Celebrating its 15th Anniversary: Pgpool-II Past, Present and Future

Part 2: New Features of Pgpool-II 4.0

December 11, 2018
PGConf.ASIA

Bo Peng
SRA OSS, Inc. Japan
pengbo@sraoss.co.jp
New features of Pgpool-II 4.0

- Detecting "false" Primary PostgreSQL Server
- Improvement of Load Balanding
- Enhancement of "SHOW POOL NODES"
- Import PostgreSQL 11 SQL Parser
- Logging Client Messages
Detecting "false" Primary PostgreSQL Server
Detecting "false" Primary PostgreSQL Server (1)

Normal cluster status

Abnormal cluster status

Can't get latest data ...

Promote
Detecting "false" Primary PostgreSQL Server (2)

How can we detect the situation and fix it?

New parameter:
**detach_false_primary**

- Detect the "false" Primary and detach it
- "true" primary : a Primary node which connects to all Standby nodes
  - "false" primary : other than above
- Check the connectivity between Primary and Standby nodes by using "pg_stat_wal_receiver"
- Require PostgreSQL 9.6 or later

```sql
postgres=# show pool_nodes;
<table>
<thead>
<tr>
<th>node_id</th>
<th>hostname</th>
<th>port</th>
<th>status</th>
<th>lb_weight</th>
<th>role</th>
<th>last_status_change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>/tmp</td>
<td>11002</td>
<td>up</td>
<td>0.333333</td>
<td>primary</td>
<td>2018-09-08 23:36:24</td>
</tr>
<tr>
<td>1</td>
<td>/tmp</td>
<td>11003</td>
<td>up</td>
<td>0.333333</td>
<td>standby</td>
<td>2018-09-08 23:36:24</td>
</tr>
<tr>
<td>2</td>
<td>/tmp</td>
<td>11004</td>
<td>down</td>
<td>0.333333</td>
<td>standby</td>
<td>2018-09-08 23:37:05</td>
</tr>
</tbody>
</table>
```

- Detecting "false" Primary PostgreSQL Server (2)
- Copyright © 2018 SRA OSS, Inc. Japan All rights reserved.
- "false" Primary
- detached
Detecting "false" Primary PostgreSQL Server (3)

- Normal status: There's only 1 primary
- There are multiple Primaries and multiple Standbys. In this case, the "false" primary detection doesn't work.

In this case, the "false" primary detection doesn't work.
Improvement of load balancing (1)
disable_load_balance_on_write
Load Balancing
- Load balancing improves system performance by distributing READ queries to any PostgreSQL server

Before Pgpool-II 4.0
- When a WRITE query is executed inside an explicit truncation, subsequent queries will be sent to Primary in order to avoid the replication delay
- Load balancing is performed again in subsequent explicit transactions
Improvement of Load Balancing (1) `disable_load_balance_on_write`

**Problems until 3.7**

- Although this feature maintains consistency of data, it may cause degradation of performance when load balancing can be executed.
- If the delay occurs, the latest data updated on the Primary may not be visible in the Standby.

**`disable_load_balance_on_write`**

- It's possible to control the behavior of load balancing when a WRITE query is executed.
- `off/transaction/trans_transaction/always`
disablen_load_balance_on_write = off

- READ queries are load balanced even if a WRITE query appears

Read queries are **load balanced** even if a WRITE query appears.
disable_load_balance_on_write = transaction

- If a WRITE query appears in an explicit transaction, the subsequent READ queries are not load balanced until the transaction ends.
- Default value.
- The same behavior as Pgpool-II 3.7 or before.
disable_load_balance_on_write = trans_transaction

- If a WRITE query appears in an explicit transaction, the subsequent READ queries are not load balanced until the transaction ends.
- Also, the READ queries are not load balanced in subsequent explicit transactions.
If a WRITE query appears in an explicit transaction, load balancing is not performed until the session ends, regardless of whether it is in an explicit transaction or not.

```sql
BEGIN
SELECT ...
READ
UPDATE ...
WRITE
SELECT ...
READ
COMMIT
SELECT ...
READ
BEGIN
SELECT ...
READ
COMMIT
```

No load balancing until the session ends

Latest data

No load balancing until the session ends

No load balancing until the session ends

No load balancing until the session ends
Improvement of Load Balancing (1) \texttt{disable_load_balance_on_write}

Compatibility with not-clustering-aware applications

- \texttt{disable_load_balance_on_write} = \texttt{always}
- \texttt{disable_load_balance_on_write} = \texttt{trans_transaction}
- \texttt{disable_load_balance_on_write} = \texttt{transaction}
- \texttt{disable_load_balance_on_write} = \texttt{off}

Performance Improvement of Load Balancing (1)
Improvement of load balancing (2)
black_query_pattern_list
User's request

Don't want to perform load balancing by specific SQL

How about /*NO LOAD BALANCE*/ ...

However, the source code of application can't be changed • • •

New parameter: black_query_pattern_list
black_query_pattern_list

- SQLs that matched the specified SQL pattern by this parameter are sent only to the Primary
- Specify a semicolon separated list of SQL patterns
- Allow to use regular expression
- Special characters need to be escaped with "¥"

black_query_pattern_list = '.*table1.*;.*f1.*';

**Note:** If SQL matches both black_query_pattern_list and white_function_list, white_function_list setting is ignored and the SQL should be sent only to the Primary node.
Improvement of load balancing (3)
Specify load balancing weight by load balancing parameters
Send 30% READ query to Primary
- backend_weight0 = 0.3
- backend_weight1 = 0.7

The load balancing weight of primary becomes 0.7 after failover...

Keep the load balancing weight of the Primary in a constant value.
Improvement of Load Balancing (3) Specify load balance weight

**app_name_redirect_preference_list**
Send READ queries to a particular backend node for a particular client application connection

**database_redirect_preference_list**
Send READ queries to a particular backend node for a particular database connection

---

**Pgpool-II 4.0 or later**
Allow to specify load balancing weight

*app_name:primary*(0.3)

- Send 30% READ queries to Primary
- Load balancing weight does not change even after failover

---

**Pgpool-II 3.7 or before**
Can't specify load balancing weight

*app_name:primary*

Send all READ queries to Primary
Enhance SHOW POOL_NODES command
Add "last_status_change" Column
Add "last_status_change" Column

**last_status_change:**
Time when "status" or "role" changed

- unused
- connect_wait
- up
- down
- quarantine

Be useful when failover happens and need to find out the cause in pgpool log file

---

```sql
postgres=# show pool_nodes;
node_id | hostname | port  | status | lb_weight | role | last_status_change
---------+----------+--------+--------+-----------+------+-------------------
0        | /tmp     | 11002  | up     | 0.500000  | primary | 2018-09-10 10:36:24
1        | /tmp     | 11003  | up     | 0.500000  | standby | 2018-09-10 10:36:24

$ pcp_node_info -U pengbo -p 11001 -n 0
/tmp 11002 2 0.500000 up primary 0 2018-09-10 10:37:36
```

```
<table>
<thead>
<tr>
<th>host</th>
<th>port</th>
<th>status</th>
<th>weight</th>
<th>role</th>
<th>replication_delay</th>
<th>last_status_change</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tmp</td>
<td>11002</td>
<td>Connection in use</td>
<td>0</td>
<td>Primary</td>
<td>0</td>
<td>2018-09-10 11:06:18</td>
</tr>
</tbody>
</table>
```
Import PostgreSQL 11 SQL Parser
Pgpool-II has SQL parser

- To accurately parse the SQLs
- To rewrite the query

In every major release, we import the latest version of PostgreSQL's SQL parser to Pgpool-II

Import PostgreSQL 11 parser to Pgpool-II 4.0

- CREATE/ALTER/DROP PROCEDURE
- CALL
- ALTER/DROP ROUTINE
- CREATE INDEX ... INCLUDE ...
- { RANGE | ROWS | GROUPS } frame_start [ frame_exclusion ]
- VACUUM/ANALYZE <table1>, <table2>
Logging Client Messages
Client messages
- Messages from the client to Pgpool-II

3.7 or before
- In order to record the client message, it was necessary to enable the debug messages
- This produces huge amount of debug logs

New parameter: log_client_messages
- If log_client_messages = on, only client messages can be logged without debugging messages

log_client_messages = on

Parse
LOG: Parse message from frontend.
DETAIL: statement: "S2", query: "SELECT 1 FROM pgbench_accounts"

Bind
LOG: Bind message from frontend.
DETAIL: portal: "P1", statement: "S2"
LOG: DB node id: 0 backend pid: 24797 statement: B message

Execute
LOG: Execute message from frontend.
DETAIL: portal: "P1"
LOG: DB node id: 0 backend pid: 24797 statement: Execute: SELECT 1 …

Close
LOG: Close message from frontend.
DETAIL: statement: "S2"
LOG: DB node id: 0 backend pid: 24797 statement: C message
...
Incompatible changes

- Recovery script now accepts 5 parameters
  - $5: node number to be recovered
  - Existing pgpool_recovery() function can be used if you don't care about information provided by the 5th parameter

- Change of parameter name
  - `fail_over_on_backend_error` => `failover_on_backend_error`
  - Now a warning message is displayed when old config name `fail_over_on_backend_error` is used instead of `failover_on_backend_error`

- Allow to specify AES encrypted password in pgpool.conf
Thank you!