TINKOFF

Разработка распределенной очереди с отложенными задачами на основе PostgreSQL

Максим Иванов

09.11.2022

TINKOFF

Максим Иванов
Старший разработчик @ TINKOFF

@QUCKLY

linkedin.com/in/maxim-ivanov/

План

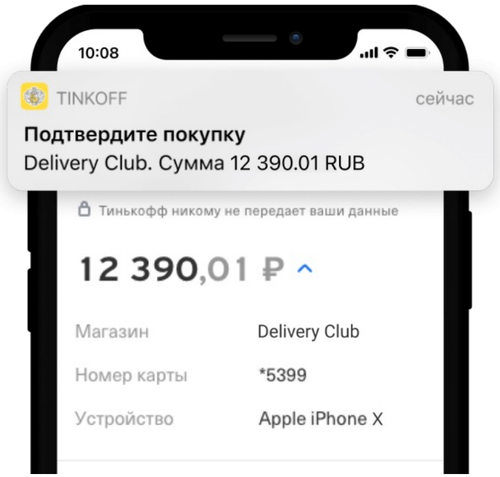
ОЧЕРЕДЬ НА POSTGRES

ОТЛАДКА БД

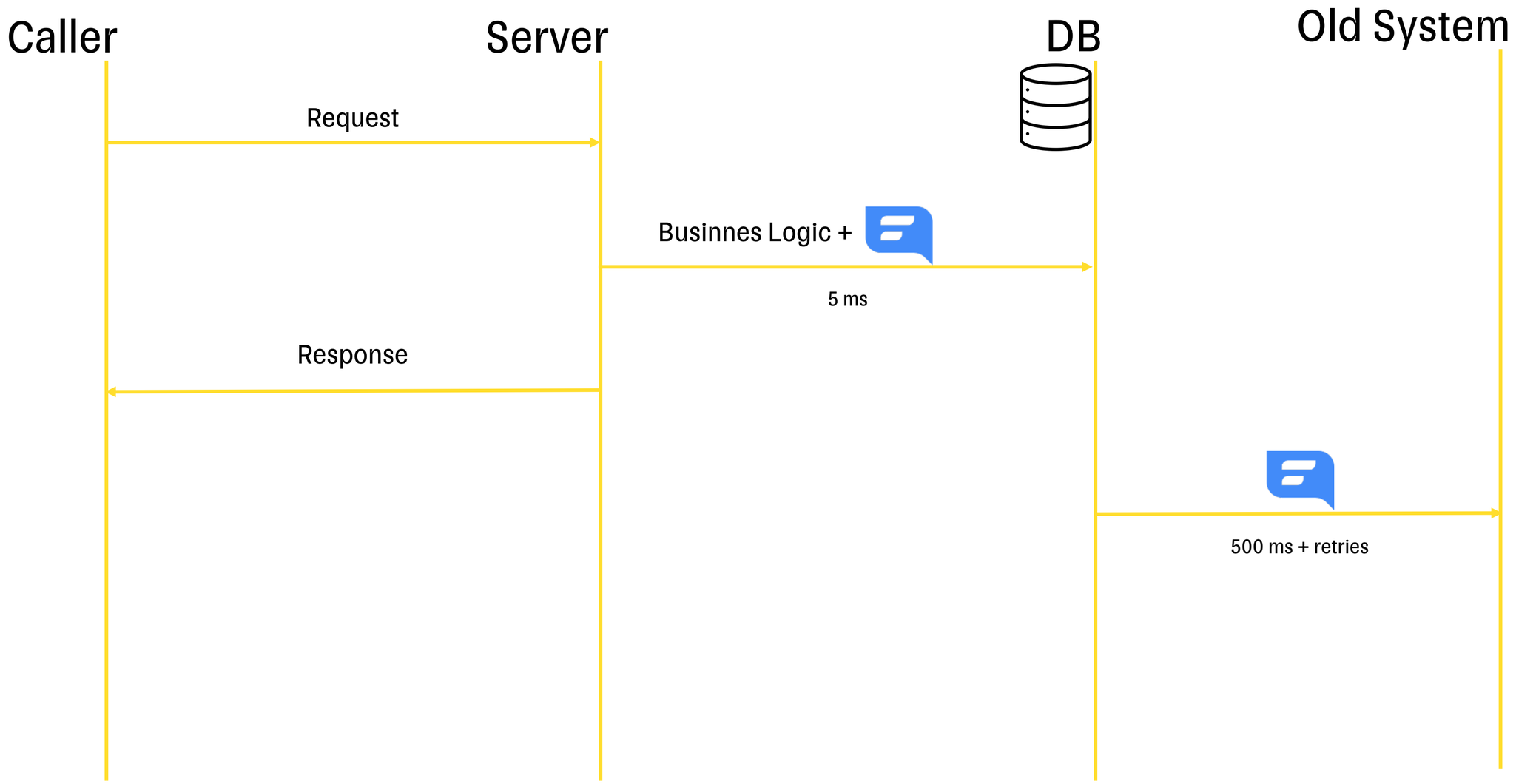
ПРОИЗВОДИТЕЛЬНОСТЬ

Проблема

Что это за задача

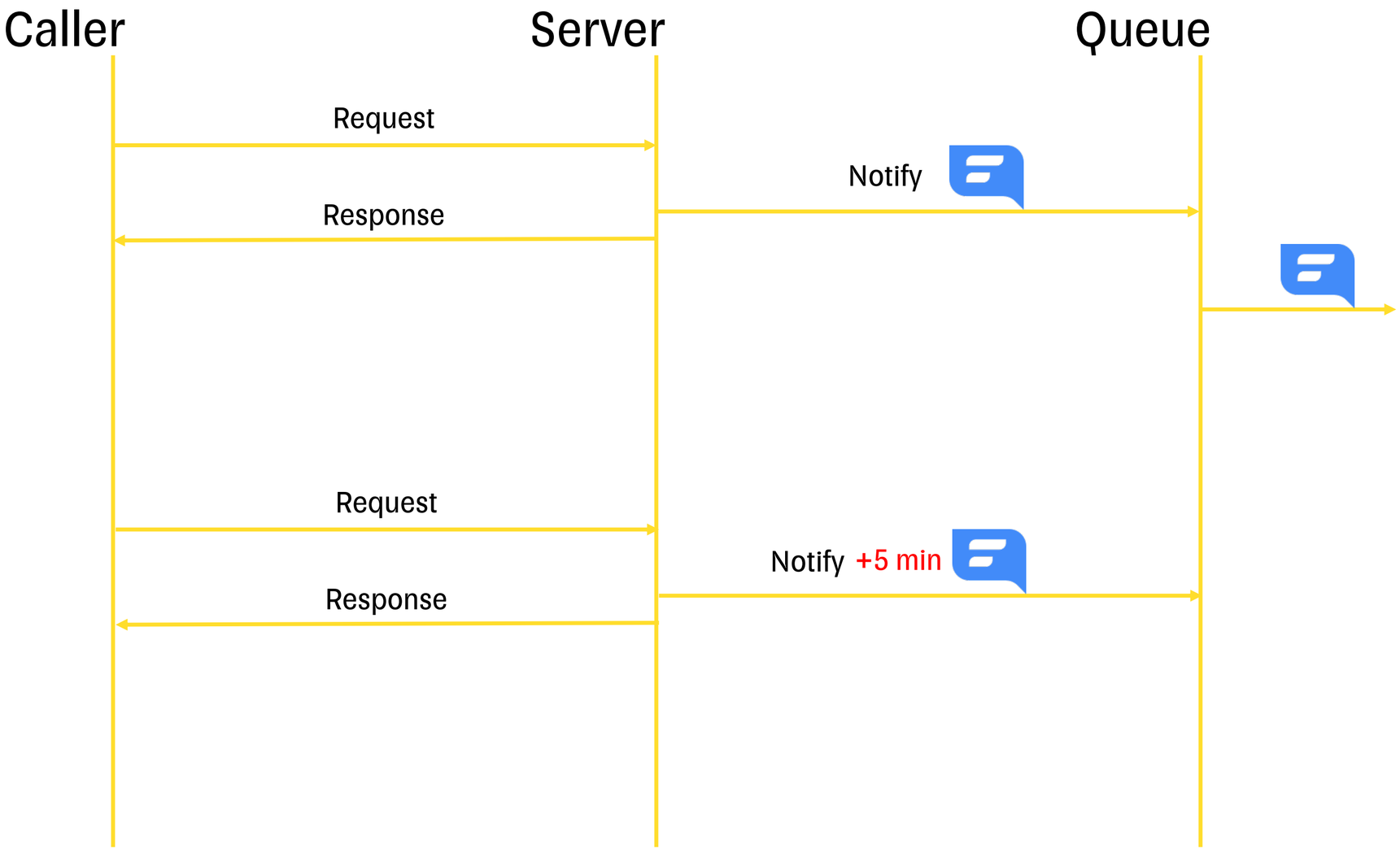


Задача в фоне



Отложенная задача

Отложенная задача



Решения

Решения

ExecutorService

NoSQL/Redis/...

MQ/ActiveMQ/Kafka/...

Требования

- At-least-once
- Distributed
- Retry
- No latency

Решение

Отложенная задача

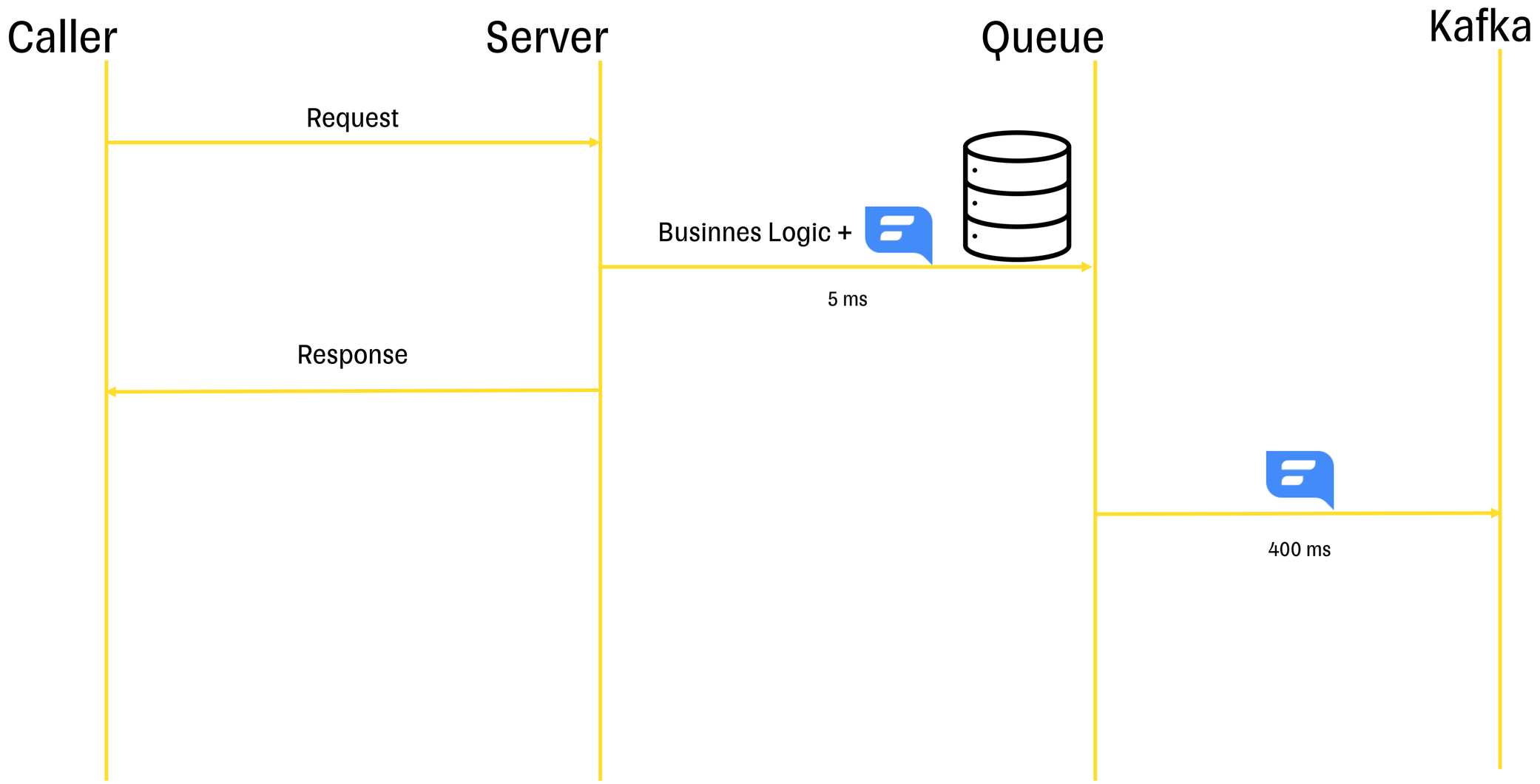


Таблица задач

```sql
CREATE TABLE task_queue
(
    ID              VARCHAR(32)                 NOT NULL,



    TASK_PAYLOAD    TEXT,

    NEXT_RETRY_TIME TIMESTAMP,
    RETRY_COUNTER   INT,
    STATUS          INT
)
```

Таблица задач

```sql
CREATE TABLE task_queue
(
    ID              VARCHAR(32)                 NOT NULL,


    ENTITY_ID       VARCHAR(32)                 NOT NULL,
    TASK_PAYLOAD    TEXT,

    NEXT_RETRY_TIME TIMESTAMP,
    RETRY_COUNTER   INT,
    STATUS          INT
)
```

Таблица задач

```
CREATE TABLE task_queue
(
    ID               VARCHAR(32)                    NOT NULL,
    CREATED          TIMESTAMP                      NOT NULL,
    UPDATED          TIMESTAMP                      NOT NULL,
    ENTITY_ID        VARCHAR(32)                    NOT NULL,
    TASK_PAYLOAD     TEXT,
    TASK_TYPE        INT                            NOT NULL,
    NEXT_RETRY_TIME  TIMESTAMP,
    RETRY_COUNTER    INT,
    STATUS           INT
)
```

Таблица задач

```
CREATE TABLE task_queue
(
    ID               VARCHAR(32)                    NOT NULL,
    CREATED          TIMESTAMP                      NOT NULL,
    UPDATED          TIMESTAMP                      NOT NULL,
    ENTITY_ID        VARCHAR(32)                    NOT NULL,
    TASK_PAYLOAD     TEXT,
    TASK_TYPE        INT                            NOT NULL,
    NEXT_RETRY_TIME  TIMESTAMP,
    RETRY_COUNTER    INT,
    STATUS           INT
) PARTITION BY RANGE (CREATED);
```

Создание новой задачи

```
@Transactional
fun processPayment() {
    ...
    businessRepository.add(BusinessEntity(...))
    taskRepository.createTask(Task(
        taskPayload = "notify"
    ))
}
```

```
INSERT INTO task_queue (id, task_payload)
VALUES ('id0123', 'notify')
```

Поиск следующей задачи

1

```
while (true) {
    fetchAndExecuteTask()

    Thread.sleep(10)
}
```

2 `await / notify`

Поиск следующей задачи

```
@Transactional
fun fetchAndExecuteTask() {
    val task = taskRepository.findNextTask(
        LocalDateTime.now() )

    try {
        handle(task)
        task.status = SUCCESS
    } catch (e: Exception) {
        task.status = ERROR
    }

    taskRepository.update(task)
}
```

Поиск следующей задачи

id	task	next_retry_time	status
…	…	…	
5	task1	2022-04-28 19:40:33	null
6	task2	2022-04-28 19:40:39	null
…	…	…	

```
SELECT *
FROM task_queue
WHERE status IS NULL
  AND next_retry_time <= '2022-04-28 19:45:33'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

Поиск следующей задачи

Worker 2

Worker 1

id	task	next_retry_time	status
…	…	…	
5	task1	2022-04-28 19:40:33	null
6	task2	2022-04-28 19:40:39	null
…	…	…	

```
SELECT *
FROM task_queue
WHERE status IS NULL
  AND next_retry_time <= '2022-04-28 19:45:33'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

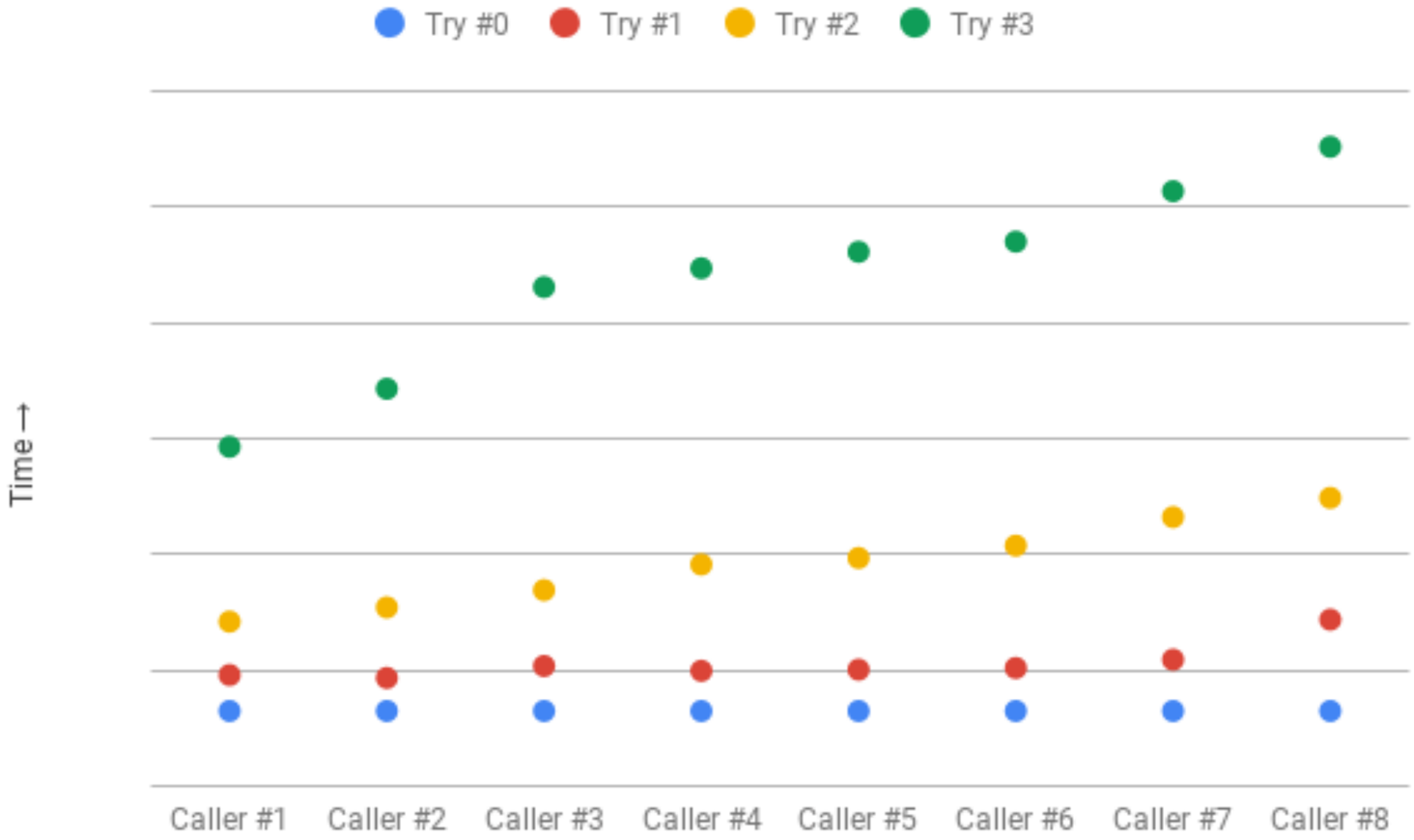
Ретраи

```
if (task.retryCounter < MAX_RETRIES) {
    task.retryCounter++
    task.nextRetryTime = LocalDateTime.now()
                         +
                         1.seconds

} else {
    task.status = ERROR
}
```

Ретраи

```
if (task.retryCounter < MAX_RETRIES) {
    task.retryCounter++
    task.nextRetryTime = LocalDateTime.now()
                         +
     randomDistribution(task.retryCounter)

} else {
    task.status = ERROR
}
```

Ретраи / Exponential Backoff + Jitter



baeldung.com

Фильтр ожидающих задач

Worker 2

id	task	next_retry_time	status
…	…	…	
5	task5	2023-12-12 00:00:00	null
6	task6	2022-04-28 19:40:39	null
…	…	…	

```
SELECT *
FROM task_queue
WHERE status IS NULL
  AND next_retry_time <= '2022-04-28 19:45:33'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

Объединение задач по entity_id

Объединение задач по entity_id

Worker 1

id	task	entity_id	next_retry_time	status
…	…	…	…	…
4	notify	1	2022-04-28 19:39:01	OK
5	task5	1	2022-04-28 20:45:33	null
6	task6	1	2022-04-28 19:40:34	null
7	push	2	2022-04-28 19:40:39	null
…	…	…	…	…

Объединение задач по entity_id

```
@Transactional
  SELECT * FROM task_queue WHERE status IS NULL AND time …
  LIMIT 1 FOR UPDATE SKIP LOCKED
```

6	task6	1	2022-04-28 19:40:34	null

Объединение задач по entity_id

```
@Transactional
  SELECT * FROM task_queue WHERE status IS NULL AND time …
  LIMIT 1 FOR UPDATE SKIP LOCKED
```

6	task6	1	2022-04-28 19:40:34	null

```
  SELECT * FROM task_queue
  WHERE status IS NULL AND entity_id = '1'
  ORDER BY created
  LIMIT 100 FOR UPDATE NOWAIT
```

5	task5	1	2022-04-28 20:45:33	null
6	task6	1	2022-04-28 19:40:34	null

Конкурентность

Worker 1

Worker 2

id	task	entity_id	next_retry_time	status
…	…	…	…	…
4	notify	1	2022-04-28 19:39:01	OK
5	task5	1	2022-04-28 19:40:33	null
6	task6	1	2022-04-28 19:40:34	null
7	push	2	2022-04-28 19:40:39	null
…	…	…	…	…

Конкурентность

Worker 1

Worker 2

id	task	entity_id	next_retry_time	status
…	…	…	…	…
4	notify	1	2022-04-28 19:39:01	OK
5	task5	1	2022-04-28 19:40:33	null
6	task6	1	2022-04-28 19:40:34	null
7	push	2	2022-04-28 19:40:39	null
…	…	…	…	…

Конкурентность

Worker 1

Worker 2

id	task	entity_id	next_retry_time	status
…	…	…	…	…
4	notify	1	2022-04-28 19:39:01	OK
5	task5	1	2022-04-28 19:40:33	nul
6	task6	1	2022-04-28 19:40:34	null
7	push	2	2022-04-28 19:40:39	null
…	…	…	…	…

Конкурентность

Worker 1

Worker 2

id	task	entity_id	next_retry_time	status
…	…	…	…	…
4	notify	1	2022-04-28 19:39:01	OK
5	task5	1	2022-04-28 19:40:33	nul
6	task6	1	2022-04-28 19:40:34	null
7	push	2	2022-04-28 19:40:39	null
…	…	…	…	…

```
LIMIT 100 FOR UPDATE NOWAIT
```

Конкурентность

Worker 1

Worker 2

id	task	entity_id	next_retry_time	status
…	…	…	…	…
4	notify	1	2022-04-28 19:39:01	OK
5	task5	1	2022-04-28 19:40:33	null
6	task6	1	2022-04-28 19:40:34	null
7	push	2	2022-04-28 19:40:39	null
…	…	…	…	…

```
LIMIT 100 FOR UPDATE NOWAIT
```

Быстрое прекращение выполнения задач по entity_id

```
SELECT * FROM task_queue
WHERE status IS NULL AND entity_id = '1’
ORDER BY created
LIMIT 100 FOR UPDATE NOWAIT
```

Проблемы

Ищем следующую задачу

300/300k

0.646 ms

```
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

Ищем следующую задачу

300/300k

0.646 ms

150k/600k

159.030 ms

```
->  Index Scan
Rows Removed by Filter:
150579
```

420k/900k

1007.132 ms

895k/1400k

1477.383 ms

```
Rows Removed by Filter:
895920
```

Между каждым EXPLAIN вызывался

ANALYZE task_queue;

```
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

Поиск следующей задачи

```
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= ‘2020-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

```
 ->  Index Scan using task_queue_partition_2021_12_02_status_idx1 …
(actual time=2672.865..2672.870 rows=0 loops=1)
Index Cond: (t_3.status IS NULL)
Filter: (t_3.next_retry_time <= '2020-12-03 00:00:00'::timestamp)
Rows Removed by Filter: 895920
Buffers: shared hit=81 read=157859 written=11
```

Поиск следующей задачи

```
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= ‘2020-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

```
 ->  Index Scan using task_queue_partition_2021_12_02_status_idx1 …
(actual time=2672.865..2672.870 rows=0 loops=1)
Index Cond: (t_3.status IS NULL)
Filter: (t_3.next_retry_time <= '2020-12-03 00:00:00'::timestamp)
Rows Removed by Filter: 895920
Buffers: shared hit=81 read=157859 written=11
```

Отладка запросов

pg_stat_all_indexes

- Размеры индексов
- Количество чтений
- «Полезность» индекса

pg_stats

- Уникальность данных в колонках
- Частота Nullable
- Узнать, что БД думает о данных

EXPLAIN (ANALYZE)

- Отладка запросов
- Показывает план выполнения запроса
- Какие индексы использует или нет
- Время работы запроса

Статистика индексов

	id	next_retry_time	status
	18 MB (5%)	13 MB (4%)	3.8 MB (1%)
idx_scan	100%	0.8%	33%

Частичный индекс

```
CREATE INDEX task_queue_idx_status_null
    ON task_queue (status) WHERE status IS NULL
```

ALL	ONLY~~FANS~~ NULLS	
300 rows / 300’000		
3792 kB	16 kB	0.42%
150500 rows / 600’000		
7408 kB	1824 kB	24%

Индекс по next_retry_time

```
CREATE INDEX task_queue_status_next_retry_time_idx
    ON task_queue (next_retry_time) WHERE status IS NULL
```

IS NULLS	IS NULLS + `next_retry_time`
6.5 MB	27 MB

1’400’000 rows

Какие получились индексы

```
CREATE INDEX ON task_queue
```

(next_retry_time)

WHERE status IS NULL

27 MB (2%) Частичный

```
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

Какие получились индексы

```
CREATE INDEX ON task_queue
```

(next_retry_time)

WHERE status IS NULL

27 MB (2%) Частичный

```
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

(entity_id, created)

WHERE status IS NULL

42 MB (3%) Композитный, частичный

```
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.entity_id = 'JOKER_ENTITY’
ORDER BY t.created
LIMIT 100 FOR UPDATE NOWAIT
```

Какие получились индексы

```
CREATE INDEX ON task_queue
```

(next_retry_time)

WHERE status IS NULL

27 MB (2%) Частичный

```
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

(entity_id, created)

WHERE status IS NULL

42 MB (3%) Композитный, частичный

```
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.entity_id = 'JOKER_ENTITY’
ORDER BY t.created
LIMIT 100 FOR UPDATE NOWAIT
```

(id)

75 MB (5%)

Таблица задач

```
CREATE TABLE task_queue
(
    ID               VARCHAR(32)                    NOT NULL,
    CREATED          TIMESTAMP                      NOT NULL,
    UPDATED          TIMESTAMP                      NOT NULL,
    ENTITY_ID        VARCHAR(32)                    NOT NULL,
    TASK_PAYLOAD     TEXT,
    TASK_TYPE        INT                            NOT NULL,
    NEXT_RETRY_TIME  TIMESTAMP,
    RETRY_COUNTER    INT,
    STATUS           INT
) PARTITION BY RANGE (CREATED);
```

Поиск следующей задачи

```
EXPLAIN
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

```
Limit  (cost=0.14..0.40 rows=1 width=1067)
  ->  LockRows  (cost=0.14..231051.79 rows=899612 width=1067)
        ->  Append  (cost=0.14..222055.67 rows=899612 width=1067)
              ->  Index Scan using task_queue_partition_2021_11_30_next_retry_
(cost=0.14..23.84 rows=40 width=1065)
                    Index Cond: (next_retry_time <= '2021-12-03 00:00:00'::tim
                    Filter: (status IS NULL)
              ->  Index Scan using task_queue_partition_2021_12_01_next_retry_
(cost=0.14..17.49 rows=20 width=1067)
                    Index Cond: (next_retry_time <= '2021-12-03 00:00:00'::tim
                    Filter: (status IS NULL)
              ->  Seq Scan on task_queue_partition_2021_12_02 t_3  (cost=0.00.
                    Filter: ((status IS NULL) AND (next_retry_time <= '2021-12
              ->  Index Scan using task_queue_partition_2021_12_03_next_retry_
(cost=0.12..8.14 rows=1 width=242)
                    Index Cond: (next_retry_time <= '2021-12-03 00:00:00'::tim
                    Filter: (status IS NULL)
              ->  Index Scan using task_queue_partition_2021_12_04_next_retry_
(cost=0.12..8.14 rows=1 width=242)
                    Index Cond: (next_retry_time <= '2021-12-03 00:00:00'::tim
                    Filter: (status IS NULL)
```

Поиск следующей задачи

```
->  Index Scan using task_queue_partition_2021_11_30 ...
->  Index Scan using task_queue_partition_2021_12_01 ...
      (never executed)
```

```sql
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.next_retry_time <= '2021-12-03'
LIMIT 1 FOR UPDATE SKIP LOCKED
```

Поиск всех задач по сущности

```
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.entity_id = 'JOKER_ENTITY’
ORDER BY t.created
LIMIT 100 FOR UPDATE NOWAIT
```

```
Limit  (cost=0.96..47.97 rows=8 width=861)
  ->  LockRows  (cost=0.96..47.97 rows=8 width=861)
        ->  Append  (cost=0.96..47.89 rows=8 width=861)
              ->  Index Scan using task_queue_partition_2021_11_30_entity_id_created
                    Index Cond: ((entity_id)::text = 'JOKER_ENTITY'::text)
                    Filter: (status IS NULL)
              ->  Index Scan using task_queue_partition_2021_12_01_entity_id_created
                    Index Cond: ((entity_id)::text = 'JOKER_ENTITY'::text)
                    Filter: (status IS NULL)
...
Planning Time: 0.282 ms
Execution Time: 0.064 ms
```

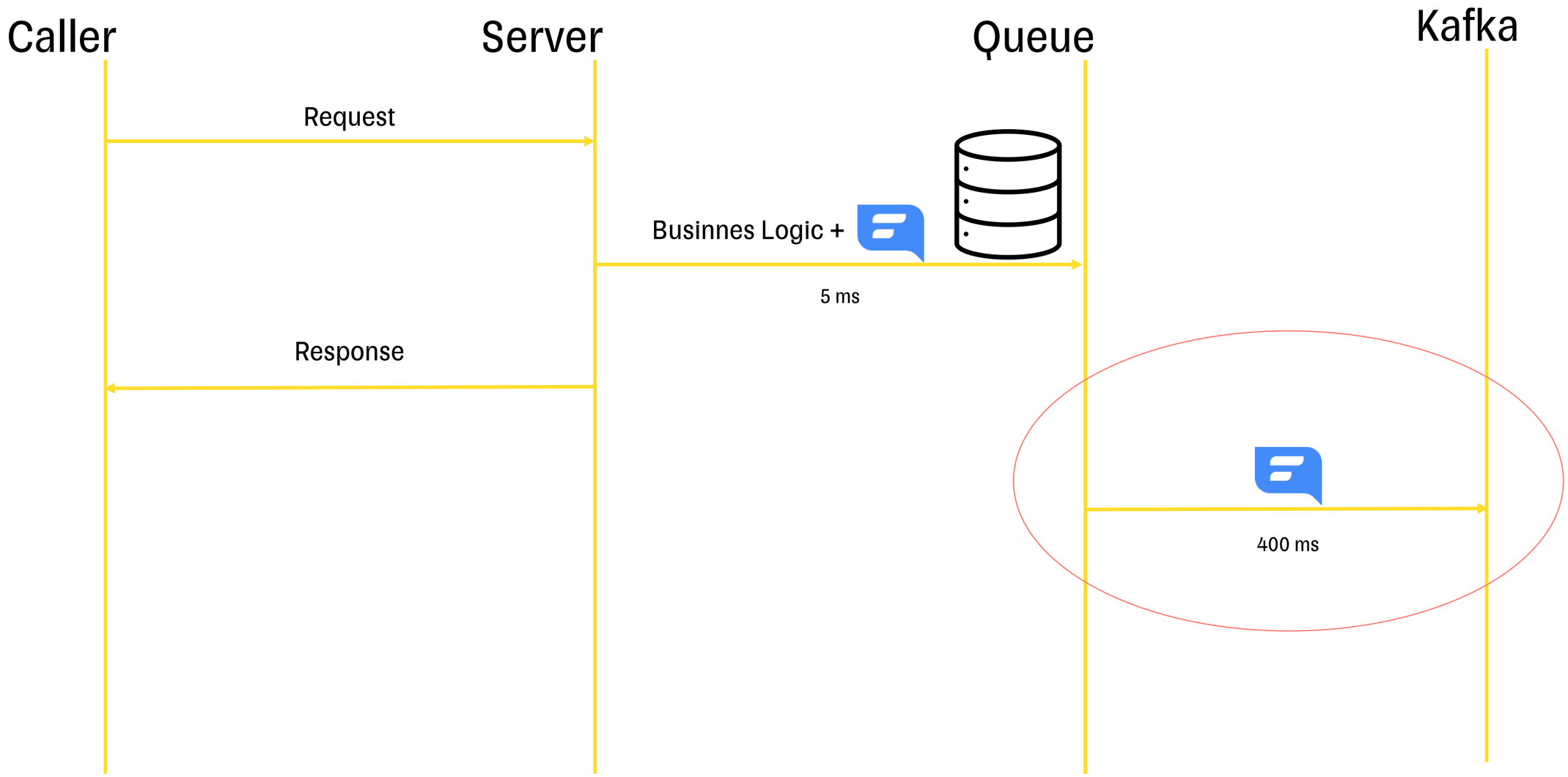
Поиск всех задач по сущности

```
Limit  (cost=0.96..47.97 rows=8 width=861)
  ->  LockRows  (cost=0.96..47.97 rows=8 width=861) (actual
        ->  Append  (cost=0.96..47.89 rows=8 width=861) (ac
              ->  Index Scan using task_queue_partition_202
                    Index Cond: ((entity_id)::text = 'JOKER
                    Filter: (status IS NULL)
              ->  Index Scan using task_queue_partition_202
                    Index Cond: ((entity_id)::text = 'JOKER
                    Filter: (status IS NULL)
...
Planning Time: 0.282 ms
Execution Time: 0.064 ms
```

```
EXPLAIN (ANALYZE, VERBOSE, BUFFERS)
SELECT * FROM task_queue t
WHERE t.status IS NULL
  AND t.entity_id = 'JOKER_ENTITY’
ORDER BY t.created
LIMIT 100 FOR UPDATE NOWAIT
```

Уточнения

Открытая транзакция



RPS

$$\text{Max TPS} = \frac{\text{DB Connections}}{\text{Execution time}}$$

$$\frac{100}{100\text{ms}} = 1\ 000\ \text{TPS}$$

Настройки PG

Разный пул коннектов для worker и бизнес логики

-c max_connections=2000

Настраиваем таймауты

-c statement_timeout=5s

-c idle_in_transaction_session_timeout=5s

Что получилось?

Индекс как очередь

Плюсы/минусы

- Требует открытый коннект к БД
- Больше подходит для синхронного (блокирующего) стека
- Нужно правильно создать индекс
- Для задач по одной сущности, работает с малым кол-вом задач

- Создаем задачу НЕ увеличивая время ответа транзакции, вместе с бизнес логикой

- Переложили почти всю ответственность на Postgres

- Гибкое решение

Задачи могут быть любые и без ретраев
Добавление приоритетов
Сохранение результатов выполнения в отдельную таблицу
Перенос выполненных задач в отдельную таблицу

Ссылки

youtube - Queues in PostgreSQL | Citus Con: An Event for Postgres 2022

habr.com - Очередь задач в PostgreSQL

amazon - Exponential Backoff And Jitter

baeldung - Better Retries with Exponential Backoff and Jitter

Выводы

- Используйте очередь внутри вашей БД, если можно
- SELECT FOR UPDATE SKIP LOCKED
- Следите за индексами

TINKOFF

Максим Иванов
Старший разработчик @ TINKOFF

@QUCKLY

linkedin.com/in/maxim-ivanov/