OWI(Oracle Wait Interface)の概要と実用ツールMaxGaugeの紹介

平成21年 11月 7日

アスター

GOAL：理解して頂きたいポイント

MaxGauge
Database Performance Maximizer

Oracleは常に稼働ログを記録している

その稼働ログを収集しておくと

快適なOracle運用が実現出来る

目次

MaxGauge
Database Performance Maximizer

Ⅰ. OWI：Oracle Wait Interface

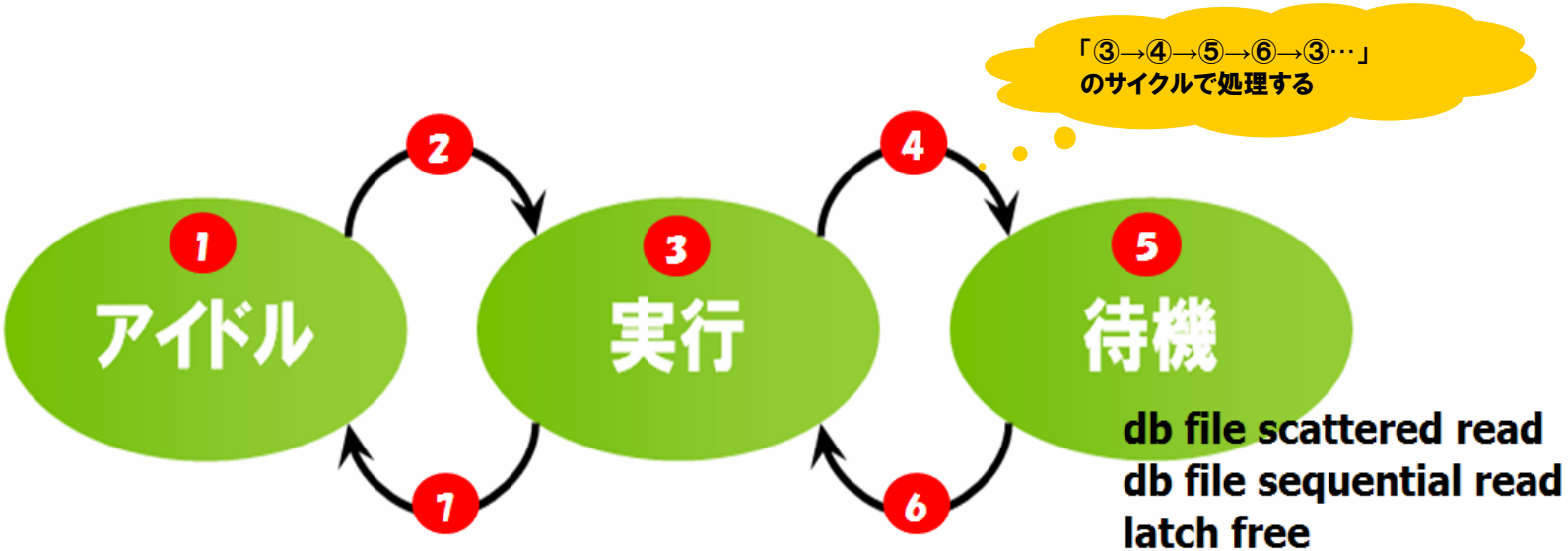
- OWI概要
- Oracle稼働ログ収集の仕組み
- 稼働ログを収集しないと時の弊害
- 収集すべき稼働ログ
- 稼働ログの収集例

Ⅱ. Oracle稼働ログの収集ツール：MaxGauge

- MaxGauge概要
- デモ
- 活用場面

プロセスの状態とOWI

MaxGauge
Database Performance Maximizer



① CPUを必要としない状態

② 作業要求が入って、CPUを使い始める

③ CPUを使って、作業中

④ 次の処理を進めるため必要なリソースを要求

⑤ CPUを使って処理を進めたいが、何らかの理由でリソースの獲得待ち状態
　一部の待機はアイドル(スリープ)状態になる

⑥ リソースが獲得できて、次の処理を進めるためCPUを使い始める

⑦ アイドル(スリープ)状態になる

プロセスの状態とOWI

MaxGauge
Database Performance Maximizer

```
SQL> select event, p1, p2, p3, seconds_in_wait from v$session where sid = 146;

EVENT                                 P1         P2         P3 SECONDS_IN_WAIT
----------------------------- ---------- ---------- ---------- ---------------
db file scattered read                 5       9548          8              12
```

```
SQL> select event#, name, parameter1, parameter2, parameter3, wait_class
  2  from v$event_name where name = 'db file scattered read';

    EVENT# NAME                           PARAMETER1      PARAMETER2      PARAMETER3      WAIT_CLASS
---------- ------------------------------ --------------- --------------- --------------- -------------
       117 db file scattered read         file#           block#          blocks          User I/O
```

OWIとは？

MaxGauge
Database Performance Maximizer

OWI = Oracle Wait Interface

データベース内部処理の待機時間を基にした、パフォーマンスボトルネック分析のための新メソッド

Oracle DBのパフォーマンスをOracleが吐き出す待機イベントを中心に管理しよう!!!

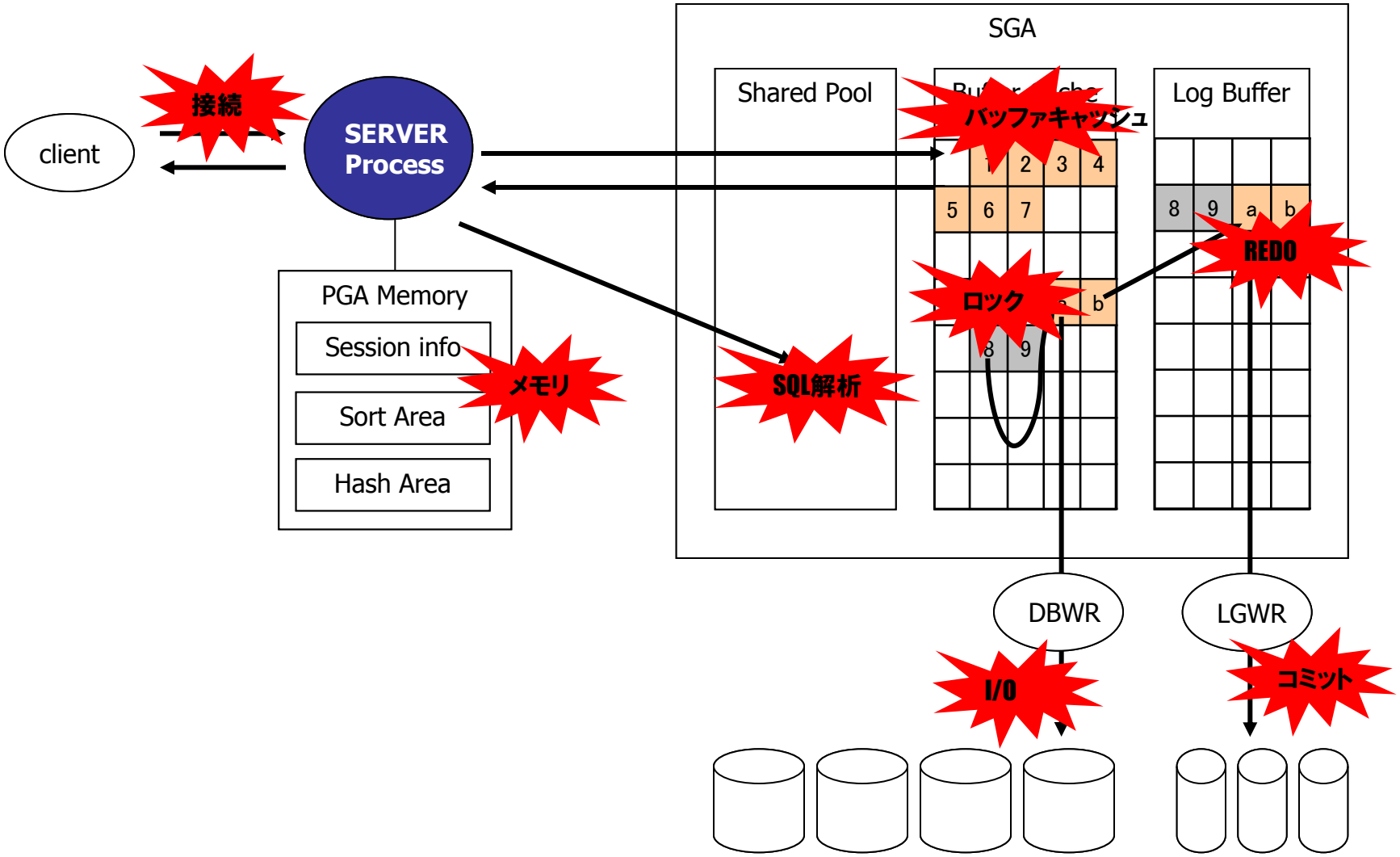
データベース内の各処理工程にセットされたタイマーを元に、各ステップでのリソース獲得の
待ち時間に着目し、レスポンスタイムを定義

処理時間（応答時間） = サービス時間（CPU使用時間） + 待機時間

レスポンスタイムの最小化 ＝ 待ち時間の最小化

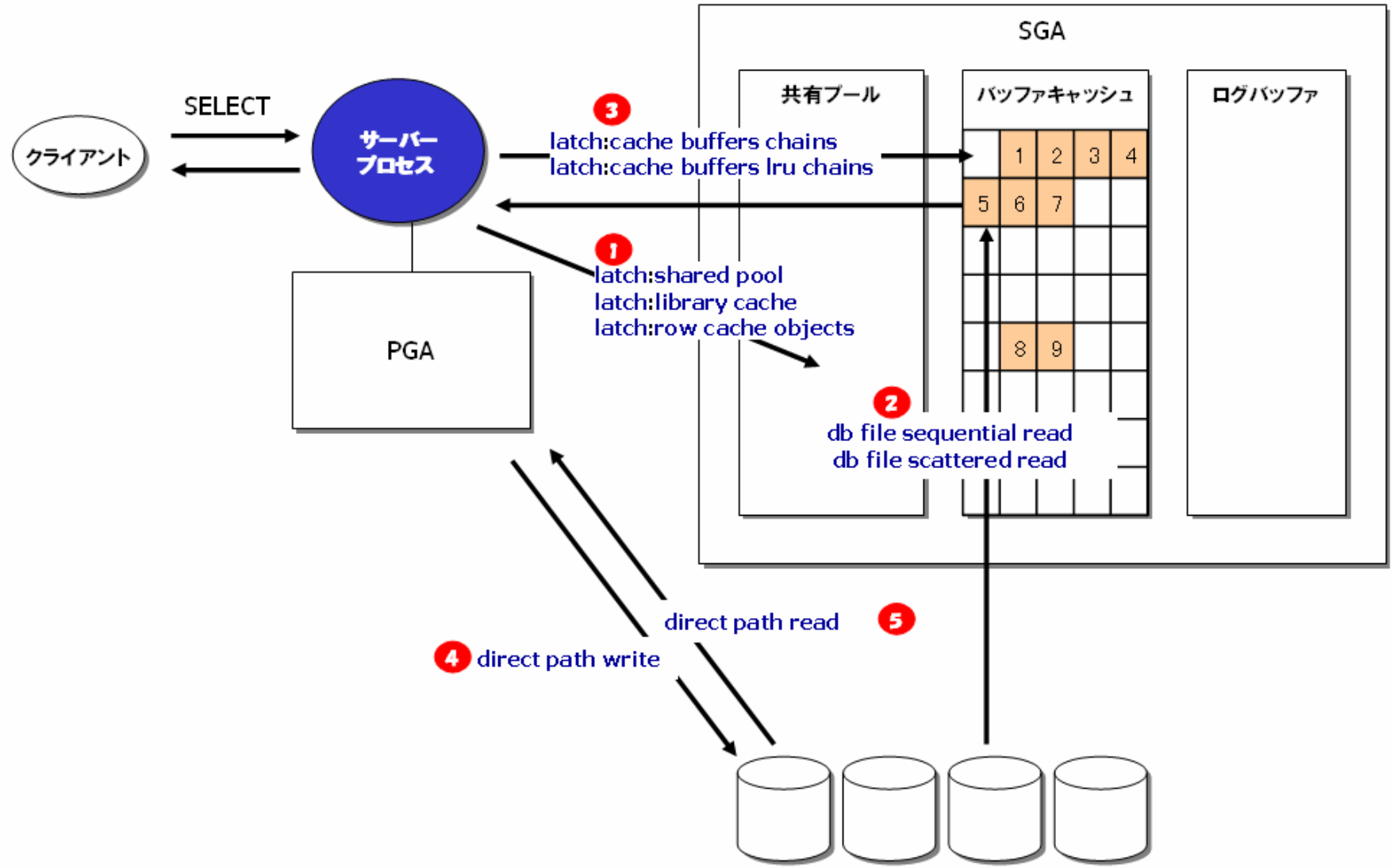
AP処理の流れ：ボトルネック箇所

MaxGauge
Database Performance Maximizer



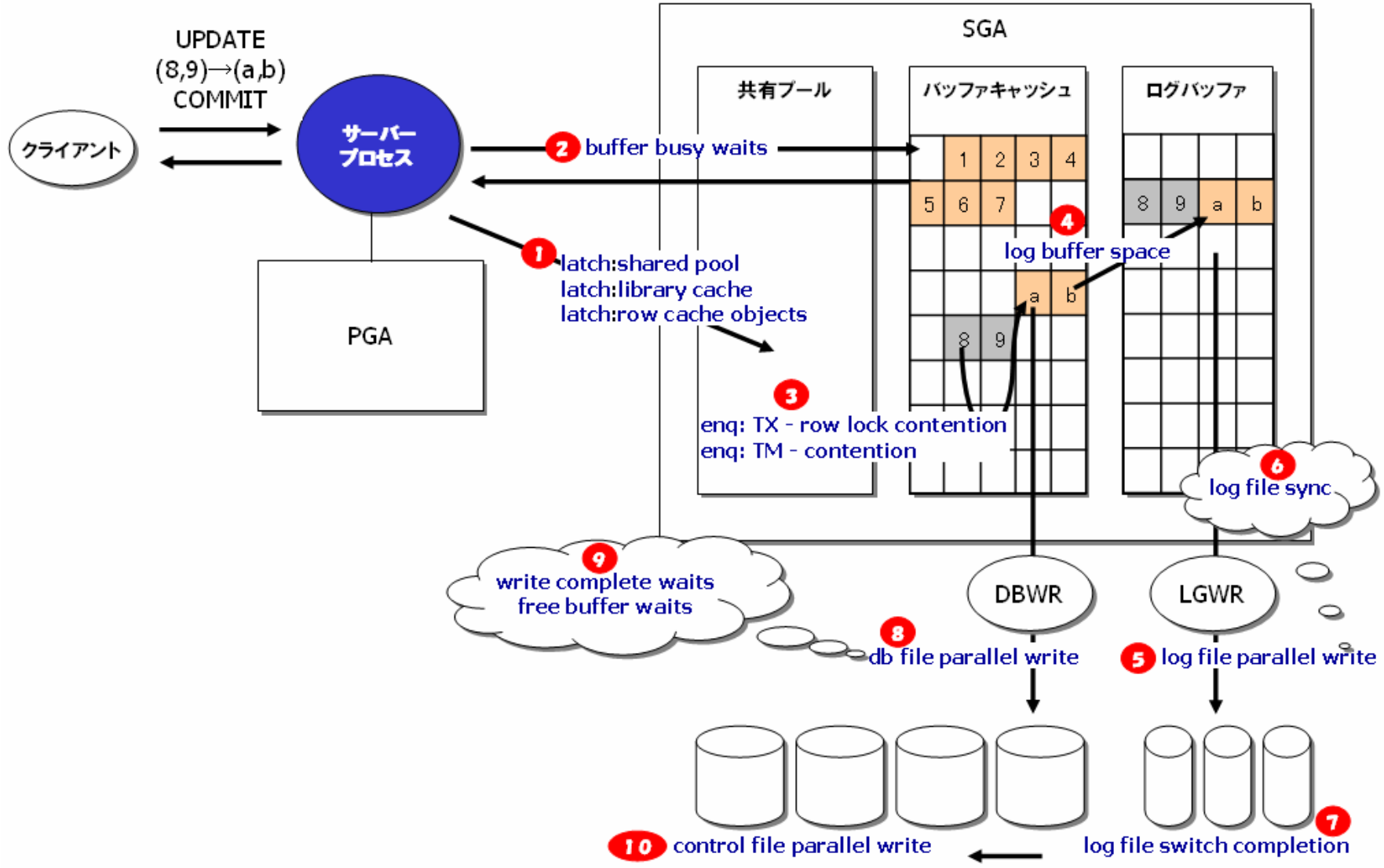
SELECT処理と待機イベント

MaxGauge
Database Performance Maximizer



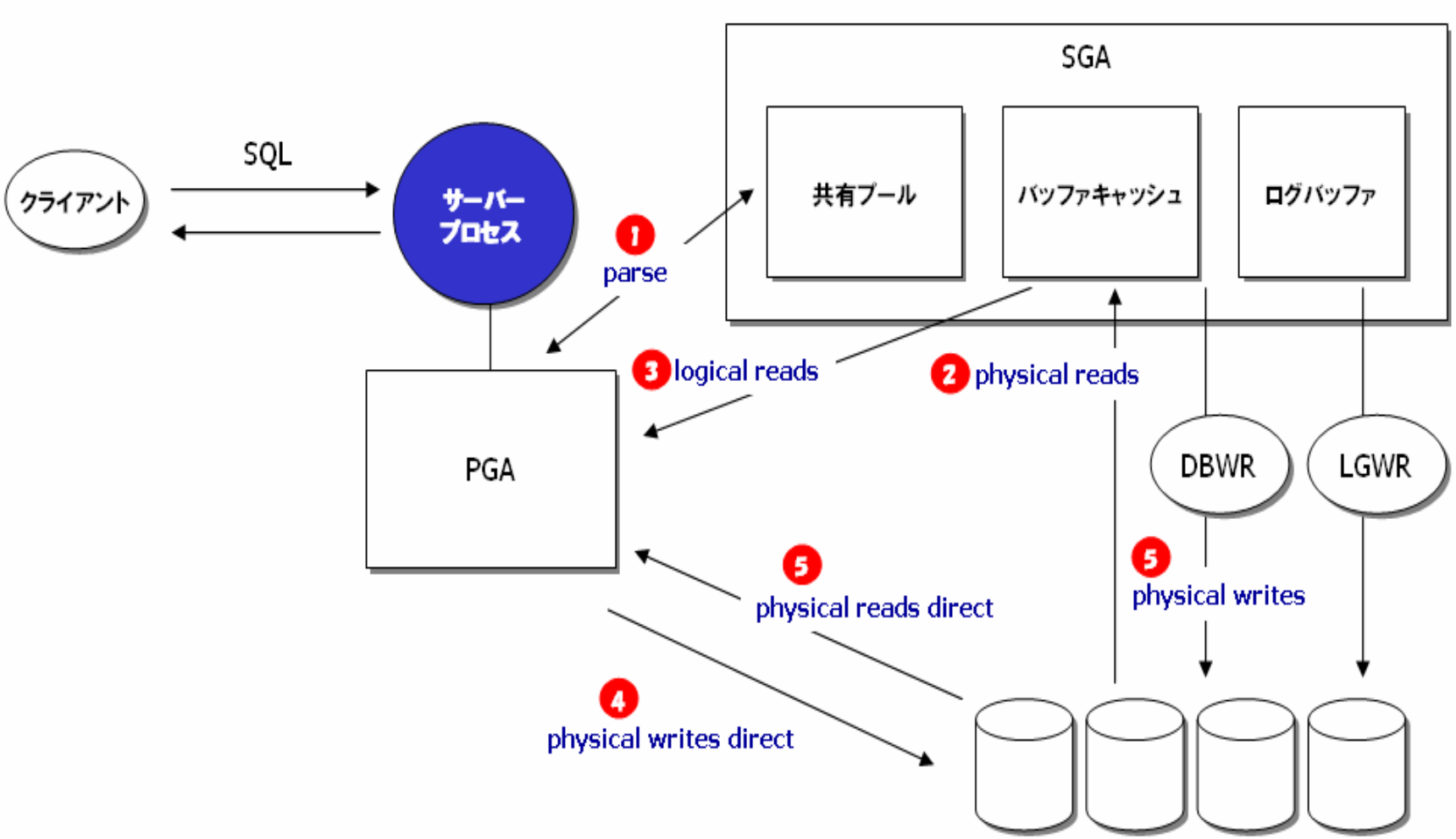
UPDATE処理と待機イベント

MaxGauge
Database Performance Maximizer



Oracleアーキテクチャーと性能統計

MaxGauge
Database Performance Maximizer



OWIの構成要素 → Oracle稼働情報

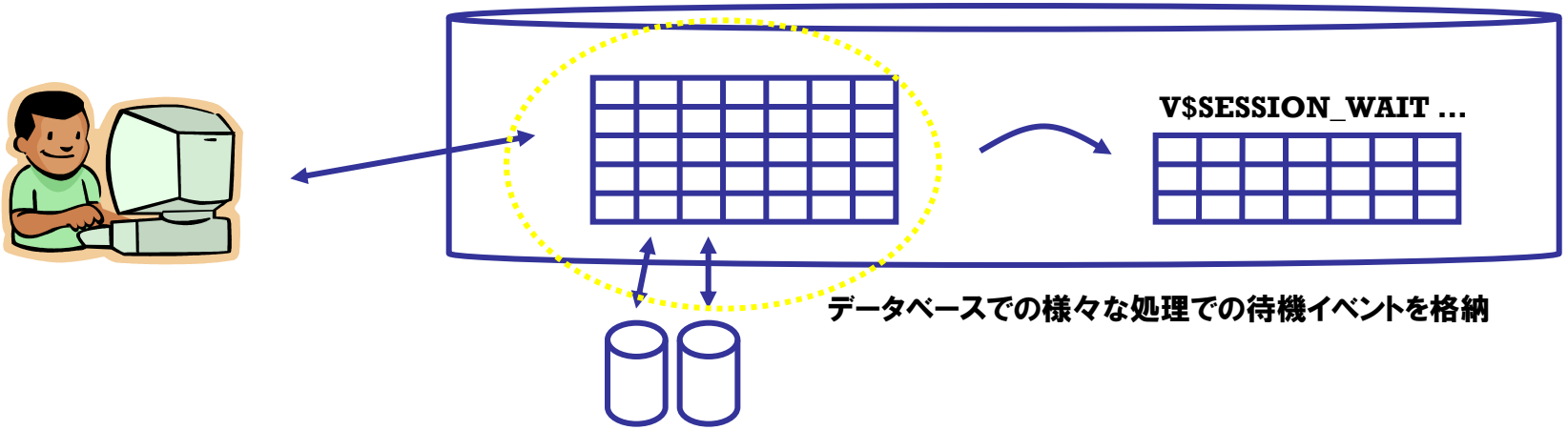
MaxGauge
Database Performance Maximizer

すべてのセッションは処理を行うためにはリソースが必要であり、各セッションがCPUを使用していないときは、何かしらの待ちが発生している状態となる。

- データファイルからのデータブロック読み書きでのI/O待ち
- メモリの獲得待ち
- 他処理との連携待ち

データベース処理にて、発生した待機イベントが、オラクルの動的ビューへ格納される。

V$EVENT_NAME
V$SESSION_WAIT
V$SESSION_EVENT
V$SYSTEM_EVENT
……



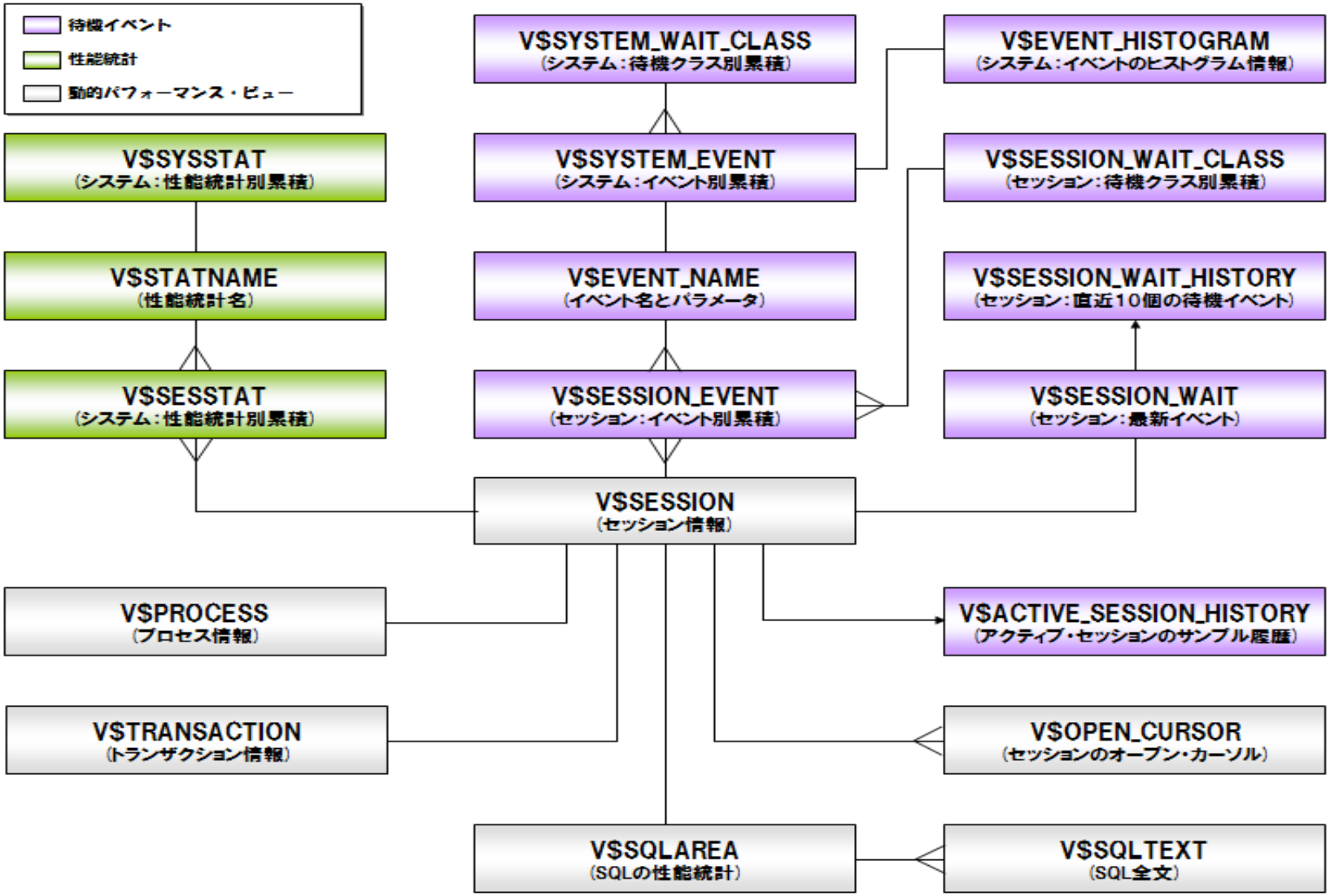
OWIの構成要素 → Oracle稼働情報

MaxGauge
Database Performance Maximizer

項目	定義	備考
V$EVENT_NAME	インスタンスで定義している待機イベントの情報	イベントの数、正確な名称、待機クラスの参照
V$SYSTEM_EVNET	インスタンスの起動後、全セッションで発生した待機イベントの累計統計値(インスタンス単位)	インスタンスの全般的な安定度を判断、デルタ情報を算出して特定時間帯の状態を診断/分析ができる → Staspack機能
V$SESSION_EVENT	現在接続されている全セッションについての、各セッション別待機イベントの累計統計値	接続中のセッションについて、各イベント別統計情報の把握ができる
V$SESSION_WAIT	各セッションが現在待機しているイベント、リソースの詳細情報を、参照時のリアルタイムで提供,	累積データではなくリアルタイムのデータであるため、短い間隔のクエリーで繰り返し参照することで、待機イベントの状況の把握に有効
V$SYSTEM_WAIT_CLASS	10g, インスタンスの起動後発生した待機クラスの累積情報	待機イベントのクラス単位で、インスタンスの安定度の把握に有効
V$SESSION_WAIT_CLASS	10g, 現在接続されている全セッションについて、セッションレベルの待機クラスの累積情報	待機イベントのクラス単位で、セッションの待機状況の把握に有効
V$SESSION_WAIT_HISTORY	10g, 直近の10個の待機イベントの情報	直近のセッションの履歴情報の把握に有効
V$EVENT_HISTOGRAM	10g, インスタンスの起動後の待機イベントのヒストグラム提供	各バケット(待機時間の区間)別の待機イベントの把握に適切
V$ACTIVE_SESSION_HISTORY	10g, アクティブセッションの履歴の情報	1秒単位でのセッションのスナップショットを保存しているため、各セッションの待機イベントなどの情報の追跡に適切
10046 Trace Event	SQLトレース、待機イベント、バインド変数などの情報を提供	履歴情報を含め、途切れの無い情報の把握やSQL/待機イベント/バインド変数の連携分析に適切

OWIの構成要素 → Oracle稼働情報

MaxGauge
Database Performance Maximizer



Oracle稼働ログ収集の仕組み

MaxGauge
Database Performance Maximizer

①動的パフォーマンス・ビュー :「v$...」→ **OWIは稼働ログ収集の一つの仕組み**

②各種ログ：アラートログ、リスナーログ

③トレース：SQLトレース、イベントトレースなど

④STATSPACK（8.1.6以降）

⑤AWR（10g以降）

①その瞬間の稼働状況のため、履歴が残らない

②Oracle運用で極一部の情報しか収集されない

③手動で収集するか、特定ケースでしか収集されない

④通常1時間おきのスナップショットデータで精度が低い、セッションデータがない

⑤通常1時間おきのスナップショットデータで精度が低い、セッションデータがない

Oracle稼働ログ収集の仕組み

MaxGauge
Database Performance Maximizer

最も一般的性能分析ツール

Oracle標準のパフォーマンスレポートツール

EE、SE、SE1でも使用可能

無償提供ツール

Oracleサポートセンターとの意思疎通ツール

一定期間の性能全般の評価に最適ツール

※ 向いて無い場合

. ログ収集の負荷が懸念される場合

. セッションデータの収集が必須な場合

. 短い間隔のデータが必要なとき：平均化は駄目

. 任意時間帯の分析が必要なとき

. 時系列の情報が必須な場合：GUI化

. リテラルSQLが多い場合

. 収集対象SQLの条件をユーザー定義する場合

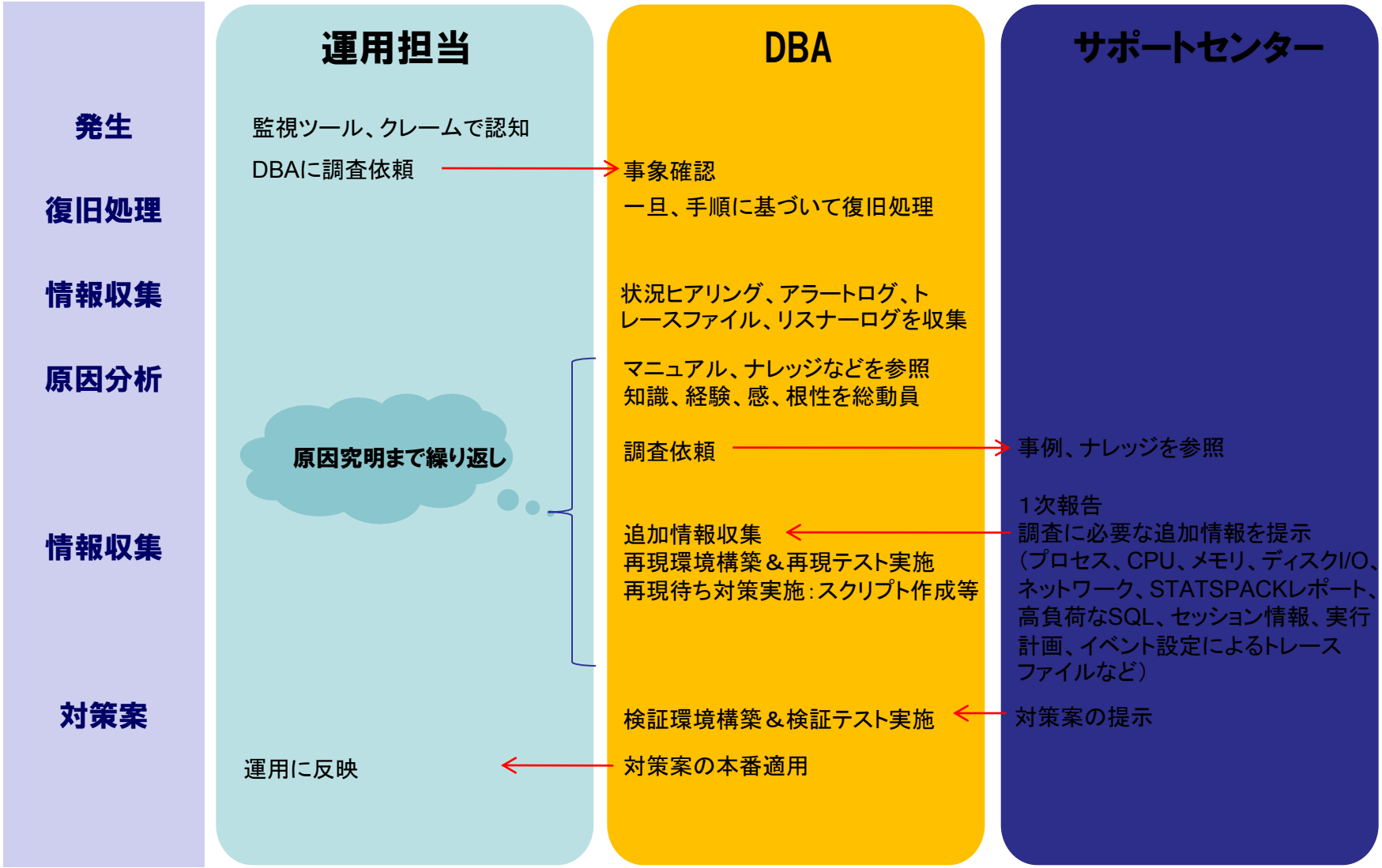
Oracle稼働ログを収集しないと？

MaxGauge
Database Performance Maximizer

- トラブルが本当に発生してからでないと認知ができない
- トラブル発生後にはリカバリが最優先となり、情報収集を綿密に、
  正確に行う時間等が取れない
- トラブル発生後の調査が手探りとなってしまう
- 情報収集が乏しいため、原因追求に非常に時間がかかる
  または、原因追求が出来ないケースが多々ある
- トラブル対処が優先となり、他の業務に多大な影響を与える
- 類似のトラブルが起きても初期調査から別対応になってしまう

Oracle稼働ログを収集しないと？

MaxGauge
Database Performance Maximizer



常時収集すべき稼働データ

MaxGauge
Database Performance Maximizer

収集データ	性能障害	システムハング	接続エラー	システムダウン
アラート・ログ		○	○	○
トレースファイル		△	△	△
リスナー・ログ		○	○	
プロセスリスト	△	○		○
CPU使用状況	◎	○	○	○
メモリ使用状況	○	○		○
ディスクI/O状況	○			
ネットワーク状況	△	△	○	△
性能統計指標(11.1.0.6.0:469個)	○	○		○
DBセッション数	○	◎	◎	◎
CPU使用時間	○			
論理読取ブロック数	○	△		△
物理読取ブロック数	○	△		△
待機指標(11.1.0.6.0:959個)	◎	○		△
物理読取待ち	○			△
ロック待ち	○	○		△
セッション情報		◎	◎	◎
ロック中セッション情報	△	○		
SQLの実行統計	◎			○
SQLの実行計画	△			

常時収集すべき稼働データ

MaxGauge
Database Performance Maximizer

- WHO　　：OSユーザー、DBユーザー
- WHERE　　：サーバー、クライアントマシン、ターミナル
- WHAT　　：プログラム、モジュール、SQL
- WHEN　　：ログインタイム、実行時刻
- HOW　　：性能統計、待機イベント、実行プラン

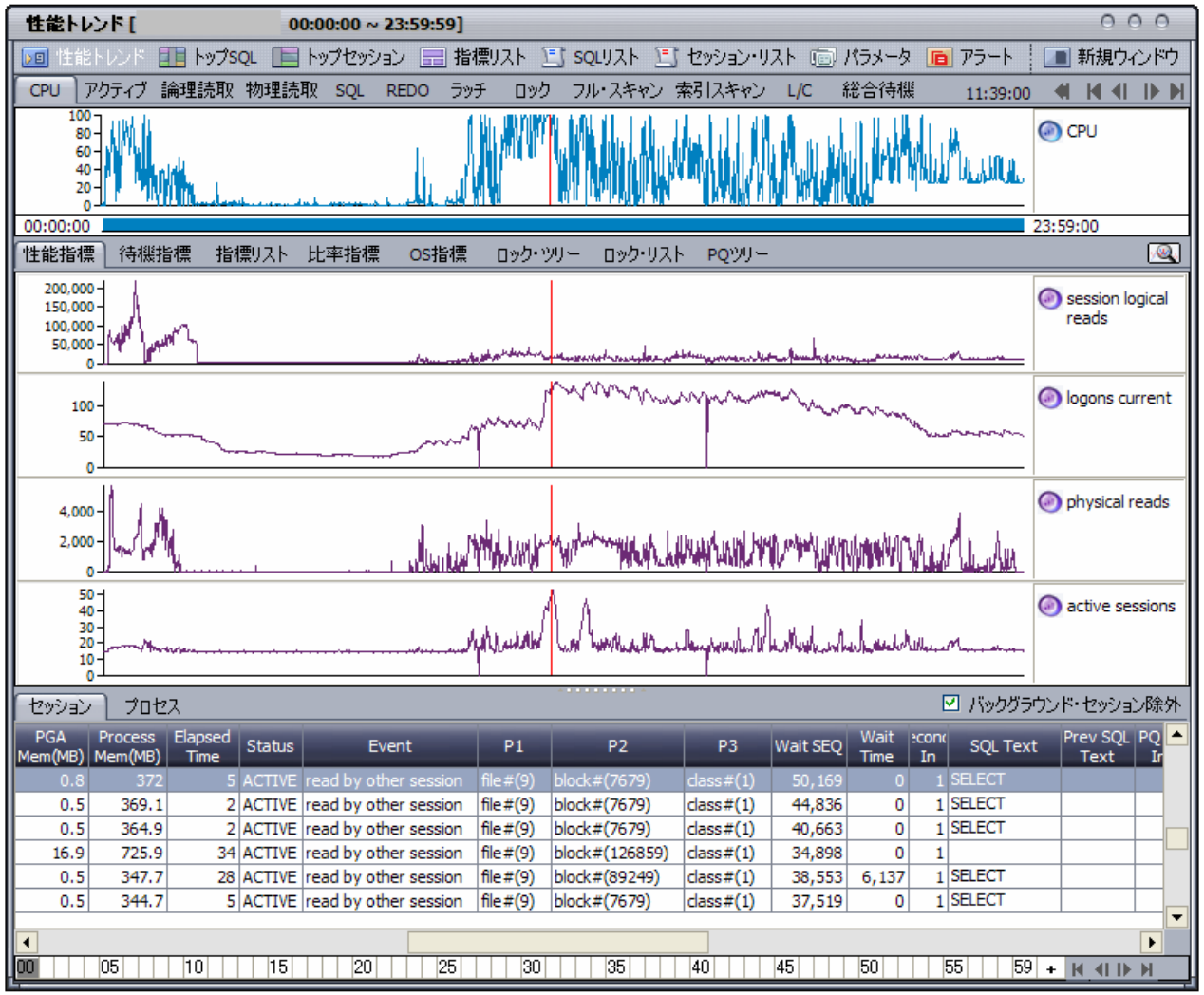
稼働ログ収集時のポイント

MaxGauge
Database Performance Maximizer

- ✓ システム/セッション/SQLレベル情報の収集
- ✓ データベースへの低い負荷
- ✓ 時系列による情報の収集
- ✓ データの精度：収集間隔（データの細かさ）
- ✓ 定常的な収集

稼動履歴データの例

MaxGauge
Database Performance Maximizer



時系列の性能指標、待機指標、OS指標

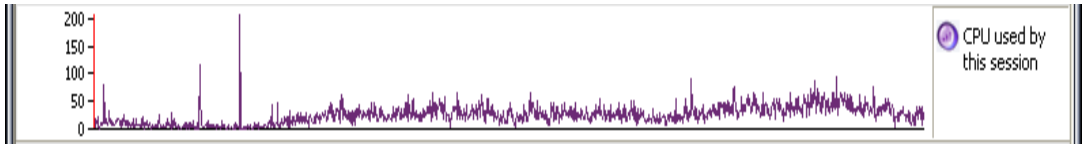
セッション情報、実行統計数値、プロセス情報、SQL情報

稼働ログ収集スクリプト

MaxGauge
Database Performance Maximizer

性能統計指標

```
set serveroutput on
declare
  fp    utl_file.file_type;
begin

  while ( 1 = 1 ) loop

    fp := utl_file.fopen('d:¥temp','instance_stats.csv','a');

    for rec in (
      select  to_char( sysdate, 'yyyy/mm/dd hh24:mi:ss' ) logging_time ,
        name ,
        value
      from    v$sysstat
    ) loop

      utl_file.put_line( fp , '="' || rec.logging_time || '",' || rec.name  || ',' || rec.value );

    end loop;

    utl_file.fclose(fp);

    dbms_lock.sleep (60); -- データ収集頻度:「1回/1分」推奨

  end loop;

exception
  when others
  then
    dbms_output.put_line('file output error' || to_char(sysdate, 'yyyy/mm/dd hh24:mi:ss') || sqlcode || ',' || sqlerrm) ;
end;
/
```



※ 出力データの編集サンプル

収集時刻	指標	値(0.01秒)	差分(0.01秒)
2009/01/10 12:04:58	CPU used by this session	4,667	978
2009/01/10 12:05:58	CPU used by this session	5,604	937
2009/01/10 12:06:58	CPU used by this session	6,430	826

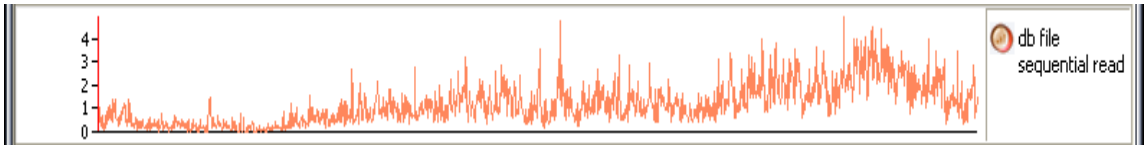
※ 適用の際は、データベースへの負荷、運用上のリスクを確認のうえ実施をお願いいたします。

稼働ログ収集スクリプト

MaxGauge
Database Performance Maximizer

待機指標

```
set serveroutput on
declare
  fp    utl_file.file_type;
begin

  while ( 1 = 1 ) loop
    fp := utl_file.fopen('d:¥temp','instance_waits.csv','a');

    for rec in (
      SELECT TO_CHAR( SYSDATE, 'yyyy/mm/dd hh24:mi:ss' ) logging_tim
           event ,
           time_waited
      FROM    v$system_event
      where   event not in (
                'ASM background timer',
                 … (中略)
                'watchdog main loop'
              )
    ) loop

      utl_file.put_line( fp , '="' || rec.logging_time || '",' || rec.event  || ',' || rec.time_waited );
    end loop;

    utl_file.fclose(fp);
    dbms_lock.sleep (60); -- データ収集頻度:「1回/1分」推奨
  end loop;

exception
  when others
  then
    dbms_output.put_line('file output error' || to_char(sysdate, 'yyyy/mm/dd hh24:mi:ss') || sqlcode || ',' || sqlerrm) ;
end;
```



※ 出力データの編集サンプル

収集時刻	指標	値(0.01秒)	差分(0.01秒)
2009/01/10 12:04:58	db file sequential read	18307	254
2009/01/10 12:05:58	db file sequential read	18329	22
2009/01/10 12:06:58	db file sequential read	18515	186

※ 適用の際は、データベースへの負荷、運用上のリスクを確認のうえ実施をお願いいたします。

稼働ログ収集スクリプト

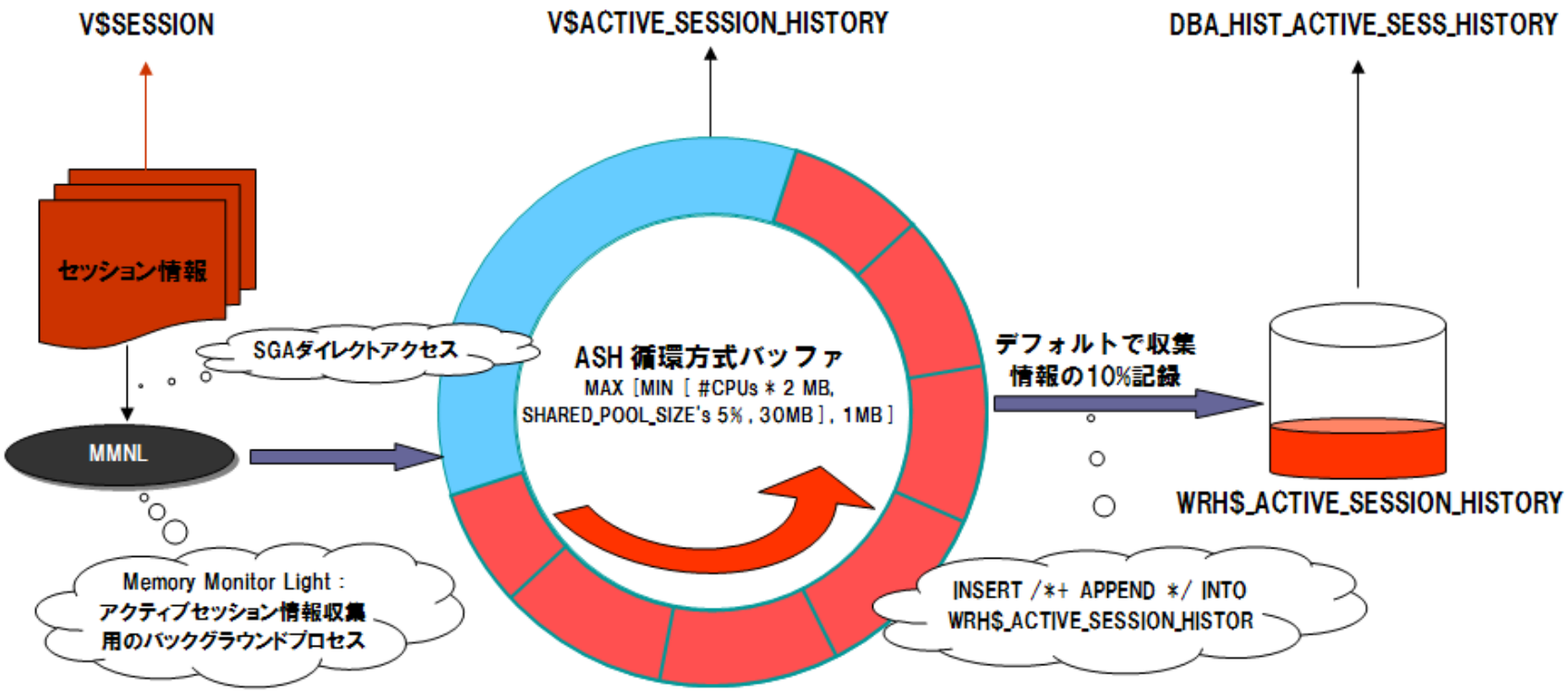
MaxGauge
Database Performance Maximizer

セッション情報

SELECT * FROM DBA_HIST_ACTIVE_SESS_HISTORY ;

SELECT * FROM V$ACTIVE_SESSION_HISTORY ;

Elapsed Time	Status	Event	P1	P2	P3	Wait SEQ	Wait Time	Seconds In Wait	SQL_Text	Pre T
3	ACTIVE	jobq slave wait	(0)	(0)	(0)	1	WAITING (0)	3	SELECT R....	
10	ACTIVE	jobq slave wait	(0)	(0)	(0)	4	WAITING (0)	10	SELECT R....	
1	ACTIVE	db file sequential read	file#(18)	block#(40781)	blocks(1)	59,378	WAITING (0)	1	SELECT P....	
10	ACTIVE	db file sequential read	file#(9)	block#(21865)	blocks(1)	62,369	WAITING (0)	1	SELECT S....	
7	ACTIVE	db file sequential read	file#(9)	block#(6487)	blocks(1)	43,263	WAITING (0)	1	SELECT CO...	
3	ACTIVE	db file sequential read	file#(33)	block#(46535)	blocks(1)	16,452	WAITING (0)	1	SELECT * F...	



※ 適用の際は、データベースへの負荷、運用上のリスクを確認のうえ実施をお願いいたします。

稼働ログ収集スクリプト

MaxGauge
Database Performance Maximizer

■ SQLと実行統計

```
set serveroutput on
declare
  fp    utl_file.file_type;
begin

  while ( 1 = 1 ) loop
    fp := utl_file.fopen('d:¥temp','sql_stats.csv','a');

    for rec in (
      SELECT TO_CHAR(SYSDATE, 'yyyy/mm/dd hh24:mi:ss') logging_time ,
             hash_value , address , elapsed_time , cpu_time , disk_reads , buffer_gets , executions
      FROM   v$sql
      WHERE  parsing_schema_id NOT IN (
                SELECT user_id
                FROM   dba_users
                WHERE  username IN (  'BI' , 'CTXSYS' , 'DBSNMP' , 'DMSYS' , 'EXFSYS' , 'HR' , 'IX' ,…(中略) 'SYSMAN','SYSTEM','SYS' )
             )
      AND    elapsed_time >= 10000
    ) loop

      utl_file.put_line( fp , '="' || rec.logging_time  || '",="' || rec.hash_value    || '",="' || rec.address    || '",' ||  rec.elapsed_time  || ',' ||
        rec.cpu_time    || ',' || rec.disk_reads    || ',' || rec.buffer_gets    || ',' || rec.executions );
    end loop;

    utl_file.fclose(fp);
    dbms_lock.sleep (600); -- データ収集頻度:「1回/10分」推奨

  end loop;
exception
  when others
  then
    dbms_output.put_line('file output error' || to_char(sysdate, 'yyyy/mm/dd hh24:mi:ss') || sqlcode || ',' || sqlerrm) ;
end;
/
```

※ 出力データの編集サンプル:SQL実行統計の収集結果と10分間の統計値(差分)

logging_time	hash_value	address	elapsed_time (1/1000000秒)	disk_reads (ブロック)	elapsed_time (差分:秒)	disk_reads (差分:ブロック)
2009/01/10 12:13:58	304208677	2AAE4744	4643534	0	4.450902	0
2009/01/10 12:23:58	304208677	2AAE4744	8760676	0	4.117142	0
2009/01/10 12:33:59	304208677	2AAE4744	13197174	0	4.436498	0

※ 適用の際は、データベースへの負荷、運用上のリスクを確認のうえ実施をお願いいたします。

MaxGaugeコンセプト

MaxGauge Database Performance Maximizer

障害解析・問題発見による品質向上

リアルタイム解析
障害をリアルタイムで状況把握
状況追跡とその場の即時対処

事後障害解析・問題発見
詳細な稼動情報を常時記録
障害状況をシミュレーション

MaxGauge Database Performance Maximizer

開発・運用での障害を確実に原因追及 ： コスト削減

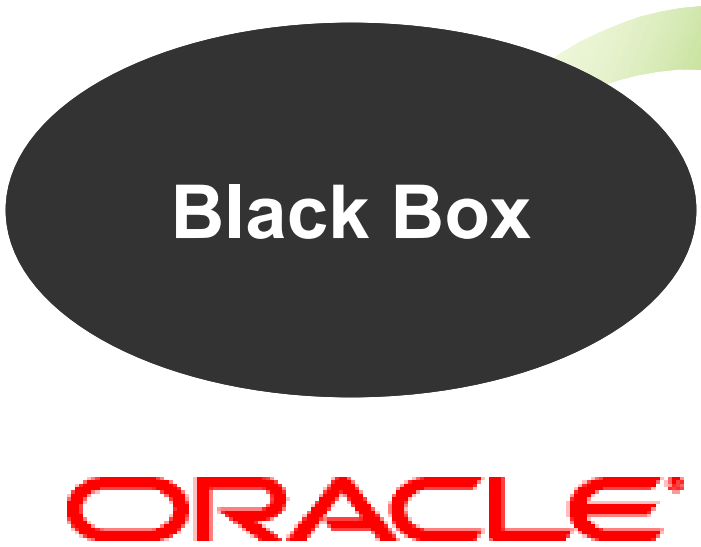
＋

情報の自動収集・GUIでの操作 ： 平準化・定型化

Oracleデータベース管理者の悩み

MaxGauge
Database Performance Maximizer

Oracleは、難しく内部状況も良くわからないのであまり触りたくない。という方が多いのでは？トラブルが発生しても内部がわからないため手探りでの調査をせざるを得ない状況を強いられています。監視やパフォーマンスチューニングは大事なのはわかるけれども、どのように行っていいのかもわからないという方が多いのが実状です。

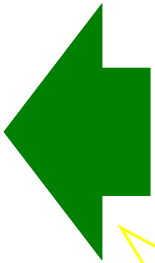


- エラーが出ても調査の方法がわからない。
- 誰が何時、何を行っているかですら把握できていない。
- パフォーマンスチューニングといっても、どこから手をつければいいのか？
- トラブルが発生した際、原因追及に非常に時間がかかってしまった。

従来の解析方法の問題点と限界

MaxGauge
Database Performance Maximizer

CPU/Memory
性能指標
待機指標
セッション状況
実行SQL文



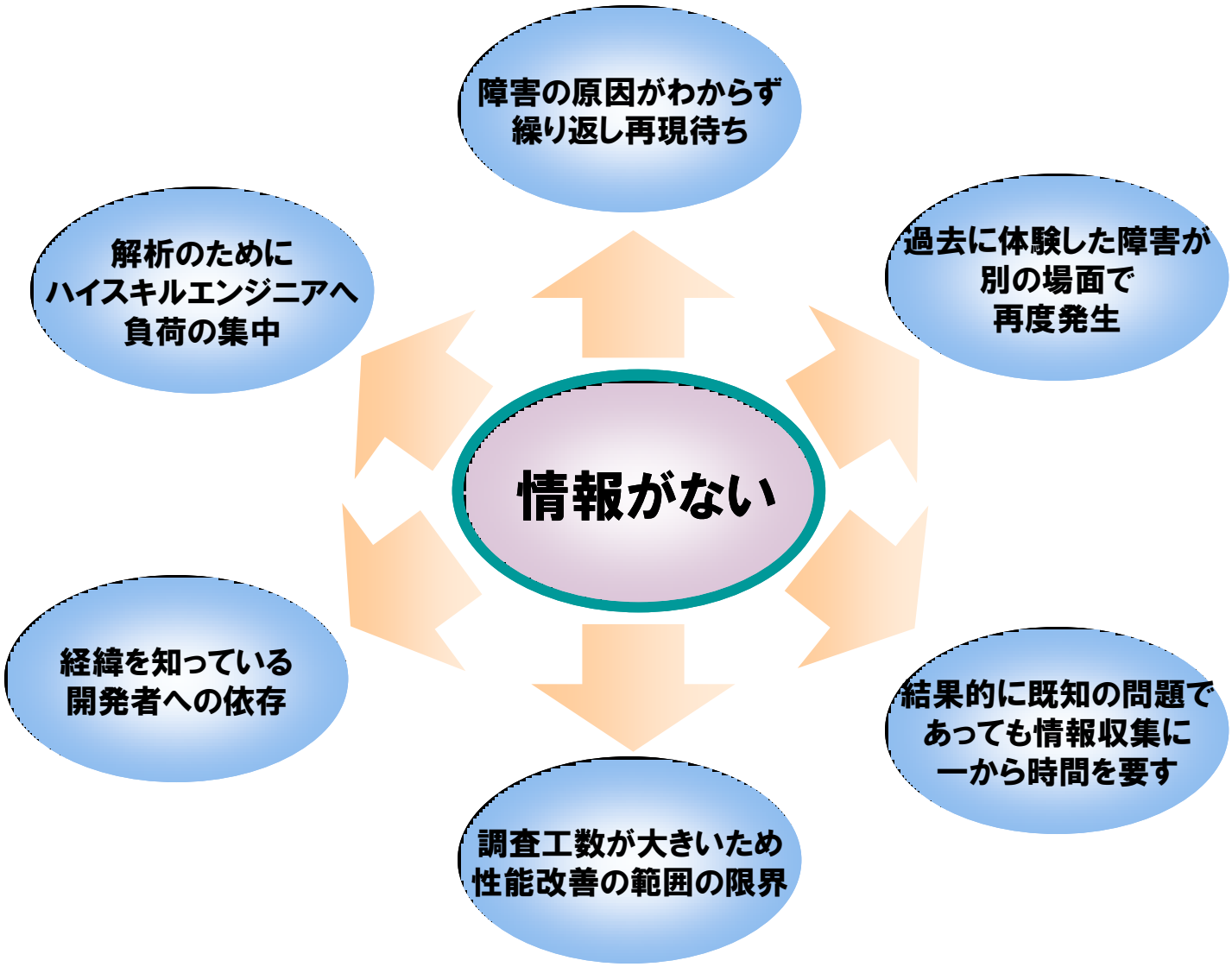
- SQL*Plus
- STATSPACK
- その他ツール　　など

SQLの発行による情報収集

- 障害分析のための情報はデータベース内部にあるため、SQLの発行により情報収集をかけていた
- データベースに負荷をかけるため、情報収集の量・頻度に限界があった
- 問題が発生してからの情報収集となってしまい、後手に回ることが多かった

情報がないことによる様々な影響

MaxGauge
Database Performance Maximizer



情報収集・分析効率向上ツール MaxGauge

MaxGauge
Database Performance Maximizer

MaxGaugeは、開発～テスト～運用での情報収集、分析効率を格段に向上させる、オラクルデータベースの『見える化』ツールです。

これまで、ハイスキルなエンジニアが、多くの工数やシステム負荷をかけて取得していた情報が自動で集計され、簡単に確認することが出来るようになります。

- **負荷をかけない**
  - 常時稼動情報を記録可能　24×365
  - ハングの際の状態が取れる
- **詳細に簡単に見れる**
  - 「誰が、いつ、何を」をGUIでマウス操作のみで確認
  - トレンドグラフと稼動セッション情報が連携しているため直感的に把握可能
  - Oracleの滞留状況から、それに該当するSQLが簡単にピックアップでき、渋滞を引き起こしているSQLがわかる
  - 実行計画も漏れなくとれ、実行計画の変化もすばやく把握
- **小回りがきく（Oracle Liteを採用）**
  - 稼動情報ログが簡単に移動できるため、リモートでの対応やトラブル対応部門へ簡単に送付可能
  - 導入時、データベース・サーバーの再起動不要で簡単

MaxGaugeが収集するデータ

MaxGauge
Database Performance Maximizer

SGA Direct Access

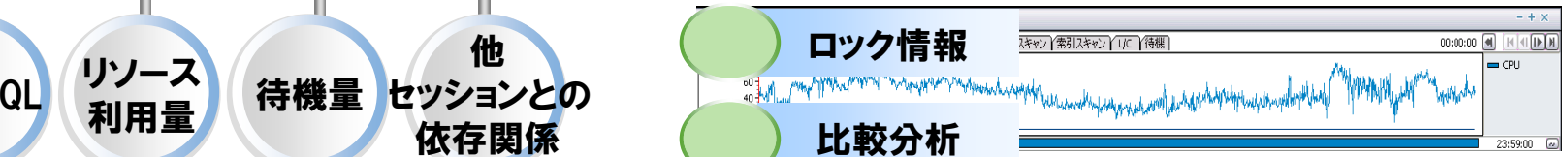
性能指標・待機指標 他(約1200種類 10g)
セッション・SQL稼動情報
実行SQLテキスト
他 OS 指標

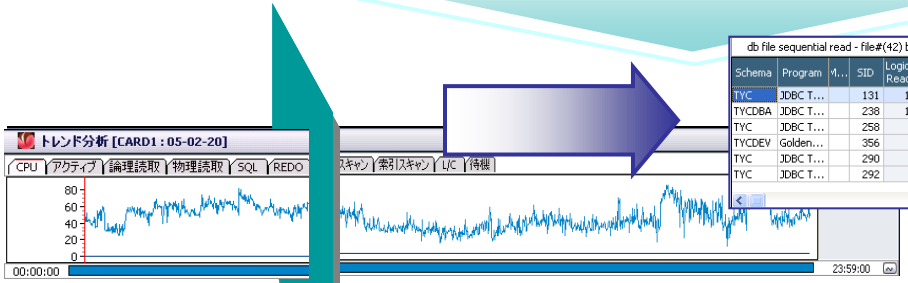
セッション情報

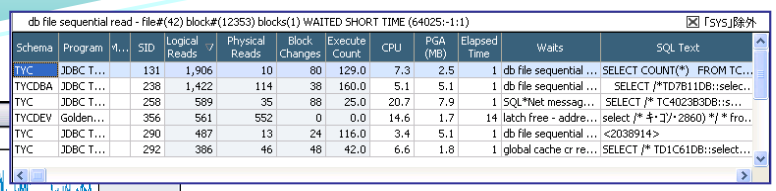
- 実行SQL
- リソース利用量
- 待機量
- 他セッションとの依存関係

時系列参照

- 各指標
- ロック情報
- 比較分析







一時点での断面を再現

MaxGaugeが収集するデータ

MaxGauge
Database Performance Maximizer

1分間隔ロギングデータ

性能統計指標、待機指標、O/S性能指標、トッププロセス情報

- オラクル性能及び待機情報を秒単位でロギングして1分間隔で保存します。
- システムのO/S性能情報を1分単位で保存します。
- システム全体のトップ・プロセス情報（プロセスID、プロセス名、アーギュメント、CPU使用率、メモリー使用率など）を保存します。

1秒間隔ロギングデータ

ロック情報、アクティブ・セッション情報

- ロックの発生履歴を秒単位で保存します。
- アクティブ・セッション情報を以下のように秒単位で保存します。

  セッションのユニック情報：sid、serial#、program、machine、DB user、OS user
  セッションの可変情報：module、action、SQL statement、elapse time
  リソース使用情報：CPU、PGA memory、logical reads、physical reads、SQL execution counts、block changes
  待機情報：wait event、sequence、wait time、seconds in wait、parameter1、parameter2、parameter3

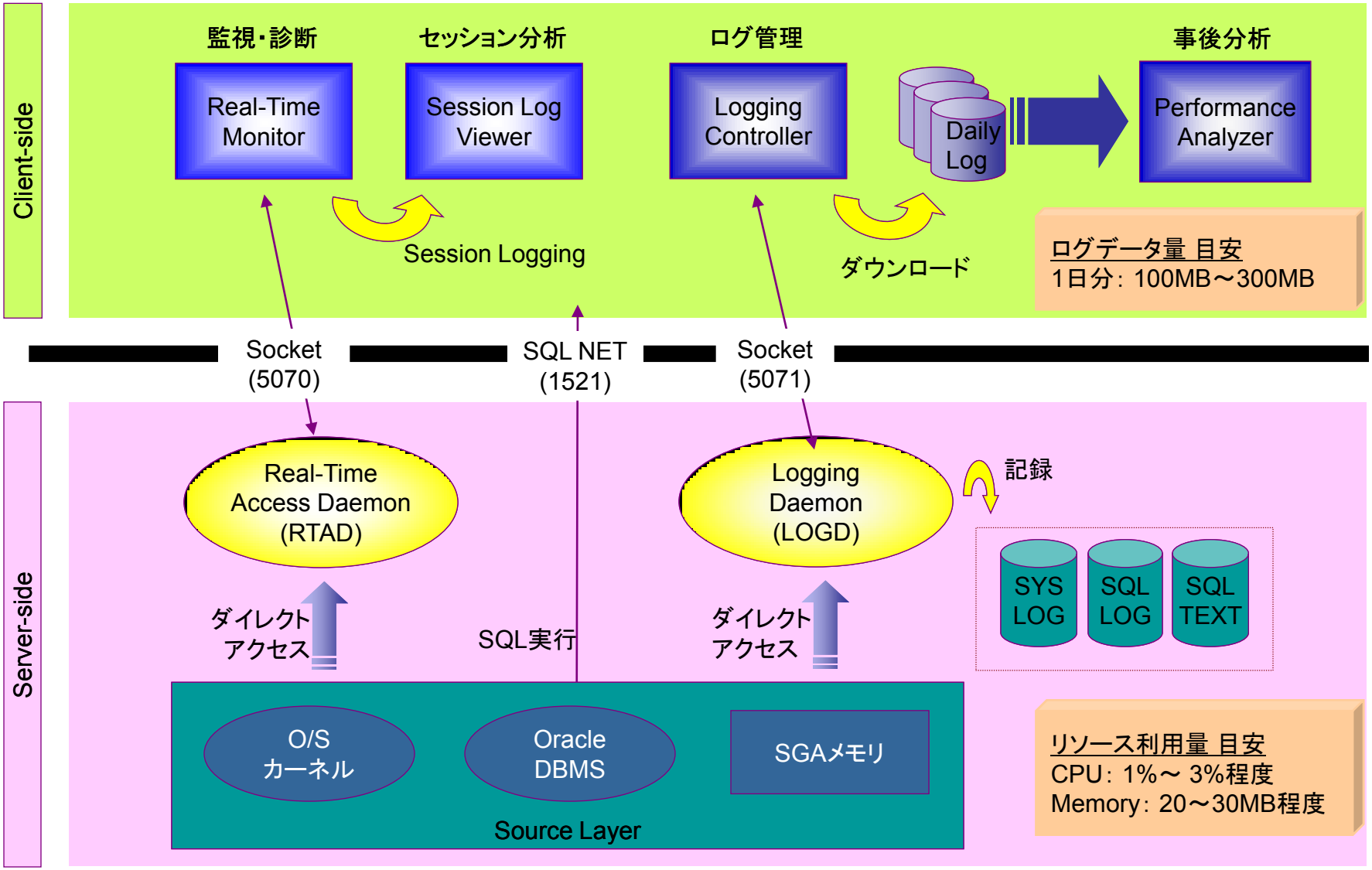
0.01秒間隔ロギングデータ

SQL遂行時の処理統計と待機情報

- リソース使用量情報：logical reads、physical reads、redo entries、block scan、row fetch、 row sort
- 実行計画（オプション）
- タイム情報：CPU time、Wait time、Elapsed time

マックスゲージの構成概要

MaxGauge
Database Performance Maximizer



MaxGaugeデモ

活用TIPの例

MaxGauge
Database Performance Maximizer

■ 開発

- AP処理(SQL)フローの追跡
- 長文のSQLを見やすくしたい(SQLの標準化)

■ ラッシュテスト

- 特定時間帯の上位SQLを確認したい
- 特定時間帯で特定SQLの実施統計を確認したい
- 高負荷のFULL TABLE SCANを行っているSQLを特定
- ユーザー定義情報を時系列で監視
- ユーザー定義情報のスナップショット取得
- ロックの発生状況を確認したい
- 特定時間帯の性能測定

■ 運用

- 性能低下階層の切り分け：DB or その他？
- 突然性能低下した、何から調べる？
- サポート依頼に必要なデータの抽出
- 接続数の変動を確認したい
- 接続エラーの回避
- CPU過負荷時の調査手順
- 「ORA-4031」対処 → リテラルSQL確認
- 過去の特定時刻実行計画を確認
- 実行計画が変わったSQLの確認
- SQLがアクセスしたオブジェクトを確認したい
- タイムアウトの原因SQLを特定

プロアクティブな定期点検

MaxGauge
Database Performance Maximizer

システムは生き物と同じく毎日刻々その稼働状況は変わっていきますので、積極的なシステム運用を行うために定期的にシステムの動きをきめ細かく観察する必要があります。例えば現場のシステム運用で以下のようなリクエストにはどう答えているでしょうか。

- 運用初期と比べ、実行時間が急増したSQLをピップアップしたい
- 特にユーザーよりクレームがないことは、今のシステム稼働状況は安定しているだろうか？
- 3ヶ月前と比べ、システム負荷はどれくらい高くなったか？
- 現行を維持すると、１年後でも何とか運用できるだろうか？
- 特定時間帯の負荷状況が大きく変わってないか？
- 実行計画が変わったSQLをリストアップしたいんだが。
- このSQLが犯人そうだけど、実行履歴を追跡してみようか。。。

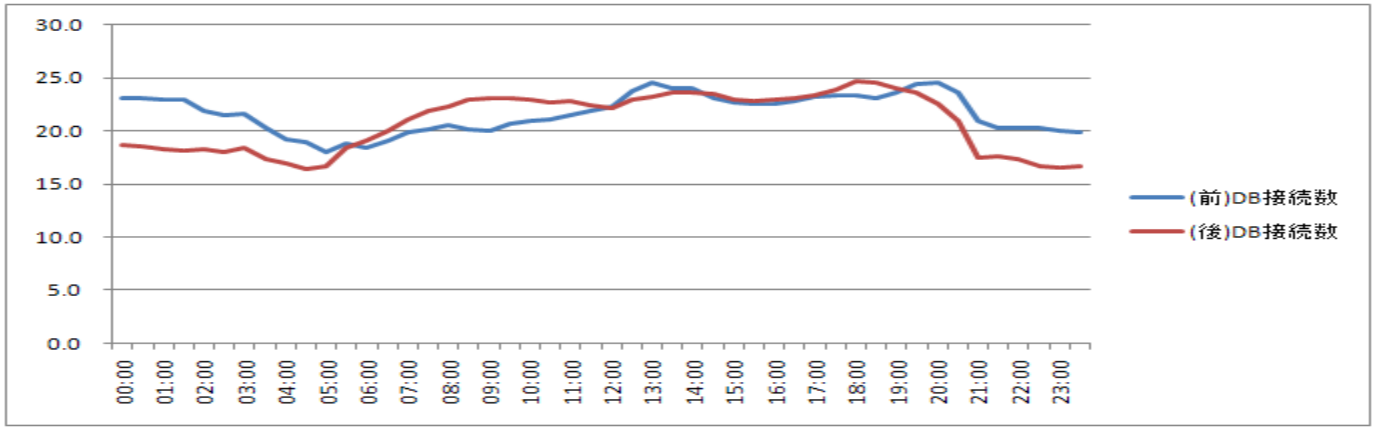
プロアクティブな定期点検

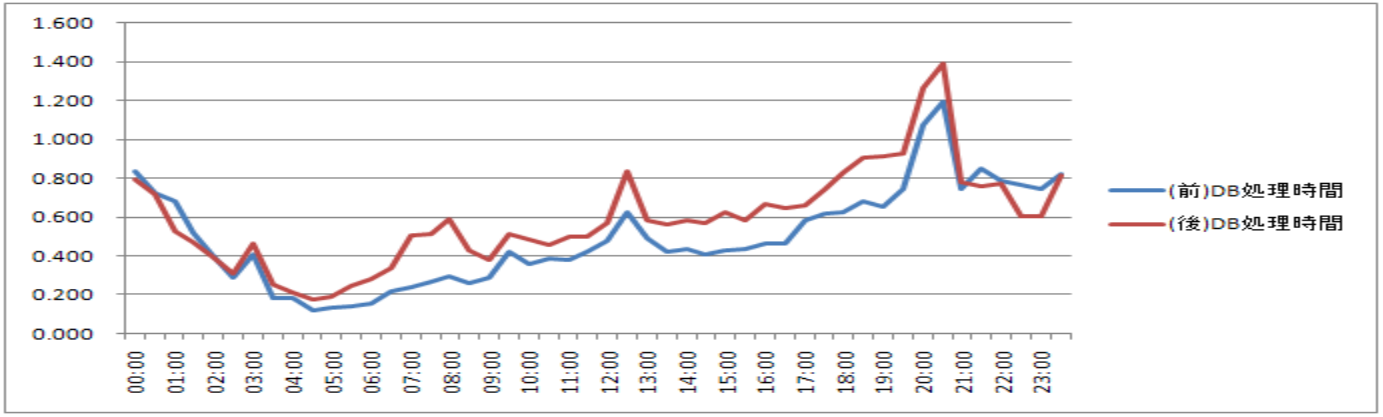
MaxGauge
Database Performance Maximizer

【24時間比較分析】

DB処理時間、DB接続数と変化率の24時間推移

2009/1/1 - 2009/1/31 vs. 2009/6/1 - 2009/6/30、金曜





プロアクティブな定期点検

MaxGauge
Database Performance Maximizer

【変動監視分析】

実行時間がN倍以上になったSQLとその実行統計及び実行計画

2009/1/1 - 2009/1/31 vs. 2009/6/23 - 2009/6/30

SQL_ID	増加率	(前)実行時間	(後)実行時間	実行回数	SQLテキスト
35E4D05C2A05A763693CF83F	571.6	1.3	745.9	163	
43A2345A21A7FE3B3FECBF89	2.3	247.7	572.1	1,468	
06FAC1FC4D6954D86907979A	2.2	252.0	560.9	250	
23E7E3123223900A2ECBB93D	2.0	277.3	556.0	157	
7E9923AE71A8CECF47FFBCAE	2.1	120.0	252.0	476	

プロアクティブな定期点検

MaxGauge
Database Performance Maximizer

【変動監視分析】

新しく上位SQL(処理時間)となったSQLとその実行統計(1日合計)及び実行計画

2009/1/1 - 2009/1/31 vs. 2009/6/23 - 2009/6/30

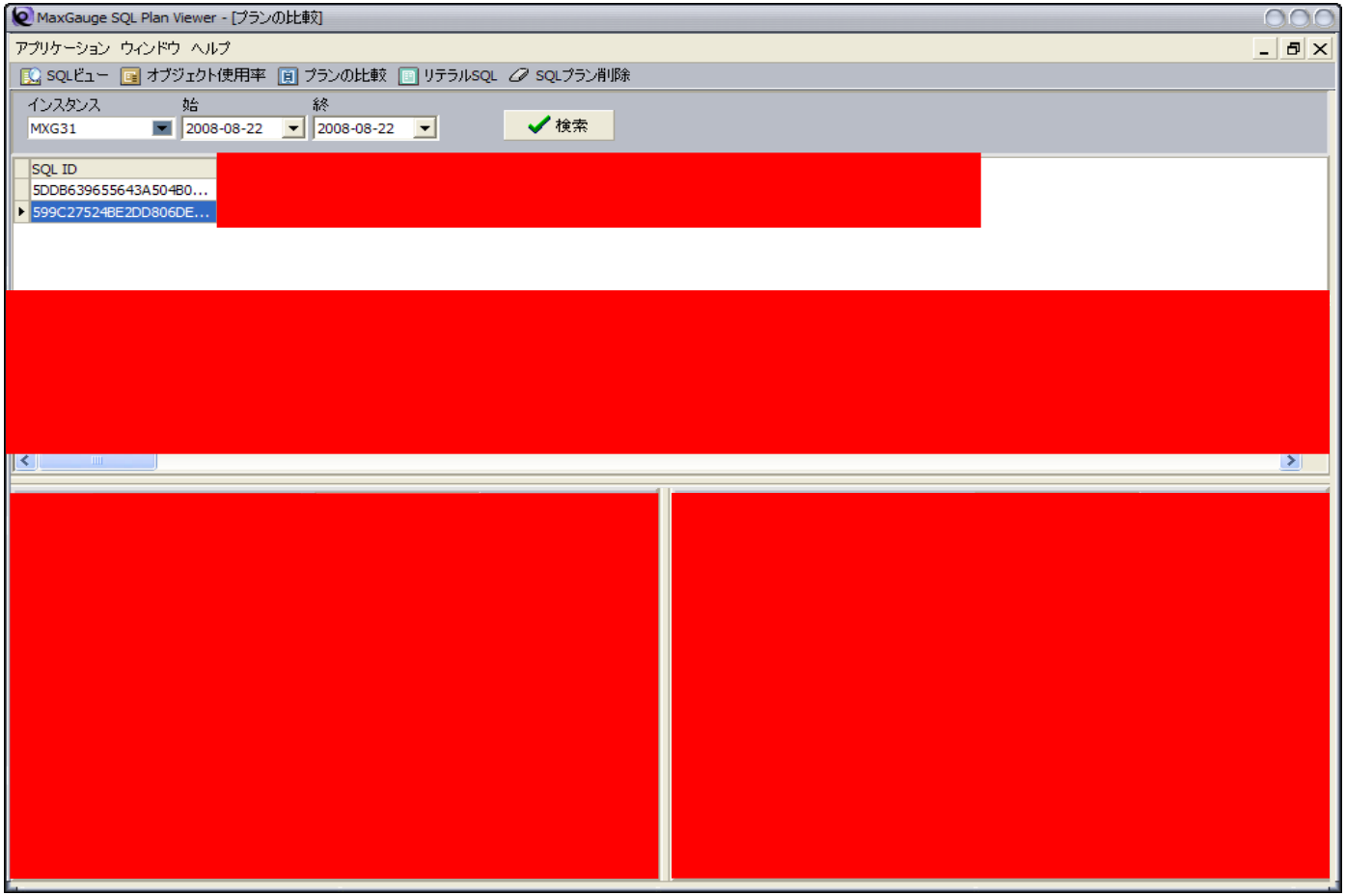
新上位SQLID	実行時間	CPU時間	待機時間	物理読取	論理読取	実行回数	新ランキング	元ランキング	SQLテキスト
061DE1C365CDF1B60D9FAC81	288,982.14	127,915.00	161,067.14	262,023	257,958,721	15,692	3		SELECT …
721B2F863329256207052702	203,112.14	126,998.57	76,113.57	141,399	248,805,026	14,492	10		SELECT …
721B2F863B9DB55A07052702	125,253.57	76,869.29	48,384.29	87,925	147,688,615	9,029	13		SELECT …
1190F8CC1323B7D92493E5E3	108,568.57	40,750.00	67,818.57	125,120	99,083,270	4,165	20		SELECT …
40BC185368D04DF11EA79EFD	107,391.43	102,469.29	4,922.14	7,582	183,210,848	9,478	21		SELECT …
0BF8743854E7A38B2D3FC146	104,180.00	101,536.43	2,643.57	4,525	187,183,003	8,897	23		SELECT …
7582645122B617CC190D6A36	91,866.43	31,751.43	60,115.00	84,361	60,301,388	4,805	29		SELECT …
5F74E6AE3E931C452C1DEE7A	76,103.57	3,020.00	73,083.57	205,359	5,549,970	57	32		SELECT …
5F74D7F053403E74663D90C0	75,097.14	43,762.86	31,334.29	69,248	100,074,287	4,266	33		SELECT …
35E4D05C2A05A763693CF83F	74,590.00	27.14	74,562.86	0	107	163	34	939	SELECT …
0BF8743878703E532D3FC146	63,427.86	61,757.86	1,670.00	2,598	112,467,298	5,501	36		SELECT …
327CBAB53E931C452C1DEE7A	58,114.29	4,791.43	53,322.86	142,325	8,863,115	103	40		SELECT …
47E05EB64A206CCD640EA155	50,663.57	11,852.86	38,810.71	59,400	15,832,161	2,794	47		SELECT …
5F74E6AE3E931C4513CA6F80	47,051.43	2,022.86	45,028.57	133,102	3,607,460	37	48		SELECT …
57D301A3186DE2FC12ACC9FA	46,793.57	12,060.71	34,732.86	55,571	29,092,310	1,533	49		SELECT …
5F74D7F05E7F68BC663D90C0	46,532.14	26,580.71	19,951.43	43,144	60,093,973	2,586	50		SELECT …
5F74E6AE3E931C45781970A8	43,117.86	1,845.00	41,272.86	108,705	3,315,014	39	53		SELECT …
327CBAB53E931C4513CA6F80	36,455.00	3,447.14	33,007.86	86,821	6,427,764	85	54		SELECT …
5F74E6AE3E931C451FF42EFD	35,410.71	1,542.14	33,868.57	97,129	2,691,209	30	58		SELECT …
327CBAB53E931C45781970A8	32,270.00	3,533.57	28,736.43	75,233	6,581,484	84	63		SELECT …
0ECEB4E6297A701376343965	31,651.43	11,788.57	19,862.86	31,390	14,058,355	2,614	65		SELECT …
5F15A2E135531625655F2780	27,820.71	25,785.00	2,035.71	2,938	40,470,129	2,754	70		SELECT …
327CBAB53E931C451FF42EFD	26,720.00	2,684.29	24,035.71	62,170	4,814,474	65	74		SELECT …
7F22594F3D17FB086D7DB793	26,286.43	9,043.57	17,242.86	29,185	9,649,599	1,481	75		SELECT …
5F74E6AE3E931C45106CEFA2	26,071.43	1,346.43	24,725.00	67,916	2,382,001	29	76		SELECT …
7E9923AE71A8CECF47FFBCAE	25,200.00	636.43	24,563.57	38,816	173,022	476	77	115	SELECT …
661AC99968ED0B2B779CBB2D	23,859.29	23,800.00	59.29	108	90,140,418	131	80	187	SELECT …
3813C3E43133165937BD437E	22,381.43	1,157.86	21,223.57	49,931	1,259,485	75	83		SELECT …
7E9923AE72FB66D80FE08C5D	22,285.00	989.29	21,295.71	41,646	257,804	696	84	133	SELECT …

プロアクティブな定期点検

MaxGauge
Database Performance Maximizer

【変動監視分析】

実行計画が変更されたSQLとその実行統計及び実行計画



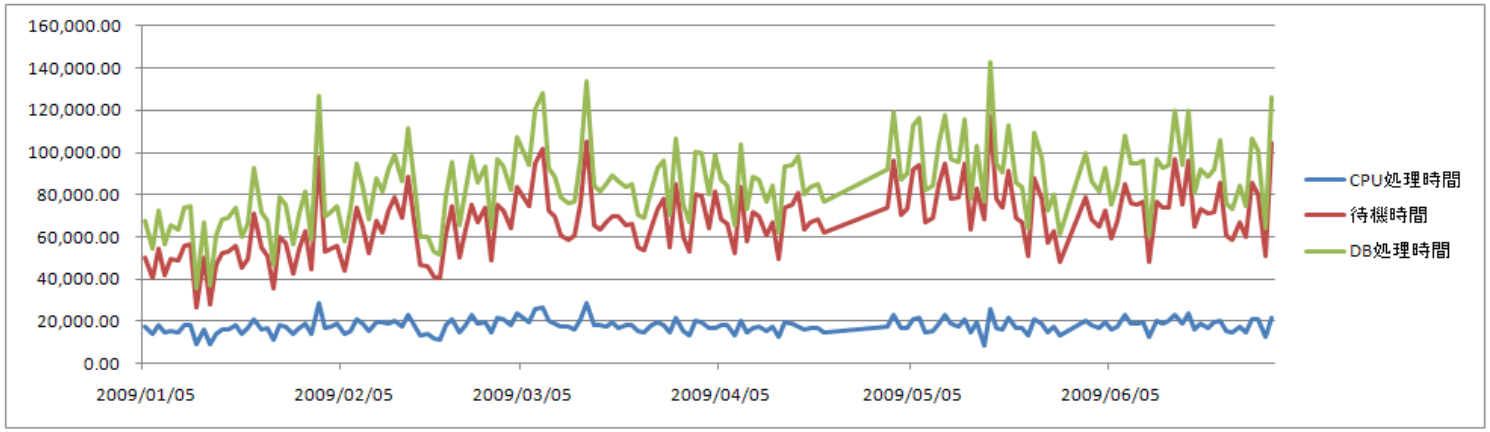
プロアクティブな定期点検

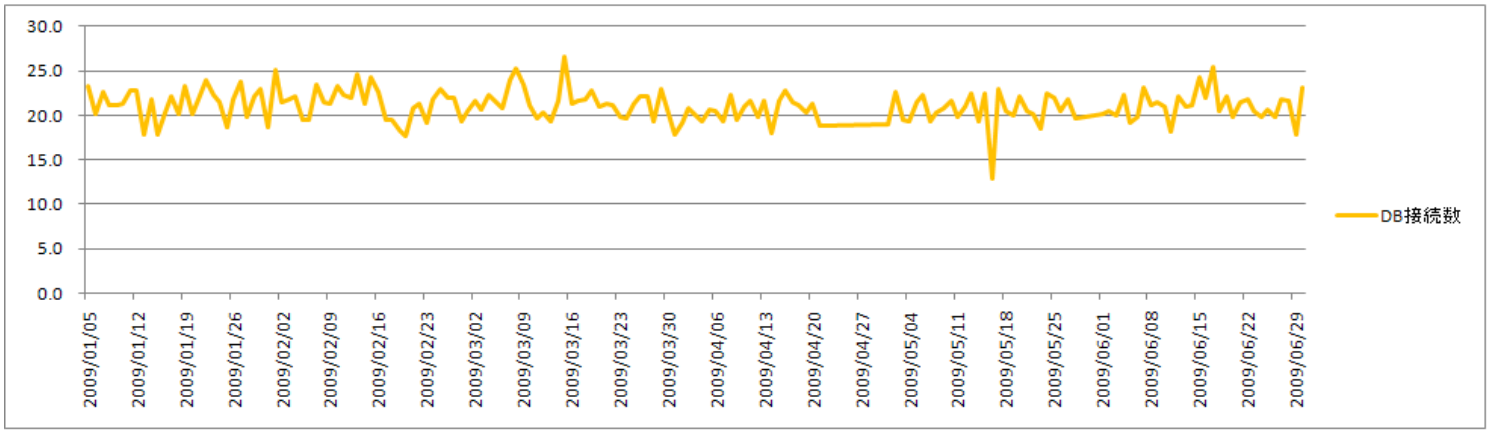
MaxGauge
Database Performance Maximizer

【長期推移分析】

DB処理時間、DB接続数(1日基準)の長期推移

2009/1/1 - 2009/6/30、24時間合計





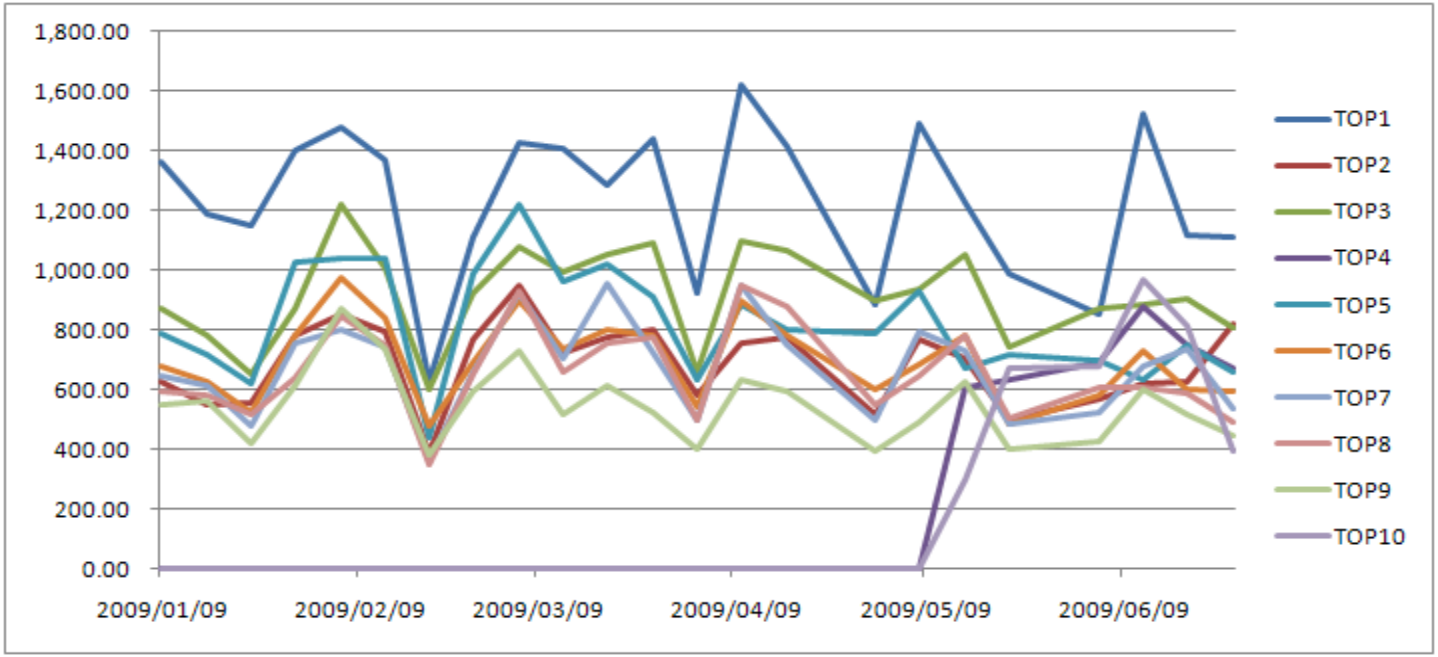
プロアクティブな定期点検

MaxGauge
Database Performance Maximizer

【長期推移分析】

上位SQL(集中時間帯基準)の合計実行時間の長期推移

2009/1/1 - 2009/6/30 、金曜、19:00-23:00時間帯の合計

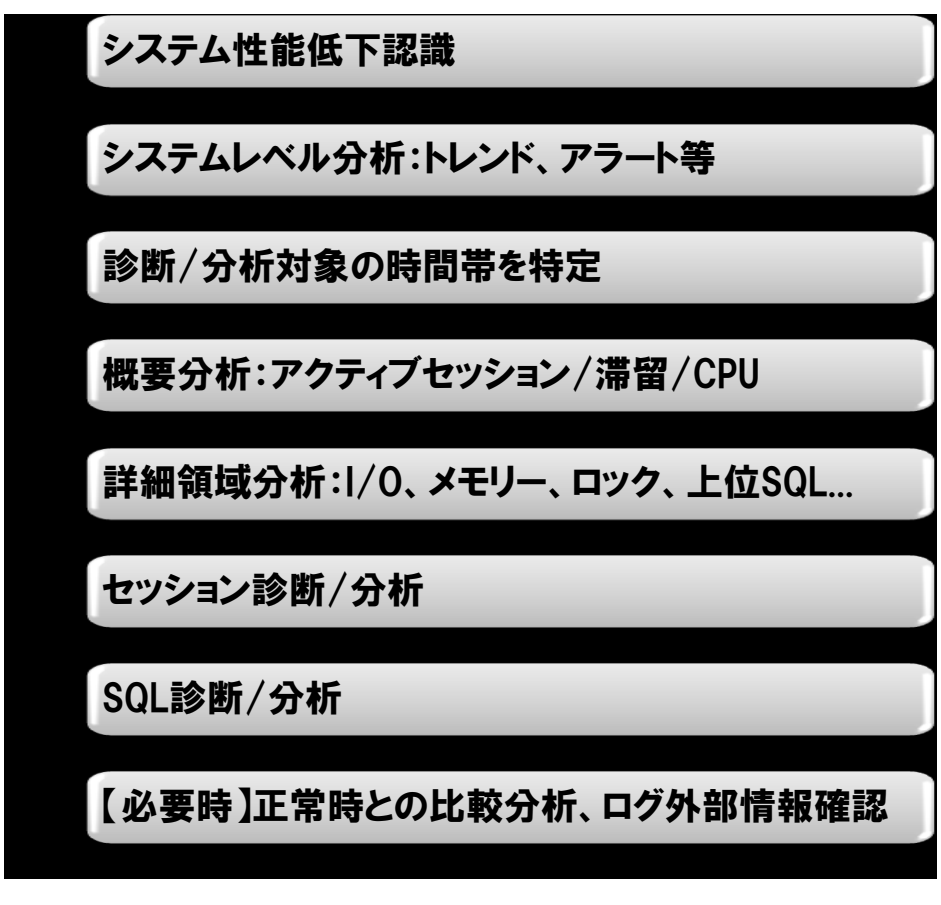


日付	TOP1	TOP2	TOP3	TOP4	TOP5	TOP6	TOP7	TOP8	TOP9	TOP10
2009/1/9	1,365.35	627.35	870.00	0.00	791.10	678.40	646.40	592.90	551.00	0.00
2009/1/16	1,190.40	551.55	782.70	0.00	720.40	627.70	614.55	581.15	564.30	0.00
2009/1/23	1,151.05	554.35	650.95	0.00	619.35	526.80	476.75	516.80	422.55	0.00
2009/1/30	1,404.95	784.95	874.40	0.00	1,028.60	782.70	756.85	642.35	611.60	0.00
2009/2/6	1,478.80	856.35	1,223.40	0.00	1,038.90	973.25	803.95	847.35	870.00	0.00
2009/2/13	1,371.55	796.10	1,007.25	0.00	1,040.15	843.25	742.50	752.05	738.70	0.00

障害解析の事例

MaxGauge
Database Performance Maximizer

記録しているデータベース全体の状況、アクセスしているセッション、詳細なSQLを情報としてリンク。クリック操作で段階的にドリルダウンをするような感覚で動きを追っていくことが出来ます。



セッション情報を網羅的に記録・参照できるツールは少なく、MaxGaugeと同様の追跡は難しい。

障害解析の事例

MaxGauge
Database Performance Maximizer

原因不明の(OS)データベース再起動問題の解決 (原因トリガーの追跡と対応)

【事象】

RAC環境において、突然Oracleクラスタウェアが、データベース停止状況であると判断し、データベースの再起動をかけてしまっていた。

【解決策】

MaxGaugeのログより、再起動時点でのデータベース内処理状況より、同一ブロックへの大量な「DELETE」、「INSERT」処理がCPUを100%を使い切って、滞留が急増していることを確認。そのため、「システムが過負荷状態 → Oracleクラスタウェアの処理(ハートビート送信)が完了出来ない →データベース停止(ハング)と認識→OS再起動」発生。対象SQLのチューニングにより、データベース負荷を軽減。データベース再起動の事象の発生を抑えた。

【効果】

MaxGaugeのログ調査以前、システム開発担当者、及びDBA担当にて原因調査を約2週間行っていたが、原因がつかめなかった。MaxGaugeのログの参照により、約2時間で障害時の状況の把握と対象SQLのピックアップを行い、顧客へレポート。

(クラスタウェアが、データベースを再起動させてしまう事象については、製品仕様の確認のため、オラクルサポート担当とのやり取りは継続)

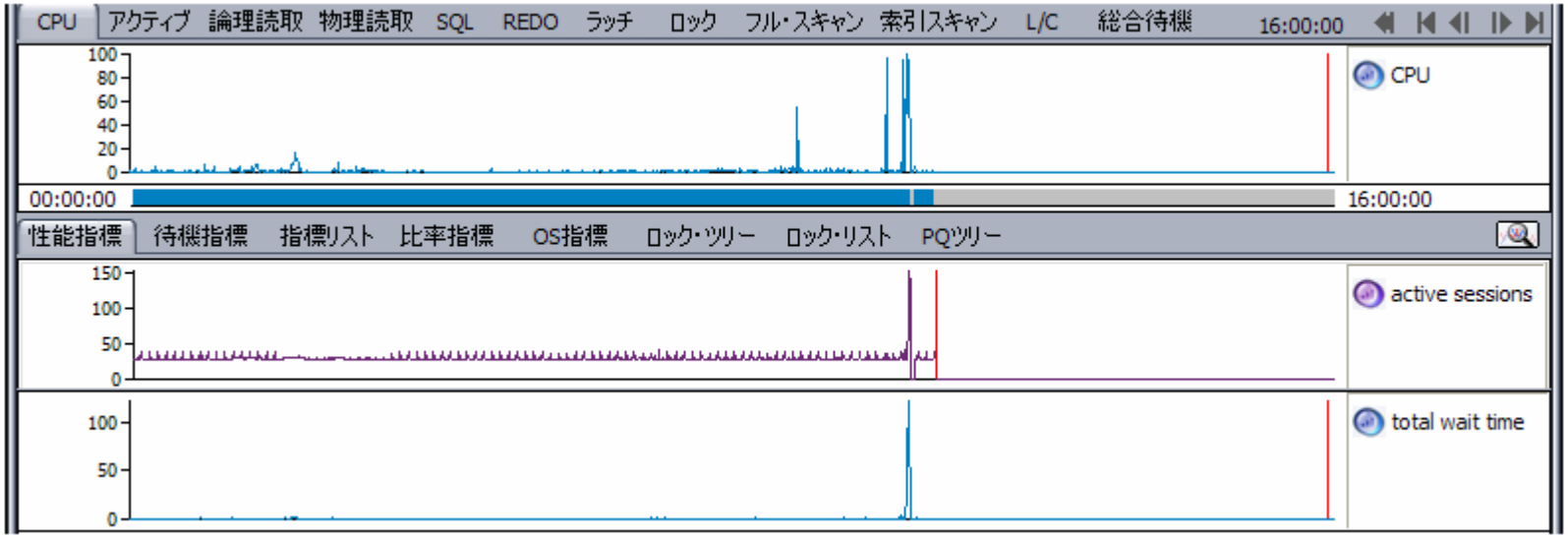
障害解析の事例

MaxGauge
Database Performance Maximizer

現象

Oracle clsomon failed with fatal status 13.
Oracle CRS failure. Rebooting for cluster integrity.

システムレベル診断/分析



・15:30前後で、CPU使用率が100%ほど使用されている。

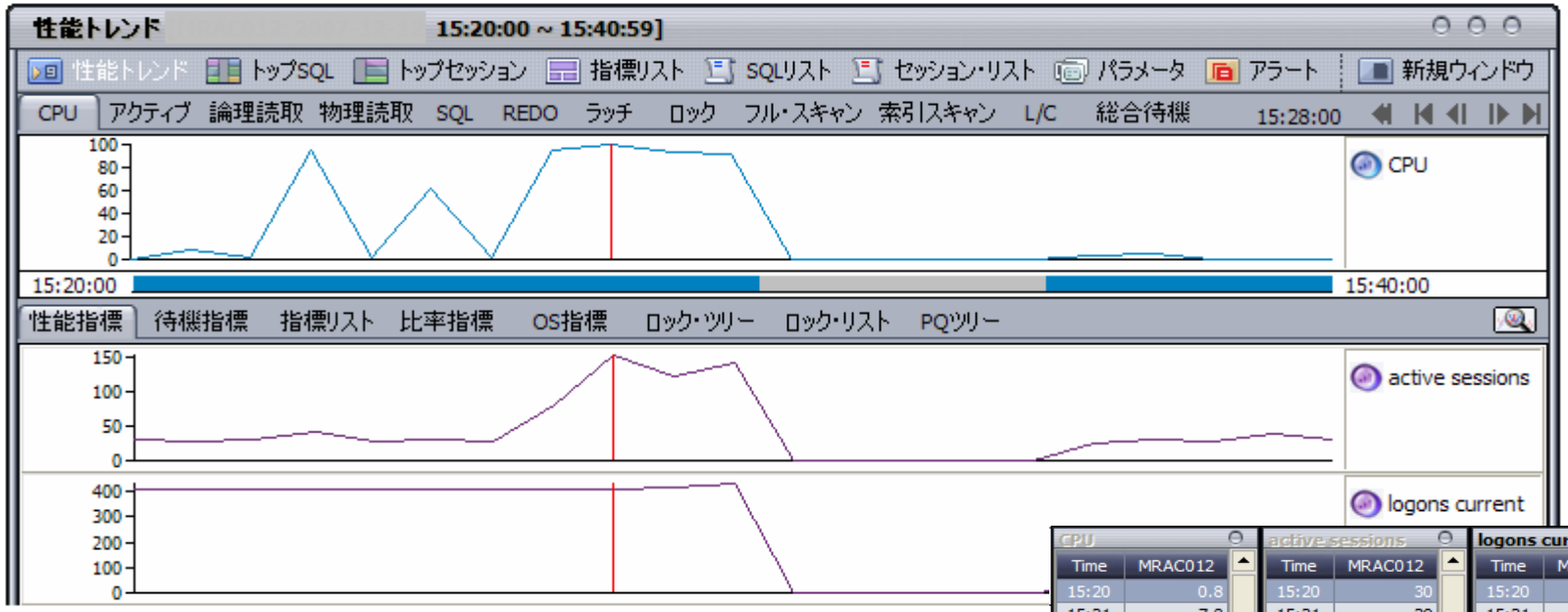
・同時間帯で、アクティブセッションが30前後から150まで急増した。

・同時間帯で、数秒程度の滞留が100秒以上まで急増した。

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定(異常現象発生)時間帯の詳細分析



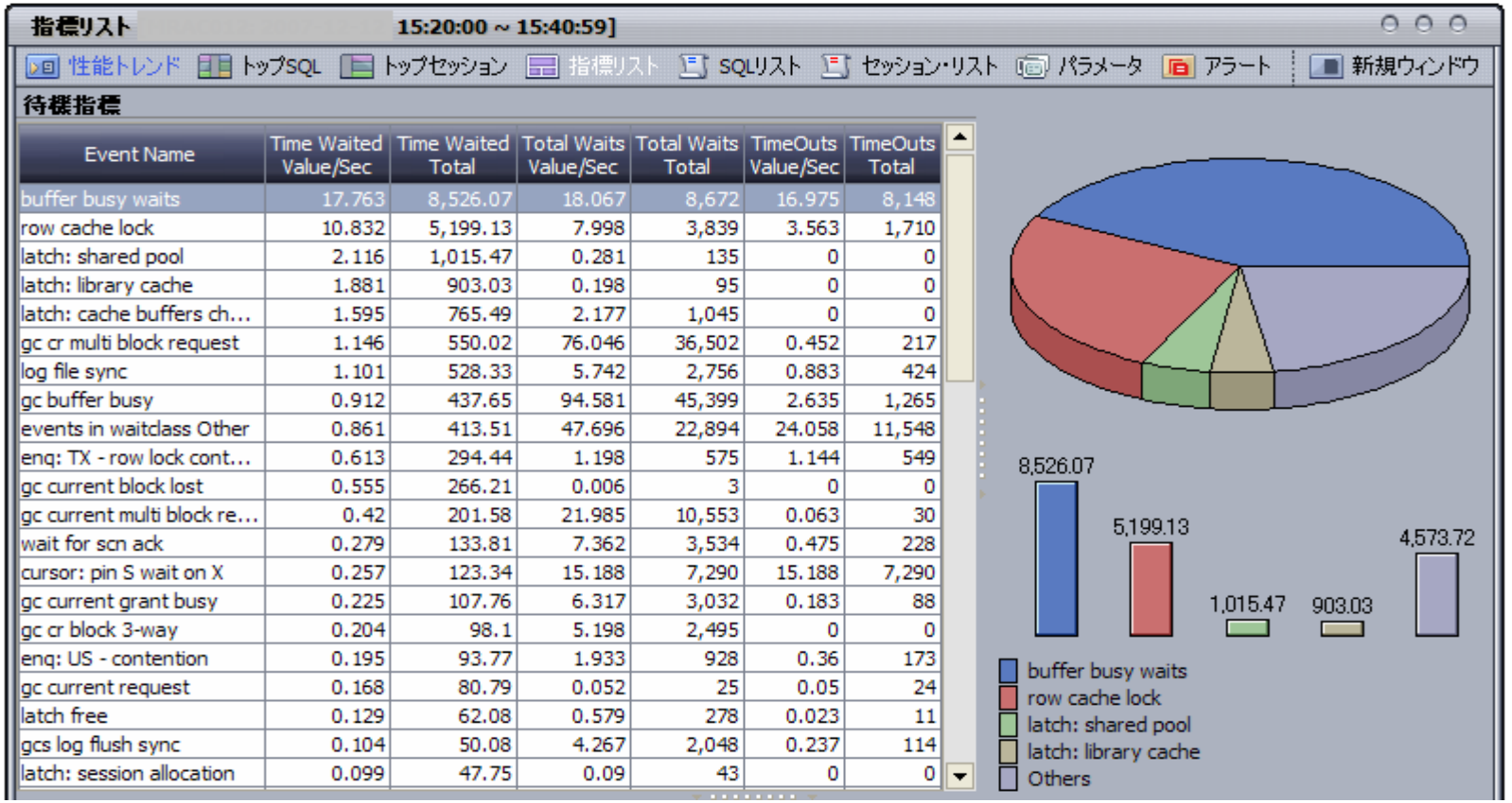
CPU		active sessions		logons current	
Time	MRAC012	Time	MRAC012	Time	MRAC012
15:20	0.8	15:20	30	15:20	411
15:21	7.8	15:21	29	15:21	410
15:22	1.5	15:22	30	15:22	410
15:23	94.8	15:23	42	15:23	411
15:24	1.5	15:24	29	15:24	410
15:25	61.5	15:25	30	15:25	411
15:26	1.7	15:26	29	15:26	410
15:27	95	15:27	82	15:27	412
15:28	100	15:28	153	15:28	412
15:29	92.8	15:29	122	15:29	420
15:30	91.1	15:30	143	15:30	432
15:31	0	15:31	0	15:31	0
15:32	0	15:32	0	15:32	0
15:33	0	15:33	0	15:33	0
15:34	0	15:34	0	15:34	0
15:35	0	15:35	0	15:35	0
15:36	3.2	15:36	25	15:36	25
15:37	4.7	15:37	30	15:37	36

- 「15:20～15:30」間で、接続数が「411→432」増加
- 同時間帯で、アクティブセッションが「30→153」増加、15:28ピーク
- CPU使用率は、「15:28」で100%になり、続いて過負荷状態

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定(異常現象発生)時間帯の詳細分析



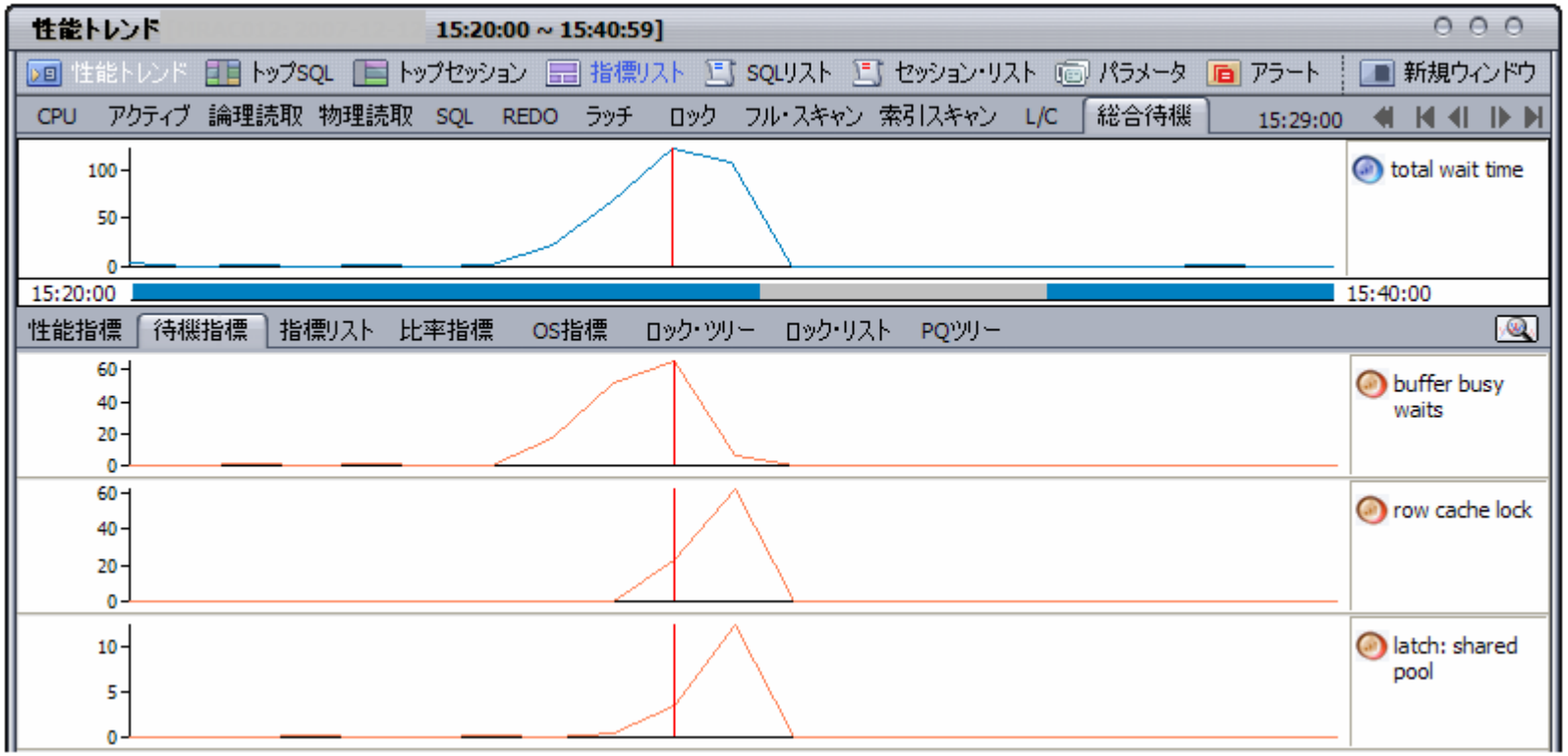
Event Name	Time Waited Value/Sec	Time Waited Total	Total Waits Value/Sec	Total Waits Total	TimeOuts Value/Sec	TimeOuts Total
buffer busy waits	17.763	8,526.07	18.067	8,672	16.975	8,148
row cache lock	10.832	5,199.13	7.998	3,839	3.563	1,710
latch: shared pool	2.116	1,015.47	0.281	135	0	0
latch: library cache	1.881	903.03	0.198	95	0	0
latch: cache buffers ch...	1.595	765.49	2.177	1,045	0	0
gc cr multi block request	1.146	550.02	76.046	36,502	0.452	217
log file sync	1.101	528.33	5.742	2,756	0.883	424
gc buffer busy	0.912	437.65	94.581	45,399	2.635	1,265
events in waitclass Other	0.861	413.51	47.696	22,894	24.058	11,548
enq: TX - row lock cont...	0.613	294.44	1.198	575	1.144	549
gc current block lost	0.555	266.21	0.006	3	0	0
gc current multi block re...	0.42	201.58	21.985	10,553	0.063	30
wait for scn ack	0.279	133.81	7.362	3,534	0.475	228
cursor: pin S wait on X	0.257	123.34	15.188	7,290	15.188	7,290
gc current grant busy	0.225	107.76	6.317	3,032	0.183	88
gc cr block 3-way	0.204	98.1	5.198	2,495	0	0
enq: US - contention	0.195	93.77	1.933	928	0.36	173
gc current request	0.168	80.79	0.052	25	0.05	24
latch free	0.129	62.08	0.579	278	0.023	11
gcs log flush sync	0.104	50.08	4.267	2,048	0.237	114
latch: session allocation	0.099	47.75	0.09	43	0	0

- 「15:20～15:40」間で、上位滞留は「buffer busy waits、row cache lock、...」順で比較的に多数の滞留が発生
- 「buffer busy waits」は同じブロックに対する競合現象で、全体の42%以上の滞留を占めている

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定(異常現象発生)時間帯の詳細分析

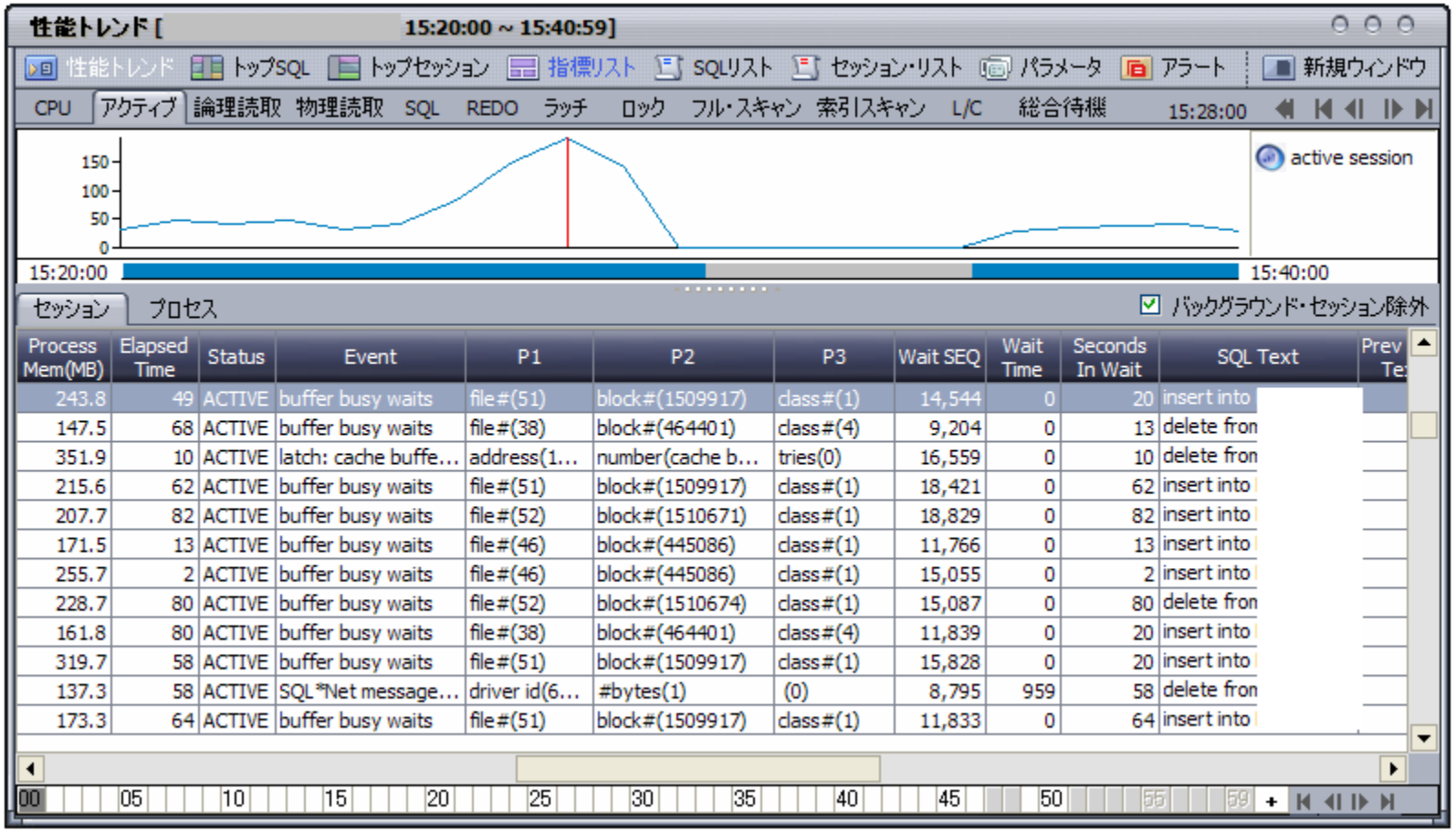


- 最初に「buffer busy waits」滞留が現れ、他の滞留はその後の性能低下の悪循環で現れたようにみえる。

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定(異常現象発生)時間帯のセッション分析



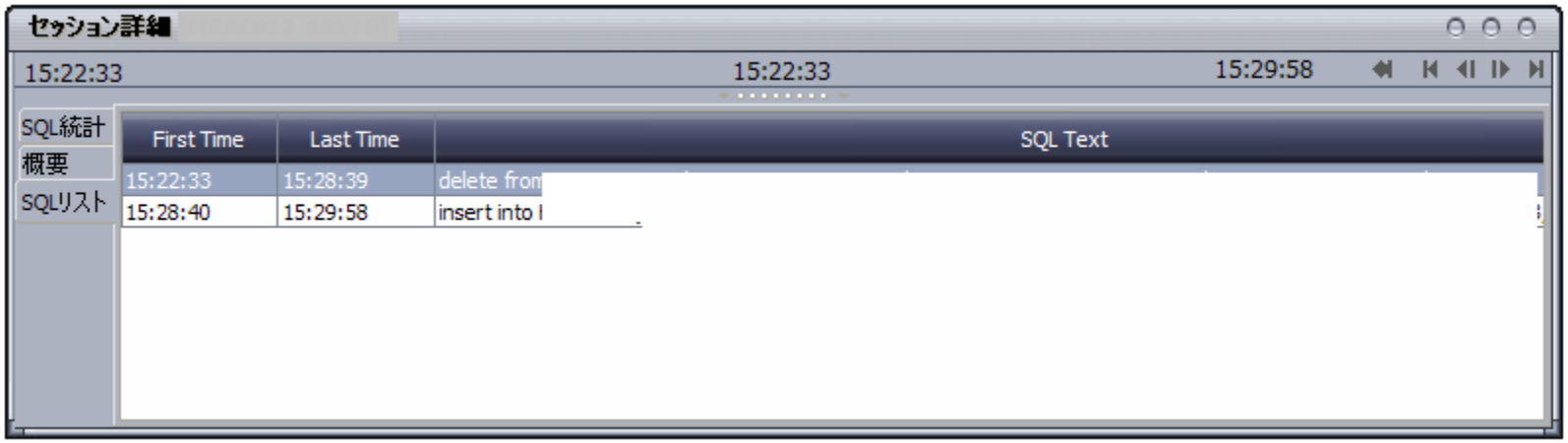
Process Mem(MB)	Elapsed Time	Status	Event	P1	P2	P3	Wait SEQ	Wait Time	Seconds In Wait	SQL Text	Prev Te
243.8	49	ACTIVE	buffer busy waits	file#(51)	block#(1509917)	class#(1)	14,544	0	20	insert into	
147.5	68	ACTIVE	buffer busy waits	file#(38)	block#(464401)	class#(4)	9,204	0	13	delete fron	
351.9	10	ACTIVE	latch: cache buffe...	address(1...	number(cache b...	tries(0)	16,559	0	10	delete fron	
215.6	62	ACTIVE	buffer busy waits	file#(51)	block#(1509917)	class#(1)	18,421	0	62	insert into	
207.7	82	ACTIVE	buffer busy waits	file#(52)	block#(1510671)	class#(1)	18,829	0	82	insert into	
171.5	13	ACTIVE	buffer busy waits	file#(46)	block#(445086)	class#(1)	11,766	0	13	insert into	
255.7	2	ACTIVE	buffer busy waits	file#(46)	block#(445086)	class#(1)	15,055	0	2	insert into	
228.7	80	ACTIVE	buffer busy waits	file#(52)	block#(1510674)	class#(1)	15,087	0	80	delete fron	
161.8	80	ACTIVE	buffer busy waits	file#(38)	block#(464401)	class#(4)	11,839	0	20	insert into	
319.7	58	ACTIVE	buffer busy waits	file#(51)	block#(1509917)	class#(1)	15,828	0	20	insert into	
137.3	58	ACTIVE	SQL*Net message...	driver id(6...	#bytes(1)	(0)	8,795	959	58	delete fron	
173.3	64	ACTIVE	buffer busy waits	file#(51)	block#(1509917)	class#(1)	11,833	0	64	insert into	

- アクティブセッションの中「108」セッションが「HISTTBL」表に対する「DELETE、INSERT」作業を行っている。

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定(異常現象発生)時間帯のセッション分析

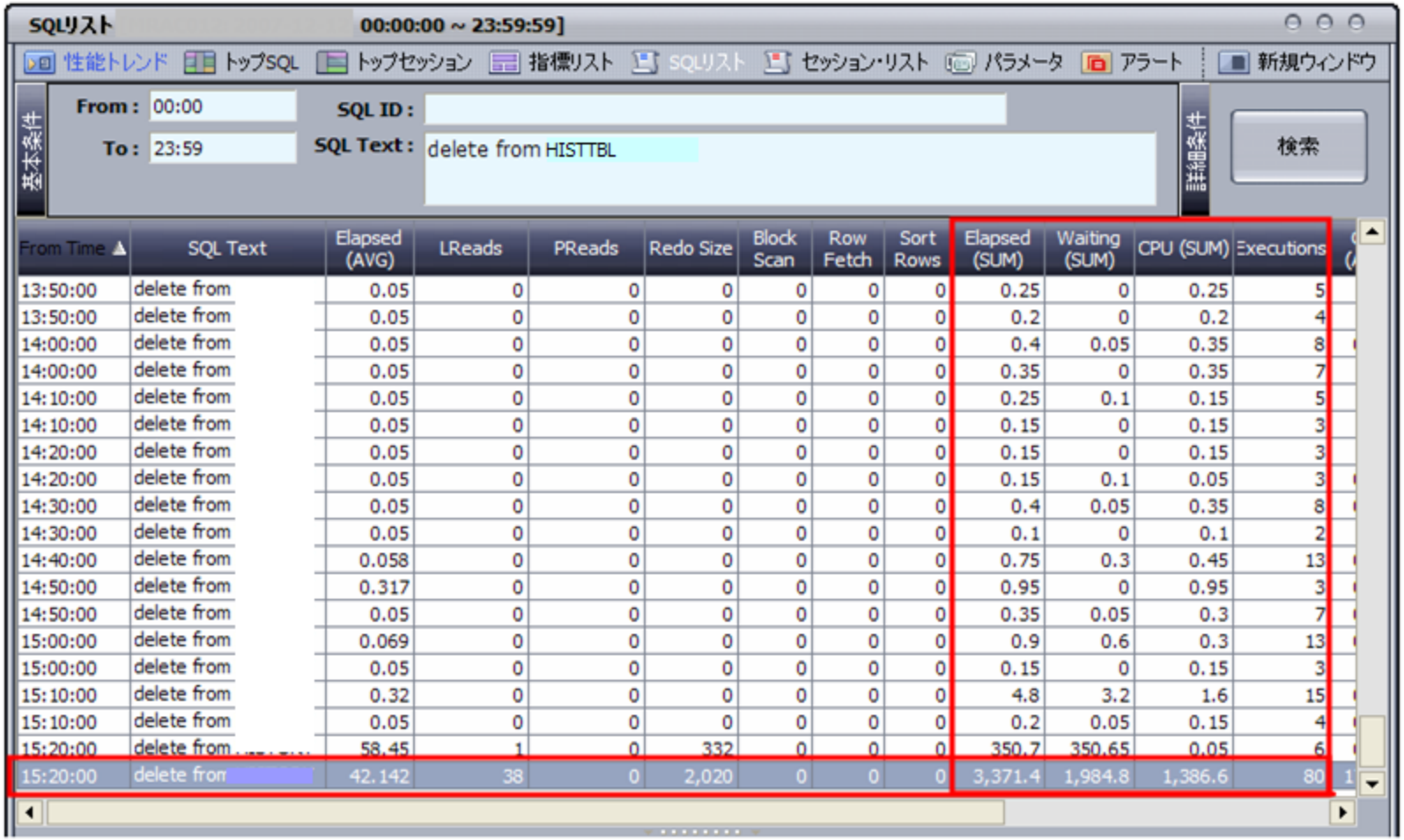


- 「HISTTBL」表に対する「DELETE、INSERT」を実施しているセッションの履歴(詳細)を確認すると、「DELETE=6:36」後、「INSERT=1:17」を発行している。

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定SQL分析



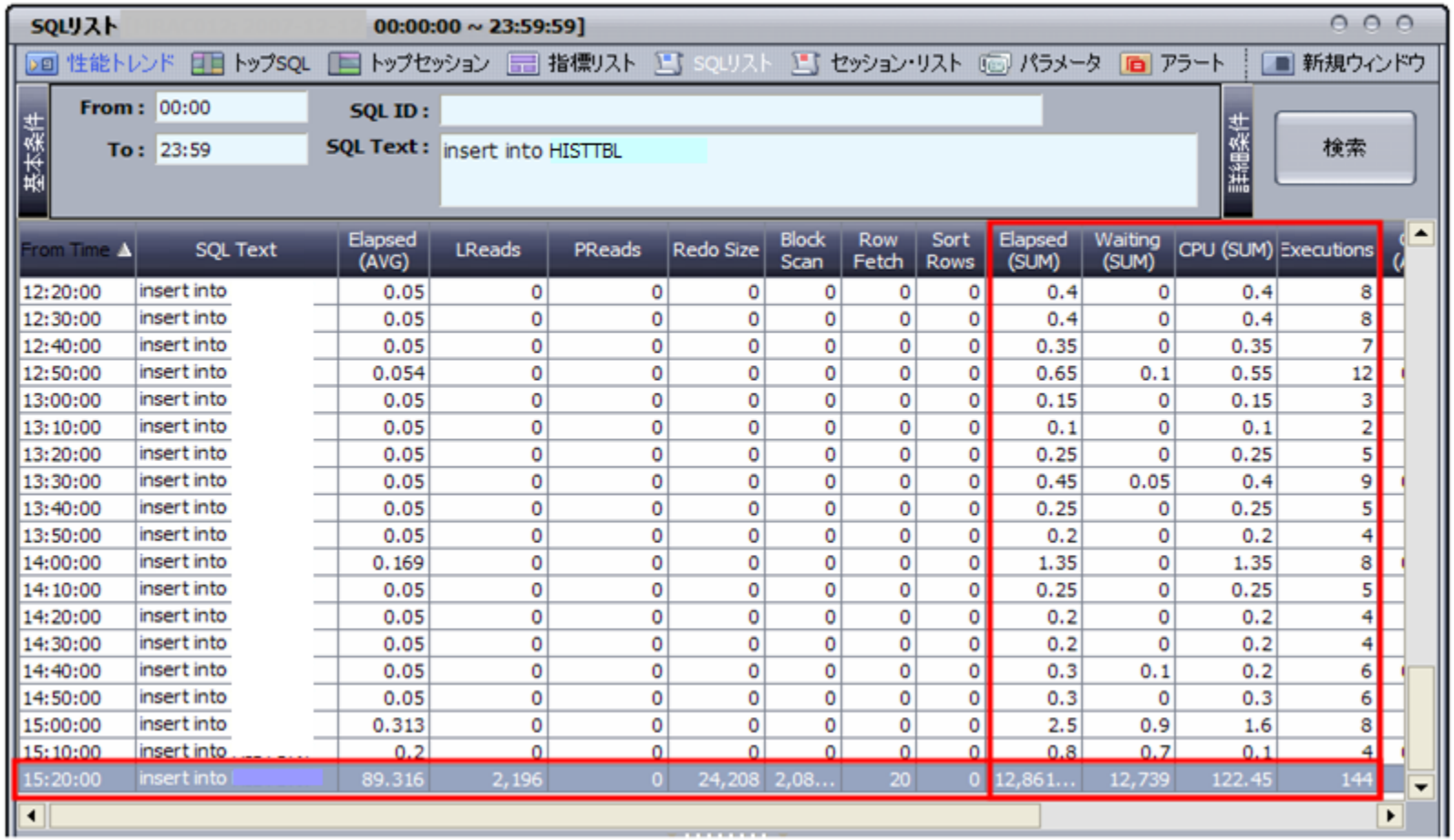
From Time	SQL Text	Elapsed (AVG)	LReads	PReads	Redo Size	Block Scan	Row Fetch	Sort Rows	Elapsed (SUM)	Waiting (SUM)	CPU (SUM)	Executions
13:50:00	delete from	0.05	0	0	0	0	0	0	0.25	0	0.25	5
13:50:00	delete from	0.05	0	0	0	0	0	0	0.2	0	0.2	4
14:00:00	delete from	0.05	0	0	0	0	0	0	0.4	0.05	0.35	8
14:00:00	delete from	0.05	0	0	0	0	0	0	0.35	0	0.35	7
14:10:00	delete from	0.05	0	0	0	0	0	0	0.25	0.1	0.15	5
14:10:00	delete from	0.05	0	0	0	0	0	0	0.15	0	0.15	3
14:20:00	delete from	0.05	0	0	0	0	0	0	0.15	0	0.15	3
14:20:00	delete from	0.05	0	0	0	0	0	0	0.15	0.1	0.05	3
14:30:00	delete from	0.05	0	0	0	0	0	0	0.4	0.05	0.35	8
14:30:00	delete from	0.05	0	0	0	0	0	0	0.1	0	0.1	2
14:40:00	delete from	0.058	0	0	0	0	0	0	0.75	0.3	0.45	13
14:50:00	delete from	0.317	0	0	0	0	0	0	0.95	0	0.95	3
14:50:00	delete from	0.05	0	0	0	0	0	0	0.35	0.05	0.3	7
15:00:00	delete from	0.069	0	0	0	0	0	0	0.9	0.6	0.3	13
15:00:00	delete from	0.05	0	0	0	0	0	0	0.15	0	0.15	3
15:10:00	delete from	0.32	0	0	0	0	0	0	4.8	3.2	1.6	15
15:10:00	delete from	0.05	0	0	0	0	0	0	0.2	0.05	0.15	4
15:20:00	delete from	58.45	1	0	332	0	0	0	350.7	350.65	0.05	6
15:20:00	delete from	42.142	38	0	2,020	0	0	0	3,371.4	1,984.8	1,386.6	80

- 「DELETE FROM HISTTBL…」は終日発行されているが、「15:20:00 ～ 15:30:00」で集中して発行されている。

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定SQL分析



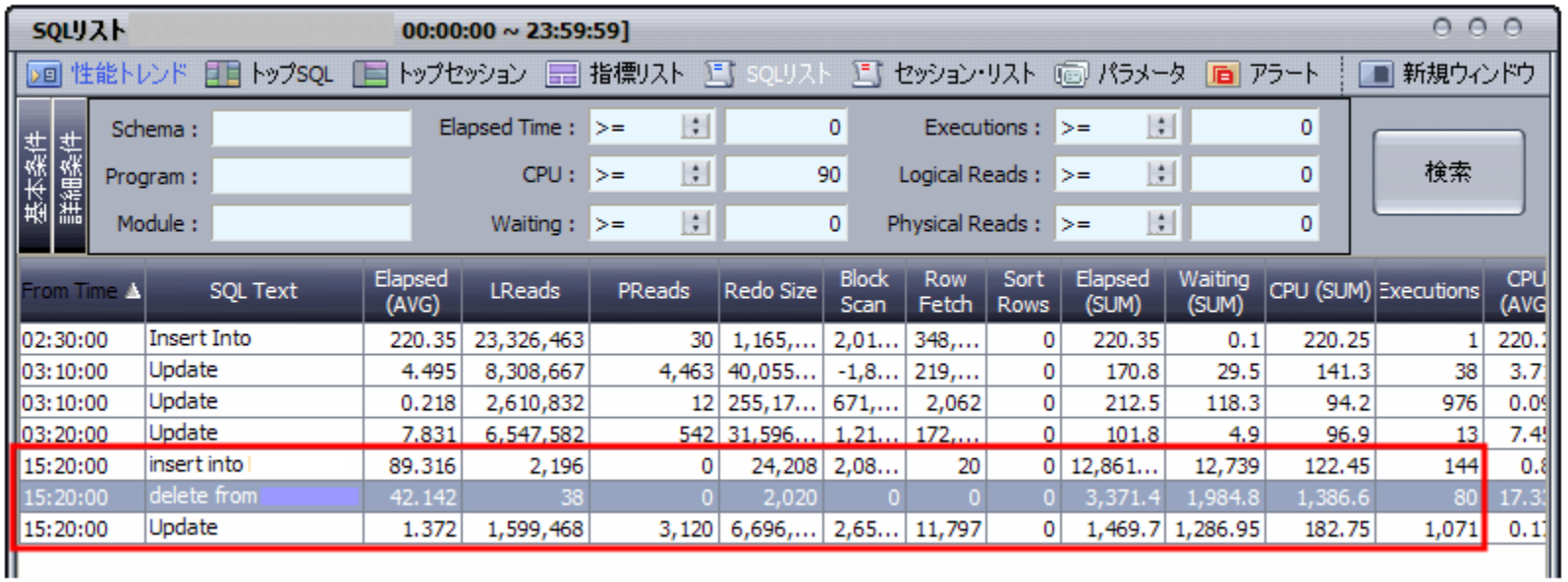
From Time ▲	SQL Text	Elapsed (AVG)	LReads	PReads	Redo Size	Block Scan	Row Fetch	Sort Rows	Elapsed (SUM)	Waiting (SUM)	CPU (SUM)	Executions
12:20:00	insert into	0.05	0	0	0	0	0	0	0.4	0	0.4	8
12:30:00	insert into	0.05	0	0	0	0	0	0	0.4	0	0.4	8
12:40:00	insert into	0.05	0	0	0	0	0	0	0.35	0	0.35	7
12:50:00	insert into	0.054	0	0	0	0	0	0	0.65	0.1	0.55	12
13:00:00	insert into	0.05	0	0	0	0	0	0	0.15	0	0.15	3
13:10:00	insert into	0.05	0	0	0	0	0	0	0.1	0	0.1	2
13:20:00	insert into	0.05	0	0	0	0	0	0	0.25	0	0.25	5
13:30:00	insert into	0.05	0	0	0	0	0	0	0.45	0.05	0.4	9
13:40:00	insert into	0.05	0	0	0	0	0	0	0.25	0	0.25	5
13:50:00	insert into	0.05	0	0	0	0	0	0	0.2	0	0.2	4
14:00:00	insert into	0.169	0	0	0	0	0	0	1.35	0	1.35	8
14:10:00	insert into	0.05	0	0	0	0	0	0	0.25	0	0.25	5
14:20:00	insert into	0.05	0	0	0	0	0	0	0.2	0	0.2	4
14:30:00	insert into	0.05	0	0	0	0	0	0	0.2	0	0.2	4
14:40:00	insert into	0.05	0	0	0	0	0	0	0.3	0.1	0.2	6
14:50:00	insert into	0.05	0	0	0	0	0	0	0.3	0	0.3	6
15:00:00	insert into	0.313	0	0	0	0	0	0	2.5	0.9	1.6	8
15:10:00	insert into	0.2	0	0	0	0	0	0	0.8	0.7	0.1	4
15:20:00	insert into	89.316	2,196	0	24,208	2,08...	20	0	12,861...	12,739	122.45	144

- 「INSERT INTO HISTTBL...」も終日発行されているが、「15:20:00 ～ 15:30:00」で集中して発行されている。

障害解析の事例

MaxGauge
Database Performance Maximizer

- 特定SQL分析



From Time	SQL Text	Elapsed (AVG)	LReads	PReads	Redo Size	Block Scan	Row Fetch	Sort Rows	Elapsed (SUM)	Waiting (SUM)	CPU (SUM)	Executions	CPU (AVG
02:30:00	Insert Into	220.35	23,326,463	30	1,165,...	2,01...	348,...	0	220.35	0.1	220.25	1	220.
03:10:00	Update	4.495	8,308,667	4,463	40,055...	-1,8...	219,...	0	170.8	29.5	141.3	38	3.7
03:10:00	Update	0.218	2,610,832	12	255,17...	671,...	2,062	0	212.5	118.3	94.2	976	0.0
03:20:00	Update	7.831	6,547,582	542	31,596...	1,21...	172,...	0	101.8	4.9	96.9	13	7.4
15:20:00	insert into	89.316	2,196	0	24,208	2,08...	20	0	12,861...	12,739	122.45	144	0.8
15:20:00	delete from	42.142	38	0	2,020	0	0	0	3,371.4	1,984.8	1,386.6	80	17.3
15:20:00	Update	1.372	1,599,468	3,120	6,696,...	2,65...	11,797	0	1,469.7	1,286.95	182.75	1,071	0.1

・CPU使用率が高いSQLの検索結果、「15:20:00 ～ 15:30:00」時間帯で集中している。

障害解析の事例

MaxGauge
Database Performance Maximizer

診断/分析サマリー

✓「15:26」頃からデータベースへ接続が増え、既存のセッションと合わせて、
「HISTTBL」表に対する集中的なデータ削除・追加作業を実施した。
✓ 作業量の増加と同じデータ(ブロック)に対する大量の同時変更作業で滞留が急増でCPUが限界に達した。
✓ このようなシステムの過負荷によって、Oracleクラスタウェアの定期的な死活監視活動のハートビート(1回/1秒)
送信が、決まった時間内に正常の応答を得られなくなった。
✓ ノード障害(データベース停止、ハングなど)と判断し、 Oracleクラスタウェアが OSの再起動を実施した。

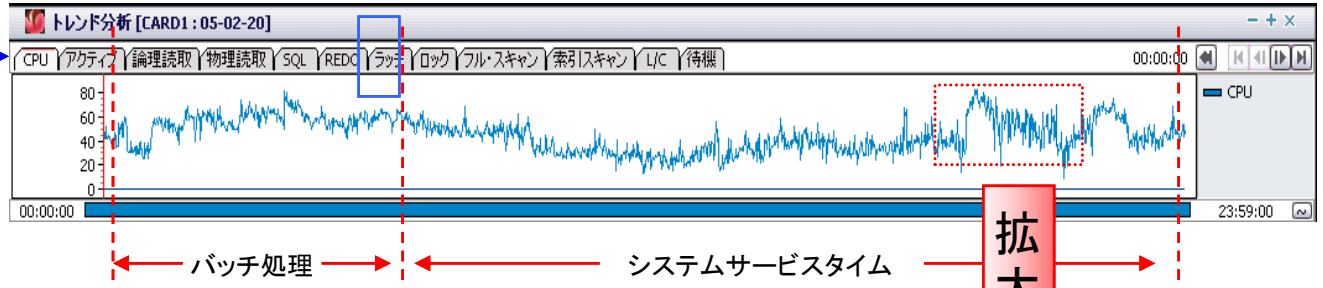
改善(チューニング)案

✓「HISTTBL」表に対するデータ削除・追加作業が集中しないようにロードバランスを行う。
→ 実施時間帯の分散、他ノードへの接続分散
✓ 同じブロックに対する競合が発生しないように「HISTTBL」表のデータを分散する。
→ パーティション化、1ブロックサイズの調整、1ブロック当りの格納データ件数の調整
✓ CPU使用率、アクティブセッション数、滞留、DB接続数に対する予兆監視を行う

簡単な使い方：全体の状況把握と拡大分析

MaxGauge
Database Performance Maximizer

一日のトレンドグラフから、負荷の高い時間帯をマウス操作（SHIFT+ドラッグ＆ドロップ）で拡大し詳細を確認。 さらに、ダブルクリックでその時点での処理をしているセッション、SQLを参照。
個別のセッション情報では、その時点でのCPU利用量・PGA利用量・データアクセス量・SQL実行時間、マシン名などが確認できます。



各メイン指標を確認
- ✓ CPU利用率
- ✓ Active Session数
- ✓ SQL実行回数
- ✓ ラッチ
- ✓ ロック　など

タイムスライス：詳細分析

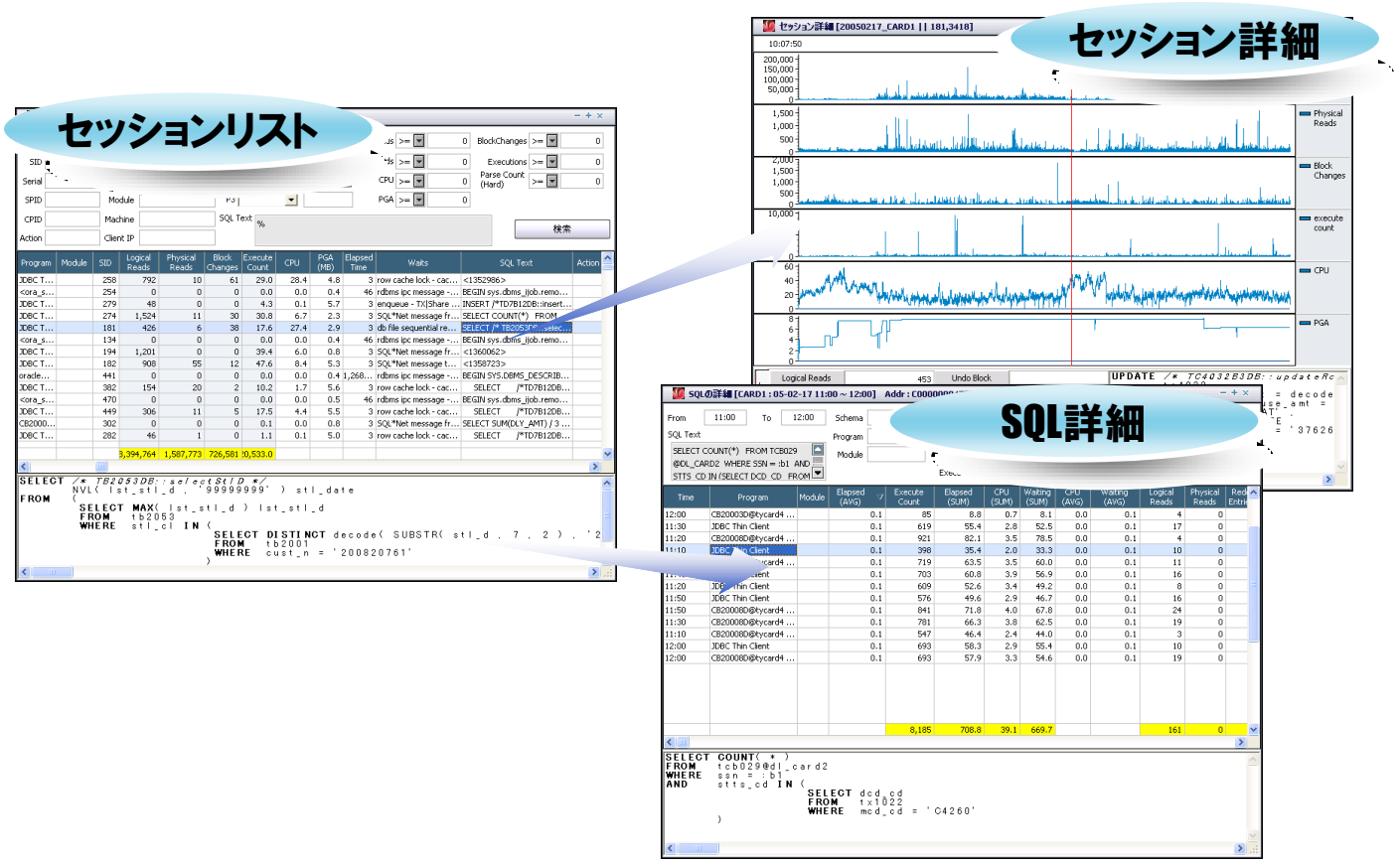
簡単な使い方：セッション毎、SQL毎の詳細追跡確認

MaxGauge
Database Performance Maximizer

セッション情報、SQL情報は多角的に分析できます。

- 1つのセッションの稼働状況をグラフ＆SQL履歴で追跡
- 1つのSQLの実行状況を時系列に参照

これらは、様々な画面から参照でき、自然にドリルダウンの感覚で参照できます。

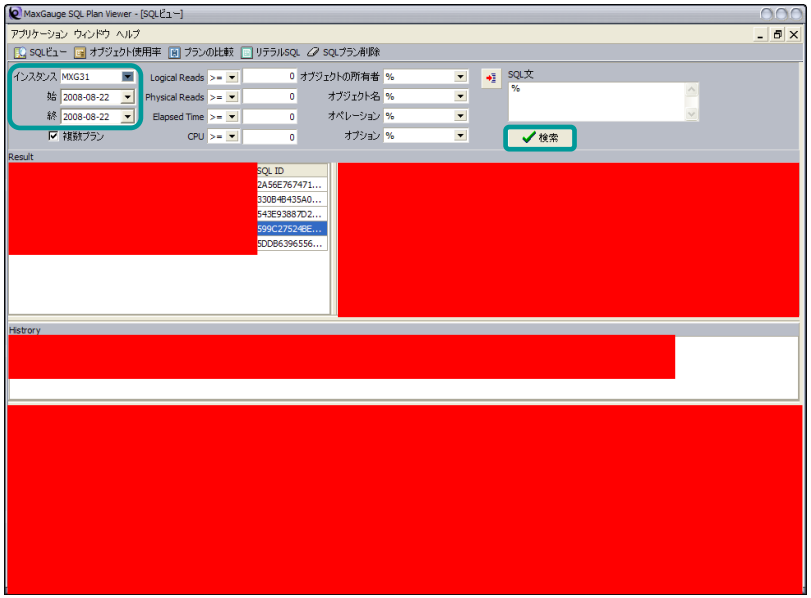


簡単な使い方：過去の実行計画確認

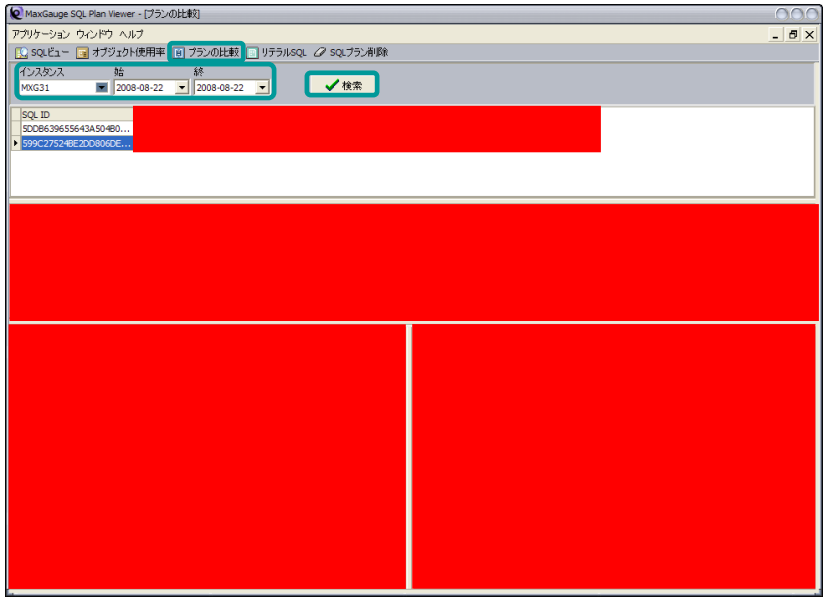
MaxGauge
Database Performance Maximizer

実行されたSQLの実行計画もつぶさに記録しておくことが出来ます。
これにより、自由に過去のSQLの実行計画が参照できるほか、実行計画の変化も簡単に参照することが出来ます。

実行計画の変化による突然のパフォーマンス低下での、対象SQLが瞬時に把握できます。



特定時間帯のSQLと実行計画の表示



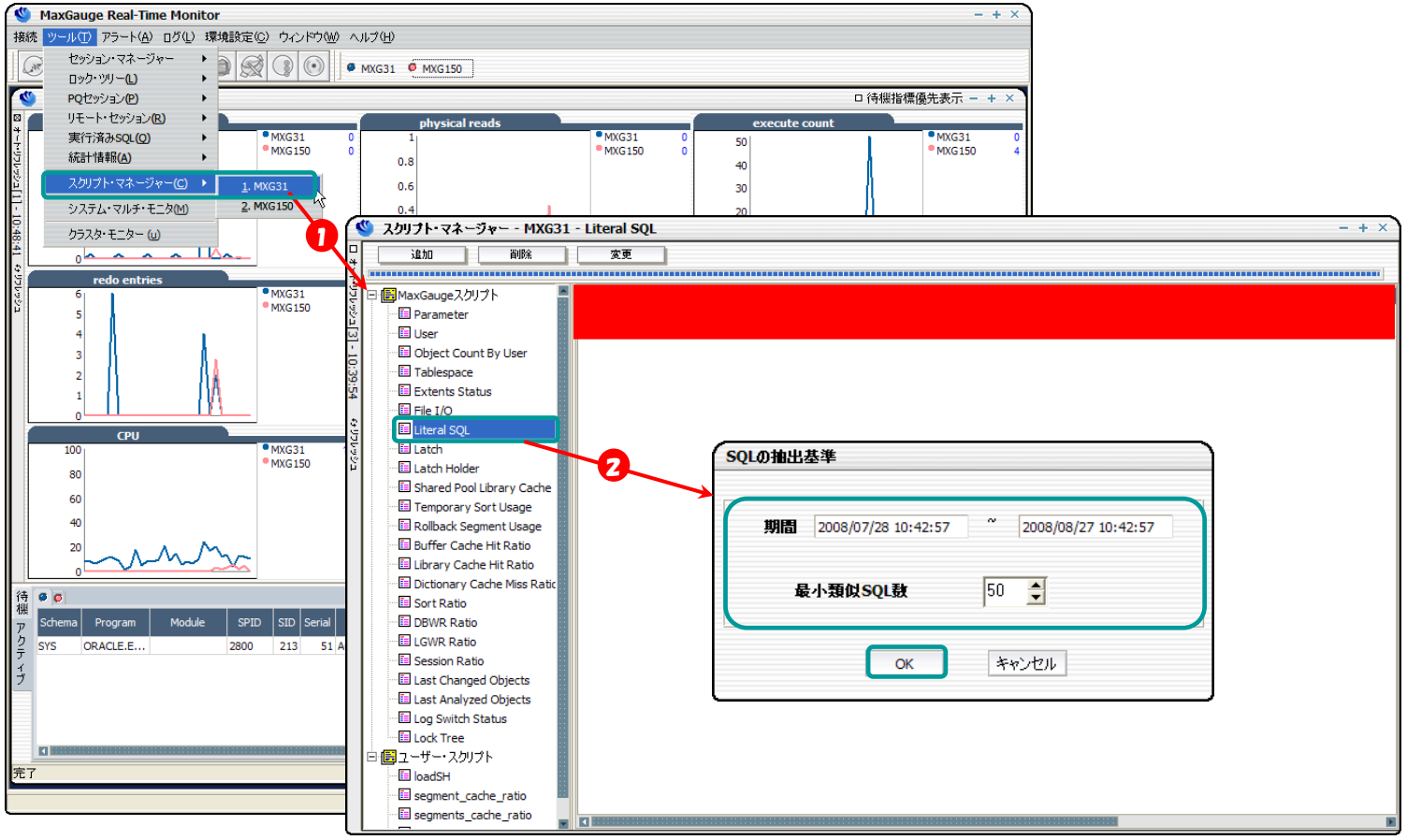
特定SQLの実行計画の変化を比較参照

※ 実行計画の取得には、別途情報蓄積のためのOracleデータベースが必要となります。

簡単な使い方：リテラルSQLの確認→「ORA-4031」対応

MaxGauge
Database Performance Maximizer

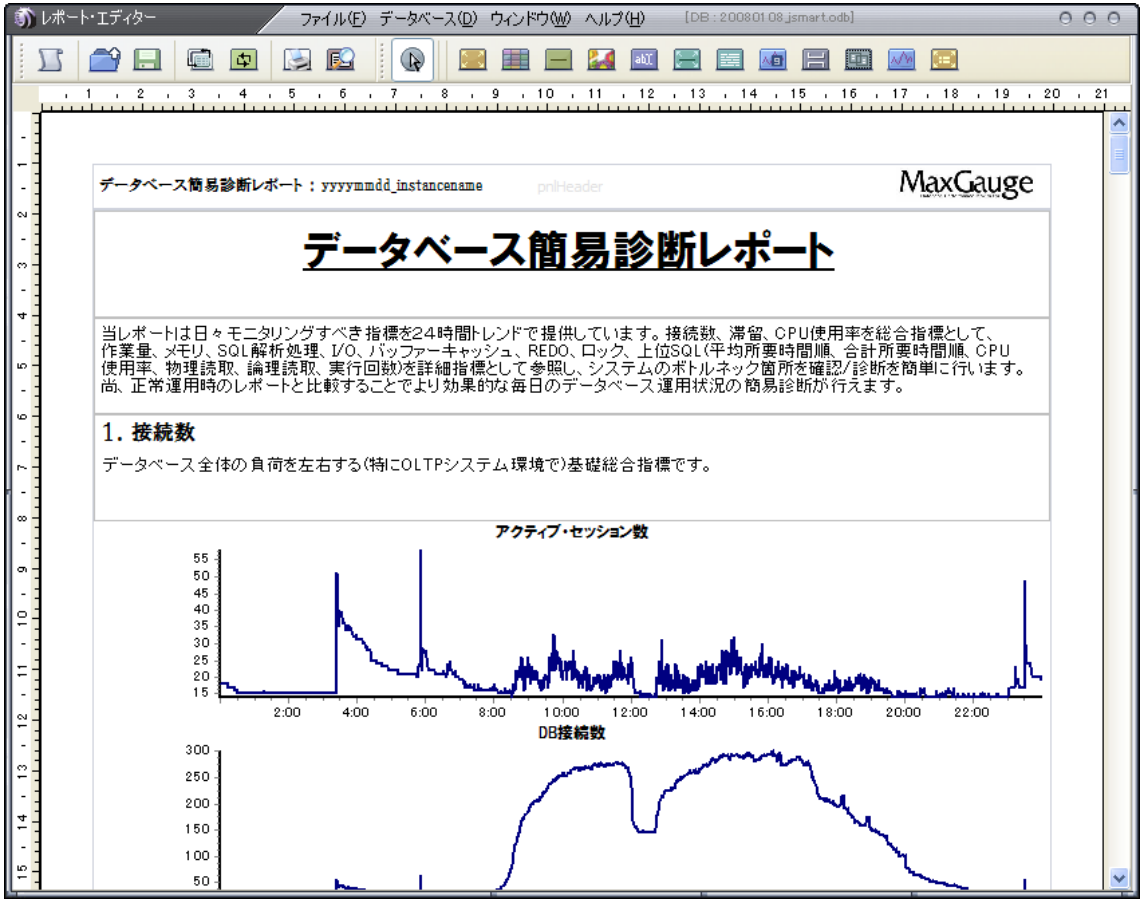
SQL解析処理は、思いのほかデータベースに負荷をかけます。 負荷の原因となるリテラルSQLを簡単にリストアップすることが出来ます。 また、データベース管理者が良く利用するスクリプト郡があらかじめ登録されています。



簡単な使い方：簡易診断・分析レポート

MaxGauge
Database Performance Maximizer

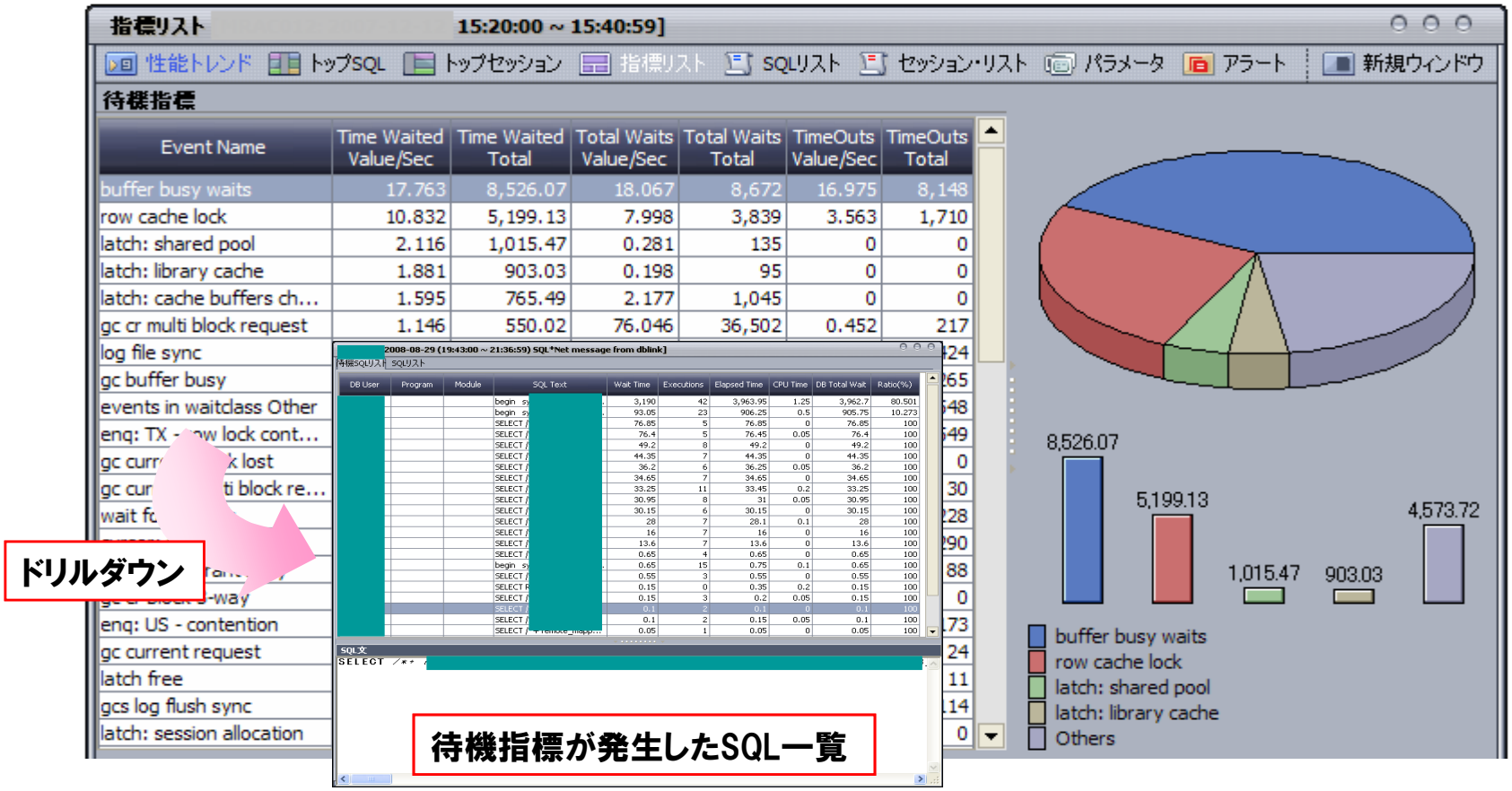
データベースの全体状況を簡単に把握する 「データベース簡易診断レポート」が付属しています。主要指標のトレンドや、各リソースごとのTop SQLなどが自動的に出力されます。レポートはカスタマイズも可能です。



簡単な使い方：ボトルネック個所の表示と該当SQLのピックアップ

MaxGauge
Database Performance Maximizer

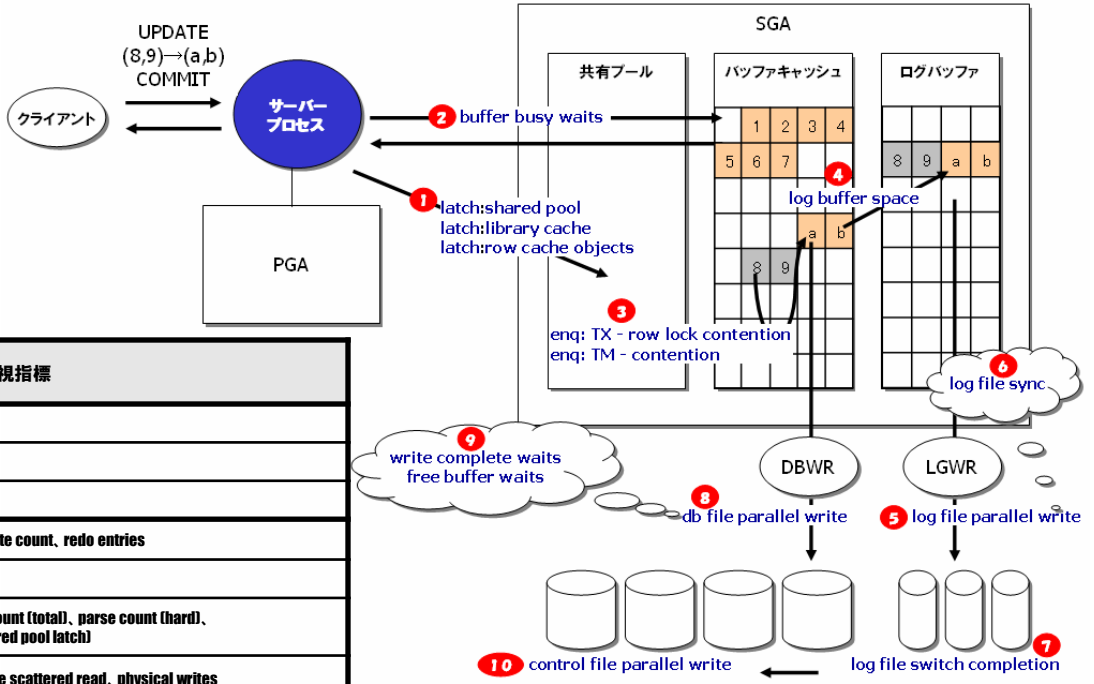
データベースの内部での滞留を表す「待機指標」がグラフと数値で確認できます。これによりボトルネックポイントが簡単に把握できます。また、各待機指標が発生したSQLを待機が長い順に表示され、ボトルネックへ一番影響を及ぼしているSQLがわかります。



参考：待機指標はデータベースのボトルネックポイントを表す

MaxGauge
Database Performance Maximizer

待機指標は、データベース内での各部処理に要した時間を集計したデータです。これによりデータベース内部のどの部分でボトルネックが発生したのかがわかります。
MaxGaugeでは、その待機指標が発生したSQLまで簡単に把握できるため、改善対象のSQLなどがわかります。



区分	監視領域	監視指標
総合指標	接続数	アクティブセッション、DB接続数
	滞留現象	総合待機時間
	CPU	CPU使用率
詳細指標	作業量	session logical reads、physical reads、execute count、redo entries
	メモリ(OS)	(OS) free memory, (OS) used memory ratio
	SQL解析処理	parse time elapsed、parse time cpu、parse count (total)、parse count (hard)、共有プール関連待機(library cache latch, shared pool latch)
	I/O	physical reads、db file sequential read、db file scattered read、physical writes
	バッファー・キャッシュ	バッファキャッシュ関連待機(cache buffers chains latch, cache buffers lru chain latch、buffer busy waits、read by other session、free buffer waits、write complete waits)、データ共有率(buffer cache hit ratio)
	REDO	log file sync、log buffer space、log file switch completion、ユーザートランザクション数
	ロック	enqueue、lock waiting sessions、ロックリスト
	上位SQL	トップSQL(所要時間、CPU、論理読取、物理読取、実行回数)

参考：主要指標のイベントヘルプ

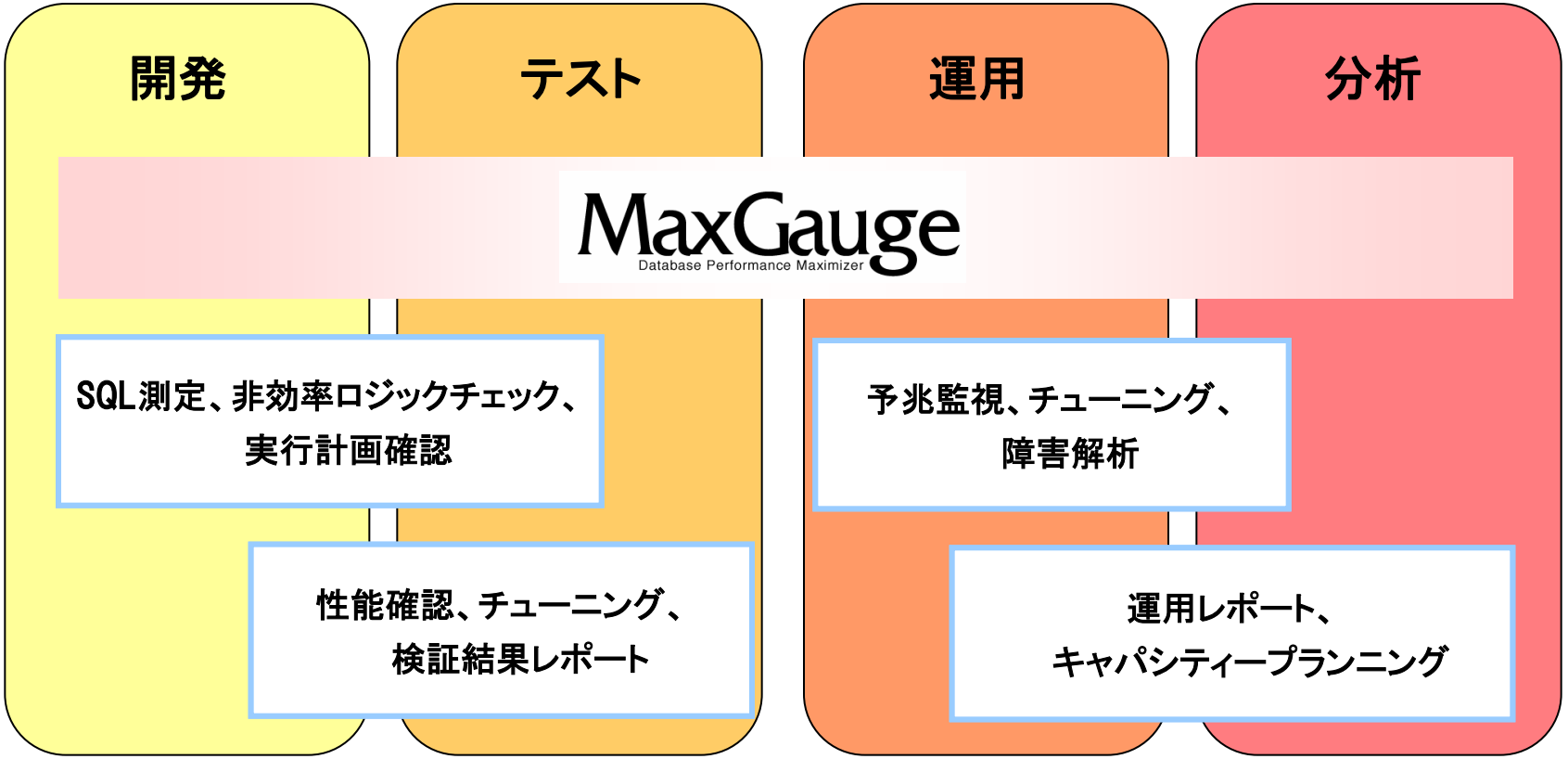
MaxGauge
Database Performance Maximizer

主要な待機指標情報はイベントヘルプで提供します。 これにより、指標の意味はもとより、改善するべきポイント、関連指標などが把握できます。

プロジェクトフェーズ全体での活用

MaxGauge
Database Performance Maximizer

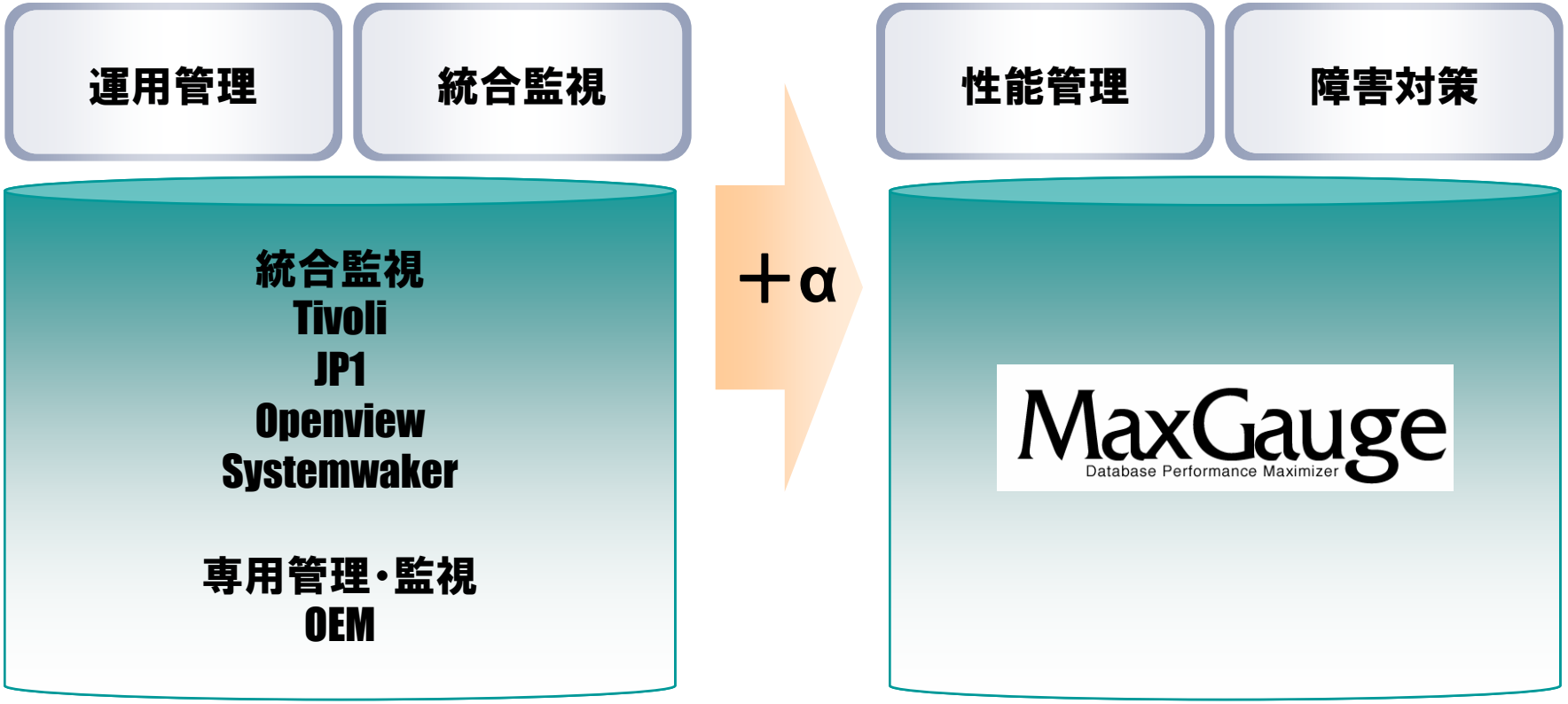
MaxGaugeは、Oracleデータベース稼動情報の「見える化」ツールとして、Oracleを利用する全工程での活用を推進しています。これまで確認・検証・解析のために個々で行われていたSQL*PlusやSTATSPACKなどでの情報収集作業が自動で行われ、さらに詳細なあらゆる情報を現場エンジニアから管理者まで共有し利用することが出来ます。



稼動情報の提供による運用意識の拡張

MaxGauge
Database Performance Maximizer

現在、多くのシステムで運用管理や統合監視は通常のことになっています。 これにMaxGaugeを加えることにより、『性能監視』、『障害対策』という運用の質の向上を推進していくことができます。



導入効果・導入場面

MaxGauge
Database Performance Maximizer

データベースの稼動情報を常時記録しておくことにより、様々な場面で有効に活用できます。それにより、工数の削減や、すばやい対応が可能となります。

導入効果

- トラブル原因調査工数が劇的に削減
  - 情報収集・分析工数は現在の1/10
  - 「トラブル再現待ち」回数が激減　1/10
- 表面化していないトラブルを発見
  - 非効率なロジックや長時間ロックなどを事前に発見
  - トラブル発生前に予兆として発見・アラート
- 迅速な意思決定
  - 改善ポイントを即座に把握
  - 他への影響範囲が見える
  - 改善後の状況Watchも安心
  - 視覚化され客観的にみえるため、メンバーへの説明・説得も簡単

導入場面

- トラブルでお困りのとき
  - トラブル発生時の状況を確実に把握
  - 調査コスト削減・機会損失の減少
- 開発プロジェクトにて
  - 情報収集・確認工数の削減
  - 負荷検証の分析工数が激減
- 定期的な診断
  - 定期的に状況を確認
  - 過去の記録との比較
  - 不穏な動きを事前に察知
  - 将来的な指針になる情報を提供

導入効果

MaxGauge
Database Performance Maximizer

- **過去の導入実績から、システム障害発生率が50%削減、ダウンタイム30%減少という成果をあげています。**
- **それをもとにすれば、以下の導入効果が期待できます。**

1.運用・保守工数			補足
A	要員数	5人	仮に5名で1システムを運用するものとして試算
B	業務において障害対応にかかる割合(%)	10%	弊社知見
C	MaxGaugeによる効率化(%)	**50%**	弊社導入実績より、障害発生率が50%削減と仮定
D	MaxGaguge導入による運用・保守削減工数	**3人月/年**	A*12ヶ月*B*Cにより算出

2.ダウンタイムによる事業の機会損失			
A	現状のシステム稼働率	99.4%	JUAS調査より、基幹系システムの稼働率平均値
B	MaxGaguge導入による稼働率向上(%)	**30%**	弊社導入実績より、ダウンタイムが30%削減と仮定
C	当該システムが関連する事業の売上	100,000百万/年	仮に100,000百万の事業に影響するものとして試算
D	ダウンタイムによる事業の機会損失	**180百万/年**	C*(100%-A)*B

導入事例

MaxGauge
Database Performance Maximizer

NRI様
「開発～テスト～運用」までの各フェーズでの利用によるトータルな情報収集・分析コストの削減による生産性向上ツールとして、各プロジェクトへ導入を推進

レコチョク様
ベンダー任せではなく、自分たちでのKnowledgeの蓄積、運用の質の向上のために導入。

大手流通A社様
SystemWalker＋MaxGaugeで、統合運用管理に加え、ユーザー処理の把握と障害の迅速対応を実現

大手金融B社様
Webアプリケーションから自動発行されるSQLの捕捉。 性能管理・障害解析に利用。

効果事例(１)

MaxGauge
Database Performance Maximizer

ラッシュテストでのSLA要件(5秒ルール)を満たせないアプリケーションの特定と原因追求

【事象】

ラッシュテストにおいて、5秒ルールを満たせないアプリケーション処理が5箇所あった。
(ランダムな60多重での処理にて5秒以内というレスポンス要件)

【解決策】

MaxGaugeでのデータ取得/分析により、5秒ルールを満たせないアプリケーションの負荷状況を把握。対象SQLの特定と、それぞれのリソースの利用量、実行時間を数値的に参照できるようになった。

【効果】

開発者は、ラッシュテストでの情報収集の仕組みを一切作成せずに、テスト結果とその状況の数値化、グラフ化を実現。対象SQLの特定と、それぞれのリソースの利用量から、どのSQLをどの程度チューニングする必要があるのかを的確に把握。
また、SQL履歴より想定外のロジックでSQLが処理されていることなどもわかった。
テスト準備がほぼゼロとなり、チューニング対象SQLの早期発見が可能となった。 作業工数はおよそ1/10程度に縮小。

効果事例(２)

MaxGauge
Database Performance Maximizer

原因不明のデータベース再起動問題の解決（原因トリガーの追跡と対応）

【事象】

RAC環境において、突然Oracle Cluster wareが、データベース停止状況であると判断し、データベースの再起動をかけてしまっていた。

【解決策】

MaxGaugeのログより、再起動時点でのデータベース内処理状況より、同一ブロックへの大量な「DELETE」、「INSERT」処理が行われていることが判明。そのため、OSレベルでの遅延が発生したと判断。対象SQLのチューニングにより、データベース負荷を軽減。データベース再起動の事象の発生を抑えた。

【効果】

MaxGaugeのログ調査以前、システム開発担当者、およびオラクルサポート担当にて原因調査を約1週間行っていたが、原因がつかめなかった。MaxGaugeのログの参照により、約2時間で障害時の状況の把握と対象SQLのピックアップを行い、顧客へレポート。（Cluster wareが、データベースを再起動させてしまう事象については、製品仕様の確認のため、オラクルサポート担当とのやり取りは継続）

効果事例(３)

MaxGauge
Database Performance Maximizer

パフォーマンス低下・トラブル時の運用部隊と開発部隊での認識相違の解決

【事象】

パフォーマンス低下を含む問題の対応にて、運用部隊がトラブルの対象となったユーザー・SQLを特定することが困難であったため、開発部隊への明確な修正指示などが難しい状況にあった。

【解決策】

MaxGaugeのログより、問題発生時のユーザー・SQLの特定から、新機能で新たに追加されたSQLであることが記録されていた。該当SQLを即座に開発部隊に改善するよう指示をした。

【効果】

トラブル発生でも、原因となるユーザー・SQLの特定が困難なことから、運用部隊と開発部隊の連携が難しかった。(開発部隊には問題認識がなく、運用部隊は確固たる証拠がなく、強制力をもてなかったため、**修正の説得まで数週間**はかかっていた。)
MaxGaugeログにより、第3者的な証跡からお互いに記録を確認。問題発生時のSQLとそのSQLのリソースの利用量、実行時間より数値的に問題があることを証明でき、**即座に修正に取り掛かれる**ようになった。

動作環境

MaxGauge
Database Performance Maximizer

マックスゲージ(MaxGauge)は、以下のシステムに対応しています。

◆ 対応プラットホームサーバー

- IBM　AIX4.3以降
- HP-UX　11.0以降
- SunOS 5.6以降
- Compaq True64　5.1A
- Redhat Linux カーネル2.4以降
- WindowsXP/VISTA/2000/2003/2008

＊ロギングのためのディスク領域が必要です。

◆ 対応クライアント

- Windows XP/VISTA/NT/2000/2003/2008

◆ 対応Oracleバージョン

Oracle　7.3.4 ～ 11g

＊別途オラクルユーザーアカウントが必要です。

Q&A

Thank you !

＜お問い合わせ＞

日本エクセム株式会社

TEL : 03-4360-3951

e-mail : info@ex-em.co.jp

MaxGauge

Database Performance Maximizer