

Oracle DBA & Developer Days 2011

日本オラクル、今年最大の技術トレーニングイベント

2011年11月9日(水)～11月11日(金) シェラトン都ホテル東京



ORACLE®

津島博士のパフォーマンス講座 「パフォーマンス問題はなぜ起きるのか」

日本オラクル株式会社 製品事業統括本部
ディレクター 津島 浩樹

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。

OracleとJavaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

Agenda

- 「津島博士のパフォーマンス講座」について
- パフォーマンス問題に対する悩み
- パフォーマンス問題はなぜ起きるか
- バッチ処理の問題点

「津島博士のパフォーマンス講座」について

oracletech.jpで連載しています (<http://oracletech.jp/products/tsushima/>)

- 第1回 パフォーマンス問題はなぜ起きるか (2011. 01. 19)
- 第2回 RAC時のバッチ処理について (2011. 02. 01)
- 第3回 Statspackから探る、パフォーマンス問題の原因特定方法 (2011. 02. 14)
- 第4回 パフォーマンスFAQ: 良く聞かれる疑問点にお答えします (2011. 03. 01)
- 第5回 オプティマイザとオプティマイザ統計の収集について (2011. 04. 05)
- 第6回 パフォーマンスの基礎である索引について (2011. 04. 19)
- 第7回 共有プールについて (2011. 06. 21)
- 第8回 断片化について (2011. 07. 20)
- 第9回 良いSQLについて (2011. 8. 30)
- 第10回 パーティションについて (2011. 09. 13)
- 第11回 良いSQLについて(2) (2011. 10. 26)



パフォーマンス問題に対する悩み

- なぜ起きるのか？
 - 事前にテストを行っているのに
 - テストの量が不足している
- どう解決するのか？
 - 何から調べるのか
 - 特定するのも難しい
 - 特定できても解決方法が

すべてを速くすることはできないことを忘れずに！

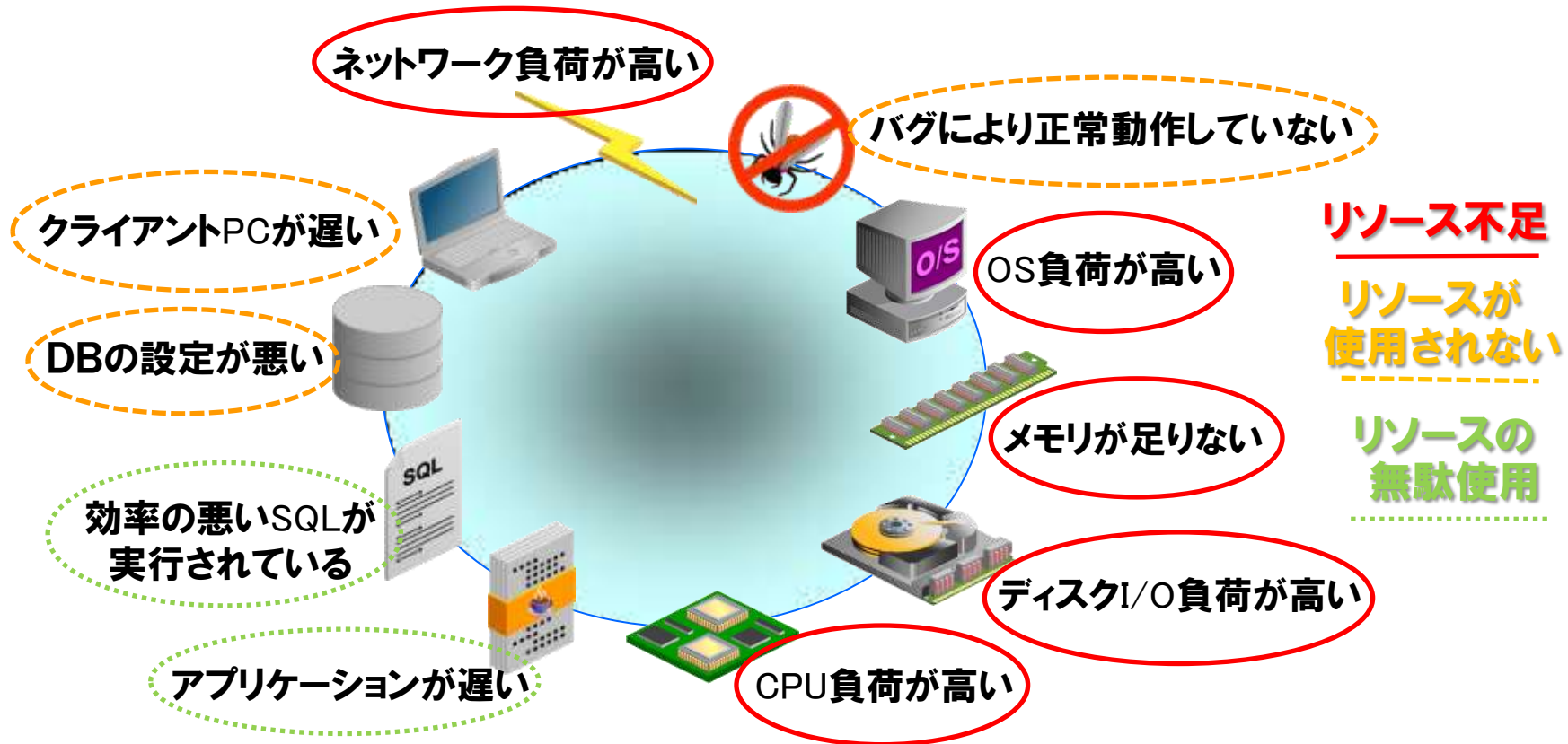
なぜ起きるのか？
どう解決するのか？



パフォーマンス問題はなぜ起きるか

パフォーマンス問題とは

- パフォーマンス問題は下記に示すとおり、さまざまな原因が考えられます



パフォーマンス問題はなぜ起きるか

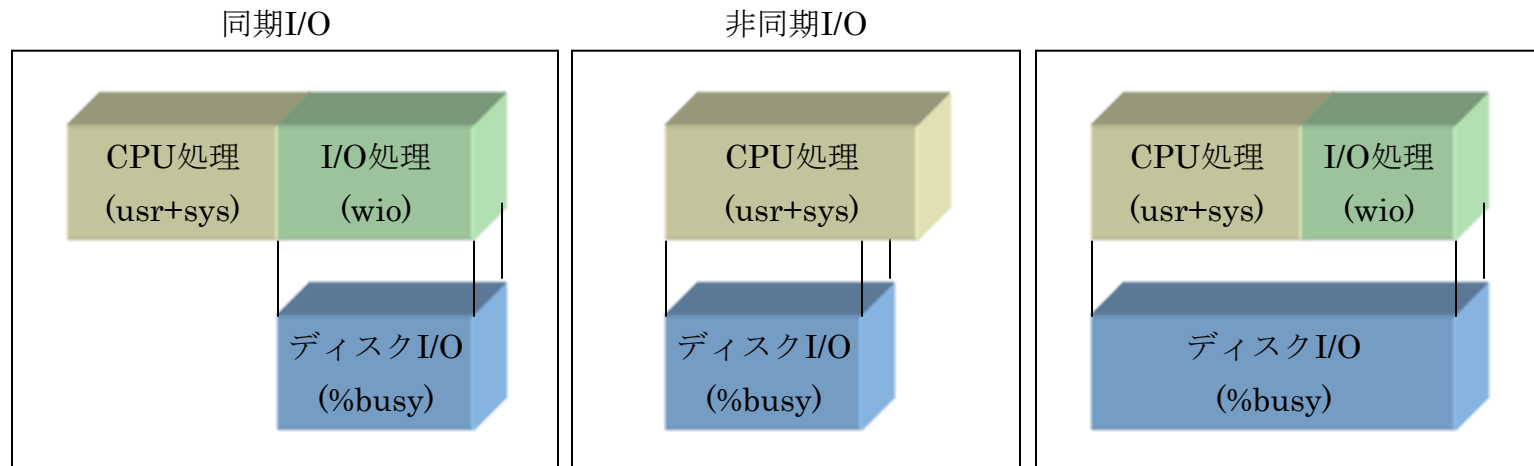
パフォーマンス問題とは

- リソース不足
 - すべてのH/Wリソースを効率よく使用できないと、そこがボトルネックになる
- リソースを使用できない
 - これはCPUリソースのことです
 - 待機が発生するから
 - 他のリソース不足でも起きる(I/O待ちなど)
 - ロック(エンキュー、ラッチ、mutexなど)待ち
 - 正常でも待機は発生することを忘れずに
 - I/O(読込み)待ちは必ず発生する
- リソースの無駄な使用
 - SQL文が悪い

パフォーマンス問題はなぜ起きるか

パフォーマンス問題とは

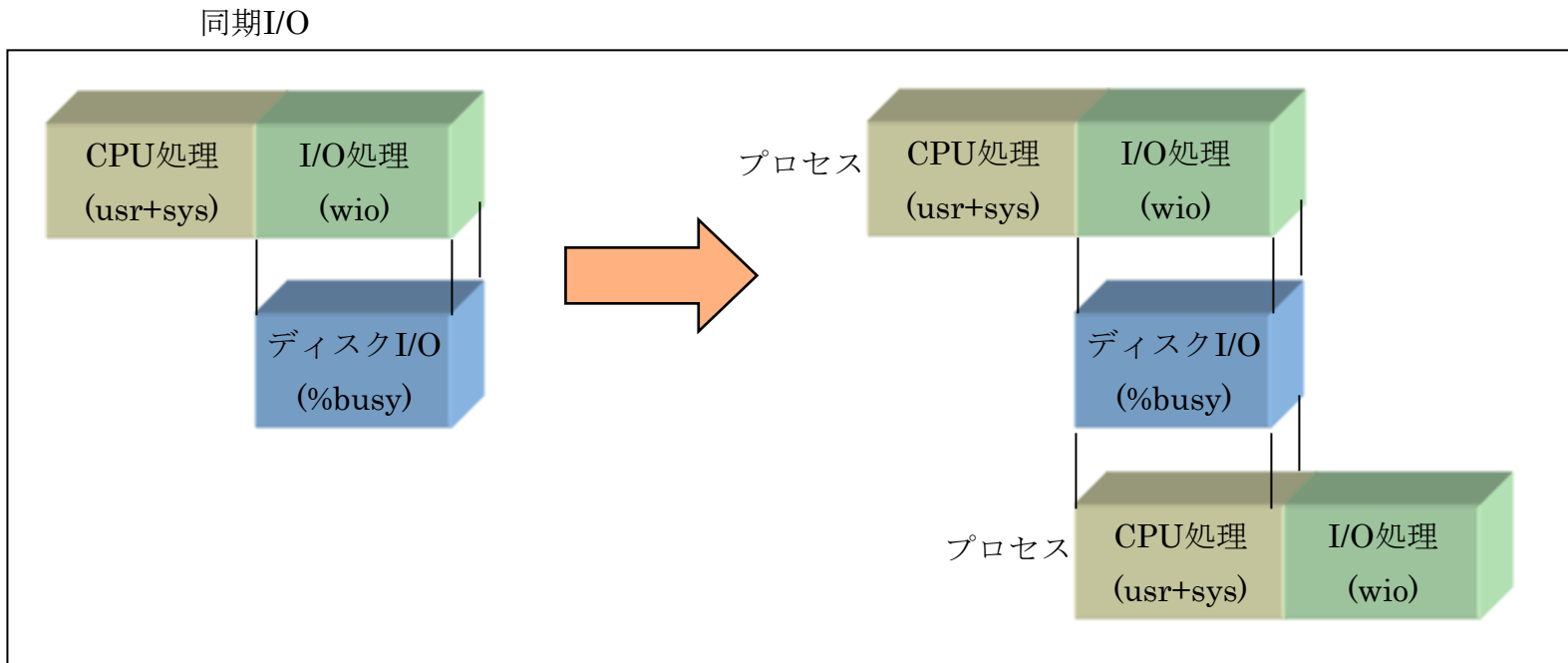
- I/O(読み込み)待ちは必ず発生する
 - I/Oは必ず発生する(キャッシュヒット率が100%でない限り)
 - 書込みは非同期ですが(DBWRが行う)、読み込みは同期なため(サーバープロセスが行う)
 - 非同期読み込み(フルスキャンの先読み、索引スキャンのData Block Prefetching)などはあるがCPUよりI/Oは遅い



パフォーマンス問題はなぜ起きるか

パフォーマンス問題とは

- 多重化することで使用率は向上する
 - CPUコア数より多い多重度でも
 - ディスク・ネックでないこと



パフォーマンス問題はなぜ起きるか

どんな時に発生するか

- ユーザ数の増加による
- データ量の増加による
- 長期運用による
- その他(そもそも...が悪い)
 - データベース設計が悪い(業務アプリの知識不足)
 - SQL文が悪い(SQLの知識不足)

パフォーマンス問題はなぜ起きるか

ユーザ数の増加による

- 競合が多発する(主に更新処理に対して)
 - セグメントのブロック
 - 同一ブロックにアクセスしてしまう可能性が高くなる
 - 行ロック(エンキュー)
 - 同一データにアクセスしてしまうから
 - エンキューは他にもあります
 - HW(エクステントの拡張が多い)
 - SQ(順序番号の獲得が多い)
 - など
 - メモリ上のラッチ(mutex)
 - 共有メモリは全てのプロセスからアクセスできる
 - メモリ上の整合性を保つ必要があるから

パフォーマンス問題はなぜ起きるか

ユーザ数の増加による(解決策)

- ブロックの競合
 - ブロック内の行数を少なくする
 - ブロック・サイズを小さくする
 - PCTFREEを大きく
 - 圧縮は逆効果
 - 同一ブロックにアクセスしないようにする
 - Freelist/FreelistGroupまたはASSM(デフォルト)
 - これはINSERTのみ
 - パーティション化

パフォーマンス問題はなぜ起きるか

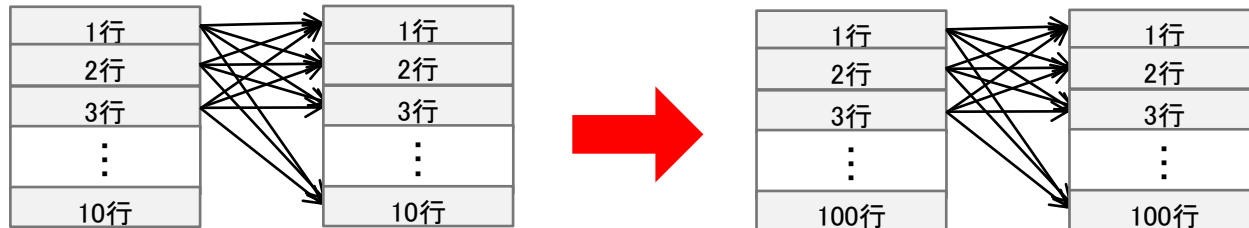
ユーザ数の増加による(解決策)

- 行ロックの競合
 - アプリケーションのロジック(又はデータベース設計)を検討
 - 同一行を更新しなくても良いようにする
 - SELECT ... FOR UPDATEを行っていないか
 - Oracleは共有ロックがないので競合は少ない
 - 頻繁に更新される列が同じテーブルになっていないか
- メモリ上のラッチ競合
 - ユーザ数を調整
 - 効率よい数(CPUコア数から)を自動的に計算しているので基本は調整できない

パフォーマンス問題はなぜ起きるか

データ量の増加による

- アクセス (CPU使用率、I/O) が増加する。
 - データ量と比例して増加するのか？
 - 結合時にはアクセスデータが更に増大 (ここが勘違いしている)
 - アクセスする件数が $10 \times 10 = 100$ だとすると
10倍になると $100 \times 100 = 10,000$ になる



- データ増加した割合のリソースを増強しただけでは性能は同じにならないのです
- データを絞り込んでアクセスする必要がある (索引などで)
- 索引 (Bツリー) の階層が増える
 - 階層はキーサイズと行数によるので

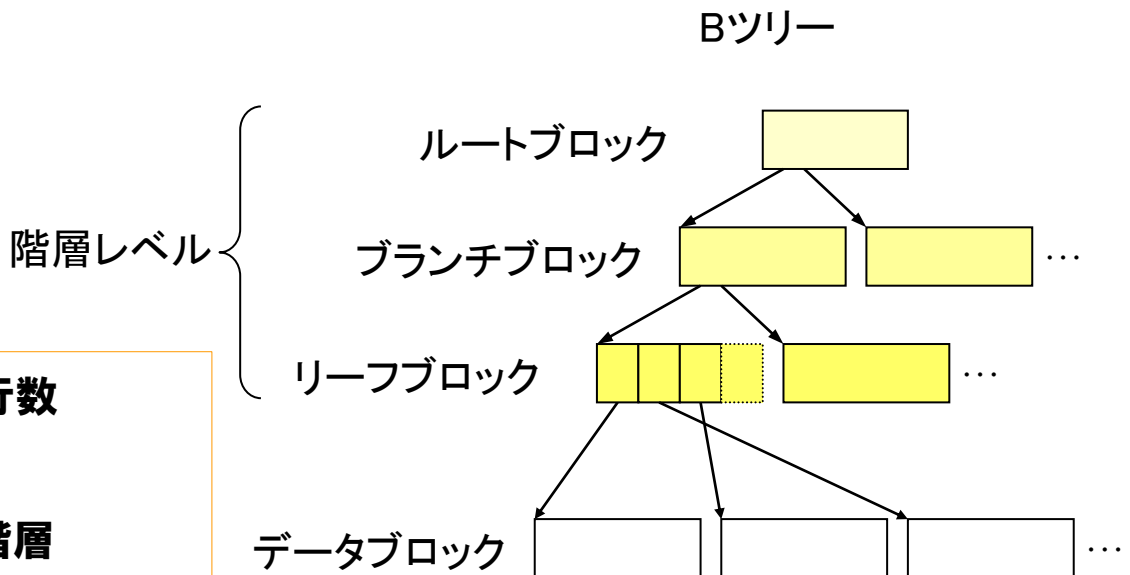
パフォーマンス問題はなぜ起きるか

データ量の増加による

- 索引(Bツリー)の階層が増える
 - $\text{ブロック内行数} = (\text{ブロックサイズ}、\text{キーサイズ})$
 - $\text{リーフ・ブロック数} = \text{行数} \div \text{ブロック内行数}$
 - $\text{ツリー階層} = (\text{リーフ・ブロック数}、\text{ブロック内行数})$

ブロック内行数が100の場合のデータ行数

10,000(100x100)行までは2階層
1,000,000(100x100x100)行までは3階層
100,000,000(100x100x100x100)行までは4階層



パフォーマンス問題はなぜ起きるか

データ量の増加による(解決策)

- I/Oの増加
 - キャッシュヒット率の改善
 - メモリ追加
 - ディスク性能の改善
 - ディスク追加、ディスク配置の改善、ASM(自動ストレージ管理)
 - パーティション化
- CPU時間の増加
 - CPUの性能アップ(性能アップするのは難しい)
 - パラレル処理
- 実行計画に影響
 - オプティマイザ統計の精度を上げる
 - 索引の追加
 - パーティション化

パフォーマンス問題はなぜ起きるか

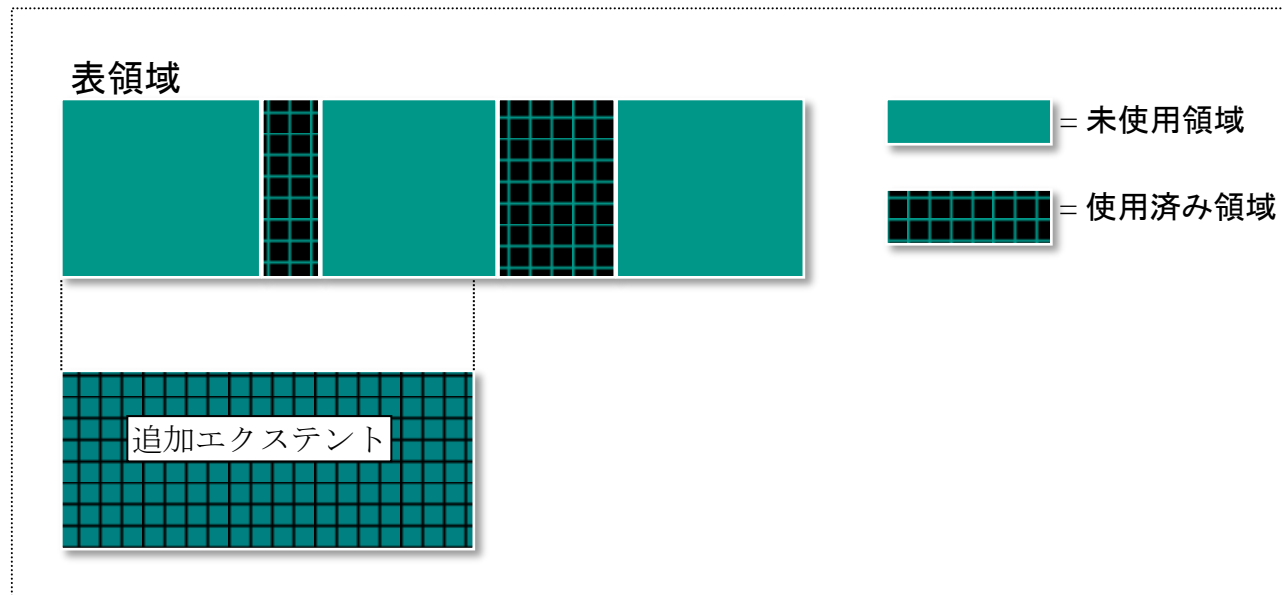
長期運用による

- 構造の劣化や情報の陳腐化によって
 - 表の断片化(エクステントの断片化)
 - I/O効率が低下
 - 索引の断片化
 - 索引アクセスのI/Oが増加
 - オプティマイザ統計が正しくない
 - 効率の悪い実行計画
- I/O時間(I/O待ちが)が増加する

パフォーマンス問題はなぜ起きるか

長期運用による

- 表の断片化(エクステントの断片化)
 - エクステントサイズが同一でないため
 - 領域が取れない、アクセスが飛び飛びになる(シークが多くなる)

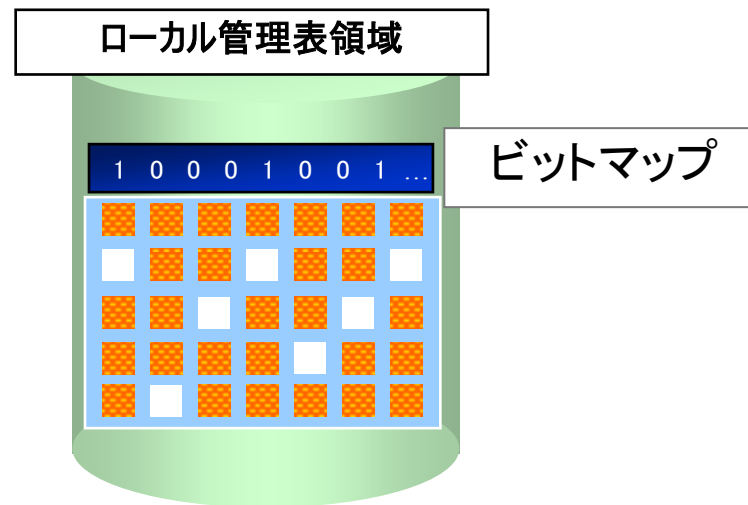


- ローカル管理表領域により削減

パフォーマンス問題はなぜ起きるか

長期運用による

- ローカル管理表領域
 - AUTOALLOCATE (デフォルト)
 - セグメントサイズに従い64KB, 1MB, 8MB, 64MBのエクステントを確保(ビットマップの管理は64KB固定)
 - UNIFORM
 - 表領域作成時に指定したエクステントサイズ固定、デフォルト1MB



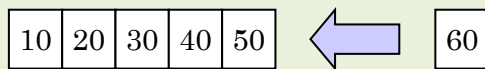
パフォーマンス問題はなぜ起きるか

長期運用による

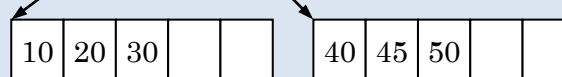
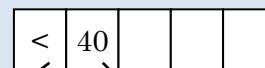
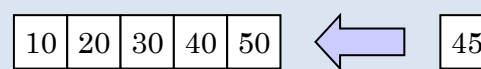
- 索引の断片化(ブロック内使用率の低下)
 - ブロックの分割
 - キーの削除

Bツリー索引の分割方法は、追加されるキーの値によって異なります。

追加キーが最大値の場合



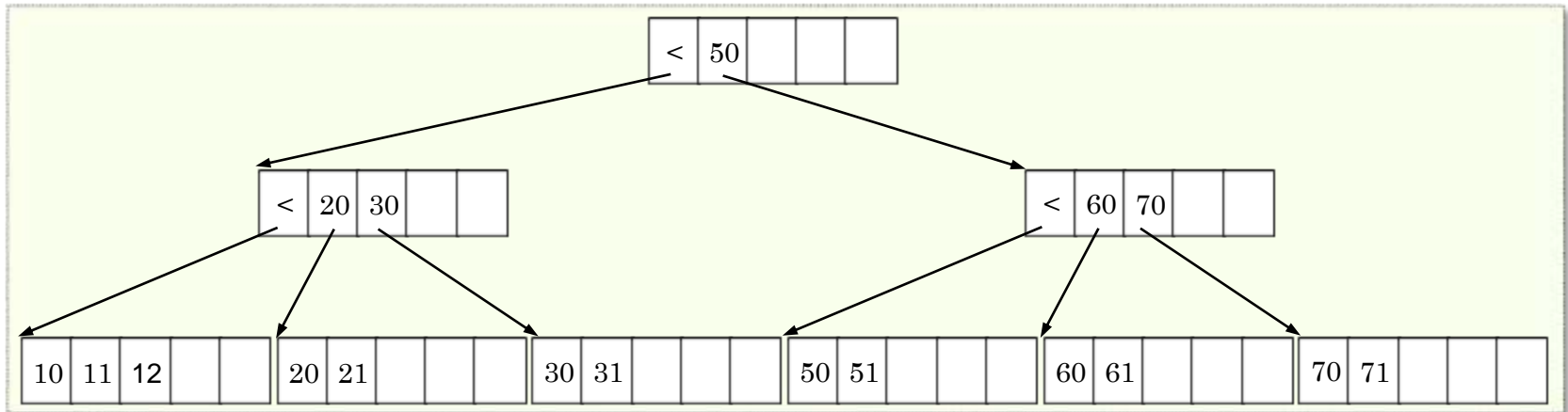
追加キーが最大値でない場合



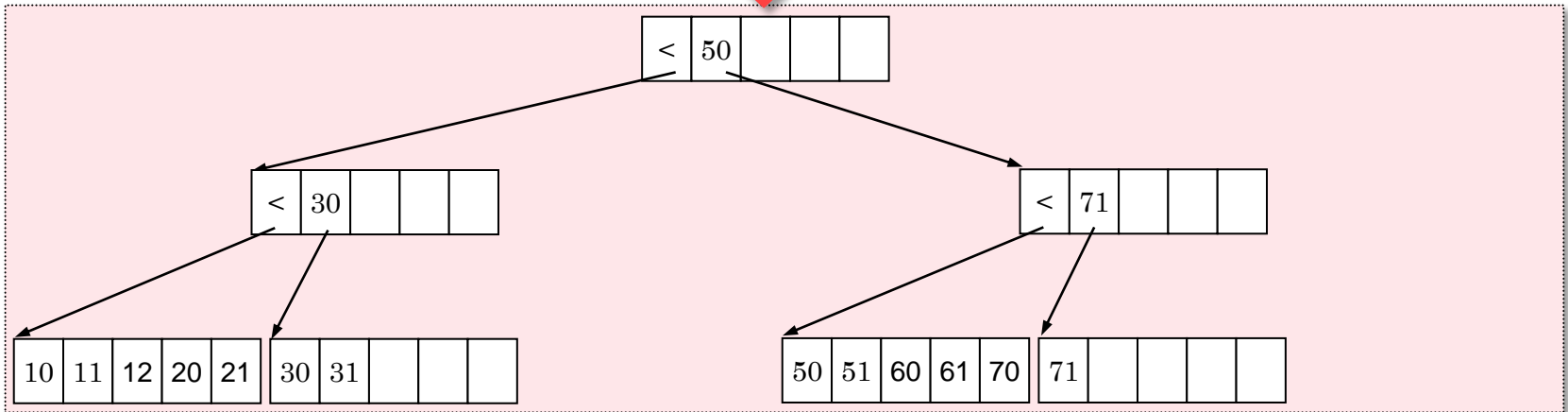
- オンライン再構築、索引の結合(COALESCE)

パフォーマンス問題はなぜ起きるか

長期運用による



ALTER INDEX <索引名> COALESCE



パフォーマンス問題はなぜ起きるか

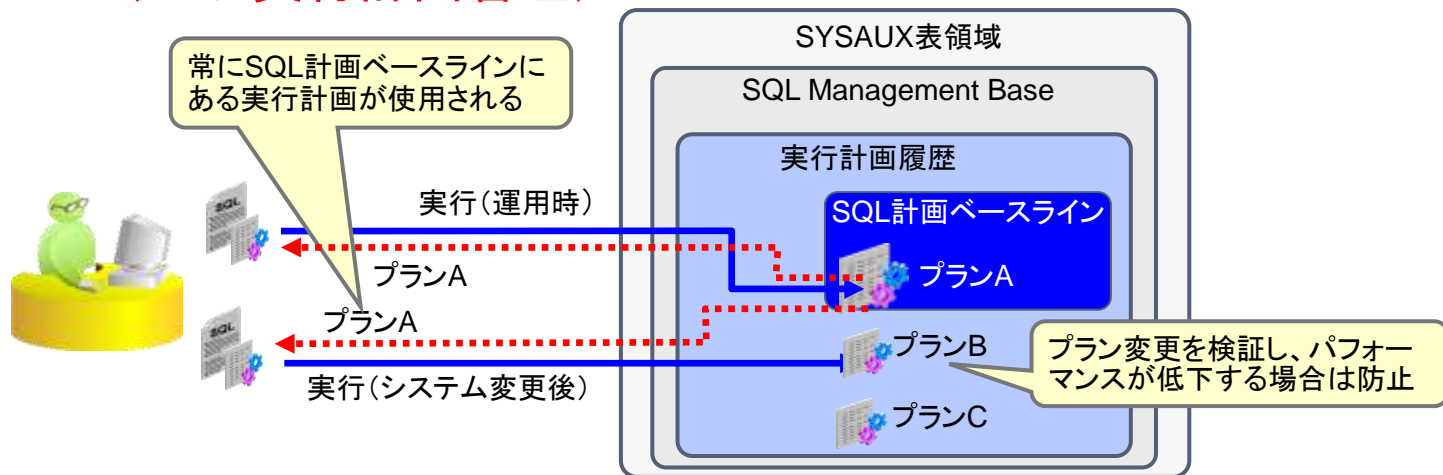
長期運用による

- オプティマイザ統計が古い(実行計画が悪くなる)
 - 自動オプティマイザ統計収集と手動収集(大量の更新)で

自動オプティマイザ統計収集

- 取得されていないオブジェクト、10%以上データが更新されているオブジェクト
- デフォルトの「スケジューラのウィンドウ」の設定: 月曜日～金曜日は22時から翌朝2時までと土曜日・日曜日は6時から翌朝2時まで

- SPM(SQL実行計画管理)



パフォーマンス問題はなぜ起きるか

データベース設計が悪い

- これは昔から考え方は変わらない
- 正規化しすぎて結合が多いなど
 - できるだけ結合しないように逆正規化などを行う
 - 冗長データなどを持つ
 - データ管理が複雑になる

パフォーマンス問題はなぜ起きるか

SQL文が悪い

- 実行計画が悪くなる
 - 索引を使用しない
 - 無駄な処理を行っている
 - データアクセスが多い
 - 同じデータを何回もアクセスしている

パフォーマンス問題はなぜ起きるか

SQL文が悪い

- 改善可能なSQL文について
 - 索引を使用しない記述である。
 - INとEXISTSについて
 - 結合は件数を絞り込んでから
 - ソート処理はできるだけ少ない件数で(できれば行わない)
 - 繰り返し副問合せはWITH句で

パフォーマンス問題はなぜ起きるか

SQL文が悪い(索引を使用しない記述)

- NULL比較やNOT(!=)を使用
(Bツリー索引使用時)

```
SQL> SELECT * FROM tab1 WHERE c1 IS NULL;
```

```
SQL> SELECT * FROM tab1 WHERE c1 IS NOT NULL;
```

```
SQL> SELECT * FROM tab1 WHERE c1 != 10;
```

パフォーマンス問題はなぜ起きるか

SQL文が悪い(索引を使用しない記述)

- 列を演算している

```
SQL> SELECT * FROM tab1 WHERE c1 / 10 < 10000;  
SQL> SELECT * FROM tab1 WHERE substr(c2, 3, 4) = 'AAAA' ;
```



```
SQL> SELECT * FROM tab1 WHERE c1 < 1000;  
SQL> CREATE INDEX ix_c2 ON tab1 (sunstr(c2, 3, 4));
```

パフォーマンス問題はなぜ起きるか

SQL文が悪い(索引を使用しない記述)

- 後方一致(中間一致)条件

```
SQL> CREATE INDEX ix_tab11 ON tab1 (c1, c2);  
SQL> CREATE INDEX ix_tab12 ON tab1 (c3);
```

```
SQL> SELECT * FROM tab1 WHERE c3 LIKE ' %AA' ;  
SQL> SELECT * FROM tab1 WHERE c3 LIKE ' %A%' ;  
SQL> SELECT * FROM tab1 WHERE c2 = 10;
```

- 索引スキップ・スキャンはする(Oracle9iから)

パフォーマンス問題はなぜ起きるか

SQL文が悪い(索引を使用しない記述)

- INリストやORを使用

```
SQL> SELECT * FROM tab1  
2   WHERE c1 = 100 OR c2 = 200 OR c3 =300;
```



```
SQL> SELECT * FROM tab1 WHERE c1 = 100  
2   UNION ALL  
3   SELECT * FROM tab1 WHERE c2 = 200  
4   UNION ALL  
5   SELECT * FROM tab1 WHERE c3 = 300;
```

パフォーマンス問題はなぜ起きるか

SQL文が悪い

- INとEXISTSについて
 - INは一意性処理(HASH UNIQUE)を行う

```
SQL> SELECT * FROM tab1 WHERE c1 IN (SELECT c1 FROM tab2);  
SQL> SELECT * FROM tab1 WHERE EXISTS  
2   (SELECT * FROM tab2 WHERE tab1.c1 = tab2.c1);
```

- 結合は件数を絞り込んでから

```
SQL> SELECT A.c2, count(*) FROM tab1 A, tab2 B  
2   WHERE A.c1 = B.c1 AND c2 = 10 GROUP BY A.c2;
```



```
SQL> SELECT A.c2, sum(ct) FROM tab1 A,  
2   (SELECT c1, count(*) ct FROM tab2 WHERE c2 = 10 GROUP BY c1) B  
3   WHERE A.c1 = B.c1 GROUP BY A.c2;
```

パフォーマンス問題はなぜ起きるか

SQL文が悪い

- ソート処理はできるだけ少ない件数で(できれば行わない)

```
SQL> SELECT * FROM (SELECT ... FROM tab1 ORDER BY c1) WHERE ROWNUM < 101;  
SQL> SELECT c1, count(*) FROM tab1 GROUP BY c1 ORDER BY c1;
```



```
SQL> SELECT /*+ NO_MERGE(A) */ FROM (SELECT c1, count(*)  
2 FROM tab1 GROUP BY c1) A ORDER BY A.c1;
```

- 繰り返し副問合せはWITH句で

```
SQL> SELECT * FROM (SELECT 部門, sum(売上) 部門売上  
2 FROM 売上表 GROUP BY 部門) w_abc  
3 WHERE 部門売上 < (SELECT avg(部門売上) FROM  
4 (SELECT 部門, sum(売上) 部門売上 FROM 売上表 GROUP BY 部門));
```



```
SQL> WITH w_abc AS (SELECT 部門, sum(売上) FROM 売上表 GROUP BY 部門)  
2 SELECT * FROM w_abc  
3 WHERE 部門売上 < (SELECT avg(部門売上) FROM w_abc);
```

バッチ処理の問題点

バッチ処理に対する悩み

- バッチ処理を速くしたいのだが、どうすれば良いか？
 - 時間内に終わらない
 - だんだん遅くなってきた
 - 難しいので諦めている
- 並列化しても性能が向上しないが？
 - 逆に遅くなった
 - CPUは余っているのに

並列化を最初から考えていますか！

バッチ処理を速くしたいのだが、
どうすれば良いか？



バッチ処理の問題点

並列処理で性能アップ

- 並列処理とは
 - なぜ並列化が必要なのか
 - マルチCPUコア環境では性能アップするためには
- 並列処理する上での注意点
 - どんな問題が発生するか
 - バッチ処理などでは更新がメインなので競合が発生し易い
- RACでの注意点
 - 更にRACでは

バッチ処理の問題点

並列処理とは

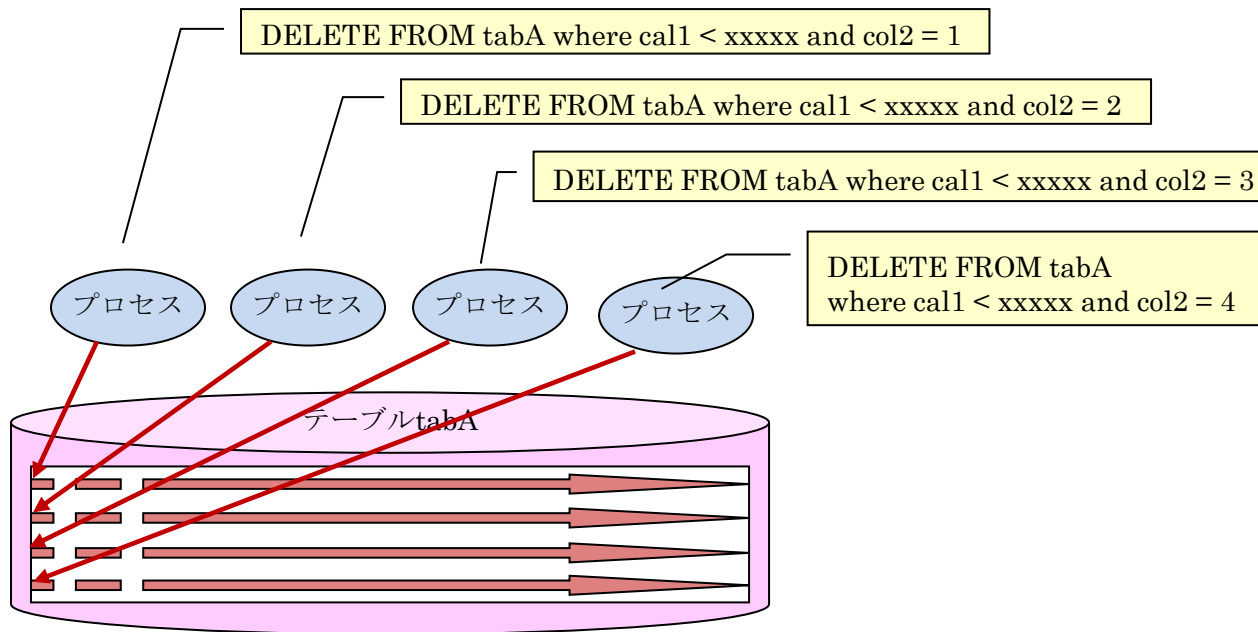
- データ量の増加とともに処理時間も増加する
 - 並列化で処理時間を短縮
 - マルチCPUコアを有効活用
- 実施方法
 - プログラム分割
 - パラレルDML (SQL文で)

バッチ処理の問題点

並列処理とは

- プログラム分割

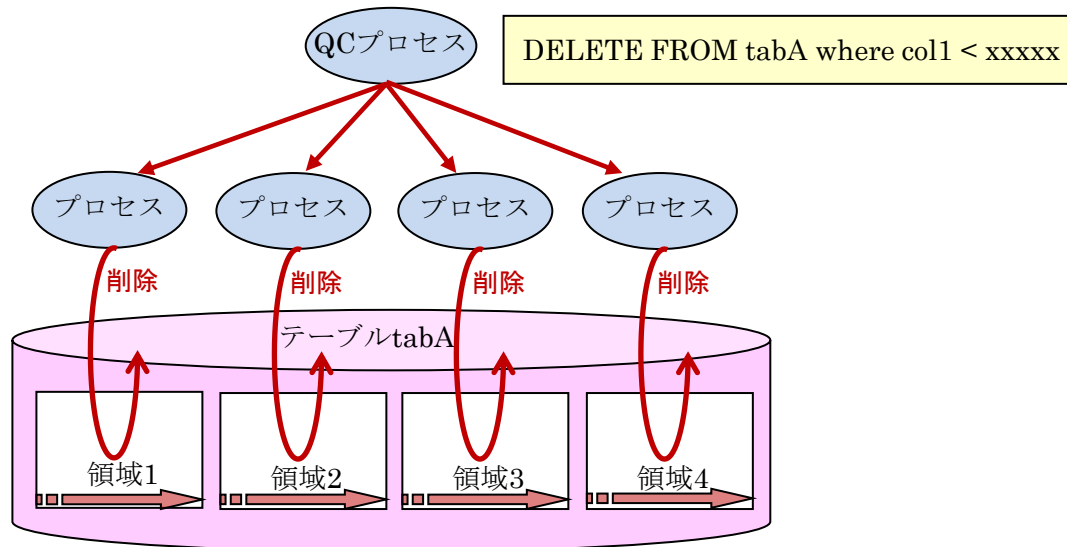
- データが分割されていないので各プログラムですべてのデータにアクセスする



バッチ処理の問題点

並列処理とは

- パラレルDML
 - 内部的にデータが分割されて並列に処理する



バッチ処理の問題点

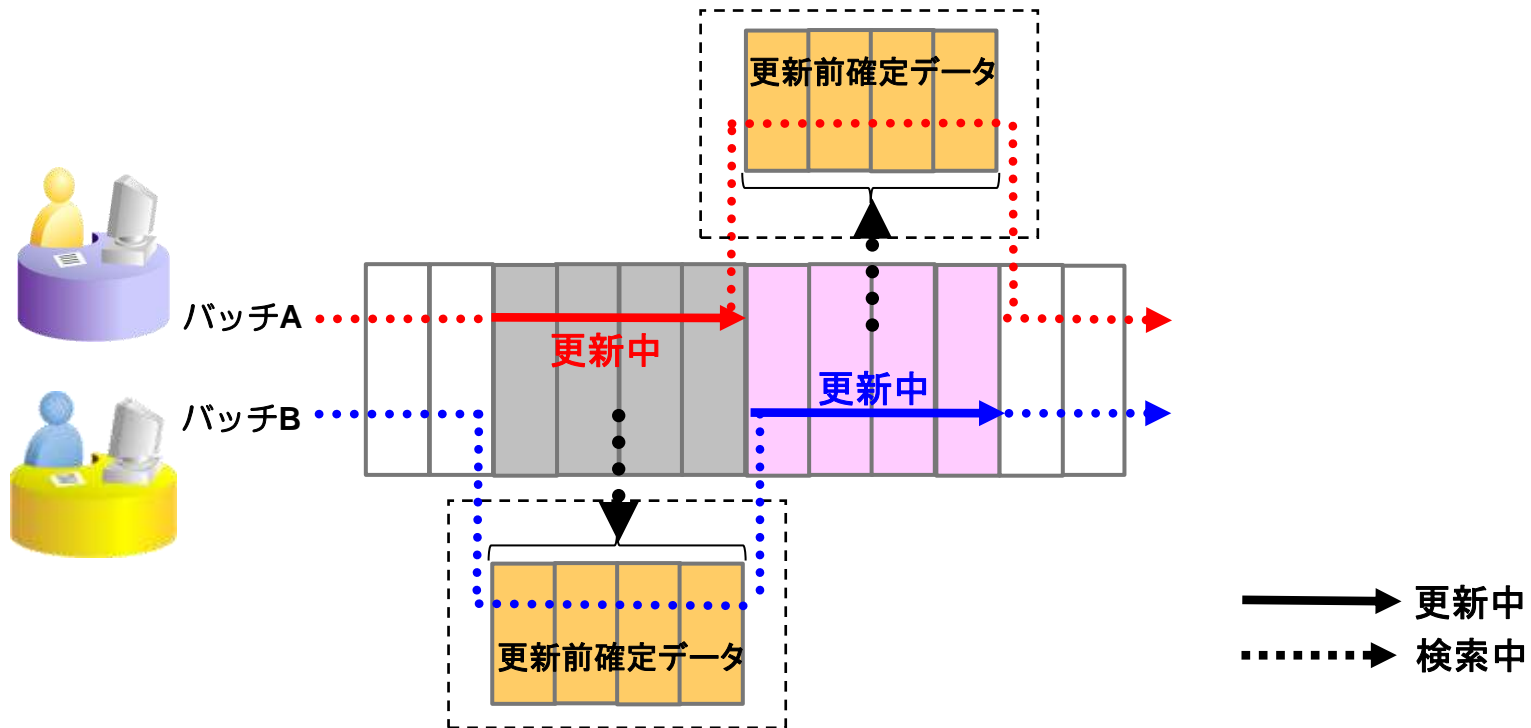
並列処理する上での注意点

- 並列化することで競合が発生する
 - バッチ処理は更新がメインなのでより発生し易い
 - ユーザ数が増加する場合と同じ
 - ブロック競合、行ロック競合、I/O競合、など
 - これ以外にも様々な競合が発生する
- 競合以外に、読み取り一貫性によるUNDO読込みが多発する
- 表分割又はパーティション化する

バッチ処理の問題点

並列化する上での注意点(競合の例)

- 読み取り一貫性によるUNDO読み込みが多発する
 - 更新中データを別プロセスで検索するため
 - 行ロックが発生しないから気にしなくても良いと思ったら大間違い

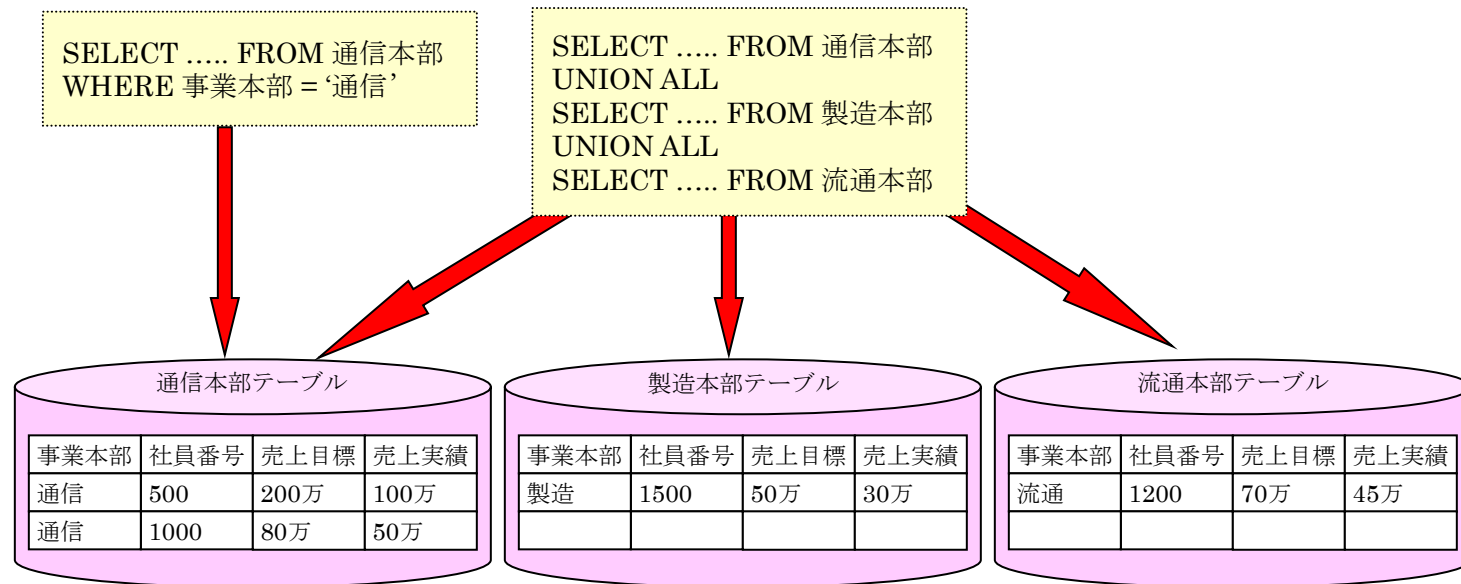


バッチ処理の問題点

並列処理する上での注意点

- 表分割

- 本部単位に別テーブルにするためアプリケーションでアクセスするデータごとにテーブル名を変える必要がある



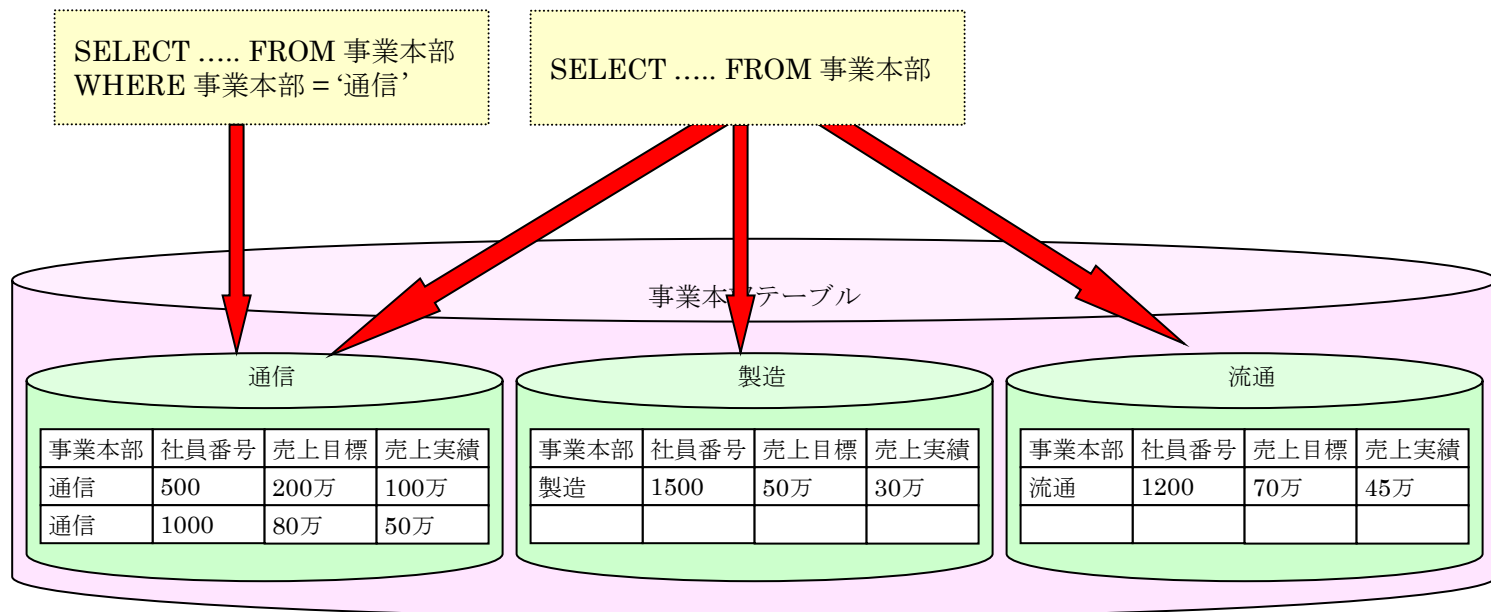
- それぞれのバッチプロセスで別々のテーブルにアクセスする

バッチ処理の問題点

並列処理する上での注意点

- パーティション化

- 事業本部テーブルを本部単位（通信、製造、流通）にパーティション化することでアクセスするデータを意識する必要がない（WHERE句の条件でそれぞれのパーティションだけにアクセス）

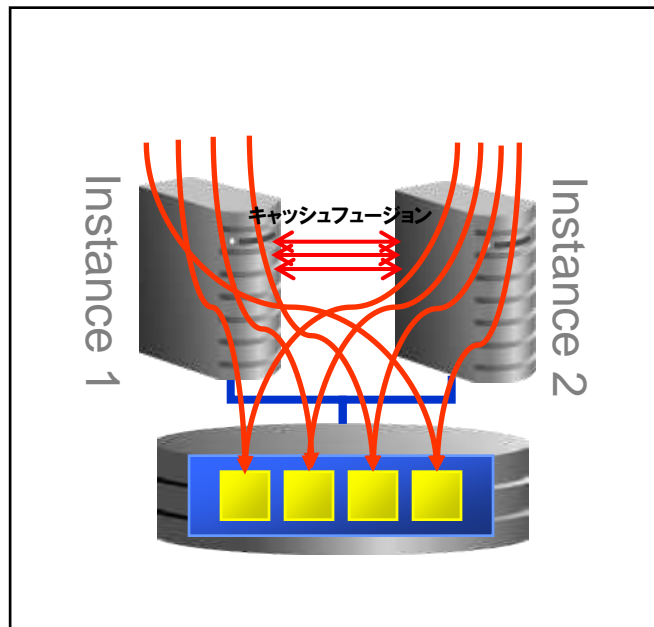


- それぞれのバッチプロセスで別々のパーティションにアクセスする

バッチ処理の問題点

並列化する上での注意点(RACでの注意点)

- RACではキャッシュ・フュージョンが発生する
 - 更に性能ダウンする
 - シングルノード(非RAC)より遅くなる場合も



並列処理の問題点

RACでの注意点

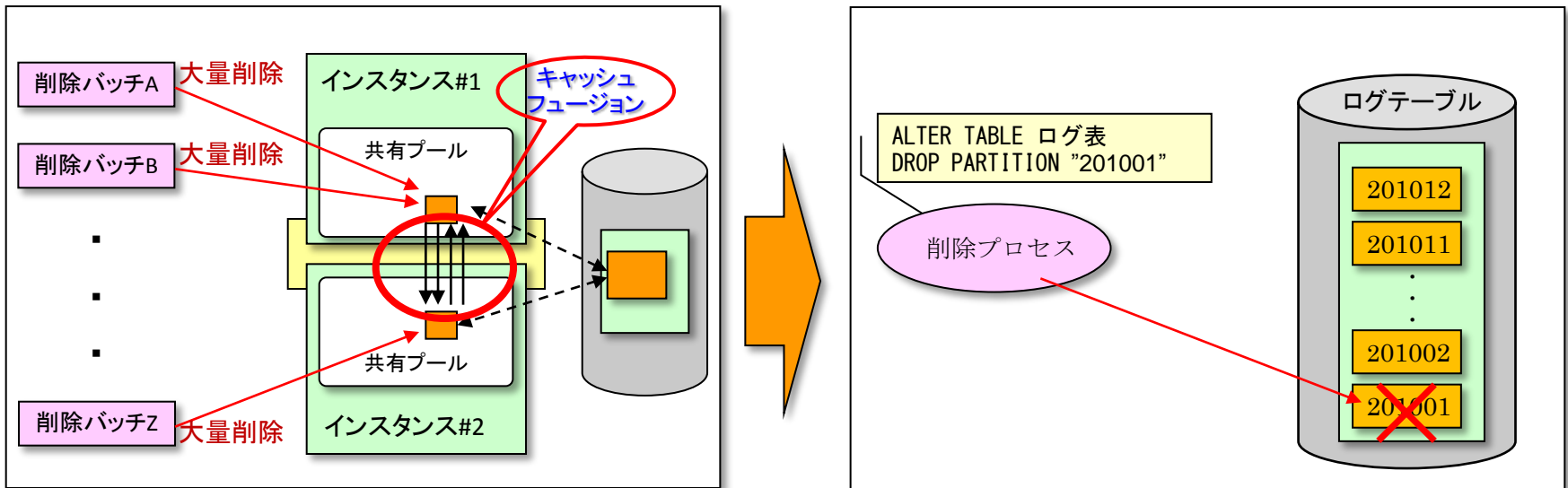
- どうするのか
 - パラレルDML
 - パーティション化(表分割でも可)
 - シングルノードで実行(どうしてもできなければ)
- パーティション化、パラレルDMLによる事例
 - ログなどのデータ削除
 - 名寄せ処理による差分更新
 - 外部ファイルの取り込み(マスタデータのチェック含む)
- 並列化が難しいと思う処理もパーティション化することで簡単に改善できたりします

並列処理の問題点

RACでの注意点(パーティション化による事例)

- ログなどのデータ削除

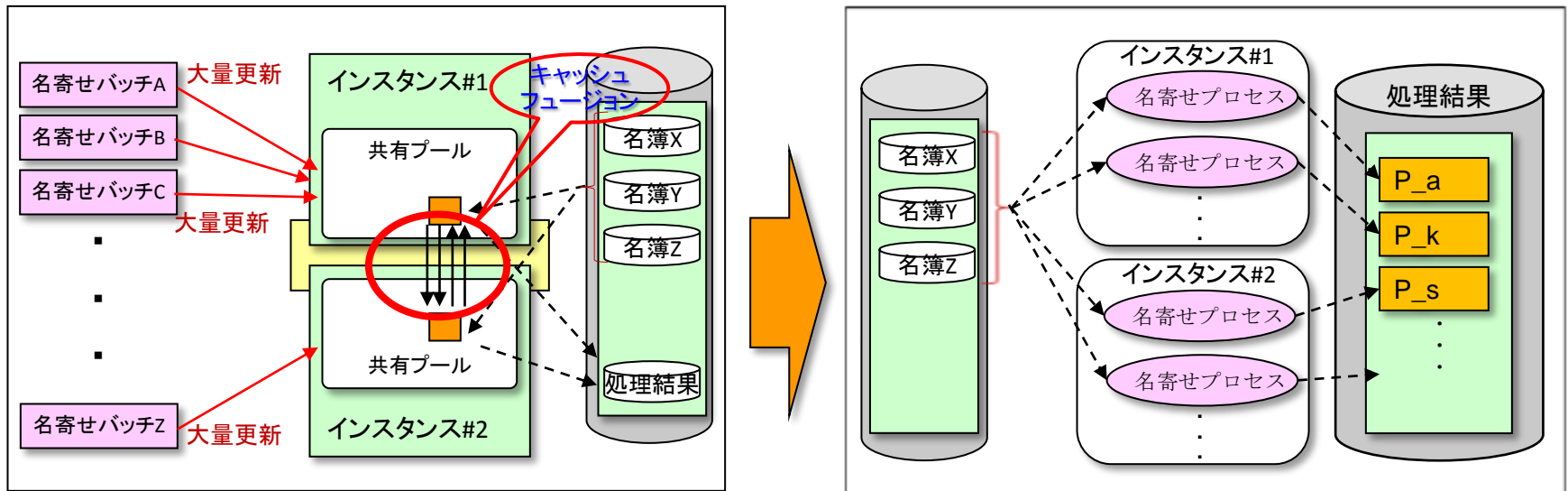
- アクセス・ログを定期的に削除するバッチ処理をRACで行うと逆に遅くなる(キャッシュ・フュージョンが多発する)のでシングルで行っていた事例です。
- 【改善策】削除する単位にパーティション化してDROP PARTITIONを行うことで一瞬で削除できます。



並列処理の問題点

RACでの注意点(パーティション化による事例)

- 名寄せ処理による差分更新(存在しないものは挿入)
 - 定期的に顧客の名寄せを行うバッチ処理をRACで行うと逆に遅くなる(キャッシュ・フュージョンが多発する)のでシングルで行っていた事例です。
 - 【改善策】並列処理する単位でパーティション化して2ノードから並列実行してもキャッシュ・フュージョンが発生しないようにします(それぞれのプロセスで更新するデータは別々のパーティションになるようにします)。

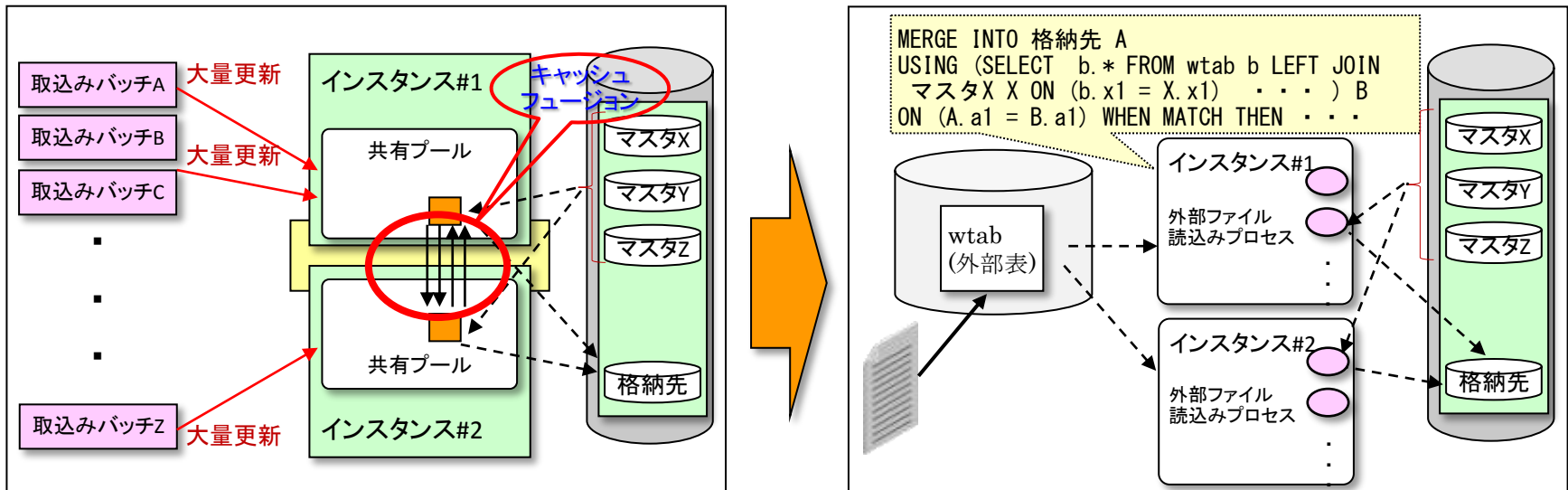


並列処理の問題点

RACでの注意点(パラレルDMLによる事例)

- 外部ファイルの取り込み(マスタデータのチェック含む)

- 外部ファイルをマスタテーブルと整合性チェックをして格納先テーブルに差分更新するバッチ処理をRACで行うと逆に遅くなる(キャッシュ・フュージョンが多発する)のでシングルで行っていた事例です。
- 【改善策】外部ファイルを外部表としてパラレルDML(MERGE文で差分更新)を整合性チェックをしながら行うことで、キャッシュ・フュージョンを発生しないようにします。



最後に、

「津島博士のパフォーマンス講座」
質問お待ちしておりますので、
よろしくお願いします。



OTNセミナーオンデマンド

コンテンツに対する
ご意見・ご感想を是非お寄せください。

OTNオンデマンド 感想



http://blogs.oracle.com/oracle4engineer/entry/otn_ondemand_questionnaire

上記に簡単なアンケート入力フォームをご用意しております。

セミナー講師/資料作成者にフィードバックし、
コンテンツのより一層の改善に役立てさせていただきます。

是非ご協力をよろしくお願いいたします。

OTNセミナーオンデマンド

日本オラクルのエンジニアが作成したセミナー資料・動画ダウンロードサイト

掲載コンテンツカテゴリ(一部抜粋)

Database 基礎

Database 現場テクニック

Database スペシャリストが語る

Java

WebLogic Server/アプリケーション・グリッド

EPM/BI 技術情報

サーバー

ストレージ



超入門! Oracle データベースって何
再生時間: 60分

100以上のコンテンツをログイン不要でダウンロードし放題

データベースからハードウェアまで充実のラインナップ

毎月、旬なトピックの新作コンテンツが続々登場

例えばこんな使い方

- 製品概要を効率的につかむ
- 基礎を体系的に学ぶ/学ばせる
- 時間や場所を選ばず(オンデマンド)に受講
- スマートフォンで通勤中にも受講可能



毎月チェック!



コンテンツ一覧 はこちら

<http://www.oracle.com/technetwork/jp/ondemand/index.html>

新作&おすすめコンテンツ情報 はこちら

<http://oracletech.jp/seminar/recommended/000073.html>

OTNオンデマンド



オラクルエンジニア通信

オラクル製品に関わるエンジニアの方のための技術情報サイト

オラクルエンジニア通信 - 技術資料、マニュアル、セミナー

Oracleエンジニアのための技術情報サイト by Oracle Japan

新着情報を知りたい

技術資料を探したい

セミナーを受けたい

About

Oracleエンジニアの方がスキルアップしていただくために、厳選した情報をお届けしています

技術資料	<p>インストールガイド・設定チュートリアルetc. 欲しい資料への最短ルート</p>	アクセスランキング	<p>他のエンジニアは何を見ているのか？人気資料のランキングは毎月更新</p>
特集テーマ Pick UP	<p>性能管理やチューニングなど月間テーマを掘り下げて詳細にご説明</p>	技術コラム	<p>SQLスクリプト、索引メンテナンスetc. 当たり前運用/機能が見違える!?</p>

<http://blogs.oracle.com/oracle4engineer/>

オラクルエンジニア通信



The screenshot shows the top section of the oracletech.jp website. On the left is the 'oracletech.jp' logo with the tagline '好奇心が、エンジニア人生を豊かにする。'. On the right is the 'ORACLE' logo, a search bar, and social media icons for Twitter, Facebook, Ustream, YouTube, and RSS. Below these is a red navigation bar with five buttons: '製品/技術情報', 'スキルアップ', 'セミナー', 'キャンペーン', and 'ちょっと一息'.

製品/技術
情報



Oracle Databaseっていくら？オプション機能も見積れる簡単ツールが大活躍

セミナー



基礎から最新技術までお勧めセミナーで自分にあった学習方法が見つかる

スキルアップ



ORACLE MASTER ! 試験頻出分野の模擬問題と解説を好評連載中

Viva!
Developer



全国で活躍しているエンジニアにスポットライト。きらりと輝くスキルと視点を盗もう

<http://oracletech.jp/>

oracletech



あなたにいちばん近いオラクル



Oracle Direct

まずはお問合せください

Oracle Direct



システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。
システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力にはログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので
ご登録の連絡先が最新のものになっているかご確認下さい。

フリーダイヤル

0120-155-096

※月曜～金曜
9:00～12:00、13:00～18:00
(祝日および年末年始除く)

ORACLE

Hardware and Software **Engineered to Work Together**

ORACLE®