

26 февраля
13:00 МСК



R•Style
Softlab

TARANTOOL



Tarantool Cartridge: разработка отказоустойчивого кластера
Григоров Андрей (R-Style Softlab)  peneksglazami

Система аутентификации интернет-банка



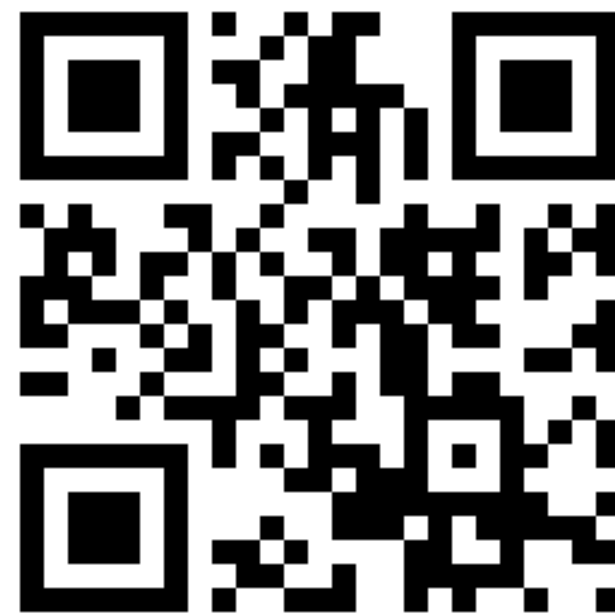
TARANTOOL

План доклада

- Что такое Tarantool?
- Варианты хранения данных: в памяти и на диске
- Шардирование данных в Tarantool
- Как построить кластер Tarantool
- Фреймворк Tarantool Cartridge
- Разработка приложений на java, работающих с Tarantool Cartridge
- Конкурс

Заходим на
menti.com

39 20 89 9



Что такое Tarantool?



Продукты ▾ Услуги ▾ Решения Клиенты Документация ▾ Контакты ▾

Войти ▾  ▾

Скачать

Что такое Tarantool?

Tarantool – это база данных с открытым исходным кодом, которая умеет хранить все в оперативной памяти. Она обслуживает до миллиона запросов в секунду, ищет по вторичным индексам и поддерживает SQL.

В Tarantool можно исполнять код рядом с данными. Встроенный язык программирования Lua поможет реализовать любую бизнес-логику и ускорить ваше решение. Избавляйтесь от устаревших записей, синхронизируйтесь с другими источниками данных, реализуйте HTTP-сервис.



<https://www.tarantool.io/>

История создания Tarantool

2008 год - Tarantool появился в рамках проекта социальной сети «Мой мир» как замена key-value хранилищу memcached для хранения сессий и профилей пользователей



В дальнейшем стал использоваться на проектах Mail.Ru и за её пределами (Badoo, Мегафон, Аэрофлот, Альфа-банк) для обработки «горячих» данных: пользовательские онлайн-сессий, настройки онлайн-приложений, различные кеши и т.д.

2016 год - Исходный код Tarantool опубликован в открытом доступе под лицензией BSD.


Версия 1.7 (2016 год) - Добавлен движок хранения данных на флеш и жестких дисках - Vinyl

Версия 2.1 (2018 год) - Добавлена поддержка SQL

Версия 2.1.2 (2019 год) - Фреймворк для разработки кластерных приложений (Tarantool Cartridge) стал частью Tarantool с открытым кодом

Версия 2.6 (2020 год) - Синхронная репликация данных

Tarantool на GitHub



Tarantool

In-memory computing platform with flexible data schema.

<https://tarantool.io> [@TarantoolDB](#) support@tarantool.io Verified


Repositories 184

Packages

People 14


Projects

Pinned repositories

 **tarantool**


Get your data in RAM. Get compute close to data. Enjoy the performance.

● Lua ☆ 2.5k 🍴 256

 **cartridge**


Out-of-the-box cluster manager for Tarantool with a modern web UI

● Lua ☆ 59 🍴 6

 **queue**


Create task queues, add and take jobs, monitor failed tasks

● Lua ☆ 188 🍴 35

 **tarantool-php**


PECL PHP driver for Tarantool

● C ☆ 75 🍴 20

 **tarantool-python**

Python driver for Tarantool

● Python ☆ 75 🍴 39

 **vshard**

The new generation of sharding based on virtual buckets

● Lua ☆ 71 🍴 18

<https://github.com/tarantool>

Поддерживаемые платформы: GNU / Linux, Mac OS и FreeBSD

Отличия от других NoSQL-решений



- Модель данных
- Кооперативный транзакционный менеджер
- Сервер приложений

<https://www.youtube.com/watch?v=sbSTZmkk5LE>

Сервер приложений Lua

app.lua

```
print('Hello, world!')
```

```
$ tarantool app.lua  
Hello, world!
```



Lua (лу́а, с порт. — «луна») — скриптовый язык программирования, разработанный в подразделении Tecgraf (Computer Graphics Technology Group) Католического университета Рио-де-Жанейро (Бразилия). Интерпретатор языка является свободно распространяемым, с открытыми исходными текстами на языке Си.

<https://ru.wikipedia.org/wiki/Lua>

Сервер приложений Lua

base.lua

```
box.cfg {  
    listen = 3301  
}  
box.once("bootstrap",  
    function()  
        box.schema.space.create('users')  
        box.space.users:format({  
            { 'login', 'string' },  
            { 'password', 'string' }  
        })  
        box.space.users:create_index('user_login_index', {  
            parts = { { field = "login", type = "string" } },  
            if_not_exists = true  
        })  
    end  
)
```

```
$ tarantool base.lua  
2021-02-25 15:00:41.250 [41436] main/103/base.lua C> version 2.5.3-0-gf93e480  
2021-02-25 16:00:44.250 [41436] main/103/base.lua C> log level 5  
...
```

Модель данных. Отличия от других

Отличие от реляционных СУБД Oracle, MySQL, PostgreSQL	Можно не иметь схему данных
Отличие от документоориентированных СУБД Couchbase, MongoDB	Может быть схема данных Схема данных не хранится в каждом документе

Tarantool - документоориентированная СУБД

Модель данных. Возможности Tarantool

- **Управление таблицами/спейсами**
Транзакции, вторичные ключи, локальные индексы, функциональные индексы
- **Обеспечение корректности данных**
Возможность задания схемы
- **Эволюция схемы данных**
Онлайн DDL (добавление/удаление индексов – неблокирующая операция)
Можно налету включать/выключать проверку форматов данных
- **Управление размером данных**
Компрессия данных

Модель данных. Пример описания модели данных

```
local users = box.schema.space.create('users',  
    { if_not_exists = true }  
)  
  
users:format({  
    { 'bucket_id', 'unsigned' },  
    { 'uuid', 'string' },  
    { 'login', 'string' },  
    { 'password', 'string' },  
    { 'status', 'string' }  
})
```

Модель данных. Пример описания модели данных

```
local users = box.schema.space.create('users',  
  { if_not_exists = true }  
)
```

```
users:format({  
  { 'bucket_id', 'unsigned' },  
  { 'uuid', 'string' },  
  { 'login', 'string' },  
  { 'password', 'string' },  
  { 'status', 'string' }  
})
```

```
users:create_index('user_login_index', {  
  parts = { { field = "login", type = "string" } },  
  if_not_exists = true  
})  
  
users:create_index('user_uuid_index', {  
  parts = { { field = "uuid", type = "string" } },  
  if_not_exists = true  
})
```


Модель данных. Пример описания модели данных

```
local users = box.schema.space.create('users',  
  { if_not_exists = true }  
)
```

```
users:format({  
  { 'bucket_id', 'unsigned' },  
  { 'uuid', 'string' },  
  { 'login', 'string' },  
  { 'password', 'string' },  
  { 'status', 'string' }  
})
```

```
users:create_index('user_login_index', {  
  parts = { { field = "login", type = "string" } },  
  if_not_exists = true  
})
```

```
users:create_index('user_uuid_index', {  
  parts = { { field = "uuid", type = "string" } },  
  if_not_exists = true  
})
```

```
box.space.users:insert { 1, '123e4567-e89b-12d3-a456-426655440000', 'misterX',  
  'c4ca4238a0b923820dcc509a6f75849b', 'ACTIVE' }
```

Семейства подсистем хранения данных

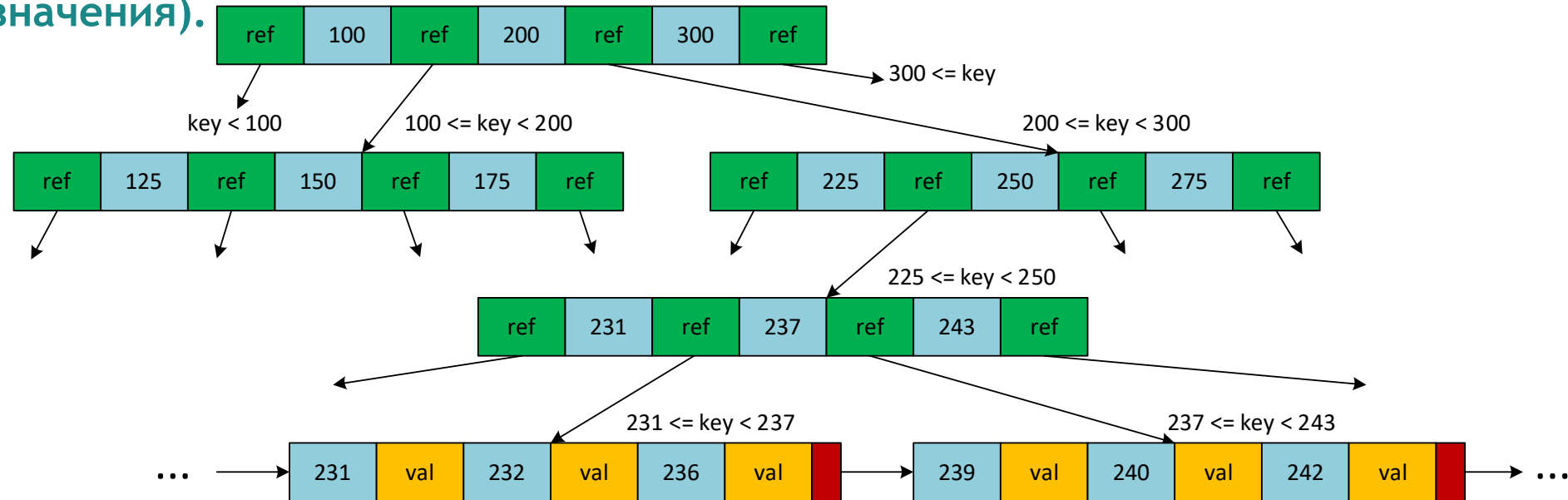
- **Постраничные**
Page-oriented storage engine
- **Журналированные**
Log-structured storage engine

Постраничные системы хранения данных

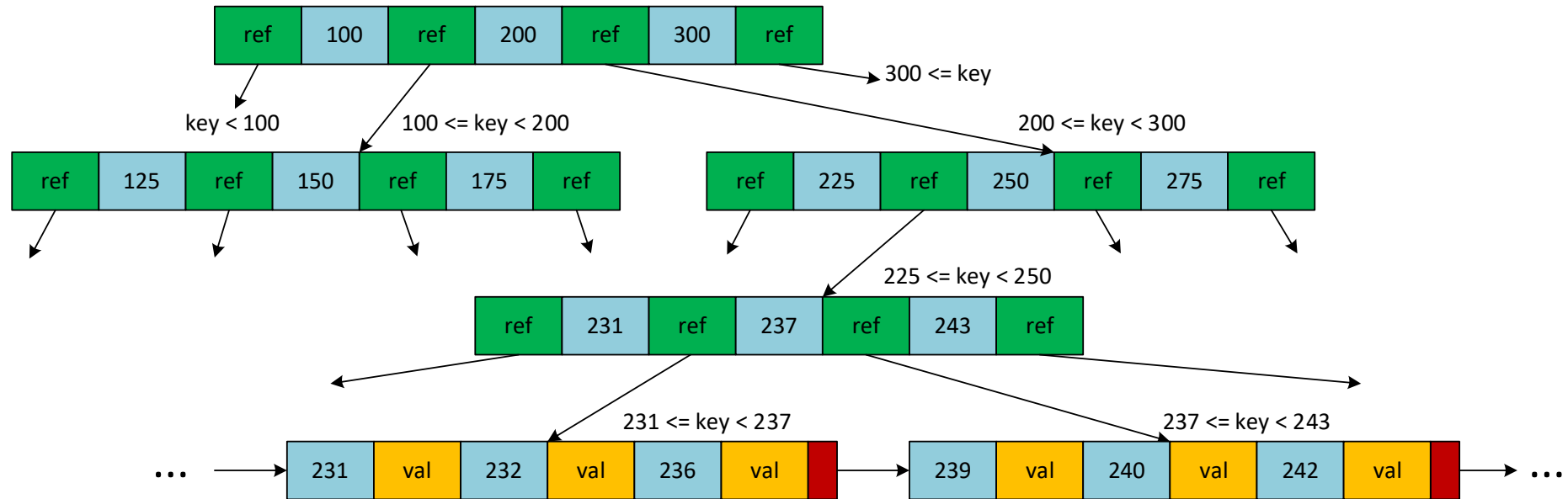
- Использует индексная структура на основе В-дерева.
- Использование В-деревьев впервые было предложено Р. Бэйером (R. Bayer) и Э. МакКрейтом (E. McCreight) в 1970 году в лаборатории Boeing.

B+tree

- В-дерево разбивают БД на **страницы** фиксированного размера (обычно 4-16 Кб).
- Страница читается и записывается полностью.
- Страницы имеют **адрес**, по которым на них могут ссылаться другие страницы.
- Одна страница назначается **корнем** В-дерева. С неё начинается поиск.
- Каждая не листовая страница содержит список ключей, которые определяют границы диапазонов ключей, и ссылки на дочерние страницы, которые соответствуют этим диапазонам.
- Страницы-листья содержат отдельные ключи и их значения (или ссылки на страницы, которые содержат значения).



Свойства B+tree

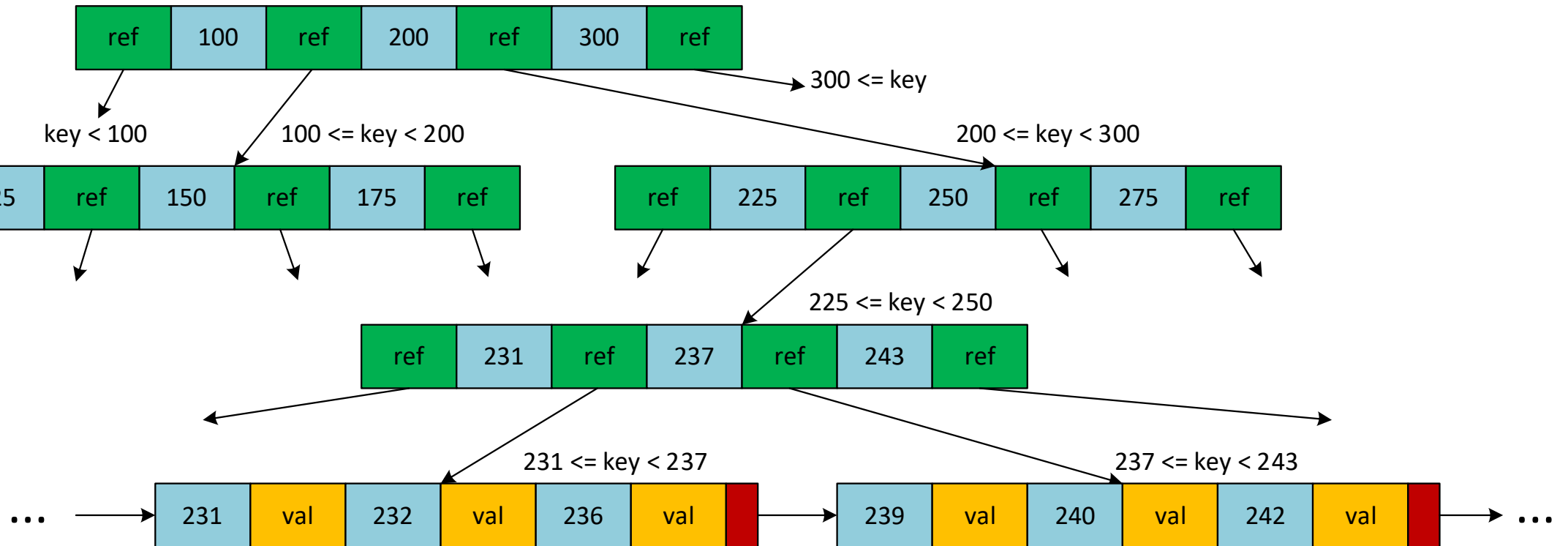


➤ Свойства B-дерева:

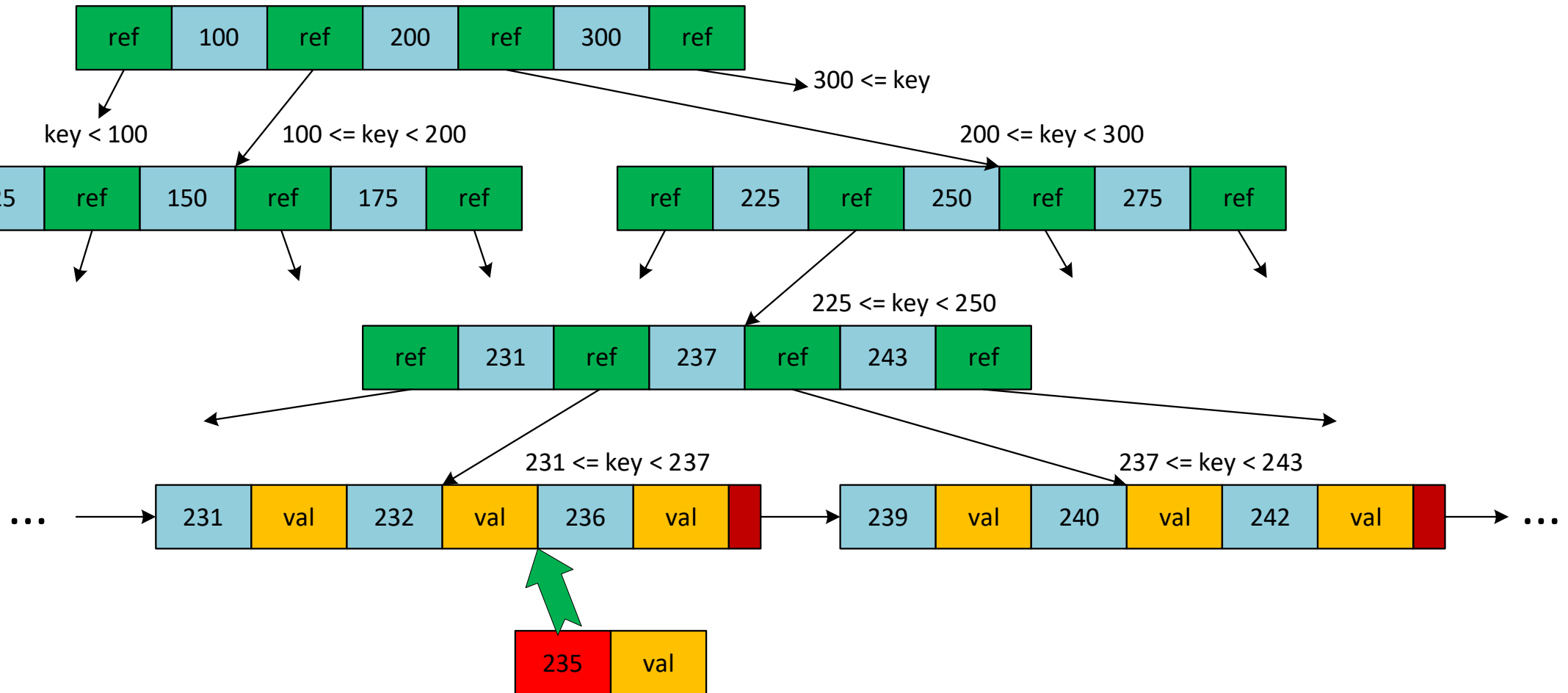
- сбалансированное
- сильноветвистое

Операция	Среднее время	Худшее время
Поиск элемента	$O(\log N)$	$O(\log N)$
Вставка элемента	$O(\log N)$	$O(\log N)$
Удаление элемента	$O(\log N)$	$O(\log N)$

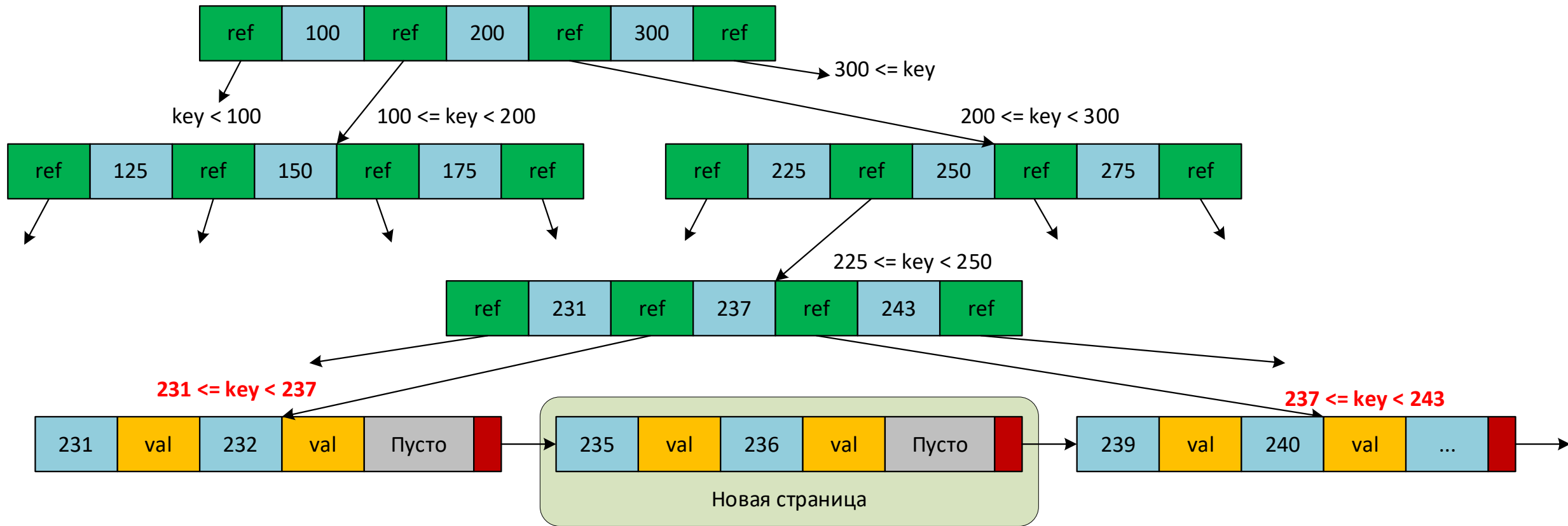
B+tree. Вставка нового ключа в дерево



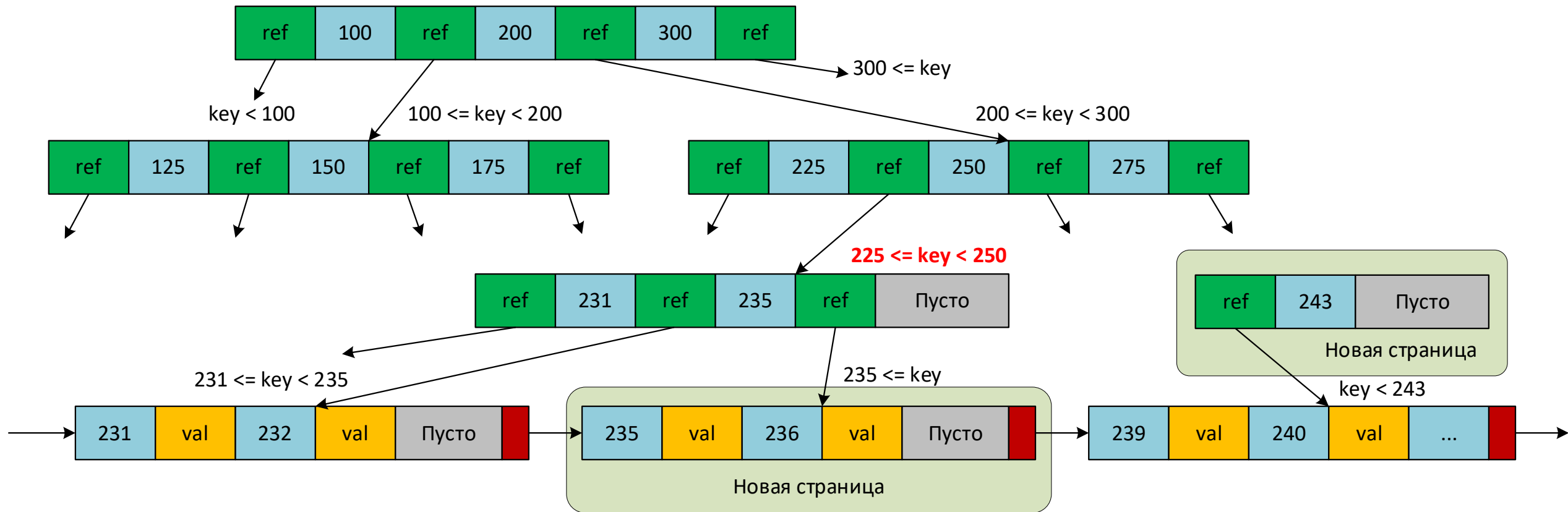
B+tree. Вставка нового ключа в дерево



B+tree. Вставка нового ключа в дерево



B+tree. Вставка нового ключа в дерево





Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

20% complete



For more information about this issue and possible fixes, visit <https://www.windows.com/stopcode>

If you call a support person, give them this info:

Stop code: CRITICAL_PROCESS_DIED

В-дерево. Обеспечение надёжности

Журнал упреждающей записи (write-ahead log) – файл, в который последовательно записываются все модификации В-дерева до того, как эти модификации в реальности будут применены к страницам дерева.

WAL в Tarantool

Режим работы журнала упреждающей записи в Tarantool определяется значением настройки **wal_mode**.

Возможные значения:

- **none** – журнал упреждающей записи не используется;
- **write** – файберы ожидают записи в журнал упреждающей записи (не `fsync(2)`); используется по умолчанию;
- **fsync** – файберы ожидают данные, синхронизация `fsync(2)` следует за каждой операцией `write(2)`.

Memtx. Используются B+*tree

Свойства bps-tree в memtx:

- Компактное хранение данных и преимущества cache-oblivious (от B-tree)
- Дешёвая итерация по списку ключей (от B+-tree)
- Большая компактность и сбалансированность (от B*-tree)
- Использование matras для MVCC и аллокации памяти

Мемtx. Для чего нужен матрас?

Матрас

Extent size: 16 kB → Block size: 16 B

Block id: 32 bit

id0 : high 11 bit id1 : middle 11 bit id2 : low 10 bit


L0 extent: array of 2048 pointers to L1 extents Use id0 as index to find pointer to L1 extent

L1 extents: arrays of 2048 pointers to L2 extents Use id1 as index to find pointer to L2 extent

L2 extents: arrays of 1024 blocks Use id2 as index to the block

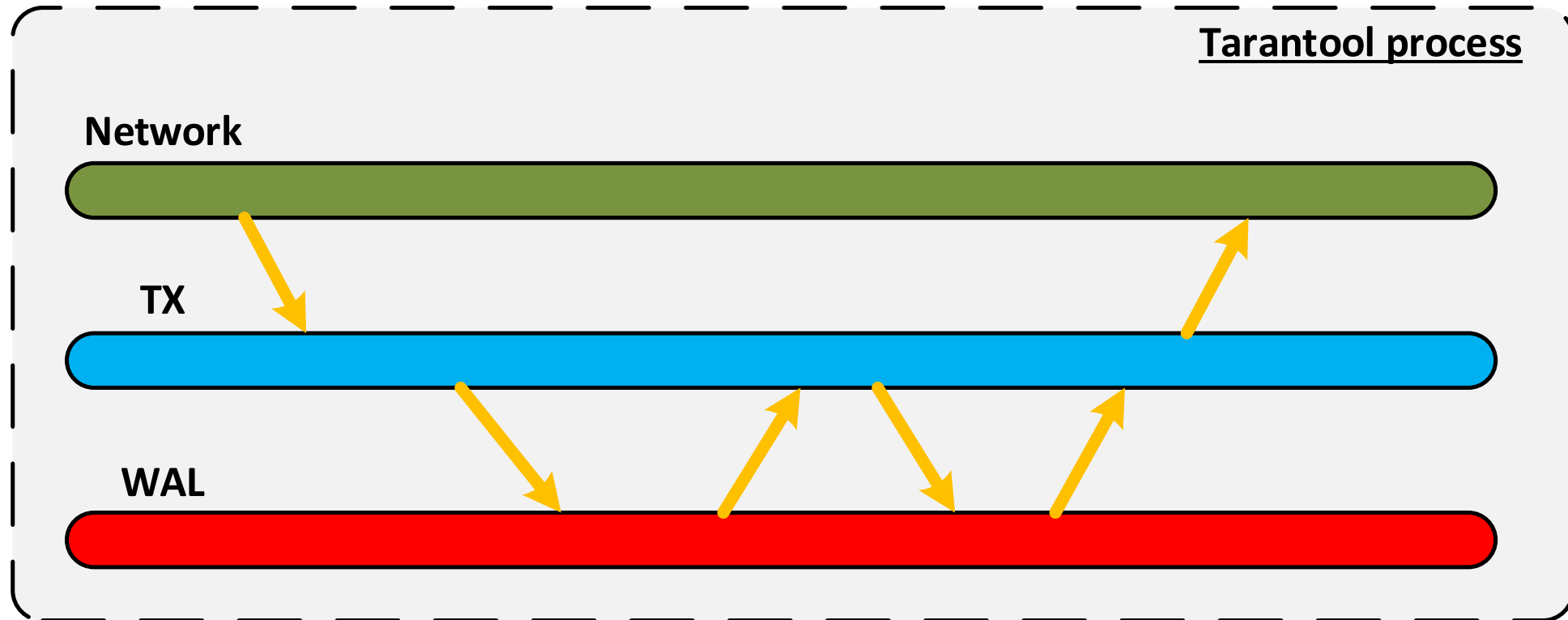
Константин Осипов
Что особенного в СУБД для данных в оперативной памяти

HighLoad++

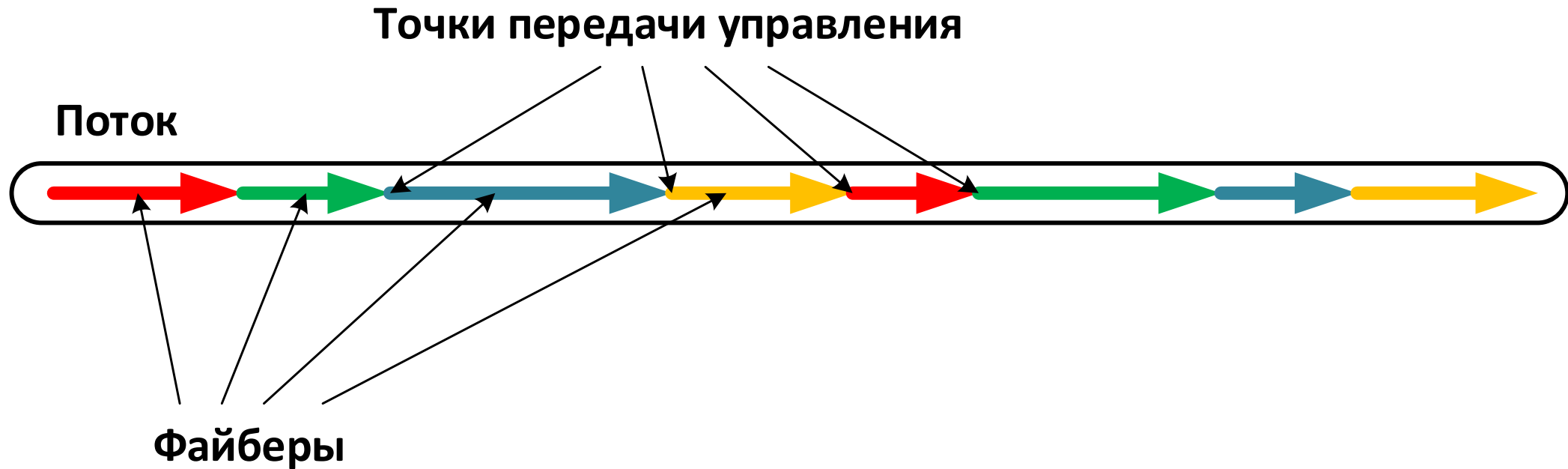


<https://www.youtube.com/watch?v=yrTF3qH8ey8>

Кооперативная многозадачность



Кооперативная многозадачность



Плюсы:

- Поток всегда занят исполнением задач (эффективная утилизация CPU)
- Отсутствие проблем с кешем CPU

Минусы:

- Нет возможности использовать все ядра процессора

Транзакции в memtx в версиях 2.5 и ниже

```
function transferMoney(from_account_id, to_account_id, amount_of_money)
    box.begin()
    box.space.accounts:update(from_account_id, { { '-', BALANCE_FIELD_NUM, amount_of_money } })
    box.space.accounts:update(to_account_id, { { '+', BALANCE_FIELD_NUM, amount_of_money } })
    box.commit()
end
```

Что мы знаем о транзакциях в memtx Tarantool версии 2.5 и ниже?

- После начала транзакции `box.begin()` все операции записи не приводят к записи в WAL, они копятся до вызова `box.commit`. Это позволяет не передавать управление другим файберам.
- Транзакции **атомарны**.
- Уровень изоляции транзакций **serializable**, если нет проблем с записью в WAL.
- Если происходит сбой при записи в WAL, то выполняется откат транзакции при этом другие транзакции могут прочесть неподтверждённые изменения (**dirty read**), то есть уровень изоляции транзакций становится **read uncommitted**.
- Выполнение **yield-операции** в файбере приводит к **откату транзакции**.
- Не поддерживаются интерактивные транзакции.

Новый менеджер транзакций в v.2.6.1



<https://www.youtube.com/watch?v=BiF7L2id-TU>



<https://habr.com/ru/company/mailru/blog/540842/>

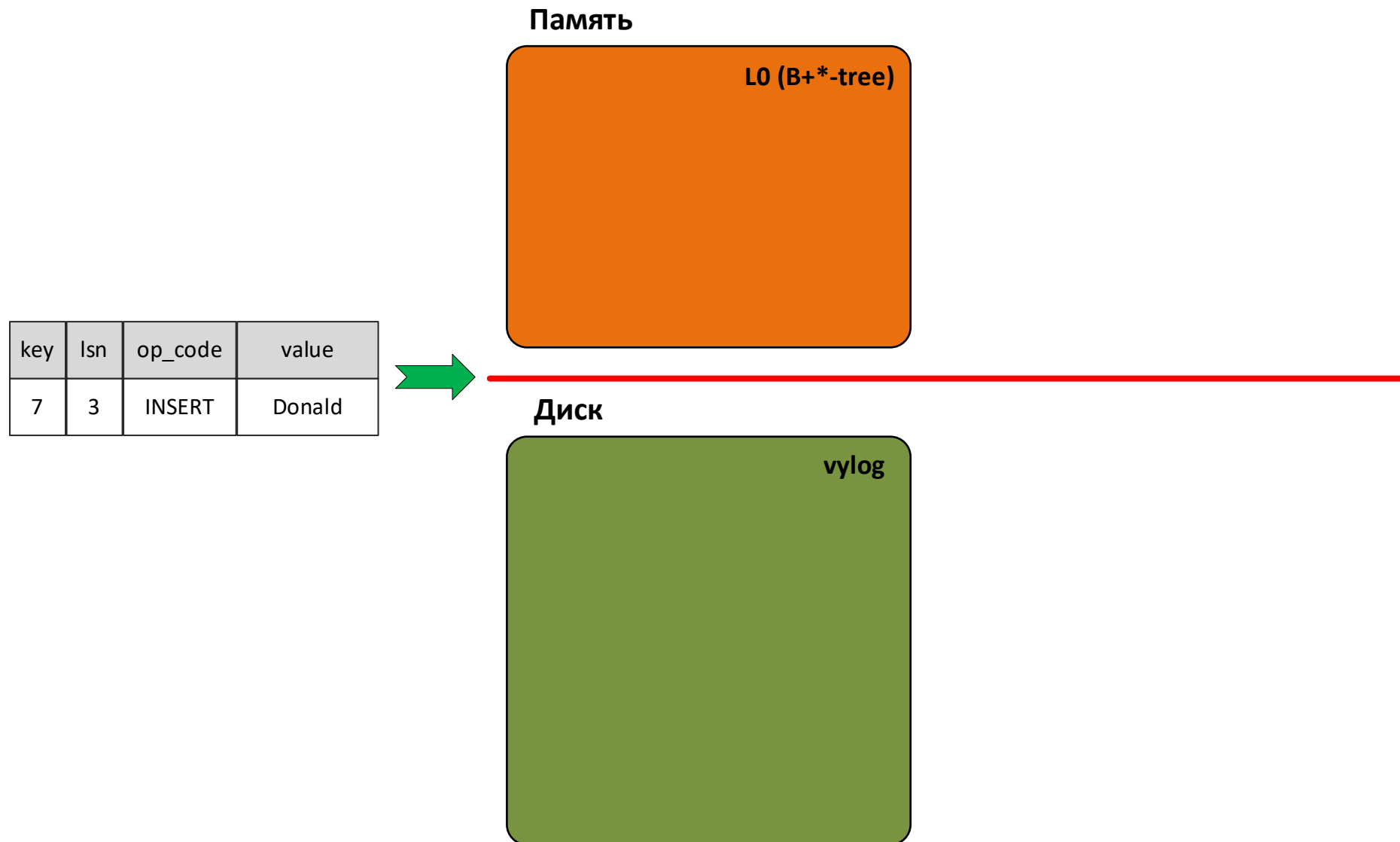
Vinyl. Движок для хранения данных на диске

- Появился в версии 1.7.
- В качестве основной структуры данных для хранения используются LSM-деревья (Log Structured Merge Tree).

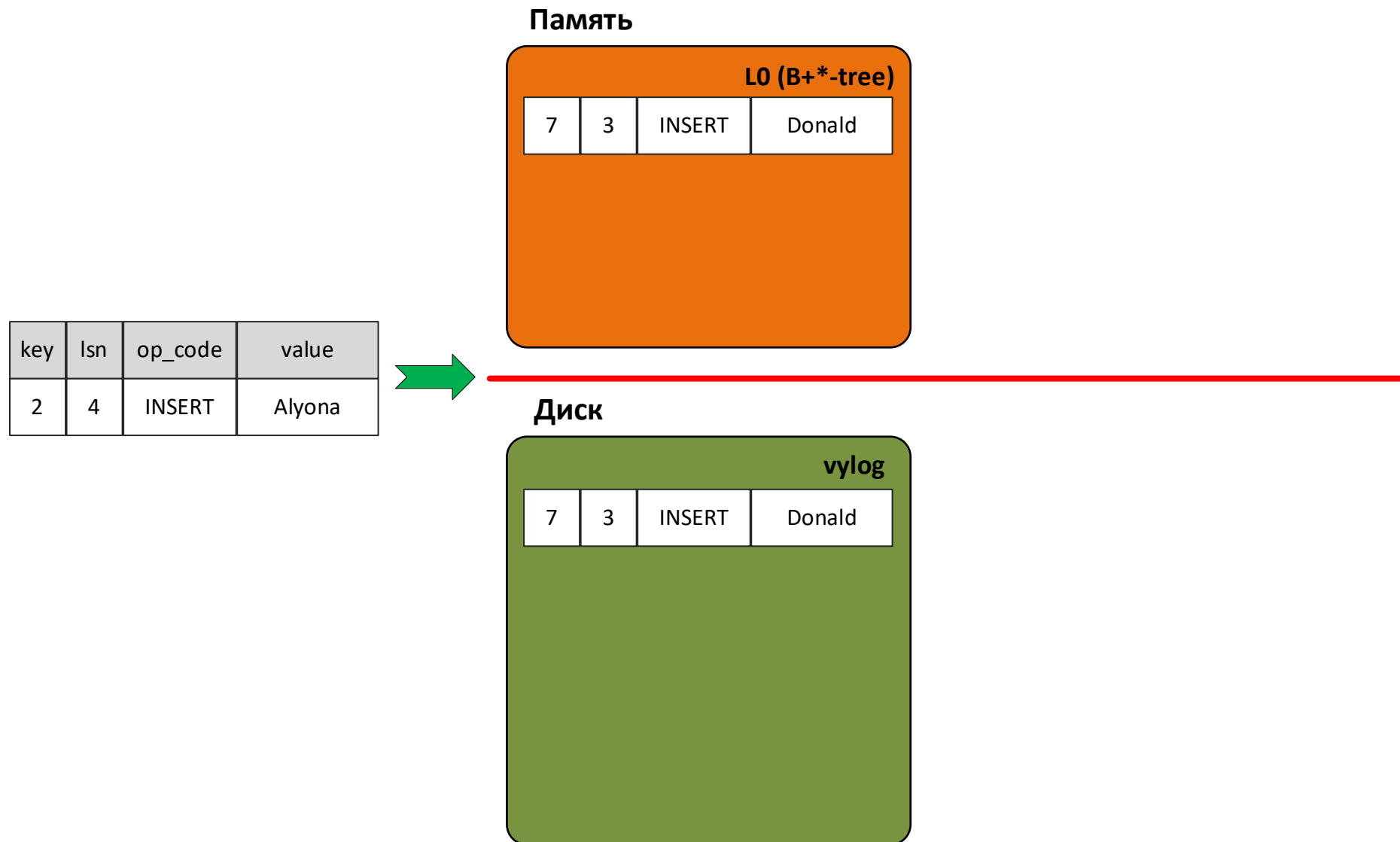
```
box.schema.space.create('users',  
    { engine = 'vinyl' }  
)
```



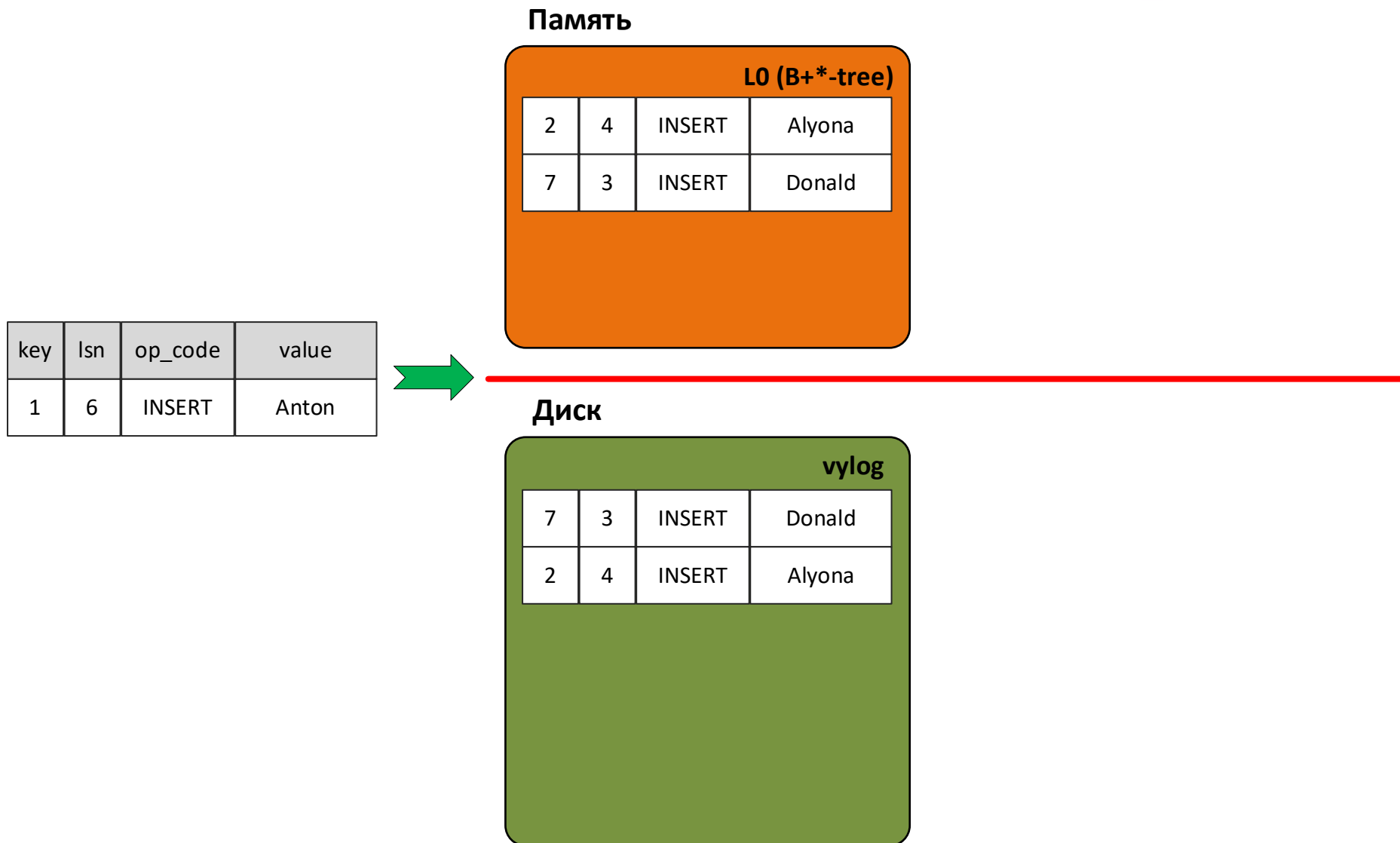
Vinyl. LSM-дерево



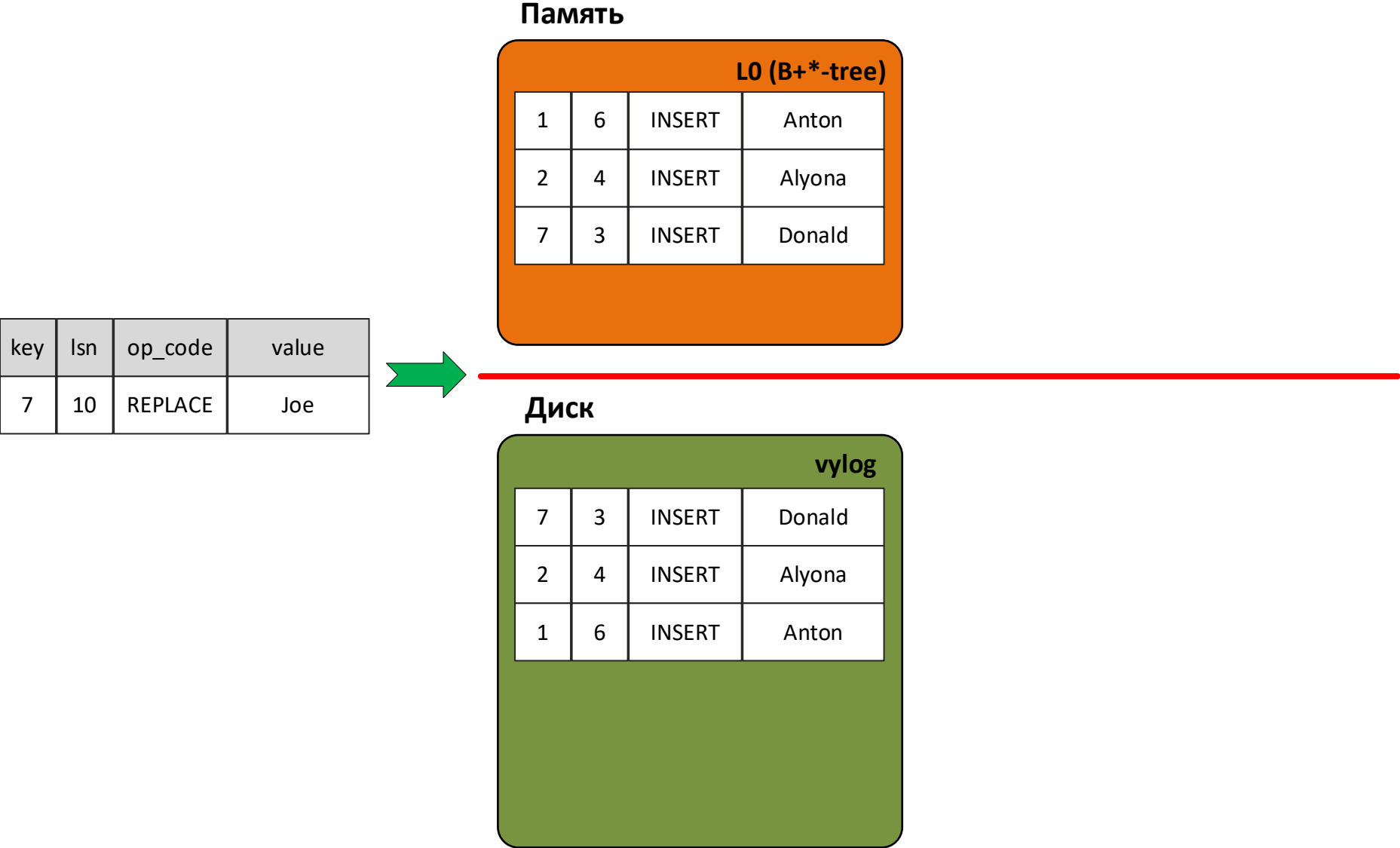
Vinyl. LSM-дерево



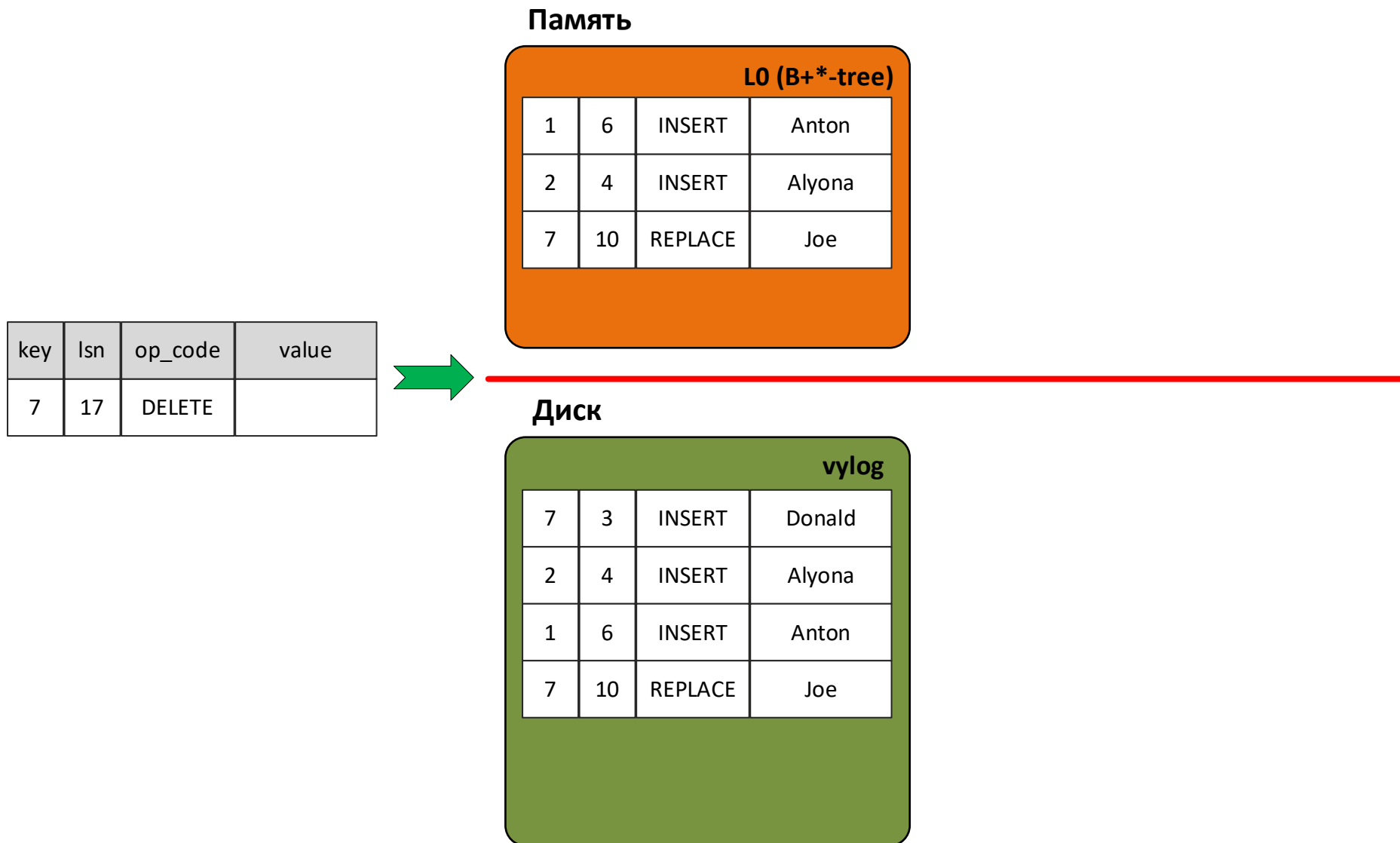
Vinyl. LSM-дерево



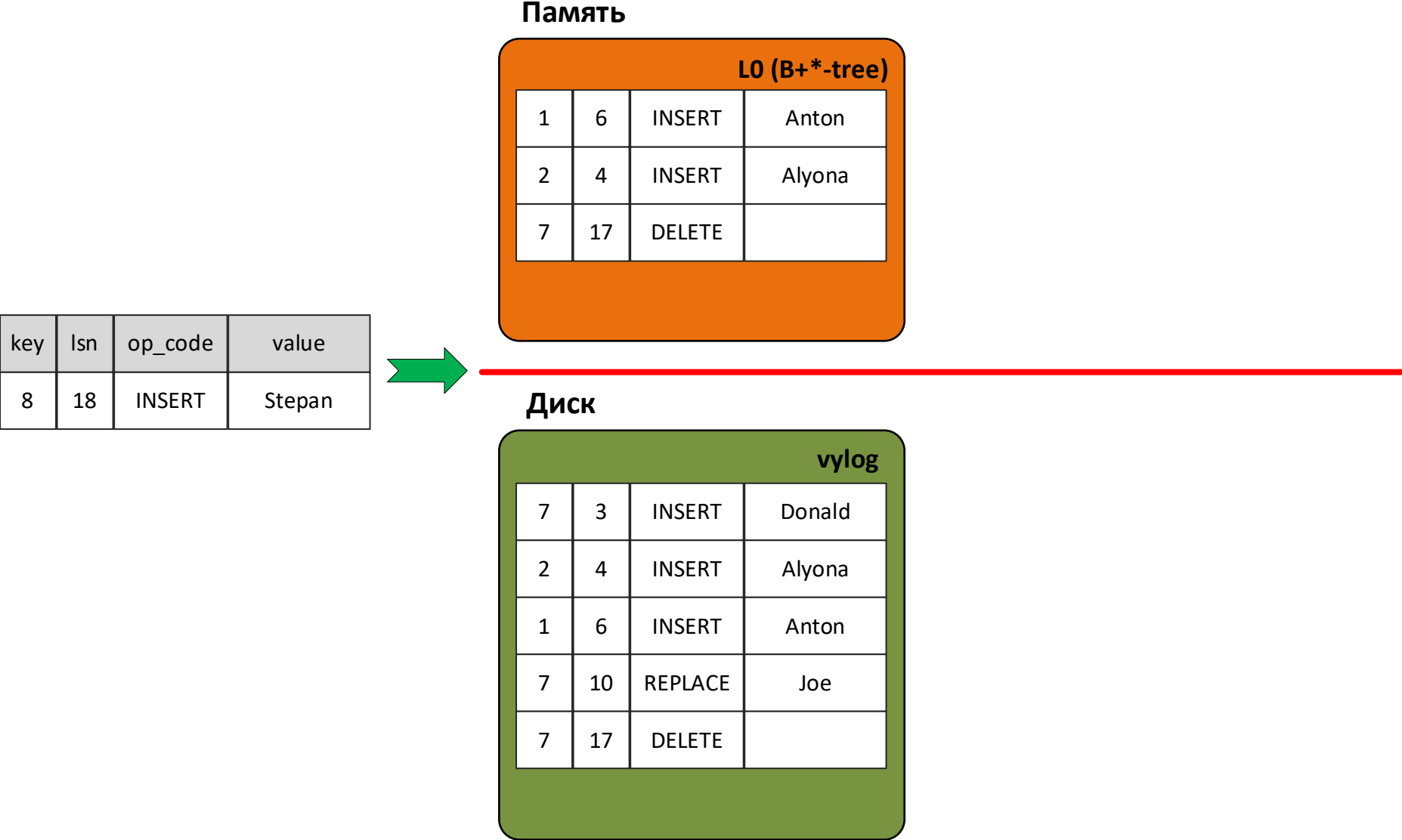
Vinyl. LSM-дерево



Vinyl. LSM-дерево



Vinyl. LSM-дерево



Vinyl. LSM-дерево

Память

L0 (B+*-tree)

1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

Диск

vylog

7	3	INSERT	Donald
2	4	INSERT	Alyona
1	6	INSERT	Anton
7	10	REPLACE	Joe
7	17	DELETE	
8	18	INSERT	Stepan

Vinyl. LSM-дерево

Память

L0 (B+*-tree)			
1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

Диск

vylog			
7	3	INSERT	Donald
2	4	INSERT	Alyona
1	6	INSERT	Anton
7	10	REPLACE	Joe
7	17	DELETE	
8	18	INSERT	Stepan

run			
1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

Vinyl. LSM-дерево

Память

L0 (B+*-tree)			
1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

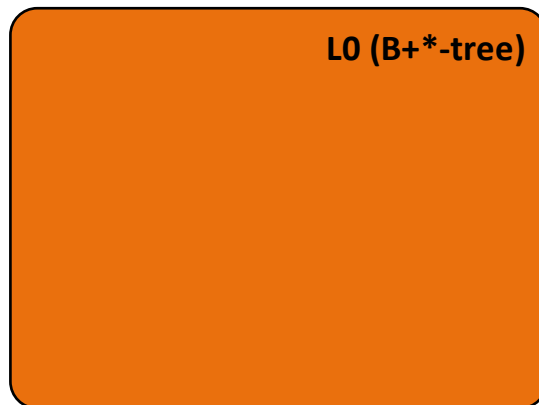
Диск

vylog			
7	3	INSERT	Donald
2	4	INSERT	Alyona
1	6	INSERT	Anton
7	10	INSERT	Joe
7	17	DELETE	
8	18	INSERT	Stepan

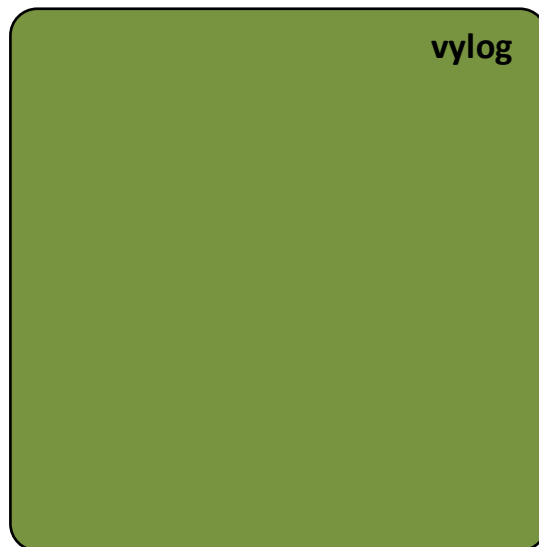
run			
1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

Vinyl. LSM-дерево

Память

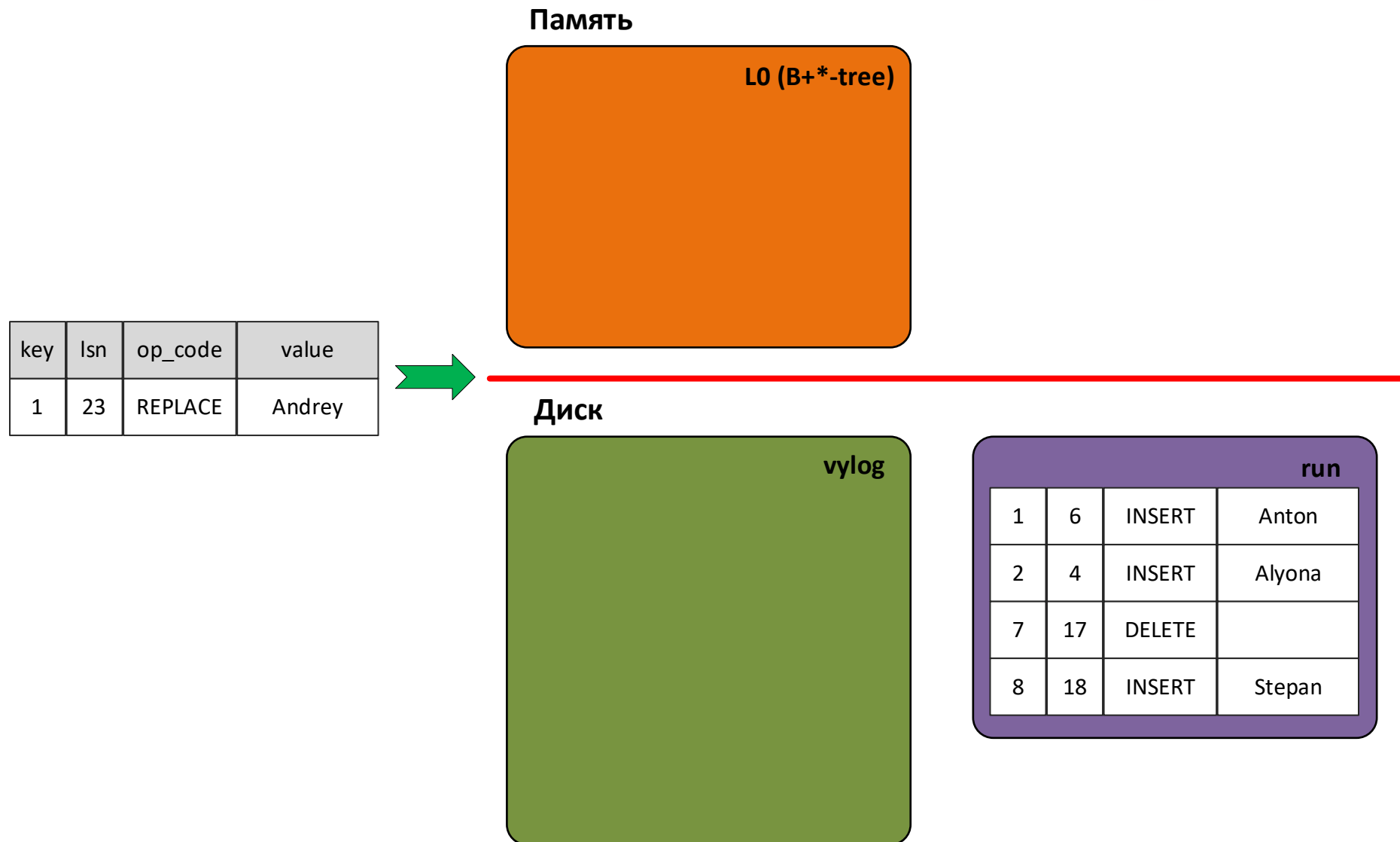


Диск



run			
1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

Vinyl. LSM-дерево



Vinyl. LSM-дерево

Память

L0 (B+*-tree)			
1	23	REPLACE	Andrey

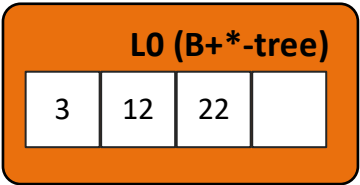
Диск

vylog			
1	23	REPLACE	Andrey

run			
1	6	INSERT	Anton
2	4	INSERT	Alyona
7	17	DELETE	
8	18	INSERT	Stepan

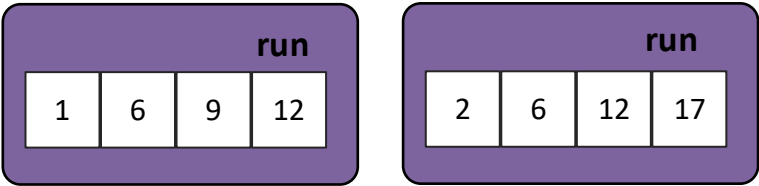
Vinyl. LSM-дерево

Память

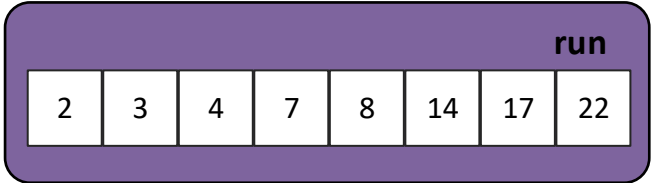


Диск

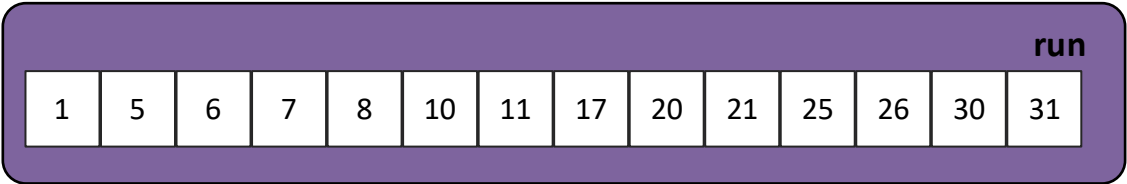
L1



L2

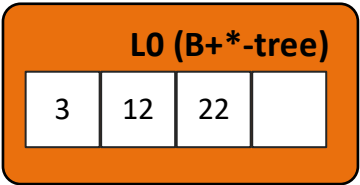


L3



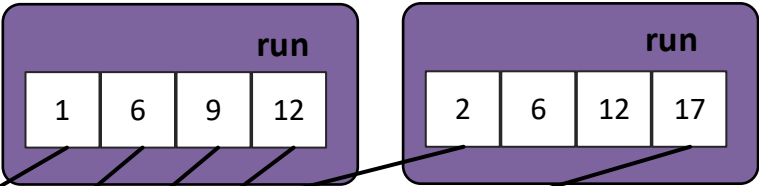
Vinyl. LSM-дерево

Память

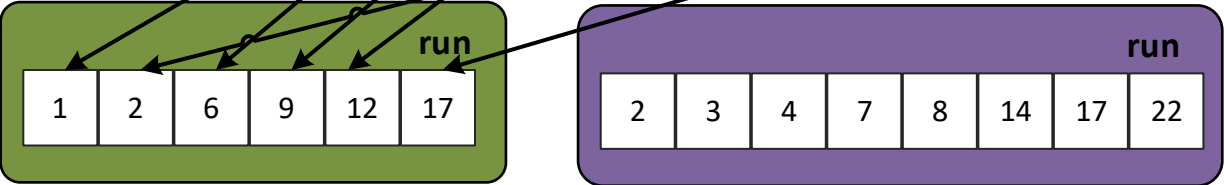


Диск

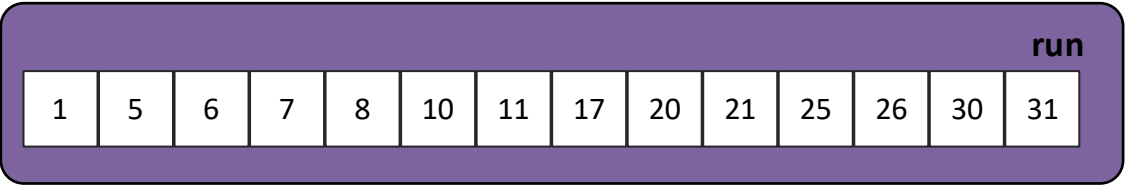
L1



L2

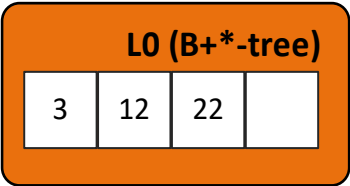


L3



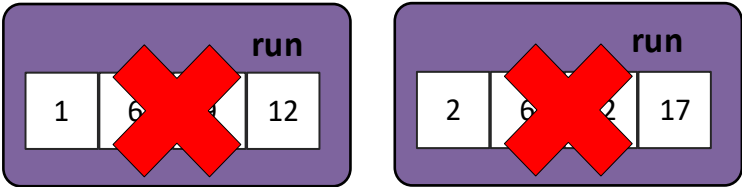
Vinyl. LSM-дерево

Память

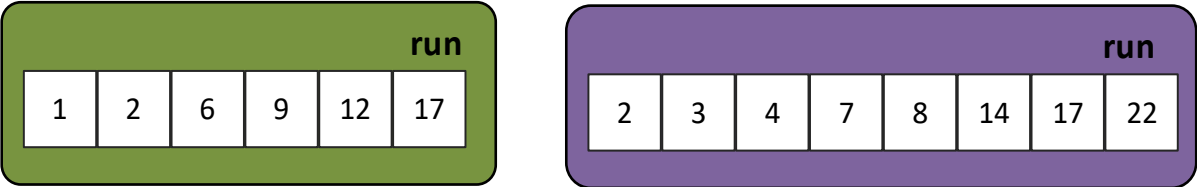


Диск

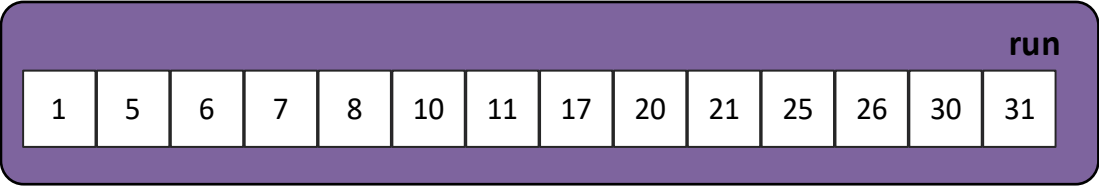
L1



L2

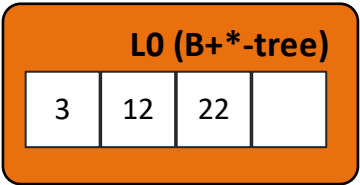


L3



Vinyl. LSM-дерево

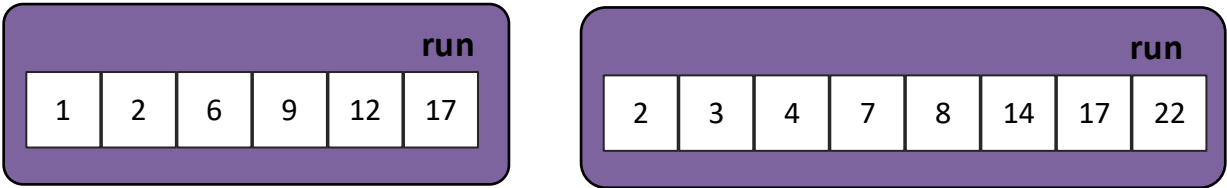
Память



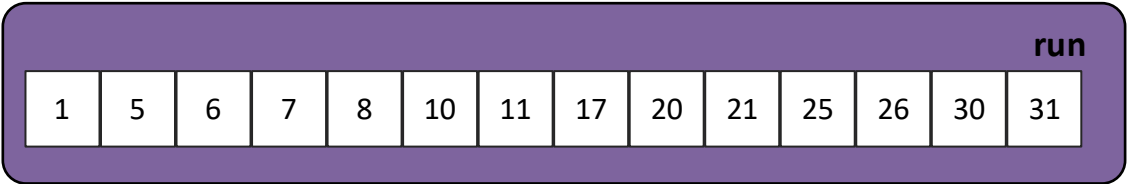
Диск

L1

L2



L3

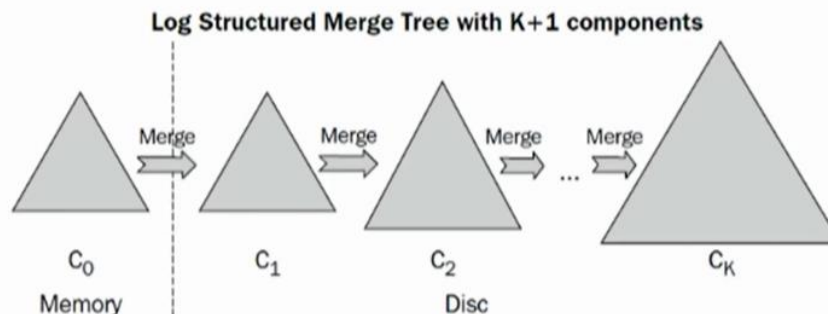


Vinyl. Внутреннее устройство, тюнинг и мониторинг

Внутреннее устройство, тюнинг и мониторинг Tarantool/Vinyl /
Константин Осипов (Tarantool)



Форма классического LSM



@mail.ru
group



@tarantoolconference



@TarantoolConfTalks
@TarantoolConfChannel

<https://www.youtube.com/watch?v=QrNTmBdQMrU>

Свойства LSM-деревьев

Преимущества

- Обычно LSM-деревья, обеспечивают большую пропускную способность на запись, чем B-деревья.
- Усиление записи (write amplification) обычно ниже, чем в B-деревьях.
- LSM-деревья сжимаются лучше, чем B-деревья. Файлы с данными занимают меньше места на диске.

Недостатки

- Один ключ может дублироваться в разных файлах с SSTable.
- Непредсказуемая скорость чтения.
- Скорость ответа зависит от фоновых задач.

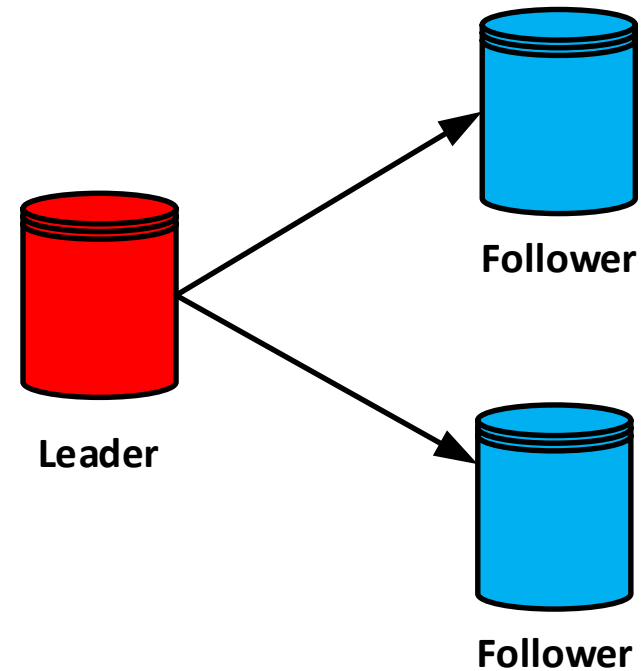
Репликация

Репликация - поддержание актуальной копии данных на нескольких узлах.

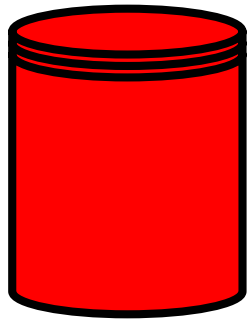
Репликационная группа или **репликaset** - множество узлов, между которыми настроена репликация данных

Задачи репликации:

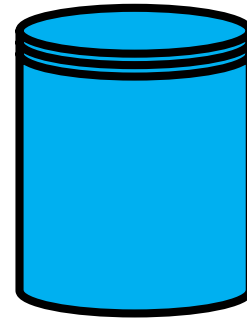
- Резервирование данных
- Балансировка нагрузки (обычно на чтение)



Асинхронная репликация

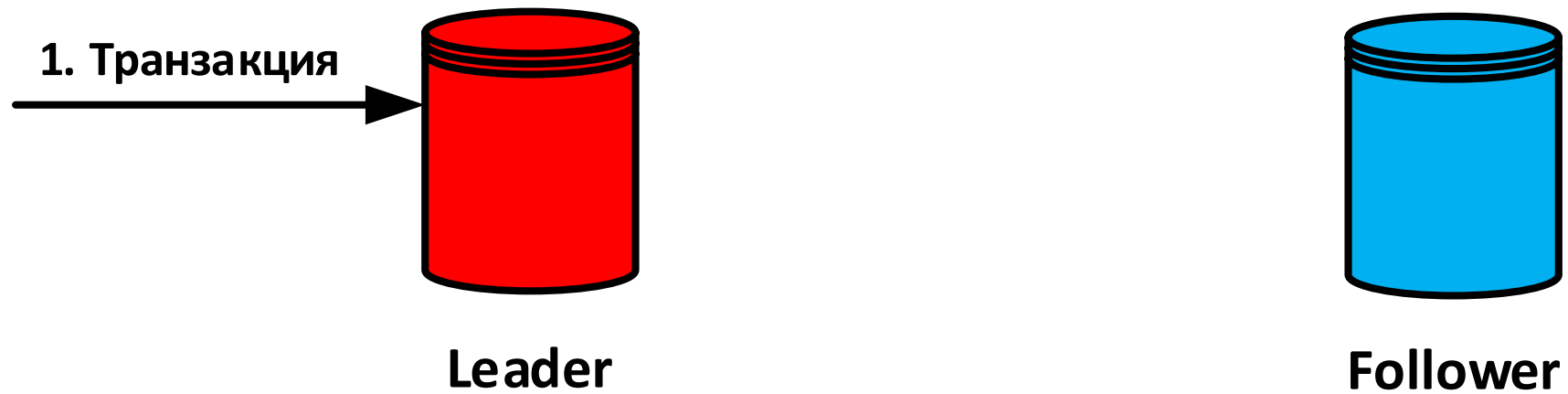


Leader

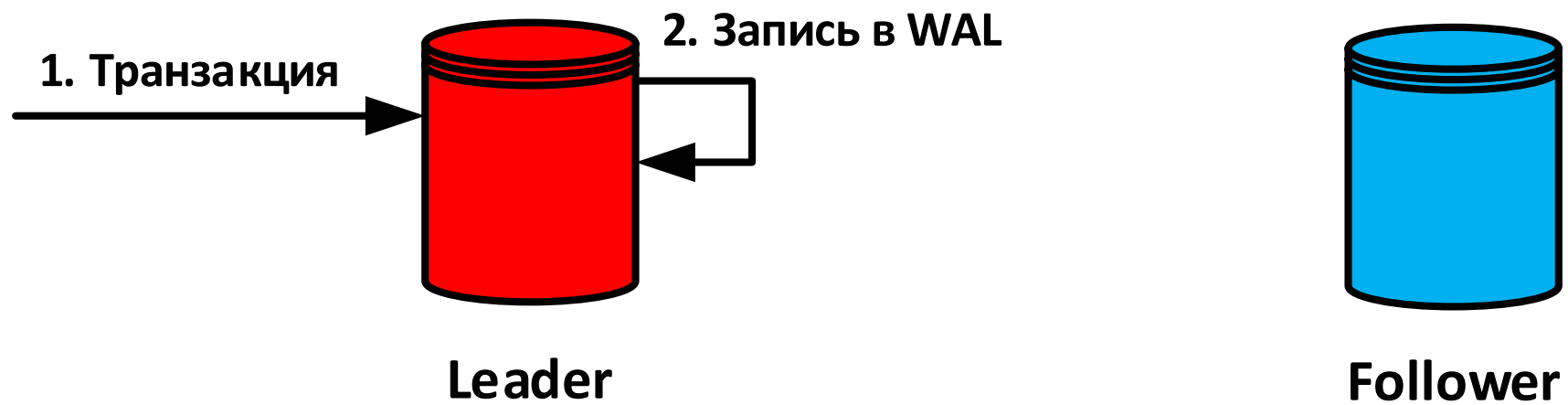


Follower

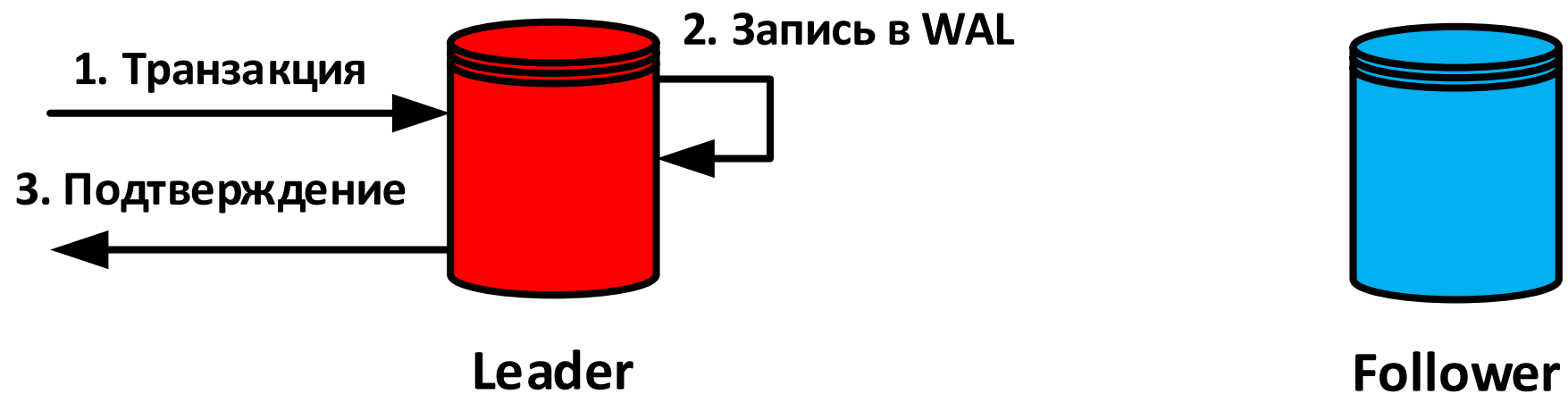
Асинхронная репликация



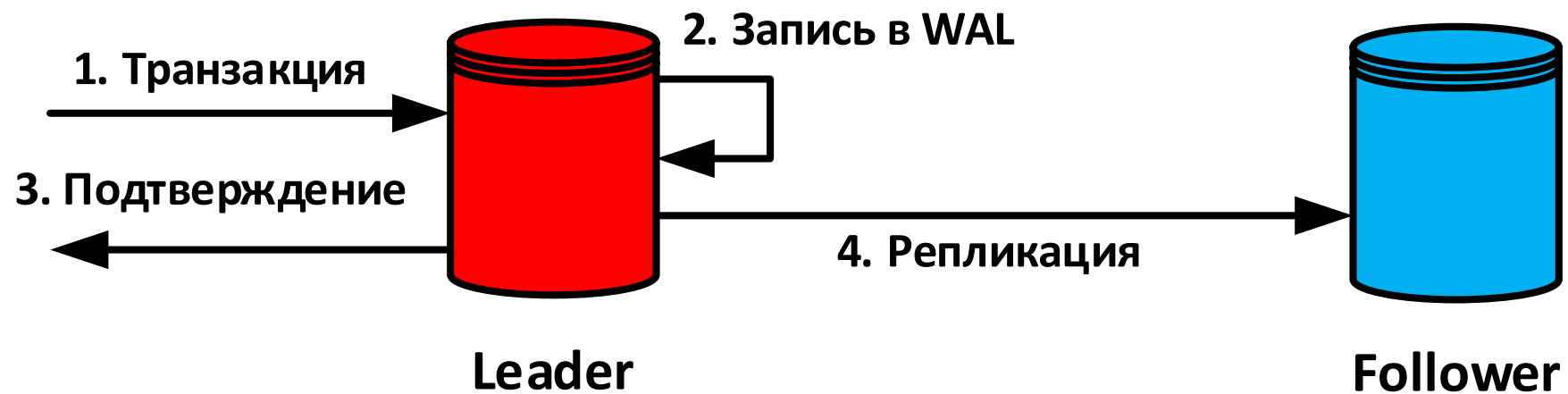
Асинхронная репликация



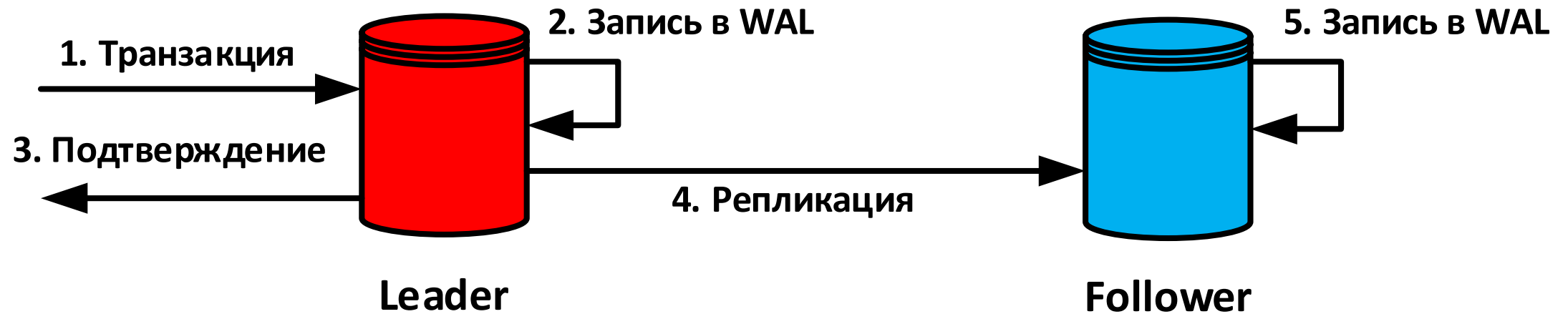
Асинхронная репликация



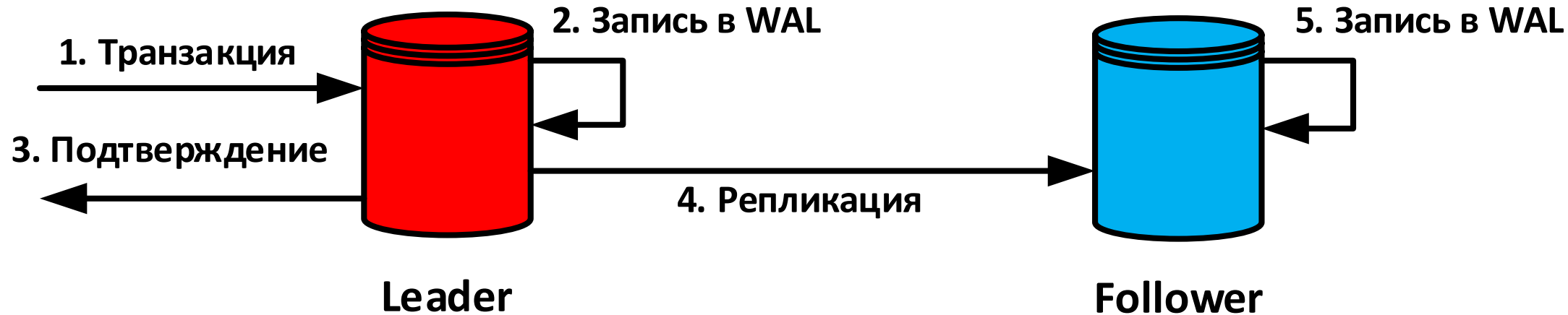
Асинхронная репликация



Асинхронная репликация



Асинхронная репликация. Плюсы и минусы



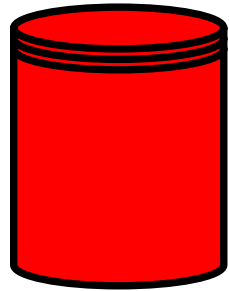
Преимущества

- Быстрый ответ, не ждём подтверждения от реплик
- Высокая доступность
- Легкая конфигурация

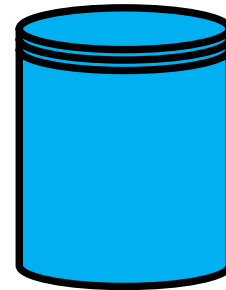
Недостатки

- Возможность потери данных. Например, Leader может умереть после шага 3.
- Возможность грязных чтений.

Синхронная репликация

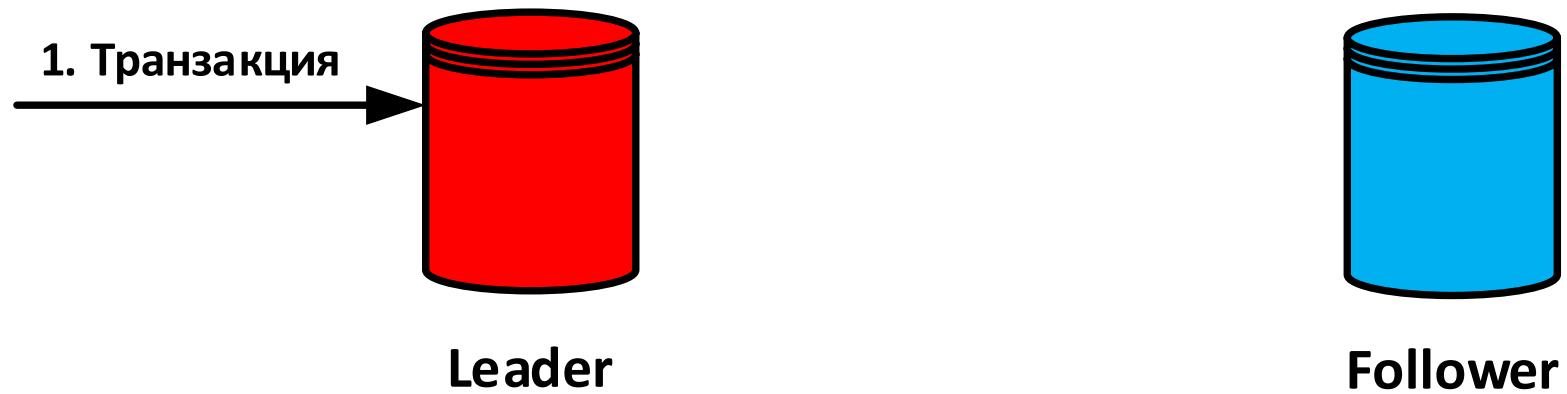


Leader

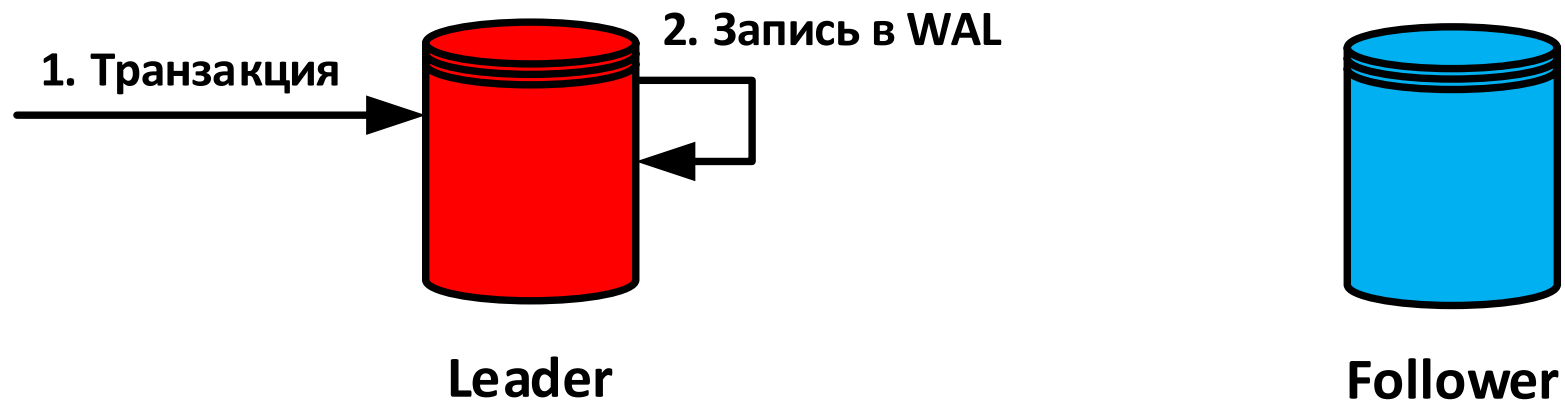


Follower

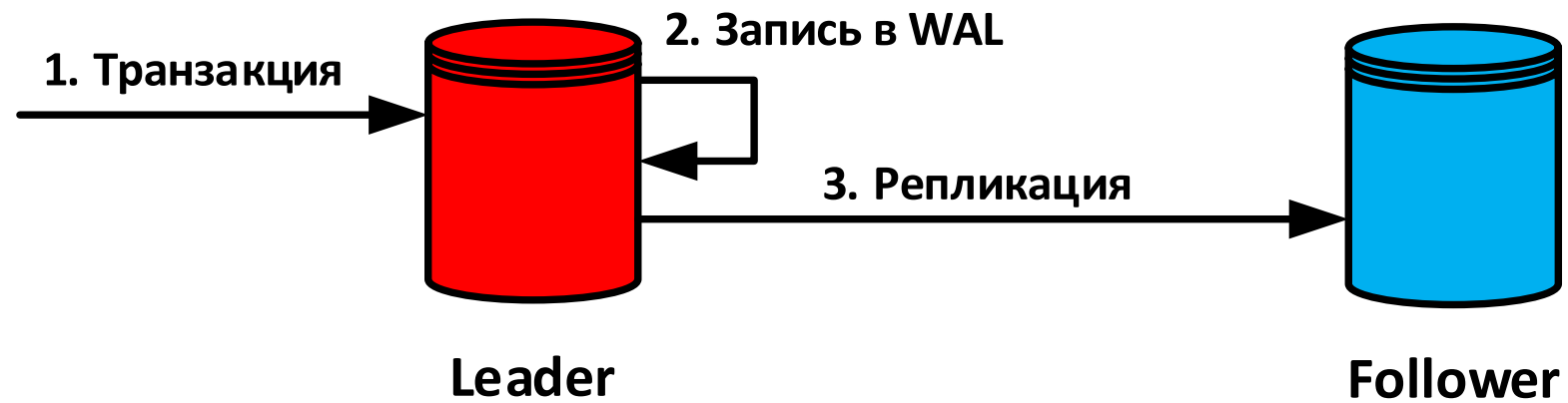
Синхронная репликация



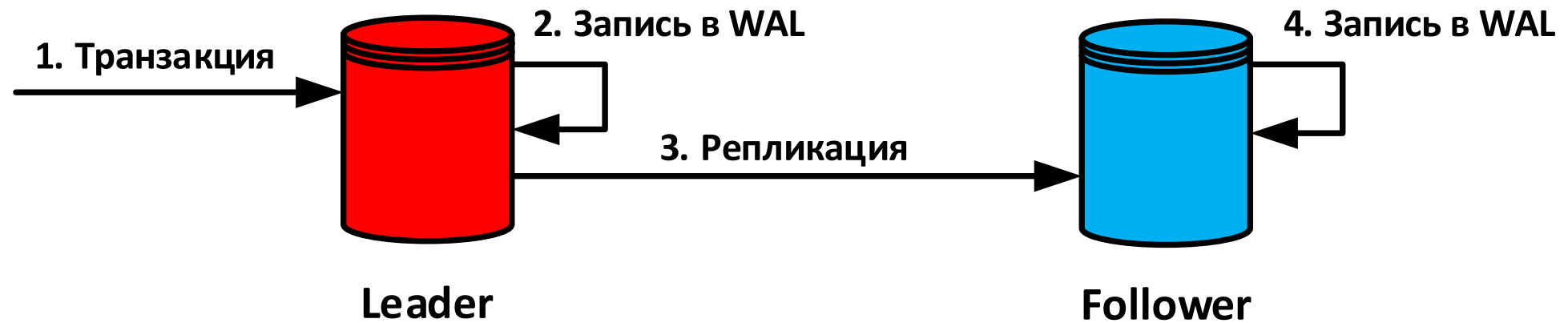
Синхронная репликация



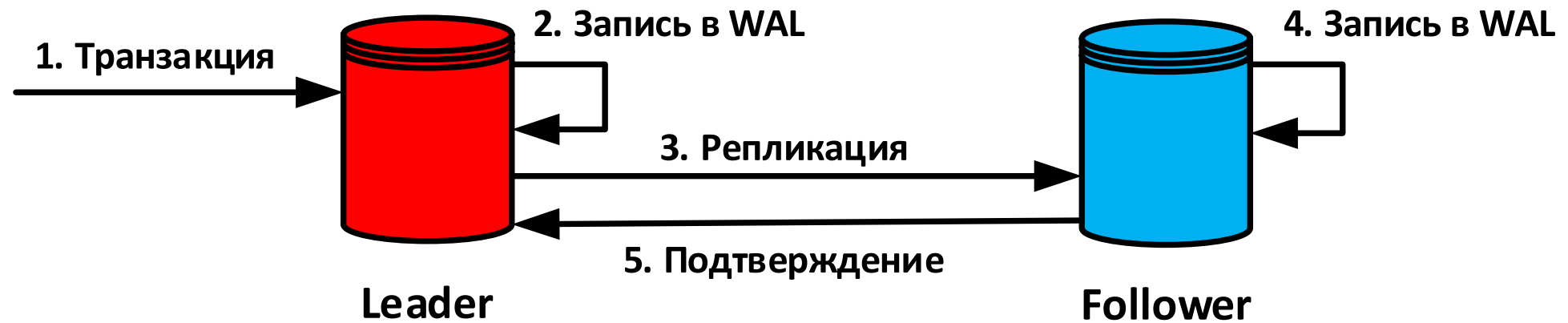
Синхронная репликация



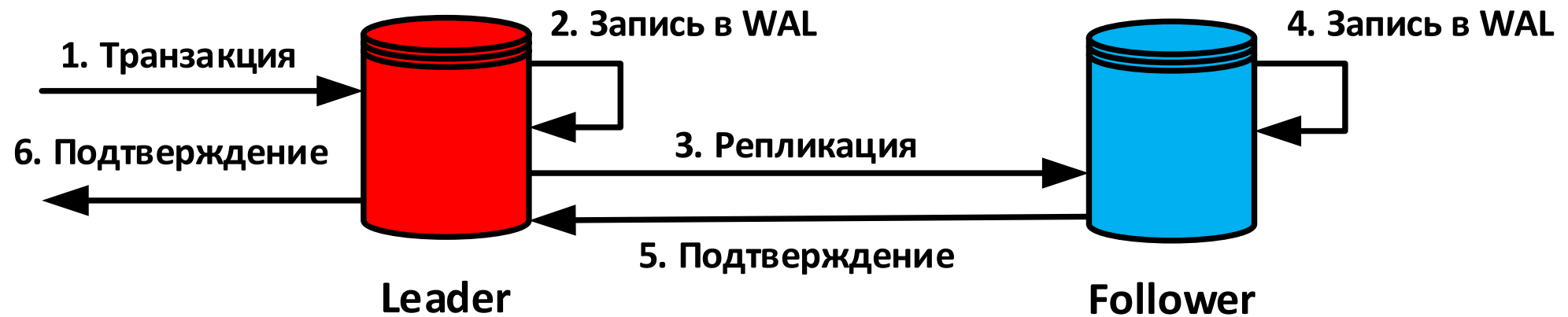
Синхронная репликация



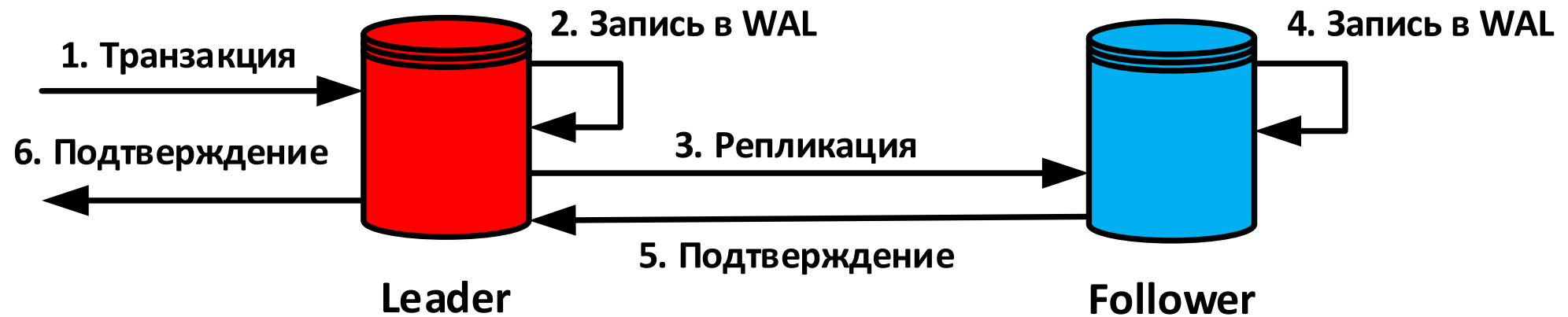
Синхронная репликация



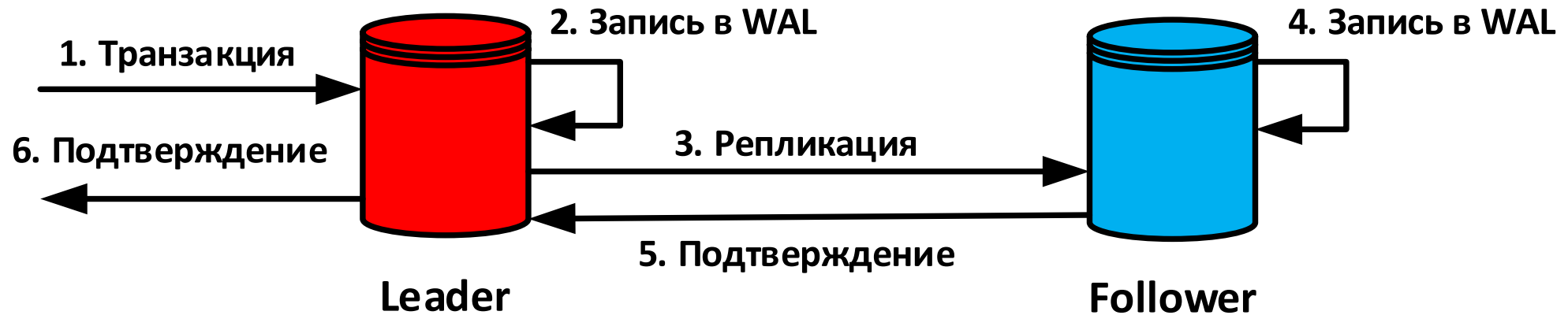
Синхронная репликация



Синхронная репликация



Синхронная репликация. Плюсы и минусы



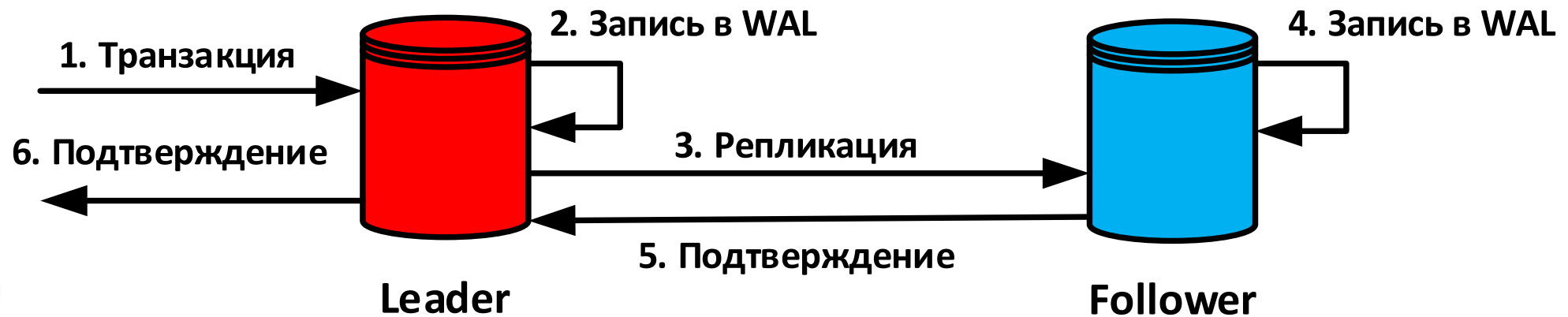
Преимущества

- Высокая надёжность. Данные сложнее потерять, чем при асинхронной репликации

Недостатки

- Низкая скорость (есть «тяжелые» походы по сети; ждём подтверждения от реплик)
- Низкая доступность (лидер может не набрать кворума для коммита транзакции)
- Сложность конфигурирования

Синхронная репликация в Tarantool. Алгоритм Raft



Raft - алгоритм синхронной репликации, реализован в Tarantool в 2020 году.

```
box.schema.space.create('users',  
    { is_sync = true }  
)
```



Tarantool 2.6 - Изобретая синхронную репликацию


Репликация [6]: кворум

Типичные конфигурации

1 + 1

Тройка

50% + 1

 **tarantool** Владислав Шпилевой
developed by @mail.ru group ведущий разработчик

https://www.youtube.com/watch?v=sOZgUQyPM_g

Шардирование данных в Tarantool

Шардирование - масштабирование БД по данным, данные распределяются по узлам кластера. В Tarantool используется шардирование по хешам, когда от каждой записи считается хеш, и записи с одинаковым хешем сохраняются на одних и тех же узлах кластера.

Основные реализации шардинга в Tarantool:

- **Tarantool Shard** - <https://github.com/tarantool/shard>

Шардирование по хешам от первичных ключей записей

Недостатки: сложно хранить связанные данные вместе; медленный решардинг; нестабильные чтения при отказах нескольких реплик в репликасете

- **Tarantool Vshard** - <https://github.com/tarantool/vshard>

Используются виртуальные хранилища (bucket) поверх реальных.

Bucket'ы переносятся между узлами атомарно.

Реализуется локальность данных по bucket_id.

Шардирование данных в Tarantool - Vshard

```
local users = box.schema.space.create('users',
    { if_not_exists = true }
)

users:format({
    { 'bucket_id', 'unsigned' },
    { 'uuid', 'string' },
    { 'login', 'string' },
    { 'password', 'string' },
    { 'status', 'string' }
})
```

Шардирование данных в Tarantool - Vshard

```
local users = box.schema.space.create('users',
    { if_not_exists = true }
)

users:format({
    { 'bucket_id', 'unsigned' },
    { 'uuid', 'string' },
    { 'login', 'string' },
    { 'password', 'string' },
    { 'status', 'string' }
})
```

```
function create_user(user_uuid, login, password_hash, status, groups)
    local bucket_id = vshard.router.bucket_id(login);

    local _, err = vshard.router.callrw(bucket_id, 'box.space.users:insert', {
        { bucket_id, user_uuid, login, password_hash, status }
    })

    insert_user_groups(bucket_id, user_uuid, groups)

    return user_uuid
end
```

Шардирование данных в Tarantool - Vshard

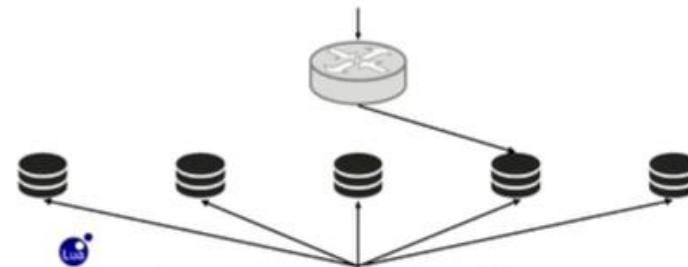
VShard — горизонтальное масштабирование в Tarantool

Владислав Шпилевой (Tarantool)



Прокси

```
r = vshard.router
accounts = r.call(bucket_id,
  'get_accounts',
  {customer_id})
```



```
function get_accounts(customer_id)
  local customer =
    box.space.accounts.index.customer
  return customer:select({customer_id})
end
```

HighLoad++

Как построить кластер Tarantool?

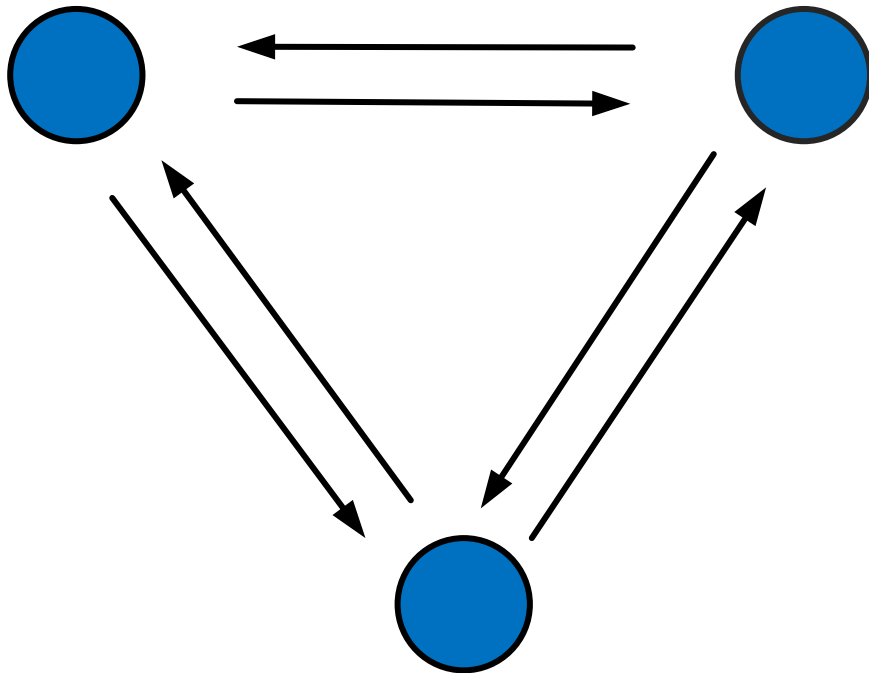
Требования

- Необходимо уметь обнаруживать отказы узлов (failure detection) в кластере для возможности принятия согласованных решений

Как построить кластер Tarantool?

Требования

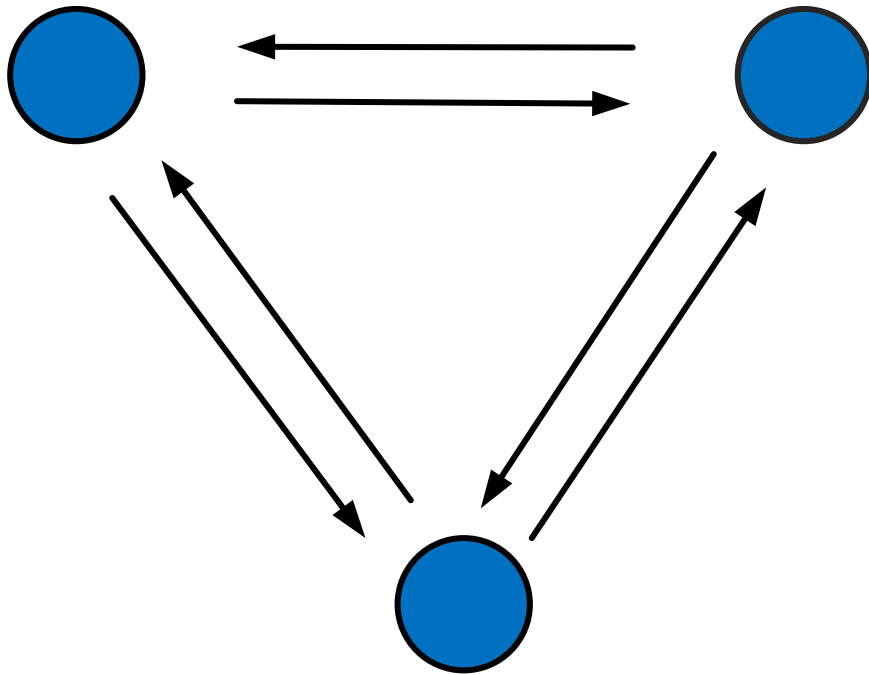
- Необходимо уметь обнаруживать отказы узлов (failure detection) в кластере для возможности принятия согласованных решений



Как построить кластер Tarantool?

Требования

- Необходимо уметь обнаруживать отказы узлов (failure detection) в кластере для возможности принятия согласованных решений

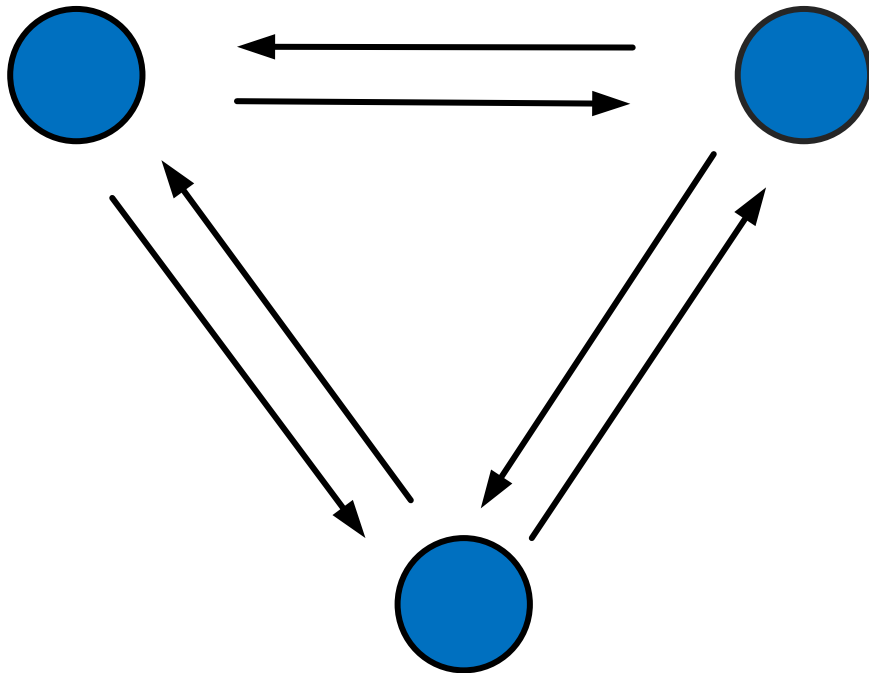


При N узлах будет $O(N^2)$ сообщений в сети.
Это не работает на больших кластерах.

Как построить кластер Tarantool?

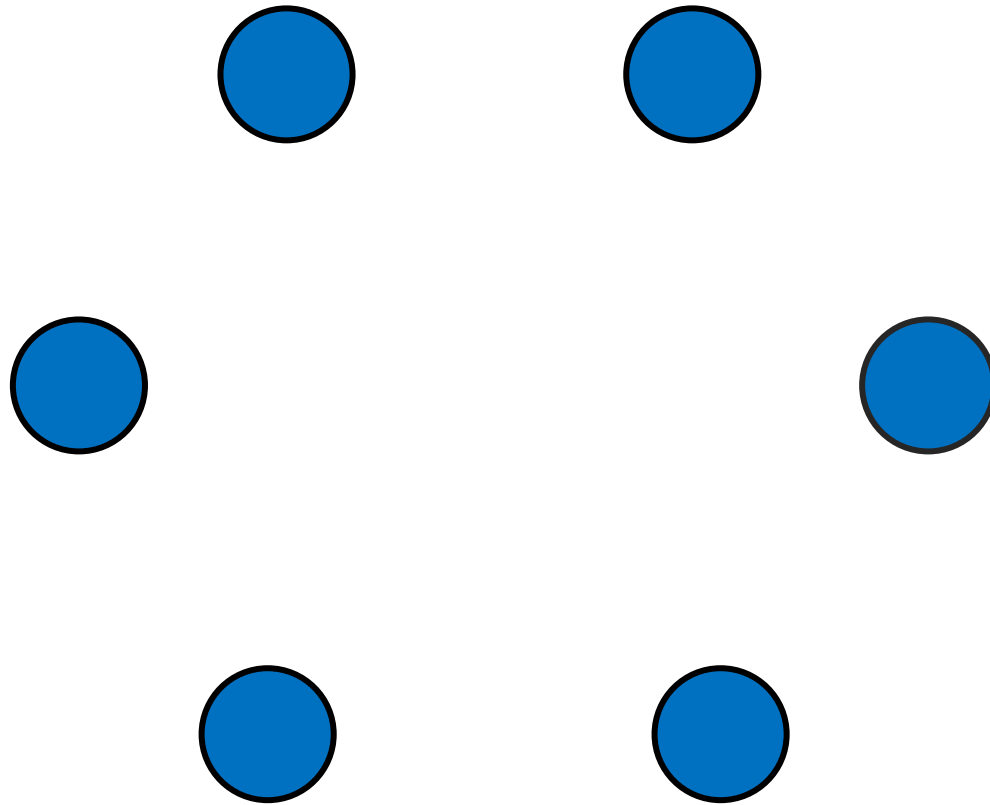
Требования

- Необходимо уметь обнаруживать отказы узлов (failure detection) в кластере для возможности принятия согласованных решений
- Необходимо делать это эффективно

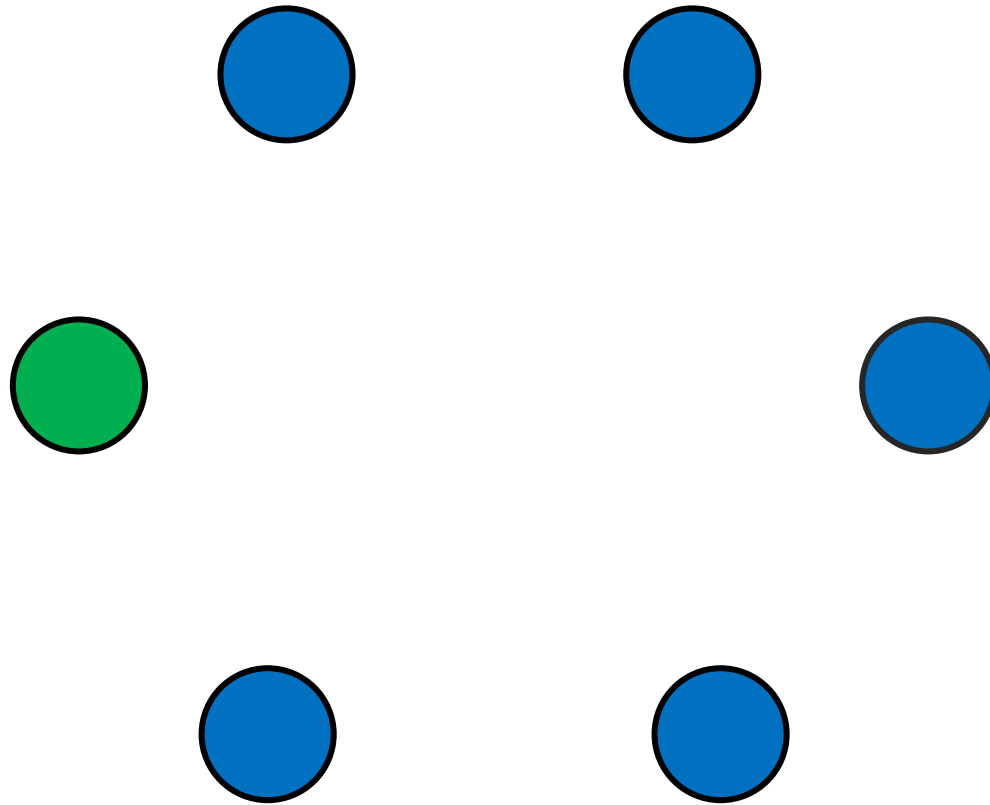


При N узлах будет $O(N^2)$ сообщений в сети.
Это не работает на больших кластерах.

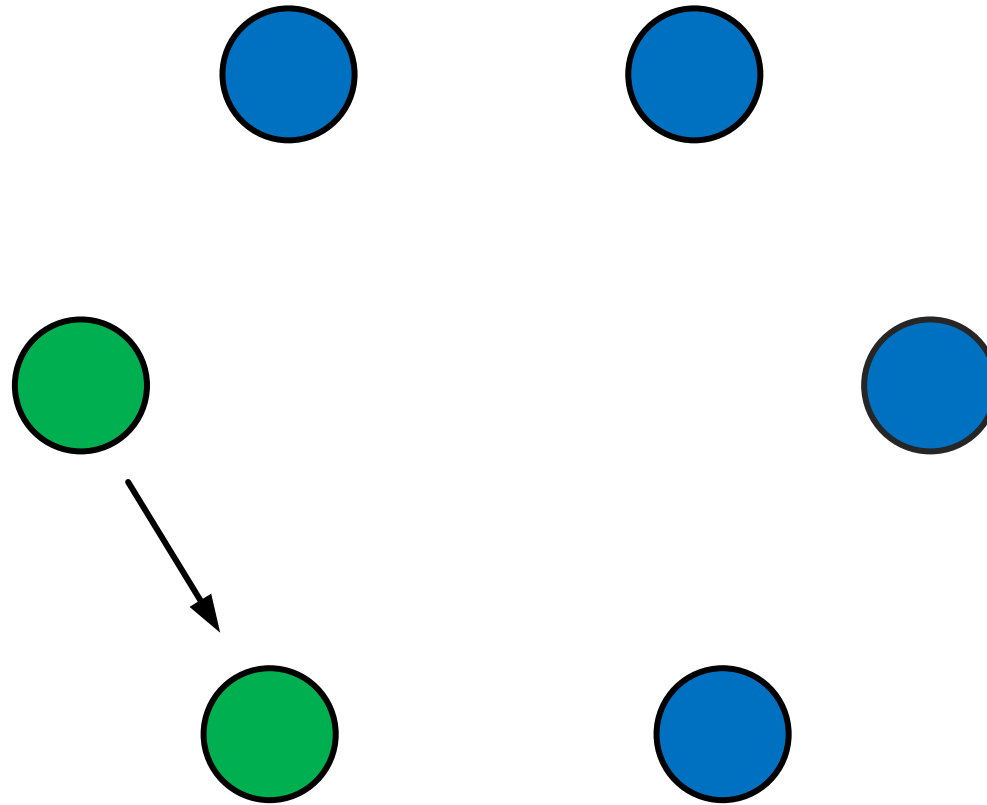
Gossip-протокол (протокол «слухов», «инфекций»)



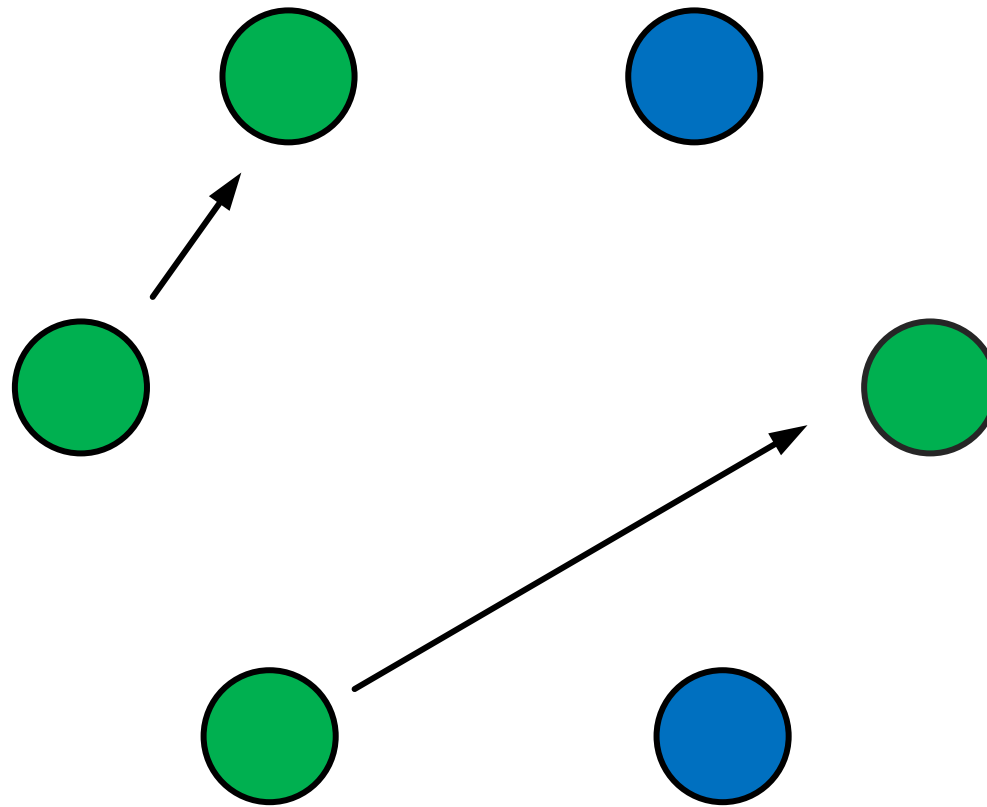
Gossip-протокол (протокол «слухов», «инфекций»)



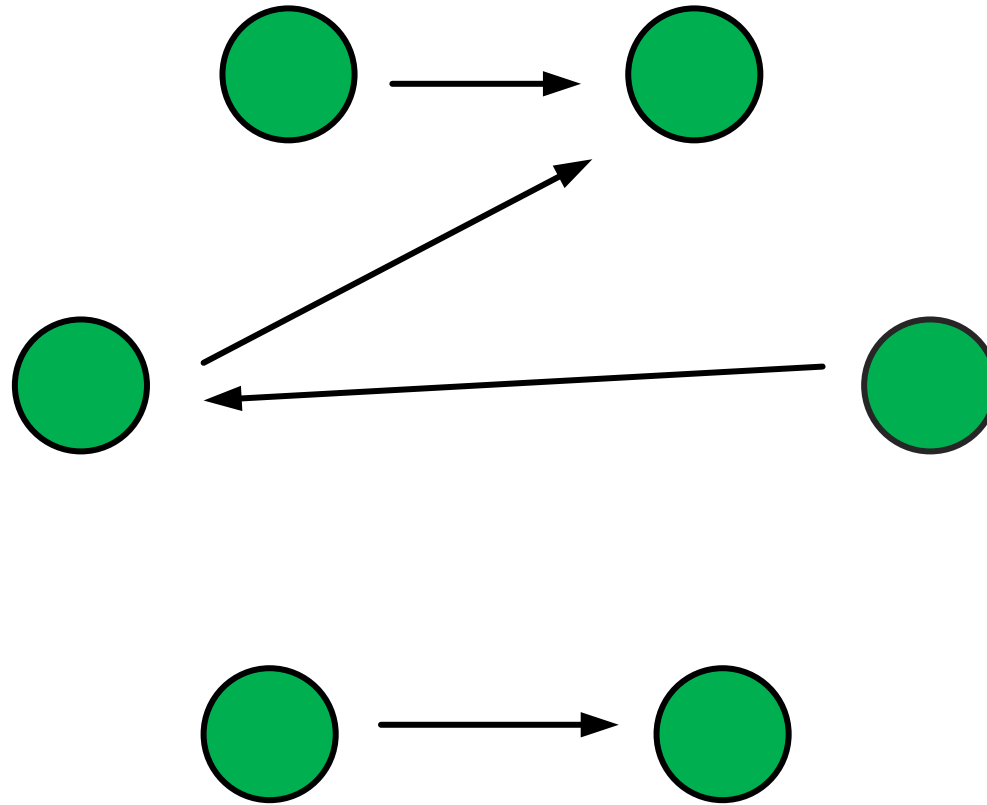
Gossip-протокол (протокол «слухов», «инфекций»)



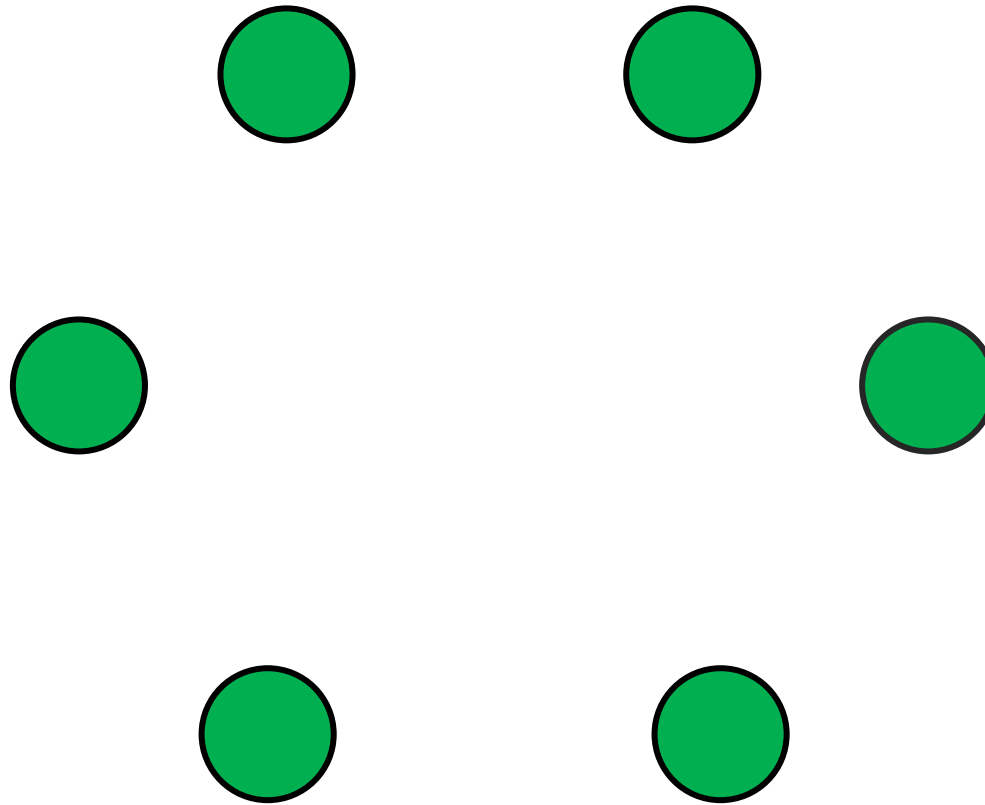
Gossip-протокол (протокол «слухов», «инфекций»)



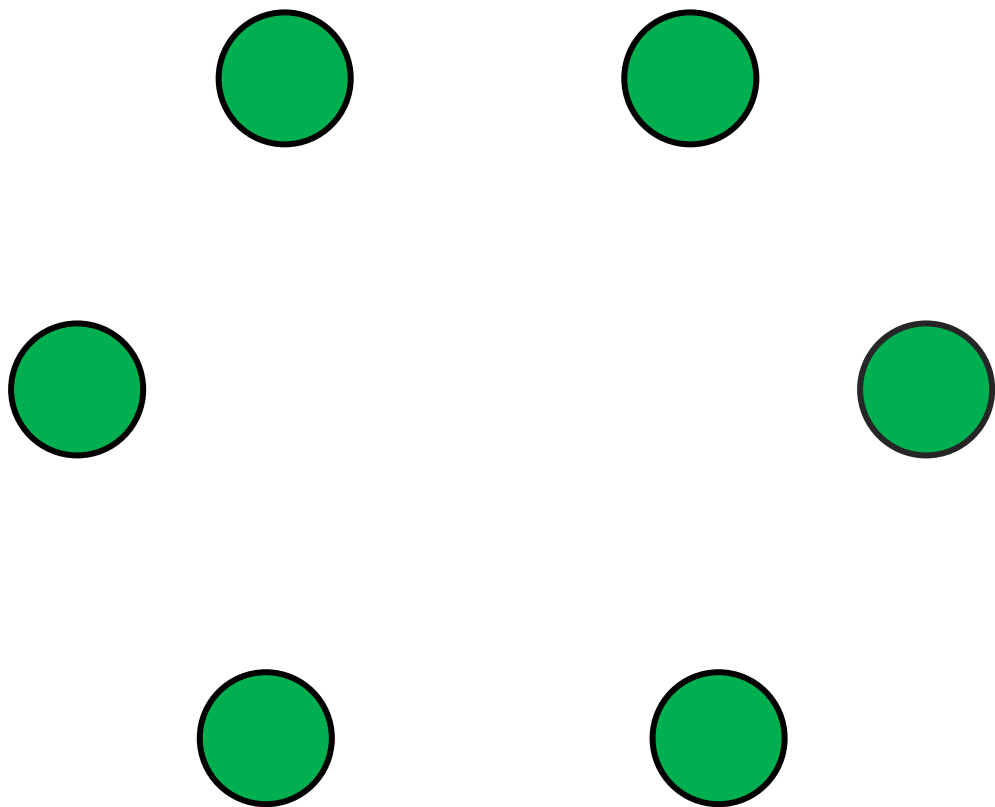
Gossip-протокол (протокол «слухов», «инфекций»)



Gossip-протокол (протокол «слухов», «инфекций»)



Gossip-протокол (протокол «слухов», «инфекций»)



При N узлах будет $O(N)$ сообщений в сети.
Это не работает на больших кластерах.

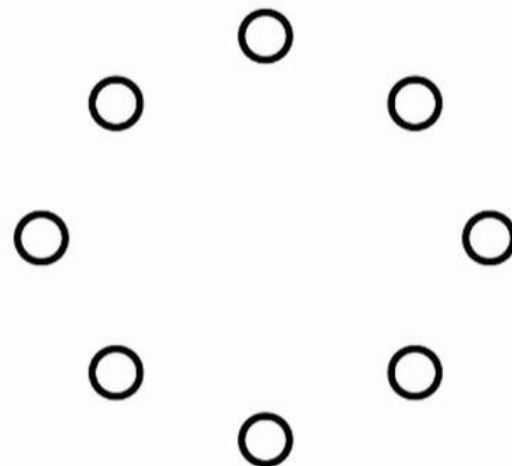
Событие распространится по кластеру за $O(\log N)$.

SWIM в Tarantool



SWIM

Scalable
Weakly-Consistent
Infection-Style Process
Group Membership

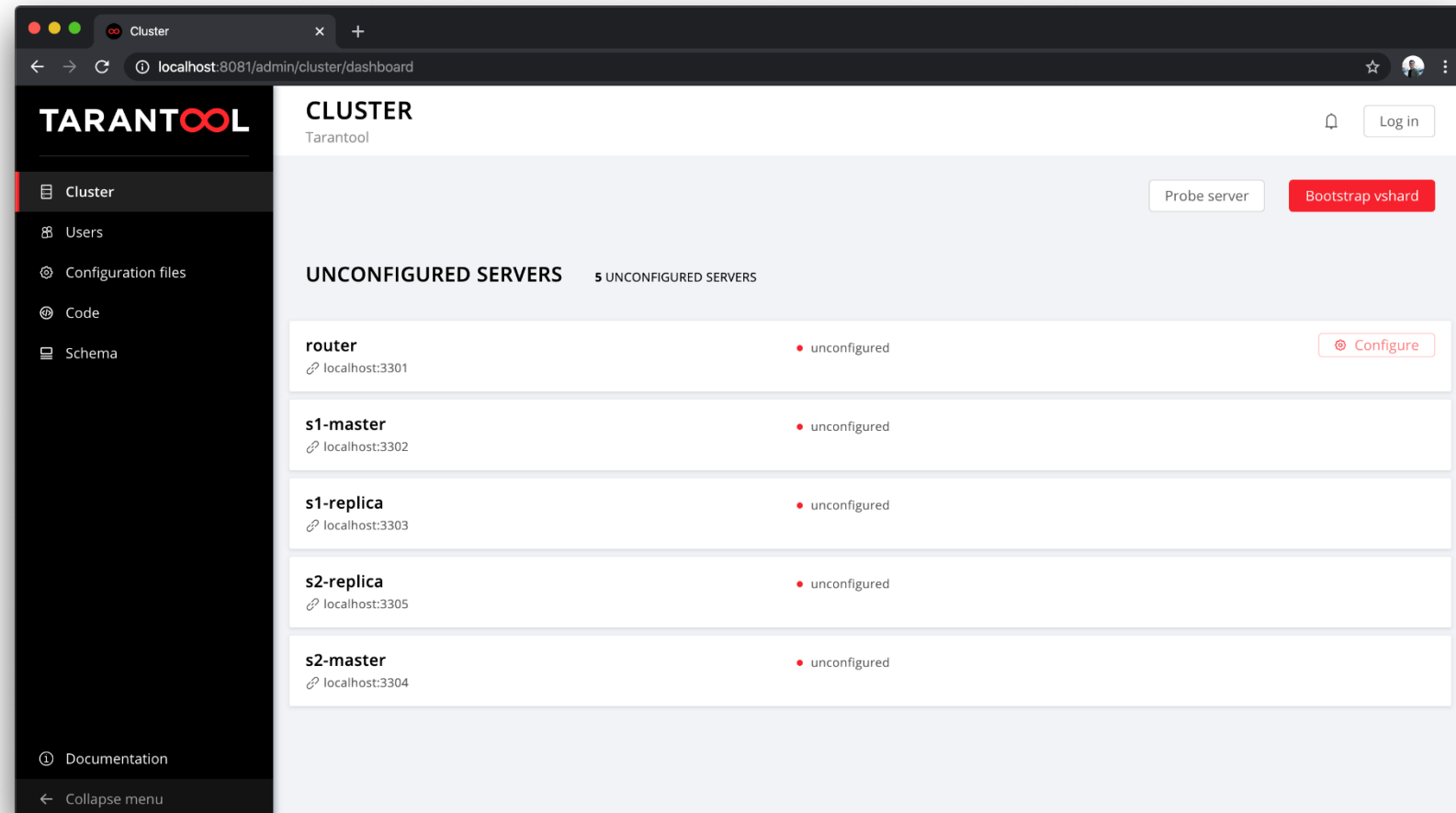


SWIM - протокол построения кластера
Владислав Шпилевой (Tarantool)

<https://www.youtube.com/watch?v=j9dXukoHDvs>

Tarantool Cartridge

```
cartridge create --name myapp  
cd myapp  
cartridge build  
cartridge start
```



Tarantool сегодня

TARANTOOL		
Sharding	☑	Tarantool VShard
Replication	☑	Synchronous/Asynchronous
In-memory	☑	memtx engine
Disk	☑	vinyl engine, LSM-tree
Persistency	☑	Both engines
SQL	☑	ANSI
Stored procedures	☑	Lua, C, SQL
Audit logging	☑	Yes
Connectors to DBMSes	☑	MySQL, Oracle, Memcached
Static build	☑	for Linux
GUI	☑	Cluster management
Unprecedented performance	☑	100.000 RPS per instance - easy!

Центр аутентификации 1.0.0 OAS3

API демонстрационного приложения "Центр аутентификации"

Authentication

POST `/login` Аутентификация пользователя по логину и паролю

User

GET `/user/{uuid}` Получение информации об учётной записи пользователя

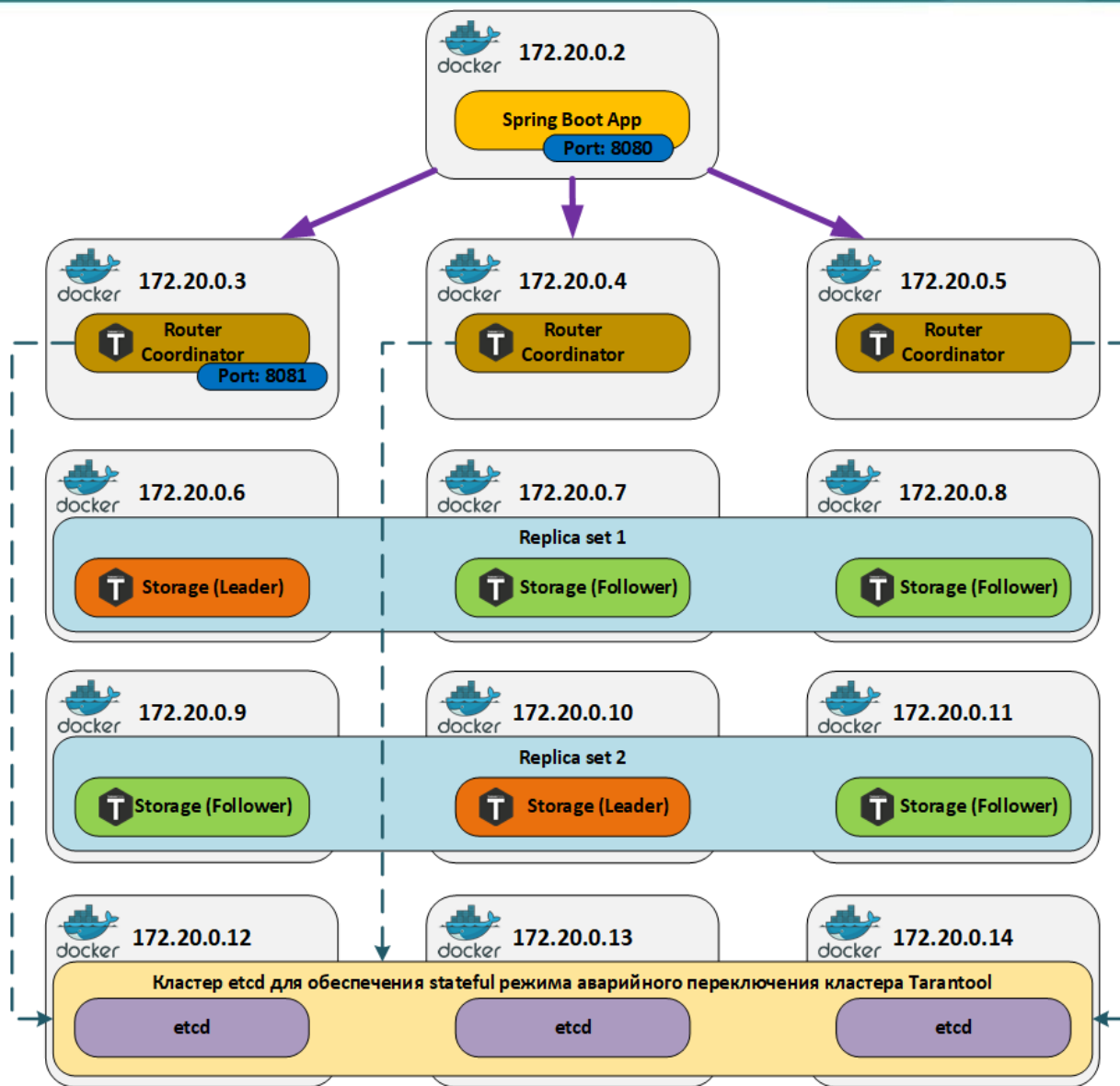
PUT `/user/{uuid}` Обновление учётной записи пользователя

DELETE `/user/{uuid}` Удаление учетной записи пользователя

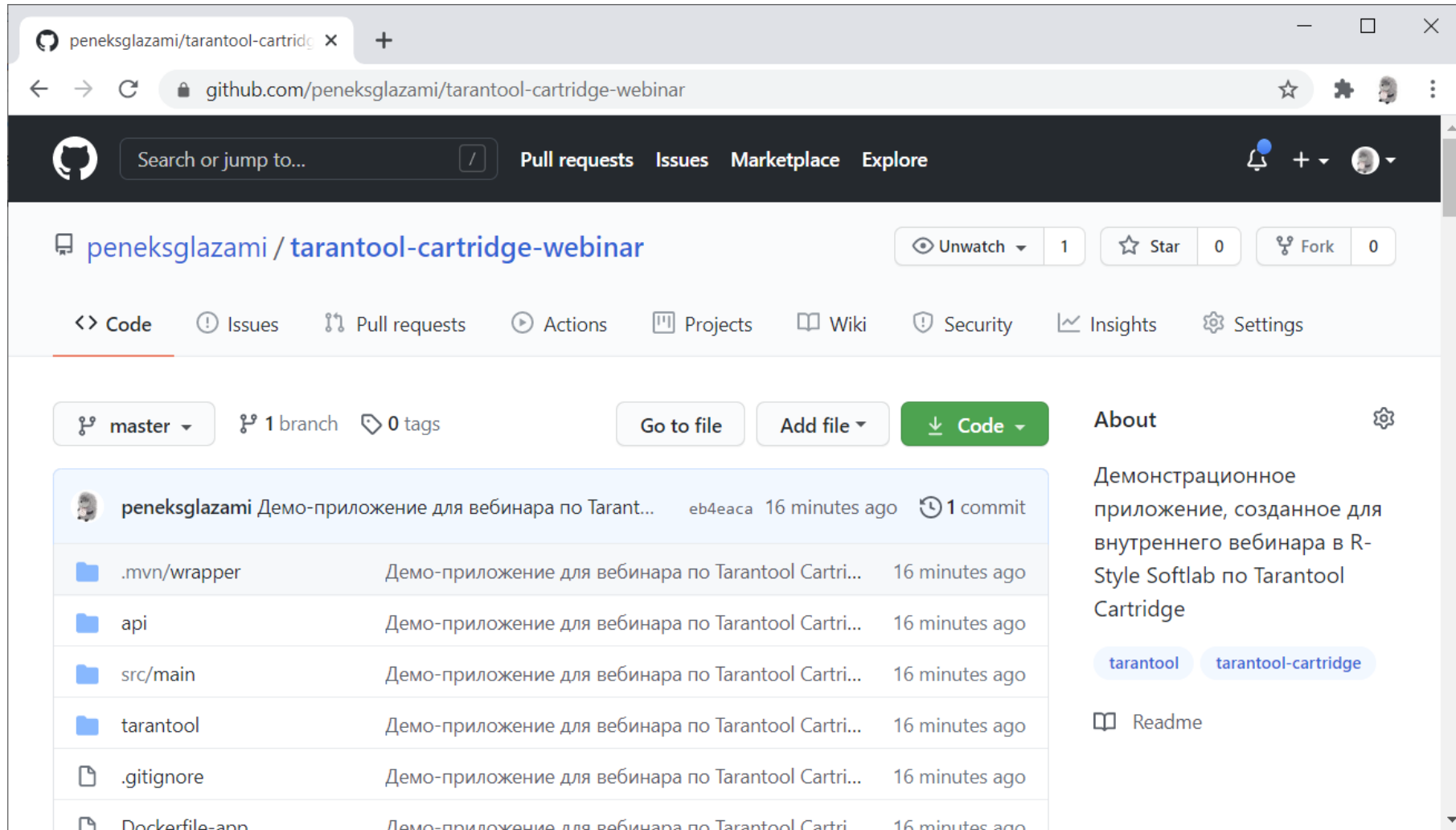
POST `/user` Создание учетной записи пользователя

PUT `/user/update-group` Обновление группы учётных записей клиентов

Демо-приложение. Разбиение приложений по контейнерам



Время переходить в IDE

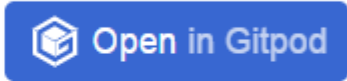



<https://github.com/peneksglazami/tarantool-cartridge-webinar>

Как посмотреть демо-приложение в Gitpod

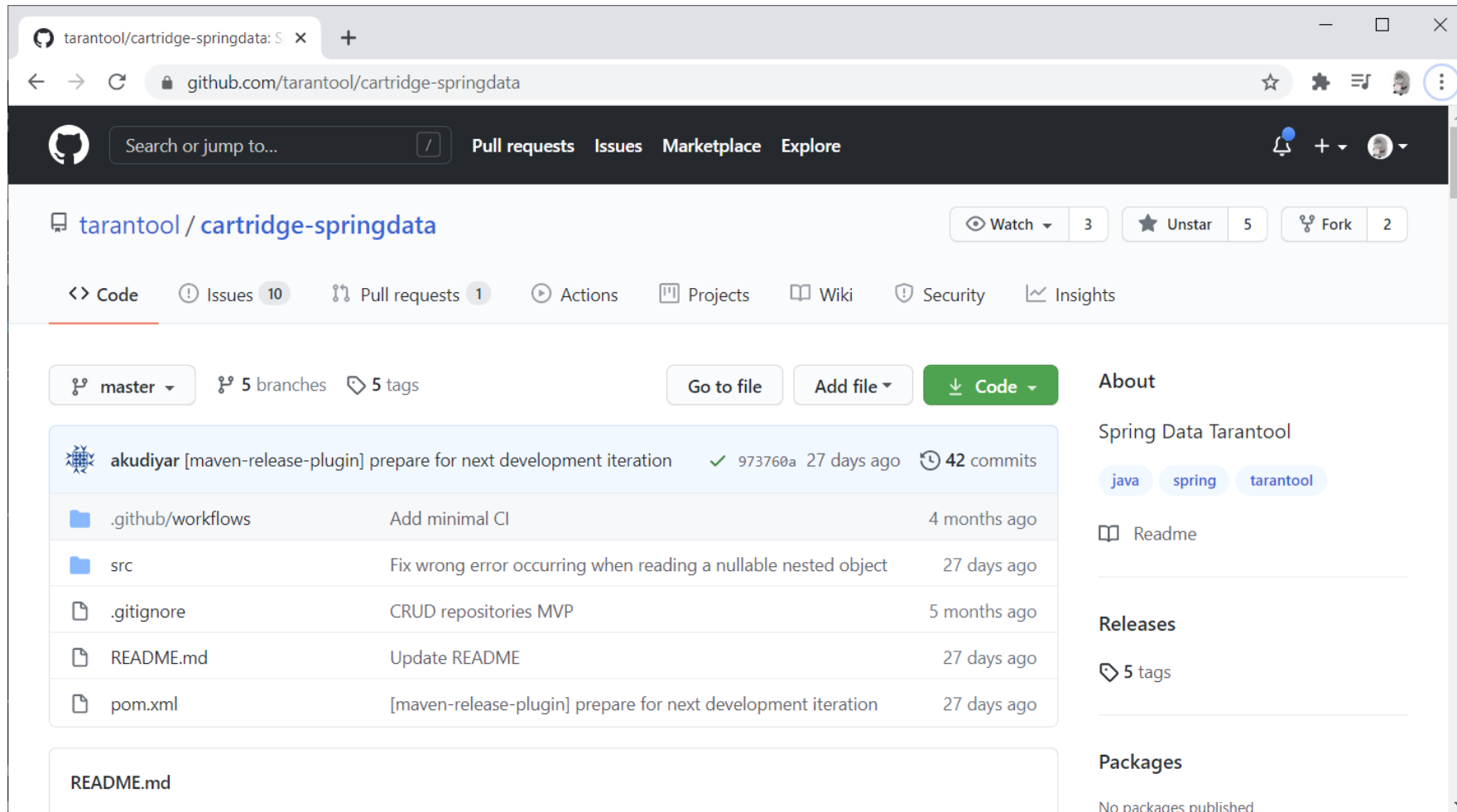
Запуск демонстративного кластера в Gitpod

Для запуска демо-приложения в облаке gitpod.io необходимо выполнить следующие действия:

1. Создайте учётную запись на gitpod.io.
2. Необходимо в личном кабинете gitpod.io зайти в раздел Settings и проставить галочку напротив настройки "Enable Feature Preview". Это позволит запускать docker-контейнеры внутри docker-контейнера с облачной IDE.
3. Запустить workspace с приложением. Создать workspace можно, перейдя по ссылке <https://gitpod.io/#https://github.com/peneksglazami/tarantool-cartridge-webinar>.  
4. В терминале выполните команду `sudo docker-up`. Эта команда запустит демон docker.
5. Откройте новое окно терминала и далее выполните команды запуска кластера, описанные выше в разделе "Запуск демонстративного кластера через docker-compose".

<https://gitpod.io/#https://github.com/peneksglazami/tarantool-cartridge-webinar>

Интеграция со Spring Data



<https://github.com/tarantool/cartridge-springdata>

Алексей Кузин — Работа с шардированными данными в памяти со вкусом Spring Data

Spring Data: a programming model



Templates

Фасад для операций,
которые можно
производить с БД



Repositories

Фасад для CRUD-операций



Object Mapping

Data conversion from raw data
(JSON, driver-specific
structures) into POJO



<https://www.youtube.com/watch?v=tjKK0O2XrFo>

Tarantool Cartridge в Kubernetes



<https://www.youtube.com/watch?v=8NvE6uooMQY>

Заходим на
menti.com

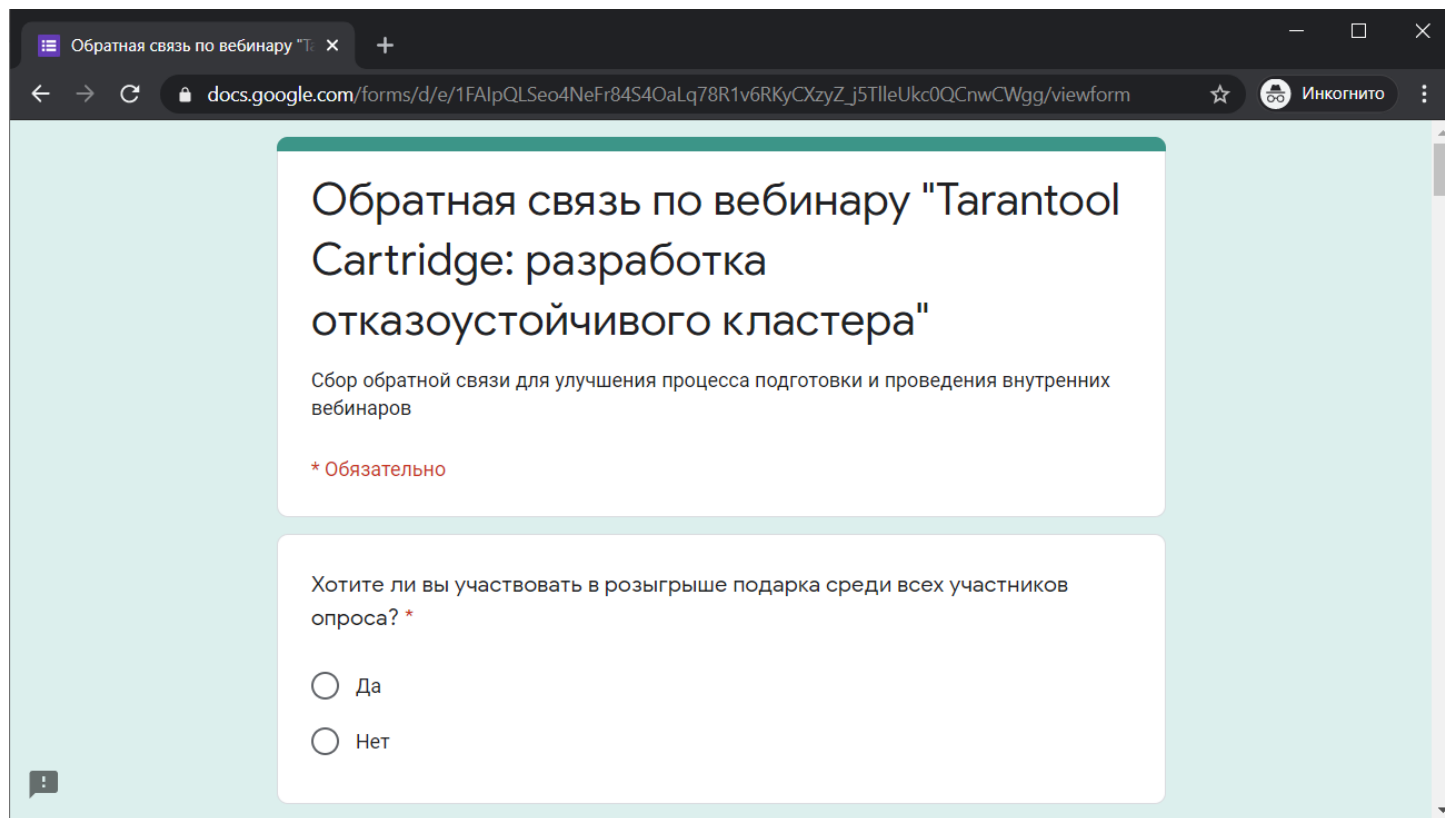
39 20 89 9



Подарок победителю!



Обратная связь



Обратная связь по вебинару "Tarantool Cartridge: разработка отказоустойчивого кластера"

Сбор обратной связи для улучшения процесса подготовки и проведения внутренних вебинаров

* Обязательно

Хотите ли вы участвовать в розыгрыше подарка среди всех участников опроса? *

☐ Да

☐ Нет



<https://forms.gle/j1Nd5C8FKt3AUhVe9>

Розыгрыш приза среди участников опроса



<https://youtu.be/X2ElZsVva4c>

Розыгрыш приза состоится 9 марта в 9:30