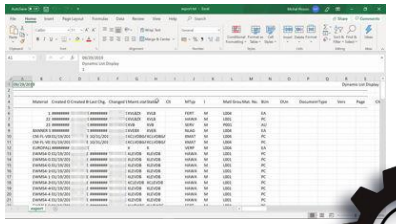
Part I: Scalable Data Access Models

MASTER SEP 2022-2023

Bart.Lamiroy@univ-reims.fr

Data Science & Analysis



Collecte
- Données brutes
- Inhomogène
- Sources variées

Nettoyage
- Débruitage
- Gestion des données aberrantes
- Gestion des données manquantes

Sélection
- Modèle des données
- Choix des données pertinentes
- Filtrage

Mise en forme
- Préparation et mise en forme informatique

Analyse
- Méthodes de traitement
- Statistique
- Algorithmique
- Apprentissage
- Intelligence Artificielle

Interprétation
- Confrontation aux connaissances contextuelles
- Validation
- Remise en cause

Visualisation
- Exploitation
- Prise de décision
- Communication

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Caveat

For the sake of ease of understanding, this lecture may occasionally make some *shortcuts* and *assumptions* that are actually *incomplete when* considered from a purely scientific or technical viewpoint.

Let some of them stand corrected in this preamble.

Goals of this Course

- « SQL vs. NoSQL » hype understanding
- Reminder of traditional transactional relational data base systems
- Exploration of limitations and bottlenecks related to Big Data requirements
- Scalability of Data storage models
  - CAP Theorem
  - Data Coherence Models
  - Data Synchronization Models
  - Data Representation Models

Preamble

ACID and RDBMS are unrelated.

- ACID is about transactions, RDMBS is software for handling relations.

NoSQL does not introduce new “theoretical” database paradigms.

- It merely revisits smart indexing techniques in the light of new underlying architectures, needs and requirements.

Outline

SQL & ACID *vs*. Scalability

The NoSQL « hype »

Looking into NoSQL

- Invariants
- Key-Value Stores
- Document Databases
- Column-Oriented Databases
- Graph Databases

SQL & ACID vs. Scalability

Relational Algebra

ACID

RDBMS

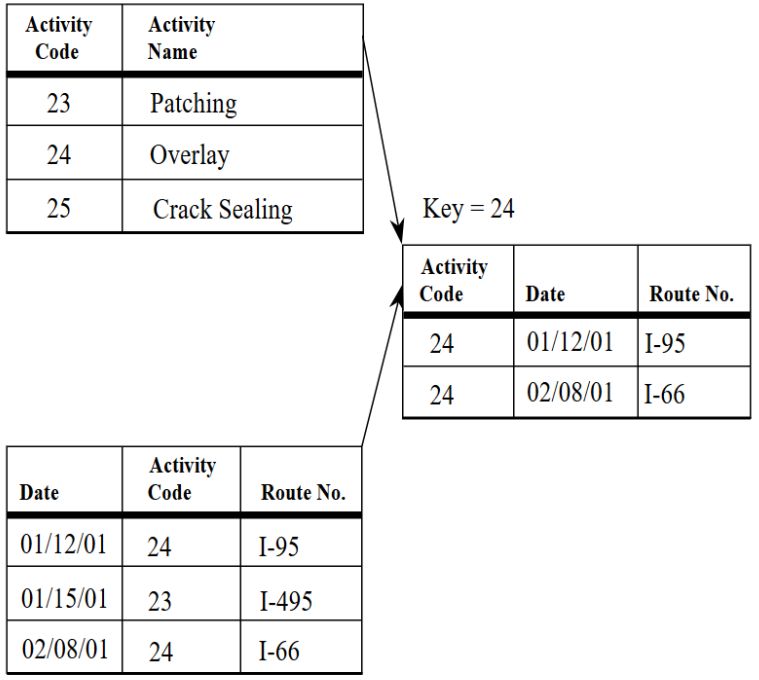
Overall Scalability

RDBMS and Scalability

Relational Algebra

- Relational Model
- First Order Logic & Set Theory
- Formalized Operators:
  - Projection $\Pi_{\{a_1,\ldots a_n\}}(R)$
  - Selection $\sigma_{a \nabla b}(R)$
  - Join(s) $R_1 \bowtie R_2$
- SQL (Structured Query Language)

```
select a1, a2 from R;
select * from R where a1 = v or a2 = a1;
select * from R1, R2;
```



Source Wikipedia: http://upload.wikimedia.org/wikipedia/commons/d/da/Relational_Model.svg

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM 13 (6): 377–387. doi:10.1145/362384.362685, 1970.

ACID

Atomicity:

all-or-nothing transactions (in any situation)

Consistency:

data is always in a coherent state

Isolation:

concurrent transactions do not interfere and produce the same result, regardless of orders

Durability:

executed transactions are permanent

Jim Gray, "The Transaction Concept: Virtues and Limitations".
Proceedings of the 7th International Conference on Very Large Databases,
pp. 144–154, September 1981.

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

RDBMS

Oracle

Microsoft (SQL-Server)

IBM (DB2)

MySQL

PostgreSQL

Scalability

Scalability? Why?

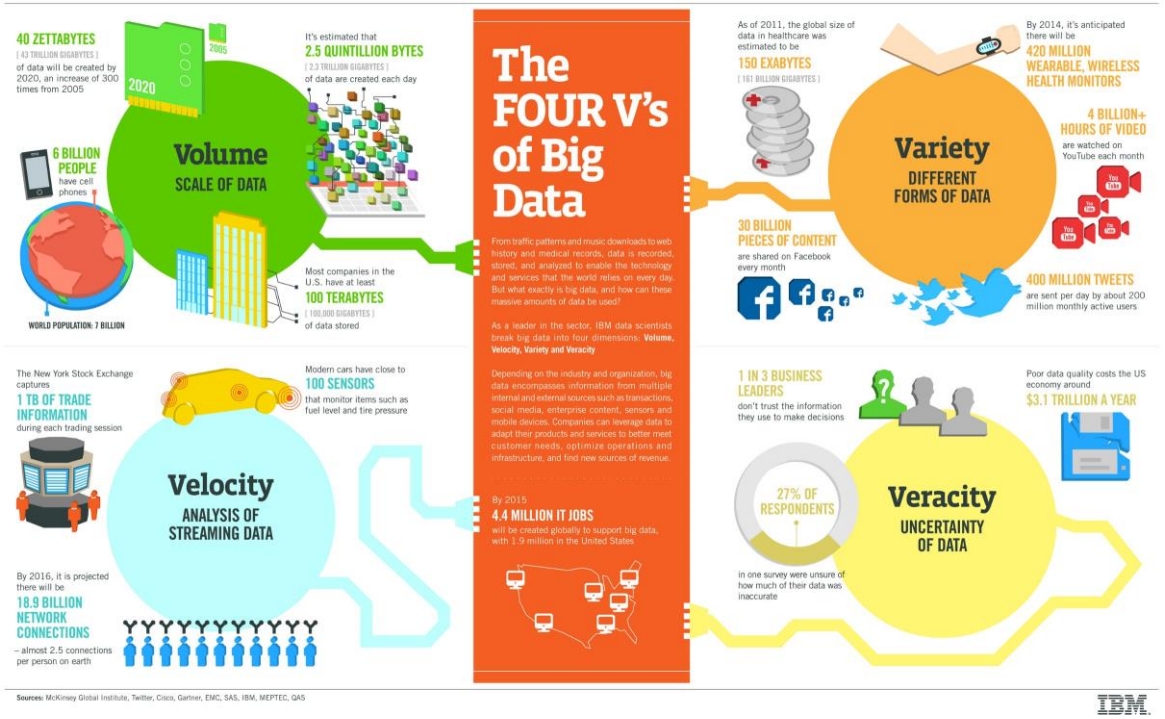
More and more data to handle

More and more queries to process

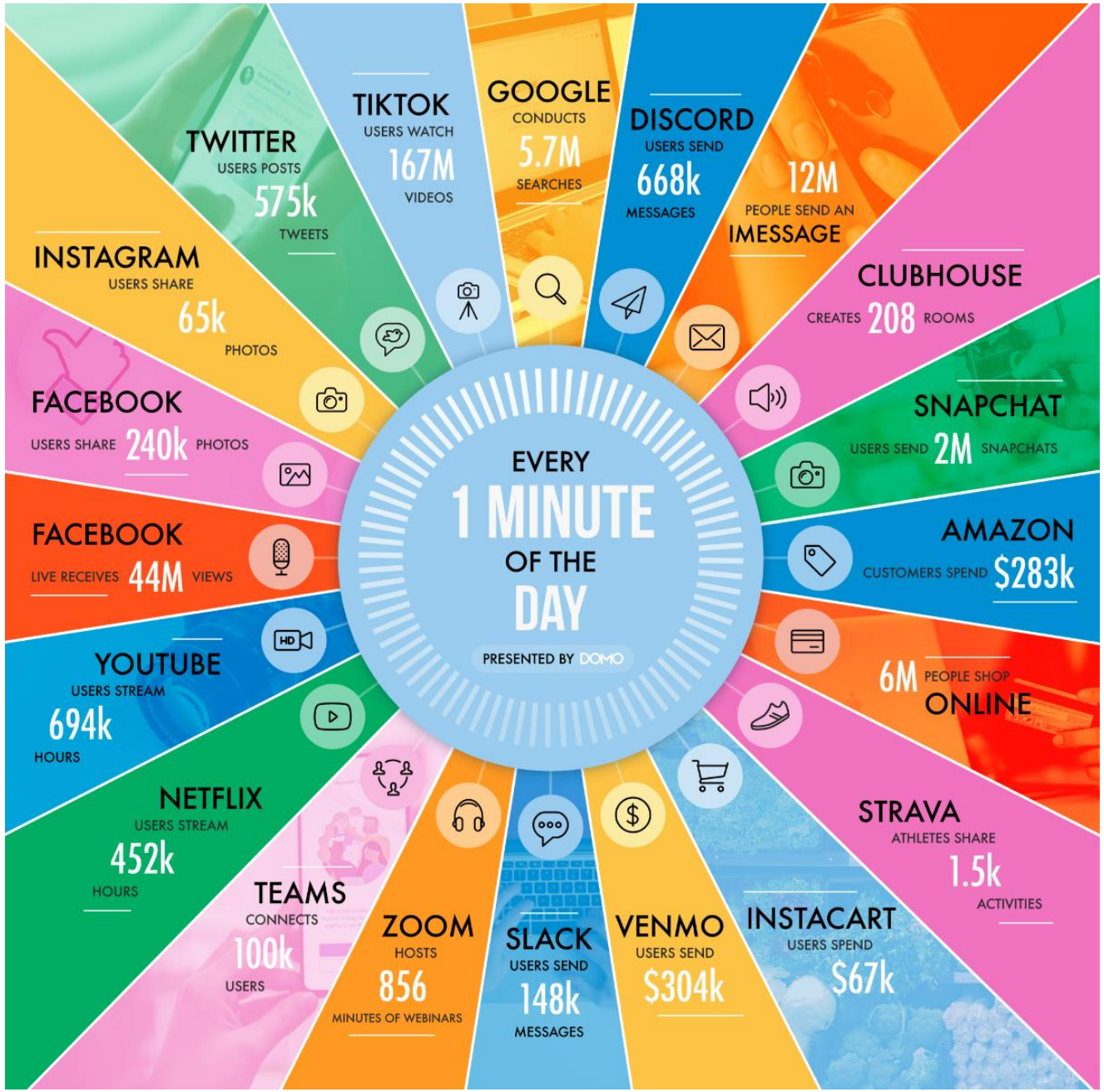
More and more complexity in queries

Large variety in data to handle

Faster and faster response times required



Scalability



Source: http://www.domo.com

Scalability

A “Neutral”, Documented and Verifiable Example:

(source: LHC – CERN)

- ~1PB/s (1PB = 10^{15} Bytes)
- Pre-filtering 1/10 000 events = 10TB/s
- 1% of the remaining is “interesting” = 1GB/s
- Result = 25PB/year (experiments do not run 24/7)

Fun fact : when mentioning storage and data, the unit generally is the byte (B); when mentioning communication speed, the unit generally is the bit (b) with 1B = 8b. Another unit is the transfer (T) for which the number of bits varies according to the underlying architecture.

Scalability

Horizontal Scalability (Scale out)

Vertical Scalability (Scale up)

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Scalability

Horizontal Scalability (Scale out)

- Parallelise
- Duplicate resources
- Introduce network

Vertical Scalability (Scale up)

- Increase resources
- Remain local

Scalability … what if

More and more data to handle	but does not fit on storage or in memory
More and more queries to process	Facebook 44M views/min ≈ 1.5µs/view ≈ 730Mhz
More and more complexity in queries	
Large variety in data to handle	one-size-fits-all relational data model ?
Faster and faster response times required	

RDBMS Adaptation to Scalability

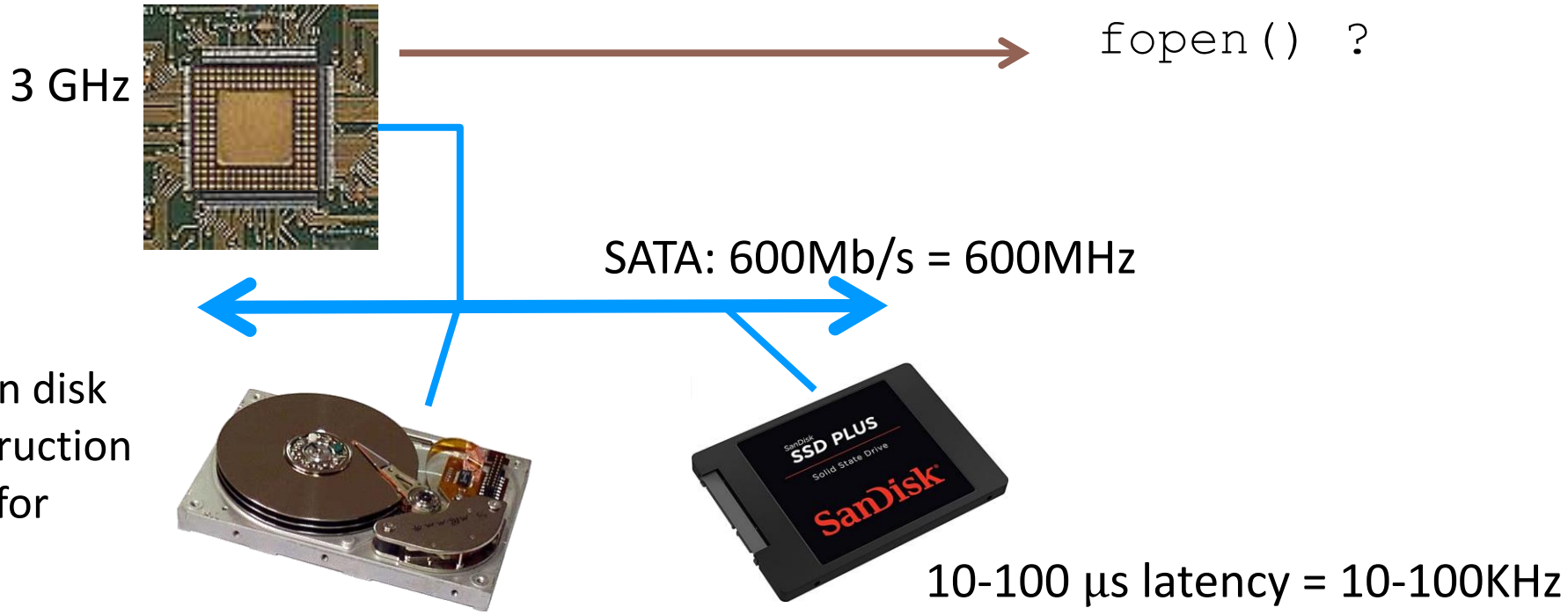
Difficulties:

- Requirement of common schema
- Bottleneck limits of vertical scaling
- ACID limits on horizontal scaling

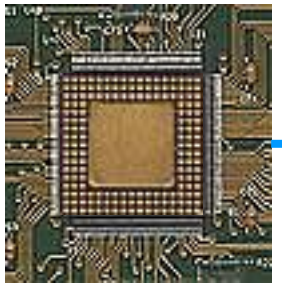
Solutions:

- Caching
- Clustering
- Sharding
- Read/Write Separation

Quick reminder on disk access



Quick reminder on disk access



3 GHz

`fopen() ?`

x50

SATA: 600Mb/s = 600MHz

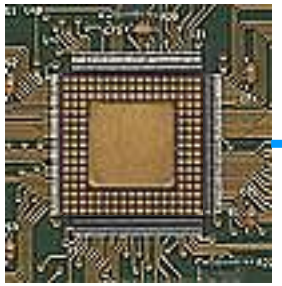
x1000000

2-10 ms latency = 100-500Hz

x5000

10-100 µs latency = 10-100KHz

Quick reminder on disk access



3 GHz

`fopen()` ?

x50

SATA: 600Mb/s = 600MHz

x1000000

2-10 ms latency = 100-500Hz

x5000

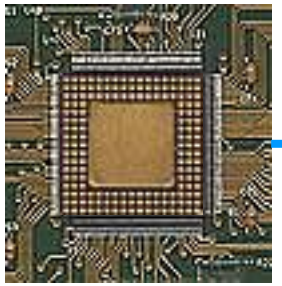
10-100 µs latency = 10-100KHz

Solutions

- Prefetching
- Caching
- Parallel reading/writing (SSD)
- RAID

Quick reminder on disk access

Fast Networks
100 Gb/s



3 GHz

`fopen() ?`

CERN Example
10 Gb/s after filtering

Facebook Example
730MHz views

x5000

x50

600Mb/s = 600MHz

x1000000

2-10 ms latency = 100-500Hz

10-100 µs latency = 10-100KHz

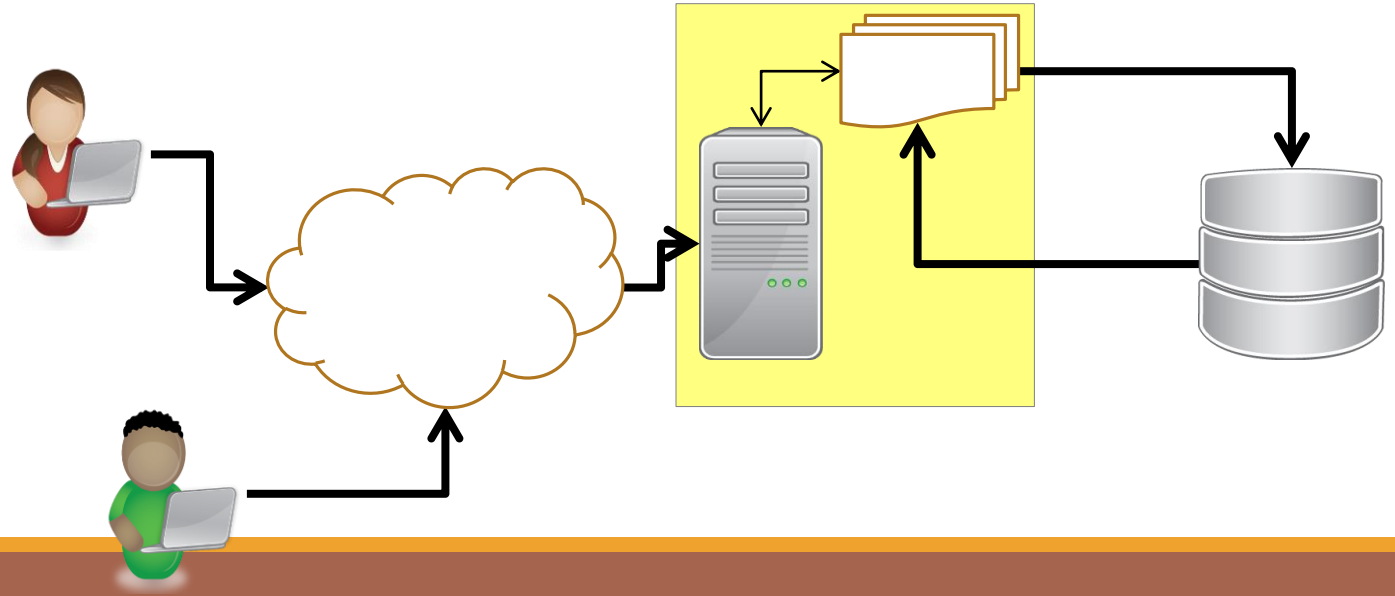
Further Reading

- “Evaluating Methods of Transferring Large Datasets”, Jakub Kopeć, Asian Conference on Supercomputing Frontiers, SCFA 2022: Supercomputing Frontiers pp 102–120 https://link.springer.com/chapter/10.1007/978-3-031-10419-0_7
-

MemCache

Cache in memory

- frequent/recent requests
- chunks of tables
- Reduces access times to database

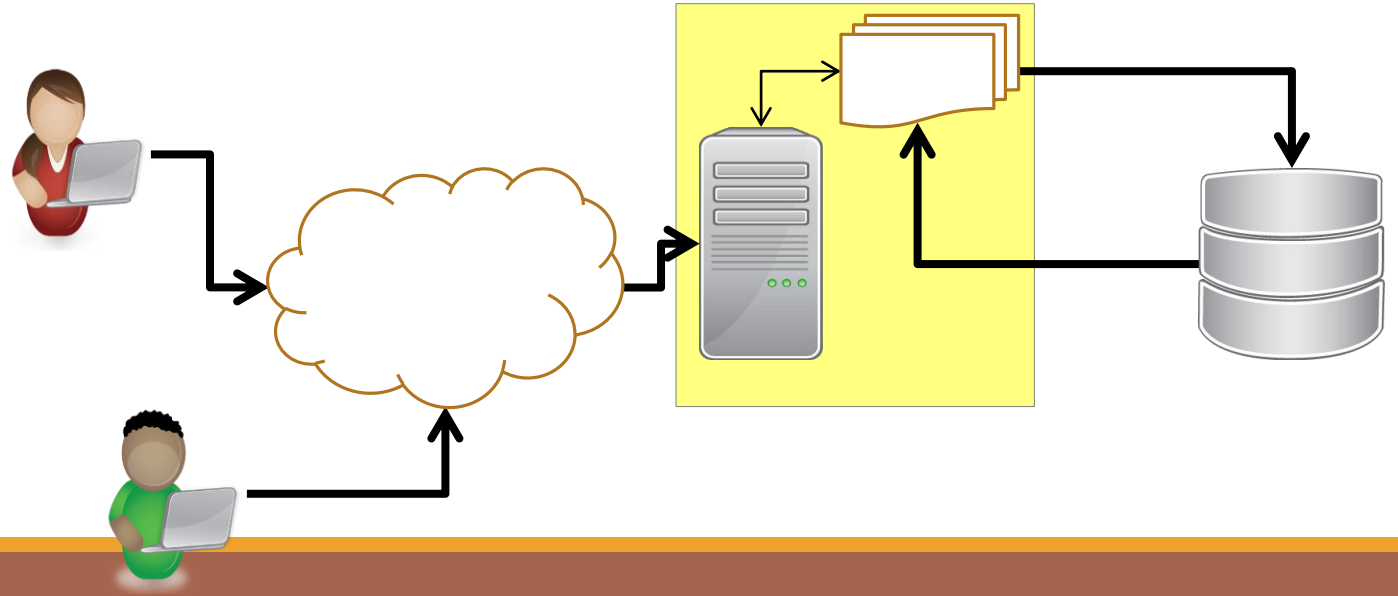


MemCache

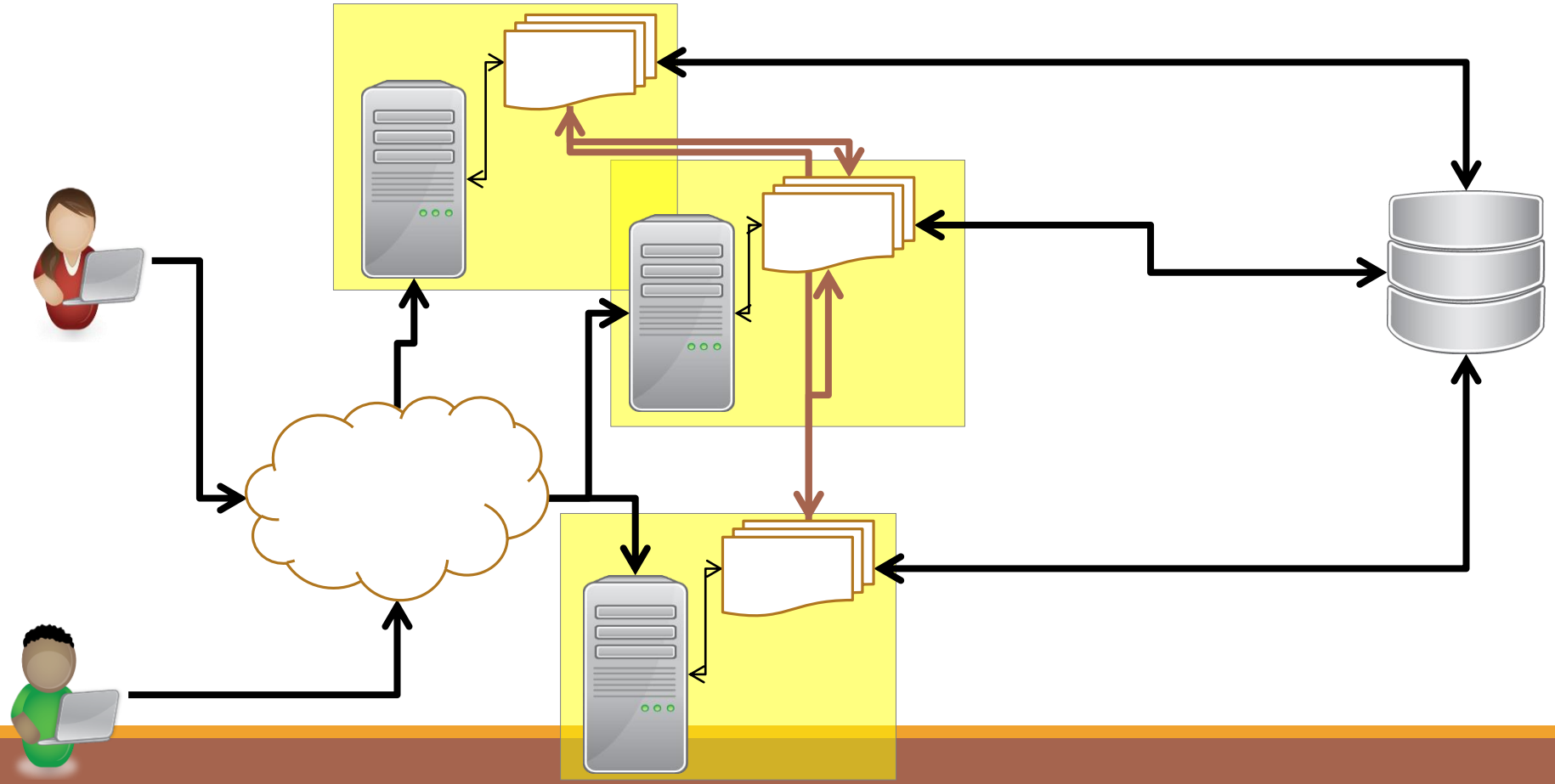
Cache in memory

- frequent/recent requests
- chunks of tables
- Reduces access times to database

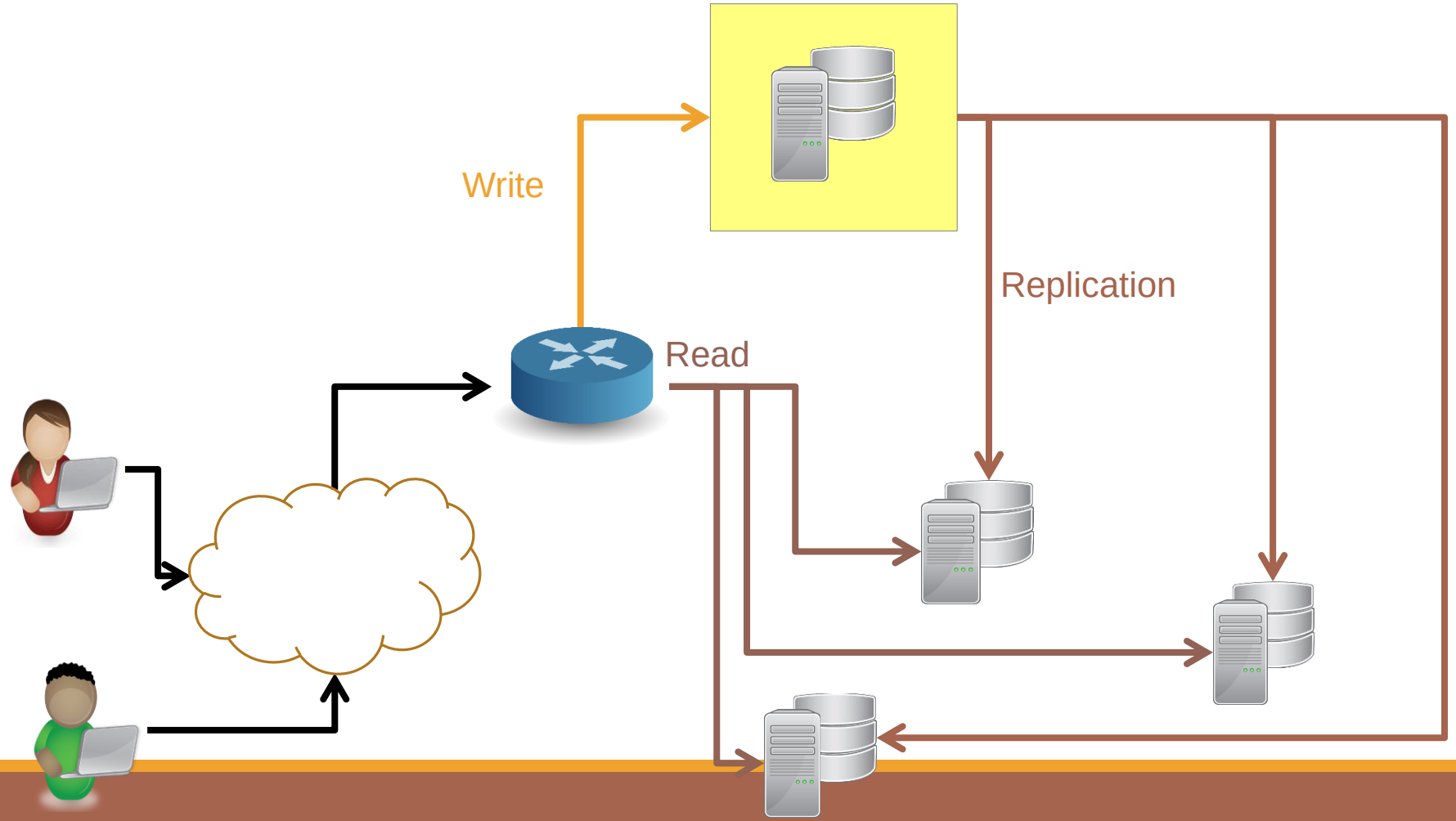
Assignment:
How does memcache handle consistency + durability & potential power failures



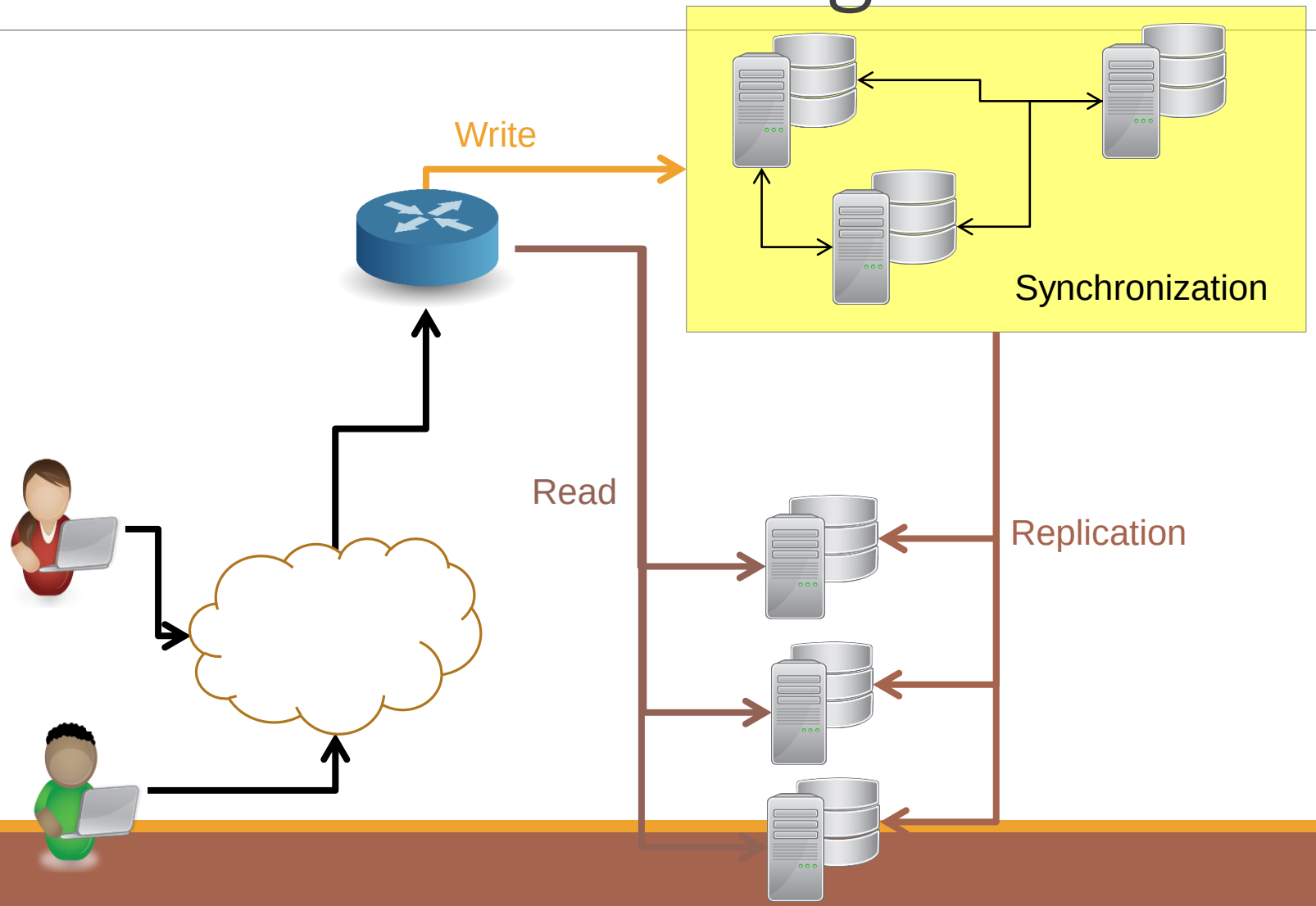
Distributed MemCache



Single Master Clustering

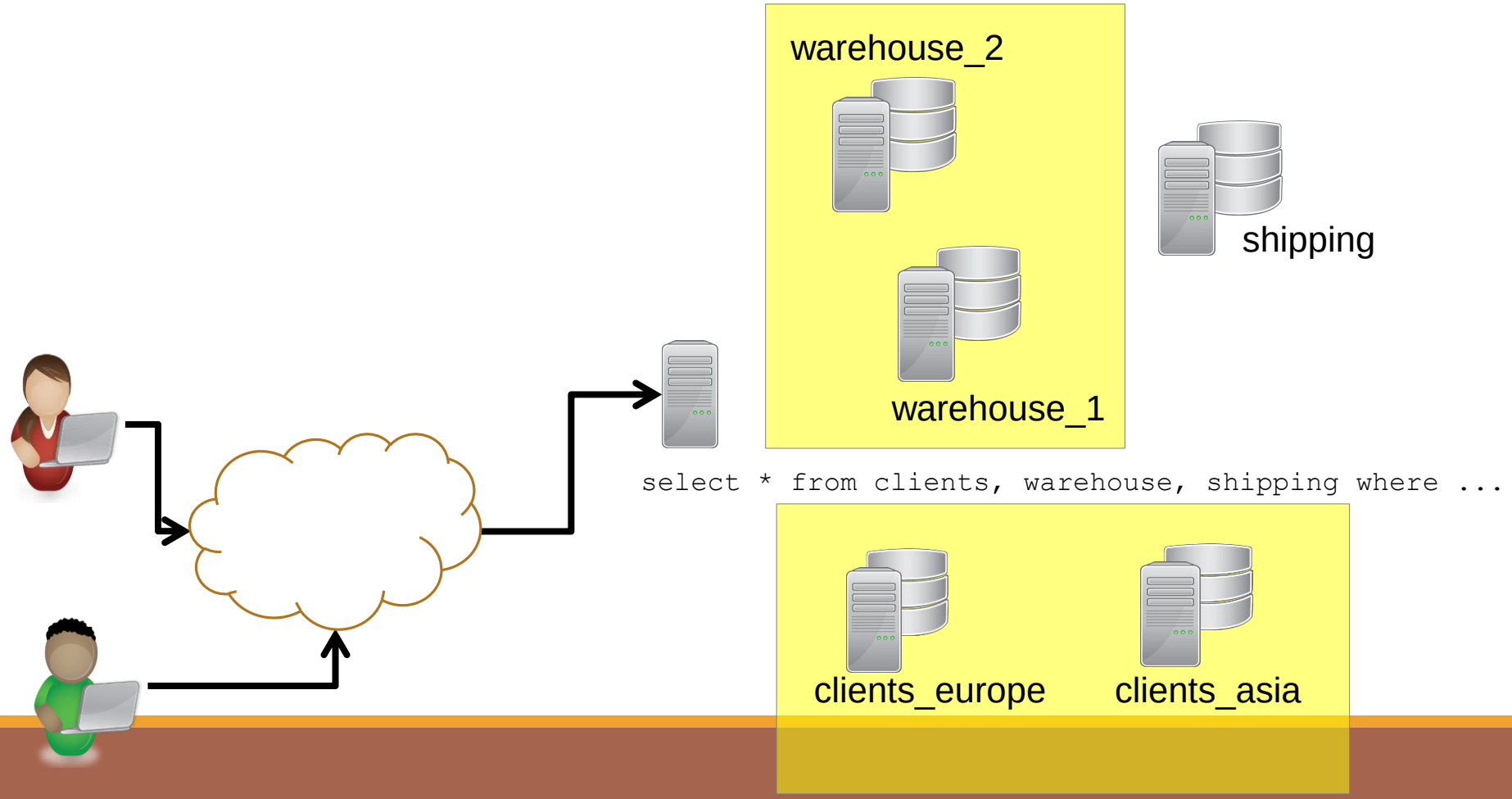


Multi-Master Clustering



Sharding

Reducing the complexity of joins over large tables...



Scalability - Problem

The 8 distributed programming fallacies (Deutsch, Gosling)

- 1. The network is reliable
- 2. Latency is zero
- 3. Bandwidth is infinite
- 4. The network is secure
- 5. Topology doesn’t change
- 6. There is one administrator
- 7. Transport cost is zero
- 8. The network is homogeneous

Outline

SQL & ACID vs. Scalability

The NoSQL « hype »

Looking into NoSQL

- Invariants
- Key-Value Stores
- Document Databases
- Column-Oriented Databases
- Graph Databases

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

NoSQL

First use: 1998 (RDBMS)

"No SQL" → Relational databases not using SQL

Recoined: 2009 (Alternatives to RDBMS)

"Not Only SQL" → Relational databases are not necessarily a solution for all data storage problems

Motivations behind NoSQL (1)

Unneeded Complexity

- no need for all ACID features always
  e.g. session data for large on-line communities

compromising reliability for performance

- High Throughput
- *e.g.* crunching the Google crawler data

Proof-by-Example of custom databases

- *e.g.* many NoSQL solutions outperform RDBMS on standard benchmarks for specific configurations.

Need for flexible data representation

- *e.g.* unstructured, incompletely structured, variable structure...

Motivations behind NoSQL (2)

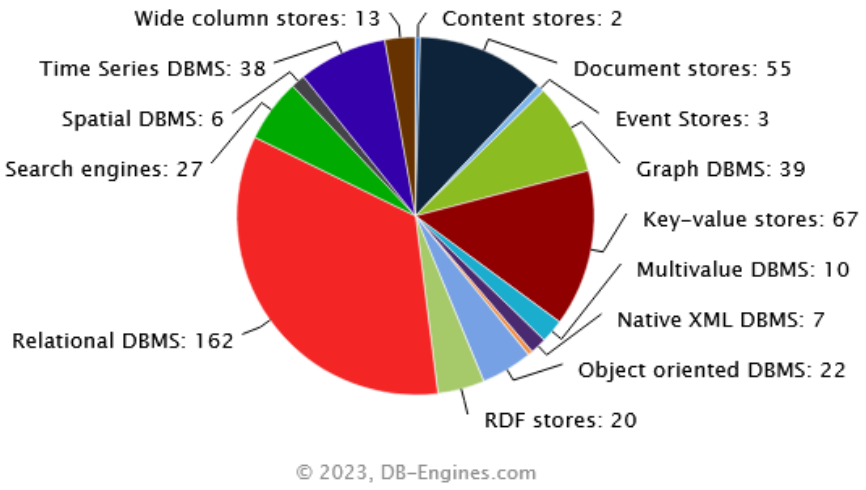
Horizontal Scalability

- cost reduction using commodity hardware
  e.g. exponential explosion of data,
- complexity/cost of relational data clusters
  e.g. sharding
- engineering and maintenance costs related to distributed and partitioning of centralized data models
- cloud computing infrastructures

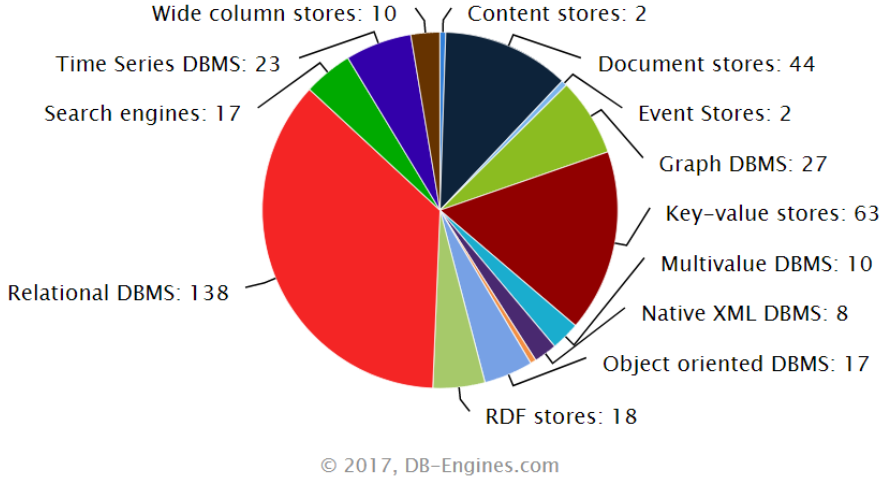
Some Statistics

Source: http://db-engines.com/en/ranking_categories

Number of Systems (Jan 2023)



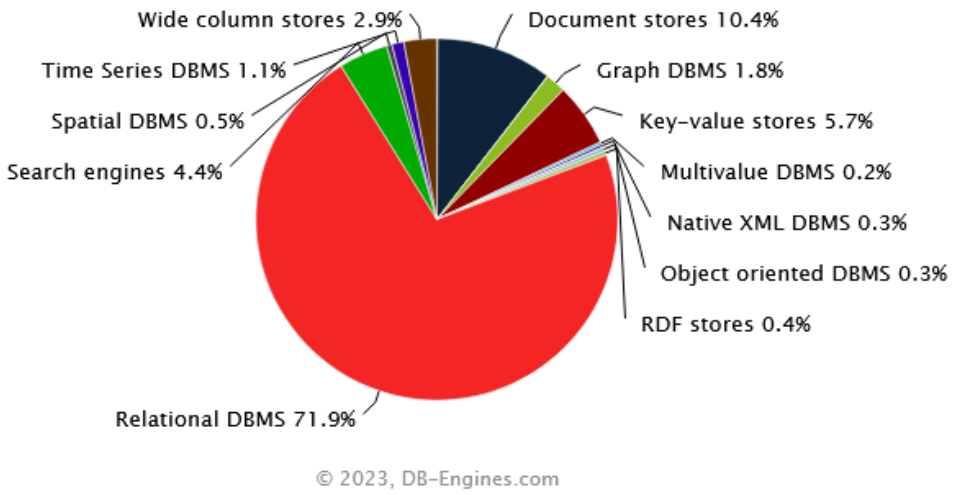
Number of Systems (Dec 2017)



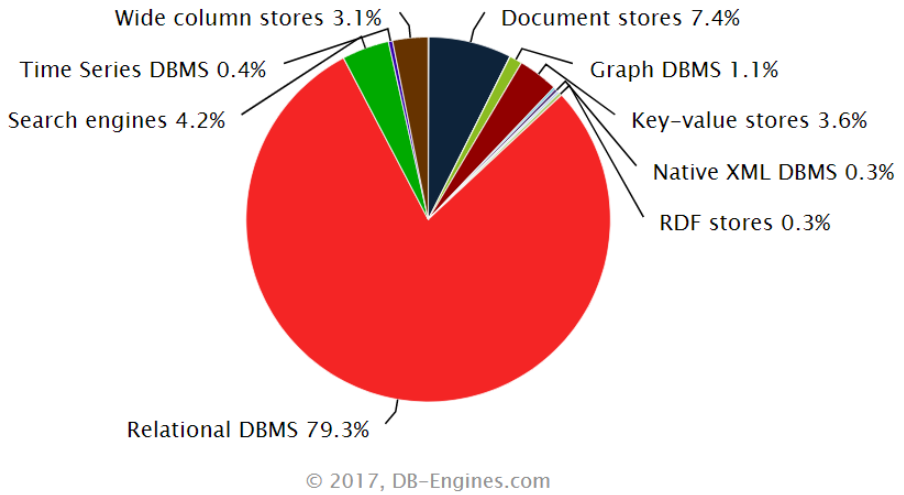
Some Statistics

Source: http://db-engines.com/en/ranking_categories

Popularity (Jan 2023)

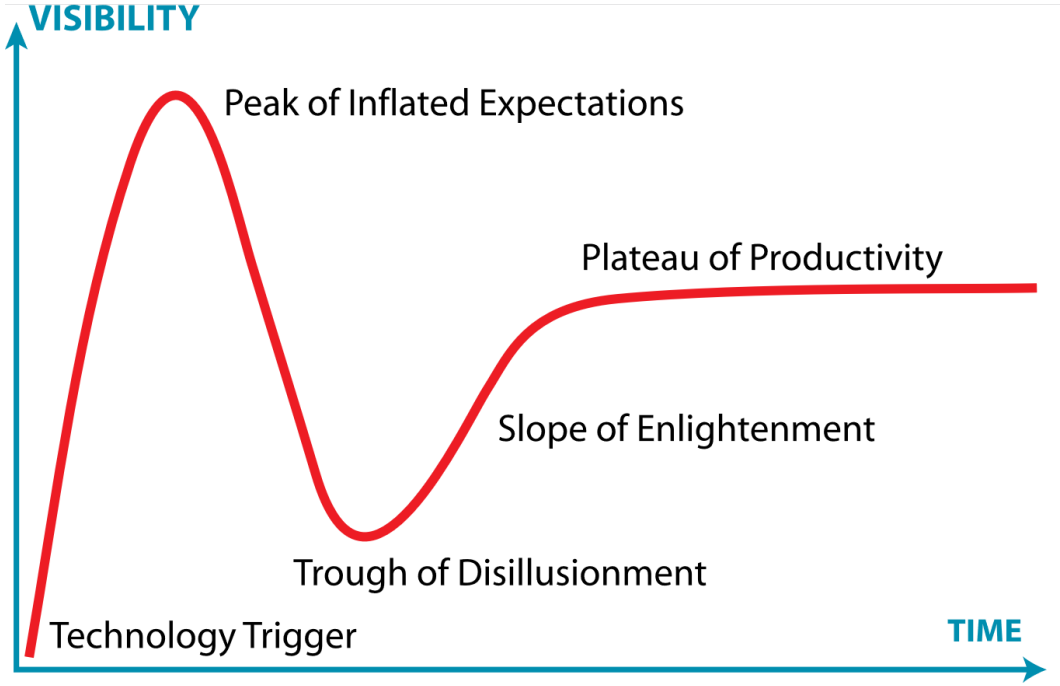


Popularity (Dec 2017)



UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

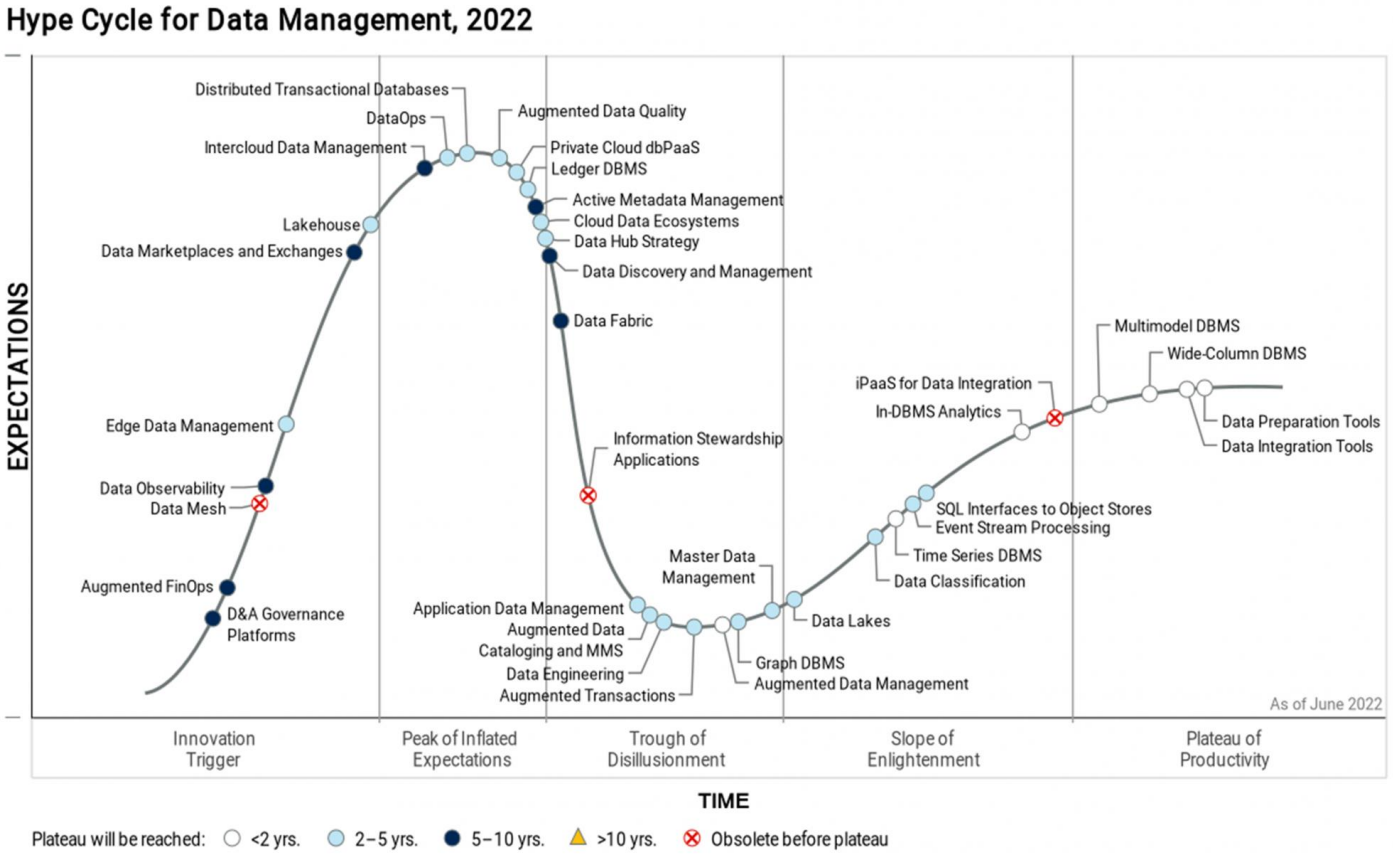
The NoSQL “Hype”



Hype Cycle: Gartner, Inc. (1995)

Source Wikipedia: http://en.wikipedia.org/w/index.php?title=File:Gartner_Hype_Cycle.svg&page=1

The NoSQL “Hype”



Source: Gartner (June 2022)

Outline

SQL & ACID vs. Scalability

The NoSQL « hype »

Looking into NoSQL

- Invariants
- Key-Value Stores
- Document Databases
- Column-Oriented Databases
- Graph Databases

Invariants

The CAP Theorem

ACID vs. BASE

Basic Techniques

- Versioning
- Hashing
- Storage
- Querying

The CAP Theorem

Consistency:

execution of an operation is guaranteed to leave the system in a consistent/coherent state;

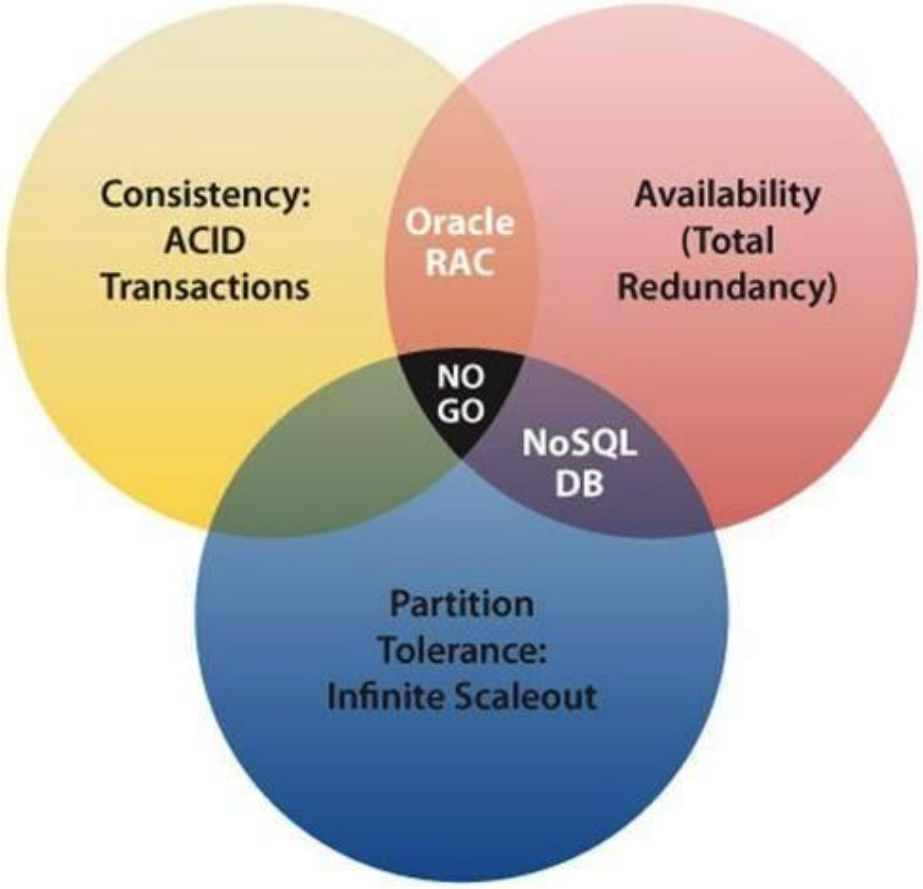
Availability:

every request is guaranteed to be answered with either success/fail return;

Partition Tolerance:

loss of network links between nodes, node failures or other partial availability incidents do not affect continuation of service;

The CAP Theorem



Source: Guy Harrison, http://dbpedias.com/wiki/NoSQL:Consistency_Models_in_Non-Relational_Databases

ACID *vs*. BASE

- **A**tomicity
  - guaranteed success/fail with rollback
- **C**onsistency
  - database is in guaranteed defined state before and after transaction
- **I**solation
  - concurrent transactions do not interfere
- **D**urability
  - executed transactions have persistent effects

- **B**asically **A**vailable
  - the system provides best effort (possibly partial) availability
- **S**oft-state
  - the overall consistency state of the system may evolve over time and persistence is not guaranteed
- **E**ventually Consistent
  - given sufficient time, the system will converge to a consistent state

BASE Caveat

BASE-Systems are defined by what they are not

- Not quite available
- Not quite consistent
- Not always persistent

→ Not All BASE-Systems are equivalent!

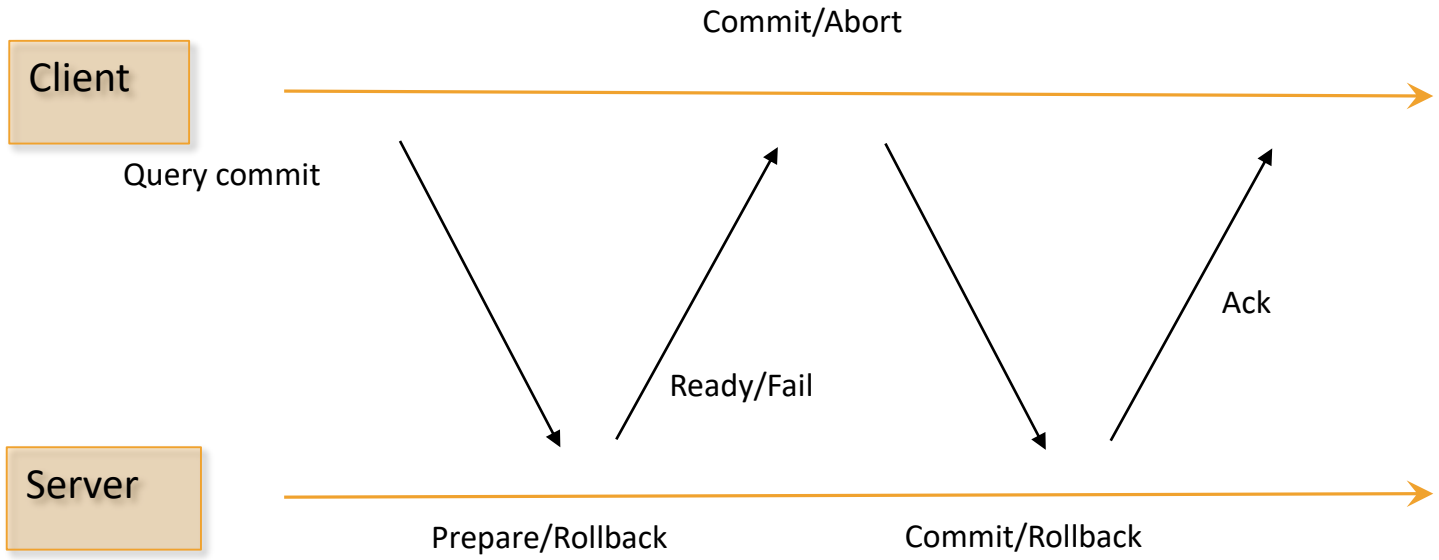
- Levels of Consistency
- Handling of Inconsistency
- Read/Write Availability
- Persistency Levels vs. TTL

Consistency

Why thinking about consistency anyway? Is there a problem with consistency?

- Consistency is the “C” in ACID
- Is consistency always needed?
- How is consistency guaranteed?
- What is the cost of maintaining consistency?

Consistency and Distributed Two Phase Commit



The two generals' problem
(Leslie Lamport. "Solved Problems, Unsolved Problems and Non-Problems in Concurrency" 1983. p. 8.)

Consistency

Strict consistency

Any read on a data item x returns a value corresponding to the result of the most recent write on x.

Weak consistency

Data is made consistent after explicit synchronization operations are called.

Eventual consistency

Data is made consistent over time, but can temporarily be inconsistent

Data-centred consistency

Client-centred consistency

Data-centred Consistency

Sequential consistency

Linearisable consistency

Causal consistency

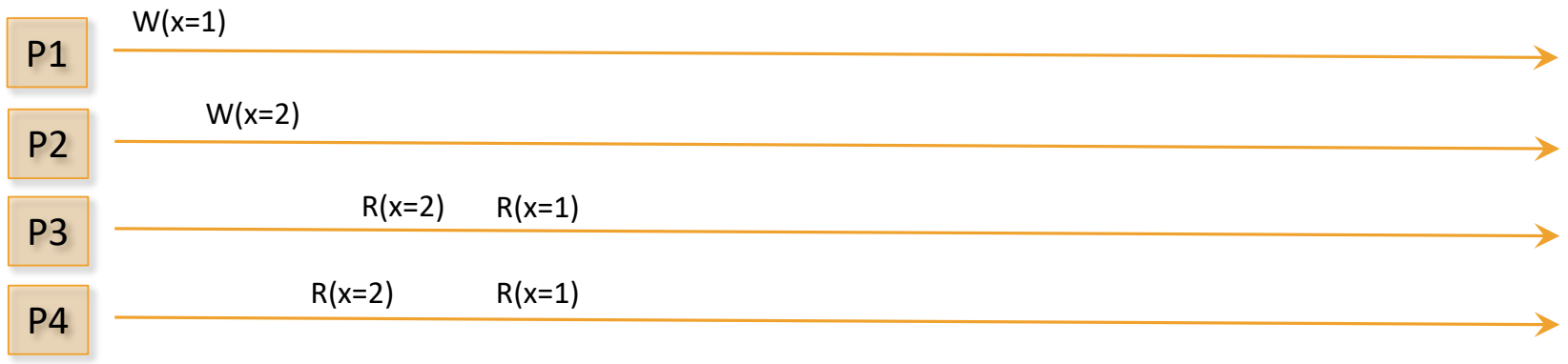
FIFO consistency

Data-centred Consistency (sequential)

Sequential consistency

the **result of any execution is the same** as if the read and write operations by all processes were **executed in some sequential order**.

Overall order may not respect absolute timing.

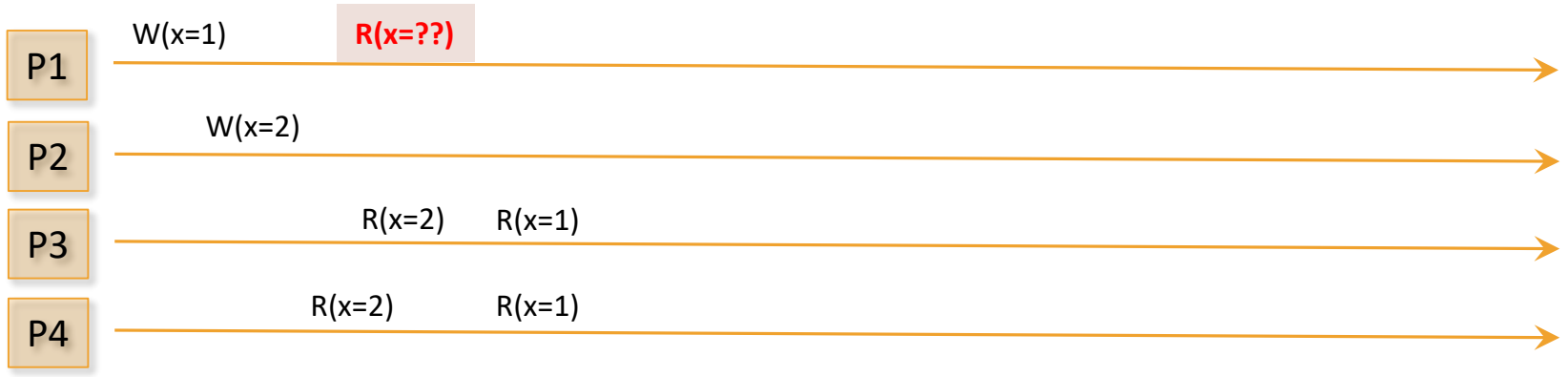


Data-centred Consistency (sequential)

Sequential consistency

the **result of any execution is the same** as if the read and write operations by all processes were **executed in some sequential order**.

Overall order may not respect absolute timing.

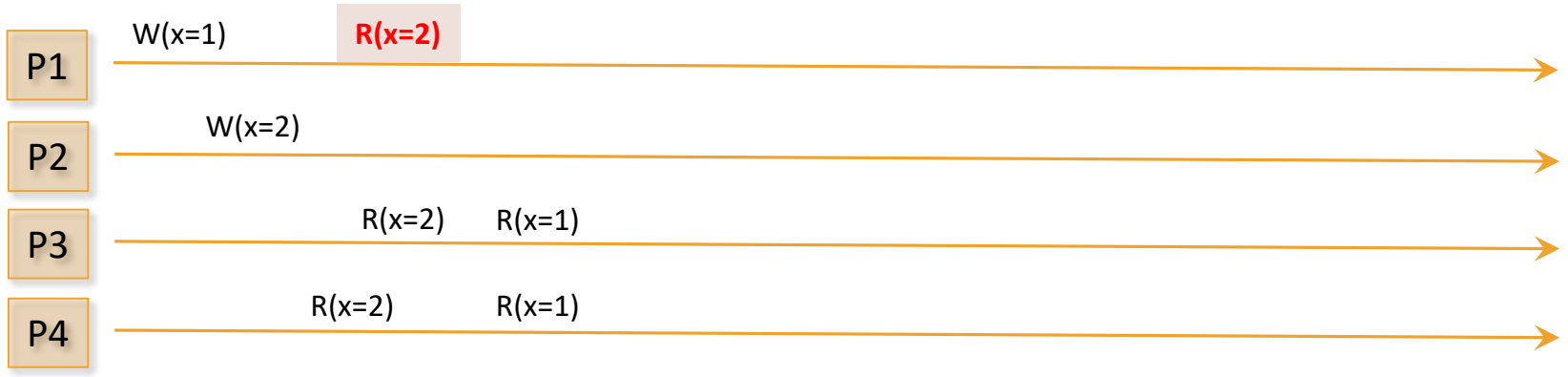


Data-centred Consistency (sequential)

Sequential consistency

the **result of any execution is the same** as if the read and write operations by all processes were **executed in some sequential order**.

Overall order may not respect absolute timing.

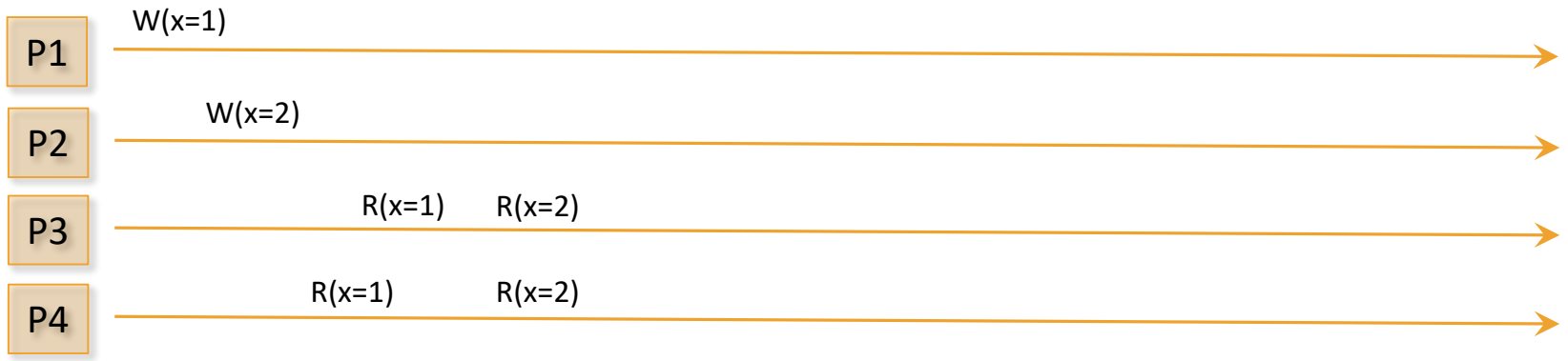


UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Data-centred Consistency (linearisable)

Linearisable consistency

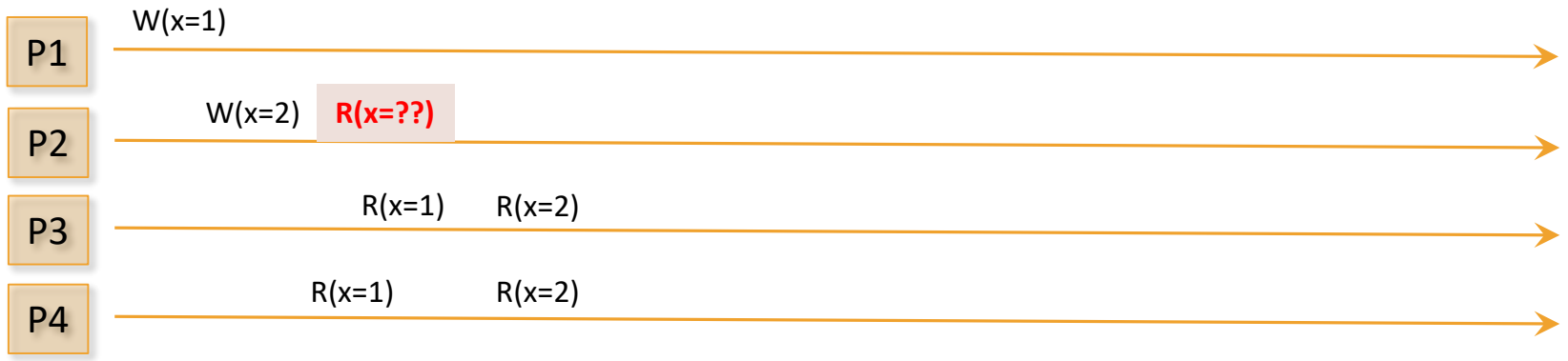
- weaker than strict consistency, stronger than sequential consistency.
- operations are assumed to receive a timestamp with a global available clock that is loosely synchronized.
- Overall order respects relative timing.



Data-centred Consistency (linearisable)

Linearisable consistency

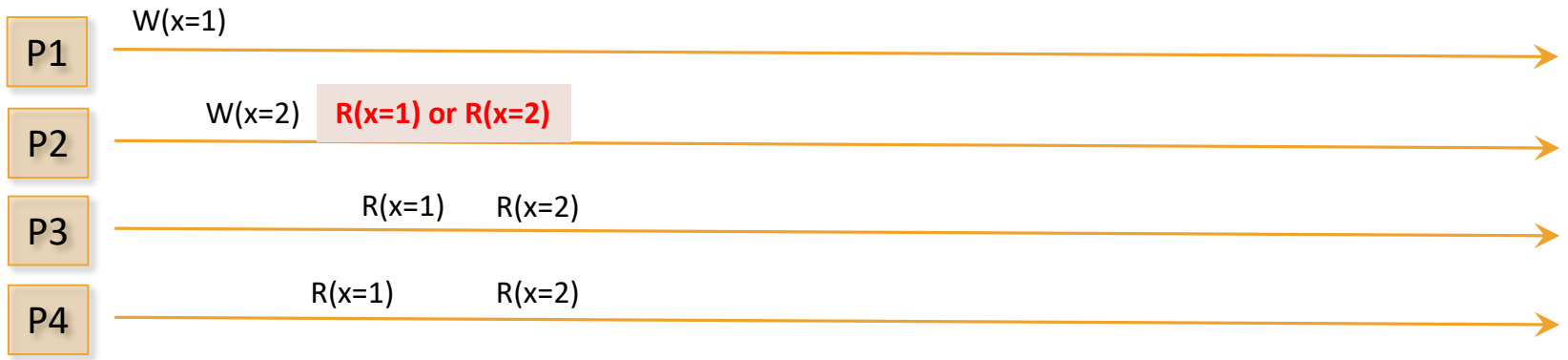
- weaker than strict consistency, stronger than sequential consistency.
- operations are assumed to receive a timestamp with a global available clock that is loosely synchronized.
- Overall order respects relative timing.



Data-centred Consistency (linearisable)

Linearisable consistency

- weaker than strict consistency, stronger than sequential consistency.
- operations are assumed to receive a timestamp with a global available clock that is loosely synchronized.
- Overall order respects relative timing.



Data-centered Consistency (causal)

Causal consistency

- Writes that are potentially causally related must be seen by all processes in the same order.
- Concurrent on causally unrelated writes may be seen in a different order on different machines.

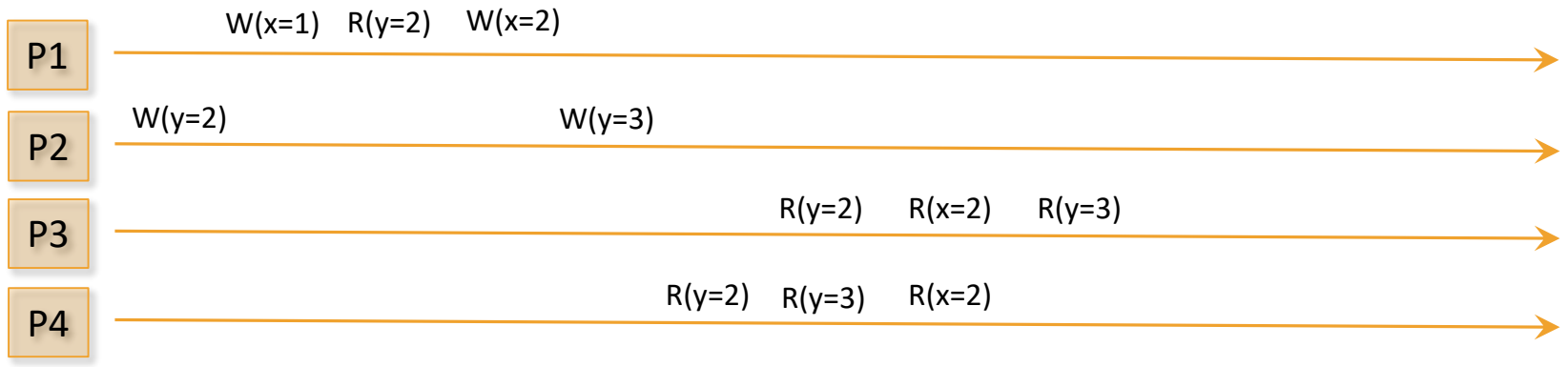
Example :

Comments on posted articles/chats

Data-centred Consistency (causal)

Causal consistency

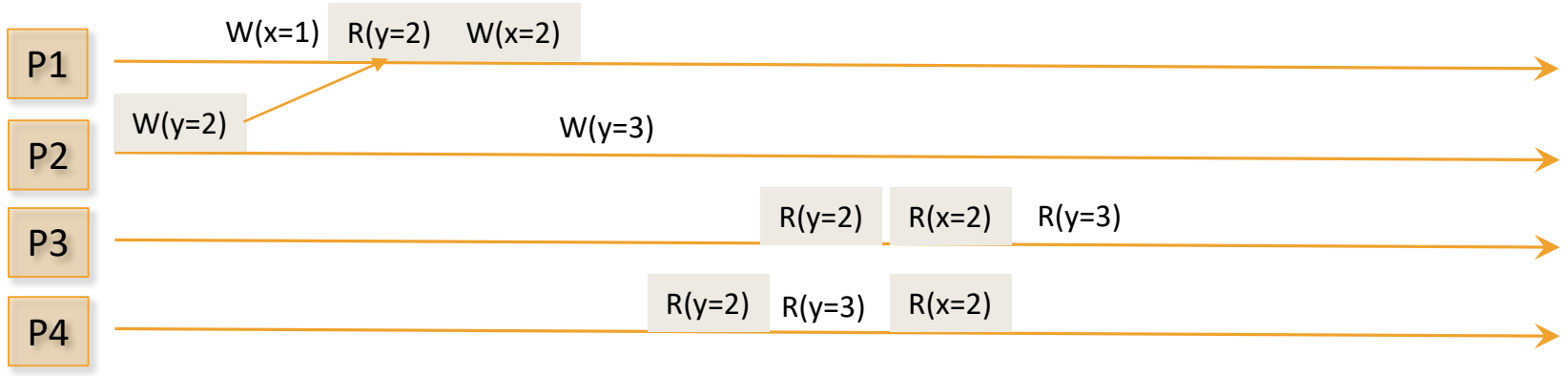
- Writes that are potentially causally related must be seen by all processes in the same order.
- Concurrent on causally unrelated writes may be seen in a different order on different machines.



Data-centred Consistency (causal)

Causal consistency

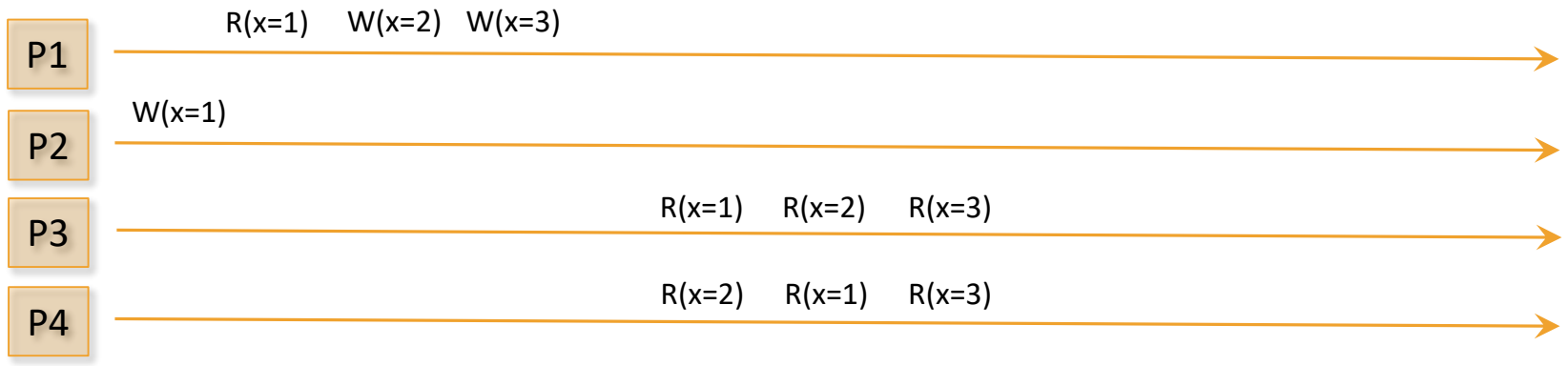
- Writes that are potentially causally related must be seen by all processes in the same order.
- Concurrent on causally unrelated writes may be seen in a different order on different machines.



Data-centred Consistency (FIFO)

FIFO consistency

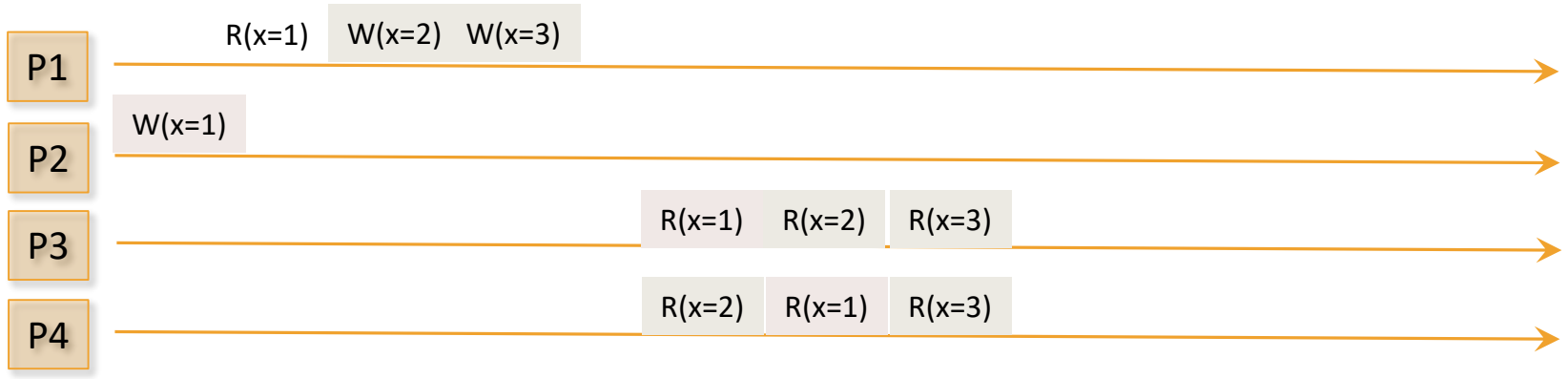
Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.



Data-centred Consistency (FIFO)

FIFO consistency

Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.



client-centred Consistency

Monotonic Reads consistency

Clients will only see more updated versions of the data in future requests

(e.g. email reads)

Monotonic Writes consistency

A write operation by a client on a data item is completed before any successive write operation on that item by the same client

(e.g. software versioning system)

Read Your Own Writes (RYOW) consistency

Client sees his own updates immediately but not necessarily those from others

Writes follow Reads consistency

A write operation by a client on a data item x following a read operation on x by the same client is guaranteed to take place on the same or a more recent value of x that was read.

Session Consistency

Client sees his own writes within a session scope (usually bound to one server)

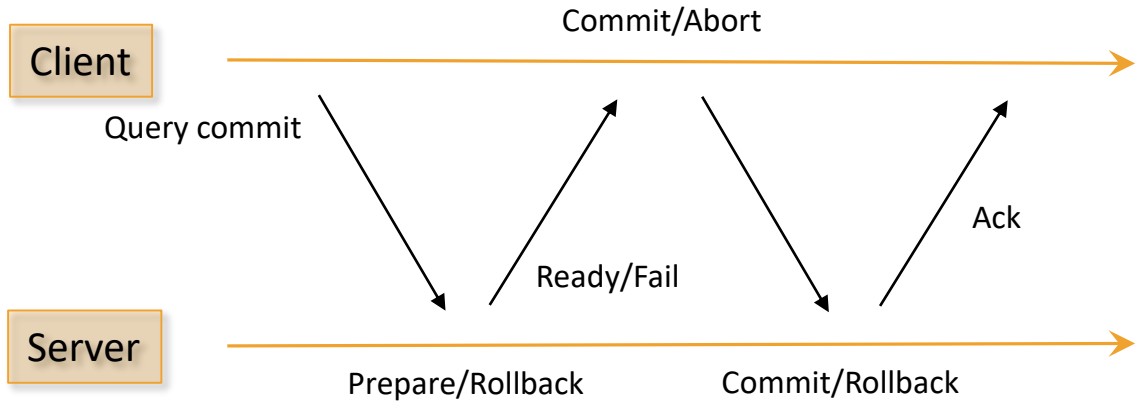
Consistency protocols

Primary-based protocols

Replicated-write protocols

Recall:

Two Phase Commit



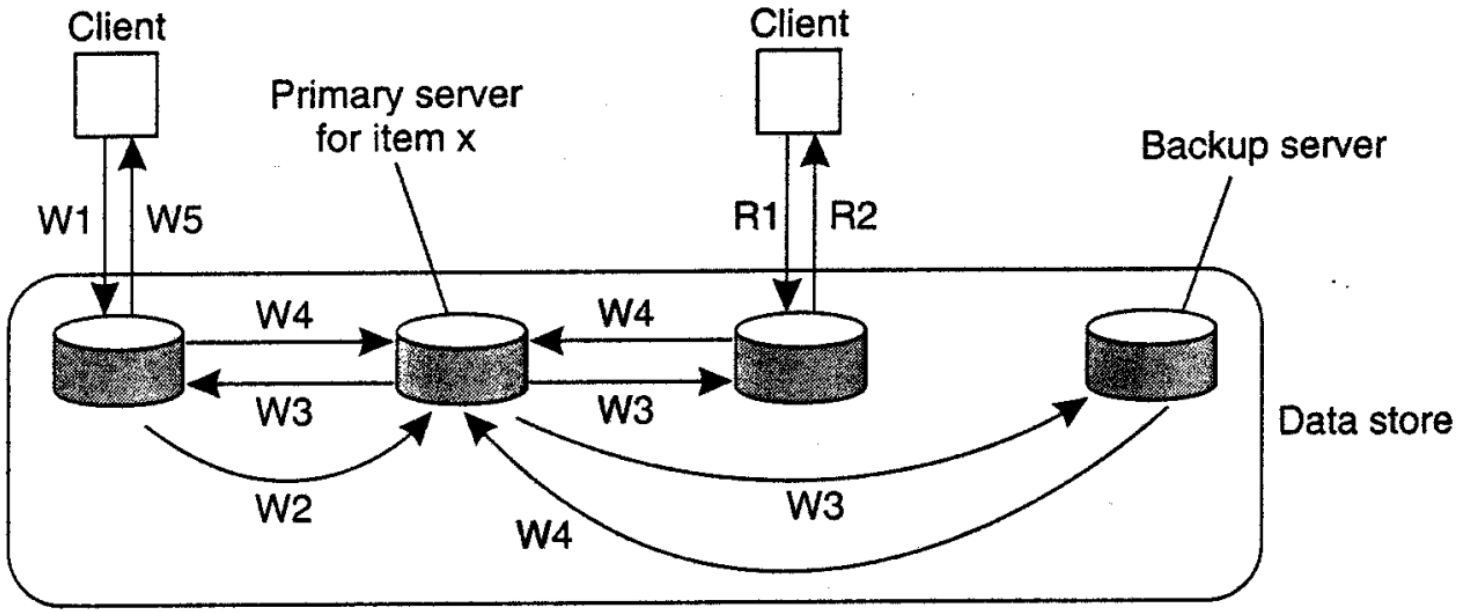
Primary based protocols

Each data item has a primary copy

Remote write

Local write

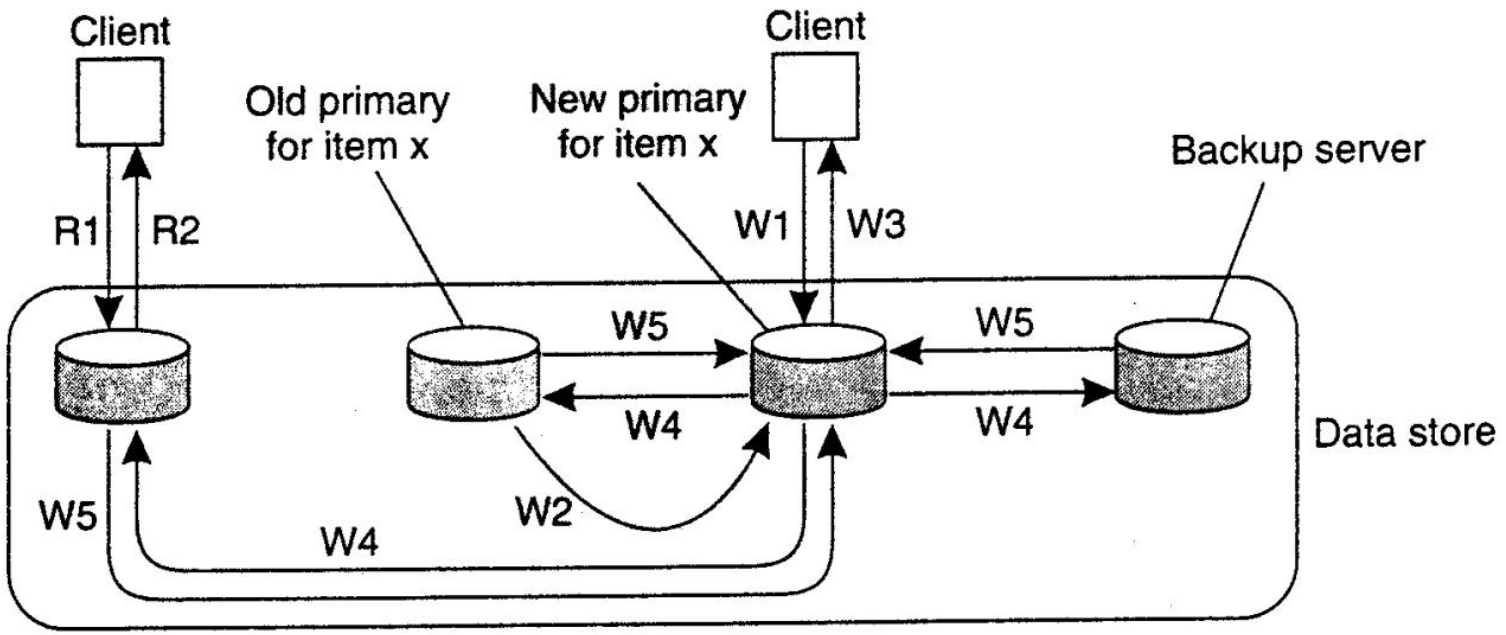
Primary based protocols



W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Primary based protocols



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

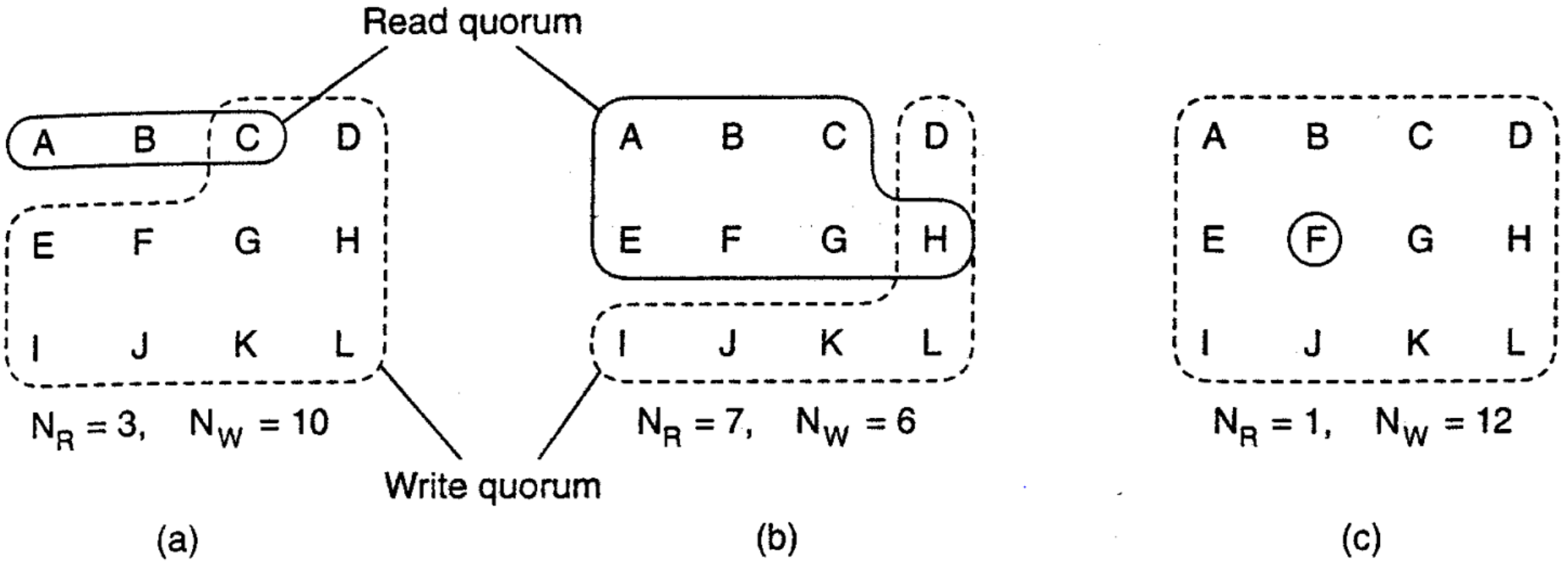
Replicated-write based protocols

Data is written simultaneously on multiple replicas

Active replication

Quorum based replication

Quorum based protocols



References and Further Reading

- Distributed Systems: Pearson New International Edition: Principles and Paradigms (2nd Edition) Andrew S. Tanenbaum, Maarten Van Steen, ISBN 9781292025520
- CAP Theorem: Eric Brewer, "Towards Robust Distributed Systems"
- Christof Strauch, "NoSQL Databases"

Assignment

Compare Oracle RAC and IBM DB2

- http://www.oracle.com/technetwork/products/clustering/rac-ds-12c-1898881.pdf
- https://public.dhe.ibm.com/common/ssi/ecm/im/en/iml14468usen/analytics-analytics-platform-im-other-papers-and-reports-iml14468usen-20170804.pdf

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE