



# Pgpool-IIの過去、現在、未来

SRA OSS, Inc. 日本支社

石井 達夫

長田 悠吾

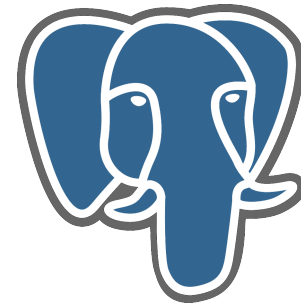
# 自己紹介

- 石井 達夫
  - Pgpool-IIコミュニティリード
  - PostgreSQLコミッタ
  - SRA OSS, Inc. 日本支社支社長
- 長田 悠吾
  - Pgpool-II開発者
  - SRA OSS, Inc. 日本支社でPostgreSQL関連のサポート、コンサル業務に従事



# 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, 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**



# 2003年6月27日: pgpoolの誕生

- **コネクションプーリング、フェイルオーバーのみ**
- 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 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オリティですが,よろしかったらお試しください.

<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なっています.

**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」というものがあるが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

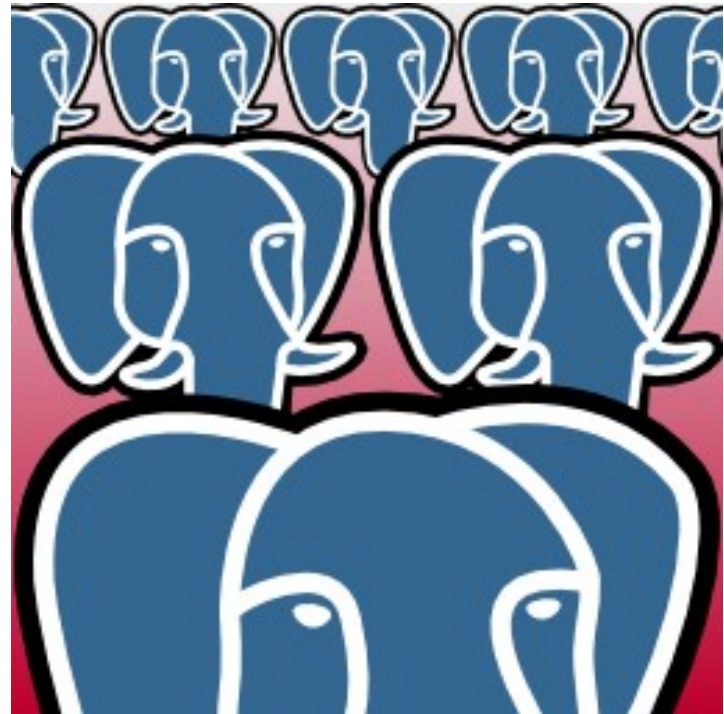
- 現在の「**ネイティブ・レプリケーションモード**」に相当する機能を実装した(まだPostgreSQLにはレプリケーション機能がなかった)
- クエリキャンセル対応
- ラージオブジェクトのレプリケーション対応
- C言語で5,890行
- この頃は、マイナーリリース(x.x)の際にも平気で機能を追加していたりして、かなりいい加減なリリース管理がされていた



現代の象になったがまだまだよちよち歩き

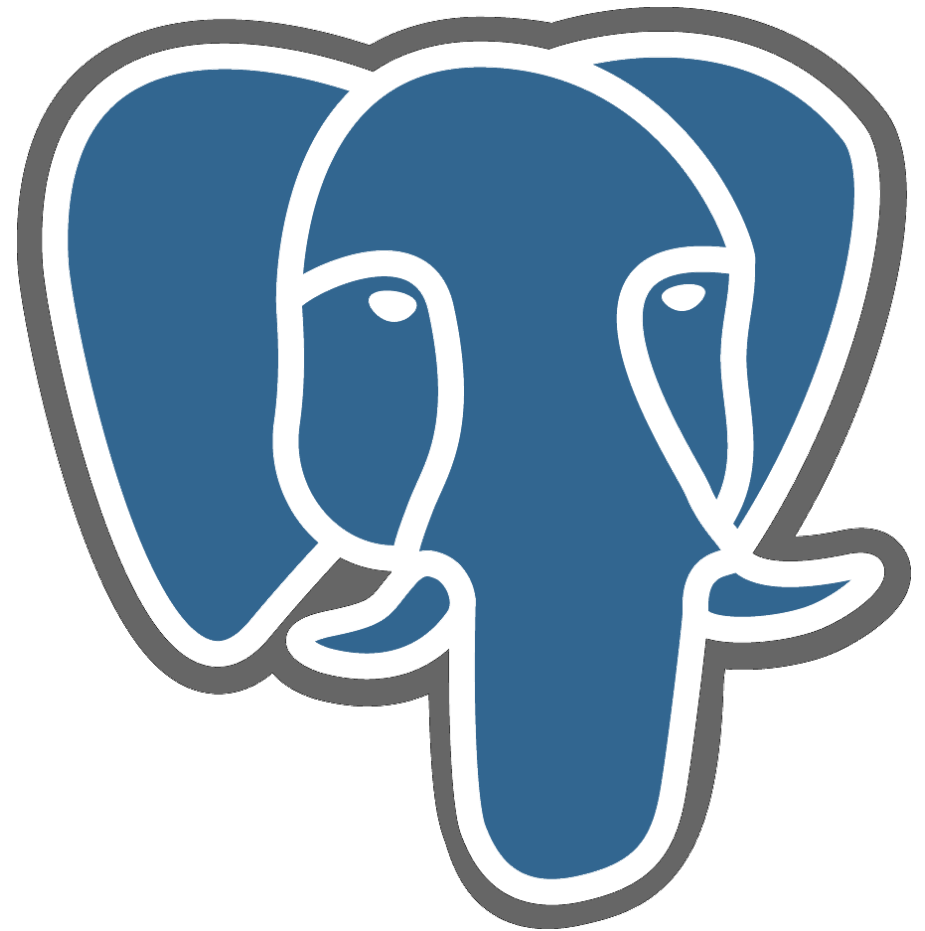
# pgpool 2.0へ進化

- 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プロトコルにネイティブ対応
- 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



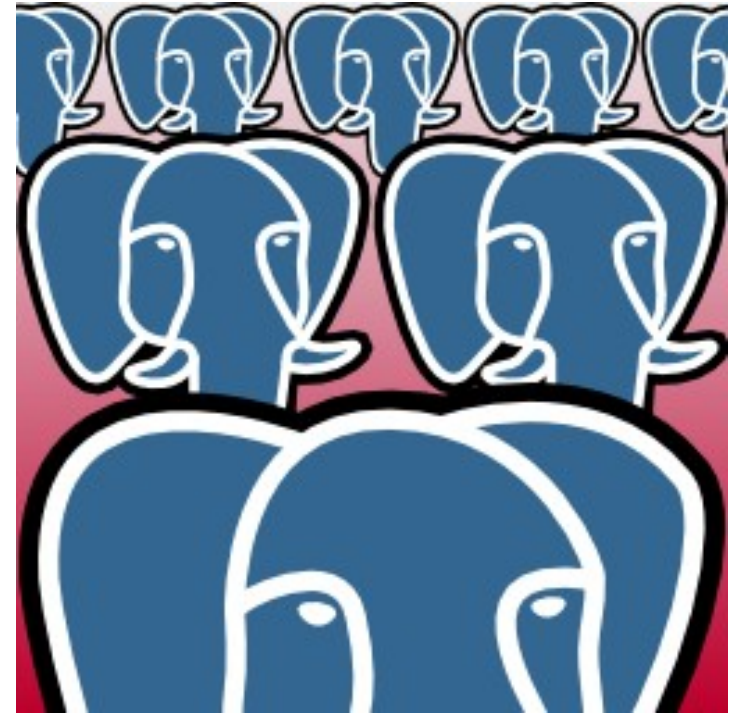
# その頃のPostgreSQL

- PostgreSQL 7.4(2003年11月)
  - V3プロトコルが導入され、プロトコルレベルのprepared queryが使えるようになった
  - autovacuumの導入
  - 全文検索機能がcontribに導入された
  - まだWindows未対応
  - レプリケーション未対応



# 2006年9月:Pgpool-II 1.0の誕生

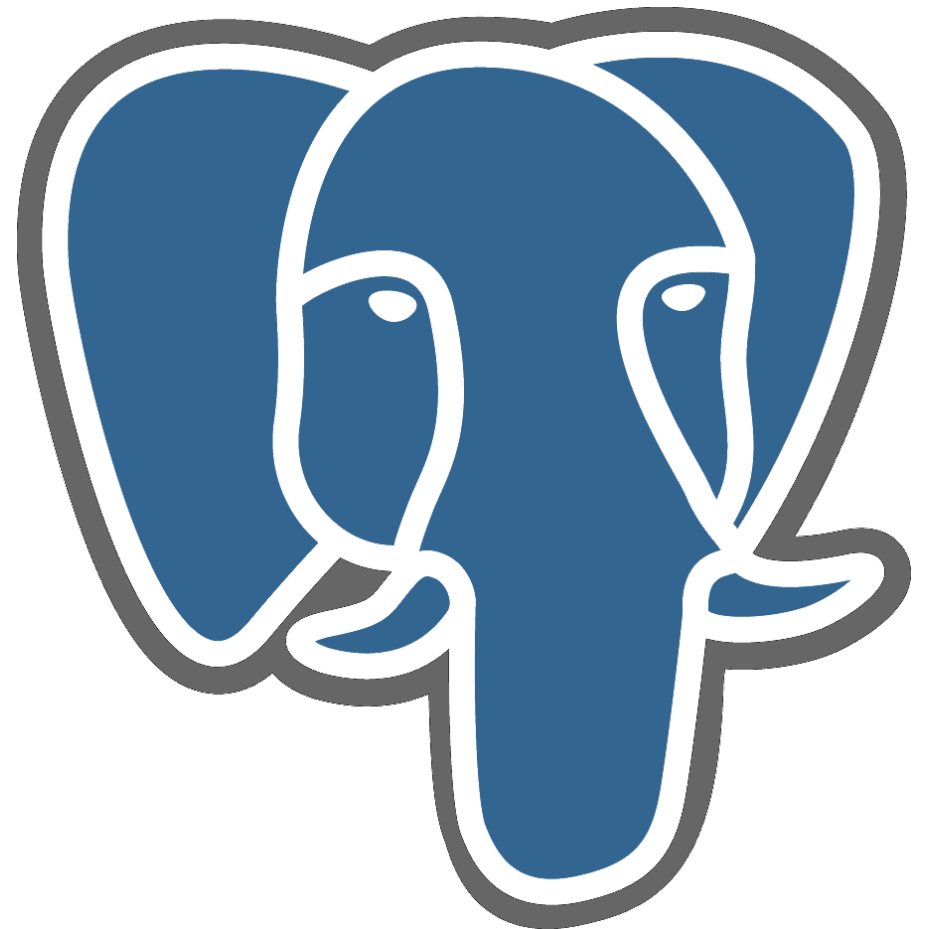
- 開発手法の変更
  - 個人プロジェクトから、チーム作業へ
  - IPAの援助で開発
- 機能の大幅追加、現在の姿にほぼ近づく
  - サーバ台数の制限撤廃
  - SQLパーサを搭載して精密な構文解析
  - 管理コマンド(pcp)の実装
  - GUI管理ツール(pgpoolAdmin)の実装
  - パラレルクエリモードの実装
  - C言語で73,511行と、一気に10倍近い規模に増えた(bison, flexコード行数を含む)





# その頃のPostgreSQL

- PostgreSQL 8.2(2006年)
  - Windows対応(8.0)
  - PITR(8.0)
  - save point(8.0)
  - テーブルスペース
  - 二相コミット(8.1)
  - ビットマップインデックススキャン(8.1)
  - 行ロック(8.1)
  - レプリケーション未対応



# 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

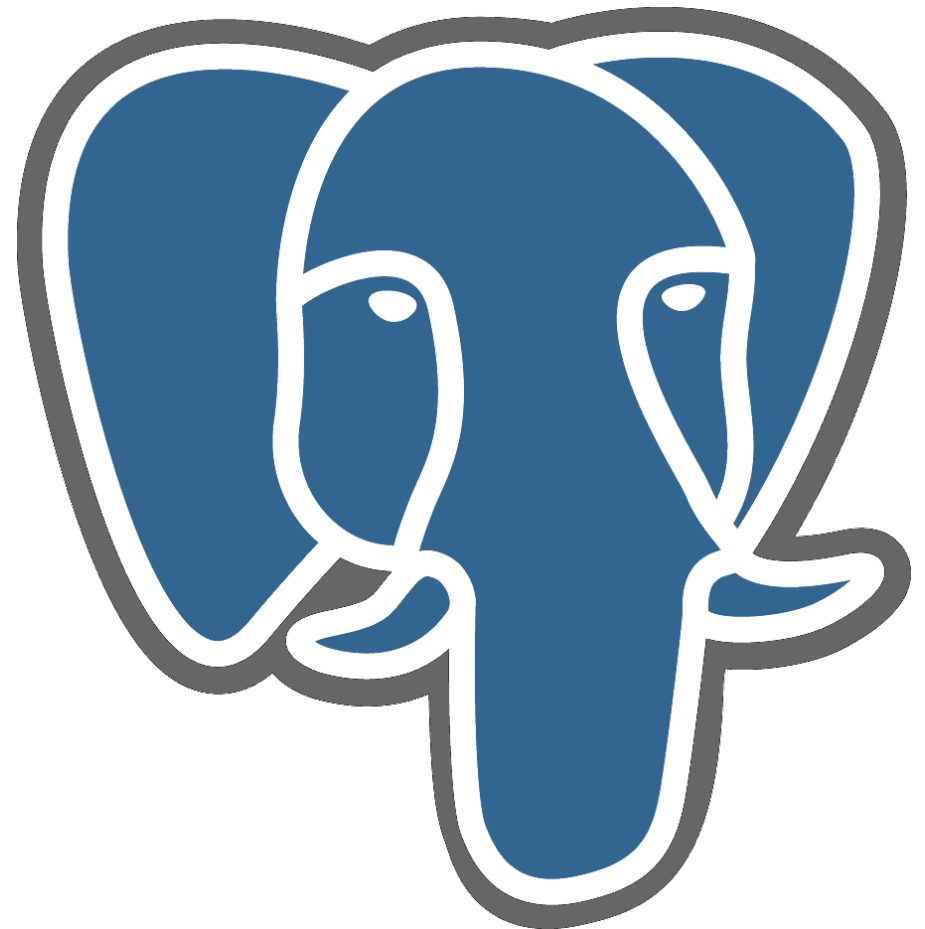


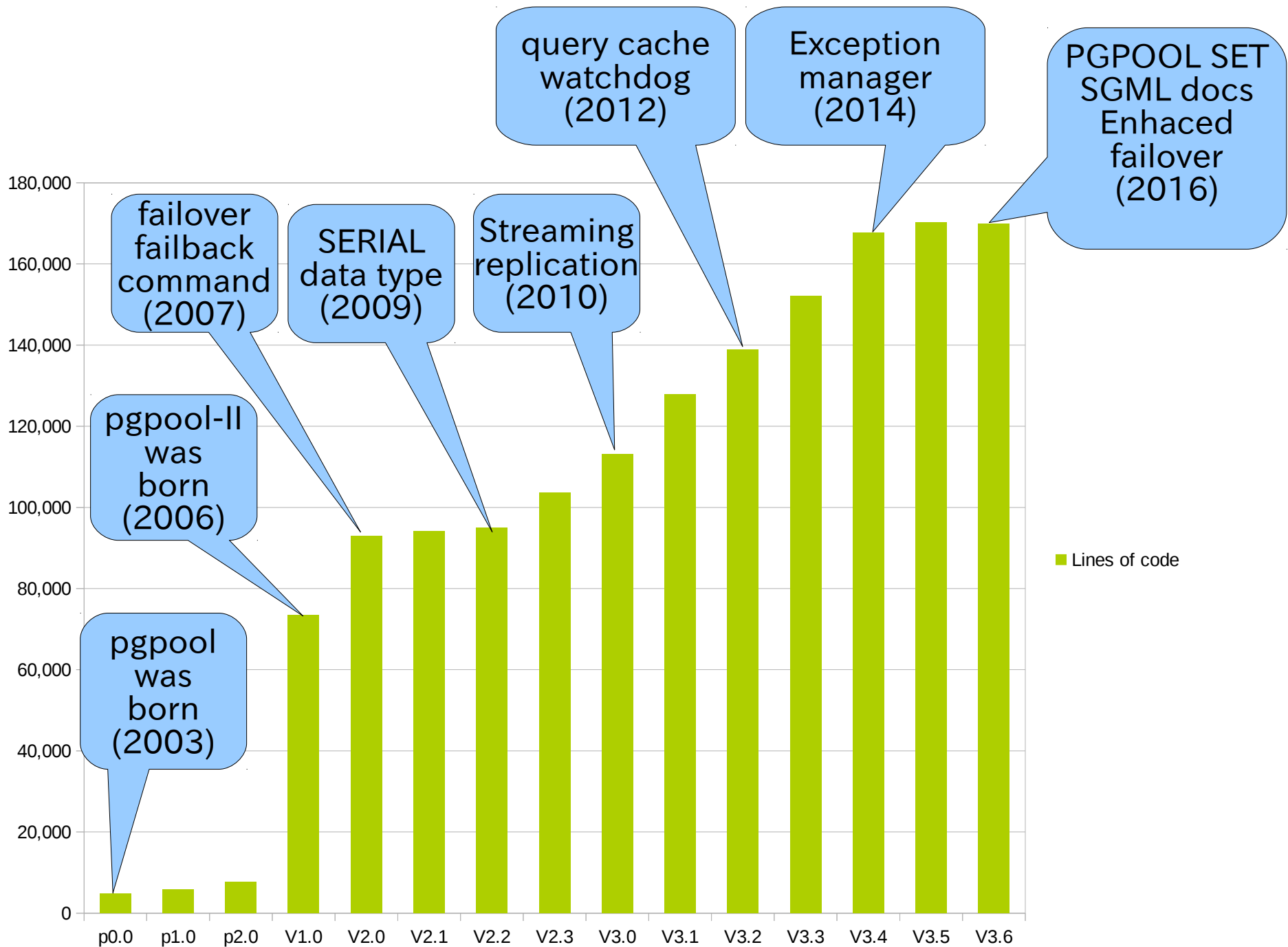
引っ越しはなかなか大変でした

- 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

# その頃のPostgreSQL

- PostgreSQL 9.1 (2011年)
  - ストリーミングレプリケーション(9.0)
  - Windows 64bit対応 (9.0)
  - pg\_upgrade(9.0)
  - 同期レプリケーション(9.1)
  - 外部テーブル(9.1)
  - 再帰問い合わせ(8.4)





# 現在の開発体制

- 石井(日本)
  - 全体のとりまとめ。コードも書きます
- Ahsan Hadi(パキスタン)
  - ユーザニーズの取り込み、ベンチマーク
- Muhammad Usama(パキスタン)
  - watchdogやその他幅広く担当。コミット
- 長田(日本)
  - watchdogを中心に担当。コミット
- 安齋(日本)
  - pgpoolAdminとインストーラを中心に担当。コミット
- 彭(中国)
  - 安齋さんの後を引き継いでリリース作業やRPM作成を実施。コミット



# Pgpool-IIの現在

- PostgreSQLのクラスタの総合管理ツールに進化
  - pgpool 1.0 (5,890行)からPgpool-II 3.6.0 (177,948行)へと、30倍の規模に成長
  - 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ツールとして
  - 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

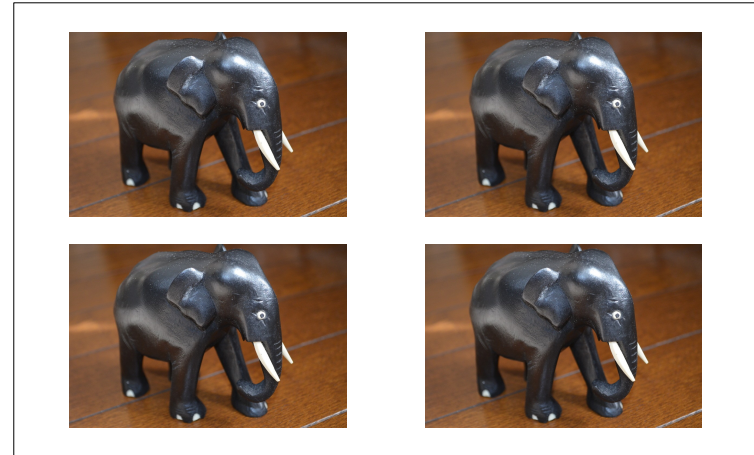
# Pgpool-IIのご紹介と使いどころ





# 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？



||

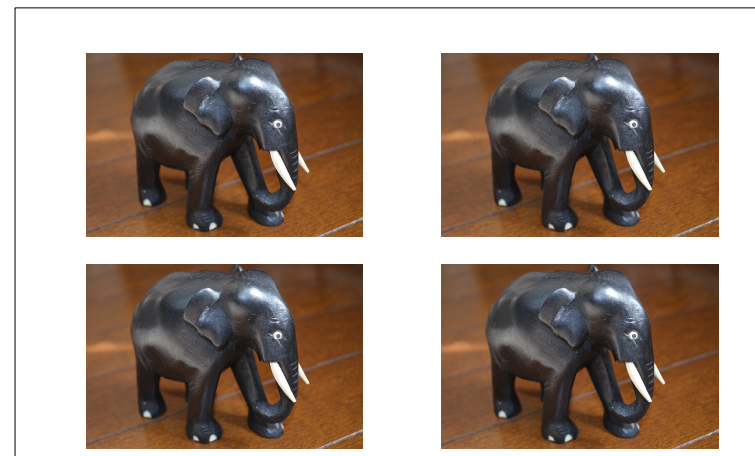


# 「象の群れ管理問題」の一例

- 群れにはリーダーが必要です(プライマリサーバ)
- もしリーダーが引退したら、新しいリーダー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ー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)

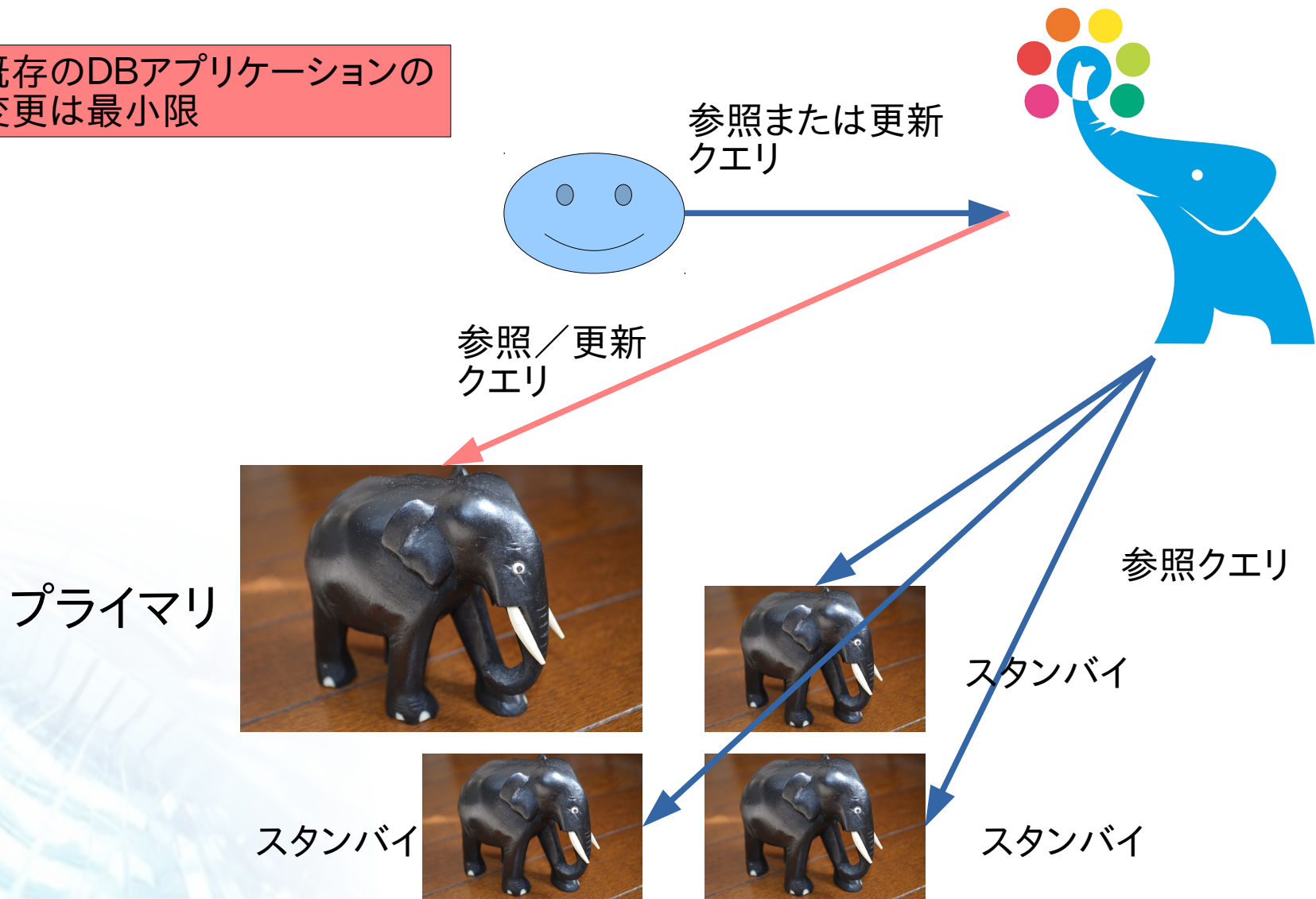
# Pgpool-IIで 「象の群れ管理問題」を解決

- 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ースキームにより、フェイルオーバー時にどのスタンバイが昇格するか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

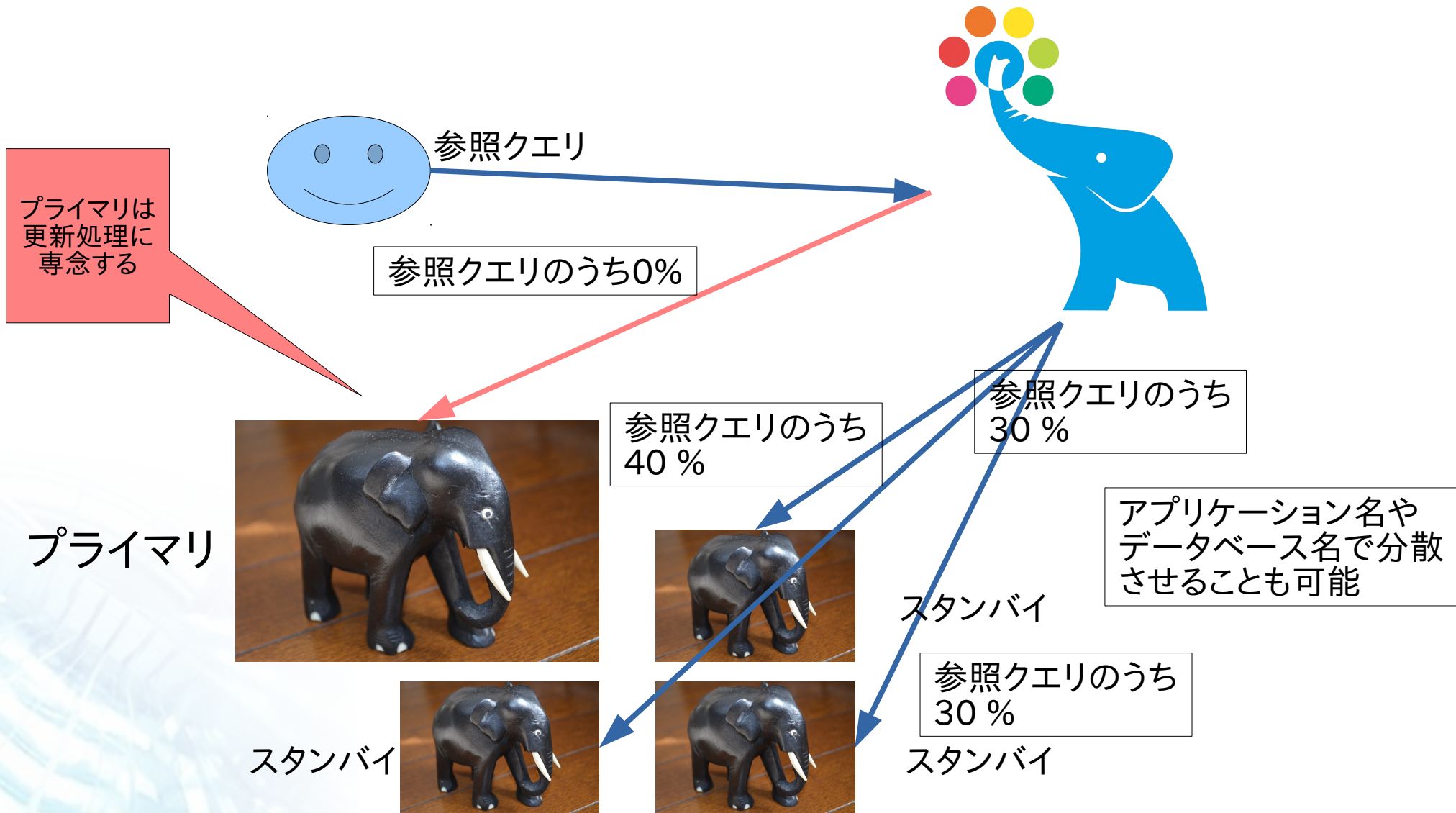


# クエリのディスパッチ/ルーティング

既存のDBアプリケーションの  
変更は最小限



# 負荷分散



# 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なこともある



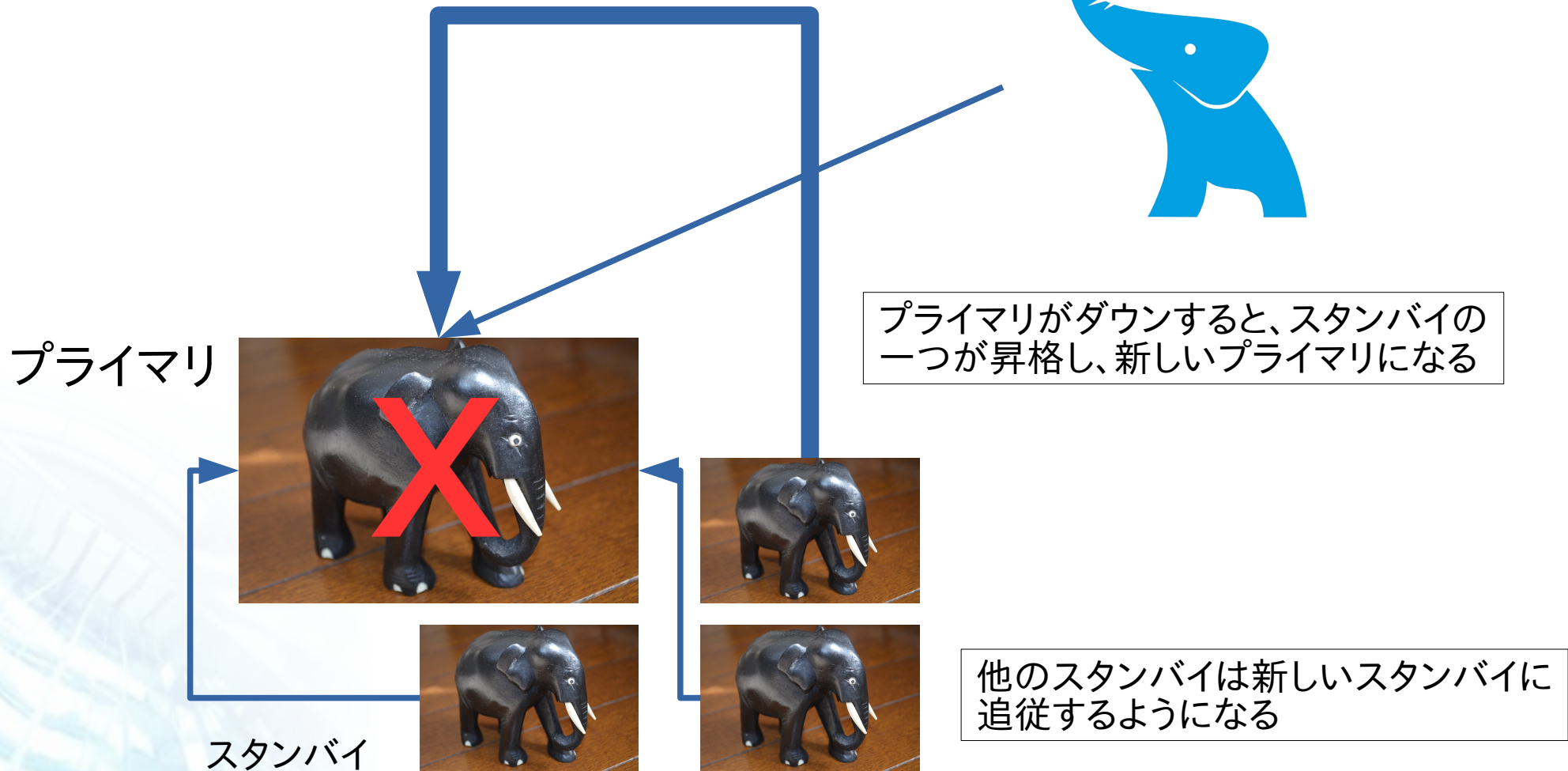
プライマリ



スタンバイ



# プライマリサーバがダウンした時



# 新しいスタンバイの追加



新しい象!



プライマリ



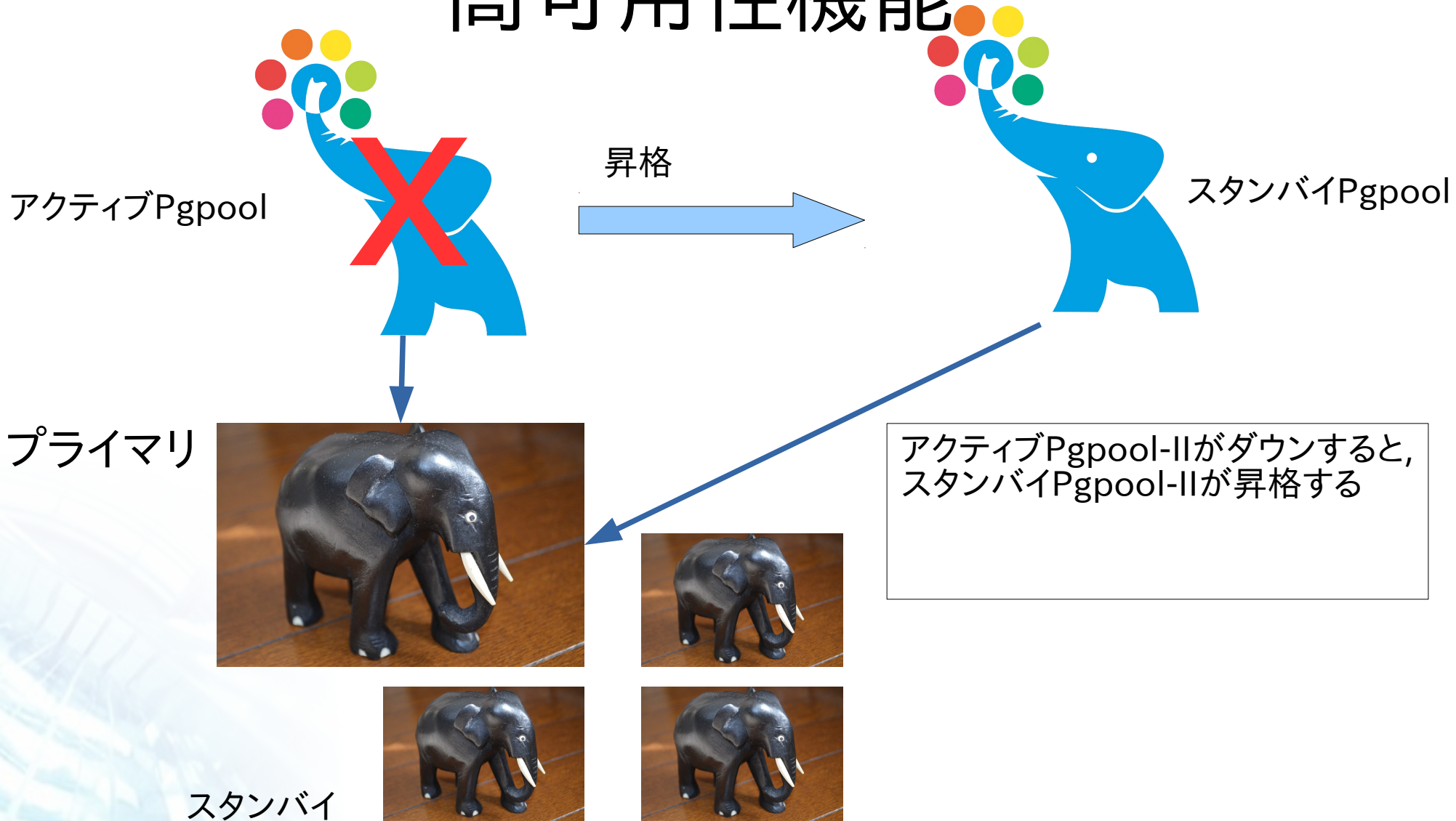
スタンバイ



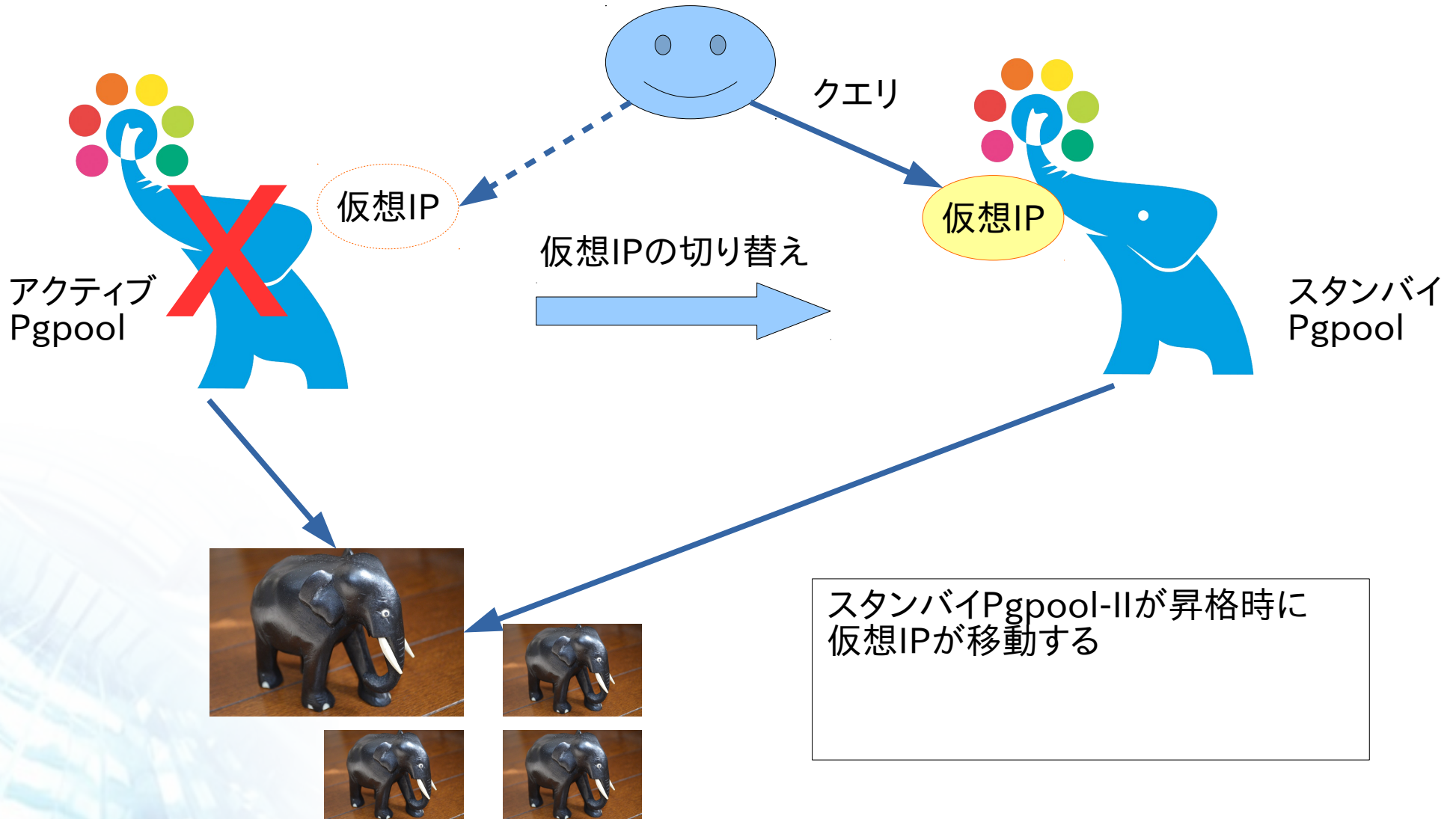
新しいサーバは簡単に追加できる。  
Pgpool-IIは新しいサーバにプライマリからデータをコピーし、他のサーバに影響を与えずにクラスタに新しいサーバを追加できる。  
既存のセッションは切断されない。



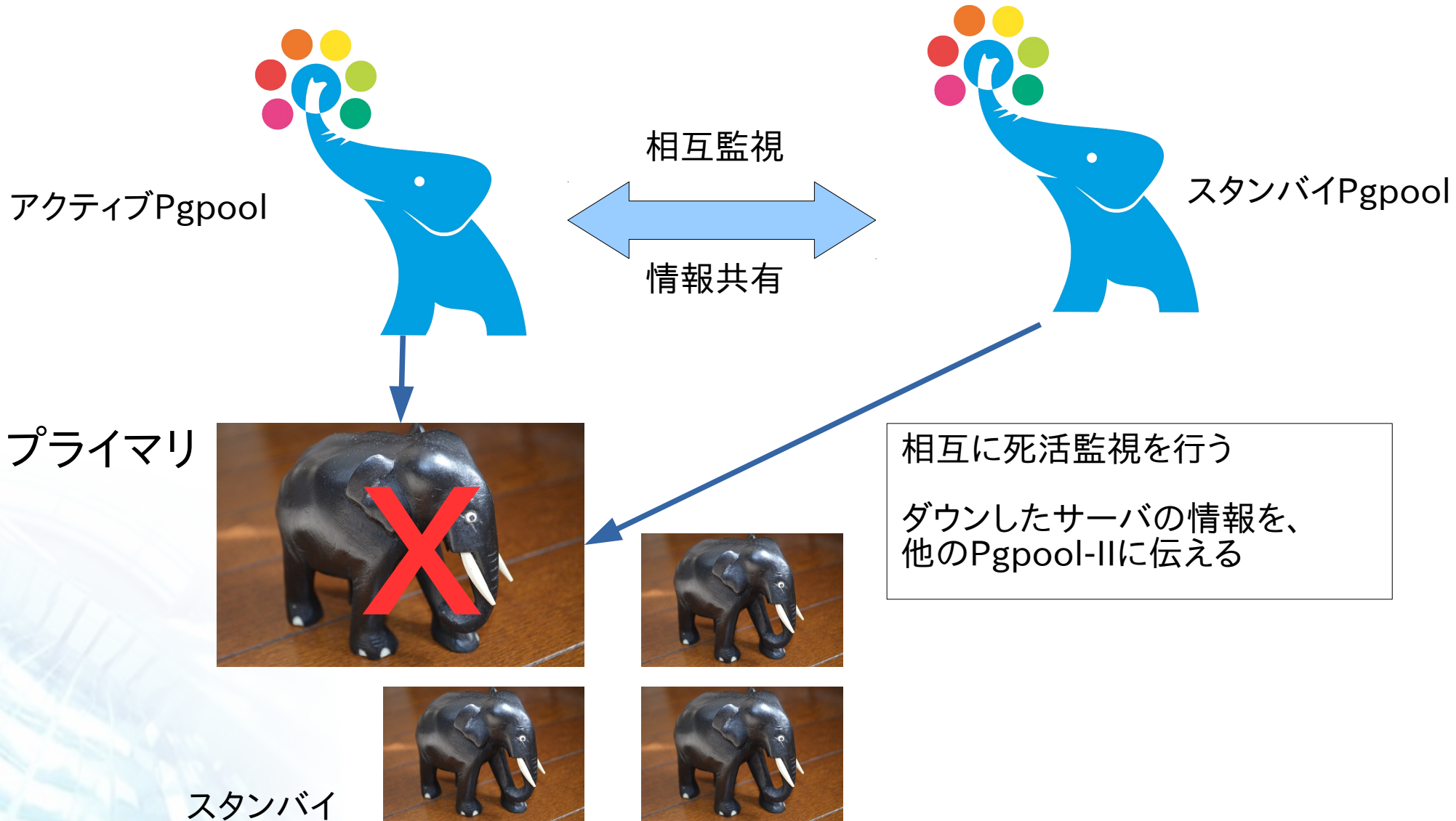
# Watchdog: Pgpool-II組み込みの 高可用性機能



# Watchdog: 仮想IPの切り替え

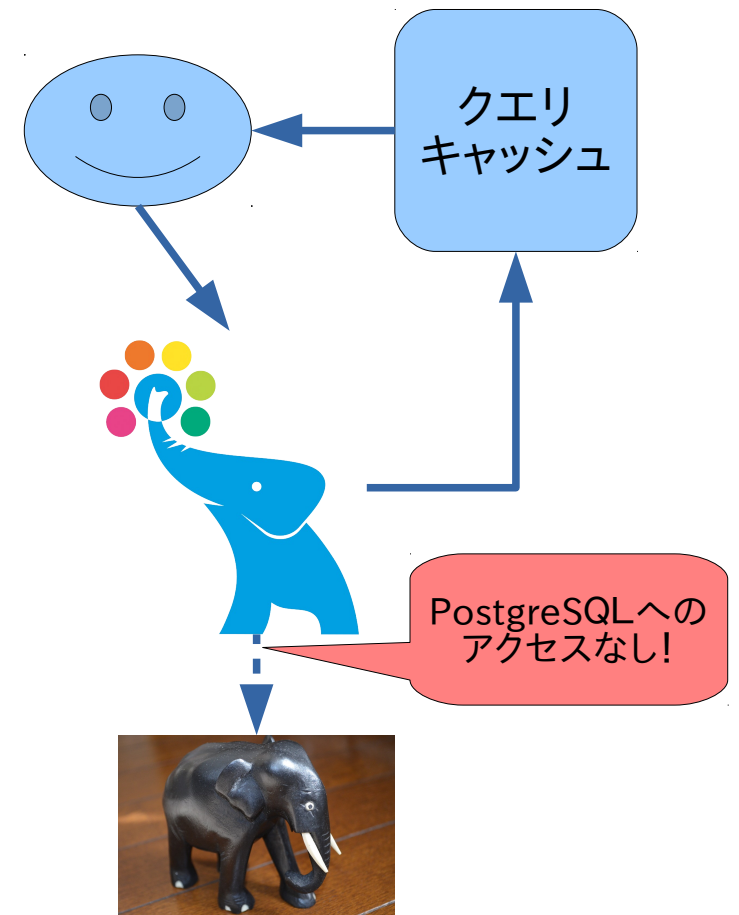


# Watchdogによる協調動作



# 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、共有メモリか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



# 最新バージョンPgpool-II 3.6の ご紹介

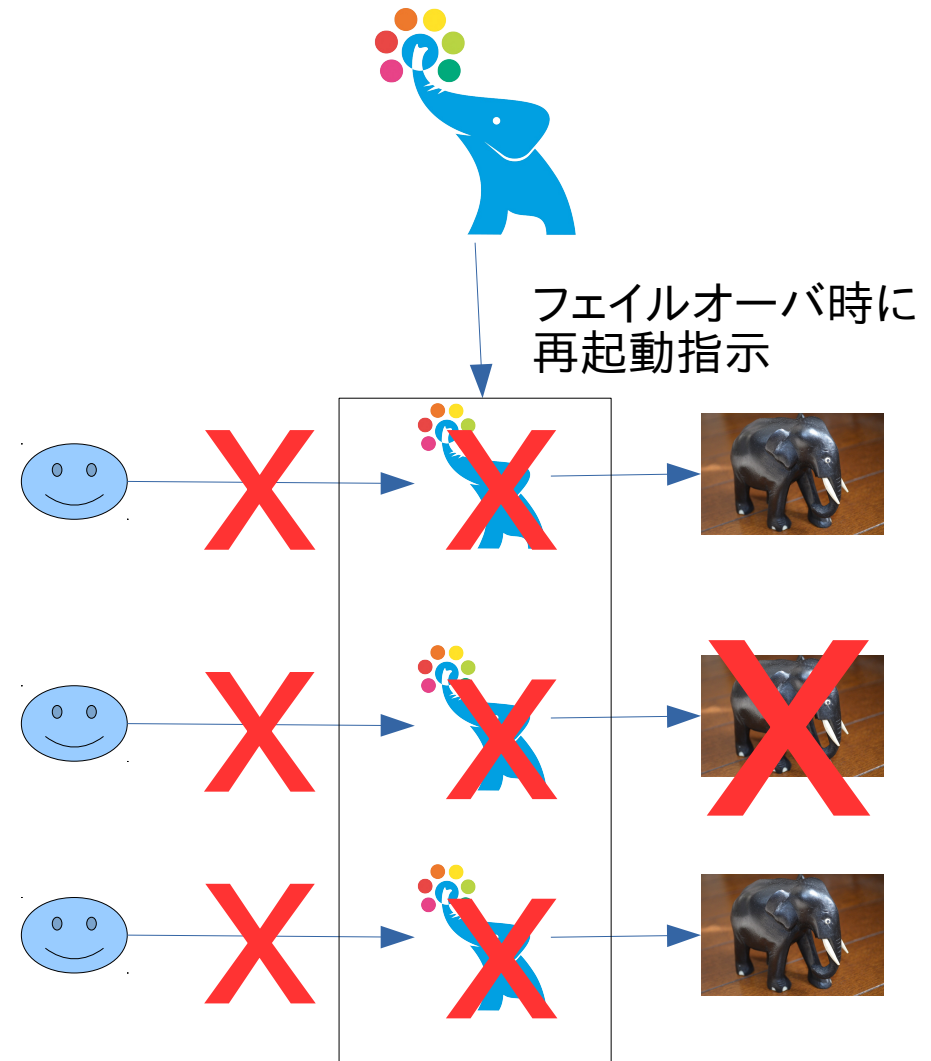
- フェイルオーバの改善
- PostgreSQL 9.6対応
- その他



# フェイルオーバーの改善

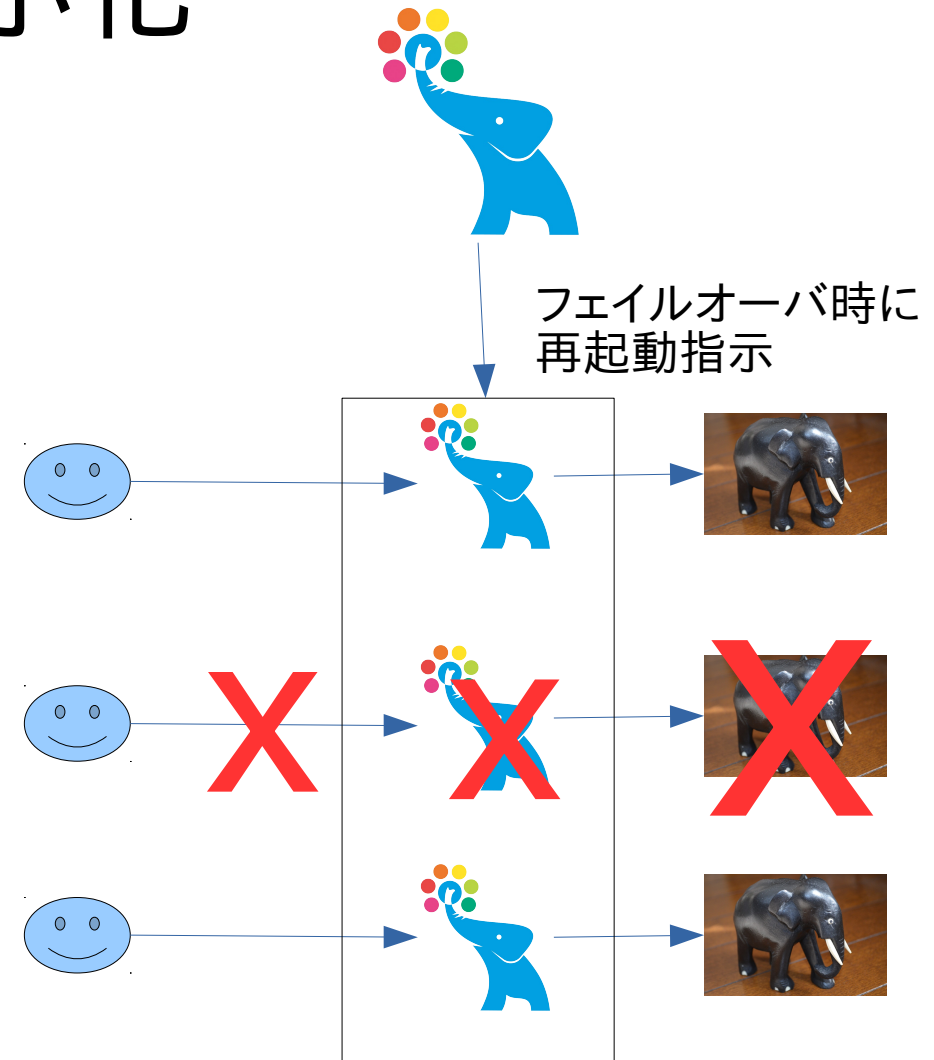
# フェイルオーバーにおける問題点

- 複数のPostgreSQLサーバを使ったクラスター構成で、そのセッションが使用していないDBサーバがダウンした時にもセッションが一旦切断される



# Pgpool-II 3.6では切断されるセッションを最小化

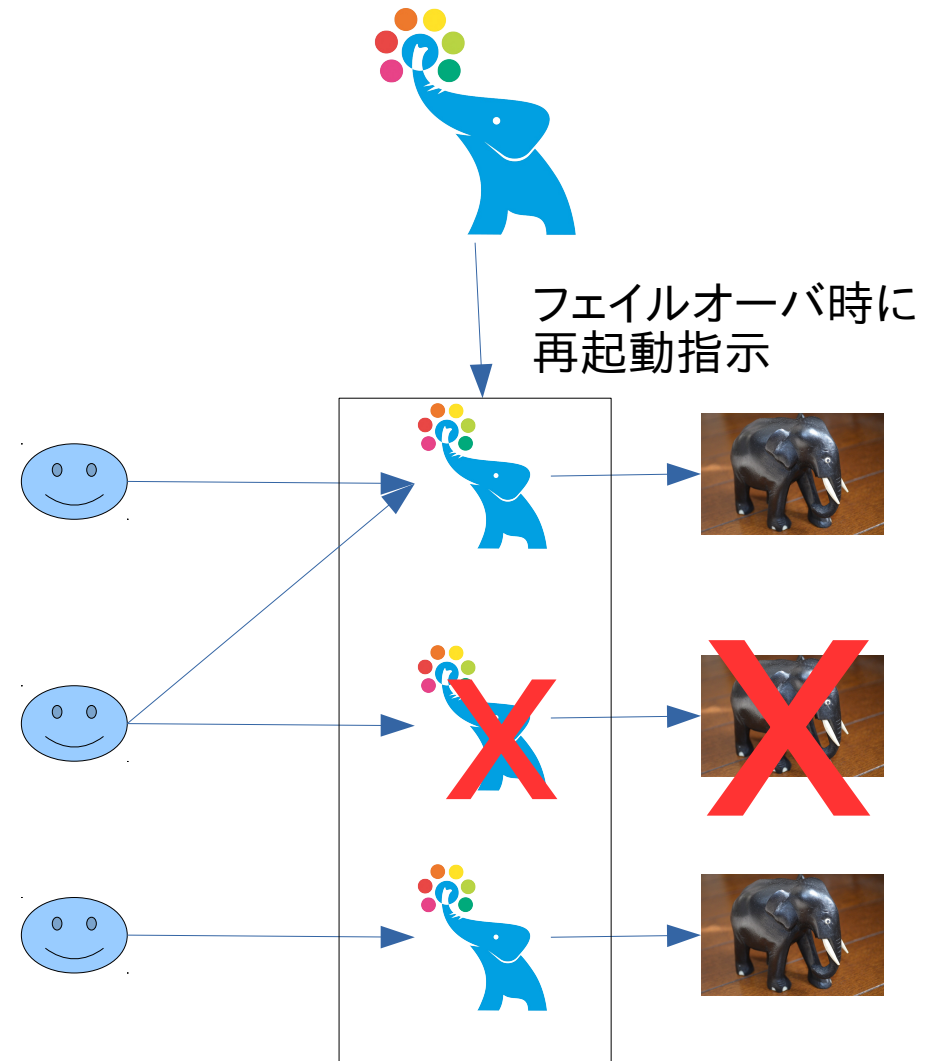
- 前提条件: ストリーミングレプリケーションモード、ダウンしたDBサーバはスタンバイ
- ダウンしたサーバを使っているセッションに関連したPgpool-II子プロセスのみ再起動指示
- それ以外のセッションに関連したPgpool-II子プロセスは、現在のセッションが終了後、自動的に再起動してダウン情報を反映する





# PostgreSQLの計画停止が容易に

- あらかじめ、停止予定のDBサーバに対する負荷分散の重みを0に設定し、設定ファイルを再ロード
- 新しいセッションでは停止予定のDBを使わなくなる
- DBを停止しても影響を受けるセッションなし



# 使用しているDBサーバの確認がセッションごとに可能に

- 使用しているDB(負荷分散ノード)はセッションごとのプロパティなので、PCPコマンドなどでは確認ができない
- そこで、show pool\_nodes で、現在のセッションの負荷分散ノードを確認可能に
- そのほか、レプリケーション遅延も確認可能に

```
psql -p 11000 -c "show pool_nodes" test
```

node_id	hostname	port	status	lb_weight	role	select_cnt	load_balance_node	replication_delay
0	/tmp	11002	up	0.333333	primary	0	false	0
1	/tmp	11003	up	0.333333	standby	0	true	0
2	/tmp	11004	up	0.333333	standby	0	false	0

(3 rows)

# PostgreSQL 9.6対応

# PostgreSQL 9.6 SQLパーサの移植

- Pgbpool-IIでは、SQL文を正確に解析するためにSQLパーサを持っている
  - SQLパーサは、最新のPostgreSQLから移植
- 以下の新しい構文をサポート
  - COPY FROM INSERT ... RETURNING TO ...
  - ALTER FUNCTION ... DEPENDS ON EXTENSION ...
  - ALTER TABLE ADD COLUMN IF NOT EXISTS ...
  - など

# Pgpool-II 3.6その他の改良

# PGPOOL SET

- PostgreSQLの“SET” コマンドに相当するもの
- Pgpool-IIの一部の設定変数をセッション内で変更可能
  - セッション終了後に自動リセットされる
  - 構文は、“PGPOOL SET 変数名 TO 値”
- SETできる変数
  - client\_idle\_limit
  - client\_idle\_limit\_in\_recovery
  - log\_statement
  - log\_per\_node\_statement
  - log\_min\_messages
  - client\_min\_messages
  - log\_error\_verbosity
  - allow\_sql\_comments
  - check\_temp\_table
  - check\_unlogged\_table

# PGPOOL SHOW

- Pgpool-IIの設定変数を個別に表示
  - PGPOOL SHOW 変数名
- 「変数グループ」(“logical group”)別の表示も可能
  - “backend”, “other\_pgpool” (他のwatchdogノード)、  
“heartbeat”

```
postgres=# pgpool show backend;
```

item	value	description
backend_hostname0	127.0.0.1	hostname or IP address of PostgreSQL backend.
backend_port0	5434	port number of PostgreSQL backend.
backend_weight0	0	load balance weight of backend.
backend_data_directory0	/var/lib/pgsql/data	data directory of the backend.
backend_flag0	ALLOW_TO_FAILOVER	Controls various backend behavior.
backend_hostname1	192.168.0.1	hostname or IP address of PostgreSQL backend.
backend_port1	5432	port number of PostgreSQL backend.
backend_weight1	1	load balance weight of backend.
backend_data_directory1	/home/usama/work/installed/pg	data directory of the backend.
backend_flag1	ALLOW_TO_FAILOVER	Controls various backend behavior.

(10 rows)

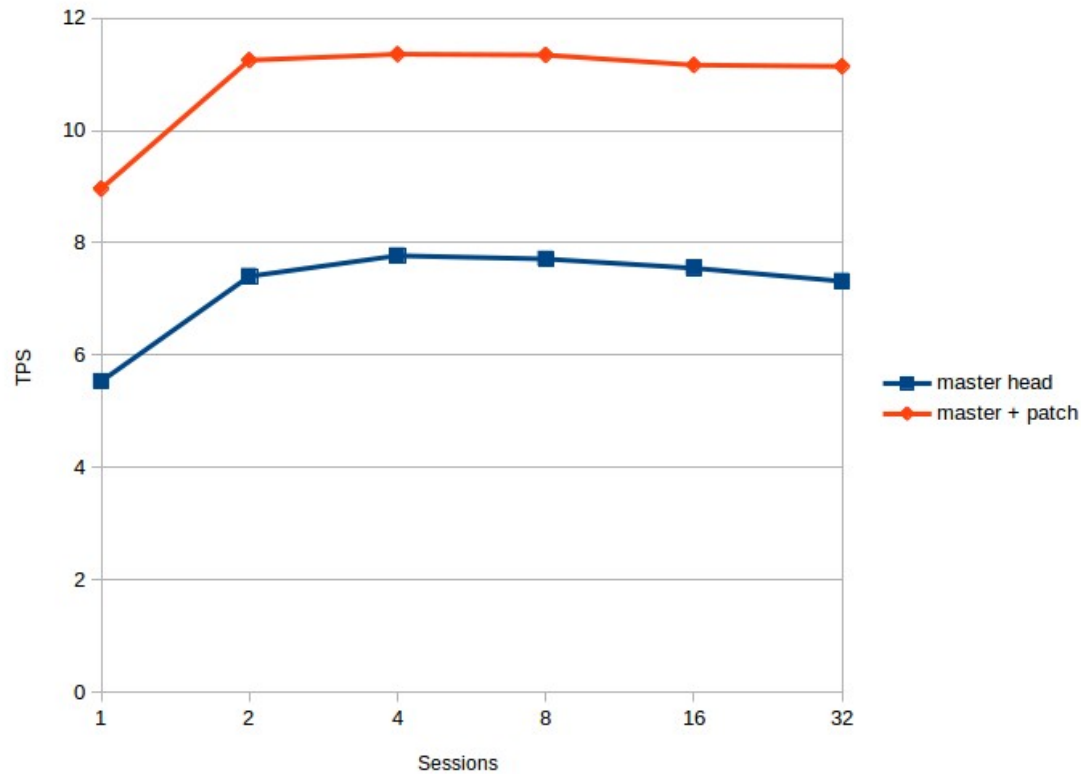
# pg\_terminate\_backend対応

- pg\_terminate\_backendとは
  - PostgreSQLの組み込み関数の一つ。プロセスIDを指定して、特定のバックエンドプロセスを終了することができる。主に、無限ループに入ってしまった、ロックを掴みっぱなしになっている、などの状態になったバックエンド終了させるための機能
- Pgpool-IIにとっての問題点
  - DBのシャットダウンと同じエラーコードがpg\_terminate\_backendの実行でもPostgreSQLから返るため、Pgpool-IIはDBがシャットダウンされたと解釈してフェイルオーバーしてしまう
- 解決策
  - pg\_terminate\_backendの引数を調べ、指定プロセスIDがどれかのセッションで使われているバックエンドである場合には、該当エラーコードがPostgreSQLから返ってきたとしても、フェイルオーバーを起こさず、該当セッションを切断するだけにする
- 制限事項
  - pg\_terminate\_backendの引数は単純整数でなければならない
    - SELECT pg\_terminate\_backend(pid) from strange\_table; とかは駄目
  - 拡張プロトコル未対応



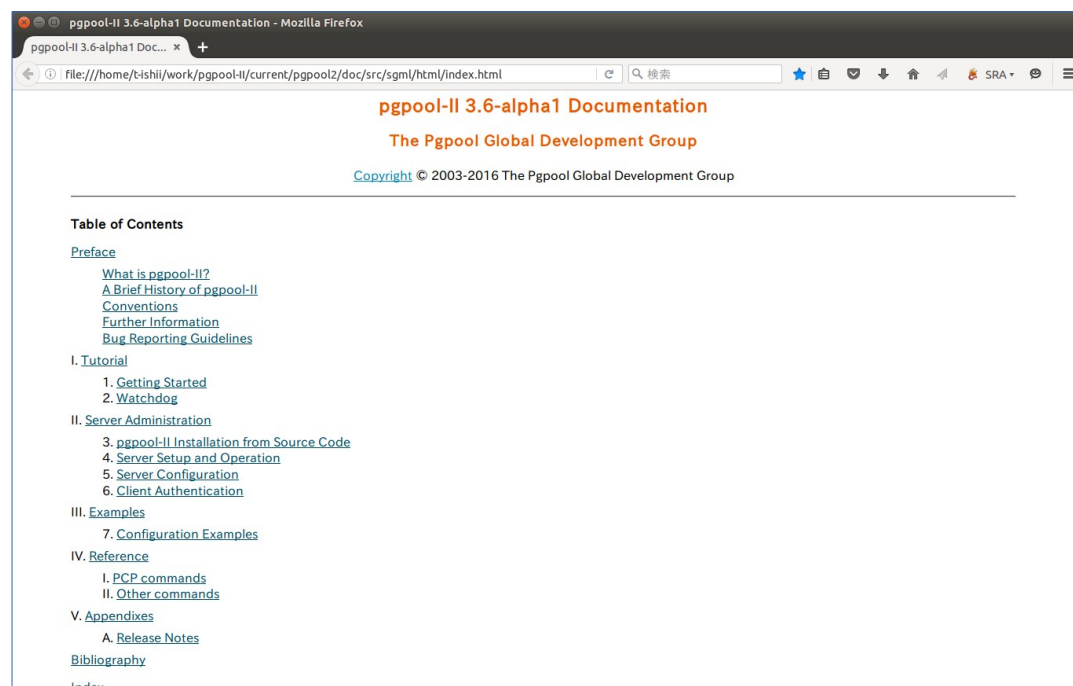
# SELECT結果が多い時の 性能改善

- 検索結果をクライアントに返すときのオーバーヘッドを改善し、特定のケースでは47%から62%の性能改善
- 従来1行返す毎にシステムコールを読んでいたのを、行単位でバッファリングを行い、最後に一括でシステムコールを呼ぶようにして性能改善



# ドキュメントフォーマットの変更

- 今までは、手打ちのHTMLを使用
  - メンテナンスが大変
  - 索引や目次などを自動的に作れない
- Pgpool-II 3.6では、PostgreSQLと同様、SGML → HTMLという仕掛けを採用
- 最初はSGMLファイルを作るのが大変だが、保守は楽になるはず(と信じてます)



pgpool-II 3.6-alpha1 Documentation - Mozilla Firefox

pgpool-II 3.6-alpha1 Doc... x +

file:///home/t-ishii/work/pgpool-II/current/pgpool2/doc/src/sgml/html/index.html

pgpool-II 3.6-alpha1 Documentation  
The Pgpool Global Development Group  
Copyright © 2003-2016 The Pgpool Global Development Group

Table of Contents

[Preface](#)

- [What is pgpool-II?](#)
- [A Brief History of pgpool-II](#)
- [Conventions](#)
- [Further Information](#)
- [Bug Reporting Guidelines](#)

I. [Tutorial](#)

- 1. [Getting Started](#)
- 2. [Watchdog](#)

II. [Server Administration](#)

- 3. [pgpool-II Installation from Source Code](#)
- 4. [Server Setup and Operation](#)
- 5. [Server Configuration](#)
- 6. [Client Authentication](#)

III. [Examples](#)

- 7. [Configuration Examples](#)

IV. [Reference](#)

- I. [PCP commands](#)
- II. [Other commands](#)

V. [Appendixes](#)

- A. [Release Notes](#)

[Bibliography](#)

[Index](#)

# Pgpool-IIの今後

- PostgreSQLのレプリケーション技術の進化に合わせてPgpool-IIも進化していきます
  - 同期レプリケーション
  - カスケードレプリケーション
  - マルチマスタレプリケーション
- クラスタ全体の状態管理を容易に
  - クラスタ全体の状態を見渡し、管理するのは難しい
  - Pgpool-IIがその作業を容易にする
- 自動フェイルバック
  - 自動的に復旧したスタンバイや、新しく追加されたスタンバイを認識してクラスタに追加する
- プラグインアーキテクチャ
  - 例: 負荷分散ロジックをユーザが定義できるように

# まとめ

- Pgpool-IIはコネクションプーリングソフトから出発して、多機能なクラスタ管理ソフトに成長
- 個人プロジェクトから、集団開発のOSSプロジェクトへ
- 独自のレプリケーション機能に加えて、ストリーミングレプリケーションに対応するなど、PostgreSQLの進歩に合わせて進化
- 今後も「PostgreSQLに寄り添う」というコンセプトを維持しつつ、PostgreSQLの進化に歩調を合わせて行きます

# URLなど

- Pgpool-II公式サイト
  - <http://www.pgpool.net>
    - ダウンロード
    - Gitリポジトリ
    - RPM
- Twitter
  - @pgpool2
- GitHub
  - <https://github.com/pgpool/pgpool2>
    - 単なるミラーです。PRやIssuesへの反応は遅いかもしれません

# Thank you!

