PG_PROBACKUP
Бэкапируем PostgreSQL

Postgres PROFESSIONAL

Быстро и
Параноидально

postgrespro.ru

History

Егор Рогов, PGCONF 2017:
https://pgconf.ru/2017/93506

Иван Картышев, DevFest Siberia 2017:
https://www.youtube.com/watch?v=tZmRbsYXkPQ

PG_PROBACKUP

Поддерживаемые версии PostgreSQL: >= 9.5

Исходники:

https://github.com/postgrespro/pg_probackup

Пакеты:

Centos(6, 7)
Debian(7, 8, 9)
Ubuntu(17.10, 16.04, 14.04, 12.04)

Документация:

https://postgrespro.ru/docs/postgrespro/current/app-pgprobackup

Фичи:

- Многопоточность
- Инкрементальность на уровне блока(PAGE, PTRACK)
- Компрессия файлов данных(zlib, pglz)
- Компрессия архивных WAL-сегментов(zlib, pglz)
- Политики Хранения(время, кол-во)
- Валидация блоков
- Валидация точки восстановления для PITR*
- Логирование(ротация, error_log)
- Бэкап с реплики*
- Каталог резервных копий
- Отправка/доставка сегментов WAL`а в архив: archive-push/archive-get
- Доставка сегментов WAL`а в архив с помощью pg_receivexlog

TODO:

- Валидация PITR с учетом таймлайнов(in development)
- Удаленный бэкап(расширение протокола репликации, in QA)
- Починка блока(in QA)
- Валидация инстанса PostgreSQL
- SQL интерфейс
- Бэкап отстающей реплики*
- Управление WAL-архивом
- WINDOWS support
- DIRECT I/O
- Глубина хранения сегментов WAL`а
- Политика хранения по размеру
- Шифрование

TODO(продолжение):

- Поддержка ленточных хранилищ
- Снапшоты ФС
- S3
- Троттлинг

Стенд

- 8 CPU Xeon X5675
- 512 GB RAM
- Ubuntu 16.04.3 LTS 64-bit
- PosgreSQL 9.6.6
- PGDATA 89 GB
- 500 GB tmpfs
- pgbackrest v1.27
- pgbarman v2.1
- pg_probackup v2.0.14

Замеры сделаны Ильей Скворцовым i.skvortsov@postgrespro.ru

Производительность

Параметры	pg_Probackup	pgBackRest	Barman (rsync)
Полный бэкап, 1 поток	2m20,053s	4m50,191s	11m0,668s
Полный бэкап, 4 потока	0m50,436s	1m22,266s	6m18,816s
Инкрементальный бэкап, 1 поток	0m5,507s	4m36,398s	10m30,896s
Инкрементальный бэкап, 4 потока	0m5,436s	1m21,660s	4m44,504s
Восстановление из бэкапа, 1 поток	3m23,051s	5m39,827s	6m9,194s
Восстановление из бэкапа, 4 потока	1m36,394s	1m47,143s	1m26,893s

Объем

	pg_Probackup	pgBackRest	Barman (rsync)
Полный бэкап	88 ГБ	88 ГБ	87,7 ГБ
Инкрементальный бэкап	186 МБ	88 ГБ	88 ГБ
Сжатый полный бэкап	7,9 ГБ	4,7 ГБ	Н/Д

Инкрементальный бэкап создавался после запуска «pgbench -c 10 -t 1000» (10 000 транзакций)

Производительность

Параметры	pg_Probackup	pgBackRest	pg_basebackup
Полный сжатый бэкап в 1 поток	30m9,122s	30m24,367s	28m55,900s
Полный сжатый бэкап в 4 потока	11m44,200s	8m11,259s	N/A

zlib, compression-level=6

Производительность

Однопоточный режим

pg_Probackup	pg_basebackup	cp
2m20,053s	3m52,583s	1m5,293s

PgBackRest и Barman

Отличные утилиты

НО ЕСТЬ ПРОБЛЕМА

POSTGRESQL

СУБД будущего

```
# recovery.conf
restore_command = 'pg_probackup archive-get -B
backup --instance node --wal-file-path %p
--wal-file-name %f'

recovery_target_action = 'promote'

recovery_target_time = '40000-01-01 00:00:00+03'
```

СУБД будущего

LOG: selected new timeline ID: 2

LOG: archive recovery complete

LOG: MultiXact member wraparound protections are now enabled

LOG: database system is ready to accept connections

СУБД будущего

LOG: selected new timeline ID: 2

LOG: archive recovery complete

LOG: MultiXact member wraparound protections are now enabled

LOG: database system is ready to accept connections

Валидация точки восстановления

```
pg_probackup restore -D $PGDATA --time='40000-01-01
00:00:00+03'

INFO: Validating backup P3R6CE
INFO: Backup P3R6CE data files are valid
WARNING: recovery can be done up to time 2018-02-07
02:53:16+03 and xid 1805
ERROR: not enough WAL records to time 40000-01-01
00:00:00+03
```

Валидация имеющихся бэкапов

1. Валидация родителей перед инкрементальным бэкапом
2. Валидация всей цепочки перед восстановлением
3. Валидация WAL`ов

Валидация блоков

1. initdb --data-checksums
2. При чтении блока можно сверять его чексумму
3. Ложно-отрицательные срабатывание при высокой нагрузке
4. pg_ptrack_get_block()

Валидация блоков

1. initdb --data-checksums
2. При чтении блока можно сверять его чексумму
3. Ложно-отрицательные срабатывание при высокой нагрузке
4. pg_ptrack_get_block()

Silent Data Corruption is silent

https://www.novell.com/support/kb/doc.php?id=7017183

https://bugzilla.redhat.com/show_bug.cgi?id=845233

http://www.pointsoftware.ch/en/4-ext4-vs-ext3-filesystem-and-why-delayed-allocation-is-bad/

Что делать?

Хранить информацию о длине релейшенов в pg_class?

Неплохой план

НО ЕСТЬ ПРОБЛЕМА

POSTGRESQL

Легализация потери сегмента

с помощью RECOVERY

Легализация потери сегмента

```
postgres=# create table eat_my_data(int int);
postgres=# insert into eat_my_data select from
generate_series(1,200);
postgres=# CHECKPOINT;
postgres=# insert into eat_my_data select from
generate_series(1,200);
postgres=# CHECKPOINT;
postgres=# insert into eat_my_data select from
```

Легализация потери сегмента

```
postgres=# select pg_relation_filepath('eat_my_data');
 base/13356/40963
```

```
postgres=# select pg_backend_pid();
      12204
```

Легализация потери сегмента

```
[#] kill -9 12204
[#] truncate $PGDATA/base/13356/40963
[#] pg_ctl start -D $PGDATA
```

Легализация потери сегмента

```
[postgres=# select count(*) from t1;
 count
-------
   400
```

¯_(ツ)_/¯

Вывод

1. Информация о размерах файлов должна быть доступа до начала RECOVERY
2. Что-то поменять в RECOVERY

Прочее

- BACKUP_END не содержит timestamp
- Удаления/изменения файлов не пишутся в WAL
- pg_start_backup принудительно включает FPW
- Нельзя отличить реплику от восстановления после сбоя (StandbyModeRequested)
- Табличные пространства в PGDATA разрешены

Прочее

- Реплика не может попросить мастера сделать switch_xlog
- archive_command не знает о System ID
- Нельзя управлять номером таймлайна
- Нет возможности отключать HINT BITs
- Debian like дистибутивы размазывают конфиги по всей системе

PTRACK

```
-rw------- 1 gsmol gsmol  221184 Feb  7 08:04 16431
-rw------- 1 gsmol gsmol    8192 Feb  7 08:04 16431_ptrack
```

- Битовая карта для таблиц и индексов
- Разделена на 8КБ страницы.
- Может хранить факт изменения 65536 обычных страниц.
- 1 означает факт изменения соответствующей ей страницы
- Ставится при любом изменении страницы в буфере
- GUC ptrack_enable

PTRACK

- Снимается пользовательской ф-цией pg_ptrack_get_and_clear(tblspace_oid, relfilenode_oid)
- Чтобы получить полную карту нужно обойти битовые карты всех релейшенов
- файл ptrack_control
- pg_ptrack_control_lsn()
- Консистентность _ptrack обеспечивается во время recovery

PTRACK

- Снимается пользовательской ф-цией pg_ptrack_get_and_clear(tblspace_oid, relfilenode_oid)
- Чтобы получить полную карту нужно обойти битовые карты всех релейшенов

Неплохая идея

НО ЕСТЬ ПРОБЛЕМА

POSTGRESQL

Нарушает собственные абстракции

- CREATE DATABASE и ALTER DATABASE SET TABLESPACE копирую файлы мимо буфера и WAL`а.
- Костыль-файл ptrack_init

PTRACK

- https://wiki.postgresql.org/wiki/PTRACK_incremental_backups
- https://www.postgresql.org/message-id/d2842a0f-d858-4e41-2805-818974646560%402ndquadrant.com

Postgres PROFESSIONAL

ВОПРОСЫ?

Postgres Professional

http://postgrespro.ru/

+7 495 150 06 91

info@postgrespro.ru

postgrespro.ru