

ORACLE®

# データベース基盤のゼロ・ダウン タイム・メンテナンスの実装方式

日本オラクル株式会社  
テクノロジー製品事業統括本部 基盤技術本部  
シニアエンジニア  
佐々木亨



 #odddtky

日本オラクル、今年最大の技術トレーニング・イベント

**Oracle DBA &  
Developer Day 2013**

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント(確約)するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

OracleとJavaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

# Program Agenda

- ゼロ・ダウンタイム・メンテナンス実現の課題
- 実装方式と特徴
- まとめ

# ゼロ・ダウンタイム・メンテナンス 実現の課題

# メンテナンスについて考えるべきこと

「リスク」、「ダウンタイム」、「複雑さ」

- 一番考慮すべきなのは「リスク」
  - 作業ミスによるデータロスなど
  - リスクと複雑さは関連がある(シンプルな手順を使えば、リスクは小さくなる)
- 一般的には、ダウンタイムを短くするには、作業は複雑になり、その結果リスクは高くなる
- しかし、多くの場合「ダウンタイムの長さ」が最重要視される
  - 「ダウンタイムを短く」、「リスクは小さく」→ DBAの方々の大きな負担

# メンテナンスの種類

本セッションの主な対象はDBアップグレード

- システム変更
  - H/W Firmware パッチ適用
  - OS アップグレード、パッチ適用
  - DB アップグレード、パッチ適用
- データ変更
  - テーブル再定義、アプリケーション・アップグレード
- アプリケーション変更
  - アプリケーション・アップグレード

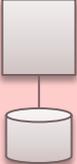
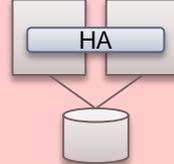
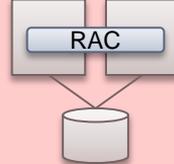
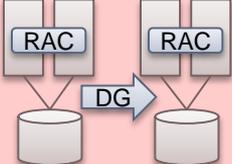


# パッチの種類とリリース頻度

種類名称	バージョン表記例	リリース頻度	説明
Patch Set Release (PSR)	11.2.0.3.0 → 11.2.0.4.0	1~2年	メジャー・リリースの間で作成され、複数の不具合に対する修正を統合したもの。ベース・リリースと過去のパッチ・セットに対して累積的なパッチ。
Exadata Bundle Patch (BP)	11.2.0.3.20 → 11.2.0.3.21	1ヶ月 (3カ月)	Exadata向けに、PSUを含んだ重要な不具合の修正を統合したもの。PSRに対する累積パッチ。
Patch Set Update (PSU)	11.2.0.3.1 → 11.2.0.3.2	3ヶ月	最も重要な修正を含んだ4半期ごとに提供されるパッチ
Security Patch Update (SPU)	N/A	3ヶ月	セキュリティ修正で構成される累積的なパッチ
Patch Set Exception (PSE) (個別パッチ)	N/A	適時	ある不具合を修正するために、次のPSU/PSRや新製品のリリースまで待つことができないお客様のために作成される1つ以上の修正を含むパッチ。

# パッチの種類と適用方法

## 計画停止時間の目安 (Oracle Database 11g Release 2)

Target	Patch Type	RAC Rolling	Single	HA	RAC	RAC + DG
						
Database	BP/PSU/CPU	Yes	DB停止後適用 (数分～数十分)	F/Oで交互適用 (数分 x 2回)	RACローリング 適用(ゼロ)	
	PSR	<b>No</b>	DB停止後適用 (数十分～数時間)			<b>S/Oで交互適用 (5分未満 x 2回)</b>
Grid Infrastructure (OCW/ASM)	BP/PSU/CPU	Yes	DB停止後適用 (数分～数十分)	F/Oで交互適用 (数分 x 2回)	RACローリング 適用(ゼロ)	
	PSR	Yes	DB停止後適用 (数十分)	F/Oで交互適用 (数分 x 2回)	RACローリング 適用(ゼロ)	
OS	-	Yes	DB停止後適用 (数分～数時間)	F/Oで交互適用 (数分 x 2回)	ローリング適用 (ゼロ)	

※ RACローリングの可否について: 2013/08末時点でリリースされたBPは全てRACローリング可

# ゼロ・ダウンタイム・メンテナンス実現への課題

Data Guardのスイッチ・オーバー時に停止時間が発生

- 一時的に、両データベースが「Standbyロール」になる
  - 一時的に(数分間)、更新処理が実行不可
  - この時間を短くする、ゼロにすることが課題解決への道
- アプリの接続先データベースが変更
  - 新プライマリがオープン後に、**コネクション再作成の時間が必要**  
(ロジカル・スタンバイではロール遷移時にコネクションは切断されない)
  - この切替を早くすることが課題解決への道

# ゼロ・ダウンタイム・メンテナンス実現への課題

## GoldenGateで解決できるか？

- 「更新処理が実行不可」という課題は解決できる
  - GoldenGateには「ロール」という概念がなく、両方のデータベースをActive-Active構成で組めるため、スイッチ・オーバーは不要
- 一方で、Data Guard の魅力も思い出しておきましょう

# Data GuardとGoldenGate

## 独自機能の比較

### Data Guard (フィジカル・スタンバイ)

- 同期転送
- データ破損対策
  - ✓ **自動ブロック修復**
  - ✓ **DB\_LOST\_WRITE\_PROTECT**
- スタンバイで取得したバックアップをプライマリにリストア可能
- 自動フェイルオーバー
- データ型の制限がない

### GoldenGate

- **Active-Active**  
(**両データベースで書込み可**)
- 異OS/異バージョンでのレプリケーションが可能
- テーブル単位でのレプリケーション
- コンソリ
- データをフィルタ、変換してのレプリケーション

# ゼロ・ダウンタイム・メンテナンスの実現手法

## PSR適用における Pros & Cons

#	手法	停止時間の目安	スイッチオーバー時間		通常運用時		PSR適用時
			ロール変換	接続性	モード	データ保護	モード
1	Physical Standby	数時間	不要	再接続時間が必要	DG (Physical)	ABMR Lost Write	DG (Physical)
2	Transient Logical Standby	5分未満 x 2回	必要	S/O時 : 再接続不要 S/B時 : 再接続必要	DG (Physical)	ABMR Lost Write	DG (Logical)
3	Multi-Physical Standby	ゼロ (Read Only期間有り)	必要	一時的にADGへ接続 →待機無し	DG (Physical)	ABMR Lost Write	DG (Physical)
4	GoldenGate	ゼロ	不要	Active-Active構成 →待機無し	GG	N/A	GG
5	Transient GoldenGate	ゼロ	不要	Active-Active構成 →待機無し	DG (Physical)	ABMR Lost Write	GG

ABMR : Auto Block Media Recovery

ORACLE

# 実現方法と特徴

# ゼロ・ダウンタイム・メンテナンス実現する 本セッションでご紹介するアップグレード方式

- 基本手法
- 一時ロジカル・スタンバイ (Transient Logical Standby) を用いた手法
- 複数のフィジカル・スタンバイ・データベース (ADG) を用いた手法
- 一時GoldenGate (Transient GoldenGate) を用いた手法

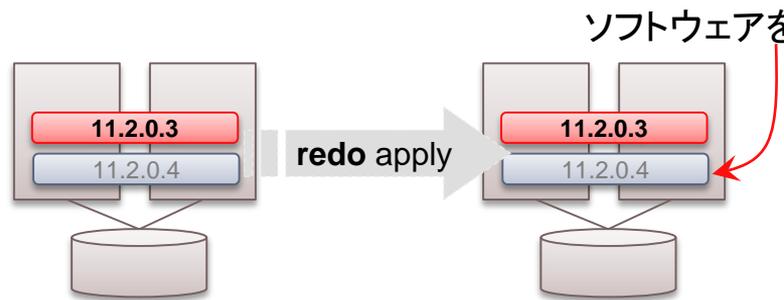
# 基本手法

# 特徴

## 基本手法

- 作業がシンプル
- アップグレード全体としての作業時間が短い
- アプリケーションを止めて作業を実施するので、アップグレード中の障害に対してもデータロスの心配がない
- アプリケーションの切替手順を考える必要がない

# 手順概要



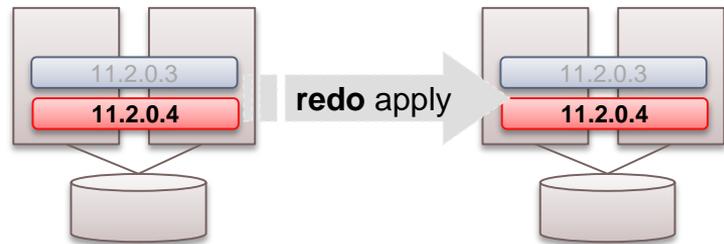
[1.]  
GIのUpgrade & DBソフトウェアのインストール  
(APは動かしたまま実施可能)

[2.]  
インスタンス停止、ORACLE\_HOMEの切替

[3.]  
プライマリでDBUAを実行

[4.]  
アップグレード完了後、AP再開

DBUA  
アップグレード



# アウトオブプレース・アップグレード

11.2.0.2～

- 別のORACLE\_HOME に新ソフトウェアをインストール
- インストール後に、新ORACLE\_HOME からデータベースをマウントする
- 切替直前まで旧ORACLE\_HOME で稼働可能
  - ※ *Grid Infrastructure* のアップグレードが終わらないと、新しいDatabaseソフトウェアのインストールはできない点には注意が必要
- プライマリで流したcatupgrd.sqlなどは、REDO Apply でスタンバイにも反映されるので、スイッチオーバーは不要

# 一時ロジカル・スタンバイ (Transient Logical Standby) を用いた手法

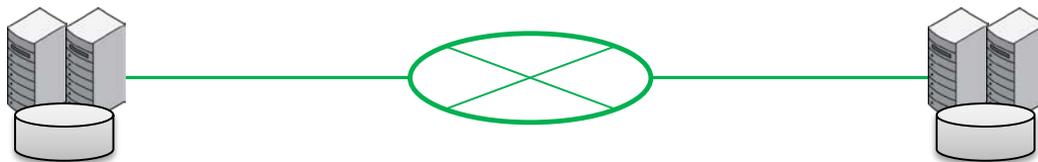
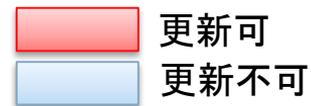
# 特徴

## 一時ロジカル・スタンバイを用いた手法

- フィジカル・スタンバイ・データベースが1つあれば実現できる
- 多くのベーシックな機能を駆使して実現(複雑さが増す)
- 数分レベルのアプリケーションの停止時間でアップグレード可能
  - 停止時間が発生するのは、スイッチオーバー処理時のみ(1回もしくは2回)
  - 一連のアプリケーション作業を終えるには数時間必要
- 一時的にロジカル・スタンバイ・データベースを使用する
  - データ型の制約を受ける
- REDOを転送できない時間帯が発生(データロスのリスクが増す)

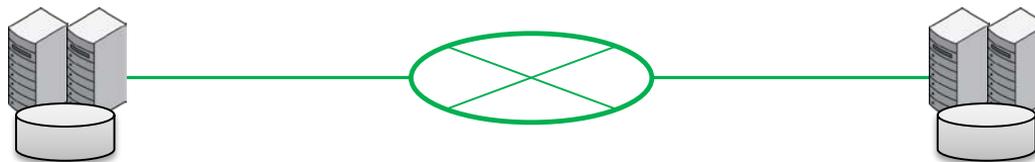


# 手順

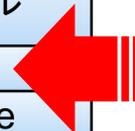


#	Version		REDO転送		Version
①	11.2.0.3	•ORACLE_HOMEインストール •保証付きリストアポイント取得	→	ORACLE_HOMEインストール	11.2.0.3
②	11.2.0.3		→	ロジカル・スタンバイに変換	11.2.0.3
③	11.2.0.3	REDO転送を停止する	-	DBUAもしくは手動でUpgrade	11.2.0.4
④	11.2.0.3	REDO転送再開	→	REDO適用開始	11.2.0.4
⑤	11.2.0.3			スイッチオーバー	11.2.0.4
⑥	11.2.0.3	DBを①時点にフラッシュバック	-		11.2.0.4
⑦	11.2.0.4	新ORACLE_HOMEに切替	-		11.2.0.4
⑧	11.2.0.4	フィジカルスタンバイに変換	-		11.2.0.4
⑨	11.2.0.4	REDO同期開始 (REDO経由でcatupgrd反映)	←		11.2.0.4
⑩	11.2.0.4			スイッチオーバー	11.2.0.4
⑪	11.2.0.4		→		11.2.0.4

# 手順



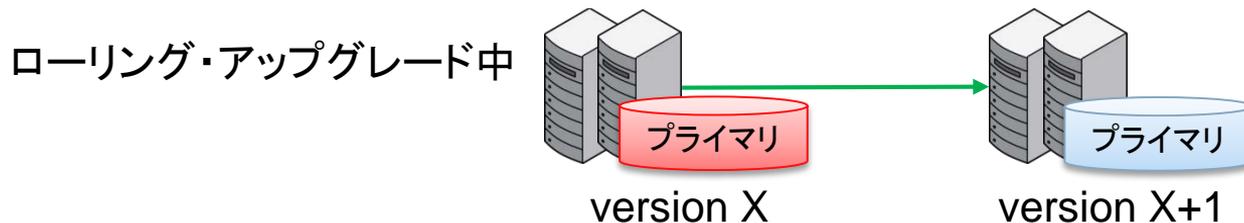
	REDO転送	
•ORACLE_HOMEインストール •保証付きリストアポイント取得	→	ORACLE_HOMEインストール
	→	ロジカル・スタンバイに変換
REDO転送を停止する	-	DBUAもしくは手動でUpgrade
REDO転送再開	→	REDO適用開始
スイッチオーバー		
DBを①時点にフラッシュバック	-	
新ORACLE_HOMEに切替	-	
フィジカルスタンバイに変換	-	
REDO同期開始 (REDO経由でcatupgrd反映)	←	
スイッチオーバー		
	→	



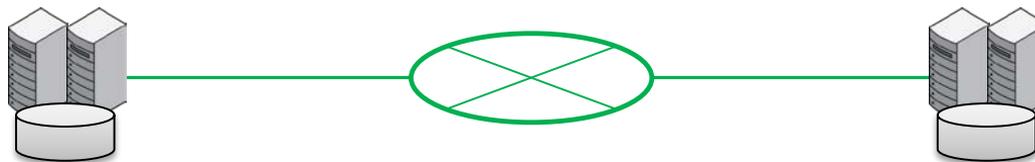
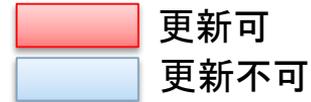
# パッチレベルに関する原則

Data Guard プライマリとスタンバイのパッチレベルは同一

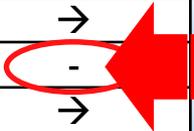
- Data Guard 構成では、プライマリ、スタンバイのパッチレベル (PSEレベルまで) が同一であることが原則求められる
- ローリング・アップグレード中における、ロジカル・スタンバイについては、「プライマリDBのパッチレベル < スタンバイDBのパッチレベル」状態での起動および、REDO転送・適用が認められている
  - 他の例外は、*Standby-First Patch Apply*



# 手順



	REDO転送	
•ORACLE_HOMEインストール •保証付きリストアポイント取得	→	ORACLE_HOMEインストール
	→	ロジカル・スタンバイに変換
REDO転送を停止する	-	DBUAもしくは手動でUpgrade
REDO転送再開	→	REDO適用開始
スイッチオーバー		
DBを①時点にフラッシュバック	-	
新ORACLE_HOMEに切替	-	
フィジカルスタンバイに変換	-	
REDO同期開始 (REDO経由でcatupgrd反映)	←	
スイッチオーバー		
	→	



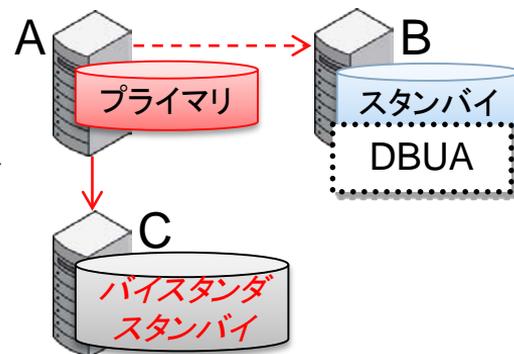
# パッチ適用中のデータ保護

## 複数スタンバイ構成

- ロジカル・スタンバイ・データベースにおいて、DBUAを使ってアップグレードをしている間はREDOを受信することができない
- この間もプライマリでアプリケーションを動かす場合、REDOの転送先がないため、通常運用時と比べてデータ保護のレベルが下がる

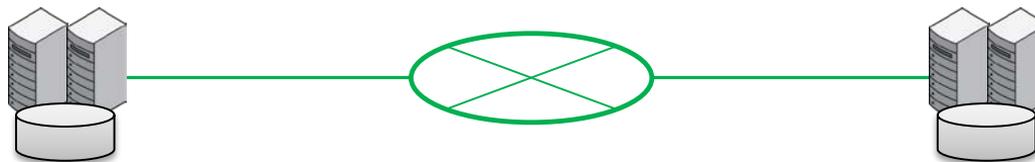
### → 策: 別のスタンバイ・データベースを用意

- スイッチオーバーには関与させない
- A $\leftrightarrow$ Bのスイッチオーバー後、Aをフラッシュバックした時点でCはAのスタンバイではなくなる
  - Cでも保証付きリストアポイントを取得しておく



# 手順

更新可  
更新不可



	REDO転送	
•ORACLE_HOMEインストール •保証付きリストアポイント取得	→	ORACLE_HOMEインストール
	→	ロジカル・スタンバイに変換
REDO転送を停止する	-	DBUAもしくは手動でUpgrade
REDO転送再開	→	REDO適用開始
スイッチオーバー		
DBを①時点にフラッシュバック	-	↙ ↘
新ORACLE_HOMEに切替	-	
フィジカルスタンバイに変換	-	
REDO同期開始 (REDO経由でcatupgrd反映)	←	↙ ↘
スイッチオーバー		
	→	

# スイッチオーバーの回数

APサーバーの配置で必要なスイッチオーバーの回数は異なる

## ■ 図1

- 両データセンターにAPサーバーが存在
- スwitchオーバーは1度で良い

## ■ 図2

- 片側のデータセンターにのみAPサーバーが存在
- スwitchオーバーを2度実施し、DCをまたがる接続とならないようにするのが望ましい

図1

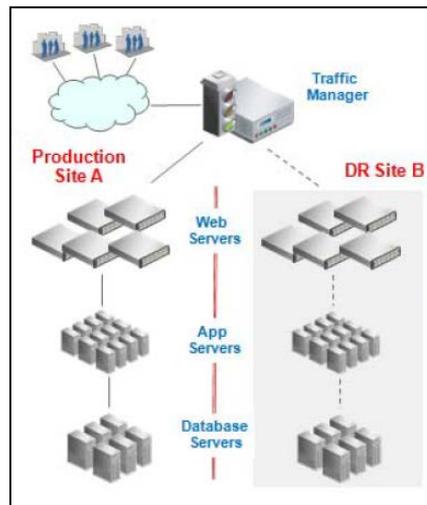
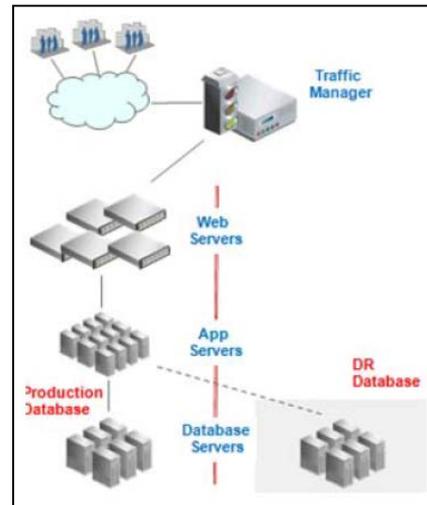


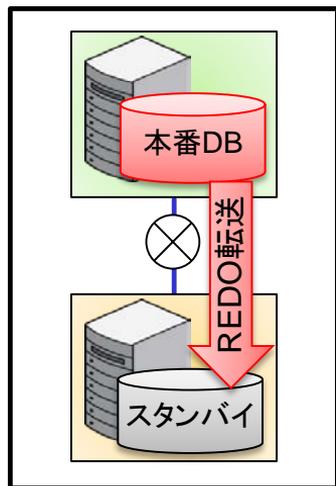
図2



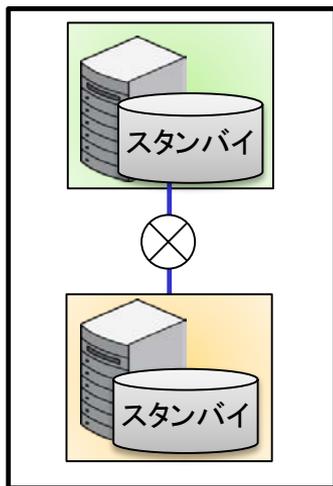
# スイッチオーバー時間の変動要素

## フィジカル・スタンバイにおけるスイッチオーバー

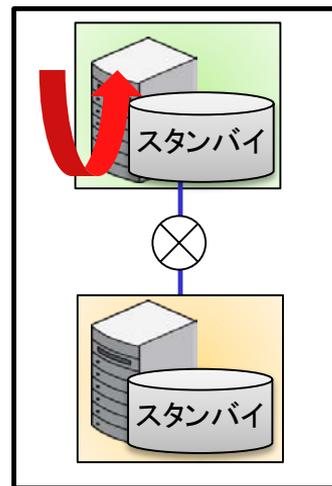
1)  
正常稼働



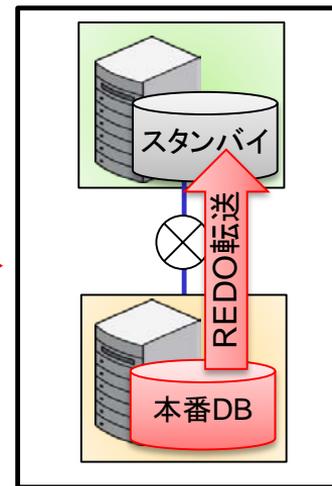
2)  
• REDO適用、ログスイッチ 新スタンバイを再起動 (チェックポイント)  
• ロール変換 (P→S)  
• 現プライマリへの既存セッション切断



3)  
• ロールの遷移 (S→P)  
• 旧スタンバイへの既存セッション切断  
• 適用開始→新スタンバイのonline redo log(ORL)初期化



4)  
• ロールの遷移 (S→P)  
• 旧スタンバイへの既存セッション切断  
• 適用開始→新スタンバイのonline redo log(ORL)初期化



# スイッチオーバー時間の変動要素

## フィジカル・スタンバイにおけるスイッチオーバー

- スwitchオーバー実施時点のプライマリとスタンバイのタイムラグ
  - 未適用のREDOを適用したのちにロール変換が行われる
    - 適用が追い付いている状態でスイッチオーバーを開始する  
(`v$dataguard_stats` で確認)
- ログスイッチが発生しスタンバイで checkpoint が発生する
  - スwitchオーバーを始める前にプライマリでログスイッチしておく  
(スタンバイでのcheckpoint は、プライマリでログスイッチした場合のみ)

# スイッチオーバー時間の変動要素

## フィジカル・スタンバイにおけるスイッチオーバー

- 既存セッションを切断するのに要する時間

<Alert.log の内容>

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WAIT WITH SESSION SHUTDOWN
ALTER DATABASE SWITCHOVER TO PRIMARY (stb11)
Maximum wait for role transition is 15 minutes.
All dispatchers and shared servers shutdown
CLOSE: killing server sessions.
Active process 10978 user 'oracle' program 'oracle@stb11 (TNS V1-V3) '
Active process 12652 user 'grid' program 'oracle@stb11 '
<snip>
Active process 10978 user 'oracle' program 'ora
Active process 11634 user 'grid' program 'ora
CLOSE: all sessions shutdown successfully.
<snip>
Switchover: Complete - Database mounted as pri
Completed: ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WAIT WITH SESSION SHUTDOWN
```

スタンバイに接続しているセッションを  
シリアルに切断する  
→ 不要セッションは事前に切断しておく

# スイッチオーバー時間の変動要素

## フィジカル・スタンバイにおけるスイッチオーバー

- データベースの再起動にかかる時間(フィジカル・スタンバイ時のみ)
  - プライマリからスタンバイへのロール遷移時に、OPEN → MOUNT になる
  - この MOUNT 状態は通常の MOUNT 状態ではないため、  
一旦 SHUTDOWN → MOUNT(or READ ONLY OPEN)する必要がある

# スイッチオーバー時間の変動要素

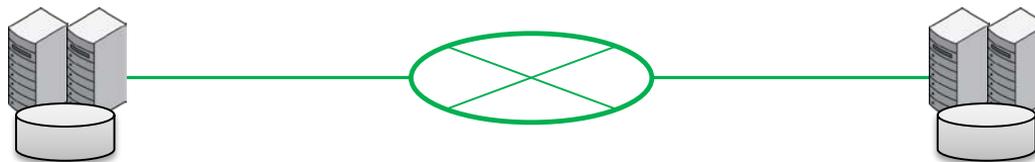
## フィジカル・スタンバイにおけるスイッチオーバー

- スwitchオーバー完了後に、スタンバイ・データベースで適用プロセスを起動する時に、オンラインREDOログのクリアリング処理が動く
  - フェイルオーバーをする場合に、新プライマリをすぐに使えるようにするために、あらかじめオンラインREDOログの初期化をしている
  - LOG\_FILE\_NAME\_CONVERTパラメータを設定している場合に起こる
  - REDOログのサイズ、数に依存

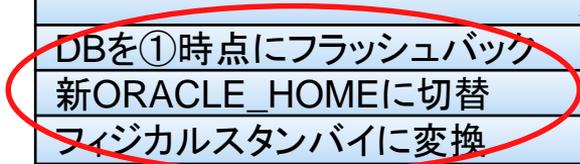
### <Alert.log の内容>

```
Clearing online redo logfile 1 +DATA/orcl/onlinelog/group_1.262.822332609
Clearing online log 1 of thread 1 sequence number 443
Clearing online redo logfile 1 complete
<以降略>
```

# 手順



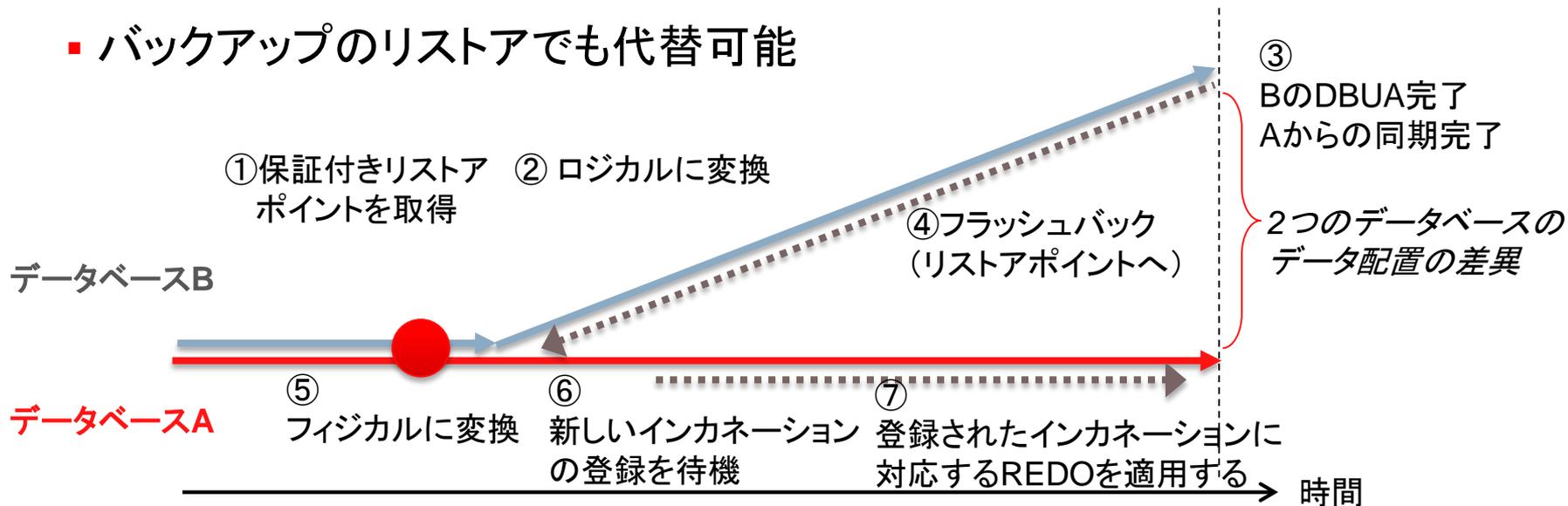
	REDO転送	
•ORACLE_HOMEインストール •保証付きリストアポイント取得	→	ORACLE_HOMEインストール
	→	ロジカル・スタンバイに変換
REDO転送を停止する	-	DBUAもしくは手動でUpgrade
REDO転送再開	→	REDO適用開始
スイッチオーバー		
DBを①時点にフラッシュバック	-	
新ORACLE_HOMEに切替	←	
フィジカルスタンバイに変換	-	
REDO同期開始 (REDO経由でcatupgrd反映)	←	
スイッチオーバー		
	→	



# ロジカルをフィジカルに戻す

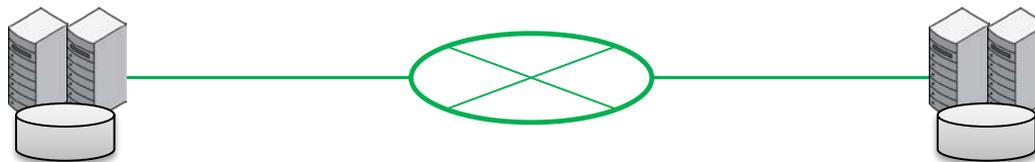
保証付きリストア・ポイント(フラッシュバック・データベース)

- ロジカルに変換後、両DBの物理構造の差分が広がる前の時点に戻す
- バックアップのリストアでも代替可能



# 手順

更新可  
更新不可



	REDO転送	
•ORACLE_HOMEインストール •保証付きリストアポイント取得	→	ORACLE_HOMEインストール
	→	ロジカル・スタンバイに変換
REDO転送を停止する	-	DBUAもしくは手動でUpgrade
REDO転送再開	→	REDO適用開始
スイッチオーバー		
DBを①時点にフラッシュバック	-	
新ORACLE_HOMEに切替	-	
フィジカルスタンバイに変換	-	
REDO同期開始 (REDO経由でcatupgrd反映)	←	
スイッチオーバー		
	→	

# フィジカルへの変換後の適用プロセス起動

新しいインカーネーション登録が行われてから適用プロセスが開始

- フラッシュバック、フィジカル・スタンバイへの変換後、適用プロセスを起動する際、新インカーネーションの登録を待って、適用プロセスが動き出す
  - 間違ったREDOを適用していかないように
- 登録するまでの間、下記メッセージが10秒おきに出力される(スタンバイ)  
<Alert.log の内容>

```
Managed Standby Recovery starting Real Time Apply
MRP0: Background Media Recovery waiting for new incarnation during transient logical
upgrade procedure
Errors in file /u01/app/oracle/diag/rdbms/orcl1/orcl1/trace/orcl1_mrp0_11138.trc:
ORA-19906: リカバリ・ターゲット・インカーネーションがリカバリ中に変更されました
Managed Standby Recovery not using Real Time Apply
```

# フィジカルへの変換後の適用プロセス起動

適用はインカネーション登録が済んでから動き出す

- 新インカネーションが登録されるのは、プライマリからREDOを受信する時
- フラッシュバックして、フィジカルに変換して、データベースをマウントした後、REDOが送られてくるのは、REOPEN属性で指定した時間経過後のログスイッチのタイミング(デフォルト300秒 +  $\alpha$ )
- この待機時間を短くするには
  - 策① `alter system set log_archive_dest_state_n=enable` を実行
  - 策② REOPEN 属性の値をあらかじめ小さく設定

# フォールバック

## 各フェーズにおけるフォールバック方法

- COMPATIBLE パラメータを変えない限りはダウングレード可能

失敗タイミング	復旧方法
② ロジカル・スタンバイに変換	フィジカル・スタンバイに戻す
③ ロジカル・スタンバイでのアップグレード	ダウングレード・スクリプト実行 (or フラッシュバック・データベース)、ORACLE_HOMEを切替
⑧ フィジカル・スタンバイに変換	フィジカル・スタンバイを再作成
⑨ フィジカル・スタンバイでのアップグレード (REDO適用)	この時点のプライマリから、フィジカル・スタンバイを再作成
⑪ テストで失敗	ダウングレード・スクリプト実行 (or フラッシュバック・データベース)、ORACLE_HOMEを切替

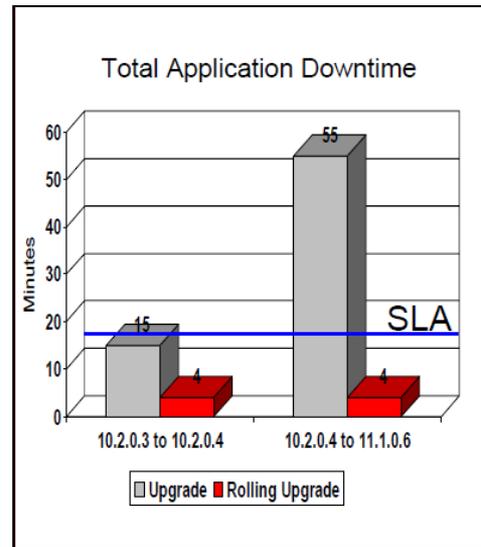
# 事例

## ■ UPS

- 計画停止のSLAは最大で15分
- 通常の方法だと難しかったが、ローリングアップグレードを使うことでSLAを達成した

## ■ Oracle 社内電子メールシステム

- 従来48分かかっていたシステム
- ローリング・アップグレードの手法で停止時間を2分に短縮した



<http://www.oracle.com/technetwork/database/features/availability/298781-2-133911.pdf>

ORACLE

# 複数のフィジカル・スタンバイ・データベース(ADG)を用いた手法

# 一時ロジカル・スタンバイを用いた手法の課題

## アプリケーションの停止が発生する理由は？

- 一時ロジカル・スタンバイを使用したローリング・アップグレードで、アプリケーションの停止が発生するフェーズはどこか？
    - スイッチ・オーバーを行うフェーズ
  - なぜスイッチ・オーバー時にダウンタイムが発生するか？
    - 更新できなくなる  
フィジカル・スタンバイの場合はセッションが切断される
- スイッチオーバーをしなければ良い？
- 更新をしなければよい？

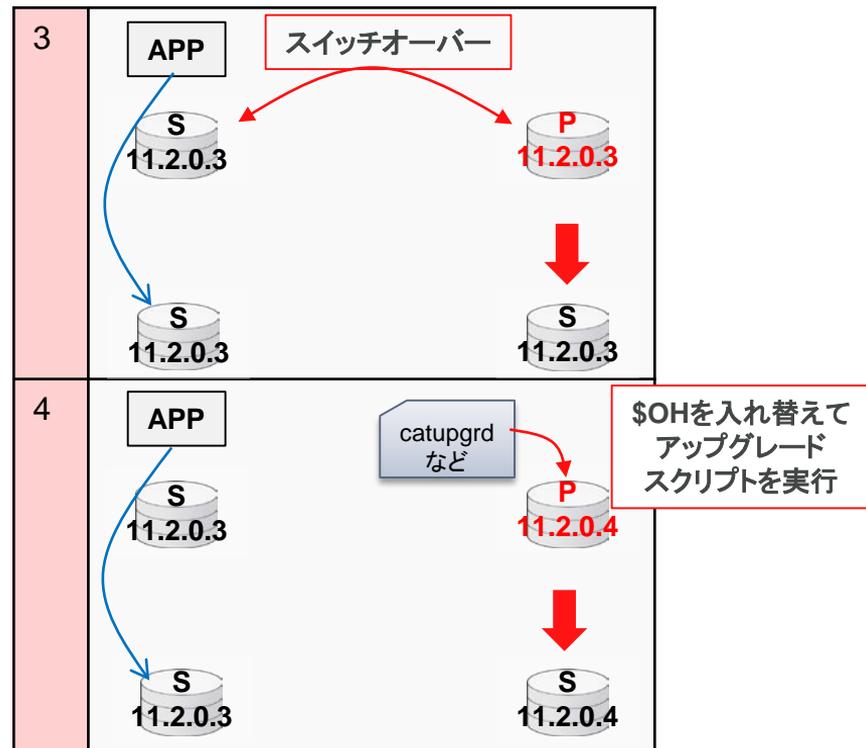
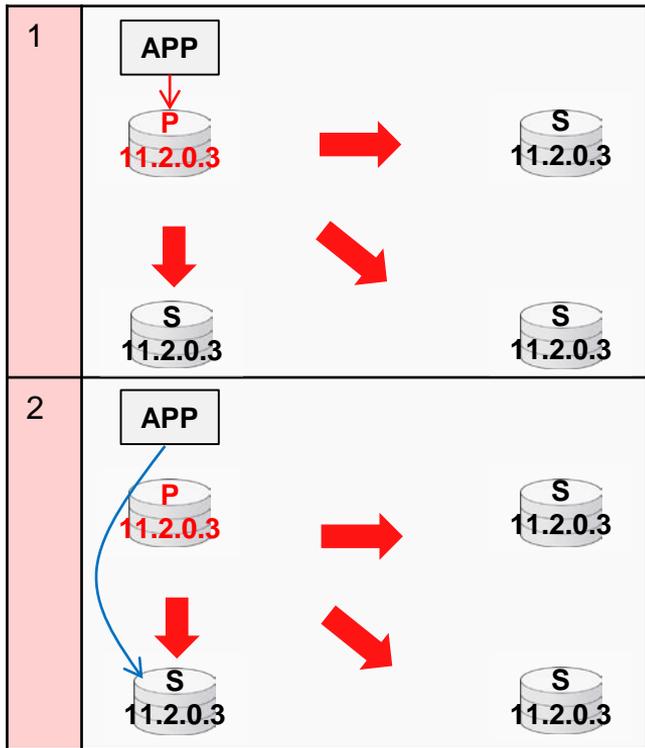
# 特徴

## 複数のフィジカル・スタンバイ・データベース(ADG)を用いた手法

- 2つ以上のフィジカル・スタンバイ(Read-OnlyでOPEN)があれば実現可
- 前述の「基本手法」を行っている間、アプリケーションをアップグレードに  
与していないスタンバイDBに一時的にRead-Onlyで退避させる
- ゼロ・ダウンタイム(Read-Only時間含む)でアップグレードを実現できる
  - アプリケーションで透過的に接続先を切り替える機構は必要
  - Read-Only アクセスとなるのは1時間程度
- 複数のフィジカル・スタンバイを常に保持しているので、ローリング・アップグ  
レード中の障害時にも迅速にフェイルオーバー可能

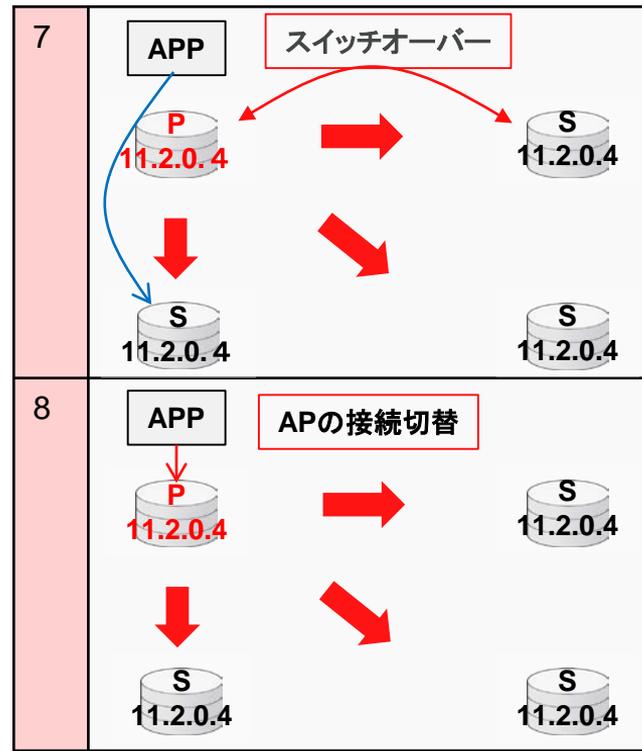
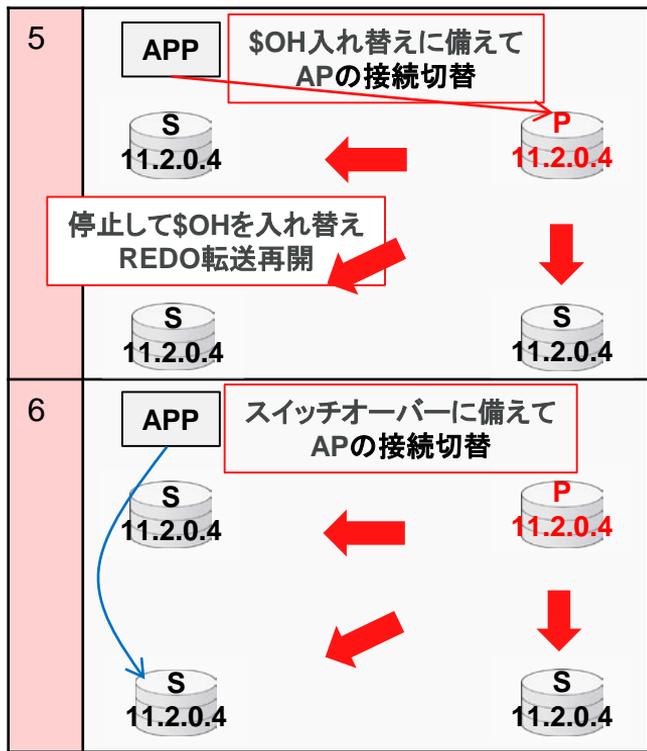
# 手順概要

-  REDO転送
-  Read-Write での接続
-  Read-Only での接続



# 手順概要

-  REDO転送
-  Read-Write での接続
-  Read-Only での接続



# アプリケーションの接続切替

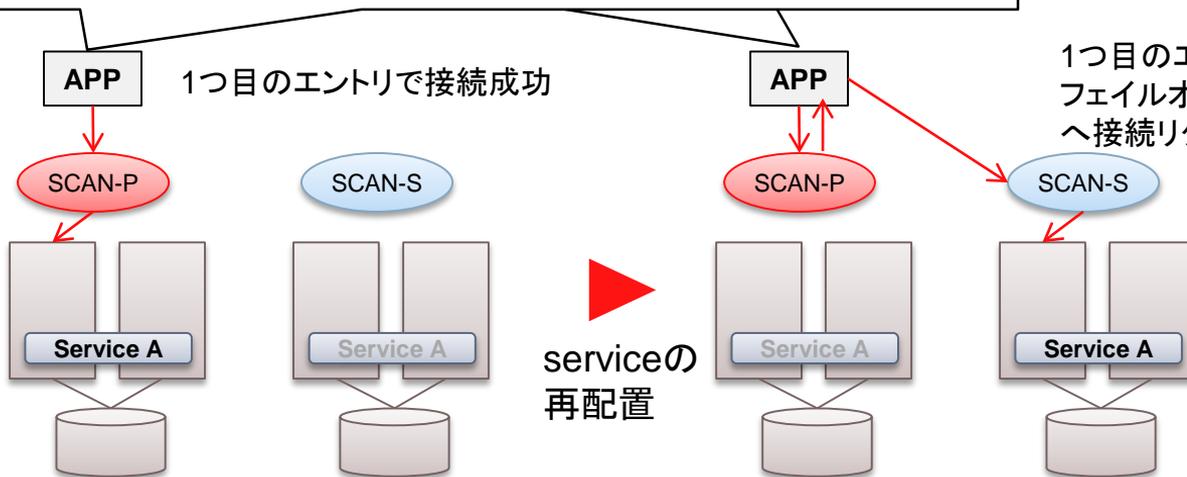
- アプリケーションが使うデータソースを切り替えるという制御で実現
  - 切替先のDBとの間で事前に接続しておき、必要なタイミングが来たらアプリケーション側で切り替える
- 切替先のDBではデータベースのロール変換がなく(=スイッチオーバーに  
関与しない)、接続断が起こらないため使える方法

# アプリケーションの接続切替

常にコネクションの切り張りをしているような場合

```
TEST =  
(DESCRIPTION_LIST =  
  (LOAD_BALANCE=OFF)  
  (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=SCAN-P)(PORT=1521))  
    (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=serviceA)))  
  (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=SCAN-S)(PORT=1521))  
    (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=serviceA)))  
)
```

接続文字列にプライマリ  
スタンバイ両方のエントリを  
記述しておく



# アプリケーションの接続切替

コネクションを張ったままの場合は？

- 既存接続を一旦切る必要がある(下記一例)
  - データベースServiceを移動 (“-f” オプション有り)
    - すべてのセッションが強制的に切断される
  - データベースを “transactional” で停止
    - アクティブなトランザクションを完了してからインスタンスを停止
    - 新しい接続や、新しいトランザクションの開始は許可されない
- 切断後は以下のような方法で接続先が切り替わる
  - アプリケーションが使うデータソースを変える(前述)
  - 事前にデータベースServiceを移動しておく

# 一時 GoldenGate (Transient GoldenGate) を用いた手法

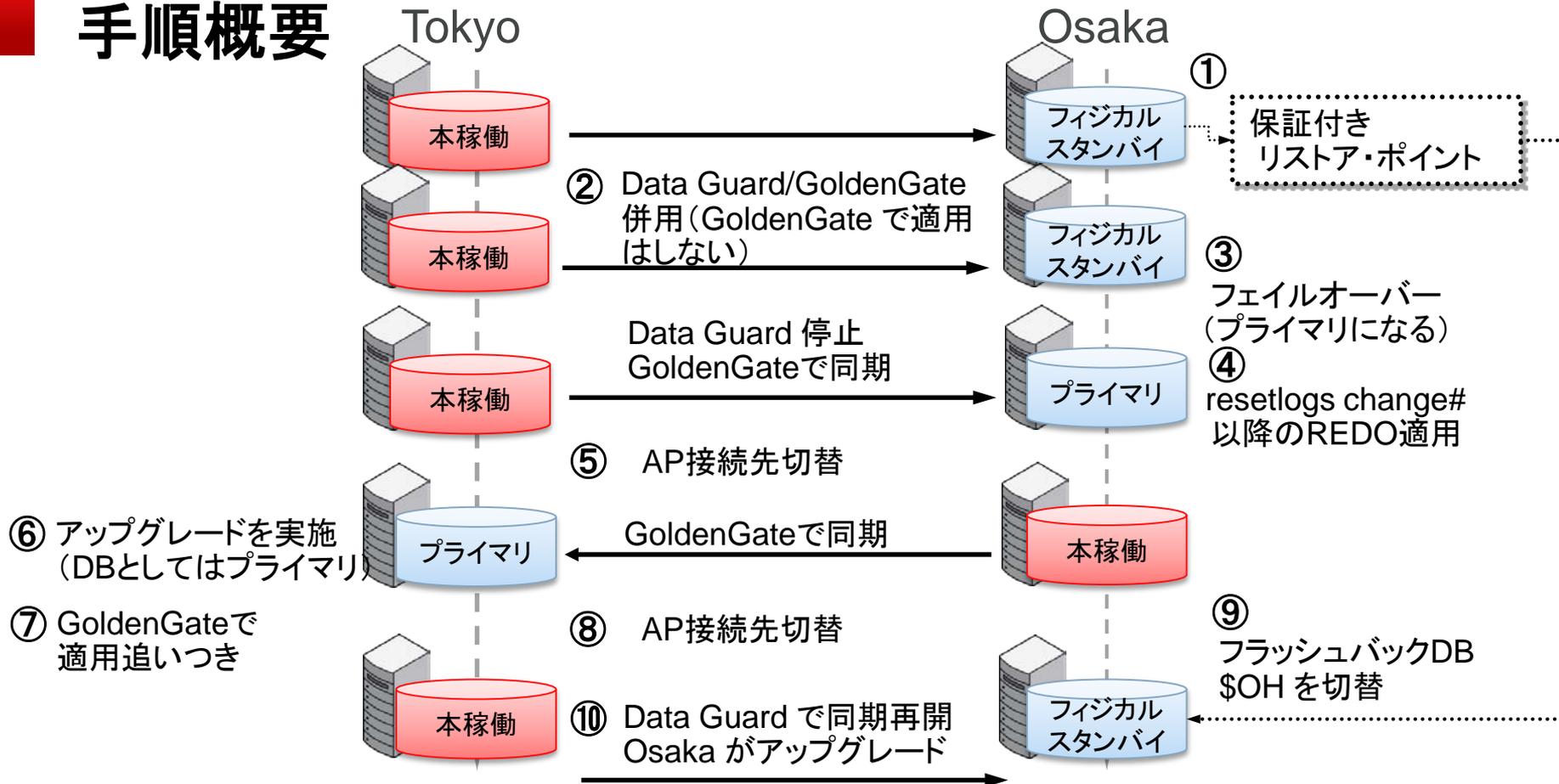
# 特徴

- 通常運用時は Data Guardのブロック破損対策機能を使える
- PSR適用時は、GoldenGateを使って同期するため、常時更新可能(ロール変換がない)
- データ競合の解消についての検討が必要
  - 一時的にActive-Active な状態になるため
- PSR適用完了後、再度 Data Guard で同期を取るためには、Flashback Database を使用
- アプリケーションの接続切替時間のみの停止時間でアップグレード可能

# 手順概要

項番	作業内容	
	Osaka リストアポイント作成	Data Guardのfailover Active-ActiveでGoldenGateの同期
	Tokyo GG起動	
	OsakaへFailoverの実施	
	Tokyo/Osaka GG起動 ※全プロセス起動	
	Tokyo -> Osaka の接続切替	
	Tokyo GG 適用停止	プライマリ側のDBにPSRの適用
	Tokyo DBアップグレード	
	Tokyo GG 適用再開	
	Osaka -> Tokyoの接続切替	スタンバイ側のDBをFailover前に Flashback REDO Apply でアップグレード
	Osaka/Tokyo GG停止	
	OsakaをFlashbackしてStandby DBへ戻す	
	Osaka アップグレード準備	
	Data Guard Redo Apply再開	

# 手順概要



# 手順概要

- GoldenGate で同期を取るために、Osaka側でフェイルオーバーを実施し、両DB(Tokyo、Osaka)をプライマリとして動作させる
- フェイルオーバーした時点以降は、GoldenGate を使ってREDOを適用

※本手法(Transient GoldenGate)の詳細については、  
公開予定(年内を目途)のホワイトペーパーを参照下さい

# まとめ

## PSR適用における Pros & Cons

#	手法	停止時間の目安	スイッチオーバー時間		通常運用時		PSR適用時
			ロール変換	接続性	モード	データ保護	モード
1	Physical Standby	数時間	不要	再接続時間が必要	DG (Physical)	ABMR Lost Write	DG (Physical)
2	Transient Logical Standby	5分未満 x 2回	必要	S/O時 : 再接続不要 S/B時 : 再接続必要	DG (Physical)	ABMR Lost Write	DG (Logical)
3	Multi-Physical Standby	ゼロ (Read Only期間有り)	必要	一時的にADGへ接続 →待機無し	DG (Physical)	ABMR Lost Write	DG (Physical)
4	GoldenGate	ゼロ	不要	Active-Active構成 →待機無し	GG	N/A	GG
5	Transient GoldenGate	ゼロ	不要	Active-Active構成 →待機無し	DG (Physical)	ABMR Lost Write	GG

ABMR : Auto Block Media Recovery

ORACLE

**Hardware and Software**

**ORACLE®**

**Engineered to Work Together**

ORACLE®