DOAG Konferenz 2014

Oracle Core für Einsteiger: Datenbank-I/O

Martin Klier
Managing Partner / Database Technology
Performing Databases GmbH

1. Inhalt und Zielgruppe

Wir kennen verschiedene IO-Typen, die die Datenbank benutzt um ihre Aufgaben zu erfüllen. Dieses Paper zeigt, wie sie sich unterscheiden, und erklärt, welche Vorteile die einzelnen Methoden bieten. Behandelt werden Zugriffe auf Tablespaces, Online/Archived Redo Logs und verschiedene Dateitypen, die mit der Instanz in Verbindung stehen. Eingeschlossen ist eine kurze Wiederholung der jeweiligen Funktion im RDBMS; ebenfalls enthalten ist ein "Crashkurs" zum Automatic Storage Management "ASM".

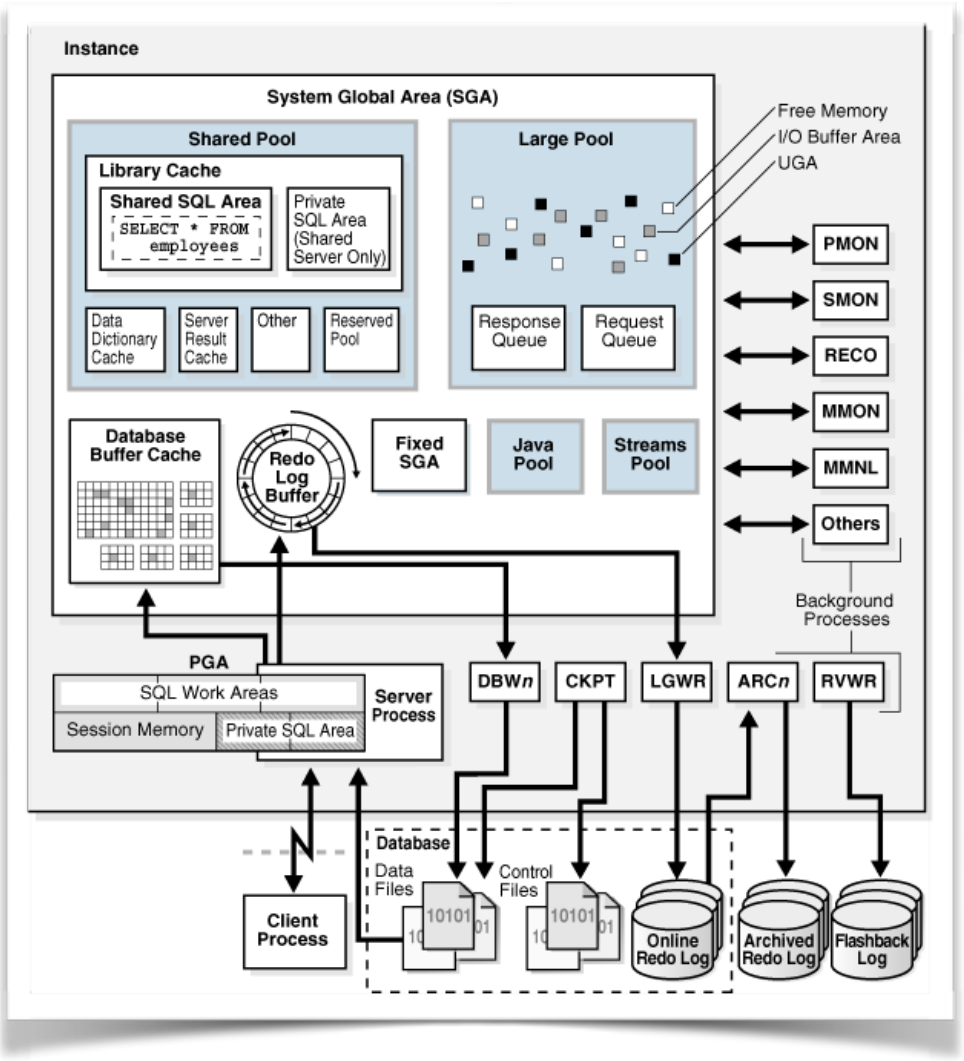
Das Ziel des Papers ist, Einsteigern das Verständnis von Massenspeicheroperationen näher zu bringen und Hintergrundwissen für ein kompetentes Storage-Sizing zu vermitteln.

Durch begrenzten Raum und die anvisierte Zielgruppe werden einige Vorgänge deutlich vereinfacht dargestellt und Sonderfälle nur selten behandelt. Detailfragen, Kritik und Verbesserungsvorschläge sind jederzeit willkommen: Im persönlichen Gespräch, oder unter martin.klier@performing-db.com.

2. Basiswissen

Aufbau des RDBMS

Das Relationale Datenbank-Management-System (RDBMS) von Oracle besteht im Wesentlichen aus zwei Teilen: Während alle Komponenten im RAM, wie Speicherbereiche und Prozesse, zur “Instanz” zählen, bilden die Datafiles, Online Redo Logs und Control Files die eigentliche “Datenbank”. Diese Unterscheidung ist sinnvoll: Denn die Datenbank stellt das Ergebnis aller Bemühungen des RDBMS dar. Die Instanz ist “nur” das Werkzeug, das zum Ziel führt.

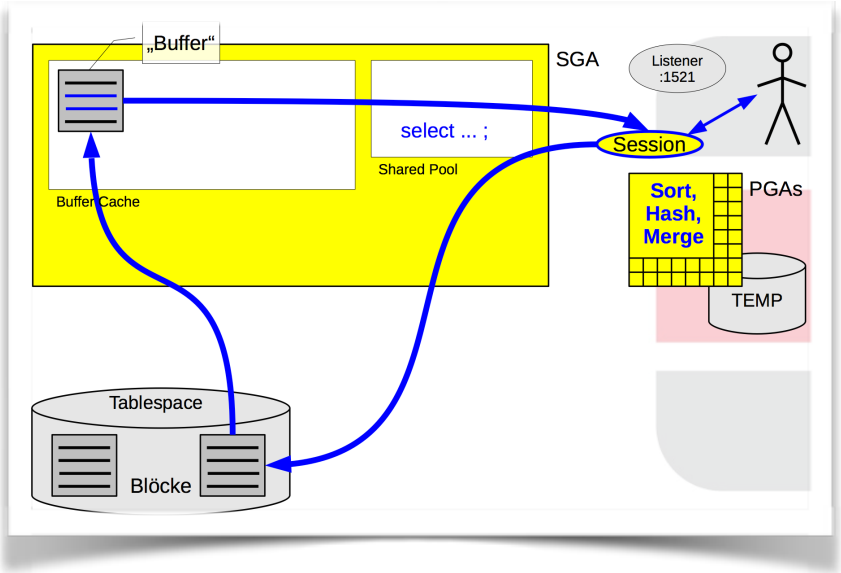


Diese Graphik aus der Oracle-Dokumentation 12cR1 zeigt das Zusammenwirken von Instanz und Datenbank.

Arbeiten mit dem Massenspeicher

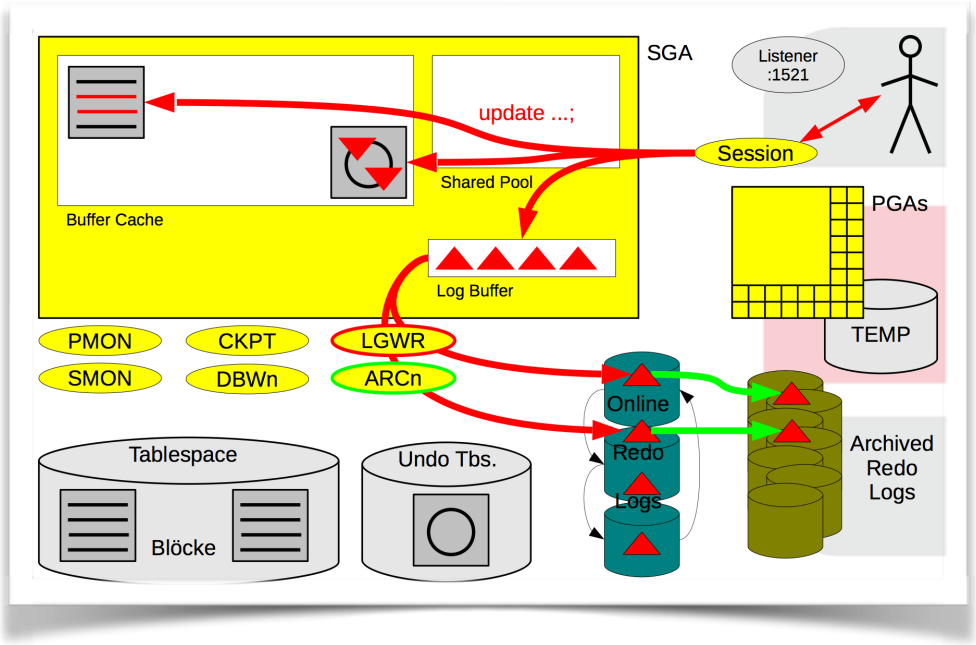
Zur Wiederholung sei kurz und vereinfacht die Handhabung von Daten im Oracle RDBMS dargestellt.

Lesen



Reguläre Lesevorgänge erfolgen durch den jeweiligen Serverprozess (“Session”) und bringen relevante Blöcke im Ganzen in den Buffer Cache. Ab hier spricht man nicht mehr vom Block, sondern vom Buffer. Von dort aus grenzt der Server auf Teilmengen/Rows ein.

Schreiben



- Verändernde Transaktionen schreiben zunächst auf den Buffer im Cache, erzeugen dort ebenso Undo-Informationen in einem weiteren Buffer.

- Für beide Vorgänge werden Redo-Daten im Redo Log Buffer eingetragen.
- Alle Buffer des Cache werden nach und nach durch den Komplex aus den Checkpoint- (CKPT) und Database Writer (DBWn) Prozesse in die Tablespace Datafiles synchronisiert.
- Der Log Writer Prozess (LGWR) schreibt die Inhalte des Redo Log Buffer in die Online Redo Logs. Sind alle Redo Logs voll und alle referenzierten Buffer synchronisiert, so werden die ältesten Einträge im Redo Log wieder überschrieben.
- Die aktuelle System Change Number wird im Control File aktualisiert, ebenso ggf. neue Datafile-Größen.
- Optional, aber sehr üblich für “Archivelog-Mode”: Die Archiver-Prozesse (ARCn) sorgen für die Duplizierung aller vollständig gefüllten Online Redo Logs in die sogenannten Archived Redo Logs (kurz: Archivelogs), die für verschiedene Recovery Szenarien benötigt werden, z.B. Vollständiges Recovery.
- Optional für “Flashback Database”, nur in Enterprise Edition: Der Recovery Writer Prozess (RVWR) kopiert angefallene Undo-Informationen in die Flashback Logs. Diese finden Verwendung um z.B. die ganze Datenbank in der Zeit zurückrollen zu können, ohne ein Backup einzuspielen.

3. Automatic Storage Management (ASM)

Das “Automatic Storage Management”, kurz “ASM” wurde mit der Datenbankversion 10g eingeführt. Der zugrunde liegende Gedanke ist, den Overhead eines vom Betriebssystem-Kernel verwalteten Dateisystems zu vermeiden. Im klassischen Fall wurde dieses Ziel mittels Raw Devices erreicht. Damit verknüpfen sich Datenbank- und Storageadministration sehr eng, was jedoch in den meisten Umgebungen ein deutlichen Nachteil darstellt. ASM geht einen Zwischenweg, um sich die operativen Vorteile eines Dateisystems und die hohe Leistung von Raw Devices zu eigen zu machen.

ASM stellt dem RDBMS sogenannte “Diskgroups” zur Verfügung, die ohne Nutzung von Dateisystem-Calls des Betriebssystems auskommen. Die Oracle Database besitzt dazu direkt im Oracle-Kernel hoch integrierte Verwaltungstreiber. Der DBA legt seine Datenbank-Dateien (Datafiles, Control Files, Redo Logs …) in den Diskgroups an, wie das auch in einem Dateisystem geschehen würde. Muß er (oder auch automatisiert das RDBMS) diese Dateien verwalten, kann er das nicht mit Betriebssystemmitteln tun, sondern wird dafür entweder die SQL-Interfaces des RDBMS oder der ASM Pseudo-Instanz nutzen. Auch eine Shell-artige Oberfläche “asmcmd” steht zur Verfügung.

ASM lässt sich am ehesten mit einem Logical Volume Manager (LVM) vergleichen, wie man ihn aus verschiedenen Betriebssystemen kennt. Es bietet diverse Mirroring- und Stripingtechnologien an, und bietet für Datenbanken und/oder das Cluster-Dateisystem ACFS eine zuverlässige Aktiv/Aktiv Replikation zwischen max. drei Storagezielen, ausgehend von beliebig vielen Servern im Cluster. Diese Funktionsweise lässt sich am leichtesten mit dem Bild eines RAID1 vergleichen, auf dem beliebig viele RAID-Controller in unterschiedlichen Servern gleichzeitig schreibenden Zugriff haben.

Beim klassischen User- oder System-IO spielt die Verwaltungsmethodik keine Rolle. Die ASM-Instanz und die genannten Tools werden durch die integrierte Funktionalität für reguläre Schreib/Lese-Operationen nicht benötigt. Damit entfallen unnötige Interprozesskommunikation, Latenz und Störquellen. Trotzdem ist im Bedarfsfall gesichert, dass der Massenspeicher (mehr oder minder) intuitiv verwaltet werden kann.

4. Tablespaces

IO Lesend

Für Lesezugriffe auf Tablespace-Daten kommen grundsätzlich drei Wege zum Einsatz:

- DB File Sequential Read: Einlesen eines einzelnen Blocks in den Buffer Cache. Diese Methode ist typisch für Index-Tabellen-Zugriffe: Ein Block aus dem Index, ein Block aus der Tabelle.
- DB File Scattered Read: Einlesen mehrerer (evtl. zusammenhängender) Blöcke in den Buffer Cache, wo sie aber nicht mehr genauso zusammenhängend abgelegt werden können. Typischerweise erfolgen damit Vollzugriffe auf Objekte (TABLE ACCESS FULL, INDEX (FAST) FULL SCAN).
- Direct Path Read: Einlesen mehrerer Blöcke unter Umgehung der SGA direkt in die PGA des ausführenden Serverprozesses. So wird der Verwaltungsaufwand für den Buffer Cache vermieden, allerdings steht das Ergebnis auch nur einer Session zur Verfügung. Beispiele, wann der Optimizer für einen Direct Path Read entscheidet, sind die Lesevorgänge aus TEMP, parallele Abfragen oder das Lesen größerer Datenmengen aus LOBs.

IO Schreibend

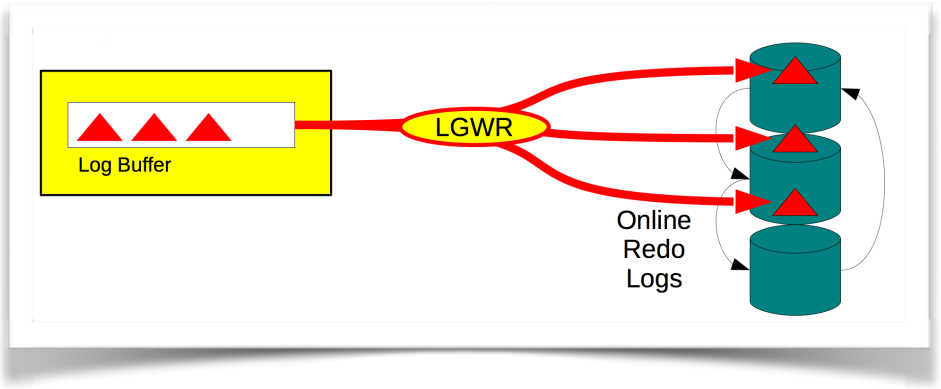
Schreibvorgänge auf den Tablespace erfolgen immer durch den Database Writer Prozess. Dieser arbeitet als “idle writer”, d.h. dieser Prozess schreibt dann, wenn Ressourcen verfügbar sind, nur selten dann wenn eine unmittelbare Notwendigkeit auftritt. Auch hier gibt es mehrere Möglichkeiten:

- DB File Parallel Write ist der Normalfall. Der Database Writer schreibt möglichst viele Blöcke gleichzeitig in die Data Files, und reduziert nach Möglichkeit die Anzahl der Repositionierungen auf dem Massenspeicher. Damit vermeidet er eine dedizierte Schwäche von magnetischen Festplatten.
- DB File Single Write: Bei den meisten Schreiboperationen auf ein Datafile müssen auch die Header des Tablespaces angepasst werden. Dabei wird immer nur ein Block modifiziert.
- Direct Path Write: Das Gegenstück zum Direct Path Read schreibt große Datenmengen asynchron von der PGA ins Datafile. Dazu ist notwendig, daß der insgesamt nötige Platz zusammenhängend im Tablespace frei ist, daher erfolgt hier das Einfügen immer am Ende aller schon vorhandener Daten (= oberhalb der High Water Mark).

5. Redo-/Archive Log

Online Redo Logs

Online Redo Logs dienen der Absicherung von im Cache modifizierten Buffers gegen Verlust. Die geschriebene Datenmenge entspricht den tatsächlichen Änderungen (Delta Vorher vs. Nachher). Damit reduziert sich das Volumen, das auf die Storageschicht geschrieben werden muss, ganz deutlich.



IO Lesend

Lesezugriffe auf Online Redo Logs sind die klare Ausnahme und werden nur beim Instance Recovery nach einem Absturz benötigt. Das dazu verwendete Verfahren ist “Log File Sequential Read”, und bedeutet, serialisiert alle Datensätze aus einem Online Redo Log einzulesen. Dieser Vorgang ist in der Regel zeitintensiv, und macht sich als Verzögerung beim Start der abgestürzten Instanz bemerkbar. Aber auch der zu Recherche- oder Debuggingzwecken einsetzbare Log Miner wendet diesen sequentiellen Zugriff auf Online Redo Logs an.

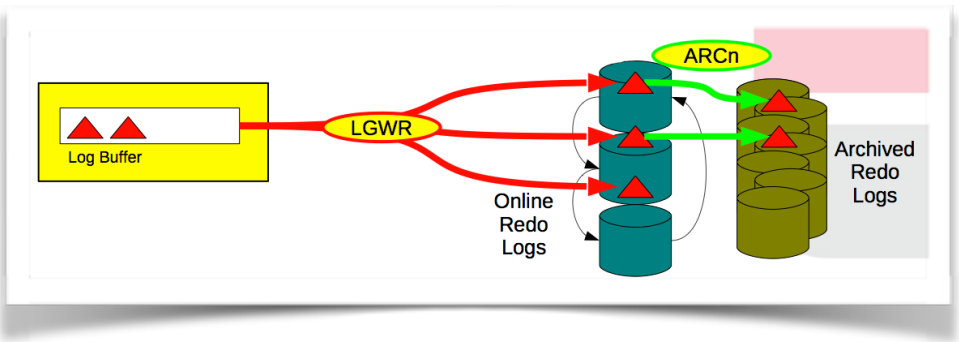
IO Schreibend

Schreibendes I/O auf Online Redo Logs ist die Domäne des Log Writers (LGWR).

- Log File Parallel Write synchronisiert die oben beschriebenen Deltas aus dem Log Buffer auf die verschiedenen Members der Online Redo Logs. Durch das vorangehende Puffern ist es möglich, größere IO-Portionen zu schreiben und die Anzahl der IO’s zu verringern.
- Log File Single Write kommt dann zur Anwendung, wenn die Header des Online Redo Logs geändert werden müssen. Das geschieht meist bei Konfigurationsänderungen, und ist im Normalbetrieb so gut wie nie der Fall.

Archived Redo Logs

Archived Redo Logs, oder kurz im Sprachgebrauch “Archivelogs”, sind eine 1:1 Kopie eines Online Redo Logs. Sie werden von den Archiverprozessen (ARCn) erstellt.



Archivelogs entstehen durch asynchrones IO - sowohl beim Lesen des gerade archivierten Online Redo Logs, als auch beim Schreiben des Ziels. Die Charakteristik des Archivers ist - ganz ähnlich dem DBWn - die eines “Idle Writers”: Die Archivierung beginnt frühestens mit dem Abschluss des Online Redo Logs, und endet spätestens bevor dieses überschrieben werden muss. Sie wird nach dem Grundsatz “so früh wie möglich, aber Lastspitzen sind zu vermeiden” ausgeführt.

Archived Redo Logs werden beim vollständigen bzw. Point-in-Time-Recovery wieder benötigt. (Sie erlauben, vom Backupzeitpunkt in der Vergangenheit den Konsistenz-Zeitpunkt in Richtung Gegenwart zu rollen (“Roll Forward). Dazu werden sie vom Serverprozess, der das Recovery betreibt, sequenziell eingelesen - ganz ähnlich einem Online Redo Log beim Crash Recovery.

6. Diverse

Neben den beschriebenen “großen” Disk-IO-Produzenten gibt es eine ganze Reihe “kleinerer” Schreib- und Lesevorgänge, die mindestens in Summe ins Gewicht fallen.

Control Files

Control Files sind ein Schlüsselelement des Oracle RDBMS. Sie beinhalten die Referenz für die aktuelle System Change Number (SCN) und u.a. die Liste aller zur DB gehörenden Dateien. Sie liegen deshalb gemäß Best Practice immer in drei identischen Kopien auf verschiedenen Dateisystemen und/oder ASM-Diskgruppen vor. Die Instanz beschreibt diese Control Files daher immer, wenn Transaktionen erfolgen (oft), Dateien hinzukommen (bei jedem Archivieren eines Redo Logs), sich Dateigrößen ändern (nicht so oft) oder die Konfiguration geändert wird (hoffentlich selten).

Backup / Restore

Beim Sichern oder Zurücksichern von Tablespaces/Datafiles und anderen Dateien werden teils erhebliche Datenmengen bewegt, z.B. beim Vollbackup. Diese Schreib- und Lesezugriffe (DB auf Backup oder Backup auf DB) erfolgen von Serverprozessen, die der DBA selbst oder z.B. der Recovery Manager (RMAN) eröffnet und verwaltet. Hierbei handelt es sich großteils um asynchrones IO, das mit möglichst großen Datenblöcken betrieben und auf optimalen Durchsatz hin optimiert wird.

In aller Regel werden größere Backups auch komprimiert (Standard im RMAN ist der bzip2 Algorithmus) und man tauscht CPU-Zeit gegen Bandbreite und Storagevolumen.

Export / Import

Administratoren und Operatoren haben nicht selten die Aufgabe, Inhalte von System zu System zu übertragen. Bei modernen Oracle-Datenbanken geschieht dies oft über die eingebauten “Datapump” genannten Export- und Importtools (expdp/impdp). Diese eröffnen über DB-interne Jobs Serverprozesse, in welchen externe Dateien in großen Blöcken eingelesen und über “klassische” SQL-Verfahren in die Datenbank gespeichert werden. Beim Umgang mit der Datenbank unterscheidet sie nichts von User-Sessions, d.h. sie benutzen wahlweise reguläre INSERT-verfahren, oder z.B. Direct Path Write/Read wie oben beschrieben. Zum Schreiben oder Lesen von Dumps kommen direkt Dateisystem-IO-Calls des Betriebssystems zum Einsatz, bzw. die entsprechenden im Oracle-Code enthaltenen ASM-Routinen. In beiden Fällen geht es um asynchrones IO von großen Datenblöcken.

Systemlogs und Tracefiles

Die Instanz und die mit ihr verbundenen Strukturen (wie TNS-Listener, ASM und die Oracle Clusterware) schreiben eine große Anzahl von Log- und Tracefiles. Zum Teil auch mit erheblichem Volumen, beispielsweise kann ein unpassend konfiguriertes Listener-Log schnell mehrere Gigabytes umfassen, ohne dass ein einziger Fehler aufgetreten wäre.

Das Schreiben von Logs und Traces geschieht in mit synchronen, kleinen IO-Calls des Betriebssystems, und in vordefinierte Verzeichnisse. Zugriffe auf Logfiles werden von der Datenbankinstanz in aller Regel vom zeitkritischen Ablauf entkoppelt, verursachen jedoch ein gewisses “Grundrauschen” auf den Massenspeichern.

7. Storage-Sizing

IOPS

In “Ein/Ausgabeoperationen pro Sekunde” = IOPS wird ein Aspekt der Leistung eines Massenspeichers gemessen. Je mehr IO’s dieser pro Zeiteinheit verarbeitet, desto mehr Transaktionen und kleine Leseoperationen, wie sie beide für Online Transaction processing (OLTP) typisch sind, kann das darauf laufende Datenbanksystem leisten.

OLTP Workload zu schätzen, gelingt umso besser, je mehr Informationen über die die Datenbank verwendenden Applikation(en) verfügbar sind. Ein guter Ansatzpunkt ist, die Anzahl der Transaktionen pro Sekunde in der Spitze zu bestimmen, und damit die minimale Schreib-IO-Leistung des Systems zu finden.

5-Finger-Faustregel

Dazu verwende ich folgende 5-Finger-Faustregel für die IO-Last pro Transaktion:

- 1x Buffer
- 1x Undo
- 1x Redo
- 1x Archiv
- 1x Control File

Dazu kommen je nach Anwendungsfall

- Mehr als ein Buffer betroffen (sehr häufig)
- 1x Flashback Log (nur EE falls aktiviert)

Sicher ist das nur im Durchschnitt betrachtet richtig (z.B. macht der ARCn nicht für jede Transaktion einen einzelnen IO), aber sicher im richtigen Maß pessimistisch. Auch das Abschätzen wie viele Buffer zu einer typischen Transaktion gehören, erfordert einiges an Erfahrung und Kenntnis (oder Zeit, diese zu erwerben) des Anwendungsfalles.

Vergleichstabelle

Für einige gängige Massenspeicher hier die typischen IOPS als kalkulatorische Anhaltswerte:

Medium	**IOPS**
Magn. Festplatte FC oder SAS, 15k rpm	140
Enterprise SSD	7.000
SAN Storagesystem	20.000 - 200.000
Storagecluster	1.000.000 +

Fertig gebaute Storage-Subsysteme oder ihre Building Blocks lassen sich vorzüglich mit dem Oracle IO Benchmark ORION testen, das kostenlos in jeder Oracle-Clusterware-Installation enthalten ist.

Durchsatz

Für Datenbanksysteme die ein Online Analytical Processing (OLAP) Lastprofil erwarten lassen, liegt der Schwerpunkt eher auf dem möglichen Maximaldurchsatz beim Schreiben und/oder Lesen. Wie viele MB oder GB pro Sekunde bewegt werden müssen, hängt natürlich bei weitem nicht nur von der zu extrahierenden Menge an Daten ab, dazu kommen alle eventuellen weiteren “Gewinnungskosten” aus Joins, Nutzung des temporären Tablespace für Sortier-, Merge oder Hashoperationen, dem Umfang der Transaktionsabsicherung im UNDO-Verfahren und einigem mehr ab. Allerdings lässt sich unter der Annahme dass wir analytisch und mit Massenzugriffen arbeiten werden, mit der folgenden, vereinfachten Faustformel schon einiges an potentiellen Engpässen im Storagelayer verdeutlichen.

b = E / R

wobei:
E = Summe der Größen aller am Join beteiligter Objekte
R = erwartete Antwortzeit
b = Bandbreite

Volumen / Größe

Das Gesamtvolumen einer Datenbank (nicht der Softwareinstalltion) errechnet sich aus den Nutzdaten plus Overhead. Wie viel Speicherplatz die Nutzdaten (Sn) im gewählten Schema verbrauchen, ist als allgemeine Regel schwer zu formulieren. Im Durchschnitt der letzten Jahre scheint sich ein Aufschlag zwischen 5% und 10% auf das rohe Plain-Text-Volumen (Sr) der Daten zu manifestieren, der für die Tablespaceorganisation und die Zerlegung in typisierte Felder “verloren” geht.

Sn = Sr * 1,1

Dazu kommt dann, wie erwähnt, noch der Overhead des RDBMS für das Data Dictionary, Temporär- und Undo-Tablespace. Hier sehen wir täglich Zahlen von deutlich über 10%, realistisch sind eher 20%. Die Gesamtgröße (Sg) errechnet sich dann mit erneutem Aufschlag nach:

Sg = Sn * 1,2

Damit hätte man bei einer Rohdatengröße (Sr) von 1TB zu planen:

1TB * 1,1 *1,2 = 1,32 TB

8. Abschluß

Das Thema wie auch dieses Dokument kennen kein Ende, nur die Aufforderung an alle Leser, sich weiterzubilden, und die “losen Enden” der Geschichte aufzugreifen, als Ansatzpunkt für das nächste Projekt, oder die eigene Fortbildung aus Interesse.

Wie so viele, steht auch dieses Paper auf den Schultern von Riesen. Insbesondere danke ich Frits Hoogland für seine Pionierarbeit, ausführlich über das Tracing von Oracle-Hintergrundprozessen zu publizieren, und die Ergebnisse und Erfahrungen bereitwillig zu erläutern. Eines der besten Papers aus seiner Serie sei hier erwähnt:
http://fritshoogland.files.wordpress.com/2013/02/about-multiblock-reads-v3-as43.pdf

Für jeden, der mehr über das Oracle RDBMS erfahren möchte, sei dessen vorzügliche Onlinedokumentation empfohlen, für die 12c derzeit hier zu finden:
http://docs.oracle.com/database/121/index.htm

How to get in touch

Martin Klier
Managing Partner / Database Technology
at Performing Databases

Wiesauer Strasse 27
95666 Mitterteich
GERMANY

Phone: +49 9633 631
Fax: +49 9633 4199
Mobile: +49 170 PERFDB7 (+49 170 7373327)

mailto:martin.klier@performing-db.com

Twitter: @MartinKlierDBA

About us

We are "**performing databases**" - and we are keen on caring for you:
"**Your reliability. Our concern.**"

Specialized in services around database technology, supreme performance and the highest availability of your systems is our ultimate and most important goal.

Performing Databases GmbH
Your reliability. Our concern.

http://www.performing-databases.com