PostgreSQL インサイド

Pgpool-II で高可用なシステム構成を考える

技術を知る

FUJITSU

- [ ] 導入／環境設定
- [ ] 移行
- [ ] 性能
- [ ] チューニング
- [ ] バックアップ／リカバリー
- [x] 冗長化／負荷分散
- [ ] 監視
- [ ] データ連携
- [ ] 災害対策
- [ ] 豆知識

エンタープライズで利用するデータベースシステムは、障害に強く、安定稼動ができるよう、高可用な構成にすることが要求されます。また、将来的にデータ量やアクセス量が増加しても、サーバー増設等による性能向上により、システムの応答速度や処理キャパシティを維持できるよう、スケーラビリティーの高いシステムが要求されることもあります。
これらの要求に対応するために、PostgreSQL の周辺ツールの 1 つである、Pgpool-II を利用した、高可用なシステム構成について解説します。

1. Pgpool-II とは

Pgpool-II は、アプリケーションとデータベースの間で稼動するミドルウェアであり、Linux や Solaris 上で動作します。Pgpool-II は、主に以下の機能を提供しています。今回は、Pgpool-II 4.0.2 をベースに説明します。

表 1 Pgpool-II の主な機能

分類	機能名	説明
性能向上のための機能	負荷分散（ロードバランス）	参照クエリを複数のデータベースサーバーへ効率的に振り分けることにより、より多くのクエリを処理します。
	コネクションプーリング	データベースへの接続を保持/再利用することにより、接続時のオーバーヘッドを低減し、再接続時の性能を向上します。
データベースの高可用のための機能	レプリケーション	複数のデータベースサーバーに常に同じデータを保存することにより、データベースを冗長化します。（注 1）
	自動フェイルオーバー	プライマリー側のデータベースサーバーの障害発生時に、自動的にスタンバイ側のデータベースサーバーに切り替えることで、業務を継続します。
	オンラインリカバリー	業務を止めることなく、データベースサーバーの復旧や追加を行います。
Pgpool-II の高可用のための機能	Watchdog	複数の Pgpool-II を連携させて、相互に死活監視やサーバー情報を共有します。障害発生時には自律的に切り替えを行います。

- （注 1） Pgpool-II には、独自のレプリケーション機能だけでなく、他のレプリケーション機能を利用する方式も選択できます。その中でも、PostgreSQL 本体の「ストリーミングレプリケーション機能」を利用する方式が推奨されています。「ストリーミングレプリケーション」は、PostgreSQL（プライマリー）のトランザクションログ（WAL）を、複数の PostgreSQL（スタンバイ）に転送することで、データベースを複製する機能です。詳細は「ストリーミングレプリケーション ～仕組み、構成のポイント～」で説明しています。

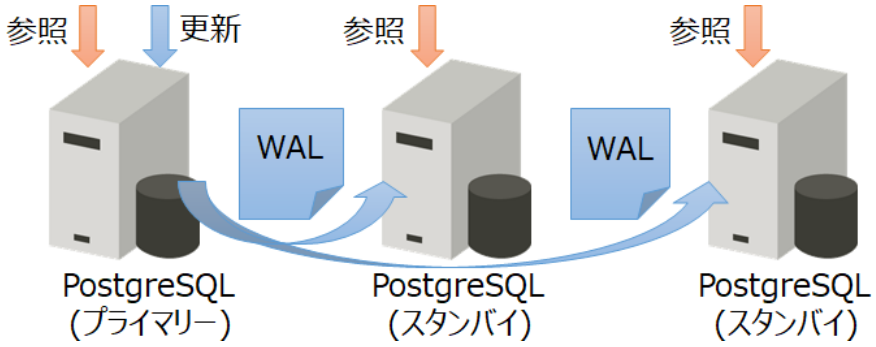

図 1 ストリーミングレプリケーション機能

1.1 負荷分散とコネクションプーリング

データベースを構成するサーバーの台数を増やし、システム全体の処理能力を高める手段として、スケールアウトがあります。PostgreSQL では、ストリーミングレプリケーション機能を使うことでスケールアウトを実現できます。また、スケールアウトを利用するためには、アプリケーションからのクエリを効率的に各データベースサーバーに振り分ける必要があります。PostgreSQL 本体にはその機能がありませんので、Pgpool-II の「負荷分散機能」を使うことによって、参照クエリを効率的に振り分け、負荷を分散することができます。
また、データベースサーバーへの接続にはオーバーヘッドがあるため、その操作が繰り返されるとスループットの低下につながります。Pgpool-II の「コネクションプーリング機能」を使うことにより、保持されたコネクションが再利用できるようになるため、オーバーヘッドを軽減することができます。
これらの処理は、Pgpool-II の設定ファイルの該当パラメーターを設定することで動作させることができます。

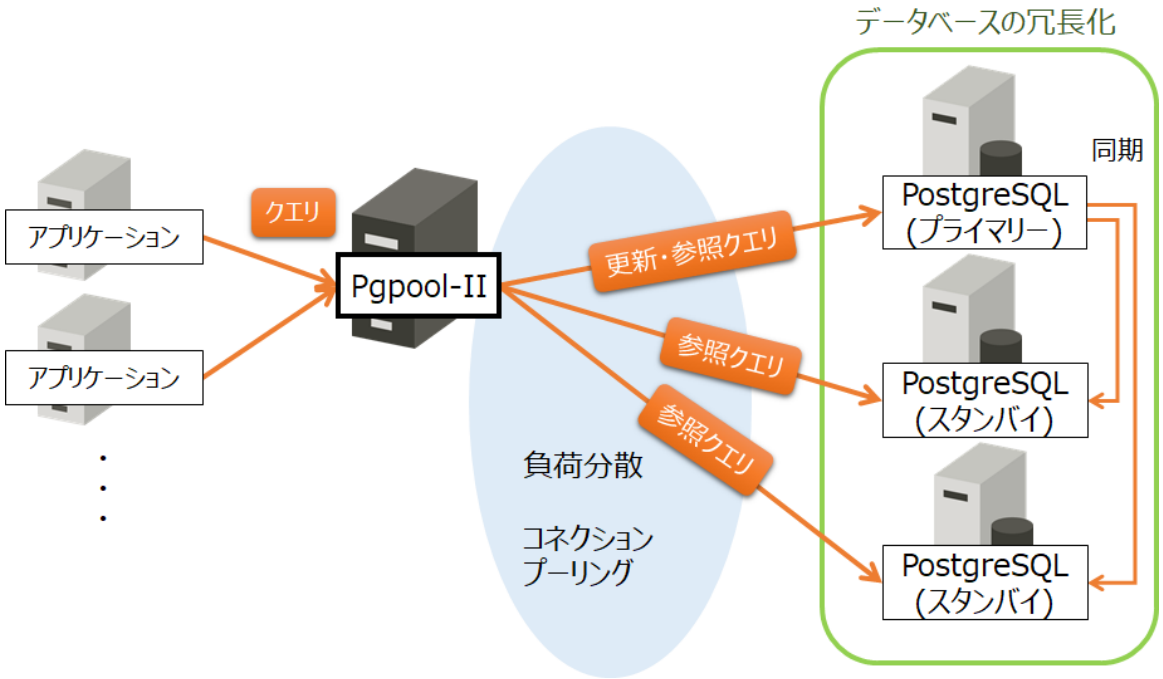

図 2 Pgpool-II の負荷分散とコネクションプーリング

1.2 自動フェイルオーバーとオンラインリカバリー

Pgpool-II の「自動フェイルオーバー機能」と「オンラインリカバリー機能」について説明します。なお、Pgpool-II では、「レプリケーション機能」に、PostgreSQL 本体の「ストリーミングレプリケーション」を利用することを推奨しているため、以降の説明は、「ストリーミングレプリケーション」の利用を前提とします。
「自動フェイルオーバー機能」は、Pgpool-II が PostgreSQL のプライマリーサーバーで異常を検知すると、以下の処理を自動で実行します。なお、②および③の処理は、あらかじめスクリプトで作成し、Pgpool-II に設定しておく必要があります。

- ① 障害が発生したプライマリーサーバーを切り離し、Pgpool-II からのクエリ振り分けを止める
- ② スタンバイサーバーを新プライマリーサーバーに昇格させる
- ③ 各 PostgreSQL について、レプリケーションの同期先を変更する

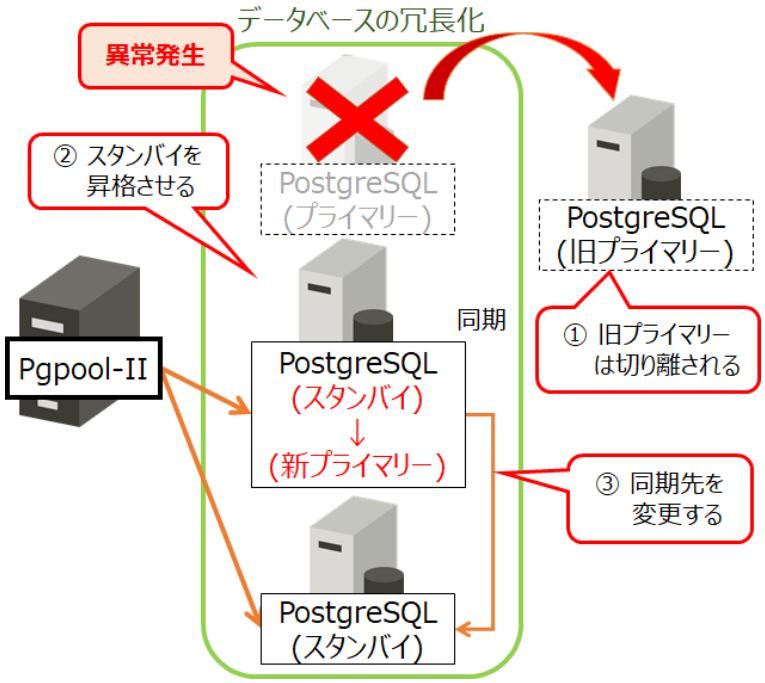

図 3 Pgpool-II の自動フェイルオーバー機能

「オンラインリカバリー機能」は、Pgpool-II からオンラインリカバリー用のコマンドを実行することで、切り離されていた旧プライマリーサーバーをスタンバイとして組み込みます。このコマンドは、以下の処理を行います。なお、これらの処理は、あらかじめスクリプトで作成し、データベースサーバー側に格納しておく必要があります。

- ④ 旧プライマリーサーバーを、新プライマリーサーバーのデータを基に再構築してスタンバイとして組み込む
- ⑤ 各 PostgreSQL について、レプリケーションの同期先を変更する

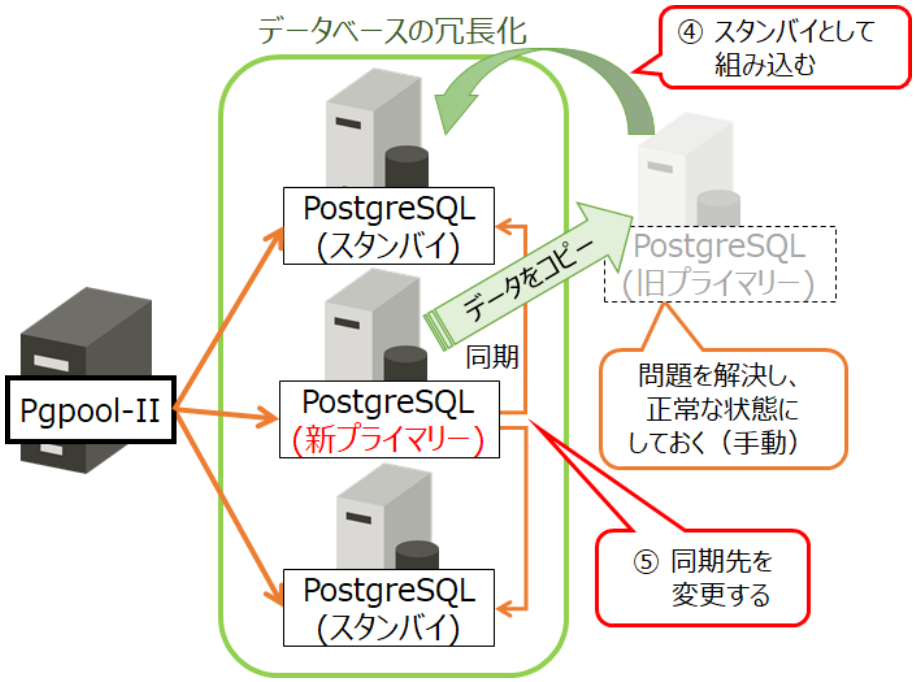

図 4 Pgpool-II のオンラインリカバリー機能

参考

Pgpool-II の「オンラインリカバリー機能」は、データベースサーバー側の PostgreSQL に対して、拡張機能としてインストールしておく必要があります。

1.3 Watchdog

システム全体の高可用を実現するためには、Pgpool-II 自身を冗長化する必要があります。その冗長化に必要な機能が「Watchdog 機能」です。「Watchdog 機能」は、複数の Pgpool-II をアクティブ / スタンバイ構成で相互に連携させ、互いに、死活監視と、サーバー情報（ホスト名、ポート番号、Pgpool-II のステータス、仮想 IP 情報、起動時間）を共有します。サービスを提供する Pgpool-II（アクティブ）に障害が発生した場合は、自律的に、Pgpool-II（スタンバイ）がそれを検知してフェイルオーバーします。その際、新しい Pgpool-II（アクティブ）は仮想 IP インターフェースを立ち上げ、以前の Pgpool-II（アクティブ）は、仮想 IP インターフェースを停止します。これにより、アプリケーション側からは、サーバーが切り替わっても同じ IP アドレスで Pgpool-II を使うことができます。また、「Watchdog 機能」を使うことで、各 Pgpool-II は連携・協調して、データベースサーバーの監視やフェイルオーバー指示を行います。その際、Pgpool-II（アクティブ）が取りまとめ役として動作します。
これらの処理は、Pgpool-II の設定ファイルの該当パラメーターを設定することで動作させることができます。

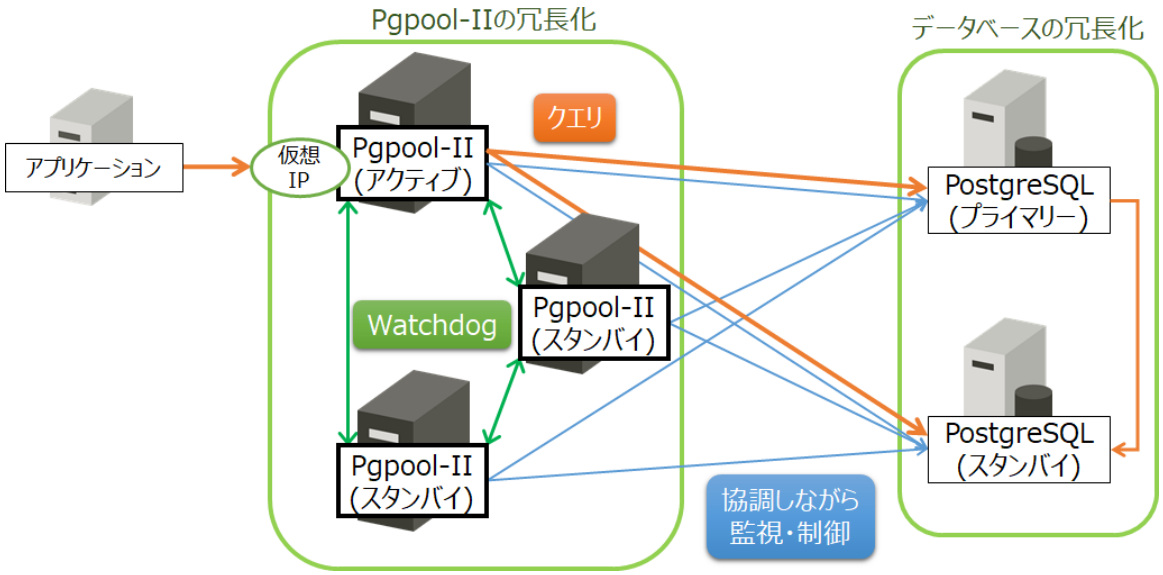

図 5 Pgpool-II の Watchdog 機能を利用した構成

冗長化構成において「Pgpool-II のスプリットブレイン」と「データベース（PostgreSQL）のスプリットブレイン」を考慮する必要があります。スプリットブレインとは、アクティブなサーバーが複数存在してしまう状態を指します。この状態でデータの更新作業を行うと、データの不整合が起こり、復旧が非常に困難な状態になります。Pgpool-II では、これらの問題を回避するために、Pgpool-II を 3 台以上、かつ、奇数台のサーバーで構成することを推奨しています。
それぞれのスプリットブレインが発生するタイミングの例は以下の通りです。

- **Pgpool-II のスプリットブレイン**
  Pgpool-II が 2 台構成のとき、Pgpool-II を結ぶネットワークのみに異常が発生（フェイルオーバーについては抑止されますが、Pgpool-II の連携・協調が動作しなくなります）
- **データベース（PostgreSQL）のスプリットブレイン**
  Pgpool-II が 2 台構成のとき、Pgpool-II（アクティブ）と PostgreSQL（プライマリー）の間を結ぶネットワークに異常が発生（2 台構成では、後述する、多数決による判定が行われません）

ここで、Pgpool-II が 3 台構成のときの「Watchdog 機能」が、データベースのスプリットブレインの発生を回避する例を見てみましょう。

① Pgpool-II（アクティブ）は、PostgreSQL（プライマリー）との間のネットワーク断線により、障害を検知する

このとき、Pgpool-II（アクティブ）は、PostgreSQL（プライマリー）が稼動しているかどうか判断できません。

② フェイルオーバーするかどうかの処理判断を、他の Pgpool-II（スタンバイ）にも投票させる

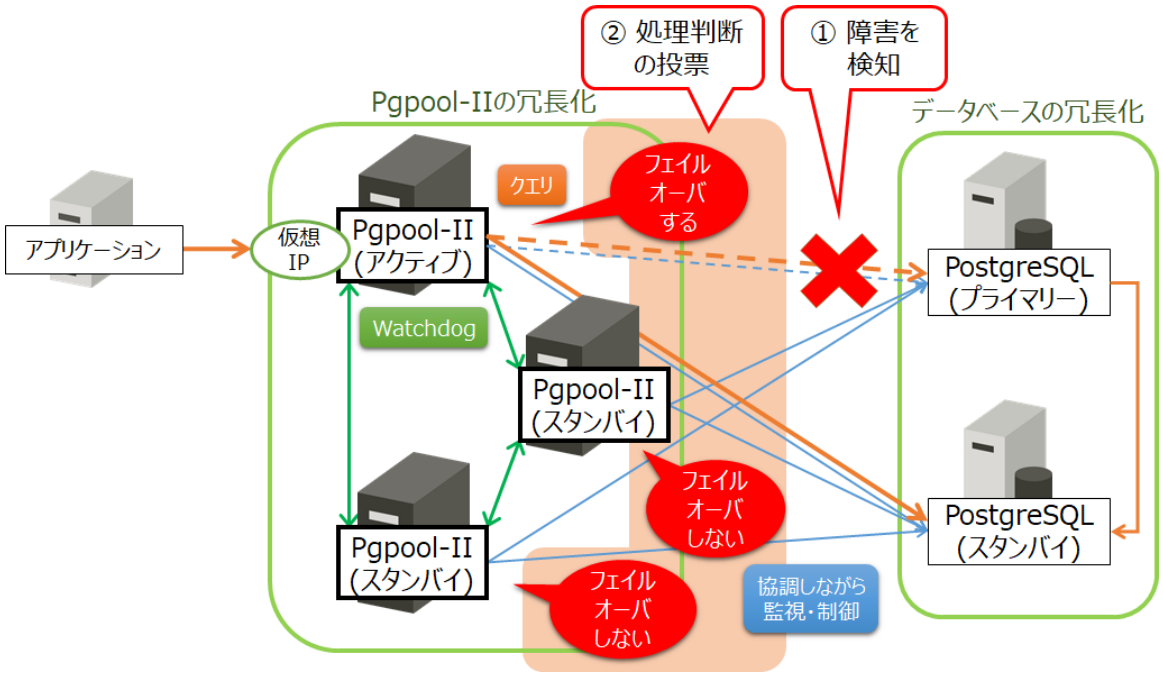

図 6 ネットワーク障害が発生したときの Watchdog 機能の動作（障害検知と投票）

③ 「フェイルオーバーする」という投票結果が過半数に達していないため、フェイルオーバーを行わない

ここで、スプリットブレインが回避されます。また、Pgpool-II（アクティブ）は、PostgreSQL（プライマリー）へのクエリ振り分けを止めます（隔離）。

④ アクティブの Pgpool-II で障害検知が行われた場合、アクティブ/スタンバイの切り替えを行う

この結果、アプリケーション側からの更新クエリが継続して実行できます。

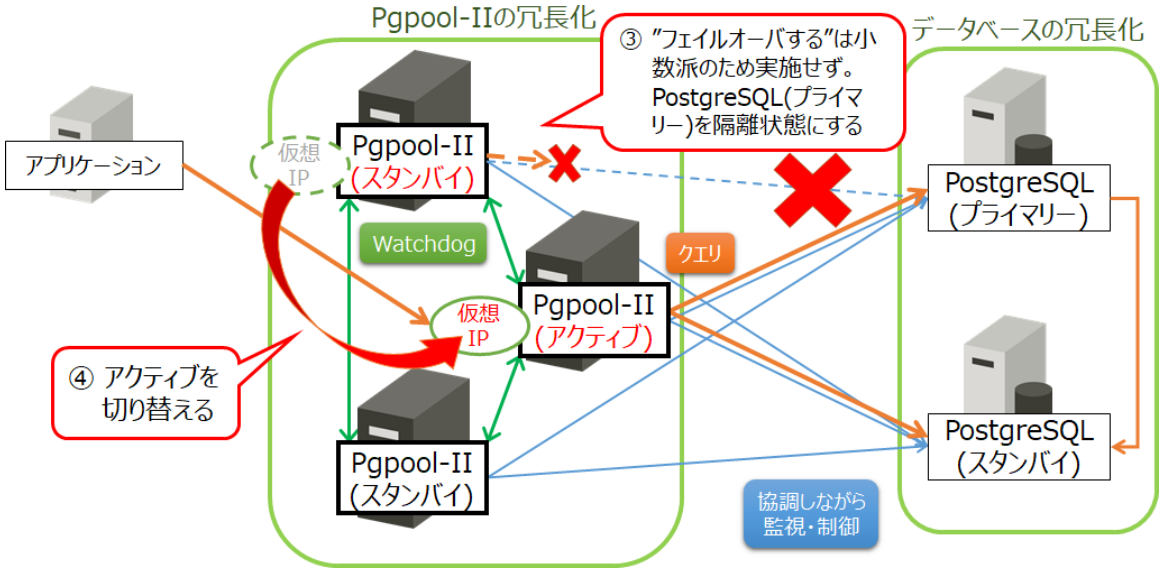

図 7 ネットワーク障害が発生したときの Watchdog 機能の動作（判断と対処）

2. Pgpool-II を利用したシステム構成

実際のビジネス利用を想定したシステム構成の検討と、簡単な動作確認を行います。

2.1 システム構成の検討

Pgpool-II を利用した実際のビジネス利用を想定し、以下の要件を満たしたシステム構成を考えてみます。

- スプリットブレインを考慮した高可用構成
- 将来的なスケールアウト
- 性能を考慮し、負荷分散機能とコネクションプーリング機能を利用

一般的に、ハードウェア、および、ソフトウェアを選定する際には、コストパフォーマンスを考慮し、シンプルで効率的な構成にします。そのために、まず検討すべきことは、Pgpool-II の配置場所です。以下に配置場所ごとの特徴（影響）を示します。

表 2 Pgpool-II の配置場所による影響比較

配置場所	サーバー負荷影響	ネットワーク影響	サーバーコスト（台数）
専用サーバー	他のソフトウェアの影響を受けない	アプリケーションと Pgpool-II 間、および、Pgpool-II とデータベース間のネットワークの影響を受ける	Pgpool-II を運用するサーバーが、別途必要（推奨は 3 台以上、かつ、奇数台）
データベースサーバー（同居）	同居先データベースの性能影響を受けると共に同居先サーバーの冗長化や拡張性を考慮する必要がある	Pgpool-II とデータベース間を一部ローカル接続にできるため、この部分のネットワーク影響が少ない	Pgpool-II の同居先サーバーは 3 台以上を推奨しており、データベースサーバーの台数によっては、別途必要
アプリケーションサーバーや Web サーバー（同居）	同居先のソフトウェアの性能影響を受けると共に同居先サーバーの冗長化や拡張性を考慮する必要がある	アプリケーション（Web サーバー）と Pgpool-II 間をローカル接続にできるため、この部分のネットワーク影響を受けない	Pgpool-II の同居先サーバーは 3 台以上を推奨しており、アプリケーションサーバー（Web サーバー）の台数によっては、別途必要

始めに、Pgpool-II をデータベースサーバーに同居した場合の構成例を見てみましょう。なお、実際のシステム構成を想定し、クライアントからのリクエストが別途ロードバランサーなどで負荷分散され、3 台のアプリケーションサーバーでアプリケーションが並行処理されることとします。図中の「APL」はアプリケーション、「Pgpool」は Pgpool-II、「(A)」はアクティブ、「(P)」はプライマリー、「(S)」はスタンバイを意味しています。Pgpool-II 同士は Watchdog で協調動作させます。スケールアウトは、データベースサーバー3 に PostgreSQL（S）を追加、さらには、データベースサーバー4 台目以降を追加していくことで実現できます。
なお、この構成では、Pgpool-II（アクティブ）が動作するデータベースサーバーに負荷が集中するため、それを考慮する必要があります。

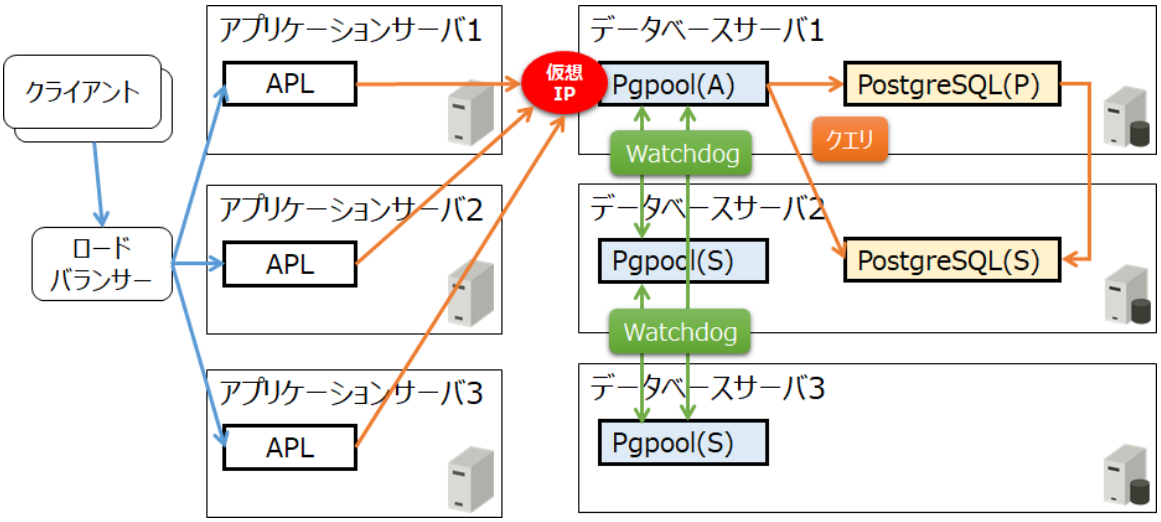

図 8 Pgpool-II を利用したシステム構成例（データベースサーバーに同居）

次に、Pgpool-II をアプリケーションサーバーに同居した場合の構成例を見てみましょう。Pgpool-II は、マルチマスタ構成で動作させるため、固定 IP を割り当てて利用します。その結果、以下のメリットが得られるようになります。

- Pgpool-II の負荷分散ができる
- すべての Pgpool-II のアクティブ / スタンバイにおいて、更新や参照クエリを受け付けることができる
- アプリケーションと Pgpool-II 間のネットワークオーバーヘッドを減らすことができる

なお、マルチマスタ構成においても、Pgpool-II（アクティブ）が取りまとめ役として動作してデータベースを制御します。また、この構成におけるスケールアウトについては、データベースサーバー3 台目以降を増やしていくことで実現できます。

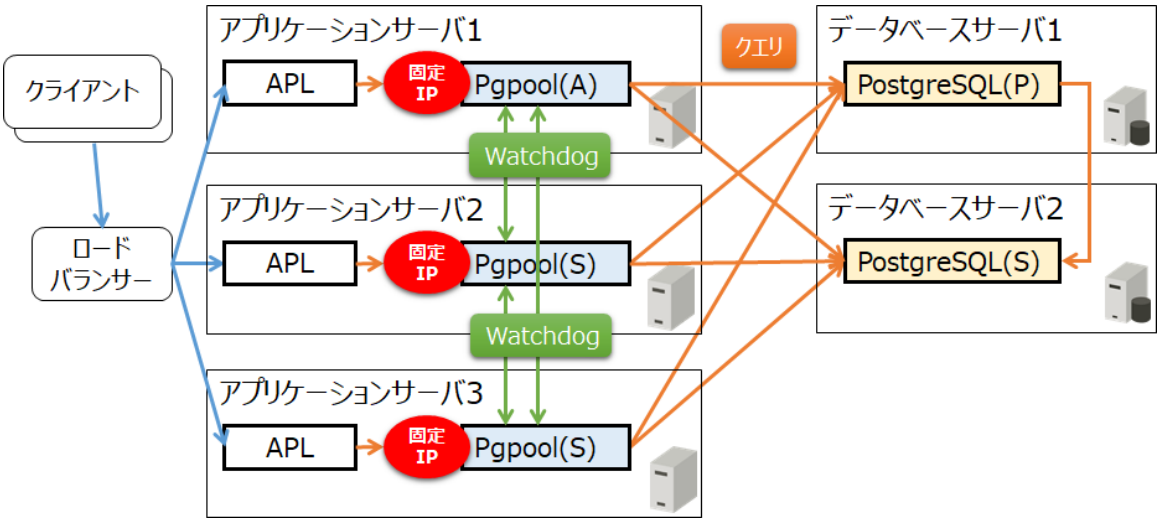

図 9 Pgpool-II を利用したシステム構成例（アプリケーションサーバーに同居）

2.2 動作確認

Pgpool-II をアプリケーションサーバーに同居した構成（図 9）において、自動フェイルオーバー、および、オンラインリカバリーの実際の動作について見てみましょう。なお、PostgreSQL と Pgpool-II はインストール、設定、起動が済んでいることを前提とします。

1. アプリケーションサーバー1 の Pgpool-II（アクティブ）に接続し、データベースサーバーの状態を確認します。なお、データベースサーバー1 のホスト名は host0、データベースサーバー2 のホスト名は host1 です。また、Pgpool-II（アクティブ）の存在するアプリケーションサーバー1 のホスト名は host5 です。

```
$ psql -h host5 -p 9999 -U postgres
postgres=# SHOW pool_nodes;
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay |
last_status_change
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------+------
---------------
 0       | host0    | 5432 | up     | 0.500000  | primary | 2          | false             | 0                 | 2019-
05-27 14:43:54
 1       | host1    | 5432 | up     | 0.500000  | standby | 2          | true              | 0                 | 2019-
05-27 14:43:54
(2 rows)
```

2. データベースサーバー1（host0）の PostgreSQL（プライマリー）を強制停止します。

```
$ pg_ctl -m immediate stop
```

3. アプリケーションサーバー1 から、再度、データベースサーバーの状態を確認します。1 度目の接続は失敗しますが、リセットが行われてデータベースへの接続が復旧するため、2 度目の接続は成功します。データベースのサーバーの状態を見ると、host0 の「status」が「down」状態になり、また、「role」の状態が入れ替わっていることから、host1 に自動的にフェイルオーバーし、プライマリーに昇格したことがわかります。

```
postgres=# SHOW pool_nodes;
server closed the connection unexpectedly
    This probably means the server terminated abnormally
    before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
postgres=# SHOW pool_nodes;
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay |
last_status_change
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------+------
---------------
 0       | host0    | 5432 | down   | 0.500000  | standby | 2          | false             | 0                 | 2019-
05-27 14:47:20
 1       | host1    | 5432 | up     | 0.500000  | primary | 2          | true              | 0                 | 2019-
05-27 14:47:20
(2 rows)
```

4. アプリケーションサーバー1 から、データベース 1（host0）に対するオンラインリカバリーコマンドを実行します。正常終了したことを確認します。

```
$ pcp_recovery_node -h host5 -p 9898 -U postgres -n 0
Password:  （パスワードを入力）
pcp_recovery_node -- Command Successful
```


5. アプリケーションサーバー1 から、再度、データベースサーバーの状態を確認します。オンラインリカバリー後も、1 度目のデータベースへの接続のときにリセットされ、2 度目の接続は成功します。host0 の「status」が「up」状態になったことから、host0 がスタンバイとして復旧したことがわかります。

```
postgres=# SHOW pool_nodes;
ERROR:  connection terminated due to online recovery
DETAIL:  child connection forced to terminate due to client_idle_limitis:-1
server closed the connection unexpectedly
    This probably means the server terminated abnormally
    before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
postgres=# SHOW pool_nodes;
 node_id | hostname | port | status | lb_weight |  role   | select_cnt | load_balance_node | replication_delay |
last_status_change
---------+----------+------+--------+-----------+---------+------------+-------------------+-------------------+------
---------------
 0       | host0    | 5432 | up     | 0.500000  | standby | 2          | true              | 0                 | 2019-
05-27 14:50:13
 1       | host1    | 5432 | up     | 0.500000  | primary | 2          | false             | 0                 | 2019-
05-27 14:47:20
(2 rows)
```

3. 留意事項

Pgpool-II は、エンタープライズ利用に向けた、PostgreSQL の有用な周辺ツールであることが確認できました。そして、実際に適用できるシステム形態としては、同時接続数が多く、スケールアウトで対応するような、中・大規模システムに向いていることも確認できました。
なお、Pgpool-II を活用する上では、以下の留意点があります。

- 環境構築における設定項目が多く、また、用意すべきスクリプトは利用者が作成する必要があるため、設計や環境構築には、十分な検討と検証を行う必要があります。
- 負荷分散については、参照系のクエリが対象になるため、更新系のクエリが多いシステムについては、性能を向上させることができません。なお、データベースサーバー数を増やすほど、参照性能は向上しますが、更新性能は劣化する場合があります。

FUJITSU Software Enterprise Postgres（以降、Enterprise Postgres と略します）では、バージョン 10 から Pgpool-II を同梱していますが、Enterprise Postgres にも以下の機能があります。

- **アプリケーションの接続先切り替え機能**
  プライマリーサーバー、スタンバイサーバーを意識せずに、データベースへの接続を切替えるクライアント側の機能です。
- **データベース多重化機能**
  ストリーミングレプリケーション上で動作する運用機能です。データベースプロセス、ディスク、ネットワークなどの障害を検知し、自動切離しやフェイルオーバー機能などを提供します。

Pgpool-II と Enterprise Postgres が持つ、同様の機能は併用して利用することができませんのでご留意ください。なお、同様の機能を併用しない、Pgpool-II の「負荷分散機能」や「コネクションプーリング機能」と、Enterprise Postgres の「データベース多重化機能」を組み合わせた利用は可能です。

高可用を実現する上での、環境構築や検証については、難易度が高いことが想定されます。Enterprise Postgres に同梱している Pgpool-II は、富士通の 24 時間 365 日保守サポートにより、PostgreSQL 本体および Pgpool-II を含む周辺ツールのご質問、トラブル対応およびバグ修正にも迅速に対応しますので、ご利用を検討ください。

2020 年 2 月 14 日