Домашнее задание

Работа с журналами

1. Настройте выполнение контрольной точки раз в 30 секунд.

Изначально по умолчанию было 5 минут:

postgres=# show checkpoint\_timeout;

checkpoint\_timeout

--------------------

5min

(1 row)

Меняем и перегружаем кластер:

postgres=# ALTER SYSTEM SET checkpoint\_timeout = 30;

ALTER SYSTEM

postgres=# show checkpoint\_timeout;

checkpoint\_timeout

--------------------

30s

(1 row)

2. 10 минут c помощью утилиты pgbench подавайте нагрузку.

pgbench -P 1 -T 600 buffer\_temp

progress: 599.0 s, 575.0 tps, lat 1.740 ms stddev 0.203

progress: 600.0 s, 559.0 tps, lat 1.787 ms stddev 0.231

transaction type: <builtin: TPC-B (sort of)>

scaling factor: 1

query mode: simple

number of clients: 1

number of threads: 1

duration: 600 s

number of transactions actually processed: 346300

latency average = 1.732 ms

latency stddev = 0.357 ms

initial connection time = 4.449 ms

tps = 577.169738 (without initial connection time)

3. Измерьте, какой объем журнальных файлов был сгенерирован за это время. Оцените, какой объем приходится в среднем на одну контрольную точку.

Текущее колличество жулнальных файлов:

buffer\_temp=# SELECT \* FROM pg\_ls\_waldir() LIMIT 10;

name | size | modification

--------------------------+----------+------------------------

000000010000000000000001 | 16777216 | 2022-04-01 05:33:09+00

(1 row)

После запуска утилиты:

buffer\_temp=# SELECT \* FROM pg\_ls\_waldir() LIMIT 10;

name | size | modification

--------------------------+----------+------------------------

000000010000000000000020 | 16777216 | 2022-04-01 05:50:12+00

000000010000000000000022 | 16777216 | 2022-04-01 05:48:10+00

000000010000000000000021 | 16777216 | 2022-04-01 05:47:42+00

000000010000000000000023 | 16777216 | 2022-04-01 05:48:39+00

Физический размер файла до:

postgres@postgres25032022:/home/anadyrov$ ls -lart /var/lib/postgresql/14/main/pg\_wal/

total 16396

drwx------ 2 postgres postgres 4096 Apr 1 05:29 archive\_status

-rw------- 1 postgres postgres 16777216 Apr 1 05:33 000000010000000000000001

После:

postgres@postgres25032022:/home/anadyrov$ du -sch /var/lib/postgresql/14/main/pg\_wal/\*

16M /var/lib/postgresql/14/main/pg\_wal/000000010000000000000020

16M /var/lib/postgresql/14/main/pg\_wal/000000010000000000000021

16M /var/lib/postgresql/14/main/pg\_wal/000000010000000000000022

16M /var/lib/postgresql/14/main/pg\_wal/000000010000000000000023

4.0K /var/lib/postgresql/14/main/pg\_wal/archive\_status

65M total

Объем приходится в среднем на одну контрольную точку - 16M

Объем журнальных файлов был сгенерирован - 65M

4. Проверьте данные статистики: все ли контрольные точки выполнялись точно по расписанию. Почему так произошло?

Контрольная точка выполняется через каждые checkpoint\_segments сегментов журнала или каждые checkpoint\_timeout секунд, в зависимости от того, какое событие наступит первым. Если с предыдущей контрольной точки ни один файл WAL так и не был записан, то новые контрольные точки будут пропущены, даже если наступил checkpoint\_timeout.

Управлять частотой смены файлов журнала для ограничения потенциальной потери данных, то нужно корректировать параметр archive\_timeout, а не параметры контрольной точки.

Либо принудительный запуск CHECKPOINT

5. Сравните tps в синхронном/асинхронном режиме утилитой pgbench. Объясните полученный результат.

-- Запускаем в синхронном режиме

postgres@postgres25032022:/home/anadyrov$ psql

psql (14.2 (Ubuntu 14.2-1.pgdg18.04+1))

Type "help" for help.

postgres=# show synchronous\_commit;

synchronous\_commit

--------------------

on

(1 row)

postgres=# \q

postgres@postgres25032022:/home/anadyrov$ pgbench -i buffer\_temp

dropping old tables...

creating tables...

generating data (client-side)...

100000 of 100000 tuples (100%) done (elapsed 0.10 s, remaining 0.00 s)

vacuuming...

creating primary keys...

done in 0.52 s (drop tables 0.05 s, create tables 0.03 s, client-side generate 0.24 s, vacuum 0.11 s, primary keys 0.09 s).

postgres@postgres25032022:/home/anadyrov$ pgbench -P 1 -T 10 buffer\_temp

pgbench (14.2 (Ubuntu 14.2-1.pgdg18.04+1))

starting vacuum...end.

progress: 1.0 s, 525.9 tps, lat 1.887 ms stddev 0.404

progress: 2.0 s, 499.0 tps, lat 2.003 ms stddev 0.407

progress: 3.0 s, 489.0 tps, lat 2.048 ms stddev 0.389

progress: 4.0 s, 526.0 tps, lat 1.897 ms stddev 0.258

progress: 5.0 s, 598.0 tps, lat 1.674 ms stddev 0.319

progress: 6.0 s, 666.0 tps, lat 1.499 ms stddev 0.229

-- Запускаем в асинхронном режиме

postgres=# show synchronous\_commit;

synchronous\_commit

--------------------

off

(1 row)

postgres=# \q

postgres@postgres25032022:/home/anadyrov$ pgbench -i buffer\_temp

dropping old tables...

creating tables...

generating data (client-side)...

100000 of 100000 tuples (100%) done (elapsed 0.10 s, remaining 0.00 s)

vacuuming...

creating primary keys...

done in 0.42 s (drop tables 0.01 s, create tables 0.00 s, client-side generate 0.25 s, vacuum 0.08 s, primary keys 0.08 s).

postgres@postgres25032022:/home/anadyrov$ pgbench -P 1 -T 10 buffer\_temp

pgbench (14.2 (Ubuntu 14.2-1.pgdg18.04+1))

starting vacuum...end.

progress: 1.0 s, 1179.0 tps, lat 0.844 ms stddev 0.135

progress: 2.0 s, 1230.0 tps, lat 0.812 ms stddev 0.120

progress: 3.0 s, 1221.0 tps, lat 0.819 ms stddev 0.131

progress: 4.0 s, 1234.1 tps, lat 0.810 ms stddev 0.112

progress: 5.0 s, 1237.0 tps, lat 0.808 ms stddev 0.120

progress: 6.0 s, 1207.9 tps, lat 0.827 ms stddev 0.155

progress: 7.0 s, 1202.1 tps, lat 0.831 ms stddev 0.163

progress: 8.0 s, 1243.9 tps, lat 0.803 ms stddev 0.126

progress: 9.0 s, 1222.0 tps, lat 0.818 ms stddev 0.119

progress: 10.0 s, 1153.1 tps, lat 0.867 ms stddev 0.180

transaction type: <builtin: TPC-B (sort of)>

scaling factor: 1

query mode: simple

number of clients: 1

number of threads: 1

duration: 10 s

number of transactions actually processed: 12131

latency average = 0.823 ms

latency stddev = 0.138 ms

initial connection time = 4.912 ms

tps = 1213.666661 (without initial connection time)

Асинхронная фиксация — это возможность завершать транзакции быстрее.

В режиме асинхронного подтверждения сервер сообщает об успешном завершении сразу, как только транзакция будет завершена логически, прежде чем сгенерированные записи WAL фактически будут записаны на диск. Это может значительно увеличить производительность при выполнении небольших транзакций.

6. Создайте новый кластер с включенной контрольной суммой страниц.

root@postgres25032022:/home/anadyrov# pg\_ctlcluster 14 main stop

root@postgres25032022:/home/anadyrov# pg\_checksums --pgdata /var/lib/postgresql/14/main --enable

Checksum operation completed

Files scanned: 1253

Blocks scanned: 8637

pg\_checksums: syncing data directory

pg\_checksums: updating control file

Checksums enabled in cluster

Создайте таблицу.

Вставьте несколько значений.

buffer\_temp=# CREATE TABLE t(id integer);

CREATE TABLE

buffer\_temp=# INSERT INTO t VALUES (1),(2),(3);

INSERT 0 3

buffer\_temp=# SELECT pg\_relation\_filepath('t');

pg\_relation\_filepath

----------------------

base/16384/16537

(1 row)

buffer\_temp=# \q

root@postgres25032022:/var/lib/postgresql/14/main/base/16384# pg\_ctlcluster 14 main stop

Что и почему произошло?

Как проигнорировать ошибку и продолжить работу?

ERROR: invalid page in block 0 of relation base/16384/16537

Необходимо включить параметр ignore\_checksum\_failure, который позволит прочитать данную таблицу даже если данные в нем повреждены или искажены.