THE SYSTEM DESIGN ATLAS

A Field Guide to Modern Backend Architecture

INTENT:
Knowledge Transfer & Reference

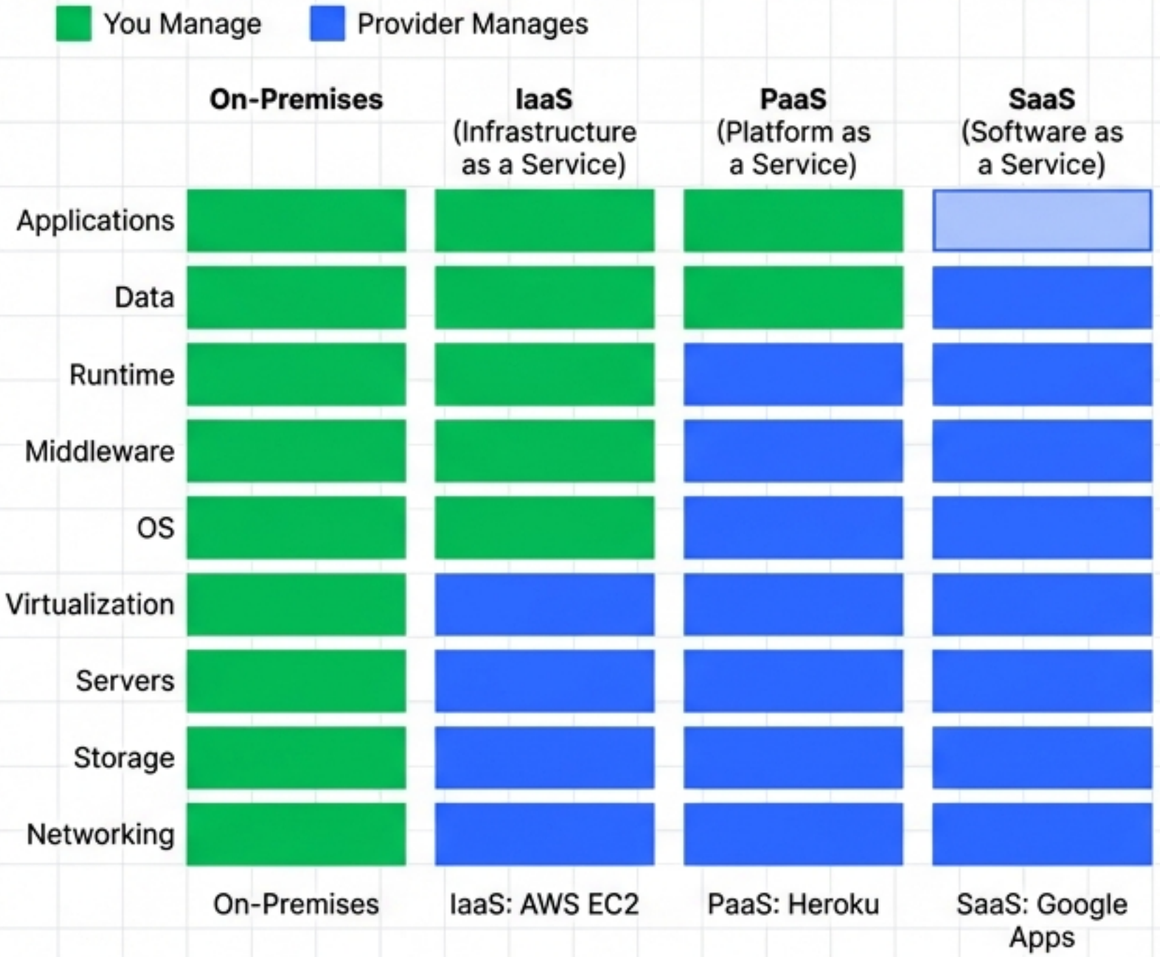
AUDIENCE:
Engineers, Architects, Managers

SCOPE:
Infrastructure · Data · Network · Patterns · Case Studies

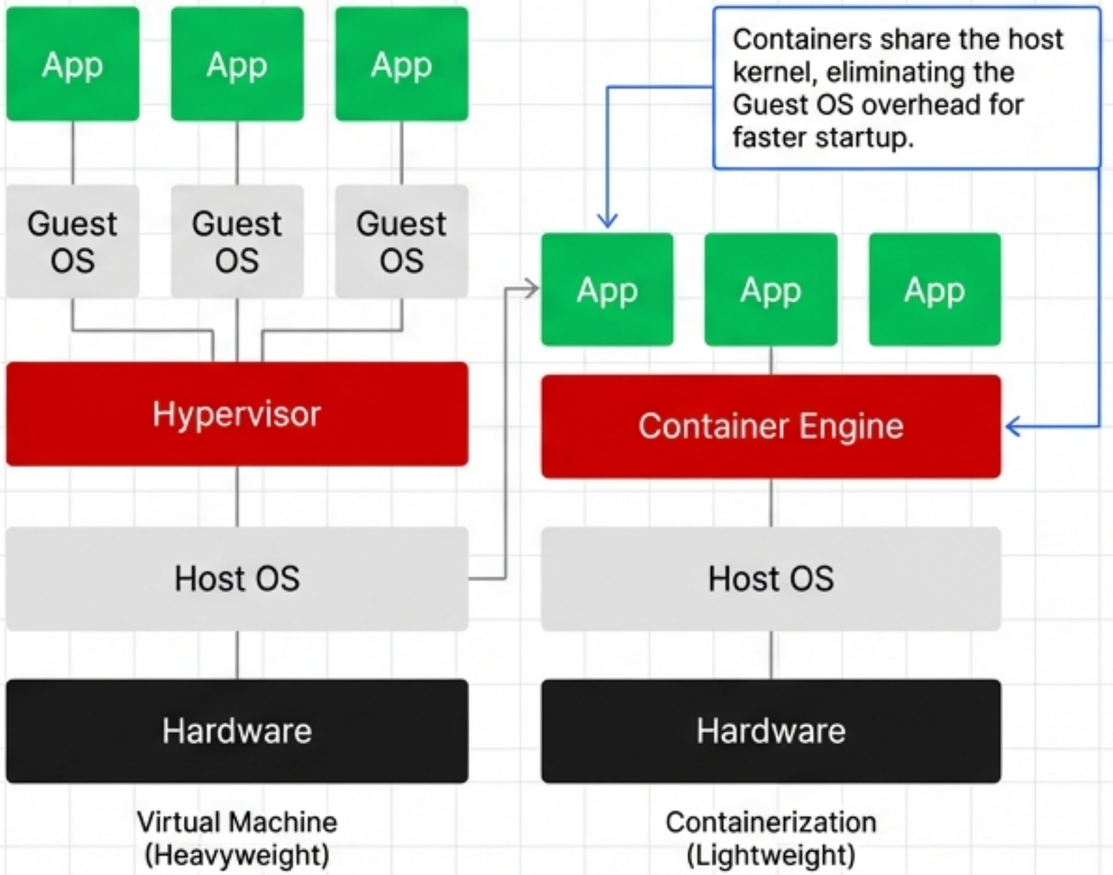
Helvetica Now Display Bold ExtraBlack

01. THE LANDSCAPE: CLOUD & EXECUTION

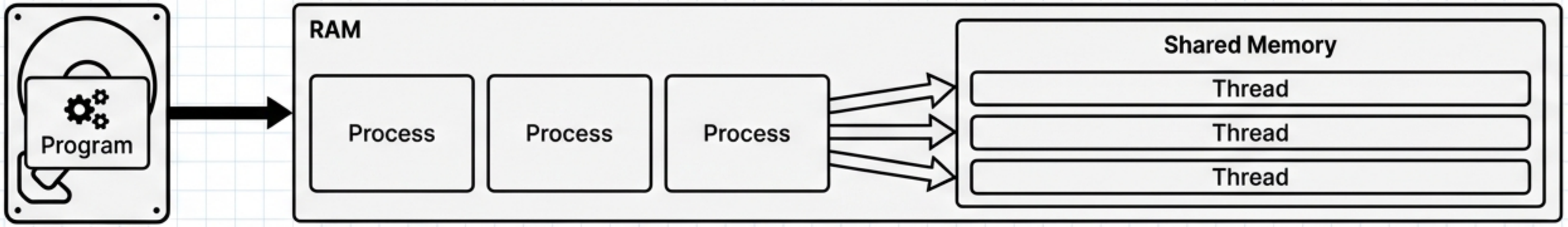
THE CLOUD RESPONSIBILITY MATRIX



VIRTUALIZATION VS. CONTAINERIZATION

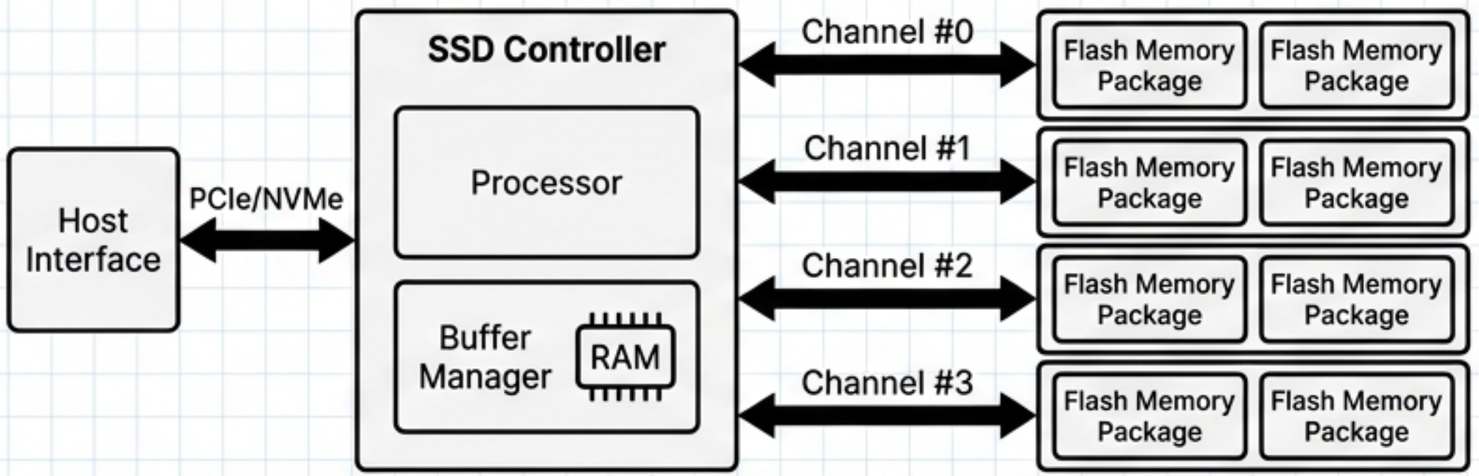


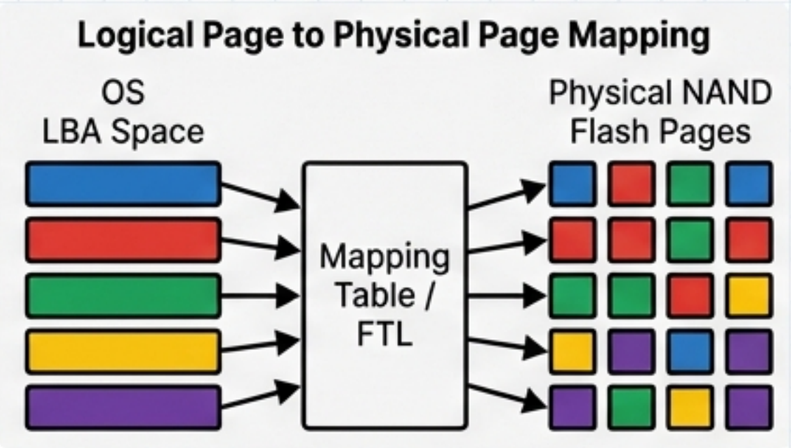
02. UNDER THE HOOD: THREADS & SSDs



Process: Independent memory space

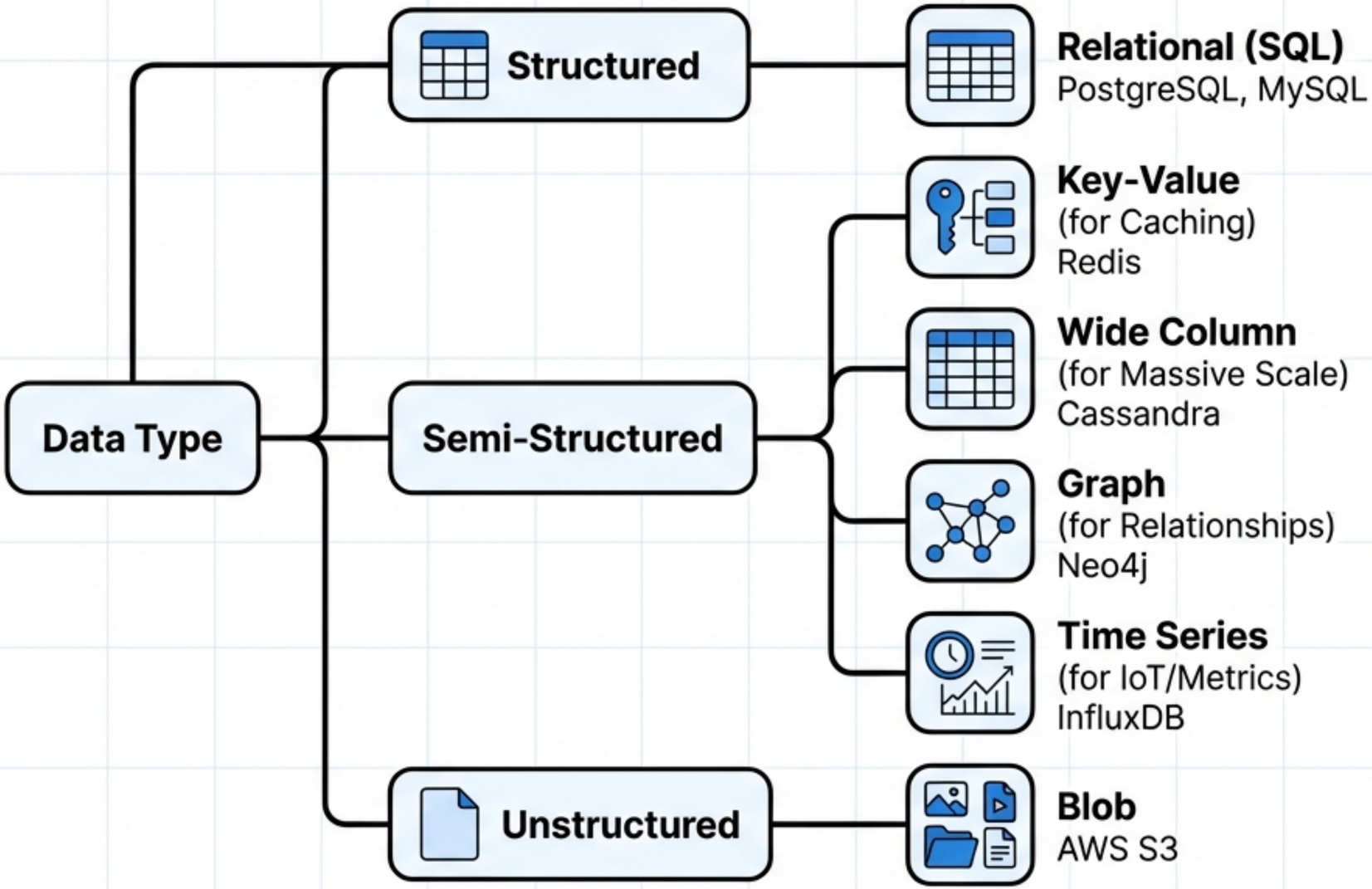
Thread: Shared memory space





Parallelism: The controller writes to multiple flash particles simultaneously via independent channels.

03. STRATEGY: CHOOSING A DATABASE



OLTP (Transactions) vs. OLAP (Analytics) - Choose based on read/write patterns.

04. CONSISTENCY & ISOLATION

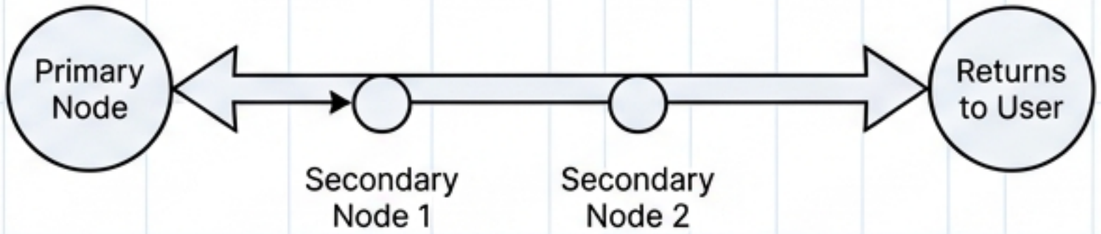
The Isolation Matrix

Isolation Level	Dirty Read	Non-repeatable Read	Phantom Read
Serializable (Strict)	Impossible	Impossible	Impossible
Repeatable Read	Impossible	Impossible	Probably
Read Committed	Impossible	Probably	Probably
Read Uncommitted	Probably	Probably	Probably

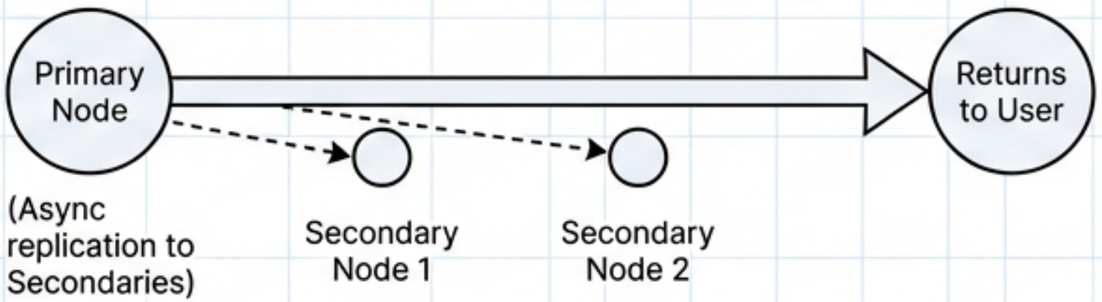
MVCC (Multi-Version Consistency Control) uses Transaction IDs to create snapshot views.

Consistency vs. Latency Trade-off

Scenario 1: High Consistency / High Latency



Scenario 2: Eventual Consistency / Low Latency

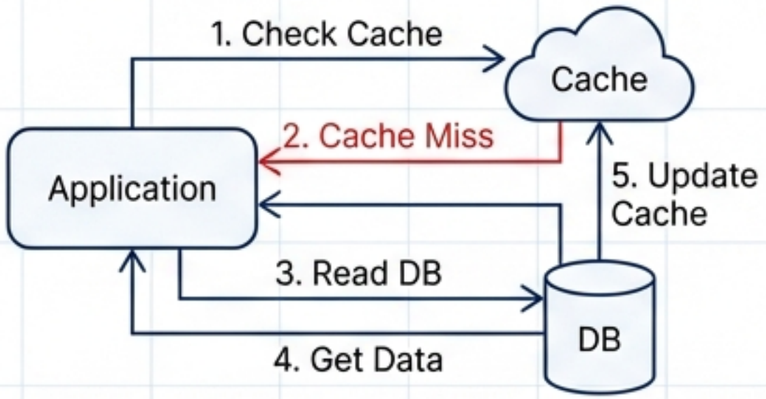


The CAP Theorem dictates you cannot have perfect Consistency and Availability simultaneously in a partitioned system.

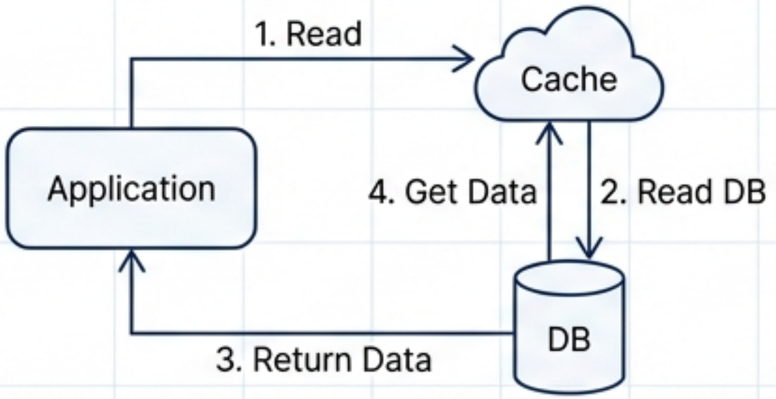
05. CACHING STRATEGIES

Caching Patterns

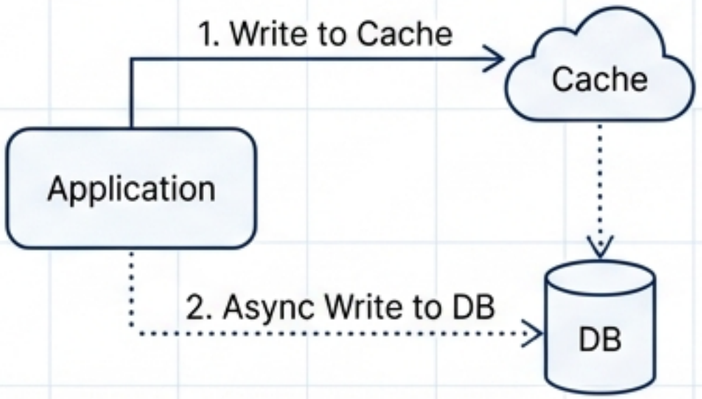
Cache-Aside



Read-Through



Write-Back

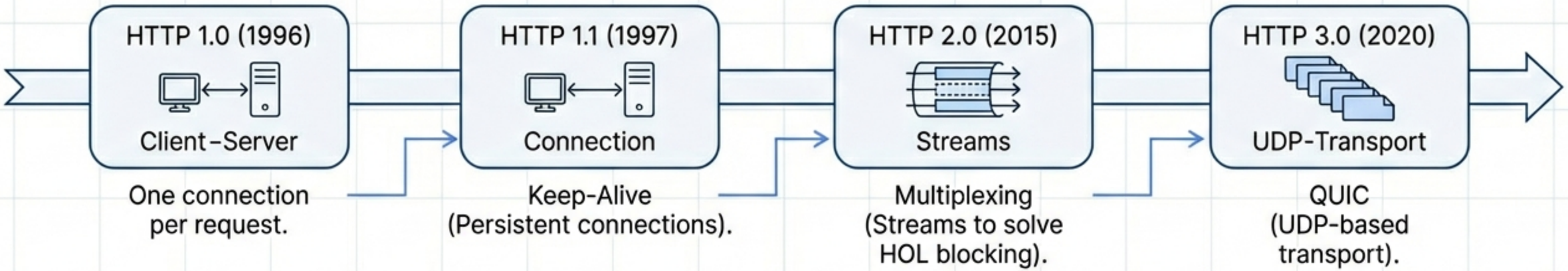


Redis vs. Memcached

Feature	Redis	Memcached
Data Structures	Complex (Lists, Sets, Sorted Sets, Bitmaps)	Simple Strings
Architecture	Single-Threaded Event Loop	Multi-Threaded
Persistence	Yes (RDB/AOF Snapshots)	No (In-memory only)
Eviction	Advanced Policies (LFU/LRU)	LRU Only

06. NETWORK: HTTP EVOLUTION & API STYLES

HTTP Evolution Timeline

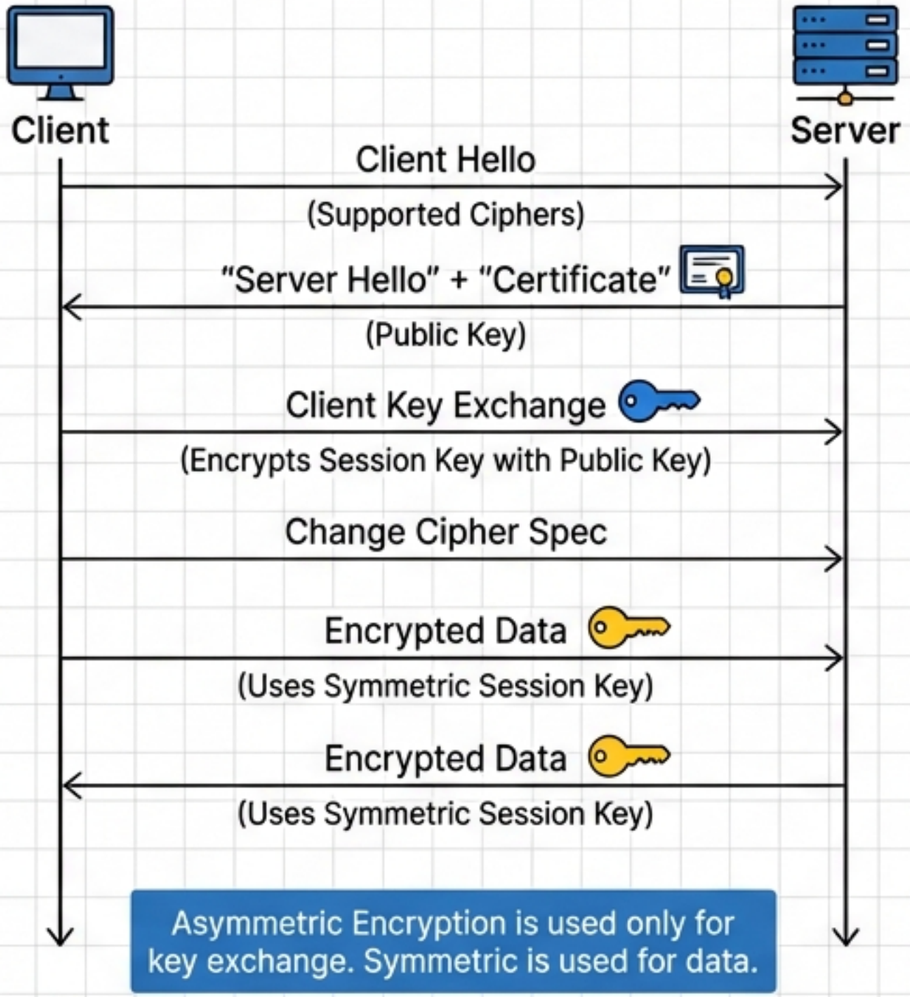


API Paradigm Matrix

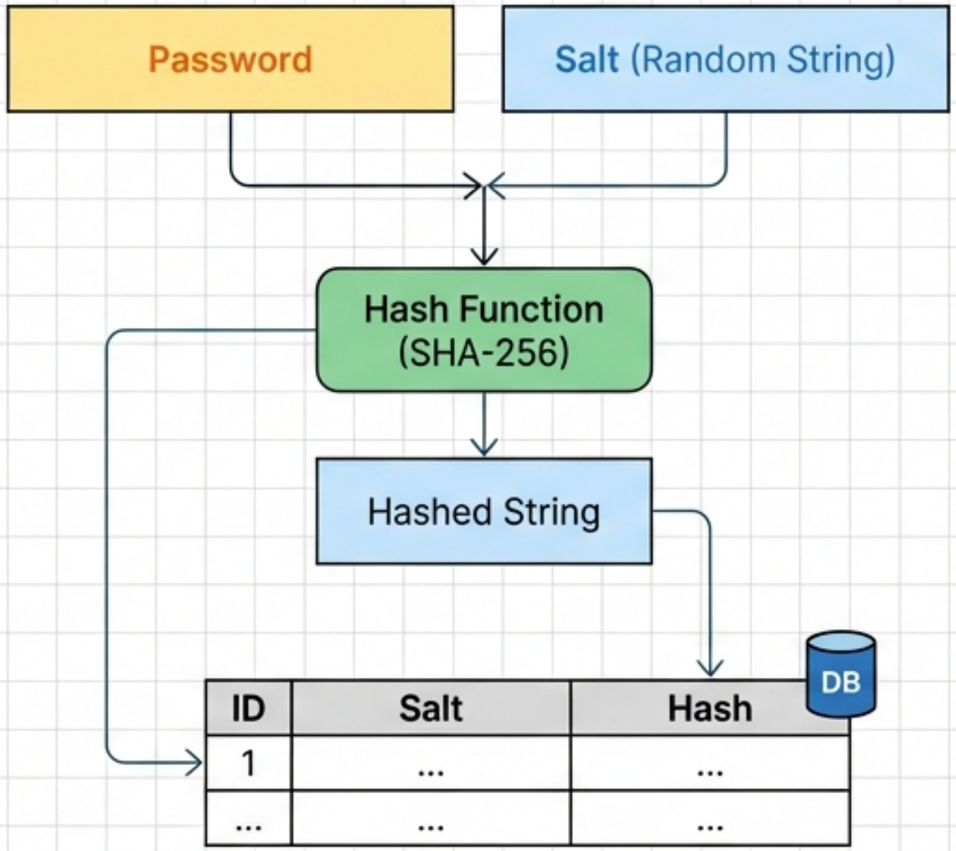
SOAP	REST	GraphQL	RPC (gRPC)
XML-only Strict Standard	Resource-based JSON/Stateless	Schema-driven Client-defined queries	Action-oriented Protobufs
Use case: Legacy Enterprise/Banking	Use case: Public Web APIs	Use case: Complex Data/Mobile	Use case: Internal Microservices

07. SECURITY: HTTPS & HASHING

The TLS Handshake Sequence



Salting and Hashing

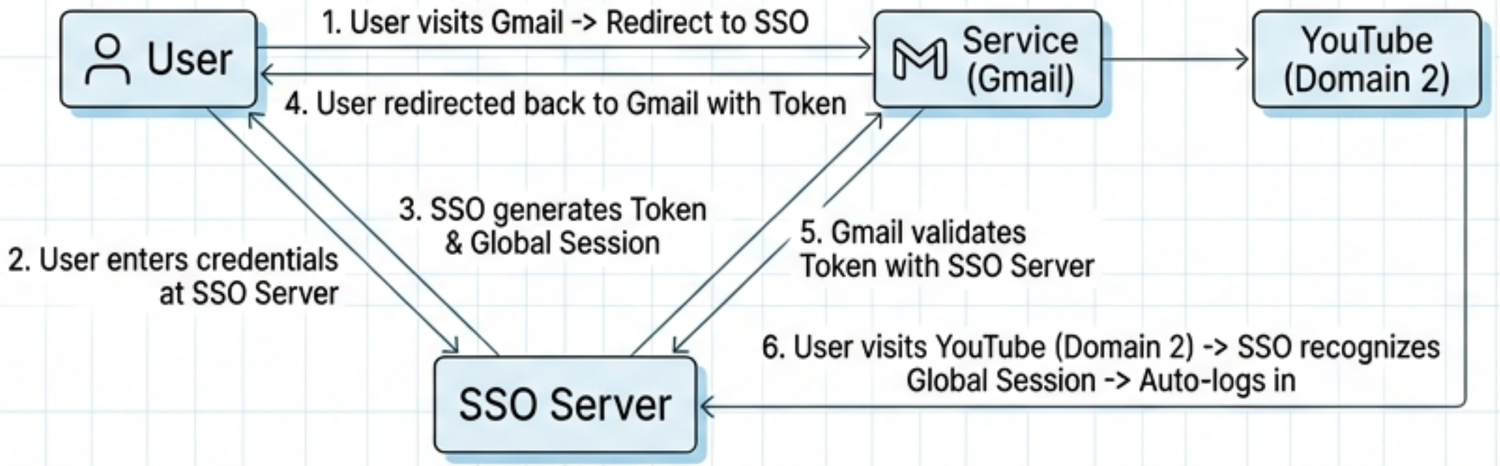


08. IDENTITY: SSO & HMAC

Helvetica Now Display

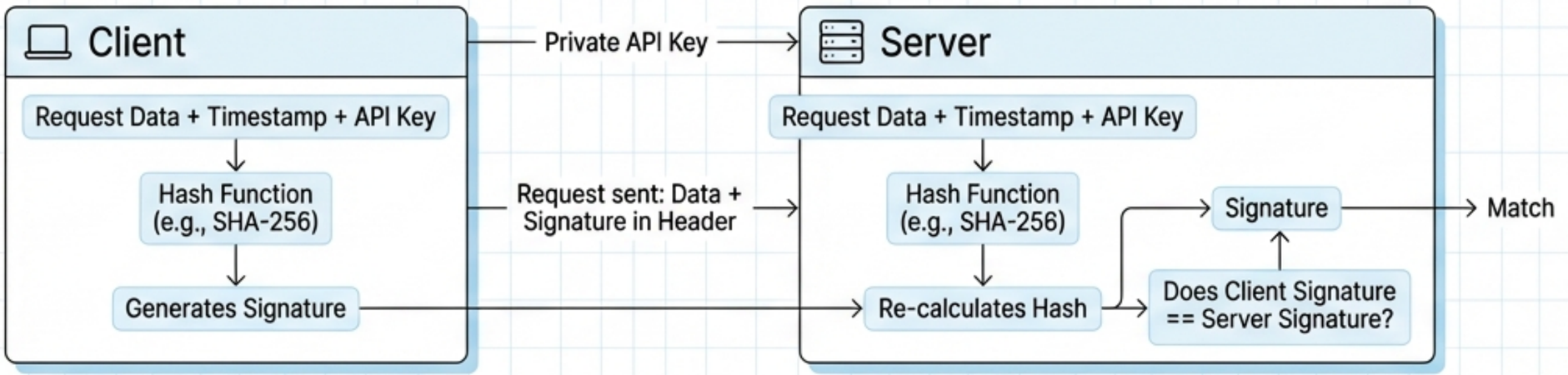
Single Sign-On (SSO) Flow

in Inter

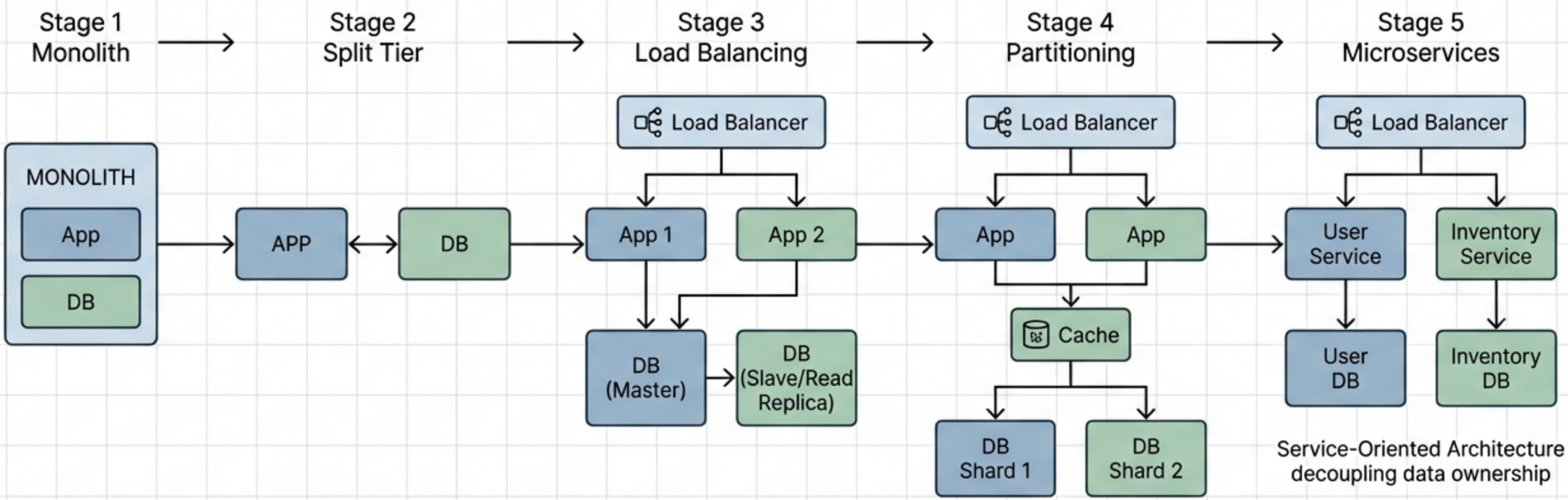


HMAC Authentication

in Inter



09. ARCHITECTURE: SCALING EVOLUTION

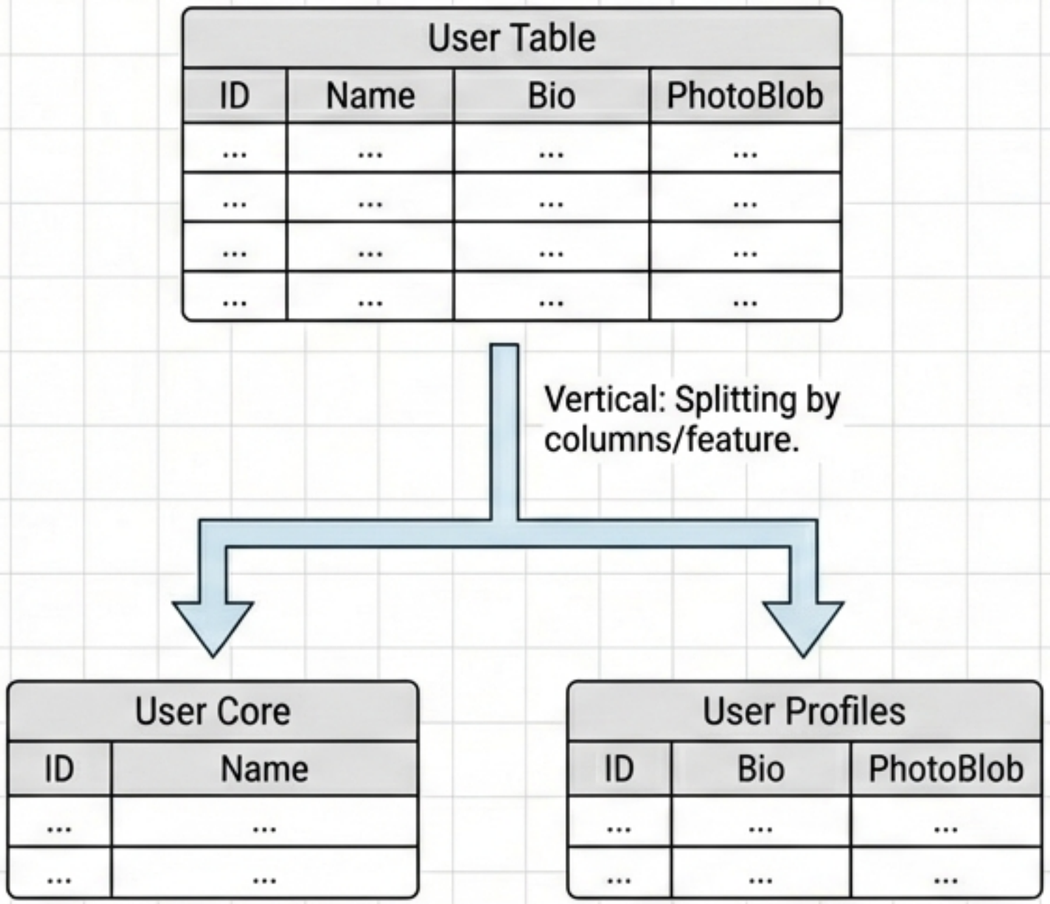


10. DATA PARTITIONING STRATEGIES

Helvetica Now Display

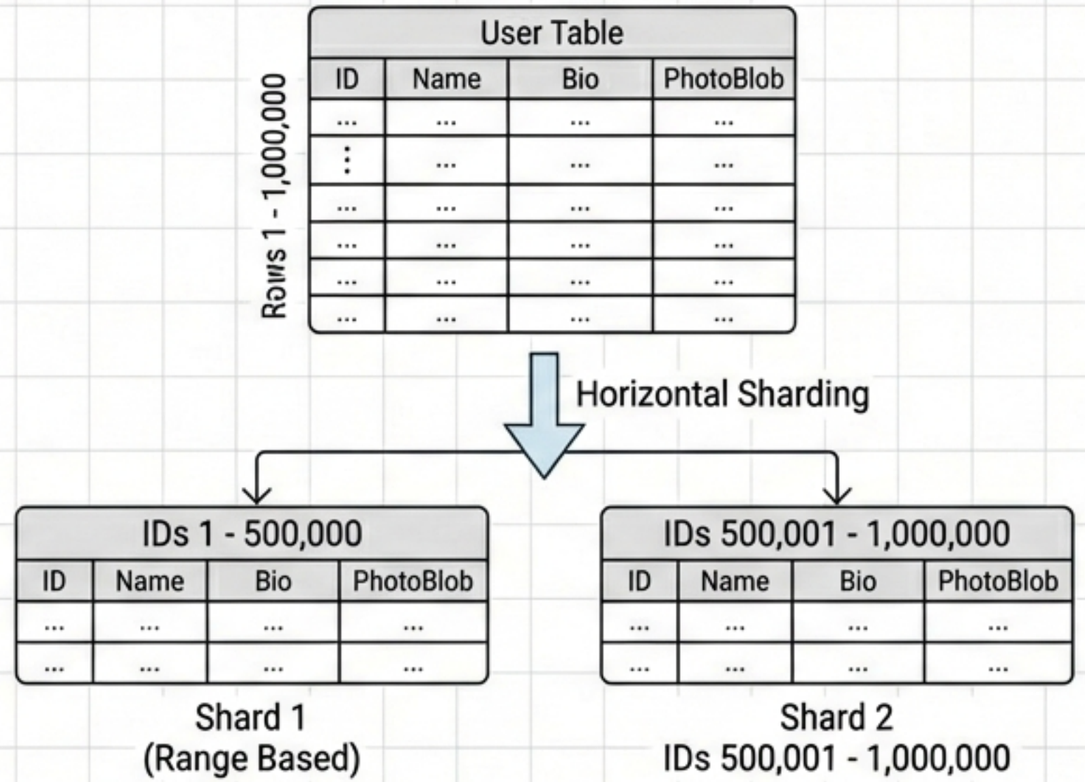
Vertical Partitioning

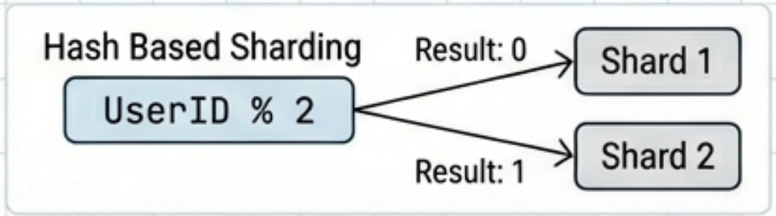
in Inter



Horizontal Sharding

in Inter

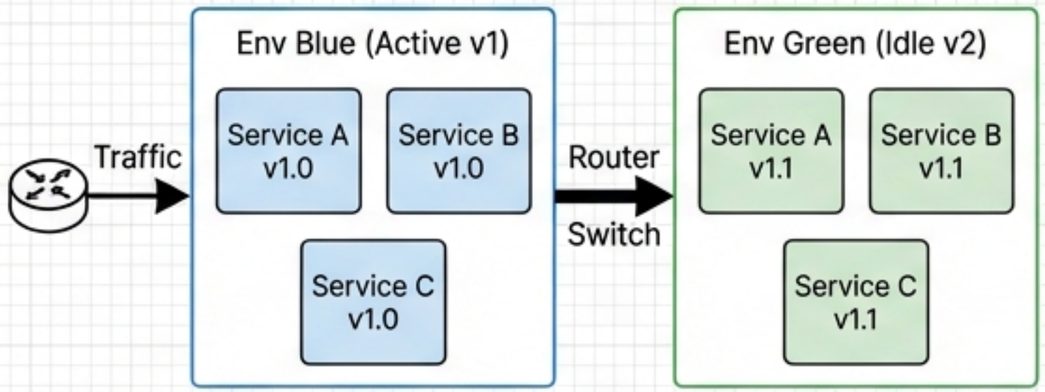




11. DEPLOYMENT & COORDINATION

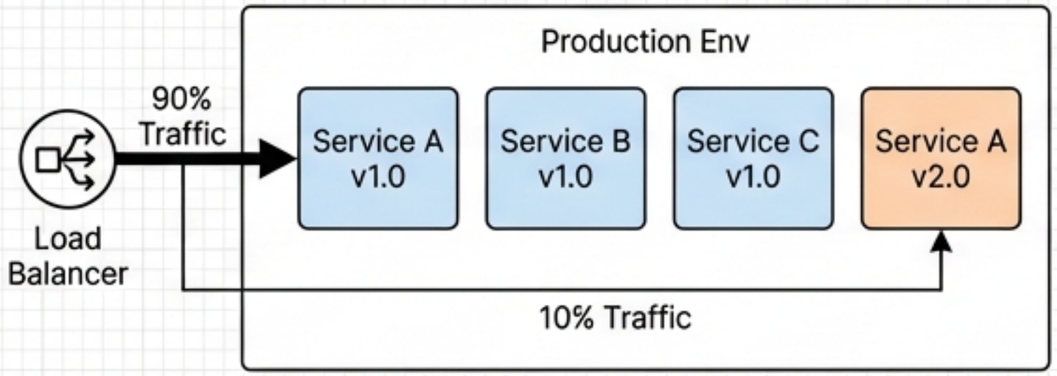
Deployment Strategies

Blue-Green Deployment



Zero-downtime deployment with instant rollback.

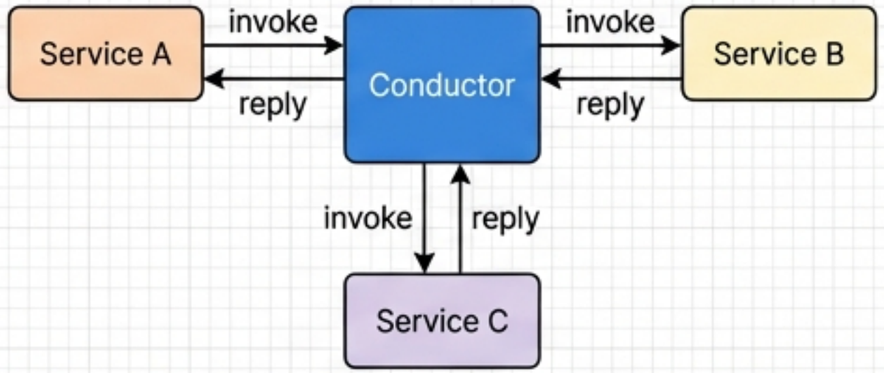
Canary Deployment



Rollout to a small user subset for testing.

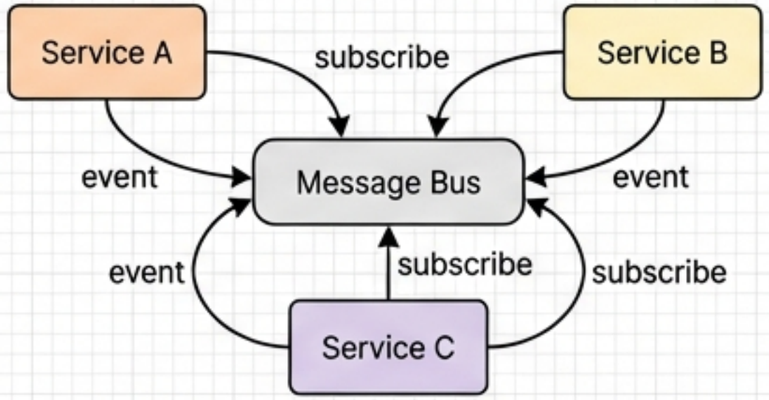
Microservices Interaction

Orchestration



Centralized control, tighter coupling.

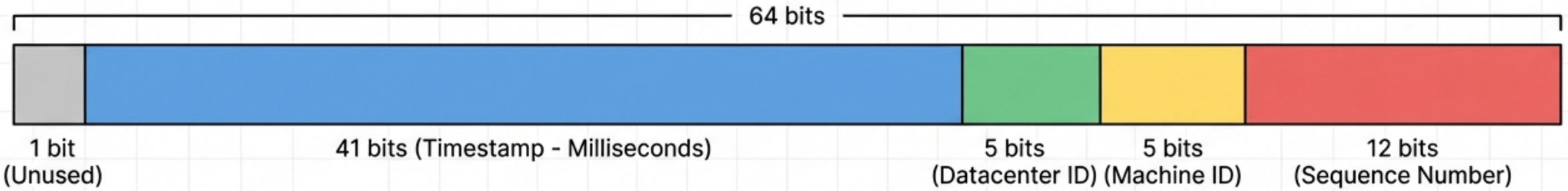
Choreography



Event-driven, loose coupling, asynchronous.

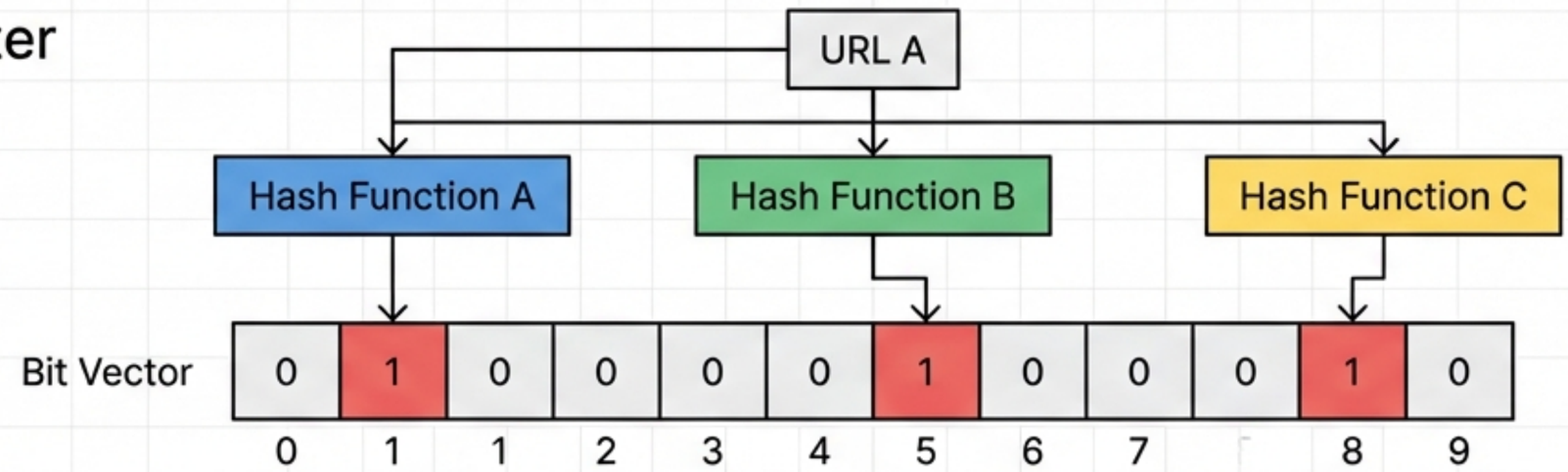
12. MECHANISMS: IDs & FILTERS

Snowflake ID Generator



Twitter Snowflake ID - Sortable & Distributed.

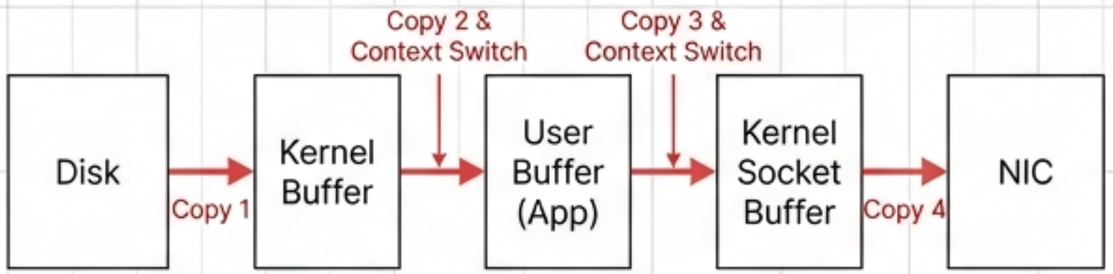
Bloom Filter



Probabilistic: If bits are 1, it MIGHT exist.
If any bit is 0, it DEFINITELY DOES NOT exist.

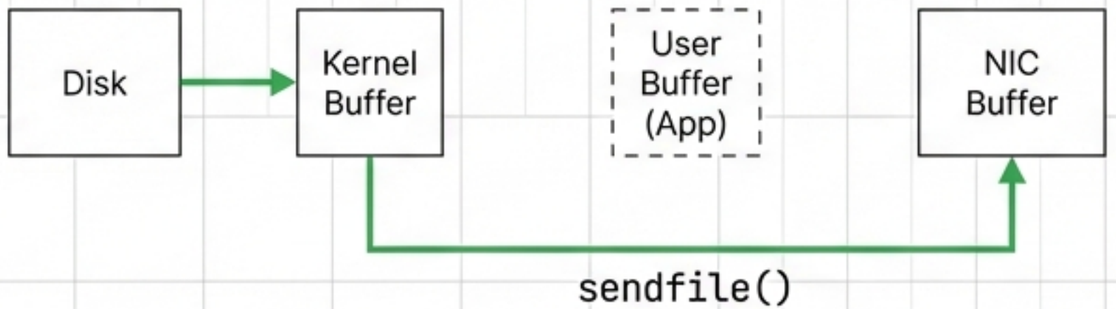
13. OPTIMIZATION: KAFKA ZERO COPY

Standard I/O (Inefficient)



Requires 4 copies and 4 context switches.

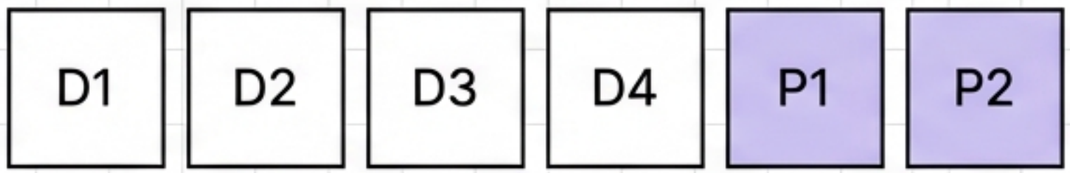
Zero Copy (Efficient)



Zero Copy reduces context switching by ~65%, allowing high throughput streaming.

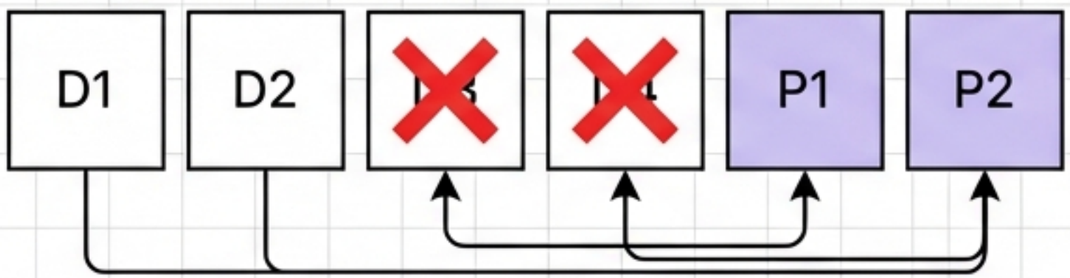
14. STORAGE: DURABILITY & UPLOADS

Erasure Coding



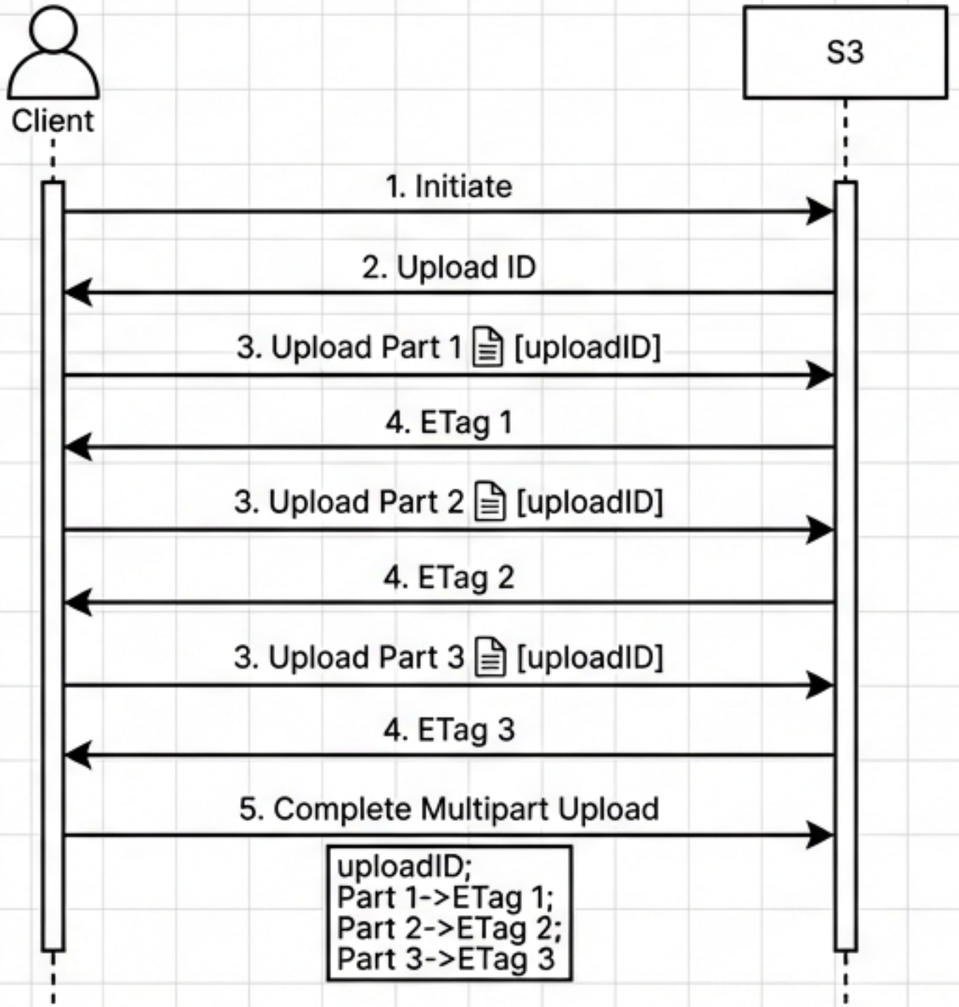
`P1 = D1 + D2 + D3 + D4; P2 = f(D1, D2, D3, D4)`

Data Reconstruction

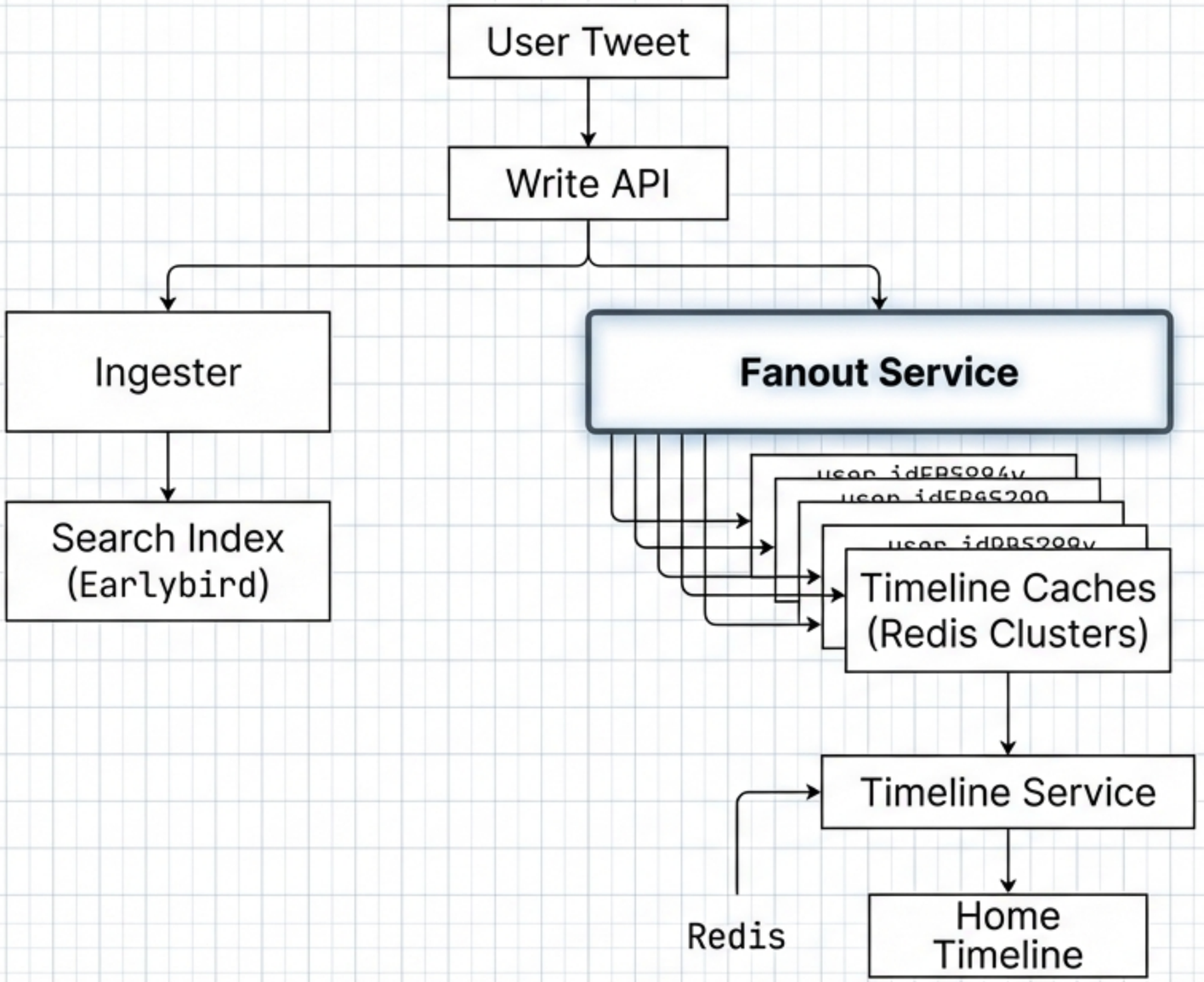


High Durability (11 nines) with only 50% storage overhead.

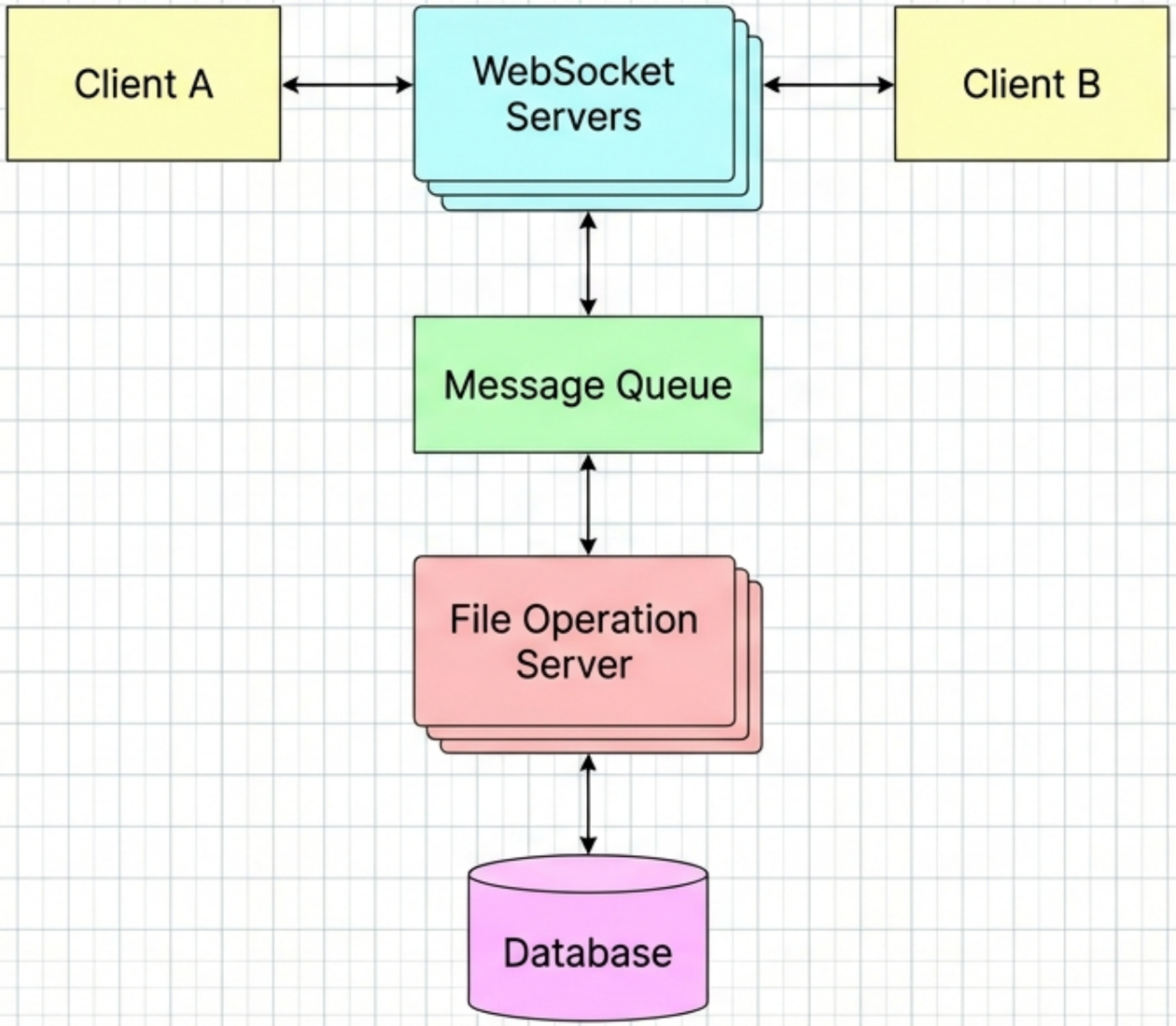
S3 Multipart Upload



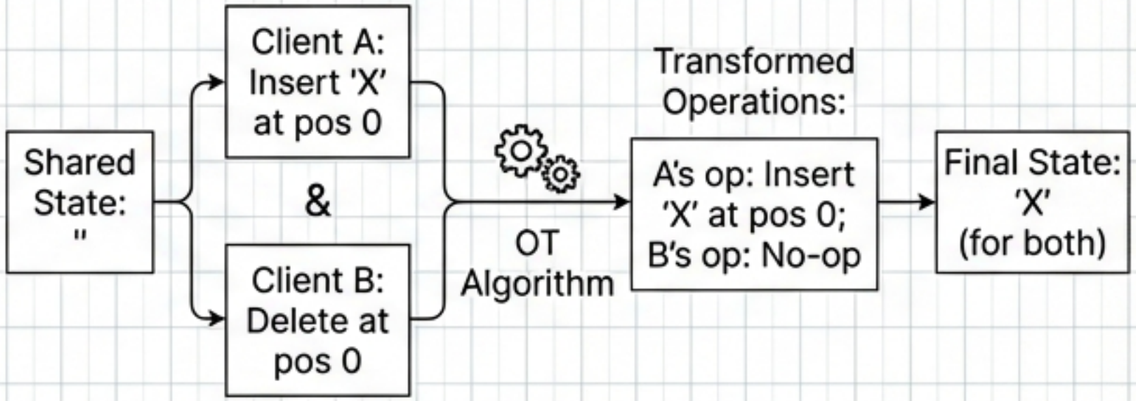
15. CASE STUDY: TWITTER ARCHITECTURE



16. CASE STUDY: GOOGLE DOCS (COLLABORATION)



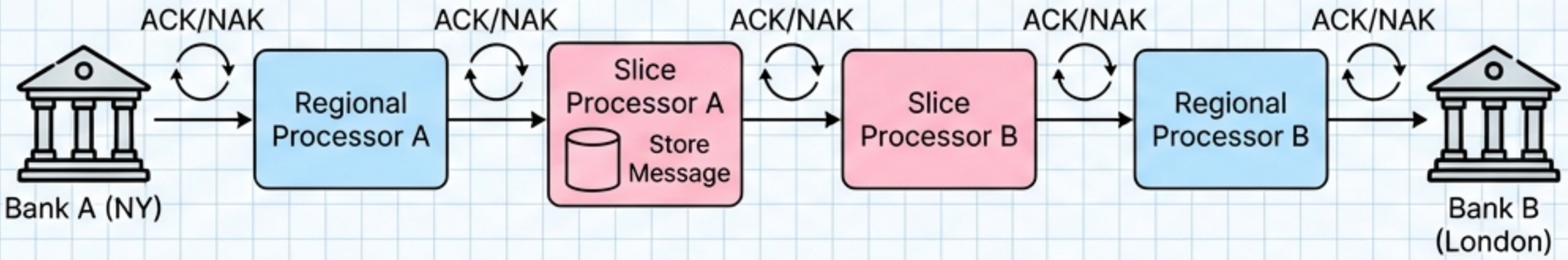
Operational Transformation (OT)



OT resolves concurrent editing conflicts to ensure eventual consistency.
Code snippets's in JetBrains Mono.

17. CASE STUDY: PAYMENTS & SWIFT

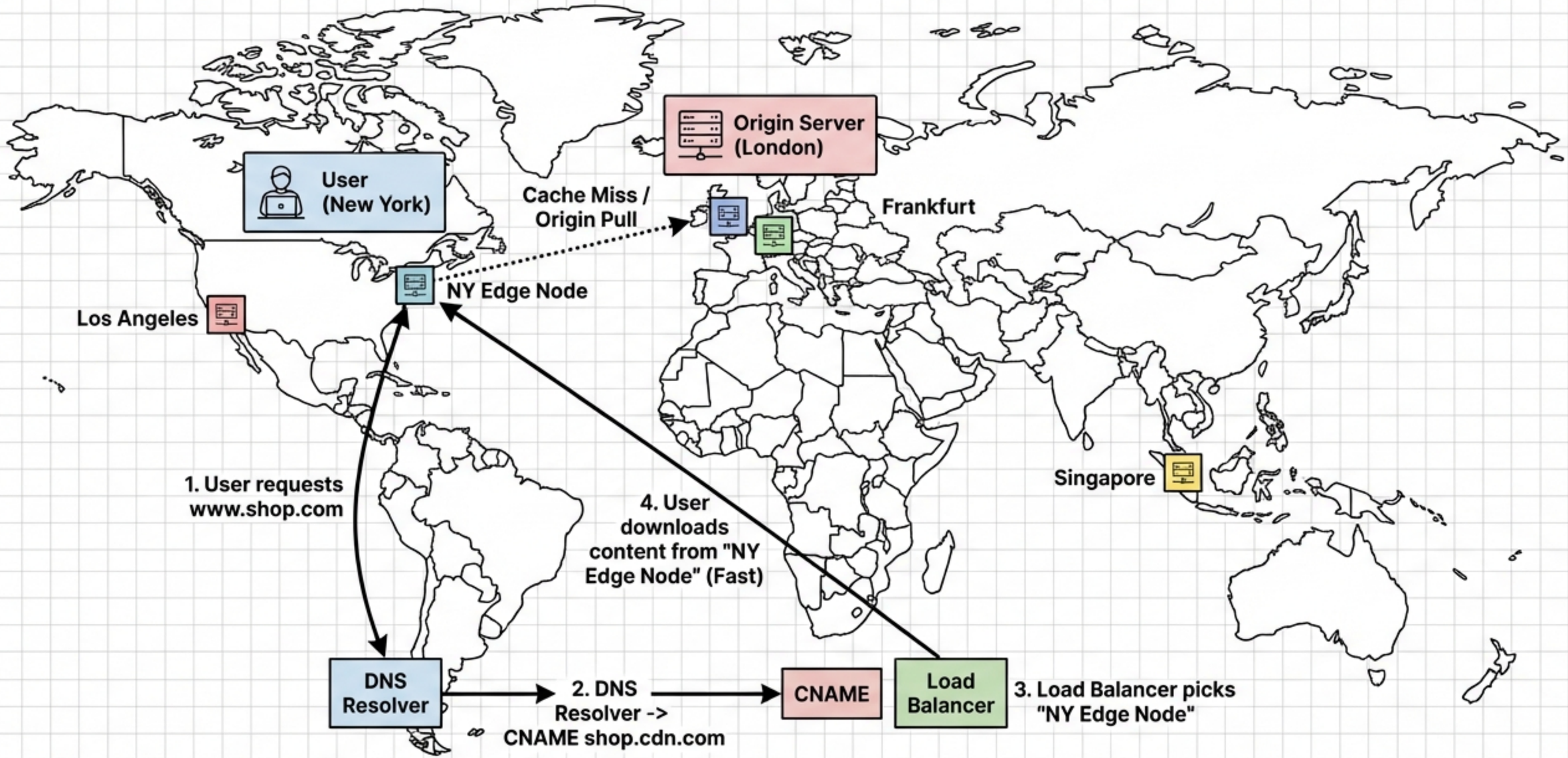
ByteByteGo



Key Patterns

1. **Idempotency** (Prevent double charge).
2. **Reconciliation** (Ledger matching).

18. GLOBAL SCALE: CDN & DNS



19. CONTINUOUS LEARNING

ByteByteGo

ARCHITECTURE
Inter Bold

- **The Amazon Builders' Library**
  Inter Regular

- **Martin Fowler's Blog**
  Inter Regular

DEVOPS & RELIABILITY
Inter Bold

- **The Phoenix Project**
  Inter Regular

- **Google SRE Book**
  Inter Regular

- **Accelerate**
  Inter Regular

CODE & PATTERNS
Inter Bold

- **Head First Design Patterns**
  Inter Regular

- **Designing Data-Intensive Applications**
  Inter Regular

"System design is not about memorizing solutions, but understanding trade-offs. **Latency** vs. Consistency. **SQL vs. NoSQL. Complexity vs. Maintainability.**"

Inter Medium