データベース運用
転ばぬ先の杖

2009年12月22日

株式会社インサイトテクノロジー
エンジニアリング本部
テクノロジーコンサルティング部
松尾　亮

システム運用監視の現状

システム運用監視の現状

- 障害が発生してからの事後対応が多い
- 画一的な運用で終始してしまう
- データベース運用監視まで手が回らない

データベース運用監視をしない理由

- コストに合うメリットを感じない
- 何を監視すればよいかわからない
- どのように監視すればよいかわからない

データベースに障害が発生した時のコスト

サービス停止による直接的コスト

＋

障害復旧作業による間接的コスト

障害調査

・原因特定が困難
→調査時間増大
・問い合わせ対応
→ログ取得などの
作業が発生

リカバリ

・リカバリ範囲の確認
→直近？
フルバックアップ？
・リカバリ手順の確認

業務復旧

・データ整合性確認
・再開後の監視
→24時間張り付き?

事後対応

・障害報告書の作成
→エンドユーザー
や関係会社など
・対応策の検討

コスト増大！！

データベース障害を未然に防ぐ為に・・・

問題発見～チューニングのプロセス

1.稼働データ取得

- 正常値の把握
- 障害予兆検知

2.データ分析

- データを分析し予兆を検出
- 予兆を解消する為のチューニング策を検討

3.対応策実施

- 検討したチューニング策を実施

4.対応策評価

- 正常値と比較

事例1

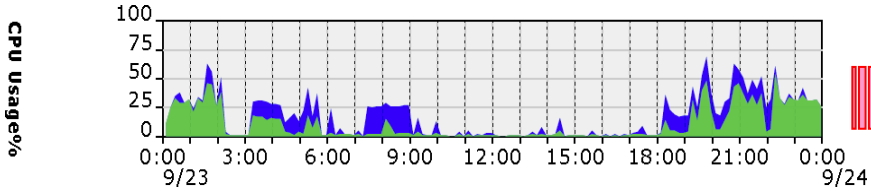
事例1　～概要～

- システム概要
  - 某製造系システムのDB
  - Oracle9iR2
- 状況
  - 日中はオンライン処理が実行
  - 深夜にバッチ処理やバックアップが実行
  - 現状では処理遅延等の問題は発生していない
  - 過去に障害が発生し、数億円の損害が生じた
- 要望
  - 問題が発生する前に未然に防ぎたい
  - 毎月一回定期的に状況を確認したい

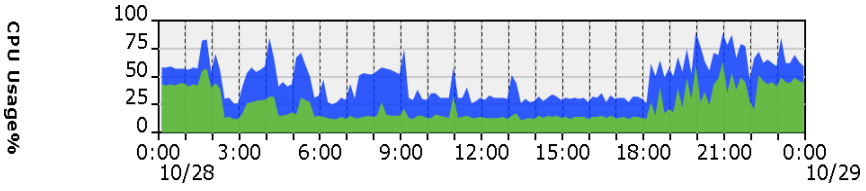
事例1　～状況～

- 9月のデータと10月のデータを比較
- 10月のCPU使用率が1日を通して上昇
- 10月14日からCPU使用率が上昇していると判明

9月のCPU使用率

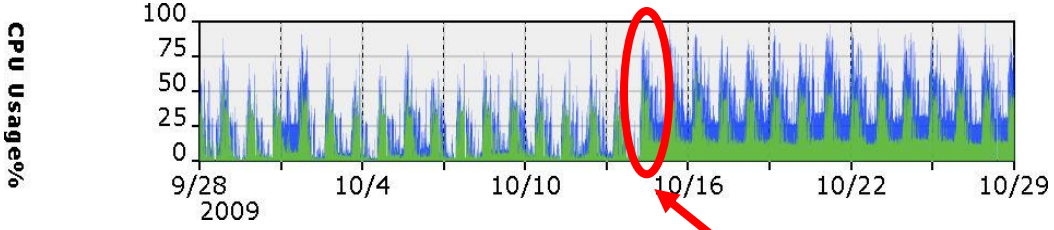


10月のCPU使用率



調査

9月～10月のCPU使用率

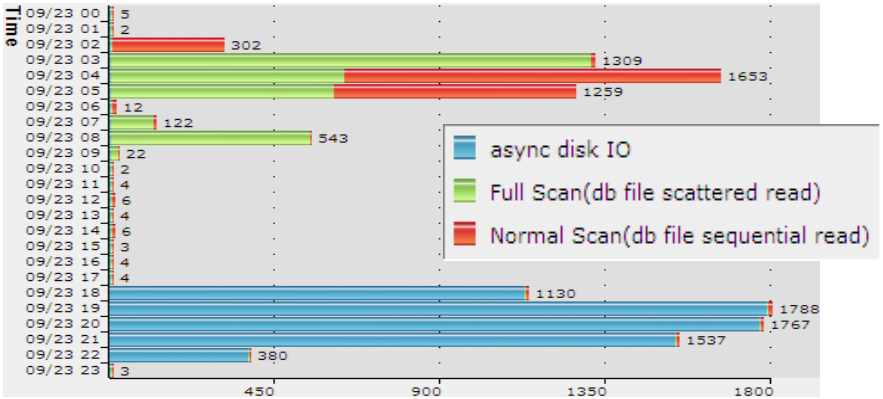


10/14から上昇している！

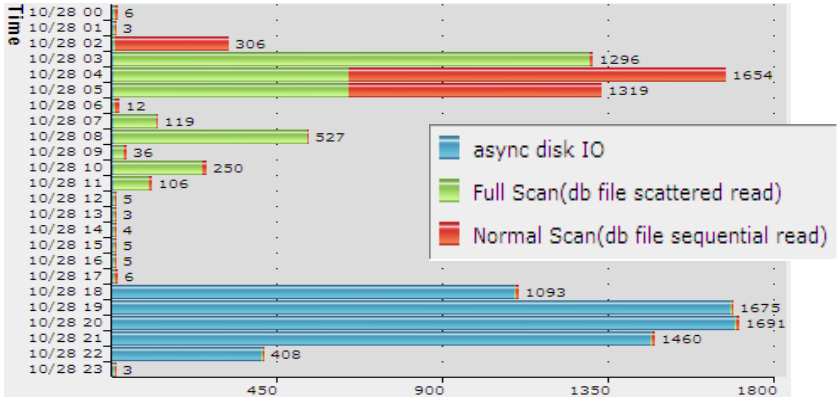
事例1　～状況～

Oracleデータ調査

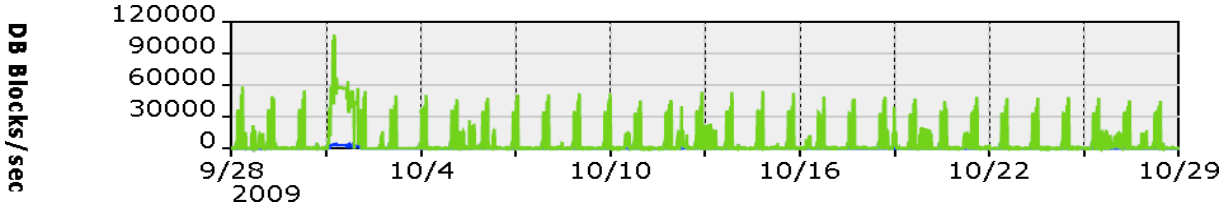
9月の待機イベント（上位3個）



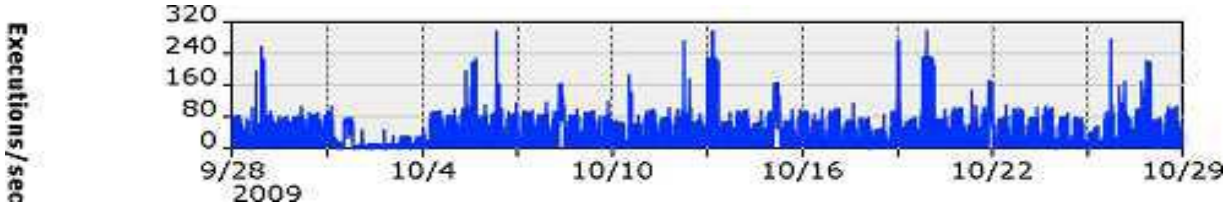
10月の待機イベント（上位3個）



9月～10月論理読込ブロック数



9月～10月SQL実行回数



CPU高騰に繋がりそうなDB統計では、特に変化なし

そういえば、
プロセス監視もしてたっけ・・・

事例1　～原因～

不審なプロセスが実行され続けていることを確認！

DATE-TIME	MESSAGE
2009/10/14 19:48	bash(25892):CPU使用時間 55.14sec(/60sec)
2009/10/14 19:58	bash(25892):CPU使用時間 54.99sec(/60sec)
2009/10/14 20:08	bash(25892):CPU使用時間 58.37sec(/60sec)
中略	
2009/10/14 23:38	bash(25892):CPU使用時間 57.21sec(/60sec)
2009/10/14 23:48	bash(25892):CPU使用時間 57.74sec(/60sec)
2009/10/14 23:58	bash(25892):CPU使用時間 57.42sec(/60sec)

10/14作業時に実行したプロセスの残存と判明。

事例1　～まとめ～

プロアクティブにデータベースを監視する為の仕組みを作っていたので、**短時間で原因特定**ができ、業務に支障がでなかった。

が、本来であれば、10/14の**作業直後にチェック**しておいた方が良かった。

データベースの監視に加え、CPU、メモリ等の基本的なOSリソース情報はもちろんのこと、**プロセス情報も取得**しておこう！

事例2

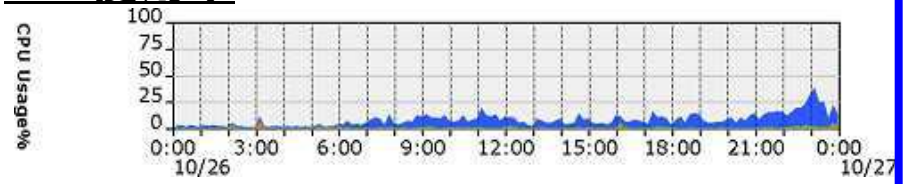
事例2　～概要～

- システム概要
  - 金融系システムのDB
  - Oracle10gR1 4node RAC
- 状況
  - 取引は月曜午前7時から土曜午前7時まで
  - 土曜の日中にメンテナンス作業実施
  - 月曜午前7時のオープン直後からDBの負荷が急上昇
  - サーバハングによる障害発生
- 要望
  - 障害調査を依頼

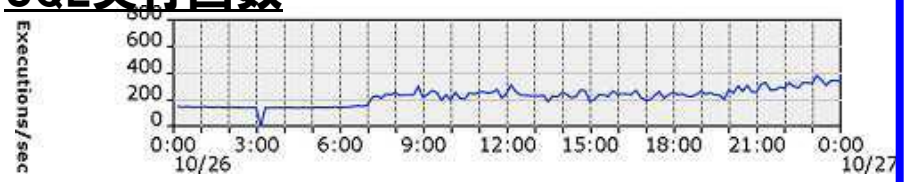
事例2　～状況～

通常時のデータ（月曜日）

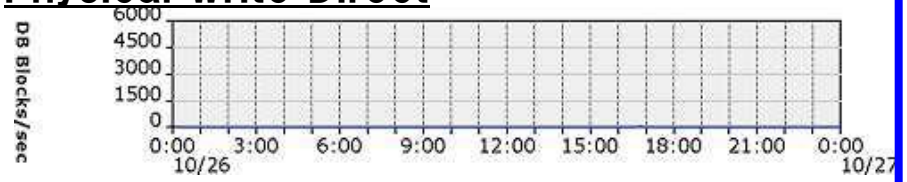
CPU使用率



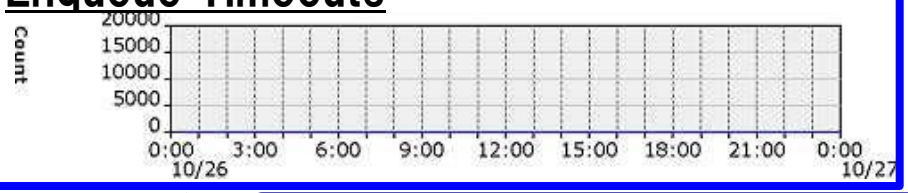
SQL実行回数



Physical Write Direct

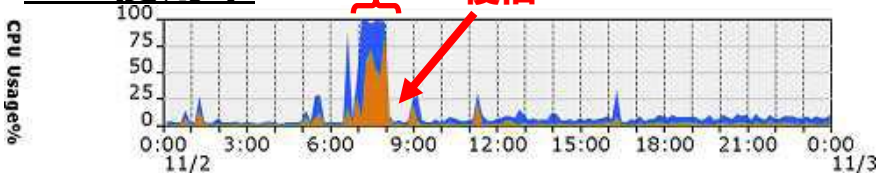


Enqueue Timeouts

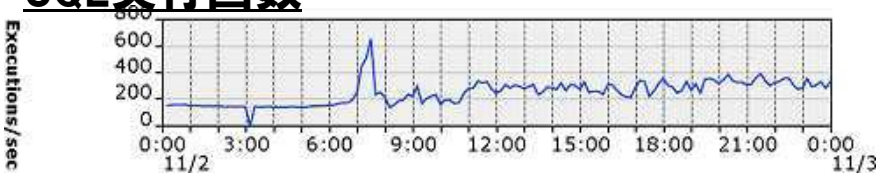


障害時のデータ（月曜日）

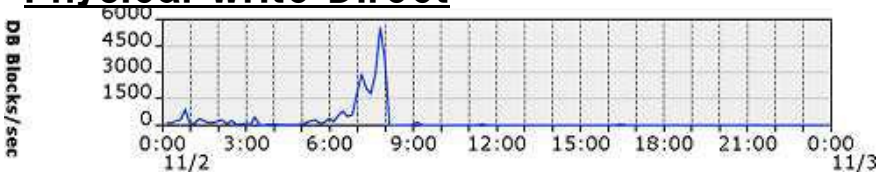
CPU使用率



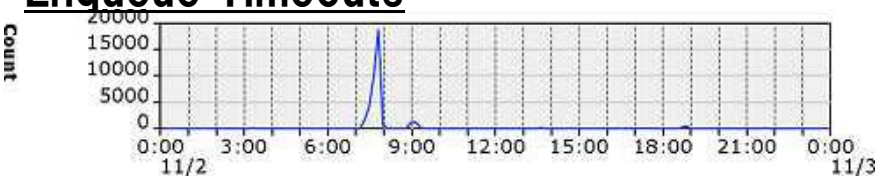
SQL実行回数



Physical Write Direct



Enqueue Timeouts

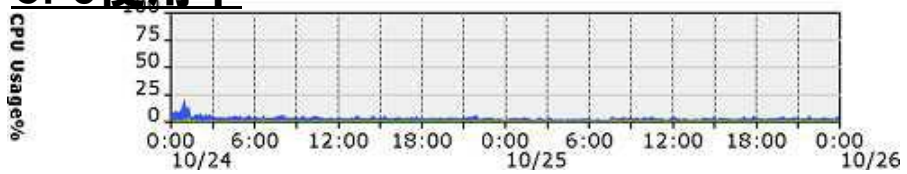


障害発生時に様々なデータの値が上昇

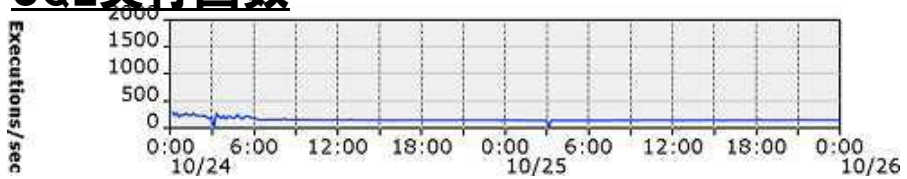
事例2　～状況～

通常時のデータ（土日）

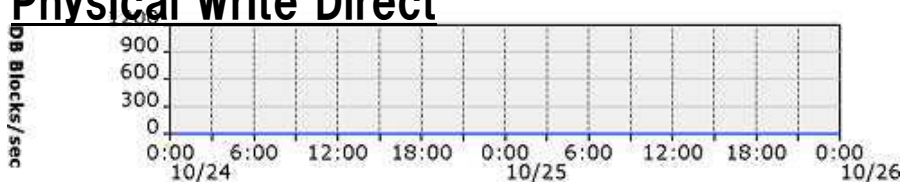
CPU使用率



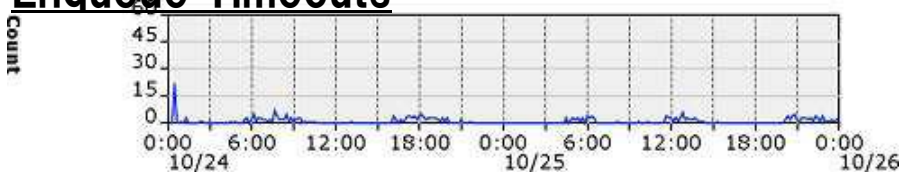
SQL実行回数



Physical Write Direct

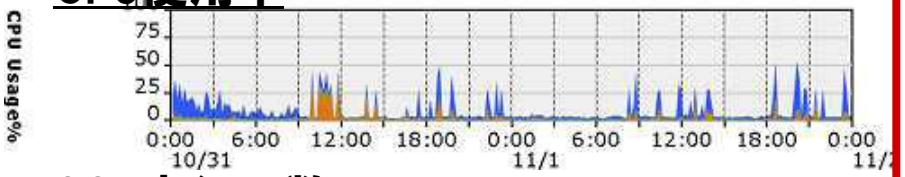


Enqueue Timeouts

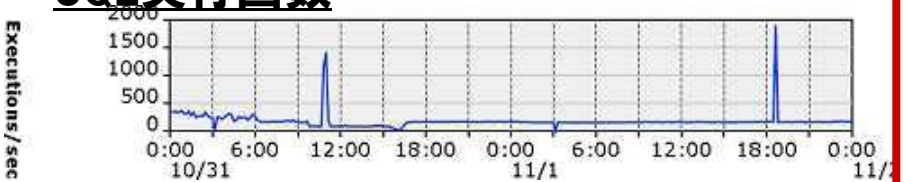


障害前のデータ（土日）

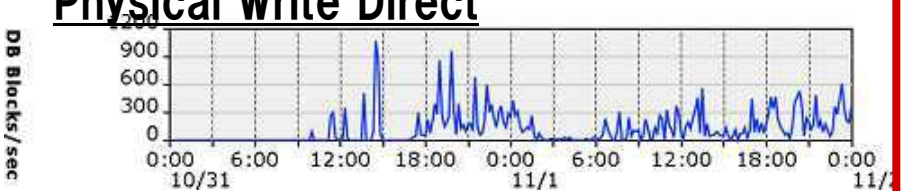
CPU使用率



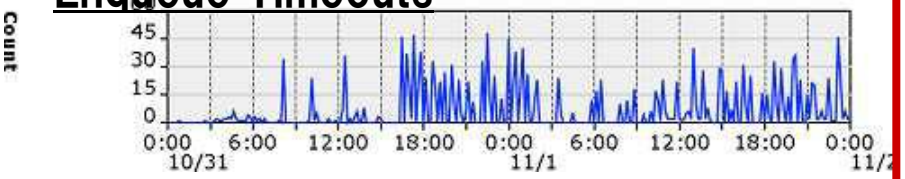
SQL実行回数



Physical Write Direct



Enqueue Timeouts



障害発生前に兆候は出ていた

事例2　～現場の様子～

この時の現場の様子をお届けします・・・

【土曜の日中】

Aさん「古いデータを大量に削除したからフラグメンテーションを解消しとかないとな。」

```
SQL> alter table EMP move;
SQL> alter index PK_EMP rebuild;
SQL> alter index IDX_EMP_1 rebuild;
SQL> alter index IDX_EMP_2 rebuild;
```

Aさん「これで良し。さ、帰ろっと。」

事例2　～現場の様子～

【月曜7時】

Aさんの携帯に運用監視オペレータから電話が・・・

OP　「Aさんっ！DBサーバのCPUが100%に張り付きっぱなしで
　　サービスが停まってます！」

Aさん「なんですとぉ！？すぐに確認します！」

Aさんはリモート接続してすぐに調査を開始。

事例2　～現場の様子～

【月曜7時30分頃】

Aさん「土曜にメンテナンスした表にアクセスしているSQLが
遅いみたいだな。実行計画が変わっちゃったみたい・・・
よし、統計情報を収集しよう！」

```
SQL> exec dbms_stats.gather_table_stats( -
     ownname => 'SCOTT', -
     tabname => 'EMP', -
     cascade => true);
```

Aさん「どうかなぁ・・・。」

事例2　～現場の様子～

駄目だ！！

実行計画も変わってないようだ。

Aさん「統計情報を再収集したから、再解析してる筈なんだけどなぁ・・・何で実行計画が変わってくれないんだろ。ブツブツ・・・」

その後しばらく試行錯誤するも状況は改善せず・・・

事例2　～現場の様子～

【月曜8時頃】

Aさん『どうしよう・・・何か他に手はないかな。。。
あ、実行計画の問題だから共有プールも関係あるか？
とりあえずフラッシュしてみよう。』

```
SQL> alter system flush shared_pool;
```

Aさん『ん？ちょっと負荷が下がってきたか？
おぉー！下がった下がった！よかった～』

これで一安心・・・
でも、かなり大きな損害が。。。

事例2　～解説～

［原因］

以下のメンテナンス作業によって、表のオプティマイザ統計情報が削除された。

1. 表の再編成（move処理）
2. 索引の再編成（rebuild処理）

その後、統計情報再収集を行わなかった為、デフォルトの統計情報が利用され、結果的に適切な実行計画が選択されなかった。

（補足）
move処理後は索引が使用不可（STATUS=UNUSABLE）となる為、rebuild処理が必須。

事例2　～解説～

[統計情報の収集後、実行計画が変わらなかったワケ]

dbms_stats.gather_table_stats の no_invalidate オプションが auto_invalidate（10gR1以降のデフォルト値）だった為。
この auto_invalidate の場合、新しい統計情報を利用した再解析が行われるまでにタイムラグがある。
（しばらくの間は既存のカーソルを再利用する）

[対策]

1. no_invalidate を false にして実行する
　　or
2. 共有プールをフラッシュする

事例2　～まとめ～

障害時には兆候があるので、それを見逃さない仕組み作りとその問題を解決する為の適切なチューニングが必要である。

統計情報を収集しても、10g以降のデフォルト値では即時に再解析されない場合がある。
9i以前のデフォルト値では、統計情報収集後のSQL初回実行時には再解析されていたが・・・

事例3

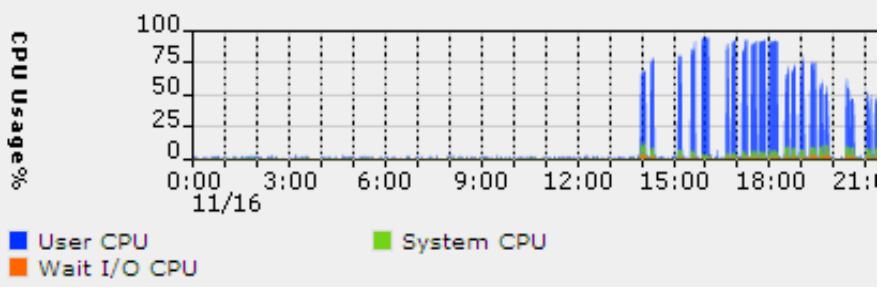
事例3　～概要～

- システム概要
  - モバイルコンテンツ配信システムのDB
  - Oracle11gR1 2node RAC
- 状況
  - アプリ改修に伴い索引を新規追加。
    追加後の動作検証(負荷テスト)で遅延が発生。
    (追加前:400処理/秒→追加後:200処理/秒)
  - 追加した表は、複数セッションから insert (1処理1件)が大量に行われるという特性がある。
- 要望
  - 索引を追加した状態で追加前と同等のパフォーマンスに戻したい。

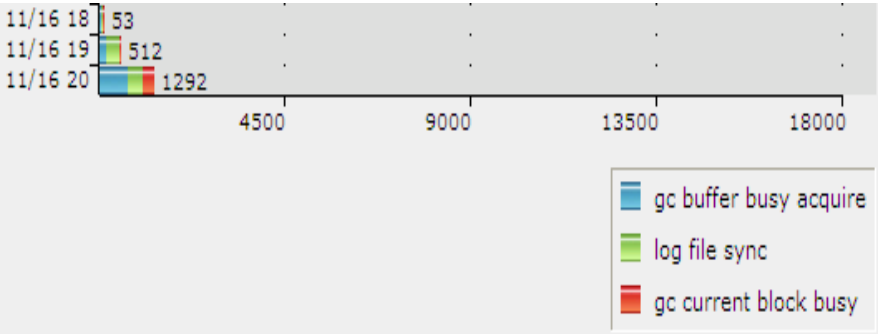
事例3　～状況～

索引追加前

CPU使用率

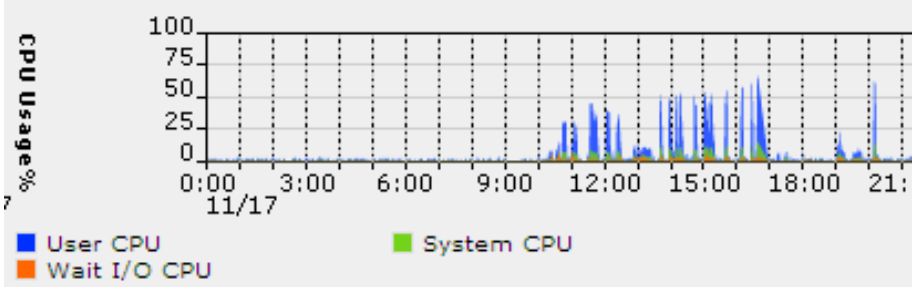


待機イベント

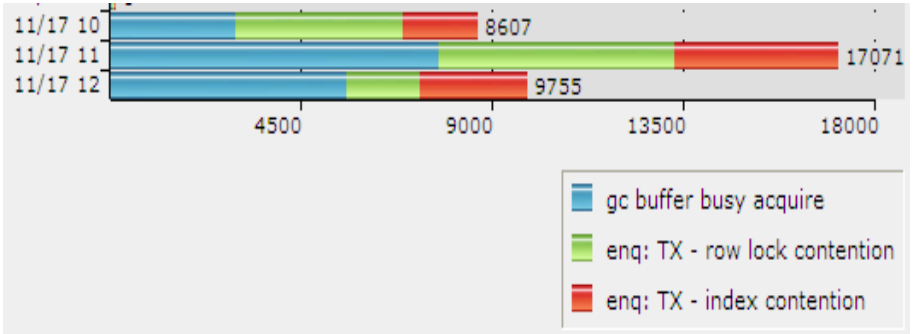


索引追加後

CPU使用率



待機イベント

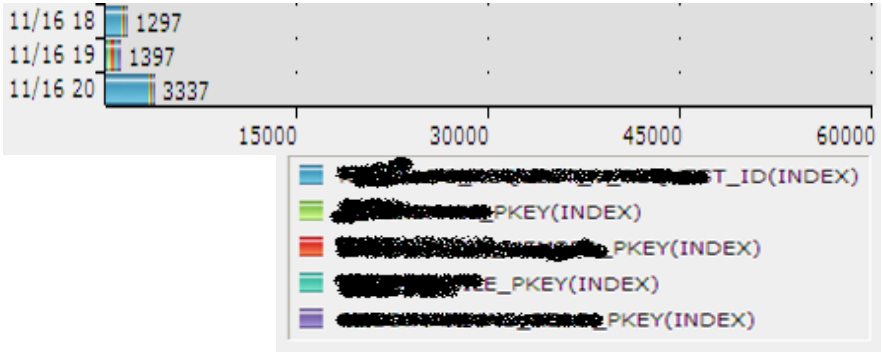


CPU使用率は減少。（CPUが使えていない）
ロック関連の待機イベントが多発！

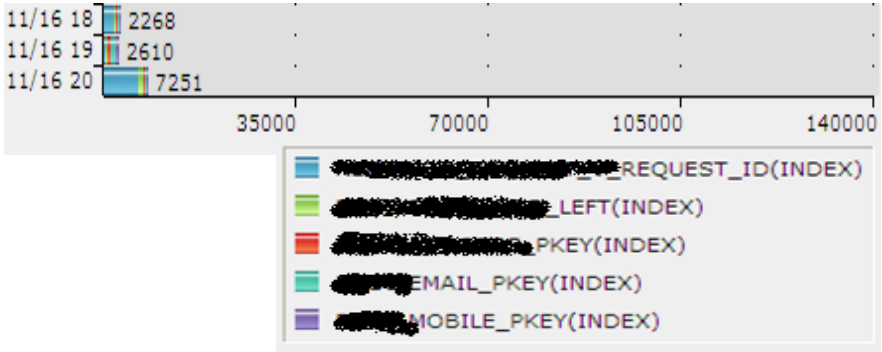
事例3　～状況～

索引追加前

ロック発生オブジェクト



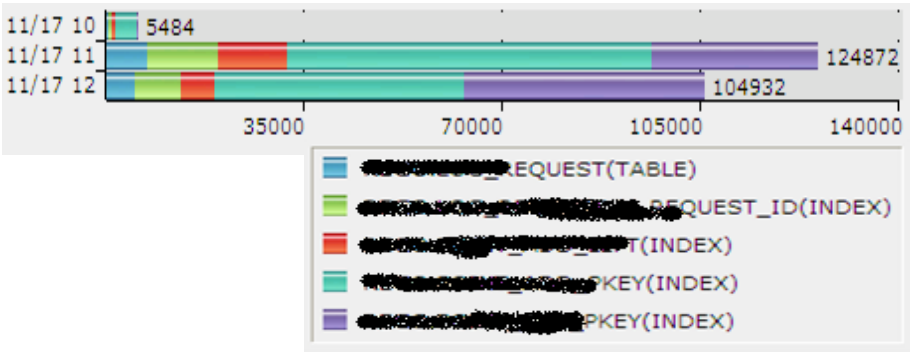
buffer busy発生オブジェクト



索引追加後

ロック発生オブジェクト

buffer busy発生オブジェクト



追加した索引に関連する表、索引に対する
ロック、buffer busy が多発している！

事例3　～現場の様子～

それではまた、現場の様子をお届けします・・・

アプリ担当者（以下、アプリ）
「アプリ改修したので、使いそうな索引を追加しました。」

DBA　「分かりました。確認します。
・・・って、めちゃくちゃいっぱいあるじゃないですか！
この索引、全部使うんですか？？」

アプリ「いやー、ちょっと分からないんですけど、使うかもしれないとこには全部作ってみました。

事例3　～現場の様子～

DBA　「そ、そうですか・・・
　　確か、今回索引を追加する表って、どれも insert が
　　大量に実行されるテーブルですよね？
　　索引があると insert は遅くなるんですよ。
　　だから、検索に使われない索引は削除したいですね。」

アプリ「はー、そうですか・・・」

DBA　「ま、とりあえず負荷テストしてみますか。
　　そんなに影響出ないかもしれないし。」

事例3　～現場の様子～

パフォーマンス50%にDown！

DBA　「あらら・・・ちょっと想像以上に遅くなりましたね。。。
　　まずは使われてない索引を削除しましょうか。
　　想定される一連の処理を実行してもらえますか？」

アプリ「はい、分かりました。」

しばし待ち、実行終了・・・

事例3　～現場の様子～

DBA　「使われてない索引がいくつかあったので削除しますね。
　　これとこれと・・・
　　これらは削除しても影響なさそうですか？
　　確実に使う筈、というものがあれば教えて下さい。」

アプリ「うーん、削除しても大丈夫だと思います。」

DBA　「じゃ、削除しちゃいまーす♪」

そして、再度負荷テスト・・・

１０%くらいパフォーマンス回復。まだ、あと40%・・・

事例3　～現場の様子～

（ちょっとbreak）

使用されていない索引の確認方法として索引のモニタリング機能がありますが、v$sql_plan でメモリに存在する実行計画を参照し、現在、実行計画に使用されていないインデックスを確認するという方法も可能です。短期間のテスト等ではお手軽に確認できます。

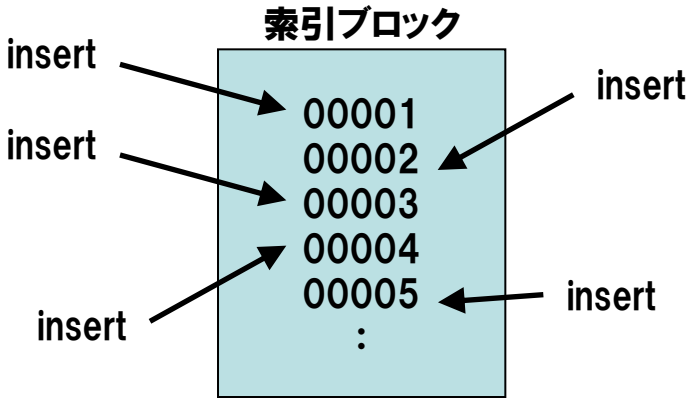
```
SQL> select object_name,object_type,hash_value
  1  from v$sql_plan
  2  where object_type like 'INDEX%' and object_name like '%EMP%'
  3  group by object_name,object_type,hash_value
  4  order by 1,2;

OBJECT_NAME    OBJECT_TYPE        HASH_VALUE
-------------- ------------------ ----------
IDX_EMP_1      INDEX              4000148954
PK_EMP         INDEX (UNIQUE)     1933697836
PK_EMP         INDEX (UNIQUE)     4074557984
```

出力されてない
→検索に利用されてない

事例3　～現場の様子～

DBA　「insert 競合を解消するには・・・
フリーリストを増やす！かな。試してみるか！
と思ったけど、、、
この表がある表領域は自動セグメント領域管理だった。
フリーリストの調整はできないじゃん・・・
えーっと、この表は連番で insert されていくから、
どうしても索引のブロックは競合するんだろうな。。。」

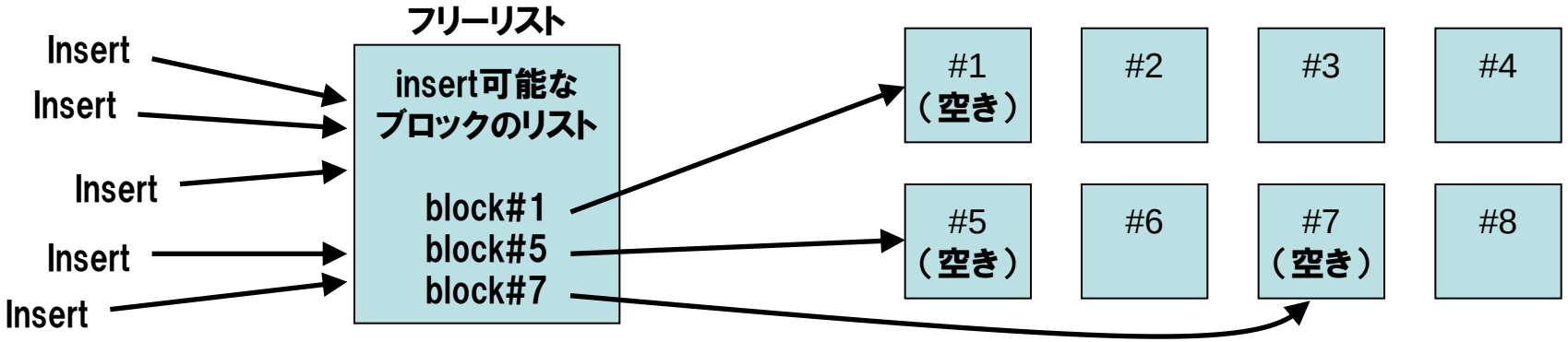


索引は順番に並んで格納される為、
このシステムの処理特性上、同一
ブロックへの競合が発生しやすい！

事例3　～現場の様子～

（ちょっとbreak）

フリーリストとは、insert 可能な空きブロックの情報を管理している領域。insert 時にはフリーリストの空きブロック情報を参照する為、insert の多重度が高いとフリーリストへのアクセスが競合する。
その場合、フリーリストを増加させることでパフォーマンスが向上する可能性がある。自動セグメント領域管理（ASSM）では、ビットマップ情報で空きブロック情報が管理され、明示的に調整はできない。



事例3　～現場の様子～

DBA　「連番で次々と insert されてくるデータを同一ブロックに
集中させない為には・・・
ハッシュパーティションだな！
逆キー索引も有効か？？？
でも、逆キー索引は範囲検索できないんだよな、確か。
パーティションオプション持ってるし、パーティショニング
してみるか！」

問題となっていそうな表と索引をハッシュパーティション化・・・

性能改善！更に、以前より20%程度の性能向上！

事例3　～解説～

[原因]

連番データの insert が多重実行されるという特定であった為、
同一ブロックへの競合が発生しやすい状況であった。
そのような状況で過剰に索引を追加したことにより、ブロック競合が多発した。

[対処]

1. 不要な索引の削除。
2. ハッシュパーティショニング。

事例3　～まとめ～

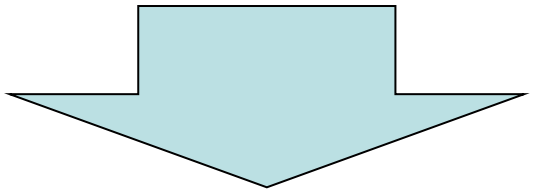
構成変更（索引追加）後に負荷テストを行ったことで、
本番稼動前に適切なチューニングが行えた。

索引がボトルネックになる場合もある。

- 更新系処理では、表データと同様に索引データの更新も必要となる為、オーバヘッドがかかる。
- 連番データの insert が多重実行されるような場合索引ブロックの競合が発生しやすい。

まとめ

定常的に稼動状況を監視し、**トレンドを把握**しておくことが重要。特に、**メンテナンスの後**は要注意！
また、万一の際にも**迅速な情報収集**が可能となり、適切な対処を行うまでの**スピードがUP**します！

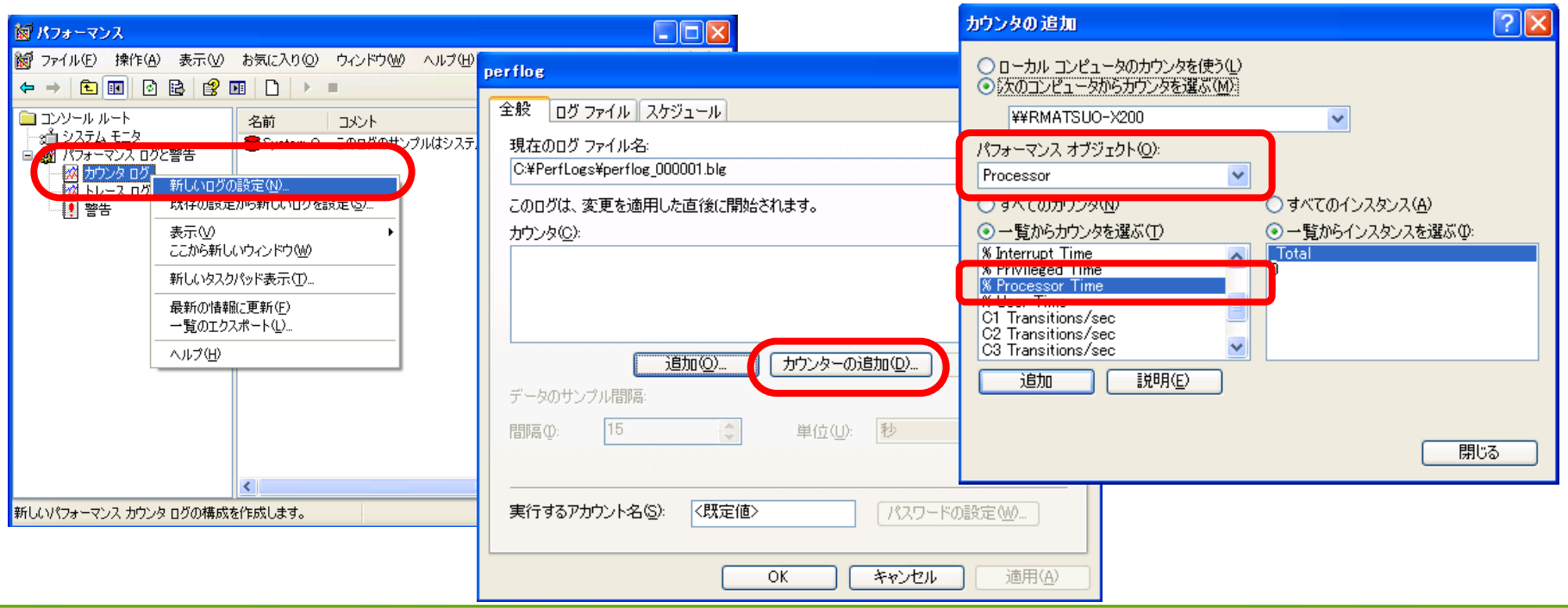


運用コスト削減！

（参考）監視情報の取得方法　～OS～

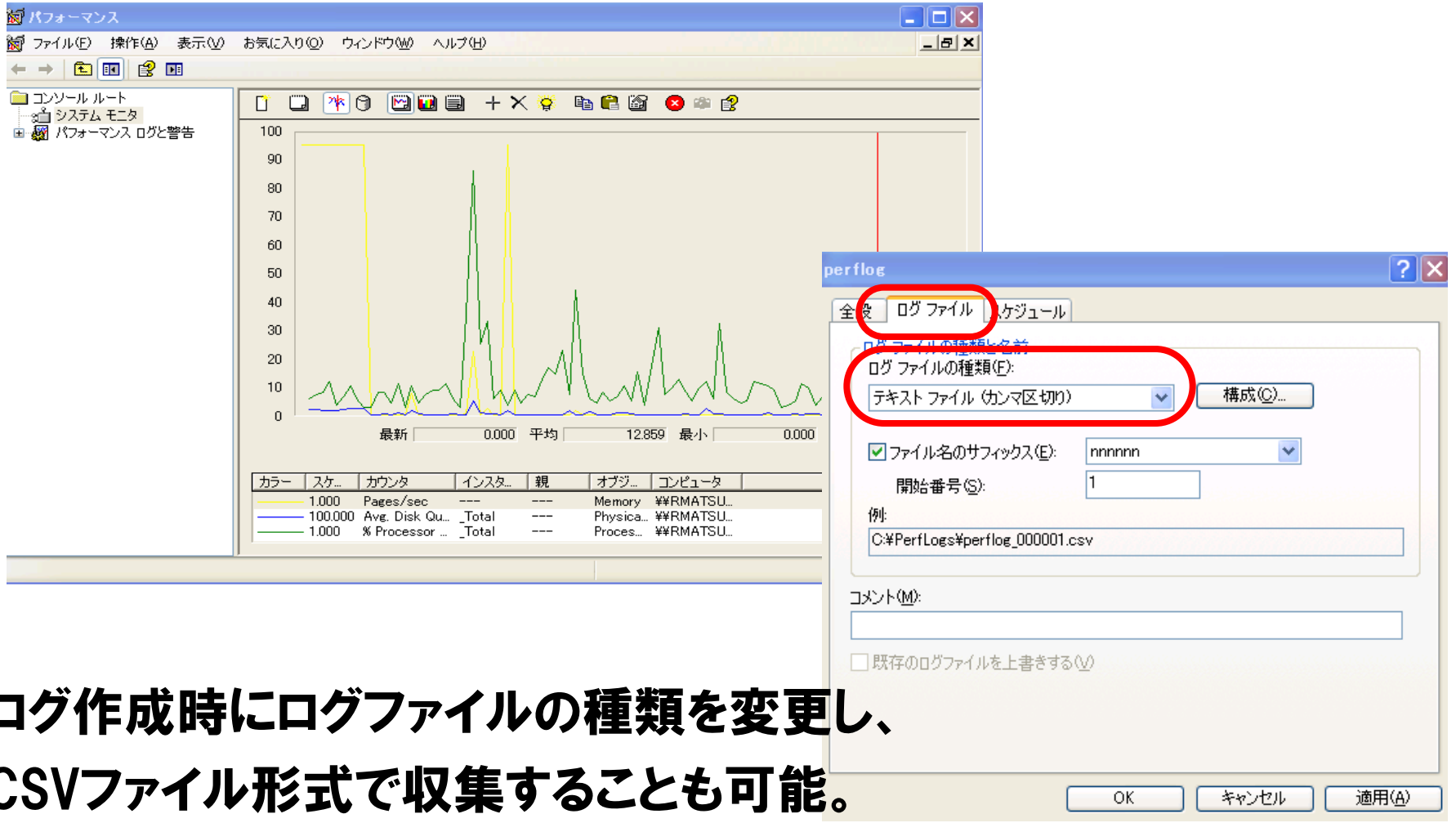
OS情報

vmstat、iostat、ps等を利用して取得。
windowsの場合、[管理ツール] → [パフォーマンス] → [カウンタログ] → [新しいログの設定] より、監視したい項目を選択する。



（参考）カウンタログサンプル

作成したログ（.blgファイル）をクリックするとグラフが表示される。



ログ作成時にログファイルの種類を変更し、
CSVファイル形式で収集することも可能。

（参考）監視情報の取得方法　～DB～

データベース情報

AWRレポート（10g～）、STATSPACKレポート（8.1.6～）等で取得。

【AWRレポートの取得コマンド】

```
SQL> @$ORACLE_HOME/rdbms/admin/awrrpt.sql
report_typeに値を入力してください: text or html を入力（ex. text）
num_daysに値を入力してください: snap_id の表示日数を入力（ex. 1）
                    :
                    :
                9392 17 12月 2009 18:00      1
                9393 17 12月 2009 19:00      1
                9394 17 12月 2009 20:00      1
                9395 17 12月 2009 21:00      1

begin_snapに値を入力してください: 開始snap_idを入力（ex. 9393）
end_snapに値を入力してください: 終了snap_idを入力（ex. 9395）
report_nameに値を入力してください: レポート名を入力（ex. 省略）
```

（参考）AWRレポート出力サンプル

```
DB Name         DB Id    Instance     Inst Num Release     RAC Host
------------ ----------- ------------ -------- ----------- --- ------------
rac11107      2896592809 rac111071           1 11.1.0.7.0  YES server201

              Snap Id      Snap Time      Sessions Curs/Sess
            --------- ------------------- -------- ---------
Begin Snap:       9394 17-12月-09 20:00:55      141       3.4
  End Snap:       9395 17-12月-09 21:00:56      114       4.1

             :
             :

Top 5 Timed Events
~~~~~~~~~~~~~~~~~~                                       Avg %Total
                                                        wait   Call
Event                                 Waits    Time (s)   (ms)   Time Wait Class
----------------------------- ------------ ---------- ------ ------ ----------
CPU time                                        17,387   78.4
db file sequential read             32,347        441     14    2.0 User I/O
row cache lock                      33,072        416     13    1.9 Concurrency
gc cr block busy                     5,663        293     52    1.3 Cluster
log file sync                       57,969        120      2     .5 Commit
```

インサイトテクノロジーのご紹介

- 設立：1995年7月7日
- 資本金：1億円（2009年4月）
- http://www.insight-tec.com
- 役員
  - 代表取締役 石井 洋一
  - 取締役　　石川 雅也
  - 取締役　　下山 勝義
  - 取締役　　岡崎 太輔
- 所在地
  - 本社：東京都渋谷区恵比寿1-19-19
    ビジネスタワー 5F
- 事業内容
  - 製品開発 / 販売

  - データベースコンサルティング
  - SCMコンサルティング

製品のご紹介

データベース監査ツール

- データベースログ管理に必要なすべてを提供するツール
- 270社、1,800ライセンスを超える販売実績でJ-SOXおよび情報漏洩対策ツールのシェアNo.1
- 大規模顧客、大規模システムへの導入に強み
- 対応データベース
  Oracle Database EE/SE
  Microsoft SQL Server（2000_32bit.2005_32bit）
  Fujitsu Symfoware Server

パフォーマンス管理ツール

- パフォーマンス管理、パフォーマンスチューニングを実現するツール
- 実績
  1995年の販売以来、1,000社、8,000ライセンスを超える販売実績。オラクルデータベースの6本に1本の割合で導入されるパフォーマンスツールのデファクトスタンダード
- 対応データベース
  Oracle Database EE/SE/SE1

※オラクルデータベースの6本に1本という数字は、
特定パートナー様からの報告内容より引用しております。

インサイトテクノロジーの情報発信コンテンツ

- おら！ オラ！ Oracle - どっぷり検証生活

Oracleをもっと知りたい人のための無料メールマガジン
おら！オラ！Oracle
－どっぷり検証生活－
www.Insight-Tec.com/jp/em/ora3.html

Oracleを徹底検証した結果を、
隔週水曜日に配信しています。

ご登録はこちら
http://www.insight-tec.com/mailmagazine/mailmagazine_index.html

おら！オラ！オラクル
著者：木脇 高太郎
出版：翔泳社

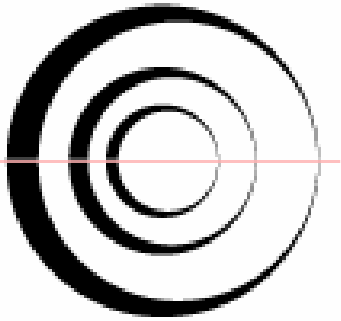
Oracle DBAコマンドブック
著者：中島　益次郎
出版：ソフトバンクパブリッシング株式会社

Oracleデータベース管理を
極める13章
著者：鵜飼　淳代
出版：翔泳社

DBスペシャリストを目指すSEのための
基礎からわかるデータベース構築ガイド
共著：インサイトテクノロジー
　　日経オープンシステム
出版：日経BP社

Question ?

Insight Technology,Inc.



無断転載を禁ず

この文書はあくまでも参考資料であり、掲載されている情報は予告なしに変更されることがあります。
株式会社インサイトテクノロジーは本書の内容に関していかなる保証もしません。また、本書の内容に関連したいかなる損害についても責任を負いかねます。
本書で使用している製品やサービス名の名称は、各社の商標または登録商標です。