pgpool-IIのご紹介

PostgreSQLで並列分散処理を実現するミドルウェア

日本PostgreSQLユーザ会/
SRA OSS Inc.
浅羽 義之

アジェンダ

1. 背景
2. pgpool-IIの紹介
3. 今後のロードマップ

PostgreSQLとは？

- 信頼性が高く，高機能のオープンソースデータベース
- 小規模システムから大規模基幹システムまで幅広く活用できる
- ハードウェアの性能を生かした大量高速処理が可能
- 広く普及し，情報も豊富
- 最新版は8.2で、8.3が年内にリリースされる予定

PostgreSQL(RDBMS)に対する要求

- とにかくダウンタイムを減らしたい
  - DBが止ってしまうと、サービス全体が止ってしまう！
- 最近、DBのレスポンスがよくない
  - チューニングやハードウェアを補強したが効果なし

速くて信頼が求められる(厳しい要求)

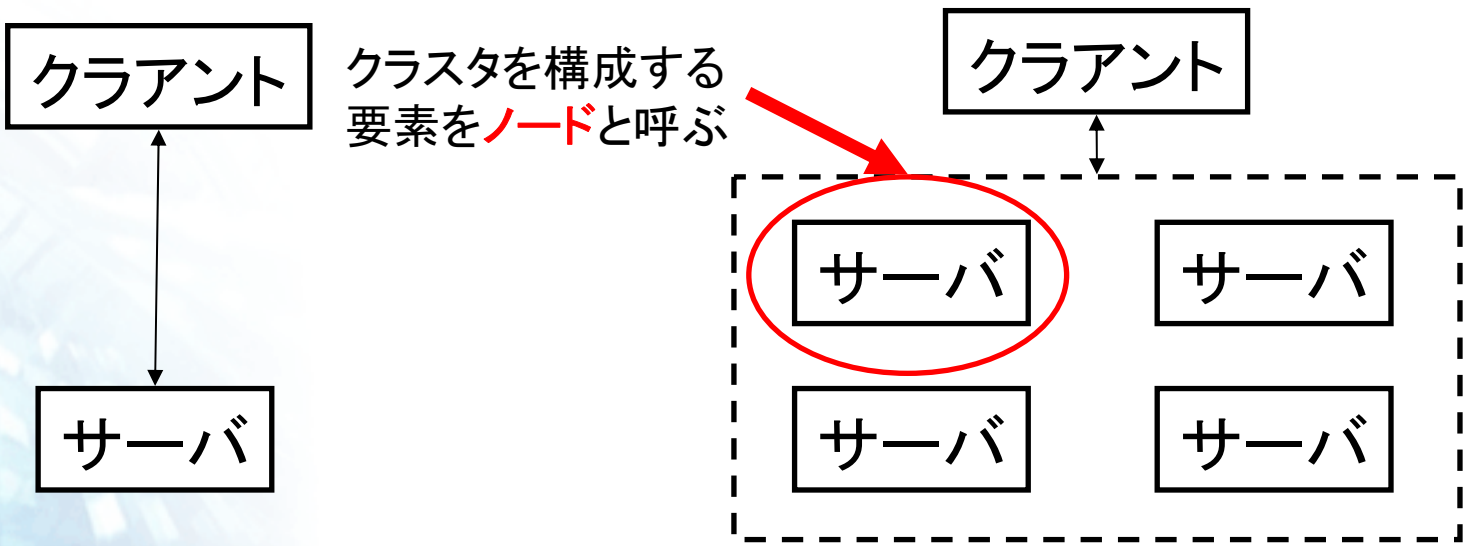
シングルサーバの限界

- CPU
  - CPU追加によるスケールアップの限界
- メモリ
  - 搭載メモリ量の限界
- ディスク
  - ハードウェア故障による可用性の低下

複数のマシンに仕事をさせればよい！

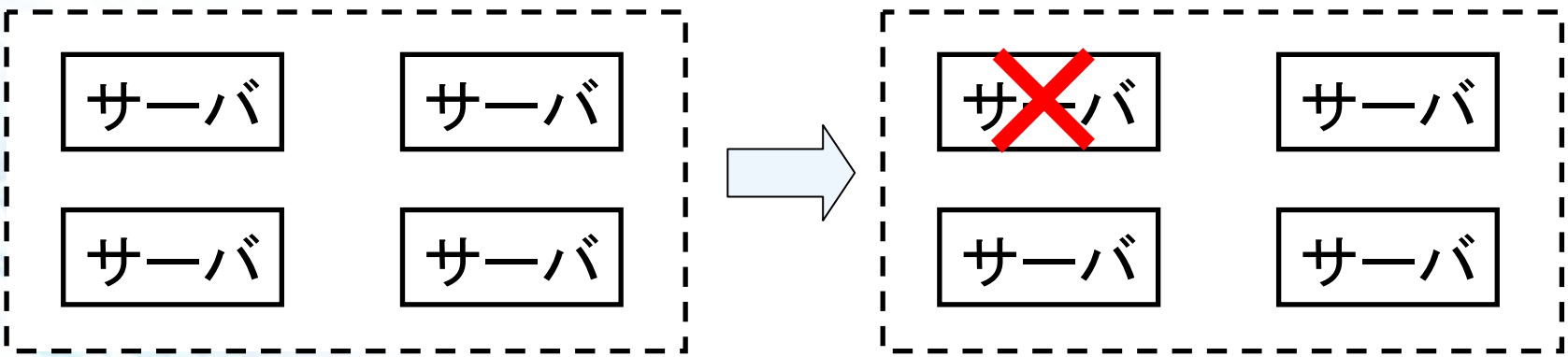
クラスタリングとは？

- 複数台のシステムを使って1つのサービスを提供する仕組み
  - ユーザから見れば、単体で動いているサービスなのかクラスタリングシステムなのかは見えない



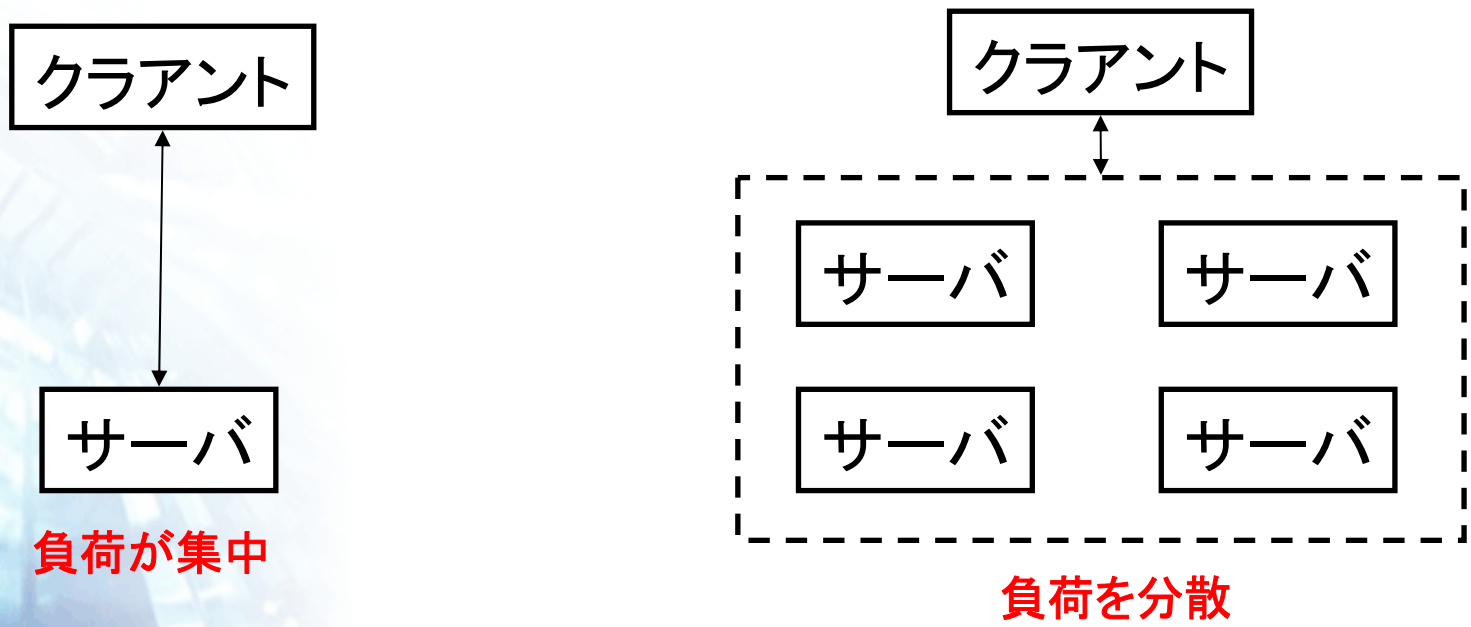
冗長化とは？

- 1台異常が発生してもサービスを提供し続けることができる
  - ユーザからは異常があったかどうかはわからない



高速化とは？

- 1つの仕事を複数のシステムが行う
  - クラスタリングをすることで各ノードの仕事量は約1/N
  - リクエストをロードバランサによって振り分ける方式
  - １ つの仕事を複数ノードで処理する方式



RDBMSのクラスタリング

- 冗長化
  - 共有ディスク方式
  - ミラーリング(レプリケーション)方式
- 負荷分散
  - リクエストの振り分け
  - リクエストの並列処理

注:PostgreSQL自体にクラスタリングの機能はないが、
クラスタリングするためのミドルウェアがいくつかある
(例:**pgpool-II**)

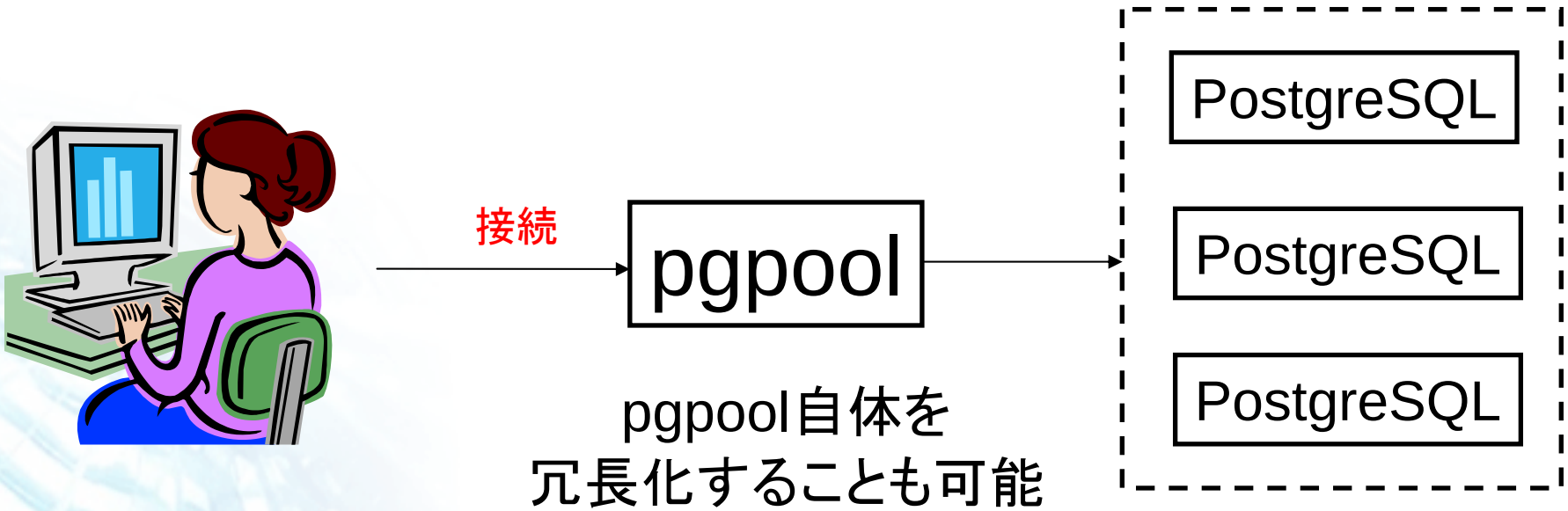
pgpool-IIの紹介

pgpool,pgpool-IIとは？

- 2003年開発スタート(pgpool)
  - 石井達夫氏が個人で開発
  - 最初はコネクションプーリング機能のみ(pgpoolという名前の由来はここからきている)
  - SRA OSS, Inc. 日本支社で機能拡張(pgpool-II)
- ライセンス
  - BSDライセンス
- 現在の開発体制
  - pgpool Global Development Groupという開発団体に移行

pgpool構成図

- クライアントとPostgreSQLの間に入る(proxy)

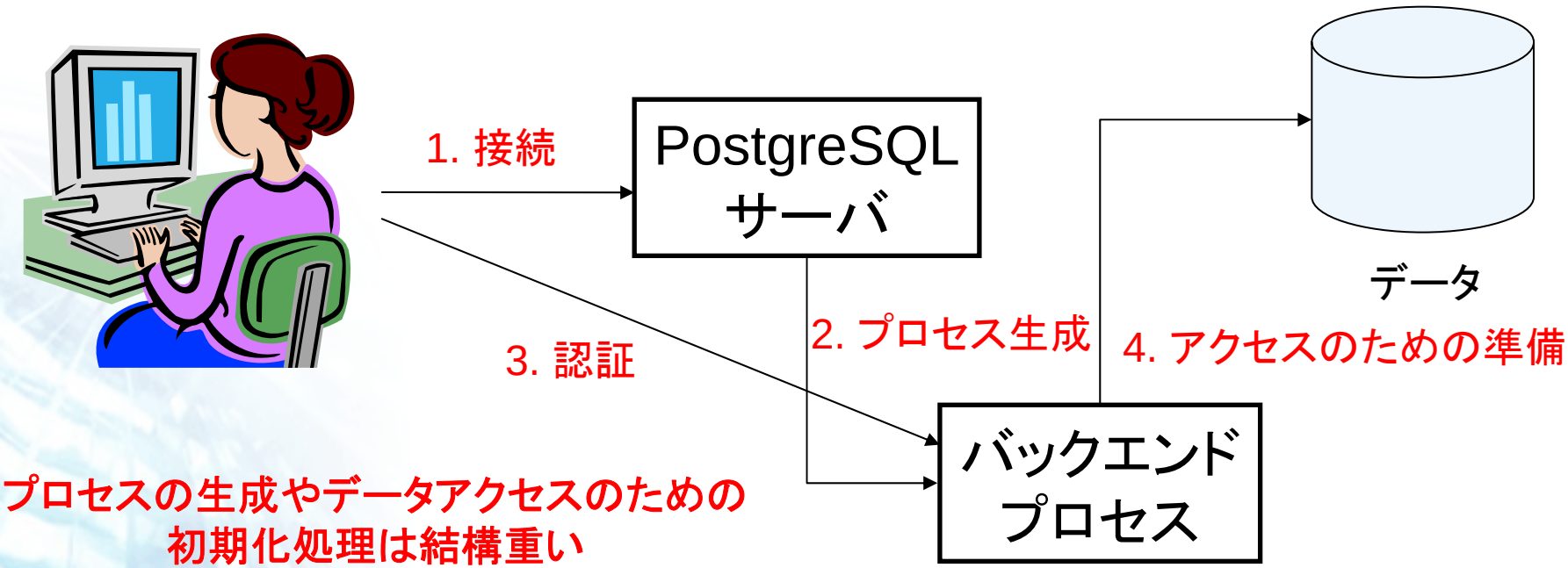


pgpool-IIの機能

1. コネクションプーリング
2. 同期レプリケーション
3. クエリの負荷分散
4. パラレルクエリ
5. 管理ツールによるpgpool-IIの管理

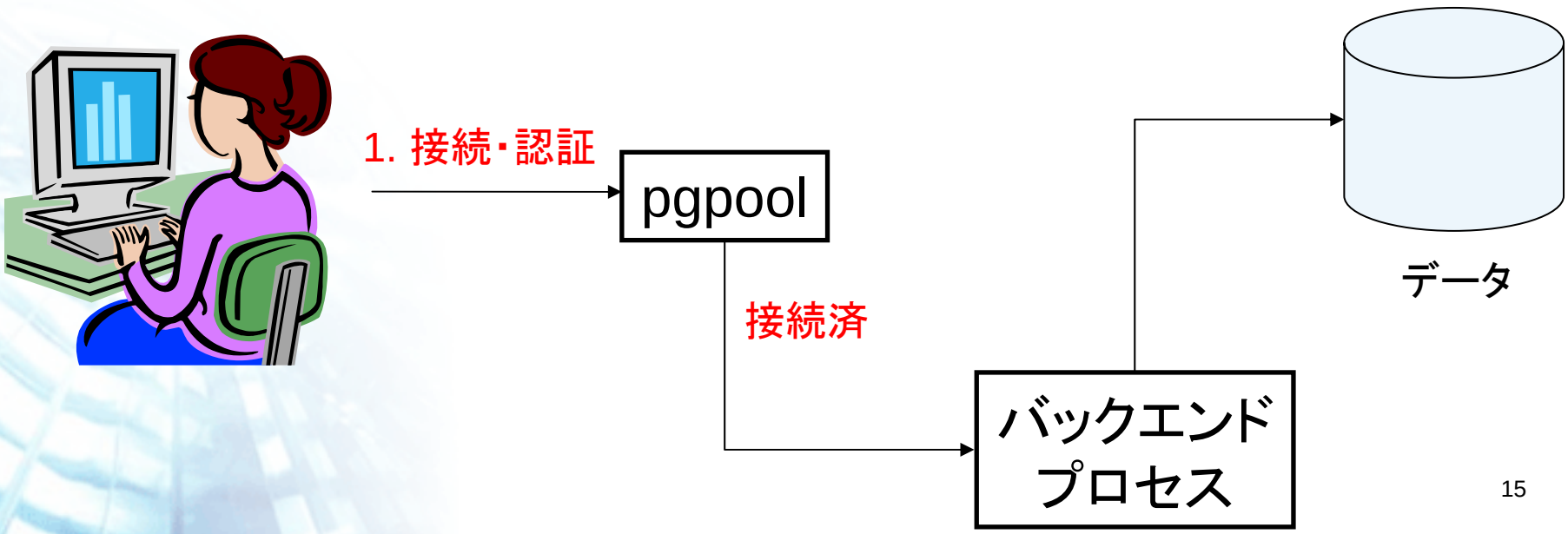
コネクションプーリングの利点

- 接続オーバーヘッドの軽減
  - PostgreSQLの接続処理の流れ



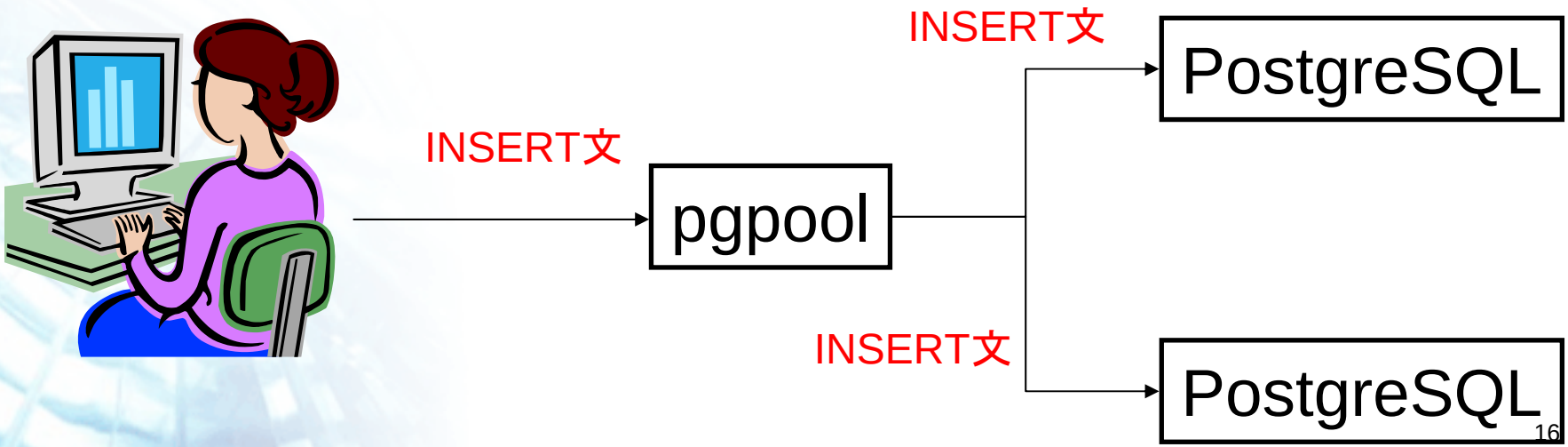
コネクションプーリングの利点

- 接続オーバーヘッドの軽減
  - pgpoolによるコネクションプールの接続処理の流れ
    - クライアントとPostgreSQLの間にpgpoolが入る
    - プロセス生成とディスクアクセスのための準備処理が省略可能



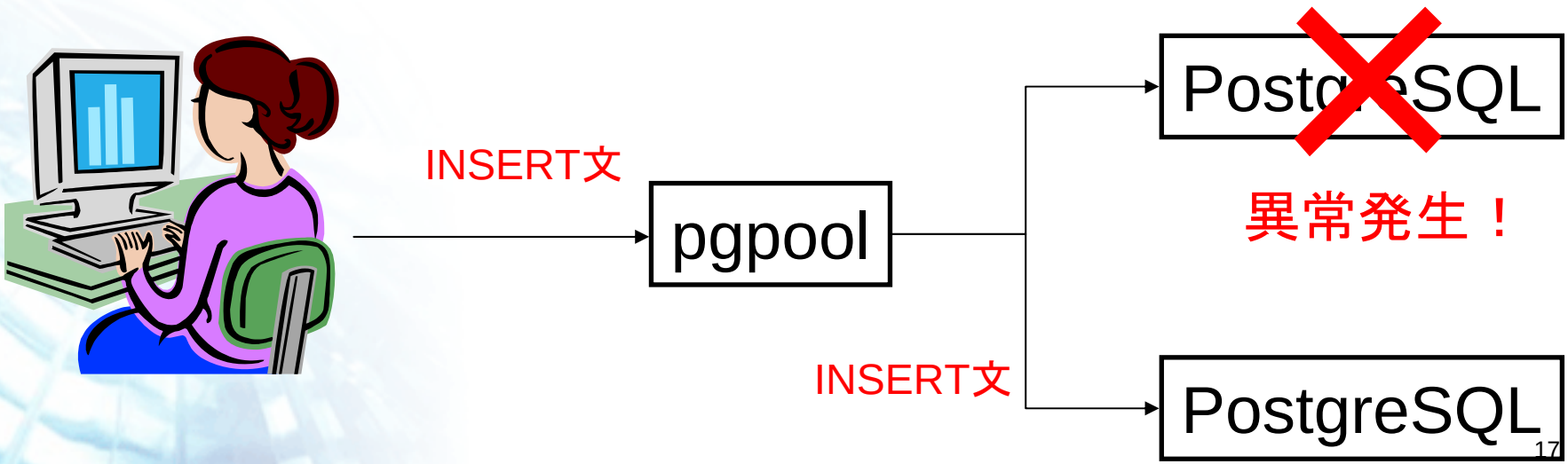
レプリケーションの利点

- 信頼性向上
  - 2台に更新系クエリを送信することで、データの同期を取る
    - 異常が発生した場合にも運用が可能(縮退運転)
    - 常に同期を取るため、通常より更新系クエリは時間がかかる



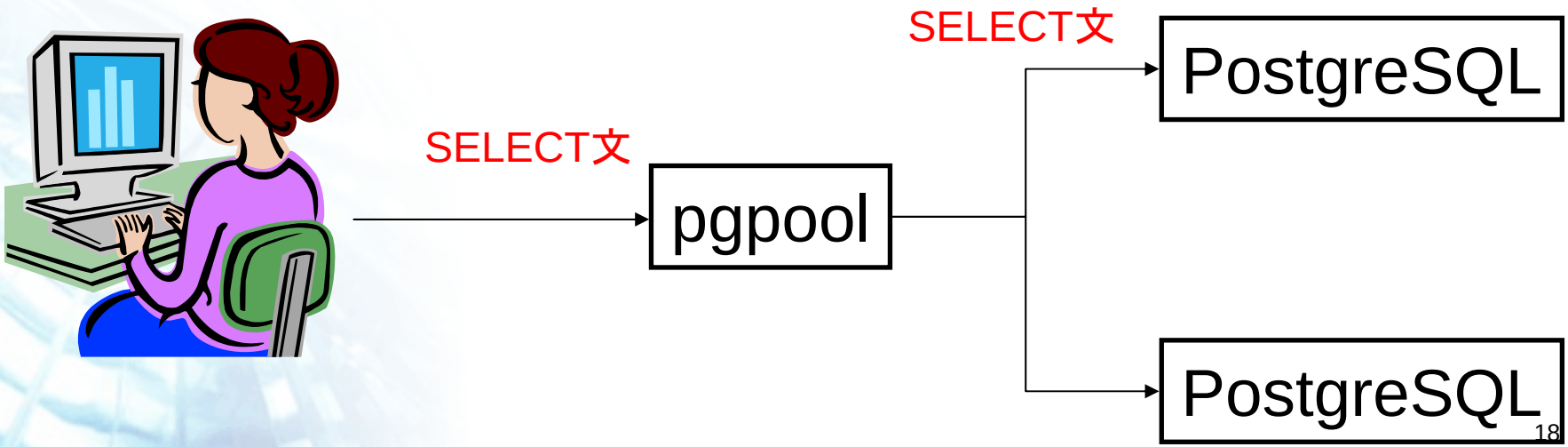
レプリケーションの利点

- 信頼性向上
  - 2台に更新系クエリを送信することで、データの同期を取る
    - 異常が発生した場合にも運用が可能(縮退運転)
    - 常に同期を取るため、通常より更新系クエリは時間がかかる



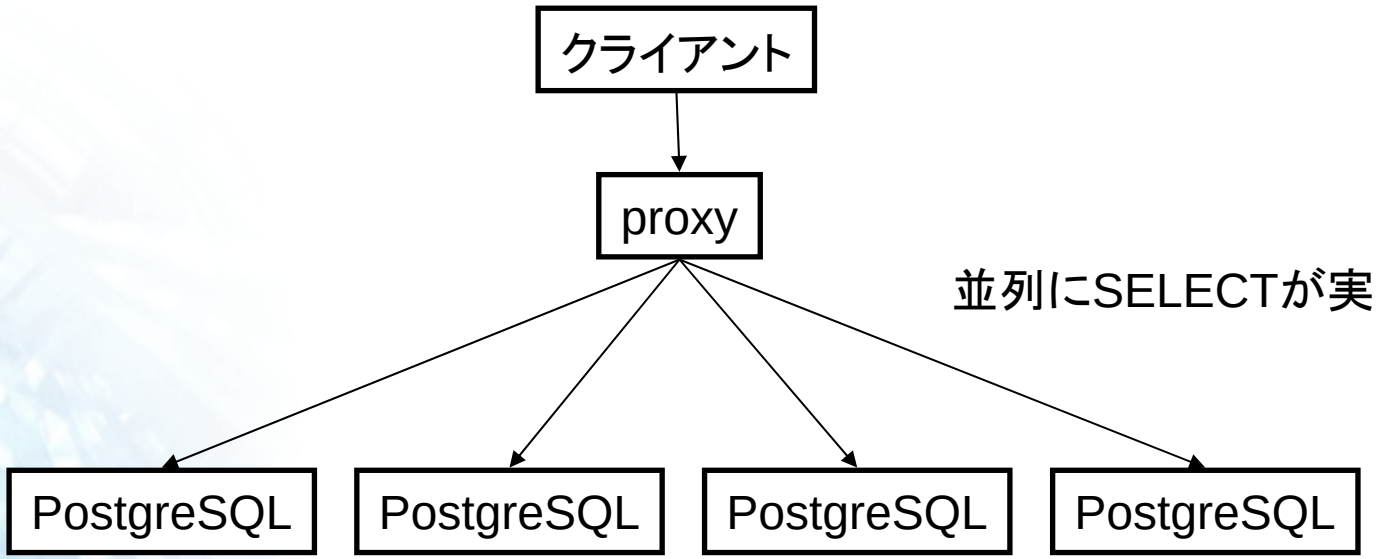
レプリケーションの利点

- ロードバランス
  - 同じデータを2台で持つので、検索についてはどちらかに問い合わせすればよいので、負荷分散が可能
  - ロードバランス先の割合を指定可能



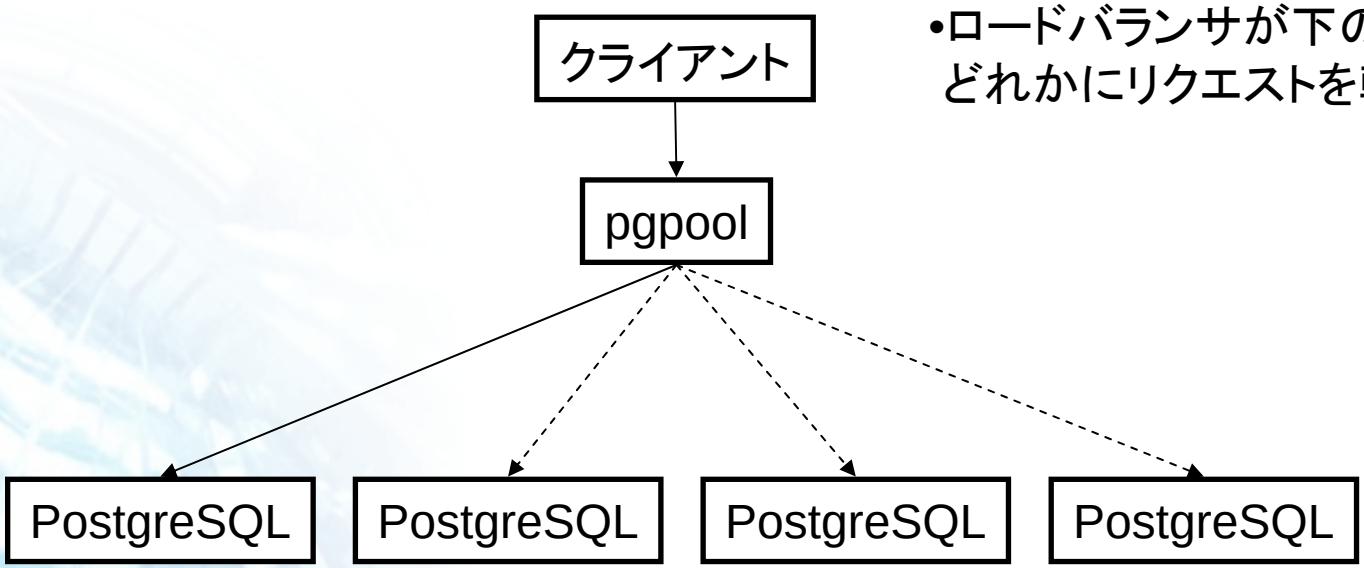
パラレルクエリとは？

- リクエストの並列処理
  - 複数のノードでリクエストを処理
    - 1つのテーブルを複数サーバに分割(テーブルパーティショニング)しておき、SELECTの結果をproxyでまとめることで、仮想的に1つのテーブルとして扱う



レプリケーション時の負荷分散 (パラレルクエリとの違い)

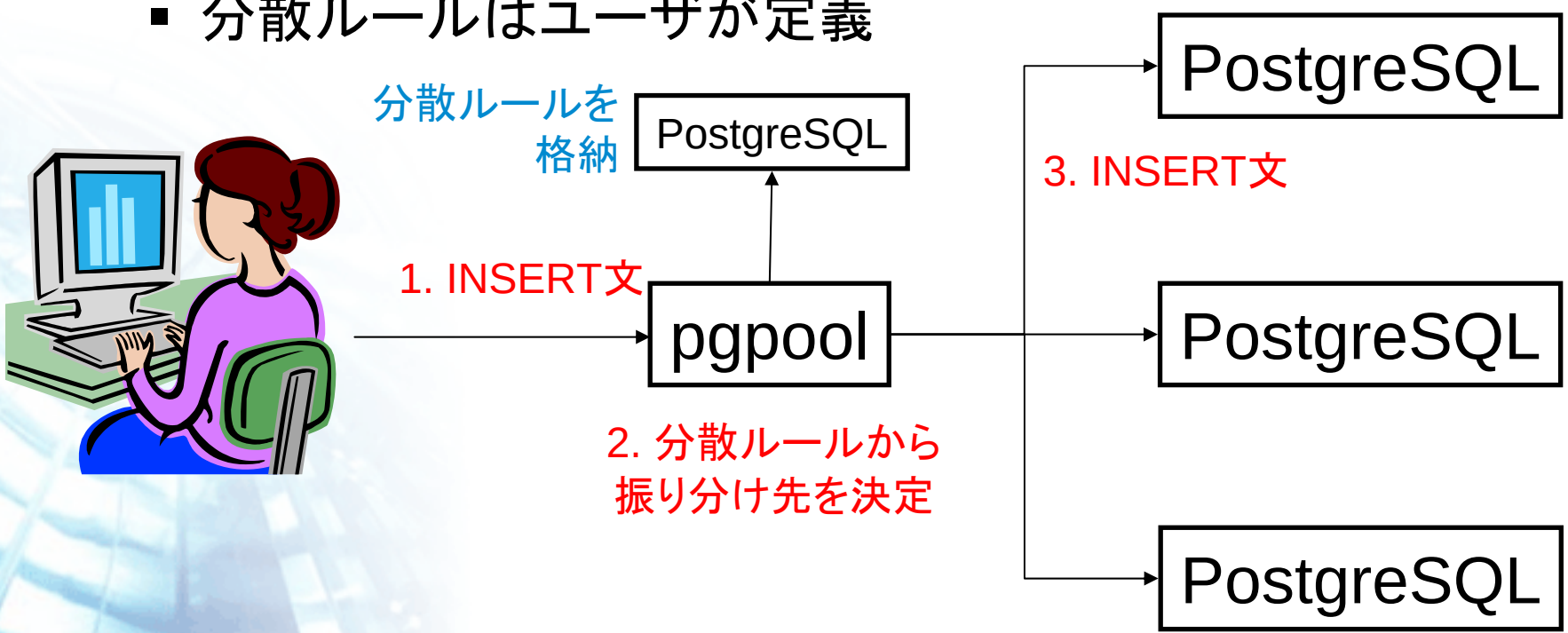
- リクエストの振り分け
  - ノードのどれかにリクエスト(RDBMSの場合はSELECT)を振る
  - 処理は振られたノード内で完結



•ロードバランサが下のPostgreSQLのどれかにリクエストを転送

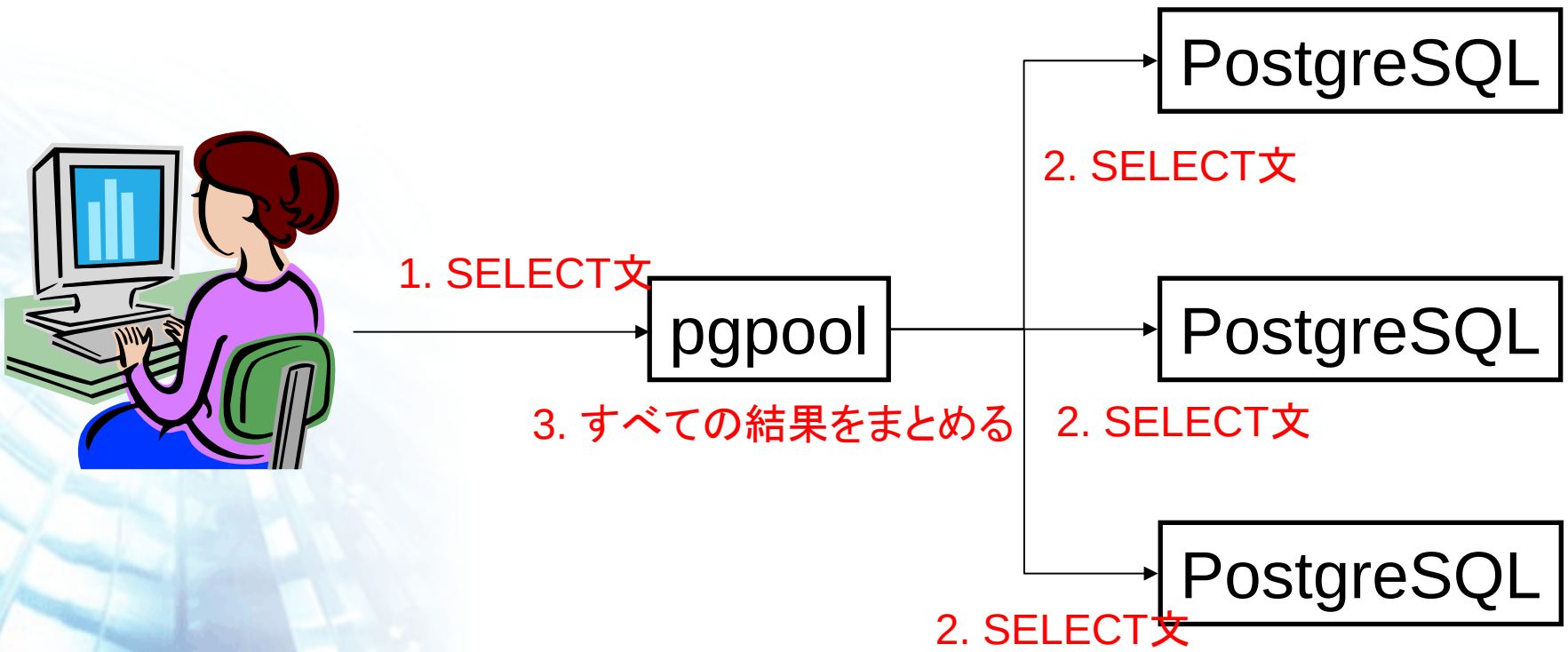
パラレルクエリの処理(INSERT)

- データを複数サーバに分散して配置
  - 複数のサーバに配置されたテーブルを仮想的に1つのテーブルとして扱う
  - 分散ルールはユーザが定義



パラレルクエリ(SELECT)

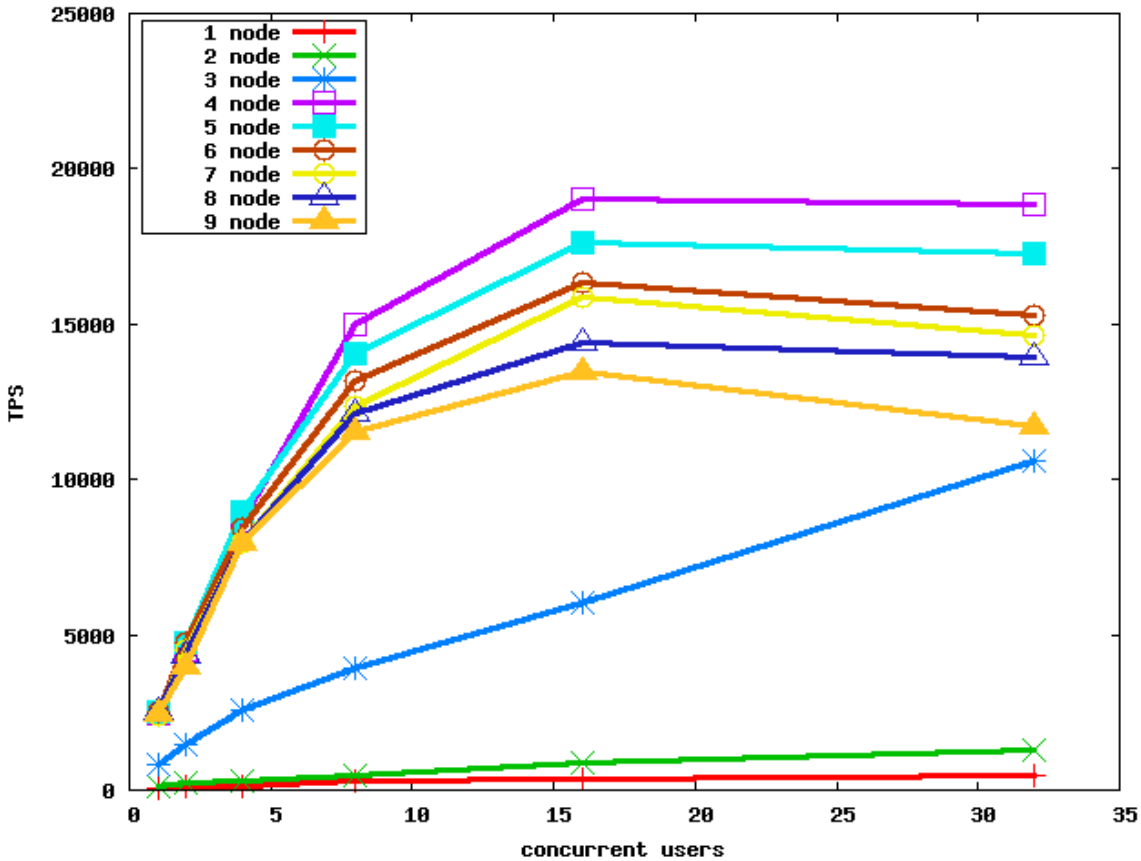
- クエリの並列処理
  - pgpoolが必要に応じてクエリを書き換え、ノードに送信
  - すべてのノードの結果をまとめて、クライアントに返す



パラレルクエリの利点・欠点

- 利点
  - 大容量のテーブルへの検索の高速化
    - PostgreSQL単体の場合、I/Oの負荷がすべて集中する
- 欠点
  - データ量が多くないとそれほど効果があがらない
    - pgpoolで検索結果の統合やクエリの書き換え処理などを実行するため、一台で十分なパフォーマンスが出ている場合はpgpoolが入ることによりオーバヘッドが増える

pgpool-IIのパラレルクエリの効果



- データ件数9000万件
- データはpgbenchで作成
- インデックス付検索

データ容量
テーブル：13GB

最大50倍
(4ノードの場合)

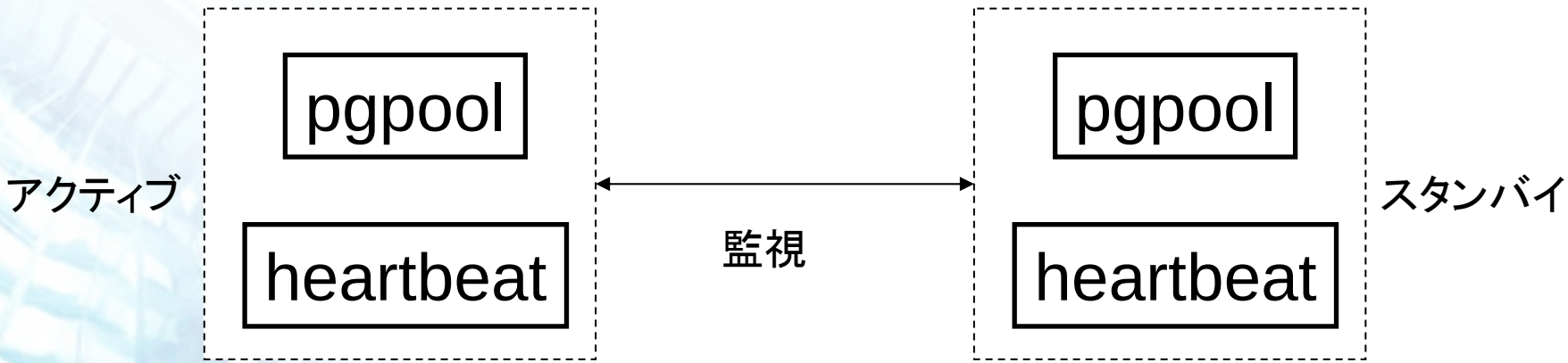
pgpool-II管理ツール

- PHPによるWebベースの管理ツール
  - 設定ファイルの編集
  - 統計情報の表示
  - ノード管理
    - ステータス表示
    - ノードの追加、切り離し


pgpoolAdmin
pgpool Administration Tool
pgpoolステータス
ノードステータス
クエリキャッシュ
分散ルール
pgpool.conf設定
管理ツール設定
パスワード変更
ログアウト
ヘルプ
pgpoolステータス
サマリー
プロセス情報
ノード情報
ノード情報
IPアドレス
ポート
ステータス
192.168.187.11
5432
ノード稼働中。接続有り
192.168.187.12
5432
ノード稼働中。接続有り
192.168.187.13
5432
ノード稼働中。接続有り
サマリー
プロセス情報
ノード情報
pgpool
pgpool停止
pgpool再起動
Version 1.0 Alpha
Copyright © 2006 pgpool Global Development Group. All rights reserved.

pgpool-HA

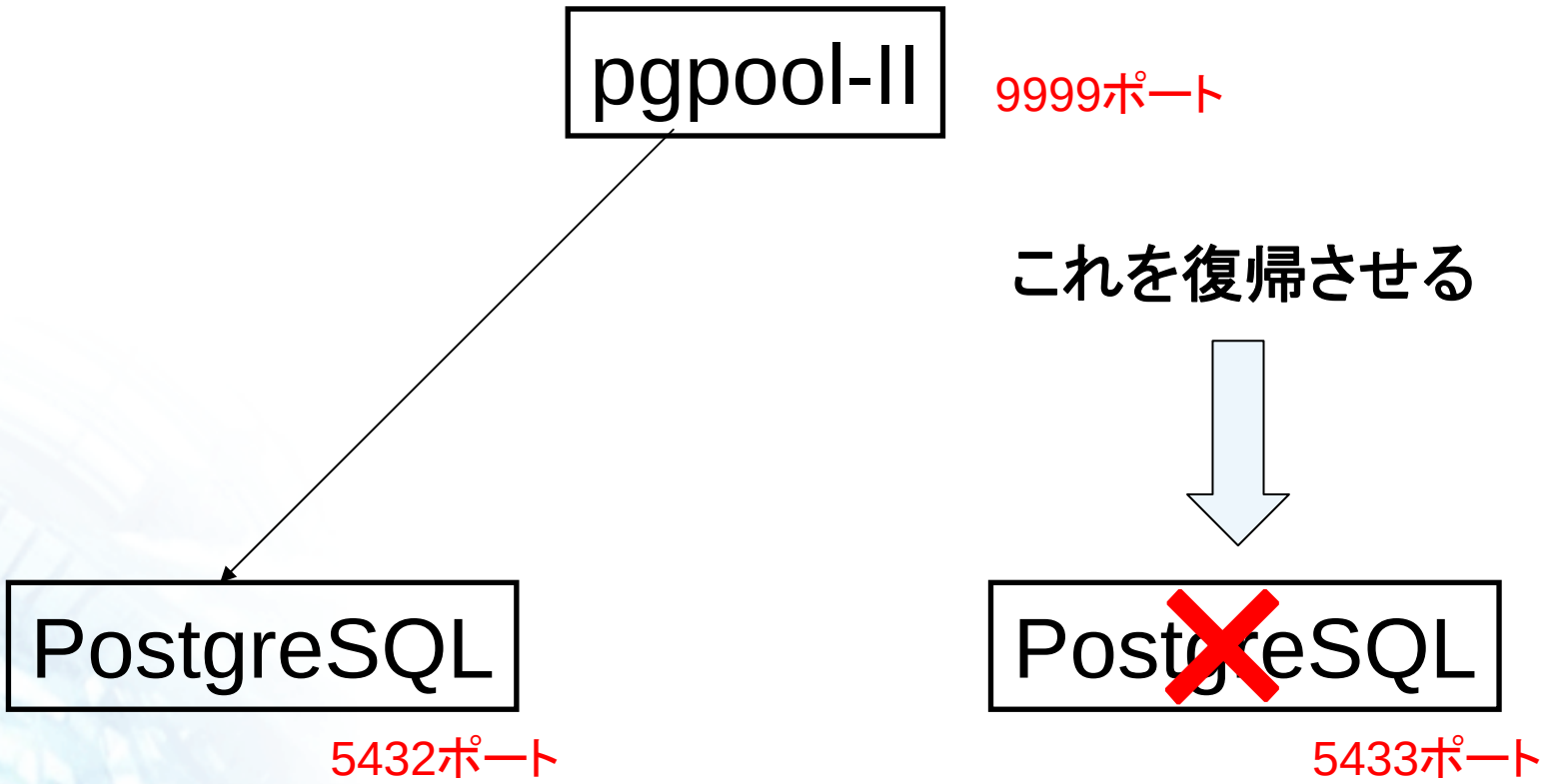
- pgpoolの弱点
  - pgpool自体が冗長化されていないため、pgpoolが落ちるとサービス停止(single point failure)
- heartbeatによるpgpoolの監視
  - heartbeatが仮想IPを割り当てるため、クライアントは接続先を仮想IPにしておけばよい



制限事項

- オンラインリカバリができない
  - CVSリポジトリにコミット済
- 一部のSQL(nowなど)には対応していない
- パラレルクエリはJDBCなどで使う問い合わせには今のところ対応できない(プリペアードステートメント等)

オンラインリカバリデモ



ロードマップ

- pgpool-II 2.0
  - 2007年10月リリースを予定
  - レプリケーションの高信頼化
  - パラレルクエリの高速化
  - オンラインリカバリ実装
  - その他制限事項の解除

pgpool-II参考URL

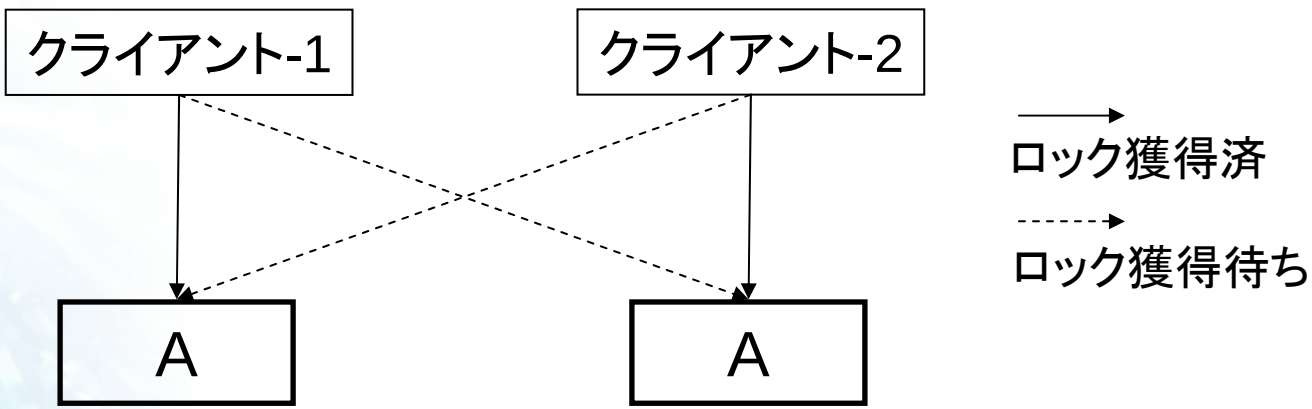
- pgpool,pgpool-II,pgpool-HA開発サイト
  - http://pgfoundry.org/projects/pgpool
- pgpoolコミュニティサイト
  - http://pgpool.sraoss.jp/

ご清聴ありがとうございました

(http://www.sraoss.co.jp/ に資料を置く予定です)

レプリケーション時の注意

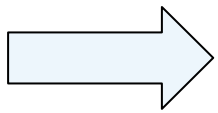
- ノードをまたいだデッドロック
  - 1つのノード内ではデッドロックが発生していないので、PostgreSQLでは検知不可
  - pgpoolでは設定(replication_strict = true)で回避
    - 片方のノードでロックを取れたのを確認してから、もう片方にクエリを送信



SERIAL型扱い時の注意

- INSERTにデフォルト値を使う時
  - INSERTの順序がマスタとセカンダリでずれると、挿入されるデータが異なってしまう可能性がある
  - pgpool.confのinsert_lock＝trueにするか、/*INSERT LOCK*/というコメントをINSERTの前に付ける

```
INSERT INTO meibo(id, name)
VALUES (DEFAULT, 'y-asaba')
```



```
BEGIN;
LOCK TABLE meibo;
INSERT INTO meibo(id, name)
VALUES (DEFAULT, 'y-asaba');
COMMIT;
```