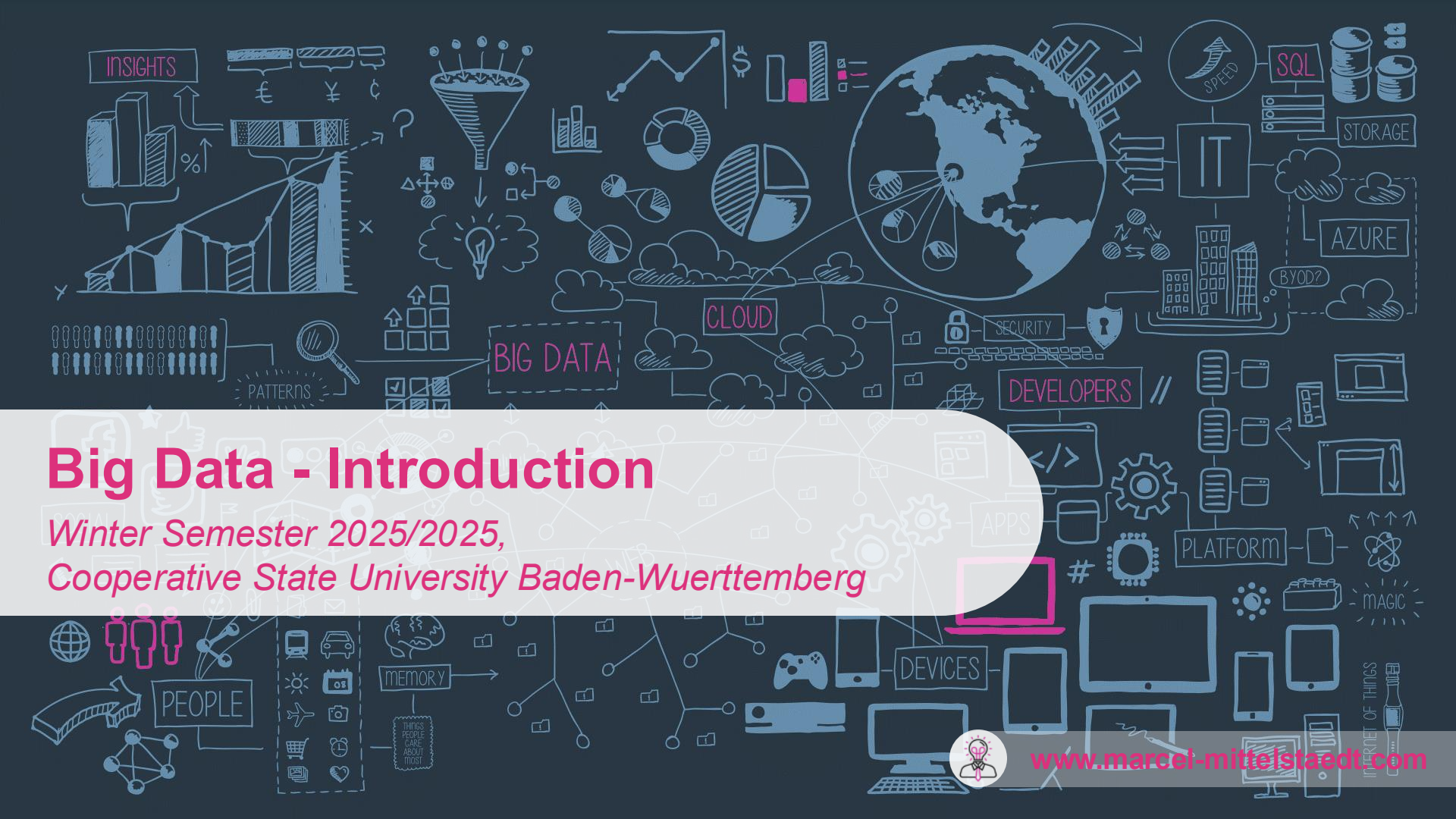


Big Data - Introduction

Winter Semester 2025/2025,
Cooperative State University Baden-Wuerttemberg

www.marcel-mittelstaedt.com

Agenda – 29.09.2025

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 – Setup Google Cloud, Hadoop and MapReduce

Quick Introduction To Google Cloud, Create Own Hadoop Cluster with YARN and Run MapReduce Jobs on it

About This Lecture

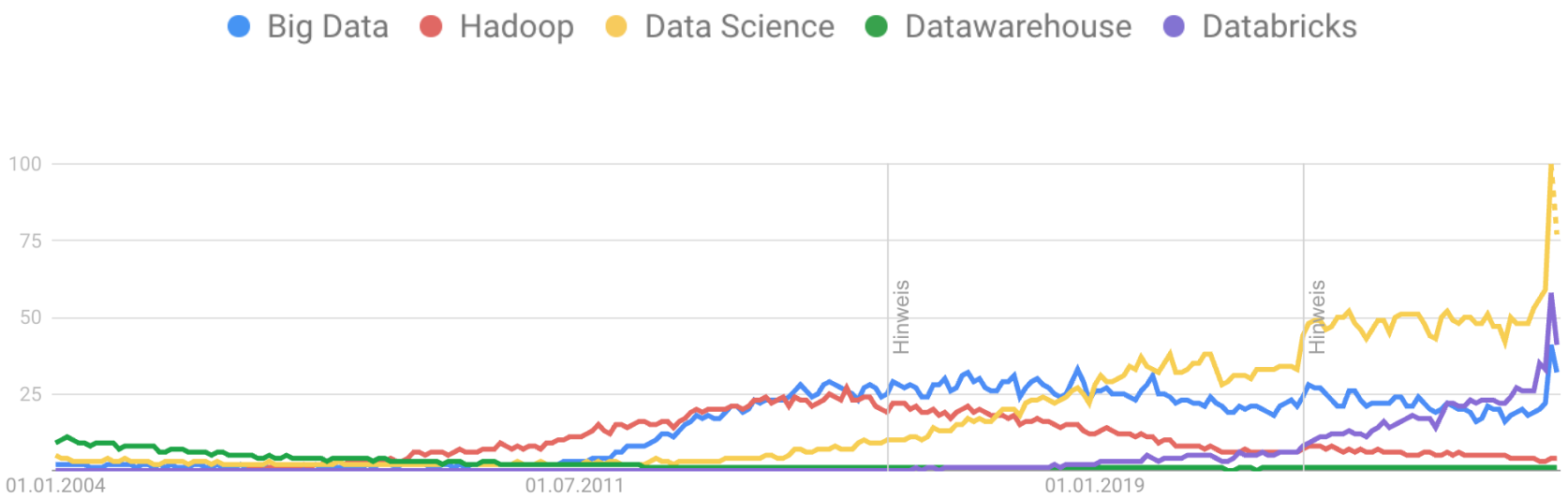
Scope, Schedule, Materials, Prerequisites, Exam, Feedback, Questions

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, 28.09.2025

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

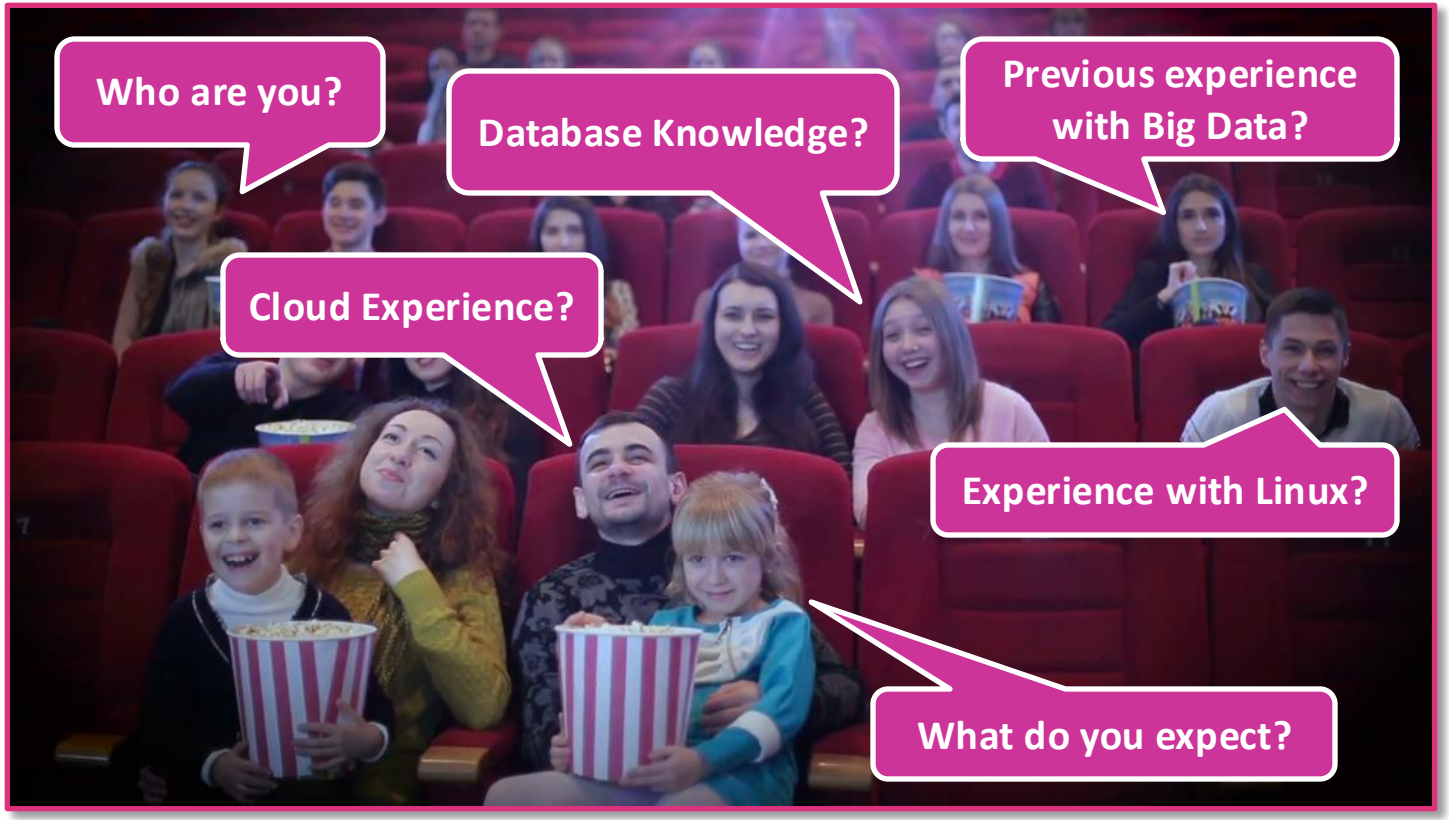
2014 - 2018
jan - nov

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?
Cloud Experience?
Experience with Linux?
What do you expect?

Schedule

			Lecture Topic	HandsOn
29.09.2025	**13:15-15:45**	**Ro. N/A**	**About This Lecture, Introduction to Big Data**	Setup Google Cloud, Create Own Hadoop Cluster and Run MapReduce
06.10.2025	13:15-15:45	Ro. N/A	(Non-)Functional Requirements Of Distributed Data-Systems, Data Models and Access	Hive and HiveQL
13.10.2025	13:15-15:45	Ro. N/A	Challenges Of Distributed Data Systems: Partitioning	Data Partitioning (with HDFS/Hive, HiveServer2)
20.10.2025	13:15-15:45	Ro. N/A	Challenges Of Distributed Data Systems: Replication	Spark and Scala
27.10.2025	13:15-15:45	Ro. N/A	ETL Workflow and Automation	PySpark and Notebooks (Jupyter)
03.11.2025	13:15-15:45	Ro. N/A	Batch and Stream Processing	Airflow
10.11.2025	13:15-15:45	Ro. N/A	Practical Exam	Work On Practical Exam
17.11.2025	13:15-15:45	Ro. N/A	Practical Exam	Work On Practical Exam
24.11.2025	**13:15-15:45**	**Ro. N/A**	**Practical Exam Presentation**	
01.12.2025	**13:15-15:45**	**Ro. N/A**	**Practical Exam Presentation**	

Schedule

29.09.2025	**13:15-15:45**	**Ro. N/A**
06.10.2025	13:15-15:45	Ro. N/A
13.10.2025	13:15-15:45	Ro. N/A
20.10.2025	13:15-15:45	Ro. N/A
27.10.2025	13:15-15:45	Ro. N/A
03.11.2025	13:15-15:45	Ro. N/A
10.11.2025	13:15-15:45	Ro. N/A
17.11.2025	13:15-15:45	Ro. N/A
24.11.2025	**13:15-15:45**	**Ro. N/A**
01.12.2025	**13:15-15:45**	**Ro. N/A**

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

Materials

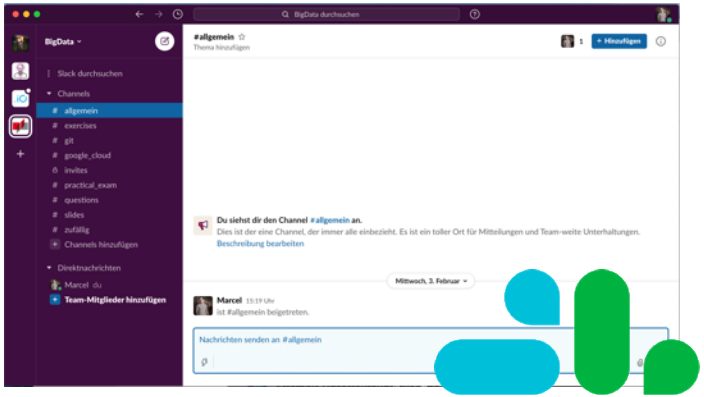
Lecture

… see calendar invite

Meet

Collaboration

https://join.slack.com/t/bigdatadhwb/shared_invite/zt-3etj0wkqo-vStGpo1HqDMg473od9d_xg

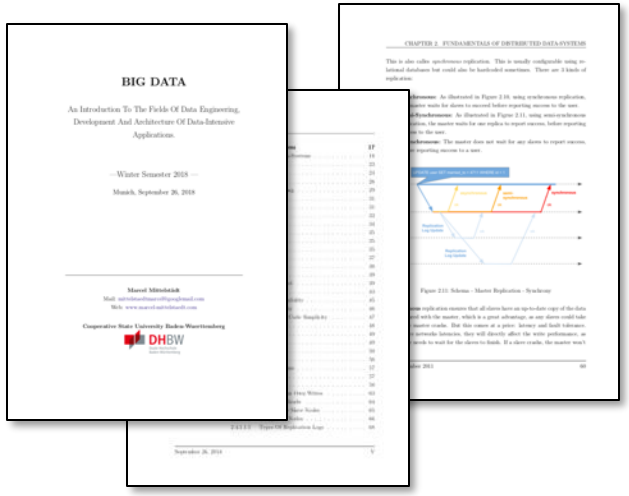


Materials

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

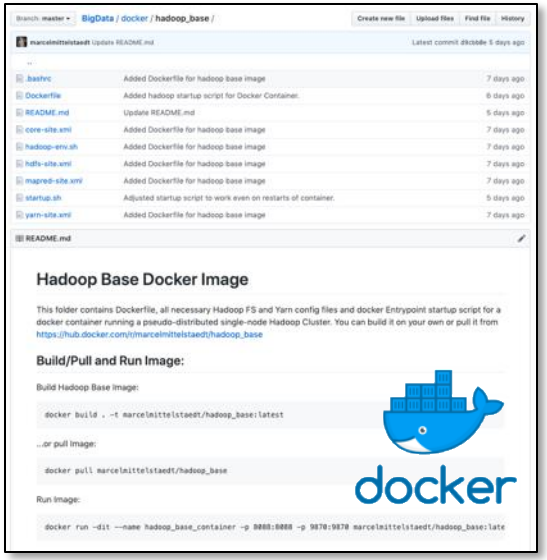
Solutions

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

Materials

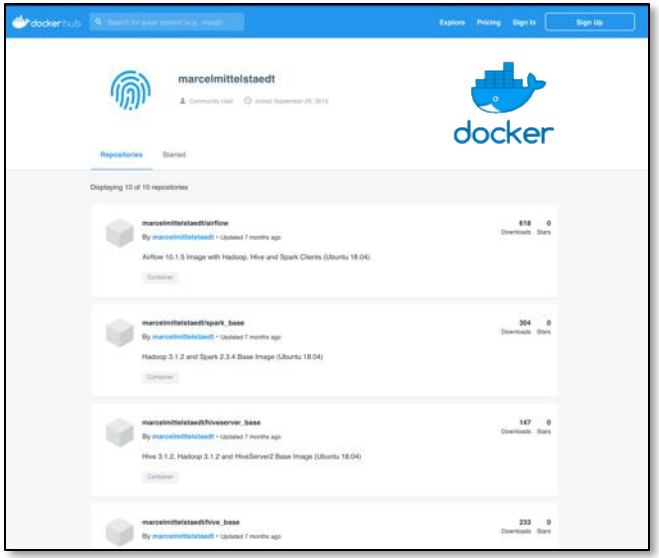
Docker Files

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

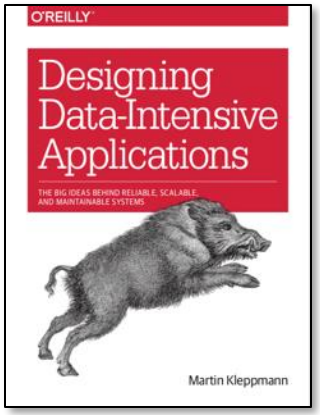


Dockerhub

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



Literature



Designing Data-Intensive Applications
Martin Kleppmann

Amazon Link

Packt›

Tech Ebooks from developer for developer

www.packtpub.com

APACHE Spark
databricks
YouTube

YouTube Channel of Developers of Spark

YouTube Link

Hands On Trainings

www.linuxhotel.de

Prerequisites

To efficiently participate:

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

For exam:

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

Feedback

Questions: anytime

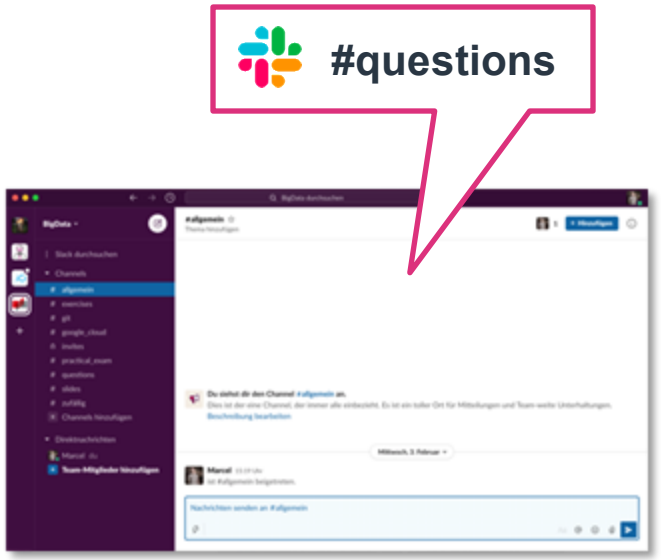
Feedback is welcome, via slack or mail

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

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

Please always post questions on public slack channels, so everybody can learn something ;)

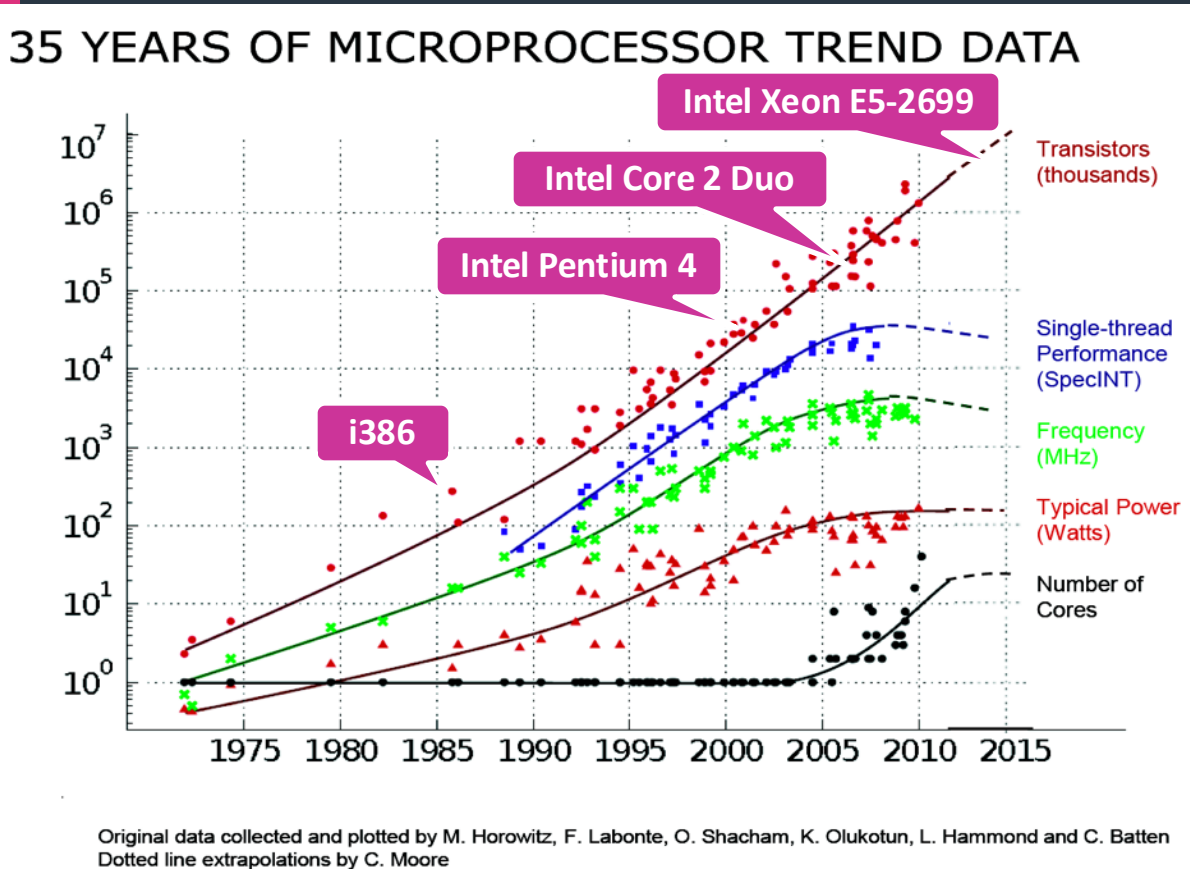
Questions



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:

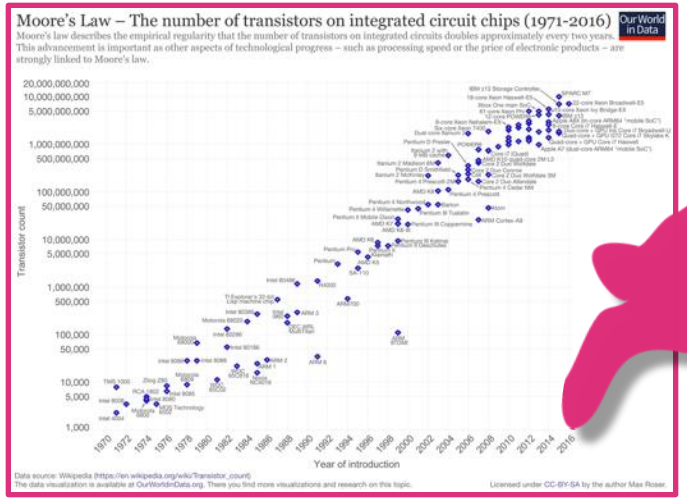
- **Back-In-Time**: → *scale vertical* – optimize code for a **single thread**
- **Today**: *scale horizontal* – run code **in parallel**

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**

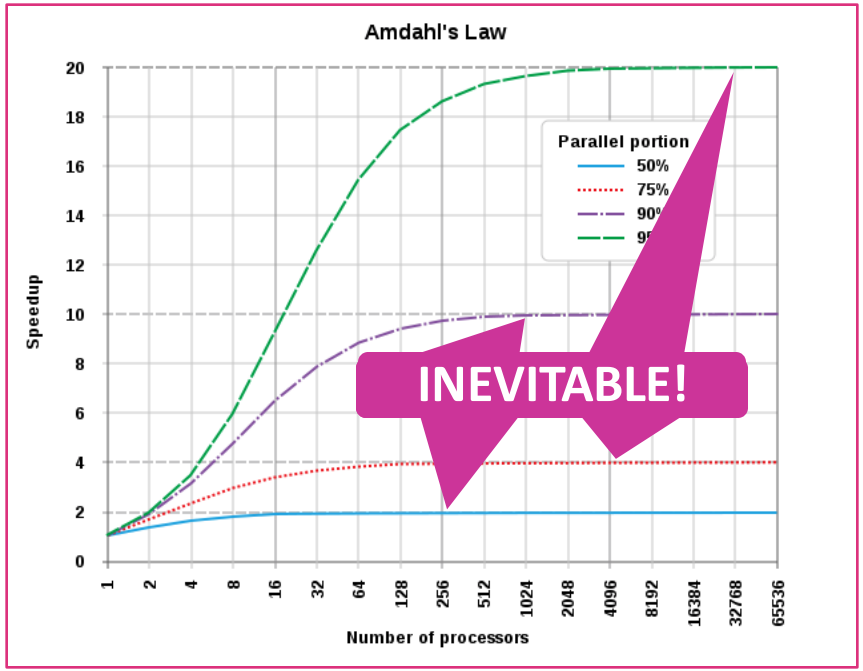


Moore's Law: *is the observation that the number of transistors in a dense integrated circuit doubles about every two years.*

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
Frontier	Oak Ridge National Laboratory (Tennessee, USA)	1.100.000,00	8,730,112 Processor: AMD Optimized 3rd Generation EPYC 64C 2GHz, 9 PB RAM	21.000 kW	Wissenschaftliche Anwendungen
Fugaku	RIKEN Center for Computational Science, Kobe, (Japan)	415.530,00	152.064 A64FX (48 Kerne, 2,2 GHz),4,85 PB RAM	15.000 kW	Wissenschaftliche Anwendungen
Summit	Oak Ridge National Laboratory (Tennessee, USA)	122.300,00 aufgerüstet auf 148.600,00	9.216 POWER9 CPUs (22 Kerne, 3,1 GHz), 27,648 Nvidia Tesla V100 GPUs	10.096 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[8]	Lawrence Livermore National Laboratory (Kalifornien, USA)	71.600,00	IBM Power9 (22 Kerne, 3,1 GHz) 1,5 PB RAM	7.438 kW	physikalische Berechnungen (z. B. Simulation von Kernwaffentests)
Tianhe-2[9]	National University for Defense Technology, Changsha, China finaler Standort: National Supercomputer Center (Guangzhou, Volksrepublik 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)
Hawk[10][11]	Höchstleistungsrechenzentrum Stuttgart (Deutschland)	26.000,00	11.264 AMD EPYC 7742(64 Kerne, 2,25 GHz), 1.44 PB RAM	3.500 kW	Wissenschaftliche und kommerzielle Anwendungen
Piz Daint	Swiss National Supercomputing Centre (CSCS) (Schweiz)	21.230,00	Cray XC50, Xeon E5-2690v3 12C 2,6 GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc. (361.760 Kerne)	2.384 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[12]	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

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

"Frontier",
Oak Ridge National Laboratory in Tennessee, USA

Racks: ~74++
Nodes: 9.472
Cores: ~8,7 Mio++
TeraFLOPS: 1.100.000,00
Energy: 21.000 kW

Frontier Supercomputer, Oak Ridge National Laboratory in Tennessee, USA

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***

Using ***spot instances*** *this could be less than*
~100$/hour

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

Google Cloud:

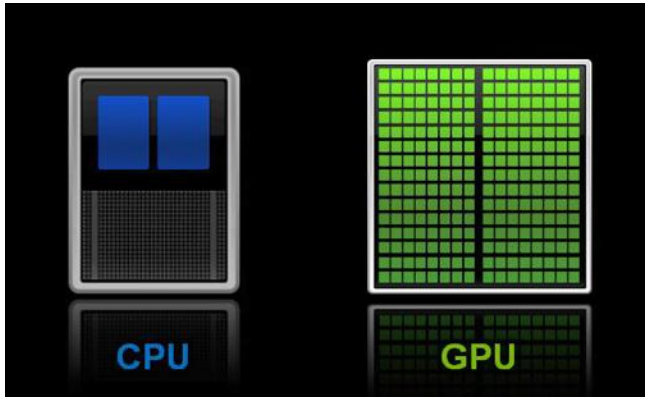
Instance: *n1.standard-64*
vCPUs *per Instance:* 64
RAM *per Instance:* 240
Costs per hour *per Instance:* *3,04 USD*

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

Using ***spot instances*** *this could be less than*
~100$/hour

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

Distributed Technologies – **GPU Computing**

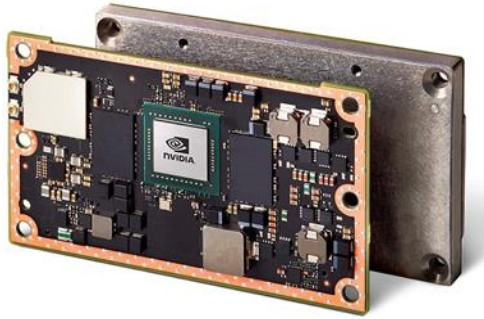


- advanced GPUs **initially developed for** (3D) **gaming**

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

Cores:	**2-18** *(e.g. 12 Cores for Intel Xeon E5-2690)*	**2.000-10.000** *(e.g. 10.496 CUDA for RTX 3090)*
Freq.:	**2-4 Ghz** *(e.g. 2,6-3,5 Ghz for Intel Xeon E5-2690)*	**1.000-2.000 MHz** *(e.g. 1.70MHz for RTX 3090)*

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

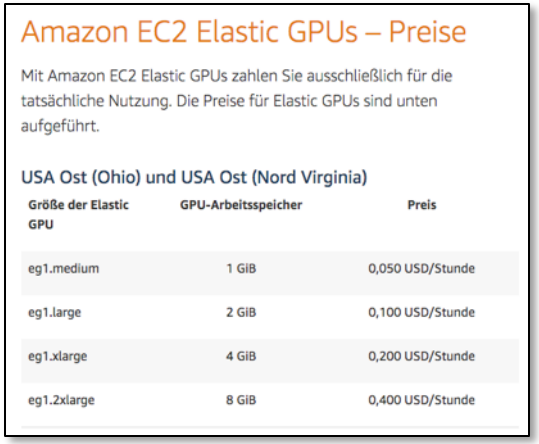
On-Premise:

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

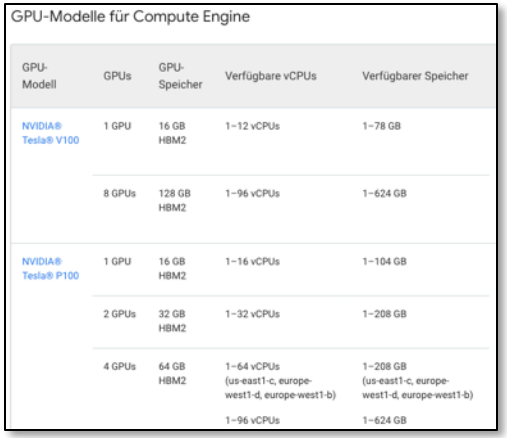
Mit Amazon EC2 Elastic GPUs zahlen Sie ausschließlich für die tatsächliche Nutzung. Die Preise für Elastic GPUs sind unten aufgeführt.

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 (Batch) Processing (Computing)

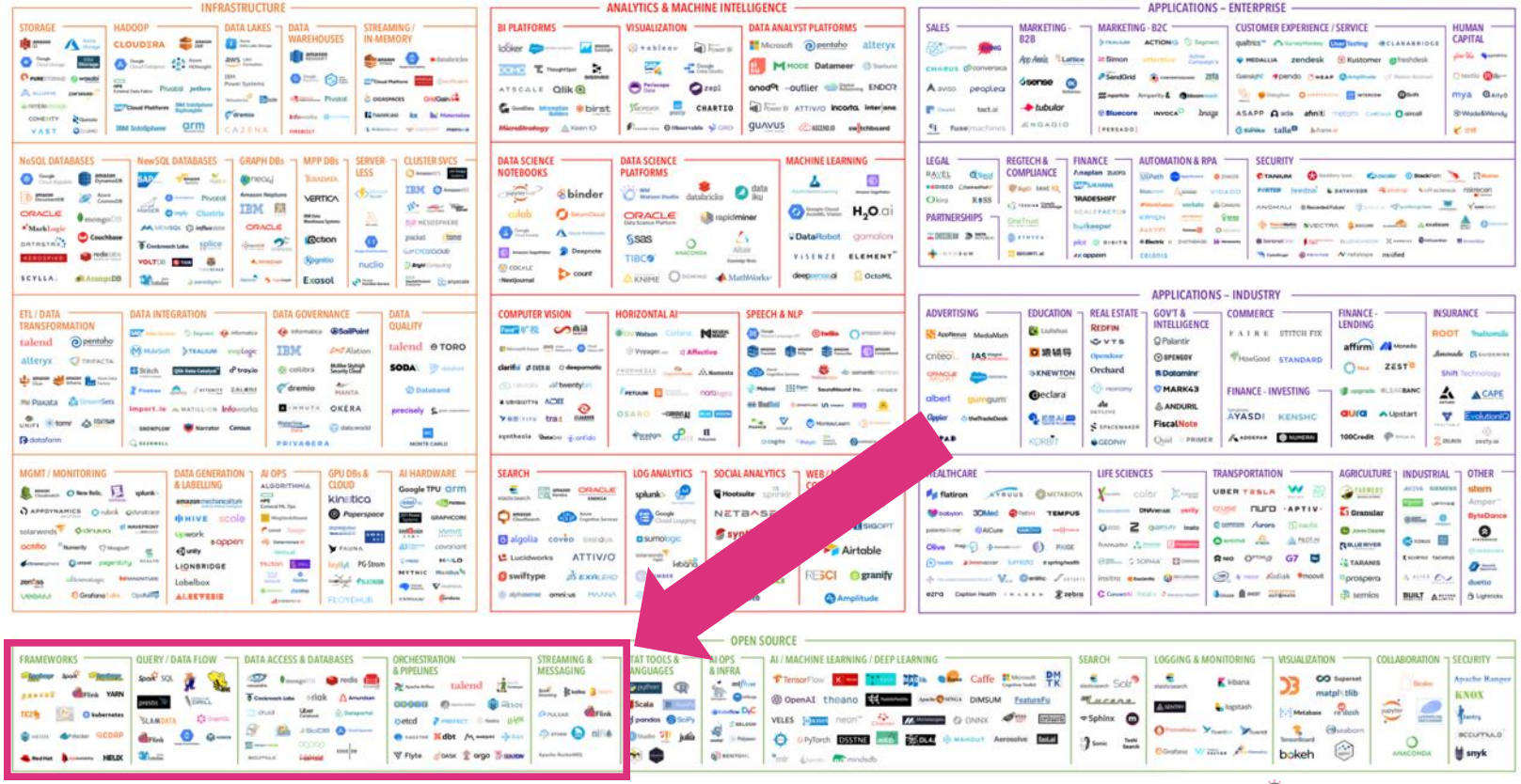
Distributed (Stream) Processing (Computing)

Distributed DataStorage/Database Engines

Distributed Cloud Data Storages

AWS S3 | Azure DataLake | Google Buckets

Distributed Technologies – Big Data Landscape



Pokemon or Big Data?

https://pixelastic.github.io/pokemonorbigdata/

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.

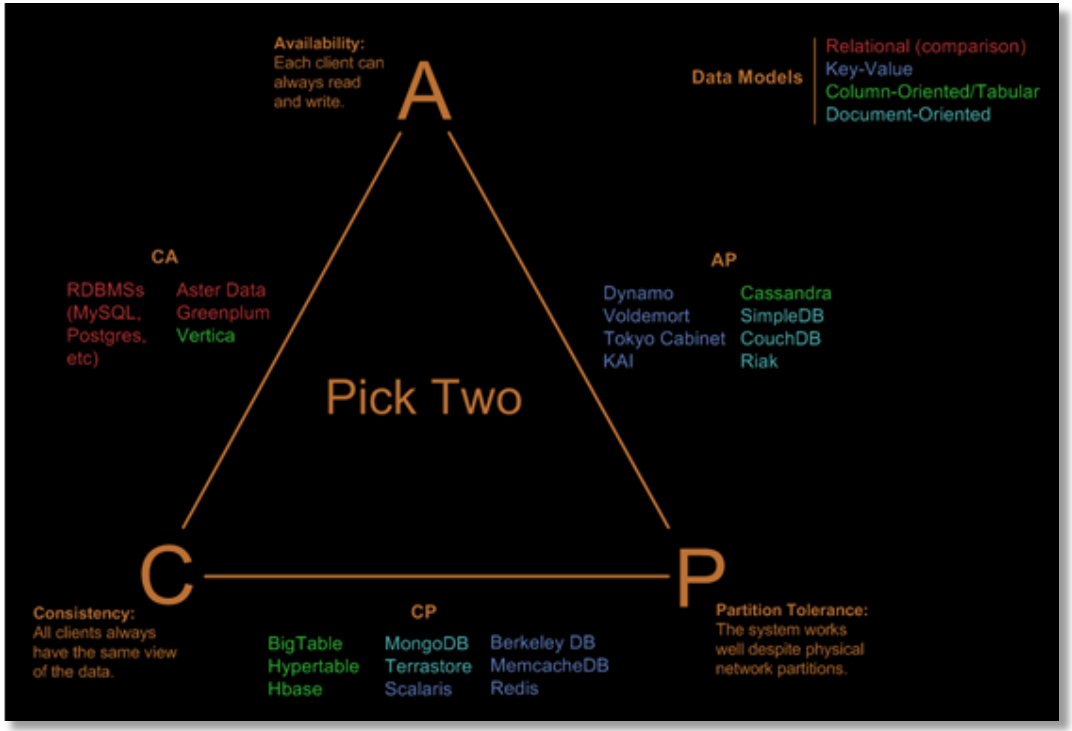
→ usually consistent within seconds/milliseconds

→ 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

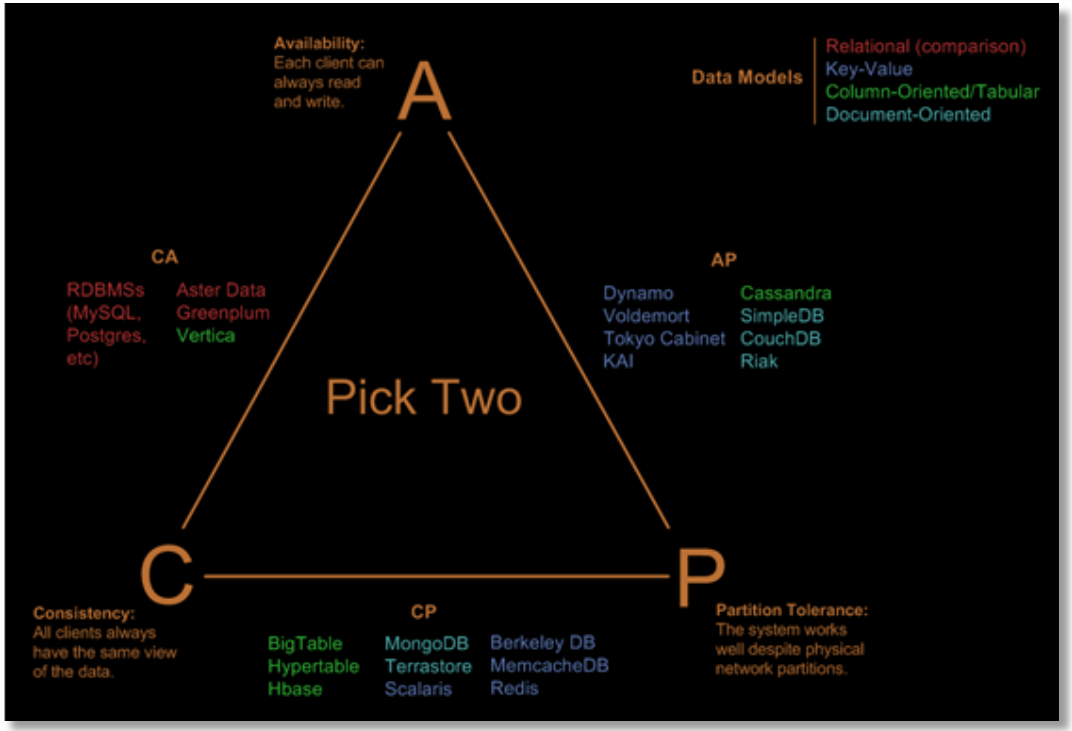
Big Data Challenges – **CAP** (Eric Brewer)



- **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**

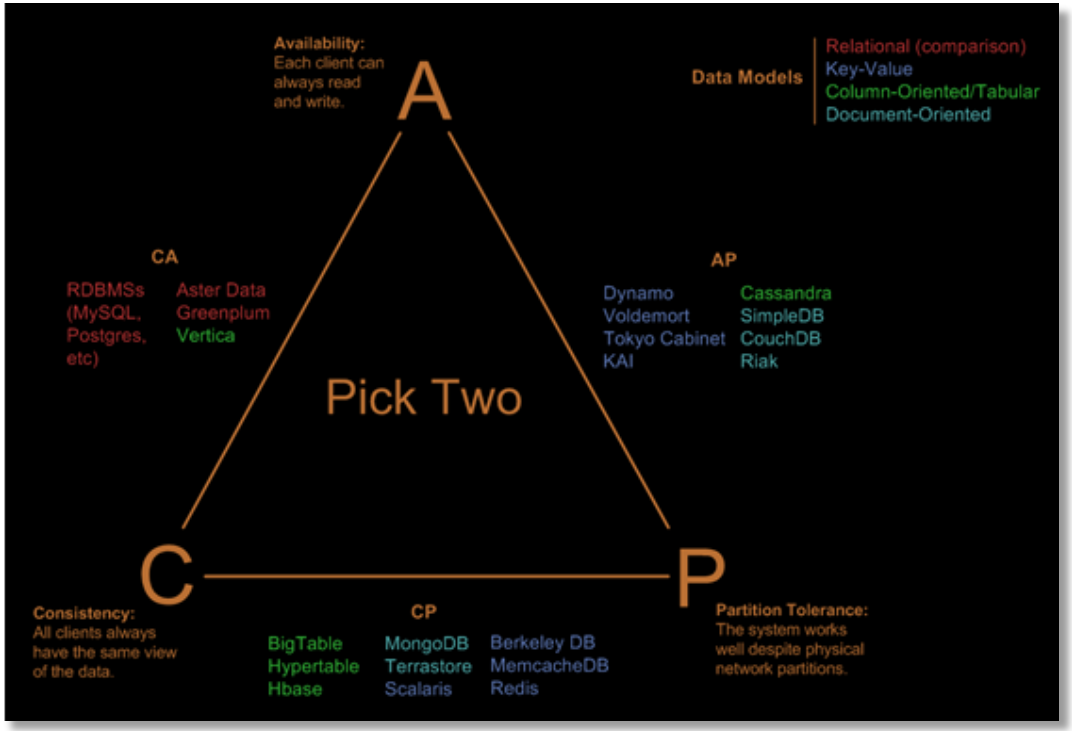
Big Data Challenges - CAP



- **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

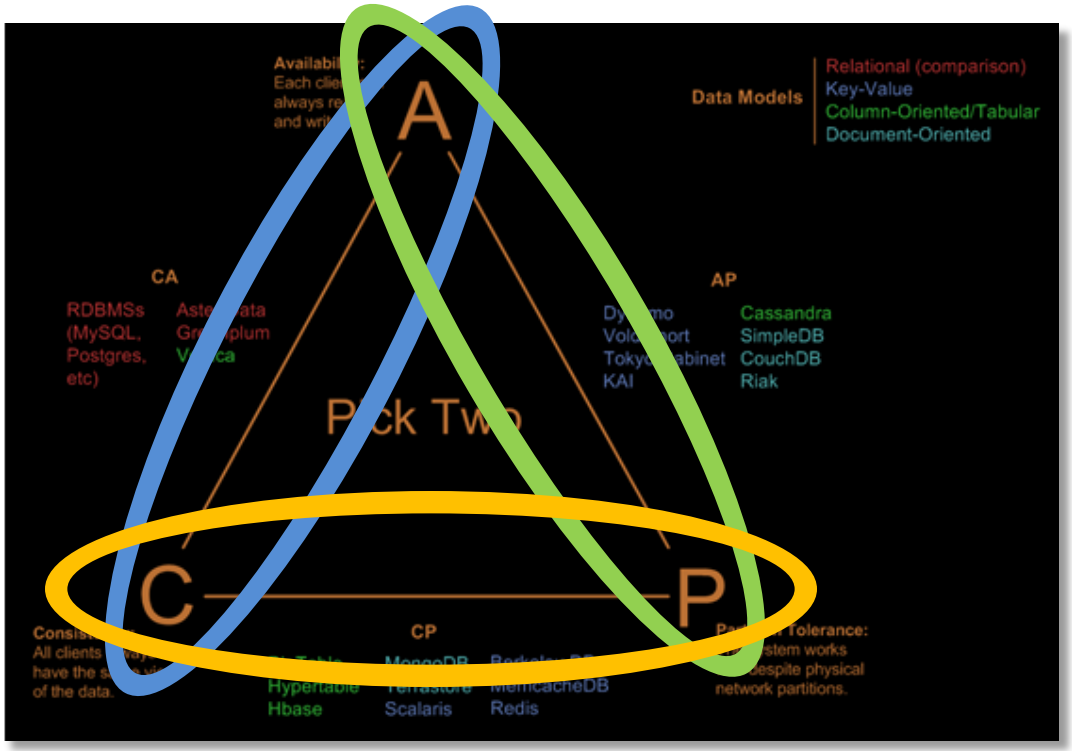
Big Data Challenges - CAP



- **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 consis tent 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/stream 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 Infra/Cloud Ops**

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

(Programming) Languages: Python, Bash, YAML (Terraform, CdK, Azure ARM, …)

Key-Skills:

– strong communication and collaboration skills
– broad knowledge of Ops Tools, best practices and automation
– frequently refreshs his knowledge about Ops tools continuously
- Infrastructure Deployment Automation (Kubernetes, Docker, Salt, Ansible, Terraform, Puppet, Chef, …)
– Continuos Integration (e.g. Jenkins, Gitlab CI, AWS CodeBuild, AWS CodePipeline, Azure Devops, …),

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, Scikit, Numpy, Scipy, Pandas, 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

50.000€ - 80.000€

Big Data Engineer

55.000€ - 90.000€

Data Scientist

60.000€ - 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-cli-darwin-arm.tar.gz?hl=de

# Untar
tar -xvzf google-cloud-cli-403.0.0-darwin-x86_64.tar.gz -C ~/

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

Windows:

```
https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe?hl=de
```


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] https://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/shar
e/keyrings/cloud.google.gpg add -

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


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/
4TgFAo7a2Nk1LmoATPPUORoagbVCwZpU4e3emnSlyUkwkcjbMw24Fzwv4mkiebSGe58AsberhYtCaoGKRpjCtEMtCG0zb
OxfC2Em4wYh9FQnLvKkdbEU1raPZ3aKIOpeDKjYLmp9r8D2akBvA3lSgNRS/0uh7YhLw4kmiCscTKlSEtMG/TNZEP2smW
fasG03LvMpC3vZ7w== hans.wurst@lehre.dhbw-stuttgart.de
```

3. Upload your SSH 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,tcp:9000,tcp:9090,tcp:9091,
tcp:9092,tcp:8080 \
             --description remote_access \
             --direction INGRESS \
             --network default \
             --source-ranges 141.31.0.0/17
```

Your IP (-Range)

- source_ranges needs to be your public IP to limit accessibility of your future VM instances to you only:
  - **141.31.0.0/17** should be IP Range of DHBW Network
  - to get **your current public IP**: http://ifconfig.me/ or:

```
curl ifconfig.me
141.31.42.37
```


Spawn VM instance

1. Spawn VM Instance with 26GB RAM, 4 Cores and Ubuntu 20.04 LTS:

Put your Google project id here

```
gcloud compute --project=[your-project-id] instances create big-data \
           --zone=europe-west3-a \
           --machine-type=n2-highmem-4 \
           --subnet=default --network-tier=PREMIUM \
           --maintenance-policy=MIGRATE \
           --image=ubuntu-2004-focal-v20210129 \
           --image-project=ubuntu-os-cloud \
           --boot-disk-size=100GB \
           --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-a  n2-highmem-4                10.156.1.3   33.89.216.84   RUNNING
```

Spawn VM instance

3. Connect To Your VM instance using SSH

Put your previously configured SSH user here (Slide 69)

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

Put EXTERNAL IP (from previous slide) here

```
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1036-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Feb  7 17:39:54 UTC 2021

  System load:  0.16              Processes:             140
  Usage of /:   3.6% of 38.60GB   Users logged in:       0
  Memory usage: 1%                IPv4 address for ens4: 10.156.0.2
  Swap usage:   0%

1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Manage VM instances

1. Stop instance

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

2. Start instance

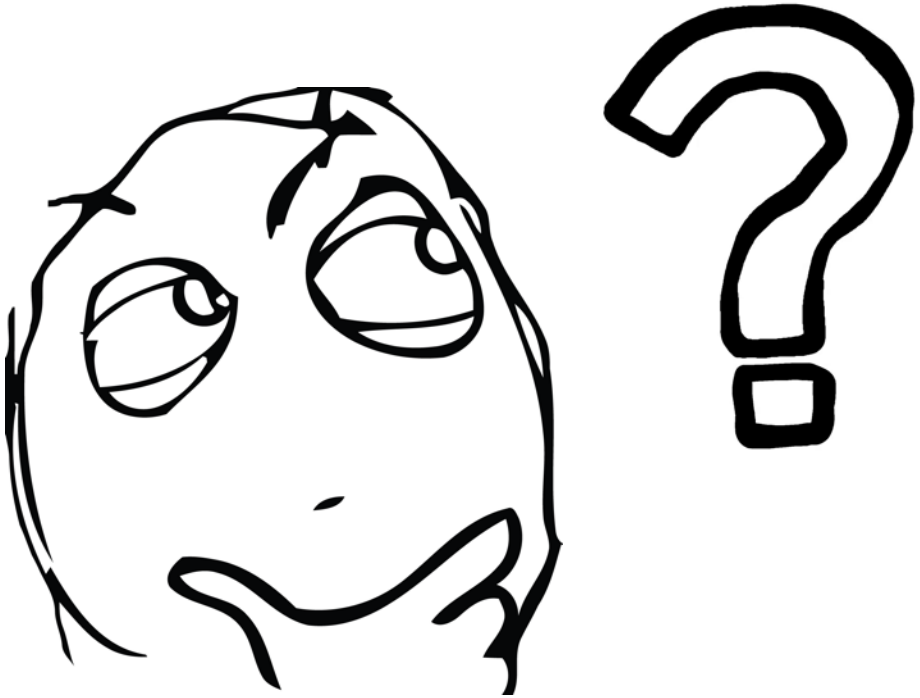
```
gcloud compute instances start big-data
```

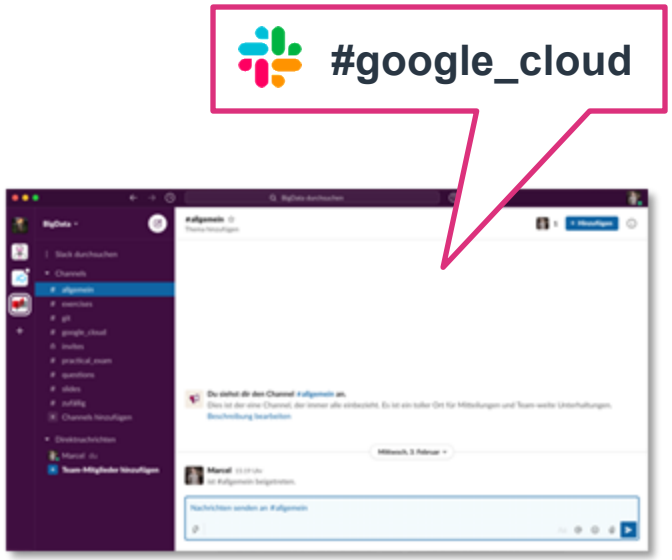
!!! ALWAYS REMEMBER TO **STOP** YOUR **INSTANCE** when **not used**,
to do not waste gcloud credits!

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  n2-standard-4               10.156.1.3                TERMINATED
```


Questions





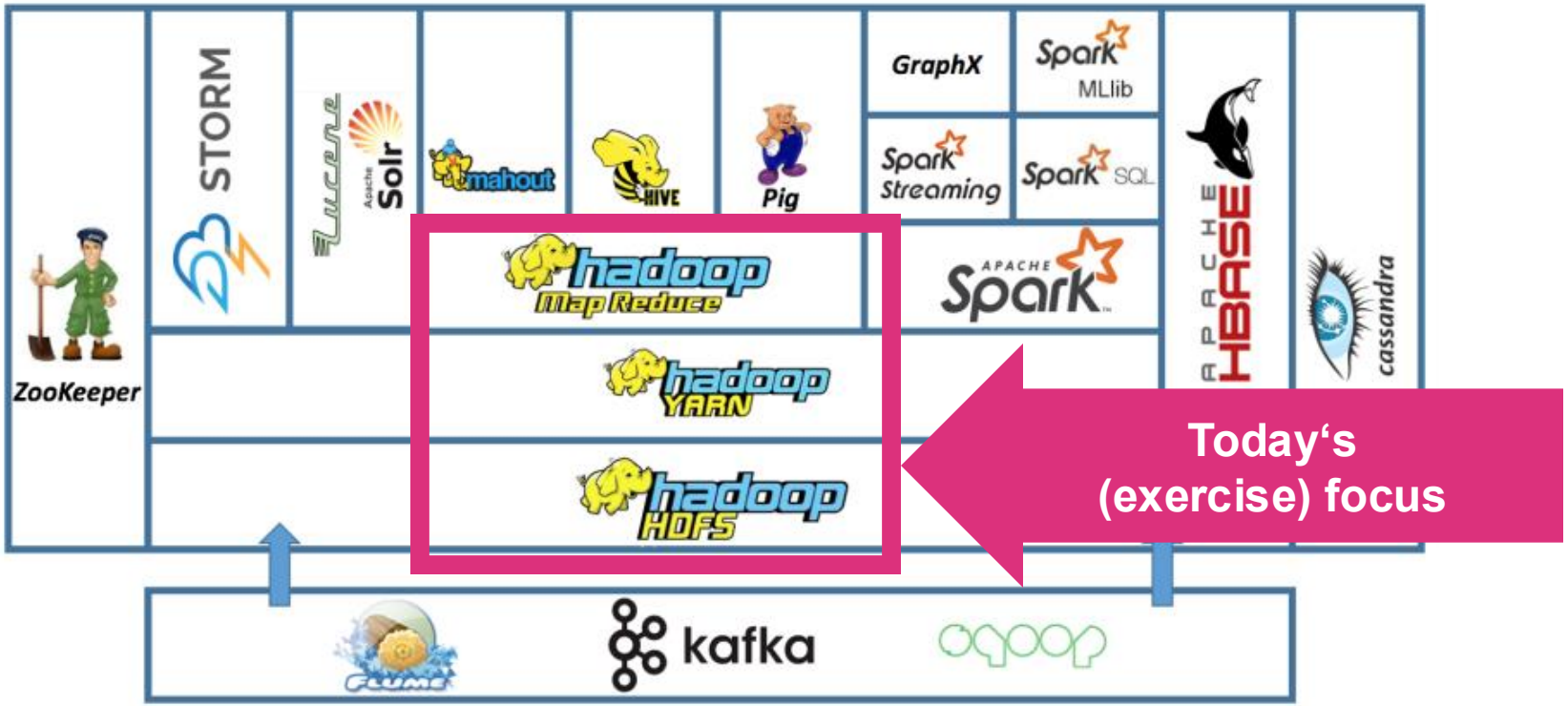
Break

TIME FOR A BREAK

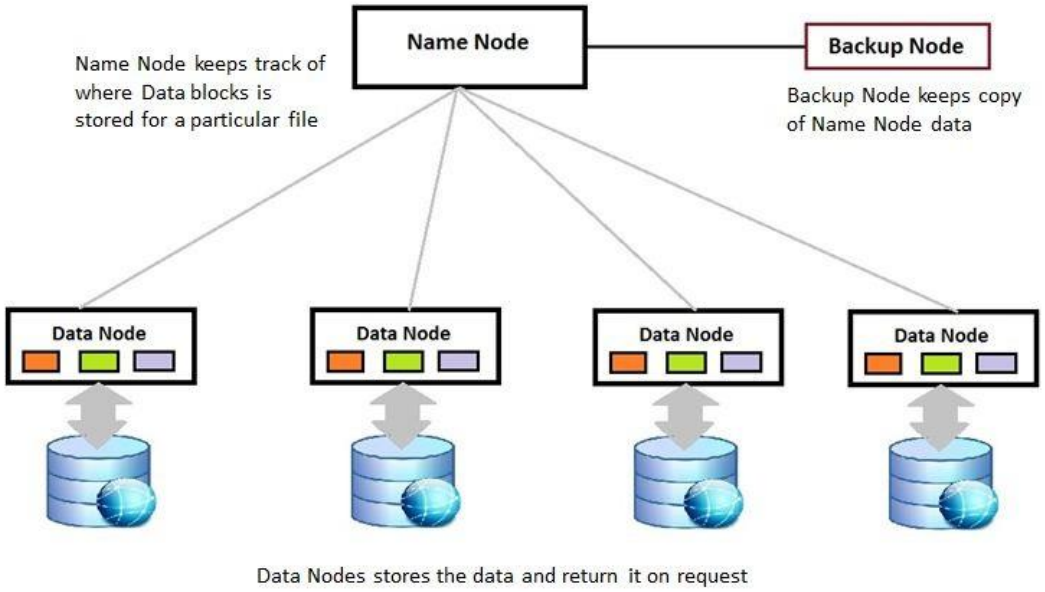
HandsOn - Hadoop

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**

Cloud-based distributed Filesystems

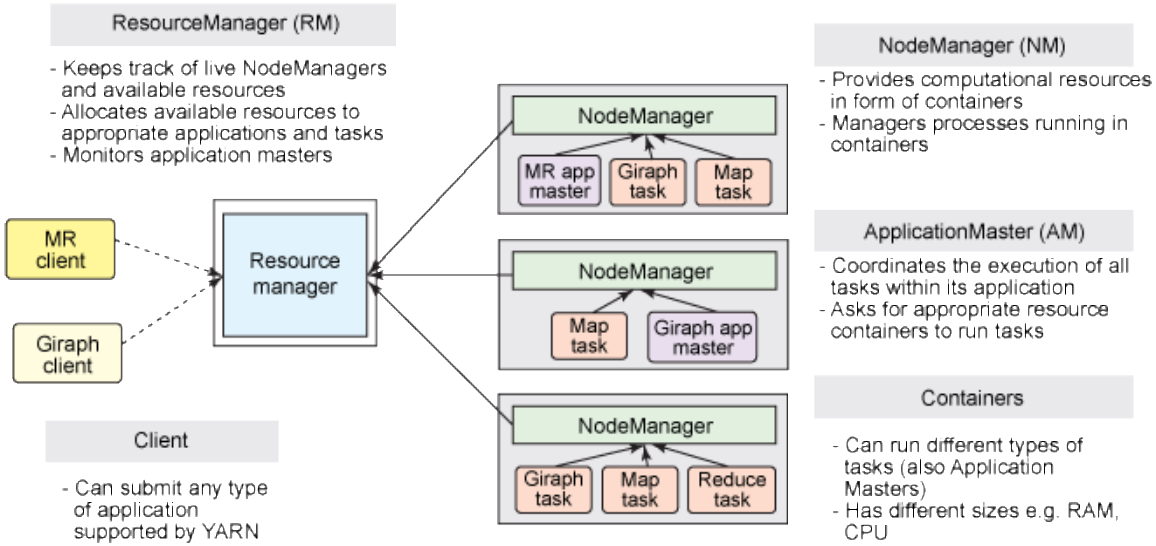
Managed Clusters	Managed Filesystems
 AWS EMR	 AWS S3
 Databricks	 Azure DataLake
 Dataproc	 Google Buckets

- ***HDFS** is usually used when using on-premise Hadoop Clusters*
- *In the Cloud you can also run and manage your own Hadoop cluster manually e.g. with Kubernetes*
- *But lot of people use **managed clusters** like:*
  - *AWS EMR,*
  - *Databricks on Azure/AWS/GCloud*
  - *Google DataProc*
  - *…*
- *and then use cloud based **managed** distributed **filesystems** **instead of HDFS**, e.g.:*
  - *AWS S3 Bucket Storage*
  - *Microsoft Azure Data Lake Gen2 Storage*
  - *Google Cloud Buckets*

YARN

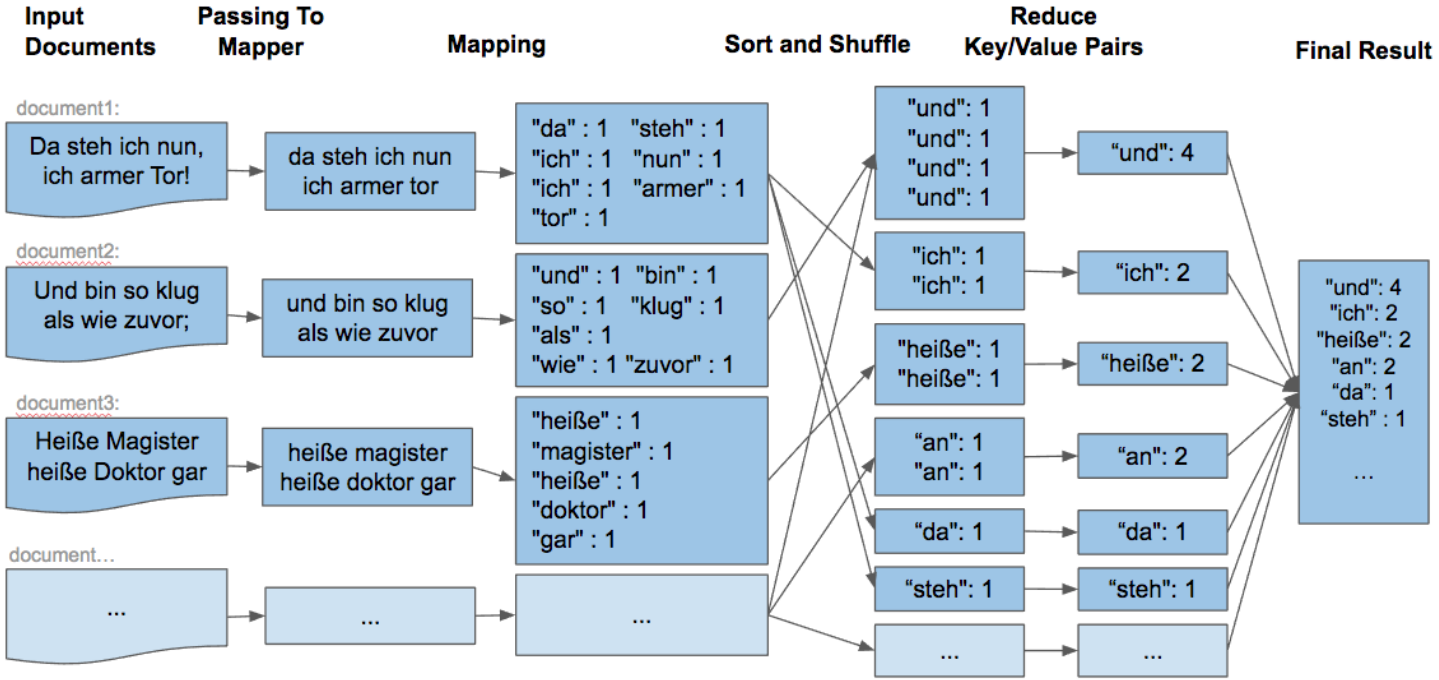
- ***Y**et **a**nother **R**essource **N**egotiator*

- 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 and Yarn manually (standalone)

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_292"
OpenJDK Runtime Environment (build 1.8.0_292-8u292-b10-0ubuntu1~20.04-b10)
OpenJDK 64-Bit Server VM (build 25.292-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 --disabled-password  --gecos "" 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://archive.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
export PDSH_RCMD_TYPE=ssh
```

```
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>
        <property>
                <name>yarn.nodemanager.resource.memory-mb</name>
                <value>16384</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

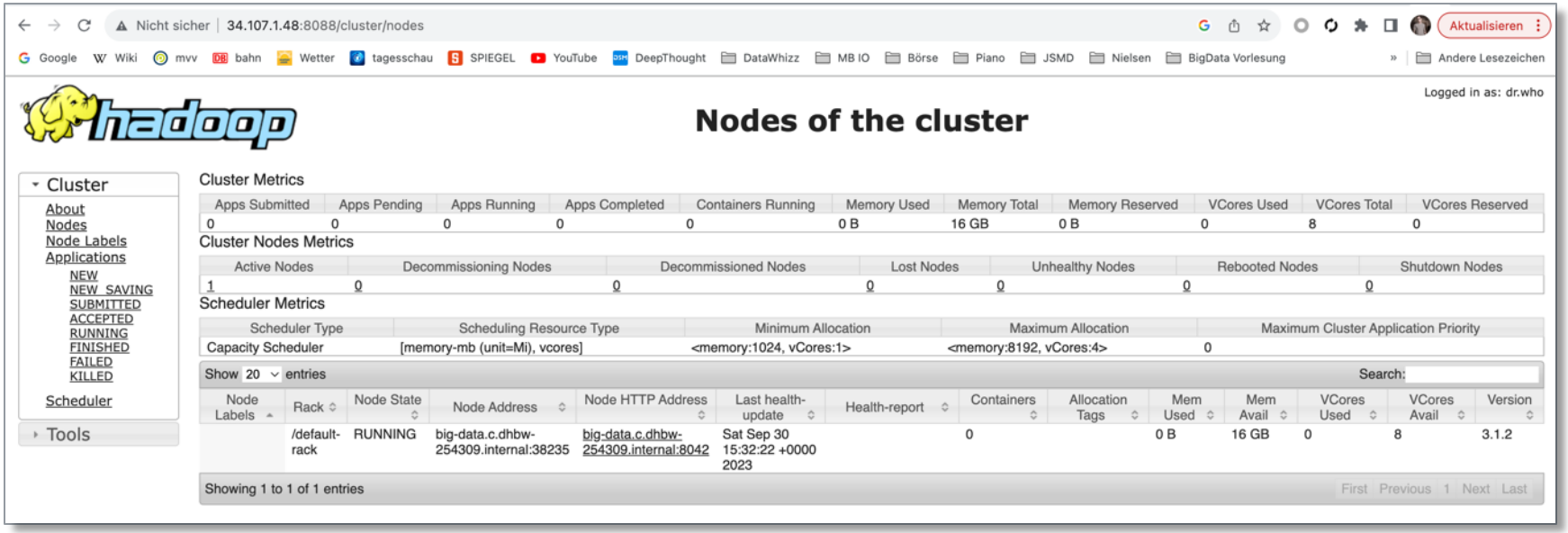
```
hdfs dfsadmin -report
```

```
Configured Capacity: 103880232960 (96.75 GB)
Present Capacity: 100415668224 (93.52 GB)
DFS Remaining: 100415643648 (93.52 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-254309.internal
Decommission Status : Normal
Configured Capacity: 103880232960 (96.75 GB)
DFS Used: 24576 (24 KB)
Non DFS Used: 3447787520 (3.21 GB)
DFS Remaining: 100415643648 (93.52 GB)
DFS Used%: 0.00%
DFS Remaining%: 96.66%
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: Sun Sep 26 16:33:43 UTC 2021
Last Block Report: Sun Sep 26 16:33:10 UTC 2021
Num of Blocks: 0
```

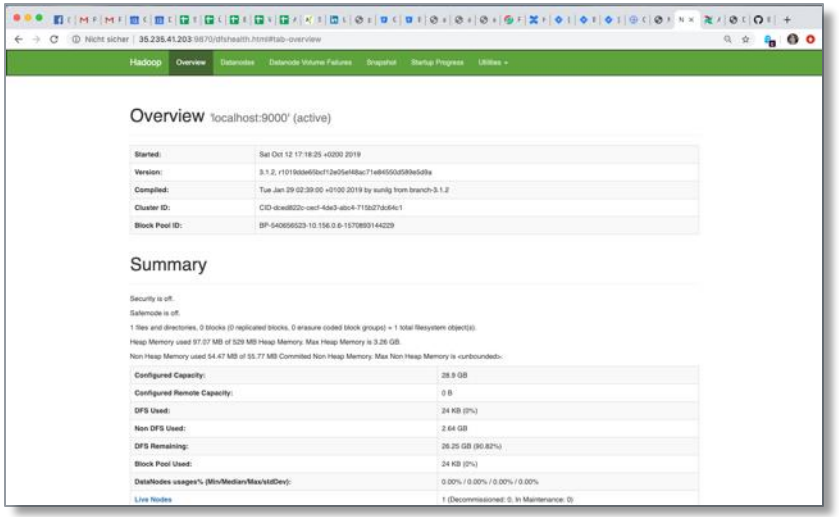
Check Hadoop/HDFS

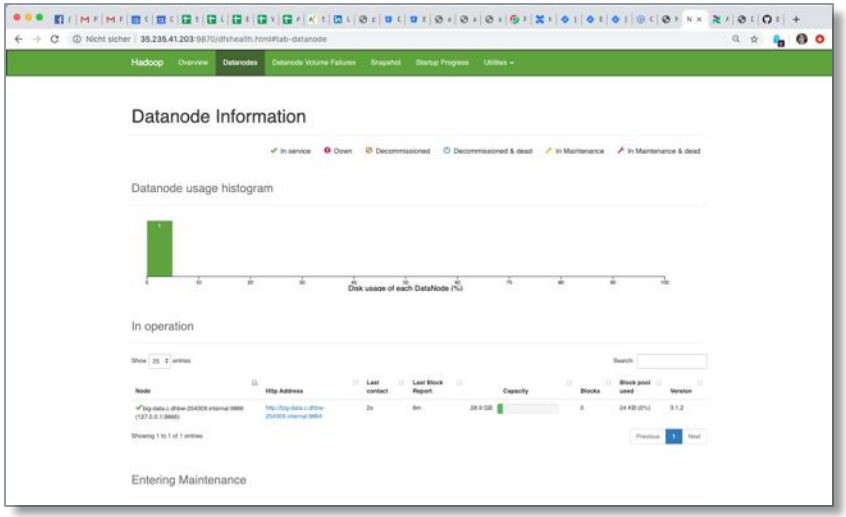
11. Check Ressource Manager Landing Page (http://XXX.XXX.XXX.XXX:8088/cluster):



Check Hadoop/HDFS

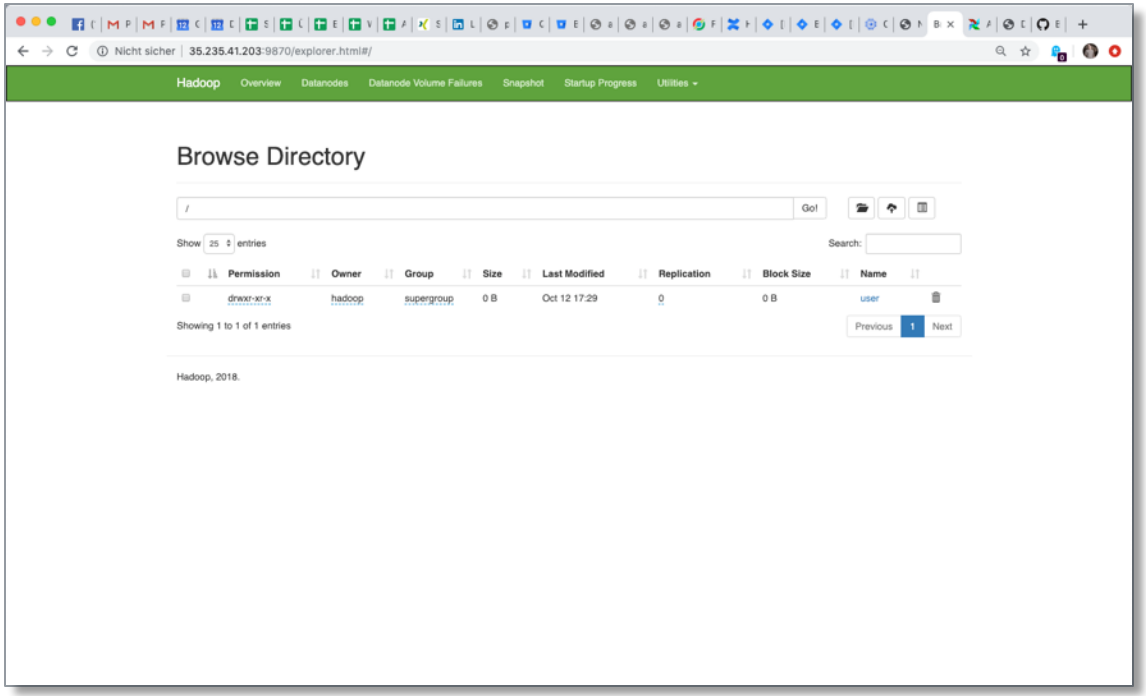
12. Check NameNode Landing and Status Page (http://XXX.XXX.XXX.XXX:9870*):*





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 2021-09-26 16:36 /user
hadoop@big-data:~$
```


Working with HDFS

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

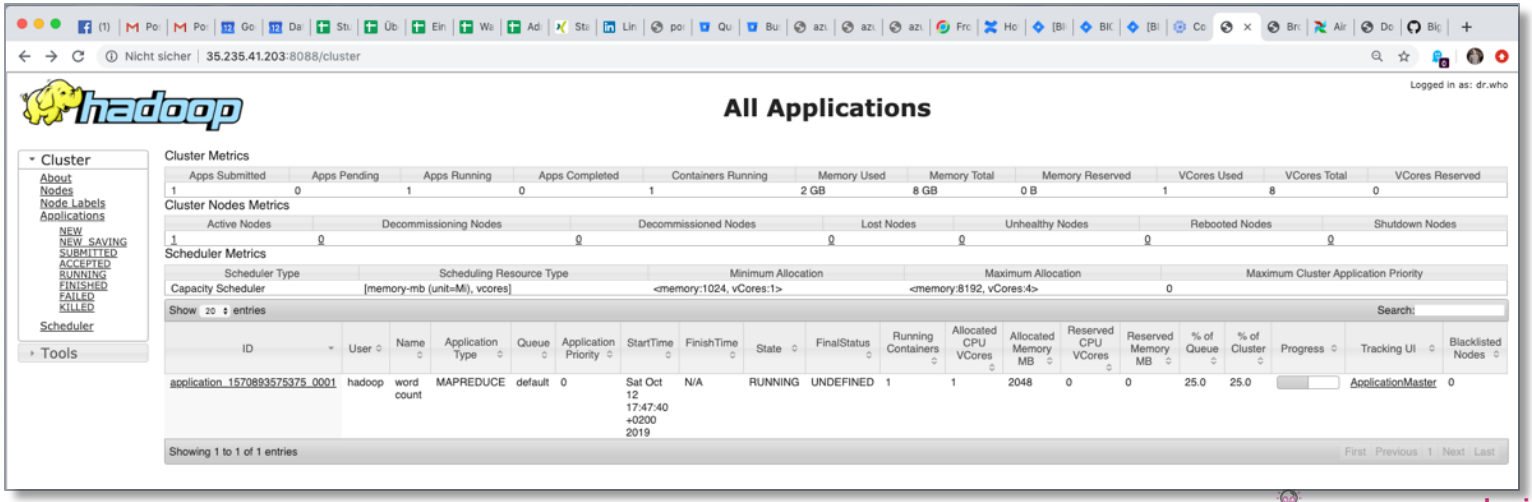
```
hadoop fs -put /var/log/dpkg.log /user/hadoop/dpkg.log
```


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:



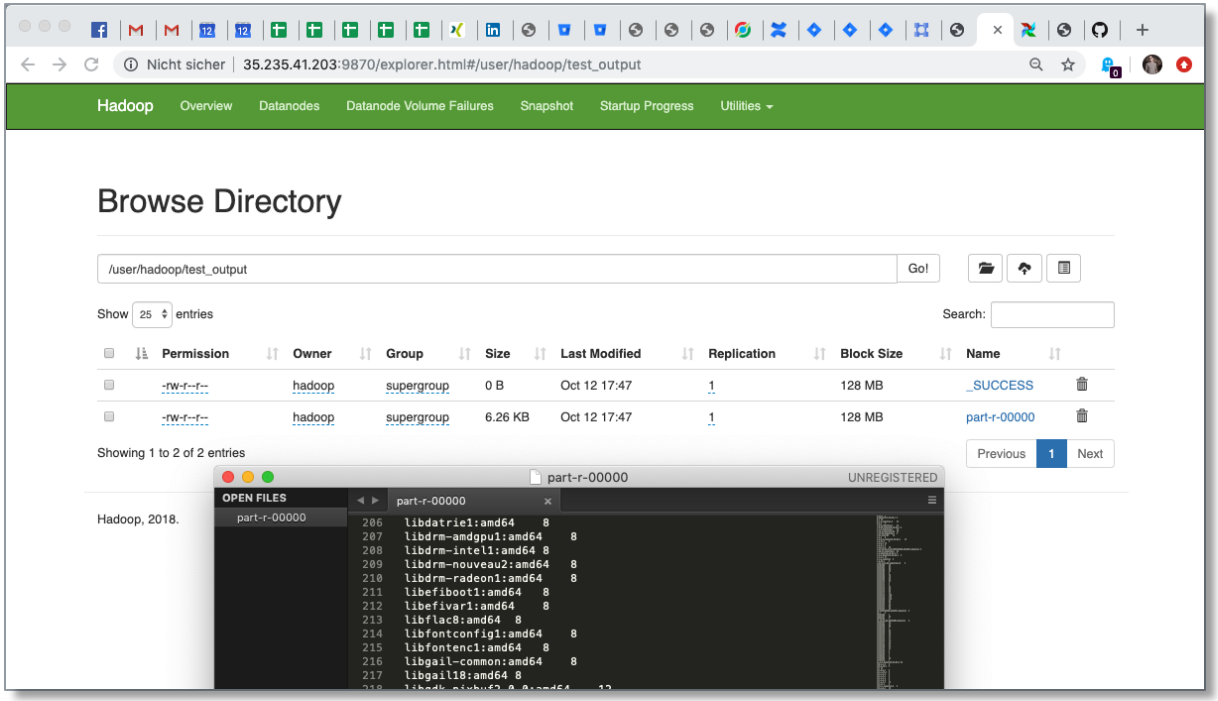
Run Example MapReduce Job

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_2025-2025/01_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.

Stop Your VM Instance

DON'T FORGET TO STOP YOUR VM INSTANCE!

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

Well Done

WE'RE DONE FOR ...TODAY