FUJITSU
shaping tomorrow with you

DBMSの課題に振り回されないための EDB Postgres活用術

株式会社 富士通ソーシアルサイエンスラボラトリ
2019年8月1日 (木)
小山田 政紀

アジェンダ

- 当社の紹介

- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
  3. Oracle互換機能でスムーズに移行しよう

- まとめ

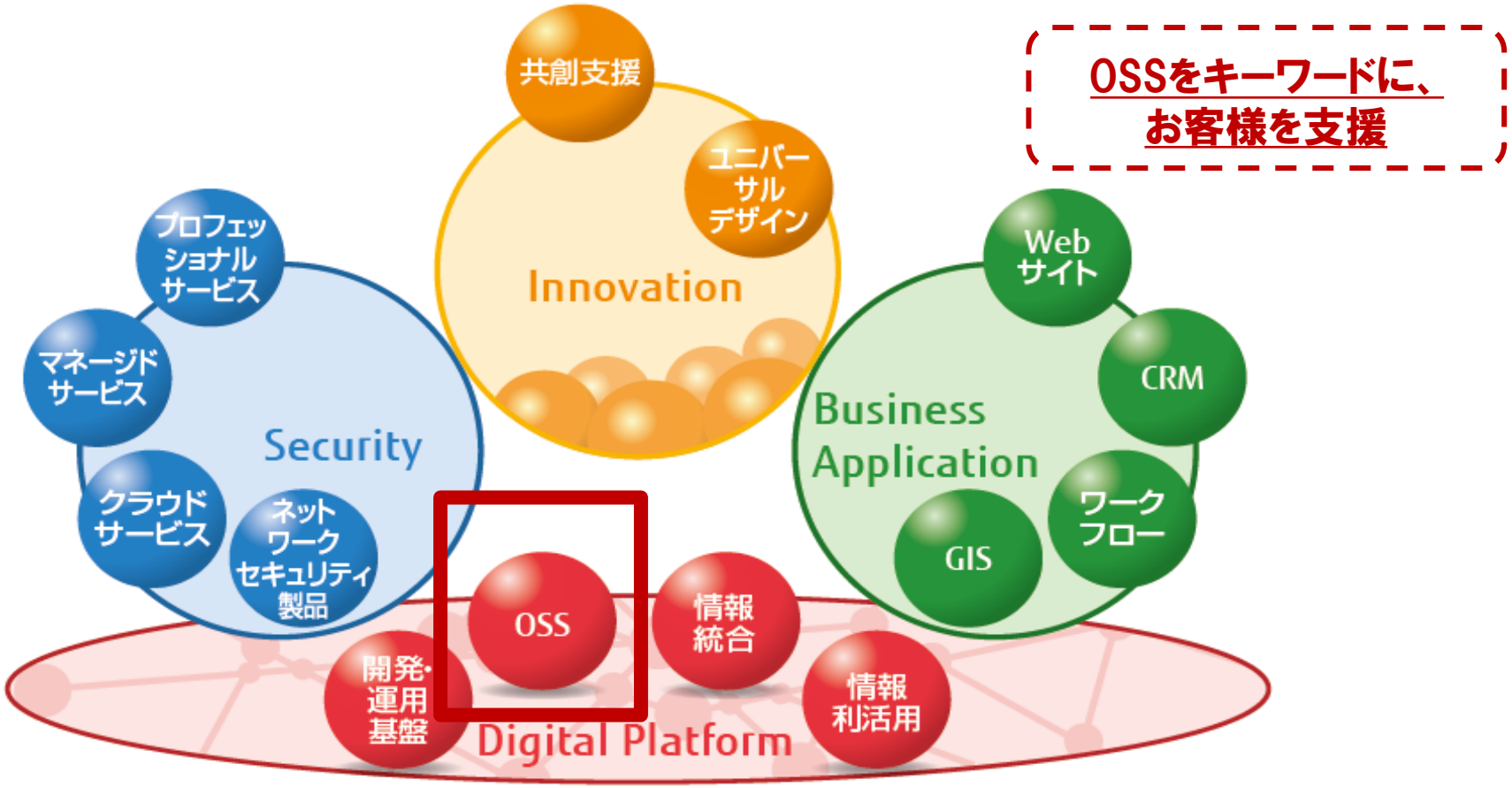
当社の紹介

- 株式会社 富士通ソーシアルサイエンスラボラトリ(略称:富士通SSL)
  - http://www.fujitsu.com/jp/group/ssl/
  - http://www.facebook.com/FujitsuSSL
- 事業所
  - 武蔵小杉本社
  - 関西事業所(大阪)
  - 東海事業所(名古屋)
  - 東海事業所刈谷分室
- 設立
  - 1972/07/12 (48年目)
- 社員数
  - 1,158名(2019年3月末現在、連結ベース)
- 事業内容
  - SI事業
  - ソリューション事業 (PoweredSolution)
- 関係会社
  - 株式会社SSLパワードサービス
  - 株式会社富士通SSLハーモニー

FUJITSU

PoweredSolution

- 富士通SSLのソリューション群 PoweredSolution は、お客様と共に、
  お客様にとっての新しい価値をデジタルテクノロジーの力で創ることにより、
  豊かな社会を実現します。



OSSソリューションマップ

FUJITSU

お客様のシステム要件に合わせて、最適なソリューションを提案します。

PoC ／ 設計 ／ 構築		運用・保守
Elasticsearch プロフェッショナルサービス	Elasticsearch	OSSサポート サービス
PostgreSQL プロフェッショナルサービス	PostgreSQL EDB Postgres Fujitsu Enterprise Postgres Symfoware	EDB Postgres サポートサービス
Zabbix プロフェッショナルサービス	Zabbix	
OSSミドルウェア プロフェッショナルサービス ※対象OSSは随時更新予定	MySQL. MongoDB その他	

※上記OSS以外についてはお問い合わせください

OSS導入ソリューション

OSSサポート ソリューション

データーベース移行ソリューション

EDB Postgres

OSS導入ソリューション

FUJITSU

OSSプロフェッショナルサービス

OSS専任エンジニアによる技術支援サービス

- **対象**
  - 富士通グループのSE
  - お客様（情シス部門）など

  OSSの適用技術（設計/導入/テスト）の技術支援を受けたいPJ
- **対応内容**
  - PoC（概念検証）
  - 設計構築支援
  - 設計構築代行
  - 性能改善支援（PostgreSQL/Elasticsearch）
- **支援イメージ**
  - 時間パターン（25時間/月～）

    技術支援内容に応じて必要時間数を決定の上契約
  - 代行パターン（設計構築代行のみ）

発表の要旨

- プロフェッショナルサービスにてご相談いただいた、
  DBMSの課題の中からよくある課題を取り上げ、対応内容をご紹介

- 同様の問題が発生した場合は、本発表内容が参考になれば幸いです。

アジェンダ

FUJITSU

- 当社の紹介

- **DBMSの課題に振り回されないためのEDB Postgres活用術**
  1. **バージョンアップをスムーズに実施しよう**
  2. **性能問題を解決しよう**
  3. **Oracle互換機能でスムーズに移行しよう**

この2テーマを中心にお話しします

**他のセミナーでも取り上げられており、
簡単な紹介程度にとどめます。**

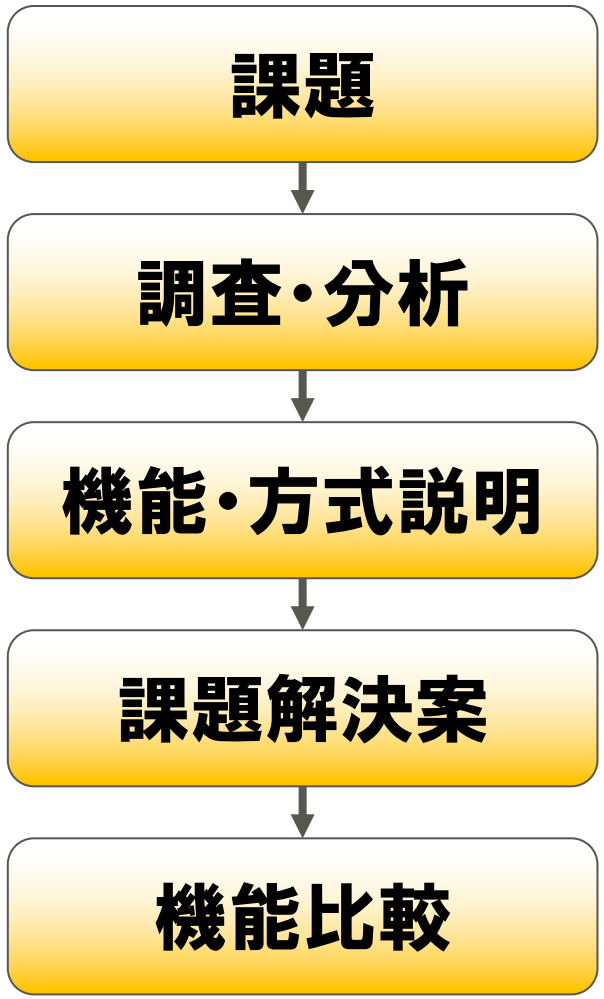
- まとめ

本発表資料のフォーマット

FUJITSU

■ 本資料ではDBMSの課題への対応について、以下の流れで説明します。

EDB ENTERPRISEDB



課題や改善要望についての説明

EDB独自機能

調査のために取得した情報や分析結果についての説明

調査や課題解決に用いる機能についての説明

課題解決や改善方法についての説明

各節で調査方法や改善方法として用いた機能の比較
(PostgreSQL vs EDB Postgres(差分がある場合のみ))

アジェンダ

- 当社の紹介

- **DBMSの課題に振り回されないためのEDB Postgres活用術**
  1. **バージョンアップをスムーズに実施しよう**
  2. 性能問題を解決しよう
  3. Oracle互換機能でスムーズに移行しよう

- まとめ

1. バージョンアップをスムーズに実施しよう-1/11

課題

FUJITSU

■課題

1. 古いバージョンのPostgreSQLを利用しているが、
   End-Of-Life（EOL）を過ぎており、セキュリティパッチなどが提供されない　※1
2. データ増加と共にSQLの実行時間が伸びているため、
   最新バージョンに搭載された機能（パラレルクエリ、パーティショニング）にて、
   性能改善を検討したい。

※1 リリースより5年間でEOL

■解決したい課題

■バージョンアップをスムーズに実施したい

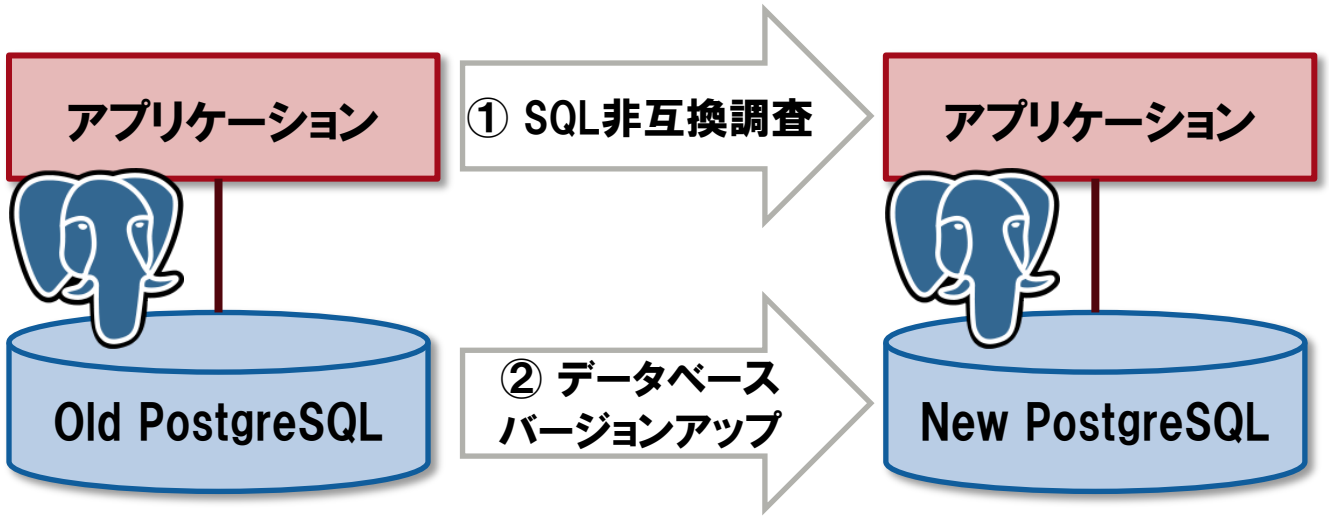
調査・分析

FUJITSU

■バージョンアップ手順

■弊社では次のような流れでバージョンアップを実施

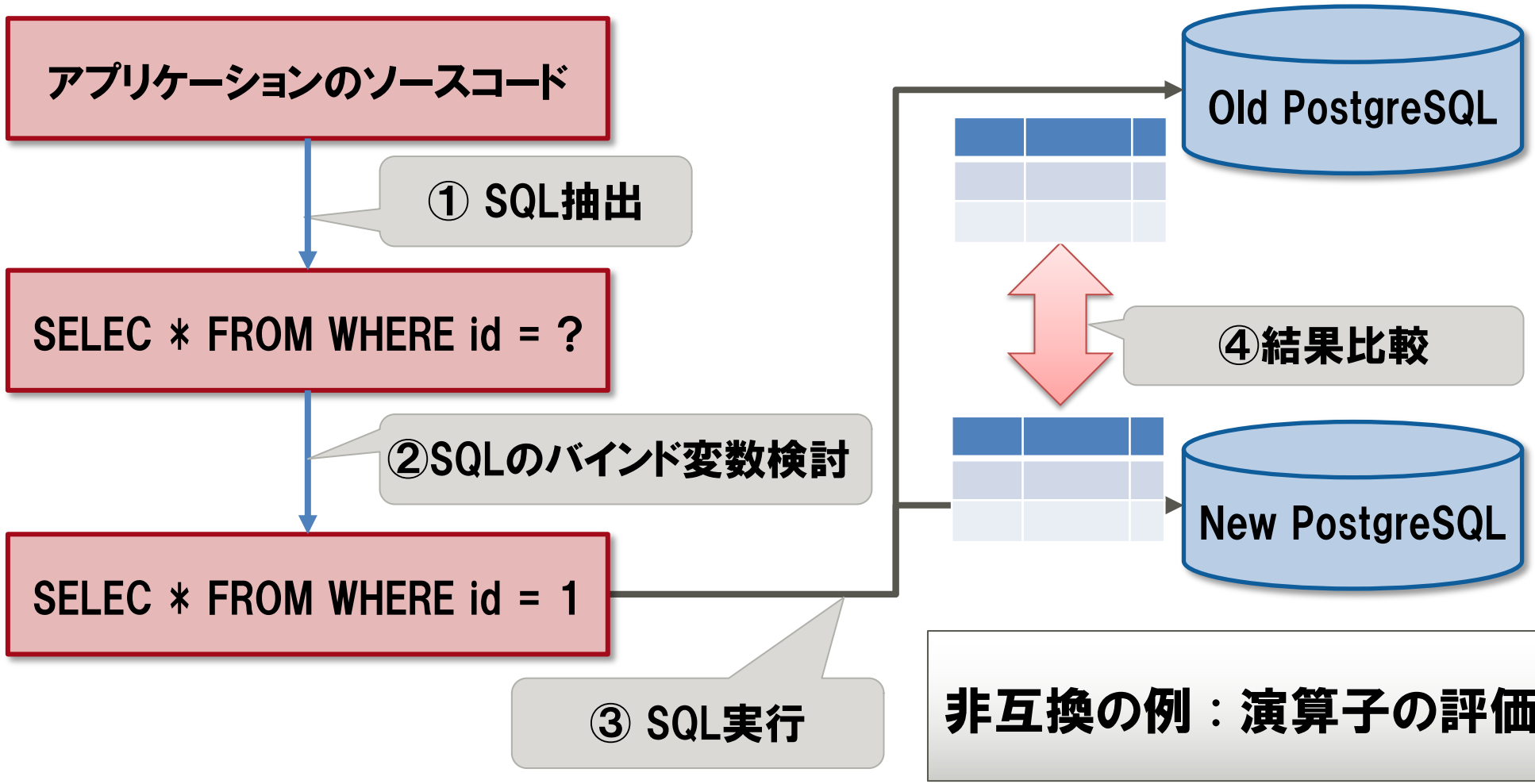
1. SQL非互換調査
2. データベースバージョンアップ手法の検討



- 非互換箇所抽出
- 非互換修正方法提示
- DB移行手順提示
- DB移行試験手順提示

SQL非互換調査

チェックを弊社独自ツールで（一部）自動化し、非互換SQL抽出を短期間で実現



非互換の例：演算子の評価順変更（9.5）

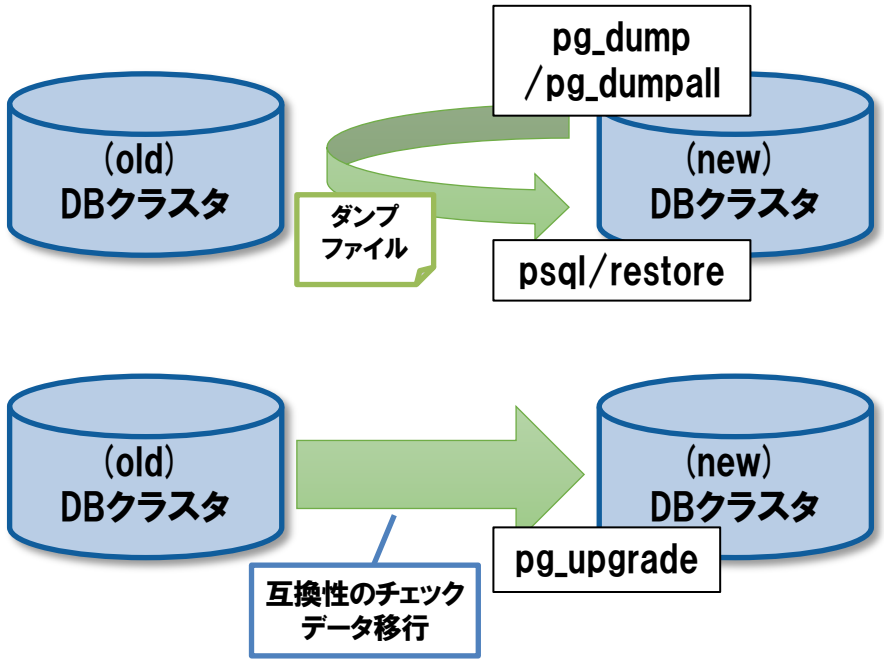
■データベースメジャーバージョンアップの主な手法

1. ダンプ/リストア

- 利用機能：pg_dump/pg_dumpall（ダンプ取得）
  pg_restore/psql（リストア）

2. アップグレード

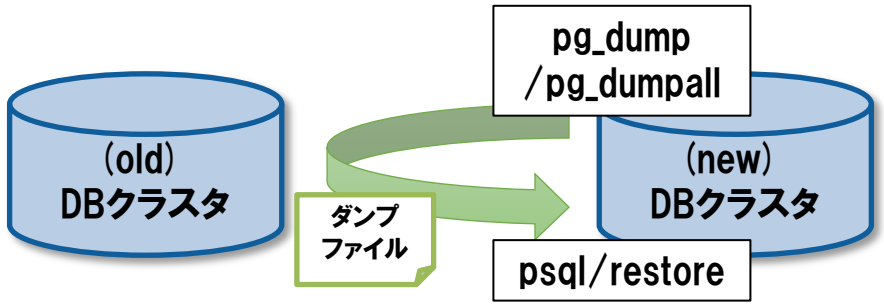
- 利用機能：pg_upgrade
- 制限：同一サーバ上での
  バージョンアップのみ実行可能



■データベースメジャーバージョンアップの主な手法

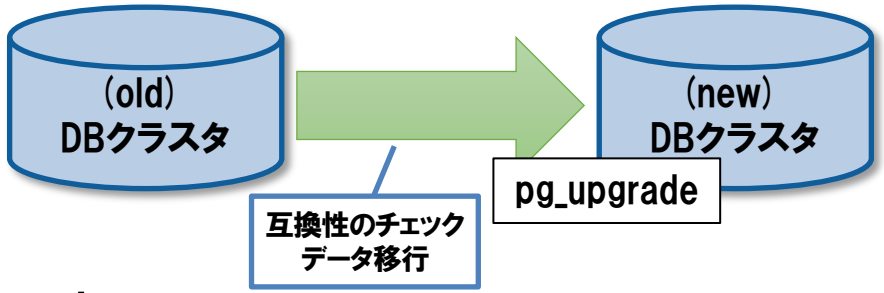
1. ダンプ/リストア

- 利用機能：pg_dump/pg_dumpall（ダンプ取得）
  pg_restore/psql（リストア）



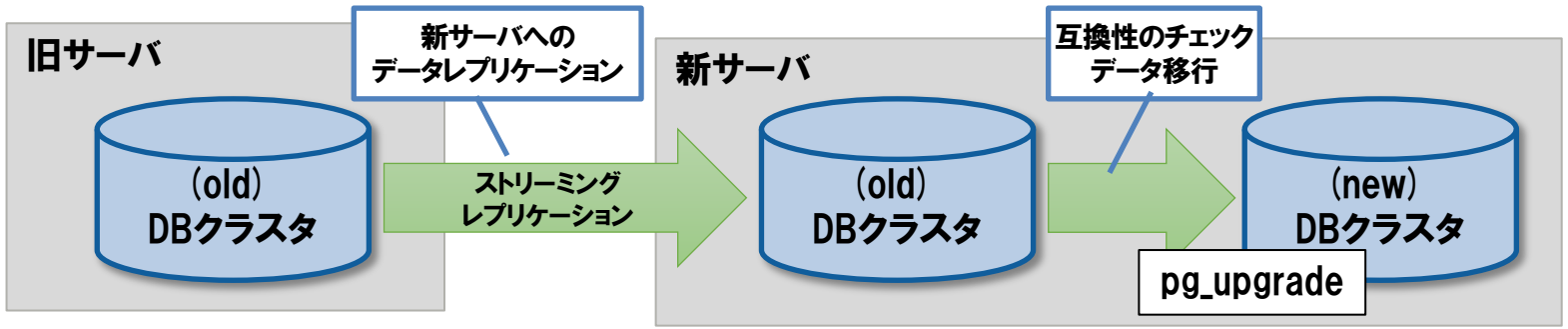
2. アップグレード

- 利用機能：pg_upgrade
- 制限：同一サーバ上での
  バージョンアップのみ実行可能



3. レプリケーション+アップグレード

- 利用機能：ストリーミングレプリケーション + pg_upgrade

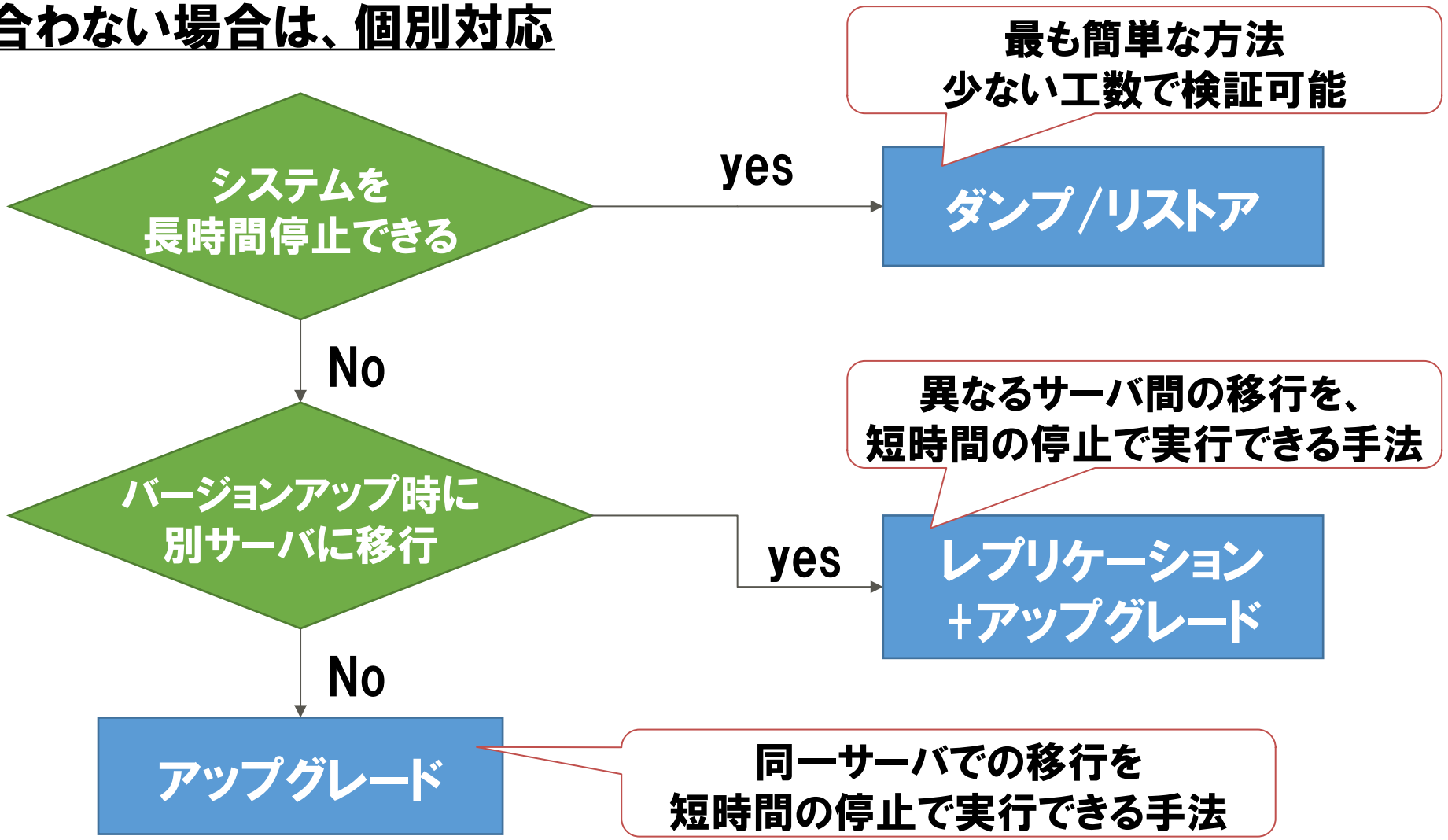


■バージョンアップ各手法の比較結果

評価項目/手法	ダンプ/リストア	アップグレード	レプリケーション +アップグレード
バージョンアップにかかる時間	△（データサイズに比例して、時間が顕著に増加）	○（短時間で完了）	○（短時間で完了）
手順の実施難易度	易	並	並
異なるDBサーバへのバージョンアップ	可能	不可	可能
対応可能な旧バージョン（PostgreSQL11時点）	8.0～	8.4～	9.0～（ストリーミングレプリケーション利用のため）
注意事項	-	古いバージョンと新バージョンで同居させる必要があるため、考慮が必要（環境変数など）	

■9.0以降のメジャーバージョンアップ手法選択

※要件に合わない場合は、個別対応



1. バージョンアップをスムーズに実施しよう-8/11

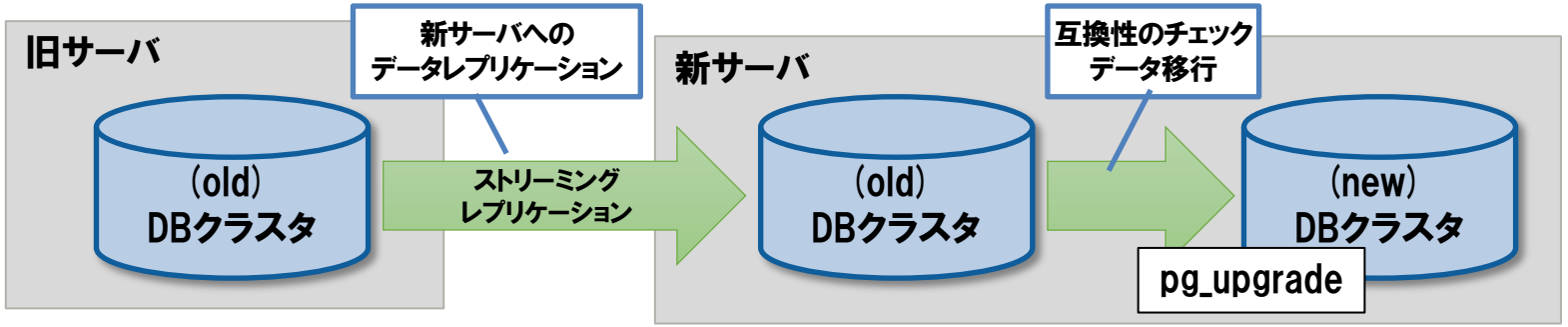
トラブル改善案

FUJITSU

■PostgreSQL 9.0以降のバージョンアップ時には、
弊社からはレプリケーション+アップグレード方式を提案・実施

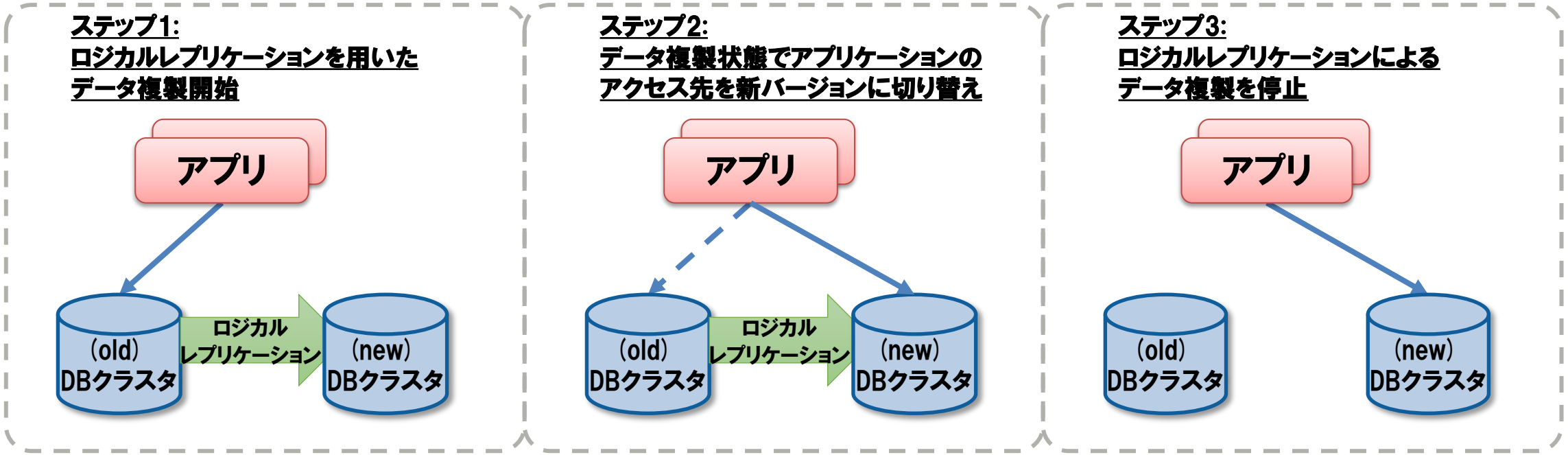
■レプリケーション+アップグレード利用時の注意点

1. レプリケーションによる性能劣化の確認
   - 更新処理では5～10%（処理内容やハードウェアに依存）
   - 物理バックアップ取得時の性能影響も考慮
     - 既に物理バックアップを取得しているのであれば、既存のバックアップファイルを活用
   - レプリケーションする期間は最小限にする工夫が必要
2. 異なるCPUアーキテクチャではレプリケーションがサポートされない

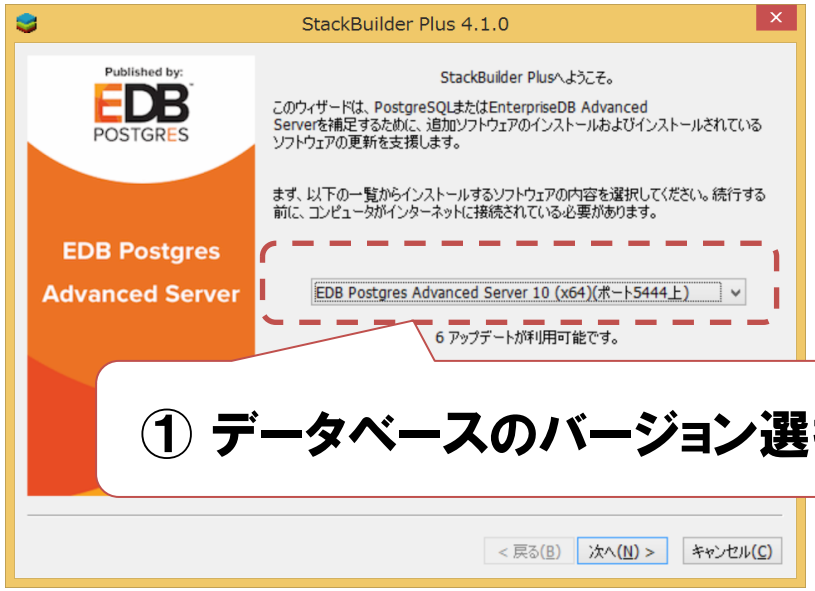


ロジカルレプリケーション（PostgreSQL10新機能）を用いたバージョンアップ

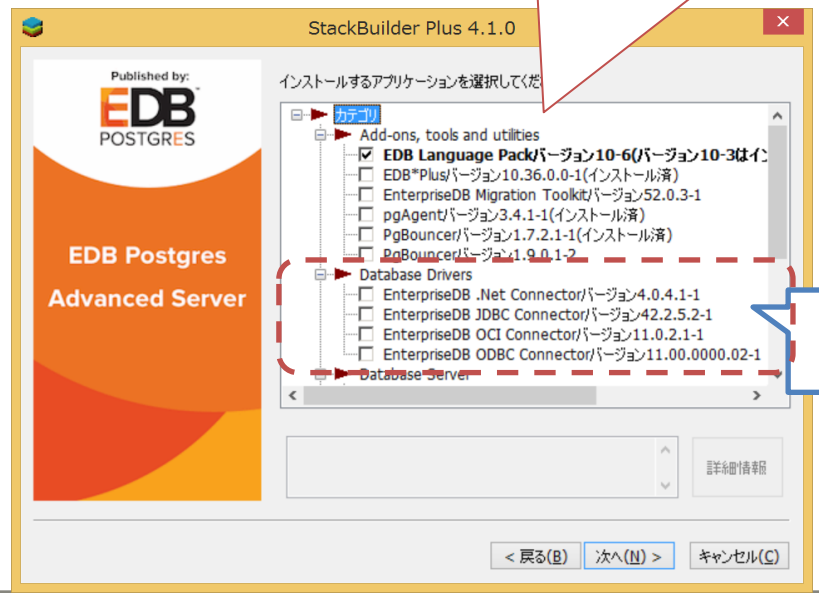
- ロジカルレプリケーションの特徴
  1. 異なるバージョン間でレプリケーションが可能（例：10と11のレプリケーション）
  2. レプリケーション先で更新処理実行が可能
- 下記ステップで停止時間を限りなく短くしたメジャーバージョンアップが可能に
- ロジカルレプリケーションで注意が必要な制限事項　：シーケンスが払い出す番号、ラージオブジェクト、マテリアライズドビューなどは複製されない



- EDB Postgresの場合、利用するバージョン（10,11など）に紐づく、周辺ソフトウェアの入手が容易である。
  - 入手可能な周辺ソフトウェアはEDB社が最適と判断したバージョン
  - EDB社が提供するStackBuilder（すたっく・びるだー）にて実現



① データベースのバージョン選択



②入手するソフトウェアを選択
（EDB社動作確認済み）

各種ドライバなど

機能比較

FUJITSU

■ バージョン11で性能の改善効果が期待できる機能

機能名	PostgreSQL	EDB Postgres（PostgreSQLに機能追加）
パラレルクエリ	○	◎
	主な処理方式で 並列実行をサポート	実行多重度をヒント句で制御可能
パーティショニング	○	◎
	宣言的パーティショニングを レンジ・リスト・ハッシュ型でサポート	・多パーティショニング時の処理速度に関する改善 （PostgreSQL12にて搭載予定）
リソース利用制御	×	○
	なし（OS機能の利用）	EDB リソースマネージャによるリソースグループ 定義が可能

■ その他、PostgreSQLの基本機能も従来の課題が解決され性能改善

⇒ バージョンアップ+新機能活用により大幅な性能改善

アジェンダ

- 当社の紹介

- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
  3. Oracle互換機能でスムーズに移行しよう

- まとめ

2. 性能問題を解決しよう

課題

FUJITSU

■ データベースが遅くなった場合の調査項目

どうやって性能問題の原因を調査し、改善していけばいいのだろう？

■ 今回は3つの事例を基に、性能改善までの調査方法を紹介

1. 想定した性能に達しない事例
2. 不定期に遅延が発生する事例

→ 遅延の発生条件が不明な場合には**待機イベントの調査**が有効

3. 特定のSQLで遅延が発生する事例

→ 特定のSQLの遅延原因を解析するには**実行計画の調査**が有効

2. 性能問題を解決しよう

FUJITSU

■ 待機イベントとは

■ DBMS内部処理の内、待ちが発生している処理

- 参考：PostgreSQL11.1文書 28.2. 統計情報コレクタ 表28.4 wait_eventの説明
  https://www.postgresql.jp/document/11/html/monitoring-stats.html#WAIT-EVENT-TABLE

■ 待機イベント取得機能

項番	機能名	概要	取得できる待機イベントの種別	PostgreSQL	EDB Postgres
1	pg_stat_activity ビュー	ビューを参照した時のセッション（データベースへの接続）毎の待機イベントの種類/待機イベント名を取得	現時点で発生している待機イベント	○	○
2	EDB Wait States モジュール	セッション毎に発生した待機イベントを定期的に記録	セッション毎に発生した待機イベント情報	-	○
3	DRITA	ある期間におけるDBクラスタ全体やセッションごとの待機イベントの発生回数や待機時間の情報をレポートとして出力可能	任意の期間で発生した待機イベントの統計情報	-	○

2. 性能問題を解決しよう

FUJITSU

■ 実行計画とは

■ SQLで結果を取得するために、DBMS内部で実行されるデータ取得方法やデータの結合方法などの処理方法が記載されたもの（EXPLAINコマンドで表示）

- 参考：PostgreSQL11.1文書 EXPLAIN
  https://www.postgresql.jp/document/11/html/sql-explain.html

■ 実行計画が変わるとSQLの実行速度も変わる（処理方法が変わると実行時間が変わる）

■ 実行計画改善※1に役立つ機能（SQLの性能改善）

項番	機能名	役割	概要	Postgre SQL	EDB Postgres
1	Index Advisor	性能改善が見込めるインデックスの提示	クエリプランナ（オプティマイザ）と連携し、SQLの処理コストを削減するために有効なインデックスを提示	-	○
2	Optimizer Hints	ヒント句で指定した実行計画の選択	ヒントで指定された実行計画をプランナに選択させる	-	○

※1 基本的にはDBMSが最適な実行計画を選択するが、必要なインデックスがないなどの条件下では最適ではない実行計画が選択される場合がある。

アジェンダ

- 当社の紹介
- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
     a. 想定した性能に達しない事例
     b. 不定期に遅延が発生する事例
     c. 特定のSQLで遅延が発生する事例
  3. Oracle互換機能でスムーズに移行しよう
- まとめ

2.1. 想定した性能に達しない事例-1/7

課題

FUJITSU

想定した性能に達しない事例

- 個別のSQLでは性能問題は発生していない
- 複数のSQLが平行・連続して実行する場合に性能問題が発生
- データベースクラスタは同一ディスク上に格納

（本システム）
性能測定結果：1272TPS

※性能要件である2000TPS以上を達成する必要がある

2.1. 想定した性能に達しない事例-2/7

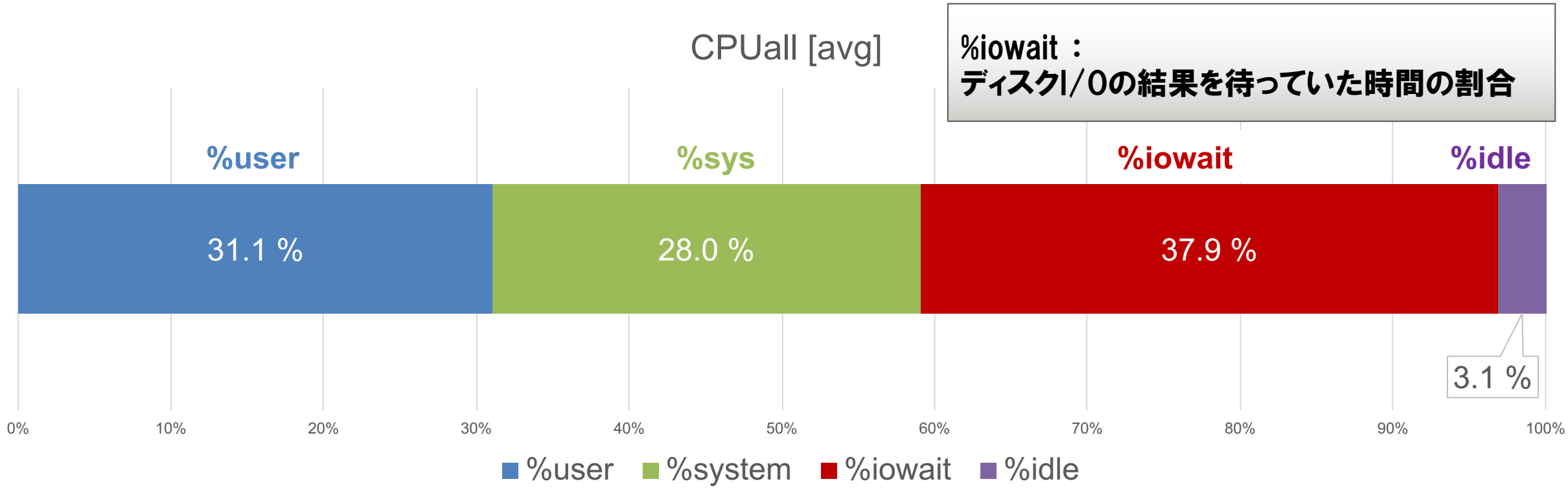
課題

FUJITSU

想定した性能に達しない事例

性能問題調査のため、事象発生期間中のDBサーバのCPU使用率を確認

- %iowaitの割合が大きく、%userの割合が小さい ⇒ CPUを効率よく使用できていない
- %iowaitの発生原因を調査 ⇒ 待機イベントを取得し、%iowaitの原因を特定



2.1. 想定した性能に達しない事例-3/7

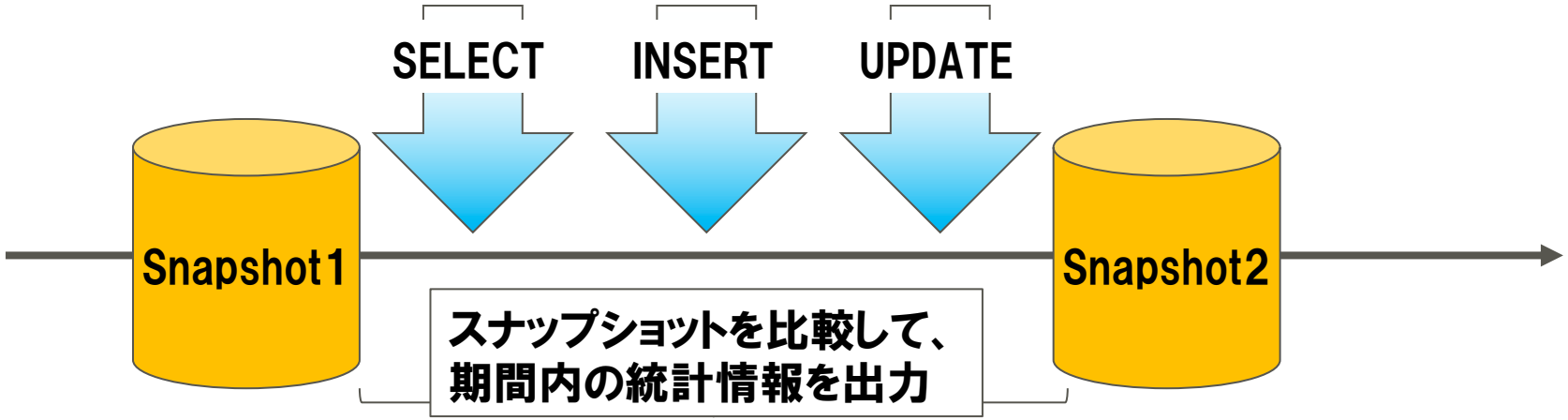
機能・方式説明

FUJITSU

EDB ENTERPRISEDB

DRITA (どりた)

- ✓ DRITAは<u>各待機イベントが発生した回数と待機時間の累計値</u>を記録
- ✓ スナップショット取得によりある時点での性能に関する情報を保存可能
- ✓ スナップショットの差分を用いて、ある時間帯の性能に関する情報をレポートで出力可能



-Snapshot内容-

・待機情報などの現在の統計情報を記録

2.1. 想定した性能に達しない事例-4/7

機能・方式説明

FUJITSU

EDB ENTERPRISEDB

DRITA (どりた)

- 取得したスナップショットを使用し、ある期間内のシステム全体やセッション毎の待機イベントの情報をレポートとして出力

項番	レポート項目	レポート内容
1	レポート概要	スナップショットを取得した時間帯や対象データベースなど
2	テーブルやインデックスのページ数	テーブルやインデックスごとのページ数の上位
3	データベースやテーブルの統計情報	キャッシュヒット率やIO発生状況など
4	待機イベントの統計情報	期間内の待機イベントの発生回数と待機時間
5	パラメータ情報	postgresql.confで定義されたデータベースのパラメータ

2.1. 想定した性能に達しない事例-5/7

機能・方式説明

FUJITSU

EDB ENTERPRISEDB

DRITAでの待機イベント調査

レポート内容（待機イベントの統計情報のみ抜粋）

```
=# SELECT * FROM sys_rpt(28, 33, 10);
                                  sys_rpt
-----------------------------------------------------------------------------
 WAIT NAME                                 COUNT      WAIT TIME       % WAIT
 ---------------------------------------------------------------------------
 wal flush                                 294277     25564.566293    64.44
 db file read                              228520     10437.173054    26.31
 wal write                                 35915      1761.193818     4.44
 wal file sync                             36009      1746.423800     4.40
 other lwlock acq                          127        117.325016      0.30
```

待機イベントの上位には、DBとWALに対しての処理
⇒DBとWALの領域を分けることが有効だと判断

2.1. 想定した性能に達しない事例-6/7

課題解決案

FUJITSU

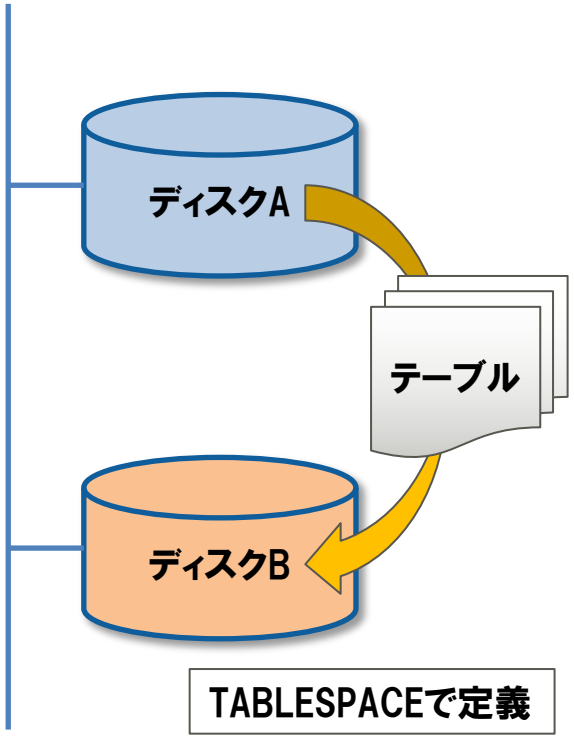
性能問題改善方法

統計情報から、以下の待機イベントの影響が大きいことが判明

- wal flush : WALのディスクへのフラッシュ待ち
- db file read : データベースファイル読み込み待ち

頻繁にディスクアクセスが行われるようなシステムでは テーブルとWALを同一のディスクに配置した状態だと、ディスクI/Oの待ちが発生し、性能が低下することがある

テーブルとWALを別のディスクに配置することで ディスクI/Oの負荷を分散させ、性能改善を図る



2.1. 想定した性能に達しない事例－7/7

課題解決案

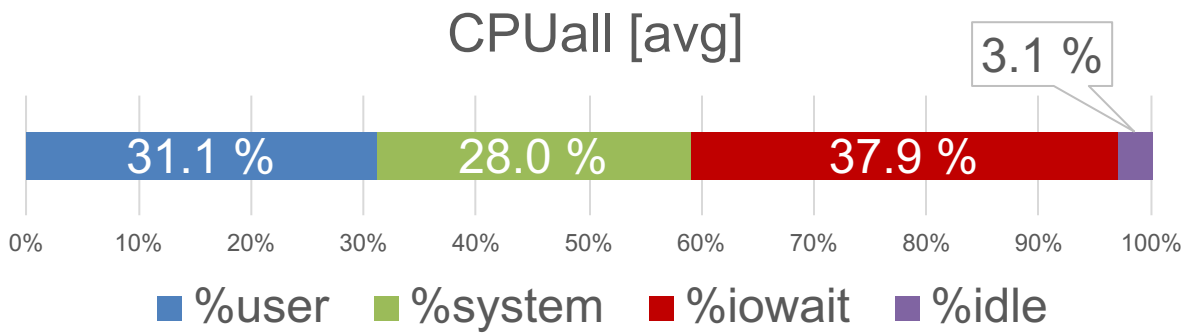
FUJITSU

性能問題改善結果

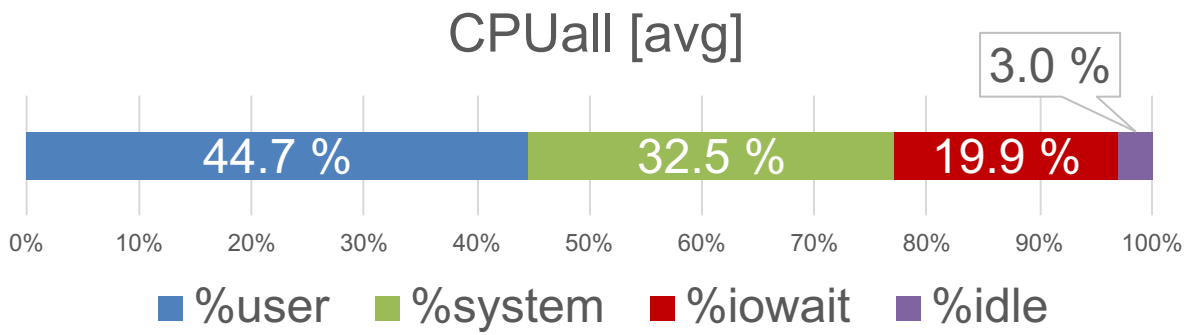
1つのディスクに集中していたI/O負荷が分散した結果、%iowaitが低下し、性能が改善

性能測定結果が2260TPSとなり、性能問題は改善したと判断

改善前



改善後



%iowaitを抑えることで性能改善

アジェンダ

- 当社の紹介
- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
     a. 想定した性能に達しない事例
     b. 不定期に遅延が発生する事例
     c. 特定のSQLで遅延が発生する事例
  3. Oracle互換機能でスムーズに移行しよう
- まとめ

2.2. 不定期に遅延が発生する事例-1/6

課題

FUJITSU

不定期に遅延が発生する事例

- 3秒以上遅延する場合のSQLはログに出力しているが、
  今までの運用では特に出力されるSQLはなかった。
- 利用者急増に伴い、最大接続数(max_connections)の値だけを変更した。
- PostgreSQLログの出力結果は以下のように特定のテーブルに対しての
  更新SQLで不定期に遅延が発生している。

```
(PostgreSQLログに出力された遅延SQL)
                                                    :
[XXXX-XX-XX XX:XX:XX.XXX JST] [XXXXX] [XXXXX] [enterprisedb] [OOOOO] [XXXX]
 LOG:  duration: 3223.444 ms  statement: UPDATE pgbench_tellers SET ～;
                                                    :
[XXXX-XX-XX XX:XX:XX.XXX JST] [XXXXX] [XXXXX] [enterprisedb] [OOOOO] [XXXX]
 LOG:  duration: 3027.313 ms  statement: UPDATE pgbench_tellers SET ～;
                                                    :
```

2.2. 不定期に遅延が発生する事例-2/6

調査・分析

FUJITSU

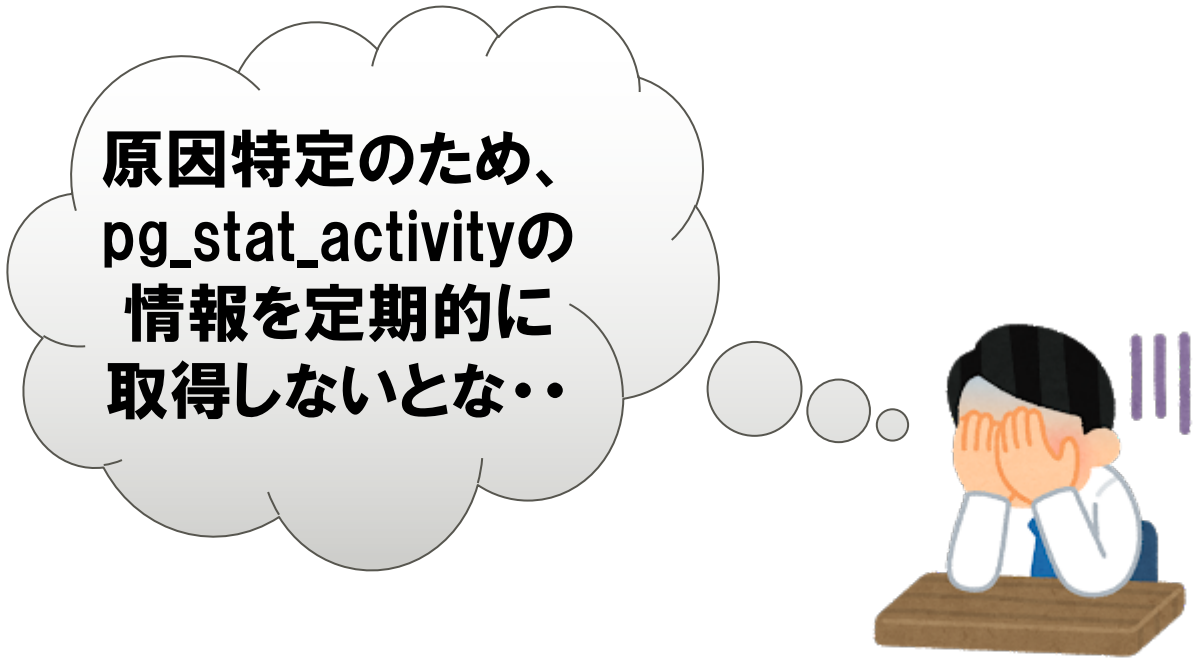
不定期に遅延が発生する事例での調査

同時間帯に該当SQLを阻害する処理が実行されていないかを調査

⇒ 調査方法として、
詳細なSQLの稼働状況はpg_stat_activityビューで確認

しかし・・・

- **遅延事象は不定期に発生**
- **短時間しか遅延事象が発生しない**



2.2. 不定期に遅延が発生する事例-3/6

機能・方式説明

FUJITSU

pg_stat_activity ビュー（ぴーじーすたっとあくてぃびてぃ）

SQLの実行状態など、現時点での情報を表示

- 各セッションの待機イベントの種類（wait_event_type）/待機イベント名（wait_event）を取得
- ある瞬間の情報（瞬間値）のため、影響の大きい待機イベントを調査するためには、定期的に情報を取得する方法（スケジュール実行）などを検討する必要がある。

```
=# SELECT wait_event_type, wait_event, state, query FROM pg_stat_activity;
 wait_event_type |      wait_event       | state  |                query
-----------------+-----------------------+--------+-------------------------------
 Extension       | Extension             |        |
 Extension       | Extension             |        |
 Activity        | AutoVacuumMain        |        |
 Activity        | LogicalLauncherMain   |        |
 Activity        | BgWriterHibernate     |        |
```

2.2. 不定期に遅延が発生する事例-4/6

機能・方式説明

FUJITSU

EDB Wait States 拡張モジュール（いーでーびー・うぇいと・すてーと）

EDB ENTERPRISEDB

- ✓ EDB Postgres Advanced Server 11よりサポート
- ✓ サンプリング間隔を最短1秒で設定可能（任意期間で履歴削除）
- ✓ セッションごとに、接続先のデータベース、ユーザ名、そのセッションで実行されているSQL、待機イベントなどの情報を収集
- ✓ 収集した各セッションの情報を関数から参照可能

```
=# SELECT query, session_id, wait_event_type, wait_event FROM edb_wait_states_data
('2019-05-31 16:18:15', '2019-05-31 16:18:25') WHERE session_id = '9139 ';
          query            | session_id | wait_event_type |  wait_event
---------------------------+------------+-----------------+--------------
 UPDATE pgbench_tellers ~ |       9139 | Lock            | transactionid
 UPDATE pgbench_tellers ~ |       9139 | Lock            | transactionid
 UPDATE pgbench_tellers ~ |       9139 | Lock            | tuple
 UPDATE pgbench_tellers ~ |       9139 | Lock            | transactionid
```

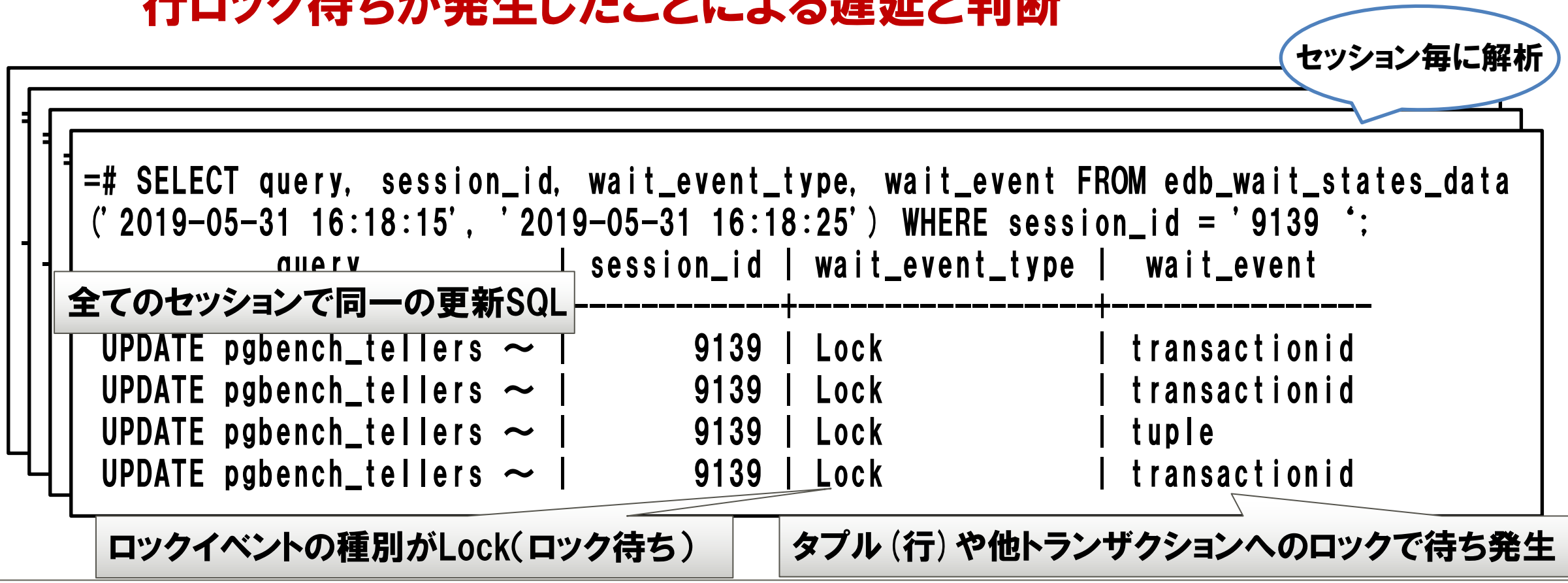
2.2. 不定期に遅延が発生する事例-5/6

課題解決案

FUJITSU

- EDB Wait Statesを用いて情報を取得。遅延したセッションの状態を調査した結果、該当の更新SQLでLockが発生していることを確認

⇒ 複数のセッションで更新処理が同時に実行され、特定の行で行ロック待ちが発生したことによる遅延と判断



2.2. 不定期に遅延が発生する事例-6/6

課題解決案

FUJITSU

性能問題改善方法

- 同一レコードの同時更新によって生じるロック待ちによる遅延
  - 最大接続数 (max_connections) の値を変更したことを起因とした可能性が高い
- アプリケーションサーバからの最大接続数を見直してもらい、同時に実行される更新SQLの数を制御することで、ロック待ち時間を軽減
- 設定変更を実施する際には性能試験などを実施するなど、必ず評価が必要
  - 同時に同一レコードを更新する処理方式にも問題あり

本番適用前に
性能評価を実施しておけば・・・
設計変更を検討できたのに・・・

2.3. 待機イベントの取得機能

機能比較

FUJITSU

■ 2.1,2.2で紹介した性能問題の調査・改善に役立つ機能

機能	PostgreSQL	EDB Postgres
待機イベント出力 (pg_stat_activityの参照)	○	
	- 稼働統計情報ビューの1つであるpg_stat_activityにおける待機イベント(wait_eventなど)を利用し、ボトルネックとなる処理を見つけるヒントを得る - 実行したタイミングでの瞬間値しか取得できない	
待機イベント情報保存と 出力(EDB Wait States)	-	○
	- 待機イベントの情報を定期的にファイルなどに出力する必要がある(crond/タスクスケジューラなど) - 集計処理を実装する必要がある	- セッション毎に発生した待機イベント情報を保存・出力できる
定期的な待機イベント 統計情報の保存と レポート出力 (DRITA)		○
		- 待機イベントの統計情報をスナップショットに保存できる (時系列で比較など可能) - 集計結果や分析に役立つ情報をレポートに出力できる

アジェンダ

- 当社の紹介

- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
     a. 想定した性能に達しない事例
     b. 不定期に遅延が発生する事例
     c. 特定のSQLで遅延が発生する事例
  3. Oracle互換機能でスムーズに移行しよう

- まとめ

2.4. 特定のSQLで遅延が発生する事例-1/6

課題

FUJITSU

■ 特定のSQLで遅延が発生する事例

■ 3秒以上遅延する場合のSQLはログに出力しており、特定のSQLで遅延が発生するようになった。

```
                                                    :
[XXXX-XX-XX XX:XX:XX.XXX JST] [XXXXX] [ [XXXXX] [enterprisedb] [OOOOO] [XXXX]
LOG:  duration: 6474.772 ms  statement: SELECT count ("事業所フラグ") FROM "全国住所"
WHERE "都道府県" = '東京都' AND "事業所フラグ" = 1;
                                                    :
[XXXX-XX-XX XX:XX:XX.XXX JST] [XXXXX] [XXXXX] [enterprisedb] [OOOOO] [XXXX]
LOG:  duration: 7067.933 ms  statement: SELECT count ("事業所フラグ") FROM "全国住所"
WHERE "都道府県" = '東京都' AND "事業所フラグ" = 1;
                                                    :
```

遅延が発生しているSQLでは、
“都道府県“が‘東京都’で“事業所フラグ”が 1 のレコード数を取得

特定のSQLで遅延が発生する事例

SQLで使用しているテーブル定義を確認

```
テーブル "public.全国住所"
     列       |  型   | 照合順序 | Null 値を許容 | デフォルト
--------------+-------+----------+---------------+------------
 住所CD       | text  |          |               |
 都道府県CD   | text  |          |               |
 市区町村CD   | text  |          |               |
 町域CD       | text  |          |               |
 郵便番号     | text  |          |               |
     :
  (省略)
     :
 事業所名     | text  |          |               |
 事業所名カナ | text  |          |               |
 事業所住所   | text  |          |               |
 新住所CD     | text  |          |               |
インデックス:
    "comb_idx" btree ("郵便番号", "都道府県", "市区町村", "事業所フラグ")
```

次の定義で複合インデックスを作成
("郵便番号","都道府県","市区町村","事業所フラグ")

2.4. 特定のSQLで遅延が発生する事例-3/6

調査・分析

FUJITSU

EXPLAINコマンドで実行計画を確認

定義された複合インデックスを操作

```
=# EXPLAIN [該当のSQL]
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=152949.67..152949.68 rows=1 width=8)
   ->  Bitmap Heap Scan on "全国住所"  (cost=105080.63..152853.94 rows=38292 width=2)
         Recheck Cond: (("都道府県" = '東京都'::text) AND ("事業所フラグ" = '1'::text))
         ->  Bitmap Index Scan on comb_idx  (cost=0.00..105071.06 rows=38292 width=0)
               Index Cond: (("都道府県" = '東京都'::text) AND ("事業所フラグ" = '1'::text))
```

複合インデックスを走査し、
絞り込み条件でのビットマップを作成
(処理コスト=105,071)

Bitmap Index Scanにて生成するビットマップの処理コストが大きい = 遅延原因

郵便番号	都道府県	市区町村	事業所フラグ
001-0000	北海道	札幌市北区	1
001-0010	北海道	札幌市北区	0
001-0011	北海道	札幌市北区	1
：			
100-0002	東京都	千代田区	1
	東京都	千代田区	1
	東京都	千代田区	0

複合インデックス内のデータ

必要のないレコードも走査して、
ビットマップを生成

0
0
0
1
1
0

ビットマップ

2.4. 特定のSQLで遅延が発生する事例-4/6

機能・方式説明

FUJITSU

EDB ENTERPRISEDB

Index Advisor（いんでっくす・あどばいざー）

- 指定されたSQLにおいて処理コストを削減するために、有効なインデックス（仮想インデックス）を提示
  - 仮想インデックスを利用した場合の実行計画と処理コストを合わせて表示
- 下記はIndex Advisorによる仮想Indexを作成した場合の実行計画を表示

```
 Result  (cost=0.00..0.00 rows=0 width=0)
   One-Time Filter: '===[ HYPOTHETICAL PLAN ]===' ::text
   ->  Aggregate  (cost=48598.09..48598.10 rows=1 width=8)
         ->  Bitmap Heap Scan on "全国住所"  (cost=729.05..48502.36 rows=38292 width=0)
               Recheck Cond: (("都道府県" = '東京都'::text) AND ("事業所フラグ" = '1'::text))
               ->  Bitmap Index Scan on "<hypothetical-index>:418"  (cost=0.00..719.48 rows=38292 width=0)
                     Index Cond: (("都道府県" = '東京都'::text) AND ("事業所フラグ" = '1'::text))
=# SELECT * FROM index_recommendations;
 backend_pid |                                                show_index_recommendations
-------------+-----------------------------------------------------------------------------------------------------------
        3575 | create index "idx_全国住所_都道府県_事業所フラグ" on public."全国住所"("都道府県", "事業所フラグ");/* size: 156408 KB, benefit: 208703, gain: 1.33435090436551 */
```

仮想インデックスを走査し、絞り込み条件でのビットマップを作成（処理コスト=719）

仮想Indexの作成SQLを表示

2.4. 特定のSQLで遅延が発生する事例-5/6

課題解決案

FUJITSU

性能問題改善方法

- 本SQLで使用されていた複合Indexが、検索条件の絞り込みに使用するには、適切でなかったことが原因
- 実行計画でIndex Scanが選択されている場合でも、cost値を確認し、Indexが適切かを判断（Index Advisor活用が有効）
- 絞り込み条件に合致したIndexを作成することで性能は改善

```
テーブル "public.全国住所"
      列       |  型   | 照合順序 | Null 値を許容 | デフォルト
---------------+------+----------+---------------+------------
 住所CD        | text |          |               |
     :
  (省略)
     :
新住所CD       | text |          |               |
インデックス:
    "comb_idx" btree ("郵便番号", "都道府県", "市区町村", "事業所フラグ")
    "idx_全国住所_都道府県_事業所フラグ" btree ("都道府県", "事業所フラグ")
```

SQLの実行時間を大幅に短縮
インデックス定義前：6秒程度
インデックス定義後: 0.1秒

2.4. 特定のSQLで遅延が発生する事例-6/6

機能・方式説明

FUJITSU

ご参考. Optimizer Hints (おぷてぃまいざ・ひんと)

EDB ENTERPRISEDB

- SQL内にキーワード(コメント)を定義することでクエリプランナが選択するクエリプランに影響を与える
- 特定のSQLだけの**個別チューニング**や**一時的な問題回避方法**として活用

```
// 実行計画で意図的に従来のIndexを使用するように指定
=# EXPLAIN SELECT /*+ INDEX("全国住所" "comb_idx") */ count("事業所フラグ") FROM "全国住所"
WHERE "都道府県" = '東京都' AND "事業所フラグ" = 1;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------
 Aggregate  (cost=152949.67..152949.68 rows=1 width=8)
   ->  Bitmap Heap Scan on "全国住所"  (cost=105080.63..152853.94 rows=38292 width=2)
         Recheck Cond: (("都道府県" = '東京都'::text) AND ("事業所フラグ" = '1'::text))
         ->  Bitmap Index Scan on comb_idx  (cost=0.00..105071.06 rows=38292 width=0)
               Index Cond: (("都道府県" = '東京都'::text) AND ("事業所フラグ" = '1'::text))
```

- 注) Aliasで別名を定義したテーブルを使用する場合は、Alias名を指定

2.5. インデックスや実行計画変更機能

機能比較

FUJITSU

■ 本節で紹介した性能問題の調査・改善に役立つ機能

機能	PostgreSQL	EDB Postgres
性能改善のための インデックス定義 (Index Advisor)	△	○
	- 実行計画を読み解き、 改善方法を検討する必要がある	- 処理コストを軽減するための、 仮のインデックスを提示
実行計画の選択 (Optimizer Hints)	△	○
	- 一部の処理方法を無効化するなどの、 大まかな制御は可能 - 詳細な制御を行う場合は、 外部モジュールの利用が必要	- SQLにヒントコメントを定義する ことでクエリプランナが選択する 実行計画を制御可能

アジェンダ

- 当社の紹介
- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
  3. Oracle互換機能でスムーズに移行しよう
- まとめ

3. Oracle互換機能でスムーズに移行しよう-1/5

機能・方式説明

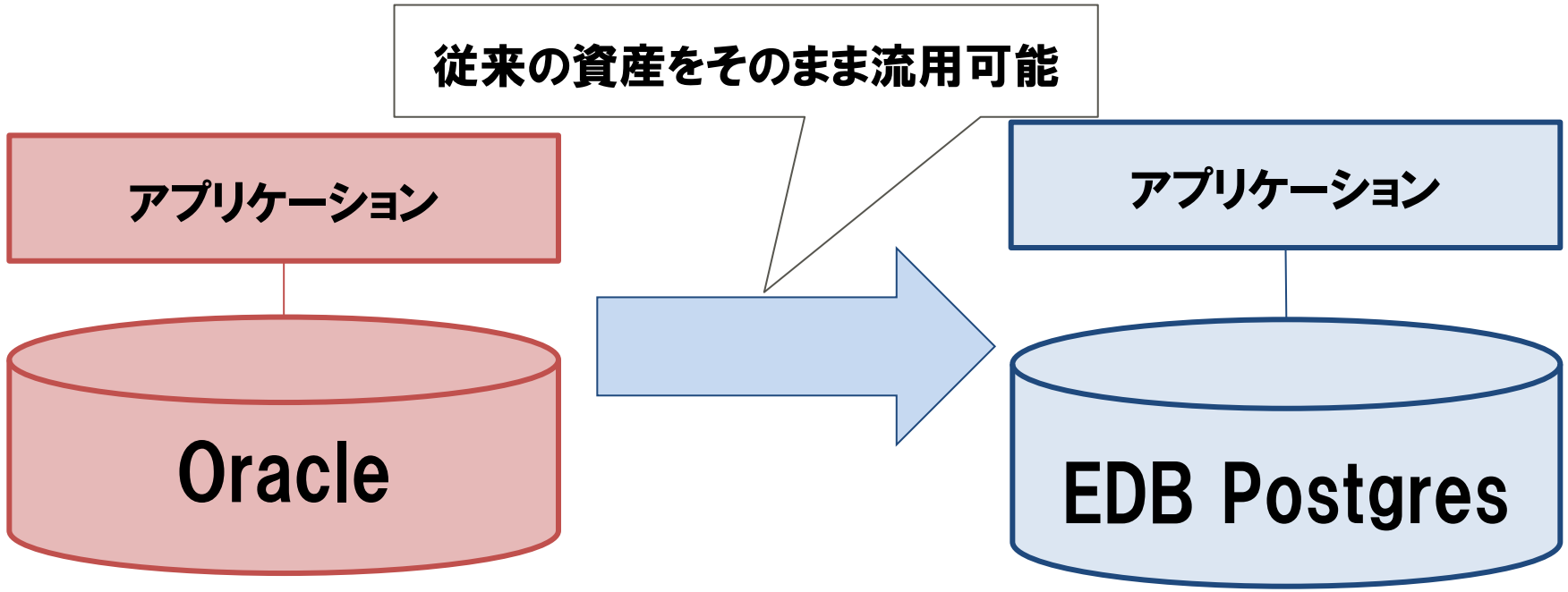
FUJITSU

EDB ENTERPRISEDB

Oracle互換機能

- Oracleのスキーマ、データおよびOracle向けに開発された
  アプリケーションを書き換え不要でEDB Postgres上で動作させる機能
- Oracleを利用しているエンジニアは、
  Oracleで学んだ構文などをEDB Postgresにおいても活用できる

※ Oracle互換機能はEDB Postgres Enterprise Editionに搭載



3. Oracle互換機能でスムーズに移行しよう-2/5

機能・方式説明

FUJITSU

EDB ENTERPRISEDB

Oracle互換機能の実力

- 一般的なOracleの資産なら、およそ90%以上が修正せずに移行が可能
- Oracleからの移行には、互換機能が実装されたEDB Postgresが最適

	Oracle 10g, 11g		EDB Postgres
事例 A社	テーブル数:800 テーブル	自動移行	96 %
		手動移行	4 %
		一部のOracle独自機能、複雑なビューの一部については手動移行	
事例 B社	テーブル数:3,200 テーブル	自動移行	99 %
		手動移行	1 %
		Oracle独自のセキュリティ(プロファイル)機能については手動移行	

3. Oracle互換機能でスムーズに移行しよう-3/5

機能・方式説明

FUJITSU

■ Oracle互換機能（一部）

EDB ENTERPRISEDB

項番	互換機能	説明
1	Oracle 独自のSQL構文	(+) 句でのテーブル結合やパーティションテーブル等のOracle独自構文をサポート
2	Oracle 独自の関数	NVL関数、DECODE関数等の多くのOracle独自関数をサポート
3	Oracleのオブジェクト	シノニム、パッケージ等をサポート
4	Oracle PL/SQL互換のプロシージャ言語	PL/SQLと互換のあるEDB-SPL （PL/SQLで実装されたプロシージャ等を移行可能）
5	Oracle組み込みパッケージ	利用頻度が高いOracleの組み込みパッケージに対応
6	互換ツール	EDB*Plus、EDB*Loader等
7	接続ドライバ/ライブラリ	Oracleの各種ライブラリ/ドライバとの互換機能
8	スキーマ移行ツール	Migration Toolkit （OracleのスキーマをEDB Postgresに投入可能な形式に変換）
9	データ連携ツール	データベースリンク、EDB Replication Server （Oracleとのデータレプリケーション）

3. Oracle互換機能でスムーズに移行しよう-3/5

機能・方式説明

FUJITSU

Oracle互換機能（一部）

EDB ENTERPRISEDB

Oracleとの互換機能は非常に充実している！！

3. Oracle互換機能でスムーズに移行しよう-4/5

FUJITSU

■ データベース移行事例：特定非営利活動法人 薬学共用試験センター様

コスト削減、システムのオープン化のため、
74大学で稼働するOracle DatabaseをEDB Postgresへスムーズに移行

■ データベース移行をスムーズに実施するための3つのポイント

1. 入念な事前検証を元にトラブルなしで移行完了
2. Oracle互換機能の活用で修正箇所はごく一部（大幅な改修の必要なし）
3. 迅速なテスト実施により、2か月の短期間で移行完了。
（性能試験にて、性能改善も確認。運用開始後も安定稼働）

3. Oracle互換機能でスムーズに移行しよう-5/5

■ データベース移行事例

■ 薬学共用試験センター様に喜んでいただけて、事例とさせていただきました。

お手元のパンフレット
をご確認下さい。

FUJITSU Enterprise Application データベース移行ソリューション

FUJITSU

非営利活動法人 薬学共用試験センター様

大学の中継サーバをOSSデータベースに移行し、
システムの安定性維持とコスト削減を実現

特定非営利活動法人 薬学共用試験センター様は、薬学共用試験の一つであるCBTシステムのオープンシステム化を目指し、74大学すべての中継サーバをオープンソース・ソフトウェア（OSS）ベースのデータベース「EDB Postgres」に移行。入念な事前検証により短期間で移行を完了し、システムの安定性維持とコスト削減を実現しました。

導入背景
- システムの安定性を維持しながら汎用的なオープンシステムに移行したい
- 限られた期間内で移行を完了したい
- 年間のランニングコストを削減したい

導入効果
- OSSの性能や既存システムとの親和性などの評価を行い「EDB Postgres」への移行を実施。安定性の維持を実現
- 入念な事前検証と移行計画により、74大学すべての中継サーバを2か月で移行完了
- 毎年発生するライセンス費用を削減し、ランニングコストを大幅に縮小

導入の背景

公平性の高いオープンシステムを目指し、初期導入から5年でOSSへの移行に着手

特定非営利活動法人 薬学共用試験センター様（以下：薬学共用試験センター様）は、臨床にかかわる実践的な能力を有する薬剤師を養成する取り組みの一環として2006年に設立され、薬学部学生向けの薬学共用試験を運営しています。薬学共用試験は、実務実習前に、学生が全国共通の統一基準に適しているかを判定するためのもので、全国74校の学生約1万人が毎年受験しています。その一つ「CBT（Computer-Based Testing）」は、オンラインで基本知識を問う客観試験で、薬学共用試験センター様のセンターサーバと受験会場である各大学の中継サーバをネットワークでつなぎ実施されています。2009年の稼働開始以来、問題作成から各大学への配布、受験や採点、評価、分析に至るまで、迅速かつ的確な試験を実施してきました。

CBTシステムの運用責任者である薬学共用試験センター 宮崎俊氏（東京理科大学 薬学部 教授）は今回のリプレイスについてこう語ります。「システムは公平性の高いものが望ましく、導入当初よりオープンシステムを目指していましたが、初期構築期間が短かったためまずは安定性を優先し、全体最適の面からOracle Databaseで構築しました。導入から5年が経過し、システムの安定性や性能に問題ないことがわかったため、今回のリプレイスを機にオープンシステム化を進めることにしました。」

導入の経緯

限られた移行期間で安定稼働を実現するため、費用対効果の高い「EDB Postgres」を選択

CBTシステムのOSSデータベース移行にあたり、薬学共用試験センター様が最も重要視したのは安定稼働です。スケジュール通りにトラブルなく試験を実施するため、従来と同等の安定性が必須要件とされていました。また、年1回の体験試験と年2回の本試験（追再試験を含む）、およびその準備期間を除く2か月という短期間で移行を完了させる必要がありました。

薬学共用試験センター様内でOSSに対する不安の声も上がるなか、当社での実績も多くサポート体制の整った商用版「EDB Postgres」への移行を提案。当時を振り返り宮崎氏は語ります。「今回のリプレイスはベンダーロックを解除してオープンシステムにすることが一番の目的でした。検討段階ではフリーのOSSも視野に入れましたが、OSSに移行したことで安定性が失われては意味がありません。連携する他システムとの整合性やセキュリティの確保も重要なため、サポート費用が発生しても商用版（OSS）の方が費用対効果が高いと判断し「EDB Postgres」を選択しました。」

shaping tomorrow with you
社会とお客様の豊かな未来のために

www.fujitsu.com/jp/group/ssl/services/application_services/enterprise-applications/oss/db_kio/

事例カタログ FUJITSU Enterprise Application データベース移行ソリューション

導入の効果

入念な事前検証により移行費用を抑制し、さらに、今後のランニングコストも大幅に削減

OSSの導入にあたり、まずは、1万人分の負荷がかかるセンターサーバより、各大学に配置され負荷が分散される中継サーバについて検証を行いました。Oracleでしか使えないSQLはどれくらいあるか、またユーザーインターフェイスのJavaとの相性など、簡単操作を使い細かくチェックした結果、連携する既存システムとの親和性や従来と同等のシステム性能を確認。さらに期間内での移行と移行コストの低減やランニングコストの削減が見込まれたことで「EDB Postgres」を活用したリプレイスに踏み切り、2015年4月に移行を開始しました。

当社プラットフォームインテグレーション本部 第四システム部 担当部長 関井 泰人は語ります。「安定稼働と短期間での移行を実現するためには十事な事前検証が必要不可欠です。また今回、Oracle Databaseと互換性が高く、同等のパフォーマンスを持つ「EDB Postgres」を活用することで、独自開発した既存アプリケーションの修正を最小限に抑えると共に、データベース処理に伴う画面操作時のレスポンス時間やバッチの処理時間についても、移行前と同等以上のレベルを保持することができました。さらに専用の移行ツールを用い、テーブルやインデックスなどできる限り移行を自動化することで作業工数を抑制しました。」

事前検証と移行計画の立案を入念に行ったことで、全国74校に配置しているすべての中継サーバを2か月という限られた期間でスムーズに移行完了。移行後の2015年6月から約10か月間で、1回の体験試験と2回の本試験を実施し、オープンシステムでの安定稼働を実証しました。また、当社の「EDB Postgres」のノウハウを活かし作業工数を抑えたことで、低コストでの移行を実現。ランニングコストも、Oracle Databaseより安価な「EDB Postgres」のサブスクリプション費用で済むため、大幅なコスト削減を実現しました。

<システム構成イメージ>

※記載の会社名、商品名は、各社の商標または登録商標です。
※記載された情報は、予告なく変更することがあります。
※記載の内容は、2016年2月現在のものです。

今後の展望

変化に対応できる汎用性の高いオープンシステムを作り、それを継承していくことがミッション

薬学共用試験センター様はCBTシステムをより汎用性の高いものとするため、中継サーバを取りまとめ全体の管理を行う、システムの中枢であるセンターサーバについても、今後「EDB Postgres」への移行を検討し、さらなるオープンシステム化を進めていく計画です。

宮崎氏は今後の展望について述べます。「2018年度、CBTは薬学教育モデル・コアカリキュラム改訂による出題形式の大幅な変更が予定されており、内容でも多くの対応作業が見込まれています。同時に次期リプレイスの検証を進めていくため、システムがトラブルなく安定的に稼働していることが大切です。私はこのCBTシステムをオープンシステム化し、試験体系の変更にも迅速かつ柔軟に対応できる汎用性の高いシステムを作り、それを継承していくことがミッションであり、今回の中継サーバのリプレイスでその足掛りを作ることができました。」

今後当社は、スマートフォンやタブレット端末などマルチデバイスへの対応について薬学共用試験センター様と検討を進め、大学や学生に向け一層充実した安全性の高いCBTシステムを提供していく考えです。

また当社は、長年培ってきた高い技術力と実績から得たノウハウを活かし、様々なお客様に向け、OSSを活用した効果的で利便性の高いシステムのご提案から構築、日々の運用までトータルでご支援していきます。

センターの概要

特定非営利活動法人 薬学共用試験センター

【所在地】 〒150-0002 東京都渋谷区渋谷2丁目12-15
【事業内容】
・薬学系大学における学生の学習到達度を判定するための共用試験の実施および評価に関する事業
・共用試験全般の管理に関する事業
・共用試験の改善に関する事業 など
【設立】 2006年10月
【Webサイト】 http://www.phcat-ol.jp/

お問い合わせ先
株式会社富士通ソーシアルサイエンスラボラトリ（富士通SSL）
お問い合わせ総合窓口
〒211-0063 川崎市中原区小杉町1-403武蔵小杉タワープレイス
E-mail：ssl-info@cs.jp.fujitsu.com
当社ホームページ http://www.fujitsu.com/jp/group/ssl/

Copyright 2016 FUJITSU SOCIAL SCIENCE LABORATORY LIMITED

SSL_case_008

アジェンダ

- 当社の紹介
- DBMSの課題に振り回されないためのEDB Postgres活用術
  1. バージョンアップをスムーズに実施しよう
  2. 性能問題を解決しよう
  3. Oracle互換機能でスムーズに移行しよう
- まとめ

まとめ

FUJITSU

1. バージョンアップをスムーズに実施しよう

- 停止時間や構成を考慮したバージョンアップについて紹介

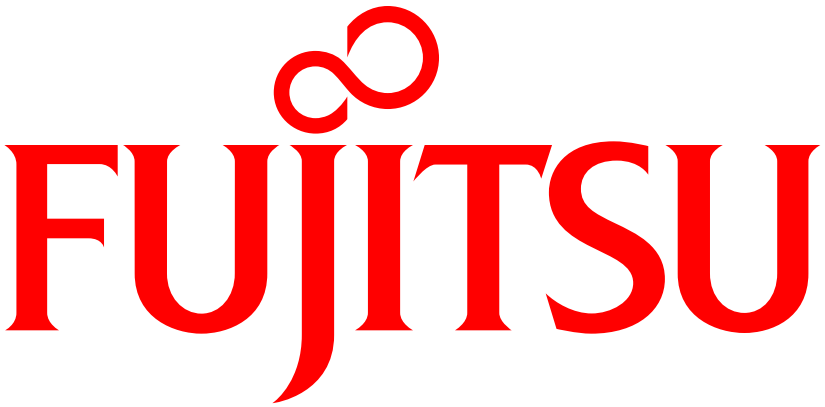
2. 性能問題を解決しよう

- 待機イベントの情報やindex Advisorなどを用いた課題解決方法を紹介
- 性能問題を起こさないため、設定変更後などは性能確認が必要

3. Oracle互換機能でスムーズに移行しよう

- 本日は簡単な紹介のみとしたため、
  ご興味があれば展示ブースにお立寄り下さい。

本日紹介したような課題が御座いましたら、
弊社にご相談下さい。



shaping tomorrow with you