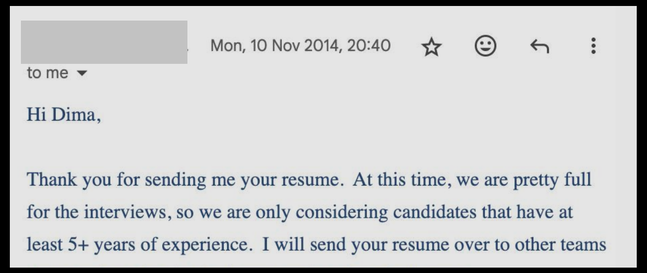
SYSTEM DESIGN

интервью для практиков

Дмитрий Волыхин
@javaswag, @faangtalk

2014

at least 5 years of experience...



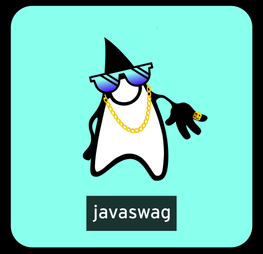
Mon, 10 Nov 2014, 20:40

to me

Hi Dima,

Thank you for sending me your resume. At this time, we are pretty full for the interviews, so we are only considering candidates that have at least 5+ years of experience. I will send your resume over to other teams

2016



2019

❌ Провал

2020

2021

✅ Успех: F, A, G

ITS JUST A GAME

Систем дизайн интервью:

субьективное

Систем дизайн интервью:

 субьективное

 нет правильных ответов

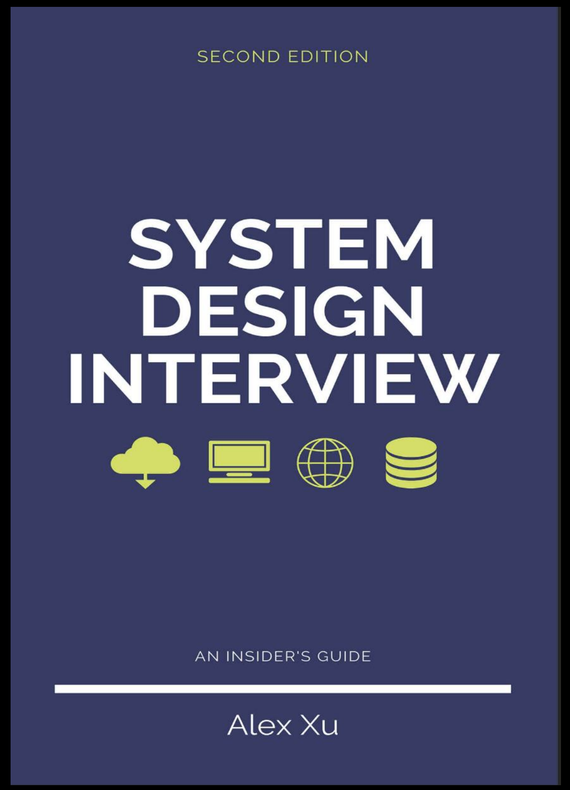
Систем дизайн интервью:

 субьективное

 нет правильных ответов

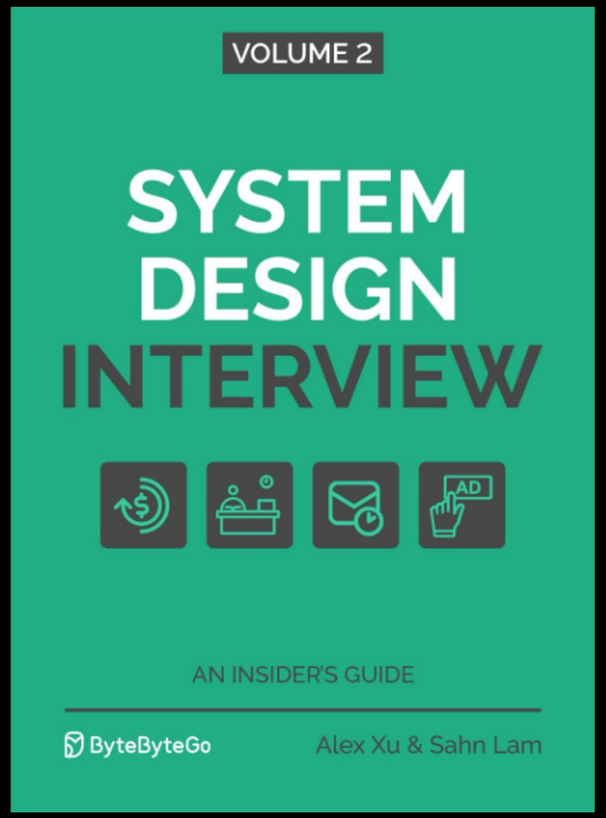
 иногда абсурдное

Типовые архитектуры



Rate Limiter
Consistent Hashing
Key-Value Store
Unique Id Generator
URL Shortener
Web Crawler
Notification System
News Feed System
Chat System
Search Autocomplete System
Youtube
Google Drive

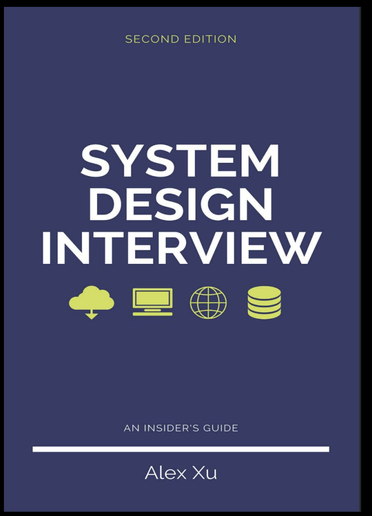
Типовые архитектуры II



Alex Xu
@alexxubyte

Many people asked about the table of contents for the System Design Interview (Volume 2). Here you go:

Table Of Contents:



Rate Limiter
Consistent Hashing
Key-Value Store
URL Shortener
Web Crawler
Notification System
News Feed System
Chat System
Search Autocomplete System
Youtube
Google Drive
Proximity Service
Nearby Friends
Google Maps
Distributed Message Queue
Metrics Monitoring
Ad Event Aggregation
Hotel Reservation
Distributed Email Service
S3-like Object Storage
Leaderboard
Payment System
Digital Wallet
Stock Exchange

Rate Limiter ID Generation Consistent Hashing

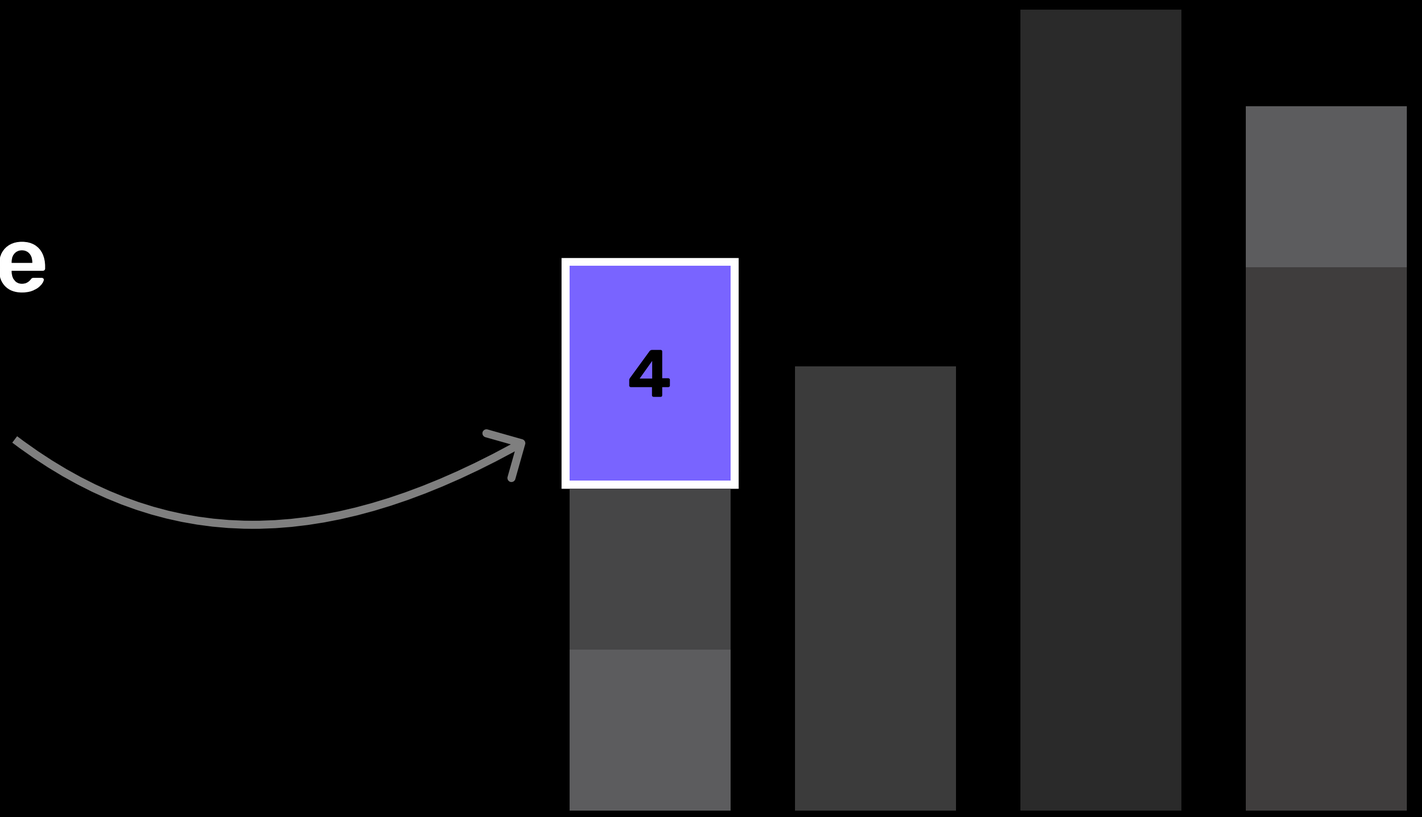
Key-Value Store News Feed Chat GeoHash

Search Autocomplete System S3

Правила игры

⏳ 4 мин

функциональные требования (фт)

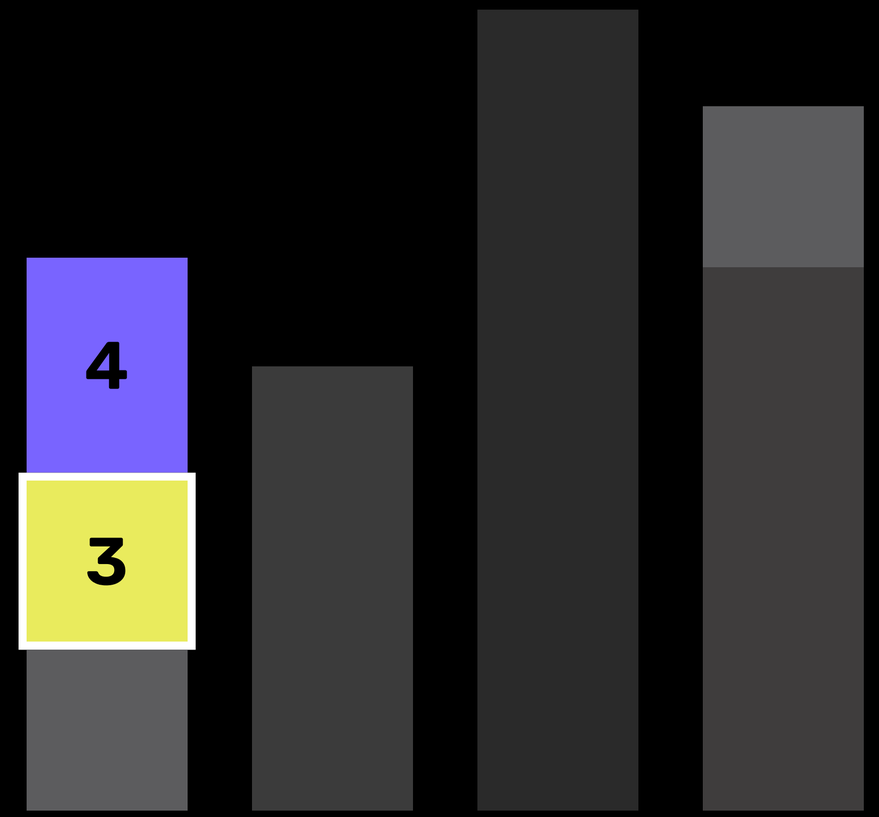


Правила игры

1. фт

⏳ 3 мин

нефункциональные требования (нфт)

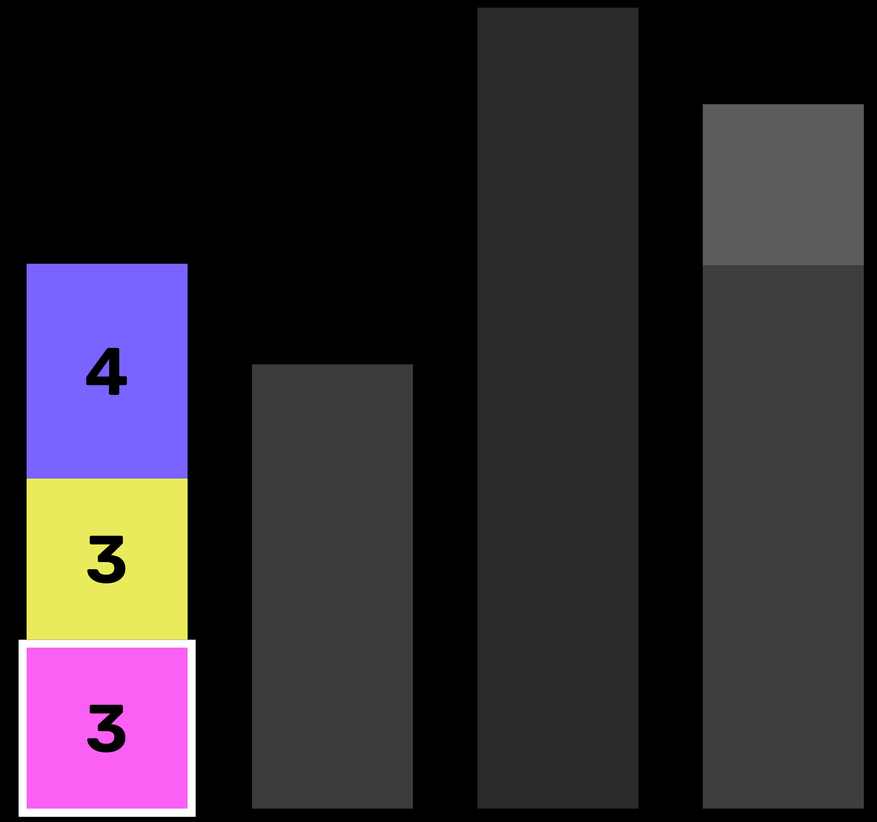


Правила игры

1. фт
2. нфт

⏳ 3 мин

оценка

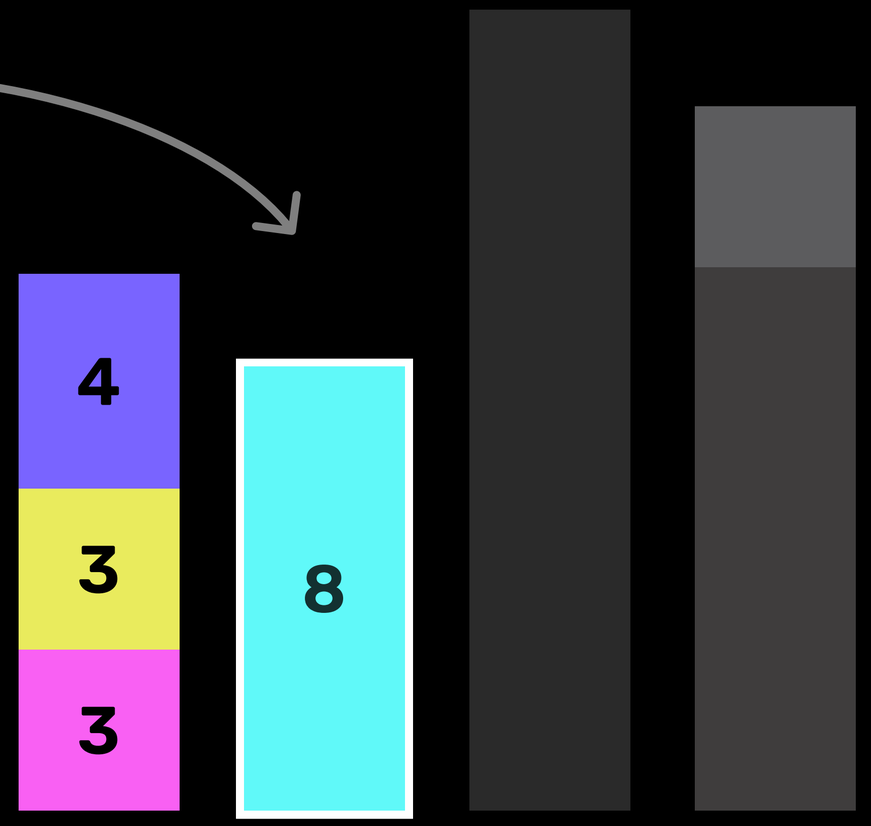


Правила игры

1. фт
2. нфт
3. оценка

⏳ 8 мин

high level дизайн

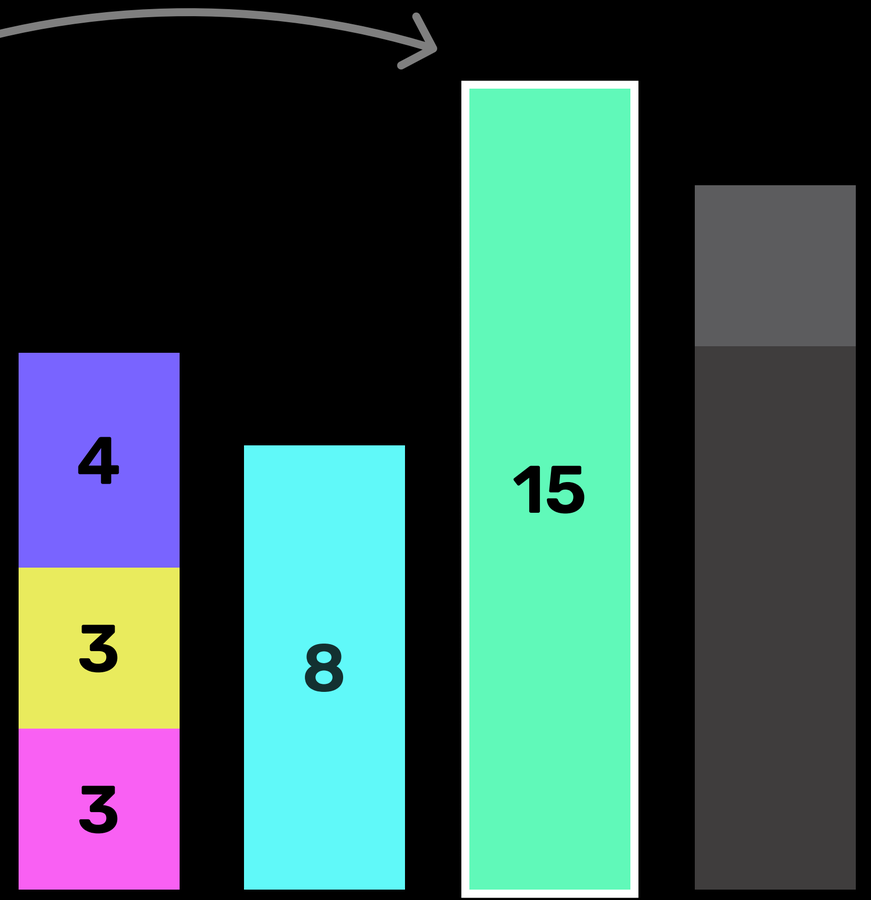


Правила игры

1. фт
2. нфт
3. оценка
4. HL дизайн

⏳ 15 мин

low level дизайн

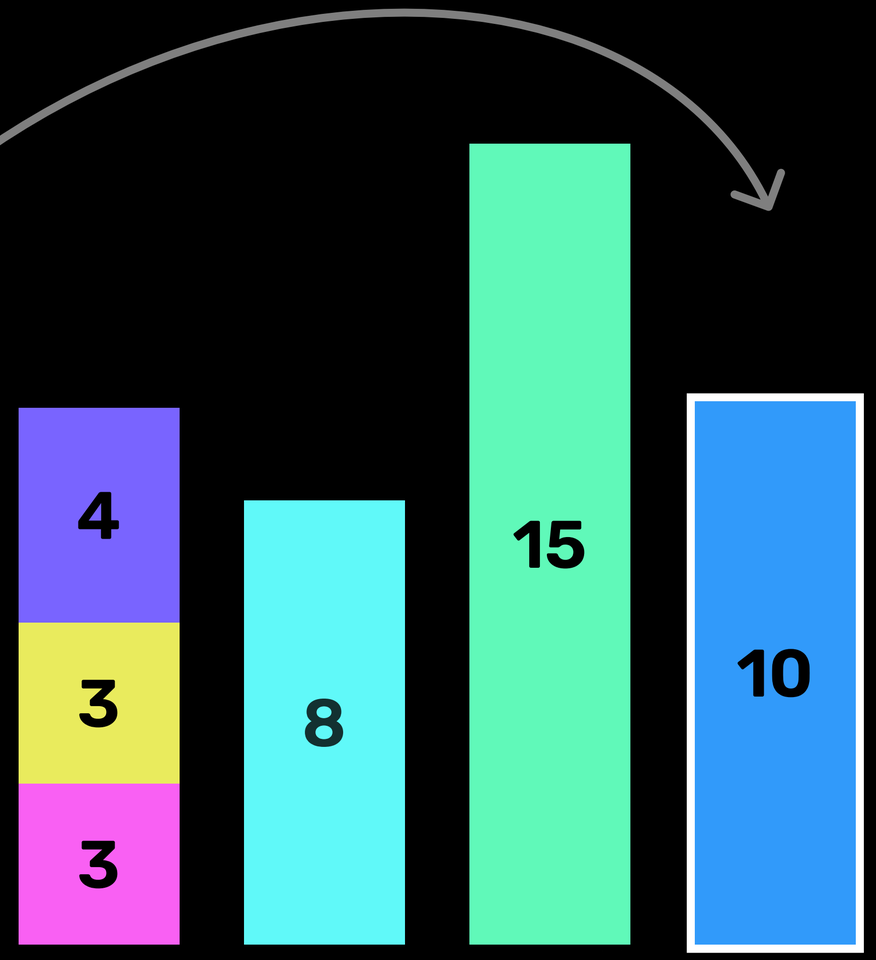


Правила игры

1. фт
2. нфт
3. оценка
4. HL дизайн
5. LL дизайн

⏳ 10 мин

погружение

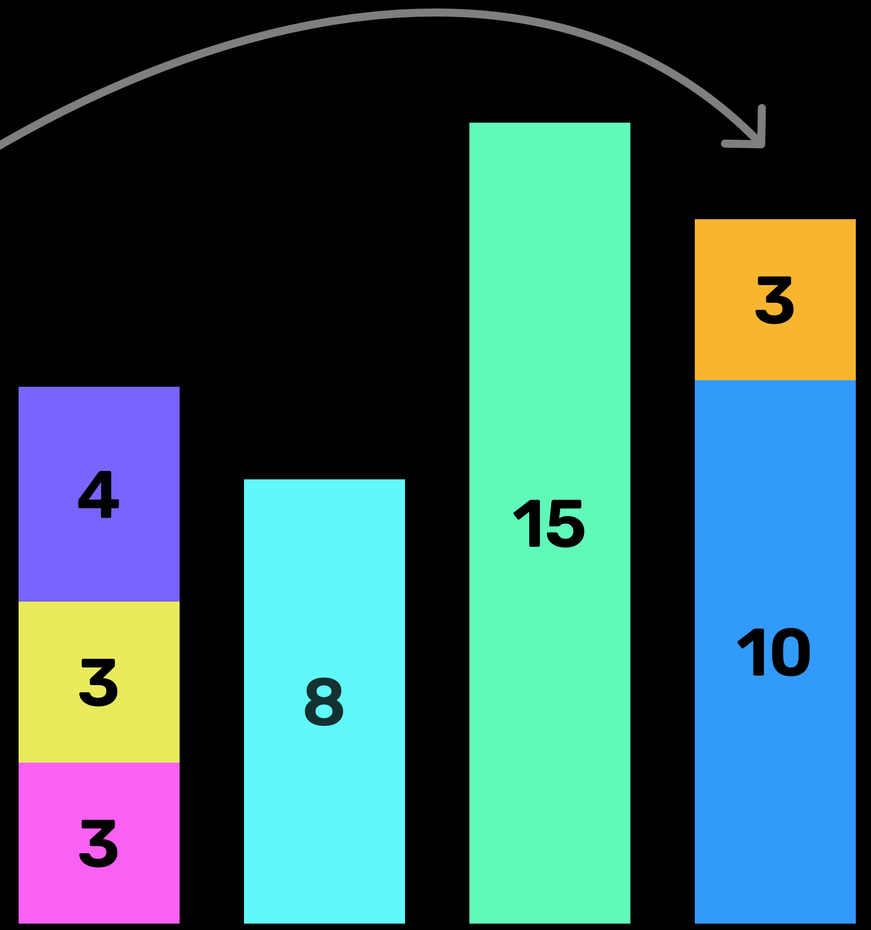


Правила игры

1. фт
2. нфт
3. оценка
4. HL дизайн
5. LL дизайн
6. погружение

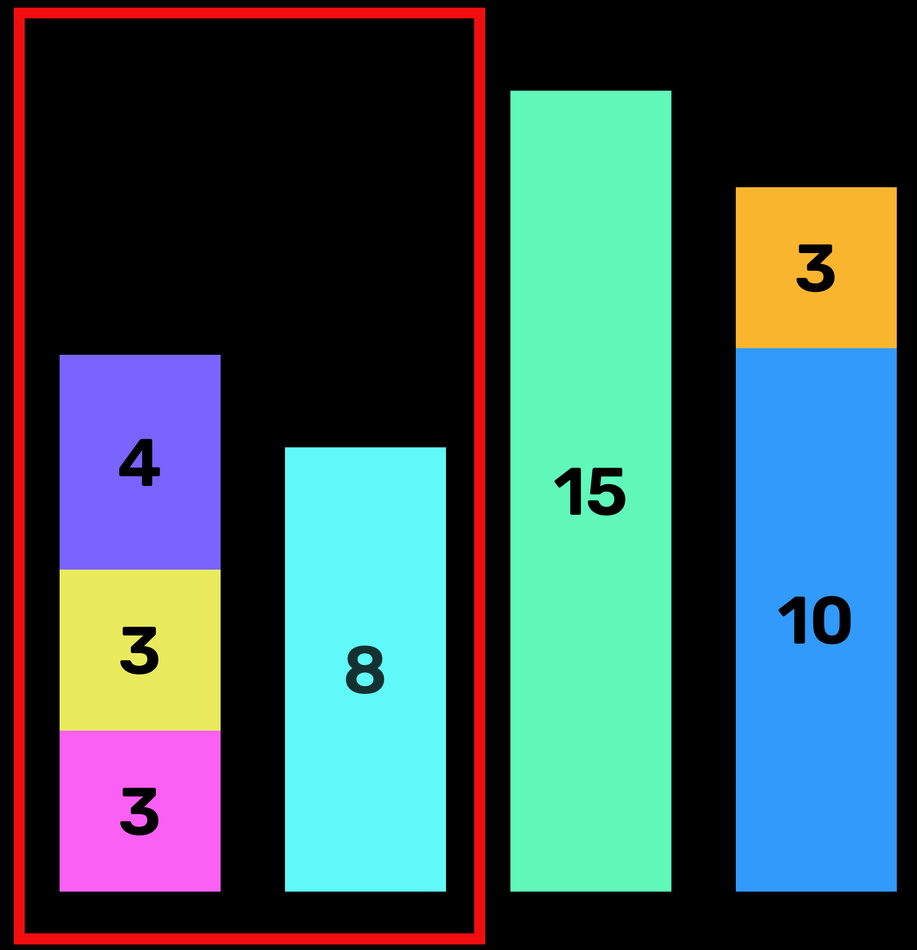
⏳ 3 мин

итог

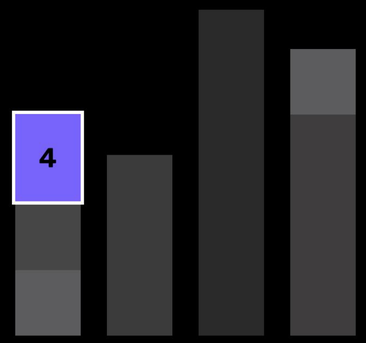


Правила игры

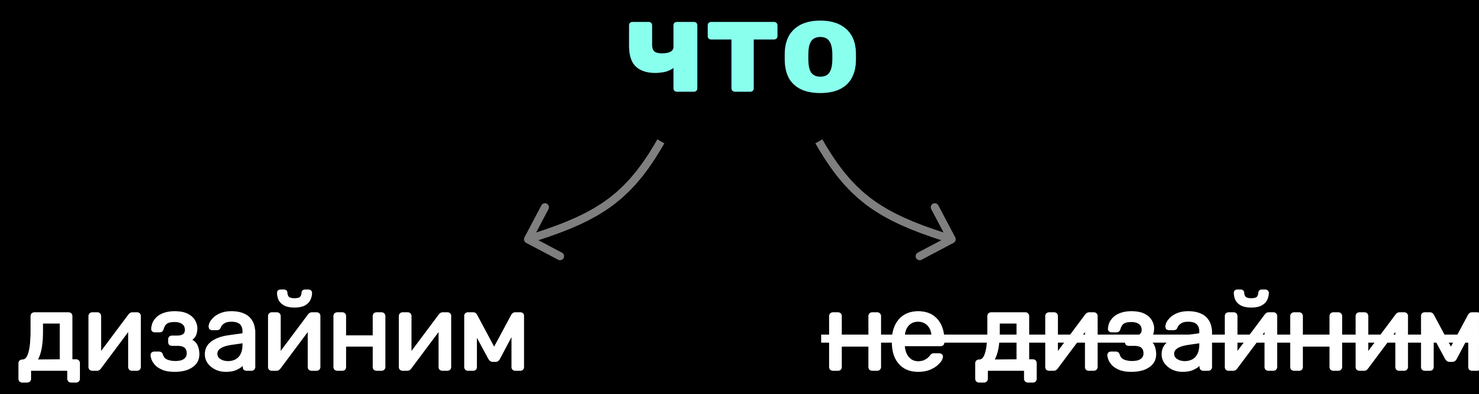
1. фт
2. нфт
3. оценка
4. HL дизайн
5. LL дизайн
6. погружение
7. итог



функциональные требования



узнать требования это ваша задача



⏳4 минуты
❓ что интервьюер реально хочет задизайнить?
📌 чем больше напишем, а затем уберем, тем меньше дизайнить

социальная сеть

✅

Лента новостей
Посты
Медиа файлы

Лайки
Комментарии
Подписка на друзей

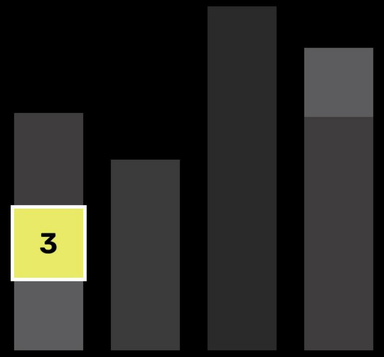
❌

~~Авторизация~~
~~Профиль~~
~~Получение списка друзей~~
~~Мониторинг~~
~~Логи~~
~~Секурити~~

YOU ARE A LEADER

вы ведете интервью

нефункциональные требования



сколько

❓ что за скейл?
❓ откуда будет нагрузка - масштабирование
❓ где самое слабое звено в системе

❓ масштабирование, шаржирование, репликация
⏳ 3 минуты
Daily/Monthly Active Users

DAILY & MONTHLY ACTIVE USERS

DAU/MAU

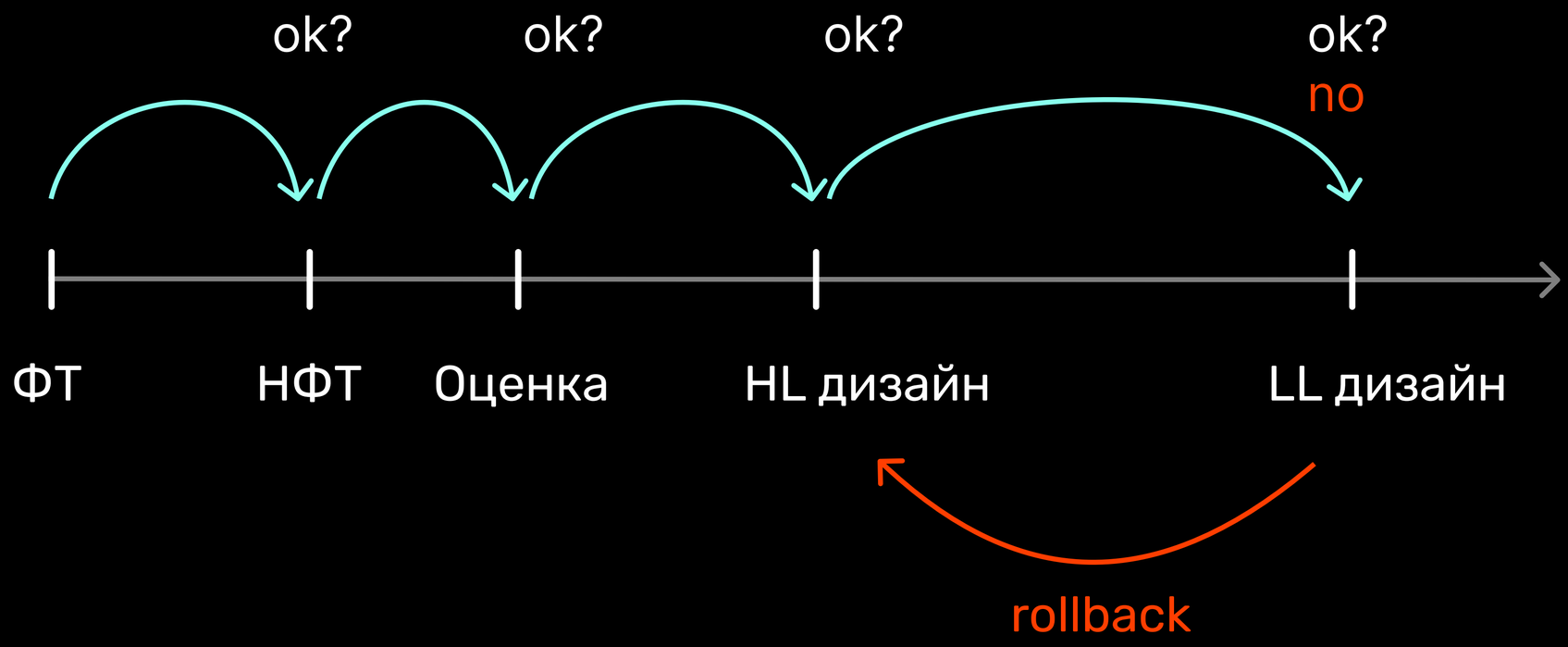
Лента новостей
Посты
Медиа файлы

Таймлайн - **5 раз в день**
DAU - **1 миллиард**
Посты - **10 миллионов** постов день
Друзей **500** в среднем
Latency < **200** мс
Durability > **Availability**
CAP → **Component**

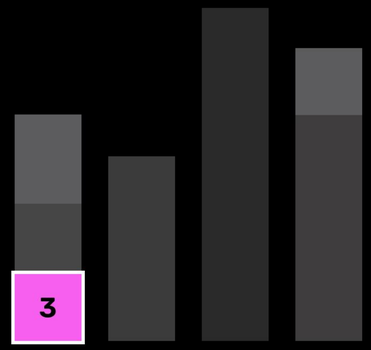
🤔 откуда будет нагрузка - масштабирование
❓ где самое слабое звено в системе

❓ масштабирование, шардирование, репликация
⏳ 3 минуты

чекпойнты



оценка



распределенная система?

📌 либо Parking Lot

❓ Traffic – Request Per Second (RPS)
❓ Storage – Daily/Monthly
❓ Bandwidth – Daily/Monthly

⏳ 3 минуты: 1 минута на каждый блок

TRAFFIC

Чтение: DAU * 5 / 10 ^ 5 =
10^9 * 5 / 10^5 = 5 * 10^4
50000 RPS
Запись: 10M / 10^5 = 10^7 / 10^5
100 RPS
Чтение к записи: 500 : 1

📌 3600 * 24 = 86400 ~ 10^5
❗ важно для выбора БД

STORAGE

10Kb * 10M = 10^4 * 10^7=10^11 bytes
100Gb Daily
30 * 100Gb = 3Tb Monthly

⏳1 минута
❓сколько храним данные? 3 - 5 лет?

📌продумать запас в 70%
❗важно для систем Google Drive, Dropbox, Youtube

COMMODITY HARDWARE

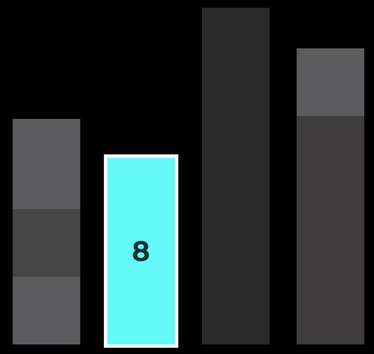
ожидание

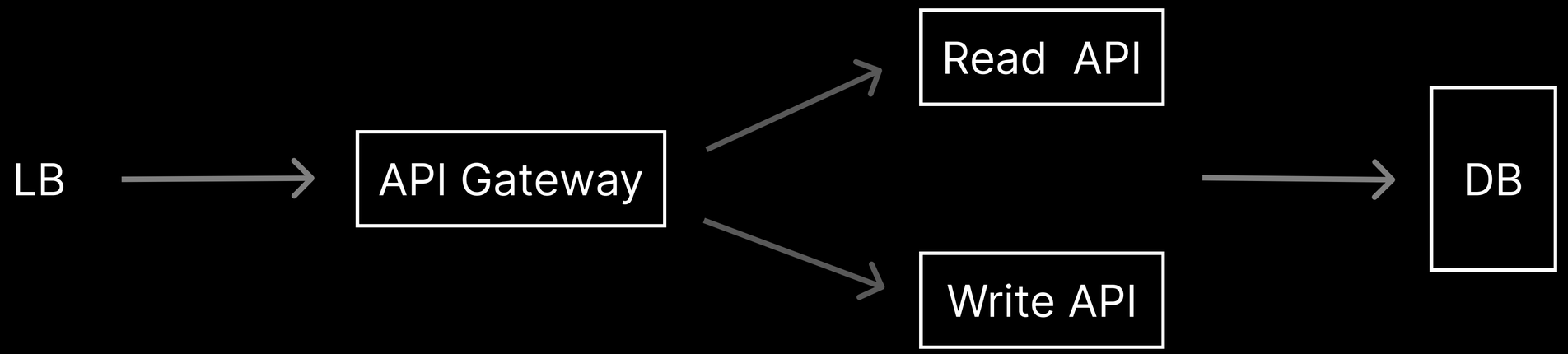
реальность

Диск 10Тб
RAM 256-512 Гб

📌 With 1 billion DAU system should be distributed

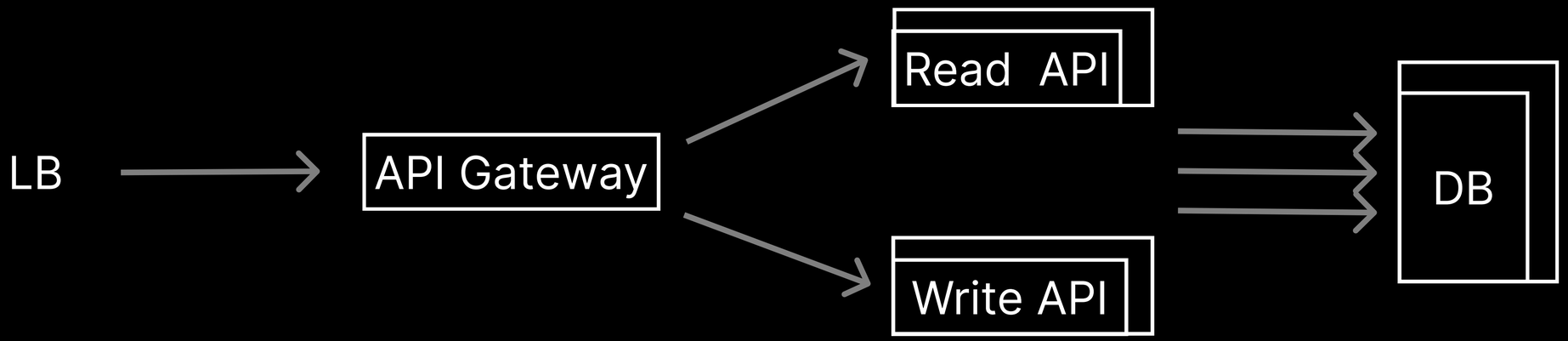
High Level дизайн





⏳8 минут
✏️ это первая схема за 12 мин дизайна

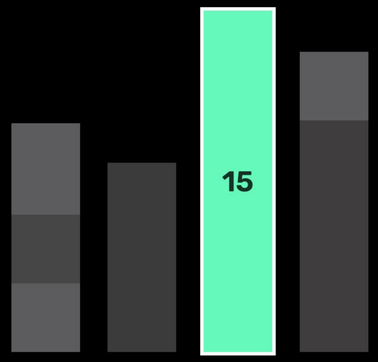
📌 легко скейлить
👤 вместо LB можно поставить человечка



📌 легко добавлять сущности

📌 легко погрузиться в каждый блок

Low Level дизайн



4

3

3

8

15

3

10

Rate Limiter ID Generation Consistent Hashing

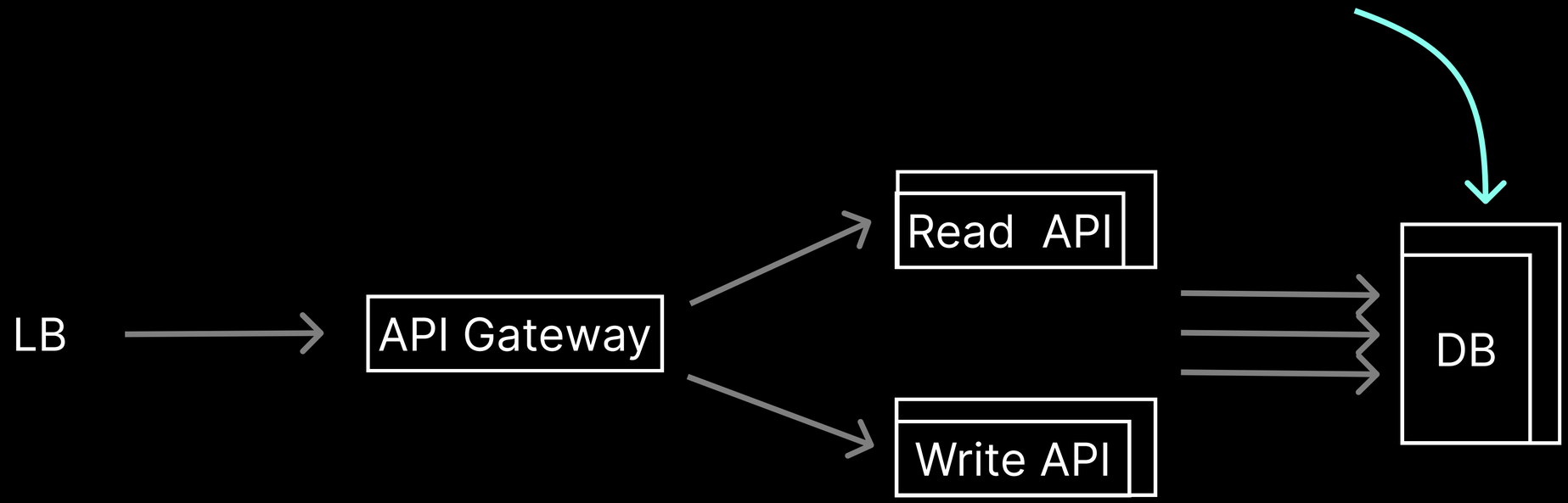
Key-Value Store News Feed Chat GeoHash

Search Autocomplete System S3

📌 компоненты Low level дизайна

Это БАЗА

Выбрать конкретную БД



Key Value,
RDBMS,
Document based,
Wide Column,
Column based,
Time series,
SQL, NoSQL,
NewSQL, ACID

BTree

Чтение

LSM Tree

Запись

📌 нужен ли SQL
📌 нужна ли схема данных

Key Value
RocksDB, Redis

RDBMS
Postgres, MySQL

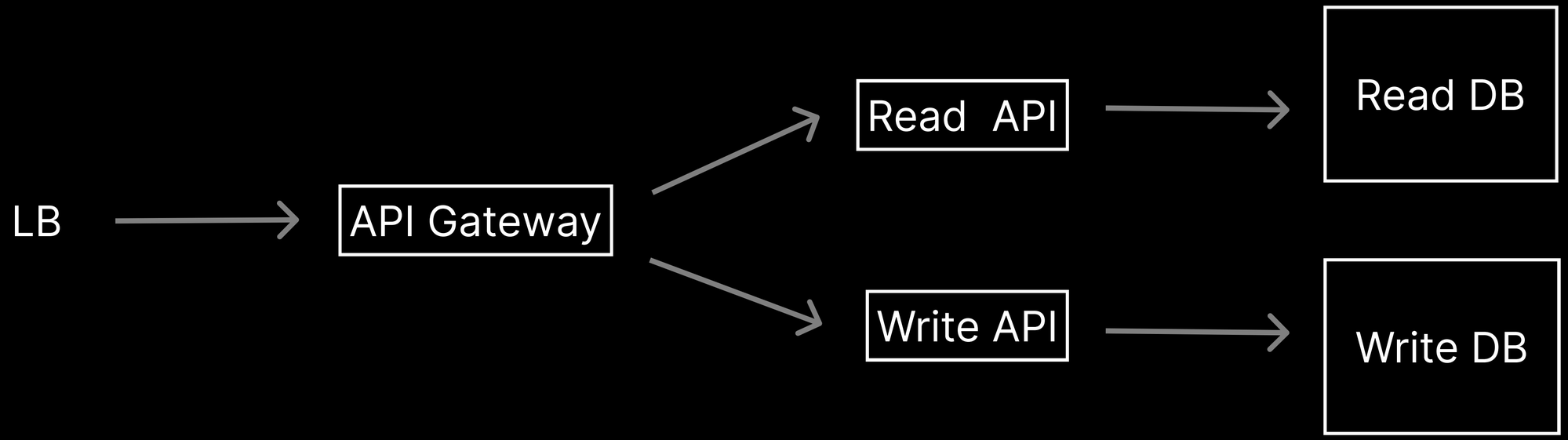
Document based
DynamoDB, MongoDB

Column based
DuckDB, Clickhouse

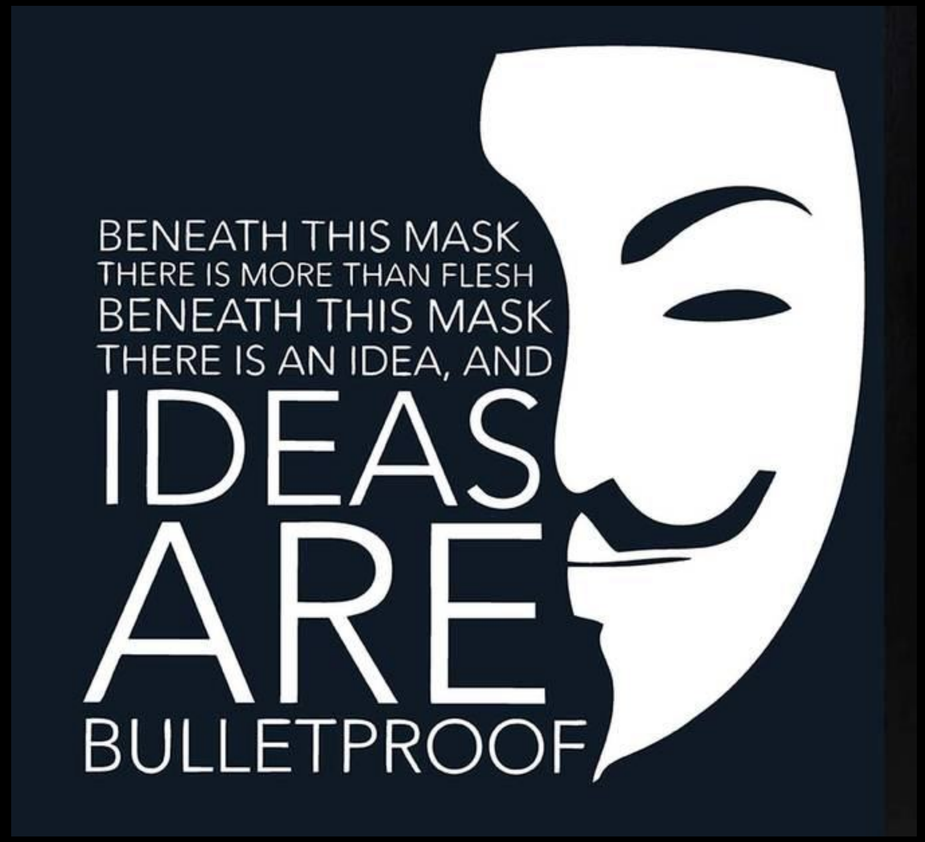
Time series
Clickhouse, Driud

SQL/NoSQL/NewSQL
CocroachDB

Wide Column
Cassandra


LB
API Gateway
Read API
Write API
Read DB
Write DB

Идеи а не технологии



❌

Postgres

✅

BTree база данных как Postgres или другая база данных, оптимизированная на чтение

❌

MongoDB

✅

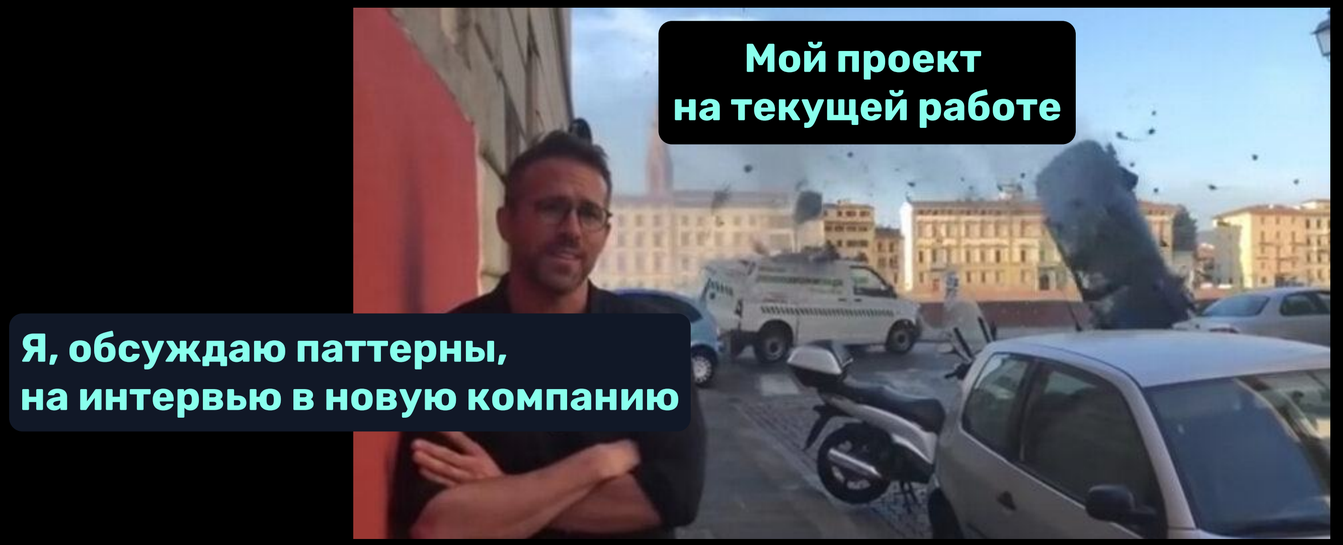
Документно ориентированая
база данных
с мульти мастер репликацией
и шардингом

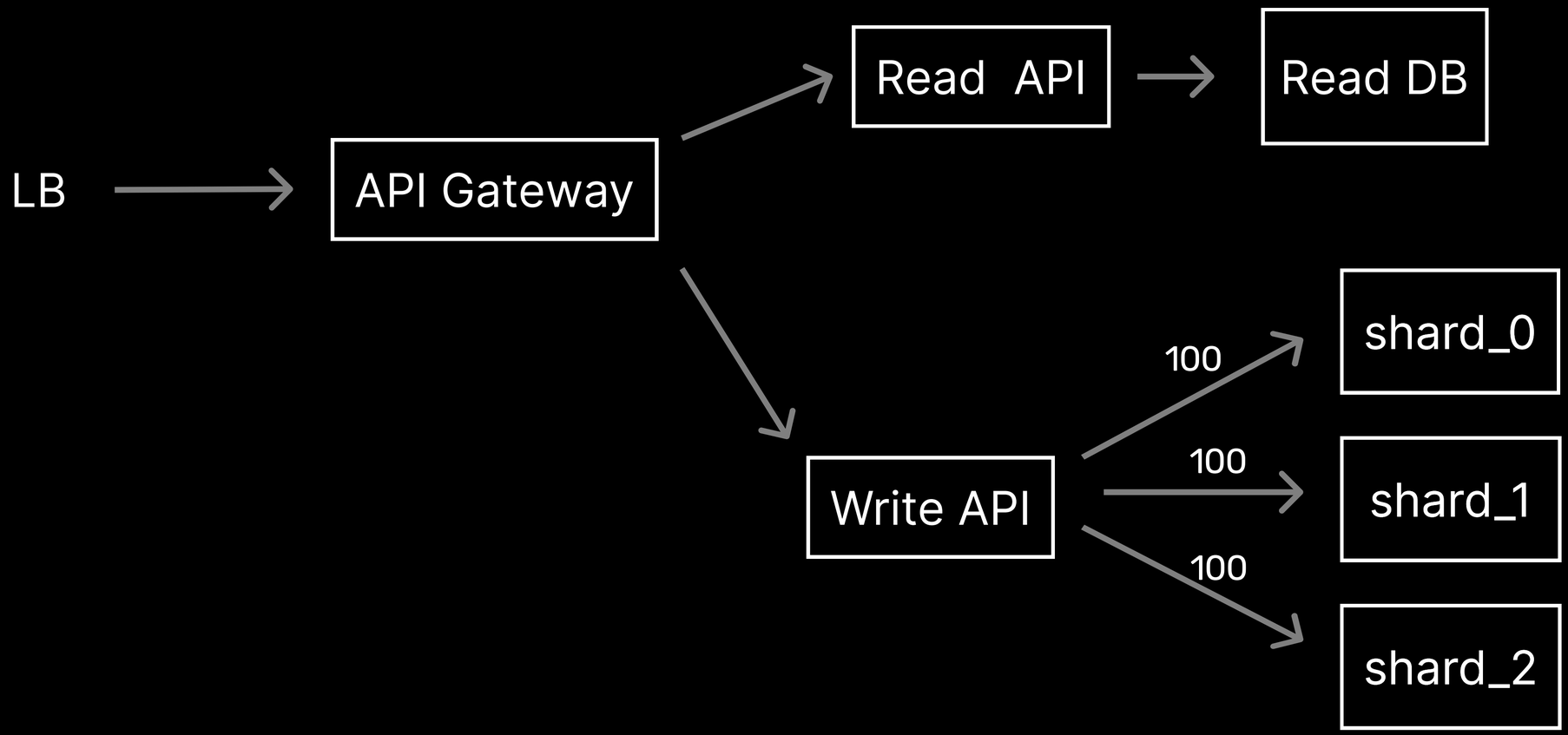
✅

“я не работал с Cassandra, но использовал HBase в одном из Pet проектов”

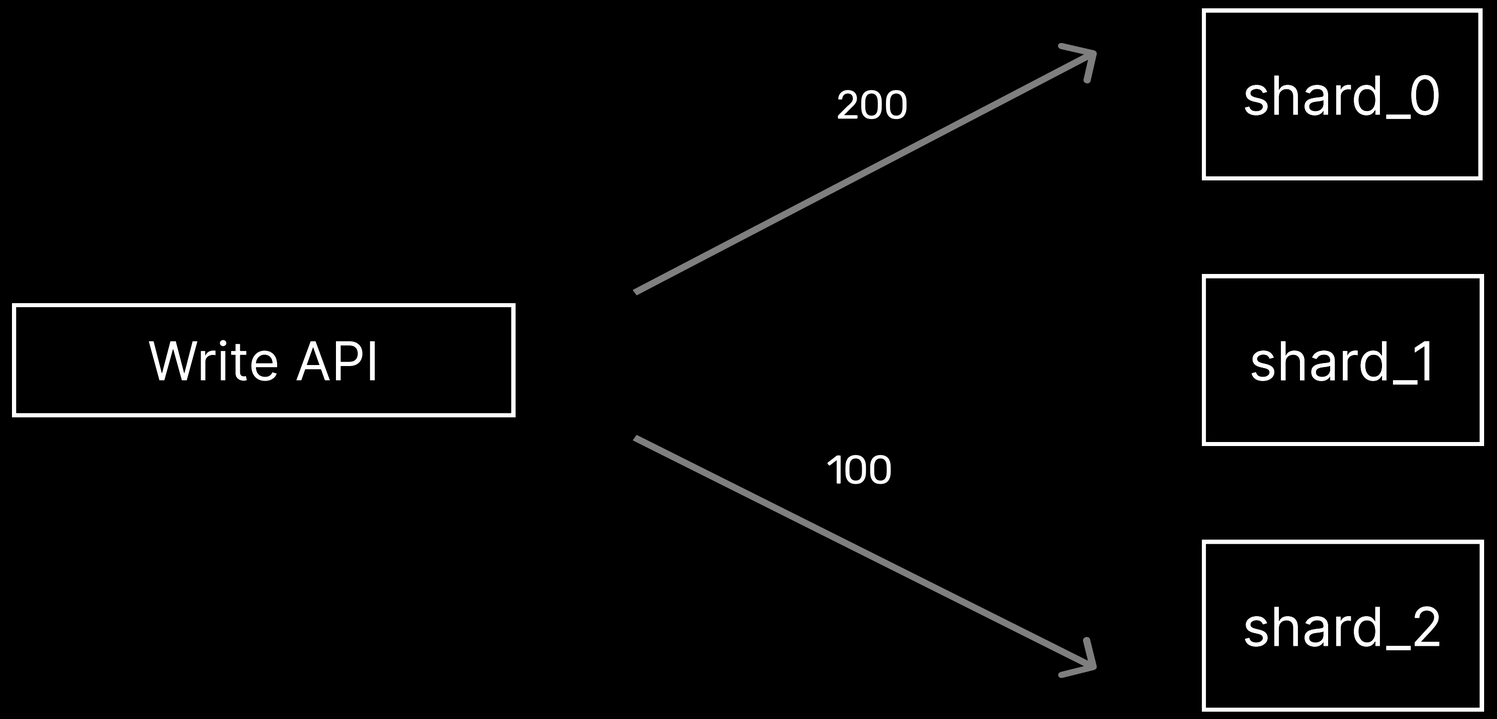
Ведем интервью в сторону подхода в котором вы разбираетесь

Каскадное падение

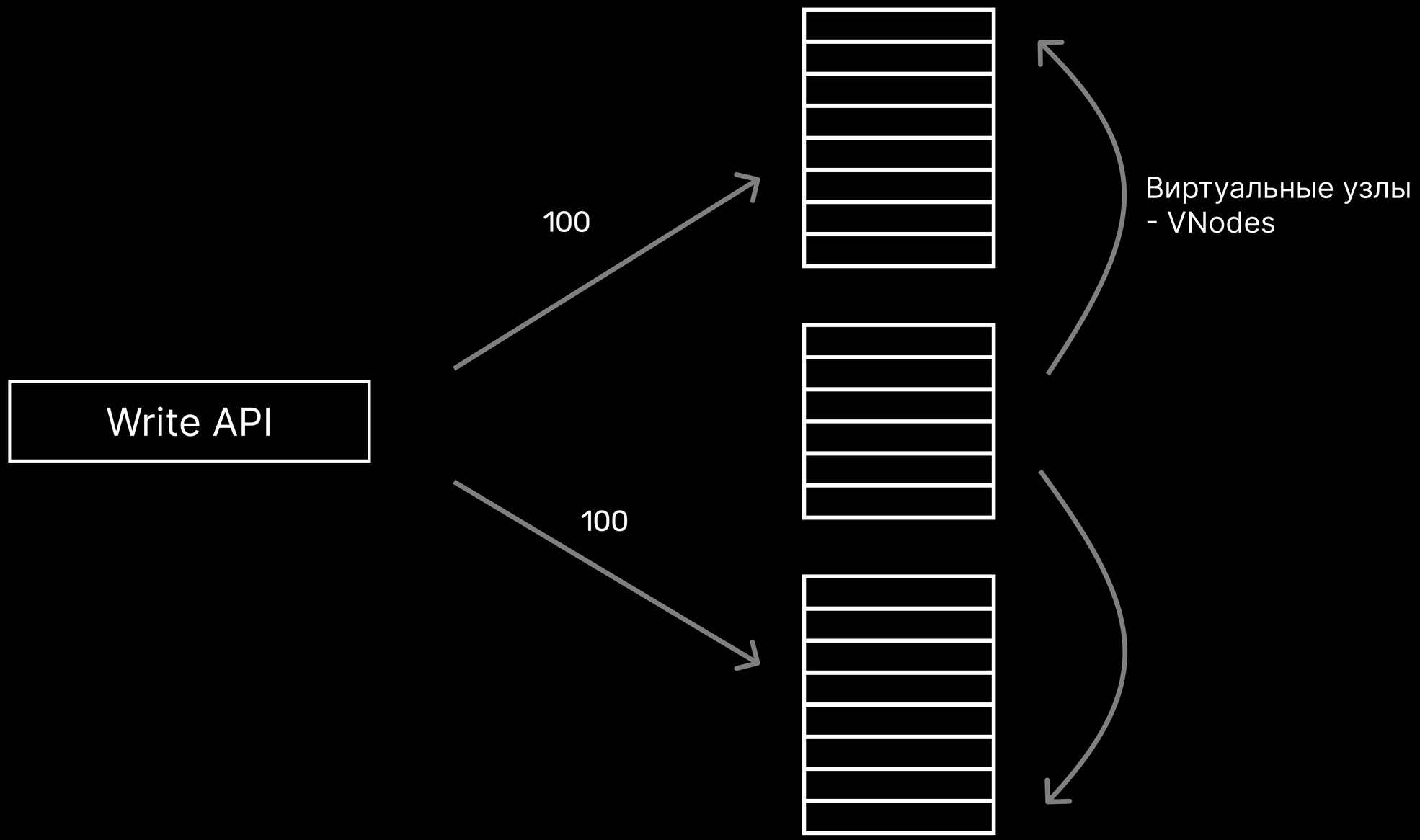




📌 что если вы распределили данные по шардам
и один из шардов упал?

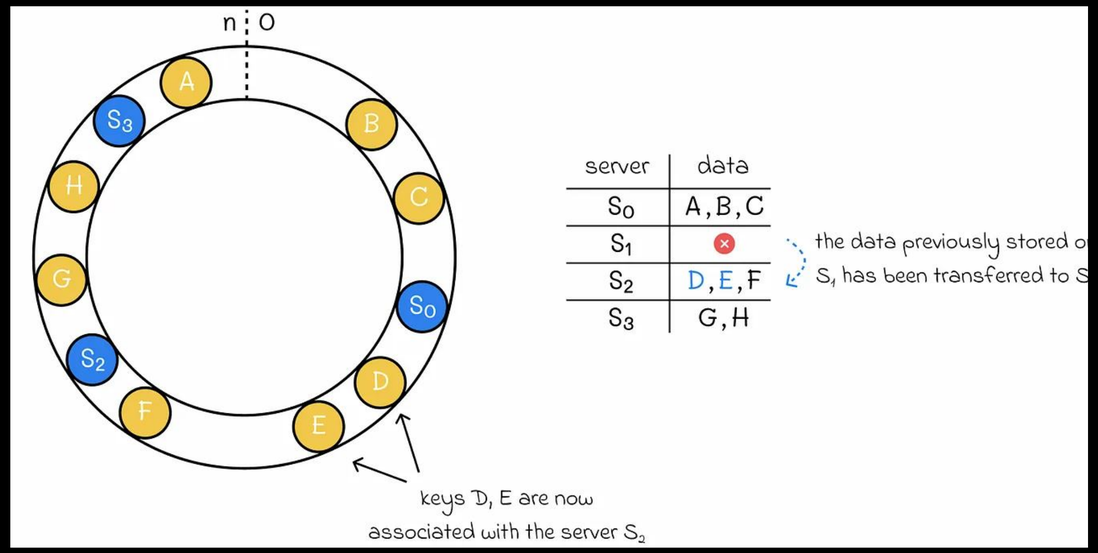


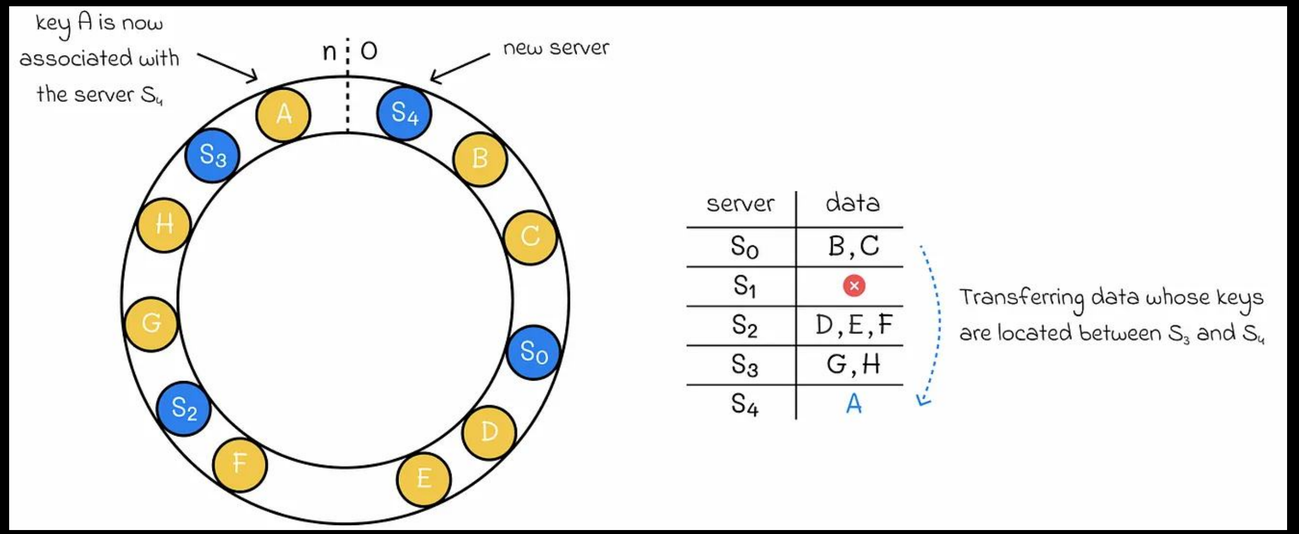
📌 нагрузка на один из шардов удвоилась



📌 а что если мы создадим больше виртуальных шардов чем серверов?

CONSISTENT HASHING





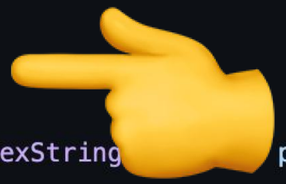
https://towardsdatascience.com/system-design-consistent-hashing-43ddf48d2d32

```java
@Override
public List<Node> routeRequest(byte[] key) {
    List<Integer> partitionList = getPartitionList(key);

    if(partitionList.size() == 0)
        return new ArrayList<Node>(0);
    // pull out the nodes corresponding to the target partitions
    List<Node> preferenceList = new ArrayList<Node>(partitionList.size());
    for(int partition: partitionList) {
        preferenceList.add(partitionToNode[partition]);
    }
    if(logger.isDebugEnabled()) {
        List<Integer> nodeIdList = new ArrayList<Integer>();
        for(int partition: partitionList) {
            nodeIdList.add(partitionToNode[partition].getId());
        }
        logger.debug("Key " + ByteUtils.toHexString(key) + " mapped to Nodes " + no
                     + " Partitions " + partitionList);
    }
    return preferenceList;
}
```

- Amazon Dynamo
  - Voldemort
  - Cassandra

Этот подход используется в Amazon Dynamo, Cassandra и Voldemort
https://github.com/voldemort/voldemort/blob/master/src/java/voldemort/routing/ConsistentRoutingStrategy.java

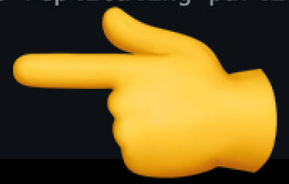
```java
@Override
public List<Integer> getPartitionList(byte[] key) {
    // hash the key and perform a modulo on the total number of partitions,
    // to get the master partition
    int index = getMasterPartition(key);
    if(logger.isDebugEnabled()) {
        logger.debug("Key " + ByteUtils.toHexString primary partition " + index
    }
    // Now based on the preference list, pick the replicating partitions and
    // return
    return getReplicatingPartitionList(index);
}
```



```java
@Override
public Integer getMasterPartition(byte[] key) {
    return abs(hash.hash(key)) % (Math.max(1, this.partitionToNode.length));
}
```

https://github.com/voldemort/voldemort/blob/master/src/java/voldemort/routing/ConsistentRoutingStrategy.java

```java
@Override
public List<Integer> getPartitionList(byte[] key) {
    // hash the key and perform a modulo on the total number of partitions,
    // to get the master partition
    int index = getMasterPartition(key);
    if(logger.isDebugEnabled()) {
        logger.debug("Key " + ByteUtils.toHexString(key) + " primary partition " + index
    }
    // Now based on the preference list, pick the replicating partitions and
    // return
    return getReplicatingPartitionList(index);
}
```

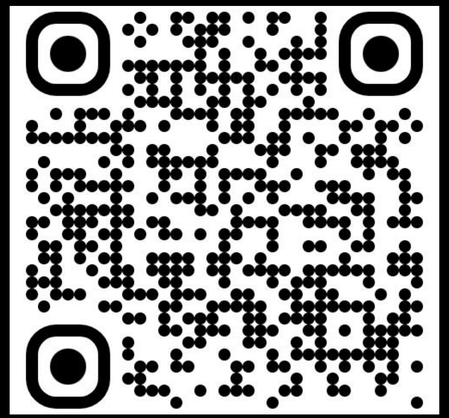


https://github.com/voldemort/voldemort/blob/master/src/java/voldemort/routing/ConsistentRoutingStrategy.java

```java
public List<Integer> getReplicatingPartitionList(int index) {
    List<Node> preferenceList = new ArrayList<Node>(numReplicas);
    List<Integer> replicationPartitionsList = new ArrayList<Integer>(numReplicas);

    if(partitionToNode.length == 0) {
        return new ArrayList<Integer>(0);
    }
    // go over clockwise to find the next 'numReplicas' unique nodes
    // to replicate to
    for(int i = 0; i < partitionToNode.length; i++) {
        // add this one if we haven't already
        if(!preferenceList.contains(partitionToNode[index])) {
            preferenceList.add(partitionToNode[index]);
            replicationPartitionsList.add(index);
        }

        // if we have enough, go home
        if(preferenceList.size() >= numReplicas)
            return replicationPartitionsList;
        // move to next clockwise slot on the ring
        index = (index + 1) % partitionToNode.length;
    }

    // we don't have enough, but that may be okay
    return replicationPartitionsList;
```

https://github.com/voldemort/voldemort/blob/master/src/java/voldemort/routing/ConsistentRoutingStrategy.java

VOLDEMORT

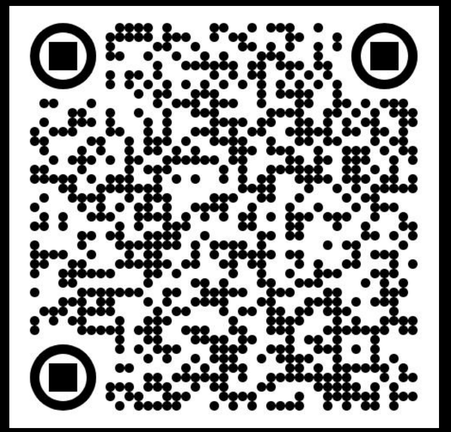


Linkedin: “People You May Know”

CASSANDRA

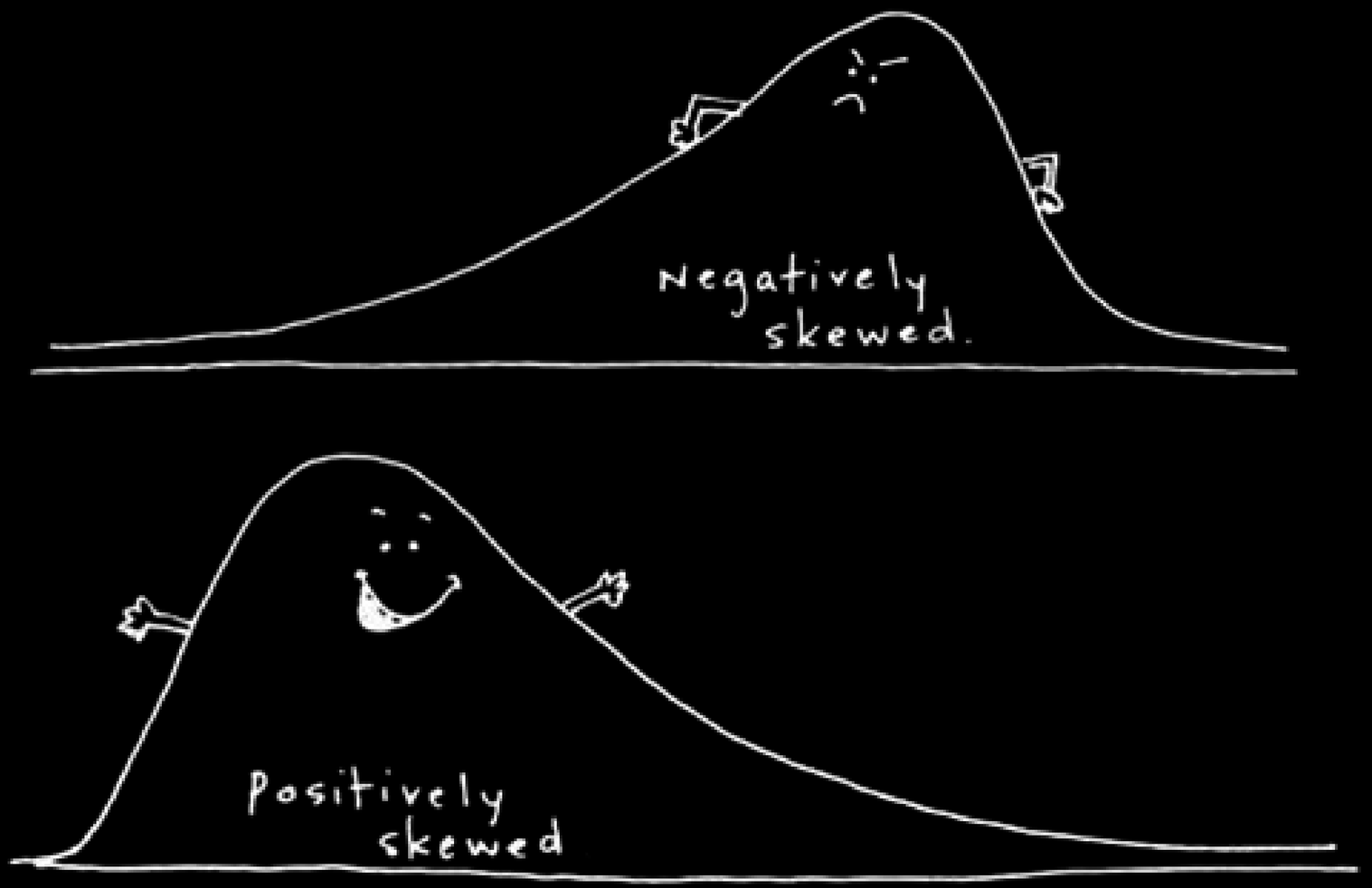
Facebook Inbox FEATURE

DYNAMO



Shopping cart at **Amazon**

“Перекос данных”



Что делать если шард большой?

```python
DISCORD_EPOCH = 1420070400000
BUCKET_SIZE = 1000 * 60 * 60 * 24 * 10


def make_bucket(snowflake):
    if snowflake is None:
        timestamp = int(time.time() * 1000) - DISCORD_EPOCH
    else:
        # When a Snowflake is created it contains the number of
        # seconds since the DISCORD_EPOCH.
        timestamp = snowflake_id >> 22
    return int(timestamp / BUCKET_SIZE)


def make_buckets(start_id, end_id=None):
    return range(make_bucket(start_id), make_bucket(end_id) + 1)
```

```
CREATE TABLE messages (
    channel_id bigint,
    bucket int,
    message_id bigint,
    author_id bigint,
    content text,
    PRIMARY KEY ((channel_id, bucket), message_id)
) WITH CLUSTERING ORDER BY (message_id DESC);
```

if we stored about 10 days of messages within a bucket that we could comfortably stay under 100MB

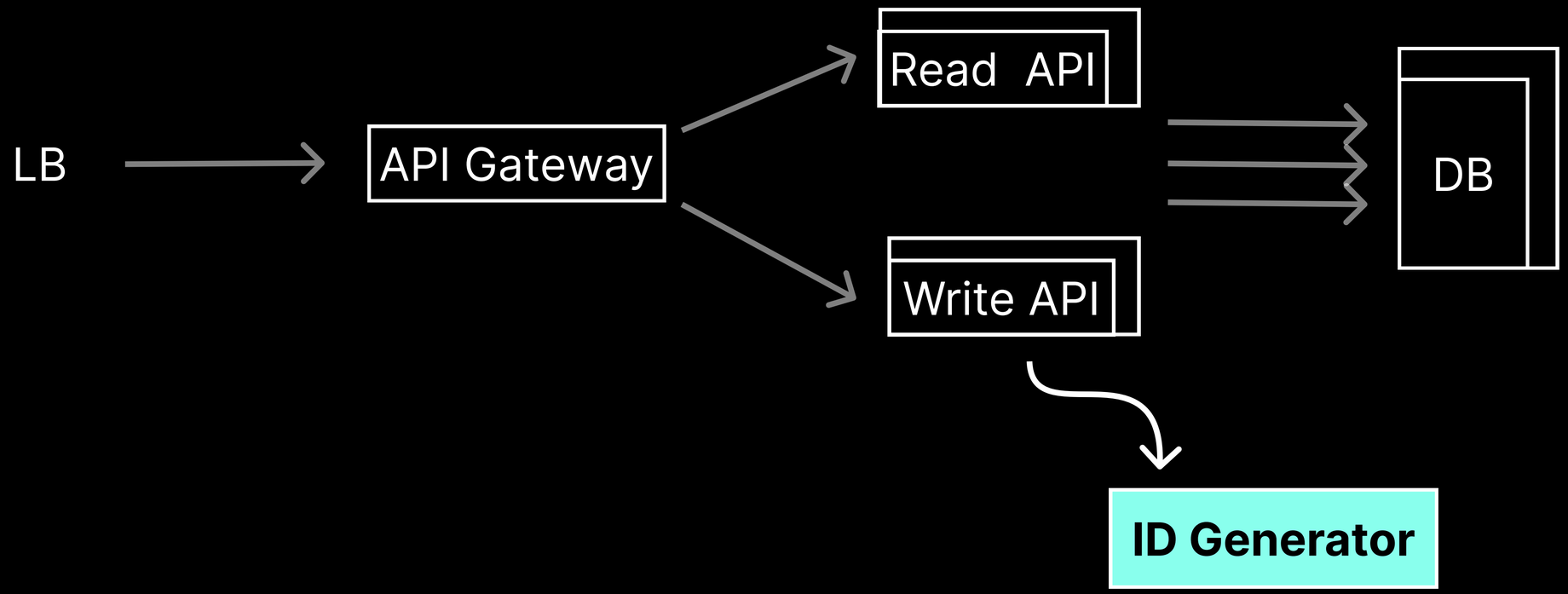
https://discord.com/blog/how-discord-stores-billions-of-messages

Что за Snowflake такой?

```python
DISCORD_EPOCH = 1420070400000
BUCKET_SIZE = 1000 * 60 * 60 * 24 * 10


def make_bucket(snowflake):
    if snowflake is None:
        timestamp = int(time.time() * 1000) - DISCORD_EPOCH
    else:
        # When a Snowflake is created it contains the number of
        # seconds since the DISCORD_EPOCH.
        timestamp = snowflake_id >> 22
    return int(timestamp / BUCKET_SIZE)


def make_buckets(start_id, end_id=None):
    return range(make_bucket(start_id), make_bucket(end_id) + 1)
```

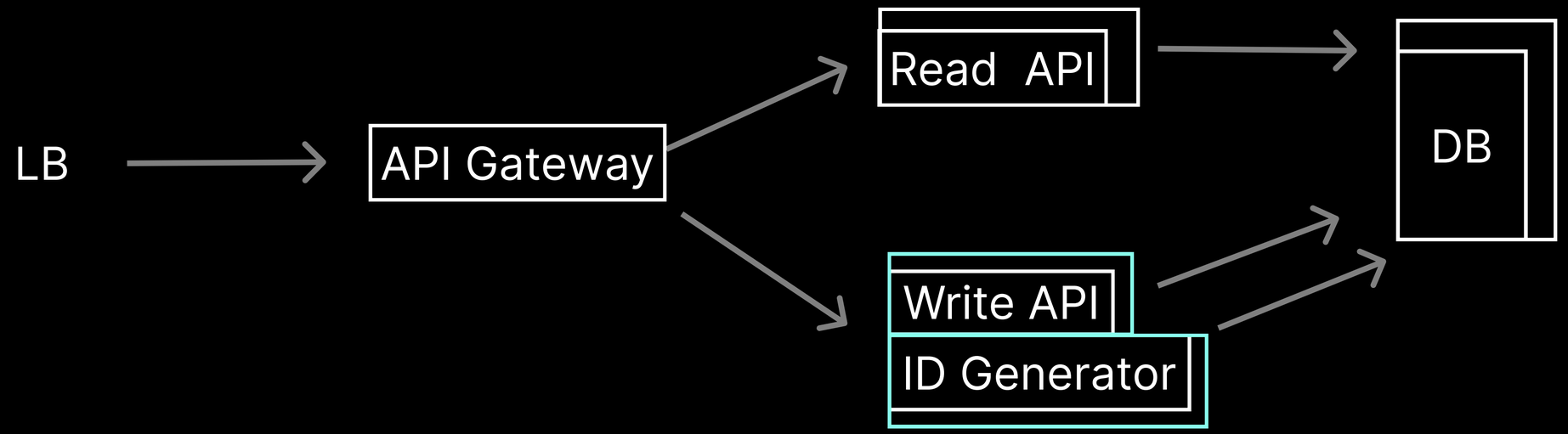
https://discord.com/blog/how-discord-stores-billions-of-messages

ID генератор



📌 single server
📌 можно разделить по четным и нечетным ID
📌 сложно масштабировать - как добавить новые?

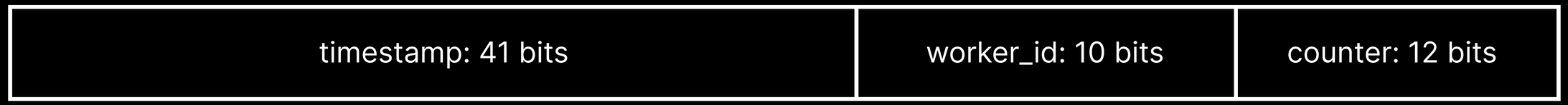
JAVA.UTIL.UUID




```
jshell> java.util.UUID.randomUUID()
31ee5e0b-630f-44a2-93ed-80e4f9046183
```

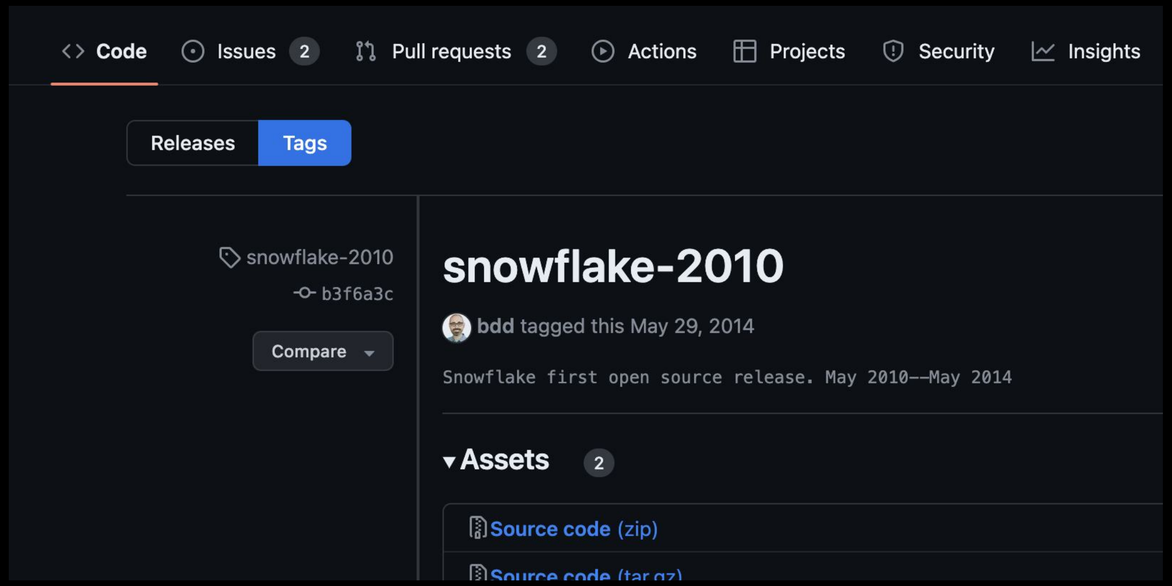
- 📌 Нет казуальности - невозможно упорядочить
- 📌 PK обычно упорядоченные
- 📌 UUID v4 рандомный, не числовой
- 📌 Слишком большой - 128 бит

SNOWFLAKE ID



📌 кастомный epoch_time

📌 Unix timestamp - только 1000 ID в секунду



IdWorker.scala

src > main > scala > com > twitter > service > snowflake > IdWorker.scala

```
16   extends Snowflake.Iface {
70
71   protected[snowflake] def nextId(): Long = synchronized {
72     var timestamp = timeGen()
73
74     if (timestamp < lastTimestamp) {
75       exceptionCounter.incr(1)
76       log.error("clock is moving backwards.  Rejecting requests until
77       throw new InvalidSystemClock("Clock moved backwards.  Refusing
78         lastTimestamp - timestamp))
79     }
80
81     if (lastTimestamp == timestamp) {
82       sequence = (sequence + 1) & sequenceMask
83       if (sequence == 0) {
84         timestamp = tilNextMillis(lastTimestamp)
85       }
86     } else {
87       sequence = 0
88     }
89
90     lastTimestamp = timestamp
91     ((timestamp - twepoch) << timestampLeftShift) |
92       (datacenterId << datacenterIdShift) |
93       (workerId << workerIdShift) |
94       sequence
95   }
```

```
CREATE OR REPLACE FUNCTION insta5.next_id(OUT result bigint) AS
DECLARE
    our_epoch bigint := 1314220021721;
    seq_id bigint;
    now_millis bigint;
    shard_id int := 5;
BEGIN
    SELECT nextval('insta5.table_id_seq') %% 1024 INTO seq_id;
    SELECT FLOOR(EXTRACT(EPOCH FROM clock_timestamp()) * 1000)
INTO now_millis;
    result := (now_millis - our_epoch) << 23;
    result := result | (shard_id <<10);
    result := result | (seq_id);
END;
```

```
CREATE TABLE insta5.our_table (
    "id" bigint NOT NULL DEFAULT insta5.n
    ...rest of table schema...
  )
```

https://instagram-engineering.com/sharding-ids-at-instagram-1cf5a71e5a5c

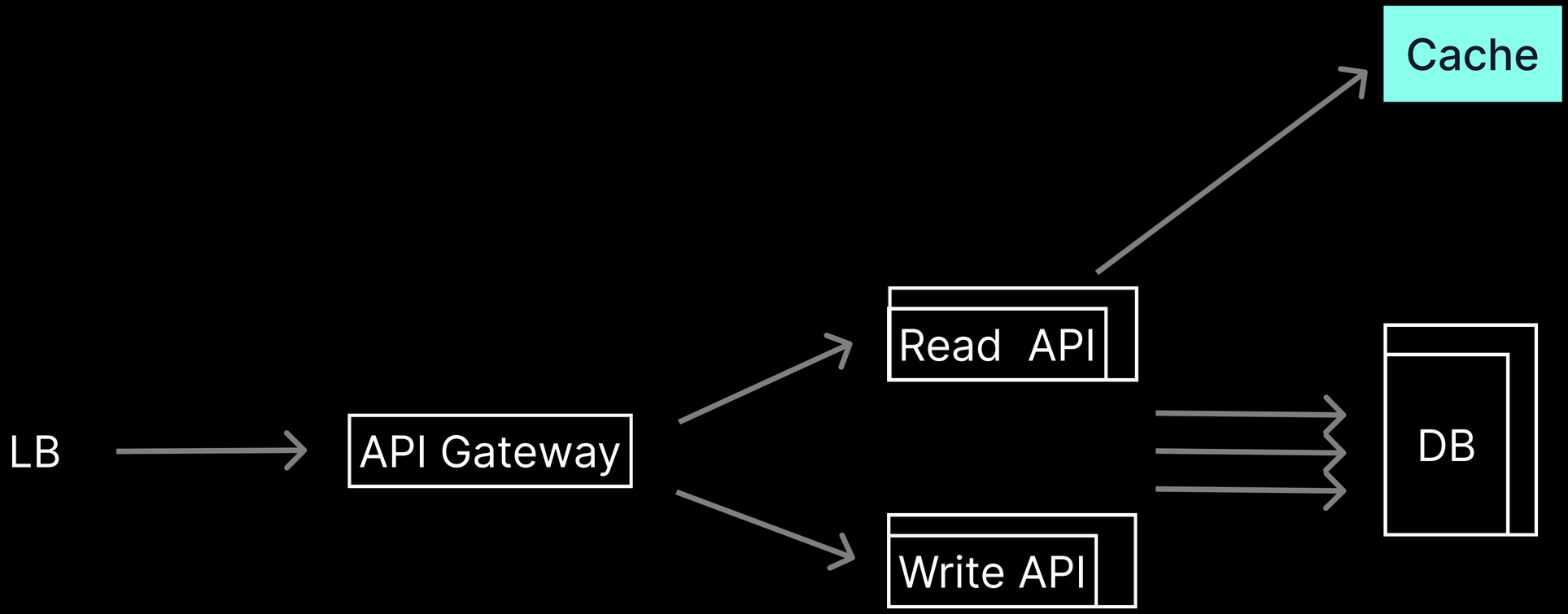
```
CREATE OR REPLACE FUNCTION insta5.next_id(OUT result bigint) AS
DECLARE
    our_epoch bigint := 1314220021721;
    seq_id bigint;
    now_millis bigint;
    shard_id int := 5;
BEGIN
    SELECT nextval('insta5.table_id_seq') %% 1024 INTO seq_id;
    SELECT FLOOR(EXTRACT(EPOCH FROM clock_timestamp()) * 1000)
INTO now_millis;
    result := (now_millis - our_epoch) << 23;
    result := result | (shard_id <<10);
    result := result | (seq_id);
END;
```

Шардирование по ID !!!

https://instagram-engineering.com/sharding-ids-at-instagram-1cf5a71e5a5c

Кэширование

? А нужно ли шардирование?


Cache
Read API
LB
API Gateway
DB
Write API

Какой кэш?

First In First Out (FIFO)

Last In First Out (LIFO)

Least Recently Used (LRU)

Most Recently Used (MRU)

Least Frequently Used (LFU)

Adaptive replacement cache(ARC)

Random Replacement (RR)

Write-through cache

Write-around cache

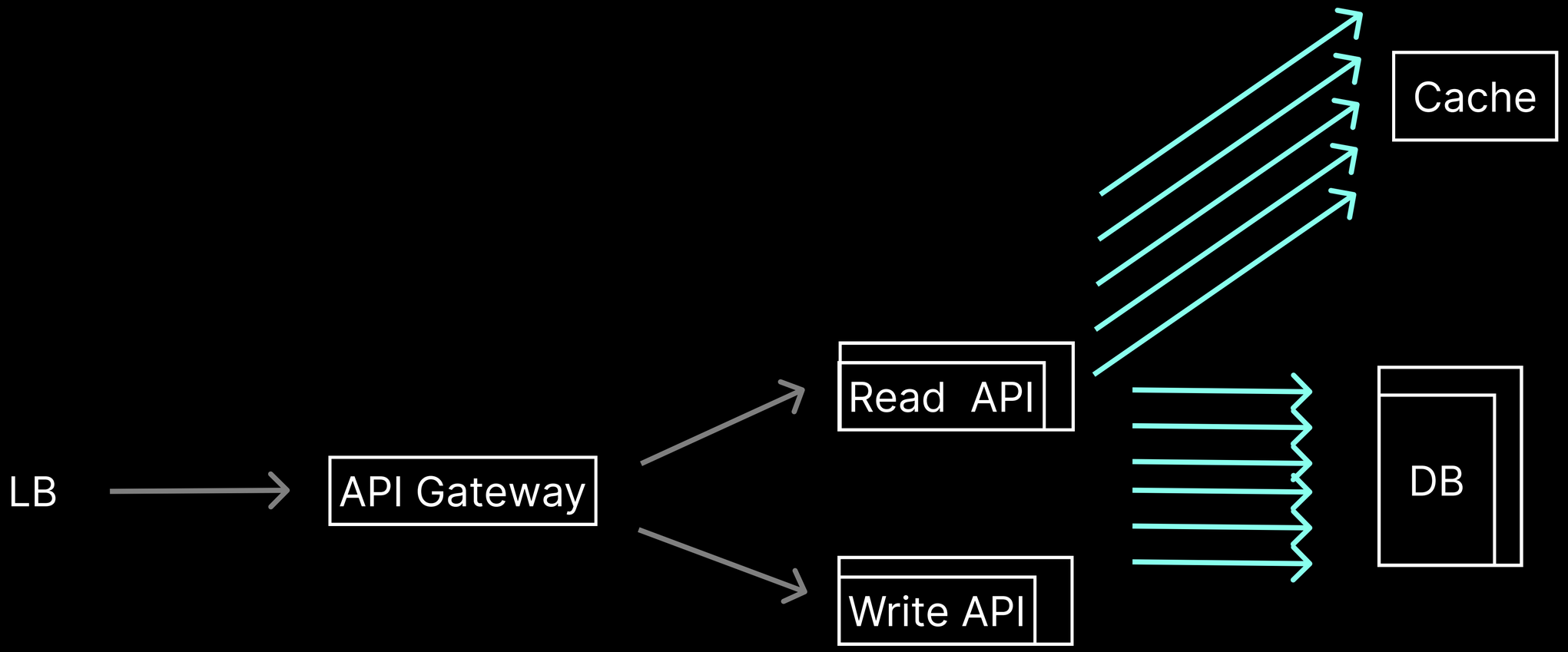
Write-back cache

ARC

```java
public class ARCMemoryCache
    extends AbstractMemoryCache
{
    private static final long serialVersionUID = 1L;

    private static final Log log = LogFactory.getLog( ARCMemoryCache.class );

    // private int[] loc = new int[0];

    // maximum size
    private int maxSize;

    private DoubleLinkedList T1 = new DoubleLinkedList();

    private DoubleLinkedList T2 = new DoubleLinkedList();

    private DoubleLinkedList B1 = new DoubleLinkedList();

    private DoubleLinkedList B2 = new DoubleLinkedList();
```

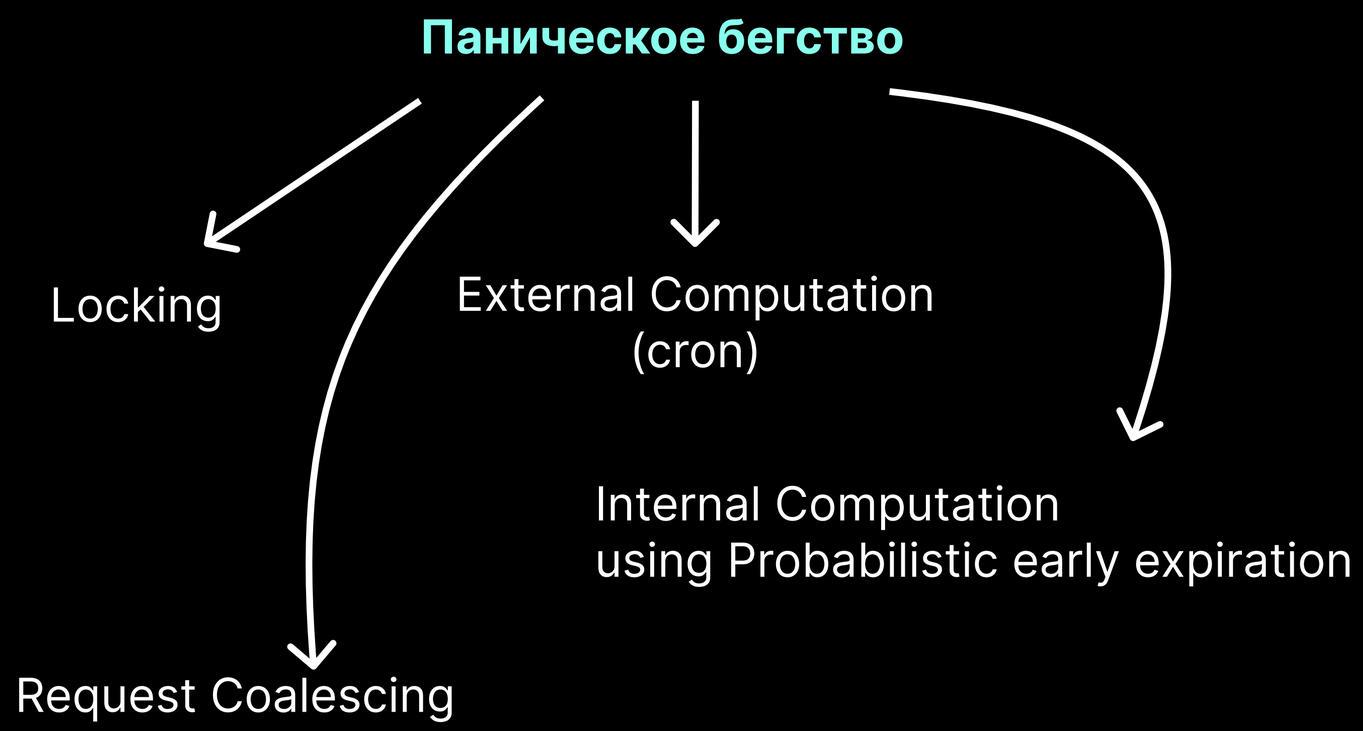
```java
170         /**
171          * This is the primary method for the ARC. It handles both puts and gets.
172          * <p>
173          * The ARC has 4 linked lists: T1, T2, B1, and B2. The 'T' lists are tops
174          * and the 'B' lists are bottoms. Bottom lists do not hold object, only
175          * keys.
176          * <p>
177          * The T1 list is an LRU (Least Recently Used) list. The T2 list is a near
178          * LFU (Least Frequently Used) list.
179          * <p>
180          * After items are removed from T1 and T2, their keys are stored in B1 and
181          * B2. The number of keys in B1 and B2 is restricted to the number of max
182          * items.
183          * <p>
184          * When there is a put or a get for an item whose key exists on one of the
185          * bottom lists, the maximum number of items in T1 is adjusted. If the item
186          * was found on B2 (the bottom LFU list) the maximum allowed in T1 (the top
187          * LRU list) is reduced. If the item is found in B1 list (the bottom LRU)
188          * the maximum allowed in T1 is increased.
189          * <p>
```

ARC has a better performance than LRU. It is achieved by keeping both the most frequently and frequently used entries, as well as a history for eviction. (Keeping MRU+MFU+eviction history.)



? А что если ключ из кеша только что пропал
и пришло очень большое кол-во запросов по этому ключу?

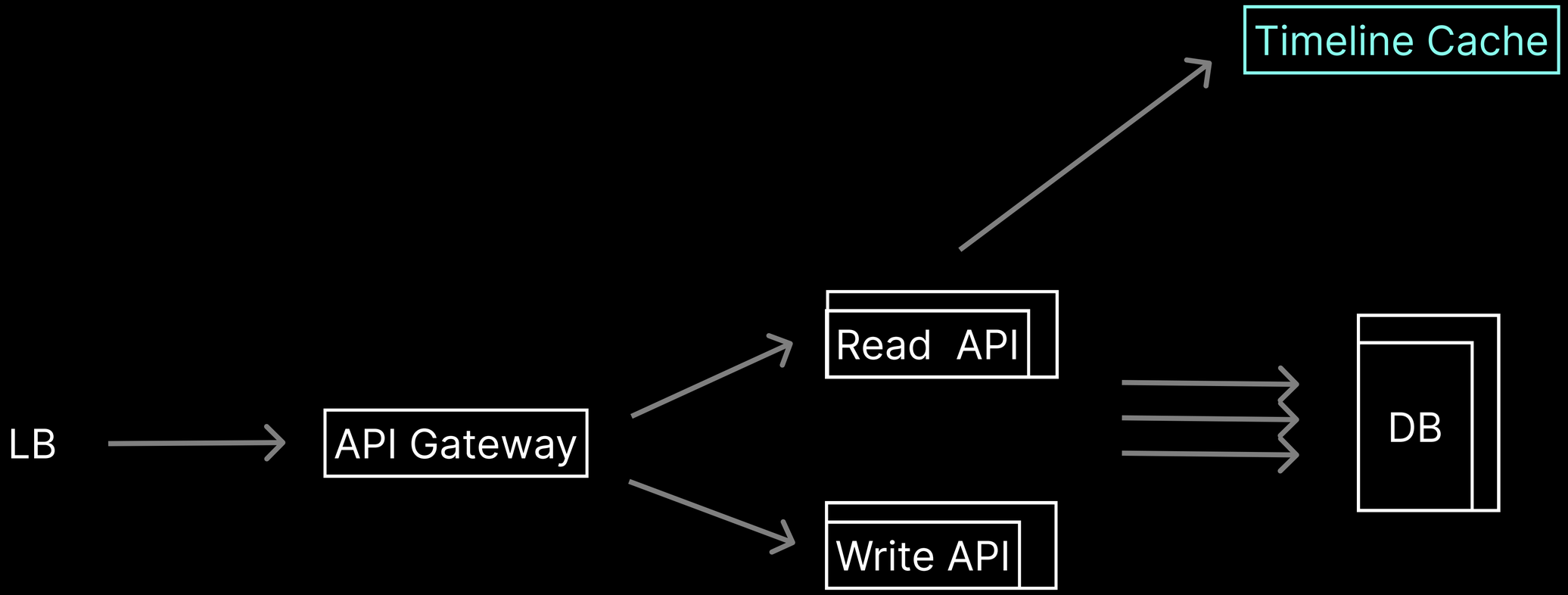
“Паническое бегство”




```java
public class MyCache < K, V > {
    private Lock lock = new ReentrantLock();
    final Map < K, V > cache = new HashMap <K, V>()
    public V get(K key) {
        V value = cache.get(key);
        if (value != null) {
            return value;
        }
        try {
            lock.lock();
            value = cache.get(key);
            if (value != null) {
                return value;
            }
            value = getFromDB(key);
            put(key, value);
        } finally {
            lock.unlock();
        }
        return value;
    }
```

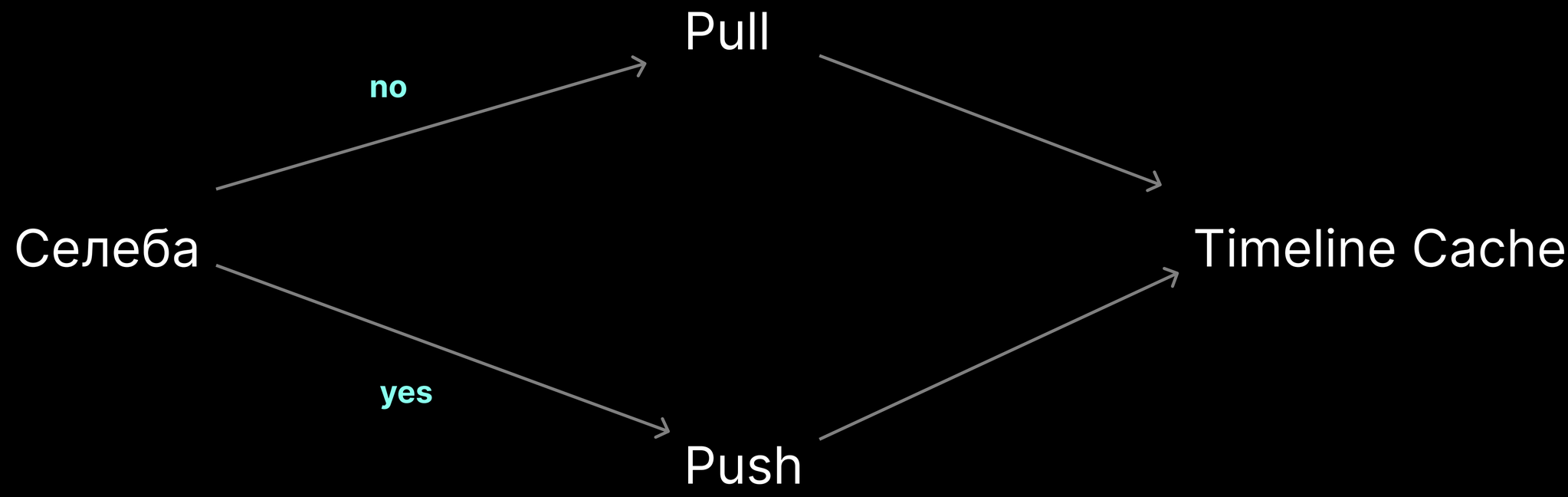
Locking

“Celebrity Problem”

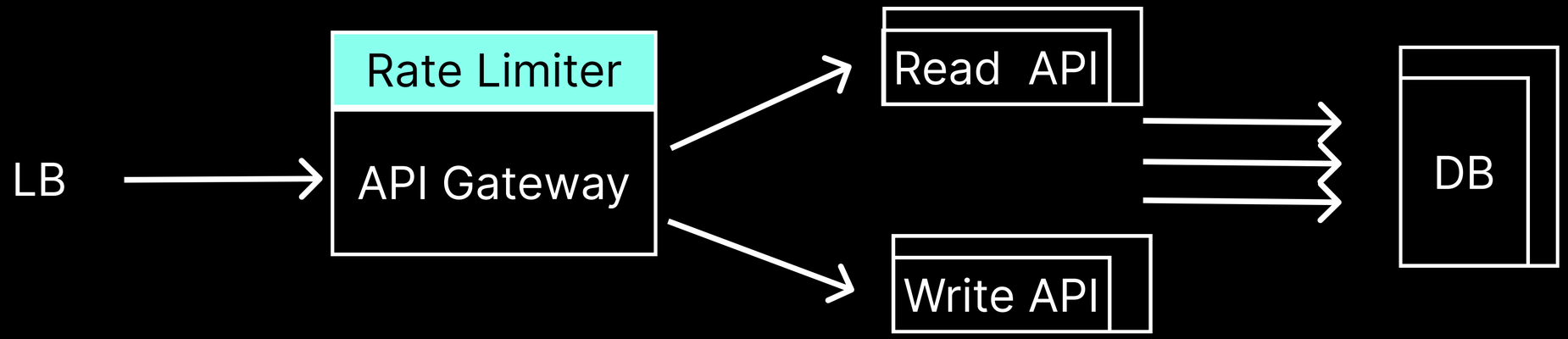


Как твит селебы попадет ко всем подписчикам в Таймлайн?
Должен ли твит ноунейма обновлять таймлайны?

Гибридная модель



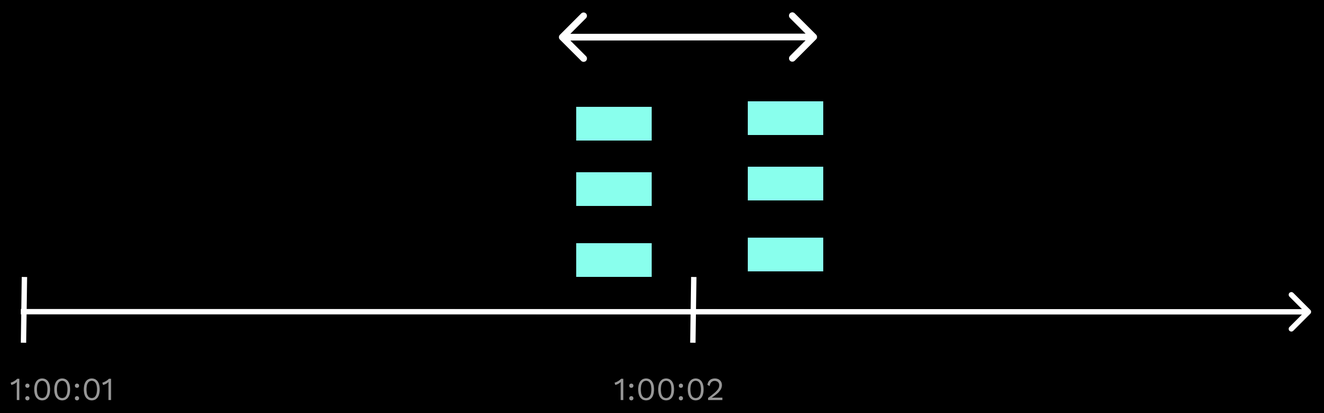
Rate Limiter

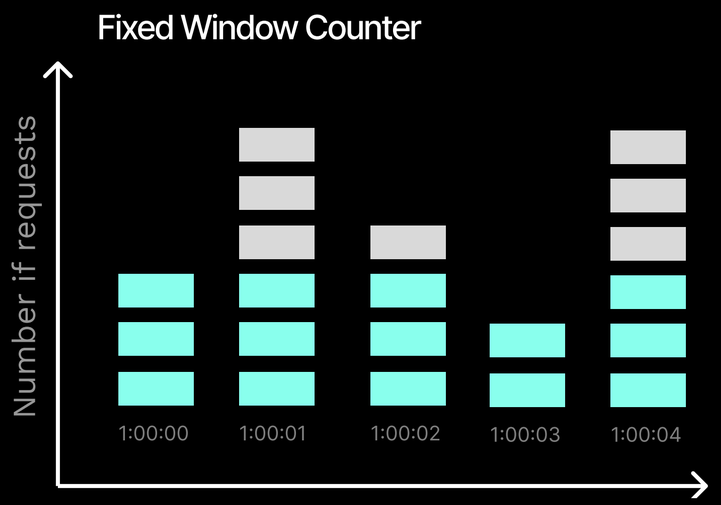

LB
Rate Limiter
API Gateway
Read API
Write API
DB

- Fixed Windows Counter
- Token Bucket
- Leaky Bucket
- Sliding Window Log/Counter

📌 Оконные алгоритмы подвержены всплескам
📌 Leaky Bucket использует поток
📌 Token Bucket можно написать без дополнительных потоков

FIXED WINDOW



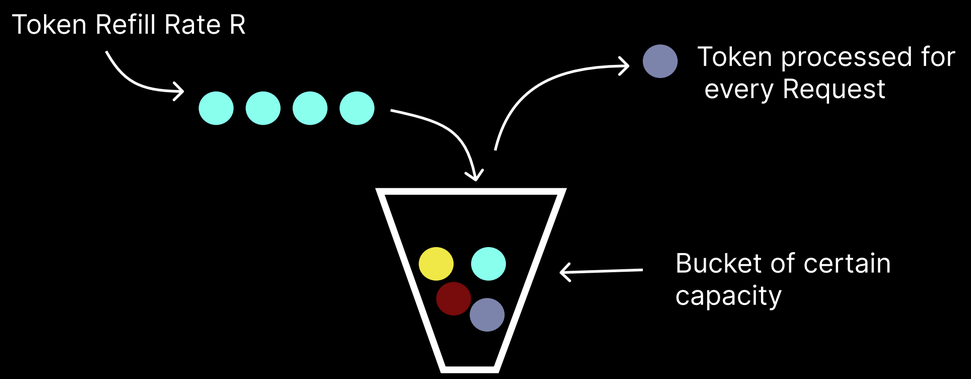


? А что если мы получили 6 запросов на стыке?

? Получается на стыках интервалов мы будем иметь больше чем 3 запроса в секунду?

TOKEN BUCKET

```java
public class TokenBucketRateLimiter {
    private final long capacity;
    private final AtomicLong tokens;
    private final Duration refillPeriod;
    private volatile Instant lastRefillTime;
   // constructor ommited

    public synchronized boolean isAllowed() {
        refillTokens();

        long currentTokens = tokens.get();
        if (currentTokens > 0) {
            tokens.decrementAndGet();
            return true; // Request is allowed
        }

        return false; // Request is not allowed
    }
```




```java
    private synchronized void refillTokens() {
        Instant now = Instant.now();
        long timeElapsed = Duration.between(lastRefillTime, now).toMillis();
        long tokensToAdd = timeElapsed / refillPeriod.toMillis();

        if (tokensToAdd > 0) {
            lastRefillTime = now;
            tokens.getAndUpdate(currentTokens -> Math.min(capacity, currentTokens +
tokensToAdd));
        }
    }
}
```

❓ Нужен ли refill thread?

❓ Нужен ли synchronized?

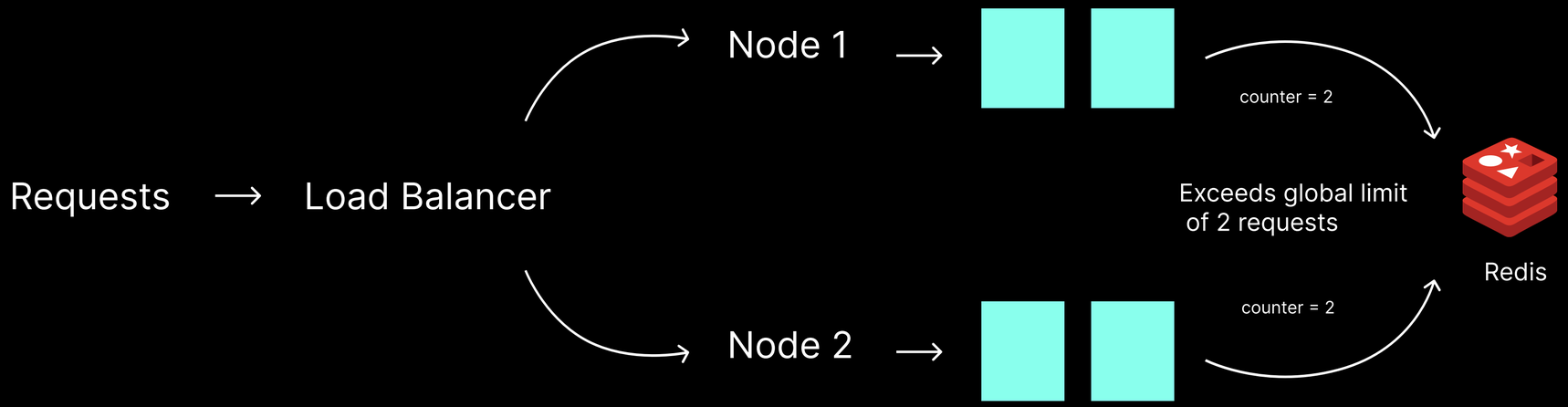
```java
public class LockFreeTokenBucket implements RateLimiter {

    private final BucketParams params;
    private final AtomicReference<BucketState> stateReference;

    public LockFreeTokenBucket(long permits, Duration period) {
        this.params = new BucketParams(permits, period);
        this.stateReference = new AtomicReference<>(new BucketState(params, nanoTime()));
    }

    public boolean tryAcquire(int permits) {
        while (true) {
            long nanoTime = nanoTime();
            BucketState previousState = stateReference.get();
            BucketState newState = new BucketState(previousState);
            newState.refill(params, nanoTime);
            if (newState.availableTokens < permits) {
                return false;
            }
            newState.availableTokens -= permits;
            if (stateReference.compareAndSet(previousState, newState)) {
                return true;
            }
        }
    }
}
```

```java
public final class BucketState {

    long availableTokens;
    long lastRefillNanoTime;

    public BucketState(BucketParams params, long nanoTime
        this.lastRefillNanoTime = nanoTime;
        this.availableTokens = params.capacity;
    }
```

Bucket4j https://github.com/bucket4j/bucket4j

Владимир Бухтояров
Пишем распределенный rate-limiter

DISTRIBUTED RATE LIMITER

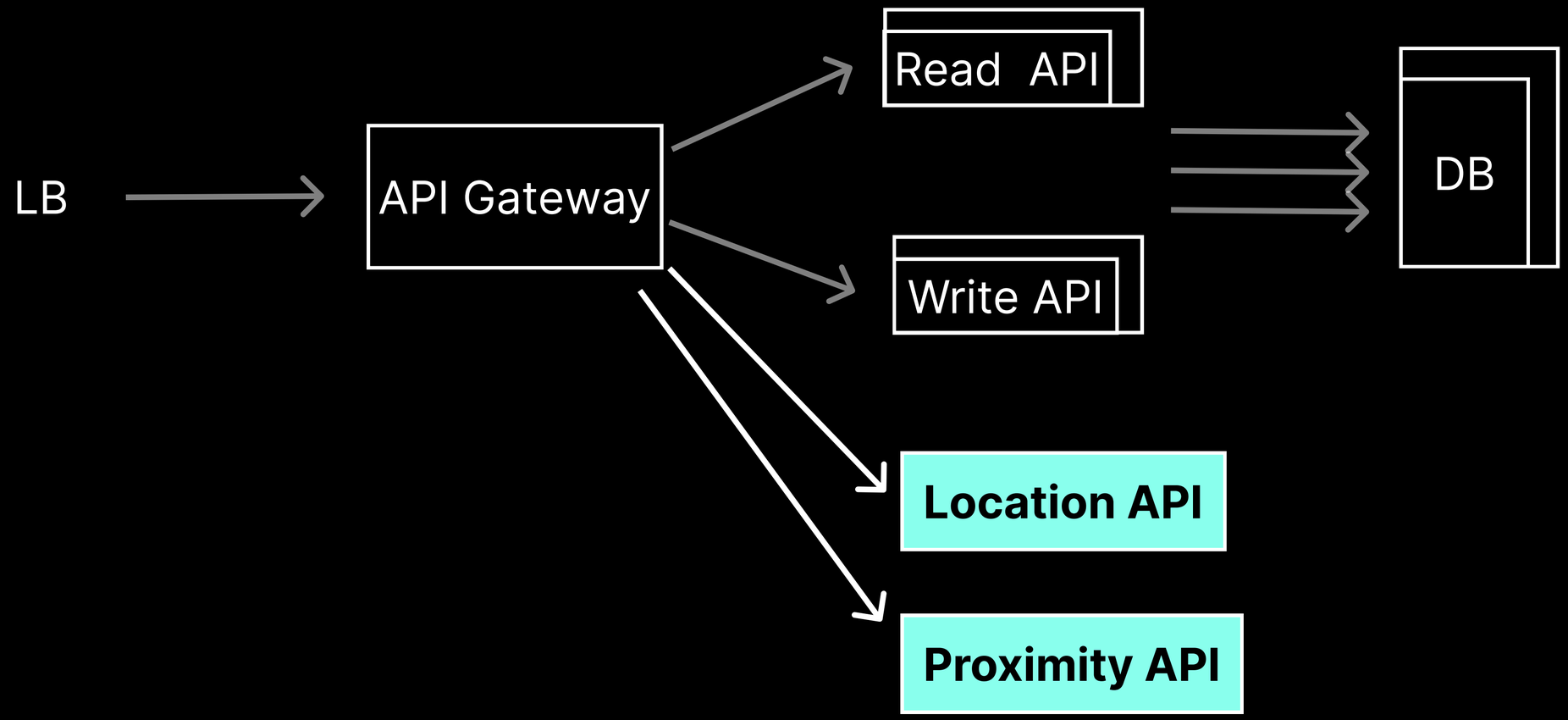


RDBMS + Select for Update → 3 сетевых запроса

MemCached, MongoDB
через CompareAndSwap → 2 сетевых заепроса

Redis, Aerospike, Taranatool + Хранимки
Hazelcast, Coherence, Ignite → 1 сетевой запрос

Карты



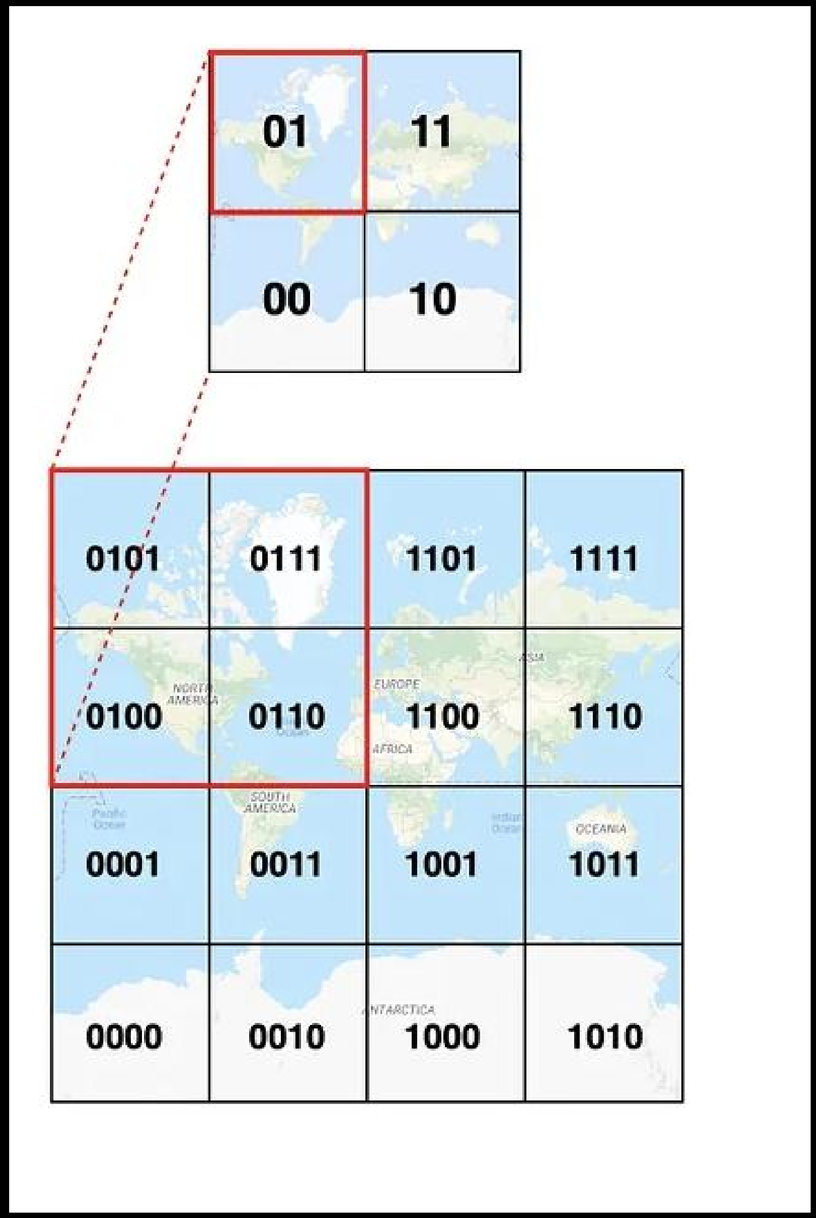
📌 CRUD для координат рестаранovrestaurant information
📌 Поиск ближайших ресторанов по координате и радиусу

GEOHASH

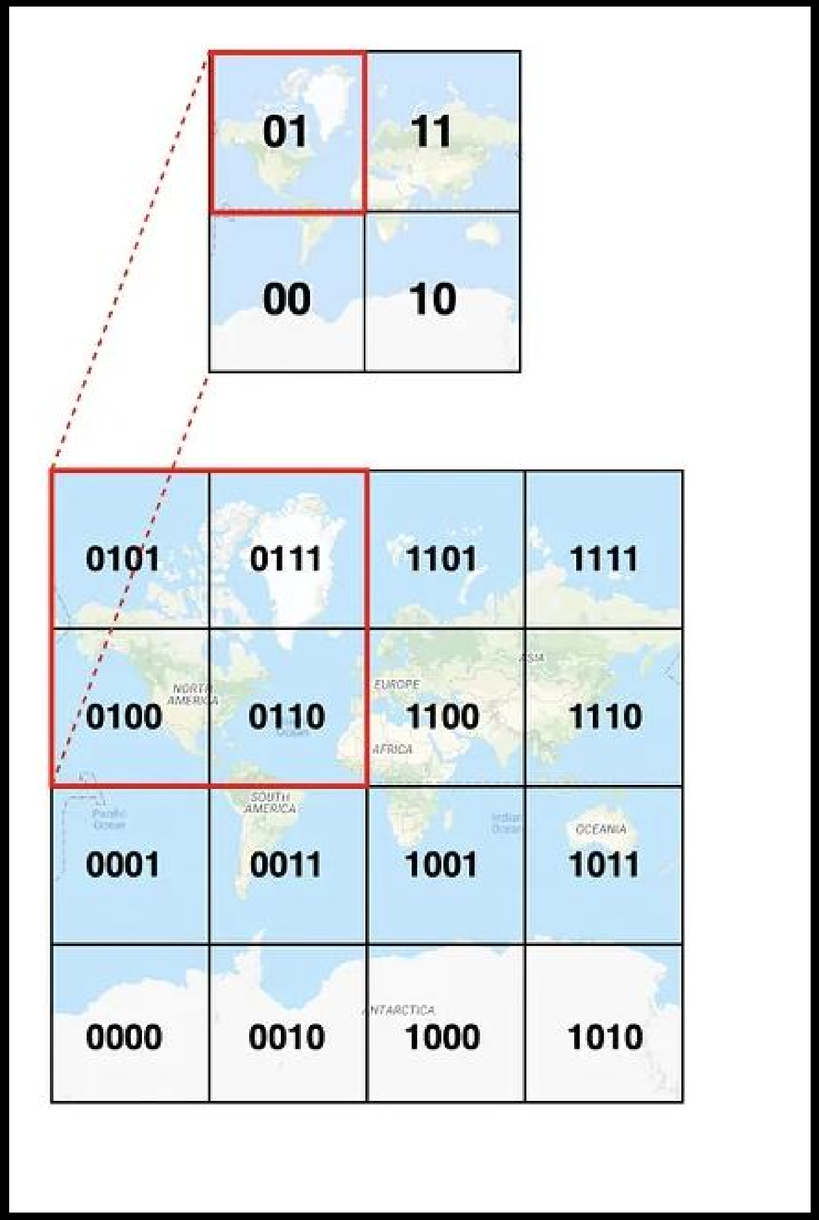
[-90, 0] левая половина 0
[0, 90] правая половина 1
[-180, 0] нижняя половина 0
[0, 180] верхняя половина 1

0	1

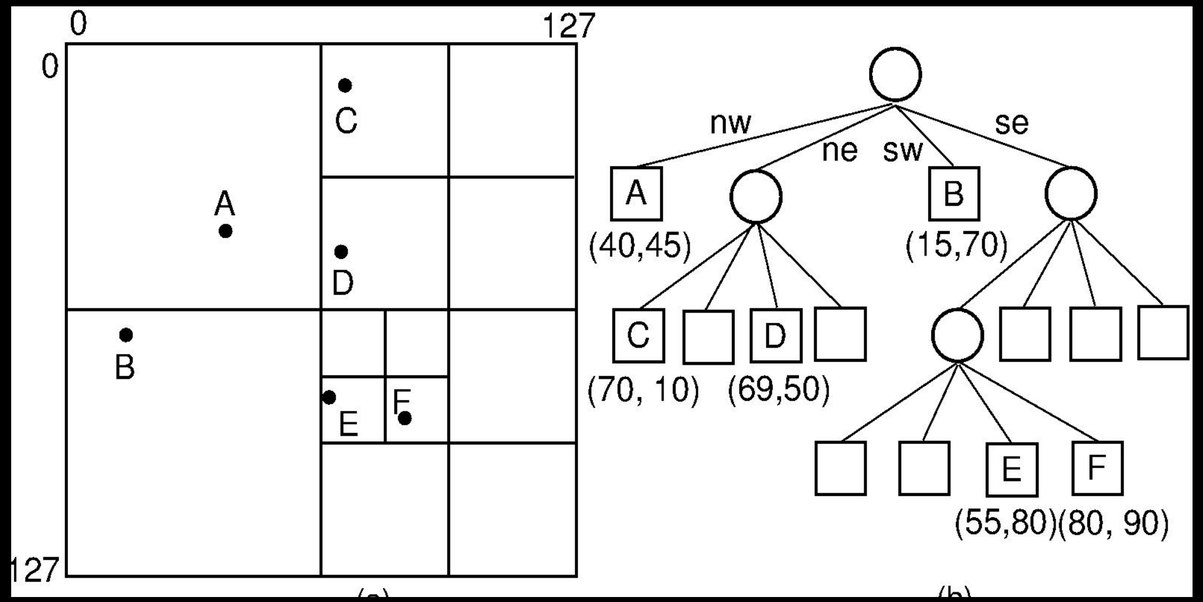
1
0



SELECT * FROM geohash_index
WHERE geohash LIKE `01%`



QUADTREE

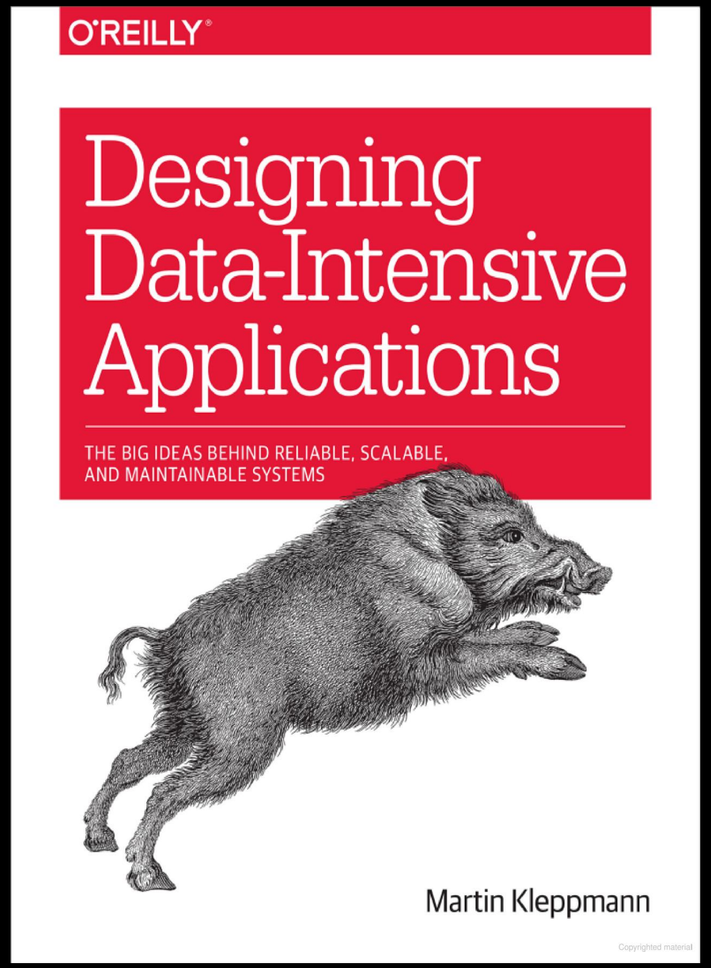



```
class Node {
    public boolean val;
    public boolean isLeaf;
    public Node topLeft;
    public Node topRight;
    public Node bottomLeft;
```

```
class Solution {
    public Node construct(int[][] grid) {
        int n = grid.length;
        return build(grid, 0, 0, n);
    }

    Node build(int[][] grid, int i, int j, int n) {

        if (n == 1) {
            return new Node(grid[i][j] == 1, true);
        }

        Node node = new Node(grid[i][j] == 1, false);
        node.topLeft = build(grid, i, j, n/2);
        node.topRight = build(grid, i, j + n/2, n/2);
        node.bottomLeft = build(grid, i + n/2, j, n/2);
        node.bottomRight = build(grid, i + n/2, j + n/2, n/2);

        if (node.topLeft.isLeaf && node.topRight.isLeaf
            && node.bottomLeft.isLeaf && node.bottomRight.isLeaf) {

            if (node.topLeft.val == node.topRight.val && node.topLeft.val ==
                && node.topLeft.val == node.bottomRight.val) {
                node.val = node.topLeft.val;
                node.isLeaf = true;
                node.topLeft = null;
                node.topRight = null;
                node.bottomLeft = null;
                node.bottomRight = null;
            }
        }

        return node;
    }
}
```

SYSTEM DESIGN 101

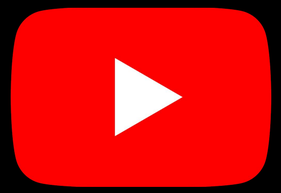
- Это не все систем дизайны
- Блок про модель данных, API
- System Design для вашего проекта
- System Design для интервью в компанию
- Вы ведете интервью
- Структура
- Концепции, а не технологии
- Мок-интевью



Спасибо!
А вот теперь кабанчик

за лучший вопрос

@jawaswag

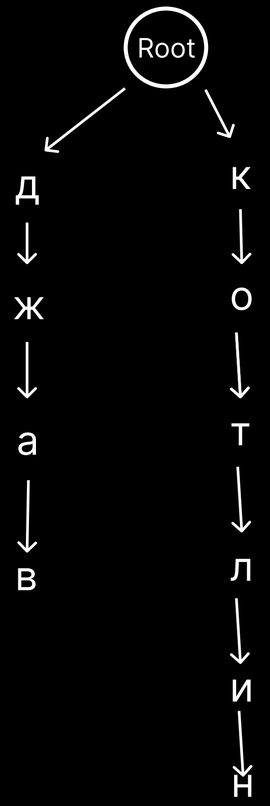


@faangtalk

Бонус

AUTOCOMPLETION SYSTEM

- Trie (Prefix Tree)
- Distributed Trie

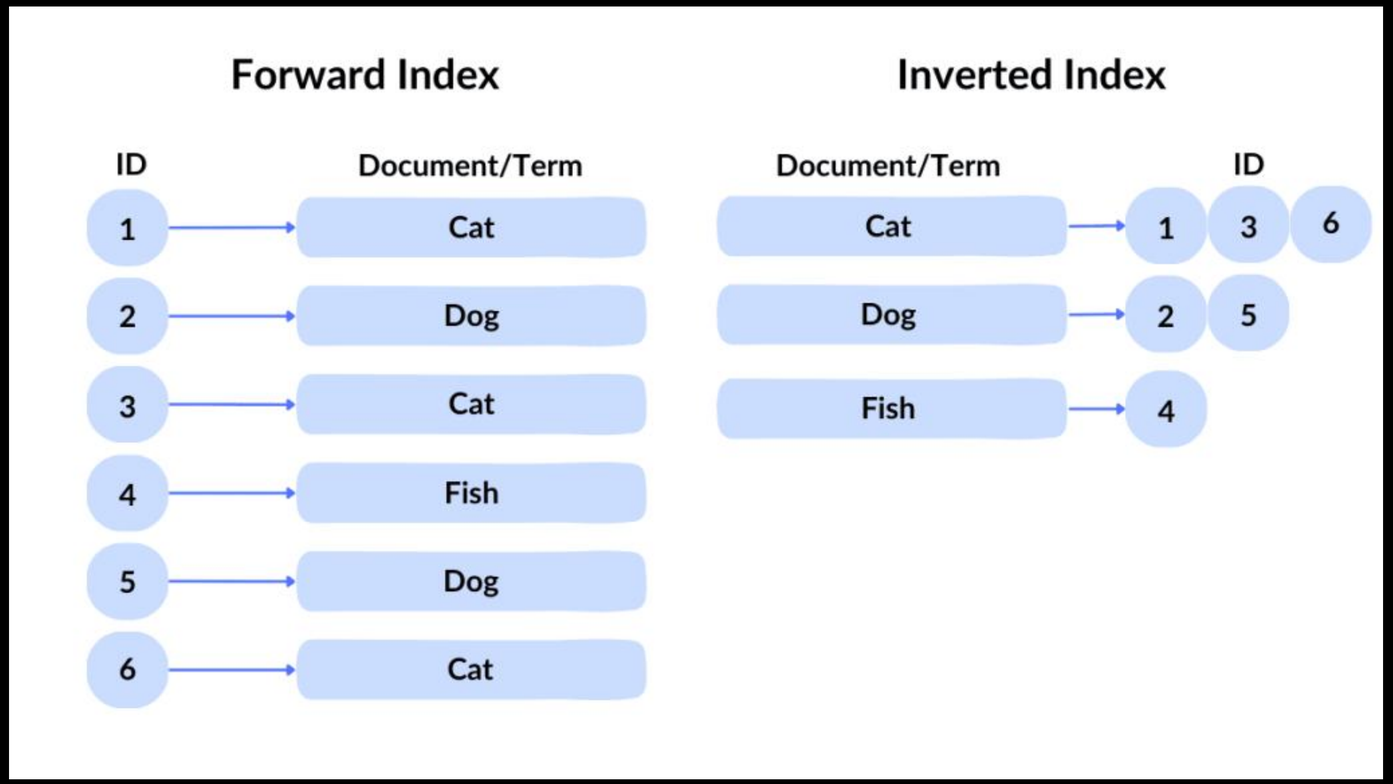



```java
class Trie {
    static final int R = 256;
    Trie[] next = new Trie[256];
    boolean isLeaf;

    /** Initialize your data structure here. */
    public Trie() {

    }

    /** Inserts a word into the trie. */
    public void insert(String word) {
        Trie node = this;
        for (char c: word.toCharArray()) {
            Trie curr = node.next[c];
            if (curr == null) {
                curr = new Trie();
            }
            node.next[c] = curr;
            node = curr;
        }
        node.isLeaf = true;
    }

    /** Returns if the word is in the trie. */
    public boolean search(String word) {
        Trie node = this;
        for (char c: word.toCharArray()) {
            Trie curr = node.next[c];
            if (curr == null) {
                return false;
            }
            node.next[c] = curr;
            node = curr;
        }
        return node.isLeaf;
    }
}
```

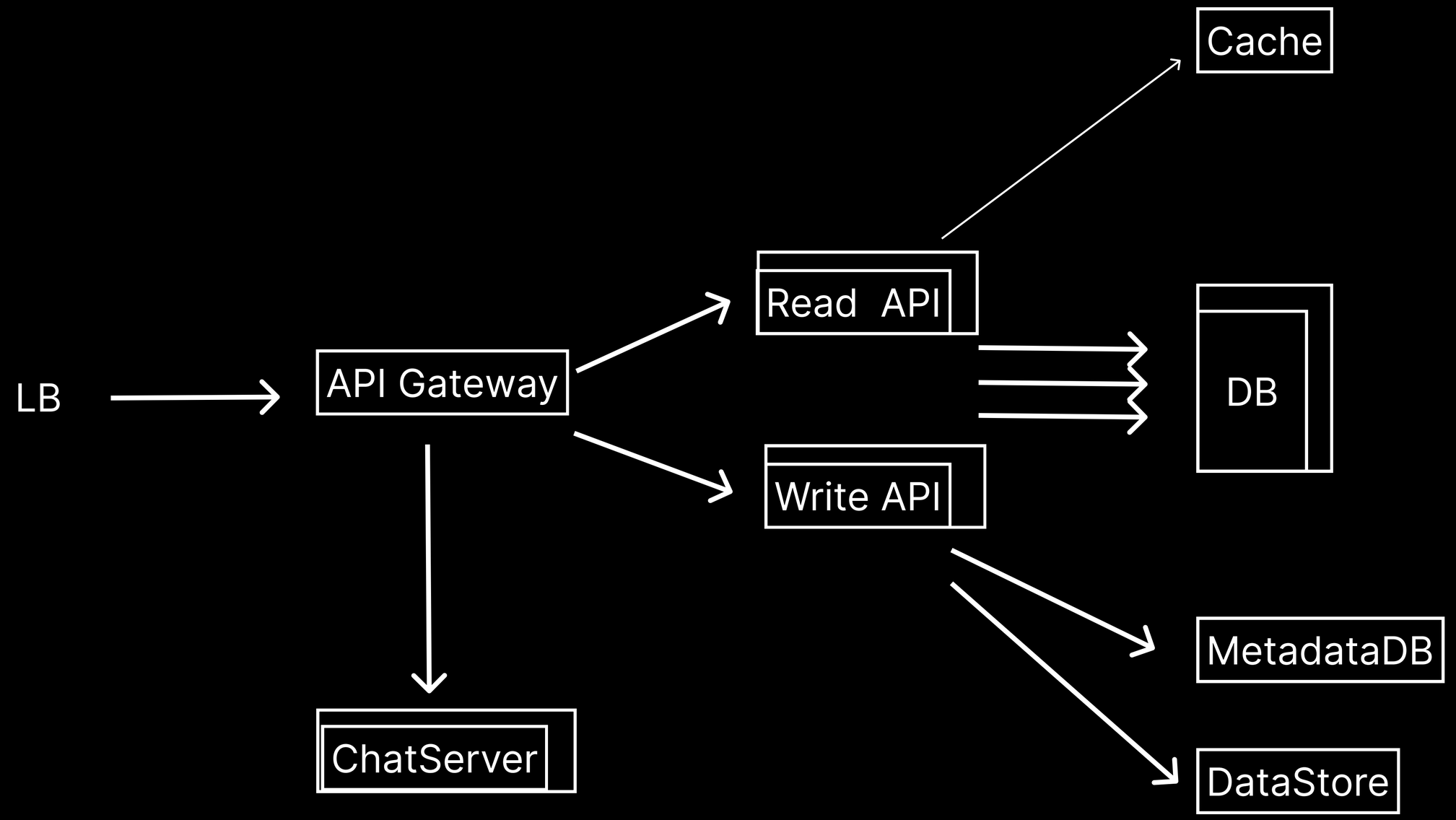
- A-F
- G-M
- N-R
- S
- T-Z

SEARCH

- Inverted Index
- Page Rank
- MapReduce



Хранение файлов


Cache
Read API
LB
API Gateway
DB
Write API
MetadataDB
ChatServer
DataStore

S3 - SIMPLE STORAGE SERVICE

- S3 like storage
-
- Google Drive
- Dropbox
- Google Photo

- Youtube
- Netflix
- Spotify

- Name Nodes / Data Nodes
- Journaling/Checkpointing

- Roles
- URL Shortener
- Notifications

- Protocol Encoding
- Streaming
- Smart Chunk splitting

GFS

- Master
- Chunk Server

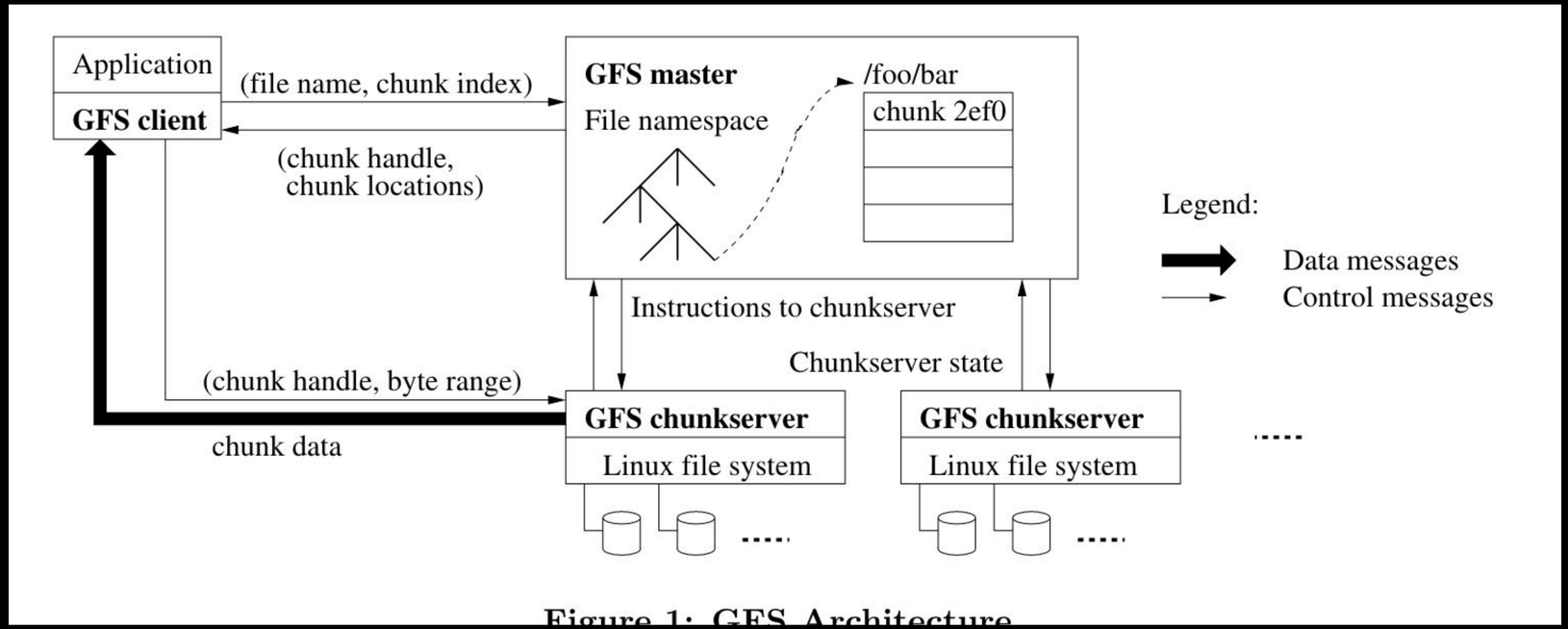
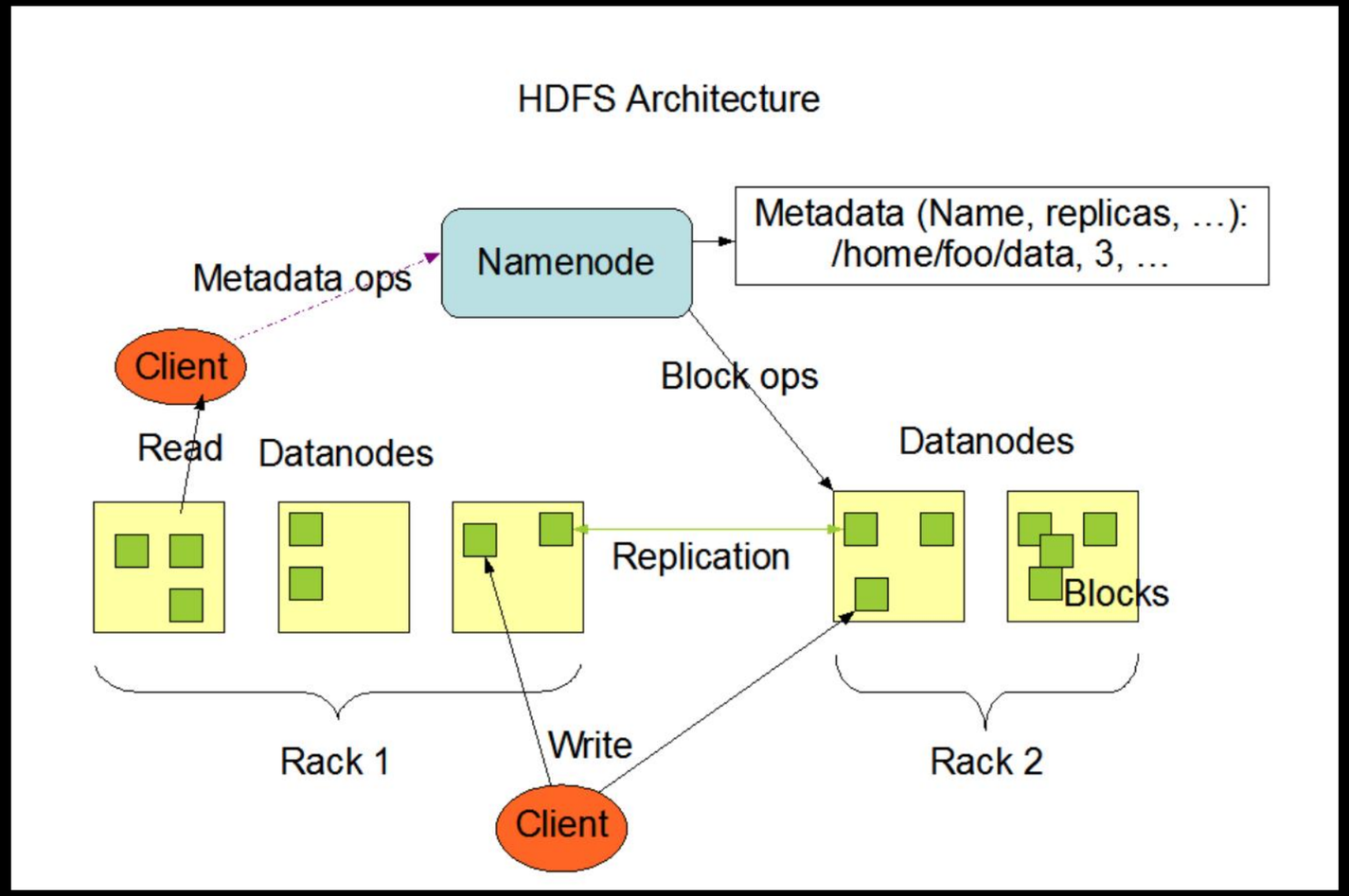


Figure 1: GFS Architecture

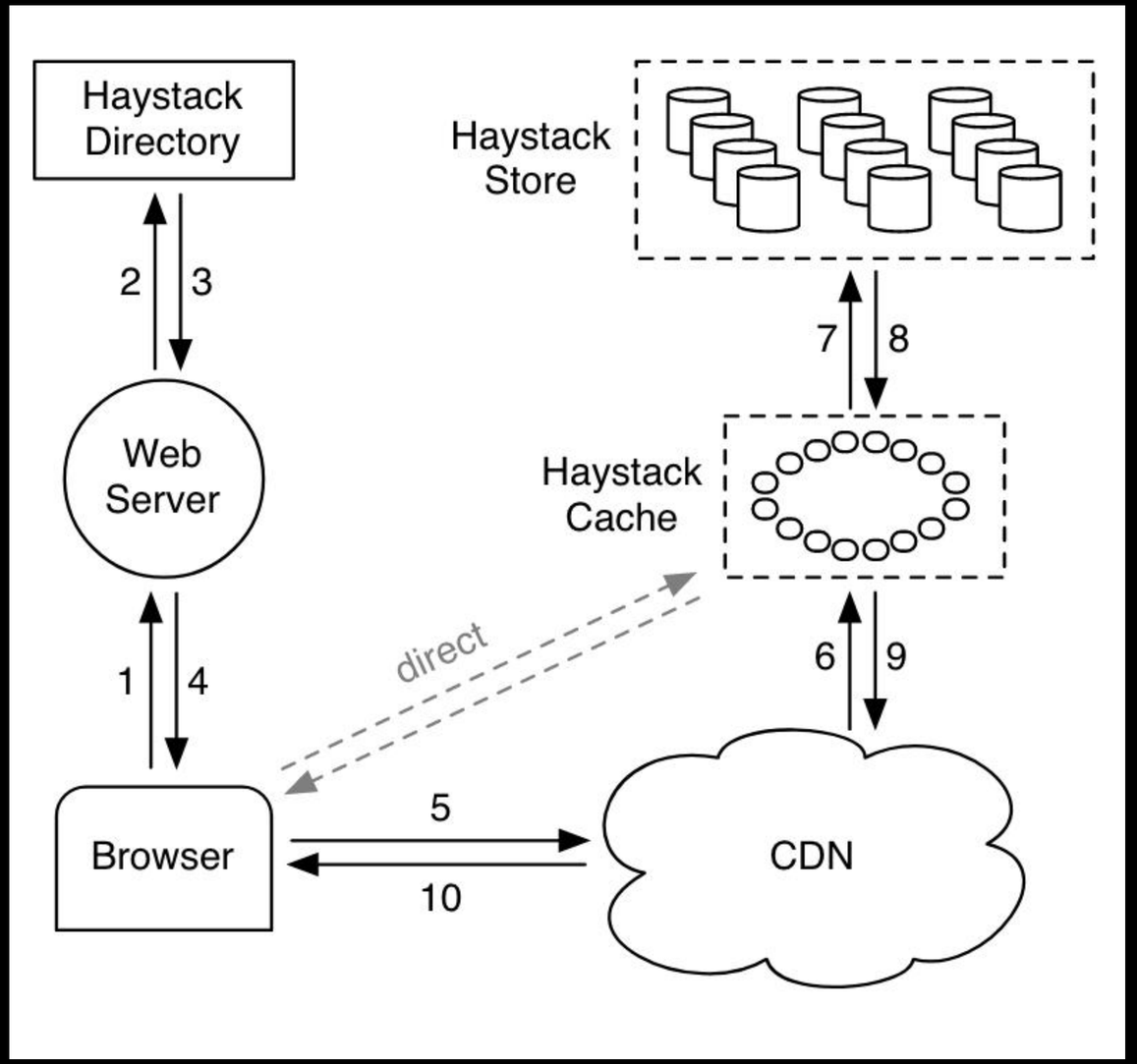
HDFS

- Name Nodes
- Data Nodes/Chunks
-



HAYSTACK

- Finding a needle in Haystack



Length and OVERFLOW in

- 38 bits = 8.7 years
- 39 bits = 17.4 years
- 40 bits = 34.8 years
- 41 bits = 69.7 years

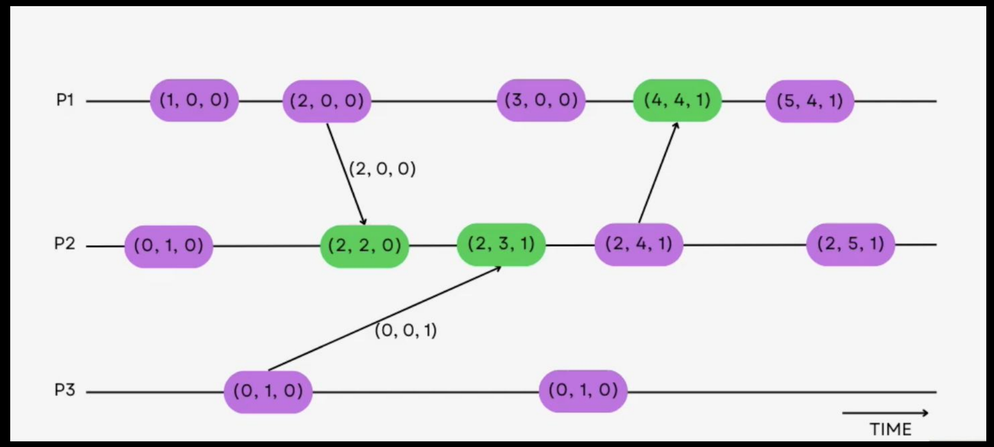
```
48      public Snowflake() {
49          this.nodeId = createNodeId();
50          this.customEpoch = DEFAULT_CUSTOM_EPOCH;
51      }
52
53      public synchronized long nextId() {
54          long currentTimestamp = timestamp();
55
56          if(currentTimestamp < lastTimestamp) {
57              throw new IllegalStateException("Invalid System Clock!");
58          }
59
60          if (currentTimestamp == lastTimestamp) {
61              sequence = (sequence + 1) & maxSequence;
62              if(sequence == 0) {
63                  // Sequence Exhausted, wait till next millisecond.
64                  currentTimestamp = waitNextMillis(currentTimestamp);
65              }
66          } else {
67              // reset sequence to start with zero for the next millisecond
68              sequence = 0;
69          }
70
71          lastTimestamp = currentTimestamp;
72
73          long id = currentTimestamp << (NODE_ID_BITS + SEQUENCE_BITS)
74                  | (nodeId << SEQUENCE_BITS)
75                  | sequence;
76
77          return id;
```

https://github.com/callicoder/java-snowflake/tree/master

MONGODB OBJECTID

```
public final class ObjectId implements /*...*/
{    // ...
     // The timestamp
     private final int timestamp;
     // The counter.
     private final int counter;
     // the first four bits of randomness.
     private final int randomValue1;
     // The last two bits of randomness.
     private final short randomValue2;
     //...
}
```

VECTOR CLOCKS



- Time, Clocks, and the Ordering of Events in a Distributed System
- Google way: TrueTime API for Spanner

FOREIGN KEYS

TAO

- One DataMode to rule them all!
- TAO paper - associations and objects

- Object: (id) → (otype, (key value)*)
- Assoc.: (id1, atype, id2) → (time, (key value)*)

TAO API

- Object API: CRUD
- Assoc API
  - assoc_add(id1, atype, id2, time, (k→v)*)
  - assoc_delete(id1, atype, id2)
  - assoc_change type(id1, atype, id2, newtype)

- Association List:
  - assoc_get(id1, atype, id2set, high?, low?)
  - assoc_count(id1, atype)
  - assoc_range(id1, atype, pos, limit)
  - assoc_time_range(id1, atype, high, low, limit)