System Design Basics

Nitin Mishra

Decision Making

Points to consider while designing large scale systems:

1. **What are the different architectural pieces that can be used?**
2. **How do these pieces work with each other?**
3. **How can we best utilize these pieces: What are the right tradeoffs?**

Distributed System Architecture

Key Characteristics of Distributed System Architecture:

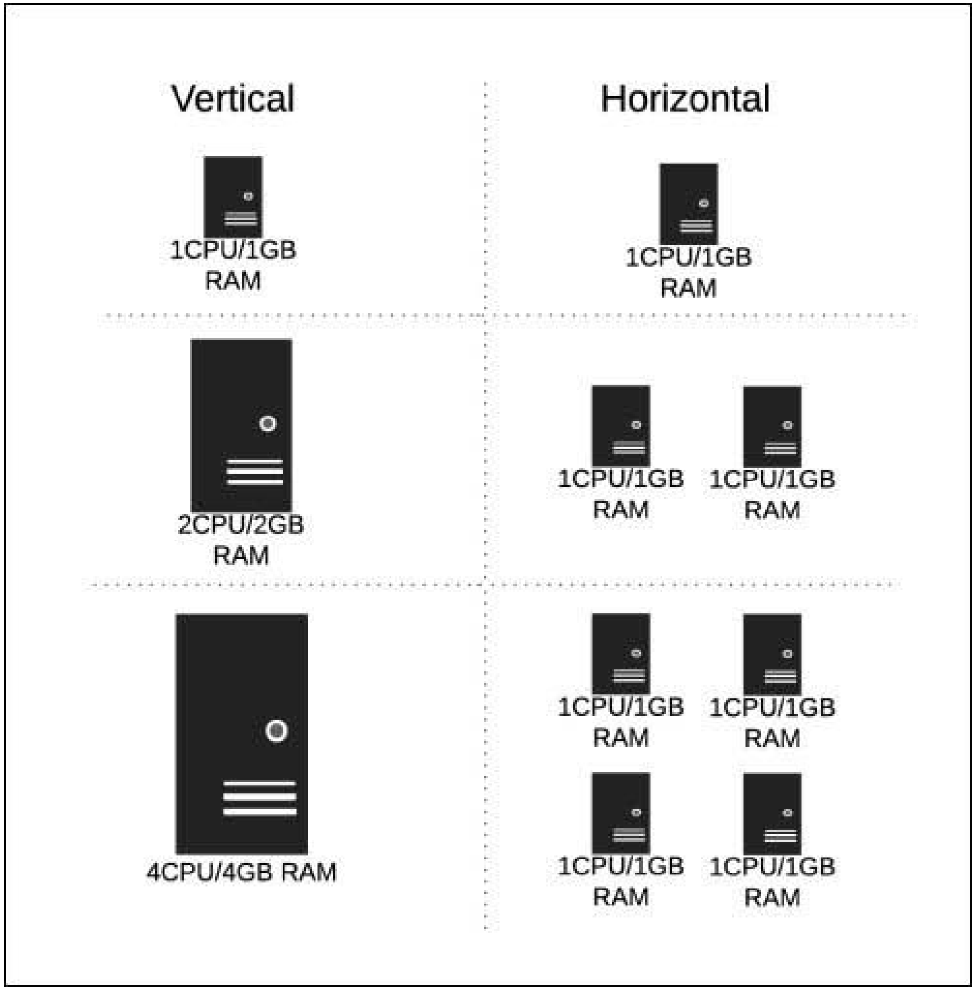
1. **Scalability**
2. **Reliability**
3. **Availability**
4. **Efficiency**
5. **Manageability**

Scalability

Scalability - The capability to perform well given increase in load and Availability of resources.

1. **Horizontal v/s Vertical Scaling**
2. **Vertical scaling involves downtime.**
3. **Vertical scaling is limited.**
4. **Horizontal scaling is unlimited.**

HS Example: Cassandra and MongoDB
VS Example: Mysql



Reliability

Reliability – The Probability a system will keep working without failure in a given period.

A system is considered reliable if it keeps delivering its services even when one or several of its software or hardware components fail.

A reliable distributed system achieves Reliability through redundancy of both the software components and data.

Obviously, redundancy has a cost and a reliable system has to pay that to achieve such resilience for services by eliminating every single point of failure.

Availability

Availability - The percentage of uptime in a given year.

If a system is reliable, it is available.

If it is available, it is not necessarily reliable.

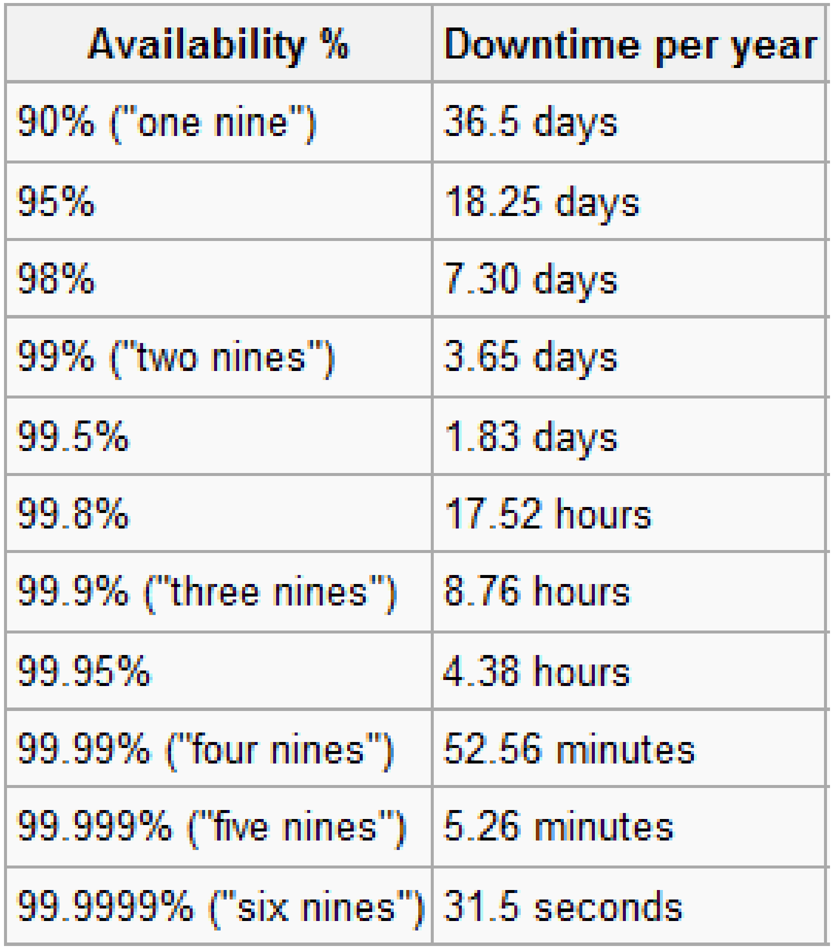
SLA (Service Level Agreement) – Downtime

One Nine – 90% Availability

Two Nine – 99% Availability

Three Nine – 99.9% Availability

Five Nine – 99.999% Availability



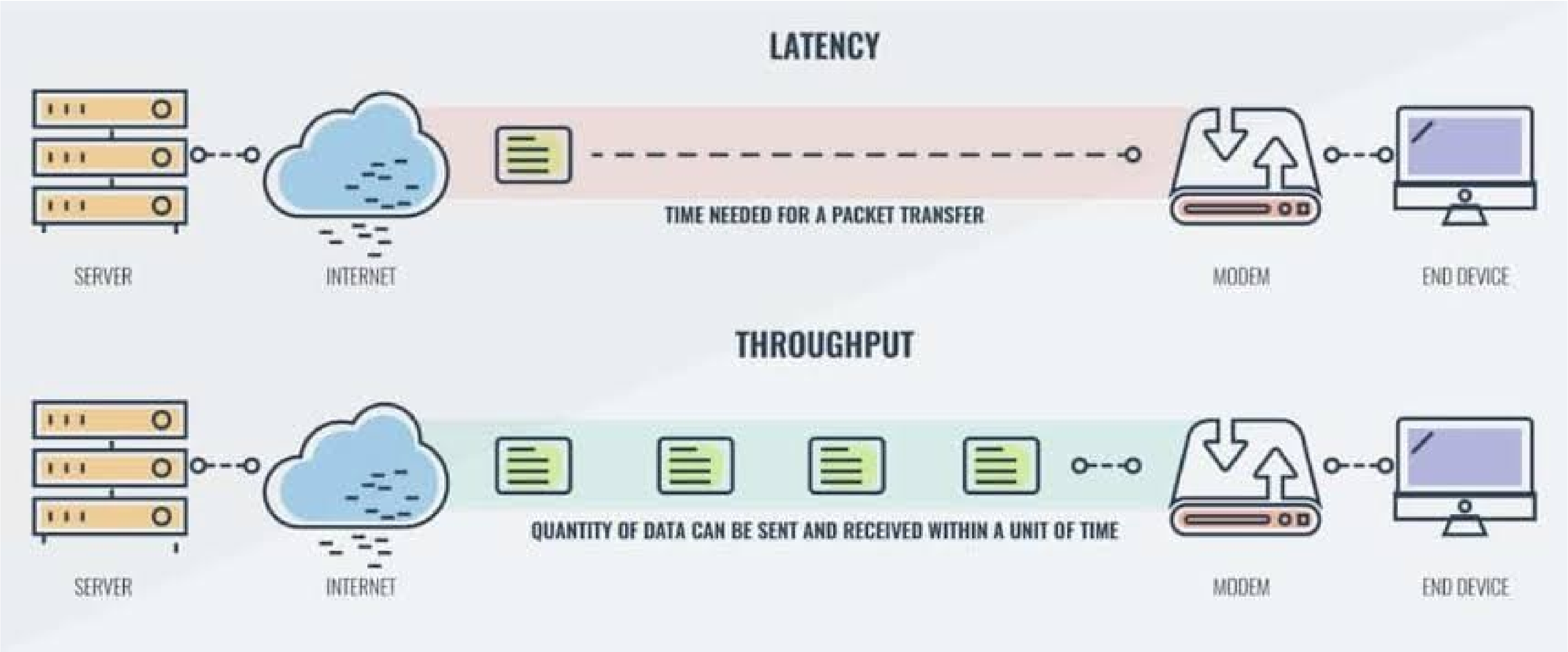
Availability %	Downtime per year
90% ("one nine")	36.5 days
95%	18.25 days
98%	7.30 days
99% ("two nines")	3.65 days
99.5%	1.83 days
99.8%	17.52 hours
99.9% ("three nines")	8.76 hours
99.95%	4.38 hours
99.99% ("four nines")	52.56 minutes
99.999% ("five nines")	5.26 minutes
99.9999% ("six nines")	31.5 seconds

Efficiency

Latency – Time required to perform some action

Throughput – No. of such actions executed per unit of time

Better Efficiency – Lower Latency & Higher Throughput


LATENCY
TIME NEEDED FOR A PACKET TRANSFER
SERVER
INTERNET
MODEM
END DEVICE
THROUGHPUT
QUANTITY OF DATA CAN BE SENT AND RECEIVED WITHIN A UNIT OF TIME
SERVER
INTERNET
MODEM
END DEVICE

Manageability

Serviceability or manageability is the simplicity and speed with which a system can be repaired or maintained.

If the time to fix a failed system increases, then availability will decrease.

Early detection of faults can decrease or avoid system downtime. For example, some enterprise systems can automatically call a service center (without human intervention) when the system experiences a system fault.

Load Balancer and Algorithms

LB spreads the traffic across a cluster of servers to improve responsiveness and availability of applications, websites or databases.

LB also keeps track of the Health Check status of all the resources while distributing requests.

Load Balancing helps in Horizontal scaling.

Algorithms for server selection:

1. Least Connection Method

2. Least Response Time Method

3. Least Bandwidth Method

4. Round Robin Method

5. Weighted Round Robin Method

5. Client IP Hashing Method

Caching

Useful when recently requested data is likely to be requested again.

Purpose is to increase data retrieval performance by reducing the need to access the underlying slower storage layer.

Application server cache is often placed on request layer node near to Front end for so that every subsequent request for same data can be served quickly.

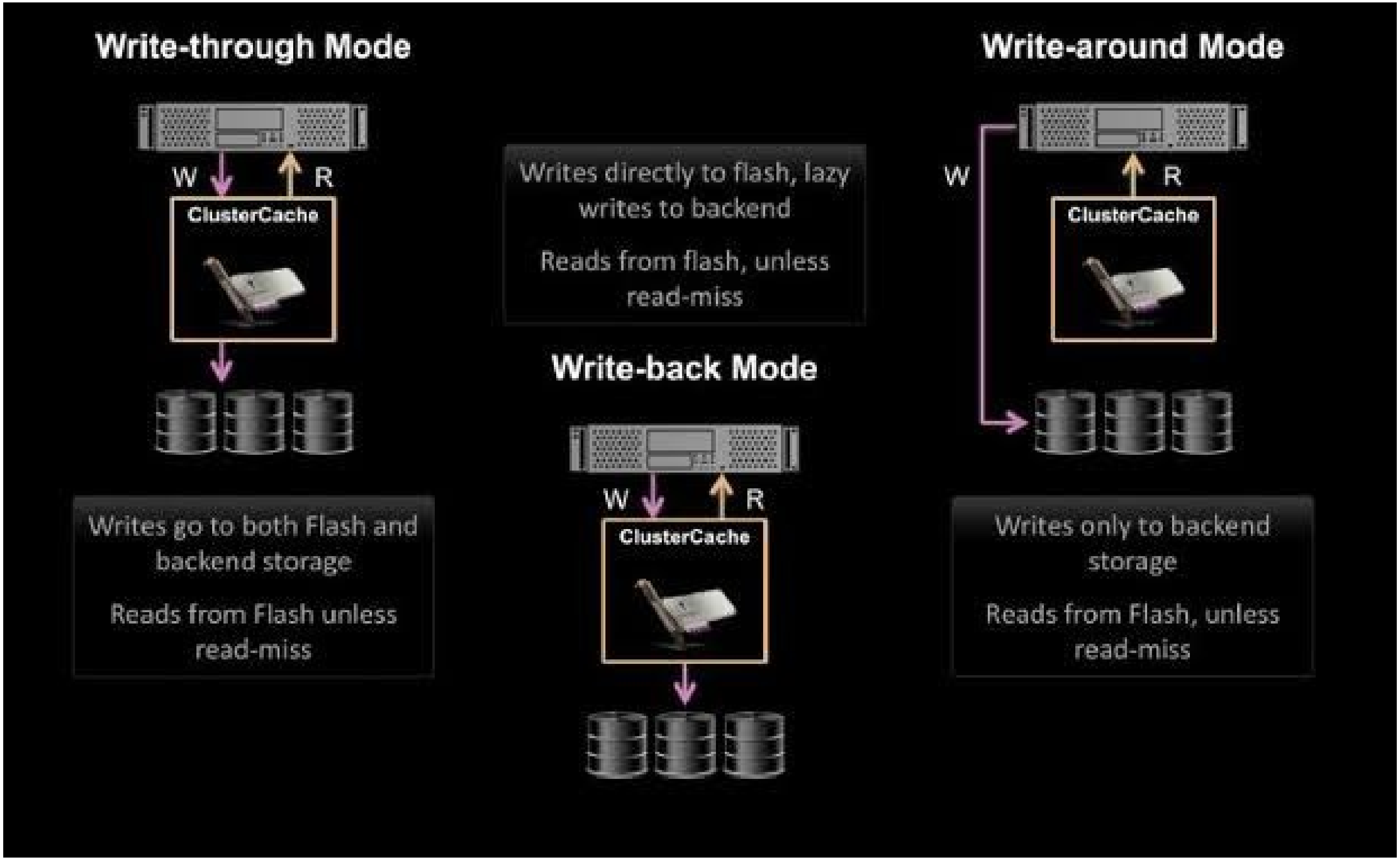
Cache Invalidation – Expiring old data in cache when there is an update in DB

Cache Invalidation strategies – Write-through, Write-around and Write-back

Cache Eviction – To make better use of limited space when cache is full

Cache Eviction policies – FIFO, LIFO, LRU, LFU and Random Replacement etc.

Write-through v/s Write-around v/s Write-back



Write-through Mode
W
R
ClusterCache
Writes go to both Flash and backend storage
Reads from Flash unless read-miss
Writes directly to flash, lazy writes to backend
Reads from flash, unless read-miss
Write-back Mode
W
R
ClusterCache
Write-around Mode
W
R
ClusterCache
Writes only to backend storage
Reads from Flash, unless read-miss

Data Redundancy and Data Replication

Redundancy – Allowing making a copy of Data as backup and using for failsafe purpose to increase reliability.

Redundancy is used to remove single point of failures in system.

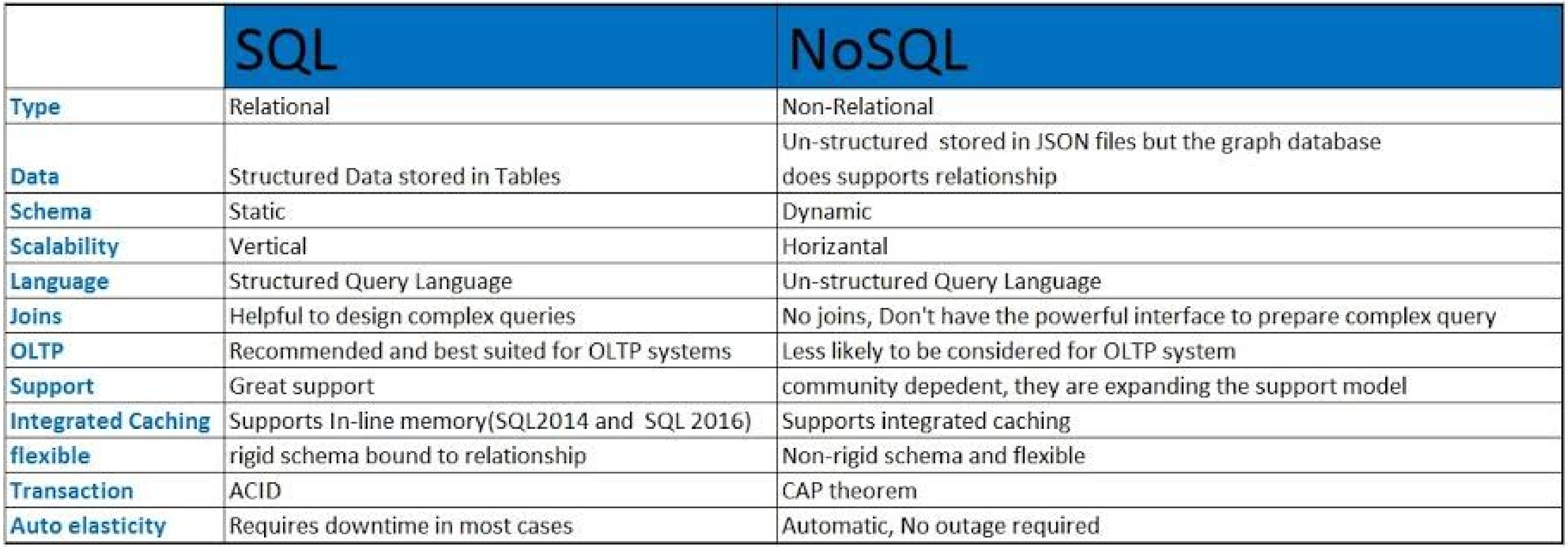
Replication - Sharing information to ensure consistency between redundant resources to improve reliability, fault-tolerance, or accessibility.

Replication is Redundancy in a distributed system. eg. Master-slave DB

SQL v/s NoSQL

SQL – Relational DB storing data in rows and colums. (Static Schema)

NoSQL – Non Relational DB (Not only SQL) storing data in key value pair, document, graph or wide columnar. (Dynamic Schema)



	SQL	NoSQL
Type	Relational	Non-Relational
Data	Structured Data stored in Tables	Un-structured stored in JSON files but the graph database does supports relationship
Schema	Static	Dynamic
Scalability	Vertical	Horizantal
Language	Structured Query Language	Un-structured Query Language
Joins	Helpful to design complex queries	No joins, Don't have the powerful interface to prepare complex query
OLTP	Recommended and best suited for OLTP systems	Less likely to be considered for OLTP system
Support	Great support	community depedent, they are expanding the support model
Integrated Caching	Supports In-line memory(SQL2014 and SQL 2016)	Supports integrated caching
flexible	rigid schema bound to relationship	Non-rigid schema and flexible
Transaction	ACID	CAP theorem
Auto elasticity	Requires downtime in most cases	Automatic, No outage required

When to use SQL: structured data, static schema and ensure ACID compliance

When to use NoSQl: Large Volume and unstructured data , dynamic schema, Rapid development and less complex queries

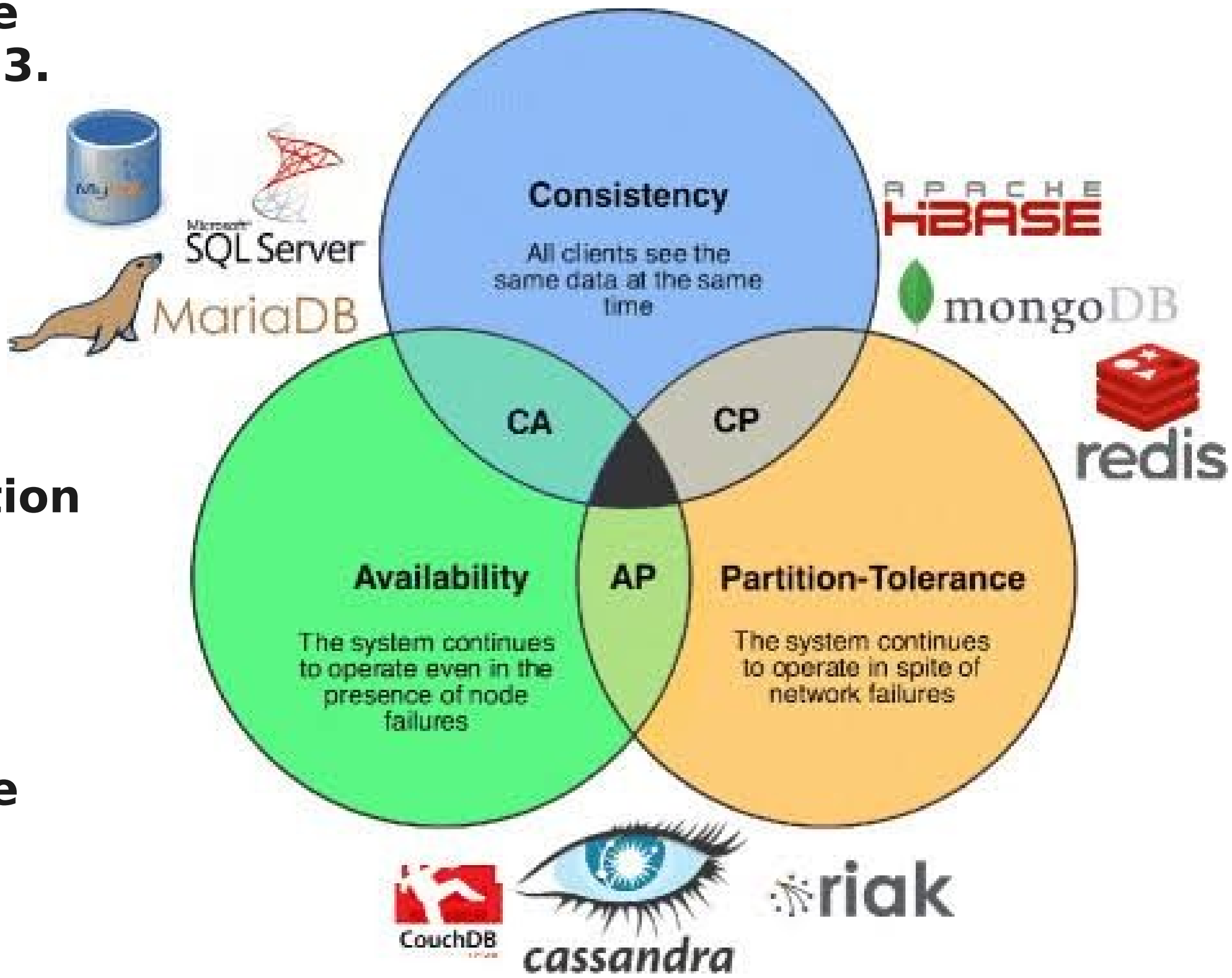
CAP Theorem

CAP - Distributed Database system can only have 2 of 3.

When a network partition failure happens should we decide to:

1. Either cancel the operation and thus decrease the availability but ensure consistency

2. Or, proceed with the operation and thus provide availability but risk inconsistency



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

Consistent Hashing

A common way of load balancing N cache nodes is to put key value pair (K, V) in cache node number hash(K) (mod N) across 0, 1 ,2 .. , N-1 Nodes But this will not work if a cache node is added or removed because N changes and every object is hashed to a new location then.

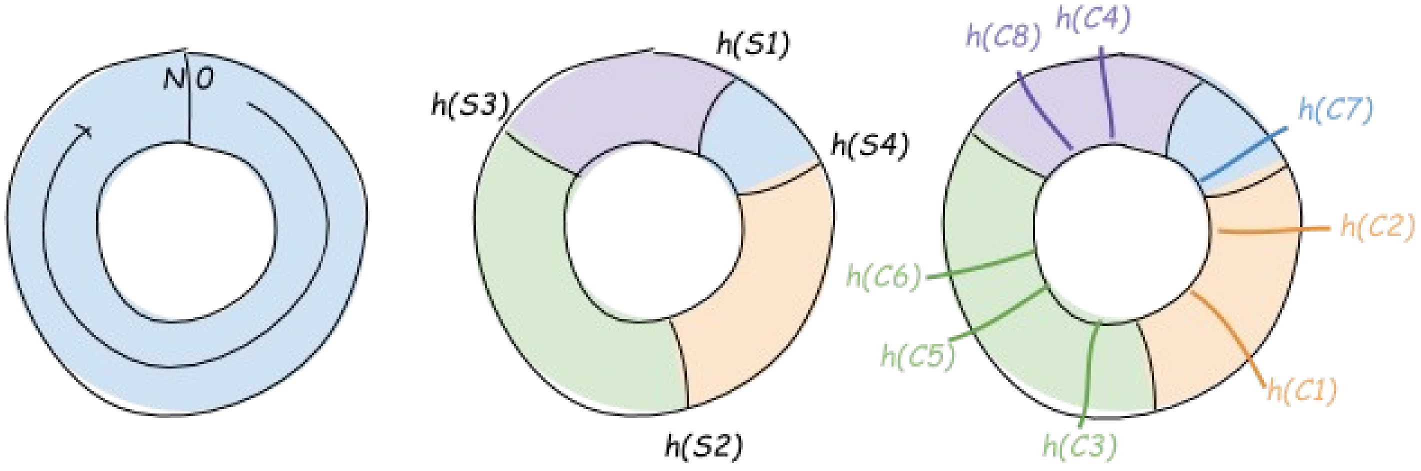


FIGURE 1 FIGURE 2 FIGURE 3

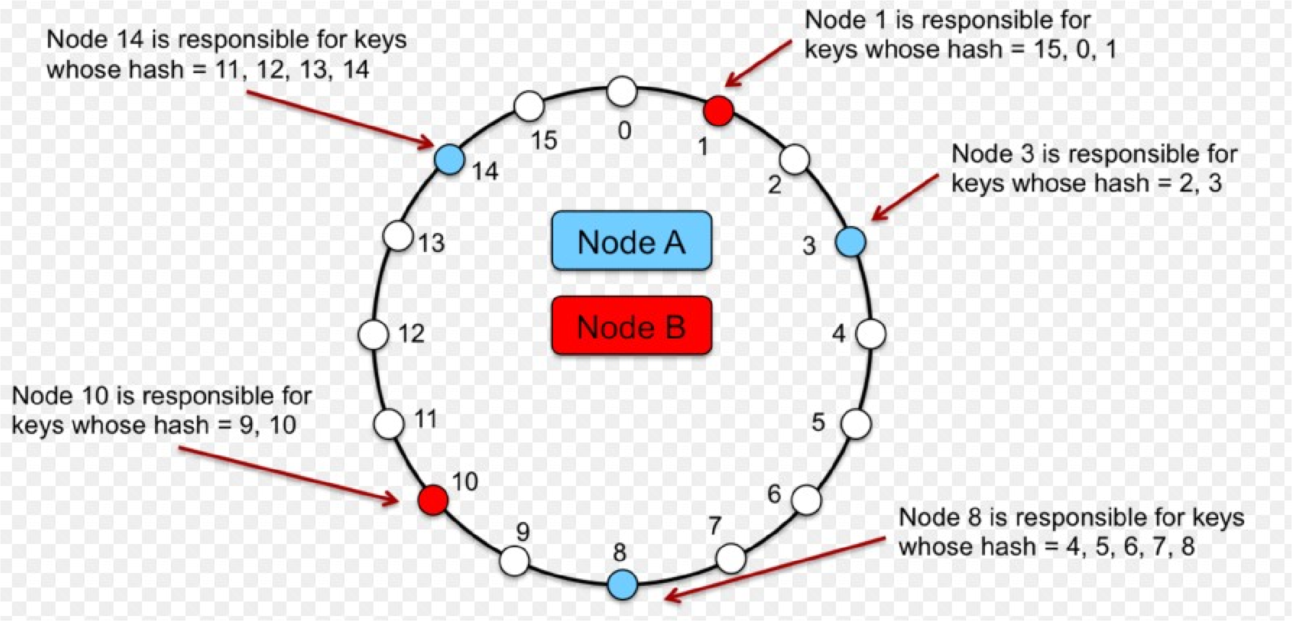
Consistent hashing allows us to distribute data across a cluster in such a way that will minimize reorganization when nodes are added or removed. Hence, the caching system or DHT will be easier to scale up or scale down.

Using Consistent Hashing, when a hash table is resized, only K/N keys need to be remapped on average, where K is the number of keys, and N is the number of nodes.

Explanation

The key of record to be stored (or whose data needs to be retrieved) is passed through a hash function and the value of partition key is calculated.

Based on the partition key value, the range of values is determined in which the key value falls. Appropriately, the computer node is determined which will be used for accessing data or storing data.

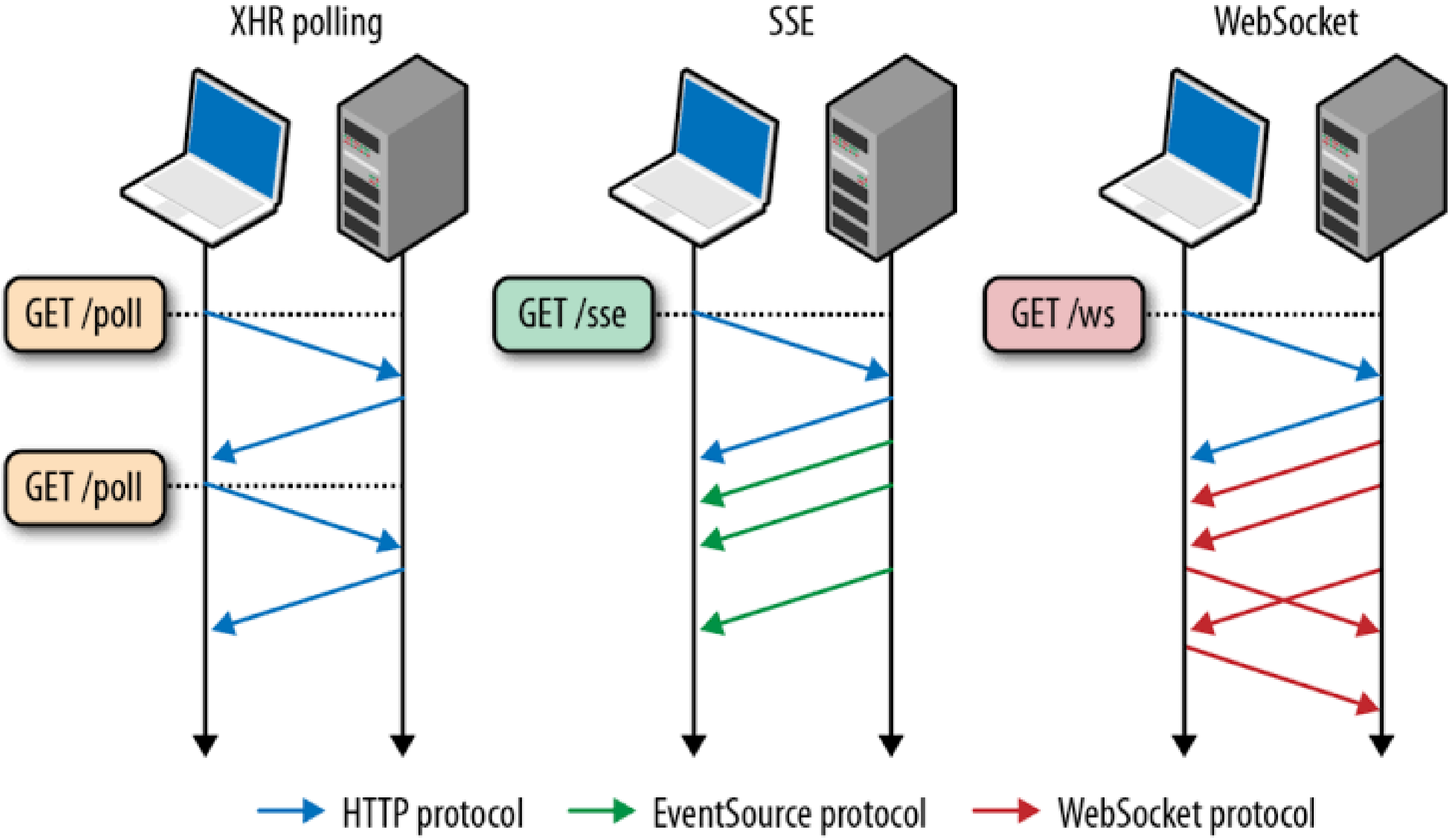

Node 14 is responsible for keys whose hash = 11, 12, 13, 14
Node 1 is responsible for keys whose hash = 15, 0, 1
Node 3 is responsible for keys whose hash = 2, 3
Node 10 is responsible for keys whose hash = 9, 10
Node 8 is responsible for keys whose hash = 4, 5, 6, 7, 8
Node A
Node B
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

In case, node 3 gets shut down, the hash value in range 1-3 is mapped to node 8.

In case, a new node is added, say node 6. In that case, the hash value such as 4 and 5 gets mapped to node 6.

This avoids issue of remapping nearly all the records to newer nodes when traditional hashing technique is used.

Polling v/s web Sockets v/s Server Sent Events


XHR polling
SSE
WebSocket
GET /poll
GET /poll
GET /sse
GET /ws
HTTP protocol
EventSource protocol
WebSocket protocol