Ist das Kunst oder kann das weg?

Holger Bär
Atos
Tübingen

Schlüsselworte
Buffercache, Librarycache, Automatic Big Table Cache, Database Smart Flash Cache

Einleitung
Die Oracle Datenbank bietet mittlerweile gut 15 getrennt zu konfigurierende Cache Bereiche, vom altbekannten Dreigespann Default-, Keep- und Recycle-Cache über den Library Cache bis hin zu selten genutzten oder gar obskur wirkenden Object-Cache oder in 12.2 der data_transfer_cache. Im Vortrag soll ein Überblick über die vorhandenen Caches gegeben und Fallstricke im Umgang mit den neueren Vertretern aufgezeigt werden.

Wie alles begann
Wie so oft war die Migration einer Kundenumgebung auf eine neue Hardware der Anlass, sich mit dem Thema Caches vertieft zu beschäftigen. Kurz nach der erfolgreichen Migration mit zunächst guter Performance kamen plötzlich Klagen, daß die Datenbank zu langsam sei. Eine nähere Analyse zeigte, daß die Leseperformance des Systems deutlich nicht den Erwartungen entsprach. Aus der Datenbanksicht deutete alles auf Probleme mit dem System selber hin, aus Systemsicht schienen die Probleme jedoch nicht so eindeutig zu sein. Eine zeitnahe Verbesserung auf der Hardware-Seite war nicht zu erwarten, daher wurden Alternativen gesucht um wenigstens die Auswirkungen auf der Datenbankseite zu minimieren. Da in der betreffenden Anwendung eine Tabelle sowohl eine zentrale Rolle spielte und gleichzeitig mit einer Größe im einstelligen Gigabyte Bereich noch überschaubar war, drängte sich der Einsatz eines geeigneten Caches direkt auf, um wenn schon nicht die Ursache so doch wenigstens die Auswirkungen für die Anwender zu minimieren. Wie alles ausging wird im Vortrag verraten.

Caching Technologien in Oracle

Cache *([kæʃ], [kaʃ][1]) bezeichnet in der EDV einen schnellen Puffer-Speicher, der (wiederholte) Zugriffe auf ein langsames Hintergrundmedium oder aufwendige Neuberechnungen zu vermeiden hilft. Daten, die bereits einmal geladen oder generiert wurden, verbleiben im Cache, so dass sie bei späterem Bedarf schneller aus diesem abgerufen werden können. Auch können Daten, die vermutlich bald benötigt werden, vorab vom Hintergrundmedium abgerufen und vorerst im Cache bereitgestellt werden (read-ahead).*

Quelle: https://de.wikipedia.org/wiki/Cache

Man kann grob 4 Bereiche unterteilen

- BUFFER_CACHEs mit Default, Keep, Recycle Cache und den verschiedenen db_*N*k_cache_size Parametern, ebenso der Big_Table_Cache sowie der Database Flash Cache
- Der Shared Pool für geparste Statements um die aufwändige Arbeit der wiederholten Statementoptimierung für häufig ausgeführtes SQL zu reduzieren
- Der Result_Cache um das Ergebnis von Selects direct als Resultset zu liefern, oder im Falle von PL/SQL die Ausführung von Funktionen ganz zu vermeiden

- Serverseitig eingestellte, aber nur auf der Clientseite relevante Caches wie der Client Result_cache oder Object_Cache

Auf spezielle Themen wie CELL_FLASH_CACHE bei Exadata oder die im weitestens Sinne ebenfalls zu den Caches gehörende IN-MEMORY Option wird nicht weiter eingegangen.

Data Caches

Beim Thema Cache denkt vermutlich fast jeder zunächst an die verschiedenen Data Caches, darum sollen diese auch zuerst behandelt werden. In der Tabelle werden die Caches und ihre Charakteristik dargestellt.

Cache Parameter	Beschreibung
Db_cache_size	Default Cache, Blockgröße entspricht der db_blocksize
Db_keep_cache_size	Keep Cache, Blockgröße entspricht der db_blocksize, storage Klausel buffer_pool keep
Db_recycle_cache_size	Recycle Cache, Blockgröße entspricht der db_blocksize, storage Klausel buffer_pool recycle
Db_Nk_cache_size	Cache für Segmente die in Tablespaces liegen deren Blockgröße nicht der db_blocksize entspricht für N=[2,4,8,16,32]
Db_big_table_cache_size	Prozent der db_cache_size die für das Caching großer Tabellen zur Verfügung gestellt wird.
Db_flash_cache_size	Größe des Smart Flash Speichers

Mit Ausnahme des Big Table Caches werden alle (Daten-)Caches nach einem modifizierten LRU (least recently used) Mechanismus verwaltet, mit dem Oracle sicherstellt dass selten genutzte Blöcke aus dem Cache entfernt werden können.

Beim Big Table Cache dagegen wird ein Objekt bezogener Mechanismus eingesetzt, der nur die Häufigkeit der Zugriffe auf das Objekt berücksichtigt (die sogenannte Temperatur).

Da die hinter den Datencaches stehenden Mechanismen mittlerweile recht gut dokumentiert sind, soll auf die Funktionsweise im Text nicht weiter eingegangen werden.

Shared Pool caches

Bei dem Shared Pool Caches handelt es sich um Bereiche wie dem Library Cache, Data Dictionary Cache, Result Cache und dem Reserved Pool. Auch der Shared Pool wird über einen modifizierten LRU Mechanismus verwaltet, allerdings mit der zusätzlichen Schwierigkeit, daß hier keine einheitlichen Speicherblöcke von 2-32 Kilobyte zu verwalten sind, sondern Chunks in sehr variablen Größen. Seit Oracle 11g wird der Zugriff auf Hash Buckets nicht mehr über Latches sondern über Mutexe kontrolliert, wodurch die früher häufige Latch Contention deutlich reduziert wurde (ein Latch deckt viele Hash Buckets ab, ein Mutex nur einen einzigen).

Result Cache

Der Result Cache ist wie der Name schon sagt ein Cache für Ergebnisse eines Selects bzw. im Falle des PL/SQL Resultcache von Funktionen. Gespeichert werden keine Blöcke, sondern Resultsets. Gedacht ist der Result Cache für häufig wiederholte Lookup-Statements. Leider zeigte sich in der ersten Implementierung noch in 11g ein Problem wenn die gleichen Statements in mehreren Sessions ausgeführt wurden, da nun statt um Buffer und Shared Pool Latches um den Latch für den Resultcache konkuriert wurde. Serialisierung und Performance Probleme waren das Ergebnis. Über die Verbesserungen in 12c wurde auf den letzten DOAGs Vorträge gehalten, daher soll hier ebenfalls nicht näher eingegangen werden.

Database Smart Flash Cache

Der Database Smart Flash Cache ist quasi eine Erweiterung des Buffer Caches – was aus dem Buffer Cache herausfällt soll in dem (idealerweise erheblich größeren) Flash Cache noch aufbewahrt werden. Dazu sollte dieser auf einem sehr schnellen Flash-Device abgelegt werden, entweder als Datei im Filesystem oder in einer getrennten ASM-Diskgroup. Die Idee dahinter ist die gleiche wie bei jedem anderen 2nd Level Cache: der Zugriff ist zwar langsamer als auf den Hauptspeicher, aber immer noch um Größenordnungen schneller als die Daten erneut von (drehenden) Platten zu laden. Allerdings wird dies ausschliesslich unter Solaris und Oracle Enterprise Linux unterstützt und in Zeiten von Full Database Caching Mode und SSD-only Storage muss man genau hinschauen ob ein Smart Flash Cache einen Vorteil bringt.

Automatic Big Table Cache

Da der Automatic Big Table Cache (im folgenden mit ABTC abgekürzt) über eine andere Strategie wie die bisher erläuterten Caches verwaltet wird (sieht man vom Resultcach ab), soll hier ein genauerer Blick auf die Mechanismen und deren evt. Nachteile geworfen werden. Der ABTC steht in einer Single Instance für serielle und parallele Statements zur Verfügung, im RAC nur für parallele. Damit der ABTC für parallele Statements genutzt werden kann, muss parallel_degree_policy auf AUTO oder ADAPTIVE stehen. Die Größe des ABTC wird über den Parameter db_big_table_cache_percent_target festgelegt und wird als prozentualer Anteil des Buffer Caches angegeben.

Mit Hilfe des ABTC können Tabellen die über der Größe für kleine Tabellen liegen (2% des Buffercache bzw. Parameter "_small_table_threshold") bei einem Full Table Scan in der SGA gecached werden anstelle die Daten über direct read in die PGA des Serverprozess zu laden. Dabei hat die Tabelle eine um so größere Chance in den Cache geladen zu werden, je häufiger sie von einem SQL-Statement verwendet wird, oder wenn der Big Table Cache noch genug freien Platz aufweist um die Tabelle ganz oder in Teilen aufzunehmen.

Jeder Zugriff auf die Tabelle (mittels FULL SCAN) erhöht den Temperaturwert für die Tabelle um 1000. Den aktuellen Inhalt des ABTC kann man der View v$bt_scan_obj_temps entnehmen:

```
select * from v$bt_scan_obj_temps;

       TS#   DATAOBJ# SIZE_IN_BLKS TEMPERATURE POLICY     CACHED_IN_MEM     CON_ID
---------- ---------- ------------ ----------- ---------- ------------- ----------
        12     161717        40524      129000 MEM_ONLY           40524          0
        12     162630        40532       80000 MEM_ONLY           40532          0
        12     162604       405007       62000 MEM_ONLY          405007          0
        12     162875        40533       57000 MEM_ONLY           40533          0
         7     157607        49171       56000 MEM_ONLY           49171          0
         6     162581        75579       40000 MEM_ONLY           75579          0
```

```
             6     162582        69441       33000 MEM_ONLY          69441          0
            12     162891        40594       17000 MEM_ONLY          40594          0
             6     161672        42843       16000 MEM_ONLY          42843          0
             6     162585        42843       15000 MEM_ONLY          42843          0
             6     162583       214707        9000 MEM_PART          87089          0
             6      97563      1556498        8000 DISK                  0          0
            12     162868        79955        5000 DISK                  0          0
             7      95902       189738        5000 DISK                  0          0
            12     162623        79956        4000 DISK                  0          0
             7     138573      1010462        3000 DISK                  0          0
             6     161670       214707        2000 DISK                  0          0
            12     161710        79933        2000 DISK                  0          0
            12     161691       403527        2000 DISK                  0          0
             3    9230112        68988        2000 DISK                  0          0
            12     160959        40528        1000 DISK                  0          0
             6     161669        69441        1000 DISK                  0          0
             6     161668        75579        1000 DISK                  0          0
[…]
```

In dem Beispiel wird erkenntlich, daß eine Tabelle mindestens 10 mal abgefragt werden müsste, um für den BigTableCache in Betracht zu kommen, da die Tabelle mit dem niedrigsten Temperaturwert bereits bei 9000 liegt. Da nun aber der zur Verfügung gestellte Speicherbereich bereits ausgenutzt ist, muss die mit der niedrigsten Temperatur auch wieder entladen werden bevor eine weitere Tabelle in den Cache geladen werden kann.

Abhängig von der größe der beiden Tabellen wird die “kältere” Tabelle teilweise oder ganz aus dem Speicher entfernt. Darüber hinaus gibt es auch einen nicht weiter dokumentierten Mechanismus der vermutlich über einen Zeitstempel die Temperatur der gecacheten Objekte wieder reduziert, so daß lange nicht mehr benutzte Informationen leichter aus dem Cache entfernt werden können. Bei der Reduzierung der Temperatur treten dann auch Temperaturwerte auf, die nicht nur auf den nächsten 1000er Schritt gerundet sind. Zeitlich fallen die Anpassungen mit dem Autotask zum Sammeln der Optimizer Statistiken zusammen, so daß hier ein Zusammenhang vermutet werden kann, insbesondere da nach Abschalten der Autotasks keine Reduzierung und kein Flushen gelöschter Objekte aus dem Cache beobachtet wurden.

Probleme des Automatic Big Table Cache

Wie so oft gilt auch für den ABTC daß man sich genau anschauen muss wie das Feature funktioniert und wann tatsächlich Vorteile damit erzielt werden können. Umgekehrt sollte man auch genau wissen, was von den aktuellen Mechanismen nicht oder nicht korrekt berücksichtigt wird, daher hier eine kurze Aufzählung was bei der Beschäftigung mit ABTC zumindest zu leichten Fragezeichen beim Autor geführt hat. Die Aussagen beziehen sich hauptsächlich auf die Darstellung der v$bt_scan* Views.

- Objekte die gedropt werden bleiben im Cache erhalten, unabhängig ob mit oder ohne Purge gedropt wurde. Erst nach den Autotask-Läufen verschwinden diese z.T. langsam aus dem Cache
- Objekte deren Data_Object_id geändert wurde (Truncate Table, Partition Exchange) bleiben im Cache erhalten
- In der Praxis wurde bei parallel Query ein FullScan mittels Direct-Read beobachtet, obwohl die Voraussetzungen für die Nutzung des ABTC gegeben waren und die fragliche Tabelle zu über 50% bereits im Cache lag

- Objekte die in Tablespaces liegen deren Blockgröße von der Defaultgröße abweicht werden in den ABTC geladen und belegen Speicher den sie nicht nutzen können dürften
- Bugs im Zusammenhang mit Datapump (**MOS DOC.ID** 2060627.1) durch die das Feature unbeabsichtig abgeschalten werden kann oder ORA-600
- Obwohl laut Dokumentation nicht auf Tabellen beschränkt konnte im Versuchsaufbau kein Index in den ABTC geladen werden
- V$bt_scan_cache und v$bt_scan_obj_temps passen von den Werten nicht immer zusammen: sum(cached_in_mem) kann kleiner als memory_buf_alloc, in extremen Fällen sogar größer!
- Ein gezieltes Flushen des ABTC ist zumindest laut Dokumentation über alter system nicht möglich, flushen des Buffer_cache lässt die Einträge in den v$bt_scan* Views zunächst unangetastet – ein erzwungenes Neuladen der gecachten Objekte ist so nicht möglich

Zusammenfassung

Automatic Big Table Cache ist Teil der In-Memory Strategie von Oracle und unterstützt Mixed- bzw. DWH-Workloads. Durch die kontrollierte Reservierung eines Anteils vom Buffer Cache für Tabellen größer des Threshhold kann bei häufig benötigten Tabellen physischer IO erfolgreich reduziert werden durch Vermeidung von wiederholten Direct Read Operationen. ETL-Operationen, die mit Truncate und Exchange Partition arbeiten müssen gut getimed werden, damit das Feature schnellstmöglich greifen kann. Non-Default Blockgrößen sollten vermieden werden.

Database Smart Flash Cache ist als Ergänzung zu sehen, da hier andere (Lese-) Operationen unterstützt werden die von normalem Cache profitieren, wie z.B. Index Fast Full Scans. Während Autotrace in SQL-Plus weiterhin lediglich physical IO meldet, findet man in den Sessionstatistiken einen neuen Wait-Event *“db flash cache multiblock physical read”*bzw. *“db flash cache single block physical read”*.

Kontaktadresse:
Holger Bär
Atos
Hagellocher Weg 73
D-72070 Tübingen

Telefon: +49 (0) 7071-9457-0
Fax: +49 (0) 7071-9457-411
E-Mail holger.baer@atos.net
Internet: pages.atos.net/de/sc/