津島博士のパフォーマンス講座
Autonomous Databaseのパフォーマンス

津島 浩樹
マスタープリンシバルソリューションエンジニア
クラウド・エンジニアリング統括 COE本部
データベース・ソリューション部

May 20, 2022

 # oradev22

Agenda

1. Autonomous Databaseとは
2. パフォーマンス・メリット
3. 手動チューニング
4. パフォーマンス・ツール

Autonomous Databaseとは

Autonomous Databaseとは

機械学習で実現するフルマネージド・クラウド・データベース

長年実績のあるインフラとデータベースを基盤とし、その管理やオペレーションを完全に自動化

インフラの完全な自動化

データベースの完全な自動化

データセンターオペレーションの自動化と機械学習

データベース技術の自動化

データベースの革新とリーダーシップを推進

9i
- Automatic Query Rewrite
- Automatic Undo Management

10g
- Automatic Memory Management
- Automatic Segment Space Mgmt
- Automatic Statistics Gathering
- Automatic Storage Management
- Automatic Workload Repository
- Automatic Database Diagnostic Monitor

11g
- Automatic SQL Tuning
- Automatic Workload Capture/Replay
- Automatic SQL Plan Management
- Automatic Capture of SQL Monitor
- Automatic Data Optimization

12c
- Autonomous Health Framework
- Automatic Diagnostic Framework
- Automatic Refresh of Clones

18c
- Automatic In-Memory population
- Automatic Columnar Flash Cache
- Automatic Application Continuity

19c
- Automatic Indexes
- SQL Quarantine
- Real-Time Statistics

21c
- Automatic Materialized Views
- Automatic Zone Maps
- Automatic SQL Tuning Set

データベース・インフラの自動化と最適化

Exadataが提供する差別化されたプラットフォーム

2008

- Smart Scan
- Infiniband Scale-Out

- Database Aware Flash Cache
- Storage Indexes
- Columnar Compression

- IO Priorities
- Data Mining Offload
- Offload Decrypt on Scans

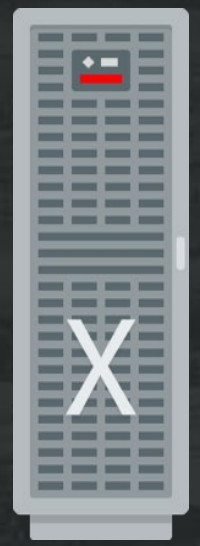
- Network Resource Management
- Multitenant Aware Resource Mgmt
- Prioritized File Recovery

- In-Memory Fault Tolerance
- Direct-to-wire Protocol
- JSON and XML offload
- Instant failure detection

- In-Memory Columnar in Flash
- Smart Fusion Block Transfer

2021

- Exadata Cloud Service



ワークロードとデプロイメント方式

共通のプラットフォームで特定のワークロードに最適化

AUTONOMOUS DATABASE (ADB)

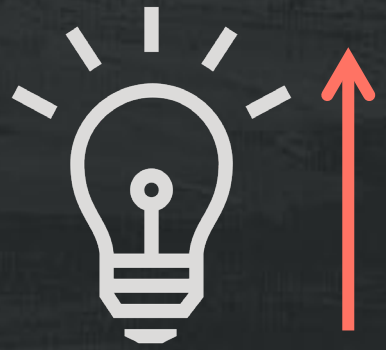
ワークロード	**Autonomous Data Warehouse (ADW)**	**Autonomous Transaction Processing (ATP)**	**Autonomous JSON Database (AJD) (*2)**	**APEX Application Development (*2)**
	(データマート/DWH)	(OLTP/混在ワークロード)	(JSONメイン)	(APEXメイン)
デプロイメント	**Shared Exadata Infrastructure**		**Dedicated Exadata Infrastructure**	
	(共有環境：Pluggable Database)		(専有環境：OCI or C@C(*1))	

*1: C@C: Oracle Cloud@Customer(お客様データセンターでのOCI提供サービス)
*2: Dedicated では未提供

ADBの主なメリット

支出を減らす

- 管理コストの削減
- ランタイム・コストの削減



より革新的に

- 人材の再配置 (管理者から)
- 開発スピードの向上

リスクの低減

- サイバー攻撃の防止
- 常時稼働 (ダウンタイムなし)
- 実績、移行の容易さ

ADBの主なメリット

何をどのように

プロビジョン

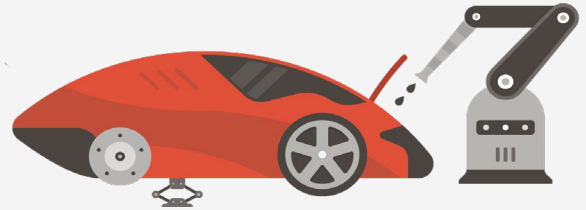
ミッションクリティカルなデータベースを
迅速かつ容易に作成可能

Exadataクラウド・インフラ、
RACスケールアウト・データベース、
Active Data Guardスタンバイを
構築

安全

外部・内部の
あらゆる脅威からデータを守る

セキュリティのオンライン
アップデート適用、DB Vaultで
管理者の盗み見を防止、
全データを暗号化



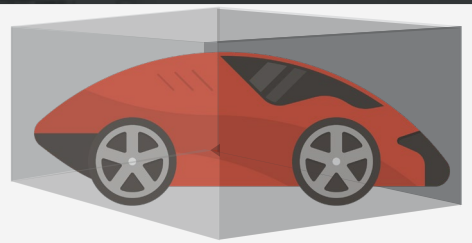
管理

すべてのインフラとデータベースの
メンテナンスを自動化

全てのソフトウェアのオンライン
パッチ適用、設定のチューニング、
すべてのOSとSYSDBA操作の
実行、エラーの診断

ADBの主なメリット

何をどのように



保護

あらゆる障害から
ダウンタイムなしに回復する

バックアップ、リストア、クラスタ内
またはリモートスタンバイへの
アプリケーション透過的フェイルオーバー
の自動化

スケール

オンライン拡張で最高の
パフォーマンスと低コストを実現

コンピューティングとストレージ
の即時、自動、オンライン・
スケーリングにより、
真の従量課金が可能

最適化

人の指示なしに
ワークロードを最適に実行

データフォーマット、インデックス、
並列処理、プランをワークロード
ごとに自動で最適化

Autonomous Databaseの パフォーマンス・メリット

ADBのパフォーマンス・メリット

問題が発生しないように事前設定されているのが特徴

- Exadataによる性能向上
- ADWとATPに対して最適な事前設定
- 代表的なパフォーマンス問題の対応
  - オプティマイア統計
    - 正確にすると収集負荷が増える（正確に取集せずにヒントなどで調整している）
      → 自動的に収集されるように事前設定
  - リソース制御
    - 同時実行での制御が難しい（シリアル実行している場合が多い）
      → 接続サービスごとに事前設定
  - SQLチューニング
    - SQLの書き換えや索引作成などによる最適な実行計画に簡単にできない（経験のある方でないと難しい）
    - SQLの数が多いと対応できない
      → 自動チューニング

Exadataによる性能向上

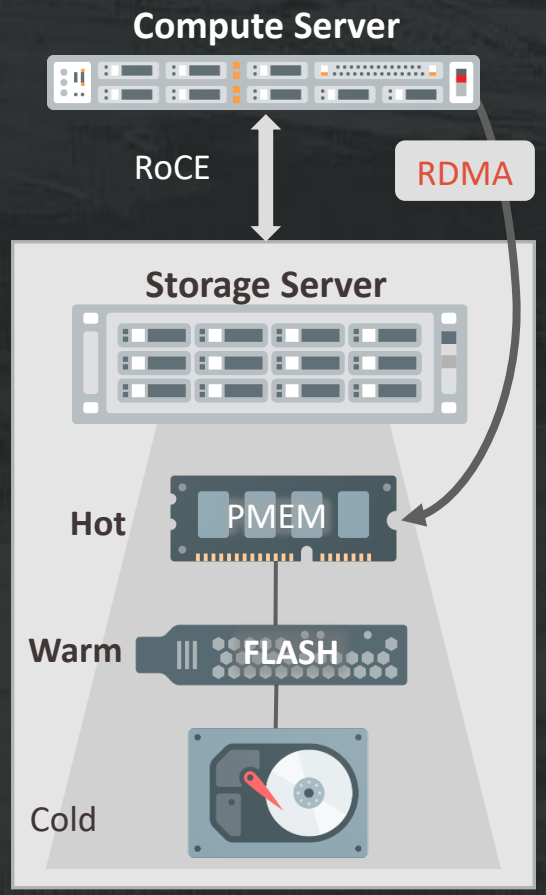
世界最速のデータベース・プラットフォーム

ADW

- I/Oの高速化
  - RoCE, ASM, Exadata Smart Flash Cache
- I/O&CPU使用率の削減
  - Exadata Smart Scan (一部をストレージにオフロード)
    - Storage Index (クエリ対象外のブロックを読み飛ばし)
  - Hybrid Columnar Compression (高圧縮率を実現)
  - In-Memory Columnar in Flashにより更に向上

ATP

- データ・アクセスの高速化
  - Persistent Memory Data Accelerator
  - Exadata Smart Flash Cache
- Redoログ書込みの高速化
  - Persistent Memory Commit Accelerator
  - Exadata Smart Flash Log



RDMAによる通信オーバーヘッドを解消

- 100 Gb/sec の RoCE (RDMA over Converged Ethernet)

Persistent MemoryによるI/Oレイテンシーの削減

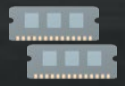
- トランザクション処理のレイテンシーを10倍改善
- 2.5倍のトランザクション処理IOを実現

SQLをストレージにオフロードすることにより、ストレージのボトルネックを解消

Intel® Optane™ DC Persistent Memory

ADWとATPに対して最適な事前設定

ワークロードに応じた最適化

	Autonomous Data Warehouse (ADW)	Autonomous Transaction Processing (ATP)
	すべてのデータ分析処理に最適化 データウェアハウス、データマート、データレイク、機械学習	**OLTPおよび混合ワークロードに最適化** トランザクション、バッチ、レポーティング、IoT、アプリケーション開発、機械学習
	列型フォーマット	行型フォーマット
	データ集計を作成	インデックス作成
	インメモリ結合、集計	I/O削減のためのデータキャッシュ
	オプティマイザ統計をリアルタイム/高頻度で更新し、実行計画の劣化を防止	

ADWとATPに対して最適な事前設定

ADWとATPの主な違い

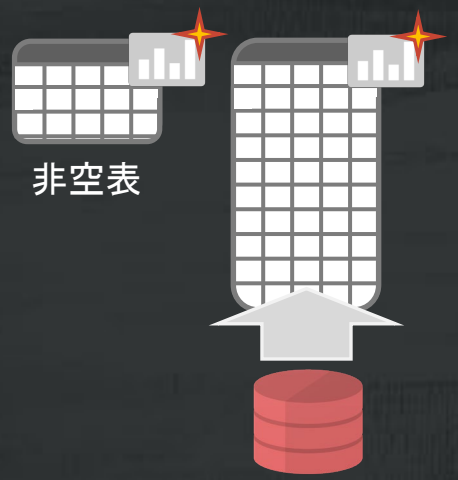
	ADW	ATP
メモリ設定	集計・ソート処理を優先する設定（SGA < PGA） INMEMORY_SIZE=1G	データのキャッシュを優先する設定（SGA > PGA） INMEMORY_SIZE=0
圧縮	有効（表領域レベルでHCC、表毎に無効化可能）	無効（表毎に圧縮設定可能）
Result Cache	全てのSQLに有効（ヒントで無効化可能）	デフォルトのまま（無効）
ヒント	無効（セッション単位で有効化が可能） NO_RESULT_CACHEヒントを除く	デフォルトのまま（有効）
パラレル処理	接続サービス毎に選択可能 ・High : 自動的にパラレル処理 ・Medium : 自動的にパラレル処理 ・Low : 無効	接続サービス毎に選択可能 ・TPURGENT : 手動で設定 ・TP : シリアル処理 ・High : 自動的にパラレル処理 ・Medium : 自動的にパラレル処理 ・Low : 無効
自動オプティマイザ統計収集	無効 高頻度自動オプティマイザ統計収集は有効	有効

代表的なパフォーマンス問題の対応

オプティマイザ統計（基本は手動取集は不要）

オンライン統計収集

- バルク・ロードのオンライン統計収集 (ダイレクト・ロード操作に対する統計情報の自動収集)
  - CREATE TABLE … AS SELECT …;
    INSERT /*+ append */ INTO … SELECT …;
  - Data PumpのImportロード
  - DBMS_CLOUDのロード
- 非ADBとは異なり、以下も動作する
  - 表が空でなくても可能
  - ヒストグラムの収集



リアルタイム統計

- 従来型DMLのオンライン統計収集
  - オプティマイザ統計のサブセットを収集
    - 行数、列の最大値、最小値など
- 21c(RU19.10)からデフォルトFALSE
  - optimizer_real_time_statisticsが追加
    - 自動索引機能など

自動オプティマイザ統計収集

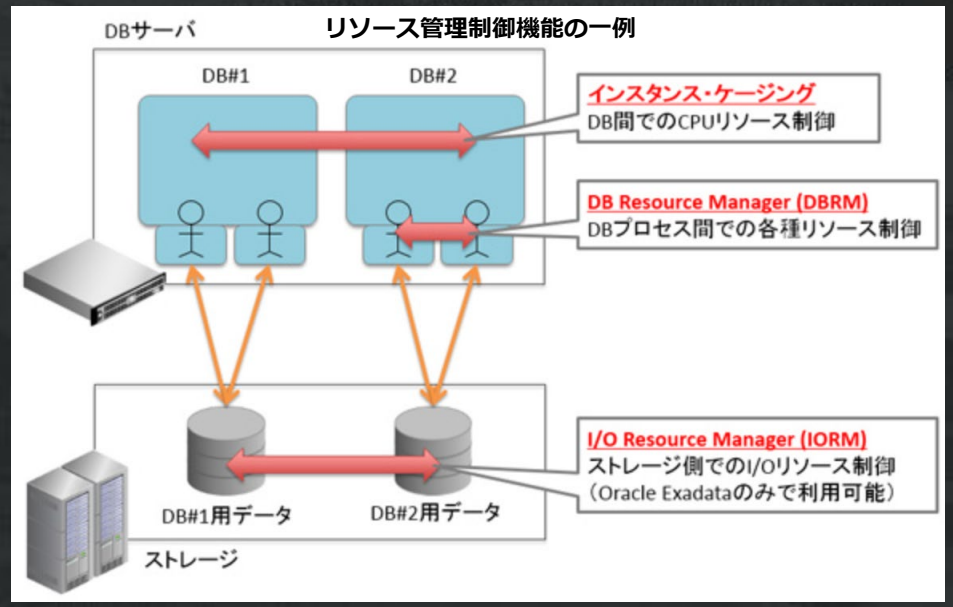
- ADWは無効
  - DWHで従来型DMLはあまり使用しない
    - ETLなどで使用する場合は一部として収集を行う
- 高頻度自動オプティマイザ統計収集
  - 非ADBと異なりデフォルト有効
    - 統計情報が古い場合、15分ごとに完全なオプティマイザ統計を収集する

代表的なパフォーマンス問題の対応

リソース制御

リソース管理機能群 (インスタンス、プロセス、I/O) を利用しやすい形で提供

- ミッションクリティカルなシステムに求められる、同時実行数制御や、CPU・ネットワーク・ストレージ帯域管理、レイヤー毎の自動キャッシュ
- DWH系システムに求められる、SQL実行の自動パラレル処理、自動キューイングなど
- ユーザーは接続サービスを選択するだけ



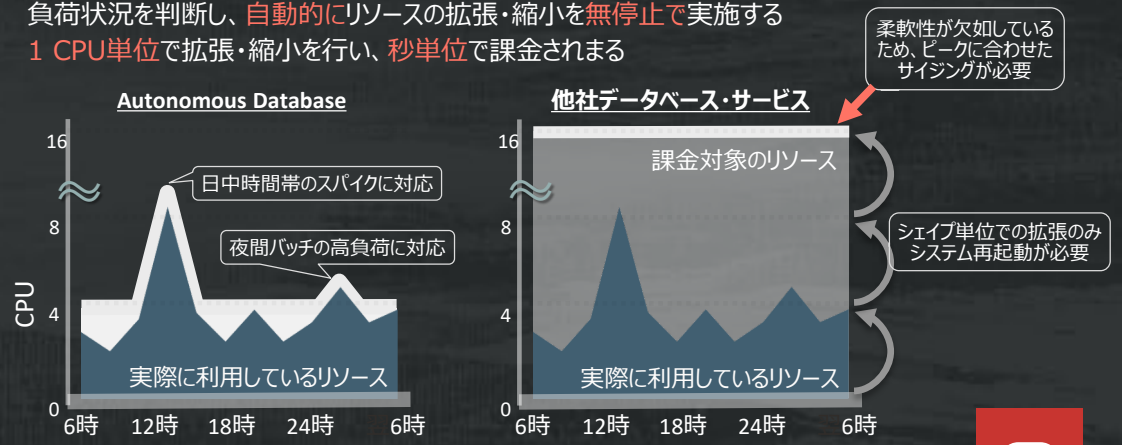
CPUのオンライン自動スケーリング

煩雑な作業なしに、負荷状況に応じてCPUリソースを柔軟に利用可能

- CPUを動的に割り当てることで性能を担保
  - アプリケーションの停止は不要
- 1 OCPU単位での柔軟なリソース配分により大幅なコストダウンが可能
  - 負荷のピークに合わせたサイジングは不要
- ユーザーはOCPU数と自動スケーリングの利用有無を選択するだけ

※1 OCPU (Oracle Compute Unit) = 1物理コア = 2vCPU

負荷状況を判断し、自動的にリソースの拡張・縮小を無停止で実施する
1 CPU単位で拡張・縮小を行い、秒単位で課金されまる



代表的なパフォーマンス問題の対応

リソース制御（接続サービスに対して事前設定）

インスタンス作成時点で、接続サービスが定義されている

- アプリケーションは、事前定義された5つのデータベースサービスのいずれかに接続
- リソース・マネージャのコンシューマ・グループにマッピングされ、優先度、同時実行性、並列処理を制御

インスタンス接続時に、接続サービスを選択するだけでリソース制御が可能(*1)

接続サービス	platform	概要	パラレル制御 (DOP)	同時実行 セッション数	リソース割り当て (SHARES*2)
TPURGENT	ATP only	最も優先度の高い処理向け	手動設定	300 x OCPUs	12
TP	ATP only	汎用的な処理向け	シリアル	300 x OCPUs	8
HIGH	ADW/ATP	大量データを扱う処理向け	自動(最大はOCPUs)	3	4
MEDIUM	ADW/ATP	大量データを扱いつつも、同時実行数も多い処理向け	自動(*3)	1.26 x OCPUs (*4)	2
LOW	ADW/ATP	優先度が低い処理向け	シリアル	300 x OCPUs	1

*1：5分以上アイドルのセッションはリソース不足時の削除対象になる
*2：サービス間でのCPU配分の相対的な優先度を示す。デフォルト値からの変更は可能
*3：OCPU<4の場合はOCPU数、OCPU≧4の場合は4が最大
*4：OCPU<4の場合は5、OCPU≧4の場合は1.26 x OCPU数
MEDIUMのみ同時実行セッション数の上限を変更可能（後述）

代表的なパフォーマンス問題の対応

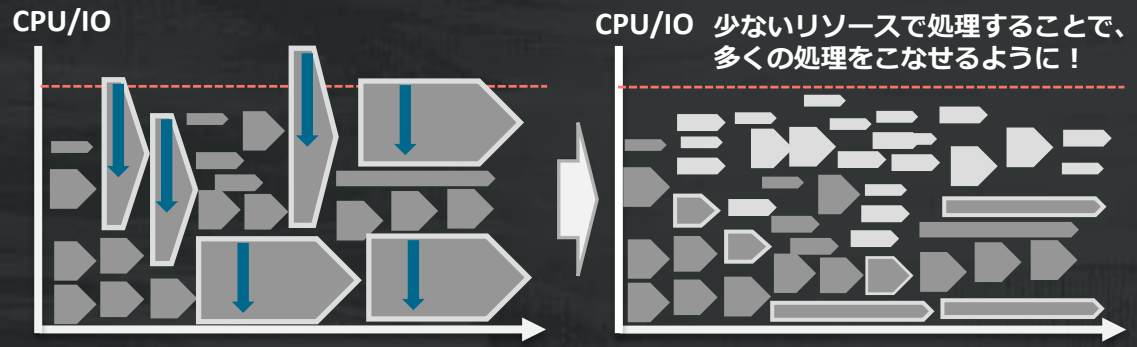
リソース制御（接続サービスの選択）

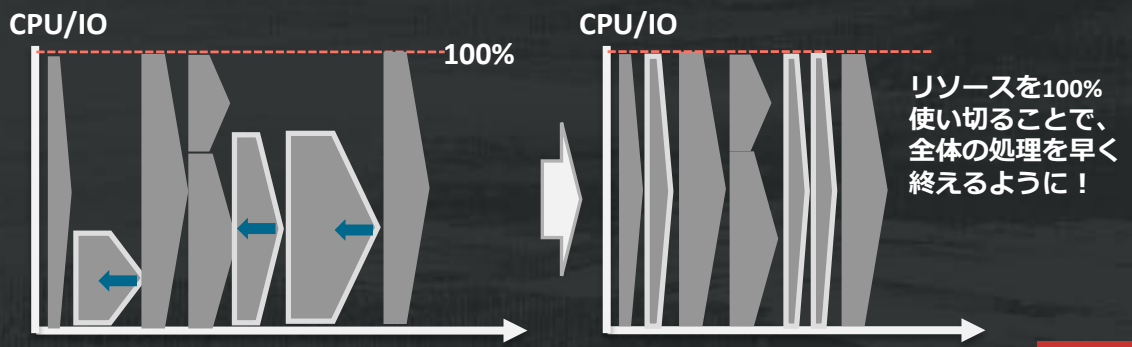
OLTPの特徴

- 少量の行にアクセスし、大量のユーザを同時実行する
- 一般的にはミリ秒レベル
- スループット (TPS) を重視し、単体処理のリソース利用の極小化を目指す
- TPの利用を推奨
  - 単一のCPUコアで処理させるため
  - 最優先したい特別な処理(もしくは手動でパラレル度を制御したい)の場合はTPURGENTを使用

バッチ・DWHの特徴

- ユーザ数は少なく、大量の行にアクセスし一括処理する
- 一般的には秒～分レベル
- 単体SQLのレスポンスタイムを重視し、単体処理でCPU、IOリソースを100%割り当てることを目指す
- MEDIUMの利用を推奨
  - 複数のCPUコアで処理させるため（動的なパラレル処理以外にキューイングも実装されていて効率よく処理可能）
  - 同時実行数が 3よりも少ない場合はHIGHを推奨





代表的なパフォーマンス問題の対応

リソース制御（ダイレクト・パスとコンシューマグループ）

ダイレクト・パスINSERTが使用される場合

- パラレルDMLとき
- ヒント/*+ append */を付加したとき
  - optimizer_ignore_hints = FALSE に設定 (ADWのデフォルトはTRUE)
  - シリアル(LOW)はヒントがないと従来型INSERT

パラレルDML(PDML)はデフォルトで有効

- CPUコア数＞1、かつコンシューマ・グループがMEDIUMまたはHIGHの場合のみ

```
-----------------------------------------------------------
| Id  | Operation                          | Name       |
-----------------------------------------------------------
|   0 | INSERT STATEMENT                   |            |
|   1 |  LOAD AS SELECT                    | TAB1_2     |
|   2 |   PX COORDINATOR                   |            |
|   3 |    PX SEND QC (RANDOM)             | :TQ10000   |
|   4 |     OPTIMIZER STATISTICS GATHERING |            |
|   5 |      PX BLOCK ITERATOR             |            |
|   6 |       TABLE ACCESS STORAGE FULL    | TAB1       |
-----------------------------------------------------------
   - automatic DOP: Computed Degree of Parallelism is 8
```

HIGH

MEDIUM

```
-------------------------------------------
| Id  | Operation                  | Nam    |
-------------------------------------------
|   0 | INSERT STATEMENT           |        |
|   1 |  LOAD TABLE CONVENTIONAL   | TAB1_2 |
|   2 |   TABLE ACCESS STORAGE FULL| TAB1   |
-------------------------------------------
   - automatic DOP: Computed Degree of Parallelism is 1 because of no
     expensive parallel operation
   - PDML disabled because object is not decorated with parallel clause
   - Direct Load disabled because no append hint given and not executing
     in parallel
```

LOW

代表的なパフォーマンス問題の対応

SQLチューニング

自動チューニング

- SQL自動変換
  - オプティマイザにより可能な限り自動的に最適なSQLに書き換えるQuery Transformation
- 様々なチューニング項目（索引、パーティション、MVIEW）の自動化
  - OLTPの高速化のための自動索引（デフォルト無効）
    - 索引はADWでも使用できる
  - 分析系処理の高速化のための自動パーティション、自動MVIEW（デフォルト無効）
- 実行計画管理の自動化
  - SQLの性能劣化を検出し修復する過程をすべて自動化
    - 自動SQLチューニングセット（デフォルト無効）、自動SQL計画管理（ベースライン設定のSQLのみデフォルト有効）、自動SPM展開アドバイザ（デフォルト有効）、高頻度自動SPM展開アドバイザ（デフォルト無効)

Query Transformationと自動SPM展開アドバイザ以外はADBとExadataのみで使用できる

代表的なパフォーマンス問題の対応

自動チューニング（SQL自動変換）

最適なSQLにするのが難しい

- 高度なスキルや経験・ノウハウが必要

オプティマイザにより可能な限り自動的に書き換え、
開発者はアプリケーションの実装に専念できるようにする

- 効率の良いSQLへ自動で変換、より高速に処理

• OR拡張	• インメモリ集計 (VECTOR GROUP BY)
• ビューのマージ	• cursor-duration一時表
• 述語のプッシュ	• 表拡張
• 副問合せのネスト解除	• 結合の因数分解　等
• MVIEWへのクエリー・リライト	
• スター型変換 (索引がないと動作しない)	

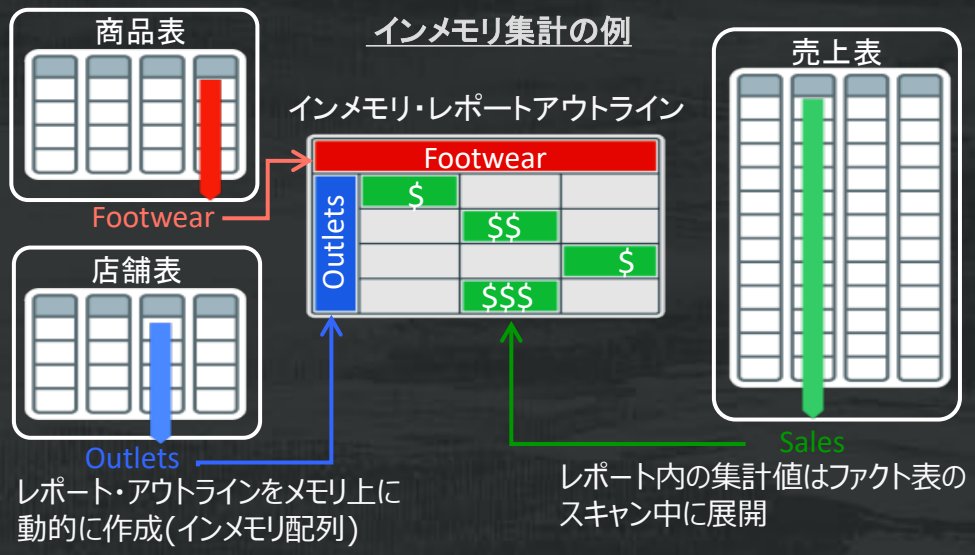
- ユーザーは特に意識する必要はない

OR拡張の例

```
SELECT * FROM employees e, departments d
WHERE  (e.email='SSTILES' OR d.department_name='Treasury')
AND    e.department_id = d.department_id;
```

```
SELECT * FROM employees e, departments d
WHERE  e.email = 'SSTILES'AND e.department_id = d.department_id
UNION ALL
SELECT * FROM employees e, departments d
WHERE  d.department_name = 'Treasury' AND e.department_id = d.department_id;
```

インメモリ集計の例



代表的なパフォーマンス問題の対応

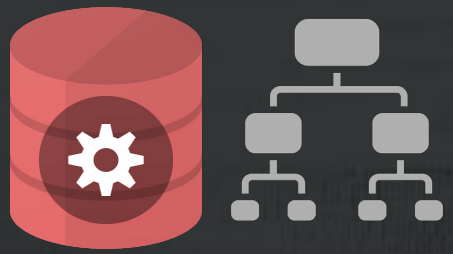
自動チューニング（自動索引、自動パーティション、自動MView）

高スキルなエンジニアによる探索的なチューニング

- エンジニアのノウハウに依存し、コストをかけて実施

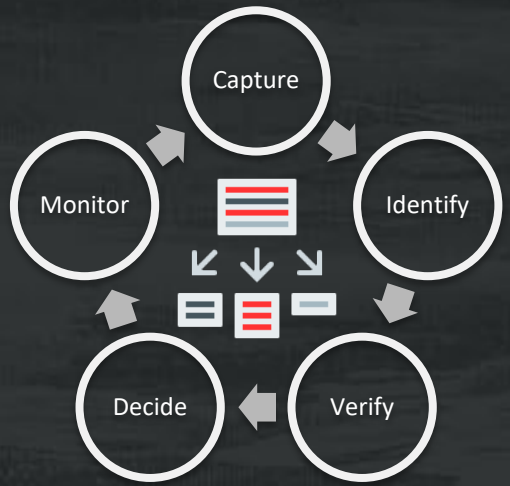
熟練のエンジニアによる一連のプロセスと同等のことを機械学習を活用し自動化

- 最新のOracle Database 21cの機能も先行して取り込み、様々なチューニング項目を順次自動化
  - OLTPの高速化のための自動索引（デフォルト無効）
  - 分析系処理の高速化のための自動パーティション、自動MVIEW（デフォルト無効）
- 最新の統計情報に基づき、最適な設定を断続的に探索、自動選択することで性能改善を図る
- ユーザーは利用するか否かを選択するだけ



自動パーティション

- 熟練したパフォーマンス・エンジニアが行うようなパーティショニング方法を実装する
- パーティショニング手法の候補を特定し、検証を行う
- プロセス全体を完全に自動化することが可能
- 透明性は高度な自動化と同様に重要
- すべての推奨事項はレポートにより監査可能

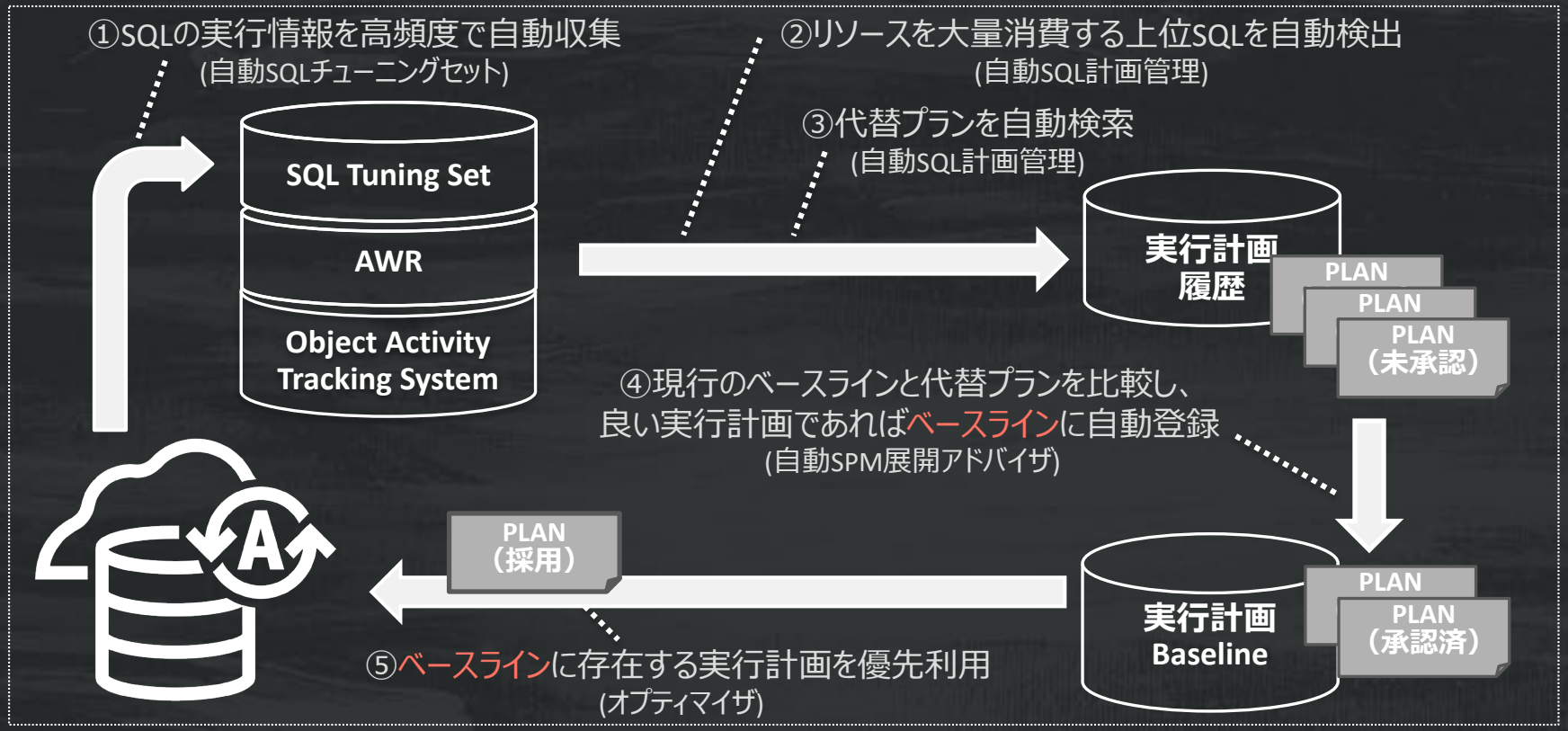


代表的なパフォーマンス問題の対応

自動チューニング（実行計画管理の自動化）

実行計画管理の全てのステップを自動化

- 最新の統計情報、データ構成を機械的に都度収集し、断続的に適用テストを行い実行計画の安定化を実現
- データ量の変動やパッチ適用、バージョンアップに伴うアプリケーションの性能劣化を抑制



さらに、①は以下にも活用

- 自動索引 [19c]
- 自動パーティション[19c]
- 自動ゾーンマップ [21c]
- 自動Materialized Views [21c]

Autonomous Databaseの手動チューニング

ADBの手動チューニング

- できること/できないこと
- よくあるパフォーマンス・トラブル

ADBの手動チューニング

できること/できないこと

できないこと

- インメモリ列ストアの使用
  - ADWはFlash上で使用
- DBブロック・サイズ (8K) の変更
- メモリ・チューニング

できること

- CPUコア数のスケールアップ
- 別のサービスを利用する
  - MEDIUMの同時実行数も変更可
- 表の圧縮/非圧縮の変更
- 制約の設定/変更
  - 例: 外部キー制約を使用して結合の排除 (クエリ変換)
- オプティマイザ統計の再収集
  - 自動チューニングの限界
- ヒントの追加
  - ADWはデフォルト無視
- インデックスの作成
- パーティション表の作成
- マテリアライズド・ビューの作成

ADBの手動チューニング

MEDIUMサービスの同時実行セッション数の設定

MEDIUMサービスの同時実行セッション数の上限 (CONCURRENCY_LIMIT)

- アプリケーションの特性に合わせた調整が可能
  - パラレル処理しながら、より多くの処理を実行したい (スループット優先)
  - 同時実行数は最小に、HIGHよりも多いDOPにしたい (レスポンス優先)
    - OCPU数よりDOPを増やしたいなど
- 特記事項
  - MEDIUMサービスでのみ変更可能
  - OCPUが2以上の場合に変更可能
  - 同時実行制限の変更に伴い、DOPも変更される

	デフォルト値	変更範囲
同時実行数上限	ベースOCPUに紐づく • OCPUが4未満の場合：5 • OCPUが4以上の場合：1.26 x OCPU数	1 から 3 x OCPU数の範囲で指定可能
DOP (パラレル度)	ベースOCPUに紐づく • OCPUが4未満の場合：OCPU数 • OCPUが4以上の場合：4	同時実行数上限をデフォルトから変更した場合、 DOPは同時実行数上限値に反比例する形で変動する。 またこの場合、AutoScalingの有無によってDOPの上限が変わる • Auto Scaling 無効時は、2から2xOCPU数の間でDOPが変動 • Auto Scaling 有効時は、2から6xOCPU数の間でDOPが変動

ADBの手動チューニング

自動チューニングの限界（オプティマイザ統計）

オプティマイザ統計をすべて最適に収集するわけではない

- 拡張統計（列グループ、式）は自動作成されない
- ヒストグラムのバケット数は最大254
- 適応統計は動作しない
  - 単一表カーディナリティだけ
  - パラレル実行時にダイナミック統計は動作する

自動SQL計画管理は存在した実行計画に改善するだけ

- オプティマイザ統計が正確でないと最適な実行計画が生成できない場合も
  - バージョンアップなどによる性能劣化の場合に使用するもの

対応策として以下が必要

- ヒントで最適な実行計画を生成させる
- 拡張統計の作成やヒストグラムの最大値を変更する
- 最適なパフォーマンスではないが遅くなければ問題ないという考え方もあり
  - どうしても遅いSQLにはCPUのスケールアップで対応する

ADBの手動チューニング

よくあるパフォーマンス・トラブル

- Smart Scanにならない
- TRUNCATE/DROP TABLEが遅い
- INSERT文が遅い
- パラレル実行されない
- Temp領域の使用
- DELETE/UPDATE/MERGE文が遅い
- 実効計画が変化して遅くなった

よくあるパフォーマンス・トラブル

Smart Scanにならない

Smart Scanの待機イベン

- AWRやSQLモニターで確認
- cell smart table scan、 cell smart index scan

direct path read待機イベントになっている

- Smart Scanの条件を確認する（第69回）
  - ヒープ表 (通常の表) でない (クラスタ表／索引構成表である)
  - HCC以外で255 を超える列を参照
  - 非インラインLOB (4000バイト超過) が参照されている
  - 行連鎖しているなど

Cell single block physical read待機イベントになっている

- 索引スキャンのためオプティマイザ統計を確認する
- パラレル実行してみる
  - 索引スキャンがなくなる場合がある
- 索引をINVISIBLEにしてみる
  - アクセス件数が多い場合は効果がある場合も
- ヒントを無効にしてみる (ADWのデフォルトは無視)
  - INDEXヒントで強制的に索引スキャンにしている場合も

よくあるパフォーマンス・トラブル

Smart Scanにならない

cell multiblock physical read待機イベントになっている

- 表サイズ (アクセス・パーティション合計サイズ) の確認
  - アクセス・サイズがバッファ・キャッシュの2%以下だと非ダイレクト
    - オプティマイザ統計の確認も (11.2からセグメント・サイズはオプティマイザ統計を使用する)
  - In-Memory Parallel Execution (バッファ・キャッシュ上のパラレル実行) が動作している
    - バッファ・キャッシュの80%以下のサイズのオブジェクトが動作する
- パラレル実行（MEDIUMまたはHIGH）にしてみる
  - パラレル実行は Direct Path Read (Smart Scan) になりやすい
  - シリアル実行はバッファ・キャッシュの状態による（大量更新のバッチ処理後には注意）
    - 同時実行している場合が多く、ミニ・チェックポイントが発生するため
    - ALTER SYSTEM FLUSH BUFFER_CACHE を実行してみる

よくあるパフォーマンス・トラブル

TRUNCATE/DROP TABLE(PARTITION)が遅い

大量ブロックのミニ・チェックポイントが動作した

- enq: RO - fast object reuse 待機イベント
  - enq: KO - fast object checkpoint はダイレクト・パス・リードのとき
- ADB (Exadata HC Storage) はディスクに強制書込み
- 大量更新後の実行に注意
  - 通常の運用ではあまり行わないと思う（テストなどではよく発生する）

よくあるパフォーマンス・トラブル

INSERT文が遅い（Direct Path Insertにならない）

シリアル (LOW) だとAPPENDヒントが必要 (パラレルは必要ない)

```
INSERT /*+ APPEND */ * INTO tab1_2 SELECT … ;
```

制約を考慮する必要がある

- 外部キー制約が定義されている場合、PDML/ダイレクト・パスは無効となる
  - 従来型ロードが使用される
- 推奨
  - 信頼性の高い制約 (RELY制約) の定義
    - 制約をTRUE状態とする

```
ALTER TABLE tab1_2
ADD FOREIGN KEY (country_id) REFERENCES countries
RELY DISABLE NOVALIDATE
```

```
-----------------------------------------------------
| Id | Operation                          | Name     |
-----------------------------------------------------
|  0 | INSERT STATEMENT                   |          |
|  1 |  LOAD TABLE CONVENTIONAL.          | TAB1_2   |
|  2 |   PX COORDINATOR                   |          |
|  3 |    PX SEND QC (RANDOM)             | :TQ10000 |
|  4 |     PX BLOCK ITERATOR              |          |
|  5 |      TABLE ACCESS STORAGE FULL     | TAB1     |
-----------------------------------------------------

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 8
   - PDML disabled because parent referential constraints
     are present
```

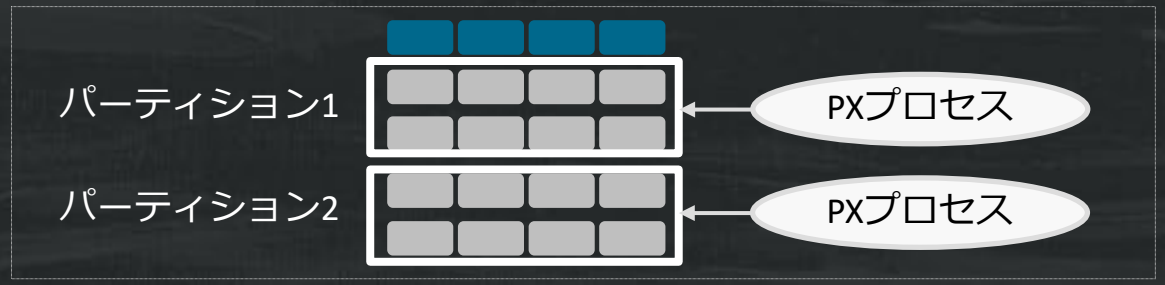
よくあるパフォーマンス・トラブル

INSERT文が遅い（Direct Path Insertが動作）

パーティション表のパラレル・インサート

- ExadataのHCCはHigh Water Mark(HWM) Loadingになる (第50回)
  - セグメントに対して単一PXプロセスで処理する (一つだけのパーティション・ロードはシリアル処理になる)
  - 改善するにはPQ_DISTRIBUTEヒントか Exchange Partitionを使用する

※Exchange Partitionを使用したロード
ダミー表にロードしてからパーティションと交換する (データの移動はない)



- HCC以外はHigh Water Mark Brokering(HWMB) Loadingになる
  - パーティションごとにHVエンキュー (HWMの更新のため) を使用してパラレル・ロードを行う

表がNologgingモードでない

- Redoログにブロック・イメージが出力される
- 表圧縮を検討する（Redoログも削減される）

よくあるパフォーマンス・トラブル

INSERT文が遅い（Direct Path Insertが動作）

ダイレクト・パス・ロード後のインデックス・メンテナンス

- 非常に時間がかかる、パラレル化できない
- 回避策：RELY DISABLE NOVALIDATEで主キーを作成する
  - 終了後にパラレルで作成する

Plan Hash Value 2802573289 | Plan Note | All Parallel Servers

Operation	Name	Line...	Estimated Ro...	Cost	Timeline(47s)	Executi...	Actual Rows	Memory (...	Temp (Max)	Ot...	IO Requests
INSERT STATEMENT		0				9	4				
PX COORDINATOR		1				9	4				
PX SEND QC (RANDOM)	:TQ10003	2	256K	113		4	4				
INDEX MAINTENANCE	H_ORDER	3				4	4				37K
PX RECEIVE		4	256K	113		4	168K				
PX SEND RANGE	:TQ10002	5	256K	113		4	176K				
LOAD AS SELECT (HYBRID TSM/HWMB)	H_ORDER	6				4	8				10
OPTIMIZER STATISTICS GATHERING		7	256K	113		4	0	3MB			

よくあるパフォーマンス・トラブル

パラレル実行にならない

MEDIUMまたはHIGHでもパラレル・クエリが動作しない

- 実行計画を確認 (PX xxxxx操作が存在しない)
  - 索引スキャンを行っている
    - 索引をパーティション化する
  - ソート処理 (Order By) を行っている
    - レンジ分割のためデータがあまり分散しない (できれば止める)

よくあるパフォーマンス・トラブル

Temp領域を使用している場合

Temp I/Oの待機イベントが多い (第68回)

- direct path read temp、 direct path write temp
  - Exadata 12.2からFlash Cacheを使用する

使用するPGAの増加

- パラレル度を増やす/スケールアップする（同時実行のために単一プロセスの最大サイズがある）
  - 同時実行が多いとPGA_AGGREGATE_TARGETを超える (そのためPGA_AGGREGATE_LIMITがある)

PGA_AGGREGATE_TARGET < 1Gバイト	MIN (PGA_AGGREGATE_TARGETの20%, 100Mバイト)
PGA_AGGREGATE_TARGET ≧ 1Gバイト	MIN (PGA_AGGREGATE_TARGETの10%, 1Gバイト)
パラレル実行	MIN (PGA_AGGREGATE_TARGETの50%÷パラレル度, 単一プロセスの最大サイズ)

使用するPGAの削減

- SQLから必要ない列を削除する
- パーティション・ワイズ処理を検討 (パーティション単位に処理が可能) <=第20回, 第45回, 第60回
  - ハッシュ結合, Group By, DISTINCT, 分析ファンクション(18cから), Order By(レンジでシリアル処理だけ)
- SQL文を変更する

よくあるパフォーマンス・トラブル

DELETE/UPDATE/MERGE文が遅い

HCC表に対して行なっている

- 大量の更新は避ける
  - 対象データはOLTP圧縮に変換される
  - 12cから行ロックをサポート

パラレル実行がスケールしない (Redoログ・ネックなど)

- INSERT文に書き換える

```
SQL> DELETE FROM tab000
  2   WHERE 日付 < TO_DATE( '20101001' ,' YYYYMMDD' ) ;
```

```
SQL> CREATE TABLE tab001 NOLOGGING PARALLEL AS
  2    SELECT * FROM tab000
  3     WHERE 日付 >= TO_DATE( '20101001' ,' YYYYMMDD' ) ;
SQL> DROP TABLE tab000 ;
SQL> RENAME tab001 TO tab000 ;
```

よくあるパフォーマンス・トラブル

実行計画が変化して遅くなった

コストベースのクエリ変換が動作

- オプティマイザ統計が正確でないため誤動
  - オプティマイザ統計を正確にする（拡張統計などを作成する）
  - SQLの書き換えやヒントを追加する
  - 何かが変化したタイミングで発生するので、SPM (自動SQL計画管理) の使用を検討する

Autonomous Databaseの パフォーマンス・ツール

パフォーマンス・ツール

AWR（Automatic Workload Repository）

- パフォーマンス・ハブ
- DBMS_WORKLOAD_REPOSITORYパッケージ
  - ADBはawrrpt.sqlスクリプトにアクセスできない
- SQL Developer

SQLモニター

- サービス・コンソール
- パフォーマンス・ハブ
- SQL Developer

パフォーマンス・ツール

AWR（Automatic Workload Repository）

- パフォーマンス・ハブ
- DBMS_WORKLOAD_REPOSITORYパッケージ
  - ADBはawrrpt.sqlスクリプトにアクセスできない
- SQL Developer

SQLモニター

- サービス・コンソール
- パフォーマンス・ハブ
- SQL Developer

パフォーマンス・ツール

データベース稼働状態（サービス・コンソール）

Overview

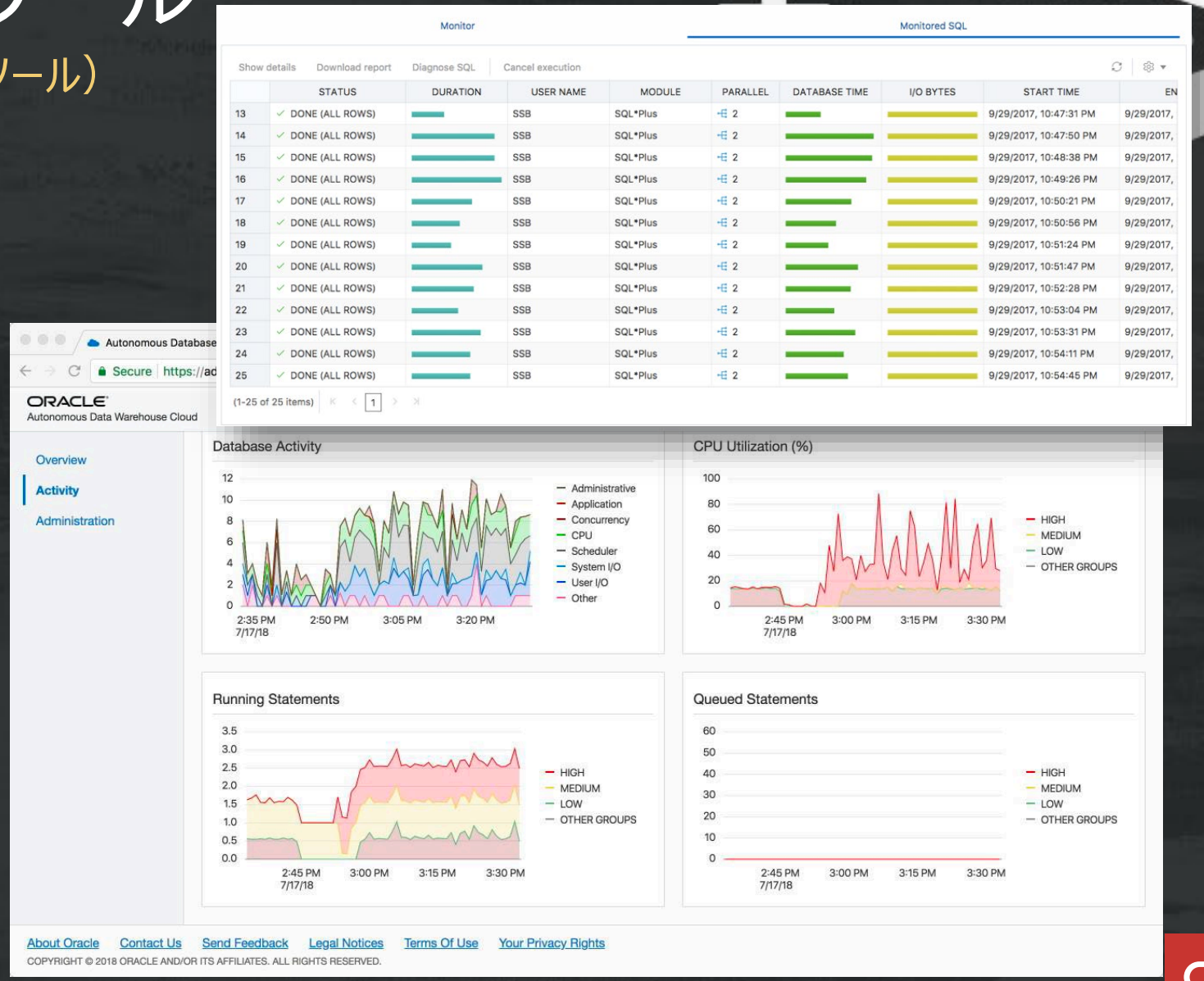
- ストレージ使用状況
- CPU使用率
- 割り当てOCPU数
- SQL実行数
- SQLの平均レスポンスタイム

Activity（モニター）

- 接続サービス毎の負荷状況やキューイング状況
- DBの待機イベント

Activity（モニターSQL対象）

- Show details
  - リアルタイムSQLモニターレポートの表示
- Download report
  - リアルタイムSQLモニターレポートのダウンロード

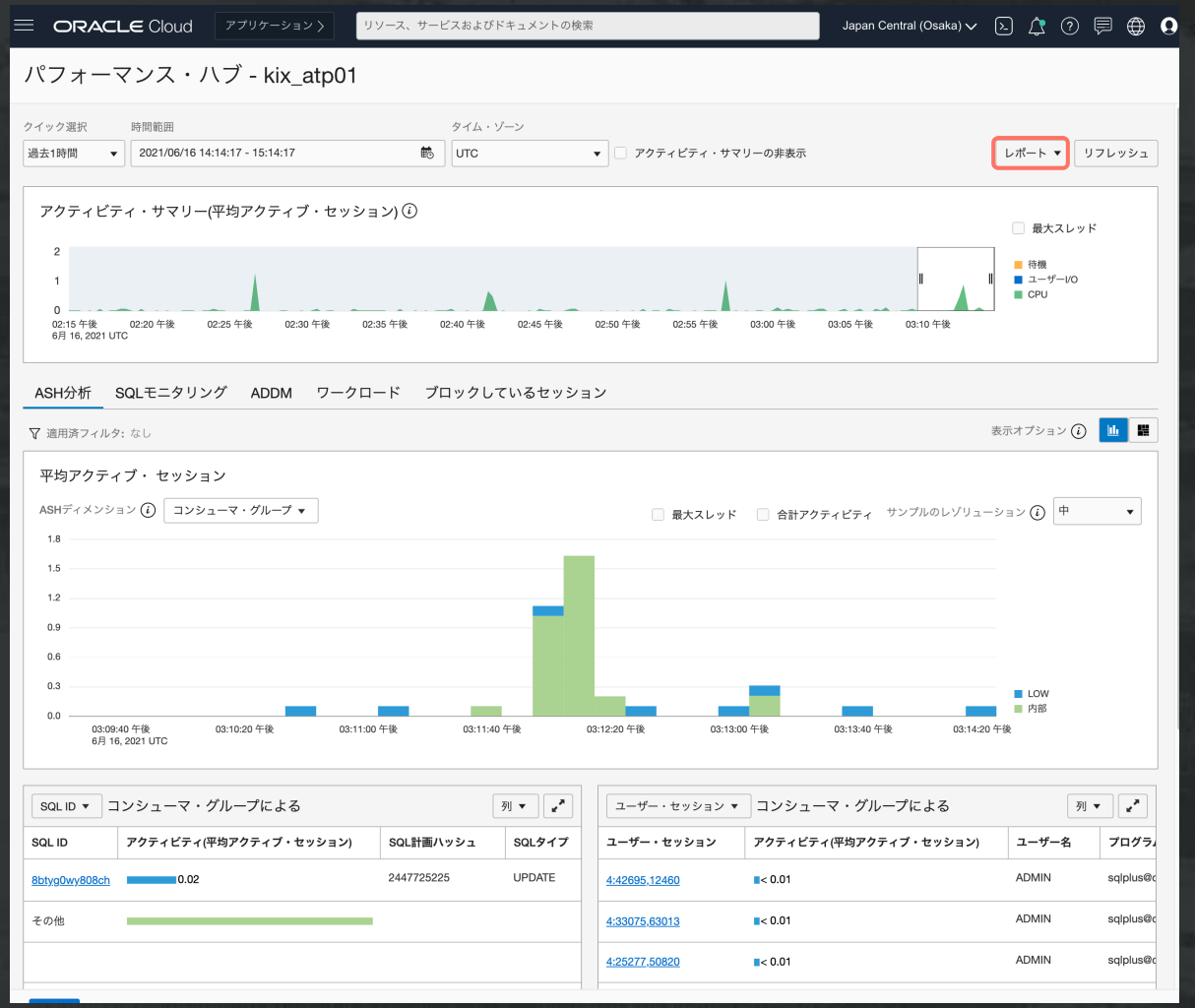


パフォーマンス・ツール

データベース稼働状態（パフォーマンス・ハブ）

- ASH分析
- SQLモニタリング
- ADDM
- ワークロード
  (データベース・ワークロード)
- ブロックしているセッション

- AWRレポートのダウンロード
  （「レポート」から）



最後に

- ADBは、 Exadataベースに最適に事前設定されているが、
  通常のOracle Databaseと同様のパフォーマンス問題は発生するので、
  どんなことが発生するかは知っておいてください
- 過度のチューニングはADBの良さがなくなるので最低限のチューニングにとどめる。そうでない
  と使いづらいOracle Databaseとなってしまう
- それ以外のパフォーマンス向上はスケールアップで対応する

Thank you