Pgpool-IIの歩みと進化
基本機能から最新ユースケースまで

2026/3/24

株式会社SRA OSS
彭 博

自己紹介

ペン ボ

- 名前： 彭 博 (Bo Peng)
  pengbo@sraoss.co.jp
- 所属：株式会社SRA OSS
  基盤技術グループ
- 職務：
  - OSS技術サポート、ミドルウェア構築、コンサルティング
  - PostgreSQLクラスタ管理ツールPgpool-II開発者

会社紹介

株式会社SRA OSS

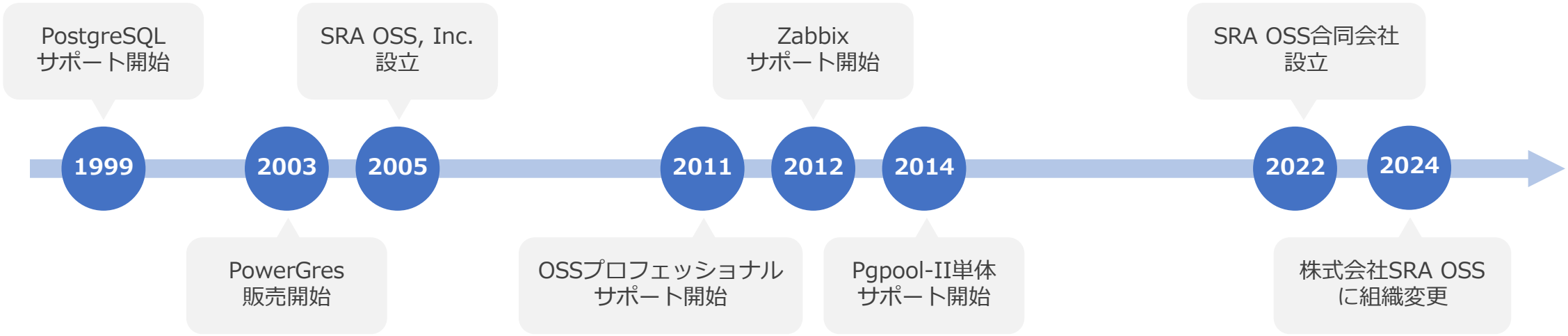
所 在 地: 東京都豊島区南池袋2-32-8
設 立 日: 2022年6月17日
株　　主: 株式会社SRA
株式会社NTTデータ
資 本 金: 7,000万円
社　　長: 稲葉 香理

事業内容

- オープンソースソフトウェア (OSS) 関連のサポート、製品開発・販売、構築・コンサル
- OSSの教育、開発、コミュニティ運営支援
- ソフトウェアの研究開発

顧　　問: 石井達夫

技術顧問: 増永 良文 (お茶の水女子大学名誉教授)

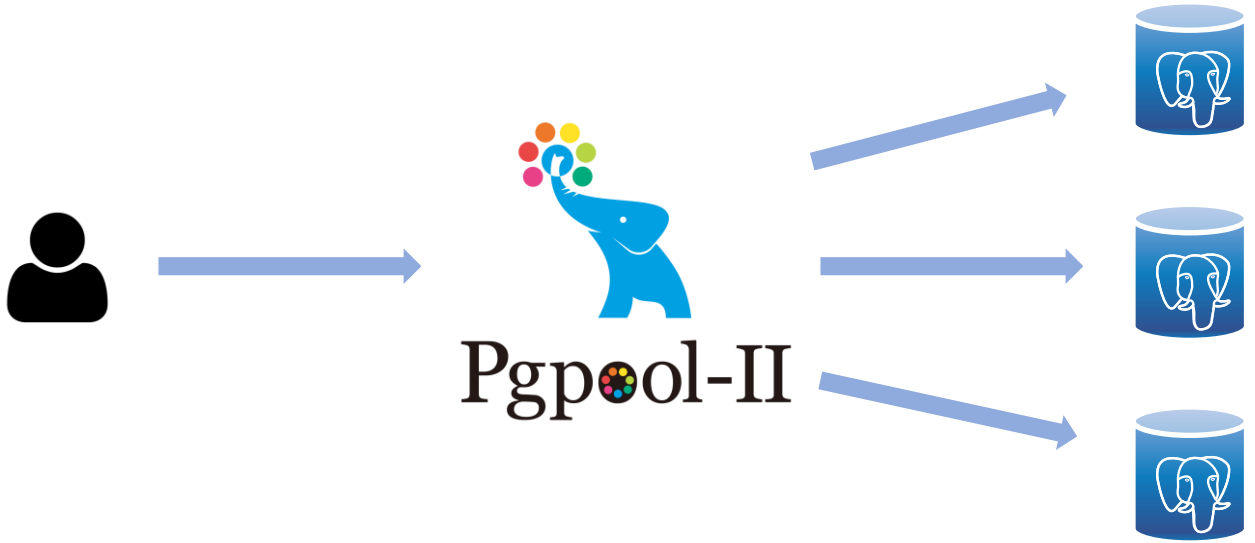


本セッションの流れ

- Pgpool-IIの歴史
- Pgpool-IIの概要および機能紹介
- Pgpool-IIの活用シーン (オンプレミスからクラウドまで)
- 今後の進化の方向性

Pgpool-IIとは

- Pgpool-IIはクライアントとPostgreSQLの間で動作するミドルウェア
- Pgpool Global Development Groupによって開発・メンテナンスされているOSS
- PostgreSQL単体では実現できない自動フェイルオーバー、負荷分散、コネクションプーリングなどの機能を提供
- ユーザは複数PostgreSQLサーバを意識せず、1台のように見える



Pgpool-IIの歴史

Pgpool-IIの誕生

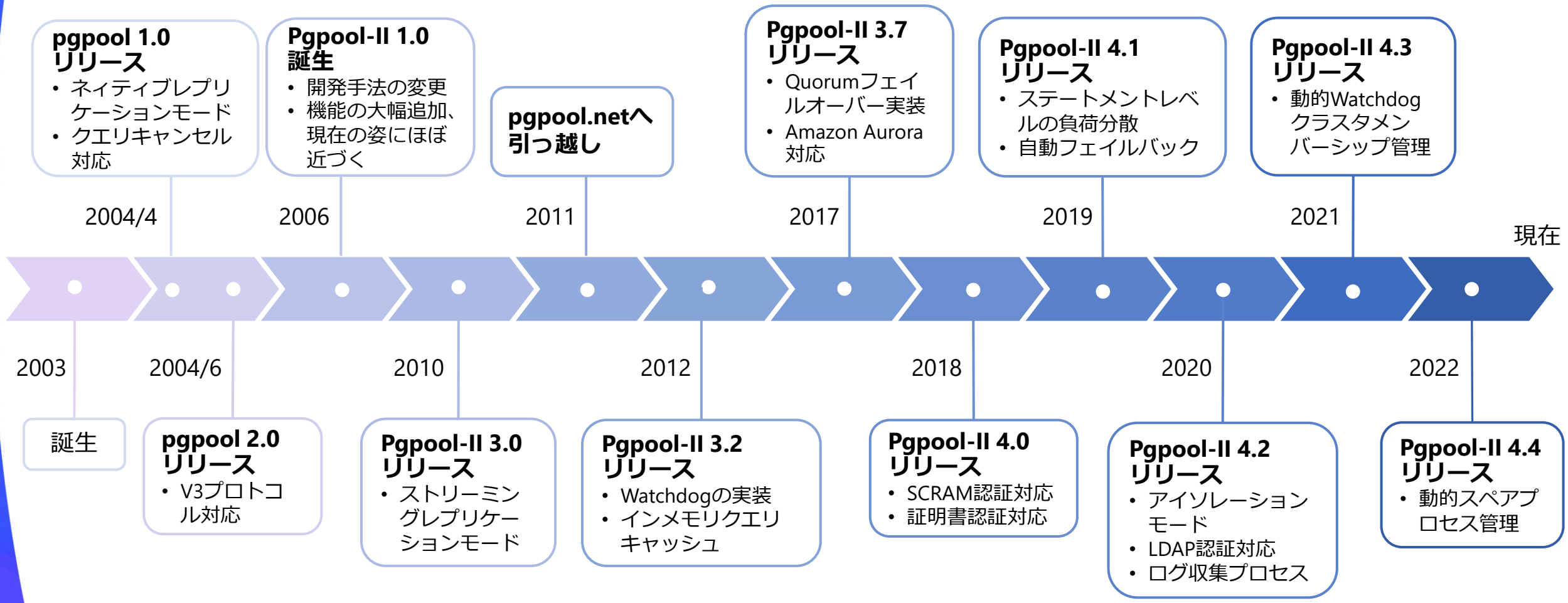
- **2003年**、PostgreSQL向けのコネクションプーリングソフトとして公開
- 当時の名称は「**pgpool**」
  （現在の「Pgpool-II」の前身）
- 初期バージョンの特徴
  - コネクションプーリング機能を提供
  - フェイルオーバー機能も同時に実装
  - サポートするPostgreSQLサーバは最大2台まで
  - 現在のRawモードに相当

(当時の投稿内容)

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．

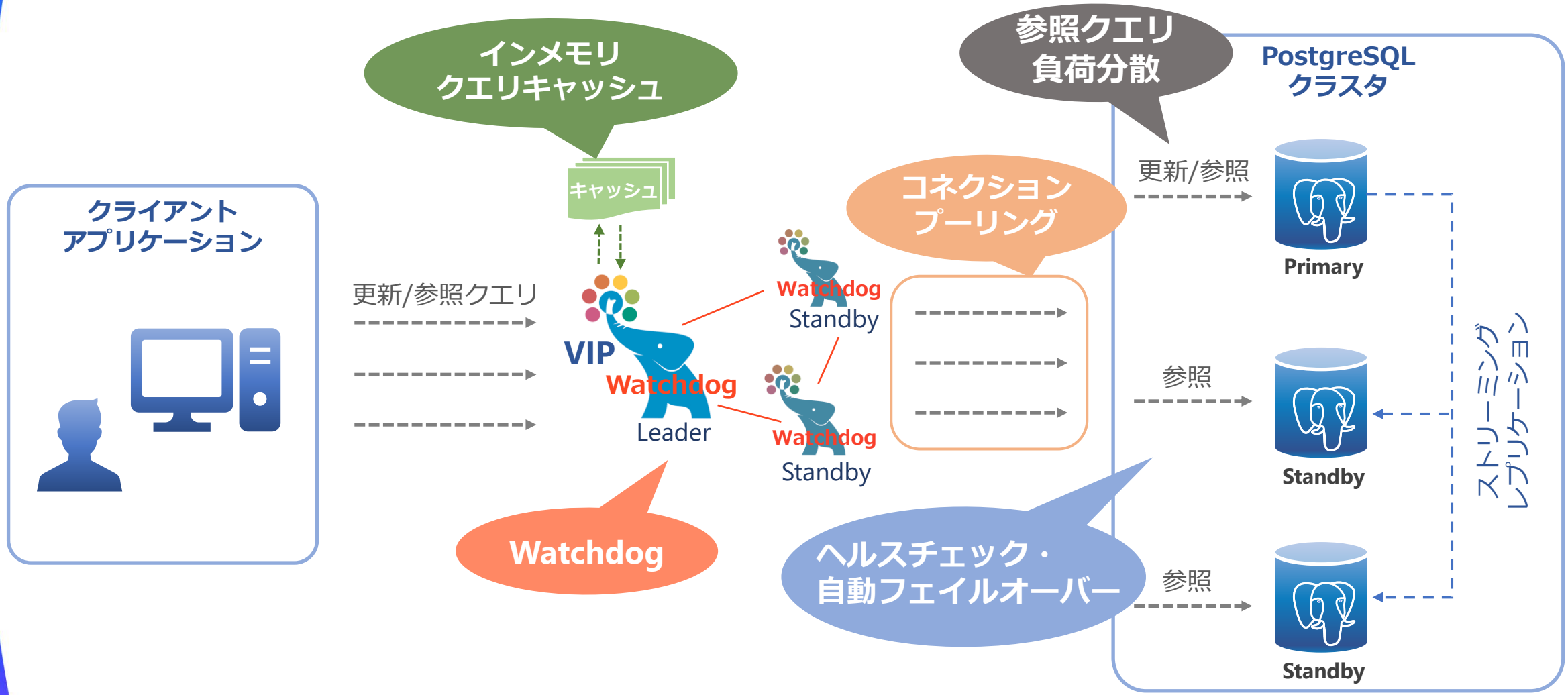
Pgpool-IIの歴史

pgpoolの誕生からPgpool-IIへ、20年以上続く進化の歩み



Pgpool-IIの主な機能

Pgpool-IIのコア機能



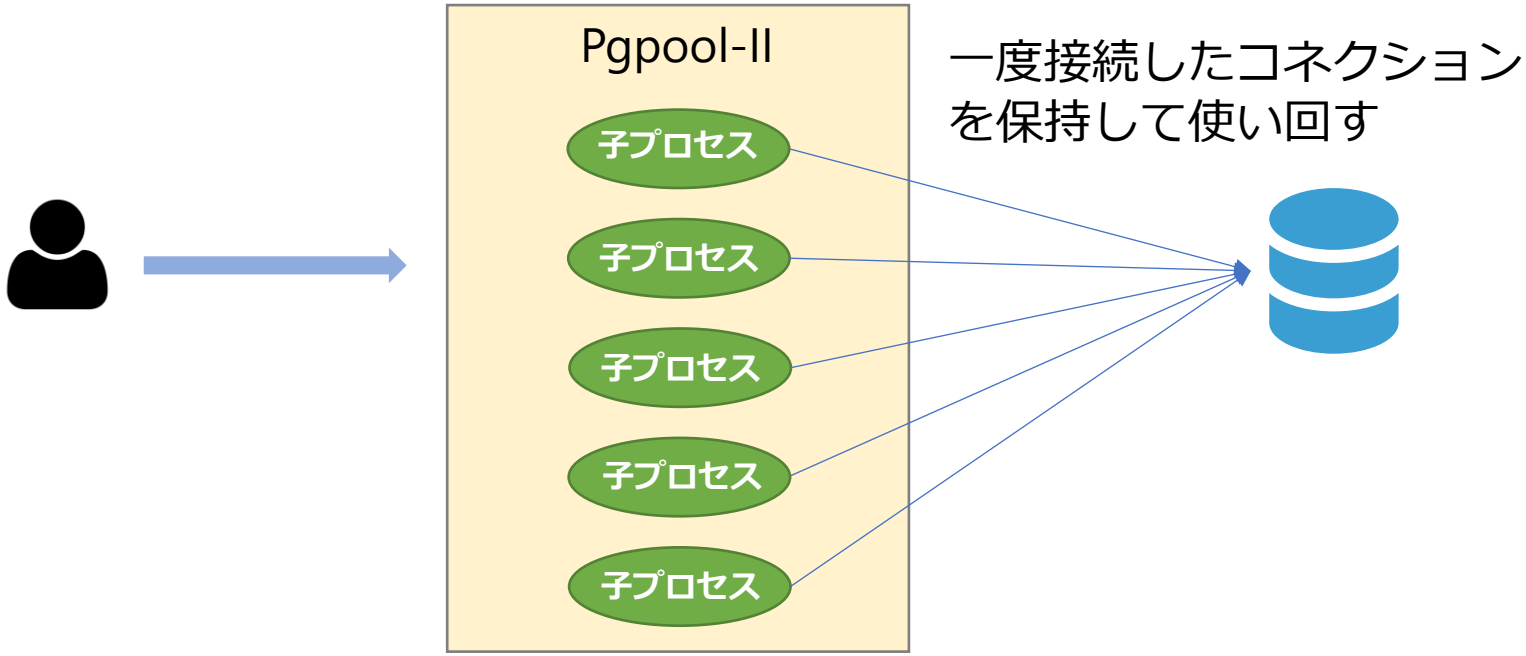
自動フェイルオーバー

- Pgpool-IIは定期的に各PostgreSQLの状態を監視する
- PostgreSQLの障害を検知すると、フェイルオーバーを実行する

コネクションプーリングによるDB接続高速化

なぜコネクションプーリング？

- データベースにアクセスするたびに、接続処理にオーバーヘッドが発生する
- この処理を毎回繰り返すと、レスポンスが遅くなる
- 接続を保持し、使い回すことで、接続確立のオーバーヘッドを削減できる



負荷分散 (参照クエリ振り分け)

負荷分散

- SQLパーサを搭載しており、クエリを解析できる
- それでも足りない情報は、Pgpool-IIが自動的にPostgreSQLに問い合わせる (例えば、関数の場合)
- 更新クエリはプライマリに送信、参照クエリは複数のPostgreSQLノード間で分散

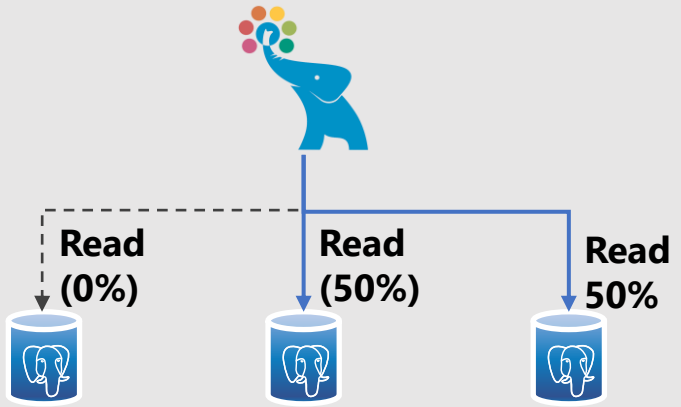
負荷分散モード

- セッションレベル (デフォルト)
- ステートメントレベル

負荷分散の比率設定

例えば、プライマリを更新処理専用にし、すべての参照クエリをスタンバイに振り分けたい場合

```
backend_weight0 = 0
backend_weight1 = 1
backend_weight2 = 1
```



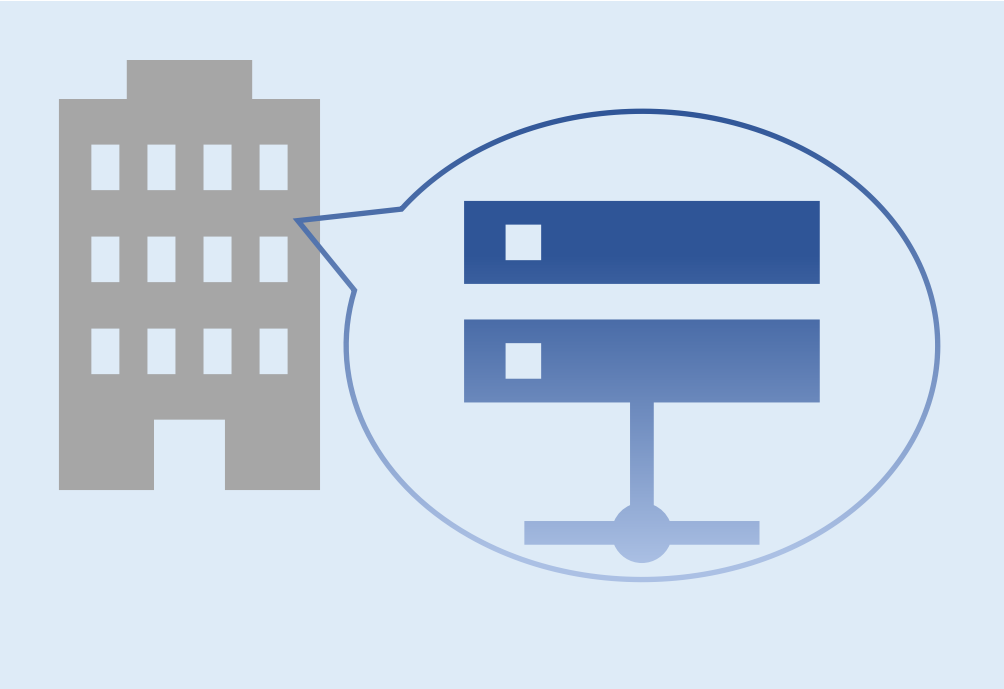
Pgpool-IIの活用シーン

オンプレミスからクラウド/マネージドDBサービスまで

Pgpool-IIの活用シーン

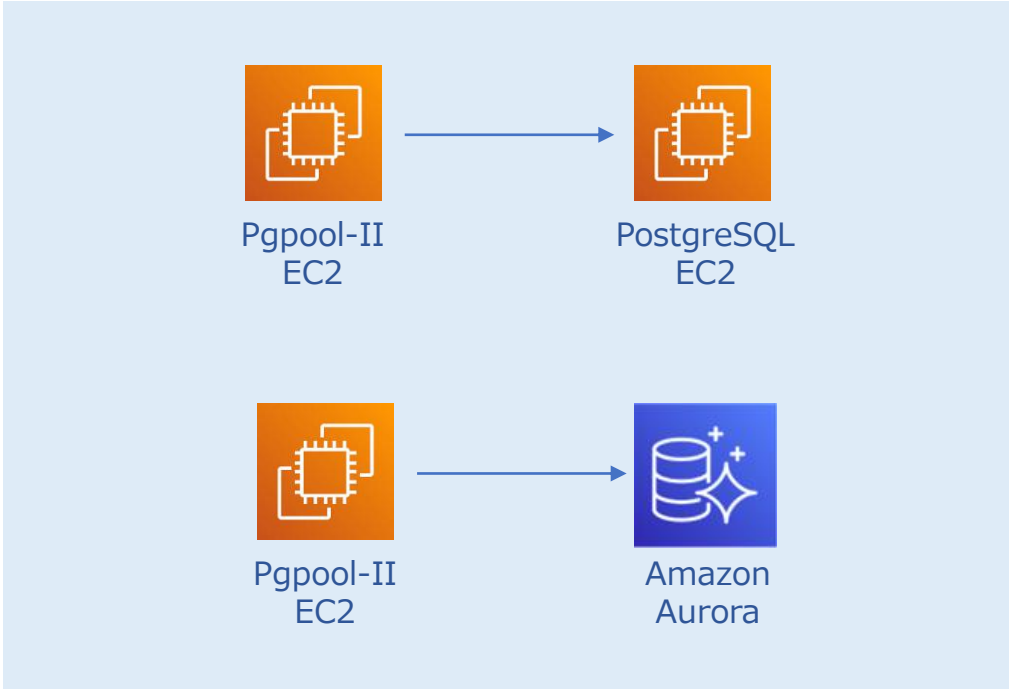
オンプレミス環境

物理サーバ、仮想マシン



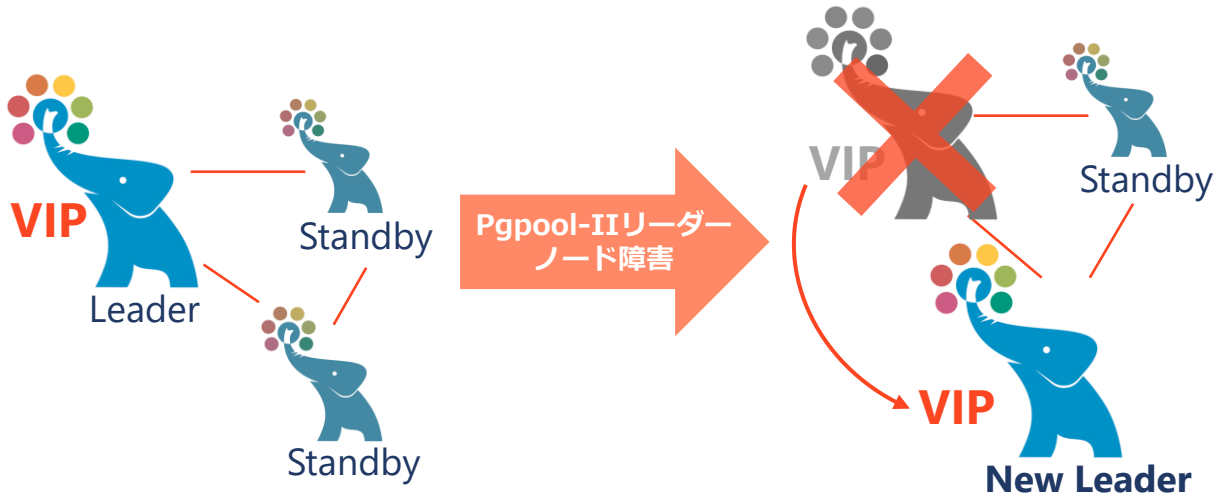
クラウド環境

EC2インスタンス
マネージドDB Amazon Auroraと組み合わせて利用可



クラウド上で使用可能な仮想IPの代替案

- Pgpool-IIではリーダーノードに仮想IPを付与
- Pgpool-IIリーダーノードのスイッチオーバー時には、新しいリーダーへ仮想IPを付け替える
- クライアントは接続先IPを変更せず利用可能



しかし、クラウドの制約により仮想IPは利用できない

クラウド上で使用可能な仮想IPの代替案

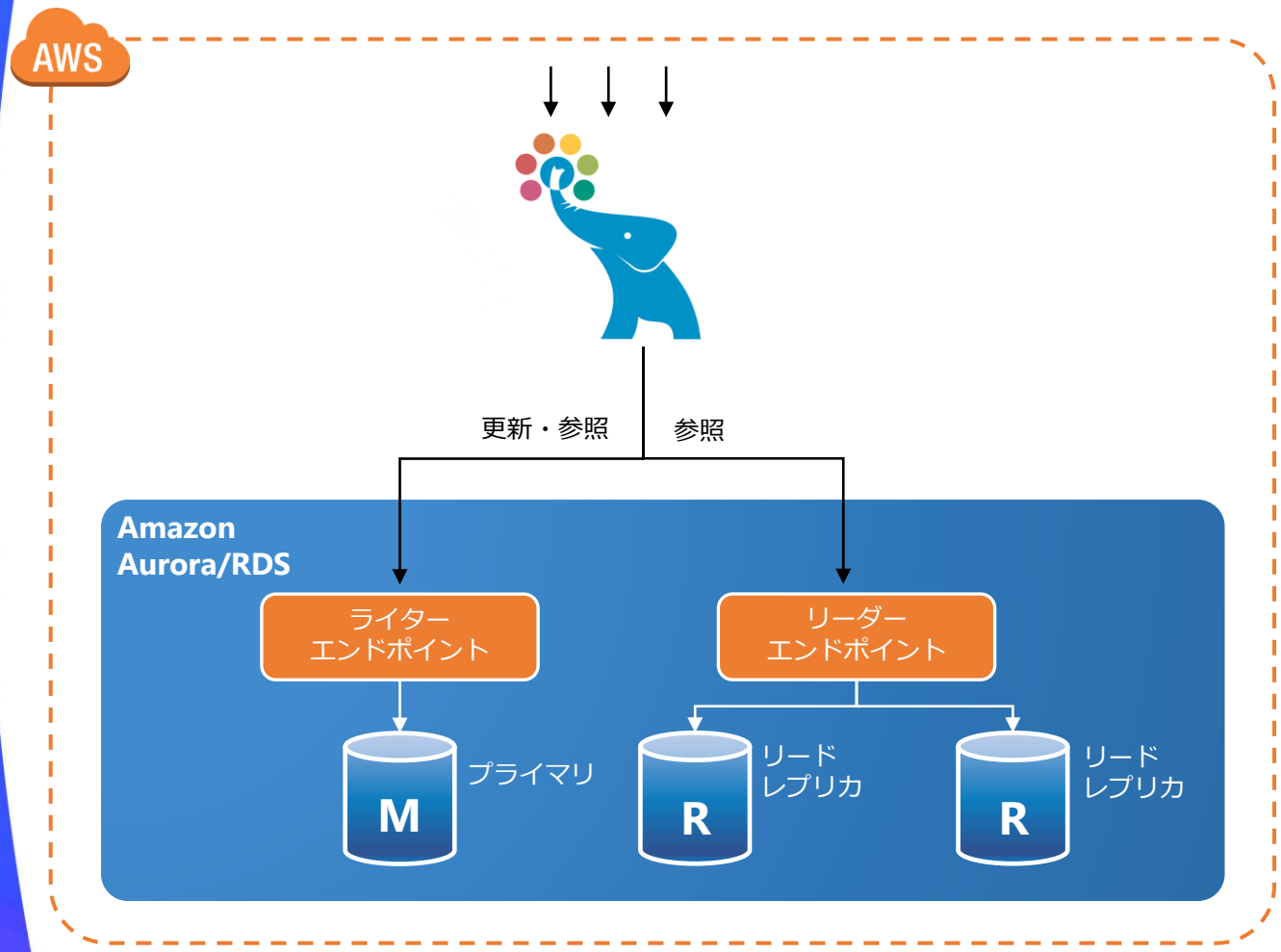
- Elastic IPの付け替え
- セカンダリプライベートIPの付け替え
- ルートテーブルの書き換え
- DNSサービス（Route53など）のAレコードの書き換え

※ Pgpool-IIは「Elastic IPの付け替え」「ルートテーブルの書き換え」用のサンプルスクリプトを提供している

クラウド上で使用可能な仮想IPの代替案

	Multi-AZ対応	クライアント (接続元)	Pgpool-II EC2インスタンスのサブネット	備考
Elastic IPの付け替え	○	インターネット経由で接続 (パブリックサブネット)	パブリックサブネットに配置する必要がある	
セカンダリプライベートIPの付け替え	×	VPC内部	プライベートサブネットで利用可	
ルートテーブルの書き換え	○	VPC内部	プライベートサブネットで利用可	
DNSサービスのAレコードの書き換え	○	VPC内部 別のピア接続VPC オンプレミス	プライベートサブネットで利用可	設定変更の反映には時間がかかる

Pgpool-II + Amazon Aurora



- ストリーミングレプリケーションモード
  (デフォルトで負荷分散、コネクションプールが有効)

```
backend_clustering_mode = 'streaming_replication'
load_balance_mode = on
connection_cache = on
```

- フェイルオーバーはAWSに任せるので、ヘルスチェックを無効にする (デフォルトで無効)

```
health_check_period = 0
```

- バックエンドエラーが発生した時にフェイルオーバーを起こさないように

```
failover_on_backend_error = off
```

- バックエンドノード情報にライターエンドポイントとリーダーエンドポイントを設定

```
backend_hostname0 = 'ライターエンドポイント'
backend_hostname1 = 'リーダーエンドポイント'
```

- プライマリノードを自動判別せず固定

```
backend_flag0 = 'ALWAYS_PRIMARY|DISALLOW_TO_FAILOVER'
backend_flag1 = 'DISALLOW_TO_FAILOVER'
```

- ストリーミングレプリケーション遅延チェックを無効にする

```
sr_check_period = 0
```

今後の進化の方向性

ー ユーザの実運用課題がきっかけとなり進化していく

今回は代表的な2つの課題とその取り組みを紹介

① マルチリージョン構成でのレイテンシ増大

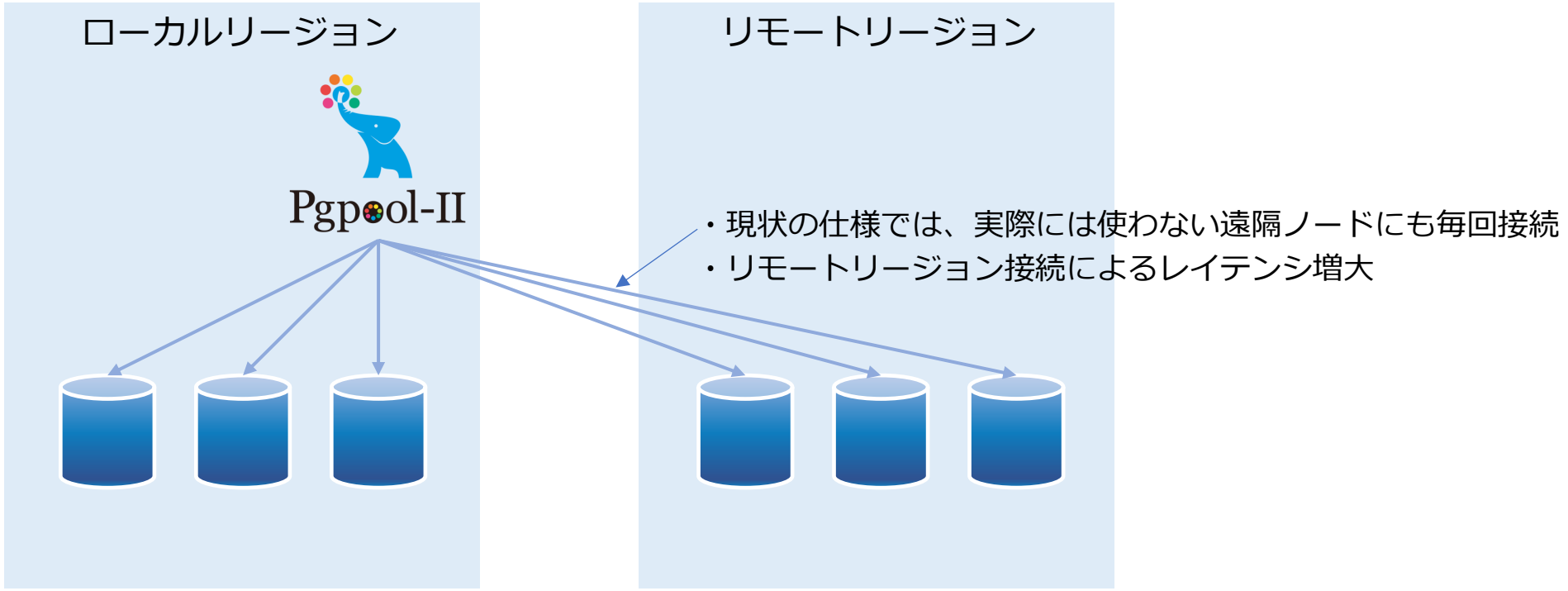
② PostgreSQL互換DBへの対応

課題① マルチリージョン構成でのレイテンシ増大

課題① マルチリージョン構成でのレイテンシ増大

ユーザ課題

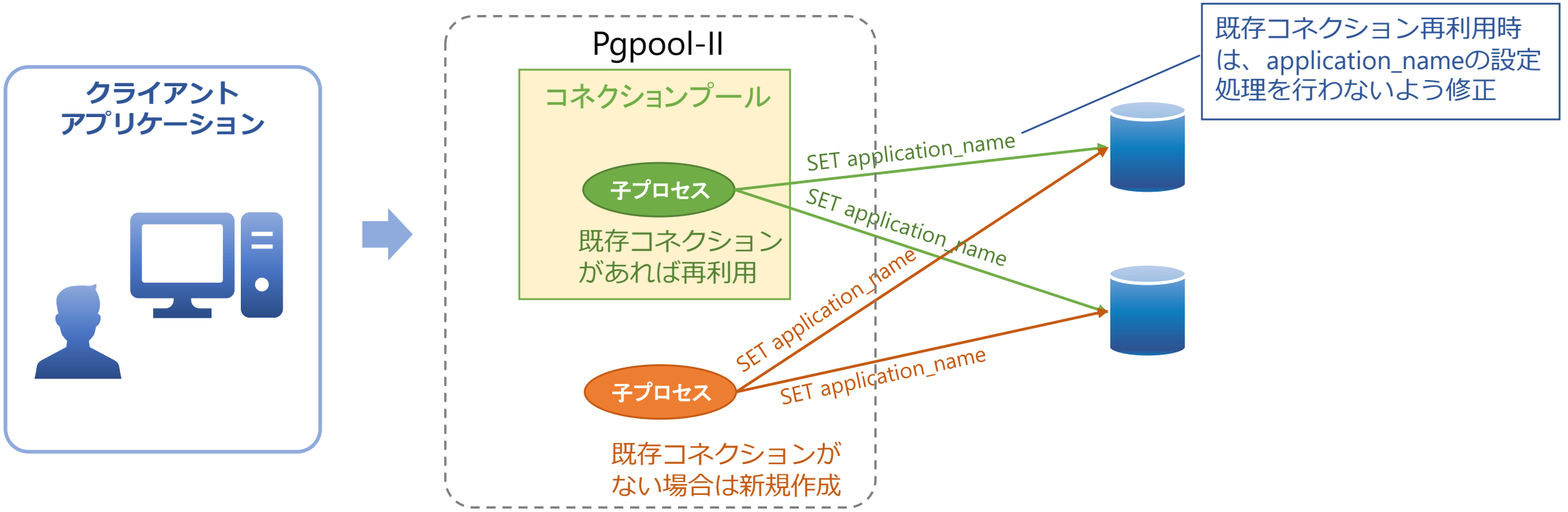
マルチリージョン構成で性能が低下



課題① マルチリージョン構成でのレイテンシ増大

原因①：接続時のSET application_name処理

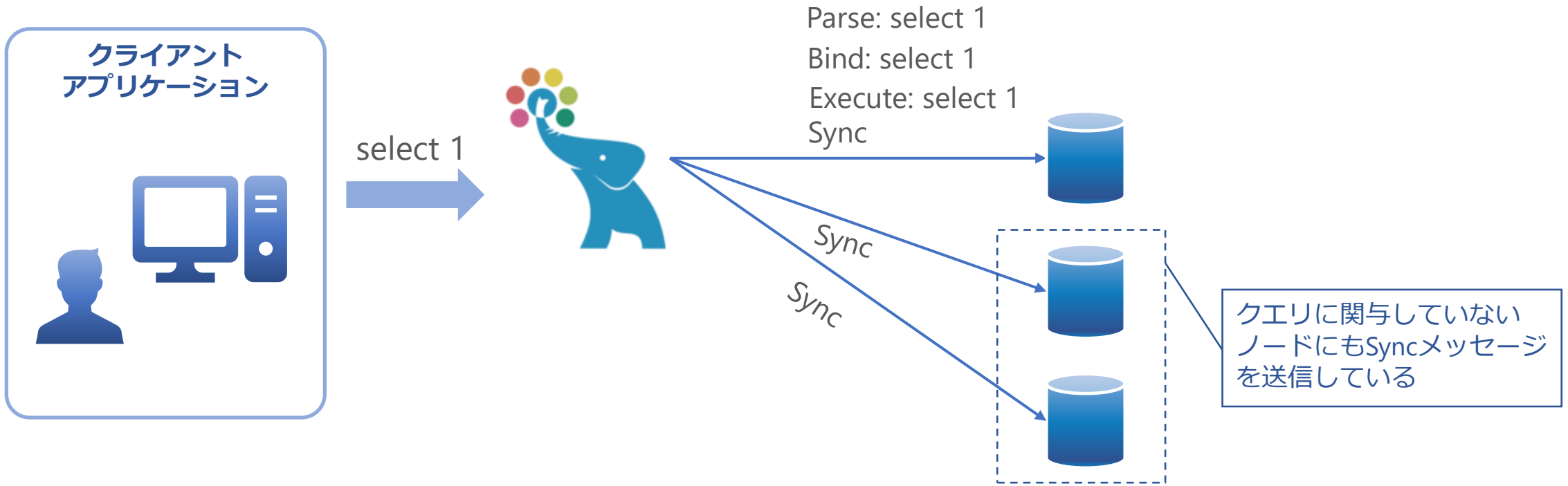
- 初回だけでなく、コネクション再利用時にも、バックエンド接続時に**SET application_name**を全ノードに対して実行し、その応答を待つため、リモートリージョン環境では接続確立時間が増大
- 調査の結果、コネクション再利用時はapplication_nameの設定処理は不要と判明
- 不要な処理を削除し、マルチリージョン環境での性能を改善 (v4.7で修正済)



課題① マルチリージョン構成でのレイテンシ増大

原因②：全バックエンドへの「Sync」メッセージ送信

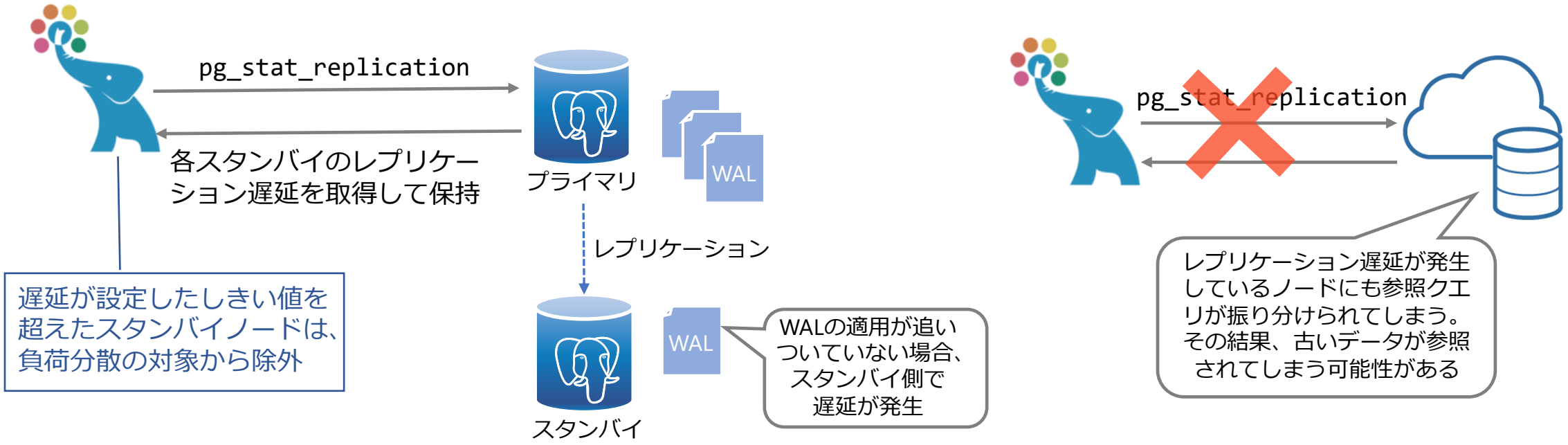
- Pgpool-IIは拡張クエリプロトコル使用時、**すべてのバックエンドノードへSyncを送信**
- 本来はクエリ実行ノードのみSyncが必要だが、実際には全ノードへ送信しているため、SQL実行時間がノード数・距離・クエリの数に比例して増大（特にリモートリージョンで影響が顕著）
- 修正方針：必要なバックエンドのみにSyncを送信 (次のメジャーリリースで対応予定)



課題② PostgreSQL互換DBのサポート

課題② PostgreSQL互換DBのサポート

- **PostgreSQL互換DBでレプリケーション遅延が取得できない問題**
  - Pgpool-IIは**pg_stat_replication**を用いてレプリケーション遅延を取得
  - しかし、PostgreSQL互換DBでは**pg_stat_replicationを利用できない**ケースがある (例：Amazon Aurora)
  - その結果、レプリカ遅延を考慮した安全なクエリ振り分けができない



課題② PostgreSQL互換DBのサポート

解決アプローチ

- レプリケーション遅延取得方法をユーザが自由に定義可能に
- 新しいパラメータ (次のメジャーリリースで対応予定)

replication_delay_source_cmd
レプリケーション遅延を取得するコマンドを指定

replication_delay_source_timeout
上記パラメータで指定した外部コマンドのタイムアウトを指定

課題② PostgreSQL互換DBのサポート

replication_delay_source_cmdの仕組み

- スタンバイノードのbackend_hostnameX / backend_portXから**<hostname>:<port>**形式のノード識別子を生成
- 生成した識別子を、replication_delay_source_cmdに指定した外部コマンドの引数として渡す

```
(この例では、node0がプライマリ)
backend_hostname0 = 'server0'
backend_port0 = 5432
backend_hostname1 = 'server1'
backend_port1 = 5432
backend_hostname2 = 'server2'
backend_port2 = 5432
```

<外部コマンド> server1:5432 server2:5432

<server1の遅延> <server2の遅延>

外部コマンドの出力例：**25.5 100**
- server1の遅延が 25.5ms
- Server2の遅延が 100ms

show pool_nodes結果抜粋

```
node_id | hostname  | port  | status | pg_status | role    | pg_role  | replication_delay
---------+----------+-------+--------+-----------+---------+----------+-------------------
 0       | server0  | 5432  | up     | up        | primary | primary | 0
 1       | server1  | 5432  | up     | up        | standby | standby | 0.025500 second
 2       | server2  | 5432  | up     | up        | standby | standby | 0.100000 second
```

まとめ

- Pgpool-IIのこれまでの歩みを紹介
  - **コネクションプーリング専用ソフトから、オールインワンのHAクラスタソリューションへ進化**
    - コネクションプーリング
    - 自動フェイルオーバー
    - 負荷分散
    - クエリキャッシュ
    - Watchdog
  - オンプレミスからクラウドまで多様な環境で利用可能
- **機能拡充を継続しながら、ユーザの実運用課題を起点に進化を推進**

参考情報

- Pgpool-II wiki
  - https://pgpool.net/mediawiki/ (英語)
  - https://pgpool.net/mediawiki/jp/ (日本語)
- Pgpool-IIドキュメント
  - https://www.pgpool.net/docs/latest/en/html/ (英語)
  - https://www.pgpool.net/docs/latest/ja/html/ (日本語)
- バグ報告
  - https://github.com/pgpool/pgpool2/issues
- ML
  - https://pgpool.net/mediawiki/index.php/Mailing_lists

ご清聴ありがとうございました。