Logical Replication Internals

Noriyoshi Shinoda

December 6, 2017
Speaker
Noriyoshi Shinoda

– Company
  – Hewlett-Packard Japan, Ltd

– Now
  – System design, tuning, consulting on RDBMS such as PostgreSQL, Oracle Database, Microsoft SQL Server, Vertica, Sybase ASE etc
  – Oracle ACE
  – Writing 15 books for Oracle Database
  – Investigation and verification of open source products

– URL
  – Slideshare
    – https://www.slideshare.net/noriyoshishinoda/
  – Oracle ACE
Agenda

– What is Logical Replication?
– Let’s try!
– Architecture
– Restrictions
– Trouble shooting
What is Logical Replication?
What is Logical Replication?
What is Logical Replication?

– Is
  – PostgreSQL 10 new features
  – Replicate per table
  – Replicate per transaction
  – Updatable replication destination tables
  – Guarantee that the results of the SQL statement will be identical (= Logical)
  – Adopt Publish / Subscribe model
  – ≅ Slony-I

– Is Not
  – Re-execute SQL
  – Physical page format match
What is Logical Replication?

Replication condition

– Conditions for replicable tables
  – Identical schema name
  – Identical table name
  – Identical column name
  – Identical data type
    – Different data types are available if implicit type conversion is possible
Let’s try it!
Let’s try it!
Publisher instance

- Replicate data1 table of pubdb database to subdb database
- Create role with REPLICATION attribute / LOGIN attribute
  - Role connected from Subscriber instance

```
pubdb=# CREATE ROLE repusr1 PASSWORD 'Passw0rd' LOGIN REPLICATION ;
CREATE ROLE
```

- Grant database CREATE privilege to table owner (pubusr1)

```
pubdb=# GRANT CREATE ON DATABASE pubdb TO pubusr1 ;
GRANT
```

- Modify pg_hba.conf file

<table>
<thead>
<tr>
<th>#</th>
<th>TYPE</th>
<th>DATABASE</th>
<th>USER</th>
<th>ADDRESS</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>host</td>
<td>pubdb</td>
<td>repusr1</td>
<td>192.168.1.100/32</td>
<td>md5</td>
</tr>
</tbody>
</table>

- 'DATABASE = replication' item not required
Let’s try it!
Publisher instance

- Modify postgresql.conf file

```bash
wal_level = logical
```
Let’s try it!
Publisher database

– Create table for replication

```
pubdb=> CREATE TABLE data1 (c1 INT PRIMARY KEY, c2 VARCHAR(5)) ;
CREATE TABLE
```

– Grant SELECT permission to the table for connected user

```
pubdb=> GRANT SELECT ON data1 TO repusr1 ;
GRANT
```

– Create the PUBLICATION object

```
pubdb=> CREATE PUBLICATION pub1 FOR TABLE data1 ;
CREATE PUBLICATION
```
Let’s try it!
Subscriber database

– Create table for replication

```
subdb=> CREATE TABLE data1 (c1 INT PRIMARY KEY, c2 VARCHAR(5)) ;
CREATE TABLE
```

– Create the SUBSCRIPTION object (require SUPERUSER)

```
subdb=# CREATE SUBSCRIPTION sub1 CONNECTION
  'host=pubhost1 dbname=pubdb user=repusr1 password=Passw0rd'
  PUBLICATION pub1 ;
CREATE SUBSCRIPTION
```
Let’s try it!
Confirmation

– Publisher instance

```sql
pubdb=# SELECT application_name, state, sync_state FROM pg_stat_replication;
 application_name | state   | sync_state
-------------------+---------|-------------
 sub1             | streaming | async
(1 row)
```

– Subscriber instance

```sql
subdb=> SELECT subname, received_lsn FROM pg_stat_subscription;
 submodule | received_lsn
-----------+-----------------
 sub1      | 0/1C8FF738
```
Architecture
Architecture
Components

Publisher Instance

Publisher Database

PUBLICATION

Tables

Logical Replication Slot

pgoutput plugin

WAL

decode

wal sender

message

Subscriber Instance

Subscriber Database

SUBSCRIPTION

Tables

logical replication launcher

logical replication worker
Architecture
Processes

- wal sender process \{user\} \{client ip\} \{state\}
  - On PUBLICATION instance
  - Send decoded WAL messages
  - Launch for each connection from SUBSCRIPTION

- bgworker: logical replication launcher
  - On PUBLICATION / SUBSCRIPTION instance
  - Start the logical replication worker process

- bgworker: logical replication worker for subscription \{oid\}
  - On SUBSCRIPTION instance
  - Connect to wal sender process
  - Receive decoded WAL message and update the table
  - Launch for each SUBSCRIPTION
Architecture
Behavior when PUBLICATION is created

- Store information in the `pg_publication` catalog
  - PUBLICATION name
  - Which DML to replicate (INSERT / UPDATE / DELETE)
  - Whether to cover all tables (FOR ALL TABLES clause)

- Store information in the `pg_publication_rel` catalog
  - Table OID and PUBLICATION OID
  - If FOR ALL TABLES clause is specified, nothing is stored in this catalog
  - Table name can be referenced from `pg_publication_tables` catalog
Architecture
Behavior when SUBSCRIPTION is created

– Store information in `pg_subscription` catalog
  – Connected instance information
  – Replication Slot name
  – Connected PUBLICATION name
  – Synchronous / Asynchronous replication attribute

– Connect to PUBLICATION instance
  – Check REPLICATION attribute of connected user
  – Create Logical Replication Slot (default name is SUBSCRIPTION name)
  – Do not check whether PUBLICATION exists
  – Register replication target tables in `pg_subscription_rel` catalog

– Synchronize initial snapshot
  – Executed asynchronously
  – Existing data on SUBSCRIPTION instance will not be deleted
Architecture
Synchronize initial snapshot

Publisher Instance

Backend process
1. CREATE SUBSCRIPTION

wal sender process
2. Create Replication Slot

3. Create Temporary Replication Slot
   \{sub name\}_\{sub oid\}_sync_\{rel oid\}

4. COPY TO STDOUT statement

Subscriber Instance

Backend process

1. CREATE SUBSCRIPTION

logical replication worker

Decoded Initial Data

Decoded Transaction Data
Architecture
Replication Slot

– Logical Replication Slot
  – One-to-one with SUBSCRIPTION
  – Manage sent WAL
  – Provide replication plugin
  – Automatic execution of the following SQL statement when creating SUBSCRIPTION
    \[
    \text{pg\_create\_logical\_replication\_slot( } name, \text{'pgoutput'} )
    \]

– Replication Slot name
  – SUBSCRIPTION name by default
  – Can change CREATE SUBSCRIPTION WITH (slot_name=name) statement
Architecture
pgoutput plugin and message

- pgoutput plugin
  - Default plugin for Logical Replication
  - Used by wal sender process
  - Create Logical Replication messages from WAL
    - Convert character encoding
    - Convert text message from binary data
  - Does not support text output by pg_logical_slot_get_changes function?

- Message
  - All text data
  - Protocol
    - https://www.postgresql.org/docs/10/static/protocol-logicalrep-message-formats.html
  - When executed UPDATE / DELET statement
    - Send data specifying the update target column and updated data
Architecture
Replica Identity

- The condition under which the UPDATE statement / DELETE statement is replicated

<table>
<thead>
<tr>
<th>Table attribute</th>
<th>PUBLICATION</th>
<th>SUBSCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY KEY</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPLICA IDENTITY FULL</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REPLICA IDENTITY USING INDEX + UNIQUE index + NOT NULL</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- UPDATE / DELETE for a table without a primary key or REPLICA IDENTITY setting is an SQL execution error

```
pubdb=> UPDATE data1 SET c2='update' WHERE c1=100 ;
ERROR: cannot update table "data1" because it does not have replica identity and publishes updates
HINT: To enable updating the table, set REPLICA IDENTITY using ALTER TABLE.
```
Architecture
Behavior during DML execution (asynchronous replication)
Architecture
Behavior during DML execution (synchronous replication)
Architecture
Storage

– Logical log snapshot
  – Created when wal sender process receives SIGUSR1 signal from wal writer process
  – Read the WAL file and create snapshot
  – File path
    
    ```
    ${PGDATA}/pg_logical/snapshots/{LSN upper}-{LSN lower}.snap
    ```

– Logical log snapshot when SUBSCRIPTION stop
  – Created if the SUBSCRIPTION instance is stopped and the wal sender process is restarted
  – It is deleted when the transfer is completed
  – File path
    
    ```
    ${PGDATA}/pg_replslot/{SLOT_NAME}/xid-{XID}-lsn-{LSN upper}-{LSN lower}.snap
    ```
Restrictions
Restrictions

SQL statement that can not be replicated

– TRUNCATE statement
– ALTER TABLE statement
– CREATE TABLE statement
  – When executing the CREATE REPLICATION FOR ALL TABLES statement
Restrictions
Objects that can not be replicated

– Only the table can be added to PUBLICATION
  – pg_class.relkind = 'r'

– Objects that can not be replicated
  – MATERIALIZED VIEW
  – INDEX
  – SEQUENCE
  – FOREIGN TABLE
  – UNLOGGED TABLE
  – INHERIT TABLE = OK (ONLY clause)
  – Partition parent table
  – Large Object
Restrictions
SERIAL column / GENERATED AS IDENTITY column

– Internally use SEQUENCE for SERIAL and GENERATED AS IDENTITY columns

– The sequence value is transferred to SUBSCRIPTION but no sequence operation is executed
Restrictions
SERIAL column / GENERATED AS IDENTITY column

**PUBLICATION**

CREATE TABLE t(a SERIAL, b CHAR)

INSERT INTO t(b) VALUES ('A')

INSERT INTO t(b) VALUES ('B')

**SUBSCRIPTION**

CREATE TABLE t(a SERIAL, b CHAR)

INSERT INTO t(b) VALUES ('A')

INSERT INTO t(b) VALUES ('B')

INSERT INTO t(b) VALUES ('C')

(1, 'A')

(1, 'A')

(2, 'B')

(2, 'B')

(1, 'C')
Restrictions
Trigger

– Partial execution
  – Only ROW TRIGGER execute
  – UPDATE OF trigger is not executed
  – STATEMENT TRIGGER is executed only at initial data transfer

– SUBSCRIPTION database
  – Require ALTER TABLE ENABLE ALWAYS or REPLICA TRIGGER statement
  – Bug on 10.0: BEFORE ROW DELETE trigger not issued
    – https://git.postgresql.org/gitweb/?p=postgresql.git;a=commitdiff;h=360fd1a7b2fe779cc9e696b813b12f6a8e83b558
    – Fixed on 10.1
Restrictions
Parameter log_statement

– Even if log_statement = 'all' set, the SQL statement by replication does not be logged
Restrictions
Bidirectional replication

– Bidirectional replication can not be performed
  – It can be configured, however WAL circulates
  – Mutual replication between databases is possible if tables are different
Restrictions
Replication within instance

– Be careful to create the replication within instance
  – The CREATE SUBSCRIPTION statement hangs when trying to simply configure it
  – It is necessary to create SUBSCRIPTION and Replication Slot individually

– Create Logical Replication Slot manually

```
pubdb=# SELECT
    pg_create_logical_replication_slot ('sub1', 'pgoutput');
```

– Create SUBSCRIPTION

```
subdb=# CREATE SUBSCRIPTION sub1 CONNECTION 'dbname=pubdb' WITH
    (CONNECT=off);
```
Restrictions
Combined with Streaming Replication

– Mixable with Streaming Replication environment

– Logical replication from slave instance is not possible
  – Logical Replication Slot can not be created on standby instance
  – Logical Decoding can not be executed on the standby instance
Trouble shooting
Trouble shooting
Log of resource shortage

- max_replication_slots shortage (PUBLICATION)

```
ERROR: could not create replication slot "sub1": ERROR: all replication slots are in use
```

- max_wal_senders shortage (PUBLICATION)

```
FATAL: number of requested standby connections exceeds max_wal_senders (currently 10)
```

- max_logical_replication_workers shortage (SUBSCRIPTION)

```
WARNING: out of logical replication worker slots
HINT: You might need to increase max_logical_replication_workers.
```

- max_worker_processes shortage (SUBSCRIPTION)

```
No log output
```
Trouble shooting

Other log

– Lack of permission at the time of the initial data copy (PUBLICATION)

```
ERROR: could not start initial contents copy for table "public.data1": ERROR: permission denied for relation data1
```

– Execution of DROP SUBSCRIPTION statement (normal)

```
FATAL: terminating logical replication worker due to administrator command
LOG: worker process: logical replication worker for subscription 16408 (PID 77332) exited with exit code 1
```
## Trouble shooting

Conflict pattern and behavior

- Conflict with the behavior that occurs in the SUBSCRIPTION database

<table>
<thead>
<tr>
<th>Conflict pattern</th>
<th>Replication</th>
<th>Log output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Key / Unique Key Violation</td>
<td>Stop</td>
<td>Yes</td>
</tr>
<tr>
<td>CHECK constraint violation</td>
<td>Stop</td>
<td>Yes</td>
</tr>
<tr>
<td>No UPDATE data</td>
<td>Continue</td>
<td>No</td>
</tr>
<tr>
<td>No DETETE data</td>
<td>Continue</td>
<td>No</td>
</tr>
<tr>
<td>No table</td>
<td>Stop</td>
<td>Yes</td>
</tr>
<tr>
<td>No partial columns</td>
<td>Stop</td>
<td>Yes</td>
</tr>
<tr>
<td>Data convert error</td>
<td>Stop</td>
<td>Yes</td>
</tr>
<tr>
<td>Lock table</td>
<td>Wait</td>
<td>No</td>
</tr>
<tr>
<td>Lock updated rows</td>
<td>Wait</td>
<td>No</td>
</tr>
</tbody>
</table>
Trouble shooting
Conflict pattern and behavior

– Impact on applications when conflict occurs
  – Even if conflicts occur, SQL to the table does not be blocked

– Conflict detection
  – Detect from logfile
  – pg_stat_replication.flush_lag / write_lag
  – pg_replication_slots.confirmed_flush_lsn not equal pg_current_wal_lsn()

– Behavior at conflict occurrence in SUBSCRIPTION instance
  – When constraint violation is detected, the logical replication worker process stop
  – Restart after 5 seconds and restart WAL apply
  – Repeat the above until constraint violation is resolved
Trouble shooting
Log at error occurrence

- Log of primary key violation on SUBSCRIPTION side

```
ERROR: duplicate key value violates unique constraint "pk_data1"
DETAIL: Key (c1)=(500) already exists.
LOG: worker process: logical replication worker for subscription 16414 (PID 9644) exited with exit code 1
```

- Log on wal sender timeout on PUBLICATION side

```
LOG: terminating walsender process due to replication timeout
LOG: starting logical decoding for slot "sub1"
DETAIL: streaming transactions committing after 0/5600ED48, reading WAL from 0/5600ED10
```
Trouble shooting

Log at error occurrence

- Memory allocation error during WAL decoding
- Occurred when transferring bytea type
  - Text transformation according to the parameter bytea_output
  - Ensure the memory of the "record size x 2 + 1" bytes by default

- PUBLICATION side log

  ERROR: invalid memory alloc request size 258291203  
  CONTEXT: slot "sub1", output plugin "pgoutput", in the change  
  callback, associated LSN 0/2B2543E8  
  LOG: could not send data to client: Broken pipe  
  FATAL: connection to client lost

- SUBSCRIPTION side log

  ERROR: could not receive data from WAL stream:  
  ERROR: invalid memory alloc request size 258291203  
  CONTEXT: slot "sub1", output plugin "pgoutput", in the change  
  callback, associated LSN 0/2B2543E8
Trouble shooting
Conflict Resolution

– The conflict is not automatically resolved

– There are two ways to solve (on SUBSCRIPTION side)
  – Delete the record where the conflict occurred (resolve the constraint violation)
  – Skipping WAL where conflict has occurred (resolution of constraint violation / elimination of memory shortage)
Trouble shooting
Conflict Resolution

– Skip WAL where conflict occurred
  – Specify the LSN that starts applying WAL

– Confirm current LSN on PUBLICATION side

```
postgres=# SELECT pg_current_wal_lsn() ;
pg_current_wal_lsn
----------------------
0/7200B4F0
(1 row)
```
Trouble shooting
Conflict Resolution

– Confirm external_id on SUBSCRIPTION side

```sql
subdb=# SELECT * FROM pg_replication_origin_status;
local_id | external_id | remote_lsn | local_lsn
--------- | ----------- | ---------- | -----------
       1 | pg_16425 | 0/7200B068 | 0/DA0078E8
(1 row)
```

– `pg_{pg_subscription.oid}` is output to the external_id column

```sql
subdb=# SELECT oid, subname FROM pg_subscription;
oid | subname
-----|---------
16425| sub1
```
Trouble shooting
Conflict Resolution

– Specify the start LSN on SUBSCRIPTION side

```
subdb=# SELECT
    pg_replication_origin_advance ('pg_16425', '0/7200B4F0');
```

```
pg_replication_origin_advance
-------------------------------
(1 row)
```

– Sometimes it gets an error

```
subdb=# SELECT pg_replication_origin_advance('pg_16399',
    '0/82708760');
ERROR: replication origin with OID 1 is already active for PID 5566
```
Thank you

noriyoshi.shinoda@hpe.com