Netcracker
An NEC Company

Cassandra, истории из жизни performance-инженера

О себе

- Дмитрий Константинов
- Системный архитектор и, по-прежнему, практикующий Java разработчик в компании Netcracker ☺

О себе

- Дмитрий Константинов
- Системный архитектор и, по-прежнему, практикующий Java разработчик в компании Netcracker ☺
- Активно работаю с различными OpenSource технологиями, такими как Apache Cassandra, Zookeeper, Kafka, Hazelcast
- Профессиональные интересы:
  - распределенные системы
  - производительность
  - отказоустойчивость

План доклада

- О чем вообще говорим, какого рода системы обсуждаем

План доклада

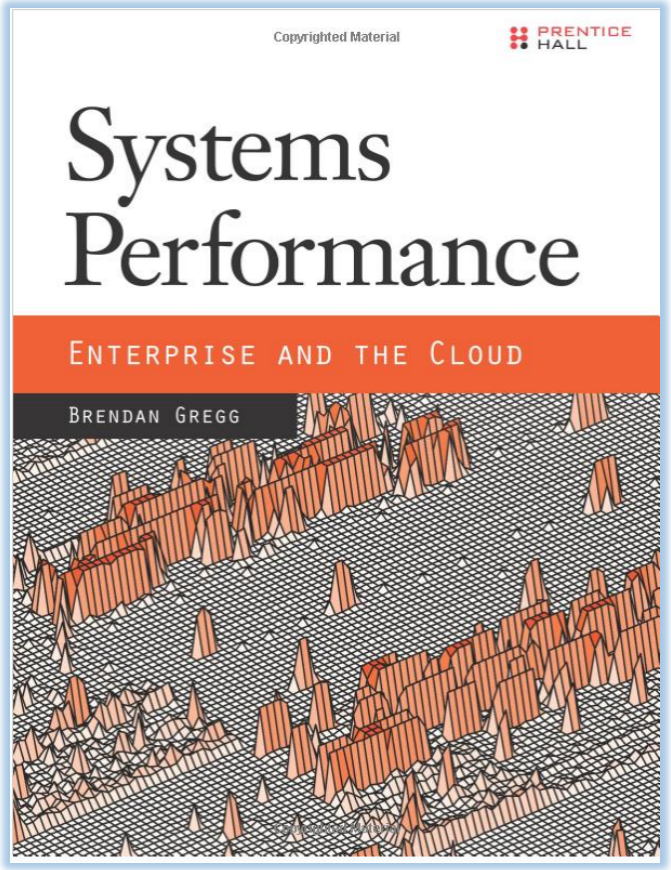
- О чем вообще говорим, какого рода системы обсуждаем
- Приложение / драйвер

План доклада

- О чем вообще говорим, какого рода системы обсуждаем
- Приложение / драйвер
- Сервер
  - Запись
  - Чтение
- Выводы

Этот доклад НЕ об этом

Систематический подход / методология по тестированию производительности и оптимизации



Какого рода системы и нагрузку обсуждаем

- Linux
- Cassandra 3.x
- Java based application, Java driver 3.x
- Несколько датацентров

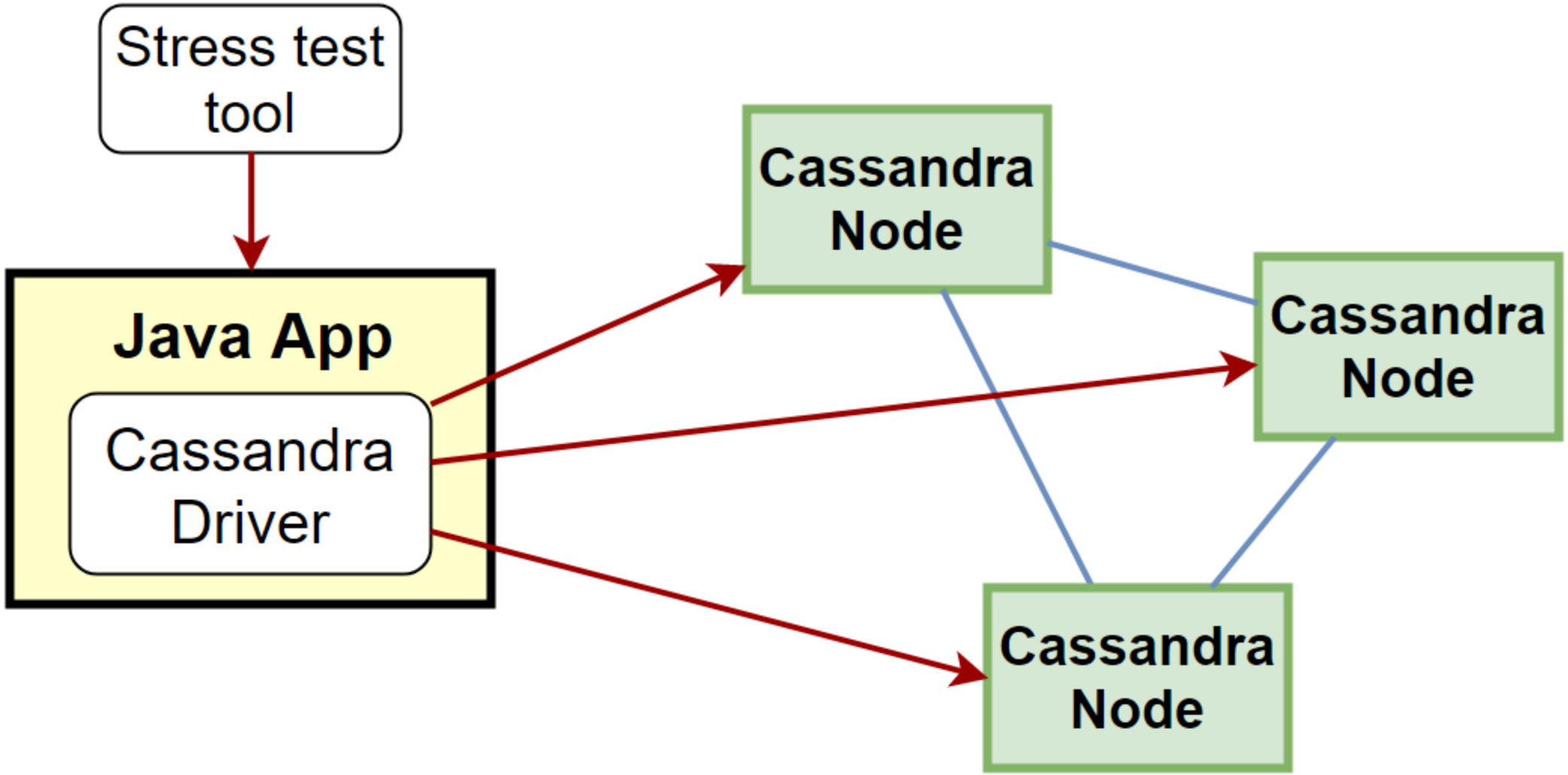
Какого рода системы и нагрузку обсуждаем

- Linux
- Cassandra 3.x
- Java based application, Java driver 3.x
- Несколько датацентров
- Consistency level, обычно - LOCAL_QUORUM
- Тип системы: OLTP или offline batch
- ~100k запросов в секунду в базу, read/write: ~50% / 50%

Чем тестируем под нагрузкой, как собираем метрики

- Нагрузка
  - E2E тесты – генератор нагрузки уровня бизнес-логики
  - Упрощенные тесты отдельных сценариев – свой генератор нагрузки для базы
  - Совсем простые бенчмарки - cassandra-stress
    - https://thelastpickle.com/blog/2017/02/08/Modeling-real-life-workloads-with-cassandra-stress.html

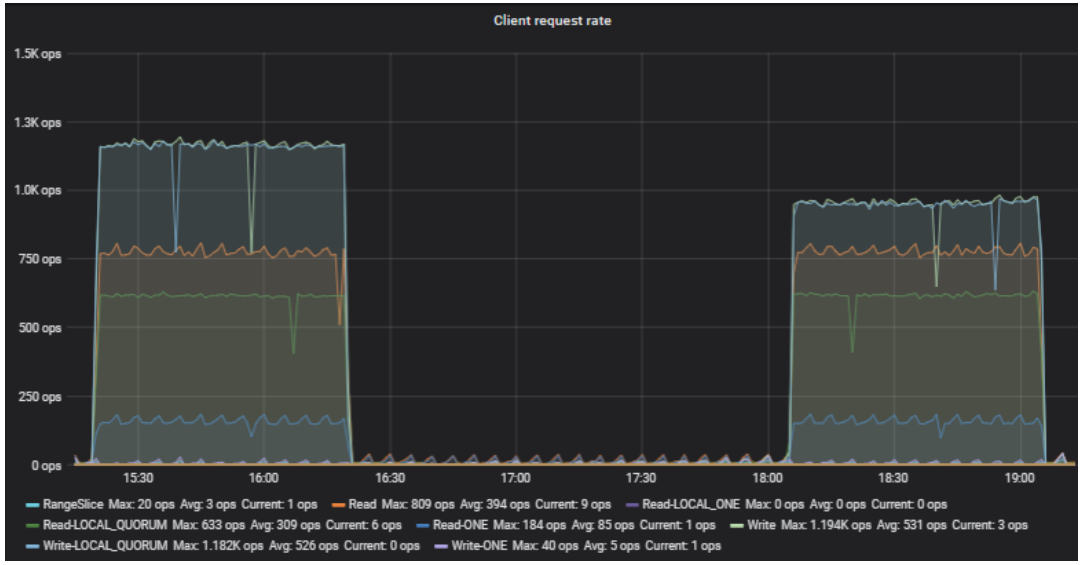
Исследуемая система

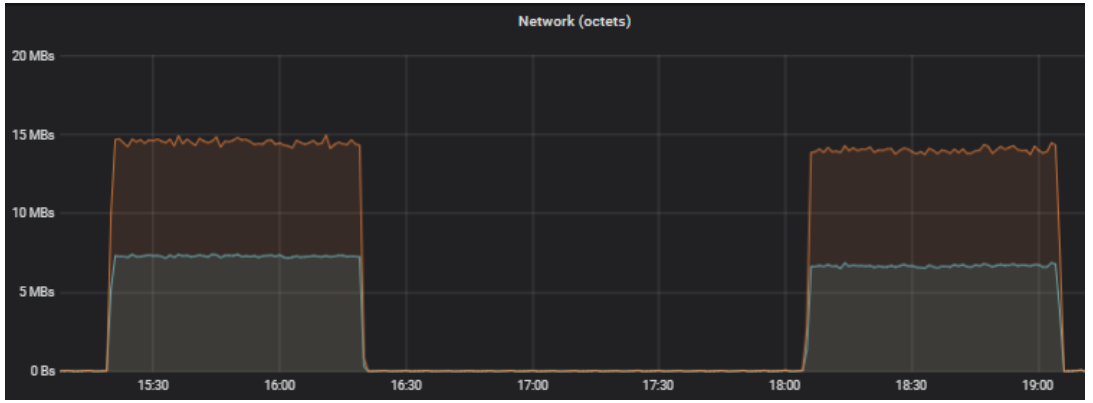


- Запустили тесты
- Все плохо, e2e тормозит - в требования не укладываемся

Собираем метрики

- Метрики
  - CollectD - сбор
    - FastJMX plugin
    - CPU/память/диск/сеть
  - Clickhouse – хранилище метрик
  - Grafana - визуализация





Метрики драйвера

- Встроенных - достаточно мало, очень общие
- Но есть LatencyTracker интерфейс - точка расширения в драйвере
- Можно сделать свои детальные метрики, например на базе HdrHistogram

2018-11-28.csv

```
timestamp, host, table, operationType, count, minMs, medianMs, meanMs, line90Ms, maxMs
2018-11-28T22:36:46.977+0300, 127.0.0.1, "test1"."primitive_test1", INSERT, 1, 31, 31, 31.00, 31, 31
2018-11-28T22:36:47.981+0300, 127.0.0.1, "test1"."primitive_test1", INSERT, 0, 0, 0, 0.00, 0, 0
2018-11-28T22:36:48.979+0300, 127.0.0.1, "test1"."primitive_test1", INSERT, 0, 0, 0, 0.00, 0, 0
2018-11-28T22:36:49.977+0300, 127.0.0.1, "test1"."primitive_test1", SELECT, 206, 3, 71, 70.09, 77, 94
2018-11-28T22:36:49.977+0300, 127.0.0.1, "test1"."primitive_test2", SELECT, 102, 2, 108, 107.11, 114, 141
2018-11-28T22:36:49.977+0300, 127.0.0.1, "test1"."primitive_test2", INSERT, 101, 6, 57, 57.80, 70, 87
2018-11-28T22:36:49.977+0300, 127.0.0.1, "test1"."primitive_test1", INSERT, 100, 3, 47, 60.02, 96, 101
2018-11-28T22:36:50.977+0300, 127.0.0.1, "test1"."primitive_test1", SELECT, 0, 0, 0, 0.00, 0, 0
2018-11-28T22:36:50.977+0300, 127.0.0.1, "test1"."primitive_test2", SELECT, 0, 0, 0, 0.00, 0, 0
```

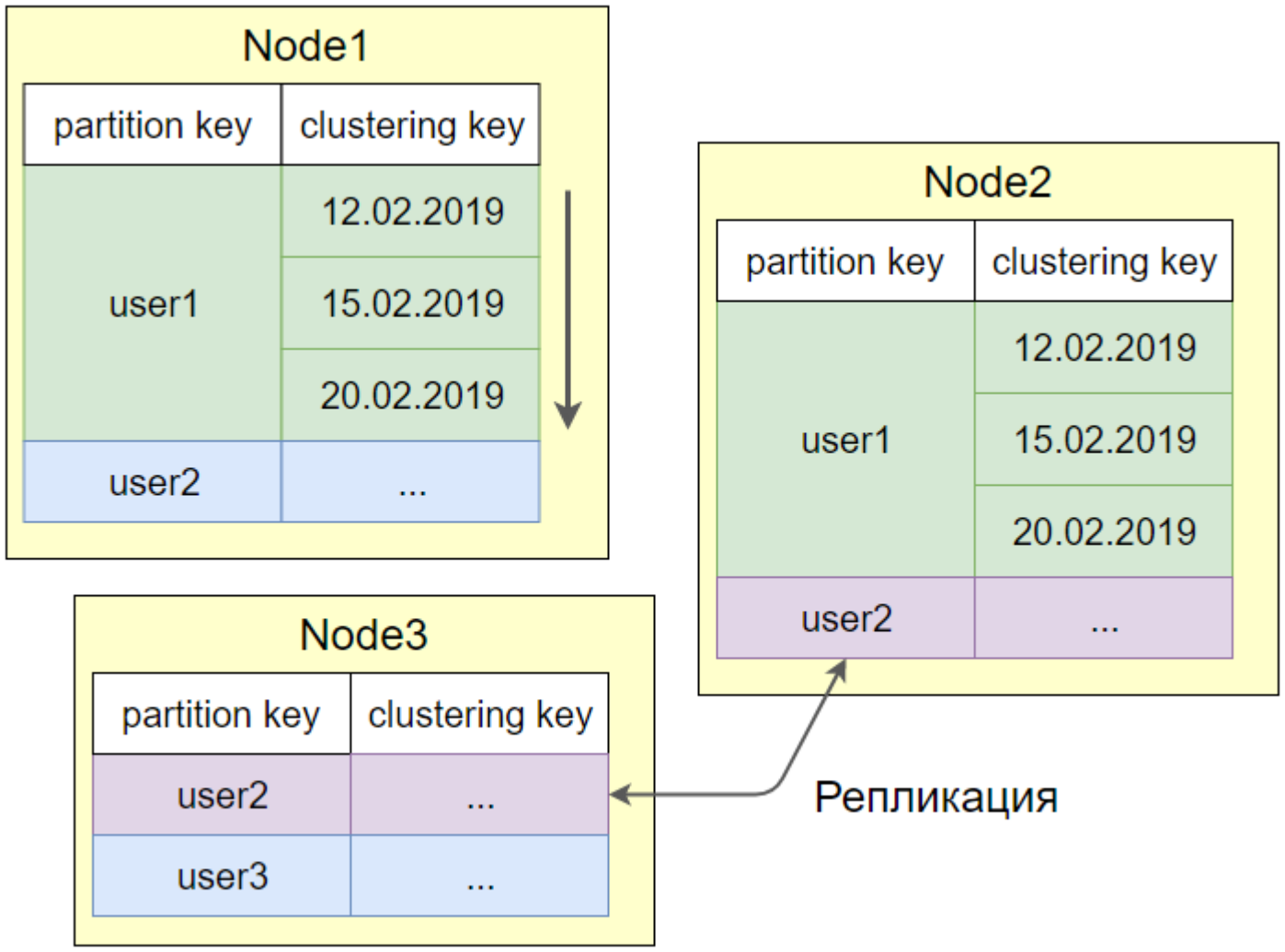
Глава 1

Некоторые концепции Cassandra за 5 минут

Некоторые концепции в Cassandra за 5 минут

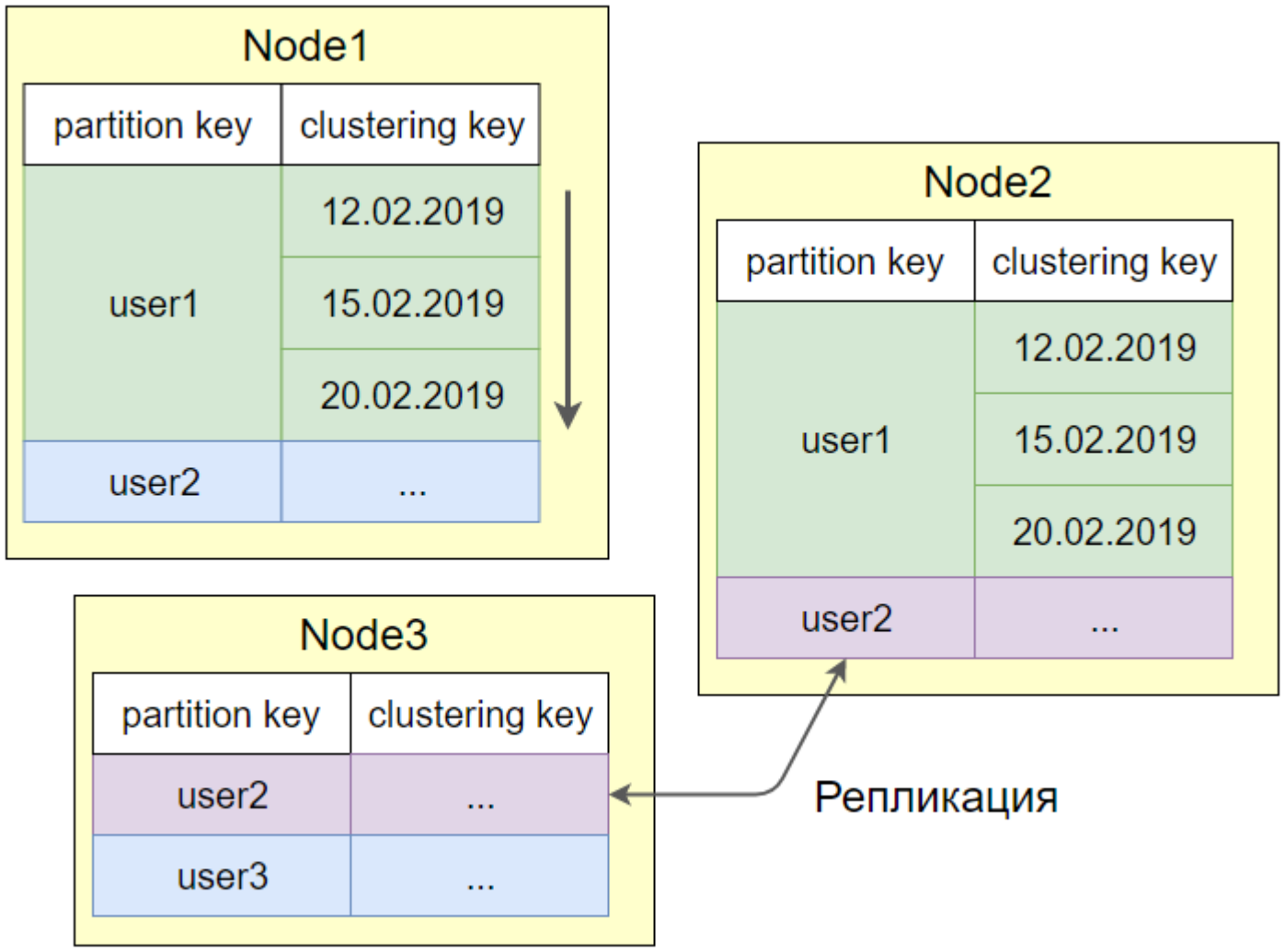
- Keyspace
  - Контейнер для таблиц
  - Задает стратегию репликации - количество реплик в каждом датацентре

Структура первичного ключа



- Таблицы имеют составной первичный ключ, состоящий из 2 частей:
  - Partition key – отвечает за распределение данных по кластеру. Каждая партиция имеет несколько реплик.

Структура первичного ключа



- Таблицы имеют составной первичный ключ, состоящий из 2 частей:
  - Partition key – отвечает за распределение данных по кластеру. Каждая партиция имеет несколько реплик.
  - Clustering key – строки внутри каждой партиции упорядочены по данной части ключа. Дает возможность делать запросы по диапазону значений.

Tombstone

- Tombstone – отметка в базе о том, что данные (ячейка, строка, диапазон строк, …) удалены

Tombstone

- Tombstone – отметка в базе о том, что данные (ячейка, строка, диапазон строк, …) удалены
- А зачем он нужен, почему не удалять сразу:
  1. Модель хранения данных – во время записи существующие данные не читаются

Tombstone

- Tombstone – отметка в базе о том, что данные (ячейка, строка, диапазон строк, …) удалены
- А зачем он нужен, почему не удалять сразу:
  1. Модель хранения данных – во время записи существующие данные не читаются
  2. Чтобы избежать “зомби” (возвращения удаленных данных). Cassandra – распределённая система без выделенного лидера, использующая last-write-win стратегию разрешения конфликтов

Tombstone

- Tombstone – отметка в базе о том, что данные (ячейка, строка, диапазон строк, …) удалены
- А зачем он нужен, почему не удалять сразу:
  1. Модель хранения данных – во время записи существующие данные не читаются
  2. Чтобы избежать “зомби” (возвращения удаленных данных). Cassandra – распределённая система без выделенного лидера, использующая last-write-win стратегию разрешения конфликтов
- Время жизни tombstone ограничено параметром на уровне таблицы - gc_grace_seconds

Глава 1

Приложение / драйвер

Драйвер

- Справка
  - Базовые рекомендации 

Базовые рекомендации

Используем PreparedStatement

- Уменьшаем накладные расходы на передачу запроса (передается только его id и параметры)
- Убираем расходы на разбор запроса на стороне сервера-координатора
- PreparedStatement в Cassandra driver потокобезопасен, создаем один раз и используем из всех потоков

Базовые рекомендации

BatchStatement

- Есть возможность объединять несколько INSERT/UPDATE/DELETE запросов в 1 сетевой запрос
- BatchStatement бывают разные:
  - Logged – атомарность применения (используем, только если действительно нужно – есть существенные накладные расходы)

Базовые рекомендации

BatchStatement

- Есть возможность объединять несколько INSERT/UPDATE/DELETE запросов в 1 сетевой запрос
- BatchStatement бывают разные:
  - Logged – атомарность применения (используем, только если действительно нужно – есть существенные накладные расходы)
  - Unlogged - просто пачка запросов, координатор их разделит и выполнит отдельно, лучше это сделать на клиенте

Базовые рекомендации

BatchStatement

- Есть возможность объединять несколько INSERT/UPDATE/DELETE запросов в 1 сетевой запрос
- BatchStatement бывают разные:
  - Logged – атомарность применения (используем, только если действительно нужно – есть существенные накладные расходы)
  - Unlogged - просто пачка запросов, координатор их разделит и выполнит отдельно, лучше это сделать на клиенте
  - Single-partition

Базовые рекомендации

Single-partition BatchStatement

- Partition ключ у всех вложенных запросов совпадает
- Keyspace у всех вложенных запросов совпадает

Базовые рекомендации

Single-partition BatchStatement

- Partition ключ у всех вложенных запросов совпадает
- Keyspace у всех вложенных запросов совпадает

- Плюсы:
  - Нет накладных расходов
  - Если имя таблицы совпадает - есть атомарность и изоляция

Базовые рекомендации

Single-partition BatchStatement

- Partition ключ у всех вложенных запросов совпадает
- Keyspace у всех вложенных запросов совпадает

- Плюсы:
  - Нет накладных расходов
  - Если имя таблицы совпадает - есть атомарность и изоляция
  - Если таблицы разные - есть атомарность с точки зрения применения, но нет изоляции (можно увидеть частичное изменение)

Больше деталей: https://www.datastax.com/blog/2012/02/coming-cassandra-11-row-level-isolation

Базовые рекомендации

Используем netty-transport-native-epoll для драйвера

- Ниже CPU usage
- Меньше выделений памяти

Драйвер

- Справка
  - Базовые рекомендации
- Анализ издержек на стороне драйвера 

Приложение/драйвер

- Наблюдение: приложение потребляет много CPU и выделяет довольно много объектов в памяти
- Базовые рекомендации уже применили
- Посмотрим детали:
  - Async profiler – CPU профиль
  - Java Flight Recorder - выделение памяти, конфликты при захвате блокировок
  - GC логи – как часто собираем мусор, какие паузы

Драйвер

- Справка
  - Базовые рекомендации
- Анализ издержек на стороне драйвера
  - Формирование запроса 

Приложение / драйвер – создание запроса

```
1  BoundStatement boundStmt = new BoundStatement(preStmt);
2  boundStmt.set("column1", value1);
3  boundStmt.set("column2", value2);
4  ResultSet result = session.execute(boundStmt);
```

BoundStatement.set:

1. Ищем Codec для типа значения
2. Сериализуем значение в ByteBuffer, используя найденный Codec
3. Ищем по имени – позиции в ByteBuffer[] values
4. Результат кладем в ByteBuffer[] values

Приложение / драйвер – создание запроса

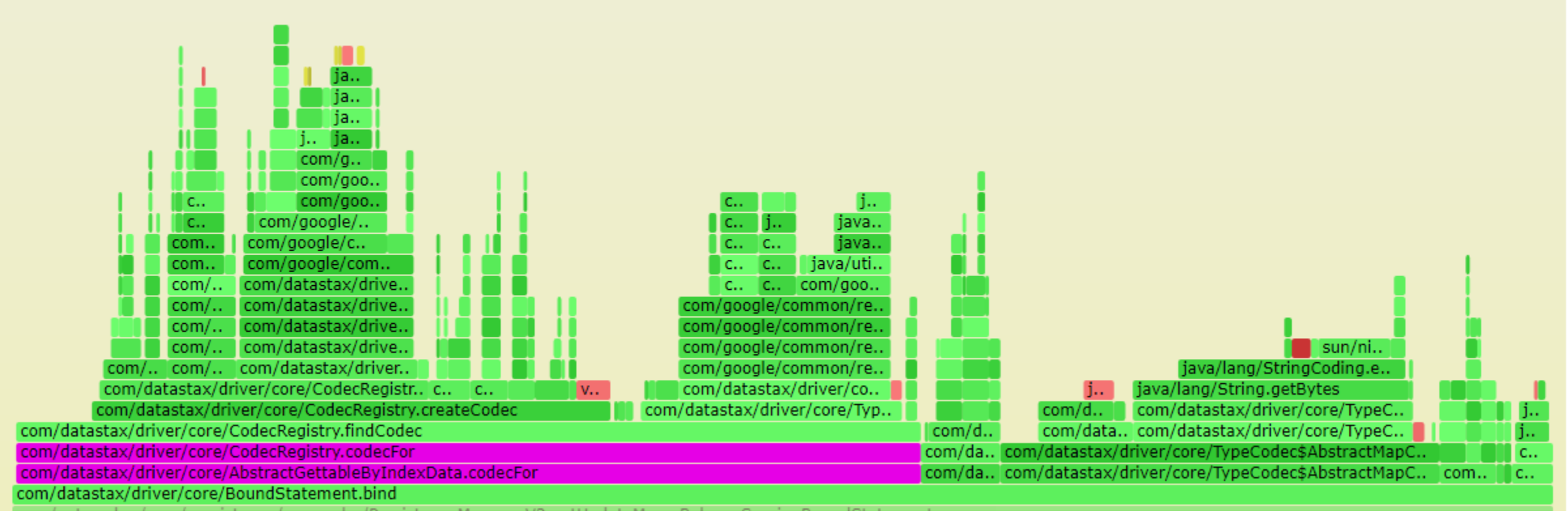
```
1  BoundStatement boundStmt = new BoundStatement(preStmt);
2  boundStmt.set("column1", value1);
3  boundStmt.set("column2", value2);
4  ResultSet result = session.execute(boundStmt);
```

BoundStatement.set:

1. **Ищем Codec для типа значения**
2. Сериализуем значение в ByteBuffer, используя найденный Codec
3. Ищем по имени – позиции в ByteBuffer[] values
4. Результат кладем в ByteBuffer[] values

Codecs и накладные расходы, связанные с ними

- “Ну, зачем ты, зачем ты туда полез?”



Codecs и накладные расходы, связанные с ними

- Все значения драйвер кодирует/декодирует через настраиваемые кодеки (Codec interface)
- Кеш кодеков достаточно ресурсоемкий
- Ускоряли:
  - 3.2.0 - JAVA-1308: CodecRegistry performance improvements.
  - 3.7.0 - JAVA-2002: Reimplement TypeCodec.accepts to improve performance.

но все еще есть накладные расходы, особенно для UDT (User Defined Types) и коллекций

Codecs и накладные расходы, связанные с ними

- WA: можно указать кодеки явно, в горячих местах:

```
row.get(1, TypeCodec.varchar())
stm.set(1, date, TypeCodec.timestamp());
```

Codecs и накладные расходы, связанные с ними

- WA: можно указать кодеки явно, в горячих местах:

```
row.get(1, TypeCodec.varchar())
stm.set(1, date, TypeCodec.timestamp());
```

  - С UDT чуть сложнее:

```
KeyspaceMetadata keyspace = cluster.getMetadata().getKeyspace("test");
UserType type = keyspace.getUserType("some");
TypeCodec<List<UDTValue>> codec = list(userType(type));
```

Codecs и накладные расходы, связанные с ними

Простой запрос:

```
CREATE TABLE test_driver.test_table_value8 (
    part_key text,
    clust_key text,
    value1 text, value2 text, value3 text, value4 text,
    value5 text, value6 text, value7 text, value8 text,
    PRIMARY KEY (part_key, clust_key)
);

SELECT clust_key, value1, value2, value3, value4,
                  value5, value6, value7, value8
  FROM test_table_value8 WHERE part_key=?
```

Codecs и накладные расходы, связанные с ними

SELECT, 8 строк * 9 колонок в результате

```
row.getString(index)
row.get(index, TypeCodec.varchar())
```

Codecs и накладные расходы, связанные с ними

SELECT, 8 строк * 9 колонок в результате

```
row.getString(index)
row.get(index, TypeCodec.varchar())
```

- Общее время выполнения запроса меняется на уровне погрешности
- CPU usage на стороне клиента: около -0.5% в случае с явных указанием codec
- Практической пользы для свежей версии драйвера нет, нет смысла тут копать?!

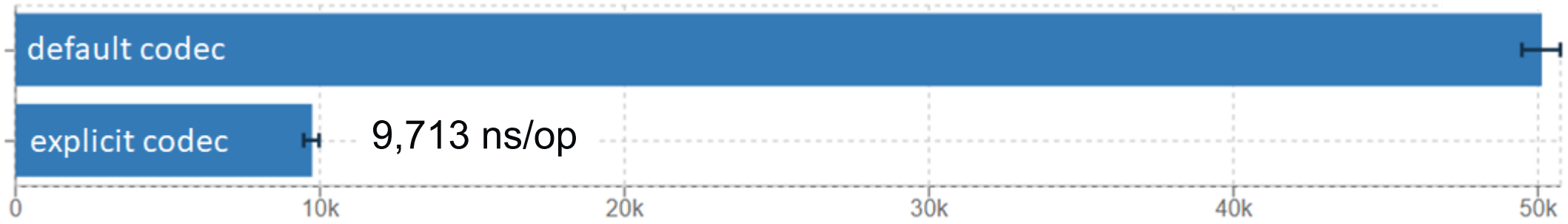
Codecs и накладные расходы, связанные с ними

Проверяем UDT:

```
CREATE TYPE test_driver.udt_type (
    field1 text,
    field2 frozen<list<text>>,
);
CREATE TABLE test_driver.test_table_udt_value (
    part_key text,
    clust_key text,
    udt_value frozen<list<udt_type>>,
    PRIMARY KEY (part_key, clust_key)
);
SELECT clust_key, udt_value FROM test_table_udt_value WHERE part_key=?
```

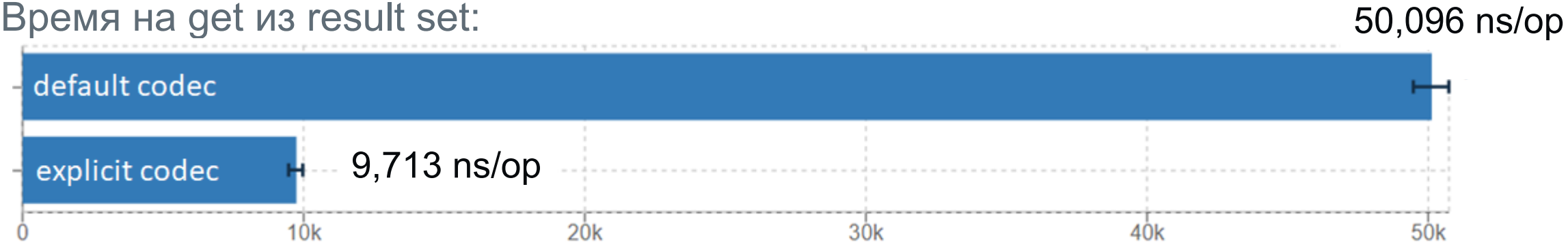
Codecs и накладные расходы, связанные с ними

- SELECT, в результате 4 строки, в каждой - 4 элемента в udt_value списке
- Время на get из result set:

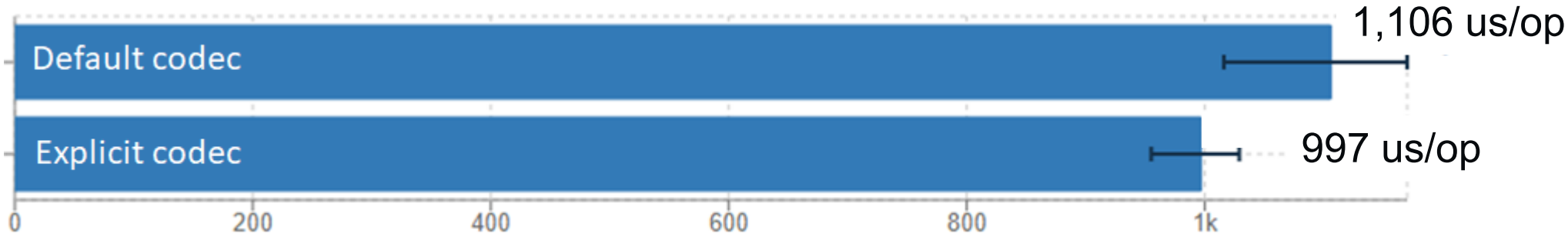


Codecs и накладные расходы, связанные с ними

- SELECT, в результате 4 строки, в каждой - 4 элемента в udt_value списке
- Время на get из result set:



- Общее время выполнения запроса:



- CPU usage на стороне клиента: **-14%** для случая с явных указанием codec

Приложение / драйвер

```
1  BoundStatement boundStmt = new BoundStatement(preStmt);
2  boundStmt.set("column1", value1);
3  boundStmt.set("column2", value2);
4  ResultSet result = session.execute(boundStmt);
```

BoundStatement.set:

1. Ищем Codec для типа значения
2. **Кодируем значение в ByteBuffer, используя найденный Codec**
3. Ищем по имени – позиции в ByteBuffer[] values
4. Результат кладем в ByteBuffer[] values

Затраты на кодирование параметров

- Если есть неравномерность для входных данных – можем кешировать ByteBuffer варианты для часто встречающихся параметров
- (String -> ByteBuffer) cache
- (Integer -> ByteBuffer) cache
- (Date -> ByteBuffer) cache

Приложение / драйвер

```
1  BoundStatement boundStmt = new BoundStatement(preStmt);
2  boundStmt.set("column1", value1);
3  boundStmt.set("column2", value2);
4  ResultSet result = session.execute(boundStmt);
```

BoundStatement.set:

1. Ищем Codec для типа значения
2. Кодируем значение в ByteBuffer, используя найденный Codec
3. **Ищем по имени – позиции в ByteBuffer[] values**
4. Результат кладем в ByteBuffer[] values

Классика: имена или индексы?

- BoundStatement

```
stmt.setInt(columnName, 28)
stmt.setInt(index, 28)
```

- Row

```
row.getInt(columnName)
row.getInt(index)
```

- UDT

```
udt.setInt(columnName, 496)
udt.setInt(index, 496)
```

Классика: имена или индексы?

- BoundStatement

```
stmt.setInt(columnName, 28)
stmt.setInt(index, 28)
```

- Row

```
row.getInt(columnName)
row.getInt(index)
```

- UDT

```
udt.setInt(columnName, 496)
udt.setInt(index, 496)
```

- Индексы немного быстрее
  - Это не просто Map.get – тут еще накладные расходы от CQL спецификации: case-insensitive, кавычки, etc.

Классика: имена или индексы?

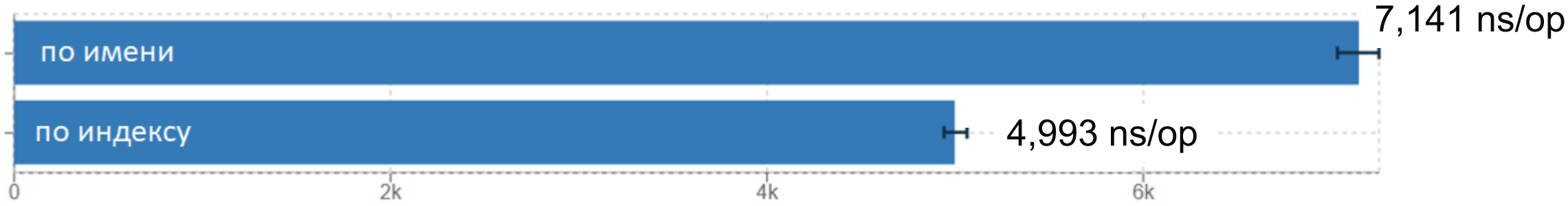
- SELECT, 8 строк * 9 колонок в результате

```
row.getString(index)
row.getString(name)
```

Классика: имена или индексы?

- SELECT, 8 строк * 9 колонок в результате

```
row.getString(index)
row.getString(name)
```

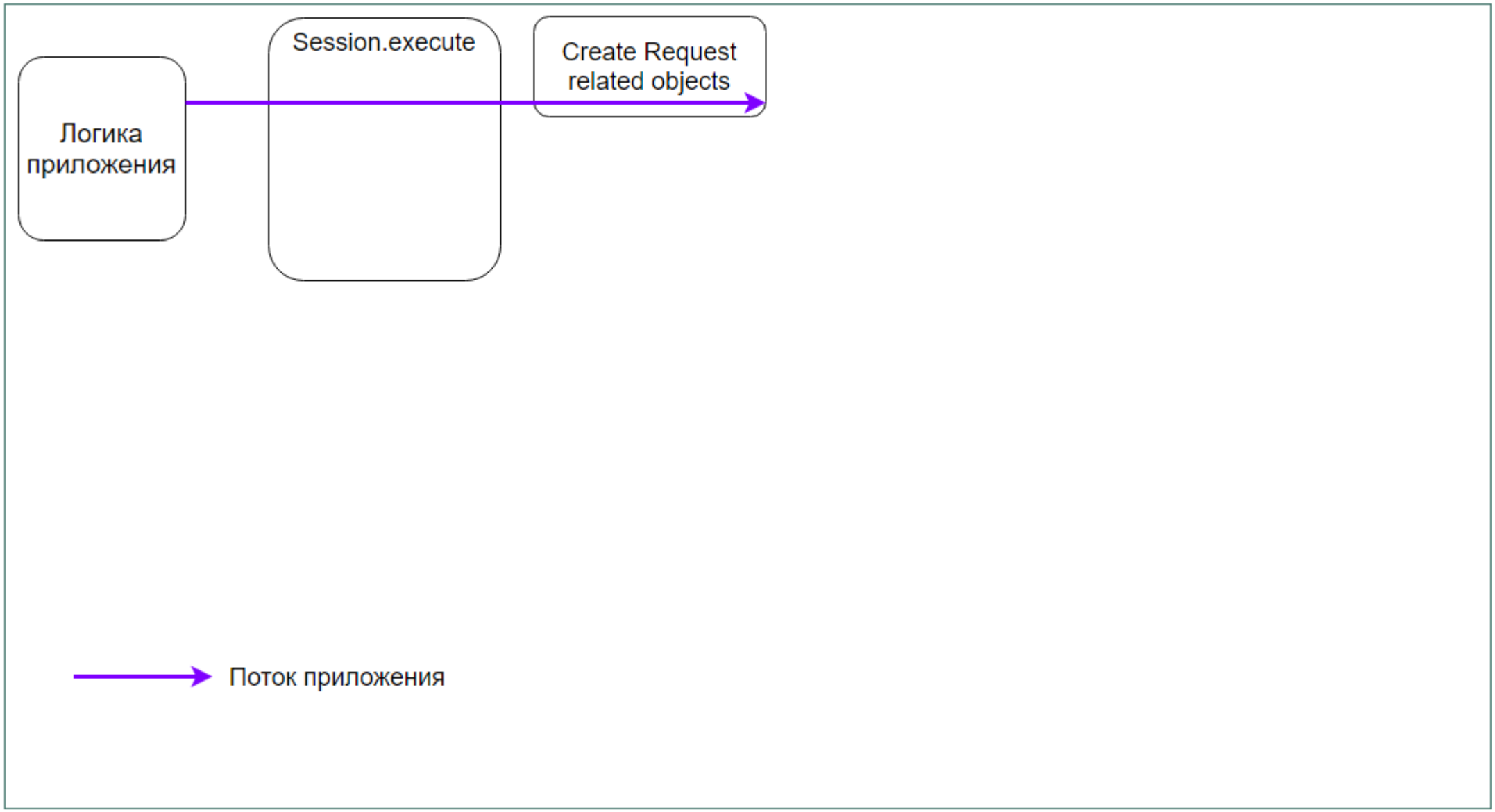


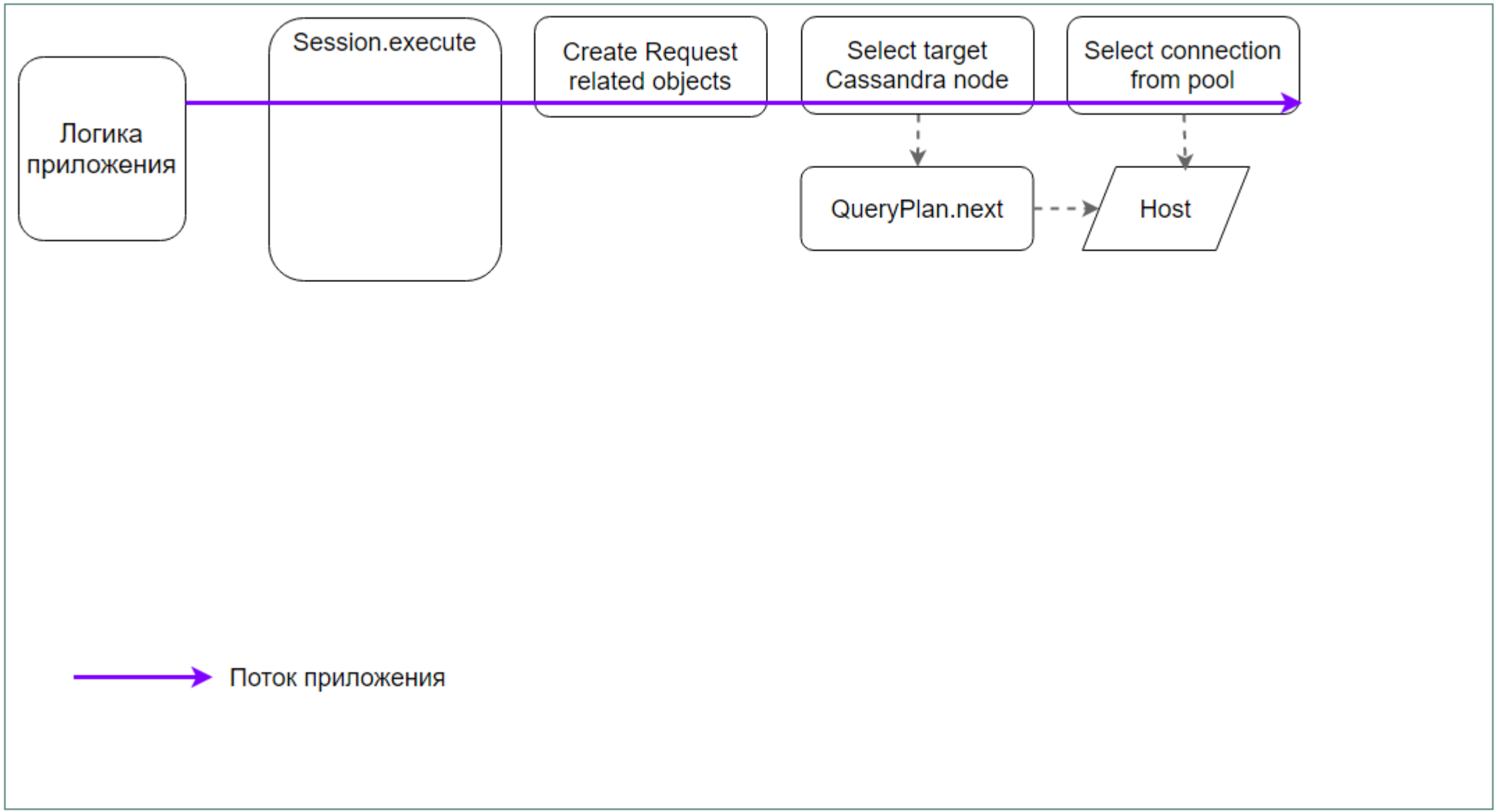
  - Общее время выполнения запроса меняется на уровне погрешности
  - CPU usage на стороне клиента: **-1%** в случае с указанием индекса

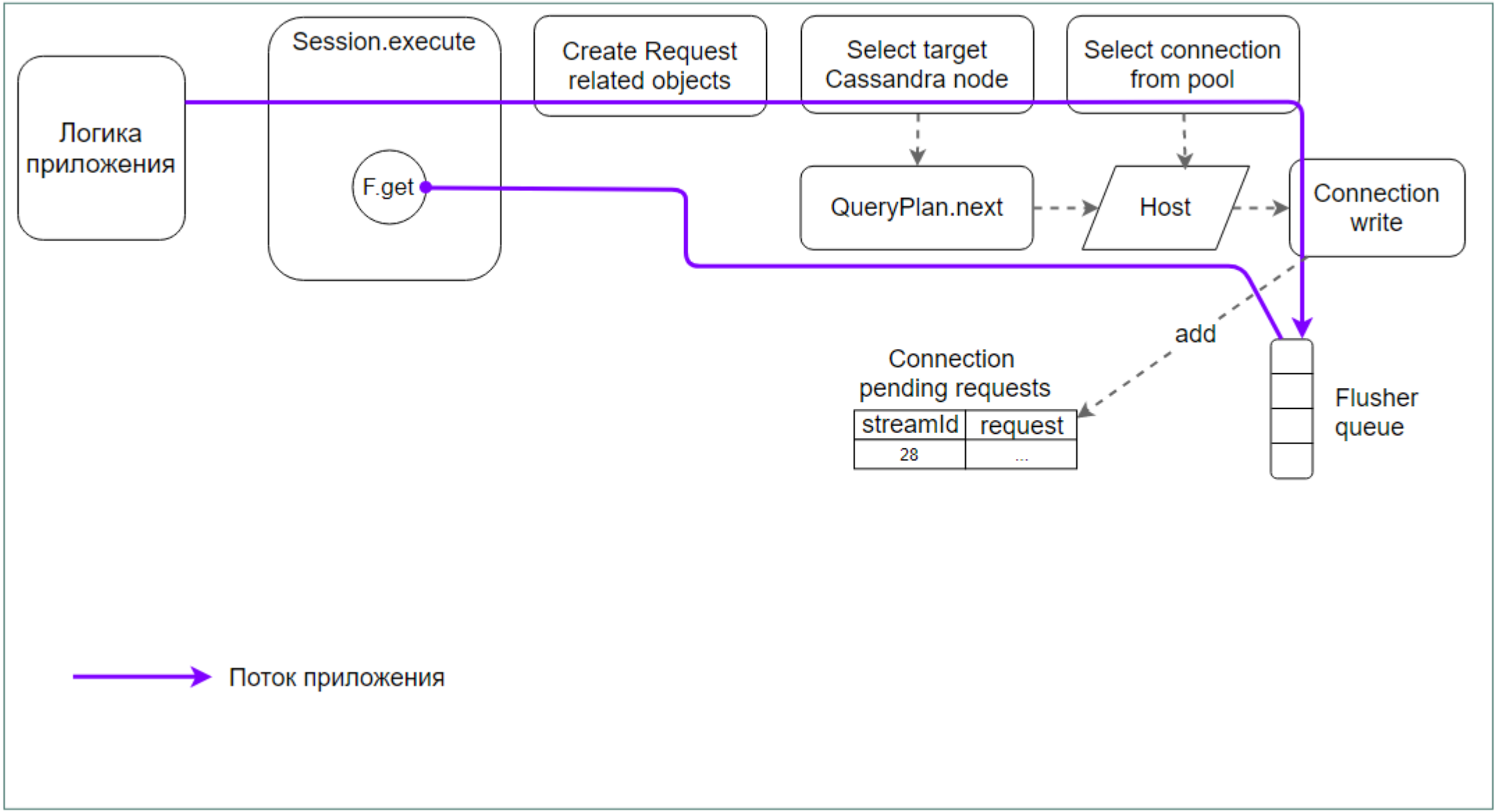
Драйвер

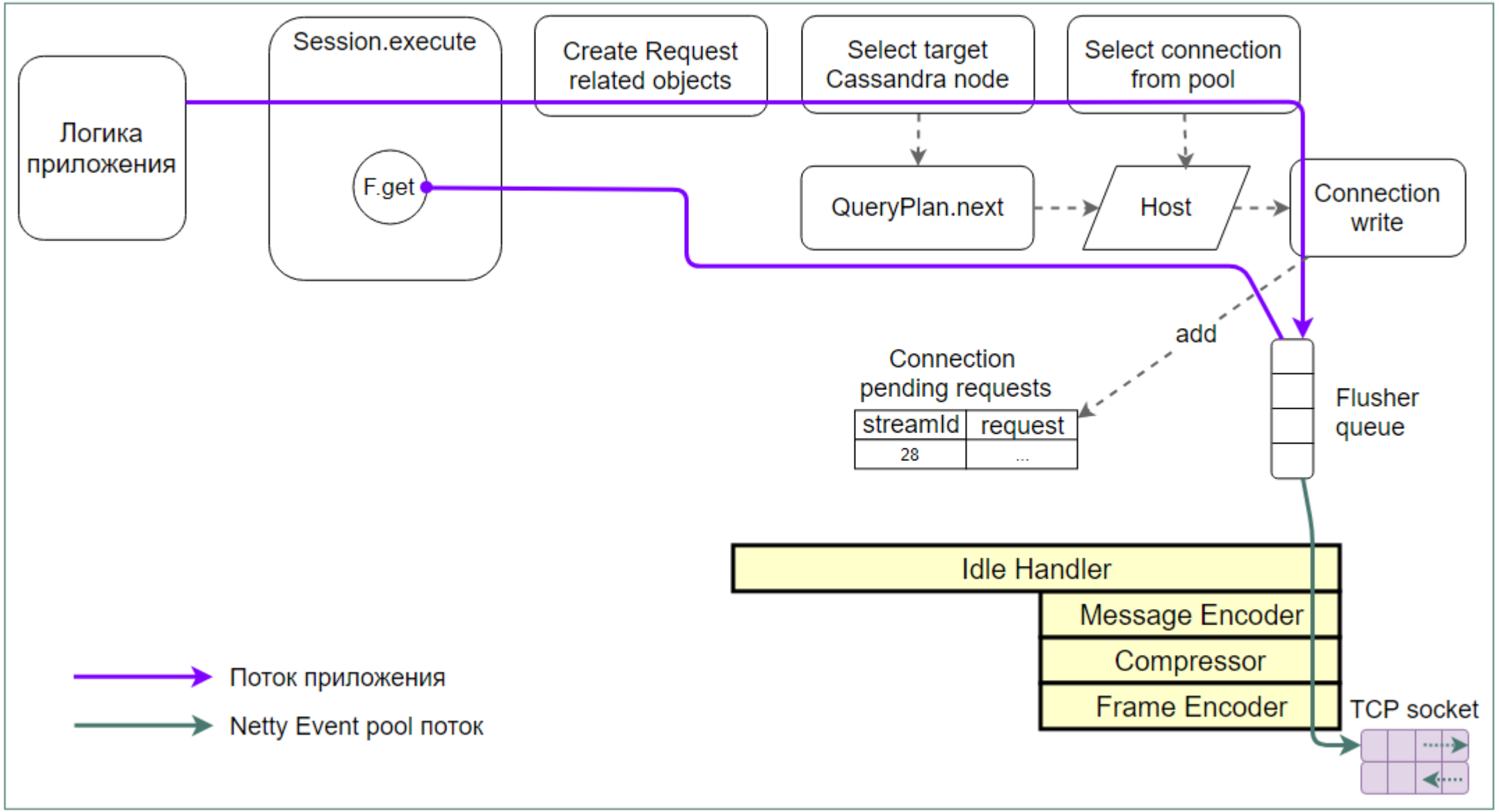
- Справка
  - Базовые рекомендации
- Анализ издержек на стороне драйвера
  - Формирование запроса
  - Выполнение запроса 

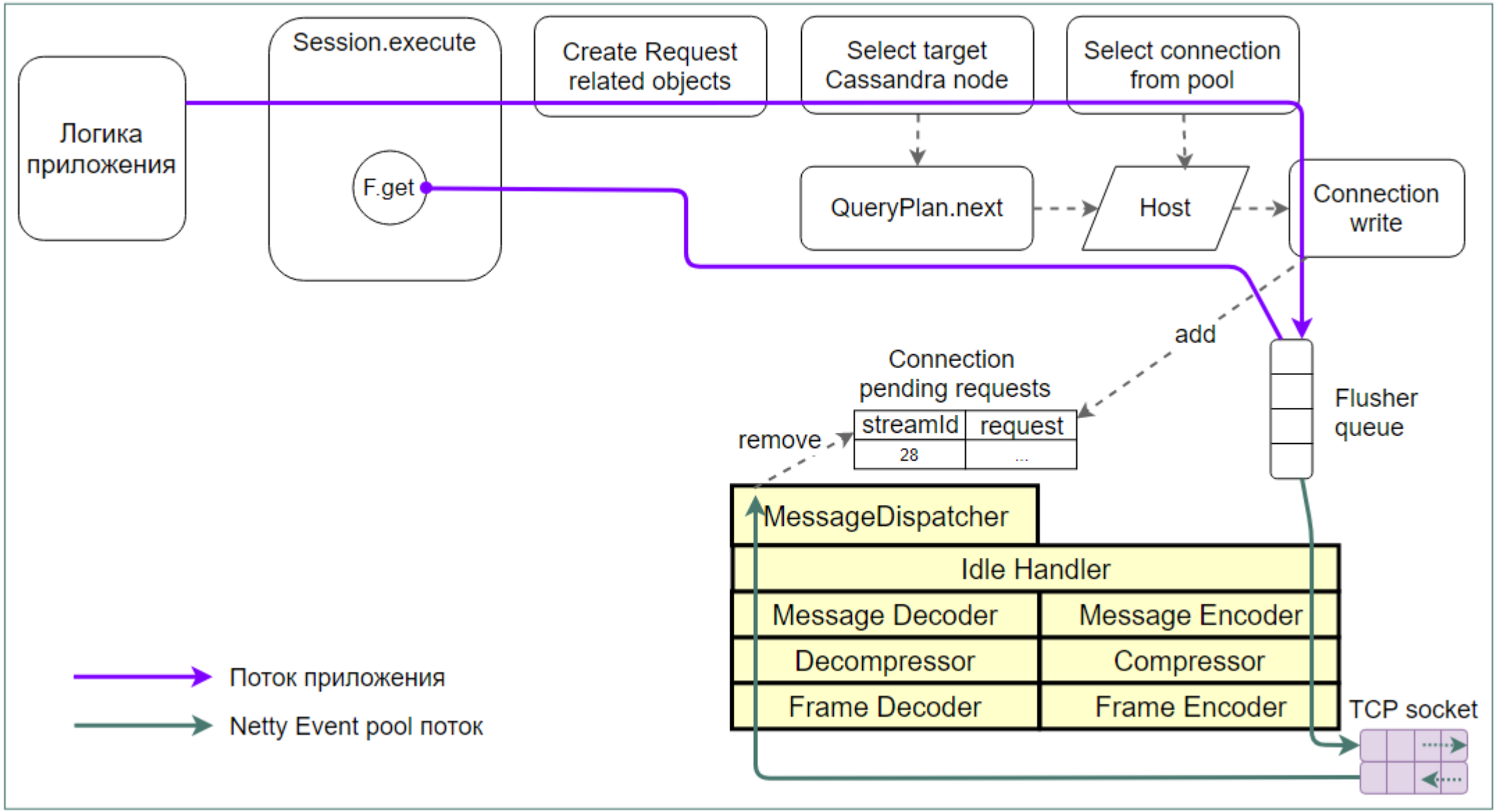
```
ResultSet result = session.execute(boundStmt);
```

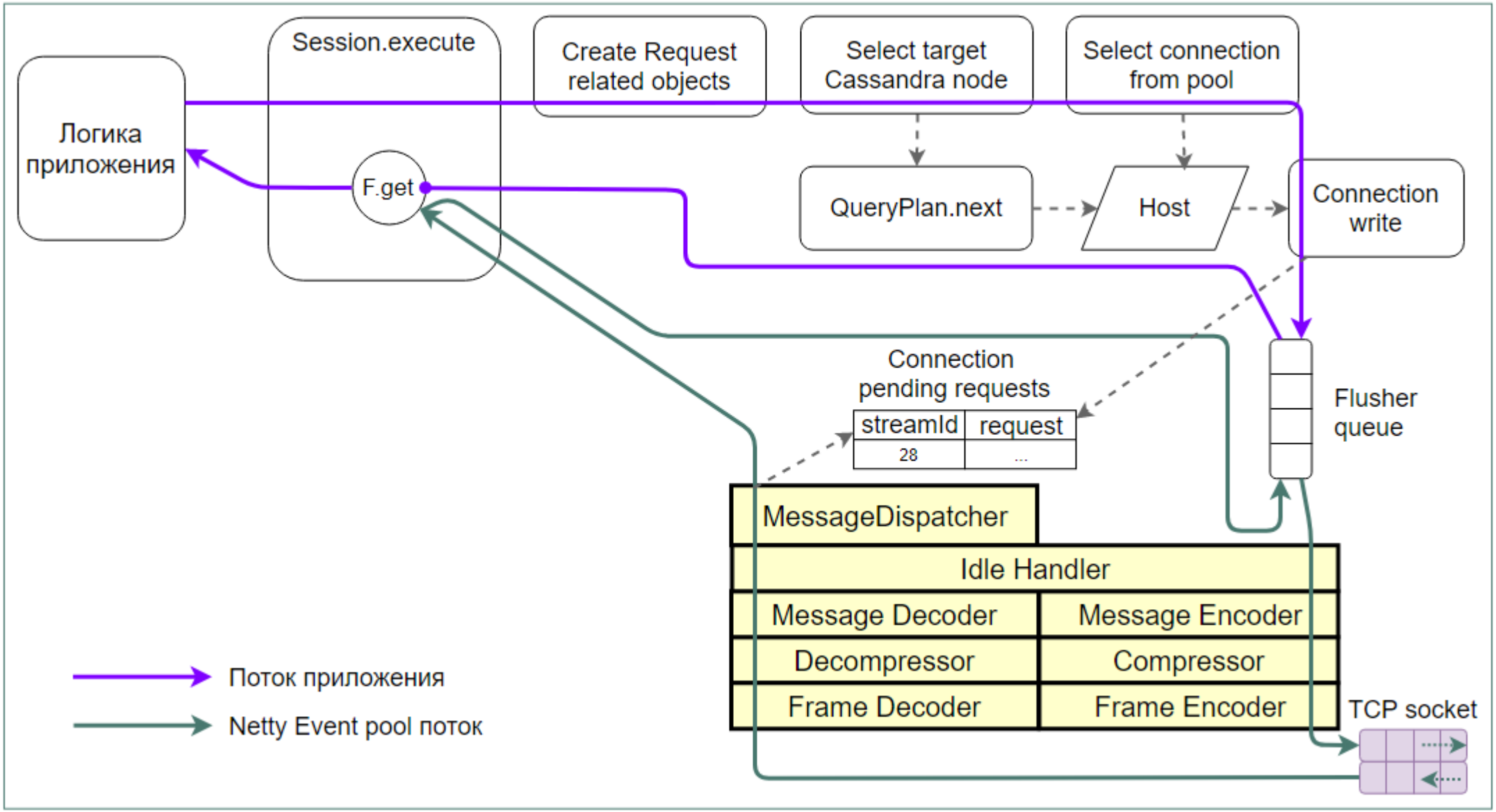










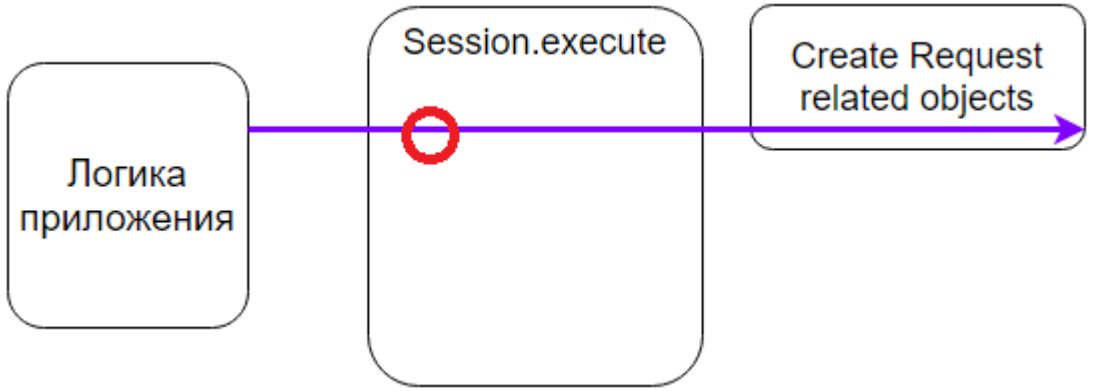


Драйвер

- Справка
  - Базовые рекомендации
- Анализ издержек на стороне драйвера
  - Формирование запроса
  - Выполнение запроса
    - Сетевое взаимодействие 

-Dcom.datastax.driver.CHECK_IO_DEADLOCKS=false

- По умолчанию драйвер проверяет, не пытаемся ли мы выполнить блокирующий вызов в Netty Event Loop потоке
- Если мы заблокируем этот поток – Netty не сможет обрабатывать сетевые события и Flusher очередь некоторое время, будут задержки и таймауты
- Такое возможно, когда мы пытаемся выполнить синхронный запрос в callback вызове для асинхронного запроса



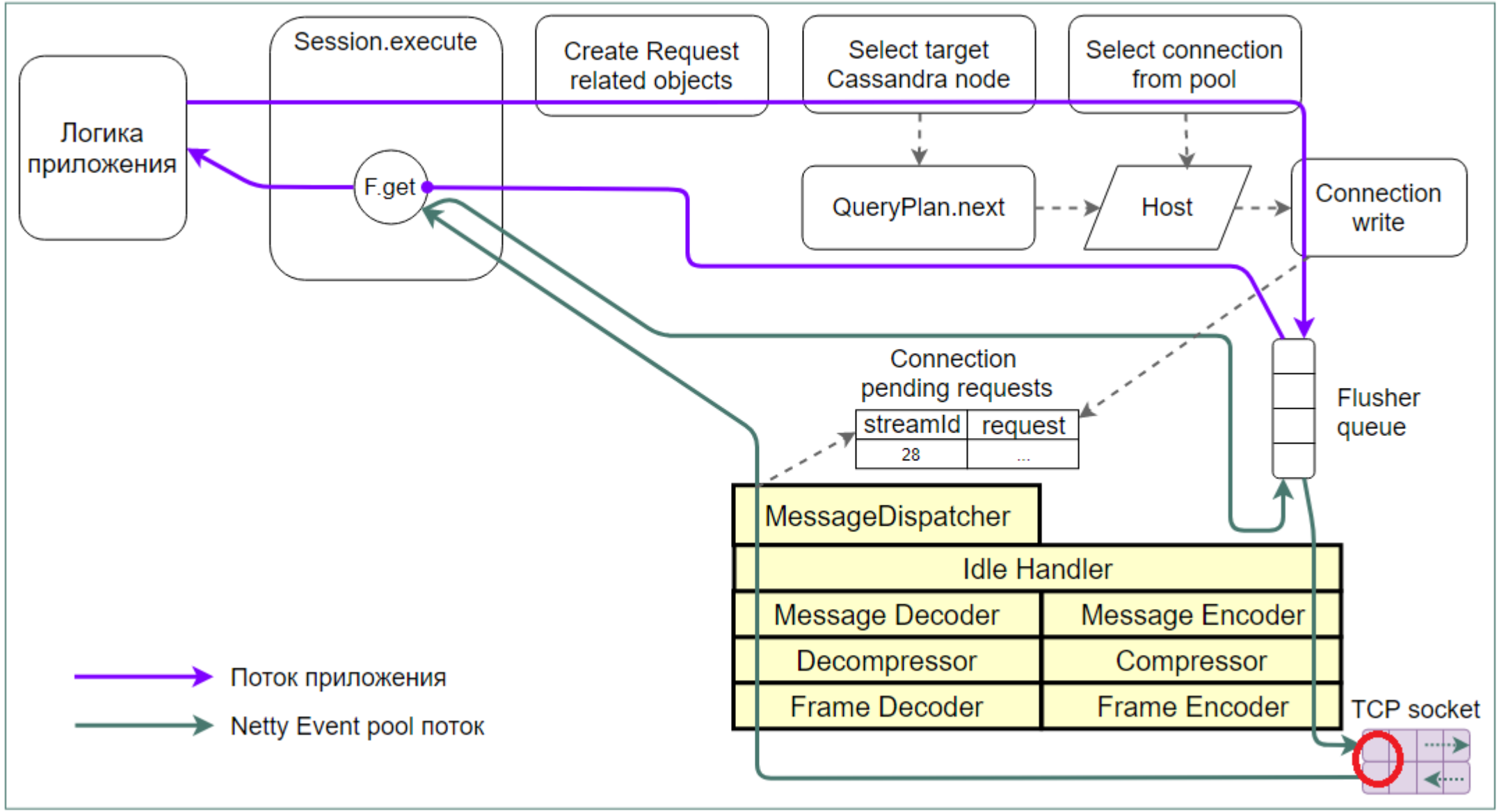
Сетевое взаимодействие

- 1659 (reduce epoll wait overheads) – версия драйвера 3.3.2
  - По мотивам: CASSANDRA-13651
  - -Dcom.datastax.driver.FLUSHER_SCHEDULE_PERIOD_NS=0
  - Избегаем лишних системных вызовов epoll_wait() / timerfd_create()
- com.datastax.driver.core.PoolingOptions#setConnectionsPerHost
  - Несколько TCP соединений к каждой из Cassandra нод

Сетевое взаимодействие

- 1659 (reduce epoll wait overheads) – версия драйвера 3.3.2
  - По мотивам: CASSANDRA-13651
  - -Dcom.datastax.driver.FLUSHER_SCHEDULE_PERIOD_NS=0
  - Избегаем лишних системных вызовов epoll_wait() / timerfd_create()
- com.datastax.driver.core.PoolingOptions#setConnectionsPerHost
  - Несколько TCP соединений к каждой из Cassandra нод
- Есть свежие оптимизации со стороны Netty:
  - https://github.com/netty/netty/pull/9192 - Netty 4.1.37
    - “a read on timerfd and eventfd when the EventLoop wakes up"
  - https://github.com/netty/netty/pull/7834 - Netty 4.1.41
    - “timerfd_settime on each call to epoll_wait is expensive”

Сетевое взаимодействие



CHECK_IO_DEADLOCKS, FLUSHER_SCHEDULE_PERIOD_NS - замеры

- Простые SELECT запросы, возвращается 1 строка
- Общее время выполнения запроса меняется на уровне погрешности
- 50 потоков
- CPU:

Тест	CPU, usr	CPU, sys	CPU, total
Baseline (driver 3.7.2)	23.6	15.5	**39.1**
CHECK_IO_DEADLOCKS=false	22.5 (-1.1)	15.5	**38 (-1.1%)**
FLUSHER_SCHEDULE_PERIOD_NS=0	21 (-2.6)	13 (-2.5)	**34 (-5.1%)**

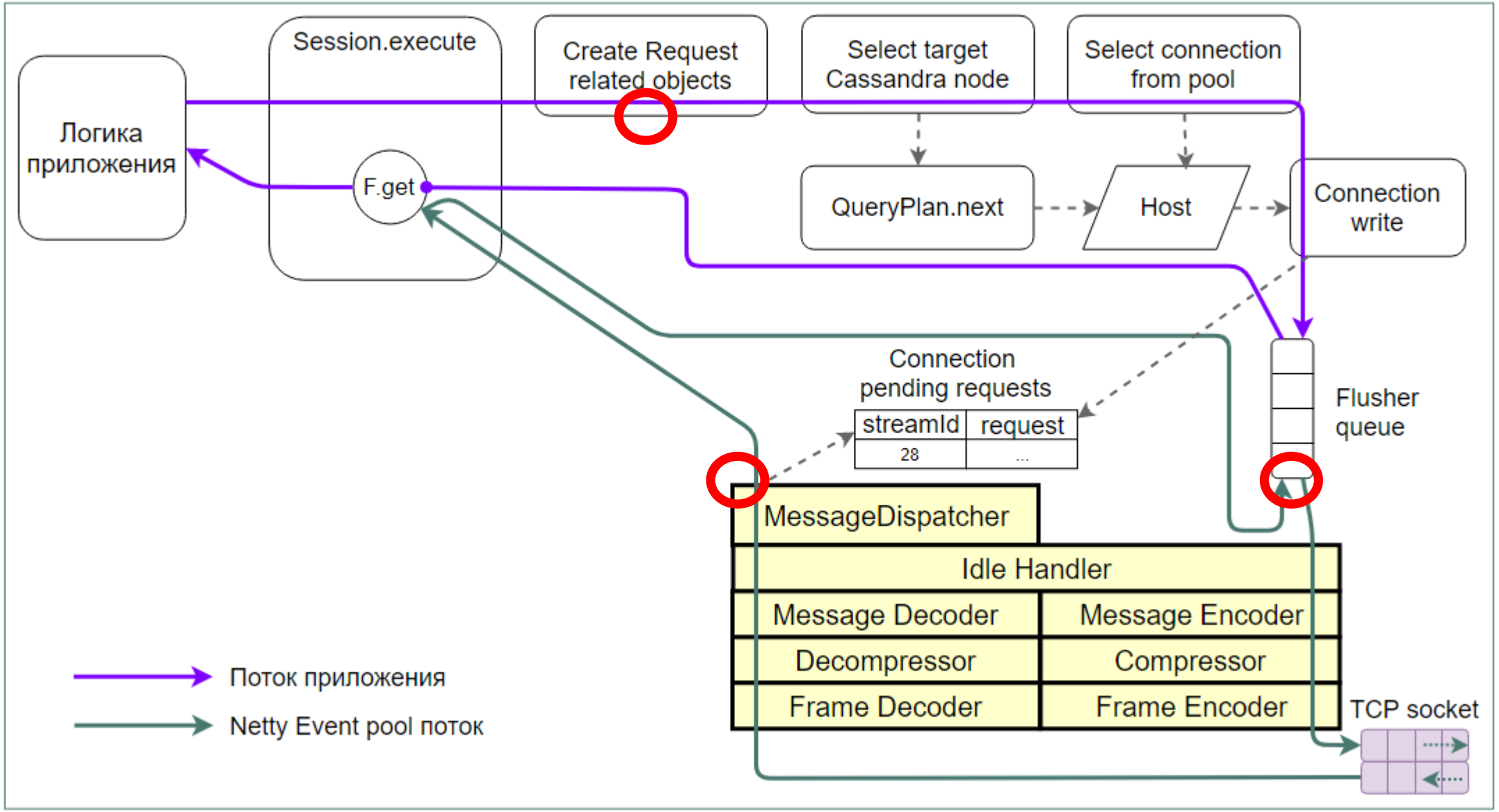
Драйвер

- Справка
  - Базовые рекомендации
- Анализ издержек на клиентской стороне
  - Формирование запроса
  - Выполнение запроса
    - Сетевое взаимодействие
    - Выделение памяти 

Выделение памяти

- https://datastax-oss.atlassian.net/browse/JAVA-1661
  - Избегаем лишних toLowerCase при работе с метаданными (имена колонок, поля UDT)
  - Доступно, начиная с версии драйвера 3.3.2
- https://github.com/datastax/java-driver/pull/1293
  - Убираем излишние выделения памяти
  - В процессе

Выделение памяти



Выделение памяти – varArgs + primitive

```
logger.trace("{}, stream {}, writing request {}",
             this, request.getStreamId(), request);
```

- int streamId
- primitive type + varargs

Class	Averag...	TLABs	Pressure
java.lang.Object[]	37 bytes	547	5,61%
io.netty.buffer.SlicedAbstractByteBuf	48 bytes	657	4,45%
com.datastax.driver.core.RequestHandler	88 bytes	132	4,36%
java.util.concurrent.locks.ReentrantLock$NonfairS...	32 bytes	116	3,74%
java.util.concurrent.CopyOnWriteArrayList	24 bytes	110	3,58%
java.util.RegularEnumSet	32 bytes	421	3,56%
io.netty.channel.DefaultChannelPromise	40 bytes	494	3,35%
java.lang.Integer	16 bytes	262	3,05%
com.datastax.driver.core.BoundStatement	80 bytes	90	3,03%

Stack Trace

Stack Trace	TLABs	Total TLAB Si...	Pressure
java.lang.Integer.valueOf(int) line: 832	262	19,31 MB	100,00%
com.datastax.driver.core.Connection$Dispatcher.add(Connection$ResponseHandler) line: 1175	25	5,10 MB	26,39%
com.datastax.driver.core.Connection.write(Connection$ResponseCallback, long, boolean) line: 734	24	5,04 MB	26,11%
com.datastax.driver.core.Connection$Dispatcher.channelRead0(ChannelHandlerContext, Message$Response) line: 1217	108	4,64 MB	24,02%
com.datastax.driver.core.Connection$10.operationComplete(ChannelFuture) line: 791	105	4,53 MB	23,49%

Выделение памяти – enum values

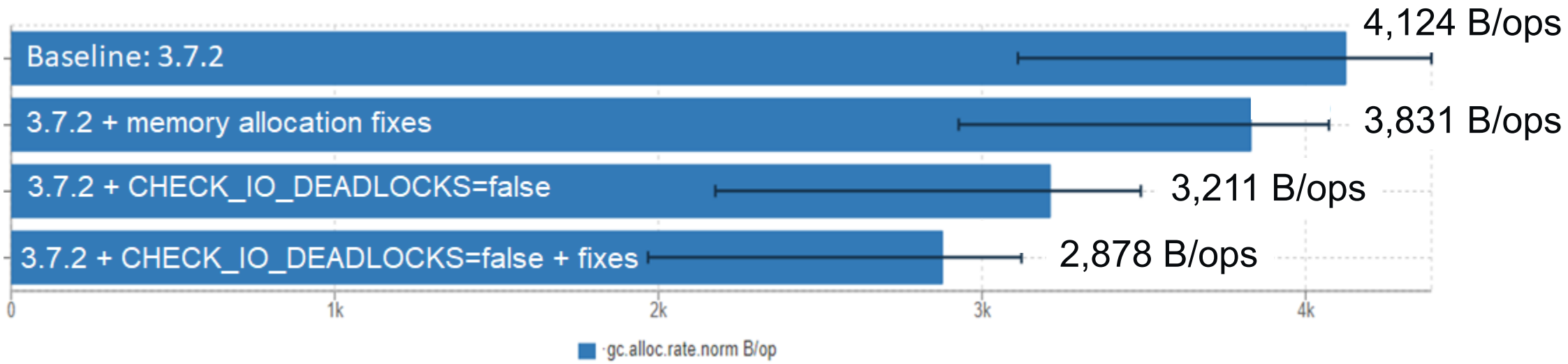
```
enum Flag { ... }

Flag[] values = Flag.values();
for (int n = 0; n < values.length; n++)
    ...
```

Stack Trace

Stack Trace	TLABs	Total TLAB Size	Pressure
com.datastax.driver.core.Frame$Header$Flag.values()	290	12,58 MB	100,00%
com.datastax.driver.core.Frame$Header$Flag.deserialize(int)	290	12,58 MB	100,00%
com.datastax.driver.core.Frame$Header.<init>(ProtocolVersion, int, int, int)	290	12,58 MB	100,00%

Выделение памяти



- Тест: SELECT, возвращающий 1 строку, GC allocation rate per operation
  - Driver 3.7.2 OOB
  - Driver 3.7.2 + memory allocation fixes
  - Driver 3.7.2 + -Dcom.datastax.driver.CHECK_IO_DEADLOCKS=false
  - Driver 3.7.2 + fixes + -Dcom.datastax.driver.CHECK_IO_DEADLOCKS=false

Драйвер

- Справка
  - Базовые рекомендации
- Анализ издержек на стороне драйвера
  - Формирование запроса
  - Выполнение запроса
    - Сетевое взаимодействие
    - Выделение памяти

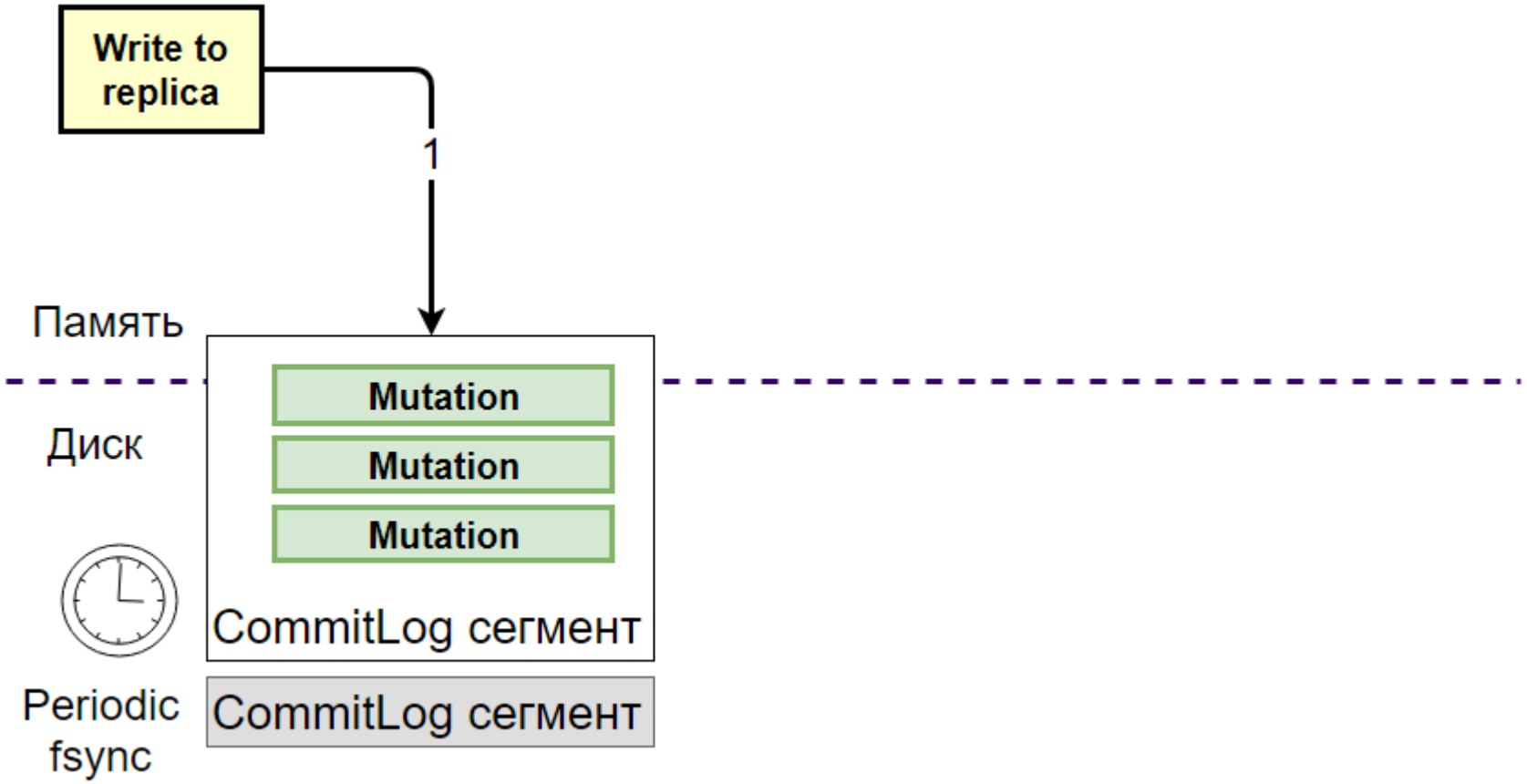
Акт 2

Сервер, запись

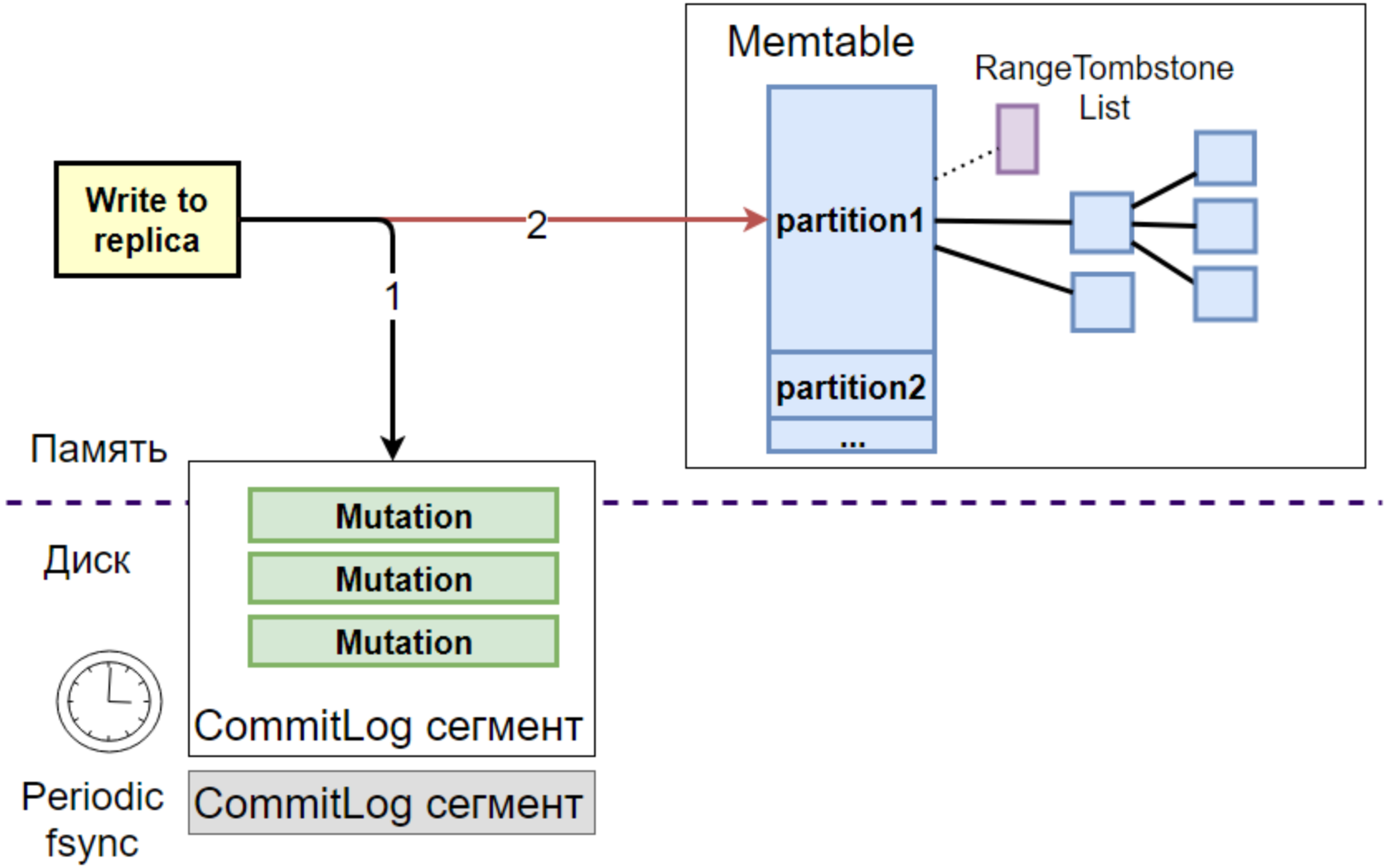
Запись

- Справка
  - Запись – что происходит внутри реплики ?

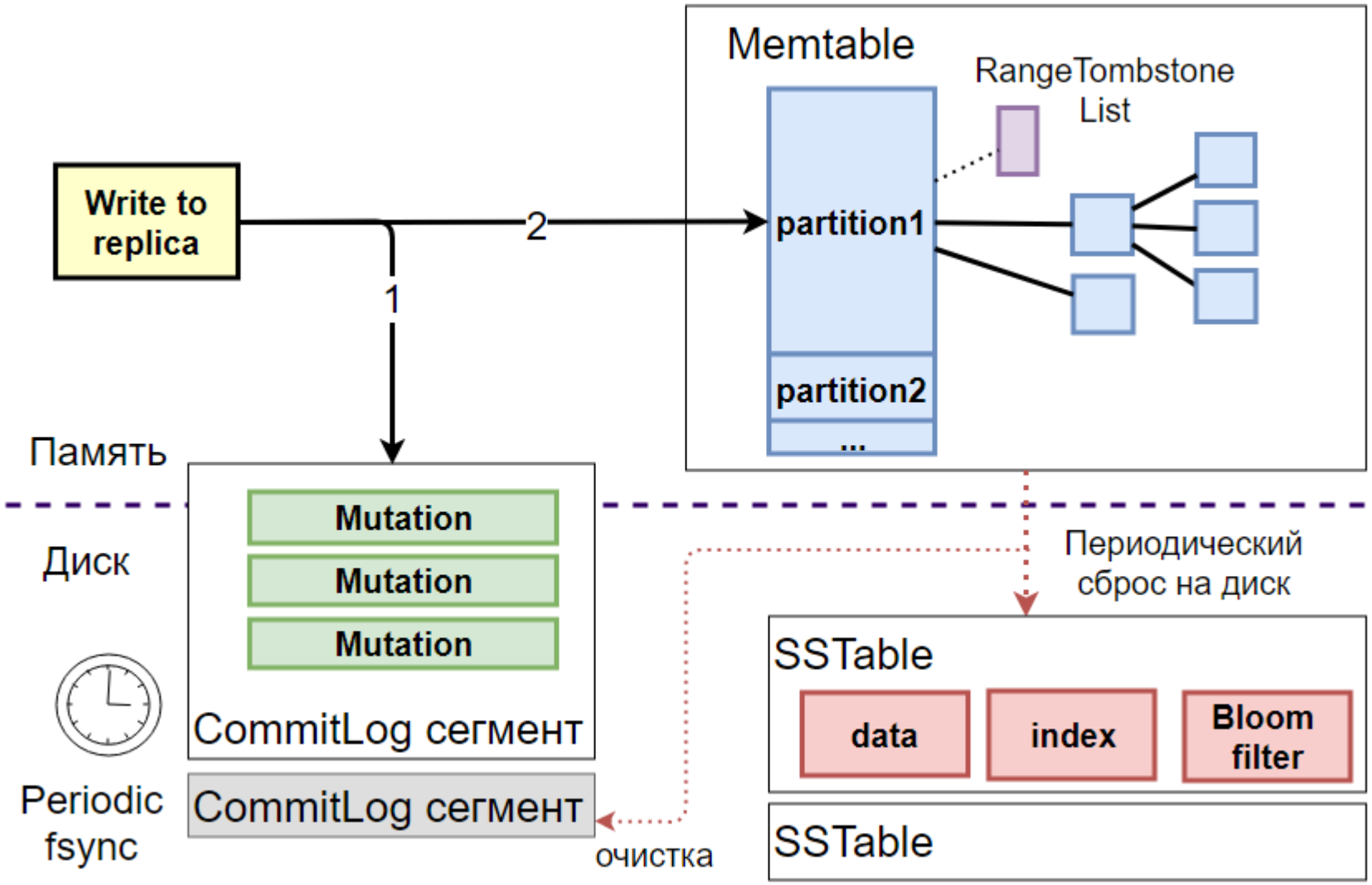
Запись – что происходит внутри реплики



Запись – что происходит внутри реплики



Запись – что происходит внутри реплики



Запись – что происходит внутри реплики

- Более детальная схема процесса записи
  - с деталями об очередях и потоках
  - с отображением на точки трассировки запросов

  есть в дополнительных слайдах к этому докладу

Запись

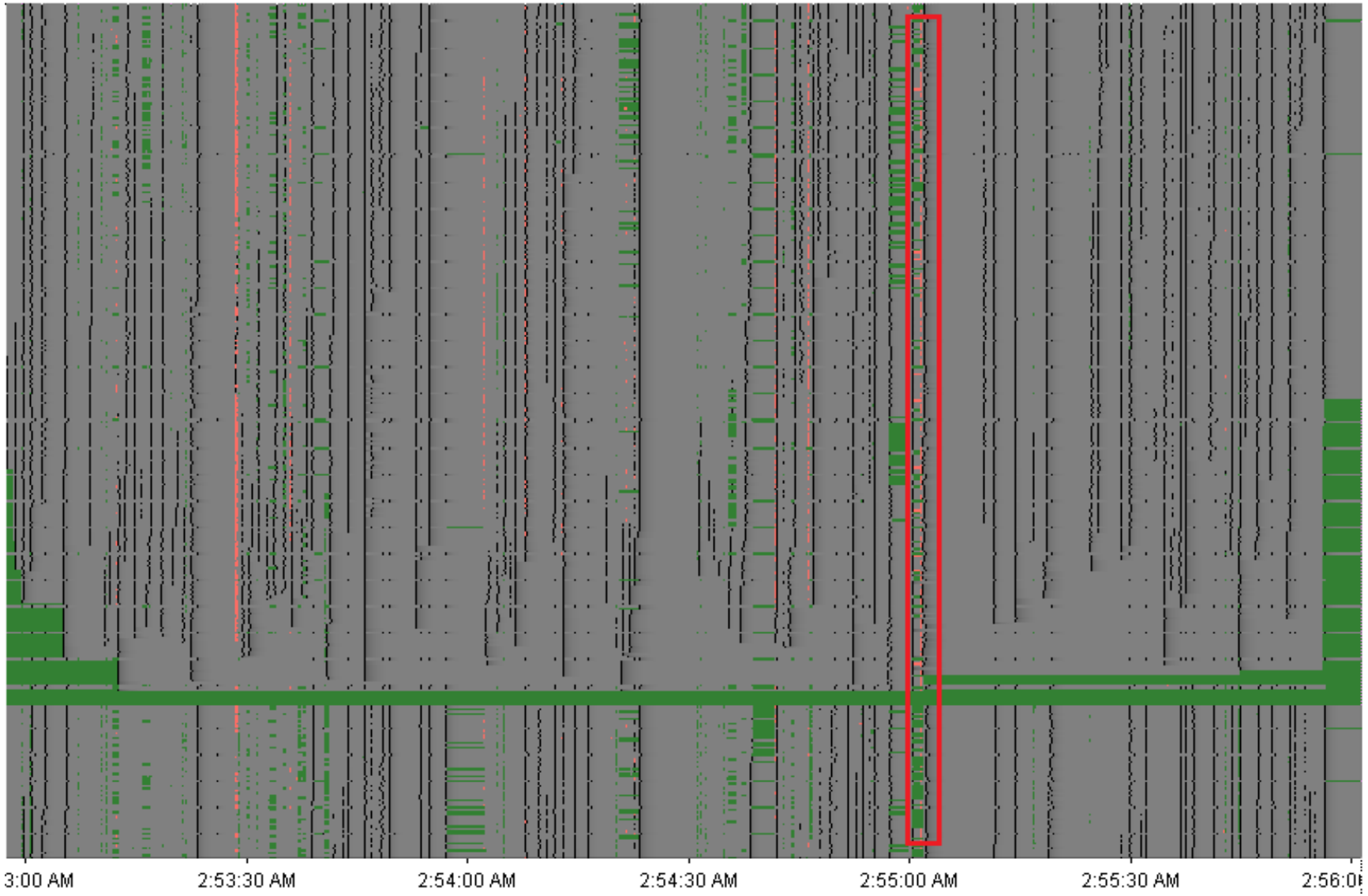
- Справка
  - Запись – что происходит внутри реплики
- Проблемы
  - Проблема при вставке списка ⚠

Проблема при вставке списка

- Делаем вставку событий в таблицу, одна из колонок - список
- Проблемы:
  - Запрос плохо масштабируется
  - Слабая утилизация ресурсов (CPU/disk/network)

Проблема при вставке списка

- JFR, диаграмма потоков



Проблема при вставке списка

Top Blocking Locks | Top Blocked Threads | Top Blocking Threads

Filter Column: Class

Class	Count	Average	Longest	Duration
org.apache.cassandra.utils.UUIDGen	23 036	95 ms 860 µs	1 s 880 ms	**36 min 48 s 232 ms**
java.lang.ref.Reference$Lock	12	219 ms 614 µs	409 ms 456 µs	**2 s 635 ms**
int[]	57	24 ms 237 µs	393 ms 539 µs	**1 s 381 ms**
java.util.Vector	15	31 ms 928 µs	47 ms 159 µs	**478 ms 930 µs**
org.apache.cassandra.db.commitlog.MemoryMappedSegment	2	20 ms 387 µs	27 ms 777 µs	**40 ms 775 µs**

Stack Trace	Count	Duration
org.apache.cassandra.utils.UUIDGen.createTimeSafe()	**23 036**	**36 min 48 s 232 ms**
org.apache.cassandra.utils.UUIDGen.getTimeUUIDBytes()	**23 032**	**36 min 48 s 54 ms**
org.apache.cassandra.cql3.Lists$Appender.doAppend(Term$Terminal, ColumnDefinition, UpdateParameters)	**23 032**	**36 min 48 s 54 ms**
org.apache.cassandra.cql3.Lists$Setter.execute(DecoratedKey, UpdateParameters)	**23 032**	**36 min 48 s 54 ms**
org.apache.cassandra.utils.UUIDGen.getTimeUUID()	**4**	**177 ms 650 µs**

UUIDGen - конфликты при захвате блокировки

```
"SharedPool-Worker-296" #6150 daemon waiting for monitor entry [0x00007f3195a2a000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at org.apache.cassandra.utils.UUIDGen.createTimeSafe(UUIDGen.java:299)
        - waiting to lock <0x00007f9b6a0009c0> (a org.apache.cassandra.utils.UUIDGen)
        at org.apache.cassandra.utils.UUIDGen.getTimeUUIDBytes(UUIDGen.java:167)
        at org.apache.cassandra.cql3.Lists$Appender.doAppend(Lists.java:396)
        at org.apache.cassandra.cql3.Lists$Setter.execute(Lists.java:302)
        at org.apache.cassandra.cql3.statements.UpdateStatement.addUpdateForKey(UpdateStatement.java:94)
```

UUIDGen - конфликты при захвате блокировки

```
INSERT INTO events (partition_key, clustering_key, param_list)
    VALUES (?,?,?)
```

- param_list = [e1, …, eN] - список

UUIDGen - конфликты при захвате блокировки

```
INSERT INTO events (partition_key, clustering_key, param_list)
    VALUES (?,?,?)
```

- param_list = [e1, …, eN] - список
- Коллекции в Cassandra могут быть frozen и non-frozen:
  - Frozen – всегда перезаписываются целиком, как blob
  - Non-frozen – могут быть обновлены инкрементально

UUIDGen - конфликты при захвате блокировки

```
INSERT INTO events (partition_key, clustering_key, param_list)
    VALUES (?,?,?)
```

- param_list = [e1, …, eN] - список
- Коллекции в Cassandra могут быть frozen и non-frozen:
  - Frozen – всегда перезаписываются целиком, как blob
  - Non-frozen – могут быть обновлены инкрементально
- При добавлении элементов в non-frozen список Cassandra присваивает им внутренний идентификатор - CellPath в формате UUID

UUIDGen - конфликты при захвате блокировки

```
INSERT INTO events (partition_key, clustering_key, param_list)
    VALUES (?,?,?)
```

- param_list = [e1, …, eN] - список
- Коллекции в Cassandra могут быть frozen и non-frozen:
  - Frozen – всегда перезаписываются целиком, как blob
  - Non-frozen – могут быть обновлены инкрементально
- При добавлении элементов в non-frozen список Cassandra присваивает им внутренний идентификатор - CellPath в формате UUID
- UUID – генерируется при вставке на стороне сервера
- UUID – на основе времени (type 1 UUID)

UUIDGen - конфликты при захвате блокировки

- UUIDGen – конфликт при захвате блокировки (lock contention)
- CASSANDRA-11517
- Проблема есть, начиная с 3.0.x
- Исправлено, начиная с 3.8
- А если нельзя обновиться? – попробовать использовать frozen<list>

Запись

- Справка
  - Запись – что происходит внутри реплики
- Проблемы
  - Проблема при вставке списка
  - Скрытая стоимость TTL ⚠

Скрытая стоимость TTL

- Делаем вставку событий в таблицу, используя TTL
- Читаем данные по partition ключу и подмножеству clustering ключей
- Симптомы:
  - Чтения со временем замедляются
  - CPU на стороне сервера и возрастает

Скрытая стоимость TTL

```
INSERT INTO test_tombstones (
    partition_key1, partition_key2,
    clustering_key1, clustering_key2,
    value1, value2, list_value)
VALUES (?,?,?,?,?,?) USING TTL 1
```

- TTL = 1 (1 секунда) - просто для примера
- flush
- смотрим получилось в SSTable через sstabledump

Скрытая стоимость TTL

```
"clustering" : [ "clustering_key1-4", "clustering_key2-4" ],
"liveness_info" : {
  "tstamp" : "13:43:07.787Z",
  "ttl" : 1,
  "expires_at" : "13:43:08Z",
  "expired" : true
},
"cells" : [
  { "name" : "value1", "value" : "value1-4" },
  { "name" : "value2", "value" : "value2-4" },
  { "name" : "list_value",
    "deletion_info" : {
      "marked_deleted" : "13:43:07.786999Z",
      "local_delete_time" : "13:43:07Z"
    }
  },
  { "name" : "list_value", "path" : [ "SOME UUID" ], "value" : "a" },
  { "name" : "list_value", "path" : [ "SOME UUID" ], "value" : "b" },
  { "name" : "list_value", "path" : [ "SOME UUID" ], "value" : "c" }
]
```

вроде все прилично…

Скрытая стоимость TTL

Прошел compaction – на каждой ячейке выросли cell tombstones:

```
"clustering" : [ "clustering_key1-4", "clustering_key2-4" ],
"liveness_info" : {
  "tstamp" : "13:43:07.787Z",
  "ttl" : 1,
  "expires_at" : "13:43:08Z",
  "expired" : true
},
"cells" : [
  { "name" : "value1", "deletion_info" : { "local_delete_time" : "13:43:07Z" }},
  { "name" : "value2", "deletion_info" : { "local_delete_time" : "13:43:07Z" }},
  { "name" : "list_value", "deletion_info" :
      { "marked_deleted" : "13:43:07.786999Z",
        "local_delete_time" : "13:43:07Z"
      }
  },
  { "name" : "list_value", "path" : [ "..." ], "deletion_info" : { "local_delete_time" : "13:43:07Z" }},
  { "name" : "list_value", "path" : [ "..." ], "deletion_info" : { "local_delete_time" : "13:43:07Z" }},
  { "name" : "list_value", "path" : [ "..." ], "deletion_info" : { "local_delete_time" : "13:43:07Z" }}
]
```

Скрытая стоимость TTL

- Концептуально – row TTL в Cassandra обрабатываются на уровне cell (link)

Скрытая стоимость TTL

- Концептуально – row TTL в Cassandra обрабатываются на уровне cell (link)
- Non-frozen коллекция –> еще + 1 tombstone на каждый элемент

Скрытая стоимость TTL

- Концептуально – row TTL в Cassandra обрабатываются на уровне cell (link)
- Non-frozen коллекция –> еще + 1 tombstone на каждый элемент
- Почему плохо:
  - Больше ресурсов требуется на объединение строк при чтении
  - Больше ресурсов требуется на объединение строк при compaction
  - Больше места на диске

Скрытая стоимость TTL

Что можно сделать:

- Не использовать без необходимости связку row TTL + non-frozen коллекции

Скрытая стоимость TTL

Что можно сделать:

- Не использовать без необходимости связку row TTL + non-frozen коллекции
- Если есть возможность – использовать table level TTL
  - Работает через удаление SSTables
  - Но TTL – одинаковые для всех записей в таблице

Запись

- Справка
  - Запись – что происходит внутри реплики
- Проблемы
  - Проблема при вставке списка
  - Скрытая стоимость TTL
  - WriteTimeoutException при удалении ⚠

WriteTimeoutException при удалении

- Вставляем и удаляем данные из различных приложений
- Симптомы
  - WriteTimeoutException на стороне различных приложений, для разных keyspaces/tables
  - Высокая write latency у одной из таблиц

WriteTimeoutException при удалении

- Вставляем и удаляем данные из различных приложений
- Симптомы
  - WriteTimeoutException на стороне различных приложений, для разных keyspaces/tables
  - Высокая write latency у одной из таблиц
  - Метрики:

```
./nodetool tpstats
Pool Name                    Active   Pending      Completed   Blocked
MutationStage                    32     20926       32405275         0
```

WriteTimeoutException при удалении

- Вставляем и удаляем данные из различных приложений
- Симптомы
  - WriteTimeoutException на стороне различных приложений, для разных keyspaces/tables
  - Высокая write latency у одной из таблиц
  - Метрики:

```
./nodetool tpstats
Pool Name                    Active   Pending      Completed   Blocked
MutationStage                    32     20926       32405275         0

JMX:
...:type=ThreadPools,name=ActiveTasks,path=internal,scope=*
...:type=ThreadPools,name=PendingTasks,path=internal,scope=*
...:keyspace=*,name=WriteLatency,scope=*,type=Table
```

WriteTimeoutException при удалении

Thread dump:

```
"SharedPool-Worker-10" #363 daemon waiting for monitor entry [0x00007f4ced125000]
   java.lang.Thread.State: BLOCKED (on object monitor)
            at sun.misc.Unsafe.monitorEnter(Native Method)
            at org.apache.cassandra.utils.concurrent.Locks.monitorEnterUnsafe(Locks.java:46)
            at org.apache.cassandra.db.partitions.AtomicBTreePartition.addAllWithSizeDelta(…)
            at org.apache.cassandra.db.Memtable.put(Memtable.java:254)
```

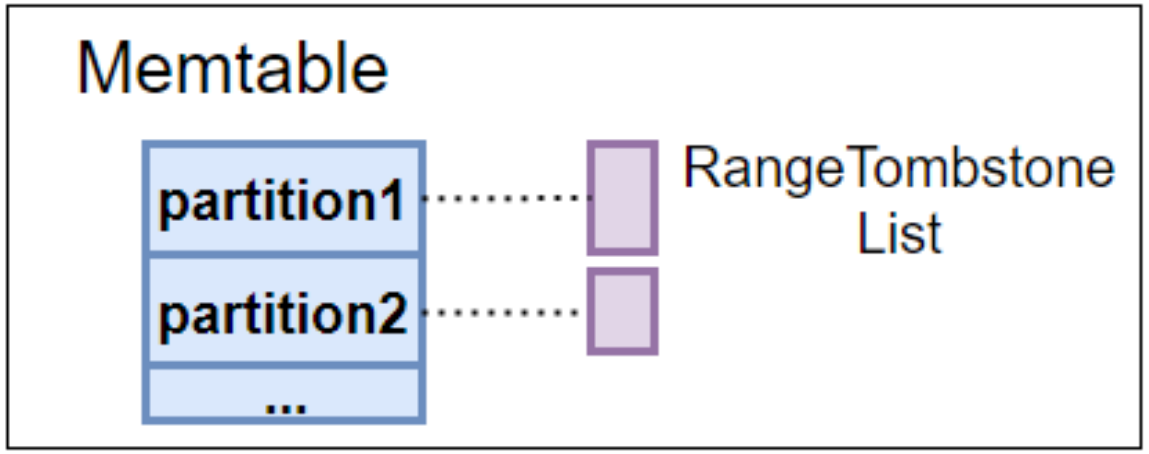
```
"SharedPool-Worker-32" #383 daemon runnable [0x00007f4cecc51000]
   java.lang.Thread.State: RUNNABLE
            at org.apache.cassandra.db.ClusteringComparator.compare(ClusteringComparator.java:137)
            at org.apache.cassandra.db.RangeTombstoneList.insertFrom(RangeTombstoneList.java:536)
            at org.apache.cassandra.db.RangeTombstoneList.add(RangeTombstoneList.java:167)
            at org.apache.cassandra.db.RangeTombstoneList.addAll(RangeTombstoneList.java:207)
            at org.apache.cassandra.db.MutableDeletionInfo.add(MutableDeletionInfo.java:141)
            at org.apache.cassandra.db.partitions.AtomicBTreePartition.addAllWithSizeDelta(...)
            at org.apache.cassandra.db.Memtable.put(Memtable.java:254)
```

WriteTimeoutException при удалении

Удаление диапазона clustering ключей:

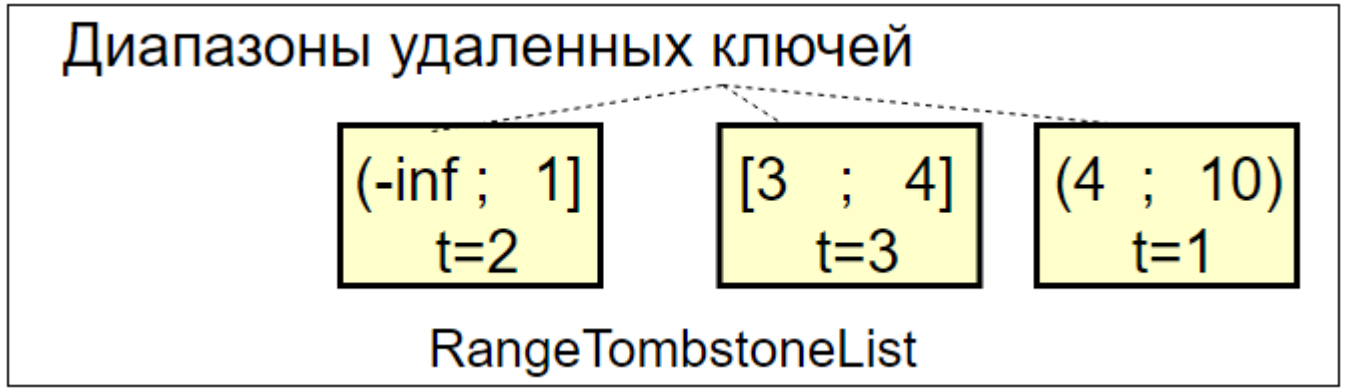
```
DELETE FROM test_range_tombstone_table
  WHERE part_key = ? AND clust_key < ?
```

WriteTimeoutException при удалении



- RangeTombstoneList – структура данных, используемая для хранения range tombstones
- Отдельная структура для каждого partition key

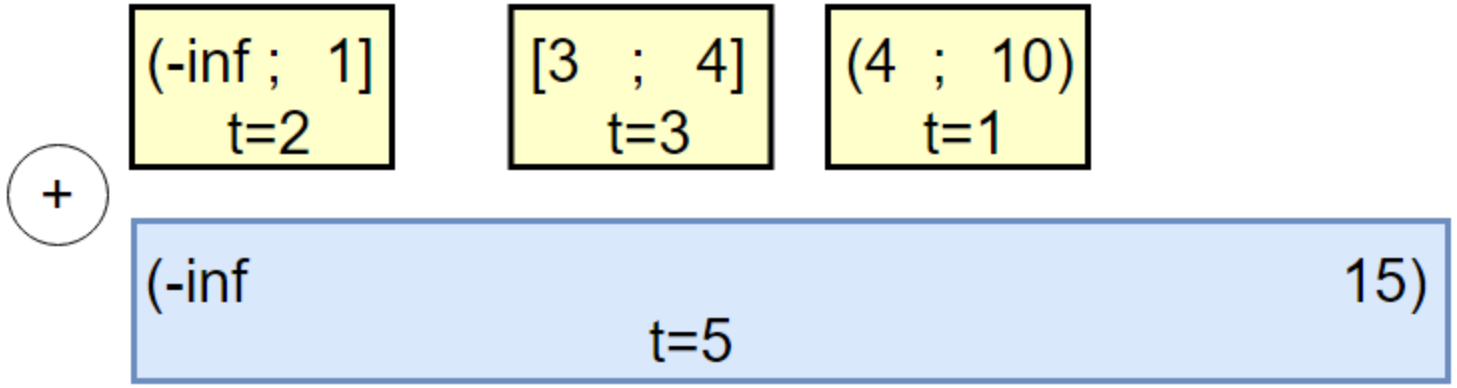
WriteTimeoutException при удалении



- Упорядоченный список интервалов
- У интервалов есть временная метка

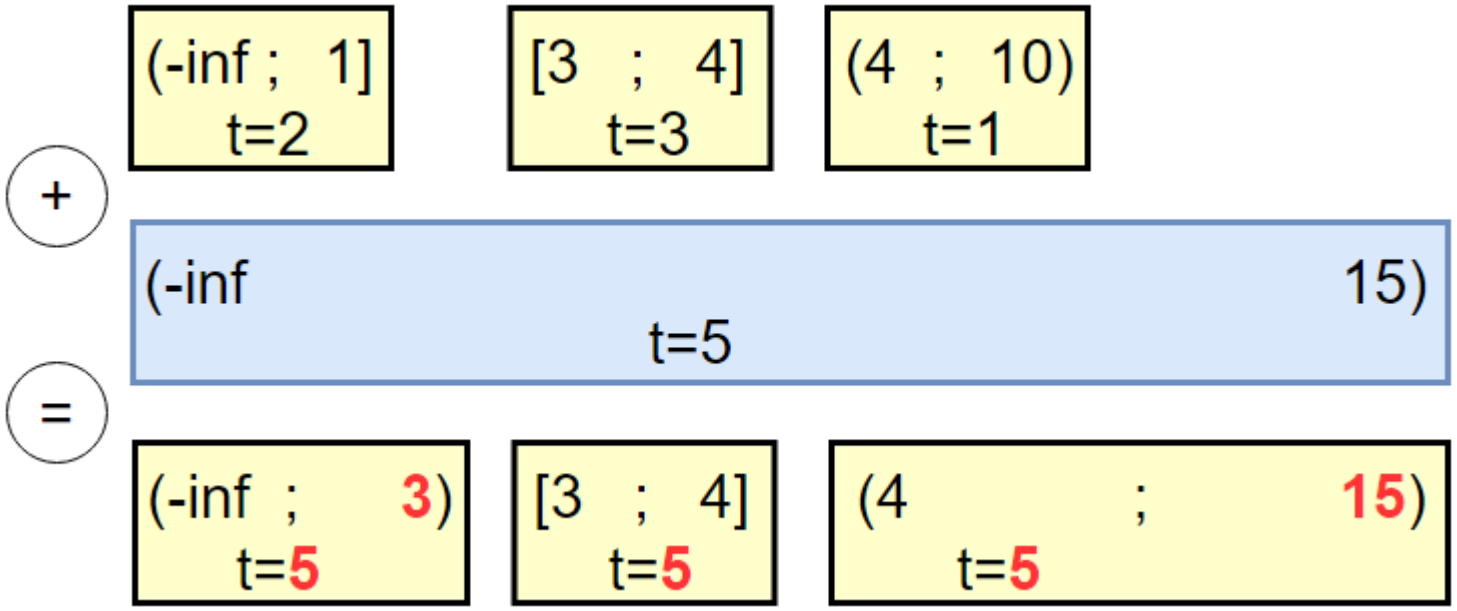
WriteTimeoutException при удалении

- При добавлении “большего” интервала – обновляются все существующие интервалы (но укрупнения не происходит)



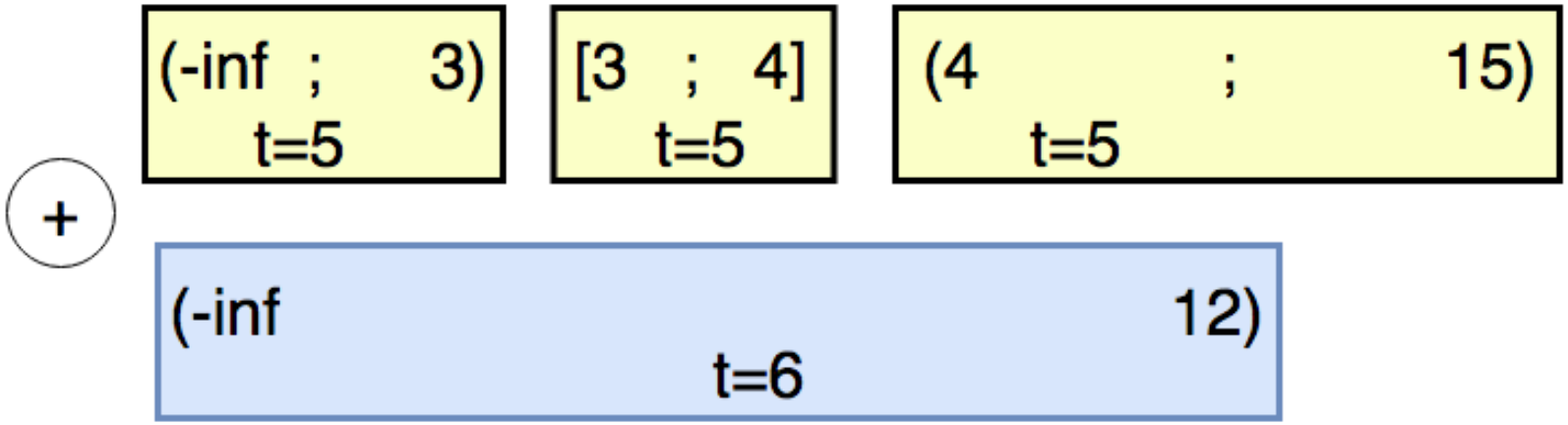
WriteTimeoutException при удалении

- При добавлении “большего” интервала – обновляются все существующие интервалы (но укрупнения не происходит)



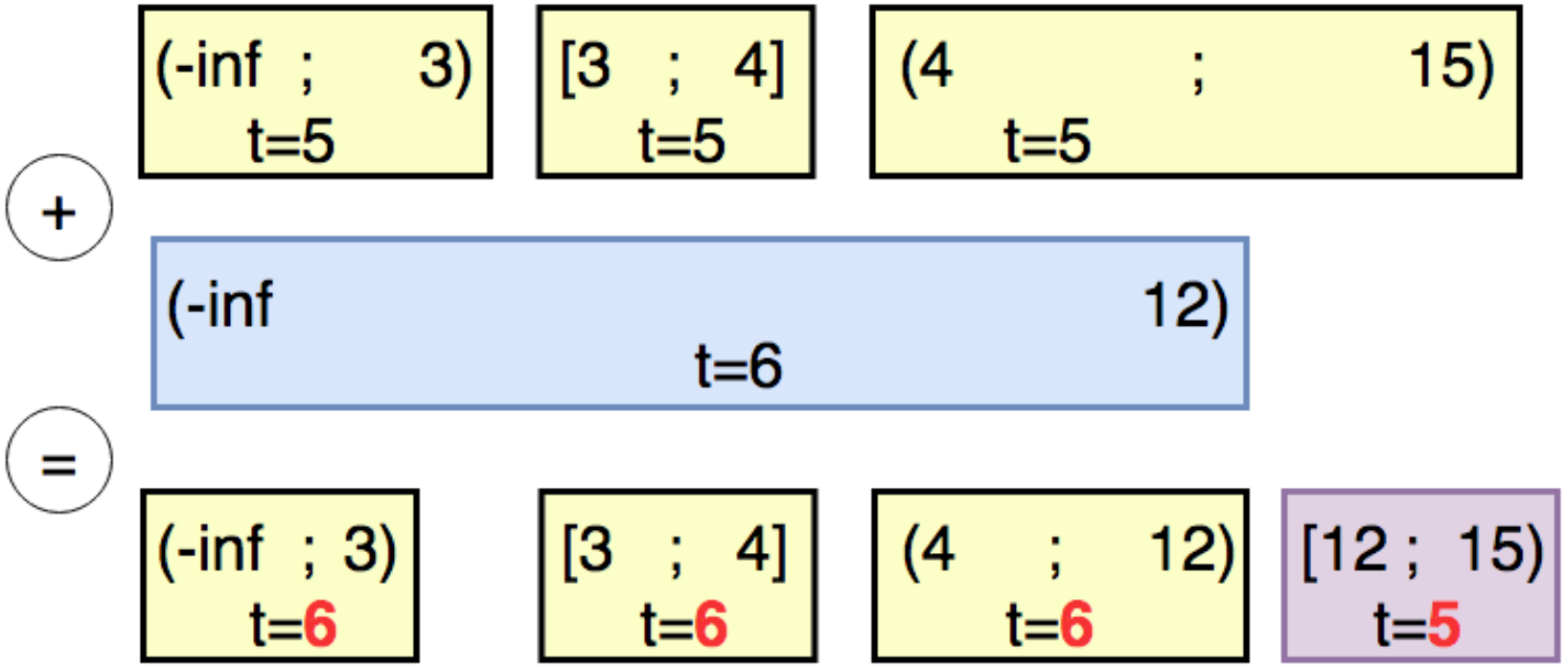
WriteTimeoutException при удалении

- При добавлении “меньшего” интервала – добавляется новый интервал



WriteTimeoutException при удалении

- При добавлении “меньшего” интервала – добавляется новый интервал



WriteTimeoutException при удалении

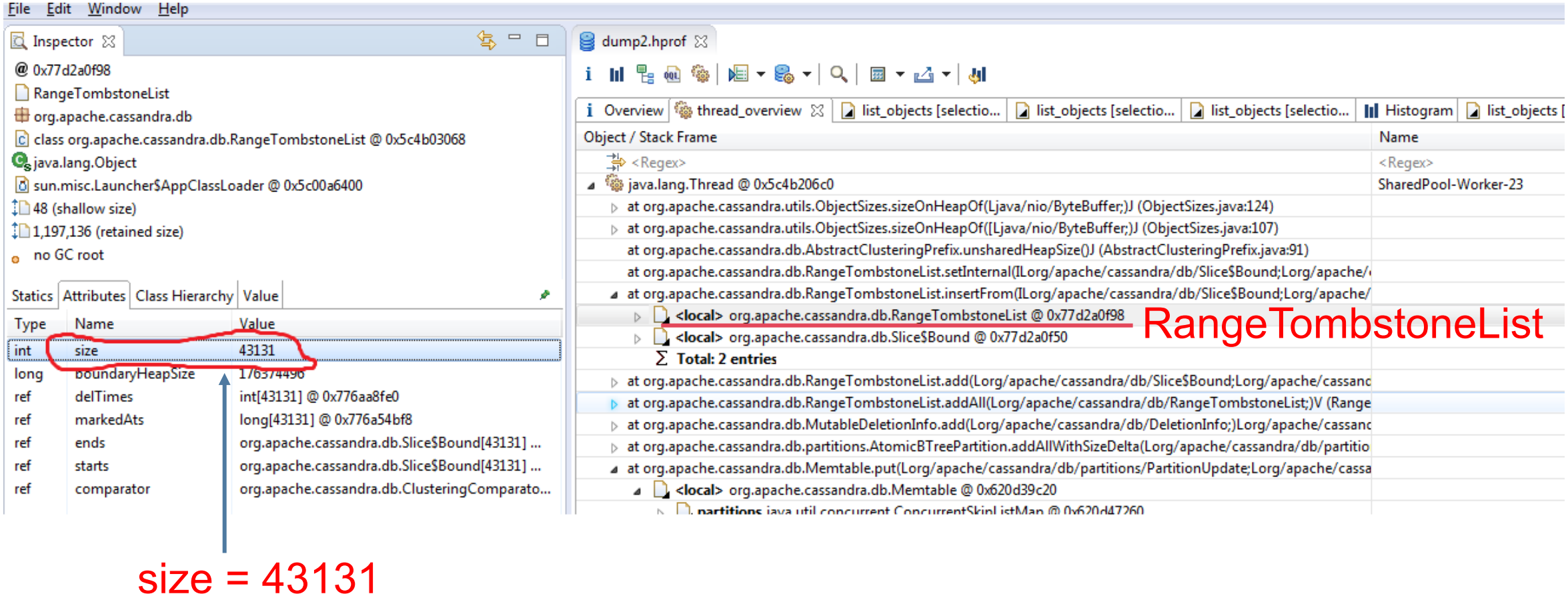
- При добавлении “меньшего” интервала – добавляется новый интервал
- При добавлении “большего” интервала – обновляются все существующие интервалы

WriteTimeoutException при удалении

- При добавлении “меньшего” интервала – добавляется новый интервал
- При добавлении “большего” интервала – обновляются все существующие интервалы

- Проблема роста числа интервалов известна, была частично исправлена в 3.4.11, но на уровне чтения, а не записи (CASSANDRA-14894)

WriteTimeoutException при удалении



WriteTimeoutException при удалении

```
DELETE FROM test_range_tombstone_table
  WHERE part_key = ? AND clust_key < ?

clust_key = new Date(currentTimeMillis() - expirationTime);
```

WriteTimeoutException при удалении

```
DELETE FROM test_range_tombstone_table
  WHERE part_key = ? AND clust_key < ?

clust_key = new Date(currentTimeMillis() - expirationTime);
```

- Несколько процессов –> clust_key значения могут быть не строго монотонными (граница интервала плавает)
- В результате – число интервалов в RangeTombstoneList растет

WriteTimeoutException при удалении

```
DELETE FROM test_range_tombstone_table
  WHERE part_key = ? AND clust_key < ?

clust_key = new Date(currentTimeMillis() - expirationTime);
```

- Несколько процессов –> clust_key значения могут быть не строго монотонными (граница интервала плавает)
- В результате – число интервалов в RangeTombstoneList растет

- Как лечить?
  - Один из путей: делать более мелкие партиции

Запись

- Справка
  - Запись – что происходит внутри реплики
- Проблемы
  - Проблема при вставке списка
  - Скрытая стоимость TTL
  - WriteTimeoutException при удалении

Акт 2

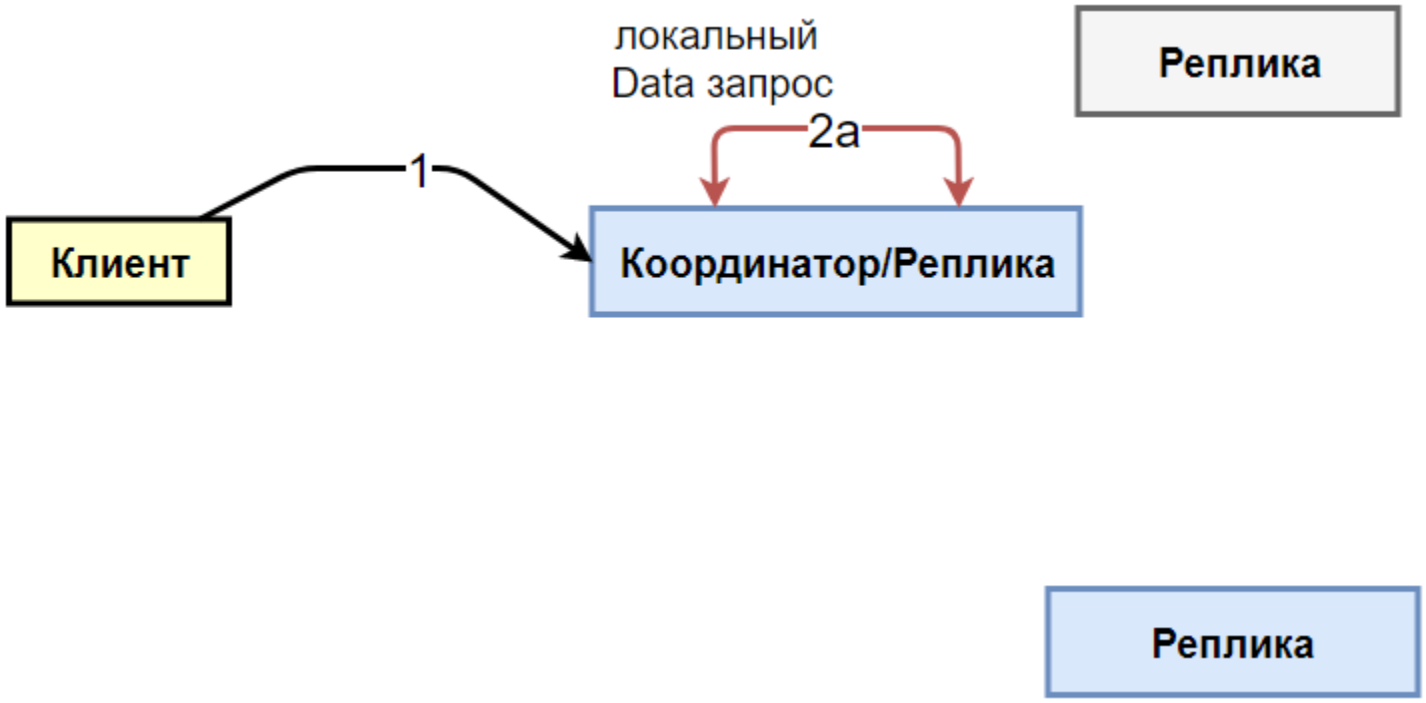
Сервер, чтение

Чтение

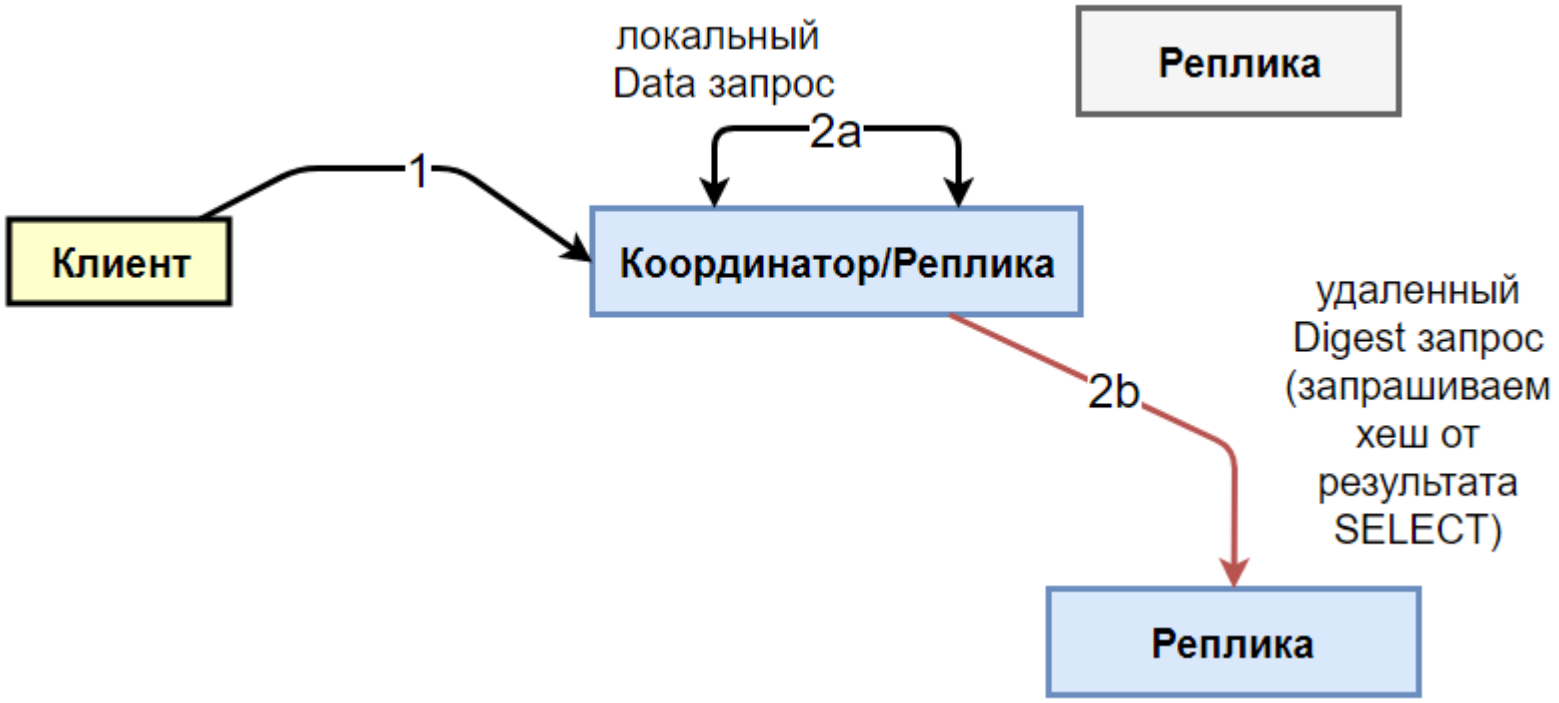
- Справка
  - Чтение – уровень кластера ?

Чтение – уровень кластера

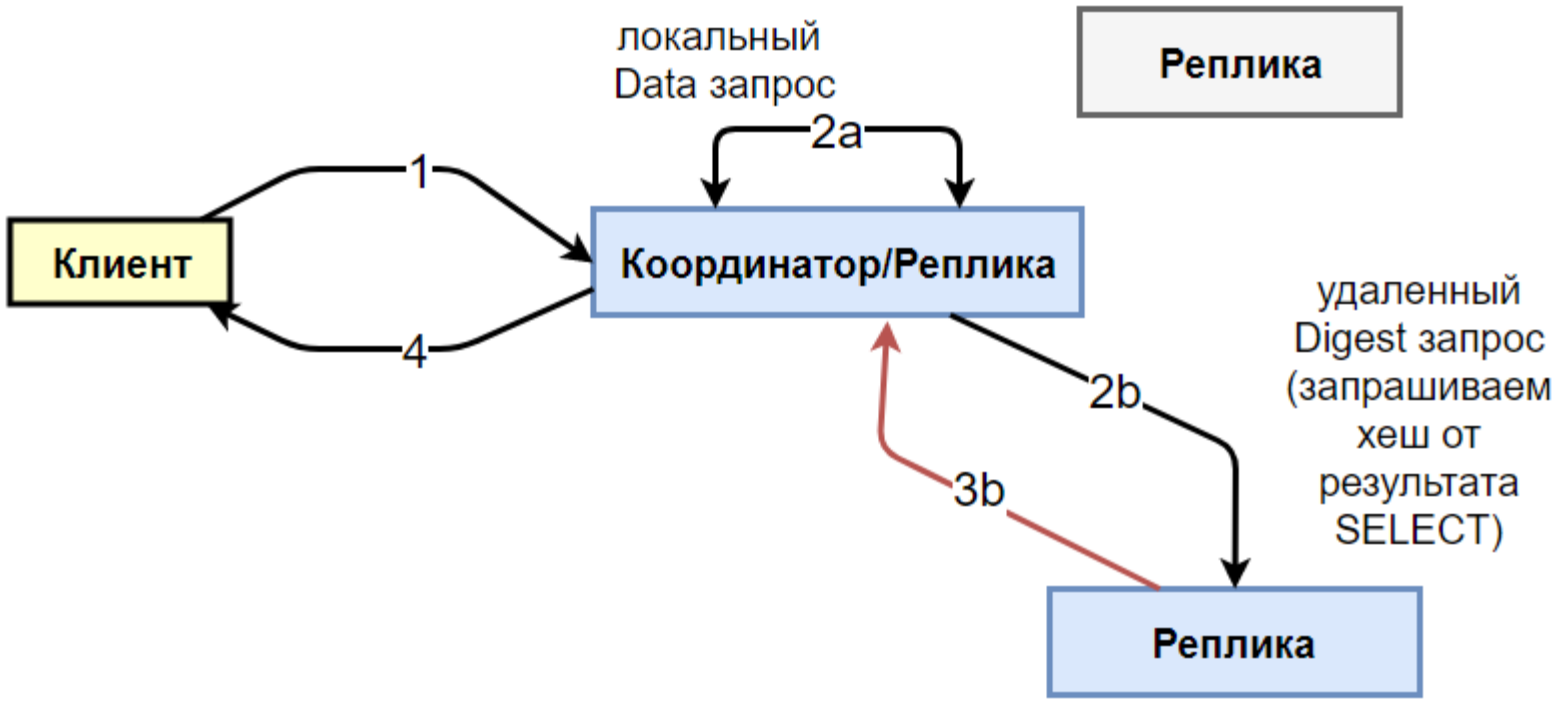
Чтение – уровень кластера



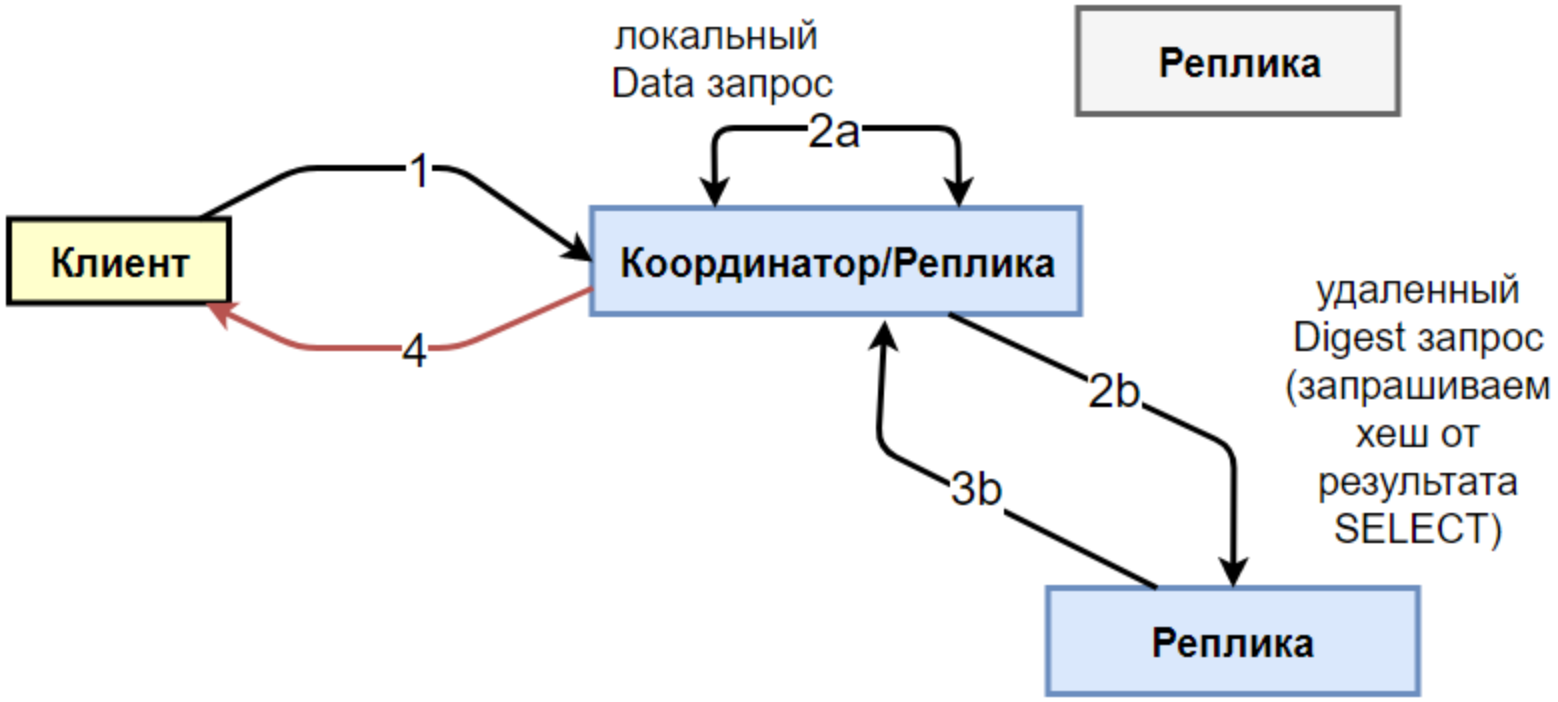
Чтение – уровень кластера



Чтение – уровень кластера



Чтение – уровень кластера



Чтение

- Справка
  - Чтение – уровень кластера
- Проблемы
  - Количество колонок и стоимость чтения 

Количество колонок и стоимость чтения

- Вопрос: есть ли зависимость между скоростью чтения и числом читаемых колонок из таблицы?
- Зачем: хотим уменьшить нагрузку на базу и снизить время отклика

Количество колонок и стоимость чтения

- Вопрос: есть ли зависимость между скоростью чтения и числом читаемых колонок из таблицы?
- Зачем: хотим уменьшить нагрузку на базу и снизить время отклика
- Интересующие опции:
  - Колонки из кластерного ключа
  - Колонки со значениями

Количество колонок и стоимость чтения

- Эксперимент:
  - Делаем таблицы с разным числом строковых колонок
  - Сохраняем суммарно один и тот же объем данных в строку

Количество колонок и стоимость чтения

- Эксперимент:
  - Делаем таблицы с разным числом строковых колонок
  - Сохраняем суммарно один и тот же объем данных в строку
  - В партиции – 10 строк
  - Фиксированная нагрузка: 6000 чтений в секунду, читаем партицию целиком

Количество колонок и стоимость чтения

- Эксперимент:
  - Делаем таблицы с разным числом строковых колонок
  - Сохраняем суммарно один и тот же объем данных в строку
  - В партиции – 10 строк
  - Фиксированная нагрузка: 6000 чтений в секунду, читаем партицию целиком

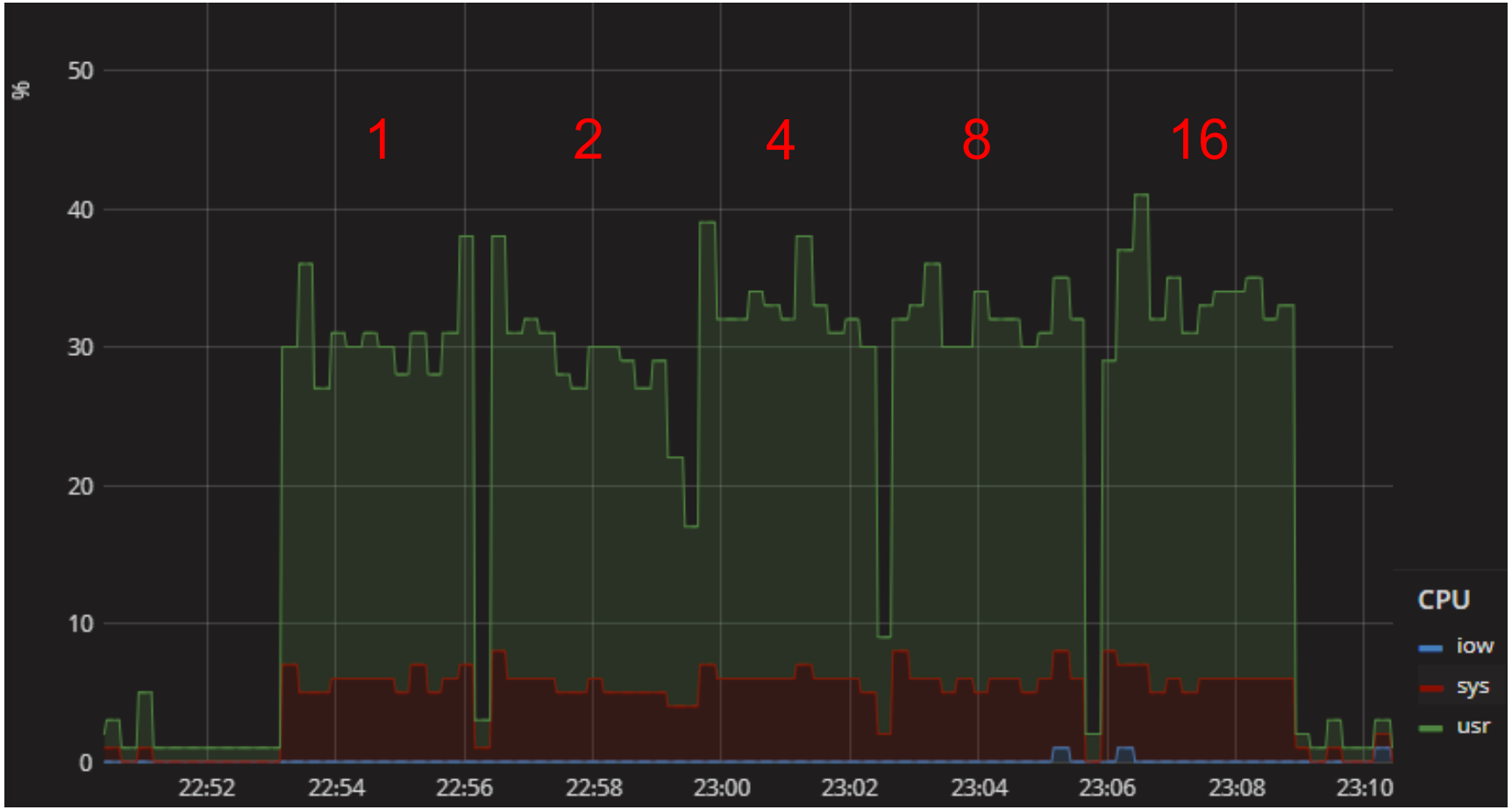
  - Кластерный ключ:
    - 1*64 байт, 2 * 32 байт, 4*16 байт, 8 * 8 байт, 16 * 4 байт
  - Колонки со значениями :
    - 1 * 512 байт, 2 * 256 байт, 4*128 байт, 8 * 64 байт, 16 * 32 байт, 32 * 16 байт

Стоимость чтения – кластерные ключи

Число кластерных колонок	Размер колонки, байт	Среднее время ответа, мс
1	64	1.6
2	32	1.6
4	16	1.7
8	8	1.8
16	4	2.0

Стоимость чтения – кластерные ключи

- CPU

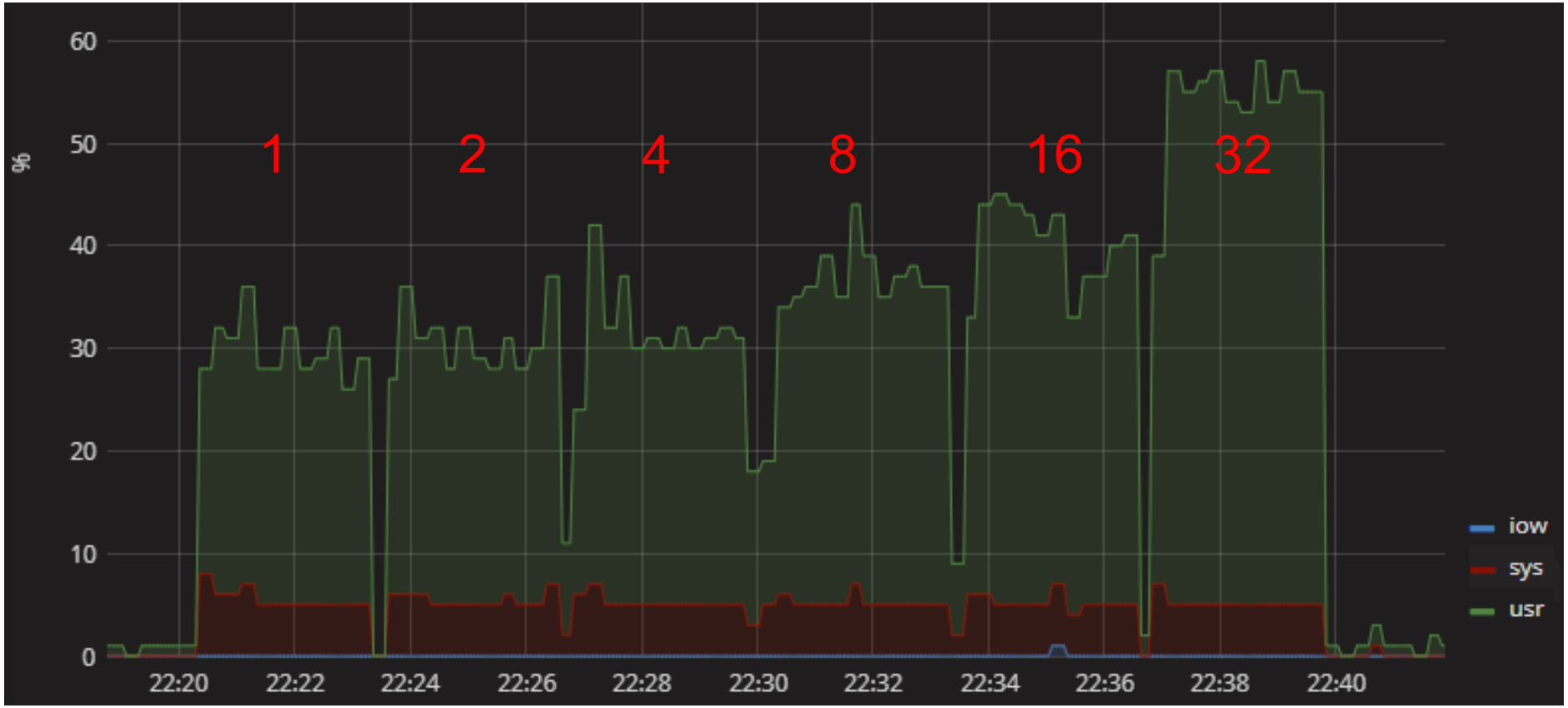


Стоимость чтения – колонки со значениями

Число колонок со значениями	Размер колонки, байт	Среднее время ответа, мс
1	512	1.3
2	256	1.4
4	128	1.6
8	64	1.8
16	32	2.1
32	16	3.4

Стоимость чтения – колонки со значениями

- CPU



Количество колонок и стоимость чтения

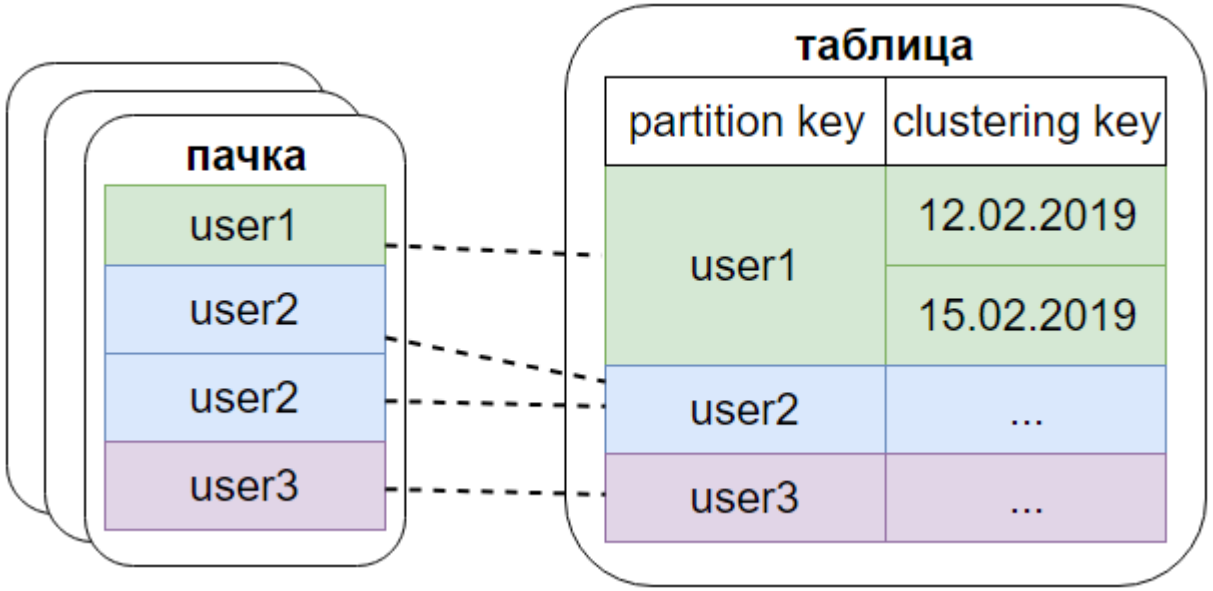
- Колонки с кластерными ключами
  - Есть рост по времени обработки и CPU
- Колонки со значением
  - Есть существенный рост по времени обработки и CPU

Чтение

- Справка
  - Чтение – уровень кластера
- Проблемы
  - Количество колонок и стоимость чтения
  - Надо прочитать набор партиций, кто быстрее? 

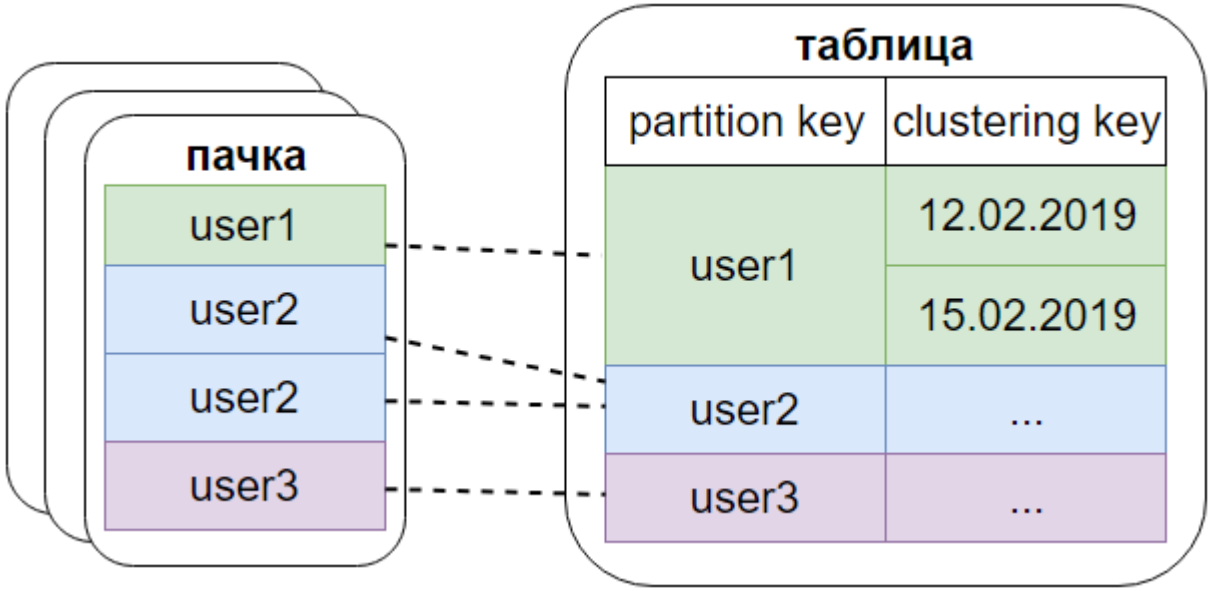
Надо прочитать набор партиций, кто быстрее?

- В рамках batch логики периодически поступают пачки входных данных
- Пачка состоит из записей, для каждой записи надо достать партицию из таблицы (“batch-table join”)



Надо прочитать набор партиций, кто быстрее?

- В рамках batch логики периодически поступают пачки входных данных
- Пачка состоит из записей, для каждой записи надо достать партицию из таблицы (“batch-table join”)



- Цель – минимизировать суммарное время чтения записей для ключей из пачки

Вариант 1 - Sync SELECT

```
for (String id : partitionIds) {
  BoundStatement boundStatement = new BoundStatement(selectStatement);
  boundStatement.setString("id", id);
  List<Row> rows = dbSession().execute(boundStatement).all();
  // process
}
```

Вариант 2 - Async SELECT

```
for (String id : partitionIds) {
  BoundStatement boundStatement = new BoundStatement(selectStatement);
  boundStatement.setString("id", id);
  ResultSetFuture future = dbSession().executeAsync(boundStatement);
  futures.add(future);
}
List<ResultSet> resultSets = Futures.successfulAsList(futures).get();
for(ResultSet resultSet : resultSets) {
  List<Row> rows = resultSet.all();
  // process
}
```

Вариант 3 - SELECT IN

- `SELECT * FROM testTable WHERE id IN ?`
- Попытка сэкономить на взаимодействиях между клиентом и сервером

Вариант 3 - SELECT IN

- `SELECT * FROM testTable WHERE id IN ?`
- Попытка сэкономить на взаимодействиях между клиентом и сервером
- Вариант напрямую, по всем ключам – плохой, координатор будет взаимодействовать с несколькими репликами для разных партиций

Вариант 3 - SELECT IN

- Стратегия: сгруппируем партиции по их репликам-владельцам
- Распределение партиций по нодам:

```
cluster.getMetadata().getReplicas(keyspace, key)
```

Вариант 3 - SELECT IN

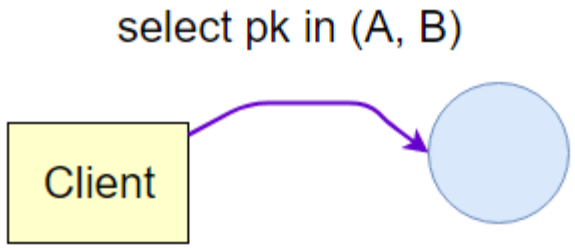
- Стратегия: сгруппируем партиции по их репликам-владельцам
- Распределение партиций по нодам:

```
cluster.getMetadata().getReplicas(keyspace, key)
```

- Максимальное число таких групп: количество вариантов выбора RF реплик из N нод = C_N^{RF}. Для 5 нод и RF = 3 => 10
- Для каждой группы - отдельный SELECT IN
- ! Из минусов – больше нагрузка на координатор

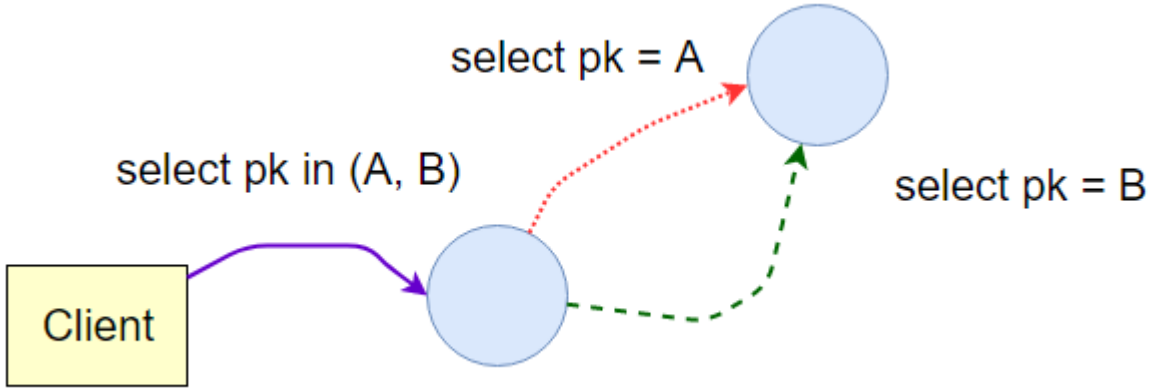
Вариант 4 - SELECT IN with LIMIT

- Делаем предварительные замеры – получаем скорость, аналогичную select sync
- Смотрим по коду Cassandra , что происходит внутри:



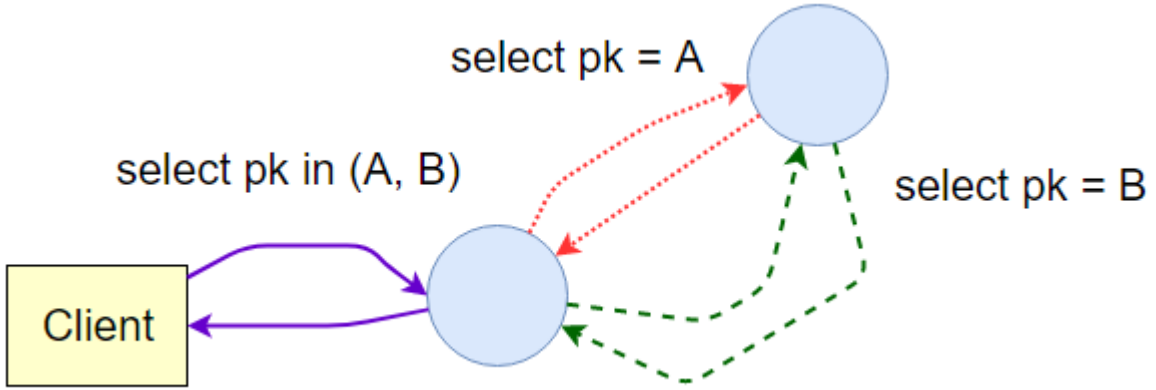
Вариант 4 - SELECT IN with LIMIT

- Делаем предварительные замеры – получаем скорость, аналогичную select sync
- Смотрим по коду Cassandra , что происходит внутри:



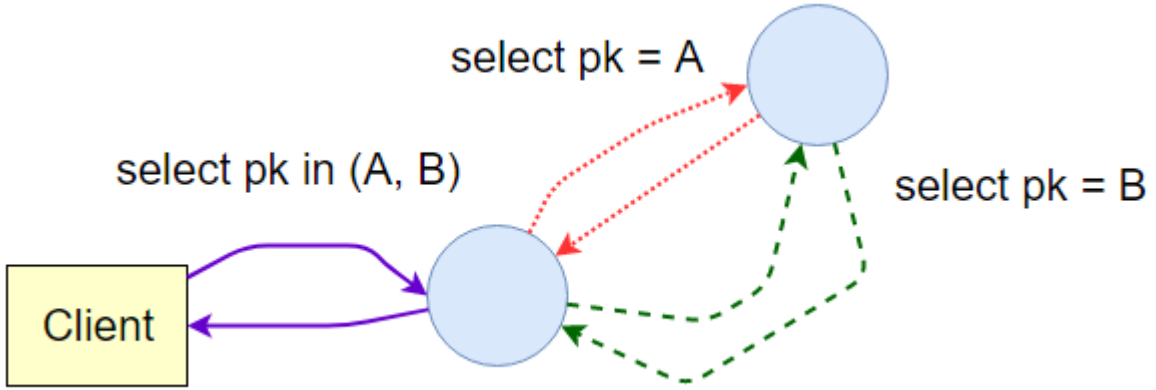
Вариант 4 - SELECT IN with LIMIT

- Делаем предварительные замеры – получаем скорость, аналогичную select sync
- Смотрим по коду Cassandra , что происходит внутри:



Вариант 4 - SELECT IN with LIMIT

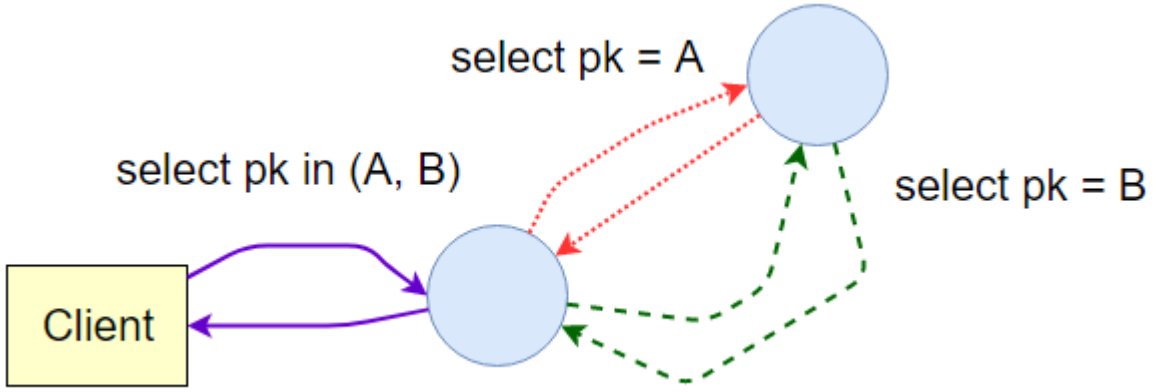
- Делаем предварительные замеры – получаем скорость, аналогичную select sync
- Смотрим по коду Cassandra , что происходит внутри:



- Вопрос: а подзапросы в реплики для SELECT IN запросов - параллельные или последовательные?

Вариант 4 - SELECT IN with LIMIT

- Делаем предварительные замеры – получаем скорость, аналогичную select sync
- Смотрим по коду Cassandra , что происходит внутри:



- Вопрос: а подзапросы в реплики для SELECT IN запросов - параллельные или последовательные?
- Ответ: для одностраничных запросов – да, для многостраничных – нет
  - CASSANDRA-7805
- Ставим явно LIMIT <= page size (50000), чтобы получить одностраничный запрос

Вариант 5 - Async SELECT IN with limit

- вариант 4 (SELECT IN with LIMIT) + вариант 2 (async SELECT)

Надо прочитать набор партиций, кто быстрее?

- Тест:
  - Пачка - 100 партиций
  - В каждой партиции в таблице 50 строк, итого читаем 5000 строк
  - 5 Cassandra нод, RF = 3
  - Время = время чтения пачки

Надо прочитать набор партиций, кто быстрее?

- Тест:
  - Пачка - 100 партиций
  - В каждой партиции в таблице 50 строк, итого читаем 5000 строк
  - 5 Cassandra нод, RF = 3
  - Время = время чтения пачки

Вариант	Среднее время ответа, мс
sync select	187
async select	72
sync select IN	189
sync select IN + LIMIT	102
async select IN	94
async select IN + LIMIT	64

Чтение

- Справка
  - Чтение – уровень кластера
- Проблемы
  - Количество колонок и стоимость чтения
  - Надо прочитать набор партиций, кто быстрее?
  - Как подвесить Cassandra одним запросом ⚠

Как подвесить Cassandra одним запросом

- Симптомы:
  - OperationTimedOutException у ряда приложений
  - Cassandra процессы живы
  - CPU load - всплеск
  - GC duration - всплеск
  - DroppedMessages метрика - всплеск

Как подвесить Cassandra одним запросом

Одно из приложений выполняет запрос вида:

```
SELECT * FROM test_table
    WHERE partitionColumn = p1
      AND clustColumnA IN (a1, a2, .., aN)
      AND clustColumnB IN (b1, b2, …, bN)
      AND clustColumnC IN (c1, c2, …, cM);
```

Как подвесить Cassandra одним запросом

"Native-Transport-Requests-2" #71 daemon runnable
at java.util.Arrays.copyOf
at java.util.ArrayList.grow
at java.util.ArrayList.ensureExplicitCapacity
at java.util.ArrayList.ensureCapacityInternal
at java.util.ArrayList.add
at org.apache.cassandra.db.MultiCBuilder$MultiClusteringBuilder.addEachElementToAll
at org.apache.cassandra.cql3.restrictions.SingleColumnRestriction$INRestriction.appendTo
at org.apache.cassandra.cql3.restrictions.ClusteringColumnRestrictions.valuesAsClustering
at org.apache.cassandra.cql3.restrictions.StatementRestrictions.getClusteringColumns
at org.apache.cassandra.cql3.statements.SelectStatement.getRequestedRows
at org.apache.cassandra.cql3.statements.SelectStatement.makeClusteringIndexFilter
at org.apache.cassandra.cql3.statements.SelectStatement.getSliceCommands
at org.apache.cassandra.cql3.statements.SelectStatement.getQuery
at org.apache.cassandra.cql3.statements.SelectStatement.execute

Как подвесить Cassandra одним запросом

```
SELECT * FROM test_table
        WHERE partitionColumn = p1
          AND clustColumnA IN (a1, a2, .., aN)
          AND clustColumnB IN (b1, b2, …, bN)
          AND clustColumnC IN (c1, c2, …, cN);
```

== Кассандра приводит к форме, аналогичной следующей == >

Как подвесить Cassandra одним запросом

```
SELECT * FROM test_table
        WHERE partitionColumn = p1
          AND clustColumnA IN (a1, a2, .., aN)
          AND clustColumnB IN (b1, b2, …, bN)
          AND clustColumnC IN (c1, c2, …, cN);
```

== Кассандра приводит к форме, аналогичной следующей == >

```
SELECT * FROM test_table
WHERE partitionColumn = p1
  AND (clustColumnA, clustColumnB, clustColumnC)
      IN ((a1,b1,c1), (a1,b1,c2), …, (aN, bN, cN));
```

// N^3 комбинаций

Как подвесить Cassandra одним запросом

- Еще одно напоминание, о том, что в Cassandra:
  - Нет изоляции по ресурсам между различными keyspace – пулы потоков общие

Как подвесить Cassandra одним запросом

- Еще одно напоминание, о том, что в Cassandra:
  - Нет изоляции по ресурсам между различными keyspace – пулы потоков общие
  - Нет жёсткого ограничения на ресурсы для выполняемых запросов

Как подвесить Cassandra одним запросом

- Еще одно напоминание, о том, что в Cassandra:
  - Нет изоляции по ресурсам между различными keyspace – пулы потоков общие
  - Нет жёсткого ограничения на ресурсы для выполняемых запросов
  - Нет жесткого ограничения на время выполнения запроса (по наследству от Java)

Как подвесить Cassandra одним запросом

- Еще одно напоминание, о том, что в Cassandra:
  - Нет изоляции по ресурсам между различными keyspace – пулы потоков общие
  - Нет жёсткого ограничения на ресурсы для выполняемых запросов
  - Нет жесткого ограничения на время выполнения запроса (по наследству от Java)

- DSE 6.x – thread per core архитектура с шардированием запросов между потоками еще более чувствительна к таким непредвиденным задержках

Чтение

- Справка
  - Чтение – уровень кластера
- Проблемы
  - Количество колонок и стоимость чтения
  - Надо прочитать набор партиций, кто быстрее?
  - Как подвесить Cassandra одним запросом

Эпилог

Итого

Итого

- Свежие версии базы и драйвера быстрее
- В Cassandra нет жесткой изоляции запросов по ресурсам, каждый запрос может повлиять на общее состояние системы

Итого

- Свежие версии базы и драйвера быстрее
- В Cassandra нет жесткой изоляции запросов по ресурсам, каждый запрос может повлиять на общее состояние системы
- Драйвер
  - UDT/Collections - есть накладные расходы на Codec
  - Есть ряд ручек покрутить, можно сэкономить немного CPU

Итого

- Свежие версии базы и драйвера быстрее
- В Cassandra нет жесткой изоляции запросов по ресурсам, каждый запрос может повлиять на общее состояние системы
- Драйвер
  - UDT/Collections - есть накладные расходы на Codec
  - Есть ряд ручек покрутить, можно сэкономить немного CPU
- Запись
  - Range tombstones могут вызвать WriteTimeoutException
  - TTL per row + unfrozen collections -> много tombstones

Итого

- Свежие версии базы и драйвера быстрее
- В Cassandra нет жесткой изоляции запросов по ресурсам, каждый запрос может повлиять на общее состояние системы
- Драйвер
  - UDT/Collections - есть накладные расходы на Codec
  - Есть ряд ручек покрутить, можно сэкономить немного CPU
- Запись
  - Range tombstones могут вызвать WriteTimeoutException
  - TTL per row + unfrozen collections -> много tombstones
- Чтение
  - Количество колонок со значениями имеет значение
  - Если нужно прочитать набор partitions, async select – самая простая и почти самая быстрая опция

Q&A

Thank You

Приложение

Дополнительные слайды

Какого рода системы и нагрузку обсуждаем

- Linux сервера
- Cassandra
  - Версии: 3.0.x, 3.11.x, DSE 6.x
  - Java 8
  - OSS DataStax Java driver 3.x
  - 2-3 датацентра в кластере
  - 5-7 инстансов в каждом датацентре
  - В каждом датацентре 3 или 5 реплик данных
  - Consistency level, обычно - LOCAL_QUORUM

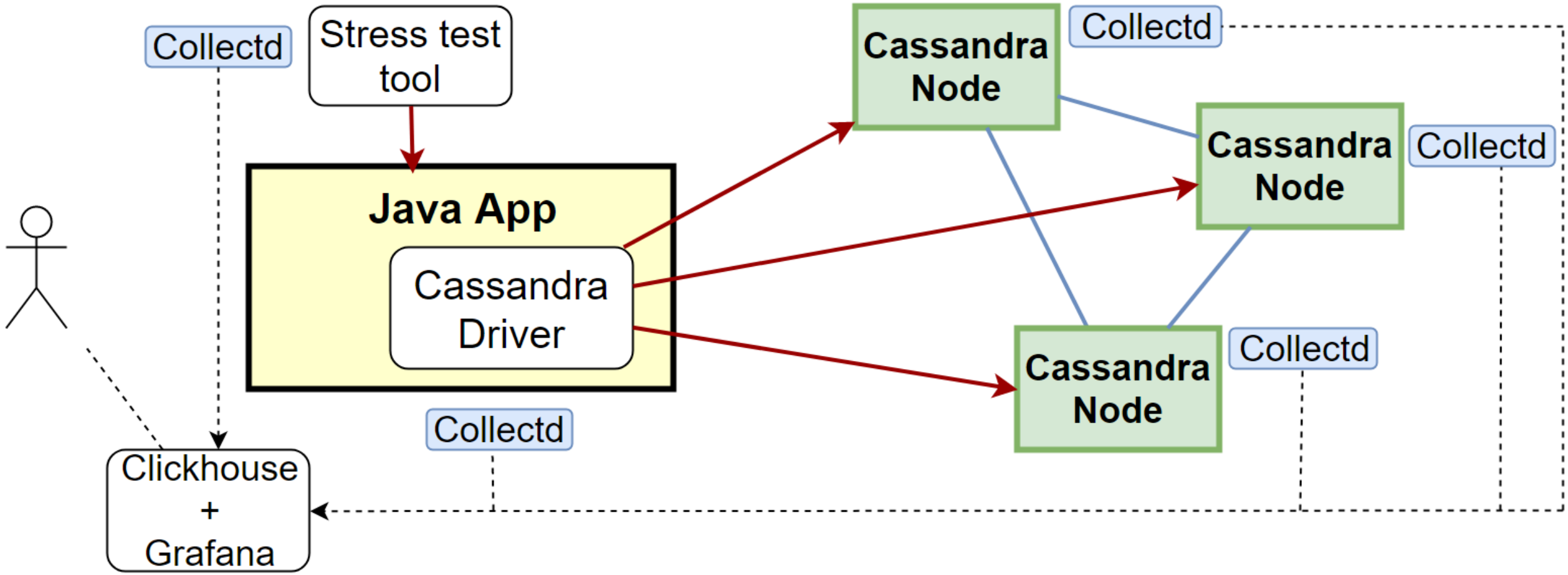
Какого рода системы и нагрузку обсуждаем

- Тип системы: OLTP
  - Нагрузки порядка 10k запросов в секунду уровня приложения
  - Время ответа: 95% персентиль <= 50 ms
  - ~ 5-10 запросов в базу на каждый запрос уровня приложения
  - ~100k запросов в базу в целом
  - Тип нагрузки: 50% чтений / 50% записей
  - Объемы данных ~ 100 Gb
  - Цель: приемлемое время ответа для заданного уровня нагрузки

Какого рода системы и нагрузку обсуждаем

- Тип системы: пакетная обработка
  - Тип нагрузки: 40% чтений / 60% записей
  - Объемы данных ~ 1-10 TBs
  - Цель: пропускная способность

Итого - исследуемая система, со сбором метрик



Метрики базы, избранное

- Встроенные метрики уровня базы - **RED** (Rate, Errors, Duration)
  - Ошибки
    - org.apache.cassandra.metrics:type=ClientRequest,scope=*,name=Timeouts
    - org.apache.cassandra.metrics:type=ClientRequest,scope=*,name=Failures
    - org.apache.cassandra.metrics:type=ClientRequest,scope=*,name=Unavailables
    -
    - org.apache.cassandra.metrics:type=DroppedMessage,scope=*,name=Dropped
    - org.apache.cassandra.metrics:type=Storage,name=Exceptions

Метрики базы, избранное

- Встроенные метрики уровня базы - **RED** (Rate, Errors, Duration)
  - Rate – производная от Count + Latency (среднее, персентили)
    - org.apache.cassandra.metrics:type=ClientRequest,scope=*,name=Latency

    - org.apache.cassandra.metrics:keyspace=*,name=WriteLatency,scope=*,type=Table
    - org.apache.cassandra.metrics:keyspace=*,name=ReadLatency,scope=*,type=Table
    - org.apache.cassandra.metrics:keyspace=*,name=CoordinatorReadLatency,scope=*,type=Table

    - CoordinatorWriteLatency – появится только в Cassandra 4.0 (CASSANDRA-14232)

Метрики базы, избранное

- Пулы потоков
  - ...:type=ThreadPools,name=ActiveTasks,path=internal,scope=*
  - ...:type=ThreadPools,name=PendingTasks,path=internal,scope=*
  - ...:type=ThreadPools,name=TotalBlockedTasks,path=internal,scope=*

- Количество SSTables, читаемых во время read операции
  - …:keyspace=*,name=SSTablesPerReadHistogram,scope=*,type=Table
- Tombstones
  - …:keyspace=*,name=TombstoneScannedHistogram,scope=*,type=Table

Трассировка запросов

- Трассировка % запросов
- nodetool settraceprobability 0.0001
- Больше деталей: https://www.datastax.com/dev/blog/tracing-in-cassandra-1-2

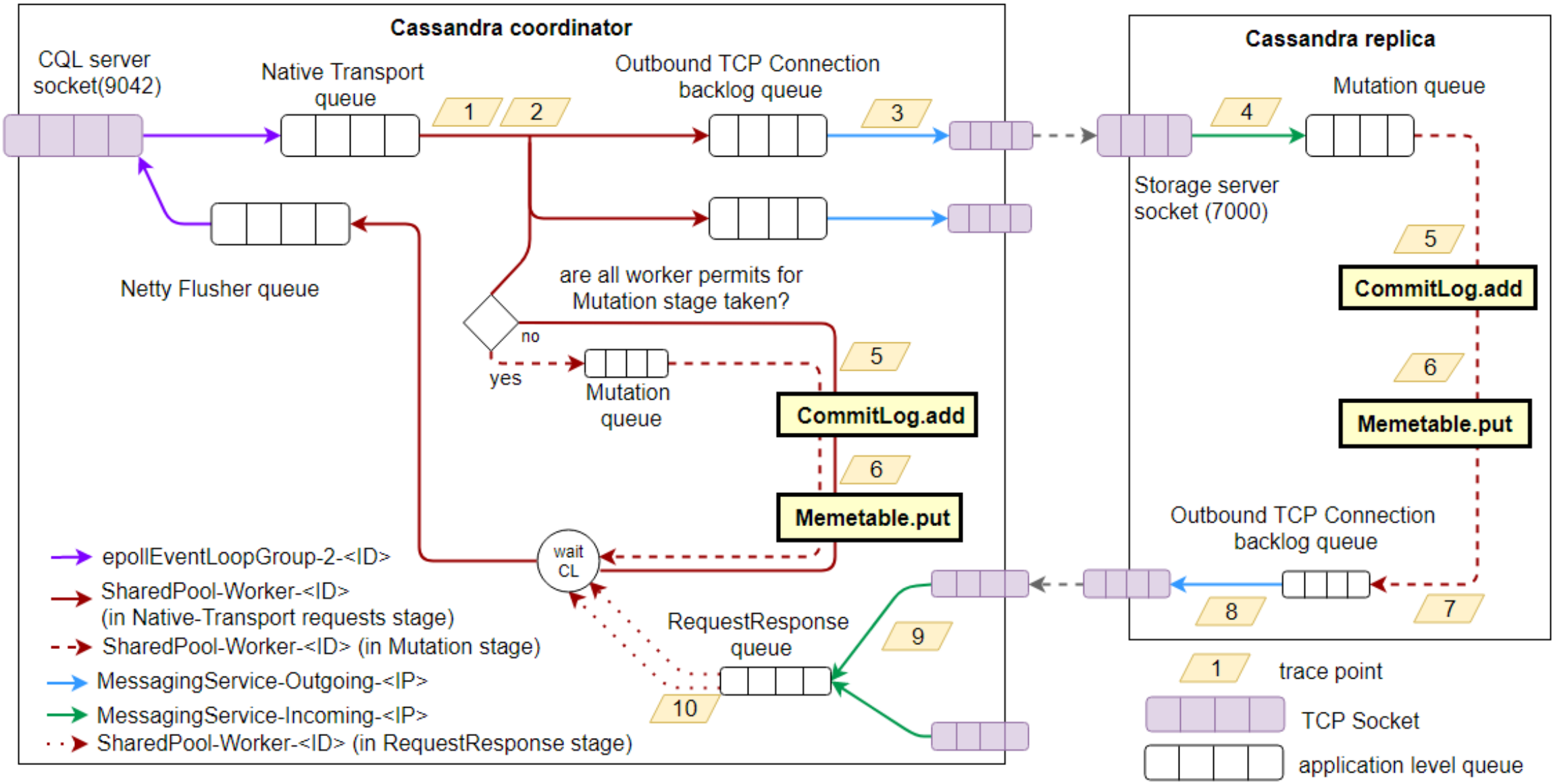
Трассировка запросов - пример

- SELECT запрос, consistency_level = LOCAL_ONE

```
Tracing session: 6930c380-ddea-11e9-97f8-952ee9e78380

                 activity                                       | timestamp       |source | elapsed |
-----------------------------------------------------------------+-----------------+-------+---------+
                                    Execute CQL3 query | 06:10:46.840000 | ...49 |       0 |
                  Parsing query <skipped> ; [CoreThread-1] | 06:10:46.840001 | ...49 |     121 |
                       Preparing statement [CoreThread-1] | 06:10:46.840001 | ...49 |     231 |
             Reading data from [/10.101.18.49] [CoreThread-1] | 06:10:46.840001 | ...49 |     362 |
Executing single-partition query on test_table [CoreThread-4] | 06:10:46.840002 | ...49 |     691 |
                   Acquiring sstable references [CoreThread-4] | 06:10:46.840003 | ...49 |     725 |
     Merged data from memtables and 0 sstables [CoreThread-4] | 06:10:46.841000 | ...49 |     763 |
        Read 1 live rows and 0 tombstone cells [CoreThread-4] | 06:10:46.841000 | ...49 |     782 |
                                  Request complete | 06:10:46.840918 | ...49 |     918 |
```

Запись – потоки и очереди



Трассировка записи

ID	Text
1	Execute CQL3 prepared query <session started - initial timestamp is stored>
2	Determining replicas for mutation
3	Sending MUTATION message to %s
4	MUTATION message received from {}
5	Appending to commitlog
6	Adding to {} memtable
7	Enqueuing response to {}
8	Sending REQUEST_RESPONSE message to %s
9	REQUEST_RESPONSE message received from {}
10	Processing response from {}

Запись

- Справка
  - Запись – что происходит внутри реплики
- Проблемы
  - Проблема при вставке элементов списка
  - Скрытая стоимость TTL
  - WriteTimeoutException при удалении
  - Выделение памяти при записи

Выделение памяти

- Mutation объекты сериализуются дважды 1) для того, чтобы определить размер буфера 2) для того, чтобы записать данные
  - Актуально для 3.0.x версий
  - Исправлено, начиная с 3.10
  - https://issues.apache.org/jira/browse/CASSANDRA-12269

Выделение памяти

- Mutation объекты сериализуются дважды 1) для того, чтобы определить размер буфера 2) для того, чтобы записать данные
  - Актуально для 3.0.x версий
  - Исправлено, начиная с 3.10
  - https://issues.apache.org/jira/browse/CASSANDRA-12269

- Излишние выделения памяти при записи, например – при вычислении hashCode в рамках Memtable.put
  - Актуально для 3.0.x версий
  - Исправлено, начиная с 3.6 https://issues.apache.org/jira/browse/CASSANDRA-11428
  - Дополнительно 3.10 - https://issues.apache.org/jira/browse/CASSANDRA-12269

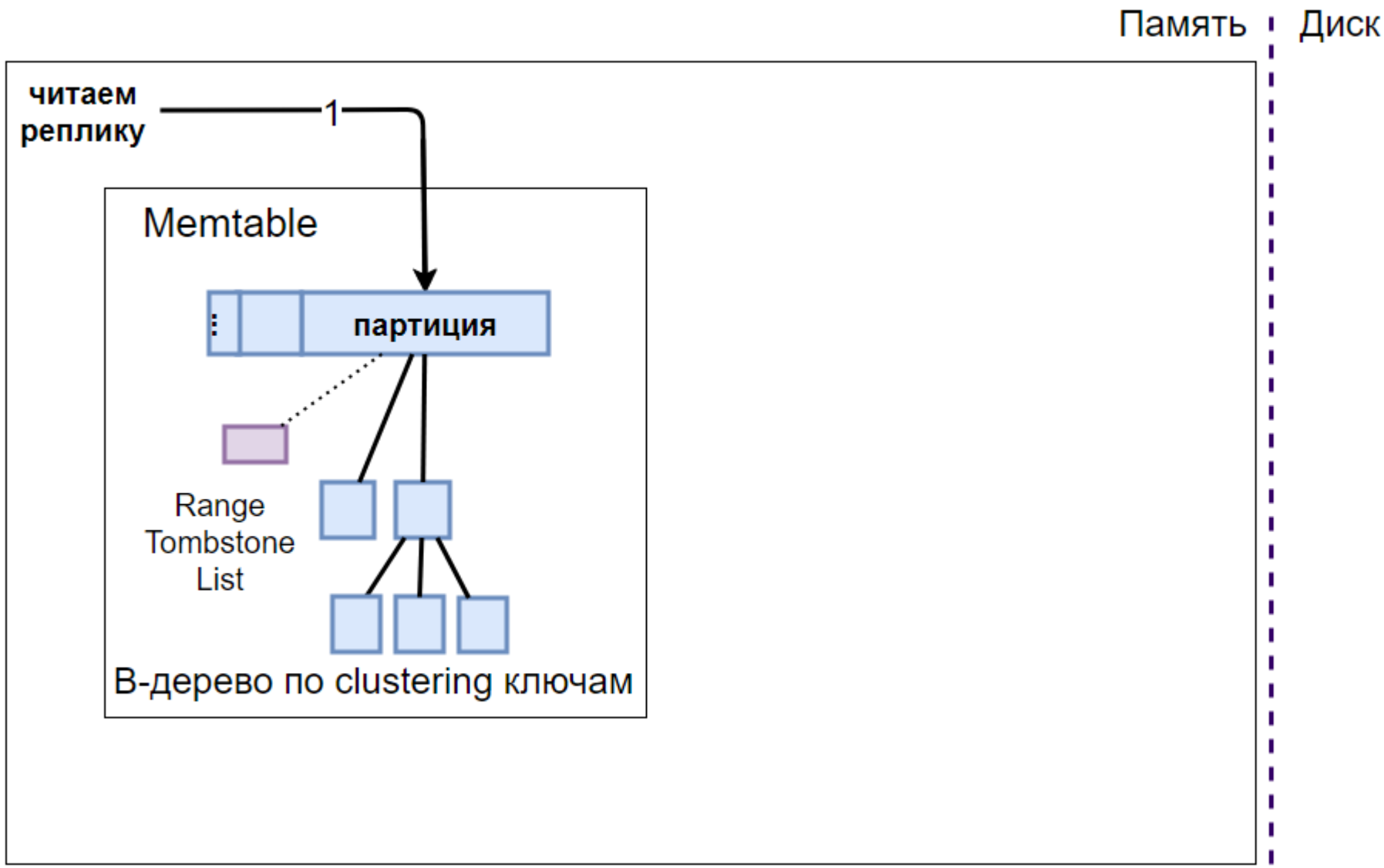
Запись

- Справка
  - Запись – что происходит внутри реплики
- Проблемы
  - Проблема при вставке элементов списка
  - Скрытая стоимость TTL
  - WriteTimeoutException при удалении
  - Выделение памяти при записи

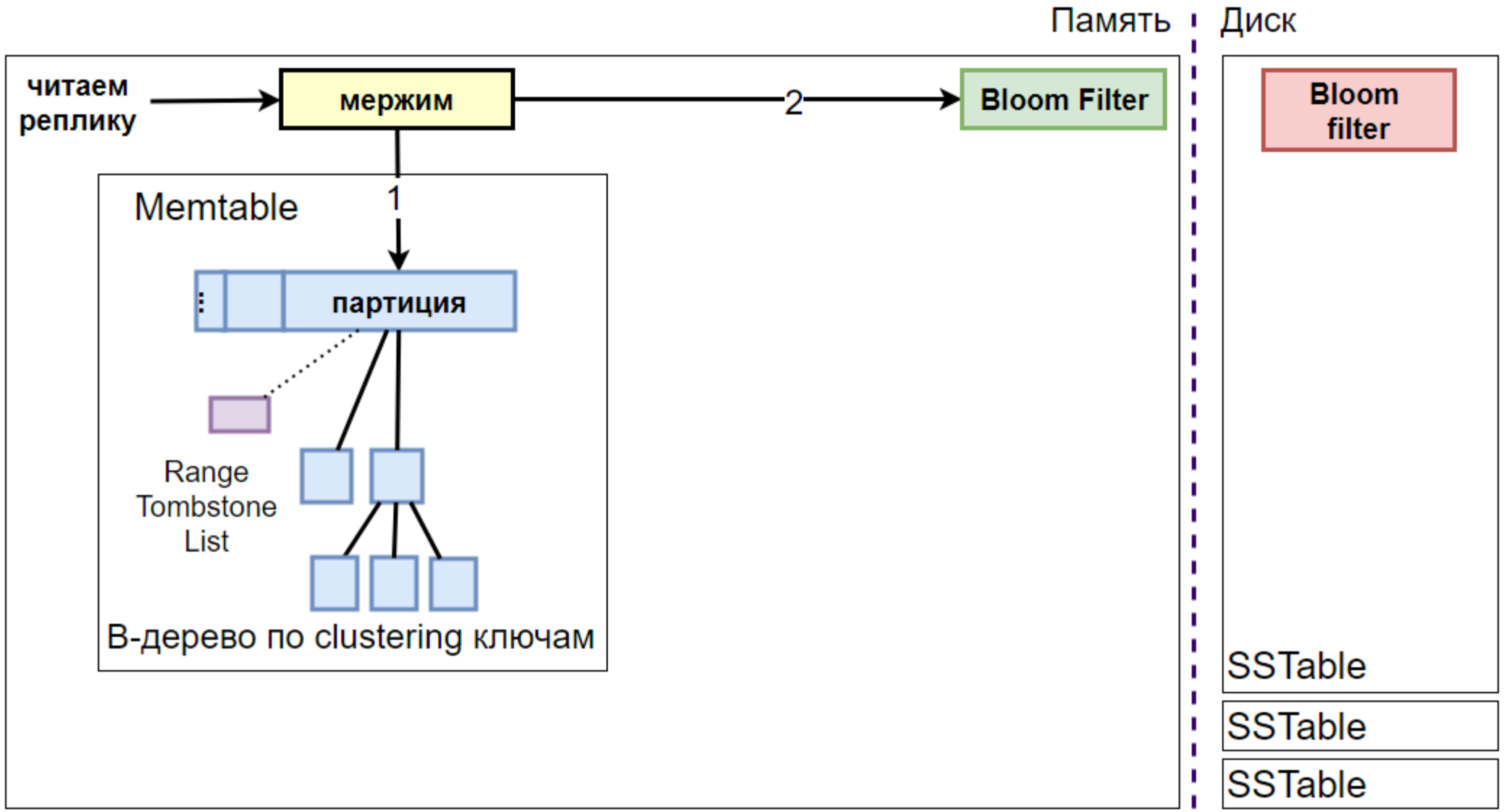
Чтение

- Справка ?
  - Чтение – уровень кластера
  - Чтение – что происходит внутри реплики

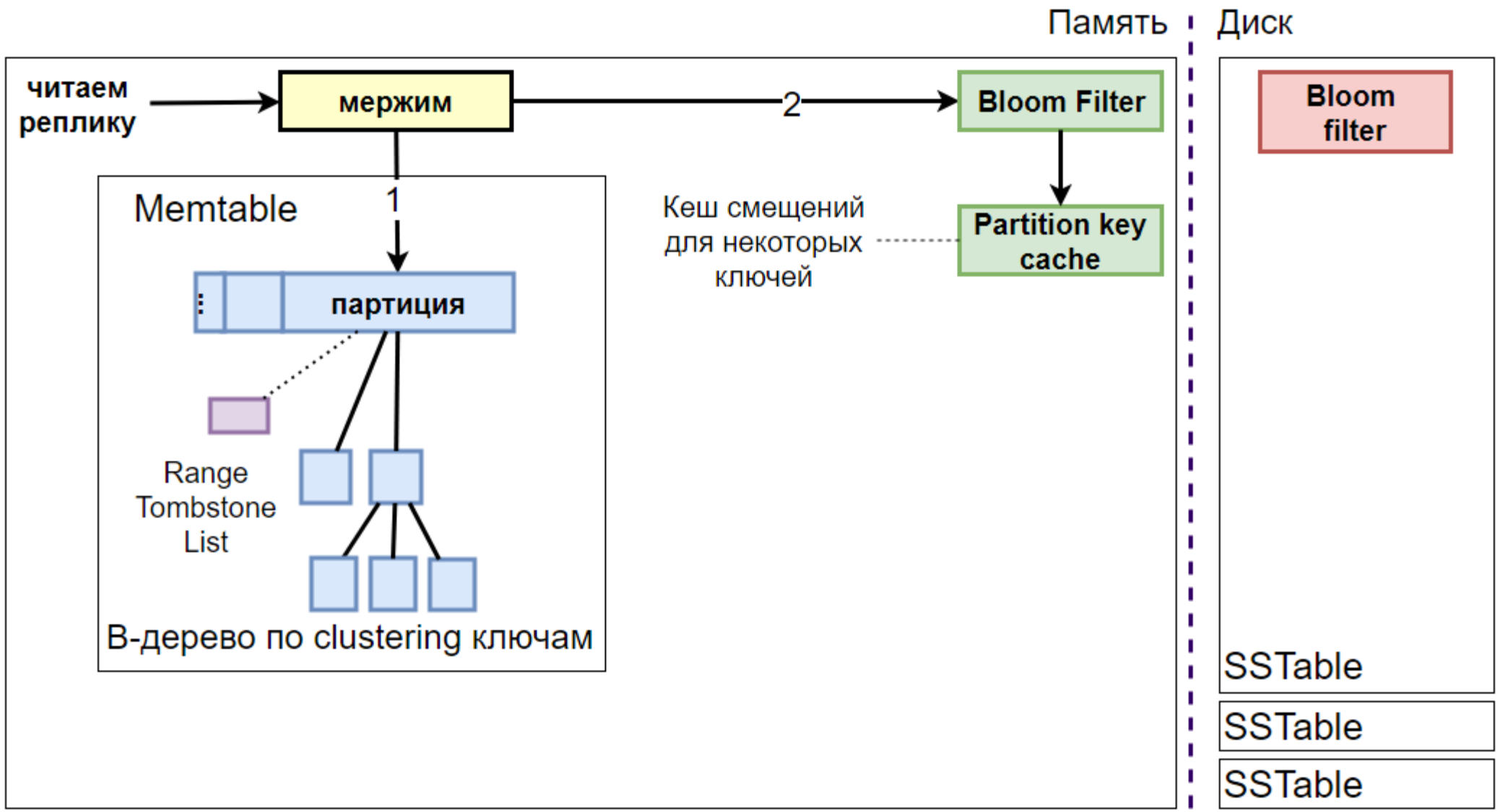
Чтение – уровень реплики



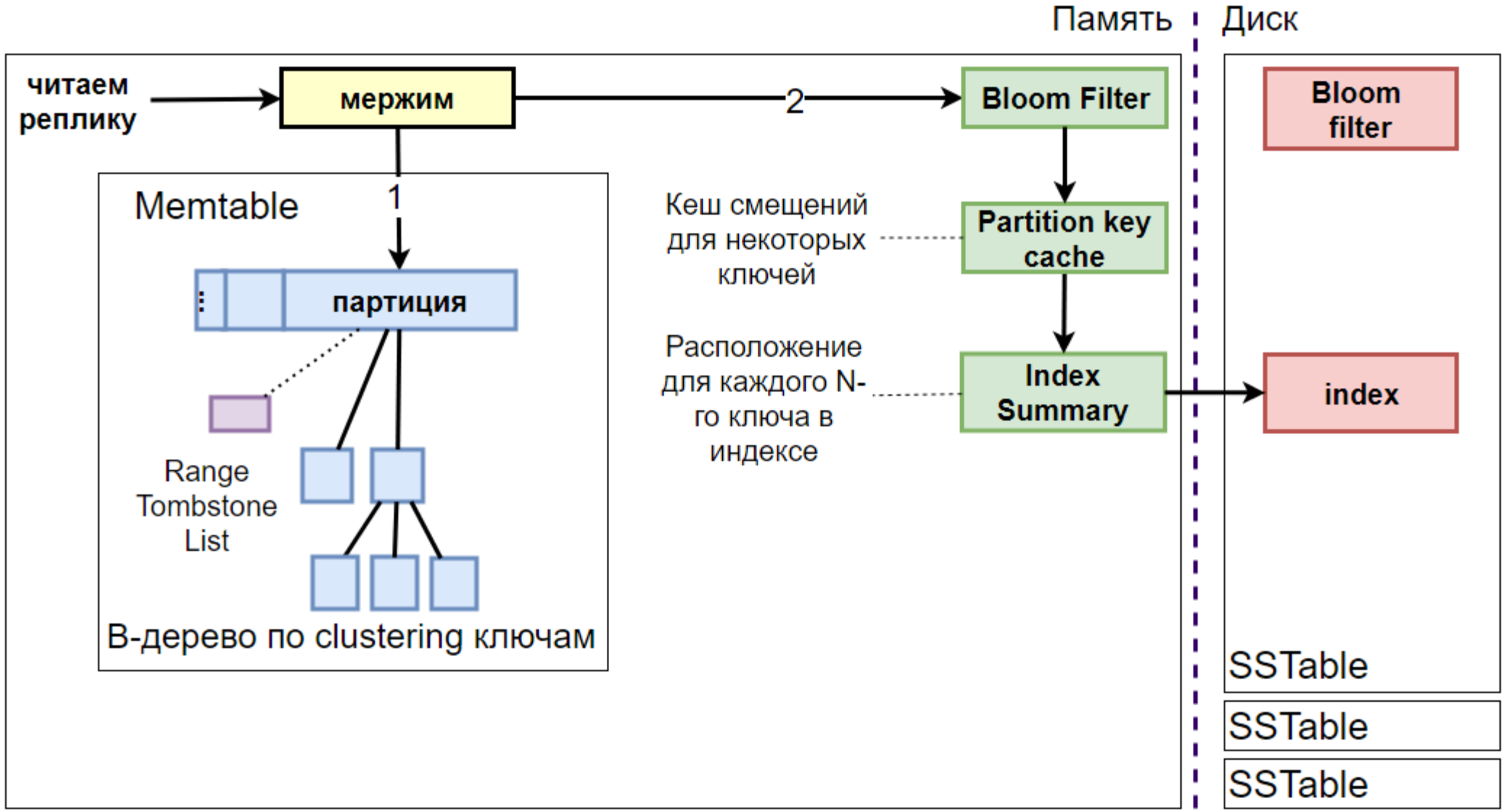
Чтение – уровень реплики



Чтение – уровень реплики



Чтение – уровень реплики



Чтение – уровень реплики