pglogical3 - nasazení a praxe

Matěj Klonfar
`klonfar.matej@ifortuna.cz`

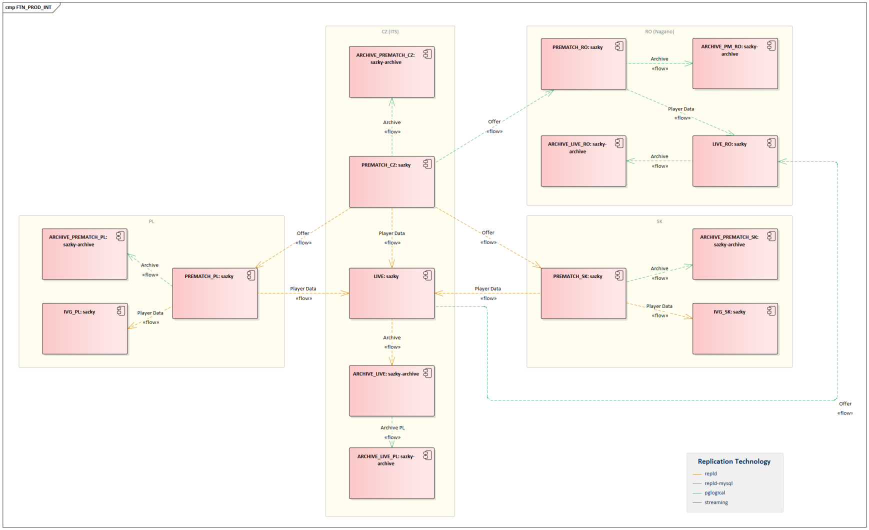
Fortuna Game a. s.

14. února 2019

Prostředí

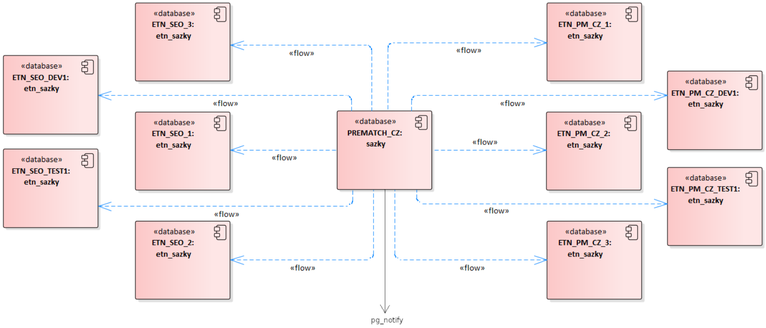
- 4 země
- 1.5 - 3TB / databáze, archivy
- 1-4k tx/s
- PostgreSQL 9.6
- trigger-based replikace

Schéma replikací



Obrázek: Schéma replikací

Schéma replikací II


«database» ETN_SEO_3: etn_sazky
«database» ETN_PM_CZ_1: etn_sazky
«database» ETN_SEO_DEV1: etn_sazky
«database» ETN_PM_CZ_DEV1: etn_sazky
«database» ETN_SEO_1: etn_sazky
«database» PREMATCH_CZ: sazky
«database» ETN_PM_CZ_2: etn_sazky
«database» ETN_SEO_TEST1: etn_sazky
«database» ETN_PM_CZ_TEST1: etn_sazky
«database» ETN_SEO_2: etn_sazky
«database» ETN_PM_CZ_3: etn_sazky
«flow»
pg_notify

Obrázek: Schéma replikací II

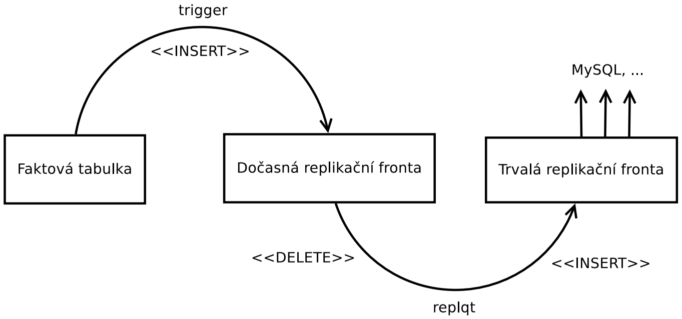
pglogical3

- rozšíření PostgreSQL 9.4+
- logická replikace pomocí logického dekódování[1]
- modulární architektura - HeapWriter, SPIWriter, RabbitMQWriter, KafkaWriter
- podpora pro "old-style" partitioning (dědičnost, check constraints) na straně subscribera - SPIWriter

pglogical3

- Stejný model jako u nativní logické replikace, lehce jiná terminologie
- Replikační set == PUBLICATION
- Subscription

Trigger-based replikace



Trigger-based replikace II

- duplikace dat pokud změna patří do více replikačních kanálů
- cca 10-30GB denně v trvalé replikační frontě, drží se 3 dny pro případ zpoždění. Dohromady přes celou produkci cca 300GB jen replikačních dat

pglogical3 vs. nativní logická replikace

- Podmíněná replikace (row filter)
- Replikace vybraných sloupců
- Řešení (některých) konfliktů
- (Re)synchronizace dat
- Replikace DDL, TRUNCATE
- forward_origins
- strip_origins
- RabbitMQWriter

pglogical3 vs. nativní logická replikace

- Replikace vybraných sloupců

```
# \d pglogical.replication_set_table
            Table pglogical.replication_set_table
     Column     |     Type     | Collation | Nullable | Default
----------------+--------------+-----------+----------+---------
 set_id         | oid          |           | not null |
 set_reloid     | regclass     |           | not null |
 set_att_list   | text[]       |           |          |
 set_row_filter | pg_node_tree |           |          |
```

Row filter - provider

- Podmíněná replikace (row filter)

```
# select set_name, set_reloid, pg_get_expr(set_row_filter, set_reloid)
  from pglogical.replication_set_table
  join pglogical.replication_set using (set_id)
  where set_row_filter is not null;

     set_name       |     set_reloid      |                     pg_get_expr
--------------------+---------------------+------------------------------------------------------
 financial_records  | tab_klient_penize   | ((typ = ANY (ARRAY[7, 9])) AND (puvod = 6))
 bonus_records      | tab_klient_penize   | ((typ = ANY (ARRAY[15, 54])) AND (puvod = 6))
```

Row filter - subscriber

- session_replication_role == replica
- BEFORE TRIGGERS
- ENABLE REPLICA, ENABLE ALWAYS

Detekce změn

- Replikuje se vždy celý řádek
- V row filteru není dostupné OLD a NEW

```
CREATE FUNCTION collect_changes() RETURNS TRIGGER AS
BEGIN
    -- list of replicated columns
    _att_list =
        (
            SELECT set_att_list
            FROM pglogical.replication_set_table
            JOIN pglogical.replication_set USING (set_id)
            WHERE set_name = TG_ARGV[0]
              AND set_reloid = TG_RELID
        );

        -- all columns but the "updated_columns" one
        NEW.updated_columns =
                hstore(akeys(hstore(NEW)), NULL) - ARRAY['updated_columns'];
    END IF;

        RETURN NEW;
END;

CREATE FUNCTION apply_changes() RETURNS trigger AS
BEGIN
        IF NEW.updated_columns IS NOT NULL THEN
                NEW = (OLD #= (slice(hstore(NEW), akeys(NEW.updated_columns))));
        END IF;

        RETURN NEW;
END;
```

Konflikty

Pokud je uzel k odběru dat z více providerů, případně v případě lokálních změn.

```
postgresql.conf:

        pglogical.conflict_resolution = {
                error
                | apply_remote
                | keep_local
                | last_update_wins
                | first_update_wins
        }
```

- HeapWriter
- SPIWriter - jen error

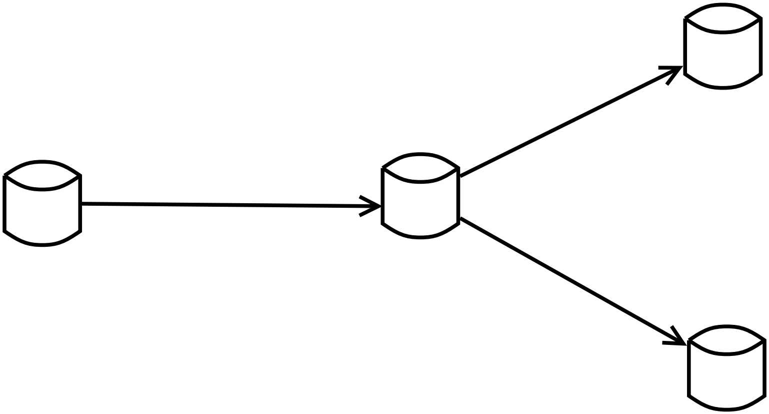
Synchronize tabulek/subscripcí

- Při vytvoření subscripce - synchronizace všech zahranutých tabulek (COPY)
- Ručně konkrétní tabulku - děla TRUNCATE, selže pokud na tabulku vede reference

```
pglogical.alter_subscription_resynchronize_table(
        subscription_name name,
        relation regclass,
        truncate BOOLEAN DEFAULT TRUE
)
```

Replication origin

- forward_origins = {all|}
- strip_origins



RabbitMQWriter

- message format

```
{"action":"U","table":["public","tbl1"],"data":{"a":11,"b":"bar"},"key":{"a":1}}
```

- routing key

```
SELECT pglogical.create_subscription(
        subscription_name := 'rabbitmqsub',
        provider_dsn := 'host=providerhost␣port=5432␣dbname=src',
        writer := 'RabbitMQWriter',
        create_slot := TRUE,
        slot_name := 'pgl_dbname_nodename_subname'
        writer_options := {
                "host", "localhost",
                "port", "5672",
                "exchange", "some-exchange",
                "routing_key", "somekey" -- v nove verzi mozne urcit dynamicky
        }
);
```

Konfigurace

- postgresql.conf

```
shared_preload_libraries = 'pglogical'
wal_level = 'logical'        # on provider node only
max_worker_processes = 10    # one per database needed on provider node
max_replication_slots = 10   # one per node needed on provider node
max_wal_senders = 10         # one per node needed on provider node
track_commit_timestamp = on  # on provider node only
                             # needed for last/first update wins conflict resolution

wal_sender_timeout
wal_reciever_timeout
```

- pg_hba.conf - all neobsahuje replication

Monitoring

- pglogical.show_subscription_status()
- pglogical.show_subscription_clock_drift()
- pglogical.worker_error
- pg_stat_replication
- pg_replication_slots

Zkušenosti

- UNIQUE CONSTRAINTS

  > *If more than one upstream is configured or the downstream accepts local writes then only one UNIQUE index should be present on downstream replicated tables. Conflict resolution can only use one index at a time so conflicting rows may ERROR if a row satisfies the PRIMARY KEY but violates a UNIQUE constraint on the downstream side.[2]*

- SPIWriter - duplicity PK/REPLICA IDENTITY
- RAISE ERROR/EXCEPTION
- dlouhé transakce

Když se to zastaví

- fix it, fix it, fix it

```
pglogical.worker_error

-- logfile
LOG:  starting receiver for subscription archive
ERROR:  duplicate key value violates unique constraint "tab_bonus_internet_idx3"
DETAIL:  Key (id_klient, typ)=(696861, 11) already exists.
CONTEXT:  SQL statement INSERT INTO public.tab_bonus_internet (id, id_klient, datum,
        status, vsazeno, id_klient_penize, ulozeno, id_user, typ, poznamka, castka)
        VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11);
        while consuming 'I' message from receiver
ERROR:  writer has died
FATAL:  writer has died
```

- skip it

```
SELECT * FROM pg_logical_slot_get_changes(
        'slotname', NULL,1,
        'min_proto_version', '1', 'max_proto_version', '1',
        'pglogical.replication_set_names', 'archive_all,archive_nodel',
        'startup_params_format','1', 'proto_format', 'json','skip-empty-xacts', '1');

-- version 11
pg_replication_slot_advance()
```

Lessons learned

- Konflikty
- Replikace nechtěných změn
- RabbitMQWriter

  > *By default, when the RabbitMQ server uses above 40% of the available RAM, it raises a memory alarm and blocks all connections that are publishing messages [3]*

- Takto zablokovaná konexe/background worker nelze ze strany Postgresu ukončit!
- Pozor na defaultní konfiguraci
  - max_channels - 2047 ve verzi RabbitMQ 3.7.5, pglogical: unlimited
  - heartbeat

    > *The broker and client will attempt to negotiate heartbeats by default. When both values are non-0, the lower of the requested values will be used. If one side uses a zero value (attempts to disable heartbeats) but the other does not, the non-zero value will be used.[3]*

Lessons learned - bad way

Někdo si tím projít musí, takže už nemusíte vy

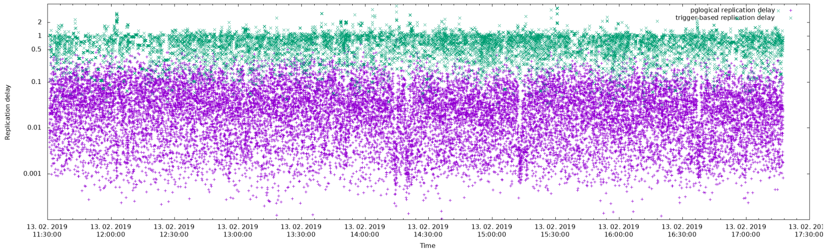
- NOTIFY publishing
- Záhadné updaty

```
CONFLICT: remote INSERT on relation public.tab_polozka (local index pk_tabpolozka).
        Resolution: apply_remote.
DETAIL: existing local tuple id_polozka[int4]:17518064,
        remote tuple id_polozka[int4]:17518065
```

- ... což nakonec vedlo k:

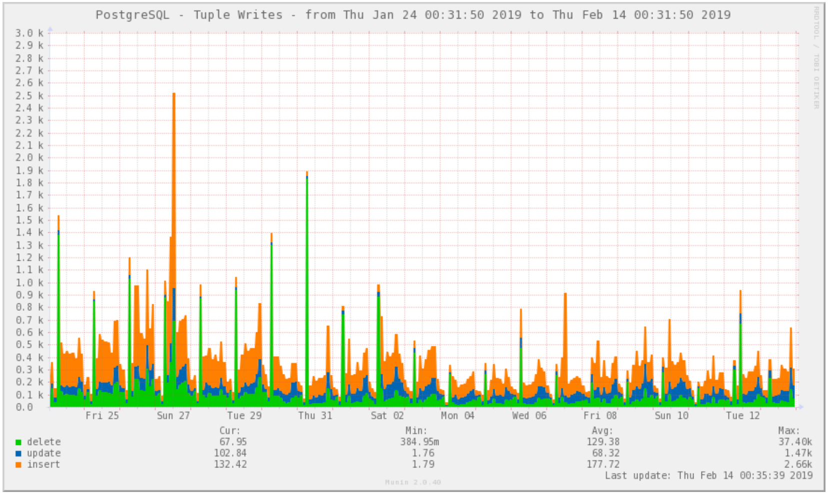
```
LOG: worker process: pglogical writer 6782737:1918997305 (PID 88011) was terminated by
signal 11: Segmentation fault
PANIC: invalid max offset number
```

Zpoždění



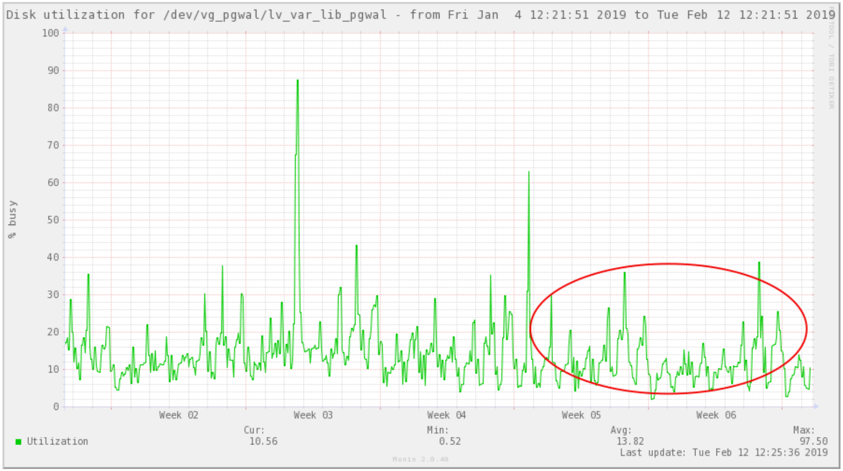
Replikace nabídkových dat, HeapWriter.

Tuple writes

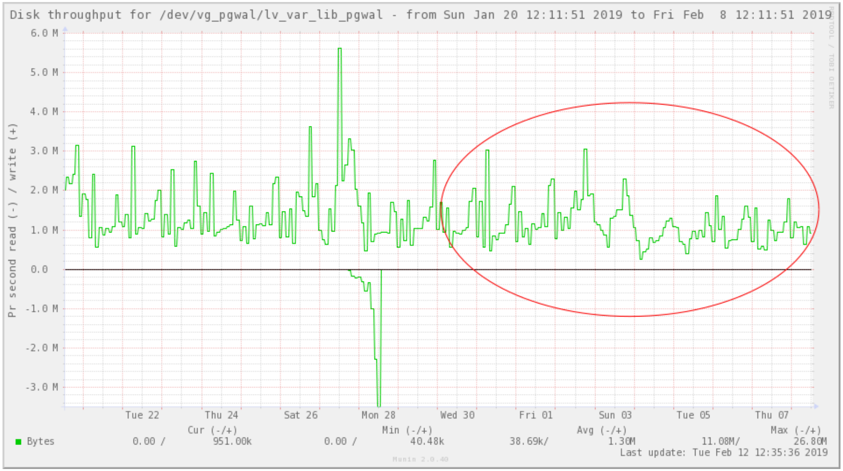


```
                 Cur:          Min:          Avg:          Max:
🟩 delete       67.95       384.95m        129.38        37.40k
🟦 update      102.84          1.76         68.32         1.47k
🟧 insert      132.42          1.79        177.72         2.66k
                                    Last update: Thu Feb 14 00:35:39 2019
```

Disk utilization

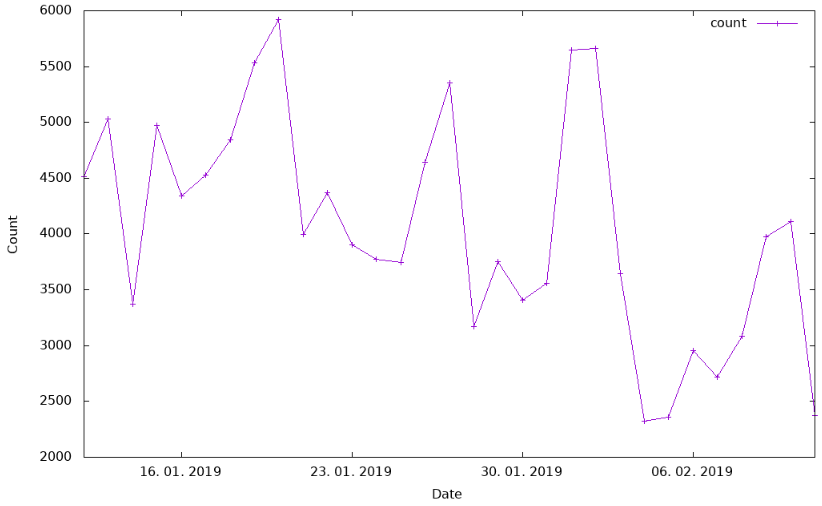


Disk throughput



```
                 Cur (-/+)                 Min (-/+)                 Avg (-/+)                 Max (-/+)
■ Bytes      0.00 /     951.00k       0.00 /     40.48k      38.69k/     1.30M        11.08M/      26.80M
                                                                         Last update: Tue Feb 12 12:35:36 2019
```

WAL log segments



Reference

[1] URL: https://www.postgresql.org/docs/9.6/static/logicaldecoding-explanation.html.

[2] URL: https://www.2ndquadrant.com/en/resources/pglogical/pglogical-docs/.

[3] URL: https://www.rabbitmq.com/.