Flash-Disks für das schnelle Commit in Oracle

Erik Neitzel, Michael Höding
FH Brandenburg, Magdeburger Straße 50, 14770 Brandenburg/Havel
neitzel@fh-brandenburg.de, hoeding@fh-brandenburg.de

Zusammenfassung

In diesem Beitrag werden Einsatzmöglichkeiten von Flash-Disks (auch Solid State Disks bzw. SSDs genannt) in DBMS-basierten Systemen betrachtet. Anhand des DBMS Oracle diskutieren wir eine Idee durch Flash-Disk-Einsatz das Commit in DBMS zu beschleunigen. Für detaillierte Messungen wird derzeit ein Testframe entwickelt. Erste Versuche und Messungen demonstrieren das Verhalten von SSDs zur Speicherung von Transaktionslogs.

1 Motivation

Die Verbesserung der Performance ist ein Hauptziel der Forschung im Bereich Datenbanken. Motivation ist letztendlich ein für jeden Benutzer akzeptables Antwortzeitverhalten im Produktivbetrieb. Dabei werden Ergebnisse aus nahezu allen Bereichen der Informatik genutzt oder Forschung in diesen Bereichen motiviert. Die technische Informatik liefert schnelle und optimierte Hardware, die theoretische Informatik Algorithmen. Sogar der KI-Bereich bietet interessante Prognoseverfahren zur Optimierung.

Das zentrale Problem ist dabei die Speicherlücke zwischen Hauptspeicherzugriff und Sekundarspeicherzugriff [6]. Physischer Zugriff soll einerseits so weit wie möglich vermieden werden. Andererseits muss der physische Zugriff beschleunigt werden.

Eine aktuelle technische Entwicklung sind Solid State Disks oder Flask-Disks, die als nichtmechanischer Speicher der traditionellen Festplatte Konkurrenz machen. Neben Aspekten wie mechanischer Robustheit und Energieeffizienz versprechen SSDs technologiebedingt einen schnelleren wahlfreien Lesezugriff. Ihr unmittelbarer Einsatz für Datenbanksysteme bringt allerdings nicht in jedem Fall eine Performancesteigerung wie in [4] und [5] gezeigt wurde. Ziel ist es nun, die Einsatzmöglichkeiten von SSDs in Datenbanksystemen näher zu untersuchen, um so Möglichkeiten und Grenzen zu erkennen.

2 Technische Aspekte, Betrachtungsebenen

SSDs bestehen aus Flash- und Controllerchips, angeordnet auf einer Leiterplatte. Die Anzahl der Chips hat hierbei analog zur RAID-Technologie großen Einfluss auf die Geschwindigkeit. Bei reinen Lesevorgängen und Multitasking-Operationen (gleichzeitigem Lesen und Schreiben) sind Flash-Disks den herkömmlichen Festplatten überlegen [3]. Beim alleinigen Schreiben waren bzw. sind SSD hingegen technologiebedingt etwas langsamer. Eine SSD ist in Blöcke unterteilt, wobei unter Umständen der gesamte Block neu geschrieben muss. Wenn ein Bit seinen Wert von 1 auf 0 ändert, müssen alle Bits des Blocks durch eine negative Spannung zurückgesetzt werden. Ein weiteres Problem ist die begrenzte Zahl der Schreibzyklen für Flash-Speicherzellen.

Wir möchten in diesem Beitrag Einsatzmöglichkeiten von SSDs in Datenbanksystemen betrachten. Wichtig sind dabei die folgenden Ebenen:

- **Anwendung**
  In [5] wurde der DBT2-Benchmark genutzt. Als typischer Anwendungsbenchmark emuliert DBT2 eine OLTP-Last und misst die Anzahl der durchgeführten Transaktionen in einem

Zeitraum. Nachteilig waren die problematische Portierbarkeit für verschiedene DBMS und die fehlenden Eingriffsmöglichkeiten für detaillierte Untersuchungen. Aus diesen Gründen wird derzeit ein Testframe entwickelt, der diese Probleme zu überwinden hilft.

- **Datenbanksystem**
  Auf der Ebene des DBMS kommen Transaktionslogs zum Einsatz, die durch schnelles sequenzielles Schreiben ein schnelles Commit ermöglichen. Eine Optimierungsmöglichkeit mit SSDs wird im nächsten Abschnitt diskutiert. Hinsichtlich des wahlfreien Schreibens von Datenbankblöcken müssen die genutzten Verfahren für den SSD-Einsatz hinsichtlich Fragmentierung und Clusterung untersucht werden.

- **Betriebssystem, Treiber, Hardware**
  Die Probleme der begrenzten Schreibzyklenzahl und der geringen Schreibgeschwindigkeit durch Flashen sind gegenwärtig die Entwicklungsschwerpunkte bei den Herstellern. So werden Schreibzugriffe auf Controller- oder Treiberebene mit dem Ziel verteilt, möglichst alle Speicherzellen der SSD gleich häufig zu beschreiben. Eine physische Fragmentierung sollte dabei kein Problem sein, da es keine Wartezeiten zur Positionierung von Lese-Schreibköpfen gibt. Allerdings ist dieser Aspekt beim schreibenden DBMS-Zugriff zu beachten. Diese Entwicklungen müssen berücksichtigt werden, da sie mit Optimierungsansätzen im DBMS wechselwirken können.

Die Entwicklung von SSDs befindet sich derzeit in einer sehr aktiven Phase. Inzwischen sind SSDs zu traditionellen Magnetplatten konkurrenzfähig, wie erste Experimente zeigen.

3 Nutzung in Oracle

Am Beispiel von Oracle möchten wir nun die gezielte Nutzung von SSDs für das Transaktionslog diskutieren. Abbildung 1 abstrahiert den schreibenden Zugriff bei Oracle. Beim Commit werden im Schritt 5b die Einträge des Redolog-Puffers in die Logdateien geschrieben. Ist diese Schreiboperation beendet, so bekommt der Anwender den Erfolg gemeldet. Die Geschwindigkeit der Schreiboperation bestimmt also entscheidend das Antwortzeitverhalten.

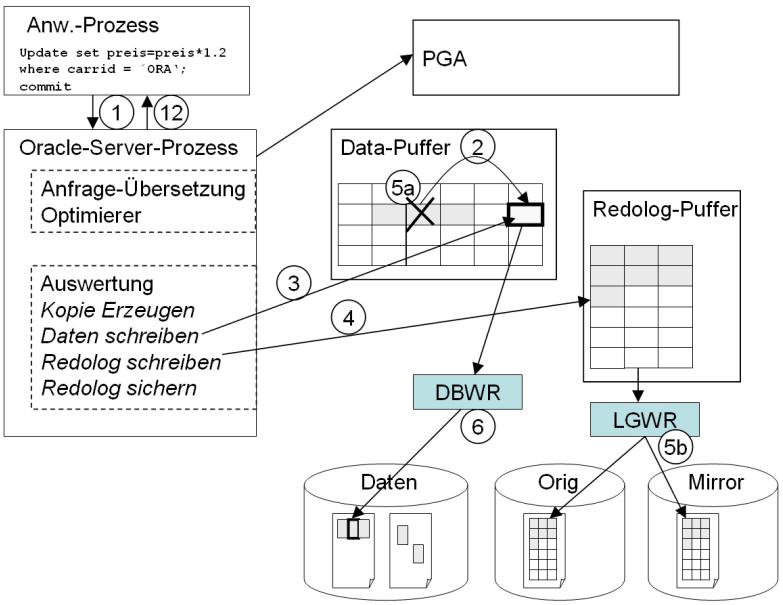

Abbildung 1: Transaktionlog am Beispiel von Oracle [1]

Dieser Mechanismus, der in ähnlicher Form in allen DBMS implementiert ist, nutzt die Tatsache, dass das sequentielle Schreiben in eine Datei (hier die Log-Datei) einen großen Geschwindigkeitsvorteil gegenüber dem verteilten Schreiben, z.B. in die Daten-Dateien bietet. In Oracle sind die Log-Dateien als Ring-Puffer organisiert. Ist eine Log-Datei voll oder wird ein expliziter Log-Switch ausgelöst, so kommt die nächste Log-Datei zum Einsatz (vgl. Abb. 2, Schritt 1). Inzwischen kann die geschlossene Log-Datei archiviert werden (Schritt 2). Ist die Log-Datei nun bei der nächsten Nutzung noch mit alten (ungültigen) Inhalten beschrieben, so stellt dies für den SSD-Einsatz ein Problem dar. Auf Hardware- oder Treiberebenen wird nicht erkannt, dass die Altdaten keine Bedeutung haben. Entsprechend ist mit jedem Schreiben eines Logeintrags ein teures Flashen und Zurückschreiben des Speicherblocks verbunden.

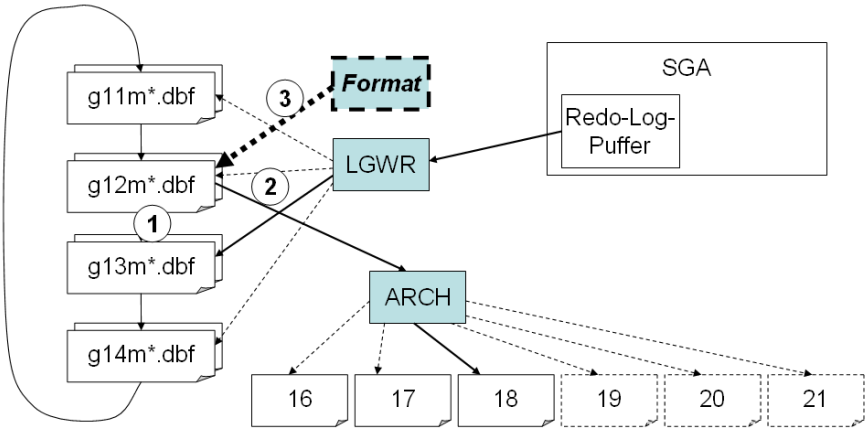

Abbildung 2: Log-Switch, Archivierung und Formatierung

Zur Lösung dieses Problems schlagen wir vor, die Log-Dateien nach der Archivierung zu formatieren, so dass ein sequenzielles Schreiben ohne Flashen möglich wird (vgl. Abb. 2, Schritt 3). Dies kann durch einfache Betriebssystemprozesse erfolgen. Problematisch bei der Untersuchung ist, dass SSD-Treibersoftware herkömmliche Dateisysteme emuliert und dadurch das Monitoring schwierig ist. So ist die Lokation von Speicherblöcken nicht zuverlässig nachvollziehbar. Über durchgeführte Experimente und Ergebnisse werden wir in Zukunft berichten.

4 Testframe

Zur Durchführung konkreter Messungen dient ein eigens in Java implementierter Benchmark-Testframe, welcher sich durch verschiedene Angaben parameterisieren lässt. Darunter fallen die Angabe konkurrierender Nutzer (die parallel DML-Befehle an die Datenbank senden), deren Startzeitpunkt (die Akteure starten um eine Zeiteinheit versetzt), die Anzahl der Tabellen und die Anzahl von `selects`, `inserts`, `updates` und `deletes` die ein Nutzer auf diesen Tabellen ausführt. Zusätzlich lässt sich über einen Parameter `repeat` festlegen, wie oft ein kompletter Durchlauf wiederholt werden soll.

Nachfolgend ein Ausschnitt des Parameter-Files:

```
1  #Benchmark Parameter File
   #Fri Mar 27 01:29:57 CET 2009
3  users=10
   sleep=100
   tablenumber=5
6  tableinsertnumber=10
   tableselectnumber=10
```

```
tableupdatenumber=10
tabledeletenumber=10
repeat=30
```

Es findet der Oracle Thin JDBC Treiber Verwendung [2]. In der Theorie bietet der Oracle OCI-Treiber zwar durch seine nativere Natur einen Performance-Vorsprung, jedoch ist man beim Benchmarking nur angewiesen auf den Vergleich von identischen Ausstattungen. Da nicht zu erwarten ist, dass sich der Datenbank-Treiber bei Zugriff auf eine SSD maßgeblich anders verhält als bei einer herkömmlichen HDD, steht hierbei die leichtere Entwicklung des Java-Testframes im Vordergrund.

Der Testframe generiert die angegebene Anzahl von Tabellen mit einem zentralen User, wobei drei Indexe auf drei Attributen (Datentypen `number`, `varchar`, `varchar2`) erzeugt werden. Danach wird die hinterlegte Anzahl von Test-Usern erzeugt. Diese arbeiten dann parallel auf den zentralen Tabellen statische SQL-Operationen ab: zuerst `inserts`, dann `selects`, gefolgt von `updates` und `deletes`.

Gemessen wird die Zeit für jeden SQL-Akteur, beginnend ab der ersten `insert`- und der letzten `delete`-Operation auf einer Tabelle, sowie die Zeit für jede der genannten SQL-Operationen selbst. Letztlich interessant ist die Gesamtlaufzeit eines Testlaufes samt all seiner User, wobei für ein verwertbares Ergebnis mehrere Testläufe vollzogen und Mittelwerte gebildet werden.

5 Testlauf-Beispiel

Es folgen konkret durchgeführte Testläufe, mit Logdateien auf einer HDD und einer SSD. Hier wurden Messdaten mit 30 Durchläufen erfasst. Um freier von dritten Einflüssen wie der Arbeit von Garbage-Kollektoren zu sein, wurden die restlichen Werte nicht zu hoch gewählt. Konkret wurde mit folgender Parameter-Konfiguration gearbeitet:

```
-------------------------------------------------------------------------
|  DB-Type          : oracle
|  AutoCommit       : true
|  Sleep Intervall  : 100
|  Test Users       : 10
|  Quota Amount     : 50
|  Table Number     : 5
|  Table Inserts    : 10
|  Table Selects    : 10
|  Table Updates    : 10
|  Table Deletes    : 10
|  Repeat           : 30
-------------------------------------------------------------------------
```

Hier das Ergebnis der HDD-Durchläufe:

```
################ Benchmark Overall Analysis ###############
 Results for all actors and all repeat cycles!
 Overall time (all cycles)         :  324.604 ms
 Average overall time (all cycles) :   10.820 ms
 Average actor time                :    1.081 ms
 Average actor insert time         :       49 ms
 Average actor select time         :       18 ms
 Average actor update time         :      425 ms
 Average actor delete time         :       18 ms
###########################################################
```

Hier das Ergebnis der SSD-Durchläufe:

```
1   ############### Benchmark Overall Analysis ###############
     Results for all actors and all repeat cycles!
3    Overall time (all cycles)         :  201.598 ms
     Average overall time (all cycles) :    6.719 ms
     Average actor time                :      671 ms
6    Average actor insert time         :       46 ms
     Average actor select time         :       16 ms
     Average actor update time         :       75 ms
9    Average actor delete time         :       16 ms
    ###########################################################
```

6 Fazit

Die ersten Messungen zeigen bereits einen spürbaren Performance-Vorteil von SSD. Es kann festgestellt werden, dass lesende und weitreichend schreibende Zugriffe schneller möglich sind als bei herkömmlichen HDDs. Die Messergebnisse zeigen auf, dass die mittleren Gesamtlaufzeiten bei der Nutzung von SSDs stets um einige hundert Millisekunden kürzer sind als bei der Verwendung von HDDs als Logdatei-Speicher.

Diese Ergebnisse basieren zu einem Großteil auf den physikalischen Vorzügen von Solid State Disks. Es sind jedoch auch weitere Optimierungen denkbar, wie beispielsweise die Variation der Logdatei-Größe von standardmäßig 51200 KB auf einen Wert, der einer Blockeinheit auf der SSD entspricht. Weiterhin könnte man durch leeres Beschreiben inaktiver Logdateien den Vorteil des sequenziellen Schreibens verstärkt zum Vorschein kommen lassen. Auch Optimierungen außerhalb der Logdateien sind denkbar - dies soll in weiteren Untersuchungen betrachtet werden.

Literatur

[1] Andre Faustmann, Michael Höding, Gunnar Klein und Ronny Zimmermann: *Oracle-Datenbankadministration für SAP*. Galileo-Verlag, 2007.

[2] Bales, Donald: *Java programming with Oracle JDBC*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[3] Benz, Benjamin und Boi Feddern: *Festplatten ade*. c't Magazin für Computer und Technik, 25(21):100–105, 2007.

[4] Härder, Theo, Karsten Schmidt, Yi Ou und Sebastian Bächle: *Towards Flash Disk Use in Databases - Keeping Performance While Saving Energy?* In *BTW*, Seiten 167–186, 2009.

[5] Höding, Michael: *Flash-Disks - Neue Speicherhierarchieen für Datenbankmanagementsysteme?* In Höpfner, Hagen und Friederike Klan (Herausgeber): *Grundlagen von Datenbanken*, Jahrgang 01/2008 Reihe *Technical Report*, Seiten 126–130. School of Information Technology, International University in Germany, 2008.

[6] Saake, Gunter und Andreas Heuer: *Datenbanken: Implementierungstechniken*. MITP-Verlag, 1999.