# RTFM

Federico Campoli

FOSDEM PGDay, Brussels, 02 February 2020

# Few words about the speaker


- Born in 1972
- Passionate about IT since 1982
- Joined the Oracle DBA secret society in 2004
- In love with PostgreSQL since 2006
- PostgreSQL tattoo on the right shoulder
- Freelance devops and data engineer

# Getting in touch

- **Blog:** https://pgdba.org
- **Twitter:** @4thdoctor_scarf
- **Github:** https://github.com/the4thdoctor
- **Linkedin:** https://www.linkedin.com/in/federicocampoli/

# RTFM

Acronym for **R**ead **T**he **F**... **M**anual
What the **F** stands for?

# RTFM

- Fantastic?
- Fabulous?
- Funny?
- Fancy?
- Well...


F******

# Defcon levels

- **Defcon 5:** Startling noise, DBA vaguely impressed
- **Defcon 4:** Tripping over feet, DBA alarmed
- **Defcon 3:** Earthquake, DBA jumping on the seat
- **Defcon 2:** Asteroid dropping from the sky, DBA freaking out
- **Defcon 1:** DALEKS!, DBA going berserk

# Dramatis personae

Who didn't RTFM?

Myself

Image source wargames Wikia https://war-games.fandom.com/

# Dramatis personae

Who didn't RTFM?

Others

Image source wargames Wikia https://war-games.fandom.com/

# Table of contents

# The Butler Did It

**Defcon 4**

# The Butler Did It

Context

- System with one expensive FusionIO card
- Working queue organised on a single table
- Table with about 100 million rows
- Two timestamp fields
- Updated twice at the start and the end of the processing
- For each row
- Which had an average width of 160 bytes
- On a table with three indices
- Plus the primary key...
- Spanning on two integer fields

# The Butler Did It

- On a SSD writes are limited
- The design caused 1.5 GB/s block writes on the pg_xlog
  (on rotating disk thanks goodness)
- The data files were hit even harder
- The table was completely rewritten every day
- Autovacuum starting every 6 hours flushing even more data to disk
- In just 8 months the available writes dropped from 80% to 44%
- At that rate only 8 months left before the doomsday

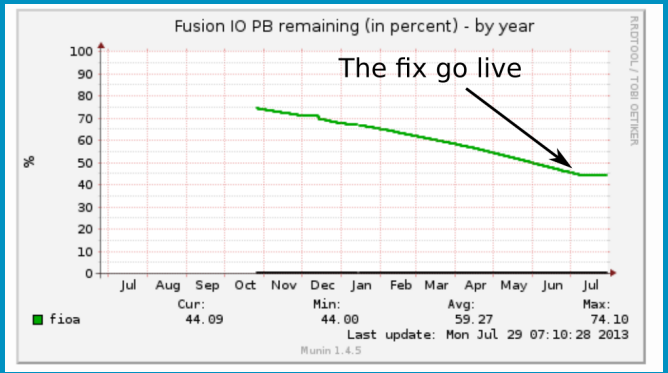
# DISK IN READ ONLY MODE

# The Butler Did It

How it was fixed:

- The primary key was aggregated in a separate table
- The first field had a common value shared with the second field
- The first field was used as grouping key
- The second field was aggregated into an `integer[]`
- The queue was implemented using the physical position inside the array
- After the deploy the WAL generation rate dropped to 40 MB/s

# The Butler Did It




The doomsday countdown

# The Butler Did It

**RTFM**

- **MVCC Unmasked (slides)**: http://momjian.us/main/writings/pgsql/mvcc.pdf
- **MVCC Unmasked (video)**: https://www.youtube.com/watch?v=sq_aO34SWZc

# Emergency shutdown

**Defcon 1**

# Emergency shutdown

Context

- Virtual machine for business intelligence
- Fairly big database about 1.4 TB
- Real time replica from MySQL with pg_chameleon
- Replay chunk 100k rows
- pl/pgSQL function to replay data and manage errors
- Monitoring yet to be implemented

# Emergency shutdown

- The day started normally
- Suddenly people got errors on their queries
- The database was up as usual
- However the nightly maintenance failed!

# Emergency shutdown

```
ERROR:  database is not accepting commands
to avoid wraparound data loss in database "analytics"

HINT:  Stop the postmaster and vacuum
that database in single-user mode.
```

# Emergency shutdown

Caused by

- Insufficient autovacuum processes
- An (apparently) undocumented behaviour of the pl/pgSQL functions

# Emergency shutdown

- A PostgreSQL function is single transaction
- Normally consumes just one XID
- However DML in blocks with `EXCEPTION` consume an extra XID every time the DML is executed
- The pl/pgSQL function in pg_chameleon
- Replays the DML in a FOR LOOP with an `EXCEPTION`
- Consuming an XID for each statement replayed

# Emergency shutdown

Consider a table with one field, integer.

```
CREATE TABLE t_test
(
  id integer
);
```

# Emergency shutdown

A function inserts into the table with a FOR LOOP.

```
CREATE OR REPLACE FUNCTION fn_loop_noexception()
RETURNS VOID AS
$BODY$
        DECLARE
                v_loop integer;
        BEGIN
                FOR v_loop IN 1..1000
                LOOP
                        INSERT INTO t_test(id) VALUES (v_loop);
                END LOOP;
        END;
$BODY$
LANGUAGE plpgsql;
```

# Emergency shutdown

Another function does the same but with an EXCEPTION block.

```
CREATE OR REPLACE FUNCTION fn_loop_withexception()
RETURNS VOID AS
$BODY$
       DECLARE
              v_loop integer;
       BEGIN
              FOR v_loop IN 1..1000
              LOOP
                     BEGIN
                            INSERT INTO t_test(id) VALUES (v_loop);
                     EXCEPTION
                            WHEN OTHERS
                            THEN
                                 NULL;
                     END;
              END LOOP;
       END;
$BODY$
LANGUAGE plpgsql;
```

# Emergency shutdown

## Function without the exception block

```
test=# SELECT datname,age(datfrozenxid) FROM pg_database  WHERE datname='test';
 datname | age
---------+-----
 test    |   3
(1 row)

test=# SELECT fn_loop_noexception();
 fn_loop_noexception
---------------------

(1 row)

test=# SELECT datname,age(datfrozenxid) FROM pg_database  WHERE datname='test';
 datname | age
---------+-----
 test    |   4
(1 row)
```

# Emergency shutdown

## Function with the exception block

```
test=# SELECT datname,age(datfrozenxid) FROM pg_database  WHERE datname='test';
 datname | age
---------+-----
 test    |   5
(1 row)

test=# SELECT fn_loop_withexception();
 fn_loop_withexception
-----------------------

(1 row)

test=# SELECT datname,age(datfrozenxid) FROM pg_database  WHERE datname='test';
 datname | age
---------+------
 test    | 1006
(1 row)
```

# Emergency shutdown

How it was fixed:

- Silence slack
- Move in an empty meeting room
- Put the message **I KNOW!!!** on the entrance door
- Start the cluster in single user mode
- Get the ageing tables
- Vacuum the ageing tables
- Do a postmortem analysis

# Emergency shutdown

## Handy query to get the ageing tables

```
SELECT
  sch.nspname as schema_name,
  tab.relname as table_name,
  greatest(age(tab.relfrozenxid),age(toa.relfrozenxid)) as age
FROM
  pg_class toa
  LEFT JOIN pg_class tab
    ON tab.reltoastrelid = toa.oid
  INNER JOIN pg_namespace sch
    ON tab.relnamespace=sch.oid
WHERE
  tab.relkind IN ('r', 'm')
ORDER BY age DESC
;
```

# Emergency shutdown

**RTFM**

- I wasn't able to find this case on the documentation
- The warning box just says that functions with EXCEPTION are more expensive.
- It may be useful to add a warning explaining that the XID are consumed by DML within an EXCEPTION block.

# Cast a spell

**Defcon 5**

# Cast a spell

Context

- Fairly large database, 2 TB
- Horrible design
- Table with metrics mediated from java straight into an hstore field
- Query runtime to retrieve 150 rows about 6 minutes
- Despite the super expensive storage (FusionIO)
- And the super expensive cpu and memory (state of the art in 2013)

# Cast a spell

- Execution plan showed nothing wrong
- Fixed a subquery with the wrong join criteria with no success
- When using the form **SELECT * FROM** the query performed much faster

# Cast a spell

What went wrong

- Addressing an HSTORE key returns a TEXT data type
- The developers, instead of doing a cast
- Decided to write a pl/pgsql function for each type they wanted to cast

# Cast a spell

The cast function was something like that

```
CREATE OR REPLACE FUNCTION cast_to_integer(text,hstore)
RETURNS integer AS
$BODY$
        DECLARE
                keyvalue ALIAS FOR $1;
                metastore ALIAS FOR $2;
        BEGIN
                RETURN  (metastore->keyvalue)::integer;
        END;
$BODY$
LANGUAGE plpgsql;
```

# Cast a spell

And the select list was something like that

```
SELECT
        url,
        id,
        rtype,
        page_title,
        cast_to_integer('tw_cnt',metastore) AS tw_cnt,
        cast_to_integer('fb_lk',metastore) AS fb_lk,
        cast_to_integer('yt_vw',metastore) AS yt_vw,
        cast_to_integer('tw_fl',metastore) AS tw_fl,
        cast_to_integer('snp_sb',metastore) AS snp_sb,
        cast_to_integer('avg_vs',metastore) AS avg_vs,
        cast_to_integer('trk_lnk',metastore) AS trk_lnk,
        cast_to_integer('avg_pg',metastore) AS avg_pg,
        cast_to_integer('impr',metastore) AS impr,
        cast_to_integer('frm_pst',metastore) AS frm_pst,
        cast_to_integer('outr',metastore) AS outr,
        .
        .
FROM ............
;
```

# Cast a spell

How it was fixed:

- Have an argument with the developers
- Rewrite the join to perform better
- Get rid of all of the functions in the select list
- The query duration dropped to 10 seconds
- Live long and prosper

# Cast a spell

**RTFM**

- https://www.postgresql.org/docs/9.1/plpgsql.html
- Ask the community for advice before implementing stuff
  - Mailing lists: https://www.postgresql.org/list/
  - IRC on freenode: channel #postgresql
  - Slack: https://postgres-slack.herokuapp.com/
  - Telegram: https://t.me/pg_sql
- Hire a DBA

# Wrap up

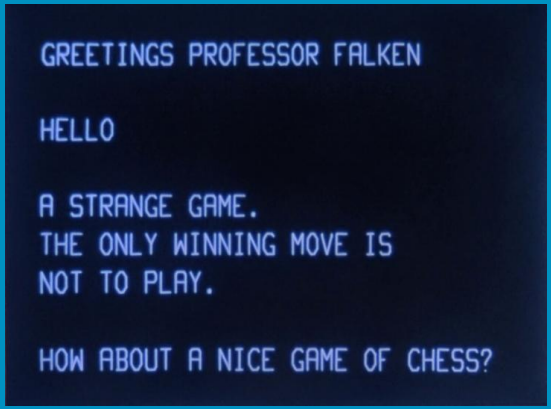



Image source imgur https://imgur.com/gallery/mUeRW0e

# License



# That's all folks!

Thank you for listening!


Any questions?

# RTFM

Federico Campoli

FOSDEM PGDay, Brussels, 02 February 2020