ADVANCED PERFORMANCE TOPICS FOR ORACLE/VXVM/SYMMETRIX CONFIGURATIONS

E. BÂRLĂDEANU*

În cursul exploatării unui sistem integrat de gestiune de tip ERP pot apare situaţii excepţionale (execuţia simultană unui volum mare de lucrări sau recuperarea după un "crash", când viteza este esenţială) care necesită un plus de performanţă, fără însă a declanşa achiziţii suplimentare de resurse. Într-un sistem informatic integrat ERP, optimizat conform specificaţiilor producătorului şi fără a achiziţiona resurse suplimentare, singura posibilitate de a avea asigurat un plus de performanţă, constă în optimizarea distribuţiei fizice a datelor, precum şi folosirea unor detalii specifice ale configuraţiei (de exemplu structura memoriei "cache"). Articolul se referă la configuraţia SAP R/3, ORACLE, Solaris, VxVM, Symmetrix, dar metoda poate fi extinsă ca o procedură de optimizare post-implementare.

During the deployment of an ERP (Enterprise Resource Planning) integrated system, exeception al situations can occur (simultaneous execution of a large number of tasks, or after a crash, when the recovery speed is essential), these situations requiring higher perfomance levels, without acquiring additional resources. In an ERP integrated information system, optimized according to the manufacturer's specifications and without using extra resources, the only possibility of achieving higher perfomance levels consists in the optimization of the physical distribution of data, along with the use of specific details of the configuration (for example the structure of the cache memory of a disk array). The paper is based on a SAP R/3, ORACLE, Solaris, VxVM, Symmetrix configuration, but the method can be extended as a post-implementation optimization procedure.

Keywords: physical layout, cache saturation, spindle contention, performance, scalability.

Index of Acronyms: DBA - **D**ata**B**ase **A**dministrator; **VxVM** – Veritas Storage Software Management [6]; **Symmetrix** - an EMC[2] networked storage solution [7].

* Eng, SAP R/3 Senior Technical Consultant; Bucharest, ROMANIA.

Introduction: goals/non-goals

The focus of this paper is to provide physical layout strategies and a performance analysis methodology for a high-end SAP R/3/Oracle/Solaris /VxVM /Symmetrix configuration.

We will carefully examine the realities of *cache saturation* and provide short term solutions for situations where increasing the size of the Symmetrix cache may not be feasible for financial reasons.

This article is not an introductory tutorial on either Oracle's RDBMS, the Veritas volume manager or EMC^2's Symmetrix. The assumption is that some very specific tools, as DBTuner/Symmetrix Optimizer, are not deployed.

First of all, we present some *performance tuning approaches.*

To oversimplify, performance of any system can be tuned by:

1. Asking for *less resources* while delivering the same results.
2. Providing those results *faster*.

Asking for *less resources*, in the case of Oracle physical I/O, typically means tuning the SQL commands to minimize demand on physical device activity. The nature of an Oracle application involves a technology stack with upper layers on the demand side and lower layers being the suppliers. If you can identify high demand spots within the upper layer (the Oracle database) and low supply spots within the lower layer (the Symmetrix) and determine where they overlap, you have identified an I/O tuning opportunity.

Providing results *faster* means:

- ensuring that the Symmetrix cache is properly sized;
- identifying and removing bottlenecks such as physical device contention in situations where the cache is not 100% efficient.

A key factor in the performance and scalability of a Symmetrix array supporting an Oracle database is *the physical layout*.

1. Why is the physical layout inside the disk array so important ?

If nothing else, the time it takes to recover an Oracle database after a crash, an extremely important time in a production system, will most likely be affected by the size of the Symmetrix cache. If the cache has been sized for normal operation, performance levels during recovery go back to those of the

underlying spinning media.

Very high end enterprises experience severe I/O performance degradation during month-end processing due to *cache saturation*. The size of the active/working set grows very aggressively [1], but the budgets for maintaining an adequate size for the cache are not growing at the same rate.

Still some things never change: even in the era of cached storage arrays, it is still a good idea to place data and indexes on separate spindles. Cases are known where ignoring that advice has prevented applications from scaling.

At high capacity sites, Symmetrix cache hit rates below 80% are a fact of life, even with 32 GB of cache. 80% cache quality means the remaining 20% is the slowest possible I/O. It's I/O that has to go all the way to the spinning media.

Additionally, cache quality may additionally be interfered with by background activities such as instant splits.

Increasing awareness of scalability issues with spinning media now prompts high end customers to require that their disks only be filled to 50% of capacity and that data reside on the outer half of the disk.

The focus of our paper is to achieve as little of the slowest thing as possible without buying more cache.

2. What logical vs. physical means at different levels

Examples featured in this paper demonstrate how the information translation is done across all levels of virtualization, as follows:

Within the Oracle database:

- *logical*: tables, indexes, rollback segments and other database objects.
- *physical*: datafiles.

Within the file system:

- *logical* = files.
- *physical* = VxVM volumes.

Within the underlying OS:

- l*ogical* = VxVM volumes.
- *physical* = c#t#d# disks / Symmetrix HyperVolumes.

Within the Symmetrix:

- *logical* = HyperVolumes Extension (HVE).
- *physical* = disk slices that make up the mirror set / MetaVolume.

At the physical level, a Symmetrix HVE could be one of the following:

- A disk partition.
- A mirror set (each side of the mirror is a disk partition).
- A stripe set (also known as MetaVolume - typically RAID 1+0).

Inside the Symmetrix [7], an HVE is uniquely identified using a 3-digit hexadecimal ID. For each HVE, the Symmetrix presents a logical disk to the operating system. Under Solaris [5], logical disks can be displayed using the format utility. This one-to-one relationship between the HVEs' 3-digit IDs and c#t#d#s2 logical disks is stored in memory in various components of the Symmetrix as well as in a special file called the BIN file. Determining the physical placement of data as defined by the BIN file can be done by the DBA with appropriate privileges.

The important problems of performance appeared [1] are usually referring to some tranzactions, implying few extensive used tables. Some tools, based on facilities of above mentioned configuration and shown later in this paper, will enable the DBA to extract interesting statistic information about:

- Average disks and service time response time;
- HyperVolumeExtensions where the "problem" objects (tables) are placed;
- Physical location of HyperVolume partitions (outer vs. inner regions of the disk).

3. Advanced Performance Topics

A performance analysis methodology, based on the above mentioned configuration, will be presented. Supposing we have a well-defined performance „problem" (tranzactions, tables), we will follow the next steps:

3.1 Analysis of overly busy disks / volumes.
3.2 Correlation with disk service times.
3.3 Spindle contention analysis.
3.4 Variable disk geometry makes the outer half more attractive.
3.5 Cache allocation and I/O performance of redo generation.

3. 1 Analysis of overly busy disks / volumes.

We start the analysis of most busy disks/volumes, using a facilitate provided by the operating system: **vxstat** command.

```
# vxstat -i 10 -dv
                    OPERATIONS        BLOCKS          AVG TIME(ms)
TYP NAME          READ    WRITE     READ    WRITE   READ   WRITE

Thu Sep 16 10:48:47 2001
dm  c1t10d1         50       34      147     1543    57.0    71.8   <------ over 20 ms - bad response time
...
Thu Sep 16 10:48:57 2001
dm  c1t10d1         63       28      285     1316    64.4    85.1   <------ over 20 ms - bad response time
...
#
```

It can see the c1t10d1 disk with a high average response time. High average disks/volumes response times indicate tuning opportunities within both the application (SQL) and the physical layout, respectively. We consider the SQL commands already optimized [3, 4], according to manufacturer's specification, The only scope of this paper is to provide physical layout strategies.

3.2. Correlation with disk service times

We can improve previous analysis using the **SE** toolkit [10, 1]. The advantage of the SE toolkit is to allow the DBA to zero in on worst offender disks by filtering out average service times below a a specified threshold.

```
# /opt/RICHPse/bin/se iomonitor.se
iomonitor.se thresholds set at 30.0 ms service time and 20.0 % busy
iomonitor.se detected slow disk(s): Wed Nov 18 14:43:15 2001
disk         r/s   w/s   Kr/s   Kw/s   wait actv  svc_t  %w    %b    delay
c1t10d1      0.0  189.6   0.0   4518.4  9.0  2.0   44.9   100   100   10969.3  ←high svc_t
c2t16d80     0.0  192.3   0.0   4462.7  9.0  2.0   43.3   100   100   10972.4
c2t9d24      0.0  273.8   0.0   8693.3  9.0  2.0   41.9   100   100   10978.4
……..
```

A high average service time for disk **c1t10d1** is detected.

Poor service times indicate physical disk contention/thrashing of actuator arms. Reasons for less than optimal disk service times may be:

- Cache is chronically saturated.
- Unusual bursts of database activity (cache temporarily saturated).

As an alternative way, we can correlate poor volume service times with:

- Activity levels derived from V$FILESTAT.
- Output from database **trace files** (10046 event, [4]): elapsed times (ela=...) for 'db file scattered / sequential read' .

Note: This analysis is especially useful if the logical and physical layout of the Oracle database is consistent, i.e. high activity tables/indexes have been isolated within their own tablespaces. In this case, volume performance information from **vxstat** can be very accurate.

Now we can use a specific command of operating system**, symrdb,** that establish a relation between disk/volume, a specified table of database and its Symmetrix location (HVE).

```
# symrdb -type ORACLE -db DSS -tbs INVTSP show TABLE  INVOICE_CUBE

DATABASE                                        : DSS
TABLESPACE NAME                                 : INVTSP
DATABASE TABLE NAME (ORACLE 8.1.7.2.0)  : INVOICE_CUBE

 Database Table Owner                       : ORADSS
 Database Table Type                        : Regular Table

 Number of Database Files: 3
 {
 1) Database File Name: /dev/vx/rdsk/oradssdg/dss_vol_01

  Absolute Path                  : /dev/vx/rdsk/oradssdg/dss_vol_01
  Resolved Object Type           : SunOS VxVM Logical Volume
  Resolved Object Size           : 1512m
  Number of Physical Devices     : 3
  Number of Mirrors for object   : 1
  {
  1) Mirror Configuration

     Mirror Stripe Columns    : 3
     Mirror Stripe Size       : 64k
     Mirror Physical Extents  : 3
     {
     ---------------------------------------------------------------------------------------
                          Sym
        Size  SymID    Dev   Offset  PdevName                Offset
```

```
    ---------------------------------------------------------------------------------------------

          504m  03731   078     1m  /dev/rdsk/c1t10d1s2          1m  ← HVE
          504m  03731   079     1m  /dev/rdsk/c1t10d2s2          1m
          504m  03731   07A     1m  /dev/rdsk/c1t10d3s2          1m

    }
...
 2) Database File Name: /dev/vx/rdsk/oradssdg/dss_vol_02

...
```

We have identified the HVE location as SymDev 078.

3.3.Spindle contention analysis

Once we know what HyperVolumes a host object is sitting on, it is interesting to go one step further and identify - for each one of the disk slices that make up the mirror set - which other HyperVolumes share the same spindle. This information will help identify contention and is especially useful when correlating information across different hosts sharing the same Symmetrix.
To get started, let's focus on HyperVolume 078, which is a mirror set. We can use the **symdev show** command:

```
# symdev show 078 | grep "Director, Interface, TID"
    Disk [Director, Interface, TID]      : [01A, D, 3]
    Disk [Director, Interface, TID]      : [16B, C, 0]
#
```

We have now identified the spindles that store HVE 078's data. They are *01A, D, 3* and *16B, C, 0*. The specified spindles store data for other HVE's as well. To find them, a command **symspindle** can be used to combine results from the above mentioned commands into one consolidated output.

```
# ./symspindle | grep "01A, D, 3"
038  2-Way Mir       : [01A, D, 3]
078  2-Way Mir       : [01A, D, 3]
1C8  BCV             : [01A, D, 3]
1E7  BCV             : [01A, D, 3]
248  BCV             : [01A, D, 3]
2B4  RDF1+Mir        : [01A, D, 3]
```

```
3A0  RDF1-BCV        : [01A, D, 3]
51C  2-Way BCV Mir : [01A, D, 3]
52B  2-Way Mir         : [01A, D, 3]
...
#
# ./symspindle | grep "16B, C, 0"
038  2-Way Mir           : [16B, C, 0]
078  2-Way Mir           : [16B, C, 0]
0C8  BCV                 : [16B, C, 0]
0E7  BCV                 : [16B, C, 0]
148  BCV                 : [16B, C, 0]
2B4  RDF1+Mir         : [16B, C, 0]
3BF  RDF1-BCV         : [16B, C, 0]
51C  2-Way BCV Mir : [16B, C, 0]
52B  2-Way Mir         : [16B, C, 0]
...
#
```

Other important items of interest can be obtained using the **symdev show** ### command, where ### is HVE number.

So that, this command provides additional information as: "HyperNumber", "MegaBytes" and "DiskCapacity".

```
# symdev show 078 | grep "Hyper Number"
      HyperNumber                    :    20
#
# symdev show 078 | grep "MegaBytes"
      MegaBytes                      :   449
#
# symdev show 078 | grep "Disk Capacity"
      DiskCapacity                   :    34733m
 #
```

This information enables a very interesting application: determining whether or not, a disk slice is/is not physically placed on the outer half of the disk platter.

HyperNumbers start with 1 at the outer edge and go up as we move along the disk radius going towards the center.

Supposing that HyperNumber 1 is 449 MB in size as well, we now know that HyperNumber 2 runs from MegaByte 450 up to 898, and so far. This analysis can very easily be scripted.

We will find the physical placement of HyperVolume partitions (outer vs. inner regions of the disk), using the **symhyperplace** command:

```
# ./symhyperplace "01A, D, 3"
Sym Hyper# Hyper     Cumul. size HalfDisk Outer
Dev       Size (MB) (MB)       (MB)     Half
--- ------ --------- ----------- -------- ------------
038     1     449       449  17366.5    Yes
6A0   2       449       898  17366.5    Yes
1C8     3     1796       2694  17366.5  Yes
1E7     4     1796       4490  17366.5  Yes
248     5     1796       6286  17366.5  Yes
267     6     1796       8082  17366.5  Yes
...
4EC     17      449      16613  17366.5  Yes
50F     18      449      17062  17366.5  Yes
5F8     19      449      17511  17366.5  No
078   20      449      17960  17366.5    No   ← HVE 078 is placed on inner half .
6F2     21      449      18409  17366.5  No
#
```

So, we can see HyperNumber 20 sitting on inner half.

3.4 Variable disk geometry makes the outer half more attractive

We supposed a permanent performance problem in our ERP system: a specified tranzaction, using a specified table/tables, needs a *faster* disk acces time. But:

- According to manufacturer's specification, the SQL commands are already optimized [3, 4] and

- Physical layout of the Oracle database is consistent, i.e. high activity tables/indexes have been isolated within their own tablespaces.
- No chance to get additional resources.

The only way to obtain an additional performance is to improve physical disks layout configuration. First, we can use a variable disk geometry.

Variable disk geometry makes the outer tracks of the disk *more attractive for highly performance-critical data.* The outer tracks of the disk can contain more sectors and thus store more data than the inner ones. This has two consequences:

- More data is retrieved from an outer track in the same amount of time (one rotation).
- Along the radius, the distance between both ends of the outer half is smaller, resulting in greatly improved seek times, the most expensive operation during an I/O request. The average seek time is an exponential function of radial distance covered.

Experience shows that storing 72 GB worth of Oracle datafiles on the outer halves of 2 disks results in 4 times better performance compared to storing it all on a single 72 GB disk. This has been demonstrated during a stress test involving extremely heavy, mixed workloads - OLTP and reporting (Other configurations may yield different results).

The remaining inner 50% needs not stay unutilized. At high capacity sites it's used for dump files (Oracle export files, data warehouse extracts), "landing strips". These file systems remain unmounted during peak business hours.
According to above considerations, the HVE 078 must be placed on outer half.

3.5.Cache allocation and I/O performance of redo generation

The system cache - shared by all SymDevs - is logically divided so that each active HyperVolume has at least one "cache allocation". The size of one "cache allocation" is approximately equal to the size of the cache divided by the number of SymDevs. No account is taken of the size of the Symmetrix Device. A small HyperVolume gets just as much space in the cache as a large one.

You can use this to your benefit to ensure that redo log I/O always takes priority and happens at electronic speeds regardless of the intensity of redo generation. You just create HyperVolumes of e.g. 20 MB in size for your redo logs - provided the cache allocation is 20 MB or larger. When Oracle generates redo intensively, log files will entirely reside in the system cache.

Almost as good as solid state disk! This is the second way to improve our sistem.

Conclusions. Recommendations

In an ERP system, optimized according to manufacturer's specification [9, 3] and without additional hardware/financial resources, the only way of achieving higher perfomance levels consist in use of:

- variable disk geometry, placing the performance-critical data on the outer half,
- specific details of the configuration (i.e. the structure of the cache memory).

The procedure is a *post*-implementation performance optimization (**"**The post-implementation optimization plan is the key to a successfully integrated enterprise resource planning system", [2]).

Using common commands of the above configuration [5, 6, 7], for a defined performance problem, we can find the physical location of tables on the disk.

Still we refer here to SAP R/3/Oracle/Solaris /VxVM /Symmetrix configuration, the procedure can be extended. Finally, some recommendations:

- Ensure the Oracle database is properly laid out - logically and physically.
- Know where the data lives. Placing database objects such as to avoid spinning media related conflicts is the Oracle recommended approach and will provide much better performance than tuning the Symmetrix.
- Eliminate spindle contention / high disk service times.
- Tune the entire technology stack, not just specific components, one at a time (holistic approach).
- Tune according to cost / benefit ratios.
- Be aware of how logical I/O translates into physical I/O across the virtualization layers.
- Revisit performance on an ongoing basis as the amount of data grows.

R E F E R E N C E S

1. *E. Bârlădeanu* – Analiza şi conducerea sistemelor informatice integrate de tip ERP. Optimizarea *performanţei*, Teză de doctorat, Universitatea "POLITEHNICA" Bucureşti, 2005.
2. *White Paper Stephen Faleti:* ERP Optimization: Maximized ROI Through Improved Performance http://www.parsongroup.com/documents/erp_exchange.pdf
3. *T. Schneider* **–** SAP R/3 Performance Optimierung – Analyse und Tuning Addison-Wesley London 1999;
4. *R. Niemec* - ORACLE Performance Tuning – Osborne/McGraw-Hill 2000
5. *A. Cockroft* - Sun Performance and Tuning - 3rd Edition. O'Reilly & Associates, 2001;
6. http://www.cuddletech.com/veritas/
7. http://www.symmetrix.ch/

8. http://www.sun.com/servers/entry/v40z/
9. *B. Rudiger* - Die Technologie des SAP - Systems: Basis fuer betriebwirtschaftliche Anwendungen, Addison – Wesley Longman 1998;
10. http://www.setoolkit.com