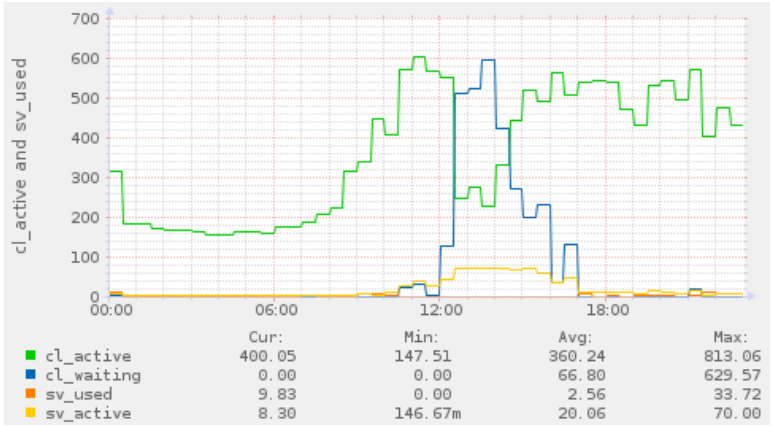
Восстановление и другие подходы к авариям

Сергей Бурладян
sburladyan@avito.ru

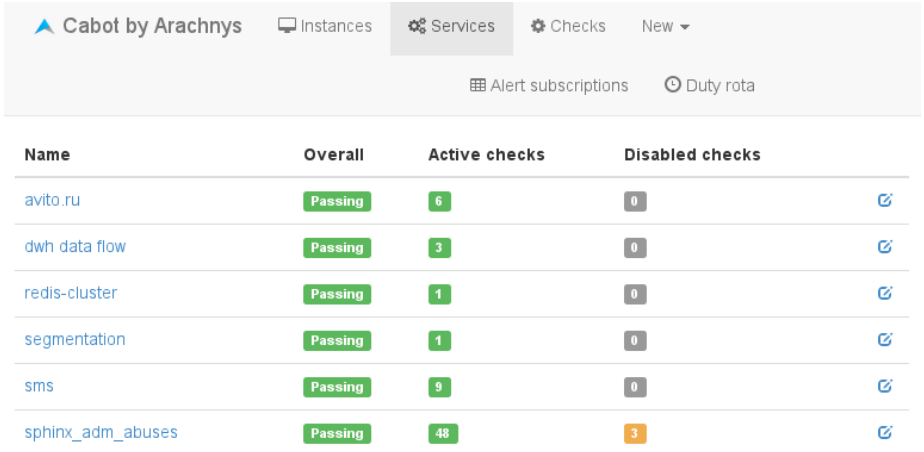
2015, Москва

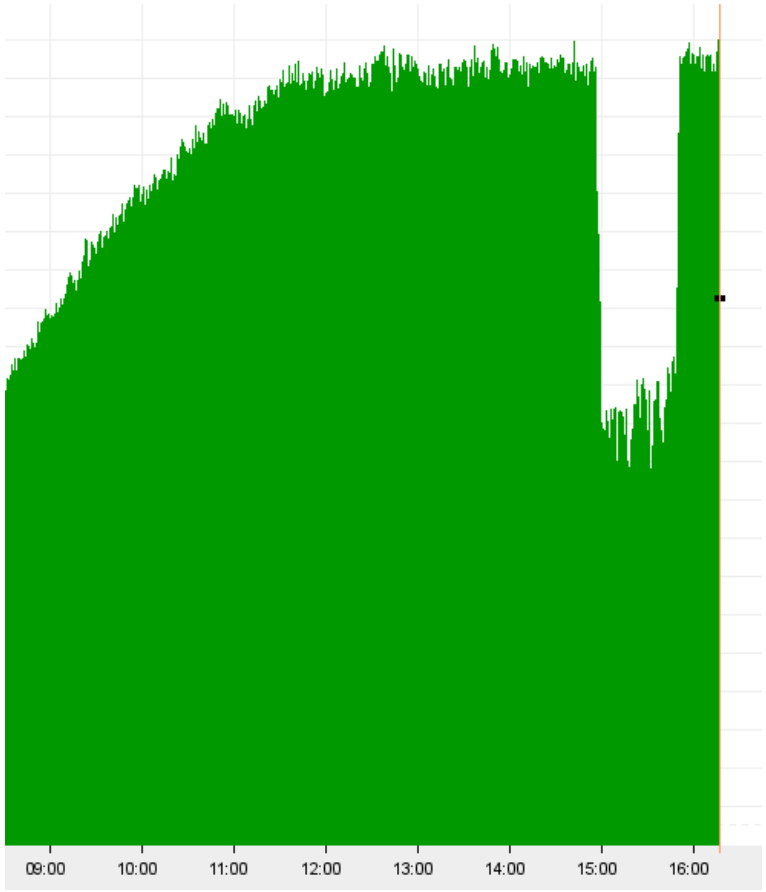
Определение источника аварии

- munin



- cabot



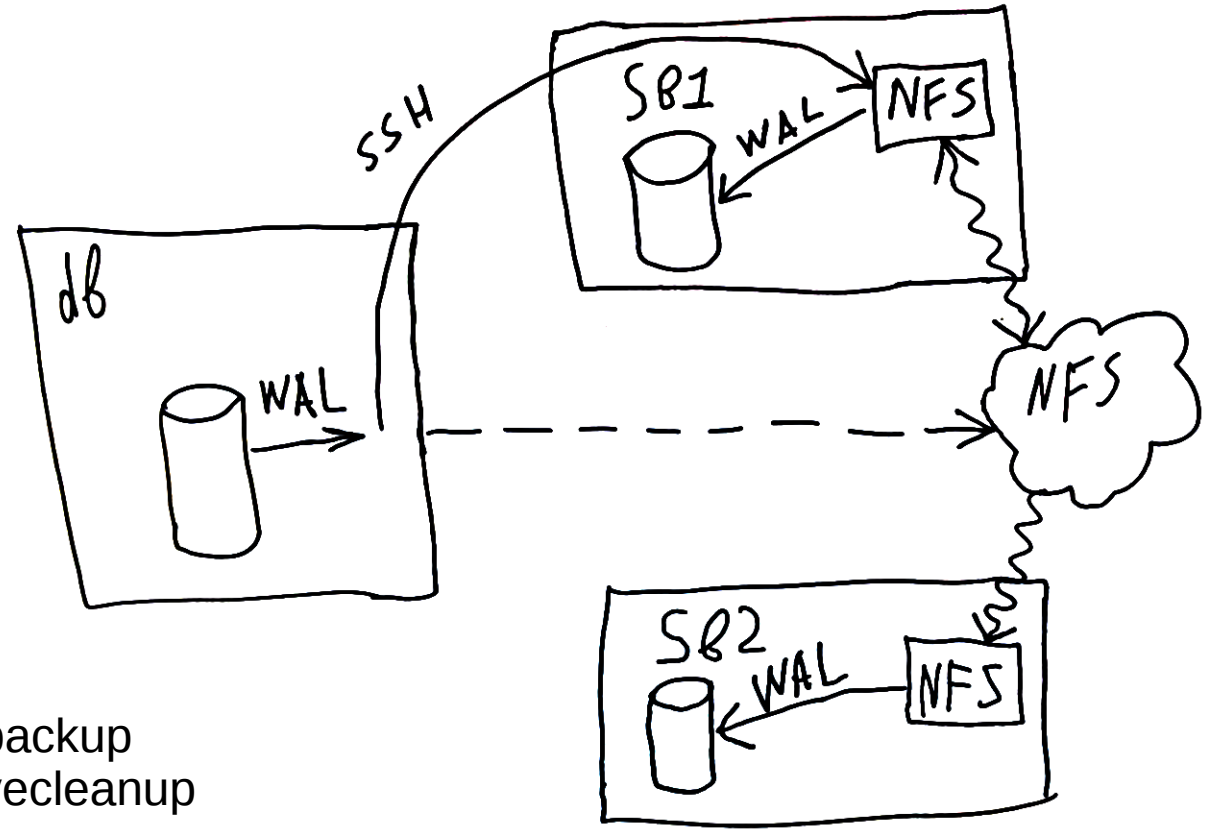


Возможные аварии

- master
  - unlogged tables — restart_after_crash = off
  - tmpfs — нет автозапуска после загрузки
- standby в бою
- узел xdb
- repca

Почему мы можем восстановиться

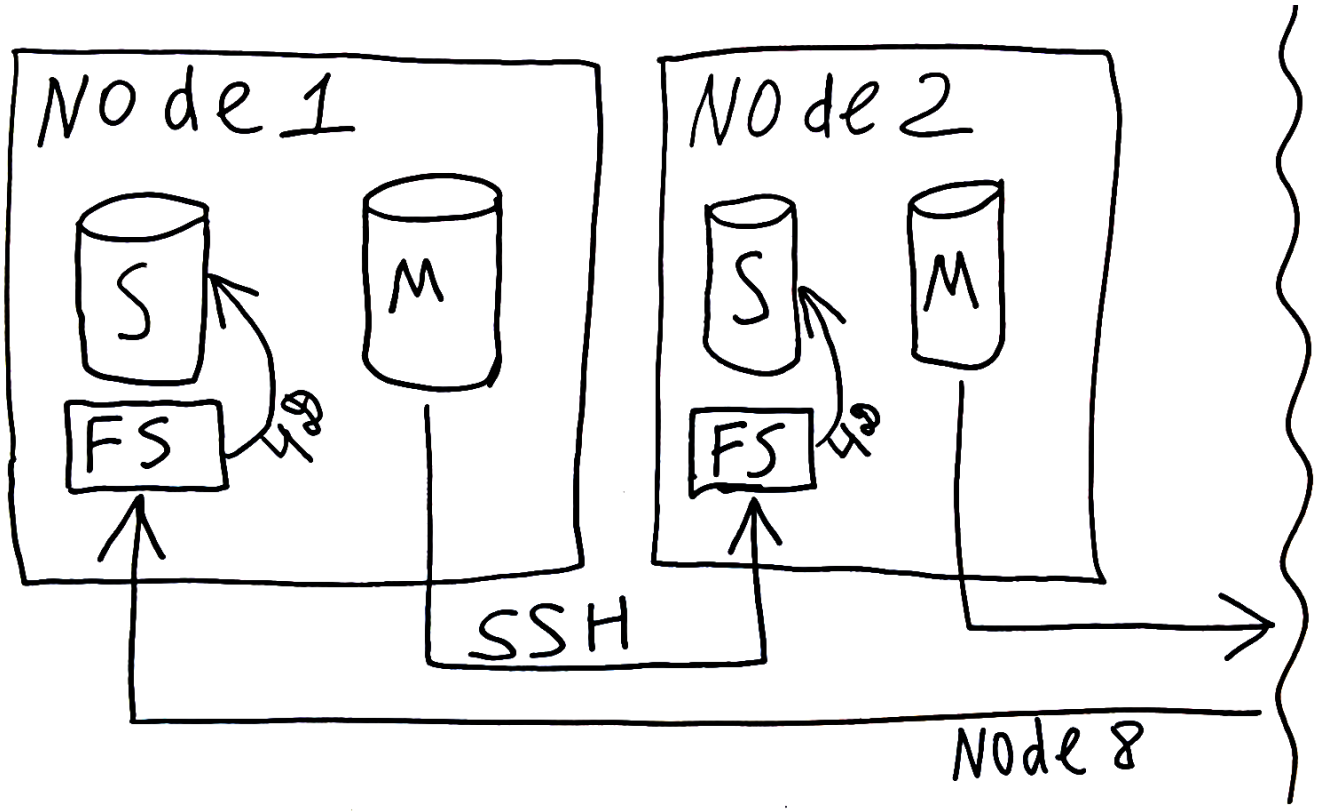
- standby (WAL)
  - archive_command = archive_cmd.sh host-sb1 /srv/walshipping/logs %p %f



- PITR
- pg_basebackup
- pg_archivecleanup

Почему мы можем восстановиться

- xdb standby
  - на соседнем узле, node01 → node02, …, nodeX → node01 с задержкой применения WAL в 4 дня

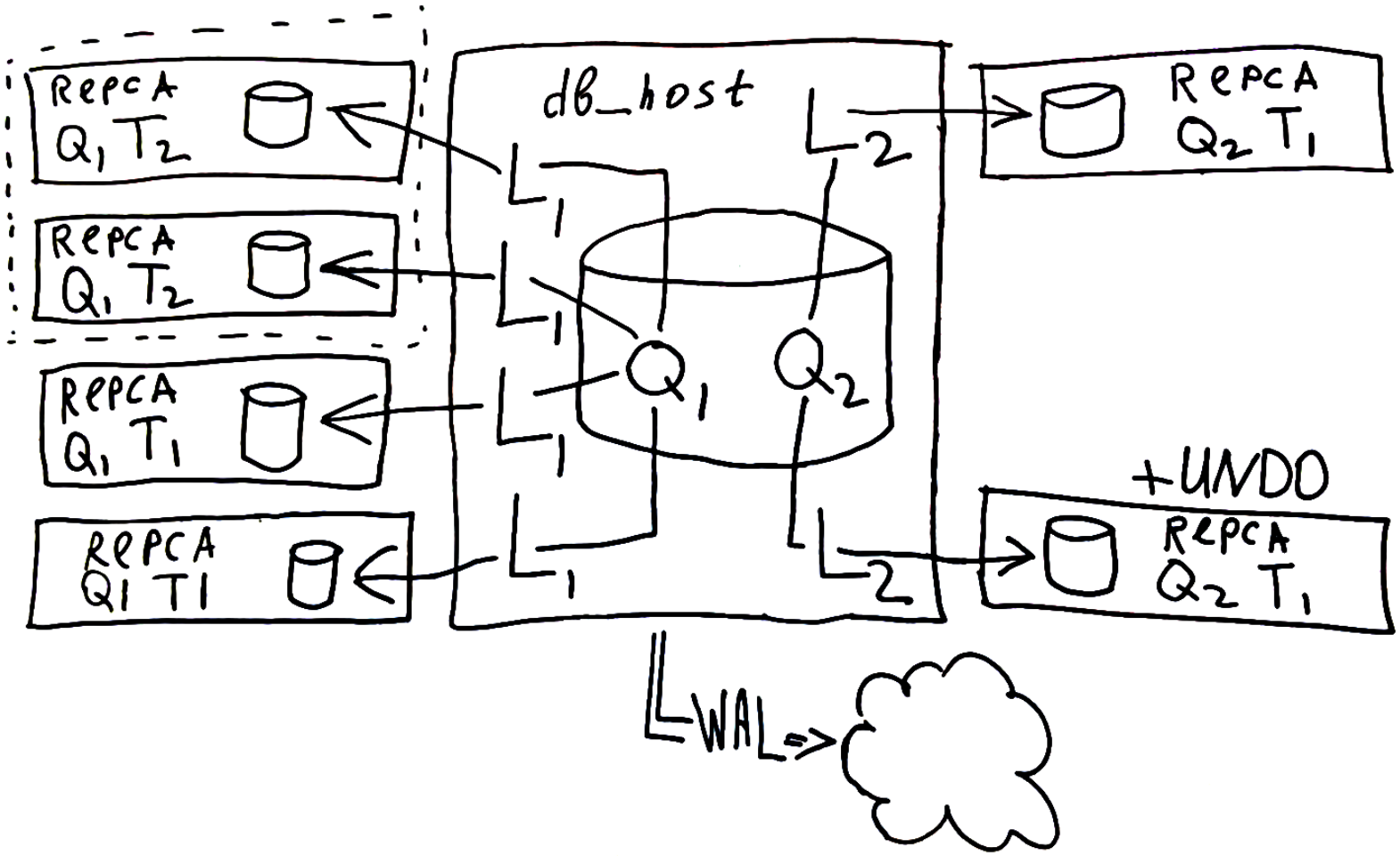


Почему мы можем восстановиться

- копии repca
  - два или больше consumer на один источник событий londiste

```
$ pgqadm /etc/skytools/ticker.conf status
Consumer                                                  Lag  LastSeen
------------------------------------------------------------------------
londiste.replika:
  londiste_avito_new_1                                    1s       0s
  londiste_avito_new_2                                    1s       0s
```

Почему мы можем восстановиться



Почему мы можем восстановиться

- londiste, pgq, undo log
  - отслеживание tick_id

    https://github.com/markokr/skytools/blob/skytools_2_1_stable/python/londiste/playback.py

    londiste: class Replicator(pgq.SerialConsumer)

```
class SerialConsumer(Consumer):
    """Consumer that applies batches sequentially in second database.
    Requirements:
     - Whole batch in one TX.
     - Must not use retry queue.
    Features:
     - Can detect if several batches are already applied to dest db.
     - If some ticks are lost. allows to seek back on queue.
       Whether it succeeds, depends on pgq configuration.
    """
```

Почему мы можем восстановиться

- londiste, pgq, undo log
  - отслеживание tick_id

    https://github.com/markokr/skytools/blob/skytools_2_1_stable/python/pgq/consumer.py

    PgQ: class SerialConsumer(Consumer)

```
def process_batch(self, db, batch_id, event_list):
        """Process all events in batch.
        """

        ...

        # check if done
        if self.is_batch_done(curs):
            for ev in event_list:
                ev.tag_done()
            return

        # actual work
        self.process_remote_batch(db, batch_id, event_list, dst_db)

        # finish work
        self.set_batch_done(curs)
        dst_db.commit()
```

Почему мы можем восстановиться

- londiste, pgq, undo log
  - сама очередь это REDO

select ev_id, ev_time, ev_txid, ev_type from pgq.event_9_1

	ev_id bigint	ev_time timestamp with time	ev_txid bigint	ev_type text
1	3692218680	2015-02-04 17:59:35	308124210	I
2	3692218679	2015-02-04 17:59:35	308124219	I
3	3692218678	2015-02-04 17:59:35	308124082	I
4	3692218677	2015-02-04 17:59:35	308124206	I
5	3692218675	2015-02-04 17:59:35	308124207	I
6	3692218673	2015-02-04 17:59:35	308124208	I
7	3692218672	2015-02-04 17:59:35	308124188	I
8	3692218671	2015-02-04 17:59:35	308124200	I
9	3692218670	2015-02-04 17:59:35	308124202	I
10	3692218669	2015-02-04 17:59:35	308124205	I

Почему мы можем восстановиться

- londiste, pgq, undo log
  - наш UNDO

```
provider curr-tick            show current tick on provider (for run undo on it)


subscriber undo TICK_ID       rewind all tables with it UNDO log
subscriber add-undo-all       enable UNDO log on all tables
subscriber remove-undo-all    disable UNDO log on all tables
subscriber curr-tick          show applied tick on subscriber


--enable-undolog  replay: generate UNDO log
```

  - крайний случай — не до конца полная репка с триггером BEFORE insert, пересоздание реплики

Почему мы можем восстановиться

- sphinx
  - выдача сайта — перестраивается каждые 15 минут
  - backoffice — не восстанавливаем, но можем переиндексировать за сутки, переиндексируем раз в несколько месяцев

Почему мы можем восстановиться

- дублирование данных, восстановление одних данных по другим
  - активный item: мастер, репка, sphinx, ...
- резервная копия — архив (два, текущий и прошлый)
  - + копия архива в облаке
  - проверка восстановимости архива внутри (обсчёт статистики по восстановленному архиву) и из облака (DevOps)
  - xdb standby с задержкой проигрывания в 4 дня

Способы восстановления

- promote standby
- переключение репки на копию
- undo репки
- крайний случай — восстановление из бекапа

Скрипт восстановления

1) выбор standby по hostname с исключением упавшего мастера
2) остановить все другие standby перед promote (timeline)
3) выполняем promote и ждём завершения promote
4) проверяем что переключились успешно (сервер запущен, pg_is_in_recovery = false)
5) проверить доступ на старый мастер по ssh
   - $ timeout ssh db id
   - $ timeout ssh db /usr/local/bin/start-master-db-services.sh stop
   - остановить pgbouncer на упавшем мастере, это позволит app быстрее переключится на новый DNS
6) запустить все сервисы (start-master-db-services.sh [1])
7) ! очистка лишних WAL

   [HACKERS] Bogus WAL segments archived after promotion

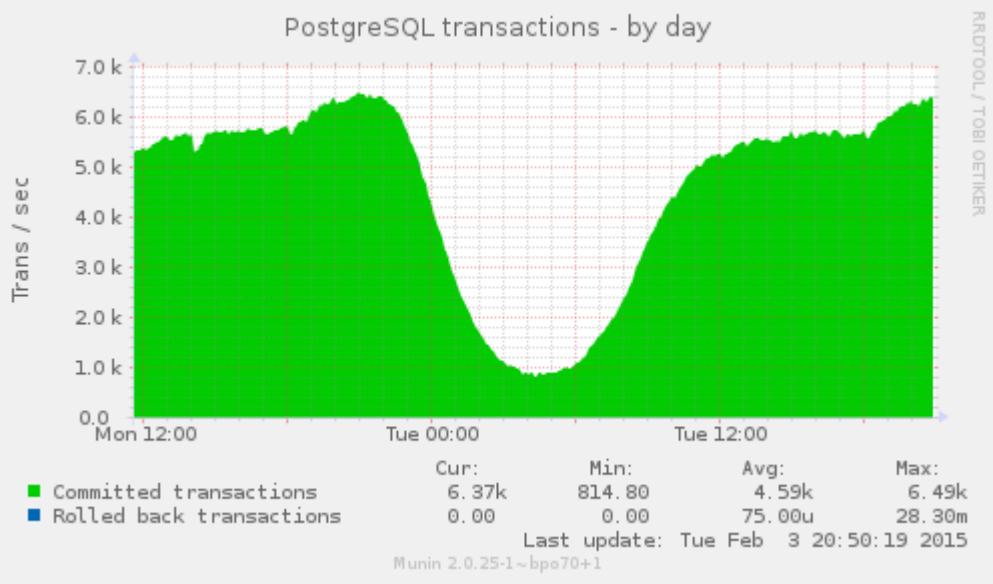
   http://www.postgresql.org/message-id/54942034.7080303@vmware.com
8) переключить archive_command на новый standby
9) запустить другие остановленные standby и ждать завершения их запуска

   пример:
   - 2014-12-22 08:41:30 (запуск)
   - 2014-12-22 08:48:45 (открыл read доступ)
   - 2014-12-22 08:50:05 (догнал мастера)

   [1] - по hostname и DNS определить какие londiste в бою, а какие — в резерве, запустить UNDO log на резервных

Проблемы

- как сделать pg_upgrade если standby в бою?



  - копирование кучи ~ 6 часов, ограничено сетью — пока можем между пиками нагрузки обновлять версию pg

Спасибо за внимание!

Сергей Бурладян
sburladyan@avito.ru

БЦ «Белые сады», ул. Лесная, 7, Москва, 125047