PostgreSQLのしくみ勉強会 (2009年6月20日)

PL/Proxy と pgbouncer

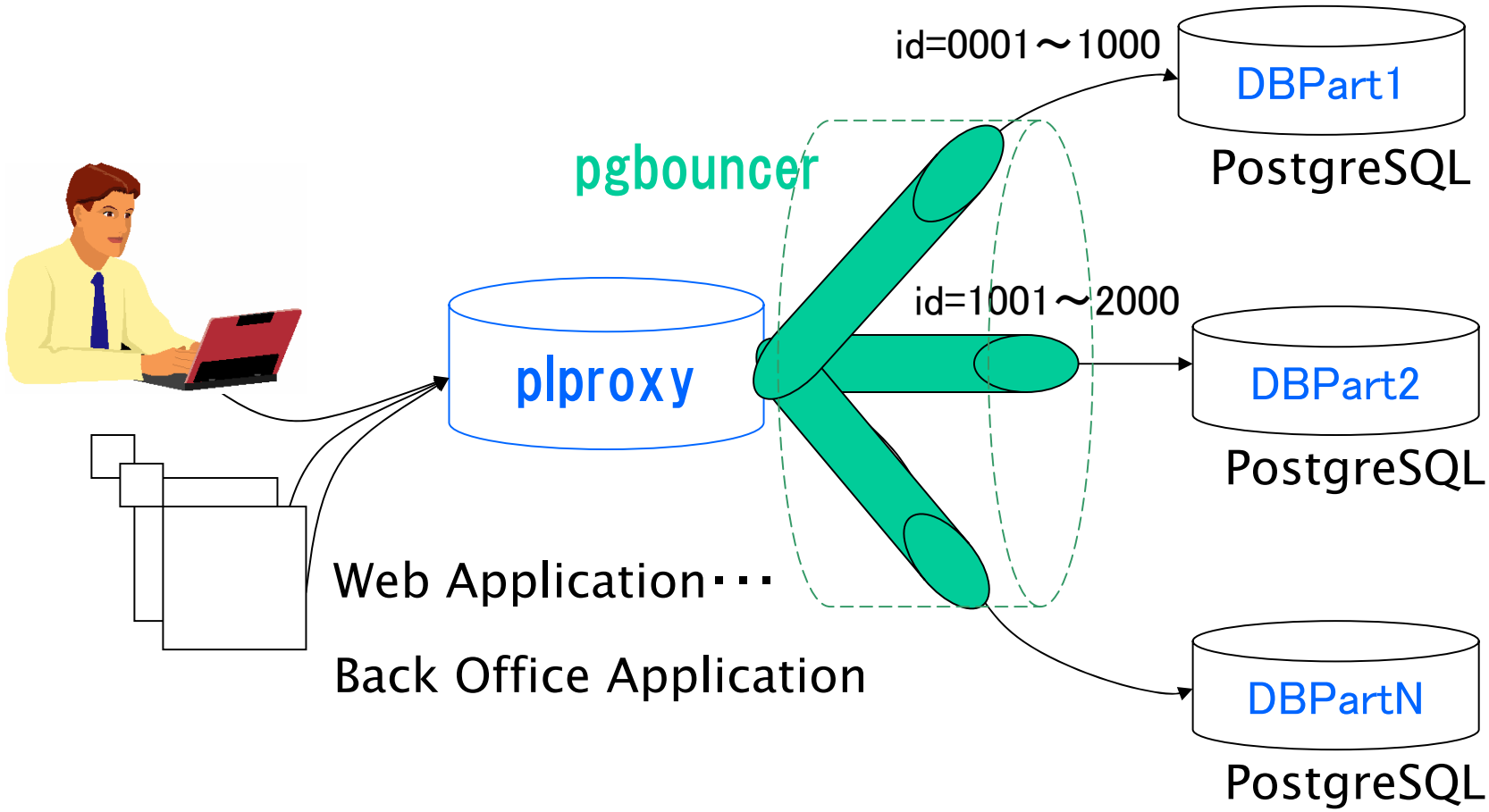
講師:桑村 潤

、補助:さいとう ひろし

PL/Proxy, pgbouncer とは

- PL/Proxy, pgbouncerは、Skype™社によって培われたテクノロジで、 軽量かつ便利にPostgreSQLを活用できるものとして、公開された機能
  - pgbouncerはコネクションプーラー
  - PL/Proxyはリモート DB呼び出し用プロキシー言語 で、レプリケーションやパーティショニングの記述が可能
- 2008年のPGConで改良安定版が紹介され、現在は、同社の Marko Kreen氏が中心になりメンテナンス。
- PostgreSQLと同様に、BSDライセンス。
- トラフィックが急激に増大するシステムを安定運用するのが目的のひとつ

概念図



pgbouncer の特長

- 軽量かつ強固なconnection pooler、必要なメモリ量は、概ねconnection当たり2kb
- 異なった複数のPostgreSQLサーバを対象
- 全て、もしくは、特定のPostgreSQLサーバに対してconnection を中断可能
- 殆どの環境設定項目がオンライン中に変更することが可能
- client connectionsは切断せずにオンラインのリスタート／アップグレードが可能
- 処理中SQLを解析しないため、CPUへの負荷は小さい

- https://developer.skype.com/SkypeGarage/DbProjects/PgBouncer

pgbouncer ：インストール

- 提供サイトの http://pgfoundry.org/projects/pgbouncer から安定版をダウンロード(pgbouncer-1.3　2009-02-18)
- ビルドは、所定の作業ディレクトリにpgbouncerを展開し, configure –with-libevent=/prefix; make; make install;を実行 (libevent-1.3b以降が必要

  http://www.monkey.org/~provos/libevent からダウンロードしてインストール　libevent-1.4.11-stable 2009-5-14)
- pgbouncer 構成ファイルを編集する

pgbouncer：プーリング方式

Session Pooling

最も堅実な方法。クライアントが接続すると、1つのサーバ接続が、接続の間、持続して割り当てられる。クライアントの接続が切れたとき、サーバ接続はプールに戻される。レガシーアプリではこの方式が有用。

Transaction pooling

サーバ接続が、1つのトランザクションの間だけ割り当てられる。pgbouncer がそのトランザクションが終了したことを認識した時点で、サーバ接続はプールに戻される。バックエンド接続したアプリケーションの例外で壊す恐れがあるので要注意。アプリケーションと協調をとって使い方を注意し、クライアントを壊す恐れのない機能のみを使うべき。

Statement pooling

最も活性的なプール方法。マルチステートメントのトランザクションには使えない。すなわち、“autocommit”モードをクライアントに強制し、PL/Proxyとの連携が主な目的。

pgbouncer 構成例

```
;; database name = connect string
[databases]
proxy = host=127.0.0.1 port=54320 dbname=jpug user=jpug password=pass
part1 = host=127.0.0.1 port=54321 dbname=jpug user=jpug password=pass
part2 = host=127.0.0.1 port=54322 dbname=jpug user=jpug password=pass

;; Configuation section
[pgbouncer]
admin_users = admin
stats_users = stat_collector

logfile = /home/jpug/pgsql/log/pgbouncer_test.log
pidfile = /home/jpug/pgsql/log/pgbouncer_test.pid

listen_addr = 127.0.0.1
listen_port = 6666

auth_type = trust                              ; any, trust, plain, crypt, md5
auth_file = /home/jpug/pgsql/pgbouncer_test_u.txt

pool_mode = session
;   session     - after client disconnects
;   transaction  - after transaction finishes
;   statement    - after statement finishes
...
```

PL/Proxyの特長

- PL/Proxyはリモートデータベースのプロシージャを呼び出すためのプロキシー言語
- 例えば、フィールド値のハッシュをもとにデータベース間にデータを分割（構成関数で定義）
- PostgreSQLのストアドプロシージャ言語で拡張モジュール
- リモート関数と同じ名前のプロキシー関数を作成し、行き先の情報はプロキシー関数の中で指定

- https://developer.skype.com/SkypeGarage/DbProjects/PlProxy

PL/Proxy：インストール

- 提供サイトの http://pgfoundry.org/projects/plproxy から
- 安定版をダウンロード(plproxy-2.0.8　2009-01-16)
- ビルドは、所定の作業ディレクトリにplproxyを展開して、

  make;　make install; を実行(問題が起きる場合は、pg_config等PostgreSQL の実行バイナリディレクトリにpathが通っているか確認のこと)
- ターゲットのデータベースへplproxy機能をインストールするために、plproxy.sqlファイルをロードする

```
# psql -f $PGSHARE/contrib/plproxy.sql mydb
```

- plproxyの動作確認のために、テスト機能を作成

PL/Proxy : Language

- 定義言語は、PL/PgSQLと類似
  ストリングの引用、コメント、行末のセミコロン
  ステートメントは4つだけ
  CONNECT, CLUSTER, RUN, SELECT
- 各々の機能は、どのデータベースでSQLを実行させるべきかを決定するために、CONNECTか、CLUSTERステートメントと RUNステートメントのペア構成が必要
- CONNECT 'libpq connstr'; クエリを接続し実行するために、場所を指定。複数機能が同じconnstrを持てば、同じ接続を使用
- CLUSTER 'cluster_name'; 実行するcluster nameを指定 cluster nameは、plproxy.get_cluster_*機能でpassされたもの
- CLUSTER cluster_func(..); proxy機能引数と同時にクラスタ名 をダイナミックに決定。cluster_funcはクラスタ名文字列を返す

PL/Proxy : Configuration

スキーマ：以下の3つの構成機能は、plproxyのために必須

plproxy.get_cluster_partitions(cluster_name text)

リモートのデータベースへ接続する際、指定されているplproxyの接続文字列で初期化。

plproxy.get_cluster_version(cluster_name text)

plproxyの構成が変更された場合、再読込みが必要。全ての機能がplproxyを経由するため、できる限り早く呼ばれるべき。

plproxy.get_cluster_config(
in cluster_name text, out key text,out val text)

接続が確立している間、plproxyパラメータを変更することができる。

PL/Proxy :Language RUN ON...

RUN ON ALL; クエリをcluster内の全パーティションで並列実行

RUN ON ANY; ランダムにいづれかのパーティションで実行

RUN ON <NR>; パーティション番号<NR>の上で実行

RUN ON partition_func(..);

１つ以上のハッシュ値(int4)を返すpartition_func()に指定した関数を実行。

RUN ON argument; 変数(例 $1)でパーティション番号を指定

※ クエリは、タグ付けされたパーティションで実行される。複数のパーティションにタグ付けされれば、クエリはそれらのパーティションで並列実行される

PL/Proxy: オンライン実習

- ユーザ名とメールアドレスからなるテーブルに、plproxyを使ってデータを格納してみる。わずかな設定変更で格納方法を変更できること、格納の仕方による振る舞いの違いを確認する。

RUN ON hashtext（名前）

- hashtext()関数の戻り値(int)によって、複数の格納先パーティションへ振り分ける

  ※ 次ページ以降のサンプル参照

RUN ON ALL

- 値セットを返す関数が必要となる(RETRUNS SET OF TEXT)
- すべてのパーティションに格納される

サンプル1：各バックエンドDBでの登録

クラスタ化するデータベースのそれぞれにテーブルの実体と関数を作成する。、関数名プロキシ上で登録するものと同じ名前、型、引数型にする

```
CREATE TABLE ユーザ (
  ユーザ名 text,
  メール text
);
-- 各リモート側データベースに定義する挿入関数
create language plpgsql;      --- 関数定義のため
CREATE OR REPLACE FUNCTION ユーザ挿入(ユーザIN text, メールIN text)
RETURNS integer AS $$                                  --- 型に注意
  INSERT INTO ユーザ (ユーザ名, メール) VALUES ($1,$2);
  SELECT 1;                                          --- 型に注意
$$ LANGUAGE SQL;
```

サンプル2：フロントエンドDBに登録するplproxy構成関数(1)
（plproxy.get_cluster_partitions）

クエリをリモートデータベースに送る必要があるとき、plproxyは plproxy.get_cluster_partitions(cluster)関数を呼び出し、各パーティションに送るための接続文字列を取得する（パーティションの数は2の階乗でなくてはならない）

```
CREATE LANGUAGE plpgsql;
CREATE SCHEMA plproxy;    --- plproxy関数用スキーマ
CREATE OR REPLACE FUNCTION
plproxy.get_cluster_partitions(cluster_name text)
RETURNS SETOF text AS $$
BEGIN
  IF cluster_name = 'クラスタ' THEN
    RETURN NEXT 'port=54321 host=127.0.0.1 dbname=jpug user=jpug';
    RETURN NEXT 'port=54322 host=127.0.0.1 dbname=jpug user=jpug';
    RETURN;
  END IF;
  RAISE EXCEPTION 'クラスタ名が見つかりません';
END;
$$ LANGUAGE plpgsql;
```

サンプル2：フロントエンドDBに登録するplproxy構成関数(2)
（plproxy.get_cluster_version）

plproxy.get_cluster_version(cluster_name) 関数は、リクエスト毎に呼び出され plproxy.get_cluster_partitions() の結果キャッシュを出力として再利用できるかどうか決定する

```
CREATE OR REPLACE FUNCTION
plproxy.get_cluster_version(cluster_name text)
RETURNS int4 AS $$
BEGIN
  IF cluster_name = 'クラスタ' THEN
    RETURN 1;
  END IF;
  RAISE EXCEPTION 'クラスタ名が見つかりません';
END;
$$ LANGUAGE plpgsql;
```

サンプル2：フロントエンドDBに登録するplproxy構成関数(3)
（ plproxy.get_cluster_config ）

plproxy.get_cluster_config() 関数はパラメータを調整する。ここでは接続の持続時間を設定している。
他のパラメータについては本体付属文書を参照のこと。

```
CREATE OR REPLACE FUNCTION plproxy.get_cluster_config(
  in cluster_name text,
  out key text,
  out val text)
RETURNS SETOF record AS $$
BEGIN
  -- lets use same config for all clusters
  key := 'connection_lifetime';
  val := 30*60; -- 30min.
  RETURN NEXT;
  RETURN;
END;
$$ LANGUAGE plpgsql;
```

サンプル3：フロントエンドDBに登録する（plproxy関数）

ここでは、ユーザ・テーブルは、ユーザ名のハッシュ値によっていくつかのデータベースに分散されていることを前提としている。パーティショニングされるデータベースへの接続文字列は get_cluster_partitions() 関数の中にある。次の関数は、プロキシーサーバで実行され、適切にパーティショニングされたリモートデータベースから、指定ユーザのメールアドレスを取得。

```
CREATE LANGUAGE plproxy;        --- plproxy言語登録
-- プロキシ側データベースに定義する挿入関数
CREATE OR REPLACE FUNCTION ユーザ挿入(ユーザIN text, メールIN text)
RETURNS integer AS $$           --- 型に注意
  CLUSTER 'クラスタ';
  RUN ON hashtext(ユーザIN);
$$ LANGUAGE plproxy;
```

謝辞

- 本セミナーの会場ならびに遠隔サーバ利用のための端末と回線をご提供くださった産業技術大学院大学(AIIT)様に感謝いたします。
- 本セミナーのためにOSSオープン・ラボをご提供くださった独立法人情報処理推進機構(IPA)様に感謝いたします。
- 本セミナー資料、及びPL/Proxy, pgbouncerの動作について、情報提供いただいたSkype™社のMarko Kreen氏に感謝いたします。

おしまいです

お疲れさまでした。

こんどのJPUGイベントはこれだ！　今のうちにスケジュールを！

PostgreSQL Conference 2009 Japan
JPUG 10th Anniversary Conference

日にち：　2009年11月20日（金）～21日（土）
場所：AP浜松町（東京都港区）
主催：　日本PostgreSQLユーザ会
運営：　PostgreSQL Conference 2009 Japan 実行委員会