管理や運用性がより一層向上！ Pgpool-II 4.2の新機能紹介

2021-01-19

SRA OSS, Inc. 日本支社
彭 博 (ペン ボ)

自己紹介

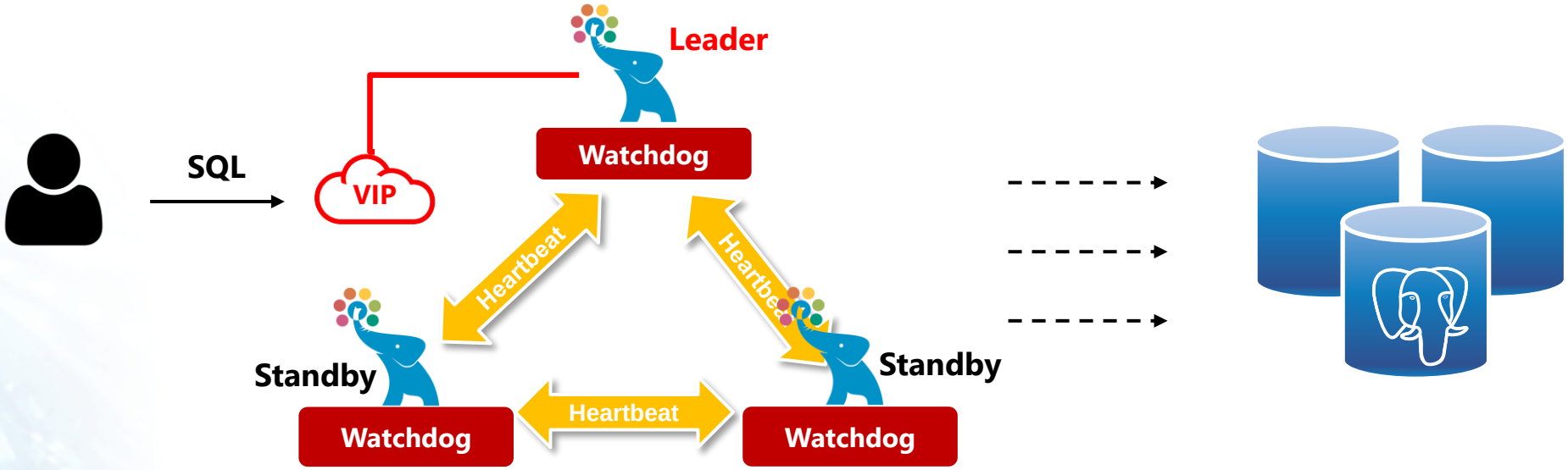
ペン ボ
- 名前： 彭 博 (Bo Peng)
  pengbo@sraoss.co.jp
- 所属：SRA OSS, Inc. 日本支社
  基盤技術グループ
- 職務：
  - OSS技術サポート
  - ミドルウェアの構築
  - PostgreSQL クラスタ管理ツールであるPgpool-II開発者
    - リリースマネジメント
    - 不具合修正
    - 新機能追加
    - PostgreSQLパーサの移植

Pgpool-II 4.2主な新機能・改良

- 設定と管理が容易に
  - 設定パラメータの改善 (watchdog関連、動作モード)
  - ログ収集プロセスの実装
  - ユーザパスワードの一括登録
  - 読み取り/書き込み関数の自動設定
- 複数PostgreSQLの間で読み取り一貫性を保証する新しい動作モードの追加
- LDAP認証のサポート
- 設定ファイルを再読み込みするためのPCPコマンドの追加
- ヘルスチェックおよび発行SQLの統計情報の取得
- ソースコードおよびドキュメントの用語の変更
- PostgreSQL 13のSQLパーサの移植

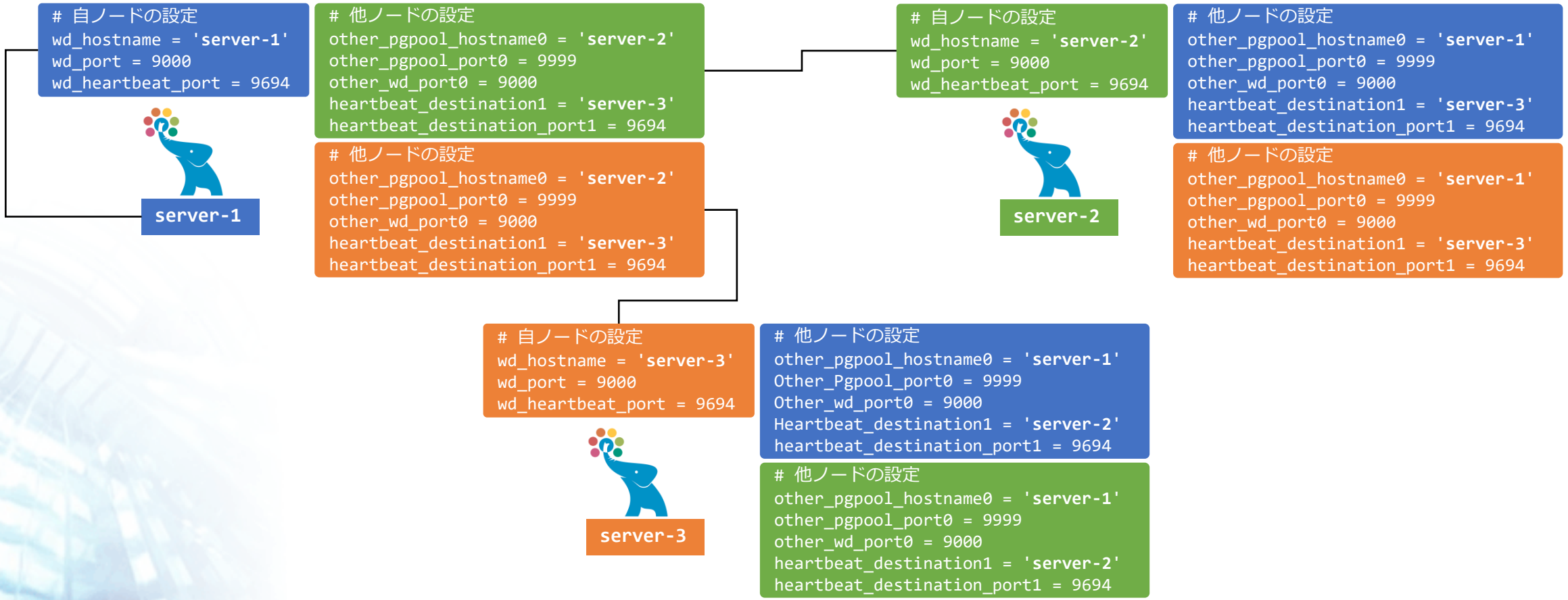
ノード間Watchdog設定の共通化 - Watchdogとは

- Pgpool-IIの単一障害点を回避
- 複数のPgpool-IIがお互いに監視することで、Pgpool-IIを冗長化するための機能
- 定期的に他のPgpool-IIノードにハートビート信号を送信
- Pgpool-IIノードの障害が検出された際に、Watchdogは投票によって新しいリーダーを決定し、切り換える
- 仮想IPの自動切換え
- バックエンドノードのフェイルオーバの動作を制御



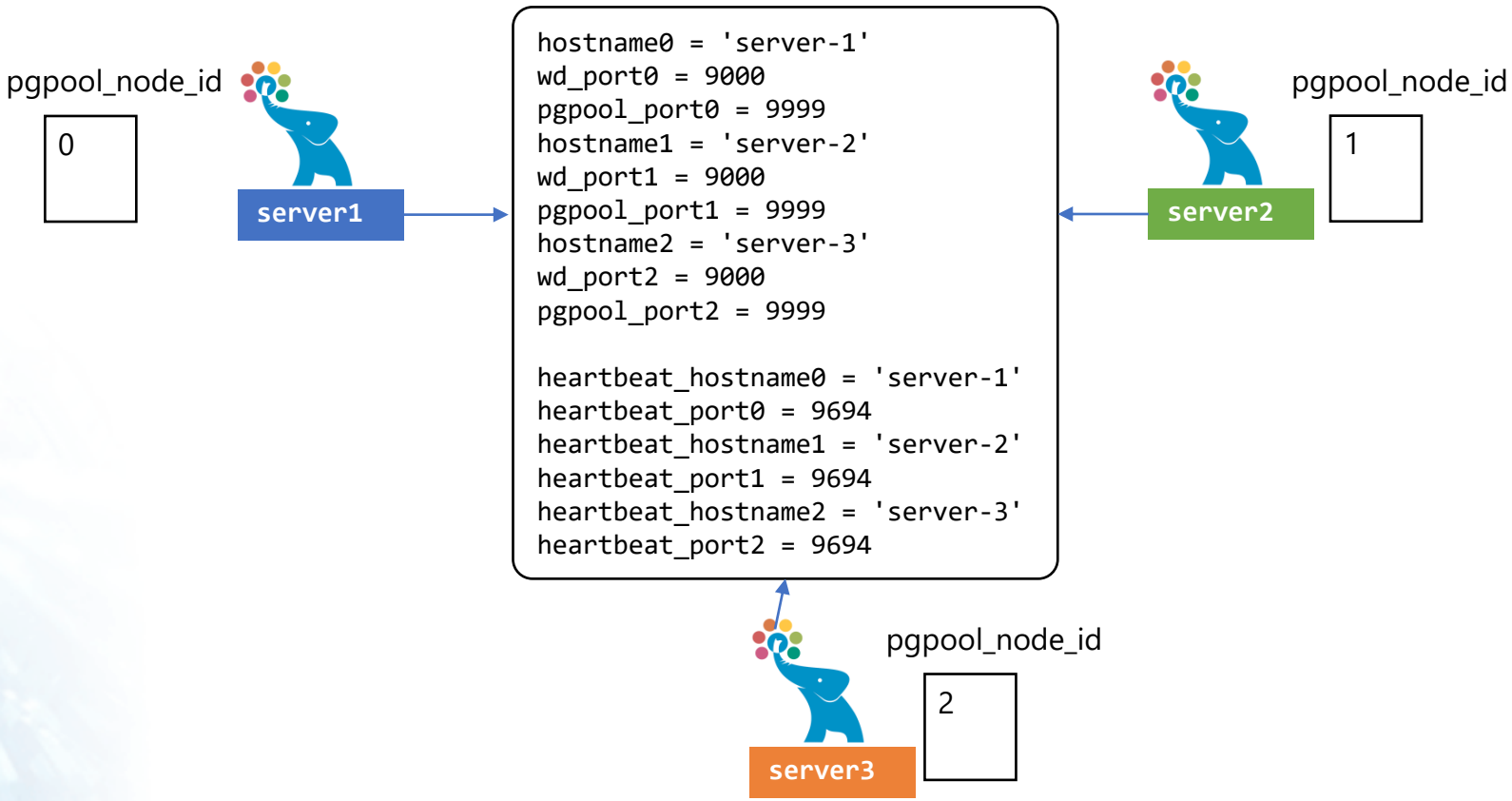
ノード間Watchdog設定の共通化 - 4.1以前の設定

自ノードと他ノードを区別するために、ノードごとに異なる設定を行う



ノード間Watchdog設定の共通化 – 4.2以降の設定

- ノード間Watchdogの設定は共通化された
- 自ノードと他ノードの設定を区別するために、**pgpool_node_id**ファイルを用いる



動作モード設定の改善 – Pgpool-IIの動作モード

動作モード	説明
ストリーミングレプリケーションモード	ストリーミングレプリケーションを構成するPostgreSQLクラスタを管理する。 PostgreSQLがデータベースクラスタの同期を行う。
ロジカルレプリケーションモード	ロジカルレプリケーションを構成するPostgreSQLクラスタを管理する。 PostgreSQLがテーブルの同期を行う。
Slonyモード	Slony-Iを使用するPostgreSQLクラスタを管理する。 Slony-Iがテーブルの同期を行う。
ネイティブレプリケーションモード	Pgpool-IIがデータベースクラスタの同期を行う。
Rawモード	Pgpool-IIを経由してPostgreSQLに接続するだけのモード。 Pgpool-IIはデータベースの同期には関与しない。負荷分散はできない。
スナップショットアイソレーションモード (Pgpool-II 4.2以降、実験的な実装)	ネイティブレプリケーションモードの拡張。 ノードをまたがる読み取りの一貫性を保証する。

動作モード設定の改善 – 動作モード設定パラメータの統合

動作モード	Pgpool-II 4.1以前			Pgpool-II 4.2以降
	master_slave_mode	**master_slave_sub_mode**	**replication_mode**	**backend_clustering_mode**
ストリーミング レプリケーションモード	on	stream	off	streaming_replication
ロジカル レプリケーションモード	on	logical	off	logical_replication
Slonyモード	on	slony	off	slony
ネイティブ レプリケーションモード	off	-	on	native_replication
Rawモード	off	-	off	raw
スナップショット アイソレーションモード (Pgpool-II 4.2以降)	-	-	-	snapshot_isolation

ログ収集プロセス

- 標準エラーに出力されたログを収集し、ログファイルに保存する
- Pgpool-IIのみでログの収集やローテーションを行えるようになった
- パラメータはPostgreSQLと同じ

```
log_destination = 'stderr'
logging_collector = on
log_directory = '/tmp/pgpool_log'
log_filename = 'pgpool-%Y-%m-%d_%H%M%S.log'
log_file_mode = 0600
log_truncate_on_rotation = off
log_rotation_age = 1d
log_rotation_size = 10MB
```

ユーザパスワードの一括登録

- Pgpool-IIがクライアント認証を行うために、パスワードファイルを保持する
- ユーザが**pg_md5/pg_enc**を利用し、暗号化したパスワードをパスワードファイルに登録する
- Pgpool-II 4.2では、user:password形式で記述したファイルを指定するオプション**-i/--input-file**が追加された

```
$ cat users.txt
user1:password1
user2:password2

$ pg_enc -m -f /etc/pgpool-II/pgpool.conf -i users.txt

$ cat /etc/pgpool-II/pool_passwd
user1:AESX9MviGxOufr/+FLHT7OH3g==
user2:AESVPeWFZsrIq5YwCX1eXvn9g==
```

読み取り/書き込み関数の自動設定

- 読み取りクエリを負荷分散する
- SQLは解析できるが、関数が書き込みを行うかは解析できない

Pgpool-II 4.1以前

- read_only_function_list、write_function_listのいずれかを設定し、関数を含むクエリの負荷分散を制御
  - read_only_function_listに指定した読み取り関数は負荷分散を行い、それ以外は行わない
  - write_function_listに指定した書き込み関数は負荷分散を行わず、それ以外は行う

Pgpool-II 4.2以降

- write_function_listおよびread_only_function_listが空文字の場合、システムカタログの情報を参照し、関数の変動性をチェックする
- VOLATILE関数であれば、書き込みを行う関数と見なされ、負荷分散されない
- write_function_listまたはread_only_function_listのいずれかが空文字以外の場合、Pgpool-II 4.1以前と同じ動作

各種統計情報の取得

SQL コマンドの統計情報

実行されたSQLコマンドの数と、バックエンドからエラーが返された回数を表示

```
test=# SHOW pool_backend_stats;
 node_id | hostname | port  | status |  role   | select_cnt | insert_cnt | update_cnt | delete_cnt | ddl_cnt | other_cnt | panic_cnt | fatal_cnt | error_cnt
---------+----------+-------+--------+---------+------------+------------+------------+------------+---------+-----------+-----------+-----------+----------
 0       | /tmp     | 11002 | up     | primary | 12         | 10         | 30         | 0          | 2       | 30        | 0         | 0         | 1
 1       | /tmp     | 11003 | up     | standby | 12         | 0          | 0          | 0          | 0       | 23        | 0         | 0         | 1
(2 rows)
```

ヘルスチェックの統計情報

ヘルスチェックの統計情報を表示

```
test=# SHOW pool_health_check_stats;
 node_id | hostname | port  | status |  role   | last_status_change  | total_count | success_count | fail_count | skip_count | retry_count |
---------+----------+-------+--------+---------+---------------------+-------------+---------------+------------+------------+-------------+ ...
 0       | /tmp     | 11002 | up     | primary | 2020-11-09 01:53:19 | 1187        | 1187          | 0          | 0          | 0           |
 1       | /tmp     | 11003 | up     | standby | 2020-11-09 01:53:19 | 1187        | 1187          | 0          | 0          | 0           |
(2 rows)
```

設定ファイルの再読み込みを行うPCPコマンドの追加

- PCPコマンドはネットワークを経由で複数のPgpool-IIを制御するコマンド
- Pgpool-II 4.1以前では、**pgpool reload**で設定ファイルの再読み込みを行う
- Pgpool-II 4.2では、設定ファイルの再読み込みを行う**pcp_reload_config**コマンドが追加された
  - リモートからPgpool-IIの設定ファイルの再読み込みを行える
  - すべてのPgpool-IIをまとめて設定ファイルの再読み込みを行える

```
# 特定のPgpool-IIに対して実行
$ pcp_reload_config -w –h localhost –U postgres –p 11001
pcp_reload_config -- Command Successful

# すべてのPgpool-IIに対して実行
$ pcp_reload_config -w –h localhost –U postgres –p 11001 --scope=cluster
pcp_reload_config -- Command Successful
```

スナップショットアイソレーションモードとは

- ネイティブレプリケーションモードの拡張
- スナップショットの管理機能を追加して、ノードをまたがる読み取りの一貫性を保証する

Pgpool-II の動作モード

- ストリーミングレプリケーションモード
  - PostgreSQLのストリーミングレプリケーションでレプリケーションを行うモード
- ネイティブレプリケーションモード
  - Pgpool-IIが各ノードで同じSQLを実行することで、レプリケーションを行うモード
  - 利点：データ同期の遅延がない
  - 欠点：ノード間での読み取りの一貫性が保証されない

ネイティブレプリケーションモードにおけるDBノード間でのデータ不整合問題

- タイミングによって、あるバックエンドではコミット済みのデータが読めるが、他のバックエンドではコミット前のデータが読み出されることがある

初期値：

```
SELECT * FROM t1;
 i
---
 0
```

スナップショットアイソレーションモードの仕組み

- 本来のネイティブレプリケーションモードに**スナップショットの管理機能**を追加
- トランザクション内の最初のSQLコマンドでスナップショットを取得するときにCOMMITと**排他制御**をすることで、ノード間での読み取り一貫性を保証
- トランザクション分離レベルは**Repeatable readのみをサポート**
- 研究論文に基づいて実装されている
  - Pangea: An Eager Database Replication Middleware guaranteeing Snapshot Isolation without modification of Database Servers

スナップショットアイソレーションモードの可能性

- PostgreSQLに一切手を入れることなくGlobal Snapshot Isolationが保証できる
  - 発表されているGlobal Snapshot Isolationを保証する方法は、すべてPostgreSQLの大幅な改造が必要
- サーバのPostgreSQLのメジャーバージョンが異なっていても構わない
- サーバはPostgreSQL互換であれば利用可能
  - PowerGres Plus、Amazon RDS、EDB Postgresなど

用語の変更

- 用語の変更
  - master -> main, leader, primary
  - slave -> replica
  - white -> read, cache safe
  - black -> write, primary routing, cache unsafe
- パラメータ名の変更
  - black_function_list   -> write_function_list
  - white_function_list   -> read_only_function_list
  - black_query_pattern   -> primary_routing_query_pattern
  - black_memcache_table_list -> cache_unsafe_table_list
  - white_memcache_table_list -> cache_safe_table_list
  - follow_master_command   -> follow_primary_command
  - backend_flagのALWAYS_MASTER   -> ALWAYS_PRIMARY
  - relcache_query_targetのmaster -> primary

PostgreSQL 13のSQLパーサの移植

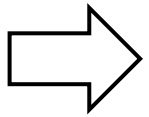
- Pgpool-IIはSQLパーサを持っている
  - 複雑なSQLを正確に解析するため
  - クエリの書き換えを行うため
- メジャーリリースを行うたびに、PostgreSQLの最新パーサを移植
- Pgpool-II 4.2はPostgreSQL 13のパーサを移植

```
ALTER TRIGGER ... NO DEPENDS ON EXTENSION ...
ALTER FUNCTION ... NO DEPENDS ON EXTENSION ...
FETCH FIRST ... WITH TIES
ALTER VIEW ... RENAME COLUMN ... TO ...
ALTER TABLE ... DROP EXPRESSION
ALTER STATISTICS ... SET STATISTICS
DROP DATABASE ... WITH (force)
CREATE DATABASE ... LOCALE
VACUUM (PARALLEL 1) ...
```

その他の変更点 (1) Pgpool-IIの内部プロセス名の出力

- Pgpool-IIの内部プロセス名をアプリケーション名としてログに出力
  - Pgpool-IIメインプロセス -> main
  - Pgpool-II子プロセス -> child
  - ヘルスチェックプロセス -> health_check%d
  - レプリケーションチェックプロセス -> sr_check_worker
  - PCPメインプロセス -> pcp_main
  - PCP子プロセス -> pcp_child

```
log_line_prefix = '%t %a [%p] '
```



```
2021-01-15 19:55:53 main [10249] LOG:
2021-01-15 19:55:53 main [10249] DEBUG:
2021-01-15 19:55:53 child [10280] DEBUG:
2021-01-15 19:55:53 sr_check_worker [10287] DEBUG:
2021-01-15 19:55:53 health_check0 [10288] DEBUG:
2021-01-15 19:55:53 health_check1 [10289] DEBUG:
2021-01-15 19:55:53 pcp_main [10286] DEBUG:
```

その他の変更点 (2) アプリケーション名出力の改良

- log_line_prefixのアプリケーション名設定パラメータ「%a」が対応する設定方法
  - スタートアップメッセージ
  - 接続パラメータ (Pgpool-II 4.2以降)
  - SETコマンド (Pgpool-II 4.2以降)

```
log_line_prefix = '%t %a [%p] '
```

```
$ PGAPPNAME=myapp1 psql -h /tmp -p 11000
postgres=# select 1;

postgres=# SET application_name TO 'myapp2';

postgres=# select 2;
```

```
main [11423] LOG:  pgpool-II successfully started. version 4.2.1
main [11423] LOG:  node status[0]: 1
main [11423] LOG:  node status[1]: 2
myapp1 [11445] LOG:  DB node id: 0 backend pid: 11469 statement: SELECT
myapp1 [11445] LOG:  pool_reuse_block: blockid: 0
myapp1 [11445] CONTEXT:  while searching system catalog, When relcache is
myapp1 [11445] LOG:  DB node id: 1 backend pid: 11470 statement: select 1;
myapp1 [11445] LOG:  DB node id: 0 backend pid: 11469 statement:  DISCARD
myapp1 [11445] LOG:  DB node id: 1 backend pid: 11470 statement:  DISCARD
myapp1 [11445] LOG:  DB node id: 0 backend pid: 11625 statement:  DISCARD
myapp1 [11445] LOG:  DB node id: 1 backend pid: 11651 statement: select 1;
myapp1 [11445] LOG:  DB node id: 0 backend pid: 11650 statement: SET
myapp1 [11445] LOG:  DB node id: 1 backend pid: 11651 statement: SET
myapp2 [11445] LOG:  DB node id: 1 backend pid: 11651 statement: select 2;
```

その他の変更点 (3) 接続終了時のログの出力

- log_disconnections (Pgpool-II 4.2以降)
- log_disconnectionsを設定することで、接続終了をログへ出力

```
log_connections = on
log_disconnections = on
```

```
2021-01-15 20:37:39 child [12330] LOG:  new connection received
2021-01-15 20:37:39 child [12330] DETAIL:  connecting host=[local]
2021-01-15 20:37:43 psql [12330] LOG:  DB node id: 0 backend pid: 12359 statement:
2021-01-15 20:37:43 psql [12330] LOG:  pool_reuse_block: blockid: 0
2021-01-15 20:37:43 psql [12330] CONTEXT:  while searching system catalog, When
relcache is missed
2021-01-15 20:37:43 psql [12330] LOG:  DB node id: 0 backend pid: 12359 statement:
2021-01-15 20:37:43 psql [12330] LOG:  DB node id: 1 backend pid: 12360 statement:
2021-01-15 20:37:43 psql [12330] LOG:  frontend disconnection: session time:
0:00:03.555 user=pengbo database=postgres host=[local]
```

まとめ

- Pgpool-II 4.2
  - 設定と管理が容易に
  - より多くのクラスタの統計情報を取得可能
  - スナップショットアイソレーションモードの追加
  - Pgpool-II 4.2にアップグレードする場合、設定パラメータや用語の変更に注意が必要
- Pgpool-II 4.2リリースノート
  - https://www.pgpool.net/docs/latest/ja/html/release-4-2-0.html

ご清聴ありがとうございました。