Geo874 | HS24
Universität Zürich

6. Lecture Introduction to Databases

Advances in databases: NoSQL databases

Esra Suel
Dept. of Geography, University of Zürich

Rolf Meile
Eidg. Forschungsanstalt für Wald, Schnee und Landschaft

kudos to Dr. Zhiyong Zhou, Dr. Cheng Fu, & Dr. Haosheng Huang for providing this document

Next week – exam!

- About 60 mins
- Stuff covered in lectures and practicals
- Not allowed to take lecture notes
- Pencil and paper
- **NO remote exam option**

- Friday 31.10.2025 **8:40 am - 9:40 am** in **Y25 - H79**

- Be here at 8:30 am!
- No lab on 31.10.2025

- Good Luck!

Summary 5.1

Correspondence between ER- and relational models

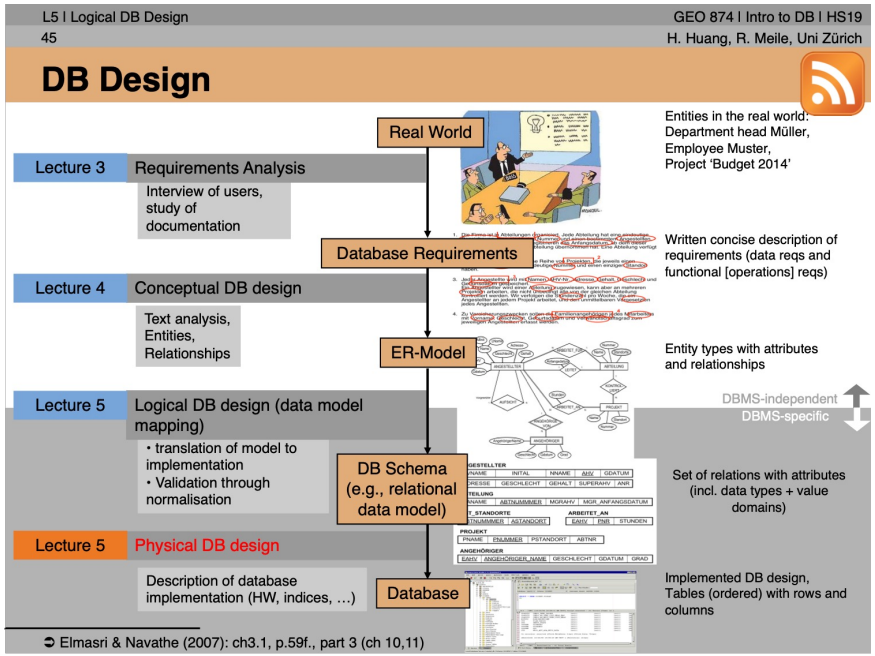
ER model	Step	Relational model
Entity type	1,2	Relation
Binary 1:1 relationship type	3	Add PK on the total participation side to the other side
Binary 1:N relationship type	4	Add PK on the 1-side to the N-side
Binary M:N relationship type	5	Relation and two FKs
n-ary relationship type	7	Relation with n FKs
Simple attribute	1, ...	Attribute
Composite attribute	1, ...	Set of simple attributes
Multivalued Attribute	6	Relation and FK

➲ Elmasri & Navathe (2014): Ch 9, pp. 287-296.

So far we covered...

Relational databases

- DB Design: Requirements -> conceptual -> logical -> physical
- Entities and relationships
- Relational model
- SQL – the language of relational databases



So far we covered...

Relational databases

- DB design process: requirements -> conceptual -> logical -> physical design
- Basic assumptions:
  - Requirements: Use-cases can be defined and data can be structured at start
  - Conceptual: modelling entity types through fixed number of attributes, and their relationships
  - Logical: relational model, dense, row-based (tuple) data structures, row-first access, and SQL as access language
  - Physical: designed for a single server, ACID (atomicity, consistency, isolation, durability) transactions.

So – what if some of above do not hold...

Learning objectives

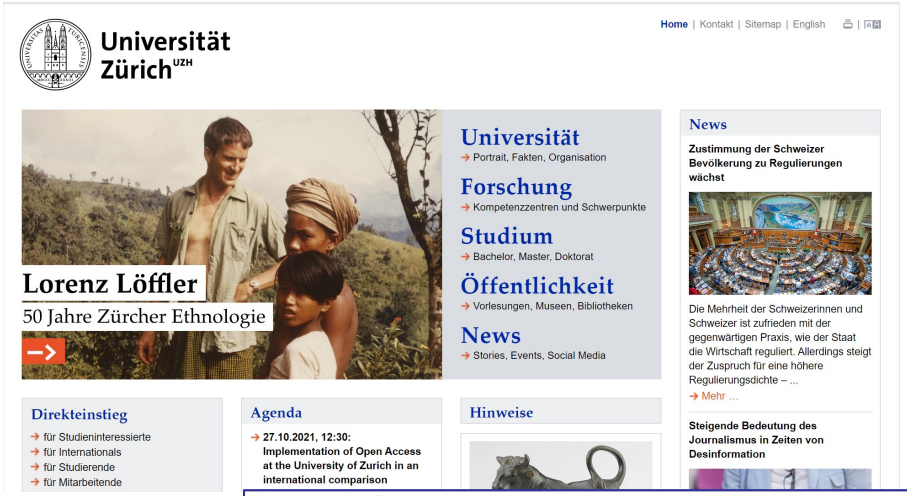
- ✓ You will understand current developments in data collection, storage and processing – Why NoSQL?
- ✓ You will understand the basic characteristics of NoSQL databases
- ✓ You will be introduced to Hadoop and Spark as frameworks for distributed data storage and processing — from batch processing to faster, in-memory computation.

The shift to digital economy

- Economy powered by the Internet and other 21st century technologies – the cloud, mobile, social media and big data
- Key to every digital economy business: Web/mobile-based applications need to
  - Support large numbers of concurrent users (tens of thousands, millions)
  - Deliver highly responsive experiences to a globally distributed base of users
  - Be always available – no downtime
  - Handle semi- and unstructured data
  - Rapidly adapt to changing requirements with new features and frequent updates
    - Relational databases: structures and data types are fixed in advance
    - NoSQL: Applications can add new fields on the fly.

Relational databases are unable to meet these new requirements!

Webpage as semi-structured data




```
<!DOCTYPE html>

<html xml:lang="de" lang="de" class="mod mod-layout skin-layout-content skin-layout-related">

<!--
 - UZH 1.9.2 (2021-08-24_13-57-25, 41, master, 1c1156425c819bc0ac00f2b3ba11b0efe5b8888d)
-->

<head>
    <!--[if IE]>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=IE7"/>
    <![endif]-->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>


<meta content="width=device-width, initial-scale=1.0" name="viewport" id="meta-viewport">
<script>
    if (screen && (screen.width >= 950 || screen.height >= 950)) {
        // force devices as ipad etc. to scale the viewport
        document.getElementById("meta-viewport").setAttribute("content", "width=1024,initial-scale=1,user-scalable=yes");
    }
</script>

<link rel="schema.dc" href="http://purl.org/dc/elements/1.1/" />
<link rel="schema.dcterms" href="http://purl.org/dc/terms/" />

<meta name="keywords" content="Universität Zürich, UZH, Universität, Hochschule, Zürich, Studium, Studieren, Forschung, Wi
<meta name="description" content="Die Universität Zürich gehört zu den besten Forschungsuniversitäten Europas und bietet d

<meta property="og:type" content="website"/>
<meta property="og:title" content=""/>
<meta property="og:url" content="http://www.uzh.ch/de.html"/>
```

A webpage has many tag-embraced contents. Each tag can have its own nested tags and contents… Like a tree

There are also links from one webpage to many other webpages.

Dataset sizes

OpenStreetMap

- 110 GB XML file / 60 GB planet PBF file (Binary format incl. compression)
- https://planet.openstreetmap.org/

Strava Heatmap (in 2017)

- 700 million activities
- 1.4 trillion GPS location fixes
- 7.7 trillion pixel
- 5 TB input data
- 100'000 years of total activity duration

"From 2015 to 2017, there were no updates to the Global Heatmap due to two engineering challenges:

- Our previous heatmap code was written in low-level C code and was only designed to be run on a single machine. It would have taken **months for us to update the heatmap** with this restriction.
- Accessing stream data required one S3 get request per activity, so reading the input data would have **cost thousands of dollars** and been challenging to orchestrate.

The heatmap generation code has been fully rewritten using Apache Spark and Scala. The new code leverages new infrastructure enabling bulk activity stream access and is **parallelized at every step from input to output**. […] The full global heatmap was built **across several hundred machines in just a few hours**, with a total compute cost of only **a few hundred dollars**. Going forward, these improvements will enable updating the heatmap on a regular basis."

https://medium.com/strava-engineering/the-global-heatmap-now-6x-hotter-23fc01d301de

Big Data – V V V

Buzzword, but...

- **Volume:** large amounts of data.
  - Large: beyond what can reasonably fit into a single memory/ hard drive
- **Velocity:** data arriving at a high rate and need to be stored and possibly processed fast
  - E.g., streaming data from sensors, from social network feeds, at the stock exchange, created on the Web...
  - Delays are costly
- **Variety/Variability:** data change their schema, formats, are inconsistent, and are of different types...
  - Need approaches that allow for easy adaptation in what is stored and how it is processed
- ... Some talk also of Veracity (how trustful the data can be, data quality)

Early talk on Big Data (coining the term): http://www.quora.com/Who-coined-the-term-big-data (Roger Magoulas, 2009)

Scalability

- To solve **Volume** and **Velocity** we need **scalable solutions: systems that can grow at least proportionally to the needs.**
- Purpose-built computers are $$$ (scale up, vertical scalability)
- What if the data outgrow it?

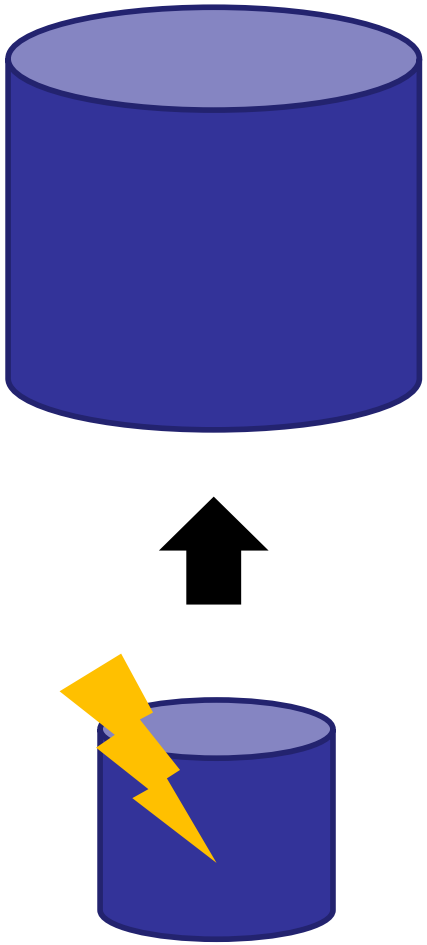
Relational databases are poor in supporting these!

Scale-up / vertical scaling

Increase resources available to system

- Add more and faster processors / CPU
- More memory / RAM
- More storage / SSD / HDD
- Specialized hardware (e.g., GPUs, ASICs)
- …

Powerful specialized hardware is very expensive!



This was your big data …

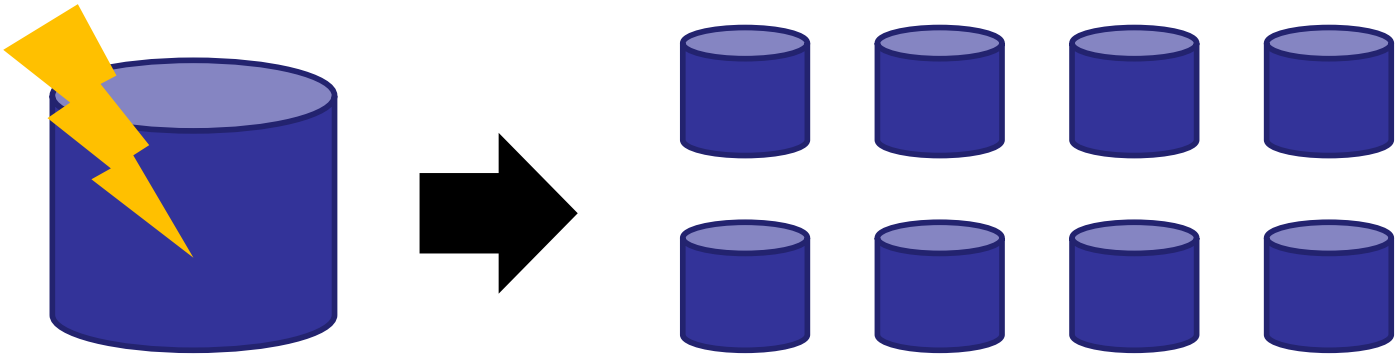
- IBM mainframe: Z server

Scale-out / horizontal scaling

Add more affordable less-specialized machines that connect to each other over a network - (**commodity) clusters**

Distribute a problem onto multiple distributed resources.

Distributed computing is complex!



Scalability

- Ability to add more small/cheap resources as requirements grow (scale out, horizontal scalability);
- Allowing partial failure by providing redundancy
- Consequence: databases are **partitioned** between multiple physical systems (computers).

Relational databases are poor in supporting these!

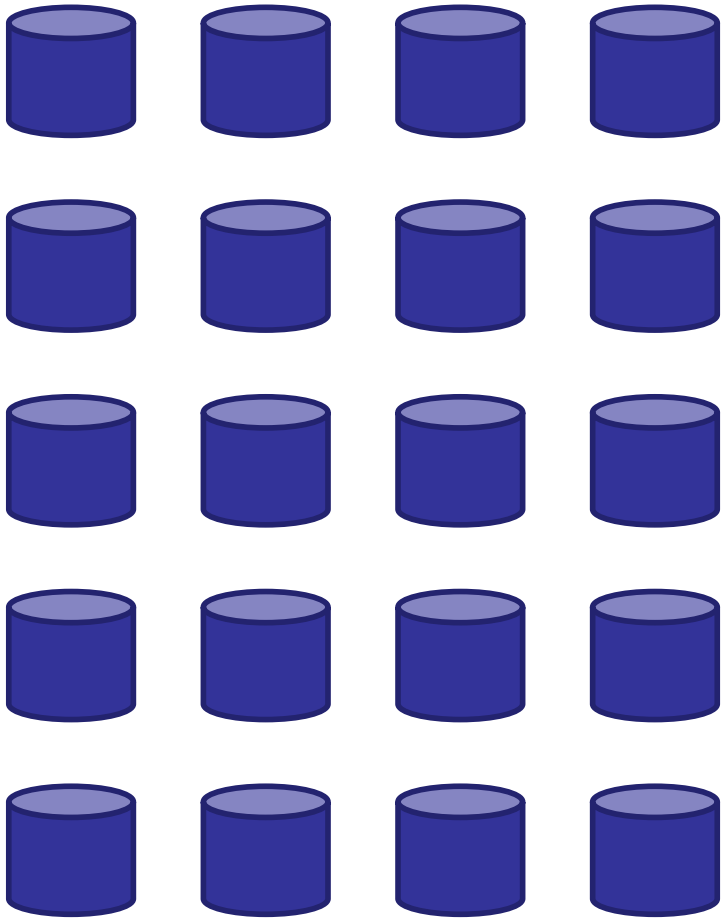
Challenges scale-out

Fault tolerance

Nodes can fail at any moment!

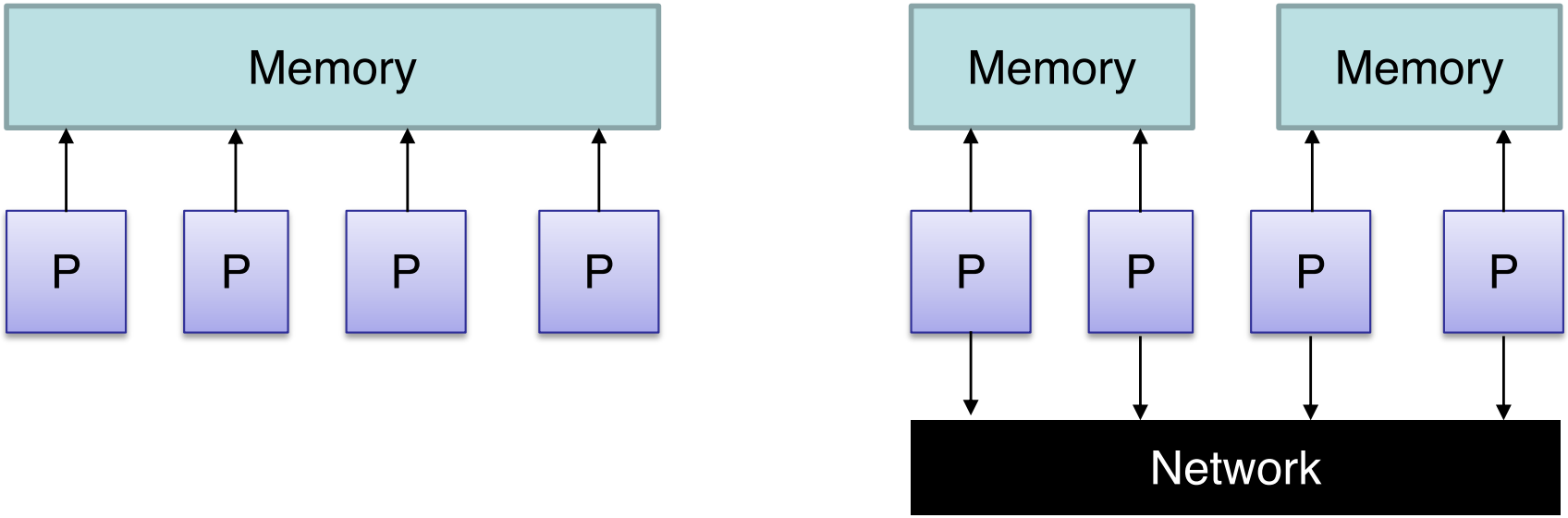
Communication using a network is prone to faults!

- Bad connections
- Network overload
- …



Challenges scale-out

Data locality: moving data over the network takes time



NoSQL

- Definition from http://www.nosql-database.org/
  - Next Generation Databases mostly addressing some of the points: **being non-relational**, **distributed**, **open-source** and **horizontally scalable**.
  - **"not only SQL"**
  - Typically not good at joins

  - Origin: Needs of Web 2.0 giants – Facebook, Google, Amazon
  - Increasingly used in big data and real-time web applications
  - More than 225 NoSQL databases
    - CouchDB
    - MongoDB
    - Hadoop & Hbase
    - Neo4J
    - Cassandra
    - …

Characteristics of NoSQL databases

- Simplicity of design: schema-less
  - SQL: structures and data types are fixed in advance
  - NoSQL: dynamic schemas – add any field on the fly
  - Applications can add new fields/attributes on the fly;
  - Able to store large volumes of rapidly changing **structured**, **semi-structured** (e.g., JSON, XML), and **unstructured** data (e.g., video, audio, image, books, documents, journals, …)

- Horizontal scaling (distribution) to clusters of machines
  - SQL: **vertical scaling** (add/remove resources to/from a computer server)
  - NoSQL: **horizontal scaling** - potentially thousands of machines, potentially distributed around the world

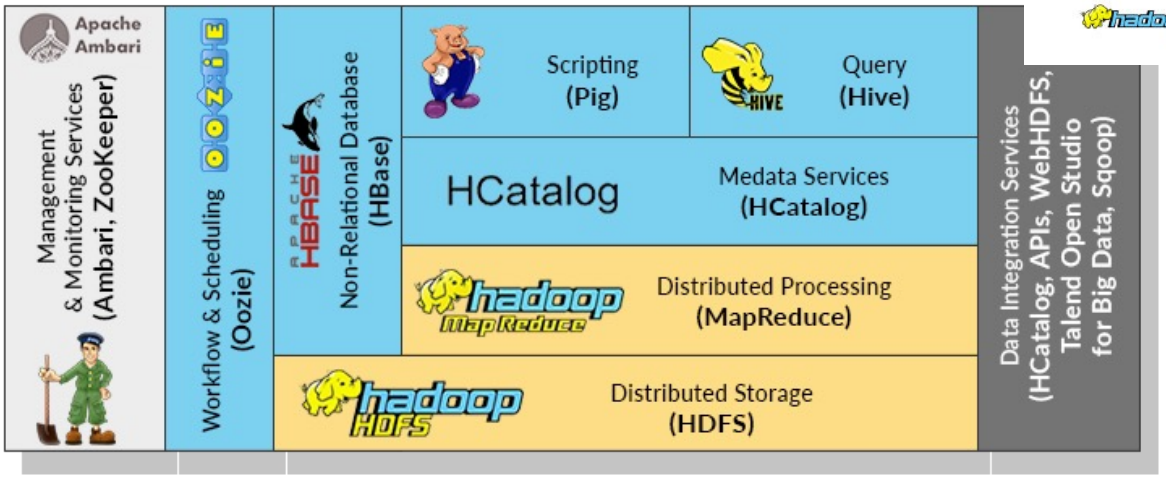
- Be always available, highly responsive queries for global users
  - NoSQL : database servers distributed around the world for high availability and low-latency queries

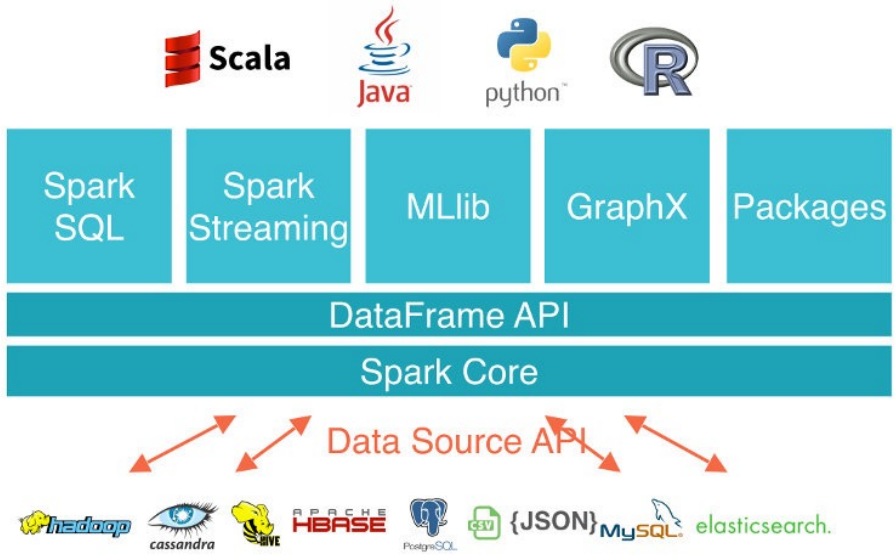
Hadoop and Spark

Motivation: use cheap machines, e.g., old servers, to compose a **distributed** cluster for processing Big Data

Hadoop framework

- 2003 Google File System
- 2004 MapReduce
- 2006 HDFS + MapReduce (Yahoo)
- 2012 Hadoop v1.0
- 2013 Hadoop v2.0
- 2017 Hadoop v3.0
- 2022 Hadoop v3.3.4

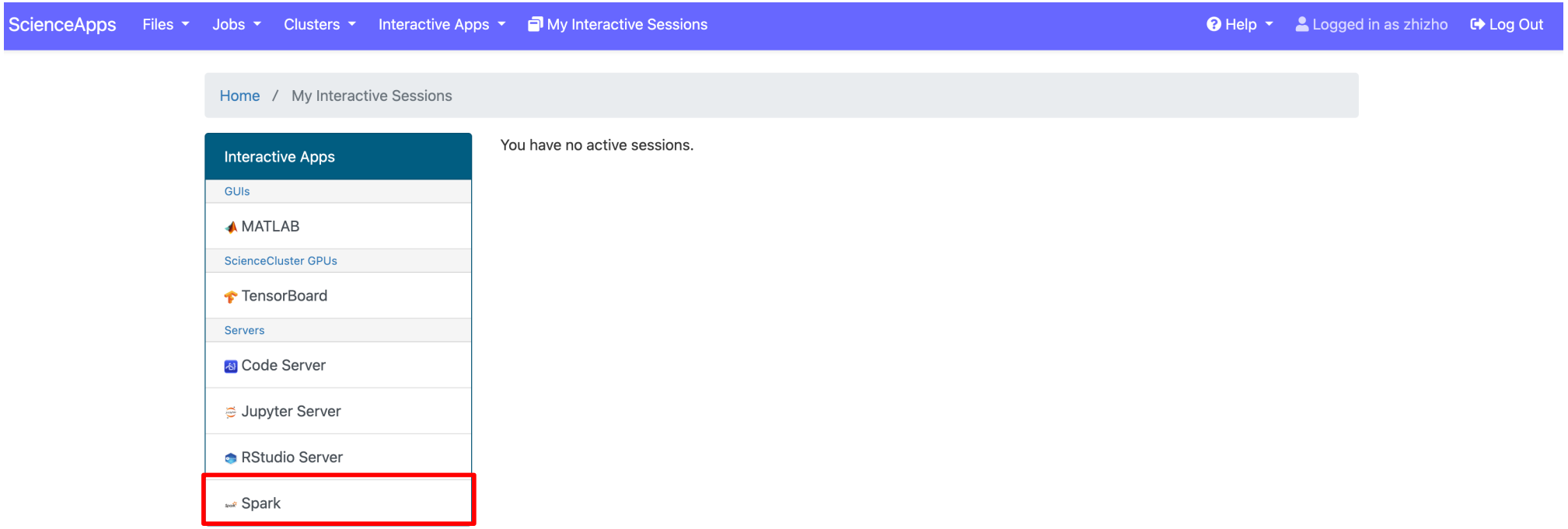




Spark framework

- 2012 UC-Berkeley v0.5
- 2014 v1.0
- 2016 v2.0
- 2020 v3.0 ANSI SQL compliance
- 2022 v3.3

ScienceCloud @ UZH



Hadoop's story

https://www.youtube.com/watch?v=ebgXN7VaIZA

Hadoop core

STORE

HDFS

(distributed storage)

PROCESS

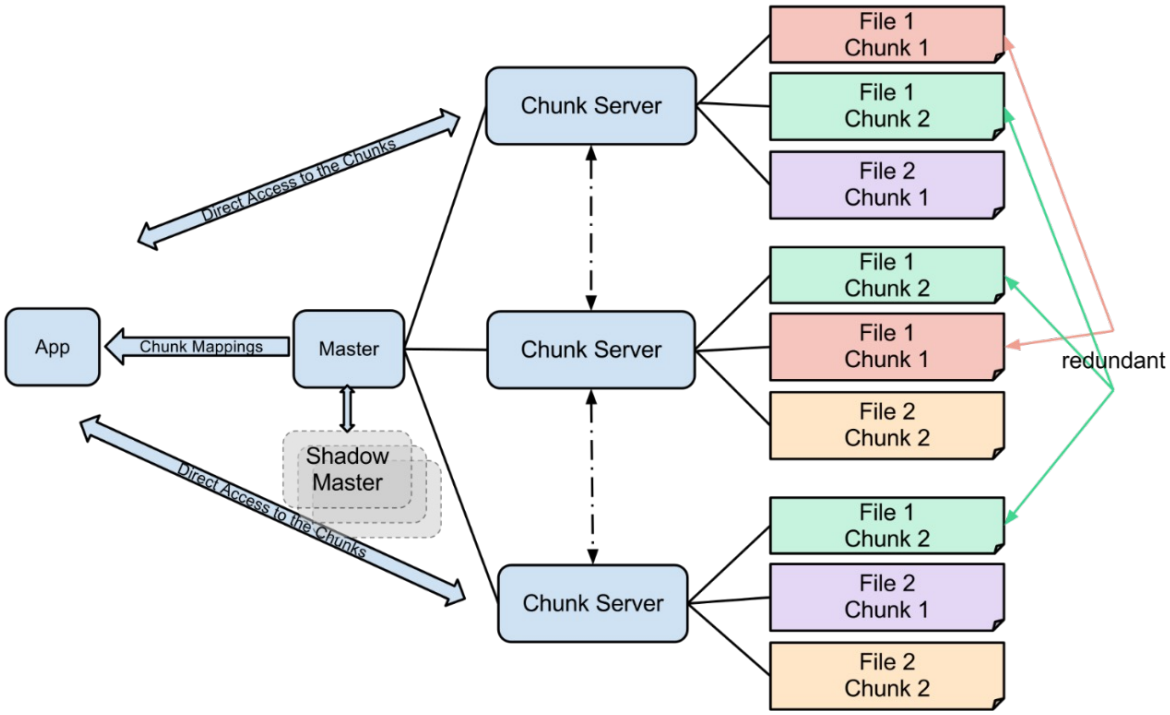
MapReduce

(distributed computing)

Distributed Storage

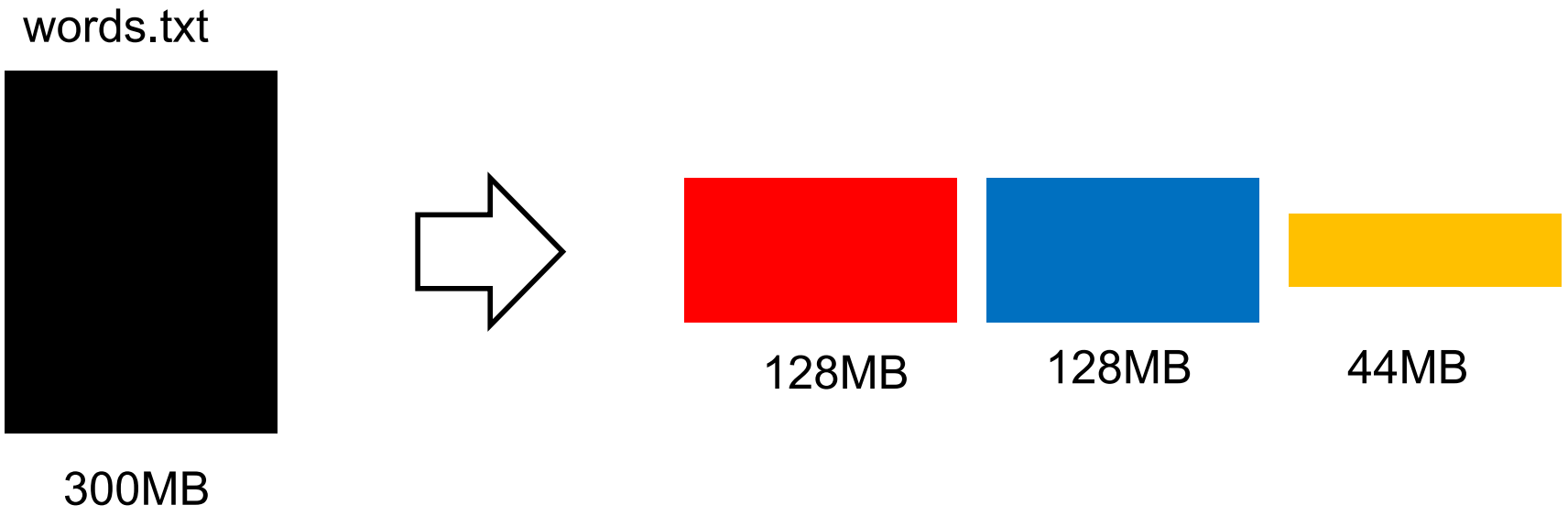
ABSTRACT:

We have designed and implemented the Google File System, a scalable **distributed file system** for large distributed data-intensive applications. It provides **fault tolerance while running on inexpensive commodity hardware**, and it delivers high aggregate performance to a large number of clients.



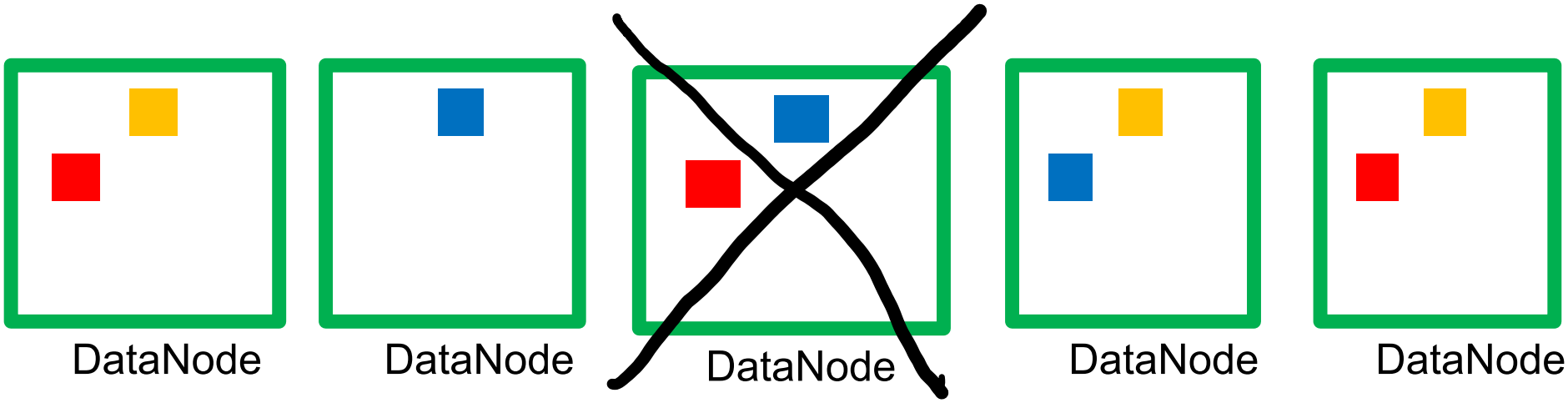
Hadoop Distributed File System (HDFS)

- Open-source implementation of GFS
- A file is split into multiple **blocks**: default size is 128MB



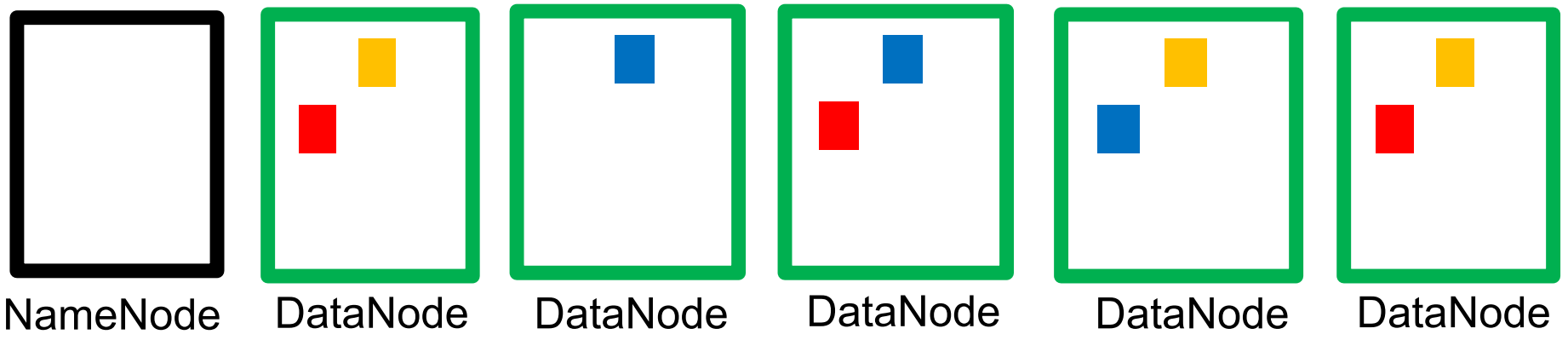
Hadoop Distributed File System (HDFS)

- **Blocks** are stored in **DataNodes**.
- Typically, there is one DataNode running on each node (computer) of the cluster.
- Every block is replicated (stored redundantly) onto multiple DataNodes.



Hadoop Distributed File System (HDFS)

- **NameNode** contains crucial metadata e.g., filenames, number of blocks, number of replicates, block ids and their locations
- The **NameNode** organizes access to the **DataNodes**.
- Running redundant **NameNode**s are possible (*Active* and *Standby* states)



GFS / HDFS

- Replication ensures that data are available even if one or multiple DataNodes fail.
- Replication increases the performance if multiple clients want to read the same file.
- If a client is running on the same computer as a DataNode it can directly / locally read the files available on this DataNode.

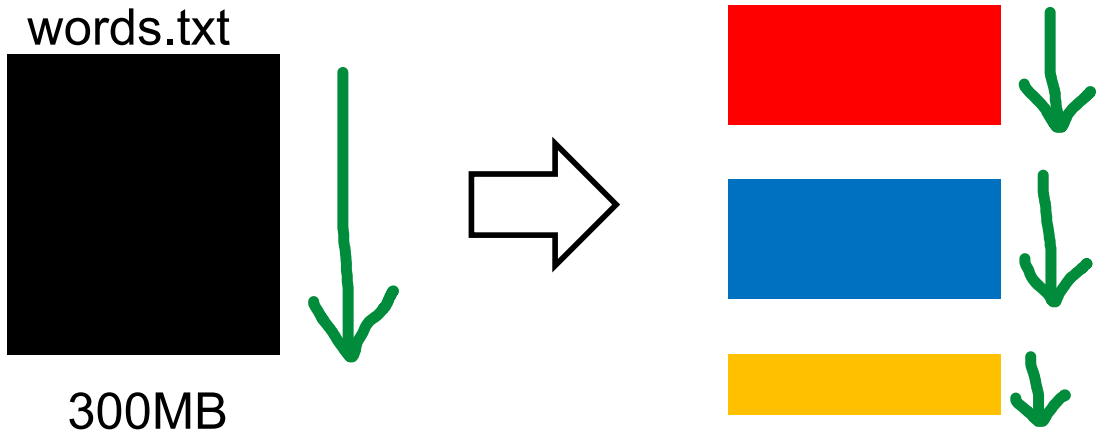
Distributed computing

MapReduce

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. https://ai.google/research/pubs/pub62

- Aim: Simple (to use) abstractions to create / specify parallelized computations

- Framework is responsible for:
  - Parallelization
  - Fault tolerance
  - Data distribution
  - Load balancing



MapReduce and relational algebra: Selection

- map: *filter* function:

```
map(row):
    if row[‘age’] >= 34:
        emit_intermediate(row, _)
```

- reduce: do nothing:

```
reduce(key, values):
    emit(key)
```

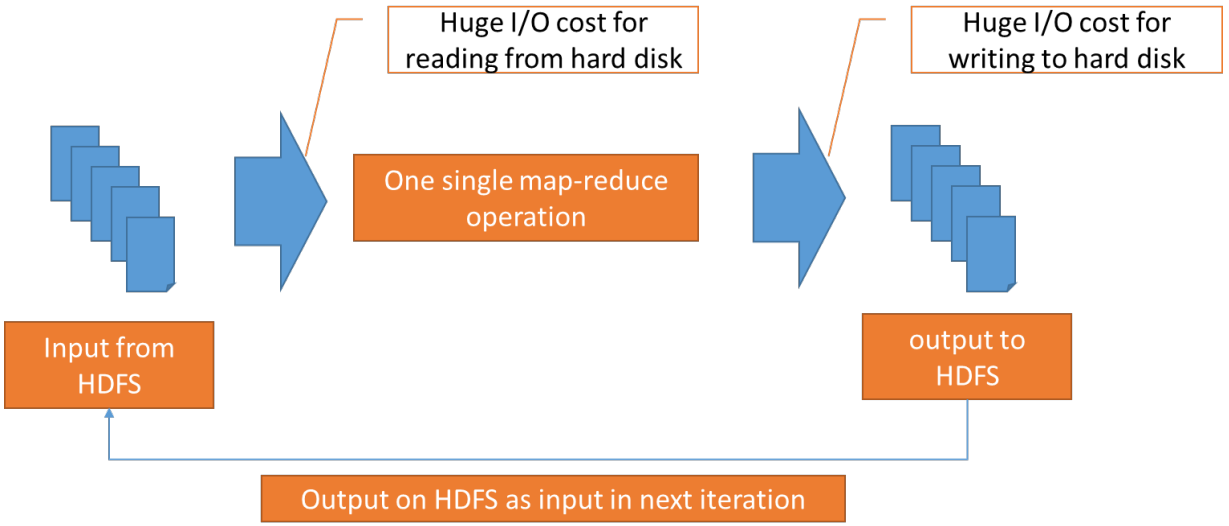
Apache Hive

- SQL-like interface for Hadoop


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

Limitation of Hadoop

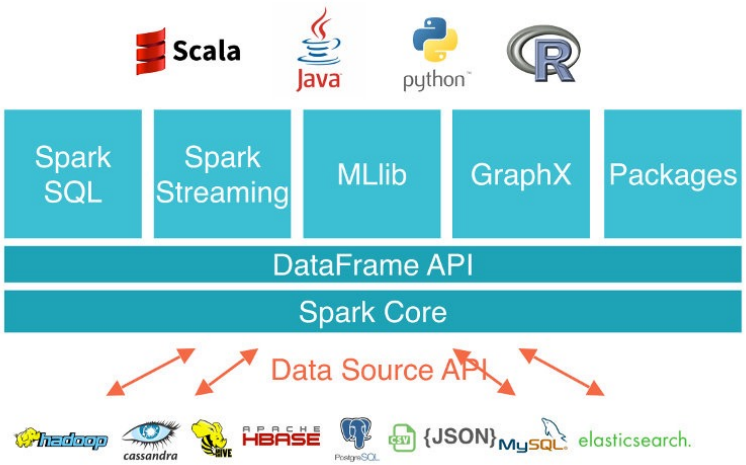
- Good for one-pass batch workflow but not good for multi-iteration workflows (e.g., machine learning models)
- No efficient primitive for data sharing
  - State between steps always go to HDFS
  - Slow due to replication & disk storage



Spark

- Based on MapReduce (developed in 2009 in UC Berkeley)
- Integrated into the Hadoop «ecosystem»

- **In-memory cluster computing:** all data are kept in memory - RDDs
- Queries are optimized in a way that minimizes unnecessary transport over the network between different map / reduce steps
- High speed: faster than Hadoop MapReduce (may be more expensive)
- Support for SQL, streaming data, machine learning, graph algorithms



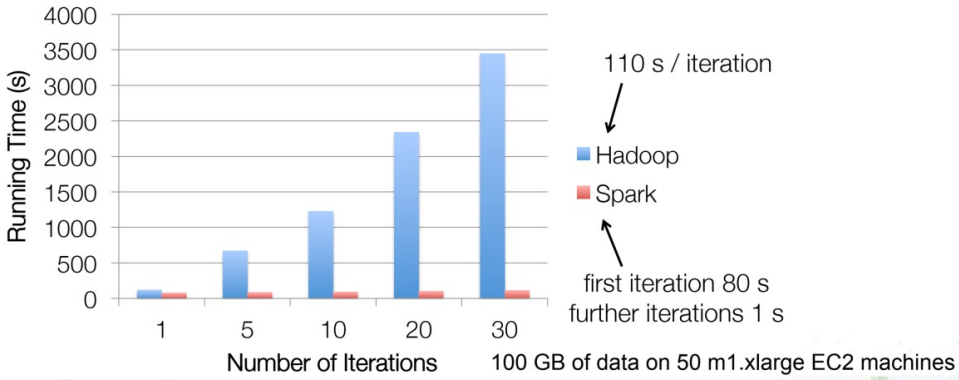
Much faster

But more expensive than Hadoop …
But that is not your concern at UZH because of the ScienceCloud

Sorting

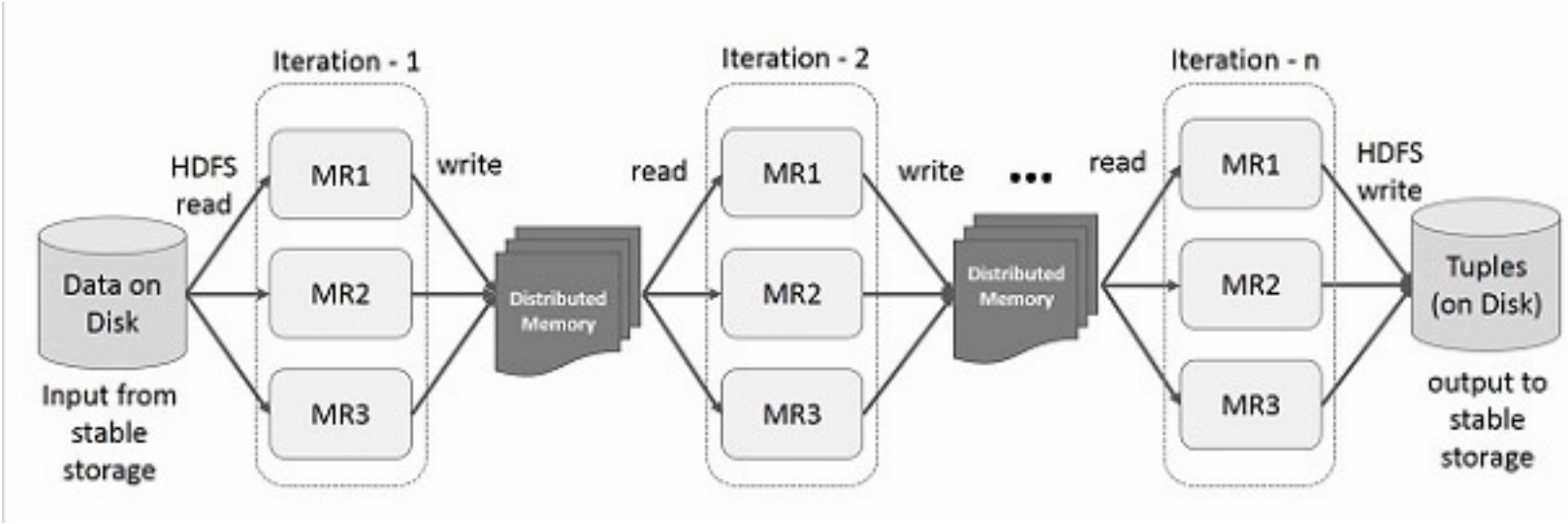
	Hadoop MR Record	**Spark Record**	**Spark 1 PB**
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Logistic Regression: memory caching



Resilient Distributed Datasets

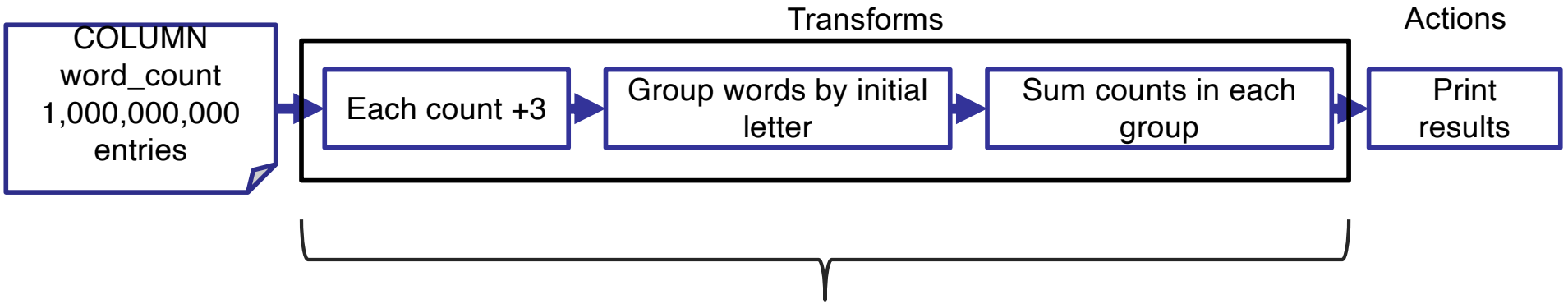
- RDD is read-only distributed (across the cluster) objects that are recomputable



MR1, MR2. MR3 are RDD

Resilient Distributed Datasets

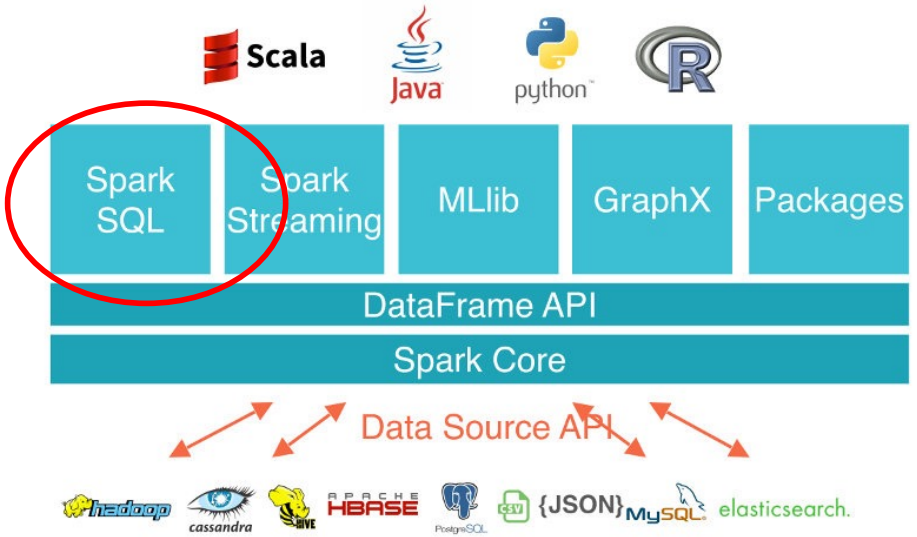
- RDD allows lazy evaluation of operations
  - Transform: Spark marks the operation on RDD and transformed to a new RDD but not computes values
  - Action: Spark evaluate the whole lineage of previous transforms to compute the whole graph



Transforms are just placeholders of operations in a workflow that no real computing will happen until an action is executed.

Use Spark like a DB: Spark SQL

- Spark SQL can handle different input sources, e.g., Hive, CSV, JSON, etc. (*Variety* in 3V)
- All inputs are transformed as uniform Spark dataframe (tables) and acts the same
- SQL queries can be applied to dataframes



Spark SQL

```
from pyspark.sql import SQLContext, Row
sqlCtx = SQLContext(sc)
```

Transform a text file in HDFS to a table

```
# Load a text file and convert each line to a dictionary
lines = sc.textFile("examples/src/main/resources/people.txt")
parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

# Infer the schema, and register the SchemaRDD as a table.
peopleTable = sqlCtx.inferSchema(people) peopleTable.registerTempTable("people")
```

```
# SQL can be run over SchemaRDDs that have been registered as a table
teenagers = sqlCtx.sql("SELECT name FROM people WHERE age >= 13 AND age <= 19")
teenNames = teenagers.map(lambda p: "Name: " + p.name)
teenNames.collect()
```

Run SQL queries just like a normal DB

Spark SQL

- From other data sources

From Hive:

```
c = HiveContext(sc)
rows = c.sql("select text, year from hivetable")
rows.filter(lambda r: r.year > 2013).collect()
```

From JSON:

```
c.jsonFile("tweets.json").registerAsTable("tweets")
c.sql("select text, user.name from tweets")
```

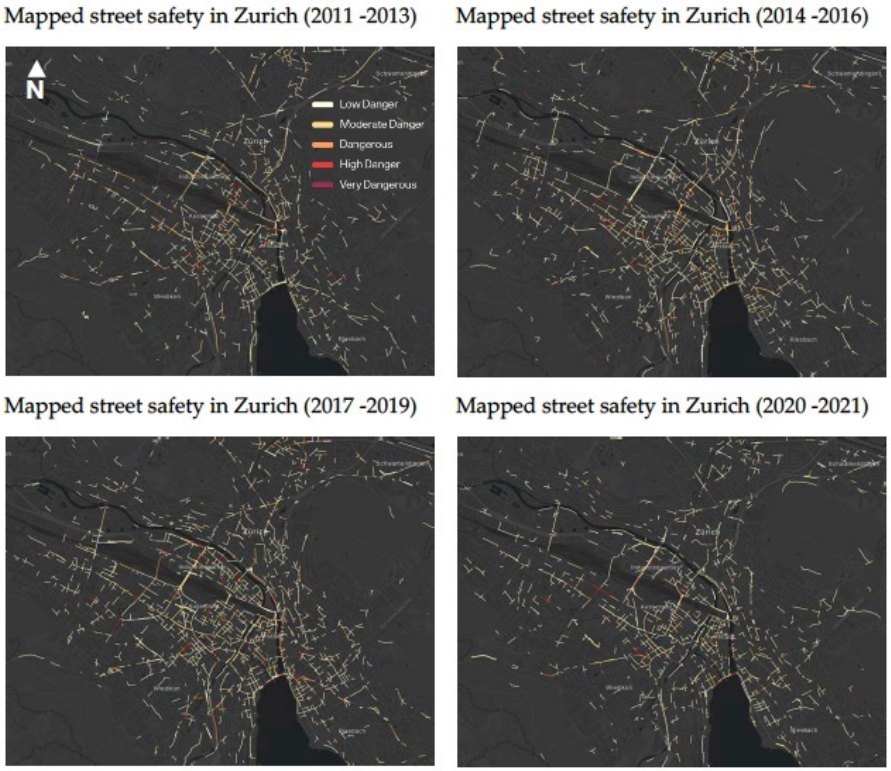
tweets.json

```
{"text": "hi",
 "user": {
   "name": "matei",
   "id": 123
}}
```

Research projects – Sustainable mobility

- **Traffic accident analysis**
  - **Data**: traffic accident data of Zurich (since 2011), Open Data Zurich
  - **Research questions**:

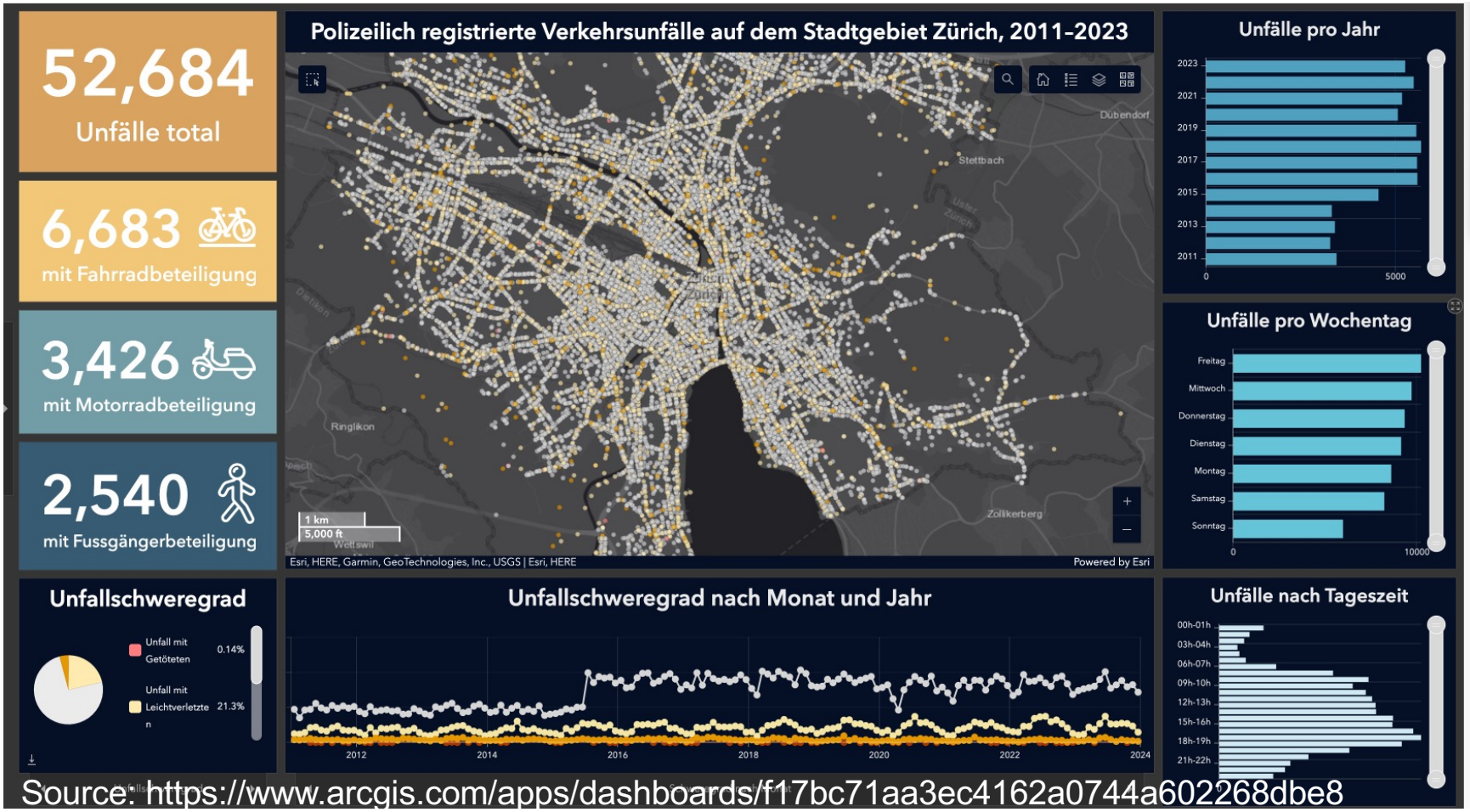
1) How do the traffic accidents vary with streets in Zurich?
2) Which street characteristics influence the occurrence of traffic accidents in Zurich?
3) Are there any temporal changes of traffic accidents in Zurich?
4) How do the Zone 30 policy influence the occurrent of traffic accidents?



Kudos to Guler Johannes, Fehr Michael

Research projects – Sustainable mobility

– **Web map of traffic accidents**



Source: https://www.arcgis.com/apps/dashboards/f17bc71aa3ec4162a0744a602268dbe8

Seven Databases in Song

https://www.youtube.com/watch?v=jyx8iP5tfCI

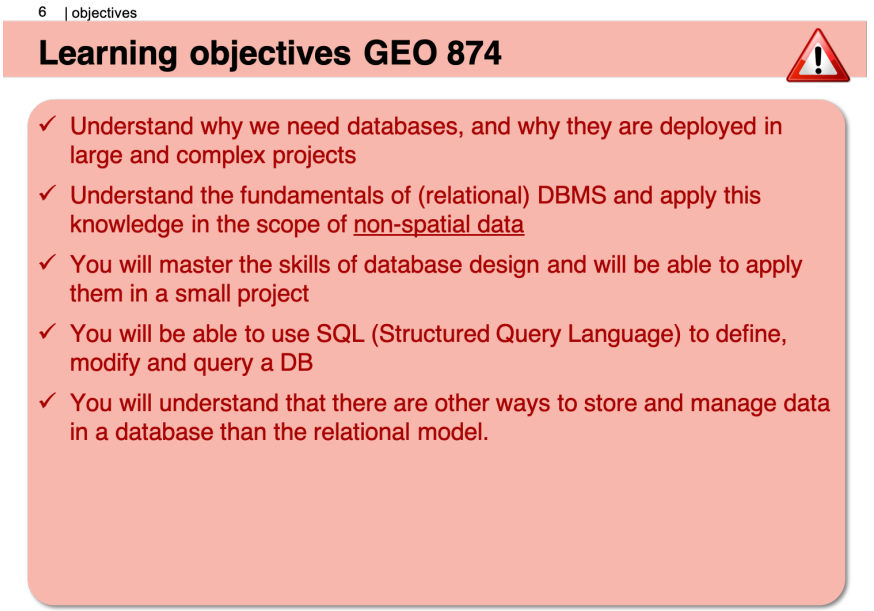
Recap

What we covered in this semester

- Why we need databases
- Basics of relational DB
- Relational DB design: requirement analysis → conceptual DB design → logical DB design → physical DB design
- Structured Query Language (SQL), PostgreSQL
- NoSQL

Hands-on in Geo 875

We hope you will find these useful in further studies and career.



Later, in lab

- 10:15-12:00
- Y25-J-09, Y25-J-10
- **Physical DB design and realization**