Oracle Database Technology Night 夏祭り
～ 集え！オラクルの力（チカラ）～

実演！SQLパフォーマンスの高速化の限界を目指せ

日本オラクル株式会社
クラウド・テクノロジー事業統括
Cloud Platform ソリューション本部
Database ソリューション部
部長　柴田 長

Live Challenge!!

SQLパフォーマンスの高速化の限界を目指せ！

developer.oracle.com

Tsukasa Shibata
Director
Database Technology,
Database & Exadata Product Management
Cloud Technology Business Unit
Oracle Corporation Japan
May 18, 2017

Live for the Code

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

“しばちょう” こと、柴田長(しばた つかさ)です。

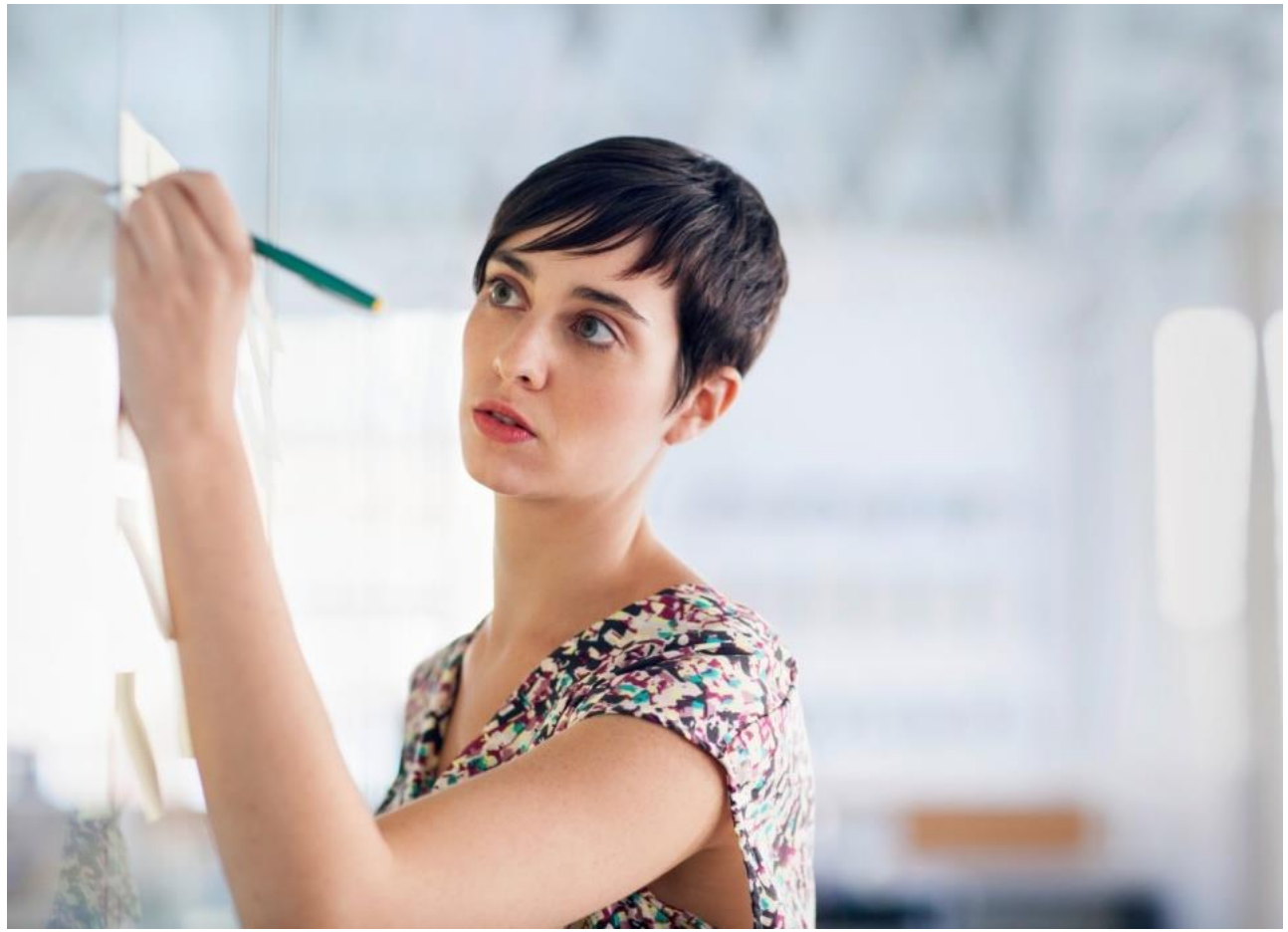
5年以上 53回

2011年11月～2017年8月17日時点

Oracle Technology Networkで、ほぼ毎月連載中
「しばちょう先生の試して納得！DBAへの道」
http://www.oracle.com/technetwork/jp/database/articles/shibacho/index.html

Twitter Account: tkssbt

Scenario: 設定シナリオ

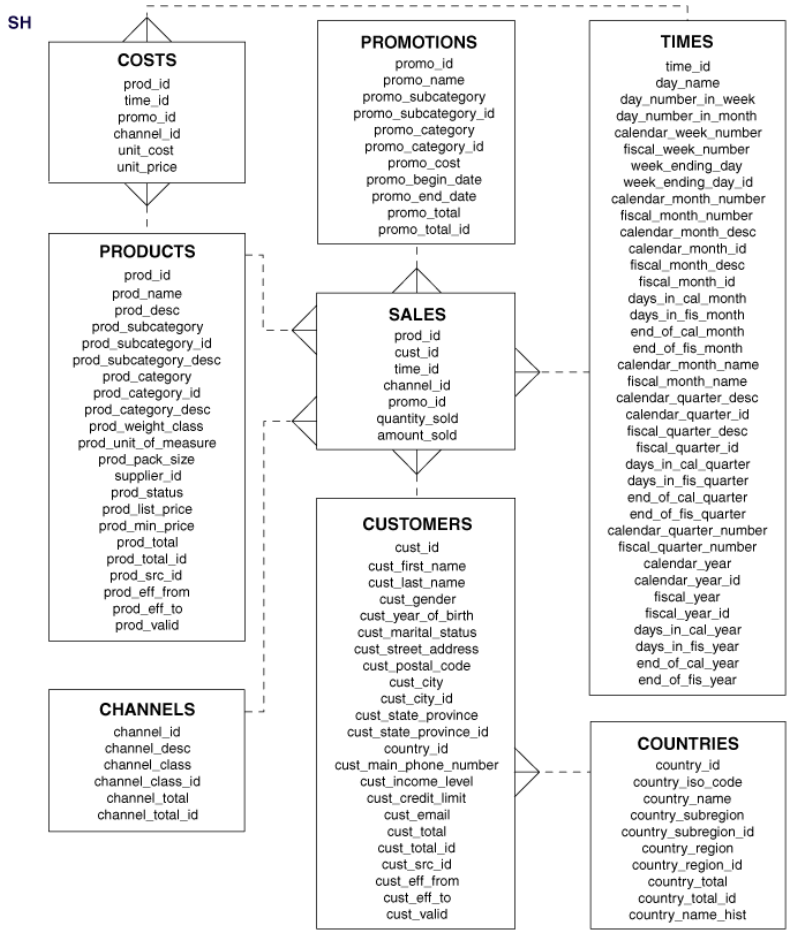


分析ツール等で自動生成されるSQL文は複雑であるために、パフォーマンスに問題があってもその構文自体をチューニングすることは非常に難しい傾向があります。

本セッションでは、最新Oracle Database 12cの機能を有効活用することで、
どこまでSQL処理が高速化していくのか?
をライブで限界にチャレンジします。

注意:
本スライドで掲載されている各種値は、実行環境やワークロード等の状況によって異なります。各製品機能の効果を保証するものではありません。

Oracle Database Sample Schemas – **SH** Schema



- サンプル・スキーマのSHがベース
- SALES表のデータ量を増幅
  - CHAR（105）のダミー列を2つ追加
    - 英数字105文字で、値の種類は30パターン

```
rpad(mod(CUST_ID,30),105,'dummy1')
rpad(mod(CUST_ID,30),105,'dummy2')
```

  - 以下のINSERT文を繰り返し実行し、約32GBへ増幅した環境

```
SQL> insert /*+append */
       into SALES nologging
       select * from SALES ;
```

http://docs.oracle.com/cd/E82638_01/COMSC/schema-diagrams.htm#COMSC00016

チューニング対象のSQL文

CPUバウンド

```
WITH /*+MONITOR */
DUMMY_SALES AS
( select * from (select 0 from CHANNELS ) D1, sales D2),
SACOMMON1340 AS
(  select sum(T220.AMOUNT_SOLD) as c1, sum(T220.QUANTITY_SOLD) as c2,
        T147.CHANNEL_CLASS as c3, T228.CALENDAR_QUARTER_DESC as c4,
        T228.CALENDAR_YEAR as c5, T185.PROD_CATEGORY as c6
    from CHANNELS T147, PRODUCTS T185,
        DUMMY_SALES T220, TIMES T228
   where ( T220.TIME_ID < to_date('2014/01/01','YYYY/MM/DD')
     and T228.TIME_ID = T220.TIME_ID
     and T147.CHANNEL_ID = T220.CHANNEL_ID
     and T185.PROD_ID = T220.PROD_ID)
   group by T147.CHANNEL_CLASS,
          T185.PROD_CATEGORY,
          T228.CALENDAR_QUARTER_DESC,
          T228.CALENDAR_YEAR),
SAWITH0 AS
(  select distinct 0 as c1, D1.c3 as c2, D1.c4 as c3, D1.c5 as c4,
             D1.c6 as c5, D1.c2 as c6, D1.c1 as c7, cast(NULL as DOUBLE PRECISION ) as c8
  from SACOMMON1340 D1),
SAWITH1 AS
(  select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4,
        D1.c5 as c5, D1.c6 as c6, D1.c7 as c7, D1.c8 as c8, sum(D1.c7) as c9
  from SAWITH0 D1
  group by D1.c1, D1.c2, D1.c3, D1.c4, D1.c5, D1.c6, D1.c7, D1.c8),
SAWITH2 AS
(  select distinct 1 as c1, D1.c3 as c2, D1.c4 as c3, D1.c5 as c4,
        D1.c6 as c5, D1.c2 as c6, D1.c1 as c7
  from SACOMMON1340 D1),
SAWITH3 AS
(  select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4,
        D1.c5 as c5, D1.c6 as c6, D1.c7 as c7, sum(D1.c6) as c8, sum(D1.c7) as c9
  from SAWITH2 D1
  group by D1.c1, D1.c2, D1.c3, D1.c4, D1.c5, D1.c6, D1.c7),
SAWITH4 AS
((    select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5,
          D1.c6 as c6, D1.c7 as c7, D1.c8 as c8,
          sum(D1.c9) over (partition by D1.c3, D1.c4, D1.c5) as c9
      from SAWITH1 D1
    union all
    select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5,
          D1.c6 as c6, D1.c7 as c7,
          sum(D1.c8) over (partition by D1.c3, D1.c4, D1.c5) as c8,
          sum(D1.c9) over (partition by D1.c3, D1.c4, D1.c5) as c9
     from SAWITH3 D1   ))
select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5,
       D1.c6 as c6, D1.c7 as c7, D1.c8 as c8, D1.c9 as c9
from SAWITH4 D1 order by c1, c3, c5, c4;
```

I/Oバウンド

```
WITH /*+MONITOR */
SACOMMON1340 AS
(  select sum(T220.AMOUNT_SOLD) as c1, sum(T220.QUANTITY_SOLD) as c2,
        T147.CHANNEL_CLASS as c3, T228.CALENDAR_QUARTER_DESC as c4,
        T228.CALENDAR_YEAR as c5, T185.PROD_CATEGORY as c6
    from CHANNELS T147, PRODUCTS T185,
        SALES T220, TIMES T228
   where ( T220.TIME_ID < to_date('2014/01/01','YYYY/MM/DD')
     and T228.TIME_ID = T220.TIME_ID
     and T147.CHANNEL_ID = T220.CHANNEL_ID
     and T185.PROD_ID = T220.PROD_ID)
   group by T147.CHANNEL_CLASS,
          T185.PROD_CATEGORY,
          T228.CALENDAR_QUARTER_DESC,
          T228.CALENDAR_YEAR),
SAWITH0 AS
(  select distinct 0 as c1, D1.c3 as c2, D1.c4 as c3, D1.c5 as c4,
             D1.c6 as c5, D1.c2 as c6, D1.c1 as c7, cast(NULL as DOUBLE PRECISION ) as c8
  from SACOMMON1340 D1),
SAWITH1 AS
(  select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4,
        D1.c5 as c5, D1.c6 as c6, D1.c7 as c7, D1.c8 as c8, sum(D1.c7) as c9
  from SAWITH0 D1
  group by D1.c1, D1.c2, D1.c3, D1.c4, D1.c5, D1.c6, D1.c7, D1.c8),
SAWITH2 AS
(  select distinct 1 as c1, D1.c3 as c2, D1.c4 as c3, D1.c5 as c4,
        D1.c6 as c5, D1.c2 as c6, D1.c1 as c7
  from SACOMMON1340 D1),
SAWITH3 AS
(  select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4,
        D1.c5 as c5, D1.c6 as c6, D1.c7 as c7, sum(D1.c6) as c8, sum(D1.c7) as c9
  from SAWITH2 D1
  group by D1.c1, D1.c2, D1.c3, D1.c4, D1.c5, D1.c6, D1.c7),
SAWITH4 AS
((    select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5,
          D1.c6 as c6, D1.c7 as c7, D1.c8 as c8,
          sum(D1.c9) over (partition by D1.c3, D1.c4, D1.c5) as c9
      from SAWITH1 D1
    union all
    select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5,
          D1.c6 as c6, D1.c7 as c7,
          sum(D1.c8) over (partition by D1.c3, D1.c4, D1.c5) as c8,
          sum(D1.c9) over (partition by D1.c3, D1.c4, D1.c5) as c9
     from SAWITH3 D1   ))
select D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5,
       D1.c6 as c6, D1.c7 as c7, D1.c8 as c8, D1.c9 as c9
from SAWITH4 D1 order by c1, c3, c5, c4;
```

Missions

1. 処理状況を確認せよ！
2. パーティション化で処理量削減を狙え！
3. パラレル化で複数CPUコアを使いこなせ！
4. データ圧縮で更なるI/O量の削減を狙え！
5. インメモリ化で1秒の壁を越えろ！

Mission

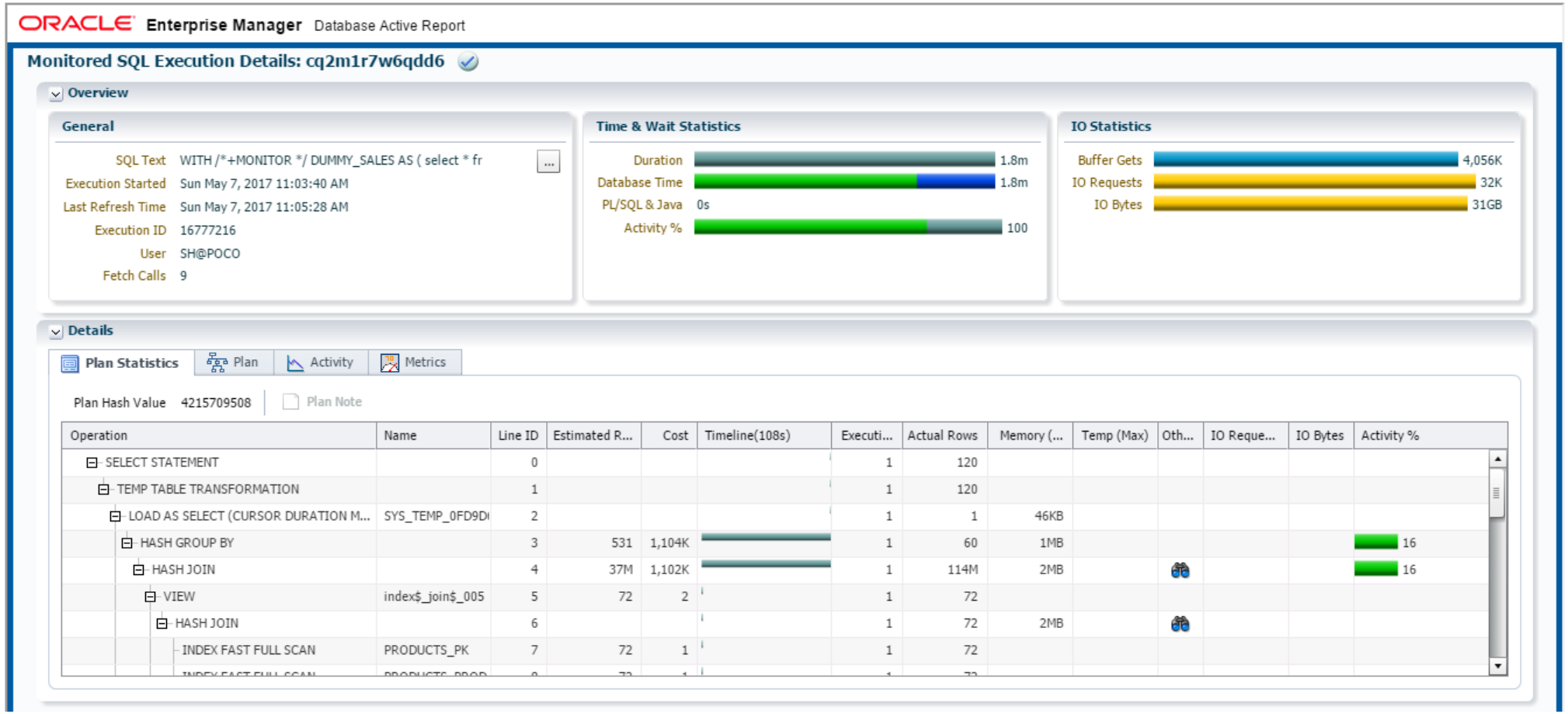
1. 処理状況を確認せよ！
2. パーティション化で処理量削減を狙え！
3. パラレル化で複数CPUコアを使いこなせ！
4. データ圧縮で更なるI/O量の削減を狙え！
5. インメモリ化で1秒の壁を越えろ！

Mission#1
処理状況を確認せよ

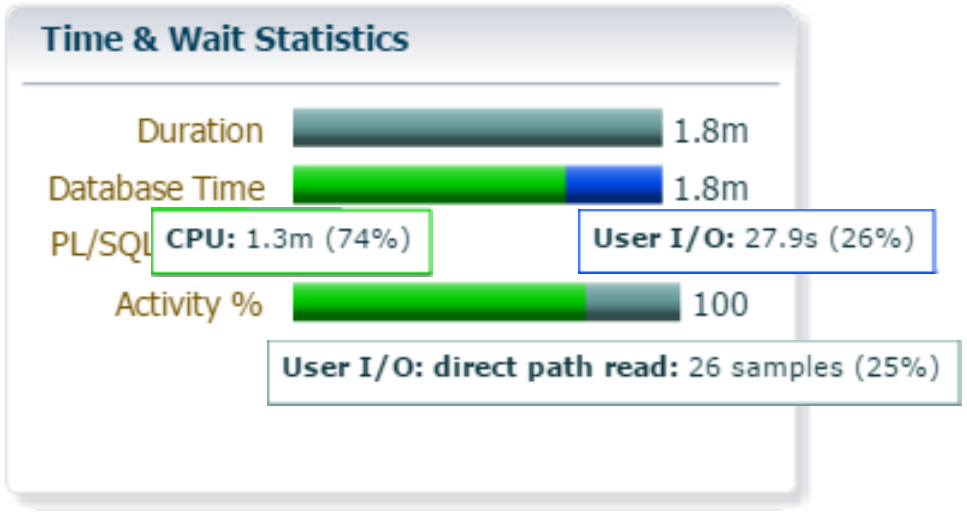
リアルタイムSQL監視　with Oracle Enterprise Manager

CPUバウンドなSQLの処理状況

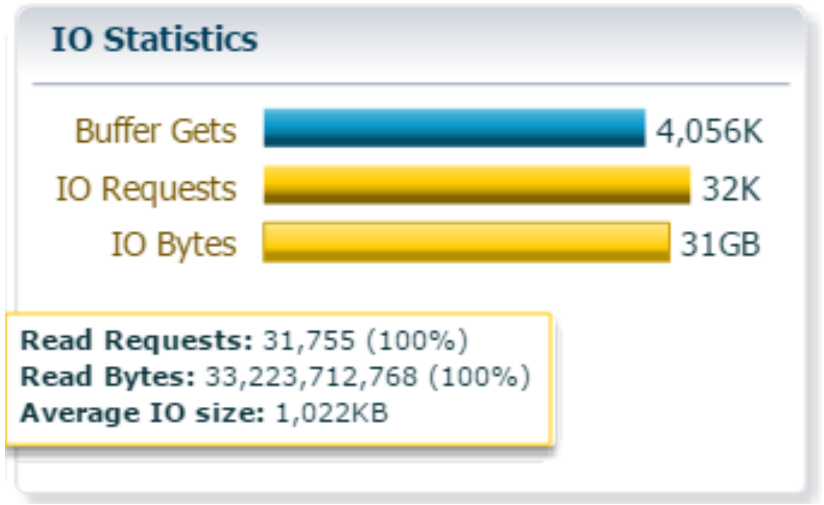
リアルタイムSQL監視アクティブ・レポート



CPUバウンドなSQLの処理状況（詳細）



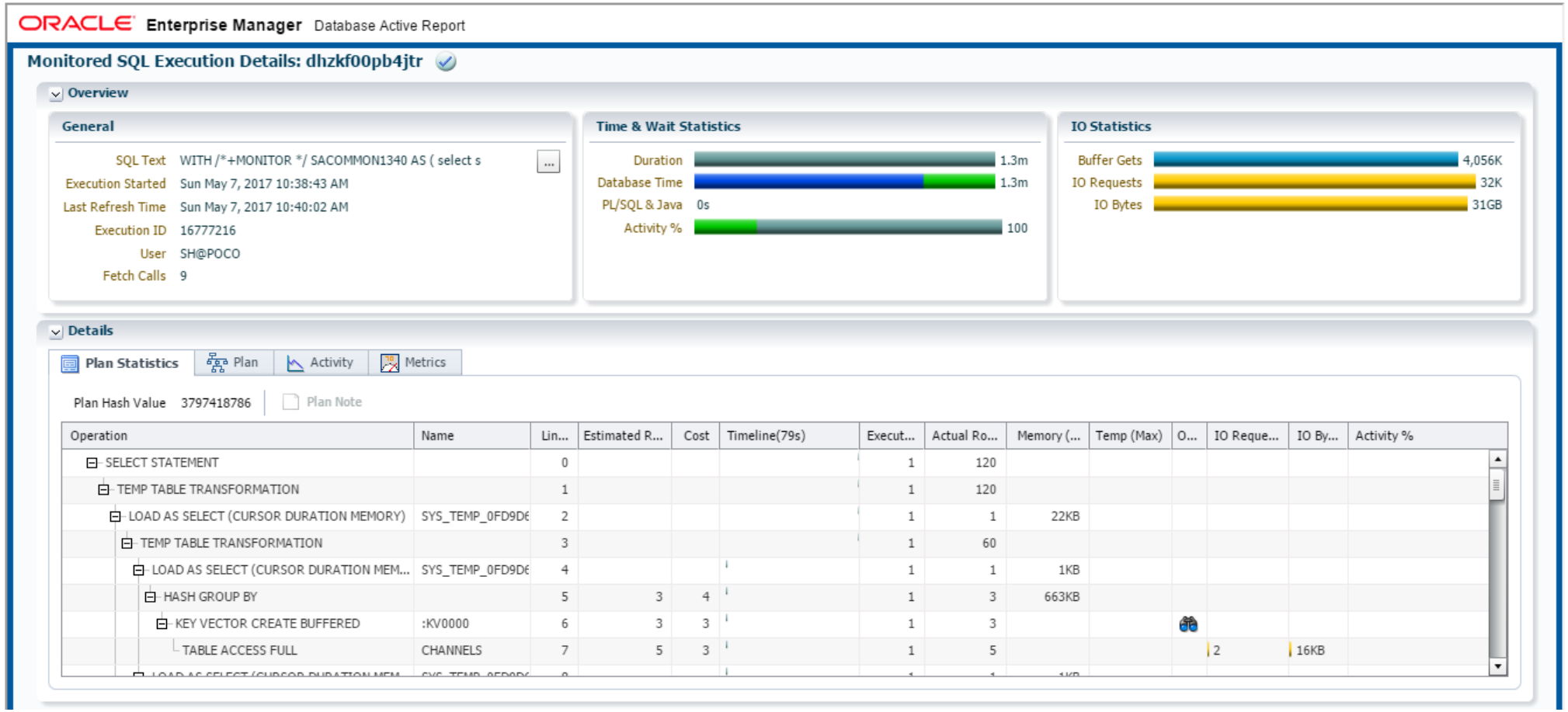
- Duration：SQL実行時間は**1.8m(≒108秒)**
- Database Time：**CPU=74%**, **User IO=26%**
- Activity%: 25%の割合で**direct path read**待機イベントが発生



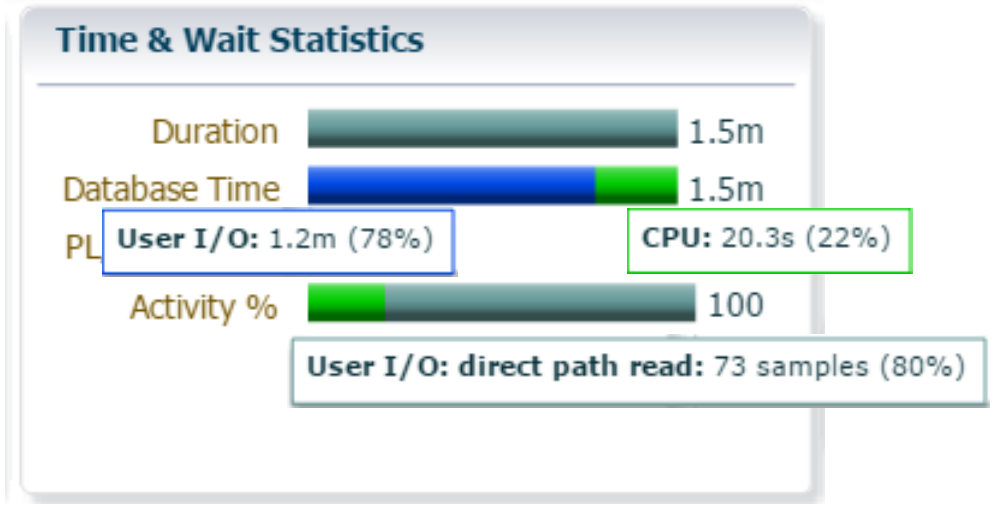
- Buffer Gets: 低いキャッシュ・ヒット率
- IO Requests: 平均I/Oサイズ1MBで32K回
- IO Bytes: **総読込み量は約31GB**

I/Oバウンドな SQLの処理状況

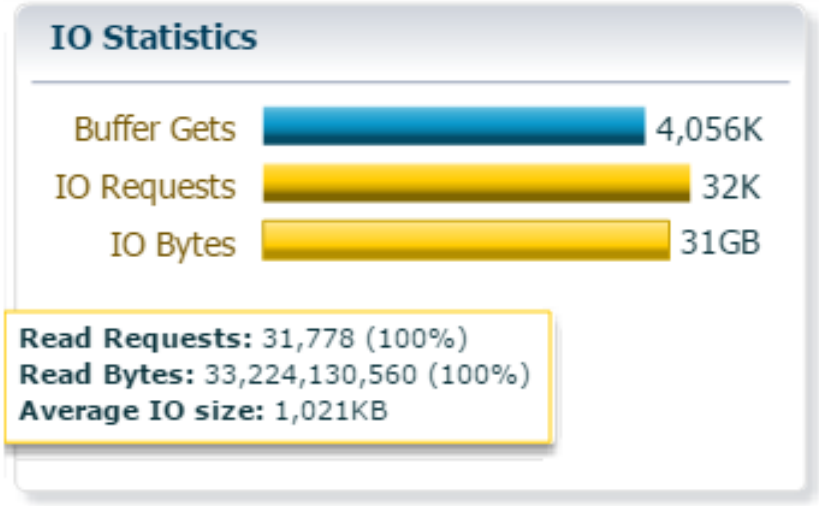
リアルタイムSQL監視アクティブ・レポート



I/Oバウンドな SQLの処理状況（詳細）



- Duration：SQL実行時間は**1.5m(≒90秒)**
- Database Time：**CPU=22%**, **User IO=78%**
- Activity%: **80%の割合**で**direct path read** 待機イベントが発生



- Buffer Gets: 低いキャッシュ・ヒット率
- IO Requests: 平均I/Oサイズ1MBで32K回
- IO Bytes: **総読込み量は約31GB**

（参考）リアルタイムSQL監視

大量のリソースを消費する長時間実行SQL文のパフォーマンス問題を効果的に特定

- 特徴
  - 経過時間、CPU時間、読取りと書込みの回数、I/O待機時間、その他の各種待機時間などの主要なパフォーマンス指標ごとに、実行計画の各ステップで追跡
- 開始方法
  - 次のどちらかの条件（デフォルト）を満たす場合、自動的にSQL監視を開始
    - SQL文がパラレルで実行される場合
    - 1回の実行で5秒以上のCPUまたはI/O時間を消費している場合
  - 明示的に、対象SQL文に“MONITOR”ヒント句を追記しても監視可能
- SQL監視アクティブ・レポートの例やFAQはコチラ
  - http://www.oracle.com/technetwork/jp/database/sqlmonitor-101860-ja.html

Mission

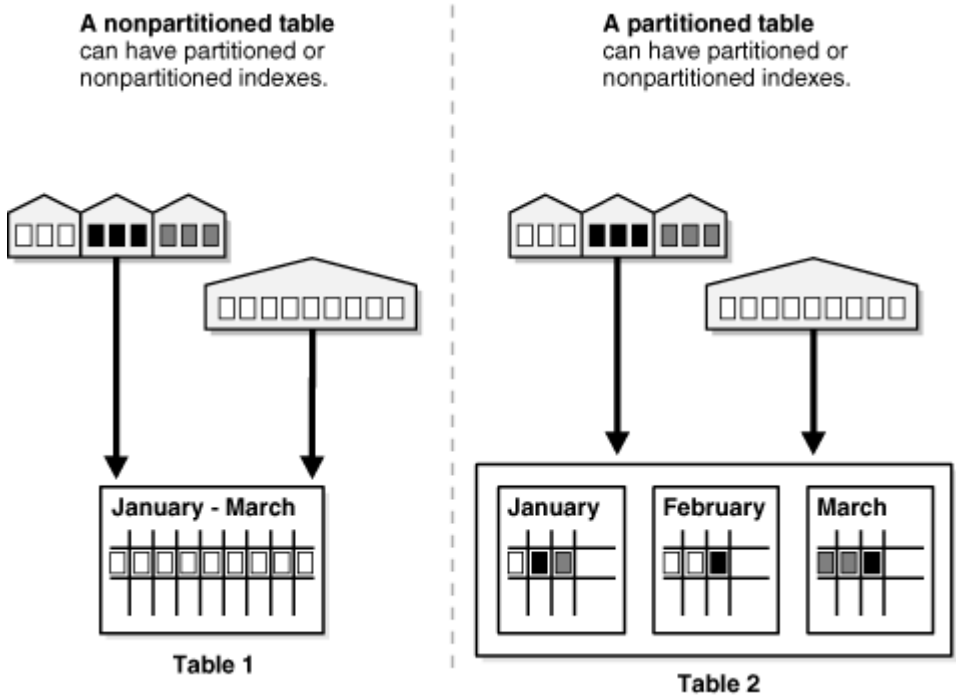
1. 処理状況を確認せよ！
2. パーティション化で処理量削減を狙え！
3. パラレル化で複数CPUコアを使いこなせ！
4. データ圧縮で更なるI/O量の削減を狙え！
5. インメモリ化で1秒の壁を越えろ！

Mission#2
パーティション化で処理量削減を狙え！

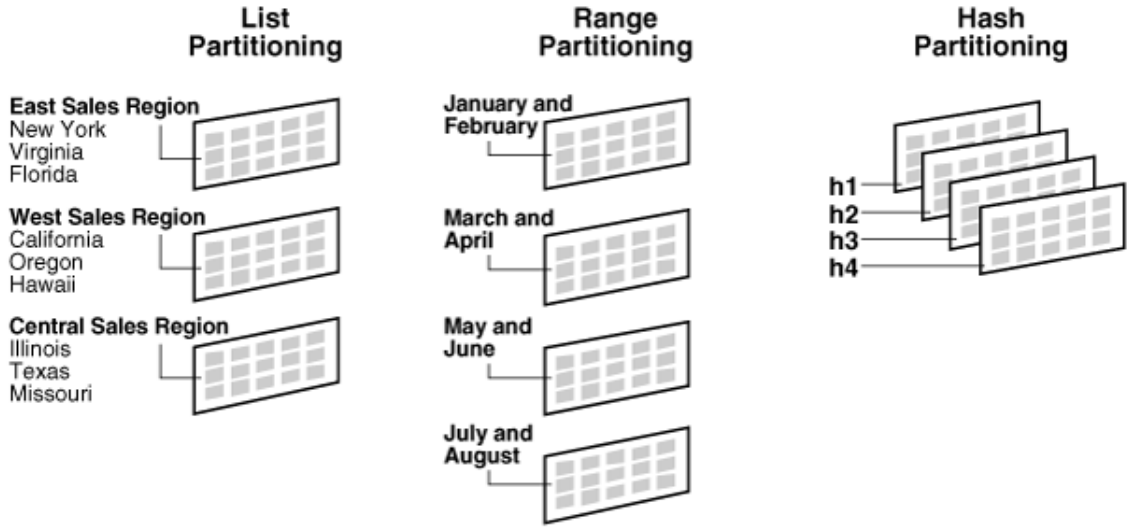
Oracle Partitioning Option
パーティション・アドバイザ（SQLアクセス・アドバイザ）
オンラインでパーティション表への変換（12.2~）

パーティションって何？

表を内部的に分割して管理することで、
パフォーマンス、管理性、可用性が向上する



3つの基本的なデータ配分方法を提供し、
アプリケーション（SQL）はそれらを意識しない



Oracle® Database VLDBおよびパーティショニング・ガイド　12c リリース2 (12.2)

パーティション・プルーニング

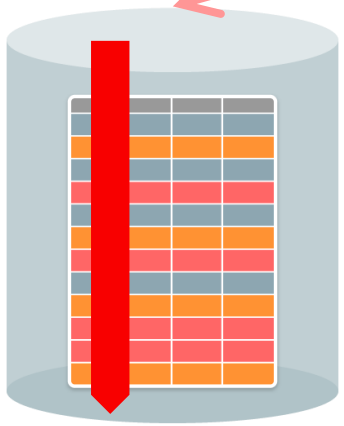
読込み対象データを限定することで、処理量（CPU時間、I/O量）を削減可能

```
SQL> select * from TABLE1
 where COLOR = 'RED' ;
```

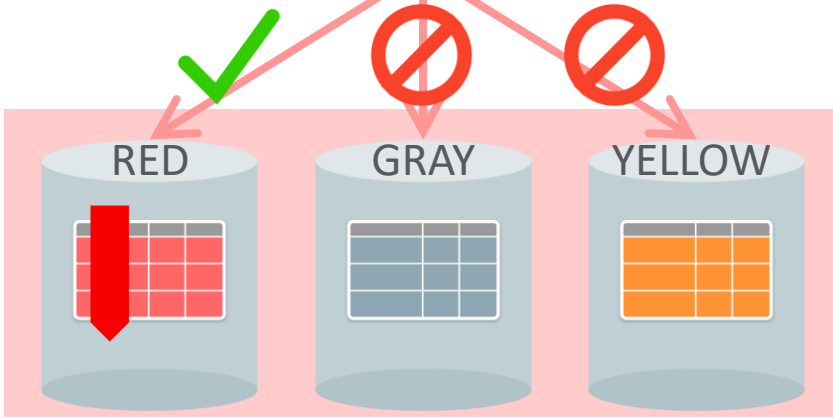
Oracle Client

必要なデータ（RED）のみを読み取り、最低限のフィルタリングを実現

全データを読み取り、DBサーバーのCPUを使ってフィルタリング



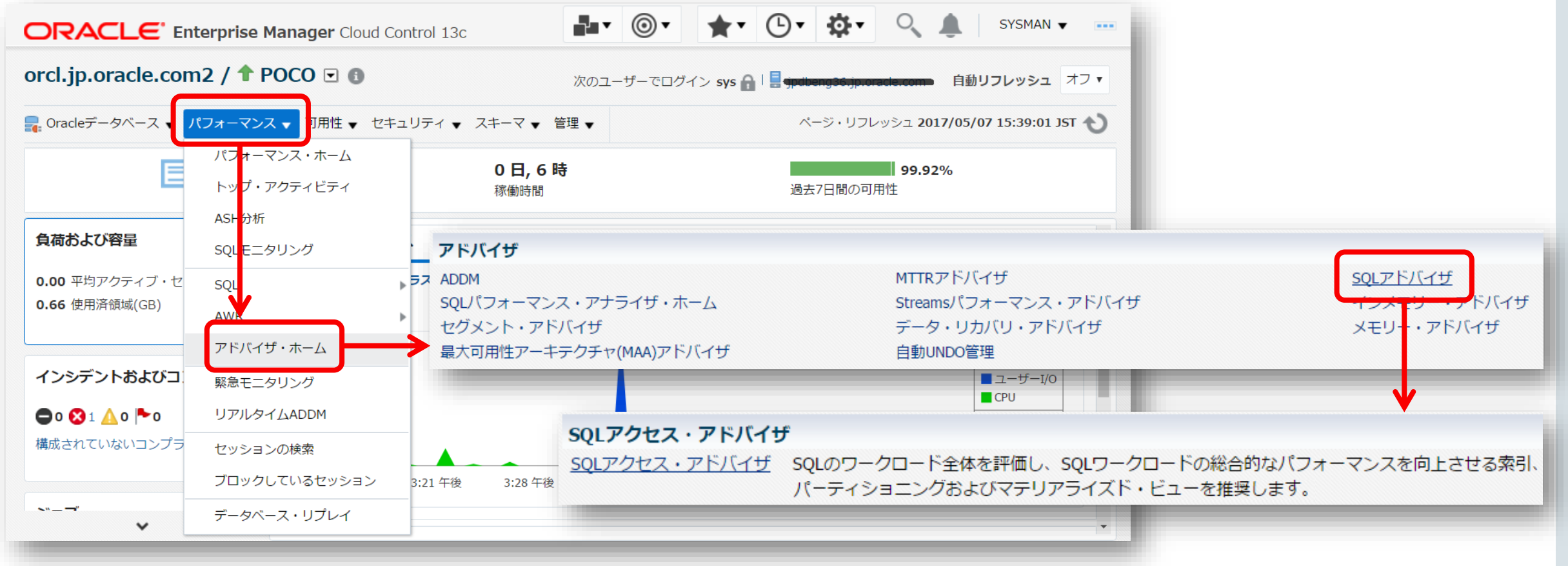
非パーティション表



パーティション表

パーティション・アドバイザ from Enterprise Manager

ワークロード(実行されるSQL文)を解析し、適切なパーティション構成を推奨する機能



パーティション・アドバイザ from Enterprise Manager

SALES表のレンジ・パーティション化を推奨

オンラインでパーティション表への変換

Oracle Database 12c Release 2 ~

- 非パーティション表をパーティション表へ変換する”modify句”を提供
  さらに、”online”キーワードを指定すると、変換中でもDML操作が可能
  - 以下は、今回のシナリオをSALES表に対して、TIME_ID列をパーティション・キーとした時間隔（一か月単位）レンジ・パーティションへとオンラインで変換する例

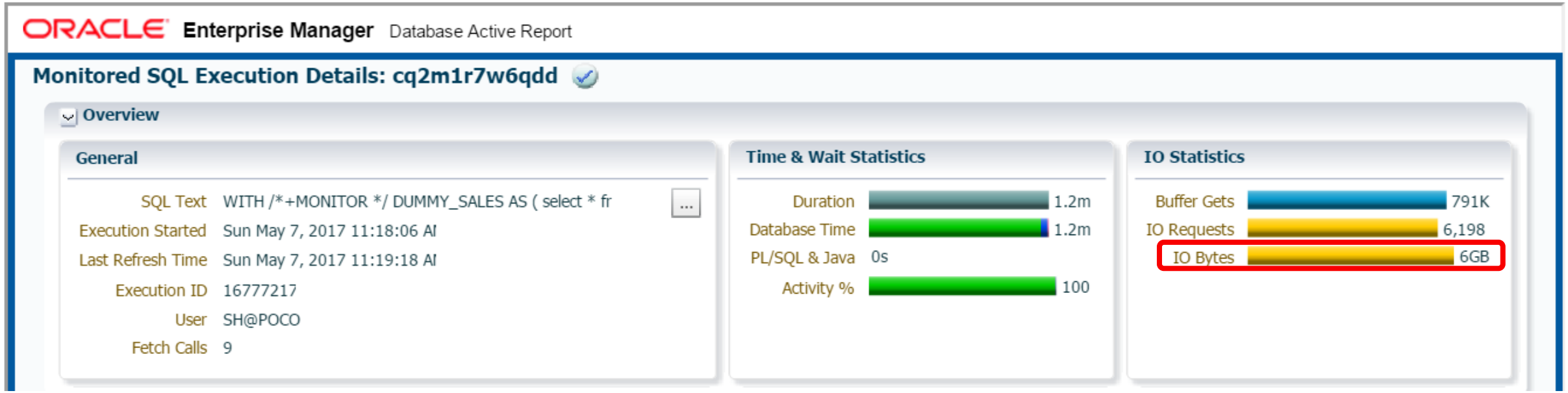
```
SQL> alter table SALES
          modify partition by range(TIME_ID)
               interval(numtoyminterval(1, 'MONTH'))
           (partition values less than
               (to_date('2013-01-01 00:00:00',
                    'YYYY-MM-DD HH24:MI:SS'))) online
          update indexes (…,…) ;
```

✔

CPUバウンドなSQL

パーティション化の効果

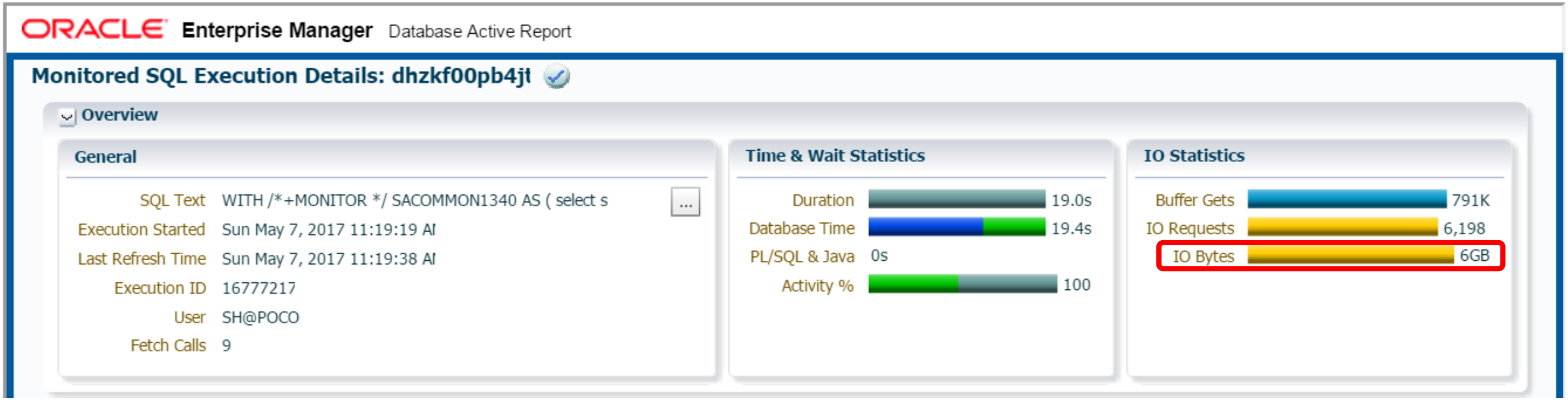
- SQL実行時間（Duration）: 1.8m(108秒) → 1.2m(72秒)へ改善
  - IO Bytes: 31GB → 6GB へ大幅に削減し、データ読み込み時間が短縮
  - 元々CPU時間が占める割合が高い為、パーティション化の効果が高くはない



I/Oバウンドな SQL

パーティション化の効果

- SQL実行時間（Duration）: 1.5m(90秒) → 19.0s(19秒)へ改善
  - IO Bytes: 31GB → 6GB へ大幅に削減し、データ読み込み時間が短縮
  - 元々User I/O時間が占める割合が高い為、パーティション化の効果が高い



対象SQLで期待されるパーティション・プルーニングの効果

答え合わせ

- 今回のSALES表は、2013年～2016年の4年間分のデータを保持
- しかし、対象の2つのSQL文では、2013年の1年間分のみが集計対象

```
WITH /*+MONITOR */
SACOMMON1340 AS
(  select sum(T220.AMOUNT_SOLD) as c1,
             ……
    from CHANNELS T147,
         SALES T220,
         ……
   where ( T220.TIME_ID < to_date('2014/01/01','YYYY/MM/DD')
     and T147.CHANNEL_ID = T220.CHANNEL_ID
     and T185.PROD_ID = T220.PROD_ID)
   group by T147.CHANNEL_CLASS, ……
```

Live Challenge!!

SQLパフォーマンスの高速化の限界を目指せ！

	Normal	+ Partitioning
CPUバウンドなSQL 実行時間（秒）	108	72
相対比	x1	x1.5
I/Oバウンドなsql 実行時間（秒）	90	19
相対比	x1	x4.7

Mission

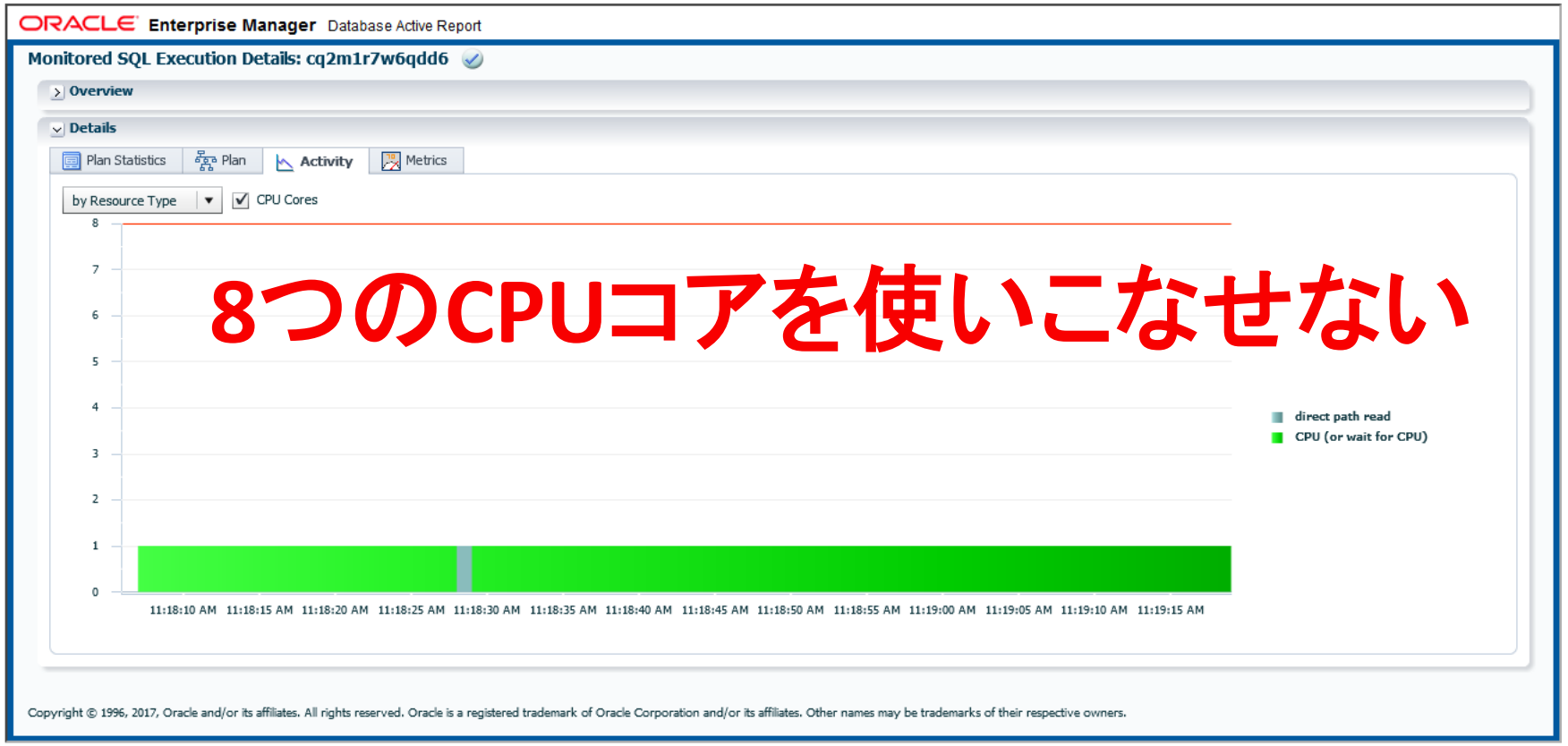
1. 処理状況を確認せよ！
2. パーティション化で処理量削減を狙え！
3. **パラレル化で複数CPUコアを使いこなせ！**
4. データ圧縮で更なるI/O量の削減を狙え！
5. インメモリ化で1秒の壁を越えろ！

Mission#3
パラレル化で複数CPUコアを使いこなせ！

SQLのパラレル実行
自動パラレル度設定（PARALLEL_DEGREE_POLICY初期化パラメータ）

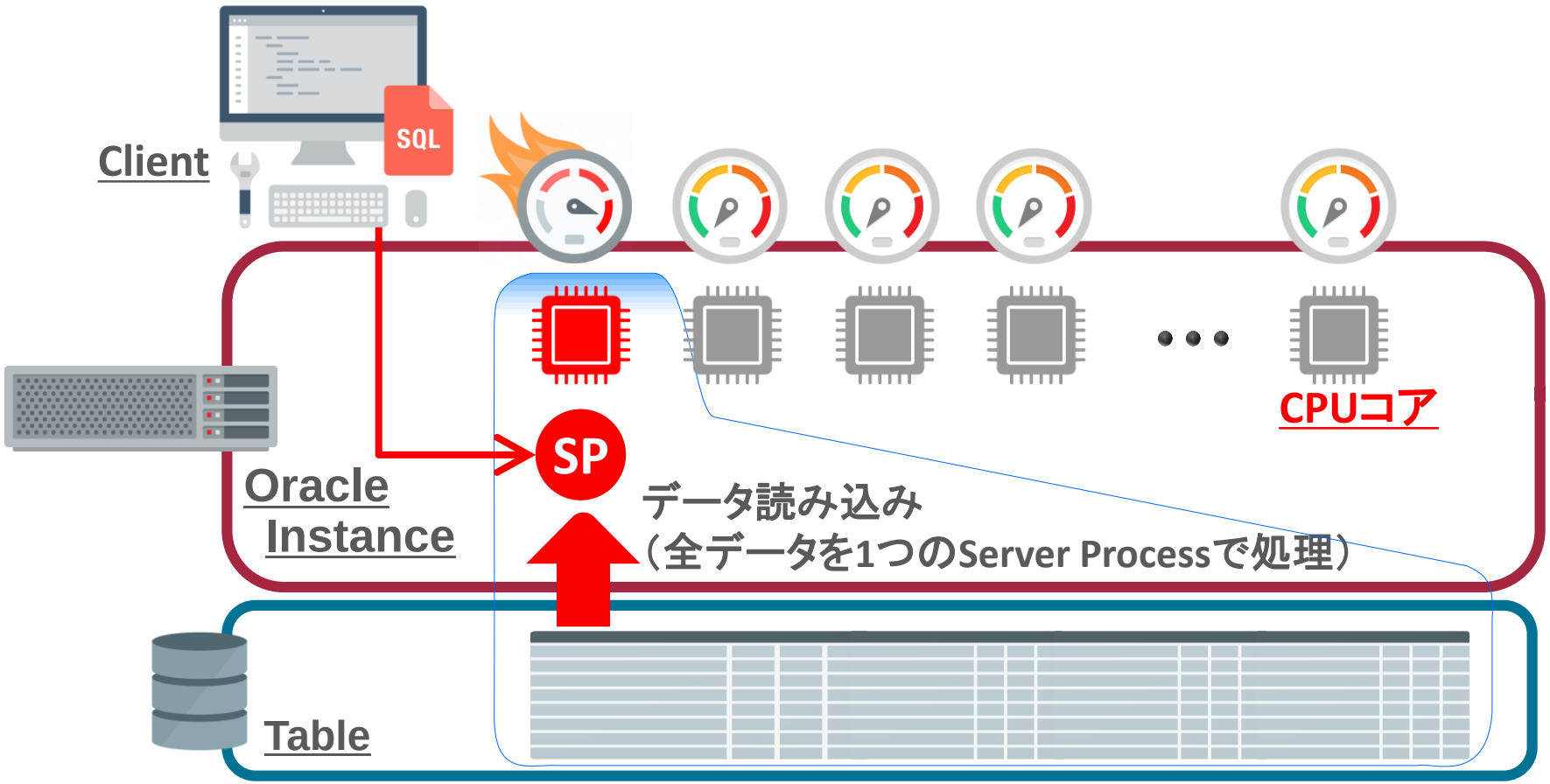
CPUバウンドなSQLのActivity（パーティション化後）

1CPUコアがボトルネックな状況



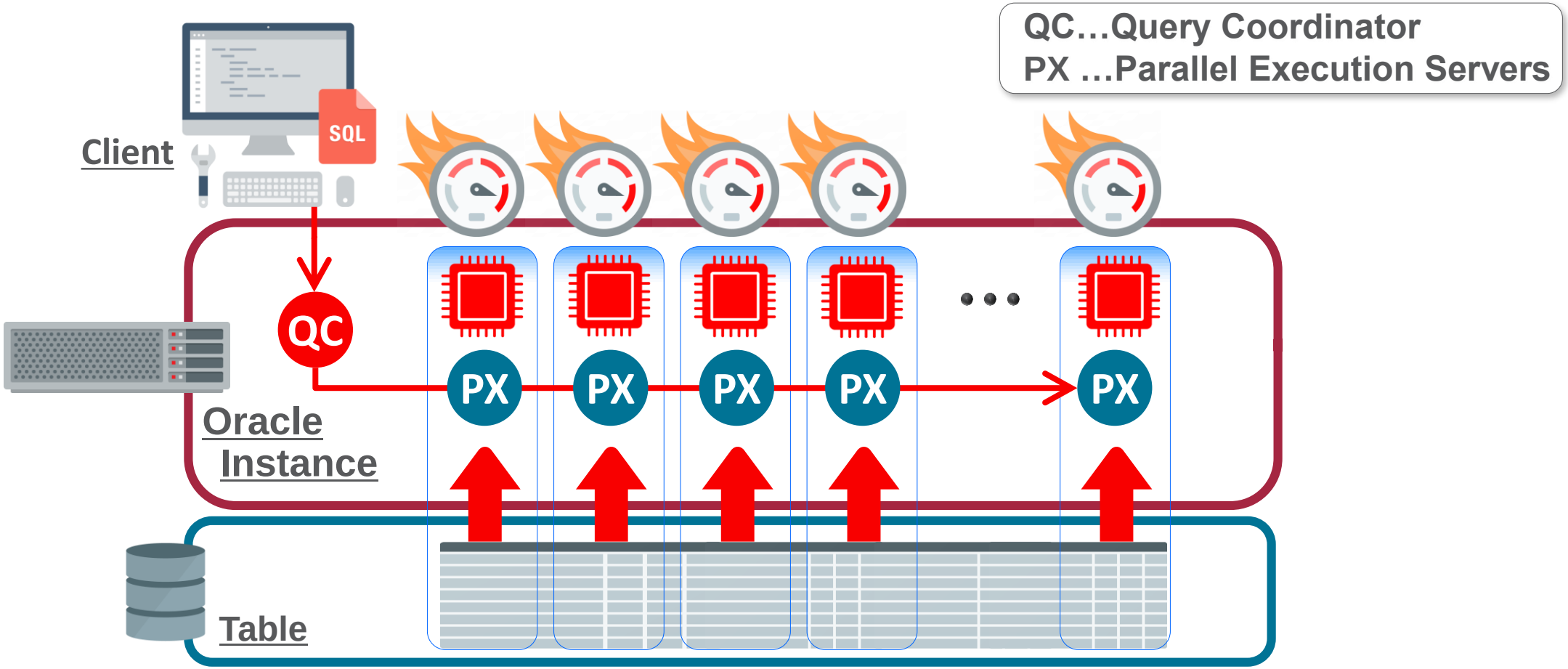
1つのSQLの実行では、1つのCPUコアしか活用できない

通常のシリアル実行の場合



パラレル実行で、マルチコアの有効活用

1つのSQLを複数のプロセスが自動的に分割実行



[White Paper] Oracle Database 12cでのパラレル実行の基本

主なパラレル実行の方法

手動と自動パラレル度設定

- 手動での強制パラレル度設定
  - 対象のSELECT文を実行する前に以下のコマンドを実行（*n*は数字を指定）

```
SQL> alter session force parallel query parallel n ;
```

- 自動パラレル度設定（自動DOP: Automatic Degree ）
  - 初期化パラメータPARALLEL_DEGREE_POLICYで有効化を制御
    - デフォルト（MANUAL）以外の設定値（LIMITED, AUTO, ADAPTIVE）へ変更
      - システム・レベルまたは、セッション・レベルで適用可能
  - 詳細は、[White Paper] Oracle Database 12cでのパラレル実行の基本
    - http://www.oracle.com/technetwork/jp/content/parallel-execution-132539-ja.pdf

その他のパラメータ

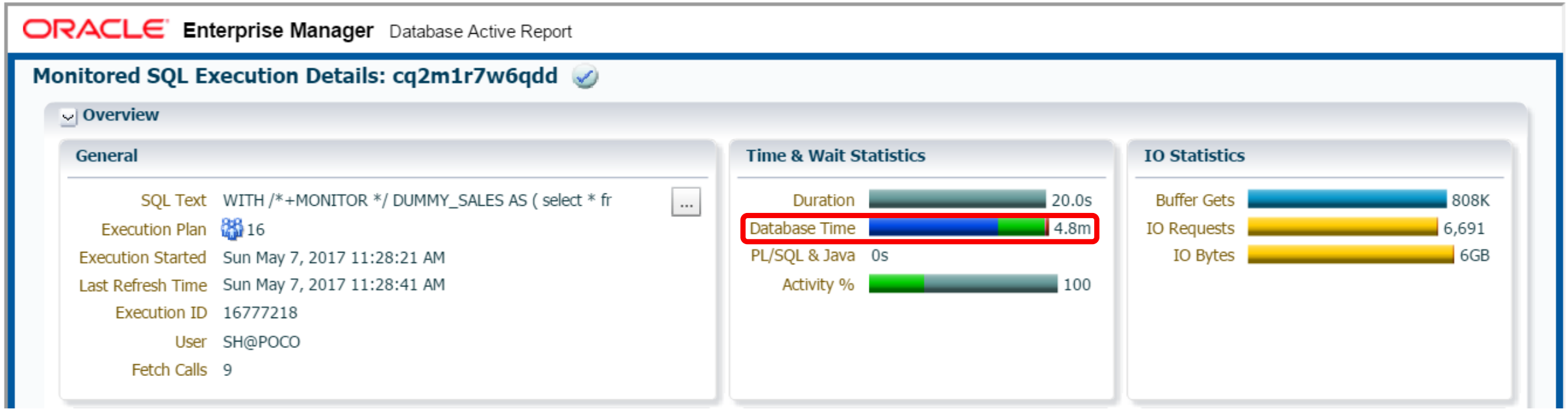
[White Paper] Oracle Database 12cでのパラレル実行の基本　からの抜粋

パラメータ名	デフォルト値	推奨される値	説明
PARALLEL_MIN_SERVERS	CPU_COUNT * PARALLEL_THREADS_PER_CPU * 2	デフォルト。連続使用される PX サーバーの最小数まで増やすことを検討してください。	データベース・インスタンスによって常に割り当てられているパラレル実行サーバーの最小数を定義します。
PARALLEL_MAX_SERVERS	PARALLEL_THREADS_PER_CPU * CPU_COUNT * *concurrent_parallel_users* * 5	デフォルト	データベース・インスタンスで割当て可能なパラレル実行サーバーの最大数を定義します。これはハード・リミットであり、これを超過することはできません。
PARALLEL_ADAPTIVE_MULTI_USER	TRUE	FALSE	同時ワークロードに基づいて文の DOP を引き下げます。応答時間を予測できなくなる場合があります。
PARALLEL_FORCE_LOCAL	FALSE	デフォルト	Oracle RAC 環境において、文が発行されたノードにパラレル・サーバー・プロセスを限定するかどうかを
PARALLEL_MIN_PERCENT	0	デフォルト	パラレル実行に必要なパラレル実行プロセスとして要求する数の最小割合です。
PARALLEL_MIN_TIME_THRESHOLD	AUTO	デフォルト	自動 DOP が起動されるまでの文の最小実行時間。デフォルトは 10 秒です。
PARALLEL_THREADS_PER_CPU	2	ハイパースレッドを有効にしたプラットフォームの場合は 1、それ以外のプラットフォームの場合は	パラレル実行中に CPU で処理できるパラレル・プロセス数です。
PARALLEL_EXECUTION_MESSAGE_SIZE	16KB	デフォルト。	パラレル・サーバー・プロセス同士およびパラレル・サーバー・プロセスと QC との通信に使用されるバッ

CPUバウンドなSQL

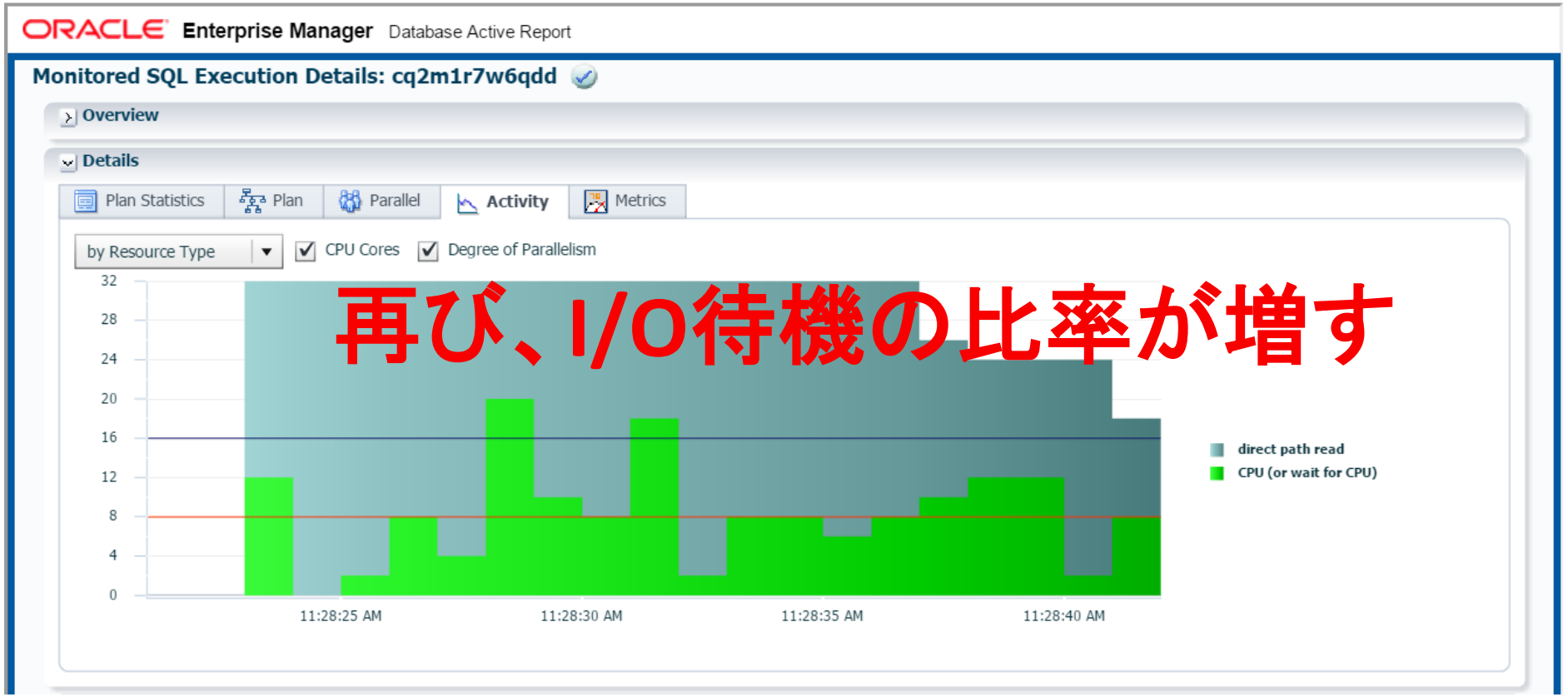
パラレル実行の効果

- SQL実行時間（Duration）: 1.2m(72秒) → 20.0s(20秒)へ改善
  - 複数プロセスが同時にCPUを使用した為、Database TimeがDurationよりも大きい
  - 元々CPU時間が占める割合が高い為、パラレル実行の効果が高い



CPUバウンドなSQLのActivity（パラレル実行の効果）

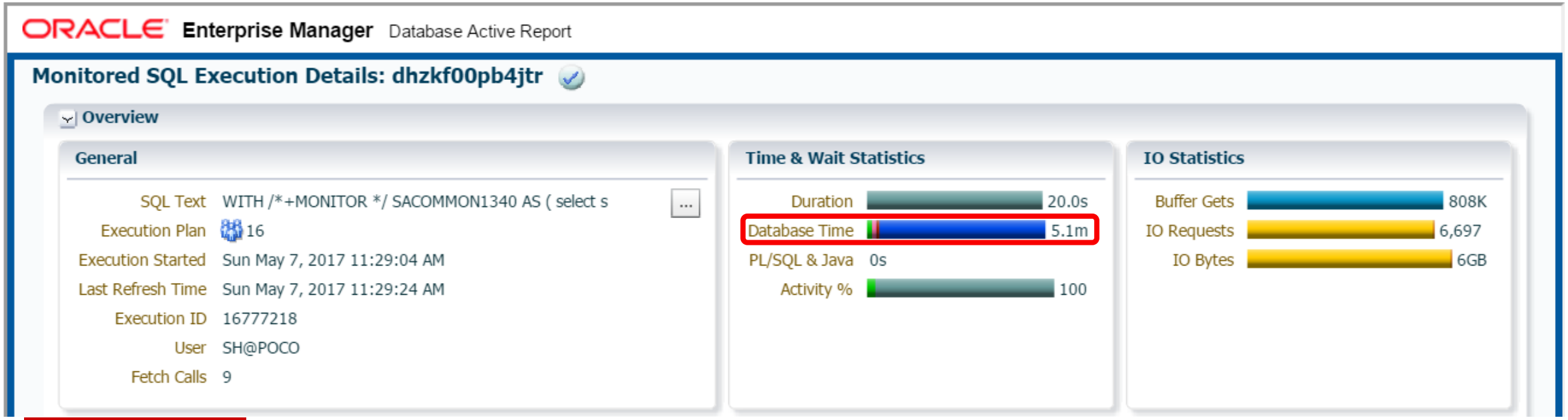
1CPUコアがボトルネック → 複数CPUコアを活用



I/Oバウンドな SQL

パラレル実行の効果

- SQL実行時間（Duration）: 19.0s(19秒) → 20.0s(20秒)で変化なし
  - 元々User I/O時間が占める割合が非常に高い為、パラレル実行の効果が無い
    - 複数プロセスが起動しているが、ディスク読み取りで待機しているだけ（**青帯の比率が高まる**）



Live Challenge!!

SQLパフォーマンスの高速化の限界を目指せ！

	Normal	+ Partitioning	+ Parallel Query
CPUバウンドなSQL 実行時間（秒）	108	72	20
相対比	x1	x1.5	x5.4
I/Oバウンドなsql 実行時間（秒）	90	19	---
相対比	x1	x4.7	---

Mission

1. 処理状況を確認せよ！
2. パーティション化で処理量削減を狙え！
3. パラレル化で複数CPUコアを使いこなせ！
4. データ圧縮で更なるI/O量の削減を狙え！
5. インメモリ化で1秒の壁を越えろ！

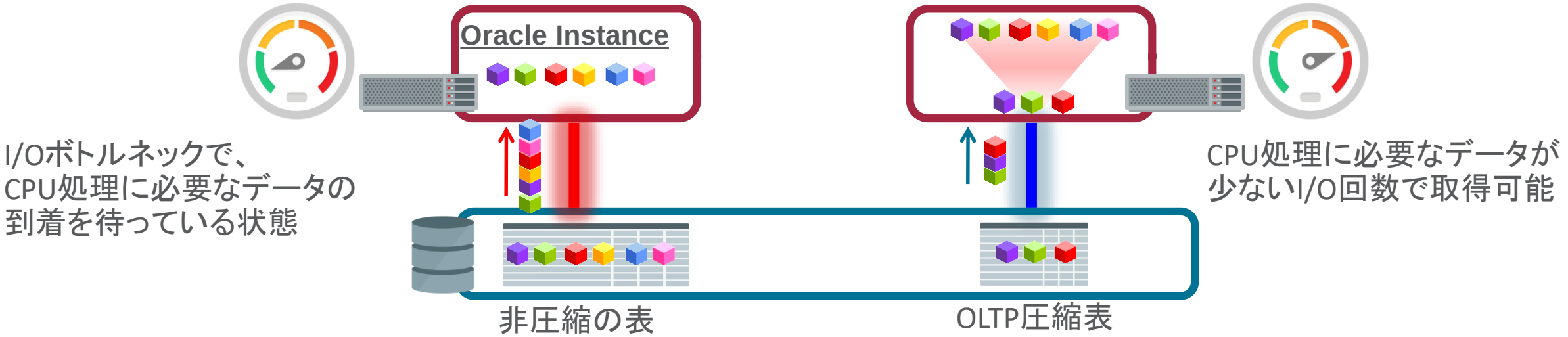
Mission#4
データ圧縮で更なるI/O量の削減を狙え！

Oracle Advanced Compression(表データの圧縮機能)
オンラインで特定パーティションを圧縮変換

表圧縮の主なメリット

Oracle Databaseでは、いくつかの表圧縮の方法をサポート

- ディスク領域の節約するだけではなく、パフォーマンス観点でもメリット有り
  - 一つのデータ・ブロック内に格納されるレコード数が増加する為
    - 総ディスクI/O回数（I/O待機時間）の削減（1回のI/Oで取得できるレコード数が増える）
    - キャッシュ・ヒット率の向上（圧縮形式でバッファ・キャッシュ上にキャッシュされる）



表圧縮の方法と特徴

Oracle® Database管理者ガイド 12cリリース2 (12.2)　20 表の管理

表 *20-1* 表圧縮方法

表の圧縮方法	圧縮レベル	CPUオーバーヘッド	アプリケーション	注意
基本表圧縮	高	最小	DSS	なし。
高度な行圧縮	高	最小	OLTP、DSS	なし。
ウェアハウス圧縮(ハイブリッド列圧縮)	より高い	より高い	DSS	圧縮レベルとCPUオーバーヘッドは、指定された圧縮レベル(LOWまたはHIGH)に応じて変化します。
アーカイブ圧縮(ハイブリッド列圧縮)	最高	最高	アーカイブ	圧縮レベルとCPUオーバーヘッドは、指定された圧縮レベル(LOWまたはHIGH)に応じて変化します。

表 *20-2* 表の圧縮の特徴

表の圧縮方法	CREATE/ALTER TABLEの構文	ダイレクト・パス挿入または配列挿入	注意
基本表圧縮	ROW STORE COMPRESS [BASIC]	行は基本表圧縮方式で圧縮されます。	ROW STORE COMPRESSとROW STORE COMPRESS BASICは同等です。 ダイレクト・パス・インサートまたは配列挿入を使用せずに挿入された行および更新された行は圧縮されません。
高度な行圧縮	ROW STORE COMPRESS ADVANCED	行は高度な行圧縮方式で圧縮されます。	ダイレクト・パス・インサートまたは配列挿入の使用に関係なく、挿入された行と更新された行は高度な行圧縮を使用して圧縮されます。
ウェアハウス圧縮(ハイブリッド列圧縮)	COLUMN STORE COMPRESS FOR QUERY [LOW|HIGH]	行はウェアハウス圧縮方式で圧縮されます。	この圧縮方式は高いCPUオーバーヘッドが発生する可能性があります。 更新された行およびダイレクト・パス・インサートまたは配列挿入を使用せずに挿入された行は、列形式ではなく行形式で格納されるため、圧縮レベルが低下します。
アーカイブ圧縮(ハイブリッド列圧縮)	COLUMN STORE COMPRESS FOR ARCHIVE [LOW|HIGH]	行はアーカイブ圧縮方式で圧縮されます。	この圧縮方式は高いCPUオーバーヘッドが発生する可能性があります。 更新された行およびダイレクト・パス・インサートまたは配列挿入を使用せずに挿入された行は、列形式ではなく行形式で格納されるため、圧縮レベルが低下します。

圧縮表の作成、特定パーティションの設定変更方法

- 高度な表圧縮が有効な表の作成例
  - 表を作成するCREATE TABLE文に、圧縮属性を指定するだけ
    - CREATE TABLE … ROW STORE COMPRESS ADVANCED;
- 変更方法
  - 今後INSERTされる新規データのみを圧縮（既存データは非圧縮のまま）
    - ALTER TABLE … MODIFY PARTITION … COMPRESS … ;
  - 既存も新規データの両方とも圧縮
    - ALTER TABLE … MOVE PARTITION … COMPRESS …;

※ 変更中にDML処理を受け付け可能なオンラインを選択する場合
- 上記のALTER TABLE … MOVE … COMPRESS …文にONLINEオプションを追加
- 表のオンライン再定義を利用

CPUバウンドなSQL

データ圧縮（高度な行圧縮）の効果

- SQL実行時間（Duration）: 19.0s(19秒) → 9.0s(9秒)へ改善
  - IO Bytes: 6GB → 2GB へ大幅に削減し、データ読込み時間が短縮
  - パラレル実行により、I/O待機時間の割合が高い状態だったため、圧縮の効果有り

I/Oバウンドなsql

データ圧縮（高度な行圧縮）の効果

- SQL実行時間（Duration）: 19.0s(20秒) → 7.0s(7秒)へ改善
  - IO Bytes: 6GB → 2GB へ大幅に削減し、データ読込み時間が短縮
  - パラレル実行により、I/O待機時間の割合が高い状態だったため、圧縮の効果有り

Live Challenge!!

SQLパフォーマンスの高速化の限界を目指せ！

	Normal	+ Partitioning	+ Parallel Query	+ Compression
CPUバウンドなSQL 実行時間（秒）	108	72	20	9
相対比	x1	x1.5	x5.4	x12
I/Oバウンドなsql 実行時間（秒）	90	19	---	7
相対比	x1	x4.7	---	x12

Mission

1. 処理状況を確認せよ！
2. パーティション化で処理量削減を狙え！
3. パラレル化で複数CPUコアを使いこなせ！
4. データ圧縮で更なるI/O量の削減を狙え！
5. インメモリ化で1秒の壁を越えろ！

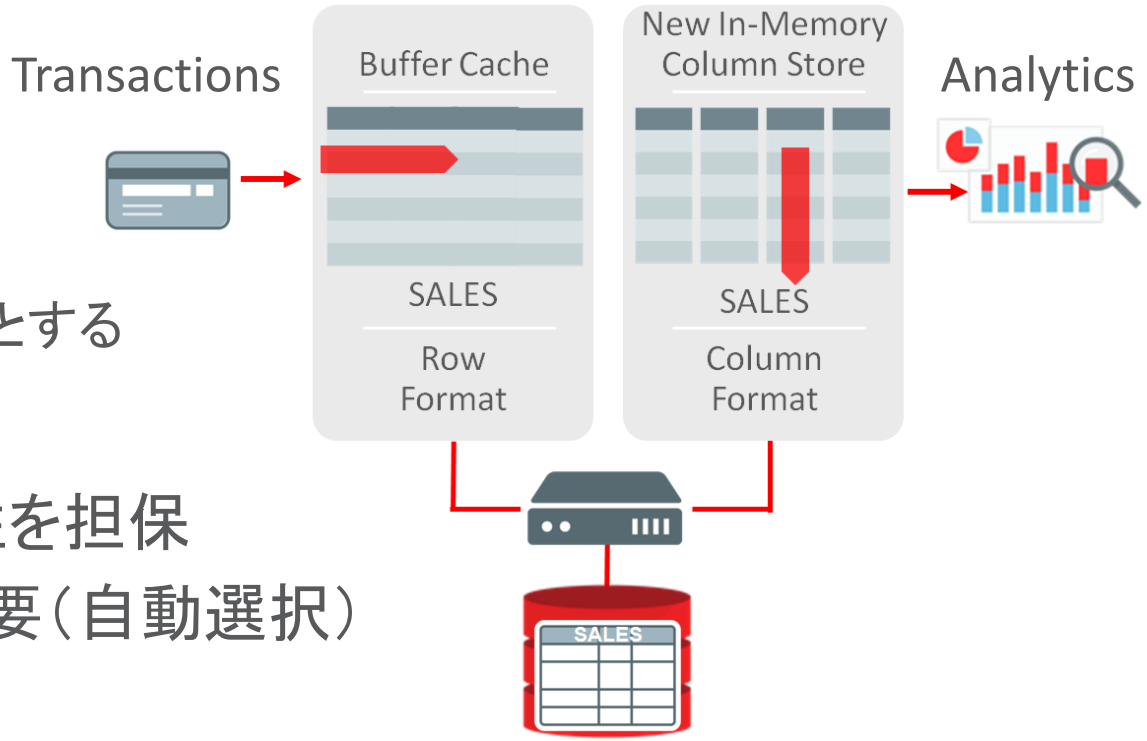
Mission#5
インメモリ化で1秒の壁を越えろ！

Oracle Database In-Memory

Oracle Database In-Memory

Oracle Database 12c Release 1 ~

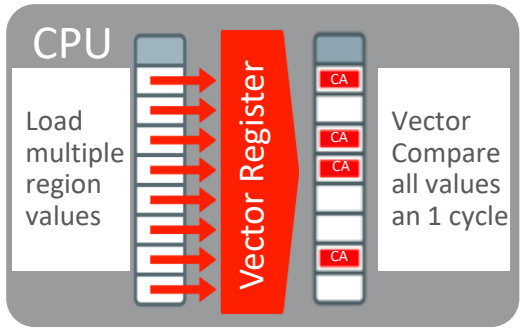
- 一つのデータベースにおいて、2つのフォーマットのデータをメモリ上で保持
  - ロー（行）型フォーマット
    - 注文データの挿入や検索等に代表される、オンライン・トランザクション処理を得意とする
  - カラム（列）型フォーマット
    - 売上合計レポート等の少数の列（カラム）と多数の行（ロー）を高速演算する分析処理を得意とする
  - 同時利用可能で、トランザクションの一貫性を担保
  - SQLに制限なし、アプリケーションの改修不要（自動選択）



Oracle Database In-Memory

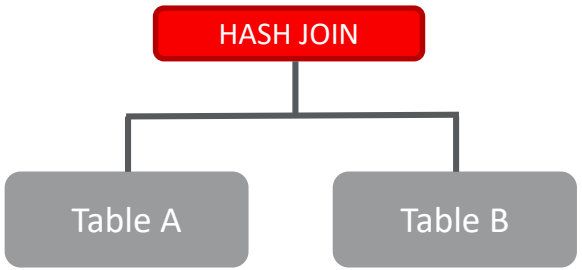
分析クエリーのあらゆる側面を改善

データ・スキャン



- メモリーの速度
- スキャンとフィルタは必要なカラムに限定
- ベクター・インストラクション（CPU命令）

ジョイン



- スター・ジョインを10倍高速なカラム・スキャンに変更
- 大きな表は、小さな表に一致する値にてスキャン

インメモリー集計

- インメモリー・レポート・アウトラインが作成され高速なスキャンと並行で同時に値が集計される
- レポート作成が高速に

Oracle Database In-Memory

設定の基本ステップ

1. In-Memory Column Storeのメモリ容量を設定
   - 初期化パラメータ”inmemory_size” を設定
2. In-Memory Column Store上に保持したい表やパーティションを選択

```
SQL> -- 本セッションでのサンプル
alter table TIMES inmemory priority high;
alter table PRODUCTS inmemory priority high;
alter table CHANNELS inmemory priority high;
alter table SALES modify partition P2013Q1 inmemory priority high;
alter table SALES modify partition P2013Q2 inmemory priority high;
alter table SALES modify partition P2013Q3 inmemory priority high;
alter table SALES modify partition P2013Q4 inmemory priority high;
```

3. （オプション）分析クエリーで使用していた不要索引を削除

✔

補足説明

（オプション）分析クエリーで使用していた不要索引を削除

- オプションの記載の背景
  - 過去、分析クエリーの高速化を目的に索引を作成していた場合、Database In-Memoryを活用することで、その索引は使用しなくなります
  - オンライン・トランザクション処理で、レコードが挿入されるたびに、索引のメンテナンス処理が行われています
  - よって、不要な索引を削除することで、オンライン・トランザクション側の処理を減らすことが可能です

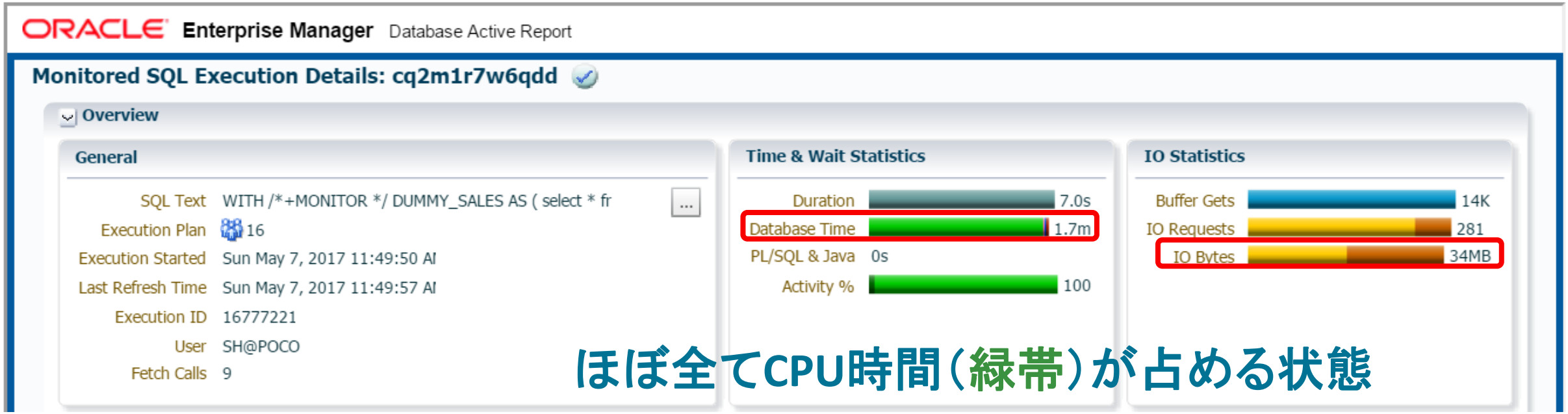
- 索引の使用状況の確認方法
  - （12.2~）DBA_INDEX_USAGE ビューと V$INDEX_USAGE_INFOビュー
  - alter index <INDEX_NAME> monitoring usage ; コマンド
    - 詳細は、「Oracle® Database管理者ガイド 12cリリース2 (12.2) 21.4.7 索引の使用状況の監視」

CPUバウンドなSQL

Oracle Database In-Memoryの効果

＊ SQL*Plusで計測した値

- SQL実行時間（Duration）: 9.0s(9秒) → 7.0s(7.67秒*)へ改善
  - IO Bytes: 2GB → 34MB へ大幅に削減し、データ読込み時間がほぼゼロへ
  - 既にCPU時間（緑帯）の割合が高かった為、改善幅は大きくはない

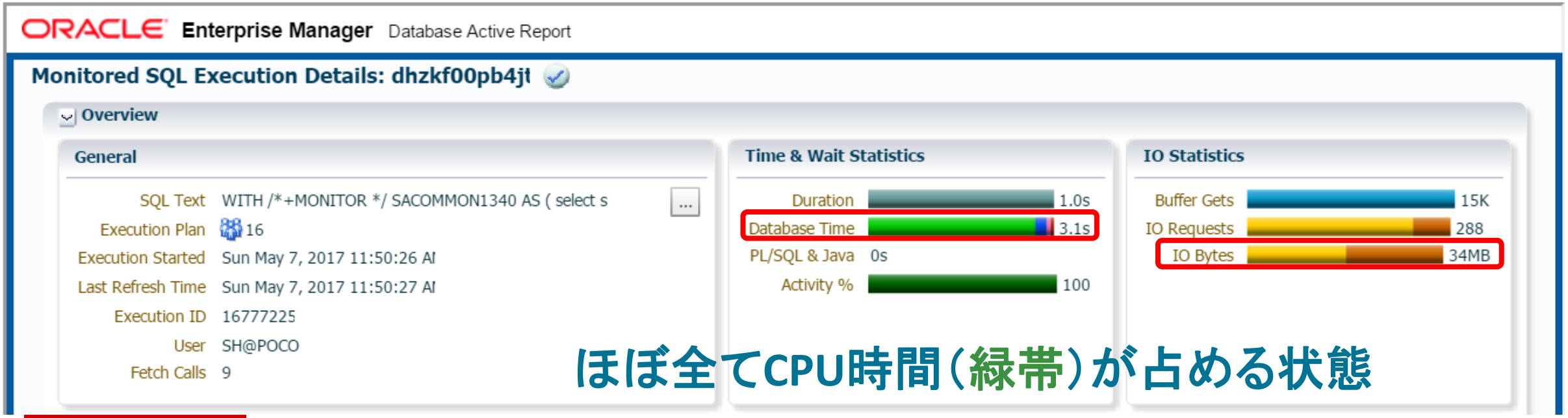


I/OバウンドなSQL

Oracle Database In-Memoryの効果

＊ SQL*Plusで計測した値

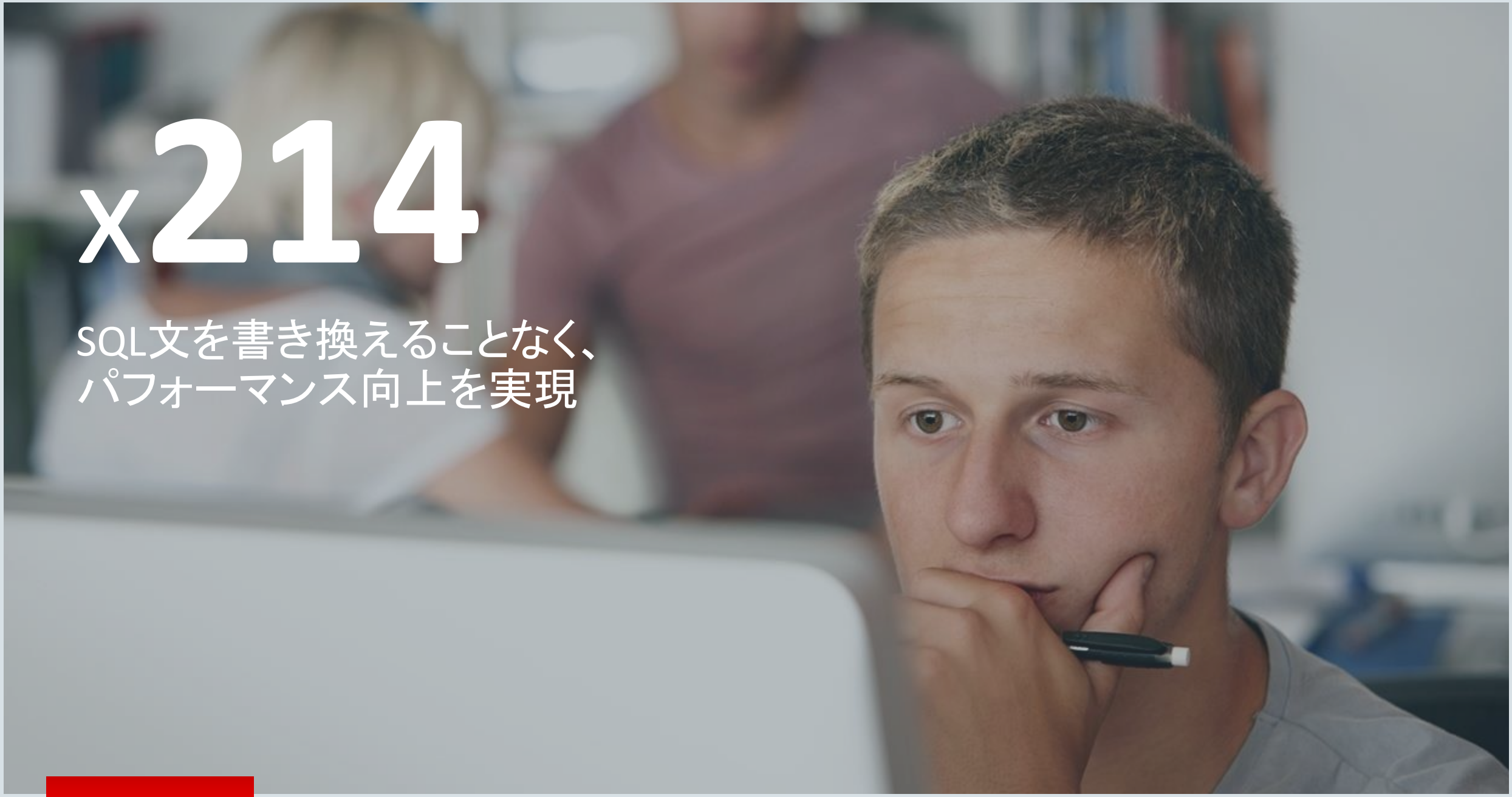
- SQL実行時間（Duration）: 7.0s(7秒) → 1.0以内(0.42秒*)へ改善
  - IO Bytes: 2GB → 34MB へ大幅に削減し、データ読込み時間がほぼゼロへ
  - I/O待機時間の割合が高い状態だったため、DBIMの効果が高い



Live Challenge!!

SQLパフォーマンスの高速化の限界を目指せ！

	Normal	+ Partitioning	+ Parallel Query	+ Compression	+ Database In-Memory
CPUバウンドなSQL 実行時間（秒）	108	72	20	9	7.67
相対比	x1	x1.5	x5.4	x12	x14
I/Oバウンドなsql 実行時間（秒）	90	19	---	7	0.42
相対比	x1	x4.7	---	x12	x214



ORACLE

パフォーマンス・チューニングの基本的な考え方

超有名な公式と同じ

時間↓ ＝ 処理量↓ / （速度 * 並列度）↑

じかん = みちのり ÷ はやさ

- 処理量を減らす
  - Index, **Partitioning**, **Compression**, Exadata Smart Scan/Storage Index, Database In-Memory Column Format/Storage Index, 実行計画の改善, …
- 高速化
  - Buffer Cache, **Database In-Memory**, Flash Device, InfiniBand, Exafusion, …
- 並列化
  - **Parallel Query**, Multi-Core, RAC, ASM, …

Oracle Database 12c Release 2

Oracle Cloud環境で今すぐ試してみてください！！

- 本セッションでご紹介した機能
  - パーティション化（Partitioning）
    - データアクセス範囲を限定
  - パラレル化（Parallel Query）
    - マルチコアの有効活用
  - データ圧縮（Compression）
    - I/Oボトルネックの改善
  - インメモリ化（Database In-Memory）
    - ディスクI/O時間の排除
    - インメモリ独自の高速演算

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

～ みなさまの投稿をお待ちしております ～

Twitter

#OracleTechNight

こんな時、かけこむ会社が増えています。

Oracle Digitalは、オラクル製品の導入をご検討いただく際の総合窓口。
電話とインターネットによるダイレクトなコニュニケーションで、どんなお問い合わせにもすばやく対応します。
もちろん、無償。どんなことでも、ご相談ください。

Oracle Digital

お問い合わせは電話またはWebフォーム

0120-155-096

受付時間:月~金9:00~12:00 / 13:00~18:00(祝日・年末年始休業日を除く)

http://www.oracle.com/jp/contact-us

ORACLE

Integrated Cloud

Applications & Platform Services


ORACLE®