Das A und O des Datenbank-IO

Dr. Günter Unbescheid
Database Consult GmbH
Jachenau

Schlüsselworte

Datenbank, Performance, I/O-Operationen, I/O Kalibrierung, Benchmark Tools

Abstract

In den letzten Oracle Releases hat sich bekanntlich viel in Sachen Performance getan. Vor allem im Bereich der Instanz wurde der Grad der Automatisierung merklich erhöht und dadurch das manuelle Eingreifen der Administratoren in diesem Bereich minimiert. Trotz allem werden wir immer noch und immer wieder mit Performance-Engpässen konfrontiert, die nicht selten im Umfeld von IO-Operationen zu finden sind. Der Vortrag gibt dem gemäß einen Überblick über die Besonderheiten von und Anforderungen an IO-Operationen im Bereich von Oracle-Datenbanken und zeigt die Möglichkeiten auf, die uns zur Messung und Optimierung dieser Operationen im aktuellen Release zu Gebote stehen.

Dabei werden sowohl die "externen" Faktoren der Plattenorganisation berücksichtigt, als auch die "interne" Parametrierung sowie die Organisation der Daten- und Indizes in der Datenbank selbst. Darüber hinaus werden auch die innovativen Technologien im Umfeld der Exadata-Familie behandelt.

Das Thema I/O ist vielschichtig und umfangreich. Der vorliegende Beitrag kann daher nur das Bewusstsein für die Abhängigkeiten und vielfältigen Möglichkeiten der Optimierung von I/O Operationen schärfen und einen dem entsprechenden Überblick geben.

Der Ausgangspunkt

Grob betrachtet lassen sich IO-Operationen in Oracle Systemen wie folgt charakterisieren:

- Zugriffe (lesend und/oder schreibend) auf einzelne Blöcke des Daten-Bereichs – in der Regel werden diese über vorangehende Index-Zugriffe gesteuert.
- Zugriffe auf „sequentiell“ hintereinander liegende Blöcke des Datenbereichs (*full scans*)
- Schreibende Zugriffe auf den Bereich der Online Redo Logs.
- Nach einem Logswitch Lesen des Online und Schreiben des Offline Redo
- Das Verhältnis der Read zu den Write Zugriffen wie das der Einzelblock- zu den sequentiellen Zugriffen wird über die Charakteristik der jeweiligen Anwendung festgelegt.

Es gibt zwei typische Szenarien über die wir gezwungen werden, uns des Themas I/O-Optimierung nähern:

- Eine Systemanalyse, die auf Grund von konkreten Laufzeitproblemen durchgeführt wird, zeigt signifikante Events – beispielsweise `db file scattered read` oder `db file sequential read` – die auf maßgebliche IO-Probleme hinweisen.
- Eine vorhandene oder neu geplante (Speicher-)Infrastruktur soll hinsichtlich ihrer Leistungsfähigkeit prophylaktisch und pro aktiv beurteilt werden.

In beiden Fällen ist es hilfreich und wichtig, die folgenden Prämissen zur Grundlage der eigenen Arbeit zu machen:

- Datenbanken im Allgemeinen und Oracle-Systeme im besonderen verfügen über spezifische IO-Charakteristiken. Diese sind darüber hinaus auch stark abhängig von den Eigenarten der zugreifenden Applikationen – OLTP auf der einen und DSS/Data Warehouse Systeme auf der anderen Seite
- Organisatorisch und personell sind Storage- und Datenbank-Abteilungen in der Regel getrennt aufgestellt. SAN-Systeme werden daher von DBAs häufig als „Black Box" betrachtet. Zur Optimierung vorhandener oder zur optimalen Konfiguration geplanter Systeme ist es jedoch extrem wichtig, den „Blick über den Zaun" zu wagen und in Zusammenarbeit mit den Storage-Kollegen die Details der SAN-Konfiguration zu erarbeiten. Das von der SQL-Optimierung bekannte und geschätzte Motto „*know your data*" kann an dieser Stelle erweitert und aufgestockt werden: „*know your infrastructure*". Mit anderen Worten: Wir brauchen eine ganzheitliche Strategie.

Die oben genannte Systemanalyse wird in vielen Fällen auf Seiten der Datenbank Werkzeuge wie Statspack oder AWR nutzen. Die häufig diskutierten Nachteile der pauschalen und verallgemeinernden Sicht dieser Werkzeuge auf die betreffenden Instanz können wir häufig durch ein klug gewähltes Zeitfenster wettmachen. Auf diese Weise erhalten wir Daten, die für die zu untersuchende Anwendung und IO-Problematik durchaus charakteristisch sind.

Die folgenden Abschnitte geben beispielhaft und ausschnittweise die für IO-Analysen wichtigen Bereiche eines AWR-Reports wieder, der im Kontext eines Order Entry Benchmarks generiert wurde. Im Bereich „Load Profile" lässt sich zunächst das globale Verhältnis von Reads zu Writes ablesen (logical reads/block changes)

```
Load Profile            Per Second    Per Transaction   Per Exec   Per Call
~~~~~~~~~~~~       ---------------    --------------- ---------- ----------
Logical reads:              757.9              57.0
Block changes:              191.3              14.4
Physical reads:              60.4               4.5
Physical writes:             27.0               2.0
```

Die fünf Top Events zeigen in unserem Beispiel mit den ersten drei Events im Daten- wie im Redo Bereich einen deutlichen IO-Schwerpunkt. Ob in diesem Falls ein IO-Problem vorliegt müssen die weiteren Kennzahlen ergeben:

```
Top 5 Timed Foreground Events
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                                    Avg
                                                   wait   % DB
Event                             Waits     Time(s)   (ms)   time Wait Class
------------------------- ------------ ----------- ------ ------ ----------
db file sequential read         85,472       5,931     69   69.7 User I/O
log file sync                    9,580       2,043    213   24.0 Commit
db file parallel read            2,724         451    166    5.3 User I/O
DB CPU                                          58             .7
read by other session               14           2    119     .0 User I/O
```

Die recht hohen *wait IO* Zeiten im folgenden Abschnitt bestätigen die hohen IO-Raten, die hohen idle-Zeiten der CPU von über 96% deuten darauf hin, dass das System nicht CPU-bound ist und daher

eine mögliche Optimierung des IO-Subsystems von der CPU durch beschleunigte Generierung von IO-Requests quittiert werden könnte:

```
Host CPU (CPUs:    2 Cores:    2 Sockets:    1)
~~~~~~~~          Load Average
                Begin       End     %User   %System      %WIO     %Idle
              --------- --------- --------- --------- --------- ---------
                   6.20      4.19       1.1       1.1      63.7      96.8
```

Das Event-Histogram sc hlüsselt die oben zitierten IO-Events noch genauer auf:

```
                      Total
Event                 Waits  <1ms  <2ms  <4ms  <8ms <16ms <32ms  <=1s   >1s
--------------------- ----- ----- ----- ----- ----- ----- ----- ----- -----
db file scattered r.     19  63.2                                36.8
db file sequential r. 85,5K  42.0    .0    .0    .5   2.5   6.5  48.4    .0
```

In einem weiteren Abschnitt wird die IO-Last nach Filetype aufgelistet. Hier lässt sich ebenfalls das Verhältnis von Reads zu Writes sowie das Datenvolumen und die Servicezeiten in ms ablesen (small Read):

```
IOStat by Filetype summary            DB/Inst: TEST01/Test01  Snaps: 28-29
            Reads:   Reqs   Data    Writes:  Reqs   Data      Small   Large
Filetype N. Data    per sec per sec Data    per sec per sec    Read    Read
----------- ------- ------- ------- ------- ------- ------- ------- -------
Data File      736M    60.3 ,471371    329M    23.0 ,210708    72.2     N/A
Log File         0M     0.0       0M     43M     7.3 ,027539    22.5     N/A
Control File    21M     0.9 ,013449     21M     0.9 ,013449     0.3     N/A
Other            0M     0.0       0M      0M     0.0       0M     0.0     N/A
Temp File        0M     0.0       0M      0M     0.0       0M   180.0     N/A
TOTAL:         757M    61.2 ,484821    393M    31.2 ,251697    71.2     N/A
```

Im Abschnitt Tablespace IO Stats schließlich bestätigen sie die schwachen IO-Zeiten von 63.3 für Reads.

```
Tablespace IO Stats                  DB/Inst: TEST01/Test01  Snaps: 28-29

Tablespace
------------------------------
             Av       Av     Av                        Av     Buffer  Av Buf
     Reads Reads/s  Rd(ms) Blks/Rd       Writes Writes/s      Waits  Wt(ms)
---------- ------- ------- ------- ------------ -------- ---------- -------
SOE
    93,495      60    63.3     1.0       35,312       23         16   123.8
```

An dieser Stelle könen wir nun eindeutig von einem schwachen IO-Subsystem ausgehen. Zusätzlich wäre – wegen des Events log file syn – auch der Redo Apparat zu prüfen und zu optimieren.

Für die konkrete Optimierung des IOs im Datenbereich bieten sich – formal betrachtet – folgende Strategien an:

- Reduktion des IOs: Hier hilft nur ein Blick auf die IO-intensivsten SQL-Statements, denn IO in Datenbanken wird über SQL ausgelöst. Zur Reduktion stehen die Überprüfung von Sekudärstrukturen wie Indizes zur Debatte, aber auch die physikalische Organisation der Segmente (Table Cluster, Partitionierung etc) oder die Algorithmen des Optimizer, die in manchen Fällen über Hints, Profiles und dergleichen gestützt werden müssen. Dieser Blick in die „logischen“ Strukturen der Datenbank ist in der Regel aufwändig und zeitraubend.

Ein Sonderfall bei der „Reduktion“ von IO stellen die Strategien im Kontext von Exadata Systemen dar, bei der Teile der „IO-Intelligenz“ in das Storage System verlagert werden und auf diese Weise weniger Daten an das RDBMS transferiert werden müssen.

- Beschleunigung des IOs durch Analyse und Optimierung der IO-Infrastruktur. Probleme in diesem Bereich können durch eine ungenügende, nicht Anforderungs-gemäße Konfiguration oder schlichte Überlastung von Storage Arrays entstehen.

Auf jeden Fall sollten wir an dieser Stelle Kontakt zu unseren Storage-Kollegen aufnehmen und im Team die Details des IO-Subsystems prüfen.

Lastgenerierung: Die Werkzeuge

Es versteht sich, dass verlässliche Metriken für die Leistungsfähigkeit von IO-Subsystemen über praktische Lasttests ermittelt werden müssen. Prinzipiell unterscheiden wir hierbei transaktionale Tests, die mit Benutzer-spezifischen oder standardisierten Benchmark-Transaktionen arbeiten, und nicht transaktionale IO-Generatoren, die das IO-Subsystem mit sinnfreien Lese- und Schreiboperationen „bombardieren“. Während erstere Tests konkrete Antwortzeit-Metriken für eine erwartete Nutzerlast (z.B. Anzahl der Sessions) bereitstellen, ergeben letztere den maximalen Durchsatz und die maximale Anzahl von IO-Operationen.

Um die Vergleichbarkeit wiederholter Tests sicherzustellen, muss darüber hinaus der Testablauf mit identischen Eingabeparametern beliebig oft ausgeführt werden können. Für transaktionale Tests ist zusätzlich auf eine identische oder vergleichbare Datenbasis zu achten.

Dem entsprechend stehen im transaktionalen Umfeld prinzipiell folgende Strategien zur Verfügung:

- Nutzung von originalen Transaktionsdaten bestehender Applikationen:
  Hierzu müssen originale Transaktionsdaten aufgezeichnet und später abgespielt werden, wie es beispielsweise die „Oracle Real Application Testing“ Database Option ermöglicht. Die Rücksetzung des Datenstandes kann dann über Cloning-Techniken oder Flashback Database sichergestellt werden.
- Dort wo eine originale Transaktionslast nicht gewünscht ist oder nicht zur Verfügung steht, lassen sich mit Benchmark-Werkzeugen standardisierte Transaktionen parametrieren und ausführen.

Die wichtigsten Werkzeuge in diesem Bereich werden in den folgenden Abschnitten kurz vorgestellt:

- **Orion** – steht für ORacle IO Numbers und gehört zur Kategorie der IO-Generatoren. Es steht für eine Reihe von Plattformen zur Verfügung, muss jedoch explizit über Oracle’s Technet heruntergeladen werden, wird aber offiziell nicht von Oracle unterstützt. Da es eine synthetische IO-Last generiert, die auf dem von der Datenbank genutzten Softwarestack basiert, benötigt das Werkzeug keine Oracle-Installation und keine konkreten Datenbanken. In einer Konfigurationsdatei werden vielmehr die zu messenden LUNS/Platten angegeben. Die genaue Charakteristik der Lese- und Schreiblast wird über unterschiedlicher Profile gesteuert*: Small random IO,* **Large Sequential IO**, *Large Random IO* und *Mixed Workload*. Die Ergebnisse der Testläufe werden jeweils in unterschiedlichen CSV- und Log-Dateien erfasst. Kritische Beiträge (Kevin Closson) bemängeln die fehlende Berücksichtigung von buffered IO.
- **Calibrate_io** – Oracle bietet im Rahmen des DB-internen Resource Manager seit der Version 11.1 eine eigene Routine zur synthetischen Generierung einer Leselast. Das Werkzeug zur „Kalibrierung“ des I/O-Systems steht als Erweiterung des Database Resource Managers bereit,

setzt aber voraus, dass das synchrone IO aktiviert wurde. Die API DBMS_RESOURCE_MANAGER.CALIBRATE_IO() erzeugt eine gemischte read-only Last bestehend aus random I/Os in der Größe der betreffenden db_block_size sowie sequentiellen I/Os mit 1MByte Blockgröße. Diese Methode benötigt keine generierten Testdaten, da nur Datenbank-Blöcke gelesen werden. Die Ergebnisse der Kalibrierung werden in Systemtabellen der Datenbank erfasst und u.a. für die automatische Bestimmung des Parallelisierungsgrades eingestzt.

- **Swingbench** – im Gegensatz zu den Vorgängern gehört das Werkzeug Swingbench von Dominic Giles zur Klasse der Benchmark-Generatoren. Neben verschiedenen, vorkonfigurierten Benchmarks aus dem OLTP- und DSS-Benreich, wie beispielsweise „Order Entry“ und „Sales History“, können auch eigene Tests über eine PL/SQL-Schnittstelle eingebunden werden. Das Werkzeug läßt sich sowohl grafische als auch per command line steuern und bietet neben den eigentlichen Benchmarks auch Routinen zur Generierung der Testdaten an. Konfiguratorische Details und Ergebnisse werden in XML-Dateien vorgehalten. Deiverse Parameter erlauben eine individuelle Steuerung hinsichtlich der Testdauer, der Anzahl der Sessions, der „Denkzeiten“ u.v.m.
- **SLOB** – von Kevin Closson soll nach eigenen Angaben die Lücke zwischen transaktionalen Benchmarks und reinen IO-Generatoren schliessen. Das Akronym steht für „Silly Little Oracle Benchmark“. Auch SLOB nutzt PL/SQL und baut daher auf eine existierende Datenbank. Es kann u.a. *physical random single-block reads* , *random single block writes* und die Skalierung von logischem IO testen. Das Werkzeug kann über das OAK Table Netzwerk heruntergeladen werden.
- **Benchmark Factory** – ist ein kommerzielles Benchmark Werkzeug von Quest/Dell, das neben den gängigsten Standard-Benchmarks auch individuelle Workloads aufzeichnen und wiedergeben kann (replay). Es wird als einfachere und kostengünstigere Variante zu Oracle’s Real Application Testing vermarktet.
- **Oracle’s Real Application Testing** – ist eine Option, die im Rahmen der Enterprise Edition der Datenbank zur Verfügung steht und wurde bereits kurz besprochen.
- **Hammerora** – ist ein Open Source Lastgenerator, der nicht nur standardisierte Benchmarks wie TPC-C und TPC-H ausführen kann, sondern ebenso Transaktionen aus spezifischen Trace-Dateien herauslesen und über ORATCL wieder geben kann. Auf diese Weise können Benutzer-eigene Transaktionen komfortabel für Lasttests herangezogen werden.
- **FIO** – steht für „Flexible IO Tester“ und erlaubt es – wie der Name nahelegt – sehr differenzierte IO-Tests über eigene oder vorgefertigte Konfigurationsdateien aufzusetzen. Es wurde in C geschrieben und steht für eine ganze Reihe von Plattformen zur Verfügung. Es unterstützt insgesamt 13 unterschiedliche IO-Engines. Das Werkzeug wird über freecode.com bereitgestellt. Es versteht sich, dass FIO keine transaktionalen IO-Tests durchführen kann.
- **bonnie** – ist ein einfach zu nutzendes Werkzeug zur IO-Messung von Filesystemen und Platten.

Die Storage-Welt

Für optimale IO-Operationen ist ein optimales Zusammenspiel der Storage-Komponenten mit den betreffenden Servern sowie den Anforderungen der Anwendungen notwendig. An dieser Stelle sollen daher die wichtigsten Grundbausteine einer Storage-Infrastruktur am Beispiel einer SAN-Infrastruktur aufgelistet werden:

- Die Plattencharakteristik – jeder Disk wird vor allem bestimmt über die Speicherkapazität sowie die Rotationsgeschwindigkeit.
- Disk Interface – ist die Schnittstelle und das Protokoll das zur Kommunikation eines Laufwerks dient, beispielsweise SATA, SCSI, Fible Channel u.a.
- Anzahl der Platten (Spindeln) – bestimmen nicht nur die Gesamtkapazität eines Arrays, sondern steigern auch den Gesamtdurchsatz durch Ausnutzung von Striping-Technologien.
- RAID-Level – bestimmen die Speichermuster von Platten, durch Aufteilung (striping), Spiegelung (mirror) und zusätzliche Parity-Informationen, die bein Ausfall von Platten zu Rekonstruktion von Daten verwendet werden können. RAID Level 5 und RAID Level 6 schreiben ein bzw. zwei getrennte Parities, verursachen dadurch aber zusätzliche Lasten im Falle von Datenänderungen. RAID Level 10 (mirror + stripe) dagegen verzichtet auf Parity-Informationen.
- RAID-Groups – organisieren in SAN-Arrays ein bestimmte Anzahl von Platten unter einem RAID-Level. Beispielsweise kann eine RAID-Gruppe mit Level 6 aus 10 Platten/Spindeln bestehen, wobei 2 Parity Platten für Parity-Informationen reserviert sind, mithin 8 Spindeln für die Nettodaten verbleiben.
- Storage-Pools – ein oder mehrere RAID-Gruppen werden im SAN-Array zu einem Storage-Pool zusammengefaßt.
- LUN – Logical Units werden aus einem SAN-Storage-Pool für bestimmte Hosts rekrutiert. Auf Seiten des Hosts werden die LUNs partitioniert und für Filesysteme genutzt.
- HBA – Host Bus Adapter sorgen auf Seiten des Hosts für den Datentransfer mit dem Storage System. Hier sind vor allem die konfigurierbaren Übertragungsgeschwindigkeiten wichtig (1-8GB). Sind mehrere SAN-Arrays spiegelbildlich konfiguriert, kann der HBA darüber hinaus im „round robin“ Verfahren die Zugriffe auf unterschiedliche SAN-Hardware verteilen.

Die vorstehend geschilderte Storage-Welt wird durch die innovativen Konzepte der EXADATA-Maschinen um einige interessante Komponenten erweitert. Die Grundidee ist hierbei einige Prozessschritte vom Datenbank-Host in das Storage-System zu verlagern und auf dieser Weise den Umfang des Datentransfers zwischen Storage und DB-Server (drastisch) zu reduzieren:

- CPU-intensive Operation wie Joins, Aggregate und Datenkonvertierungen werden nach wie vor auf dem DB-Host durchgeführt.
- Speicher-intensive Operationen werden dagegen nicht nur auf der Storage-Seite durchgeführt, sondern dort auch bereits vorgefiltert. Auf diese Weise werden Tabellenzeilen und Spalten die nicht relevant für die Abfrage sind bereits aussortiert bevor sie den Weg über das Netzwerk zu dem DB-Host antreten.
  Aus dem Gesagten wird deutlich, das diese „smart scan“ genannte Technik vor allem in Datawarehouse-Umfeld zum tragen kommt, wo häufige und umfangreiche full scan Operationen durchführt werden.

Messung: Maßgebliche Wait Events

Die für den IO-Bereich maßgeblichen Wait Events werden auf den Vortragsfolien im Einzelnen vorgestellt.

Software: Datenbank-Welt

Die Oracle Datenbank bietet verschiedenste Technologien, die direkten Einfluss auf den IO-Bereich haben. Hierzu zählen beispielsweise Bitmap-Indizes, die auf Grund ihrer kompakten Speicherung Scan-Operationen optimieren, Result-Caches die häufig benötigte Datensätze separat vorhalten oder verschiedene Partionierungstechniken.

Auch die in diesem Abschnitt relevanten Techniken werden auf den Folien detailliert erläutert.

Kontaktadresse:

Dr. Günter Unbescheid

Database Consult GmbH

Laich 9

D-83676 Jachenau

Telefon:	+49 (0) 8043 1010
Fax:	+49 (0) 8043 1011
E-Mail	g.unbescheid@database-consult.de
Internet:	www.database-consult.de