Analyse und Visualisierung von Statspack und AWR Daten

Marcus Mönnig
Lichtblick AG
Hamburg

Schlüsselworte

Statspack, AWR, Performance-Analyse, Performance-Troubleshooting, Mumbai

Einleitung

Die Aufzeichnung von Performance-Statistiken in Oracle hat mit den UTLBSTAT/UTLESTAT Skripten vor Oracle 8i, Statspack seit Oracle 8i und dem Automatic Workload Repository (AWR) seit Oracle 10g eine recht lange Geschichte.

Während die Menge und Qualität der aufgezeichneten Daten über die Jahre wuchs, sind die Möglichkeiten zur Auswertung dieser Daten aber leider nicht in gleichem Ausmaß mitgewachsen.

Im Folgenden soll gezeigt werden, wie sich die durch Statspack und AWR gesammelten Daten analysieren und visualisieren lassen und wie diese Konzepte in "Mumbai" [1], einem frei-verfügbaren Windowsprogramm des Autors, u.a. zur Oracle Performanceanalyse, implementiert wurden.

Weiterhin wird erläutert welche Probleme bei der Aufzeichnung der Performance-Statistiken existieren und welche Folgen dies für die Aussagekraft der Ergebnisse einer auf den Statspack/AWR-Daten beruhender Performanceanalyse haben.

Soweit nicht explizit darauf hingewiesen wird, gilt im Folgenden alles gleichermaßen für Statspack und AWR.

Das Grundprinzip

Beim „Snappen“ (aus dem Englischen „to snap“ = „to make a snapshot“) wird der Datenstand einer oder mehrerer Tabellen/Views ganz oder teilweise (nur bestimmte Datensätze und/oder Spalten) in eine bzw. mehrere Tabellen kopiert und die Daten mit dem aktuellen Zeitstempel versehen. Auch ist es möglich, dass nicht die Quelldaten selbst kopiert werden, sondern Aggregate aus diesen Daten berechnet werden und nur diese gespeichert werden.

Der Zeitstempel für die Daten ergibt sich bei Statspack und AWR über eine fortlaufend erhöhte, mitgeführte ID des Snapshot die also Foreign Key Beziehung auf die Tabelle STATS$SNAPSHOT (Statspack) bzw. DBA_HIST_SNAPSHOT (AWR) zeigt.

Das Prinzip der historisierten Aufzeichnung von Daten-Schnappschüssen findet nicht nur in AWR und Statspack Anwendung. Weitere Beispiele sind das von Tanel Poder entwickelte „Session Snapper“ PL/SQL-Skript [2], das Daten aus V$SESSION betrachtet oder das PL/SQL Package „Snap_Anything“ [3] des Autors, das sich zur Aufzeichnung beliebiger Daten in festen Intervallen eignet.

Wichtige Unterschiede zwischen Statspack und AWR

- Statspack ist kostenfrei, liegt mit PL/SQL Quellcode vor und ist in allen Oracle Editions nutzbar, während AWR Teil des kostenpflichtigen Diagnostic Packs ist, nur für die Oracle

Enterprise Edition lizensierbar ist und in C implementiert ist, also nicht mit Quellcode vorliegt.
- AWR zeichnet Top-SQL mittels Active Session History (ASH) Daten auf, während STATSPACK aggregierte Daten aus V$SQL aufzeichnet. In diesem Punkt liefert AWR deutlich aussagekräftigere Daten (mehr dazu im nächsten Abschnitt).
- AWR bietet die Möglichkeit textuelle Vergleichsreports für zwei Intervalle zu erstellen.
- Historisierte AWR Tabellen für SQL- und Segment-Statistiken enthalten bereits (einige) berechnete Deltas zum Vorgänger-Snapshot (WRH$_SQLSTAT, WRH$_SEG_STAT) Hintergrund ist wohl die schnellere Darstellungsmöglichkeit über Oracle Enterprise Manager.
- AWR speichert aus den Quell-Perfomanceviews viele Spalten, die Statspack nicht speichert
- Statspack speichert aus den Quell-Performanceviews einige Spalten, die AWR nicht speichert (z.B. PROGRAM_ID und PROGRAM_LINE# aus V$SQL)
- AWR beinhaltet Import/Export Skripte mit denen die Daten für alle oder ausgewählte Snapshot-Intervalle exportiert/importiert werden können und zum Beispiel ein Datawarehouse mit lang zurückliegenden Performancedaten für unterschiedliche Datenbanken aufgebaut werden kann. Dies ist mit Statspack auch möglich, erfordert aber einen gewissen Aufwand in der Implementierung der Export/Import-Logik.
- 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 (STAT4S$PROPAGATION_SENDER, STATS$PROPAGATION_RECEIVER)
- AWR nutzt Partitionen für viele historisierten Datentabellen (PARTITION BY RANGE ("DBID","SNAP_ID") - auch ohne lizensierte Partitioning Option)

Es mag der Eindruck entstehen, dass AWR Statspack deutlich überlegen ist, daher der Hinweis, dass sich Statspack in den allermeisten Punkten bei der Analyse von zurückliegenden Performanceproblemen genauso gut eignet wie AWR. Lediglich die Tatsache, dass AWR ASH-Daten für die Ermittlung der Top SQL Statements heranziehen kann, ist ein klarer Vorteil gegenüber Statspack.

Probleme und Eigenheiten von Statspack und AWR

Die Aussagekraft von Analysen auf den vorhandenen Statspack-Daten hängt von der Qualität der aufgezeichneten Daten ab. Hier gibt es eine Reihe von Eigenheiten und Problemen die man sich bewusst machen sollte:

- **Aggregation beim Snap**:
  Von den 55 INSERT AS SELECT Statements bei einem Statspack Snapshot sind in 11 Statements GROUP BY Ausdrücke vorhanden. Bei AWR sind es 74 Statements wovon 15 GROUP BY Ausdrücke enthalten (jeweils unter Oracle 11.2.0.3 bei höchstem Detailgrad/Snaplevel). Außerdem sind in AWR und Statspack eine Vielzahl von WHERE Prädikaten in den Statements vorhanden.
  Bei der Analyse der von AWR/Statspack aufgezeichneten Daten sollte also genau hinterfragt werden, was eigentlich aufgezeichnet wurde und welche Detailinformationen durch Aggregation und Filterung verloren gegangen sind.
  Hier hat Statspack den Vorteil, dass die INSERT Statements unverschlüsselt als PL/SQL Package vorliegen, während man die ausgeführten INSERTs unter AWR nur durch ein 10046

Extended SQL Trace sicher ermitteln kann.
Besonders problematisch ist die Aggregation bei Statspack für die Daten in STATS$SQL_SUMMARY.
Das INSERT STATEMENT sieht vereinfacht so aus:

INSERT INTO STATS$SQL_SUMMARY (...) SELECT ... SUM(ELAPSED_TIME) ELAPSED_TIME ... FROM V$SQL ... GROUP BY OLD_HASH_VALUE, ADDRESS

Hier werden also beim Snap die ELAPSED_TIME Werte (und auch alle anderen Spalten) für alle SQL Cursor mit dem gleichen Hashvalue und der gleichen Adresse des Parent-Cursors (nicht etwa der Child-Cursor-Adresse) aufaddiert, was dazu führt, dass wenn einer der Child-Cursor den Shared Pool verlässt, die Summe der ELAPSED_Time im Vergleich zum vorherigen Snapshot sinkt. Reports zeigen dann, je nach Implementierung für die Deltas zwischen den beiden Snapshots Null, negative oder übertrieben große Werte für dieses SQL-Statement an.

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

Abbildung 1 - 5 Childcursor mit einer Summe von 1,58 Sekunden

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

Abbildung 2 - Nur noch 4 Childcursor mit einer Summe von 1,11 Sekunden

- **Relevanz des Snapintervalls**:
  Neben der naturgemäß besseren Möglichkeit zur zeitlichen Zuordnung von Aktivitäten oder Problemen bei kurzen Intervallen, ist die gewählte Größe des Snapintervalls insbesondere auch für die Erfassung von Statistiken über SQL Statements von großer Bedeutung. SQL Cursor die nur zwischen zwei Snapshots im Shared Pool sind, aber nicht zum Zeitpunkt des Snapshots, tauchen eben nicht in den aufgezeichneten Daten auf.
  AWR löst dieses Problem, indem es nicht V$SQL für die Snapshots betrachtet, sondern die gesampelten Daten der Active Session History (ASH) heranzieht. Damit ist sichergestellt, dass auch nur kurz im Shared Pool vorhandene SQL Statements in den historischen AWR Daten mit erfasst werden.

- **Mehrfachzählung von ELPASED_TIME für SQL Statements**:
  Bei den aufsummierten Zeiten für ELAPSED_TIME von SQL Statements werden die Zeiten je nach Aufruflogik unter Umständen mehrfach erfasst. Beispiel: Ein Scheduler Job startet und ruft eine PL/SQL Prozedur auf in der ein SELECT ausgeführt wird, das für jeden zurückgelieferten Datensatz den Wert einer Spalte an eine PL/SQL Funktion übergibt. In dieser Funktion wird wiederum ein einfaches SELECT ausgeführt. Angenommen dieser vielfach durchgeführte, innerste SELECT Aufruf ist für so gut wie die komplette angefallene Laufzeit des Jobs verantwortlich, dann wird diese Gesamtlaufzeit jeweils komplett gleich drei SQL Statements zugeschrieben: Der vom Oracle-internen DBMS_SCHEDULER Modul ausgeführten anonymen PL/SQL Prozedur zum Aufruf des Jobs, dem äußeren SQL Statement in der PL/SQL Prozedur und dem inneren SQL Statement in der PL/SQL Funktion.

- **Werte tauchen erst zeitversetzt in Snapshots auf**:
  In manchen Konstellationen kann es passieren, dass Werte für ein SQL Statement erst in dem Snapshot auftauchen der nach beendeter Ausführung des Statements angelegt wird.
  Beispiel: Ein SQL Statement läuft zwei Stunden (ohne parallele Abarbeitung) und es werden alle 30 Minuten Statspack-Snapshots erstellt => In den ersten drei Snapshots taucht das SQL Statement nicht auf - im vierten Snapshots taucht es mit einer ELAPSED_TIME von 120 Minuten (innerhalb eines 30 minütigen Intervalls!) auf.

- **Überläufe von Countern**:
  Bei einigen, zahlenmäßig sehr großen statistischen Werten kann es zu einem Überlauf kommen, d.h. nach Erreichen des maximal darstellbaren Wertes, wird wieder bei 0 begonnen. Passiert ein solcher Überlauf bei einem Snapshot, dann ergibt sich bei der Errechnung des Deltas im Report ein negativer Wert, der je nach Implementierung als Null, negativer Wert oder zu großer, positiver Wert dargestellt sein kann.

- **Fehlende Bindevariablen**:
  Fehlende Bindevariablen für oft ausgeführte, semantisch gleiche SQL Statements sind bekanntermaßen ein Problem für Oracle Datenbanken. In AWR und Statspack führen fehlende Bindevariablen und die daraus folgenden unterschiedlichen SQL_IDs für semantisch gleiche SQL Statements dazu, dass diese SQL Statements möglicherweise gar nicht erfasst werden oder dass sie in ihrer Gesamtheit nicht als kritisch erkennbar sind. Für letzteres Problem liefert Mumbai über die Gruppierung nach FORCE_MATCHING_SIGNATURE in seinem TOP SQL Report eine Lösungsmöglichkeit.

- **FILTER/ACCESS Prädikate fehlen in aufgezeichnete Ausführungsplänen in Statspack**:
  Die Funktionalität ist im Package STATSPACK seit Oracle 9.2.0.6 auskommentiert:

```
...
                , 0 -- should be max(sp.access_predicates) (2254299)
                , 0 -- should be max(sp.filter_predicates)
...
```

  Nach Erfahrung des Autors unter Oracle 10.2 und 11.2 können die Spalten ohne Probleme wieder mit in die Query aufgenommen werden.

Analysemöglichkeiten „out-of-the-box“

Statspack und AWR liefern die folgenden SQL*Plus Skripte mit, mit denen sich textuelle Reports aus den gesammelten Daten erstellen lassen:

	AWR Skript	Statspack Skript
Standard Datenbank-Report	awrrpt.sql	spreport.sql
Standard SQL-Report	awrsqrpt.sql	sprepsql.sql
Vergleichender Datenbank-Report	awrddrpt.sql	-
I/O Intensitäts-Report	spawrio.sql	-

Bemerkenswert ist weiterhin, dass die Statspack-Skripte komplett in SQL*Plus implementiert sind, während die AWR Skripte lediglich PL/SQL Code in der Datenbank aufrufen, in denen dann die Funktionalität implementiert ist. Dies hat den Vorteil, dass man für AWR auch mit einem direkten PL/SQL Aufruf einen textuellen Report generieren kann und nicht über SQL*Plus arbeiten muss. Mehr dazu z.B. unter [4].

Weitere, auch graphische Analysemöglichkeiten, sind über den Enterprise Manager zugänglich, allerdings ausschließlich für AWR Daten.

Erweiterte Analysemöglichkeiten mit Mumbai

Ein Hauptproblem der textuellen Reports die mit denen von Oracle mitgelieferten Skripten erstellt werden können, ist die hierbei stattfindende Durchschnittsbildung. Wird beispielsweise ein Report für ein Intervall von 24 Stunden generiert, dann werden nur Daten des ersten und letzten Snapshots des Intervalls für den Report betrachtet und aus diesen beiden Snapshots die Deltas errechnet die im Report dargestellt sind.
Will man die Information über den zeitlichen Verlauf der Deltas innerhalb des 24-Stunden-Intervalls nicht verlieren, so muss man für jeden Snapshot innerhalb dieses Intervalls einen eigenen Report erstellen. Wird beispielweise alle 30 Minuten ein Snapshot erstellt, so müsste man 48 Reports für ein 24-Stunden-Intervall erstellen und - und das ist der Haken an der Sache - auch analysieren und die im Report vorhandenen Werte in Relation zueinander setzen. Dieser Weg ist für textuelle Reports so nicht praktikabel.

In Mumbai ist ein anderer Ansatz implementiert: Während die mit den von Oracle mitgelieferten Skripten erstellten textuellen Reports Daten aus unterschiedlichen Quellen in einem Report ohne zeitlichen Verlauf zusammenfassen, bietet Mumbai die Möglichkeit Daten aus jeweils einer Quelle im zeitlichen Verlauf zu analysieren und graphisch darzustellen.

Abbildung 3 unten zeigt die normierte (Einheit: Mikrosekunden pro vergangener Sekunde), absolute, aufgelaufene Zeit für verschiedene wait events, sowie für die DB CPU Zeit (die im Oracle-technischen Sinne keine wait event ist) im zeitlichen Verlauf einer Woche.
Eine Datenbank-Session kann in einer Sekunde maximal eine Sekunden (=1.000.000 Mikrosekunden) an Warte- bzw. CPU-Zeit belegen, wobei in der graphischen Darstellung „idle events“ nicht mit aufgeführt werden. Werte über 1.000.000 Mikrosekunden müssen also von mehr als einer Session verursacht worden sein.

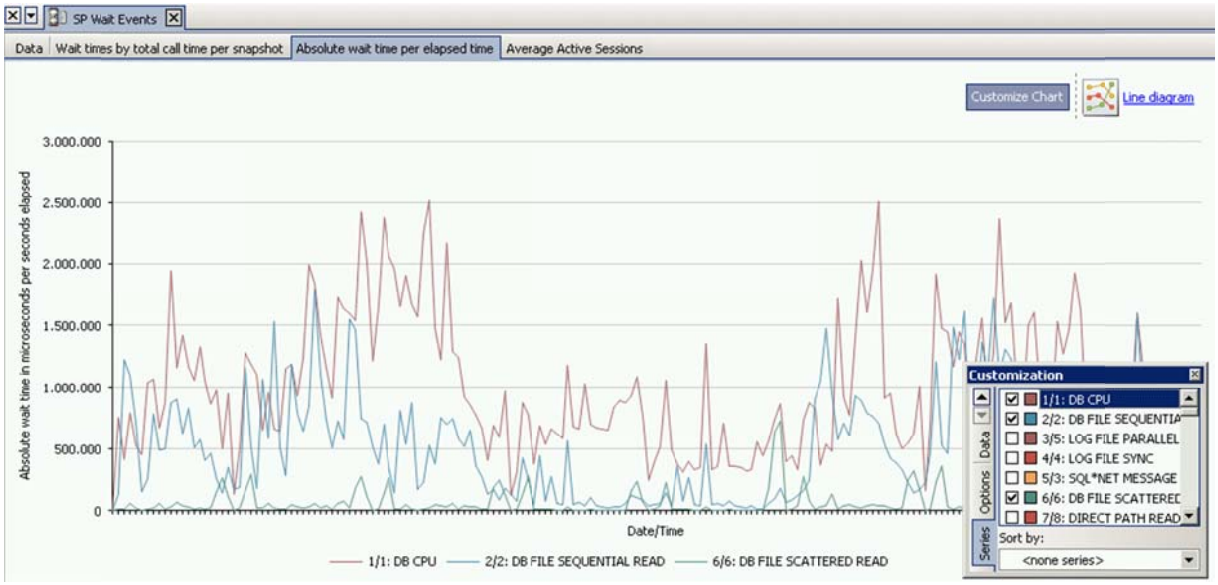

Abbildung 3 - Wait events im zeitlichen Verlauf einer Woche

Neben der oben gezeigten Ansicht existieren noch ein Graph für die Average Active Sessions (AAS), also ein Graph für die aufsummierten Werte von allen Wait- und DB-CPU-Zeiten zu jeweils einem Snapshot-Zeitpunkt.

Abbildung 4 bis 10 unten zeigen weitere graphische Reports die mittels Mumbai aus Statspack Daten einer Woche erzeugt wurden.

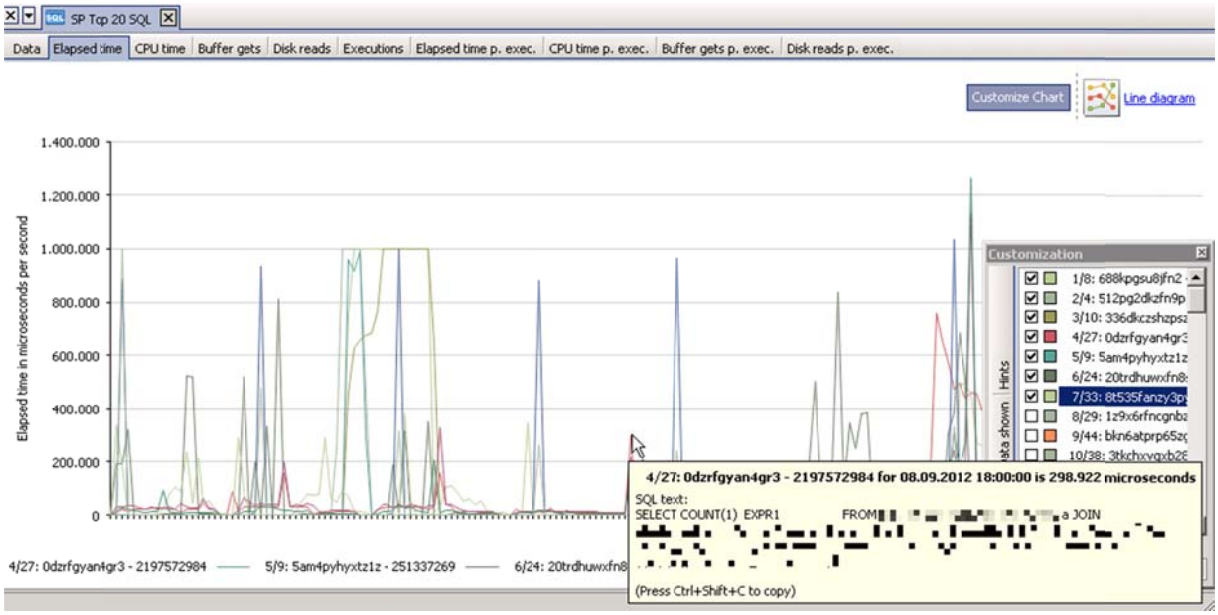

Abbildung 4 - Top SQL Statements nach ELAPSED_TIME im zeitlichen Verlauf einer Woche

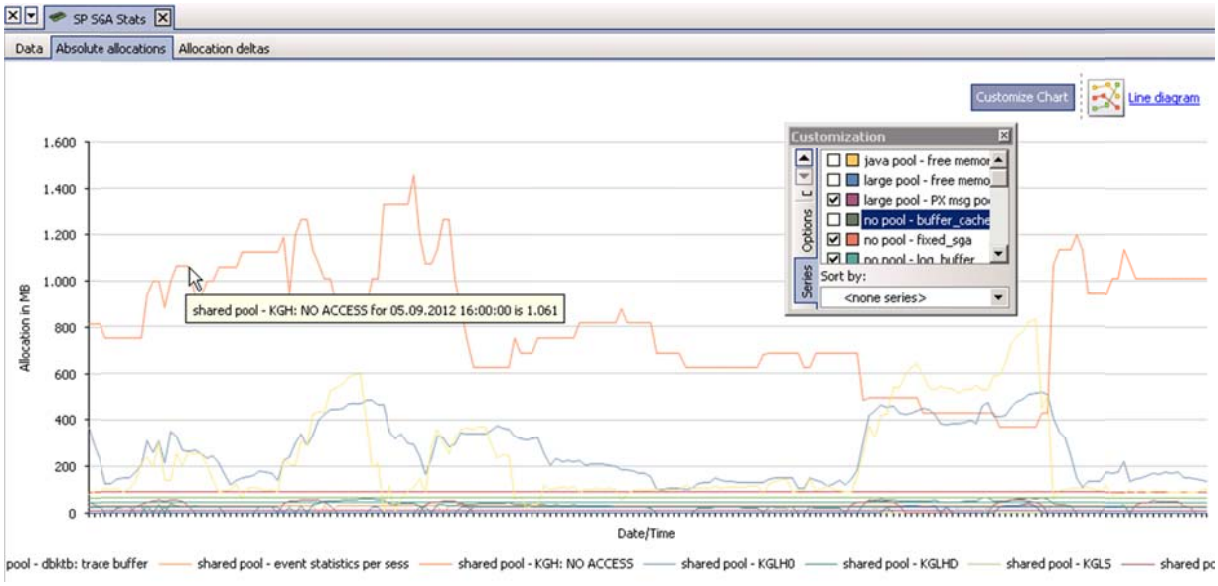

Abbildung 5 - SGA Speicherbereiche im zeitlichen Verlauf einer Woche

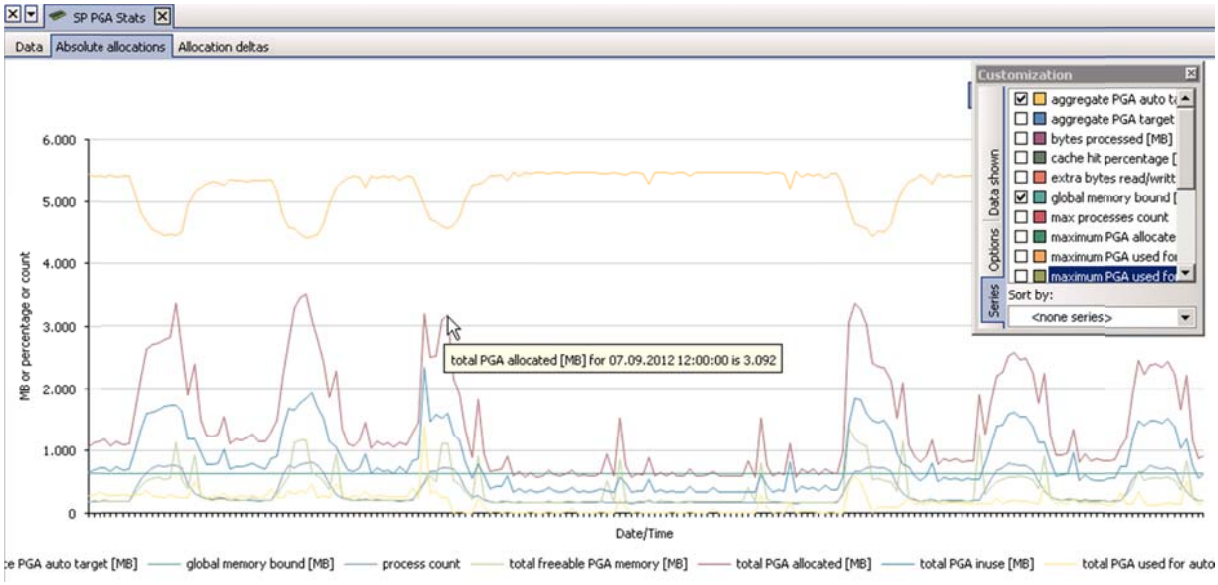

Abbildung 6 - PGA Speicherbereiche im zeitlichen Verlauf einer Woche

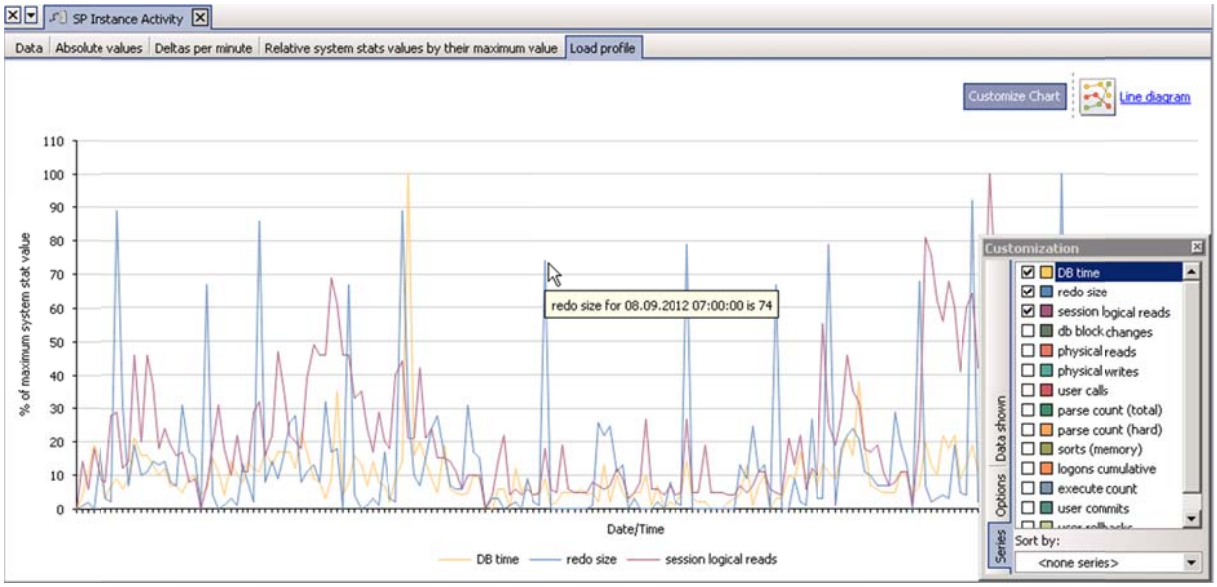

Abbildung 7 - Instanzstatistiken im zeitlichen Verlauf einer Woche

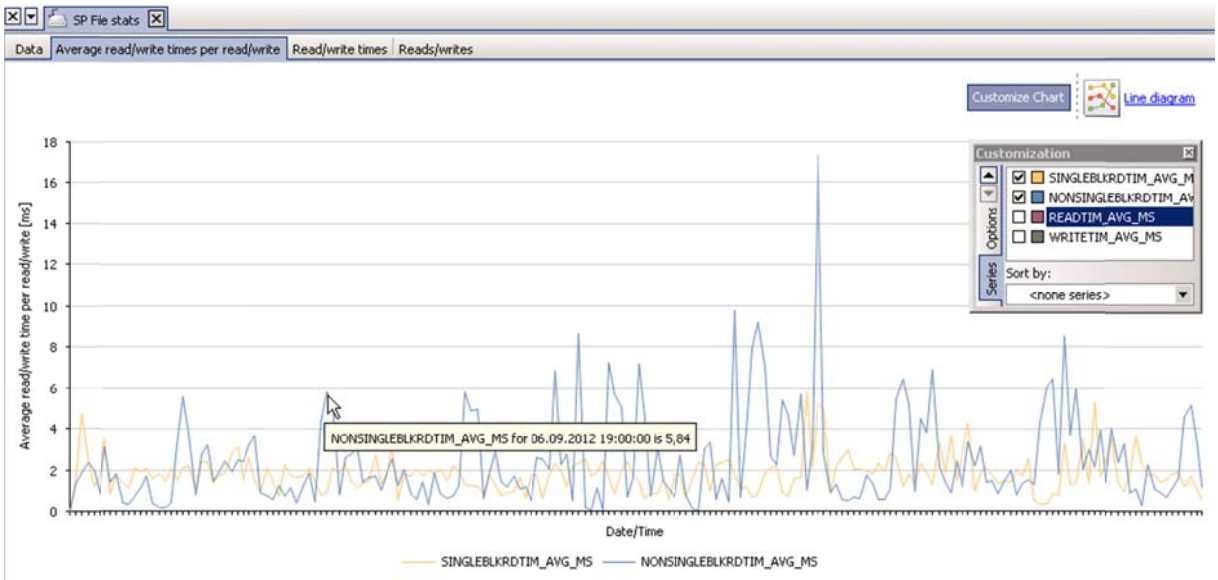

Abbildung 5 - Durchschnittliche Lesezeiten im zeitlichen Verlauf einer Woche

Die dargestellten Lesezeiten, getrennt für single und multi block reads eignen sich sehr gut zum manuellen Setzen von System-Statistiken

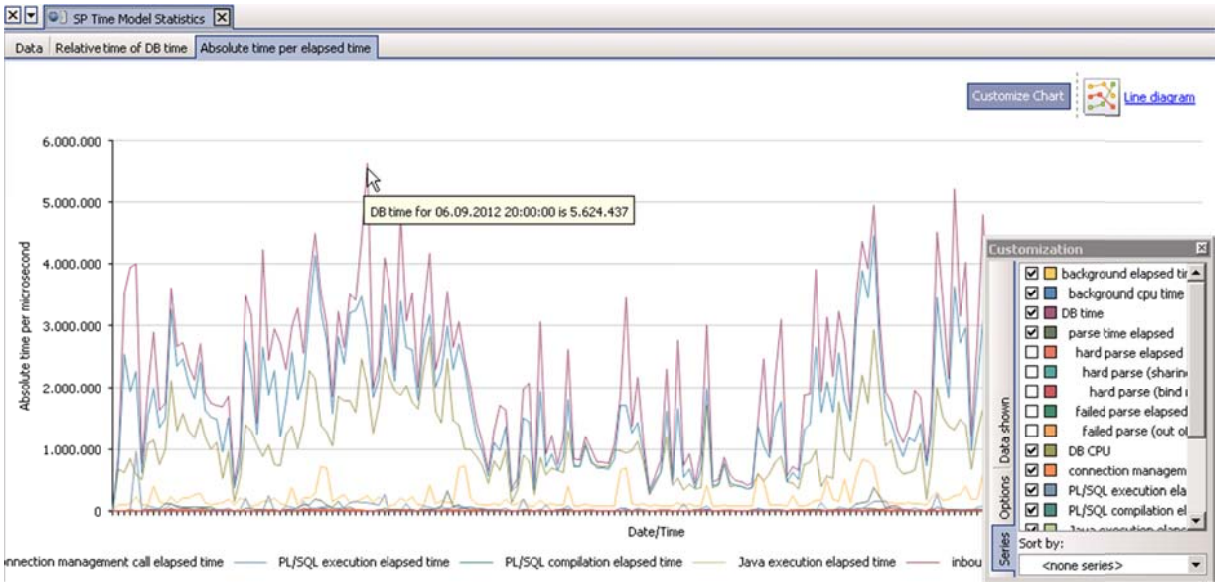

Abbildung 6 - Time Model Statistics im zeitlichen Verlauf einer Woche

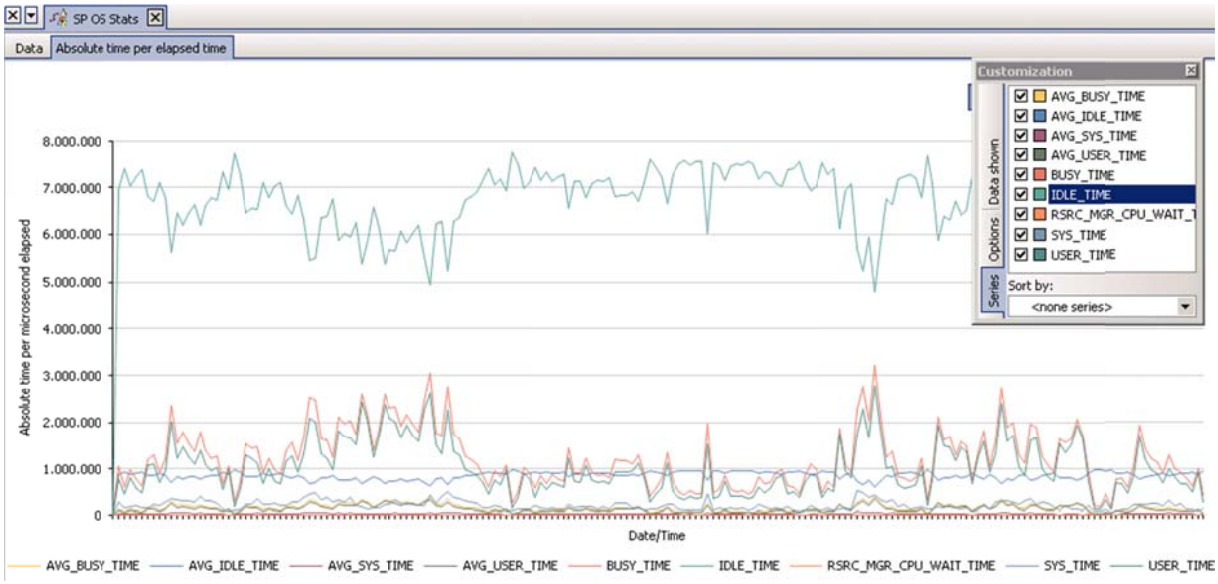

Abbildung 7 - OS Statistiken im zeitlichen Verlauf einer Woche

Korrelationen in Mumbai

Für die Daten in den bereits geöffneten Reports lässt sich in Mumbai noch eine zeitliche Korrelationsanalyse durchführen. Hierbei wird berechnet welche statistischen Reihen sich über dem gemeinsamen zeitlichen Verlauf im Werteverlauf ähneln. Dieser gleichartige Verlauf wird in einer prozentualen Korrelationszahl ausgedrückt, welche dann in den graphischen Reports mit eingeblendet werden kann (siehe Abbildung 11).

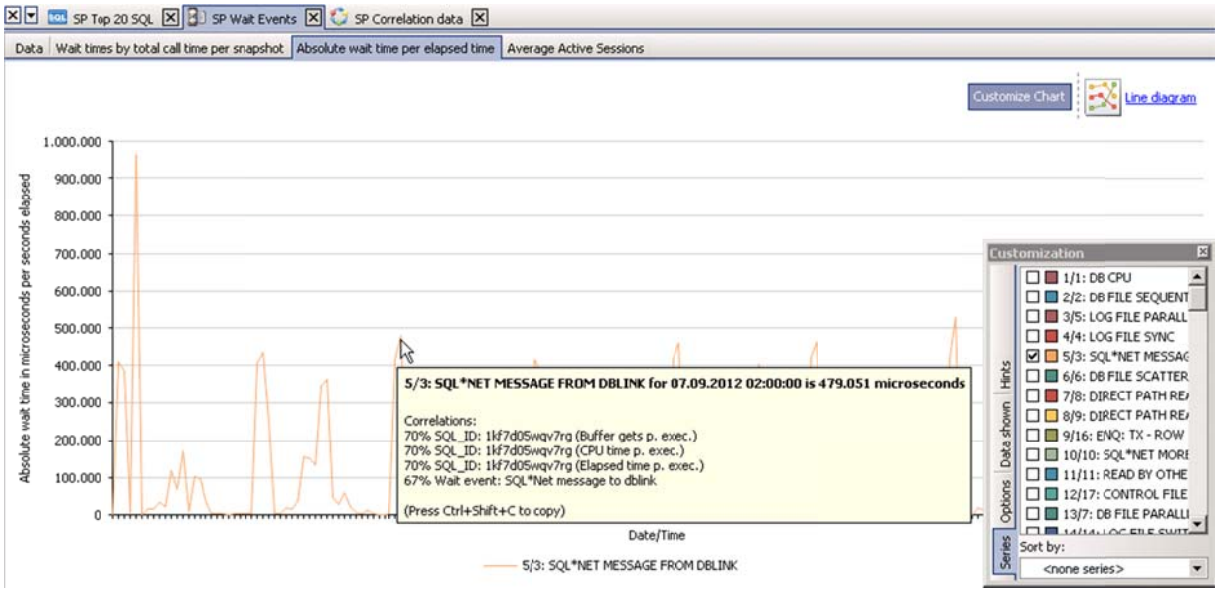

Abbildung 8 - Korrelation des wait events "SQL*NET MESSAGE FROM DBLINK" mit anderen wait events und Werten für SQL Ausführungen

Zu betonen ist, dass es sich hier ausschließlich um eine zeitliche Korrelationsanalyse handelt. Ob sich hinter diesen zeitlichen Korrelationen auch tatsächlich logische Zusammenhänge oder sogar Ursache-Wirkung-Zusammenhänge verbergen, lässt sich nur mit Kenntnis der fachlichen Zusammenhänge sagen.
Oft lassen sich über die Korrelationen Aufrufhierachien schneller erfassen. Auch die Zuordnung sonst eher ungewöhnlicher wait events zu Ausführungen von einzelnen SQL Statements gelingt über die Korrelationswerte recht gut.

Quellen

[1] Marcus Mönnig's Oracle and Mumbai Blog - http://marcusmonnig.wordpress.com

[2] Tanel Poder's Session Snapper - http://tech.e2sn.com/oracle-scripts-and-tools/session-snapper

[3] 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/

[4] Oracle DBMS_WORKLOAD_REPOSITORY Package: http://psoug.org/reference/dbms_wrkld_repos.html

Kontaktadresse

Marcus Mönnig
Lichtblick AG
Zirkusweg 6
D-20359 Hamburg

Telefon: +49 (0) 40-6340-1305
E-Mail mm@marcusmoennig.de
Internet: http://marcusmoennig.wordpress.com/