
ORACLE®

ORACLE

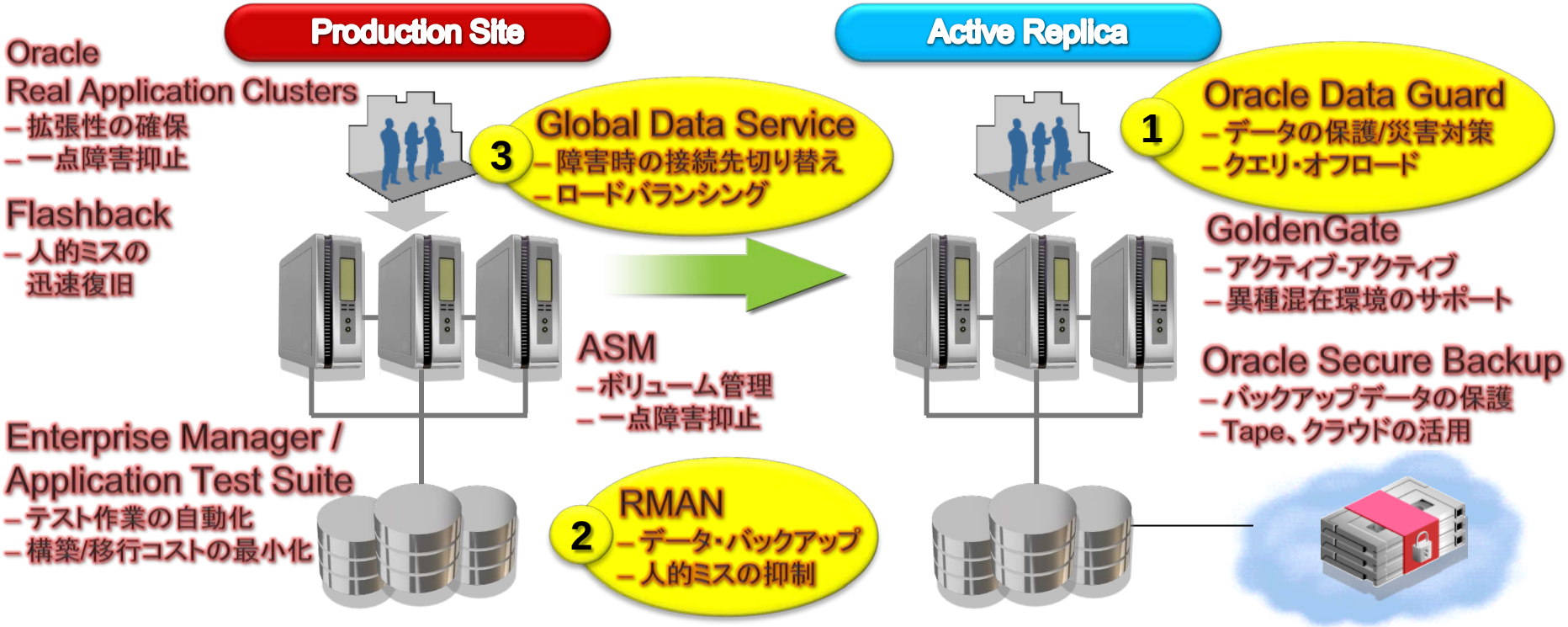
Oracle Database 12*c* Release 1

Data Guard

日本オラクル株式会社

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。

オラクルが提供するテクノロジーとアーキテクチャ



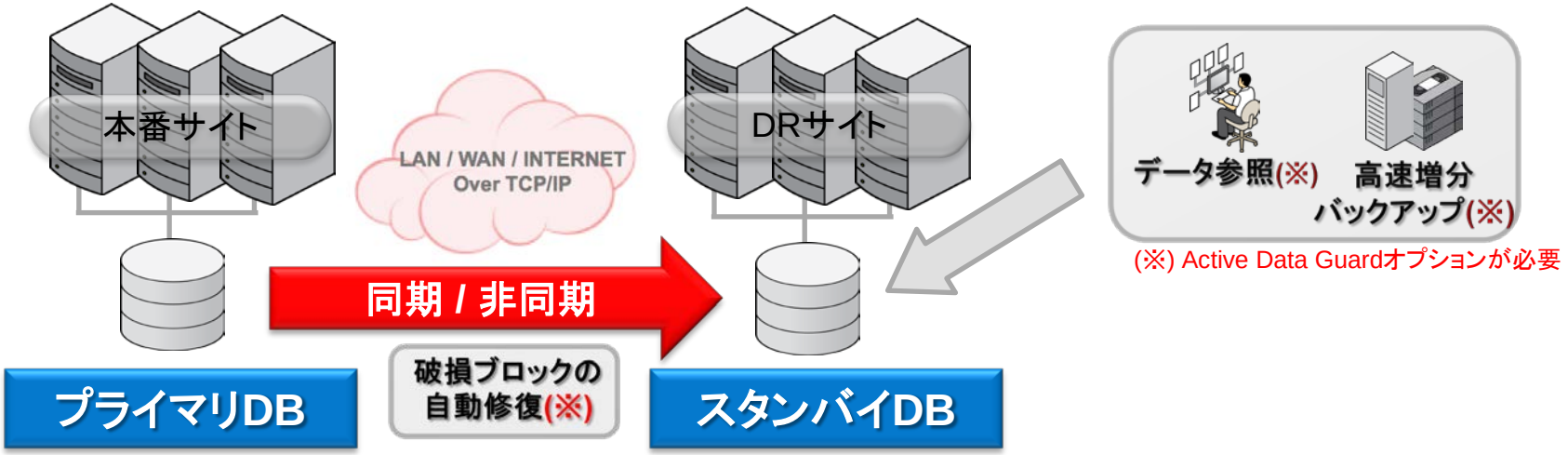
Data Guard

“データの高可用性、データ保護、
障害時リカバリの提供”

Data Guard とは

データ保護・可用性の提供 + 適切なROIを実現

自動データ同期 **迅速な切り替え** **リソースの活用**



(※) Active Data Guardオプションが必要

バックアップサイトとの自動的なデータ同期機能の提供
バックアップサイトのリソースを有効活用し、本番サイトの負荷を低減

Data Guard 新機能

1. 同期処理の高速化

"データの保全性を高めた上での
パフォーマンスへの影響の極小化"

Data Guard 新機能：同期処理の高速化

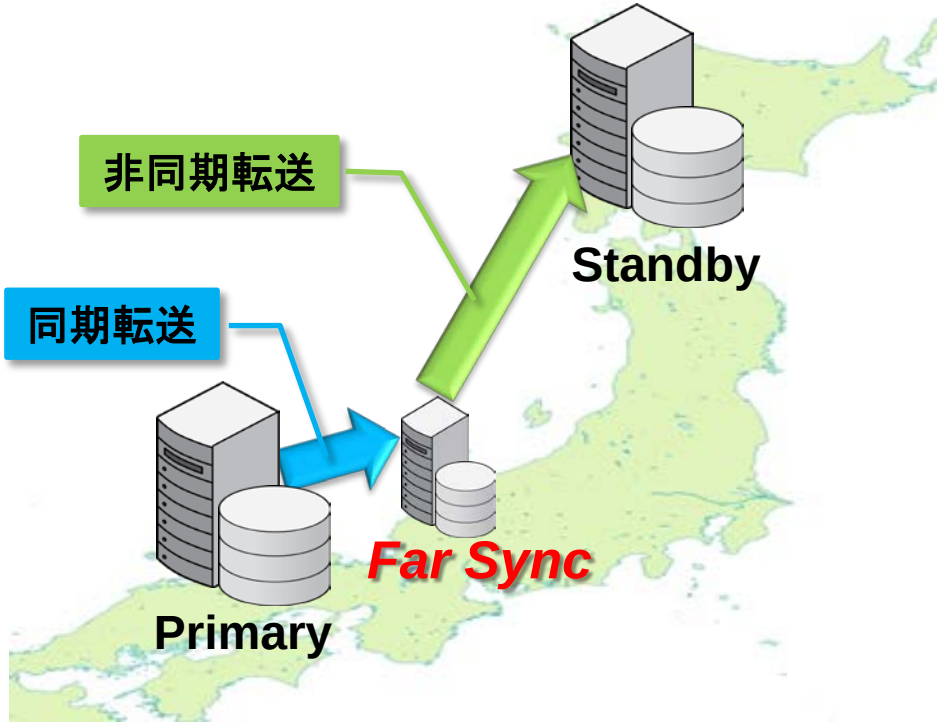
- **遠隔同期インスタンス**
  - 遠隔地のスタンバイ DB との同期転送を実現
- **リアルタイム・カスケード**
  - カスケード・スタンバイへリアルタイムに REDO ログを転送
- **FastSync**
  - 最大可用性モードでのパフォーマンス向上

Data Guard 新機能：同期処理の高速化

- **遠隔同期インスタンス**
  - 遠隔地のスタンバイ DB との同期転送を実現

遠隔同期インスタンス

概要



遠隔地間の Data Guard 構成

- 遠隔同期インスタンス(Far Sync)を設置

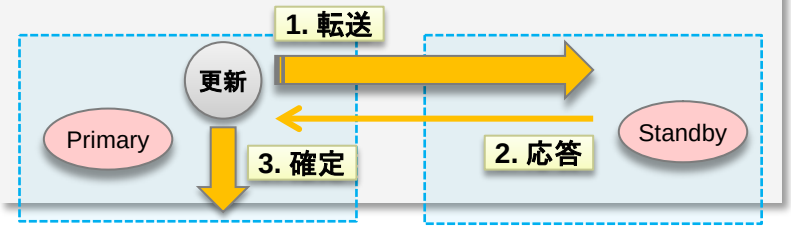
- 同期転送によるデータロスなしの同期を実現
- パフォーマンスへの影響を極小化

Data Guard によるデータ保護の仕組み

同期転送と非同期転送

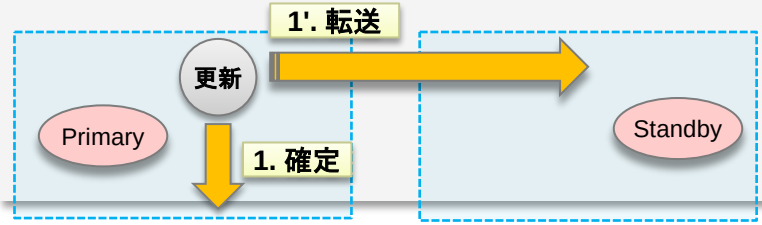
同期転送 (SYNC)

- **データ保護**
  - プライマリ DB でのデータ更新はスタンバイ DB への転送完了後に確定する
- **パフォーマンスへの影響**
  - スタンバイ DB への転送時間に依存してプライマリ DB の更新処理が待機する



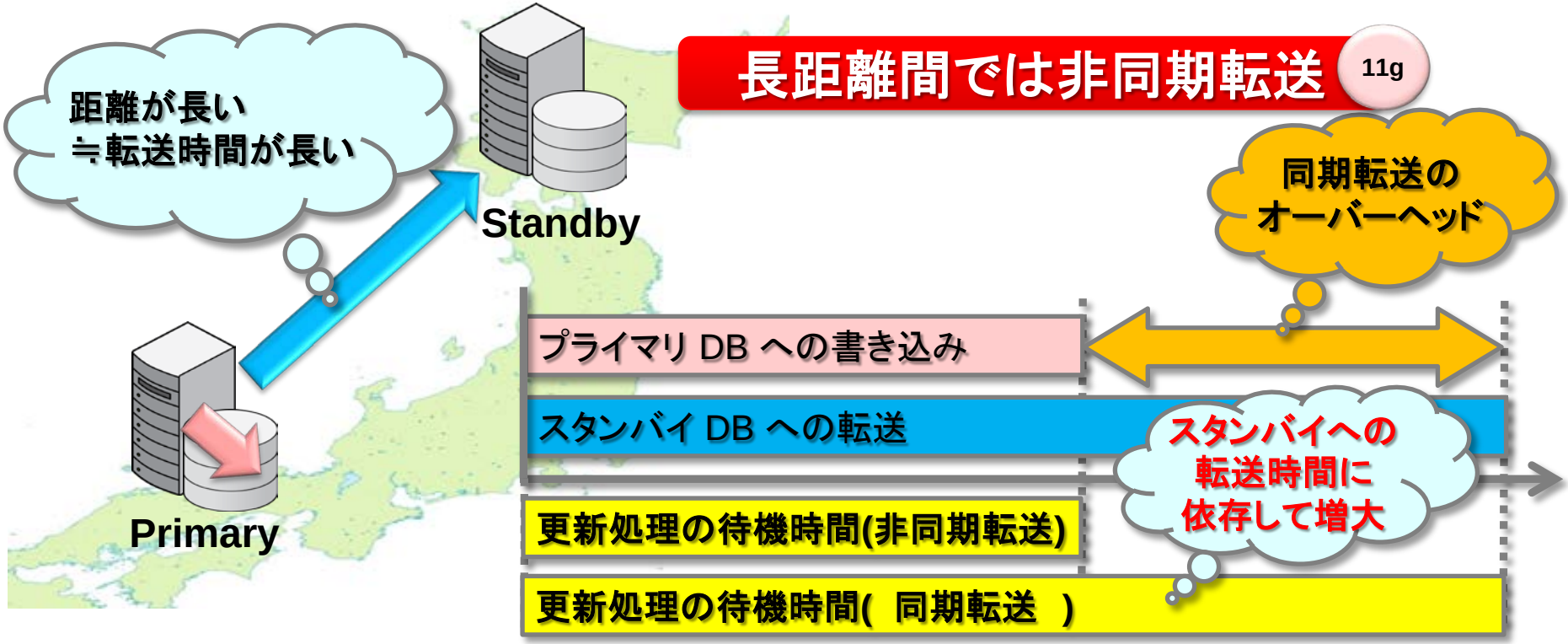
非同期転送 (ASYNC)

- **データ保護**
  - プライマリ DB で更新されたはスタンバイ DB への転送未完了でも確定する
- **パフォーマンスへの影響**
  - プライマリ DB への更新処理はスタンバイ DB への転送を待機しない



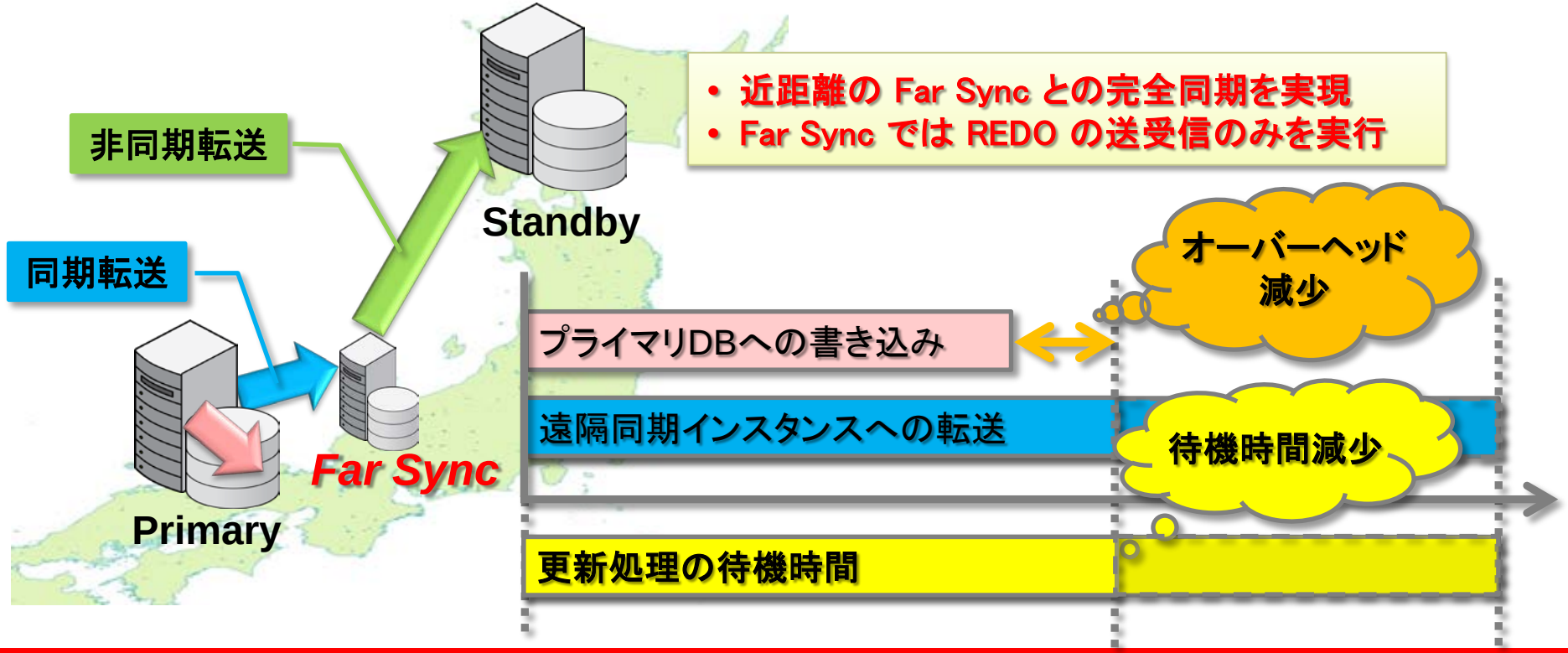
ゼロ・データロスの同期

REDO データの転送時間が長いケース



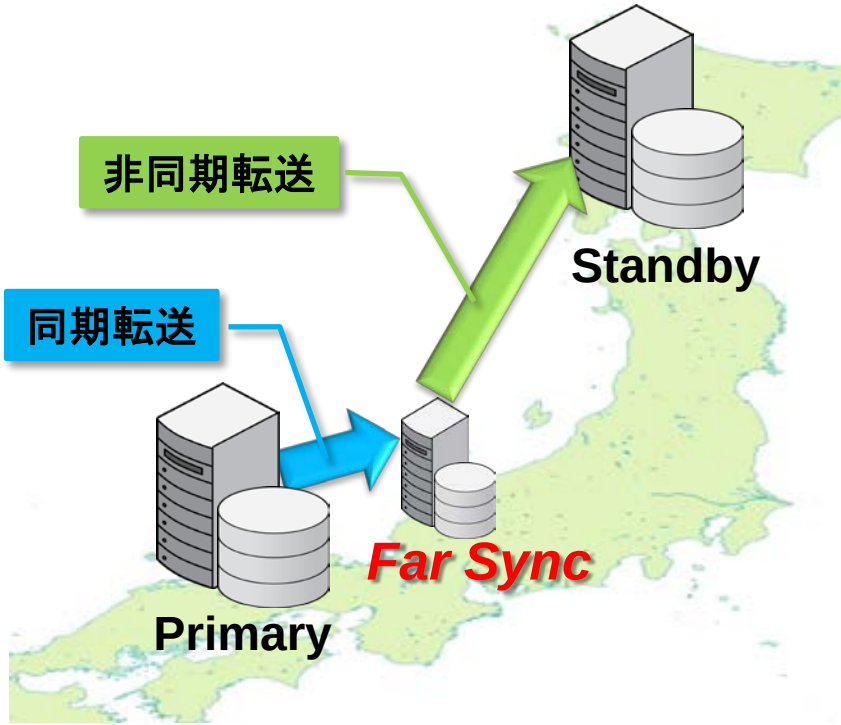
遠隔同期インスタンス

遠隔同期インスタンスを使用した Data Guard 構成



遠隔同期インスタンス

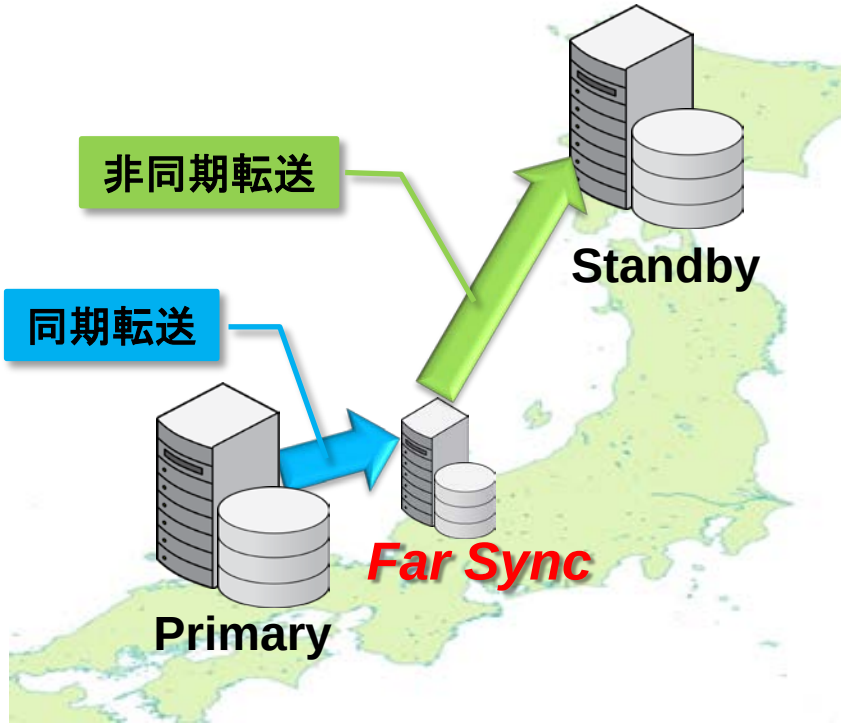
特徴



- **同期転送のオーバーヘッド軽減**
  - ✓ 近距離の遠隔同期インスタンスまでの同期転送
- **ゼロ・データロスの実現**
  - ✓ Primary DB 停止時にも、必要な REDO データは遠隔同期インスタンスへ転送済み
- **最小限のファイル構成**
  - ✓ 遠隔同期インスタンスは制御ファイルと REDO ログファイルのみから構成
- **シームレスなロール変換**
  - ✓ 遠隔同期インスタンスを意識せずスイッチオーバーの実行が可能

遠隔同期インスタンス

特徴：同期転送のオーバーヘッド軽減



- **同期転送のオーバーヘッド軽減**
  - ✓ 近距離の遠隔同期インスタンスまでの同期転送
- **ゼロ・データロスの実現**
  - ✓ Primary DB 停止時にも、必要な REDO データは遠隔同期インスタンスへ転送済み
- **最小限のファイル構成**
  - ✓ 遠隔同期インスタンスは制御ファイルと REDO ログファイルのみから構成
- **シームレスなロール変換**
  - ✓ 遠隔同期インスタンスを意識せずスイッチオーバーの実行が可能

遠隔同期インスタンス

特徴：ゼロ・データロスの実現

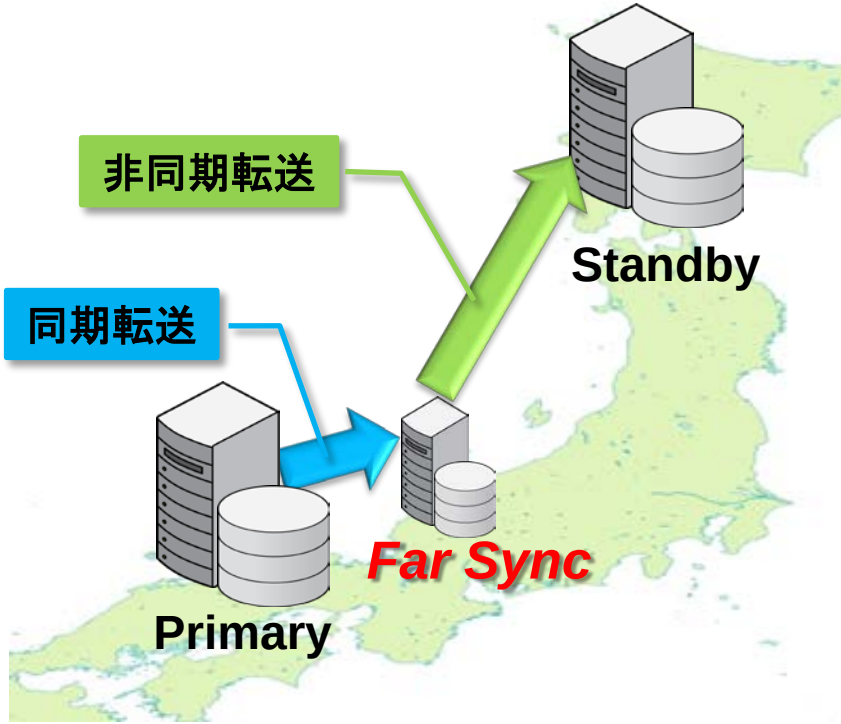
- **ゼロ・データロスの実現**
  - ✓ Primary DB 停止時にも、必要な REDO データは遠隔同期インスタンスへ転送済み

すべてのデータをStandbyが受信 ＝ データロス無し

- **シームレスなロール変換**
  - ✓ 遠隔同期インスタンスを意識せずスイッチオーバーの実行が可能

遠隔同期インスタンス

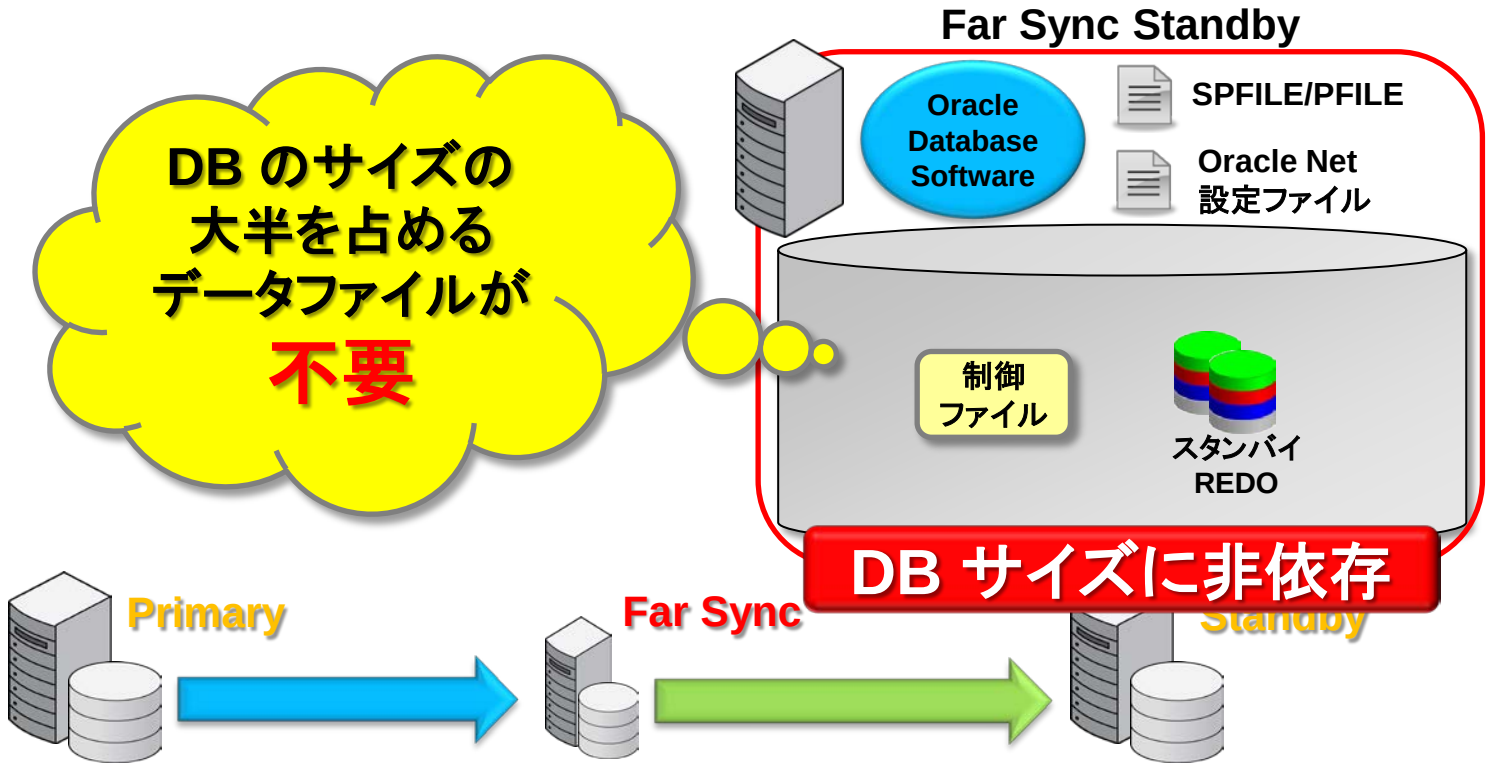
特徴：最小限のファイル構成



- 同期転送のオーバーヘッド軽減
  - ✓ 近距離の遠隔同期インスタンスまでの同期転送
- ゼロ・データロスの実現
  - ✓ Primary DB 停止時にも、必要な REDO データは遠隔同期インスタンスへ転送済み
- **最小限のファイル構成**
  - ✓ 遠隔同期インスタンスは制御ファイルと REDO ログファイルのみから構成
- シームレスなロール変換
  - ✓ 遠隔同期インスタンスを意識せずスイッチオーバーの実行が可能

遠隔同期インスタンス

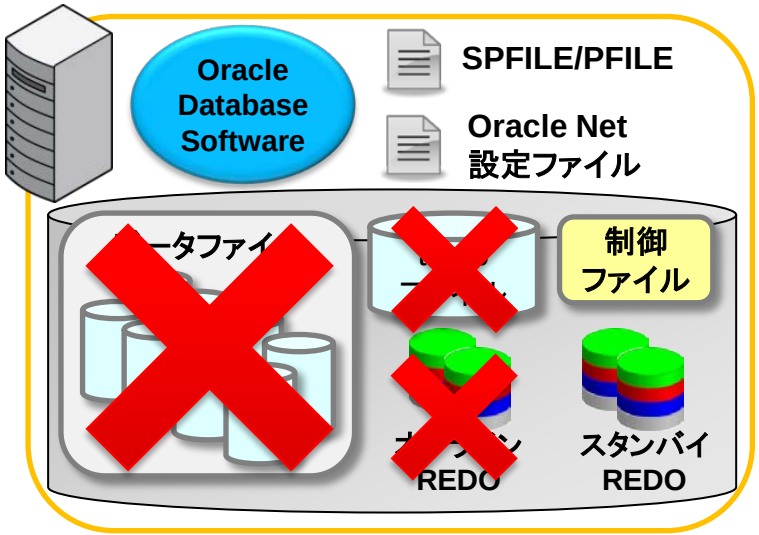
特徴：最小限のファイル構成



遠隔同期インスタンス

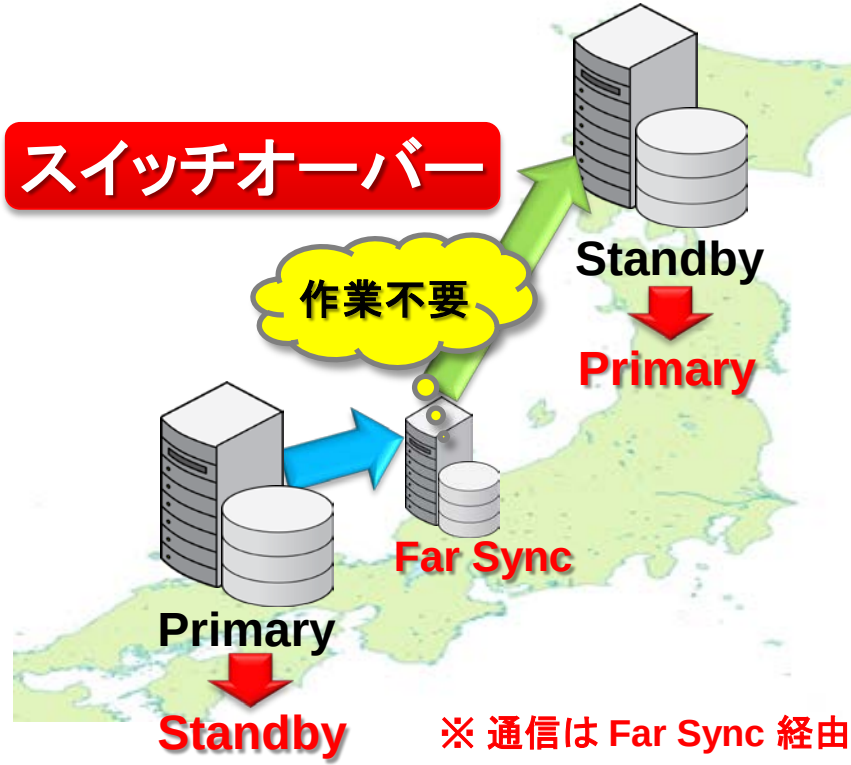
特徴：最小限のファイル構成

- 遠隔同期インスタンスの構成に必要となるファイル
  - 初期化パラメータファイル (pfile / spfile)
  - パスワードファイル
  - tnsnames.ora
  - listener.ora
  - 制御ファイル
  - スタンバイ REDOログファイル
  - × データファイル
  - × 一時ファイル (temp ファイル)
  - × オンライン REDO ログファイル



遠隔同期インスタンス

特徴：シームレスなロール変換



※ 通信は Far Sync 経由

- **同期転送のオーバーヘッド軽減**
  - ✓ 近距離の遠隔同期インスタンスまでの同期転送
- **ゼロ・データロスの実現**
  - ✓ Primary DB 停止時にも、必要な REDO データは遠隔同期インスタンスへ転送済み
- **最小限のファイル構成**
  - ✓ 遠隔同期インスタンスは制御ファイルと REDO ログファイルのみから構成
- **シームレスなロール変換**
  - ✓ 遠隔同期インスタンスを意識せずスイッチオーバーの実行が可能

遠隔同期インスタンス

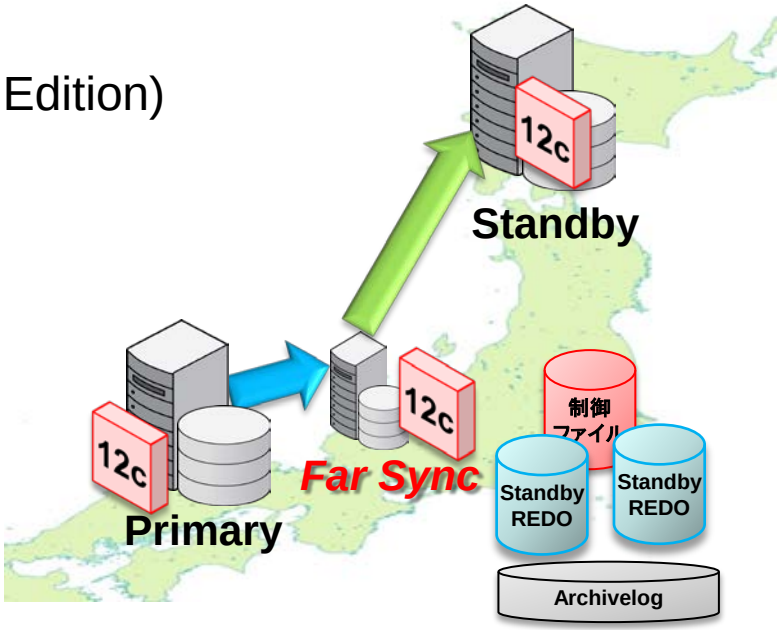
考慮点、注意点、設定

1. 構築手順
2. 起動モード
3. 遠隔同期インスタンス停止時の自動的なログ転送先の切り替え
4. 活用例
5. スイッチオーバー後を考慮した構成
6. スイッチオーバー後の遠隔同期インスタンスの活用
7. 制限

遠隔同期インスタンス

1. 環境構築手順

- 遠隔同期インスタンス作成のための要件
  - Oracle Database Software (Enterprise Edition)
    - ✓ Primary / Standby と同一バージョン
  - DISK領域の確保
    - ○ スタンバイ REDO ログファイル
    - ○ 制御ファイル
    - ○ アーカイブログファイル
    - × データファイル
    - × 一時ファイル(tempfile)
    - × オンライン REDO ログファイル



遠隔同期インスタンス

1. 環境構築手順

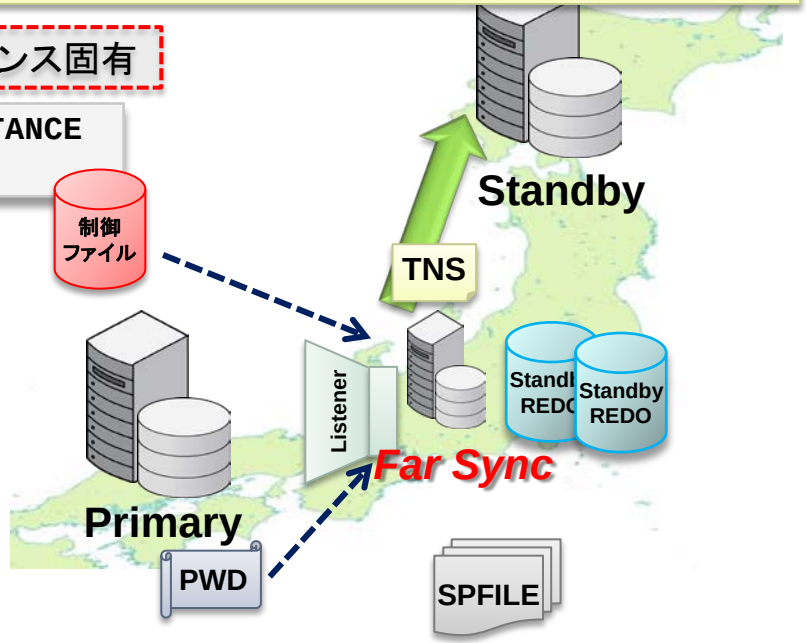
◆ 手順1以外はスタンバイ DB 作成時と同一

- 遠隔同期インスタンスに関する作業
  1. 制御ファイルの作成

遠隔同期インスタンス固有

```
SQL> ALTER DATABASE CREATE FAR SYNC INSTANCE
     CONTROLFILE AS '<file_name>';
```

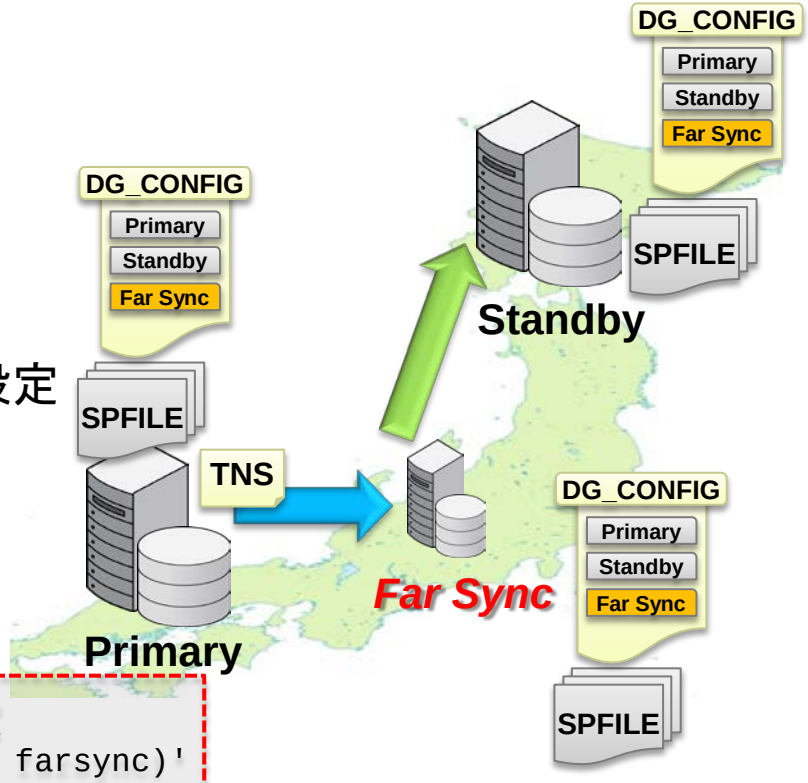
  2. 初期化パラメータファイルの作成
  3. ネットワークの設定とリスナーの起動
     - tnsnames.ora、listener.ora
  4. パスワードファイルのコピー
     - Primary DB のファイルをコピー
  5. スタンバイ REDO ログファイルの作成
     - 未作成の場合、MOUNT 後に作成



遠隔同期インスタンス

1. 環境構築手順

- Primary に関する作業
  1. LOG_ARCHIVE_CONFIG パラメータに遠隔同期インスタンスを追加
  2. ネットワークの設定 (tnsnames.ora)
  3. ログ転送先を遠隔同期インスタンスに設定
- Standby に関する作業
  1. LOG_ARCHIVE_CONFIG パラメータに遠隔同期インスタンスを追加



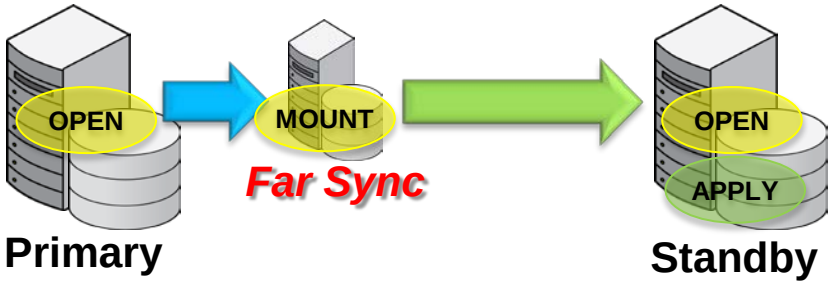
◆ Data Guard 構成の DB_UNIQUE_NAME を指定

```
log_archive_config='dg_config=(primary,standby, farsync)'
```

遠隔同期インスタンス

2. 起動モード

- 遠隔同期インスタンス
  - MOUNT のみ可能
  - OPEN 及び REDO 適用は不可



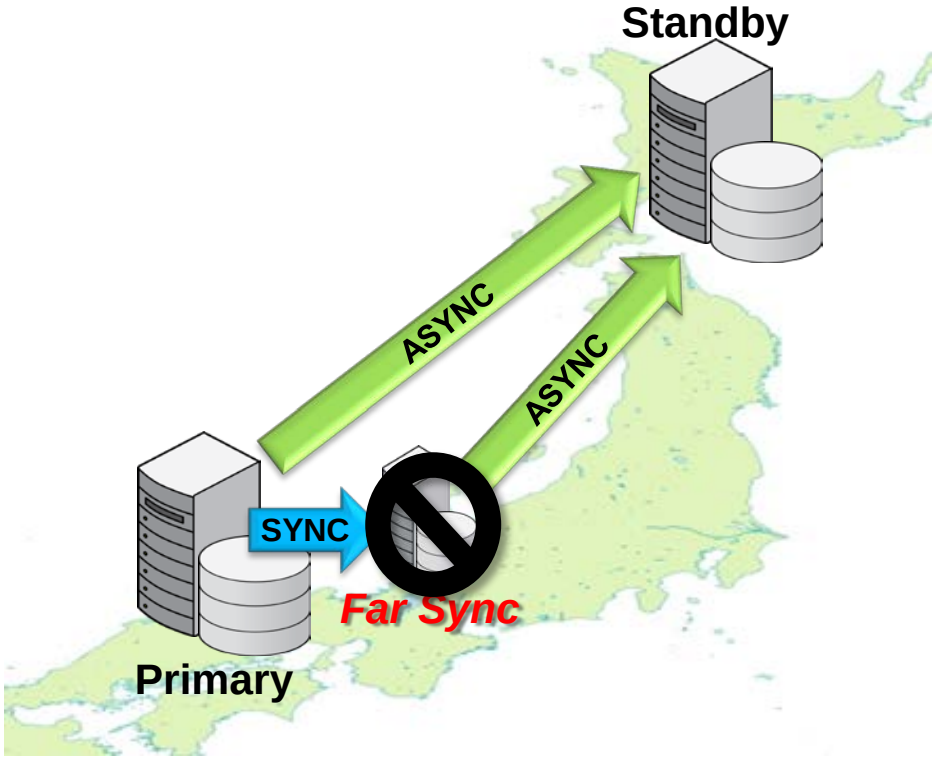
	Primary	Far Sync	Standby
MOUNT	○	●	○
MOUNT + REDO APPLY	×	×	●
OPEN (Read Write)	●	×	×
OPEN (Read Only)	○	×	○
OPEN + REDO APPLY	×	×	● (※)

× 設定不可
○ 設定可
● 設定可(推奨)

※ Active Data Guard オプションが必要

遠隔同期インスタンス

3. 遠隔同期インスタンス停止時の自動的なログ転送先の切り替え



遠隔同期インスタンス停止

↓

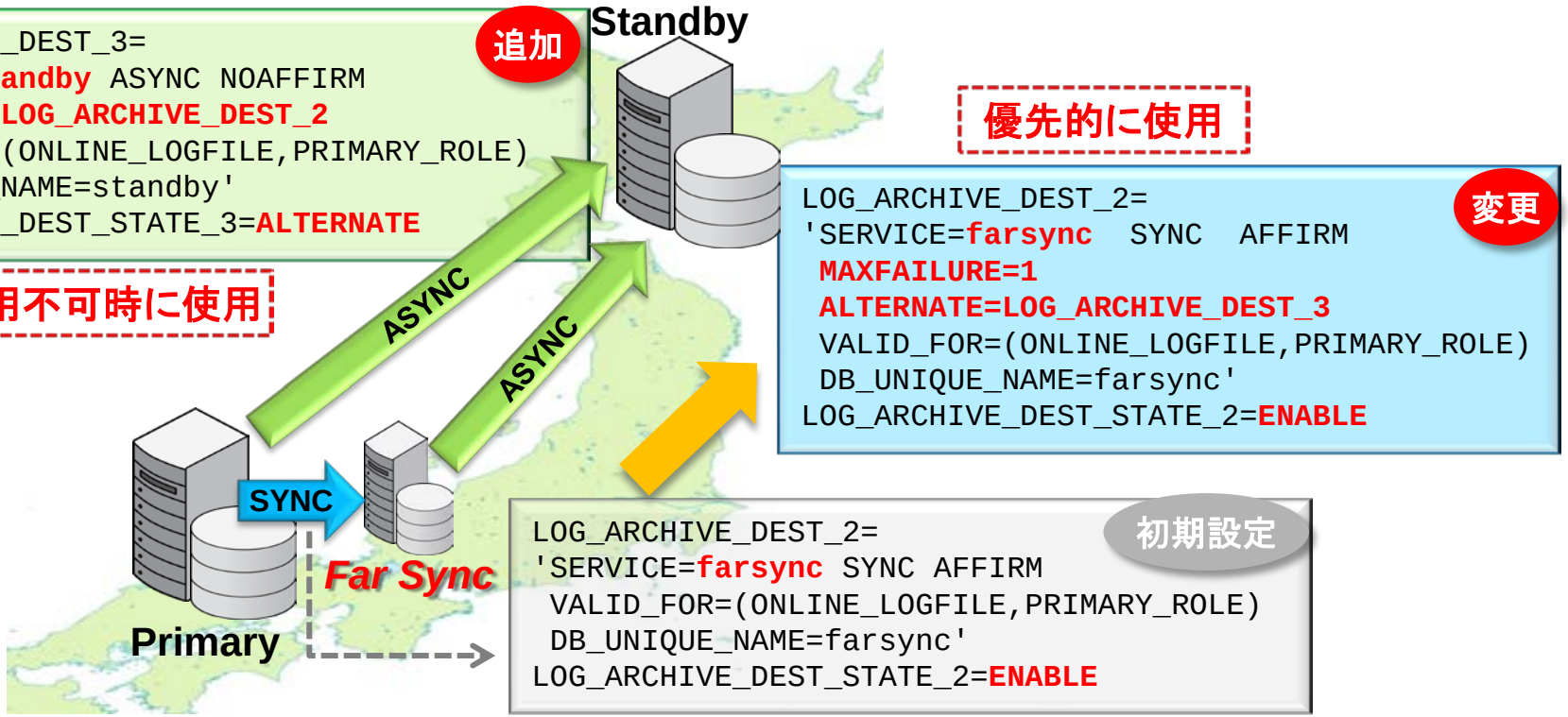
Primary からのログ転送失敗

↓

Standby へのログ転送開始

遠隔同期インスタンス

3. 遠隔同期インスタンス停止時の自動的なログ転送先の切り替え

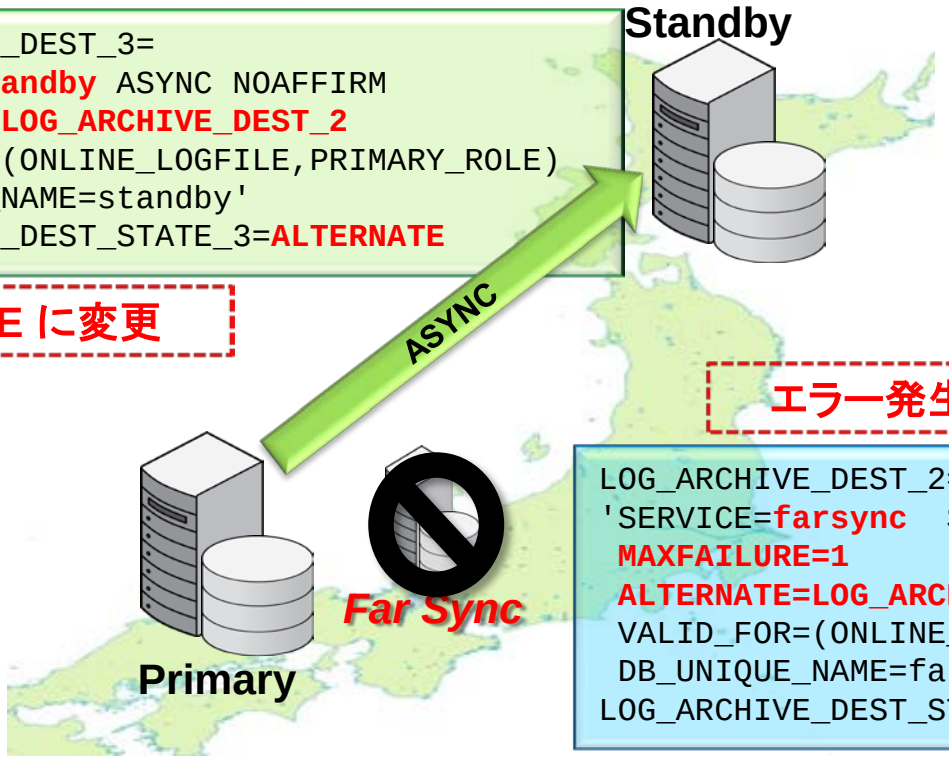


遠隔同期インスタンス

3. 遠隔同期インスタンス停止時の自動的なログ転送先の切り替え

```
LOG_ARCHIVE_DEST_3=
'SERVICE=standby ASYNC NOAFFIRM
 ALTERNATE=LOG_ARCHIVE_DEST_2
 VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)
 DB_UNIQUE_NAME=standby'
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

ENABLE に変更



Far Sync 停止時

エラー発生時に ALTERNATE に変更

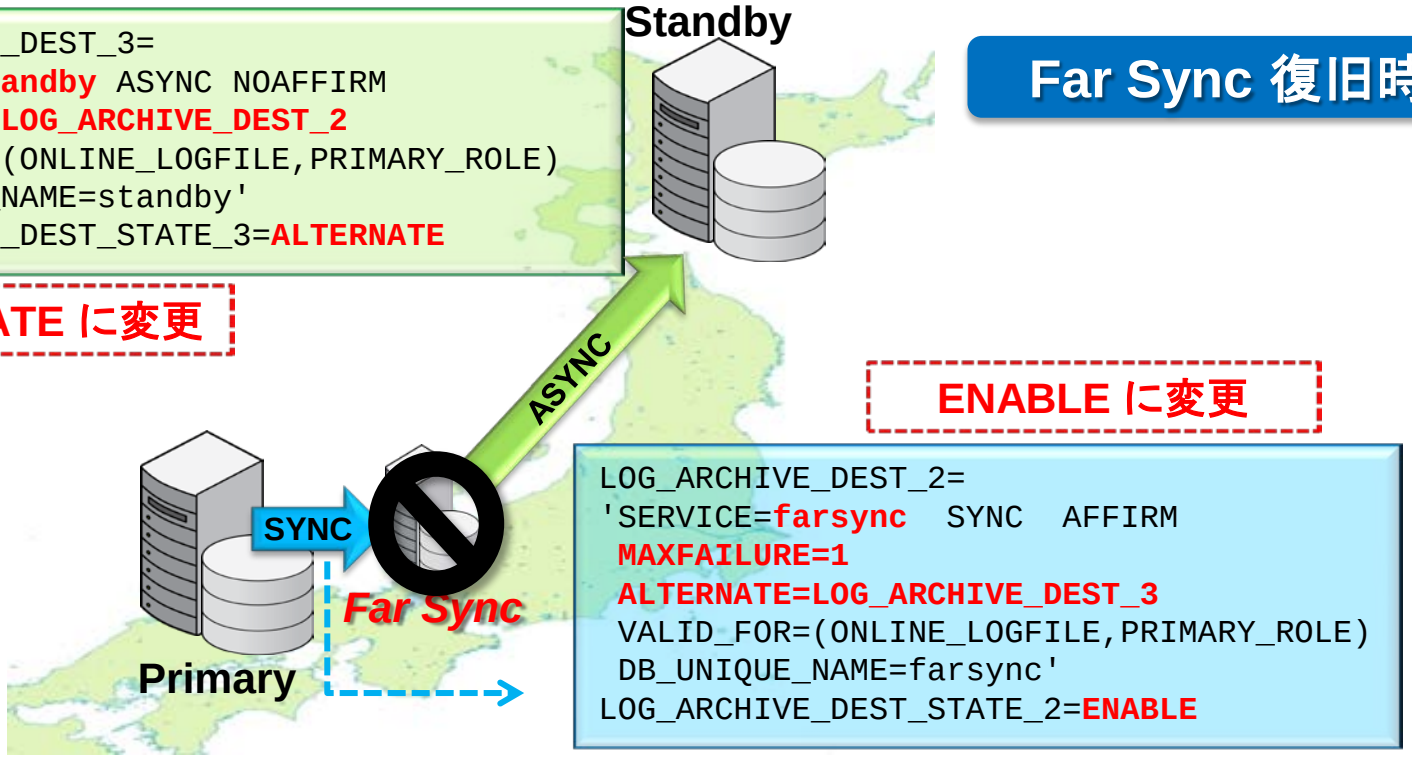
```
LOG_ARCHIVE_DEST_2=
'SERVICE=farsync  SYNC  AFFIRM
 MAXFAILURE=1
 ALTERNATE=LOG_ARCHIVE_DEST_3
 VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)
 DB_UNIQUE_NAME=farsync'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

遠隔同期インスタンス

3. 遠隔同期インスタンス停止時の自動的なログ転送先の切り替え

```
LOG_ARCHIVE_DEST_3=
'SERVICE=standby ASYNC NOAFFIRM
 ALTERNATE=LOG_ARCHIVE_DEST_2
 VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)
 DB_UNIQUE_NAME=standby'
LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

ALTERNATE に変更



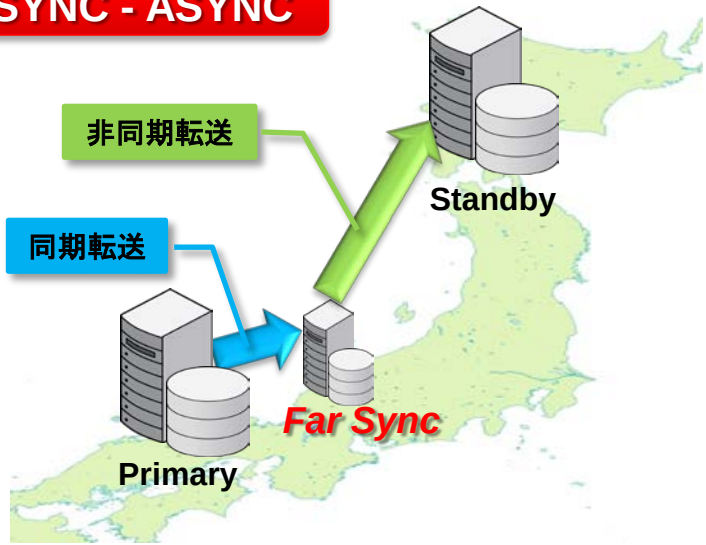
ENABLE に変更

```
LOG_ARCHIVE_DEST_2=
'SERVICE=farsync  SYNC  AFFIRM
 MAXFAILURE=1
 ALTERNATE=LOG_ARCHIVE_DEST_3
 VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)
 DB_UNIQUE_NAME=farsync'
LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

遠隔同期インスタンス

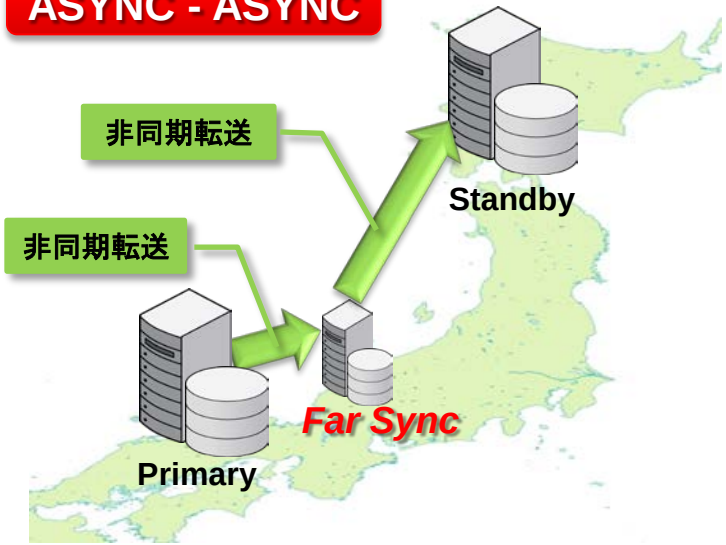
4. 活用例

SYNC - ASYNC



- ゼロデータロスの同期
- 同期転送によるオーバーヘッド減少

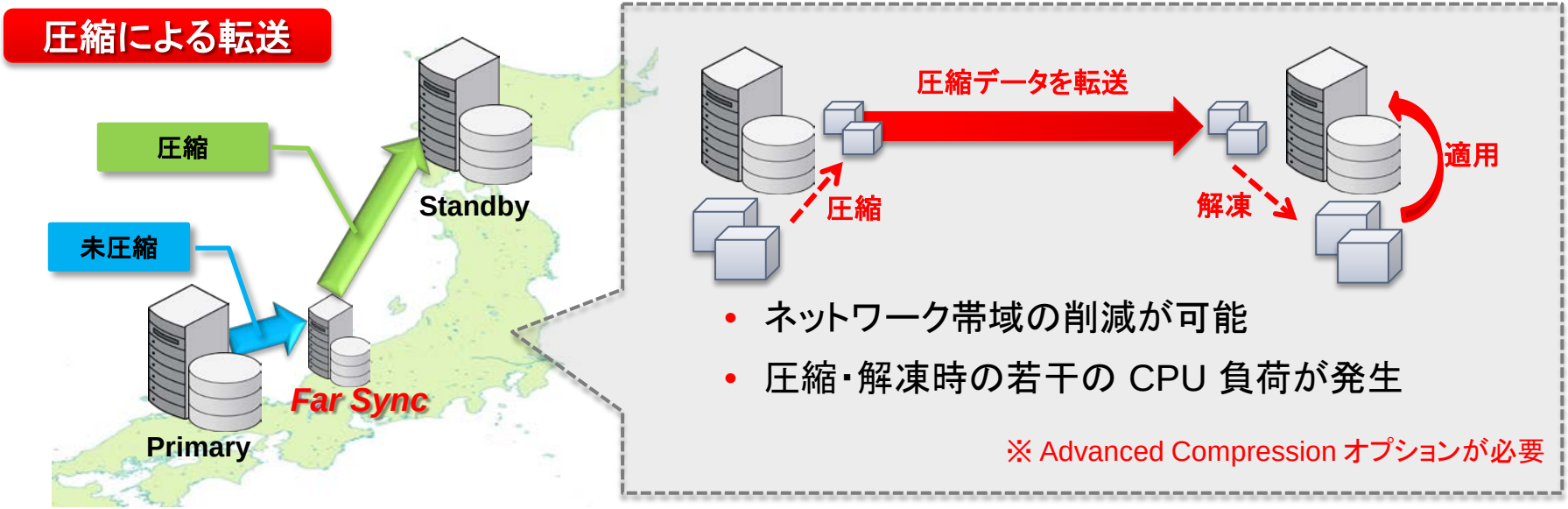
ASYNC - ASYNC



- 非同期転送によるオーバーヘッド軽減
- ニアゼロデータロスの同期

遠隔同期インスタンス

4. 活用例：圧縮による転送

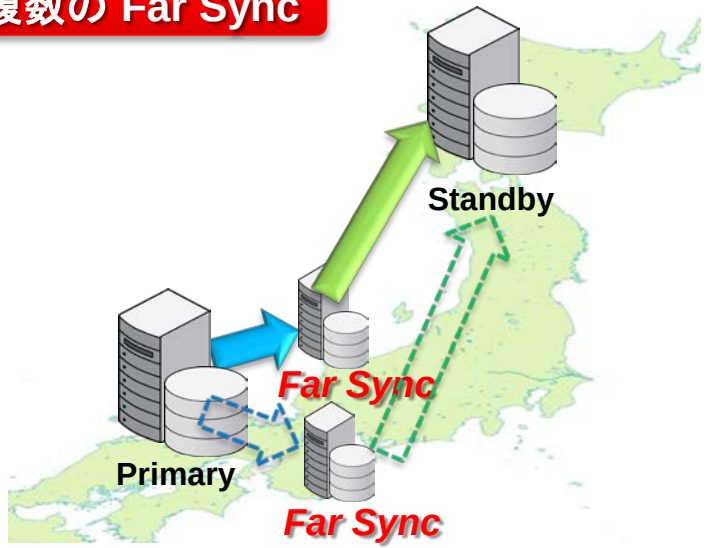


- 圧縮によるCPU負荷をオフロード
- 低帯域の効率的な使用が可能

遠隔同期インスタンス

4. 活用例：複数の遠隔同期インスタンス構成

複数の Far Sync



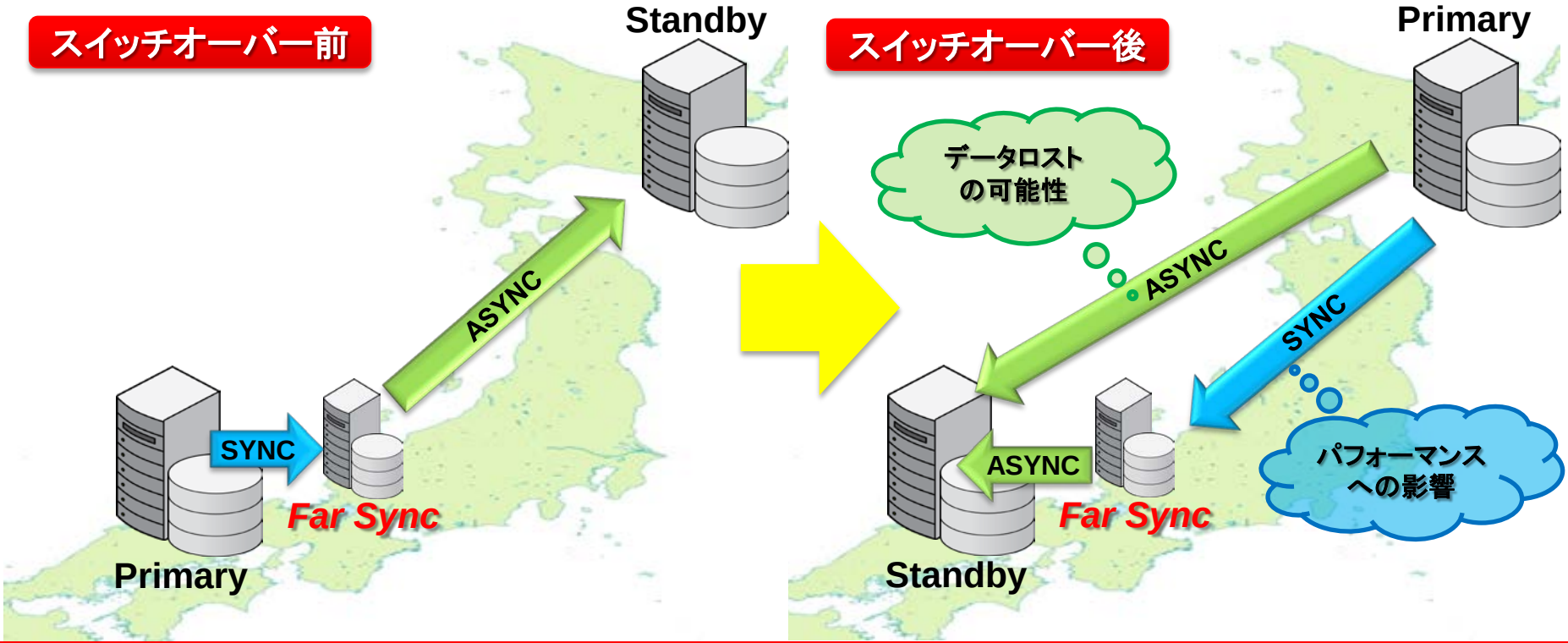
- 遠隔同期インスタンスの可用性向上
- 僅かなDISK領域で多重化可能

```
LOG_ARCHIVE_DEST_2=
'SERVICE=farsync1  SYNC  AFFIRM
 MAXFAILURE=1
 ALTERNATE=LOG_ARCHIVE_DEST_3
 VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)
 DB_UNIQUE_NAME=farsync'

LOG_ARCHIVE_DEST_STATE_2=ENABLE
```

```
LOG_ARCHIVE_DEST_3=
'SERVICE=farsync2  SYNC  AFFIRM
 ALTERNATE=LOG_ARCHIVE_DEST_2
 VALID_FOR=(ONLINE_LOGFILE, PRIMARY_ROLE)
 DB_UNIQUE_NAME=farsync2'

LOG_ARCHIVE_DEST_STATE_3=ALTERNATE
```

遠隔同期インスタンス

5. スイッチオーバー後を考慮した構成



遠隔同期インスタンス

5. スイッチオーバー後を考慮した構成

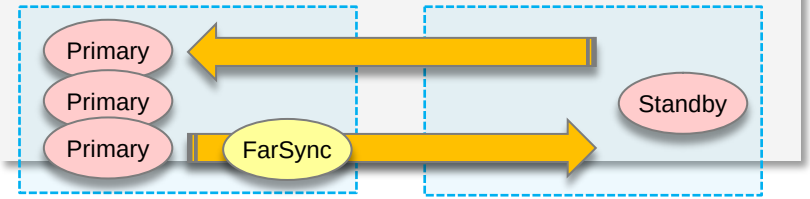
スイッチオーバー状態での運用を想定しない場合

想定ケース

- スイッチバックまで縮退稼働
- スタンバイはバックアップサイトとしての使用を想定

構成

- **プライマリ側のみ遠隔同期インスタンスを用意**
- ASYNC 転送　OR
  (パフォーマンス低下を前提とした) SYNC 転送



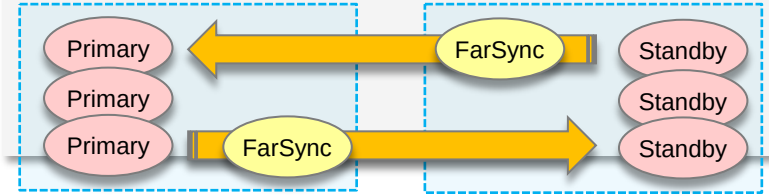
スイッチオーバー状態での運用を想定する場合

想定ケース

- スイッチオーバー前後で同様の稼働が必要
- 定期的なスイッチオーバー実施を前提とした運用

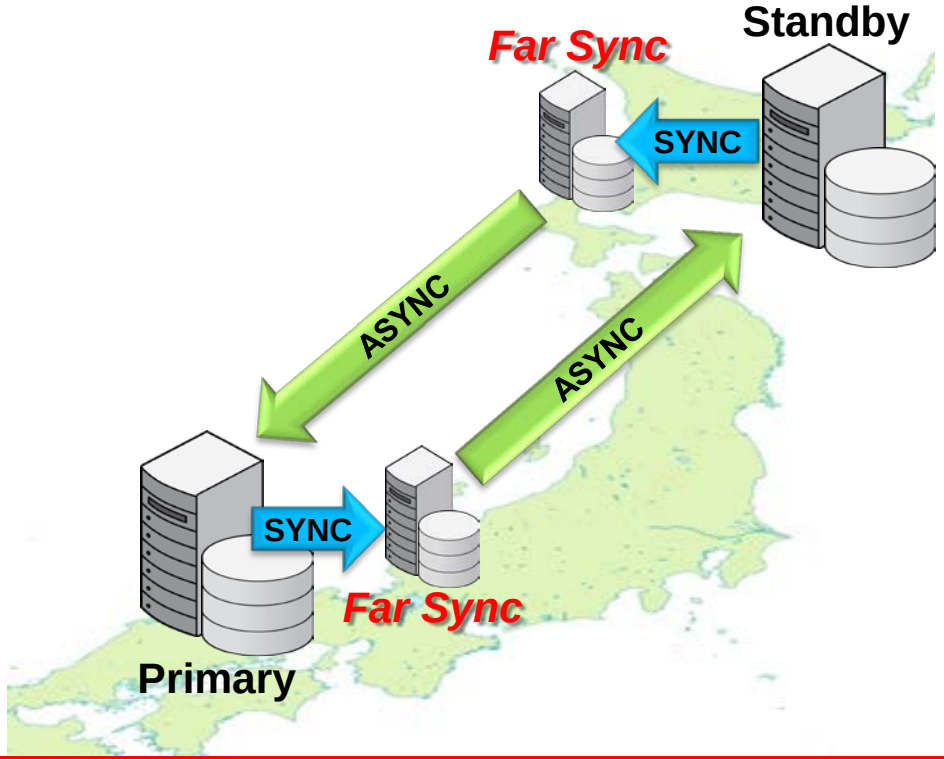
構成

- **プライマリ/スタンバイにそれぞれ遠隔同期インスタンスを用意**



遠隔同期インスタンス

6. スイッチオーバー後の遠隔同期インスタンスの利用



スイッチオーバー状態での運用を想定する場合

プライマリ側

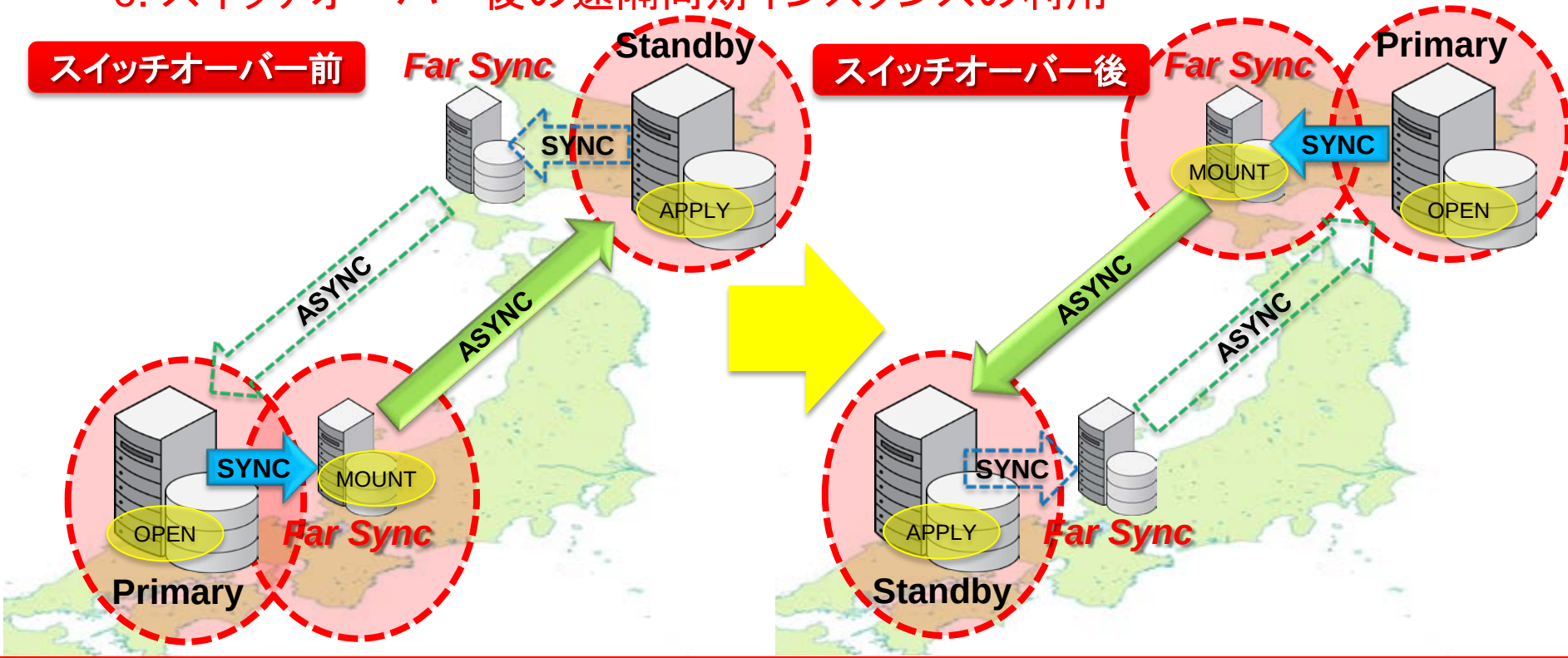
- 近隣に遠隔同期インスタンスを配置
- 遠隔同期インスタンスを経由し Standby にログ転送

スタンバイ側

- 近隣に遠隔同期インスタンスを配置
- ログ転送先をその遠隔同期インスタンスに設定
- 遠隔同期インスタンスは MOUNT 状態で運用
  - ✓ LOG_ARCHIVE_DEST_n パラメータの VALID_FOR 属性で転送動作を制御

遠隔同期インスタンス

6. スイッチオーバー後の遠隔同期インスタンスの利用



遠隔同期インスタンス

7. 制限

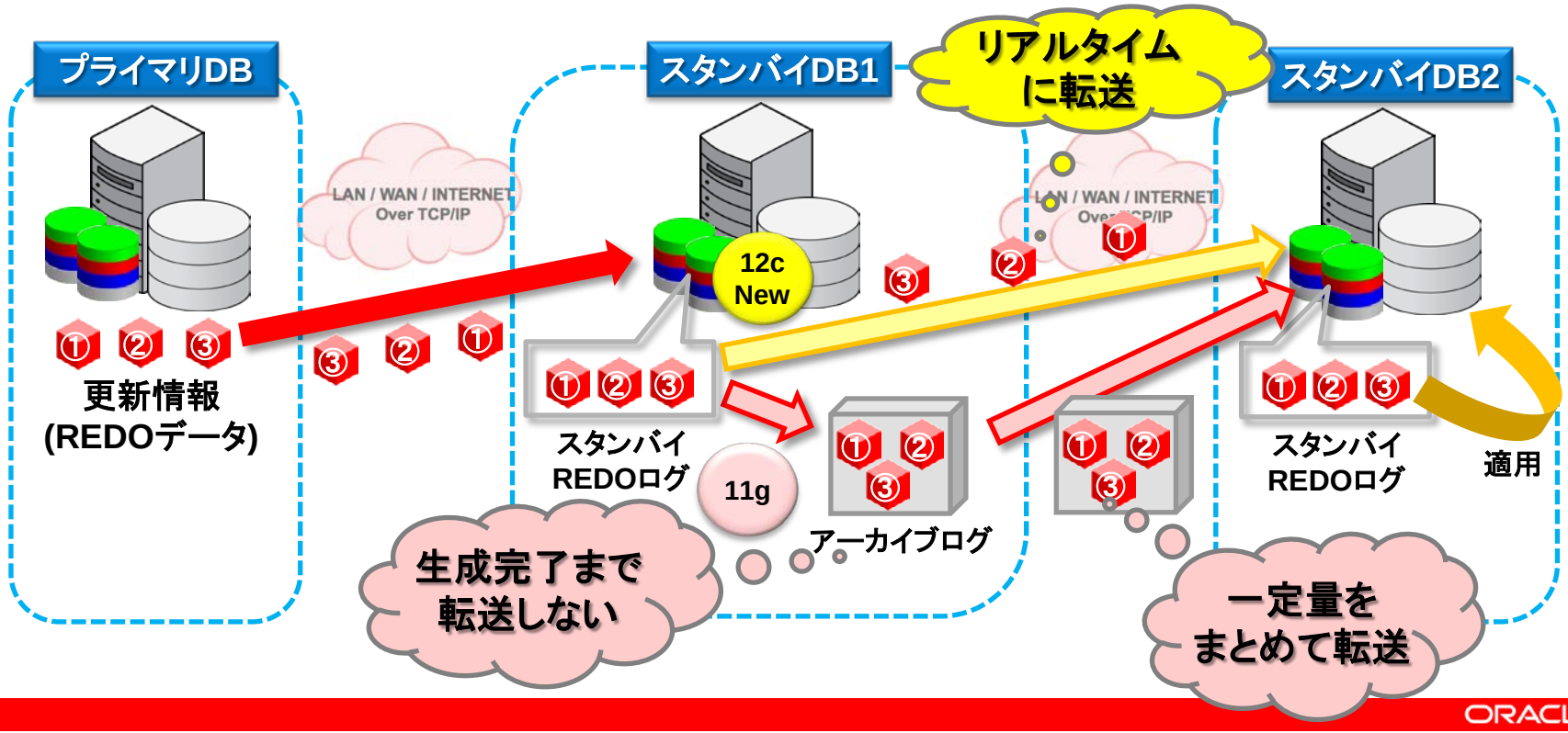
- 遠隔同期インスタンス環境での制限
  - 保護モード
    - 最大パフォーマンス or 最大可用性モードは使用可能
    - 最大保護モードは使用不可能

Data Guard 新機能：同期処理の高速化

- **リアルタイム・カスケード**
  - カスケード・スタンバイへリアルタイムに REDO ログを転送

リアルタイム・カスケード・スタンバイ

スタンバイREDOログのデータをリアルタイムに転送



リアルタイム・カスケード・スタンバイ

遠隔同期インスタンスとの違い

リアルタイム・カスケード

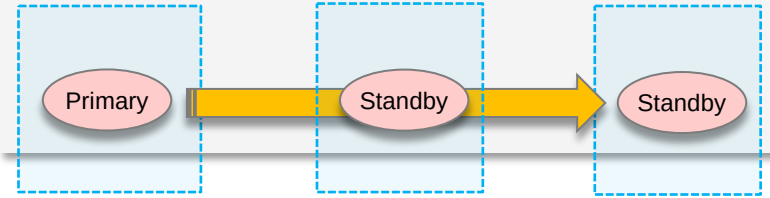
Positive

- 中間のスタンバイを DB として利用可能
  - ✓ 参照処理、バックアップ
- プライマリDB停止後のData Guard 構成が可能

Negative

- 中間のスタンバイDBのためのDISK領域が必要

◆ 2つのスタンバイDBの活用



遠隔同期インスタンス

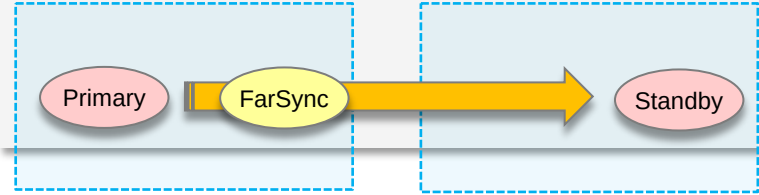
Positive

- Far Sync 構築のためのDISK領域が極小
- 複数台設置による可用性向上が容易

Negative

- Far Sync は REDO の送受信のみ実行
  - ✓ 参照処理、バックアップなどのユーザ処理不可

◆ 1つのスタンバイDBの活用



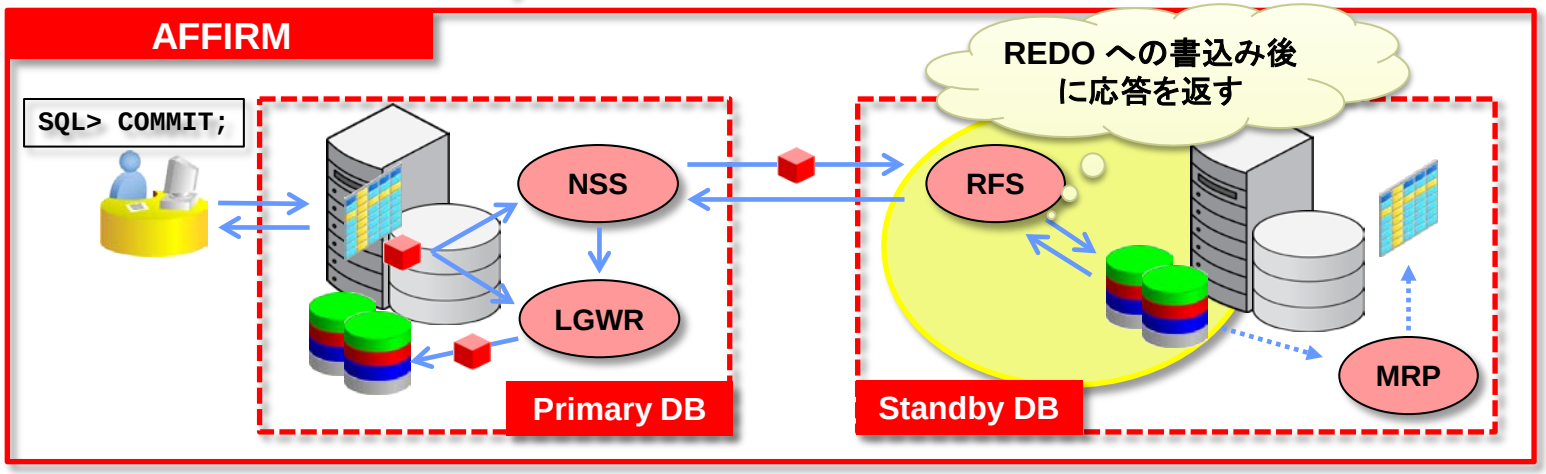
Data Guard 新機能：同期処理の高速化

- **FastSync**
  - 最大可用性モードでのパフォーマンス向上

FastSync

最大可用性モードにおける従来リリースの動作

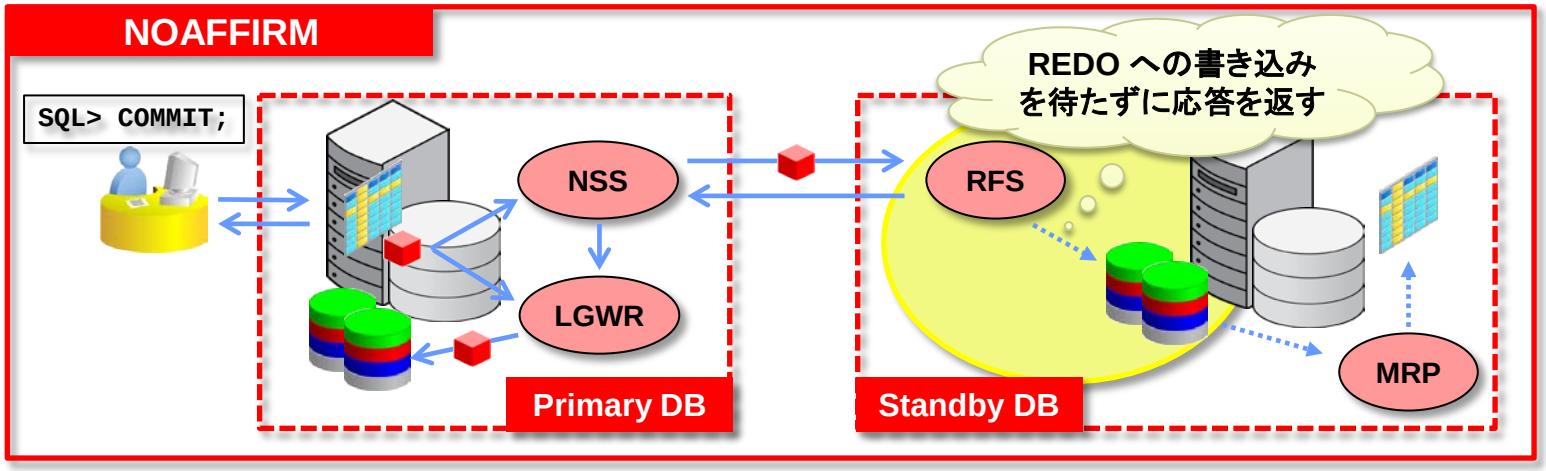
- 最大可用性モード (11gR2)
  - SYNC
  - AFFIRM が必須 ➡ スタンバイ REDO への I/O による影響



FastSync

最大可用性モードにおける12c での動作

- 最大可用性モード (12c)
  - SYNC
  - AFFIRM / **NOAFFIRM** の選択が可能

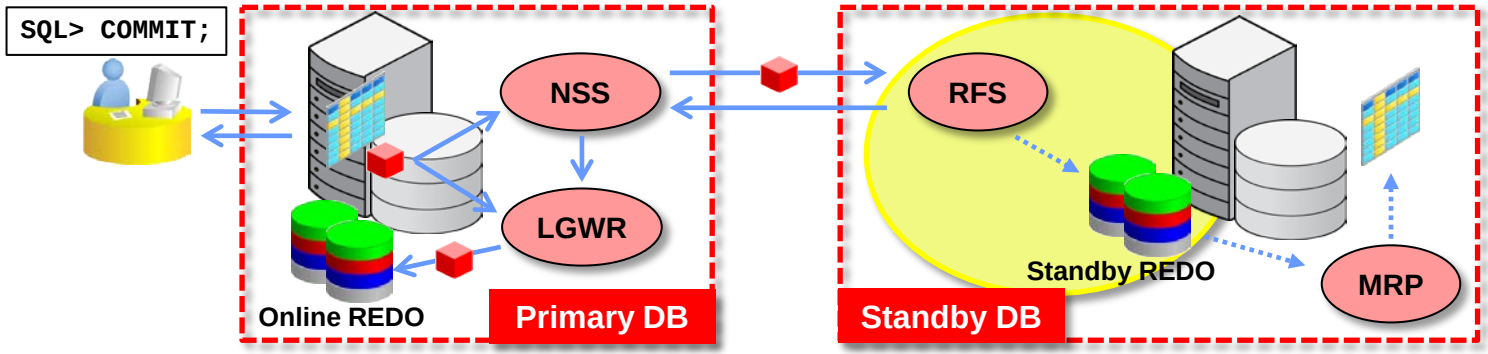


FastSync

AFFIRM / NOAFFIRM の違い

	AFFIRM	NOAFFIRM
プライマリへの応答	スタンバイ REDO への書き込みを待機	RFS による受信まで待機
データの保証	スタンバイ REDO への書き込み	RFS でのデータ保持
最大可用性モード	9i～	12c～
	データ保護重視	パフォーマンス重視

◆用途・目的に応じた選択が可能



2. スタンバイDBの活用

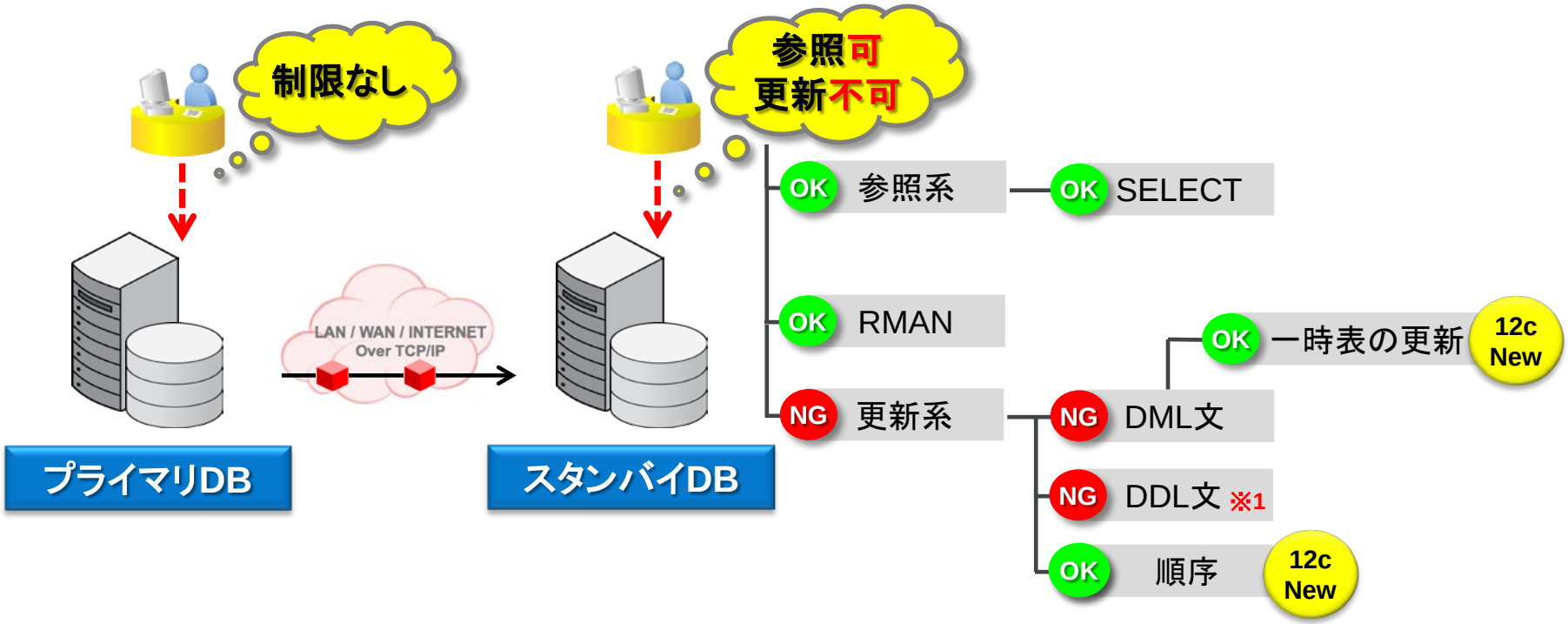
"Active Data Guard での実行可能な処理の増加"

Data Guard 新機能：スタンバイ DB の活用

- **一時表への DML のサポート**
  - スタンバイ DB で一時表への DML 文実行をサポート
- **順序のサポート**
  - スタンバイ DB で順序の使用をサポート

スタンバイDB の活用

Primary / Standby で実行可能な処理



※1 REDOログファイルの追加・削除、データファイル名変更など一部の処理は変更は実行可能

一時表への DML 文の実行

一時表とは

- トランザクションやセッションの存続中のみデータを保持する表
  - セッション固有の一時表とトランザクション(TX)固有の一時表
  - CREATE GLOBAL TEMPORARY TABLE文で作成
- DML 実行時の動作
  - REDOレコード : 生成されない
  - UNDOレコード : UNDO 表領域に格納される

	セッション固有の一時表	TX 固有の一時表
データの保持期間	セッション終了時まで	トランザクション終了時まで
COMMIT実行時の動作	データは保持される	データは削除される
他のセッションからの参照	不可	不可

一時表への DML 文の実行

一時 UNDO

- 一時UNDOとは
  - 一時表の UNDO を TEMP 表領域に格納
  - REDO の生成は行わない
  - Active Data Guard 環境では自動的に使用
    - ✓一時表の作成(CREATE文)は プライマリDB でのみ実行可能

REDO 生成量減少による
パフォーマンス向上

	一時 UNDO 未使用	一時 UNDO 使用
REDOレコード	生成されない	生成されない
UNDOレコード	UNDO 表領域に格納	TEMP 表領域に格納
使用方法	TEMP_UNDO_ENABLED=FALSE (デフォルト)	TEMP_UNDO_ENABLED=TRUE (ADG では設定不要)

一時表への DML 文の実行

スタンバイ DB での一時表の使用方法

- 一時表の作成 ・・・ プライマリDBで実行
  - トランザクション固有の一時表

```
create global temporary table emp_temp1(empno number,ename varchar2(10))
on commit delete rows;
```

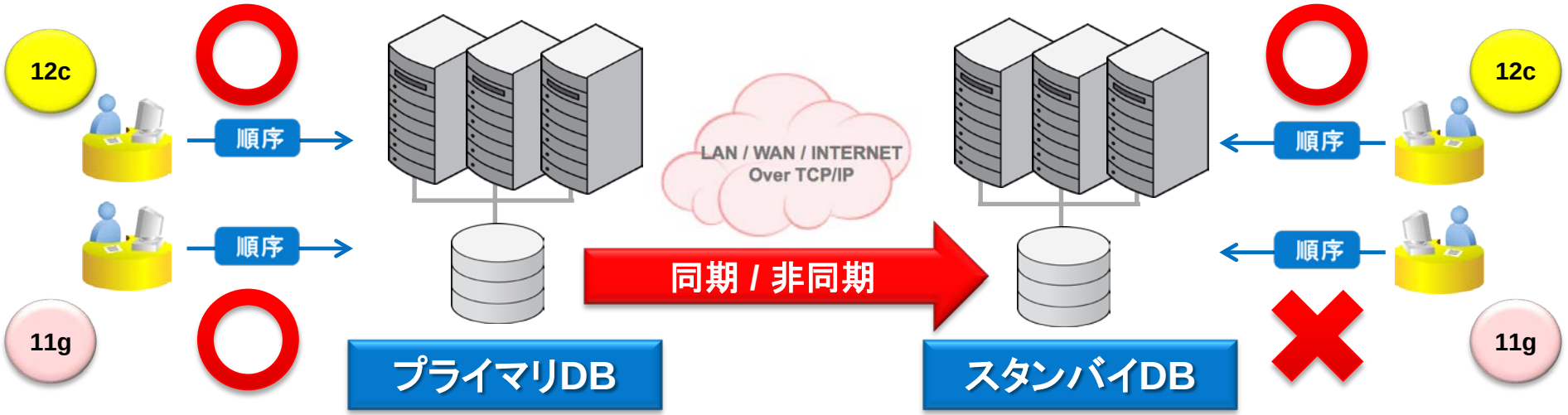
  - セッション固有の一時表

```
create global temporary table emp_temp2(empno number,ename varchar2(10))
on commit preserve rows;
```

- DML の実行・・・ スタンバイ DB で実行
  - 一時表に対して INSERT / DELETE / UPDATE を実行
  - プライマリ DB でも実行可能

順序の使用

スタンバイ DB での順序の使用のサポート



順序の使用

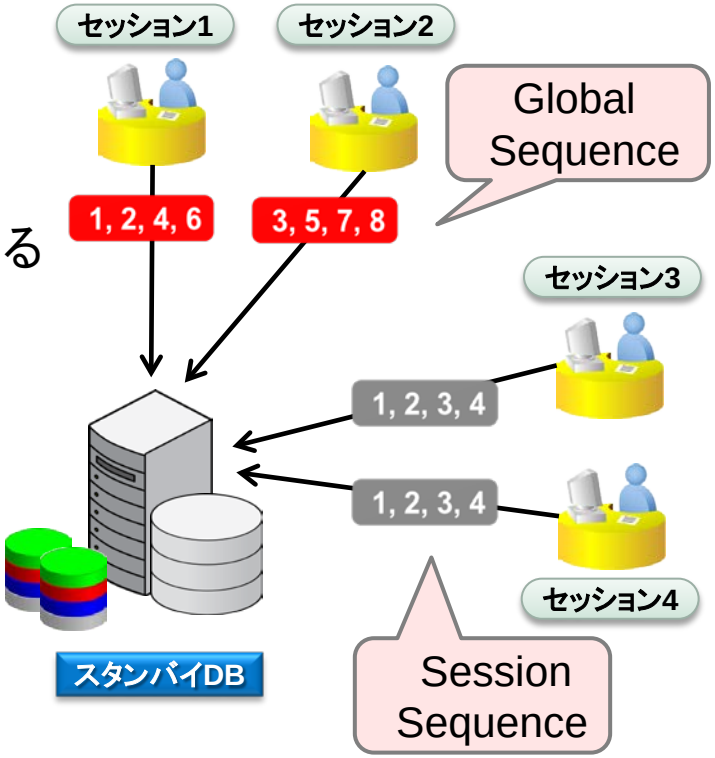
順序の種類

- **Global Sequence（グローバル順序）**
  - DB 内で共有して使用される順序
  - 値の増減は他のセッションの処理にも影響する
  - 12c 以前で使用されていた順序

```
create sequence … [GLOBAL];
```

- **Session Sequence（セッション順序）**
  - セッション内でのみ有効な順序
  - 値の増減はそのセッション内でのみ有効
  - 12c から実装された順序

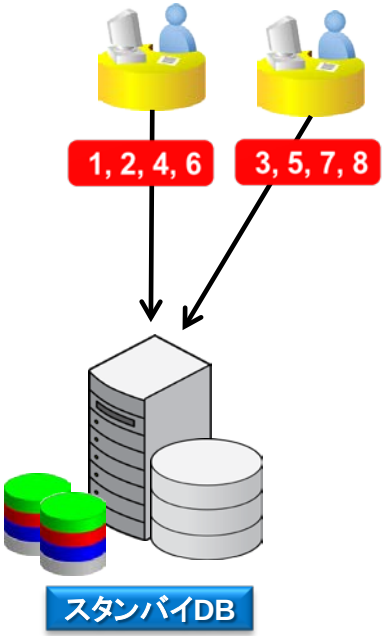
```
create sequence … SESSION;
```



順序の使用

Global Sequence

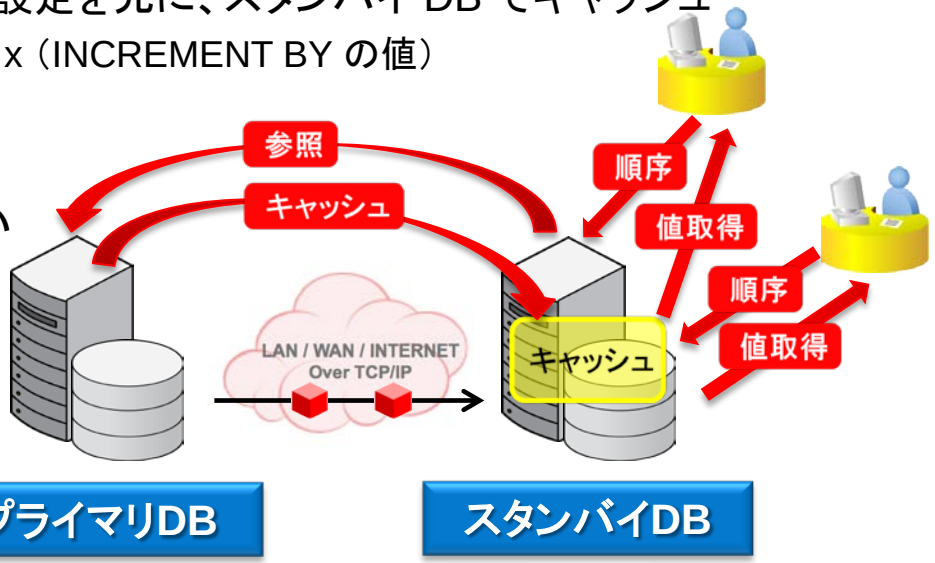
- **DB内で値を共有して使用する順序**
  - 12c 以前で使用されていたものと同じ
    - ✓ プライマリ DB では 12c 以前でも使用可能
    - ✓ スタンバイ DB では 12c 以前では使用不可
- **作成方法**
  - スタンバイ DB では作成不可、プライマリ DB での作成必須
- **スタンバイ DB での使用時における制限**
  - プライマリ DB の起動が必須
  - CACHE 及び NOORDER が必須
    - デフォルトは CACHE 20、NOORDER
    - NOCACHE または ORDER の順序は使用不可



順序の使用

Global Sequence

- **スタンバイ DB での順序使用時の動作**
  - スタンバイ DB での順序参照時にプライマリ DB に接続し、順序の割り当て状況を確認
  - 現在までに割り当て済みの値と順序の設定を元に、スタンバイ DB でキャッシュ
    - （CACHE する値の数）＝（CACHE の値）x（INCREMENT BY の値）
      - CACHE のデフォルト : 20
      - INCREMENT BY のデフォルト : 1
  - プライマリ DB 停止時には使用できない
    - 既にキャッシュされている場合、その値は使用可能

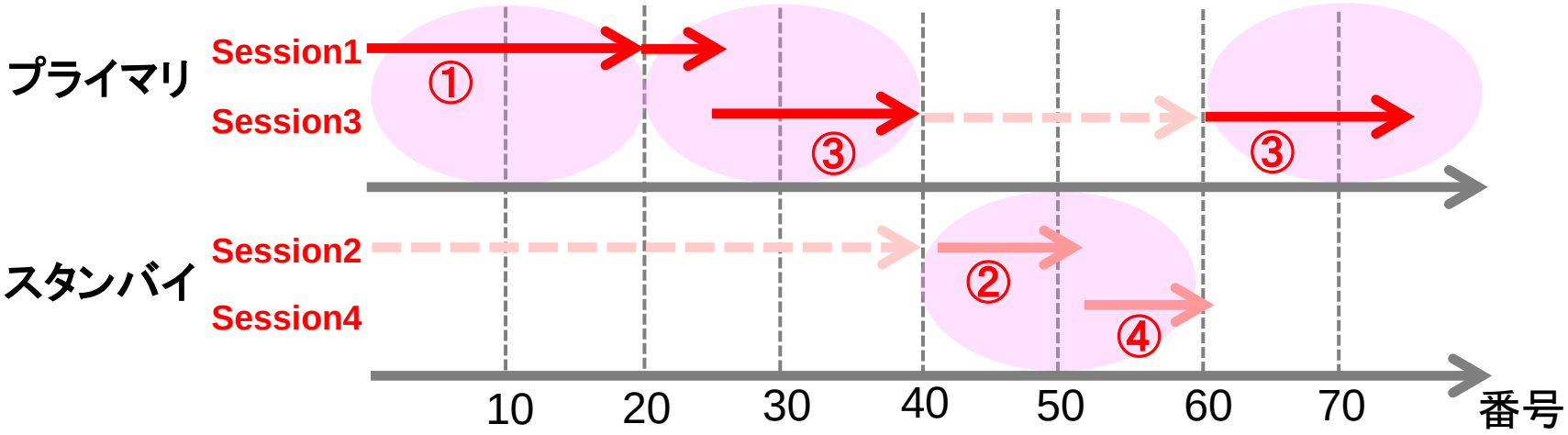


順序の使用

Global Sequence

- プライマリ / スタンバイ DB での順序の割り当て

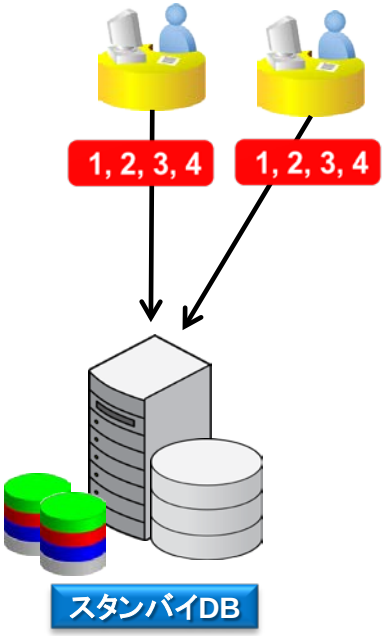
（例）CACHE 20、START WITH 1、INCREMENTAL BY 1 の場合



順序の使用

Session Sequence

- **単一セッション内でのみ有効な 順序**
  - 値のカウントは各セッション毎に行われる
  - 12c より新たに実装
- **作成方法**
  - スタンバイ DB では作成不可、プライマリ DB での作成必須
- **スタンバイ DB での使用時における制限**
  - なし
    - プライマリ DB 停止時にも使用可能
    - NOCACHE、ORDER も使用可能

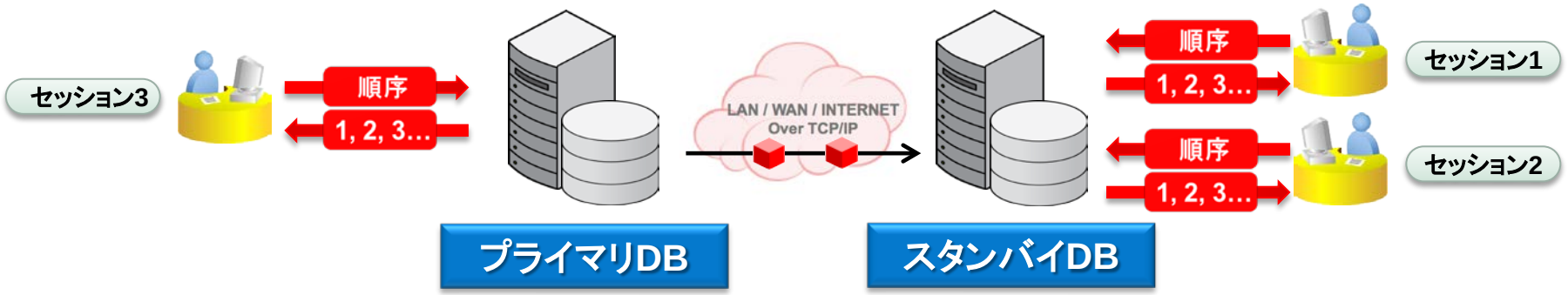


順序の使用

Session Sequence

スタンバイ DB での順序使用時の動作

- セッション毎に個別に値を割り当て
  - セッション1で使用中の順序の値は、セッション2での処理の影響を受けない
  - プライマリ DB のセッションの処理にも影響を受けない
- 順序使用時には START WITH で指定した値から採番される
  - START WITH が未指定の場合、MINVALUE または INCREMENT BY から算出される

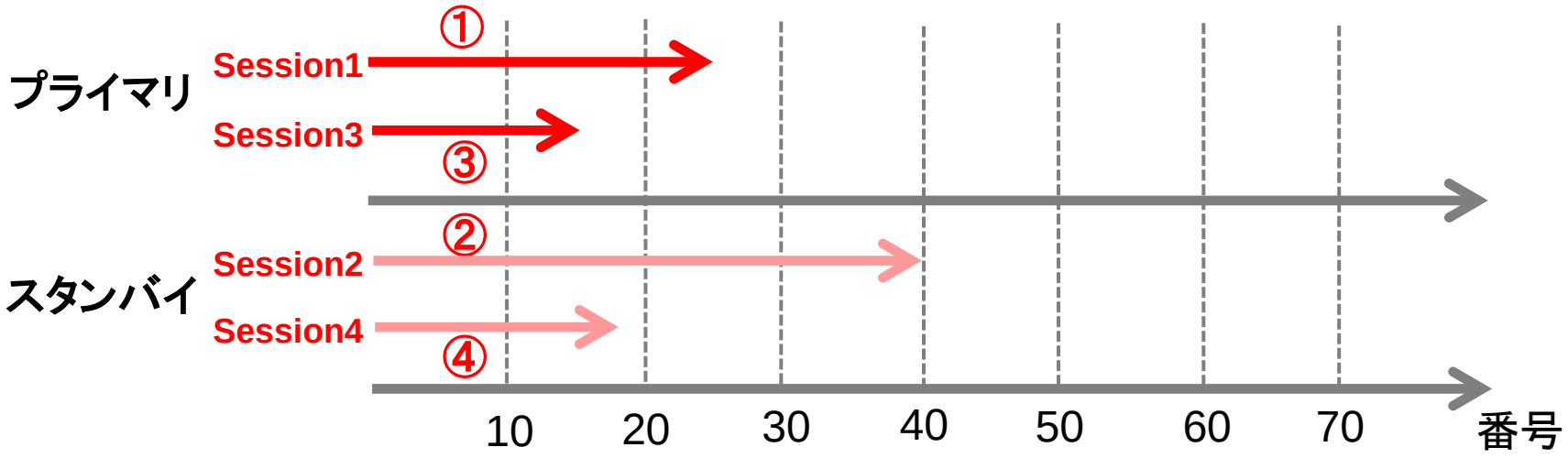


順序の使用

Session Sequence

- プライマリ / スタンバイ DB での順序の割り当て

（例）CACHE 20、START WITH 1、INCREMENTAL BY 1 の場合



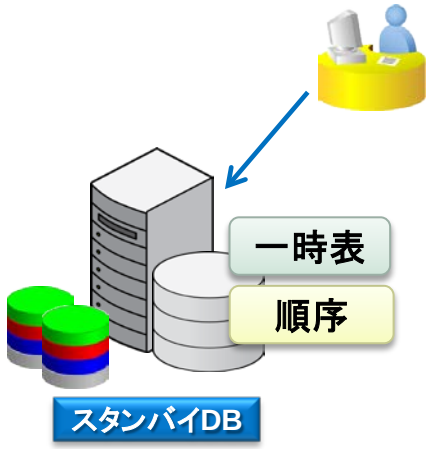
一時表・順序の使用用途

スタンバイ DB での利用例

- **一時表**
  - レポート結果を算出するための一時的なデータ格納に使用
  - 未確定のデータを暫定的に保持し、確定したデータを永続表に格納

- **Global Sequence**
  - レポート結果にユニークな値を付与して出力

- **Session Sequence**
  - 一時表に格納するデータの採番に使用



3. 操作の簡略化

"運用・管理をより簡単に"

Data Guard 新機能：操作の簡略化

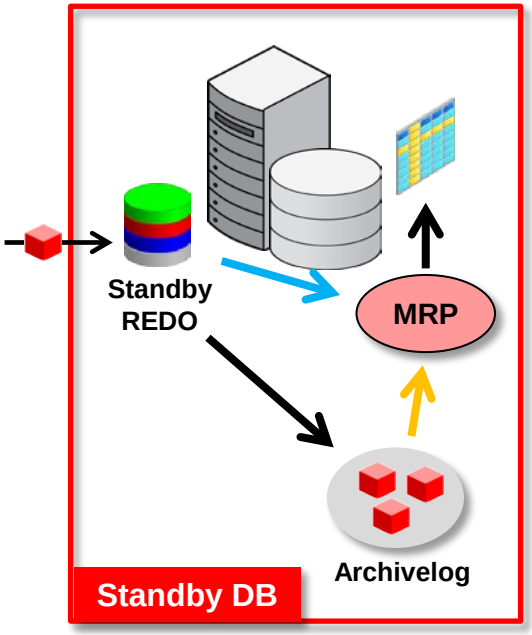
- **リアルタイム適用**
  - ログの適用モードのデフォルトをリアルタイム適用に変更
- **ロール変換**
  - スイッチオーバー・フェイルオーバーのコマンド簡略化

Data Guard 新機能：操作の簡略化

- **リアルタイム適用**
  - ログの適用モードのデフォルトをリアルタイム適用に変更

リアルタイム適用のデフォルト化

リアルタイム適用開始コマンドの変更



リアルタイム適用

11g
```
recover managed standby database using current logfile disconnect;
```

12c
```
recover managed standby database disconnect;
```

アーカイブログ適用

11g
```
recover managed standby database disconnect;
```

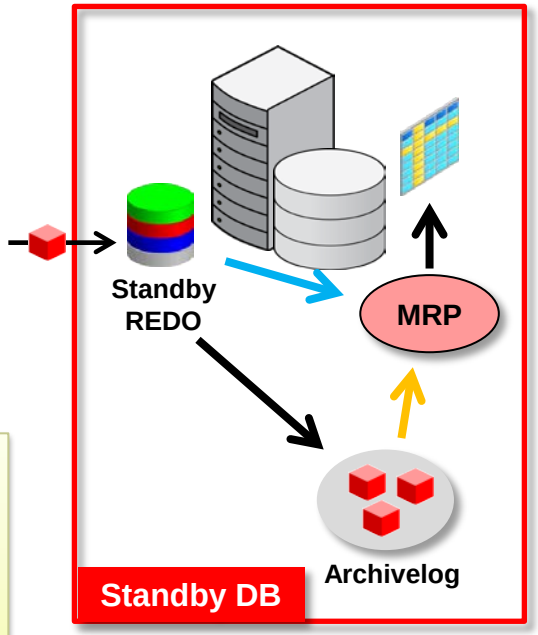
12c
```
recover managed standby database using archived logfile disconnect;
```

リアルタイム適用のデフォルト化

リアルタイム適用での注意点

- リアルタイム適用での注意点
  - 必須要件
    - スタンバイ REDO ログファイルが作成されている
  - アーカイブログ適用が必須のケース
    - タイム・ディレイ適用使用時

◆ タイム・ディレイ適用
- スタンバイ DB でのログ適用タイミングを遅らせるモード
- プライマリ DB の log_archive_dest_n に DELAY=xx (単位:分)を指定することで有効となる

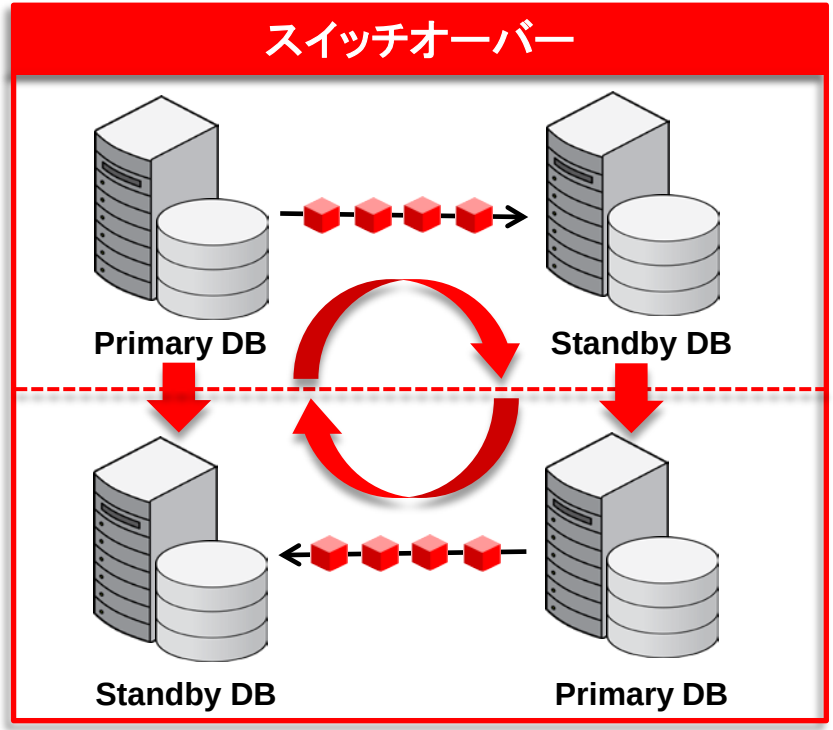


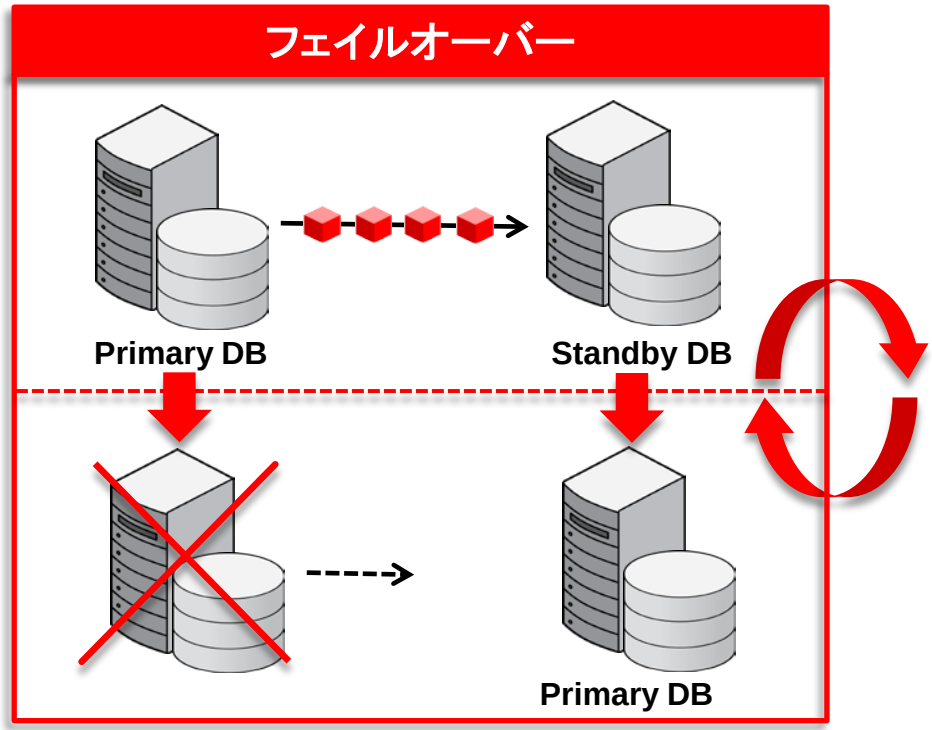
Data Guard 新機能：操作の簡略化

- **ロール変換**
  - スイッチオーバー・フェイルオーバーのコマンド簡略化

ロール変換コマンドの簡略化

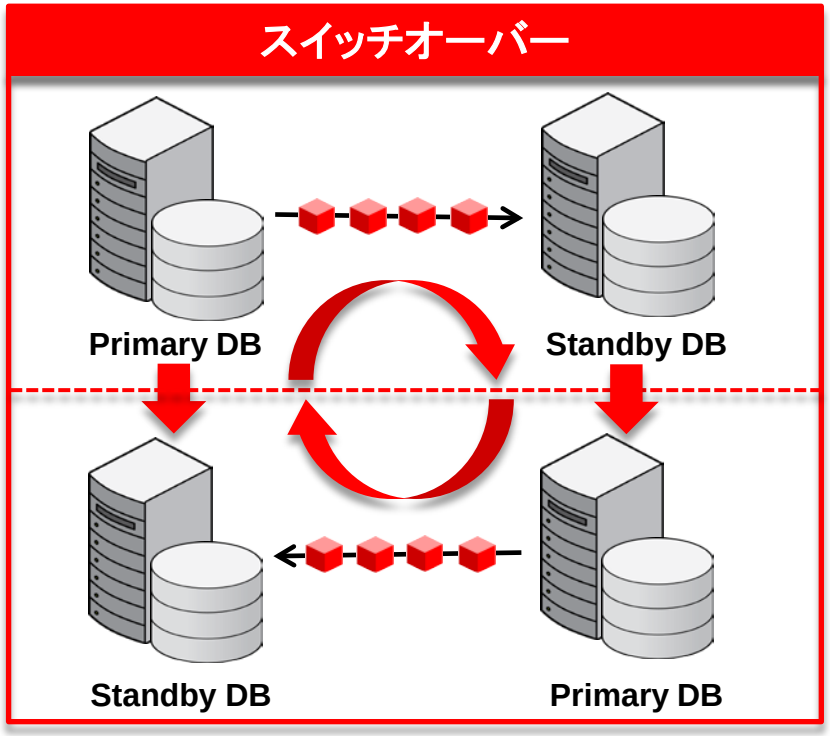
スイッチオーバー、フェイルオーバーのコマンド簡略化





ロール変換コマンドの簡略化

スイッチオーバーのコマンド簡略化



- スイッチオーバー可能かどうかの確認

```
alter database switchover to <standby> verify;
```

- スイッチオーバーの実行

```
alter database switchover to <standby>;
```

※ <standby> はスタンバイ DB の db_unique_name を指定

ロール変換コマンドの簡略化

スイッチオーバーコマンドの差異

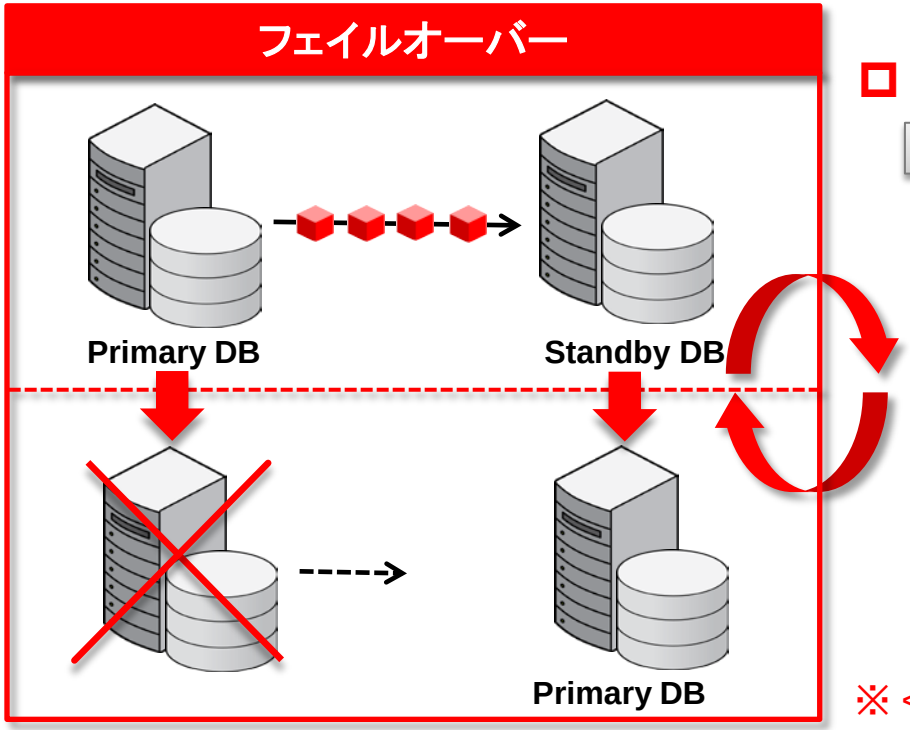
	P/S	11gR2	12c
1. ステータス確認	P	select switchover_status from v$database;	**alter database switchover to <standby> verify;**
2. ロール変換	P	alter database commit to switchover to physical standby with session shutdown;	**alter database switchover to <standby> ;**
3. 新スタンバイ起動	P	shutdown abort startup mount	**startup mount;**
4. ステータス確認	S	select switchover_status from v$database;	
5. ロール変換	S	alter database commit to switchover to primary with session shutdown;	
6. 新プライマリ起動	S	alter database open;	**alter database open;**
7. Redo Apply 開始	P	recover managed standby database using current logfile disconnect;	**recover managed standby database disconnect;**

12c スタンバイデータベースからのコマンドの実行が不要

※ <standby> にはスタンバイDBの db_unique_name を指定

ロール変換コマンドの簡略化

フェイルオーバーのコマンド簡略化



- フェイルオーバーの実行

```
alter database failover to <standby>;
```

※ <standby> はスタンバイDBの db_unique_name を指定

ロール変換コマンドの簡略化

フェイルオーバーコマンドの差異

	11gR2	12c
1. Redo Apply 停止	recover managed standby database cancel;	recover managed standby database cancel;
2. 未適用 REDO の適用	recover managed standby database finish;	alter database failover to <standby> ;
3. ステータス確認	select switchover_status from v$database;	
4. ロール変換	alter database commit to switchover to primary with session shutdown;	
5. 新プライマリ起動	alter database open;	alter database open;

12c 必要ステップ数の削減

※ <standby> にはスタンバイDBの db_unique_name を指定

ロール変換コマンドの簡略化

注意点・留意点

- 11gR2 での構文を 12c でも使用可能
  - 12c では従来の構文、新しい構文の両方が使用可能

- 従来の構文でのデフォルト動作の変更

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY;
```

  - 11g
    - 自動的なセッションの切断を行わない WITHOUT SESSION SHUTDOWN がデフォルト
  - 12c
    - 自動的なセッションの切断を行う WITH SESSION SHUTDOWN がデフォルト

4. 12cの新機能への対応

"新たに実装された機能を
Data Guard 構成でも使用可能に"

Data Guard 新機能：12c の機能への対応

- **マルチテナント・アーキテクチャ**
  - プラガブル・データベース構成での Data Guard をサポート
- **オンラインでのデータファイルの移動**
  - 表領域を ONLINE のまま、データファイルを任意のパスに移動可能

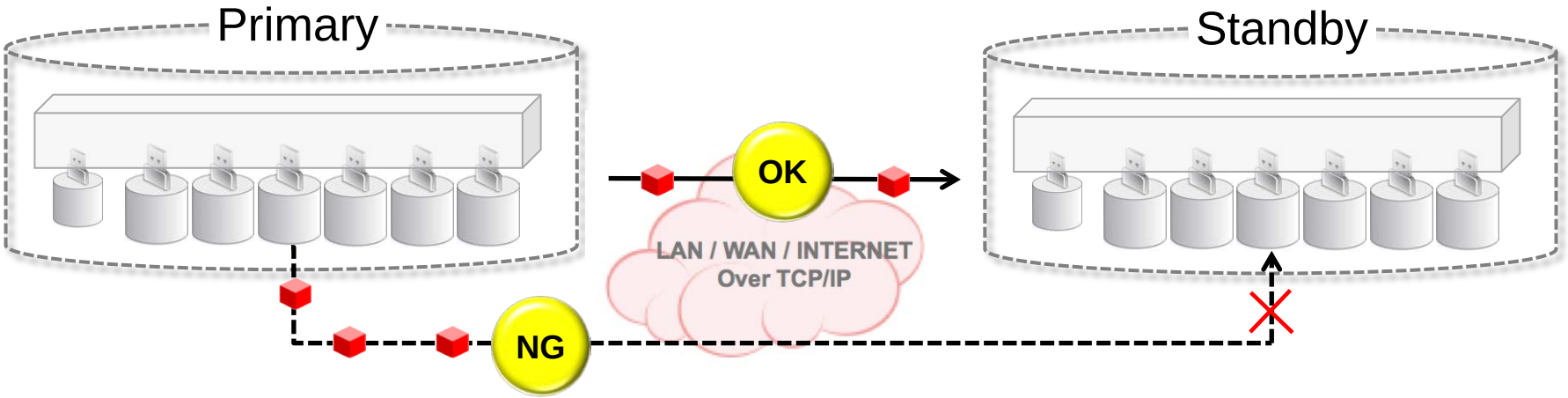
Data Guard 新機能：12c の機能への対応

- **マルチテナント・アーキテクチャ**
  - プラガブル・データベース構成での Data Guard をサポート

マルチテナント・アーキテクチャへの対応

Data Guard 構成の作成

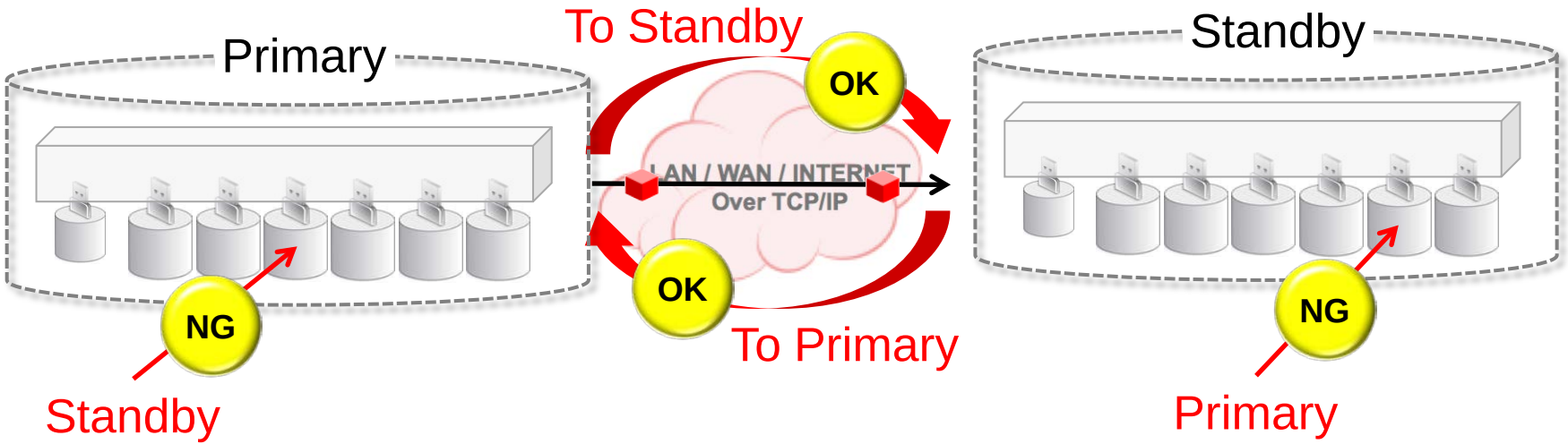
- CDB 単位でスタンバイ・データベースの作成が可能
  - フィジカル・スタンバイ、ロジカル・スタンバイのいずれも使用可能
  - PDB 単位でのスタンバイ・データベースの作成は不可



マルチテナント・アーキテクチャへの対応

ロール変換

- ロールの設定・変換処理は CDB 単位で設定
  - フェイルオーバー
  - スイッチオーバー

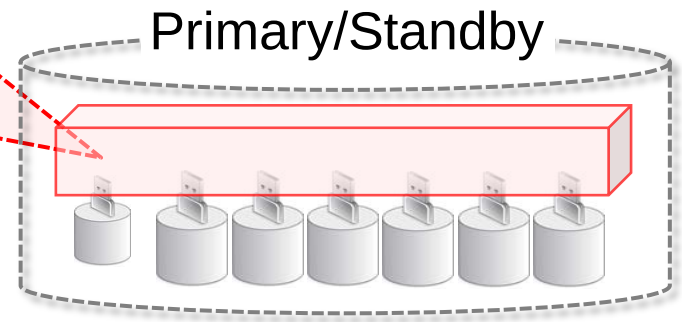


マルチテナント・アーキテクチャへの対応

管理コマンド

- Root (CDB$ROOT) で実行が必要

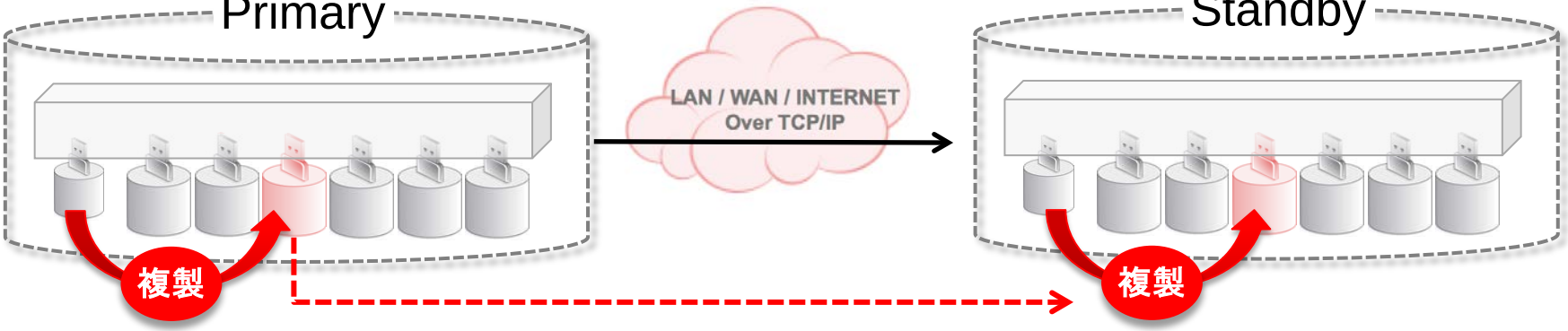
- Redo Apply の開始・停止
- スイッチオーバー
- フェイルオーバー
- 保護モードの変更
- Guard レベルの変更
- ログスイッチ
  - ✓ alter system switch logfile
  - ✓ alter system archive log current



マルチテナント・アーキテクチャへの対応

PDB の追加・削除

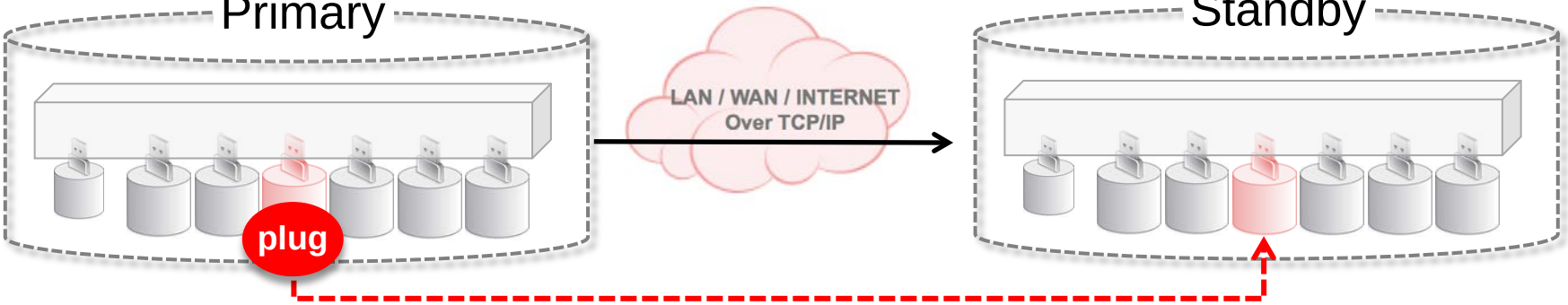
- PDB の追加
  - プライマリ DB での PDB の追加は、スタンバイ DB にも反映される
    - クローンによる作成では、スタンバイDB側のファイルより自動的に作成
    - Plug 時にはスタンバイ DB への手動でのファイルのコピーが必要
  - 作成した PDB の OPEN は、プライマリ DB 側を先に実行する必要がある



マルチテナント・アーキテクチャへの対応

PDB の追加・削除

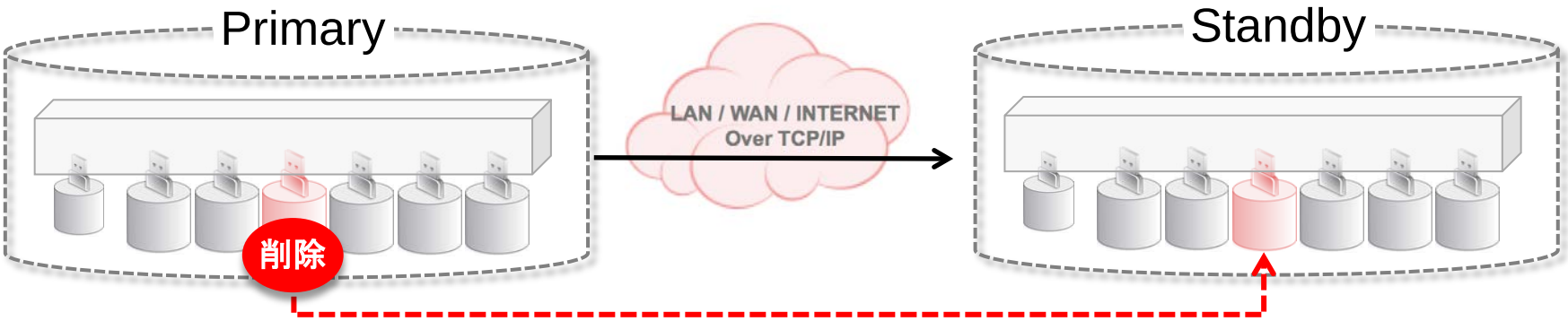
- PDB の追加
  - プライマリ DB での PDB の追加は、スタンバイ DB にも反映される
    - クローンによる作成では、スタンバイDB側のファイルより自動的に作成
    - Plug 時にはスタンバイ DB への手動でのファイルのコピーが必要
  - 作成した PDB の OPEN は、プライマリ DB 側を先に実行する必要がある



マルチテナント・アーキテクチャへの対応

PDB の追加・削除

- PDB の削除
  - プライマリ DB での処理はスタンバイ DB にも反映される
  - スタンバイDB側では PDB を手動で CLOSE しておく必要がある
    - OPEN 状態ではログ適用が停止する
    - CLOSE した後にログ適用を開始することで対処可能



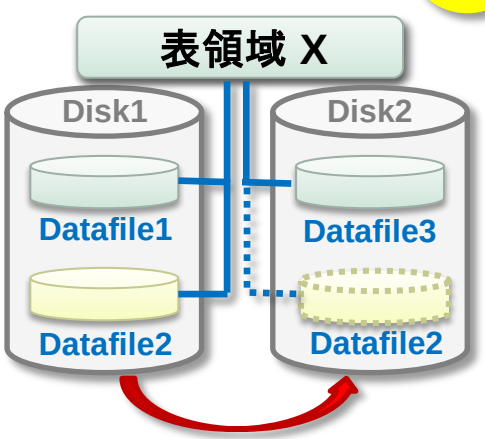
Data Guard 新機能：12c の機能への対応

- **オンラインでのデータファイルの移動**
  - 表領域を ONLINE のまま、データファイルを任意のパスに移動可能

オンラインでのデータファイルの移動

MOVE DATAFILE コマンドによるファイルの移動

- 11g でのデータファイルの MOVE/RENAME



実行手順

1. 表領域を OFFLINE に変更
2. ファイルのコピー/移動
3. ALTER DATABASE 文での DB 上のファイルパス変更
4. 表領域を ONLINE に変更

特徴・留意点

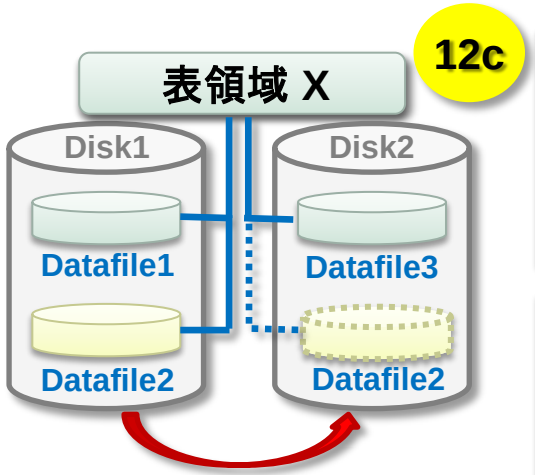
- 表領域の OFFLINE 化が必要
- 手動でのファイルのコピー、移動が必要

オンラインでのデータファイルの移動

MOVE DATAFILE コマンドによるファイルの移動

- 12c でのデータファイルの MOVE/RENAME

```
ALTER DATABASE MOVE DATAFILE '<移動前のパス>' TO '<移動後のパス>';
```



オンライン実行

- DB が ONLINE の状態で実行可能
- 表領域の OFFLINE 化不要
- コマンド実行中の更新も移動先に反映

ファイルの自動コピー

- 物理ファイルのコピーも実行
- 手動でのファイルのコピー、移動は不要
- ASM にも対応

オンラインでのデータファイルの移動

デフォルト動作・オプションと注意点

- デフォルト動作
  - 移動先の同名ファイルは上書きしない
    - REUSE オプションで上書き動作

```
ALTER DATABASE MOVE DATAFILE '<移動前のパス>' TO '<移動後のパス>' REUSE;
```

  - 移動前のファイルは自動的に削除
    - KEEP オプションでファイルを保持

```
ALTER DATABASE MOVE DATAFILE '<移動前のパス>' TO '<移動後のパス>' KEEP;
```

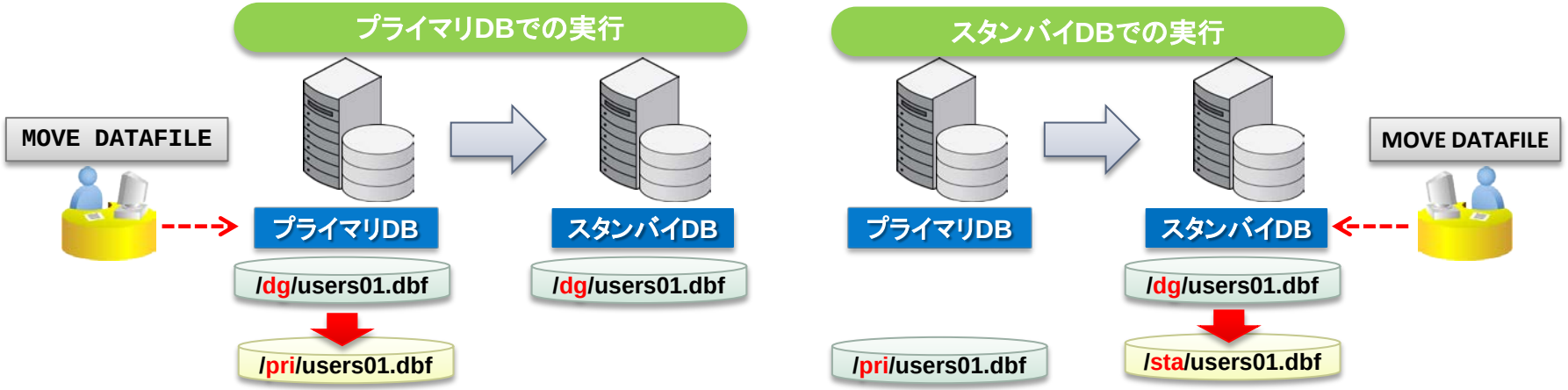
- 注意点
  - データベースが OPEN 状態でのみ実行可
  - コマンド実行中は最大で (移動するファイルサイズ x 2) の DISK 領域が必要

オンラインでのデータファイルの移動

Data Guard 構成での利用

- Data Guard 構成での利用
  - プライマリ DB での変更は伝播しない
  - スタンバイ DB での実行も可能
    - 変更はプライマリ DB に伝播しない

自由なファイルパスの構成
- プライマリ・スタンバイで独自構成が可能



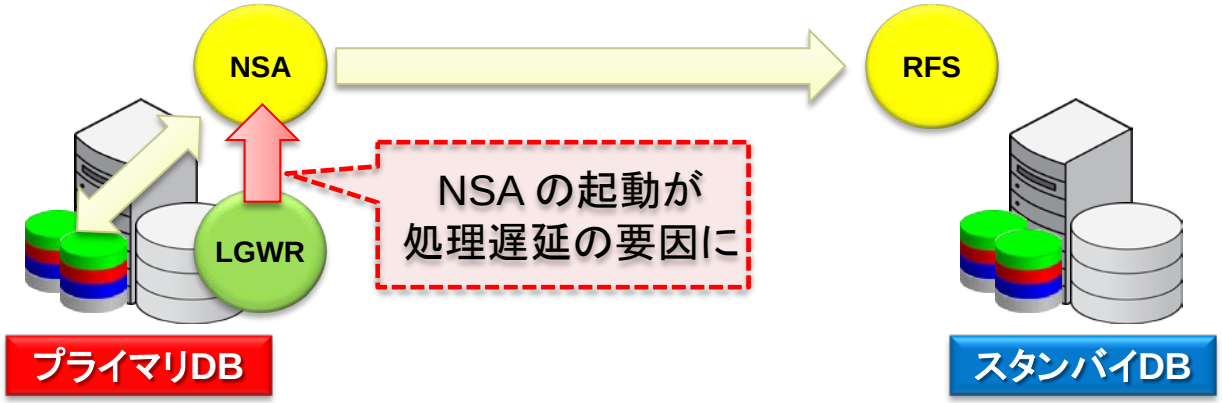
5. その他

"Data Guard 環境で起動するプロセスの変更点"

プロセスの変更

非同期転送の課題

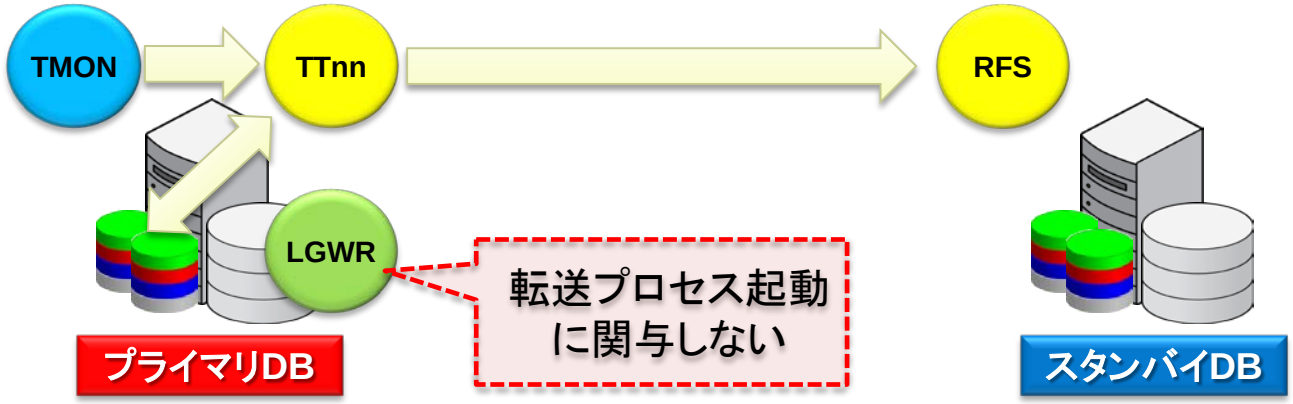
- 非同期転送 (11g)
  - プライマリ DB への影響を極小化した転送
  - ただし、ログ転送を行う NSA プロセスの起動は LGWR が実行するため、プロセス起動時には数秒間 LGWR の処理が滞る



プロセスの変更

12c での非同期転送

- 非同期転送 (12c)
  - ログ転送は TTnn (TT00 など)が実行
  - TTnn の起動は TMON (Transport Monitor)が実行
    - LGWR は関与しない



Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®