SRA OSS, INC.

AWS 上で Pgpool-II を使ってみよう

PostgreSQL Conference Japan 2017
2017/11/3

SRA OSS, Inc. 日本支社
長田 悠吾

自己紹介

- 長田　悠吾　（ナガタ　ユウゴ）
  - SRA OSS, Inc. 日本支社
  - マーケティング部 PostgreSQL 技術グループ

- PostgreSQL の技術サポート・コンサルティング
- PostgreSQL 関連の技術調査・研究・開発
- Pgpool-II 開発者

本日の内容

- Amazon Web Service (AWS) での PostgreSQL の利用
  - 初期投資不要で、費用は使った分だけの従量制
  - RDS を使えば管理も AWS 任せできる

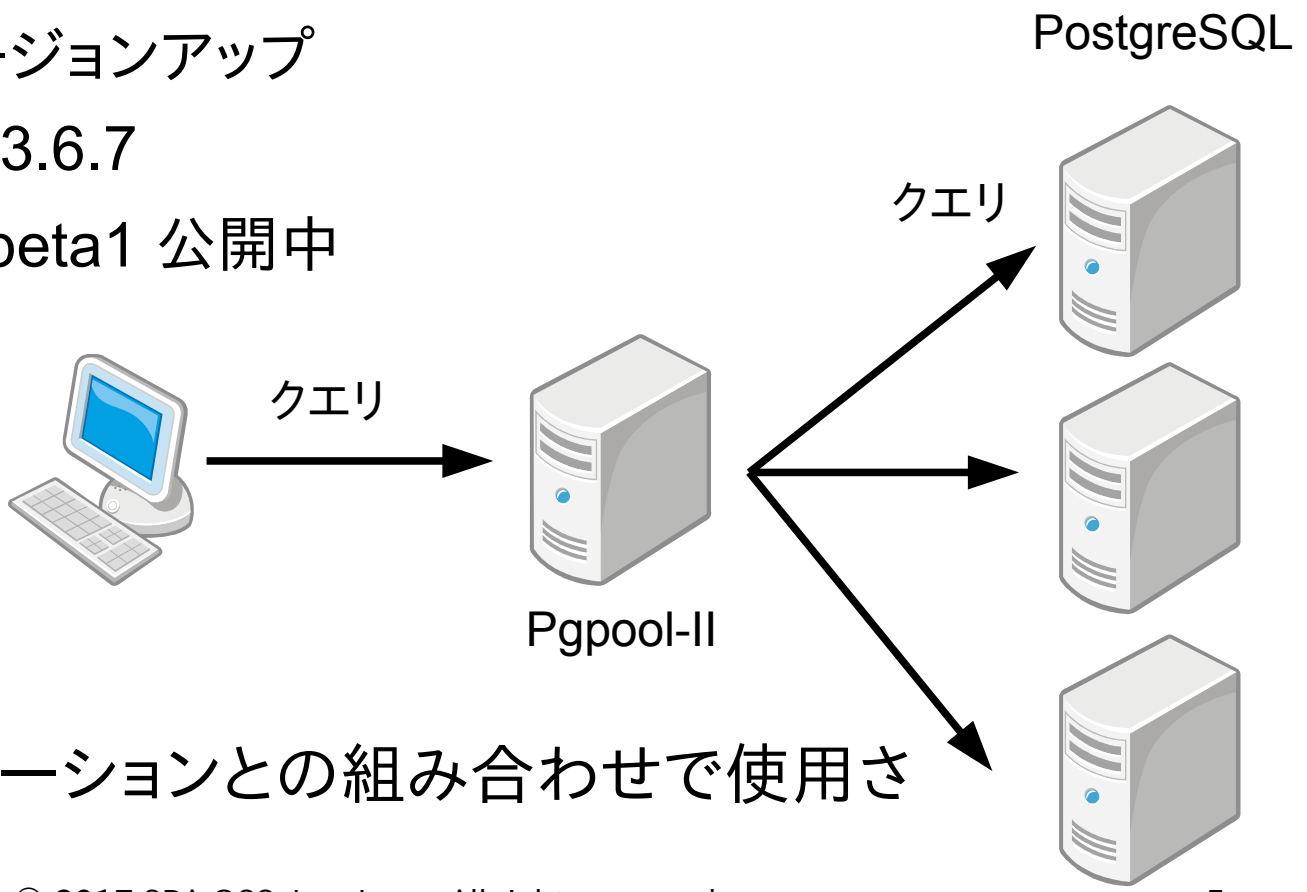
Pgpool-II も一緒に AWS で使ってみよう

- Pgpool-II の概要
- PostgreSQL on AWS
- Pgpool-II on AWS

Pgpool-II 概要

Pgpool-II とは

- アプリケーションとPostgreSQLの間に入って、クラスタリングの管理機能を提供するツール
- オープンソースソフトウェア（BSDライセンス）
  - 年1回のメジャーバージョンアップ
  - 現最新バージョンは 3.6.7
  - 次期バージョン 3.7 beta1 公開中

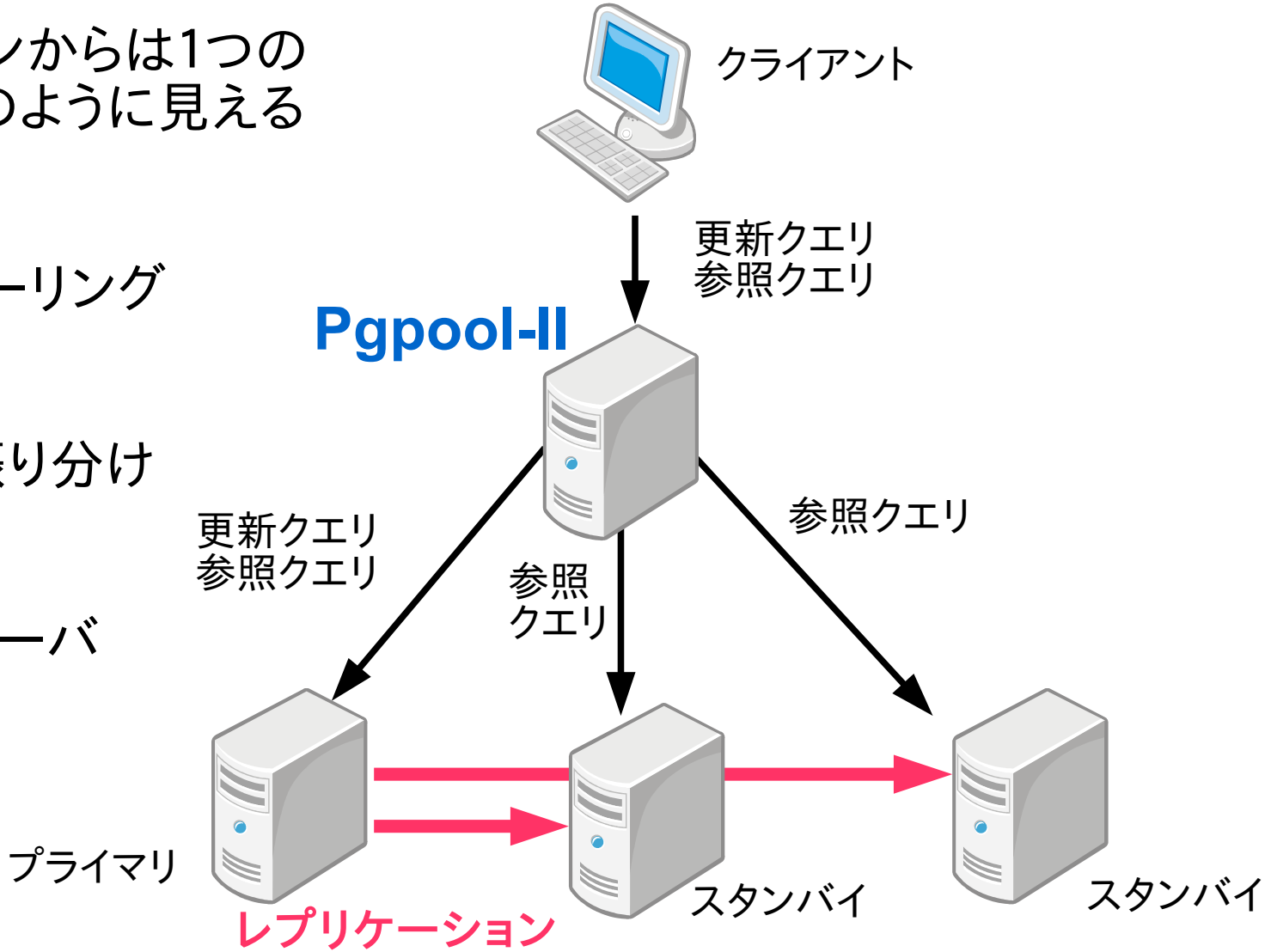


- ストリーミングレプリケーションとの組み合わせで使用されることが多い

Pgpool-II の機能

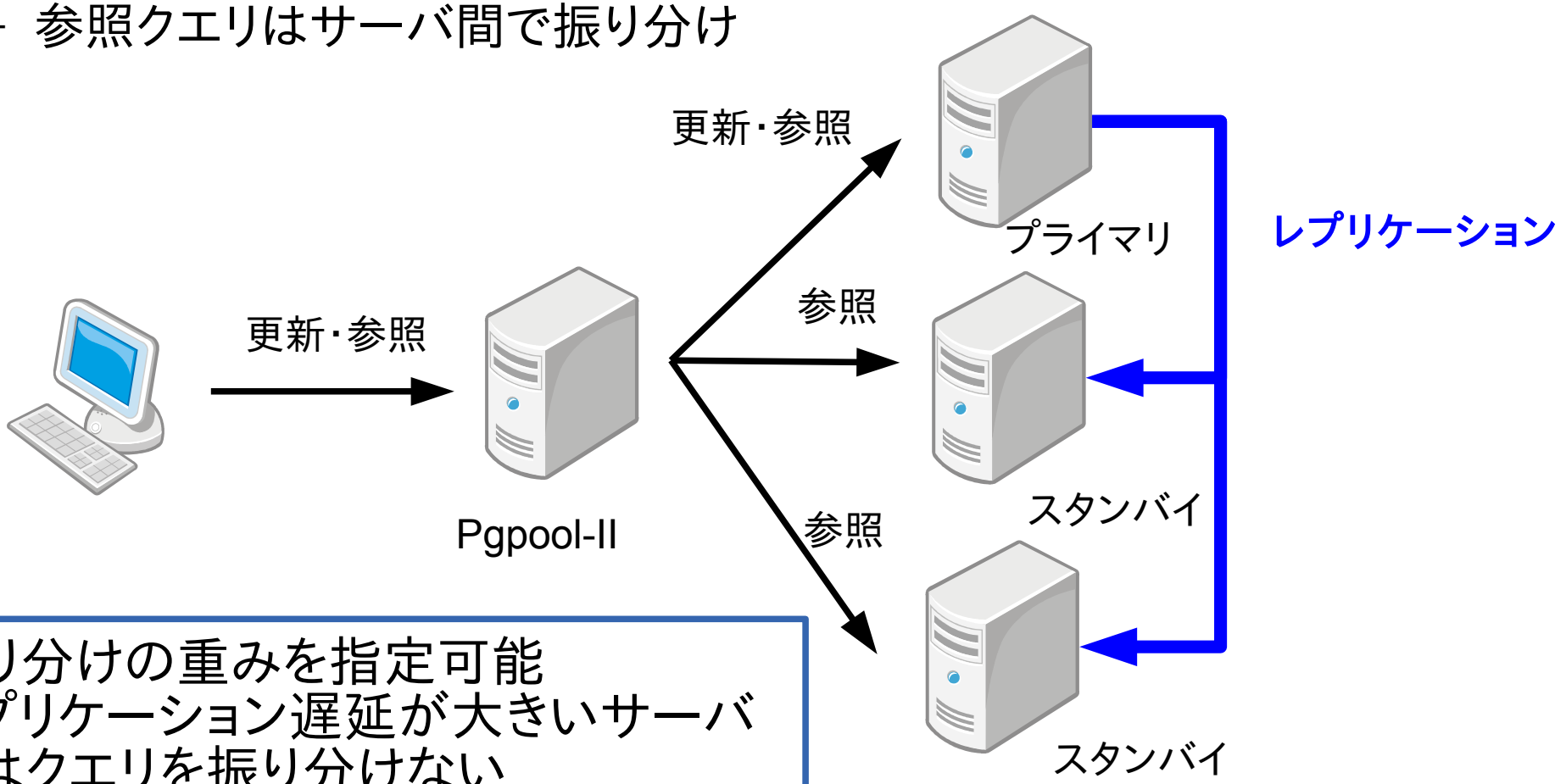
- アプリケーションからは1つの PostgreSQL のように見える

- コネクションプーリング
- 参照負荷分散
- クエリの自動振り分け
- ヘルスチェック
- 自動フェイルオーバ
- ・・・



参照負荷分散

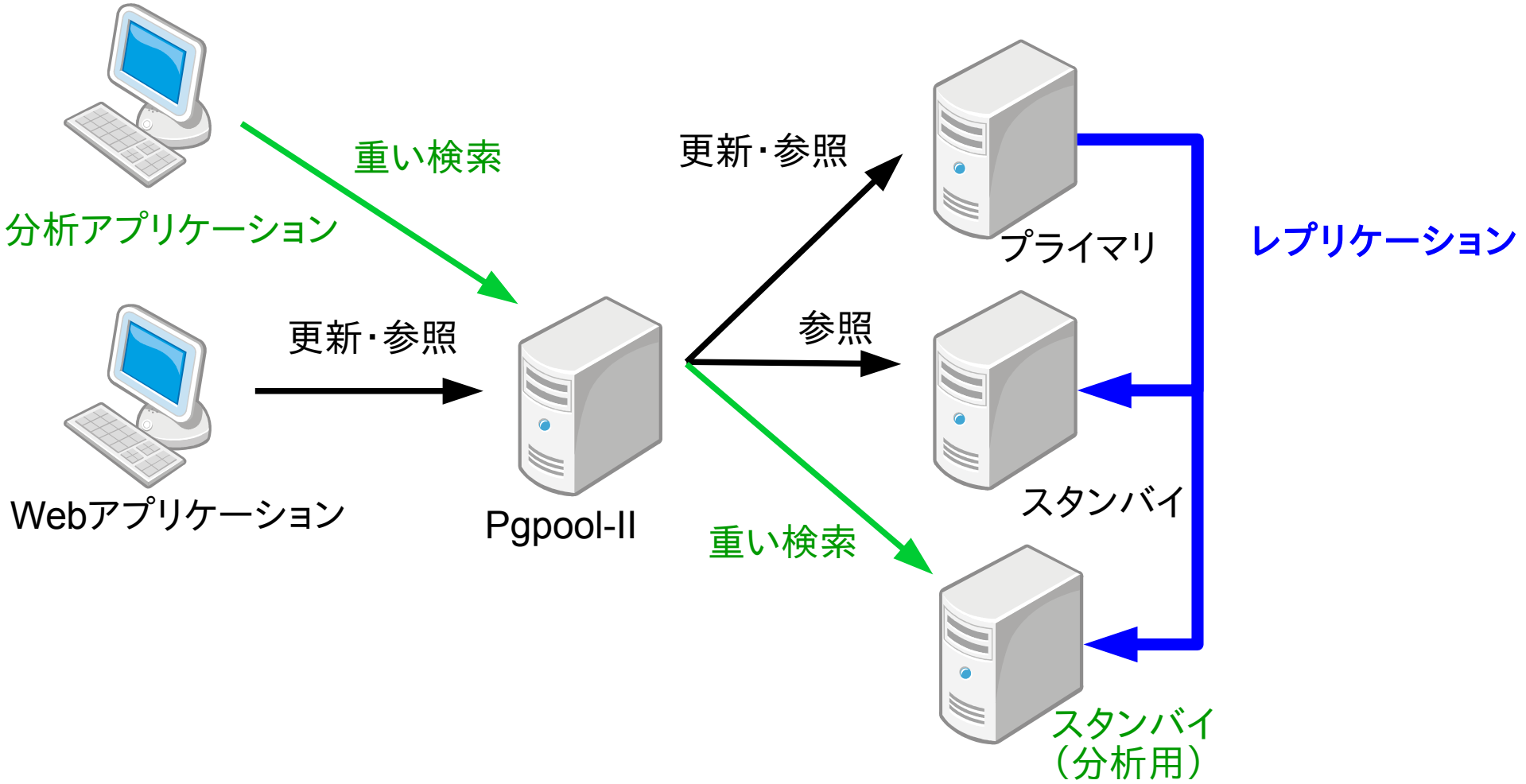
- クエリの自動振り分け
  - 更新クエリはプライマリサーバへ
  - 参照クエリはサーバ間で振り分け



- 振り分けの重みを指定可能
- レプリケーション遅延が大きいサーバにはクエリを振り分けない

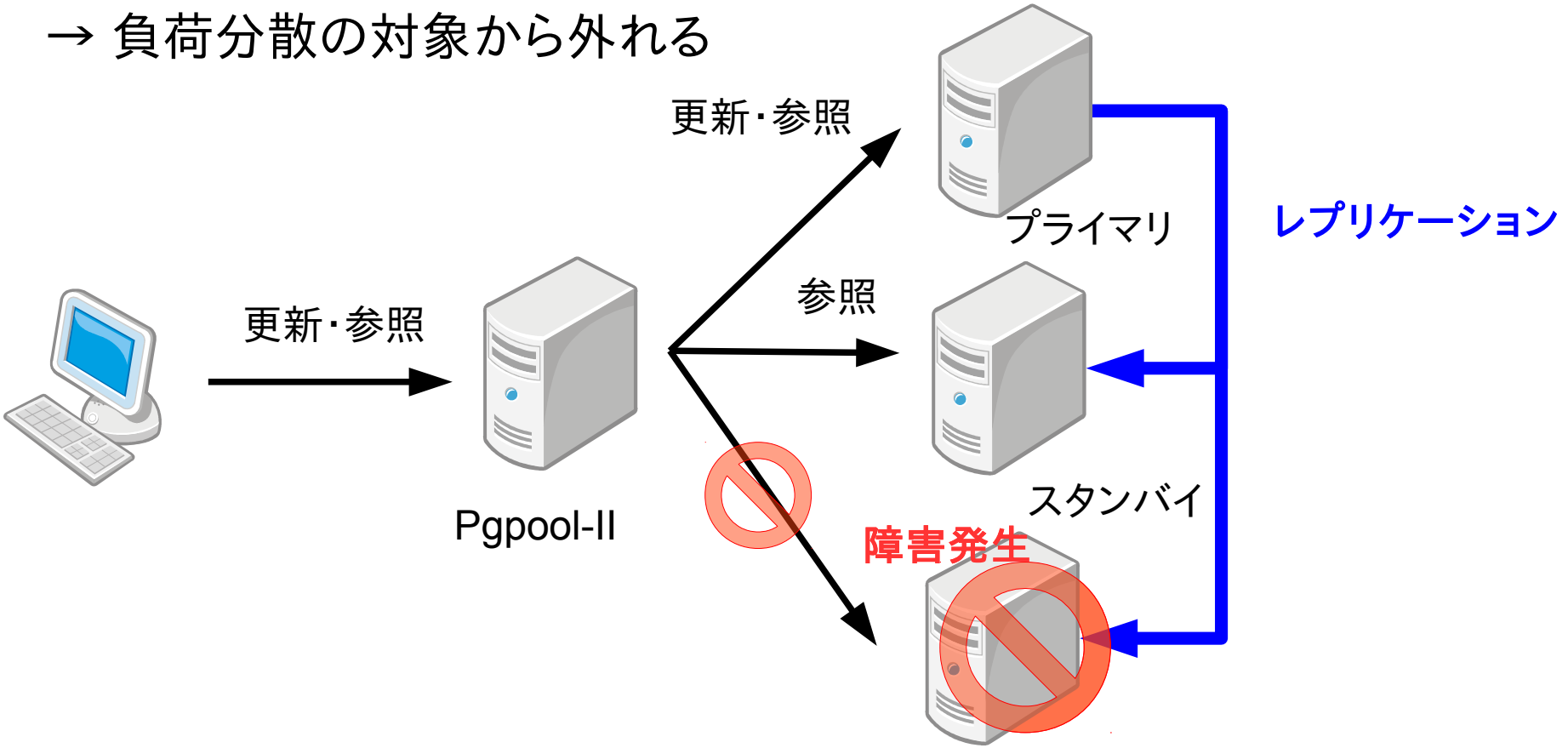
きめ細かな負荷分散（3.4～）

- アプリケーション名、DB名によって接続先が指定できる



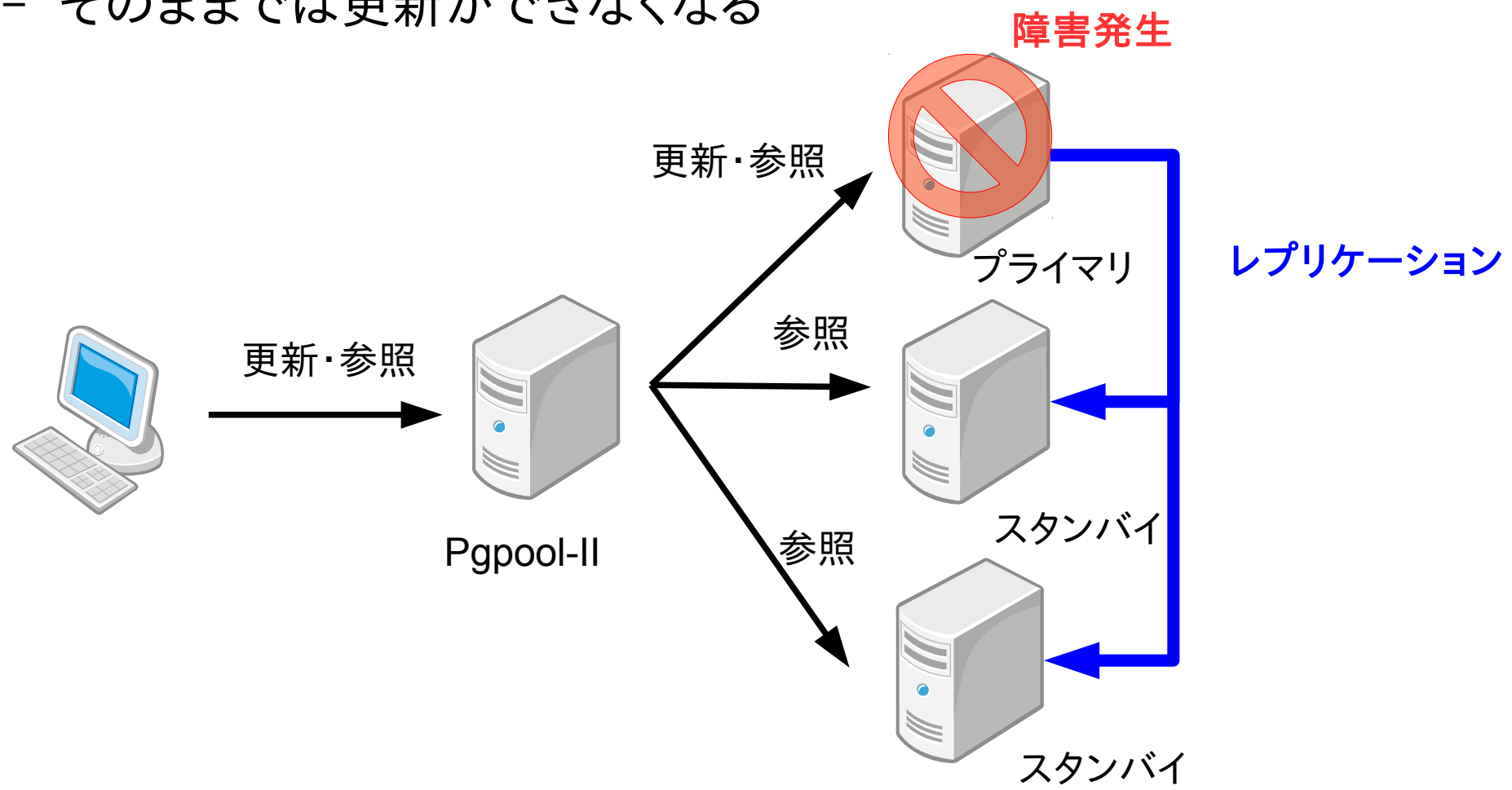
自動フェイルオーバ

- DBサーバの障害を自動検出（ヘルスチェック機能）
  - ダウンしたPostgreSQLを切り離す
    → 負荷分散の対象から外れる



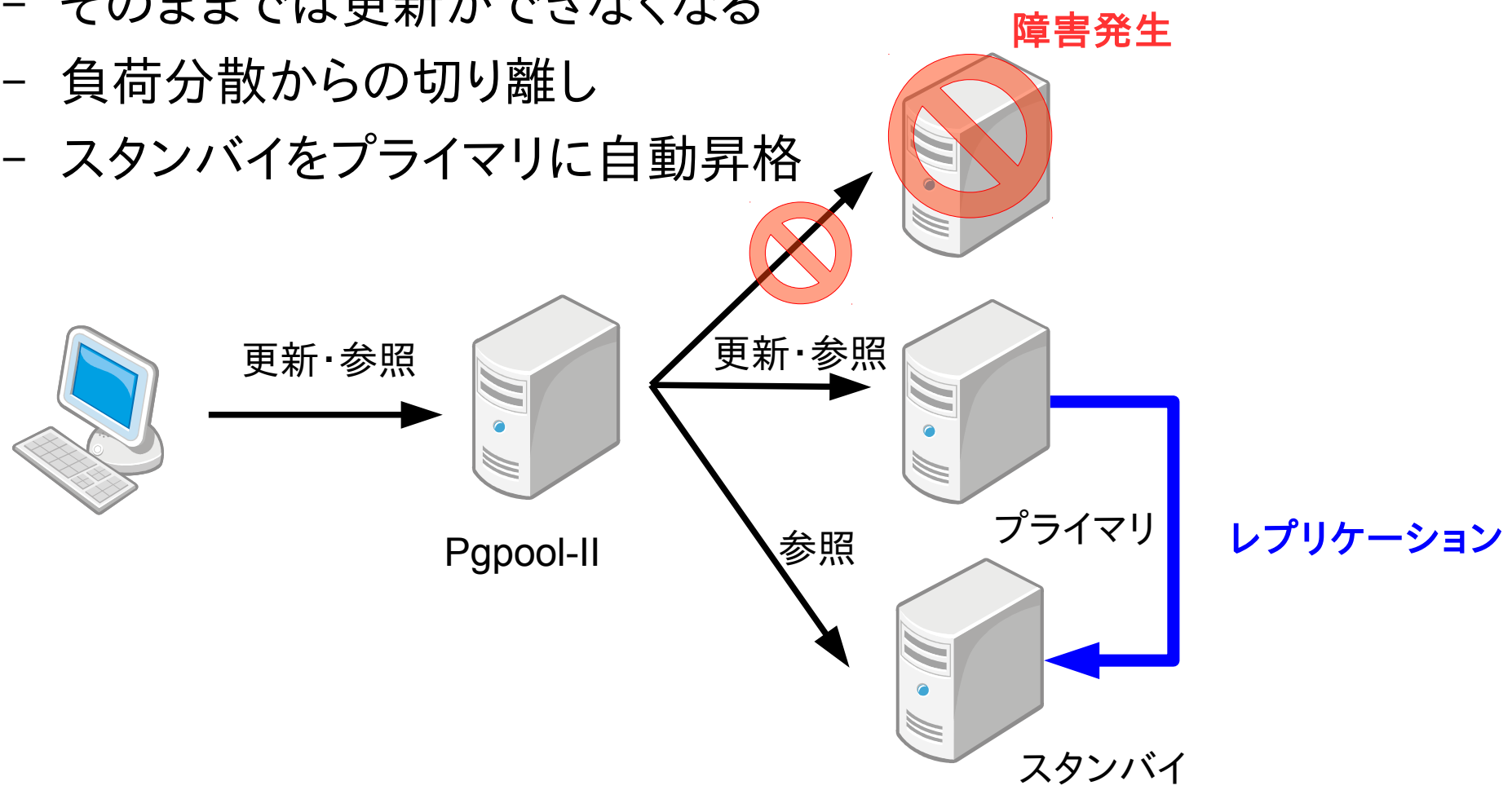
自動フェイルオーバ

- プライマリサーバに障害が発生した場合
  - そのままでは更新ができなくなる



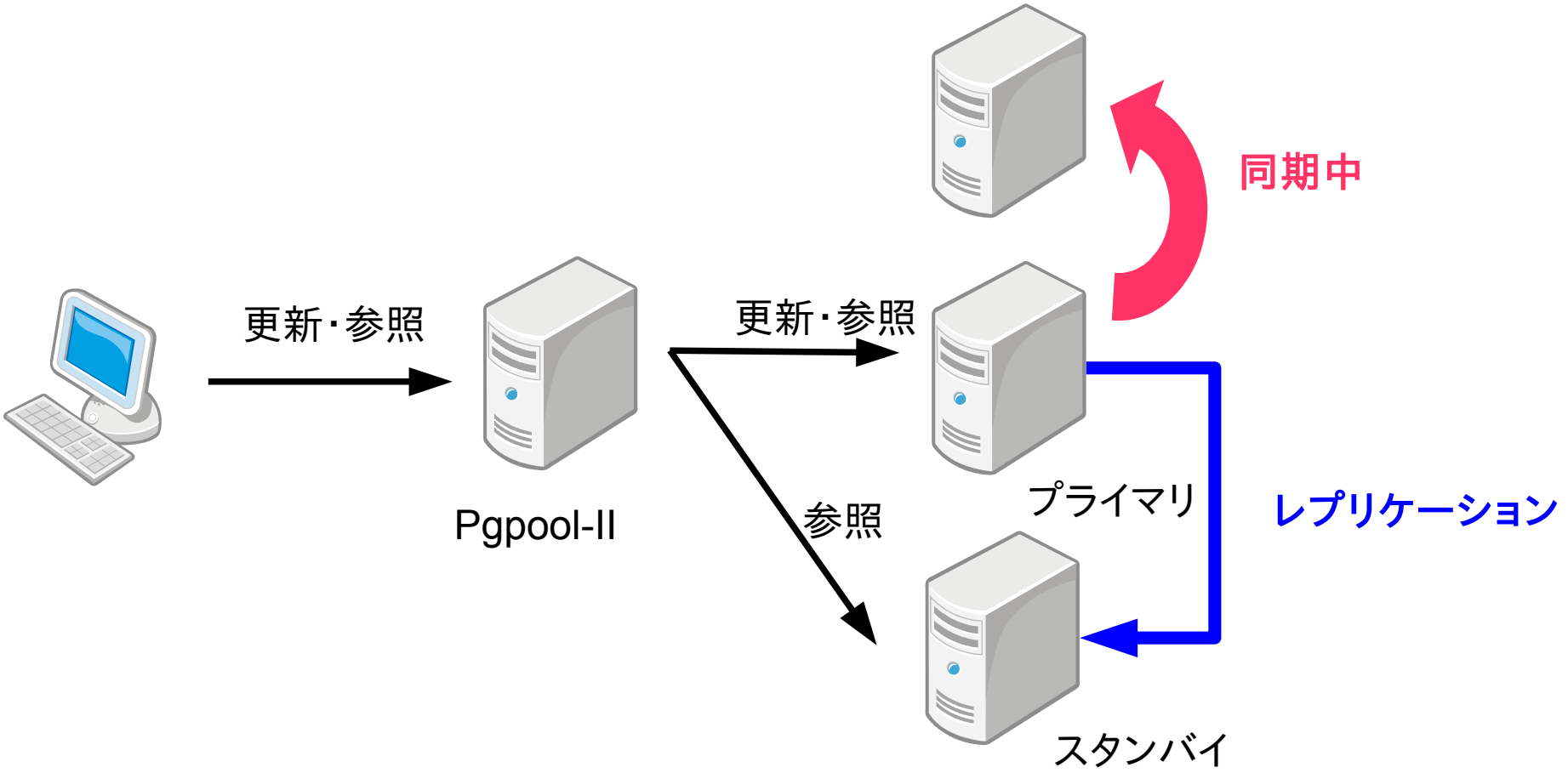
自動フェイルオーバ

- プライマリサーバに障害が発生した場合
  - そのままでは更新ができなくなる
  - 負荷分散からの切り離し
  - スタンバイをプライマリに自動昇格



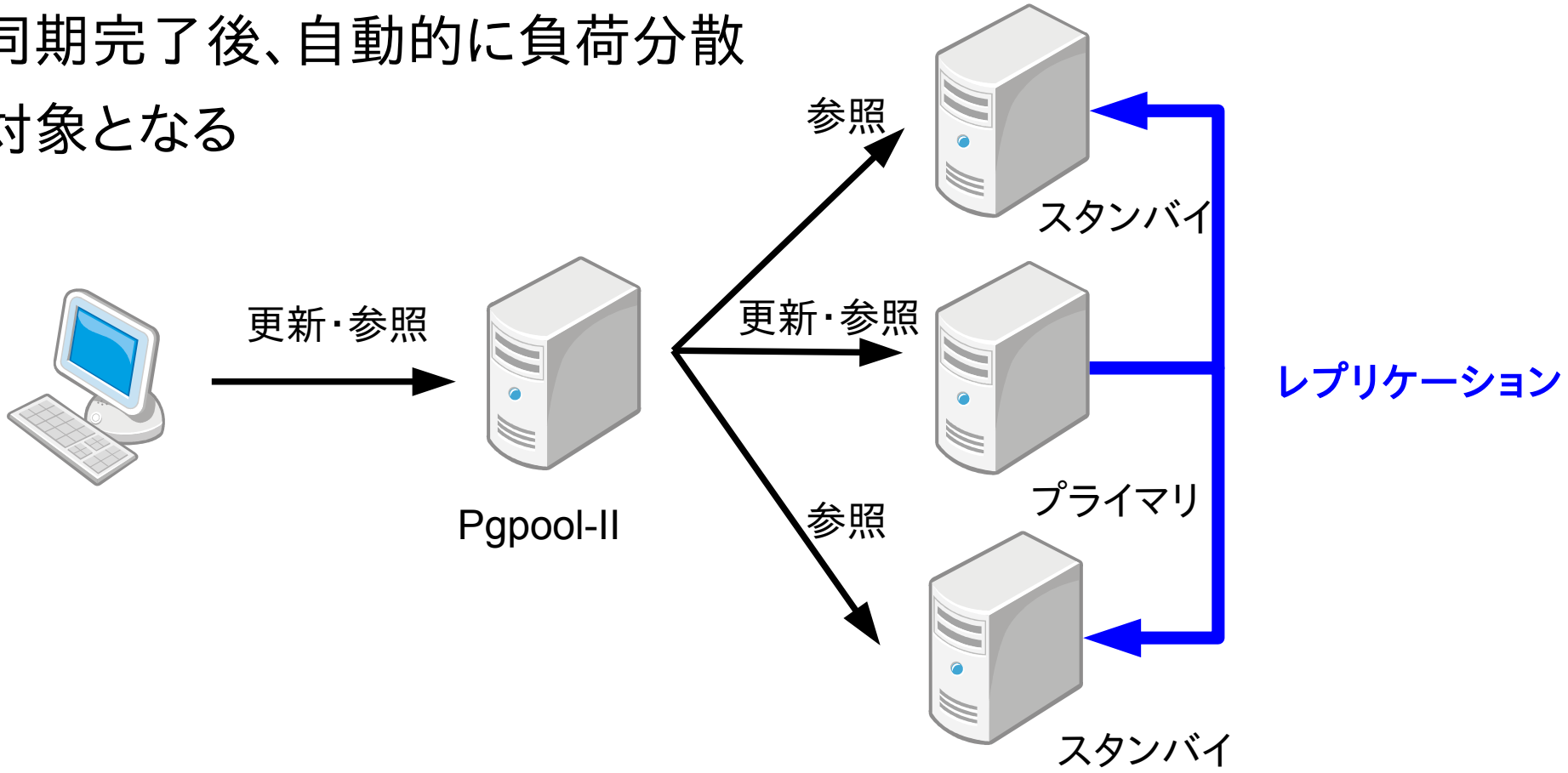
オンラインリカバリ

- ダウンしたスタンバイをプライマリに同期させる
- 同期中も更新が可能



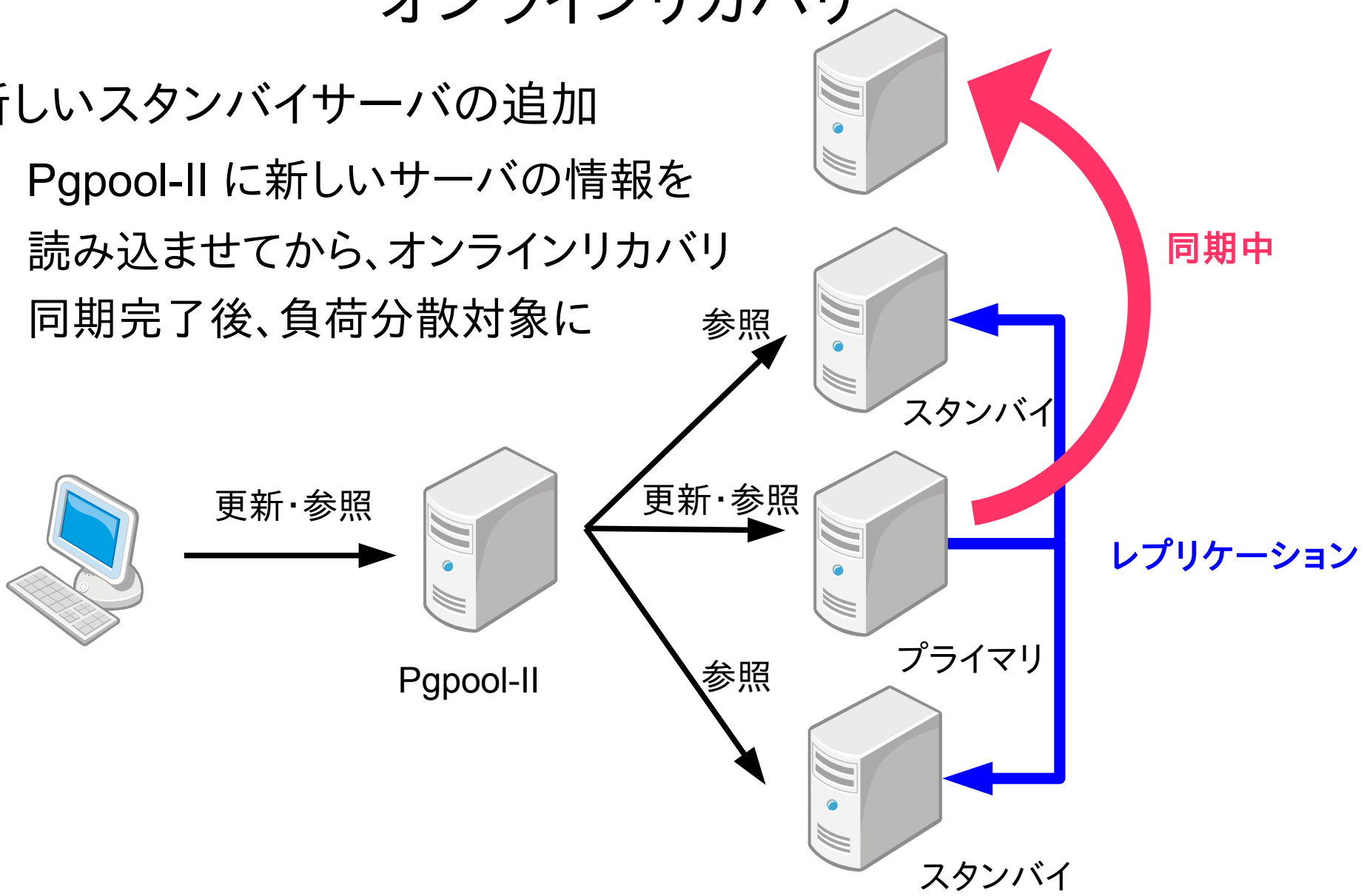
オンラインリカバリ

- ダウンしたスタンバイをプライマリに同期させる
- 同期中も更新可能
- 同期完了後、自動的に負荷分散対象となる



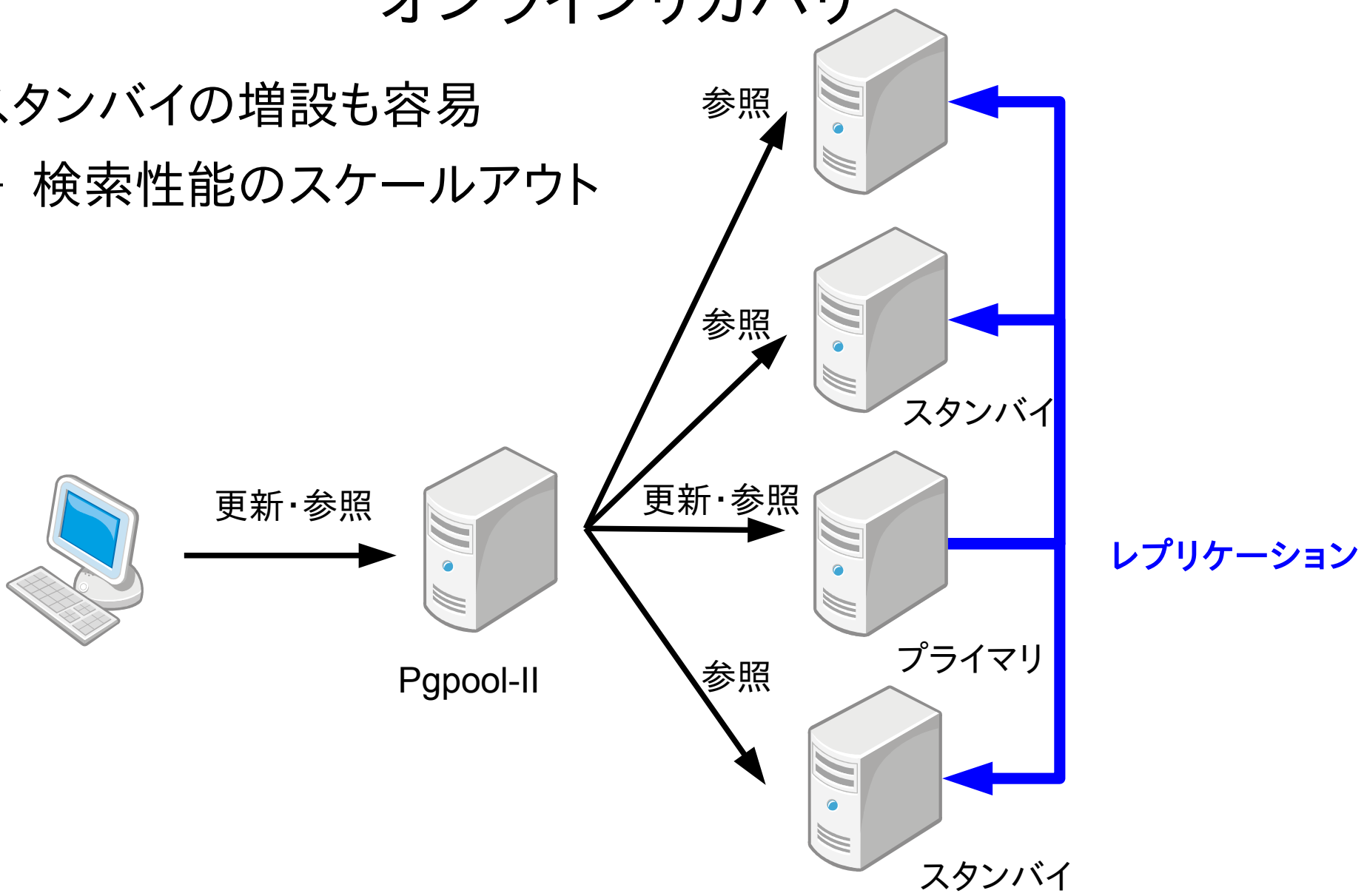
オンラインリカバリ

- 新しいスタンバイサーバの追加
  - Pgpool-II に新しいサーバの情報を読み込ませてから、オンラインリカバリ
  - 同期完了後、負荷分散対象に



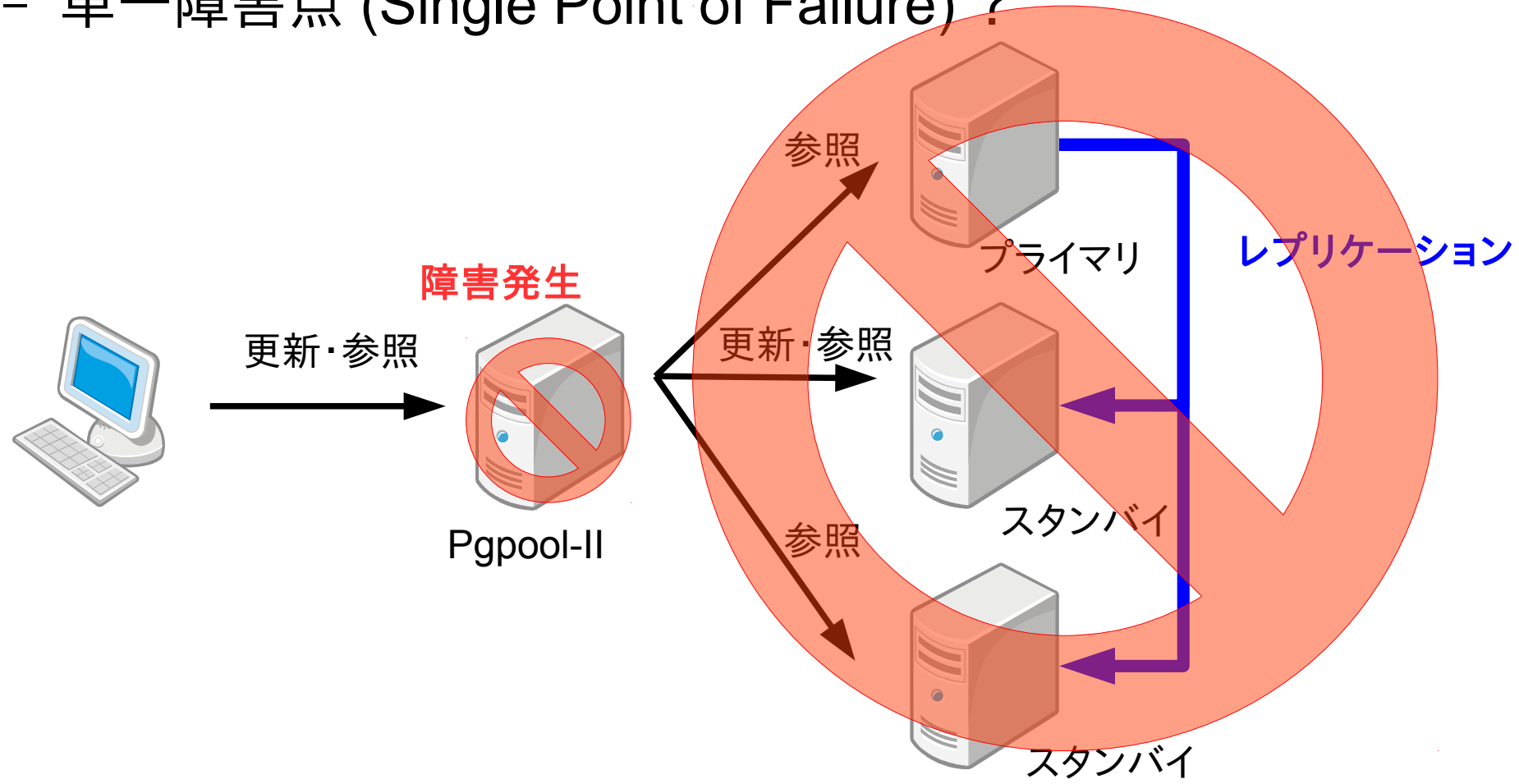
オンラインリカバリ

- スタンバイの増設も容易
  - 検索性能のスケールアウト



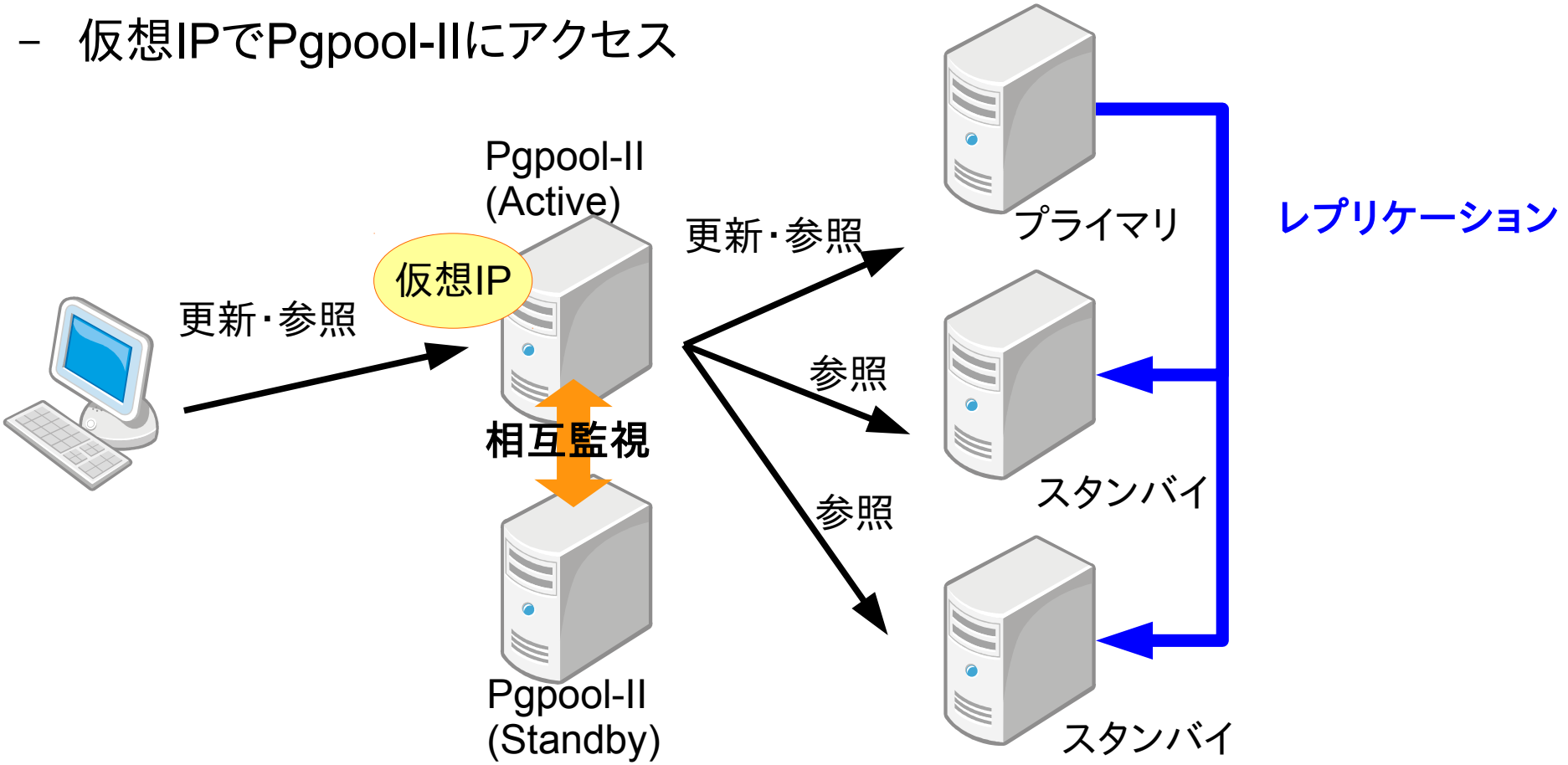
Pgpool-II の単一障害点回避

- もし、Pgpool-II に障害が発生したら?!
  - 単一障害点 (Single Point of Failure) ?



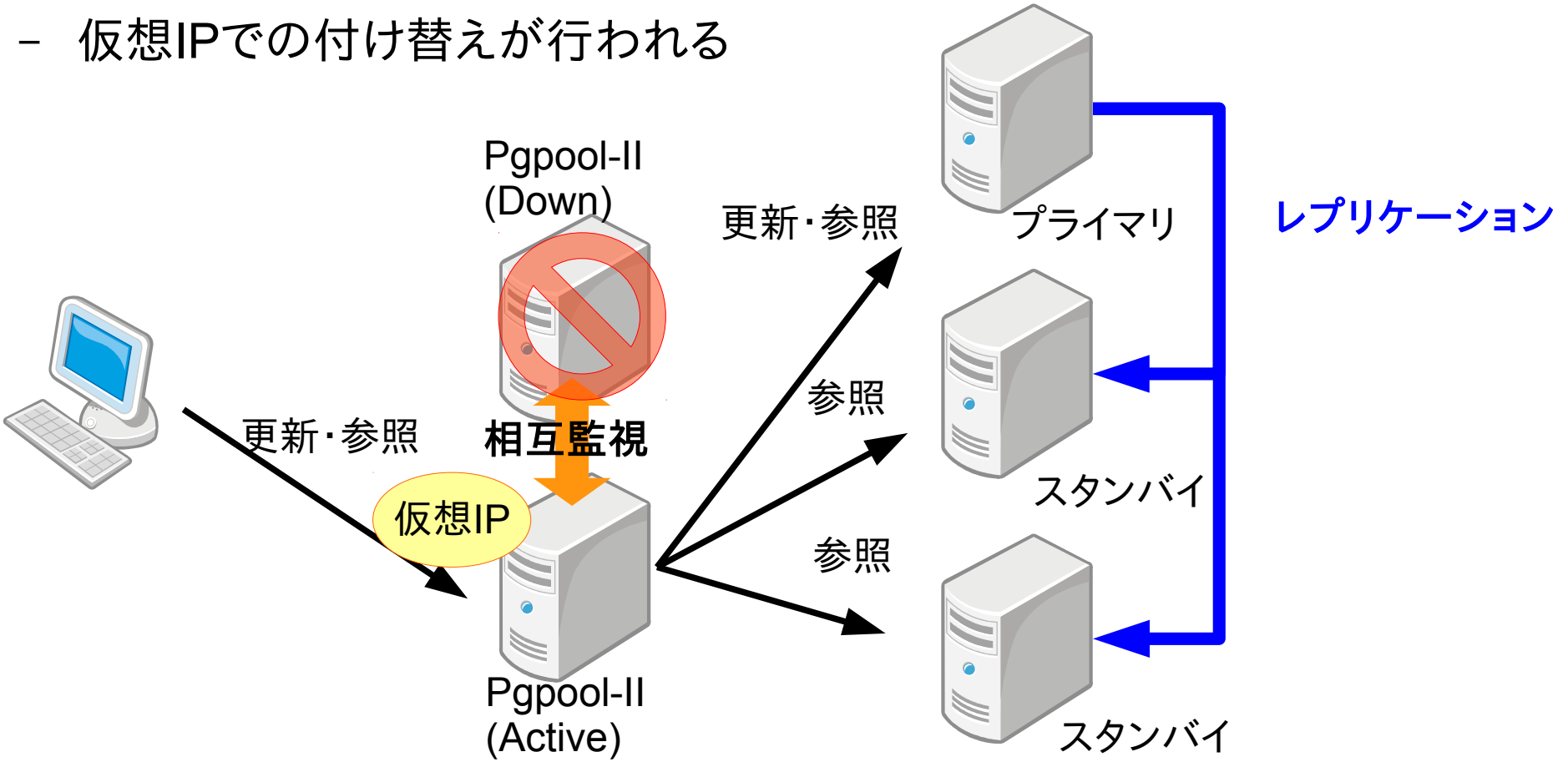
Watchdog

- Pgpool-II 組み込みのHA機能
  - Pgpool-II を Active/Standby 構成にする
  - 仮想IPでPgpool-IIにアクセス



Watchdog

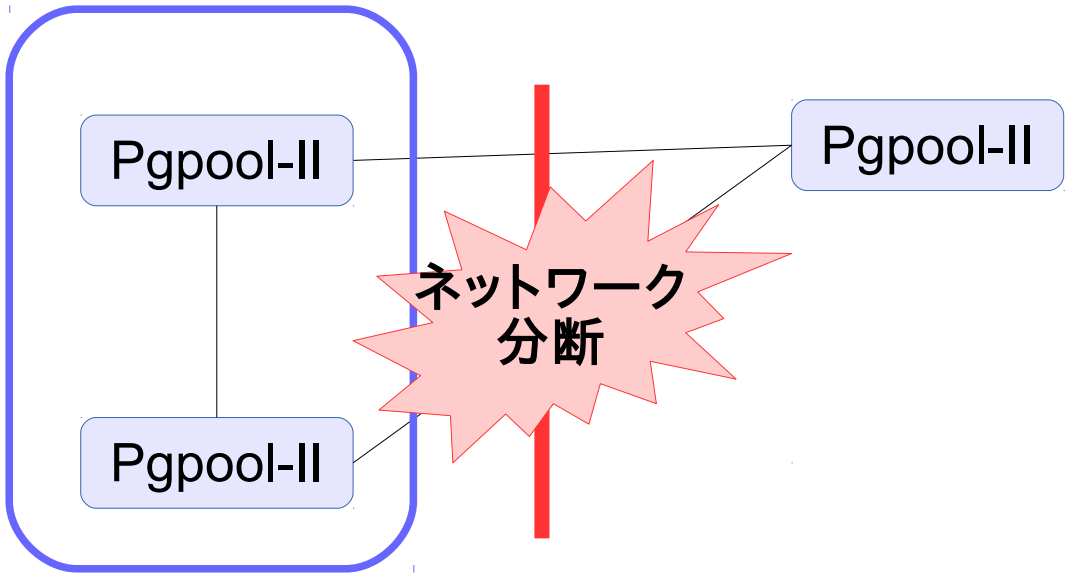
- Active Pgpool-II に障害発生すると・・・
  - Standby Pgpool-II が Active に昇格
  - 仮想IPでの付け替えが行われる



Watchdog のスプリットブレイン対応

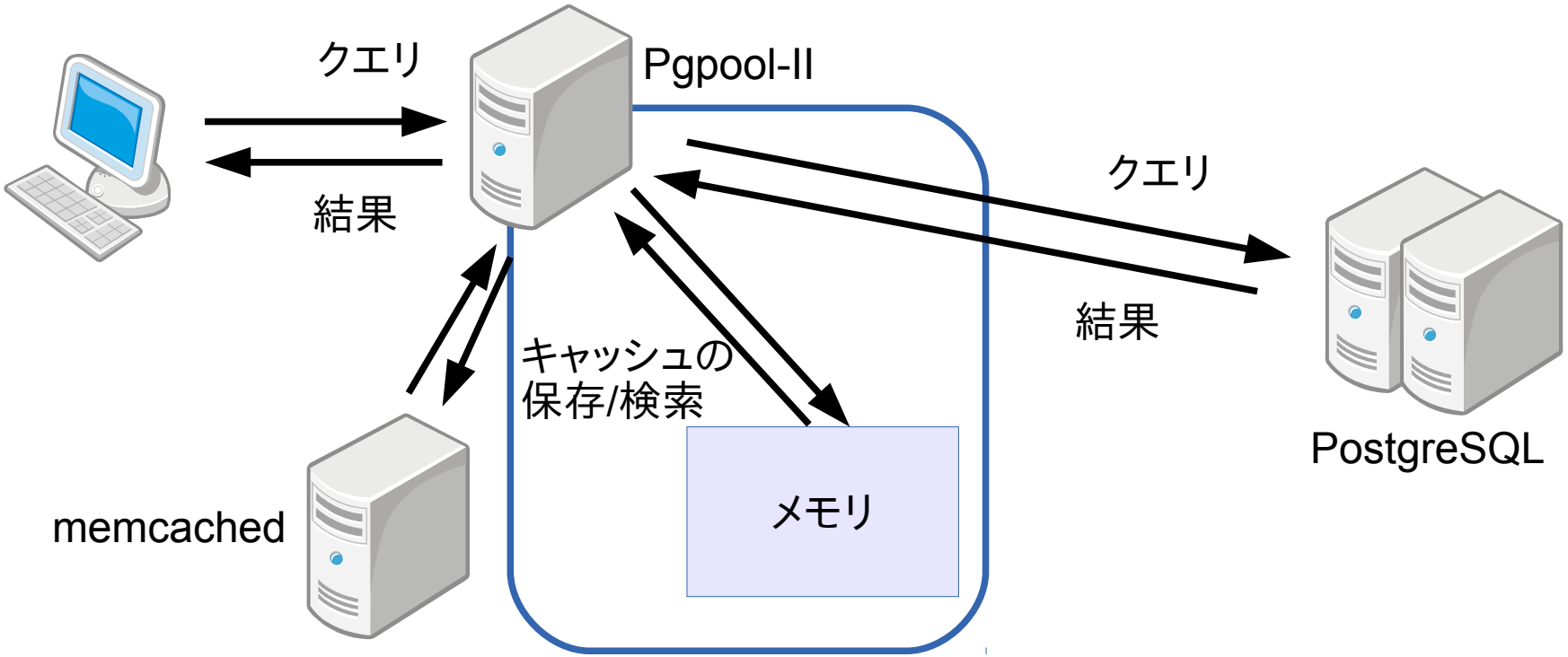
- スプリットブレイン問題
  - ネットワークが分離された時に、どの Pgpool-II がアクティブとなるべきか決められなくなる問題
  - Quorum（定足数）の充足を常に保証
    - クラスタに参加している全ノードのうち半数以上が自分と同じネットワークに属しているかどうか、をチェックする

アクティブはこちらのグループから選出



インメモリクエリキャッシュ

- SELECTクエリの結果をメモリ内にキャッシュする機能
  - 同じクエリが来たときに再利用する
  - DBへのアクセスが減り、応答速度が向上

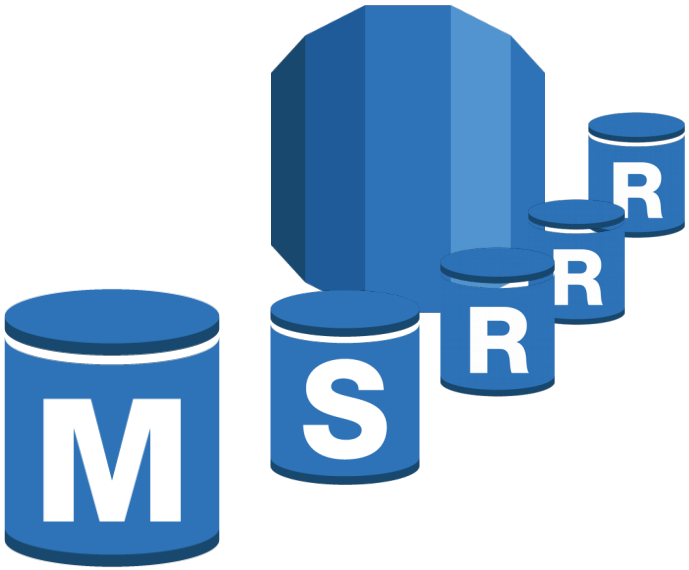


Pgpool-II 機能まとめ

- クラスタ管理機能
  - 参照負荷分散
  - クエリの自動振り分け
  - 自動フェイルオーバ
  - オンラインリカバリ

- その他の付加機能
  - Watchdog（= Pgpool-II 自体の高可用化）
  - インメモリキャッシュ

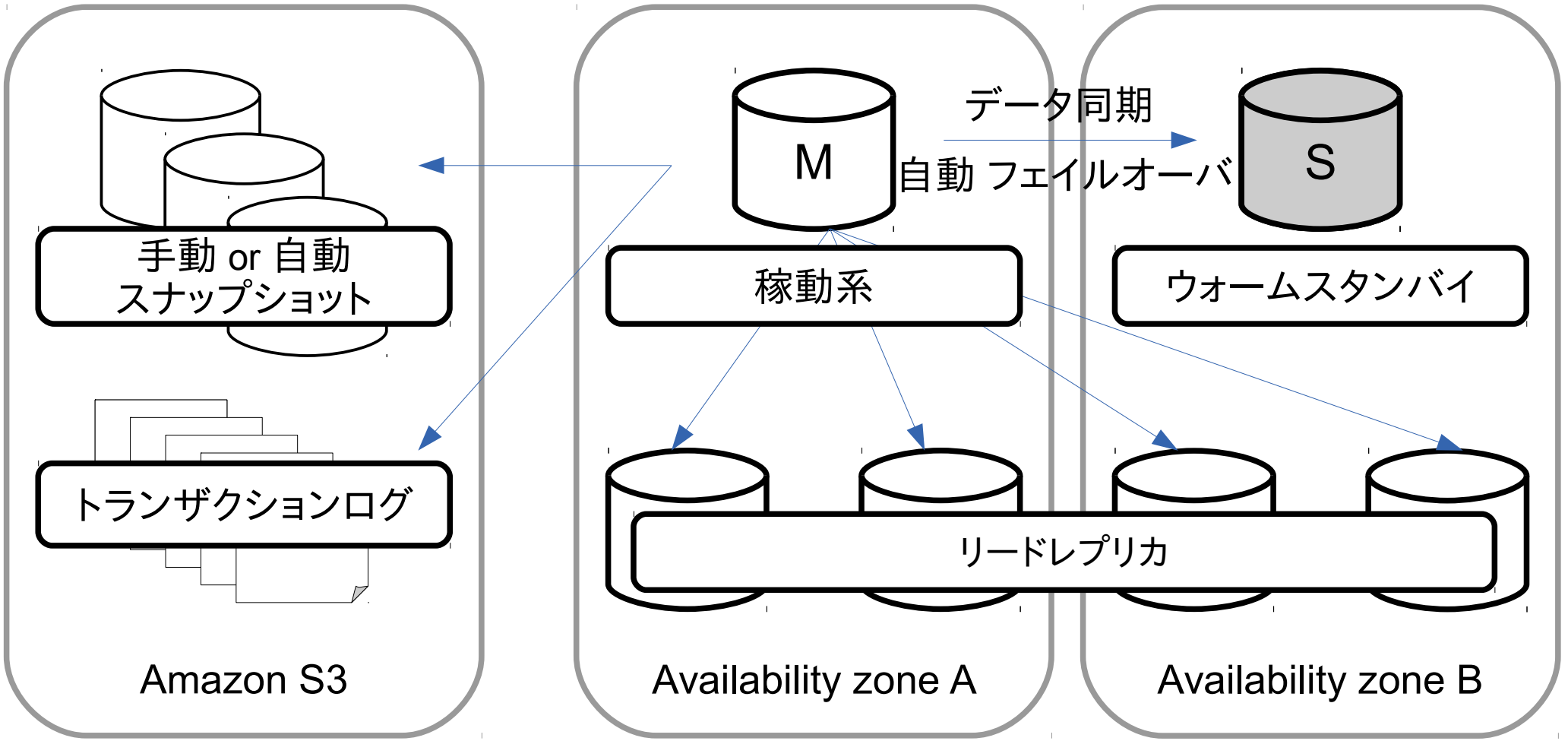
PostgreSQL を AWS で使う



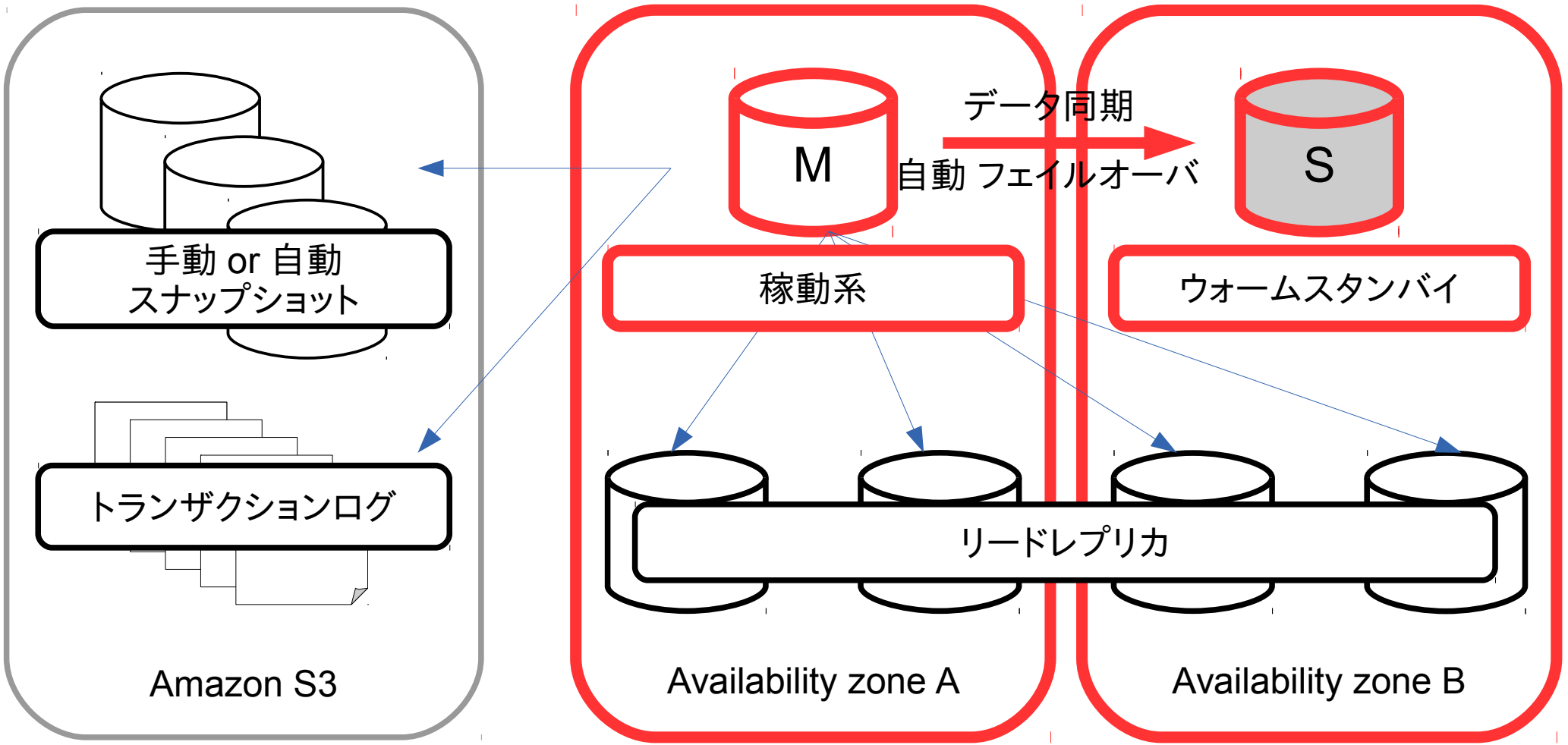
PostgreSQL on AWS の選択肢

- EC2
  - AWSが提供する仮想マシン
  - 通常のマシンと同じように PostgreSQL をインストールしてDBサーバを構築する
- RDS for PostgreSQL
  - 完全マネージド型の Relational Database サービス
- Aurora with PostgreSQL Compatibility も選べる
  - より高性能な完全マネージド型のRDBサービス

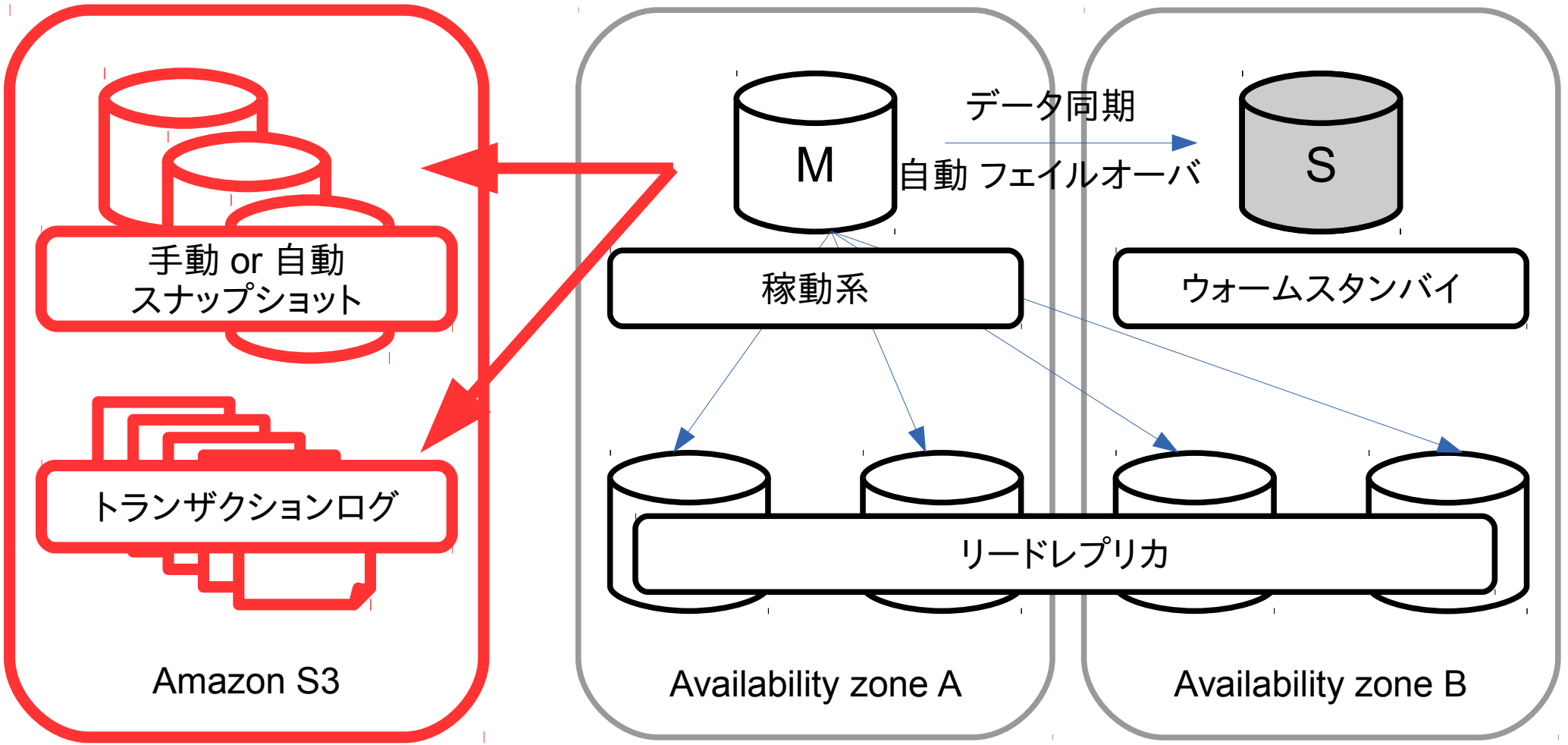
Amazon RDS for PostgreSQLのアーキテクチャ



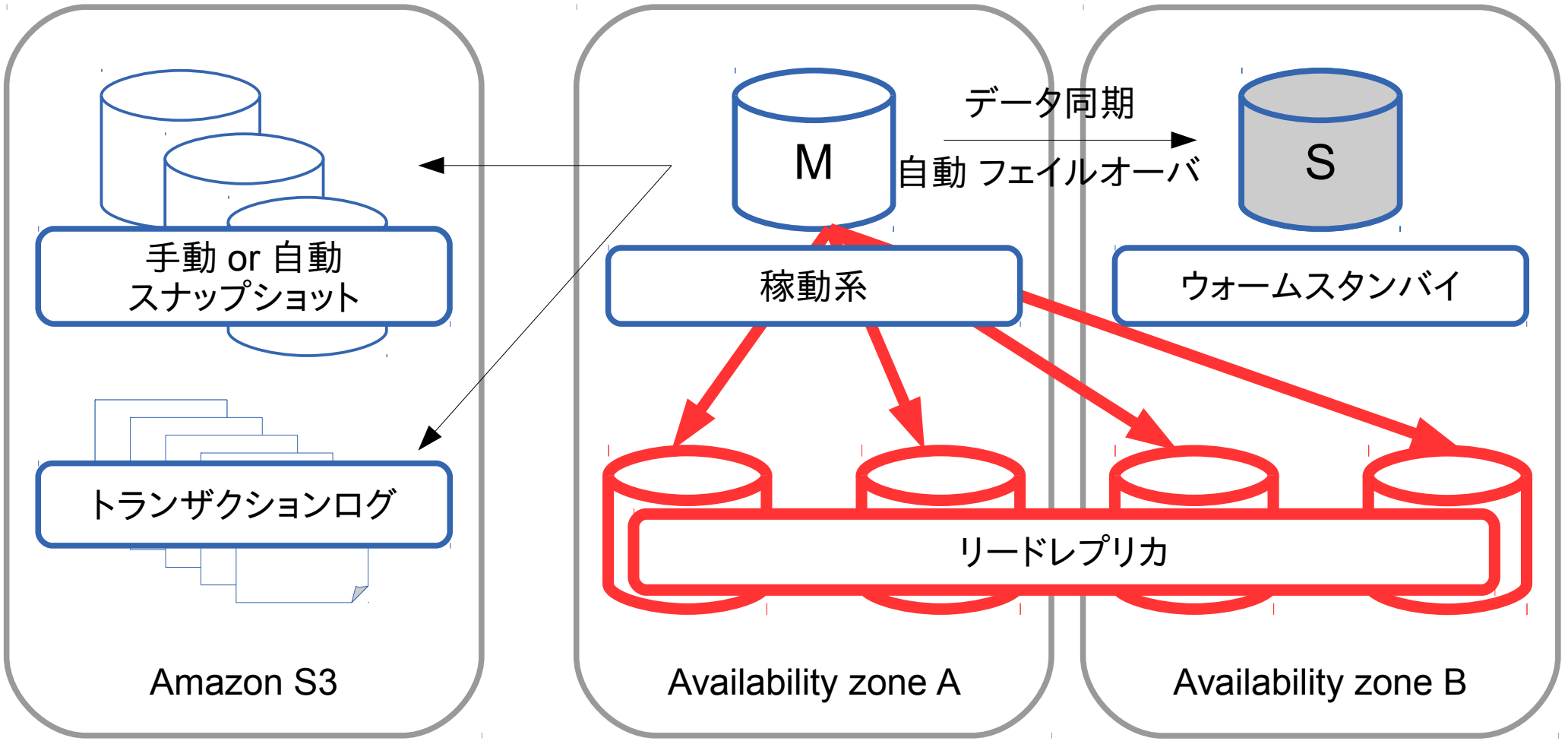
耐障害性と可用性の向上



バックアップとリカバリ / リストア

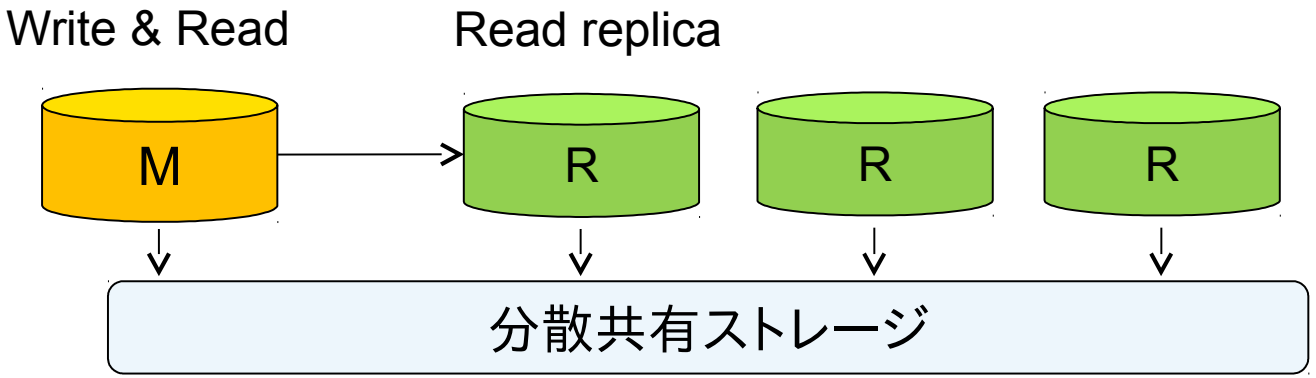


スケールアウト



Amazon Aurora with PostgreSQL Compatibility

- 高性能な完全マネージド型のRDBサービス
  - AWS独自の分散共有ストレージ
    - 更新性能が高い
    - データは、自動的に3つのAZにわたる6つのレプリカに格納
  - 最大15台のリードレプリカ
    - 障害発生時には、リードレプリカが自動的にマスタにフェイルオーバ



RDS for PostgreSQL の制限

- クライアント認証
  - pg_hba.conf による認証ができない
    - ユーザとデータベースに基づいた接続認証ができない
  - AWS のセキュリティグループを設定して接続元を制限
- スーパユーザになれない
  - 拡張モジュールは予め用意されたものしか使えない

- 自由度の高いことをしたければ EC2 を使う
  - サポートされていない拡張モジュールや独自の拡張
    - 日本語全文検索（pg_bigm, PGroonga)、PL/Python、C言語関数、・・・
  - PostgreSQL 10 を使いたい

  ・・・など

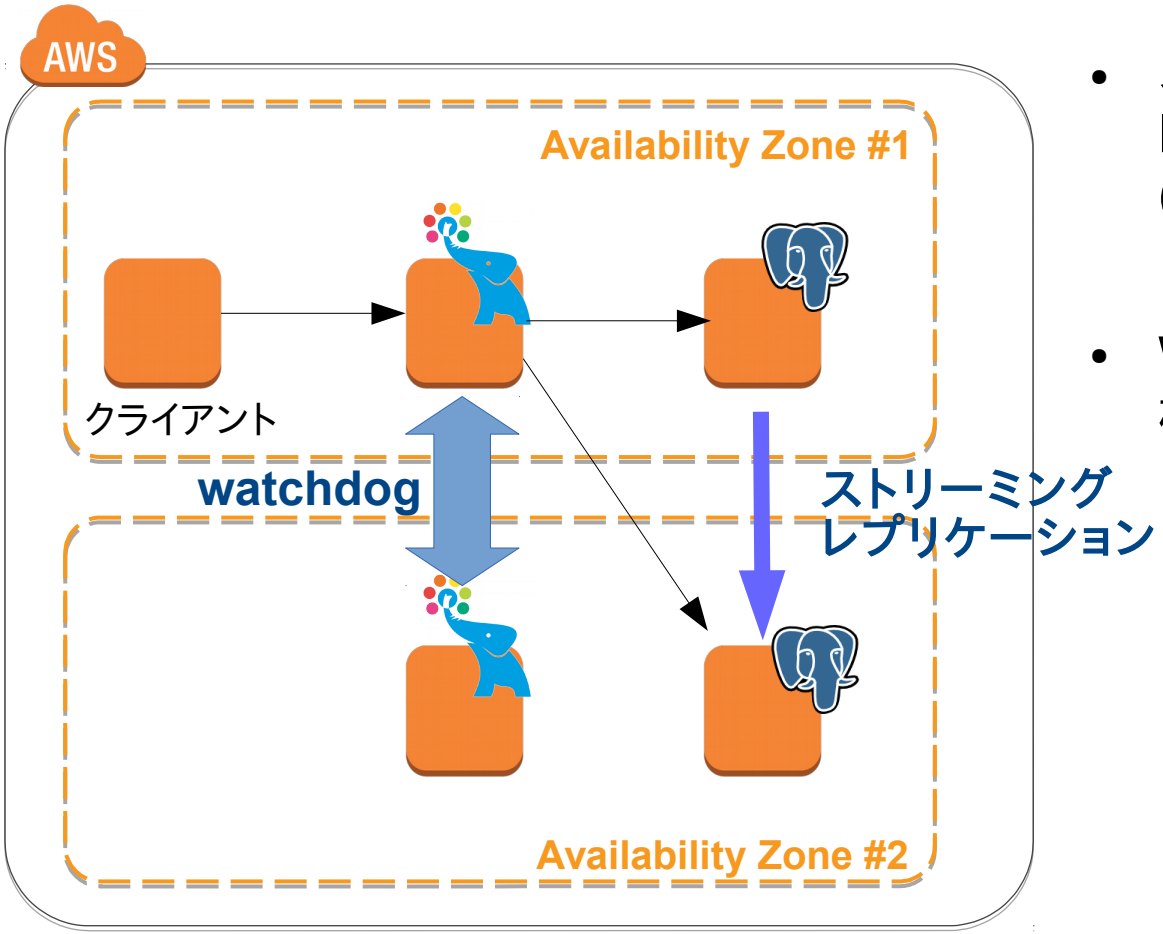
PostgreSQL と Pgpool-II を一緒に AWS で使う

AWS で Pgpool-II を使うメリット

- クエリの自動振り分け & 負荷分散
  - 更新クエリはプライマリ（マスタ）に
  - 参照クエリはスタンバイ（リードレプリカ）に
- 高可用性
  - EC2 で PostgreSQL を使う場合は自動フェイルオーバ機能を
    - EC2インスタンスの障害やメンテナンスに備える
  - RDSやAuroraの場合は、フェイルオーバはAWSにお任せ
- その他
  - コネクションプーリング
  - インメモリクエリキャッシュ
  - ユーザ・データベースに基づくクライアント認証

PostgreSQL on EC2 + Pgpool-II

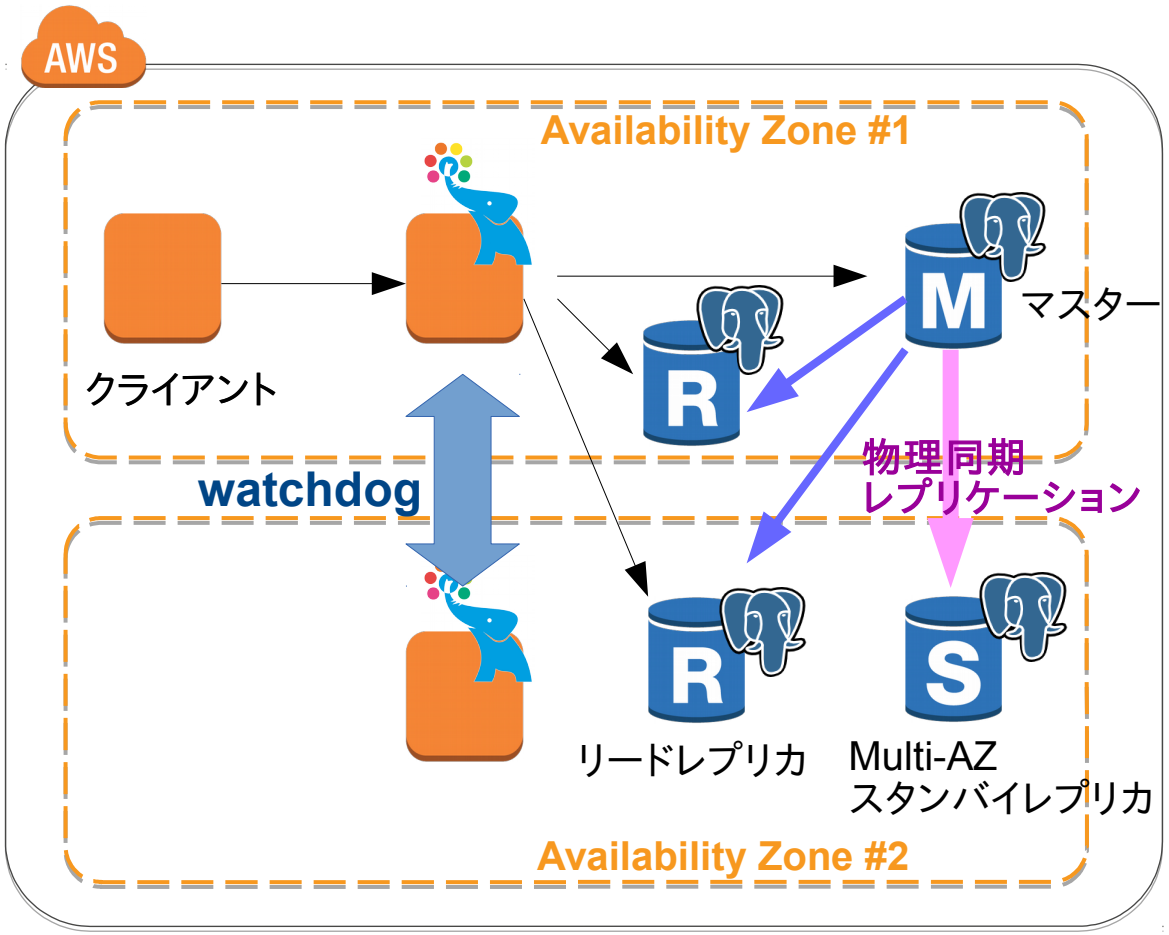
- EC2 に PostgreSQL と Pgpool-IIをインストール＆設定
  - 物理サーバの場合とほぼ同じ



- ネットワークが不安定な場合には Pgpool-II の接続タイムアウト設定 (connect_timeout) を大きめにしておくとよい
- Watchdog クラスタは、Pgpool-II 3台構成を推奨
  - Quorum を確保してスプリットブレインを防ぐため
    ※ ただし、本資料では Pgpool-II 2台の最小構成で解説

RDS for PostgreSQL + Pgpool-II

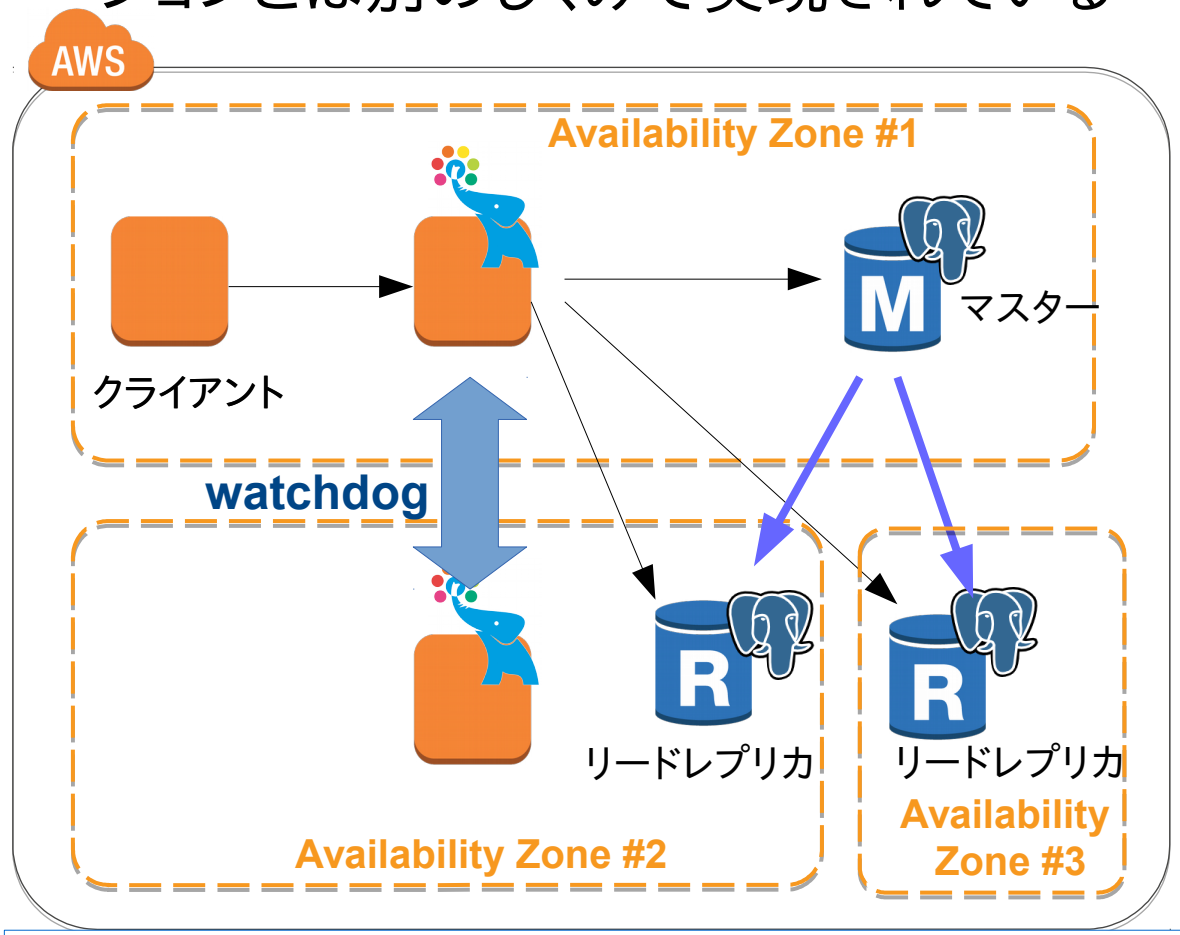
- 基本的にストリーミングレプリケーション構成と同様



- Pgpool-II はマスタをプライマリとして認識
  - 更新クエリはマスタに
  - 参照クエリは負荷分散
- フェイルオーバは AWS におまかせ
  - Multi-AZ スタンバイレプリカがマスタに昇格
- フェイルオーバには時間がかかる
  - ヘルスチェックのリトライ回数を十分にとっておく
  - 接続失敗時にもバックエンドが切り離されないようにしておく (fail_over_on_backend_error=off)
- Pgpool-II のクライアント認証で md5 を有効に設定

Aurora with PostgreSQL + Pgpool-II

- リードレプリカはストリーミングレプリケーションとは別のしくみで実現されている



設定例がPgpool-II 3.7 のドキュメントに同梱

- 明示的にマスターノードを指定
  - ALWAYS_MASTER フラグ (= Pgpool-II 3.7 の新機能)
- ストリーミングレプリケーションの遅延チェックは無効にする
- フェイルオーバは AWS にお任せ
  - リードレプリカがマスタに昇格
- フェイルオーバには時間がかかる & 全DBノードが一時的に停止
  - ヘルスチェックのリトライ回数を十分にとっておく
  - 接続失敗時にもバックエンドが切り離されないようにしておく (fail_over_on_backend_error=off)
- Pgpool-II のクライアント認証で md5 を有効に設定

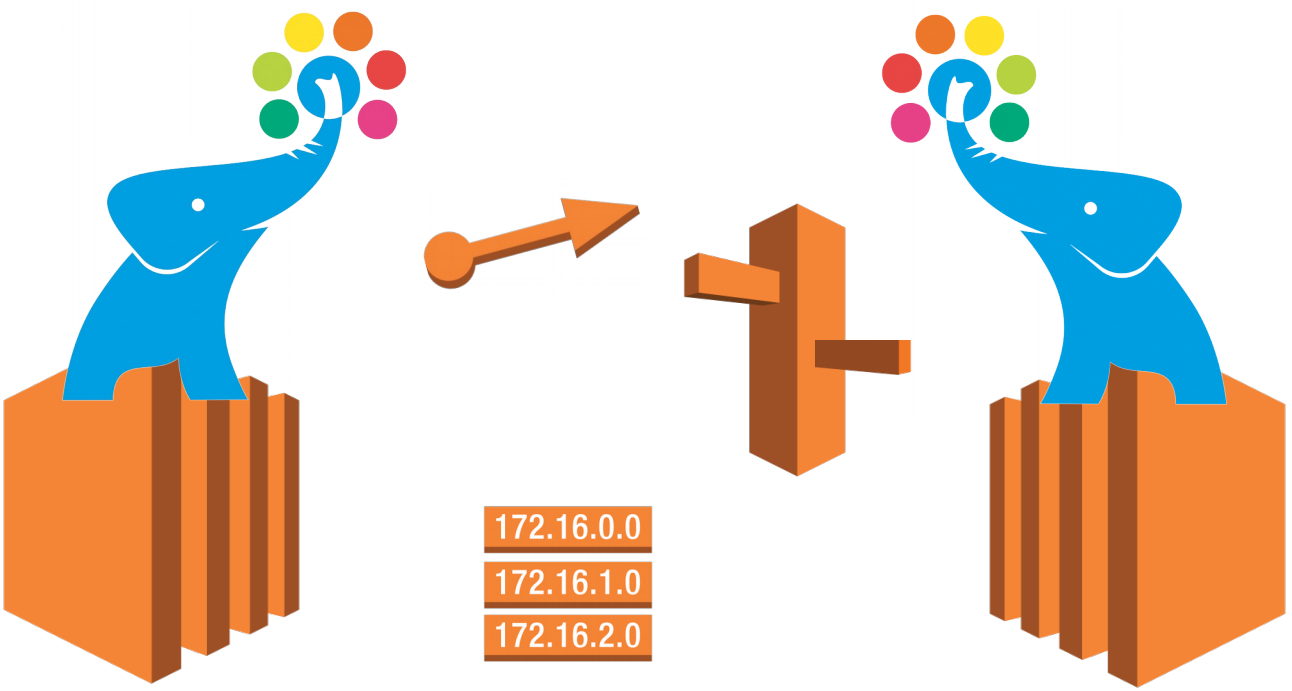
DBフェイルオーバ時間の比較

- 方法:
  - 1秒毎に Pgpool-II に更新クエリを送信している最中にフェイルオーバを発生させ、クエリが実行できなかった時間を調べる
- 結果:

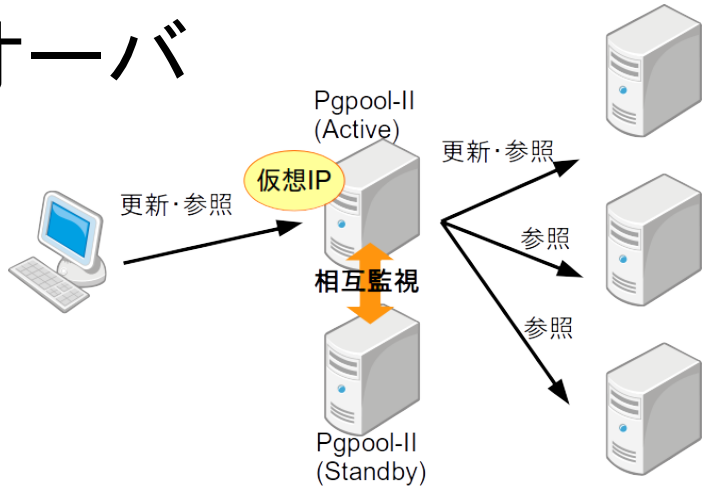
EC2 （PostgreSQL 2台 ストーリミング レプリケーション）	RDS (Multi-AZ構成 + レプリカ1台)	Aurora （マスタ+ レプリカ2台）
約1秒	約80秒	約20秒

- PostgreSQL on EC2
  - Pgpool-II がフェイルオーバを実施
  - クエリ実行が失敗した直後にフェイルオーバ開始（Pgpool-II の設定次第）
- RDS、Aurora
  - AWS がフェイルオーバを管理
  - Aurora の方がフェイルオーバが速い結果

AWS で Pgpool-II の Watchdog を使う



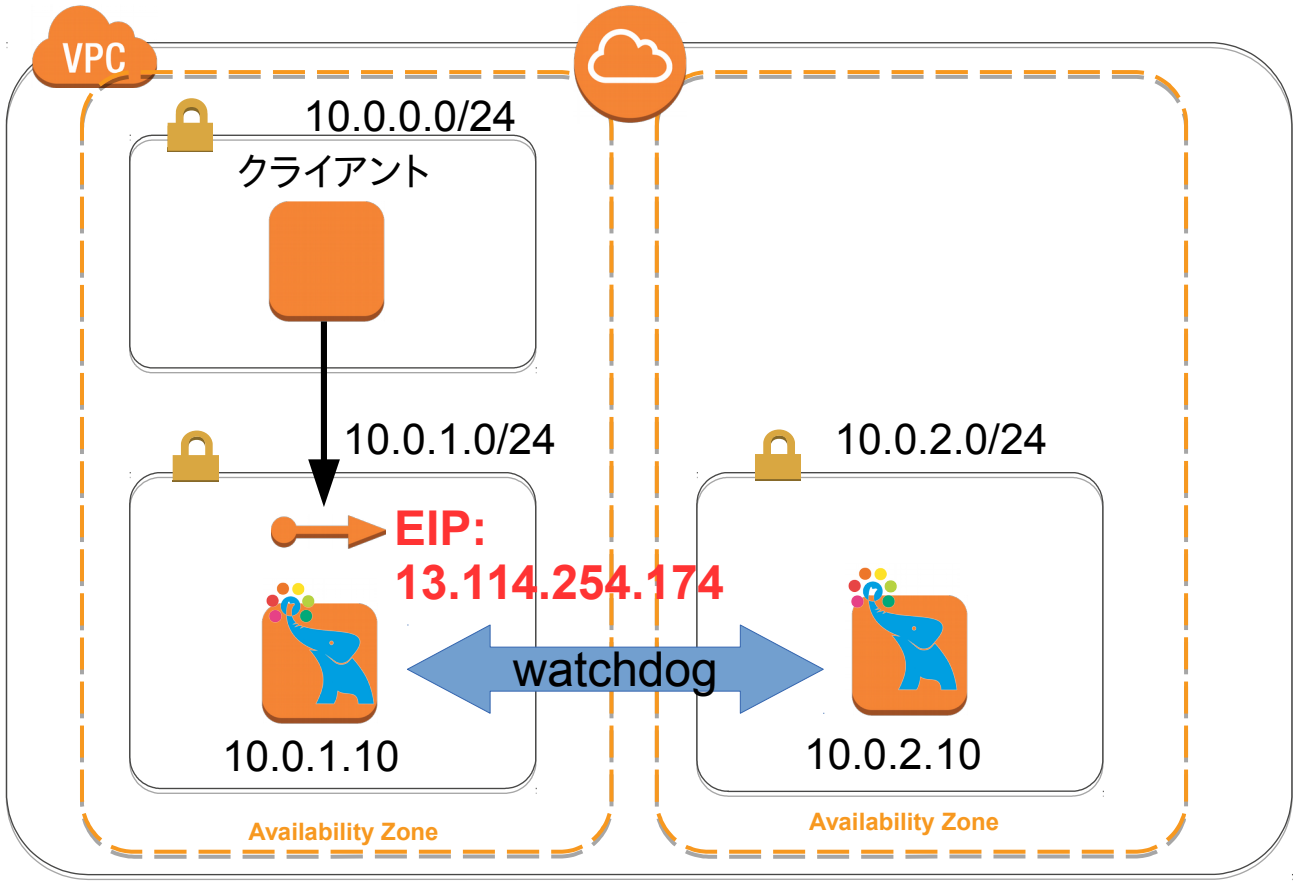
Pgpool-II のフェイルオーバ



- Watchdog
  - Pgpool-II 自身の可用性を高める機能
  - 障害発生時には仮想IP の付け替えでフェイルオーバ
- しかし・・・AWS の EC2 では仮想IPが使えないので代りの方法を使う
  - Elastic IP
  - Route53
  - Route tabales
- Pgpool-II はアクティブに昇格および降格する時に任意のコマンドを実行可能
  (wd_escalation_command, wd_de_escalation_command)
  - ここで AWS CLI コマンドを呼び出すことで、AWS のネットワークを制御する
    - AWS CLI の実行には、Pgpool-II がインターネットにアクセスできる必要がある
    - パブリックサブネットに配置するか、Nat インスタンス/ゲートウェイを利用

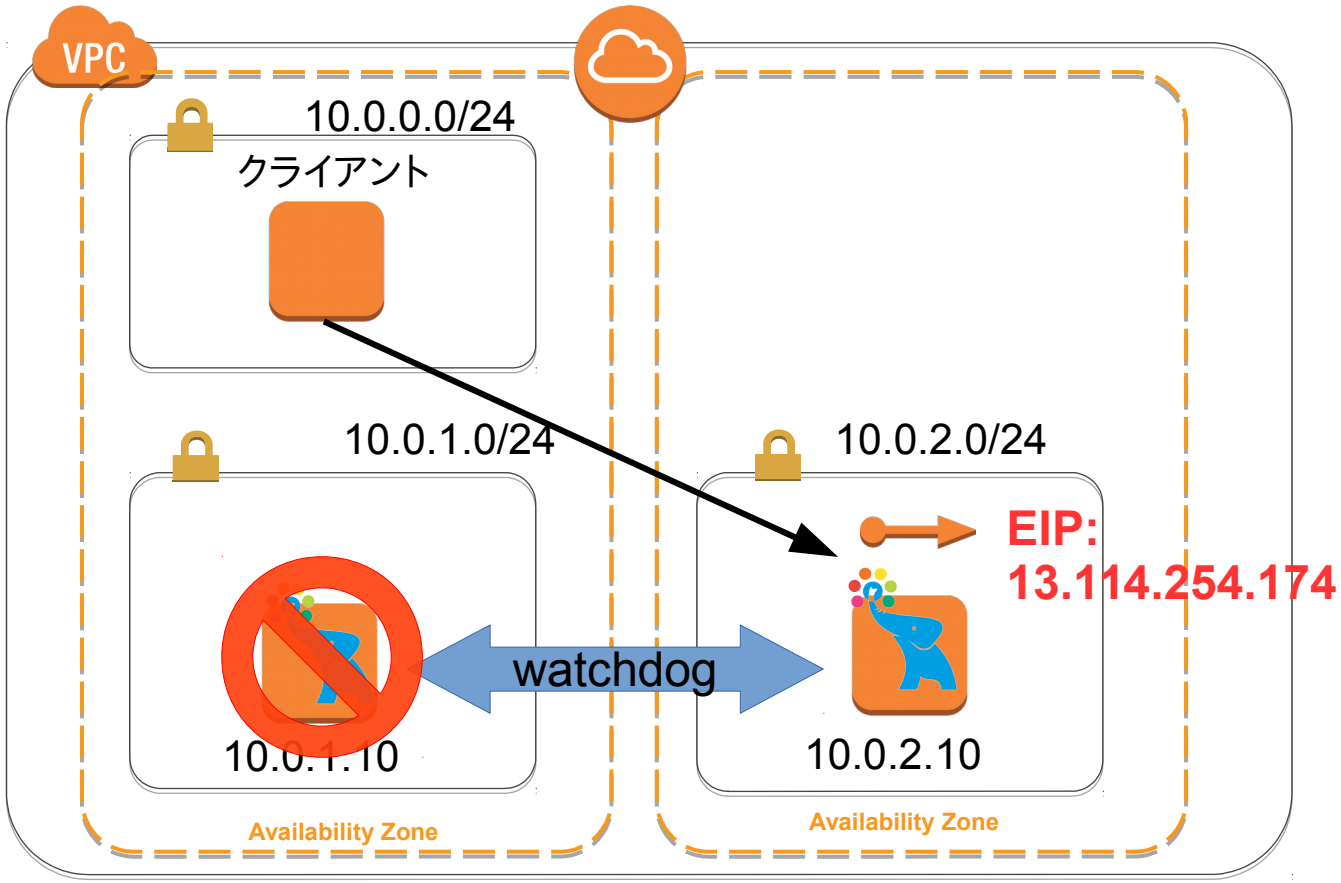
方法1: Elastic IP の付け替え

- 稼動系の Pgpool-II に Elastic IP (EIP) を割り当て、フェイルオーバ時に付け替える
  - Elastic IP = AWS から提供される固定のパブリック IP
  - 仮想IPの付け替えと同じ考え方で直感的にわかりやすい



方法1: Elastic IP の付け替え

- 稼動系の Pgpool-II に Elastic IP (EIP) を割り当て、フェイルオーバ時に付け替える
  - Elastic IP = AWS から提供される固定のパブリック IP
  - 仮想IPの付け替えと同じ考え方で直感的にわかりやすい



方法1: Elastic IP の付け替え(続き)

- 昇格時の処理

```
ELASTIC_IP=13.114.254.174
INSTANCE_ID=`curl http://169.254.169.254/latest/meta-data/instance-id`

# 自サーバのインスタンスIDに Elastic IP を関連付ける
aws ec2 associate-address --instance-id $INSTANCE_ID --public-ip $ELASTIC_IP
```

- 降格時の処理

```
ELASTIC_IP=13.114.254.174

# Elastic IP の関連付けを解除
aws ec2 disassociate-address --public-ip $ELASTIC_IP
```

- Elasitic IP はパブリックIP

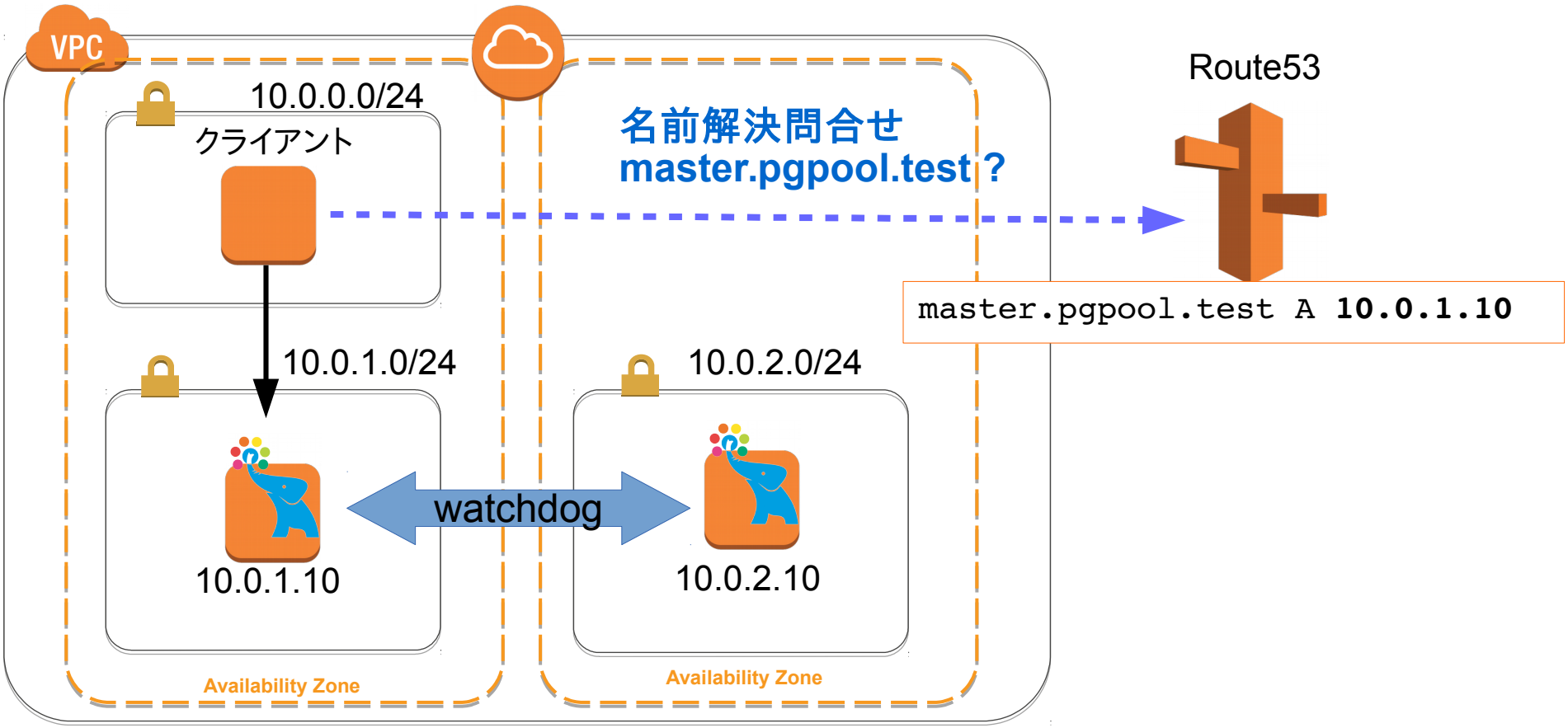
設定例がPgpool-II 3.6～ のドキュメントに同梱

  - クライアントはインターネットにアクセス可能である必要がある
  - Pgpool-II はインターネットからアクセス可能なネットワークに配置

    → セキュリティグループを慎重に設定しなければならない

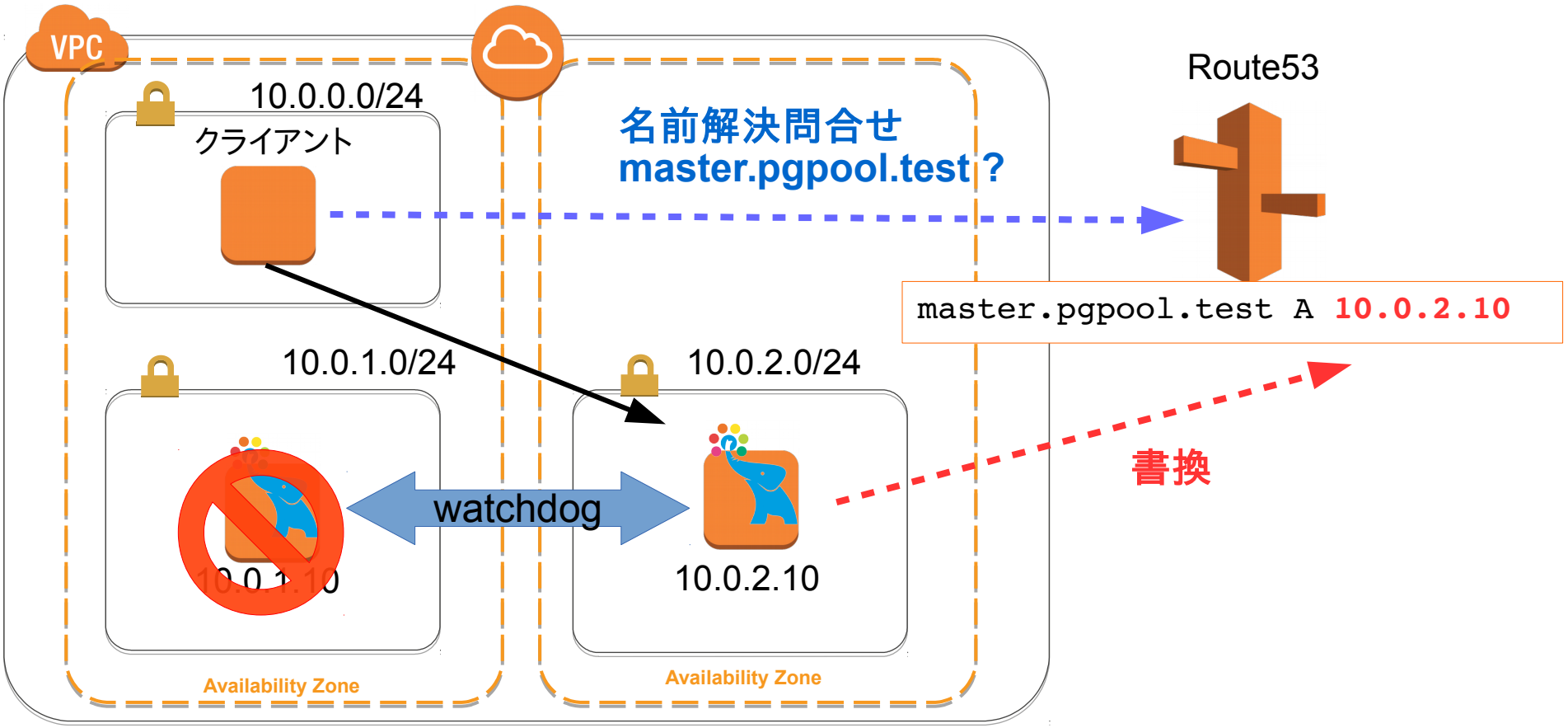
方法2: DNS レコード書換

- ローカルなDNS名を用意し、稼動系の Pgpool-II のプライベートIPに解決させる
  - フェイルオーバ時には別の Pgpool-II の IP に解決するようにレコードを書き換える
  - Route53 をつかってプライベートな DNS サーバを立てる（利用には費用が必要）



方法2：DNS レコード書換

- ローカルなDNS名を用意し、稼動系の Pgpool-II のプライベートIPに解決させる
  - フェイルオーバ時には別の Pgpool-II の IP に解決するようにレコードを書き換える
  - Route53 をつかってプライベートな DNS サーバを立てる（利用には費用が必要）



方法2: DNS レコード書換(続き)

- 昇格時の処理

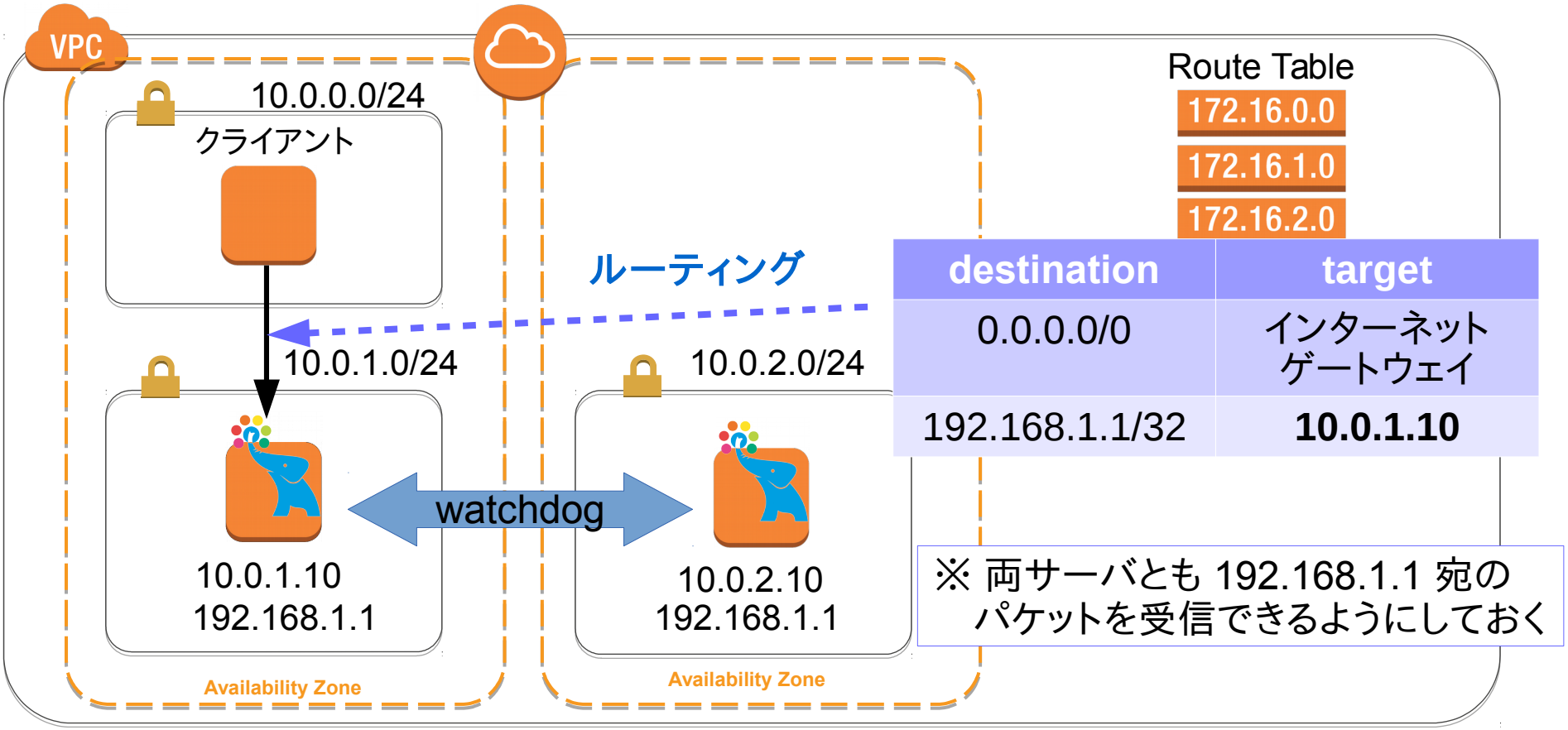
```
HOST=master.pgpool.test.
PRIVATE_IP=`curl http://169.254.169.254/latest/meta-data/local-ipv4`
HOSTZONE_ID=XXXXXXXXXXX

# DNSキャッシュの生存期間(TTL)は小さめに
TTL=1

# 自サーバのプライベート IP に DNS 名を解決させるように A レコードを更新
aws route53 change-resource-record-sets --hosted-zone-id $HOSTZONE_ID \
 --change-batch \
'{
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "'$HOST'",
        "Type": "A",
        "TTL": '$TTL',
        "ResourceRecords": [{ "Value": "'$PRIVATE_IP'" }]
      }
    }
  ]
}'
```

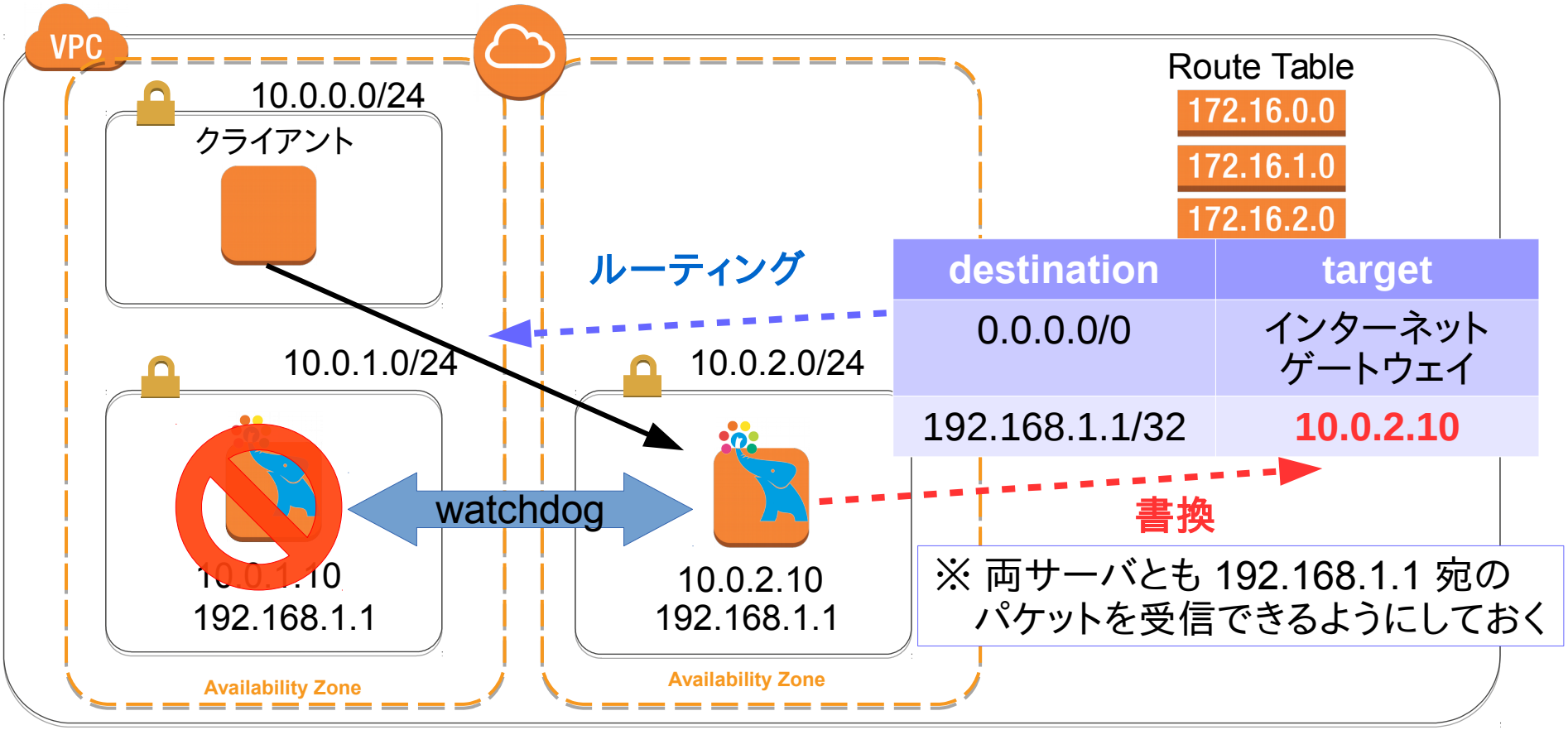
方法3: ルートテーブルの書換

- 特定のIPへのルーティングを稼動系の Pgpool-II のインスタンスに向けさせる
  - フェイルオーバ時には別の Pgpool-II インスタンスに向くようルートテーブルを更新する



方法3: ルートテーブルの書換

- 特定のIPへのルーティングを稼動系の Pgpool-II のインスタンスに向けさせる
  - フェイルオーバ時には別の Pgpool-II インスタンスに向くようルートテーブルを更新する



方法3: ルートテーブルの書換（続き）

- 昇格時の処理

```
VIP_CIDR=192.168.1.1/32
ROUTE_TABLE_ID=rtb-XXXXXX
INSTANCE_ID=`curl http://169.254.169.254/latest/meta-data/instance-id`

# 特定 IP アドレスが自サーバのインスタンスIDに向くようにルートテーブルにルールを追加（または、書換）
aws ec2 create-route --route-table-id $ROUTE_TABLE_ID \
       --destination-cidr-block $VIP_CIDR --instance-id $INSTANCE_ID
if [ $? != 0 ];then
  aws ec2 replace-route --route-table-id $ROUTE_TABLE_ID \
        --destination-cidr-block $VIP_CIDR --instance-id $INSTANCE_ID
fi
```

- 両 Pgpool-II サーバは特定IPへのパケットを受信できるようにしておく必要がある
  - ループバックアップインタフェースにセカンダリ IP として追加

```
# ifconfig lo:0
lo:0      Link encap:Local Loopback
          inet addr:192.168.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

  - EC2インスタンスの設定「ソース/宛先チェック」を無効にしておく

Pgpool-II フェイルオーバ時間

- 方法：
  - 1秒毎に Pgpool-II に更新クエリを送信している最中にフェイルオーバを発生させ、クエリが実行できなかった時間を調べる
- 結果：

Elastic IP	Route53	Route Table
約17秒	約30～60秒	約12秒

  - Elastic IP の付け替え、Route Table の書換が速いという結果
  - Route53 の DNS レコード書換は、キャッシュの生存期間（TTL)を 1 秒としても切り替えに時間がかかった

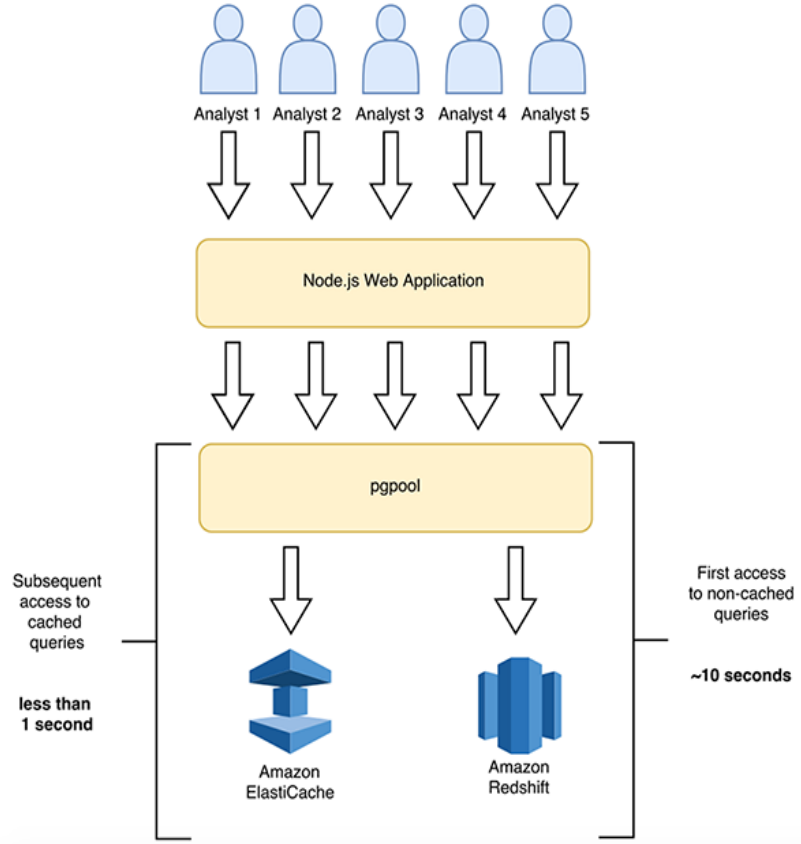
3つの方法の比較

- Elastic IP
  - Pgpool-II をパブリックサブネットにおく必要がある
  - 簡単
  - 切り替えが速い
- Route53
  - プライベートサブネットで利用可能
  - 切り替え時間はDNSキャッシュの TTL に依存？
  - 費用が必要
- Route Table
  - プライベートサブネットで利用可能
  - 設定がやや煩雑
  - 切り替えが速い

おまけ: Redshift + ElastiCache + Pgpoo-II

(AWS ソリューションアーキテクトのブログより)

- Redshift のクエリの結果を Pgpool-II のインメモリキャッシュ機能を使用して ElastiCache にキャッシュ



- Amazon Redshift:
  - PostgreSQL ベースのデータウェアハウス
- Amazon ElastiCache:
  - Memcached 互換のメモリキャッシュ

https://aws.amazon.com/jp/blogs/big-data/using-pgpool-and-amazon-elasticache-for-query-caching-with-amazon-redshift/

まとめ

- PostgreSQL on AWS
  - Amazon EC2
  - Amazon RDS for PostgreSQL
  - Amazon Aurora with PostgreSQL Compatibility

- どのケースでも Pgpool-II を利用可能
  - クエリ振り分け、参照負荷分散、自動フェイルバック、
    クエリキャッシュ、コネクションプーリング、クライアント認証・・・
  - Watchdog も利用可能
    - Elastic IP, Route53, Route Tables

AWS でも Pgpool-II を使ってみよう!

参考URL

- Pgpool-II オフィシャルサイト

  http://www.pgpool.net/ （日本語）

  - http://www.pgpool.net/jp/ （英語）
    - Pgpool-II 3.7 beta が公開中!

- AWS ドキュメント

  https://aws.amazon.com/jp/documentation/

- SRA OSS, Inc. 日本支社
  - セミナー資料、事例情報、技術情報

    http://www.sraoss.co.jp/

    - Pgpool-II 最新情報
    - Amazon RDS for PostgreSQL の概要
    - Amazon Aurora for PostgreSQL の概要と検証結果

      ・・・など

オープンソースとともに

SRA OSS, INC.

URL：http://www.sraoss.co.jp/
E-mail：sales@sraoss.co.jp
Tel：03-5979-2701