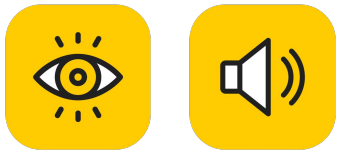
OTUS

PostgreSQL для администраторов баз данных и разработчиков

otus.ru

Меня хорошо видно & слышно?



Защита проекта
Тема: Сравнение производительности PostgreSQL в различных архитектурных решениях

Ряполов Ярослав

Разработчик в компании Loymax Solutions

План защиты

Цели проекта	Сравнить производительность различных конфигураций PostgreSQL
Что планировалось	Сравнить много конфигураций PostgreSQL
Используемые технологии	pgbench, citus, pgbouncer, Yandex Cloud
Что получилось	Далеко не всё :(
Немного деталей	...
Выводы	

Цели проекта

1. На практике смоделировать ситуации недостатка различных ресурсов при нагрузке на кластер БД PostgreSQL
2. Сравнить работу различных конфигураций кластера БД PostgreSQL под нагрузкой

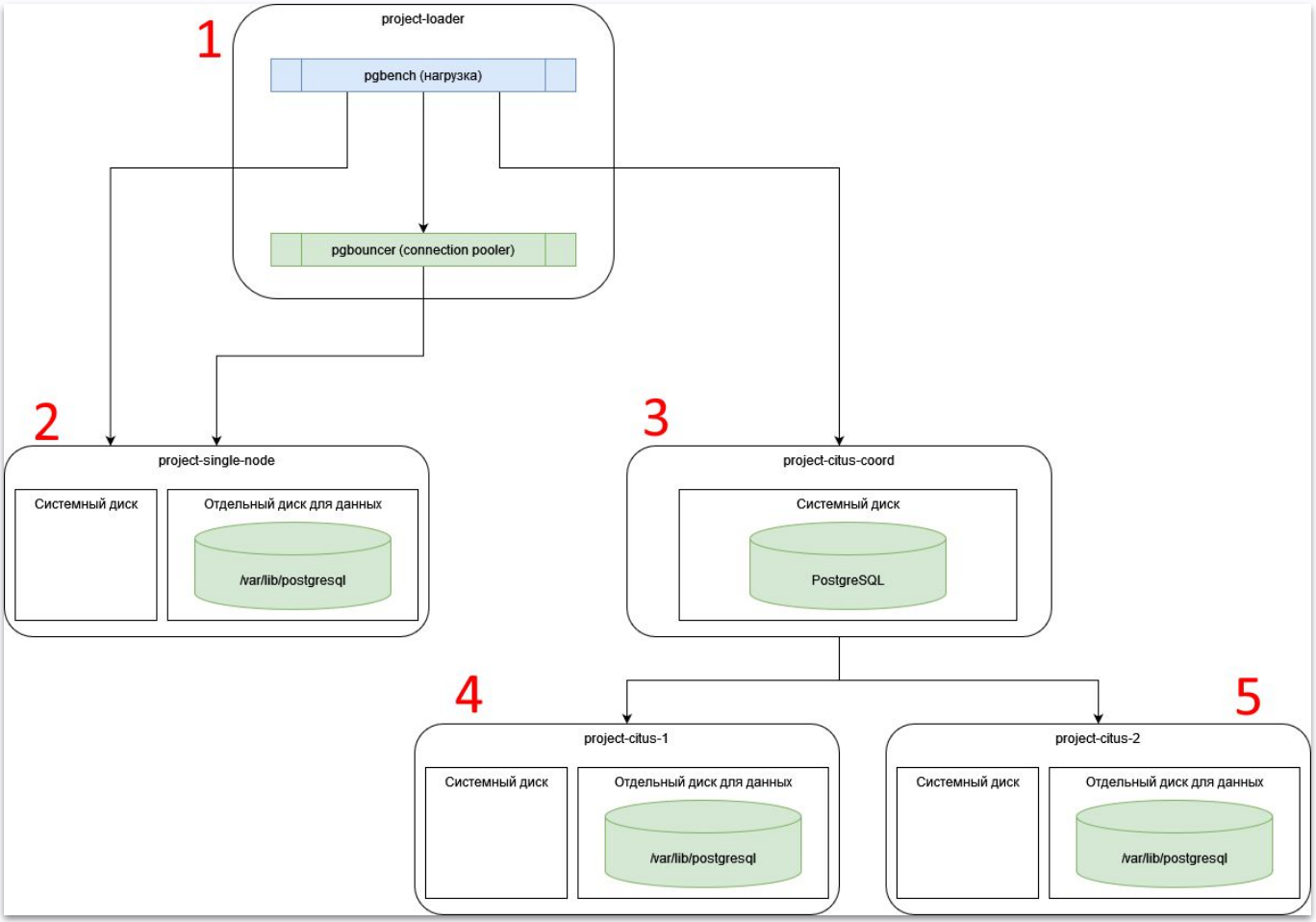
Что планировалось (слишком многое…)

1. Выбрать, подготовить и развернуть конфигурации PostgreSQL для тестирования
2. Настроить расширенный мониторинг для развернутых инструментов
3. Использовать для тестирования утилиты sysbench и pgbench и, возможно, сравнить их возможности
4. Попробовать разные нагрузки TCP-B и TCP-C и провести сравнение настроенных конфигураций на их основе на их основе
5. Спроектировать свой сценарий нагрузки

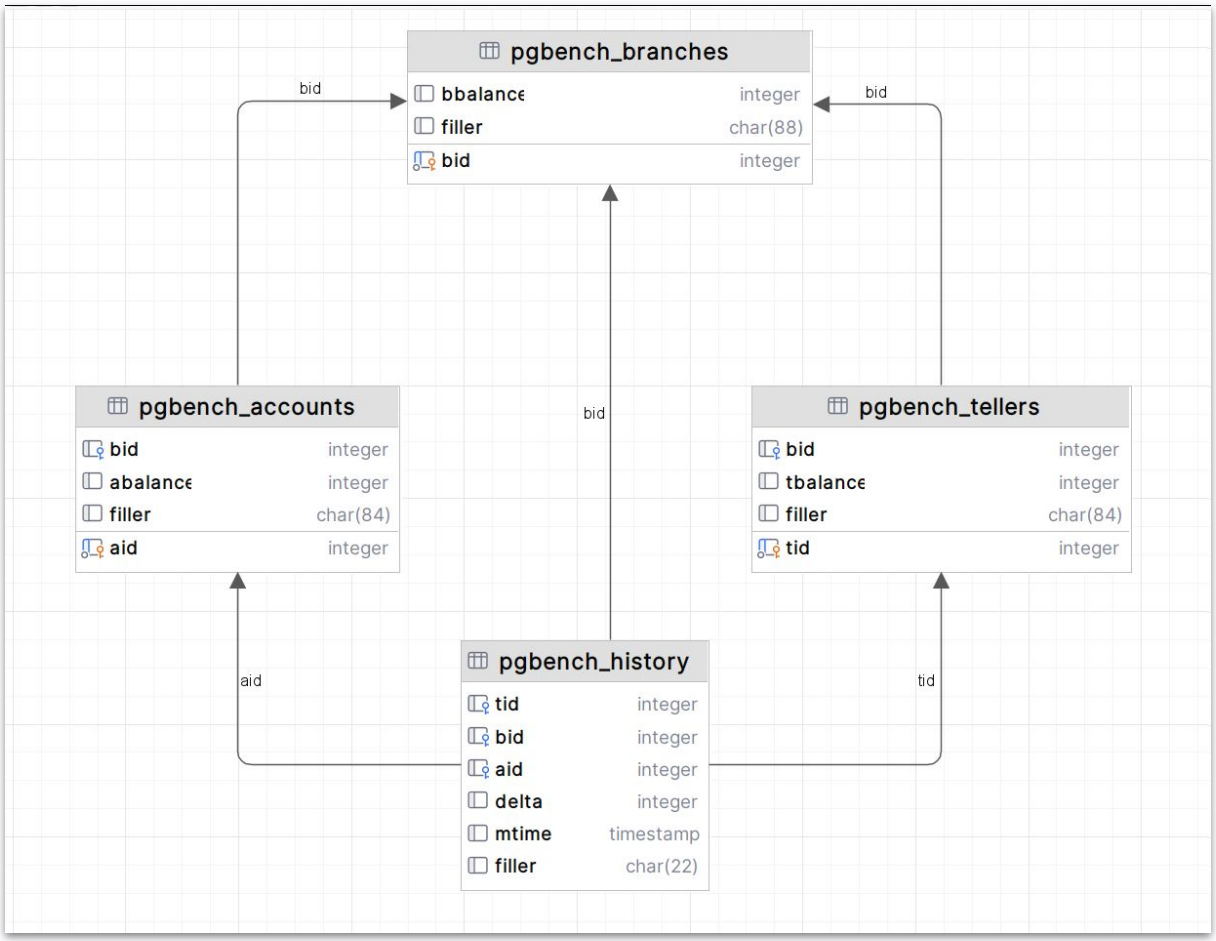
Используемые технологии

1. PostgreSQL **15**
2. Yandex Cloud (https://yandex.cloud)
3. pgbench (https://www.postgresql.org/docs/current/pgbench.html)
4. Citus (https://www.citusdata.com/) под PostgreSQL **15**
5. pgbouncer (https://www.pgbouncer.org/)
6. Для тюнинга настроек узлов использовал https://www.pgconfig.org

Конфигурации



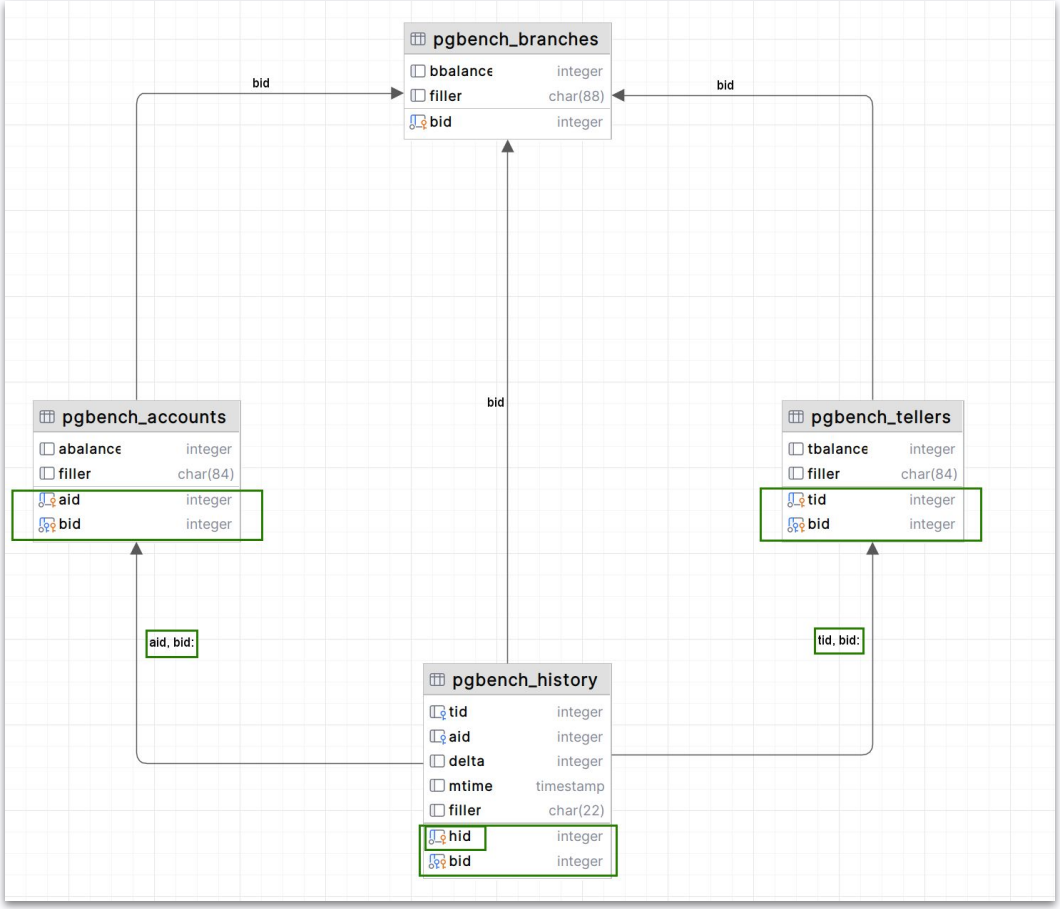
pgbench TPC-B (sort of) схема БД



Немного более детальная информация о типах нагрузкочного тестирования БД:
https://www.ibm.com/docs/en/informix-servers/14.10?topic=throughput-standard-benchmarks

pgbench TPC-B (sort of) схема БД

адаптированная для шардирования на citus по подразделениям (pgbench_branches)



pgbench и его TPC-B (sort of) сценарий нагрузочного тестирования

Исходный скрипт pgbench

```
\set aid random(1, 100000 * :scale)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
END;
```

Правки для citus

```
\set bid random(1, 1 * :scale)
\set aid ((:bid - 1) * 100000 + random(1, 100000))
\set tid random(10 * :bid - 9, 10 * :bid)
\set delta random(-5000, 5000)
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid AND bid = :bid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid AND bid = :bid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid AND bid = :bid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
END;
```

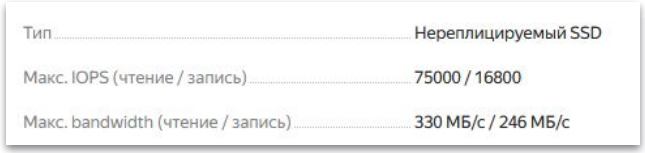
Citus

Что важно помнить при настройке

1. Это расширение, потому явно подключаем и задаем координатора с воркерами отдельно для каждой БД, которую планируем шардировать
2. distribution column должен быть составной частью уникального или первичного ключа
3. Важно не забывать включать в запросы условия для фильтрации по distribution column, иначе производительность может значительно просесть из-за распределенных запросов и особенно распределенных транзакций (но в эту тему не углублялся)

I. Узкое место - процессор

Быстрый SSD



Тип	Нереплицируемый SSD
Макс. IOPS (чтение / запись)	75000 / 16800
Макс. bandwidth (чтение / запись)	330 МБ/с / 246 МБ/с

2 vCPU и 4ГБ RAM

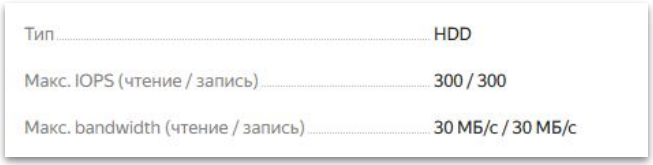
	24 | tps	cpu	i/o	64 | tps	cpu	i/o
1kkk single	1 298	~100%	3k/3k	1 415	~100%	3k/3k
1kkk citus		80%	25		90-100%	30
worker 1	825	40%	1.5k/1.5k	767	40%	1k/1k
worker 2		40%	1.5k/1.5k		40%	1k/1k

4 vCPU и 4ГБ RAM

	96 | tps	cpu	i/o	96 | readonly tps	cpu	i/o
1kkk single	3 405	~100%	7k/7k	13 826	~100%	0/14k
1kkk citus		100%	25		100%	5
worker 1	1 885	80%	2.5k/2.5k	4 306	80%	0k/4k
worker 2		80%	2.5k/2.5k		80%	1k/4k

II. Узкое место - диск

Сетевой HDD



Дополнительная информация по квотам на скорость работы HDD доступна по ссылке https://yandex.cloud/ru/docs/compute/concepts/storage-read-write

2 vCPU и 4ГБ RAM

	32 | tps	cpu	i/o	64 | tps	cpu	i/o	72 | tps	cpu	i/o
100kk single	324	30%	250/400	334	35%	500/600	357	40%	400/600
100kk citus		25%	0/10		60%	0/20		55%	0/20
worker 1	258	10%	200/200	546	25%	300/450	471	25%	350/350
worker 2		10%	150/200		25%	300/450		25%	400/400

III. Зависимость нагрузки от размера БД

Быстрый SSD

Тип	Нереплицируемый SSD
Макс. IOPS (чтение / запись)	75000 / 16800
Макс. bandwidth (чтение / запись)	330 МБ/с / 246 МБ/с

Зависимость нагрузки от размера (8 vCPU 16ГБ RAM)

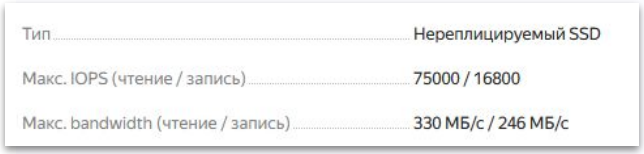
	24 | tps	cpu	i/o	64 | tps	cpu	i/o	96 | tps	cpu	i/o
1kkk single	3 401	80%	7k/7k	4 652	80%	8.5k/8.5k	4 746	80%	8.5k/8.5k
1kk single	3 471	70%	0/2k	3 670	80%	0/2k	3 794	85%	0/2k

```
SELECT COUNT(*)
FROM pg_locks pl LEFT JOIN pg_stat_activity psa ON pl.pid = psa.pid;
```

```
\set aid random(1, 100000 * :scale)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
END;
```

IV. С какого объема нагрузки полезен pgbouncer?

Быстрый SSD



Тип	Нереплицируемый SSD
Макс. IOPS (чтение / запись)	75000 / 16800
Макс. bandwidth (чтение / запись)	330 МБ/с / 246 МБ/с

2 vCPU и 4ГБ RAM

		16 | tps	24 | tps	64 | tps	96 | tps
прямое подключение	1kkk single	1 379	**1 366**	1 320	1 279
	1kk single	1 531	**1 685**	1 477	1 271
pgbouncer на 24 подключения	1kkk single				**1 424**
	1kk single				**1 649**

Выводы

1. Детальный мониторинг может помочь быстрее разобраться в проблеме и принять правильные решения по настройке параметров СУБД и подсказать пути решения проблемы
2. Польза от использования connection pooler видна на самых небольших нагрузках

Планы по развитию

1. Проверить поможет ли секционирование по разным тейблспейсам обойти ограничение по диску (не будем ли упираться в диск для wal сильно быстро)
2. Попробовать нагрузку с TCP-C с более сложными запросами на процессор при сравнении с citus
3. Настроить для тестирования более детальный мониторинг (с информацией о внутренних операциях СУБД, по типу блокировок)
4. Сравнение PostgreSQL с MS SQL на сопоставимых нагрузках

Спасибо за внимание!