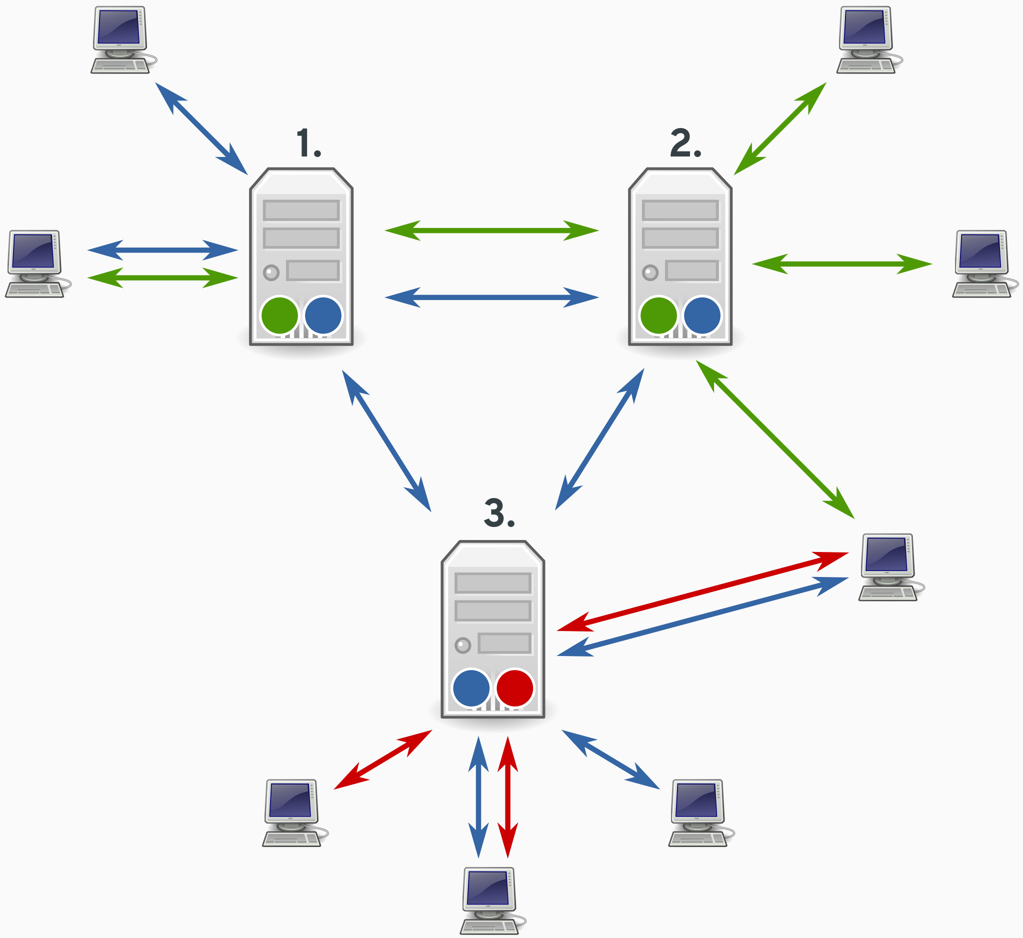
CONSISTENCY IN DISTRIBUTED SYSTEMS

ONDRA CHALOUPKA

http://narayana.io, @_chalda

DISTRIBUTED SYSTEM



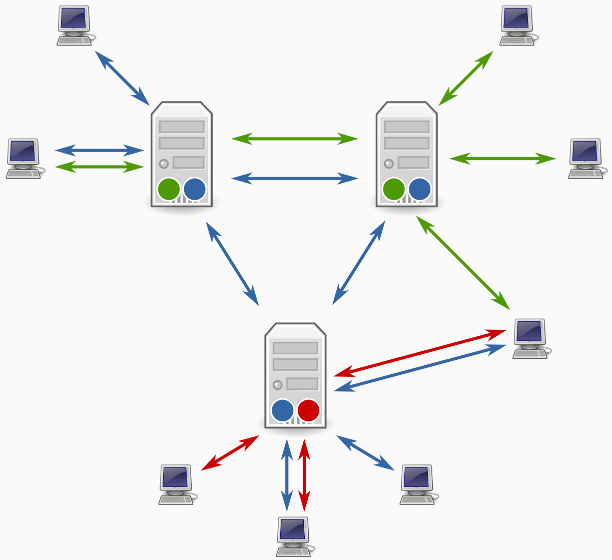
https://en.wikipedia.org/wiki/Usenet

ISN'T IT THE DATABASE DOING THAT WORK?

SINGLE-NODE DATABASE

- Data update on one node
  - consistent on client data access
- Bigger data volume needs better HW
  - vertical scalability could be expensive
- Single point of failure

MULTI-NODE DATABASE

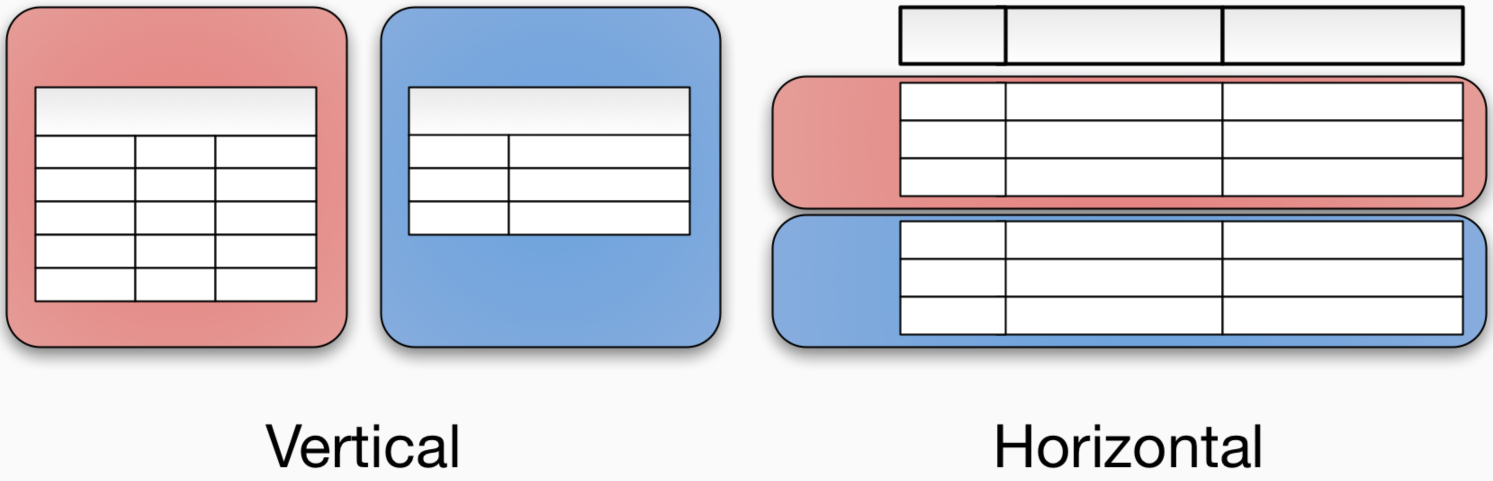


- No-single point of failure
- Scaling the load over multiple nodes
  - may accommodate bigger data volume
- communication overhead
  -> multiple nodes has to agree on one particular value to be saved

MULTI-NODE DATABASE

DISTRIBUTE YOUR DATA - PARTITIONING

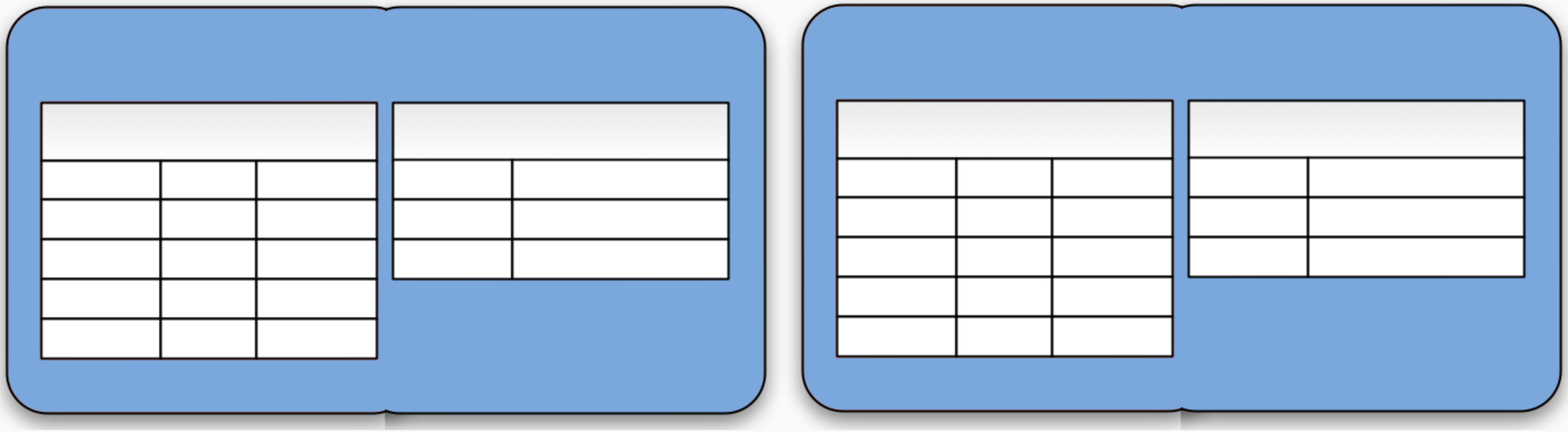
- Partitioning
- Sharding (vertical partitioning)



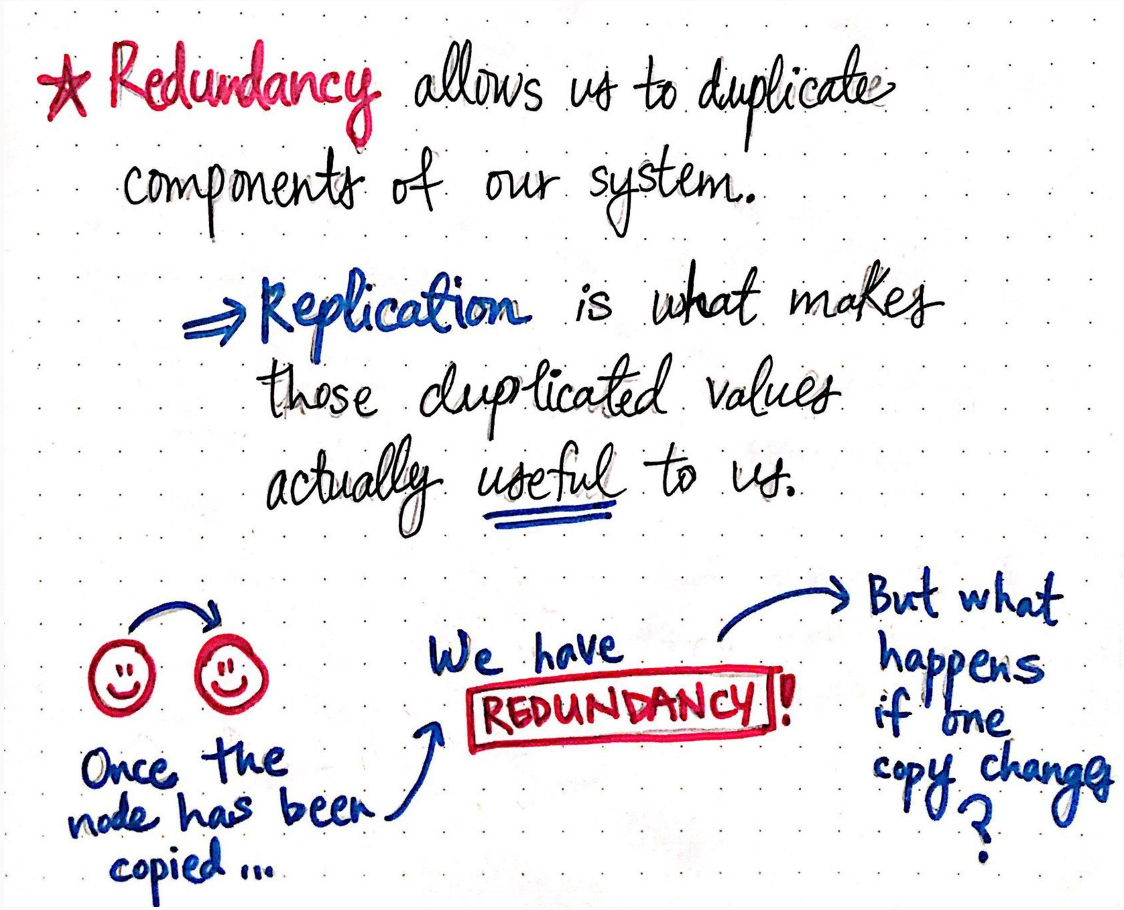
https://medium.com/@jeeyoungk/how-sharding-works-b4dec46b3f6

MULTI-NODE DATABASE

DISTRIBUTE YOUR DATA - REPLICATION



MULTI-NODE DATABASE

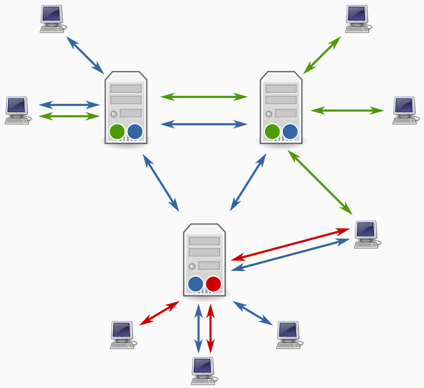


https://medium.com/baseds/redundancy-and-replication-duplicating-in-a-distributed-system-7ab4322d7378

SINGLE VS. MULTI NODES DATABASE

Single-node database

- Contention for reads and writes
  -> performance loss
  - -> **ACID isolation levels**



Multi-node database

- Contentions on parallel updates
  -> performance loss
  - -> **Consistency levels**

CONSISTENCY ≠ CONSISTENCY

ACID consistency talks about consistent data from application perspective.

CAP consistency says that multiple clients accessing database can see the same data.

CAP consistency =~ ACID Isolation

CAP

It's impossible to build an implementation of **read-write storage** in an **asynchronous network** that satisfies all of the following three properties:

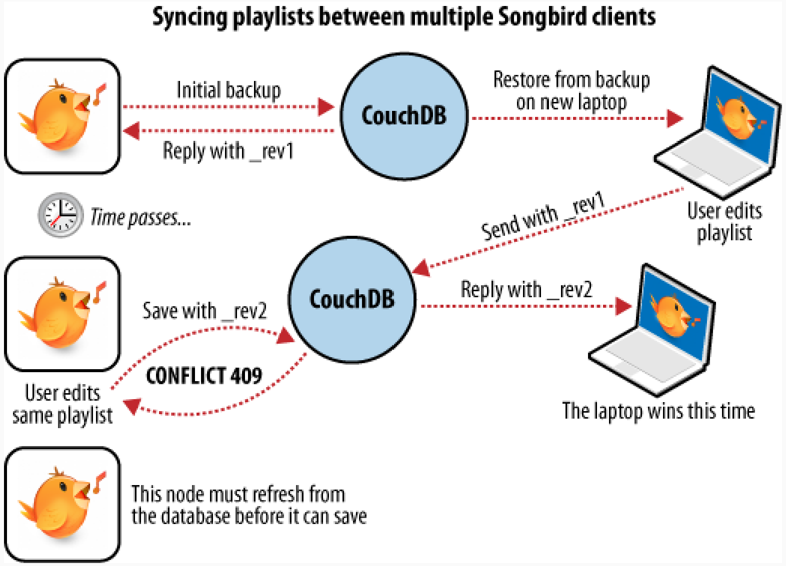
- **Availability** - will a request made to the data store always eventually complete?
- **Consistency** - will all executions of reads and writes seen by all nodes be *atomic* or *linearizably* consistent?
- **Partition tolerance** - the network is allowed to drop any messages.

https://www.the-paper-trail.org/page/cap-faq/

CONSISTENCY…

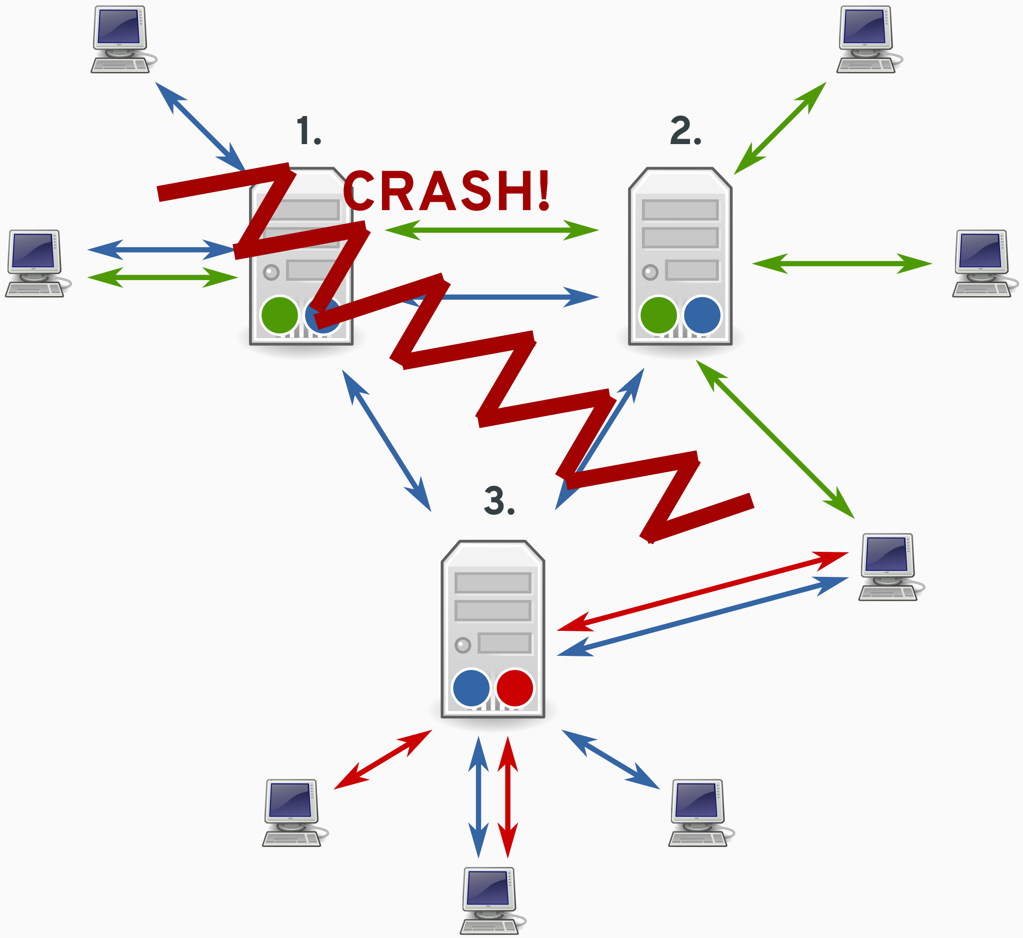
It's all about parallel processing

- All database clients see the same data, even with concurrent updates.



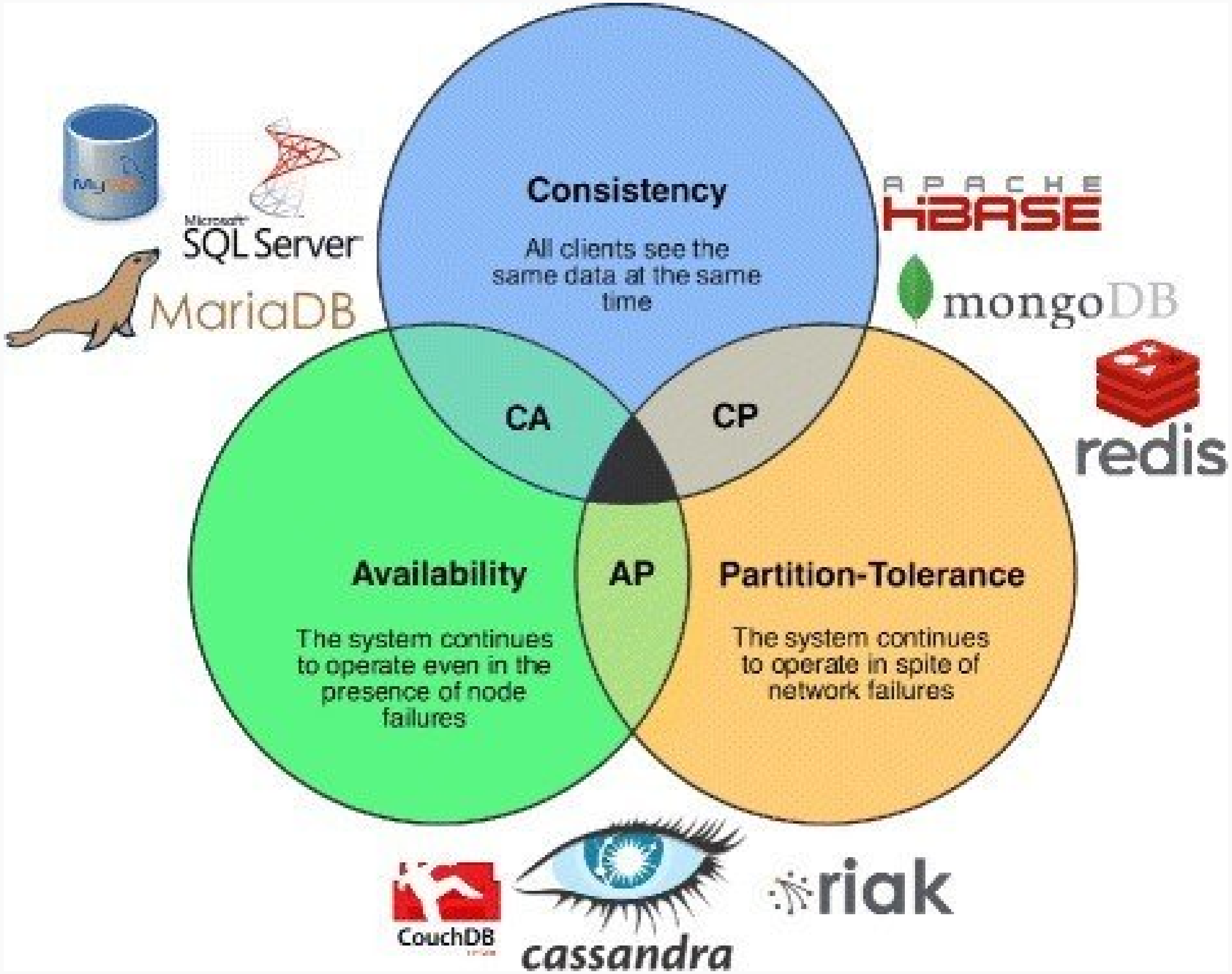
https://www.oreilly.com/library/view/couchdb-the-definitive/9780596158156/ch02.html

PARTITION IN SYSTEM



https://en.wikipedia.org/wiki/Usenet

CAP SYSTEMS


Consistency
All clients see the same data at the same time
CA
CP
AP
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak

CAP CRITIQUE

CAP theorem does reflect a **real world database** and does not take into account **latency**

Daniel Abadi:

"**CAP** should really be **PACELC** --- if there is a partition (**P**) how does the system tradeoff between availability and consistency (**A** and **C**); else (**E**) when the system is running as normal in the absence of partitions, how does the system tradeoff between latency (**L**) and consistency (**C**)?"

- https://martin.kleppmann.com/2015/05/11/please-stop-calling-databases-cp-or-ap.html
- http://dbmsmusings.blogspot.com/2010/04/problems-with-cap-and-yahoos-little.html

CONSISTENCY AND SINGLE-NODE DATABASE

What about consistency in the world of the old good SQL databases like ***MySQL***, ***PostgreSQL***, ***Oracle*** etc.?

CA from **CAP** perspective

- Consistency
- Availability

...and utilizes **ACID** transactions

SINGLE-NODE DATABASE

ACID ISOLATION LEVELS

Updates on multiple records

Read phenomena

- Dirty reads
- Non-repeatable reads
- Phantom reads

Isolation levels

- Serializable
- Snapshot isolation
- Repeatable reads
- Read committed
- Read uncommitted

https://medium.com/abrai/demystifying-consistency-and-isolation-for-a-distributed-systems-engineer-64a064c52f6e

SERIALIZABILITY

ACID ISOLATION LEVELS

Identifies data transactions as occurring serially, independent of one another, even though they may have occurred concurrently.

A schedule or list of transactions is deemed to be correct if they are serialized,

http://www.businessdictionary.com/definition/serializability.html

MULTI-NODE DATABASE

REPLICATED DATA CONSISTENCY EXPLAINED THROUGH BASEBALL

Strong Consistency	See all previous writes.
Eventual Consistency	See subset of previous writes.
Consistent Prefix	See initial sequence of writes.
Bounded Staleness	See all “old” writes.
Monotonic Reads	See increasing subset of writes.
Read My Writes	See all writes performed by reader.

Table 1. Six Consistency Guarantees

https://www.microsoft.com/en-us/research/wp-content/uploads/2011/10/ConsistencyAndBaseballReport.pdf

CONSISTENCY LEVELS

DATA CONSISTENCY THROUGH BASEBALL

Guarantee	Consistency	Performance	Availability
Strong Consistency	excellent	poor	poor
Eventual Consistency	poor	excellent	excellent
Consistent Prefix	okay	good	excellent
Bounded Staleness	good	okay	poor
Monotonic Reads	okay	good	good
Read My Writes	okay	okay	okay

Table 2. Consistency, Performance, and Availability Trade-offs

https://www.microsoft.com/en-us/research/wp-content/uploads/2011/10/ConsistencyAndBaseballReport.pdf

CONSISTENCY LEVELS

DATA CONSISTENCY THROUGH BASEBALL

	1	2	3	4	5	6	7	8	9	RUNS
Visitors	0	0	1	0	1	0	0			2
Home	1	0	1	1	0	2				5

Figure 3. The Line Score for this Sample Game

https://www.microsoft.com/en-us/research/wp-content/uploads/2011/10/ConsistencyAndBaseballReport.pdf

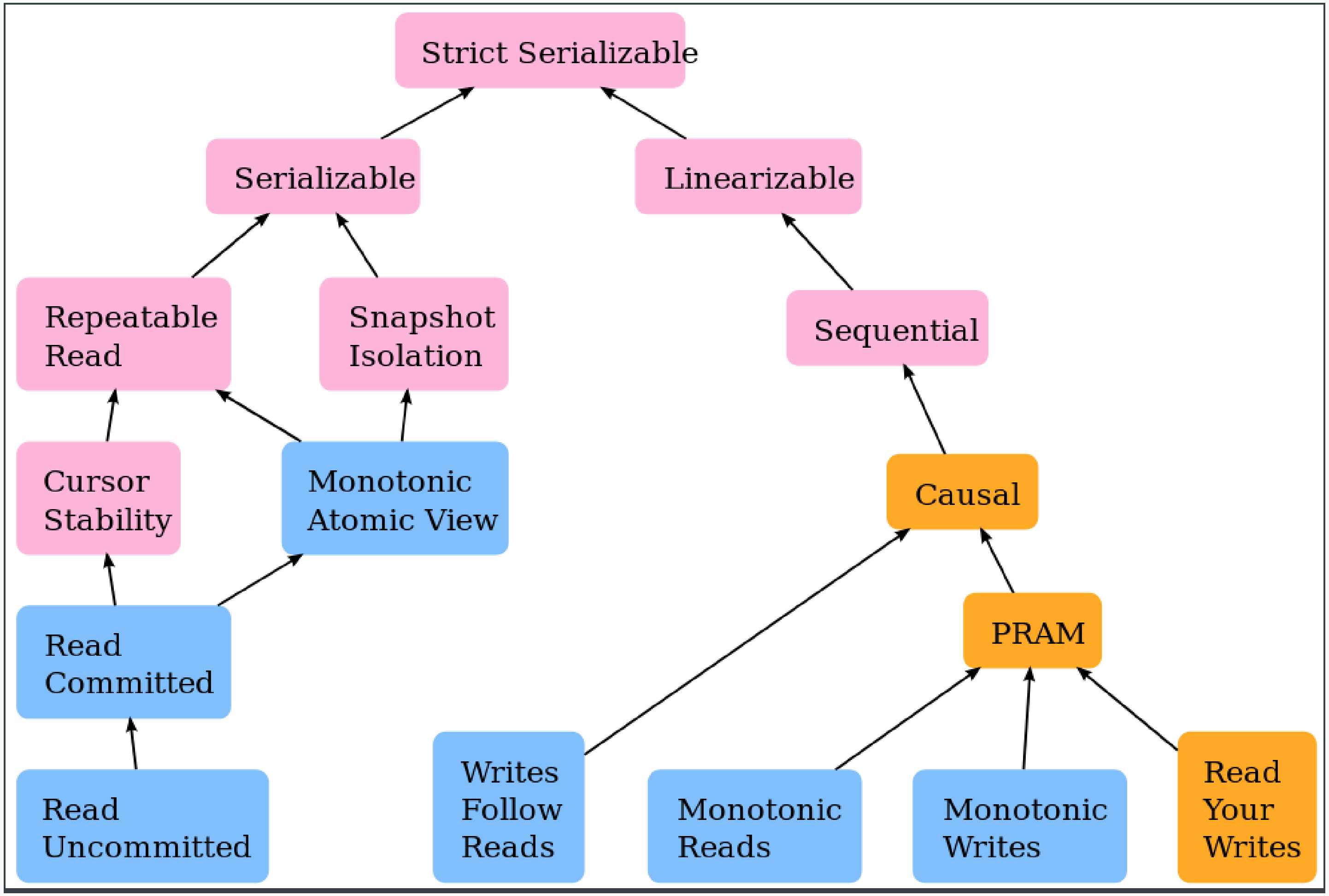
	1	2	3	4	5	6	7	8	9	RUNS
Visitors	0	0	1	0	1	0	0			2
Home	1	0	1	1	0	2				5

Figure 3. The Line Score for this Sample Game

Strong Consistency	2-5
Eventual Consistency	0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5
Consistent Prefix	0-0, 0-1, 1-1, 1-2, 1-3, 2-3, 2-4, 2-5
Bounded Staleness	scores that are at most one inning out-of-date: 2-3, 2-4, 2-5
Monotonic Reads	after reading 1-3: 1-3, 1-4, 1-5, 2-3, 2-4, 2-5
Read My Writes	for the writer: 2-5 for anyone other than the writer: 0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5

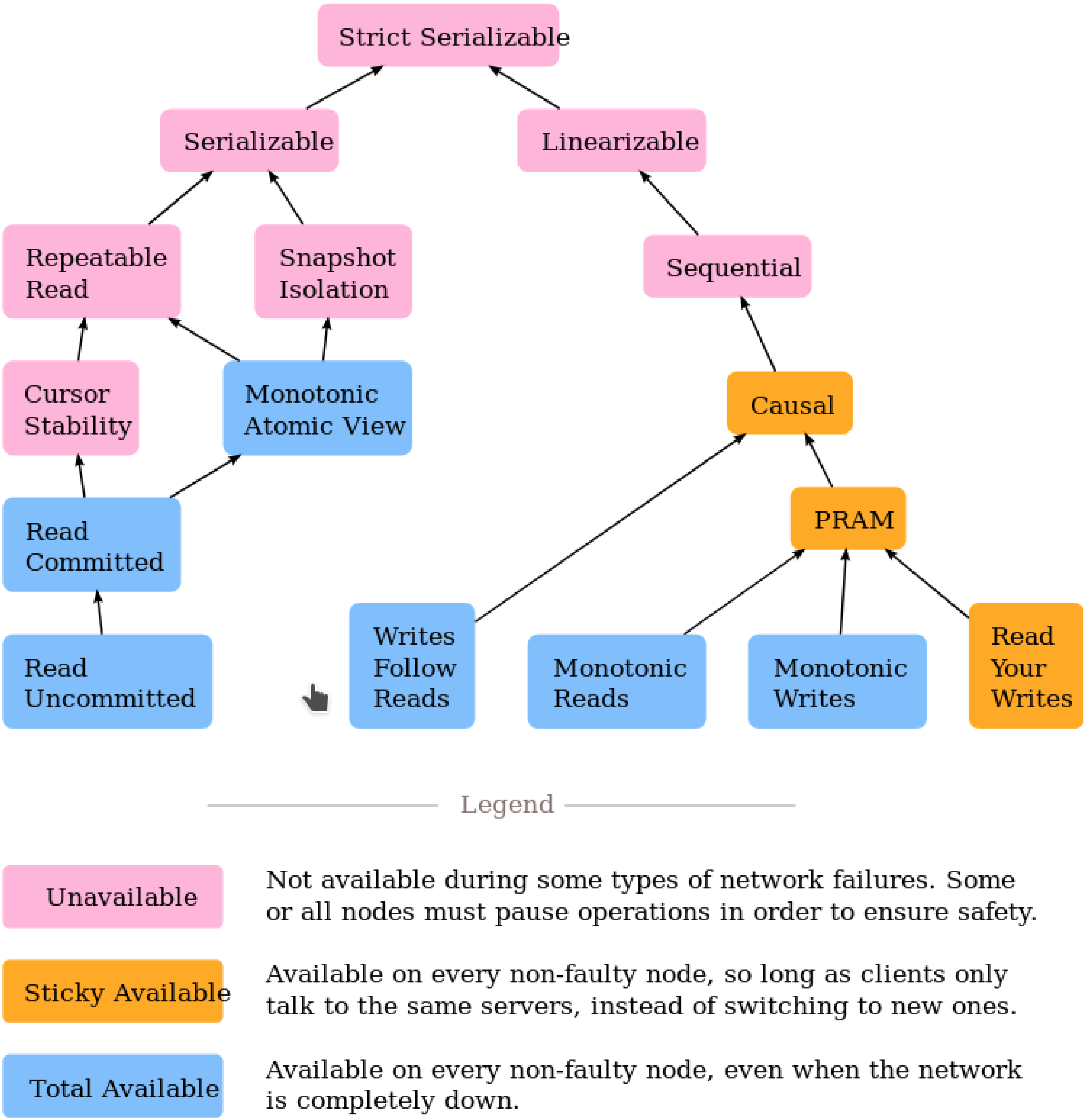
Table 3. Possible Scores Read for Each Consistency Guarantee

CONSISTENCY TYPES



https://jepsen.io/consistency

http://www.vldb.org/pvldb/vol7/p181-bailis.pdf


Strict Serializable
Serializable
Linearizable
Repeatable Read
Snapshot Isolation
Sequential
Cursor Stability
Monotonic Atomic View
Causal
PRAM
Read Committed
Read Uncommitted
Writes Follow Reads
Monotonic Reads
Monotonic Writes
Read Your Writes
Legend
Unavailable
Not available during some types of network failures. Some or all nodes must pause operations in order to ensure safety.
Sticky Available
Available on every non-faulty node, so long as clients only talk to the same servers, instead of switching to new ones.
Total Available
Available on every non-faulty node, even when the network is completely down.

EVERTHING IN SYNC

STRICT SERIALIZABILITY

Strict serializability is a *transactional* model: operations (usually termed “transactions”) can involve several primitive operations performed in order. Strict serializability guarantees that operations take place *atomically*: a transaction’s sub-operations do not appear to interleave with sub-operations from other transactions.

https://jepsen.io/consistency

SINGLE OBJECT IN SYNC

LINEARIZABILITY

Linearizability is one of the strongest single-object consistency models, and implies that every operation appears to take place atomically, in some order, consistent with the real-time ordering of those operations: e.g., if operation A completes before operation B begins, then B should logically take effect after A.

https://jepsen.io/consistency

DEPENDENT ACTIONS IN SYNC

CAUSAL CONSISTENCY

Causal consistency captures the notion that causally-related operations should appear in the same order on all processes—though processes may disagree about the order of causally independent operations.

For example, consider a chat between three people, where Attiya asks “shall we have lunch?”, and Barbarella & Cyrus respond with “yes”, and “no”, respectively. Causal consistency allows Attiya to observe “lunch?”, “yes”, “no”; and Barbarella to observe “lunch?”, “no”, “yes”. However, no participant *ever* observes “yes” or “no” prior to the question “lunch?”.

https://jepsen.io/consistency

GET WHAT YOU WROTE

READ YOUR WRITES

Read your writes, also known as *read my writes*, requires that if a process performs a write *w*, then that same process performs a subsequent read *r*, then *r* must observe *w*'s effects.

https://jepsen.io/consistency


THANK YOU!