Transactions Across Multiple Datastores

pgConf.eu Tallinn 2016

Cédric Villemain cedric@2ndQuadrant.fr

Nov. 3, 2016

Cédric Villemain

2ndQuadrant®
PostgreSQL

PostgreSQL Expertise & Development
Training
24x7 Support & RDBA

2ndQuadrant

PostgreSQL Platinum Sponsor

- **9.2**, group commit improvment
- **9.3**, checksum on data pages
- **9.4**, logical decoding
- **9.5**, track_commit_timestamp
- **9.6**, 2PC performances improvment
- **9.4 - NEXT**, Bi-Directionnal Replication

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Datastore

- Filesystem
- IMAP
- LDAP
- Key-Value Store
- Relationnal Database

It is a way of «*storing bytes*»

Transactions

ACID

Atomicity **C**onsistency **I**solation **D**urability

not **BASE**

Basically **A**vailable, **S**oft state, **E**ventual consistency

Transactions

ACID

Atomicity **C**onsistency **I**solation **D**urability

not **BASE**

Basically **A**vailable, **S**oft state, **E**ventual consistency

Transactions

ACID

Atomicity **C**onsistency Isolation **D**urability

not **BASE**

Basically **A**vailable, **S**oft state, **E**ventual consistency

Transactions

ACID

Atomicity **C**onsistency **I**solation **D**urability

not **BASE**

Basically **A**vailable, **S**oft state, **E**ventual consistency

Transactions

ACID

Atomicity **C**onsistency **I**solation **D**urability

not **BASE**

Basically **A**vailable, **S**oft state, **E**ventual consistency

Transactions

ACID

Atomicity **C**onsistency **I**solation **D**urability

not **BASE**

Basically **A**vailable, **S**oft state, **E**ventual consistency

Accross

Transaction on each datastore

Global Transaction accross multiple datastores

Accross

Transaction on each datastore

Global Transaction accross multiple datastores

PATTERNS

Complex Transaction

I want to check warehouse, check for payment, do the sale, validate payment, send items, and update stocks.

Digital Safe

I must have double transaction: original plus legal one on a distinct safe.
Online gambling website look at that.

Distinct Datastores

Distinct datastore per activity, optionnaly with distinct technology.

TECHS

PostgreSQL

Foreign Data Wrappers allows to commit transactions on distinct datastores.

PostgreSQL

```
CREATE TABLE demo (
  id serial primary key, d timestamptz, t text, i int
);
CREATE FOREIGN TABLE f_demo (
  id serial, d timestamptz, t text, i int
) SERVER loopback OPTIONS (table_name 'demo');

BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE ;
TABLE demo;
TABLE f_demo;
INSERT INTO f_demo values (1, now(), 'remote insert', 1);
INSERT INTO demo values (2, now(), 'local insert', 2);
COMMIT;

ERROR:  could not serialize access due to read/write
        dependencies among transactions
```

PostgreSQL

```
CREATE TABLE demo (
  id serial primary key, d timestamptz, t text, i int
);
CREATE FOREIGN TABLE f_demo (
  id serial, d timestamptz, t text, i int
) SERVER loopback OPTIONS (table_name 'demo');

BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE ;
TABLE demo;
TABLE f_demo;
INSERT INTO f_demo values (1, now(), 'remote insert', 1);
INSERT INTO demo values (2, now(), 'local insert', 2);
COMMIT;
```

```
ERROR:  could not serialize access due to read/write
        dependencies among transactions
```

PostgreSQL

```
TABLE demo;
 id |                 d                  |       t        | i
----+------------------------------------+----------------+---
  1 | 2016-11-02 18:59:13.207825+01 | remote insert | 1
```

NEXT: add a 2 Phase Commit to rollback remote transactions ? (at least)

PostgreSQL

```
TABLE demo;
 id |                  d                  |       t        | i
----+-------------------------------------+----------------+---
  1 | 2016-11-02 18:59:13.207825+01 | remote insert | 1
```

NEXT: add a 2 Phase Commit to rollback remote transactions ? (at least)

Logical Replication

Distributed/Spread accros several datastores, **BASE** by default.

Use **synchronous replication** (including replay) to reach **ACID**!

Logical Replication

Distributed/Spread accros several datastores, **BASE** by default.

Use **synchronous replication** (including replay) to reach **ACID**!

Compensation or Rollback transaction

Those solutions are complex and most of the time require a *Transaction Compensation* instead of a Rollback.

Two Phase Commit

Problem: be sure as much as possible that commit will be successfull !

Solution: two-phase commit protocol

Transaction Manager

```
START
DEFINE XIDs
1/ PREPARE
...
2/ COMMIT/ROLLBACK
```

Transaction A

```
BEGIN
...SOME SQL...
PREPARE TRANSACTION
...
COMMIT PREPARED
```

Transaction B

```
BEGIN
...SOME SQL...
PREPARE TRANSACTION
...
COMMIT PREPARED
```

Two Phase Commit

Problem: be sure as much as possible that commit will be successfull !

Solution: two-phase commit protocol

Transaction Manager

```
START
DEFINE XIDs
1/ PREPARE
...
2/ COMMIT/ROLLBACK
```

Transaction A

```
BEGIN
...SOME SQL...
PREPARE TRANSACTION
...
COMMIT PREPARED
```

Transaction B

```
BEGIN
...SOME SQL...
PREPARE TRANSACTION
...
COMMIT PREPARED
```

X/Open DTP Distributed Transaction Processing

X/Open XA eXtended Architecture

XA propose 3 layers to handle Global Transactions

- APPlication
- Resource Manager
- Transaction Manager

X/Open DTP Distributed Transaction Processing

X/Open XA eXtended Architecture

XA propose 3 layers to handle Global Transactions

- APPlication
- Resource Manager
- Transaction Manager

X/Open DTP Distributed Transaction Processing

X/Open XA eXtended Architecture

XA propose 3 layers to handle Global Transactions

- APPlication
- Resource Manager
- Transaction Manager

open/XA Features

- Asynchronous Operations
- Suspend Transaction
- Migrate Transaction
- Join Transaction

open/XA Features

- Asynchronous Operations
- Suspend Transaction
- Migrate Transaction
- Join Transaction

open/XA Features

- Asynchronous Operations
- Suspend Transaction
- Migrate Transaction
- Join Transaction

open/XA Features

- Asynchronous Operations
- Suspend Transaction
- Migrate Transaction
- Join Transaction

open/XA Features

Asynchronous Operation

Purpose is to have Transaction Manager and, if relevant, APPlication be able to do other processing instead of just waiting.

PostgreSQL offers asynchronous operations with libpq

https://www.postgresql.org/docs/current/static/libpq-async.html

open/XA Features

Asynchronous Operation

Purpose is to have Transaction Manager and, if relevant, APPlication be able to do other processing instead of just waiting.

PostgreSQL offers asynchronous operations with libpq

https://www.postgresql.org/docs/current/static/libpq-async.html

open/XA Features

Suspend Transaction

It is ... just like a «pause». But you can disconnect.

Partial support from PostgreSQL («idle in transaction» if you prefer)

Option to configure a timeout: *idle_in_transaction_session_timeout*

PgBouncer

A patch for pgbouncer complete this feature by providing the client the ability to disconnect and reconnect later to resume its transaction.

open/XA Features

Suspend Transaction

It is ... just like a «pause». But you can disconnect.

Partial support from PostgreSQL («idle in transaction» if you prefer)

Option to configure a timeout: *idle_in_transaction_session_timeout*

PgBouncer

A patch for pgbouncer complete this feature by providing the client the ability to disconnect and reconnect later to resume its transaction.

open/XA Features

Suspend Transaction

It is ... just like a «pause». But you can disconnect.

Partial support from PostgreSQL («idle in transaction» if you prefer)

Option to configure a timeout: *idle_in_transaction_session_timeout*

PgBouncer

A patch for pgbouncer complete this feature by providing the client the ability to disconnect and reconnect later to resume its transaction.

open/XA Features

Migrate transaction

It is an interesting feature:
TM can complete work (for example just issue a PREPARE TRANSACTION on behalf of the APP when APP asks it to do so)
And it allows APP to resume the transaction with another transaction branch (that is: another «client»)

PgBouncer

This is not natively supported by PostgreSQL but a patch to pgbouncer allows that.

open/XA Features

Migrate transaction

It is an interesting feature:
TM can complete work (for example just issue a PREPARE TRANSACTION on behalf of the APP when APP asks it to do so)
And it allows APP to resume the transaction with another transaction branch (that is: another «client»)

PgBouncer

This is not natively supported by PostgreSQL but a patch to pgbouncer allows that.

open/XA Features

Join transaction

It is also an interesting feature: several clients can issue queries on the same transaction, and allowing the RM to serialize them the way it wants.

PgBouncer

This is not natively supported by PostgreSQL but a patch to pgbouncer allows that.

open/XA Features

Join transaction

It is also an interesting feature: several clients can issue queries on the same transaction, and allowing the RM to serialize them the way it wants.

PgBouncer

This is not natively supported by PostgreSQL but a patch to pgbouncer allows that.

open/XA Features

Recover transactions

This is about prepared transaction, and also already commited transaction.

PostgreSQL offers both options

one is easy:

```
SELECT transaction, gid, prepared, owner, database
FROM pg_prepared_xacts
```

this other is less convenient to use but is offered by logical slot and WAL exploration (pg_xlogdump is an example).

open/XA Features

Recover transactions

This is about prepared transaction, and also already commited transaction.

PostgreSQL offers both options

one is easy:

```
SELECT transaction, gid, prepared, owner, database
FROM pg_prepared_xacts
```

this other is less convenient to use but is offered by logical slot and WAL exploration (pg_xlogdump is an example).

open/XA Features

Recover transactions

This is about prepared transaction, and also already commited transaction.

PostgreSQL offers both options

one is easy:

```
SELECT transaction, gid, prepared, owner, database
FROM pg_prepared_xacts
```

this other is less convenient to use but is offered by logical slot and WAL exploration (pg_xlogdump is an example).

PostgreSQL driver for XA

PostgreSQL JDBC include a support for XA (partial).

libpqXA is a C driver extending libpq for supporting open/XA. It includes all of the required API:

```
int (*xa_open_entry)(char *, int, long);
int (*xa_close_entry)(char *, int, long);
int (*xa_start_entry)(XID *, int, long);
int (*xa_end_entry)(XID *, int, long);
int (*xa_rollback_entry)(XID *, int, long);
int (*xa_prepare_entry)(XID *, int, long);
int (*xa_commit_entry)(XID *, int, long);
int (*xa_recover_entry)(XID *, long, int,
int (*xa_forget_entry)(XID *, int, long);
int (*xa_complete_entry)(int *, int *, int, long);
```

PgBouncer still required to handle advanced features.

PostgreSQL driver for XA

PostgreSQL JDBC include a support for XA (partial).

libpqXA is a C driver extending libpq for supporting open/XA. It includes all of the required API:

```
int (*xa_open_entry)(char *, int, long);
int (*xa_close_entry)(char *, int, long);
int (*xa_start_entry)(XID *, int, long);
int (*xa_end_entry)(XID *, int, long);
int (*xa_rollback_entry)(XID *, int, long);
int (*xa_prepare_entry)(XID *, int, long);
int (*xa_commit_entry)(XID *, int, long);
int (*xa_recover_entry)(XID *, long, int,
int (*xa_forget_entry)(XID *, int, long);
int (*xa_complete_entry)(int *, int *, int, long);
```

PgBouncer still required to handle advanced features.

Summary

open/XA is a good standard for managing ACID transaction on multiple datastores.

PostgreSQL has 2PC in-core. Missing features are covered by external tool pgBouncer.

libpqXA is an extension to *libpq* for supporting open/XA in C and C++.

The complete solution including libpqXA and improved pgBouncer is in use at Bibliothèque Nationnale de France (http://www.bnf.fr/) who sponsored the open-source development.

Summary

open/XA is a good standard for managing ACID transaction on multiple datastores.
PostgreSQL has 2PC in-core. Missing features are covered by external tool pgBouncer.
libpqXA is an extension to *libpq* for supporting open/XA in C and C++.

The complete solution including libpqXA and improved pgBouncer is in use at Bibliothèque Nationnale de France (http://www.bnf.fr/) who sponsored the open-source development.

Summary

open/XA is a good standard for managing ACID transaction on multiple datastores.
PostgreSQL has 2PC in-core. Missing features are covered by external tool pgBouncer.
libpqXA is an extension to *libpq* for supporting open/XA in C and C++.

The complete solution including libpqXA and improved pgBouncer is in use at Bibliothèque Nationnale de France (http://www.bnf.fr/) who sponsored the open-source development.

Summary

open/XA is a good standard for managing ACID transaction on multiple datastores.
PostgreSQL has 2PC in-core. Missing features are covered by external tool pgBouncer.
libpqXA is an extension to *libpq* for supporting open/XA in C and C++.

The complete solution including libpqXA and improved pgBouncer is in use at Bibliothèque Nationnale de France (http://www.bnf.fr/) who sponsored the open-source development.

Any questions ?

Please ask !