PowerGres

PostgreSQLにおける高可用性実現方法

SRA OSS, Inc. 日本支社

SRA OSS,INC.

可用性とは？

- システムの信頼度を評価する観点の一つ
  - 信頼性(Reliability)
    - MTBF(Mean Time Between Failure)
  - 可用性(Availability)
    - 稼働率
  - 保守性(Serviceability)
    - MTTR(Mean Time To Repair)
  - 保全性(Integrity)
  - 安全性(Security)

·高可用性 = High Availability (HA)

PowerGres

- 可用性の尺度
  - 稼働率 = MTBF/(MTBF+MTTR)
  - 「可用性が高い = 高可用性」とは，稼働率が100%に近いこと
  - 稼働率を上げるためには，MTTRを小さくする

年間あたりの故障時間	稼働率
3日	99.2%
1日	99.7%
1時間	99.98%
5分	99.99996%

システムの可用性が高いことによるメリット

PowerGres

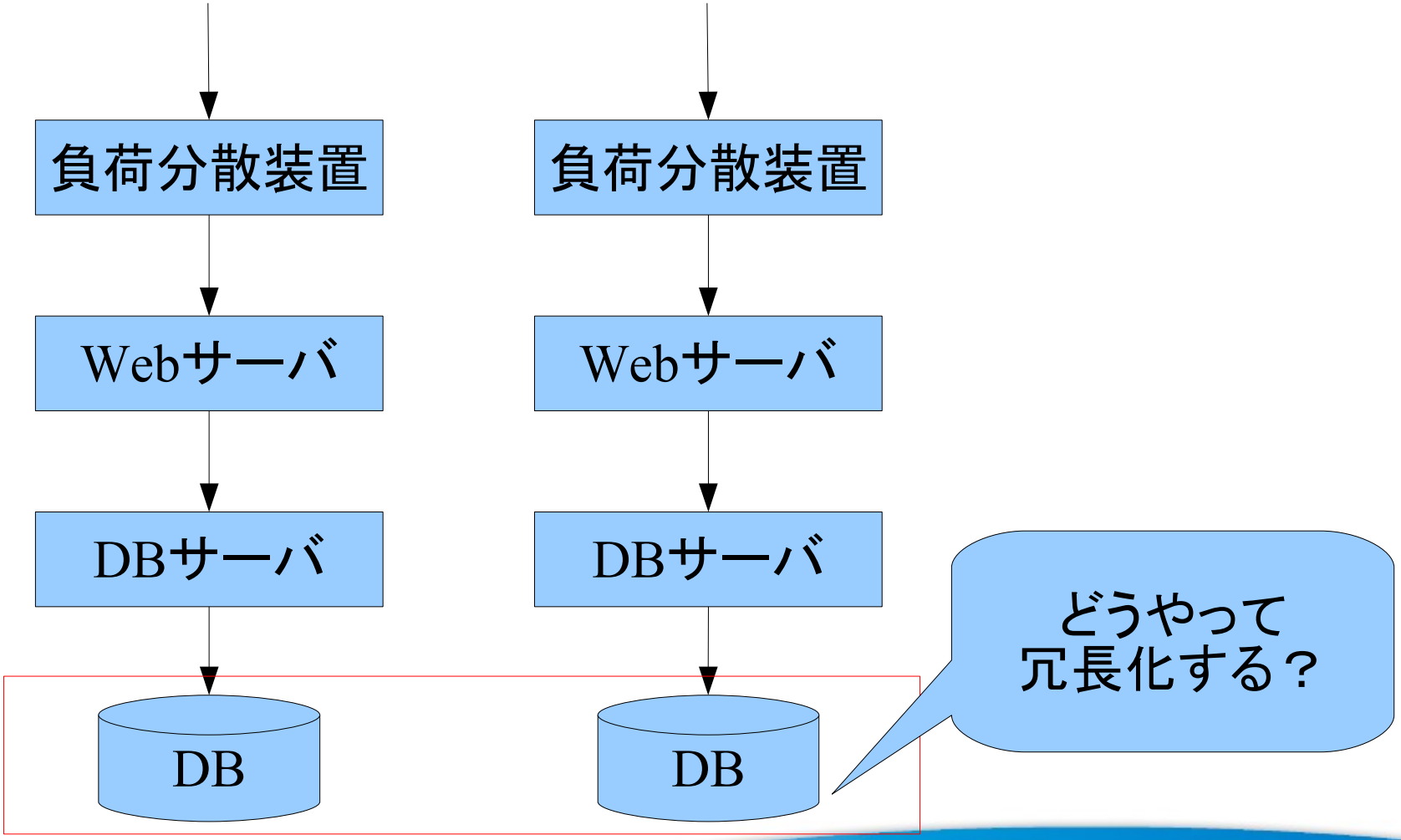
- ビジネスの機会損失の防止
  - インターネット経由での無人販売
- 顧客離れの防止
  - ダウンしているサイトには2度と行かない
- 顧客からの信頼感
  - 管理や技術力の高いことの証し
- 保守費用の低減
  - 緊急対応要員不要
- データ保全

高可用性戦略

PowerGres

- バックアップとリストア
  - 可用性は低いがローコスト
- 基本は冗長化
  - ハードウェアの冗長化
  - ソフトウェアの冗長化
- SPF(Single Point of Failure)をなくす
  - 稼働率は，システム構成要素の中のもっとも弱い部分に足を引張られる
  - ただし，故障するか/しないかで評価するのは無意味．稼働率が問題

DBサーバ冗長化の難しさ



1) バックアップ/リストア方式

PowerGres

- PostgreSQLの機能だけで実現可能
- ローコスト
- バックアップした時点までしか戻れない
- リストアに時間がかかるので，稼働率は下がる
  - データ容量にもよるが，30分から数時間は見ておく必要がある
    - 参考: OSS iPedia(http://ossipedia.ipa.go.jp/)
- 代替用のハードウェアリソースが必要
  - 用意しておかないと悲惨なことに...
  - 代替用ハードウェアリソースは有効活用できない

1-2)PITRによるバックアップ/リストア

- ファイルシステムバックアップ(tar, rsyncなど)＋アーカイブログによる差分バックアップ
- 最新の状態までリカバリ出来る(任意の時点までリカバリすることも可能)
- ログを保存しておく必要がある
- 操作が煩雑

PITR STEP 1

- 事前準備
  - GUIよりオンラインバックアップを有効にする
  - バックアップファイル格納場所の指定
  - postgresの再起動

PITR STEP 2

- 事前準備
  - ベースバックアップの取得

PITR STEP 3

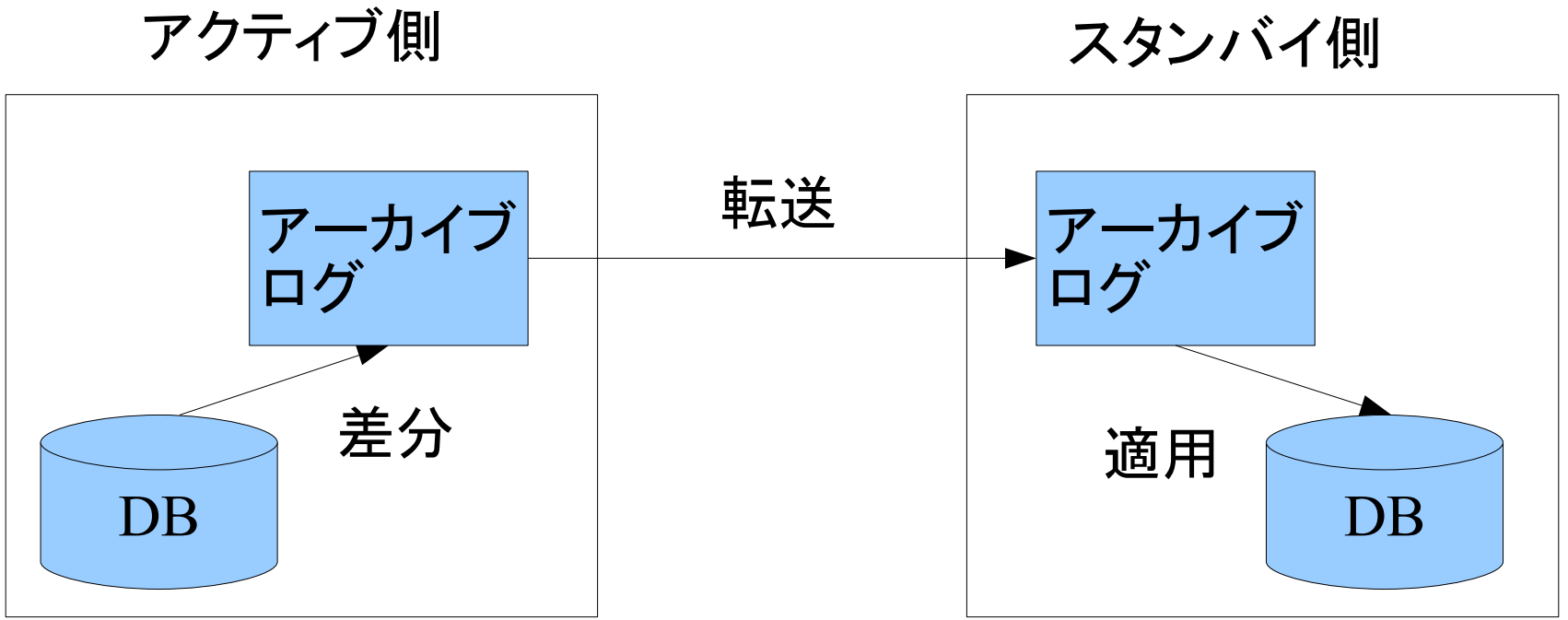
- リカバリの実行
  - 最新の状態までリカバリするか日時を指定するか選択する

2) ウォームスタンバイ

PowerGres

- 2台のマシンと，PITRを使う
- アクティブ側から，スタンバイ側に連続してアーカイブログを転送する
- スタンバイ側は，リカバリをし続ける

ウォームスタンバイのシステムイメージ



ウォームスタンバイの評価

PowerGres

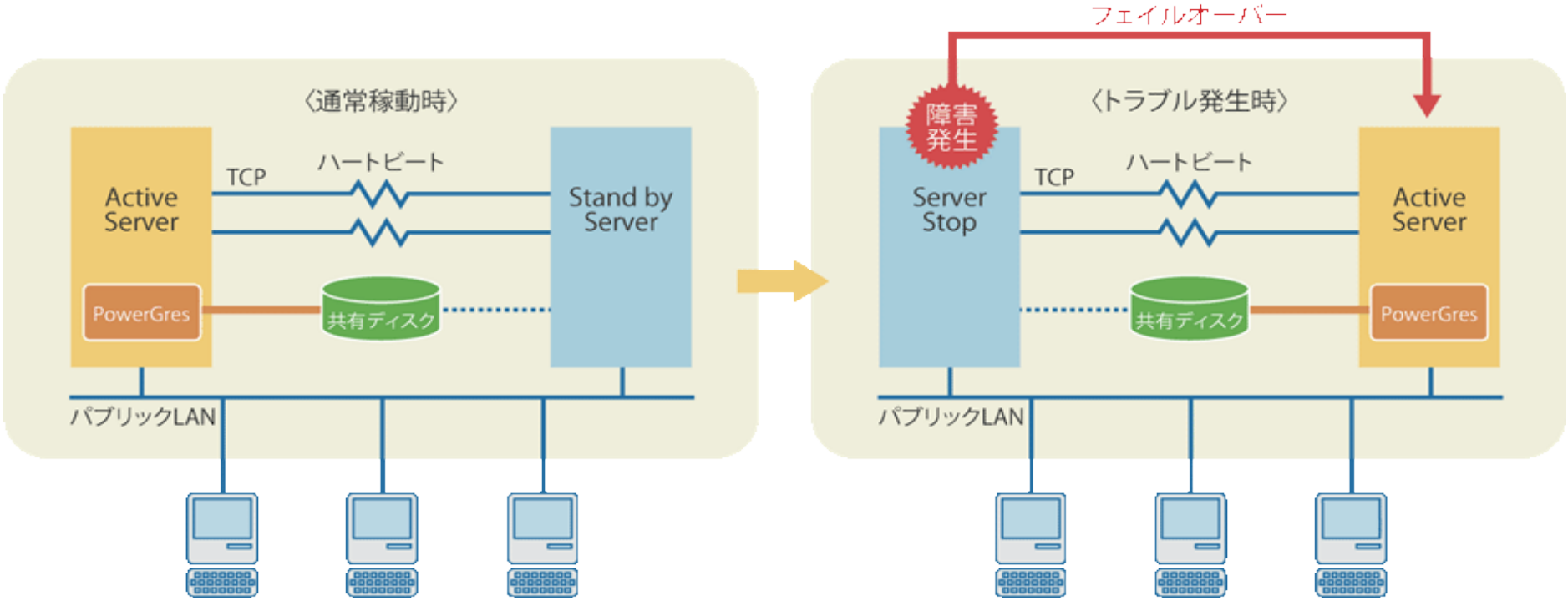
- メリット
  - アプリケーションやDB設定に変更不要
  - DBの性能にあまり影響を与えない
  - コストが安い
- デメリット
  - スタンバイ側のDBにはアクセスできない
  - 転送されなかったログの分のデータは失われる
  - 設定が煩雑
  - アクティブ側がダウンしたことの検知と，IPの切り替え，ウォームスタンバイ側のオンライン状態への移行などは作り込みの必要あり

3) 共有ディスク型HAシステム

PowerGres

- HA監視ソフト＋2台のDBサーバ
- 2台のシステムがお互いにディスクを共有することにより，DBデータのコピーの必要性をなくす

共有ディスク型HAシステムのシステムイメージ



共有ディスク型HAシステムの評価

PowerGres

- メリット
  - アプリケーションやDB設定に変更不要，性能にも影響なし
  - 稼働率が高い(ダウンタイム数分)
  - アクティブ/アクティブ構成にすれば待機側のリソースも有効活用
  - 実績がある
- デメリット
  - 共有ディスク装置が必要なので，コストが高い
  - 共有ディスク装置が故障するとデータが失われる

4) ハードウェアによる高可用性実現

- フォールトトレラント(無停止)構成
  - Stratus ftServer 6200
- CPU，メモリ，ディスク，電源など，すべてがハード的に二重化
- ソフトからは1台のコンピュータに見える

ハードウェアによる高可用性システムの評価

PowerGres

- メリット
  - ソフトウェアの互換性が高い
    - Red Hat Linux, PostgreSQL, PowerGresでSRAOSSで性能評価済
  - フェイルオーバがないので極めて可用性が高い(ソフトからは停止時間がないように見える)
- デメリット
  - 専用ハードウェアが必要
  - ソフト障害には対応できない
  - 負荷分散やパラレルクエリなどの性能向上を提供するものではない

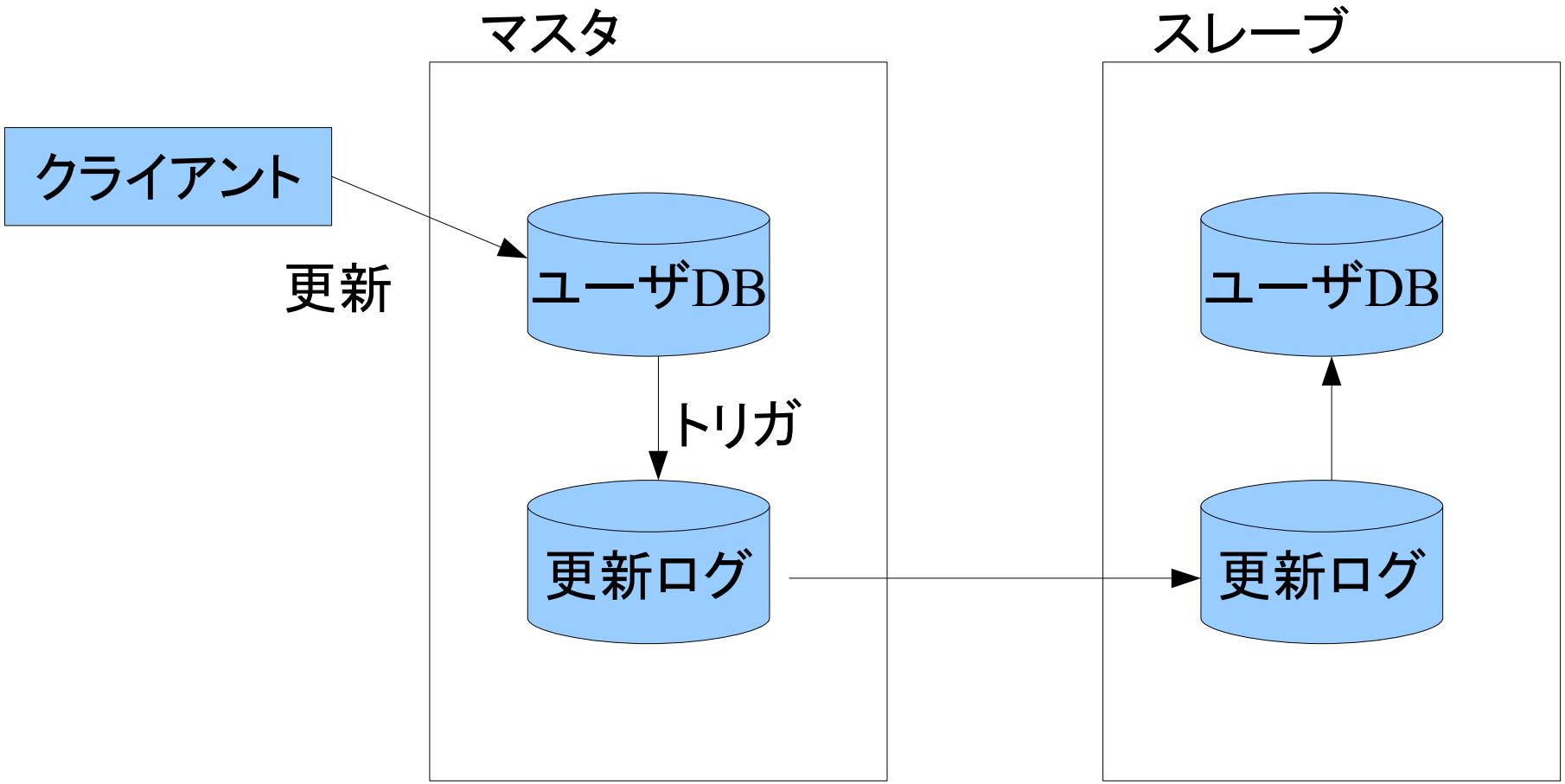
レプリケーションソリューション

PowerGres

- ソフト的にDBの同期を取る
- 可用性だけでなく，性能向上も狙えることがある
- もっともホットな分野
- 今回紹介するもの
  - Slony-I
  - PostgresForest
  - PGCluster
  - pgpool-II

5) Slony-I

- PostgreSQLの開発メンバーが開発，維持
- BSDライセンスで配布
- トリガベースの非同期レプリケーション
  - マスタ，スレーブ間の更新遅延が問題になる場合は要注意
- 一つのマスタ，複数のスレーブ．マスタのみが更新を受け付ける


マスタ
スレーブ
クライアント
更新
ユーザDB
トリガ
更新ログ
ユーザDB
更新ログ

Slony-Iの評価

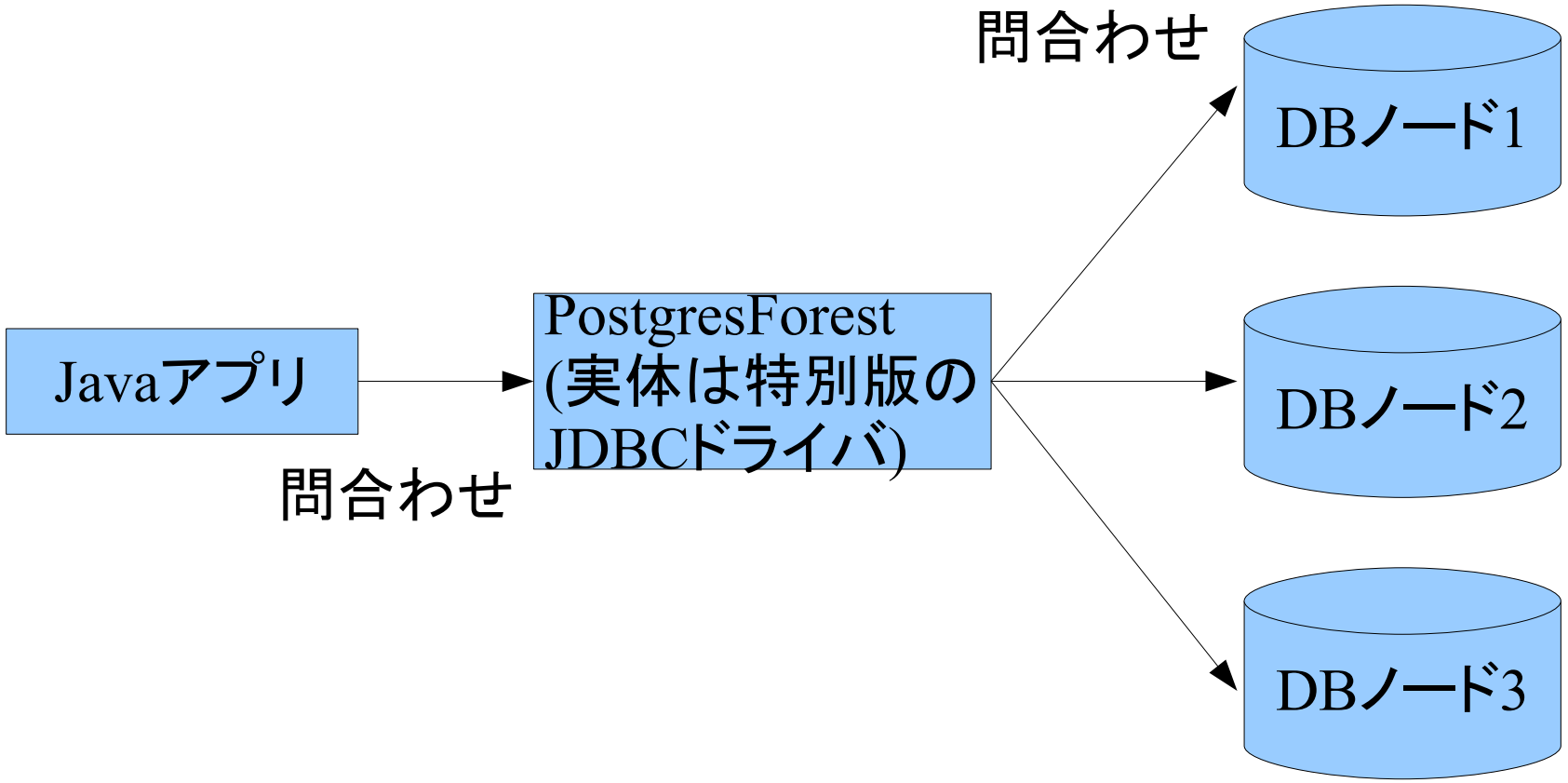
- メリット
  - ノードを柔軟に追加できる
  - 実績がある
- デメリット
  - ノード間でデータの一時的なずれがある
  - 負荷分散ができない
  - 設定が面倒
  - レプリケーション出来ないデータがある
  - ノード障害に自動対応できない
  - パラレルクエリに対応していない

6) PostgresForest

PowerGres

- NTTデータが開発
- BSDライセンスで配布
- JDBCドライバを拡張することにより，以下の機能を実現
  - レプリケーション，負荷分散，パラレルクエリ
- 使用できる言語はJavaのみ

PostgresForestのアーキテクチャ



PostgresForestの評価

- メリット
  - 負荷分散，パラレルクエリによる検索系の性能向上が期待できる
  - ノード障害に自動対応できる
- デメリット
  - APIはJavaのみ
  - 更新系の性能向上は期待できない
  - パラレルクエリで対応できない問合わせがある

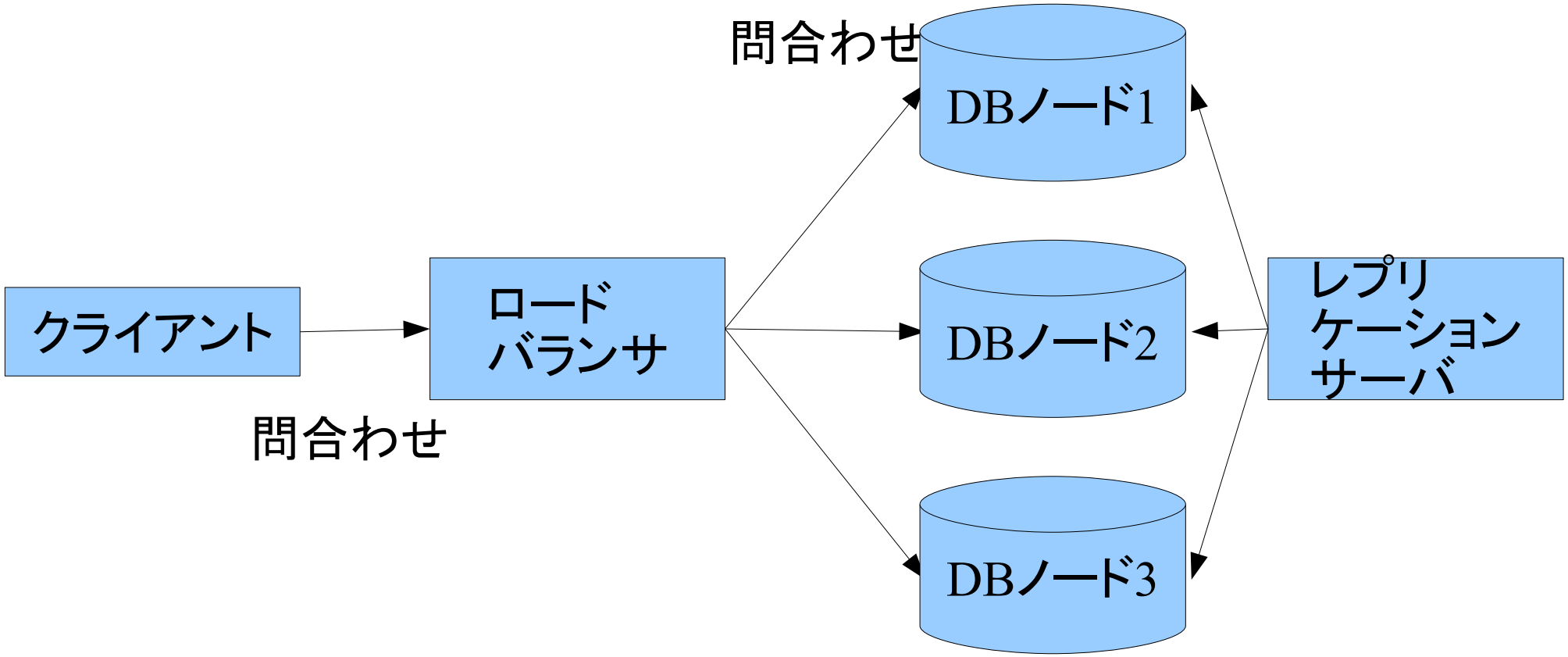
7) PGCluster

PowerGres

- 三谷氏が個人的に開発
- BSDライセンスで配布
- PostgreSQLをベースに改造
- コネクションプーリング，負荷分散，レプリケーション
- 使用言語を選ばない

SRA OSS, INC.

PGClusterのアーキテクチャ



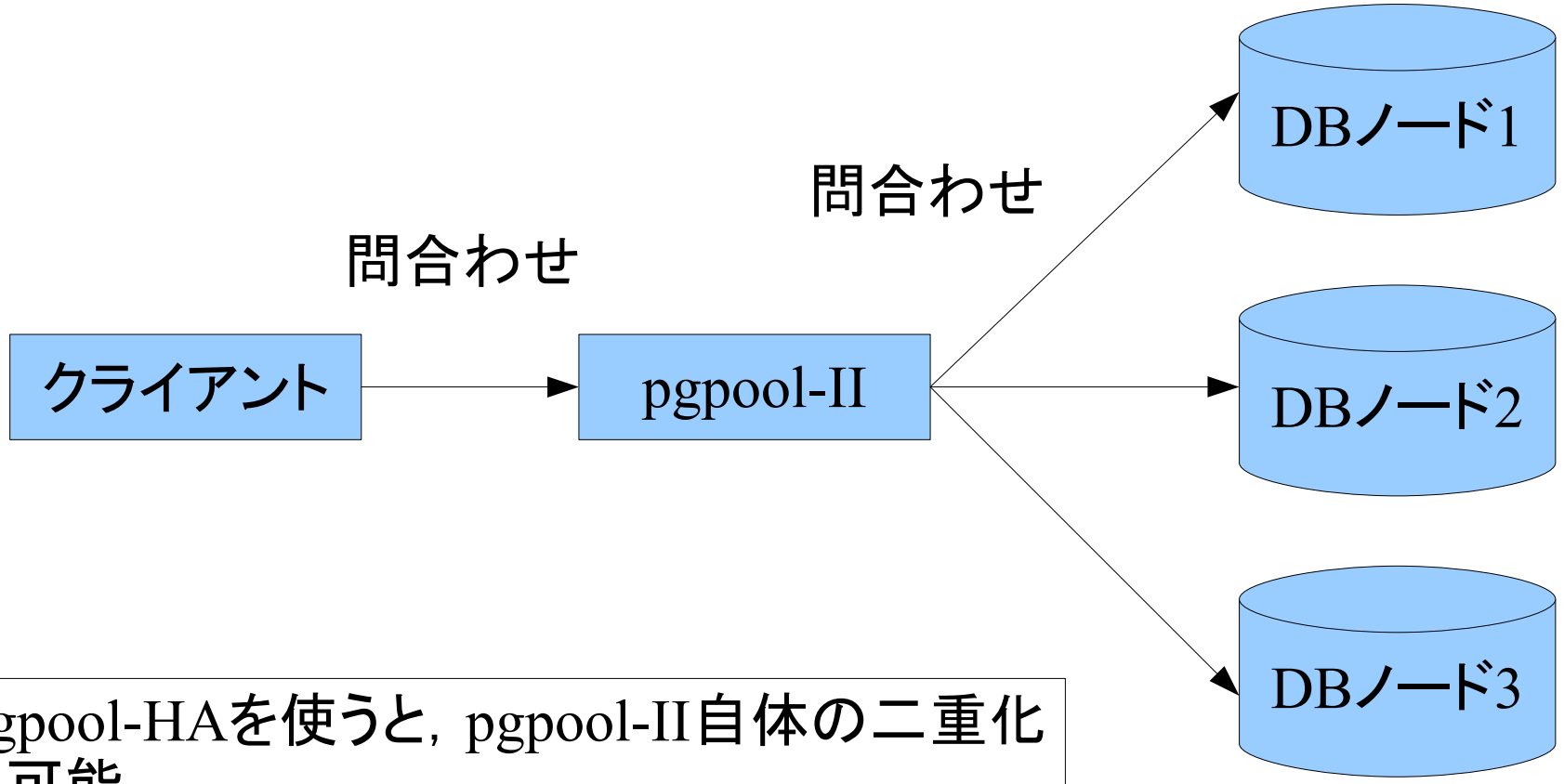
PGClusterの評価

- メリット
  - 負荷分散による検索系の性能向上が期待できる
  - ノード障害に自動対応できる
- デメリット
  - 設定がやや面倒
  - PostgreSQLにパッチを当てているので，最新バージョンへの追随が遅れる
  - 更新系の性能が非常に低い
  - パラレルクエリには対応していない
  - フル機能を利用するためには5ノード必要

8) pgpool-II

- pgpool Global Development Groupが開発
- BSDライセンスで配布される
- 多彩な機能
  - コネクションプーリング，レプリケーション，負荷分散，パラレルクエリ
- 設定が容易．GUI管理ツールも附属
- PostgreSQLに手を入れていないので，バージョン追従が容易
- 使用言語を選ばない

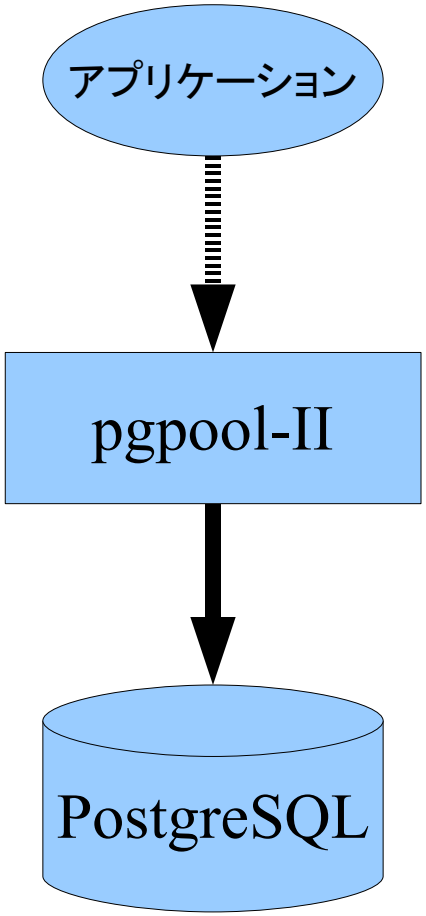
pgpool-IIのアーキテクチャ



pgpool-HAを使うと，pgpool-II自体の二重化も可能

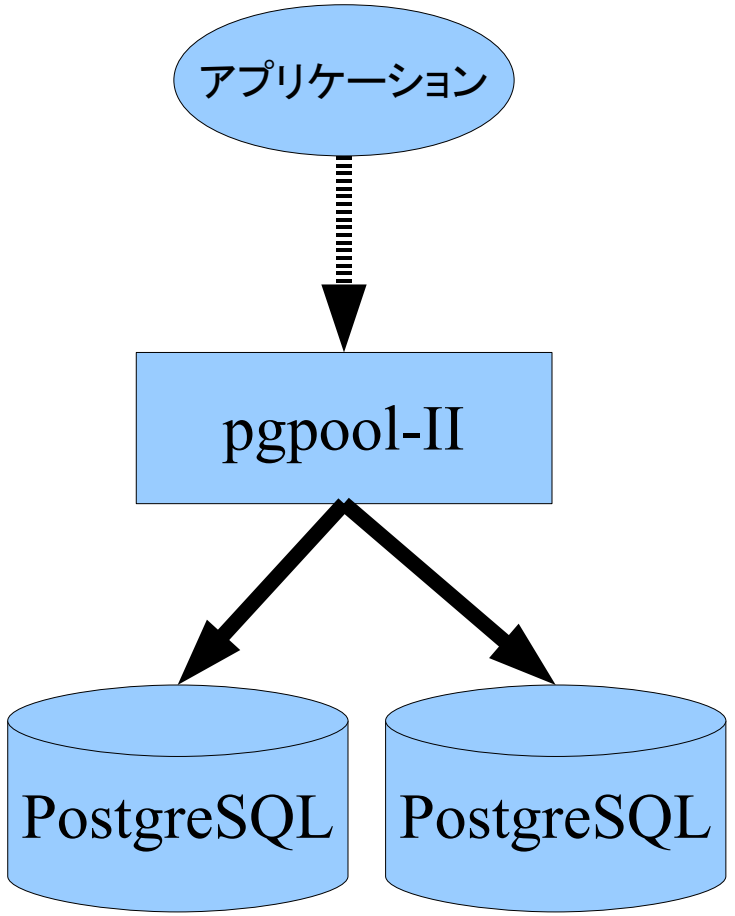
pgpool-IIの機能

- コネクションプーリング
  - データベースへの接続オーバヘッドを軽減
  - PostgreSQLへのコネクションを保持しておき，再利用する
  - Webシステムのように，頻繁に接続/切断を繰り返すシステムで効果あり



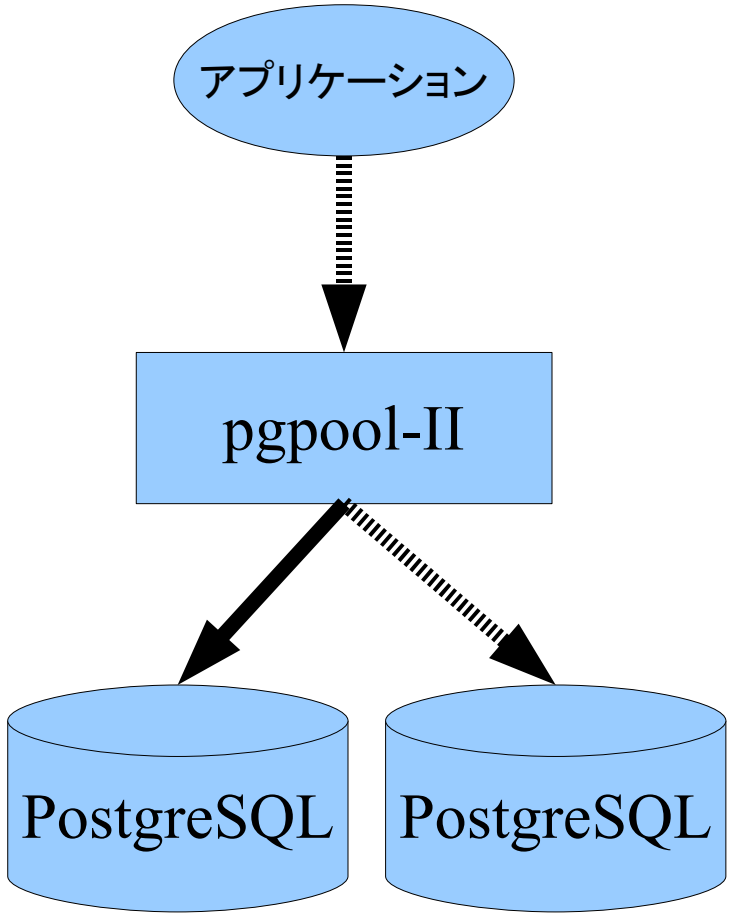
pgpool-IIの機能

- レプリケーション
  - SQL文を複製してデータベースのコピーをリアルタイムに作る
  - 片方のDBが障害を起しても自動的に切り離して運用を継続
  - 障害復旧後は運用を停めずにDBを同期，復帰させることが可能（オンラインリカバリ）



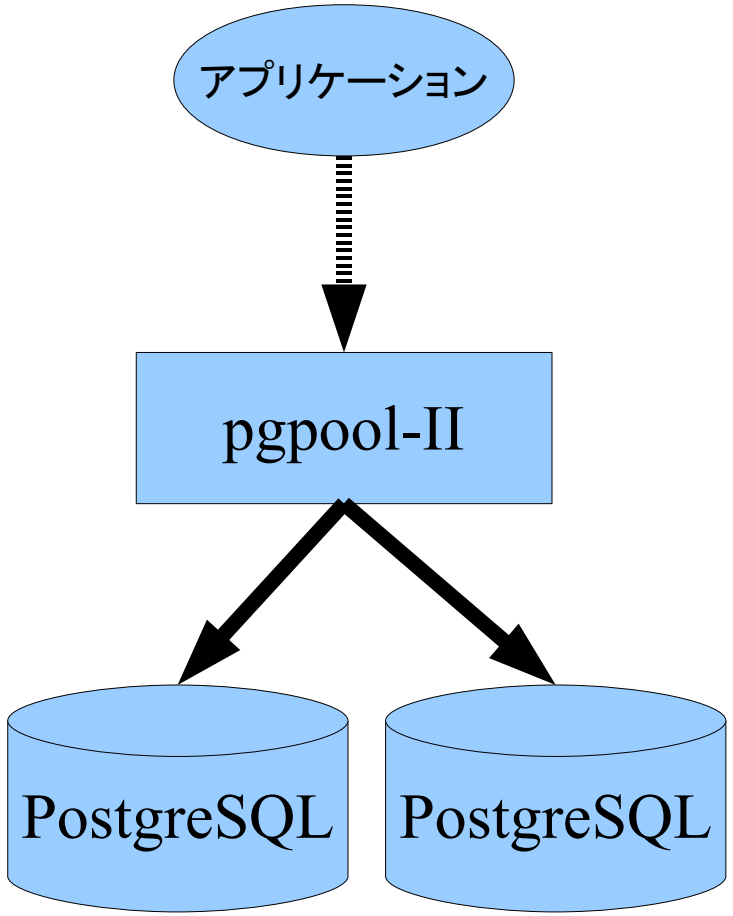
pgpool-IIの機能

- 負荷分散
  - 検索問合わせをランダムに決めたPostgreSQLに振り向ける
  - 負荷をPostgreSQL間で分かち合うので，検索性能が向上



pgpool-IIの機能

- パラレルクエリ
  - データを分割してPostgreSQL間で分担
  - 検索問合わせを複数のPostgreSQLで一斉に実行，結果をpgpool-IIでまとめる
  - 問合わせを並列に処理するので高速



pgpool-IIの評価

- メリット
  - 負荷分散，パラレルクエリによる検索系の性能向上が期待できる
  - ノード障害に自動対応できる
  - APIを選ばない
  - GUI管理ツールがあるので設定，管理が容易
- デメリット
  - パラレルクエリで対応できない問合わせがある
  - 更新系の性能向上は期待できない

レプリケーションの総合比較

PowerGres

	pgpool-II	PGCluster	Postgres Forest	Slony-I	
対応PostgreSQLバージョン	7.3以降	7.3以降	7.4-8.1	7.4以降	注1
対応プラットフォーム	△	△	△	○	注2
対応言語	○	○	△	○	注3
コネクションプーリング	○	○	×	×	
同期レプリケーション	○	○	○	×	
自動縮退	○	○	○	×	
オンラインリカバリ	○	○	○	×	
負荷分散	○	○	○	×	
パラレルクエリ	○	×	○	×	
検索性能	○	○	○	△	
更新性能	△	×	△	△	
DDL対応	○	○	○	×	
ラージオブジェクト対応	○	○	○	×	
シーケンス対応	○	○	○	○	
SERIAL型対応	△	○	×	○	注4
拡張問合わせ	△	△	○	○	注5

注1：各ソフトの最新バージョンの対応状況
注2：Slony-IのみWindowsでも動作可能
注3：PostgresForestのみJava限定

注4：pgpoolはロックを併用することで対応可能
注5：pgpoolはパラレルクエリが拡張と問合わせに対応していない
　　PGClusterは無名PREPAREのSELECTだけが負荷分散される

どのソリューションを使うべきか？

PowerGres

- コストをかけられない，可用性はほどほどでよい
  - バックアップ/リストア，ウォームスタンバイ，Slony-I
- コストをかけても可用性を高めたい
  - ftServer, PowerGres HA
- 可用性だけでなく，検索性能も高めたい
  - pgpool-II, PostgresForest(Javaのみ)
- DBが遠隔地にある，あるいは接続が保証されていない
  - Slony-I