PGECons
PostgreSQL Enterprise Consortium

サービスの安定継続を支えるPostgreSQLの構成とは

2013年度活動成果中間報告

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

アジェンダ

- WG3について
  - 位置づけ、活動概要
  - PGEConsへ寄せられたコメント
  - エンタープライズで求められる非機能要件とは
  - DBMSに求められる要件とは
  - 2013年度の活動テーマ
  - 実施体制
  - 予定成果物
- 現在までの活動内容紹介
  - 可用性
  - バックアップ
  - 監視
  - 実機運用例
- まとめと今後の予定

WG3の位置づけ、活動概要

2013年度 新設

- 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	監視運用、バックアップ運用
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

PGEConsに寄せられたコメント

- 運用面の注意なども知りたい。
- PostgreSQLの普及の課題は、移行ではなく、ユーザーを納得させられる信頼性と可用性の情報だと思う。そのあたりでOracleとの比較や代替手段等も今後期待します。
- HA構成の案について、さまざまな視点からの評価をいただければと思いました。
- 障害時の動作検証の結果が欲しかった。
- 移行というテーマも興味があるが、それ以前にPostgreSQLを選定して安心だと言えるようなテーマを検証してほしかった。
- 既にpostgreSQLを導入しているため、設計運用WG(WG3)の今後の活動に期待している。
- WG3においてストリーミングレプリケーション機能をpacemaker+Redhatの連携が可能か課題確認いただけると助かります。

※2012年度成果報告会アンケートより抜粋

エンタープライズで求められる非機能要求(要件)

- さまざまな要件が求められる

大項目	概要	要求例
可用性	システムサービスを継続的に利用可能とするための要求	・運用スケジュール(稼働時間・停止予定など) ・障害、災害時における稼働目標
性能・拡張性	システムの性能および将来のシステム拡張に対する要求	・業務量および今後の増加見積 ・システム化対象業務の特性(ピーク時、通常時、縮退時)
運用・保守性	システム運用と保守サービスに関する要求	・運用中に求められるシステム稼働レベル ・問題発生時の対応レベル
移行性	現行システム資産の移行に関する要求	・新システムへの移行期間および移行方法 ・移行対象資産の種類および移行量
セキュリティ	情報システムの安全性の確保に関する要求	・利用制限 ・不正アクセスの防止
システム環境・エコロジー	システムの設置環境やエコロジーに関する要求	・耐震/免震、重量/空間、温度/湿度、騒音などシステム環境に関する事項 ・CO_2排出量や消費エネルギーなどエコロジーに関する事項

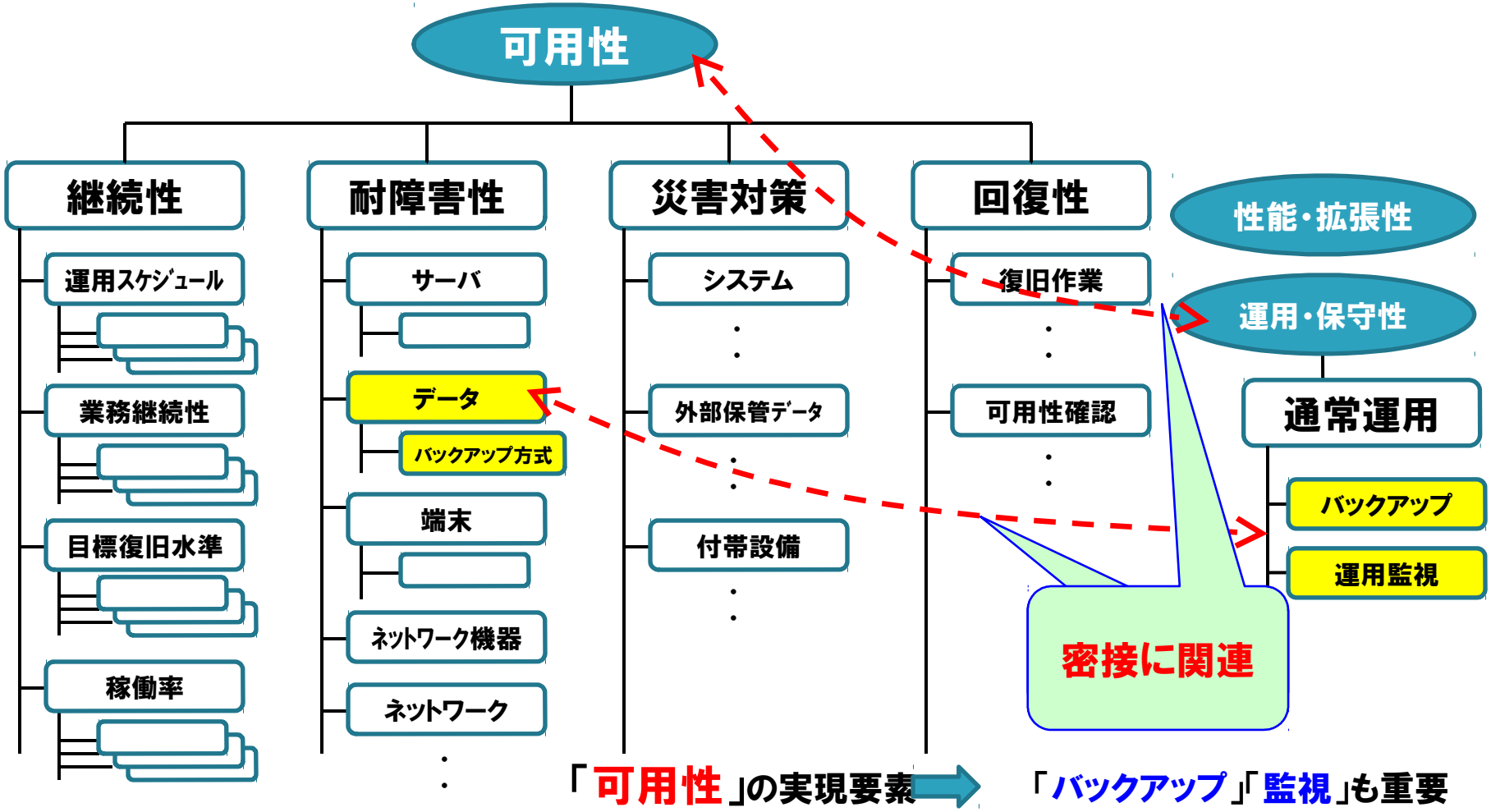
※独立行政法人　情報処理推進機構「非機能要求グレード」より

サービスを安定継続するには？

可用性に着目

エンタープライズで求められる非機能要求(要件)

- 各要求は仔細な事項にわたる



DBMSに求められる要求（要件）

活動テーマ：★マークの項目

★可用性	・データベースが停止しないこと ・障害時の復旧時間が短いこと ・データの破壊、トランザクションの消失がないこと ・災害に対する耐性（ディザスタリカバリ）
性能	以下項目の期待値を満たすこと ・応答時間 ・スループット ・クライアントの同時接続数 ・同時実行トランザクション数
拡張性	・スケールアップ ・スケールアウト
★運用性	★・監視（統合運用監視） ★・バックアップの取得 ・運用自動化のためのジョブ管理 ・各種メンテナンス処理（索引再構成、フラグメンテーション解消など）
接続性	・アプリケーションからの接続 ・他システムとのデータ連携
セキュリティ	・情報が漏洩しないこと ・情報が改ざんされないこと ・権限がない人によるデータ利用がないこと

➡ 活動テーマは「可用性」と「バックアップ」「監視」

2013年度の活動テーマ

- 可用性
  - PostgreSQLの代表的なシステム構成（シングル・HA・レプリケーション）の概要・特徴・考慮事項を調査し、適用領域を整理
  - 上記調査から導出された考慮事項に対する実機検証
- バックアップ
  - 各システム構成ごとにバックアップ手法を洗い出し、バックアップ要件の対応度合いを整理
  - 実機検証による各システム構成ごとの運用例
- 監視
  - DBサーバとDBの死活・性能監視に必要な情報の洗い出し
  - 収集した情報からの分析および対処方針を各システム構成ごとに整理
  - 実機検証による監視ケーススタディと対処法の効果

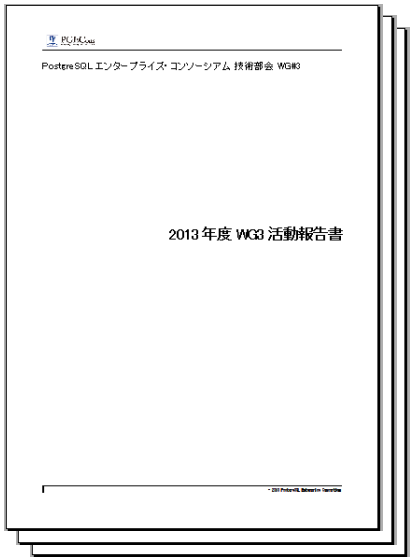
実施体制(14社)

- 株式会社アイ・アイ・エム
- 株式会社アシスト（主査）
- 株式会社インフォメーションクリエーティブ
- SRA OSS, Inc. 日本支社
- NTTソフトウェア株式会社（副査）
- クオリカ株式会社
- TIS株式会社（副査）
- 日本電気株式会社
- 日本電信電話株式会社（副査）
- 日本ヒューレット・パッカード株式会社
- 株式会社日立製作所
- 株式会社日立ソリューションズ
- 富士通株式会社
- フューチャーアーキテクト株式会社

※企業名順

予定成果物

- PostgreSQLの代表的なシステム構成（シングル/可用性構成）
  - 各構成の概要、特徴、考慮事項
  - バックアップ手法の概要、特徴、考慮事項
  - 死活・性能監視項目および収集済み情報の分析・対処法
  - 実機運用例
    - 運用例（2～3ケースを想定）を前提とした可用性、バックアップ、監視に関する実機検証結果



現在までの活動内容紹介
- 可用性 -

一般的なデータベースシステム構成

シングルサーバ構成

1台で構成

HAクラスタ(Active-Standby構成)

Activeサーバでの障害発生時、障害を検知し、Standbyサーバにフェイルオーバして運用を継続

- 共有ストレージ方式
- ストレージレプリケーション方式(ミラーリング)

データベースレプリケーション

データを複数のサーバにデータの複製(レプリカ)を作ることでデータの冗長性を確保

- ログシッピングレプリケーション
  - 同期
  - 非同期
- その他のレプリケーション

マルチマスタ負荷分散クラスタ

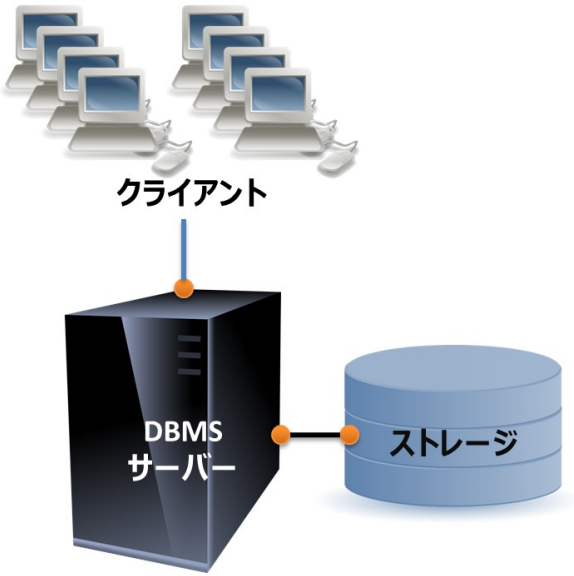
複数のサーバにまたがってデータを保持し、各サーバで並行に処理することで負荷分散を実現

- シェアードエブリシング
- シェアードナッシング

PostgreSQLはどこまで対応可能？

シングルサーバ構成(基本構成)

- 特徴
  - 1台で構成されるPostgreSQLサーバ
  - 基本構成のため運用もシンプル
  - PostgreSQL単体で運転継続性を高めるチューニングが必須
- データ同期性
  - なし
- 耐障害性
  - 低
    - バックアップからのリストア・リカバリが必要
- 拡張性
  - なし
- コスト
  - 低



PostgreSQLで構築可能な構成

シングルサーバ構成

HAクラスタ(Active-Standby構成)

Activeサーバでの障害発生時、障害を検知し、Standbyサーバにフェイルオーバして運用を継続

- 共有ストレージ方式
- ストレージレプリケーション方式(ミラーリング)

データベースレプリケーション

データを複数のサーバにデータの複製(レプリカ)を作ることでデータの冗長性を確保

- ログシッピングレプリケーション
  - 同期
  - 非同期
- その他のレプリケーション

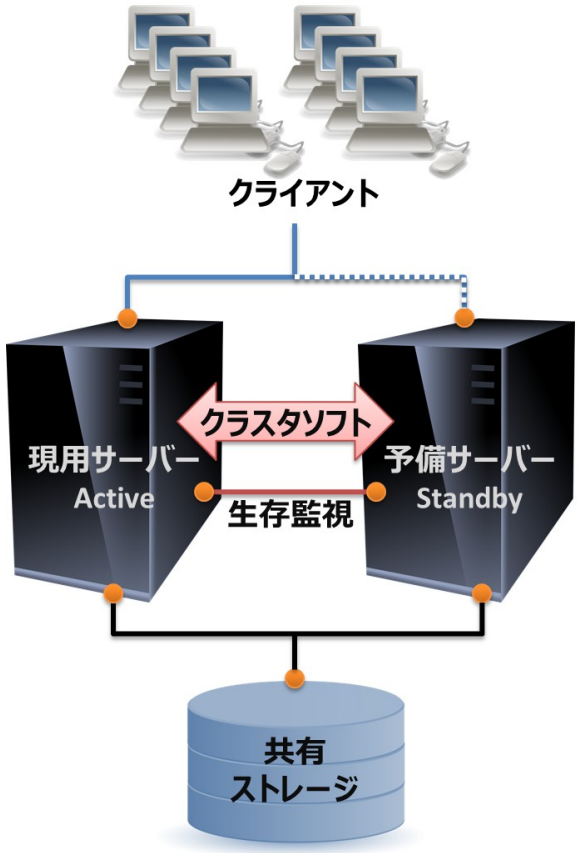
マルチマスタ負荷分散クラスタ

複数のサーバにまたがってデータを保持し、各サーバで並行に処理することで負荷分散を実現

- シェアードエブリシング
- シェアードナッシング

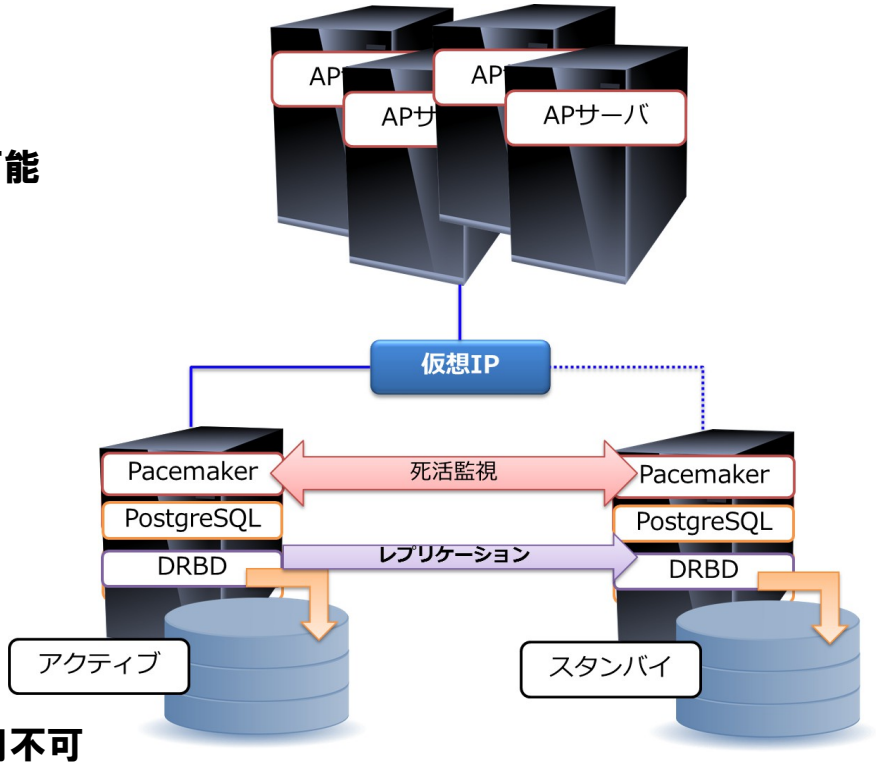
HAクラスタ構成（共有ストレージ方式）

- 特徴
  - 2台のPostgreSQLサーバによるActive-Standby構成
  - 通常時はActiveサーバで処理を行い、Standbyサーバはトラブル発生に備え待機
  - データを両系で共有するシンプルで実績のある方式
- データ同期性
  - あり
    - 共有のため同期
- 耐障害性
  - 中
    - 共有ディスクがSPOFになりうる
- 拡張性
  - なし
- コスト
  - 高
    - 待機系はサービス停止中のため、リソース活用不可
    - 共有ディスクが高価



HAクラスタ構成（ストレージレプリケーション方式）

- 特徴
  - DRBDを利用したファイルシステムレベルのブロックレプリケーションで、データを複製
  - ウォームスタンバイ状態で待機
- データ同期性
  - あり
    - ブロック単位の非同期/同期モードが選択可能
- 耐障害性
  - 中
    - サーバ多重障害に弱い
- 拡張性
  - なし
- コスト
  - 低
    - 待機系はサービス停止中のためリソース活用不可
    - シンプルで低コスト



PostgreSQLで構築可能な構成

シングルサーバ構成

HAクラスタ(Active-Standby構成)

Activeサーバでの障害発生時、障害を検知し、Standbyサーバにフェイルオーバすることで運用を継続

- 共有ストレージ方式
- ストレージレプリケーション方式(ミラーリング)

データベースレプリケーション

複数のサーバにデータ複製(レプリカ)を作ることで冗長性を確保

- ログシッピングレプリケーション
  - 同期
  - 非同期
- その他のレプリケーション

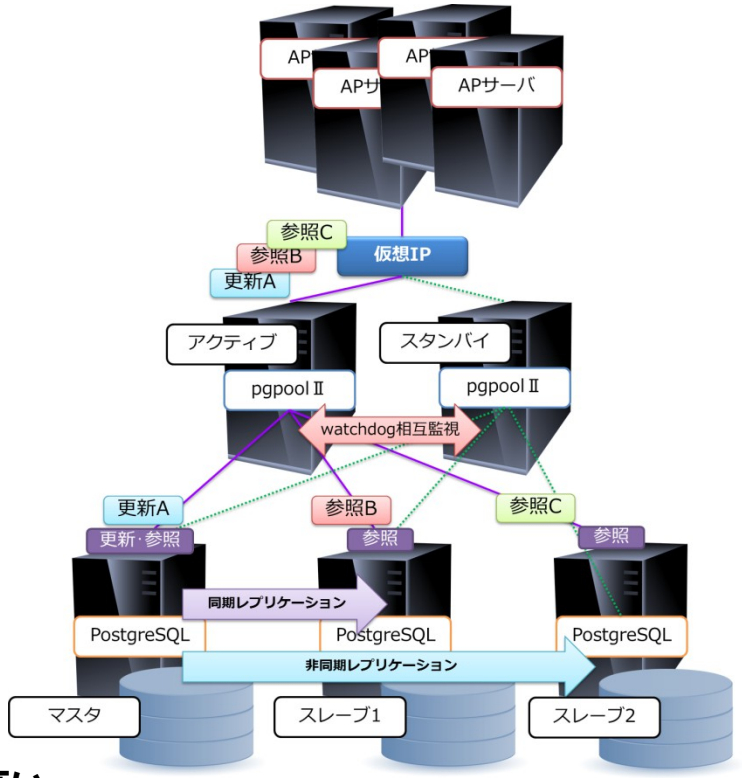
マルチマスタ負荷分散クラスタ

複数のサーバにまたがってデータを保持し、各サーバで並行に処理することで負荷分散を実現

- シェアードエブリシング
- シェアードナッシング

データベースレプリケーション（ログシッピング）

- 特徴
  - PostgreSQL標準のストリーミング・レプリケーションによりデータを複製
  - pgpool-II経由により、APからは1台のDBサーバとしてアクセス可能
- データ同期性
  - あり
    - WAL操作単位の非同期/同期モードが選択可能
- 耐障害性
  - 高
    - 複数台のスレーブサーバ構成が可能
- 拡張性
  - あり
    - 参照系処理の負荷分散が可能
- コスト
  - 低
    - 待機系も参照可能なため、リソースの利用効率は高い
    - 共有ディスク等の高価なH/Wが不要

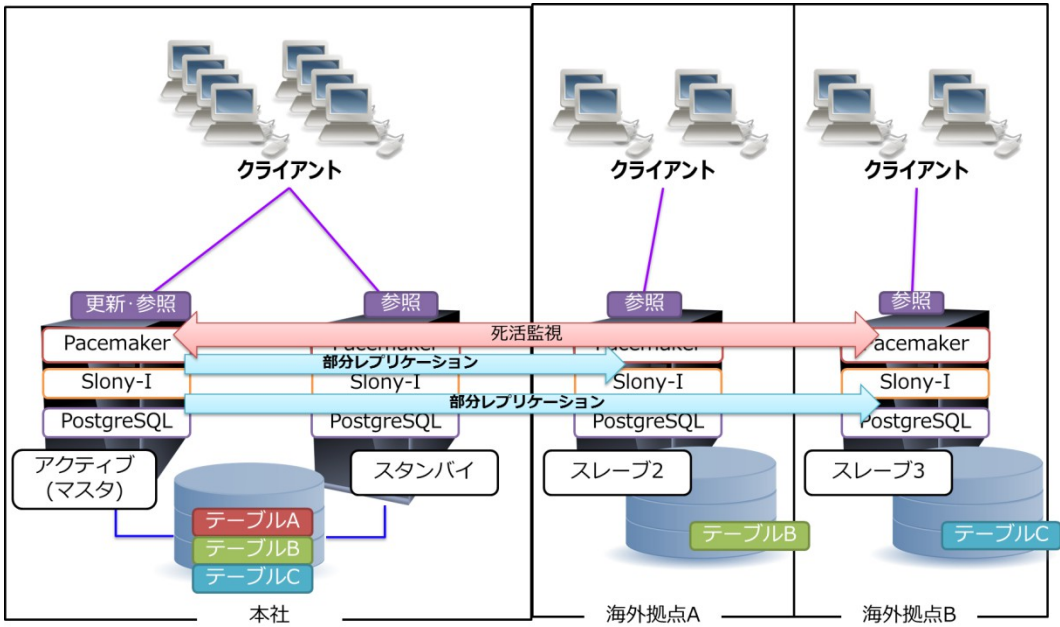


（参考）切替の自動化（pgpool-II）

- pgpool-IIはPostgreSQL専用のミドルウェア
- 実装機能
  - コネクションプール
  - レプリケーションモード
  - ストリーミング・レプリケーションモード
  - 負荷分散
  - 接続数制限
  - パラレルクエリ
- ストリーミング・レプリケーションモード
  - PostgreSQLストリーミング・レプリケーション環境でクラスタウェアとして機能

その他のレプリケーション（Slony-I）

- 特徴
  - テーブル単位のデータ複製
  - スレーブは参照処理が可能
- データ同期性
  - あり
    - トリガーベースで非同期
- 耐障害性
  - 中
    - スレーブサーバ障害のマスタへの影響なし
- 拡張性
  - あり
    - 参照系負荷分散が可能
- コスト
  - 低
    - 待機系も参照可能なため、リソースの利用効率は高い
    - 共有ディスク等の高価なH/Wが不要



PostgreSQLで構築可能な構成

シングルサーバ構成

1台で構成

HAクラスタ(Active-Standby構成)

Activeサーバでの障害発生時、障害を検知し、Standbyサーバにフェイルオーバすることで運用を継続

- 共有ストレージ方式
- ストレージレプリケーション方式(ミラーリング)

データベースレプリケーション

複数のサーバにデータ複製(レプリカ)を作ることで冗長性を確保

- ログシッピングレプリケーション
  - 同期
  - 非同期
- その他のレプリケーション

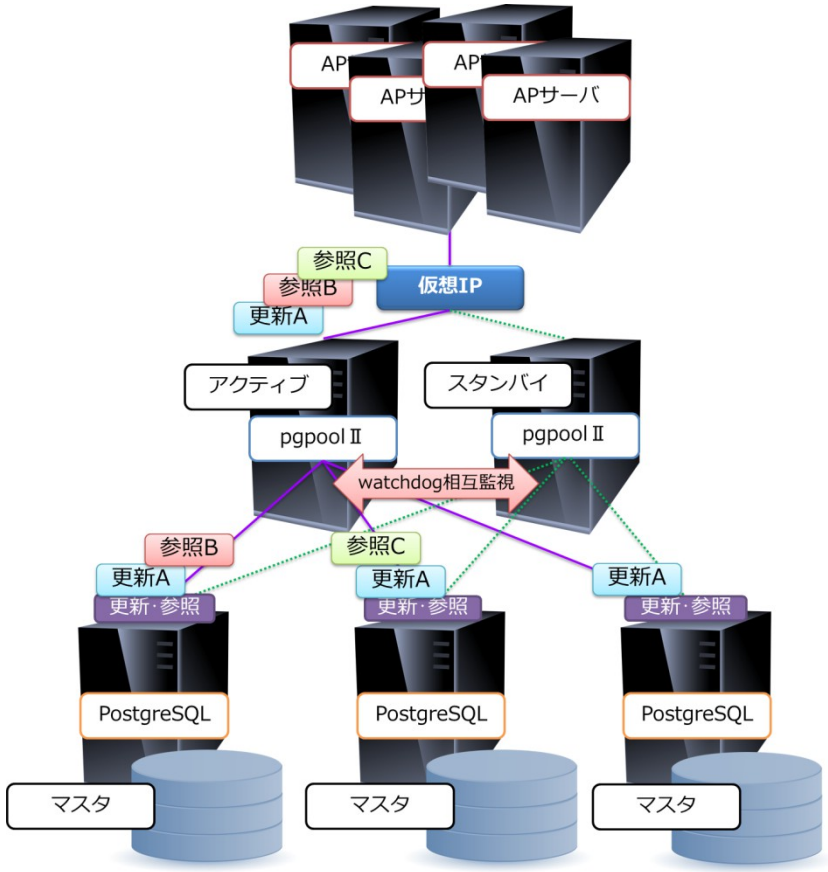
マルチマスタ負荷分散クラスタ

複数のサーバにまたがってデータを保持し、各サーバで並行に処理することで負荷分散を実現

- シェアードエブリシング
- シェアードナッシング

マルチマスタ負荷分散クラスタ(シェアードナッシング/pgpool-II)

- 特徴
  - pgpool-II配下の各PostgreSQLに対して、同一の参照・更新処理を送ることでデータを複製
- データ同期性
  - あり
    - SQL単位で同期
- 耐障害性
  - 高
    - 複数サーバ構成が可能
- 拡張性
  - あり
    - サーバの動的追加が可能
- コスト
  - 低
    - 全サーバに対して参照・更新処理が可能なため、リソース効率は高い



PostgreSQLで構築可能な構成

シングルサーバ構成

1台で構成

HAクラスタ（Active-Standby構成）

Activeサーバでの障害発生時、障害を検知し、Standbyサーバにフェイルオーバすることで運用を継続

- 共有ストレージ方式
- ストレージレプリケーション方式（ミラーリング）

データベースレプリケーション

複数のサーバにデータ複製（レプリカ）を作ることで冗長性を確保

- ログシッピングレプリケーション
  - 同期
  - 非同期
- その他のレプリケーション

マルチマスタ負荷分散クラスタ

複数のサーバにまたがってデータを保持し、各サーバで並行に処理することで負荷分散を実現

- シェアードエブリシング
- シェアードナッシング

可用性要件に応じた様々な構成が可能

PostgreSQL構成一覧

	シングル サーバ	HAクラスタ		データベースレプリケーション		マルチマスタ 負荷分散 クラスタ
		共有 ストレージ	ストレージ レプリケーション	ログ シッピング	Slony-I	Pgpool-II
データ 同期性	なし	あり	あり	あり	あり	あり
耐障害性	低	中	中	高	中	高
拡張性	なし	なし	なし	あり	あり	あり
コスト	低	高	低	低	低	低

実機検証を行い、構築・運用における考慮点を整理予定

24

現在までの活動内容紹介
– バックアップ –

バックアップ方式

- オフラインバックアップ／リカバリ
- 論理バックアップ／リストア
- オンラインバックアップ／リカバリ
- ストレージローカルコピー／リストア

バックアップ方式の比較(イメージ)

	オフラインバックアップ	オンラインバックアップ(論理バックアップ)	オンラインバックアップ(物理バックアップ)		レプリケーション	
			OSレベルのファイルコピー	ストレージローカルコピー	同期	非同期
No	1	2	3-1	3-2	4-1	4-2
バックアップ概要	cp rsync	pg_dump/pg_dumpall	pg_basebackup等	各種	Streaming Replication	
業務への影響度						
必要構成						
バックアップ最小単位						
バックアップ取得時間						
リストア・リカバリ時間(RTOに関連)						
障害時に過去のどの時点にデータを戻せるか(RPOに関連)						
オペレーションミスによるデータ障害への対応						
リストア・リカバリに必要なデータ						
注意・課題事項						

実際に実機検証した結果も加味し、
各方式を整理した形で一覧表に

現在までの活動内容紹介
– 監視 –

シングルサーバ構成における監視

- 死活監視・性能監視項目
  - 死活監視
    - DBサーバ
      - サーバ応答
      - ネットワーク など
    - PostgreSQL
      - PostgreSQLプロセス
      - SQL実行可否 など
    - 間隔：数秒
  - 性能監視
    - DBサーバ
      - パケット転送量
      - IO待ち要求数 など
    - PostgreSQL
      - ロック状態
      - キャッシュヒット率 など
    - 間隔：数分

シングルサーバ構成における監視

- 監視項目一覧のイメージ

対象	いつ	どこで	何を	どのように
DBサーバ	(短期間) 1分程度	対象サーバ	サーバの応答 (OSの死活)	ping
			ネットワーク (NW/NICの死活)	ping
			サーバログ (サーバ上の問題有無)	/var/log/messages
PostgreSQL			PostgreSQLプロセス (プロセスの存在有無)	psコマンド, pg_stat_activity
			SQL実行可否 (プロセスの正常動作)	psql
			PostgreSQLサーバログ (PostgreSQL上の問題有無)	/var/log/messages, pg_log/*
……				

シングルサーバ構成における監視

- 情報分析と対処
  - データベース異常時の分析と対処
    - データベース異常時の分析観点とそれぞれの状況に応じた対処案
    - DBサーバの死活、PostgreSQLプロセスの死活に着目
  - エラー発生時の分析と対処
    - エラー発生時の分析観点とそれぞれの状況に応じた対処案
    - OSログ、PostgreSQLのサーバログの内容に着目
  - 性能低下時の分析と対処
    - 性能低下時の分析観点とそれぞれの状況に応じた対処案
    - スロークエリの特定とEXPLAINの内容に着目

現在までの活動内容紹介
- 実機運用例 -

目的と背景

- PostgreSQL可用性構成の運用例が少ない（アンケートより）
- 監視やバックアップ、切替など個別の情報はあるものの、トータルでの実機を使った運用例の情報が少ない
- 机上で論理的にわかっていることでも、実際にやってみて気づくこともある
- 作業時間について目安レベルで良いので把握しておきたい。バックアップが1日で完了するレベルなのか、復旧に要する時間は日レベルなのか時間レベルなのかなど

実機運用例

- 2013年度の活動で実施予定の実機検証(の項目)

シングル構成

- インスタンス障害時のクラッシュ・リカバリ
- 論理バックアップからのリカバリ(pg_dump/pg_restore)
- Point-In-Time Recovery(PITR)
- 監視ケーススタディ

HAクラスタ構成

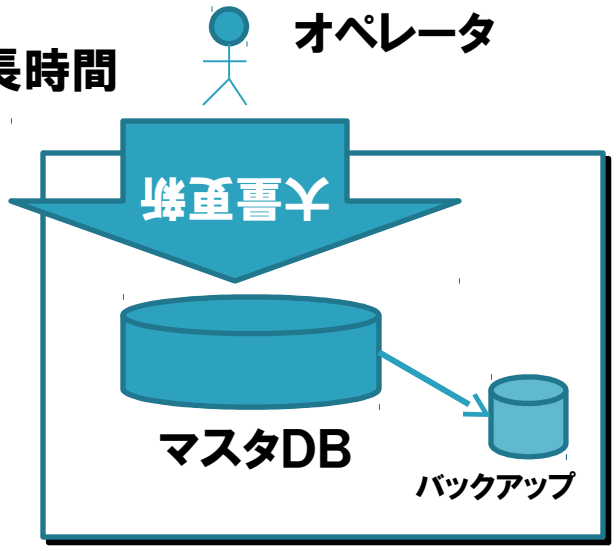
- スタンバイへの切替
- マスタのリカバリ

システム概要 –クラッシュ・リカバリ–

- 構成
  - 検証環境1で実施（シングル構成）
- 概要
  - 大量更新があるシステム
  - 性能要件によりチェックポイント間隔が比較的長いシステム
  - DBアクセスは数名で限定的
  - 元データは別にあるが、蓄積データが大量なため長時間要するリストアは避けたい
  - 組織内システムで緊急性はさほど高くない
  - 週一でバックアップを実施しているが、監視なし
- システム要件

項目	指標値	非機能要求グレード
サービス切替時間	24時間未満	A.1.2.2：レベル1
目標復旧地点（RPO）	5営業日前の時点	A.1.3.1：レベル1
目標復旧時間（RTO）	1営業日以内	A.1.3.2：レベル1

※「非機能要求グレード」可用性項目から、上記3項目を抜粋



検証シナリオ –クラッシュ・リカバリ–

- 検証目的
  - DBMSとしての基本的な障害回復が実施されることを実際に確認
- 検証イメージ
  - チェックポイントの間隔を調整
    - checkpoint_segments = 1000
    - checkpoint_timeout = 1h
  - データ更新中に電源断を実施
    - 後で確認できるように決まったデータを更新
  - サーバ再起動
  - PostgreSQL再起動
  - 更新したデータを確認

検証結果 –クラッシュ・リカバリ–

- 結果
  - 再起動時にロールフォワードが行われることで、最終トランザクションまでのデータが反映

    ⇒ 電源断のような障害発生時、データ復旧が可能

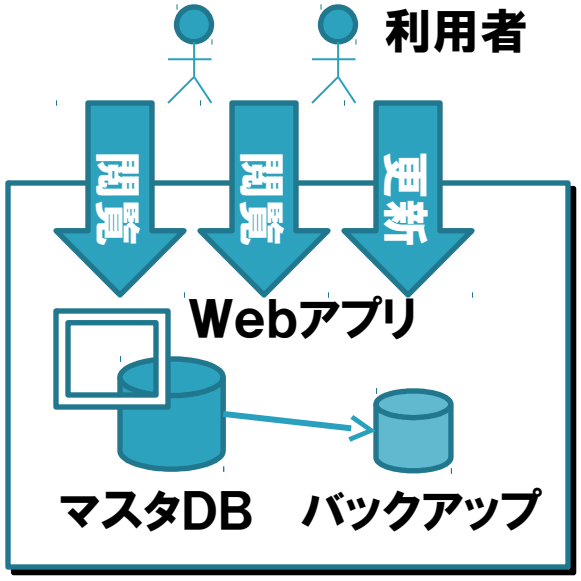
```
 [postgres@pgecons1 pg_log]$ cat postgresql-2013-10-10_142041.log
[2013-10-10 14:20:41 JST]LOG:  database system was interrupted; last known up at 2013-10-10 12:36:17 JST
[2013-10-10 14:20:41 JST]LOG:  database system was not properly shut down; automatic recovery in progress
[2013-10-10 14:20:41 JST]LOG:  redo starts at 135/7B000090
[2013-10-10 14:20:52 JST]LOG:  redo done at 135/D7FFD2C8
[2013-10-10 14:20:52 JST]LOG:  last completed transaction was at log time 2013-10-10 14:11:59.994082+09

testdb=# select * from pgbench_history order by mtime desc;
  tid   |  bid  |    aid    | delta |           mtime            | filler
--------+-------+-----------+-------+----------------------------+--------
  58045 |  7647 | 521189773 |  1435 | 2013-10-10 14:11:59.993444 |
  93451 |  9583 | 409026996 | -4937 | 2013-10-10 14:11:59.992281 |
  63657 |  9706 | 494699262 |  2752 | 2013-10-10 14:11:59.990973 |
  43897 |   461 | 239405160 |  2752 | 2013-10-10 14:11:59.989828 |
  20564 |  3888 | 764911388 | -3846 | 2013-10-10 14:11:59.989503 |
  46136 |  3762 | 945423791 |  3273 | 2013-10-10 14:11:59.988407 |
  60703 |  4469 | 858075989 | -2745 | 2013-10-10 14:11:59.988401 |
```

システム概要 –論理バックアップからのリカバリ–

- 構成
  - シングル構成
- 概要
  - 参照系中心のシステム
  - 更新は土日のバッチで月２回程度、その他は散発的に少量更新
  - DBアクセスは２０名程度
  - 直近データは再投入可能だが、蓄積データを失うと致命的
  - 障害からの復旧は時間レベルで可能
  - 週に一度休日にバックアップを実施しているが、監視なし
- システム要件

項目	指標値	非機能要求グレード
サービス切替時間	2時間未満	A.1.2.2：レベル2
目標復旧地点（RPO）	障害発生時点	A.1.3.1：レベル3
目標復旧時間（RTO）	12時間以内	A.1.3.2：レベル2

※「非機能要求グレード」可用性項目から、上記3項目を抜粋



検証シナリオ –論理バックアップからのリカバリ–

- 検証目的
  - データサイズ増加によるバックアップ&リカバリ時間の傾向を確認
- 検証イメージ
  - 火曜日からたまにエラーが発生
  - データが壊れた可能性が高いが対象データの特定ができてない
  - カスタム形式でバックアップを取得
  - 正しいデータでリカバリしたい
- 計測対象
  - pg_dump実行時間
  - pg_restore実行時間
  - バックアップデータ量

検証結果 –論理バックアップからのリカバリ–

- 論理バックアップの特徴
  - データは以下のようにデータベースを検索した結果を保存
    - pg_dump実行時のスナップショットデータ
    - オプションで圧縮形式で取得するのが一般的
  - pg_dumpはクライアントとしてデータ取得するため、取得可能であれば**データ破損は発生していない可能性が非常に高い**
  - ただし、バックアップ取得時点以後の更新分はリカバリ不可

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET client_encoding = 'SQL_ASCII';
SET standard_conforming_strings = on;

・・・
584     1       0
585     1       0
586     1       0
\.
```

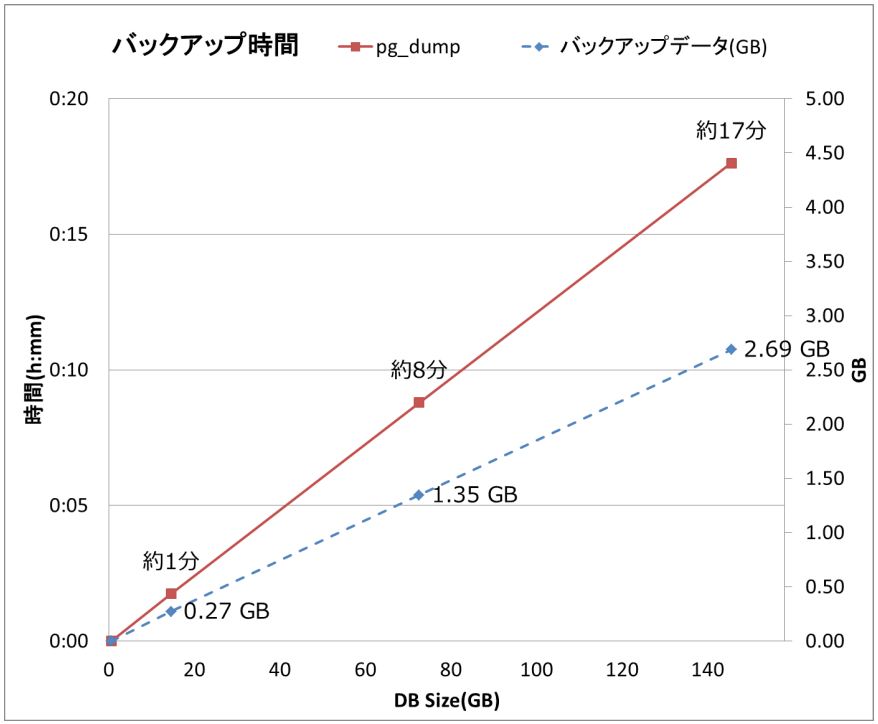
例）データが壊れている際にpg_dumpを取得した例

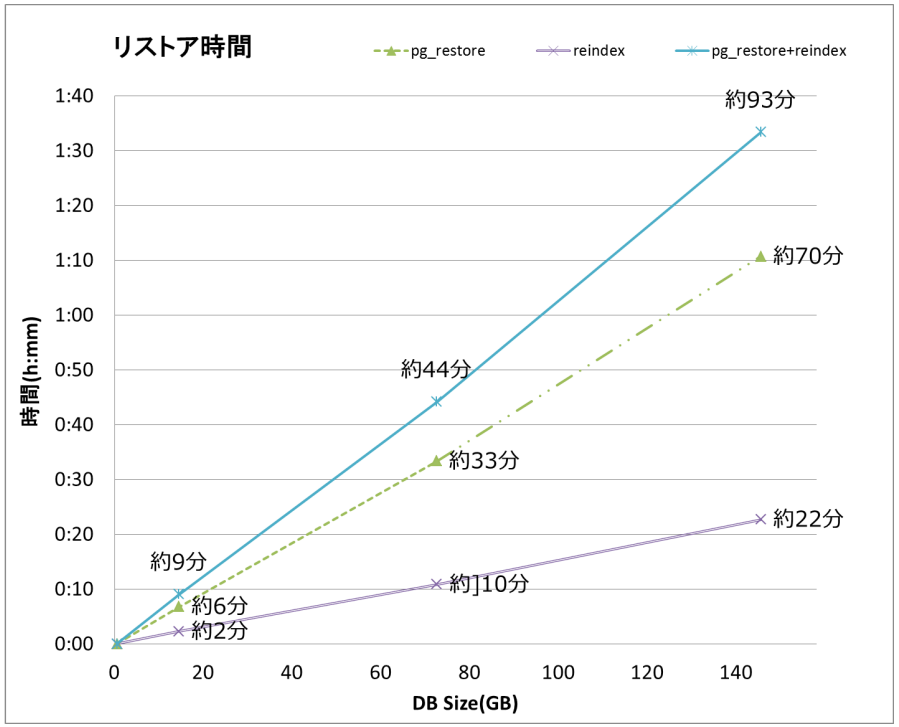
万が一、テーブルデータが破損している場合、pg_dump実行時にエラーになる
この状態で物理バックアップをとっても、**データ破損に気がつかない**

```
pg_dump: Dumping the contents of table "pgbench_accounts" failed: PQgetCopyData() failed.
pg_dump: Error message from server: pg_dump: The command was: COPY public.pgbench_accounts (aid, bid,
abalance, filler) TO stdout;
```

検証結果 –論理バックアップからのリカバリ–

- データサイズとバックアップリストア所要時間の目安
  - データ量に比例して線形で増加するため見積もりは簡単
  - リストア時間の方が長い
    - リストア時REINDEXが必要であることに注意





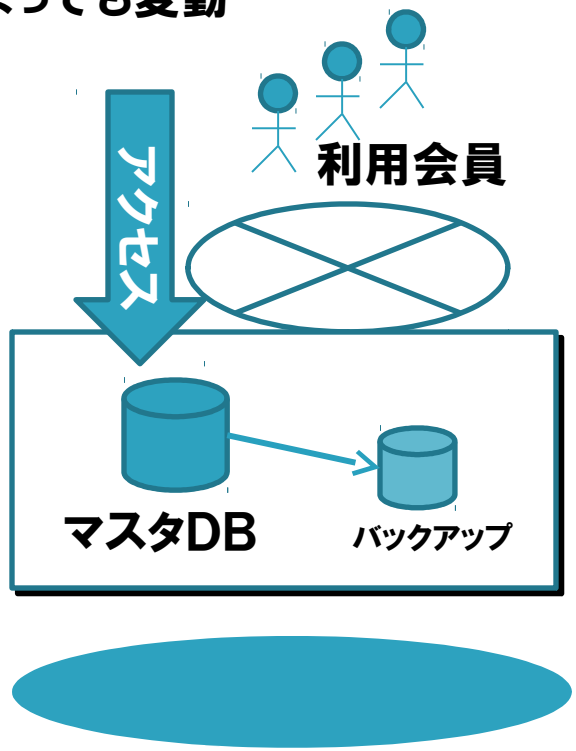
両グラフの時間軸のスケールが違うことに注意

システム概要 –監視ケーススタディ–

- 構成
  - シングル構成
- 概要
  - インターネット経由でのDBアクセス
  - 利用者数は多数だが同時利用は時期や時間帯によっても変動
  - 増減はあるもののトータルのデータ量は年々増加
  - 会員向けサービスで、長時間のサービス停止に至らないレベルであれば許容可能
  - 毎日バックアップを実施し、監視あり
- システム要件

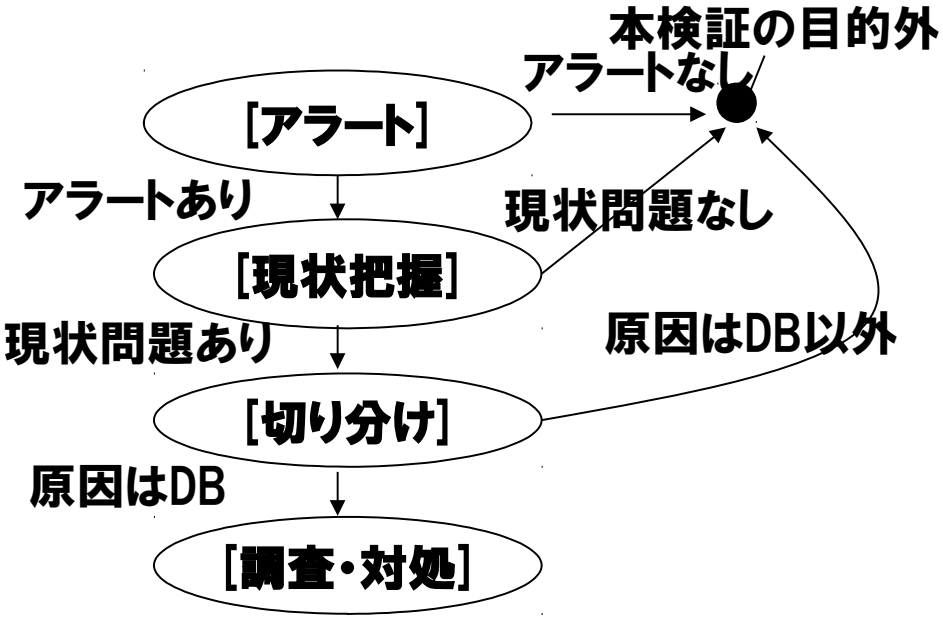
項目	指標値	非機能要求グレード
サービス切替時間	60分未満	A.1.2.2：レベル3
目標復旧地点（RPO）	障害発生時点	A.1.3.1：レベル3
目標復旧時間（RTO）	12時間以内	A.1.3.2：レベル2

※「非機能要求グレード」可用性項目から、上記3項目を抜粋



検証結果(予定)-監視ケーススタディ-

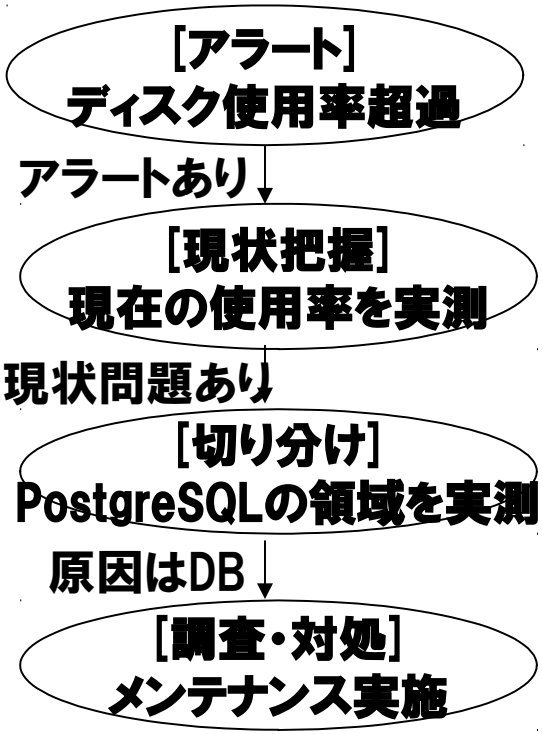
- 死活監視で収集した情報、エラーメッセージ、性能監視で収集した情報からの分析(解釈)と対処方針案を整理
  - それぞれの項目での注意点
    - アラート
    - 現状把握
    - 切り分け
    - 調査・対処



検証シナリオ －監視ケーススタディ－

■ 検証シナリオ1：ディスク容量不足

[アラート]
ディスク使用率が超過したことを示すアラートがあがった
[現状把握]
アラートを受け、現状把握として現在のディスク使用率を
dfコマンドを用いて実測
```
# df -h
```
[切り分け]
現状把握の結果、あるデバイスの使用率が閾値を超えていた。
そのデバイス上にはデータベースクラスタが配置されていたため
下記のSQL関数等を用いてデータベースサイズを測定
```
$ SELECT pg_database_size ( 'mydb' ) ;
$ SELECT pg_relation_size ( 'mytbl' ) ;
```
[調査・対処]
切り分けの結果、mytblが必要以上に肥大化していた。さらに下
記のSQL関数で肥大化している原因を調査し、不要な領域が膨
れあがっていることが判明した。根本原因への対処は引き続き
行うこととし、暫定対処として夜間にVACUUM FULLを実行する
こととした
```
$ SELECT n_dead_tup FROM pg_stat_user_tables
                    WHERE relname = 'mytbl' ;
```



まとめ＆今後の予定

- PostgreSQLのシステム構成
  - 可用性要件に応じた様々な構成が可能
  - 各構成での適切なバックアップと監視が重要なポイント
- 今後の予定
  - 2014年3月までに以下内容にて成果物を作成
    - 各構成の概要、特徴、考慮事項
    - バックアップ手法の概要、特徴、考慮事項
    - 死活・性能監視項目および収集済み情報の分析・対処法
    - 実機運用例
  - 成果報告
    - 2014年4月（予定）のPGEConsイベントにて報告

ぜひ、ご期待ください！

付録：検証環境について

- 実機運用例でご紹介した各検証は、株式会社日立製作所様から検証環境をご提供いただきました
- PostgreSQLエンタープライズ・コンソーシアムとしてお礼申し上げます

付録：検証環境１ （提供： 株式会社 日立製作所）

■ 実機検証での使用機器／設備

ハーモニアス・コンピテンス・センター

★BS500 (BladeSymphony BS540A)

PostgreSQLサーバとして使用

CPU:インテルXeon E5-4610 (2.40GHz) 6コア×4
メモリ:256GB
内蔵HDD:900GB×2 RAID1

★HA8000 (HA8000/RS220-h HM)

クライアントとして使用

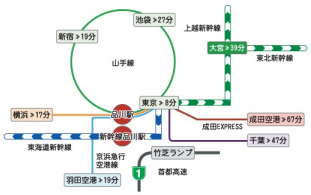
CPU:インテルXeon E5-2690 (2.90GHz) 8コア×2
メモリ:32GB
内蔵HDD:900GB×4 RAID6

検証ルーム

LAN

FC接続

サーバルーム



JR品川駅　港南口より徒歩約3分
京浜急行品川駅　高輪口より徒歩約5分

★HUS100 (Hitachi Unified Storage 130)

ストレージとして使用

HDD (600GB x 5 RAID5) データ用
SSD (200GB x 3 RAID5) WAL用
HDD (600GB x 5 RAID5) アーカイブ用
HDD (600GB x 5 RAID5) バックアップ用

HDD (600GB x 5 RAID5) データ用
HDD (600GB x 5 RAID5) WAL用
HDD (600GB x 5 RAID5) アーカイブおよびバックアップ用

PostgreSQL Enterprise Consortium

PostgreSQL Enterprise Consortium