amd64 8
libice6:amd64 8
libjbig0:amd64 8
libjpeg-turbo8:amd64 8
libjpeg8:amd64 8
libnss3:amd64 8
libogg0:amd64 8
libpango-1.0-0:amd64 8
libpangocairo-1.0-0:amd64 8
…
```


Run Example MapReduce Job

4. Take A Look At The Output/Result (*via Web HDFS File Browser*):



Exercises I

Hadoop, HDFS, Yarn

Exercises

1. **Clone git repo** (to get sample data):

```
git clone https://github.com/marcelmittelstaedt/BigData.git
```

2.

- **Copy sample file** (*/BigData/exercises/winter_semester_2019-2020/01_hadoop/sample_data/Faust_1.txt*) from Git Repo **to HDFS**.
- Use and **run** default **MapReduce** Jar (*hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar*) **to calculate wordcount** for text file.
- **Copy result** of MapReduce job **back to local** ubuntu **filesystem**.

3.

- Use and **run** default **MapReduce** Jar (*hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar*) to **get the count of occurences** of the exact string ***‚Faust‘*** within text file.
- **Copy result** of MapReduce job **back to local** ubuntu **filesystem**.
- **Tip:** don‘t use *wordcount* part of jar but another MapReduce program on next slide.

Help

MapReduce Examples within *hadoop-mapreduce-examples-3.1.1.jar*:

aggregatewordcount:	An Aggregate based mapreduce program that counts the words in the input files.
aggregatewordhist:	An Aggregate based mapreduce program that computes the histogram of the words in the input files.
bbp:	A mapreduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
dbcount:	An example job that counts the pageview logs stored in a database.
distbbp:	A mapreduce program that uses a BBP-type formula to compute exact bits of Pi.
grep:	A mapreduce program that counts the matches of a regex in the input.
join:	A job that performs a join over sorted, equally partitioned datasets.
multifilewc:	A job that counts words from several files.
pentomino:	A mapreduce tile laying program to find solutions to pentomino problems.
pi:	A mapreduce program that estimates Pi using a quasi-Monte Carlo method.
randomtextwriter:	A mapreduce program that writes 10 GB of random textual data per node.
randomwriter:	A mapreduce program that writes 10 GB of random data per node.
secondarysort:	An example defining a secondary sort to the reduce phase.
sort:	A mapreduce program that sorts the data written by the random writer.
sudoku:	A sudoku solver.
teragen:	Generate data for the terasort.
terasort:	Run the terasort.
teravalidate:	Checking results of terasort.
wordcount:	A mapreduce program that counts the words in the input files.
wordmean:	A mapreduce program that counts the average length of the words in the input files.
wordmedian:	A mapreduce program that counts the median length of the words in the input files.
wordstandarddeviation:	A mapreduce program that counts the standard deviation of the length of the words in the input files.

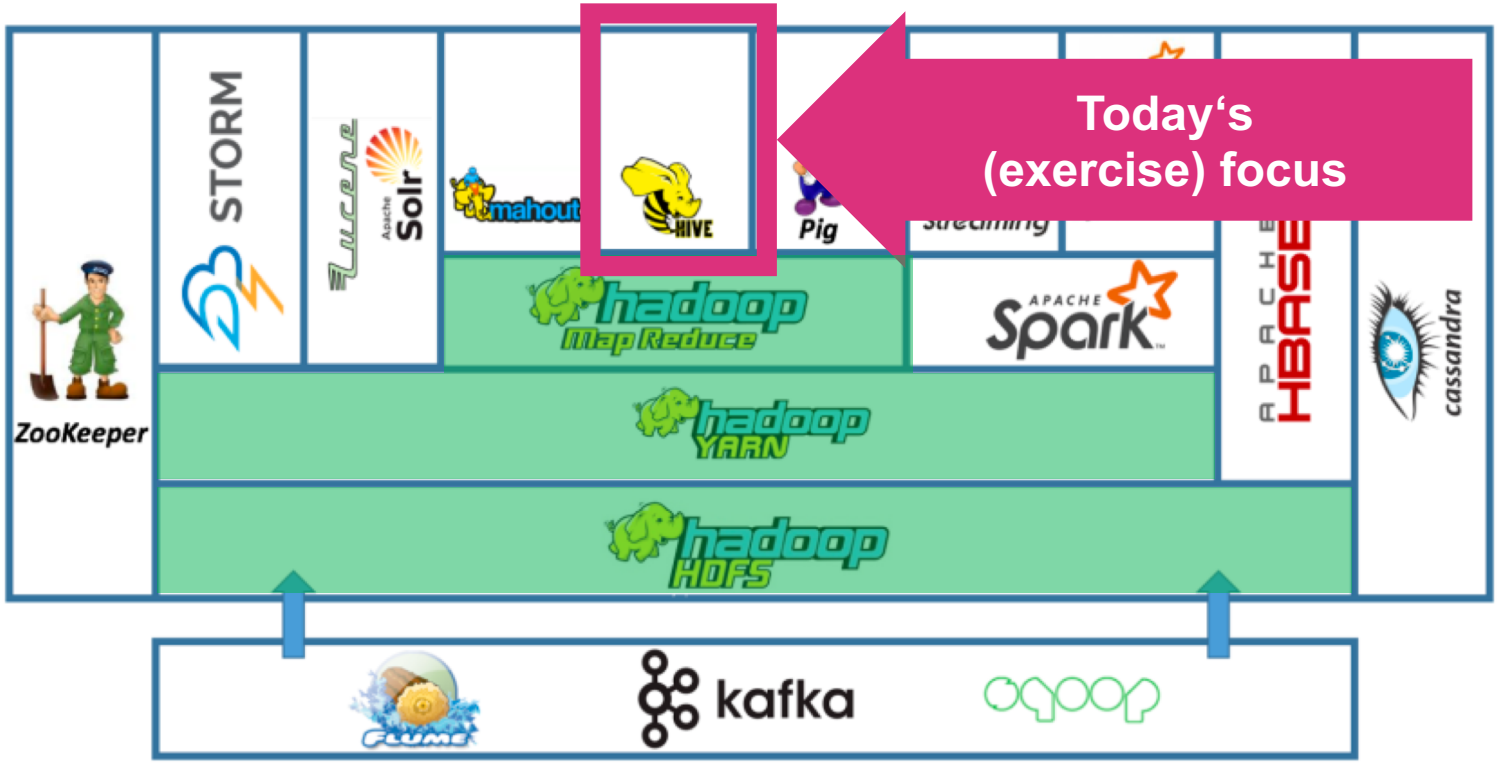
Break

TIME FOR A BREAK

HandsOn – **Hive and HiveQL**

Quick Introduction To Hive and HiveQL

The Hadoop Ecosystem



Apache Hive

Apache Hive

- Initial Release in October 2010
- written in Java
- current Version: 3.1.2

http://hive.apache.org

- **What is Hive:**
  - **easy to use** (HiveQL)
  - **based on Hadoop** (HDFS, YARN)
  - **good scale-out capabilities** (e.g. by partitioning and underlying HDFS and YARN)
  - **best used for:**
    - (BigData) **datawarehousing tasks**
    - a **DataLake**
    - **interface** for Analysts, DataScientists, Develo pers
    - **adhoc** (batch) **querying, aggregation** and **a nalysis** of large amounts of data (**PB!**) and hundreds of nodes
- **What Hive is NOT**:
  - **transactional database**
  - **highly responsive**

HiveQL

- SQL like **query language**
- Supported by: Hive CLI (*deprecated*), Beeline CLI and most JDBC clients
- Hive supported HDFS file formats:
  - **TextFile** (even compressed by gzip or bzip2)
  - **SequenceFile**
  - **RCFile**
  - **ORC**
  - **Parquet**
  - **Avro**

HDFS/Hive - **Wordcount**

- Previous WordCount example achieved by using Hive and HiveQL:

```
CREATE TABLE faust (line STRING);

LOAD DATA INPATH '/user/hadoop/faust' OVERWRITE INTO TABLE faust;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\\s')) AS word FROM faust) temp
 GROUP BY word ORDER BY word;
```

```
SELECT * from word_counts ORDER BY count DESC LIMIT 10;
```

word	count
und	509
die	463
der	440
ich	435
Und	400
nicht	346
zu	319
[…]	

Exercises Preparation

Install and Setup Hive

Create VM Instance

1. Delete previously created VM instance:

```
gcloud compute instances delete big-data
```

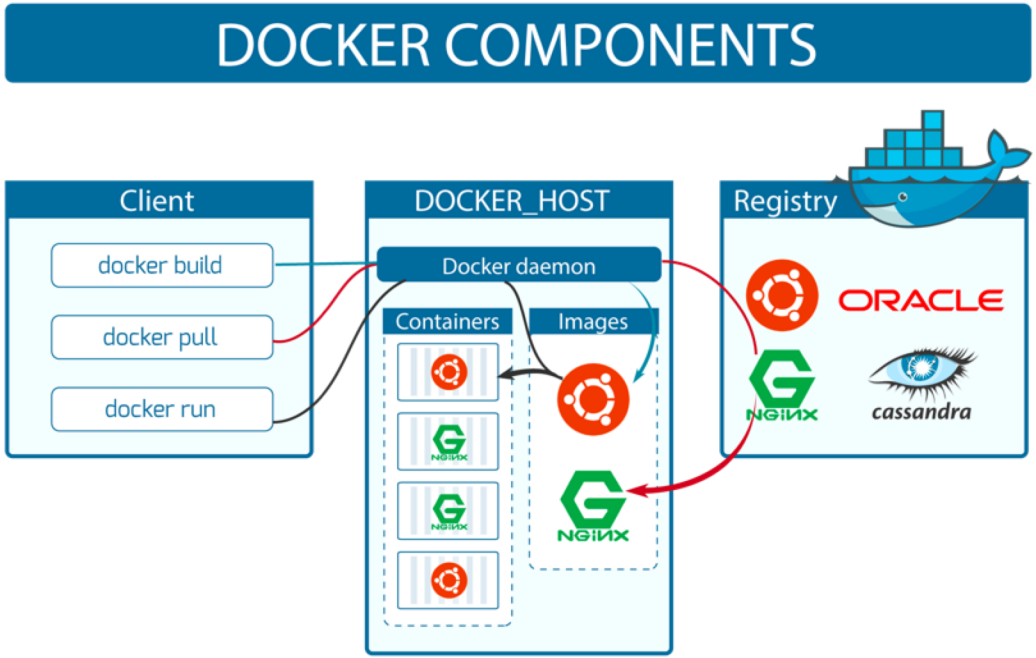
2. Create new instance:

```
gcloud compute --project=[your-project-id] instances create big-data \
            --zone=europe-west3-c \
            --machine-type=n1-standard-4 \
            --subnet=default --network-tier=PREMIUM \
            --maintenance-policy=MIGRATE \
            --image=ubuntu-1804-bionic-v20190918 \
            --image-project=ubuntu-os-cloud \
            --boot-disk-size=30GB \
            --boot-disk-type=pd-standard \
            --boot-disk-device-name=big-data
```

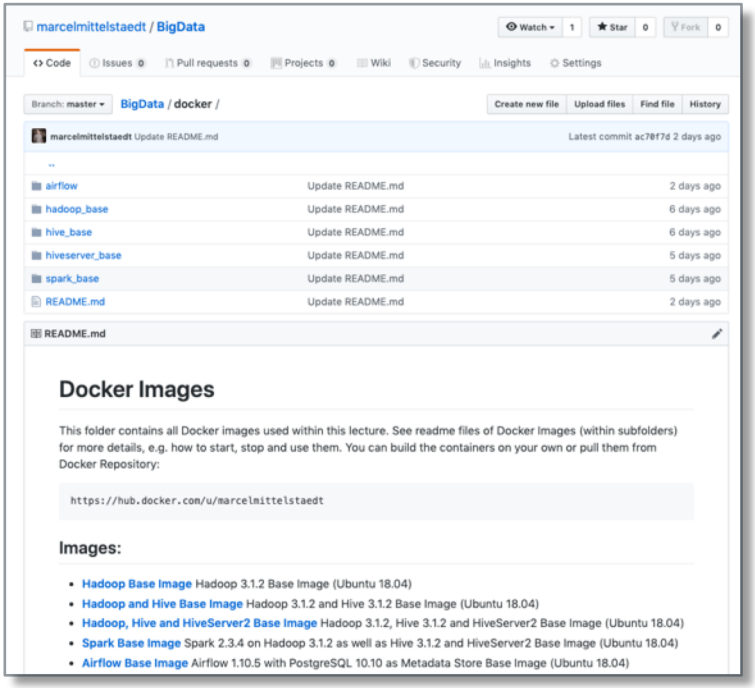
```
ssh hans.wurst@XXX.XXX.XXX.XXX
```


Docker

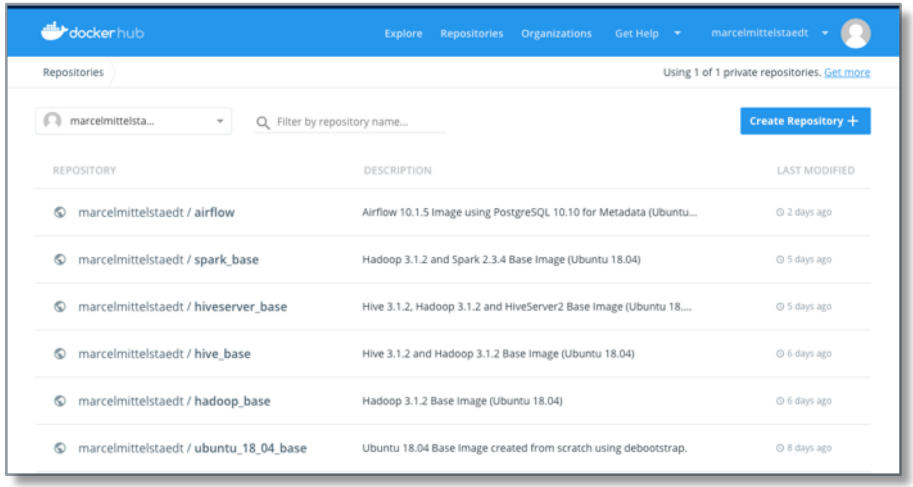
To speed things up and not waste time on installation and configuration of Hive and other tools, we will make use of docker container I‘ve already prepared.



Docker Images/Dockerfiles



https://github.com/marcelmittelstaedt/BigData/tree/master/docker



https://hub.docker.com/u/marcelmittelstaedt

Setup Docker Container

3. Install and setup docker

```
sudo apt-get update
sudo apt-get install docker.io
sudo usermod -aG docker $USER
# exit and login again
```

4. Pull Hadoop with Hive Image

```
docker pull marcelmittelstaedt/hive_base:latest
```

5. Start Container from pulled image:

```
docker run -dit --name hive_base_container -p 8088:8088 -p 9870:9870 -p 9864:
9864 marcelmittelstaedt/hive_base:latest
```


Setup Docker Container

6. Show Running Container:

```
docker ps -a
```

```
marcel.mittelstaedt@big-data:~$ docker ps -a
CONTAINER ID        IMAGE                                  COMMAND             CREATED             STATUS              PORTS                                        NAMES
c821a0e1bdcf        marcelmittelstaedt/hive_base:latest    "/startup.sh"       6 minutes ago       Up 6 minutes        0.0.0.0:8088->8088/tcp, 0.0.0.0:9870->9870/tcp   hive_base_container
marcel.mittelstaedt@big-data:~$
```

7. Show Logs of container (wait till finished):

```
docker logs hive_base_container

[…]
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [c821a0e1bdcf]
Stopping nodemanagers
Stopping resourcemanager
Container Startup finished.
```

Setup Docker Container

8. Get a shell inside the container:

```
hans.wurst@big-data:~$ docker exec -it hive_base_container bash
root@c821a0e1bdcf:/#
```

9. Switch to hadoop user:

```
root@c821a0e1bdcf:/# sudo su hadoop
hadoop@c821a0e1bdcf:/$ cd
hadoop@c821a0e1bdcf:~$
```

10. Start DFS and YARN:

```
start-all.sh
```


Install and Setup Hive

11. Test if Hive Installation and Configuration is successful. Start Hive:

```
hive
```

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7
.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = c120d0b1-9025-43db-96e4-48ccfb875f1a

Logging initialized using configuration in jar:file:/home/hadoop/hive/lib/hive-common-3.1.0.jar
!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider us
ing a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = fdd6f06a-d4e4-48e6-8971-997e8a0a8e2c
hive>
```


Install and Setup Hive

12. Execute First SQL Query:

```
hive> show databases;
OK
default
Time taken: 0.083 seconds, Fetched: 1 row(s)
hive>
```


Hive: Create and Work with External Tables

Using public dataset of IMDb.com

Get IMDb Data And Move It To HDFS

1. Get **IMDb Data** (*https://www.imdb.com/interfaces/*):

```
wget https://datasets.imdbws.com/title.basics.tsv.gz
wget https://datasets.imdbws.com/title.ratings.tsv.gz
```

2. Uncompress IMDb Data:

```
gunzip title.basics.tsv.gz
gunzip title.ratings.tsv.gz
```

3. Create HDFS Directories for IMDb Data:

```
hadoop fs -mkdir /user/hadoop/imdb
hadoop fs -mkdir /user/hadoop/imdb/title_basics
hadoop fs -mkdir /user/hadoop/imdb/title_ratings
```

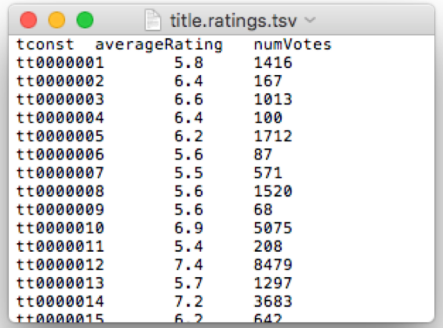

Create External Tables In Hive

4. Transfer IMDb data files to HDFS:

```
hadoop fs -put title.basics.tsv /user/hadoop/imdb/title_basics/title.basics.tsv
hadoop fs -put title.ratings.tsv /user/hadoop/imdb/title_ratings/title.ratings.tsv
```

5. Create External Table `title_ratings` (file *title.ratings.tsv*) in Hive:

```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS title_ratings(
          tconst STRING,
          average_rating DECIMAL(2,1),
          num_votes BIGINT
) COMMENT 'IMDb Ratings'
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS
TEXTFILE LOCATION '/user/hadoop/imdb/title_ratings'
TBLPROPERTIES ('skip.header.line.count'='1');
```



Create External Tables In Hive

6. Create External Table `title_basics` for file *title.basics.tsv* in Hive:

title.basics.tsv

```
tconst  titleType       primaryTitle    originalTitle   isAdult startYear       endYear runtimeMinutes  genres
tt0000001       short   Carmencita      Carmencita      0       1894    \N      1       Documentary,Short
tt0000002       short   Le clown et ses chiens  Le clown et ses chiens  0       1892    \N      5       Animation,Short
tt0000003       short   Pauvre Pierrot  Pauvre Pierrot  0       1892    \N      4       Animation,Comedy,Romance
tt0000004       short   Un bon bock     Un bon bock     0       1892    \N      \N      Animation,Short
tt0000005       short   Blacksmith Scene        Blacksmith Scene        0       1893    \N      1       Comedy,Short
tt0000006       short   Chinese Opium Den       Chinese Opium Den       0       1894    \N      1       Short
tt0000007       short   Corbett and Courtney Before the Kinetograph     Corbett and Courtney Before the Kinetograph     0       1894    \N      1       Short,Sport
tt0000008       short   Edison Kinetoscopic Record of a Sneeze  Edison Kinetoscopic Record of a Sneeze  0       1894    \N      1       Documentary,Short
tt0000009       movie   Miss Jerry      Miss Jerry      0       1894    \N      45      Romance
tt0000010       short   Employees Leaving the Lumière Factory   La sortie de l'usine Lumière à Lyon      0       1895    \N      1       Documentary,Short
tt0000011       short   Akrobatisches Potpourri Akrobatisches Potpourri 0       1895    \N      1       Documentary,Short
tt0000012       short   The Arrival of a Train  L'arrivée d'un train à La Ciotat        0       1896    \N      1       Documentary,Short
tt0000013       short   The Photographical Congress Arrives in Lyon     Neuville-sur-Saône: Débarquement du congrès des photographes à Lyon     0       1895    \N      1       Documentary,Short
tt0000014       short   Tables Turned on the Gardener   L'arroseur arrosé       0       1895    \N      1       Comedy,Short
tt0000015       short   Autour d'une cabine     Autour d'une cabine     0       1894    \N      2       Animation,Short
tt0000016       short   Barque sortant du port  Barque sortant du port  0       1895    \N      1       Documentary,Short
tt0000017       short   Italienischer Bauerntanz        Italienischer Bauerntanz        0       1895    \N      1       Documentary,Short
tt0000018       short   Das boxende Känguruh    Das boxende Känguruh    0       1895    \N      1       Short
tt0000019       short   The Clown Barber        The Clown Barber        0       1898    \N      \N      Comedy,Short
tt0000020       short   The Derby 1895  The Derby 1895  0       1895    \N      1       Documentary,Short,Sport
tt0000022       short   Blacksmith Scene        Les forgerons   0       1895    \N      1       Documentary,Short
tt0000023       short   The Sea Baignade en mer 0       1895    \N      1       Documentary,Short
tt0000024       short   Opening of the Kiel Canal       Opening of the Kiel Canal       0       1895    \N      \N      News,Short
tt0000025       short   The Oxford and Cambridge University Boat Race   The Oxford and Cambridge University Boat Race   0       1895    \N      \N      News,Short,Sport
tt0000026       short   The Messers. Lumière at Cards   Partie d'écarté 0       1896    \N      1       Documentary,Short
tt0000027       short   Cordeliers' Square in Lyon      Place des Cordeliers à Lyon     0       1895    \N      1       Documentary,Short
tt0000028       short   Fishing for Goldfish    La pêche aux poissons rouges    0       1895    \N      1       Documentary,Short
tt0000029       short   Baby's Dinner   Repas de bébé   0       1895    \N      1       Documentary,Short
tt0000030       short   Rough Sea at Dover      Rough Sea at Dover      0       1895    \N      1       Documentary,Short
```

Create External Tables In Hive

6. Create External Table **`title_basics`** for file *title.basics.tsv* in Hive:

```
hive > CREATE EXTERNAL TABLE IF NOT EXISTS title_basics (
    tconst STRING,
    title_type STRING,
    primary_title STRING,
    original_title STRING,
    is_adult DECIMAL(1,0),
    start_year DECIMAL(4,0),
    end_year STRING,
    runtime_minutes INT,
    genres STRING
) COMMENT 'IMDb Movies' ROW FORMAT DELIMITED FIELDS TERMINATED BY
'\t' STORED AS TEXTFILE LOCATION '/user/hadoop/imdb/title_basics'
TBLPROPERTIES ('skip.header.line.count'='1');
```


Create External Tables In Hive

7. Query Table `title_basics` in Hive using SQL (HiveQL):

```
hive> select * from title_basics limit 3;
OK
tt0000001 short Carmencita Carmencita 0 1894 NULL 1 Documentary,Short
tt0000002 short Le clown et ses chiens Le clown et ses chiens 0 1892 NULL 5 Animation,Short
tt0000003 short Pauvre Pierrot Pauvre Pierrot 0 1892 NULL 4 Animation,Comedy,Romance
Time taken: 0.139 seconds, Fetched: 5 row(s)
hive>
```

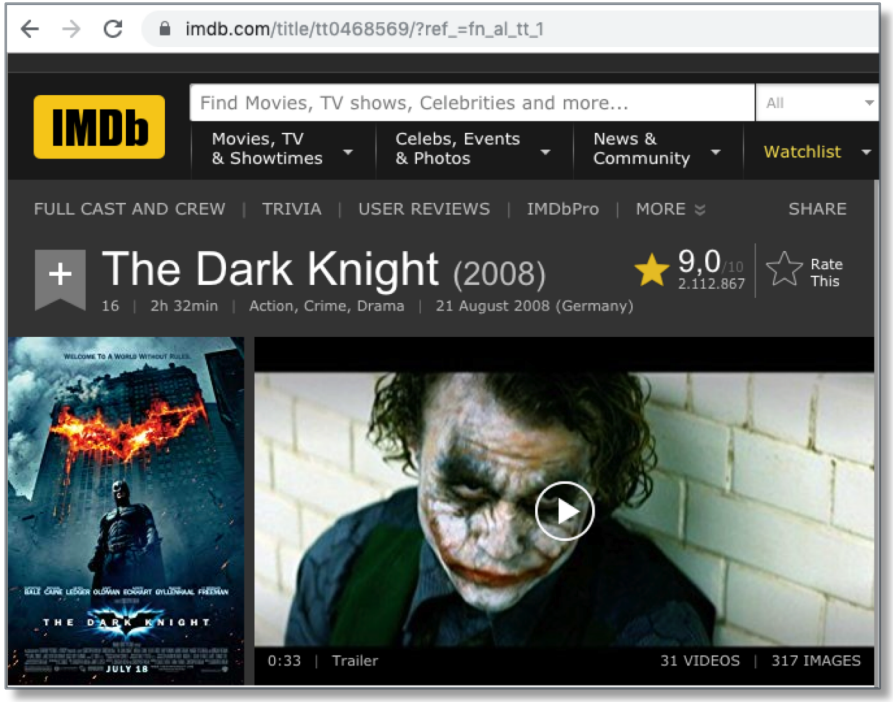
8. Query Table `title_ratings` in Hive using SQL (HiveQL):

```
hive> select * from title_ratings limit 3;
OK
tt0000001 5.6 1540
tt0000002 6.1 186
tt0000003 6.5 1199
Time taken: 0.119 seconds, Fetched: 3 row(s)
hive>
```


Create External Tables In Hive

9. Run a complex query which starts a MapReduce Job on Yarn, e.g. get Rating of movie „*The Dark Knight*“:

```
SELECT
   *
FROM
   title_basics b
   JOIN title_ratings r ON (b.tconst=r.tconst)
WHERE
   original_title = 'The Dark Knight'
   AND title_type='movie';
```



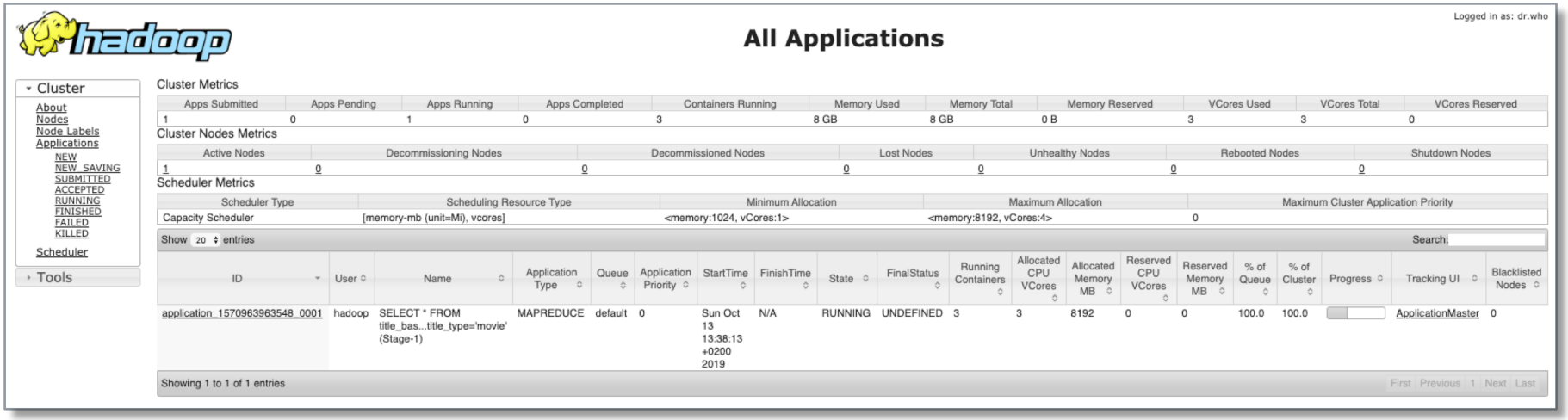
Create External Tables In Hive

9. Execute Query

```
hive> SELECT * FROM title_basics b JOIN title_ratings r ON (b.tconst=r.tconst) WHERE original_title =
'The Dark Knight' and title_type='movie';
[…]
Starting Job = job_1570963963548_0001, Tracking URL = http://c821a0e1bdcf:8088/proxy/application_1570963963548_0001/
Kill Command = /home/hadoop/hadoop/bin/mapred job  -kill job_1570963963548_0001
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 3
2019-10-13 11:38:22,730 Stage-1 map = 0%,  reduce = 0%
2019-10-13 11:38:34,307 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 18.13 sec
2019-10-13 11:38:45,726 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 28.54 sec
2019-10-13 11:38:48,897 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 36.24 sec
2019-10-13 11:38:53,112 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 43.51 sec
2019-10-13 11:38:56,229 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 49.27 sec
MapReduce Total cumulative CPU time: 49 seconds 270 msec
Ended Job = job_1570963963548_0001
MapReduce Jobs Launched:
Stage-Stage-3: Map: 2   Cumulative CPU: 34.83 sec   HDFS Read: 529291906 HDFS Write: 289 SUCCESS
Total MapReduce CPU Time Spent: 34 seconds 830 msec
OK
tt0468569 movie The Dark Knight The Dark Knight 0 2008 NULL 152 Action,Crime,Drama tt0468569 9.0 2111245
Time taken: 53.094 seconds, Fetched: 1 row(s)
hive>
```


Create External Tables In Hive

9. Take a look at YARN (http://XXX.XXX.XXX.XXX:8088/cluster/):



Exercises II

Hive: Create and Work with External Tables

HDFS and HiveQL Exercises - **IMDB**

1. Execute Tasks of previous HandsOn Slides
2. Download https://datasets.imdbws.com/name.basics.tsv.gz
3. Create HDFS Directory **/user/hadoop/imdb/name_basics**/ for file name.basics.tsv
4. Create External Hive Table `name_basics` for name.basics.tsv
5. Use HiveQL to answer following questions:
   a) How many **movies** and how many **TV series** are within the IMDB dataset?
   b) Who is the **youngest** actor/writer/… within the dataset?
   c) Create a list (`tconst, original_title, start_year, average_rating, num_votes`) of movies which are:
   - equal or newer than year 2010
   - have an average rating equal or better than 8,1
   - have been voted more than 100.000 times

   d) How many movies are in list of c)?

HDFS and HiveQL Exercises - **IMDB**

5. Use HiveQL to answer following questions:

e) We want to know which years have been great for cinema.
Create a list with one row per year and a related count of movies which:
- have an average rating better than 8
- have been voted more than 100.000 times

ordered descending by count of movies.

Well Done

WE'RE DONE FOR ...TODAY