Prof. Dr. Florian Heinz
florian.heinz@sysv.de

Modern Database Concepts

Chapter 1: Big-Data Management

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Lecture and Exercises

Lecture (starting 2022-03-22)

- Tuesday, 13:45 Uhr (Room K018)

Exercises (starting 2022-03-25)

- Friday, 11:45 Uhr (Room K222)
- Friday, 13:45 Uhr (Room K222)

Exam

Written exam

- Examination mode: written exam
- Time: 90 minutes
- Permitted aids: TBA
- More details later this semester.

Contents

Big Data

- Big-Data Platforms
- Data Formats
  (CSV, XML, JSON)
- MapReduce
- NoSQL Databases
- RDF, Graphs

Advanced SQL

- SQL/JSON
- Recursive SQL
- Spatial Data
- Temporal Data
- Spatio-Temporal Data

Relational Databases

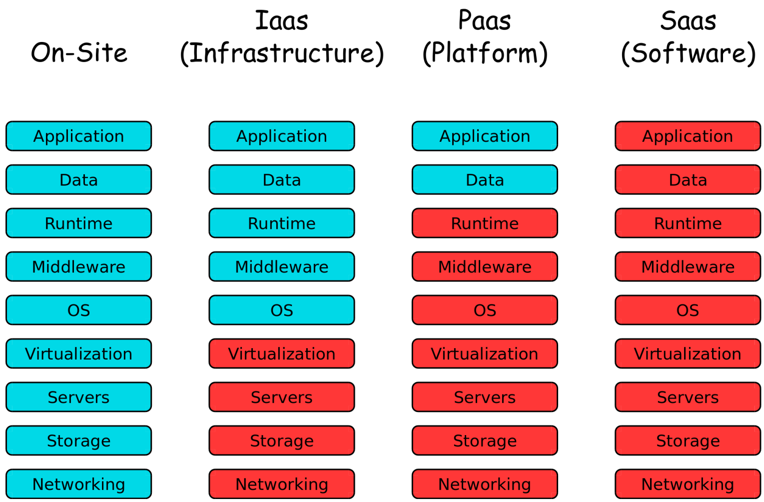
Oracle, MySQL, Microsoft SQL Server, PostgreSQL, IBM Db2, SQLite, MariaDB, ...

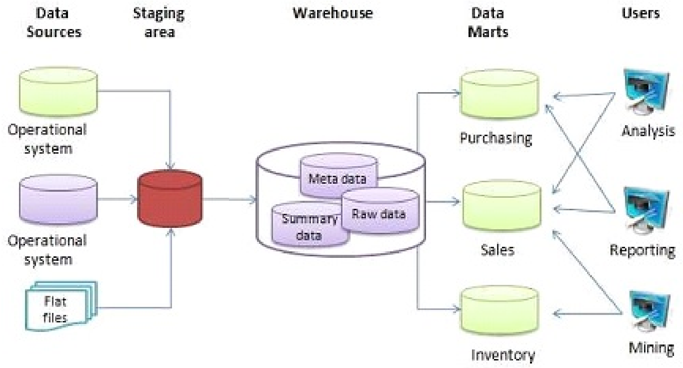
product_id	**description**	**price**
17	chocolate bar	0.89
29	dishwasher tabs	3.99

- normalized tables (3NF), free of redundancy
- fixed schema (`CREATE TABLE ...`)
- joins ⋈
- ACID transactions (atomicity, consistency, isolation, durability)
- SQL

Current Trends

- **Big Data**
  Data-Analytic Platforms, Stream-Processing, …
- **Cloud**
  IaaS / PaaS / SaaS (everything as a service)
- **Real-Time Data-Processing**
- **Unknown / flexible data formats**
- **Strong consistency is no longer mission critical**
- **NoSQL Databases**
- **Distributed File Systems**
- **Data Warehouses / Data Lakes**
- **Large-Scale Machine Learning, Data Mining**

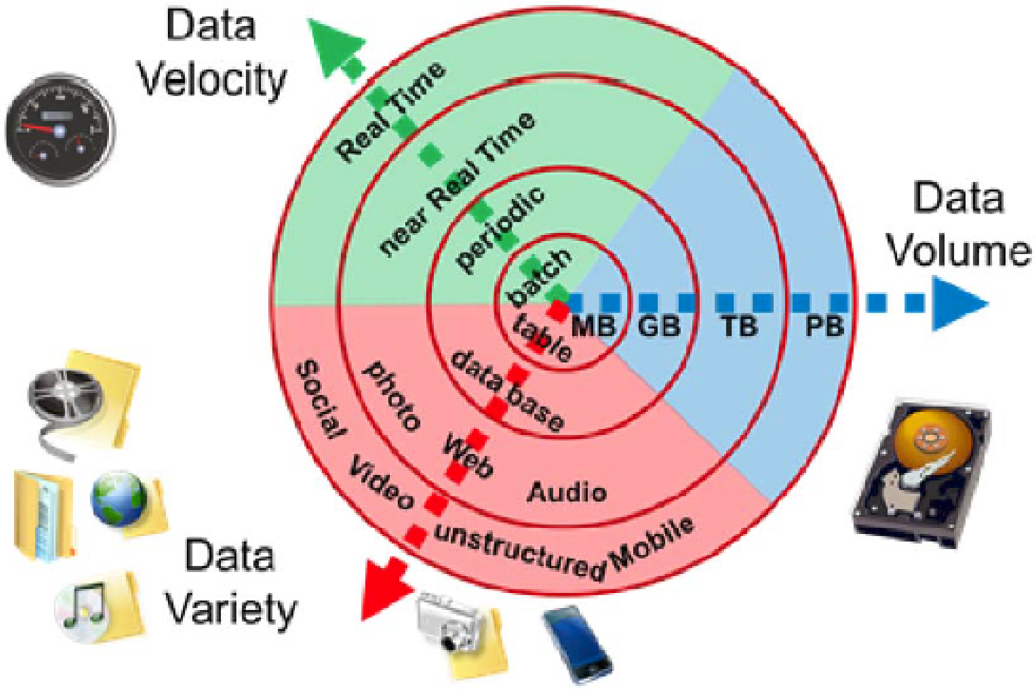




Big Data

4 Vs

- Volume
- Velocity
- Variety
- Veracity

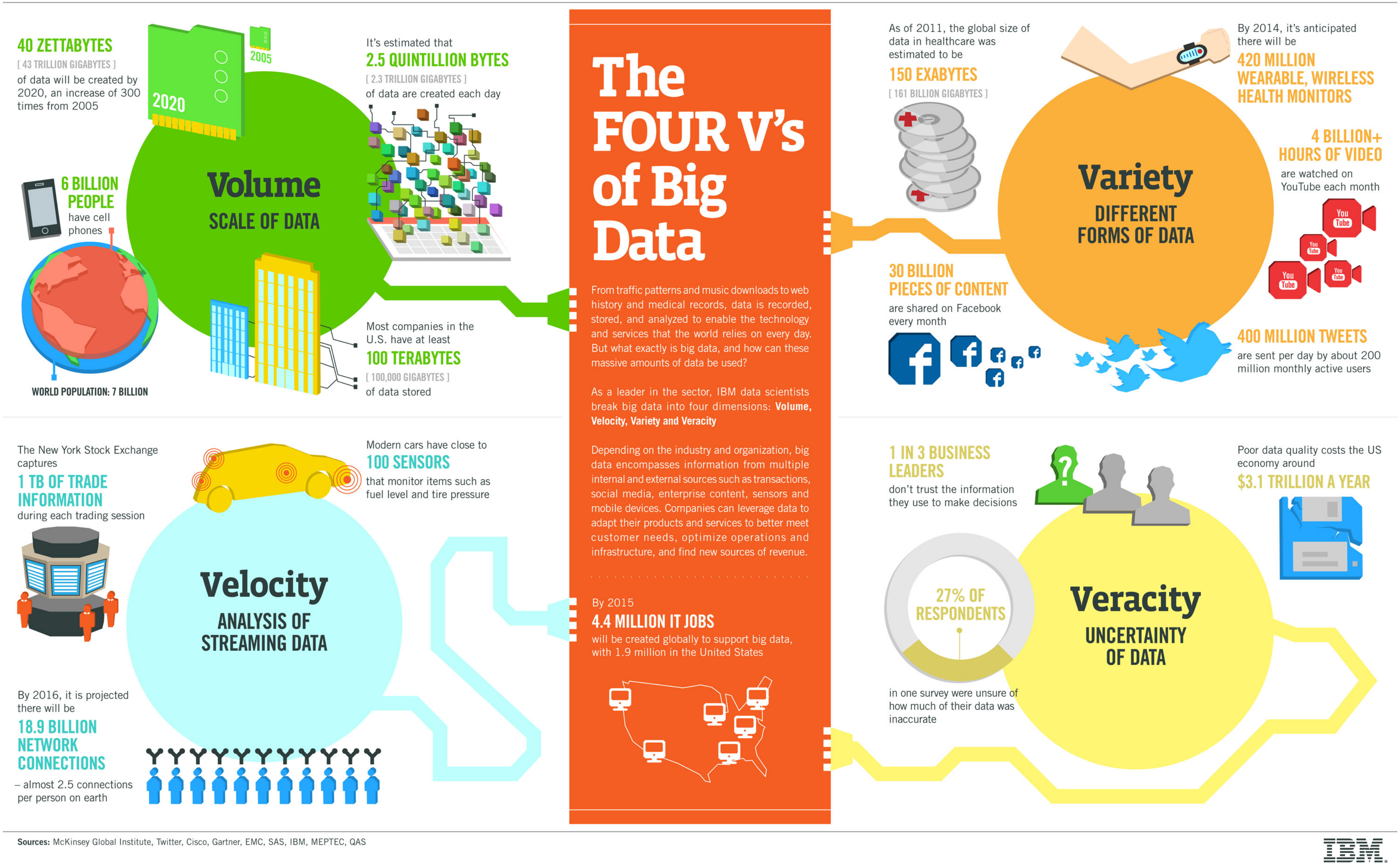


https://commons.wikimedia.org/wiki/File:Big_Data.png

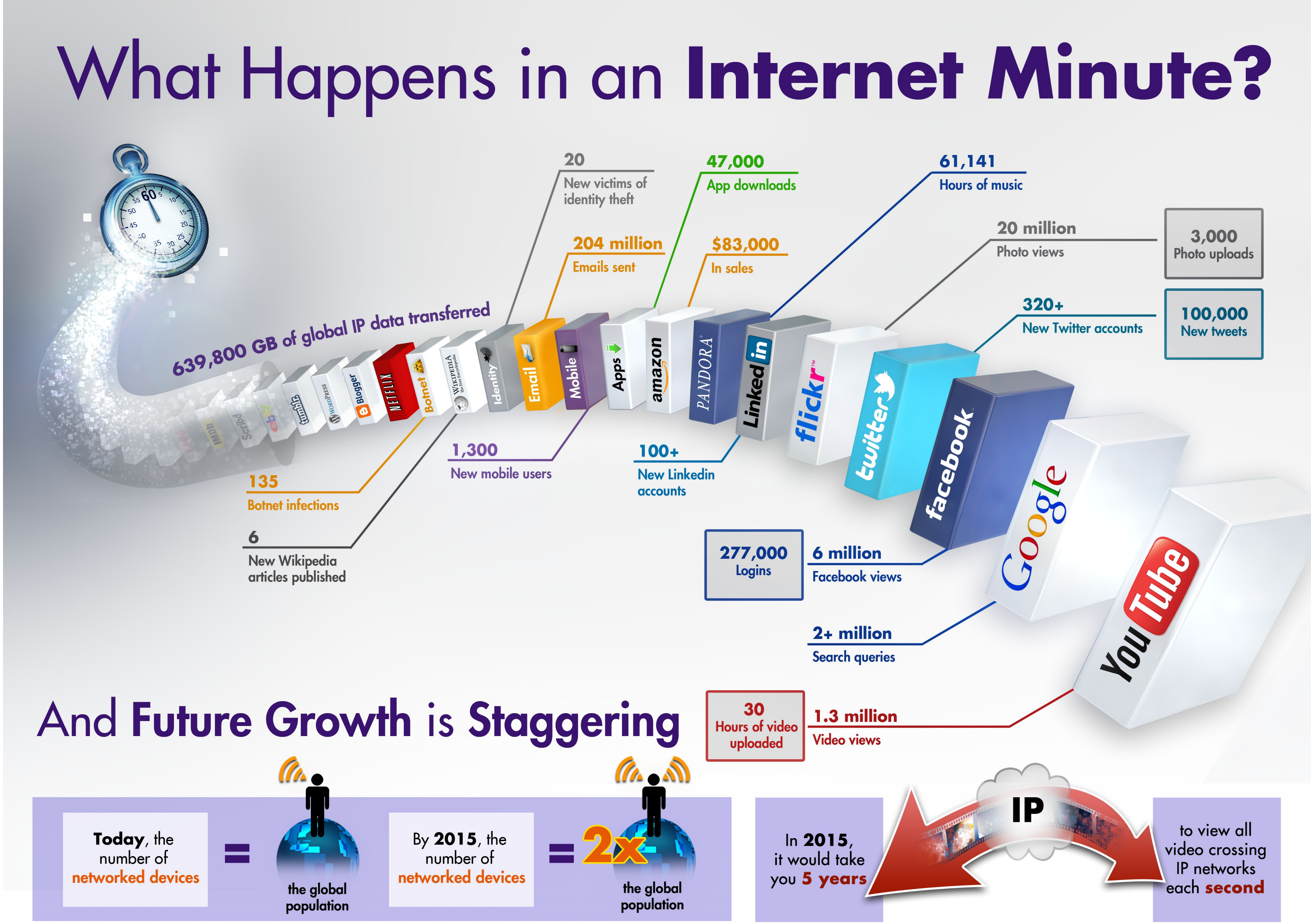
Sometimes also:

- (Value)
- (Variability)

Big Data is characterized by the 4 Vs (in other definitions 3, 5, ...). It is large (**Volume**), is created at a high **Velocity**, can have heterogeneous structures (**Variety**; structured, semi-structured, unstructured, multi-media data), and its information is often uncertain and not trustworthy (**Veracity**).



https://www.flickr.com/photos/84593672@N05/9427663067/in/photostream/



https://www.flickr.com/photos/intelfreepress/6780720740 (from 2011)

4Vs

Volume

Too large data sets to handle with traditional approaches

⇒ Scaling up / out, distributed storage / processing

Velocity

Many inserts; demand on (near) real-time processing

⇒ NoSQL-Databases, Stream-Processing Frameworks

Variety

Data without a fixed schema

⇒ XML, JSON, NoSQL-Databases

Veracity

Not clear whether the data contains true or false information

⇒ ML-Algorithms (Machine Learning), Natural-Language-Processing (NLP), ...

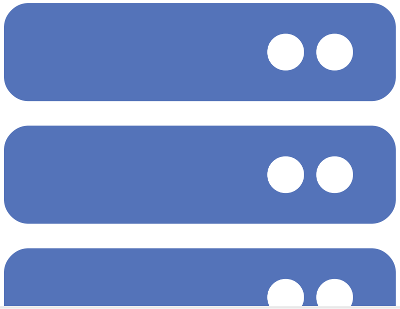
Scaling up

Scaling vertically:

- add CPUs
- add memory
- add HDDs / SSDs

(+) easy, no changes in software

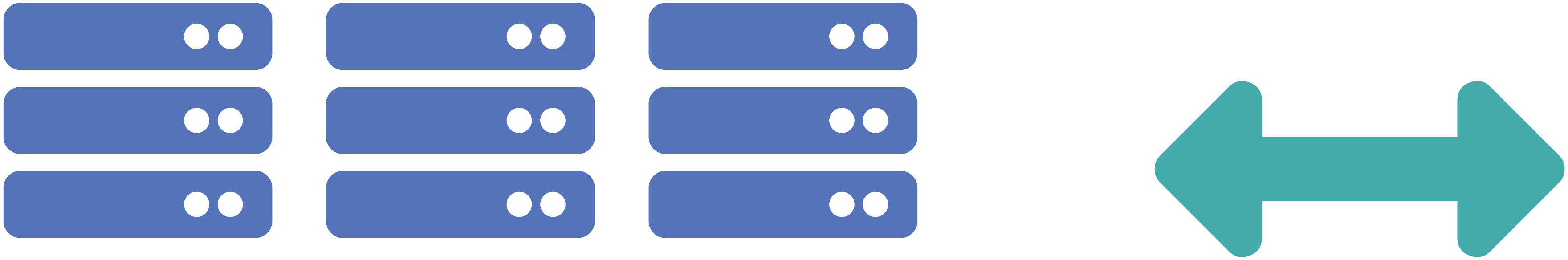
(-) expensive, limited



Scaling up means adding more resources (CPU, memory, storage, ...) or improving the resources (faster network, ...) of one machine. This approach is very limited. Special hardware is required which is often very expensive. It's cheaper to buy commodity hardware and scale out instead (see next slide).

Scaling out

Scaling horizontally: adding more nodes (machines) to a cluster



Replication

Storing copies of the same data on multiple nodes

Partitioning / Sharding

Distributed storage of data across the cluster nodes

Linear scalability means that a system becomes twice as fast when you double the number of machines in the cluster. Scaling out is cheaper than scaling up, but special algorithms, frameworks and programming patterns have to be used. The machines in the cluster have to communicate with each other to achieve a distributed storage of data and efficient distributed computations.

Replication

Storing copies of the same data on multiple nodes

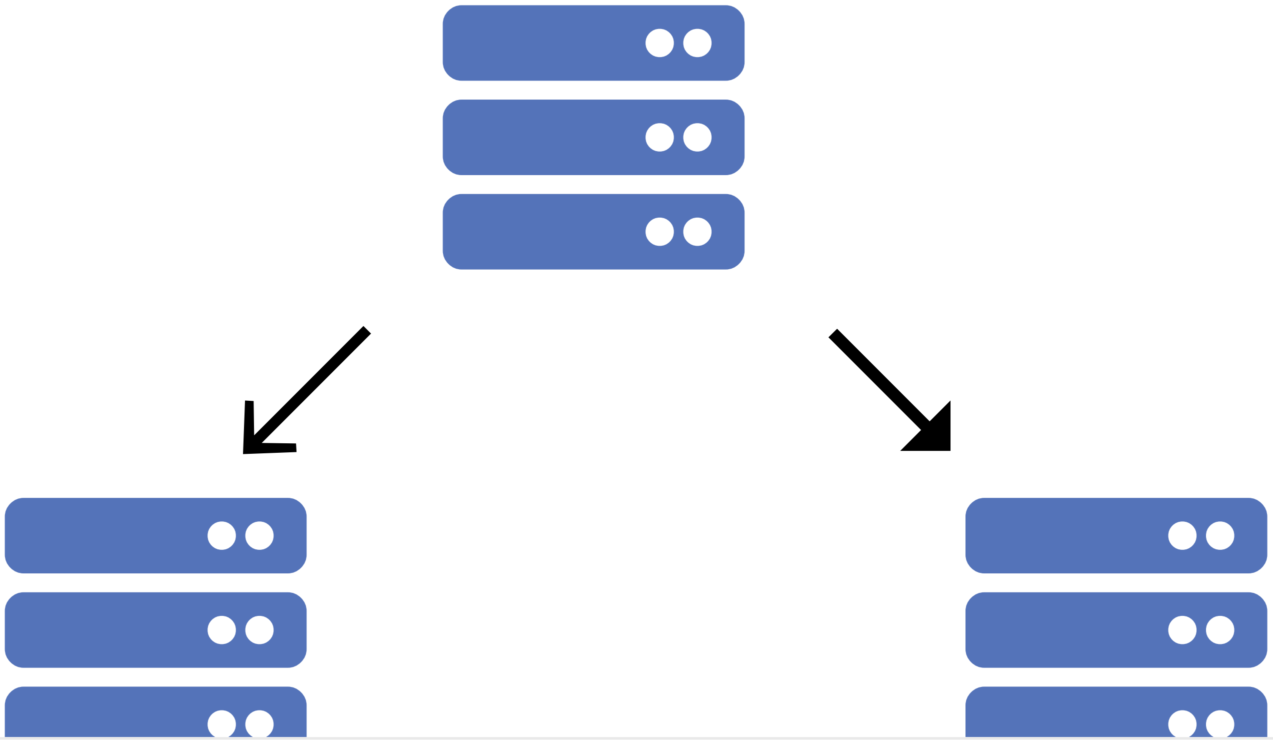
```
UPDATE products SET price = 4.50 WHERE product_id = 29;
COMMIT;
```

Example: Replication factor = 3

The main benefit of replication is high availability. This means that the system stays available (accepts and answers queries) even when one (or even multiple) node(s) in the cluster is/are down. Load balancing is a technique where clients read data from different nodes so that each individual cluster node handles less queries. This results in a higher read speed (also a higher write speed as the nodes are less busy with reading). Distributed computing is efficient when replication is used because less data has to be moved to another node. Often, replicas (the replication nodes) are geographically distributed, e.g. one in the USA, one in Europe. This increases the high availability even when a whole computing center goes down. Furthermore, clients can connect to their closest replica to improve performance.

Master-Slave Replication

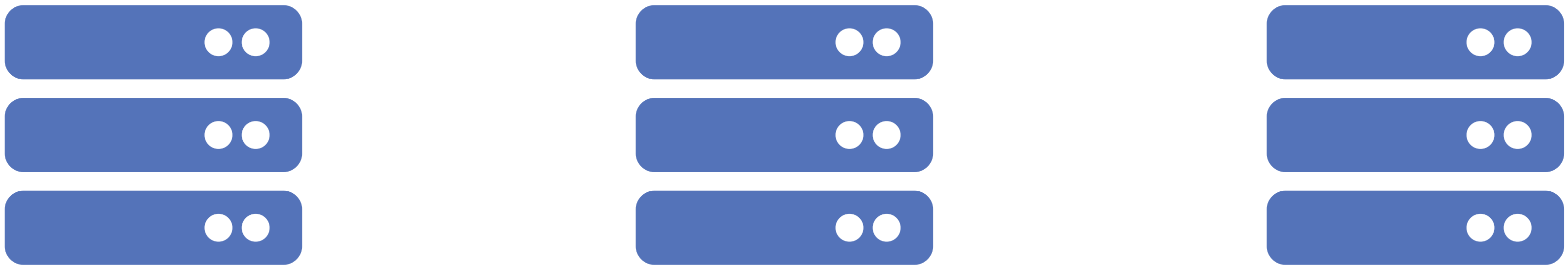
```
UPDATE products SET price = 4.50 WHERE product_id = 29;
COMMIT;
```



A Master-Master or Multi-Master Replication allows writes on multiple nodes which then synchronize those write across the cluster. As this behavior can cause inconsistencies - which write wins? - a common approach is Master-Slave replication. A master node is either selected manually when configuring the cluster or it is automatically elected by the nodes themselves (this is the case in MongoDB).

Synchronous vs. Async. Replication

```
UPDATE products SET price = 4.50 WHERE product_id = 29;
COMMIT;
```



Synchronous Replication

The client's `COMMIT` gets acknowledged when the replication is finished.

Asynchronous Replication

Replication is not part of the transaction.

`COMMIT` is acknowledged when the changes are written to one node.

The drawback of Synchronous replication is that it decreases the performance of writing transactions. When one replica is unavailable, the whole system is unavailable. The drawback of asynchronous replication is that strong consistency is not guaranteed (see next slide).

Consistency Levels

Strong Consistency

Clients read up-to-date data

Eventual Consistency

Clients may read stale data

```
SET price = 4.50
```

```
GET price;            -- 3.99


GET price;            -- 4.50
```

In traditional databases, strong consistency is often a must. The ACID paradigm says that each committed transaction has to durably (D) write all its changes atomic (A) into the database. Transactions have to run in an isolated (I) fashion so that multi-user anomalies are avoided, and each transaction has to remain the database in a consistent state (C). In modern NoSQL databases, ACID and strong consistency is not a must anymore. For a lot of applications (e.g., social media apps), eventual consistent is enough. Eventual consistency means that eventually, at some point in the future, a consistent state of the database is reached.