
ORACLE®

ORACLE

Oracle Database 12*c* Release 1

Global Data Services

日本オラクル株式会社
井上 克己

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。

アジェンダ

1. 複製データを活用している環境での現状
2. Global Data Services と RAC
3. 想定されるユース・ケース
4. グローバル・サービス
5. ロード・バランシング

GDS実装の背景

※ GDS … Global Data Services

コンソリデーション vs ディストリビューション

マルチテナント・アーキテクチャー

コンソリデーションの動機

- *管理の容易さ*
- *コスト削減*

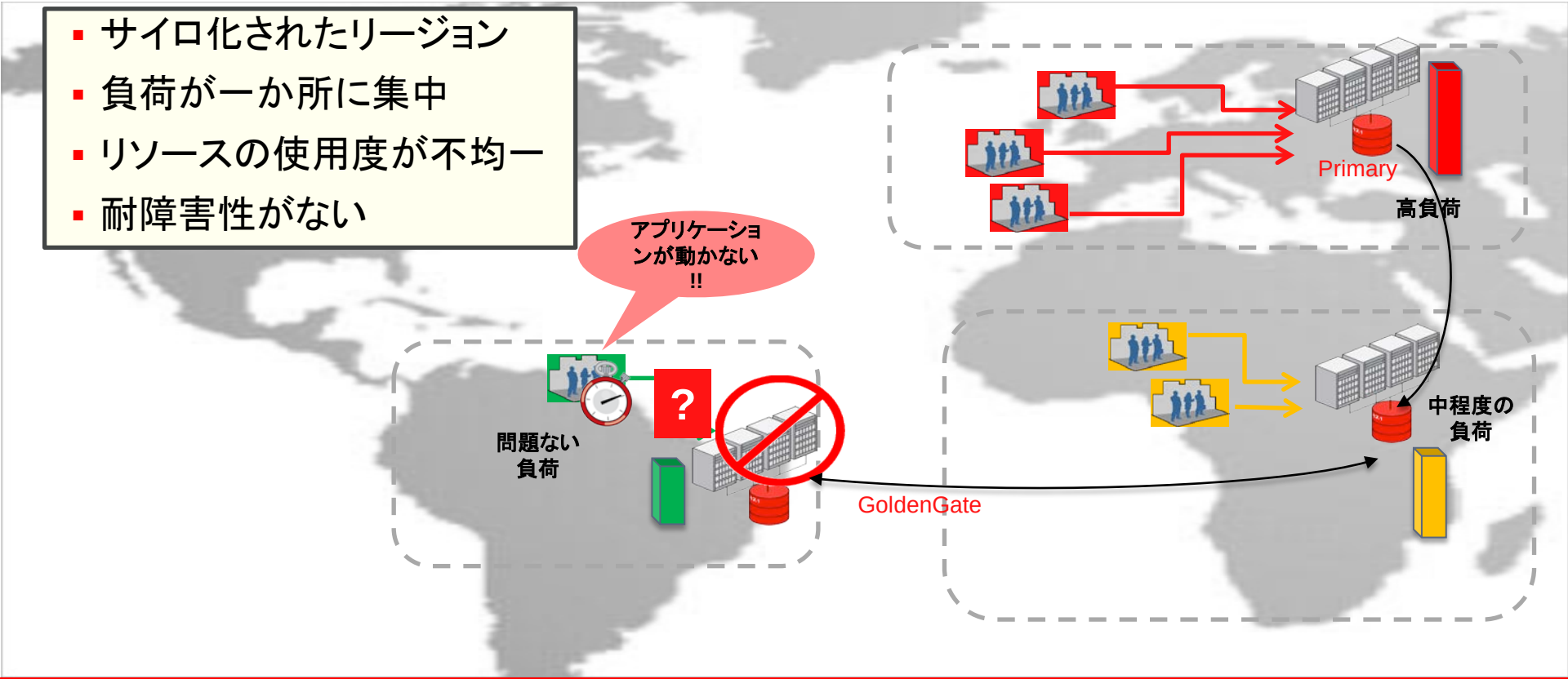
GDS

ディストリビューションの動機

- *DR ディザスタリカバリー*
- *ロードバランシング/ パフォーマンス*

複製データを活用している環境での問題

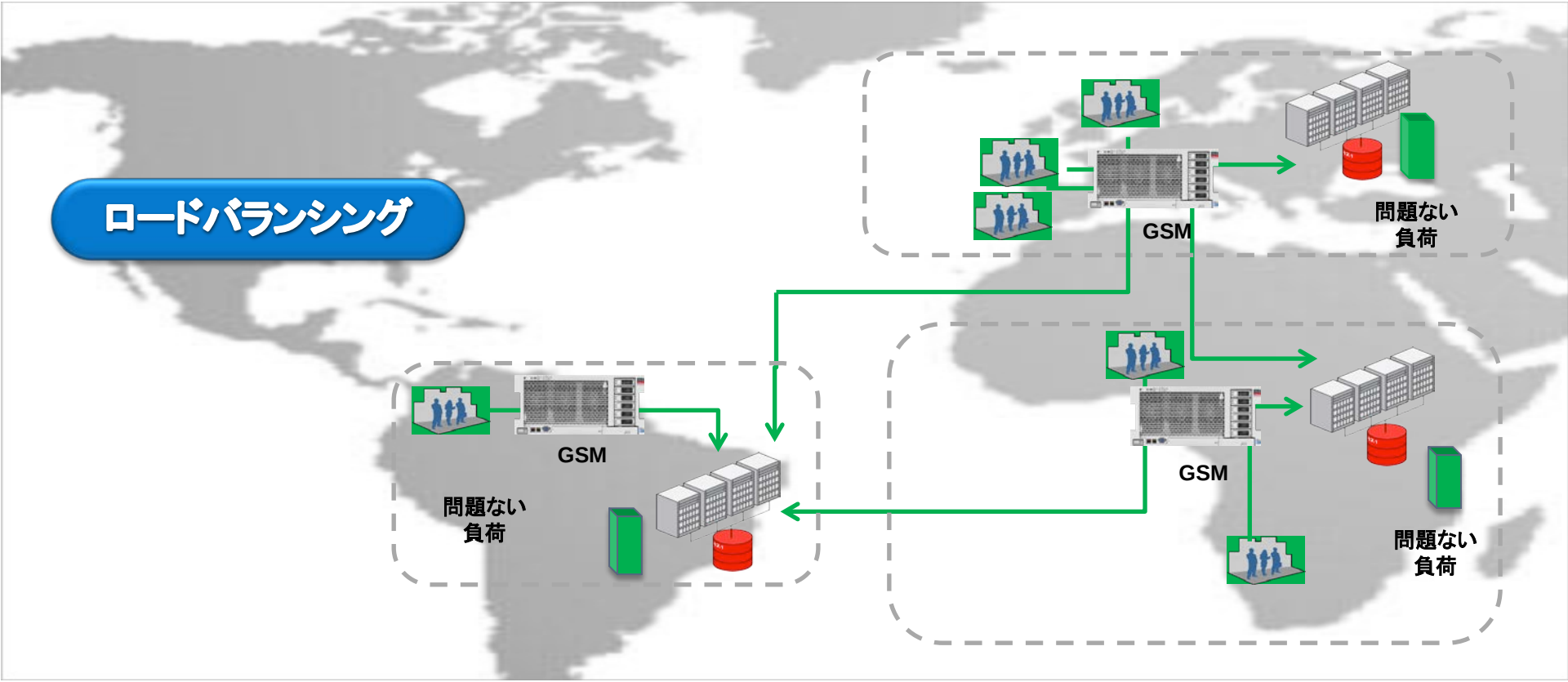
- サイロ化されたリージョン
- 負荷が一か所に集中
- リソースの使用度が不均一
- 耐障害性がない



アジェンダ

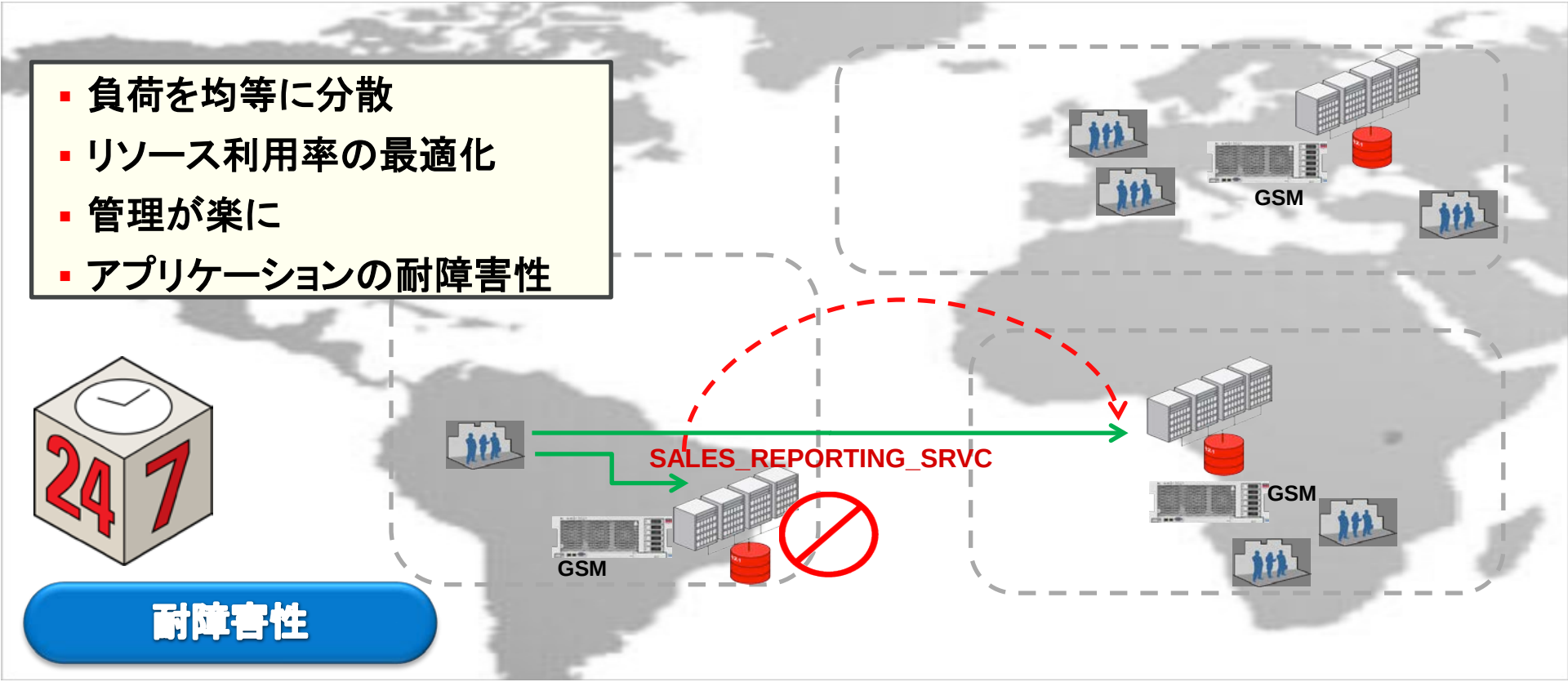
1. 複製データを活用している環境での現状
2. Global Data Services と RAC
3. 想定されるユース・ケース
4. グローバル・サービス
5. ロード・バランシング

複製データを活用している環境 – GDS 導入後



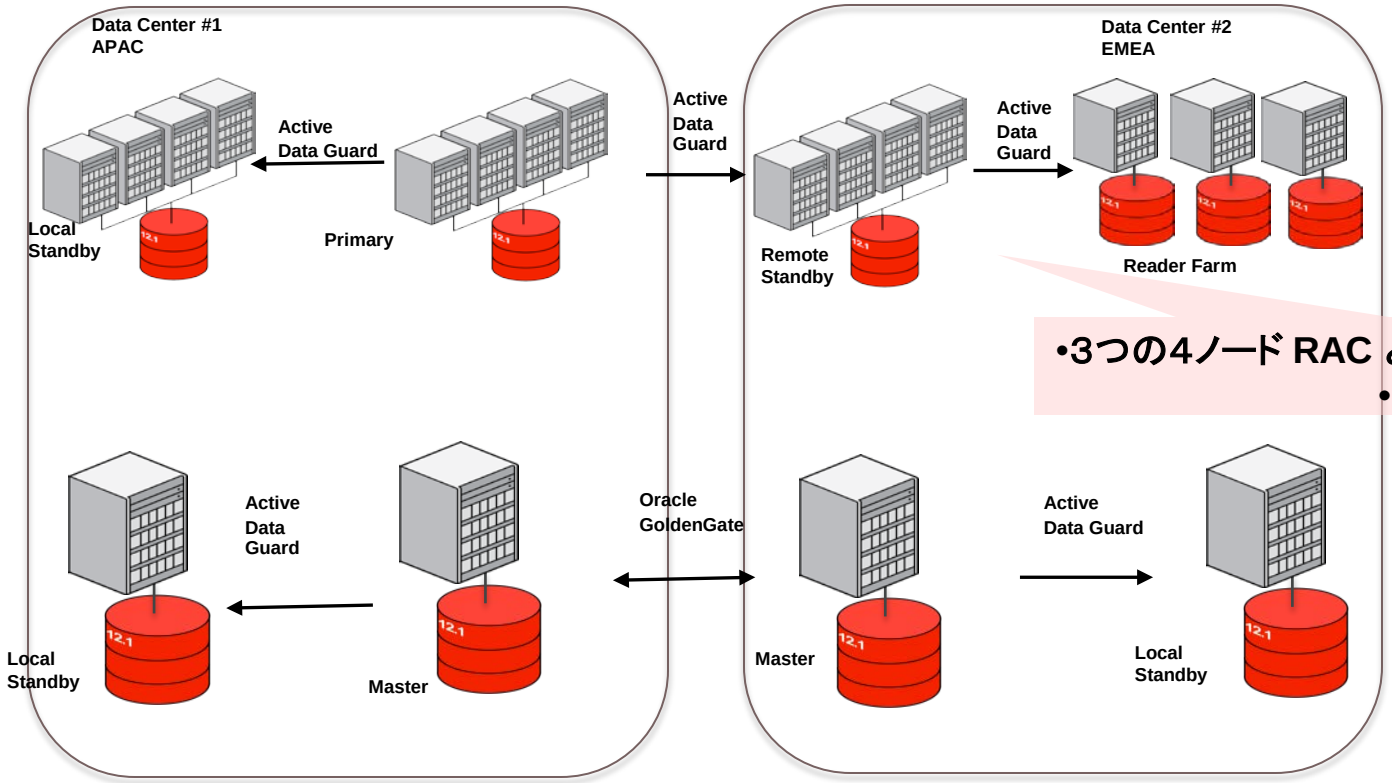
複製データを活用している環境 – GDS 導入後

- 負荷を均等に分散
- リソース利用率の最適化
- 管理が楽に
- アプリケーションの耐障害性



耐障害性

GDS 導入前後の物理的なイメージ – 導入前

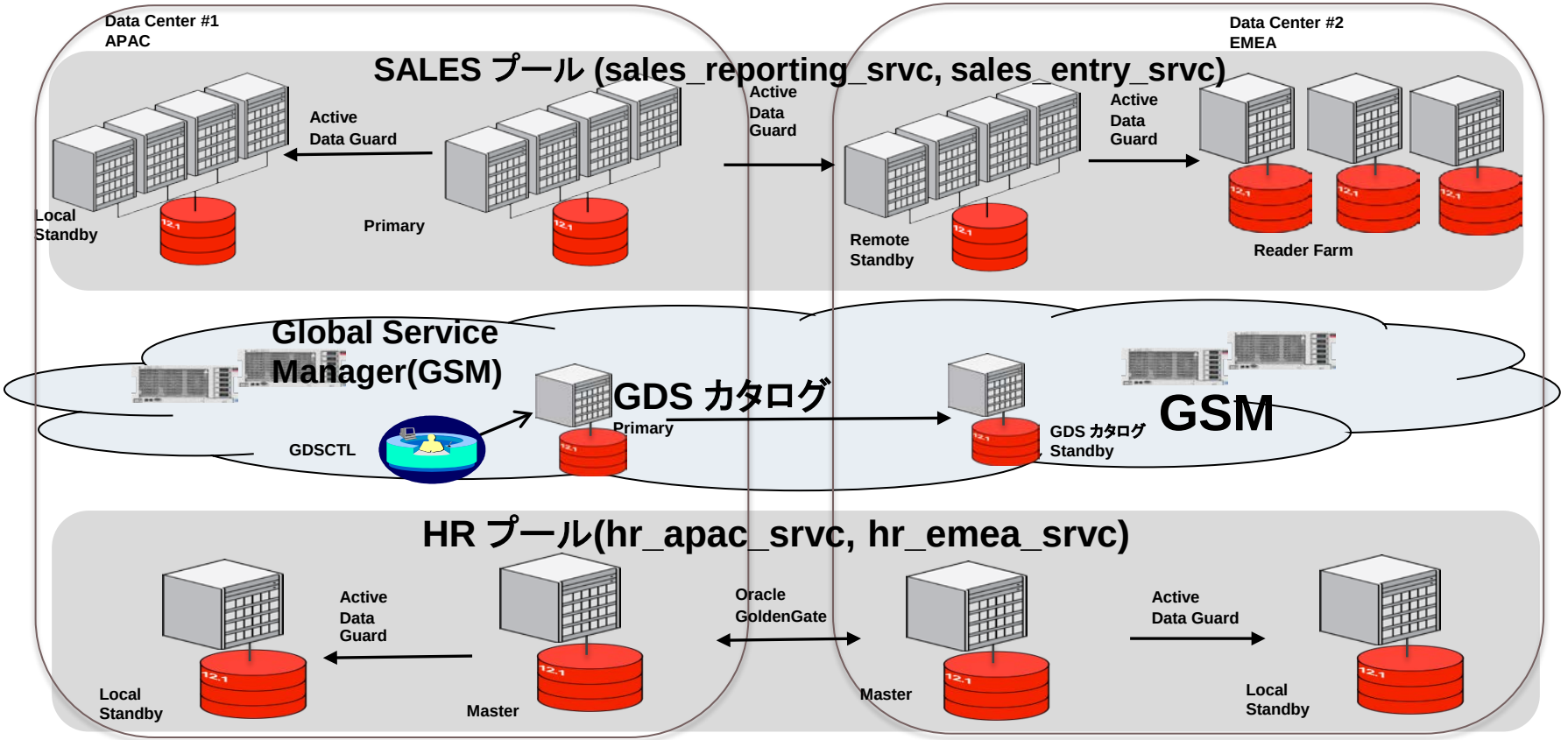


- データセンター2ヵ所
- 2組の独立した複製グループ

•3つの4ノード RAC と 3 シングル・インスタンスDB
•混在可能

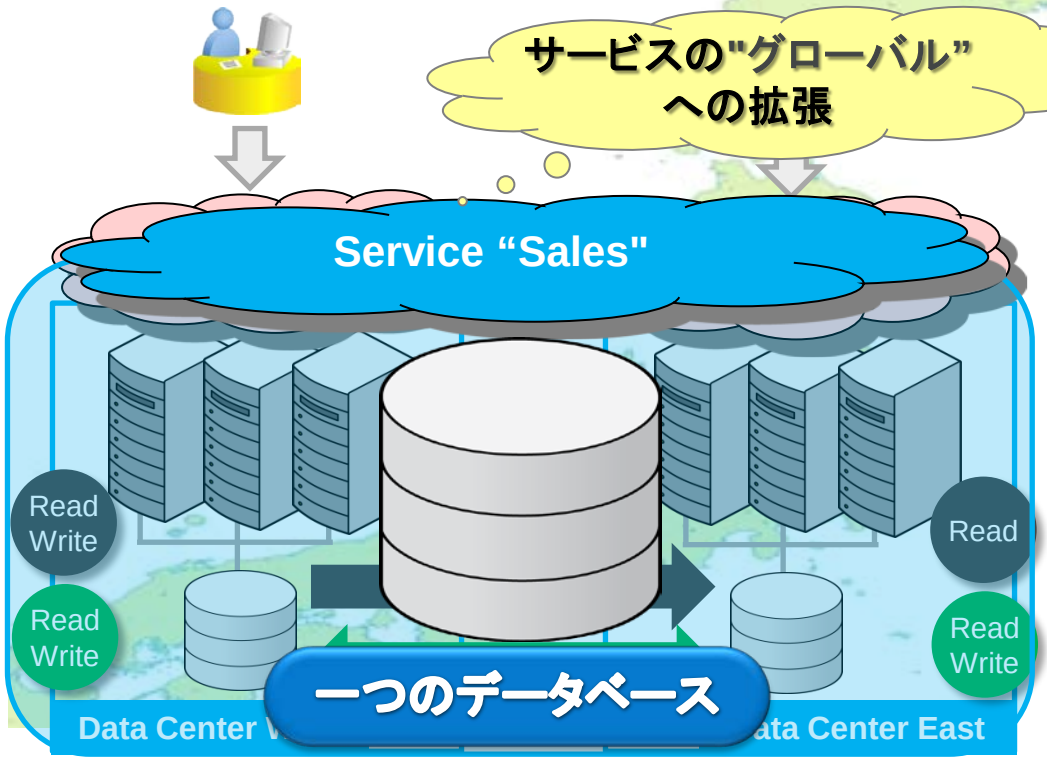
- レプリケーション手段は問わない
  例）Streams, Advanced Repl, スクリプト, ストレージ・コピー

GDS 導入前後の物理的なイメージ – 導入後



Global Data Services – RAC クラスター

独立したDB群を1つのRACクラスターのように使うことが可能に



GDSの特徴

- **負荷分散**

  DCを跨いだロードバランシングを実現

- **障害時フェイルオーバー**

  自動でのフェイルオーバーが可能

- **集中管理**

  単一(or少数)のサービスで一括管理

アジェンダ

1. 複製データを活用している環境での現状
2. Global Data Services と RAC
3. 想定されるユース・ケース
4. グローバル・サービス
5. ロード・バランシング

業務、アプリケーション　の適性

全てのサーバーで最新データにアクセスできるとは限らないので制約がある

読み取りのみの処理と書き込みありの処理の分離	データが最新でなくても良いか	コネクション・プール使用
例）レポーティング系	例） 30秒前 前日 先月末	ORACLE製プール

ユース・ケース ― データベース側

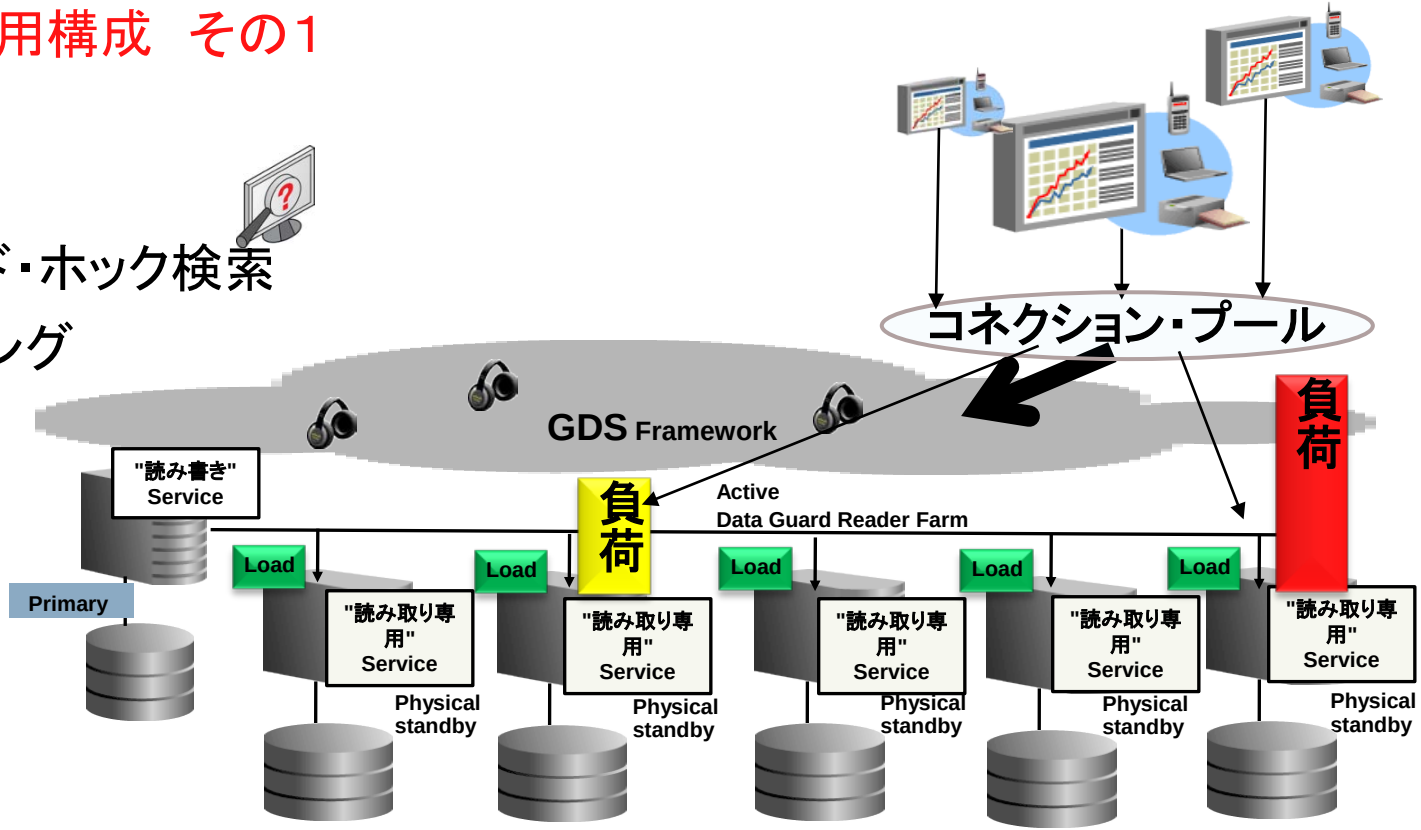
大分類

読み取り専用ファームでの負荷分散	フェイル・オーバー スイッチ・オーバー	マルチ・マスター環境での負荷分散
(リーダー・ファーム、Reader Farm) Active Data Guard GoldenGate	Data Guard Active Data Guard コールド・スタンバイ	GoldenGate Active-Active 構成

リーダー・ファームの使用用途と負荷の偏り

想定される使用構成　その1

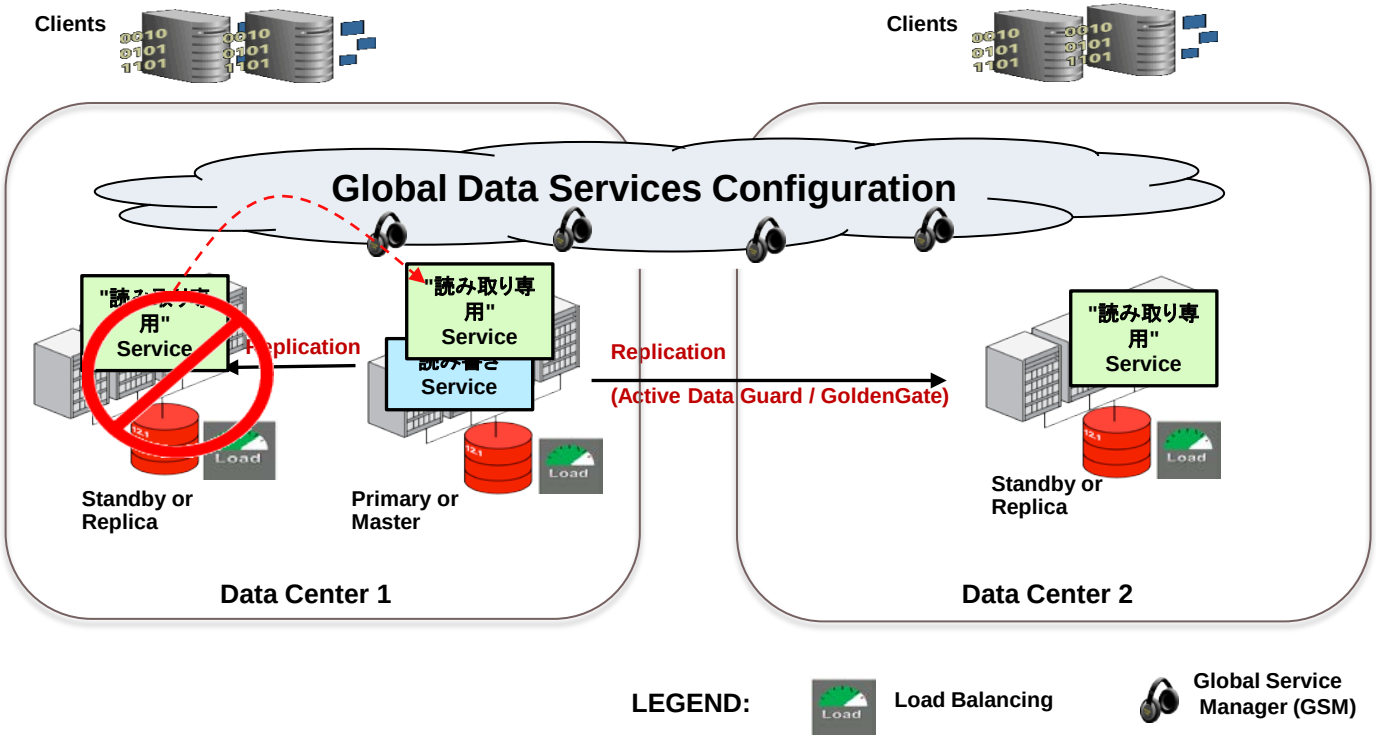
- バックアップ
- レポート作成
- 制限のないアド・ホック検索
- データ・マイニング



フェイル・オーバー　－ Data Guard, GoldenGate

想定される使用構成　その2

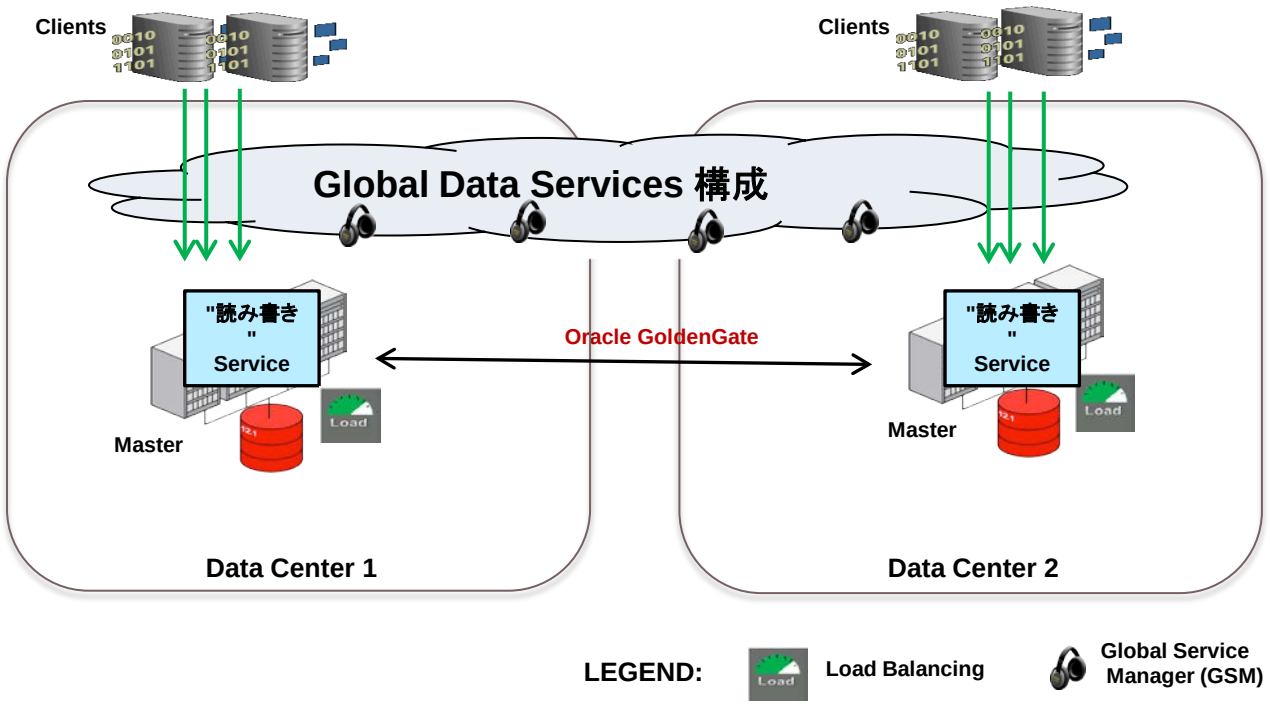
- “DBサービス”のフェイル・オーバー
- カーディナリティーを維持する
- DBが”OPEN”状態であることが前提です



GoldenGate Active-Active でのロード・バランシング

想定される使用構成　その3

- データ更新を含むアプリケーションのロードバランシング
- データ競合(コンフリクト)の解決はGoldenGate またはアプリケーションで対応されることが想定されています。

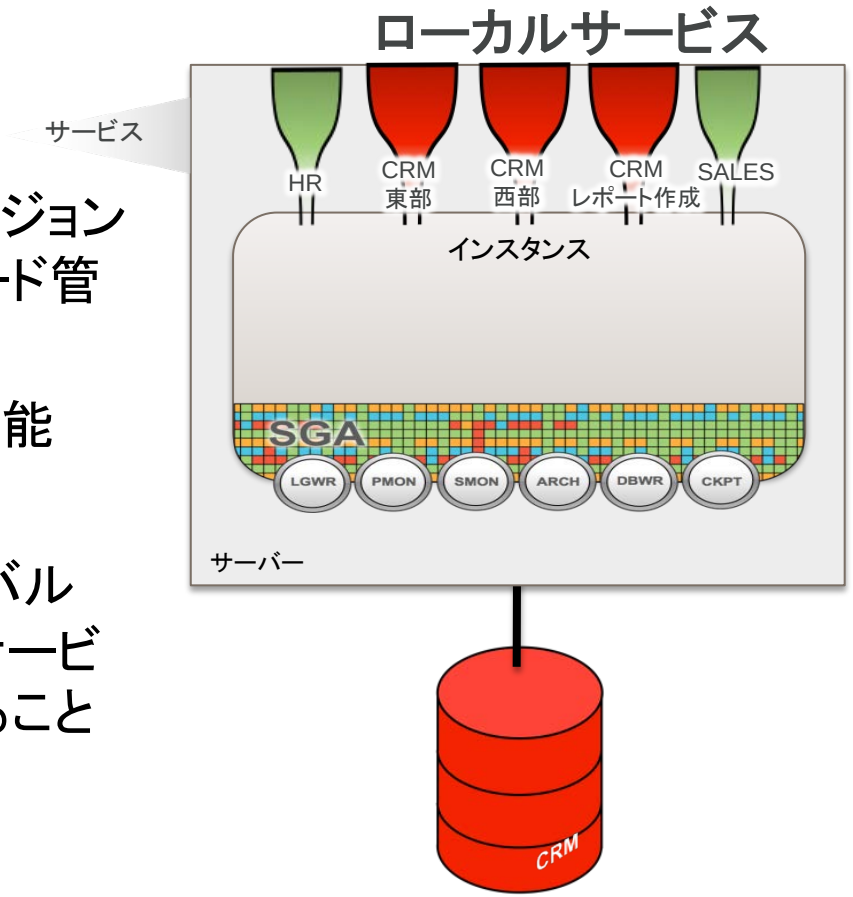


アジェンダ

1. 複製データを活用している環境での現状
2. Global Data Services と RAC
3. 想定されるユース・ケース
4. グローバル・サービス
5. ロード・バランシング

データベース “サービス”

- ローカルな”サービス”は以前のバージョンからあった仕組みで主に ワークロード管理 のために使われていました。
- シングルインスタンスDBでも作成可能
- クラスタリソースとして作成可能
- Global Data Services でのグローバルサービスと対比させるため旧来のサービスが “ローカル” サービスと呼ばれることがあります。



グローバル・サービス固有の 3 属性

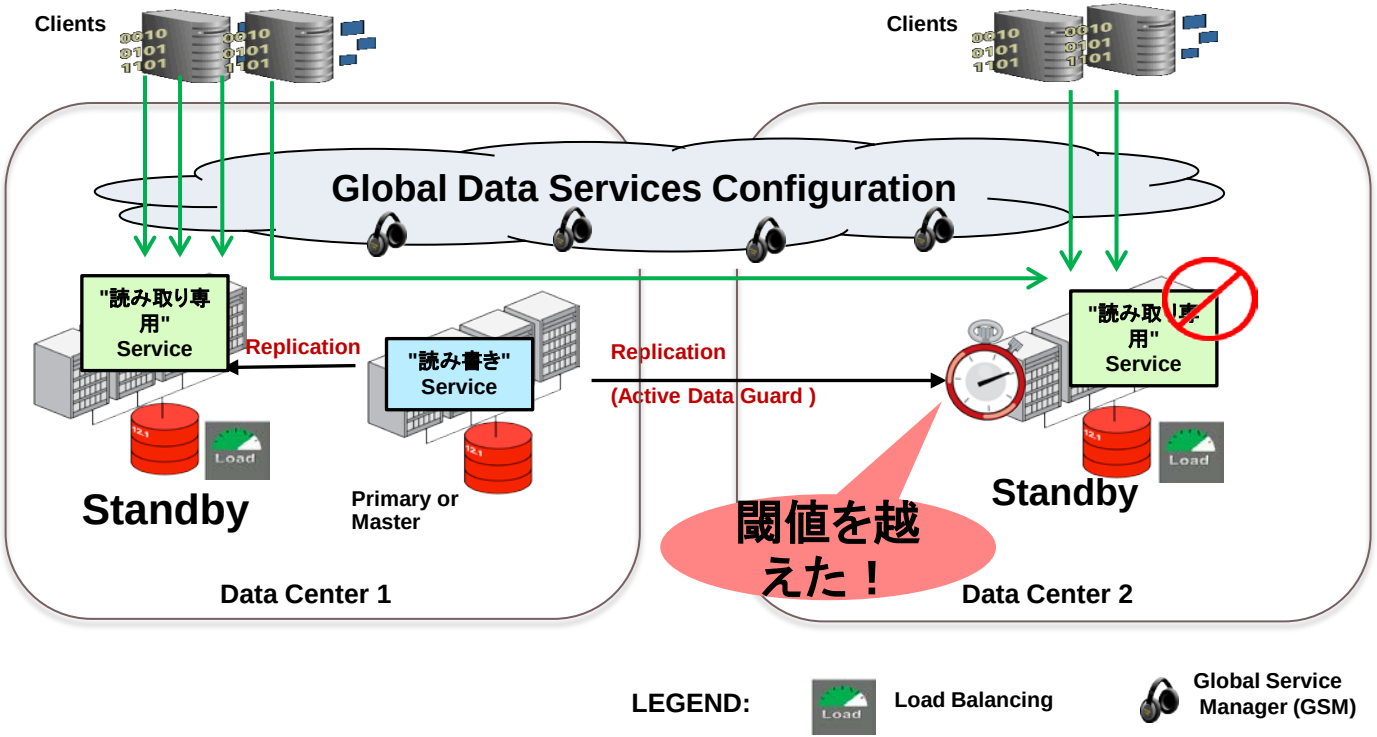
ローカル・サービスには存在しない新しい属性

カーディナリティー	遅延(ラグ、LAG)	リージョン・アフィニティ
•サービスを提供するDBの数 •リージョン毎に設定可能	•Active Data Guard 固有 •REDO適用遅延	遠隔データセンターのDBへの接続を許可するかどうか

ラグ属性を持った”グローバル”サービスの動作

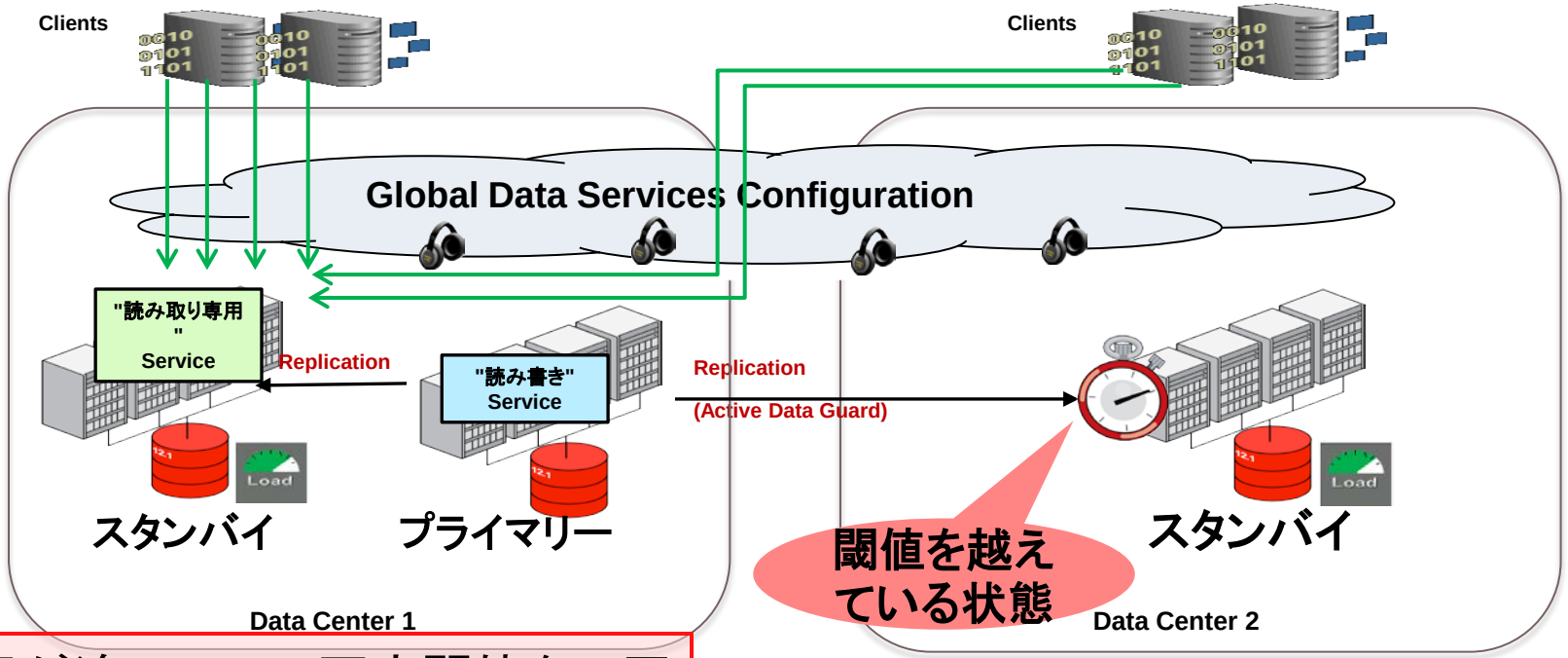
Active Data Guard スタンバイでのREDO適用遅延

- １種のSLA (Service Level Agreement)
- 右図はプライマリー1に対し、ローカルとリモートのDCにそれぞれスタンバイが１個



ラグ属性を持った”グローバル”サービスの動作

Active Data Guard スタンバイでのREDO適用遅延



適用が追いついて再度閾値を下回ればサービスは再起動されます

LEGEND: Load Balancing Global Service Manager (GSM)

“リージョン・アフィニティー” サービス属性

“グローバル”サービス<->クライアント、および、サービス<->DB　間

全リージョン	ローカル・リージョン	Local with Inter-Regional Fail-Over
•デフォルト •ロード・バランシングの際はネットワーク・レイテンシーが計算に入る	•クライアントは自分自身が属しているリージョンで稼働しているサービスにのみ接続可能 •サービス・カーディナリティーはリージョン毎に保たれる	ローカルに該当サービスが一個も起動してない場合のみ他リージョンを探す

アジェンダ

1. 複製データを活用している環境での現状
2. Global Data Services と RAC
3. 想定されるユース・ケース
4. グローバル・サービス
5. ロード・バランシング

サービスのロード・バランシング属性

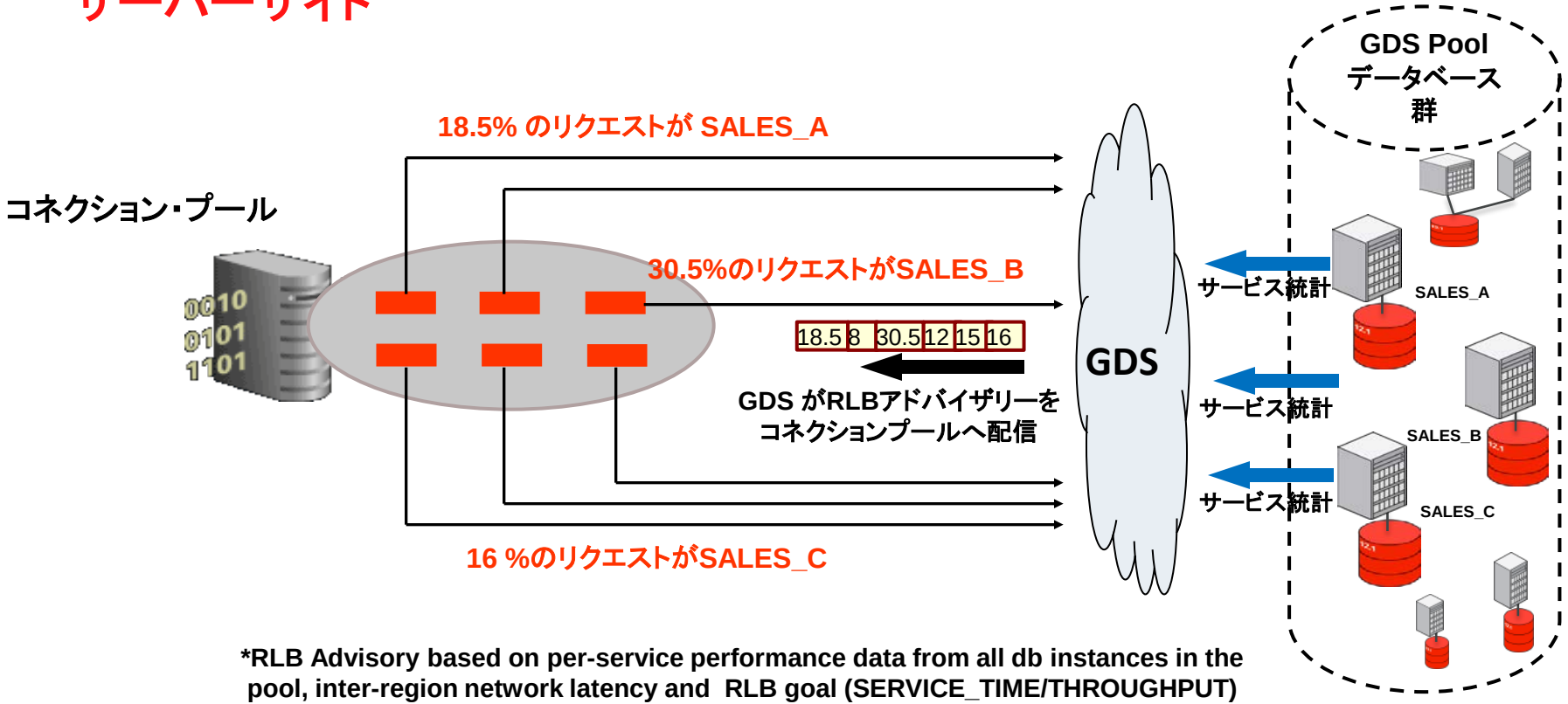
```
GDSCTL>config service -service sales_read_only
名前: sales_read_only
ネットワーク名: sales_read_only.hr.oradbcloud
プール: hr
起動済: はい
すべて優先: はい
ローカリティ: ANYWHERE
リージョン・フェイルオーバー: いいえ
ロール: PHYSICAL_STANDBY
プライマリ・フェイルオーバー: はい
ラグ: ANY
ランタイム・バランス SERVICE_TIME
接続バランス LONG
通知: はい
TAFポリシー: NONE
ポリシー: AUTOMATIC
DTP: いいえ
フェイルオーバー・メソッド: NONE
フェイルオーバー・タイプ: NONE
フェイルオーバー再試行:
フェイルオーバー遅延:
エディション:
PDB:
コミット結果:
保存タイムアウト:
再生開始タイムアウト:
セッション状態一貫性: DYNAMIC
SQL翻訳プロファイル:

データベース
------------------------
データベース                    優先       ステータス
------                    --       -----
dbvm3                     はい       有効
dbvm4                     はい       有効
```

SERVICE_TIME または THROUGHPUT または NONE

LONG または SHORT

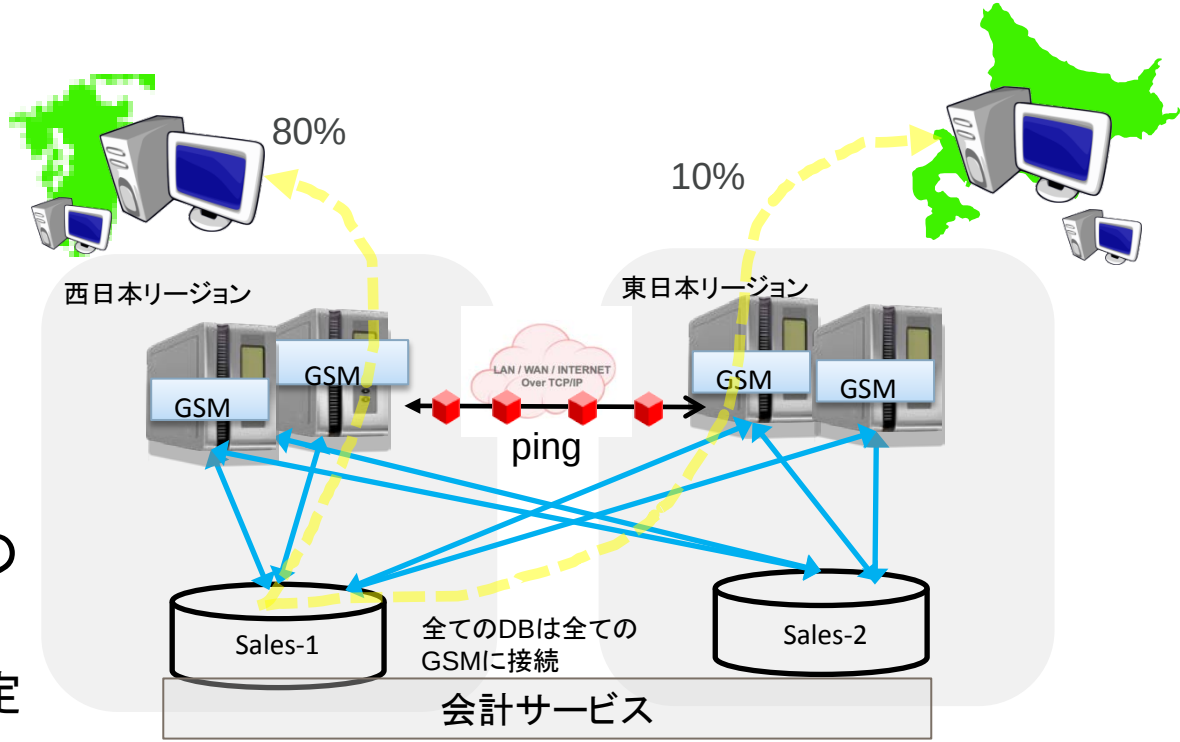
グローバル・ランタイム・ロード・バランシング (RLB)
サーバーサイド



*RLB Advisory based on per-service performance data from all db instances in the pool, inter-region network latency and RLB goal (SERVICE_TIME/THROUGHPUT)

DB12c RAC でのロードバランシングとの比較

- ロードバランシングの設定や計算方法は 11g,12c RAC と 12c の GDS は似ています。
- 異なる点: GSMは各DBから取ってくる負荷情報にREGION 間のネットワーク的な距離を加味した値をクライアントにガイドします
- RLB_GOAL が THROUGHPUT の時はDBの負荷情報のみでガイド
  - (DB時間) ≫ network と仮定



ORACLE

Oracle Database 12*c* Release 1

Application Continuity

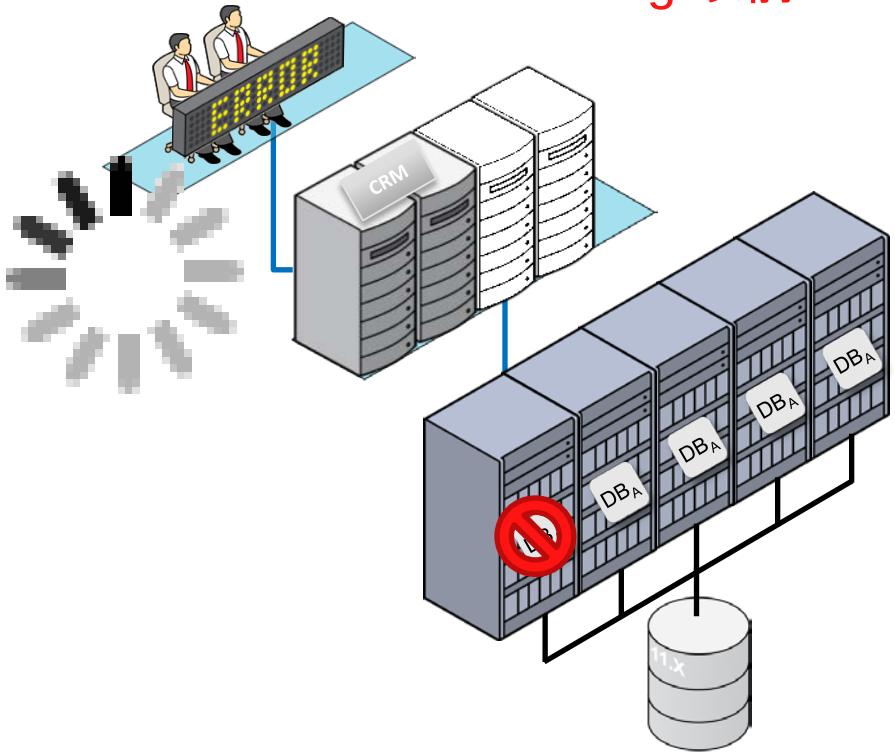
日本オラクル株式会社
井上 克己

アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

想定外（計画外）の停止、クラッシュ

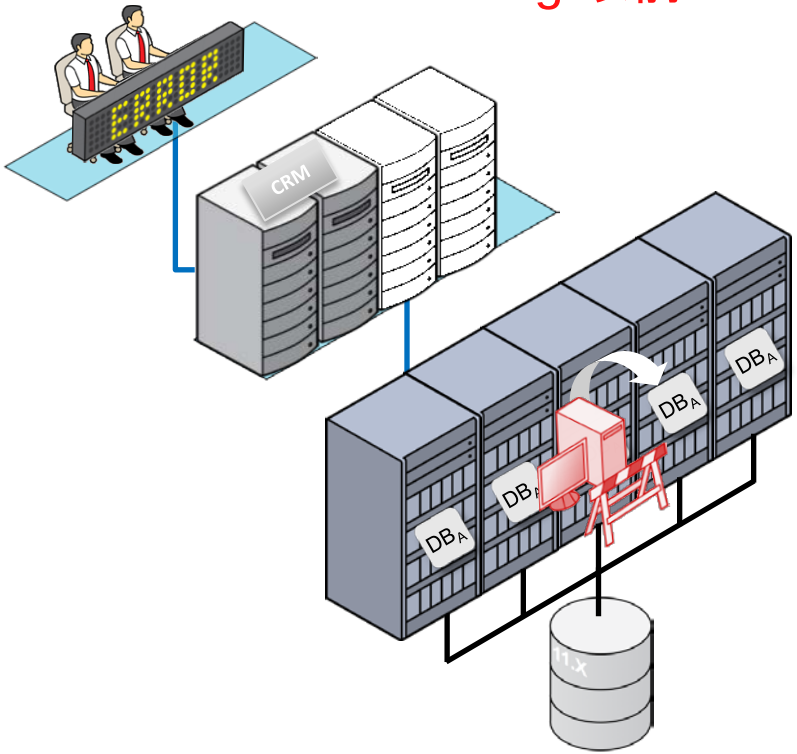
Oracle Database 11g 以前



- データベースの異常停止により実行中だったトランザクションのどこまでをコミットされたのか、どこから再入力する必要があるのか判断できないケースがありました。
- 2重コミット
- 不明トランザクションをクリーンアップするための中間層の初期化
- DB異常に対応するためのアプリケーションコードの複雑化

計画停止時にトランザクションが失われるケース

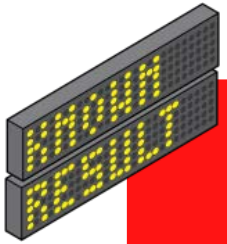
Oracle Database 11g 以前



- 計画停止の方が非計画停止より頻度が多い傾向があります
- FAN と Oracle Connection Pool(Weblogic,UCP,OCI,..) の組み合わせによりアイドル（使われていない）接続のクリーンアップは 10gR2 から可能
- FAN と Oracle Connection Pools でも長時間接続を離さないアプリには対応できていなかった

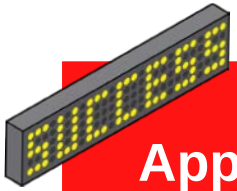
アプリケーション開発で実装が難しい点を解決

Oracle Database 12c 新機能



Transaction Guard

最後のトランザクション結果を確実に取り出すためのプロトコルとAPI



Application Continuity

想定外の停止、計画停止時に実行中だったセッションを安全に再実行する

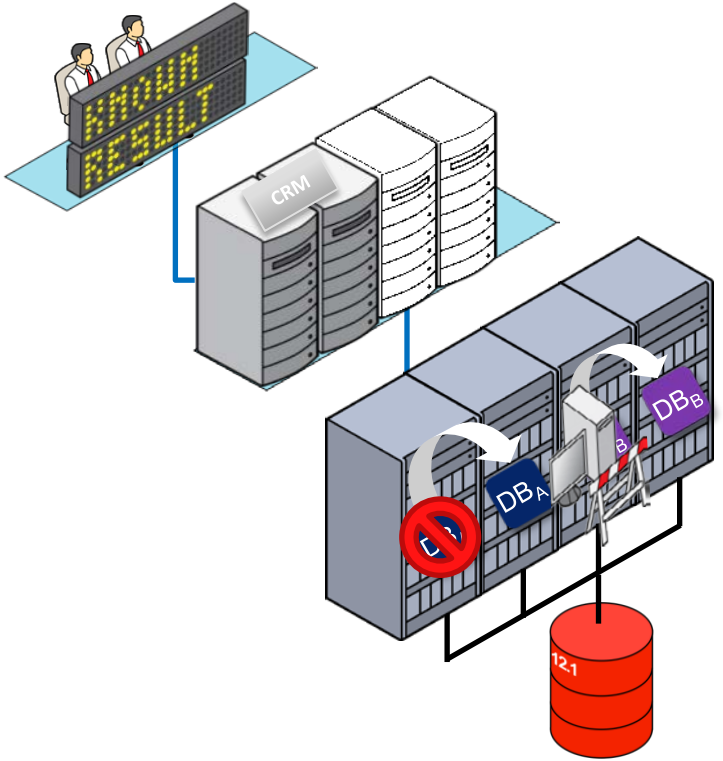
アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

Transaction Guard

Transaction Guard

コミット結果を保存。後から取り出し可能



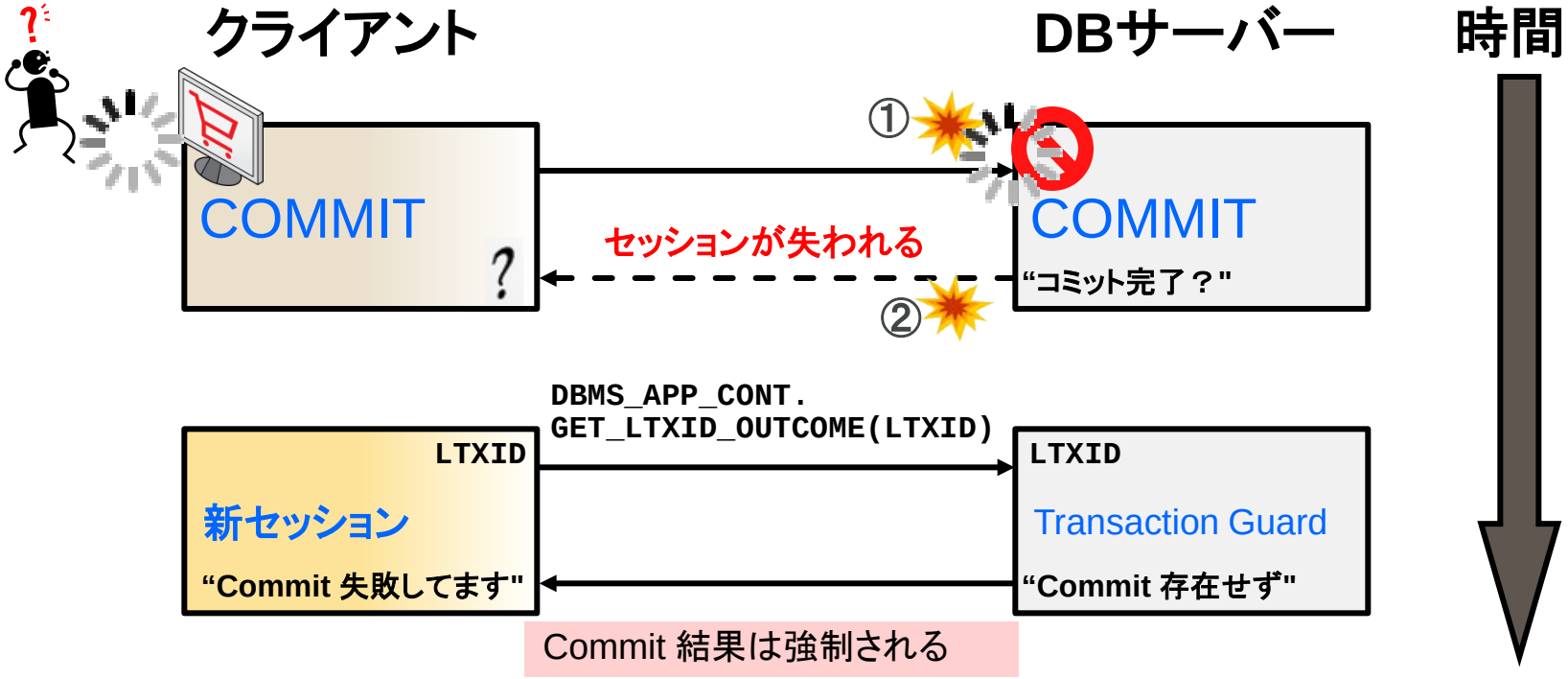
- トランザクションのコミット結果を取り出すAPI
- このAPIを使うことにより2重発注などのよくある問題を未然に防ぐことができる
- Transaction Guard APIによりアプリケーションは自分で再実行するのか、または、エンドユーザーに再入力を促すかなどの判断が可能に
- “Application Continuity for Java” 機能でも使われる
- JDBC-Thin, C/C++ (OCI/OCCI), ODP.Net クライアント用 API

アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

API使用 および 動作

コミット発行後に返事がなく実際には 失敗 していた場合



アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

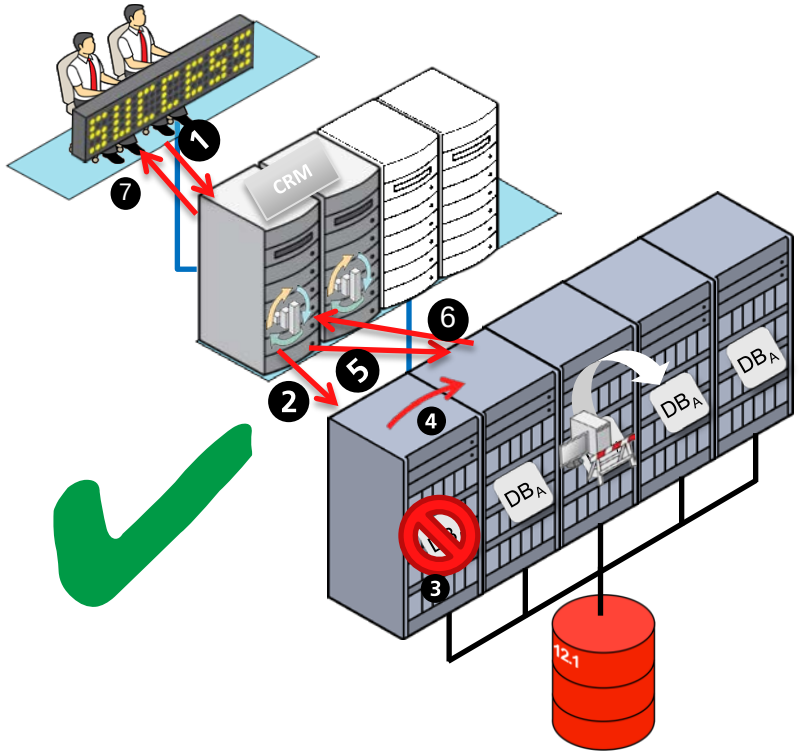
Application Continuity for Java

アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

Application Continuity for Java

計画外停止、および、計画停止をエンドユーザーに気づかせない



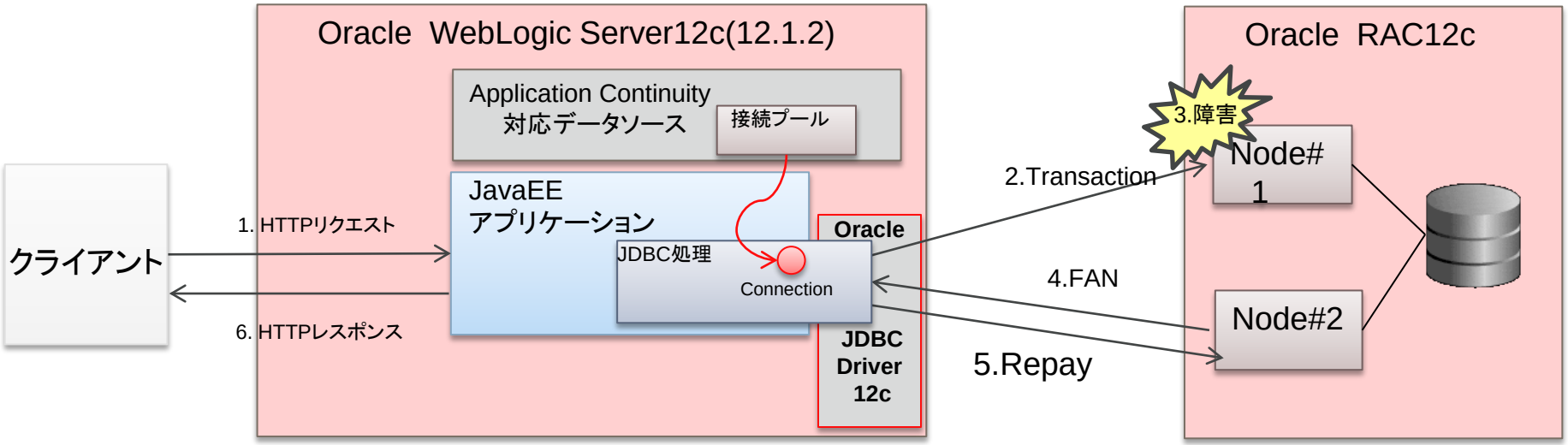
- Recoverable(回復可能)エラー発生時にDML(トランザクション)を再実行し再度データ投入
- 多くのハードウェア、ネットワーク、OS, ストレージのエラーや停止をエンドユーザーに気づかせずにトランザクション実行

アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

WebLogicデータソースのApplication Continuity対応

- WebLogic12c(12.1.2)では Application Continuityに対応したデータソースを構成できます。
- これにより、WebLogicで動作するアプリケーションがDB障害発生に透過的にJDBC処理をフェイルオーバー(Replay)させることが可能になります。



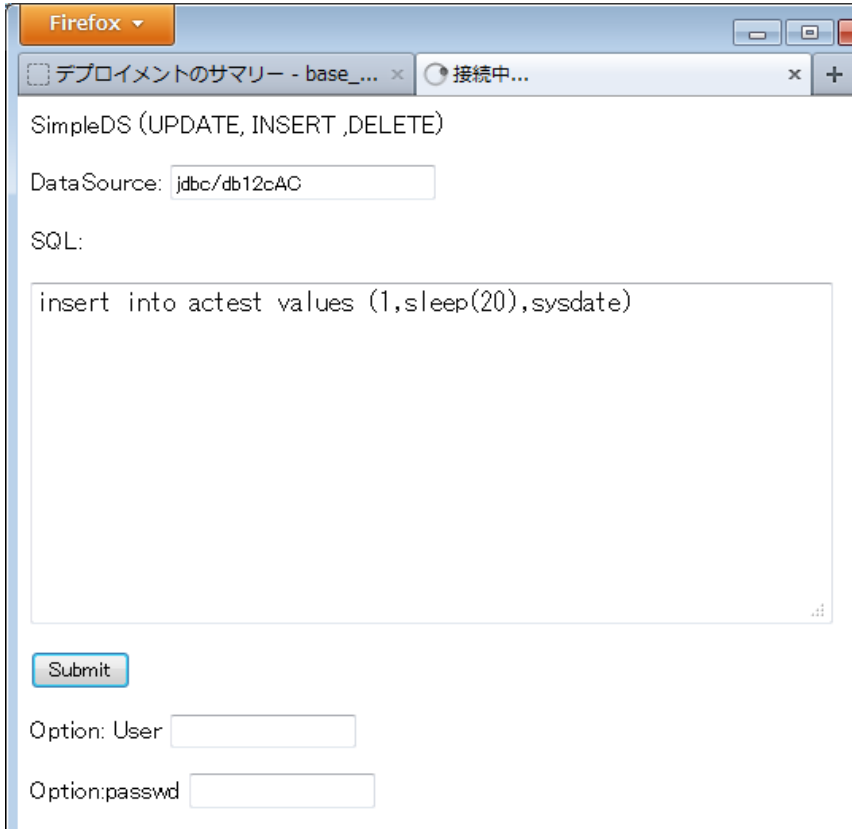
データソースの作成

新規接続ファクトリクラス

- 「ドライバ・クラス名」が「oracle.jdbc.replay.OracleDataSourceImpl 」になっていることを確認
- DB12c に含まれるJDBCドライバーに含まれる
  - ojdbc6.jar
  - ojdbc7.jar
- UCP(Universal Connection Pool) でもこのクラスを使い Application Continuity 機能を利用することができます

使用例①: アプリからのINSERT文発行

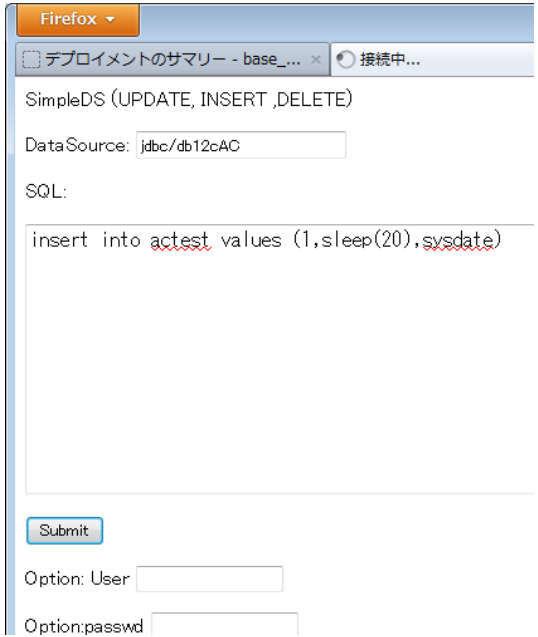
- Application Continuity データソースを利用し、INSERT文を発行するテスト用のJavaEEアプリケーションをWebLogicにデプロイします。
- INSERT文の対象となる表は、行が入っていない状態にしておきます。
- アプリケーションから（敢えて時間のかかる）INSERT文を発行します。



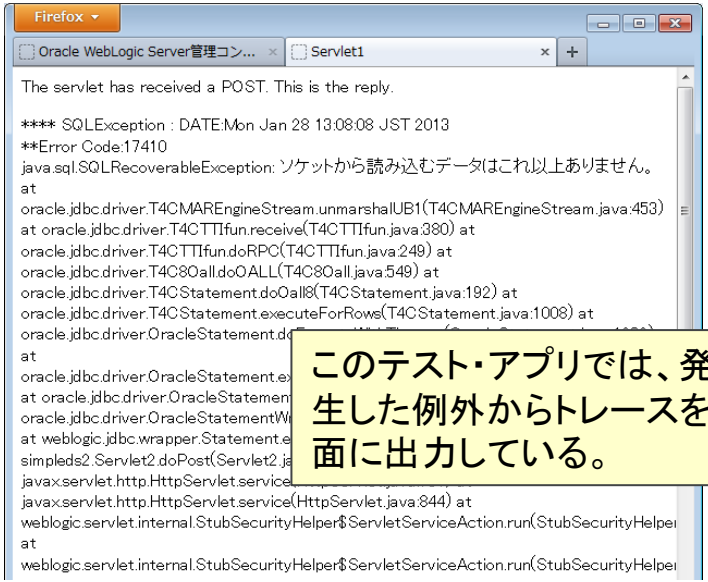
使用例②: アプリ側は実行中の状態であることを確認

- データベースを強制停止(shutdown abort) しても、アプリはまだ実行状態であることを確認します。

アプリはエラーにならず、実行中の状態



【参考】Application Continuity対応データソースを使用していない場合、同じ処理でも下図のようにエラーになる。

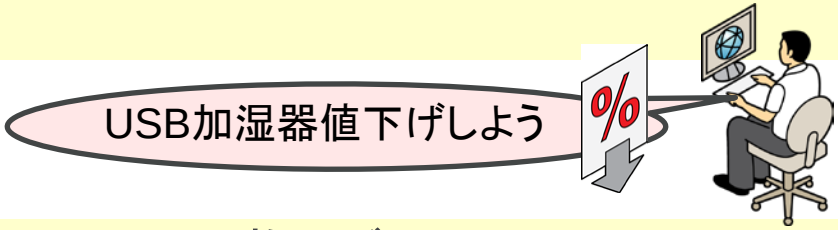


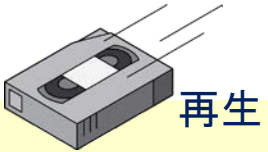
アジェンダ

1. **アプリケーションのフェイルオーバー: 現状**
2. Transaction Guard
   i. 動作
3. Application Continuity for Java
   i. 動作概要
   ii. Weblogic Server 12.1.2 での使用
   iii. 動作詳細

再生時に結果が変わる場合、再生は中断される

UPDATE 買い物カゴ SET OWNER='U12345' WHERE …
SELECT メーカー、値段 FROM 商品 WHERE .. 値段 < 3000
INSERT INTO 買い物候補 values(2480,'BrandCo','USB加湿器')





UPDATE 買い物カゴ SET OWNER='U12345' WHERE …
SELECT メーカー、値段 FROM 商品 WHERE .. 値段 < 3000

INSERT INTO 買い物候補 values(2480,'BrandCo','USB加湿器')

SQLException: ORA-41412: results changed during replay; failover cannot continue

初回実行時と同じ値を使うよう設定が可能なMutable

Mutable: 呼び出す毎に違う値を返す関数

シーケンス	タイムスタンプ	ユニークID生成
mySeq.NEXTVAL	SYSDATE SYSTIMESTAMP	SYS_GUID *再生時に Parallel Query で実行される場合同じ値は使われません

補足：Database Replay(RAT,Real Application Testing) にある機能と似た機能

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®