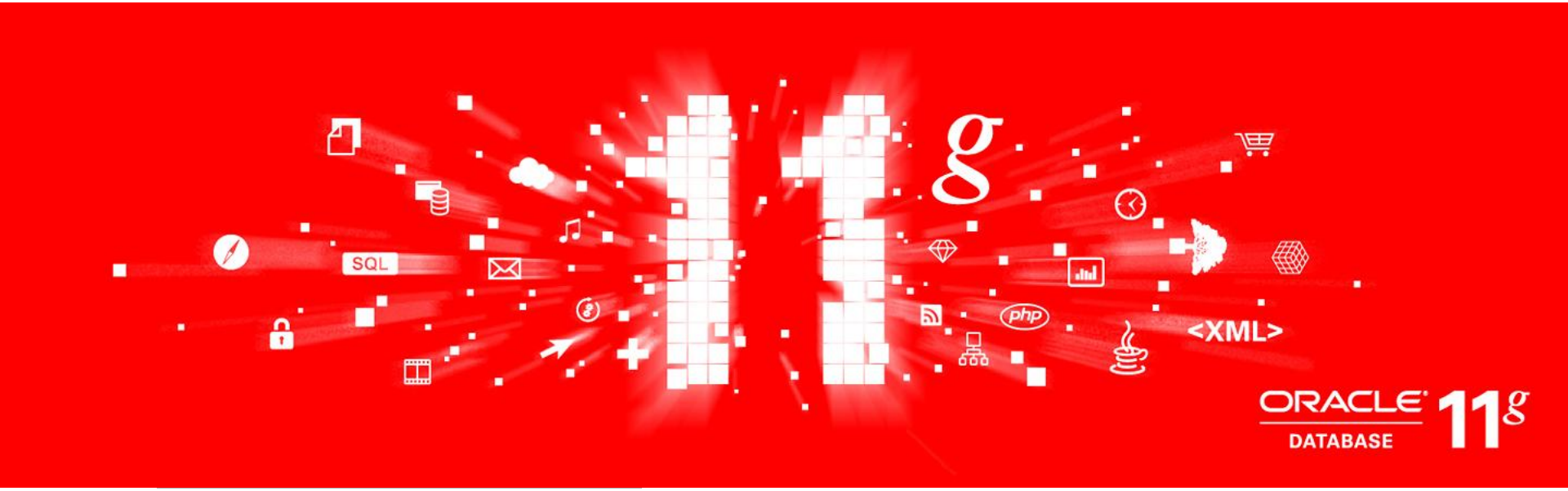


ORACLE®

ここまでできる!! Oracle Databaseのパラレル処理

日本オラクル株式会社　テクノロジー製品事業統括本部
アライアンス技術本部
データベース製品技術部

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。

Agenda

- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

近年のCPUはマルチコア化

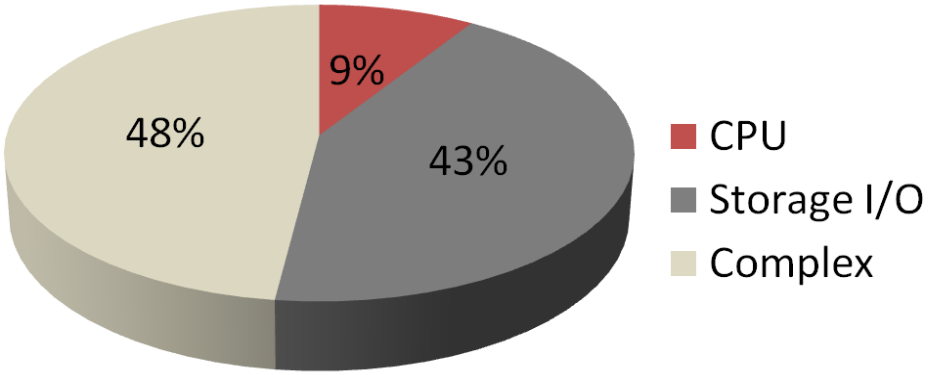
- 最近のCPU関連に関する話題
  - 2010/3/29　AMDは世界初の12コア/8コア 搭載 x86プロセッサー「AMD Opteron 6000 Series」を発表
    http://www.amd.com/jp/press-releases/Pages/amd-sets-the-new-standard-29mar2010.aspx
  - 2010/3/31　Intelは「Intel Xeon 7500番台」を発表。1チップ当たり最大8コア内臓、Hyper-Threading機能搭載
    http://www.intel.com/jp/intel/pr/press2010/100331a.htm
- 果たしてデータベースシステムでは、CPUの性能を使いこなせているのだろうか？

マルチコア化とデータベースの性能

Oracle Directのパフォーマンスクリニックの現状

CPUを追加すれば、性能問題は解決？

CPUがボトルネックだったケースは、わずか9%（弊社統計*）
→ マルチコアを使いきることができていない



性能ボトルネックの原因の傾向

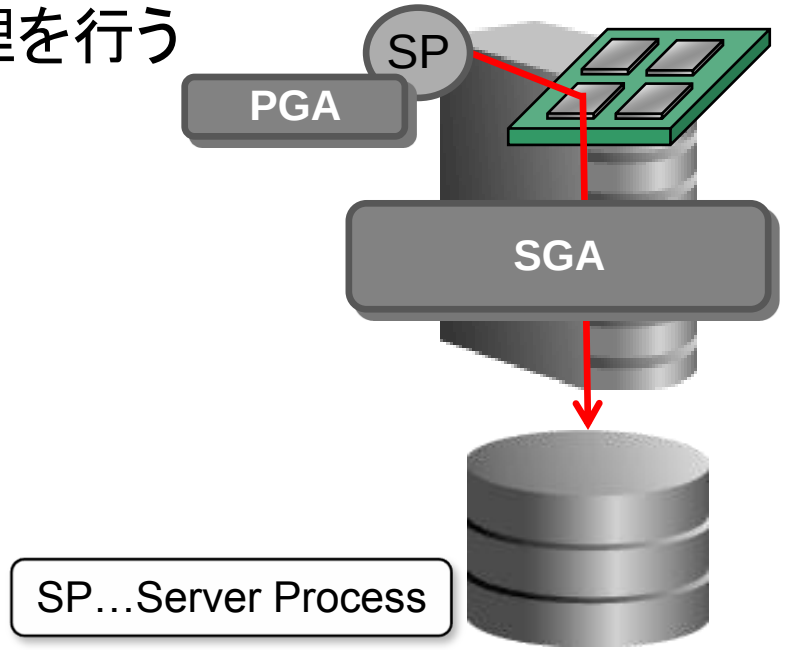
- CPU：9%
- ストレージI/O： 43%
- 非効率なSQL文、索引の設計等 ：48%

*データ：Oracle Directが直近で実施したパフォーマンスクリニック

http://www.oracle.com/lang/jp/direct/service/pc.html

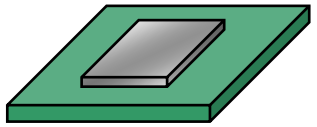
データベースアクセスとサーバープロセス

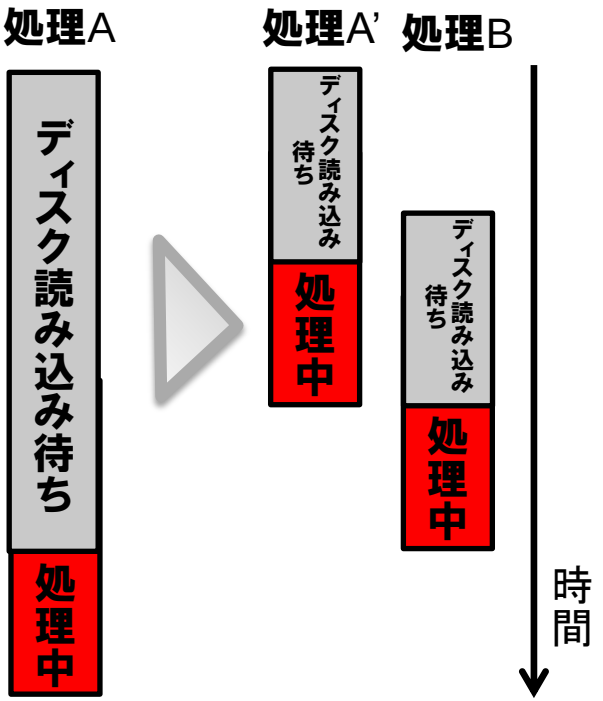
- クライアントの接続に対して、一つのサーバープロセスが生成される
  - 専用サーバー構成の場合
- SQLはサーバープロセスが処理を行う
- データへのアクセス方法
  - 基本は以下の2パターン
    - 全表スキャン
    - 索引アクセス



CPU使用率

- シリアル実行では、待ち時間に比例して、CPU使用率は低下
  - 大量データの検索によりディスクI/O待ち　など
- より短時間にデータを検索できれば、CPU使用率は高くなる
  - キャッシュヒット率の向上
    - OLTP系システムでのチューニング
  - キャッシュヒット率が向上しない
    - DWH系システムのクエリー



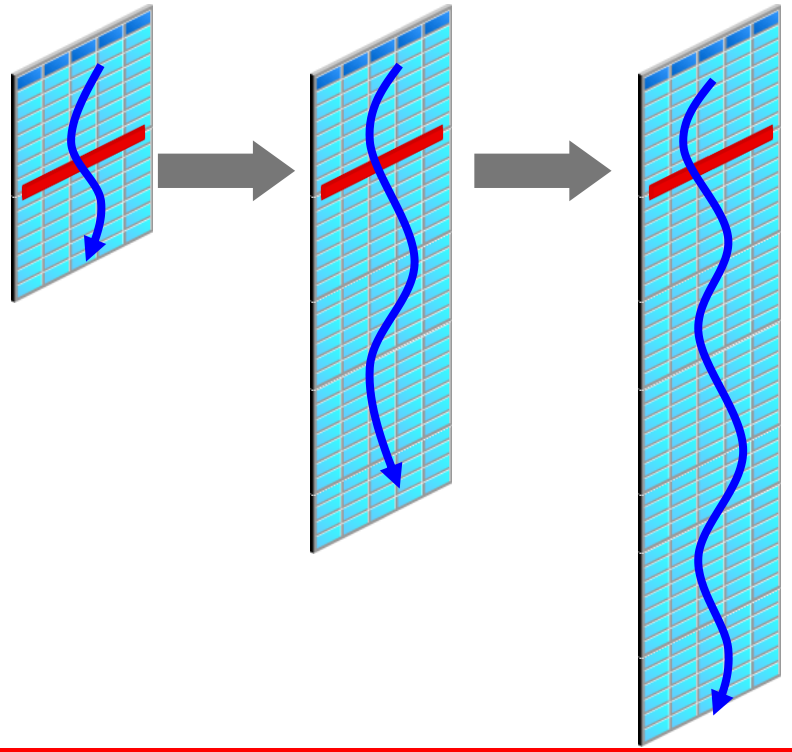


Oracle Databaseでの検索

行を特定する

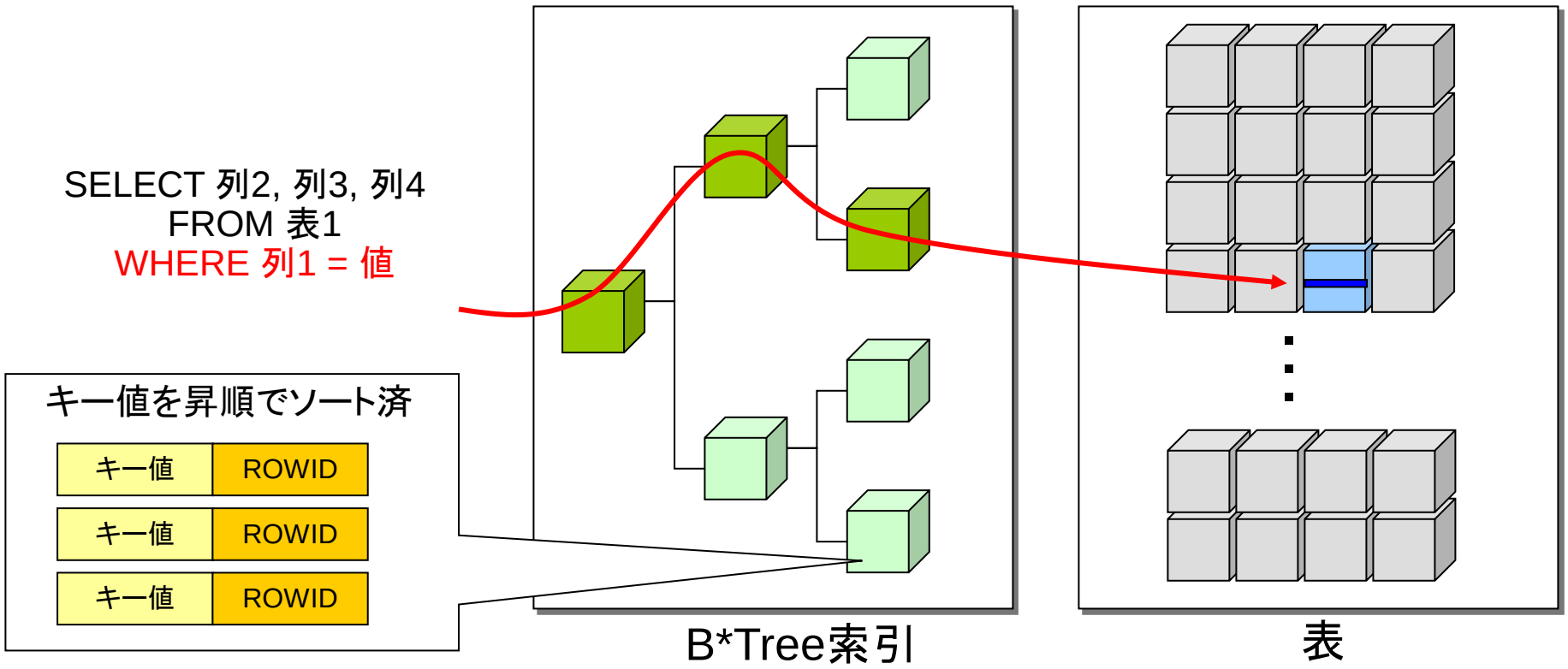
- 該当する行を表の最初から最後まで検索する
  →表フルスキャン
- 表フルスキャンでは、表の行数の増加に比例して、実行時間も増加してしまう

SELECT 列2, 列3, 列4 FROM 表1
WHERE 列1 = 値



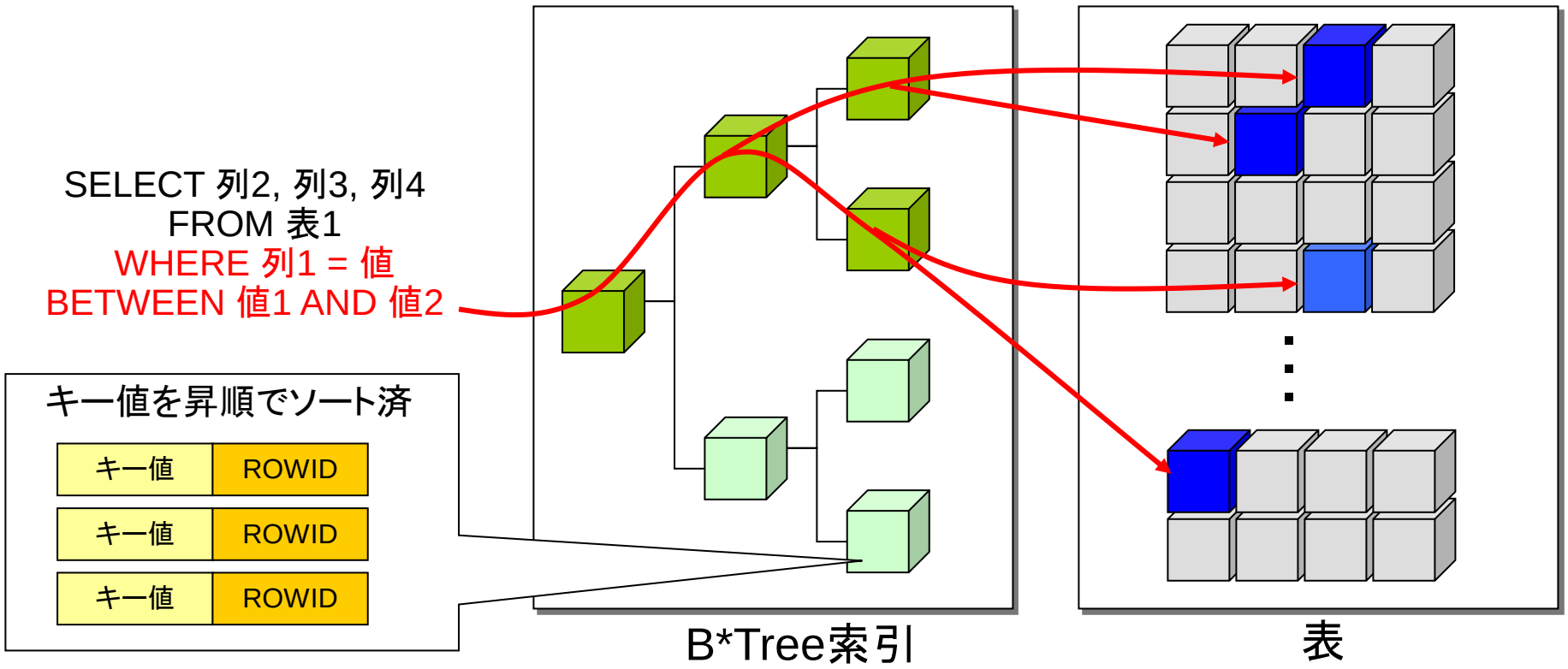
行を短時間で特定する
索引の使用

- 索引を使用することで、表フルスキャンよりもはるかに少ないブロック数へのアクセスで済む



複数の行にアクセスする

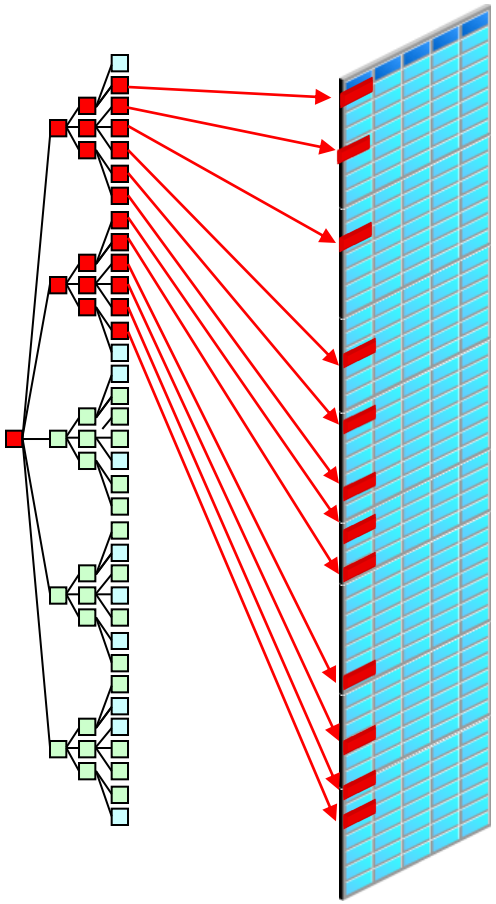
- B*Tree索引は範囲検索は得意



さらに多くの行にアクセスする

- 索引ブロック→表ブロックのアクセス

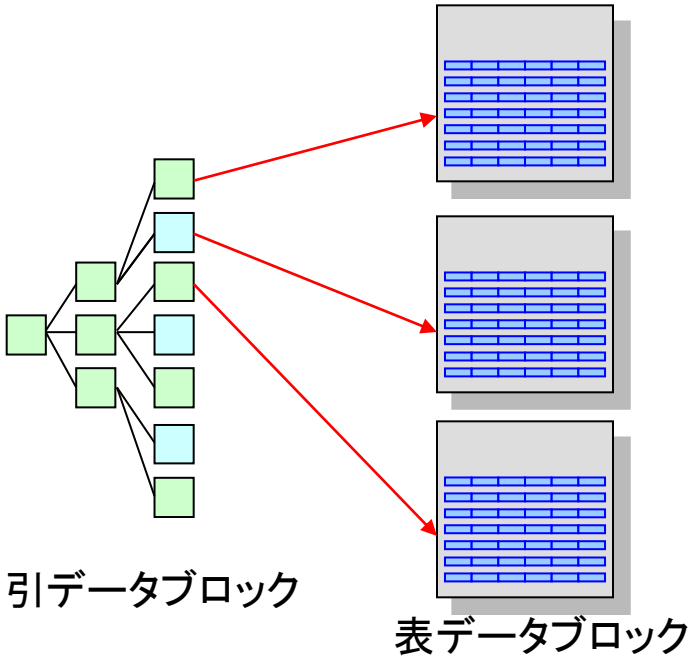
一般的に
「1つのSQLが10%～20%以上の
行数にアクセスするならば
表フルスキャンのほうが高速」
とされている



索引アクセス

10%～20%の行にアクセスすると

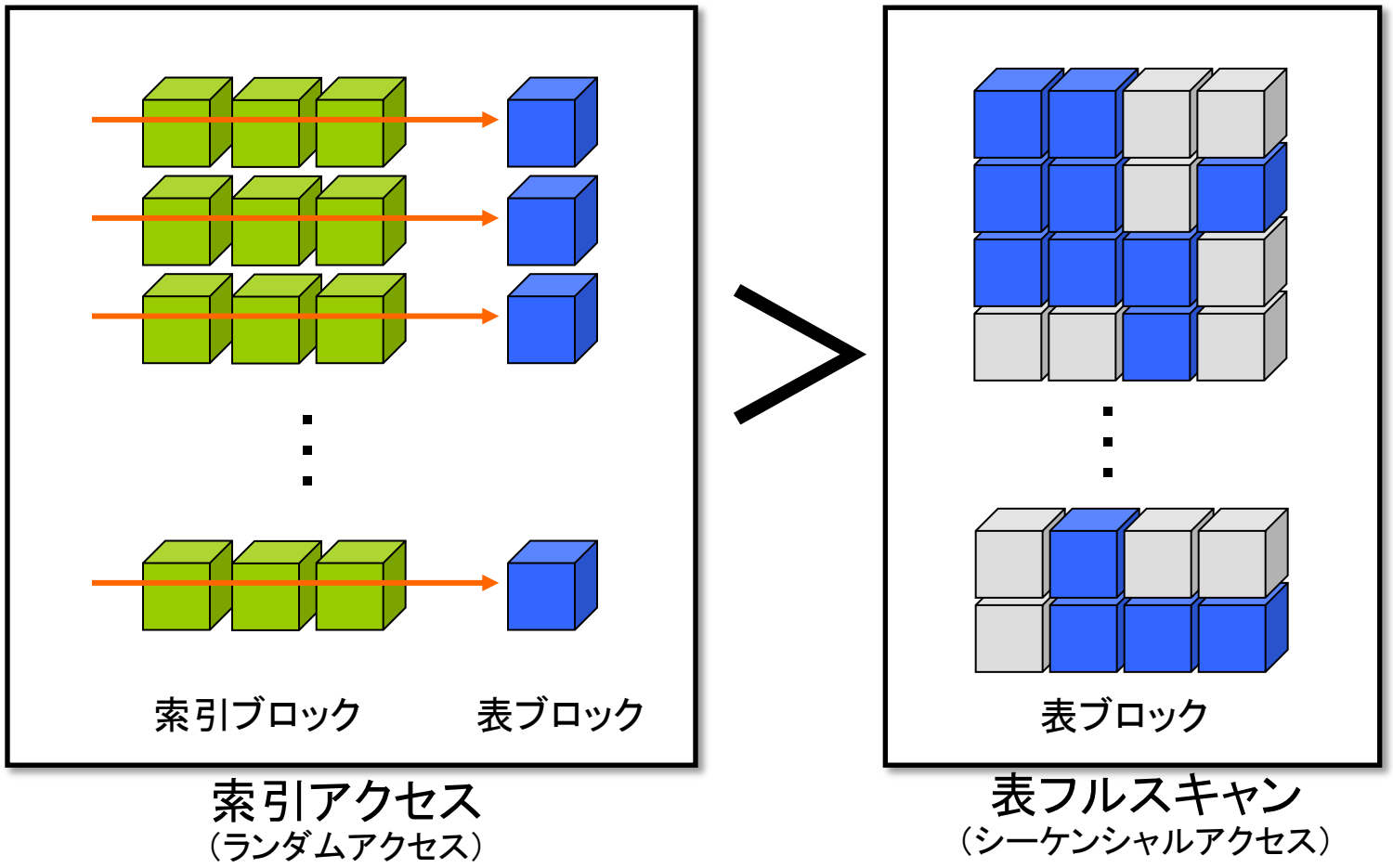
- 1つのブロックには複数の行が格納されている
- 仮に、アクセスする行が均等に分散しているとすると、ほとんどのブロックにアクセスすることになる



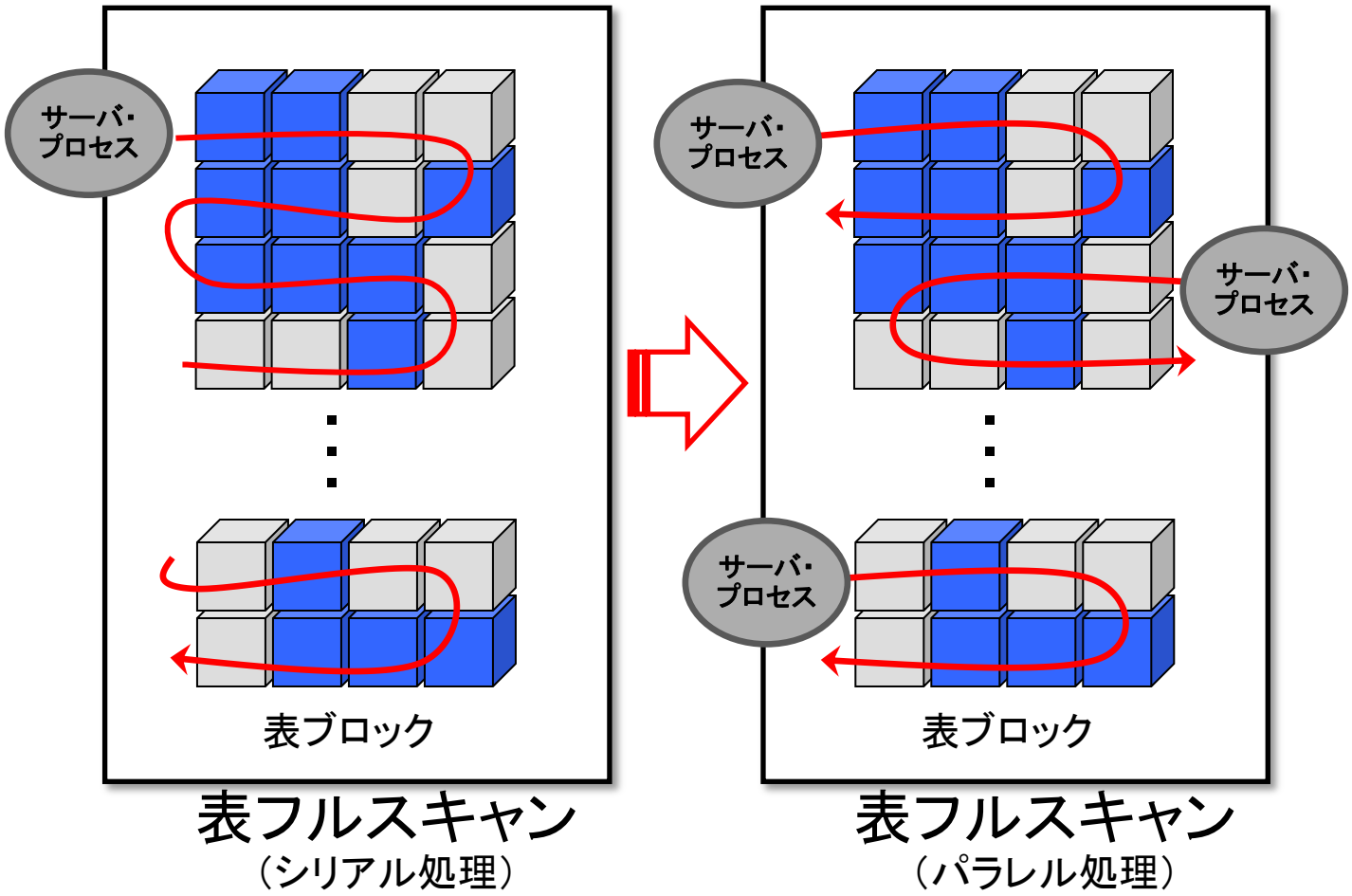
ディスクI/Oの最小単位は「データブロック」
→ 1行読むのにも1ブロックを取得する

アクセスコストが逆転する

1つのSQLが多くの行にアクセスする場合



表フルスキャンを早くする



- 単一プロセスでの処理を分割して実行
- パラレル処理

Oracle Databaseでできるパラレル処理

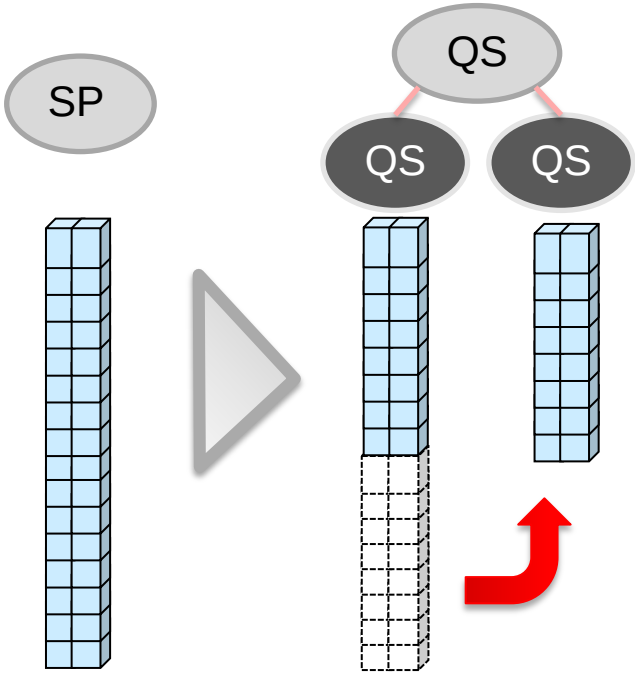
- 検索処理のパラレル化
  - パラレルクエリー
- メンテナンス/データロードのパラレル化
  - パラレルDDL/パラレルDML
- バックアップの取得のパラレル化
  - Datapumpのパラレル化
- 統計情報の取得のパラレル化
  - DBMS_STATSパッケージのParallel実行

Agenda

- 最新CPUとデータベースシステム
- 検索処理のパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

パラレルクエリー
パラレルクエリーとは

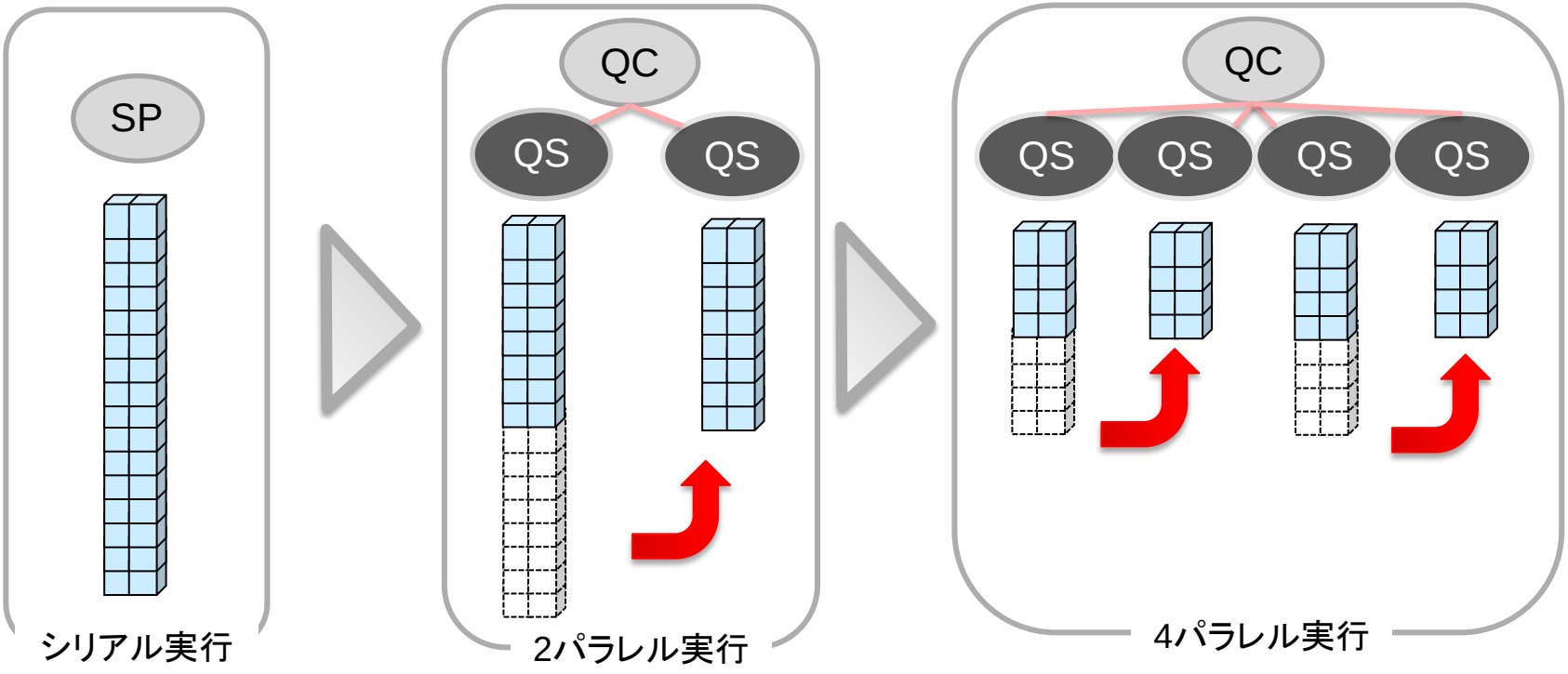
- 単一のクエリーを複数プロセスを使用して実行する機能
- **Oracle Database Enterprise Edition**の標準機能
- **アプリケーションからは透過的**
- クエリーコーディネータ(QC)と
  クエリースレーブプロセス(QS)
  - QC:クエリーの解析、並列度の決定、QSへ命令を出す
  - QS:QCからの命令に基づき、実際に処理を実行する



パラレルクエリー

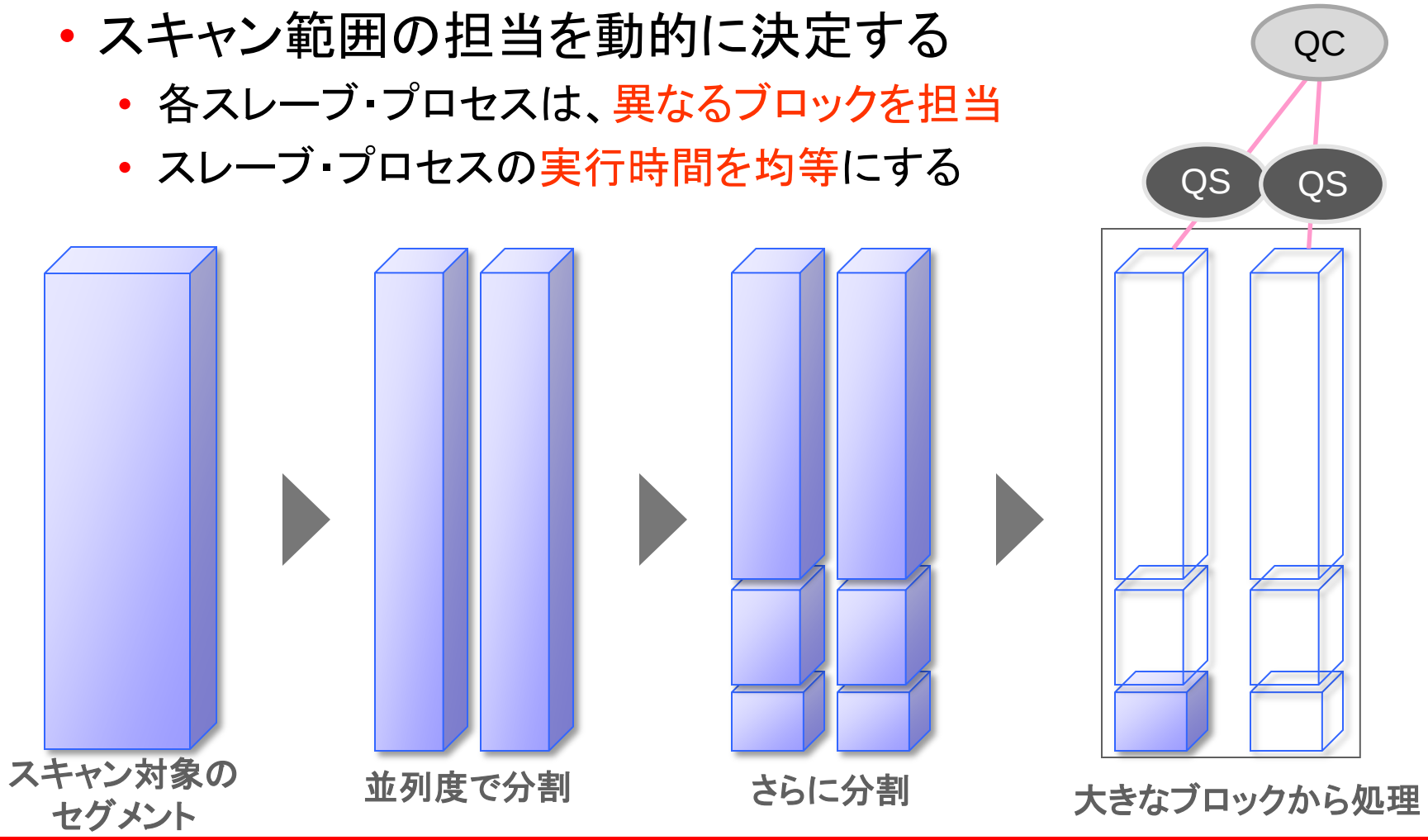
パラレル度の向上による高速化

- パラレル度をX倍すれば実行時間も約1/X倍になる
  （リソースが許す限り）
- ディスクI/Oが激しいDWH系のクエリに対して非常に有効



スレーブプロセスのデータ読み込みの方法

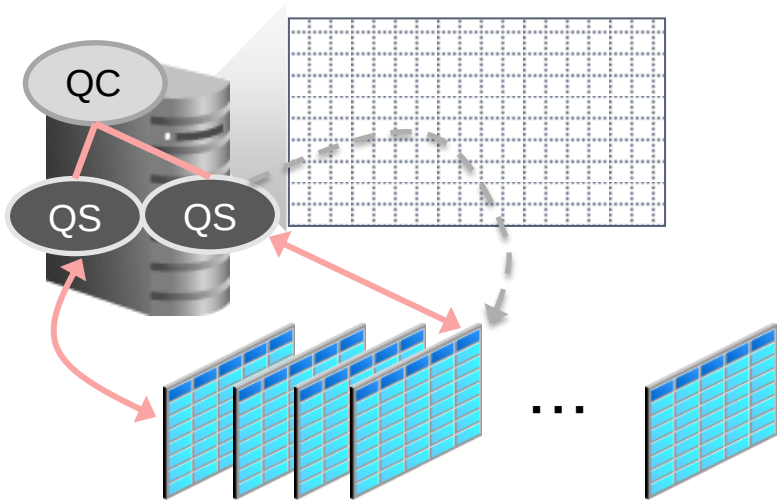
- スキャン範囲の担当を動的に決定する
  - 各スレーブ・プロセスは、異なるブロックを担当
  - スレーブ・プロセスの実行時間を均等にする



パラレルクエリーでのデータアクセス
ダイレクト・パス・リード

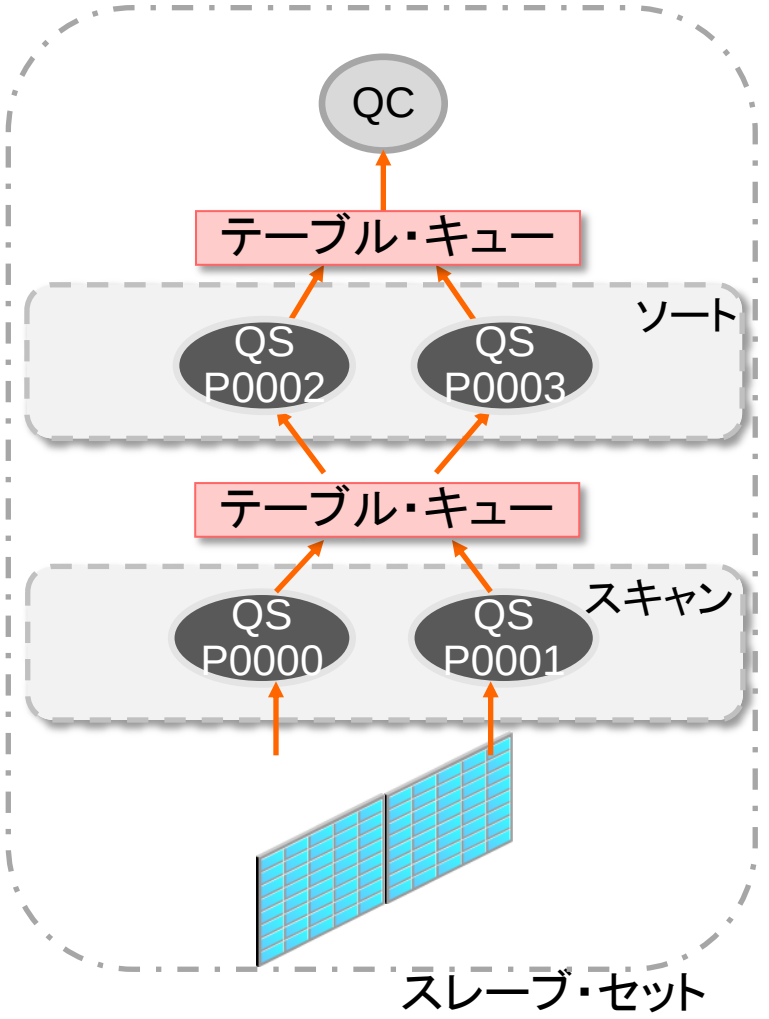
- メモリサイズとアクセスするデータ量の関係
  - メモリサイズ< パラレルクエリーがアクセスするデータ量
  - キャッシュされたデータがすぐにキャッシュアウトされてしまう可能性
  - キャッシュ管理のオーバーヘッドが無駄に生じてしまう

- パラレルクエリー実行時には Direct Path Readによるアクセス
  - メモリ上のデータへのアクセスをバイパス
  - アクセスしたデータをメモリ上にキャッシュをしない
  - キャッシュ管理のオーバーヘッドの削減



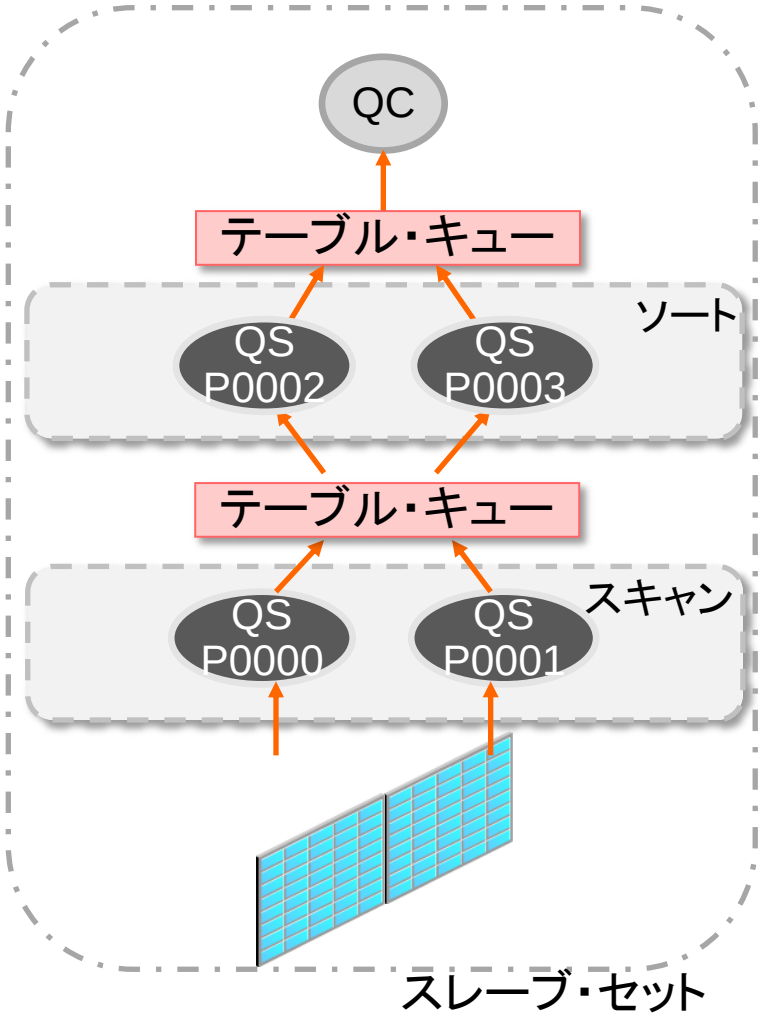
パラレル化のアーキテクチャー

- クエリ・コーディネータ(QC)
  - パラレル問合せを発行したセッションのサーバ・プロセス
  - 問合せを解析し、並列度を決定し、クエリ・スレーブにパラレル処理の命令を出す
- クエリ・スレーブ(QS)
  - バックグラウンド・プロセスのパラレル実行サーバ(Pxxx)
  - パラレル化された処理を実施
- メッセージ・バッファ
  - プロセス間の通信、データのやりとりで使用
  - デフォルトでは、共有プールからパラレル実行バッファが割り当てられる
  - SGA_TARGETが設定されている場合、ラージプールから割り当てられる



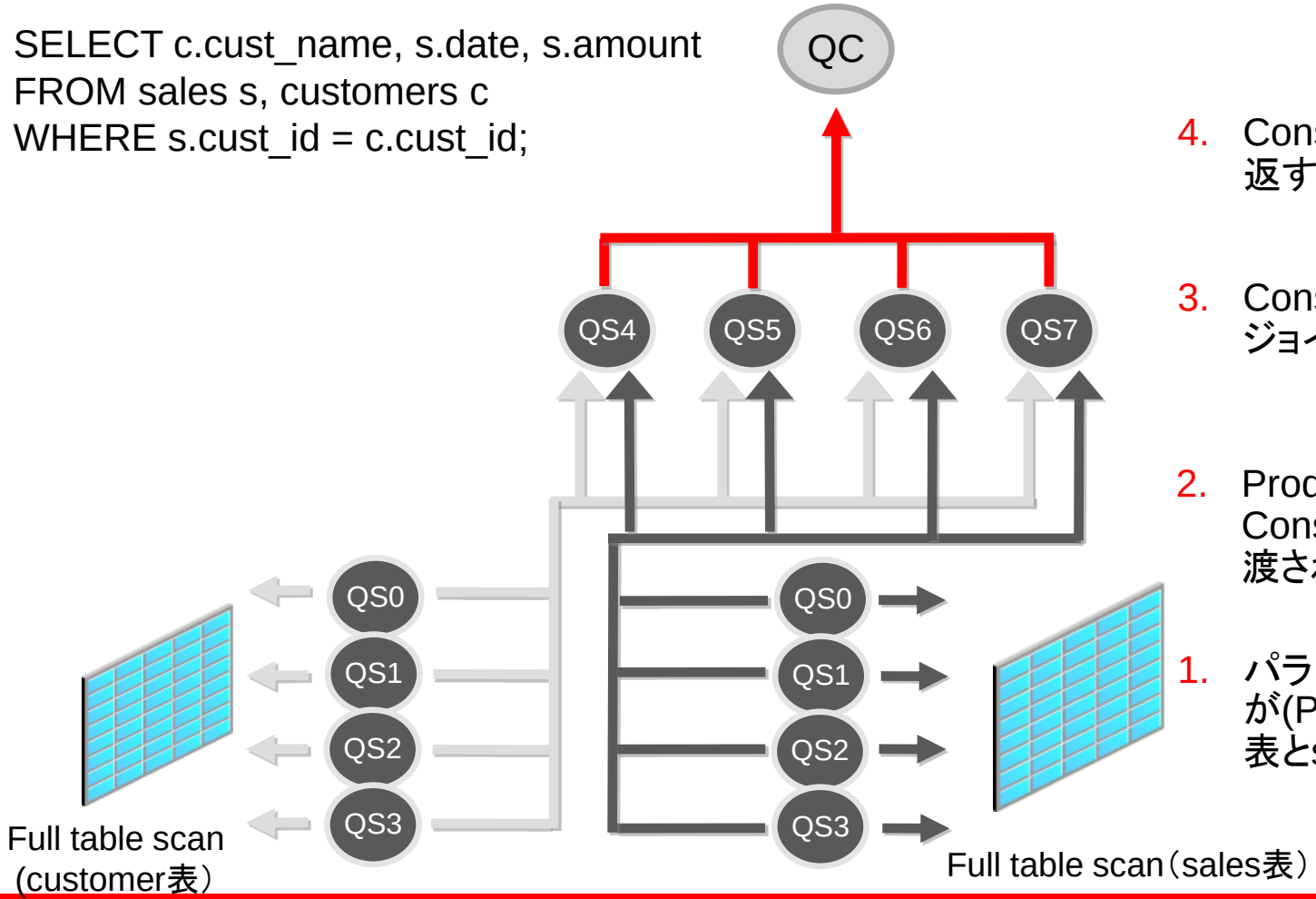
パラレル化のアーキテクチャー

- テーブル・キュー（TQ）
  QCとQS、またはQS同士がプロセス間でデータの受渡しを実装している構造の総称
- プロデューサー：
  実行計画中のある処理を実行してTQに結果を送るプロセス
- コンシューマー：
  TQからデータを取得して処理を行うプロセス。あるQSがタイミングによってはプロデューサーの役割を担い、別のタイミングではコンシューマの役割を担う
- スレーブセット：
  同一のオペレーションをパラレルに実行するQSのグループ



パラレルクエリーでの結合処理
プロデューサー&コンシューマモデル

```
SELECT c.cust_name, s.date, s.amount
FROM sales s, customers c
WHERE s.cust_id = c.cust_id;
```



4. ConsumerはQCに結果を返す
3. Consumerはハッシュジョインを実行
2. Producerのセットから Consumerのセットに行が渡される
1. パラレルサーバープロセスが(Producer)がCustomer表とsales表を検索

並列度（パラレル度）の決定

- 並列度
  - その処理で使用するスレーブ・プロセスの個数
- 並列度の指定
  - 表、索引のパラレル属性で定義
  - パラレルクエリー有効化時に指定
  - ヒント句にて指定
  - Oracle Databaseによる自動設定 11g R2新機能

パラレル化の方法

- 表や索引に対してパラレル属性を定義する

```
create table tablename … parallel 4;
```

- alter session force parallelでセッションに対して設定する

```
alter session force parallel query parallel 4;
        --  実行するクエリー  --
```

- ヒント句をSQLに埋め込む

```
select /* parallel (emp) */ * from emp;
```

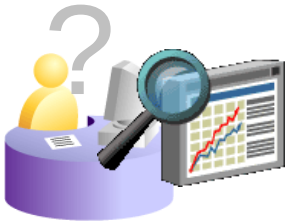
- 初期化パラメータを変更する

```
alter system set parallel_degree_policy=limited;
          または
alter system set parallel_degree_policy=auto;
```

11g R2 新機能 自動並列度設定

従来のパラレル度設定

- 最適なパラレル実行のためには、コストがかかる
  - 全てのクエリーに対して、単一のパラレル度が最適とは限らない
    - それぞれのクエリーに対して、最善のパラレル度を設定
    - データ量の増減に合わせたパラレルどの設定
  - DBAの大きな負担
    - コストが高いクエリーの調査、調整



Oracle Database 11g R2以降

- 最適なパラレル実行の容易な実行
  - クエリーの特性に合わせた最適なパラレル度の設定
    - Oracle自身がパラレル度を設定する
- DBAの負担の大幅な削減
  - 初期化パラメータの設定のみ

自動パラレル度設定
設定方法

- PARALLEL_DEGREE_POLICYで設定
  - ✓PARALLEL_DEGREE_POLICY=LIMITED もしくは AUTOに設定
  - ✓alter session 文もしくはalter system 文で変更可能
    - ✓ alter system 文での変更

```
alter system set parallel_degree_policy=AUTO scope=both;
```

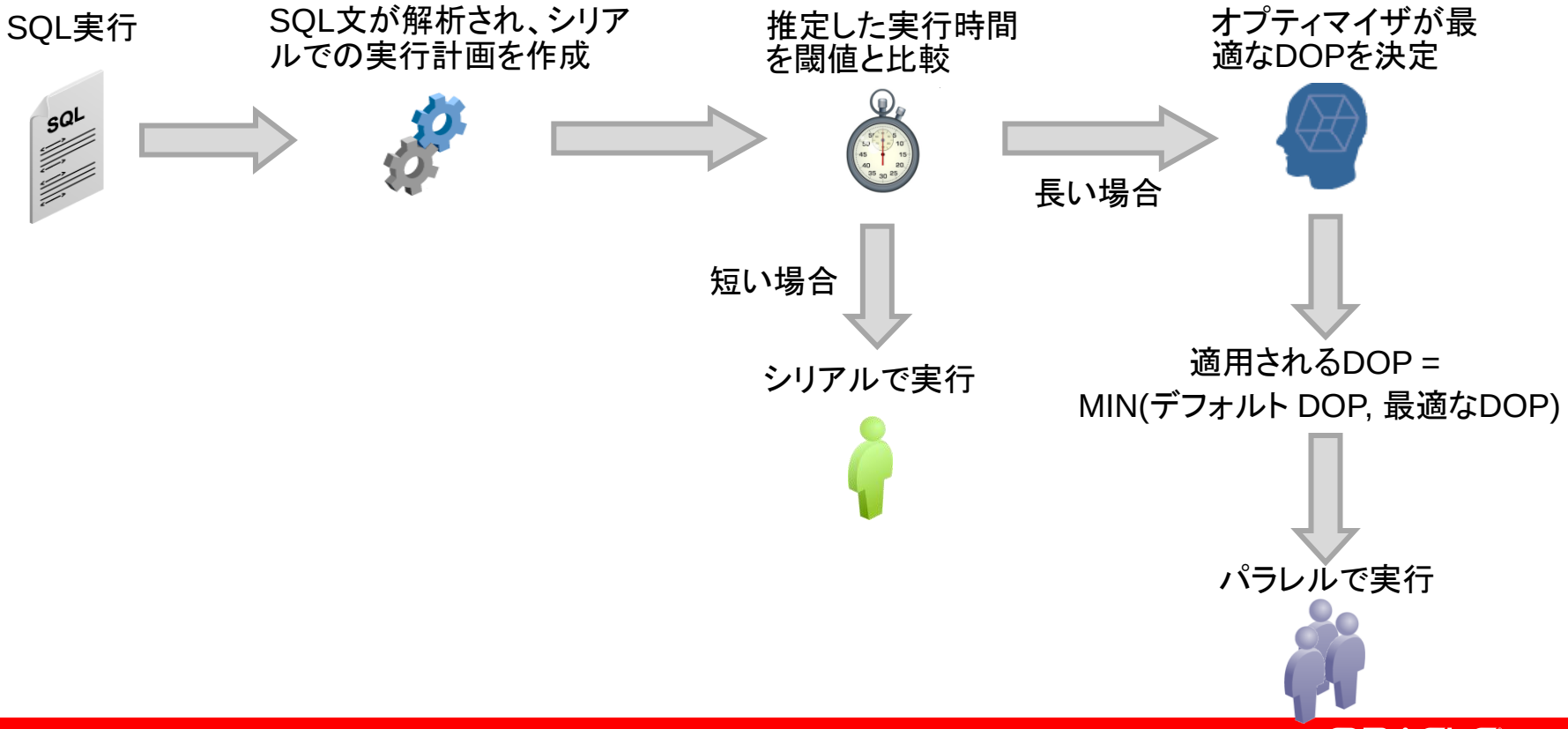
    - ✓ alter session 文での変更

```
alter session set parallel_degree_policy=AUTO;
```

自動パラレル度設定

動作概要

- 自動パラレル度設定の動作概要は以下の通り

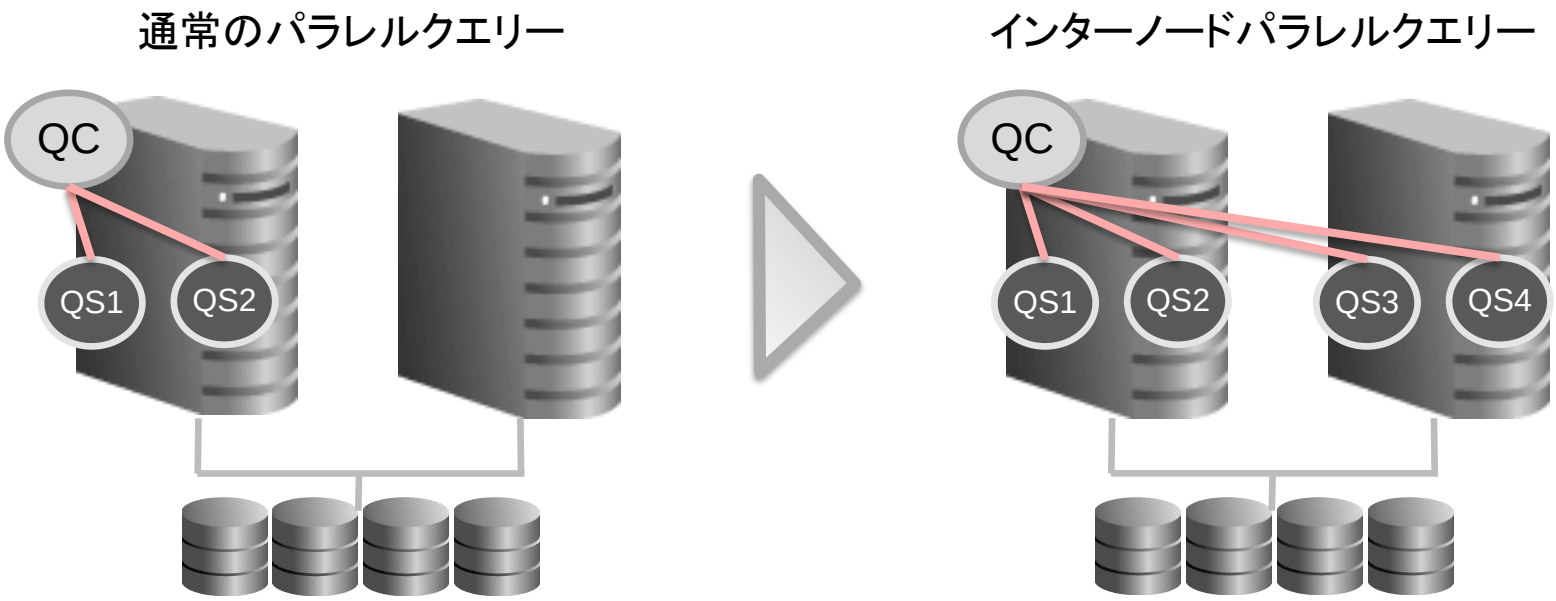


Agenda

- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

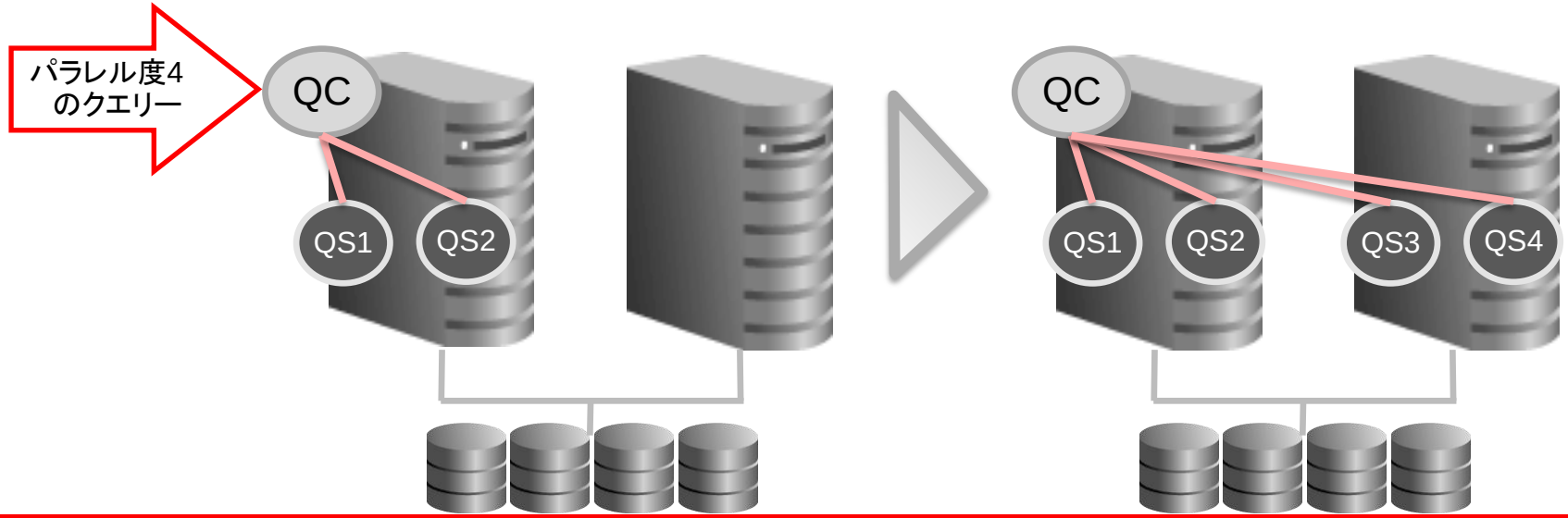
RAC環境でのパラレルクエリー

- SQLを並列化することで、ノード追加による性能向上が可能
  - 1つのSQLを内部的に並列化 → Parallel Query/DML/DDL
  - 1つのSQLを複数ノードで並列化 → Internode Parallel Query/DML/DDL
    - 並列度は、CPU数や負荷状況等に依存



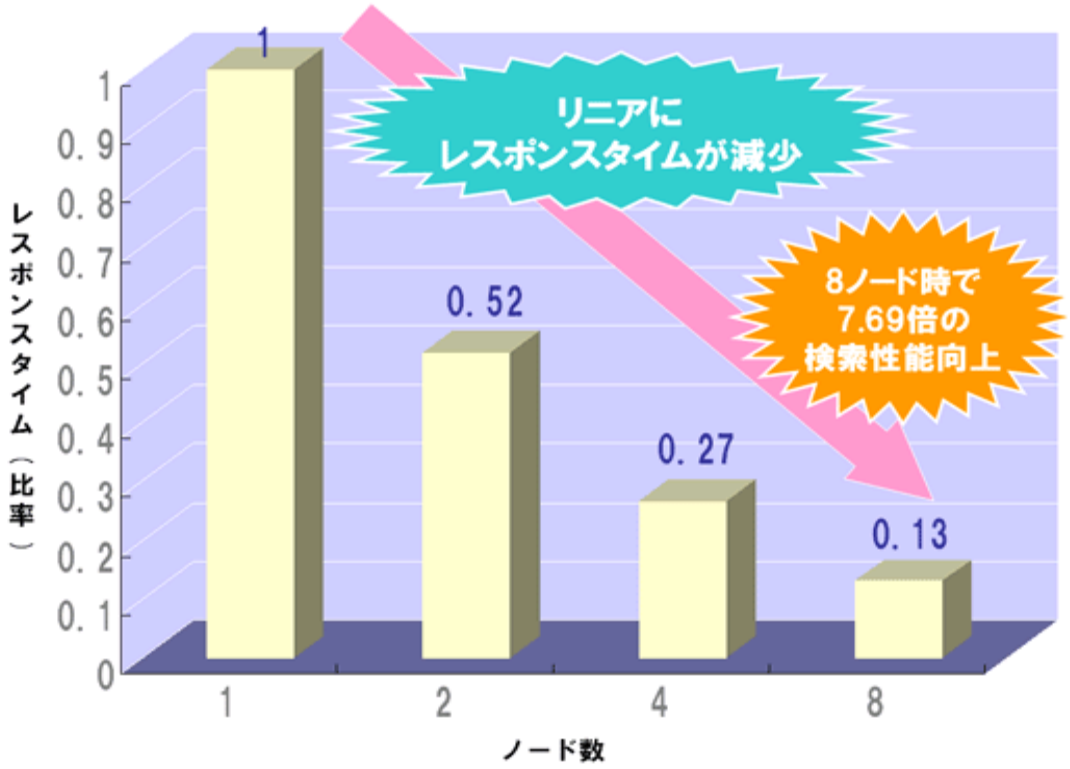
RAC環境でのパラレルクエリー

- 基本的には、単一インスタンス上で実行される
  - インターコネクト上のトラフィックを最小限にするため
- QS数が単一インスタンス上では足りない場合、複数ノードのインスタンスを利用し、実行される
  - ✓例：単一インスタンス上で起動できるQSが2の場合



Internode Parallel Queryによる性能向上

- http://www.nec.co.jp/middle/oracle/gc1.html
- Internode Parallel Query機能により、ノード追加に伴い性能が向上することを実証

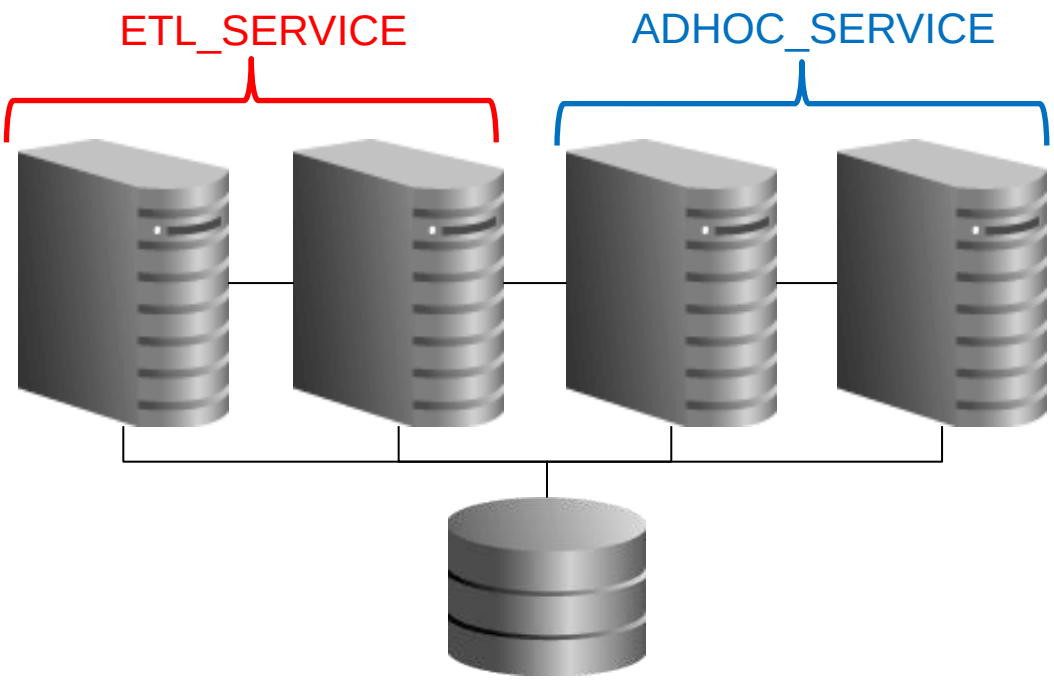


RACサービスとパラレルクエリー

- RACサービスと組み合わせることで、QSが起動するインスタンスを調整可能

```
srvctl add service –d dwhvm
      -s ETL_SERVICE
      -n dwhvm1,dwhvm2

srvctl add service –d dwhvm
      -s AHOC_SERVICE
      -n dwhvm3,dwhvm4
```

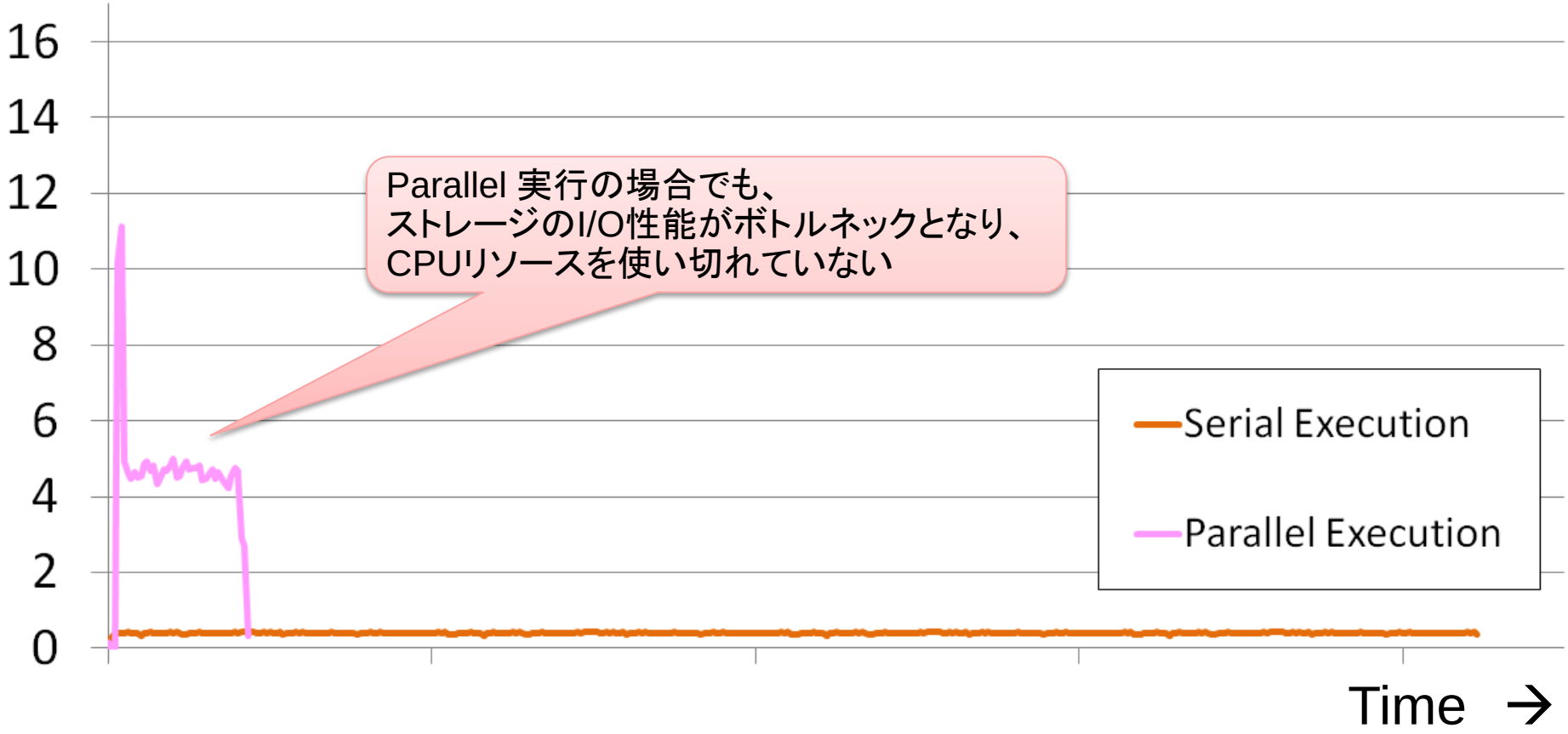


パラレルクエリーに関する昨今の課題

- 課題
  - パラレルクエリーを利用した際の性能の伸び悩み
- 背景
  - ユーザーが所持するデータ量の大容量化
  - CPUの大幅な性能向上と低価格化
  - サーバーに搭載可能なメモリの大容量化と低価格化
  - 旧世代のハードウェアに最適化されたままのアーキテクチャ

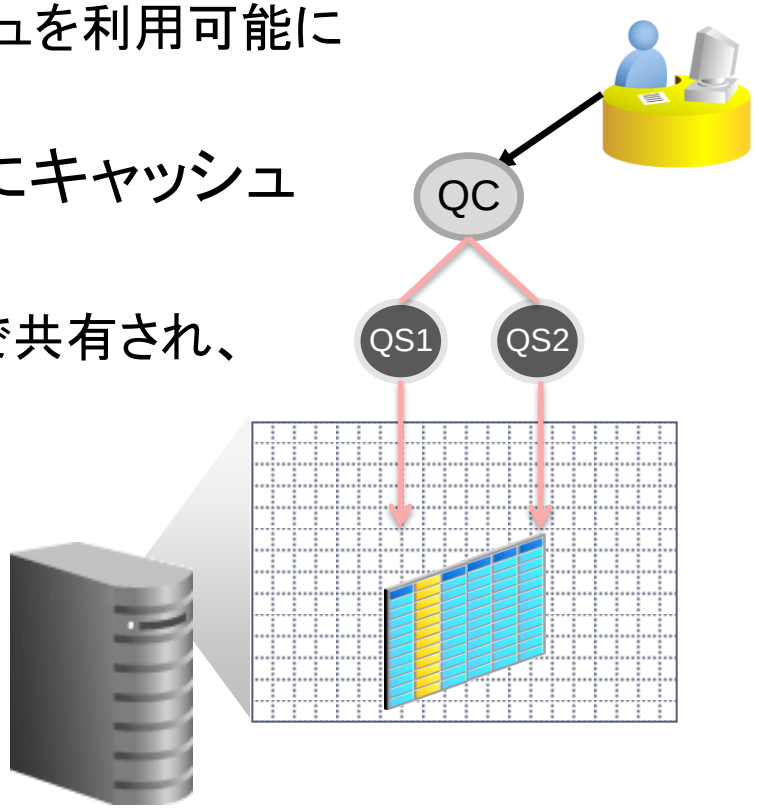
パラレル実行によるSQLの高速化

検証結果（CPU使用率）



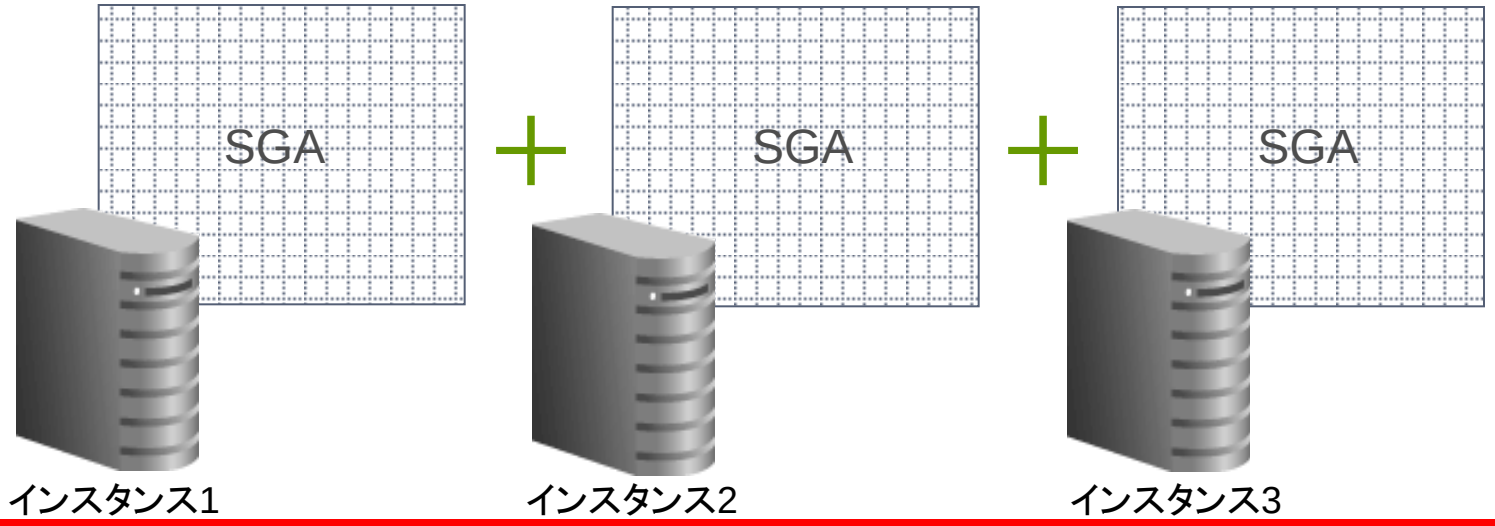
11g R2新機能　In-Memory Parallel Query
概要

- パラレルクエリー実行時の、メモリ使用効率の最適化
  - ✓パラレルクエリーでもバッファ・キャッシュを利用可能に
- パラレルクエリー実行時、メモリ上にキャッシュされたセグメントにアクセス
  - ✓キャッシュされたデータはユーザー間で共有され、クエリレスポンスを高速化
  - ✓メモリやCPUリソースを有効活用
- 設定方法
  - ✓PARALLEL_DEGREE_POLICYをAUTOに設定する



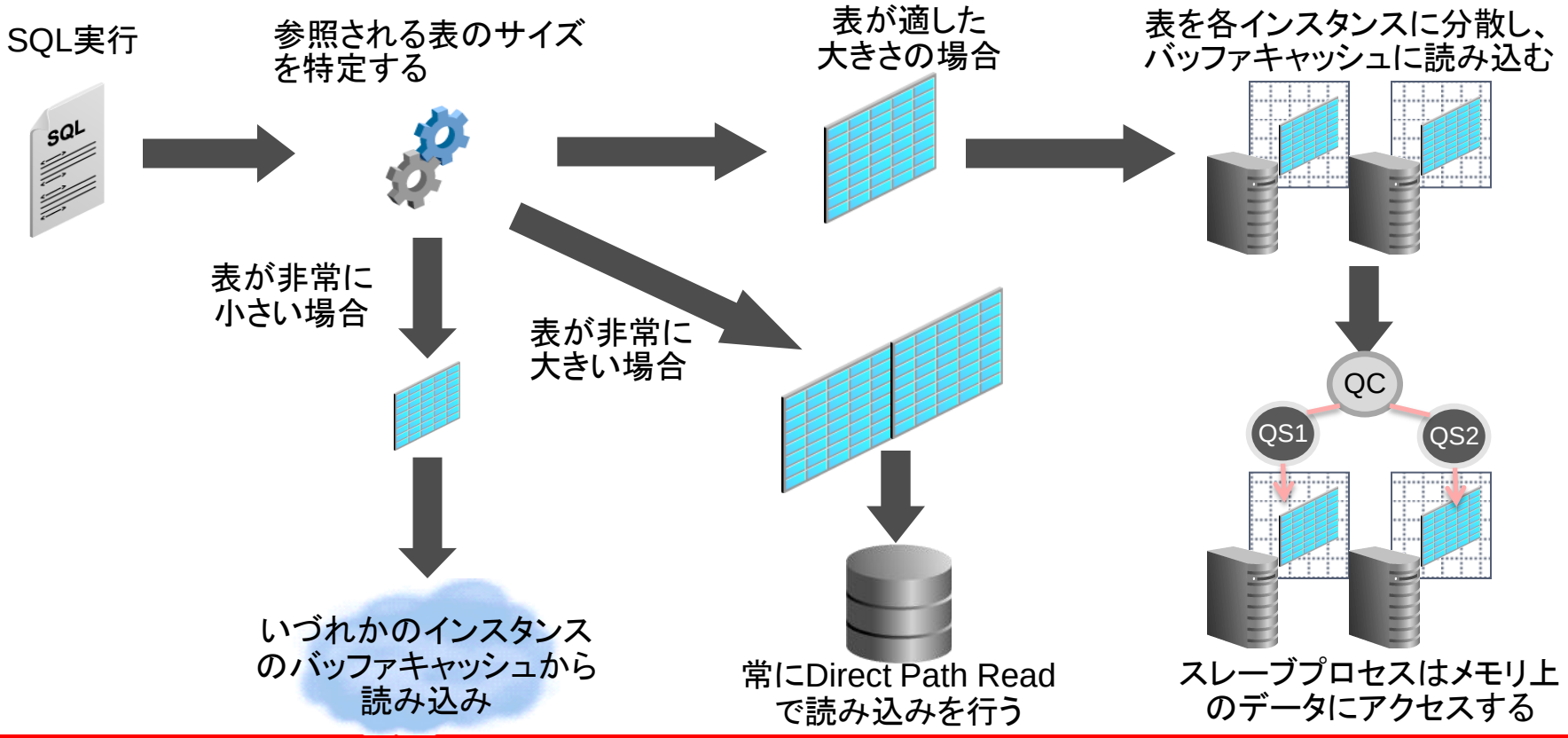
In-Memory Parallel Query
複数インスタンスのSGA利用

- 複数インスタンスのSGAを利用してデータをキャッシュ
  - RAC環境では、複数インスタンスのSGAを利用可能
  - インスタンス全体でメモリ空間を有効活用できる
  - 複数インスタンスにセグメントを分散してキャッシュ可能



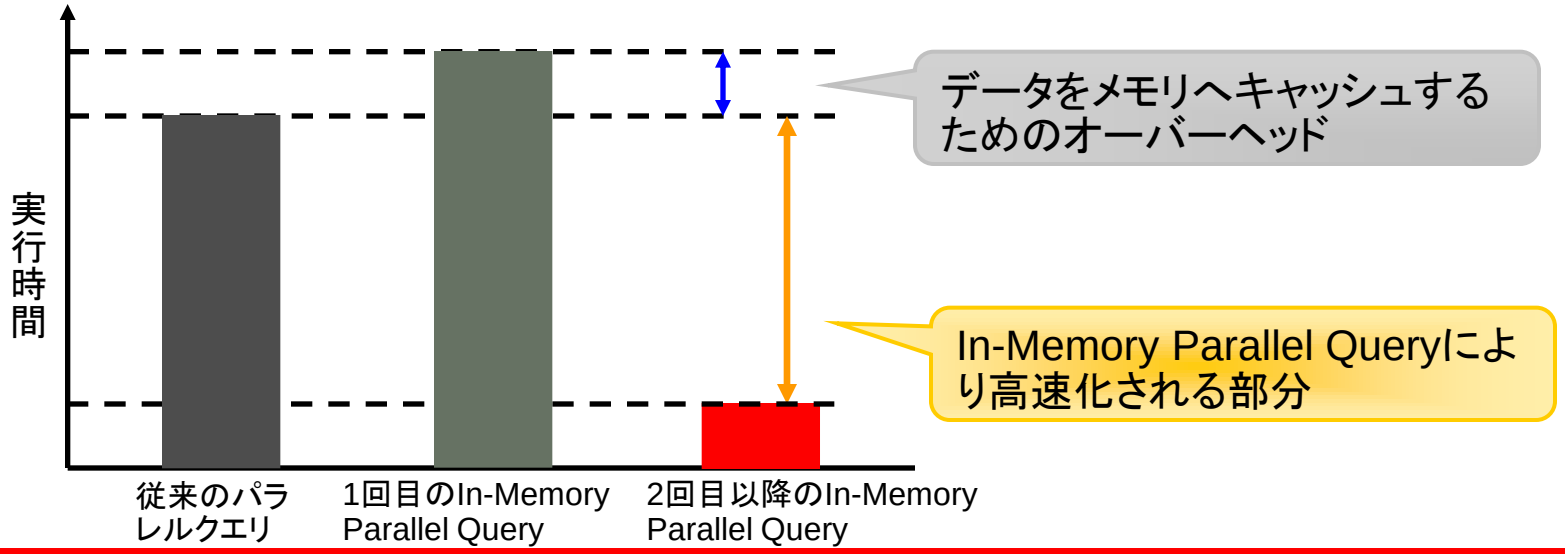
In-Memory Parallel Query
動作概要

- In-Memory Parallel クエリーの動作概要は以下の通り



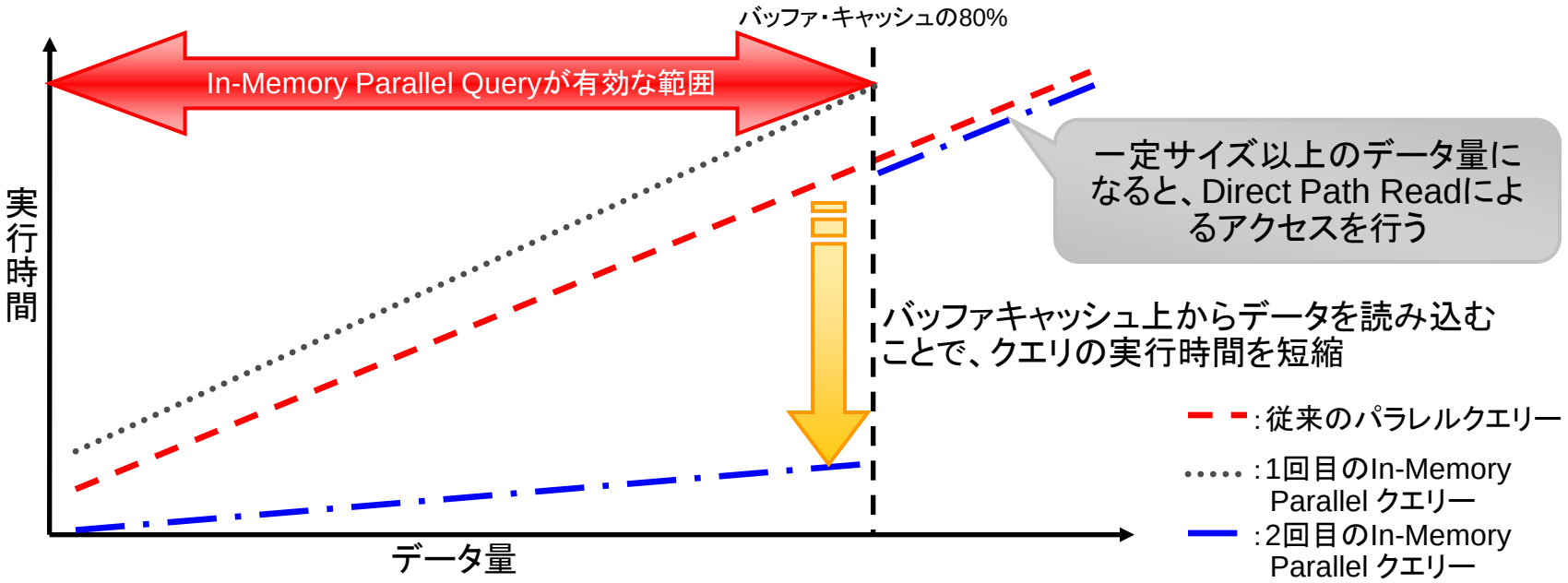
In-Memory Parallel Query
メリットとオーバーヘッド

- 高速化のためには、メモリへの読み込みが必要
  - ✓最初のアクセス時にバッファ・キャッシュ上へデータを読み込む
  - ✓多少のオーバーヘッドが生じる
  - ✓その後のクエリーの高速化により、オーバーヘッドは相殺される
  - ✓大量のデータに複数回アクセスする処理に非常に効果的



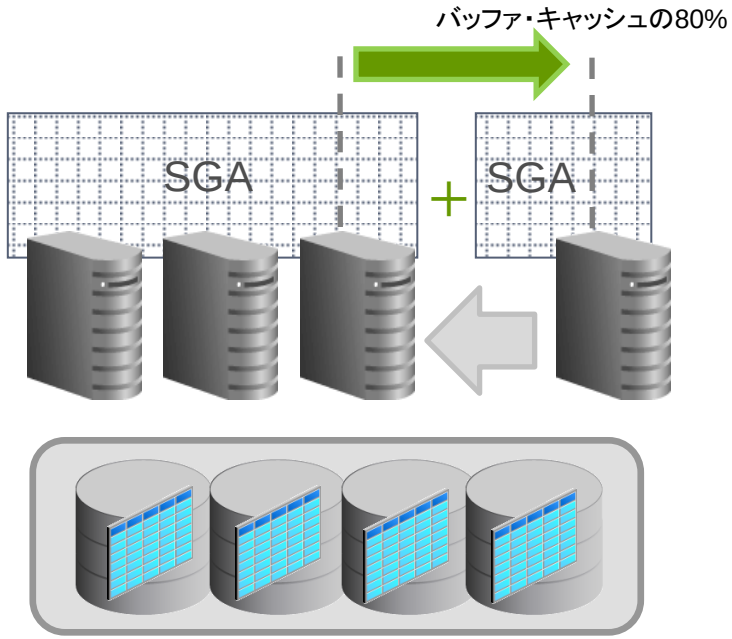
In-Memory Parallel Query
クエリー実行時間のイメージ

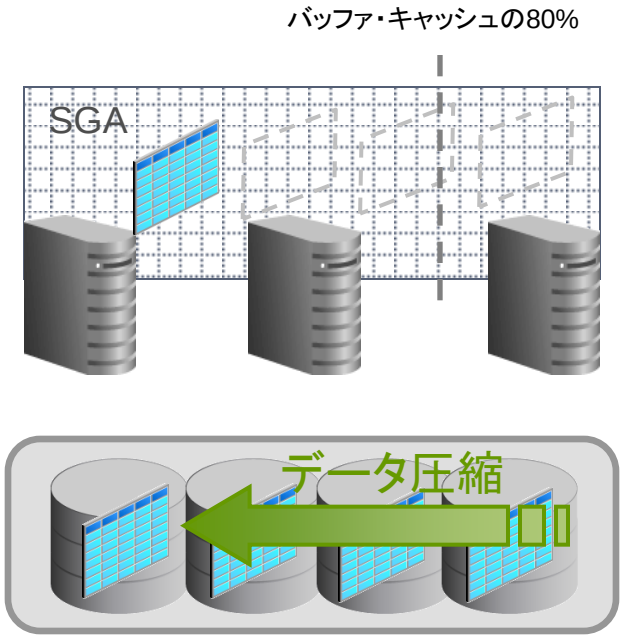
- 処理時間の変化のイメージ
  - ✓ In-Memory Parallel Queryを利用した場合の処理時間のイメージ
    - ✓ キャッシュするデータ量が多くなるほど、In-Memory Parallel Queryによるメリットは大きくなる



In-Memory Parallel Queryと他の機能の組み合わせ

- RACとの組み合わせ
  - バッファ・キャッシュのサイズを増やすことにより、キャッシュ可能なデータ量を増やす
- データ圧縮との組み合わせ
  - データサイズを圧縮することで、圧縮率に応じてキャッシュできるデータ量を増やす



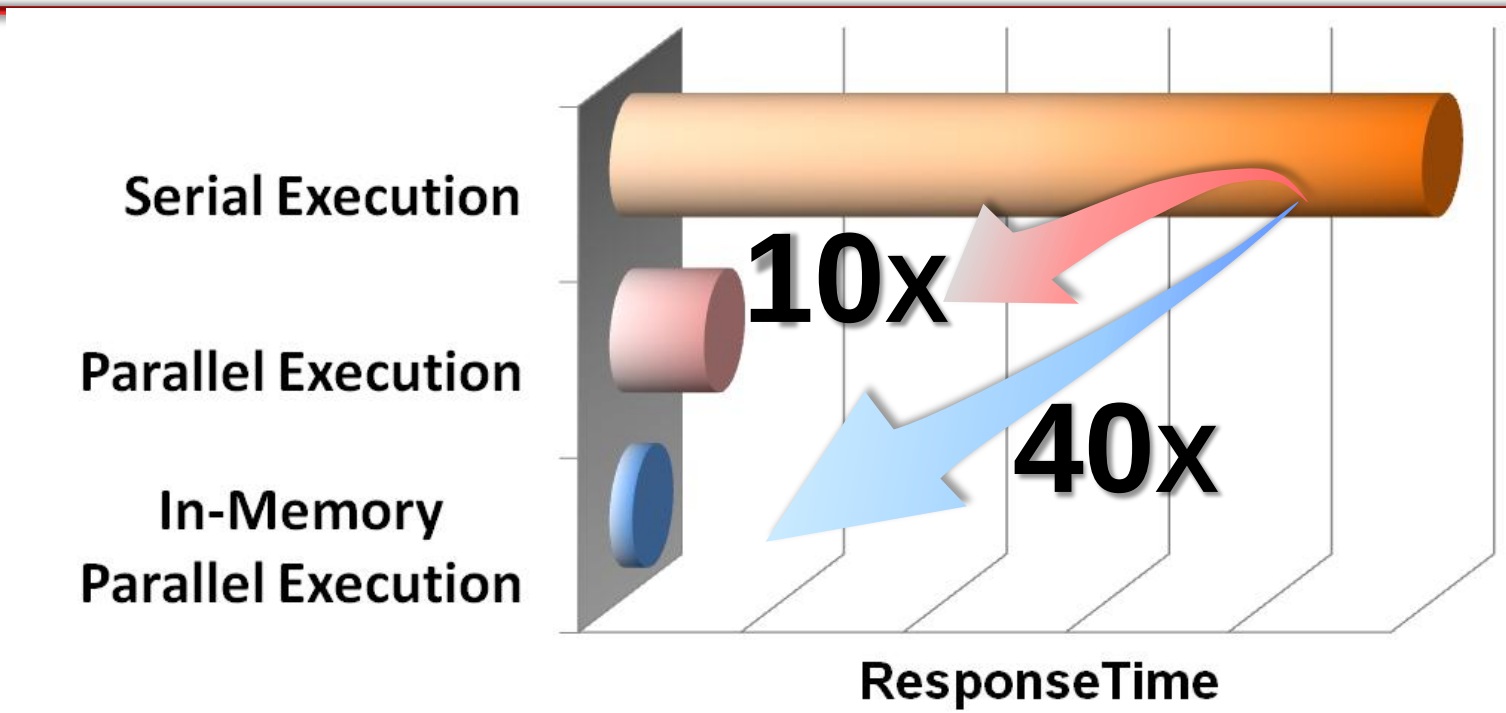


In-Memory Parallel Queryの効果

検証結果（レスポンスタイム）

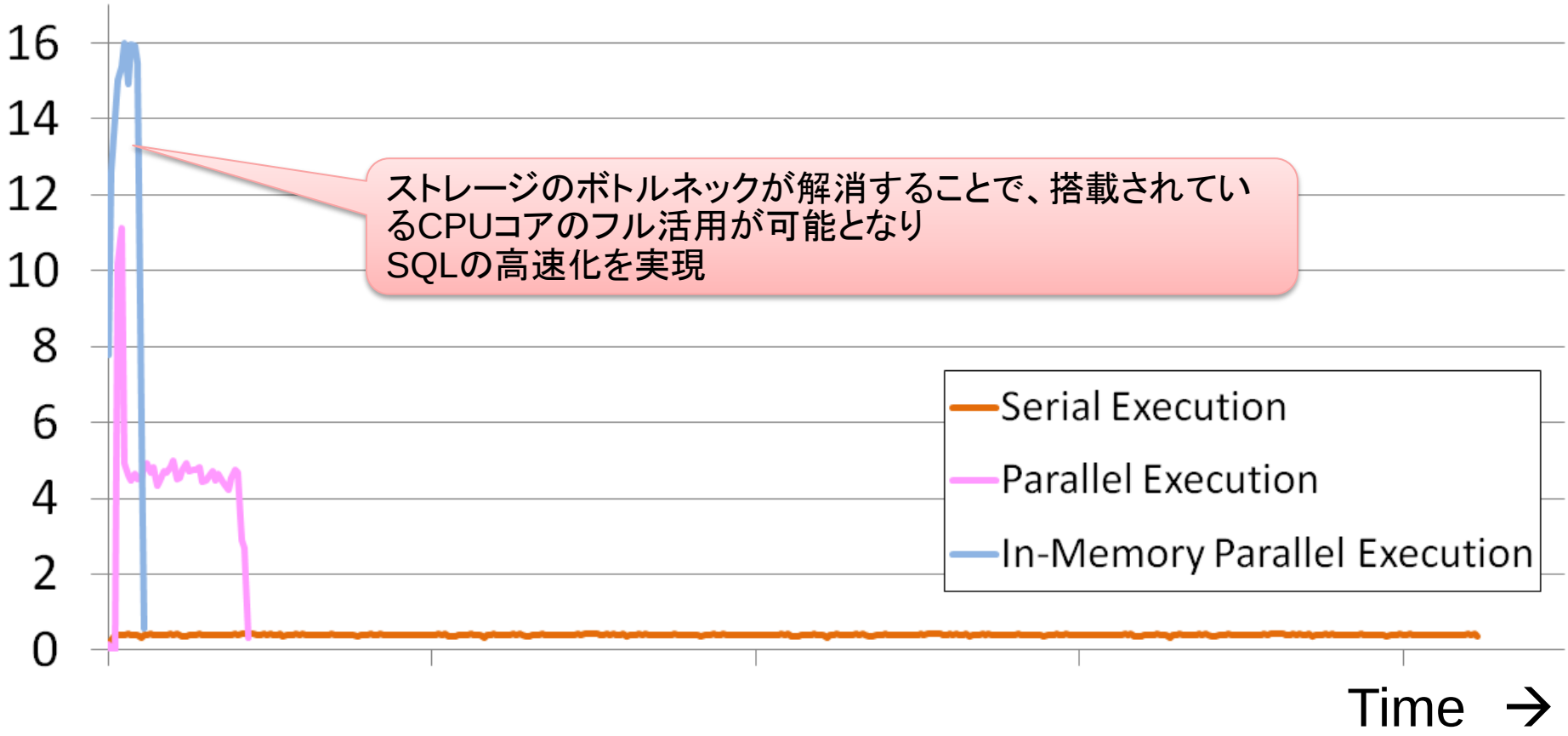
6000万件のデータの集計処理を

- Parallel Executionを使用することで、約10倍高速
- In-Memory Parallel Executionを使用することで約40倍高速



In-Memory Parallel Queryの効果

検証結果（CPU使用率）



検証結果

Oracle Grid Centerでの検証結果

- パートナー様との共同検証センターである、Oracle Grid Centerでは、In-Memory Parallel Queryに関して様々な検証を実施
  - 新日鉄ソリューションズ株式会社様
    - 「Oracle Database 11g R2　Real Application Cluster上でのIn-Memory Parallel Queryによる効率的なリソース活用」

      http://www.oracle.co.jp/solutions/grid_center/nssol/pdf/wp-impq-gridcenter-nssol_v1.0.pdf
  - 日本電気株式会社様
    - 「Oracle Database 11g R2　In-Memory Parallel QueryによるNEC Express5800/スケーラブルHAサーバー上でのData Warehouseシステム全体の性能向上」

      http://www.nec.co.jp/middle/oracle/wp-impq-gridcenter-nec.pdf

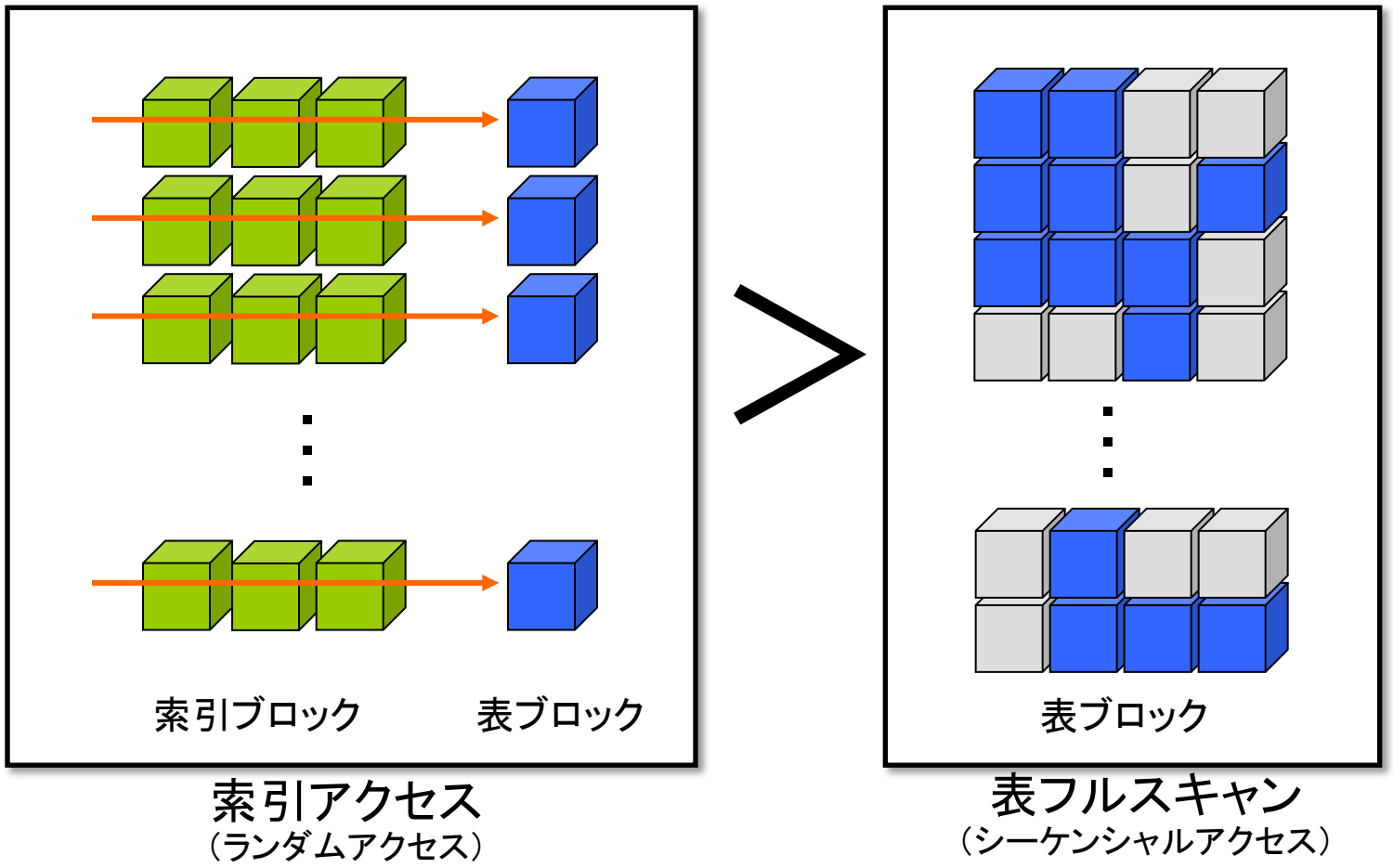
* 50音順

Agenda

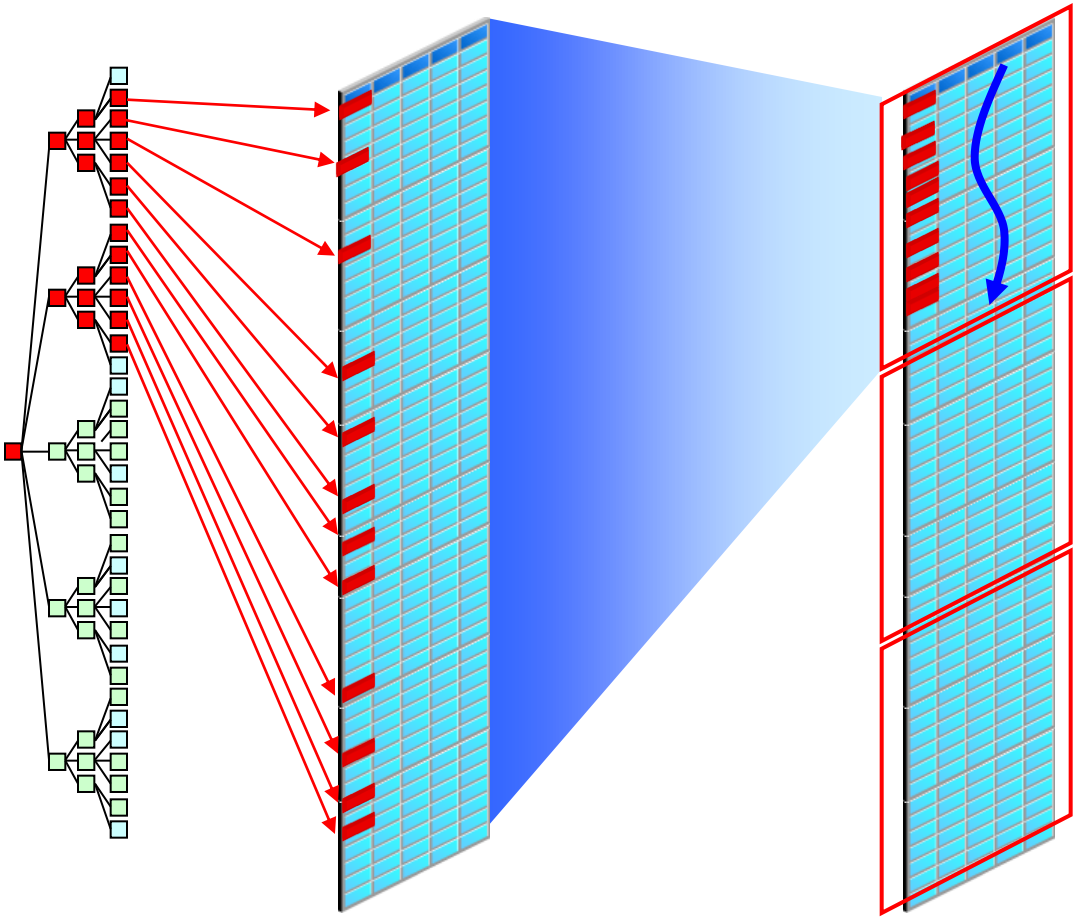
- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

アクセスコストが逆転する

1つのSQLが多くの行にアクセスする場合



高速化のアイデア



索引アクセス

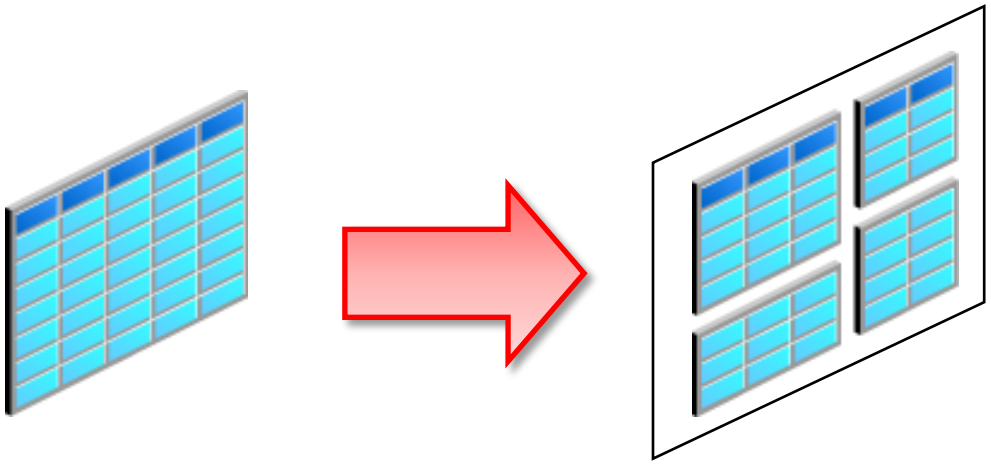
表フルスキャン

一定のルールに従って
表のデータを寄せる

パーティション表

パラレルクエリーとパーティション

- パーティションとは
  - 表や索引を**内部的に**分割する機能
    - 分割しても「一つの表」として扱われる
    - SQLなどの処理単位が扱うデータ量の削減
    - Oracle Database Enterprise Editionの有償オプション



パーティションのメリット

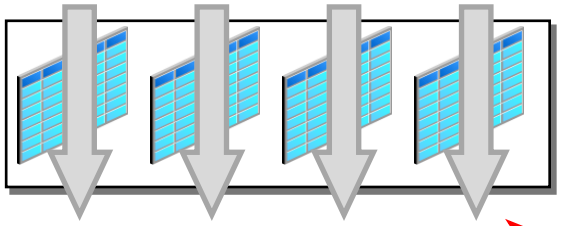
- SQL実行の高速化
  - パーティション単位のデータアクセス
- メンテナンス時間の短縮
  - パーティション単位でのメンテナンス
  - パーティション単位でのバックアップ取得
- 可用性の向上
  - 障害の局所化

分割してもアプリケーションSQLは変更不要

パーティション単位のデータアクセス

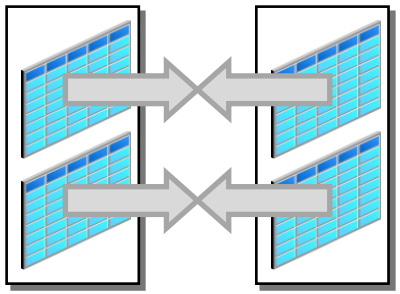
パーティション・パラレル処理

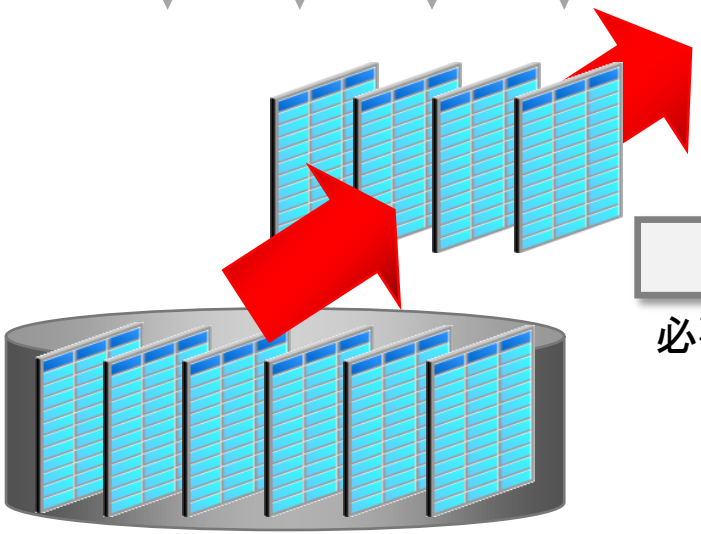
複数パーティションを並列処理



パーティション・ワイズ・ジョイン

パーティション単位でジョイン





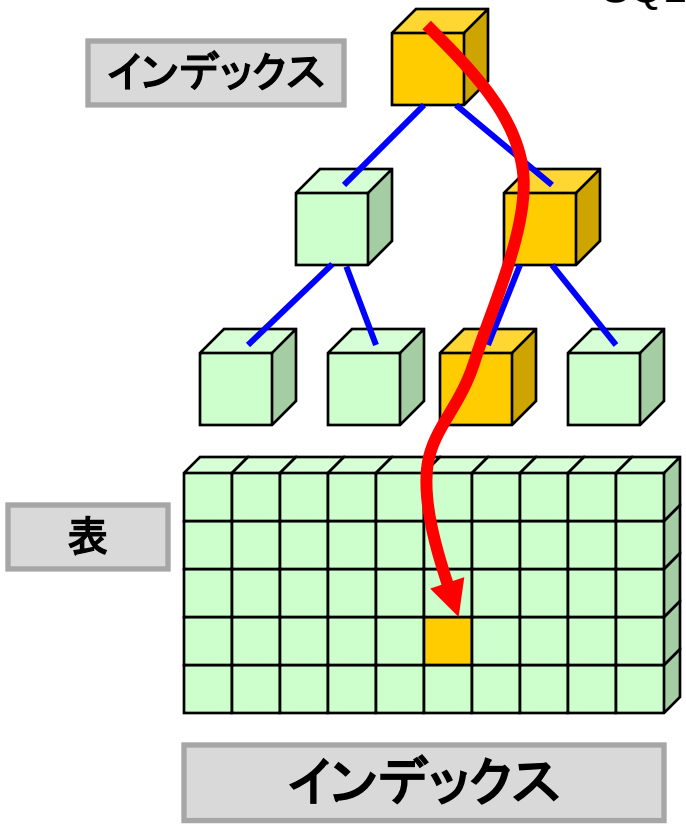
パーティション・プルーニング

必要なデータを持つパーティションにのみアクセス

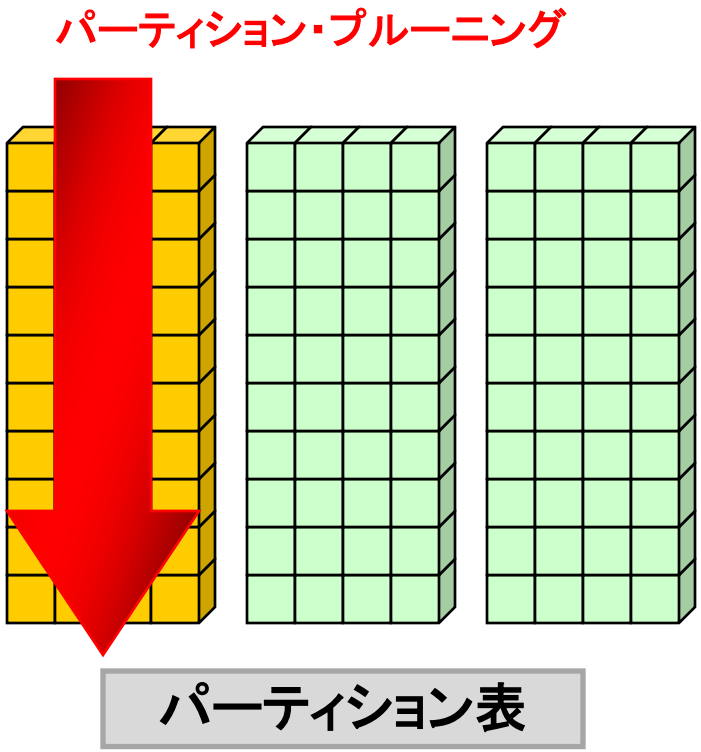
インデックスとパーティション・プルーニング

フルテーブルスキャンと比較して、どちらもアクセスブロック数を減らす効果がある

SQLチューニングの基本は「アクセスするブロック数を減らす」こと



取り出す行数が少ない場合に大きな効果

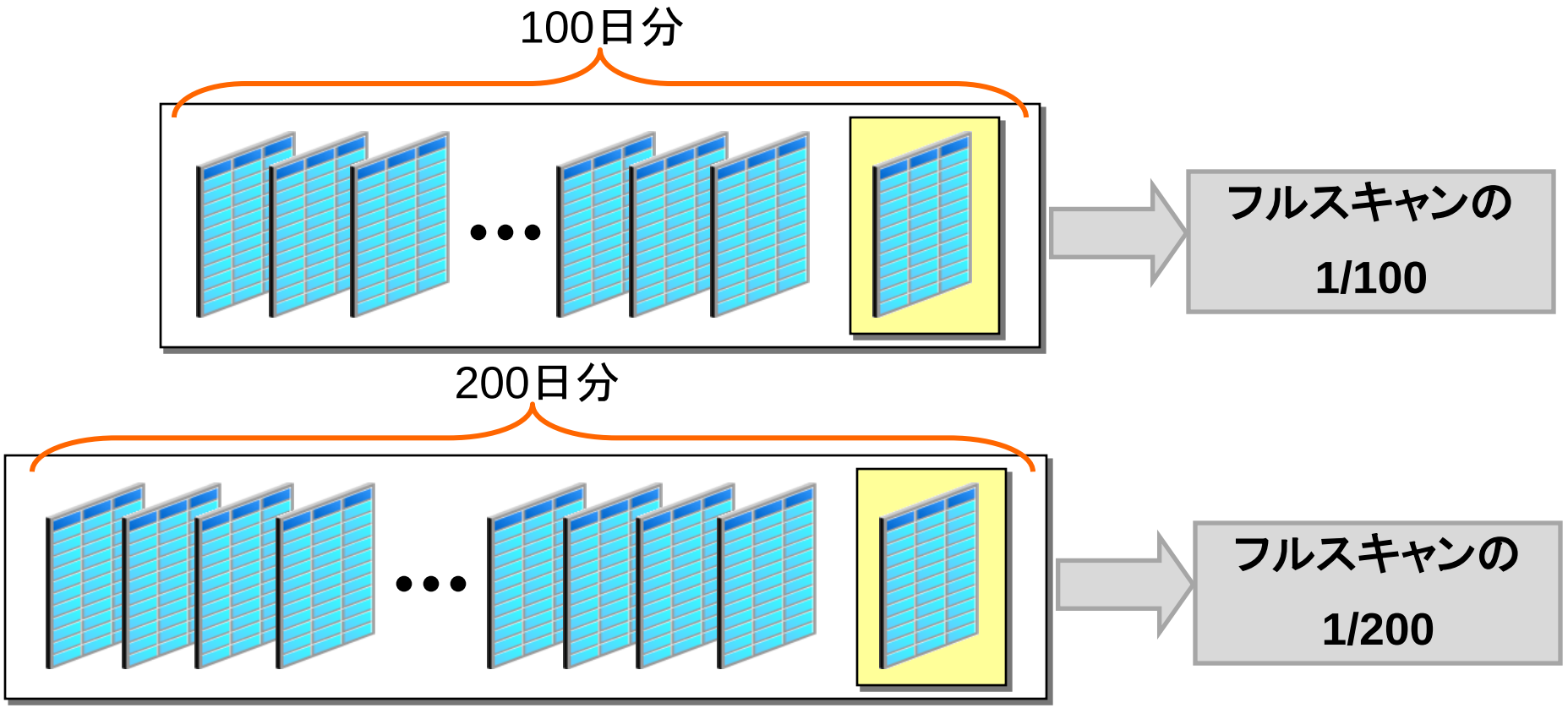


取り出す行数が多い場合に大きな効果

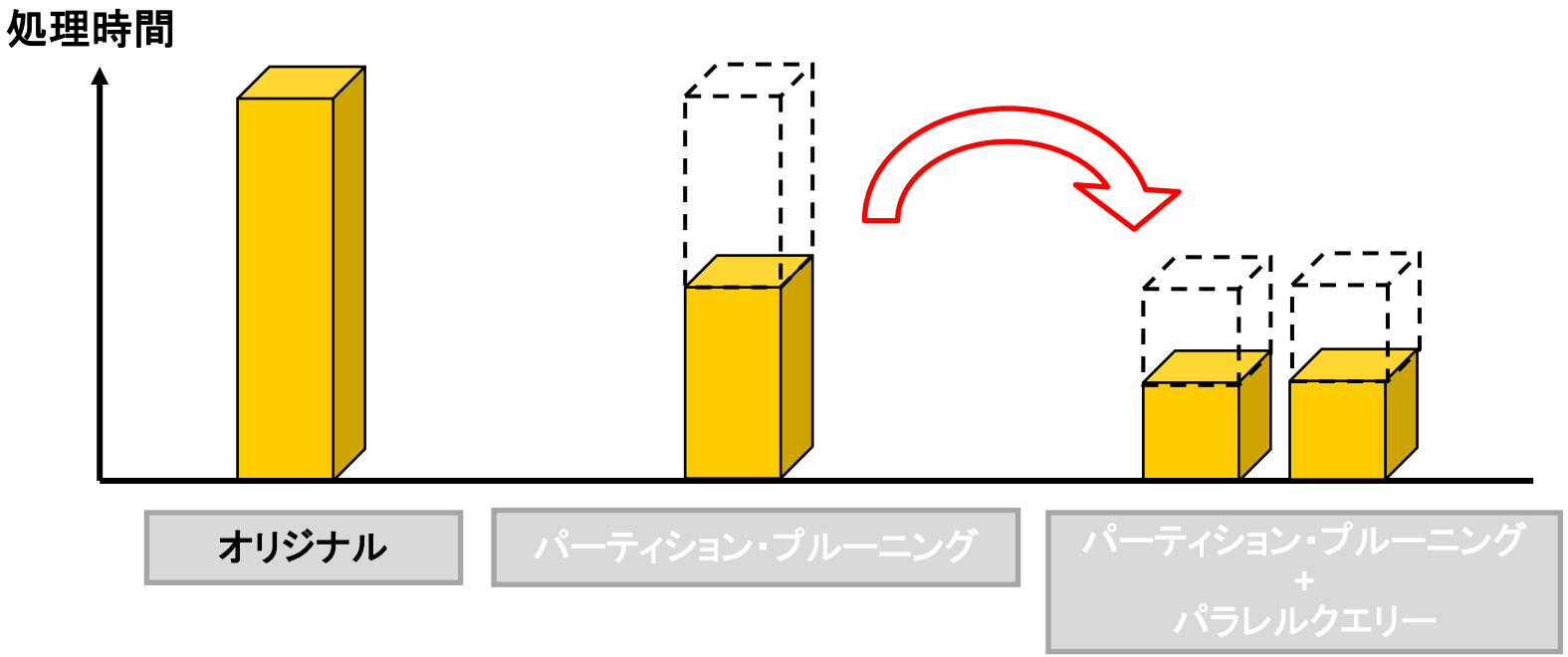
パーティション・プルーニング

1日単位でレンジ・パーティションした表

保持日数によらず、今日1日分の処理は1日分のパーティションへのアクセス



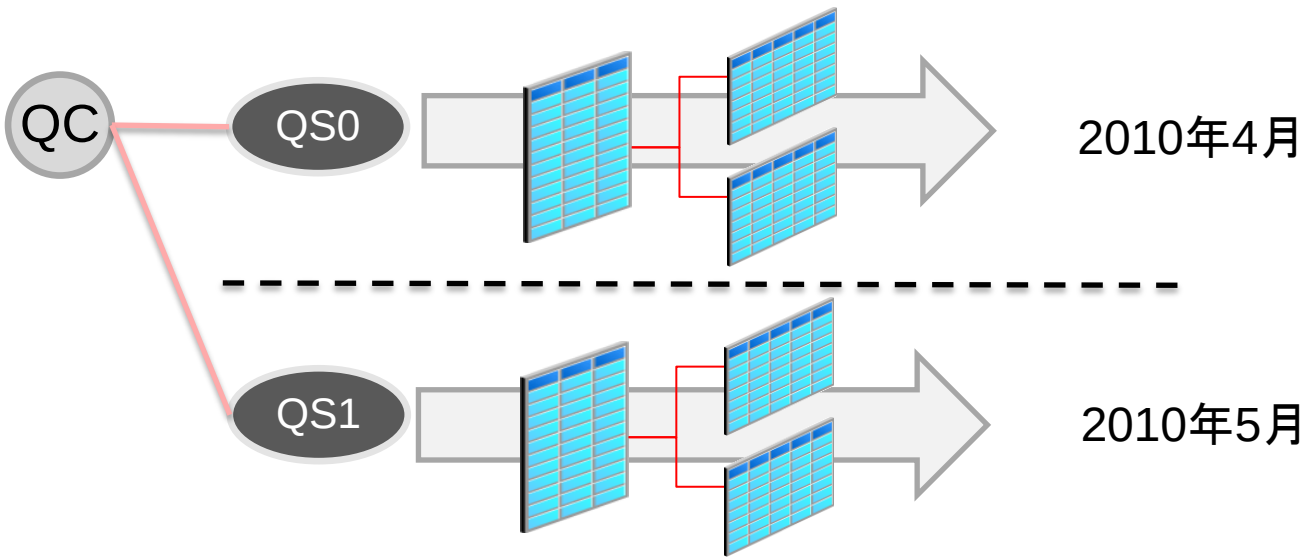
パーティション・プルーニングとパラレルクエリー



Oracleはパーティション・プルーニングしてから並列化する

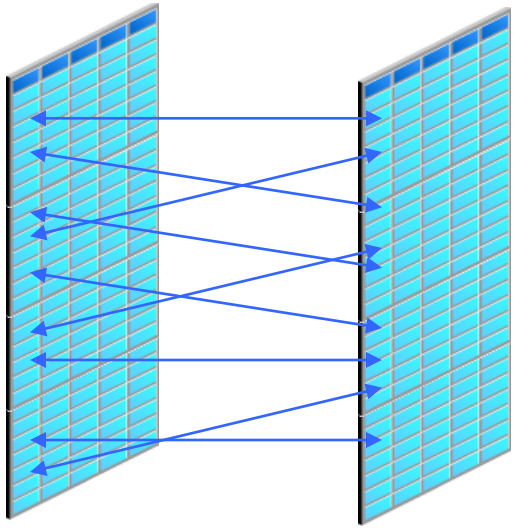
スレーブプロセスのデータ読み込みの方法
パーティション表の場合

- パーティション表に対してのアクセス方法
  - スレーブプロセスは各パーティションもしくはサブパーティション全体を処理
  - スキャン範囲の担当は動的には決定されない



大量の行を処理する

- パーティション・プルーニングによる絞込み
- 大量の行を持つ表をジョインする

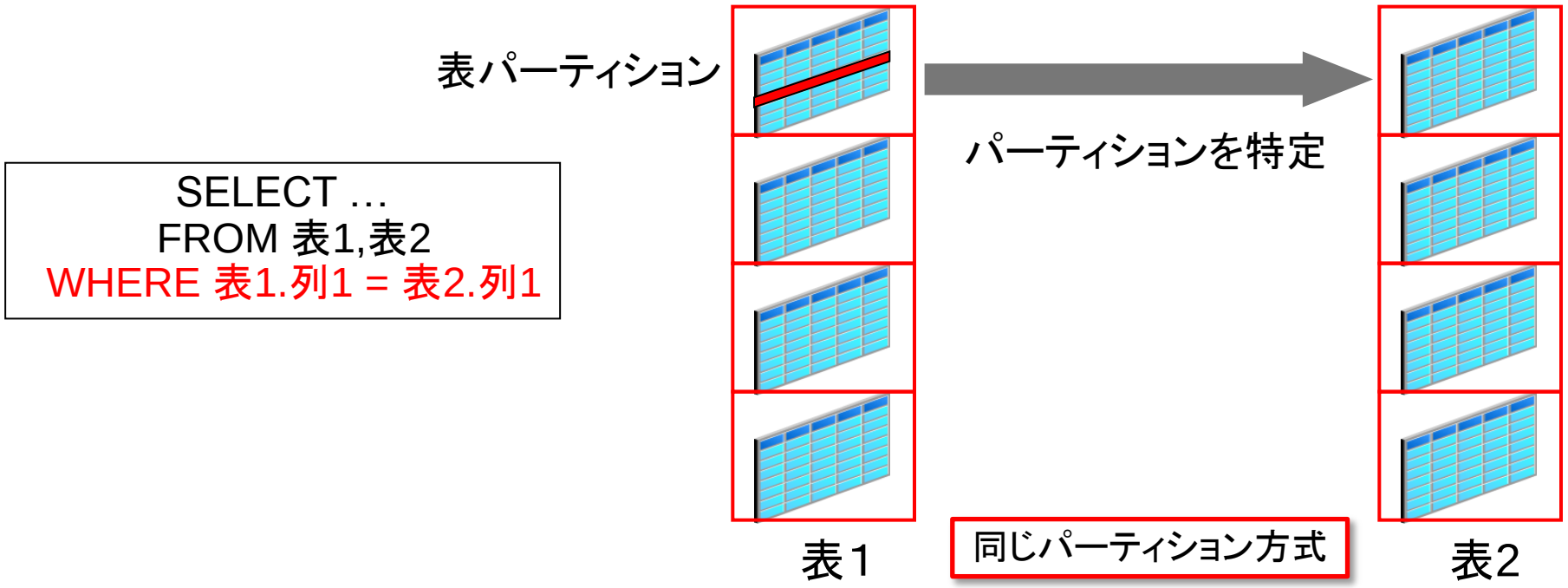


パーティション表同士のジョイン

同じパーティション方式、かつパーティション・キー同士

- パーティション・キー同士のジョインなら、結合対象の行がある対象パーティションを特定できる

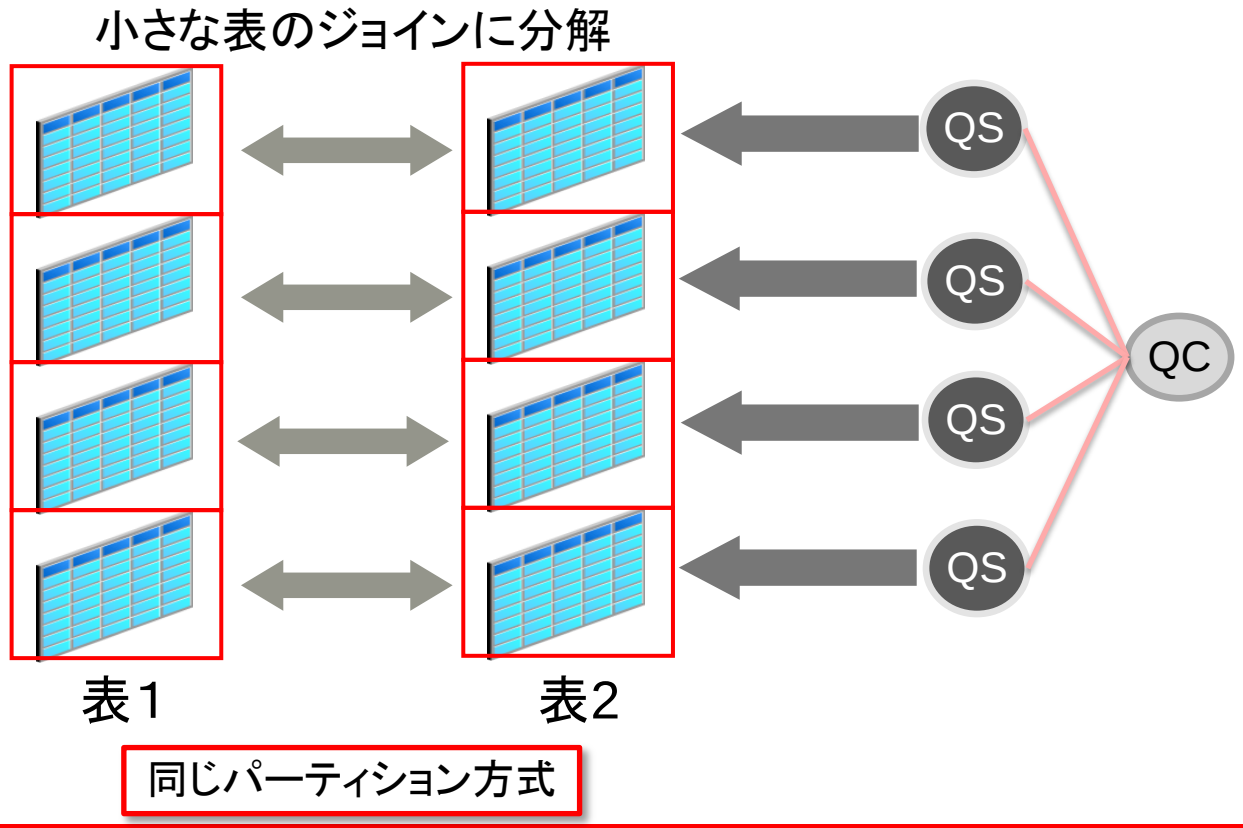
  → 小さな表のジョインに分解



フル・パーティション・ワイズ・ジョイン

同じパーティション方式、かつパーティション・キー同士

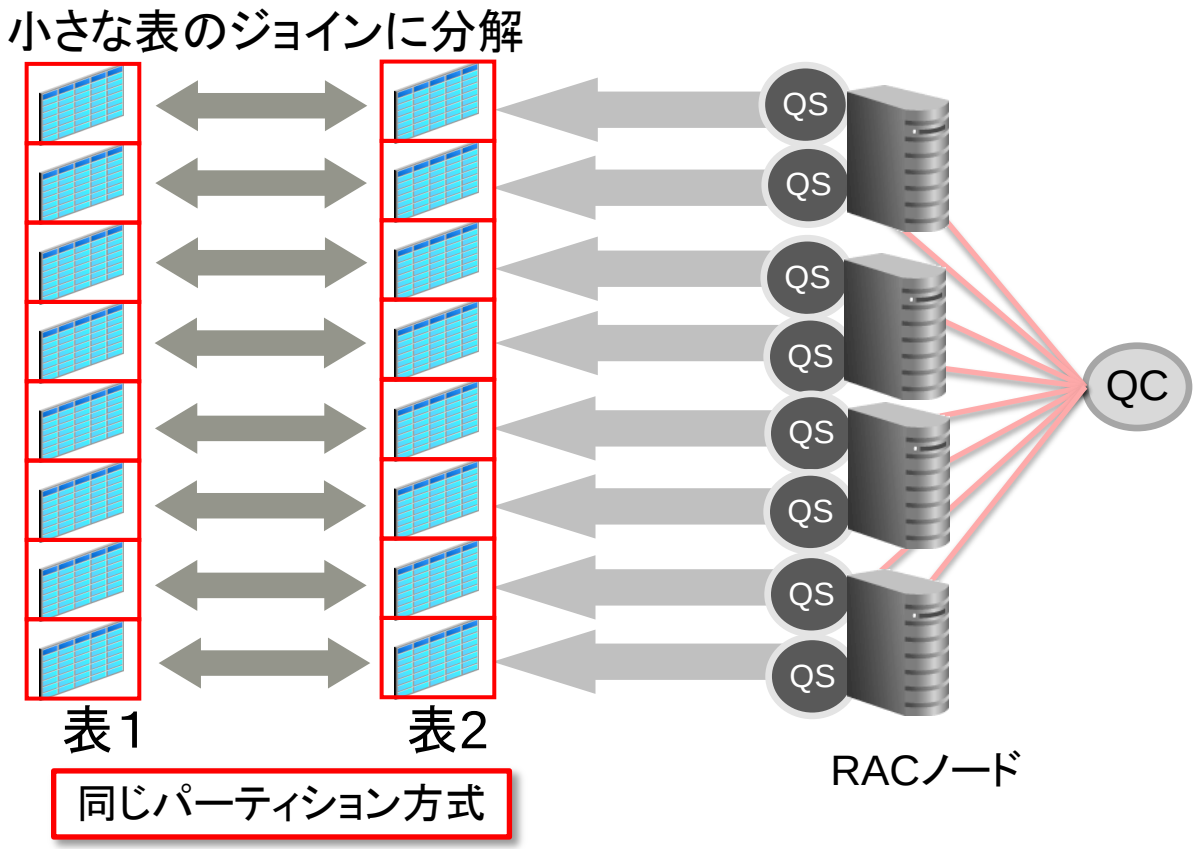
- フル・パーティション・ワイズ・ジョインとパラレルクエリー
  →各スレーブプロセスごとにジョインを並列実行実行可能



RACでフル・パーティション・ワイズ・ジョイン

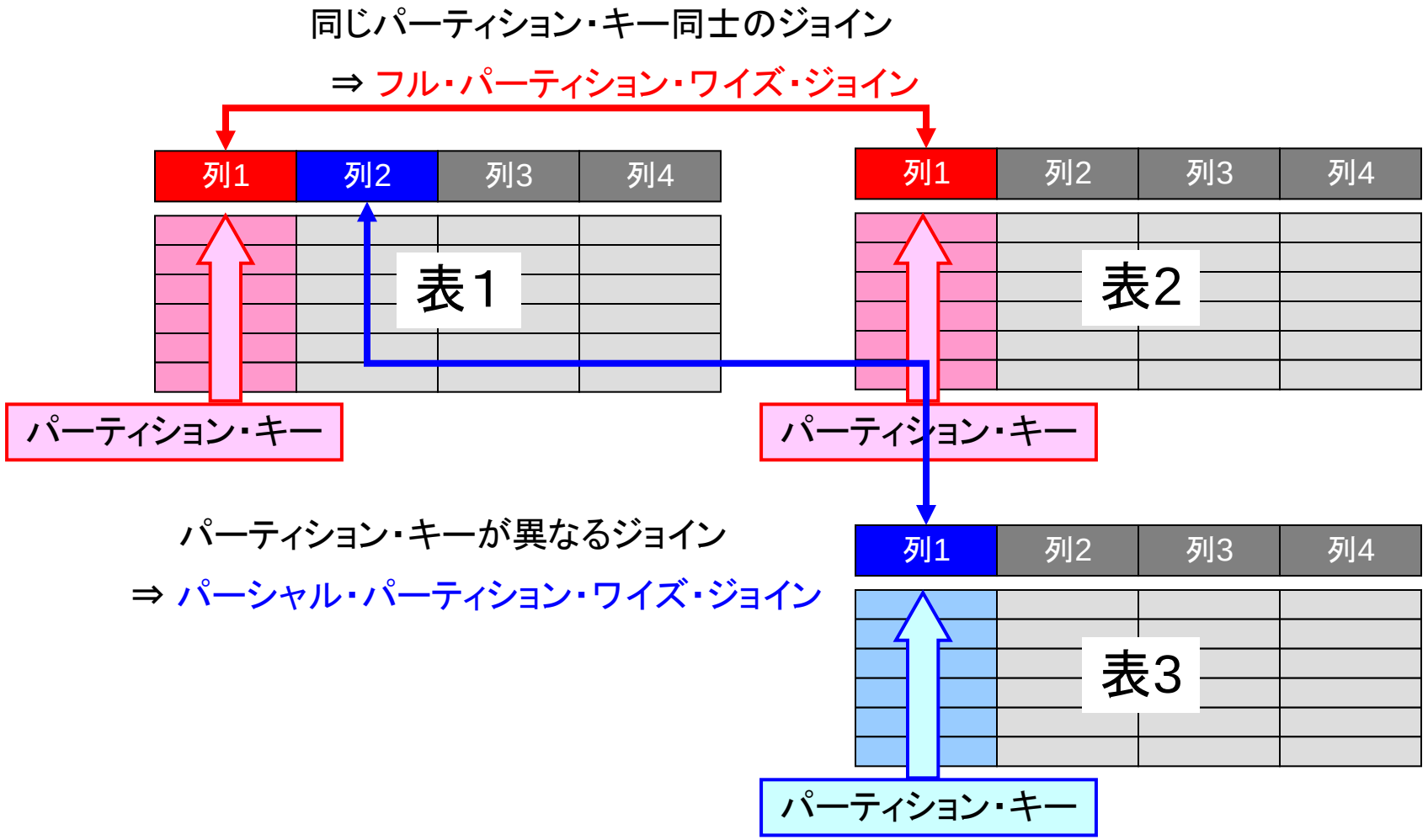
同じパーティション方式、かつパーティション・キー同士

- ノード毎にパーティションのジョインを割り当てる



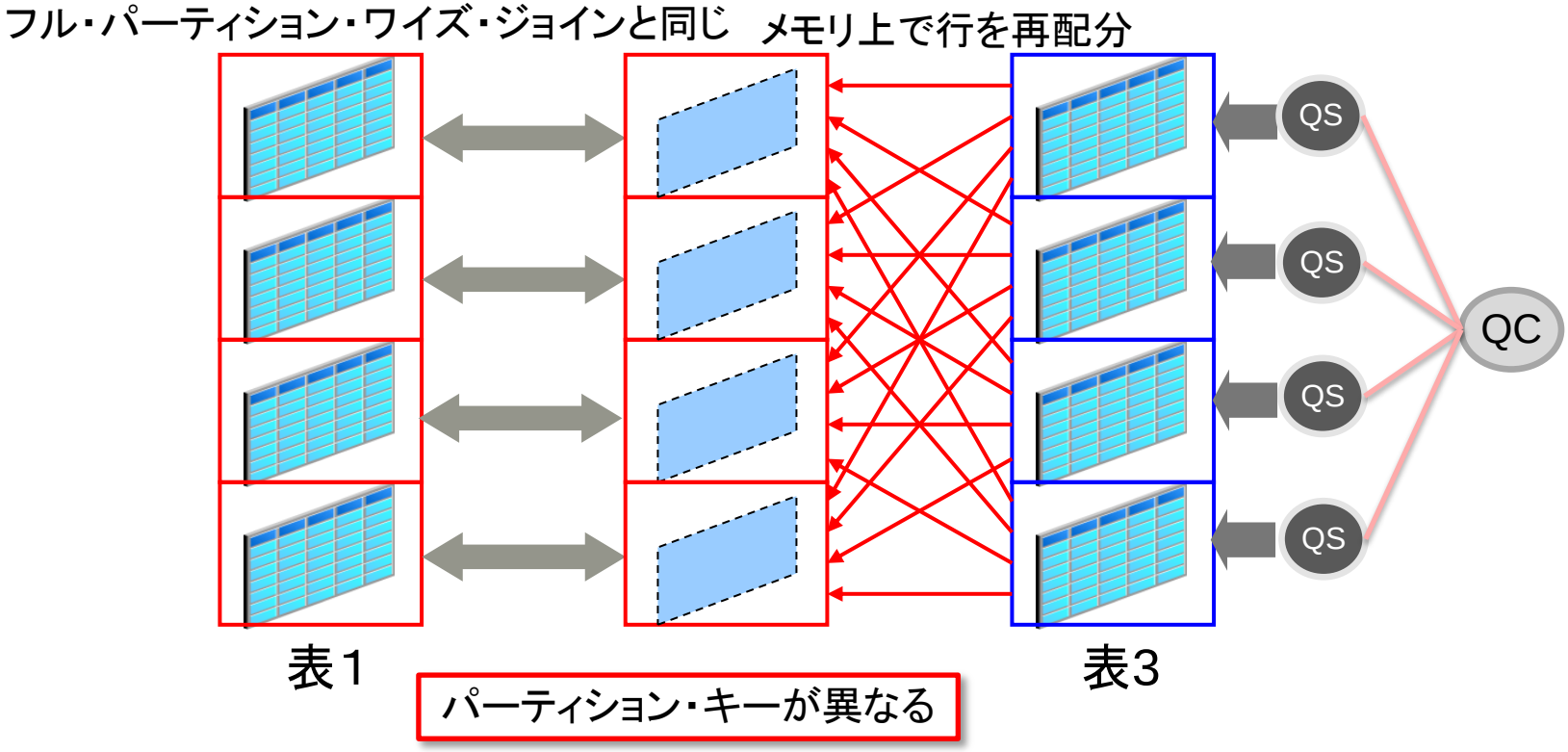
フル・パーティション・ワイズ・ジョインできない場合

同じパーティション・キー同士でジョイン可能とは限らない



パーシャル・パーティション・ワイズ・ジョイン

- パーシャル・パーティション・ワイズ・ジョイン
  →一方の表を再パーティション化する

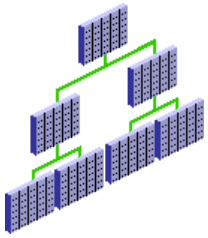


Agenda

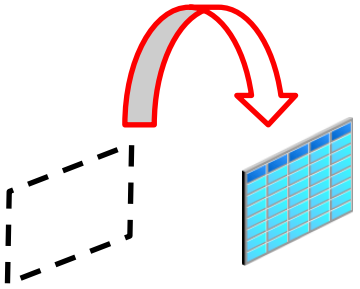
- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

パラレルDDL

- 非パーティション表に対して可能なパラレル処理は以下の3つ
  - create index
  - create table … as select
  - alter index … rebuild



- パーティション表に対して可能なパラレル処理は以下の4つ
  - create index
  - create table … as select
  - alter table … move/split/coalesce partition
  - alter index … rebuild/split partition



- メンテナンス作業の高速化を実現可能

DDLのパラレル化の方法

1. パラレルDDLを有効化させる

```
alter session enable parallel ddl;
```

2. DDL文をパラレル化させる
   - 各DDL文のパラレル化は次頁以降で紹介
   - create index / alter index … rebuild / alter index … rebuild partition の場合、パラレル属性として定義される

- 並列度の決定
  - alter session force parallel ddl parallel integerによって指定可能

```
alter session force parallel ddl parallel 6;
```

  - オブジェクトで定義する

パラレルDDLのポイント - その1

パラレルcreate index / alter index … rebuildのルール

- パラレル化の方法
  - パラレル句

```
create index ind_test on test(col1) parallel 4;
```

  - ALTER SESSION FORCE PARALLEL DDL文によってパラレル化

```
alter session force parallel ddl parallel 6;
create index ind_test on test(col1) ;
```

  - 表スキャン操作は対応するcreate / rebuild 処理と同じ並列度
  - 並列度が指定されていない場合、CPU数に基づきパラレル度が自動で調整

パラレルDDLのポイント - その2

パラレルMOVE/SPLIT PARTITIONのルール

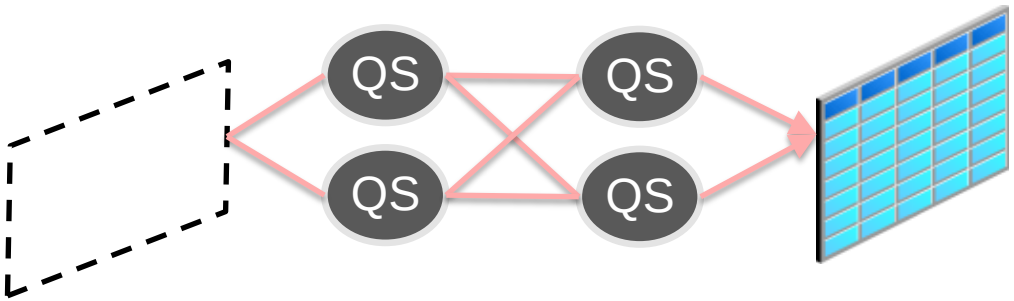
- パラレルMOVE PARTITION / SPLIT PARTITIONのルール
  - パラレル句

```
create index ind_test on test(col1) parallel 4;
```

  - ALTER SESSION FORCE PARALLEL DDL文によってパラレル化

```
alter session force parallel ddl parallel 6;
alter table … move partition partition_name …
```

  - スキャン操作は対応するMOVE / SPLIT操作と同じ並列度
  - 並列度が指定されていない場合、CPU数に基づきパラレル度が自動で調整



パラレルDDLのポイント - その3

create table … as selectのルール

- create 部分
  - パラレル句

```
create table tmp_test parallel 4 as select * from test;
```

  - ALTER SESSION FORCE PARALLEL DDL文によってパラレル化

```
alter session force parallel ddl parallel 6;
create table tmp_test as select * from test;
```

  - 全表スキャン/複数パーティション及びindex range scanで実行される場合、select部分もパラレル化される

パラレルDDLのポイント - その3

create table … as selectのルール

- select部分
  - create部分にparallel句が指定されている

```
create table tmp_test parallel 4 as select * from test;
```

  - select部分にパラレルヒントが含まれる

```
create table tmp_test as select /*+ parallel (test,4) */ * from test;
```

  - 参照する表にパラレル属性が定義されている

```
alter table test parallel 4;
create table tmp_test as select * from test;
```

  - create 操作がパラレル化される場合、select操作もパラレル化される
    - ただし、以下の場合はパラレル化されない
      - select文にNO_PARALLELヒント

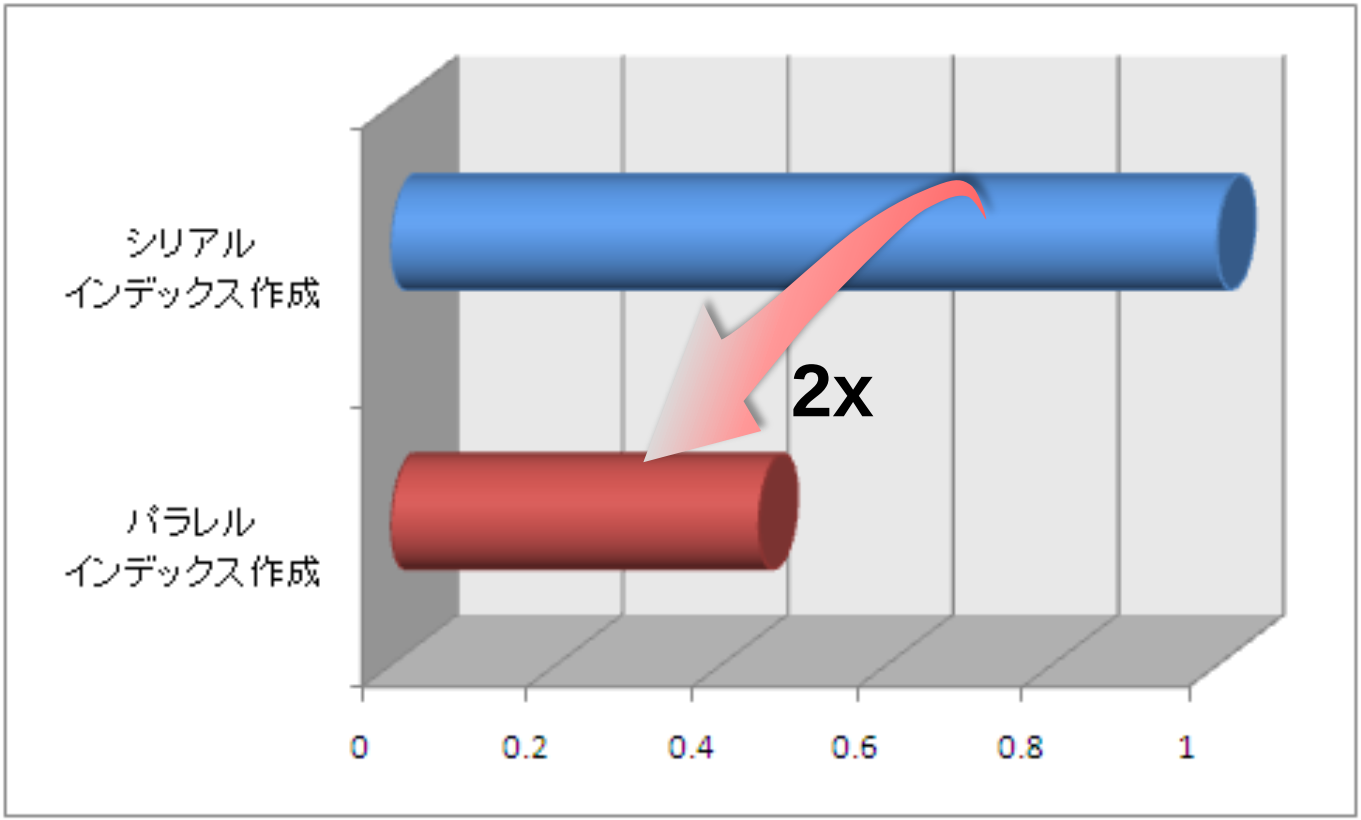
```
create table tmp_test parallel 4 as select /*+ no_parallel */ * from test;
```

      - 非パーティション表の索引がスキャンされる

検証結果

パラレルDDLによるインデックス作成の高速化

- パラレルDDLを利用することで、煩わしいメンテナンス作業も高速化可能



パラレルDML

- パラレルDML（PARALLEL INSERT/UPDATE/DELETE およびMERGE）
- 大規模オブジェクトにアクセスするDWH/DSS環境に有効
- パラレル設定のオーバーヘッドが生じるため、OLTP系システムには向かない
  - ただし、OLTPシステムで実施されるバッチ処理には効果的

パラレルDMLの設定方法

- DML実行前に、パラレルDMLを有効化する

```
alter session enable parallel dml;
```

  - 上記SQLを実行しない場合、DML文にPARALLELヒントを設定しても、パラレル化されない
  - ただし、DML文に問い合わせ処理が入っている場合、その部分のみはパラレル化される
- 並列度の決定
  - 以下の優先順位で決定する
    - DML文のパラレル・ヒントに指定されている値
    - alter session enable parallel dml parallel文で指定した値
    - 表作成時に指定したパラレル度

update/merge/deleteのルール

- パラレル化されるのは以下のいずれかの場合
  - alter session enable parallel dml文が発行されている
  - 更新/削除される表の定義でパラレル句を指定されている

```
alter table test parallel 4;
update test set col2=100 where col1 between 100 and 500;
```

  - Update/merge/delete文でパラレル・ヒントを有効化する

```
update /*+ parallel */ test set col2=100 where col1 between 100 and 500;
```

Insert … selectのルール

- 検索表と挿入表それぞれのアクセスに対してパラレル度を指定可能(パラレルDMLの有効化が前提)
  - 検索表
    - 文でのSELECTパラレル・ヒントの指定
    - 選択対象表の定義でのパラレル句の指定
  - 挿入表
    - 文でのINSERTパラレル・ヒントの指定
    - 挿入対象表の定義でのパラレル句の指定

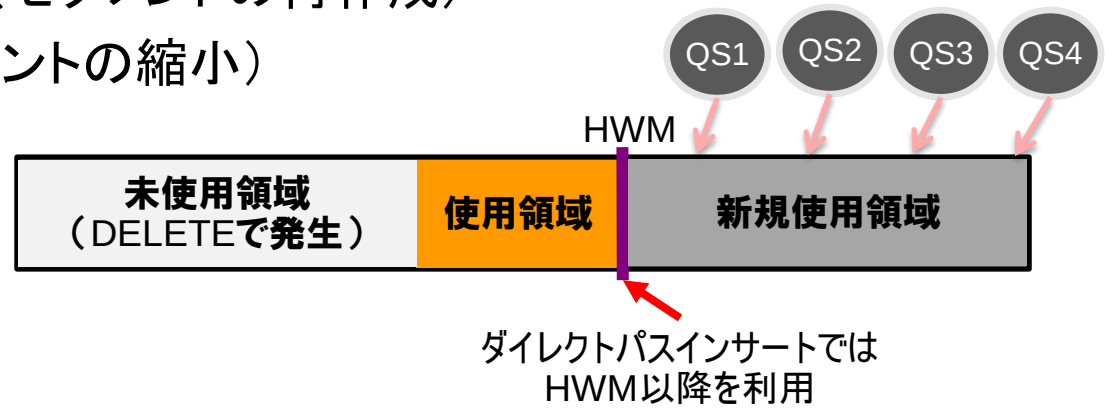
パラレルDMLの高速化

- 以下の方法を用いることで、パラレルDMLの高速化が可能
  - /*+ append */ ヒント句を用いる
    - キャッシュをバイパスして、直接データファイルに書き込み
    - ダイレクトパスインサート
  - /*+ nologging */ ヒント句を用いる
    - REDO生成量を抑制する

ダイレクトパスインサートの領域確保

- ダイレクトパスインサートではHigh Water Mark（HWM）以降からデータの書き込みが行われる
  - ダイレクト・ロード INSERT（INSERT /*+ APPEND */ INTO … SELECT …;)
  - パラレル INSERT
  - CREATE TABLE <table_name> AS SELECT;

- HWMを引き下げるためには、以下の処理を行う
  - Alter table … move（セグメントの再作成）
  - Shrink space（セグメントの縮小）



Agenda

- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

Datapumpとは

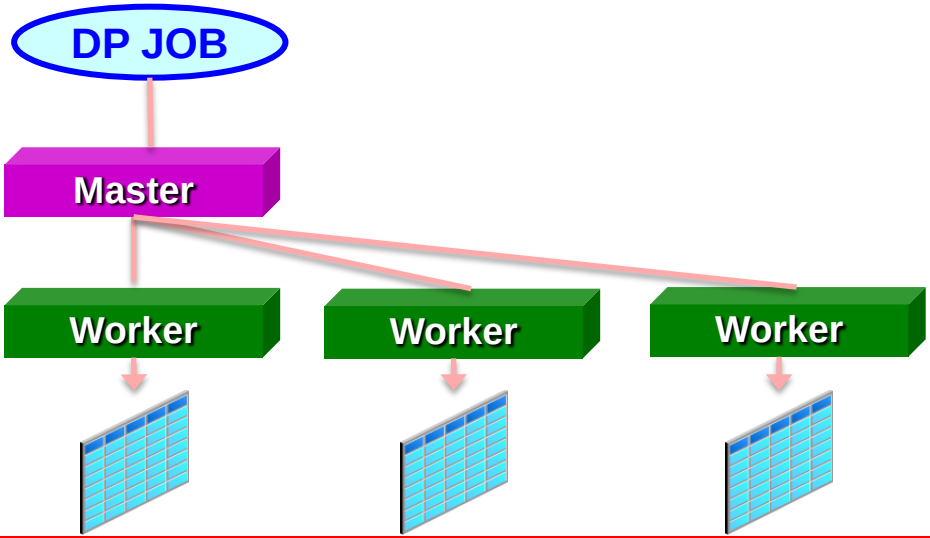
- Datapumpとは
  - Oracle Database 10g 以降で利用可能なユーティリティーツール
  - Oracle Database 9i までのexp/impにさらなる付加機能を持つ新たな機能

- 特徴
  - データおよびメタデータの高速なロード、アンロード
  - Exp/impに比べて、数倍高速
  - Exp/imp と同等の機能と、さらなる付加機能
    - パラレル処理、外部表など

Datapumpのパラレル化

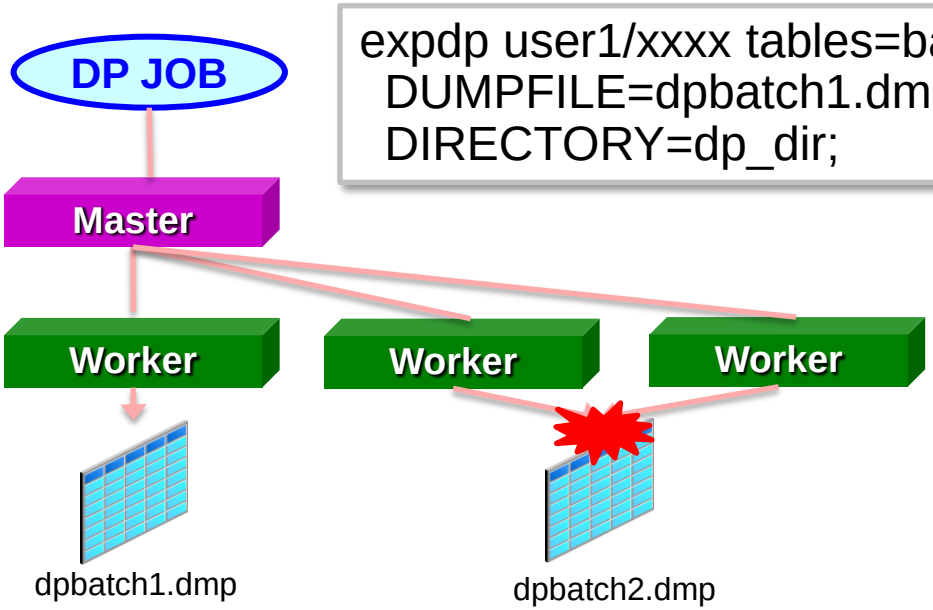
- DatapumpのPARALLELオプションで使用するWORKERプロセスの数を指定する
  - マスター制御プロセスは加算されない
- 実行例

```
expdp user1/xxxx tables=batch_tbl parallel=3
  DUMPFILE=dpbatch%U.dmp DIRECTORY=dp_dir;
```



Datapumpパラレル化のポイント -その1

- PARALLEL句に指定する値は、ダンプ・ファイル・セット内のファイル数以下にする、もしくはダンプファイル指定に置換変数を指定する必要がある
  - Workerプロセスが1つのダンプ・ファイルに対して排他的アクセスを行うため




```
expdp user1/xxxx tables=batch_tbl parallel=3
  DUMPFILE=dpbatch1.dmp,dpbatch2.dmp
  DIRECTORY=dp_dir;
```

ファイルの競合により一部のWorkerプロセスがアイドル状態になるため、パフォーマンスの向上が見込みにくい

ファイル・セットの値を置換変数化するなどして対処

Datapumpパラレル化のポイント -その2

- 11g R1 までの制限事項:
  - DataPump ジョブを実行できるのは、RAC 環境の場合でも 1 インスタンスのみ
  - Workerプロセスが起動されるのはジョブが実行されているインスタンス上のみ
- Oracle Database 11g R2より、DataPump ジョブを RAC 環境の複数のインスタンスで同時に実行することが可能
  - 並列実行により、より短時間で処理が完了
  - ワーカープロセスを複数ノードで起動
  - Cluster/Service_name パラメータで制御
    - **Oracle Database Enterprise Edition** で使用可能

DatapumpのRAC対応

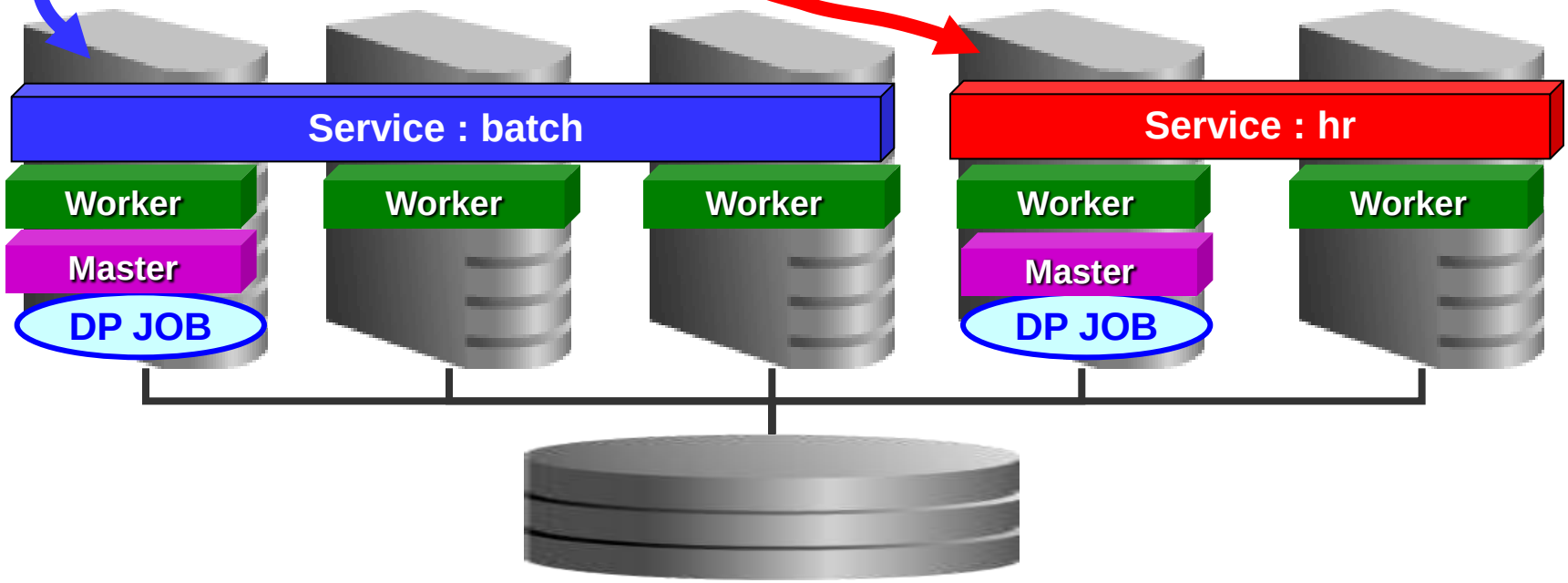
- CLUSTERパラメータ（デフォルト：Y）
  - RACのリソースを使用できるか、Workerプロセスを他のRACインスタンス上でも開始できるかどうかを指定
  - Datapumpジョブが大きい（大きな表にアクセスする）場合に効果的
  - 特定のRACサービスを指定したい場合、以下のSERVICE_NAMEパラメータも合わせて指定する

- SERVICE_NAMEパラメータ
  - Workerプロセスが起動するノード（RACサービス）を指定する
  - CLUSTER=Yとともに使用することが可能
  - CLUSTER=Nの場合、このパラメータは無視される

複数ノードでの DataPump 並列実行

```
[oracle@node1]$ expdp user1/xxxx tables=batch_tbl parallel=3
  service_name='batch' DUMPFILE=dpbatch%U.dmp DIRECTORY=dp_dir
```

```
[oracle@node4]$ expdp user2/xxxx tables=hr_tbl parallel=2
  service_name='hr' DUMPFILE=dphr%U.dmp DIRECTORY=dp_dir
```



その他Datapumpのパラレル化についてのポイント

- RAC環境でDatapumpを実行する場合、ディレクトリ・オブジェクトのパスをクラスタ・ファイルシステム上に配置する

- SERVICE_NAMEパラメータで指定できるのは、ジョブを開始するWorkerプロセスを起動するノードのみ
  - Masterプロセスはあくまでも接続されたノードで起動される

Agenda

- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

オプティマイザ統計情報とは

- Oracle Databaseにおける統計情報
  - 表統計情報
    - 行数、ブロック数、行の平均の長さ
  - 列統計情報
    - 列内の個別値数、列内のNULL数、データ配分（ヒストグラム）など
  - 索引統計情報
    - リーフ・ブロック数、クラスタ化係数
  - システム統計情報
    - I/Oパフォーマンス、CPUパフォーマンス
- オプティマイザはこれらの統計情報を元に、実行計画を作成
- 正確な統計情報を取得することは最適なパフォーマンスを得るために必要

オプティマイザ統計情報の取得方法

- Oracle Database 9i以降、DBMS_STATSパッケージの登場
  - Oracle Database 8i までは、analyzeコマンドによる取得
  - 課題：シリアル実行のため、遅い
    →速度向上のため、サンプリング率を減らす
    →正確な統計情報との差
- Oracle Database 10g以降、DBMS_STATSパッケージの使用を推奨
  - 以下の用途には、引き続きanalyze文を使用可能
    - VALIDATE / LIST CHAINED ROWS句を使用する場合
    - 空きリスト・ブロックの情報を収集する場合

（参考）DBMS_STATSパッケージの統計収集プロシージャ

プロシージャ名	収集対象
GATHER_INDES_STATS	索引統計
GATHER_TABLE_STATS	表、列及び索引の統計
GATHER_SCHEMA_STATS	スキーマ内の全てのオブジェクトの統計
GATHER_DICTIONARY_STATS	すべてのディクショナリ・オブジェクトの統計
GATHER_DATABASE_STATS	データベース内の全てのオブジェクトの統計

統計情報の取得のパラレル化

- DBMS_STATSプロシージャのDEGREE句で指定する
- 統計情報取得のパラレル化ができないオブジェクト
  - クラスタ索引
  - ドメイン索引
  - ビットマップ・ジョイン索引など

- 実行例

```
execute
 dbms_stats.gather_table_stats(‘test’,‘TEST_TBL’ degree => 4)
 ;
```

統計情報の取得のパラレル化

- Tips
  - DEGREE句は、DBMS_STATS.AUTO_DEGREEに設定することをお薦め
  - AUTO_DEGREEに指定することで、オブジェクトのサイズ及び初期化パラメータの設定に基づいて、Oracle Database側で適切な並列度を選択する

Agenda

- 最新CPUとデータベースシステム
- クエリーのパラレル化
  - パラレルクエリー
  - RACでのパラレルクエリー
  - パラレルとパーティション
- メンテナンス/データロードのパラレル化
- Datapumpのパラレル化
- 統計取得のパラレル化
- まとめ

まとめ

- CPUはマルチコア化、高速化
  - しかし、CPUを使いきれていないという現実
  - 特にシリアル処理では顕著に表れる
- Oracle Databaseでのパラレル処理
  - Oracle Database Enterprise Editionの標準機能
  - パラレルクエリー
  - パラレルDDL/DML
  - Datapumpのパラレル実行
  - 統計情報取得のパラレル化

Oracle Database Enterprise EditionによりマルチコアCPUの性能向上を享受可能!!

関連する初期化パラメータ

パラメータ名	デフォルト値	説明
PARALLEL_MAX_SERVERS	CPU_COUNT、PARALLEL_THREADS_PER_CPUおよびPGA_AGGREGATE_TARGETの値から導出	1インスタンスで起動できるQSプロセスの最大数
PARALLEL_MIN_SERVERS	0	インスタンス起動時に作成されるQSの数
PARALLEL_DEGREE_POLICY	MANUAL	Oracle Database 11g R2のパラレル実行に関する新機能の制御
PARALLEL_EXECUTION_MESSAGE_SIZE	16384	パラレル実行時に使用されるメッセージサイズ
PARALLEL_MIN_PERCENT	0	パラレル実行時のQS要求数の最小割合
PARALLEL_MIM_TIME_THRESHOLD	AUTO（10秒）	自動並列度によるパラレル実行対象判別の閾値
PARALLEL_THREADS_PER_CPU	2	パラレル実行中にCPUが処理できるQSの数

参考資料

- 『Oracle Database パフォーマンス・チューニング・ガイド』

  http://download.oracle.com/docs/cd/E16338_01/server.112/b56312/toc.htm
- 『Oracle Database VLDBおよびパーティショニング・ガイド』

  http://download.oracle.com/docs/cd/E16338_01/server.112/b56316/toc.htm
- 『Oracle Database ユーティリティ』

  http://download.oracle.com/docs/cd/E16338_01/server.112/b56303/toc.htm
- 『Oracle Database SQL言語リファレンス』

  http://download.oracle.com/docs/cd/E16338_01/server.112/b56299/toc.htm
- 『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』

  http://download.oracle.com/docs/cd/E16338_01/appdev.112/b56262/toc.htm

SOFTWARE. HARDWARE. COMPLETE.

ORACLE


ORACLE®