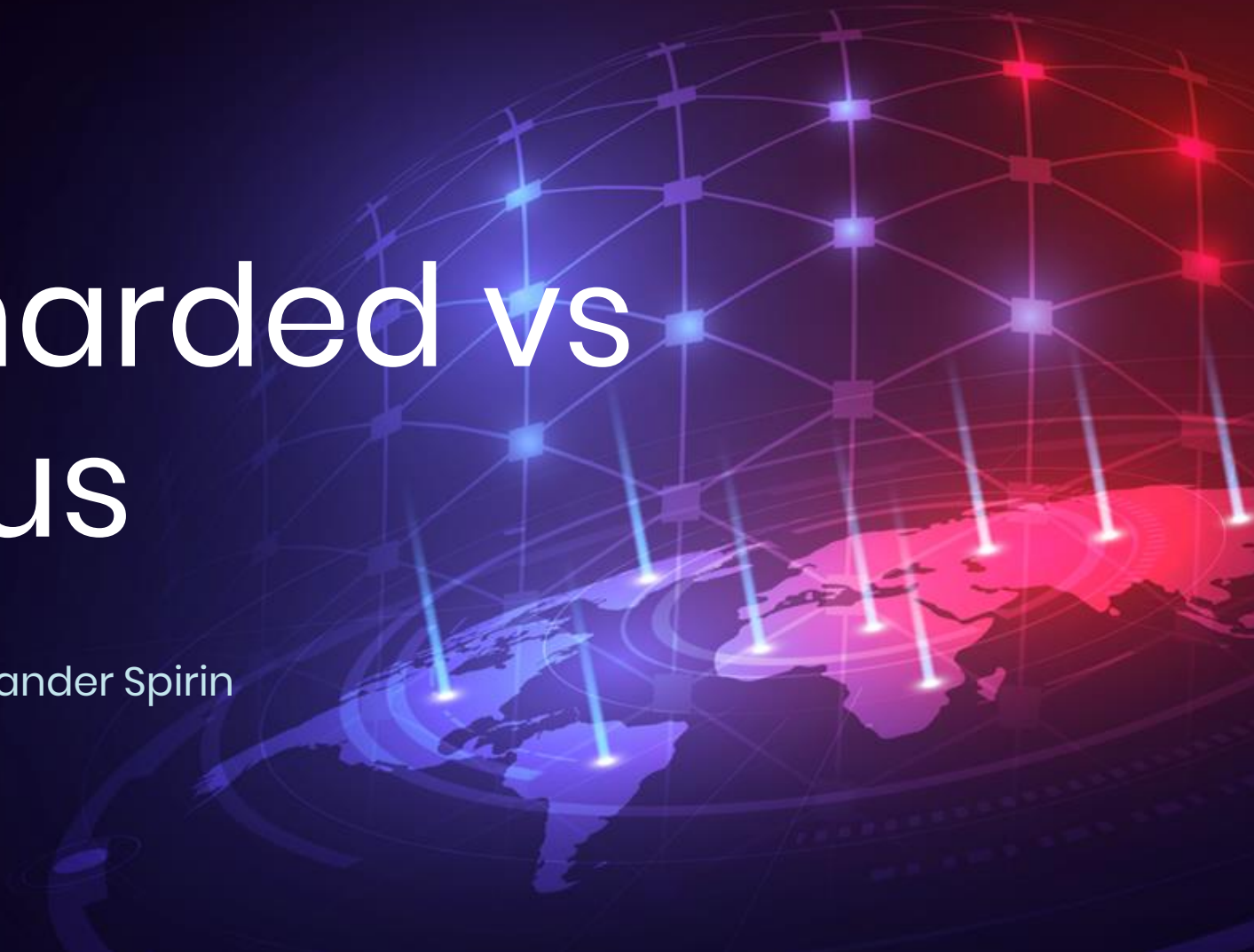
In Country

MongoDB Sharded vs Postgres Citus

Kirill Kalistratov | Vasiliy Soshnikov | Alexander Spirin



Кто мы?

Калистратов Кирилл
Senior Performance

Engineer, InCountry

Василий Сошников
DBA Consultant,

InCountry

Спирин Александр

DBA, InCountry

Что используем?

Taurus

mongoDB

aws

docker

Java

APACHE JMeter™

influxdb

PostgreSQL

ZABBIX

Grafana

Jenkins

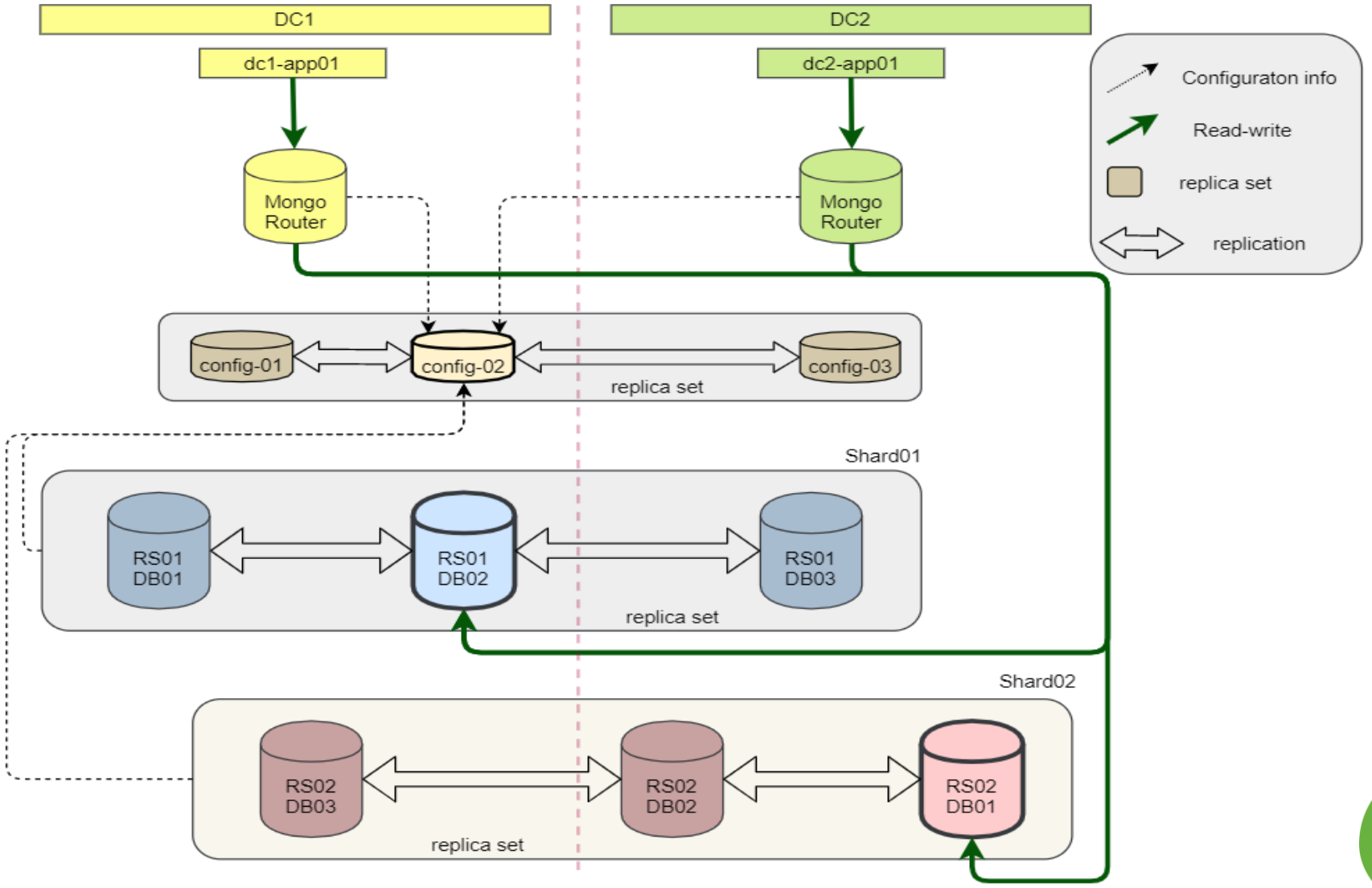
redis

Каких целей хотим достичь?

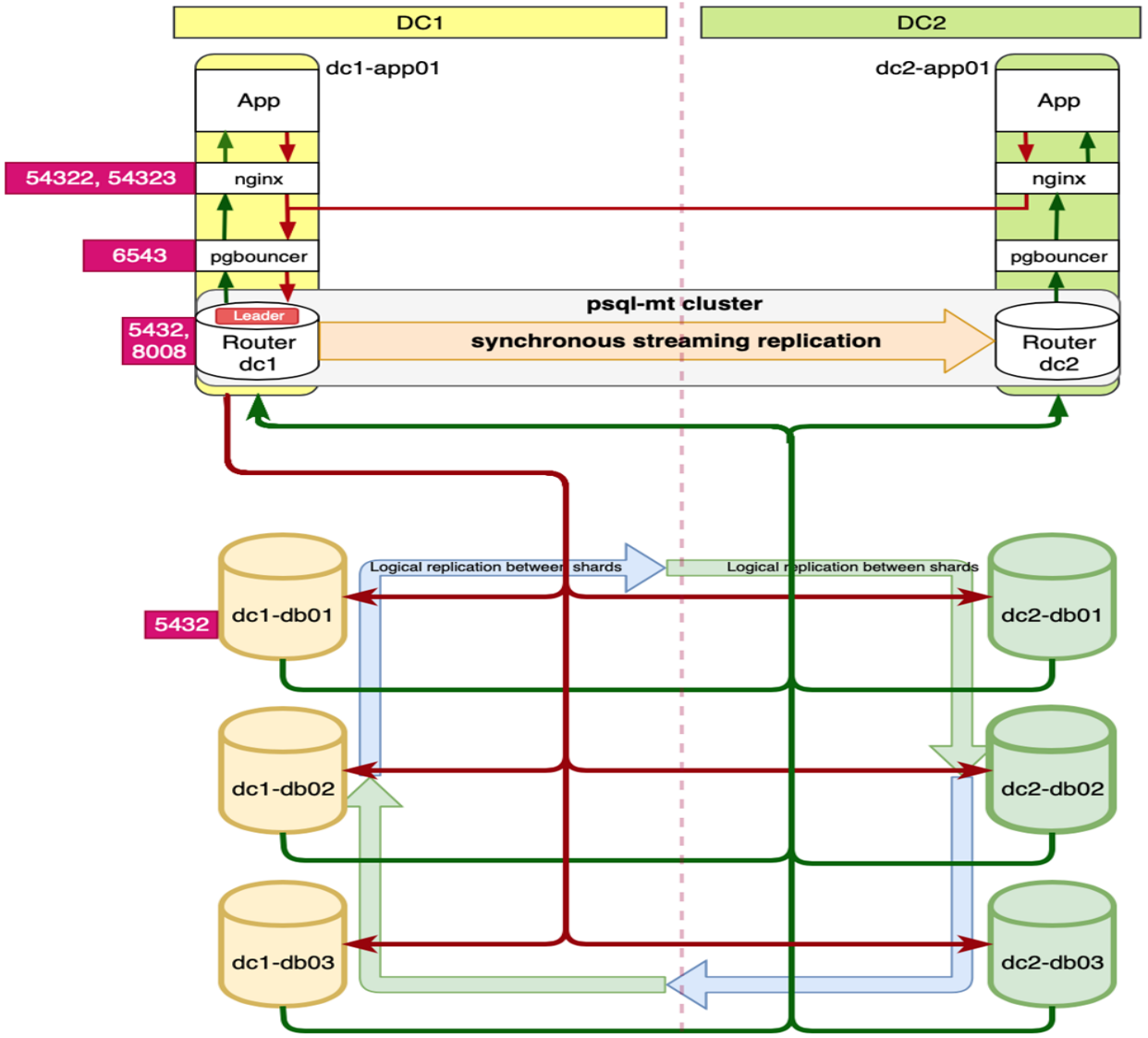
PostgreSQL

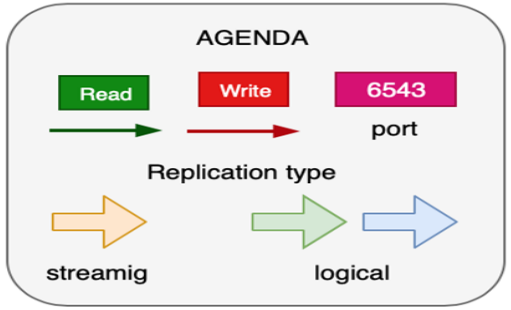
mongoDB

Архитектура решения на базе MongoDB



Архитектура решения на базе CitusDB





Community Edition!

У обоих:

- статическое шардирование;

MongoDB

- CE ~ EE

Citus

- 1 маршрутизатор
- невозможно добавить новый шард “на ходу”

- Особенности:
  - MongoDB - балансировщик;
    долгое построение индексов.
  - Citus -отсутствие контроля за расположением копии данных (Replication Factor)

Схема таблицы

```
incountry=# \d data_table
                    Table "public.data_table"
  Column   |            Type             | Collation | Nullable | Default
-----------+-----------------------------+-----------+----------+---------
 id        | uuid                        |           | not null |
 filter1   | character varying(128)      |           | not null |
 filter2   | character varying(128)      |           |          |
 filter3   | bigint                      |           |          |
 filter4   | character varying(256)      |           |          |
 filter5   | character varying(256)      |           |          |
 document  | text                        |           |          |
 edition   | integer                     |           |          |
 created   | timestamp without time zone |           | not null | now()
 updated   | timestamp without time zone |           | not null | now()
Indexes:
    "pk_data_table" PRIMARY KEY, btree (id, filter1)
    "idx_filter1_hash" hash (filter1)
    "idx_filter4_hash" hash (filter4)
    "idx_filter5_hash" hash (filter5)
    "idx_id_filter2_btree" btree (id, filter2)
    "idx_id_filter3_btree" btree (id, filter3)
```

MongoDB. Аппаратные средства

Для тестирования производительности в наличии 8 экземпляров:

- 2 x router - 2 сервера для роутинга Mongo (c5.xlarge)
- 3 x config - 3 сервера конфигурации (m5.large)
- 6 x DB - 6 серверов с базой данных (m5.large)

Назначение	Конфигурация
Роутинг	4 core cpu with HT, 8Gb, 30G SSD (nvme)
Конфигурация	2 core cpu without HT, 2GB, 20+40Gb SSD (nvme)
База данных	2 core cpu without HT, 8GB, 20+40Gb SSD (nvme)

PostgreSQL. Аппаратные средства

Для тестирования производительности в наличии

8 aws экземпляров m5.large:

- 2 x app - 2 сервера приложения
- 6 x DB - 6 серверов с базой данных

Назначение	Конфигурация
Приложение	4 core cpu with HT, 8Gb, 30G SSD (nvme)
База данных	2 core cpu without HT, 8GB, 20+40Gb SSD (nvme)

Условия проведения тестов

- Разный размер документа
  - 2, 4, 8, 16 КБ и переменный размер
- DB size / RAM ~= 2,6
- Запрос как правило обратится к диску
- Не читаем со вторичных нод MongoDB

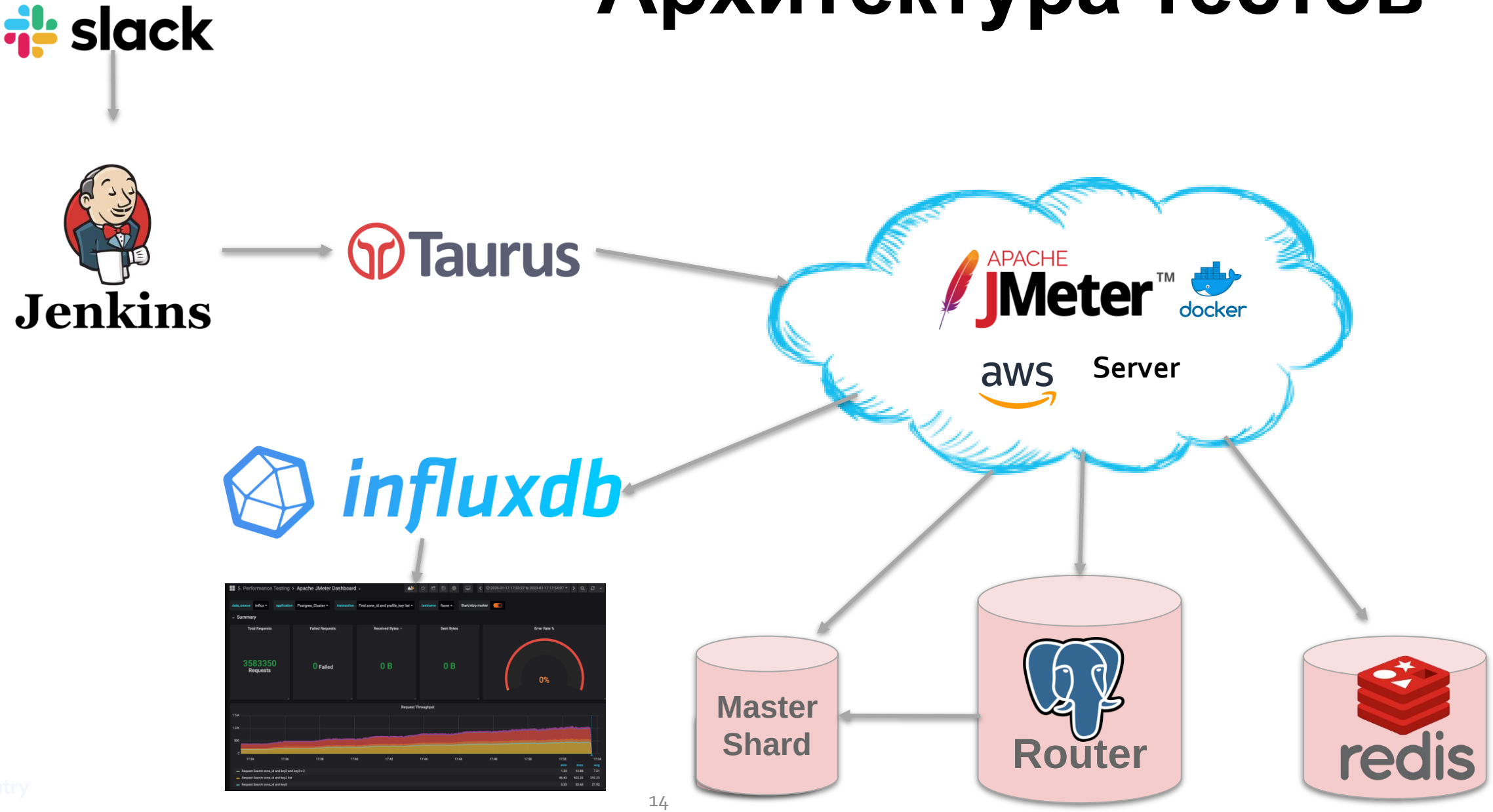
Что мы можем менять?

- Типы индексов
  - Hash, B-Tree
- Режим хранения данных
  - без TOAST
  - TOAST (с компрессией, без компрессии)

Базовая схема

```
incountry=# \d+ data_table
                                        Table "public.data_table"
  Column   |            Type             | Collation | Nullable | Default | Storage  |
-----------+-----------------------------+-----------+----------+---------+----------+
 id        | uuid                        |           | not null |         | plain    |
 filter1   | character varying(128)      |           | not null |         | extended |
 filter2   | character varying(128)      |           |          |         | extended |
 filter3   | bigint                      |           |          |         | plain    |
 filter4   | character varying(256)      |           |          |         | extended |
 filter5   | character varying(256)      |           |          |         | extended |
 document  | text                        |           |          |         | extended |
 edition   | integer                     |           |          |         | plain    |
 created   | timestamp without time zone |           | not null | now()   | plain    |
 updated   | timestamp without time zone |           | not null | now()   | plain    |
```

Архитектура тестов



Скрипт нагрузки

- JDBC драйвера
  - MongoDB JDBC driver 3.11.1
  - PostgreSQL JDBC driver 42.2.8
  - Redis JDBC driver 3.1.0
- Statement запрос
- Соединение, поиск и чтение отдельно

Профиль нагрузки -PostgreSQL

Транзакция	Распределение, %
Update row values (filter2, filter3, filter4, filter5, edition)	5
Delete row by id and filter1	5
Insert row (id, filter1, filter2, filter3, filter4, filter5, document, edition)	20
Select id and filter4 and filter5 x 2 LIMIT 100 OFFSET 0;	0.7
Select id and filter4 list LIMIT 100 OFFSET 0;	28
Select id and filter5 LIMIT 100 OFFSET 0;	2.1
Select id and filter2	7
Select id and filter2 list LIMIT 100 OFFSET 0;	28
Select id and filter3 lte LIMIT 100 OFFSET 0;	0.7
Select id and filter1	3.5

Профиль нагрузки - MongoDB

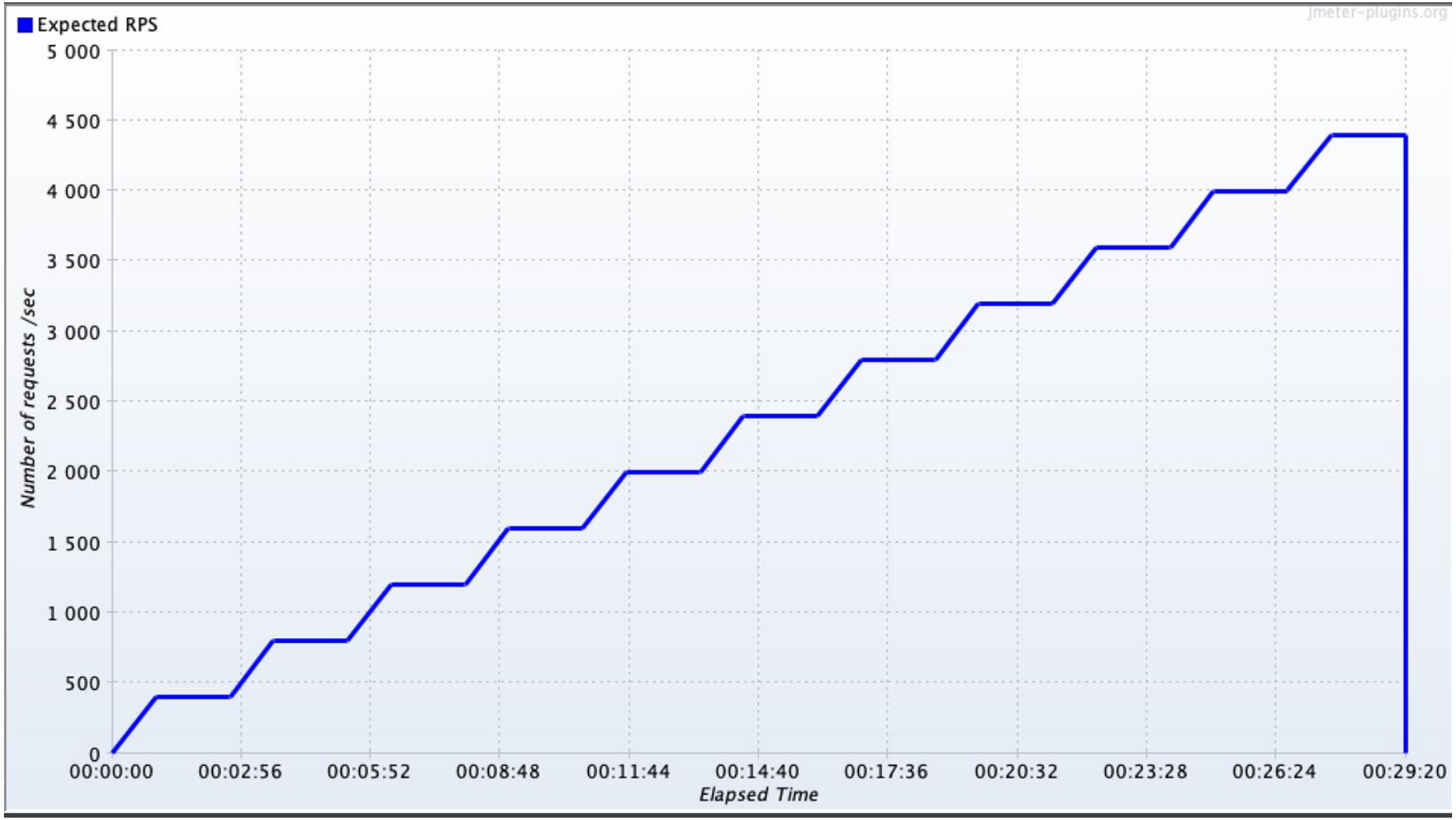
Транзакция	Распределение, %
db.collection.update() (filter2, filter3, filter4, filter5, edition)	5
db.v2storage.db.collection.deleteOne()	5
db.v2storage.insertOne()	20
db.v2storage.find({id: 1, filter4: 'aaa', $or: [{filter5: 'aaa'}, {filter5: 'abv'}]}).skip(0).limit(100)	0.7
db.v2storage.find({id: 1, '$or': [{filter4: 'aaa'}, {filter4: 'aaa_1'}]}).skip(0).limit(100)	28
db.v2storage.find({id: 1, filter5: 'aaa'}).skip(0).limit(100)	2.1
db.v2storage.find({id: 1, filter2: 'aaa'}).first()	7
db.v2storage.find({id: 1, $or: {filter2: 'aaa', filter2: 'aaa_1',..}}).skip(0).limit(100)	28
db.v2storage.find({id: 1, filter3: {$lte : value }}).skip(0).limit(100)	0.7
db.v2storage.find({id: 1, filter1: 'aaa'}).first()	3.5

Taurus скрипт

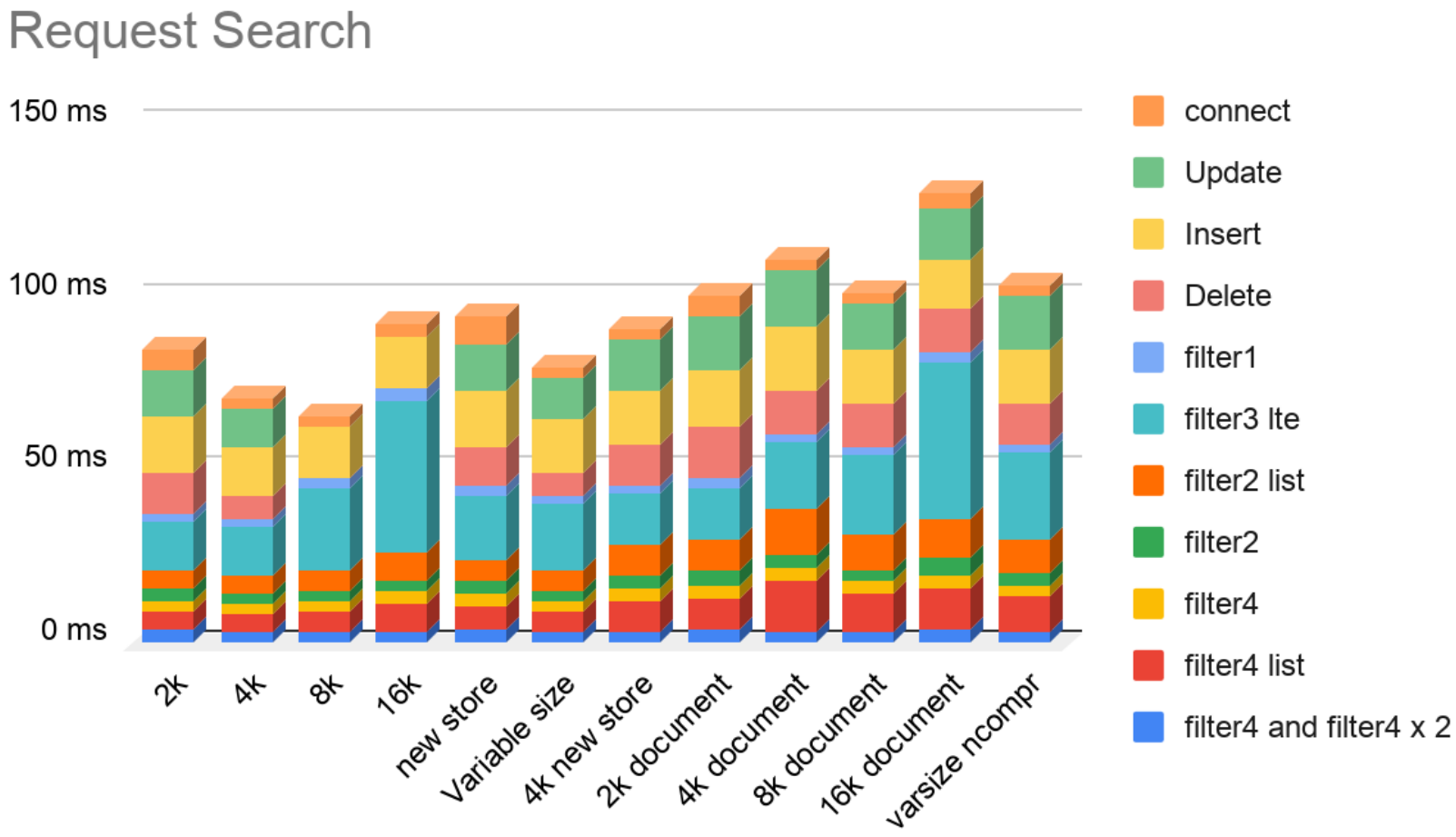
```
execution:
 - concurrency: 5
   distributed:
     - ${jmeter_host}
   hold-for: ${test_time}
   steps: 10
   rump-up: 10m
   throughput: 50
   scenario: Postgres_Thread_id_and_filter1

scenarios:
 Postgres_Thread_insert:
   variables:
     case: "insert"
   script: ./Postgres Insert.jmx
 Postgres_Thread_delete:
   variables:
     case: "delete"
     select_field_2: "filter1"
   script: ./Postgres Delete.jmx
```

Taurus

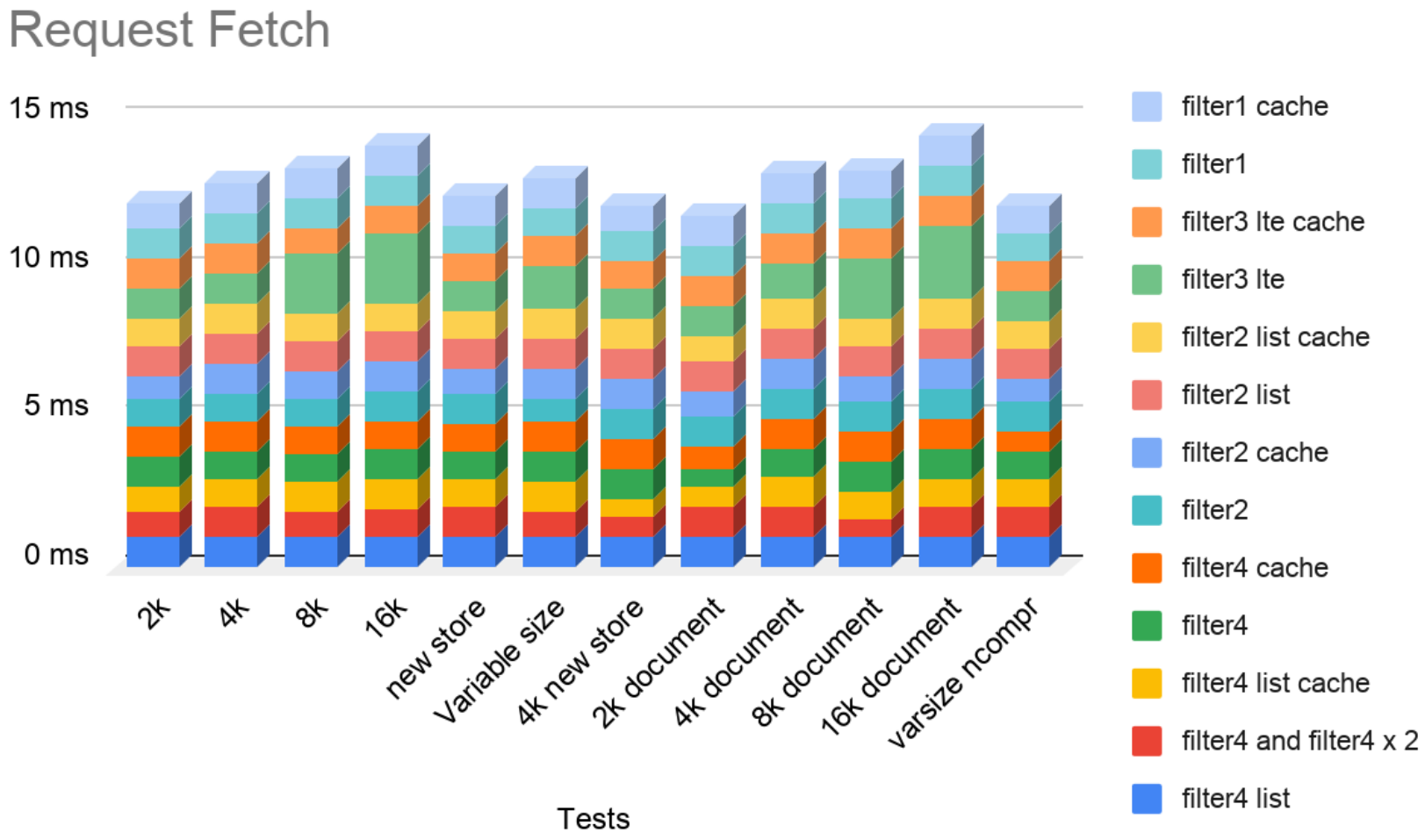
Сценарий Max Capacity



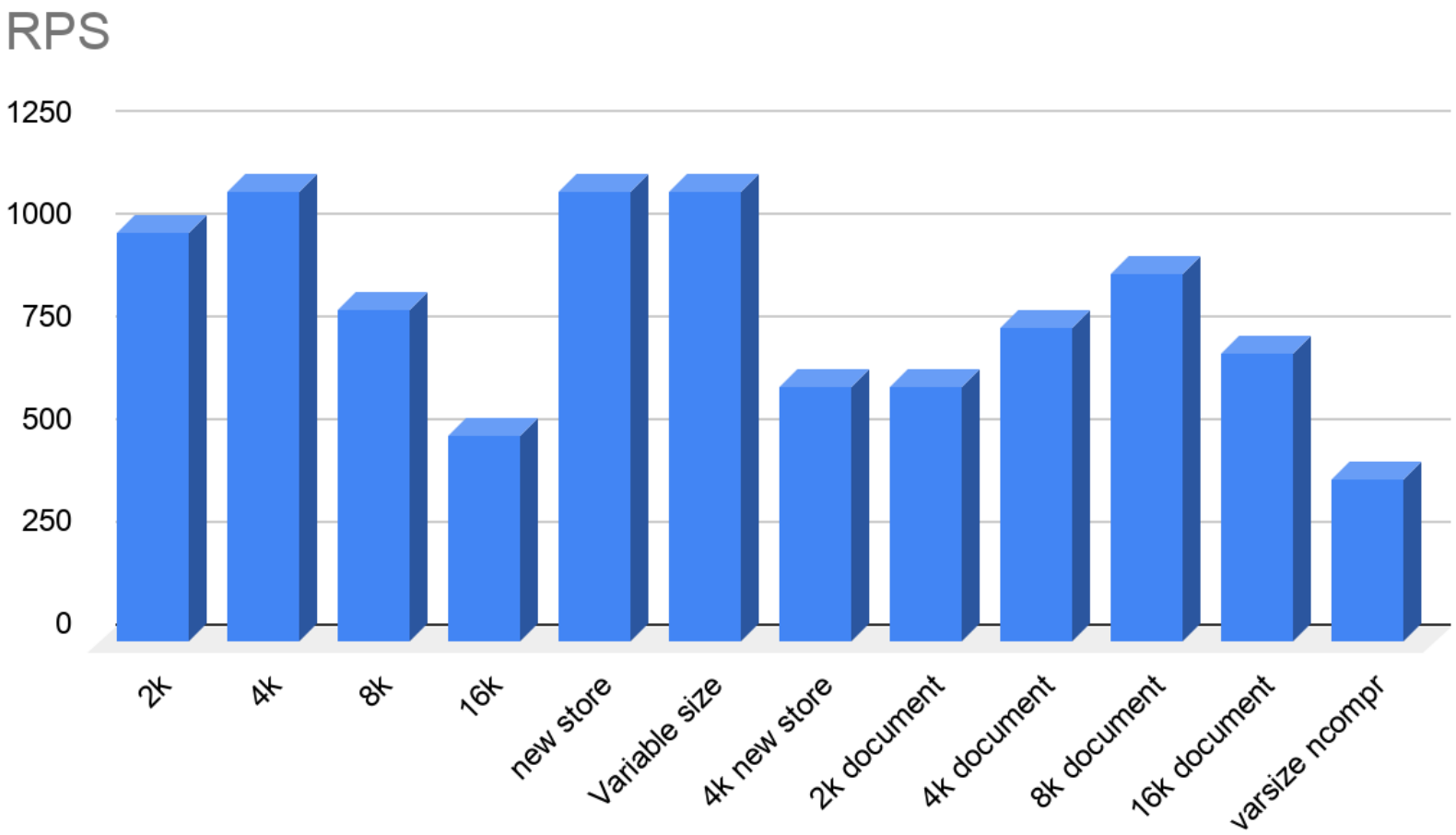
Результаты - Postgres



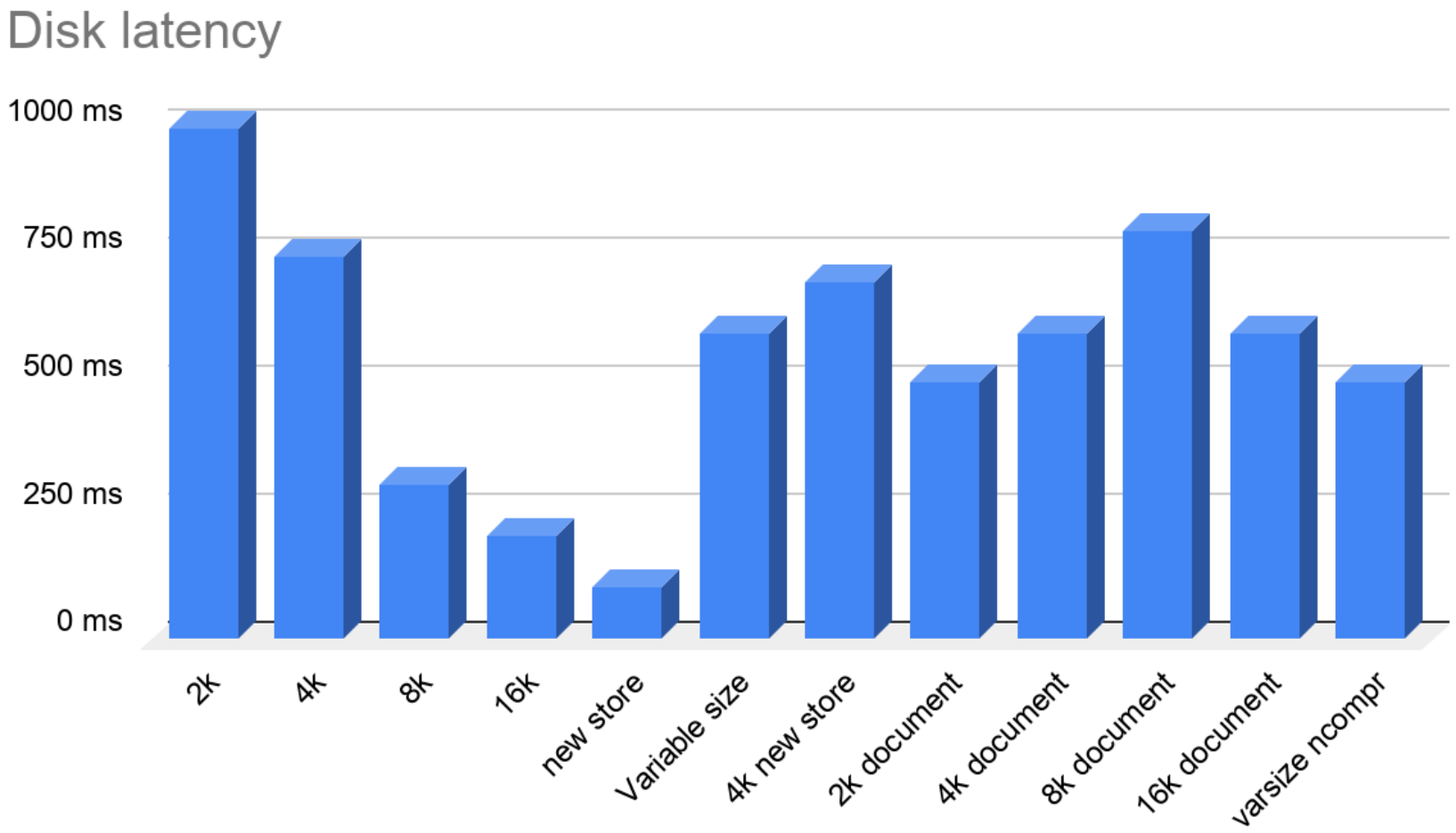
Результаты


Request Fetch
15 ms
10 ms
5 ms
0 ms
2k
4k
8k
16k
new store
Variable size
4k new store
2k document
4k document
8k document
16k document
varsize ncompr
Tests
filter1 cache
filter1
filter3 lte cache
filter3 lte
filter2 list cache
filter2 list
filter2 cache
filter2
filter4 cache
filter4
filter4 list cache
filter4 and filter4 x 2
filter4 list

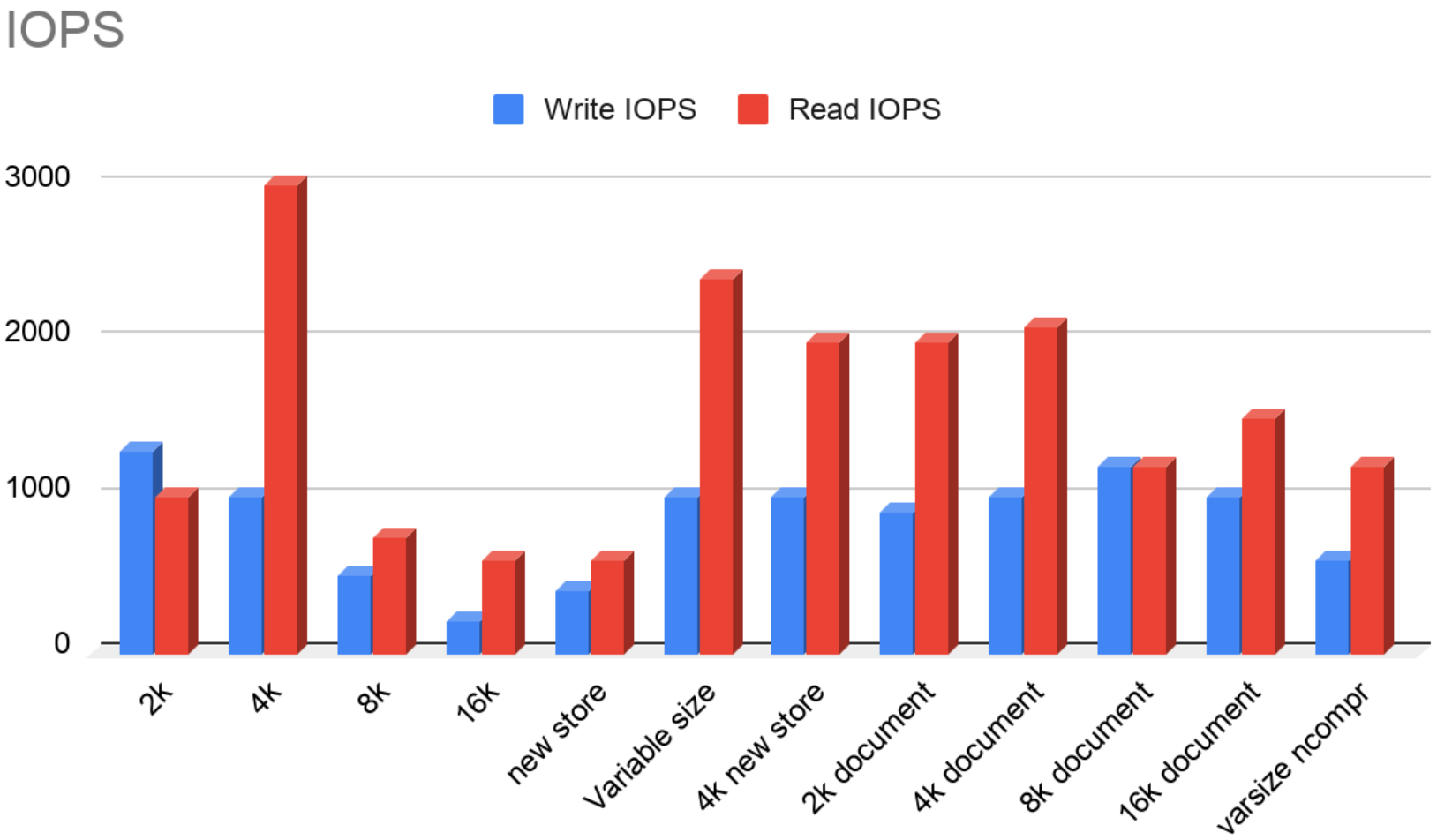
Результаты


RPS
1250
1000
750
500
250
0
2k
4k
8k
16k
new store
Variable size
4k new store
2k document
4k document
8k document
16k document
varsize ncompr

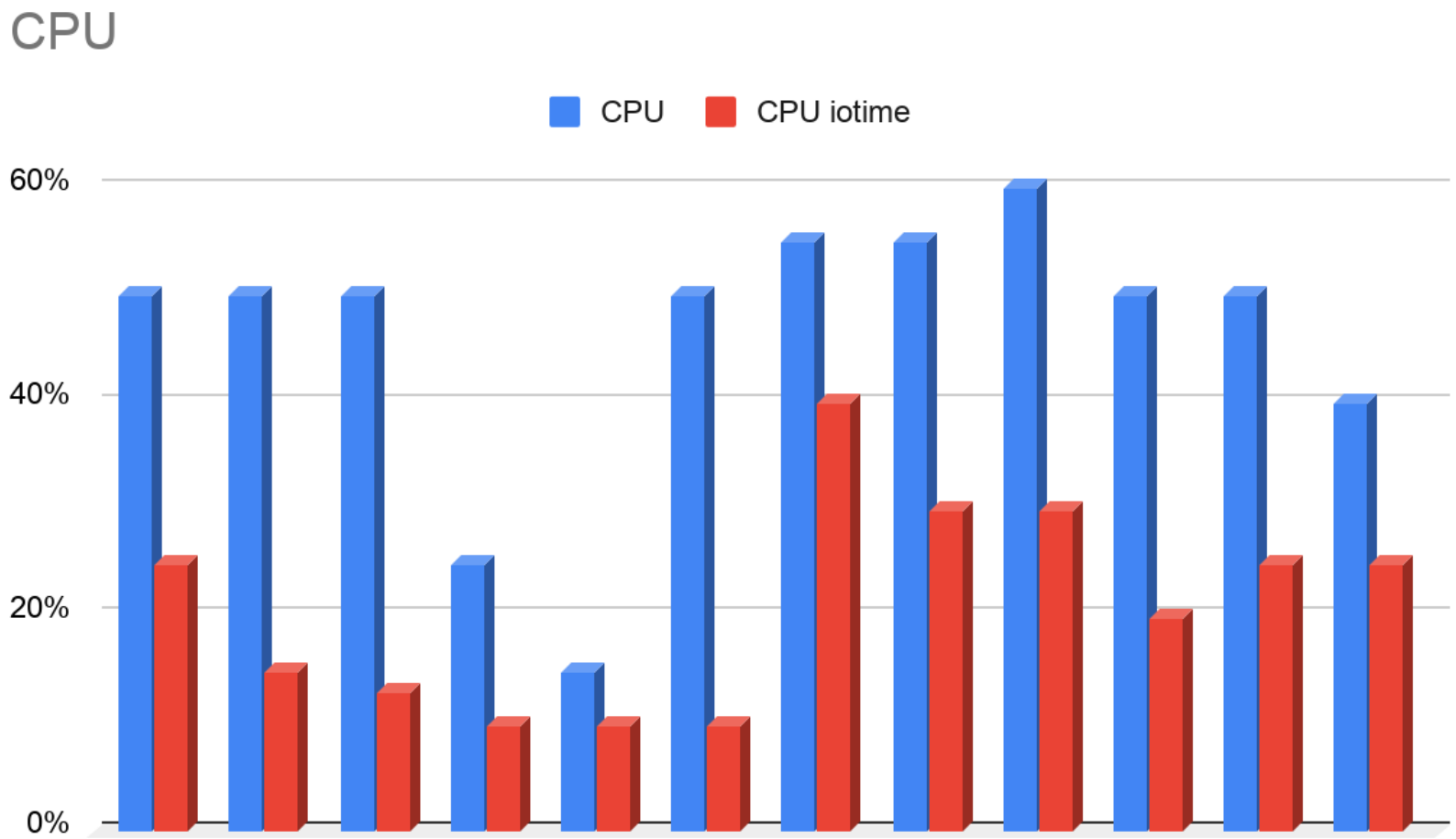
Результаты


Disk latency
1000 ms
750 ms
500 ms
250 ms
0 ms
2k
4k
8k
16k
new store
Variable size
4k new store
2k document
4k document
8k document
16k document
varsize ncompr

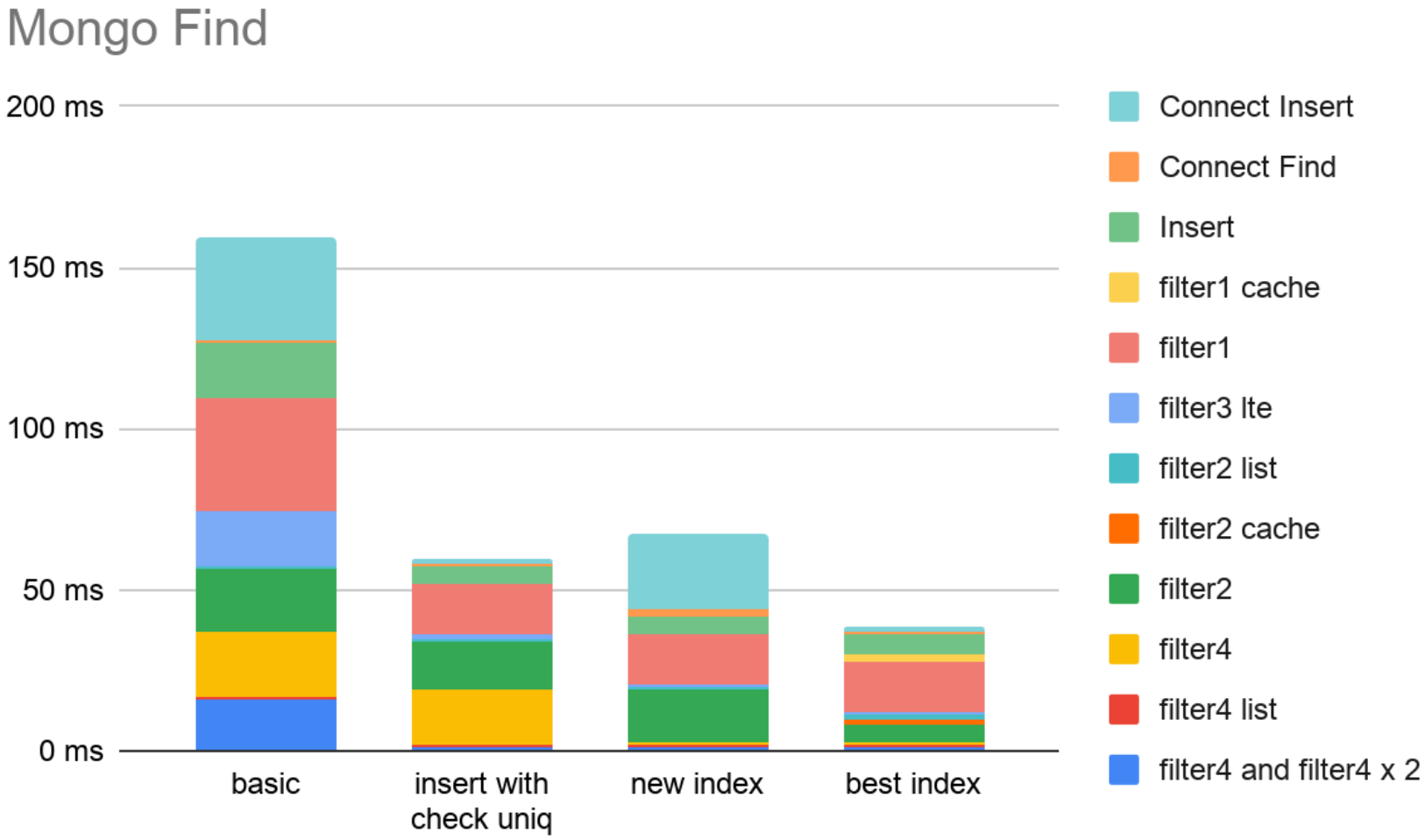
Результаты


IOPS
Write IOPS
Read IOPS
3000
2000
1000
0
2k
4k
8k
16k
new store
Variable size
4k new store
2k document
4k document
8k document
16k document
varsize ncompr

Результаты

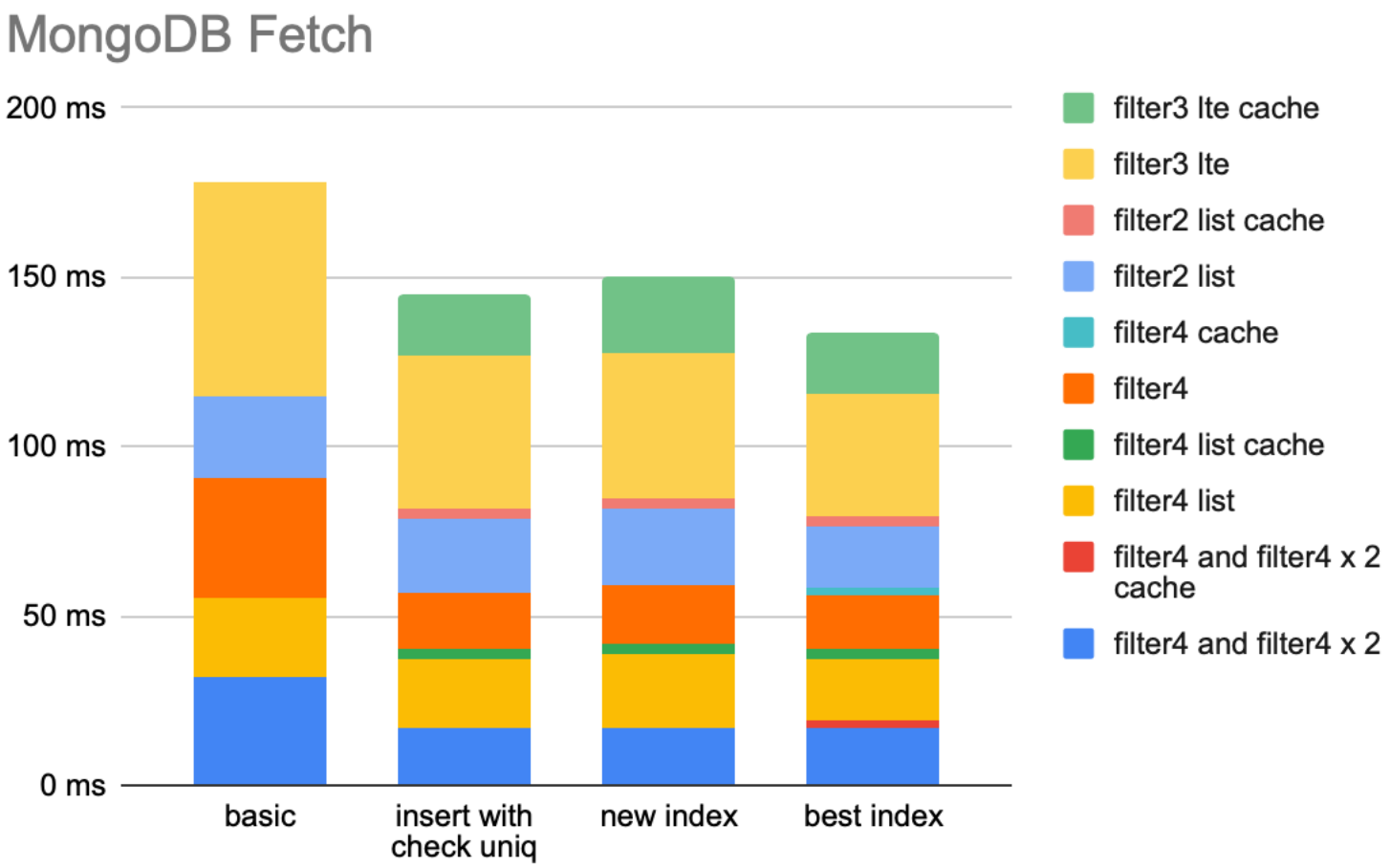


Результаты - MongoDB

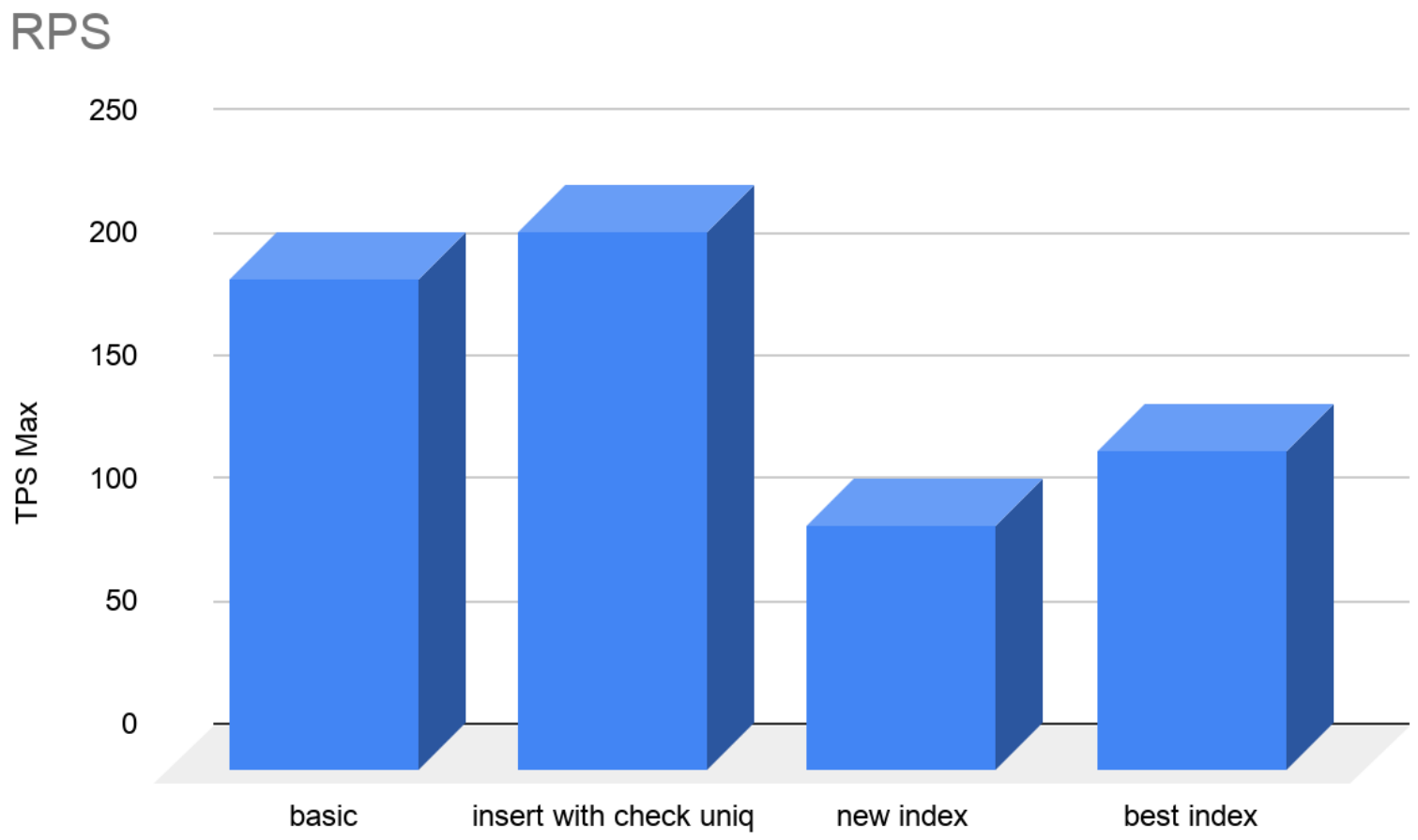


mongoDB

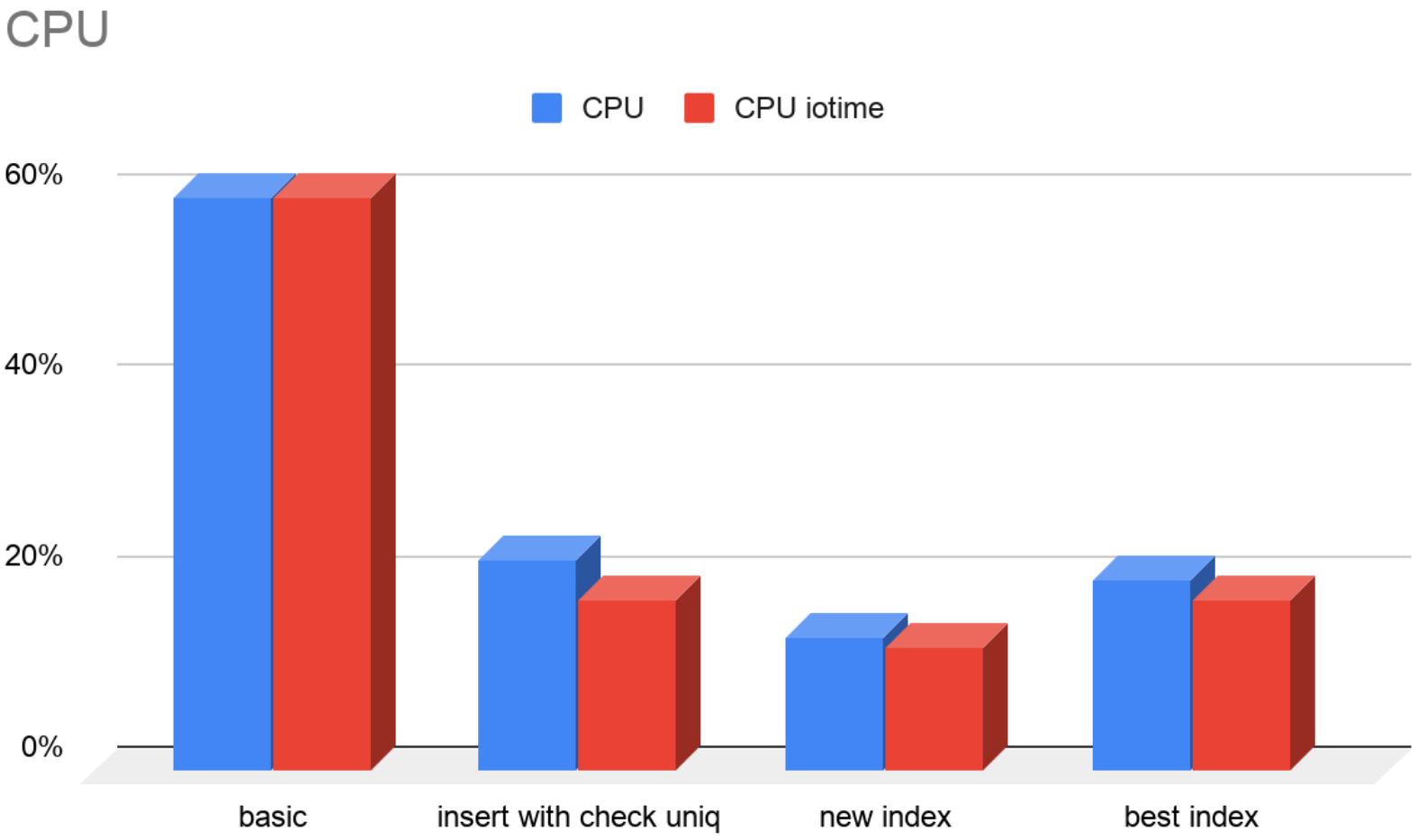
Результаты


MongoDB Fetch
200 ms
150 ms
100 ms
50 ms
0 ms
basic
insert with check uniq
new index
best index
filter3 lte cache
filter3 lte
filter2 list cache
filter2 list
filter4 cache
filter4
filter4 list cache
filter4 list
filter4 and filter4 x 2 cache
filter4 and filter4 x 2

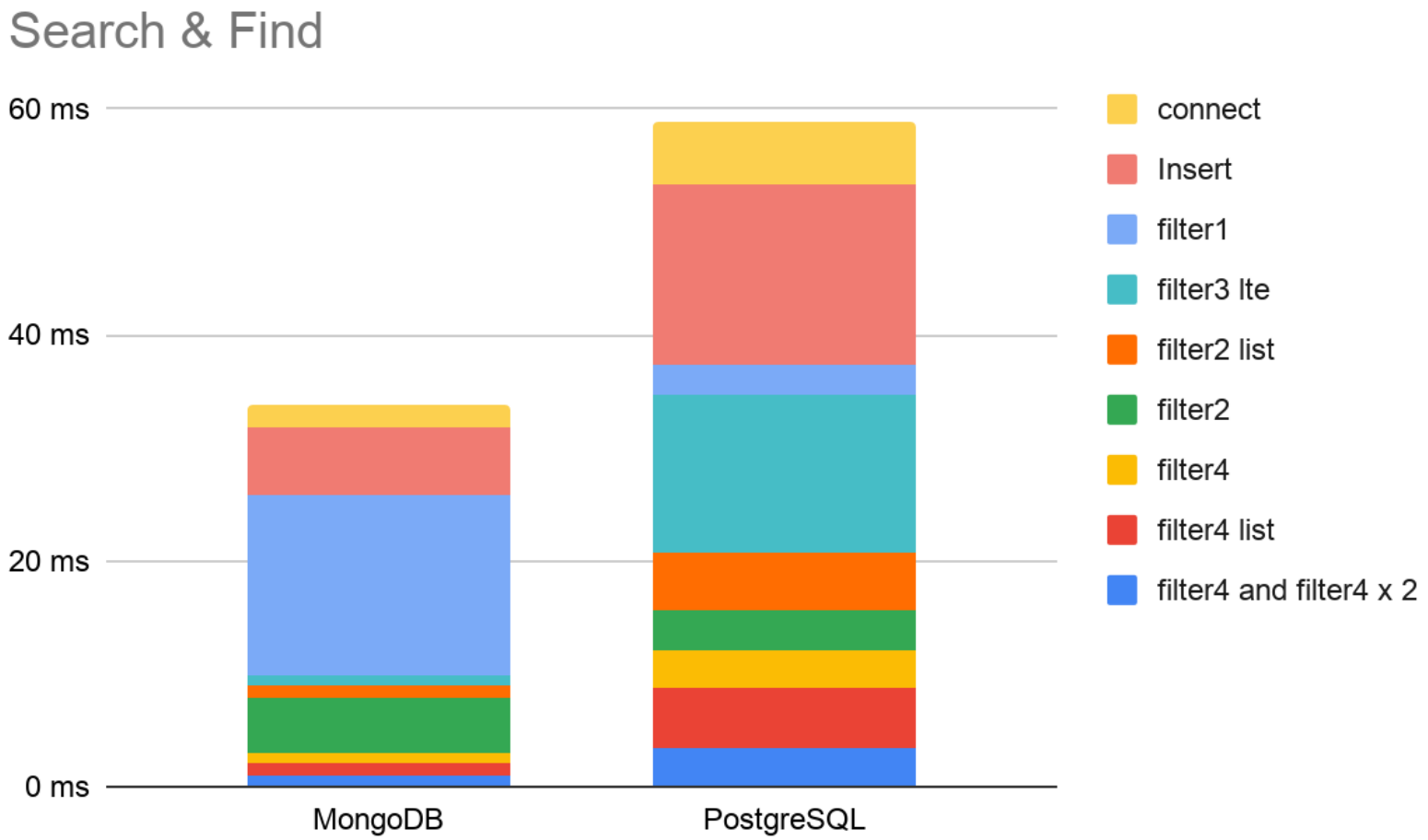
Результаты


RPS
250
200
150
100
50
0
TPS Max
basic
insert with check uniq
new index
best index

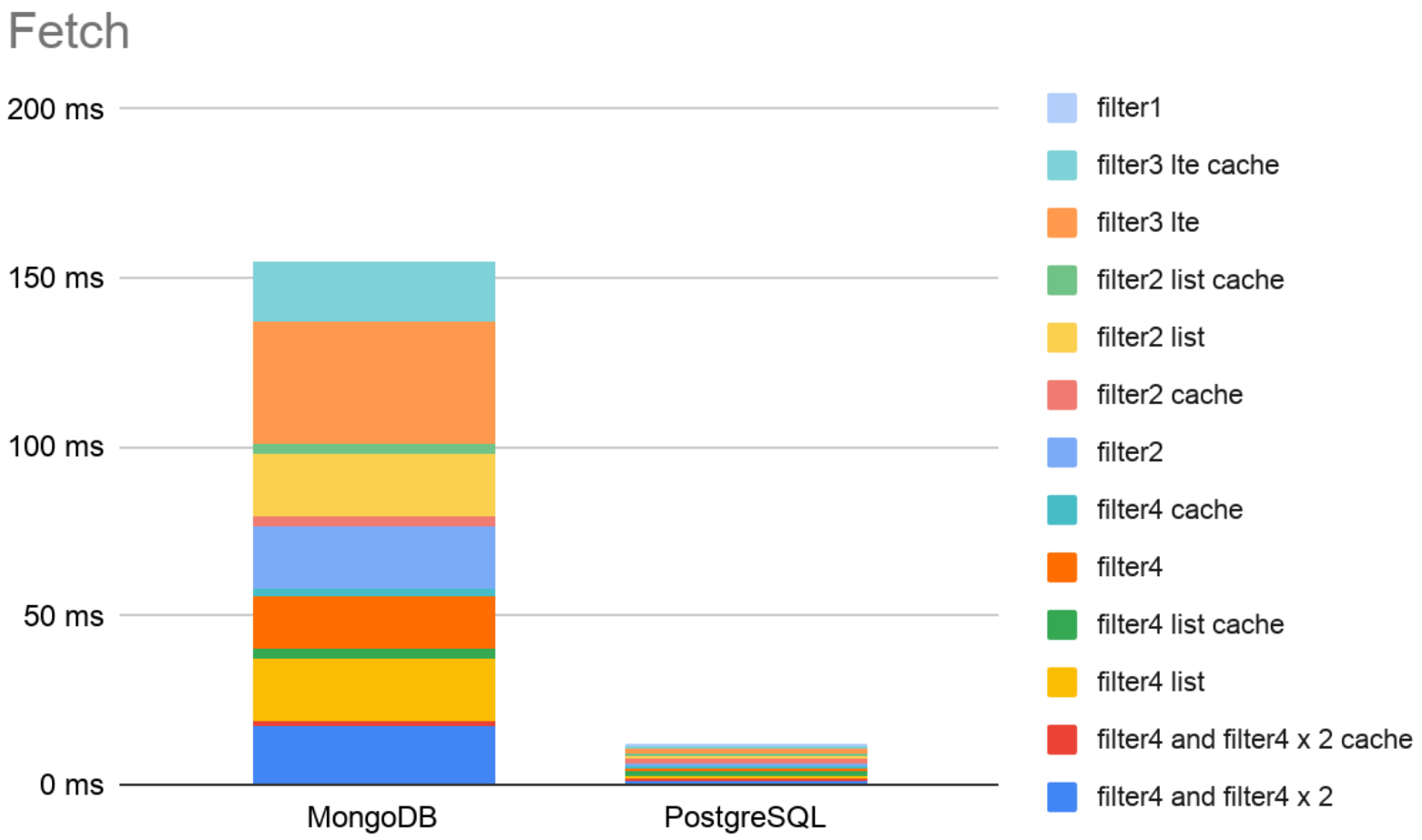
Результаты


CPU
CPU
CPU iotime
60%
40%
20%
0%
basic
insert with check uniq
new index
best index

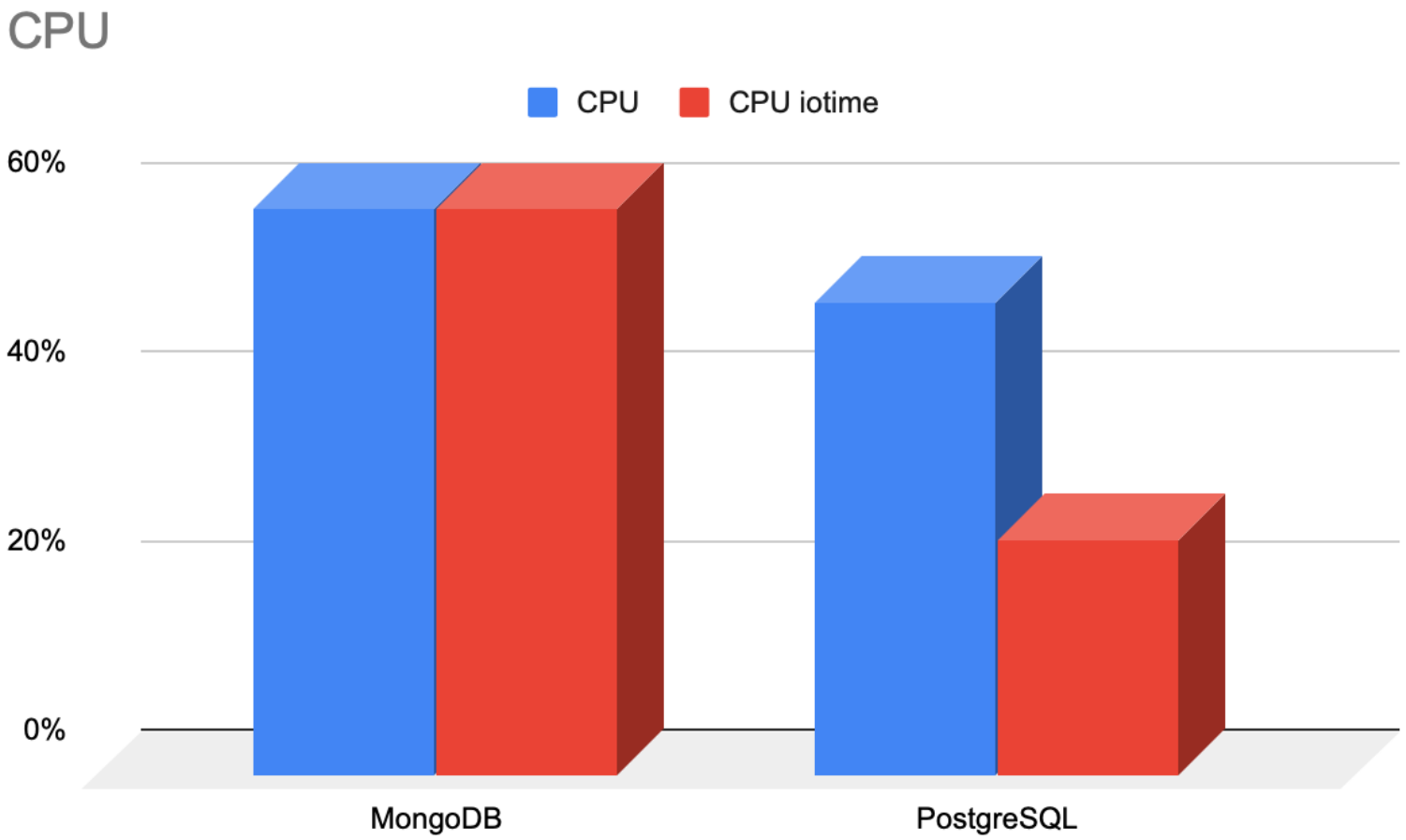
MongoDB vs Citus/PostgreSQL



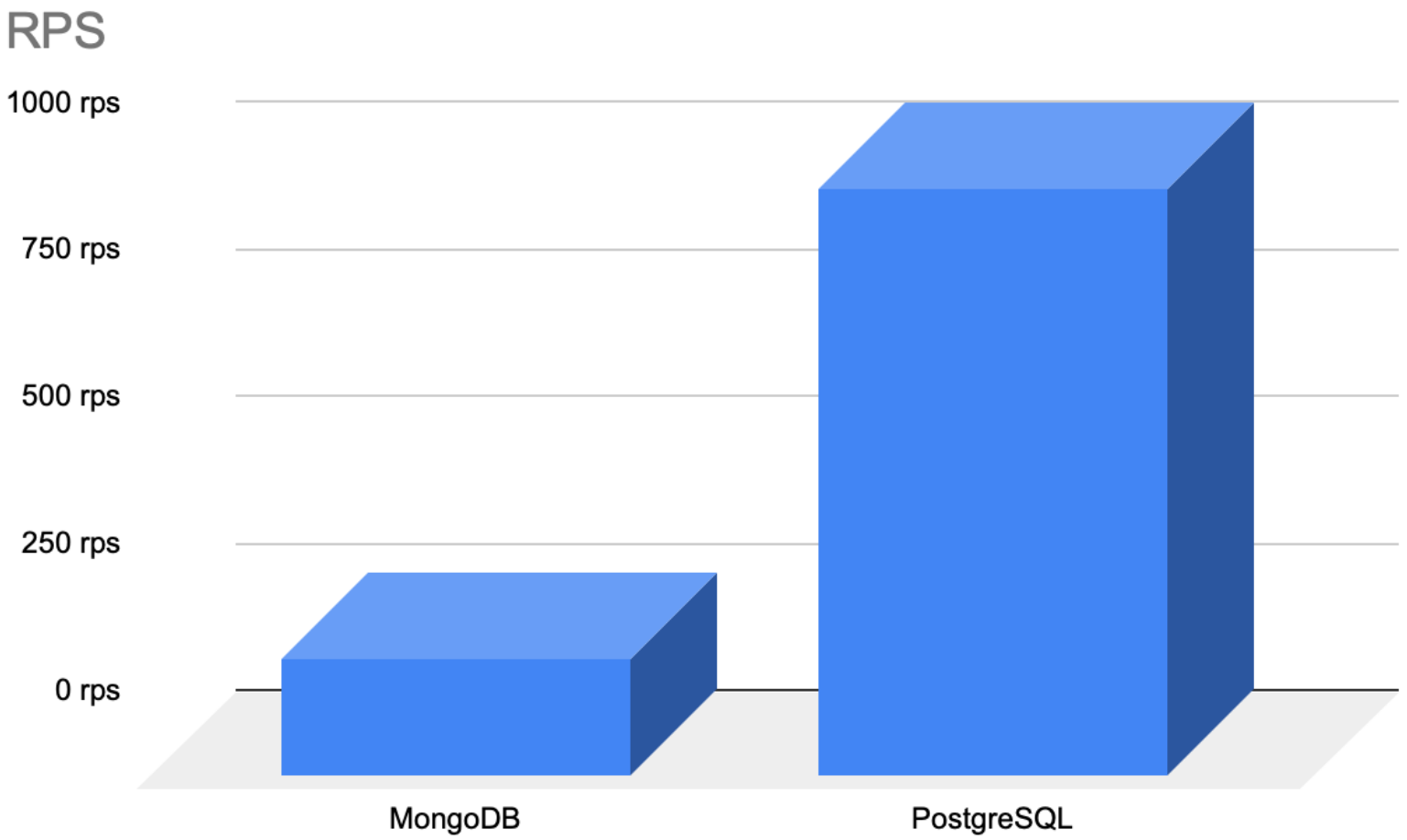
MongoDB vs Citus/PostgreSQL



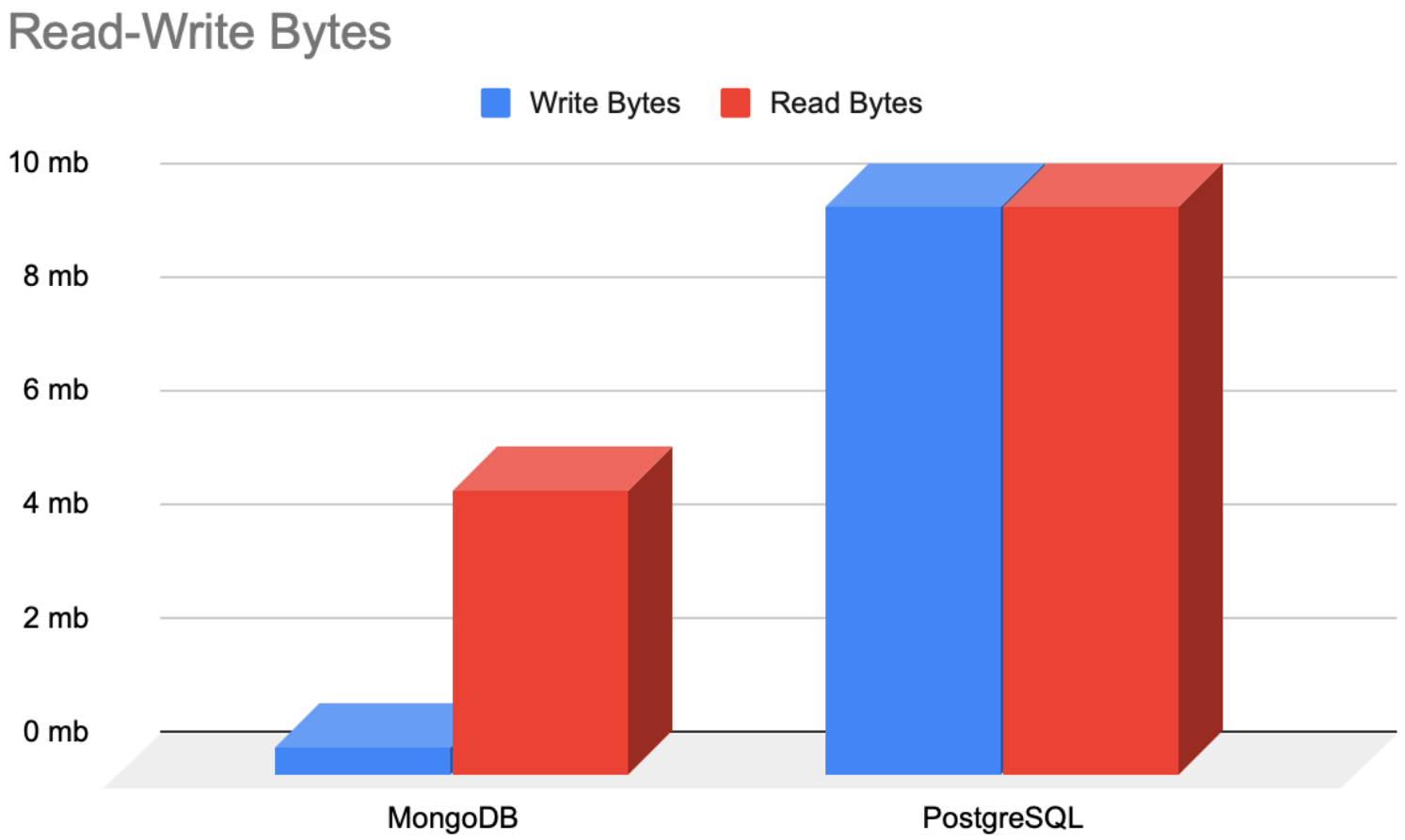
MongoDB vs Citus/PostgreSQL



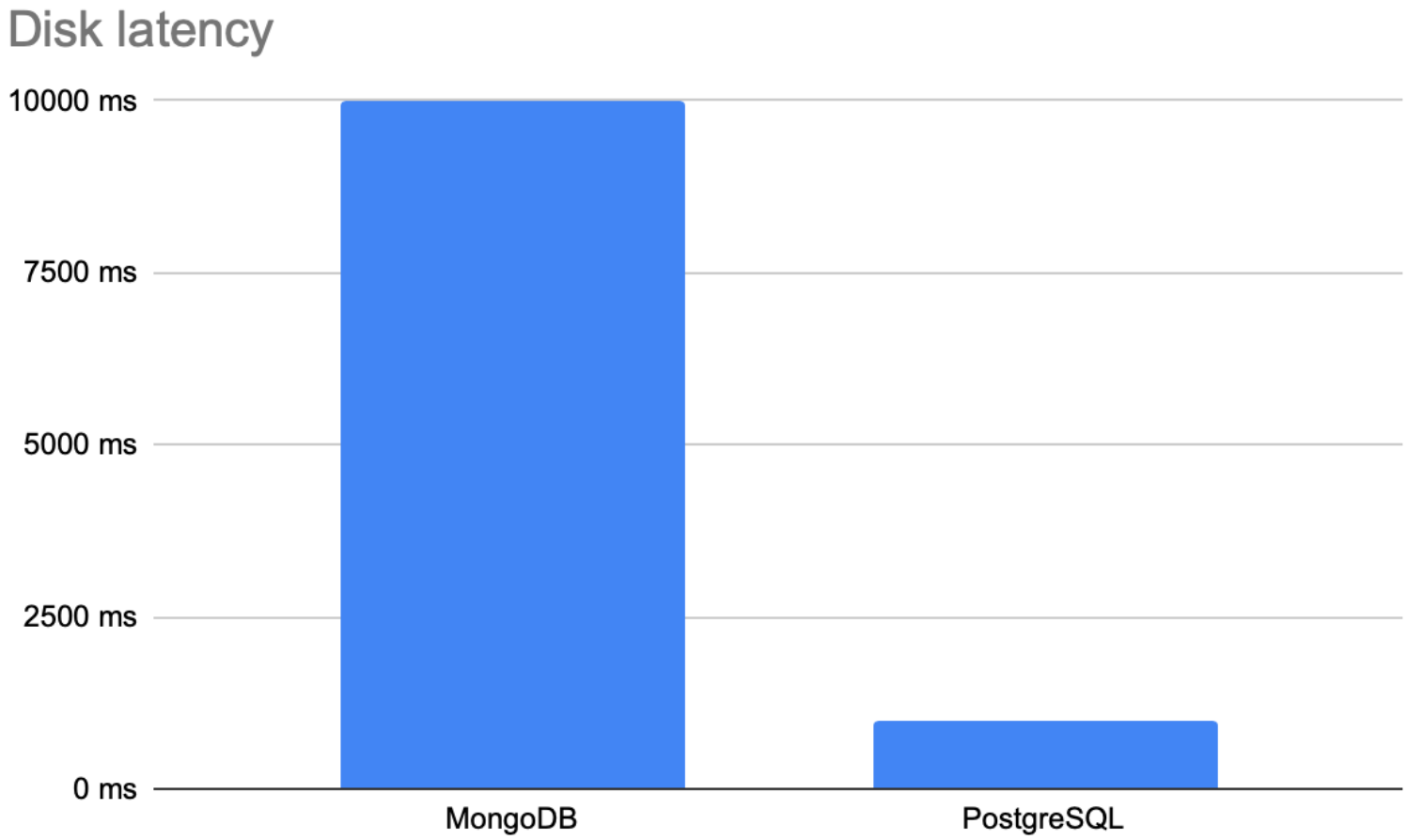
MongoDB vs Citus/PostgreSQL



MongoDB vs Citus/PostgreSQL



MongoDB vs Citus/PostgreSQL



MongoDB vs Citus/PostgreSQL

Итоги

- Оба продукты хороши
- Но обладают особенностями
  - PostgreSQL более предсказуем и требует меньше инфраструктурных серверов (Config сервер и replica sets).
  - Минимальное развертывание MongoDB в разы (~3) дороже.
  - Горизонтальное масштабирование PostgreSQL реализовать сложно, подходящего продукта для этого пока нет.

Спасибо за внимание!

Вопросы?

Наши контакты

kirill.kalistratov@incountry.com

vasiliy.soshnikov@gmail.com

spirin.alexander@incountry.com