PostgreSQL 10で作るクラスタ構成

SRA OSS, Inc. 日本支社
取締役支社長
石井　達夫

SRA OSS,INC.

自己紹介

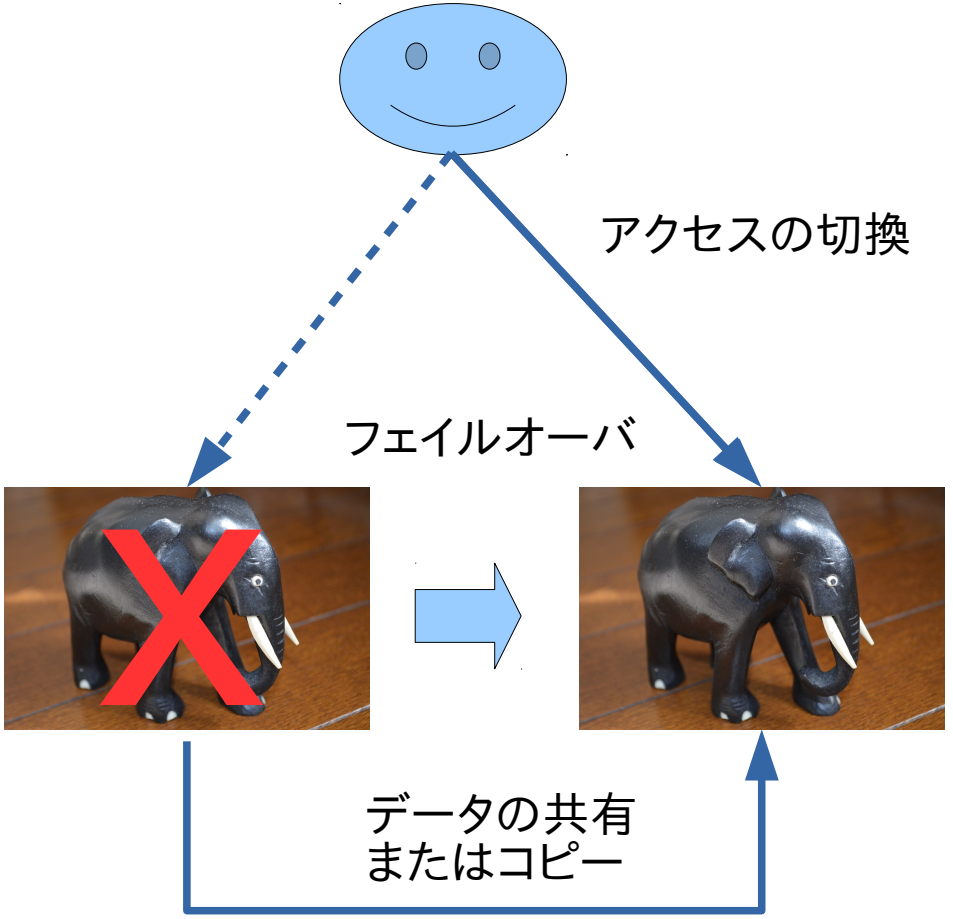
- OSSの開発とビジネスに携わっています
  - PostgreSQLのコミッタ
  - PostgreSQL用のクラスタソフト Pgpool-II の開発

クラスタリングの目的

- 複数のDBサーバを使って、1台のDBサーバでは実現できないメリットを得る
- 可用性の向上を狙うのか、性能向上を狙うのか分けて考えると整理しやすい

PostgreSQLにおける クラスタリングの目的(1)

- 可用性の向上(High Availability)
  - もっとも古典的かつ基本的なリクエスト
    - ダウンタイムの短縮
    - データ損失可能性の低減
  - なんらかの方法で同じデータベース内容を持つ待機系のシステムを用意し、稼動系が故障した時には待機系へアクセスを切り替える
  - 待機系は休止状態なので、負荷分散による性能向上は見込めない



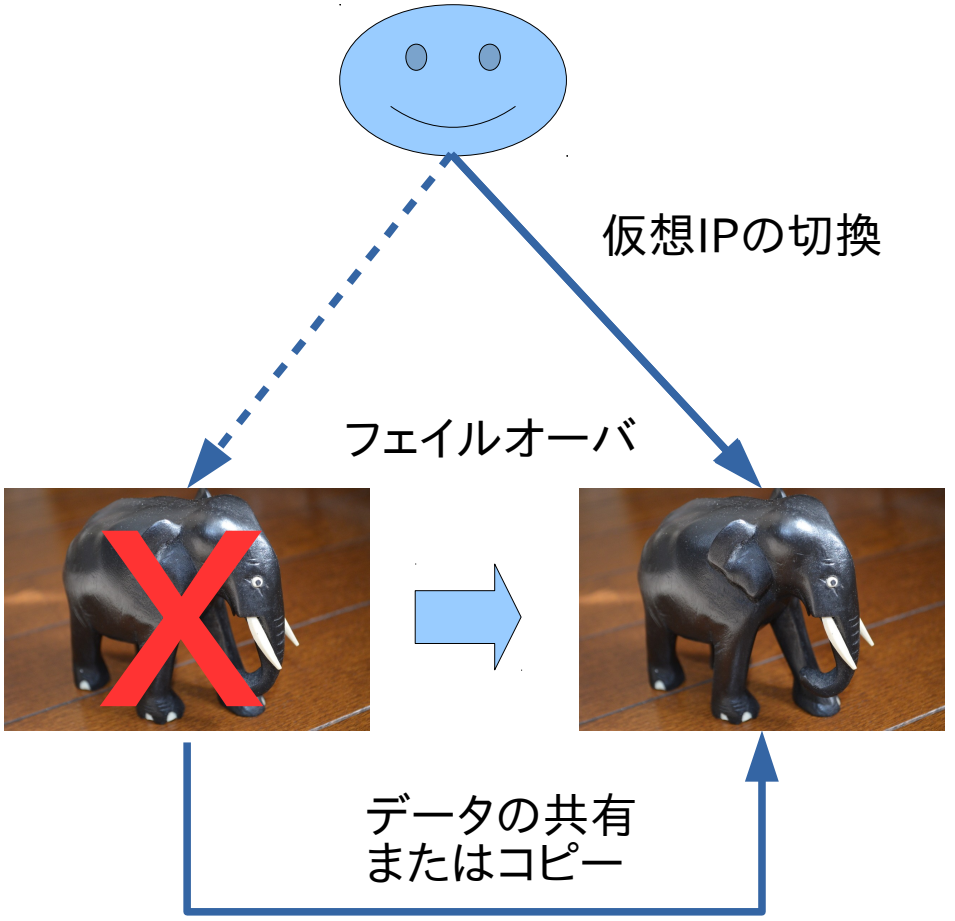
PostgreSQLにおける クラスタリングの目的(2)

- 性能の向上(Scale out)
  - 検索性能(read)の向上
    - データのコピー(レプリカ)を用意し、検索性能の向上を狙う
      - PostgreSQLの組み込みレプリケーション
    - シャーディングにより検索性能の向上を狙う
      - 現在のところ、サードパーティによる実装のみ
      - 例: Citus、Postgres-XL
  - 更新性能(write)の向上
    - シャーディングにより更新性能の向上を狙う
      - 現在のところ、サードパーティによる実装のみ
      - 例: Citus、Postgres-XL

可用性向上のための クラスタ技術

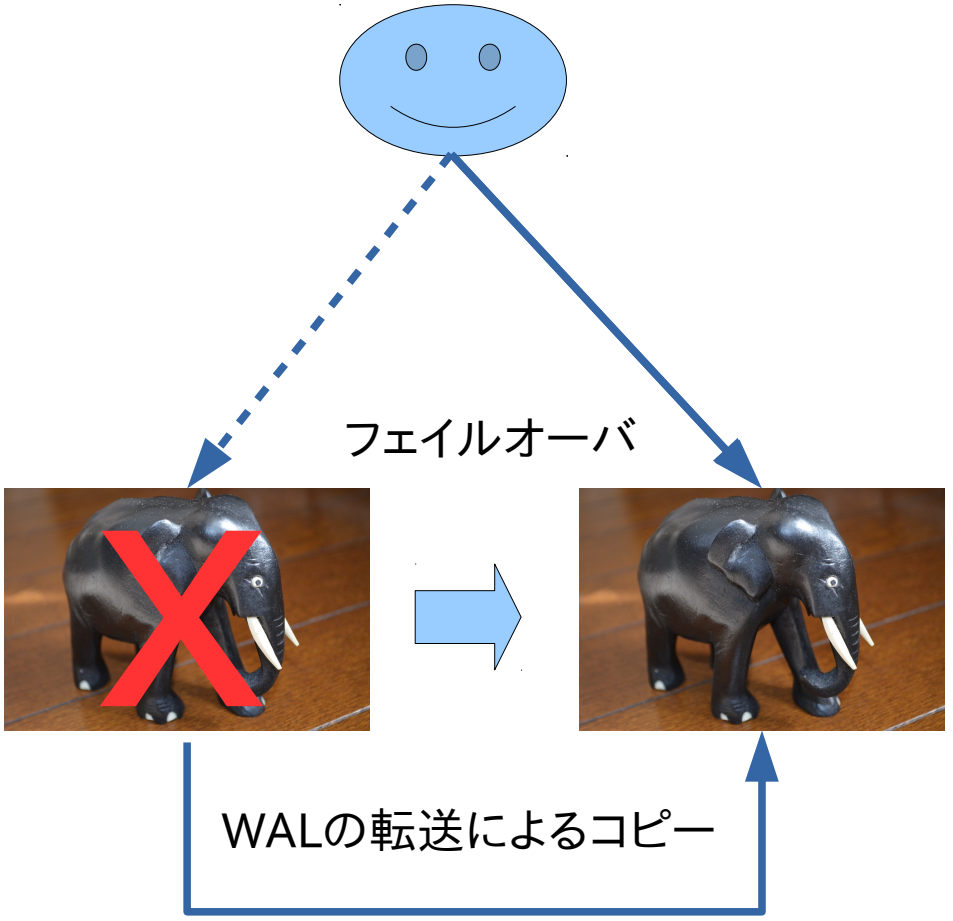
可用性向上のためのクラスタ技術：アクティブ・スタンバイ方式

- Pacemakerなどの汎用HAソフトを使って複数のPostgreSQLを管理
- DBの入ったディスク装置を共有する共有ディスク方式と、共有しない方式がある
- 性能は向上しない
- アプリケーションの修正は必要ない
  - フェイルオーバ中はセッションが切断されるのでその対応は必要
- 待機系のPostgreSQLは利用できない



可用性向上のためのクラスタ技術：ストリーミングレプリケーション(1)

- PostgreSQL組み込みのレプリケーション技術である「ストリーミングレプリケーション」を利用
- プライマリからスタンバイへトランザクションログを転送してデータをコピーする
- 書き込み性能は向上しないが、複数のPostgreSQLで読み出し負荷を共有して性能を向上させることも可能
- アプリケーションの修正は必要
  - フェイルオーバ中はセッションが切断されるのでその対応は必要
  - 書き込み処理はプライマリにしか投げてはいけない（そのほか、ある種のロックはプライマリだけに投げるなどの考慮が必要）
- これらの対応が困難な場合には、すべてのDB処理をプライマリにのみ投げる（その場合でもフェイルオーバへの対応は必要）



可用性向上のためのクラスタ技術：ストリーミングレプリケーション(2)

- プライマリが故障した時に、どのスタンバイを昇格させるか決めておく必要がある(スタンバイの数が2以上の場合)
  - 同期レプリケーションを使っている場合は、同期スタンバイを昇格させる
    - プライマリにコミットされたデータが失われないことが保証できる
  - 非同期レプリケーションを使っている場合は、固定ルールで昇格するスタンバイを決めておく方式と、動的に決める(最も遅延が少ないものを選ぶ、など)方式がある

プライマリ

スタンバイ1

スタンバイ2

スタンバイ3

性能向上のための
クラスタ技術

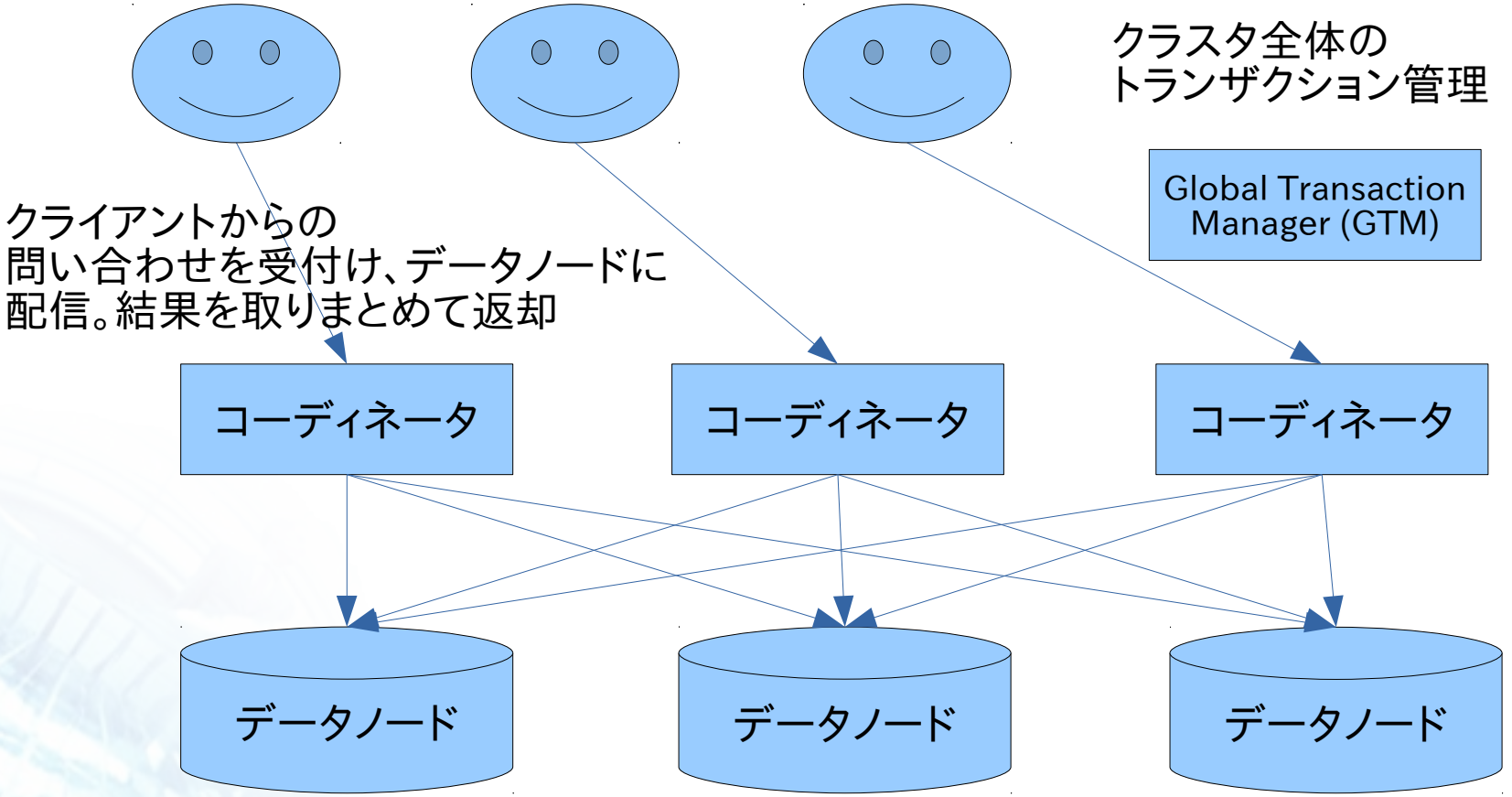
性能向上のためのクラスタ技術：ストリーミングレプリケーション

- スタンバイサーバを複数設けて検索性能を向上させる（負荷分散）
  - 一つのSQLが分散処理されるわけではないので、多数のセッションが同時に実行されるような環境で効果が上がる
  - プライマリサーバが過負荷のときに、プライマリに検索処理をさせないようにするのも効果がある
  - どのセッションがどのスタンバイサーバに接続するかを決める必要がある
    - アプリケーションで行う（アプリケーションの改造が必要）
    - ミドルウェアで透過的に実施(Pgpool-II)
  - 更新性能は向上しない

性能向上のためのクラスタ技術：Postgres-XL

- PostgreSQLで並列分散クラスタをサポートするために始まったプOSSロジェクトで、PostgreSQLに大量のパッチをあてるフォークとして実装されている
- 当初「Postgres-XC」という名前でスタートしたが、現在は「Postgres-XL」という後継プロジェクトに引き継がれている
- Postgres-XLの機能
  - シャーディング
  - レプリケーション
  - 複数シャードをまたぐ検索(および問い合わせの最適化)
  - 分散トランザクション管理
  - 更新性能の向上が狙える
  - マルチマスター

性能向上のためのクラスタ技術：Postgres-XLのアーキテクチャ



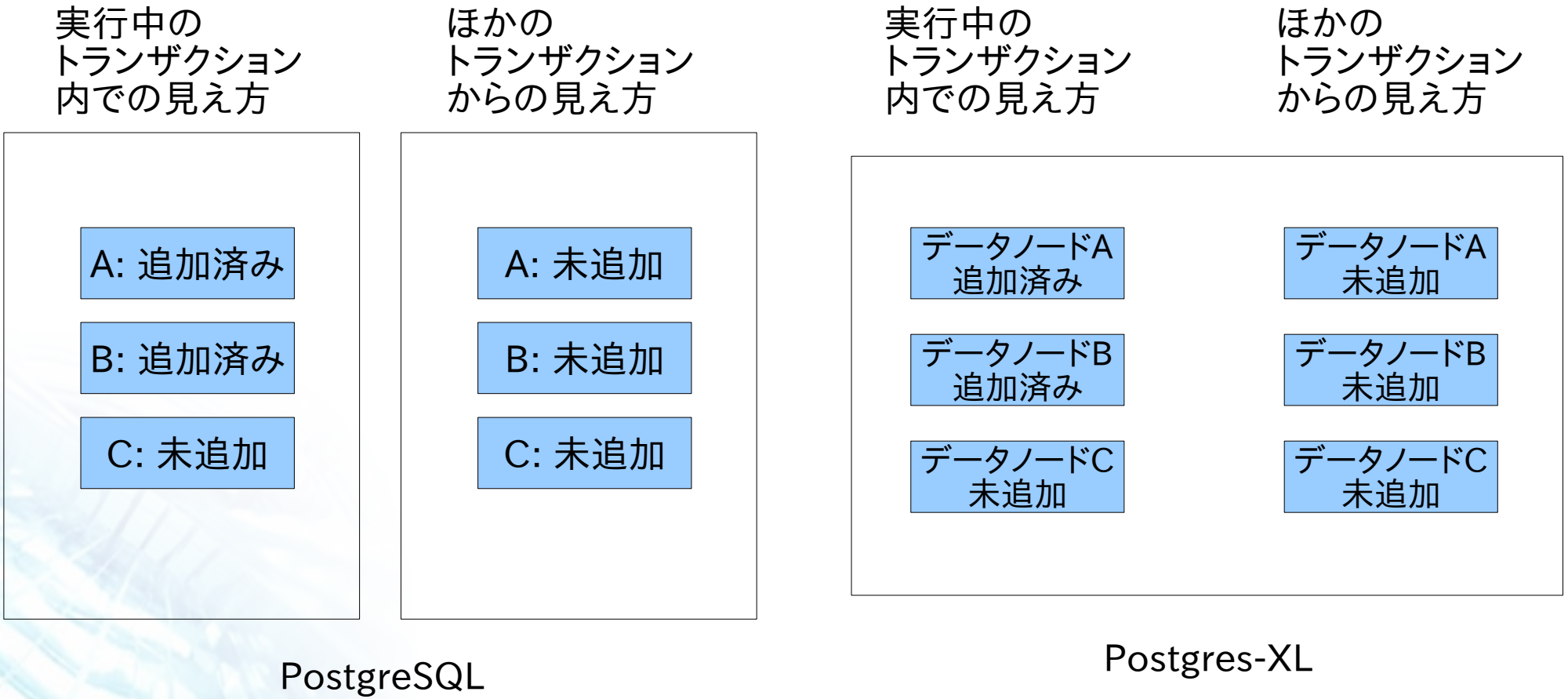
実際の問い合わせ処理を実行

SRA OSS, INC.

性能向上のためのクラスタ技術：Postgres-XLの特徴

- データノード間で並列に問い合わせが実行可能
- データをデータノードに分散（シャーディング）可能なので、検索性能のみならず、更新性能も向上
- 更新遅延がなく、データノードをまたがった読み取り一貫性も保証されている
- マルチマスター
- 可用性は向上しない(むしろ低下する)
  - データノードのレプリケーションで対応

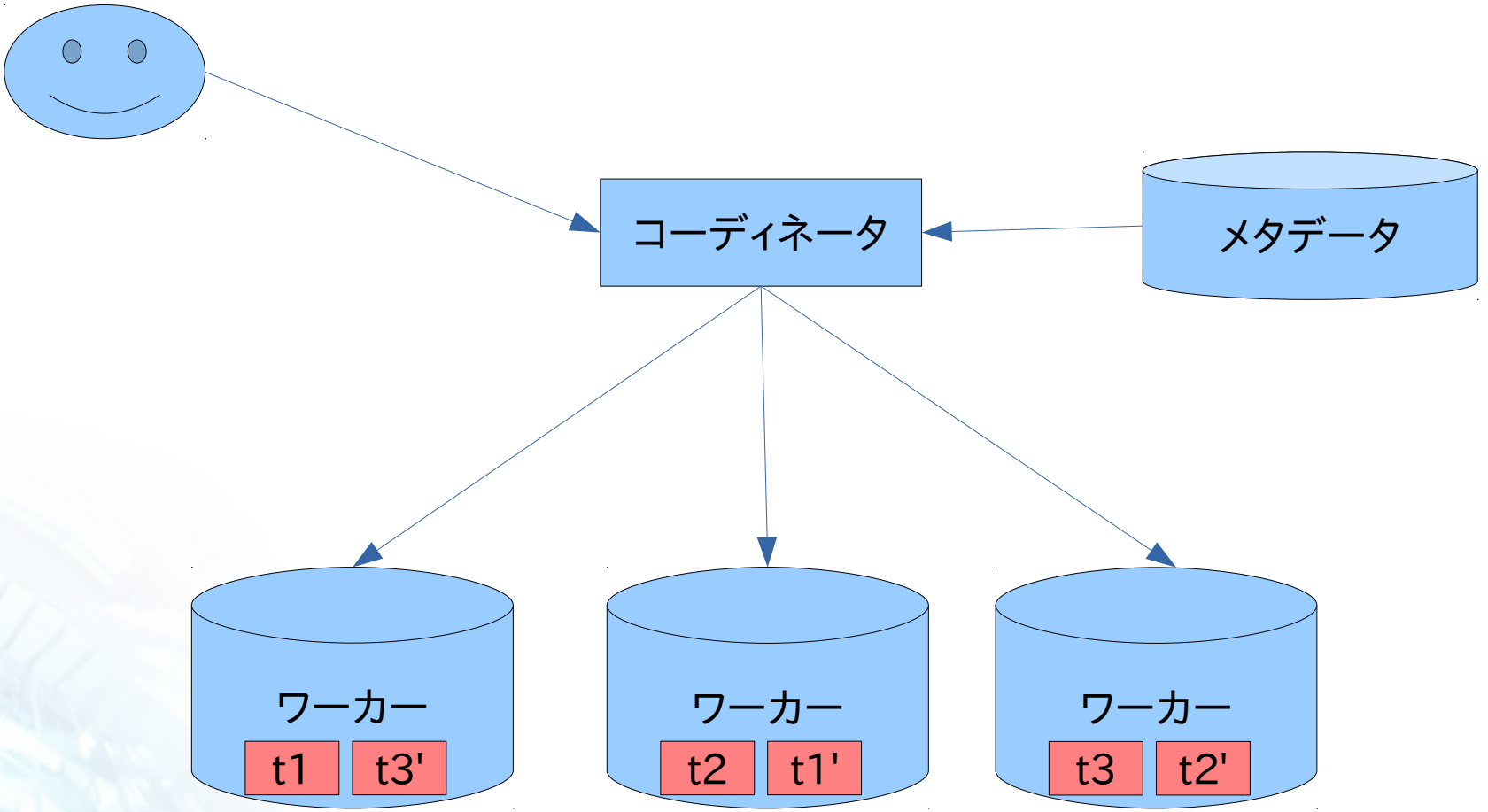
性能向上のためのクラスタ技術：読み取り一貫性



性能向上のためのクラスタ技術：Citus

- PostgreSQLでシャーディングを行うための拡張モジュール(extension)
- Citus data社が開発し、OSSとしてGithubで公開
- 「pg_shard」と呼ばれていたプロジェクトの後継
- 向き、不向き(Citusのマニュアルより抜粋)
  - 向いている処理
    - 大量のログ追加、サマリの検索
  - 向いていない処理
    - 多数のシャードにまたがるOLTP処理
    - 複雑なクエリを扱うデータウェアハウス的な問い合わせ

性能向上のためのクラスタ技術：Citusのアーキテクチャ



性能向上のためのクラスタ技術：CitusとPostgres-XLの違い

- CitusはPostgreSQLの拡張モジュール(extension)として実装されており、軽量、シンプル
- Citusはグローバルトランザクションマネージャを持たないので、複数シャードにまたがった読み取り一貫性は保証されない
  - worker1でコミットされたデータは、worker2, worker3のデータがコミットされる前に、他のトランザクションから見えるようになる
- デフォルトではデータ更新は、2 phase commitではなく、1 phase commitで更新される(設定で2 phase commitを使用することも可能)

各種クラスタ技術のまとめ

	可用性向上	検索性能向上	更新性能向上	アプリケーション変更度合い	PostgreSQL変更必要	実用性・実績
Pacemaker	○	X	X	○	○	○
Streaming replication	○	○	X	X	○	○
Postgres-XL	X	▲	○	X	X	▲
Citus	○	▲	○	X	○	▲

○:寄与する
X:寄与しない
▲:制限事項あり／工夫すれば寄与できる

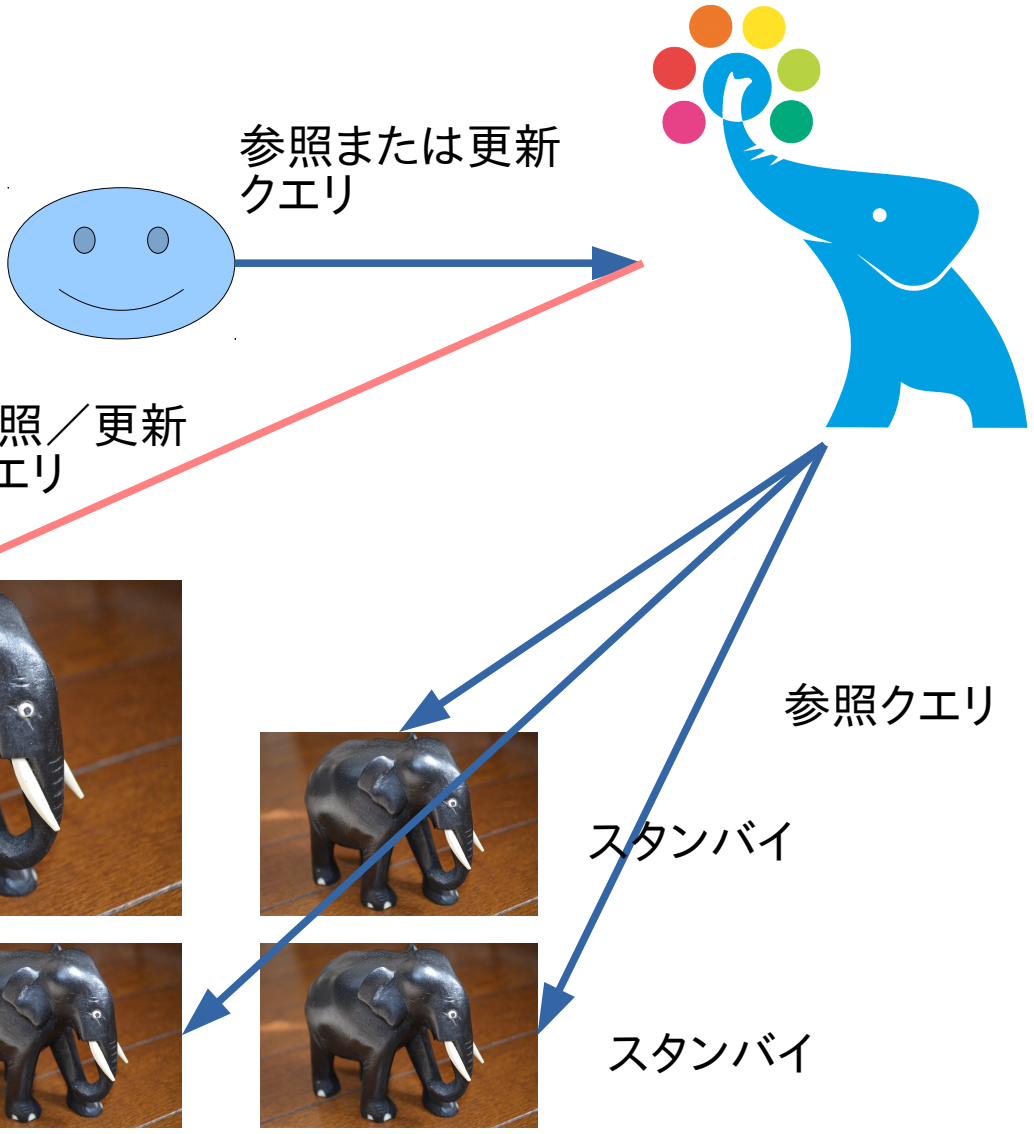
Pgpool-IIのご紹介と使いどころ

Pgpool-IIとは

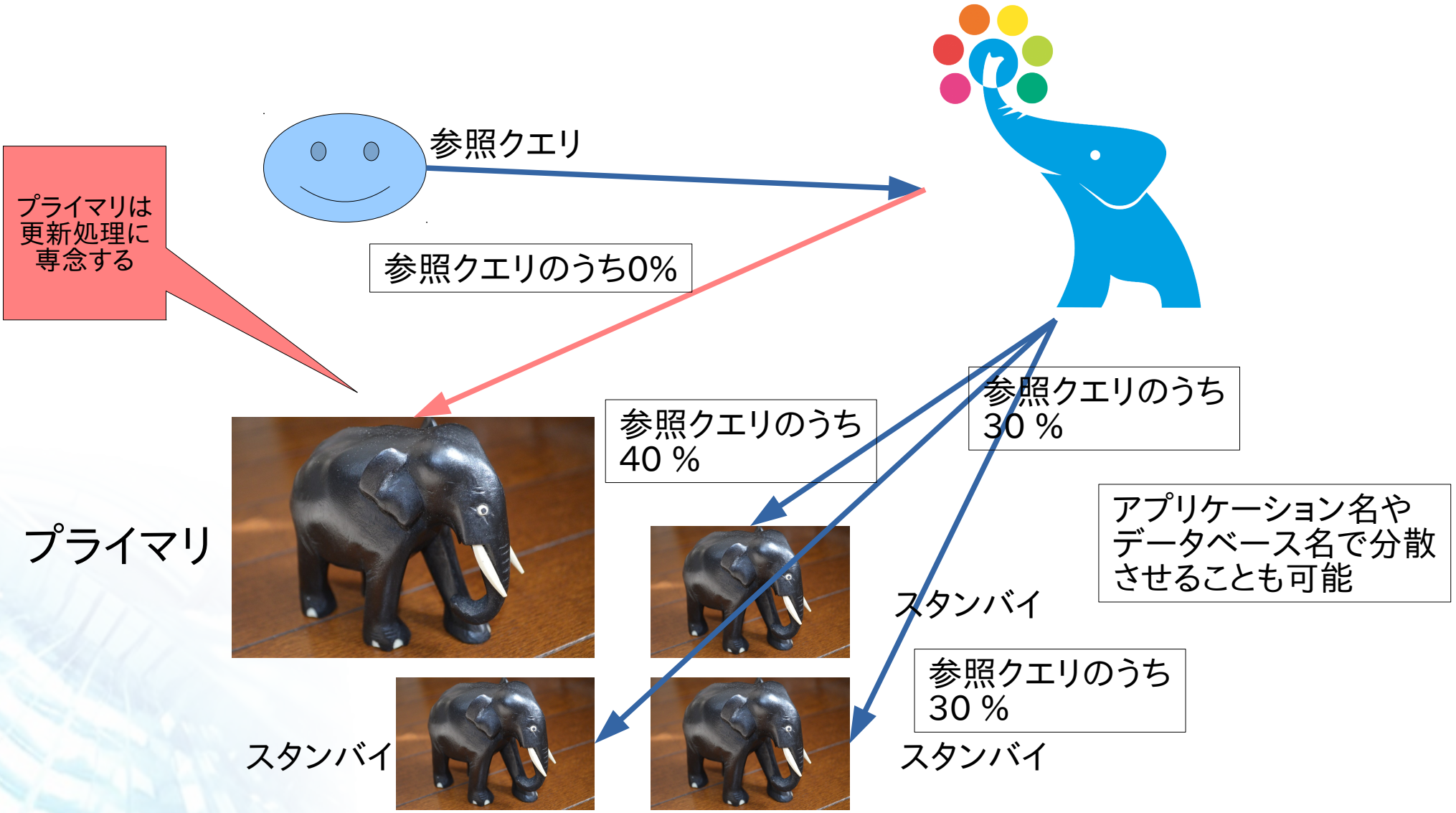
- PostgreSQLとクライアントの間に入るproxy型のミドルウェアで、OSSとして公開
- クライアントからは複数のPostgreSQLが、あたかも単一のPostgreSQLに見えるような管理を行う
- 自動フェイルオーバやクエリの振り分け、クエリキャッシュなど多くの機能
- 本資料では、PostgreSQLのストリーミングレプリケーションとの組み合わせを中心にお話します

クエリのディスパッチ/ルーティング

既存のDBアプリケーションの
変更は最小限



負荷分散



スタンバイサーバがダウンした時

スタンバイサーバがダウンしたら、
Pgpool-IIがそのことを検知し、
クラスタリングの対象から取り除く

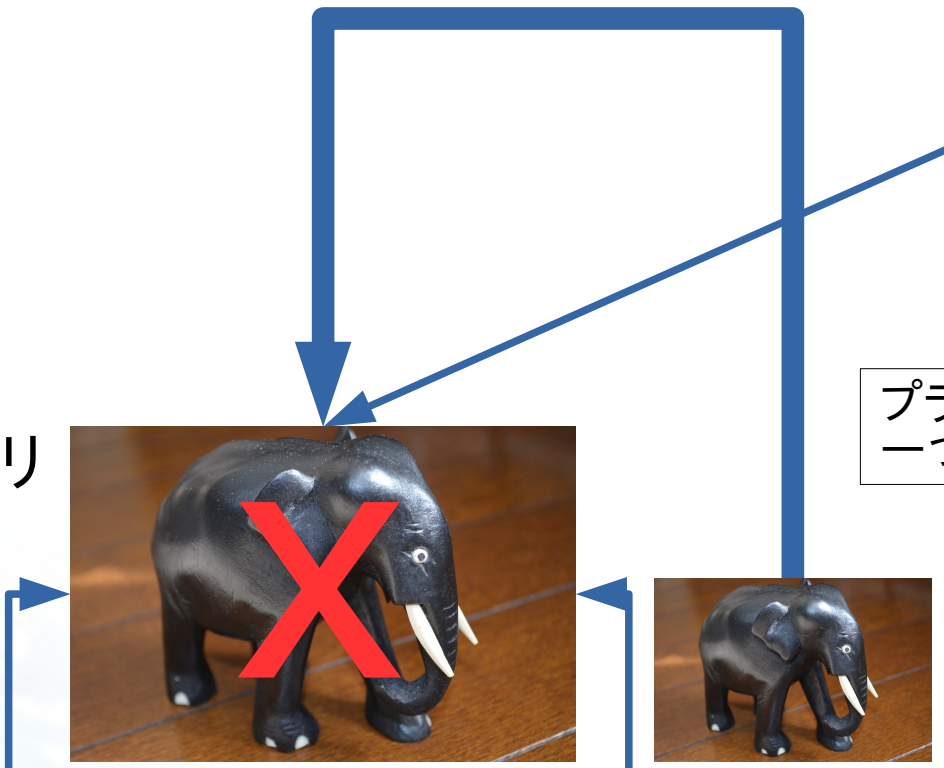
既存のセッションは再接続が
必要

プライマリ

スタンバイ

SRA OSS, INC.

プライマリサーバがダウンした時



プライマリがダウンすると、スタンバイの一つが昇格し、新しいプライマリになる

他のスタンバイは新しいスタンバイに追従するようになる

新しいスタンバイの追加

新しい象！

プライマリ

スタンバイ

新しいサーバは簡単に追加できる。
Pgpool-IIは新しいサーバにプライマリからデータをコピーし、他のサーバに影響を与えずにクラスタに新しいサーバを追加できる。
既存のセッションは切断されない。

SRA OSS, INC.

Watchdog: Pgpool-II組み込みの高可用性機能

アクティブPgpool

昇格

スタンバイPgpool

プライマリ

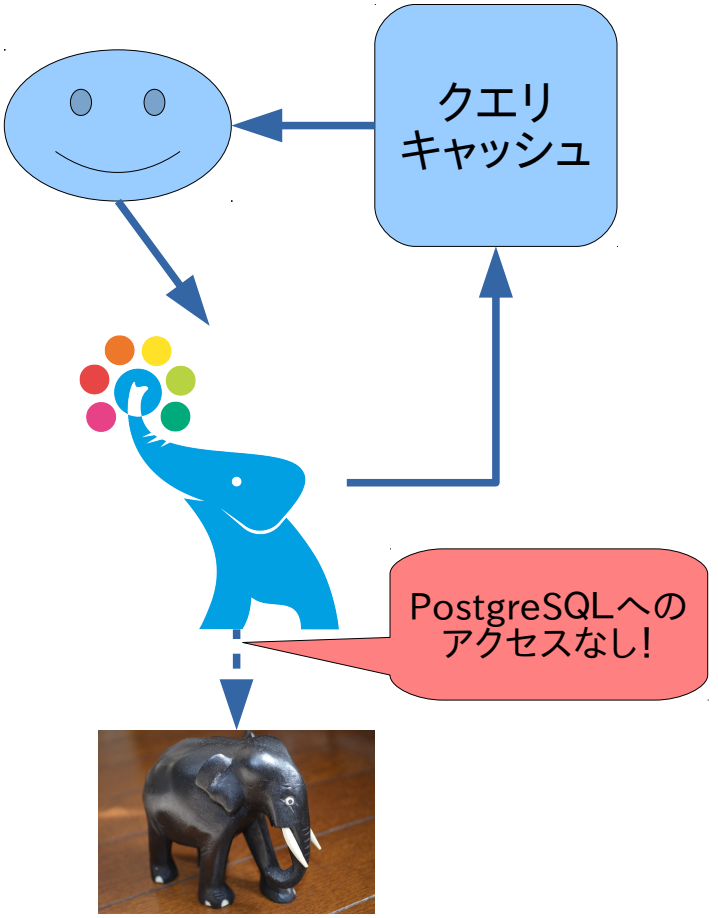
スタンバイ

アクティブPgpool-IIがダウンすると,
スタンバイPgpool-IIが昇格する

SRA OSS, INC.

インメモリクエリキャッシュ

- Pgpool-IIはクエリキャッシュを使ってクエリの結果を再利用する
- クエリキャッシュはメモリ上に置かれるので非常に高速
- その際にPostgreSQLアクセスは一切なし
- キャッシュ用のストレージは、共有メモリ火memcachedから選べる
- テーブルが更新されると、そのテーブルを参照したクエリキャッシュはすべて廃棄される
- タイムアウトベースのキャッシュ更新も可能

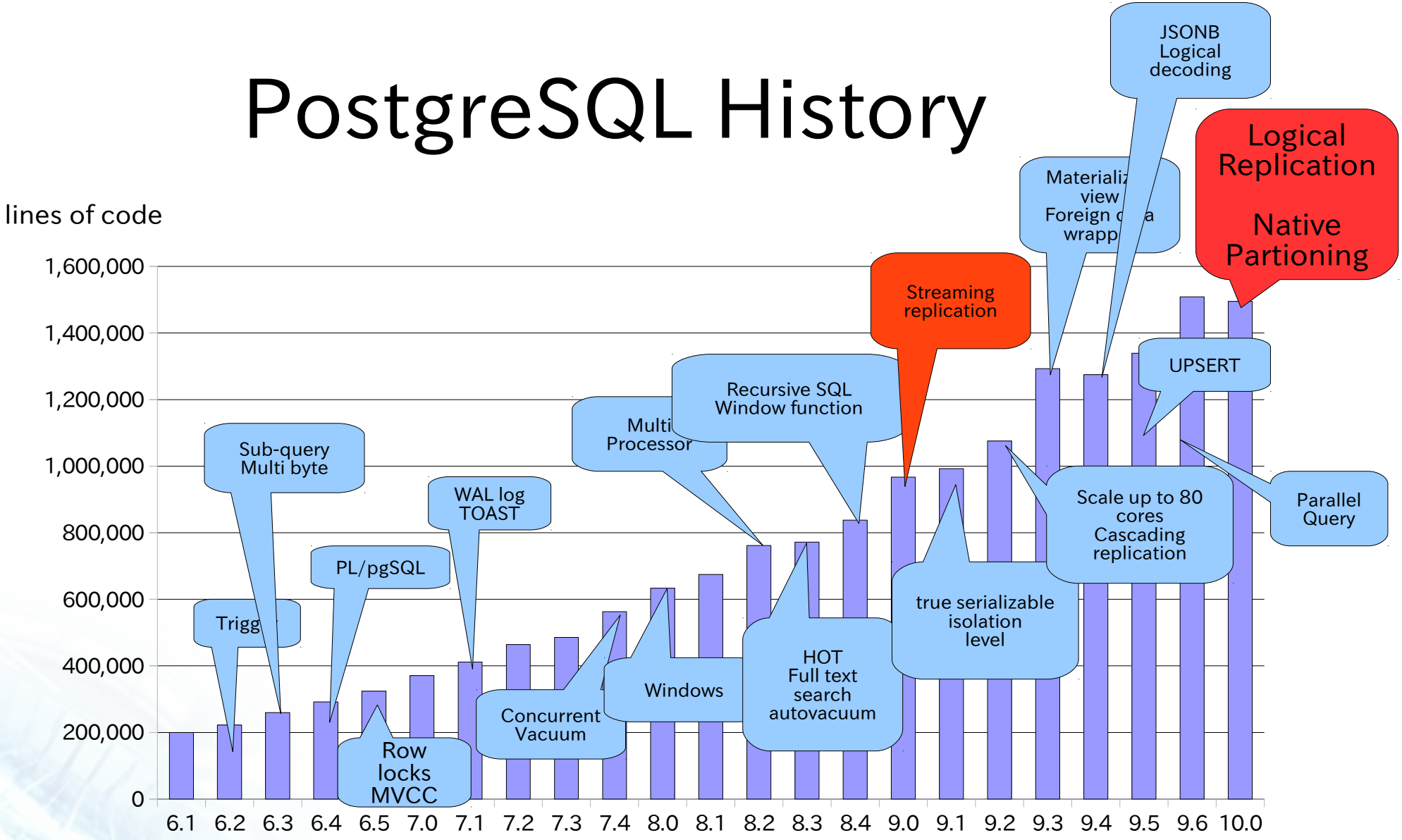


PostgreSQL 10登場!

PostgreSQLの20年間の成長

- 規模は7倍以上に
  - 20万ステップから150万ステップに
- 本格的なRDBMSに
  - トランザクション、行ロック、MVCC、SQL標準対応、本格的なクエリオプティマイザ
- 高性能化
  - マルチプロセッサ対応、パラレルクエリ、パーティショニング
- 高可用性化
  - レプリケーション
- 多機能化
  - 全文検索、JSON対応

PostgreSQL History



1996 | MySQLレプリケーション (2000) | Pgpool:|レプリケーション (2004) | 2010 | 2017

SRA OSS, INC.

PostgreSQL 10.0の主な新機能

- ロジカルレプリケーション
- 宣言型パーティショニング
- パラレルクエリの強化
- 同期レプリケーションの強化
- and more...

ロジカルレプリケーション

ロジカルレプリケーションとは？

- 従来からある、ストリーミングレプリケーションと同じように、マスターからスレーブにレプリケーションする機能
  - ただし、マスター側を “Publisher”、スレーブ側を “Subscriber” と呼ぶ
- ロジカルレプリケーションの特徴
  - ロジカル（論理的）な更新トランザクションログを転送する
  - 一部のテーブルのみレプリケーションすることができる
  - レプリケーション対象のテーブルには、主キーが付いていることが望ましい
    - なくてもレプリケーションできるが、転送ログが大きくなる
  - PostgreSQLのメジャーバージョンが異なってもレプリケーションできる
  - 一部レプリケーションされないSQLコマンドがある
    - DDL、TRUNCATE、シーケンス、ラージオブジェクト
    - COPYはINSERTとしてレプリケーション。大量のCOPYは遅くなる可能性あり(初期COPY除く)

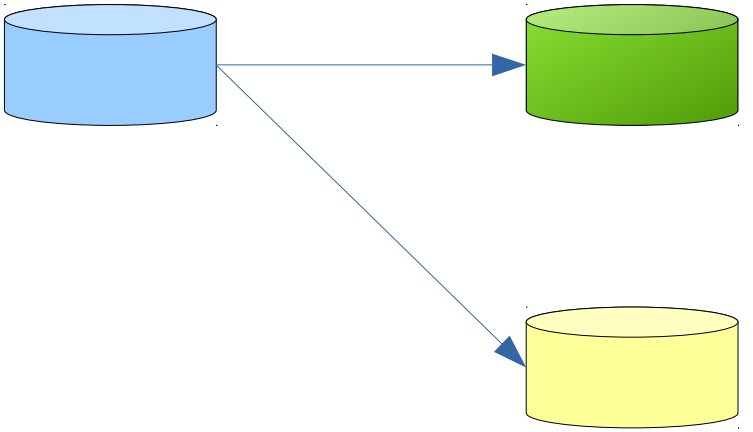
ストリーミングレプリケーションとの違い

- Publisher側もSubscriber側の通常のPostgreSQLで良い
  - Subscriber側はread onlyのstandbyではない
- Subscription対象のテーブルをSubscriber側でも変更できる
  - ただし、conflictが起きないようにユーザが管理しないといけないので、あまりお勧めできない
  - Publisher側とSubscriber側のテーブルの初期データが一致していなくても良い
    - 初期状態で、Subscriber側のテーブルが空でも良い

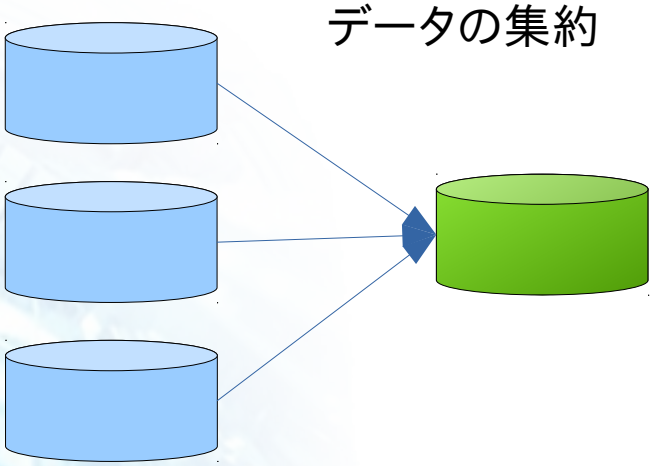
ロジカルレプリケーションのユースケース

バージョンアップ

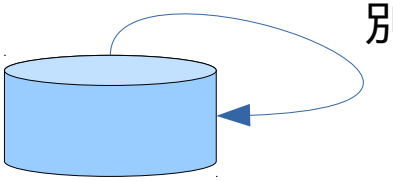
ストリーミングレプリケーション
との併用



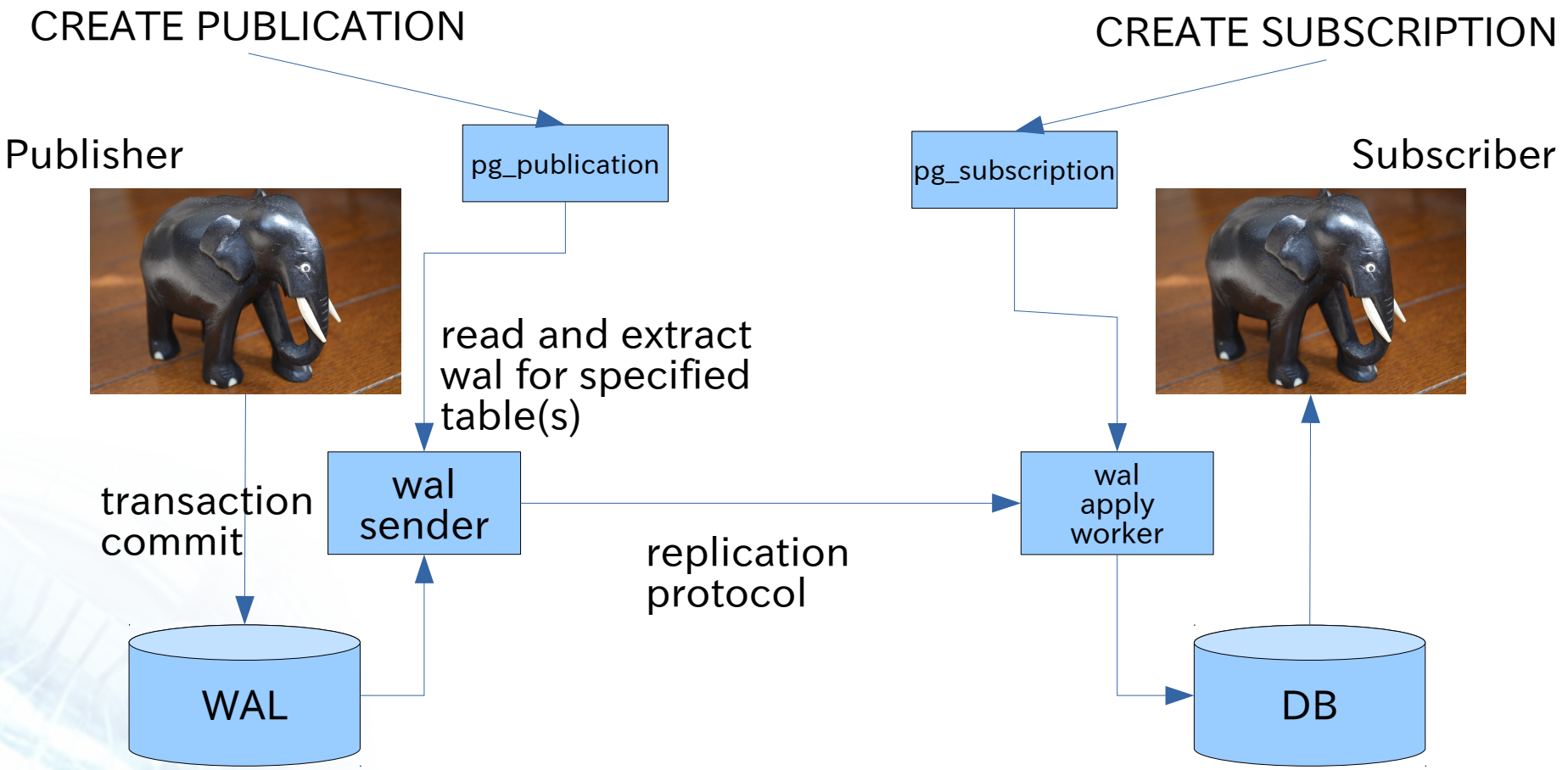
データの集約



同一サーバ内での
別DBへのレプリケーション

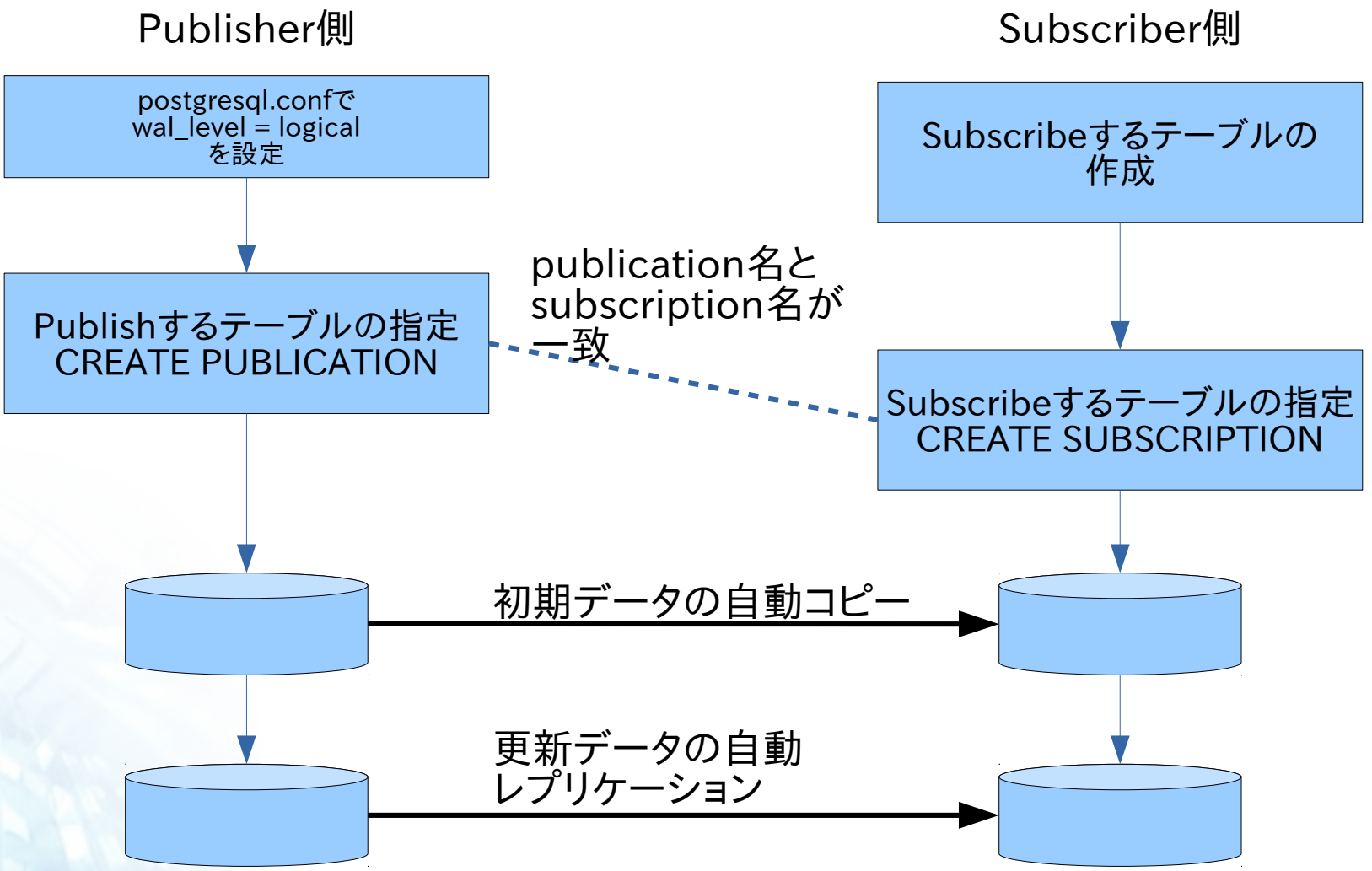


ロジカルレプリケーションの仕組み



SRA OSS, INC.

ロジカルレプリケーション利用の流れ



ストリーミングレプリケーションの住み分け?

- ストリーミングレプリケーションの使いどころ
  - すべてのテーブルをレプリケーションしたい
  - スレーブ側を決して更新してほしくない
- ロジカルレプリケーションの使いどころ
  - 一部のテーブルだけをレプリケーションしたい
  - 違うバージョンのPostgreSQLの間でレプリケーションしたい
  - レプリケーションするテーブルのスキーマ定義をコピー元とコピー先で変えたい
    - subscriberで列を追加するのはOK
  - コピー先でもデータ変更を行いたい
  - 複数のコピー元から一箇所のコピー先にデータを集約したい

ロジカルレプリケーションの制限事項と注意事項

- DDL、TRUNCATE、シーケンス、ラージオブジェクトはレプリケーションされない
- レプリケーションされるのは基底(base)テーブルに限る
  - VIEW、Materialized View、パーティションテーブルのRoot、foreign tableはレプリケーションされない
- 一時テーブル、UNLOGGEDテーブルはレプリケーションされない
- Subscriber側でもデータ更新ができるので、データの不整合が起きないようにアプリケーションで気を付ける必要がある

最新バージョン：Pgpool-II 3.7 が今秋リリースされます！

- ロジカルレプリケーション対応
- ヘルスチェックの改善
- watchdogの改善
- PostgreSQL 10パーサ対応
- Amazon Aurora対応

Pgpool-II 3.7 ロジカルレプリケーションへの対応

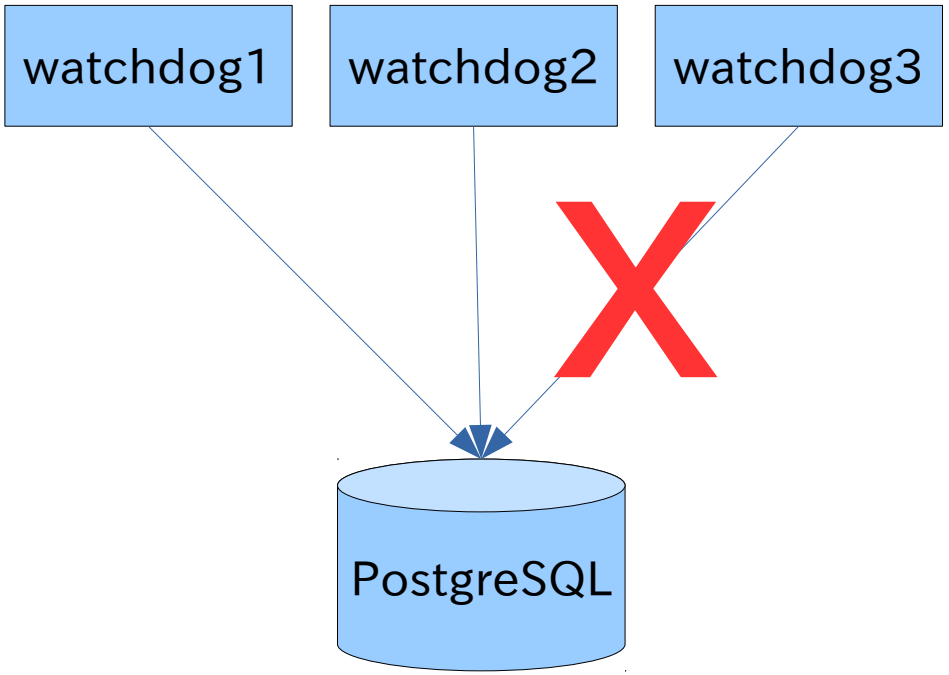
- 「ロジカルレプリケーションモード」の追加
  - 「マスターノード」を指定、それ以外を「スレーブノード」として認識
  - ノード間で検索負荷分散可能
  - テスト環境作成ツール「pgpool_setup」でロジカルレプリケーションをサポート

ヘルスチェックの改善

- PostgreSQLノード毎にヘルスチェックパラメータの設定が可能に
  - タイムアウト時間やチェック間隔時間など
- 並列にヘルスチェックを実行
  - ノードに同時に障害が発生した時にお互いに影響を受けない

watchdogの改善

- Quorum aware failover
  - PostgreSQLの障害に関して複数のwatchdogで合意を取る
  - 誤検知の低減



PostgreSQL 10.0 SQLパーサの移植

- Pgpool-IIでは、SQL文を正確に解析するためにSQLパーサを持っている
  - SQLパーサは、最新のPostgreSQLから移植
- 以下の新しい構文をサポート
  - 宣言型パーティショニング
    - CREATE TABLE … PATITION BY …
  - ロジカルレプリケーション
    - CREATE PUBLICATION
    - CREATE SUBSCRIPTION
  - 列間にまたがる統計情報
    - CREATE STATISTICS
  - など

Amazon Aurora対応

- Amazon Aurora with PostgreSQL Compatibilityは、AWSの提供する高性能PostgreSQLクラウドサービス
  - 独自のストレージ管理を行い、更新性能が高い
  - 複数のリードレプリカを持てる
  - 自動フェイルオーバ
- Pgpool-IIを使って更新クエリと検索要求を自動的にマスターノードとリードレプリカに振り分け
- コネクションプーリングやクエリキャッシュ機能も利用可能

URLなど

- PostgreSQL
  - http://www.postgresql.org
- Pgpool-II公式サイト
  - http://www.pgpool.net
- Postgres-XL
  - http://www.postgres-xl.org
- Citus
  - https://github.com/citusdata/citus

今年もやります、PGConf.ASIA!

- 2017/12/4-6
- 秋葉原コンベンションホール（去年と同じ場所）
- 間もなくプログラム公開、チケット販売開始！
- http://pgconf.asia

Thank you!