Batch select, корутины и 1 млн RPS

Александр Маторин

Об авторе

Александр Маторин

- Работаю в департаменте Рисков в Сбере
- Разрабатываю на Java 15 лет
- Преподаю курсы по Java и распределенным системам в магистратуре СберТеха в МФТИ

Почему Batch select?

Антифрод система банка

- Online проверка кредитных заявок клиентов.
- Offline прескоринг клиентов для подбора предложений.

Для каждого клиента выполняется несколько сотен антифрод правил.

Legacy антифрод система банка

Несколько ТБ данных в памяти.

Собственный язык программирования для написания антифрод правил бизнес-пользователями.

Kotlin DSL для запросов

СБЕР

```
val clients = findAll(client, limit = 5) fetchFields { listOf(lastName, firstName, address.house) } where {
    passport `=` clientPassport
    lastName `=` clientLastName
    firstName `=` clientFirstName
}
```

Kotlin DSL для запросов

СБЕР

```
val clients = findAll(client, limit = 5) fetchFields { listOf(lastName, firstName, address.house) } where {
    passport `=` clientPassport
    lastName `=` clientLastName
    firstName `=` clientFirstName
    address {
        cityName `=` "Moscow"
        house `=` 12
    }
    phones {
        number `=` clientNumber
        type `in` listOf("work", "home")
    }
}
```

Пакетный прескоринг клиентов

Для каждого клиента выполнить несколько сотен антифрод правил.

Каждое правило запрашивает данные из бд и выполняет бизнес логику.

5_000_000клиентов * 300правил *2селекта = 3млрд селектов.

Пакетный прескоринг клиентов

Для каждого клиента выполнить несколько сотен антифрод правил.

Каждое правило запрашивает данные из бд и выполняет бизнес логику.

5_000_000клиентов * 300правил *2селекта = 3млрд селектов.

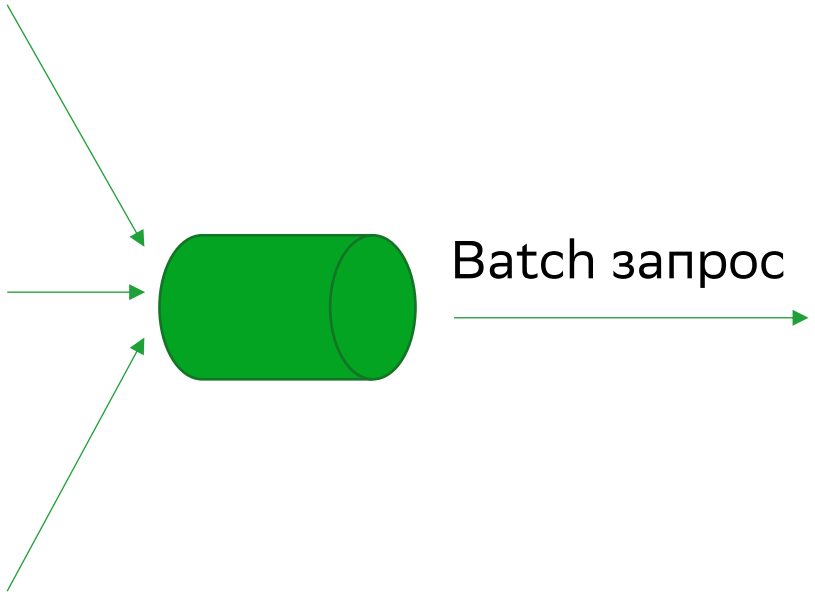
Запросы к данным через DSL писать удобно, но работает это не очень быстро.

Как ускорить скоринг

- Шардинг данных
- Кеши
- NoSQL
- Ускорить сами запросы

Kotlin DSL для BATCH запросов

```
findAll(client) fetchFields { listOf(lastName, firstName) } where {
    passport `=` clientPassport1
}

findAll(client) fetchFields { listOf(lastName, firstName) } where {
    passport `=` clientPassport2
}

findAll(client) fetchFields { listOf(lastName, firstName) } where {
    passport `=` clientPassport3
}
```



Стандартный способ делать Batch обработку

СБЕР

```
public int process(Client client) {
    ...
    selectBalanceOf(client);
    ...
}
```

```
public List<Integer> process(List<Client> clients) {
    ...
    selectBalancesOf(clients);
    ...
}
```

Стандартный способ делать Batch обработку

СБЕР

```
public int process(Client client) {
    ...
    selectBalanceOf(client);
    ...
}
```

```
public List<Integer> process(List<Client> clients) {
    ...
    selectBalancesOf(clients);
    ...
}
```

не хочется менять сигнатуру

Стандартный способ делать Batch обработку

СБЕР

```
public int process(Client client) {
    ...
    selectBalanceOf(client);
    ...
}
```

→ Batch select

```
public List<Integer> process(List<Client> clients) {
    ...
    selectBalancesOf(clients);
    ...
}
```

не хочется менять сигнатуру

Обсудим

- Можно ли переписать любой селект в Batch вид
- Производительность Batch select
- Как оптимально собирать запросы для создания Batch select’а

JDBC селект

```
long passport = getRandomPassport();

try (var c = dataSource.getConnection()) {
    String sql = "select last_name from client where passport = ?";

    try (var ps = c.prepareStatement(sql)) {
        ps.setLong(1, passport);

        try (ResultSet rs = ps.executeQuery()) {
            return parse(rs);
        }
    }
}
```

Таблица для тестирования

PostgreSQL 13, 16cpu/128gb

```
create table client
(
    id              integer primary key,
    name            varchar,
    passport_str    varchar, // +индекс
    passport        bigint   // +индекс
);
```

Добавим 200млн записей (13gb данные + (4 + 6 + 6)gb индексы)

```
insert into client(id, name, passport_str, passport_number) select i,
       (FLOOR(RANDOM() * 9_000_000_000) + 1_000_000_000)::text,
       (FLOOR(RANDOM() * 9_000_000_000) + 1_000_000_000)::text,
       FLOOR(RANDOM() * 9_000_000_000) + 1_000_000_000
from generate_series(0, 200 * 1000 * 1000) as t(i);
```

JDBC селект

```
long passport = getRandomPassport();

try (var c = dataSource.getConnection()) {
   String sql = "select last_name from client where passport = ?";

   try (var ps = c.prepareStatement(sql)) {
      ps.setLong(1, passport);

      try (ResultSet rs = ps.executeQuery()) {
         return parse(rs);
      }
   }
}
```

1 connection: 950 RPS
200 connections: 160тыс RPS

JDBC Batch селект

```
List<Long> passports = get200RandomPassports();

try (var c = dataSource.getConnection()) {
    String sql = "select last_name from client where passport = ?;".repeat(passports.size());

    try (var ps = c.prepareStatement(sql)) {
        for (int i = 0; i < passports.size(); i++) {
            ps.setLong(i + 1, passports.get(i));
        }

        try (ResultSet rs = ps.executeQuery()) {
            return parse(rs);
        }
    }
}
```

JDBC Batch селект

```
List<Long> passports = get200RandomPassports();

try (var c = dataSource.getConnection()) {
    String sql = "select last_name from client where passport = ?;".repeat(passports.size());

    try (var ps = c.prepareStatement(sql)) {
        for (int i = 0; i < passports.size(); i++) {
            ps.setLong(i + 1, passports.get(i));
        }

        try (ResultSet rs = ps.executeQuery()) {    //org.postgresql.util.PSQLException: Multiple
            return parse(rs);                       ResultSets were returned by the query.
        }
    }
}
```

JDBC Batch селект

```
List<Long> passports = get200RandomPassports();

try (var c = dataSource.getConnection()) {
   String sql = "select last_name from client where passport = ?;".repeat(passports.size());

   try (var ps = c.prepareStatement(sql)) {
      for (int i = 0; i < passports.size(); i++) {
         ps.setLong(i + 1, passports.get(i));
      }

      ps.execute();
      var results = new ArrayList<String>();
      do {
           try (ResultSet rs = ps.getResultSet()) { //1
              results.addAll(parse(rs));
           }
       } while (ps.getMoreResults()); { //2
      return results;
   }
}
```

JDBC Batch селект

```
List<Long> passports = get200RandomPassports();

try (var c = dataSource.getConnection()) {
   String sql = "select last_name from client where passport = ?;".repeat(passports.size());

   try (var ps = c.prepareStatement(sql)) {
      for (int i = 0; i < passports.size(); i++) {
         ps.setLong(i + 1, passports.get(i));
      }

      ps.execute();
      var results = new ArrayList<String>();
      do {
          try (ResultSet rs = ps.getResultSet()) { //1
             results.addAll(parse(rs));
          }
       } while (ps.getMoreResults()); { //2
      return results;
   }
}
```

1 connection: 45тыс RPS, x50!
25 connections: 750тыс RPS, x5

JDBC Batch селект

```
List<Long> passports = get200RandomPassports();

try (var c = dataSource.getConnection()) {
   String sql = "select last_name from client where passport = ?;".repeat(passports.size());

   try (var ps = c.prepareStatement(sql)) {
      for (int i = 0; i < passports.size(); i++) {
         ps.setLong(i + 1, passports.get(i));
      }

      ps.execute();
      var results = new ArrayList<String>();
      do {
           try (ResultSet rs = ps.getResultSet()) { //1
              results.addAll(parse(rs));
           }
       } while (ps.getMoreResults()); { //2
      return results;
   }
}
```

1 connection: 45тыс RPS, x50!
25 connections: 750тыс RPS, x5

А можно еще быстрее?

Жизненный цикл селекта PgJDBC

СБЕР

```
PreparedStatement = c.prepareStatement("select last_name from client where passport = ?");
```



```
public PreparedStatement prepareStatement(String sql)  {
    return new PgPreparedStatement(connection, connection.borrowQuery(sql));
}
```

Жизненный цикл селекта PgJDBC


```
PreparedStatement = c.prepareStatement("select last_name from client where passport = ?");
```



```
public PreparedStatement prepareStatement(String sql)  {
    return new PgPreparedStatement(connection, statementCache.borrow(sql));
}
```



```
private final LruCache<String, CachedQuery> statementCache;
```

Жизненный цикл селекта PgJDBC

```
public class CachedQuery {
  public final String key;
  public final Query query;
  private int executeCount;
}
```

Если executeCount < 5 в бд отправляются sql + параметры.
План запроса строится при каждом выполнении.

Жизненный цикл селекта PgJDBC

```
public class CachedQuery {
  public final String key;
  public final Query query;
  private int executeCount;
}
```

Если executeCount < 5 в бд отправляются sql + параметры.
План запроса строится при каждом выполнении.

Если executeCount >= 5 на клиенте генерируется Query.statementName(S_1).
На сервере сохраняется named ServerPreparedStatement с generic планом запроса.
В бд отправляются только statementName + параметры запроса.

Жизненный цикл селекта PgJDBC

```
public class CachedQuery {
  public final String key;
  public final Query query;
  private int executeCount;
}
```

Если executeCount < 5 в бд отправляются sql + параметры.
План запроса строится при каждом выполнении.

Если executeCount >= 5 на клиенте генерируется Query.statementName(S_1).
На сервере сохраняется named ServerPreparedStatement с generic планом запроса.
В бд отправляются только statementName + параметры запроса.

Имеет смысл в jdbcUrl добавить prepareThreshold=1, чтобы строить план запроса только один раз.
https://github.com/pgjdbc/pgjdbc/issues/2886

LruCache<String, CachedQuery> statementCache

СБЕР

```
private final int maxSizeEntries; //256
private final long maxSizeBytes; //5MB

private final Map<K, V> cache = new LinkedHashMap<>() {
    @Override
    protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
        return size() > maxSizeEntries || currentSizeBytes > maxSizeBytes;
    }
}
```

LruCache<String, CachedQuery> statementCache

СБЕР

```
private final int maxSizeEntries; //256
private final long maxSizeBytes; //5MB

private final Map<K, V> cache = new LinkedHashMap<>() {
    @Override
    protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
        return size() > maxSizeEntries || currentSizeBytes > maxSizeBytes;
    }
}

public class CachedQuery implements CanEstimateSize {
    public final String key;

    @Override
    public long getSize()
        return key.length() * 2L * 2 + 100L; {// 2 bytes per char, revise with Java 9 compact strings
     }
}
```

Batch SQL как ключ кеширования

СБЕР

```
String sql = "select last_name from client where passport = ?;".repeat(100);
var ps = c.prepareStatement(sql);
```

Batch SQL как ключ кеширования

СБЕР

```
String sql = "select last_name from client where passport = ?;".repeat(100);
var ps = c.prepareStatement(sql);

public class CachedQuery {
   public final String key = "select last_name from client where passport = ?;select...".;//x100
   public final Query query = new CompositeQuery(
      new SimpleQuery("select last_name from client where passport = ?", "S_1"),
      new SimpleQuery("select last_name from client where passport = ?", "S_2"),
      new SimpleQuery("select last_name from client where passport = ?", "S_3"),
      new SimpleQuery("select last_name from client where passport = ?", "S_4"),
      new SimpleQuery("select last_name from client where passport = ?", "S_5")
      ...
   );
   private int executeCount;
}
```

Batch SQL как ключ кеширования

При разном количестве SQL в Batch sql будет происходить вытеснение CachedQuery и перестройка планов запросов.

Построение плана может занимать в несколько раз больше времени, чем выполнение запроса.

План запроса строится для запроса в рамках connection после 5 выполнений.

Попробуем переписать BATCH SQL

Чтобы для разного количества одинаковых запросов результирующий SQL был один и тот же.

И план запроса строился один раз и не вытеснялся.

Производительность селекта по списку параметров


```
explain analyze
select c1 from table1 where c2 = ?
Planning Time: 0.98 ms
Execution Time: 0.053 ms
```

Производительность селекта по списку параметров


```
explain analyze
select c1 from table1 where c2 = ?
Planning Time: 0.98 ms
Execution Time: 0.053 ms
```

```
explain analyze
select c1 from table1 where c2 in (?, ?, ?...) //100 параметров
```

Производительность селекта по списку параметров

СБЕР

```
explain analyze
select c1 from table1 where c2 = ?
Planning Time: 0.98 ms
Execution Time: 0.053 ms
```

```
explain analyze
select c1 from table1 where c2 in (?, ?, ?...) //100 параметров
Planning Time: 0.152 ms
Execution Time: 0.862 ms
```

Прирост пропускной способности в 6 раза!

Пробуем переписать любой select в batch


```
select passport from client where last_name = ?
```

Пробуем переписать любой select в batch


```
select passport from client where last_name = ?
```

```
select passport from client where last_name in (?, ?, ?) // а как понять кому принадлежит
                                                            passport из ResultSet?
```

Пробуем переписать любой select в batch

СБЕР

```
select passport from client where last_name = ?
```

```
select passport from client where last_name in (?, ?, ?) // а как понять кому принадлежит
                                                            passport из ResultSet?
```

```
select passport, last_name from client where last_name in (?, ?, ?) // большой ResultSet
```

Пробуем переписать любой select в batch

СБЕР

```
select passport from client where last_name = ?
```

```
select passport from client where last_name in (?, ?, ?) // а как понять кому принадлежит
                                                            passport из ResultSet?
```

```
select passport, last_name from client where last_name in (?, ?, ?) // большой ResultSet
```

```
select passport, last_name, first_name from client where (last_name, first_name) in ((?,?), (?,?))
```

Пробуем переписать любой select в batch

СБЕР

```
select passport from client where last_name = ?

select passport from client where last_name in (?, ?, ?) // а как понять кому принадлежит
                                                            passport из ResultSet?

select passport, last_name from client where last_name in (?, ?, ?) // большой ResultSet

select passport, last_name, first_name from client where (last_name, first_name) in ((?,?), (?,?))

select passport, last_name, first_name from client where last_name = ? or first_name = ?
// отдельные in уже не работают
// как сделать limit или order by для результата каждого входного параметра?
```

Проблемы с оператором in. JPA

СБЕР

```
entityManager.createQuery("select passport from client where lastName in (:lastNames)")
        .setParameter("lastNames", lastNames)
        .getResultList();
```

Проблемы с оператором in. JPA

```
entityManager.createQuery("select passport from client where lastName in (:lastNames)")
        .setParameter("lastNames", lastNames)
        .getResultList();
```



```
select passport from client where last_name in (?, ?, ?, ?, ?, ?, ?, ?, ?...)
```

(:lastNames) под капотом переписывается в ? на каждый элемент списка lastNames. На каждый запрос.

Мусор от строк.
Для разного количества параметров - разные планы запросов.

Оператор in и план запросов. JPA


```
hibernate.query.in_clause_parameter_padding=true
```

Hibernate для уменьшения количества разных планов запросов может использовать количество ?, равное ближайшей степени двойки.

Для 10 и 14 параметров будет 2^4=16 плейсхолдеров.
Для 129 и 250 параметров будет 2^8=256 плейсхолдеров.

Spring JdbcTemplate

```
val names = listOf("Ivanov", "Petrov", "Sidorov")
jdbcTemplate.queryForList(
    "select passport from client where last_name in (:names)", mapOf(”names", names)
)
```

org.springframework.jdbc.core.namedparam.NamedParameterUtils


```
jdbcTemplate.queryForList(
    "select passport from client where last_name in (?, ?, ?)", mapOf(”names", names)
)
```

Spring JdbcTemplate

СБЕР

```
val names = arrayOf("Ivanov", "Petrov", "Sidorov")
jdbcTemplate.queryForList(
    "select passport from client where last_name = any(:names)", mapOf("names", names)
)
```

Один план запроса для разного количества параметров.

Пробуем переписать любой select в batch

```
select passport from client where last_name = ?
```

```
select passport from client where last_name = $1
union all
select passport from client where last_name = $2
union all
select passport from client where last_name = $3
```

Пробуем переписать любой select в batch

СБЕР

```
select passport from client where last_name = ?
```

```
select passport from client where last_name = $1
union all
select passport from client where last_name = $2
union all
select passport from client where last_name = $3
```

//Разные планы запросов для разного количества параметров

unnest

```
select unnest(array ['Ivanov', 'Morozov', 'Fedorov']) as lastName
```

lastName
Ivanov
Morozov
Fedorov

unnest

```
select unnest(array ['Ivanov', 'Morozov', 'Fedorov']) as lastName,
       unnest(array ['Alex', 'Vasia', 'Dima']) as firtName
```

lastName,	firstName
Ivanov,	Alex
Morozov,	Vasia
Fedorov,	Dima

unnest

```
select generate_series(0, 2) as index,
       unnest(array ['Ivanov', 'Morozov', 'Fedorov']) as lastName,
       unnest(array ['Alex', 'Vasia', 'Dima']) as firtName
```

```
index, lastName, firstName
0,     Ivanov,   Alex
1,     Morozov,  Vasia
2,     Fedorov,  Dima
```

Хороший Batch select. PostgreSQL

```
select passport from client where last_name = ?
```

```
select input.index, client.passport
from
(select generate_series(0, ?) as index, unnest(?) as lastName) as input,
client
where client.last_name = input.lastName
```

Хороший Batch select. PostgreSQL

```
select passport from client where last_name = ? or first_name = ?
```

```
select input.index, client.passport
from
(select generate_series(0, ?) as index, unnest(?) as lastName, unnest(?) as firstName) as input,
client
where client.last_name = input.lastName or client.first_name = input.firstName
```

Хороший Batch select. PostgreSQL

```
select passport from client where last_name = ? or first_name = ?
```

```
select input.index, client.passport
from
(select generate_series(0, ?) as index, unnest(?) as lastName, unnest(?) as firstName) as input,
client
where client.last_name = input.lastName or client.first_name = input.firstName
```

Сопоставление входных параметров с результатом по индексу

index,	passport
0,	122342432
0,	432432423
1,	654654654
2,	534543543

Batch select. PostgreSQL, limit + order by внутри группы

СБЕР

```
select passport from client where last_name = ? or first_name = ? order by age limit 10
```

```
select input.index, main.*
from
(select generate_series(0, ?) as index, unnest(?) as lastName, unnest(?) as firstName) as input
join lateral (
    select client.passport
    from client
    where client.last_name = input.lastName or client.first_name = input.firstName
    order by age
    limit 10
) main on true
```

Batch JDBC селект

```
try (var c = dataSource.getConnection()) {
   String sql = "select input.index, main.* from " +
   "(select generate_series(0, ?) as index, unnest(?) as passport) as input join lateral " +
   "(select client.name from client where client.passport = input.passport) main on true";

   try (var ps = c.prepareStatement(sql)) {
      ps.setInt(1, passports.size() - 1);
      ps.setArray(2, c.createArrayOf("bigint", passports.toArray()))

      try (ResultSet rs = ps.executeQuery()) {
         return parse(rs);
      }
   }
}
```

Batch JDBC селект

СБЕР

```
try (var c = dataSource.getConnection()) {
   String sql = "select input.index, main.* from " +
   "(select generate_series(0, ?) as index, unnest(?) as passport) as input join lateral " +
   "(select client.name from client where client.passport = input.passport) main on true";

   try (var ps = c.prepareStatement(sql)) {
      ps.setInt(1, passports.size() - 1);
      ps.setArray(2, c.createArrayOf("bigint", passports.toArray()))

      try (ResultSet rs = ps.executeQuery()) {
         return parse(rs);
      }
   }
}
```

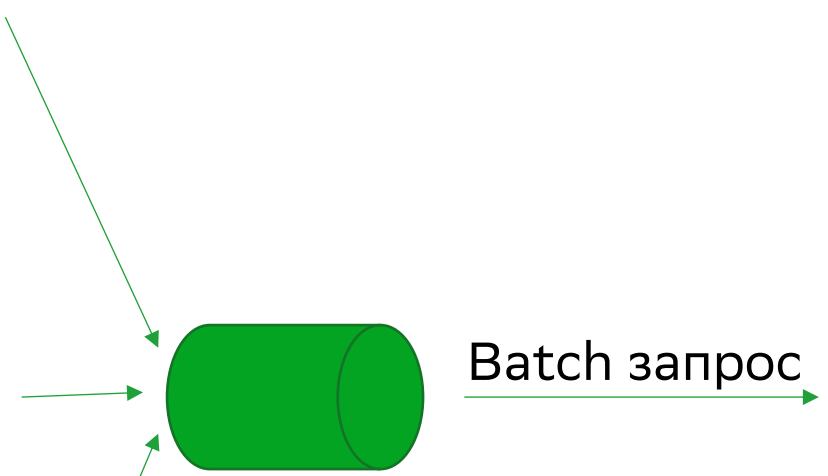
1 connection: 100тыс RPS, x100!

30 connections: 2млн RPS, x12!

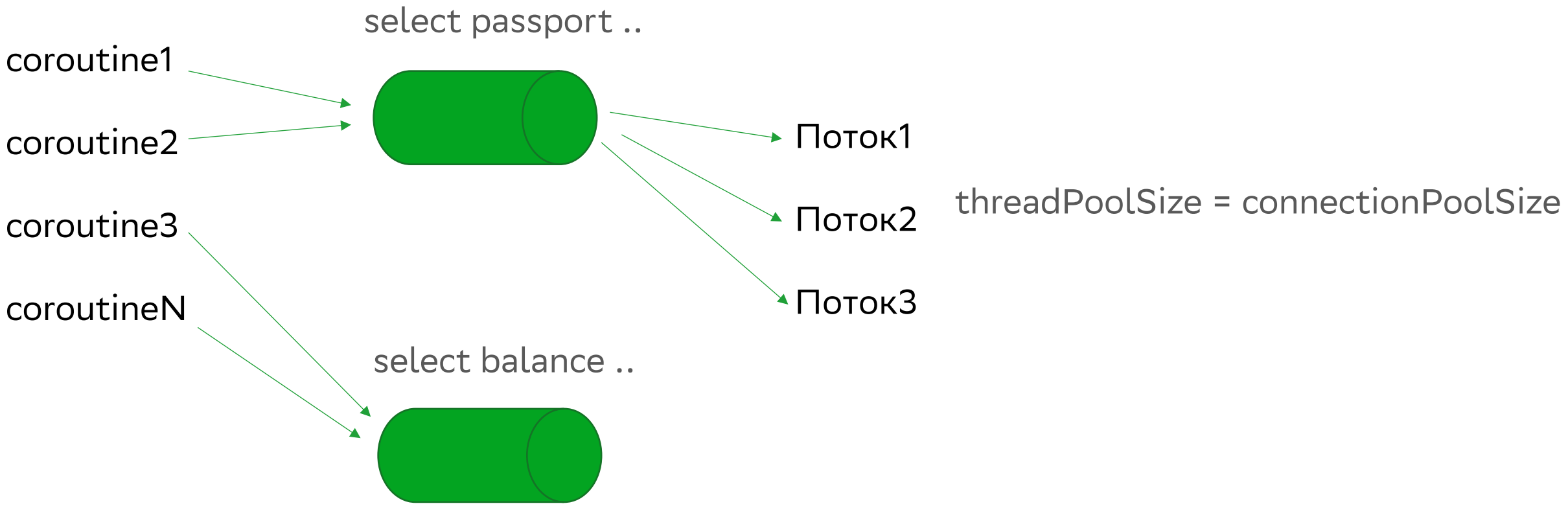
Как
собирать запросы для Batch селекта

Kotlin DSL для BATCH запросов

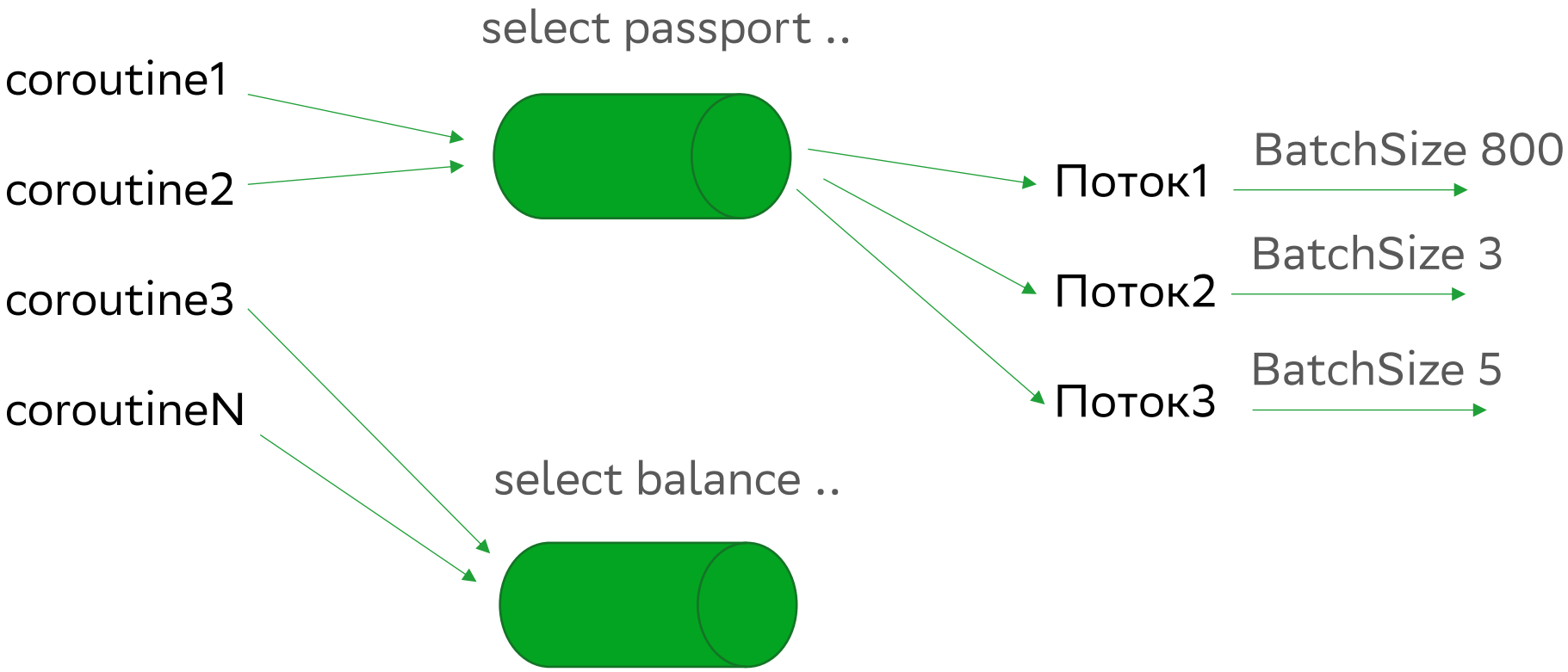
```
coroutine {
    findAll(client) fetchFields { listOf(lastName, firstName) } where {
        passport `=` clientPassport1
    }
}

coroutine {
    findAll(client) fetchFields { listOf(lastName, firstName) } where {
        passport `=` clientPassport2
    }
}

coroutine {
    findAll(client) fetchFields { listOf(lastName, firstName) } where {
        passport `=` clientPassport3
    }
}
```



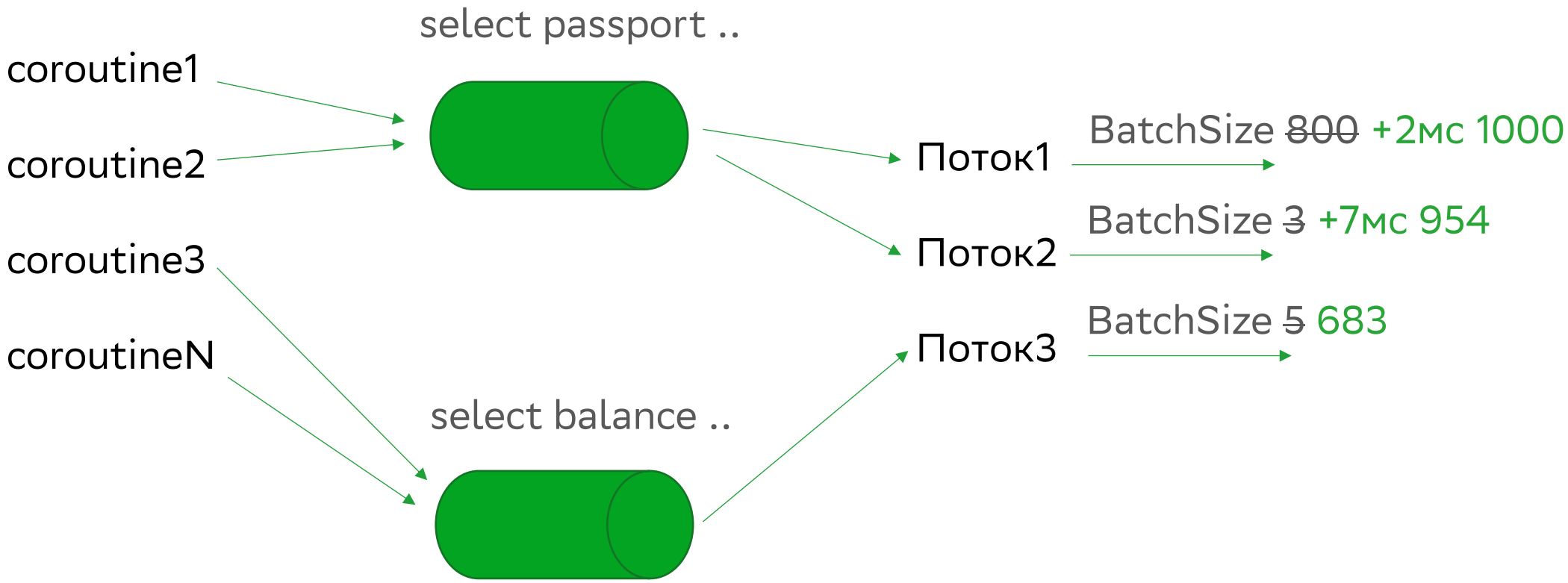
Как каждому потоку набирать 1тыс задач для Batch селекта?



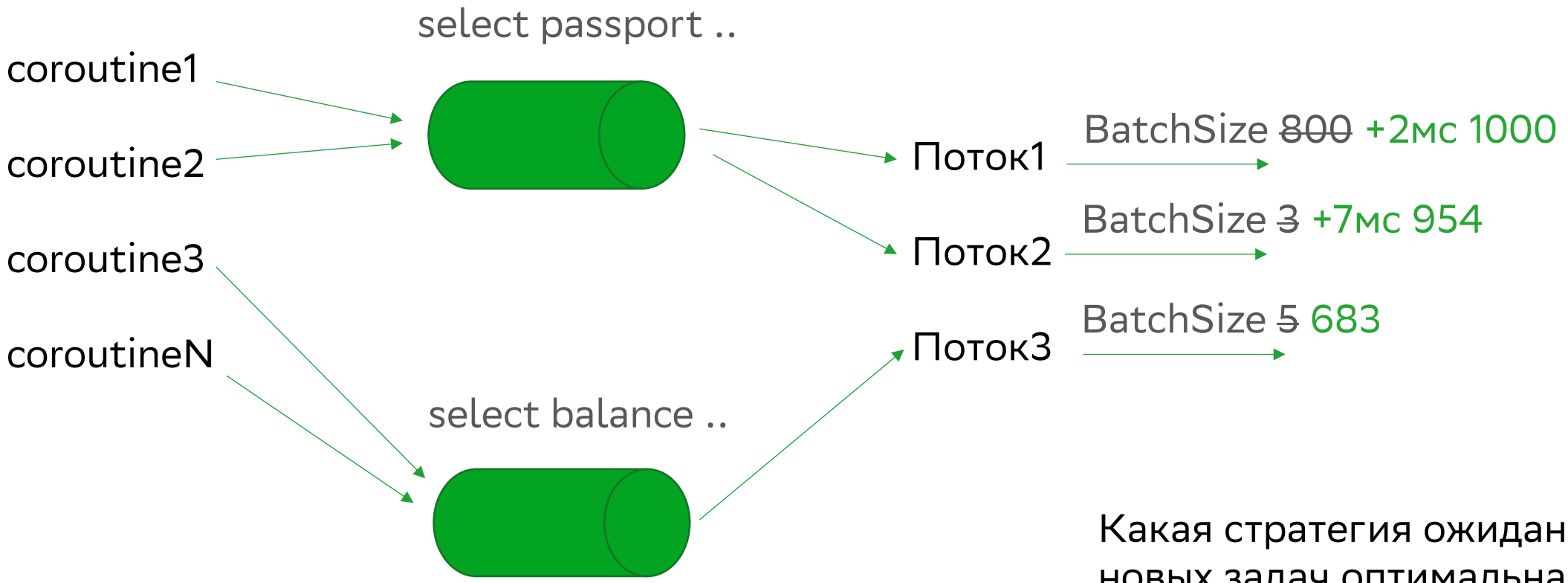
Как каждому потоку набирать 1тыс задач для Batch селекта?



Как каждому потоку набирать 1тыс задач для Batch селекта?



Как каждому потоку набирать 1тыс задач для Batch селекта?



Какая стратегия ожидания новых задач оптимальная?

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
```

Мы можем считать количество живых корутин, чтобы принимать решение: ждать добавление новых задач в очередь или нет.

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (client in clients) {
    for (rule in rules) {
        rule.score(client)
    }
}
```

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    for (client in clients) {
        rule.score(client) //больше вероятность, что данные для SQL в бд будут в кеше
    }
}
```

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope { //ожидает завершения всех корутин, запущенных внутри
        for (client in clients) {
            launch { //запускает корутину
                rule.score(client)
            }
        }
    }
}
```

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope { //ожидает завершения всех корутин, запущенных внутри
        for (client in clients) {
            launch { //запускает корутину
                rule.score(client)
            }
        }
    }
}
```

Скорость создания + выполнения корутин ~2 млн/сек.
Размер корутины ~600 байт. 1млн~600мб.

https://youtu.be/zluKcazgkV4?si=gRR6AD-8fABJOVNu&t=1965

Coroutines and Loom behind the scenes by Roman Elizarov

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope { //ожидает завершения всех корутин, запущенных внутри
        for (client in clients) {
            launch { //запускает корутину
                rule.score(client)
            }
        }
    }
}
```

Скорость создания + выполнения корутин ~2 млн/сек.
Размер корутины ~600 байт. 1млн~600мб.

connectionPoolSize: 30
batchSize: 500

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope { //ожидает завершения всех корутин, запущенных внутри
        for (client in clients) {
            launch { //запускает корутину
                rule.score(client)
            }
        }
    }
}
```

Скорость создания + выполнения корутин ~2 млн/сек.
Размер корутины ~600 байт. 1млн~600мб.

connectionPoolSize: 30
batchSize: 500

Если в каждой корутине создается 1 запрос в бд, то для набора 500 запросов для 30 соединений надо запустить 500 * 30 = 15тыс одновременных корутин.

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope { //ожидает завершения всех корутин, запущенных внутри
        for (client in clients) {
            launch { //запускает корутину
                rule.score(client)
            }
        }
    }
}
```

Скорость создания + выполнения корутин ~2 млн/сек.
Размер корутины ~600 байт. 1млн~600мб.

connectionPoolSize: 30
batchSize: 500

Если в каждой корутине создается 1 запрос в бд, то для набора 500 запросов для 30 соединений надо запустить 500 * 30 = 15тыс одновременных корутин.

Имеет смысл создать min(clients.size, connectionPoolSize * batchSize * 2) корутин.
В каждой корутине обрабатываем 1млн/30тыс=~33 клиента.

Что если заранее знаем размер входных данных

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope {
        val corCount = calcCoroutineCount(clients.size) //30тыс
        clients.splitToChunks(corCount).forEach { chunk -> //chunk.size = 33
            launch {
                for (client in chunk) {
                    rule.score(client)
                }
            }
        }
    }
}
```

Имеет смысл создать min(clients.size, connectionPoolSize * batchSize * 2) корутин.
В каждой корутине обрабатываем 1млн/30тыс=~33 клиента.

Считаем живых корутин

СБЕР

```
val clients = get1MillionClients()
val rules = get300Rules()

for (rule in rules) {
    coroutineScope {
        val corCount = calcCoroutineCount(clients.size)
        val batchContext = BatchContext(coroutines = corCount, finished = 0, parked = 0)

        clients.splitToChunks(corCount).forEach { chunk ->
            launch(batchContext) {
                for (client in chunk) {
                    rule.score(client)
                }
                batchContext.onCoroutineFinish()
            }
        }
    }
}
```

BatchContext

```
val clients = findAll(client) fetchFields { listOf(lastName, firstName) } {..}
```



```
val task = createSqlTask(dslFilters)

val context = batchContext()
if (context == null) return useSharedQueue(task)

context.batchAndProcess(task)
return task.awaitResult()
```

```
class BatchContext<R>(
    private val coroutineCount: Int,
    private var parkedCoroutines: Int,
    private var finishedCoroutines: Int
) {
    private val queue = ArrayList<R>(batchSize)

    fun batchAndProcess(task: R) {
        queue.add(task)

        if (lastAliveCoroutine() || queue.size == batchSize) {
            execute(queue)
        }
        ++parkedCoroutines
    }
}
```

executeInBatchContext

СБЕР

```
val result = clients.executeInBatchContext { client -> process(client) }
```

- Рассчитываем оптимальное количество корутин
- Запускаем корутины
- Каждой корутине добавляем в контекст BatchContext
- Отслеживаем количество завершенных и ожидающих корутин

На какую производительность можно рассчитывать

СБЕР

```
while (true) {
    val passports: List<Long> = getOneMillionRandomPassports()

    volatileField = passports.executeInBatchContext { passport ->
        val byNumbers = findByNumberPassport(passport)
        val byStrings = findByStringPassport(passport.toString())
        byNumbers + byStrings
    } // 1.5kk select/sec.  PostgreSQL 95% cpu
}


private suspend fun findNameByNumber(passport: Long): String? = findFirst(client.name) where {
    client.passport_number `=` passport
}

private suspend fun findNameByString(passport: String): String? = findFirst(client.name) where {
    client.passport_string `=` passport
}
```

Что мы обсудили

- Как работают PgPreparedStatement
- Batch селекты могут повысить производительность в 10+ раз
- Можно не менять сигнатуры методов для batch обработки
- Использовать оптимальные алгоритмы для набора нужного batchSize
- Использовать DSL для абстрагирования от batch обработки

alx.matorin@gmail.com Александр Маторин

Спасибо!

Рад ответить на вопросы!

alx.matorin@gmail.com Александр Маторин