

# ***pgpool-II* で実現する PostgreSQLの高可用性**

SRA OSS, Inc. 日本支社

技術部 盛 宣陽

y-mori@sraoss.co.jp

# SRA OSS, Inc. 日本支社

- **2005年設立**

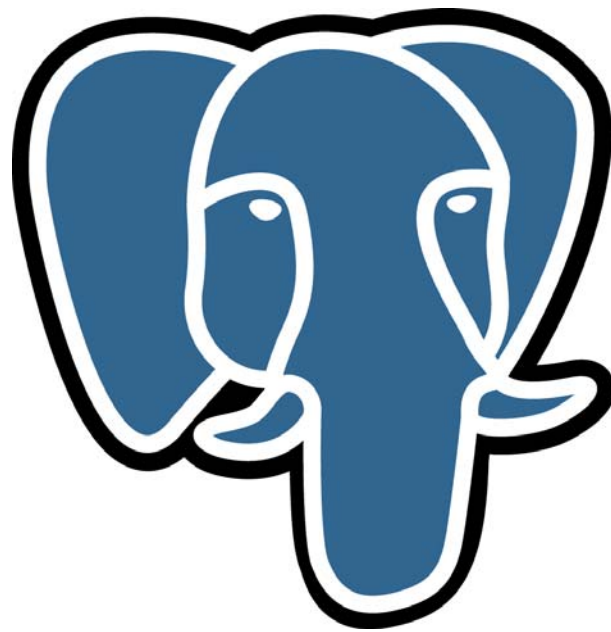
- SRA OSS Inc. (米国カリフォルニア) に本社
- 以前は SRA の一部署 (オープンソースカンパニー)

- **オープンソースソフトウェアビジネスを展開**

- PostgreSQL
- Sylpheed
- Linux
- gcc
- パッケージ製品「PowerGres」
- Tomcat
- PHP
- その他 OSS

# まず PostgreSQL とは？

- **オープンソース RDBMS**
- **高機能**
  - トランザクション
  - 多言語対応
  - 全文検索
  - その他多数
- **導入事例は数え切れない**



**誕生は10年以上前！  
祖先は Ingres (1970年代)**

# pgpool-II 開発背景

|     | PostgreSQL への要求    | シングルサーバの限界   |
|-----|--------------------|--|
| 可用性 | 限りなくダウンタイムをゼロにしたい！ | ハードウェアの故障による可用性の低下   |
| 性能  | もっと速く！             | リソースの物理的な限界 <ul style="list-style-type: none"> <li>• CPU スケールアップ</li> <li>• 搭載メモリの限界</li> <li>• I/O の集中</li> </ul> |

**1 PostgreSQL、1 サーバでは超えることのできない壁**

これを解決するために・・・

**pgpool-II** を開発

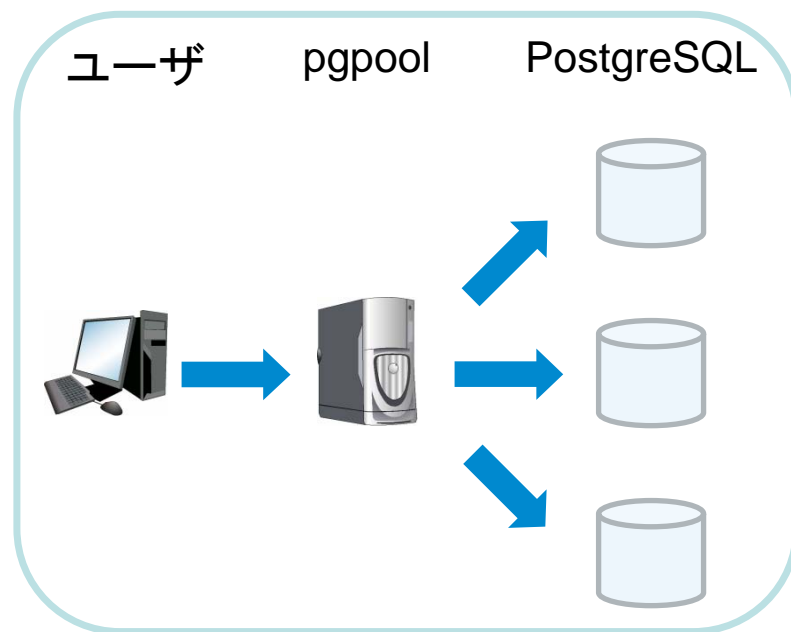
**可用性と性能の両方を向上させる  
オープンソースミドルウェア！！**

# pgpool-II の特徴

- ユーザと複数台の PostgreSQL の間に pgpool-II を設置することで、ユーザからは1台のデータベースに見える

## pgpool-II の歴史

|       |  |
|-------|--|
| 2004年 | pgpool 初期版リリース                             |
| 2006年 | IPA の支援により SRA OSS で pgpool-II 開発          |
| 2007年 | SRA OSS で開発を継続し pgpool-II v2.0 リリース        |
| 現在    | pgpool Global Development Group で開発&メンテナンス |



**BSDライセンスでソースコードを公開**

# pgpool-II の機能

## pgpool-II の主な提供機能

- 可用性
  - 同期レプリケーション
    - オンラインリカバリ
  - Slony-Iとの連携
  - ログ SHIPPING
- 性能
  - 参照クエリの負荷分散
  - パラレルクエリ
  - クエリキャッシュ
  - コネクションプール

## 運用管理

- Webベース管理ツール
- 独自の管理プロトコルの提供

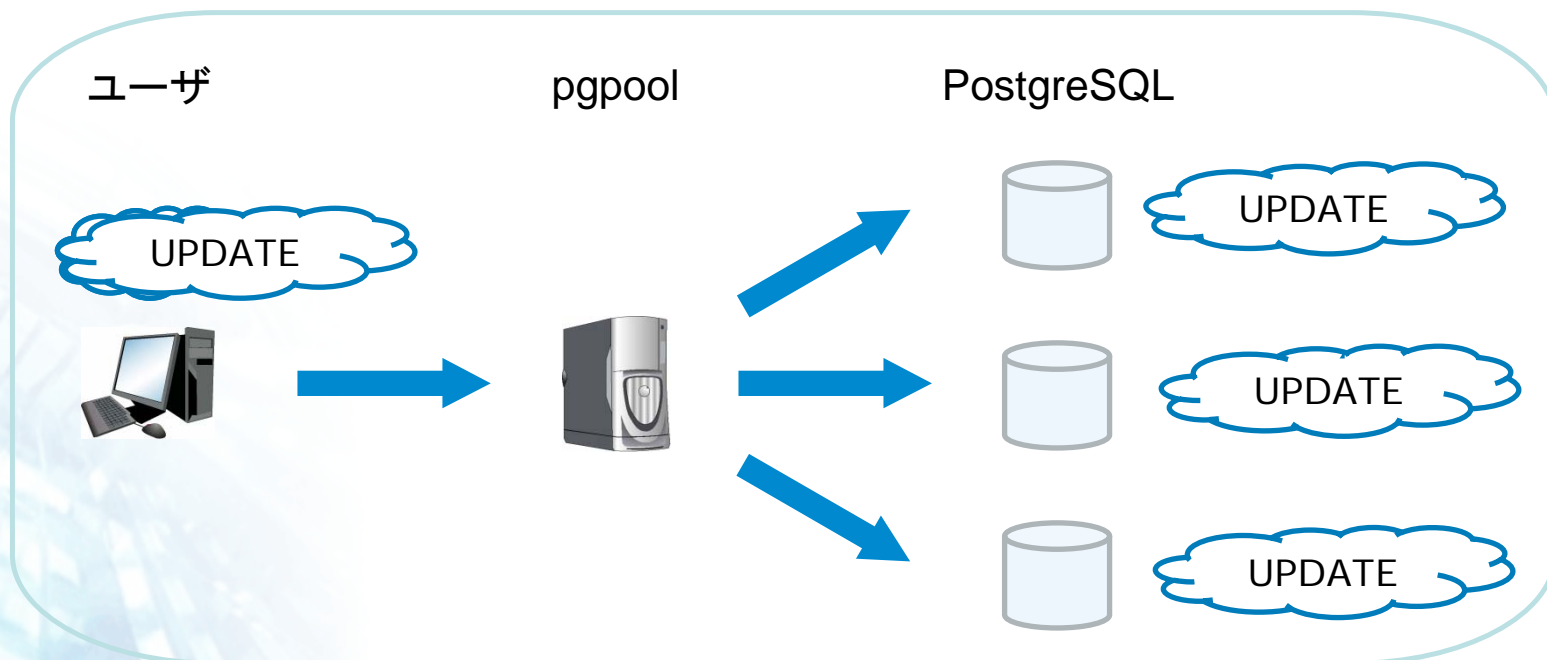


可用性機能 其の一

# 同期レプリケーション

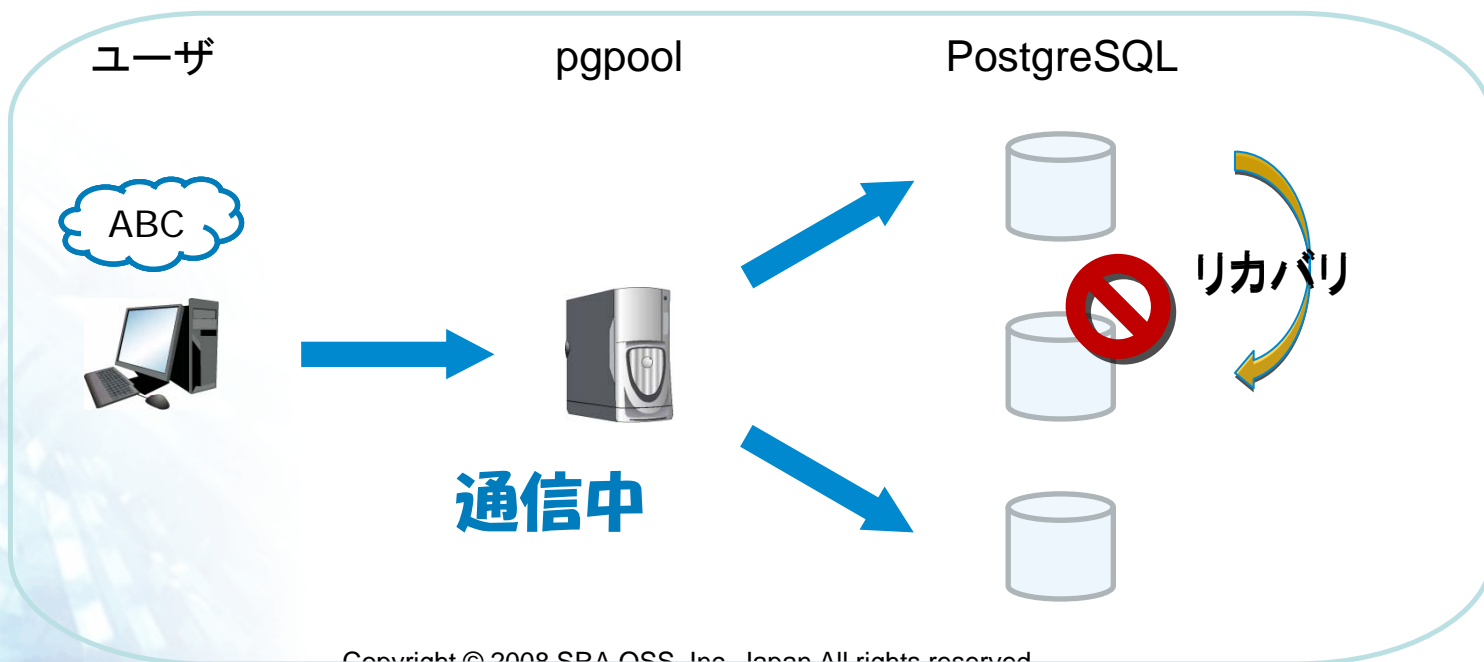
# 同期レプリケーション

- 更新クエリを複数サーバに送信し同期を行う
- 障害が発生した場合にはフェイルオーバーを行い運用を継続
- 1行の更新性能は約1/2



# 同期レプリケーション オンラインリカバリ

- サービスを停止することなくノード復旧が可能
  - 新規ノードの追加も可能
  - 無停止でアップデートも可能
- 洗練されたログベースによるリカバリ
- 障害が発生してから、オンラインリカバリの設定ができる!
- 管理ツールからボタン一発操作!



# 同期レプリケーション 更新処理について

- 更新処理はマスターで行ロックを取る!
- 処理の結果(何件更新したか)によってデータ整合性チェックが行われる

NODE1 (MASTER)

NODE2

NODE3

BEGIN;

BEGIN;

BEGIN;

UPDATE; Masterで、最初に行ロックを獲得

UPDATE;

UPDATE; Master以外は  
並列にクエリを送信

COMMIT;

COMMIT;

COMMIT;

## 同期レプリケーション 注意が必要なクエリ

- サブクエリを伴う更新クエリには注意!

NODE1 (MASTER)

```
BEGIN;
```

```
UPDATE t1 ..  
  where t1.a in (select t2.a from t2);
```

NODE2

```
BEGIN;
```

```
UPDATE t1 ..  
  where t1.a in (select t2.a from t2);
```

## 同期レプリケーション 注意が必要なクエリ

- サブクエリを伴う更新クエリには注意!

NODE1 (MASTER)

BEGIN;

UPDATE t1 ..  
where t1.a in (select t2.a from t2);

INSERT INTO t2 ..

UPDATEの処理中に、他のユーザ  
がINSERTされる可能性がある!

NODE2

BEGIN;

INSERT INTO t2 ..

UPDATE t1 ..  
where t1.a in (select t2.a from t2);

ノード間でUPDATEの実行結果の件  
数が異なる場合には、トランザクショ  
ンがアボートされる!

## 同期レプリケーション 注意が必要なクエリ

- サブクエリを伴う更新クエリには注意!

### NODE1 (MASTER)

```
BEGIN;
```

```
LOCK TABLE t2  
  IN SHARE ROW EXCLUSIVE MODE;
```

```
UPDATE t1 ..  
  where t1.a in (select t2.a from t2);
```

テーブルがロックされているため、他のユーザはINSERT INTO t2を実行できない

### NODE2

```
BEGIN;
```

```
LOCK TABLE t2  
  IN SHARE ROW EXCLUSIVE MODE;
```

```
UPDATE t1 ..  
  where t1.a in (select t2.a from t2);
```

## 同期レプリケーション その他 注意が必要なクエリ

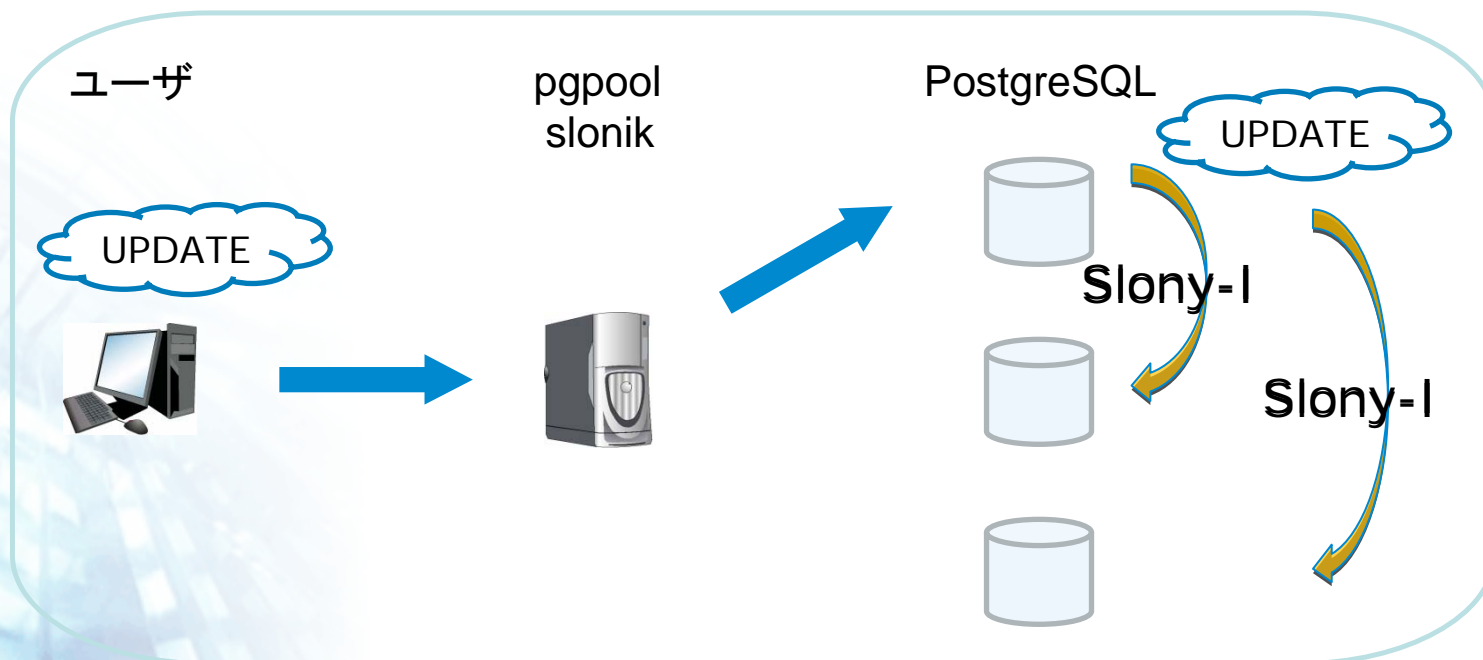
- 同じテーブルに対して、  
UPDATEとINSERTが行われる場合  
かつ  
新規に追加されるINSERT文がUPDATEの対象に  
含まれる場合
- INSERT INTO.. SELECT構文やVALUE句の中の  
SELECT
- ユーザ定義関数内で、SELECTの実行結果を使って更新処  
理を行う処理

## 可用性機能 其の二

# Slony-I との連携

## Slony-I との連携

- Slony-Iによる非同期レプリケーション
- 更新クエリはマスタのみに送信
- 自動フェイルオーバー+ Slony-Iのマスタの切り替え
- 新規ノードの追加も可能
- テーブルの追加などスキーマの変更ができない
- 非同期レプリケーションのため、フェイルオーバー時にデータをロスする可能性がある

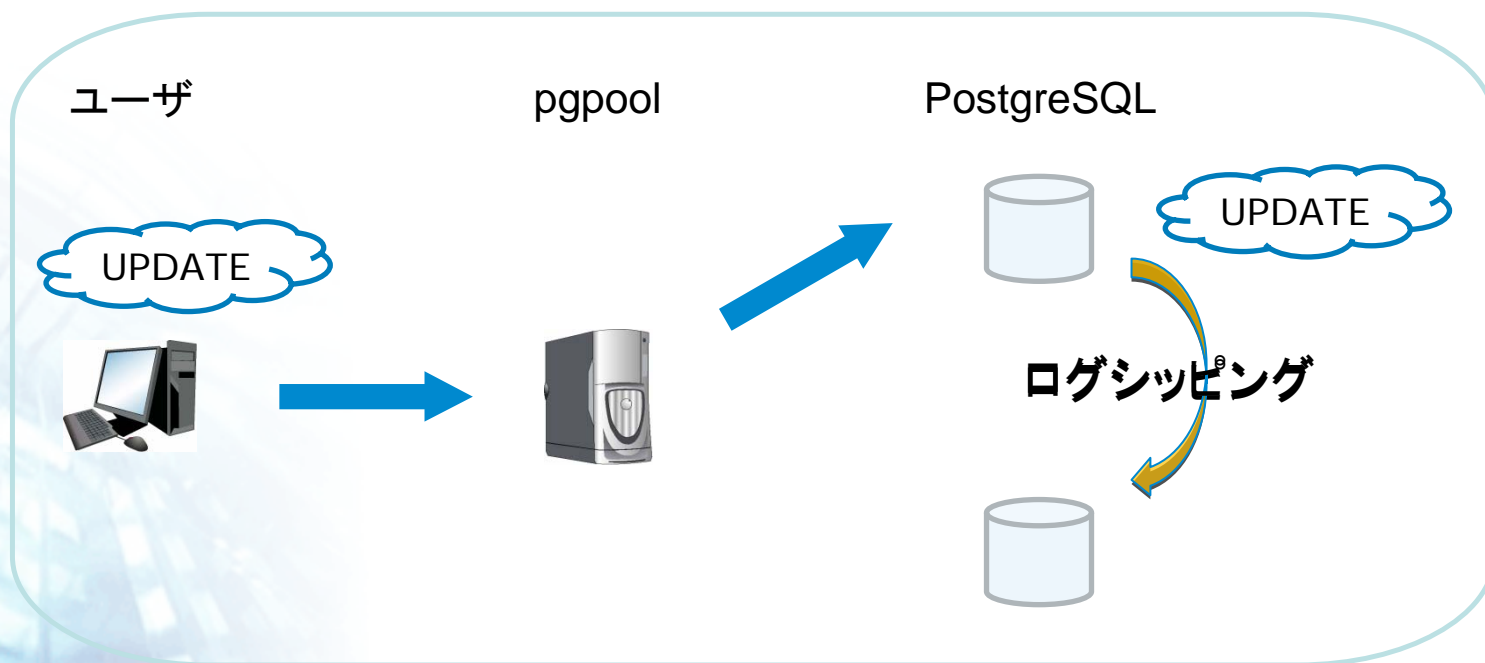


## 可用性機能 其三

# ログシッピング

## ログ SHIPPING

- PostgreSQLのPITRによるログ SHIPPINGを行う
- 更新クエリはマスタのみに送信
- 障害が発生した場合には、自動フェイルオーバー処理を実行
- フェイルオーバー後に、サーバを一台追加してバックアップサーバの構築も可能
- 障害発生時にWAL領域のデータがロストする



## 可用性 まとめ

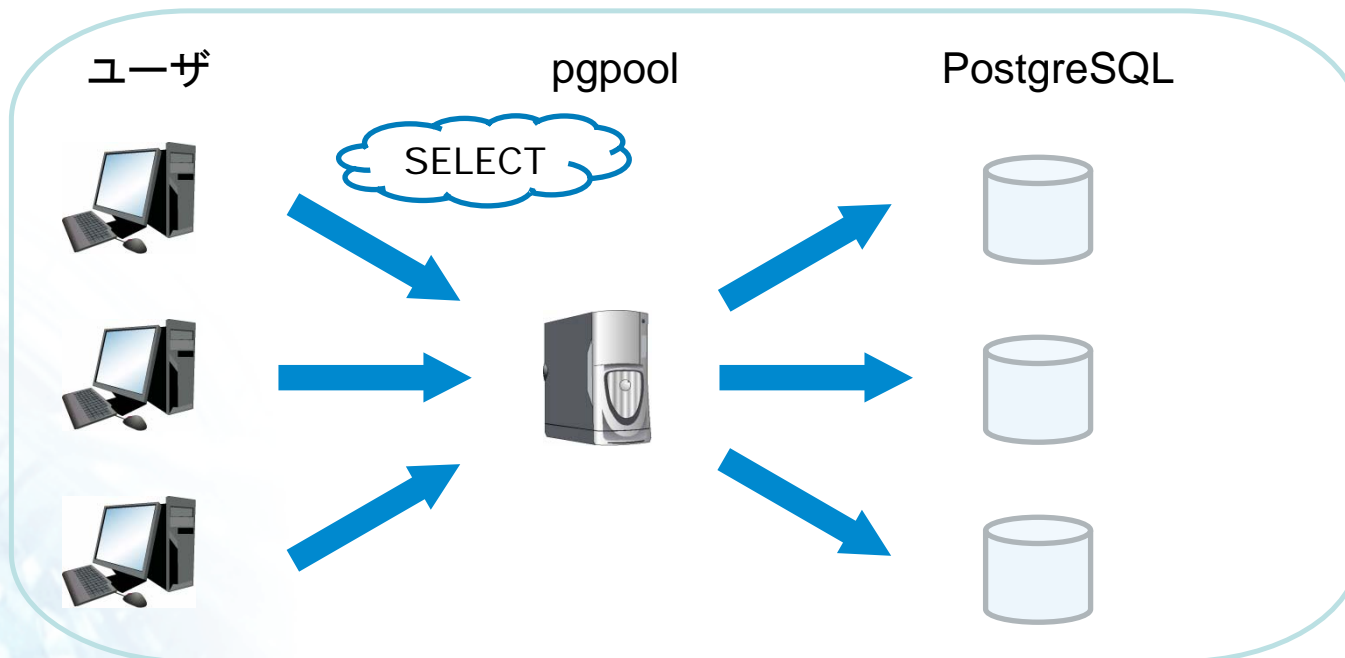
- 同期レプリケーション
  - メリット
    - 同期レプリケーション
  - デメリット
    - クエリを選び既存のシステムからの移行はアプリケーションの修正が必要
- Slony-Iとの連携
  - メリット
    - クエリの自由度は高い
  - デメリット
    - 非同期レプリケーション
    - フェイルオーバー時にデータロストの可能性がある。
    - Slony-Iの操作がやや困難
- ログ SHIPPING
  - メリット
    - PostgreSQLの機能のみを利用したレプリケーションシステム
    - クエリは何でも使える
  - デメリット
    - フェイルオーバー時にデータロストする可能性がある。
    - ログのメンテナンスが大変

**性能機能 其の一**

**負荷分散**

# 負荷分散

- 同期レプリケーション、Slony-Iとの連携を行っている場合のみ有効
- 検索クエリは重みをつけてランダムに振り分ける
- 負荷を PostgreSQL 間で分かち合うので検索性能が向上



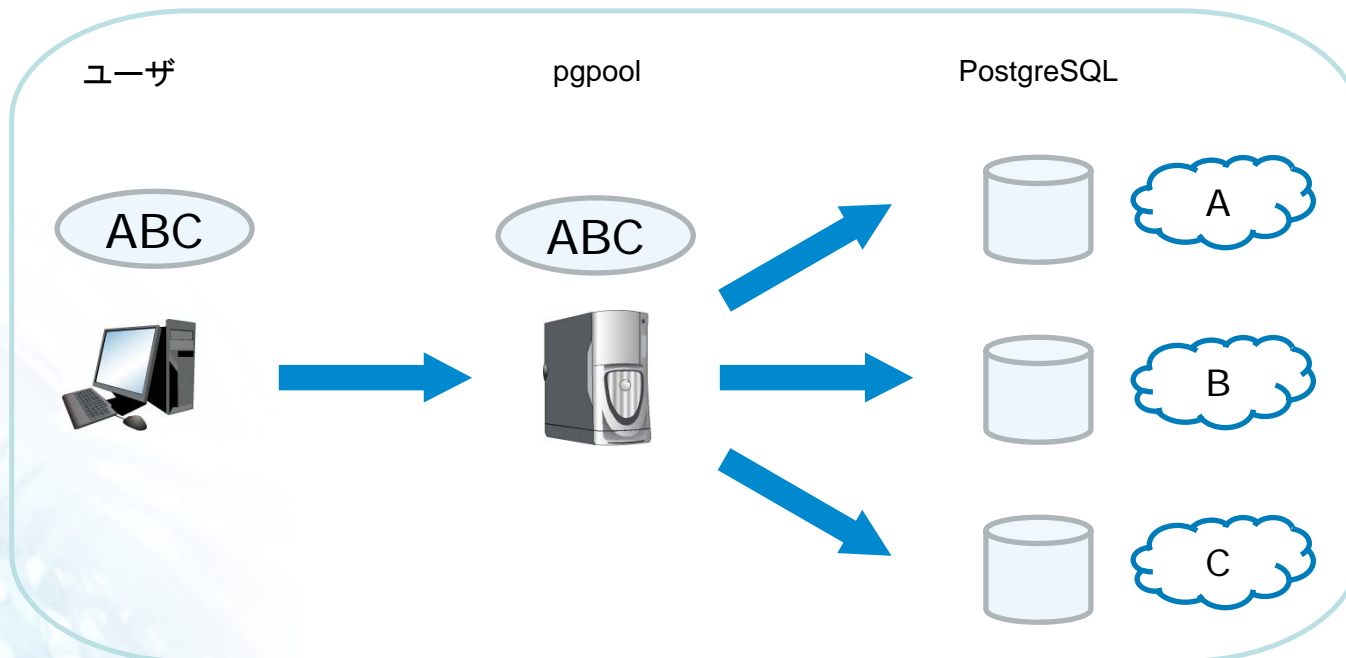
Web システムなどに最適

## 性能機能 其の二

# パラレルクエリ

# パラレルクエリ

- データを分割してPostgreSQL間で分担
- 問い合わせを並列に処理するので高速
- 検索クエリを一斉に実行して結果を pgpool-II でまとめる



会計システムなどの意思決定システムなどに最適

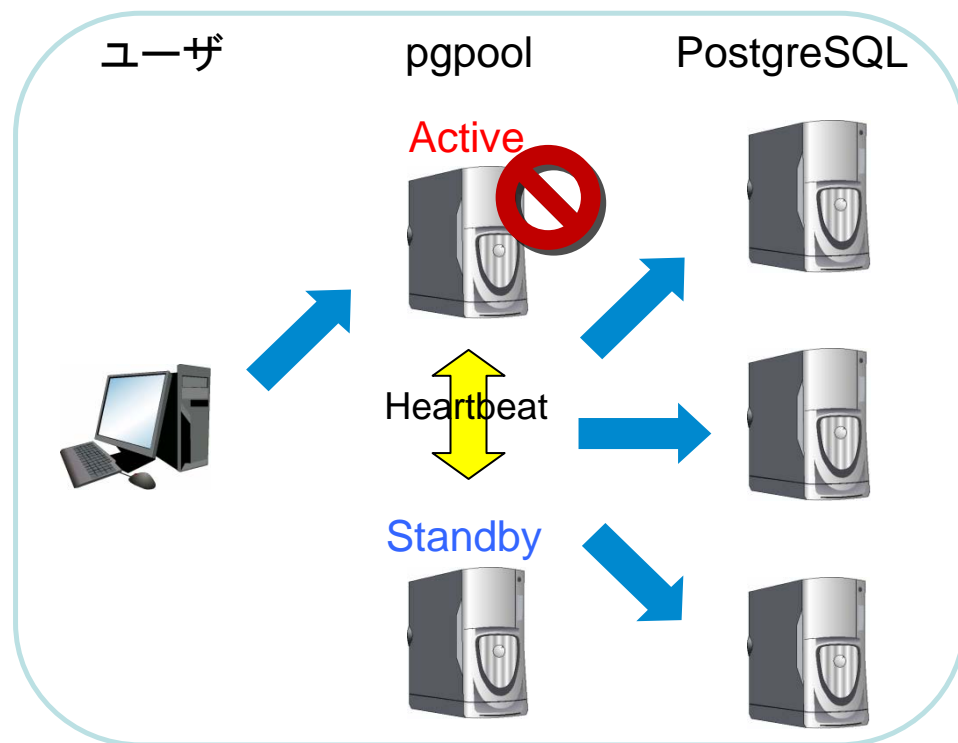
# その他の機能

## その他の機能

- コネクションプーリング（性能）
- クエリキャッシュ（性能）
- 障害時に指定したコマンドを実行（運用管理）
  - 管理者へのメール送信など

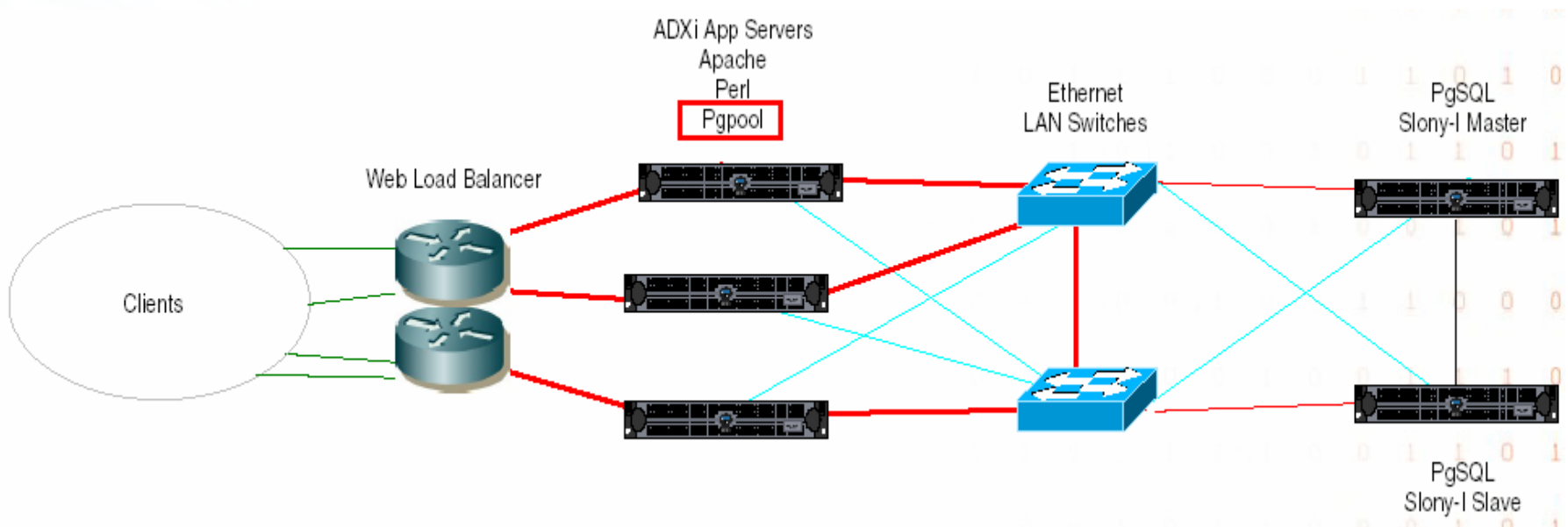
# pgpool-HA

- オープンソースの HeartBeat と組み合わせてさらなる可用性
- Active-Standby 構成で pgpool の死活監視
- 仮想 IP を利用することでクライアントはサーバが切り替わったことを意識する必要がない
- 現状ではオンラインリカバリはできない



# pgpool 利用事例（1）

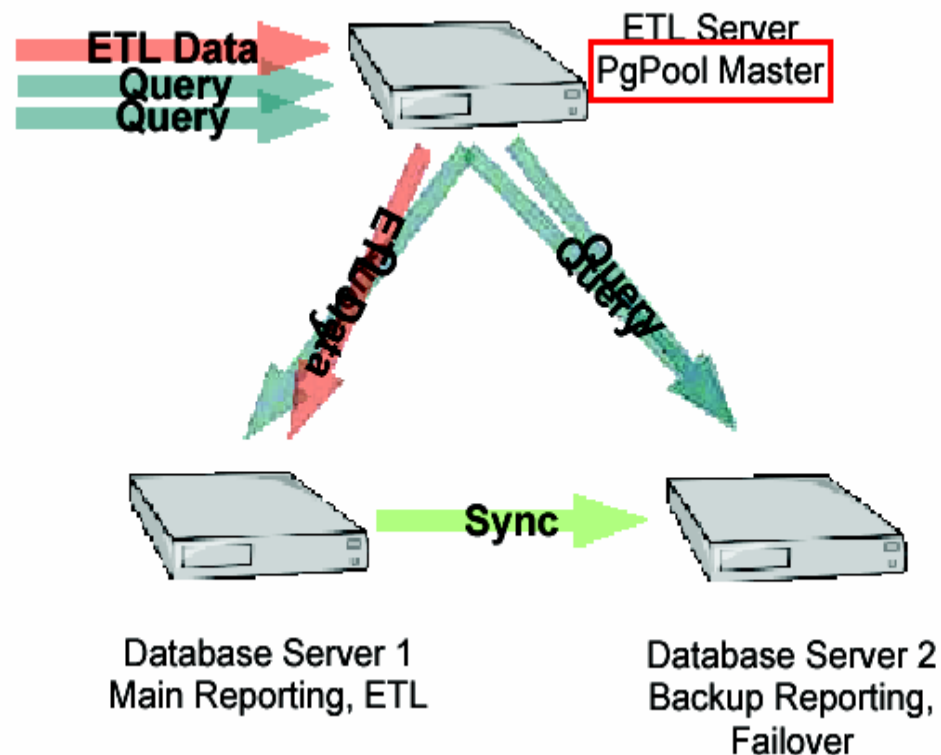
- 自動車産業のデータ交換システム
  - <http://www.postgresql.org/communityfiles/14.pdf>
  - クライアントはGM, Ford, 本田, 日産などの自動車メーカー
  - データベースの負荷分散に pgpool を利用



## pgpool 利用事例 (2)

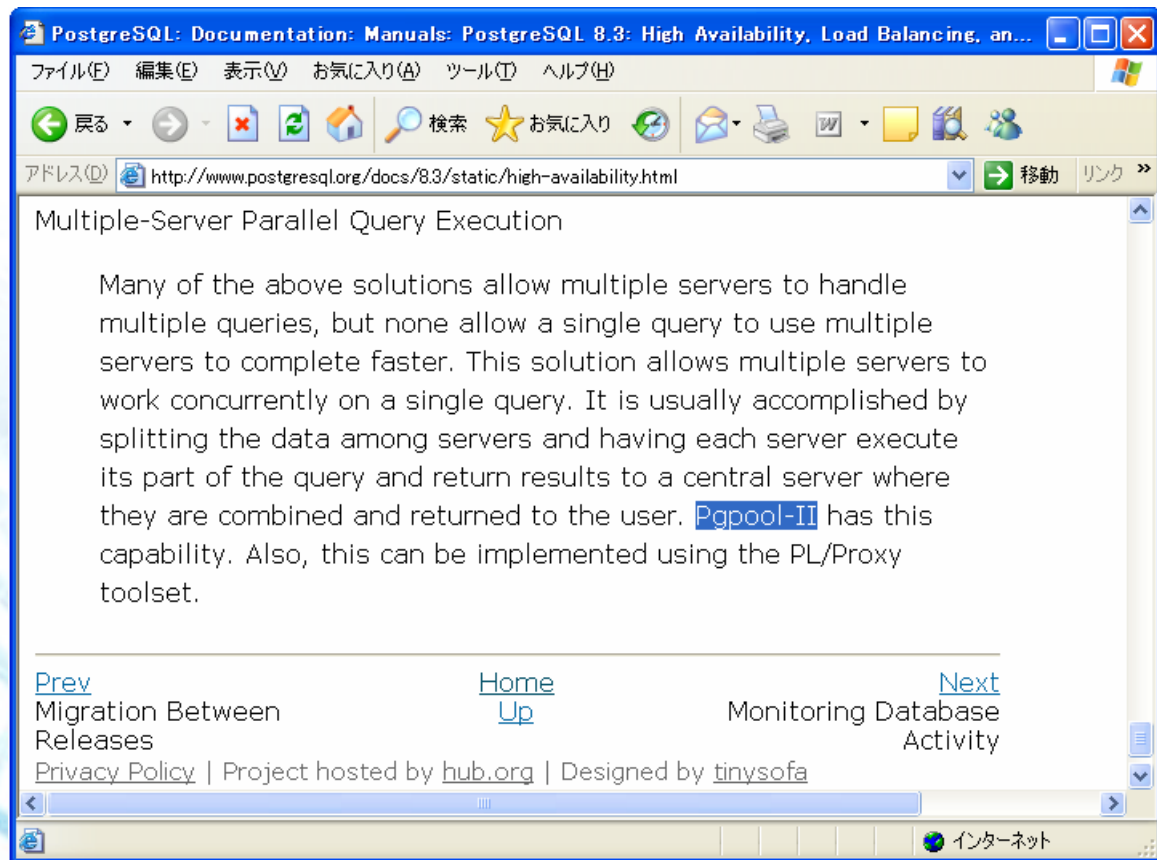
- Weblog 解析システム

- [http://joeconway.com/terabytes\\_osc2005.pdf](http://joeconway.com/terabytes_osc2005.pdf)
  - 520GB のデータを解析
  - pgpool のレプリケーション機能を利用



# PostgreSQL 公式マニュアルでの紹介

- Chapter 25.  
High Availability, Load Balancing, and Replication

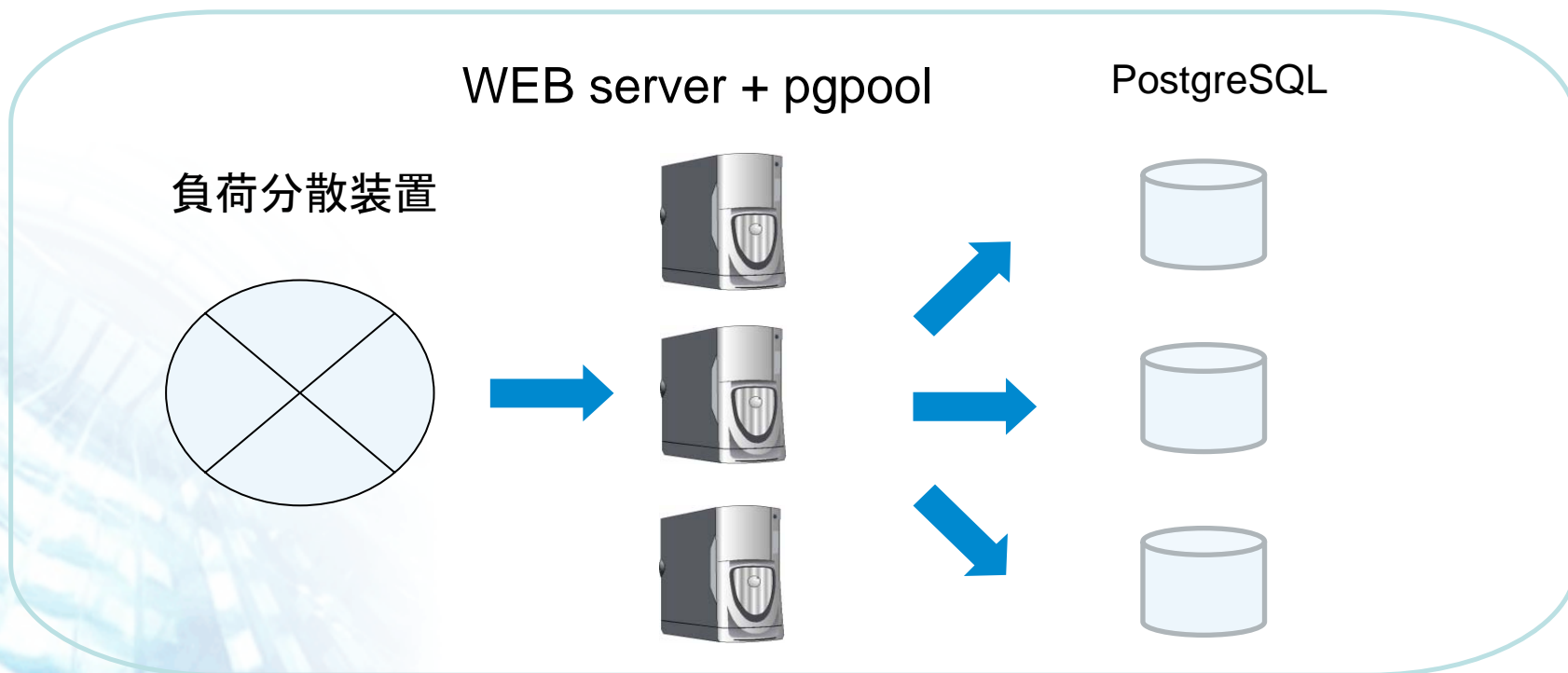


和訳:  
複数サーバ並列クエリ処理

上記のソリューション(レプリケーションソフト)は複数のクエリを複数のサーバで負荷分散することができるが、1つのクエリを複数のサーバで並列に処理することができない。これはデータを分割してそれぞれのサーバでクエリの一部を処理した結果を返す必要がある。[pgpool-II](#)ではこれが可能である。...

## 今後の予定

- オンラインリカバリも含めpgpool-II複数台構成のサポート



ご清聴ありがとうございました