PGECons
PostgreSQL Enterprise Consortium

設計運用WG 2014年度活動報告

－企業が求める堅牢なデータベースを実現するために－
～ディザスタ・リカバリとセキュリティ～

PostgreSQL エンタープライズ・コンソーシアム
WG3（設計運用WG）

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

アジェンダ

- WG3について
  - WG3の位置づけ、活動の概要
  - 2014年度の活動テーマ、実施体制
- 可用性テーマの成果紹介
  - 災害対策とディザスタ・リカバリ(DR)
  - DRを実現するシステム構成
  - 運用技術検証
- セキュリティテーマの成果紹介
  - PCI DSSのDB関連項目要件調査
  - PostgreSQLの適用技法
- エンディング
  - 2014年活動を振り返って

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

WG3の位置づけ、活動概要

- 技術部会で設定された課題に具体的に取り組む実働部隊の一つで主に設計・運用面に関する課題をテーマとする。
  - ミッションクリティカル性の高いエンタープライズ領域へのPostgreSQL適用に向けて、安定稼動のための技術ノウハウを提供

PGEConsにおける課題領域

性能	性能評価手法、性能向上手法、チューニングなど
可用性	高可用クラスタ、BCP
保守性	保守サポート、トレーサビリティ
運用性	バックアップ運用、監視運用
セキュリティ	監査
互換性	データ、スキーマ、SQL、ストアドプロシージャの互換性
接続性	他ソフトウェアとの連携

昨年度のテーマは高可用クラスタ、バックアップ、監視

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

WG3の位置づけ、活動概要

- 技術部会で設定された課題に具体的に取り組む実働部隊の一つで主に設計・運用面に関する課題をテーマとする。
  - ミッションクリティカル性の高いエンタープライズ領域へのPostgreSQL適用に向けて、安定稼動のための技術ノウハウを提供

PGEConsにおける課題領域

性能	性能評価手法、性能向上手法、チューニングなど
可用性	高可用クラスタ、BCP
保守性	保守サポート、トレーサビリティ
運用性	バックアップ運用、監視運用
セキュリティ	監査
互換性	データ、スキーマ、SQL、ストアドプロシージャの互換性
接続性	他ソフトウェアとの連携

今年度はBCP、セキュリティに着目

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

WG3 2014年度活動テーマ

- 可用性（昨年度からの継続テーマ）
  - 昨年度：PostgreSQLで可用性を担保するシステム構成(HA・レプリケーション)の特徴を整理し、実機検証まで実施
  - 今年度：災害対策を想定し、検討範囲をDRサイト構成に発展

- セキュリティ（今年度からの新設テーマ）
  - 利用できる機能の整理と実用レベルのバランス調査
  - 他DBMS向けに記述されたDBSCによる文書を参考に、PCIDSS準拠のためのPostgreSQLにおける対処策をまとめる。
  - ※DBSC: DataBase Security Consortium

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

WG3 2014年度活動内容と活動体制

【可用性】

活動内容

- DRサイトをPostgreSQLで構築する場合の構成と特徴を机上調査で整理
  - 遠隔地バックアップ
  - ストレージレベルでのデータ同期
  - PostgreSQLストリーミング・レプリケーション(SR)
- SRを使ったDRサイトの実機検証
  - DRサイトの初期構築、障害時の運用手順を整備
  - Amazon Web Service環境で遠隔拠点間のレプリケーションを検証

活動体制 （企業名アイウエオ順、敬称略）

- TIS株式会社
- 日本電信電話株式会社
- 株式会社日立ソリューションズ

【セキュリティ】

活動内容

- PostgreSQL版『データベースセキュリティガイドライン』を作成
  - DBSCによる『データベースセキュリティガイドライン 第2.0版』の「製品別機能対応表」をチェック
  - PCI DSSv2.0での要求項目に対し、PostgreSQLでの実現手法を整理
  - PostgreSQLの標準機能で実現できない箇所は、外部ソフトの利用やスクリプト記述などを使った対策方法を例示

活動体制 （企業名アイウエオ順、敬称略）

- 株式会社アシスト
- SRA OSS, Inc. 日本支社
- NTTソフトウェア株式会社
- サイオステクノロジー株式会社
- 大日本印刷株式会社
- 日本電気株式会社

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

可用性テーマの成果紹介

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

災害対策とディザスタリカバリ

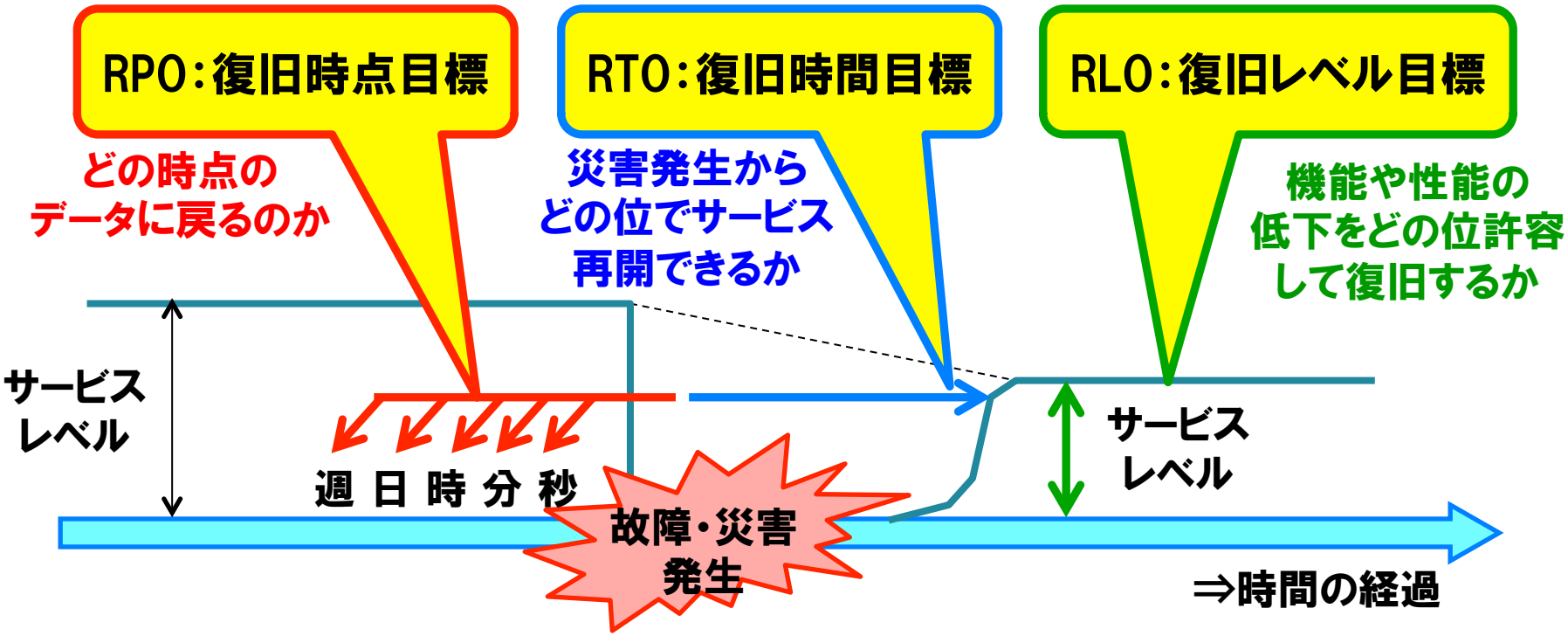
- ITシステムの停止が与える経済／社会的影響は大
  - システム停止による利益の損失を最小限に抑える対策が必要 ⇒ 災害対策
- ディザスタリカバリ（Disaster Recovery：DR）
  - 災害等で被害を受けたシステムを復旧・修復すること。また、そのための備えとなる機器やシステム、体制
- DRのサービスレベルは高ければよいわけではない。
  - システム規模や特性に応じて相応のコストがかかる。
  - いつくるかわからない災害にどの程度投資して備えるか？

ITシステムのサービス継続レベルを検討しておくことが DR のポイント

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを検討する指標

主要な3つの「目標」
- 復旧時間目標　：RTO（Recovery Time Objective）
- 復旧時点目標　：RPO（Recovery Point Objective）
- 復旧レベル目標：RLO（Recovery Level Objective）



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

構成		データ保全				サービス継続				
		構成1	構成2	構成3	構成4	構成5	構成6	構成7	構成8	構成9
		フル バックアップ ＆ データ 保管のみ	差分 バックアップ ＆ データ 保管のみ	H/W(ストレージ) で レプリケー ション	S/W (DRBD)で レプリケー ション	フルバック アップ＆ 事前リストア	差分バック アップ＆ 事前リストア	マスタ⇒遠 隔地DB 非同期レプ リケーション	マスタ⇒遠 隔地DB 部分レプリ ケーション	マスタ⇒ローカル スタンバイ⇒遠 隔地DB カスケードレプリ ケーション
構成パターン		シングル	シングル	Active- Standby	Active- Standby	Active- Standby	Active- Standby	Active- Active	Active- Active	Active-Active
復旧 目標	RTO 目標時間	長い	長い	長い	中	長い	長い	短い	短い	短い
	RPO 復旧ポイント	長い(バック アップ時点)	中(バック アップ時点)	短い(方式 次第)	短い	長い(バック アップ時点)	中(バック アップ時点)	短い	短い	短い
	RLO 復旧レベル	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	低	(方式 次第)
コスト(初期・構築・運用費用等)		低	低	高	中	低	低	中	中	中
運用性		楽	楽	楽	中	楽	楽	中	中	難
構築期間		短い	短い	短い	長い	短い	短い	長い	長い	長い
DRサイト活用		×	×	×	×	△	△	○	○	○
実機検証対象		ー	ー	ー	ー	ー	ー	○	ー	○

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

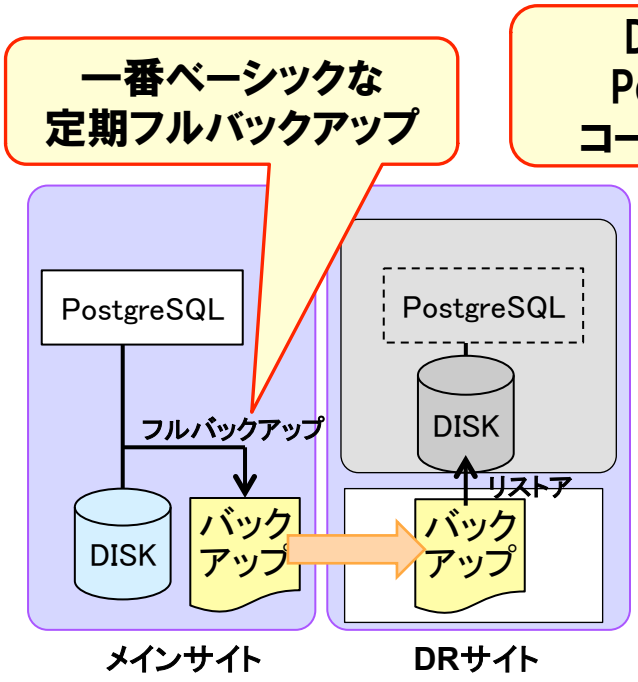
DRを実現するシステム構成

構成		データ保全				サービス継続				
		構成1	構成2	構成3	構成4	構成5	構成6	構成7	構成8	構成9
		フルバックアップ＆データ保管のみ	差分バックアップ＆データ保管のみ	I/W(ストレージ)でレプリケーション	S/W(DRBD)でレプリケーション	フルバックアップ＆事前リストア	差分バックアップ＆事前リストア	マスタ⇒遠隔地DB非同期レプリケーション	マスタ⇒遠隔地DB部分レプリケーション	マスタ⇒ローカルスタンバイ⇒遠隔地DBカスケードレプリケーション
構成パターン		シングル	シングル	Active-Standby	Active-Standby	Active-Standby	Active-Standby	Active-Active	Active-Active	Active-Active
復旧目標	RTO 目標時間	長い	長い	長い	中	長い	長い	短い	短い	短い
	RPO 復旧ポイント	長い(バックアップ時点)	中(バックアップ時点	短い(方式次第)	短い	長い(バックアップ時点)	中(バックアップ時点)	短い	短い	短い
	RLO 復旧レベル	(方式次第)	(方式次第)	(方式次第)	(方式次第)	(方式次第)	(方式次第)	(方式次第)	低	(方式次第)
コスト(初期・構築・運用費用等		低	低	高	中	低	低	中	中	中
運用性		楽	楽	楽	中	楽	楽	中	中	難
構築期間		短い	短い	短い	長い	短い	短い	長い	長い	長い
DRサイト活用		×	×	×	×	△	△	○	○	○
実機検証対象		ー	ー	ー	ー	ー	ー	○	ー	○

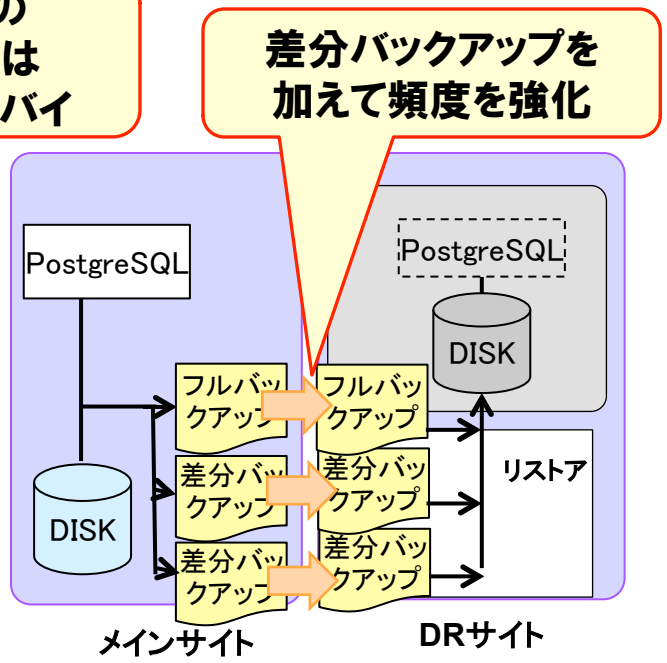
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

- 遠隔地バックアップでデータ保全を実現するパターン
  - 復旧にはバックアップのリストア、サービス起動が必要（RTO⬇）
  - 復旧後のデータ鮮度はバックアップ取得頻度に依存 （RPO⬇）



構成1：フルバックアップ＆
データ保管のみ



構成2：差分バックアップ＆
データ保管のみ

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

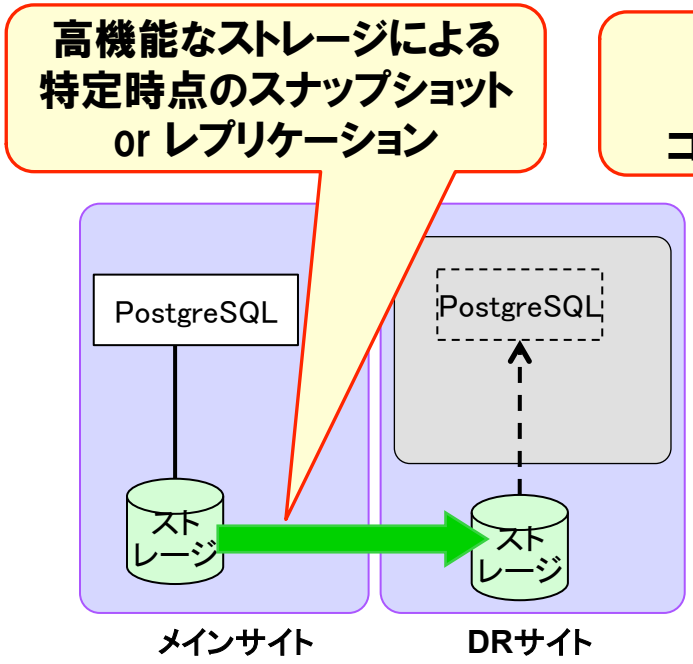
DRを実現するシステム構成

構成		データ保全				サービス継続				
		構成1	構成2	構成3	構成4	構成5	構成6	構成7	構成8	構成9
		フル バックアップ & データ 保管のみ	差分 バックアップ & データ 保管のみ	H/W(ストレージ) で レプリケー ション	S/W (DRBD)で レプリケー ション	フルバック アップ& 事前リストア	差分バック アップ& 事前リストア	マスタ⇒遠 隔地DB 非同期レプ リケーション	マスタ⇒遠 隔地DB 部分レプリ ケーション	マスタ⇒ローカル スタンバイ⇒遠 隔地DB カスケードレプリ ケーション
構成パターン		シングル	シングル	Active- Standby	Active- Standby	Active- Standby	Active- Standby	Active- Active	Active- Active	Active-Active
復旧 目標	RTO 目標時間	長い	長い	長い	中	長い	長い	短い	短い	短い
	RPO 復旧ポイント	長い(バック アップ時点)	中(バック アップ時点)	短い(方式 次第)	短い	長い(バック アップ時点)	中(バック アップ時点)	短い	短い	短い
	RLO 復旧レベル	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	低	(方式 次第)
コスト(初期・構築・運用費用等)		低	低	高	中	低	低	中	中	中
運用性		楽	楽	楽	中	楽	楽	中	中	難
構築期間		短い	短い	短い	長い	短い	短い	長い	長い	長い
DRサイト活用		×	×	×	×	△	△	○	○	○
実機検証対象		ー	ー	ー	ー	ー	ー	○	ー	○

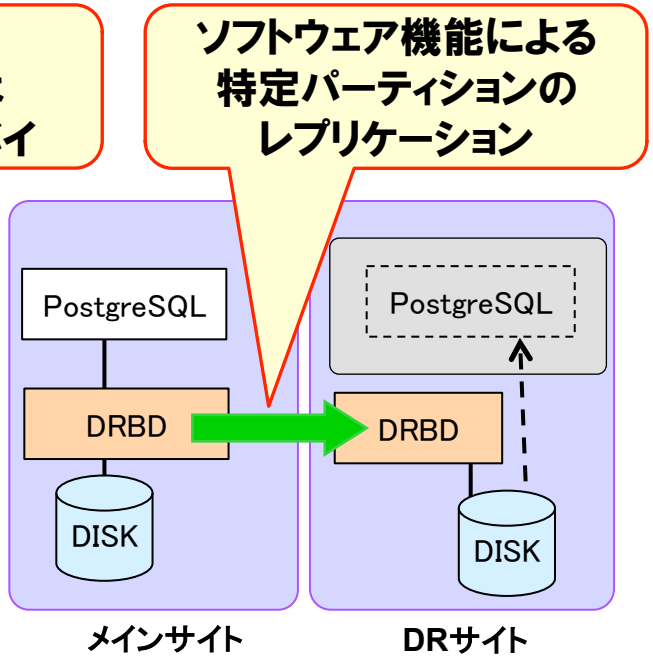
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

- H/W,S/Wのデータ同期でデータ保全を実現するパターン
  - 復旧にはバックアップのリストア、サービス起動が必要（RTO⬇）
  - リアルタイムなデータ同期により復旧後のデータ鮮度は向上（RPO⬆）



構成3：H/Wでレプリケーション



構成4：S/Wでレプリケーション

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

構成		データ保全				サービス継続				
		構成1	構成2	構成3	構成4	構成5	構成6	構成7	構成8	構成9
		フル バックアップ & データ 保管のみ	差分 バックアップ & データ 保管のみ	H/W(ストレージ) で レプリケー ション	S/W (DRBD)で レプリケー ション	フルバック アップ& 事前リストア	差分バック アップ& 事前リスト	マスタ⇒遠 隔地DB 非同期レプ リケーション	マスタ⇒遠 隔地DB 部分レプリ ケーション	マスタ⇒ローカル スタンバイ⇒遠 隔地DB カスケードレプリ ケーション
構成パターン		シングル	シングル	Active- Standby	Active- Standby	Active- Standby	Active- Standby	Active- Active	Active- Active	Active-Active
復旧 目標	RTO 目標時間	長い	長い	長い	中	長い	長い	短い	短い	短い
	RPO 復旧ポイント	長い(バック アップ時点)	中(バック アップ時点)	短い(方式 次第)	短い	長い(バック アップ時点)	中(バック アップ時点	短い	短い	短い
	RLO 復旧レベル	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	(方式 次第)	低	(方式 次第)
コスト(初期・構築・運用費用等)		低	低	高	中	低	低	中	中	中
運用性		楽	楽	楽	中	楽	楽	中	中	難
構築期間		短い	短い	短い	長い	短い	短い	長い	長い	長い
DRサイト活用		×	×	×	×	△	△	○	○	○
実機検証対象		―	―	―	―	―	―	○	―	○

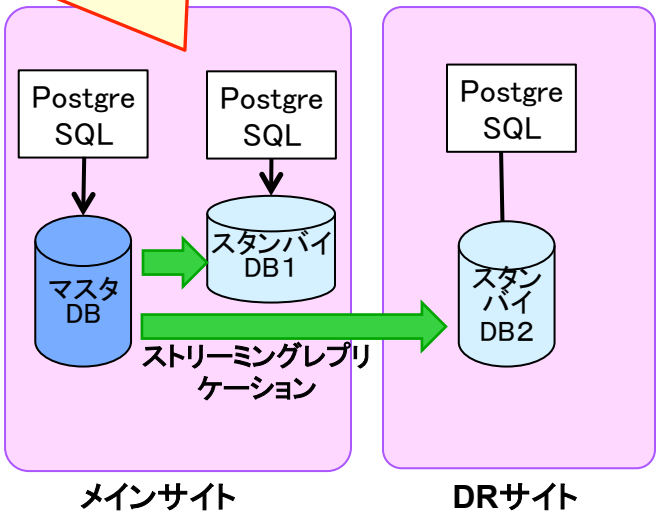
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

- PostgreSQLのSRによりサービス継続を実現するパターン
  - 平常時からDRサイトのPostgreSQLサービスは起動（RTO⬆）
  - リアルタイムなデータ同期により復旧後のデータ鮮度も向上（RPO⬆）
  - 平常時にDRサイトを参照用途で活用することも可能

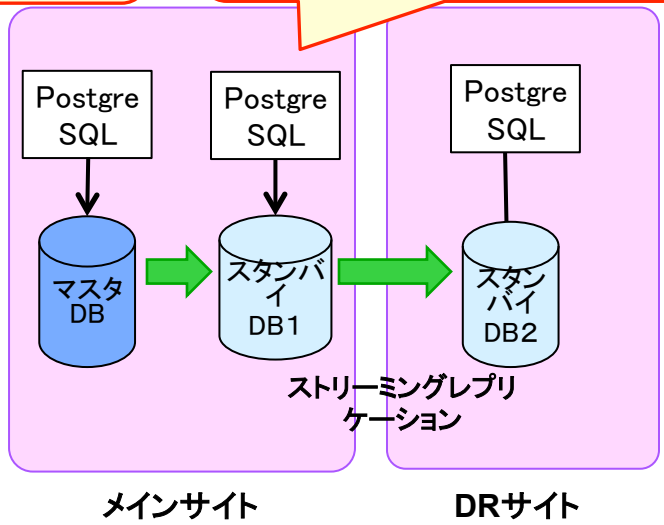
マルチスタンバイ構成の
ストリーミングレプリケーション
◎後半で検証！

DRサイト側の
PostgreSQLは
ホットスタンバイ



構成7：マスタ⇒遠隔地DB
非同期レプリケーション

カスケード構成の
ストリーミングレプリケーション
◎後半で検証！

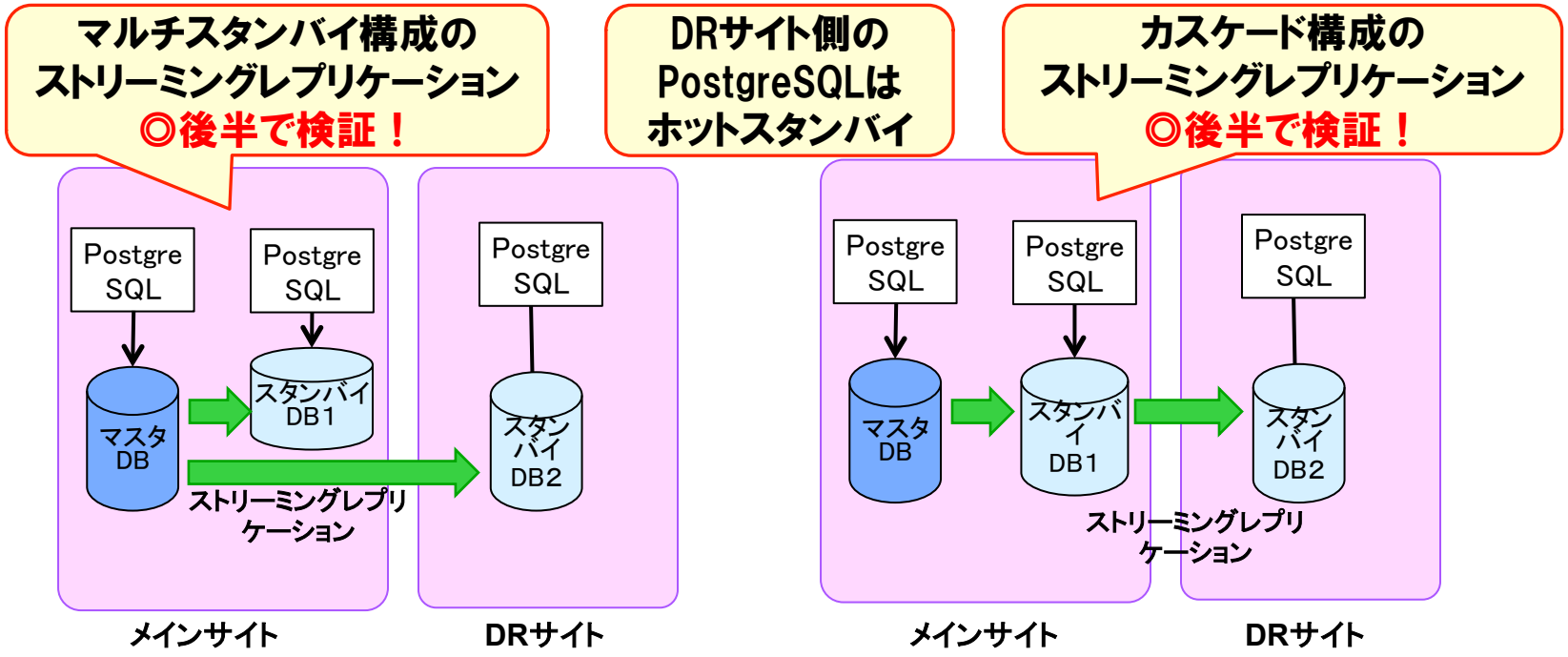


構成9：マスタ⇒ローカルスタンバイ
⇒遠隔地DBカスケードレプリケーション

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

- PostgreSQLのSRによりサービス継続を実現するパターン
  - 平常時からDRサイトのPostgreSQLサービスは起動（RTO⬆）
  - リアルタイムなデータ同期により復旧後のデータ鮮度も向上（RPO⬆）
  - 平常時にDRサイトを参照用途で活用することも可能



ストリーミングレプリケーションの運用性は実際どう？

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

運用技術検証
～構築手順と復旧手順～

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

DRを実現するシステム構成

検証対象の選定

構成		データ保全				サービス継続				
		構成1	構成2	構成3	構成4	構成5	構成6	構成7	構成8	構成9
		フル バックアップ & データ 保管のみ	差分 バックアップ & データ 保管のみ	H/W(ストレージ) で レプリケーション	S/W (DRBD)で レプリケーション	フルバック アップ& 事前リストア	差分バック アップ& 事前リストア	マスタ⇒遠隔地DB 非同期レプリケーション	マスタ⇒遠隔地DB 部分レプリケーション	マスタ⇒ローカルスタンバイ⇒遠隔地DB カスケードレプリケーション
構成パターン		シングル	シングル	Active-Standby	Active-Standby	Active-Standby	Active-Standby	Active-Active	Active-Active	Active-Active
復旧目標	RTO 目標時間	長い	長い	長い	中	長い	長い	短い	短い	短い
	RPO 復旧ポイント	長い(バックアップ時点)	中(バックアップ時点)	短い(方式次第)	短い	長い(バックアップ時点)	中(バックアップ時点)	短い	短い	短い
	RLO 復旧レベル	(方式次第)	(方式次第)	(方式次第)	(方式次第)	(方式次第)	(方式次第)	(方式次第)	低	(方式次第)
コスト(初期・構築・運用費用等)		低	低	高	中	低	低	中	中	中
運用性		楽	楽	楽	中	楽	楽	中	中	難
構築期間		短い	短い	短い	長い	短い	短い	長い	長い	長い
DRサイト活用		×	×	×	×	△	△	○	○	○
実機検証対象		ー	ー	ー	ー	ー	ー	○	ー	○

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

- 目的
  - PostgreSQLでDRサイトを構築する手順と運用手順を確認
- 観点
  - ストリーミングレプリケーションによるデータ同期でDRサイトを構築
  - 各レプリケーション構成毎にDRサイト運用の違いを考察
- 環境
  - 検証構成のメインサイト・DRサイトは仮想環境で構築

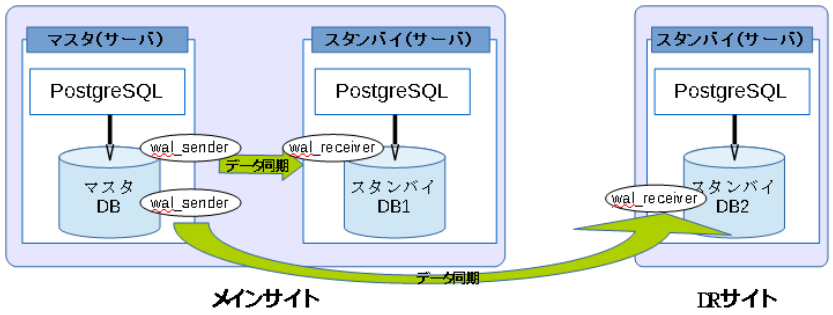


図.マルチスタンバイ構成

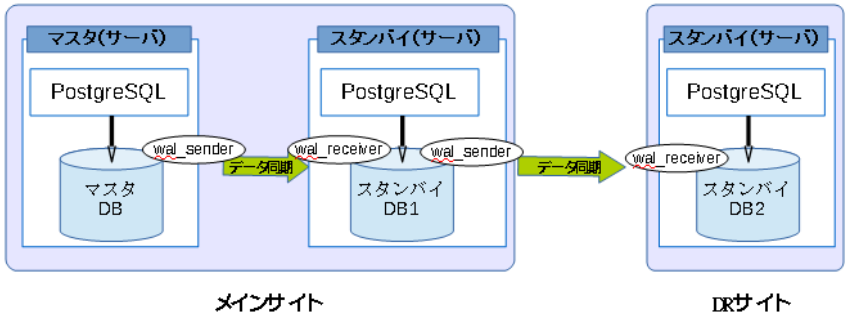
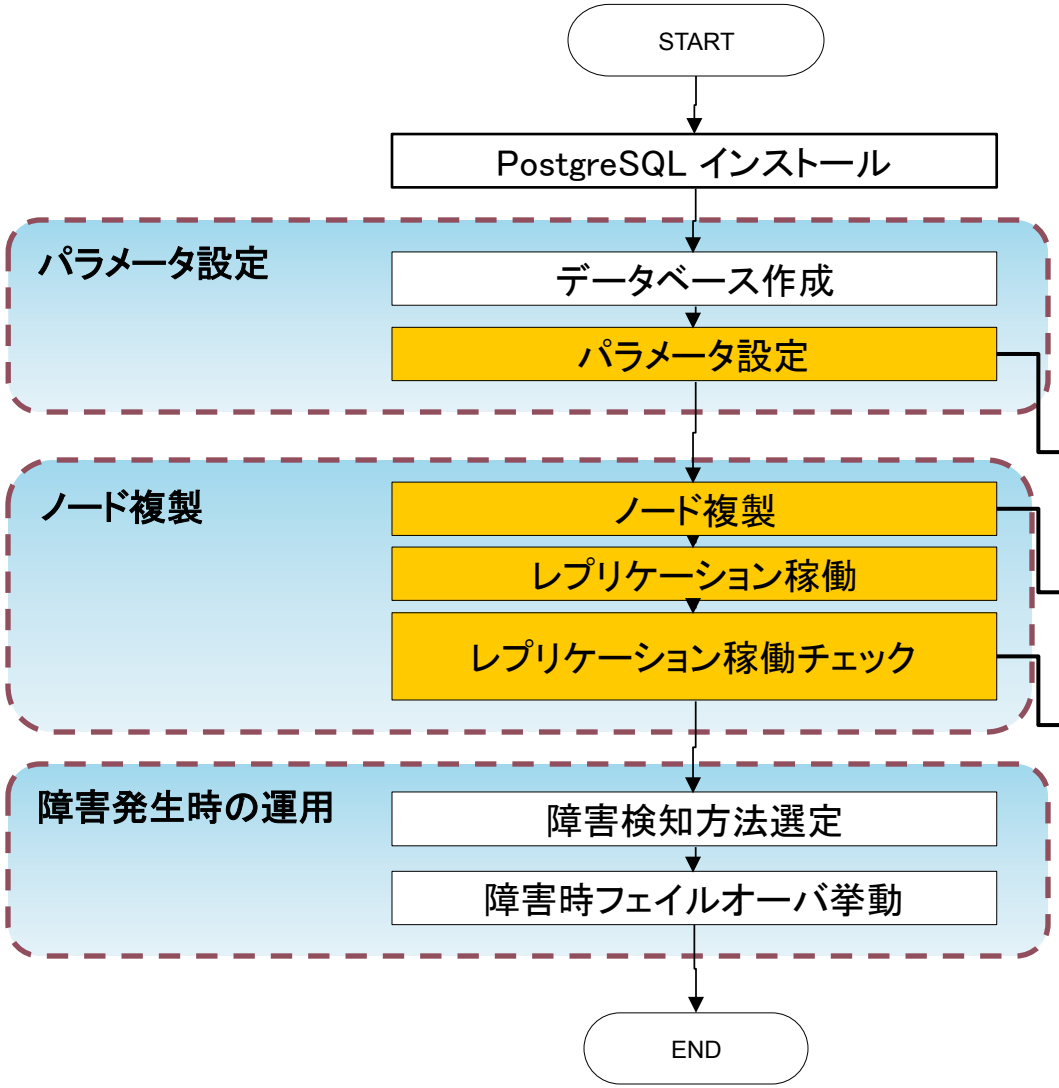


図.カスケード構成

実際にSRによるデータ同期するDRサイトを構築してどんな課題があるか確認してみよう！

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証内容1（構築手順）



各リリースの実装機能
- Ver. 9.0 ・ストリーミング・レプリケーションが採用
- Ver. 9.1 ・同期レプリケーション機能実装
- Ver. 9.2 ・synchronous_commit による 4種類の同期モードが可能
  ・カスケードレプリケーションが可能
- Ver. 9.3 ・タイムラインIDの追従
  ・高速フェイルオーバ機能
- Ver. 9.4 ・レプリケーションスロットの導入
  ・ロジカルデコーディングのサポート

wal_level = hot_standby
max_wal_senders = 4
hot_standby = on

※その他の設定パラメータは2014年度成果物ドキュメントをご確認ください

pg_basebackup

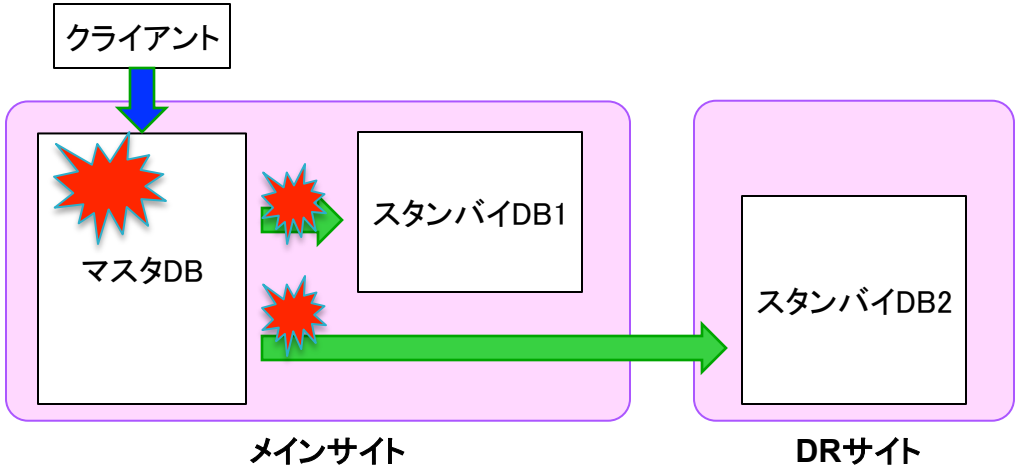
pg_xlog_location_diff
pg_stat_replication

構築手順は成果物としてドキュメント化しました ご参考ください。

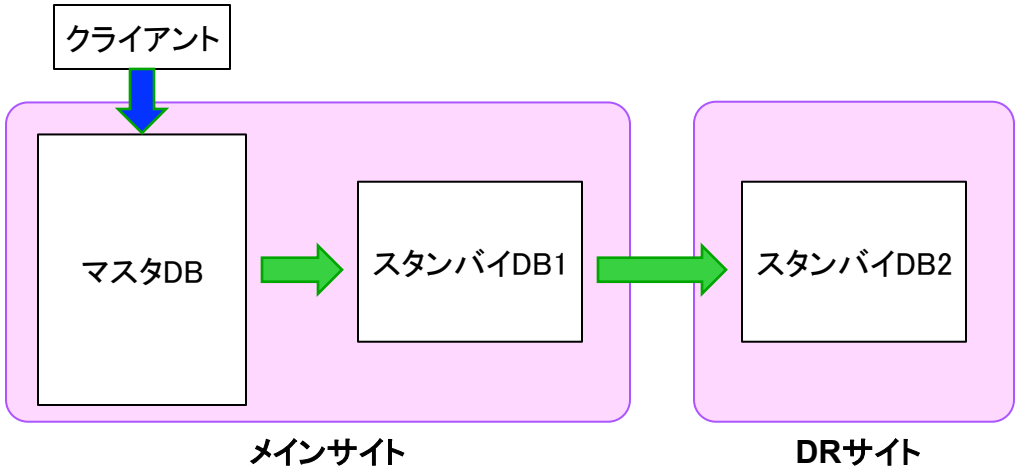
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証内容2(マスタDB障害からの復旧)

構成7:マスタ⇒遠隔地DB 非同期レプリケーション



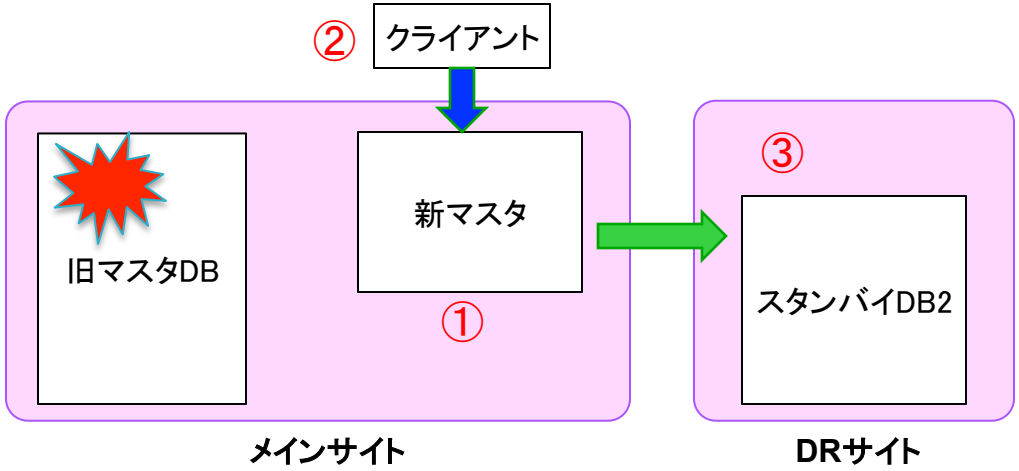
構成9:マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

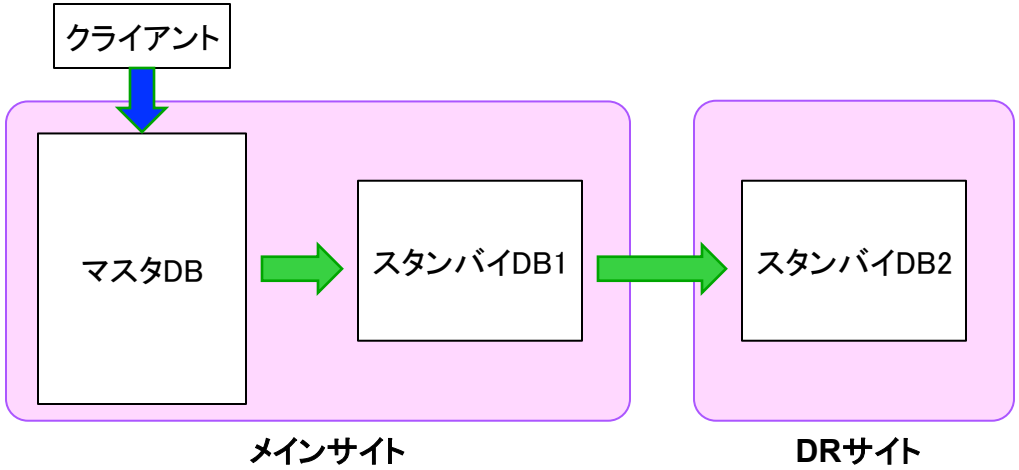
検証内容2(マスタDB障害からの復旧)

構成7:マスタ⇒遠隔地DB 非同期レプリケーション



① promoteコマンドでマスタ昇格
② クライアントの更新先を変更
③ recovery.confを書き換えて再起動

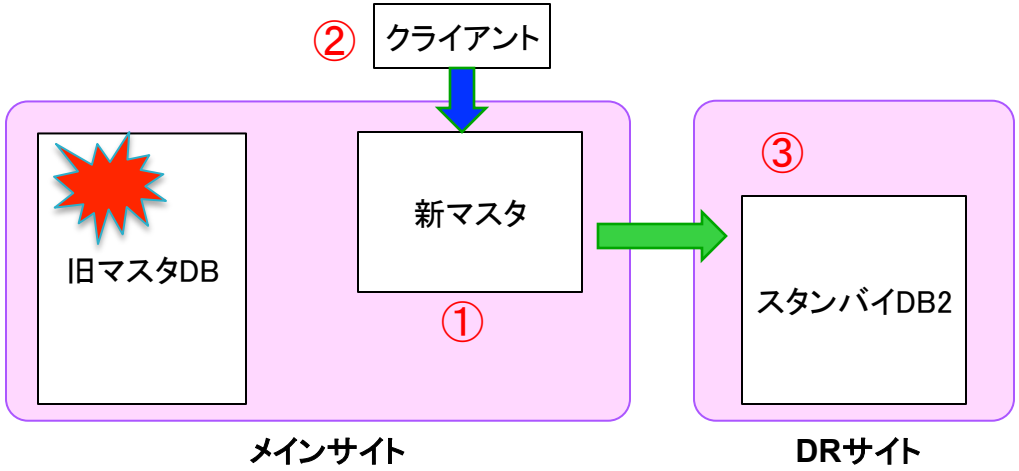
構成9:マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

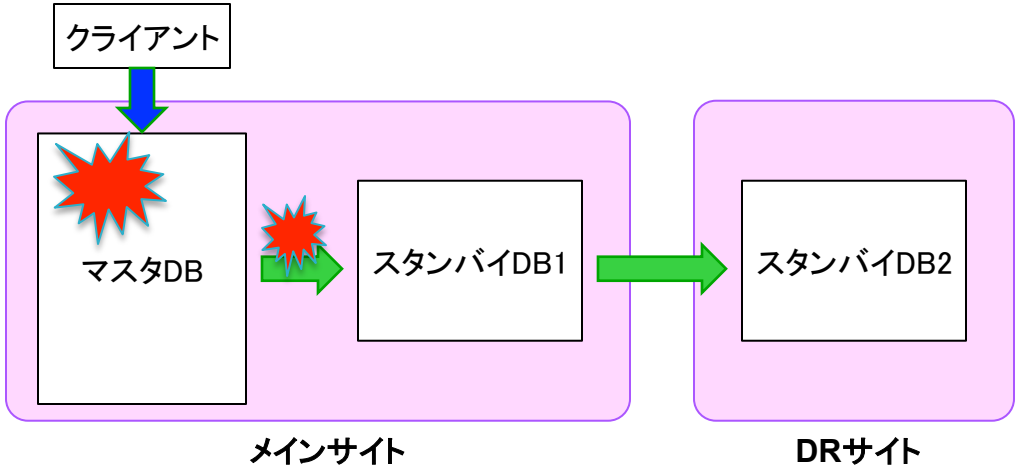
検証内容2(マスタDB障害からの復旧)

構成7:マスタ⇒遠隔地DB 非同期レプリケーション



① promoteコマンドでマスタ昇格
② クライアントの更新先を変更
③ recovery.confを書き換えて再起動

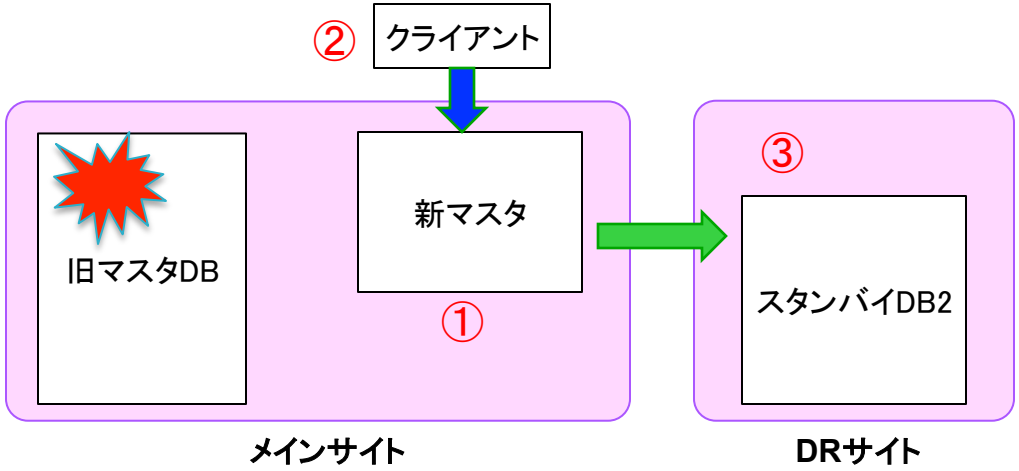
構成9:マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

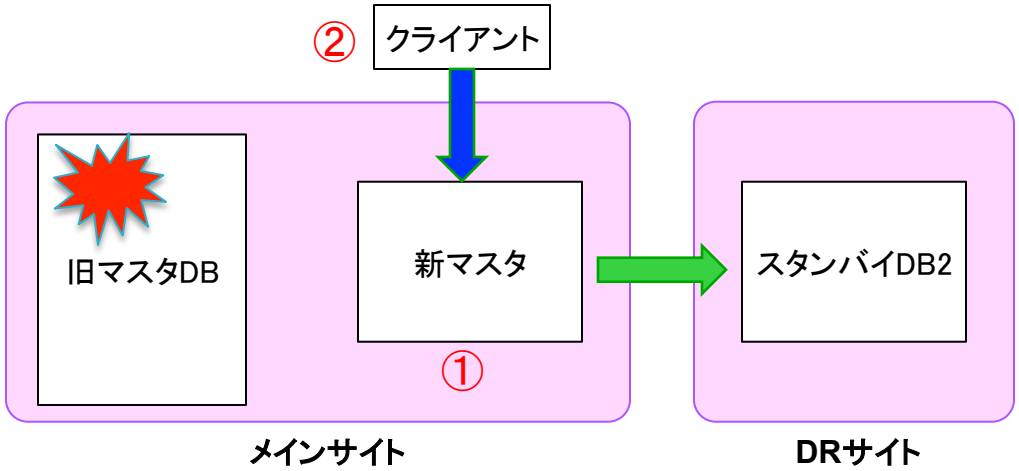
検証内容2(マスタDB障害からの復旧)

構成7:マスタ⇒遠隔地DB 非同期レプリケーション



① promoteコマンドでマスタ昇格
② クライアントの更新先を変更
③ recovery.confを書き換えて再起動

構成9:マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



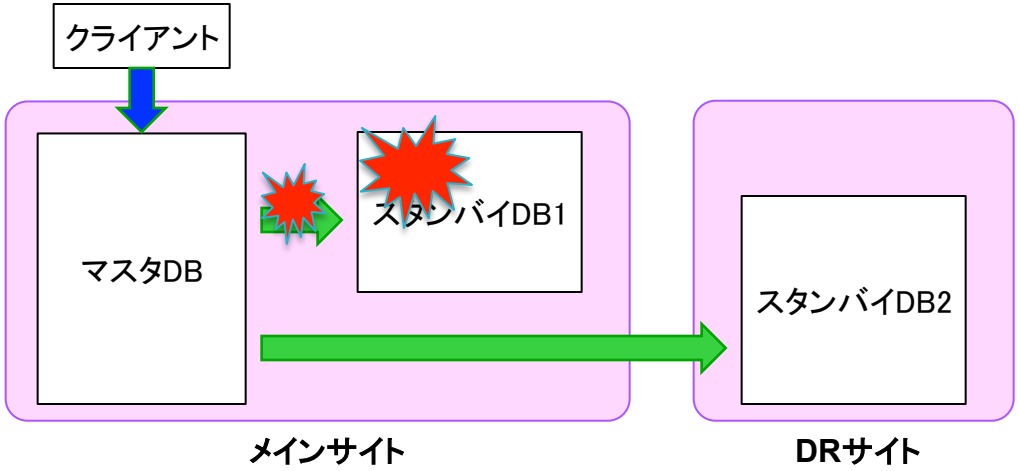
① promoteコマンドでマスタ昇格
② クライアントの更新先を変更

※ DRサイト側は操作不要

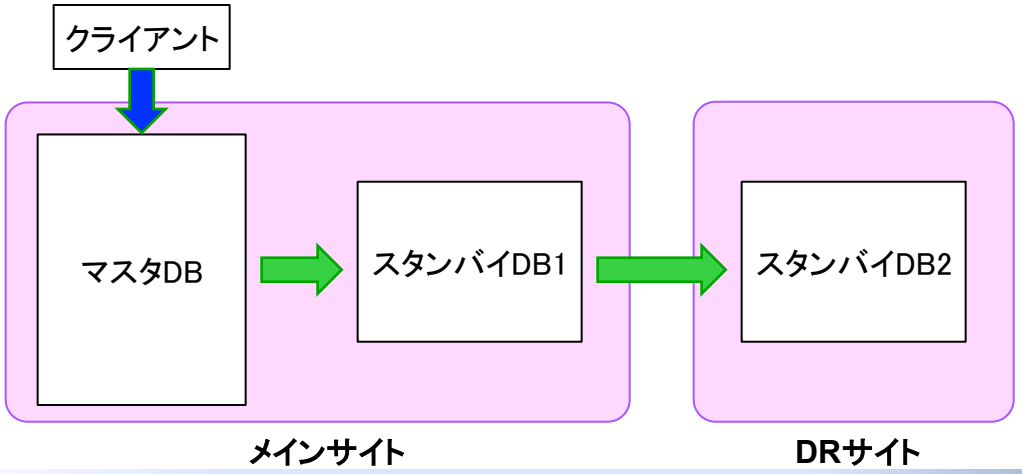
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証内容2
（メインサイトスタンバイDB障害からの復旧）

構成7：マスタ⇒遠隔地DB 非同期レプリケーション



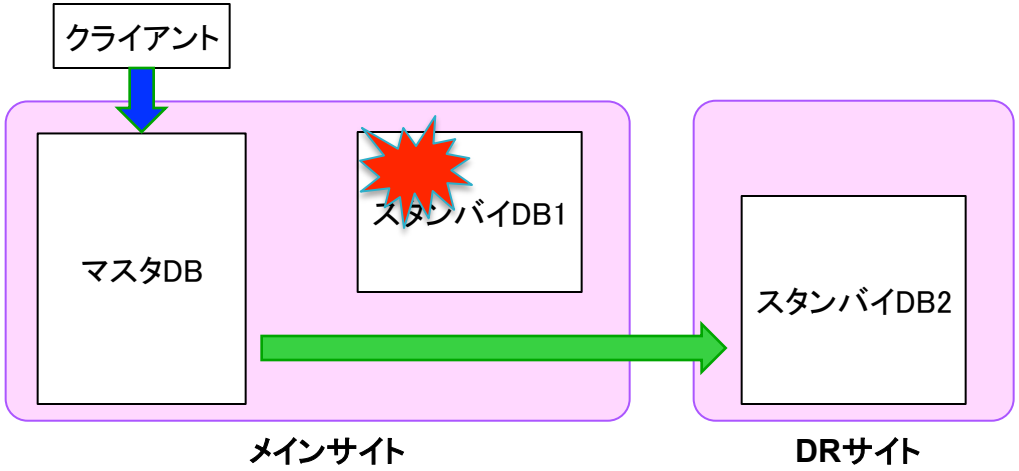
構成9：マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

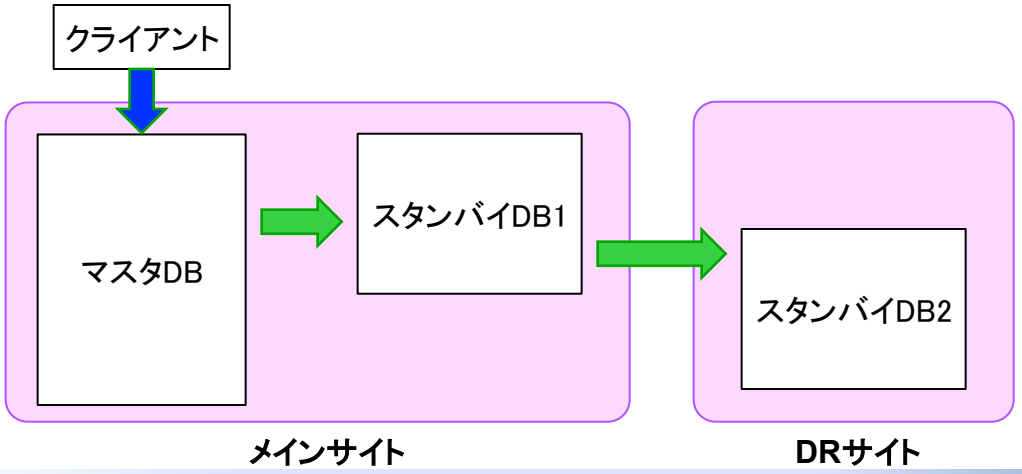
検証内容2
（メインサイトスタンバイDB障害からの復旧）

構成7：マスタ⇒遠隔地DB 非同期レプリケーション



スタンバイDBを急いで復旧しなくても、
大きな影響なし。

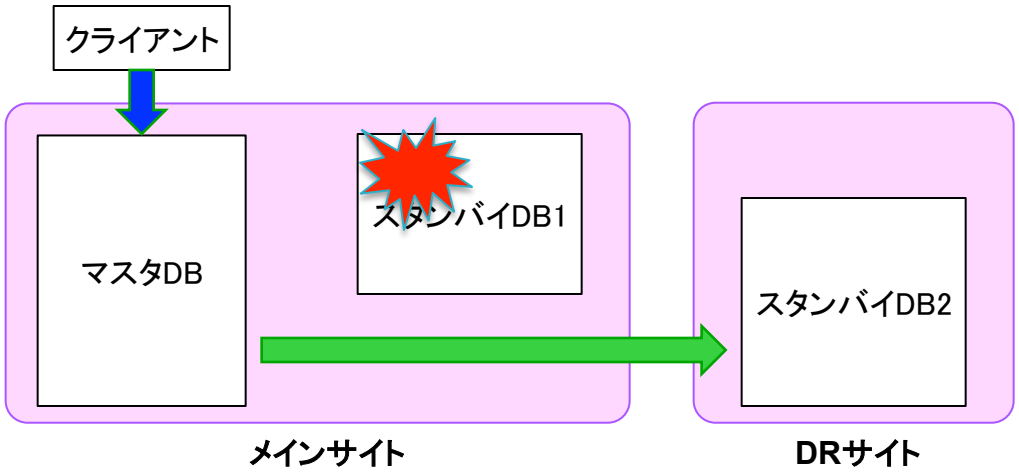
構成9：マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

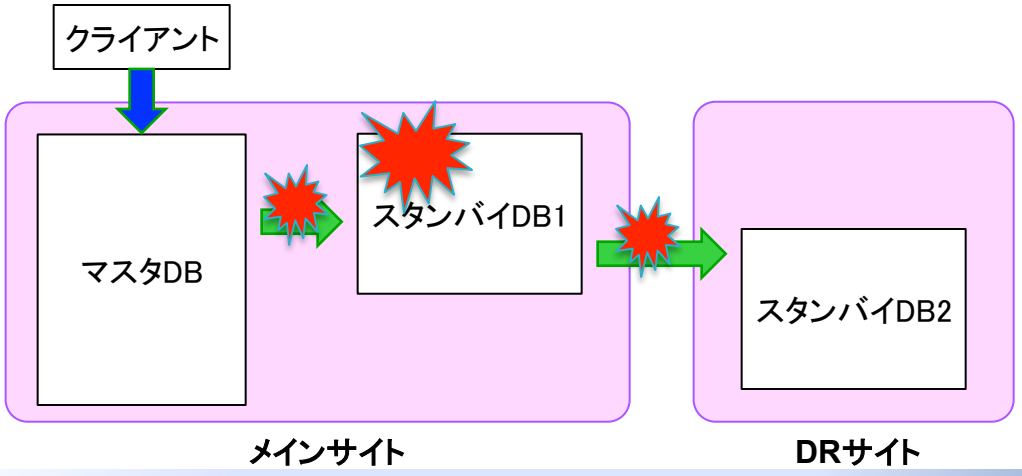
検証内容2
（メインサイトスタンバイDB障害からの復旧）

構成7：マスタ⇒遠隔地DB 非同期レプリケーション



スタンバイDBを急いで復旧しなくても、大きな影響なし。

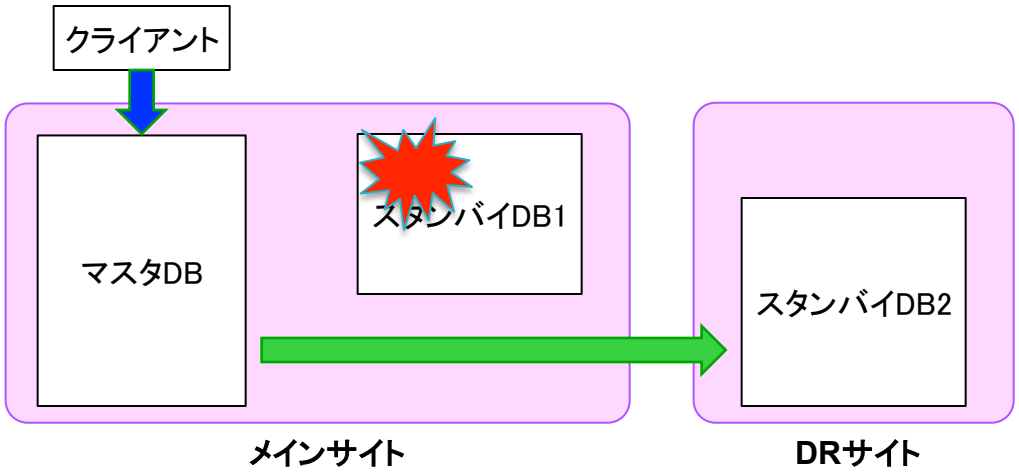
構成9：マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

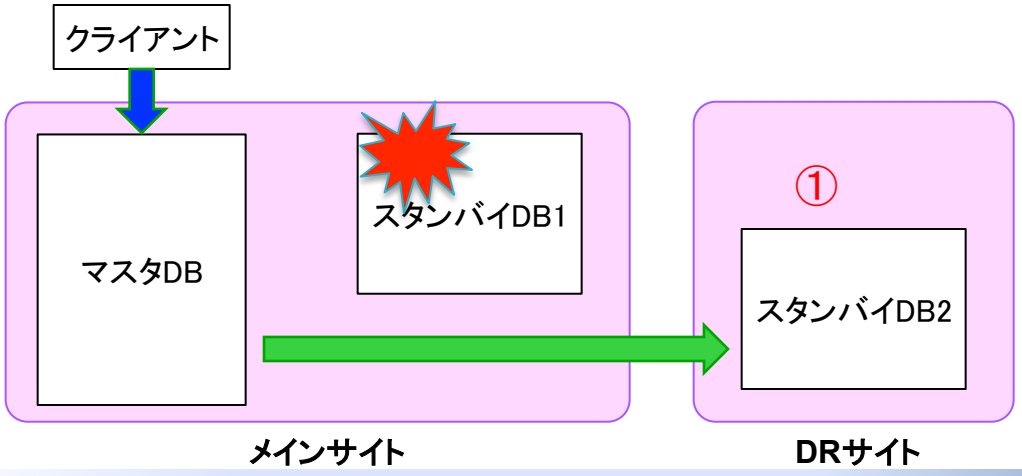
検証内容2
（メインサイトスタンバイDB障害からの復旧）

構成7：マスタ⇒遠隔地DB 非同期レプリケーション



スタンバイDBを急いで復旧しなくても、
大きな影響なし。

構成9：マスタ⇒メインサイト側スタンバイ⇒遠隔地DBカスケードレプリケーション



① recovery.confを書き換えて再起動

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証から得られた知見

- SRでデータ同期するDRサイトは**簡単**に構築できる。
- 構成パターン、障害箇所により復旧手順は異なるので、環境に合わせたリカバリ手順を**整備**しておくこと。

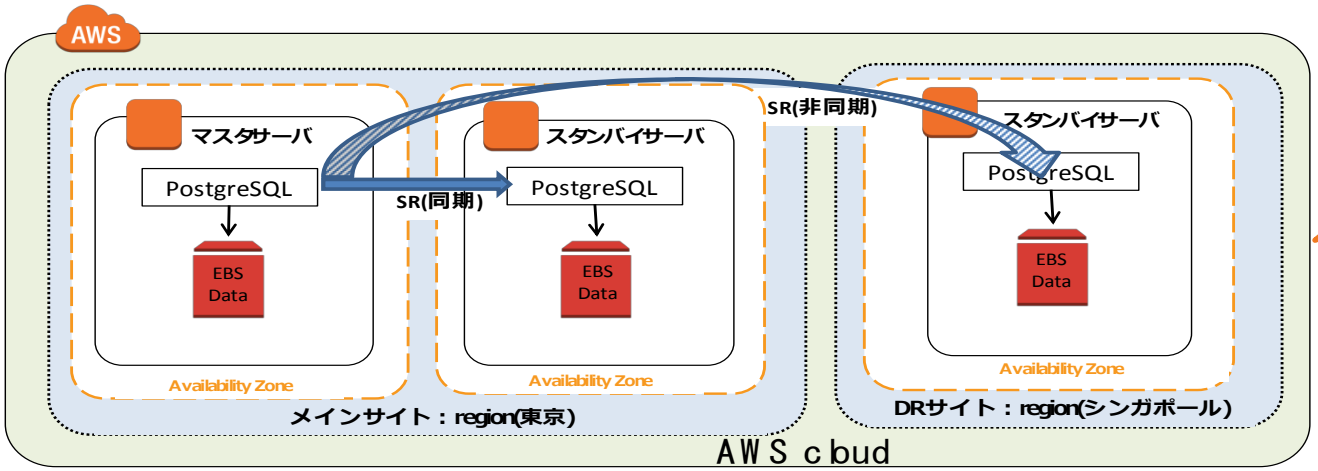
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

運用技術検証
～遠隔地レプリケーションの実態～

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証の目的、観点、環境

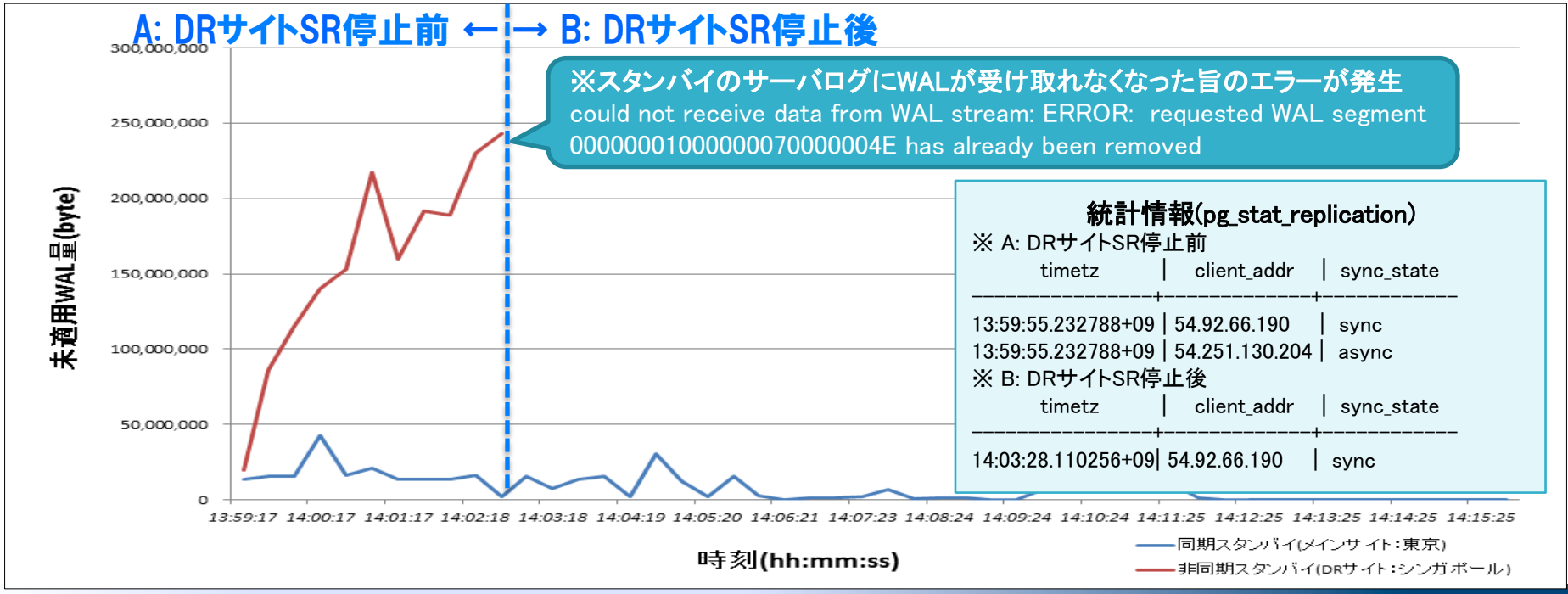
- 目的
  - PostgreSQLのストリーミングレプリケーションでDRを実現する場合に発生する運用上の課題を調査する。
- 観点
  - 遠隔地レプリケーションによるレプリケーションの遅延の実態
  - レプリケーション方式特有のリスクと対策方法
- 環境
  - メインサイト（東京）－DRサイト（シンガポール）間でレプリケーション環境を構築



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証ケース(1)レプリケーション遅延の実態

- **検証方法**
  - 500万件のテーブルを全件更新する更新負荷を掛け続けて計測
  - マスタDBと各スタンバイのレプリケーション遅延量を定期的に取得
- **検証結果**
  - DRサイトのレプリケーション遅延が拡大し、レプリケーションが停止した。



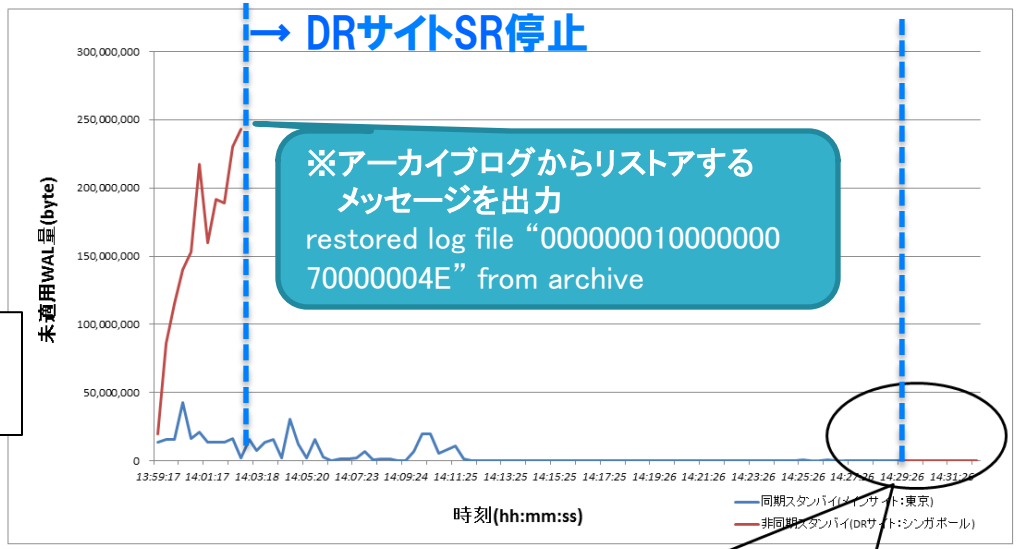
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証ケース(2)レプリケーション停止への対策

■ 検証方法

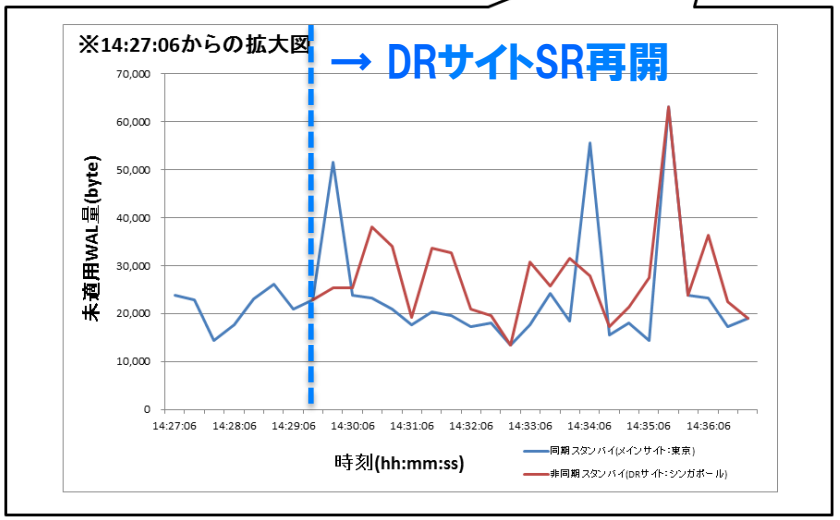
- recovery.confのrestore_commandにマスタDBのアーカイブファイルを直接コピーするコマンドを設定して稼働させる。

```
restore_command =
'scp (マスタのIP):/usr/local/pgsql/arc/%f %p'
```



■ 検証結果

- レプリケーションが停止してから、約30分後にレプリケーションが再開
- 別途アーカイブログを転送しておくことで、さらに再開が早まることを確認

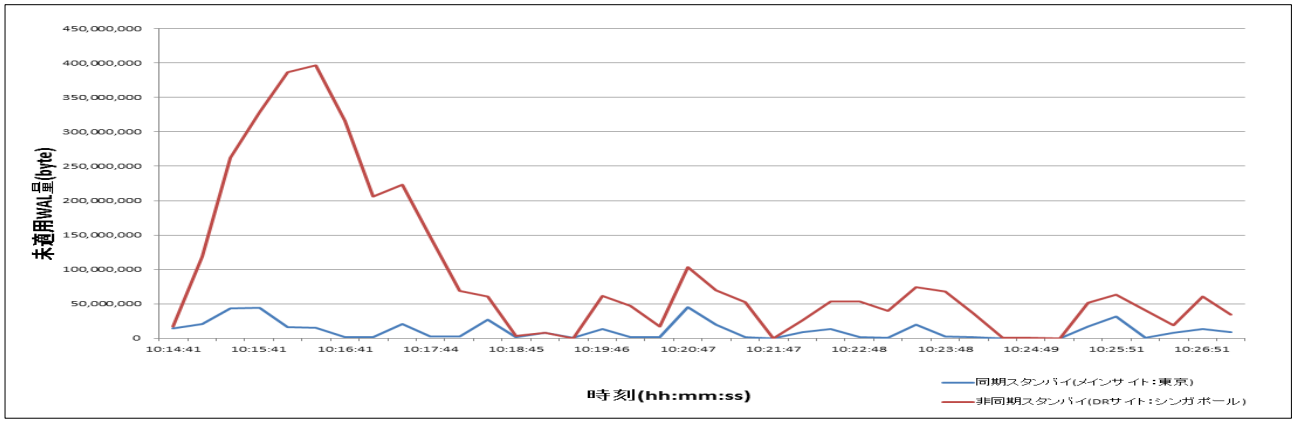


レプリケーション停止のリスクを小さくするには、アーカイブログを随時スタンバイに転送するとよい。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証から得られた知見

- wal_keep_segmentsパラメータの有効性
  - 0→32 に設定すると、レプリケーションは停止しなかった。



wal_keep_segments を大きく設定すれば、
レプリケーション停止のリスクは小さくなる。

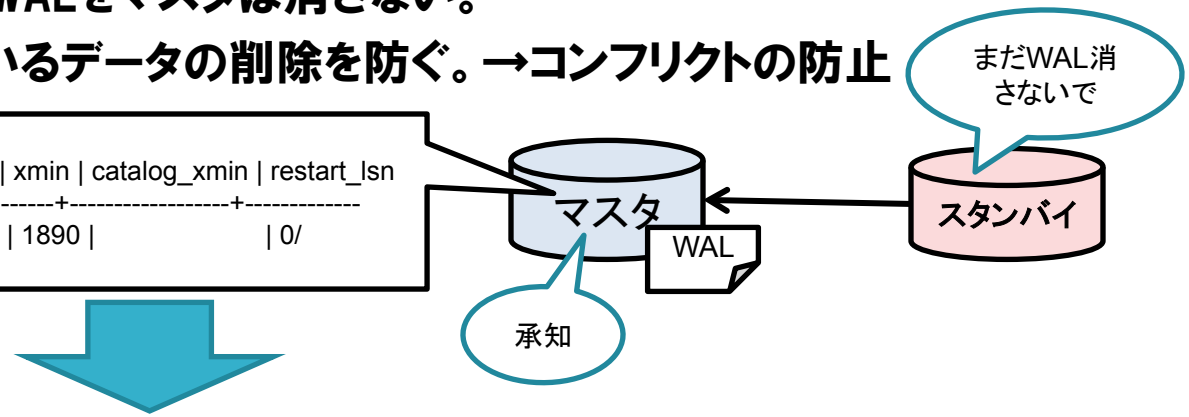
注意する点
DISK容量（マスタサーバのWALファイルが残り続けるため）

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

新機能：レプリケーションスロット

- 9.4から追加された新機能
- マスタはスタンバイのレプリケーション状況を保持する。
  - スタンバイが未適用のWALをマスタは消さない。
  - スタンバイが参照しているデータの削除を防ぐ。→コンフリクトの防止

```
postgres=# SELECT *  FROM pg_replication_slots;
slot_name | plugin | slot_type | datoid | database | active | xmin | catalog_xmin | restart_lsn
-----------+--------+-----------+---------+----------+--------+------+--------------+-------------
    a_slot   |        | physical  |         |          | t      | 1890 |              | 0/
D001218
```



有効な場面

- スタンバイが複数存在する等適切な wal_keep_segments の見積が困難
- wal_keep_segments の値が適切でなかった場合の保険

注意する点

- DISK容量（マスタがWALを保持し続ける可能性があるため）

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

検証から得られた知見

- スタンバイのレプリケーションがマスタからどれ位遅れているかを知るのは意外と難しい。

pg_stat_replication ビューの項目

項目	マニュアルの説明
sent_location	接続上に送信された最後のトランザクションログ位置
write_location	スタンバイによってディスクに書き出された最後のトランザクションログ位置
flush_location	スタンバイによってディスクに吐き出された最後のトランザクションログ位置
replay_location	スタンバイのデータベースに再生された最後のトランザクションログ位置

- 「sent_location ≠ マスタのトランザクションログ位置」に注意！
  - マスタのトランザクションログ位置は別途関数で取得しなければならない。

```
postgres=# select client_addr, pg_xlog_location_diff(master,replay_location) as replaydiff from
                (select pg_current_xlog_insert_location() master)as m,pg_stat_replication;")

   client_addr   | replaydiff
-----------------+-----------
  54.92.66.190   |    1463080
 54.251.130.204  |  315867024
```

各スタンバイのアドレス

未適用WAL量(単位はbyte)

- カスケード構成でマスタに直結しないスタンバイのトランザクションログ位置を知るのはもっと困難

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

可用性テーマのまとめ

- PostgreSQLでDRサイトを構築する構成パターンとそれぞれの特徴を整理した。
- ストリーミングレプリケーションでDRサイトを構築する構成に対して実機検証を行った。
  - DRサイトの構築手順と障害復旧手順を確認
  - 遠隔地間レプリケーションの挙動を確認
- 導入は簡単。運用にはいくつか注意点があった。
  - 構成パターン、障害箇所に合わせたリカバリ手順の整備
  - 遅延拡大によるレプリケーション停止のリスク
- 実運用のノウハウは今年度の検証で得られたものを含めて、今後も整備していく必要がある。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

セキュリティテーマの成果紹介

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

調査の目的・背景

- 背景
  - 過去の調査、検証でPostgreSQLはエンタープライズでも使える機能、性能を有することがわかっており、機能への要求が高度化している。その1つがセキュリティ。
- 課題
  - PostgreSQLが一般的なセキュリティ要件にどこまで対応しているかを検証・周知している資料が少ない。
  - PostgreSQLの機能で解決できないセキュリティ要件への対応方法が不明

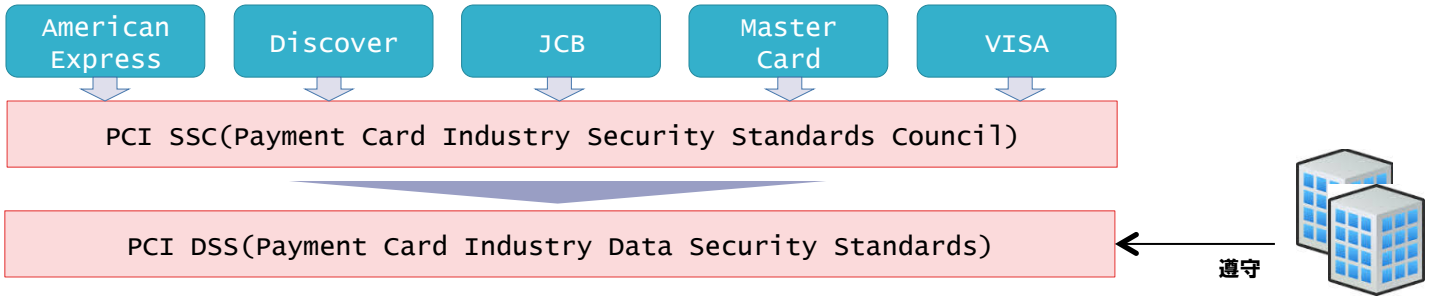
  ⇒強固なセキュリティ要件を求める案件にPostgreSQLを適用できるかがわからない。
- 目的
  - PostgreSQLが強固なセキュリティ要件に対応できるかを検証する。

強固なセキュリティ要件の1つとして「PCI-DSS」がある。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

PCI DSSとは

- PCI DSS (Payment Card Industry Data Security Standards) とは
  - クレジットカード会員データを安全に取り扱う事を目的として策定された、クレジットカード業界のセキュリティ基準
- 経緯
  - 従来、各国際カードブランドが独自でリスク管理プログラムを運用。加盟店が各基準に対応する負荷が高い。
  - ネット決済のグローバル化によりカード情報漏洩のリスク増加。



- 基準の統一化による加盟店の負荷軽減
- セキュリティ基準の遵守による加盟店の企業価値向上

【導入を必要とする想定企業】
イシュアー、アクワイアラー、サービス・プロバイダー、加盟店

- 認定メリット
  1. カード情報漏洩のリスク低減
  2. 基準に準拠による信頼性により企業価値向上
  3. 万が一情報流出事件が発生した場合でもペナルティが免責されるケースがある。

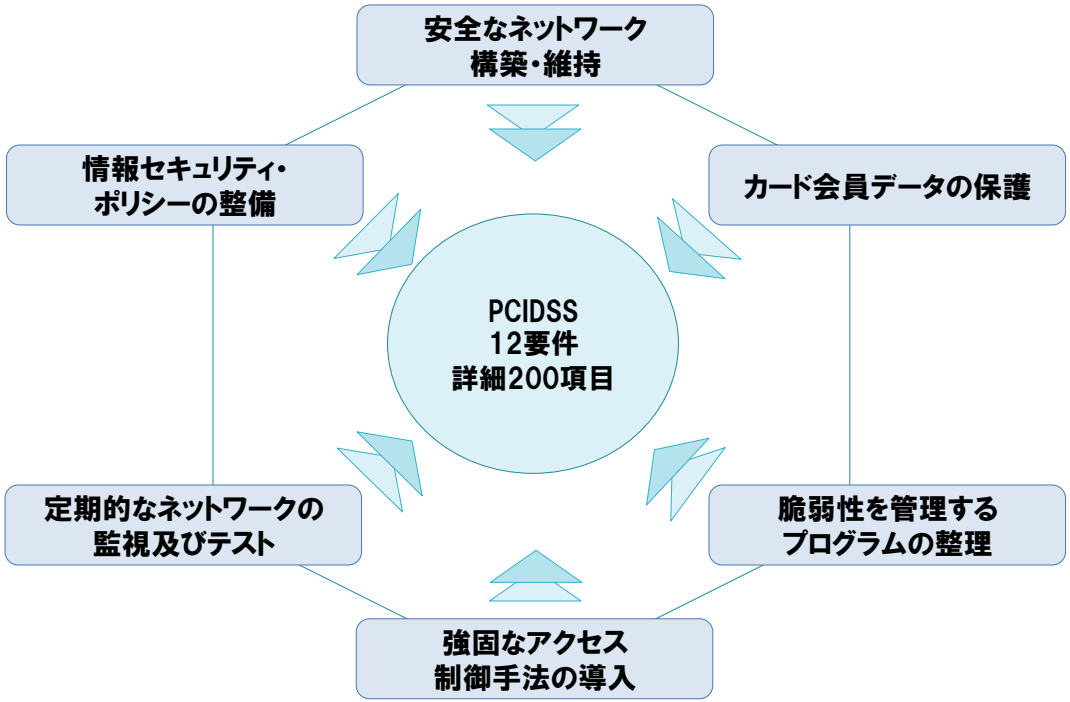
カードビジネスにおける必須条件

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

PCI DSS要件

- PCI DSSでは、安全なネットワークの構築、データの保護、プログラムの脆弱性、アクセス制御手法、ネットワークの監視及びテスト、セキュリティポリシーの整備といった分野が12の要件に分類され、それがさらに詳細な200項目にブレイクダウンされている。

参考：PCI Security Standards Council：https://ja.pcisecuritystandards.org/minisite/en/pci-dss-supporting-docs-v20.php



これらの項目に対応する必要がある。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

調査方法

手順

PCI DSS 調査

- PCI DSS要件でデータベースが対応する必要がある項目を抽出
  ※DataBase Security Consortiumの成果物を引用

PostgreSQL 対応検討

- 抽出した項目に対して、PostgreSQLでの対応可能か確認
- 検証が必要となる箇所の抽出

検証作業

- 検証作業が必要となる箇所について検証作業を実施

- DataBase Security Consortium (DBSC) が作成したPCI DSSのセキュリティガイドラインについて
  - DBSCが過去にPCI DSSデータベースセキュリティガイドラインを作成して公開している。
  - このガイドラインをPostgreSQLに適用することとした。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

PostgreSQLの適用技法

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

適用技法紹介

- PCI DSSでDBが対応すべき分野

アクセス制御

- アカウントポリシー機能の実現
- 不正アクセスのチェック
- 定期的なセッション情報の分析
- 不正アクセスの動的遮断
- アクセス時間外の接続検知
- 長時間アイドル中の接続を自動的に遮断

暗号化

- 格納データの暗号化
- ファイルシステムの透過的暗号化

ログ監査

- 出力したログの保全

次ページ以降で具体的適用技法を紹介

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

アクセス制御

暗号化

ログ監査

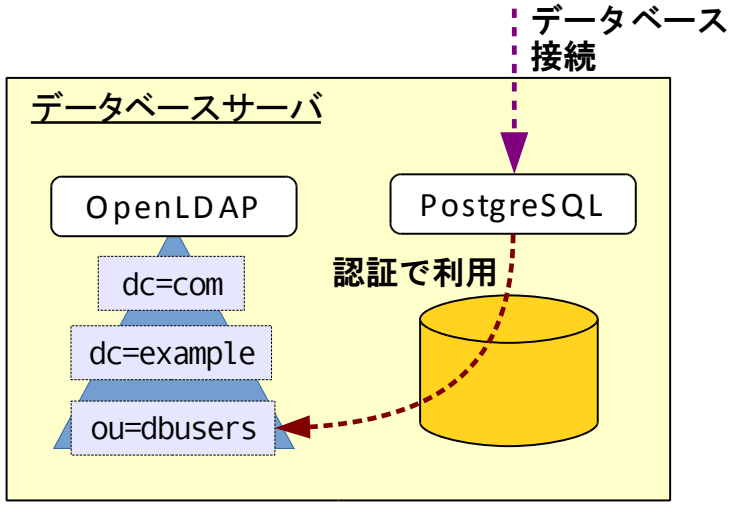
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

アカウントポリシー機能の実現

- アカウントポリシーとは？
  - データベースユーザアカウントに付加する様々な条件
  - 以下を強制する仕組みが要求されている

「最初の接続時にパスワード変更、また、少なくとも90日ごとパスワード変更」
「パスワードは数字、英文字の両方を含む7文字以上で、前回、前々回と同じものはダメ」
「パスワード認証を6回失敗したら、少なくとも30分はロックアウト」 など。

- PostgreSQLでは？
  - CREATE ROLE のオプションでは不足
  - 外部認証機能を使うことで対応
  - LDAP, GSSAPI, SSPI, RADIUS, PAM が利用可能
  - OpenLDAP + LDAP外部認証 を使った実現例（右図）の構築手順を解説



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

不正アクセスのチェック

- 不正なアクセスを試みられていないか
  - 例） 短時間に一定回数以上のパスワード認証の失敗は、パスワード攻撃の可能性を疑う。

- PostgreSQLでは？
  - 接続失敗の証跡はサーバログに出力できる。
  - ただし検索にはgrepでは限界があるためコーディングが必要（でもコーディングは面倒..）

- 対処

サーバログをPostgreSQLのテーブルにロードして
SELECT文で自在に検索！

  - CSVLOG形式のログをCOPY FROMコマンドでロード
  - 条件や集計を自在に指定

検索例）パスワード認証エラーが1分間に100回発生していなか?
postgres_logテーブルについては「2014年度 WG3活動報告 セキュリティ編」参照

```
SELECT to_char(log_time,'YYYY-MM-DD HH24:MI') AS log_time,
       count(*) AS cnt
FROM   postgres_log
WHERE  message LIKE '%password authentication failed%'
GROUP  BY to_char(log_time,'YYYY-MM-DD HH24:MI')
HAVING count(*) >= 100;
```

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

定期的なセッション情報の分析

- セッションを定期的に分析して不正な動きがないか確認
- 例1）ログイン失敗回数が多い接続試行
  - 単なるパスワードミス？　不正アクセスの試行？
  - エラー原因や接続元ホストや接続先ユーザ、データベース　などの傾向を確認

検索例）認証エラーの種類別の集計

```
SELECT message,
       count(*) AS count
FROM   postgres_log
WHERE  sql_state_code IN ('28000','28P01')
GROUP  BY message;
```

主な認証エラー

メッセージ	エラー原因	SQL STATE
FATAL: no pg_hba.conf entry for host "[local]", user "xxxxx", database "xxxxx", SSL off	pg_hba.confに該当設定が無い	28000
FATAL: pg_hba.conf rejects connection for host "[local]", user "xxxxx", database "xxxxx", SSL off	接続拒否	28000
FATAL: Peer authentication failed for user "xxxxx"	Peer認証エラー	28000
FATAL: password authentication failed for user "xxxxx"	パスワードミス　または パスワードの有効期限切れ	28P01

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

定期的なセッション情報の分析

- 例2） 長時間に渡りログインしているセッション
  - なぜ長時間？　なにか不正してない？

検索例） 30分以上接続していたセッション

```
SELECT log_time,
       user_name,
       database_name,
        ～中略～
FROM   postgres_log
WHERE  message LIKE 'disconnection:%'
AND    substr(message,30,7) > '0:30:00';
```

  - 接続元ホスト名、接続先ユーザ名、接続先データベース名、アプリケーション名などから不審なセッションかどうかを確認
  - 不審な点がある場合は、別途上記セッションの詳細を検索

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

不正アクセスの動的遮断

- PostgreSQLには、自動で不正アクセスを動的切断する機能はなく、手動で行う、個別に工夫したスクリプトが必要

- 実現手段
  - 管理者がサーバログを見ることで不正アクセスをしているユーザを調べる。
  - pg_stat_activityビューで、ユーザのプロセスIDを特定する。
  - pg_terminate_backend () 関数を利用

- 適用例
  - 不正アクセスしているユーザが「user1」と分かっている場合に手動で動的遮断をする。

```
SELECT pg_terminate_backend(pid),
FROM   pg_stat_activity
WHERE  username='user1';
```

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

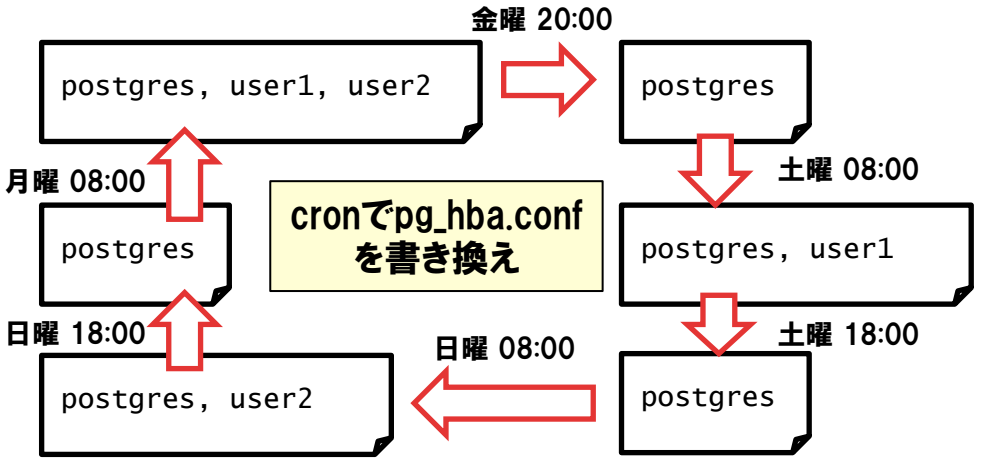
アクセス時間外の接続検知

- PostgreSQLの標準機能には、時間帯や曜日でアクセス制限をかけることができないので、作りこみが必要

- 実現手段
  - pg_hba.confの元となるファイルを時間帯・曜日別に複数用意する。
  - cronで所定のpg_hba.confに上書きし、pg_ctrl reloadで反映
  - 時間外の場合はpg_hba.confによってクライアント認証が失敗するので、クライアント認証エラーメッセージから、どのユーザによるものか検知する。

- 適用例：アクセス時間を下記表のように定義した場合

ユーザ名	曜日・時間帯
user1	月曜08:00～金曜20:00、 土曜08:00～18:00
user2	月曜08:00～金曜20:00、 日曜08:00～18:00
postgres （管理者）	常時アクセス可能



Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

長時間アイドル中の接続を自動的に遮断する

- OSのネットワーク機能や、ネットワーク機器の機能では？
  - 「長時間の応答待ち」と「アイドル中（＝何もしてない）」の区別ができない。

- PostgreSQLでは？
  - アイドル状態を自動的に遮断（＝接続されているのを切断）する機能は無い。
  - 指定した接続を終了させる機能はある　・・・　pg_terminate_backend() 関数
  - 「接続はアイドル状態か？」「現在の状態になってからの経過時間」を知る手段はある。
    ・・・　pg_stat_ativity システムビュー

- 定期SQL実行で代替実現
  - 以下を cron で定期実行

```
SELECT pg_terminate_backend(pid),
  pid, datname, usename, application_name, client_addr,
  client_hostname, client_port, query,
  backend_start, xact_start, query_start, state_change
FROM pg_stat_activity
  WHERE pid <> pg_backend_pid() AND state LIKE 'idle%'
  AND state_change < current_timestamp - '15 min'::interval;
```

遮断した接続の情報を取得

遮断する条件

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

アクセス制御

暗号化

ログ監査

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

格納データの暗号化

- やりたいこと
  - データ暗号化機能/ツールを利用し、格納データを暗号化する。

- PostgreSQLでは？
  - contribモジュールの「pgcrypto」を利用して、暗号化/復号が可能

- pgcryptoでできること/できないこと
  - できること
    - ・共有鍵方式による暗号化/復号
    - ・公開鍵方式による暗号化/復号
  - できないこと
    - ・鍵の管理
      →鍵をどのように管理して運用するかは十分に検討しておくこと。
    - ・透過的な暗号化/復号
      →次ページの「ファイルシステムの透過的暗号化」などを参考に。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

ファイルシステムの透過的暗号化

- やりたいこと
  - データ暗号化機能/ツールを利用し、格納データを暗号化する。
  - また、可能な限りSQLの書き換えは行いたくない。

- PostgreSQLでは？
  - PostgreSQL本体の機能だけでは、透過的暗号化を行うことはできない。
  - OSの機能や、商用製品を利用する必要がある。

- OSによる代替案
  - OSに付属するdm-crypt暗号化機能で、ファイルシステムレベルで暗号化/復号する。
    →ディスクの抜き取りや仮想イメージデータの奪取などのデータ盗み出しに対するデータ保護が可能になる。
  - 注意点
    →データへのアクセスについては、通常のファイルシステムを利用する場合と違いはない。ただし、マシンの再起動時にパスフレーズを入力するなどの適切な手順を踏む必要がでてくることに注意が必要である。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

アクセス制御

暗号化

ログ監査

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

出力したログの保全

- やりたいこと
  - 監査証跡ファイルを不正な変更（改ざん）から保護する。
- PostgreSQLでは？
  - リアルタイムに監査証跡ファイル（サーバログ）の改ざんを防止する仕組みはない。
  - 改ざん防止のためには、別途OSやツールと組み合わせる必要がある。
- 代替手段

  防止はできないまでも、下記の点を考慮した運用を検討する
  - 改ざんされにくくする

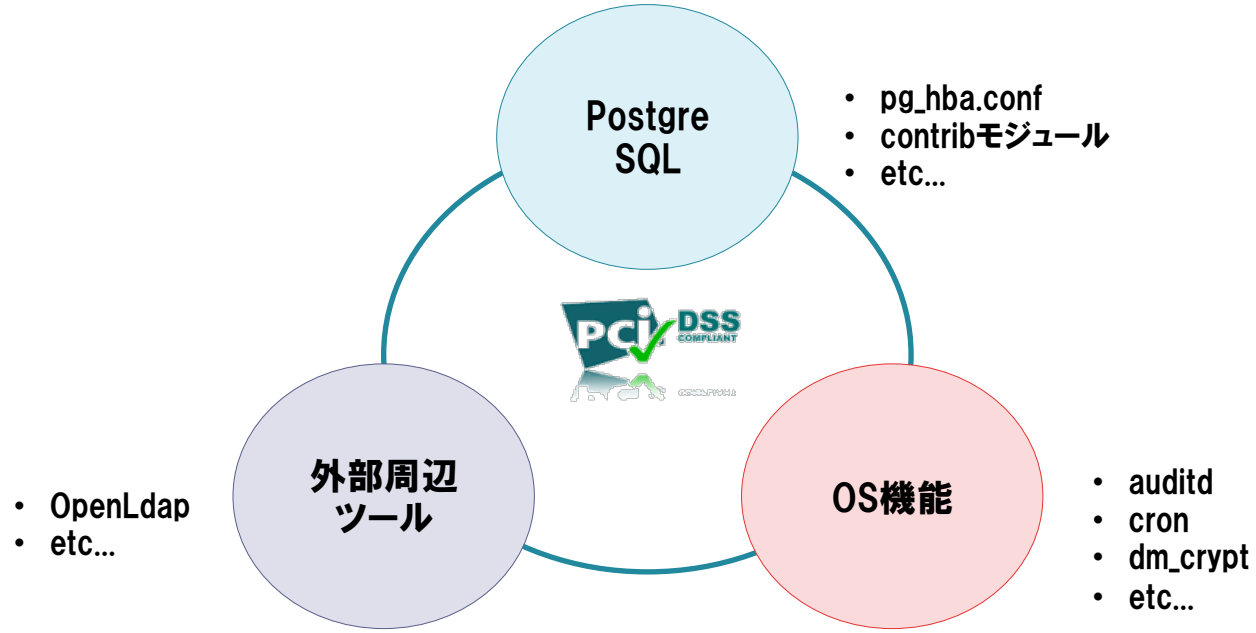
    →サーバログへのアクセス権限を、log_file_modeパラメータで適切に設定する。
  - 改ざんを検出する

    →万が一、改ざんが発生してもすぐに検知できる仕組みを用意する。例えば、OS付属のauditdでサーバログへのアクセスを監視するなどがある。
  - 改ざんされていないことを保証する

    →OpenSSLなどでサーバログに対する電子署名を生成し、サーバログの信ぴょう性を担保する。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

セキュリティテーマまとめ



PCI DSS要件をクリアするためには、PostgreSQL単体の機能のみではなく外部周辺ツールやOS機能を組み合わせて対応することがポイントになる。

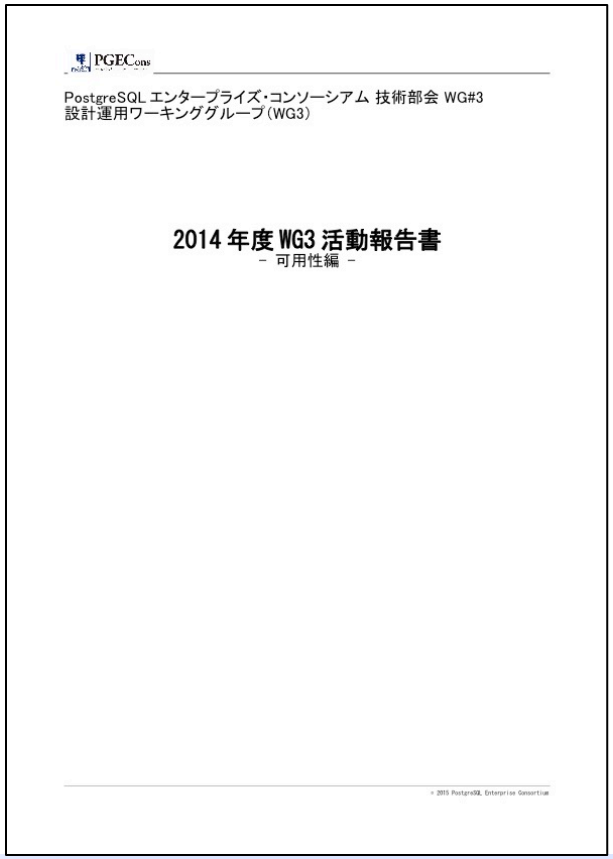
Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

エンディング

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

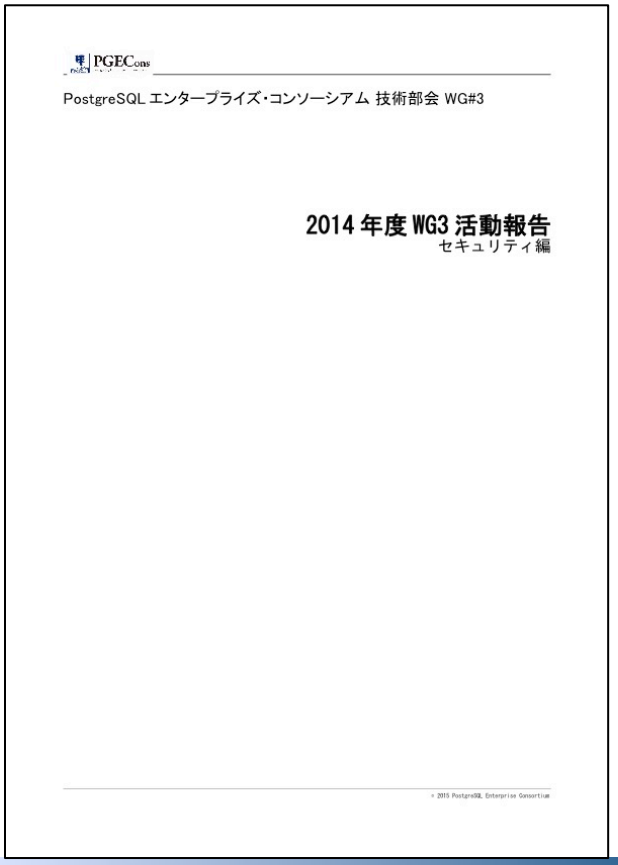
2014年度の活動成果

- 「2014年度WG3活動報告書」として、PGEConsのサイトにドキュメントを公開しています。
  - 可用性編：63ページ、セキュリティ編：83ページ


PGECons

PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#3
設計運用ワーキンググループ(WG3)

2014 年度 WG3 活動報告書
－ 可用性編 －


PGECons

PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#3

2014 年度 WG3 活動報告
セキュリティ編

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

2014年度活動を振り返って

- 参加企業の皆様からいただいたコメント
  - 1つの企業のみでは取り組みづらい技術的な課題に対しPGEConsを通して効率的、効果的に調査ができました。
  - 普段は競合他社である方々から様々な視点で議論したりレビューを受けたりするのは新鮮で有意義でした。
  - 普段とは異なるメンバと議論することで、技術面、ビジネス面ともに通常の業務では得られない知識が得られたと思います。

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.

PGECons
PostgreSQL Enterprise Consortium

PGEConsで一緒に活動しませんか？

Copyright © PostgreSQL Enterprise Consortium, All Rights Reserved.