pgpool-IIの過去、現在、そして未来

pgpool-II Global Development Group
石井 達夫

2003年6月27日: pgpoolの誕生

- コネクションプーリング、フェイルオーバのみ
- サポートするPostgreSQLサーバは2台まで
- Version 2プロトコルのみサポート(まだVersion 3プロトコル=PostgreSQL 7.4はリリースされていなかった)
  - C言語で4,719ステップの規模

ナウマン象(古代の象)

2003年6月27日(金) 22:54:46 JST
[pgsql-jp: 30256] PostgreSQL用コネクションプールサーバ pgpool

石井です.

PHPをはじめ,Perlなど,言語を問わず使える「pgpool」というPostgreSQL用のコネクションプールサーバを作ったので公開します.できたなのでまだアルファ版程度のクォリティですが,よろしかったらお試し下さい.

ftp://ftp.sra.co.jp/pub/cmd/postgres/pgpool/pgpool-0.1.tar.gz

もちろんpgpoolはオープンソースで,ライセンスはPostgreSQLのBSDライセンスと同様のものにしています.

pgpoolを作った動機は,PHPでコネクションプールが使えないことに不満を持ったからです.

一応PHPには「パーシスタントコネクション」というものがあってDBへの接続への接続をキャッシュできますが,少なくともapacheのプロセスの数だけコネクションができるので,DBへ過大な負荷がかかりがちです.
pgpoolを使うとコネクションをキャッシュできるだけでなく,DBへの接続数を適切な数に制限できるので,DBの性能を引き出すことができます.

2004年4月:pgpool 1.0の誕生

- 現在の「ネィティブ・レプリケーションモード」に相当する機能を実装した(まだPostgreSQLにはレプリケーション機能がなかった)
- クエリキャンセル対応
- ラージオブジェクトのレプリケーション対応
- C言語で5,890行
- この頃は、マイナーリリース(x.x)の際にも平気で機能を追加していたりして、かなりいい加減なリリース管理がされていた

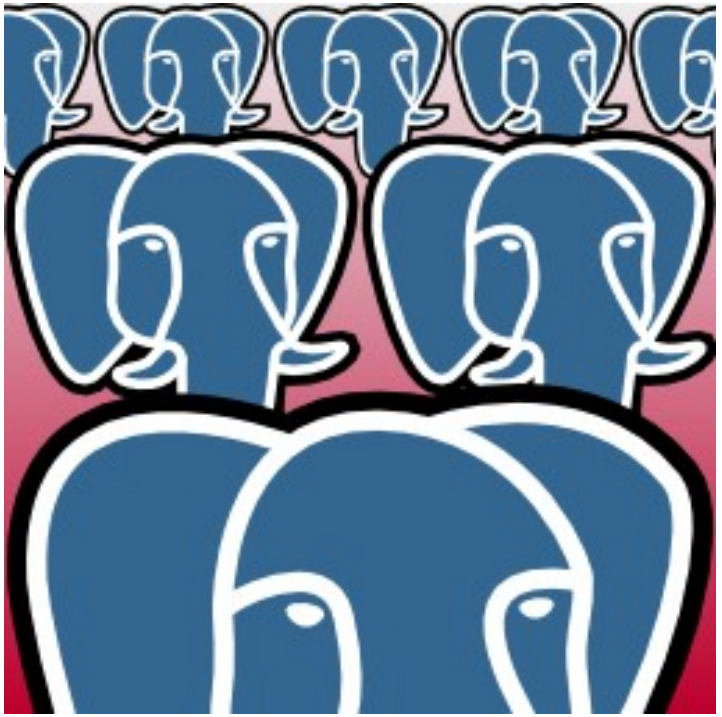
現代の象になったがまだまだよちよち歩き

pgpool 2.0へ進化

- 2004年6月リリース
- 1.0のわずか2ヶ月後にリリース
  - かなり頑張って開発していたようだ
- V3プロトコルにネィティブ対応
- C言語で7,750行
- この後2.5を2005年2月にリリース。ヘルスチェックや、マスタースレーブモードへの対応を追加
  - これでpgpoolとしてのリリースは完了

2006年9月:pgpool-II 1.0の誕生

- 開発手法の変更
  - 個人プロジェクトから、チーム作業へ
  - IPAの援助で開発
- 機能の大幅追加、現在の姿にほぼ近づく
  - サーバ台数の制限撤廃
  - SQLパーサを搭載して精密な構文解析
  - 管理コマンド(pcp)の実装
  - GUI管理ツール(pgpoolAdmin)の実装
  - パラレルクエリモードの実装
  - C言語で73,511行と、一気に10倍近い規模に増えた(bison, flexコード行数を含む)



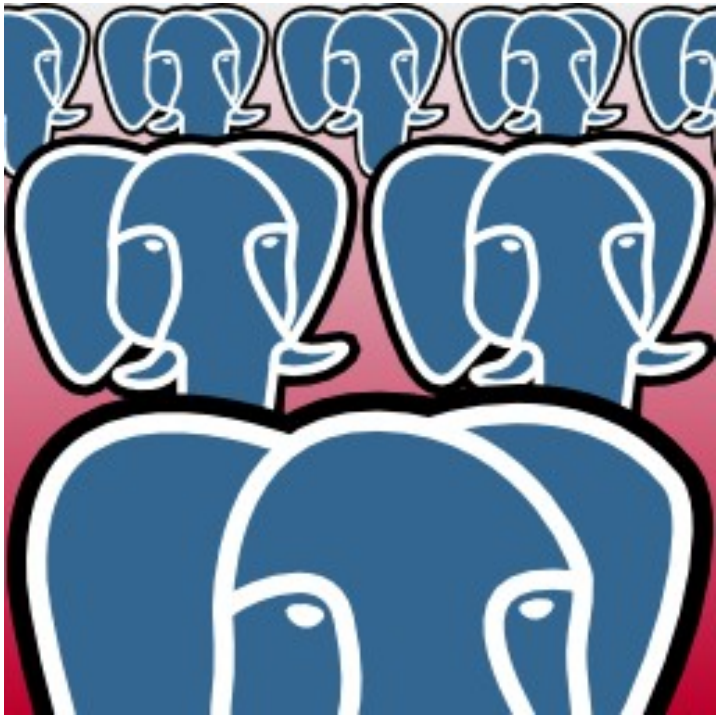
2011年11月pgpool.netへの引っ越し

- それまでホスティングさせてもらっていた pgfoundry の不安定さに手を焼く
- 新しいホストティングサイト pgpool.net を作ることを決意
- pgpool.netをオープン、ソースコード管理も CVS から git に移行した
  - gitへの移行にあたって、フランスのコミュニティのご支援をいただきました

引っ越しはなかなか大変でした

- pgpool.netでは、英語の情報と日本語の情報を同時発信することにした(難しい場合は英語を優先)

現在の開発体制



- 石井
  - 全体のとりまとめ。コードも書きます
- Ahsan Hadi
  - ユーザニーズの取り込み、ベンチマーク
- Muhammad Usama
  - 3.4の開発からコミッタに就任
- 長田悠吾
  - watchdogを中心に担当。リリース作業やRPM作成も
- 安齋希美
  - pgpoolAdminとインストーラを中心に担当。リリース作業やRPM作成も

pgpool-IIの現在

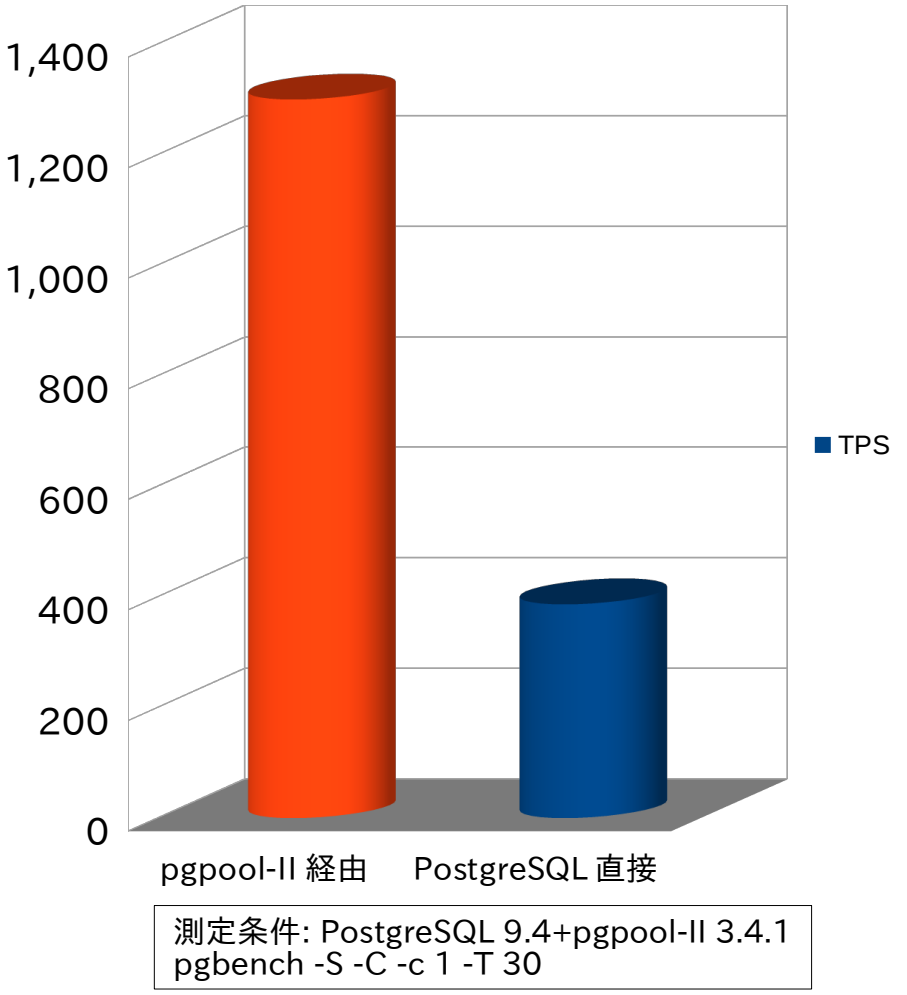
- PostgreSQLのクラスタの総合管理ツールに進化
  - ストリーミングレプリケーションの管理ツールとして
  - クエリをプライマリとスタンバイに振り分ける
  - スタンバイに対するread onlyクエリの負荷分散
  - フェイルオーバの管理
  - クエリキャッシュ
- pgpool-II自体のHA化
  - watchdog

pgpool-IIの現在の主な機能

性能向上	コネクションプーリング 検索負荷分散 クエリキャッシュ
高可用性	自動フェイルオーバ フェイルオーバスクリプト フォローマスタスクリプト watchdog
クラスタ管理	オンラインリカバリ
クラスタとアプリケーションの親和性	クエリの自動振り分け

性能向上(1)

- コネクションプーリング
  - PostgreSQLは接続に時間がかかる
  - 接続しっぱなしにして接続時間を節約
  - コネクションプーリングの有効時間を設定することも可能
  - Javaなどの環境では自前のコネクションプーリングを持っていることがあり、その場合は効果はない

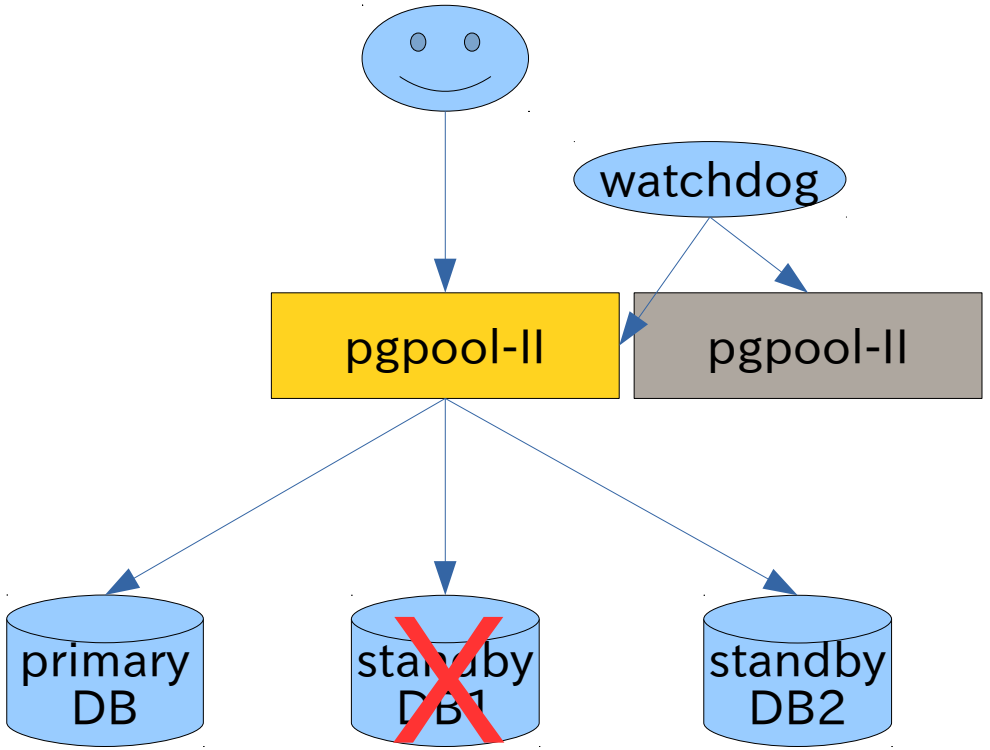


性能向上(2)

- 検索負荷分散
  - 複数のPostgreSQLに検索クエリを分散させて全体として性能向上
  - 重いクエリほど効果的
  - 最大でPostgreSQLの数に比例した性能向上が期待できる
- クエリキャッシュ
  - 検索結果をキャッシュしておき、2回目以降はキャッシュを返すことにより、高速化。PostgreSQLにアクセスしないのでDB負荷も軽減できる
  - 更新が頻繁に行われるシステムでは効果がない(キャッシュヒット率70%以上を推奨)

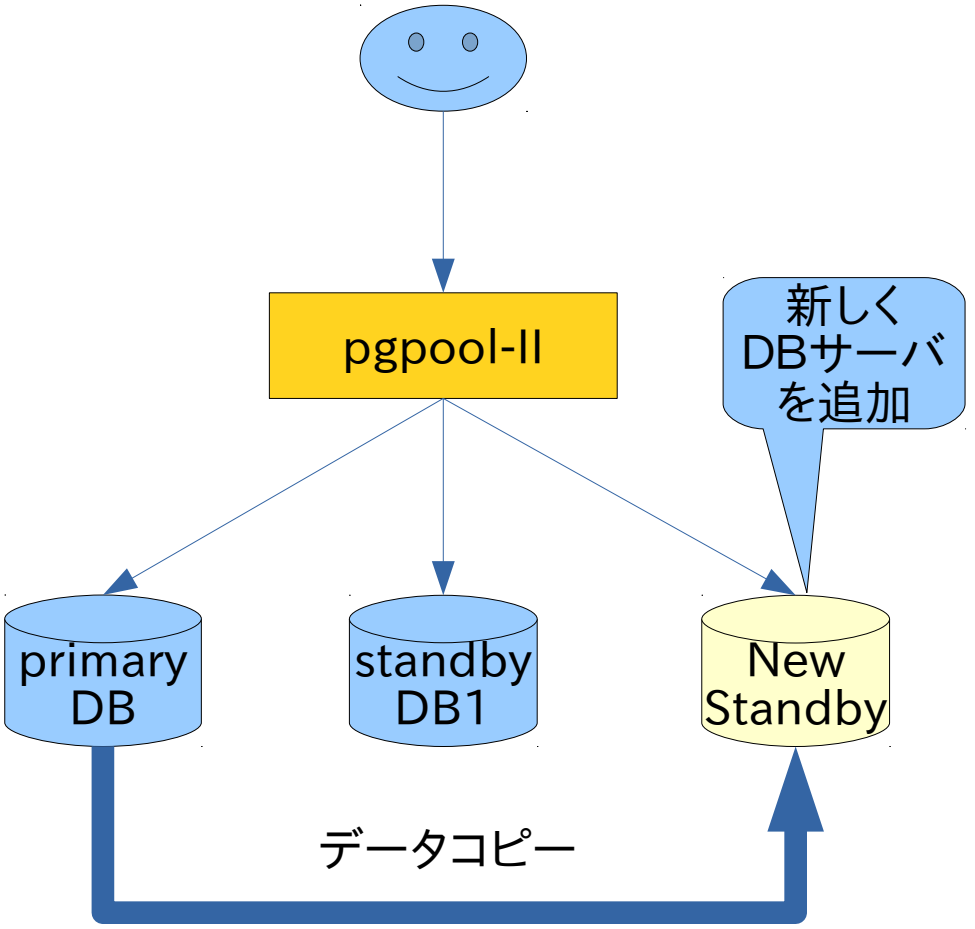
高可用性

- 自動フェイルオーバ
  - 複数のPostgreSQLを用意しておき、1台のPostgreSQLがダウンしたら自動的に切り離して残りのPostgreSQLで運用を継続する
  - フェイルオーバ時の挙動はユーザスクリプトで定義できる
- watchdog
  - pgpool-II自体を二重化する
  - pgpool-HAよりも細かいコントロールが可能、追加ソフトが不要



クラスタ管理

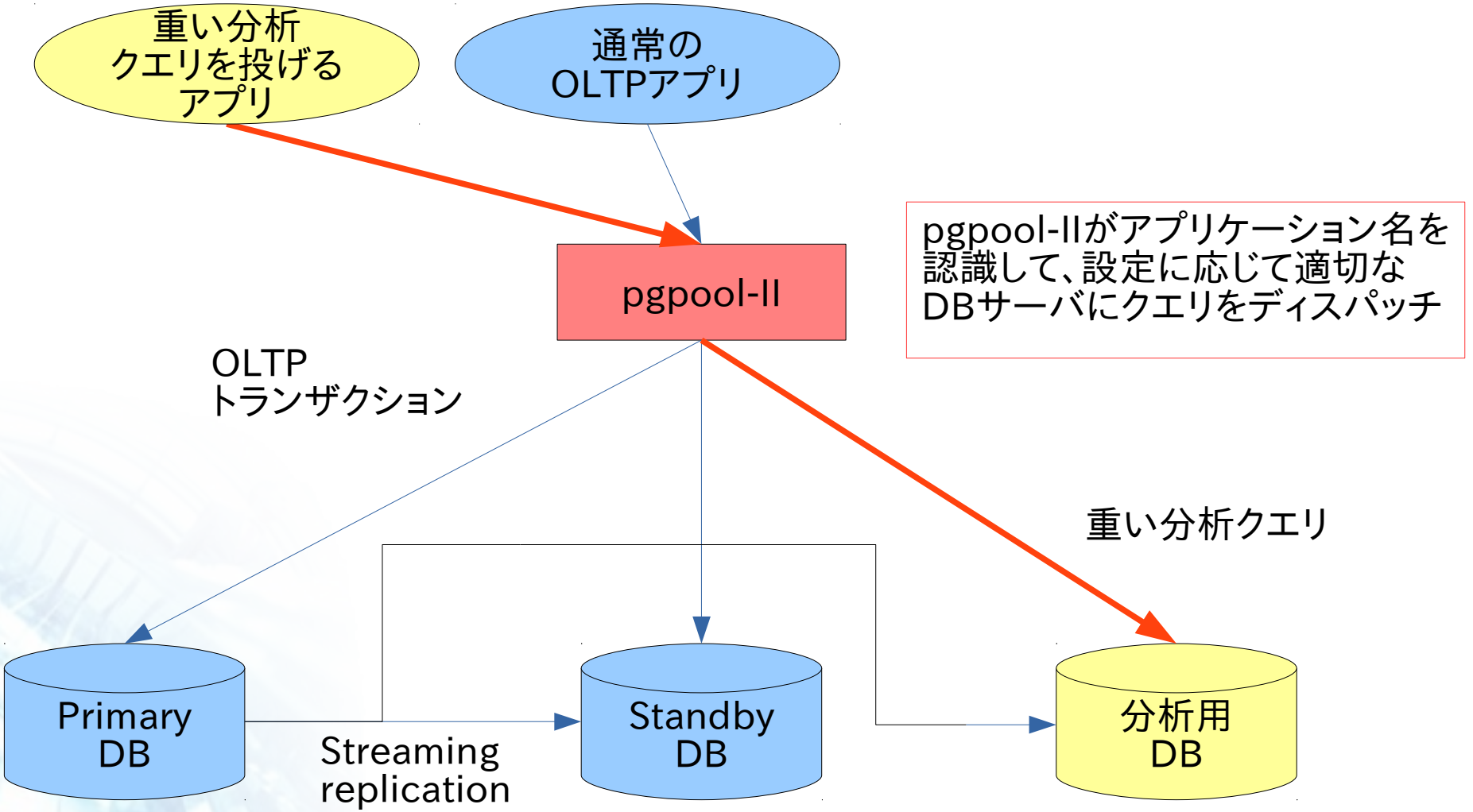
- DBクラスタの運用を止めることなく新しいスタンバイサーバを追加できる(オンラインリカバリ)
  - スクリプトを定義しておくと、コマンド一発でプライマリサーバからスタンバイサーバにデータコピー、立ち上げまで行える
  - 新しいスタンバイサーバを追加する際にも利用できる
  - 新しいサーバは、新しいセッションから随時利用できる



クラスタとアプリケーションの親和性

- ストリーミングレプリケーションでは、スタンバイサーバに投げることのできるクエリに制限がある。以下のクエリは投げてはいけない
  - 更新クエリ
  - DBの追加削除
  - ユーザの追加削除
  - VACUU/REINDEX
  - LOCK文の一部
  - 一時テーブルを使うクエリ
  - シリアライザブル隔離レベルの使用
  - などなど...
- こうした考慮をアプリケーションで行うのは煩わしい
- pgpool-IIが自動的にプライマリとスタンバイにクエリをディスパッチ

OLTPトランザクションの邪魔をせずに重い分析系のクエリを実行



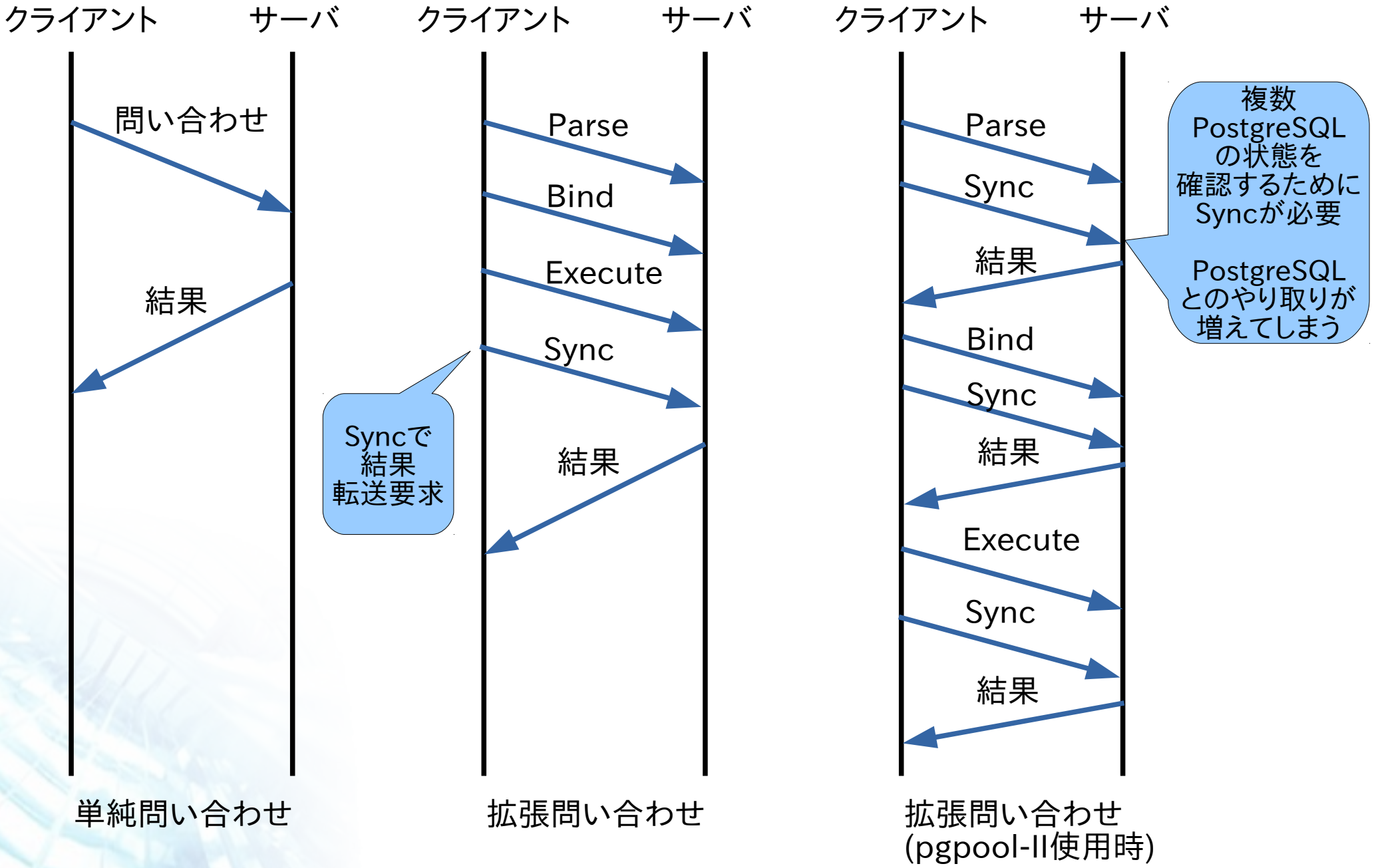
pgpool-IIの今後

- 次期バージョン3.5について
  - 性能改善
  - watchdog機能の改善
  - pcpコマンドのオーバホール
    - 引数の与え方の改善
    - パスワードをコマンドラインで渡さなくても良いように
    - 複数pcpコマンドの同時実行
      - たとえば時間のかかる pcp_recovery_node の実行中に他のpcpコマンドを実行できる
  - パラレルクエリモードの廃止
    - ユーザが少なく、その割に維持が大変
  - リリースは2015年秋を予定

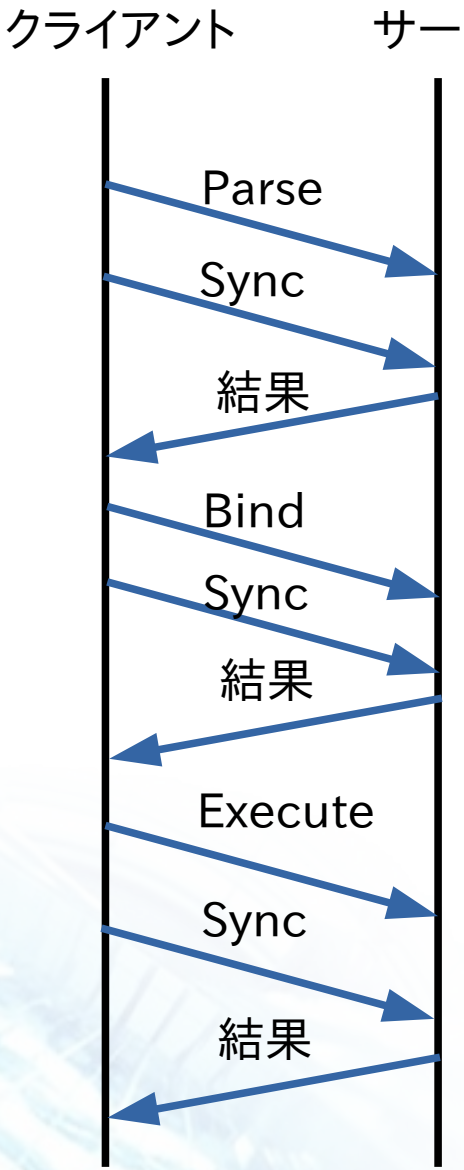
性能改善

- pgpool-IIを拡張プロトコル(Javaなどのprepared statementで使う)と遅い(最悪単純プロトコルの半分位のスピード)
- 遅い主な原因は、pgpool-II での拡張プロトコルの処理の実装による
  - ただし、pgpool-IIを使わなくてもPostgreSQLの拡張プロトコルは単純プロトコルの8割くらいのスピードしか出ない
- そもそも、拡張プロトコルはどのように処理されるのか？

一部やり取りを省略しています



SRA OSS, INC.

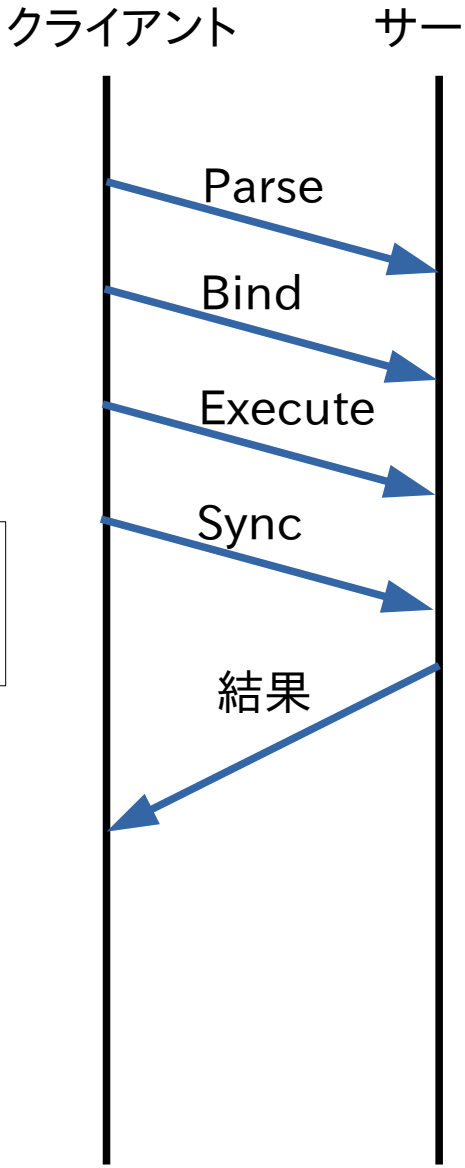


拡張問い合わせ
(pgpool-II使用時)

Syncの発行回数が多いのが性能劣化の原因

ストリーミングレプリケーションでは、Syncを省略できる場合がある

改良!

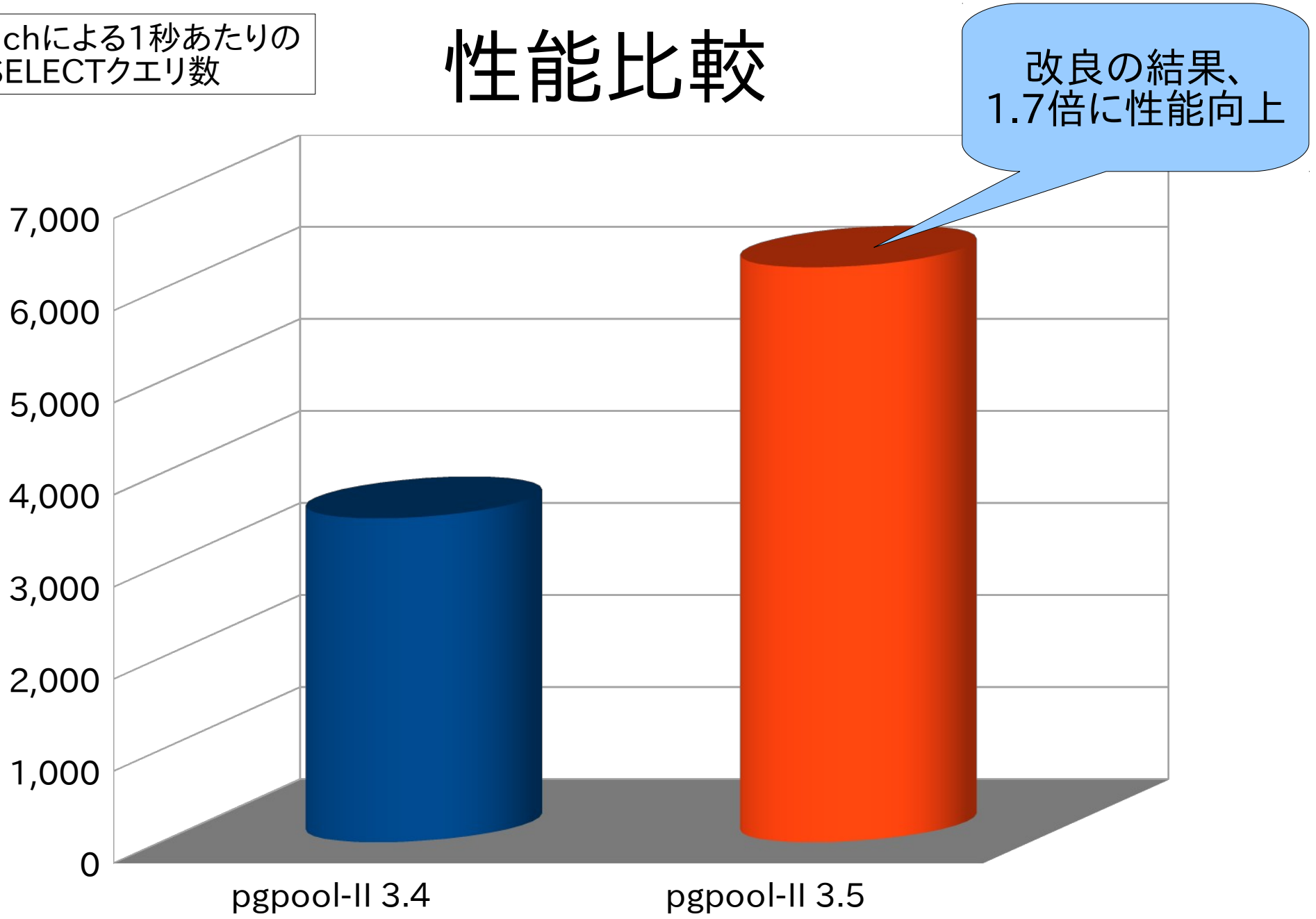


拡張問い合わせ
(pgpool-II使用、改良後)

SRA OSS, INC.

pgbenchによる1秒あたりの
SELECTクエリ数

性能比較



将来の計画

- PostgreSQLの進化に対応
  - ロジカルレプリケーション
  - BDR (Bi Directional Replication)
  - パラレルクエリ
  - どう対応するかは、PostgreSQLの機能を見極めてから
- ドキュメントのオーバーホール
  - 構成が今一つでわかりにくい
    - 構成をPostgreSQLのマニュアルのような感じにしてはどうだろう?
  - ベタのHTMLで書いてあってメンテが困難
    - ツールを使ってベタHTMLから脱出
    - Sphinxはどうだろう?

最後に

- 開発に参加してくださる方を募集しています！
- pgpool-IIの開発に参加するメリット
  - PostgreSQLよりは敷居が低い
    - 規模は1/10位
    - それでいてSQLパーサや例外処理のような重要な部分がポートされているので、PostgreSQLの理解に役立つ
  - ネットワークプログラミングや、マルチプロセスプログラミングの経験蓄積に最適