日本PostgreSQLユーザ会
第25回しくみ＋アプリケーション勉強会

pgpool-II と PostgreSQL ストリーミング レプリケーションを組合わせた 高性能、高可用性クラスタの検証

2013年2月9日

TIS 株式会社
中西 剛紀 （Yoshinori Nakanishi）

はじめに

- 今回は、PostgreSQLでクラスタを組む時のお話
  - PostgreSQLストリーミングレプリケーションと pgpool-II を組合せて高性能、高可用性を狙います。
  - 実際の検証結果から、このクラスタの特徴や 運用のノウハウや注意点を紹介します。
- クラスタを運用するノウハウがメインです。
  - PostgreSQLのコアな仕組みはあまり解説しません。
  - PostgreSQLストリーミングレプリケーションや pgpool-IIがどんなものか知っている前提で話します。

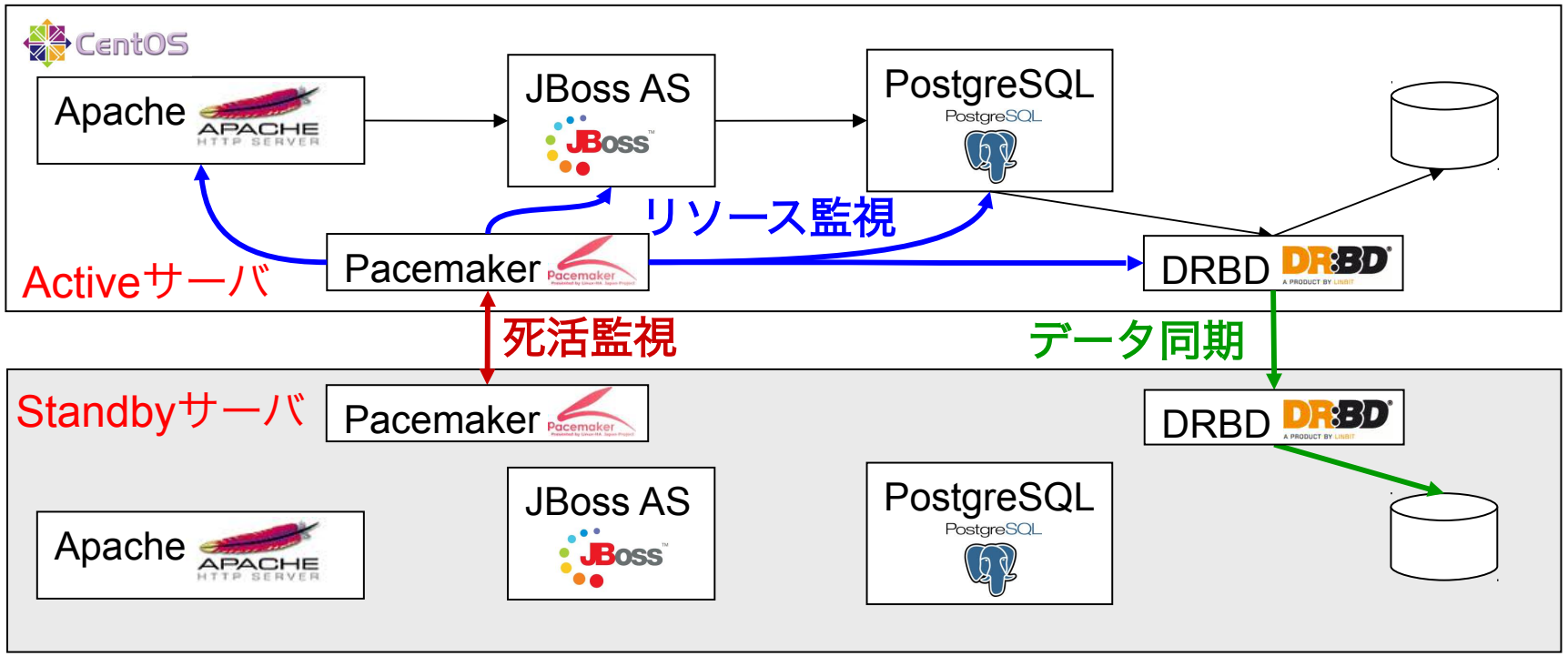
自己紹介(あんた誰よ?)

- 中西 剛紀 (なかにし よしのり)
- 所属 : TIS株式会社 戦略技術センター
  - 「海岸沿いのSIer」とか呼ばれてた会社
  - いわゆる R&D 部門です。
- OSS活用推進を担当しています。
  - 社内でOSSを使ってもらうための支援
  - 元々は商用DBMSメインのDB屋さん
  - 3年前から PostgreSQL ばっかり触ってます。

OSS推奨スタック「ISHIGAKI Template」

- 推奨OSSを組合せたアプリケーション基盤
  - 今後も発展が期待でき、商用サポートも利用可能なミドルウェアを中心に選定した OSS で構成
  - 組合せた状態での独自検証を経て、カスタマイズした設定やドキュメント等も整備
- 続きは WEB で
  - OSSマイグレーションサービス http://www.tis.jp/service_solution/ossmigration/

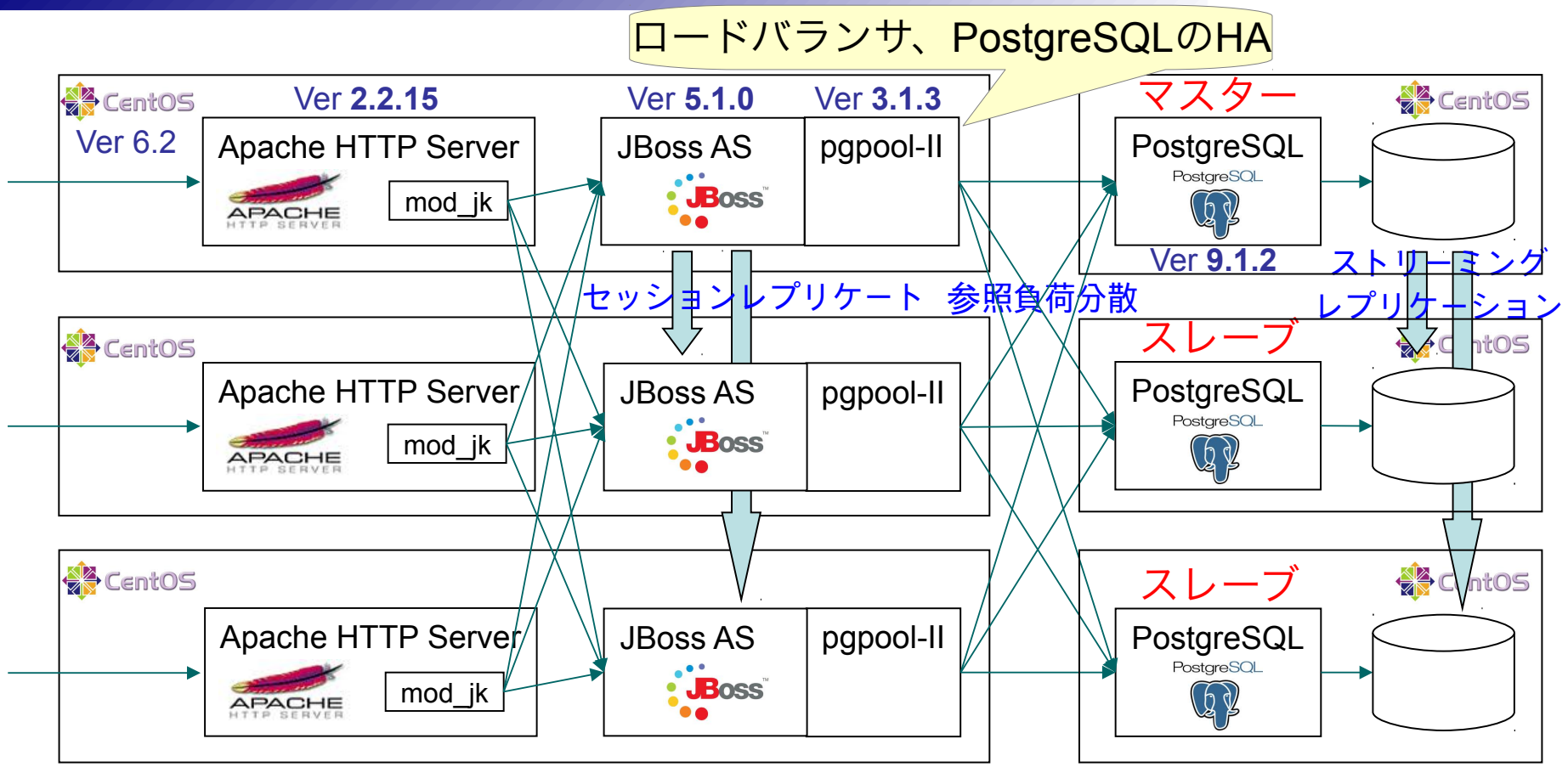
ISHIGAKI HA



Active/Standby型でサーバーを冗長化したクラスタ

○ シンプルな構成でサーバーの可用性を向上
× 性能はスケールしない

ISHIGAKI Cluster



Active/Active型でサーバーを冗長化したクラスタ

○ スケールアウトに向く構成
× 構成が複雑で運用の難易度は高い

pgpool-II の配置について

専用のサーバに配置	【長所】 • 分かりやすい • 他ソフトの影響を受けず、最も安全に運営できる。 【短所】 • サーバを1台余計に増やす必要がある。 • pgpool-II のサーバが単一障害点になる。
Webサーバやアプリケーションサーバ(APサーバ)と同居	【長所】 • Web/APサーバと pgpool-II がローカル通信で高速 • 複数のWeb/APサーバがあれば、自然と単一障害点を回避できる 【短所】 • 複数の pgpool-II を並行動作させる運用が煩雑
DBサーバと同居	【長所】 • pgpool-IIが単一障害点になることがない(?)。 • 余計なサーバを追加する必要もない。 【短所】 • アプリケーションがどのDBサーバに接続するのかを自ら判断する必要がある。

※ pgpool-II ユーザマニュアルより抜粋
http://pgpool.projects.pgfoundry.org/pgpool-II/doc/pgpool-ja.html

ISHIGAKI Cluster の検証パターン

分類	テストケース
可用性	PostgreSQLマスターサーバ障害時のフェイルオーバ
	PostgreSQL同期スレーブサーバ障害時のフェイルオーバ
	PostgreSQL非同期スレーブサーバ障害時のフェイルオーバ
	JBoss、pgpool-IIサーバ障害時のフェイルオーバ
性能	PostgreSQLサーバ数の増加による性能向上
	JBossサーバ数の増加による性能向上
運用性	停止したPostgreSQLマスターサーバのフェイルバック
	サービス稼働中のJBossサーバ追加/削除
	サービス稼働中のPostgreSQLサーバ追加/削除
	PostgreSQLのDBデータのバックアップ

本発表は、上記の検証結果の一部をご紹介します。

本発表の協賛

- 本発表は SRA OSS Inc. 日本支社様にもご協力いただきました。
  - 我々の検証結果に対するレビュー、アドバイス
- 石井様には特にお世話になりました。
  - pgpool-II の修正パッチも提供いただきました。

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ
検証2: PostgreSQLスレーブのフェイルオーバ
検証3: 停止したPostgreSQLマスタのクラスタ復帰
検証4: PostgreSQLサーバ増加時のスケールアウト
検証5: PostgreSQLサーバ減少時の運用
検証6: マルチマスタ構成でのpgpool-II watchdog
検証7: PostgreSQL 9.2でのバックアップ方式

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ

検証2: PostgreSQLスレーブのフェイルオーバ

検証3: 停止したPostgreSQLマスタのクラスタ復帰

検証4: PostgreSQLサーバ増加時のスケールアウト

検証5: PostgreSQLサーバ減少時の運用

検証6: マルチマスタ構成でのpgpool-II watchdog

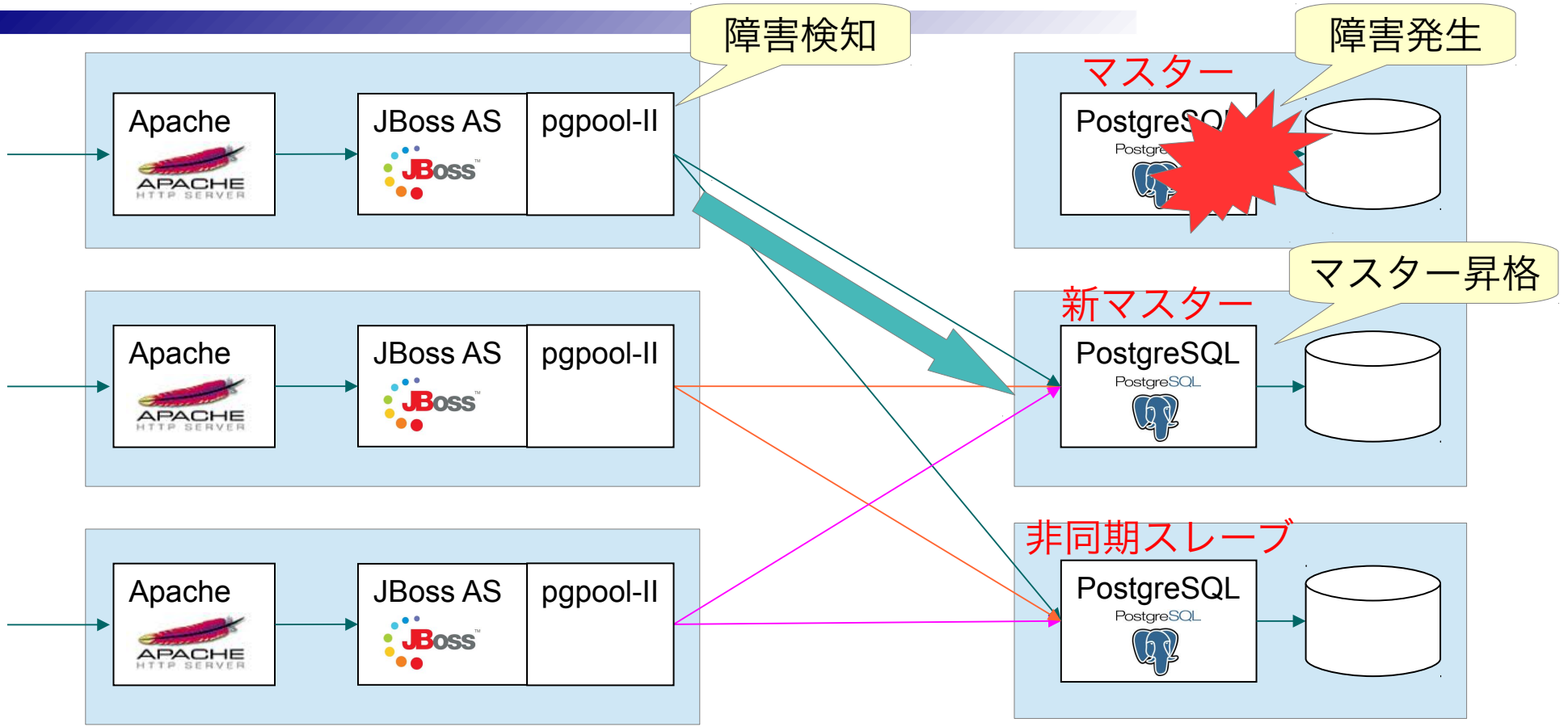
検証7: PostgreSQL 9.2でのバックアップ方式

検証で明らかにしたいこと

- PostgreSQLのマスターサーバが停まっても
  サービスは継続できるのか？

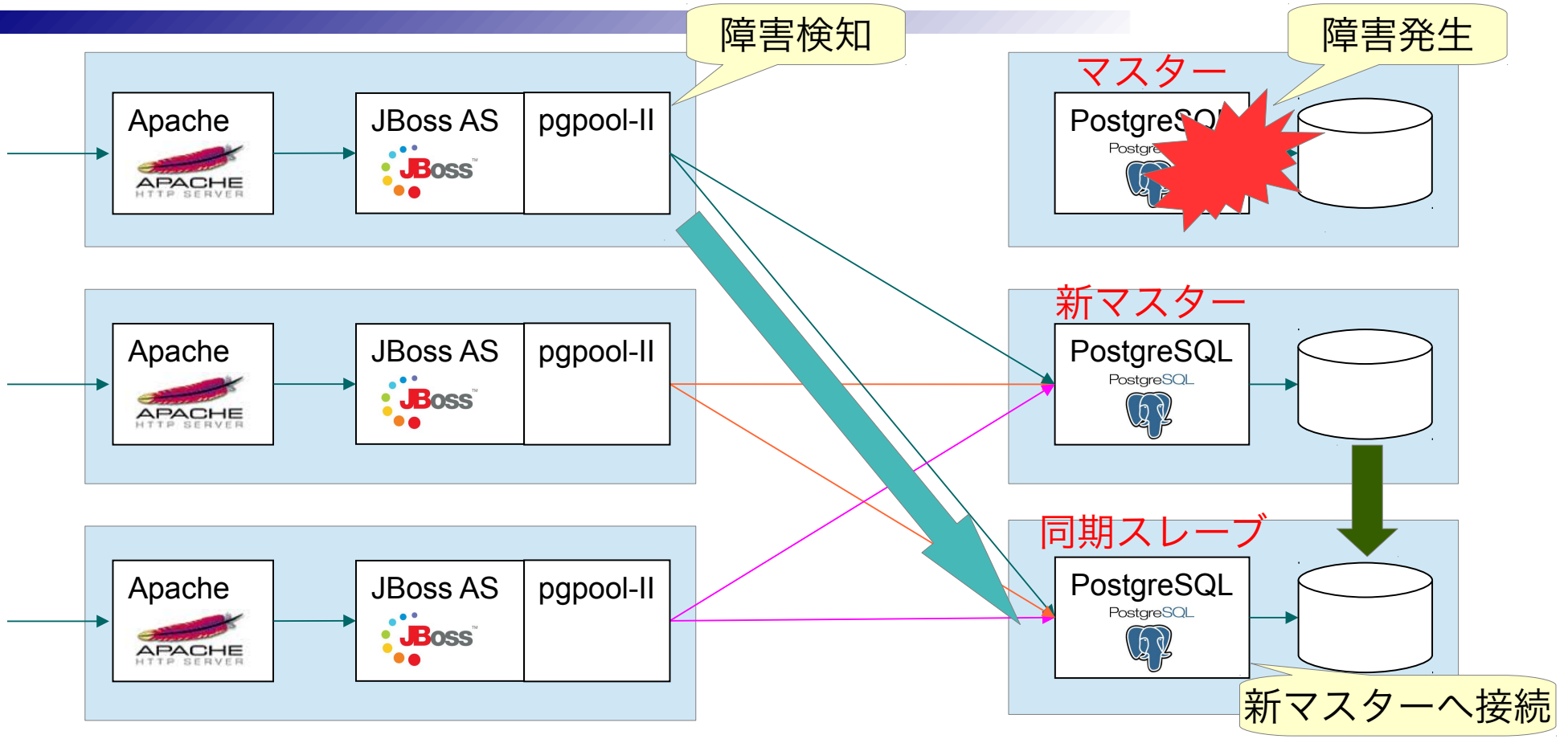
- マスターサーバをフェイルオーバしてサービス
  継続を図るのであれば、マスターサーバ停止
  ～サービス再開までどれ位時間がかかるのか？

マスターフェイルオーバの動作



1. 各APサーバ上のpgpool-IIがPostgreSQL(マスター)の障害を検知する。
   該当サーバはクラスタから切り離され、DB処理を振り分けなくなる。
2. 障害を検知したpgpool-IIがスレーブのPostgreSQLに対して、
   マスタへの昇格を実行する。
3. pgpool-IIが、残ったスレーブのPostgreSQL全てに、レプリケーション先の
   変更設定を行い、PostgreSQLを再起動する。

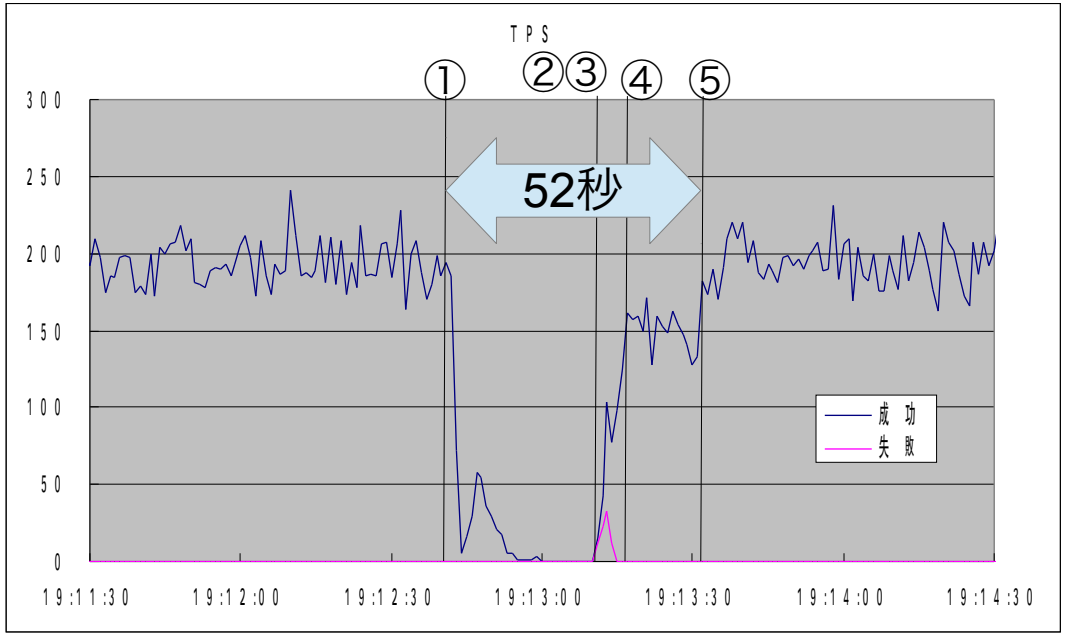
マスターフェイルオーバの動作



1. 各APサーバ上のpgpool-IIがPostgreSQL(マスター)の障害を検知する。
   該当サーバはクラスタから切り離され、DB処理を振り分けなくなる。
2. 障害を検知したpgpool-IIがスレーブのPostgreSQLに対して、
   マスタへの昇格を実行する。
3. pgpool-IIが、残ったスレーブのPostgreSQL全てに、レプリケーション先の
   変更設定を行い、PostgreSQLを再起動する。

検証結果

フェイルオーバ前後のスループット遷移



マスターサーバ停止～
サービス完全再開まで1分弱

① マスター強制停止（OSハングアップ）
マスターに割り振られた処理が待機状態になり、
スループットが急激に低下する

② マスターの停止を検知し、フェイルオーバ開始
待機状態になっていたリクエストがエラーで返され、
スループットが少し回復する。

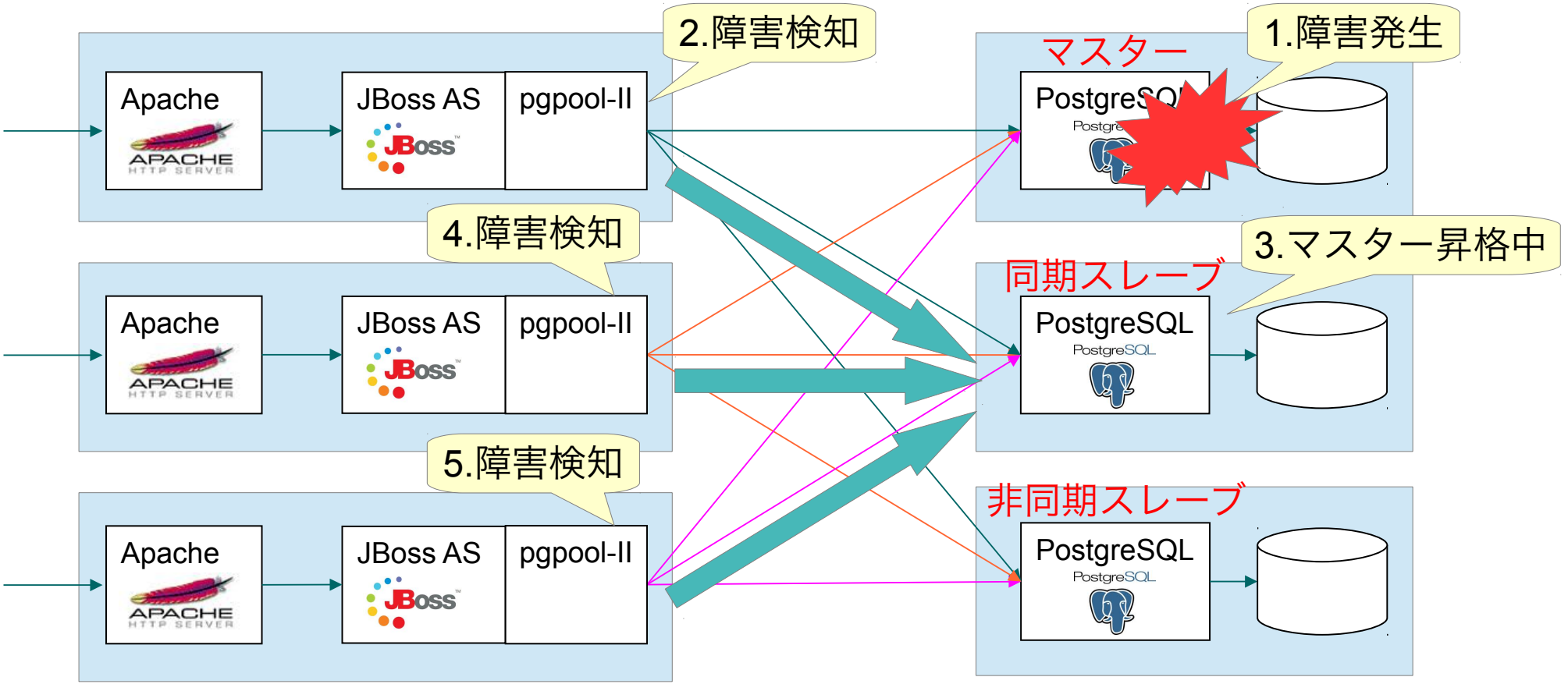
③ 新マスタの昇格完了
引き続き非同期スレーブから同期スレーブへの
昇格処理が始まる。この時点では同期スレーブが
存在せず、更新処理は待機状態になる。

④ 新同期スレーブの再起動完了
新同期スレーブサーバでの参照が可能になる。

⑤ 新同期スレーブの昇格完了
新同期スレーブのデータの再同期が終了して
昇格完了。マスタでの更新が可能になり、
フェイルオーバ前と同等のスループットに戻る。

TIPS: pgpool-IIマルチマスタ構成でのフェイルオーバ運用注意点

フェーズ1：スレーブサーバのマスタ昇格　（failover_stream.sh）

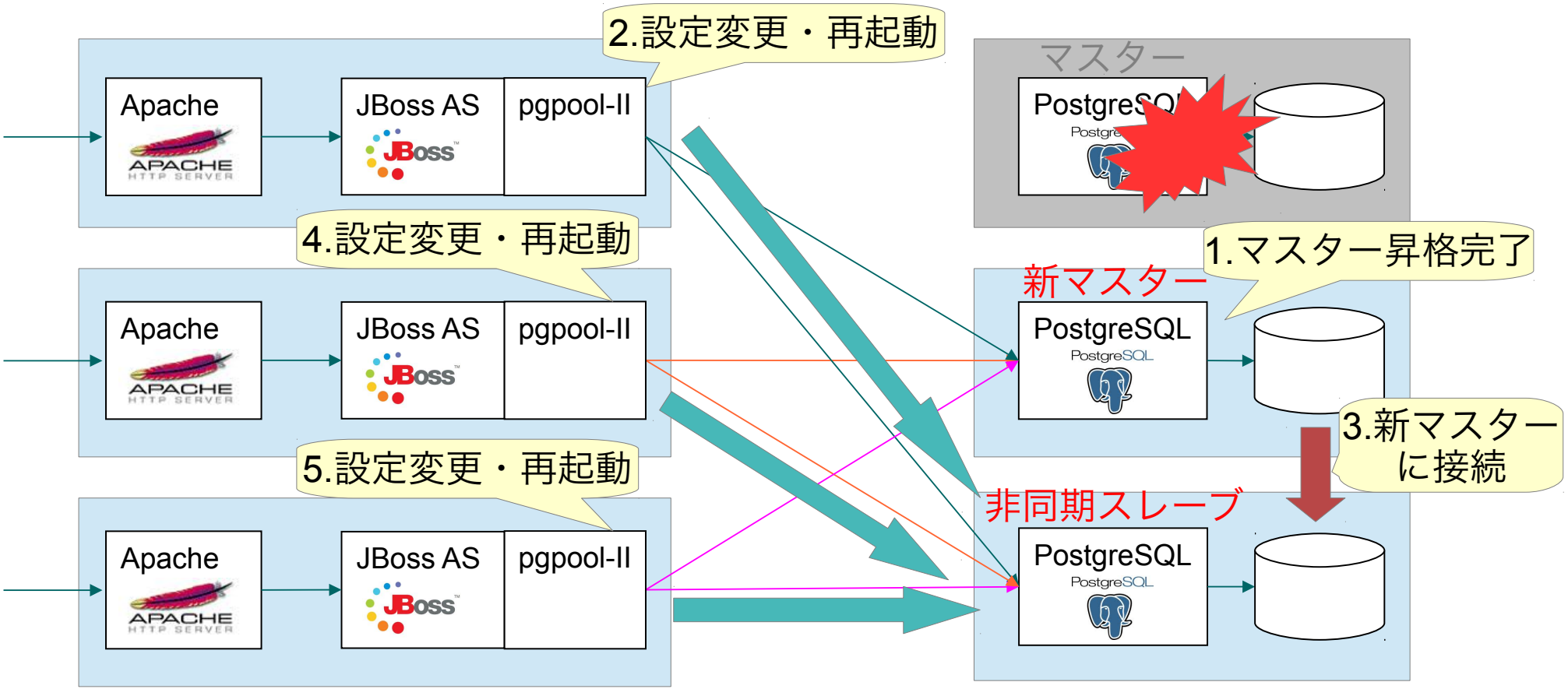


- 初回の昇格要求でマスタ化処理開始
- 以降の昇格要求はメッセージを返すのみ

➡ 問題なし

TIPS: pgpool-IIマルチマスタ構成でのフェイルオーバ運用注意点

フェーズ2: 他スレーブを新しいマスタへ再接続 （follow_master.sh）



- **無駄な再起動がかかってしまう**

⇒ ちと問題

対策案

- ロックファイルを作って余計な再起動を抑制する。
- フェーズ2で実行される follow_master.sh から呼び出される各スレーブの follow.sh に一工夫。

```
if [ -f $LOCKFILE ] then
    // ロックファイルが存在すれば、何もしない
else
    // ロックファイルがなければ、ロックファイルを作成してメイン処理を実行
    touch $LOCKFILE

    # スレーブのrecovery.confを作成
    echo "standby_mode = on" > $SLAVE_BASEDIR/$CHECKFILE
    echo "primary_conninfo = 'host=$NEW_MASTER_HOST port=5432 … '" >> $SLAVE_BASEDIR/$CHECKFILE
    echo "recovery_target_timeline = 'latest'" >> $SLAVE_BASEDIR/$CHECKFILE
    echo "restore_command = 'cp $BACKUPDIR/%f %p'" >> $SLAVE_BASEDIR/$CHECKFILE
    /usr/bin/cmp -s $SLAVE_BASEDIR/$CHECKFILE $SLAVE_BASEDIR/recovery.conf

    if [ $? -ne 0 ] ; then
        mv $SLAVE_BASEDIR/$CHECKFILE $SLAVE_BASEDIR/recovery.conf
        # スレーブのPostgreSQLを停止
        $PGCTL -m fast -w -D $SLAVE_BASEDIR stop 2>/dev/null 1>/dev/null
        # スレーブのPostgreSQLを起動
        $PGCTL -w -D $SLAVE_BASEDIR start 2>/dev/null 1>/dev/null
    else
        rm -f $SLAVE_BASEDIR/$CHECKFILE
    fi

    // ロックファイルを削除して終了
    rm $LOCKFILE
fi
```

結論

- PostgreSQLマスターがダウンしても、
  pgpool-II がキチンとフェイルオーバしてくれます。
- フェイルオーバが完了するまでの時間は約1分。
  - 障害検知までの時間は設定次第
  - 切り替え時間は直前の更新負荷量にも依存
- pgpool-II が複数稼動する環境では、
  無駄なスレーブ再起動を防止する工夫が必要。

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ

検証2: PostgreSQLスレーブのフェイルオーバ

検証3: 停止したPostgreSQLマスタのクラスタ復帰

検証4: PostgreSQLサーバ増加時のスケールアウト

検証5: PostgreSQLサーバ減少時の運用

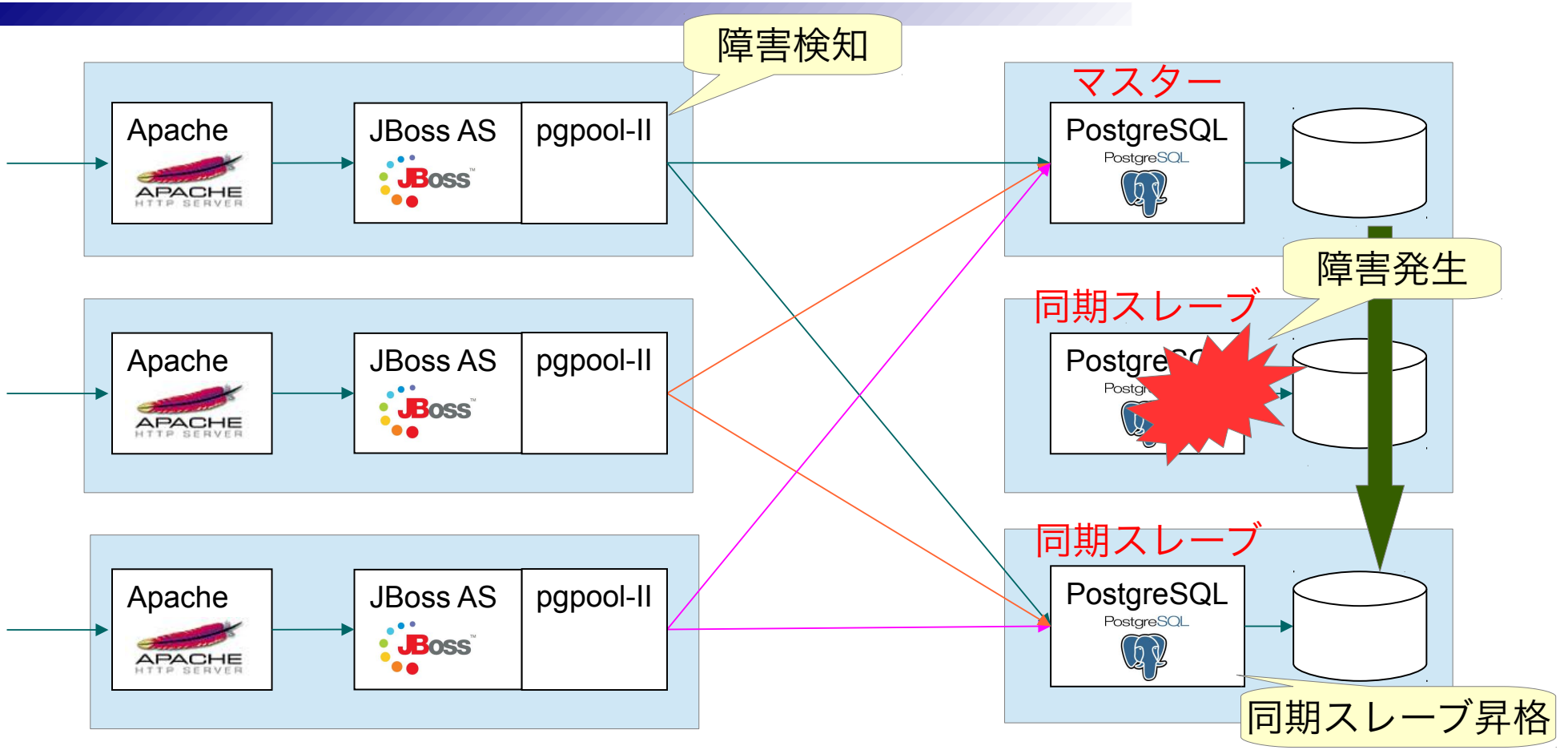
検証6: マルチマスタ構成でのpgpool-II watchdog

検証7: PostgreSQL 9.2でのバックアップ方式

検証で明らかにしたいこと

- PostgreSQL の同期/非同期スレーブが停まってもサービスは継続できるのか？

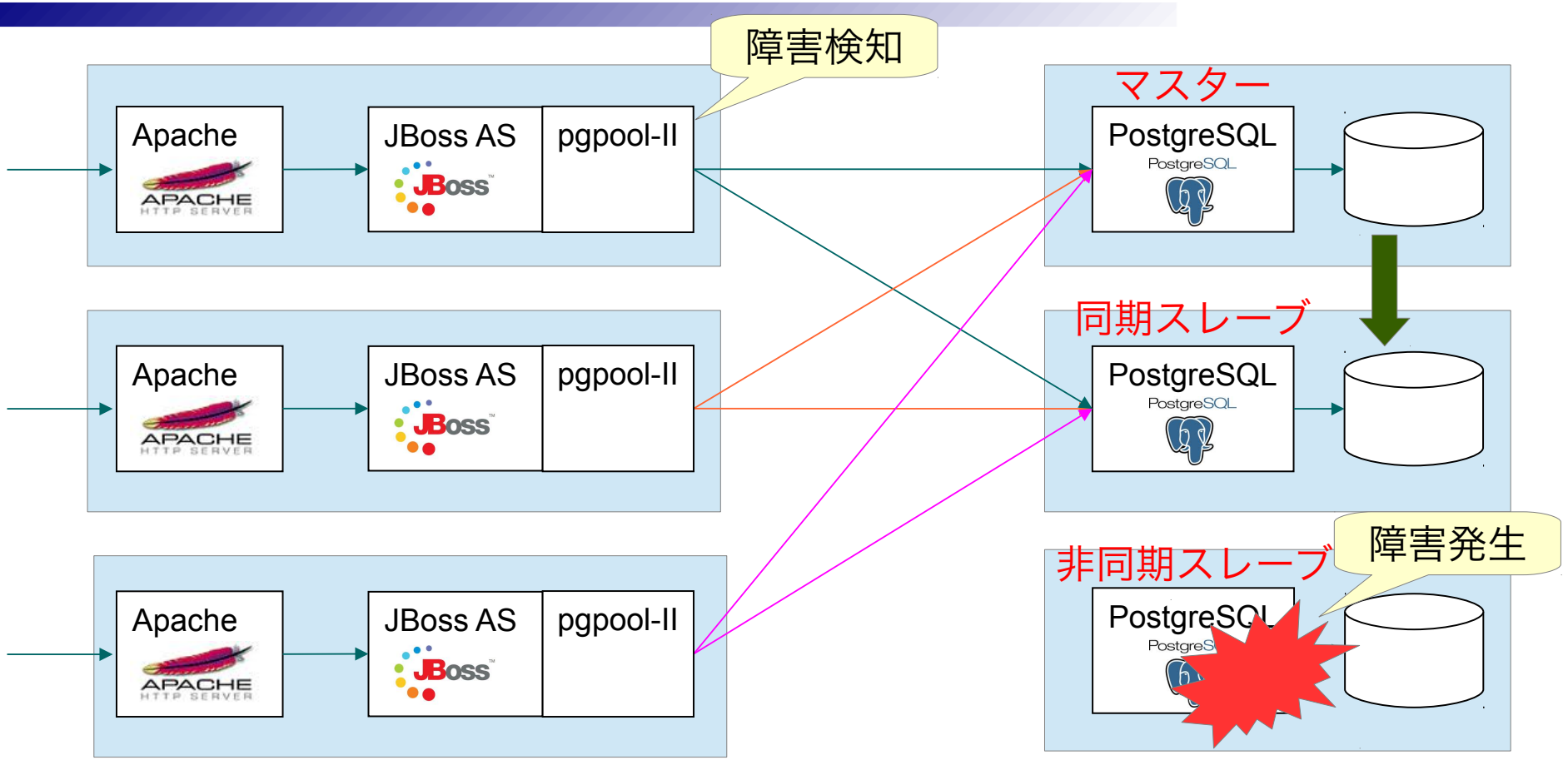
同期スレーブフェイルオーバ時の動作



1. 各APサーバ上のpgpool-IIがPostgreSQL(同期スレーブ)の障害を検知する。
   該当サーバはクラスタから切り離され、DB処理を振り分けなくなる。
2. PostgreSQLの機能により、非同期スレーブから同期スレーブへの
   昇格が実行される。

※ 非同期⇒同期スレーブの昇格は pgpool-II が行うわけではないことに注意

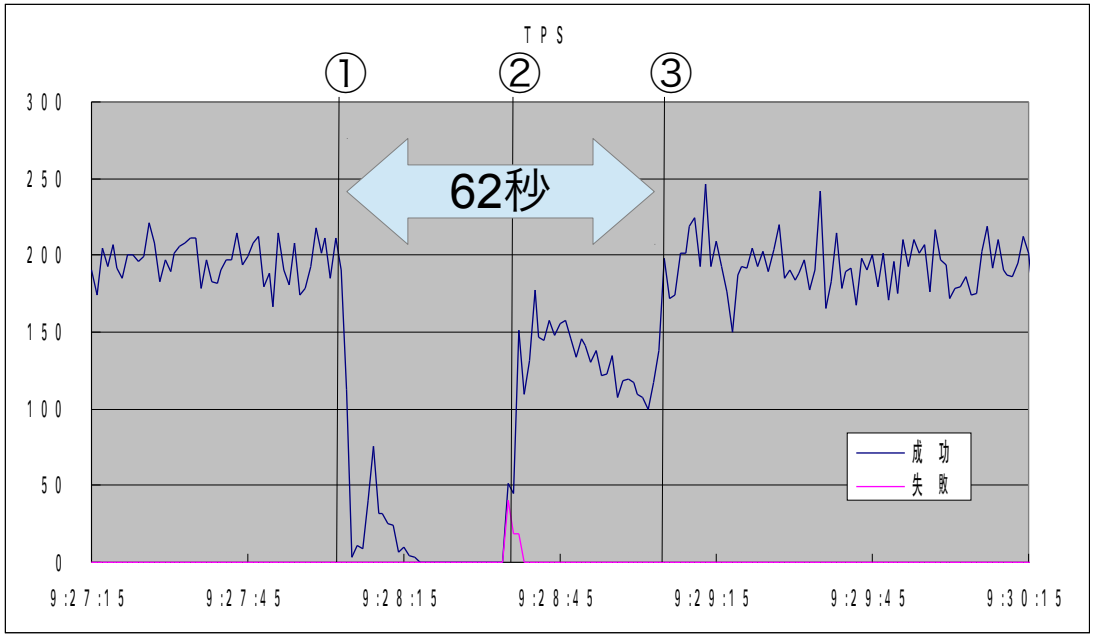
非同期スレーブ停止時の内部動作



1. 各APサーバ上のpgpool-IIがPostgreSQL(非同期スレーブ)の障害を検知する。
   該当サーバはクラスタから切り離され、DB処理を振り分けなくなる。
※ 障害が発生した非同期スレーブを切り離す以外は何もしない。

検証結果（同期スレーブ）

フェイルオーバ前後のスループット遷移



① 同期スレーブ強制停止（OSハングアップ）
同期スレーブに割り振られた処理が待機状態になり、スループットが急激に低下する

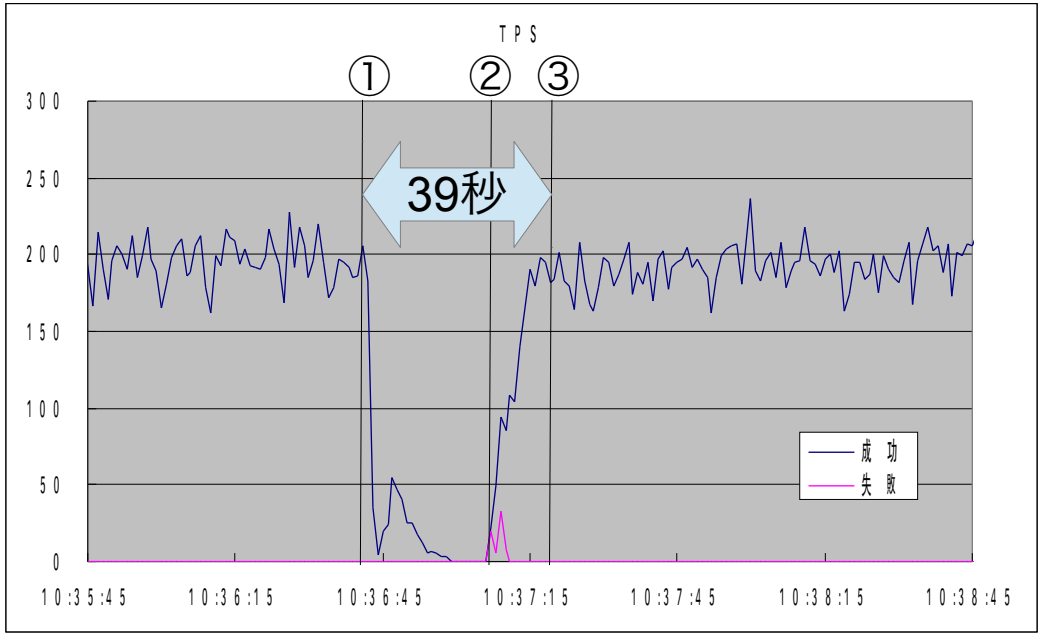
② 同期スレーブ停止を検知し、フェイルオーバ開始
参照処理のみ可能な状態になり、
スループットが少し回復する。

③ 新同期スレーブの昇格完了
更新可能な状態になり、スループットが回復する。

マスターサーバ停止～サービス完全再開まで約1分

検証結果（非同期スレーブ）

フェイルオーバ前後のスループット遷移



① 非同期スレーブ強制停止（OSハングアップ）
非同期スレーブに割り振られた処理が待機状態になり、スループットが急激に低下する

② 非同期スレーブ停止を検知
非同期スレーブを切り離して、スループットが回復し始める。

③ スループットが障害前レベルに回復する。

マスターサーバ停止～サービス完全再開まで約40秒

結論

- 同期スレーブのフェイルオーバもキチンとできます。
  - まあ、PostgreSQL側でやっているわけですが。
- 非同期スレーブは切り離されるだけなので、
  障害発生時の影響が最も小さい。

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ

検証2: PostgreSQLスレーブのフェイルオーバ

検証3: 停止したPostgreSQLマスタのクラスタ復帰

検証4: PostgreSQLサーバ増加時のスケールアウト

検証5: PostgreSQLサーバ減少時の運用

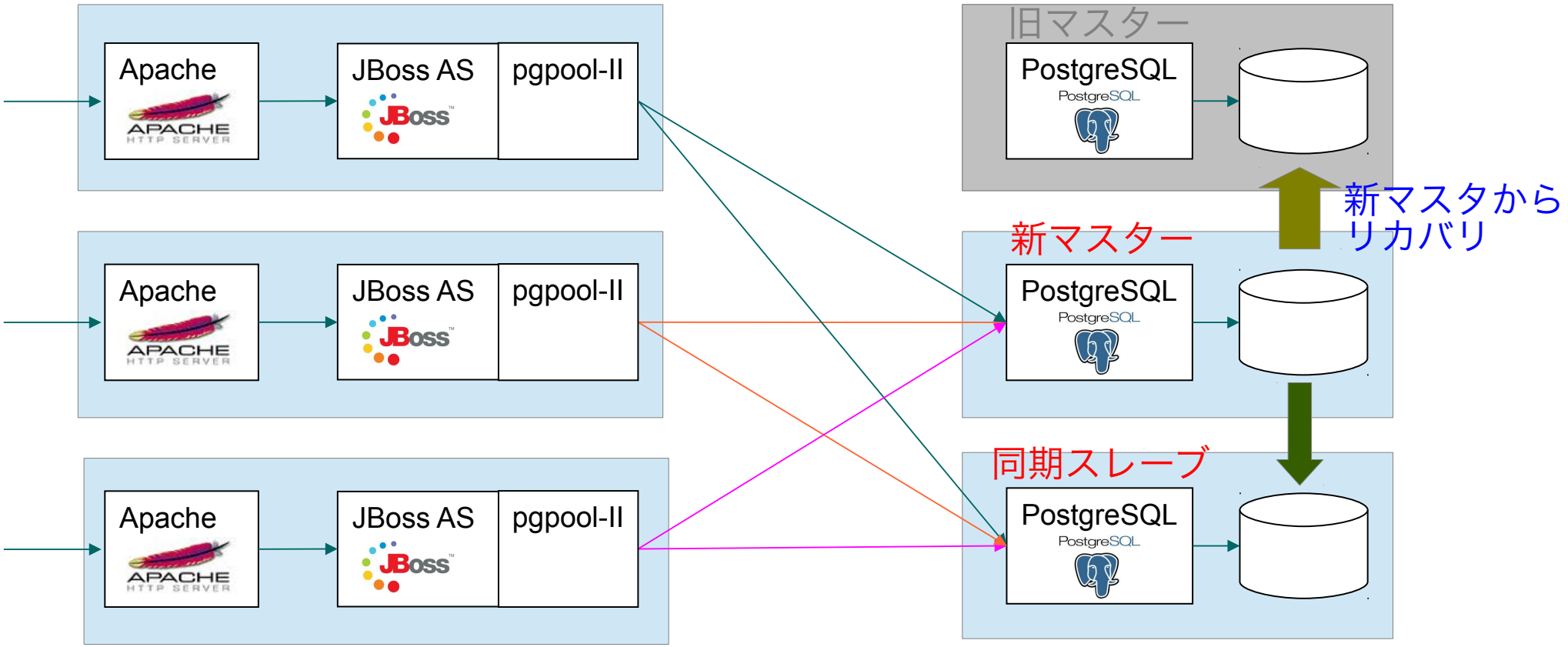
検証6: マルチマスタ構成でのpgpool-II watchdog

検証7: PostgreSQL 9.2でのバックアップ方式

検証で明らかにしたいこと

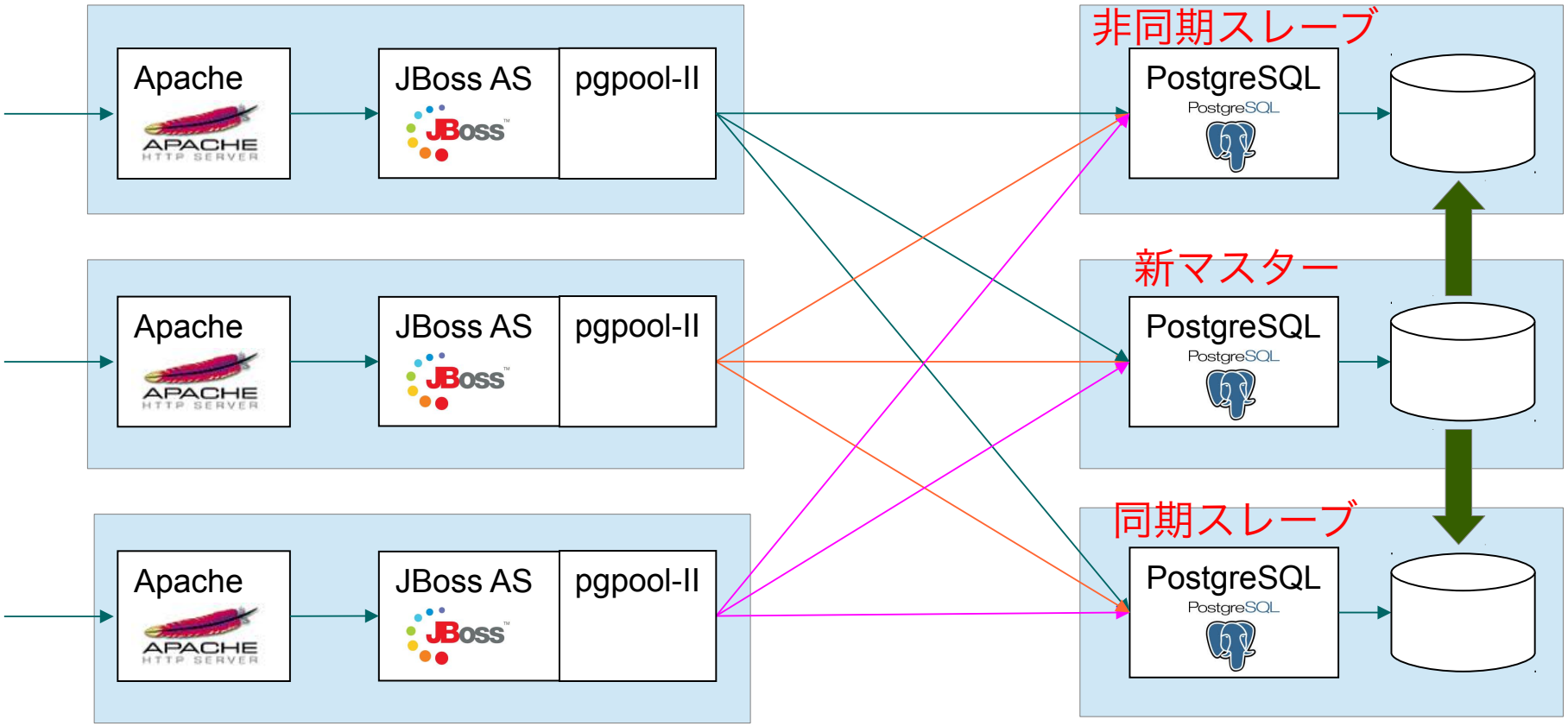
- 停止したマスタをクラスタに復帰できるのか？
- サービスを稼動させた状態でも、クラスタに復帰できるのか？

フェイルバック時の内部動作



1. 旧マスターのDBを一度消去し、新マスターのDBからリカバリする。
   ※pg_basebackup コマンドを使う。
2. 非同期スレーブとして起動し、新マスターとのレプリケーションを開始する。
   ※recovery.conf を作成して起動する。
3. 2.で起動した非同期スレーブを pgpool-II 配下に参加させる。
   ※pgpool-II の pcp_attach_node コマンドを使う。

フェイルバック時の内部動作



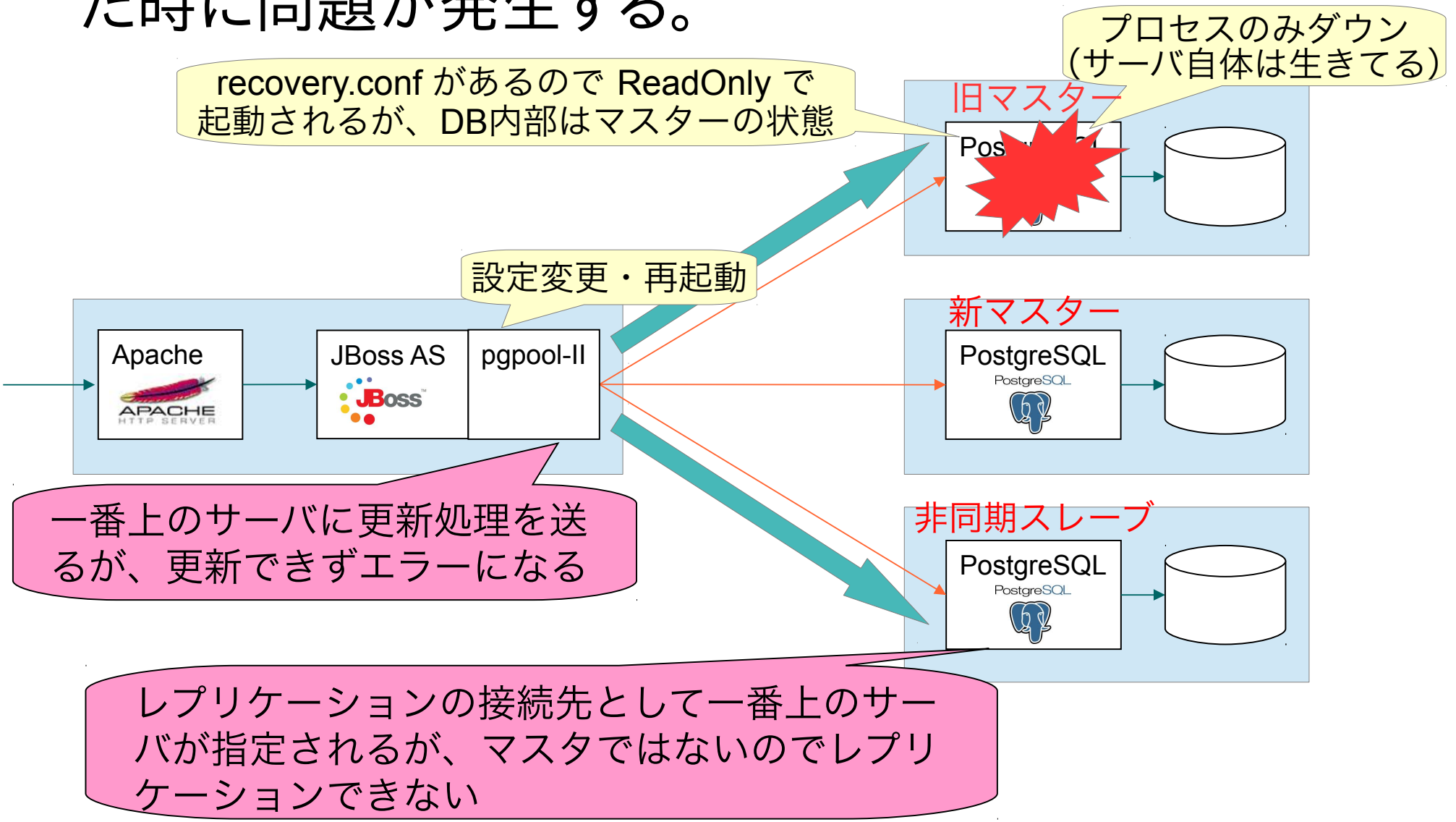
1. 旧マスターのDBを一度消去し、新マスターのDBからリカバリする。
   ※pg_basebackup コマンドを使う。
2. 非同期スレーブとして起動し、新マスターとのレプリケーションを開始する。
   ※recovery.conf を作成して起動する。
3. 2.で起動した非同期スレーブを pgpool-II 配下に参加させる。
   ※pgpool-II の pcp_attach_node コマンドを使う。

検証結果

- 前述の手順で、ダウンしたマスタを復帰できた。
- サービスを稼動させた状態で復帰させてもOK
  - クライアントから見たスループットに変化はなかった。
  - 復帰作業中、特にエラーも発生しなかった。

TIPS: フェイルバック時の注意事項

- PostgreSQLマスターのプロセスのみがダウンした時に問題が発生する。



対策案

- 自ノードが旧マスタであれば、PostgreSQLを
  自動的に再起動しないようスクリプトを改修
- フェーズ2で実行する follow_master.sh に一工夫。

```
# これから実行するサーバが旧マスターであれば、何も行わない。
if [ $NODE_ID = $OLD_MASTER_NODE_ID ] ; then
        exit 1
fi

# 旧マスター以外のスレーブは follow.sh を呼び出し、recovery.conf の作成と再起動を行う。
MSG=`ssh $SLAVE_HOST /etc/ishigaki/postgresql/follow.sh
            $NEW_MASTER_HOST $SLAVE_HOST $SLAVE_BASEDIR 2>&1`
```

結論

- 停止したマスターのフェイルバックもできます。
- マスタープロセスがダウンするケースに備えて、運用スクリプトを実装しておきましょう。

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ

検証2: PostgreSQLスレーブのフェイルオーバ

検証3: 停止したPostgreSQLマスタのクラスタ復帰

検証4: PostgreSQLサーバ増加時のスケールアウト

検証5: PostgreSQLサーバ減少時の運用

検証6: マルチマスタ構成でのpgpool-II watchdog

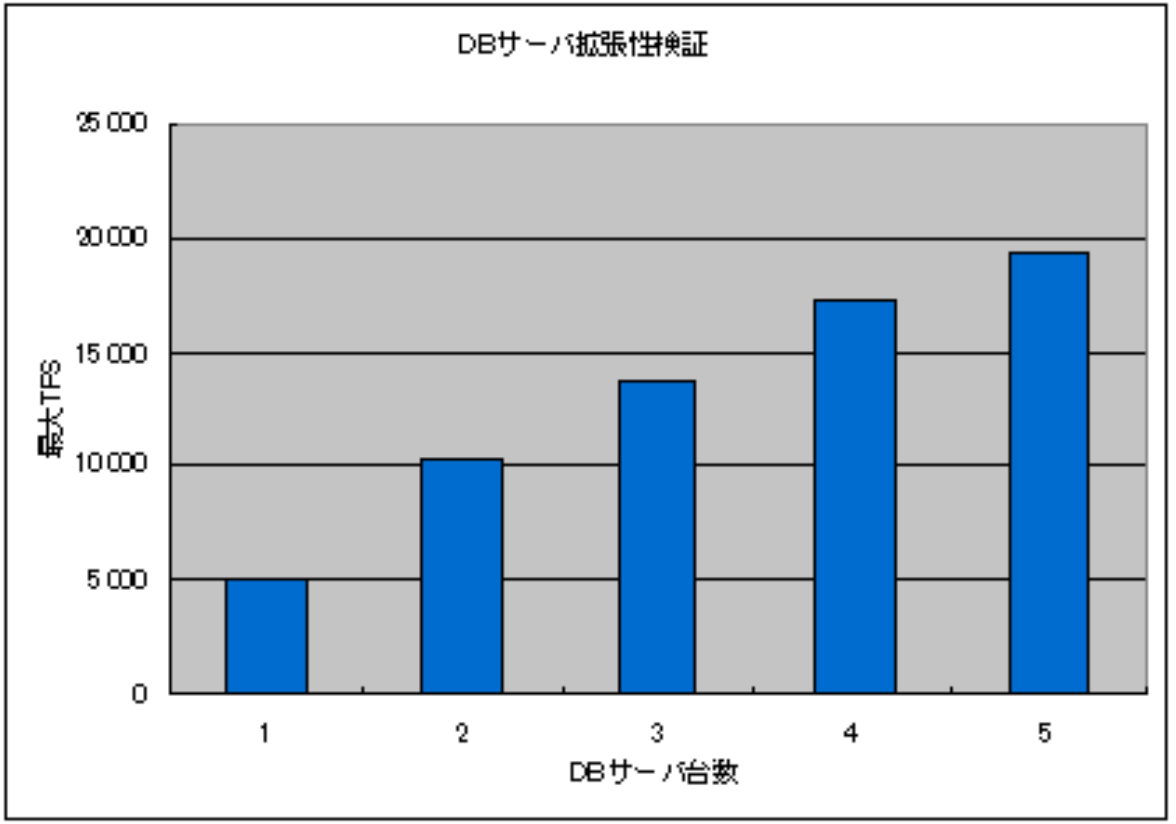
検証7: PostgreSQL 9.2でのバックアップ方式

検証で明らかにしたいこと

- サーバー数に応じて参照性能はスケールアウトするのか？

検証結果

- サーバ数1台⇒5台でのスループット



	1台	2台	3台	4台	5台
最大TPS	5099.08	10330.90	13725.50	17132.00	19263.91
1台当たりのTPS	5099.08	5165.45	4575.17	4283.00	3852.78
TPSの伸び率	1.00	2.03	2.69	3.36	3.78

測定方法のこぼれ話

- 性能測定方法
  - 使用アプリケーション: JPetStore
  - 使用データベース: JPetStore のスキーマを利用
  - データ件数: ユーザ 1000, 商品 250, 注文 10万
  - 実行SQL

```
SELECT
    I.ITEMID,LISTPRICE,UNITCOST,SUPPLIER,I.PRODUCTID,NAME,DESCN,
    CATEGORY,I.STATUS,ATTR1,ATTR2,ATTR3,ATTR4,ATTR5,QTY,O.ORDE
    RDATE, A.email
FROM ITEM I, INVENTORY V, PRODUCT P, ORDERS O, LINEITEM L,
    ACCOUNT A
WHERE
    P.PRODUCTID = I.PRODUCTID and I.ITEMID = V.ITEMID
    and I.ITEMID = L.ITEMID and O.ORDERID = L.ORDERID
    and A.USERID=O.USERID and O.USERID = ${user_name} :
```

意図的に高負荷の参照SQLを発行

結論

- 参照性能はスケールする。
- かどうかはケースバイケース。。。
  - スケールしやすい / しにくいケースがあります。
  - 適用シーンに合わせた検証を事前に行いましょう。

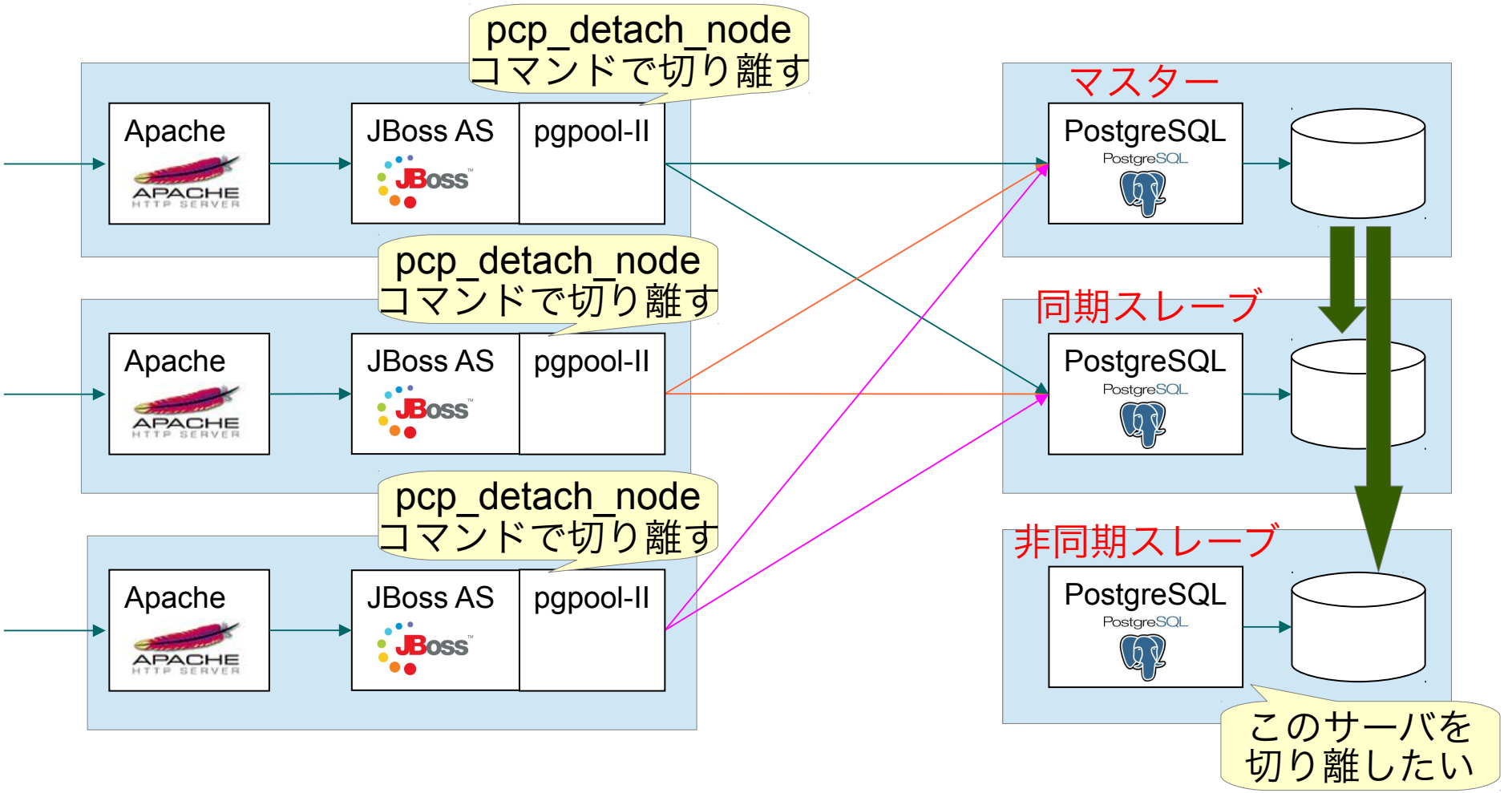
報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ
検証2: PostgreSQLスレーブのフェイルオーバ
検証3: 停止したPostgreSQLマスタのクラスタ復帰
検証4: PostgreSQLサーバ増加時のスケールアウト
検証5: PostgreSQLサーバ減少時の運用
検証6: マルチマスタ構成でのpgpool-II watchdog
検証7: PostgreSQL 9.2でのバックアップ方式

検証で明らかにしたいこと

- サーバをクラスタからスマートに切り離せるか？

シュリンク時の運用手順



1. 対象の非同期スレーブを pgpool-II のSQL振り分け対象から外す。
   ※pcp_detach_node コマンドを使う。
2. 非同期スレーブの PostgreSQL サーバを停止する。

検証結果

- 負荷をかけた状態で pcp_detach_node コマンドを実行し、PostgreSQL非同期スレーブを切り離そうとすると、エラーが返されてしまう。

- pcp_detach_node コマンドを実行すると、pgpool-II は全てのバックエンドプロセスを再起動してしまうのが原因。

結論

- サービス稼動状態で、クライアントに影響を与えずに PostgreSQL サーバをシュリンクさせることはできない。
- ただ、サービス稼動したままシュリンクさせたいという要望は強くない気がする。

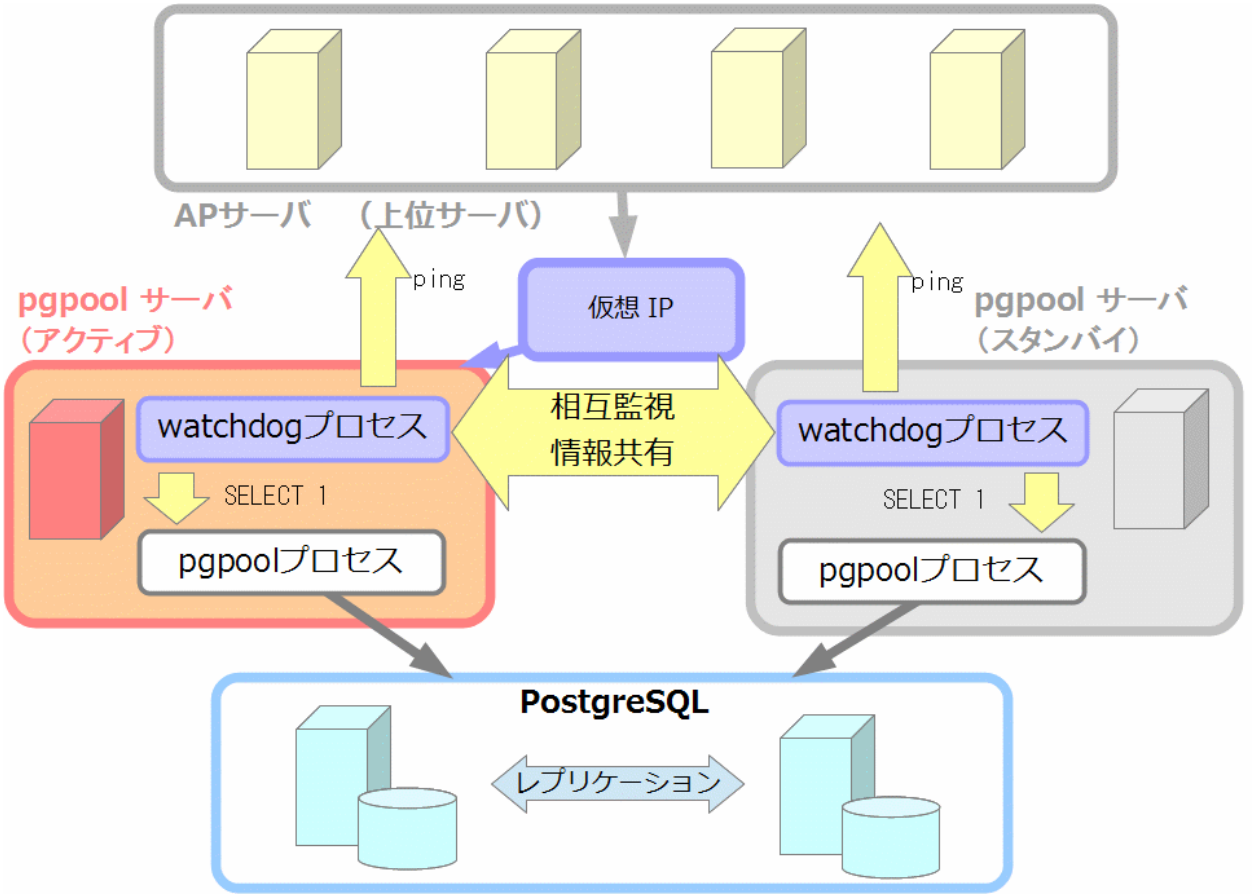
ここからは PostgreSQL 9.2、
pgpool-II 3.2 の新機能に
フォーカスを当ててみます。

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ
検証2: PostgreSQLスレーブのフェイルオーバ
検証3: 停止したPostgreSQLマスタのクラスタ復帰
検証4: PostgreSQLサーバ増加時のスケールアウト
検証5: PostgreSQLサーバ減少時の運用
検証6: マルチマスタ構成でのpgpool-II watchdog
検証7: PostgreSQL 9.2でのバックアップ方式

pgpool-II watchdog とは?

- pgpool-II 3.2 で追加された新機能
- pgpool-II の死活監視、サーバ間の情報共有が可能



詳細は http://lets.postgresql.jp/documents/technical/pgpool-II_3.2/watchdog を参照

マルチマスタ構成と watchdog

- マルチマスタ構成での pgpool-II 運用の課題
  - マスターフェイルオーバ時の無駄なスレーブ再起動を抑制するため、自前でロックファイルを制御するといった運用の工夫が必要だった。（検証1参照）
  - 複数の pgpool-II が互いに連携しないのが原因。
- watchdog 機能に寄せる期待
  - サーバ情報を共有することで、フェイルオーバ時の運用をシンプルにできるのでは？

検証で明らかにしたいこと

- pgpool-II の watchdog 機能は、pgpool-II サーバを複数並べたマルチマスタ構成でも有効なのか？

検証結果

- PostgreSQLサーバのダウン検出時には、全ての pgpool-II サーバで、フェイルオーバ用スクリプトが実行される仕様だった。
- つまり、フェイルオーバスクリプトのロック制御は必要だということ。

結論

- Ver 3.2.1 ではマルチマスタ構成において、watchdog 機能のメリットを十分享受できず、利用を推奨しない。
- pgpool-II を Active/Standby 構成で運用するケースでの利用を推奨する。

報告ケース一覧

検証1: PostgreSQLマスタのフェイルオーバ
検証2: PostgreSQLスレーブのフェイルオーバ
検証3: 停止したPostgreSQLマスタのクラスタ復帰
検証4: PostgreSQLサーバ増加時のスケールアウト
検証5: PostgreSQLサーバ減少時の運用
検証6: マルチマスタ構成でのpgpool-II watchdog
検証7: PostgreSQL 9.2でのバックアップ方式

PostgreSQLのバックアップ方式

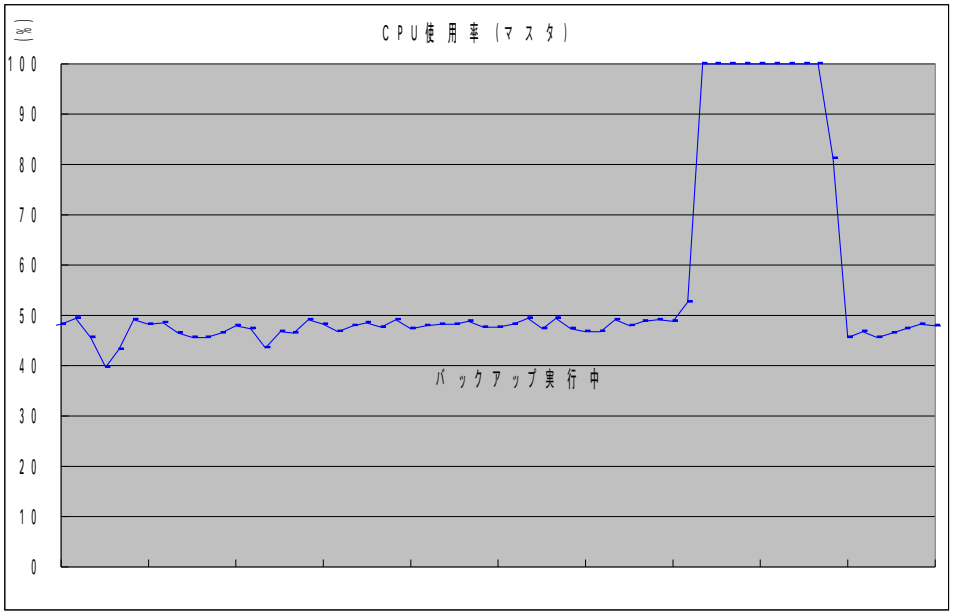
- PostgreSQL のバックアップ方式
  - オンラインで論理バックアップ　⇒ 性能 ×
  - オフラインでマスターから取得 ⇒ サービス継続性 ×
  - オンラインでマスターから取得 ⇒ マスター負荷 ×
- 9.2より、オンラインでスレーブから取得が可能に
  - バックアップ中のマスター負荷軽減を期待
  - pg_basebackup コマンドで実施すること
    pg_start_backup(), pg_stop_backup() は NG

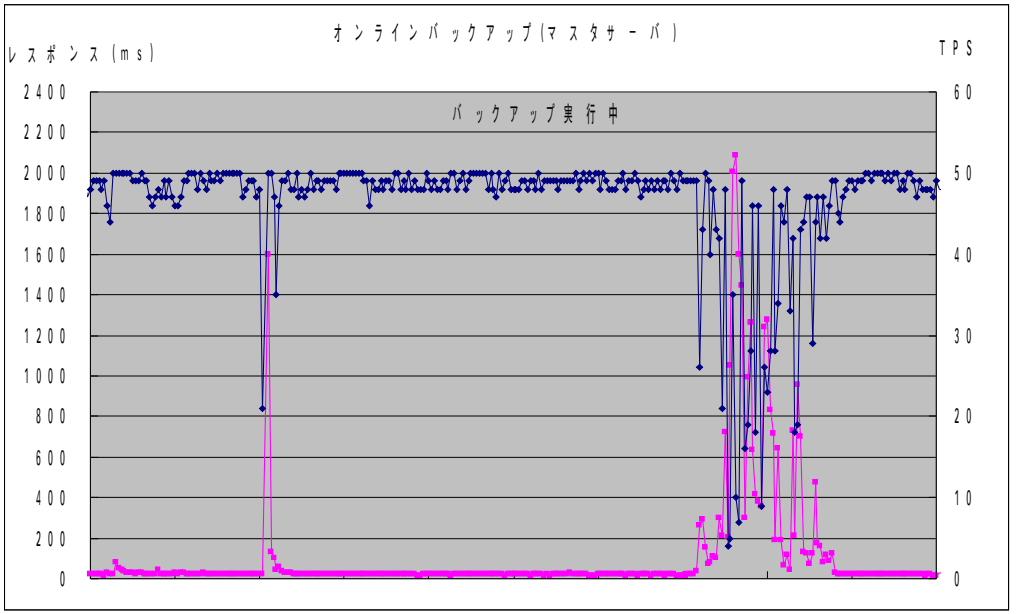
検証で明らかにしたいこと

- サービス稼動中に、PostgreSQL スレーブサーバからバックアップが問題なく取得できるか。
- PostgreSQLスレーブサーバからバックアップを取得する間のマスターサーバへ与える影響は？

検証結果

- マスターサーバからオンラインバックアップ取得
  - pg_basebackup コマンドを使用
  - NFSマウント先のディスクへバックアップ

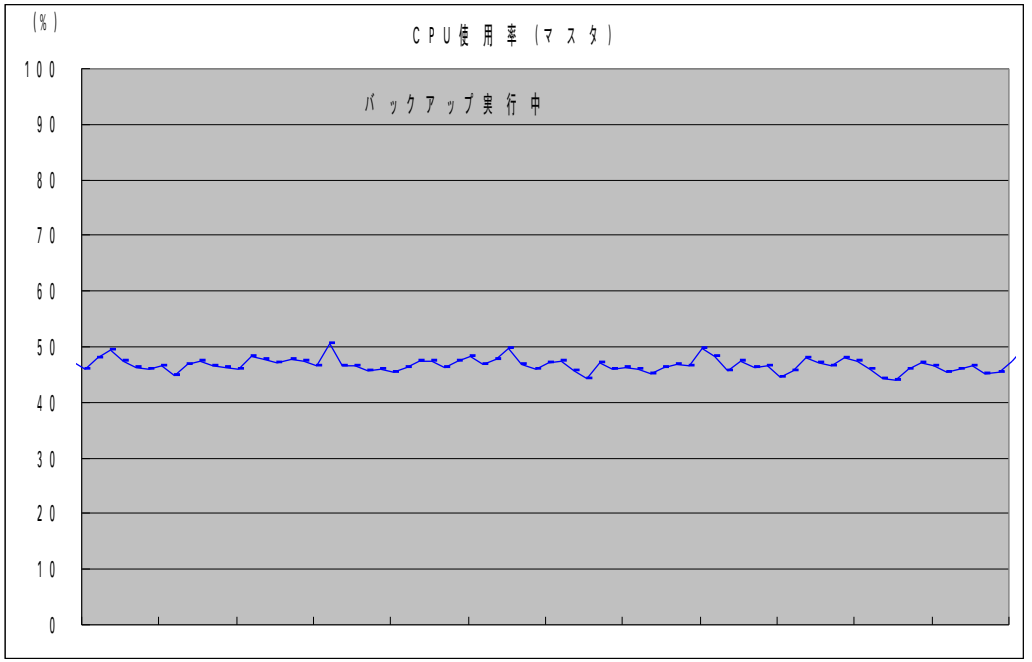


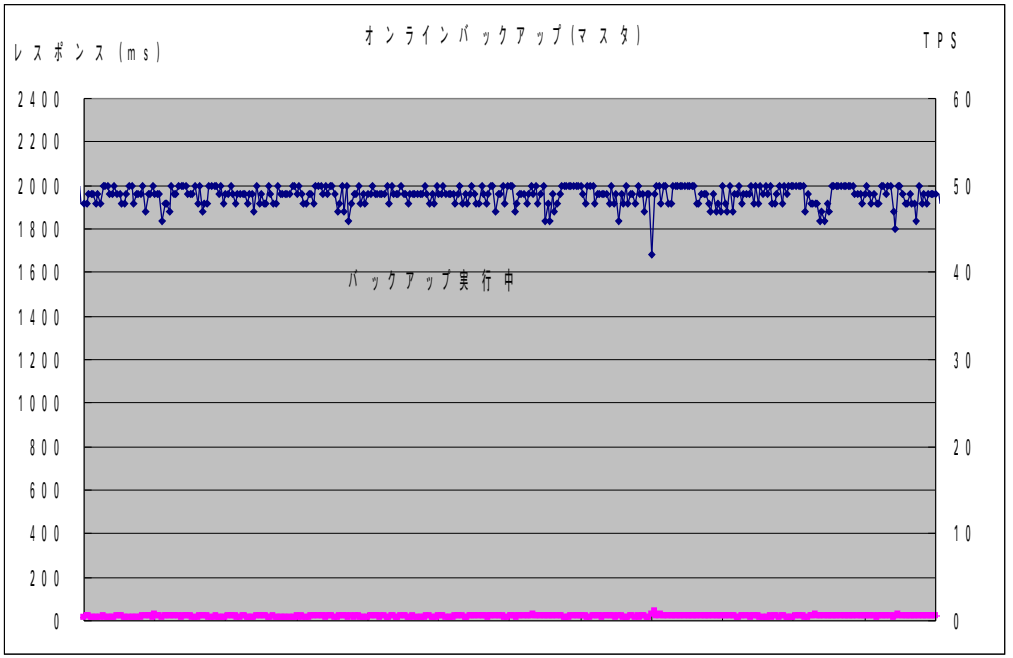


バックアップ処理がマスターサーバのCPUを占有し、
サービス全体のスループットを下げてしまっている。

検証結果

- スレーブサーバからオンラインバックアップ取得
  - pg_basebackup コマンドを使用
  - NFSマウント先のディスクへバックアップ





バックアップ処理がサービスへ悪影響を与えていない

結論

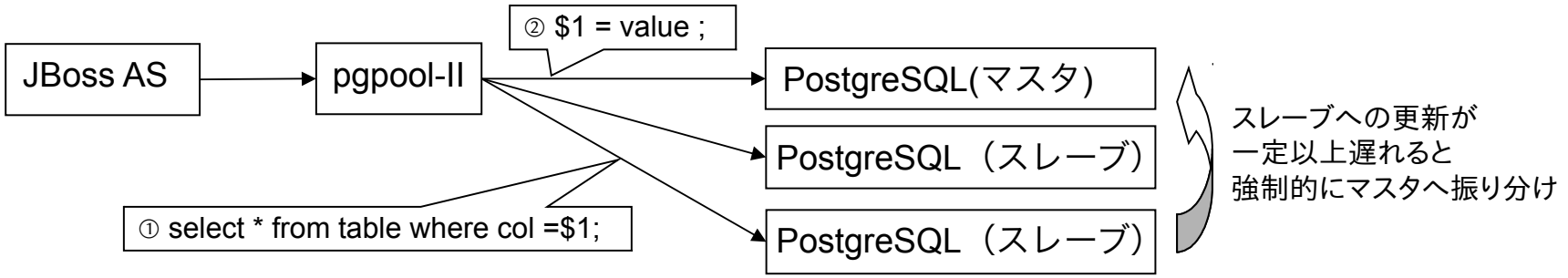
- スレーブサーバからバックアップを取得すれば、PostgreSQLストリーミングレプリケーション構成で重要なマスターサーバに悪影響を及ぼすことなくバックアップが取得できるため、お勧めです。

おまけ
Prepared Statement 使用時の問題点

- 検証中、「Prepared Statement が存在しない」旨のエラーが PostgreSQLサーバで発生した。

```
ERROR:  portal "" cannot be run
ERROR:  bind message supplies 0 parameters, but prepared statement "" requires 1
```

- 発生条件：delay_threshold 設定を有効にした時



delay_threshold 有効時は、①が届いていないマスタへ②を発行することがあり得るために Prepared Statement が存在しないエラーが起こる？

対策

- 石井さんに相談してみました。その結果、、、
- １日で修正されました!ありがとうございます。

pgpool-II + PostgreSQL ストリーミング レプリケーション構成の特徴

- pgpool-II は高可用な PostgreSQL クラスタを制御する HA クラスタソフトとしても機能する。
  - PostgreSQLサーバのフェイルオーバ/フェイルバック
  - コネクションプール、負荷分散も同時利用可
- pgpool-II がSPOFとなる問題は冗長化で解決
  - 複数APサーバに配置して、pgpool-II を並行動作
    - クラウド基盤等でスケールアウトさせる用途に向く
  - シンプルに運用したい場合は watchdog 機能で pgpool-II を Active/Standby 構成にしてもよい。

Pacemaker RA + PostgreSQL ストリーミングレプリケーション構成との違い

- pgpool-II は PostgreSQL がターゲット
  - PostgreSQLの機能、特徴に合わせた設計
    - ストリーミングレプリケーション構成での HA とクエリ内容に応じた SQL 負荷分散が連動する。
- Pacemaker はいろんなミドルウェアを制御
  - サーバ全体を丸ごと HA 化できる。
  - Pacemaker RA の方が上手な機能もある。
    - 同期/非同期スレーブの自動切換
    - pgpool-II は同期/非同期のステータスを制御しないので、同期スレーブがダウンした場合にマスター停止の恐れ

今後（pgpool-IIに）期待すること

- 複数台の pgpool-II を並行稼動させる場合、運用面で工夫が必要な部分がある。
  - マスターフェイルオーバ時の独自ロック制御
  - watchdog 機能がカバーしてくれないかなぁ。。
- 複数台の pgpool-II を並行稼動させる場合、運用難易度が高いのも課題
  - 同じ設定を全ての pgpool-II サーバに反映する。
  - サーバ数が増えると設定管理に難あり。
  - クラスタ環境の設定を1箇所にまとめられないかなぁ。。

最後に

- pgpool-II と PostgreSQL ストリーミング
  レプリケーションを組合せて、
  高性能、高可用なクラスタを構築しませんか？

- 運用にはいくつか工夫、ノウハウが必要です。
  本番稼働させる際は、十分な動作検証と
  運用テストを「自分の手」で行い、習熟しておく
  ことをお勧めします。