
ORACLE®

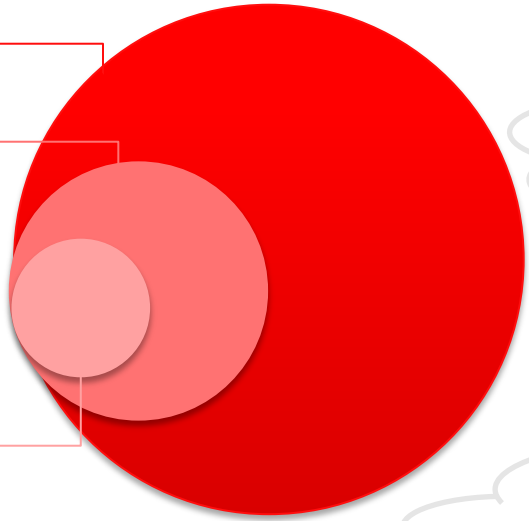
ORACLE

Oracle Real Application Testingによるアップグレード時のテストの効率化

日本オラクル株式会社
辻研一郎

はじめに

- アップグレードに伴うパフォーマンス劣化



チューニング対象

一部の SQL のために
全アプリケーションのテストを
するには工数がかかる

エラーが出る SQL や
パフォーマンス劣化が起こる SQL を
簡単に特定したい

Real Application Testing (RAT) とは？

データベースに特化したテストを効率化するための
Oracle Database Enterprise Edition の Option

- 本番環境でのワークロード(SQL) を Capture & Replay
  - 9i / 10g で取得した情報を 11g で再現
- Upgrade や Patch 適用などによるデータベースの変化に伴うパフォーマンス影響を判断可能
- アプリケーションに手を加えずに実業務のワークロードに即したテストが可能
- 疑似クライアントで高負荷状態も再現可能
  - 一度の Capture で Transaction のタイミング等を変えてテストの実行が可能

RAT とは？

- Real Application Testing
  - 下記の 2 つの機能を包含し、RAT と総称
    - SQL Performance Analyzer (SPA)
      - システム変更に伴うSQLのパフォーマンスの影響を調査
    - Database Replay (DB Replay)
      - システム変更に伴うアプリケーションのワークロードの影響を調査
        - » 実行時間、SQL実行結果行数、エラー内容

SPA と DB Replay の違い

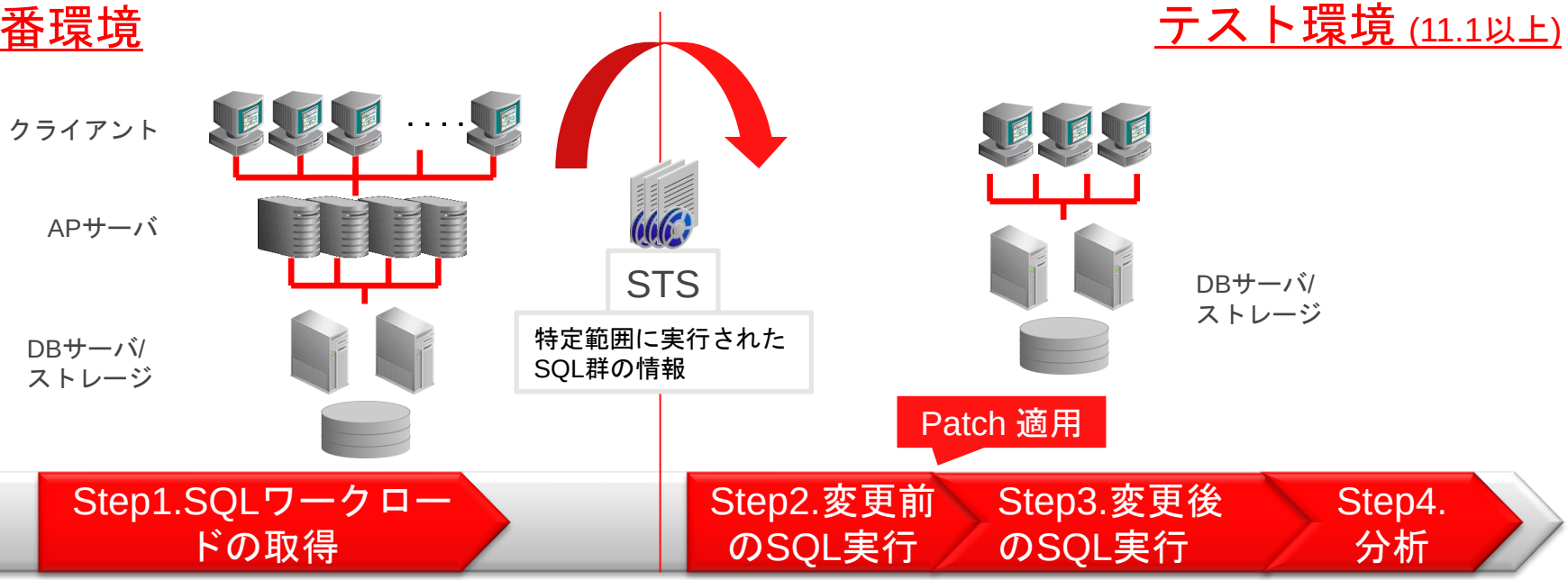
用途と機能

	SQL Performance Analyzer	Database Replay
どんな場面で使い分ける?	特定の重要な SQL に関してシステム変更によるパフォーマンス影響の有無を確認する SQL の単体テスト	データベースサーバー本番環境での負荷を用いサブシステムも含めた包括的なテストを実施する場合
何ができる機能?	システム変更に伴う SQL 応答時間の変化の影響を受ける前に確認する	テストシステム上において本番環境の負荷を再現
機能の目的は?	SQL の応答時間に対する影響度を評価	システムにおけるワークロードに対する影響度を評価
仕組みは?	SQL Tuning Set に格納された個別の SQL 文を実行しシステム変更前後における実行計画や実行時の統計値の比較を行う	本番環境にて収集された負荷を同時実行性やタイミングおよびトランザクション間の依存性も含め再現を行う

※ SQL Performance Analyzer とDatabase Replay はそれぞれ補完しあう機能

SQL Performance Analyzer (SPA)

基本概念

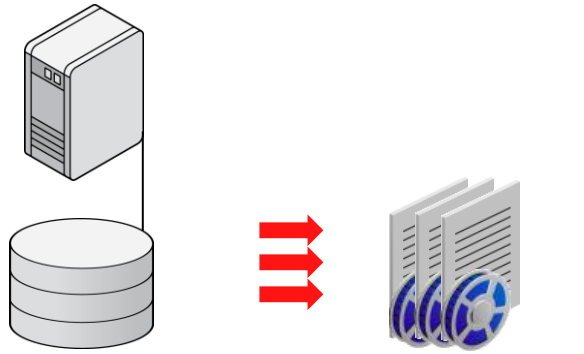


- 図は Patch 適用前後のSQLのパフォーマンス分析を想定
- アップグレード時は、本番環境の STS とテスト環境での SQL 実行結果を比較/分析

Step 1. SQL ワークロードの取得

SQL Performance Analyzer (SPA)

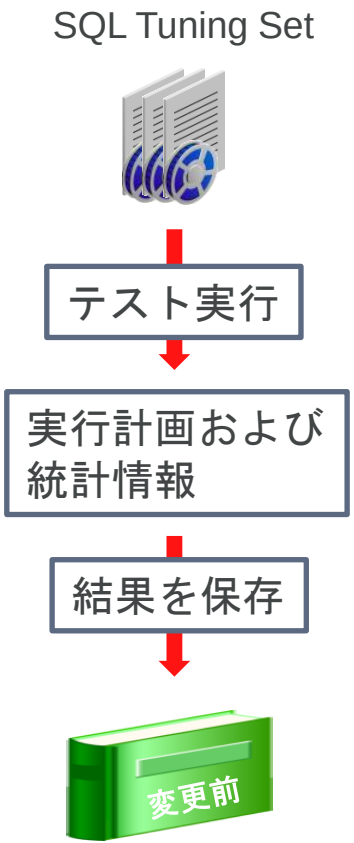
- SQL ワークロードを SQL Tuning Set (STS) に格納
- STS に含まれるデータ
  - SQL テキスト
  - バインド変数
  - 実行計画
  - 統計情報
- 既存の STS に対して SQL を追加可能
- 一定時間ごとにカーソル・キャッシュから情報収集し、STS に格納
- STS に格納する SQL はフィルタリング可能
- ただし、9i と 10gR1 は SQL トレースを取得し、STS への変換が必要



Step 2. 変更前のSQL実行

SQL Performance Analyzer (SPA)

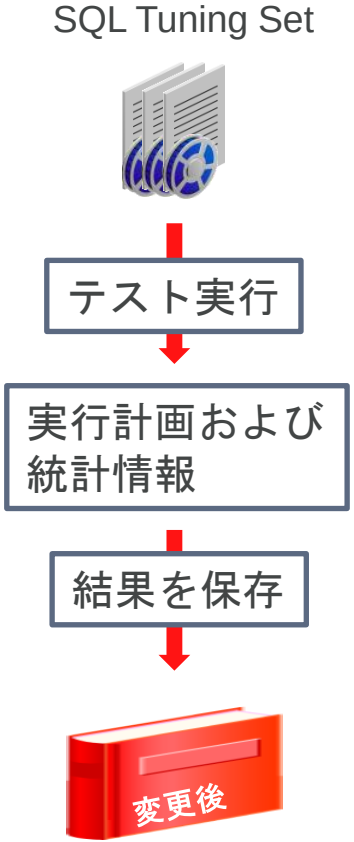
- 環境変更後のテストと比較するための
  ベースラインを作成
- SQL の実行計画と統計情報を取得
- SQL をシリアルに実行
- 各 SQL を1回ずつ実行
- DML / DDL は実行されない(default)
  - DBMS_SQLPA.EXECUTE_ANALYSIS_TASKプロシージャを使用して EXECUTION_PARAMS の EXECUTE_FULLDML パラメータを TRUE にすることにより、DML も実行可能 (SPA 実行後、ロールバックされる)
- SQL を実行せず、
  実行計画だけを取得することも可能



Step 3. 変更後のSQL実行

SQL Performance Analyzer (SPA)

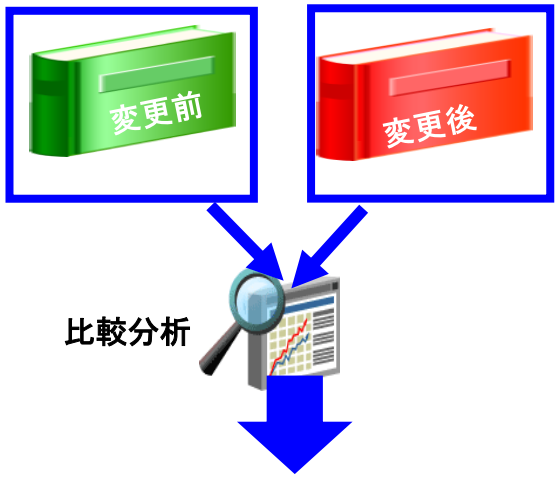
- テスト環境に変更を適用
  - DB アップグレード、パッチ適用
  - オプティマイザ統計情報の更新
  - スキーマ変更
  - チューニング
- 変更を適用後、同じ SQL を再実行
  - 実行計画と統計情報を収集



Step 4. 分析

SQL Performance Analyzer (SPA)

- 比較して表示される項目
  - 実行時間
  - CPU 時間
  - バッファ読み取りブロック数
  - ディスク読み取りブロック数
  - ダイレクト・パス書き込み
  - パース時間
  - オプティマイザ・コスト
- 変更適用により、影響のあった SQL を表示
  - 性能が改善した SQL
  - 性能が劣化した SQL
  - 性能が変わらなかった SQL
- SQL Tuning Advisor を利用して性能劣化した SQL をチューニング



分析レポートの活用

SQL Performance Analyzer (SPA)

- パフォーマンスが劣化した / 改善した SQL を特定できる
- SQL Tuning Advisor や SQL Plan Managemetを利用して、劣化してしまったSQL のチューニングを実施

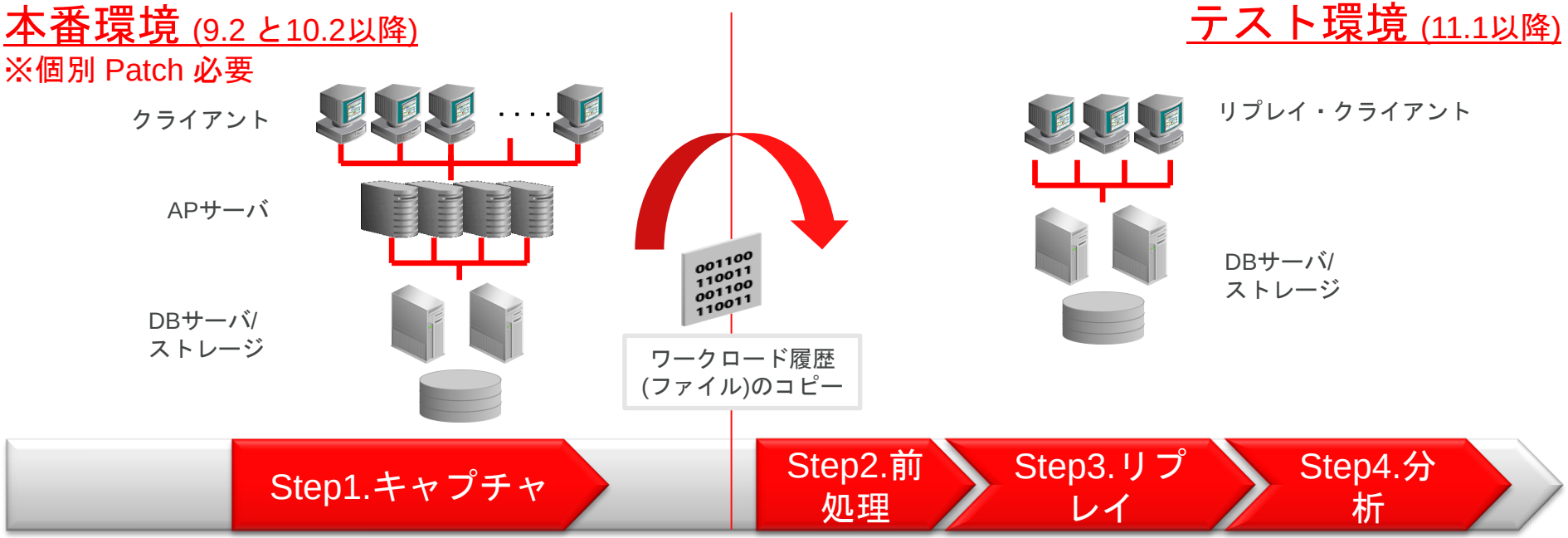
SPA 活用のポイント

- ワークロードの取得
  - 特に監視を必要とするワークロードやピーク時のワークロードの選択を推奨
- テスト環境
  - 本番環境とテスト環境が同程度のオプティマイザ統計情報を持つようにする
  - 変更適用前後の Buffer Cache のフラッシュを推奨
- パフォーマンス劣化の修正
  - SQL Tuning Advisor および SQL Plan Management を使用して、パフォーマンス劣化を修正

Database Replay (DB Replay)

基本概念

- 本番環境の負荷を取得し、そのままテスト環境で再現

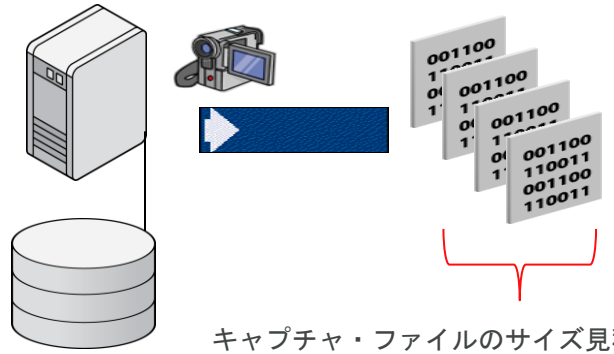


Step 1. キャプチャ

Database Replay (DB Replay)

- クライアントからの全リクエストを
  バイナリファイルにキャプチャ可能
  (キャプチャ・ファイル)
  - バックグラウンド処理は除外
- セッションを指定し、特定のワークロードの取得/除外が可能
  - セッションごとに、***.rec というファイルを作成
  - SYS、SYSTEM、DBSNMP、RMAN の除外を推奨
- キャプチャの実行時間を指定可能
- 9.2.0.8 or 10g R2 以降でキャプチャ可能
- RAC 環境ではキャプチャ・ファイルの保存先に共有ファイルとローカルファイルの両方をサポート

キャプチャ時のオーバーヘッド
CPU使用率:3～5%程度



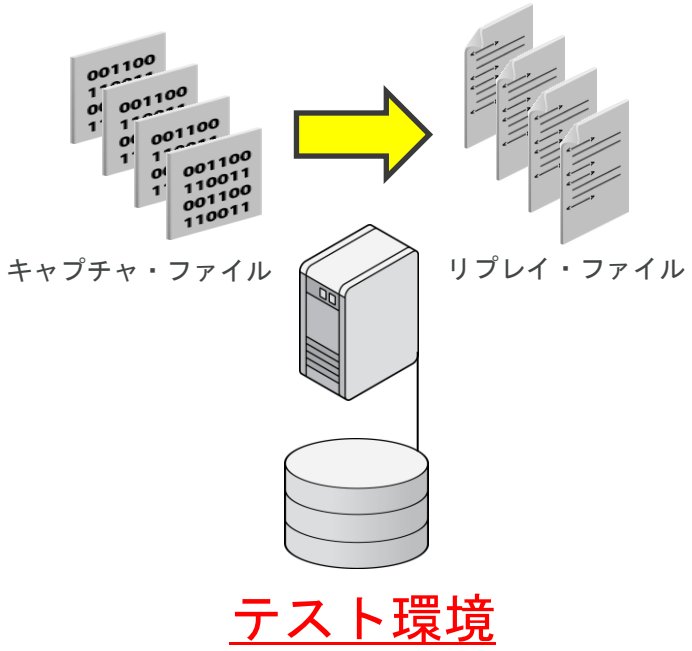
キャプチャ・ファイルのサイズ見積もり
(SQL*Net bytes from client) × 2

本番環境

Step 2. 前処理

Database Replay (DB Replay)

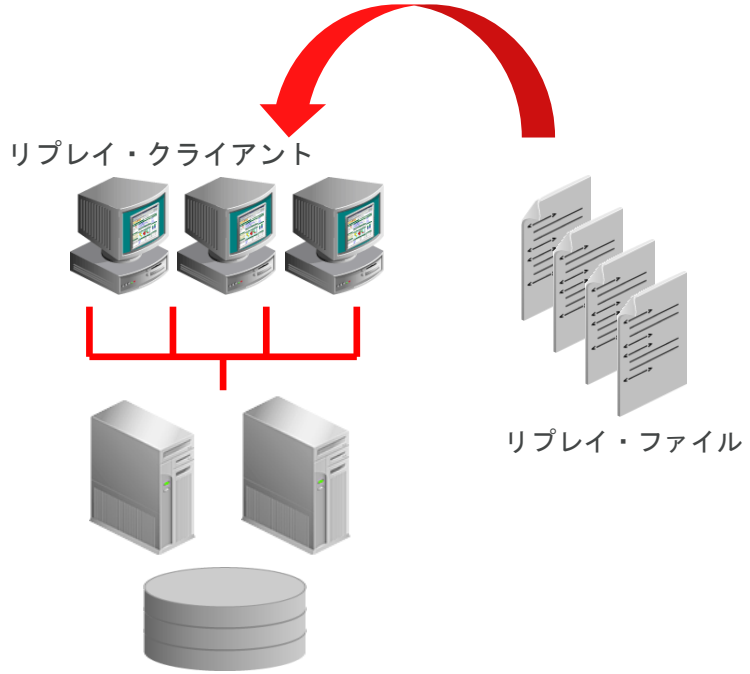
- 事前準備
  - 本番環境で取得したキャプチャ・ファイルをテスト環境へコピー
  - データを本番環境でのキャプチャ開始時点の内容に合わせておく
- キャプチャ・ファイルをリプレイ可能なフォーマットに変換(リプレイ・ファイル)
  ※リプレイする環境で実施することを推奨
- リプレイ・ファイルは何度でも実行可能
- RAC 環境でキャプチャした場合は、全インスタンスで生成されたキャプチャ・ファイルを一か所に集め、前処理を実施



Step 3. リプレイ

Database Replay (DB Replay)

- 本番環境で取得されたワークロードの実行時間、並列度、コミット順を再現してリプレイ
- リプレイ・クライアントがリクエストをDBサーバに送信
- クライアントはスレッドで実行されるため、1プロセスで複数並列度を再現可能。しかし負荷が大きい場合には必要に応じてクライアントを複数プロセス起動する（1プロセス最大50セッション）
- 11.2.0.3 (+Patch)以上で Consolidated Database Replay が可能 (後述)
  - スキーマレベルの Database 統合のテストを実施する際に利用する機能



(ご参考)リプレイ・オプション

Database Replay (DB Replay)

パラメータ	説明	備考
Synchronization	このパラメータはワークロード・リプレイ時に使用される同期のタイプを決定します。このパラメータがSCNに設定された場合、取得されたワークロードの COMMIT順序はリプレイ中もグローバルに保持され、取得時間SCNが小さいすべてのCOMMITアクションが完了した後でのみ、リプレイされたすべてのリクエストが実施されます。このパラメータがOBJECT_IDに設定されている場合、取得時間SCN値とデータベース・オブジェクトの両方に基づくより精度の高い同期方法を使用して、リプレイされたコール間の依存性が計算されます。デフォルト値はSCNです。	OFFに設定した場合、COMMIT発行順序性が担保されない事から、通常においては必ずSCNを設定します。
connect_time_scale	ワークロード取得が開始されてから、指定した値でセッションが接続されるまでの経過時間を変更します。 入力は、%値として解釈されます。 ワークロードのリプレイ中に同時ユーザー数を増加または削減する場合に使用できます。DEFAULT VALUEは100です。	何を評価するのかに依存しますが、DBのみのActivityを評価する場合は0に設定することも検討します。
think_time_scale	同じセッションからの2つの連続したユーザー・コール間の経過時間を変更します。 入力は、%値として解釈されます。 ワークロードのリプレイ中に同時ユーザー数を増加または削減する場合に使用できます。 DEFAULT VALUEは100です。	何を評価するのかに依存しますが、DBのみのActivityを評価する場合は0に設定することも検討します。
think_time_auto_correct	リプレイでのユーザー・コールの完了にかかる時間が、元の取得で同じユーザー・コールの完了にかかった時間よりも長くなる場合に、コール間の思考時間を適切に自動修正します。 DEFAULTのTRUEでは、リプレイが取得よりも遅くなった場合に思考時間が短縮されます。	通常TRUEに設定することで対応します。

Step 4. 分析

Database Replay (DB Replay)

- キャプチャ時とリプレイ時の違いをレポート
  - パフォーマンスの違い
  - エラーの違い
    - リプレイ時に発生した新規エラー
    - キャプチャ時に発生していたがリプレイ時に発生しなかったエラー
    - キャプチャ時に発生していたがリプレイ時に変異したエラー
  - データの違い
    - キャプチャ時と異なる行数が変更された DML
    - キャプチャ時と異なる件数が返された SELECT
- リプレイ実行後、比較レポートを生成
  - ワークロード・リプレイ レポート
  - AWR 比較レポート
  - ASH レポート

Replay Information

Information	Replay	Capture
Name	10GREPLAY	10g_capture
Status	COMPLETED	COMPLETED
Database Name	IZIZ1JIZ	P0PD1TOM
Database Version	11.2.0.3.0	10.2.0.3.0
Start Time	06-09-12 11:01:53	05-09-12 13:43:23
End Time	06-09-12 13:21:22	05-09-12 14:56:16
Duration	2 hours 19 minutes 29 seconds	1 hour 12 minutes 53 seconds
Directory Object	RECDIR	RECDIR
Directory Path	/home/oracle/ocs/RECDIR	/home/oracle/ocs/RECDIR
AWR DB Id	2083442946	
AWR Begin Snap Id	2863	
AWR End Snap Id	2878	
Replay Schedule Name		

Replay Options

Option Name	Value
Synchronization	SCN
Connect Time	0%
Think Time	0%
Think Time Auto Correct	TRUE
Number of WRC Clients	7 (7 Completed, 0 Running)

分析レポートの活用

Database Replay (DB Replay)

結果の概要

DB Replay Report for REPLAY-orcl-20121220194416

DB Name	DB Id	Release	RAC	Replay Name	Replay Status
ORCL	1318567314	11.2.0.3.0	NO	REPLAY-orcl-20121220194416	COMPLETED

Replay Information

Information	Replay	Capture
Name	REPLAY-orcl-20121220194416	RAT1217
Status	COMPLETED	COMPLETED
Database Name	ORCL	ORCL
Database Version	11.2.0.3.0	9.2.0.8.0
Start Time	20-12-12 19:44:43	21-12-12 19:32:55
End Time	20-12-12 19:45:27	21-12-12 19:33:49
Duration	44 seconds	54 seconds
Directory Object	RAT1217	RAT1217
Directory Path	/home/oracle/rat_poc/1217_rat	/home/oracle/rat_poc/1217_rat

詳細 AWR

- 開始時間と終了時間や、取得時と Replay 時の処理時間を比較し、全体性能の劣化がないかを確認
- エラーの有無も確認可能

違いのあるコールの確認

違いのあるコール

次に示すのは、違いの出たコールを、指定したフィルタ条件で絞り込んだ表です。

ヒント 各コールのSQL IDリンクからドリルダウンすると、そのコールのSQLテキストやバインドされた変数を表示できます。このリンクは、コール詳細がまだサーバーにロードされていない場合は無効になっています。ロード」ボタンをクリックすると、現在表示されている各コールのコール詳細がロードされます。

表示中のコールをロード

SQL ID	タイムスタンプ	相違のタイプ	相違の詳細	対象行	対象エラー	サービス	モジュール	アクション	セッション
f8y4ubckk4pfh	2013/02/21 17時51分03秒 JST	異なる行数がフェッチされたSELECT	SELECTによって返されると予測される行: 3	4	エラーなし	SYS$USERS	emagent_SQL_oracle_database	dbjob_status	24:6695

ドリルダウン

"REPLAY-orcl-20130221175000"に違いの出た文をリプレイ

閉じる

コール属性

対象エラー	エラーなし	サービス	SYS$USERS
SQL ID	f8y4ubckk4pfh	モジュール	emagent_SQL_oracle_database
セッション(ID:シリアル番号)	24:6695	アクション	dbjob_status

▶SQL文テキスト

```
/* OracleOEM */
select uname, mname, TO_CHAR(count(uname)) , concat(concat(uname,'_'), mname) username_machine
from
(select trim(chr(0)
from trim(username)) uname, trim(chr(0)
from trim(machine)) mname
from v$session
where type <> 'BACKGROUND' and username is not null ) group by uname, mname
```

閉じる

- 違いのあるコール部分の SQL ID 列をドリルダウンすると、詳細の SQL を確認できる
- 違いの内容まで確認することは不可

DB Replay の活用ポイント

- Capture 前の考慮点
  - 出力ファイルを保存するためのストレージ容量
  - CPU オーバーヘッド
- Capture 期間
  - 最初は極力短い時間で実施し、その後期間を延長していくことを推奨
  - 特に監視を必要とするワークロードやピーク時のワークロードの選択を推奨
- Capture 時のデータの整合性の担保
  - Replay 時の再現性を高めるため、Capture 前に Oracle Database を RESTRICT モードで起動することを推奨。(Capture 開始後は自動的に通常 OPEN 状態に戻る)

SPA と DB Replay の違い（再掲）

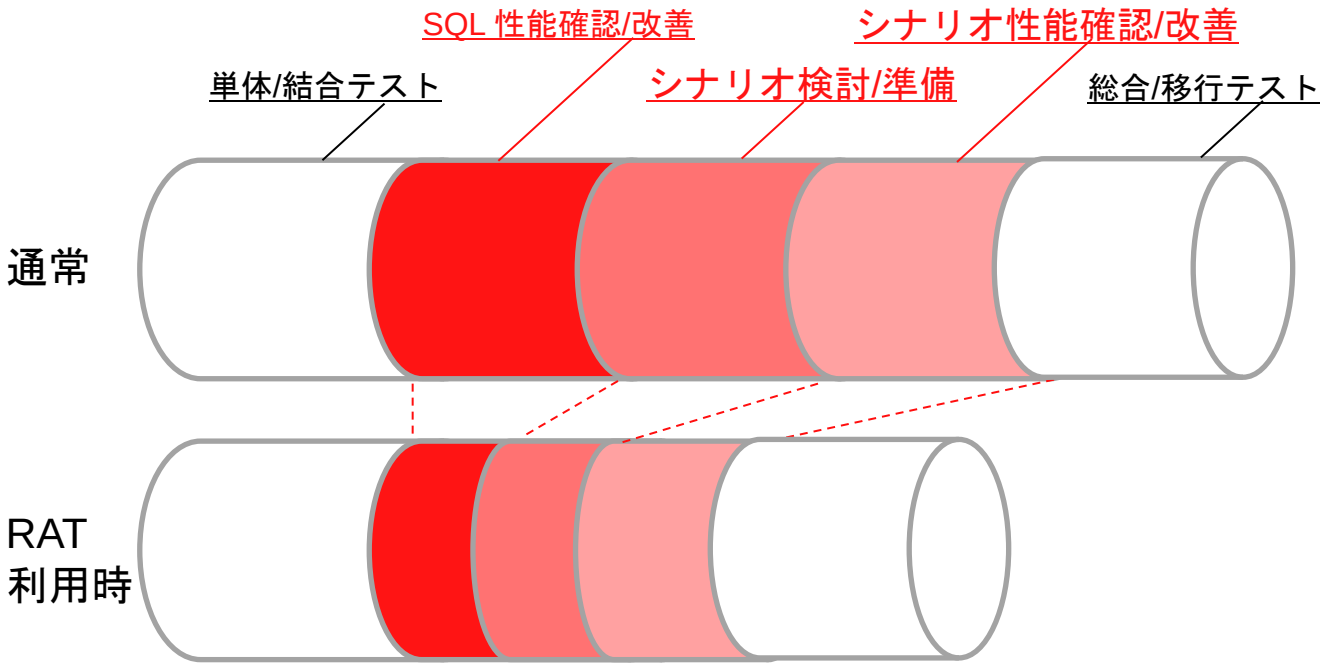
用途と機能

	SQL Performance Analyzer	Database Replay
どんな場面で使い分ける?	特定の重要な SQL に関してシステム変更によるパフォーマンス影響の有無を確認する SQL の単体テスト	データベースサーバー本番環境での負荷を用いサブシステムも含めた包括的なテストを実施する場合
何ができる機能?	システム変更に伴う SQL 応答時間の変化の影響を受ける前に確認する	テストシステム上において本番環境の負荷を再現
機能の目的は?	SQL の応答時間に対する影響度を評価	システムにおけるワークロードに対する影響度を評価
仕組みは?	SQL Tuning Set に格納された個別の SQL 文を実行しシステム変更前後における実行計画や実行時の統計値の比較を行う	本番環境にて収集された負荷を同時実行性やタイミングおよびトランザクション間の依存性も含め再現を行う

※ SQL Performance Analyzer とDatabase Replay はそれぞれ補完しあう機能

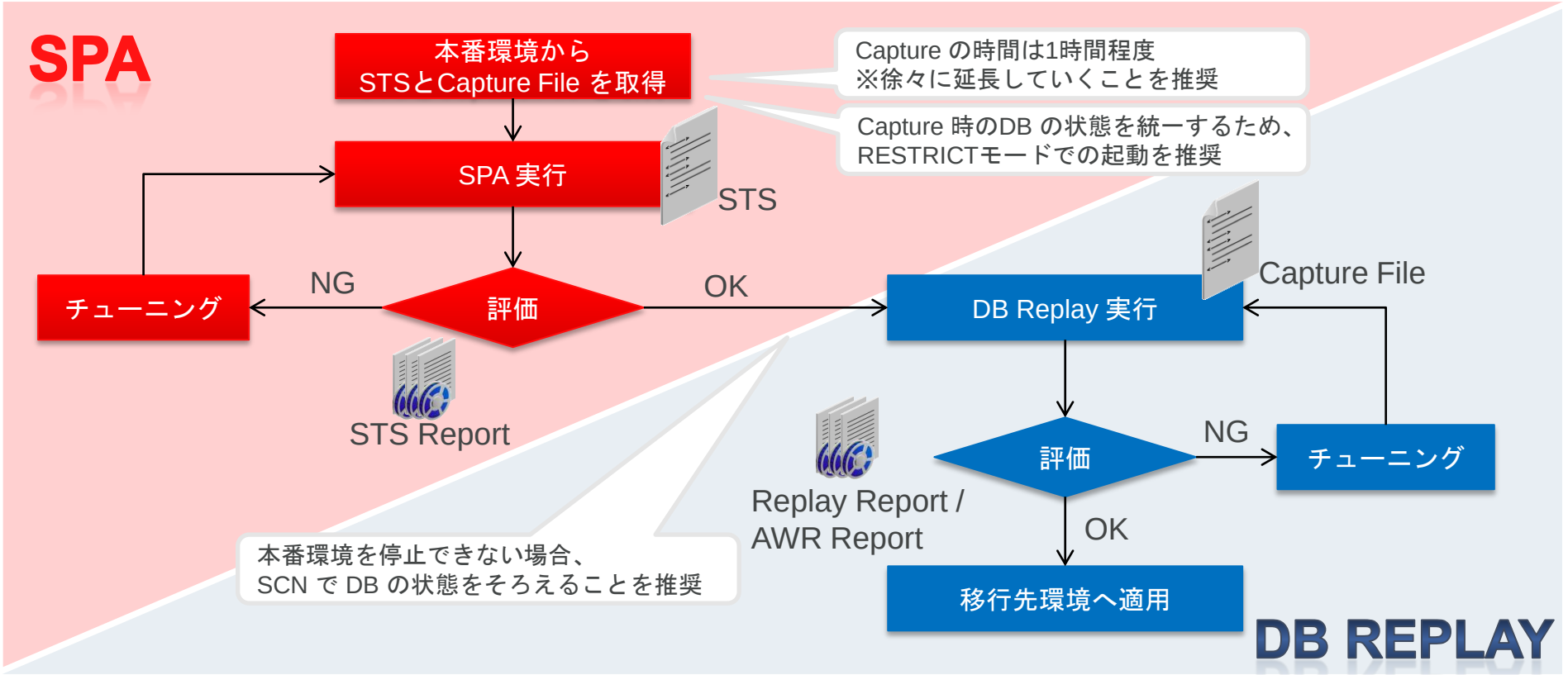
RAT のまとめ

工数削減のポイント



- SQL 性能確認/改善
  - 劣化する SQL を探す工数を削減
  - SQL Tuning Advisor によりチューニングを実施
- シナリオ検討/準備
  - DB Replay により、本番環境と同じシナリオを利用
- シナリオ性能確認/改善
  - 問題のあった SQL を瞬時に特定
  - 何度でも同じテストを実行

RAT の活用手順 (Best Practice)



Agenda

- RAT 機能概要
- RAT の Best Practice
- 活用事例 / Tips

活用事例

RAT / Veridata を活用した Upgrade

国内大手製造業 様

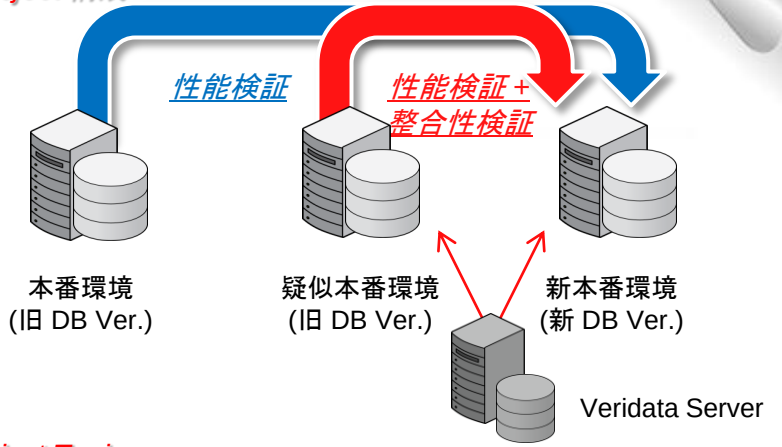
Upgrade 時の課題と Project Scope

- Upgrade 作業が属人化してしまい、平準化できない
- インスタンス数が多数あるため、平準化された技術を用いて横展開していきたい
- 改善すべき SQL の簡易的な特定を実施できず、全ての SQL チェックをしなければならない
- 最終的なデータの確認作業を自動化・省力化したい

RAT / Veridata 活用後の現状

- 本番環境をキャプチャし、テスト環境でリプレイするという単純作業で平準化し、全インスタンスへ適応可能
- RAT を用いることにより、システムの性能劣化を瞬時に判定可能
- 旧環境と新環境間でのデータは Veridata を利用し、自動的に内容を確認可能
- Upgrade 工数の 30% 削減を目標に実施

Project 構成

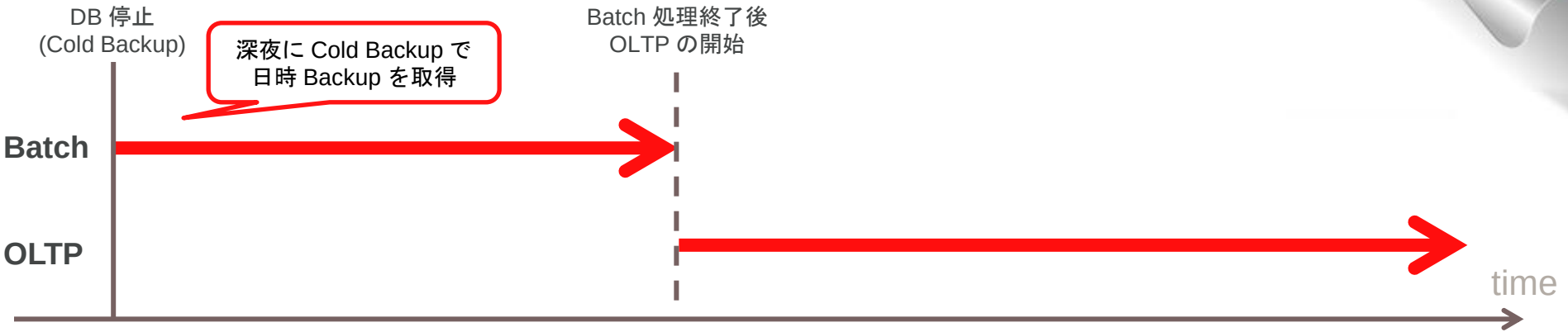


Project Topics

- データの完全一致を目指すため、本番環境ではなく疑似本番環境を構築し、整合性チェック
  (※本番環境は常に更新され続けるため)
- 本構成を用いることで、OLTP / Batch などあらゆるワークロードでの性能 / 整合性テストが可能
- インフラ担当部門のテスト範囲の拡大と品質向上

テスト手法の切り分け

お客様業務に基づくテスト手法



- 夜間 Batch 処理の前に一旦システムを停止しバックアップを取得するため、制止点を確保可能
- Batch と OLTP において別々のテストを実施
  - Batch 処理ではデータの整合性テストと性能検証を実施
  - OLTP では、別途テスト環境を構築し、性能検証のみを実施

テスト内容

ワークロードの検証	整合性テスト	単体 SQL の性能検証	個別チューニング / 全体チューニング
•Capture / Replay •SQL Trace	•Replay Report •GoldenGate Veridata	•STS Report •SPA •自作ツール	•SQL Tuning Advisor •AWR •再 Replay

- Best Practice では、最初に SPA を実施することを推奨しているが、ユーザーの要望により、最初に Capture / Replay を実施することが多い
- GoldenGate Veridata : 移行/レプリケーション時のデータの整合性チェックをする製品
- 自作ツールの利用
  - Oracle 9i Database R2 では DB time を取得できないため、各 SQL の性能比較が不可能。お客様自身で、SQL Trace の Elapsed Time を利用し、性能比較を実施

要件と機能の Fit & Gap

お客様の要望

1. とりあえず長時間 Capture し、全ワークロードを Replay したい
2. 本番 DB と開発 DB で全く同じ処理を流し、整合性テストを行いたい
3. DB Replay で各 SQL の DB Time を比較したい

製品仕様 / Best Practice

1. ファイルの容量や本番環境のパフォーマンス影響から、最初に行う Capture は 1,2時間を推奨
2. 整合性テストについては、restricted mode で実現可能だが、本番環境を止める運用を行っていない場合、データの完全一致は困難
3. 9i で DB Time は取得できないため、SQL トレースの Elapsed Time で代用

実行結果の整合性テスト

- 「違いのあるコール」として件数のみ表示
- 整合性テストのため、GoldenGate Veridata を利用し、各 DB でどのような違いが起こったのかの検証を実施

違いのあるコール

次に示すのは、違いの出たコールを、指定したフィルタ条件で絞り込んだ表です。

ヒント 各コールのSQL IDリンクからドリルダウンすると、そのコールのSQLテキストやバインドされた変数を表示できます。このリンクは、コール詳細がまだサーバーにロードされていない場合は無効になっています。「表示中のコールをロード」ボタンをクリックすると、現在表示されている各コールのコール詳細がロードされます。

表示中のコールをロード　前へ 1-15 / 20 次の行

SQL ID	タイムスタンプ	相違のタイプ	相違の詳細	対象行	対象エラー	サービス	モジュール	アクション	セッション
	2012/12/03 18時32分37秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		211:123
	2012/12/03 18時32分39秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		244:2613
	2012/12/03 18時33分03秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		211:123
	2012/12/03 18時33分06秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		211:123
	2012/12/03 18時33分13秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		211:123
	2012/12/03 18時33分49秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		35:339
	2012/12/03 18時33分49秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		211:123
	2012/12/03 18時33分56秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		35:339
	2012/12/03 18時34分00秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		35:339
	2012/12/03 18時34分03秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		35:339
	2012/12/03 18時34分08秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		35:339
	2012/12/03 18時34分29秒 JST	異なる行数が変更されたDML	DMLによって修正されると予測される行: 1	0	エラーなし	SYS$USERS	JDBC Thin Client		211:123

DB time による単体 SQL の性能比較

- 劣化する SQL と改善する SQL を比較するために、AWR Report の DB time よりその差を判断しようとしていた

例)

SQL A : 0.5 s × 1000本
SQL B : 20 s × 1本

- SQL A が5秒遅くなる場合、業務への影響は多大だが、SQL B が10秒遅くなっても全体への影響は少ない
- 各 SQL の特徴をつかみ、改善計画を立てることが重要

対処策

- SQL Trace の Elapsed Time で比較するためのツールを作成
  - 10g 以降であれば DB time での比較が可能なため、この手間を削減できる

その他のお客様でのユースケース

		課題	解決策
1	サーバー更改におけるプラットフォーム選定	● AP 層や周辺システムとの連携に多大な時間がかかる ● 各ベンダーへのテストデータ/アプリの提供や稼働確認への対応時間がかかる ● ストレステストを簡易的に実施したい	● RAT でキャプチャした SQL とキャプチャ直前のバックアップデータのみで各ベンダーに於いて短時間でテストが可能 ● RAT のパラメータを変更することで、高負荷状態を再現
2	新環境で旧環境を再現	● 本番環境でのCaptureが実施できない(Patch適用ができない)ため、RATを利用できない	● 新環境で optimizer_features_enable パラメータを 9i に設定してからキャプチャし 11g に変更してReplay

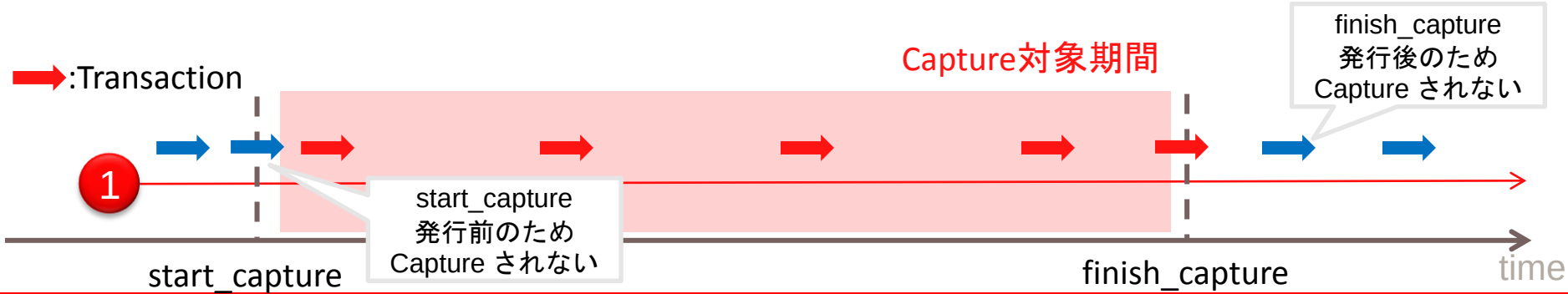
DB Replay Tips

Agenda

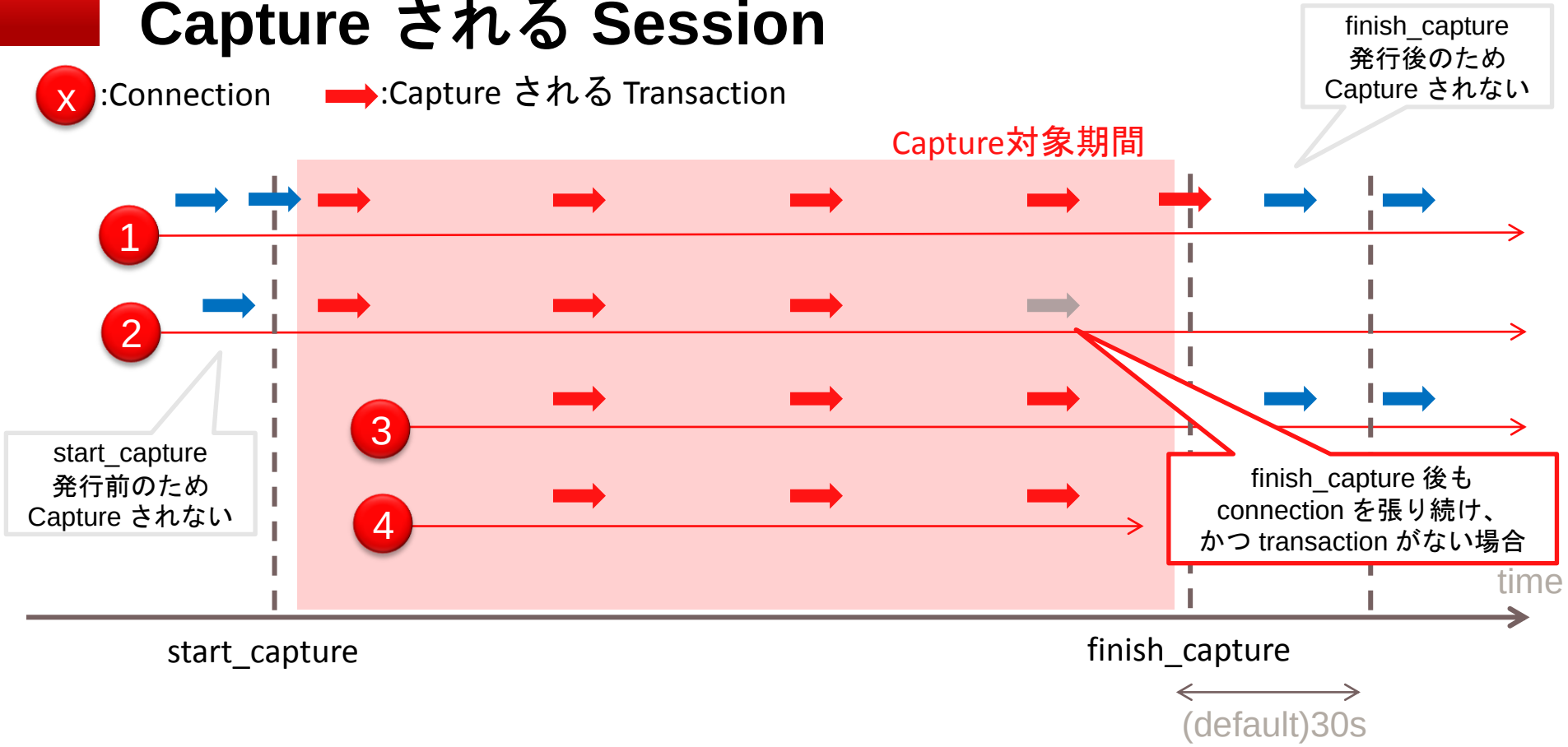
- In-Fright Transaction の取り扱い
- Capture されるセッションとされないセッションの比較
- PL/SQL の Capture
- Capture 時のオーバーヘッド
- 統合環境における RAT の活用

In-flight Transaction の取り扱い

- start_capture コマンドと finish_capture コマンドの間の期間は Capture されるが、start_capture 時に実行中のトランザクションについては、Capture することができない
  - SQL が発行される際の SQL 文のみを記録する仕様のため
- SCN をもとに、明確な分岐点を見極める必要がある

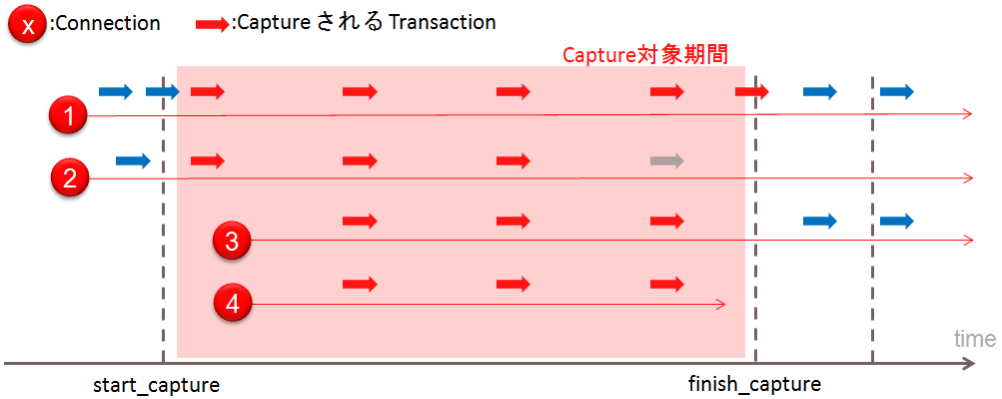


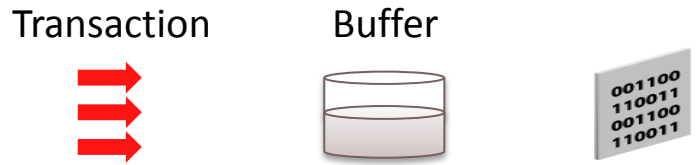
Capture される Session



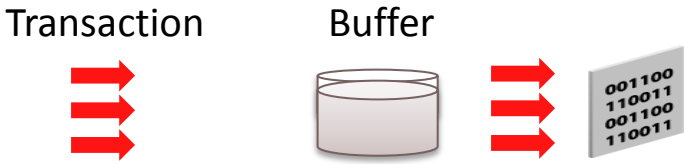
Capture される Session

- キャプチャ・ファイルに書き出される条件
  - セッションがログアウトした場合 4
  - finish_capture プロシージャが発行された場合 3
    - その後にトランザクションが必要
    - finish_capture の待ち時間は設定可能
  - Buffer が75% 以上になった場合(default 64kb) 2

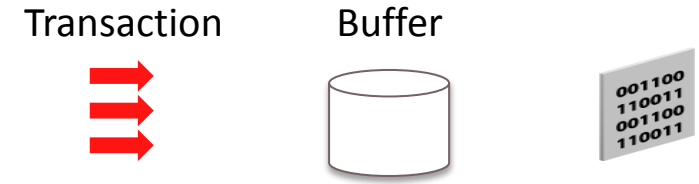




75%に達しない場合は書き出されない



75%を超えると書き出される

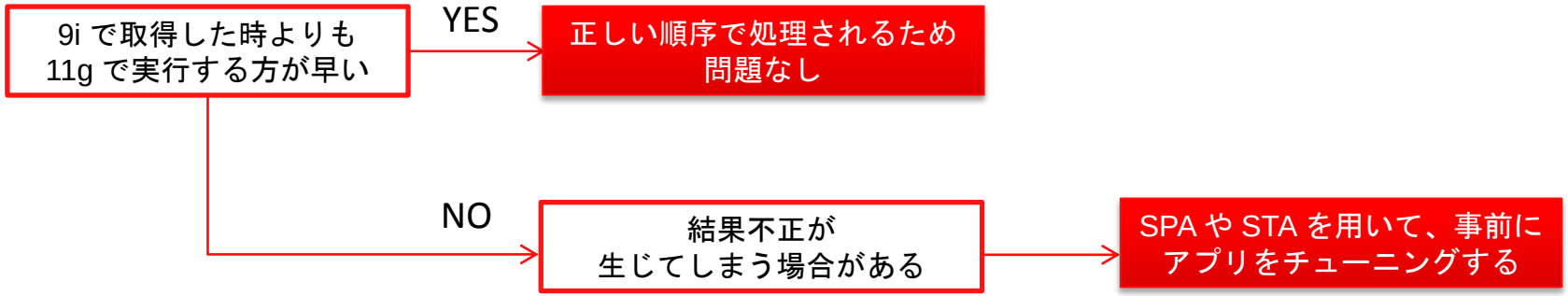


75%に達しない場合は書き出されない

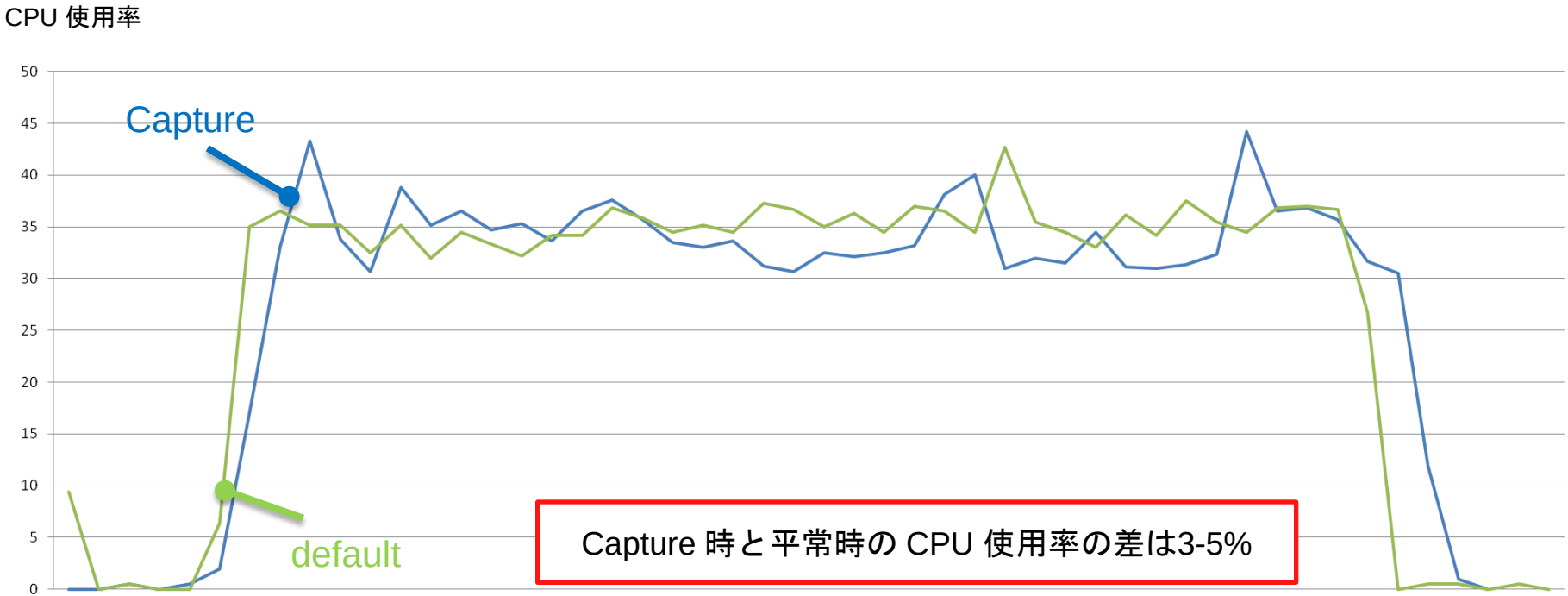
PL/SQL の仕様による注意点

- PL/SQL やストアド・プロシージャなどは、一つ一つの SQL が Capture されるわけではなく、コールのみが Capture される
  - 下記のようなプログラムを組んでいる場合、Replay 時に影響が出ると考えられる
    - 時間で制御しているプログラム
    - 順序性を担保していないプログラム

例) 「Batch A が終わった数秒後に、Batch B を流す」というようなロジックを組んでいる場合

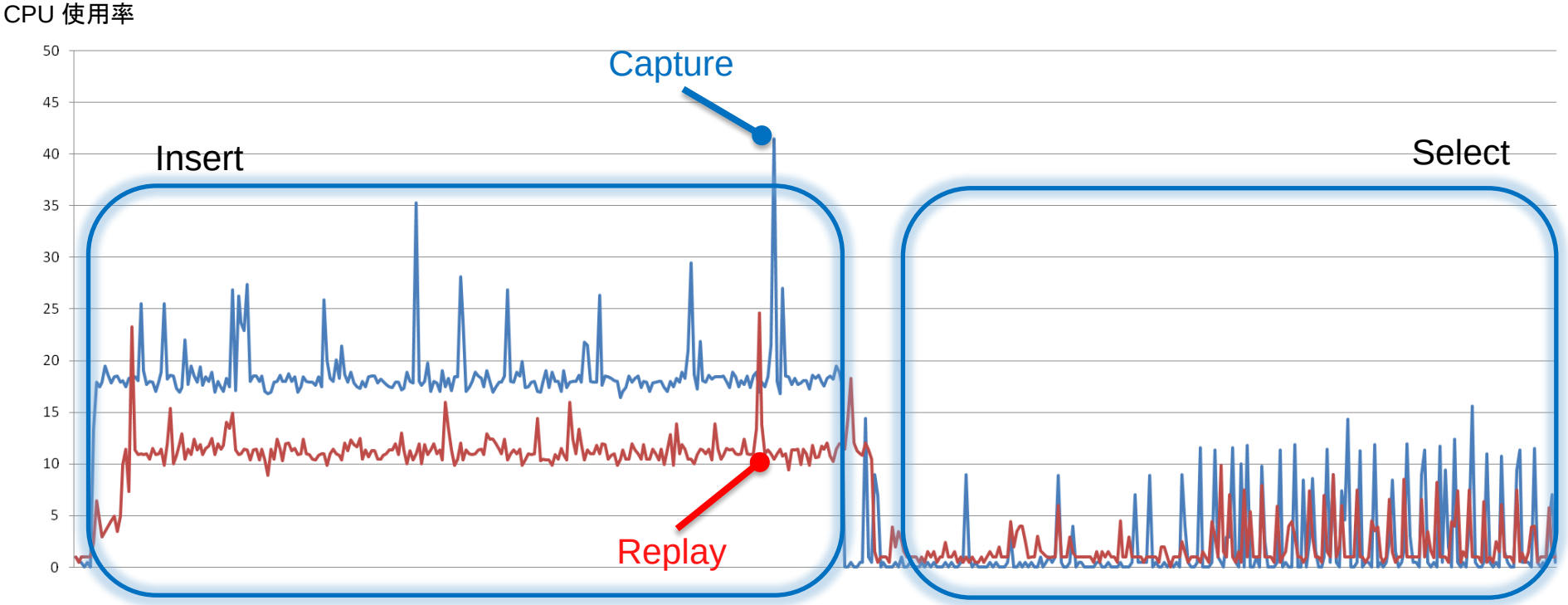


Capture 時と default 時の CPU 使用率の比較

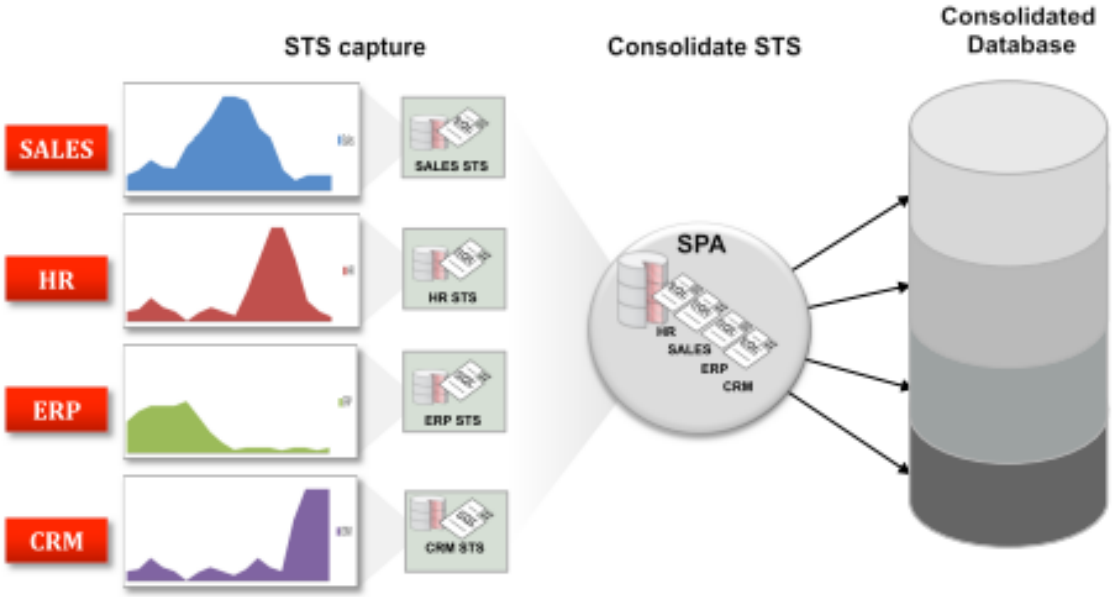


Capture 時と Replay 時の CPU 使用率の比較

同一環境での Capture & Replay 検証結果



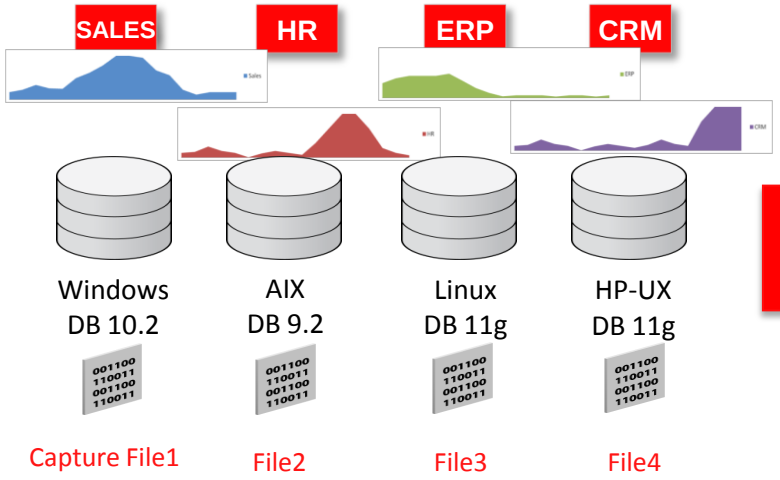
Consolidated SQL Performance Analyzer

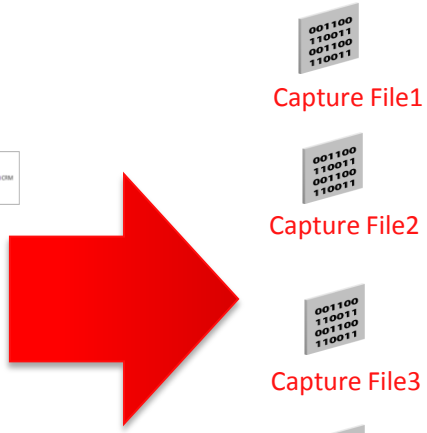


- それぞれの DB で取得した STS を一つにまとめる
- 統合環境において、全てのワークロードの SPA 実行が可能

Consolidated Database Replay

本番環境





テスト環境

- 異なるデータベースのCapture File を統合先の環境で同時に Replay 可能
- スキーマ単位での統合の際に使用 (Server 統合や OS 統合の際は通常の DB Replay を使用)
- アプリケーション間のスケーラビリティや同時実行における問題も識別し、修正可能

まとめ

まとめ

RAT を利用することで・・・

- アップグレードや DB の各種変更に対するテスト工数が削減可能
- 特別なスキルを必要とせず、チューニングすべき SQL の特定が可能
- SQL 単体の性能テストと、ワークロードを再現するテストの際に有効

ぜひ一度 RAT を使ってみてください！

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®

(ご参考) GoldenGate Veridata

データの整合性チェック

- Veridata Server
  - データ比較エンジン、Veridataタスクの取りまとめ
  - データの比較、レポートの生成、out-of-syncデータの確認
- Veridata Web
  - Rich Client /WebBrowserBaseのGUI
  - 比較するジョブのオブジェクトおよびルールの設定
  - レポートおよびout-of-syncデータの参照
- Veridata Agent
  - データの取得
- Veridata Repository
  - 設定情報を保持するデータベース
- Veridata Command Line Interface
  - GoldenGate Serverのコマンドライン・クライアント