ANALYSE UND VISUALISIERUNG VON STATSPACK UND AWR DATEN

DOAG Regio Nord, Juni 2013

Marcus Mönnig
Lichtblick SE, Hamburg

Vorstellung

Marcus Mönnig, Informatiker (B.Sc.), OCP 10/11

DBA bei 

E-Mail: mm@marcusmoennig.de
Internet: http://marcusmonnig.wordpress.com/

Vorstellung „Mumbai“

What is Mumbai?

Mumbai is a freeware Windows application targeted at Oracle DBAs and consultants, with a special focus on performance analysis

Features

- Instant access to important database views vital for DBAs and consultants
- Retrieved data in Mumbai can be slices and dices in various ways (sorting, filtering, grouping, group arithmetic, etc.)
- Retrieve alert.log and trace files from the database server to Mumbai and process/analyze them there.
- Start and stop 10046 traces for any session or process and trace the statements from your SQL console session with just a few clicks
- Powerful 10046 trace file viewer if you need to dig into the details of a TKPROF or OraSRP report. OraSRP reports can be generated from within Mumbai.
- Heap dump analyzer that aggregates data from trace files
- …

Agenda

- Historie und Grundprinzip von SP und AWR
- Unterschiede zwischen Statspack und AWR
- Probleme und Eigenheiten
- Analysemöglichkeiten in Mumbai
- ASH (auch ohne Diagnostic Pack)

Historie

- UTLBSTAT.SQL/UTLESTAT.SQL (vor Oracle 8i)
  Begin und End Skripte zum manuellen Erzeugen von einzelnen textuellen Reports zwischen zwei Zeitpunkten

- Statspack (ab Oracle 8.1.6 bis heute)
  Historisierte, automatisierte Sammlung, Speicherung und Auswertung von Performance Daten

SP

- Automatic Workload Repository (AWR) (seit Oracle 10g)
  Weiterentwicklung von Statspack mit gleichbleibendem Prinzip, aber weiteren und teilweise „besseren“ Datenquellen

AWR

Grundprinzip von Statspack und AWR (I)

- Snapshots
  Kopieren von Performance Daten in historisierte Tabellen in regelmäßigen Intervallen
- Beispiel: Quelle **V$SGASTAT**

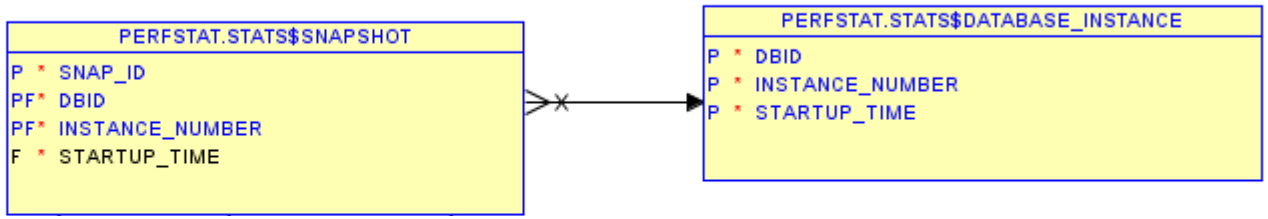
SP

```
INSERT INTO STATS$SGASTAT
   (SNAP_ID, DBID, INSTANCE_NUMBER, POOL , NAME, BYTES)
SELECT
  :B4 ,:B3, :B2, POOL, NAME, BYTES FROM ( SELECT POOL , NAME , BYTES , 100*(BYTES) /(SUM(BYTES) OVER (PARTITION BY
POOL)) PART_PCT
FROM V$SGASTAT)
  WHERE PART_PCT >= :B1 OR POOL IS NULL OR NAME = 'free memory'
```

AWR

```
INSERT INTO WRH$_SGASTAT
  (SNAP_ID, DBID, INSTANCE_NUMBER, POOL, NAME, BYTES)
SELECT
  :SNAP_ID, :DBID, :INSTANCE_NUMBER, POOL, NAME, BYTES FROM  (SELECT POOL, NAME, BYTES, 100*(BYTES) /
(SUM(BYTES) OVER (PARTITION BY POOL)) PART_PCT
FROM V$SGASTAT)
WHERE
 PART_PCT >= 1 OR POOL IS NULL OR NAME = 'free memory'
ORDER BY NAME, POOL
```

Grundprinzip von Statspack und AWR (II)



Grundprinzip von Statspack und AWR (III)

- Analyse der gesammelten Daten
  - Verlauf über einen Zeitraum betrachten
  - Differenzbildung zwischen zwei Snapshots

Agenda

- Historie und Grundprinzip von SP und AWR
- Unterschiede zwischen Statspack und AWR
- Probleme und Eigenheiten
- Analysemöglichkeiten in Mumbai
- ASH (auch ohne Diagnostic Pack)

Unterschiede zwischen Statspack und AWR (I)

- SP: kostenfrei, PL/SQL Quellcode, Edition-übergreifend
  AWR: Teil des kostenpflichtigen Diagnostic Pack, nur in Oracle Enterprise Edition, in C implementiert
- AWR: Top-SQL aus Active Session History (ASH) Daten
  STATSPACK: Aggregierte Daten aus V$SQL
  => AWR liefert deutlich aussagekräftigere Daten
- AWR: Textuelle Vergleichsreports für zwei Intervalle
- AWR speichert aus den Quell-Performanceviews **viele** Spalten, die Statspack nicht speichert
- Statspack speichert aus den Quell-Performanceviews **einige** Spalten, die AWR nicht speichert (z.B. PROGRAM_ID und PROGRAM_LINE# aus V$SQL)

Unterschiede zwischen Statspack und AWR (II)

- AWR zeichnet Metriken auf
  - WRH$_FILEMETRIC_HISTORY
  - WRH$_SESSMETRIC_HISTORY
  - WRH$_SYSMETRIC_HISTORY
  - WRH$_WAITCLASSMETRIC_HISTORY
  - WRH$_SYSMETRIC_SUMMARY
- AWR zeichnet Service-Statistiken auf
  - WRH$_SERVICE_NAME
  - WRH$_SERVICE_STAT
  - WRH$_SERVICE_WAIT_CLASS
- Statspack zeichnet Oracle Streams Performance Views auf
  - STATS$PROPAGATION_SENDER
  - STATS$PROPAGATION_RECEIVER

Unterschiede zwischen Statspack und AWR (III)

- Theorie: Im direkten Vergleich scheint AWR überlegen.
- Praxis: In den meisten Fällen lassen sich zurückliegende (Performance)Probleme mit Statspack-Daten genauso gut analysieren.
- Aber: AWR nutzt ASH Daten für die Top-SQL-Statement Analyse und liefert hier klar bessere Daten.

Agenda

- Historie und Grundprinzip von SP und AWR
- Unterschiede zwischen Statspack und AWR
- Probleme und Eigenheiten
- Analysemöglichkeiten in Mumbai
- ASH (auch ohne Diagnostic Pack)

Probleme / Eigenheiten von Statspack und AWR (I)

- Analyse der gesammelten Daten muss in Betracht ziehen wie bzw. welche Daten gesammelt wurden

- Vier Aspekte näher betrachtet:
  - Qualität der Quelldaten und der historisierten Daten
  - Intervall für die Datensammlung
  - Mehrfachzählung
  - Fehlende Bindevariablen

Probleme / Eigenheiten von Statspack und AWR (II)

Qualität der gesammelten Daten

- Analyseergebnis immer nur so gut wie die Daten die zur Verfügung stehen
  V$SGASTAT Beispiel von oben:

SP

```
INSERT INTO STATS$SGASTAT
( SNAP_ID , DBID , INSTANCE_NUMBER , POOL , NAME , BYTES )
SELECT :B4 , :B3 , :B2 , POOL , NAME , BYTES FROM
( SELECT POOL , NAME , BYTES , 100*(BYTES) /(SUM(BYTES) OVER (PARTITION BY POOL)) PART_PCT FROM
V$SGASTAT)
WHERE PART_PCT >= :B1 OR POOL IS NULL OR NAME = 'free memory'
```

- Wie aussagekräftig sind die Quelldaten?
- Datenverlust durch Projektion? Selektion? Aggregation?

- Noch ein Beispiel: V$SQL => STATS$SQL_SUMMARY

Probleme / Eigenheiten von Statspack und AWR (II)

Qualität der gesammelten Daten

SP

```
INSERT INTO STATS$SQL_SUMMARY   ( SNAP_ID , DBID , INSTANCE_NUMBER , …)

SELECT
:B3 , :B2 , :B1 , MAX(SUBSTRB(SQL_TEXT,1,31)) TEXT_SUBSET , MAX(SQL_ID) SQL_ID , SUM(SHARABLE_MEM) SHARABLE_MEM ,
SUM(SORTS) SORTS , MAX(MODULE) MODULE , SUM(LOADED_VERSIONS) LOADED_VERSIONS , SUM(FETCHES) FETCHES ,
SUM(EXECUTIONS) EXECUTIONS , SUM(PX_SERVERS_EXECUTIONS) PX_SERVERS_EXECUTIONS , SUM(END_OF_FETCH_COUNT)
END_OF_FETCH_COUNT , SUM(LOADS) LOADS , SUM(INVALIDATIONS) INVALIDATIONS , SUM(PARSE_CALLS) PARSE_CALLS ,
SUM(DISK_READS) DISK_READS , SUM(DIRECT_WRITES) DIRECT_WRITES , SUM(BUFFER_GETS) BUFFER_GERS ,
SUM(APPLICATION_WAIT_TIME) APPLICATION_WAIT_TIME , SUM(CONCURRENCY_WAIT_TIME) CONCURRENCY_WAIT_TIME ,
SUM(CLUSTER_WAIT_TIME) CLUSTER_WAIT_TIME , SUM(USER_IO_WAIT_TIME) USER_IO_WAIT_TIME , SUM(PLSQL_EXEC_TIME)
PLSQL_EXEC_TIME , SUM(JAVA_EXEC_TIME) JAVA_EXEC_TIME , SUM(ROWS_PROCESSED) ROWS_PROCESSED ,
MAX(COMMAND_TYPE) COMMAND_TYPE , ADDRESS , MAX(HASH_VALUE) HASH_VALUE , OLD_HASH_VALUE , COUNT(1)
VERSION_COUNT , SUM(CPU_TIME) CPU_TIME , SUM(ELAPSED_TIME) ELAPSED_TIME , NULL AVG_HARD_PARSE_TIME ,
MAX(OUTLINE_SID) OUTLINE_SID , MAX(OUTLINE_CATEGORY) OUTLINE_CATEGORY , MAX(CHILD_LATCH) CHILD_LATCH ,
MAX(SQL_PROFILE) SQL_PROFILE , MAX(PROGRAM_ID) PROGRAM_ID , MAX(PROGRAM_LINE#) PROGRAM_LINE# ,
MAX(EXACT_MATCHING_SIGNATURE) EXACT_MATCHING_SIGNATURE , MAX(FORCE_MATCHING_SIGNATURE)
FORCE_MATCHING_SIGNATURE , MAX(LAST_ACTIVE_TIME) LAST_ACTIVE_TIME
FROM V$SQL
WHERE
 IS_OBSOLETE = 'N'
 AND SQL_ID IN (SELECT SQL_ID FROM STATS$V$SQLSTATS_SUMMARY SQLSTATS WHERE ( BUFFER_GETS > :B9 OR DISK_READS >
 :B8 OR PARSE_CALLS > :B7 OR EXECUTIONS > :B6 OR SHARABLE_MEM > :B5 OR VERSION_COUNT > :B4 ) )
GROUP BY OLD_HASH_VALUE, ADDRESS
```

Probleme / Eigenheiten von Statspack und AWR (III)

Qualität der gesammelten Daten – SUM(ELAPSED_TIME)

- V$SQL enthält Daten über Child-Cursor
- STATS$SQL_SUMMARY gruppiert über Parent-Cursor

V$SQL bei Snapshot 1:

SQL_ID	OLD_HASH_VALUE	ADDRESS	CHILD_NUMI	ELAPSED_TIME
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	0	658.954
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	1	186.473
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	2	468.987
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	3	187.830
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	4	79.249
				1581493,00

V$SQL bei Snapshot 2:

SQL_ID	OLD_HASH_VALUE	ADDRESS	CHILD_NU	ELAPSED_TIME
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	0	658.954
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	1	186.473
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	3	187.830
adu5n2ua8qxt9	3.070.929.891	00000005F7046618	4	79.249
				1112506,00

⇨ Delta für ELAPSED_TIME zwischen S2 und S1: 1,11sec – 1,58sec = - 0,47sec

⇨ In Reports tauchen negative, zu kleine oder zu große Werte auf.

Probleme / Eigenheiten von Statspack und AWR (IV)

Qualität der gesammelten Daten – MAX(MODULE)

```
SQL ordered by CPU  DB/Inst: PSB/PSB  Snaps: 8696-8697
-> Total DB CPU (s):           8,539
-> Captured SQL accounts for  232.5% of Total DB CPU
-> SQL reported below exceeded  1.0% of Total DB CPU

    CPU                  CPU per             Elapsd                     Old
  Time (s)   Executions  Exec (s)  %Total   Time (s)    Buffer Gets  Hash Value
---------- ------------ ---------- ------ ---------- --------------- ----------
   4407.08       20,294       0.22   51.6   10006.66   1,228,369,784 3703299877
Module: JDBC Thin Client
SELECT SEQ_THM FROM (SELECT CAST(SUBSTR(COLUMN_VALUE,1,20) AS CH
AR(20)) THM_ID, CAST(SUBSTR(COLUMN_VALUE,21,4) AS CHAR(4)) SEQ_T
HM FROM TABLE(MAYO_RPS.GET_CUSTOMER_ORDER_SEQUENCES(:B1 , :B2 ,
:B3 , :B4 ))) WHERE THM_ID = :B5

   3943.14      157,316       0.03   46.2    6915.60   1,034,202,723 1127338565
Module: sel_ancomm_vss_06.tsk@c2aixprod (TNS V1-V3)
SELECT * FROM ( SELECT L.AREA, L.AISLE, L.X, L.Y, L.Z FROM LOCAT
ION L, MHV_STOCK_LEVEL M, LOC_GROUP LG WHERE L.AREA = 'AKL_' AND
 L.AISLE = :B1 AND L.X LIKE '00__' AND L.Y LIKE '00__' AND L.Z L
IKE '_00_' AND L.LOC_GROUP_ID LIKE 'AISLE%' AND L.CNT_THM_IN >=
```

<= MAX(MODULE) über Parent Cursor

<= MAX(MODULE) über Parent Cursor

Probleme / Eigenheiten von Statspack und AWR (V)

Qualität der gesammelten Daten

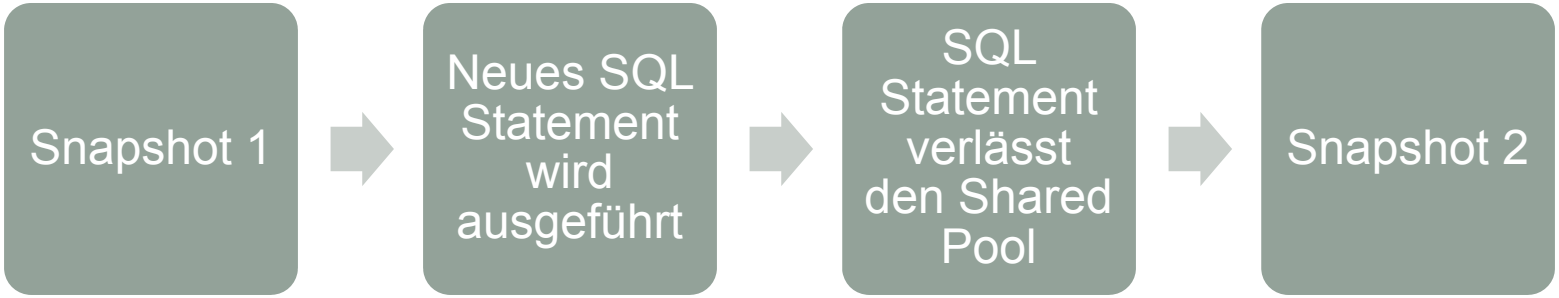
- SP: 11 von 55 INSERT AS SELECT Statements enthalten GROUP BY Aggregationen
- AWR: 15 von 74 INSERT AS SELECT Statements enthalten GROUP BY Aggregationen
- Vielzahl von Selektions-/Projektionseinschränkungen

⇨ Über PL/SQL Quellcode (SP) oder 10046 extended SQL trace Statements analysieren

Probleme / Eigenheiten von Statspack und AWR (VI)

Größe des Snapintervalls

- Kurze Intervalle => bessere zeitliche Einordnung von DB-Aktivität oder Problemen möglich
- Lange Intervalle ungeeignet für SQL-Statement-Analyse



SQL Statement taucht in keinem Report auf

Probleme / Eigenheiten von Statspack und AWR (VII)
Mehrfachzählung

- Zeiten von SQL Statements werden u.U. mehrfach erfasst

Probleme / Eigenheiten von Statspack und AWR (IIX)

Fehlende Bindevariablen

```
INSERT INTO STATS$SQL_SUMMARY   ( SNAP_ID , DBID , INSTANCE_NUMBER , …)

SELECT
:B3 , :B2 , :B1 , MAX(SUBSTRB(SQL_TEXT,1,31)) TEXT_SUBSET , MAX(SQL_ID) SQL_ID , SUM(SHARABLE_MEM) SHARABLE_MEM ,
SUM(SORTS) SORTS , MAX(MODULE) MODULE , SUM(LOADED_VERSIONS) LOADED_VERSIONS , SUM(FETCHES) FETCHES ,
SUM(EXECUTIONS) EXECUTIONS , SUM(PX_SERVERS_EXECUTIONS) PX_SERVERS_EXECUTIONS , SUM(END_OF_FETCH_COUNT)
END_OF_FETCH_COUNT , SUM(LOADS) LOADS , SUM(INVALIDATIONS) INVALIDATIONS , SUM(PARSE_CALLS) PARSE_CALLS ,
SUM(DISK_READS) DISK_READS , SUM(DIRECT_WRITES) DIRECT_WRITES , SUM(BUFFER_GETS) BUFFER_GERS ,
SUM(APPLICATION_WAIT_TIME) APPLICATION_WAIT_TIME , SUM(CONCURRENCY_WAIT_TIME) CONCURRENCY_WAIT_TIME ,
SUM(CLUSTER_WAIT_TIME) CLUSTER_WAIT_TIME , SUM(USER_IO_WAIT_TIME) USER_IO_WAIT_TIME , SUM(PLSQL_EXEC_TIME)
PLSQL_EXEC_TIME , SUM(JAVA_EXEC_TIME) JAVA_EXEC_TIME , SUM(ROWS_PROCESSED) ROWS_PROCESSED , MAX(COMMAND_TYPE)
COMMAND_TYPE , ADDRESS , MAX(HASH_VALUE) HASH_VALUE , OLD_HASH_VALUE , COUNT(1) VERSION_COUNT , SUM(CPU_TIME)
CPU_TIME , SUM(ELAPSED_TIME) ELAPSED_TIME , NULL AVG_HARD_PARSE_TIME , MAX(OUTLINE_SID) OUTLINE_SID ,
MAX(OUTLINE_CATEGORY) OUTLINE_CATEGORY , MAX(CHILD_LATCH) CHILD_LATCH , MAX(SQL_PROFILE) SQL_PROFILE ,
MAX(PROGRAM_ID) PROGRAM_ID , MAX(PROGRAM_LINE#) PROGRAM_LINE# , MAX(EXACT_MATCHING_SIGNATURE)
EXACT_MATCHING_SIGNATURE , MAX(FORCE_MATCHING_SIGNATURE) FORCE_MATCHING_SIGNATURE , MAX(LAST_ACTIVE_TIME)
LAST_ACTIVE_TIME
FROM V$SQL
WHERE
 IS_OBSOLETE = 'N'
 AND SQL_ID IN (SELECT SQL_ID FROM STATS$V$SQLSTATS_SUMMARY SQLSTATS WHERE ( BUFFER_GETS > :B9 OR DISK_READS >
 :B8 OR PARSE_CALLS > :B7 OR EXECUTIONS > :B6 OR SHARABLE_MEM > :B5 OR VERSION_COUNT > :B4 ) )
GROUP BY OLD_HASH_VALUE, ADDRESS
```

- Aber: Sind Einzelstatements alle in STATS$SQL_SUMMARY erfasst?

Probleme / Eigenheiten von Statspack und AWR (X) Empfehlungen

- Trotz der Eigenheiten: AWR und Statspack sind sehr nützliche Tools
- Hinterfragen Sie die Statspack/AWR Daten und die Reports!
  - Analyse des PL/SQL Quellcodes, Tracen der Snapshots-Operationen
  - Analyse des Applikationsquellcode bzw. Zusammenarbeit mit den Entwicklern
- Überprüfen Sie die Daten mit anderen Mitteln:
  - Live-Analyse von V$... Views
  - V$SESSION Sampling mit ASH wenn verfügbar oder alternativ:
    - Snapper-Skript von Tanel Poder
    - S-ASH von Kyle Hailey
    - BASH von Marcus Mönnig
  - Eigene Snapshots relevanter V$... Views („Snap_Anything” Package)

Agenda

- Historie und Grundprinzip von SP und AWR
- Unterschiede zwischen Statspack und AWR
- Probleme und Eigenheiten
- Analysemöglichkeiten in Mumbai
- ASH (auch ohne Diagnostic Pack)

Analysemöglichkeiten

- Textuelle Reports
  - Deltas bzw. Absolutwertvergleichen zwischen Anfangs- und End-Snapshot – dazwischenliegende Snapshots werden ignoriert
  - Daten aus unterschiedlichen Quellen ohne zeitlichen Verlauf
- Detailinformationen im zeitlichen Verlauf gewünscht
  => Große Anzahl von textuellen Reports

- Graphische Auswertungen
  - Daten aus jeweils einer Quelle im zeitlichen Verlauf
  - EM/Cloud Control
  - Mumbai

Analysemöglichkeiten in Mumbai

- Demo

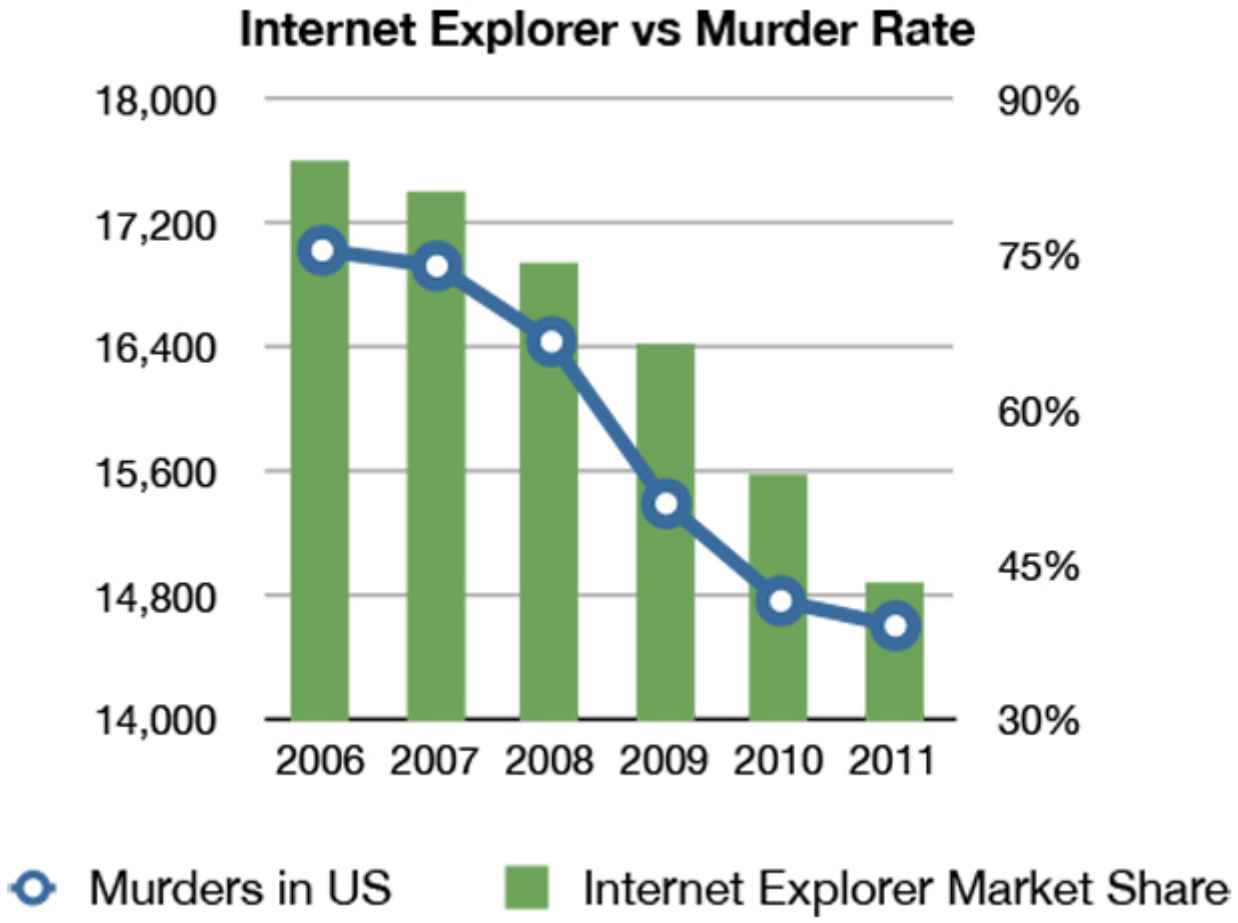
Erweiterte Analysemöglichkeiten in Mumbai

Korrelationen

- Daten bereits geladener Auswertungen werden in zeitliche Korrelation gesetzt
- Korrelation anschaulich: Ähnlichkeit zweier Graphen von statistischen Reihen im zeitlichen Verlauf
- Ausgedrückt in prozentualer Korrelationzahl

- Rein zeitliche Korrelation, möglicherweise ohne Kausalität

Correlation vs. Causation



Erweiterte Analysemöglichkeiten in Mumbai

Korrelationen

- Daten bereits geladener Reports werden in zeitliche Korrelation gesetzt
- Korrelation anschaulich: Ähnlichkeit zweier Graphen von statistischen Reihen
- Ausgedrückt in prozentualer Korrelationzahl

- Rein zeitliche Korrelation, möglicherweise ohne Kausalität
- Geeignet für:
  - Aufrufhierarchien lassen sich schneller erfassen
  - Zuordnung eher ungewöhnlicher wait events zu SQL Statements
  - …

Analysemöglichkeiten in Mumbai

- Demo Korrelationen

ASH ohne Diagnostic Pack

- S-ASH
  - Zentrale Repository Datenbank
- BASH
  - PL/SQL Package und Tabellen in eigenem Schema BASH
  - Snapshots von V$SESSION jede Sekunde nach BASH$ACTIVE_SESSION_HISTORY
  - Kopie jedes 10ten Snapshots nach BASH$HIST_ACTIVE_SESS_HISTORY
  - Snapshotgenerierung über Scheduler Job (oder manuelle blockierende Session)
  - Einstellungen über Tabelle BASH.BASH$SETTINGS
    - sample_every_n_centiseconds NUMBER (Default: 100 = 1 second)
      Number of centiseconds V$SESSION is sampled
    - max_entries_kept NUMBER (Default: 30000)
      How many entries are kept in BASH$ACTIVE_SESSION_HISTORY
    - cleanup_every_n_samples NUMBER (Default: 100)
      How often the data in BASH$ACTIVE_SESSION_HISTORY is purged
    - persist_every_n_samples NUMBER (Default: 10)
      How many of the samples are persisted to BASH$HIST_ACTIVE_SESS_HISTORY

Analysemöglichkeiten in Mumbai

- Demo ASH Viewer

Danke für Ihr Interesse!

- Quellen
  - Marcus Mönnig's Oracle and Mumbai Blog - http://marcusmonnig.wordpress.com
  - Tanel Poder's Session Snapper - http://tech.e2sn.com/oracle-scripts-and-tools/session-snapper
  - Marcus Mönnig's BASH Package - http://marcusmonnig.wordpress.com/bash/
  - Kyle Hailey's S-ASH- http://dboptimizer.com/ash-masters-2/s-ash/
  - Marcus Mönnig's Snap_Anything Package - http://marcusmonnig.wordpress.com/2012/09/10/snap_anything-tiny-little-solution-for-snapping-and-recording-anything-that-you-can-query/

- Kontaktdaten

Marcus Mönnig
E-Mail: mm@marcusmoennig.de
Internet: http://marcusmonnig.wordpress.com/