Exadata X3-2 Database In Memory Machine – Wie geht das?

Konrad Häfeli
Trivadis AG
CH-8152 Glattbrugg

Schlüsselworte:
In-Memory, Exadata, Oracle Database Machine, Flash Cache, Hochverfügbarkeit, High Availability, HA, High Performance

Einleitung
Oracle propagiert die Exadata X3-2 als "Database In Memory Machine".

- ➔ "From disk based with memory for acceleration to primarily memory based with disks for capacity"

Das ist die Aussage von Oracle. Reicht dazu ein grosser Flash-Cache Speicher in den Storage Cells um das Label "In Memory DB" zu setzen, oder ist das nur ein Marketinggeplänkel? In diesem Artikel werden die Ideen und Konzepte von Oracle vorgestellt und die Auswirkungen auf die Praxis beschrieben. Anhand von mehreren Proof of Concepts konnten diese in verschiedenen Kundensituationen getestet werden. Zudem werden auch die Konfigurationsmöglichkeiten der Memory Hierarchie und deren Auswirkungen auf die Performance beleuchtet.

Abb. 1: Exadata X3 Präsentation

Definition von In-Memory
Der Begriff „In-Memory“ ist schon länger bekannt, wird aber in der letzten Zeit von den verschiedenen Anbietern, vor allem im Bereich Appliances, vermehrt strapaziert. Nebst Angeboten von IBM oder Microsoft hat SAP mit HANA[1] (High Performance Analytic Appliance) Druck auf Oracle gemacht diesen Begriff Im Exadata Beschrieb zu haben.
Gartner hat im Hype Cycle for Emerging Technologies, (Source: Gartner.com August 2013[2]) die Begriffe “In-Memory Database Management System” und “In-Memory Analytics” aufgelistet, wobei der Zweite auf dem Weg zum produktiven Einsatz ist während der erste gerade den Gipfel der überhöhten Erwartungen überschritten hat.
Oracle hat mit den Produkten Times-Ten[3] und Oracle In-Memory Database Cache den Begriff auch schon kommerziell geprägt. Kürzlich würde von Larry Ellison an der Oracle Open World 2013 die Database In-Memory Option für 12c[4] vorgestellt.

In den verschiedenen Blogs[5] wurde die Vorstellung der Exadata X3-2 als „Database In-Memory Machine“ kontrovers diskutiert und auch eine Definition für „echte“ IMDBs (In Memory Datebank) gemacht. Vermutlich hat Oracle, nachdem die 12c In-Memory Option auf der Roadmap war, den Bedarf nicht mehr gehabt Exadata als solche zu bezeichnen, sodass nun der Begriff In-Memory wieder aus den (Marketing-) Beschreibungen rausgefallen ist.

Abb. 2: Exadata Header nach dem Launch von Oracle 12c

Darum wird sich dieser Artikel nicht mehr um vage definierte Exadata In-Memory Funktionalität handeln, sondern um real implementierte Flash Cache Devices in den Storage Zellen der Exadata und deren optimalen Einsatz.

Exadata Smart Flash Cache
In den Storage Servern einer Exadata Plattform werden jeweils 4 Sun Flash Accelerator F40 PCIe Karten eingesetzt, welche je 400GB Kapazität haben. Für ein voll ausgebautes Exadata Rack ergibt das über 22TB flash memory. Der Umstand dass diese Flash Devices nicht als Disks angebunden sind, sondern über den PCIe Bus, ermöglicht eine sehr gute Skalierung über die Karten und Server hinweg. Limitierende Disk-Controller wie sie für SSDs (solid state disks) eingesetzt werden können so sehr gut umgangen werden.

[1] Beschreibung SAP HANA: http://www.saphana.com/community/about-hana
[2] Gartner Hype Cycle for Emerging Technologies: http://www.gartner.com/newsroom/id/2575515
[3] Oracle In-Memory Produkte: http://www.oracle.com/technetwork/products/timesten/overview/index.html
[4] OOW13: https://blogs.oracle.com/oracleopenworld/entry/larry_ellison_keynote_everything_runs
[5] http://flashdba.com/category/blog/, http://kevinclosson.wordpress.com/

Es gibt drei Hauptfunktionen der Flash Karten

- Caching von Datenbankobjekten im flash memory
  - Einerseits automatisch
  - Andererseits durch den DBA gemanaged (object pinning)
- Device für das Schreiben des Datenbank logging
- Bereitstellen von Zell-Disken aus flash memory

Übrigens haben diese Devices eine automatische Fehlererkennung, welche defekte Komponenten aus dem Betrieb nimmt und somit die Funktionalität auch hochverfügbar zur Verfügung stellt.

Flash Cache Funktionalität
Exadata Smart Flash Cache versteht unterschiedliche Typen von Datenbank I/O, entsprechend diesen werden die Speicherzugriffe gecached oder eben nicht.

Caching:
- Regelmässig zugegriffenen Daten- und Index-Blöcke
- Controlfile Schreib- und Lesezugriffe
- Fileheader Schreib- und Lesezugriffe
- Vom DBA definierte Objektblock-Zugriffe (gepinnte Objekte)

No-Caching:
- I/O auf gespiegelte Daten
- Backup bezogene I/Os
- Datapump bezogene I/Os
- Datenfile Formatierungen
- Scans über Tabellen hinweg

Jede I/O Operation eines Datenbankobjektes ist mit einem „Tag“ versehen welches die CELL_FLASH_CACHE (DEFAULT/KEEP/NONE) Funktion betrifft. Der Default spezifiziert, dass Smart Flash Cache für Lese-Operationen eingesetzt wird. Mit KEEP kann das Caching forciert und mit NONE ausgeschalten werden. Die Schreiboperationen gehen direkt auf die (langsamen Disks).

```
alter table <table_name> storage (cell_flash_cache KEEP);
```

Flash Cache Write-Back
Damit der Flash-Speicher auch für Write-Operationen eingesetzt werden kann muss das zuerst auf den jeweiligen Storage-Zellen konfiguriert werden. Der bestehende Flash Cache Modus (Default: „WriteThrough“) muss zuerst gelöscht werden, danach muss der Zellserver gestoppt, der „WriteBack“ Modus gesetzt und der Zellserver wieder gestartet werden

```
CellCLI> list cell attributes flashcachemode
WriteThrough

CellCLI> drop flashcache
Flash cache ... successfully dropped
```

```
CellCLI> alter cell shutdown services cellsrv
Stopping CELLSRV services...
The SHUTDOWN of CELLSRV services was successful.

CellCLI> alter cell flashCacheMode = WriteBack
Cell ... successfully altered

CellCLI> alter cell startup services cellsrv
Starting CELLSRV services...
The STARTUP of CELLSRV services was successful.

CellCLI> create flashcache all
Flash cache ... successfully created
```

Der Einsatz von „WriteBack“ Caching bringt Vorteile bei Applikationen mit grosser Schreibintensität. Im Gegensatz zu Leseoptimierungen durch Caching, sind diese aber nicht immer offensichtlich, da der Databasewriter (DBWR) immer im Hintergrund schreibt und vielfach andere „Flaschenhälse“ als das DBWR I/O zum Tragen kommen. Wenn dieses eingeschalte wird, dann sollte man durch Monitoring prüfen, dass der gewünschte Effekt erreicht wird. Dies kann durch Zellstatistiken (read/write auf disk und flash celldisks) gemacht werden, oder aber ab Oracle 11.2.0.4 (verfügbar seit dem 28. August 2013) mit den DB Statistiken “Physical Write Requests Optimized”.

Flash logging Funktionalität

Es ist ein Faktum, dass für Transaktionssysteme die Schreibgeschwindigkeit des Logwriters ausschlaggebend ist für die Performance der Applikation. Die durchschnittliche Wartezeit kann in der DB-Statistik „log file sync wait“ eingesehen werden. Das Problem ist, dass nebst generell limitierenden Antwortzeiten auch sogenannte „hiccups“ auftreten können. Das sind zeitweise sehr lange Antwortzeiten, die nicht nur auf Disks sondern auch bei Flashspeicher Einsatz vorkommen können. Dies ist bedingt durch Lösch-Zyklen und automatische Datenumverteilung, welche sicherstellen soll dass die Speicherzellen gleichmässig genutzt werden (sogenanntes „wear leveling“).

Die Funktion wird sichergestellt, indem eine vernachlässigbare Menge an flash memory (default ist 512MB) als temporärer Storage für die Redolog I/Os bereitgestellt wird. Die Redologfiles selbst sind immer noch in voller Grösse auf den Disks abgelegt (ASM Diskgroups). Die interne Funktion macht zwei Schreiboperationen, die Eine auf Disk, die Andere auf Flash. Diejenige welche zuerst das Acknowledge zurückbringt lässt die Applikation weiterarbeiten. Dieser Mechanismus läuft im Hintergrund ab und ist vollumfänglich Transparent für die Applikation.

Alle bestehenden Best Practices bezüglich Anzahl und Grösse der Redologfiles sind weiterhin gültig, der einzige Unterschied ist:

- ➔ **Konstante „low latency“ beim Schreiben von Redolog Daten**

Die Funktionalität kann pro Zellserver über eine Abfrage der Metrikhistorie geprüft werden:

```
CELLCLI> LIST METRICHISTORY WHERE objectType = 'FLASHLOG' AND -
metricValue != 0 AND name like 'FL_EFFICIENCY_PERCENTAGE.*' -
ATTRIBUTES name, metricObjectName, metricValue, collectionTime

FL_EFFICIENCY_PERCENTAGE         FLASHLOG 100 %   2013-09-25T22:59
FL_EFFICIENCY_PERCENTAGE_HOUR    FLASHLOG 100 %   2013-09-25T22:59
FL_EFFICIENCY_PERCENTAGE         FLASHLOG 100 %   2013-09-25T23:00
FL_EFFICIENCY_PERCENTAGE_HOUR    FLASHLOG 100 %   2013-09-25T23:00
FL_EFFICIENCY_PERCENTAGE         FLASHLOG 100 %   2013-09-25T23:01
FL_EFFICIENCY_PERCENTAGE_HOUR    FLASHLOG 100 %   2013-09-25T23:01
```

Flash Disk Funktionalität

Das Default-Verhalten weist alles flash memory dem Flash Cache resp. auch dem Flash Logging zu. Es können aber auch optional aus den Flash Devices logische flash disks erstellt werden. Diese können wie gewöhnliche Exadata cell disks eingesetzt und für ASM Diskgruppen zur Verfügung gestellt werden, welche dann vollumfänglich auf flash memory liegen. Die folgende Grafik veranschaulicht das Storage Layout.

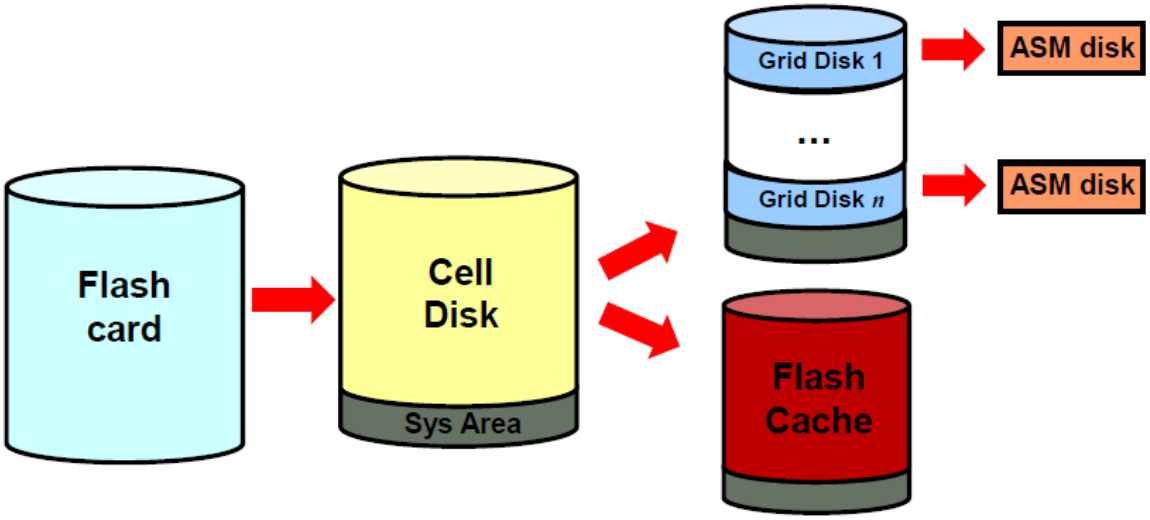

Abb. 3: Flash Card Storage Layout

Die Limite ist maximal 4 cell disks pro Flashkarte, das heisst maximal 16 pro Zellserver.

Erstellung von flash griddisks mit CELLCLI Kommandos:

```
CELLCLI>
drop flashlog;
drop flashcache;

create griddisk FLS_CD_00_dm01cel01 celldisk=FD_00_dm01cel01
create griddisk FLS_CD_01_dm01cel01 celldisk=FD_01_dm01cel01
create griddisk FLS_CD_02_dm01cel01 celldisk=FD_02_dm01cel01
create griddisk FLS_CD_03_dm01cel01 celldisk=FD_03_dm01cel01

create flashlog celldisk  ='FD_04_dm01cel01,FD_05_dm01cel01, -
                            FD_06_dm01cel01,FD_07_dm01cel01'
create flashcache celldisk='FD_04_dm01cel01,FD_05_dm01cel01, -
                            FD_06_dm01cel01,FD_07_dm01cel01'
```

Erstellung von flash diskgroups mit SQL Kommandos:

```
SQL>
create diskgroup FLASH_DG external redundancy disk
               'o/192.168.10.3/FLS_CD_00_dm01cel01,
               'o/192.168.10.3/FLS_CD_01_dm01cel01,
               'o/192.168.10.3/FLS_CD_02_dm01cel01,
               'o/192.168.10.3/FLS_CD_03_dm01cel01'
               'o/192.168.10.4/FLS_CD_00_dm01cel02,
               'o/192.168.10.4/FLS_CD_01_dm01cel02,
               'o/192.168.10.4/FLS_CD_02_dm01cel02,
               'o/192.168.10.4/FLS_CD_03_dm01cel02,
               'o/192.168.10.5/FLS_CD_03_dm01cel03,
               'o/192.168.10.5/FLS_CD_01_dm01cel03,
               'o/192.168.10.5/FLS_CD_02_dm01cel03,
               'o/192.168.10.5/FLS_CD_00_dm01cel03,
attribute
               'cell.smart_scan_capable'='TRUE',
               'compatible.asm'='11.2.0.3.0',
               'compatible.rdbms'='11.2.0.3'
               'au_size’=’4M';
```

Verschieben von Tablespaces auf flash diskgroups mit RMAN Kommandos:

```
rman> sql 'alter tablespace exatest offline';
rman> copy datafile 6 to '+FLASH_DG';
rman> copy datafile 7 to '+FLASH_DG';
rman> switch datafile 6 to copy;
rman> switch datafile 7 to copy;
rman> recover tablespace exatest;
rman> sql 'alter tablespace exatest online';
```

Test Resultate über die verschiedenen Funktionalitäten

Natürlich interessiert bei solchen Funktionen immer auch welche Konfigurationen welche Performancevorteile bringen. Wir haben ein relativ einfaches Testsetup gewählt und von zwei Knoten aus eine RAC Datenbank mit Swingbench[6] belastet, dabei wurde die SGA mit 1 GB eher klein gewählt, damit möglichst viel Aktivität auf den Disken resp. dem Flash Cache forciert werden konnte. Es wurde der Swingbench Stress Test Benchmark ausgewählt, mit einer Anpassung der Konfiguration auf 70 zu 30 Prozent zu Gunsten der Schreiboperationen. Nach einem Warm-Up Durchgang wurden 4 Zyklen a 5 Minuten mit dem jeweiligen Testszenario gefahren.

Folgende Szenarien wurden geprüft:

- WA: Kein writeback, kein flashcache, kein flashlog, kein cell keeping (als Baseline)
- FCO: nur flashcache
- FLO: nur flashlog
- FCLO: flashcache und flashlog (Exadata default configuration)
- …WB: zusätzlich writeback
- …OP: zusätzlich Object Pinning
- …DG: asm diskgroup mit flashcache, flashcache und flashlog

Dabei ergaben sich folgend grafisch dargestellte Testresultate:

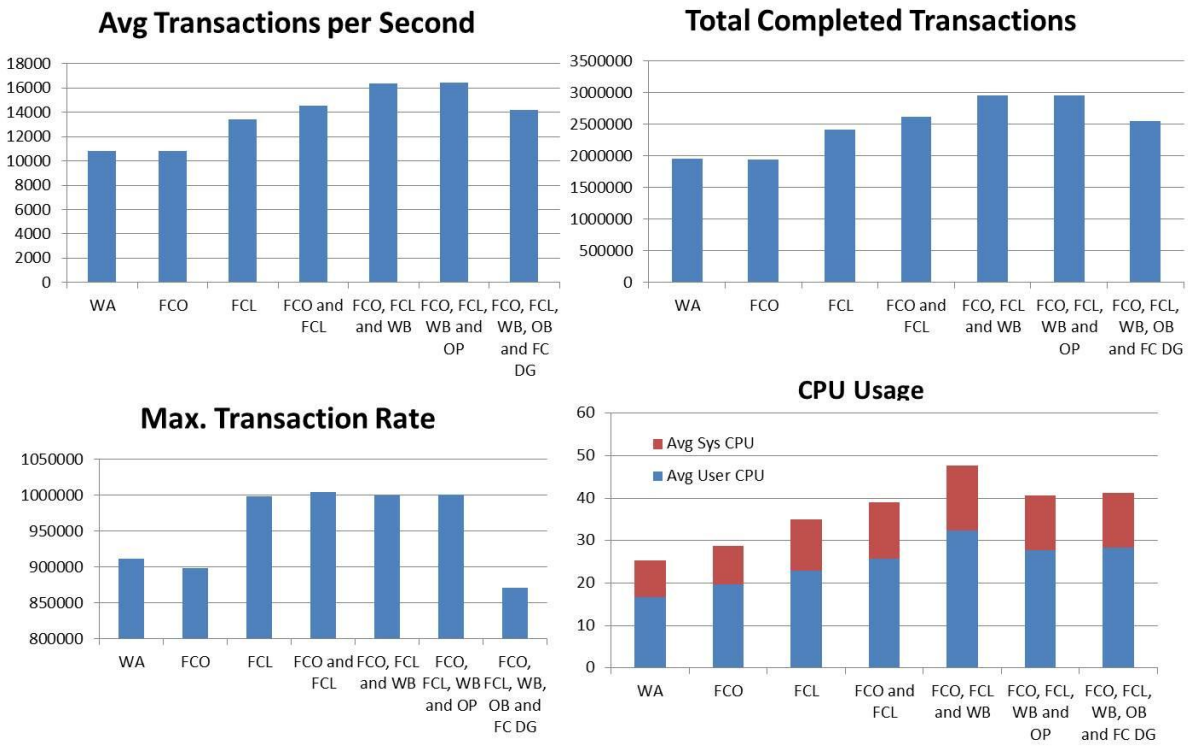

Abb. 4: Testresultate Flash-Cache Einsatzszenarien

[6] Swingbench Tool: http://dominicgiles.com/swingbench.html

Analyse der Resultate:

- Flash Logging verbessert die Performance um 24%
- Der Default Einsatz von flash cache und log ergibt Verbesserungen um 34%
- Einschalten des Write-Back Cache verbessert um weitere 13% gegenüber dem Default
- Object pinning hatte keinen messbaren Vorteil gegenüber dem normalen Caching, hingegen war die CPU Last markant tiefer
- Der Einsatz von Flash Diskgroups kann sich negativ auswirken, denn sie limitieren die Ressourcen für das automatische Caching

Fazit

In der Einführung zum Thema wurde klar dargelegt, dass die Exadata X3-2 keine In-Memory Database Machine im Sinne der Terminologie ist. Nichts destotrotz ist sie eine eindrückliche flash cache basierte Datenbank Plattform. Innerhalb des Systems hat es einzigartige Funktionalität welche einsatzspezifisch konfiguriert werden kann.

- Flash caching
- Flash logging
- Write-Back caching
- Flash cache diskgrouping

Wie bei den meisten Dingen der Exadata Plattform sind die vordefinierten Einstellungen gut. Es bedingt also praktisch kein zusätzliches Tuning in dem Bereich. Wir schlagen aber dennoch vor den Einsatz von Write-Back cache zu prüfen, denn das beschleunigt die Hintergrundprozesse (DBWR), was sich bei unseren Tests positiv auf die allgemeine Performance ausgewirkt hat.

Wenn Sie aber „richtige“ In-Memory Funktionalität in Ihrer Exadata Oracle Datenbank haben möchten, dann freuen Sie sich auf den Release der 12c Oracle Database In-Memory Option „irgendwann“ im nächsten Jahr…

Viel Erfolg beim Konfigurieren und Betreiben Ihrer Exadata Umgebung. Wenn Sie Fragen dazu haben steht Ihnen unser Exadata-Competence-Team gerne zur Seite und unterstützt Sie mit den Erfahrungen aus dem täglichen Einsatz.

Kontaktadresse:

Konrad Häfeli
Trivadis AG
Europastrasse 5
CH-8152 Glattbrugg

Telefon: +41 (0) 58 459 55 55
Fax: +41 (9) 58 459 55 95
E-Mail konrad.haefeli@trivadis.com
Internet: www.trivadis.com