Transactions Across Multiple Datastores

pgConf.eu Tallinn 2016

Cédric Villemain cedric@2ndQuadrant.fr

Nov. 3, 2016
PostgreSQL Expertise & Development
Training
24x7 Support & RDBA
PostgreSQL Platinum Sponsor

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

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

It is a way of «storing bytes»
• 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
- Relational 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  Consistency  Isolation  Durability

not BASE

Basically Available, Soft state, Eventual consistency
Transactions

ACID

Atomicity  Consistency  Isolation  Durability

not BASE

Basically Available, Soft state, Eventual consistency
Transactions

ACID

Atomicity  Consistency  Isolation  Durability

not BASE

Basically Available, Soft state, Eventual consistency
Transactions

ACID

Atomicity  Consistency  Isolation  Durability

not BASE

Basically Available, Soft state, Eventual consistency
Transactions

ACID

Atomicity  Consistency  Isolation  Durability

not BASE

Basically Available, Soft state, Eventual consistency
Transactions

ACID

Atomicity  Consistency  Isolation  Durability

not BASE

Basically Available, Soft state, Eventual consistency
Accross

Transaction on each datastore

Global Transaction across multiple datastores
Transaction on each datastore

Global Transaction accross multiple datastores
PATTERNS
I want to check warehouse, check for payment, do the sale, validate payment, send items, and update stocks.
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
Foreign Data Wrappers allows to commit transactions on distinct datastores.
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
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
TABLE demo;

<table>
<thead>
<tr>
<th>id</th>
<th>d</th>
<th>t</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2016-11-02 18:59:13.207825+01</td>
<td>remote insert</td>
<td>1</td>
</tr>
</tbody>
</table>

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2016-11-02 18:59:13.207825+01</td>
<td>remote insert</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

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

Distributed/Spread across 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 successful!

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 successful!

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
XA propose 3 layers to handle Global Transactions

- APPlication
- Resource Manager
- Transaction Manager
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
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.
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.
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.
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.
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.
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.
Recover transactions
This is about prepared transaction, and also already committed 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).
Recover transactions
This is about prepared transaction, and also already committed 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).
Recover transactions
This is about prepared transaction, and also already committed transaction.

PostgreSQL offers both options

one is easy:

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

d this other is less convenient to use but is offered by logical slot and WAL exploration (pg_xlogd dump 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:

```c
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:

```c
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.
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.
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.
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!