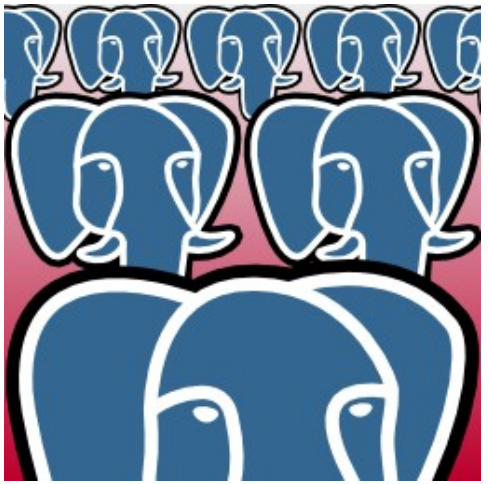


象の群れを飼い慣らす方法

PostgreSQL用クラスタリング技術の最新動向

SRA OSS, Inc. 日本支社
取締役支社長
石井　達夫

SRA OSS,INC.

自己紹介

- OSSの開発とビジネスに携わっています
  - PostgreSQLのコミッタ
  - PostgreSQL用のクラスタソフトpgpool-II の開発
- OSSの普及活動も行っています
  - オープンソースビジネス推進協議会
    - OSS全般の普及促進活動
  - PostgreSQLエンタープライズコンソシアム
    - PostgreSQLの技術的評価や普及促進活動

SRA OSS, Inc.のご紹介

- 1999年よりPostgreSQLサポートを中心にOSSビジネスを開始、2005年に現在の形に至る
- 主なビジネス
  - PostgreSQL, Hinemos, ZabbixなどのOSSのサポート、コンサルティング、導入構築
  - Postgres Plusの販売
  - PowerGresファミリーの開発、販売
  - PostgreSQL用の各種トレーニング

PowerGres

Hinemos

ZABBIX

CERTIFIED PARTNER

EDB ENTERPRISEDB

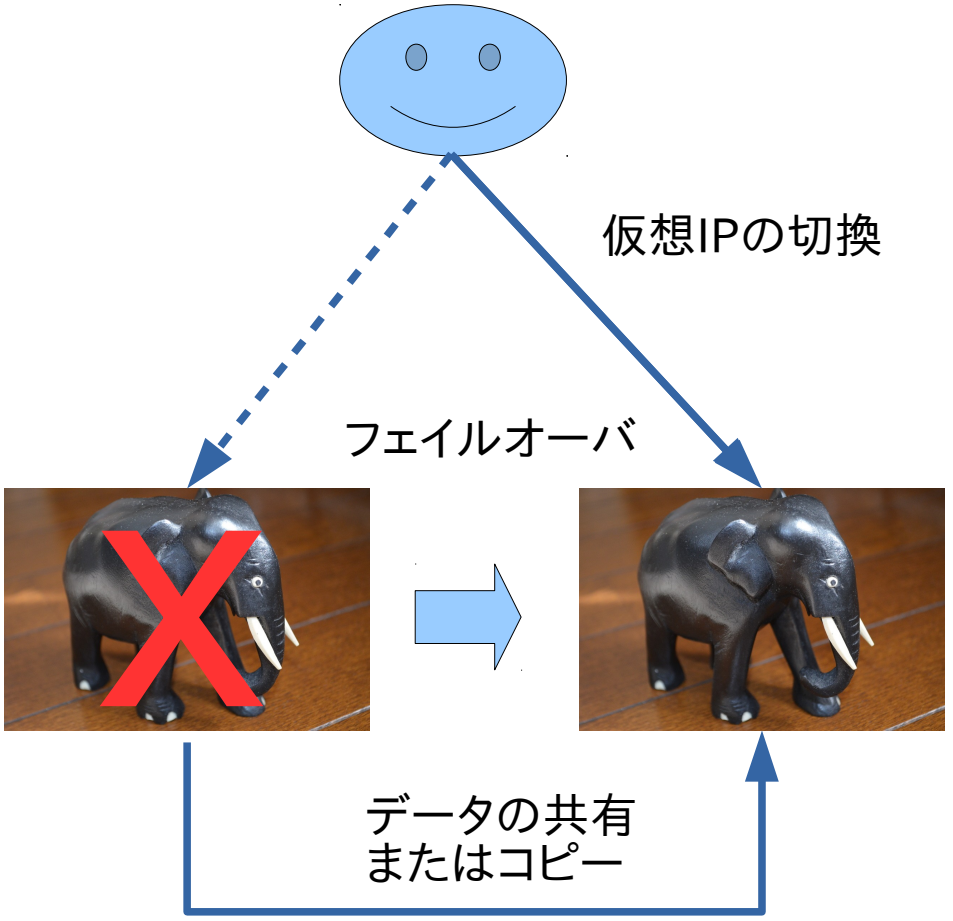
SRA OSS,INC.

PostgreSQLにおける クラスタリングの目的

- 可用性の向上(High Availability)
  - もっとも古典的かつ基本的なリクエスト
    - ダウンタイムの短縮
    - データ損失可能性の低減
- 性能の向上(Scale out)
  - 検索性能(read)の向上
    - ストリーミングレプリケーションによる検索負荷分散
    - postgres_fdwによる分散処理
    - Postgres-XC/XL/X2
  - 更新性能(write)の向上
    - Postgres-XC/XL/X2

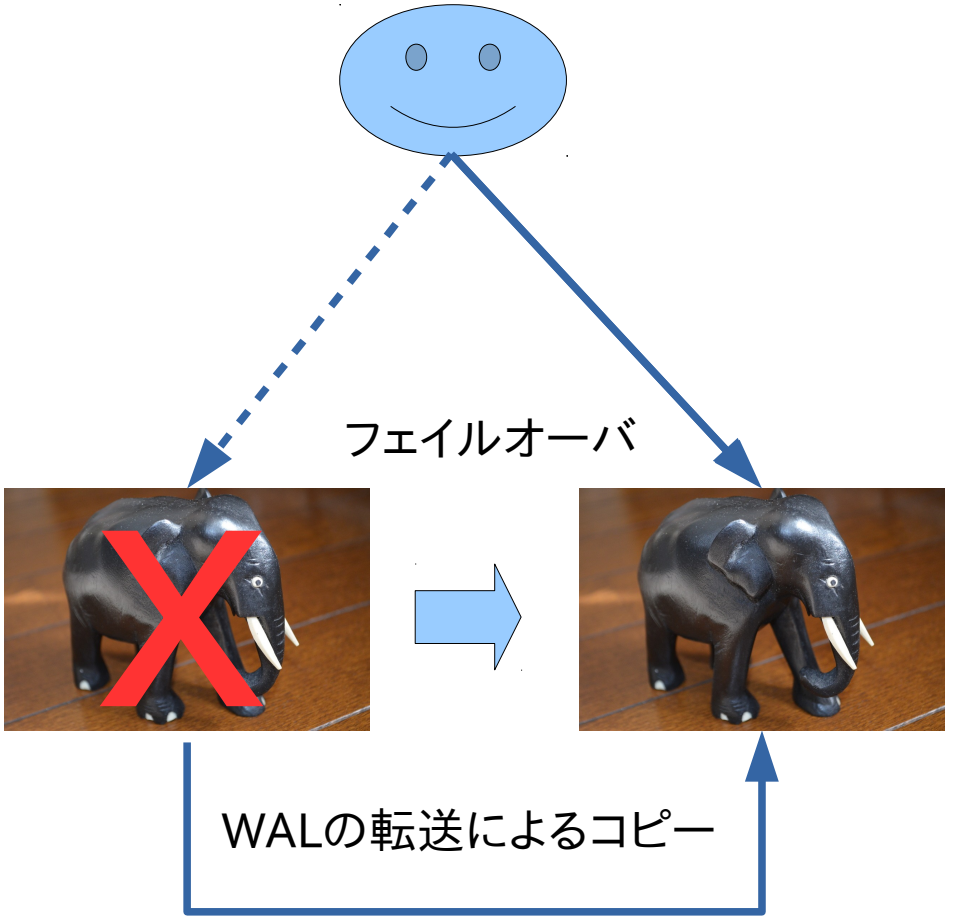
可用性向上のためのクラスタ技術：アクティブ・スタンバイ方式

- Pacemakerなどの汎用HAソフトを使って複数のPostgreSQLを管理
- DBの入ったディスク装置を共有する共有ディスク方式と、共有しない方式がある
- 性能は向上しない
- アプリケーションの修正は必要ない
  - フェイルオーバ中はセッションが切断されるのでその対応は必要



可用性向上のためのクラスタ技術：ストリーミングレプリケーション(1)

- PostgreSQL組み込みのレプリケーション技術である「ストリーミングレプリケーション」を利用
- プライマリからスタンバイへトランザクションログを転送してデータをコピーする
- 書き込み性能は向上しないが、複数のPostgreSQLで読み出し負荷を共有して性能を向上させることも可能
- アプリケーションの修正は必要
  - フェイルオーバ中はセッションが切断されるのでその対応は必要
  - 書き込み処理はプライマリにしか投げてはいけない（そのほか、ある種のロックはプライマリだけに投げるなどの考慮が必要）
- これらの対応が困難な場合には、すべてのDB処理をプライマリにのみ投げる（その場合でもフェイルオーバへの対応は必要）



可用性向上のためのクラスタ技術：ストリーミングレプリケーション(2)

- プライマリが故障した時に、どのスタンバイを昇格させるか決めておく必要がある(スタンバイの数が2以上の場合)
  - 同期レプリケーションを使っている場合は、同期スタンバイを昇格させる
  - 非同期レプリケーションを使っている場合は、固定ルールで昇格するスタンバイを決めておく方式と、動的に決める(最も遅延が少ないものを選ぶ、など)方式がある

プライマリ

スタンバイ1

スタンバイ2

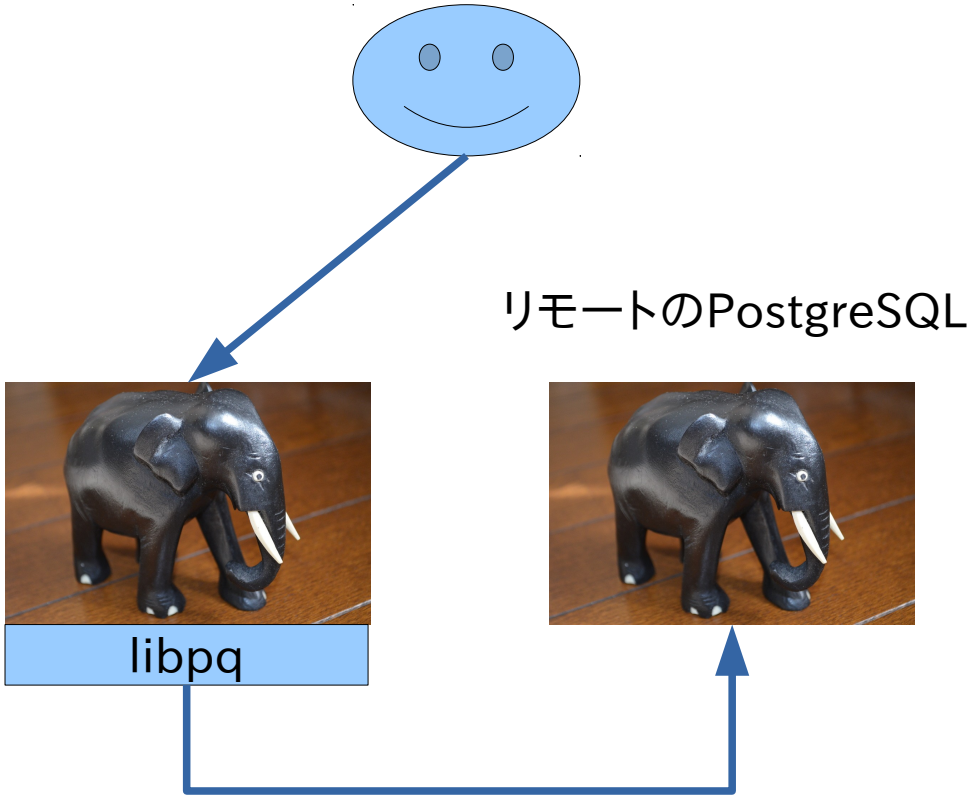
スタンバイ3

性能向上のためのクラスタ技術：ストリーミングレプリケーション(3)

- スタンバイサーバを複数設けて検索性能を向上させる（負荷分散）
  - 一つのSQLが分散処理されるわけではないので、多数のセッションが同時に実行されるような環境で効果が上がる
  - プライマリサーバが過負荷のときに、プライマリに検索処理をさせないようにするのも効果がある
  - どのセッションがどのスタンバイサーバに接続するかを決める必要がある
    - アプリケーションで行う（アプリケーションの改造が必要）
    - ミドルウェアで透過的に実施(pgpool-IIなど)

可用性向上のためのクラスタ技術：postgres_fdw(1)

- あるPostgreSQLから別のPostgreSQLに接続、クエリを実行する拡張機能
- 外部データラッパのフレームワークを使っている
- “CREATE FOREIGN TABLE”文を使って、リモートのPostgreSQLにあるテーブルをあたかもローカルのテーブルであるかのように扱うことができる



性能向上のためのクラスタ技術：postgres_fdw(2)

- 外部テーブルへのSELET, UPDATE, DELETE, INSERが可能
  - 外部テーブルへのアクセスは、常にREPEATABE READ分離レベルのトランザクションの内側で実行されるので、ローカルのトランザクションもREPEATABE READもしくはSERIALIZABLEで実行するのが良い
- 大きなテーブルを複数のサーバに分けて性能向上を狙う、などの用途が考えられる
  - PostgreSQL 9.5からは、継承を使ったパーティショニングに外部テーブルが利用可能になった
  - ただし、外部テーブルへのアクセスが並列に行われるわけではない

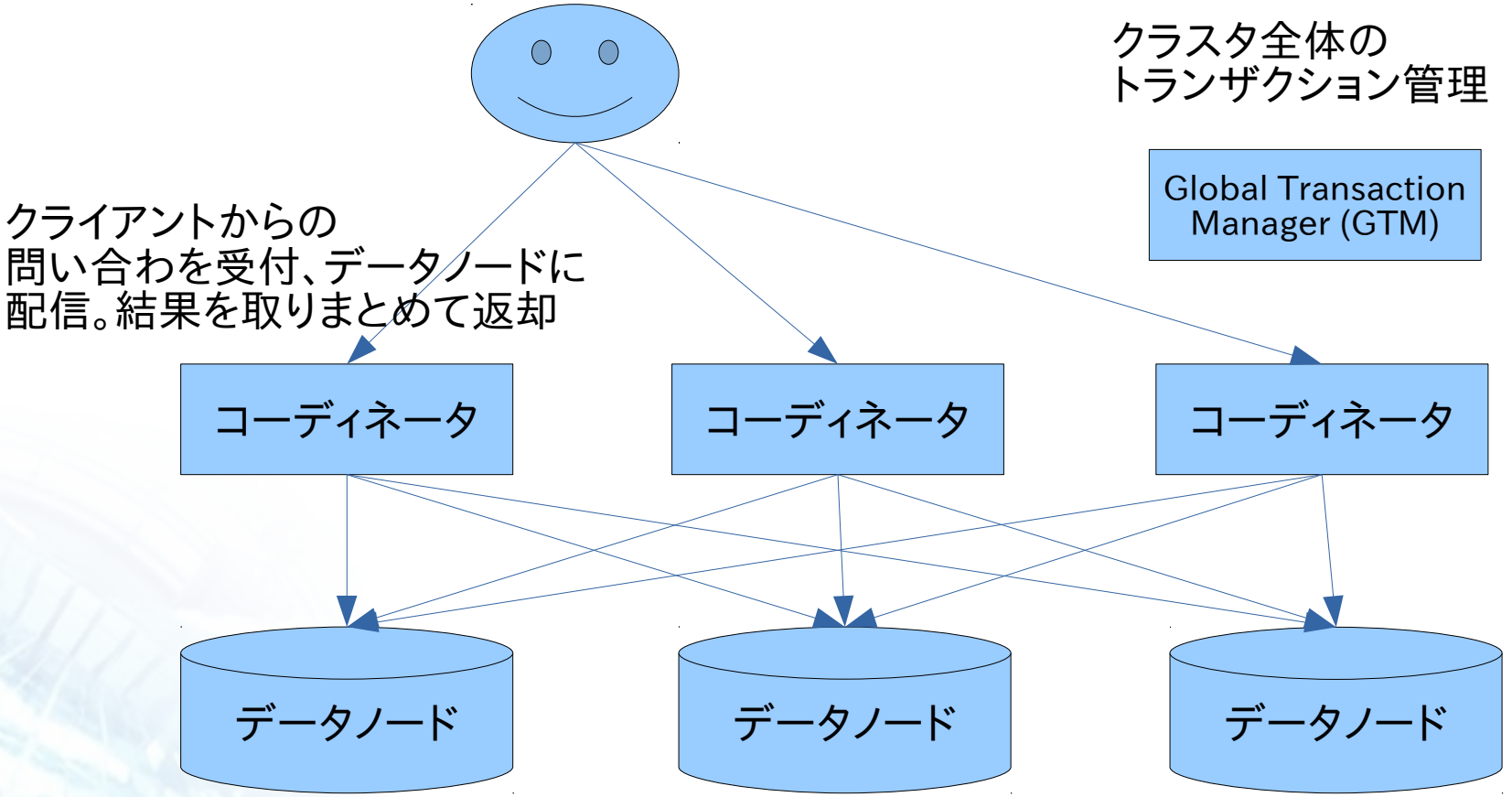
性能向上のためのクラスタ技術：postgres_fdw(3)

- 外部テーブルへのアクセスの最適化は、通常のオプティマイザのへpostgres_fdwが提供するフックが組み込まれることで行われる
  - WHERE/JOIN push down
    - (外部テーブルの結合の際に、)WHERE句を外部サーバに送って実行し、その結果だけを転送することによってデータ量を減らす
  - 処理に必要なカラムだけを転送することによってデータ量を減らす
  - 集約(countやsumなど)は最適化されない
- 外部テーブルの統計情報の取得方法を選択可能
  - 都度EXPLAINを外部テーブルで実施する
  - ANALYZEした結果をローカルに保存しておく

性能向上のためのクラスタ技術：postgres_fdw(4)

- 現状はまだ分散トランザクション機能を提供しているに過ぎず、性能向上のための工夫はユーザに任されている
- 外部テーブルの今後
  - 外部テーブルを使って性能向上のための本格的なクラスタを作ろうと考えている開発者がいる
  - 多くの機能拡張が必要になる
    - 集約関数のリモート実行
    - 並列処理
    - 複数サーバにまたがる読み取り一貫性の実現
    - 並列処理用のオプティマイザ
      - 現状は一部の最適化がpostgres_fdwの中で行われているに過ぎない

性能向上のためのクラスタ技術：Postgres-XC(1)



実際の問い合わせ処理を実行

性能向上のためのクラスタ技術：Postgres-XC(2)

- データノード間で並列に問い合わせが実行可能
- データをデータノードに分散（シャーディング）可能なので、検索性能のみならず、更新性能も向上
- 更新遅延がなく、ノード間の読み取り一貫性も保証されている
- 可用性は向上しない(むしろ低下する)
  - データノードのレプリケーションで対応

性能向上のためのクラスタ技術：Postgres-XC(3)

- PostgreSQLからのフォークであり、PostgreSQLへ追従するのが大変(2016/2現在、PostgreSQL 9.3相当)
- 一部の問い合わせは処理できない
- なかなかPostgres-XCが安定しなかったため、更にフォークを産んだ
  - Postgres-XL
- 最近Postgres-XCはPostgres-X2と名前を変えた
- Postgres-XLなどとの統合の動きもある

各種クラスタ技術のまとめ

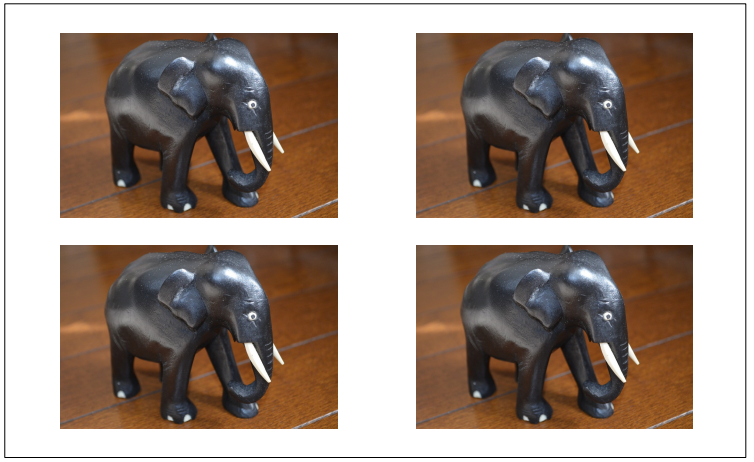
	可用性向上	検索性能向上	更新性能向上	アプリケーション変更度合い	PostgreSQL変更必要	実用性・実績
Pacemaker	○	X	X	○	○	○
Streaming replication	○	○	X	X	○	○
Postgres_FDW	X	▲	▲	○	○	○
Postgres-XC	X	▲	○	X	X	X

○:寄与する
X:寄与しない
▲:制限事項あり／工夫すれば寄与できる

Pgpool-IIのご紹介と使いどころ

PostgreSQLの群れ

- 象はパワフル
- 象が群れになればもっとパワフル!
- でも群れになれば管理が大変なのでは?



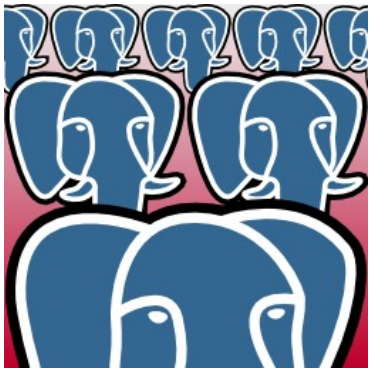
「象の群れ管理問題」の一例

- 群れにはリーダが必要です(プライマリサーバ)
- もしリーダが引退したら、新しいリーダを立てなければならない(フェイルオーバ、昇格)
- その際、他の象は新しいリーダに追従しなければならない
- リーダ以外の象は、働けなくなったら引退する(スタンバイのフェイルオーバ)
- 新しい象が群れに加わるときはスムーズに行われなければならない|
- 群れの象はお互いに助けあわなければならない (負荷分散)
- リーダにしかできない仕事がある (更新クエリ)

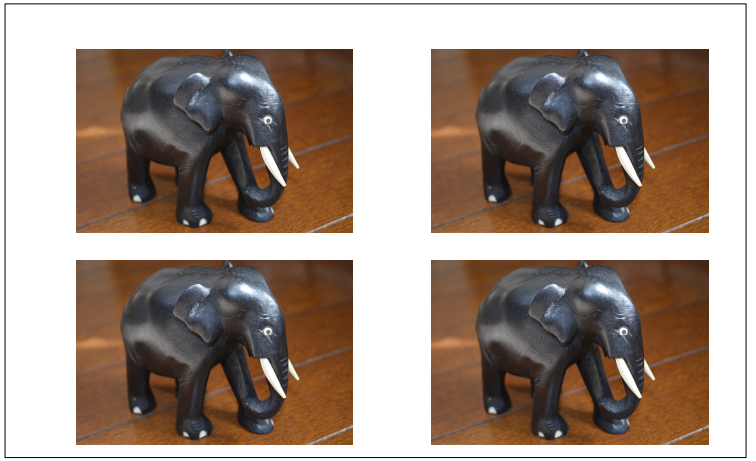
pgpool-IIで「象の群れ管理問題」を解決

- pgpool-IIを使うことにより、象の群れは単独の象のように見える
- ユーザ定義のフェイルオーバスクリプトにより、フェイルオーバ時にどのスタンバイが昇格するかのポリシーを決められる
- 「フォローマスターコマンド」で新しいプライマリへの自動追従も可能
- スタンバイがダウンしたら、自動的に群れからそのスタンバイは外されるので、クラスタとしての運用を継続できる
- クエリが検索クエリなら、負荷分散の対象となる
- クエリが更新クエリなら、プライマリサーバに送るI

pgpool-II



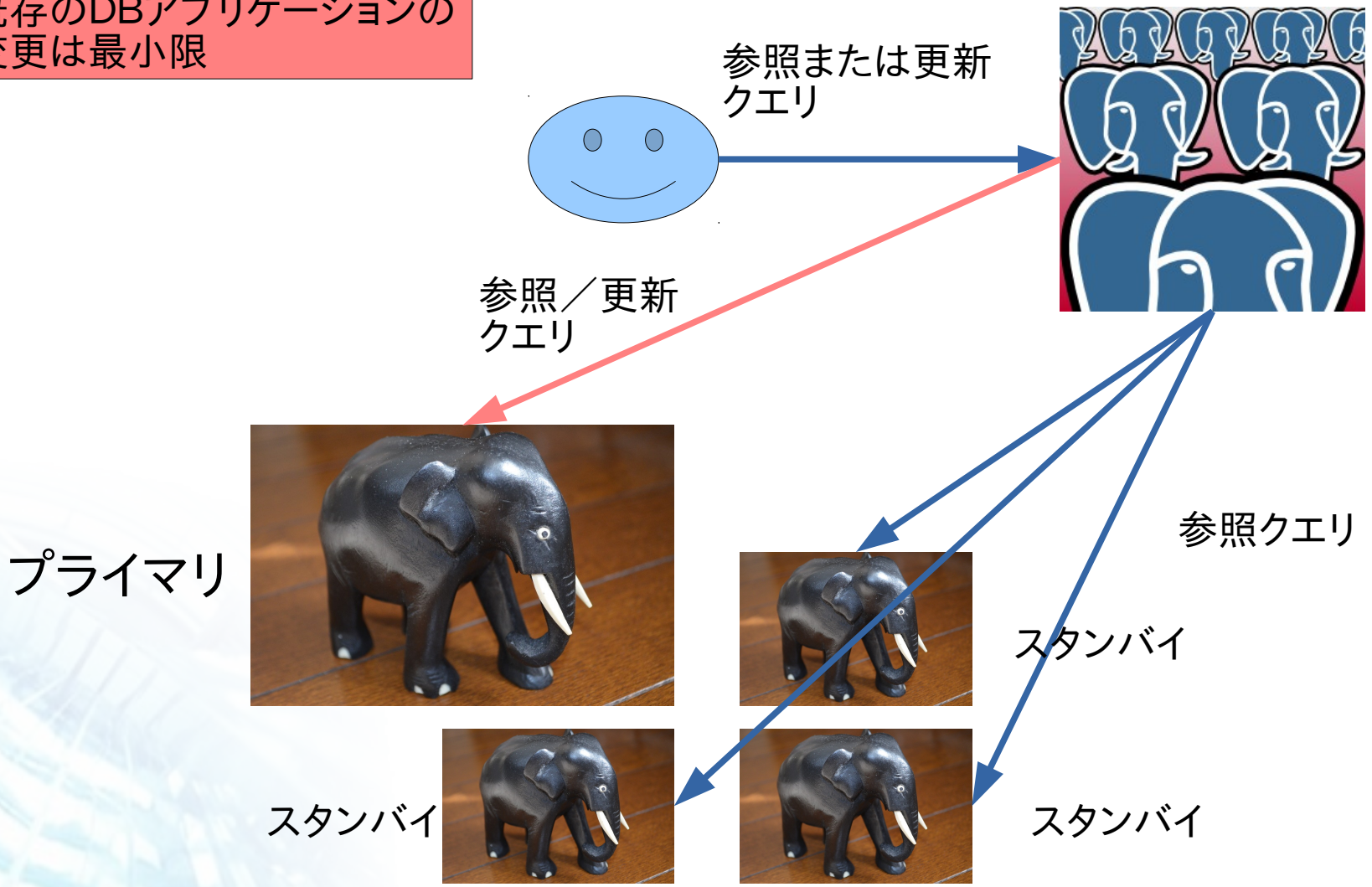
象の群れは単独の象のように見える



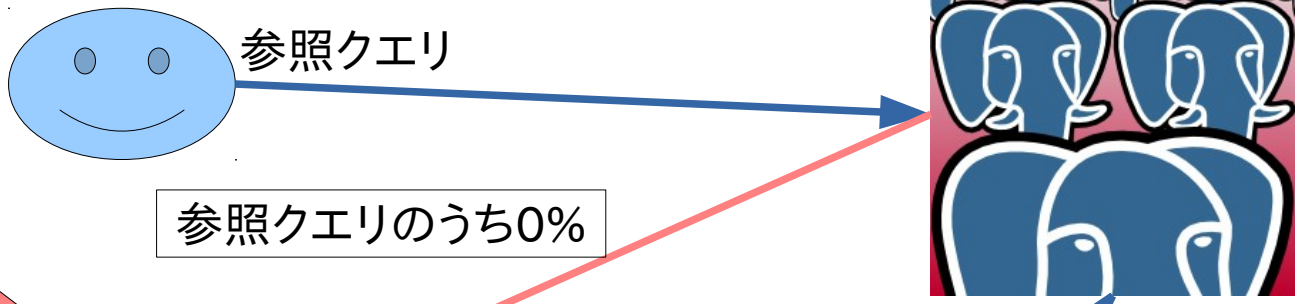
SRA OSS, INC.

クエリのディスパッチ/ルーティング

既存のDBアプリケーションの
変更は最小限



負荷分散



参照クエリ

プライマリは
更新処理に
専念する

参照クエリのうち0%

参照クエリのうち
40 %

参照クエリのうち
30 %

アプリケーション名や
データベース名で分散
させることも可能

プライマリ

スタンバイ

参照クエリのうち
30 %

スタンバイ

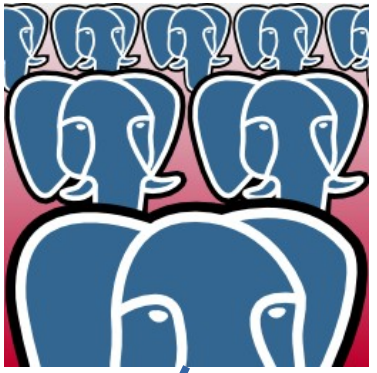
スタンバイ

SRA OSS, INC.

スタンバイサーバがダウンした時

スタンバイサーバがダウンしたら、
pgpool-IIがそのことを検知し、
クラスタリングの対象から取り除く

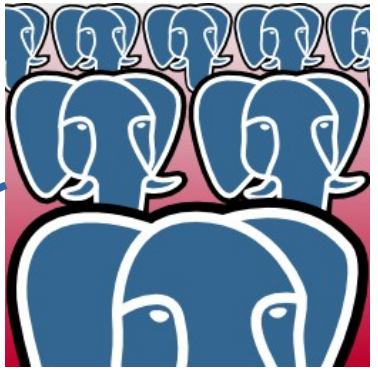
既存のセッションは再接続が
必要



プライマリ

スタンバイ

プライマリサーバがダウンした時



プライマリ

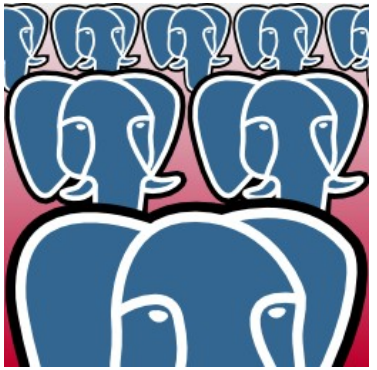
スタンバイ

プライマリがダウンすると、スタンバイの一つが昇格し、新しいプライマリになる

他のスタンバイは新しいスタンバイに追従するようになる

SRA OSS, INC.

新しいスタンバイの追加



新しい象！

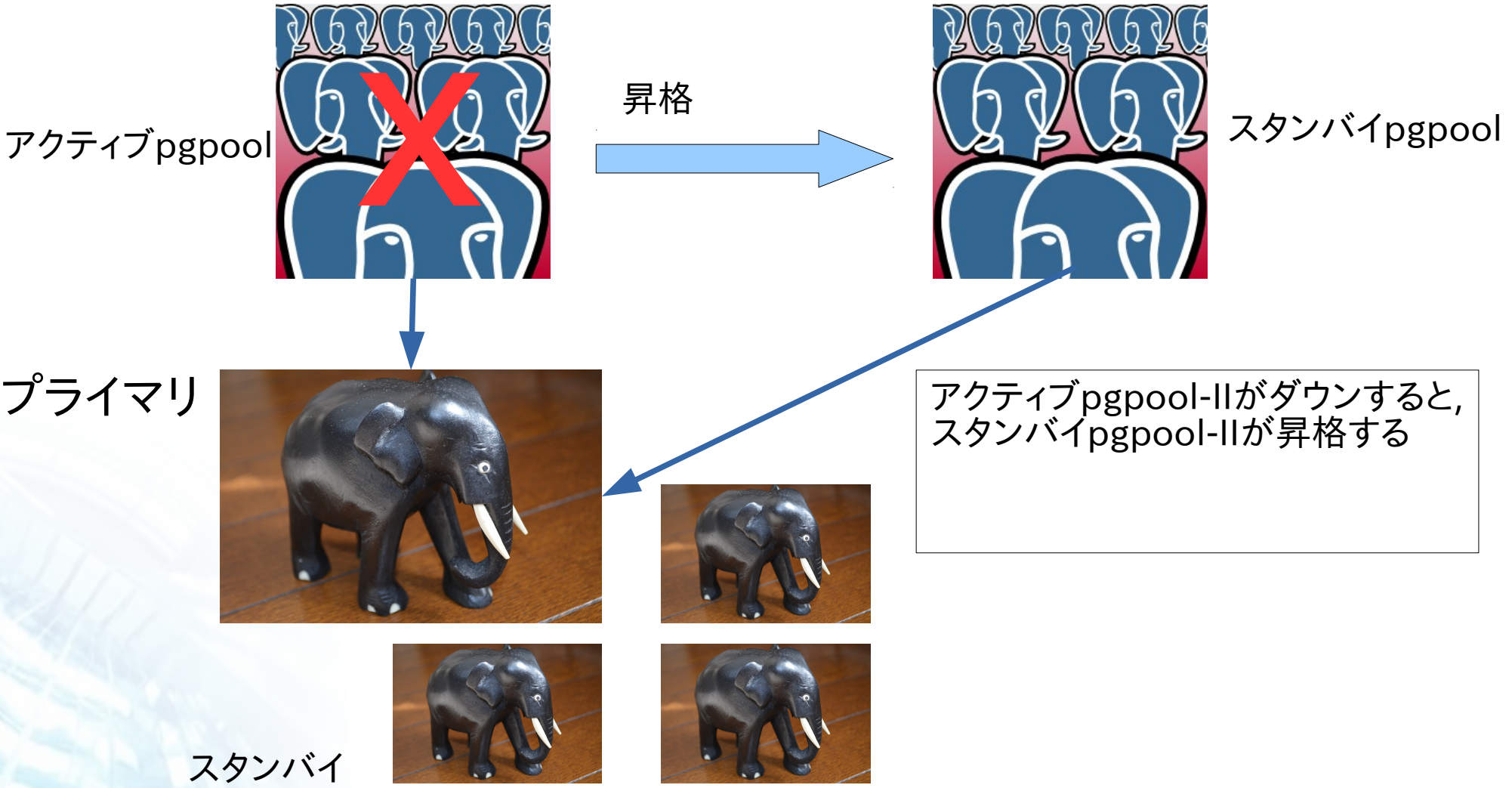
プライマリ

スタンバイ

新しいサーバは簡単に追加できる。
pgpool-IIは新しいサーバにプライマリからデータをコピーし、他のサーバに影響を与えずにクラスタに新しいサーバを追加できる。
既存のセッションは切断されない。

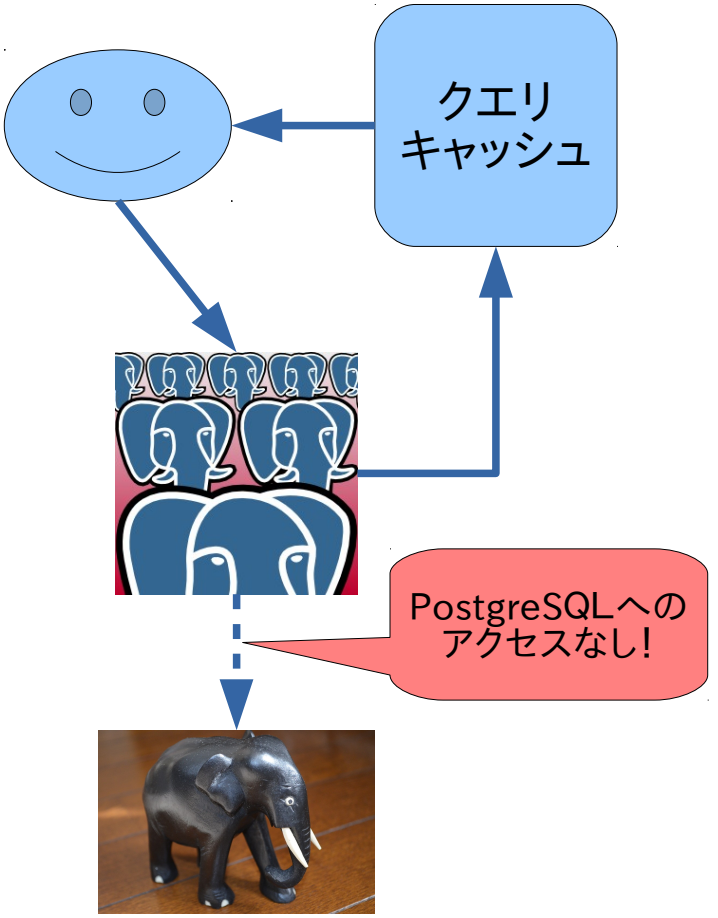
SRA OSS, INC.

Watchdog: pgpool-II組み込みの高可用性機能



インメモリクエリキャッシュ

- Pgpool-IIはクエリキャッシュを使ってクエリの結果を再利用する
- クエリキャッシュはメモリ上に置かれるので非常に高速
- その際にPostgreSQLアクセスは一切なし
- キャッシュ用のストレージは、共有メモリ火memcachedから選べる
- テーブルが更新されると、そのテーブルを参照したクエリキャッシュはすべて廃棄される
- タイムアウトベースのキャッシュ更新も可能



コミュニティサポートポリシー

- PostgreSQLと同様メジャーリリースとマイナーリリースがある
  - 3.x.y – x:vメジャーバージョン, y: マイナーバージョン
    - “3.4.2”: “3.4” がメジャーバージョンで, “2”がマイナーバージョン
  - 年に一度のメジャーバージョン
  - 年に3-4回のマイナーリリース
  - マイナーリリース間では互換性が保たれる
  - メジャーバージョン間では互換性は保証されない
- 最初のリリースから5年間バックパッチ(保守)を行う
  - つまり、常に5-6個のバージョンを保守している
- 5年以上のサポートが必要ならば、弊社にご相談ください

最新バージョン：pgpool-II 3.5 が1月にリリースされました！

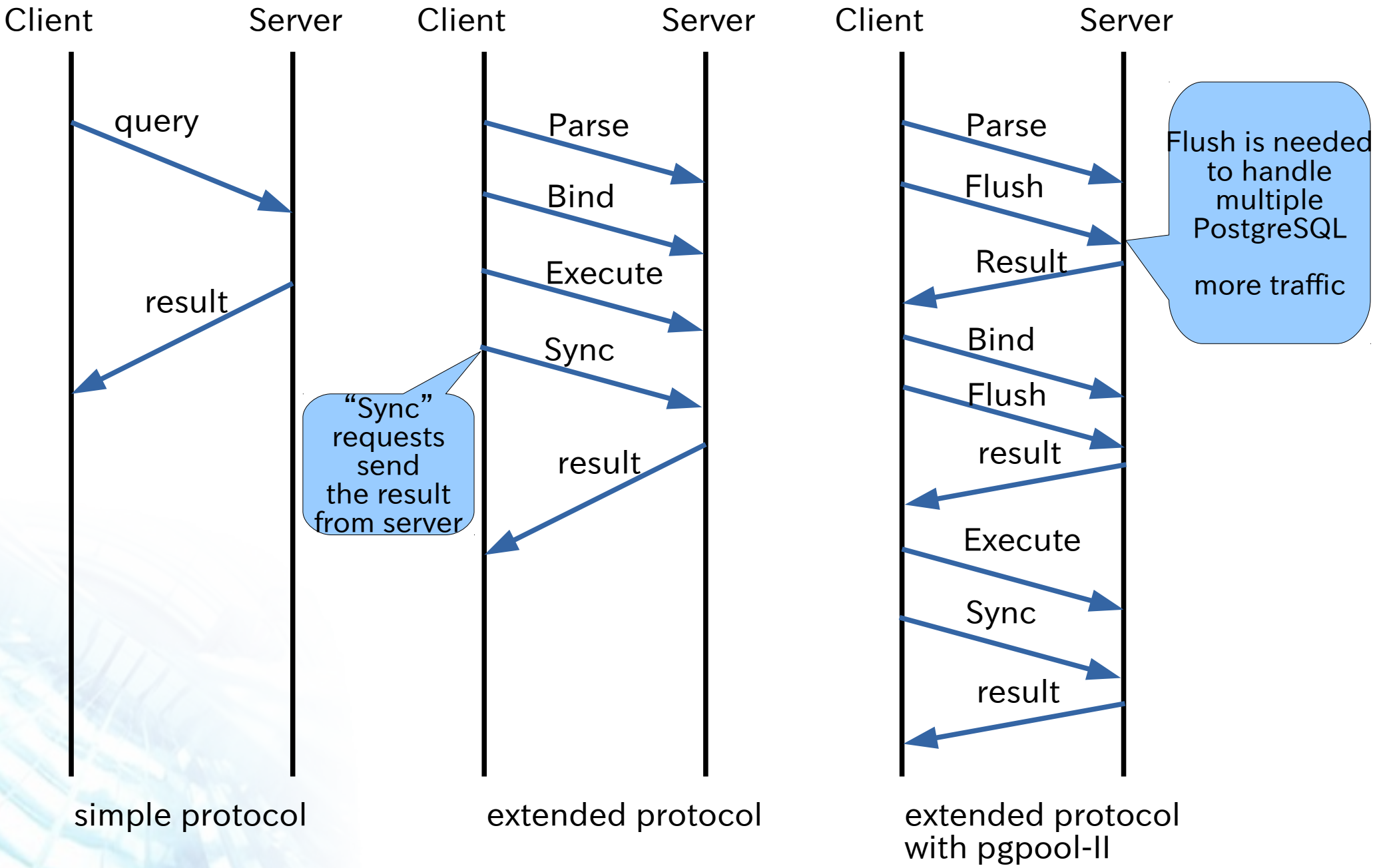
- 性能改善
- watchdogの改善
- PostgreSQL 9.5対応
- pcpコマンドの改善
- など

性能改善

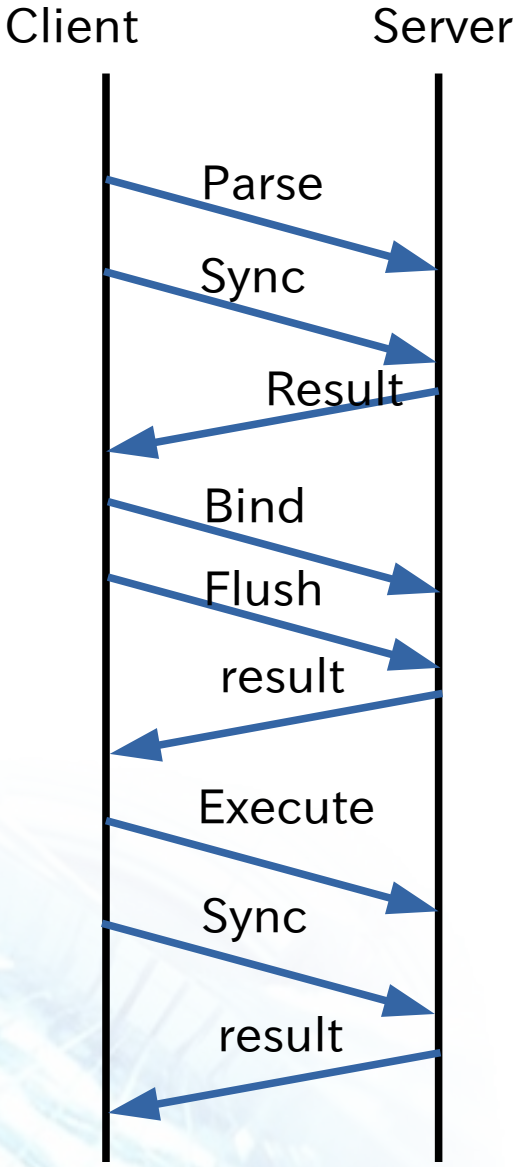
拡張プロトコルでの性能改善

- 従来pgpool-IIで拡張プロトコル(Javaなどが利用)が非常に遅かった(大体拡張プロトコルでない場合の半分)
- 問題は実装にあり: ネットワーク転送のオーバヘッドが大きかった
- まずは「拡張プロトコル」とは何かから説明します

Some details are omitted



SRA OSS, INC.

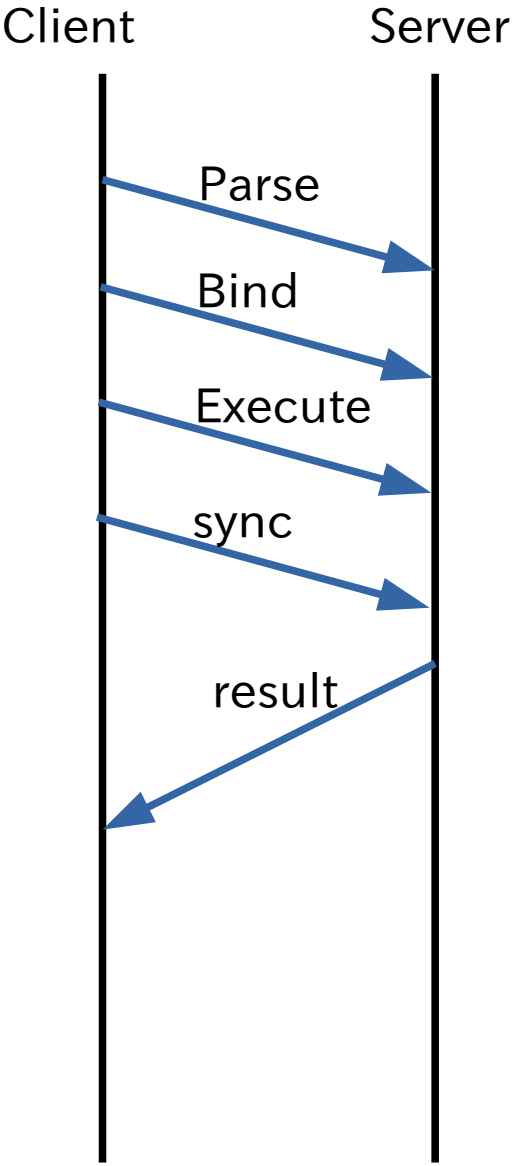


extended protocol
with pgpool-II

Too many Flush messages

In streaming replication we could omit some of Flush messages

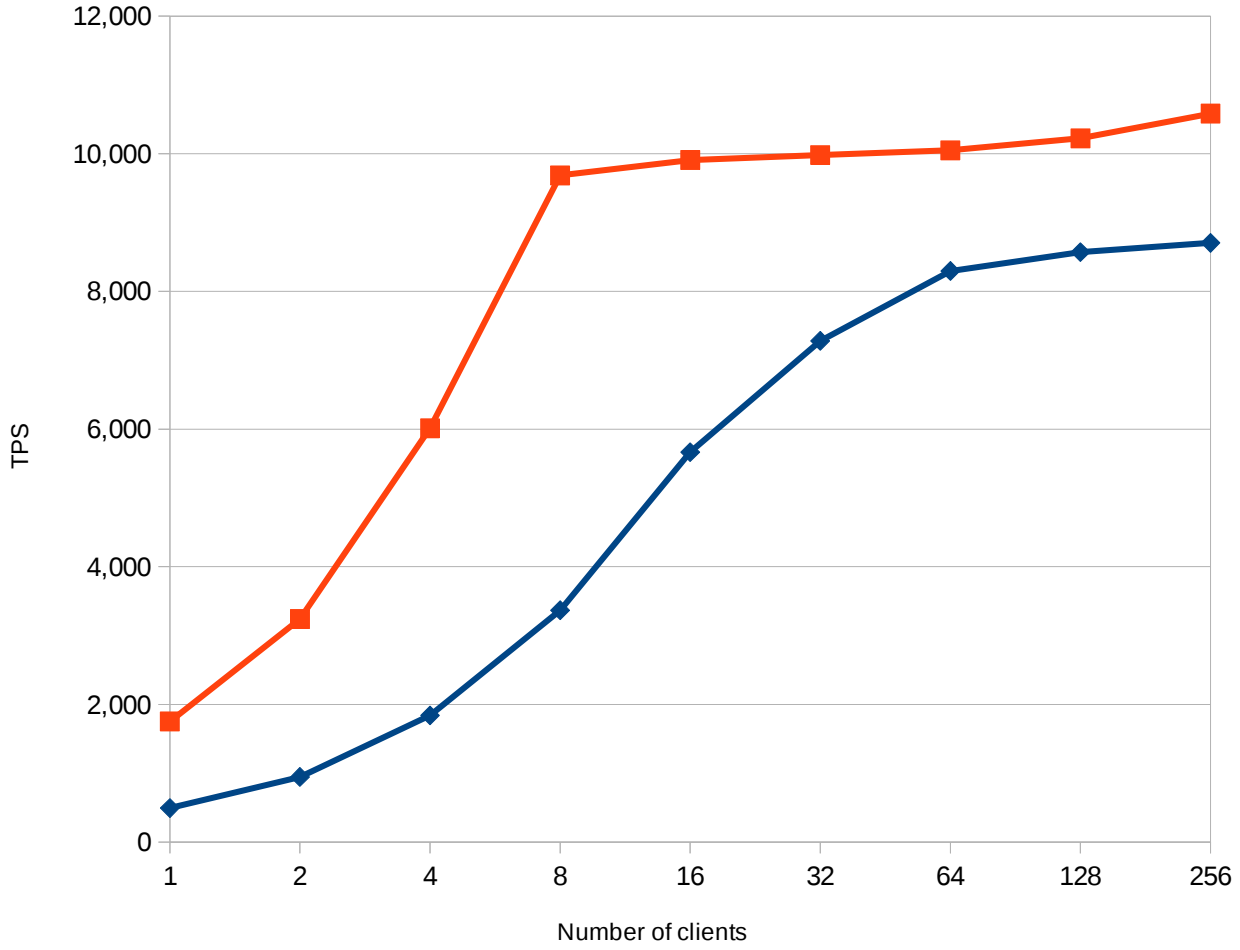
enhanced!



extended protocol
with pgpool-II in 3.5

SRA OSS, INC.

拡張プロトコルでのベンチマーク結果

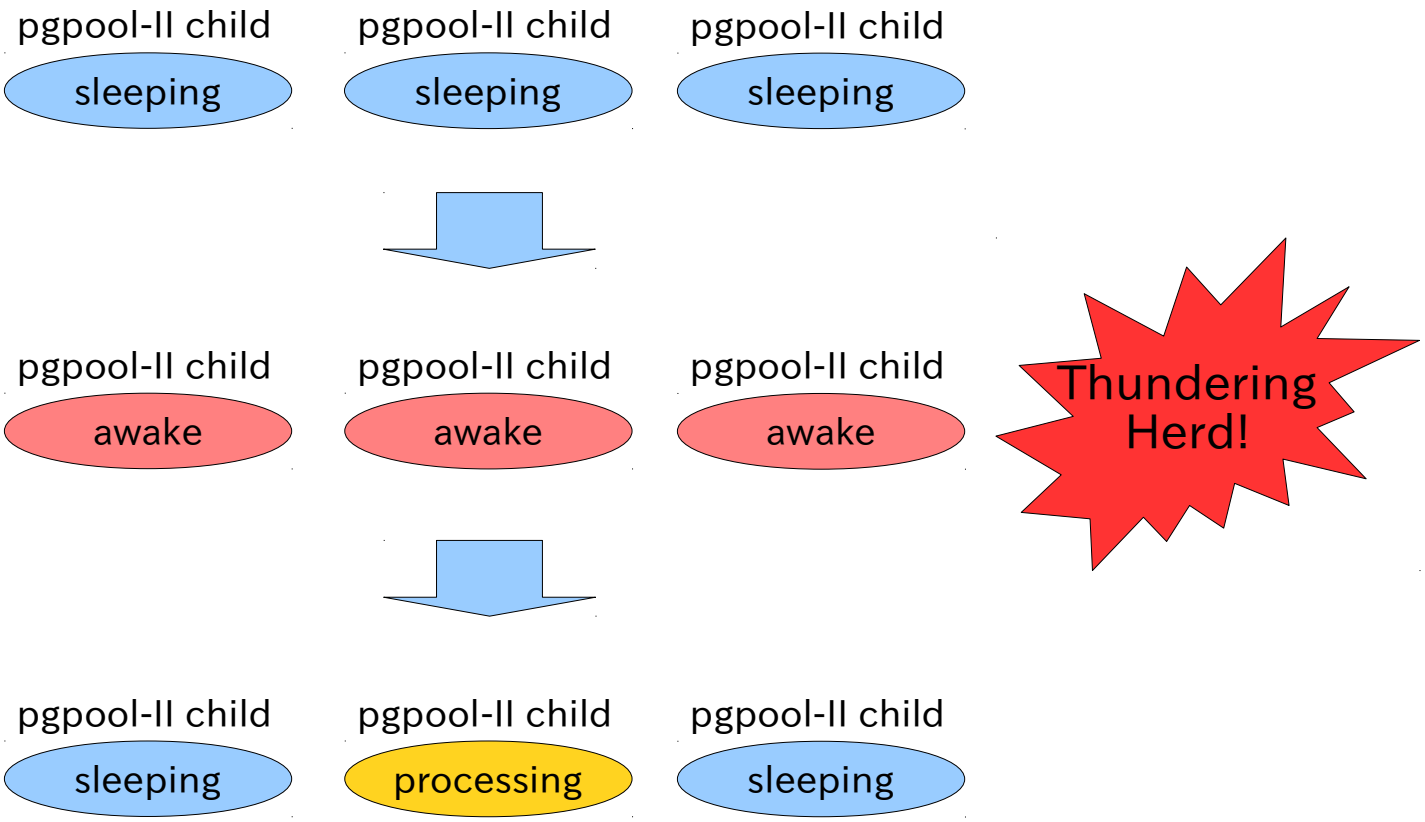


pgpool-II 3.5は
20% から250%
pgpool-II 3.4
より高速!

AWS m4.large instance
CentOS 6
PostgreSQL 9.4 x2
(streaming replication)
pgbench -S

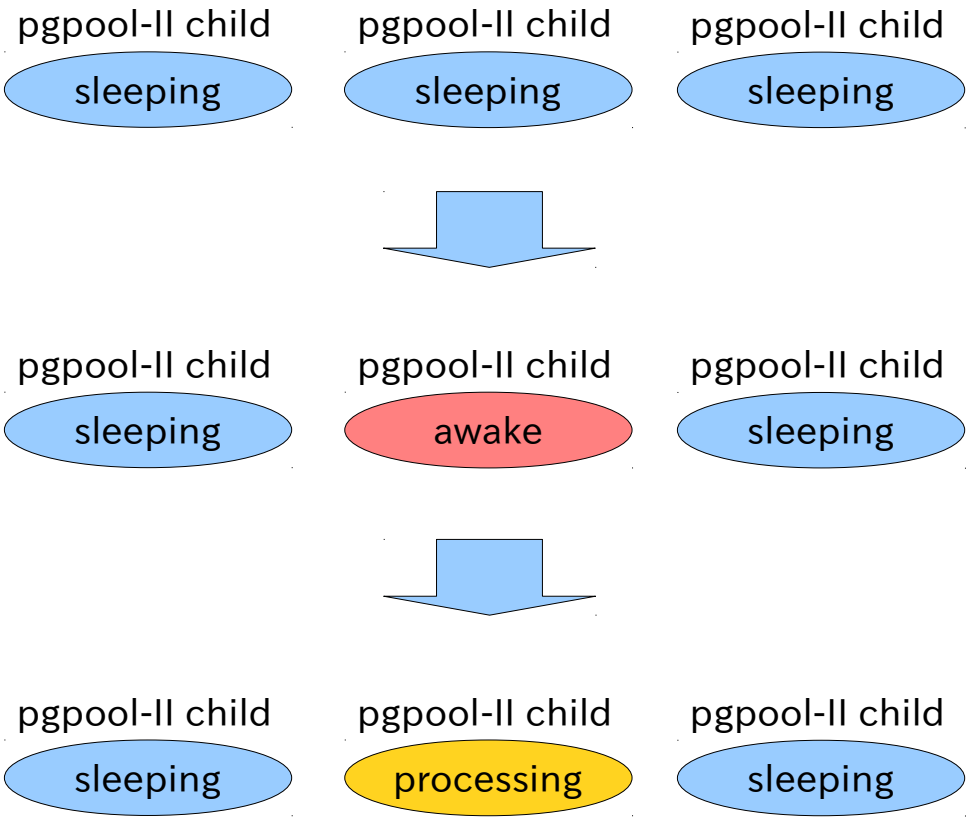
「Thundering herd」問題への対応 (1)

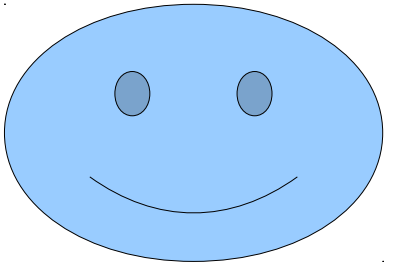
pgpool-II 3.4



「Thundering herd」問題への対応 (2)

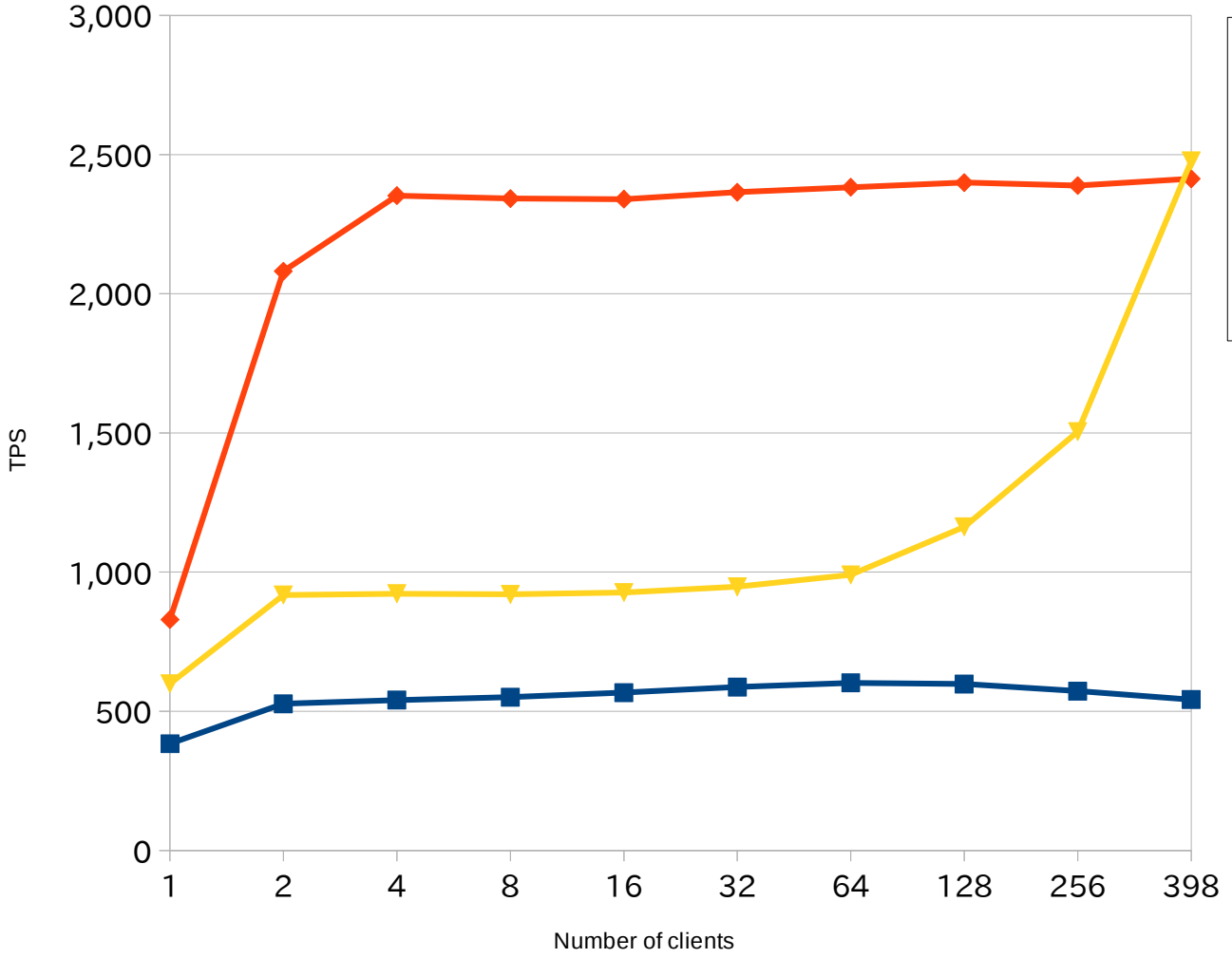
pgpool-II 3.5





No thundering
Herd problem

「Thundering herd」問題への対応 (3)



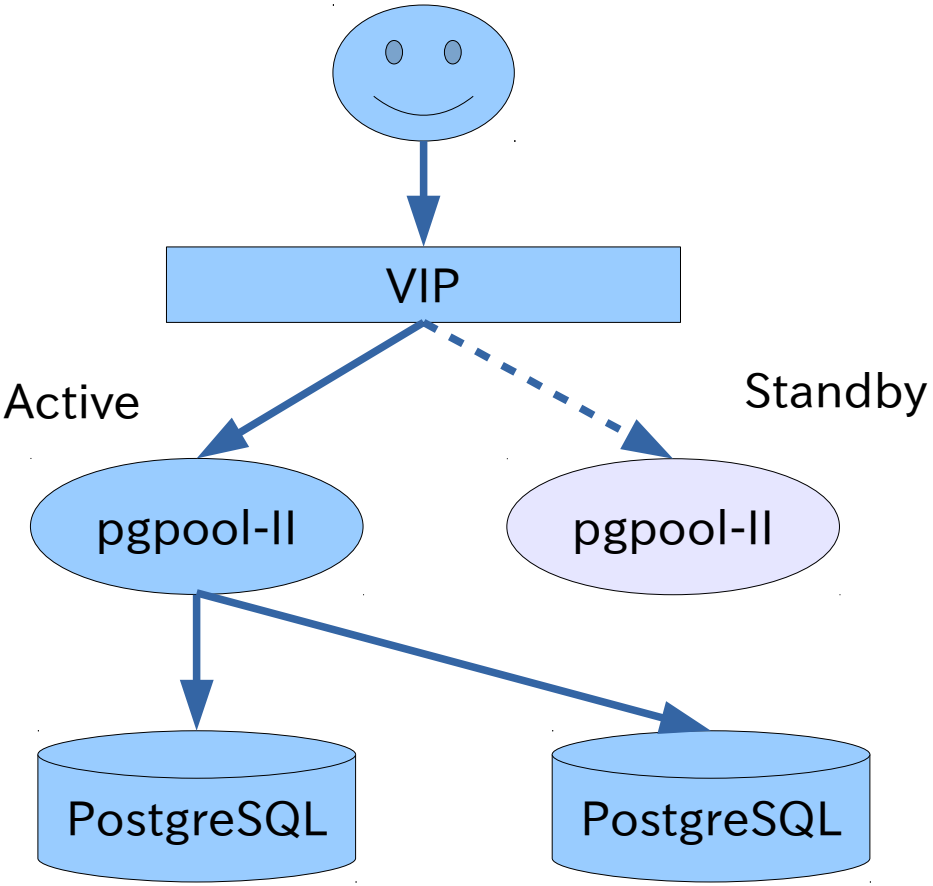
同時接続数がpgpoolの子プロセスよりも少ない場合
pgpool-II 3.5は
40%から150%
pgpool-II 3.4よりも高速

Note PC with 16GB Mem,
CORE i7 x2, 512GB SSD
Ubuntu 14.04
PostgreSQL 9.4 x2
(streaming replication)
pgbench -S -C -T 300

watchdogの改善

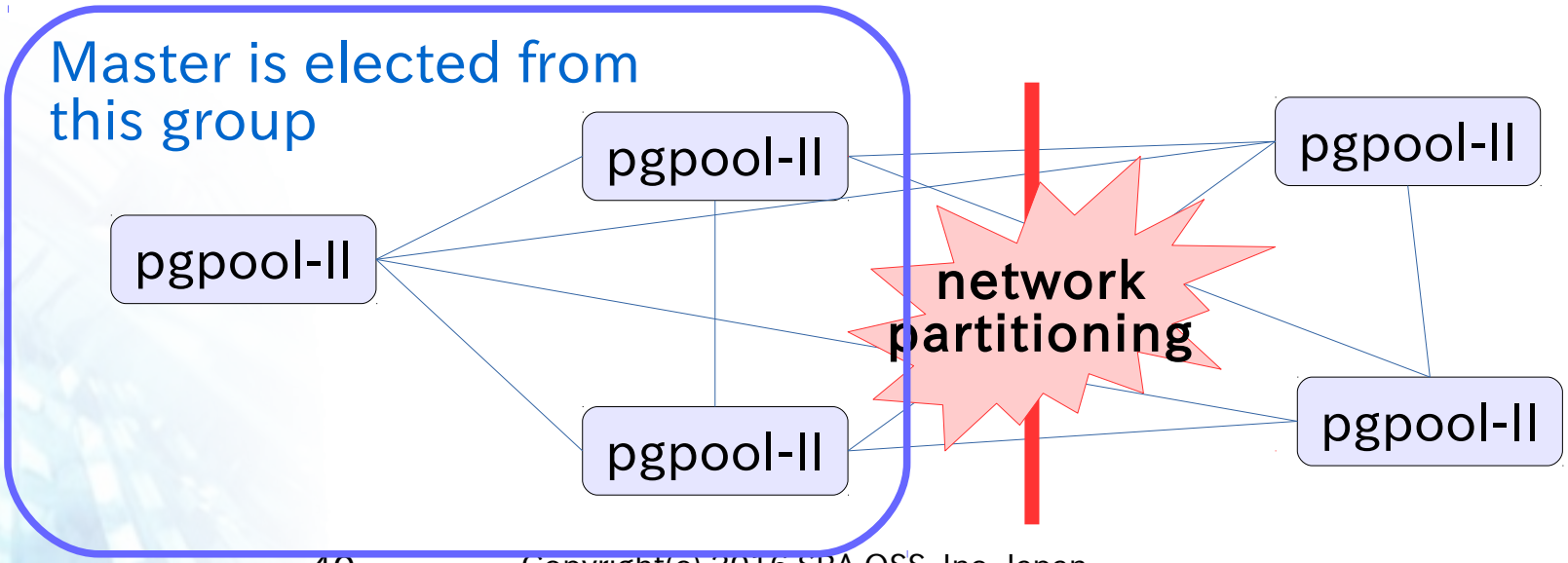
“Watchdog”とは?

- pgpool-IIはproxyとして動くので, pgpool-II 自身が “Single point of failure” (SPOF)になり得る
- “Watchdog”はpgpool-IIの組み込みHA機能
- 2つ以上のpgpool-IIインスタンスがお互いに監視し合い、アクティブpgpool-IIがダウンしたら、スタンバイpgpool-IIが取って代わる
- pgpool-IIへのアクセスは仮想IP経由でActive pgpool-IIへアクセス。常に同じIPが使える



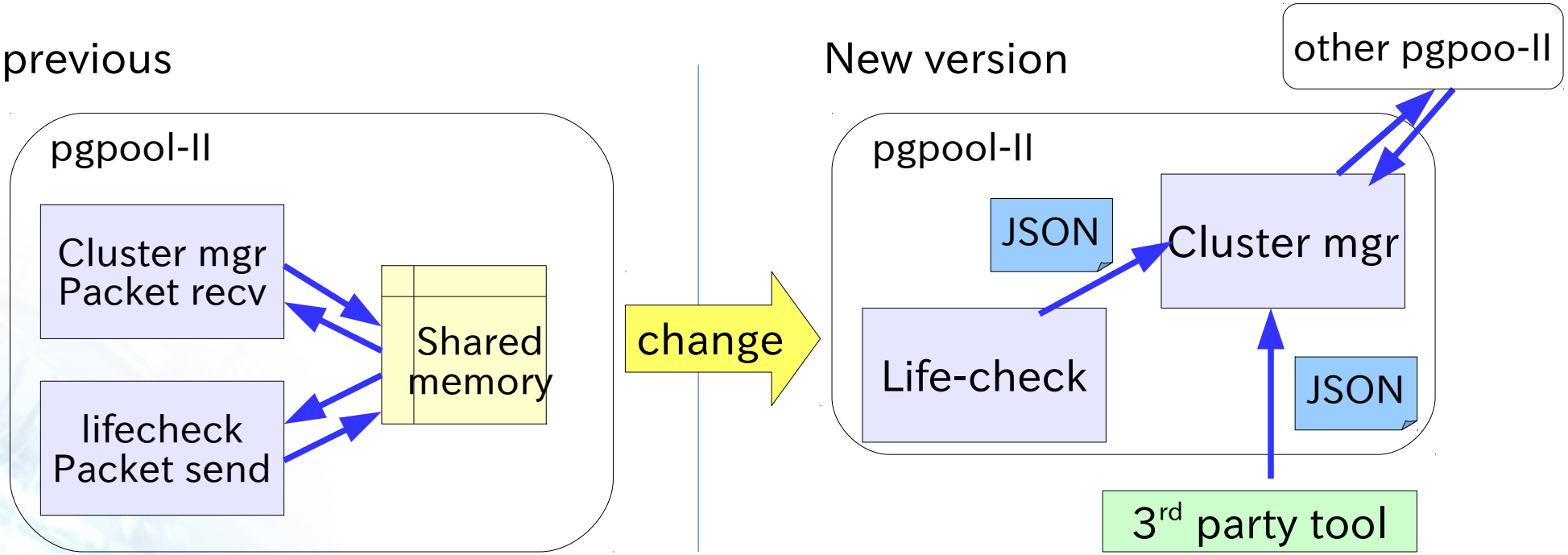
スプリットブレイン問題への対応改良

- スプリットブレイン問題とは
  - ネットワークが分離した時に、昇格すべきpgpool-IIが決まらない
  - Quorum（定足数）の利用
    - 自分のネットワークに所属するpgpool-IIの数が定足数に達していれば、その中からマスタpgpool-IIを選ぶ
    - pgpool-IIインスタンスの数は、3以上かつ奇数でなければならない



プロセス間通信の改良

- UNIX domain socketとJSONフォーマットの利用
- サードパーティがLife checkを独自に実装することも可能



PostgreSQL 9.5対応

Importing PostgreSQL 9.5 parser

- pgpool-IIのパーサがPostgreSQL 9.5から移植された(以前は9.4)
- 負荷分散とクエリキャッシュが以下の構文をサポート
  - GROUPING SET, CUBE, ROLLUP,
  - TABLESAMPLE
- ネィティブレプリケーションモードが新しい構文をサポート
  - INSERT … ON CONFLICT
  - UPDATE tab SET (col1,col2,...) = (SELECT ...), ...

pgpool-II 3.5その他の改良

ヘルスチェックとレプリケーション遅延のチェック対象データベースが指定可能に

- Herokuなど、“postgres” or “template1” データベースへの接続を認めないシステムへの対応
- ヘルスチェック
  - DBの死活監視
- レプリケーション遅延チェック
  - ストリーミングレプリケーションにおいて、遅延が一定以上になるとそのスタンバイサーバにクエリを送らない
- 新しい設定項目
  - health_check_database
  - sr_check_database

PCPコマンドの改良

- “PCP”コマンドとは
  - “Pgpool Control Protocol”
  - 一群のpgpool-II管理コマンド
    - PostgreSQLサーバのpgpool-IIへのアタッチ、デタッチ
    - on-lineリカバリ
    - pgpool-IIから様々な情報の取得
- 多くの問題点に対応
  - 単一セッションでしか使えない
  - 引数指定が不便
  - パスワードをコマンド引数で渡す必要がある

SELECT発行数の表示

- “show pool_nodes”で各DBに発行されたSELECTの数を表示

New!

```
test=# show pool_nodes;
 node_id | hostname | port  | status | lb_weight |  role   | select_cnt
---------+----------+-------+--------+-----------+---------+------------
 0       | /tmp     | 11002 | 2      | 0.500000  | primary | 338230
 1       | /tmp     | 11003 | 2      | 0.500000  | standby | 163939
(2 rows)
```

中国語ドキュメント

- Pgpool-II ドキュメント本体
- pgpoolAdminメッセージカタログ

注意!

- “パラレルモード”は削除されました
  - 制限事項が多すぎ
  - ユーザが少ない
  - コードが複雑

今後の計画

- Pgpool-II 3.6を今年の終わり頃にリリース予定
  - 安定性の向上、使い勝手の向上を目指す
  - 現在何をやるかを議論中
    - テストケースの追加
    - SETコマンドの実装
    - ドキュメントの改善(SGML?)
    - フェイルオーバ時のセッション切断を最小限に

各種クラスタ技術のまとめ

	可用性向上	検索性能向上	更新性能向上	アプリケーション変更度合い	PostgreSQL変更必要	実用性・実績
Pacemaker	○	X	X	○	○	○
Streaming replication	○	○	X	X	○	○
Postgres_FDW	X	▲	▲	○	○	○
Postgres-XC	X	▲	○	X	X	X
pgpool-II	○	○	X	○	○	○

○:寄与する
X:寄与しない
▲:制限事項あり／工夫すれば寄与できる

URLなど

- pgpool-II公式サイト
  - http://www.pgpool.net
- Postgres-X2
  - https://github.com/postgres-x2
- SRA OSS
  - http://www.sraoss.co.jp

Thank you!