Performance-Analyse und Tunings werden üblicherweise sehr pragmatisch angegangen. Bei Problemen sucht man nach potenziellen Verursachern, je nach Interpretation der Informationen wird optimiert.

Oracle Performance-Analyse – erweiterte Möglichkeiten mit Statistiken und Wartezeiten

Félix Castillo Sánchez

Bei der Analyse von Performance-Problemen liegt das Augenmerk auf den Wait-Events und teilweise auch auf dem CPU-Verbrauch über das Time Model. 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 Bedeutung. Der Artikel zeigt, wie mithilfe von selbst erfassten Daten über Statistiken und Wait-Events die vorhandenen Informationen und die daraus entstehenden Auswertungen besser interpretiert werden können und welche Möglichkeiten sich zusätzlich daraus ergeben.

Oracle-Statistiken wurden lange vor der Einführung des Wait-Interface für die Analyse verwendet und sind für die Lösung akuter Probleme an sich ungeeignet. Das beste Beispiel hierfür ist die „Buffer Cache Hit Ratio“, die das Verhältnis zwischen logischen und physikalischen Blockzugriffen beschreibt. In der üblichen Betrachtungsweise als isolierter Wert werden die für eine bekannte Aktion notwendigen Gesamtzugriffe und der zeitliche Verlauf der Entstehung außer Acht gelassen, weshalb der einzelne Wert selbst zu falschen Schlüssen führen kann und dies in der Regel auch tut. Wird jedoch die Hit-Ratio im Kontext einer bekannten Aktion betrachtet, ist eine Aussage über die Effizienz der Speichernutzung durchaus möglich.

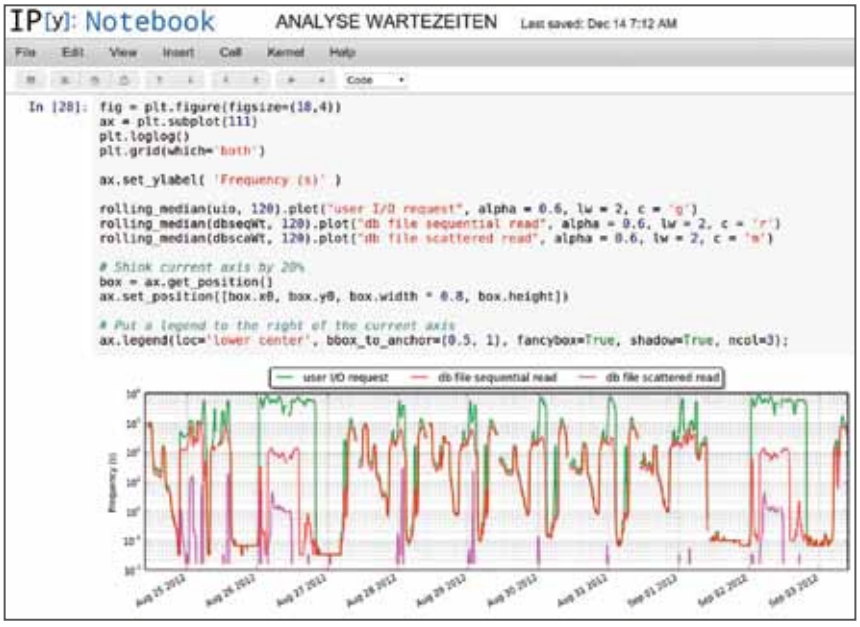

Abbildung 1: Beispiel für eine Ad-hoc-Analyse mit iPython, pandas und matplotlib

Performance-Analyse mithilfe des Wait-Interface

Die Vorgehensweise bei der Problemsuche mithilfe des Wait-Interface ist eine an sich simple Angelegenheit. Es werden die prominentesten Wartezustände gesucht, die durchschnittlichen Wartezeiten bewertet und je nach Häufigkeit der Zustände interpretiert. Dazu ist aber einiges an Erfahrung notwendig, da ein Durchschnittswert je nach Art des Events unterschiedliche Aussagen zulässt. So sind beispielsweise durchschnittliche Wartezeiten von unter 5 ms bei Disk-Zugriffen als gut, jedoch bei SAN oder SSDs als schlecht bis katastrophal zu bewerten.

Das Wait-Interface wird mittlerweile von Oracle in allen relevanten Performance-Views redundant eingesetzt. Eigentlich führt Oracle Wartezeiten analog den Statistiken als kumulierte Zähler mit. Je nach View wird bei jedem Refresh die Differenz aus „Wait Count“ und „Wait Time“ ermittelt und daraus der Quotient berechnet. So handelt es sich bei dem in der View „v$system_event“ gespeicherten Durchschnittswert um das arithmetische Mittel seit Start der Instanz – und demnach um einen völlig unbrauchbaren Wert. Die in den AWR-Views gespeicherten Durchschnittswerte spiegeln den Durchschnitt zwischen den

Snaps wieder. Die Untersuchung großer Zeiträume mit einer wesentlich höheren Abtastrate (etwa 120 Samples/Stunde) zeigt jedoch, dass auch diese Werte per se ohne konkrete Aussagekraft sind, da sie die Schwankungen innerhalb der zusammengefassten Zeiträume völlig außer Acht lassen.

Bei weiteren Views ist die Berechnungsgrundlage nicht von vornherein klar. „Werden bei „v$session" die Werte instantan berechnet?" „Was genau stellt Oracle in der Grid Console dar – bei der historischen Übersicht, der 15-Sekunden- Übersicht oder der manuellen Aktualisierung?" Erfahrungsgemäß kann davon ausgegangen werden, dass Werte, die über einen längeren Zeitraum annähernd unverändert bleiben, aus einer nicht sehr häufig aktualisierten View stammen oder aus den Werten einer solchen berechnet wurden.

Performance Data Warehouse

Eine von Oracle unabhängige Datensammlung und Weiterverarbeitung bietet gegenüber der Oracle-eigenen Sammlung über ASH und AWR den Vorteil, versionsneutral und kostenfrei zu sein, in Abhängigkeit der abgefragten Views. Zudem hat man nicht nur den Vorteil, die Daten selbst auswerten zu können, sondern auch die Abtastrate selbst zu bestimmen und mit Daten vergleichen zu können, die nicht aus Oracle stammen (wie sar, iostat, vmstat etc.)

Die Sammlung von Oracle-Daten funktioniert in der Regel über SQL-Abfragen und erzeugt damit per se Last. Um diese so gering wie möglich zu halten, werden Daten so einfach wie möglich abgefragt. Auswertungen werden keine in der Datenbank durchgeführt, die Weiterverarbeitung sollte idealerweise auf einem anderen Rechner durchgeführt werden.

Die Speicherung und Auswertung der Performance-Daten kann grundsätzlich mit verschiedenen Mitteln durchgeführt werden. Der Autor verwendet hierfür „Python" im Allgemeinen, die Module „pandas" für die Zeitreihenanalyse sowie „matplotlib" für die Darstellung im Speziellen. Für die Ad-hoc-Analysen stellt „iPython" mit der Notebook-Schnittstelle hervorragende Möglichkeiten zur Verfügung. Abbildung 1 zeigt exemplarisch die Erstellung einer Grafik mit den genannten Hilfsmitteln.

Performance-Analyse mithilfe der Oracle-Statistiken

Die meisten Statistikwerte in den Performance-Views sind einfache Zähler, die nach einem Neustart der Instanz wieder zurückgesetzt werden. Zudem werden verschiedene Statistikwerte als aktuelle Werte mitgeführt. Bei jedem Release-Wechsel kommen weitere Informationen hinzu. So sind in der Version 11*g* R2 selbst Informationen über die kumulierte Wartezeit der Wait-Gruppen abrufbar. Der wesentliche Aspekt dabei ist, dass die kumulierten Werte selbst unbrauchbar sind. Erst durch die Berechnung einer Änderungsrate (Rate/s) sind die Werte vergleichbar und es können daraus wesentliche Informationen gewonnen werden.

Statistiken aus „v$sysstat" sind in der AWR-View „dba_hist_sysstat" abgelegt. Im Gegensatz zu den Wait-Informationen werden diese jedoch nicht in der „Active Session History" mitgeführt und lediglich in verarbeiteter Form in der View „v$sysmetric" angeboten. Diese View ist ausgesprochen nützlich, da sie die wichtigsten Kennzahlen wie die Transaktionsrate liefert. Sie hat aber den Nachteil, dass die Aktualisierungsrate und der zusammengefasste Zeitraum nicht nachvollziehbar sind und sie in ihrem Aufbau nicht mit der korrespondierenden View „v$sessmetric" vergleichbar ist. Deswegen kann mit „v$sysmetric" nur auf Systemebene gearbeitet werden. Eine Betrachtung einzelner Sessions oder Abfragen ist nicht möglich.

Bei der Betrachtung von Systemen, die mehrere Instanzen fahren, sind die Statistiken zu addieren, um die von den Oracle-Prozessen erzeugte Last zu quantifizieren. Auch bei RAC-Systemen können RAC-spezifische Statistiken addiert werden. Hierbei ist aber auf die logische Zusammengehörigkeit zu achten.

Wartezeiten jedoch lassen sich nicht einfach arithmetisch zusammenzählen. Lediglich die Anzahl der Events – ein statistischer Wert – kann als Summe behandelt werden. Wartezeiten selbst können hingegen nur miteinander verglichen werden. Zur Bewertung der Auslastung eines Systems kann man einzig Statistiken heranziehen. So lassen sich die Anzahl der I/O-Requests

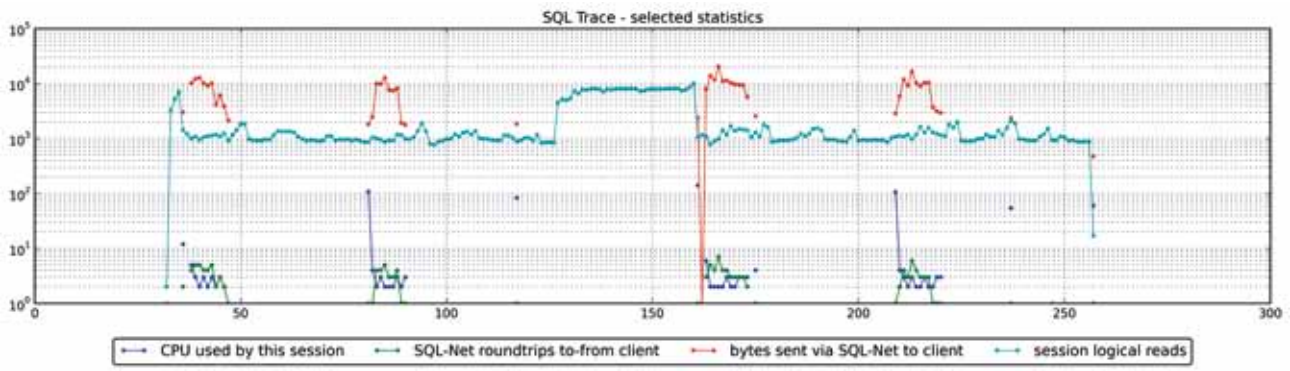

Abbildung 2: Grafische Darstellung der Statistiken während einer SQL-Abfrage

verschiedener Datenbank-Instanzen vergleichen, um die entstehende Last zu bewerten.

Anwendung in der Praxis

Die Optimierung von SQL-Statements erfolgt in der Regel entweder durch Analyse von Ausführungsplänen oder durch SQL-Tracing. Doch auf der einen Seite ist das Lesen von Ausführungsplänen nicht jedermanns Sache, geschweige denn entsprechend Hints einzupflegen oder gar das Statement umzuschreiben. Auf der anderen Seite ist es nicht immer möglich, mit Traces zu arbeiten, da diese sehr schnell enorme Datenmengen generieren und die Ausführung massiv beeinträchtigen.

Die Grafik in Abbildung 2 zeigt beispielhaft eine Abfrage, die aus Sicht des Ausführungsplans unauffällig war, aber nicht verfolgt werden konnte. Die Ausführungszeit stieg von 15 Sekunden auf mehr als 5 Minuten an, das Trace File war über 10 GByte groß.

Durch die hochfrequente Abfrage der View „v$sesstat“ mit ca. 13 Datenpunkten pro Sekunde und der entsprechenden Aufbereitung konnte der Grund für die in diesem Beispiel entstehenden Performance-Probleme dargestellt werden. Die Lösung: Der Anstieg der „session logical reads“ zwischen den Snaps 126 bis 160 wiederholt sich bei jeder Ausführung. Wurden die Einschränkungskriterien verändert, verschob sich dieser Anstieg, blieb jedoch auch hier bei der wiederholten Ausführung an derselben Stelle. Der Grund bestand darin, dass bei einer Anpassung des Datenmodells eine Spalte unnötigerweise auf „NCLOB“ umgestellt worden war und die ausgelesenen Datensätze an der auffälligen Stelle Daten gespeichert hatten (unter 100 Zeichen).

Diese hohe Abtastrate bei Abfragen kann nicht nur in dem gezeigten Beispiel nützlich sein, sie lässt sich in einer wesentlich häufiger auftretenden Situation verwenden: wenn das System keine relevanten Waits aufweist, die CPUs jedoch massiv ausgelastet werden oder ein Statement hauptsächlich auf der CPU läuft (siehe Abbildung 3). Im Gegensatz zu den Waits werden beim Anklicken des CPU-Anteils keine weiteren Informationen eingeblendet. Es kann nicht ermittelt werden, was genau die CPU für dieses Statement ausführen muss.

In den AWR-Reports ist der CPU-Ressourcen-Verbrauch mithilfe des „DB Time Model“ beschrieben. Dieses von Oracle mit Version 10*g* eingeführte Modell bietet leider nur sehr ungenaue Informationen darüber, wie die CPU-Ressourcen verwendet werden. Die nachfolgende Auflistung aus der Oracle-Dokumentation zeigt die Aufteilung:

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

Die Bereiche überlappen sich größtenteils und lassen keine Rückschlüsse auf die effektiven Tätigkeiten zu. Trotzdem werden die Werte in allen relevanten Views mitgeführt. Die relevanten Werte „DB time“ und „DB CPU“ werden zudem in den Statistik-Views mitgeführt und lassen sich dort entsprechend auswerten.

Auswertungsmöglichkeiten

So wie uns das Wait-Interface darüber informiert, welcher Event wie lange und wie oft gewartet hat, entnimmt man den Statistiken, was wir tun und somit, wofür die CPU-Ressourcen verwendet werden. Bei hoher CPU-Auslastung sind demnach die Statistiken des Statements relevant, um das Problem einzugrenzen. Leider werden diese Informationen nur sehr eingeschränkt in den Performance-Views geführt.

In den Trace Files stehen im Zusammenhang mit Wait-Events die „Physical I/O“, die „Logical I/O“, die dafür notwendigen CPU-Zyklen und die effektive Zeit – in der Active-Session-History sind sie vollständig ausgeblendet. Dabei kann man bei der grafischen Aufbereitung der Statistiken analog den Darstellungen des Enterprise Manager Unregelmäßigkeiten und Probleme entdecken.

Das Beispiel in Abbildung 4 zeigt die Aufbereitung der auffälligsten Statistiken einer einzelnen User-Session. Werden neben den Statistiken auch die SQL-IDs aufgezeichnet, können unterschiedliche Ausführungen vergli-

Abbildung 3: Beispiel „CPU-Verwendung“

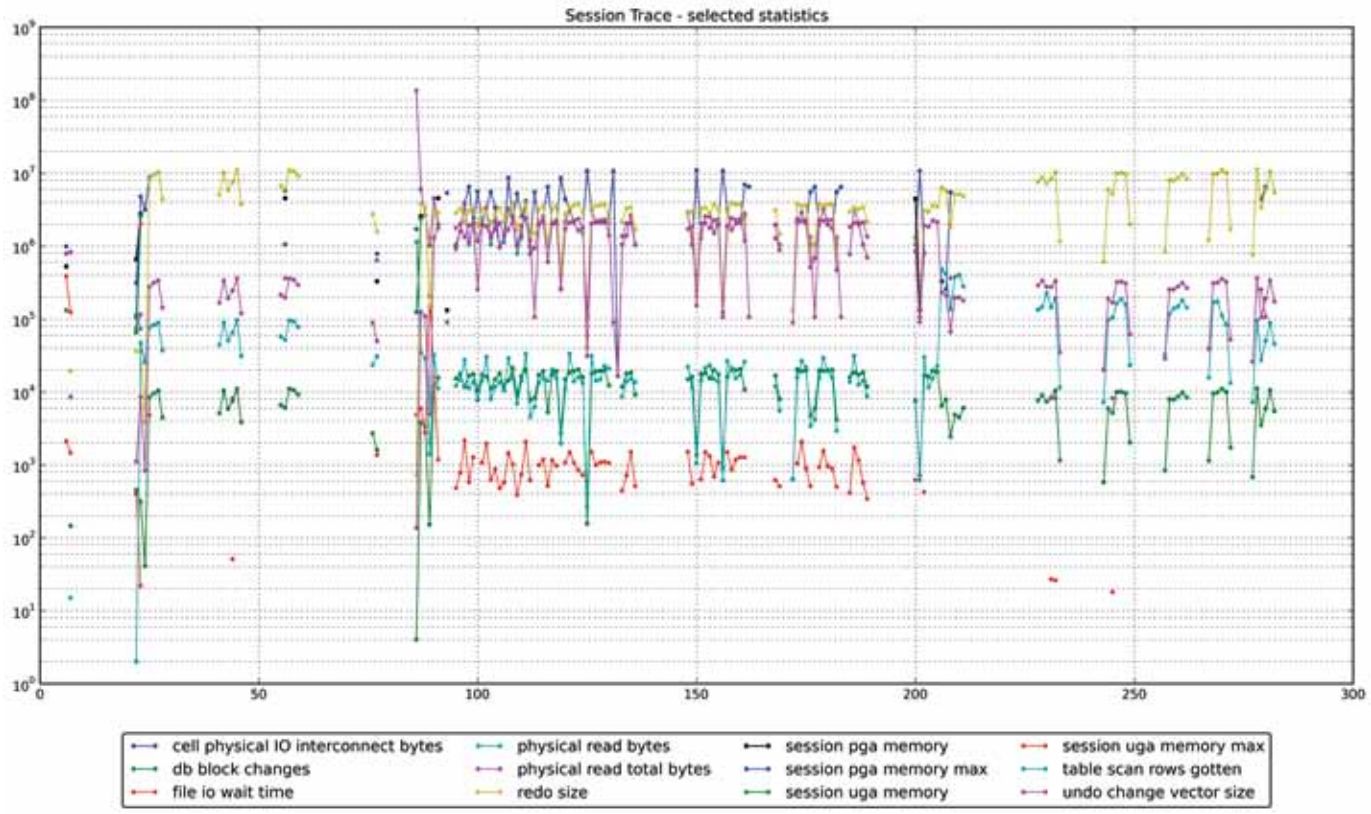

Abbildung 4: Ausgewählte Statistiken einer Session

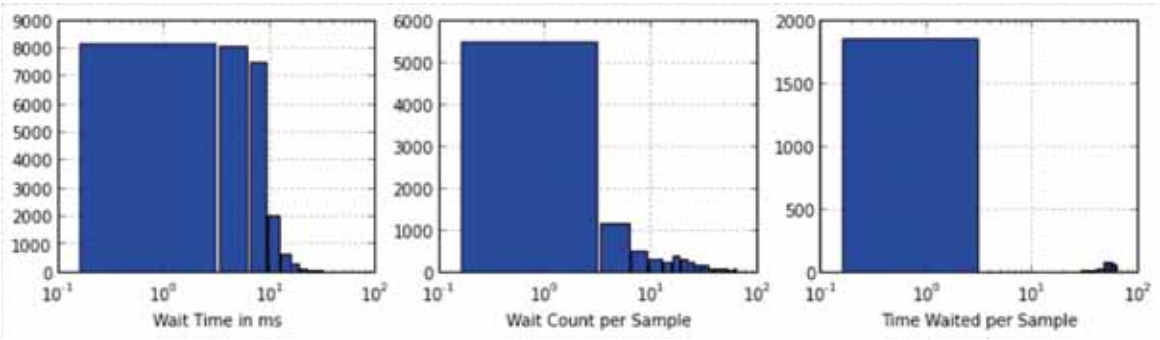

Abbildung 5: Histogramme für Wait-Event log file sync

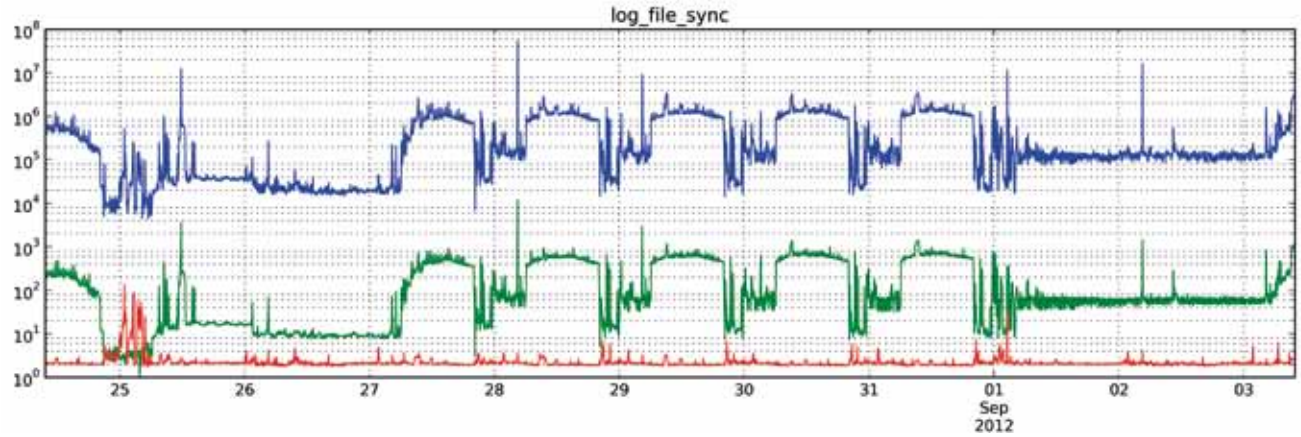

Abbildung 6: Time Plot Wait-Event log file sync

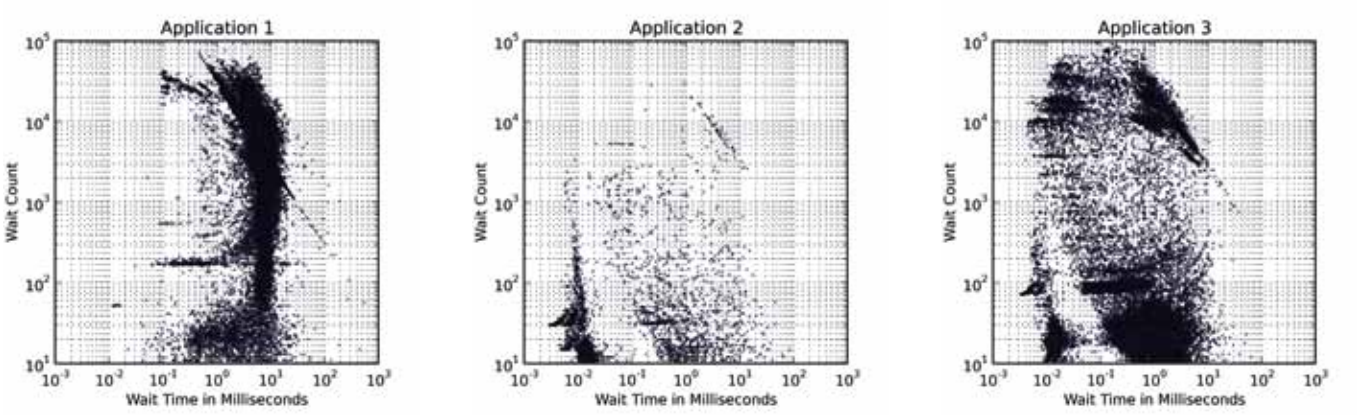

Abbildung 7: Verteilung der Wait-Events – drei unterschiedliche Anwendungen

chen werden, um problematische Unterschiede zu erkennen.

Die Wartezeit-Analyse hat sich als Dreh- und Angelpunkt für die Performance-Analyse durchgesetzt. Die Einbindung in das Modell der „Average Active Sessions" ist der Versuch, CPU-Ressourcen und Wartezeiten zusammenzufassen. Dieses Modell wird jedoch in erster Linie nur für die grafische Aufbereitung in den Verwaltungskonsolen verwendet. Mit diesen können die Daten aus dem ASH und dem AWR betrachtet werden, wobei folgende Probleme bestehen:

- Die Durchschnittsbildung über große Zeiträume liefert unbrauchbare Ergebnisse.
- Die als „Event-Histogramme" bezeichneten Grafiken sind einfache Balkendiagramme. Sie sind zu ungenau und zudem können sie zu falschen Schlüssen führen, da ihre Klassifizierung in Zweier-Potenz-Schritten viel zu grob ist.

Die folgenden Histogramme zeigen exemplarisch die Daten des Events „log file sync" über den Zeitraum von zehn Tagen. Bei Histogrammen ist nicht nur die Höhe der Balken relevant, sondern auch die Breite. Zudem sollten neben der Wartezeit selbst auch die Häufigkeit und die pro Zeitintervall gewartete Zeit betrachtet werden (siehe Abbildung 5). Der erste Graph stellt die durchschnittlichen Wartezeiten dar, die Datengrundlage umfasst jeweils über 28.500 Werte. Analog zeigen die weiteren Histogramme die Häufigkeit der Wait-Counts und der Wartezeit insgesamt. Der Verlust des zeitlichen Kontextes ist der große Nachteil von Histogrammen. Der Time Plot bietet in diesem Zusammenhang wesentlich mehr Möglichkeiten (siehe Abbildung 6).

Wie der Time Plot in Abbildung 6 zeigt, kann man mit der Darstellung der hinter der Wartezeit stehenden statistischen Werte Wait-Count und Time-Waited verwertbare Muster erkennen. Das Beispiel zeigt ein Performance Incident während des 25. Septembers. Die Wartezeiten betrugen bis zu 140 ms, obwohl der Wait-Count (grüne Linie) nur unerheblich anstieg. Weiter ist ersichtlich, dass sich nach dem 1. September die Häufigkeit der Events gegenüber der Vorwoche erhöht hat. Grund dafür ist ein an diesem Tag eingespielter Software-Patch

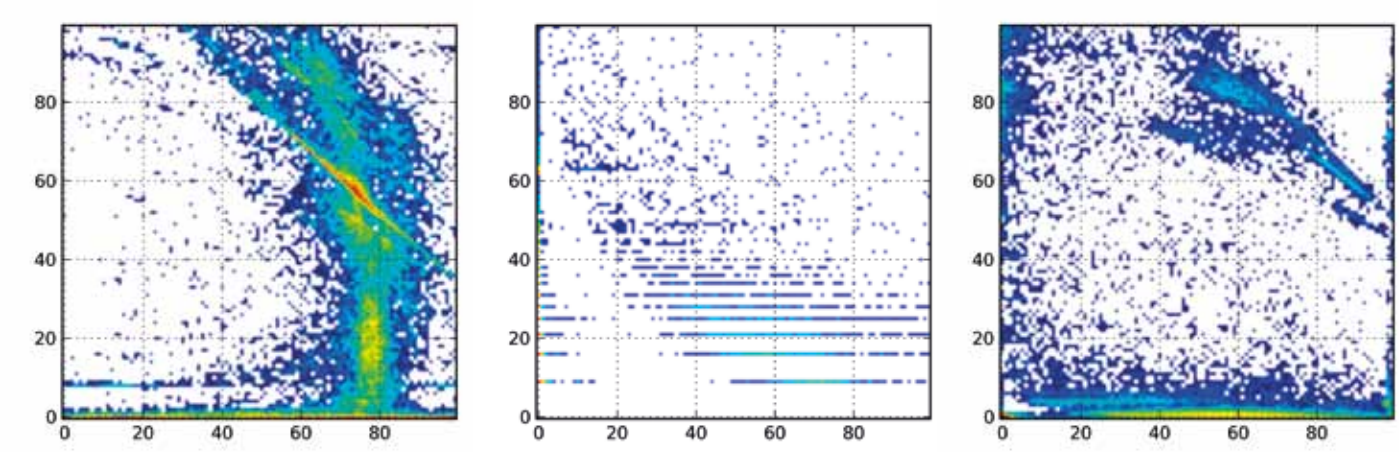

Abbildung 8: Hotspots – drei unterschiedliche Anwendungen

Abbildung 9: I/O-Statistiken mit relevanten Wait-Events

seitens der Anwendung. Da das System offensichtlich noch freie Kapazitäten hat, spiegelt sich dieser Anstieg nicht in erhöhten durchschnittlichen Wartezeiten wider. Der Umstand wäre also unentdeckt geblieben.

Eine weitere Möglichkeit bietet die Darstellung der Wait-Daten mithilfe von Punktewolken. Damit können unterschiedliche Anwendungen und/oder Zeiträume sehr gut verglichen werden. Wie Abbildung 7 zeigt, unterscheiden sich die Plots dreier Anwendungen wesentlich. Interessanterweise bleiben diese unverändert, solange sich am System nichts Wesentliches verändert.

Anmerkung: Selbst in solchen Darstellungen lassen sich Muster entdecken. So gehören die Anwendungen 2 und 3 einem Kunden, Anwendung 1 einem anderen. Bei 2 und 3 ist bei beiden Punktewolken ein fast identisches Muster erkennbar, das sich im unteren linken Bereich befindet. Um die Punktewolken besser beurteilen zu können, müssen noch entsprechende Dichteplots erstellt werden, um die wirklichen Hotspots zu finden. Obwohl die Darstellungen sich (noch) nicht mit den Punktewolken decken, kann trotzdem ermittelt werden, wo sich diese drei Anwendungen potenziell in die Quere kommen würden, sollten sie auf einem einzelnen System zusammengeführt werden (siehe Abbildung 8). Weitere Möglichkeiten ergeben sich auch in der Kombination von Wait-Events und Statistiken (siehe Abbildung 9). So können I/O-relevante Wait-Events mit den in den Statistiken geführten Werten zusammengeführt und analysiert werden. Im Beispiel ist ersichtlich, dass unter der Woche die I/O-Wartezeiten fast ausschließlich von „db filesequential read“ und „db file scattered read“ verursacht werden, am Wochenende jedoch weitere hinzukommen.

Fazit und Ausblick

Mit Oracle-Bordmitteln lassen sich bestehende Probleme in der Regel gut identifizieren. In speziellen Situationen ist es jedoch notwendig, sowohl entstehende Wartezeiten als auch Statistiken im zeitlichen Kontext zu betrachten, indem sie grafisch aufbereitet werden. Dadurch lassen sich viele Aspekte ermitteln, die mit den bekannten Methoden nicht erkannt werden können, da die Daten entweder nicht zur Verfügung stehen oder viel zu ungenau sind. Die Performance-Analyse mit AWR-Reports kann durch ergänzende Graphen wesentlich vereinfacht und erweitert werden.

Weitere Möglichkeiten ergeben sich bei der Betrachtung der Performance-Daten aus unterschiedlichen Perspektiven. So können unterschiedlichste Zeiträume miteinander verglichen werden, um entsprechende Muster zu finden, die dann für Performance-Monitoring und/oder Forecasting verwendet werden können.

Die Betrachtung der Statistiken und Wait-Events mithilfe mathematischer Modelle helfen bei der Klassifizierung und Vergleichbarkeit des Leistungsverhaltens eines Systems. Es sind damit nicht nur genaue Analysen möglich, es können zudem Veränderungen abseits der gedachten identifiziert werden.

Die grafische Aufbereitung der Performance-Daten bietet sehr viele Möglichkeiten, ist jedoch zeitlich aufwändig. Sie kann automatisiert werden, die Bewertung bleibt jedoch weiterhin eine manuelle Aufgabe.

Durch eine automatische Analyse der Performance-Daten können mathematisch auswertbare Performance-Signaturen erstellt werden, die sowohl unmittelbare als auch langsame Veränderungen erfassen, die durch ihre Subtilität mit normalen Mitteln nicht erkennbar sind. Diese Performance-Signaturen bieten weitere Möglichkeiten wie das Einbinden in ein Performance-Monitoring, fortlaufendes Forecasting etc.

Félix Castillo Sánchez
felix.castillo@oraconsult.de