Pgpool-II 4.5新機能紹介

2024/02/27

SRA OSS LLC
彭 博 (ペン ボ)

Pgpool-II 4.5の新機能ハイライト

長年の問題が解決

- マルチステートメント内のトランザクション検出
- フェイルオーバ時のセッション切断問題の軽減

設定・管理がより容易に

- プリペアドステートメントの負荷分散
- ユーザ名に基づく負荷分散
- pcp_socket_dirパラメータの複数ディレクトリ指定
- delay_threshold_by_timeのミリ秒単位指定
- など

マルチステートメント内のトランザクション検出①

- Pgpool-IIでは、マルチステートメントはプライマリのみに送られる
- 4.4以前、トランザクションの開始・終了を検出できなかった
  - トランザクション内でしか実行できないクエリが、トランザクション外で実行されてしまう可能性があった

例：ノード0がプライマリ、ノード1がロードバランスノードの場合、

SAVEPOINTがトランザクションが未開始のスタンバイ（ノード1）で実行されてしまう

4.4

```
test=# BEGIN¥;SELECT 1;
BEGIN
 ?column?
----------
        1
(1 row)
test=*# SAVEPOINT sp;
WARNING:  packet kind of backend 1 ['E'] does not match with main/majority
nodes packet kind ['C']
FATAL:  failed to read kind from backend
DETAIL:  kind mismatch among backends. Possible last query was: "SAVEPOINT
sp;" kind details are: 0[C]1[E: SAVEPOINT can only be used in
transaction blocks]
HINT:  check data consistency among db nodes
server closed the connection unexpectedly...
```

← マルチステートメントなので、プライマリに送られる

← SAVEPOINT関連のクエリはプライマリとロードバランスノードの両方に送られる

マルチステートメント内のトランザクション検出②

- 4.5から、マルチステートメント内のトランザクション開始・終了の検出が可能に
  - PostgreSQLのpsqlscanモジュールの取り込みによって、トランザクションの開始・終了を検出
  - マルチステートメントによって開始された明示的なトランザクションを検出し、後続のクエリがトランザクション内でしか実行できない場合（PREPARE、EXECUTE、DEALLOCATE、SAVEPOINT、COMMIT/ROLLBACKなど）、プライマリのみに送信
  - バグ修正の意味合いもあるので、4.1～4.4系列にも適用している

4.5

```
LOG:  DB node id: 0 backend pid: 32013 statement: BEGIN;SELECT
1;
LOG:  DB node id: 0 backend pid: 32013 statement: SAVEPOINT sp;
```

プライマリのみに送信

プリペアドステートメントの負荷分散

- 4.4以前はプリペアドステートメントはプライマリのみに送信
  - プリペアド文の内容までは解析しない
  - プリペアド文の内容が参照のみの場合でもプライマリに送信

4.4

```
test=# PREPARE pst(int) AS SELECT c2 FROM t1 WHERE c1 = $1;
test=# EXECUTE pst(1);
test=# ¥c
test=# PREPARE pst(int) AS SELECT c2 FROM t1 WHERE c1
test=# EXECUTE pst(1);
test=# ¥c
test=# PREPARE pst(int) AS SELECT c2 FROM t1 WHERE c1
test=# EXECUTE pst(1);
```

解析しない。
プリペイド文であれば、プライマリに

```
LOG:  DB node id: 0 backend pid: 7599 statement: PREPARE pst(int)
AS SELECT c2 FROM t1 WHERE c1 = $1    ;
LOG:  DB node id: 0 backend pid: 7599 statement: EXECUTE pst(1);
LOG:  DB node id: 0 backend pid: 7597 statement: DISCARD ALL
LOG:  DB node id: 2 backend pid: 7599 statement: DISCARD ALL
LOG:  DB node id: 0 backend pid: 7743 statement: PREPARE pst(int)
AS SELECT c2 FROM t1 WHERE c1 = $1    ;
LOG:  DB node id: 0 backend pid: 7743 statement: EXECUTE pst(1);
LOG:  DB node id: 0 backend pid: 7742 statement: DISCARD ALL
LOG:  DB node id: 1 backend pid: 7743 statement: DISCARD ALL
LOG:  DB node id: 0 backend pid: 7599 statement: PREPARE pst(int)
AS SELECT c2 FROM t1 WHERE c1 = $1    ;
LOG:  DB node id: 0 backend pid: 7599 statement: EXECUTE pst(1);
```

プリペアドステートメントの負荷分散

- 4.5からはプリペアドステートメントの負荷分散が可能に
  - プリペアド文の内容を解析
  - プリペアド文の内容が参照クエリの場合、いずれかのノードに送信

4.5

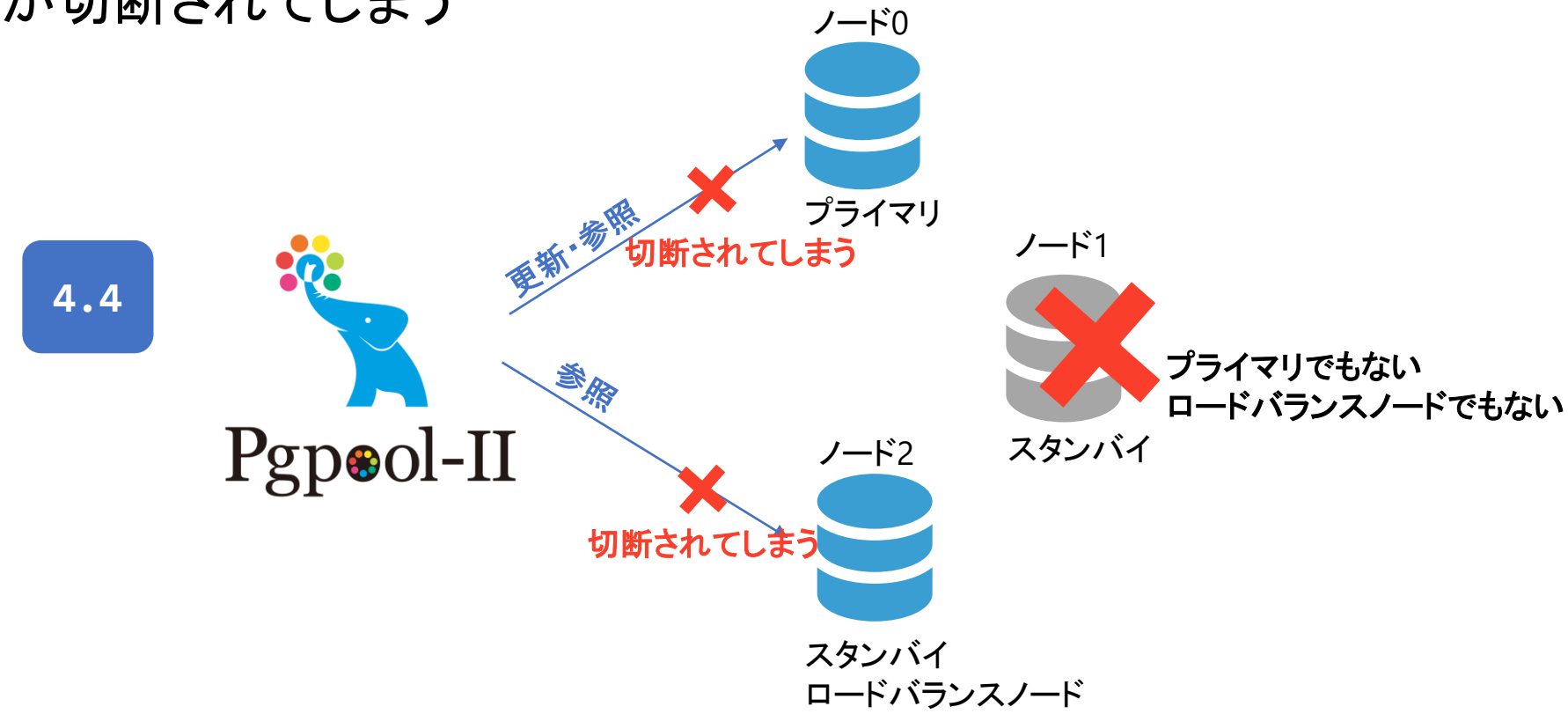
```
test=# PREPARE pst(int) AS SELECT c2 FROM t1 WHERE c1 = $1;
test=# EXECUTE pst(1);
test=# ¥c
test=# PREPARE pst(int) AS SELECT c2 FROM t1 WHERE
test=# EXECUTE pst(1);
test=# ¥c
test=# PREPARE pst(int) AS SELECT c2 FROM t1 WHERE
test=# EXECUTE pst(1);
```

負荷分散される

```
LOG:  DB node id: 2 backend pid: 7599 statement: PREPARE pst(int)
AS SELECT c2 FROM t1 WHERE c1 = $1    ;
LOG:  DB node id: 2 backend pid: 7599 statement: EXECUTE pst(1);
LOG:  DB node id: 0 backend pid: 7597 statement: DISCARD ALL
LOG:  DB node id: 2 backend pid: 7599 statement: DISCARD ALL
LOG:  DB node id: 1 backend pid: 7743 statement: PREPARE pst(int)
AS SELECT c2 FROM t1 WHERE c1 = $1    ;
LOG:  DB node id: 1 backend pid: 7743 statement: EXECUTE pst(1);
LOG:  DB node id: 0 backend pid: 7742 statement: DISCARD ALL
LOG:  DB node id: 1 backend pid: 7743 statement: DISCARD ALL
LOG:  DB node id: 2 backend pid: 7599 statement: PREPARE pst(int)
AS SELECT c2 FROM t1 WHERE c1 = $1    ;
LOG:  DB node id: 2 backend pid: 7599 statement: EXECUTE pst(1);
```

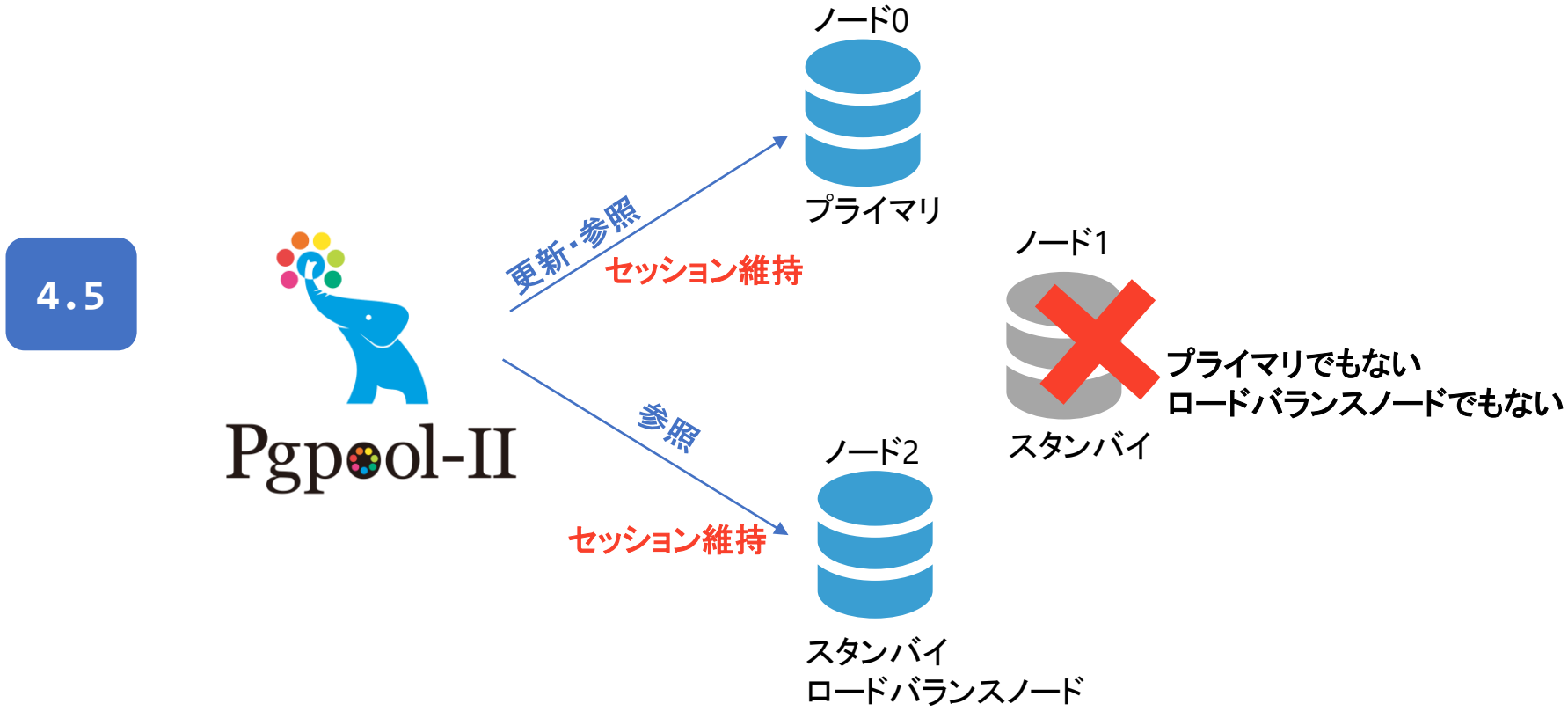
フェイルオーバ時のセッション切断問題の軽減

- Pgpool-IIでは、内部処理の至る所でバックエンドノードの状態をチェックしている
- それがフェイルオーバー中にされると、そのノードが使われていなくても（ダウンしたノードがプライマリまたはメインノードではなく、ロードバランスノードでもない）、セッションが切断されてしまう



フェイルオーバ時のセッション切断問題の軽減

- 4.5からは、バックエンドノードの状態をチェックするときに、フェイルオーバー中の場合、フェイルオーバーが完了するまで待つように
- この修正によって、フェイルオーバー時のセッション切断問題が軽減された



セッションロードバランスノードの確認

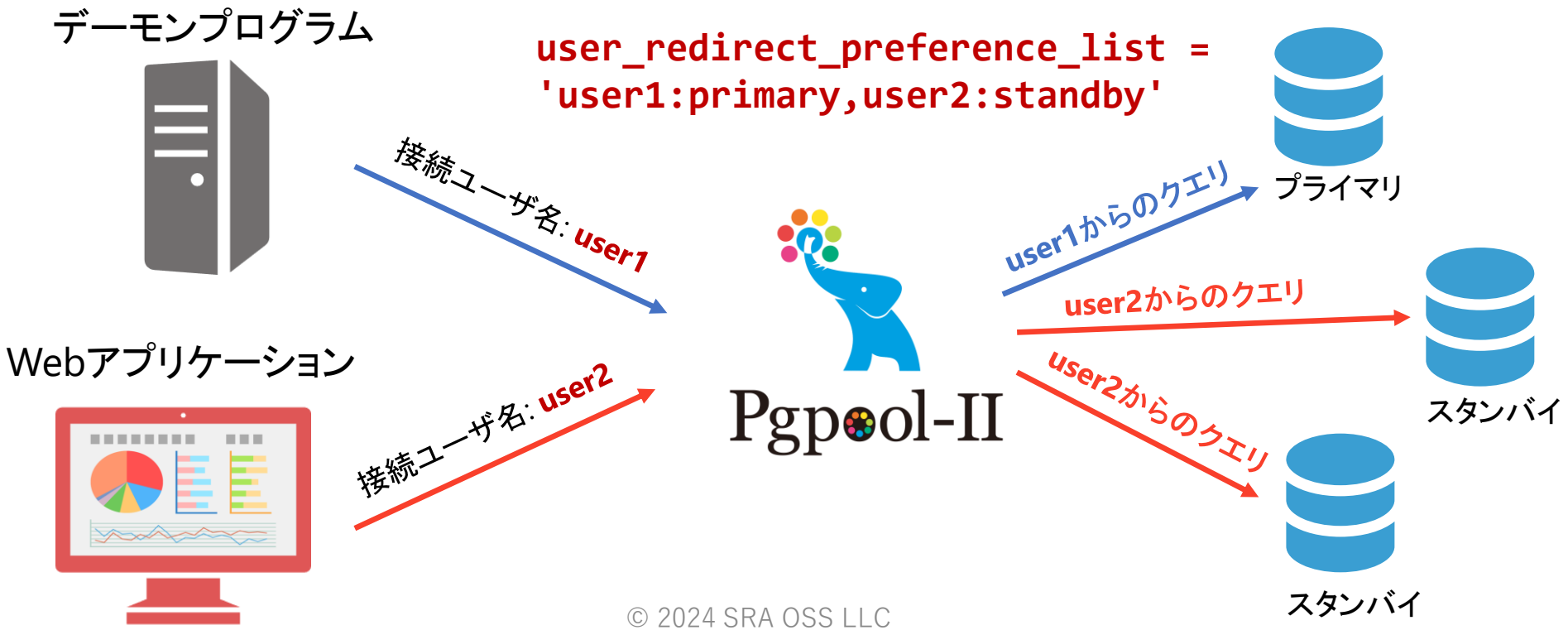
- 新しいフィールド**load_balance_node**を**SHOW POOL_POOLS**と**pcp_proc_info**に追加
  - フロントエンドが接続中でバックエンドが負荷分散ノードならば1, そうでなければ0
  - 他のセッションに影響を与えずに、バックエンドノードを安全に停止できるかどうかを確認できるように
  - 0（ロードバランスノードではない）であれば、安全に停止できる

```
postgres=# SHOW POOL_POOLS;
 pool_pid | backend_id      | status              | load_balance_node
----------+-----------------+---------------------+-------------------
 2370     | 0               | Wait for connection | 0
 2370     | 1               | Wait for connection | 0
 2370     | 2               | Wait for connection | 0
 2371     | 0               | Execute command     | 0
 2371     | 1               | Execute command     | 0
 2371     | 2               | Execute command     | 1
 2372     | 0               | Wait for connection | 0
 2372     | 1               | Wait for connection | 0
 2372     | 2               | Wait for connection | 0
 2373     | 0               | Idle                | 0 (ノード0: プライマリ)
 2373     | 1               | Idle                | 0
 2373     | 2               | Idle                | 1 (ノード2:ロードバランスノード)
```

ノード1を停止しても、クライアント
セッションは切断されない

ユーザ名に基づく負荷分散

- 4.4まで、アプリケーション名とデータベース名に基づく負荷分散が可能
- 4.5から、ユーザ名に基づく負荷分散も可能に
  - user_redirect_preference_list = '**ユーザ名:ノードID(比率),ユーザ名:ノードID(比率)**, ...'
  - アプリケーション名が変更できないようなアプリケーションの場合に便利



delay_threshold_by_timeのミリ秒単位指定

- 4.4以前、s（秒）、min（分）、h （時間）、d（日）単位で指定可能
- 4.5から、ms（ミリ秒）単位で設定できるように

```
delay_threshold_by_time = 1ms
                # Threshold before not dispatching query to standby node
                # The default unit is in millisecond(s)
                # Disabled (0) by default
```

単位を指定しなかった場合、4.4以前は秒に対し、4.5ではミリ秒。
単位が異なるため、バージョンアップの際には要注意!

pcp_socket_dirの複数ディレクトリ指定

- 4.4まで、ディレクトリを1つしか指定できない
- 4.5から、カンマ区切りで複数のディレクトリを指定できるように

```
pcp_socket_dir = '/var/run/postgresql,/tmp'
                                # Unix domain socket path(s) for pcp
                                # The Debian package defaults to
                                # /var/run/postgresql
                                # (change requires restart)
```

```
$ pcp_node_info -p 11001 -h /var/run/postgresql
$ pcp_node_info -p 11001 -h /tmp
```

PostgreSQL 16パーサの移植

- Pgpool-IIは読み取りクエリを振り分けするために、PostgreSQLのSQLパーサを移植している
- Pgpool-II 4.5ではPostgreSQL 16のパーサを取り込んでいる

PostgreSQL 16パーサのおもな変更点

- COPY FROMの新しいオプションDEFAULTの追加
- CREATE TABLEでSTORAGEタイプの指定が可能に
- VACUUMオプションの追加
- など

4.5へバージョンアップの注意点①

- 4.4以前、パスワードファイル（~/.pcppassまたはPCPPASSFILEによって定義されたファイル）に格納されたパスワードを使用するために、**-w/--no-passwordオプションの指定が必要**
- 4.5から、**-w/--no-passwordオプションを指定しなくても**、パスワードファイルから読み取る

4.4

```
$ pcp_node_info -p 11001 -w
```

4.5

```
$ pcp_node_info -p 11001
```

パスワード入力プロンプトを表示するためには、-Wオプションの指定が必要

4.5へバージョンアップの注意点②

通常のERRORメッセージをDEBUGメッセージにダウングレード

```
ERROR:  unable to flush data to frontend

ERROR:  unable to read data from frontend
DETAIL: EOF encountered with frontend

ERROR:  unable to read data
DETAIL: child connection forced to terminate due to
client_idle_limit:30 is reached
```

4.5へバージョンアップの注意点③

delay_threshold_by_timeパラメータ

単位を指定しなかった場合、4.4以前は秒に対し、4.5ではミリ秒。
単位が異なるため、バージョンアップの際には要注意!

参考情報

- Pgpool-II Wiki
  - https://pgpool.net/
- 日本語版ドキュメント
  - https://www.pgpool.net/docs/latest/ja/html/
- Pgpool-II 4.5 リリースノート
  - https://www.pgpool.net/docs/latest/ja/html/release-4-5-0.html

ご清聴ありがとうございました。