Distributed postgres. DTM, MultiMaster

Stas Kelvich, github.com/kelvich

Research about distributed postgres in PgPro

- Distributed Transaction Manager;
- Multimaster replication.

Distributed Transaction Manager

Distributed transaction manager is a communication protocol along with specific algorithm that allows to create multi-node transactions with proper isolation.

Can be used with:

- app-based sharding;
- postgres_fdw;
- pg_shard;
- multimaster.

Distributed Transaction Manager

Small example. Imagine following architechture:

shard1: local table t

shard2: local table t

nodeX:

```
CREATE FOREIGN TABLE t_fdw1()
inherits (t) server shard1 options(table_name t)
CREATE FOREIGN TABLE t_fdw2()
inherits (t) server shard2 options(table_name t)
```

Distributed Transaction Manager

Two threads with transactions:

```
begin;
update t set v = v - 1 where u=%d;
update t set v = v + 1 where u=%d;
commit;

select sum(v) from t;
```

Distributed Transaction Manager

Without dtm extension:

```
> cd xtmbench
> make
> ./xtmbench -c h́ost=192.168.99.100 user=xtm-́n 300
10000 accounts inserted
Total=-1
Total=0
Total=1
Total=0
Total=1
...
3300 tx finished.
```

Distributed Transaction Manager

```
shared_preload_libraries = pg_tsdtm

CREATE EXTENSION pg_tsdtm;

> cd xtmbench
> make
> ./xtmbench -c h́ost=192.168.99.100 user=xtm-́n 300
10000 accounts inserted
3300 tx finished.
>
```

Multimaster

Our implementation:

- Built on top of pg_logical;
- Make use of tsDTM;
- Pool of workers for tx replay;
- Raft-based storage for dealing with failures and distributed deadlock detection.

Multimaster

Our implementation:

- Approximately half of a speed of standalone postgres;
- Same speed for reads;
- Deals with nodes autorecovery;
- Deals with network partitions (debugging right now).
- Can work as an extension (if community accept XTM API in core).