新バージョン対応！
基礎から学ぶpgpool-IIの使い方

SRA OSS, Inc. 日本支社
pgpool-II開発者
北川 俊広

2011/10/5

SRA OSS,INC.

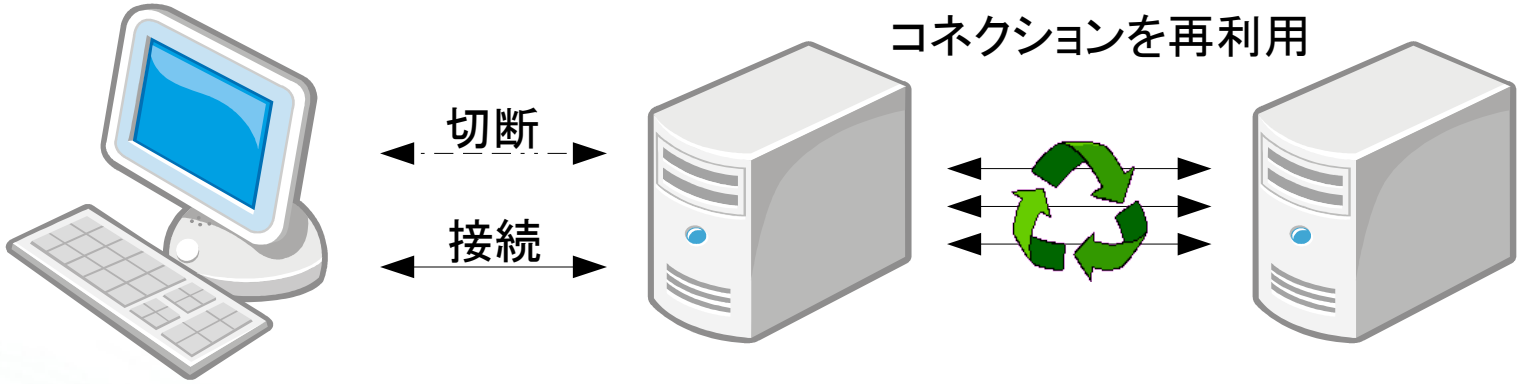
pgpool-IIとは

- アプリケーションとPostgreSQLの間に入って、便利な機能を提供するソフトウェア
- オープンソースソフトウェア（BSDライセンス）
  - pgpool Global Development Group にて開発
- 多彩な機能
  - 同期レプリケーション、ロードバランス、コネクションプーリング、自動フェイルオーバーなど
  - 他のレプリケーションソフトウェアとの連携
    - Streaming replication, Warm standby, Slony-I
- 設定が容易、GUI管理ツールも用意

Streaming Replication/Hot Standby とpgpool-IIを組み合わせる

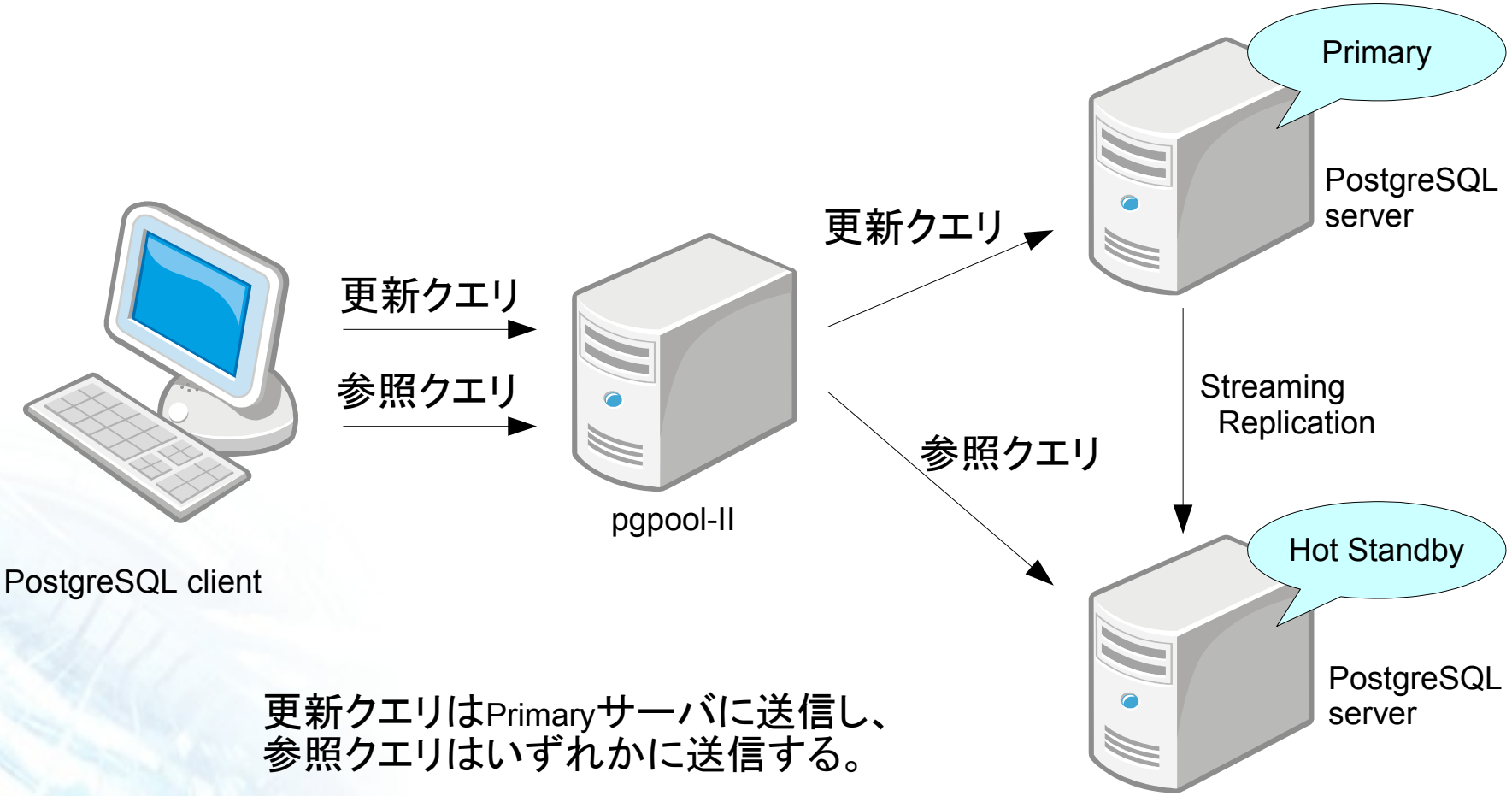
アプリケーション側を作りこまずに、
高可用性と性能向上を簡単に実現できる

コネクションプーリング



すでに確立しているコネクションを再利用することにより、PostgreSQLが接続時に行っている、認証、子プロセスの生成、データアクセスのための前処理などを省いて効率化できる。

クエリ振り分けの自動化、ロードバランス

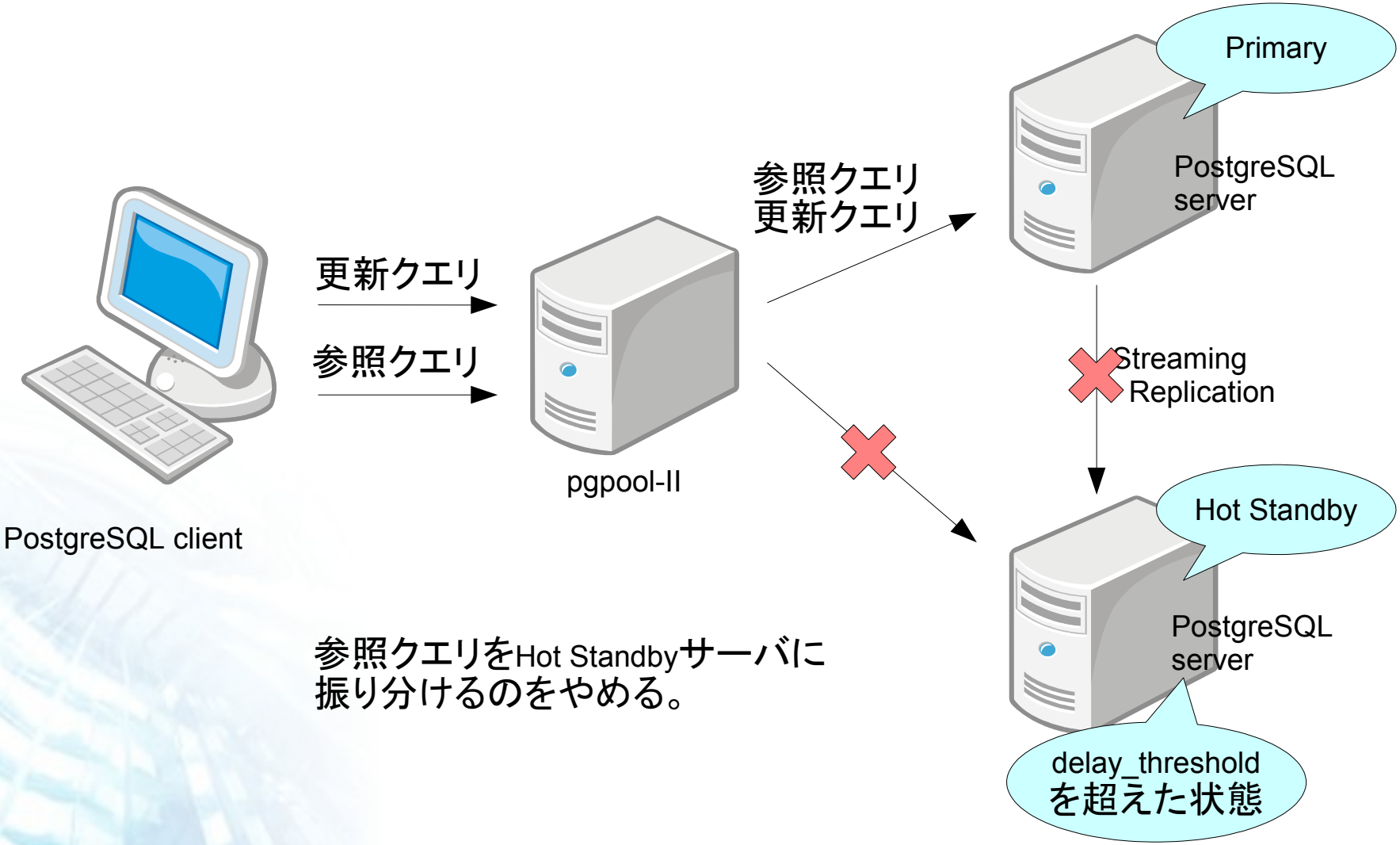


更新クエリはPrimaryサーバに送信し、
参照クエリはいずれかに送信する。

Hot Stanby側の制約を考慮したクエリ振り分け
～ Hot Standby側で実行できないクエリ ～

- Data Manipulation Language (DML) - INSERT, UPDATE, DELETE, COPY FROM, TRUNCATE. Note that there are no allowed actions that result in a trigger
- being executed during recovery.
- Data Definition Language (DDL) - CREATE, DROP, ALTER, COMMENT. This also applies to temporary tables also because currently their definition causes
- writes to catalog tables.
- SELECT ... FOR SHARE | UPDATE which cause row locks to be written
- Rules on SELECT statements that generate DML commands.
- LOCK that explicitly requests a mode higher than ROW EXCLUSIVE MODE.
- LOCK in short default form, since it requests ACCESS EXCLUSIVE MODE.
- Transaction management commands that explicitly set non-read-only state:
- BEGIN READ WRITE, START TRANSACTION READ WRITE
- SET TRANSACTION READ WRITE, SET SESSION CHARACTERISTICS AS TRANSACTION READ WRITE
- SET transaction_read_only = off
- Two-phase commit commands - PREPARE TRANSACTION, COMMIT PREPARED, ROLLBACK PREPARED because even read-only transactions need to write
- WAL in the prepare phase (the first phase of two phase commit).
- Sequence updates - nextval(), setval()
- LISTEN, UNLISTEN, NOTIFY

賢いロードバランス
(delay_threshold)

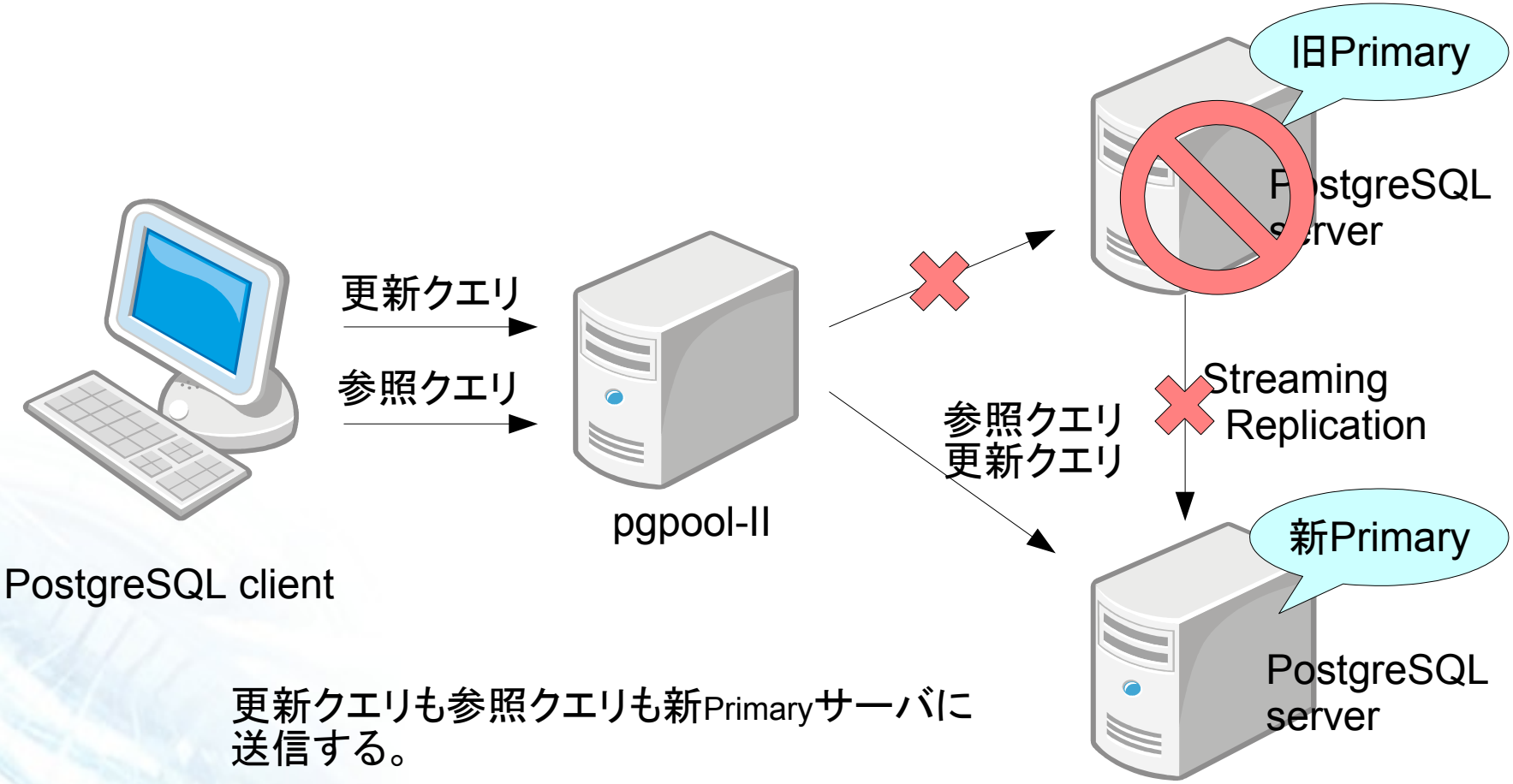


参照クエリをHot Standbyサーバに
振り分けるのをやめる。

ロードバランス詳細

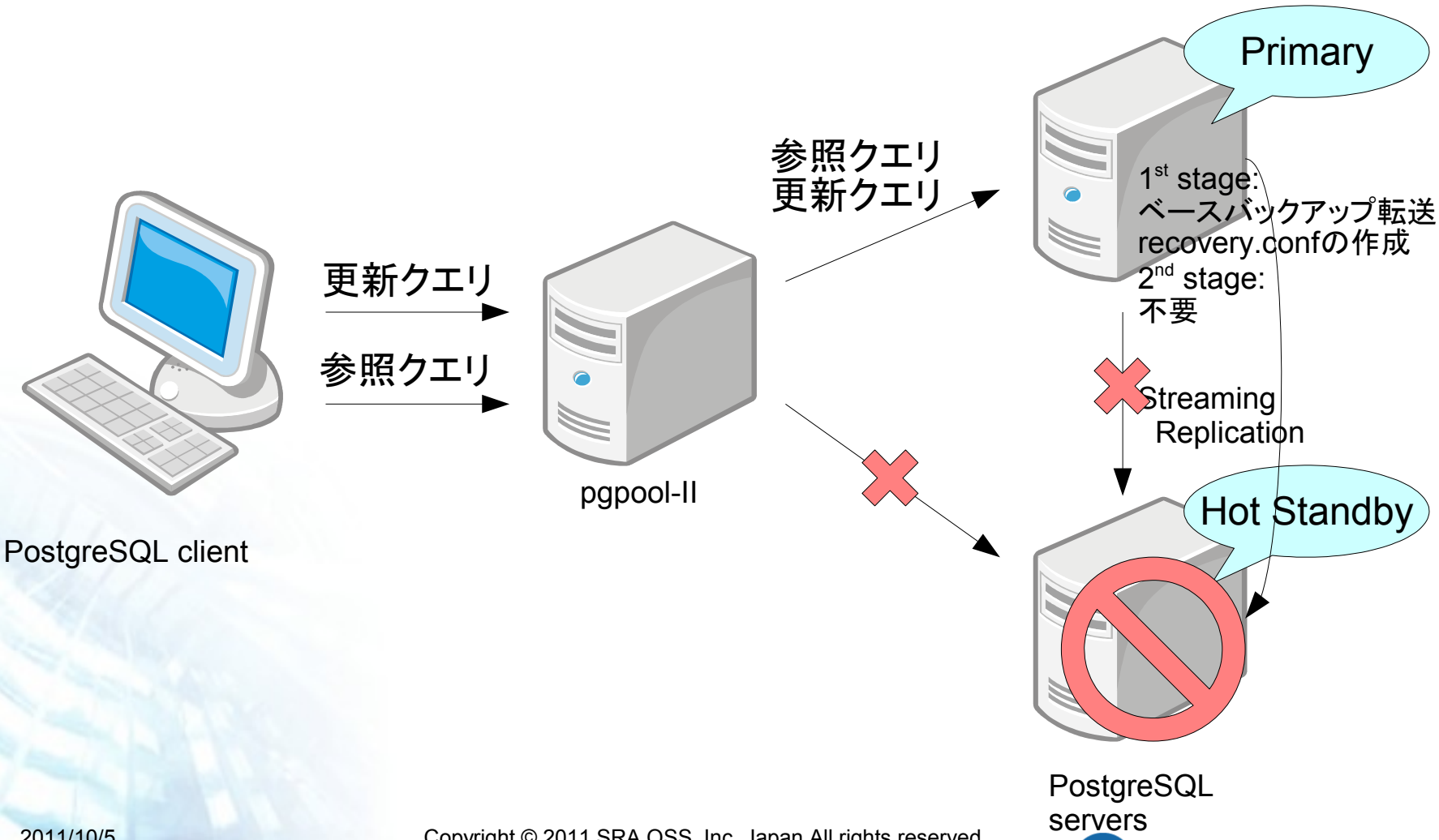
- 遅延を監視し、閾値を超えたノードには参照クエリを振り分けない
- SERIALIZABLEのトランザクションは参照クエリを振り分けない
- 明示的なトランザクション内の参照クエリもロードバランス可能に
- 一時テーブル、システムカタログを検索する参照クエリは、Primaryに送信
- トランザクション内で更新クエリが発行されたら、以後トランザクション内の参照クエリはPrimaryに送信

自動フェイルオーバー



更新クエリも参照クエリも新Primaryサーバに送信する。

オンラインリカバリを利用した Hot Standbyノードの復旧



SRA OSS, INC.

pgpool-IIの配置

- アプリケーションサーバに配置
- データベースサーバに配置
- データベースサーバに配置してHAクラスタ化
  - クライアントアプリケーションからpgpool-IIへは仮想IPを用いて接続。
  - pgpool-HA

インストール方法

- 一般的なOSSと同じインストール方法
  - ./configure
  - make
  - make install
- initスクリプトのサンプル
  - pgpool-II-3.1/redhat/pgpool.init
  - pgpool-II-3.1/redhat/pgpool.sysconfig
- 運用ユーザはPostgreSQLと合わせる

C言語関数のインストール(1/2)

- pgpool_regclass()
  - 異なるスキーマに同じ名前のテーブルが存在し、SQL文でスキーマ名を省略している場合に不具合が生じることがある。これはそれを解決するための関数。
  - pgpool-II-3.1/sql/pgpool-regclass
- オンラインリカバリ用関数
  - オンラインリカバリで使用される関数群
  - pgpool-II-3.1/sql/pgpool-recovery

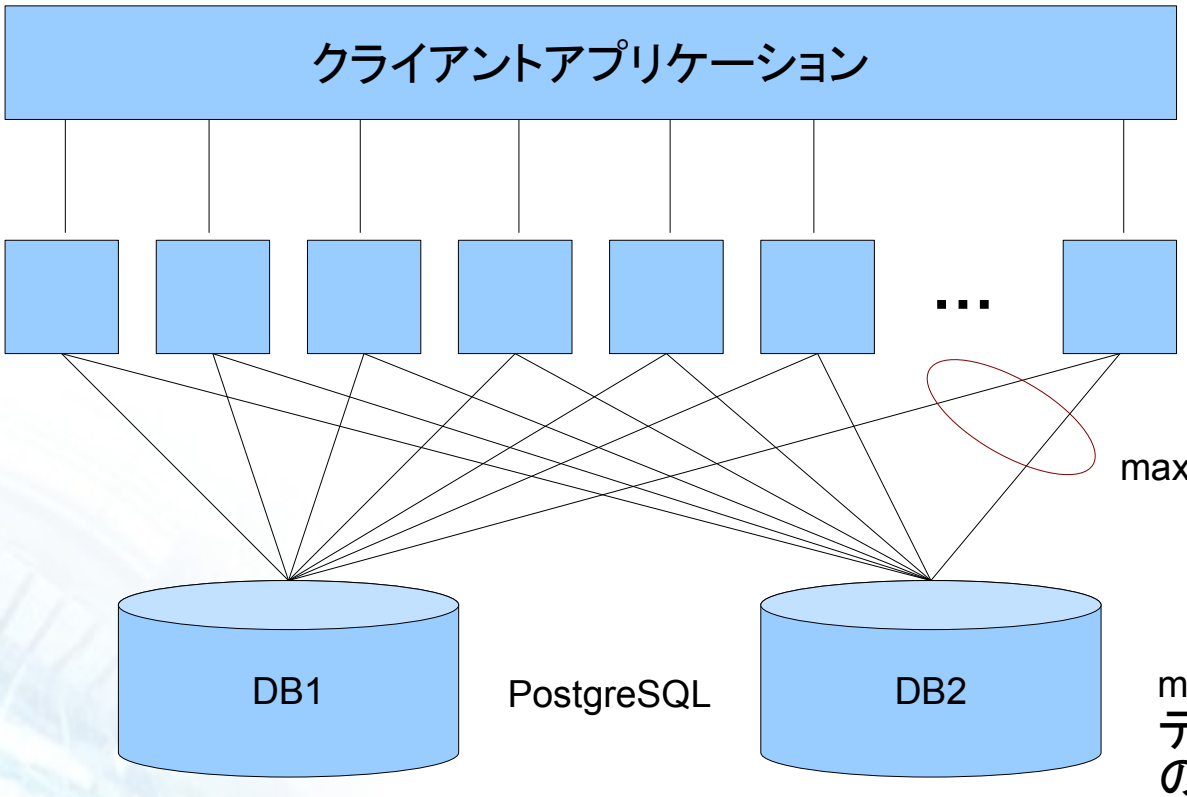
C言語関数のインストール(2/2)

- 例
  - cd pgpool-II-3.1/sql/pgpool-recovery
    make
    make install
    psql -f pgpool-recovery.sql template1
    psql -f pgpool-recovery.sql postgres
  - すべてのDBノードにインストールする。
  - pgpool_regclass()のみすべてのデータベースに作成する必要がある。

コネクションプーリング関連のパラメータ

- connection_cache = true
  コネクションプーリングの有効・無効を指定
- max_pool
  プーリングするコネクションの最大数を指定
- connection_life_time
  コネクションを切断するアイドル時間を秒単位で指定
- child_max_connections
  子プロセスを再起動する接続回数を指定

max_pool



max_poolにはユーザとデータベースの組み合わせの数を指定するとよい。

SR/HS関連のパラメータ

- master_slave_mode = true
  マスタースレーブモードの有効・無効を指定
- master_slave_sub_mode = 'stream'
  組み合わせるレプリケーションソフトウェアを指定
- delay_threshold
  スタンバイサーバにクエリを振り分けないようにする遅延バイト数を指定

レプリケーションの遅延を確認する

- log_standby_delay
  - 'none': ログ出力しない
  - 'if_over_threshold': 閾値を超えたらログ出力する
  - 'always': 常にログ出力する

```
2010-06-28 15:51:32 LOG: pid 13223: Replication of node:1 is behind 1228800 bytes
from the primary server (node:0)
2010-06-28 15:51:42 LOG: pid 13223: Replication of node:1 is behind 3325952 bytes
from the primary server (node:0)
2010-06-28 15:51:52 LOG: pid 13223: Replication of node:1 is behind 974848 bytes
from the primary server (node:0)
2010-06-28 15:52:02 LOG: pid 13223: Replication of node:1 is behind 2990080 bytes
from the primary server (node:0)
2010-06-28 15:52:12 LOG: pid 13223: Replication of node:1 is behind 901120 bytes
from the primary server (node:0)
2010-06-28 15:52:22 LOG: pid 13223: Replication of node:1 is behind 2433024 bytes
from the primary server (node:0)
```

ロードバランス関連のパラメータ

- load_balance_mode = true
  - ロードバランスモードの有効・無効を指定
- backend_weight0 = 1
  backend_weight1 = 2
  - クエリ振り分けの重みを指定
- white_function_list、black_function_list
  - 更新を伴う関数呼び出しを行うSELECT文を制御する
  - 以前は、/*REPLICATION*/、/*NO LOAD BALANCE*/といったコメントで一文ずつ対応

自動フェイルオーバー関連のパラメータ

- health_check_timeout = 20
  - 死活監視の応答がなかった場合にフェイルオーバーする期限を秒数で指定
- health_check_period = 10
  - 死活監視を行う間隔を指定
- health_check_user = 'postgres'
  - 死活監視を行うユーザを指定
- failover_command = 'failover.sh %d %P'
  - フェイルオーバー時に実行されるコマンドを指定

failover.sh

```
#!/bin/sh
# Execute command by failover.
# special values:  %d = node id
#                  %h = host name
#                  %p = port number
#                  %D = database cluster path
#                  %m = new master node id
#                  %M = old master node id
#                  %H = new master node host name
#                  %P = old primary node id
#                  %% = '%' character
failed_node_id=$1
old_primary_node_id=$2
trigger=/var/log/pgpool/trigger/trigger_file1

if [ $failed_node_id = $old_primary_node_id ];then  # master failed
    touch $trigger    # let standby take over
fi
```

オンラインリカバリ関連のパラメータ(1/2)

- recovery_user = 'postgres'
  - オンラインリカバリを行うユーザ
- recovery_password = ''
  - オンラインリカバリを行うユーザのパスワード
- recovery_1st_stage_command = 'basebackup.sh'
  - リカバリの第一段階のスクリプトを指定
- recovery_timeout = 90
  - オンラインリカバリのタイムアウト時間を指定

オンラインリカバリ関連のパラメータ(2/2)

- client_idle_limit_in_recovery = -1
  - セカンドステージ移行時に指定した秒数クライアントからクエリが来ない接続を切断する。-1を指定すると直ちに切断してセカンドステージに移行する。

オンラインリカバリ関連のファイル

- pgpool_remote_start

```
#! /bin/sh
#
# Start PostgreSQL on the recovery target node
#
if [ $# -ne 2 ]
then
    echo "pgpool_remote_start remote_host remote_datadir"
    exit 1
fi

DEST=$1
DESTDIR=$2
PGCTL=/usr/local/pgsql/bin/pg_ctl

$PGCTL -w -D $DESTDIR start 2>/dev/null 1>/dev/null < /dev/null &
```

basebackup.sh

```
#/bin/sh -x
PRIMARY_PORT=5432
STANDBY_PORT=5433
PRIMARY=/usr/local/pgsql/data
STANDBY=/usr/local/pgsql/standby

master_db_cluster=$1
recovery_node_host_name=$2
recovery_db_cluster=$3

if [ $master_db_cluster = $PRIMARY ];then
  PORT=$PRIMARY_PORT
  SOURCE_CLUSTER=$PRIMARY
  DEST_CLUSTER=$STANDBY
else
  PORT=$STANDBY_PORT
  SOURCE_CLUSTER=$STANDBY
  DEST_CLUSTER=$PRIMARY
fi

psql -p $PORT -c "SELECT pg_start_backup('Streaming Replication', true)" postgres

rsync -C -a -c --delete --exclude postgresql.conf --exclude postmaster.pid \
--exclude postmaster.opts --exclude pg_log \
--exclude recovery.conf --exclude recovery.done \
--exclude pg_xlog \
$SOURCE_CLUSTER/ $DEST_CLUSTER/

mkdir $DEST_CLUSTER/pg_xlog
chmod 700 $DEST_CLUSTER/pg_xlog
rm -f $DEST_CLUSTER/recovery.done
cat > $DEST_CLUSTER/recovery.conf <<EOF
standby_mode          = 'on'
primary_conninfo      = 'port=$PORT user=postgres'
trigger_file = '/var/log/pgpool/trigger/trigger_file1'
EOF

psql -p $PORT -c "SELECT pg_stop_backup()" postgres
```

pgpool-II 3.0から3.1への変更点

互換性のない変更

insert_lockのロック方法

- insert_lockの変遷
  - ～2.3： INSERT対象のテーブルをロック
  - 3.0： シーケンステーブルの行をロック
  - 3.1： insert_lockテーブルの行をロック
    - 各DBにpgpool_catalog.insert_lockを作成
      - sql/insert_lock.sql
    - INSERT対象のテーブルのoidがinsert_lockテーブルに自動的に追加され、その行が排他制御に使われる。
    - insert_lockテーブルがない場合は2.3までと同等の動作
- --enable-table-lock, --enable-sequence-lock

backend_socket_dirの廃止

- backend_hostnameNが'/'で始まる場合は、UNIXドメインへのパスとみなす
  - libpqインタフェースに準じた仕様
  - 空文字を指定すると/tmpを利用
- backend_socket_dirも使用可能

pgpool_walrecrunning()の廃止

- プライマリノード判断
  - 3.0: !(pg_is_in_recovery() AND pgpool_walrecrunning())
  - 3.1: !pg_is_in_recovery()
    - プライマリノードがなかなか現れない場合は、recovery_timeoutまで待ち、その間接続はブロックされる
    - recovery_timeoutを超えた場合は、ノード番号の一番小さいバックエンドをプライマリとして扱う

pool_nodes

- pool_nodesにノードIDを追加

```
=# SHOW pool_nodes;
  id  |  hostname   | port | status | lb_weight
------+-------------+------+--------+-----------
   0  | 127.0.0.1   | 5432 | 2      | 0.5
   1  | 192.168.1.7 | 5432 | 3      | 0.5
(2 lignes)
```

nextval(), setval()

- nextval(), setval()の扱い
  - 3.0： 常に書き込み関数として処理
  - 3.1： black/white_function_listの設定に従う

新機能

- syslogに対応
  - log_destination = 'syslog'
- application_nameに対応
  - コネクションが再利用されても追随
- relcache_expireパラメータを追加
  - システムカタログキャッシュの有効期限を設定可能に
- follow_master_commandパラメータを追加
  - failover_commandの後に実行するコマンドを設定可能に
- pcp_promote_node
  - プライマリノードを手動で変更可能に

- pcp_pool_statusコマンドを追加
  - SHOW pool_status;のコマンド版
- sr_check_period/user/passwordパラメータを追加
  - health_chek_period/userから遅延チェックを分離
- health_check_passwordパラメータを追加
  - 現時点では未実装
- pgpool_adm関数を追加
  - pcpコマンド(一部)のユーザ定義関数版
- UNLOGGEDテーブルに対応
  - UNLOGGEDテーブルは常にプライマリで処理
- 中文マニュアル、チュートリアルを追加

- backend_flagNパラメータを追加
  - フェイルオーバーしないノードの指定が可能に
  - 'ALLOW_TO_FAILOVER'または'DISALLOW_TO_FAILOVER'を指定
  - 'DISALLOW_TO_FAILOVER'の場合は、pcp_detach_nodeでもデタッチ不可

改良

- ストリーミングレプリケーション構成に関する変更
  - オンラインリカバリ時に子プロセスを再起動しないよう変更
    - 既存セッションはオンラインリカバリ中も処理を継続可能に
  - pcp_attache_nodeを既存のセッションを切断しないよう変更
  - 遅延チェックにpg_last_xlog_replay_location()を使用するよう変更
    - 3.0： pg_last_xlog_receive_location()
- black_function_listとwhite_function_listパラメータに正規表現が使用可能に

参考URL

- ストリーミングレプリケーションとpgpool-IIの組み合わせ、簡単設定法
  - ローカルホスト編
    - http://pgpool.projects.postgresql.org/contrib_docs/simple_sr_setting/index-ja.html
  - 複数サーバ編
    - http://pgpool.projects.postgresql.org/contrib_docs/simple_sr_setting2/index-ja.html