Университет ИТМО
Факультет программной инженерии и компьютерной техники

Распределённые системы хранения данных. Лабораторная работа №3.

Группа: P33121
Студенты: Гиниятуллин Арслан Рафаилович
Бекмухаметов Владислав Робертович
Преподаватель: Афанасьев Дмитрий Борисович
Вариант: 721

2024

Ключевые слова

База данных, PostgreSQL, резервное копирование, повреждение данных

Содержание

1 Цель работы

Настроить процедуру периодического резервного копирования базы данных, сконфигурированной в ходе выполнения предыдущей лабораторной работы, а также разработать и отладить сценарии восстановления в случае сбоев.

2 Текст задания

2.1 Описание

2.1.1 Этап 1. Резервное копирование

- Настроить резервное копирование с основного узла на резервный следующим образом: Первоначальная полная копия + непрерывное архивирование. Включить для СУБД режим архивирования WAL; настроить копирование WAL (scp) на резервный узел; создать первоначальную резервную копию (pg_basebackup), скопировать на резервный узел (rsync).
- Подсчитать, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:
  - Средний объем новых данных в БД за сутки: 850МБ.
  - Средний объем измененных данных за сутки: 600МБ.
- Проанализировать результаты.

2.1.2 Этап 2. Потеря основного узла

Этот сценарий подразумевает полную недоступность основного узла. Необходимо восстановить работу СУБД на РЕЗЕРВНОМ узле, продемонстрировать успешный запуск СУБД и доступность данных.

2.1.3 Этап 3. Повреждение файлов БД

Этот сценарий подразумевает потерю данных (например, в результате сбоя диска или файловой системы) при сохранении доступности основного узла. Необходимо выполнить полное восстановление данных из резервной копии и перезапустить СУБД на ОСНОВНОМ узле.

Ход работы:

- Симулировать сбой:
  - удалить с диска директорию конфигурационных файлов СУБД со всем содержимым.
- Проверить работу СУБД, доступность данных, перезапустить СУБД, проанализировать результаты.
- Выполнить восстановление данных из резервной копии, учитывая следующее условие:
  - исходное расположение директории PGDATA недоступно - разместить данные в другой директории и скорректировать конфигурацию.
- Запустить СУБД, проверить работу и доступность данных, проанализировать результаты.

2.1.4 Этап 4. Логическое повреждение данных

Этот сценарий подразумевает частичную потерю данных (в результате нежелательной или ошибочной операции) при сохранении доступности основного узла. Необходимо выполнить восстановление данных на ОСНОВНОМ узле следующим способом:

- Генерация файла на резервном узле с помощью pg_dump и последующее применение файла на основном узле.

Ход работы:

- В каждую таблицу базы добавить 2-3 новые строки, зафиксировать результат.
- Зафиксировать время и симулировать ошибку:
  - удалить любые две таблицы (DROP TABLE)
- Продемонстрировать результат.
- Выполнить восстановление данных указанным способом.
- Продемонстрировать и проанализировать результат.

3 Этапы выполнения работы

Основной кластер: pg105 Резервный кластер: pg121

3.1 Этап 1

Базовая настройка основного кластера

```
echo "wal_level = replica" >> postgresql.conf
echo "archive_mode = on" >> postgresql.conf
echo "archive_command = 'scp %p postgres1@pg121:/var/db/postgres1/u02/gsd65/%f'"
>> u01/gsd65/postgresql.conf
```

- *wal_ level = replica* устанавливает уровень записи WAL на лидере, достаточный для резервного копирования с другого узла
- *archive_ mode = on* указывает, что полные сегменты WAL передаются в хранилище архива командой *archive_ command*
- *archive_ command = 'scp %p postgres1@pg121:/var/db/postgres1/u02/gsd65/%f'* определяет команду, которая будет выполняться над завершенными WAL сегментами для архивации. (Любое вхождение %p в этой строке заменяется путём архивируемого файла, а вхождение %f заменяется только его именем.)

Сохранение ssh ключа на резервный кластер.

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub postgres1@pg121
```

Создание базового бекапа и настройка rsync для директории с файлами бекапа.

```
pg_ctl -D /var/db/postgres0/u01/gsd65 -l logfile start
pg_basebackup -P -Ft -p 9006 -X fetch -D backups
rsync --archive --verbose --progress --rsync-path=/usr/local/bin/rsync \
/var/db/postgres0/backups/ postgres1@pg121:/var/db/postgres1/backups
```

Подсчитаем, каков будет объем резервных копий спустя месяц работы системы, исходя из следующих условий:

- Средний объем новых данных в БД за сутки: 850МБ.
- Средний объем измененных данных за сутки: 600МБ.

Для начала посчитаем объем бекапа

```
[postgres0@pg105 ~/backups]$ du -sh
7,4M .
```

current_backup_data_volume = 7.4МБ

Далее экстраполируем средний объём новых данных за месяц

total_new_data_volume = 850МБ * 30 = 22.5ГБ
total_changed_data_volume = 600МБ * 30 = 18ГБ
total_backup_data_volume = current_backup_data_volume + total_new_data_volume
+ total_changed_data_volume = 40507,4МБ

3.2 Этап 2

Восстановление работы СУБД на резервном узле

```
tar xvf backups/base.tar -C $HOME/u01/gsd65
tar xvf backups/16384.tar -C  $HOME/u03/gsd65
tar xvf backups/16385.tar -C $HOME/u04/gsd65
tar xvf backups/16386.tar -C $HOME/u05/gsd65

ln -s u03/gsd65 u01/gsd65/pg_tblspc/16384
ln -s u04/gsd65 u01/gsd65/pg_tblspc/16385
ln -s u05/gsd65 u01/gsd65/pg_tblspc/16386
```

Настраиваем restore_command в postgresql.conf и расширяем маппинг пользоваталей, так как пользователь изменился с postgres0 -> postgres1

```
echo  "restore_command = 'cp /var/db/postgres1/u02/gsd65/%f %p'"
>> $HOME/u01/gsd65/postgresql.conf
echo "postgres0 postgres1 s000000" >> $HOME/u01/gsd65/pg_ident.conf
```

Создаем standby.signal для запуска СУБД в режиме standby

```
touch u01/gsd65/standy.signal
```

Запускаем

```
pg_ctl -D u01/gsd65 -l файл_журнала start
```

3.3 Этап 3

Повреждение файлов бд и полное восстановление данных на основном узле с резервного.
Симулируем сбой: удаляем с диска директорию конфигурационных файлов

```
rm -fr ~/u01/gsd65/*
mkdir ~/u10/gsd65
chmod 700 ~/u10/gsd65
```

Подтягиваем данные с резервного узла для восстановления

```
mkdir ~/new_backups
scp postgres1@pg121:/var/db/postgres1/backups/base.tar ~/new_backups/
scp postgres1@pg121:/var/db/postgres1/backups/16384.tar ~/new_backups/
scp postgres1@pg121:/var/db/postgres1/backups/16385.tar ~/new_backups/
scp postgres1@pg121:/var/db/postgres1/backups/16386.tar ~/new_backups/
```

Восстанавливаем данные в других директориях

```
tar xvf ~/new_backups/base.tar -C ~/u10/gsd65
tar xvf ~/new_backups/16384.tar -C ~/u03/gsd65
tar xvf ~/new_backups/16385.tar -C ~/u04/gsd65
tar xvf ~/new_backups/16386.tar -C ~/u05/gsd65
```

Задаем команду для восстановления, создаем сигнал для запуска в recovery mod и запускаем сервер

```
echo "restore_command = 'scp postgres1@pg121:/var/db/postgres1/u02/gsd65/%f %p'"
>> $HOME/u10/gsd65/postgresql.conf
touch $HOME/u10/gsd65/recovery.signal
pg_ctl -D $HOME/u10/gsd65 -l logfile start
echo "postgres0 postgres1 s000000" >> $HOME/u01/gsd65/pg_ident.conf
```

3.4 Этап 4

Вставим строки на основном узле

```
greatercapybara=# INSERT INTO wolf (name, type, age) VALUES ('name_1', 'default', 1);
INSERT 0 1
greatercapybara=# INSERT INTO wolf (name, type, age) VALUES ('name_2', 'default', 2);
INSERT 0 1
greatercapybara=# INSERT INTO wolf (name, type, age) VALUES ('name_3', 'default', 3);
INSERT 0 1
greatercapybara=# select * from wolf;
 id |  name  |  type   | age
----+--------+---------+-----
  1 | name_1 | default |   1
  2 | name_2 | default |   2
  3 | name_3 | default |   3
(3 строки)
```

Проверим их наличие на резервном узле

```
greatercapybara=# select * from wolf;
 id |  name  |  type   | age
----+--------+---------+-----
  1 | name_1 | default |   1
  2 | name_2 | default |   2
  3 | name_3 | default |   3
(3 строки)
```

Фиксируем время

```
greatercapybara=# select now();
              now
-------------------------------
 2024-04-28 20:15:12.259434+03
(1 строка)
```

Создадим dump на резервном сервере

```
$ mkdir ~/dumps
$ pg_dump -p 9006 -d greatercapybara -U postgres0 -Fc > ~/dumps/db-$(date +"%m-%d-%Y-%H-%M-%S").
```

Отправим его на основной

```
$ rsync --archive ~/dumps postgres0@pg105:
```

Правим данные

```
greatercapybara=# INSERT INTO wolf (name, type, age) VALUES ('corrupted_name', 'default', 4);
INSERT 0 1
greatercapybara=# INSERT INTO wolf (name, type, age) VALUES ('corrupted_name_2', 'default', 4);
INSERT 0 1
greatercapybara=# select * from wolf;
 id |       name       |  type   | age
----+------------------+---------+-----
  1 | name_1           | default |   1
  2 | name_2           | default |   2
  3 | name_3           | default |   3
  4 | corrupted_name   | default |   4
  5 | corrupted_name_2 | default |   4
(5 строк)
```

Восстановимся на основном сервере из dump-а

```
$ pg_restore -p 9006 -d greatercapybara -c db-04-28-2024-20-34-55.dump
```

Проверим данные

```
greatercapybara=# select * from wolf;
 id |  name  |  type   | age
----+--------+---------+-----
  1 | name_1 | default |   1
  2 | name_2 | default |   2
  3 | name_3 | default |   3
(3 строки)

greatercapybara=# select now();
              now
-------------------------------
 2024-04-28 20:38:42.713888+03
(1 строка)
```