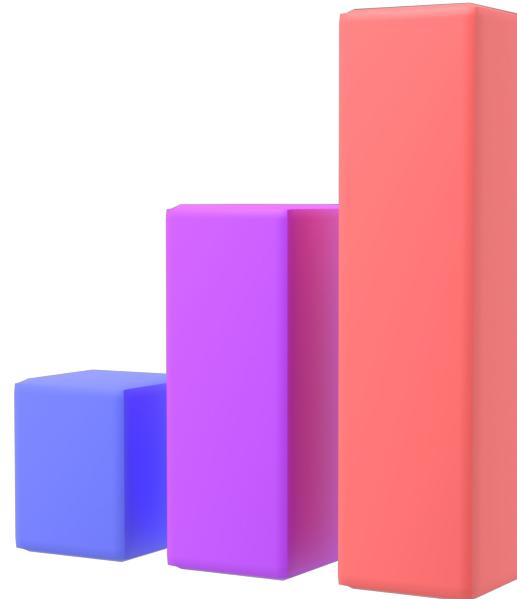


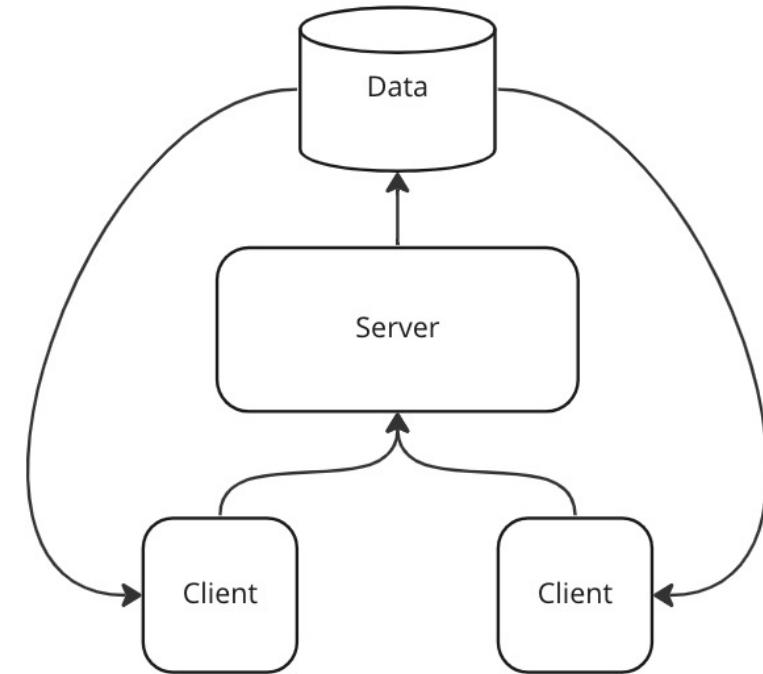
System Design

Основные термины и компоненты



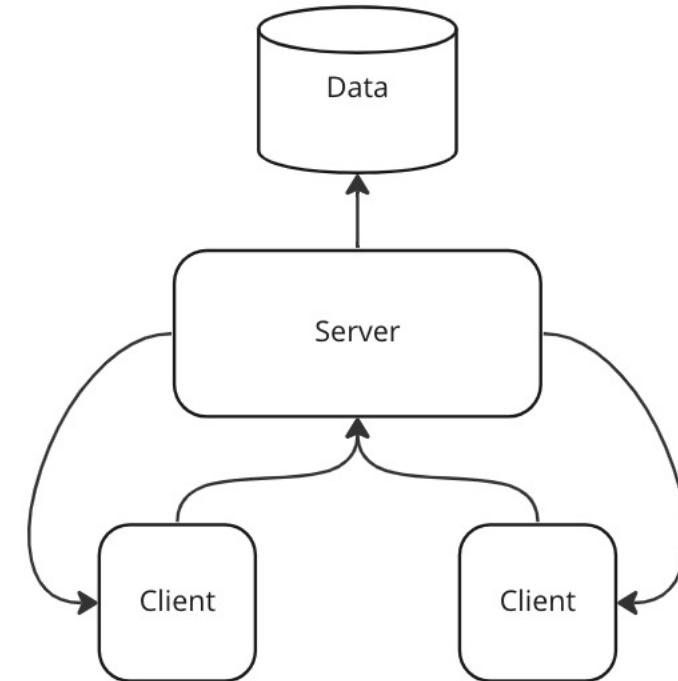
Файл-сервер

Файл-сервер только извлекает данные из файла или базы данных и передает их клиенту для дальнейшей обработки



Клиент-сервер

Клиент-сервер извлекает данные из файла или базы данных, обрабатывает и затем передает результат клиенту



Основные критерии систем

Надежность

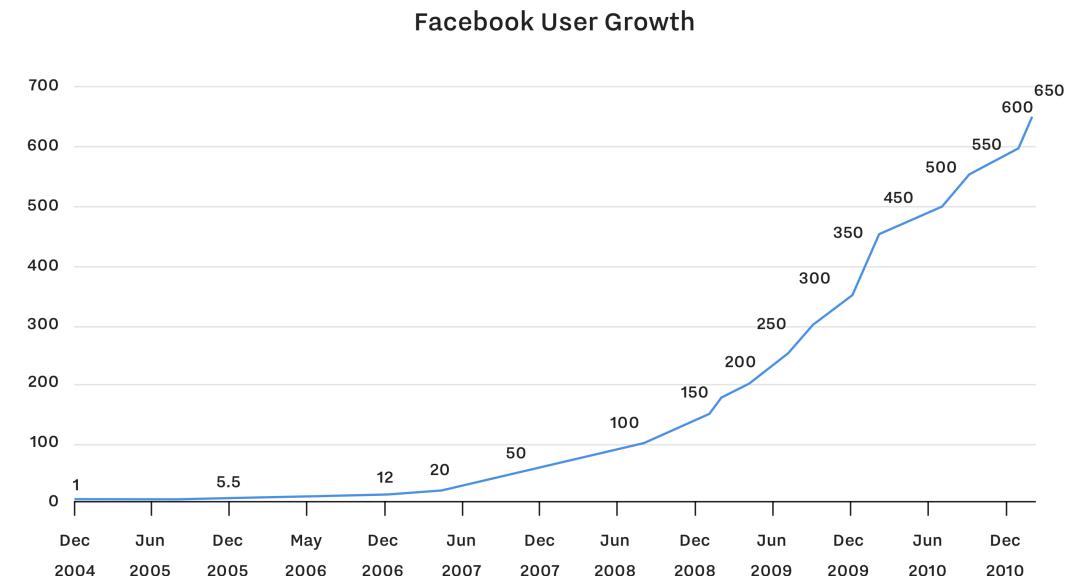
Система должна продолжать работать корректно даже при неблагоприятных обстоятельствах



Availability Level		Average Yearly Downtime
Conventional Server	99%	87 hours, 40 minutes
Public Cloud Service	99.5%	43 hours, 50 minutes
	99.9%	8 hours, 46 minutes
High-Availability Cluster	99.95%	4 hours, 23 minutes
Virtual Fault Tolerance	99.995%	26 minutes, 18 seconds
Continuous Availability	99.999%	5 minutes, 16 seconds
The Stratus Zone	99.9999%	31.6 seconds

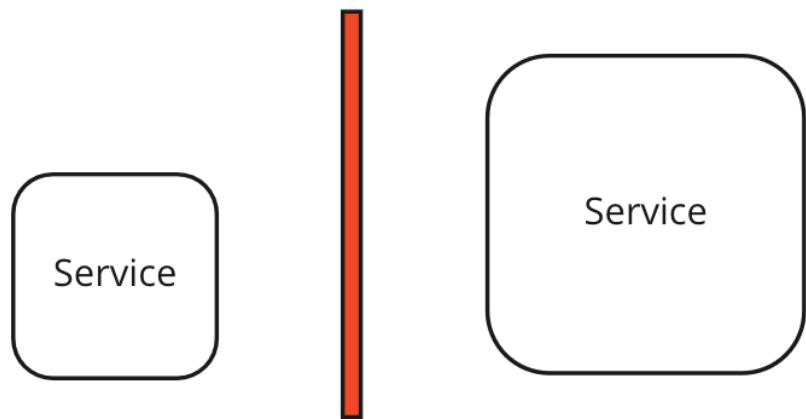
Масштабируемость

Должны быть предусмотрены разумные
способы решения возникающих при росте
системы проблем



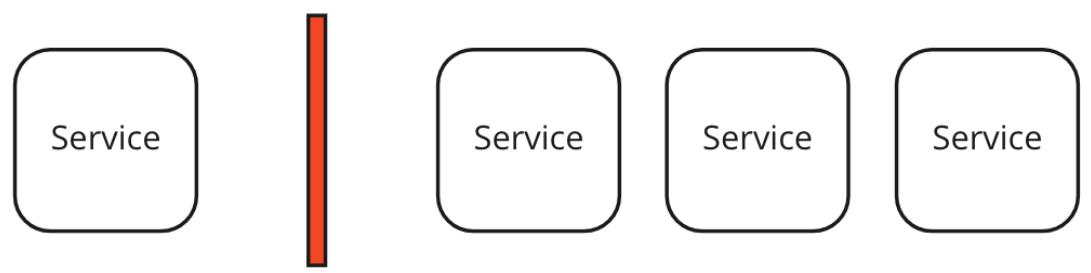
Вертикальное

Стоимость увеличения ресурсов растет не линейно, относительно их увеличения мощности (возможен downtime)



Горизонтальное

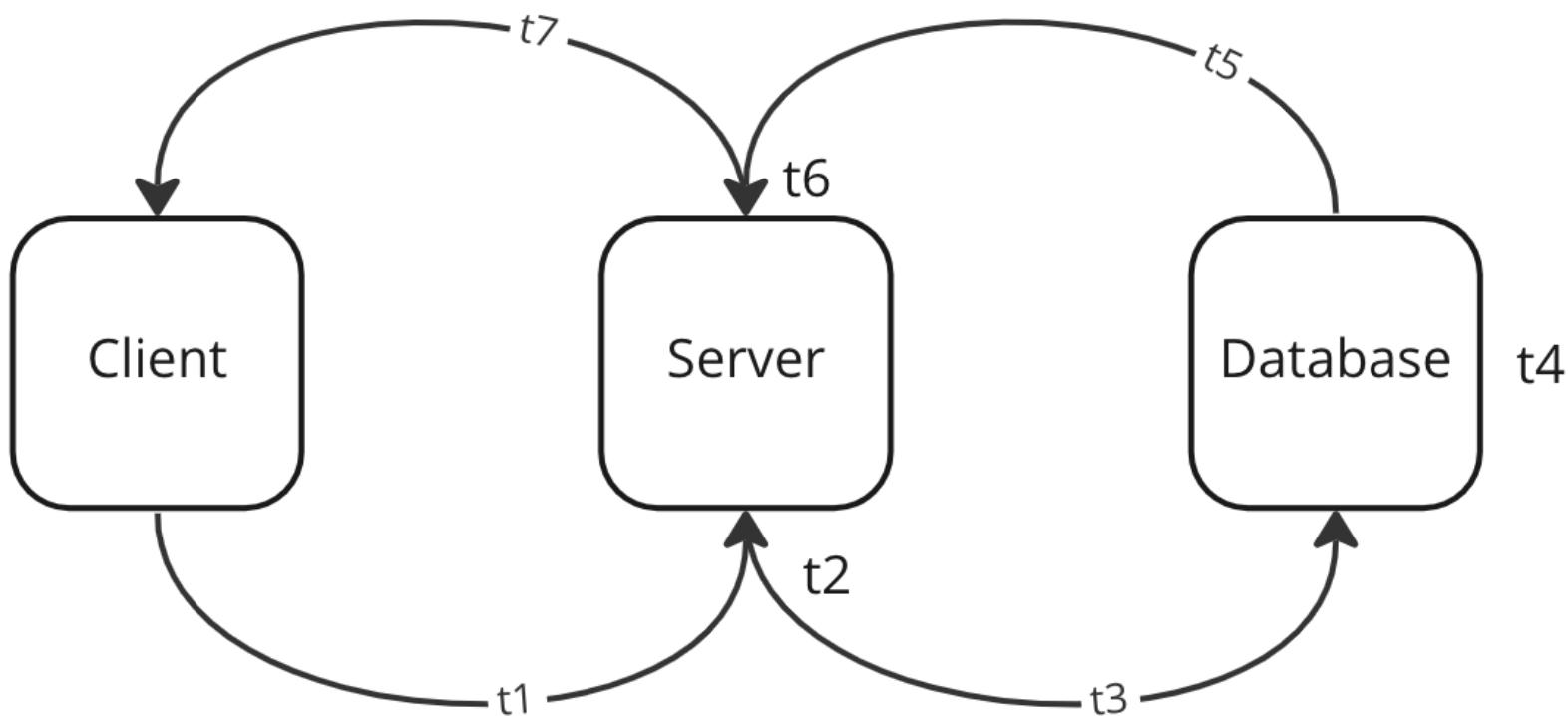
Чаще всего масштабирование
происходит без каких-либо проблем



Stateless and Statefull

Производительность

Latency и Response Time



Latency

Read 1M from RAM = 250 000 нс

Read 1M from SSD = 1 мс

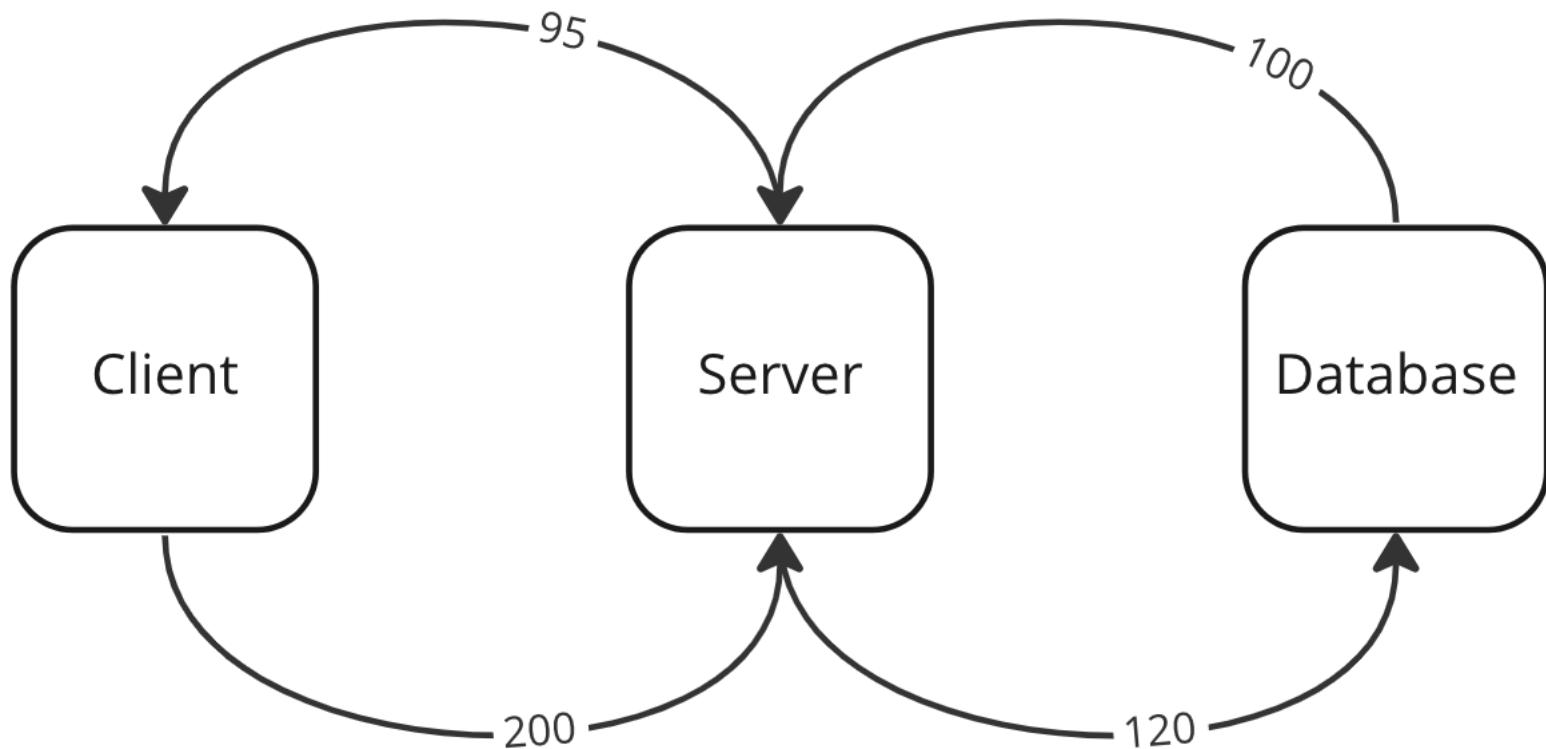
Read 1M from HDD = 20 мс



Low-latency приложения

Такие приложение надо с самого начала писать с оптимизациями, с прямой ориентацией на low-latency. Подход, когда пишут сначала просто работающее приложение, а потом оптимизируют его, здесь не работает

Throughput



Throughput

HDD = 200-300 МБ/с

SSD = 600-700 МБ/с

DDR3 = 17000 МБ/с



Удобство сопровождения

Необходимо обеспечить возможность
эффективной работы с системой
множеству различных людей



Удобство сопровождения

- Observability
- Улучшение процессов
- Дополнительный инструментарий

SLA / SLO / SLI

- **SLA (Service Level Agreement)** – соглашение, которое вы заключаете со своими клиентами
- **SLO (Service Level Objectives)** - цели, которые команда должна решить, чтобы выполнить соглашение
- **SLI (Service Lever Indicators)** – показатели системы

Google Drive

Covered Service	Monthly Uptime Percentage
Standard storage class in a multi-region or dual-region location of Cloud Storage	>= 99.95%
Standard storage class in a regional location of Cloud Storage; Nearline, Coldline, or Archive storage class in a multi-region or dual-region location of Cloud Storage	>= 99.9%
Nearline, Coldline, or Archive storage class in a regional location of Cloud Storage; Durable Reduced Availability storage class in any location of Cloud Storage	>= 99.0%

Google Drive

Monthly Uptime Percentage	Percentage of monthly bill for the Standard storage class in a multi-region or dual-region location of Cloud Storage that does not meet SLO that will be credited to Customer's future monthly bills
99.0% – < 99.95%	10%
95.0% – < 99.0%	25%
< 95.0%	50%

Что из этого HighLoad?

20 RPS или 20 000 RPS

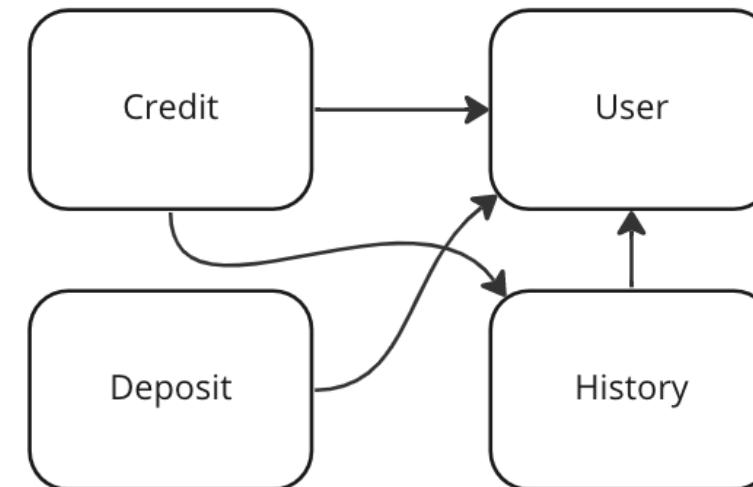
Data-intensive

- Нужно сохранять большие данные
- Нужно запоминать результаты ресурсоемких операций
- Нужно предоставлять пользователям возможность искать или фильтровать данные

Compute-intensive

- Нужно отправлять сообщения другим процессам
- Нужно «перемалывать» большие объемы данных

Монолит VS Микросервисы



Микросервисы

- Микро необязательно про размер, но про зону ответственности
- Самодостаточны, идеальны для горизонтального масштабирования
- Разные технологии для разных задач
- Распределенная кодовая база

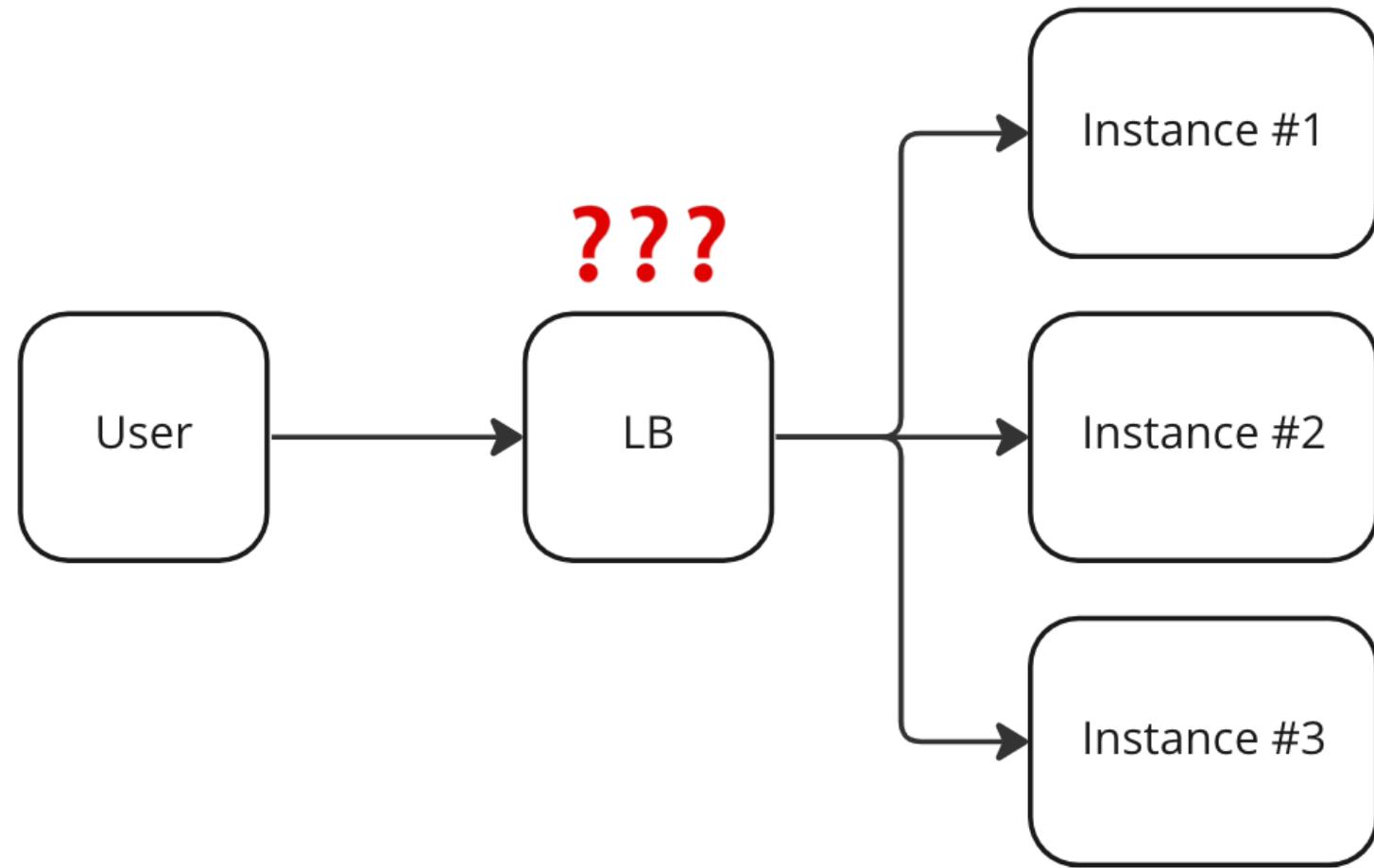
Микросервисы

- + Независимые релизы и разработка
- + Независимая масштабируемость
- + Независимая деградация
- + Возможность пробовать новые технологии
- Зоопарк технологий
- Сетевой вызов отвалится вероятнее, чем внутренний
- Распределенность и транзакционность
- Удаленные вызовы дороже локального исполнения
- Понимание всего контекста запроса
- Сложность тестирования всей системы

The Bezos Mandate

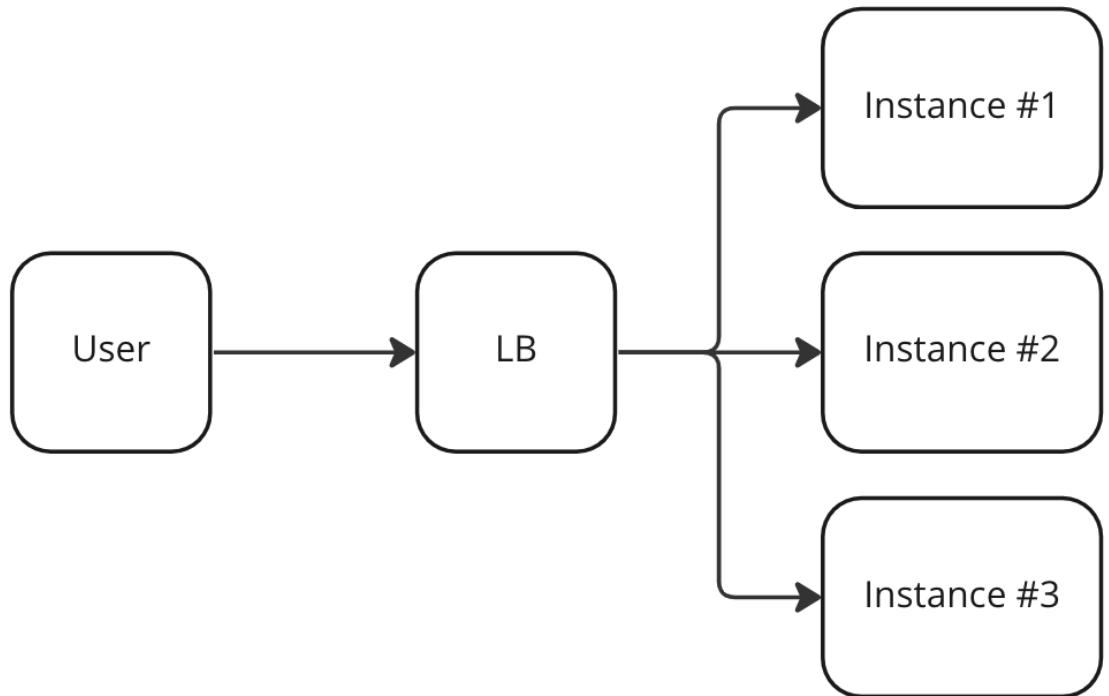
1. All teams will henceforth expose their data and functionality through service interfaces.
2. Teams must communicate with each other through these interfaces.
3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols – doesn't matter.
5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
6. Anyone who doesn't do this will be fired.
7. Thank you; have a nice day!

Балансировка нагрузки



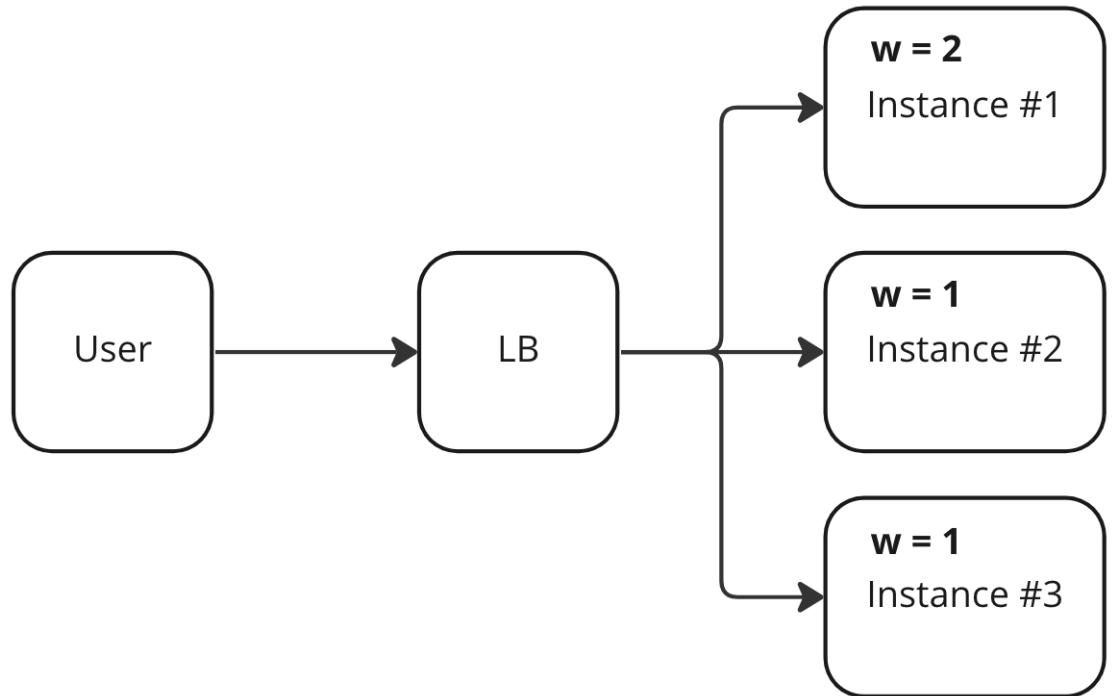
Round Robin

- 1: User -> Instance #1
- 2: User -> Instance #2
- 3: User -> Instance #3
- 4: User -> Instance #1
- 5: User -> Instance #3
- 6: User -> Instance #2



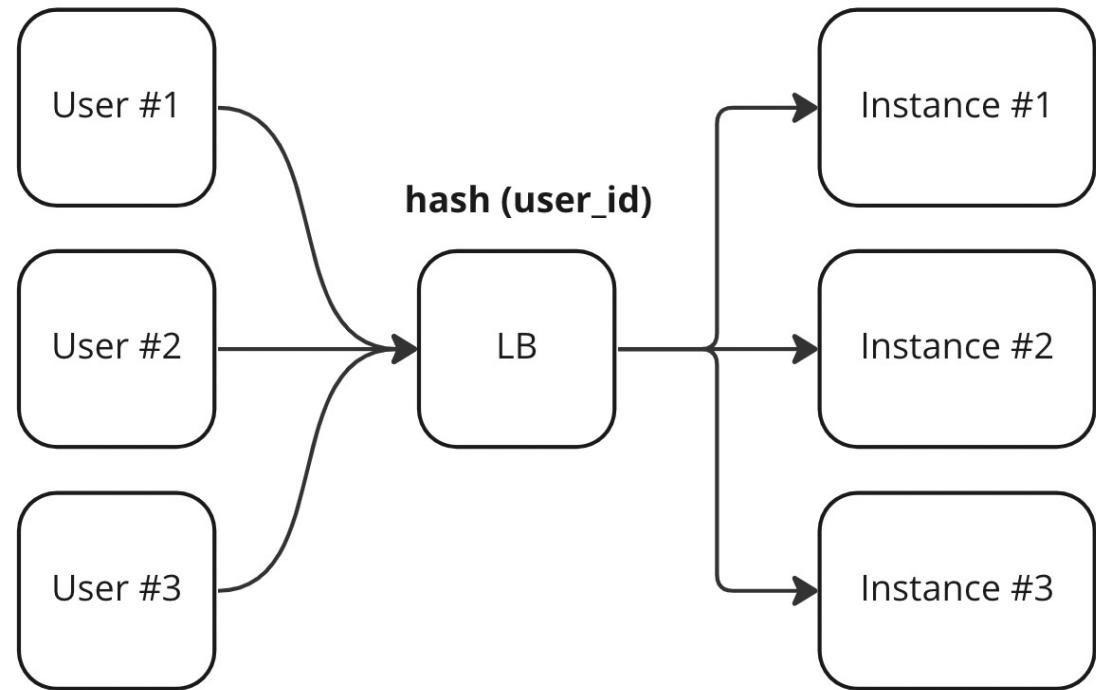
Weighted RR

- 1: User -> Instance #1
- 2: User -> Instance #1
- 3: User -> Instance #2
- 4: User -> Instance #3
- 5: User -> Instance #1
- 6: User -> Instance #3

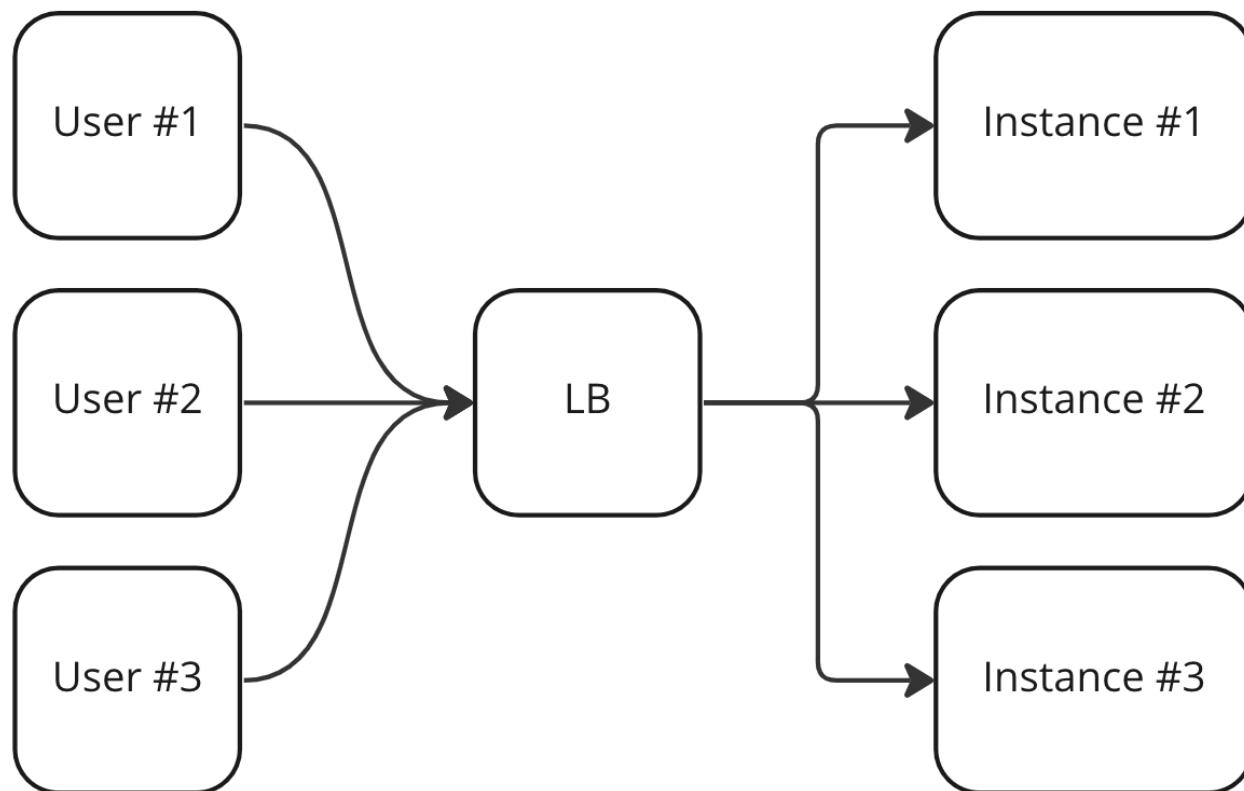


Sticky Sessions

- 1: User #1 -> Instance #2
- 2: User #2 -> Instance #1
- 3: User #3 -> Instance #3
- 4: User #2 -> Instance #1
- 5: User #3 -> Instance #3
- 6: User #1 -> Instance #2



Least Connections





Nginx

```
1 upstream backend {
2     server backend1.example.com:8080 weight=5;
3     server backend2.example.com:8080;
4 }
5
6 server {
7     location / {
8         proxy_pass http://backend;
9     }
10 }
```

Проксирование

Проксирование

Взлом / защита

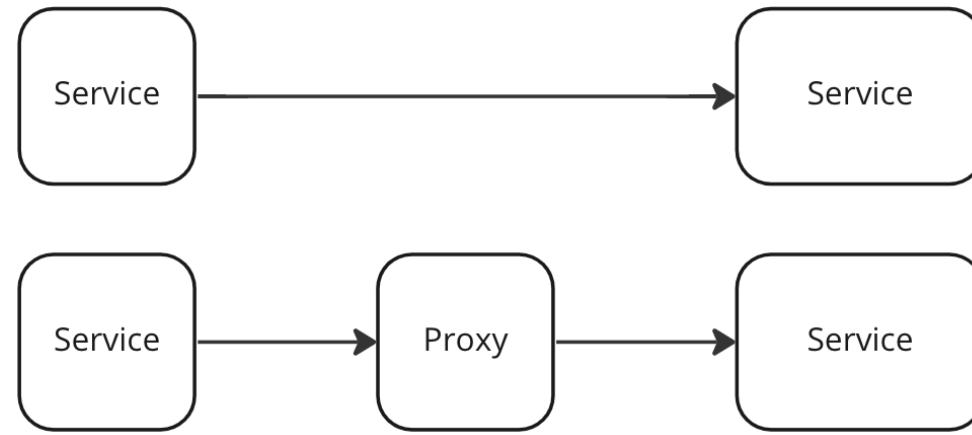
Кэширование данных

Ограничения трафика

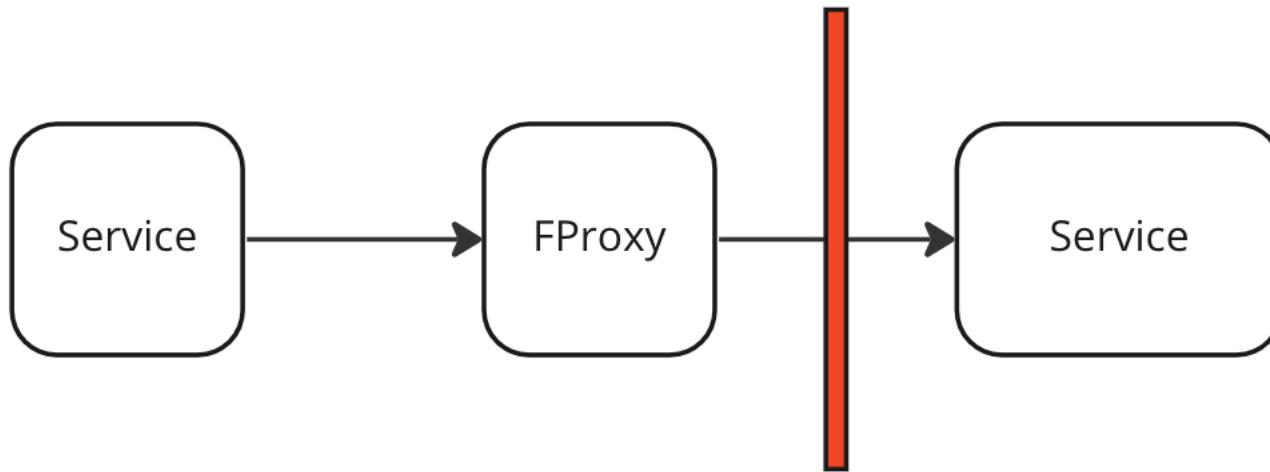
Обход ограничений доступа

Анонимность пользователей

Сжатие и модификация данных



Forward Proxy



14 марта 2022, 09:56 / Медиа

Роскомнадзор заблокировал Instagram в России

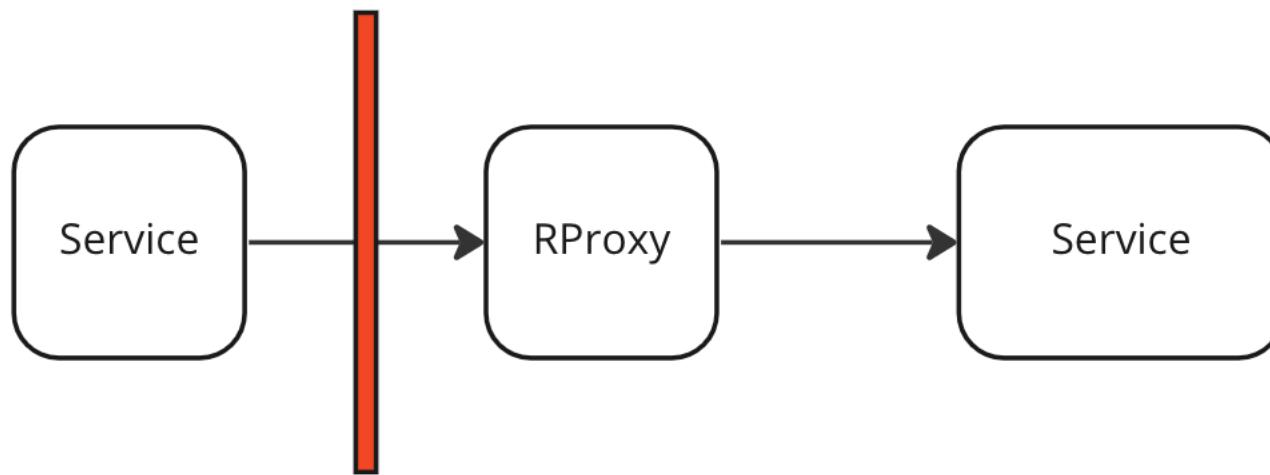
Ведомости



↗ Прочту позже

Роскомнадзор внес социальную сеть Instagram в реестр запрещенных сайтов. Согласно [сервису](#) ведомства по проверке блокировок страниц и сайтов, доступ к Instagram.com ограничен по требованию Генпрокуратуры от 11 марта. По [данным](#) сервиса по мониторингу блокировок Globalcheck, доступность соцсети в России составляет 0%.

Reverse Proxy





Google

Поиск в Google

Мне повезёт!

```
1 vladimirbalun@Vladimirs-MacBook-Pro ~ % nslookup google.com
2 Server:      192.168.1.254
3 Address:     192.168.1.254#53
4
5 Non-authoritative answer:
6 Name:        google.com
7 Address:    173.194.222.100
8 Name:        google.com
9 Address:    173.194.222.101
10 Name:       google.com
11 Address:   173.194.222.113
12 Name:       google.com
13 Address:   173.194.222.139
14 Name:       google.com
15 Address:   173.194.222.138
16 Name:       google.com
17 Address:   173.194.222.102
```

Кэширование

Кэширование

- Сокращение response time сервисов
- Снижение лишней нагрузки с поставщиков данных
- Переиспользованию ранее полученных или вычисленных данных
- Стабилизация работы при кратковременных отказах систем - поставщиков данных

Основные термины

Cache miss - промах кэша, запрошенный ключ не был найден в кэше

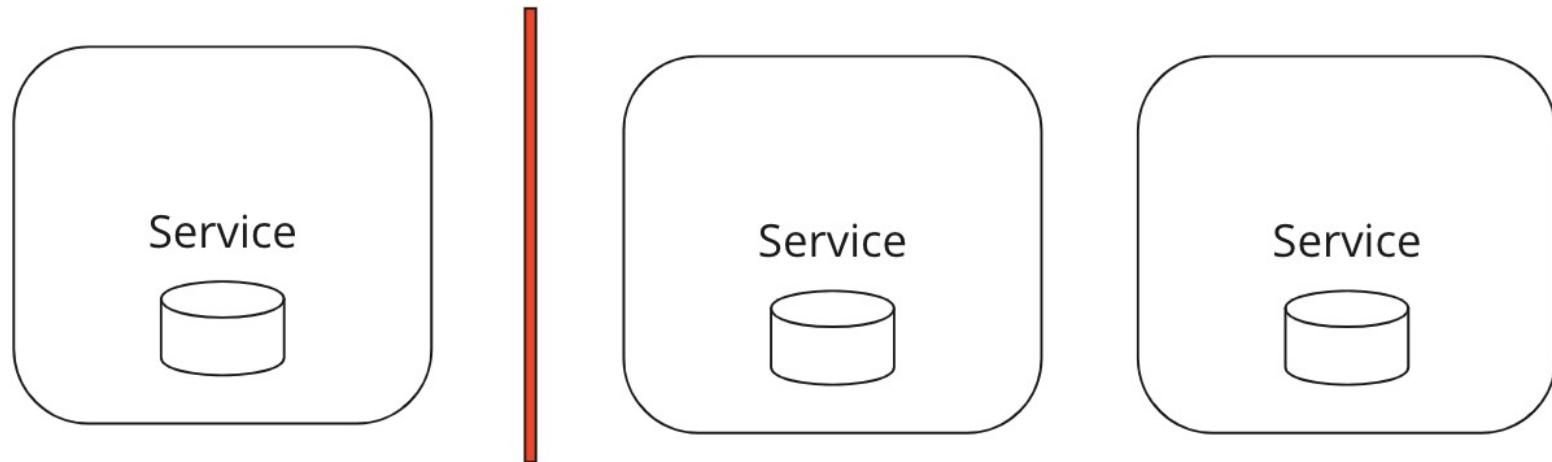
Cache Hit - попадание в кэш, запрошенный ключ найден в кэше

Hit ratio - процент попаданий запросов в кэш, характеризует эффективность кэширования

Прогрев кэша - процесс наполнения кэша данными

Инвалидация - удаление кэшированных данных

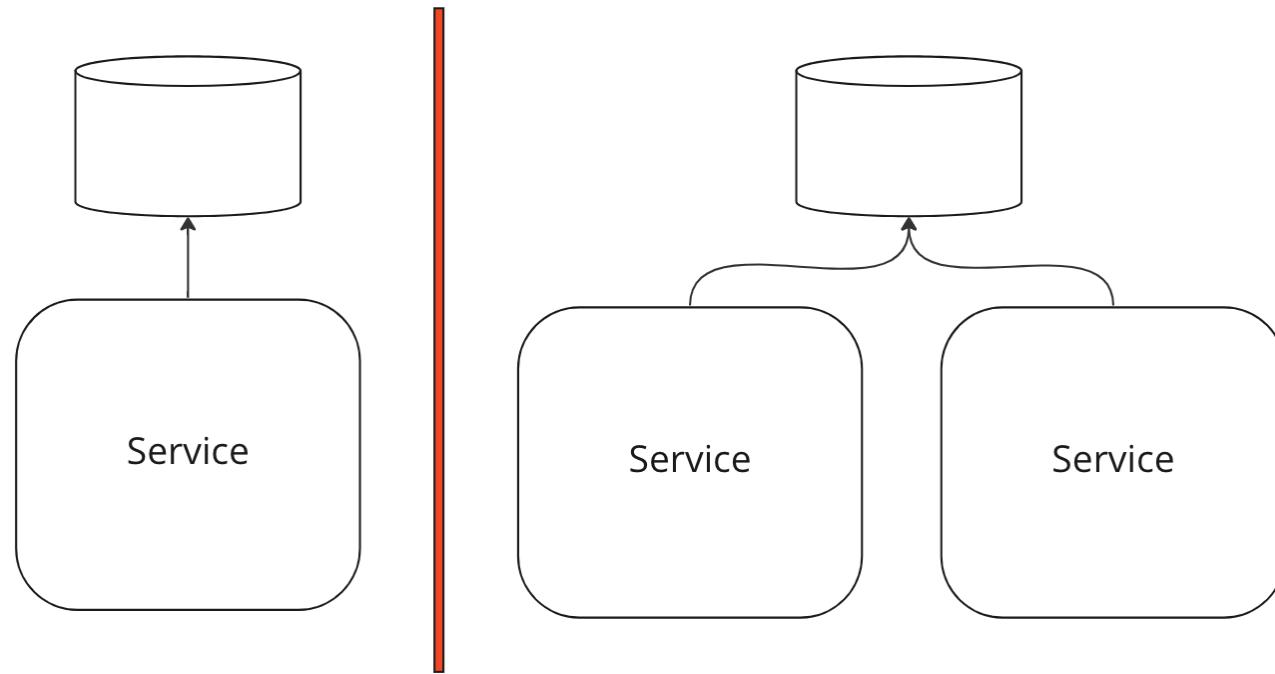
Внутреннее кэширование



Внутреннее кэширование

- + Высокая скорость
- + Отсутствие сетевых запросов
- + Нет расходов на Marshaling/Unmarshalling данных
- Горизонтальное масштабирование
- Прогрев кэша после падения сервиса

Внешнее кэширование

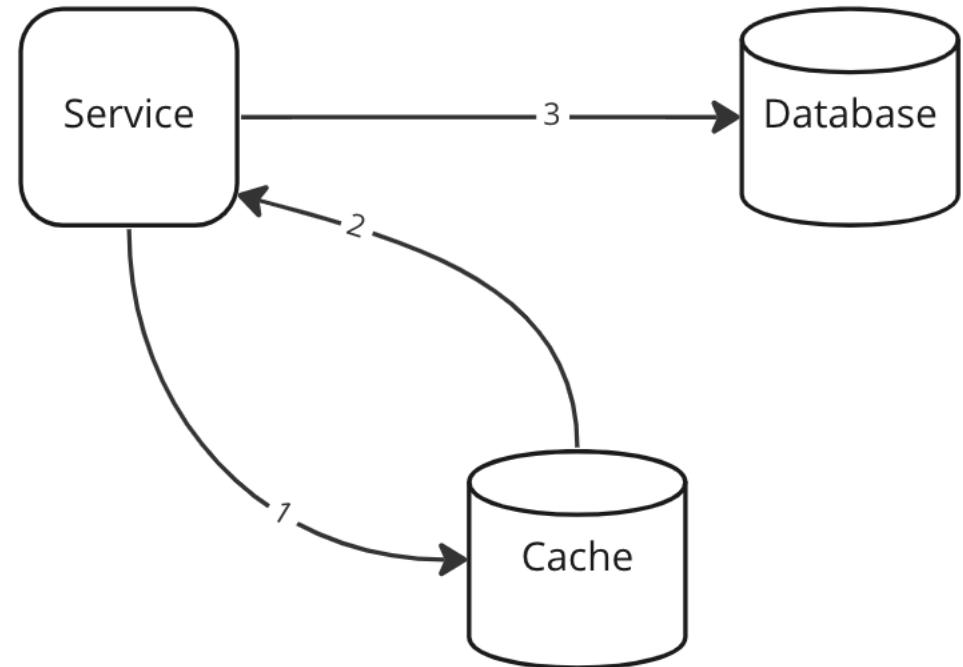


Внешнее кэширование

- + Хранение большого объема данных
- + Простое горизонтальное масштабирование
- + После падения сервиса данные кэша не теряются
- + Простой прогрев кэша и простая логика инвалидации
- Скорость работы

Lazy Caching

Идем в кэш, если элемент найден
отдаем из кэша, если нет то получаем
данные из базы данных и записываем
данные в кэш

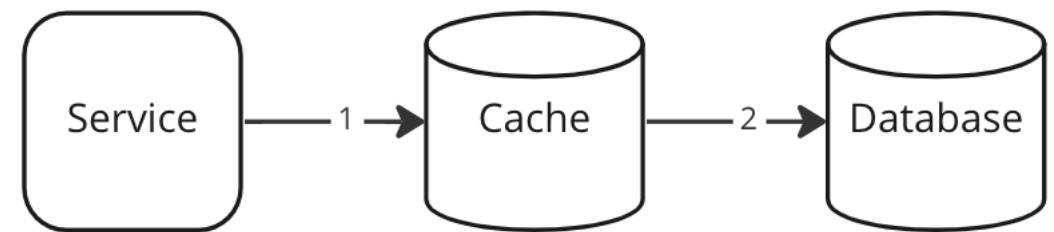


Lazy Caching

- + Кэш содержит только запрашиваемые объекты
- + Новые объекты добавляются в кэш только по мере необходимости
- + Система устойчива к сбоям кэша
- Каждый промах кэша требует совершить три операции
- Данные записываются напрямую в основное хранилище, что может привести к тому, что в кэше окажутся неактуальные данные

Write-Through

Данные всегда записываются в кэш, а
затем в основное хранилище

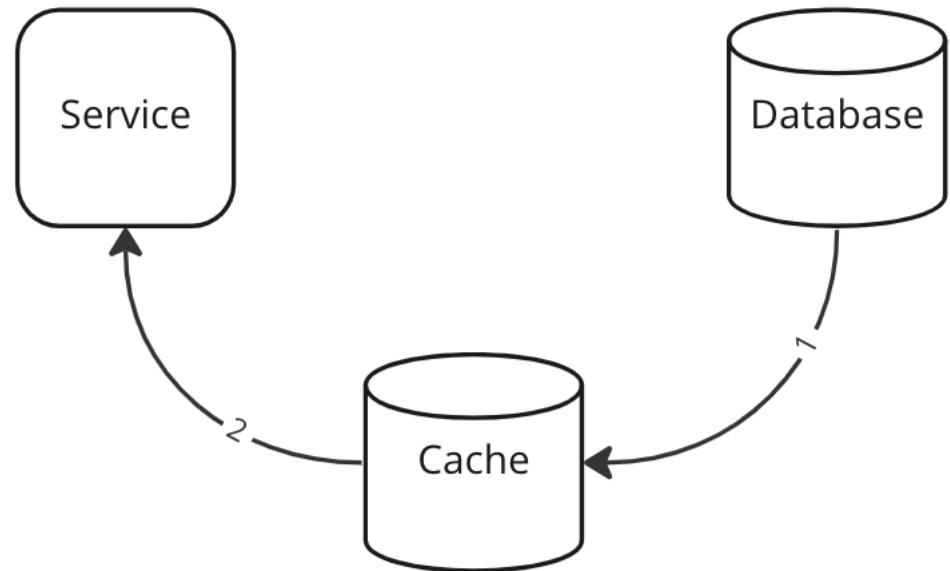


Write-Through

- + Кэш всегда актуален
- + Долгая запись при обновлении данных, но быстрое чтение
- Кэш может быть заполнен ненужными объектами, к которым нет запросов, но потребляют лишнюю память
- Нужен дополнительный механизм, позволяющий заново заполнить кэш при его потере

Read-Through

Приложение всегда получает данные
только из кэша, при этом логика
наполнения кэша выносится в
отдельный модуль

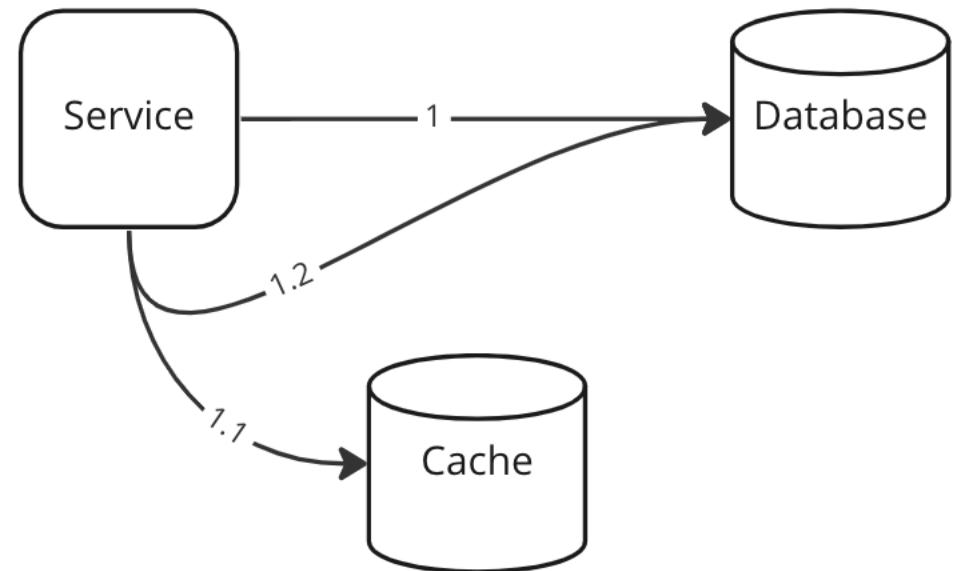


Read-Through

- + Логика приложения становится проще
- Система не устойчива к сбоям кэша
- Сложность реализации

Write-around

Если элемент есть в кэше, то обновляем его в кэше и записываем в основное хранилище, если элемента нет, то сразу записываем в основное хранилище



Алгоритмы замещения данных

FIFO

add - user_1

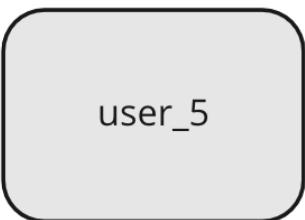
add - user_2

add - user_3

add - user_4



add - user_5



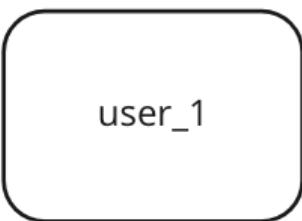
LIFO

add - user_1

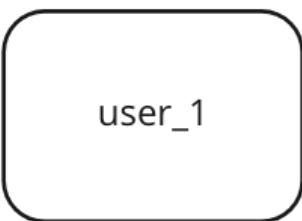
add - user_2

add - user_3

add - user_4



add - user_5



LRU

add - user_1

add - user_2

add - user_3

add - user_4

user_1

user_2

user_3

user_4

get - user_1

add - user_5

user_1

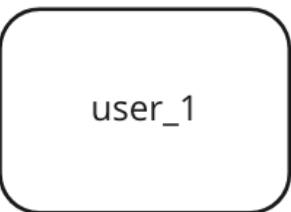
user_5

user_3

user_4

MRU

**add - user_1
add - user_2
add - user_3
add - user_4**

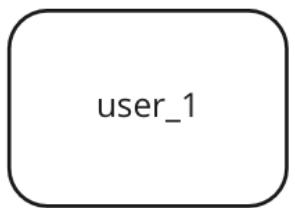


**get - user_3
add - user_5**

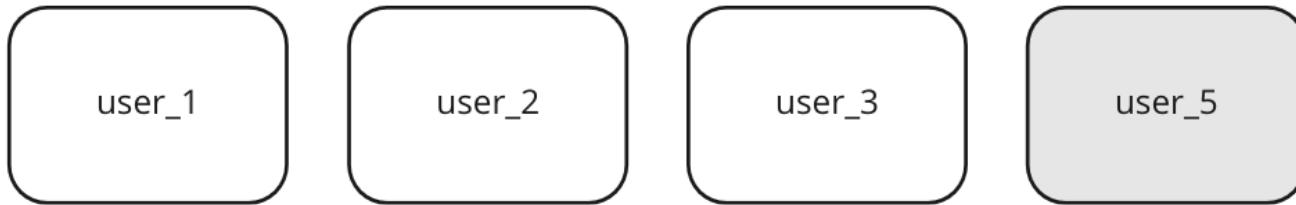


LFU

add - user_1
add - user_2
add - user_3
add - user_4



get - user_3
get - user_1
get - user_4
add - user_5



API

CRUD

Акроним, обозначающий четыре базовые функции, используемые при работе с данными: создание (C), чтение (R), модификация (U), удаление (D)



REST

Глагол – метод запроса (GET, POST, DELETE, PUT)

Существительное – URI запроса

Дополнительная информация – хедеры запроса

Содержимое – тело запроса и ответа в Json

Результат – код ответа

```
1 Request:  
2 GET ${address}/employee/14  
3  
4 Response:  
5 200 - Ok  
6 {  
7   "name": "Petr",  
8   "surname": "Ivanov",  
9   "city": "Moscow"  
10 }  
11
```

SOAP



```
1 <?xml version="1.0"?>
2 <soap:Envelope
3   xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
4   soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
5 <soap:Body>
6   <m:GetPrice xmlns:m="https://online-shop.ru/prices">
7     <m:Item>Dell Vostro 3515-5371</m:Item>
8   </m:GetPrice>
9 </soap:Body>
10 </soap:Envelope>
```



```
1 <?xml version="1.0"?>
2 <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/">
3   soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
4 <soap:Body>
5   <m:GetPriceResponse xmlns:m="https://online-shop.ru/prices">
6     <m:Price>37299</m:Price>
7   </m:GetPriceResponse>
8 </soap:Body>
9 </soap:Envelope>
```

RPC

Класс технологий, позволяющих программам
вызывать функции или процедуры в другом
адресном пространстве



gRPC

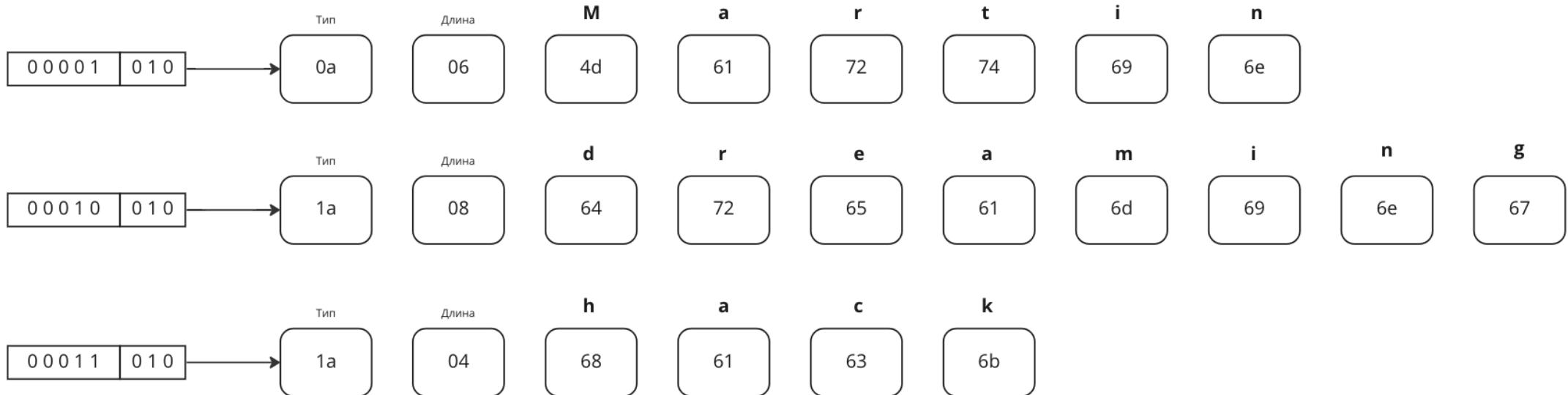
```
● ○ ●

1 message Person {
2   string name = 1;
3   int32 id = 2;
4   string email = 3;
5
6   enum PhoneType {
7     MOBILE = 0;
8     HOME = 1;
9     WORK = 2;
10 }
11
12 message PhoneNumber {
13   string number = 1;
14   PhoneType type = 2;
15 }
16
17 repeated PhoneNumber phones = 4;
18 }
19
20
21 service PersonsList {
22   rpc SavePerson(Person) returns (google.protobuf.Empty) {}
23 }
```

```
● ○ ●

1 person := pb.Person{
2   Id:    1234,
3   Name: "John Doe",
4   Email: "jdoe@example.com",
5   Phones: []*pb.Person_PhoneNumber{
6     {Number: "555-4321", Type: pb.Person_HOME},
7   },
8 }
9
10 _, err := client.SavePerson(context.Background(), person)
11 if err != nil {
12   ...
13 }
```

Внутреннее устройство



Under / Over fetching

GraphQL

```
1 query {  
2   users {  
3     fname  
4     age  
5   }  
6 }
```

```
1 data : {  
2   users [  
3     {  
4       "fname": "Joe",  
5       "age": 23  
6     },  
7     {  
8       "fname": "Betty",  
9       "age": 29  
10    }  
11  ]  
12 }
```

GraphQL



```
1 mutation createUser{  
2   addUser(fname: "Richie", age: 22) {  
3     id  
4   }  
5 }
```



```
1 data : {  
2   addUser : "a36e4h"  
3 }
```

GraphQL

```
1 subscription listenLikes {  
2   listenLikes {  
3     fname  
4     likes  
5   }  
6 }
```

```
1 data: {  
2   listenLikes: {  
3     "fname": "Richie",  
4     "likes": 245  
5   }  
6 }
```

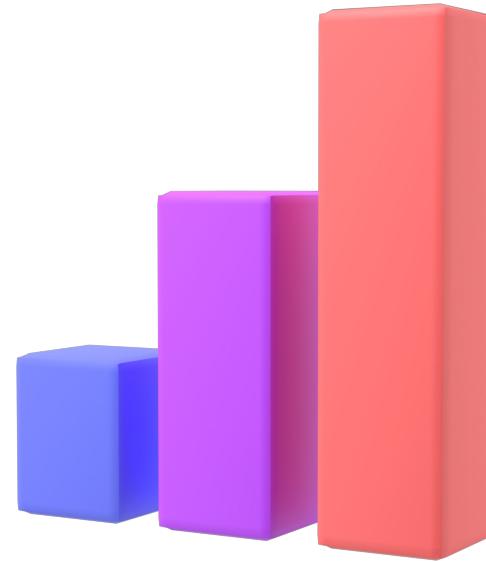
Свой собственный API???

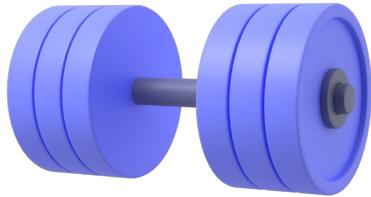
Резюме

REST для клиентского взаимодействия

gRPC для внутреннего взаимодействия

GraphQL для кастомизации запросов





Let's practise

Базы данных

OLAP vs OLTP

Online Analytical Processing и Online Transaction Processing

Реляционные

UserID	Name	CityID
23	user_1	1
233	user_2	2
12	user_5	1

CityID	Name
1	Moscow
2	Berlin

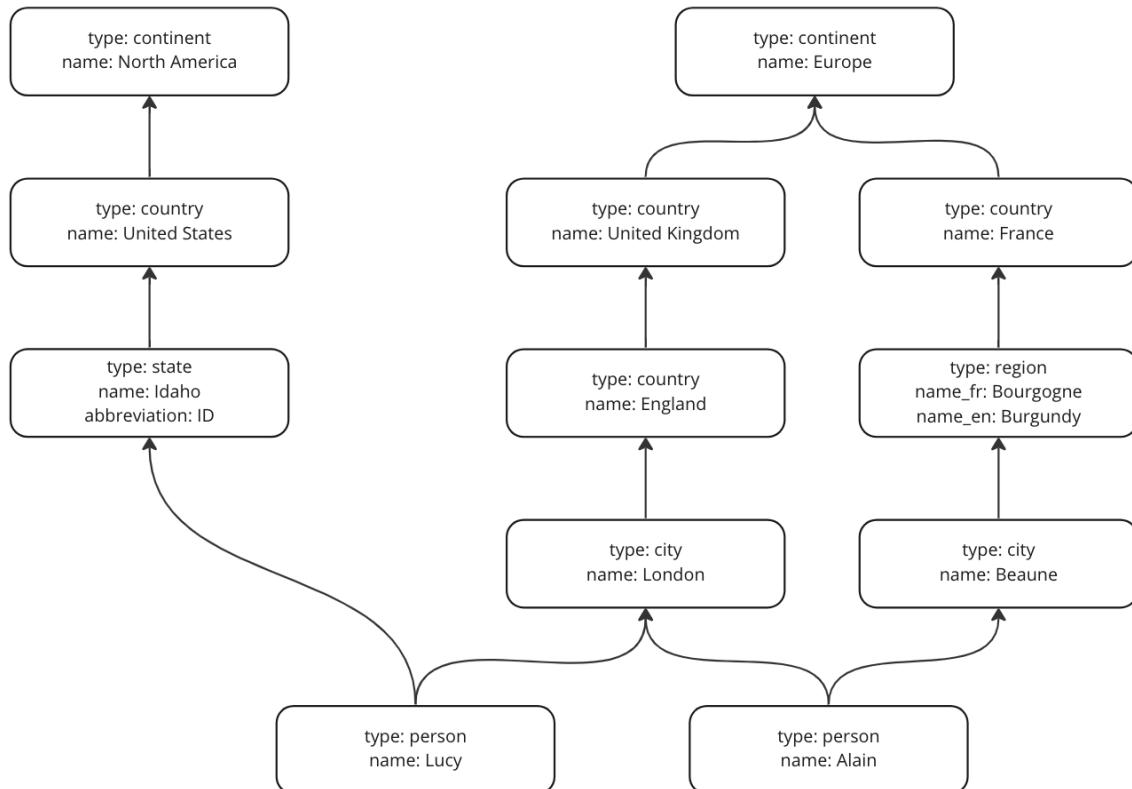


Документоориентированные

```
{  
    "name": "Petr",  
    "surname": "Ivanov",  
    "city" : "Moscow",  
    "interests" : [  
        "football",  
        "hockey"  
    ],  
}
```



Графовые



Key-value

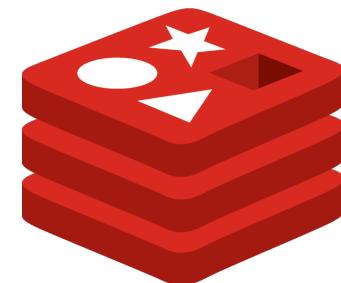
Key	Value
video_1	4232
video_2	23
video_3	2342344
video_4	2323
video_5	434



MEMCACHED



Tarantool



redis

Колоночные

UserID	Name	City
23	user_1	Moscow
233	user_2	Berlin
12	user_5	Moscow

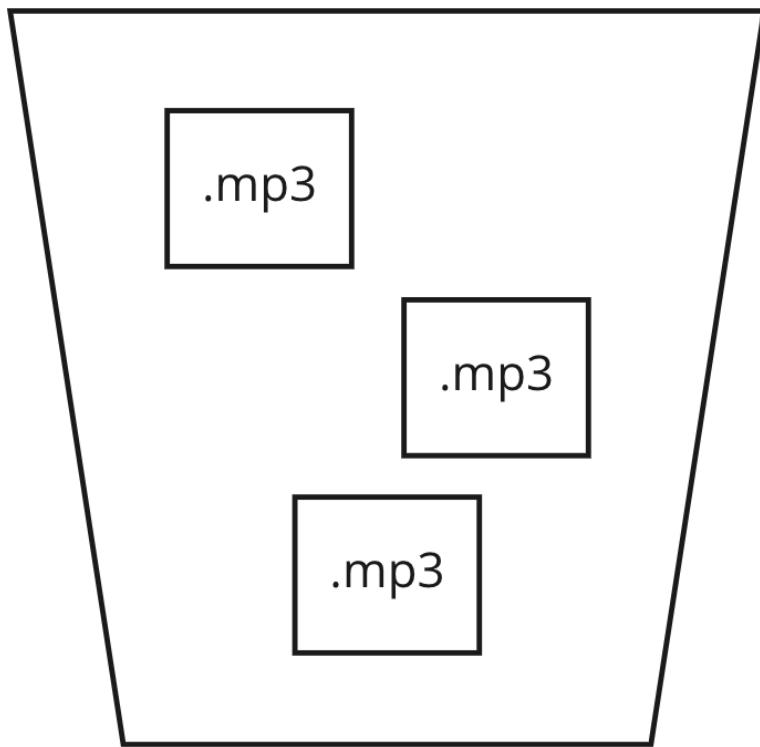


Time series

Timestamp	Temperature
10.01.2022 17:00	37.2
10.01.2022 17:01	37.1
10.01.2022 17:02	36.8



Blob store



Выбор базы данных

1. Транзакции
2. Формат данных
3. Навык работы с технологией
4. Характер обращений к данным
5. Сообщество и зрелость технологии
6. Частота изменяемости формата данных

Data retention

Хранение данных определяет политики управления постоянными данными и записями для соблюдения юридических и бизнес-требований к архивированию данных.

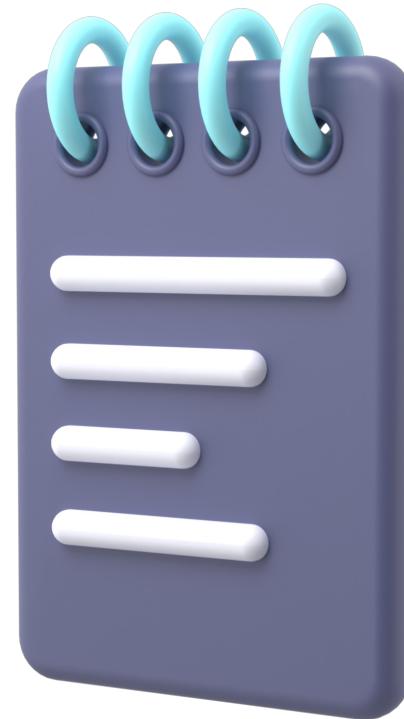


Индексы

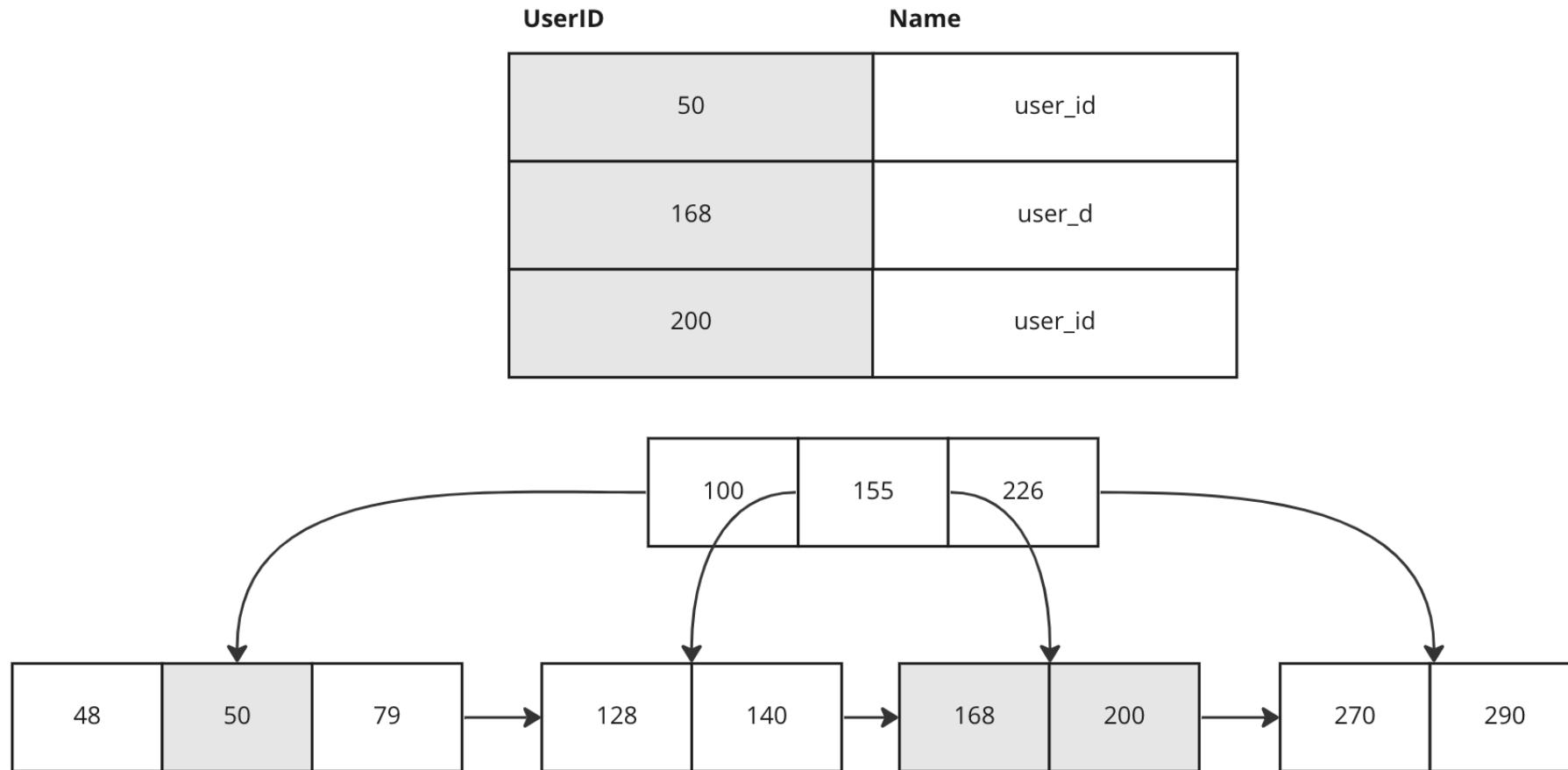
Ускоряют чтение

Замедляют запись

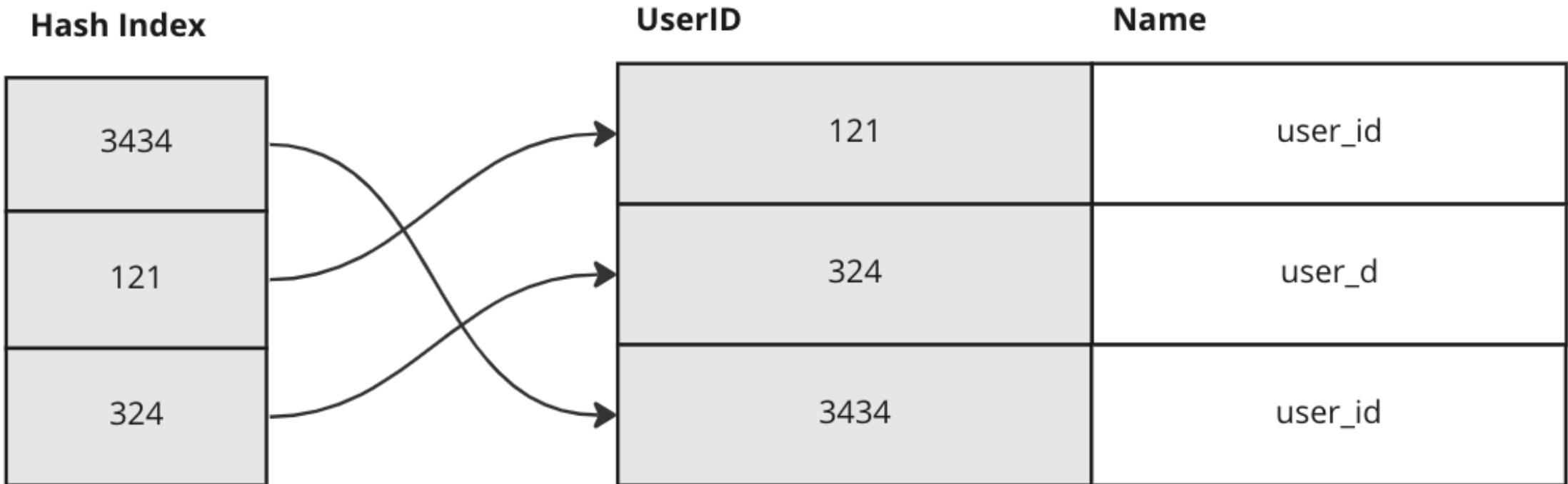
Используют дополнительную память



BTree



Hash



Bitmap

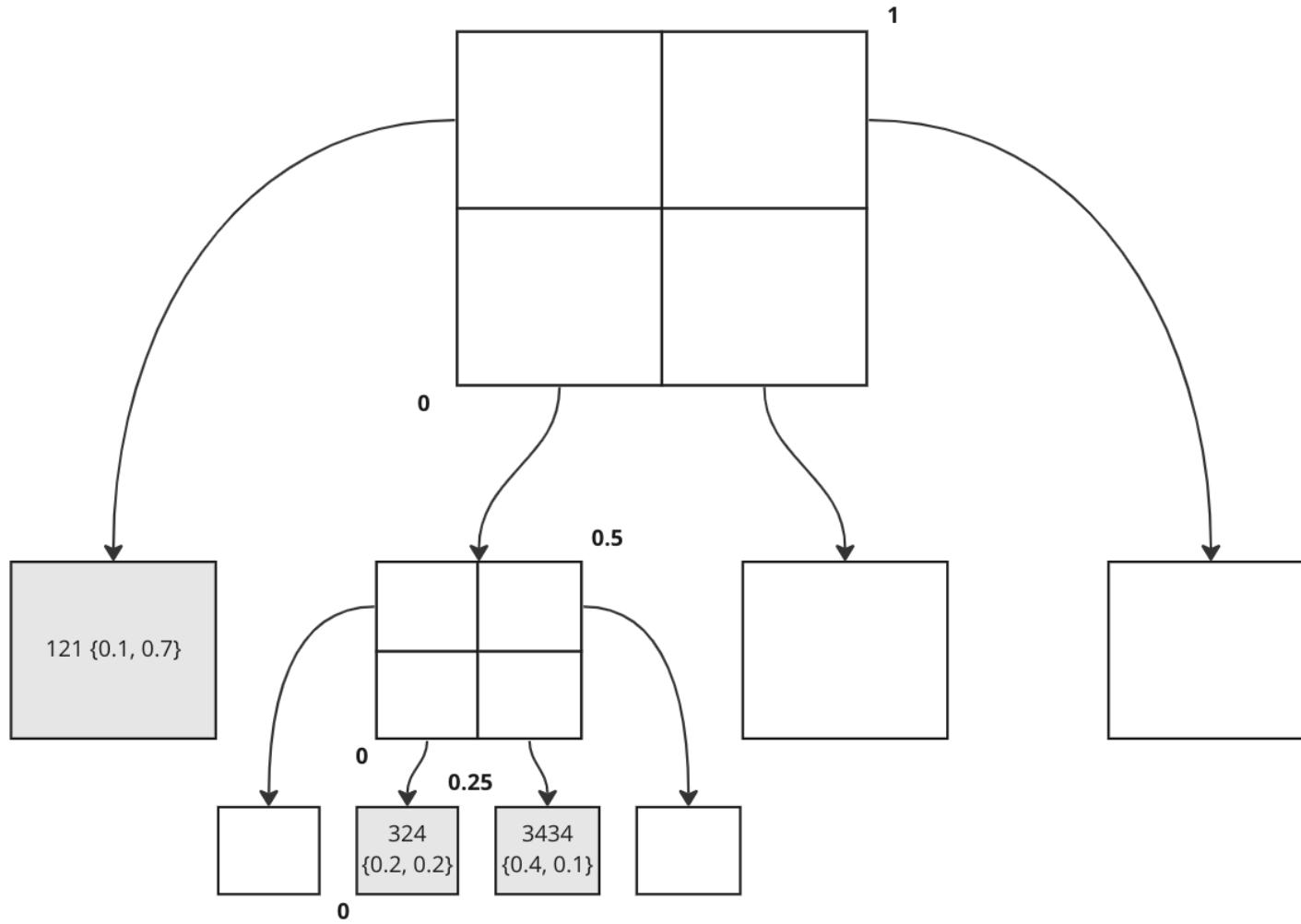
Bitmap Index

3434	1	1
121	1	0
324	0	1

UserID

UserID	Smoking	Drinking
121	true	false
324	false	true
3434	true	true

Spatial



UserID	X	Y
121	0.1	0.7
324	0.2	0.2
3434	0.4	0.1

Reversed

Word	TextIDs
a	[2]
banana	[2]
is	[0, 1, 2]
it	[0, 1, 2]
what	[0, 1]

ID	Text
0	it is what it is
1	what is it
2	it is a banana

Кластерные и некластерные

При наличии кластерного индекса строки таблицы упорядочены по значению ключа этого индекса. Если в таблице нет кластерного индекса, таблица называется кучей. Некластерный индекс, созданный для такой таблицы, содержит только указатели на записи таблицы

Покрывающий индекс

Покрывающими индексами называются **некластеризованные индексы**, которые разрешают один или несколько схожих результатов запроса напрямую, без доступа к базовой таблице и без уточняющих запросов.

Транзакции

ACID

Популярные SQL базы данных, появились как раз на почве ACID.

ACID – это стандарт того, какие гарантии должна давать база данных, чтобы поддерживать транзакции (он не указывает деталей реализации)

Атомарность

Каждая транзакция базы данных является единым блоком,
который использует подход «все или ничего» к выполнению.

Если какой-либо оператор в транзакции терпит неудачу, вся
транзакция откатывается

user_id (111) = 1500
user_id (222) = 1500

```
UPDATE account  
SET balance = balance - 500  
WHERE user_id = 111
```

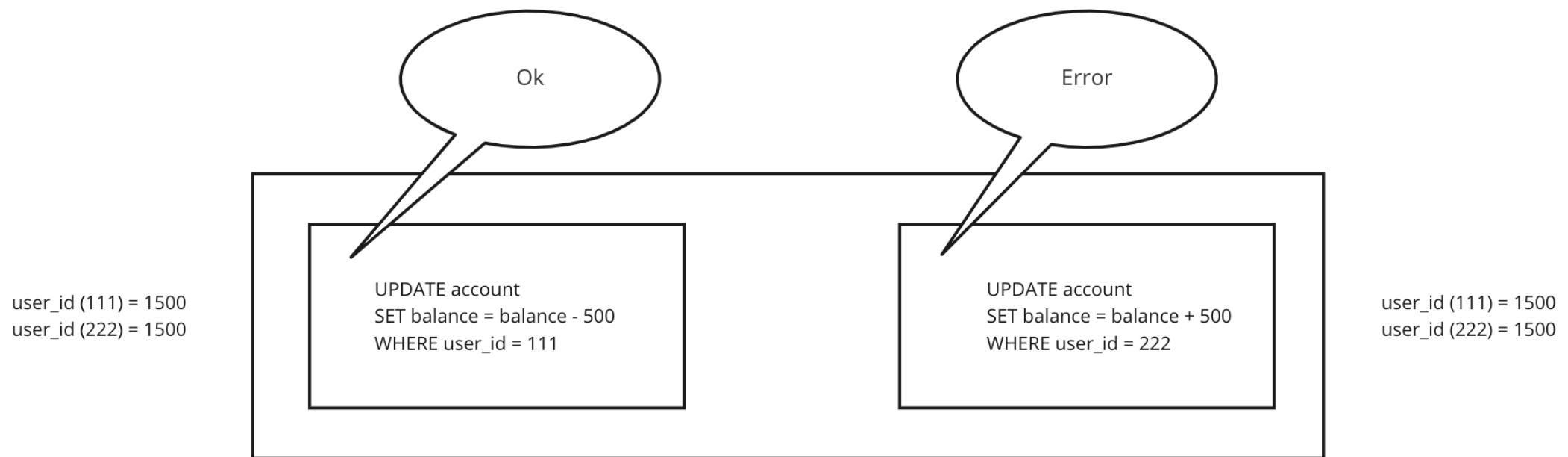
Ok

user_id (111) = 1000
user_id (222) = 1500

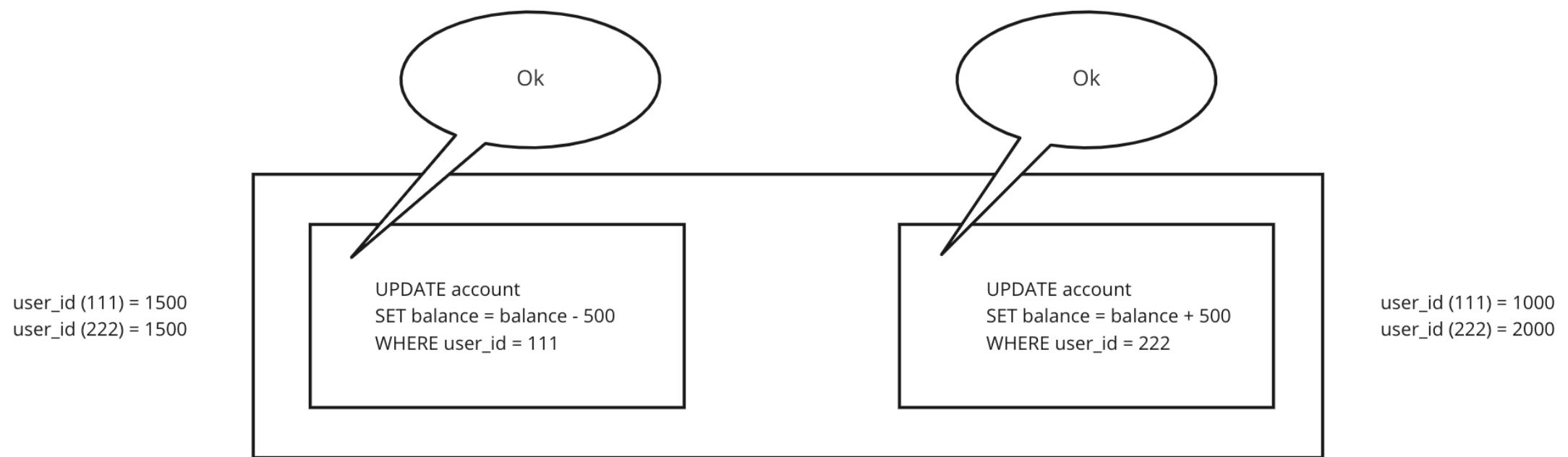
```
UPDATE account  
SET balance = balance + 500  
WHERE user_id = 222
```

Error

Rollback



Commit



Изоляция транзакций

Каждая транзакция происходит до или после каждой другой транзакции, и представление базы данных, которое транзакция видит в своем начале, изменяется только самой транзакцией до ее завершения. Ни одна транзакция не должна видеть промежуточный продукт другой транзакции

Потерянное обновление

video_id (10) = 100

```
UPDATE view  
SET count = count + 1  
WHERE video_id = 10;
```

video_id (10) = 101

video_id (10) = 100

```
UPDATE view  
SET count = count + 1  
WHERE video_id = 10;
```

video_id (10) = 101

Грязное чтение

user_id (111) = 1500

```
START TRANSACTION;  
  
UPDATE account  
SET balance = balance - 500  
WHERE user_id = 111;
```

```
ROLLBACK;
```

user_id (111) = 1500

user_id (111) = 1500

```
SELECT balance  
FROM account  
WHERE user_id = 111;
```

user_id (111) =
1000

user_id (111) = 1500

Неповторяющееся чтение

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;  
  
UPDATE account  
SET balance = balance + 500  
WHERE user_id = 111;  
  
COMMIT;
```

user_id (111) = 2000
user_id (222) = 2500

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;
```

```
SELECT SUM(balance)  
FROM account;
```

sum = 4000

```
SELECT SUM(balance)  
FROM account;
```

sum = 4500

user_id (111) = 2000
user_id (222) = 2500

Чтения «фантомов»

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;  
  
INSERT INTO  
    account(user_id, balance)  
VALUES (333, 1000);  
  
COMMIT;
```

user_id (111) = 2000
user_id (222) = 2500
user_id (333) = 1000

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;  
  
SELECT SUM(balance)  
FROM account;  
  
SELECT SUM(balance)  
FROM account;
```

sum = 4000

sum = 5000

user_id (111) = 2000
user_id (222) = 2500
user_id (333) = 1000

Уровни изоляции

READ_UNCOMMITTED - могут происходить грязные чтения, неповторяющиеся чтения, фантомные чтения и потерянное обновление

READ_COMMITTED - грязные чтения предотвращены, но могут возникать неповторяющиеся чтения, фантомные чтения и потерянные обновления

REPEATABLE_READ - грязные чтения, неповторяющиеся чтения и потерянное обновления предотвращены, но могут возникать фантомные чтения

SERIALIZABLE - транзакции полностью изолированы - исключено влияние одной транзакции на другую в момент выполнения

Согласованность

Различные утверждения относительно данных
(инварианты) должны всегда оставаться справедливыми

Constraint	Description
NOT NULL	values cannot be null
UNIQUE	values cannot match any older value
PRIMARY KEY	used to uniquely identify a row
FOREIGN KEY	references a row in another table
CHECK	validates condition for new value
DEFAULT	set default value if not passed
CREATE INDEX	used to speedup the read process

Deferrable transactions

```
CREATE TABLE husbands (
    id int PRIMARY KEY,
    wife_id int NOT NULL
);

CREATE TABLE wives (
    id int PRIMARY KEY,
    husband_id int NOT NULL
);

ALTER TABLE husbands ADD CONSTRAINT h_w_fk
FOREIGN KEY (wife_id) REFERENCES Wives;

ALTER TABLE wives ADD CONSTRAINT w_h_fk
FOREIGN KEY (husband_id) REFERENCES husbands;
```

Устойчивость

Гарантирует, что после фиксации транзакции в базе данных она постоянно сохраняется с помощью резервных копий и журналов транзакций. В случае сбоя эти механизмы могут использоваться для восстановления зафиксированных транзакций

WAL

Изменения в файлах с данными должны записываться только после того, как эти изменения были занесены в журнал. Записывать страницы данных на диск после подтверждения каждой транзакции нет необходимости, потому что мы знаем, что если случится сбой, то у нас будет возможность восстановить базу данных с помощью журнала

BASE

Как правило, NoSQL базы данных предоставляют ограниченные версии атомарности и изоляции, но им нужно продавать себя в красивой упаковке, поэтому они придумали свою аббревиатуру

Basically Available

Базы данных будут обеспечивать доступность данных, распространяя и реплицируя их по узлам кластера базы данных

Soft State

Из-за отсутствия строгой согласованности значения
данных могут меняться со временем. Модель BASE разрывается с
концепцией базы данных, которая обеспечивает собственную
согласованность, делегируя эту ответственность разработчикам

Eventual Consistency

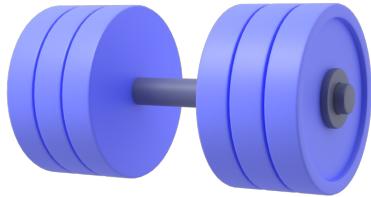
Вы в конце концов увидите действительные данные,
но есть вероятность, что ваша транзакция прочитает
недействительные значения – то есть, временные, или частично
обновлённые, или устаревшие. Возможно, данные станут
согласованными в «ленивом» режиме при чтении

Embedded database

An **embedded database** is a system, which is tightly integrated with an application software it is embedded in the application

In-memory базы данных

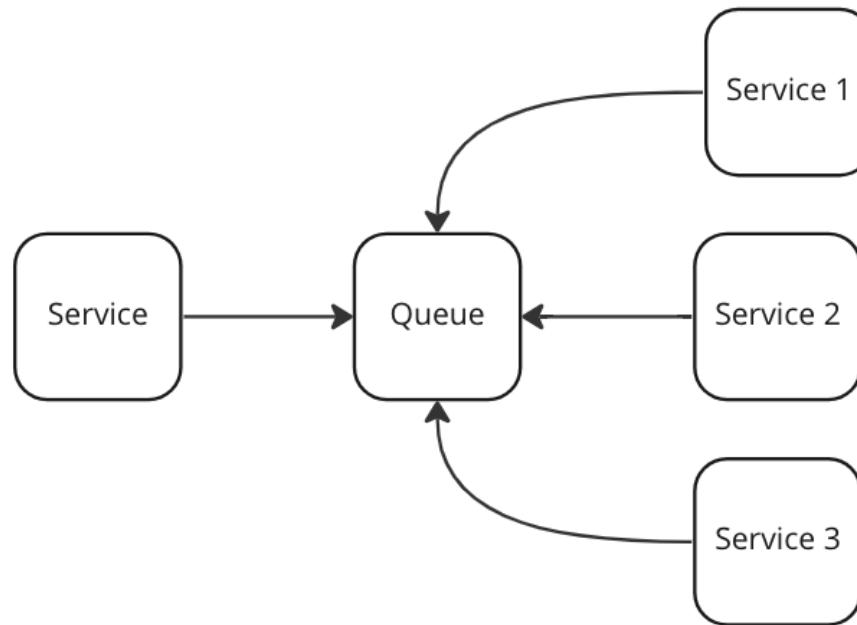
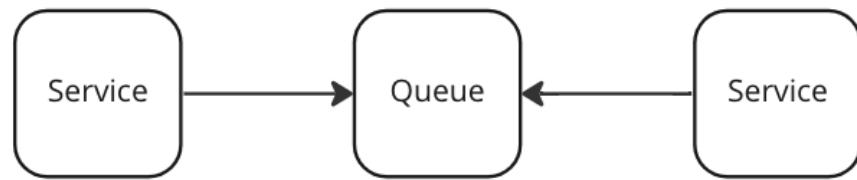
- Можно периодически записывать на диск копии состояния БД
- Можно записывать на диск журналы изменений (что-то вроде упомянутых выше WAL), но не сами данные
- Можно проводить репликацию состояния оперативной памяти на другие машины



Let's practise

Брокеры сообщений

Брокеры сообщений



Брокеры сообщений

Буферизация

Асинхронная связь

Слабое связывание

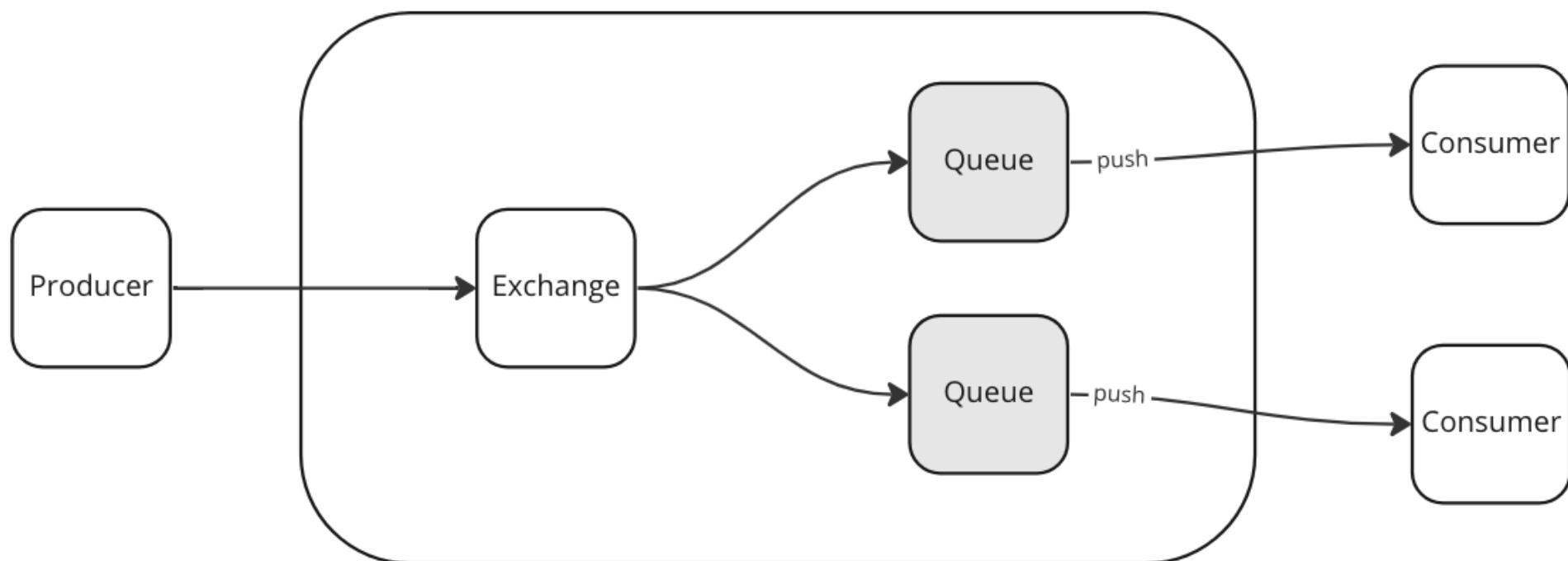
Масштабируемость

Отказоустойчивость

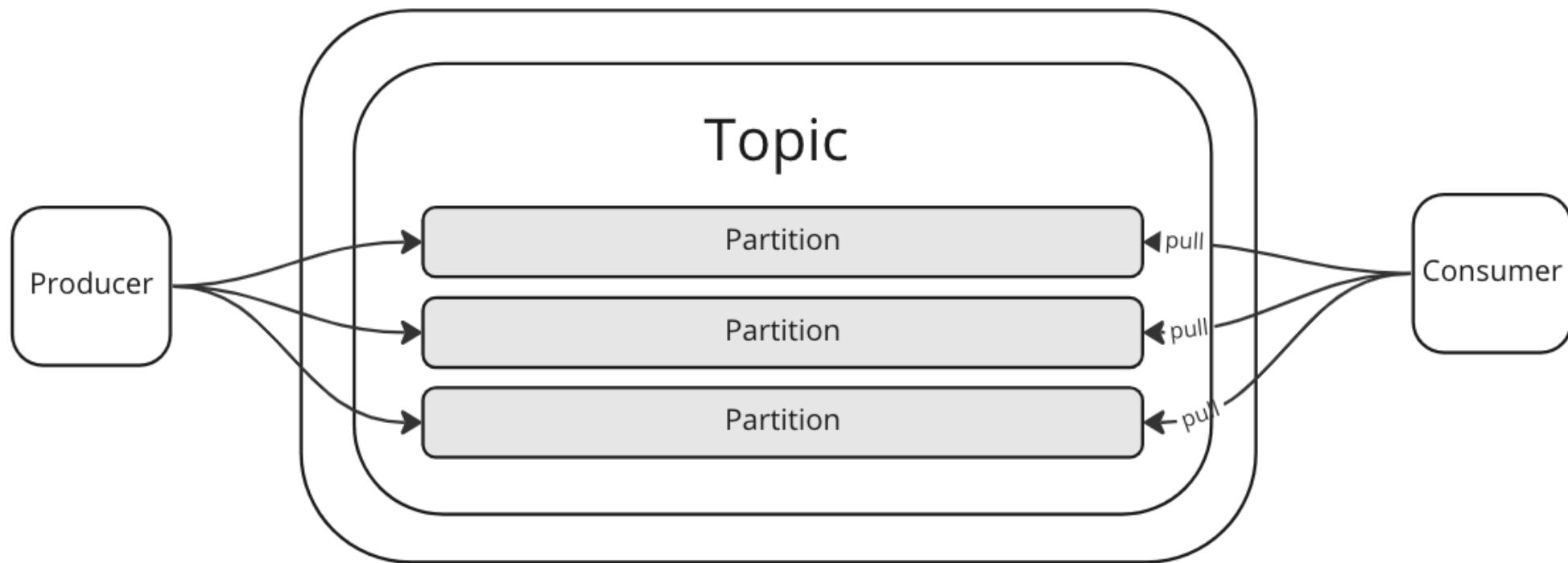
Понимание потоков данных



RabbitMQ



Kafka

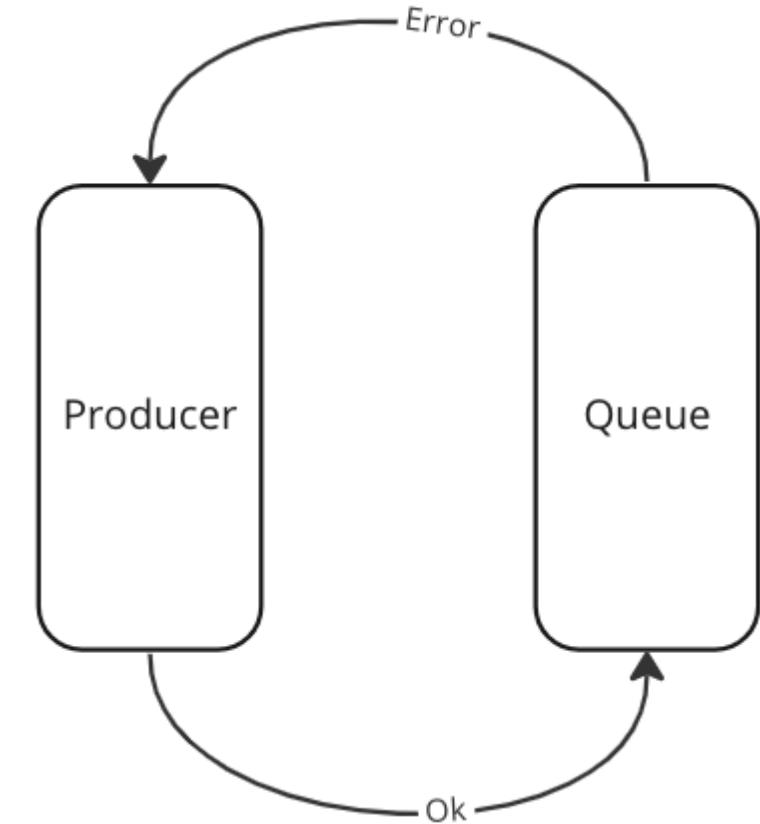


Гарантии доставки

At least once - сообщение будет доставлено хотя бы один раз

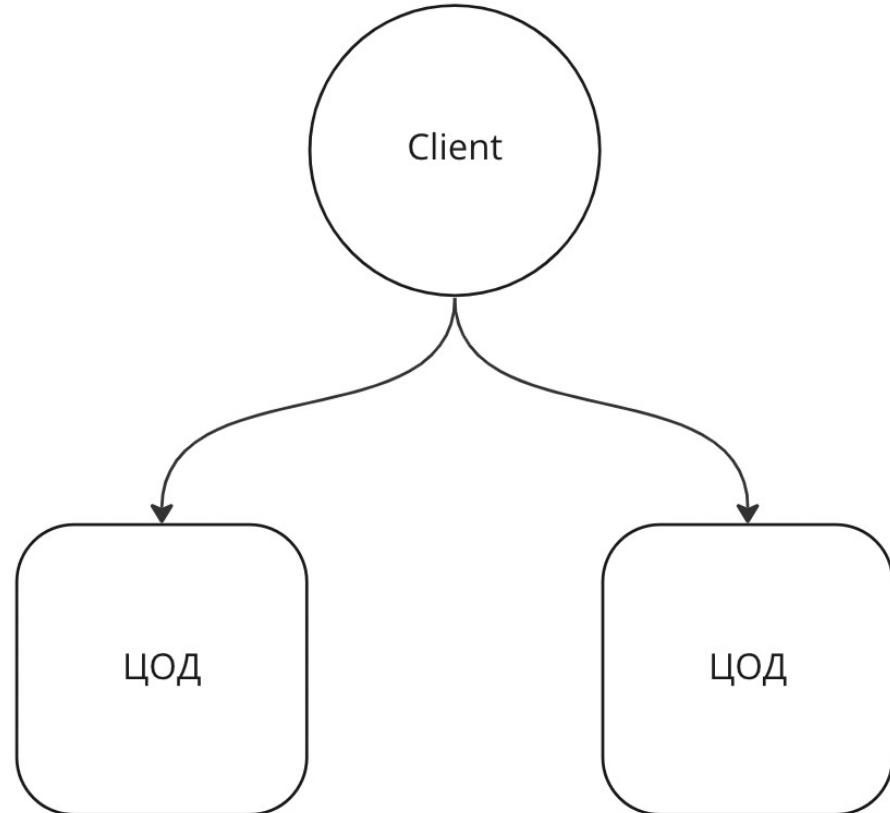
At most once - сообщение будет доставлено не более одного раза

Exactly once - сообщение будет доставлено ровно один раз



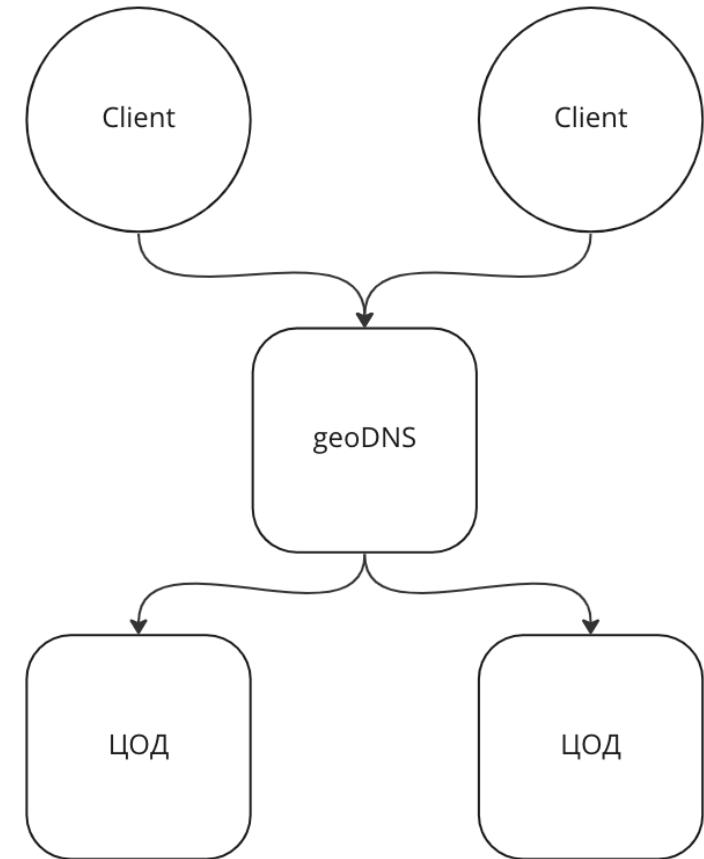
ЦОД

Центр обработки данных



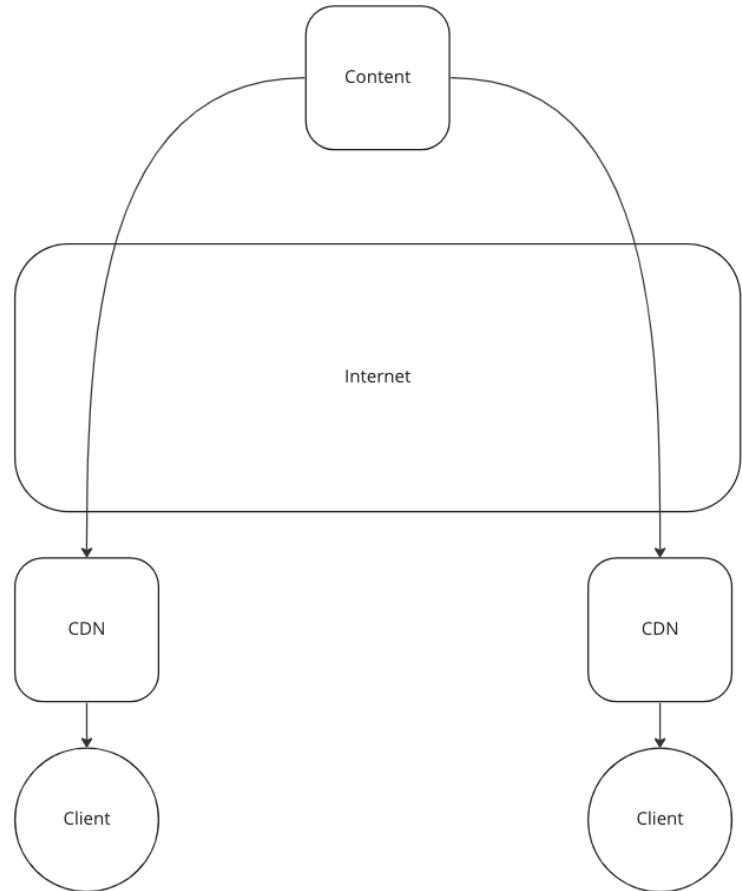
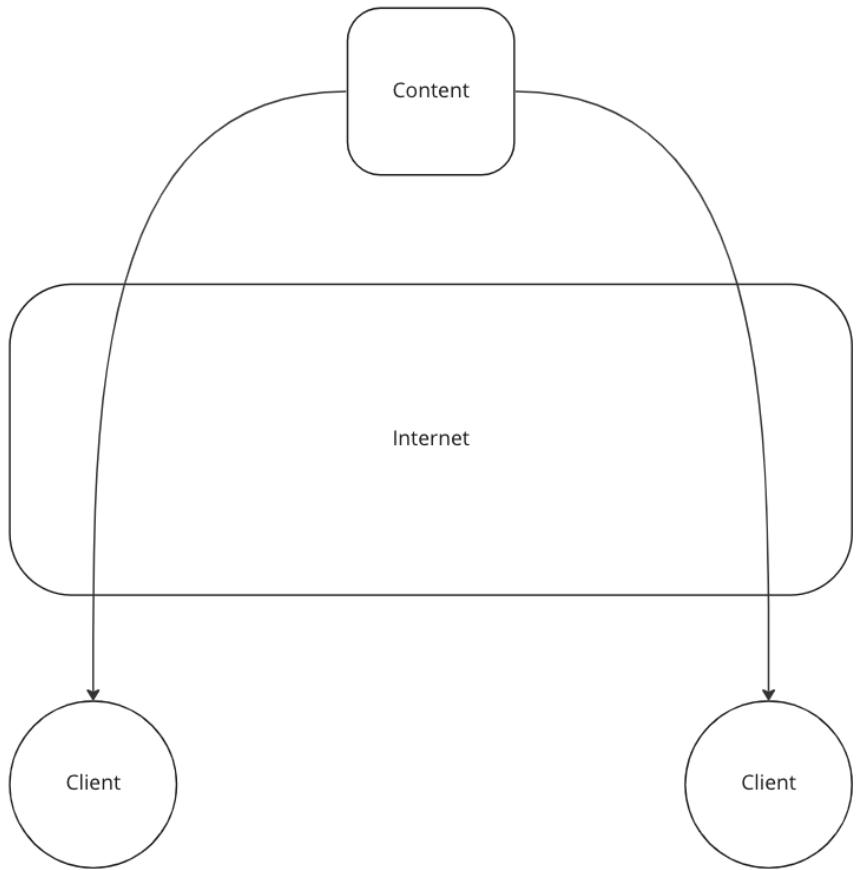
geoDNS

С помощью GeoDNS можно привязать к одному доменному имени несколько IP-адресов. В зависимости от географического положения (определяется по IP-адресу, с которого пришел запрос) пользователь перенаправляется на ближайший сервер



- В 2010 году средний размер веб-страницы составлял 481 КБ, а в 2019 – уже 1936.7 КБ
- По данным опросов 25% пользователей уходят с веб-страницы, если она загружается дольше 4 секунд. 74% пользователей, загружающих сайт с мобильного устройства, предпочитают не ждать, если загрузка длится более 5 секунд

CDN



CDN

- **Исходный (origin)**, на котором размещен запрашиваемый сайт, а также связанный с ним визуальный, музыкальный, видеоконтент
- **PoP (point of presence)** – точка присутствия вспомогательных серверов. Их сеть размещается в различных регионах
- **Proxy-сервер** – это промежуточное звено между пользователем и исходным сервером. Он отвечает за перенаправление, оптимизацию и преобразование передаваемого трафика.

Observability

Мониторинг

Prometheus уже как стандарт...



```
1 func recordMetrics() {
2     go func() {
3         for {
4             opsProcessed.Inc()
5             time.Sleep(2 * time.Second)
6         }
7     }()
8 }
9
10 var (
11     opsProcessed = promauto.NewCounter(prometheus.CounterOpts{
12         Name: "myapp_processed_ops_total",
13         Help: "The total number of processed events",
14     })
15 )
16
17 func main() {
18     recordMetrics()
19
20     http.Handle("/metrics", promhttp.Handler())
21     http.ListenAndServe(":2112", nil)
22 }
```

```
1 # HELP myapp_processed_ops_total The total number of processed
2 # events
2 # TYPE myapp_processed_ops_total counter
3 myapp_processed_ops_total 5
```

Grafana



Основные метрики

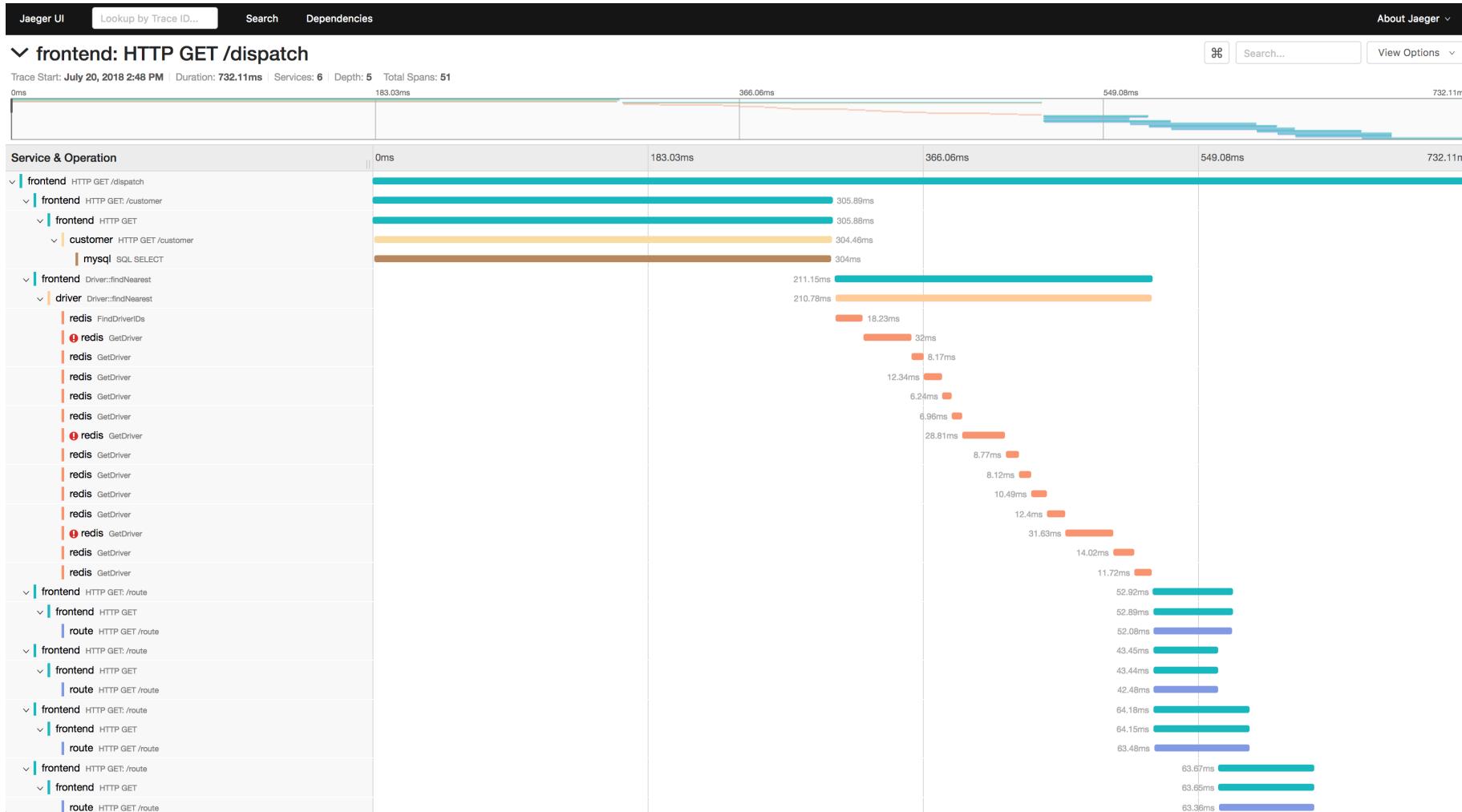
- PRS, TPS, QPS
- Response time
- Errors rate
- Traffic, CPU, RAM, HDD/SSD
- Uptime / Downtime
- Размеры очередей
- Количество процессов / потоков

Трейсинг

Jaeger или Zipkin по умолчанию

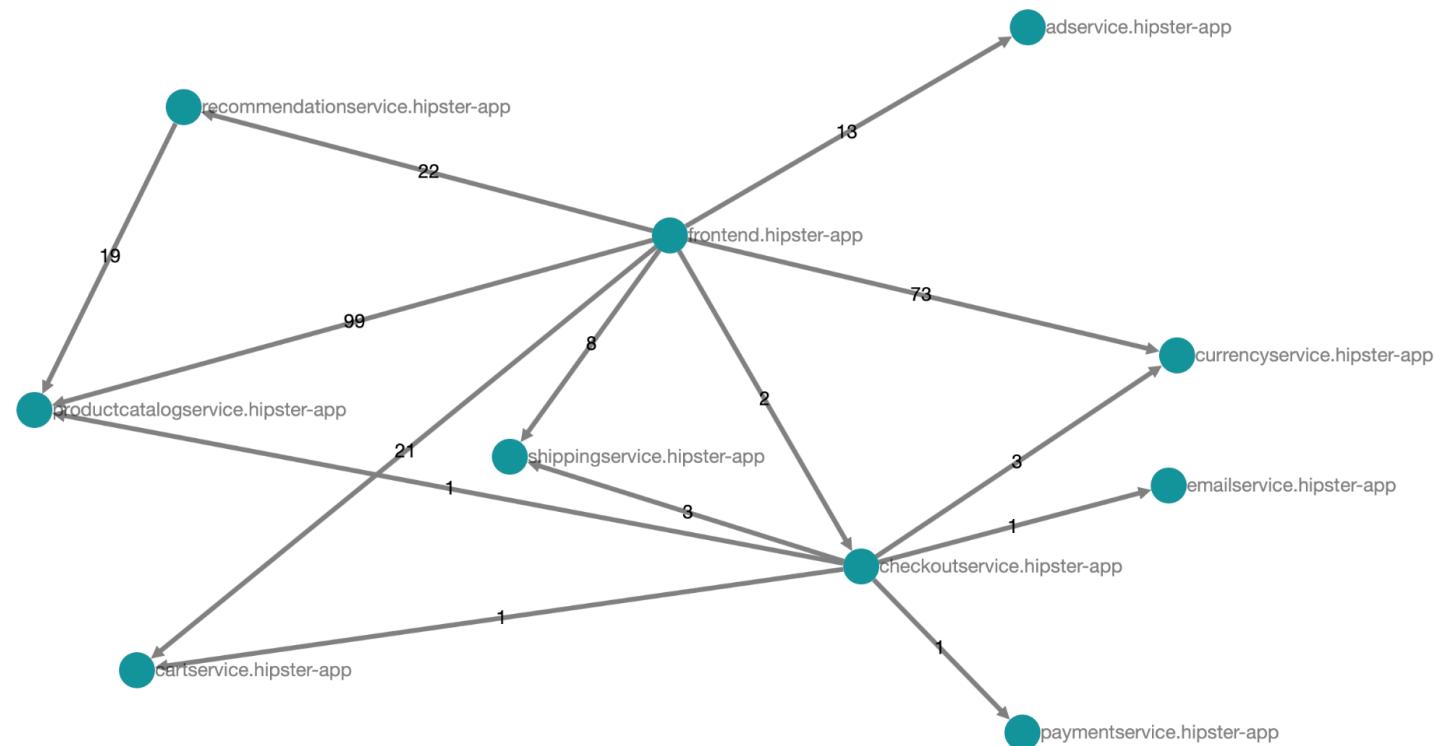


Jaeger



Force Directed Graph

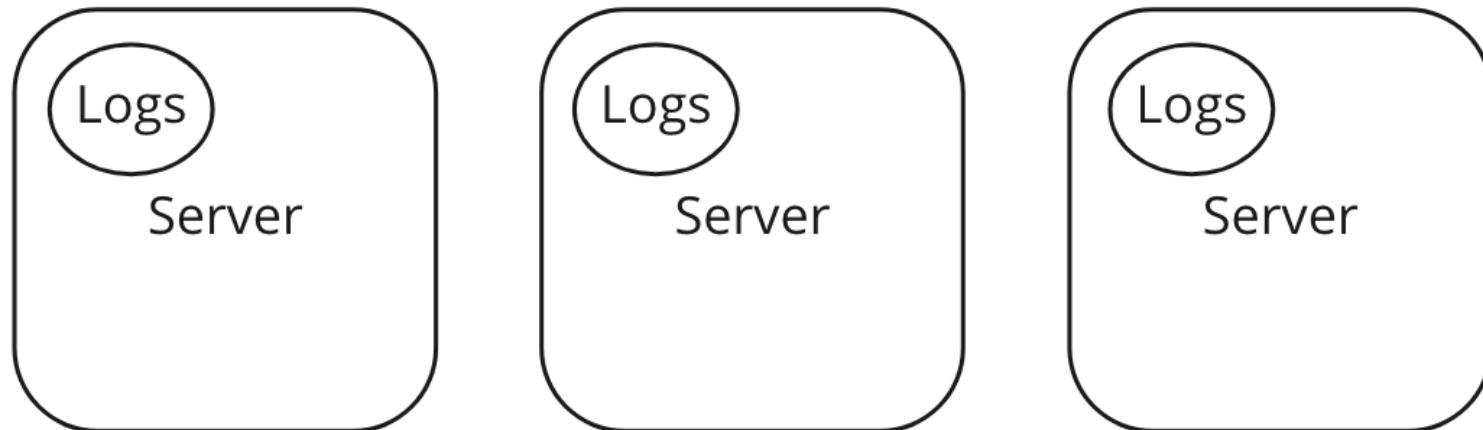
DAG



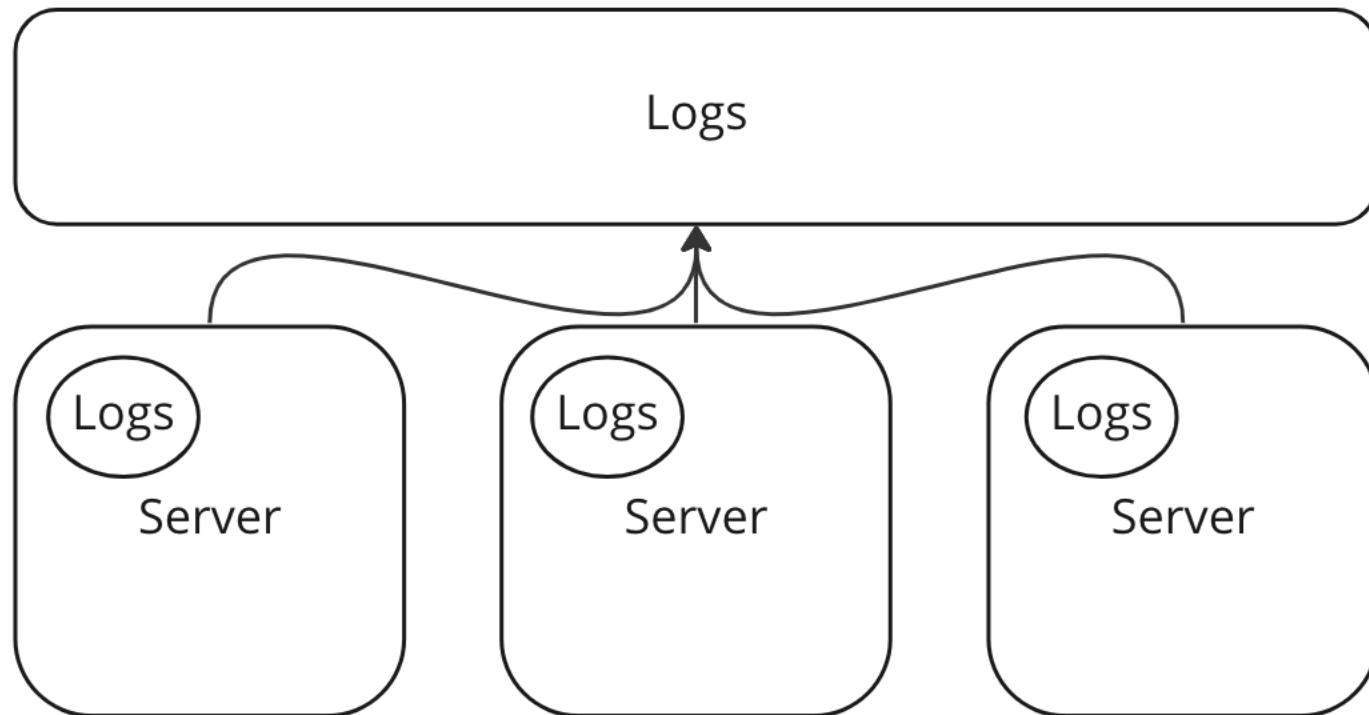


```
1 func isLoggedIn(ctx context.Context, r *http.Request) bool {  
2     span, ctx := opentracing.StartSpanFromContext(ctx, "isLoggedIn")  
3     defer span.Finish()  
4     ...  
5  
6  
7 func getCityByCountryName(ctx context.Context, countryName string) ([]City, error) {  
8     span, ctx := opentracing.StartSpanFromContext(ctx, "getCityByCountryName")  
9     defer span.Finish()  
10    ...
```

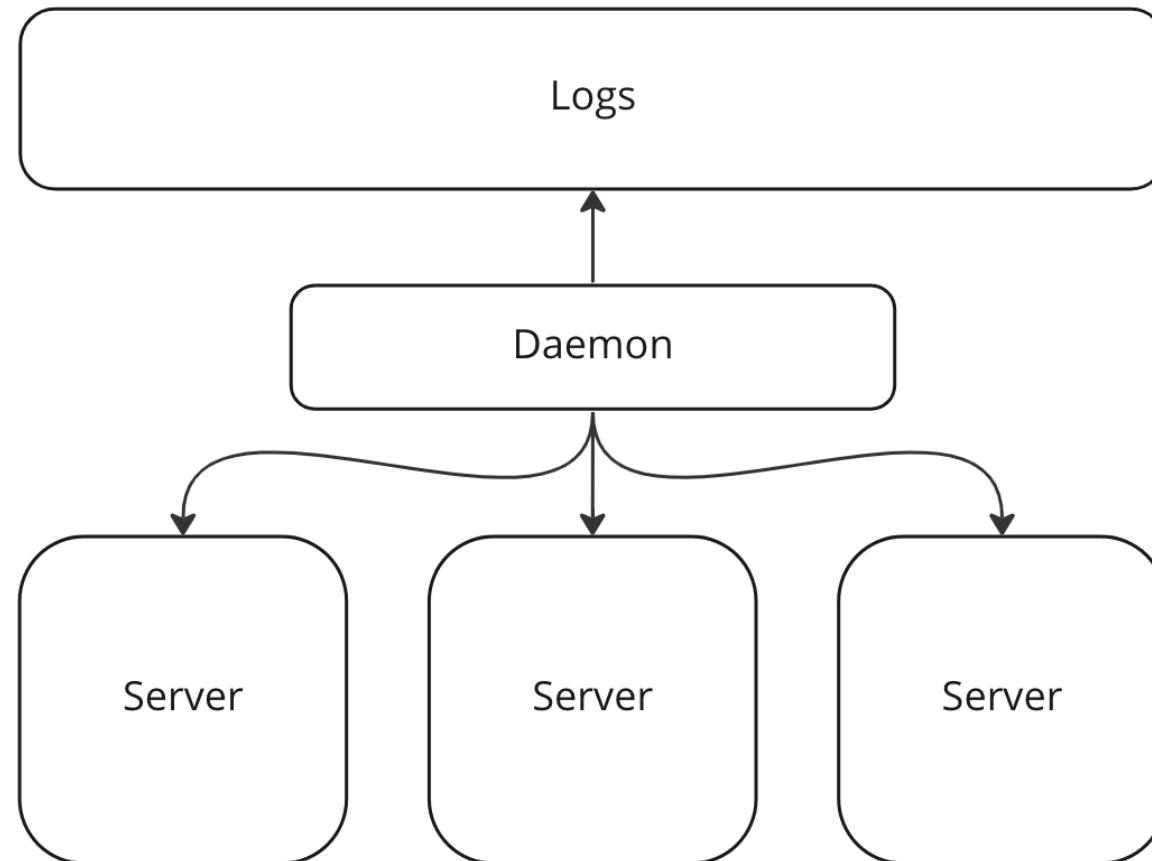
Логирование



Логирование



Логирование



Логирование



elasticsearch

logstash

kibana

graylog



```
1 package main
2
3 import "go.uber.org/zap"
4
5 func main() {
6     logger, _ := zap.NewProduction()
7     logger.Info("INFO log level message")
8     logger.Warn("Warn log level message")
9     logger.Error("Error log level message")
10 }
```



```
1 {"level":"info","ts":1665814987.2550712,"caller":"zapLogg
er/main.go:8","msg":"INFO log level message"}
2 {"level":"warn","ts":1665814987.2551033,"caller":"zapLogg
er/main.go:9","msg":"Warn log level message"}
3 {"level":"error","ts":1665814987.255109,"caller":"zapLogg
er/main.go:10","msg":"Error log level
message","stacktrace":"main.main\n\t/home/GolinuxCloud/Go
Zap/zapLogger/main.go:10\nruntime.main\n\t/usr/local/go/s
rc/runtime/proc.go:250"}
```

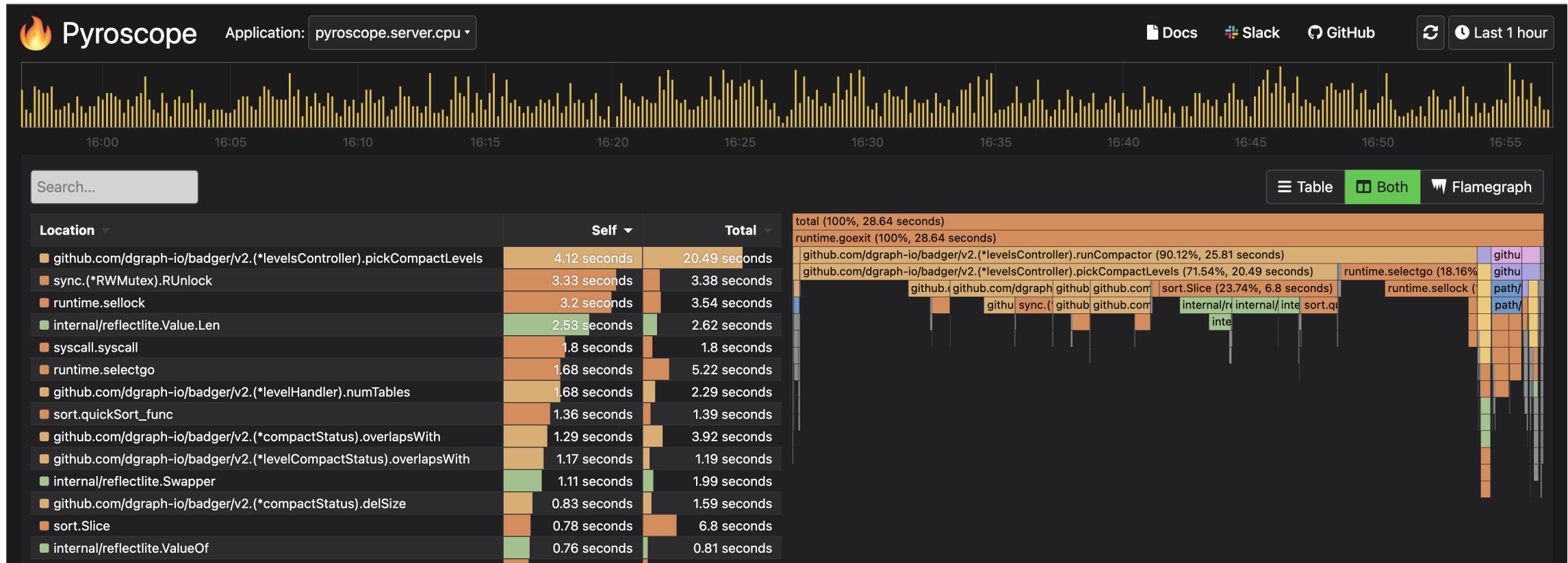
Непрерывное профилирование

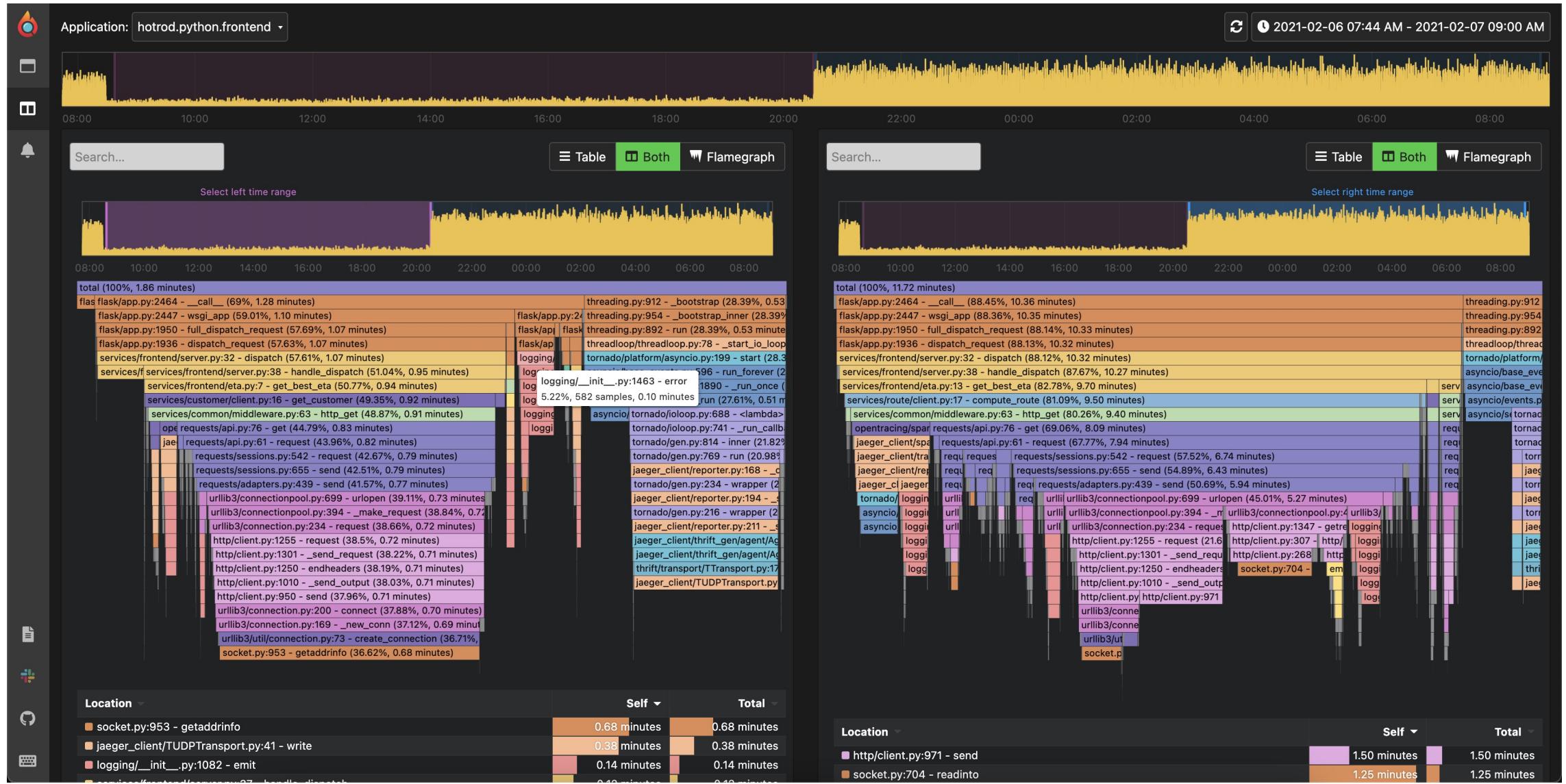
parco



Pyroscope

Pyroscope





```
1 pyroscope.Start(pyroscope.Config{  
2     ApplicationName: "simple.golang.app",  
3     // replace this with the address of pyroscope server  
4     ServerAddress:  "http://pyroscope-server:4040",  
5     // you can disable logging by setting this to nil  
6     Logger:          pyroscope.StandardLogger,  
7     // by default all profilers are enabled,  
8     // but you can select the ones you want to use:  
9     ProfileTypes: []pyroscope.ProfileType{  
10        pyroscope.ProfileCPU,  
11        pyroscope.ProfileAllocObjects,  
12        pyroscope.ProfileAllocSpace,  
13        pyroscope.ProfileInuseObjects,  
14        pyroscope.ProfileInuseSpace,  
15    },  
16 })
```

Sentry

Empower Plant Jane Schmidt

All Projects All Environments Last 14 days

IntegrityError pytest.runttest.call tests/sentry/api/endpoints/test_sentry_apps_stats

ISSUE # 5 EVENTS 0 USERS ASSIGNEE

DJANGO-32

✓ Resolve □ Ignore 📄 Share Open in Discover ⚡

Details Activity User Feedback Attachments Tags Events Merged Issues Similar Issues

Event bc74353509dn098y08hf March 24, 2021 9:10:21 PM UTC [JSON \(14.4kb\)](#)

charlotte_alameda@sentry.io ID: 387936 Chrome Version: 85.0.5353 Mac OSX Version: 10.15.1

Tags

browser Chrome 89.0.4389 browser.name Chrome duration 16675 environment prod groupId 2292909355
interval 1000 level warning organization 557498 organization.slug caffeinatedduo os.name Linux
plan am1_f plan.category metered plan.max_members 1 plan.name Developer plan.tier am1

EXCEPTION (most recent call first)

App Only Full Raw

IntegrityError

```
IntegrityError('duplicate key value violates unique constraint "sentry_project_organization_id_slug_0ac4d5ae_uniq"\nDETAIL: Key (organization_id, slug)=(1538, climbing-beetle) already exists.\n')  
SQL: INSERT INTO "sentry_project" ("slug", "name", "forced_color", "organization_id", "public", "date_added", "status", "first_event", "flags", "platform") VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s) RETURNING "sentry_project"."id"
```

sentry/db/postgres/base.py in execute at line 75

```
70.     @capture_transaction_exceptions  
71.     @auto_reconnect_cursor  
72.     @less_shitty_error_messages  
73.     def execute(self, sql, params=None):  
74.         if params is not None:  
75.             return self.cursor.execute(sql, clean_bad_params(params))
```

All Environments LAST 24 HOURS

LAST 30 DAYS

LAST SEEN 3 minutes ago in release 25c62c66dd9e

FIRST SEEN 7 months ago in release 026451250466

Linked Issues

Link GitHub Issue +
Link Jira Issue +
Link Asana Issue +
Link Azure DevOps Issue +

Tags

Tag	Value	Percentage
browser	Chrome 89.0.4389	100%
browser.name	Chrome	100%
device	Mac	100%



```
1 func() {  
2     defer sentry.Recover()  
3     // do all of the scary things here  
4 }()
```



```
1 f, err := os.Open("filename.ext")  
2 if err != nil {  
3     sentry.CaptureException(err)  
4 }
```

Домашнее задание #1

Спроектировать API для Яндекс Такси (REST / gRPC / со звездочкой GraphQL):

- рассчитать время и стоимость маршрута (со звездочкой еще и разные тарифы);
- сделать заказ такси по выбранному маршруту (со звездочкой еще и разные пути);
- посмотреть профиль водителя и клиента
- узнать статус загруженности водителей
- посмотреть историю поездок
- оставить отзыв о поездке
- изменить свои данные

Домашнее задание #2

Спроектировать базу(ы) данных для социальной сети ВКонтакте:

- анкеты людей (имя, описание, фото, город, интересы, ...)
- посты (описание, фото / видео / аудио, хэштеги, лайки, просмотры, комментарии ...)
- личные сообщения и чаты (текст, прочитанность сообщений)
- отношения (друзья, подписчики)
- сообщества (люди, посты)
- медиа (фото, аудио, видео)

Домашнее задание #3

- Пересмотреть лекцию в записи
- Погрузиться поподробнее в те темы, по которым осталось какое-то недопонимание
- Найти себе партнера в чате и договориться с ним о «проверке» друг друга
- Оставить отзыв о занятии в Google форме