Эффективные надежные микросервисы

Одноклассники

6 тыс машин

4 облака

15 тыс задач

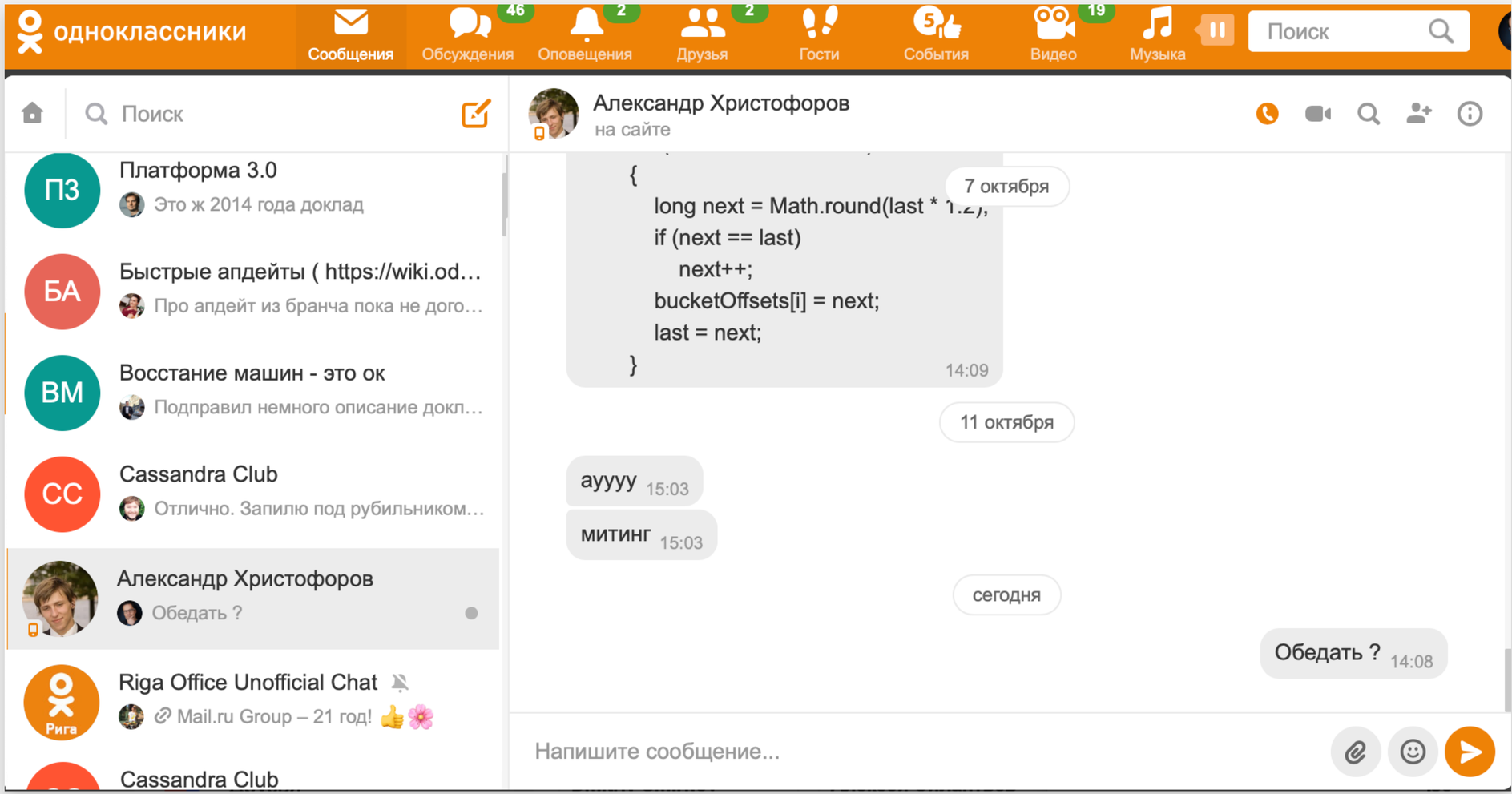
200 сервисов

Общение - это ОК

5%
активных чатов

95%
запросов

80%
последние 13 сообщений



Сообщения в чатах

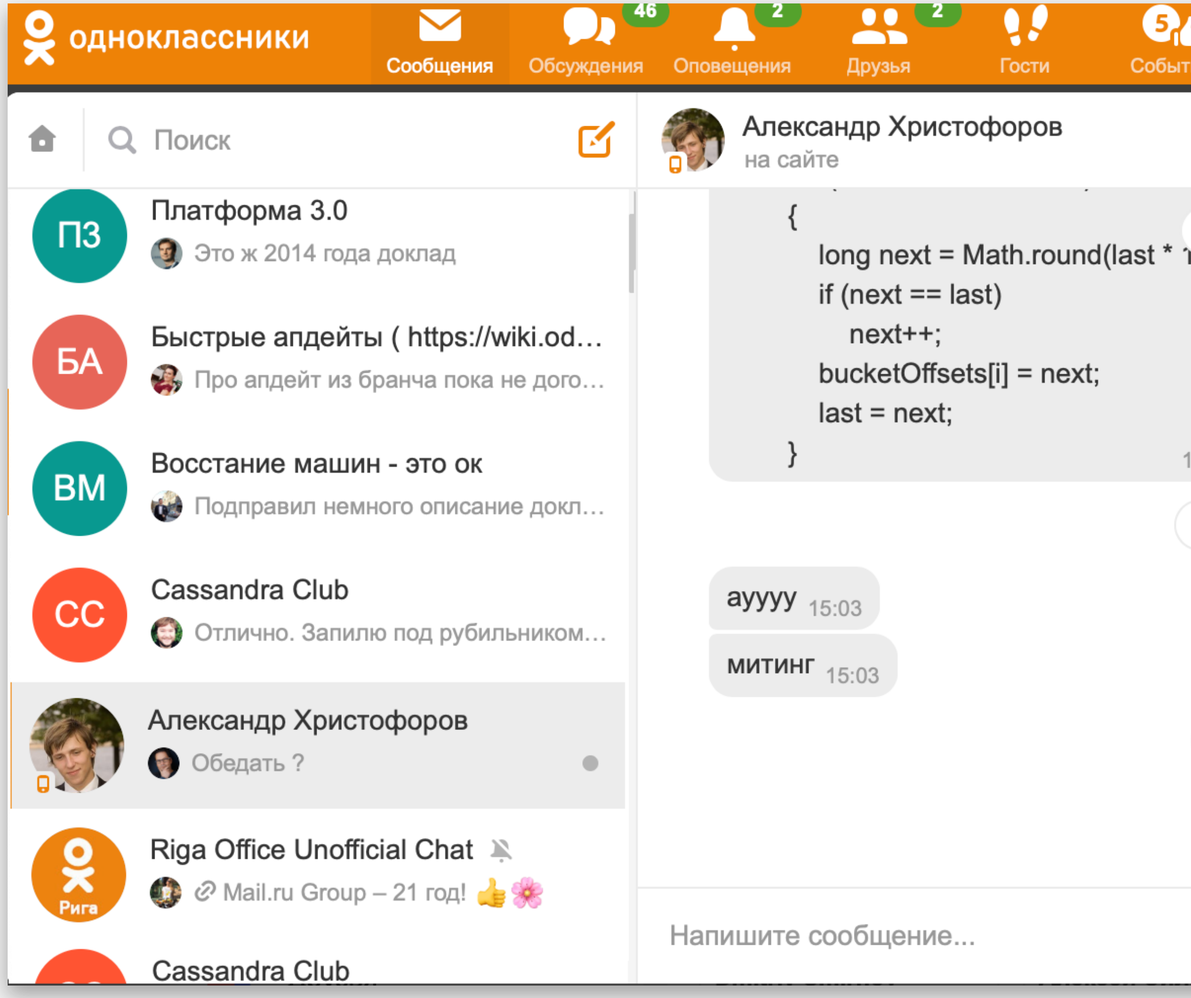
600 млрд
сообщений

5 млрд
чатов

100 тб
чистых данных

120 тыс/сек
чтений

8 тыс/сек
записей



Сообщения в чатах: поиск

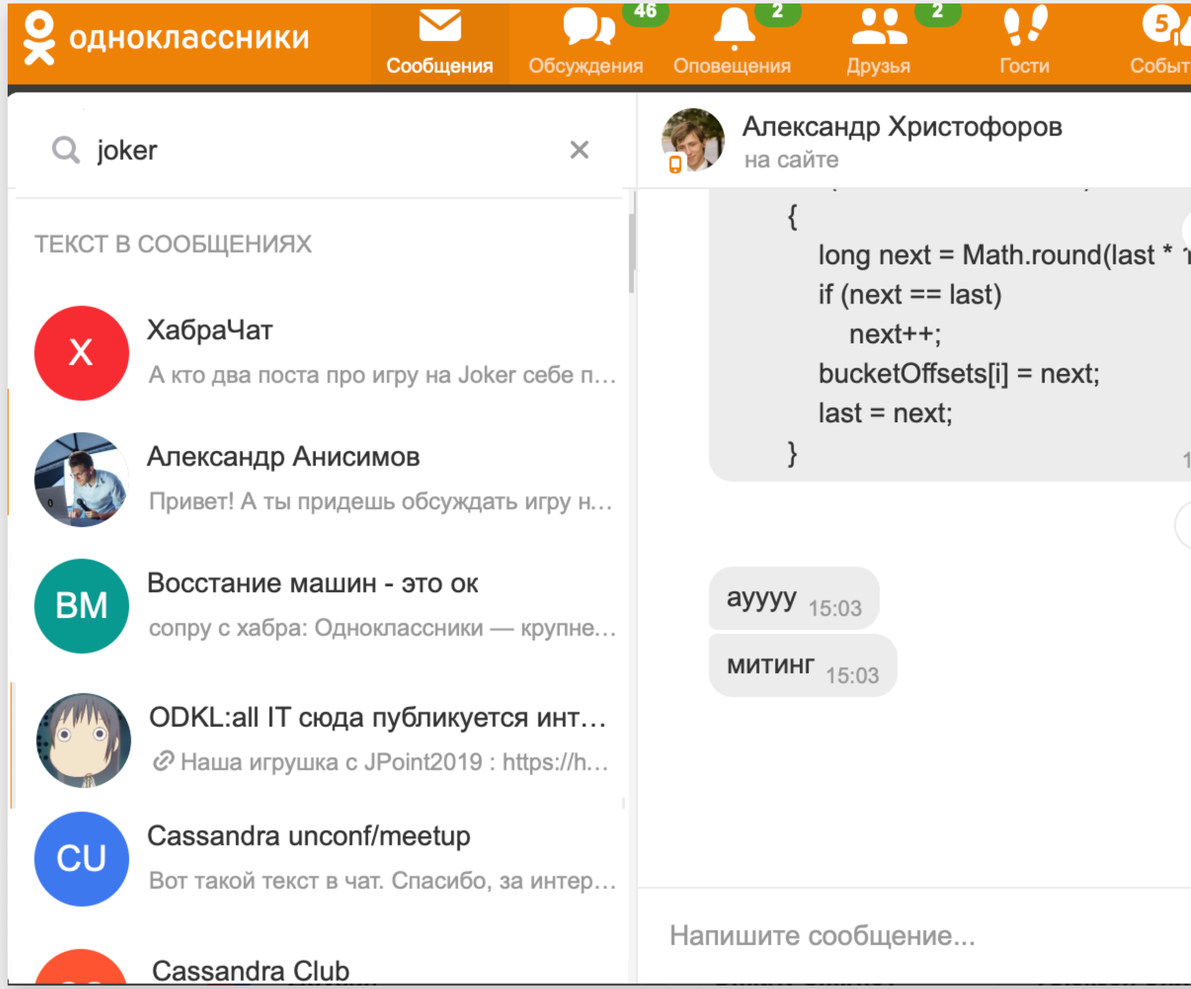
600 млрд
сообщений

5 млрд
чатов

100 тб
чистых данных

120 тыс/сек
чтений

8 тыс/сек
записей



Сервис сообщений в чатах

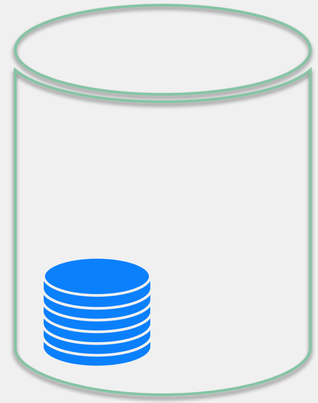
операции:

- getMessages(viewer, chat, from, to)
- getLastMessages(viewer, chats)
- add(chat, message)
- search(viewer, text)
- indexMessages()

	ИД	Автор	Тип	Текст	Аттачи[]
1	14:30	Олег	TXT	Леха, привет! Звоню ?	
1	14:31	Олег	B3	callto: Леха, miss	
1	14:32	Проктор	СПАМ	Чистите зубы нашими памперсами!	зубы.гиф
1	14:35	Леха		Ты кто ?	

```
CREATE TABLE Messages (
    chatId, msgId

    user, type, text, attachments[], terminal, deletedBy[], replyTo,...

    PRIMARY KEY ( chatId, msgId )
)
```

Современные микросервисы



DB

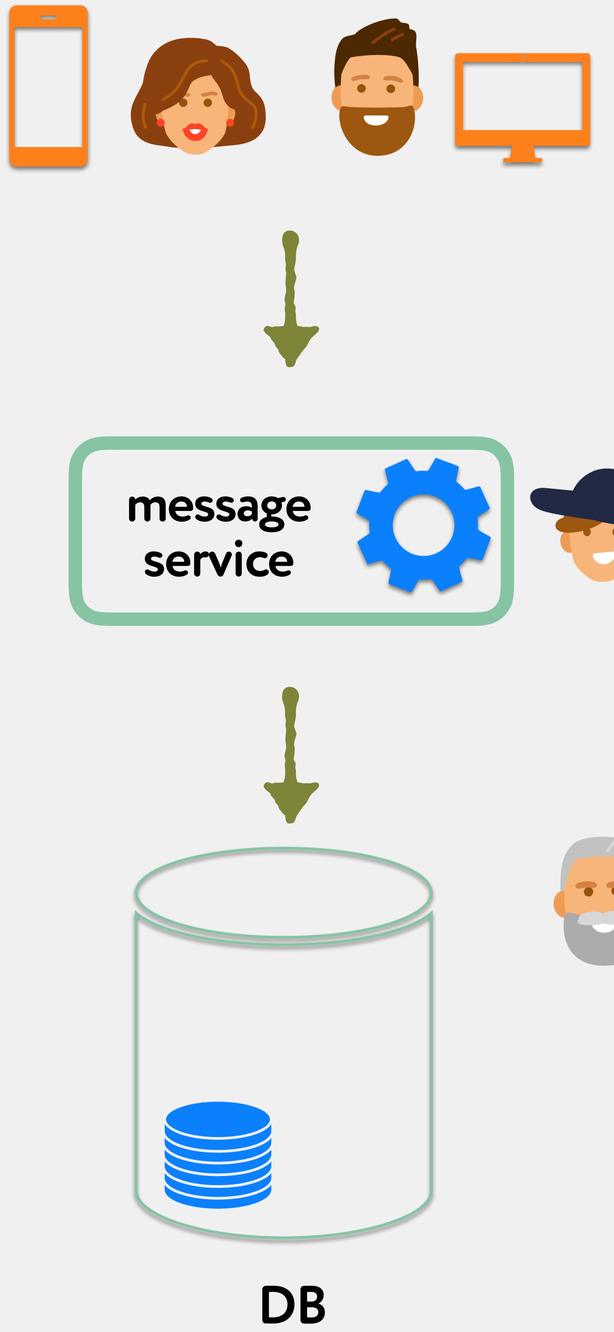
Прикладная логика

Состояние (данные)

Современные микросервисы

- getMessages(viewer, chat, from, to)

```
SELECT FROM Messages
    WHERE chatId = ? AND
    msgId BETWEEN :from AND :to
```



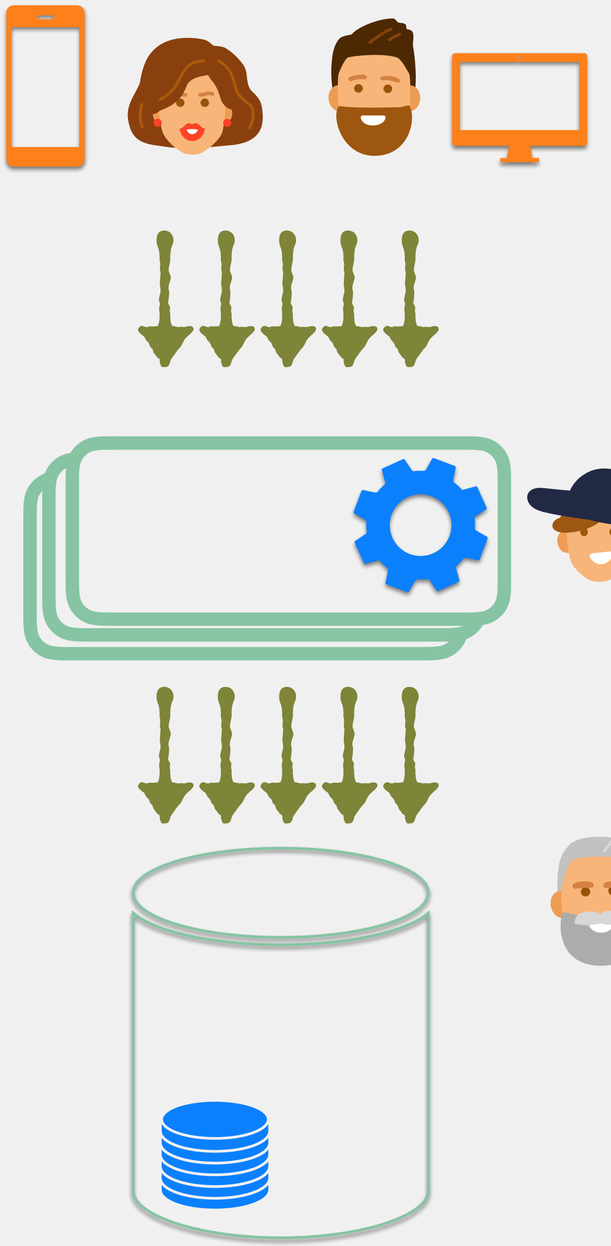
Современные микросервисы

- getMessages(viewer, chat, from, to)
- getLastMessages(viewer, chats)
- indexMessages()

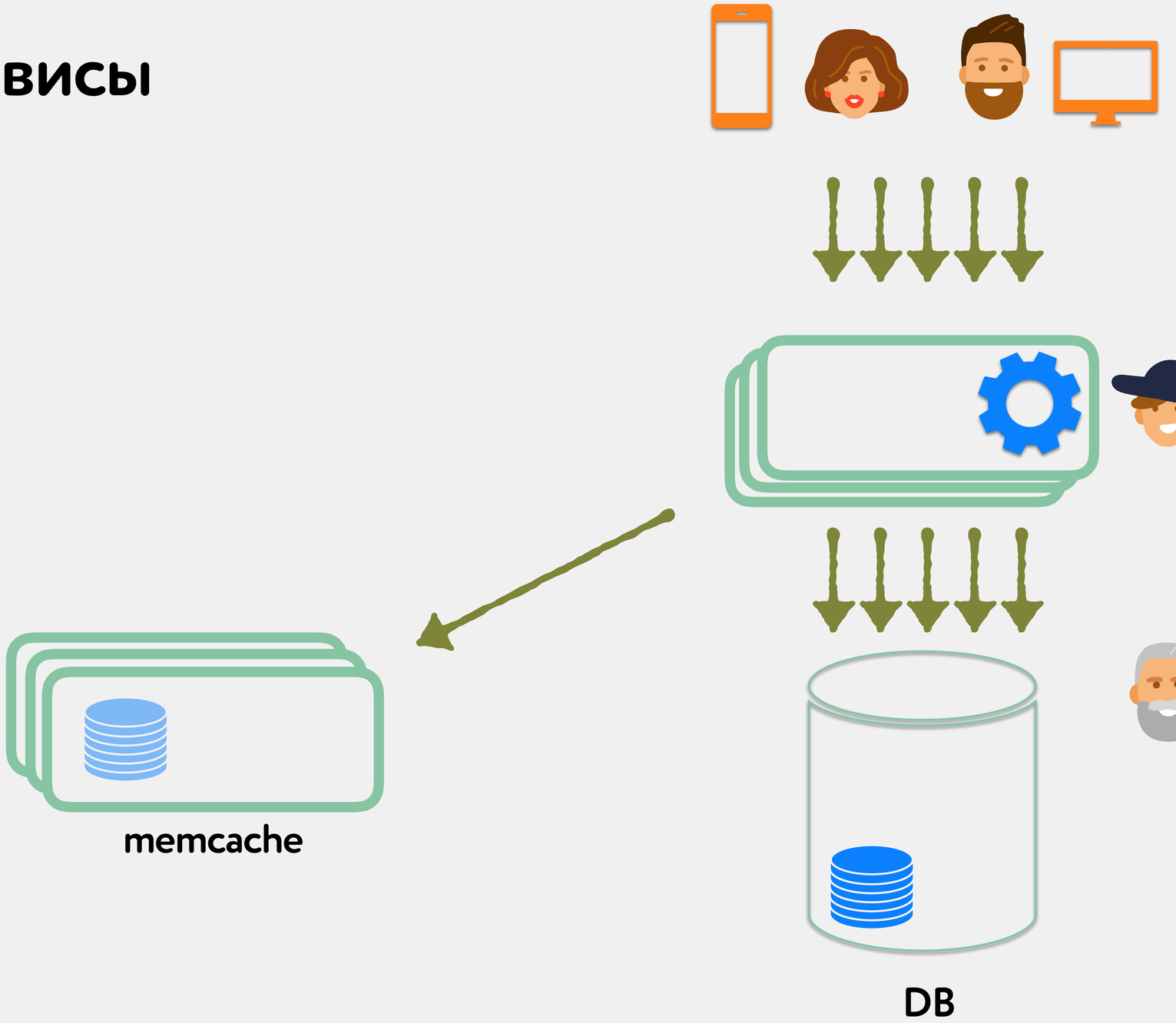
100+ k/sec

5% активных чатов

95% запросов



Современные микросервисы



Микросервисы: потери

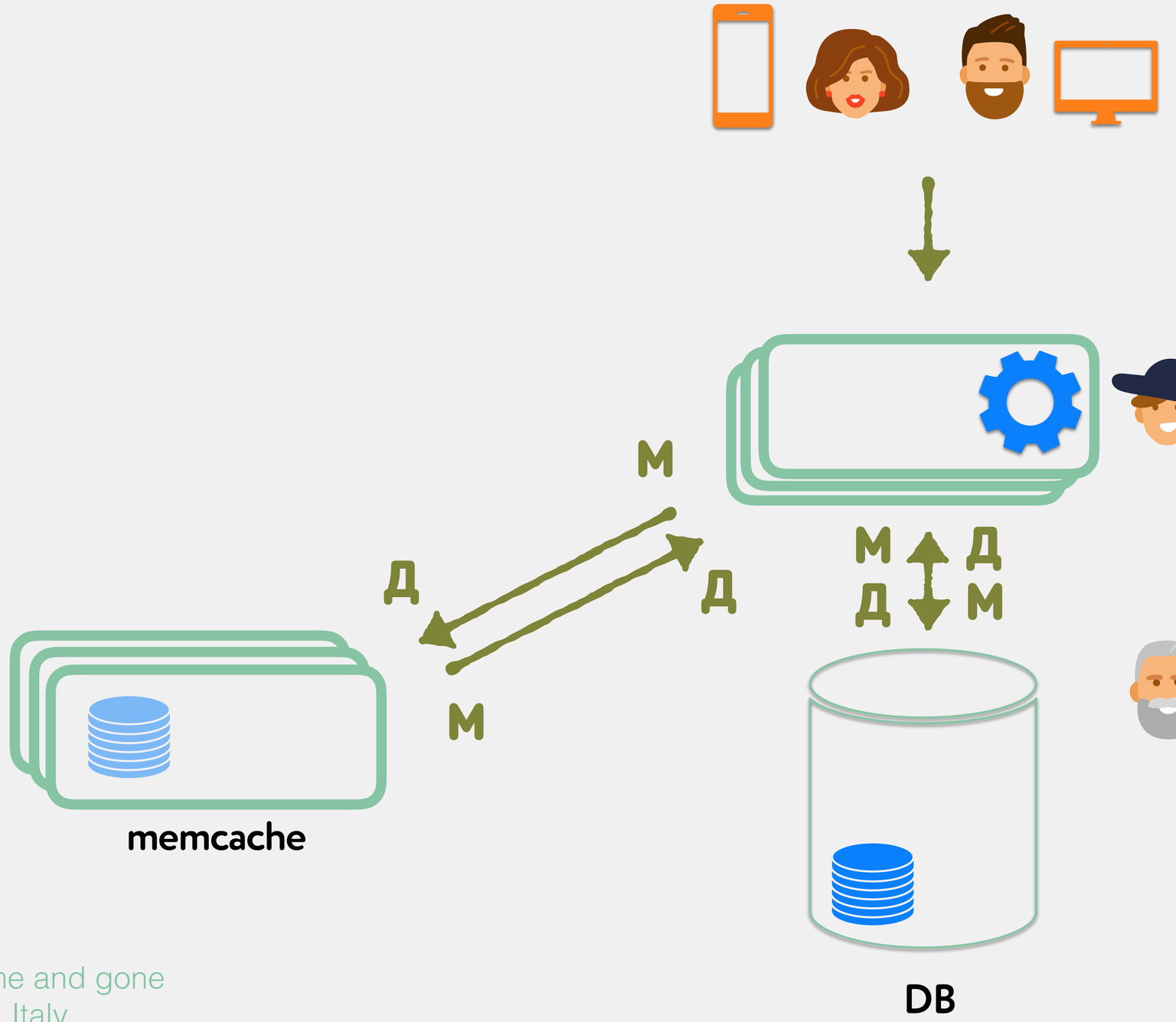
- CPU: (Де) Маршалинг

+85%

к нагрузке на ЦПУ(1)

+27%

к медианной задержке(1)



(1) Fast key-value stores: An idea whose time has come and gone
Adya et al. HotOS '19, May 13–15, 2019, Bertinoro, Italy

Микросервисы: потери

- **CPU: (Де) Маршалинг**
- **Чрезмерные чтения/запись**

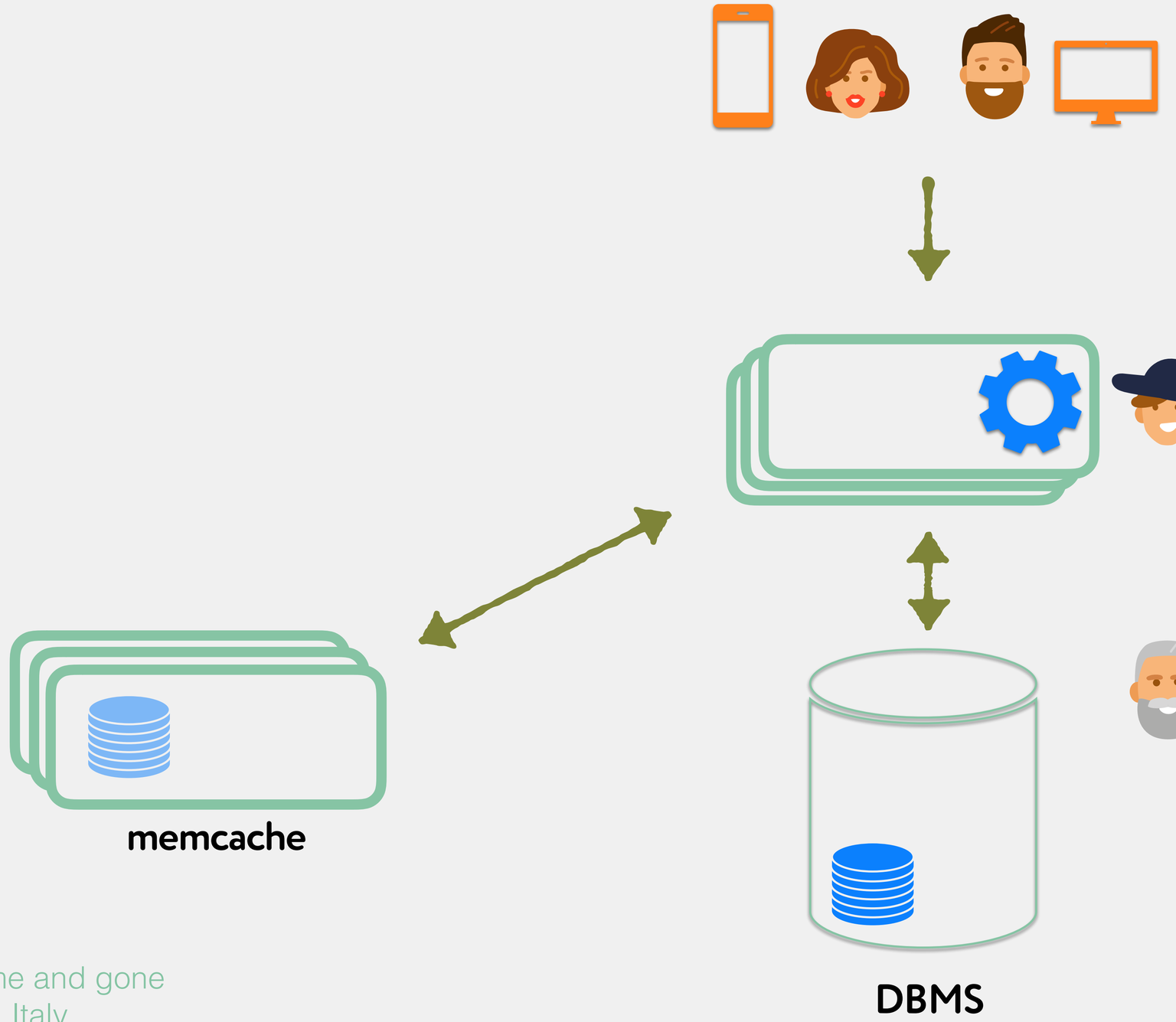
если запись содержит только

10%

полезной информации, то

+46% CPU **+86%** Net

потрачено зря [1)]



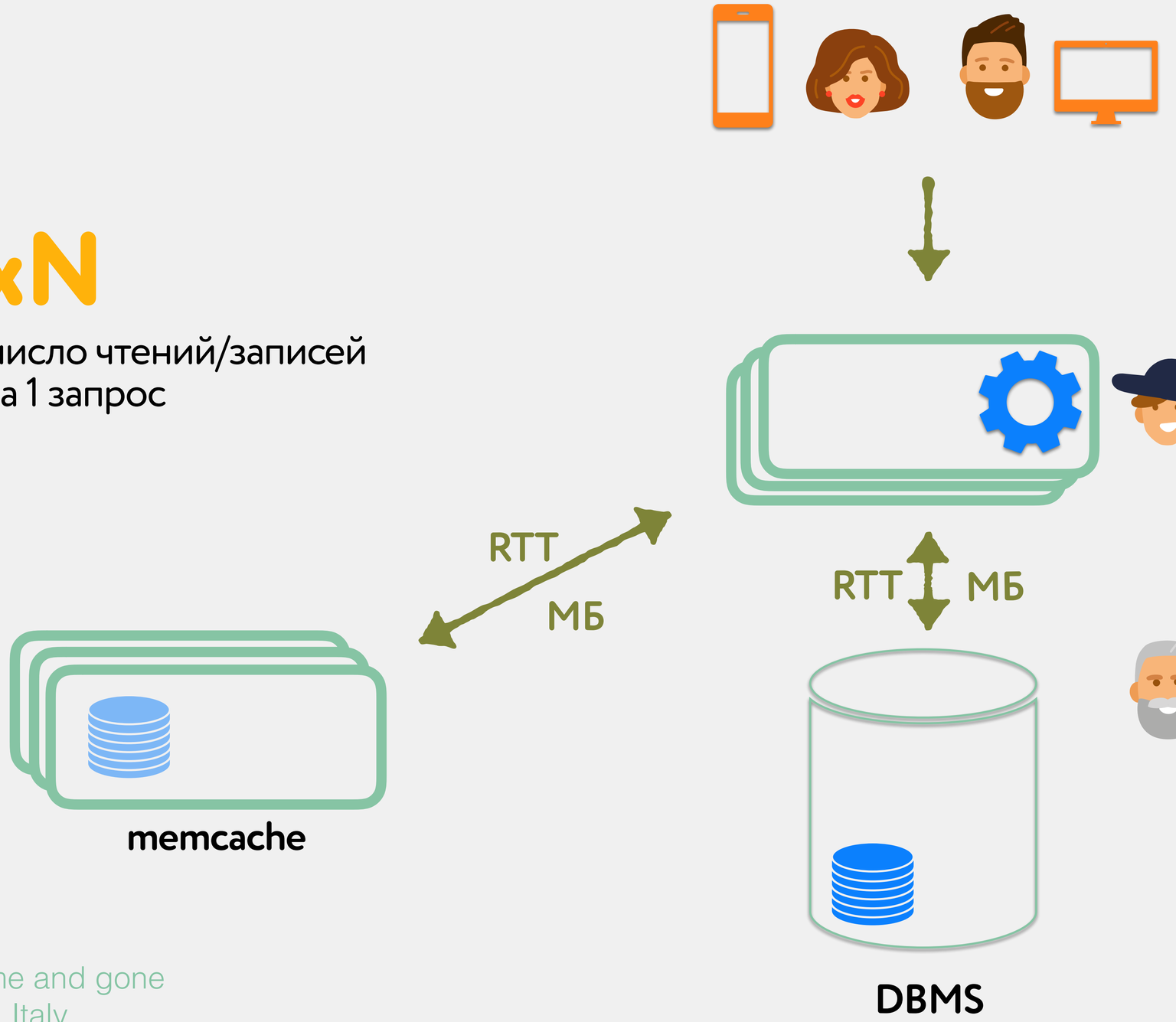
(1) Fast key-value stores: An idea whose time has come and gone
Adya et al. HotOS '19, May 13–15, 2019, Bertinoro, Italy

Микросервисы: потери

- CPU: (Де) Маршалинг
- Чрезмерные чтения/запись
- Сетевые задержки, трафик

xN

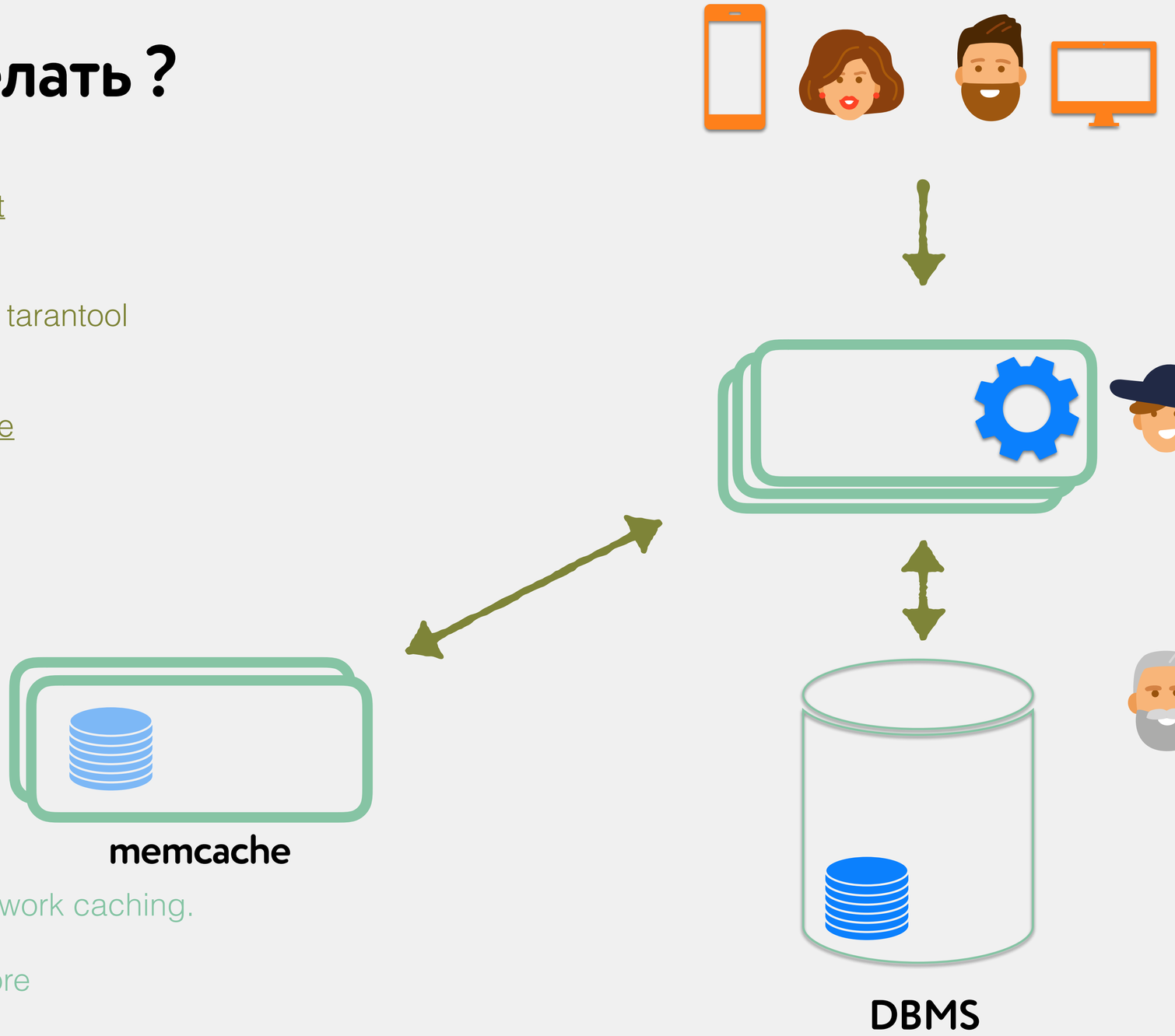
число чтений/записей
на 1 запрос



(1) Fast key-value stores: An idea whose time has come and gone
Adya et al. HotOS '19, May 13–15, 2019, Bertinoro, Italy

Микросервисы: чтожеделать ?

- **CPU: (Де) Маршалинг** (5) KV-Direct
- **Чрезмерные чтения/запись** (2) redis, (3) tarantool
- **Сетевые задержки, трафик** (4) NetCache

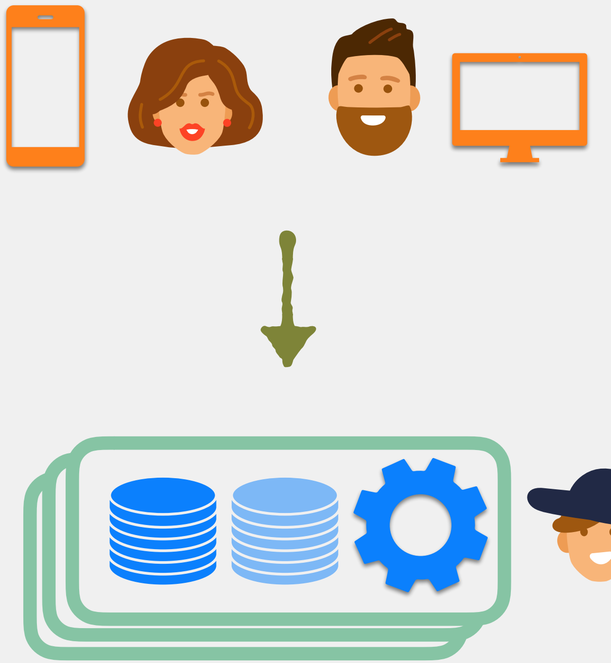


(2) http://redis.io
(3) https://tarantool.io
(4) Netcache: Balancing key-value stores with fast in-network caching.
In X. Jin et al, Stoica. SOSP, 2017.
(5) Kv-direct: high-performance in-memory key-value store
with programmable nic.
B. Li et al, In SOSP, 2017.

Микросервис с состоянием

Микросервис с состоянием

- ✓ CPU: (Де)Маршалинг
- ✓ Чрезмерные чтения/запись
- ✓ Сетевые задержки, трафик

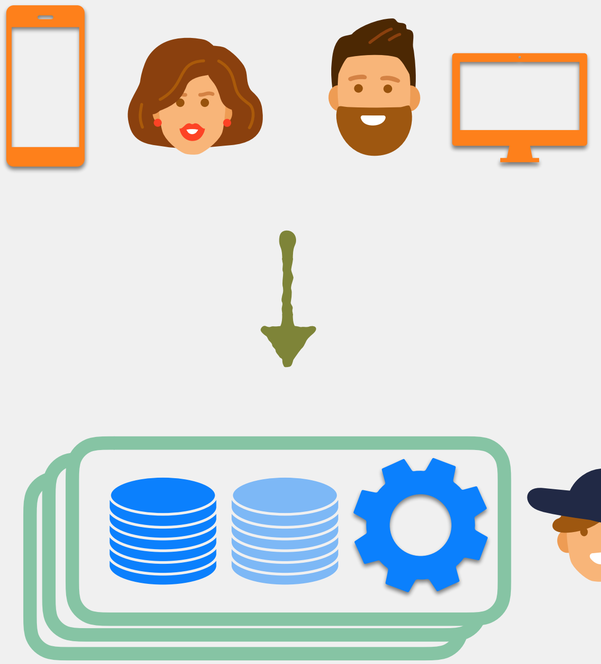


Прикладная логика

Встроенный прикладной кеш
custom in-memory store

Встроенная распределенная БД
embedded distributed store

Микросервис с состоянием



всегда быстрее !

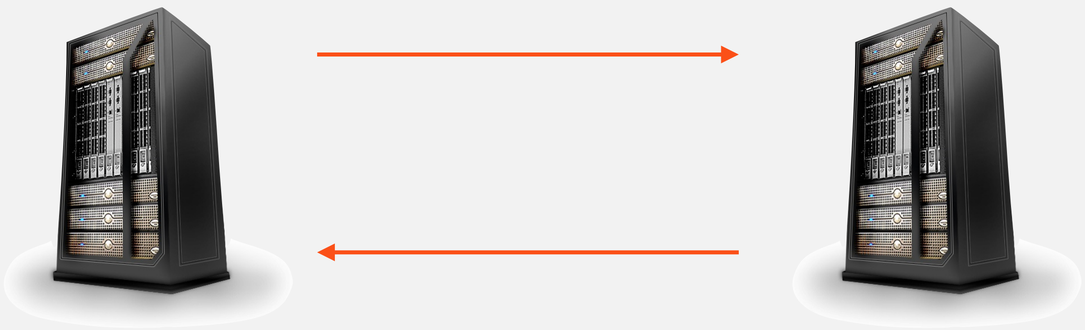
а что с отказоустойчивостью ?

Прикладная логика

Встроенный прикладной кеш
custom in-memory store

Встроенная распределенная БД
embedded distributed store

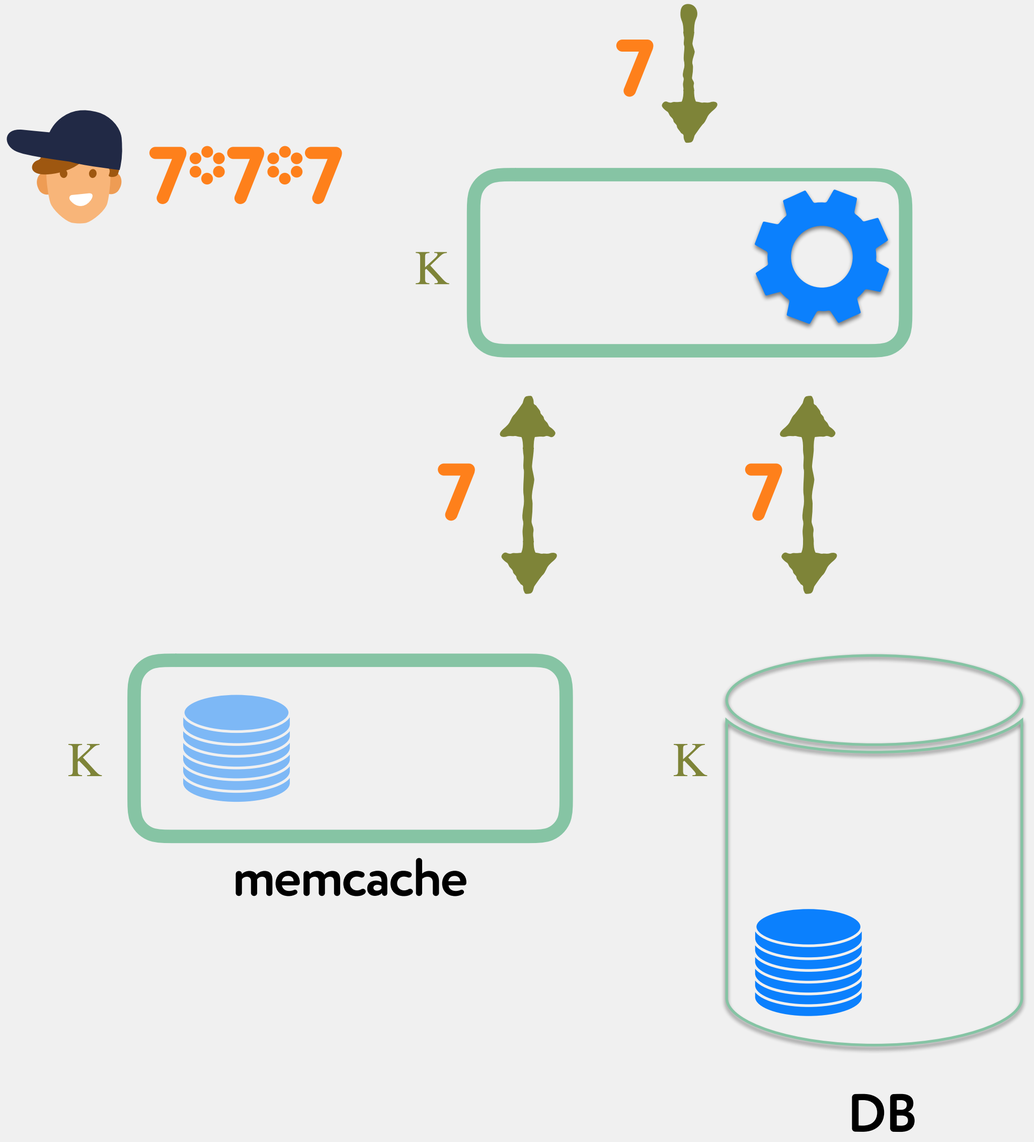
Что может пойти не так ?

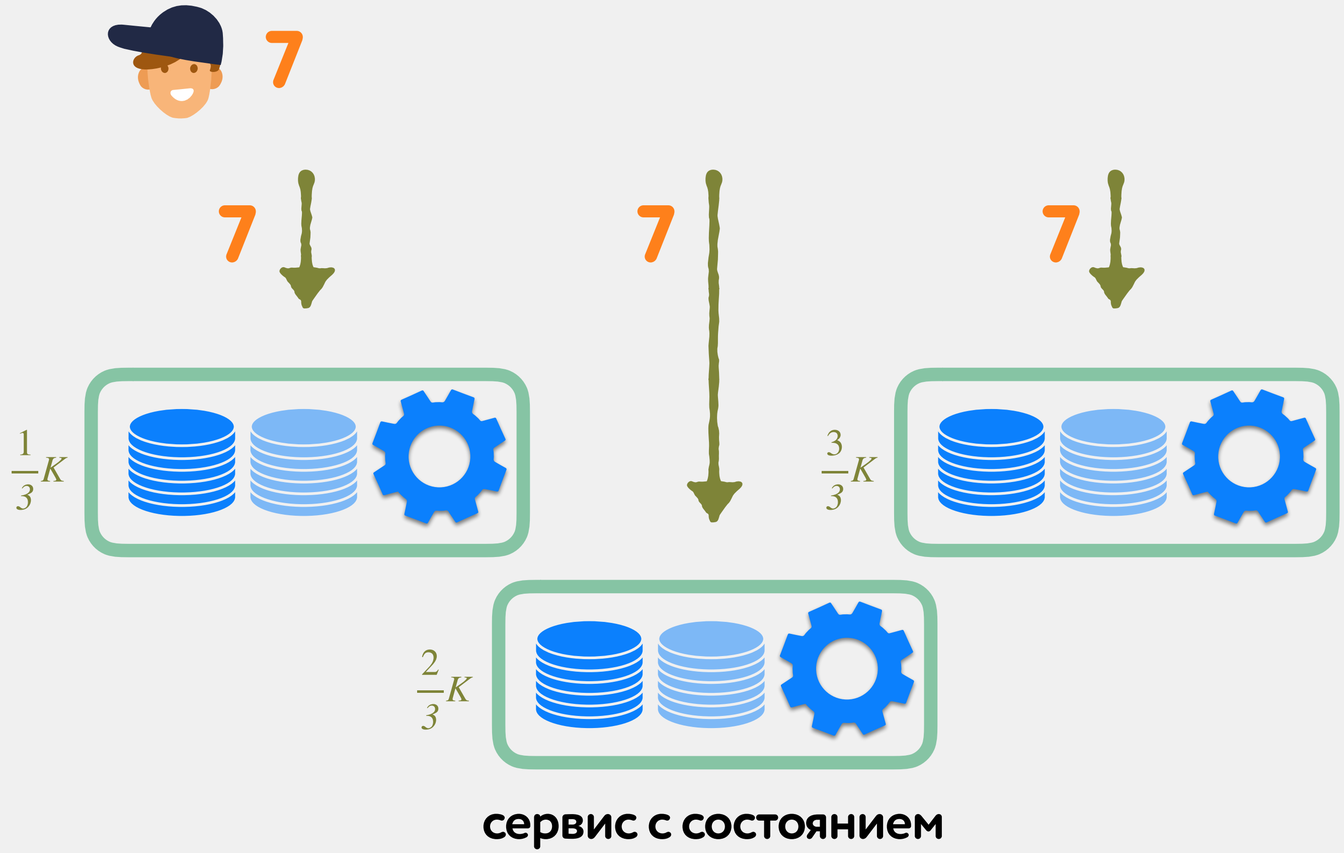


1. Пропажа клиента
2. Пропажа сервера
3. Потеря исходящего сообщения
4. Потеря входящего сообщения
5. Таймаут сервера
6. Неправильный ответ
7. Произвольный отказ

Распределенные системы в Одноклассниках
Олег Анастасьев, Joker 2015

Сценарии отказов

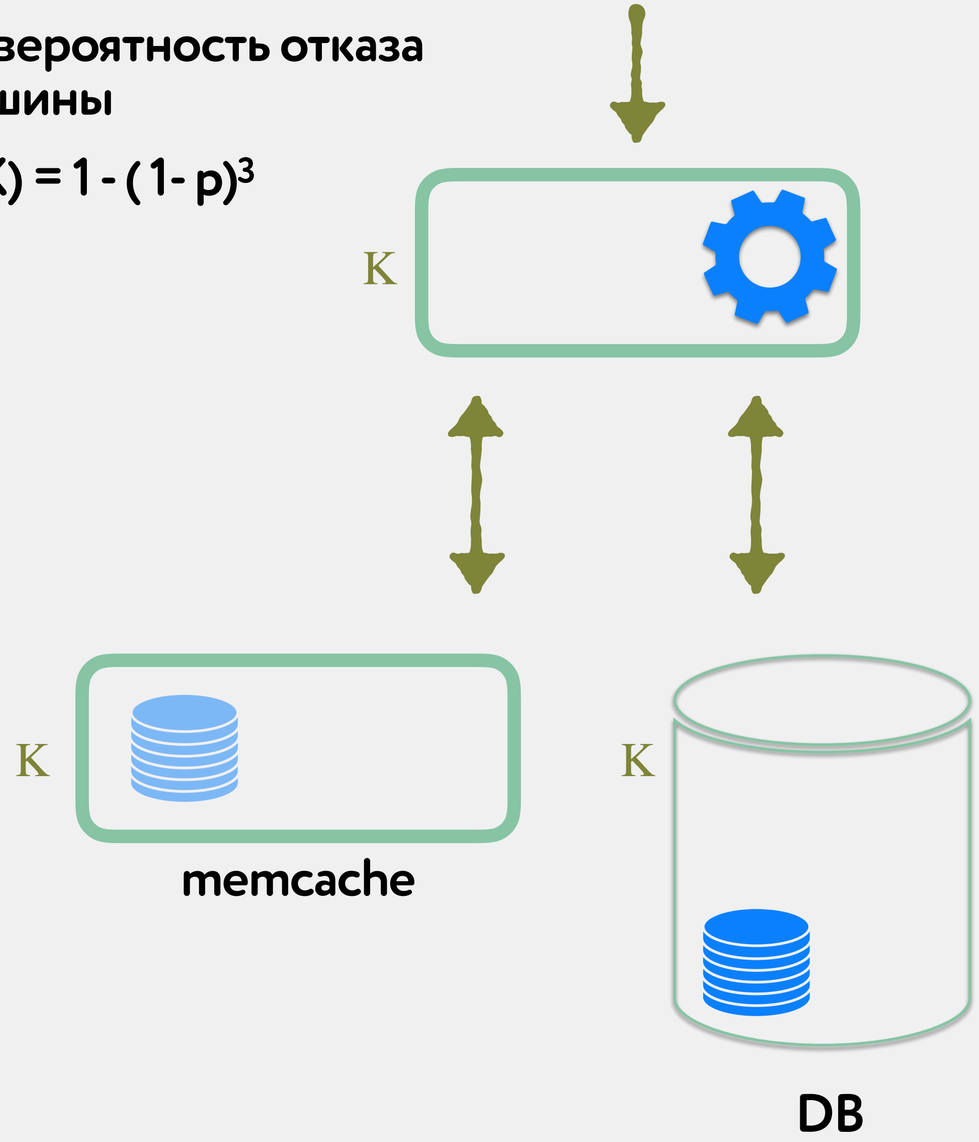




Вероятности отказов

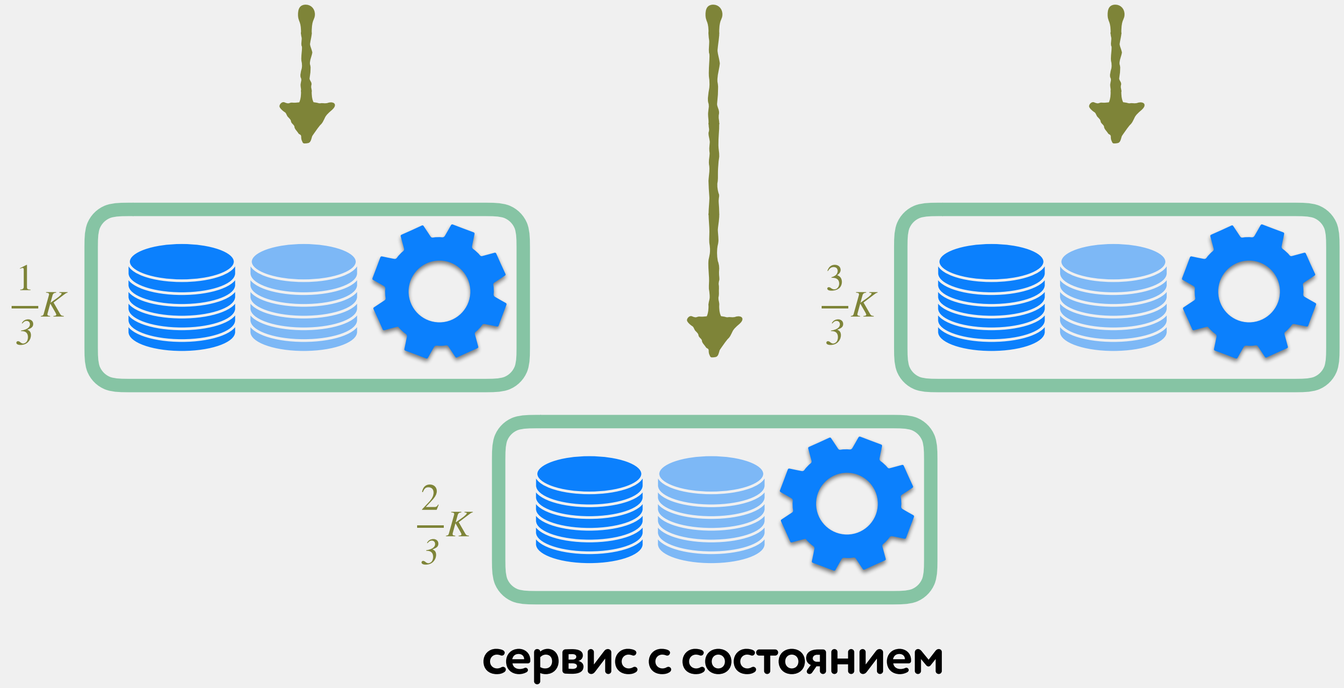
p - вероятность отказа машины

$P(K) = 1 - (1 - p)^3$



$P(K) = p^3$

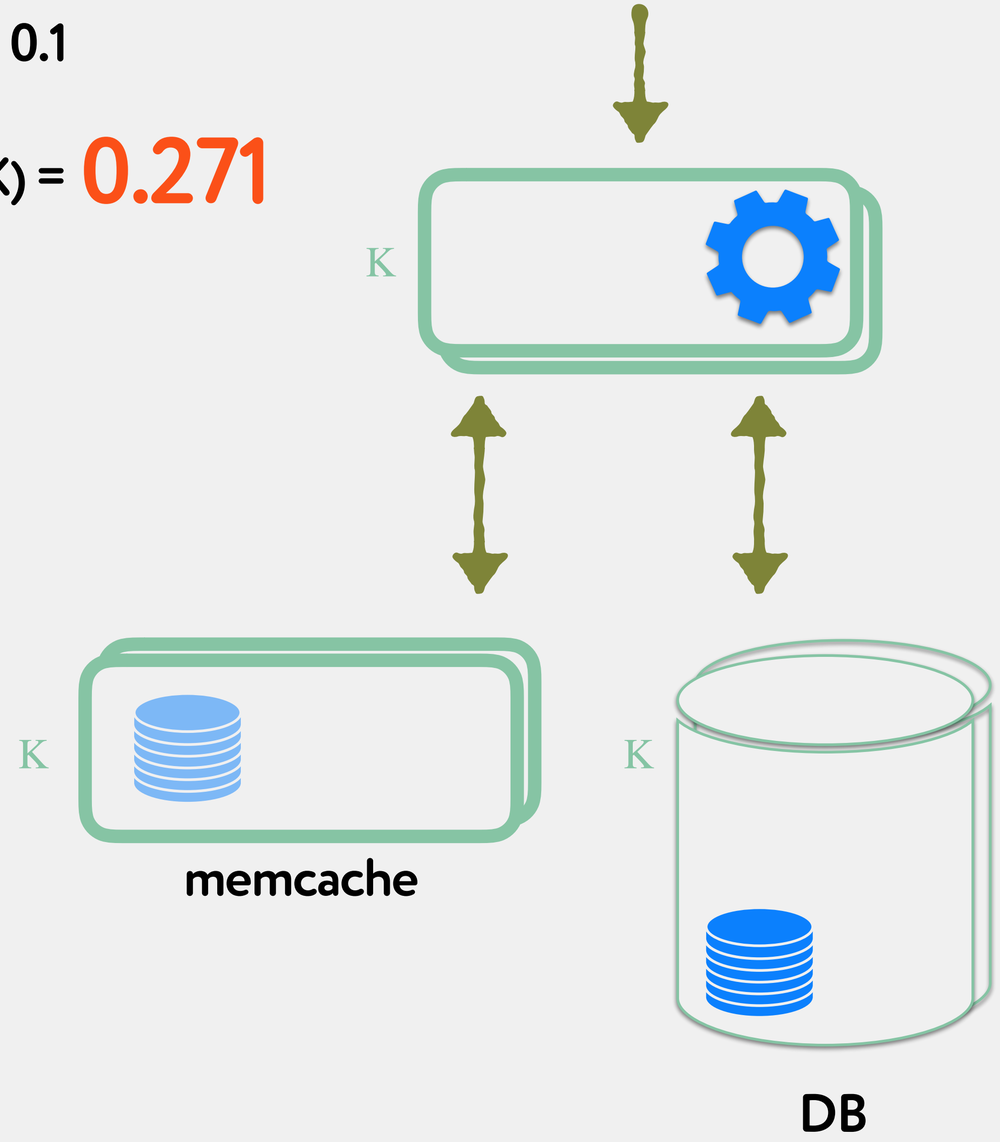
$P(⅓\ K) = 1 - (1 - p)^3$



Вероятности отказов

p = 0.1

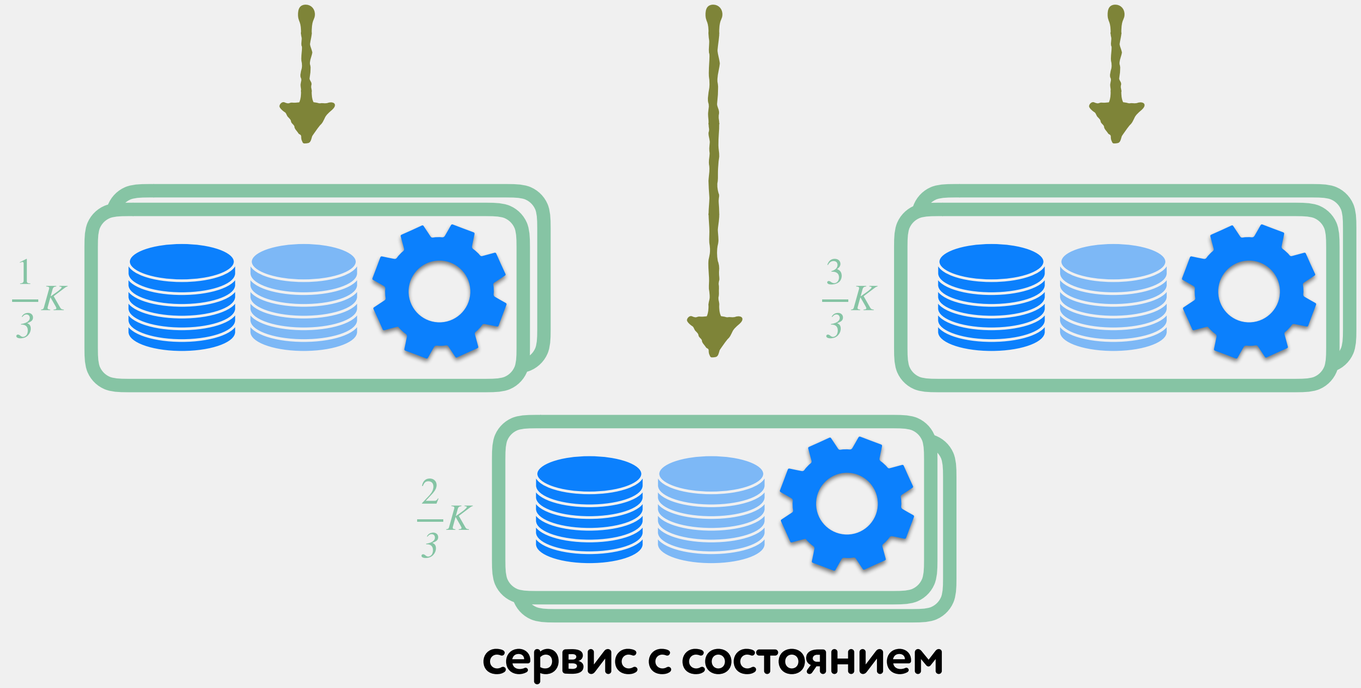
P(K) = 0.271



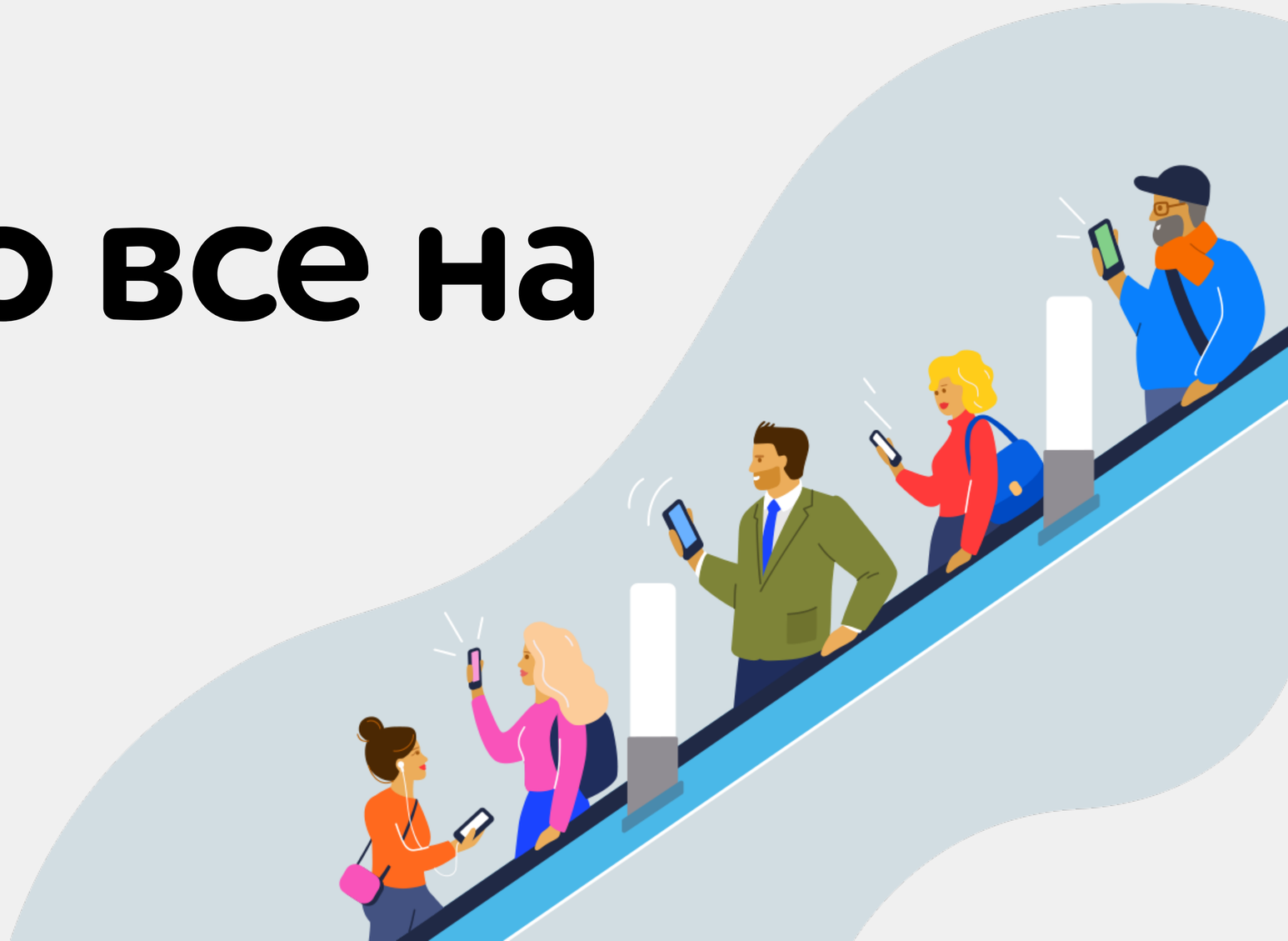
P(K) = 0.001

P(⅓ K) = 0.271

Всегда надежнее!



Хорошо все на бумаге



Встраиваем БД в сервис

должно быть:

- **Высокодоступным**
  Репликация, консистентность
- **Масштабируемым**
  Решардинг
- **На языке приложения**
  Минимизация (де)маршалинга,
  Интеграция с приложением
- **Open Source**
  для доработок

Встраиваем БД в сервис

1. -cp cassandra/lib/*.jar

```
package org.apache.cassandra.service;

public class CassandraDaemon
{
```

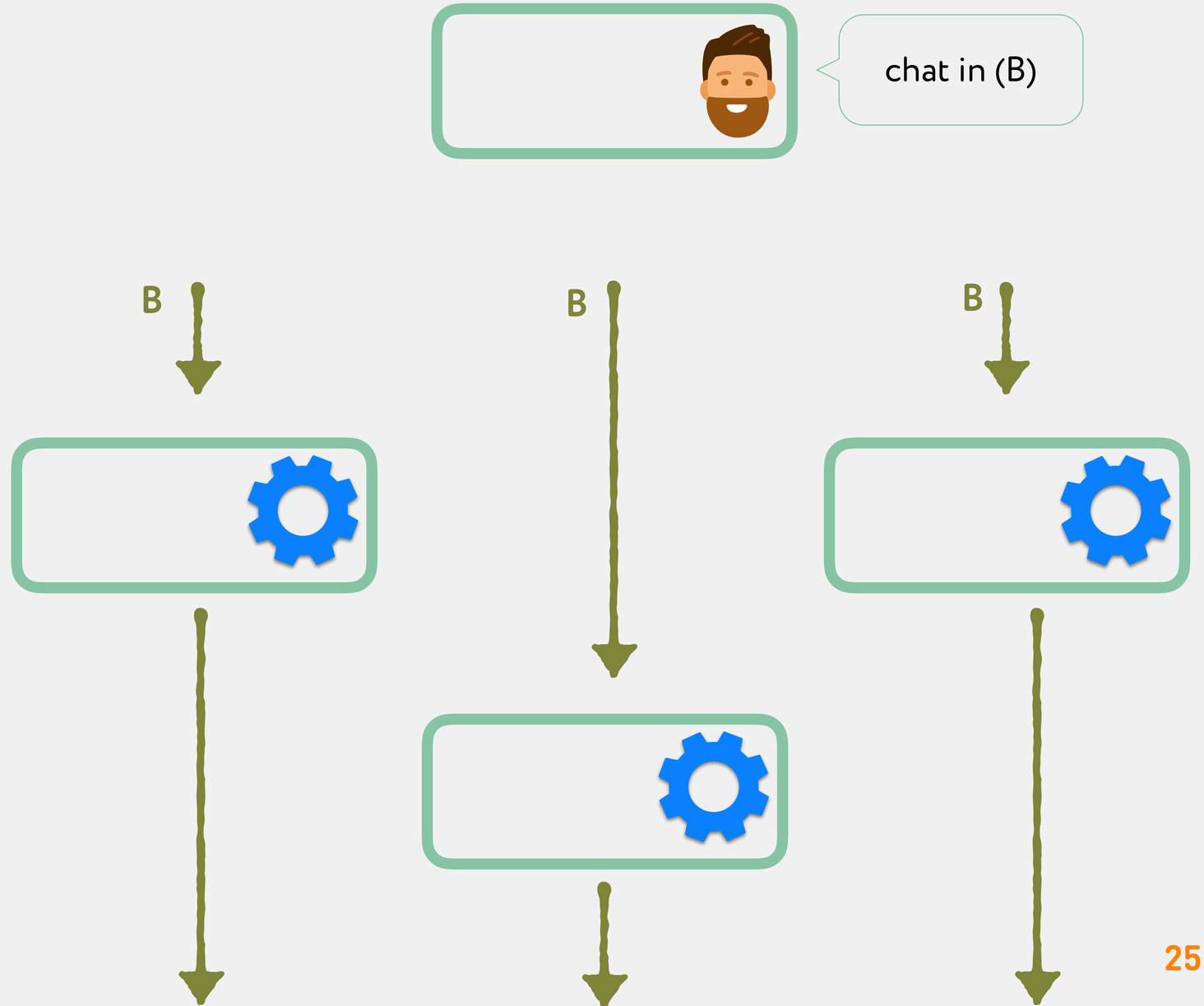
2.

```
System.setProperty( "cassandra.config", "file://whatever/cassandra.yaml" );

CassandraDaemon.instance.activate();
```

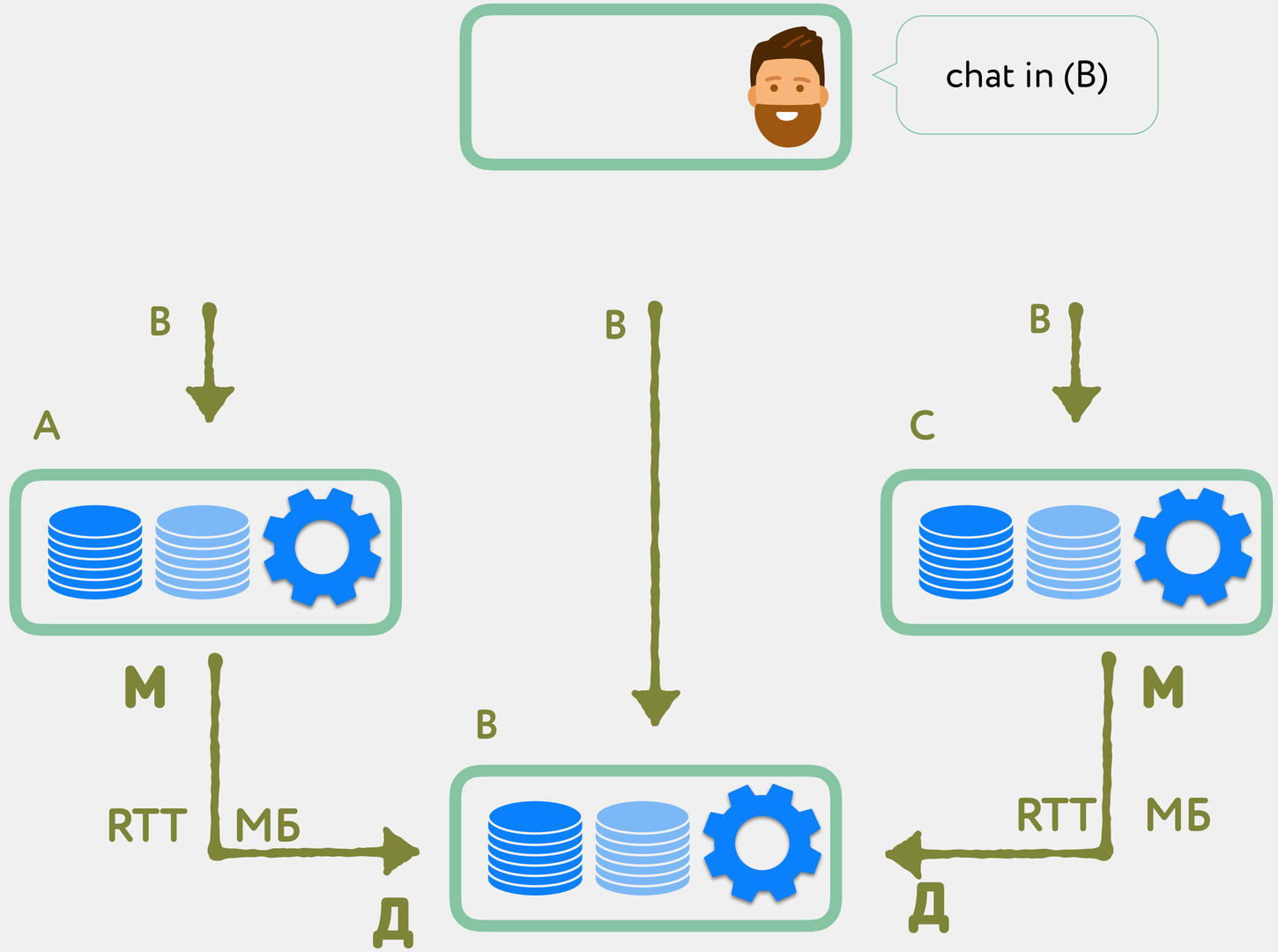
3.

```
public interface ConfigurationLoader
{
    Config loadConfig() ;
}
```

Маршрутизация запросов



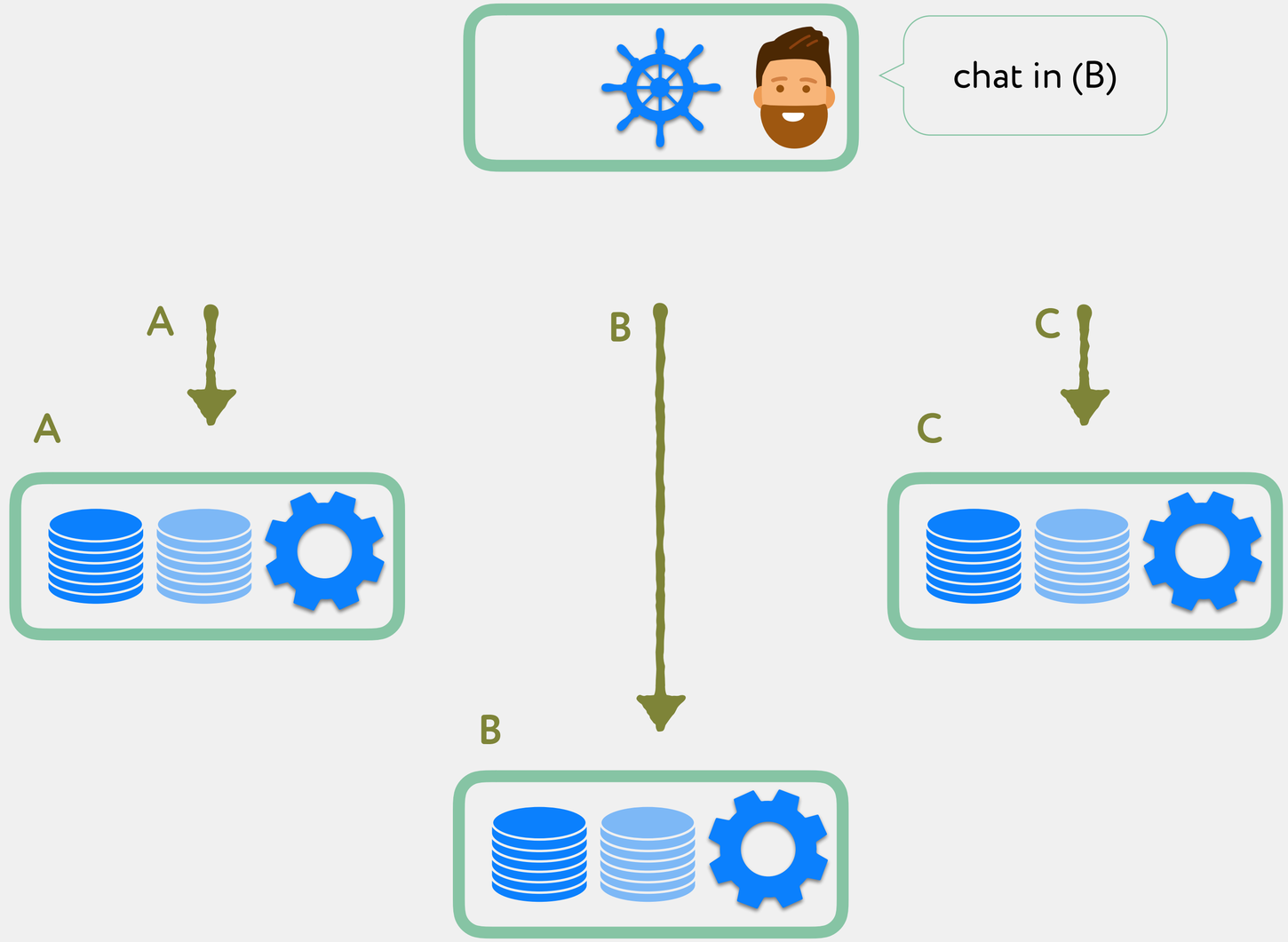
Маршрутизация запросов



Маршрутизация запросов

- **Partition-aware client routing library**

  Обеспечивает вызов реплики владеющей данными по ключу на основании информации о кластере



Распределение данных

- **Partition Key (chatId)**

  Определяет положение записи на ноде

- **Clustering Key (msgId)**

  Сортирует записи внутри партиции

```
CREATE TABLE Messages (
    chatId, msgId

    user, type, text, attachments[], terminal, deletedBy[], replyT

    PRIMARY KEY ( chatId, msgId )
)
```

Распределение данных

- **Partitioner**

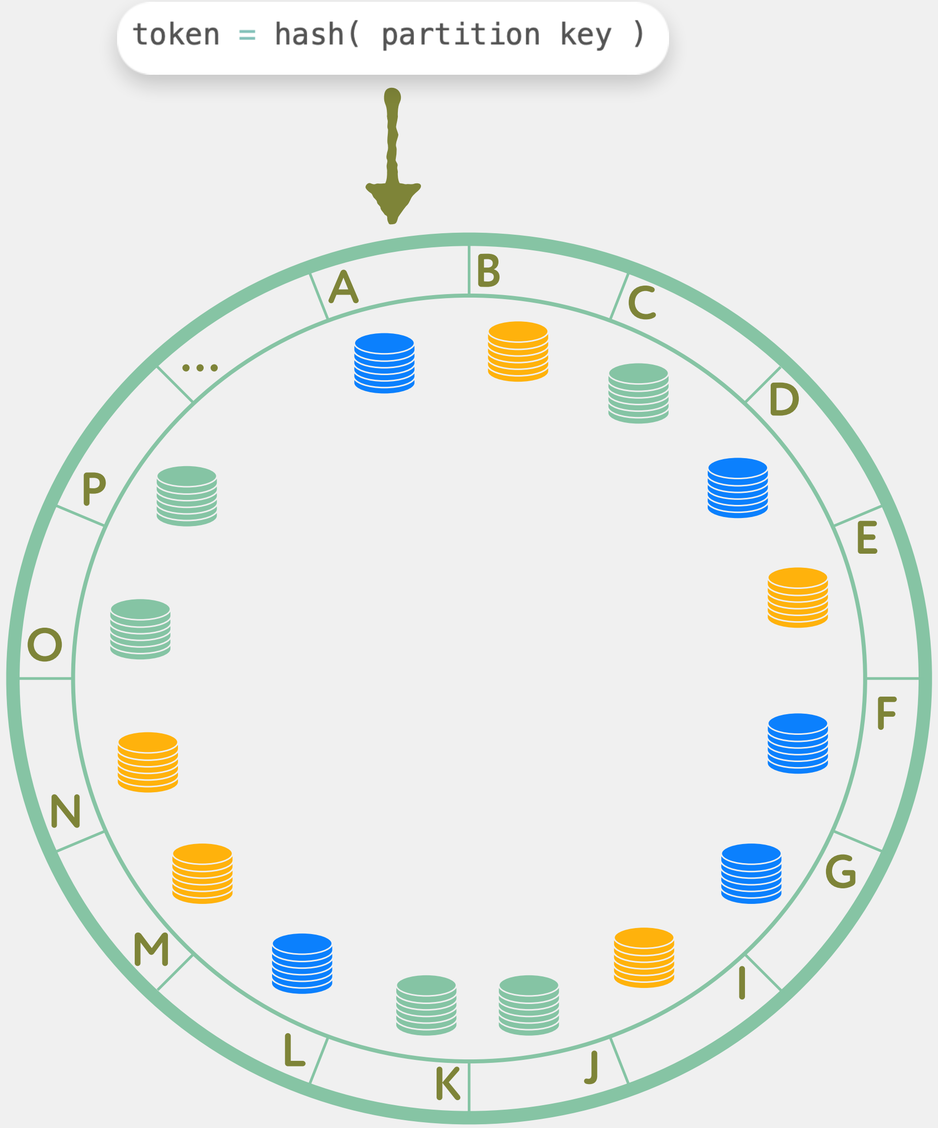
  Вычисляет токен, положение на кольце

- **TokenMetadata**

  Сопоставляет первичный интервал нодам кластера

- **Replication Strategy**

  Определяет распределение реплик данных



Распределение данных

- **Partitioner**

  Вычисляет токен, положение в кольце

- **TokenMetadata**

  Сопоставляет первичный интервал нодам кластера

- **Replication Strategy**

  Определяет распределение реплик данных

```
SortedMap<Token, List<InetAddress>> endpointMap = ...

AbstractReplicationStrategy replication = ...

for ( Token token : tokenMetadata.sortedTokens() ) {
    endpointMap.put( token,  replication.getNaturalEndpoints( token ) );
}
```

+ **Изменение топологии**

  Обновление,

  устаревшая информация

Мессенджер: работаем с БД

- getMessages(viewer, chat, from, to)
- add(chat, message)

```
CREATE TABLE Messages (
    chatId, msgId

    user, type, text, attachments[], terminal, delete

    PRIMARY KEY ( chatId, msgId )
)
```

```
package org.apache.cassandra.cql3;

import java.nio.ByteBuffer;

public class QueryProcessor
{
    public static UntypedResultSet execute(String query,
                                           ConsistencyLevel cl, Object... values)
                                           throws RequestExecutionException
```

```
UntypedResultSet rs = QueryProcessor.execute(
                    "SELECT * FROM Messages "
                    + "WHERE chatId = ? AND msgId < ? AND msgId > ?",
                    ConsistencyLevel.QUORUM, chatId, from, to );

rs.forEach( row -> {} );
```

(6) https://github.com/datastax/java-driver/tree/4.x/manual/core

Кеш сообщений

600 млрд
сообщений

5 млрд
чатов

100 тб

5%
активных чатов

80%
последние 13 сообщений

кеш сообщений в памяти

3+ млрд
сообщений

250 млн
чатов

500 гб

Кеш сообщений

- getMessages(viewer, chat, from, to)
- getLastMessages(viewer, chats)
- add(chat, message)

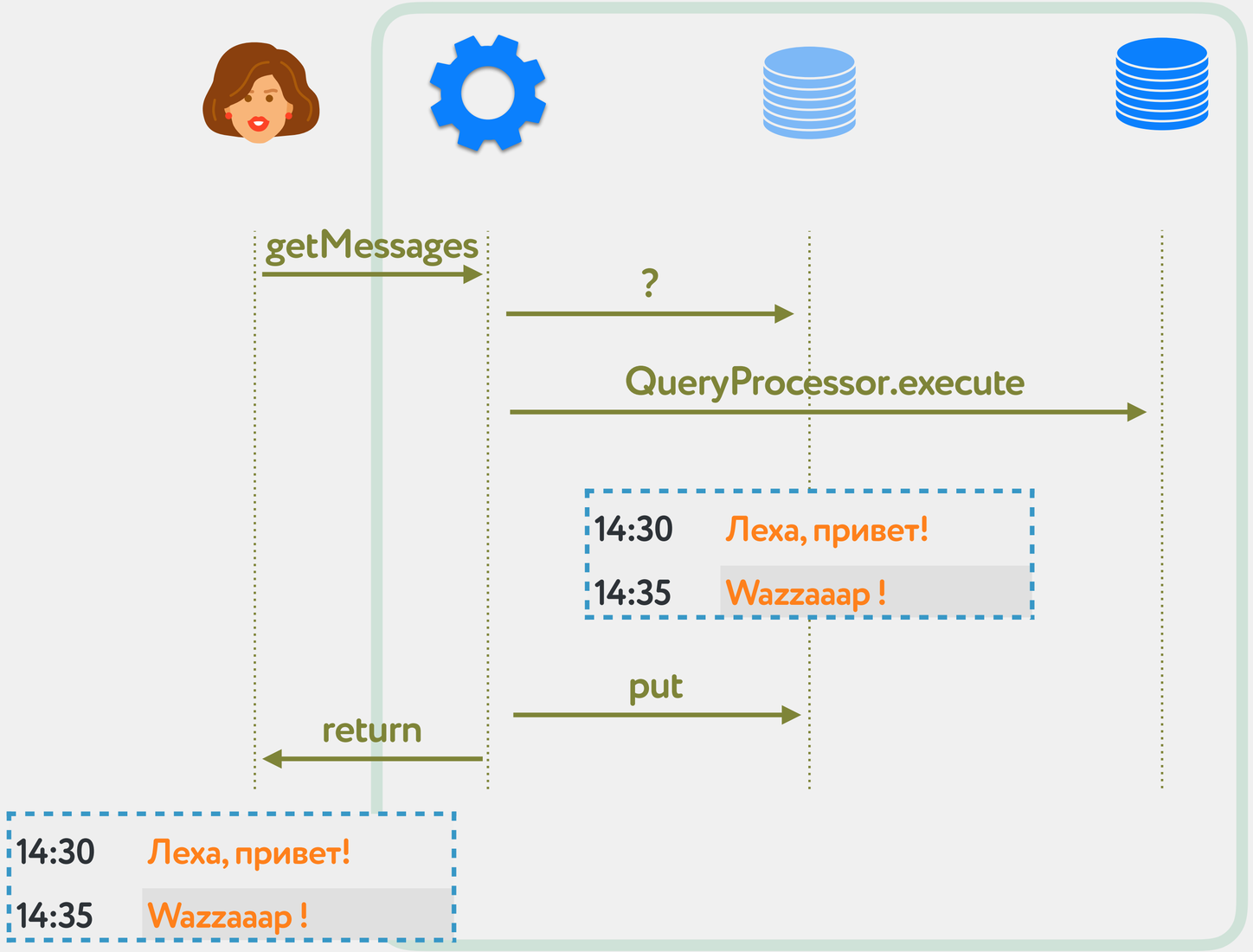
кеш сообщений в памяти

3+ млрд
сообщений

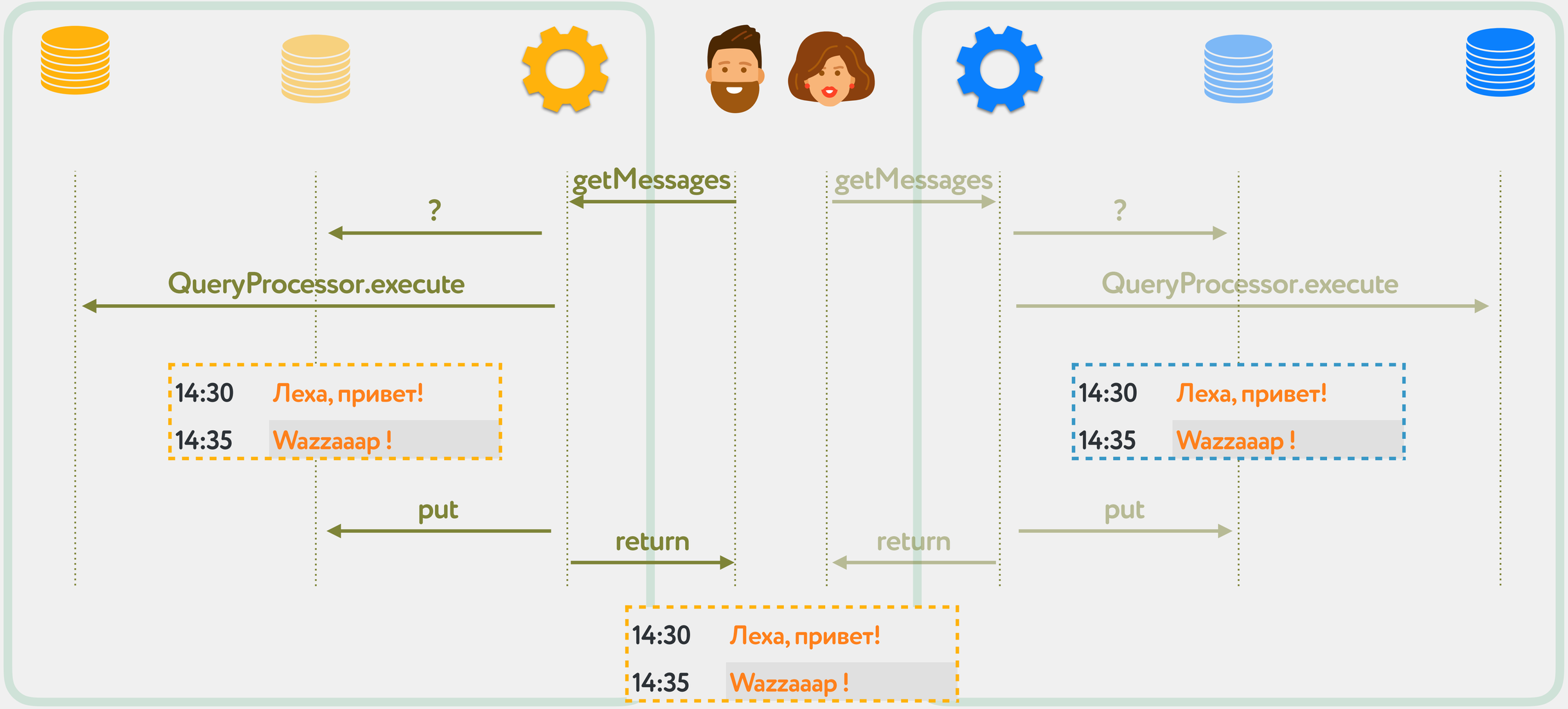
250 млн
чатов

500 гб

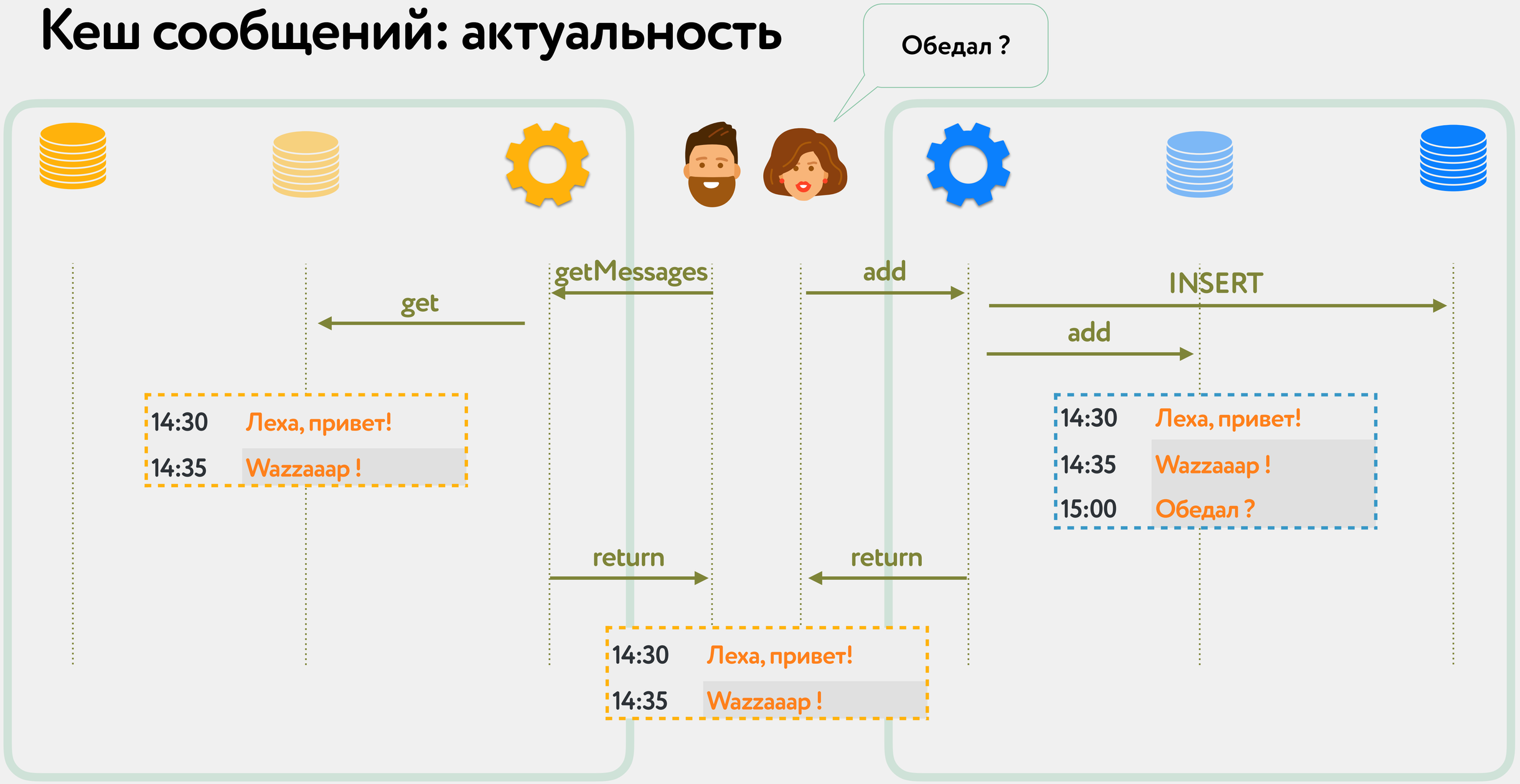
Кеш сообщений: getMessages



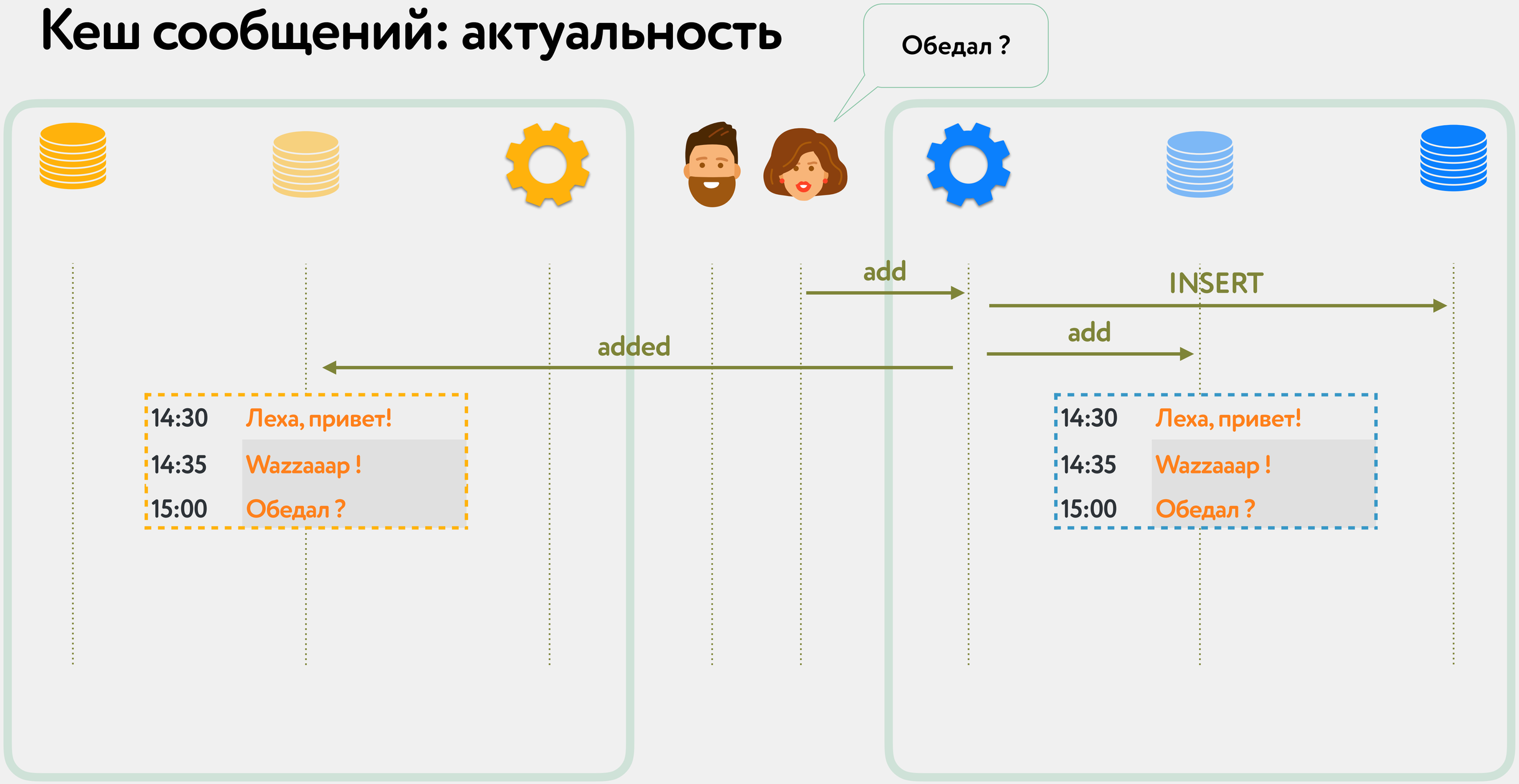
Кеш сообщений



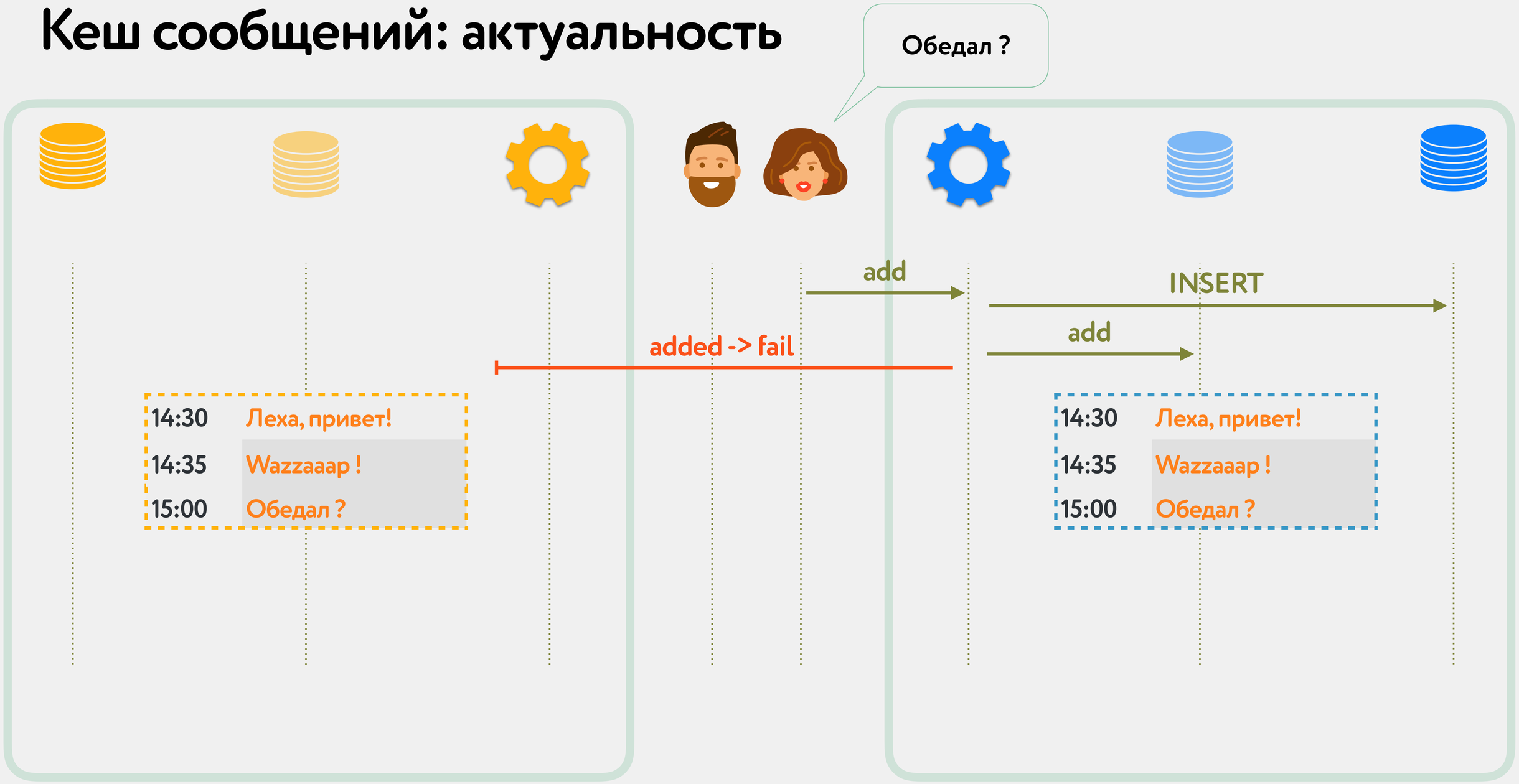
Кеш сообщений: актуальность



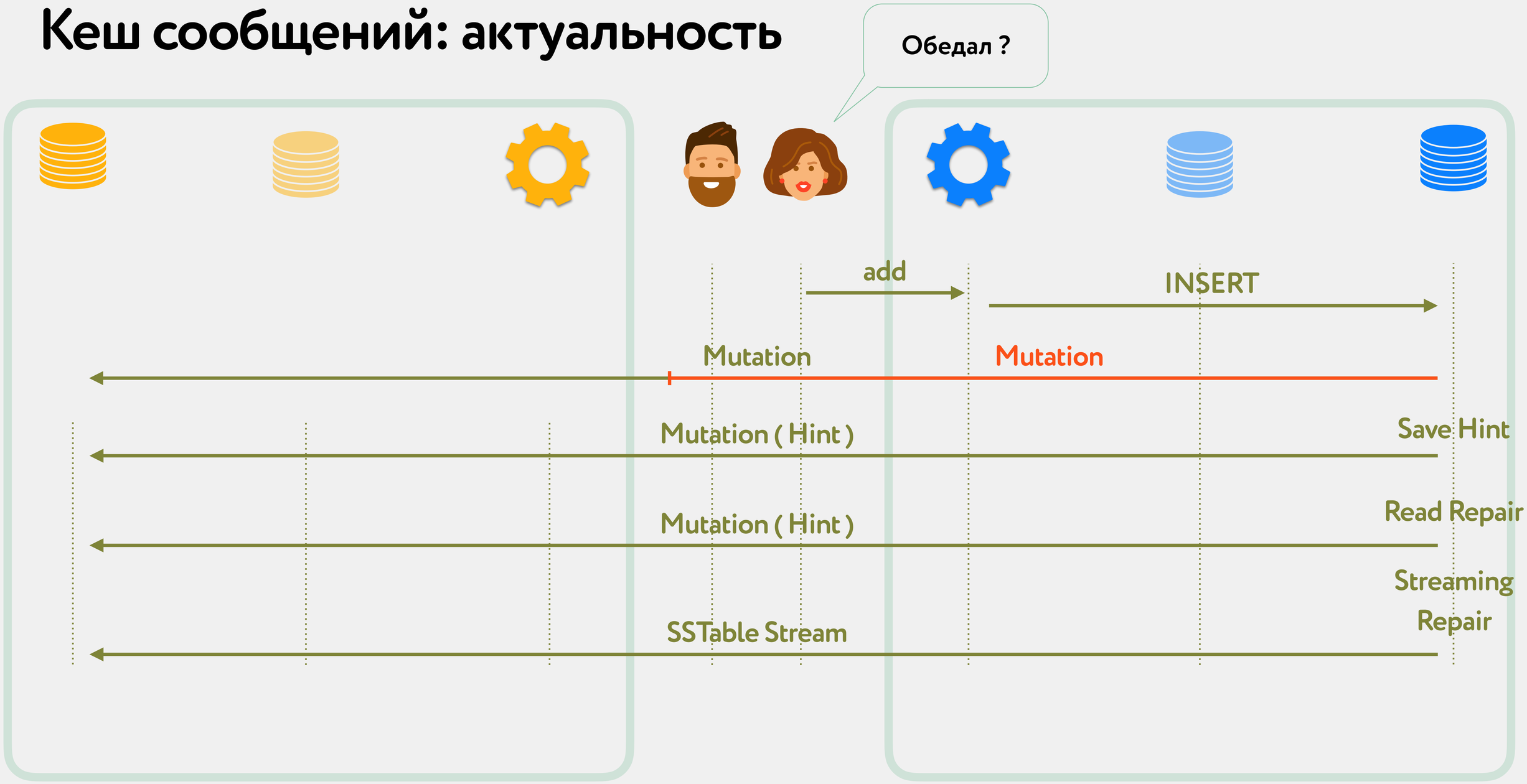
Кеш сообщений: актуальность



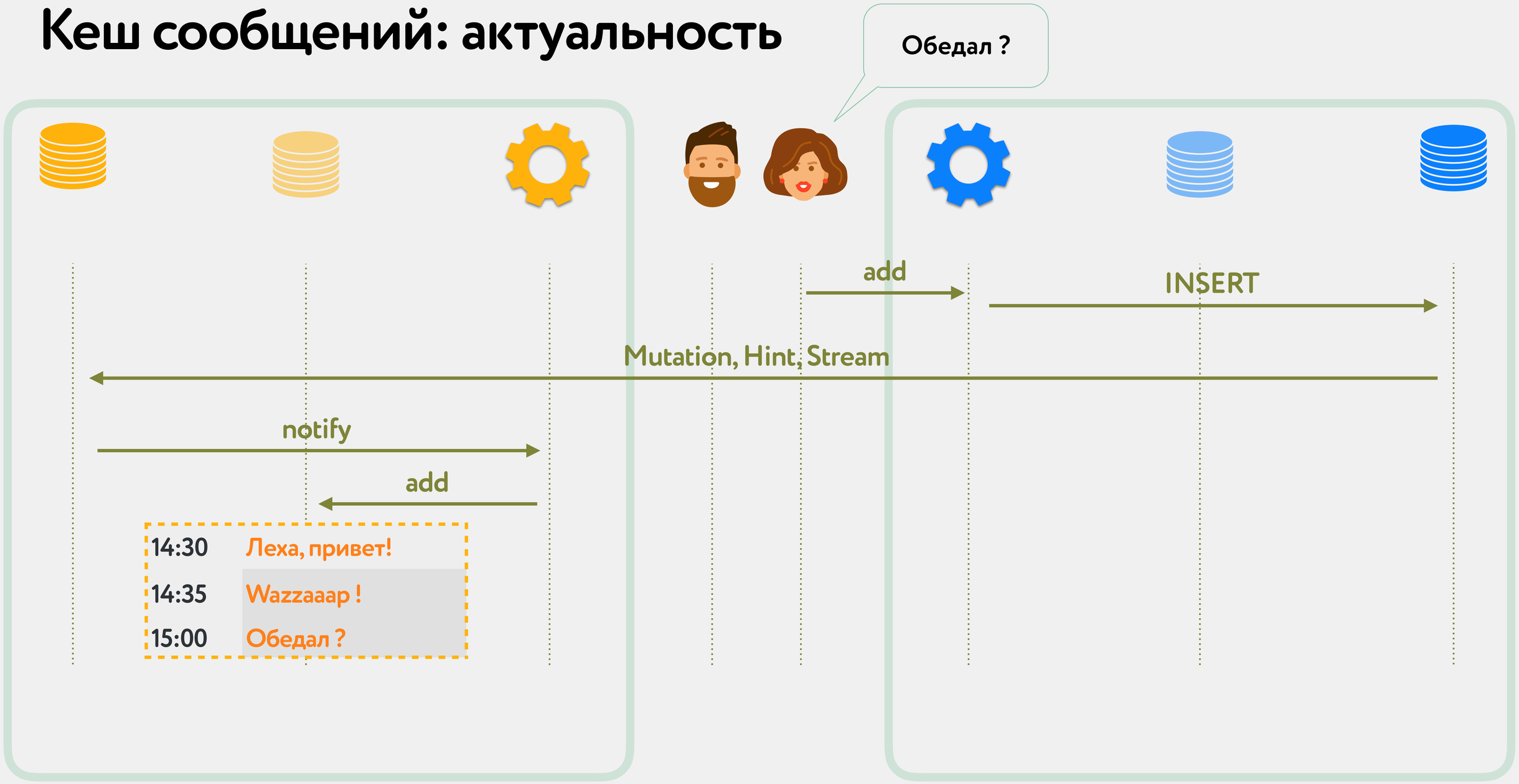
Кеш сообщений: актуальность



Кеш сообщений: актуальность



Кеш сообщений: актуальность



```
interface ApplyMutationListener
{
    void onApply(ByteBuffer key,
                 DeletionTime deletion,
                 Iterator<Unfiltered> atoms);
}
```

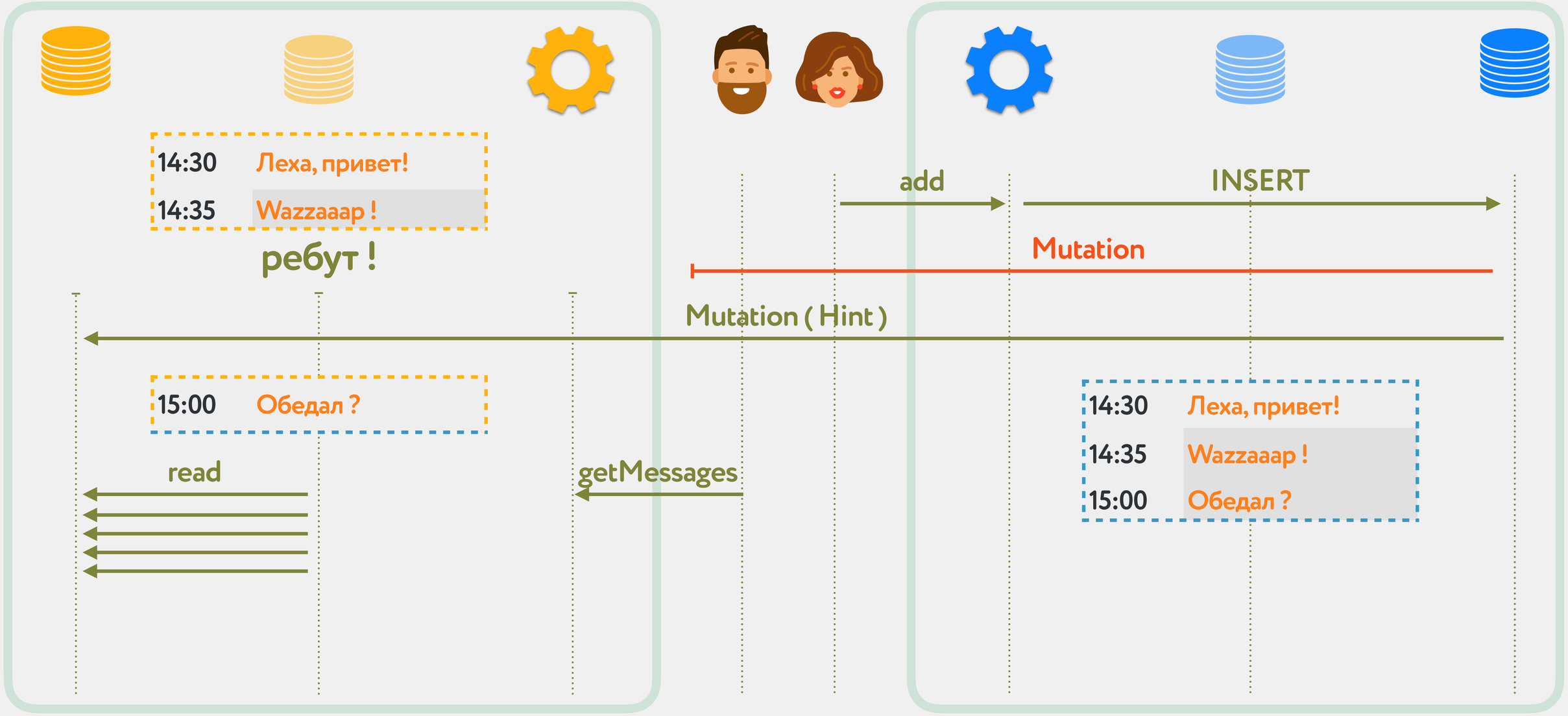
Mutation - записи:

```
package org.apache.cassandra.db;

public class Keyspace
{
    public void apply( Mutation mutation,
                       boolean writeCommitLog,
                       boolean updateIndexes,
                       boolean isDroppable )
```

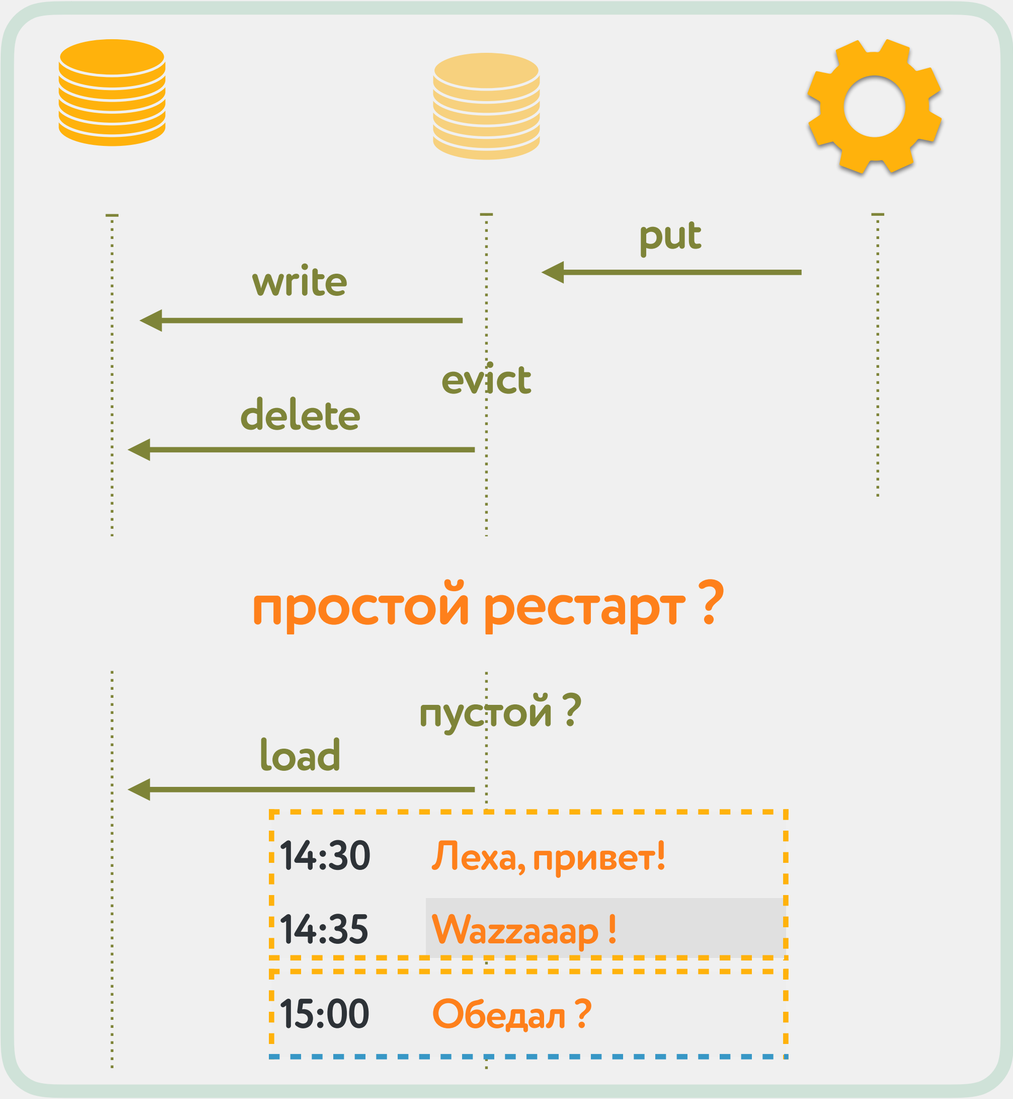
Streaming - таблицы:

```
package org.apache.cassandra.streaming;

public class StreamReceiveTask extends StreamTask
{
    //  holds references to SSTables received
    protected Collection<SSTableReader> sstables;

    private static class OnCompletionRunnable implements Runnable {
```

Кеш сообщений: потеря состояния



Кеш сообщений: потеря состояния




```
CREATE KEYSPACE Caches
    WITH REPLICATION = {
    'class': 'LocalStrategy'
    }
```

```
CREATE TABLE Caches.MessagesSnapshot (
    rowkey blob,
    value blob,
    PRIMARY KEY ( rowkey )
)
```

```
SELECT * FROM MessagesSnapshot
```

Кеш: оптимизируем рестарты

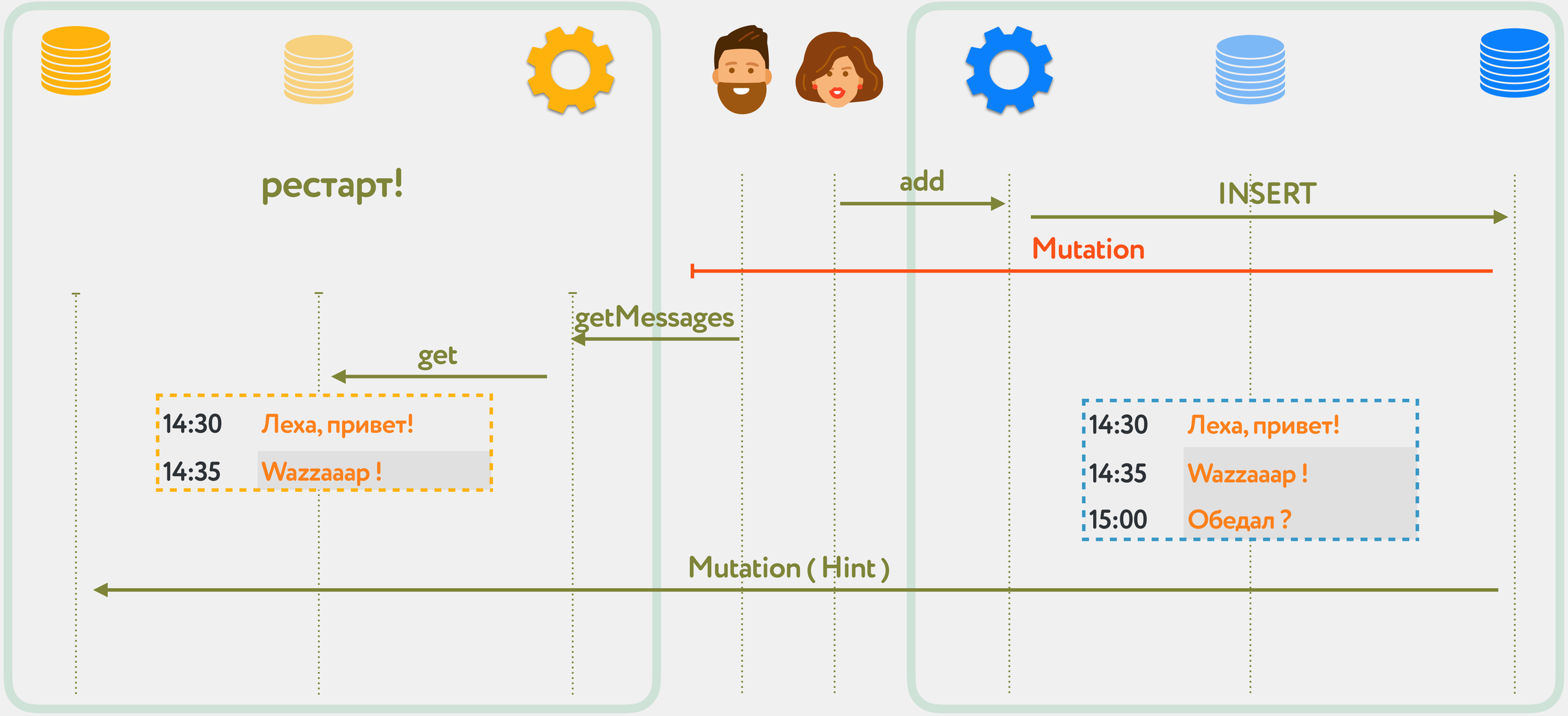
- **Разделяемая память**
  Shared Memory
- **/dev/shm/msgs-cache.mem**
  но это необязательно
- **tmpfs**
- **hugetlbfs**
  4K страницы -> 2M, 1G

https://github.com/odnoklassniki/one-nio

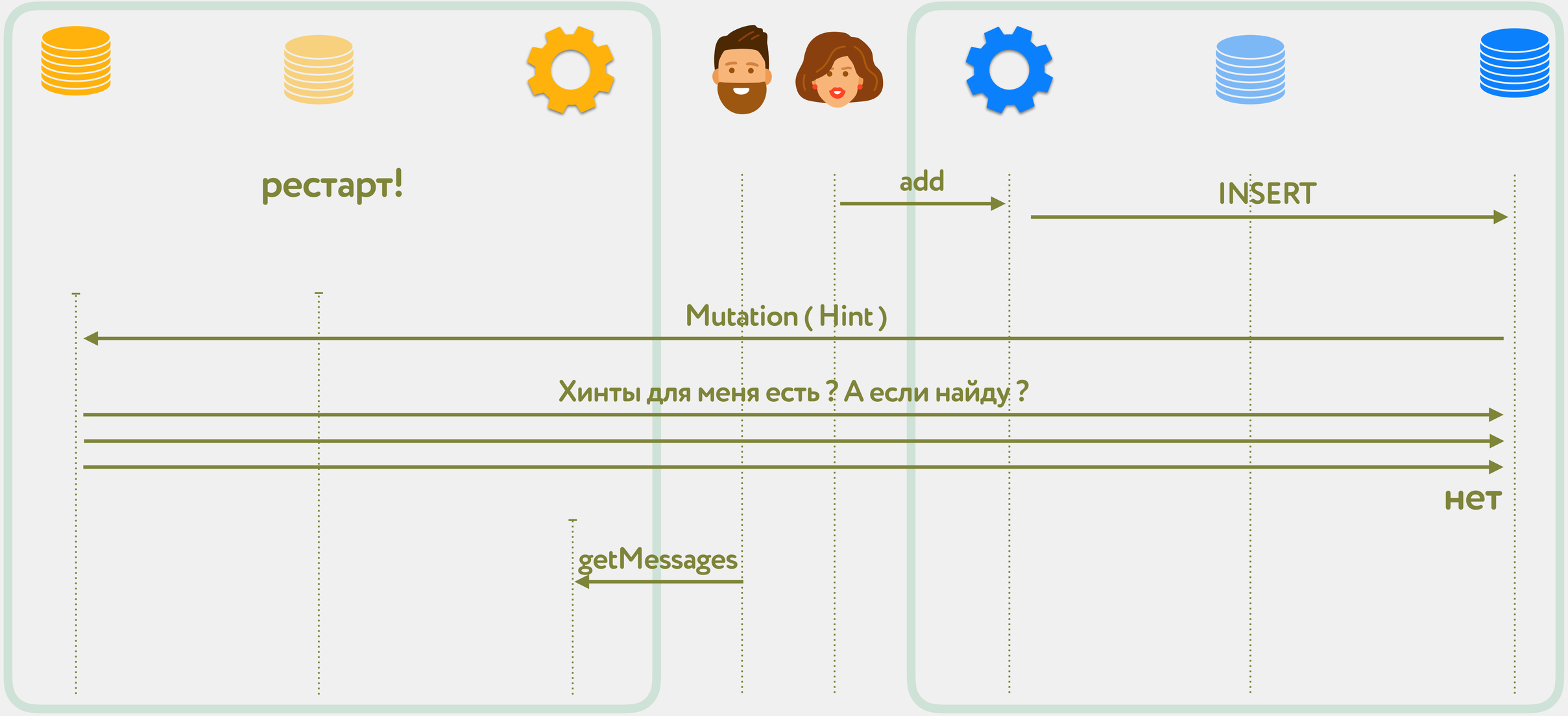
one.nio.mem

SharedMemoryMap

Кеш сообщений: ожидание согласованности



Кеш сообщений: ожидание согласованности



Мы хотим большего !

getLastMessages(chats[])

- **Множество чатов**

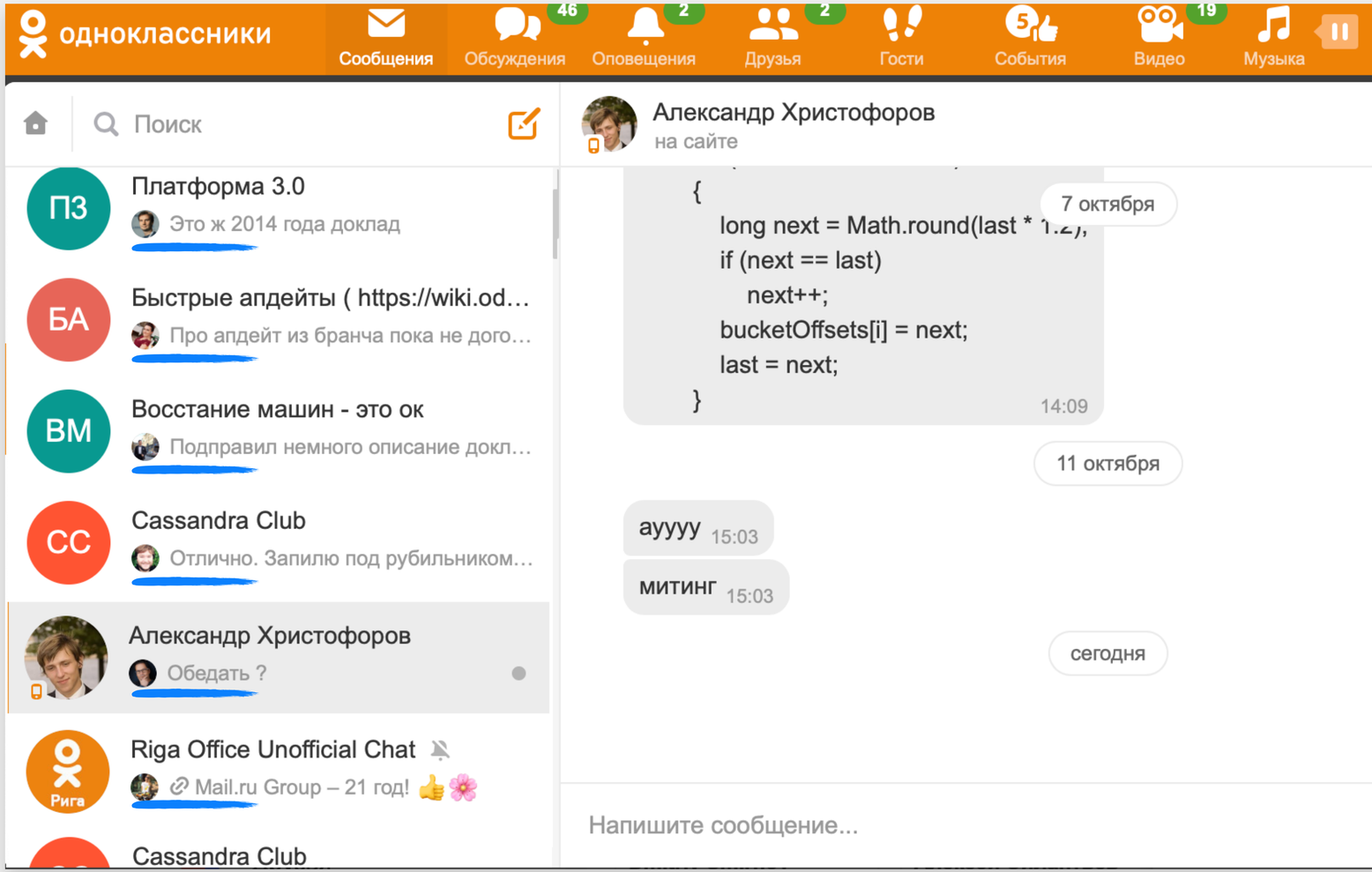
  Нет 1 ноды владеющей необходимыми данными

- **Часть в кеше часть нет**

  Грузить в кеш старые чаты смысла мало

- **Старые чаты не в кеше**

  Они согласованы



getLastMessages(chats[])

- **Множество чатов**

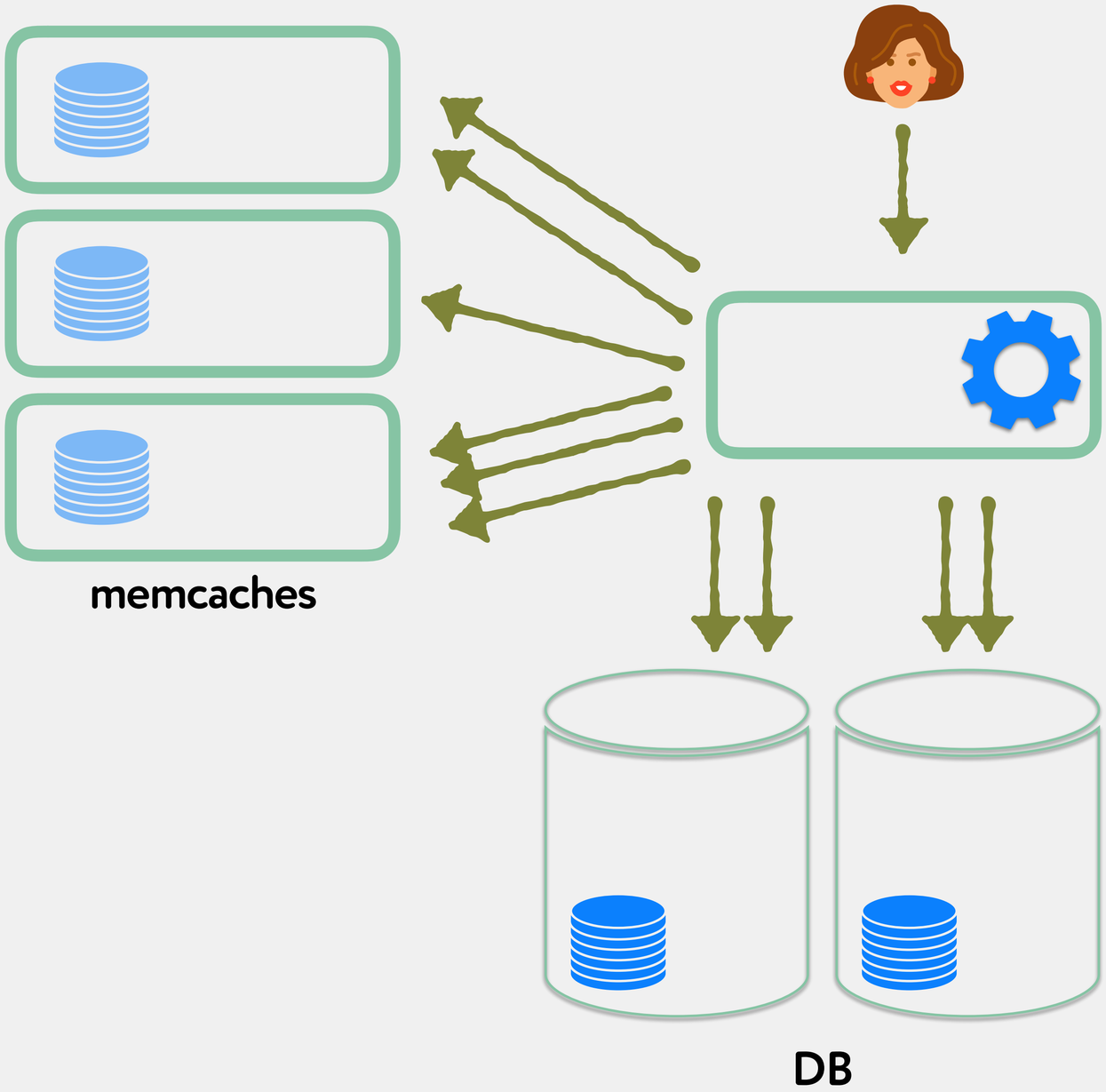
  Нет 1 ноды владеющей необходимыми данными

- **Часть в кеше часть нет**

  Грузить в кеш старые чаты смысла мало

- **Старые чаты не в кеше**

  Они согласованы



split & merge

```
Map<Long, Message> getLastMessages( long[] chatIds )
```

- **Множество чатов**

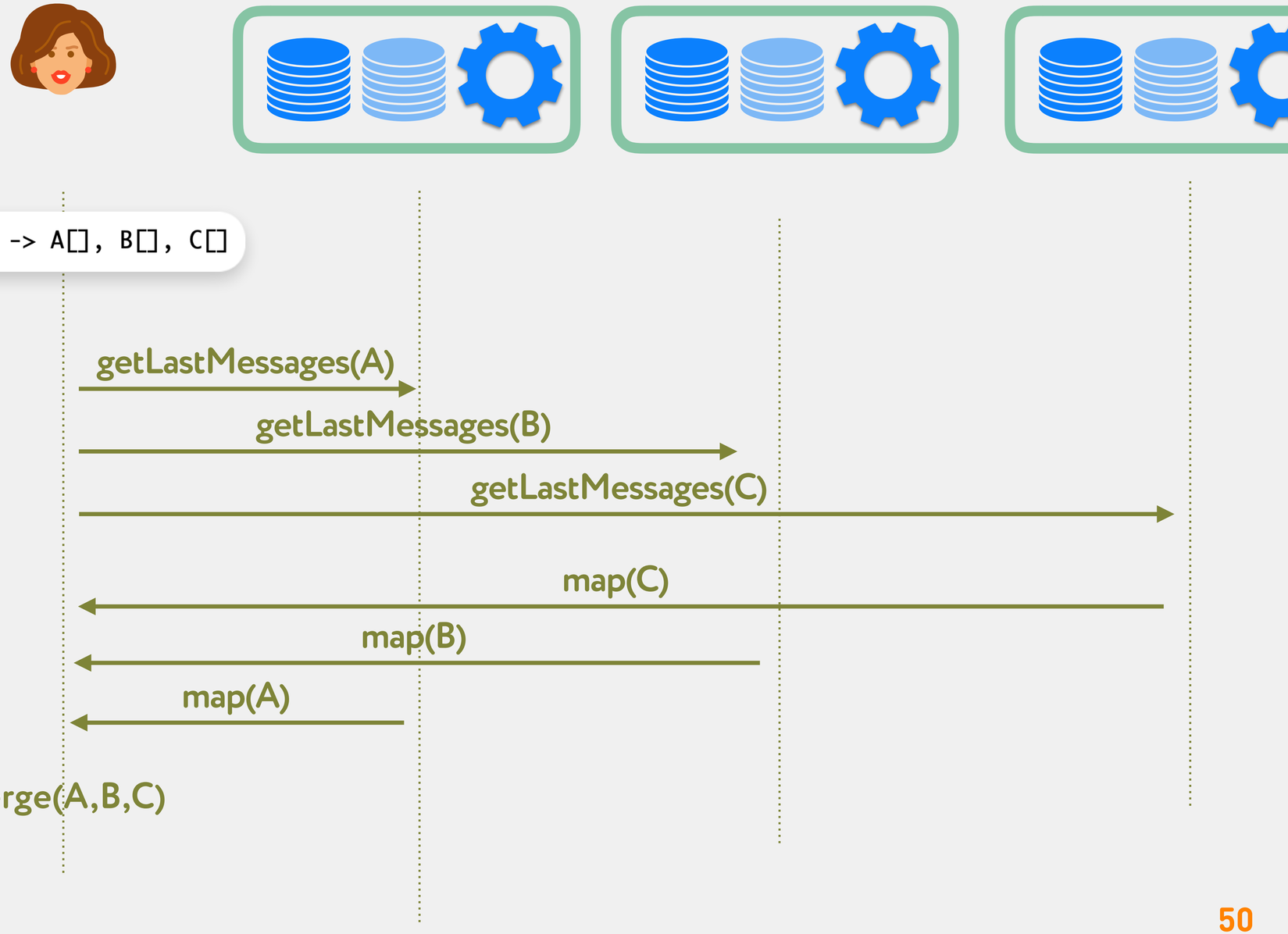
  Нет 1 ноды владеющей необходимыми данными

- **Часть в кеше часть нет**

  Грузить в кеш старые чаты смысла мало

- **Старых чатов больше**

  Согласованность не так важна



getLastMessages: локальное чтение

- **Множество чатов**

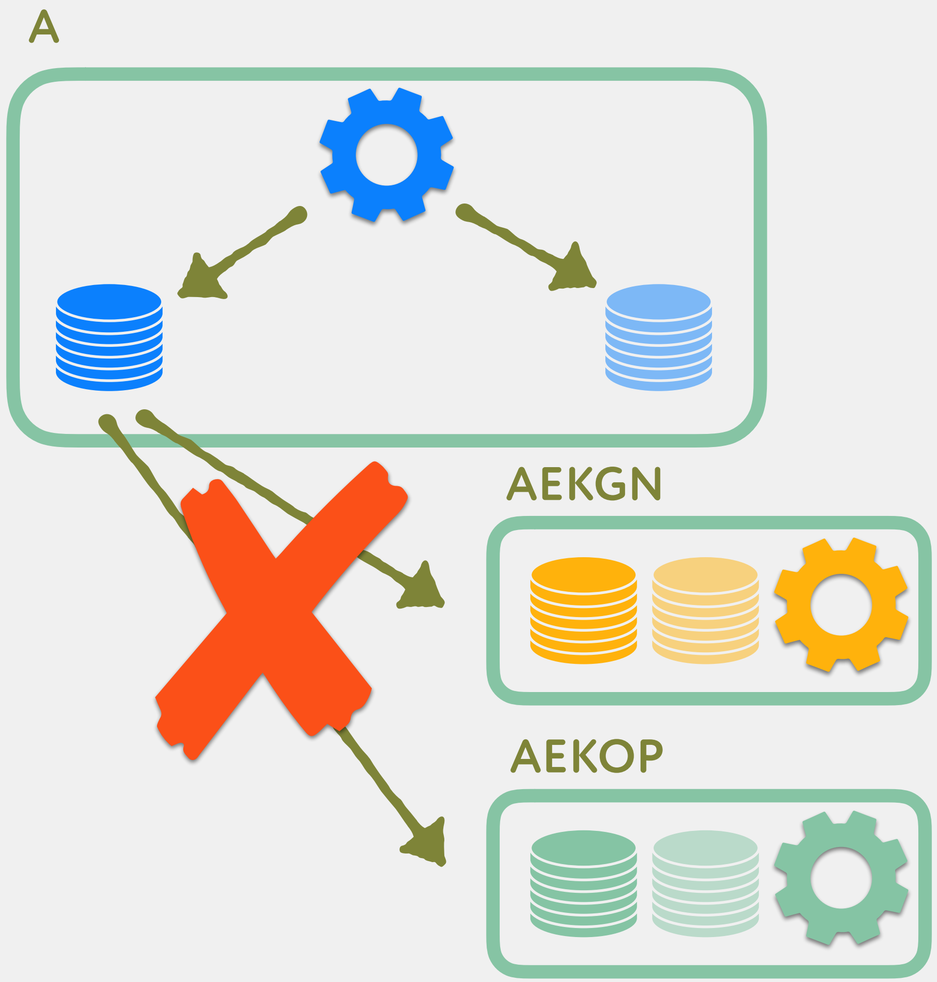
  Нет 1 ноды владеющей необходимыми данными

- **Часть в кеше часть нет**

  Грузить в кеш старые чаты смысла мало

- **Старых чатов больше**

  Согласованность не так важна



Локальное чтение

```
public Message getLastMessage(Long chatId) {

    Prepared prepared
        = QueryProcessor.prepareInternal( "SELECT ... WHERE chatId=?" );
    SelectStatement select
        = (SelectStatement) prepared.statement;

    QueryOptions opt
        = QueryProcessor.makeInternalOptions( prepared, chatId );

    ReadQuery query = select.getQuery( opt, FBUtilities.nowInSeconds() );

    UnfilteredPartitionIterator partitions
        = query.executeLocally( query.executionController() );

    UnfilteredRowIterator partition = partitions.next();
    partition.forEachRemaining( atom -> { /* unmarshalling code */ } );
}
```

Полнотекстовый П

- **Строим индекс**

  lucene.apache.org

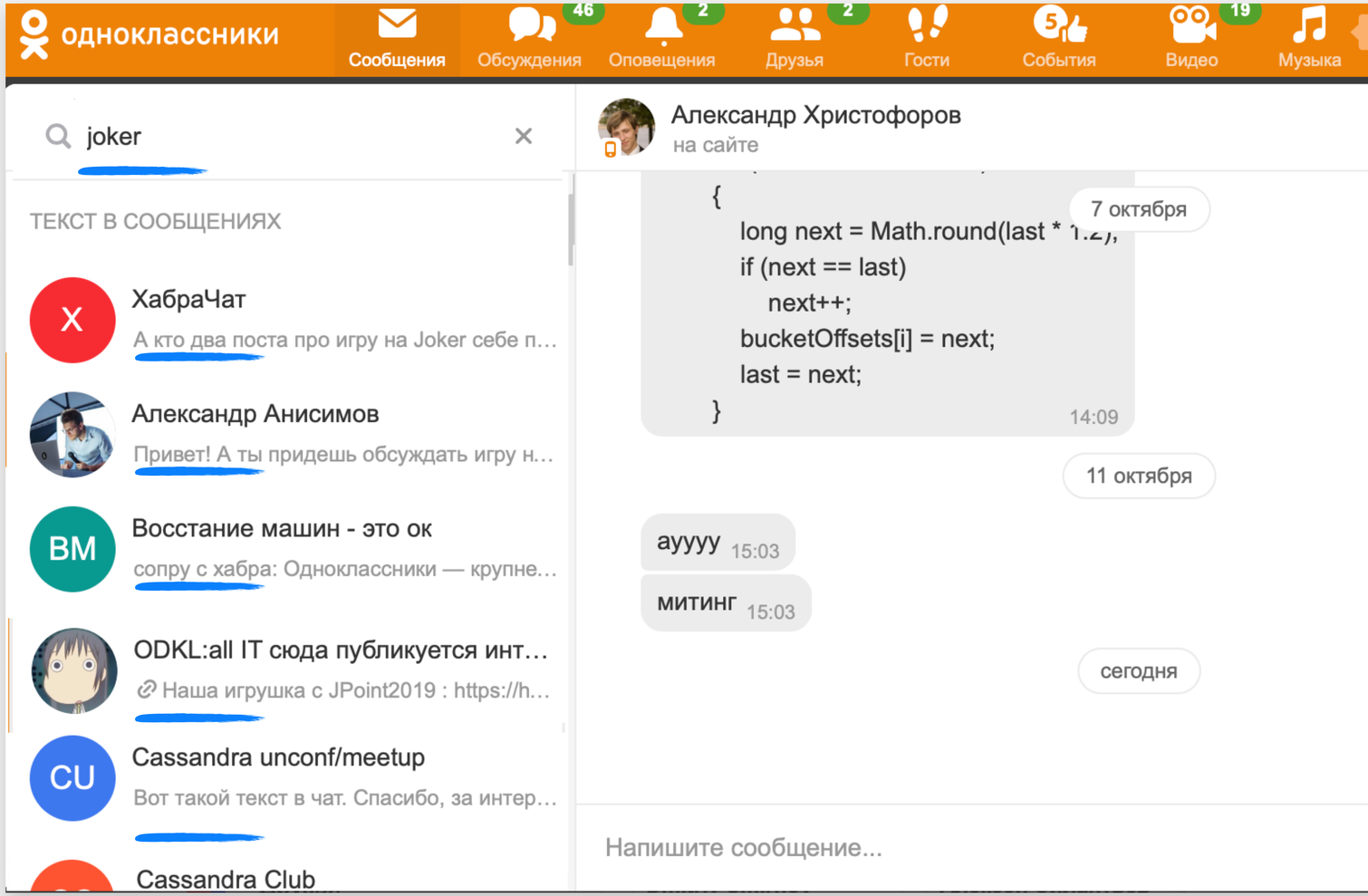
- **Отдельный для каждого чата**

  Один большой индекс не сработает

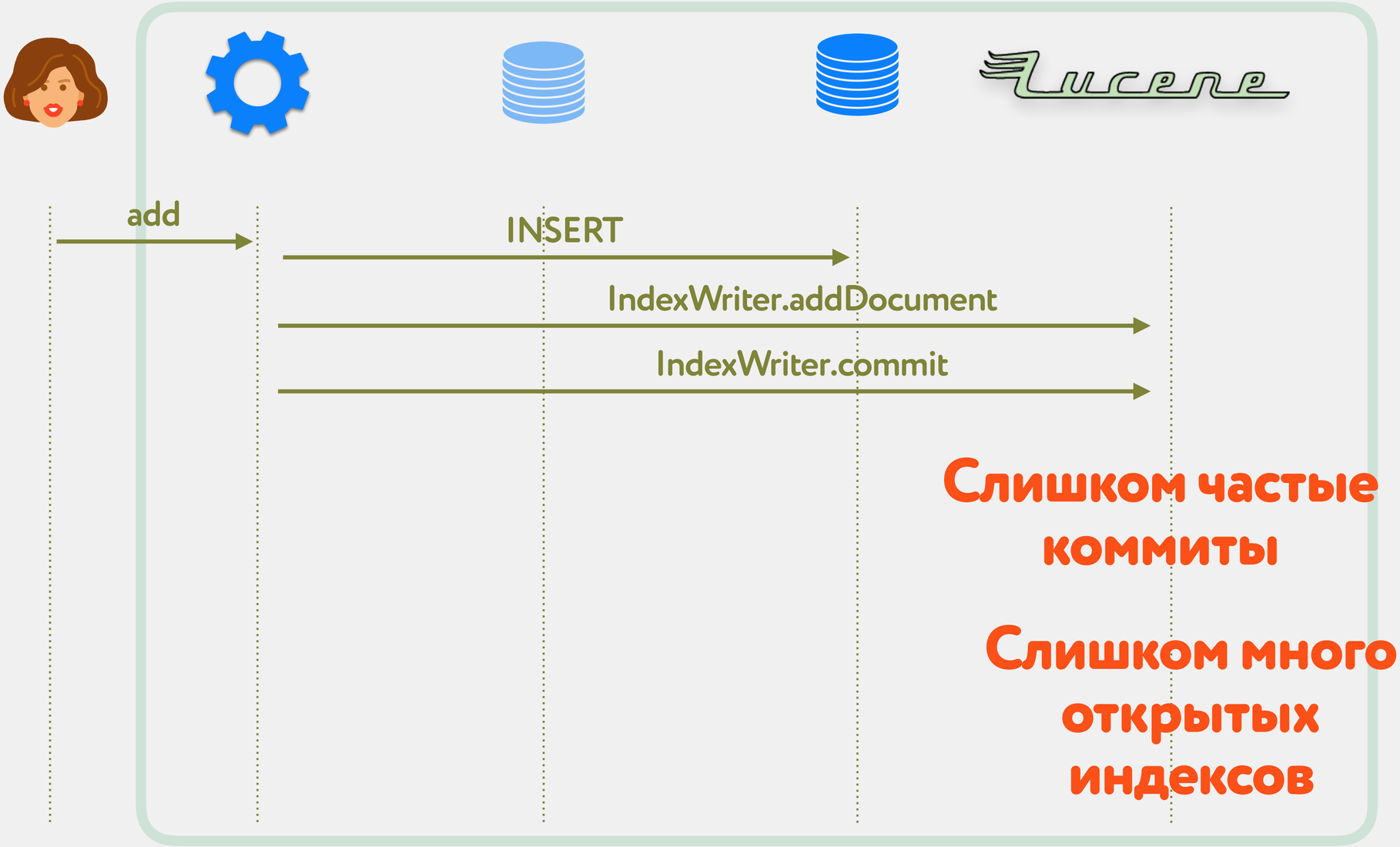
- **Для больших чатов**

  Для небольших индекс можно строить

  непосредственно перед поиском



Полнотекстовый П



Compaction

- **Слияние поколений данных**

  В файлах на диске.

- **В порядке следования ключей**

  PartitioningKey, Clustering Key

```
package org.apache.cassandra.db.compaction;

public class CompactionManager implements CompactionManagerM
{
```

Бесплатная массовая обработка данных

(7) http://cassandra.apache.org/doc/latest/operating/compaction.html

Эксплуатация

Более долгий старт

кто виноват:

- **Инициализация БД**

  Зависит от скорости дисков с данными

  Скорости проигрывания WAL

- **Загрузка кеша из CF**

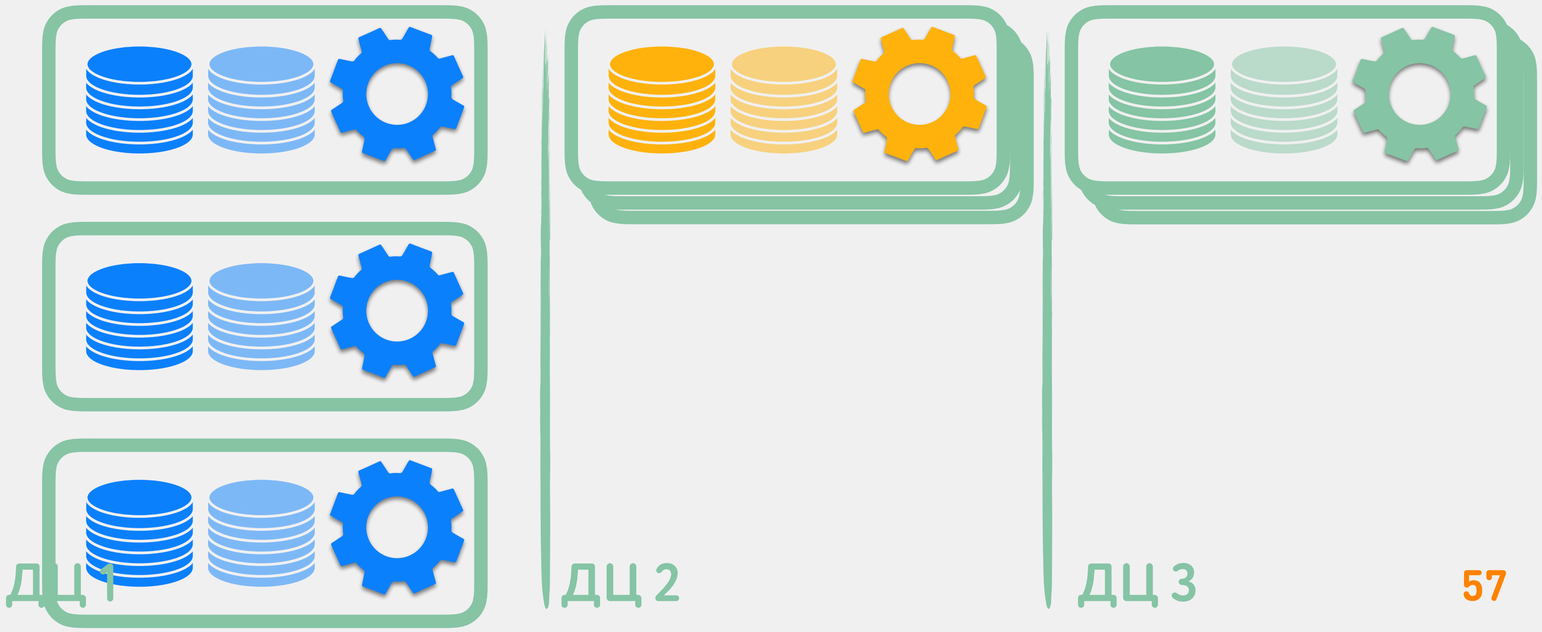
  Размер кеша, contention, CPU

- **Ожидание согласованности**

  Количество пропущенных мутаций

что делать:

- **Параллелизировать выкладку по зонам доступности**



Более частые рестарты БД

кто виноват:

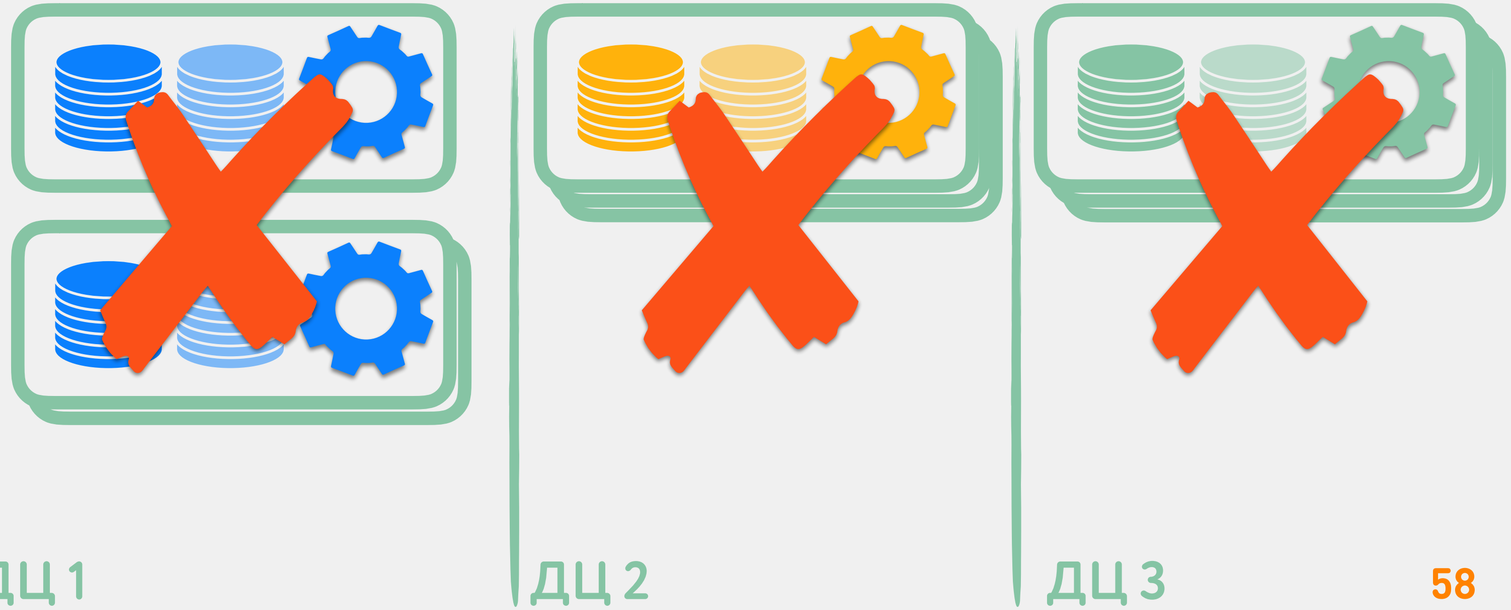
- **БД теперь вместе с приложением**

что делать:

- **Ничего, это хорошо**

  Позволяет отладить отказы зон доступности в контролируемой среде

Не трогай, это же БАЗА ДАННЫХ, ее нельзя рестартать - мы же все п*ем[1] !!11



(1) Потеряем все данные

Взаимное влияние прикладухи на БД

кто виноват:

- **БД теперь вместе с приложением**

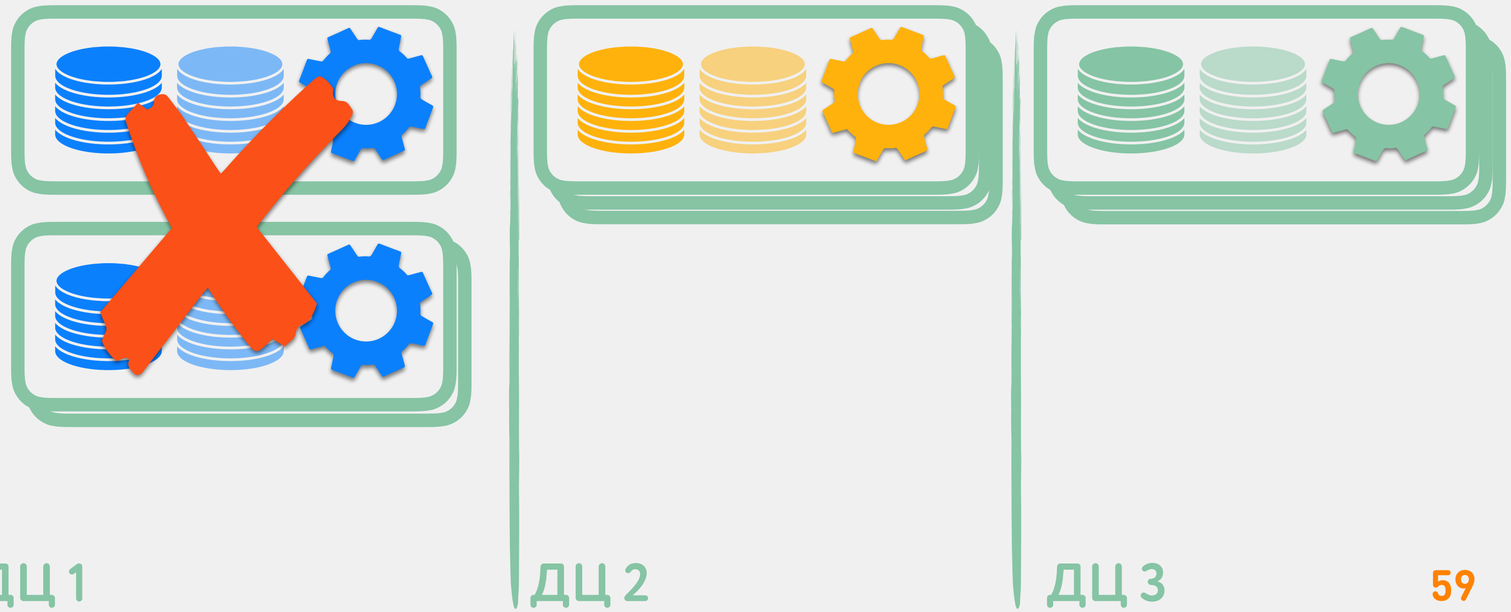
  Прикладные баги в потреблении CPU, памяти могут уложить все реплики сервиса

что делать:

- **Продленная тестовая эксплуатация новой версии**

  В одной зоне доступности (остальные остаются на старой)

- **Рубильники на опасный функционал**



Апгрейд на новую версию встроенной БД

кто виноват:

- **Внутреннее API БД**

  Которое может измениться от версии к версии БД

ОНИ все поменяют и мы застрянем на старой версии БД - и потом мы же все п*ем[1] !!11

что делать:

- **Архитектура БД меняется медленно**

  9 лет наблюдения за Cassandra выявило только страсть к переименованию.

- **Современные БД поддерживают rolling upgrades**

  Делаем версию сервиса с новой библиотекой БД,
  Выкатываем на 1 ДЦ,
  Ждем что сломается,
  В тяжелых случаях - откат и восстанавливаем данные из реплик

(1) Потеряем все данные, клиентов, работу и не сможем их потом восстановить

Итог

- **Эффективнее и надежнее**

  за счет отсутствия потерь на маршалинг и сеть

- **Гарантии консистентности в кешах**

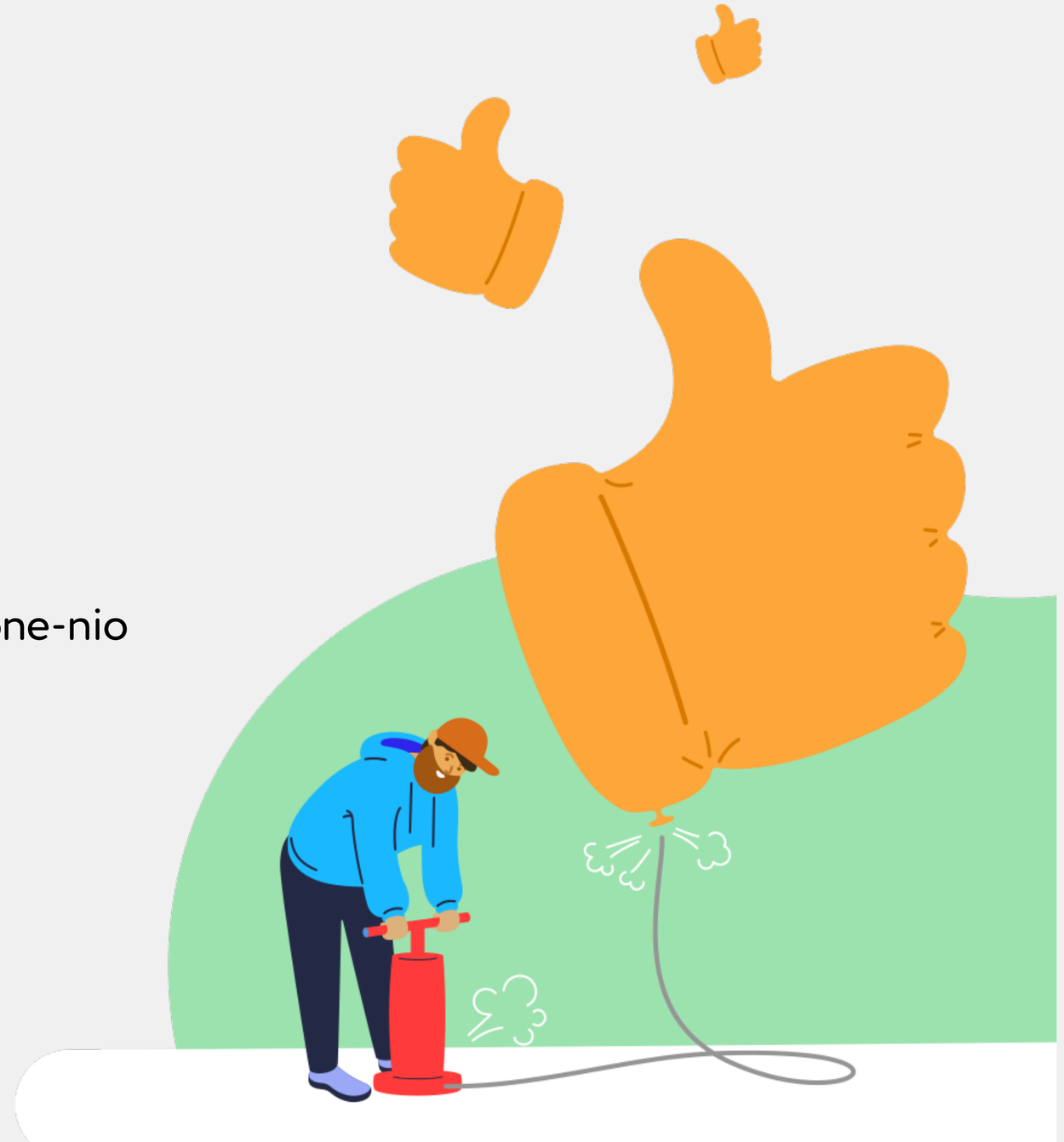
  За счет совмещения кеша и БД в 1 процессе

- **Относительно просто реализуется**

  Все самое сложное уже реализовано в Cassandra и one-nio

- **Мы широко и давно используем**

  лента, обсуждения, посты, нотификации ,…



Тут можно узнать больше:

Распределенные системы в Одноклассниках

Олег Анастасьев

ok.ru/video/97105087088

Новый граф Одноклассников

Антон Иванов

highload.ru/moscow/2019/abstracts/5793

Восстание машин - это ОК

Леонид Талалаев

highload.ru/moscow/2019/abstracts/5784


одноклассники