Oracle Performance Analyse
Die Wiederentdeckung der Oracle Statistiken

Felix Castillo Sanchez

felix.castillo@oraconsult.de - http://blog.oraconsult.de

Zusammenfassung

Das Thema der Performance Analyse und des Tunings wird üblicherweise sehr pragmatisch angegangen. Bei Problemen wird nach potentiellen Verursachern gesucht, je nach Interpretation der Informationen wird optimiert. Dabei liegt das Augenmerk auf den Wait-Events und teilweise auf dem CPU-"Verbrauch" (DB Time). Oracle-Statistiken hingegen und die daraus abgeleiteten Ratios sind in der Vergangenheit mehr und mehr in den Hintergrund getreten. Dabei sind gerade diese für die Bewertung der Wartezeiten und des CPU-Verbrauchs von enormer Wichtigkeit.
Der Vortrag zeigt, wie mit Hilfe der Oracle Statistiken die vorhandenen Informationen über Wait-Events und CPU-Verbrauch besser interpretiert werden können und daß es durchaus Situationen gibt, die mit den üblichen Methoden und Hilfsmitteln schwerlich identifiziert werden können.

Inhaltsverzeichnis

1 Einführung

Als Berater stehe ich häufig in der Situation, ohne genaue Kenntnisse des Systems Performance-Analysen durchzuführen. Die normalerweise vorliegenden Hilfsmittel wie AWR-Reports, Enterprise Manager u.a. sind zwar eine Hilfe, doch stößt man damit häufig schnell an ihre Grenzen. Bei systembedingten Problemen müssen diese offensichtlich genug sein, um auf den ersten Blick aufzufallen zu sein. Bei einer Verkettung verschiedener Probleme wird es schon schwieriger, da eventuell keines prominent genug ist, um im AWR-Report entsprechende Spuren zu hinterlassen. Auch bei Performance-Problemen mit SQL-Befehlen kann es sich um einen einzelnen handeln oder die Summe/Verkettung einiger oder aller Abfragen. Werden nun in der jeweiligen Situation lediglich die Wait- und die CPU-Informationen (via DB Time) ausgewertet, kann dies die Analyse massiv erschweren.

2 Datenquellen

Oracle stellt eine Vielzahl von Performance-Views zur Verfügung, die je nach Fragestellung entsprechende Werte liefern. Doch unabhängig davon was abgefragt wird, handelt es sich dabei entweder um Oracle Statistiken, Wartezeiten oder Informationen des Time Models.

Die eingehende Betrachtung der Analysemöglichkeiten zeigt, dass die Abkehr von den Oracle Statistiken zwar in puncto Ratios sicherlich von Vorteil war, jedoch nicht was die Statistiken an sich angeht. Dass die Wait-Analyse alleine nicht zum Ziel führt, beweist die Einführung des Time-Models mit Version 10g. Diese Schnittstelle ist jedoch viel zu ungenau um die Verwendung der CPU zu beschreiben (siehe Seite 10).

Alle Informationsquellen werden i.d.R. instantan aktualisiert, jedoch bieten nur die lizenzpflichtigen AWR- und ASH-Views eine Historisierung über unterschiedliche Zeiträume.[1] Dabei werden die ASH-Daten stündlich zusammengefasst und fließen zusammen mit einem Snapshot der Performance-Views in die *dba_hist*-Views. Dieses Intervall kann zwar verkürzt werden, erreicht jedoch keinesfalls eine Aktualisierungsfrequenz wie sie z.B. mit *SAR* oder anderen Betriebssystemtools möglich ist, welche i.d.R im Sekundenbereich liegen. Die *v$active_session_history*-View wird zwar wesentlich häufiger aktualisiert, konzentriert sich jedoch auf aktive Sessions und die SQL-Statements.[2]

3 Auswertungen

Im AWR-Report stehen außer dem Abschnitt *Load Profile*, *Instance Efficency Percentages* und der simplen Auflistung der *v$sysstat*-View Statistikwerte zur Verfügung. Das Hauptaugenmerk der Informationen liegt bei Wait-Events und SQL-Statements. Die Aussagekraft der Informationen über durchschnittliche Wartezeiten und deren Verteilung über Wait-Histogramme ist teilweise irreführend[3].

Wesentlich genauer lassen sich akute Performance-Probleme mit der View *v$active_session_history* analysieren. Doch auch hierbei werden „lediglich“ Wait- und DB Time-Informationen ausgewertet, Statistiken fehlen komplett.

Was beide Reports gemeinsam haben, ist ihre textuelle Darstellungsform. Dabei geht es immer um die Rangfolge der größten Verbraucher oder Verursacher. Eine grafische Darstellung über den *Verlauf der Werte über die Zeit* wäre bei beiden möglich. Der ASH-Report wird dabei immer über mehrere Snaps gehen, beim AWR-Report wäre das bei Reports über mehrere Snaps auch möglich, würde aber dann weiter verfälscht werden, da die Zeiträume zu groß sind.

[1] Die Ausnahme bildet hierbei Statspack.

[2] Sollen alle Statements untersucht werden, so kann dies über den Parameter *_ash_enable_all* eingestellt werden. Es werden dann jedoch auch alle inaktiven Sitzungen mit aufgezeichnet.

[3] Siehe hierzu meinen DOAG-Vortrag 2011

4 Datensammlung

Um die vorhandenen Daten und Auswertungen zu ergänzen, kann eine zusätzliche Datensammlung die fehlenden Informationen liefern. Besondere Bedeutung kommt dieser Datensammlung zu, wenn aus Lizenzgründen die vorhandenen Views nicht ausgelesen werden dürfen.

Die Daten für die folgenden Beispiele wurden mit Hilfe von Python-Skripten erfasst, die in der aktuellen Version als CSV-Dateien abgelegt werden. Die Grafen wurden mit Ausnahme von Abbildung (1)mit Matplotlib für Python erstellt. Abbildung (1) wurde mit Mathematica erzeugt.

4.1 Abtastrate

Die Abtastrate sollte so hoch sein, dass auch zeitlich kurz auftretende Probleme und Tendenzen erkennbar sind. Eine Abtastrate von 12 Samples pro Stunde – also alle 5 Minuten – ist für eine grobe Übersicht ausreichend. Für genauere Werte hat sich ein Intervall von 30 Sekunden als praktikabel erwiesen. Die durch das Auslesen verursachte Last ist vernachlässigbar.

Ist die wesentlich höhere Abtastrate überhaupt sinnvoll? Der folgende Graph zeigt beispielhaft die Wartezeiten des Events *db file sequential read* verteilt über einen Tag, im Stundenrythmus zusammengefasst. Jeder Boxplot besteht aus 120 Messwerten. Die durchschnittliche Wartezeit fluktuiert zwischen 1.2 und fast 90 ms!

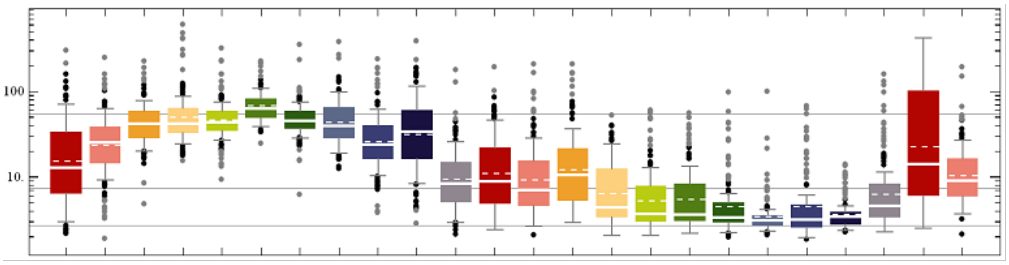

Fig. 1: Tagesübersicht des Events *db file sequential read* pro Stunde zusammengefasst

Genauso wie die Werte selbst schwanken, schwankt auch die Aussagekraft eines jeden Reports, nur dass dieser auf glattgerechneten Durchschnitten basiert und dementsprechend irreführend sein kann.

5 Oracle Statistiken für die System-Analyse

Oracle Statistiken dienten lange Zeit als einzige Möglichkeit zur Performance Analyse. Das Bilden von Ratios, wie z.B. dem bekannten *Buffer Cache Hit Ratio*, war eine der Möglichkeiten, die Probleme einzugrenzen. Dass dieser Weg letzten Endes nicht sehr hilfreich ist, hat sich allgemein durchgesetzt und soll nicht in Frage gestellt werden.

Dennoch sind im Zuge der Entwicklung der Analysemöglichkeiten die Oracle Statitistken mehr und mehr in den Hintergrund geraten. Aktuell werden noch einige Ratios in den AWR-Reports angeboten. Unter *Instance Effiency Percentages (Target 100%)* wird sogar noch eine Buffer Hit Ratio unter *Buffer Hit %* angeboten. Auch unter *Load Profile* können statistische Werte betrachtet werden. Dieser kleine Abschnitt ist jedoch für die Bewertung der nachfolgenden Informationen über Waits und DB Time absolut essentiell. Denn nur mit dem Wissen über die Last lassen sich die Werte korrekt interpretieren. Doch genauso wie der Durchschnitt eines Wait-Events über eine Stunde keine wirklich konkrete Aussagekraft hat, ist ein einzelner statistischer Wert oder eine Ableitung davon völlig nutzlos. Anders sieht es aus, wenn die Werte im zeitlichen Kontext dargestellt werden. Da bei einer Abtastrate von 120 Samples pro Stunde pro Tag 2880 Datenpunkte zur Verfügung stehen, lassen sich diese nun auch entsprechend darstellen. So zeigt der Verlauf über 24 Stunden ein Absacken der Buffer Cache Effizienz kurz nach 21:30 Uhr.

Fig. 2: Buffer Cache Effizienz über 24 Stunden

Wenn man nun die *physical read total I/O requests* dagegen hält, stellt sich die Frage, warum die durchschnittliche Wartezeit im selben Zeitraum bis über 20 ms ansteigt..

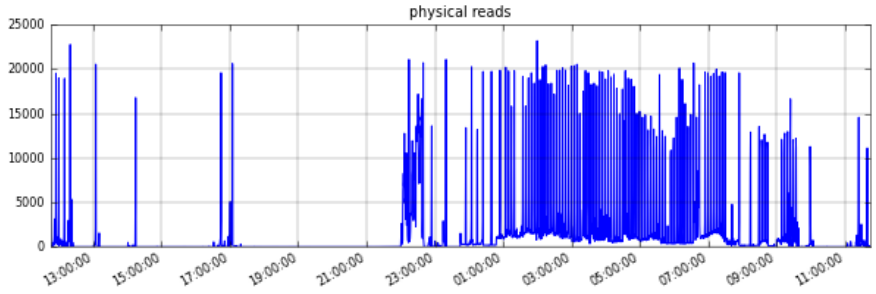

Fig. 3: *physical read total I/O requests*

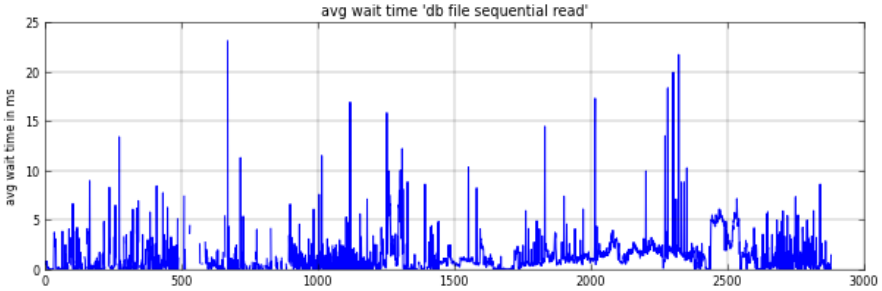

Fig. 4: Durchschnittliche Wartezeit für *db file sequential read*

Wenn man bedenkt, dass die einzelnen Werte über einen 30-Sekunden-Intervall gemittelt sind und es sich beim Plattenspeicher um eine mit Cache gut ausgestattete SAN handelt.

Um auszuschliessen, dass der Durchschnitt wegen weniger Wait-Events verfälscht wird, noch die Darstellung des Wait-Counts.

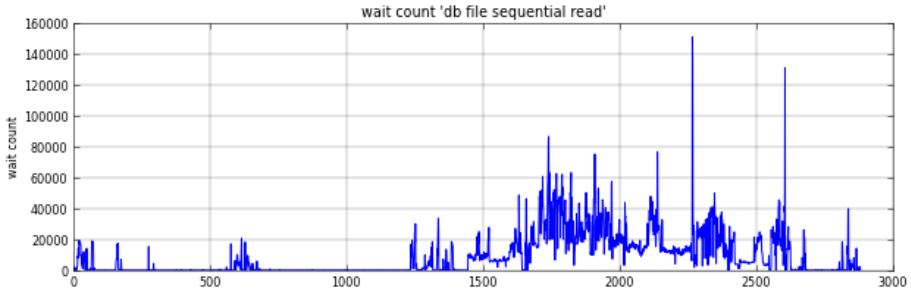

Fig. 5: Wait Count für *db file sequential read*

Es zeigt bis über 1.100 Waits/s und interessanterweise korrellieren die Durchschnittswerte nicht direkt mit der Anzahl der Events.

Fazit

Die Performance Analyse eines Systems mit Hilfe von z.B. AWR-Reports allein kann nur dann erfolgreich sein kann, wenn die Probleme so evident sind, dass sie auffallen müssen. Sind die Probleme jedoch komplexer, können diese nur durch die Analyse genauerer Daten erfolgen. Werden die Daten im zeitlichen Verlauf dargestellt, können sich Einsichten ergeben, die die Problemfindung erst möglich machen.

6 Oracle Statistiken für die SQL-Analyse

Ein ähnliches Problem kann sich bei der Analyse von SQL-Befehlen ergeben. Neben der Analyse des Ausführungsplanes kann es notwendig sein, die entstehenden Wait-Events zu betrachten.

6.1 Sampling einer Abfrage

Das externe hochfrequente Sampling von SQL-Statements und/oder Sessions ist spätestens seit dem von Tanel Poder erstellten Skript *Snapper* eine etablierte Möglichkeit zur Performance-Analyse. Für die Analyse einzelner SQL-Abfragen ist es jedoch nicht immer geeignet, denn auch hier werden die Werte je nach Parametrisierung mehr oder weniger zusammengefasst. Eine weitere Schwierigkeit besteht in der Darstellung selbst. Es ist wesentlich einfacher, grafisch Probleme zu erkennen, als dies textuell möglich ist.

Was aber, wenn das Tracing die Verarbeitung selbst massiv ausbremst und dabei riesige Trace-Dateien erzeugt? Mit Hilfe der View *v$active_session_history* können SQL-Befehle, die über eine Sekunde laufen, analysiert werden. Doch auch hierbei werden neben den Wait-Events und dieS DB Time keine Statistiken mitgeführt. Zudem ist für die Verwendung eine Lizenz erforderlich, die nicht zwingend überall verfügbar ist.

6.2 Hochfrequenz-Sampling

Werden bei einer hochfrequenten Abtastung eines SQL-Befehls oder einer Session neben den Wait-Events auch die Statistiken gesammelt, kann genau nachgezeichnet werden, *was* genau ausgeführt wurde und dies kann vergleichend zu den Wartezeiten und der Häufigkeit dargestellt werden.

Beispiel

Das folgende Beispiel zeigt die Auswertung einer Abfrage, die mit sieben Sekunden Ausführungszeit für den Kunden zu langsam war.

```
Reading SQL file ...
SQL Id: j744yfy0x2hd
Connecting to the instance ... session 0
Setting environment ...
Starting threads ...
Sampling ...
0.11s for execute (000)
0.06s fetched 200 (000)
0.02s fetched 300 (000)
...
1.12s fetched 1900 (000)
...
0.88s fetched 3200 (000)
1.45s fetched 3300 (000)
...
1.11s fetched 5400 (000)
...
0.56s fetched 7000 (000)
0.61s fetched 7012 (000)
execution + fetch: 7.02 seconds (000)
number of rows 7012 (000)
```

Durch das stufenweise Auslesen der Ergebnismenge wurde ersichtlich, dass die hauptsächliche Wartezeit offensichtlich durch wenige Datensätze verursacht wurde. Über die Analyse der Wartezeiten konnte nichts gefunden werden, da diese neben den zu erwartenden *SQL Net ...* Meldungen lediglich 2 Events *Disk File Operations I/O* zeigte.

Durch die Auswertung der gesammelten Statistikwerte zeigte sich dann z.B. bei den übertragenen Datenmengen folgendes Bild:

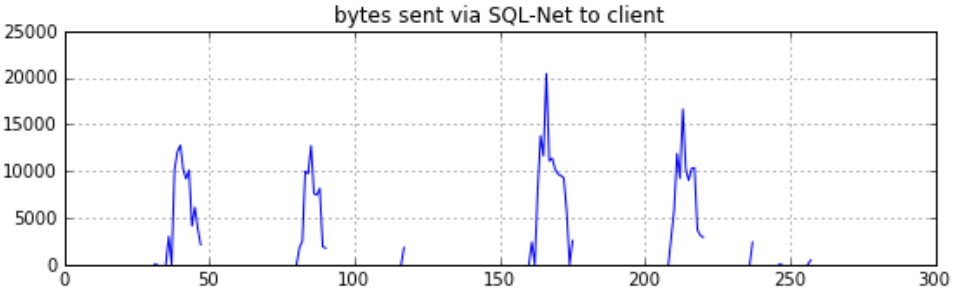

Fig. 6: Datenübertragung zum Client

Die vier textuell erkennbaren Spitzen bei der Übertragung der Datensätze waren auch bei wiederholter Ausführung konstant. Probleme mit der Netzwerkübertragung konnten also ausgeschlossen werden. Zudem konnte gezeigt werden, dass die verlängerte Ausführungszeit mit erhöhtem Datenvolumen einhergingen.

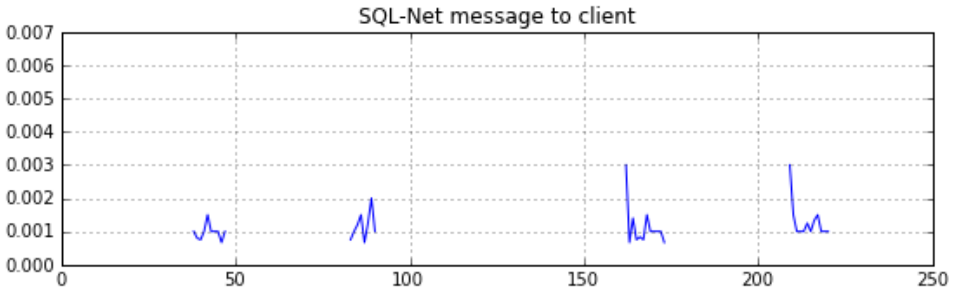

Fig. 7: Wartezeiten bei der Kommunikation mit der Anwendung

Wartezeiten bei der Übertragung bestätigen zusätzlich die Statisiken. Der CPU-„Verbrauch“ zeigt weitere Spitzen, bestätigt die bereits erkannten Aussetzer.

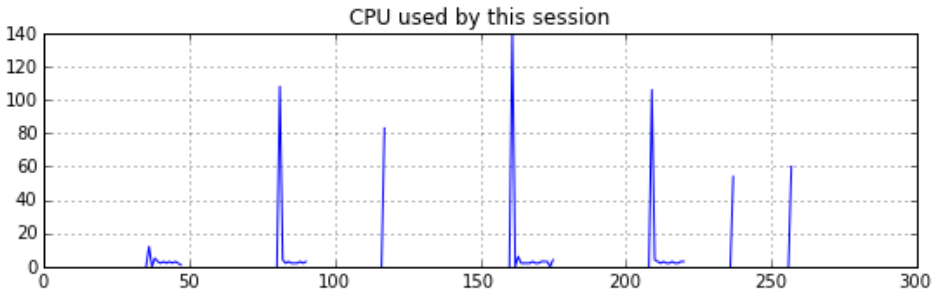

Fig. 8: CPU-”Verbrauch”

Da sich das Bild bei der Verwendung der selben Werte für die gebundenen Variablen wiederholte, konnte die Ursache also nur an den Daten selbst liegen.

Ergebnis der Untersuchung

Nach Betrachtung der Abfrage, die über mehrere Views auf Tabellen zugreift, wurde schlußendlich das Problem gefunden. Unter den vielen selektierten Spalten befand sich eine mit CLOB-Datentyp. Durch die Verwendung von CLOB-Spalten muss Oracle von der Übertragung im Bulk-Modus bei Zeilen ohne CLOB-Daten auf die Übertragung einzelner Zeilen umschalten, wenn diese Spalten befüllt sind. Dabei kommt es massgeblich auf die verwendete Schnittstelle an. Je nachdem wie diese optimiert ist (oder eben nicht), spielt die effektive Länge eine Rolle. So überträgt die *cx_ Oracle*-Schnittstelle für Python auch CLOBs im Bulk, solange die Länge des Inhalts 4K nicht übersteigt.

Das Ergebnis wäre bei einer homogenen Verteilung der CLOB-Daten sicherlich nicht so eindeutig gewesen. Doch genau diese Ausnahmefälle können mit den bekannten Methoden nicht ohne Weiteres gefunden werden.

Fazit

Ohne die Betrachtung der statistischen Werte und der Wartezeiten im *zeitlichen* Kontext ist eine Analyse unter Umständen nicht möglich. Dies gilt vor allem dann, wenn die in den Reports aufgeführten Waits keine direkten Rückschlüsse zulassen oder fast nicht vorkommen. Ein weitere Einsatzmöglichkeit besteht dann, wenn hauptsächlich die CPU gefordert ist und es entscheidend ist zu wissen, was genau Oracle ausführt.

7 Zusammenfassung

Um ein System zu analysieren, muss man es „verstehen“. Sich einzig auf die Wartezeiten zu konzentrieren ist eine massive Einschränkung, die die Analyse stark erschwert. Um zu wissen, *warum* Oracle wartet, müssen wir wissen, *was es tut*. Das DB Time Model ist dazu zu ungenau und eigentlich unnötig, denn die Oracle-Statistiken beschreiben exakt, was Oracle tut. Dies gilt vor allem auch bei der Analyse von SQL-Statements, die CPU-lastig sind.

Da diese Informationen jedoch Oracle-seitig unberücksichtigt bleiben, müssen diese über eine separate, möglichst akurate Datensammlung erfasst werden. Bei der Analyse von SQL-Statements kommt man um eine hochfrequente Abtastung der ausführenden Session nicht vorbei. Die Kombination aus Wartezeiten und Statistiken, dargestellt im zeitlichen Verlauf, ergibt viele neue Möglichkeiten.

Eine langfristige Speicherung der systemseitig gesammelten Daten ermöglicht nicht nur ein akurates Capacity-Management, sondern kann auch zur Trendanalyse und für Prognosen verwendet werden.

Kontakt

Felix Castillo Sanchez
Akazienweg 6
D-61479 Glashütten-Schloßborn
Telefon: +49 (0) 174 900 49 89
Mail: felix.castillo@oraconsult.de
Internet: http://blog.oraconsult.de

A Anhang

View *v$sys_time_model*

```
1) background elapsed time
    2) background cpu time
        3) RMAN cpu time (backup/restore)
1) DB time
    2) DB CPU
    2) connection management call elapsed time
    2) sequence load elapsed time
    2) sql execute elapsed time
    2) parse time elapsed
        3) hard parse elapsed time
            4) hard parse (sharing criteria) elapsed time
                5) hard parse (bind mismatch) elapsed time
        3) failed parse elapsed time
            4) failed parse (out of shared memory) elapsed time
    2) PL/SQL execution elapsed time
    2) inbound PL/SQL rpc elapsed time
    2) PL/SQL compilation elapsed time
    2) Java execution elapsed time
    2) repeated bind elapsed time
```

Fig. 9: *v$sys_time_model* Version 11g

Die einzelnen Einträge beschreiben immer den vorgelagerten Punkt. Die Beschreibungen auf gleicher Höhe sind als unterschiedliche Ausprägungen zu interpretieren und können nicht addiert werden, d.h., sie überlappen sich mehr oder weniger.