



Optimizing Oracle 10g on Linux Using Automated Storage Management

White Paper

© Copyright Quest® Software, Inc. 2005. All rights reserved.

The information in this publication is furnished for information use only, does not constitute a commitment from Quest Software Inc. of any features or functions discussed and is subject to change without notice. Quest Software, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this publication.

Last revised: April 2005

TABLE OF CONTENTS

10G - EXCITING AND USEFUL	4
TRADITIONAL METHODS	5
THE NEW METHODS.....	7
EASIER TO SETUP	9
EASIER TO CHANGE	14
SOME PERFORMANCE TOO	17
ABOUT THE AUTHOR.....	18
ABOUT QUEST SOFTWARE, INC.	19
CONTACTING QUEST SOFTWARE	19
TRADEMARKS.....	19

10G - EXCITING AND USEFUL

I often react to new Oracle releases like Steve Martin acted in the movie “The Jerk” when the new phone books arrive. I cannot help but to get excited about all the new technology that Oracle routinely delivers. But with the release of Oracle 10g, that enthusiasm cannot be overstated. And while this Oracle version delivers numerous nifty new features, we’ll examine just one, Automated Storage Management (ASM), which makes Linux database disk space management a snap.

With today’s explosive database size requirements and the proliferation of SAN (Storage Area Network) and NAS (Network Attached Storage) disk technologies, both the system administrator and the DBA must manage hundreds to thousands of disks – sometimes just for one database! Thus the task of planning, initializing, allocating, managing and tuning of so many disks becomes unwieldy. Capitulation is often inevitable. Thus many shops simply treat the disk storage farm as a black box, thus abstracting that complexity away from the database. The phrases “you don’t need to know” and “just trust the hardware to handle it” are often given as justification. Frequently this black box approach can lead to database IO bottlenecks that are time consuming to diagnose and remedy. While for those few shops where the DBA does still in fact manage all those disks, the task consumes far too much of their precious time. Neither scenario should be generally acceptable, as both represent the extremes.

Oracle 10g’s new Automated Storage Management directly addresses these real world scenarios by providing an effective and simple middle ground solution. Now instead of spending inordinate amounts of time managing disk complexities or abstracting it away from the database in the hopes that all is well, we can simply permit Oracle 10g’s ASM to manage it all. We can now merely allocate disks to Oracle with preferences for striping and mirroring stated via templates, and let ASM manage the space – thus eliminating the need for traditional tools used to manage lots of disk space: a Logical Volume Manager (LVM), file systems and the numerous commands necessary to manage both. Thus Linux database servers can now be deployed more easily and quickly for ever growing database size requirements - and with far greater IO efficiency than before.

In this article, we’ll be examining the detailed differences on Linux of using a LVM with cooked files versus Oracle 10g’s ASM. For a more general ASM overview, I recommend reading the OTN article titled [“Storage on Automatic”](#) by Morris-Murphy.

TRADITIONAL METHODS

To begin, let's review how DBA's have historically allocated disk space to Oracle. We do this for two reasons. First and foremost, to ensure that we clearly see just how much simpler ASM makes the diagram. That simplicity will expose itself as "cleaner pictures", which translates to fewer steps to setup and fewer commands to manage. And second, we'll mention some scenarios where each option might make sense. This will help us to see how the newer technologies and all their options fit into the overall database storage picture. **Figure 1** shows the traditional options.

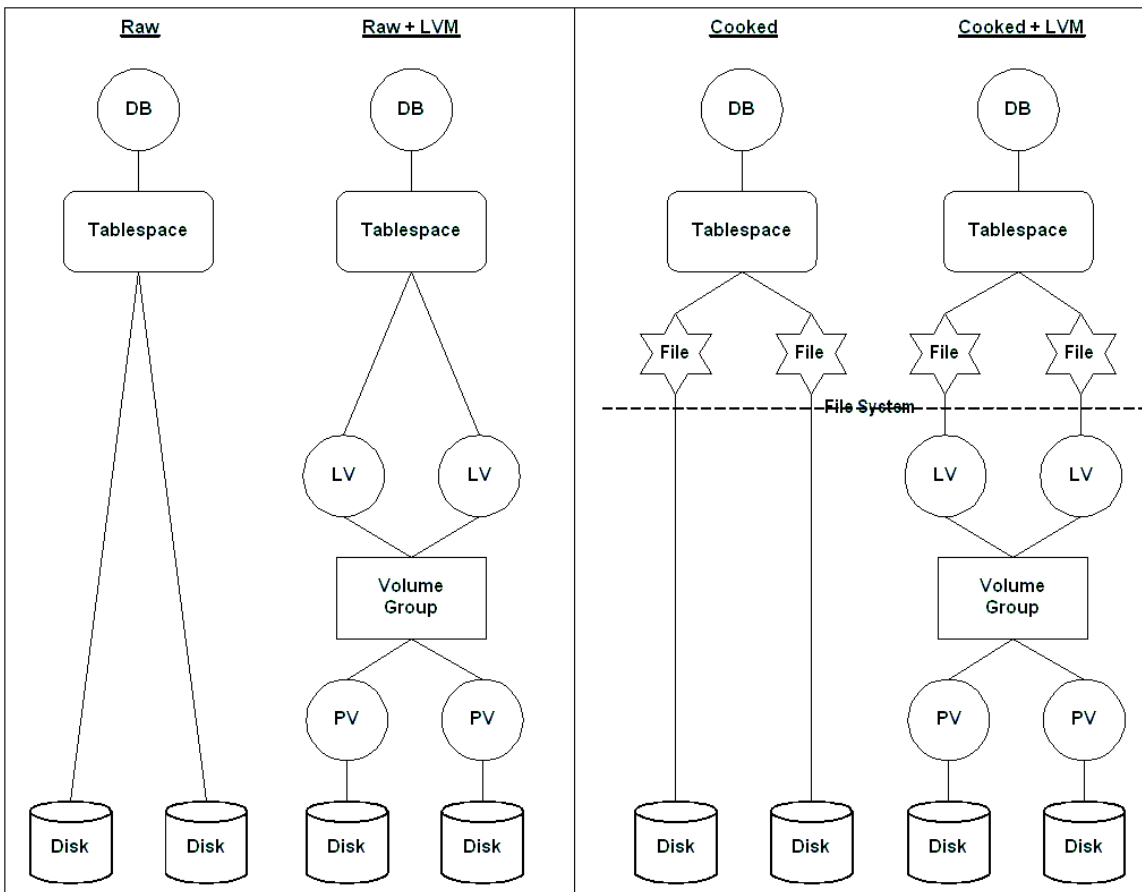


Figure 1

Figure 1 shows that there were essentially two choices to be made: raw versus cooked and with or without a LVM. These pictures could further be complicated since what's labeled a disk might in fact be a pseudo-disk. For example in an EMC disk array, disks are referred to as spindles and can be sub-divided into hyper disks. And in Linux, what we label as a disk is simply the device reference – which might just be a partition of an actual disk. To keep things manageable and thus discussable, let's assume that a disk is not to be subdivided. So each disk in the diagram means a separate physical disk drive.

Returning to our original challenge (ever bigger databases with lots and lots of disks), we often had to choose an option from **Figure 1** that more easily and best scales to hundreds or even thousands of disks. With such large numbers of disks, the general consensus is to use a LVM. And while on some operating systems raw devices are an attractive choice, I've found that raw devices on Linux are not the best way to go (for complete details, see my prior article "[Tuning an Oracle8i Database running Linux](#)"). Plus of course there are the extra management headaches associated with using raw devices. So we can conclude that a cooked file system with a LVM (right most example) is really our best choice – and therefore the scenario we'll compare against Oracle 10g's ASM.

THE NEW METHODS

Now let's look at Automated Storage Management in more detail. **Figure 2** shows both 9i's Oracle Managed Files (OMF) and 10g's ASM. Like many Oracle powerful features, there were incremental steps in delivering ASM – namely 9i's OMF, although we did not know it at the time. Hence why I show these two technologies side by side.

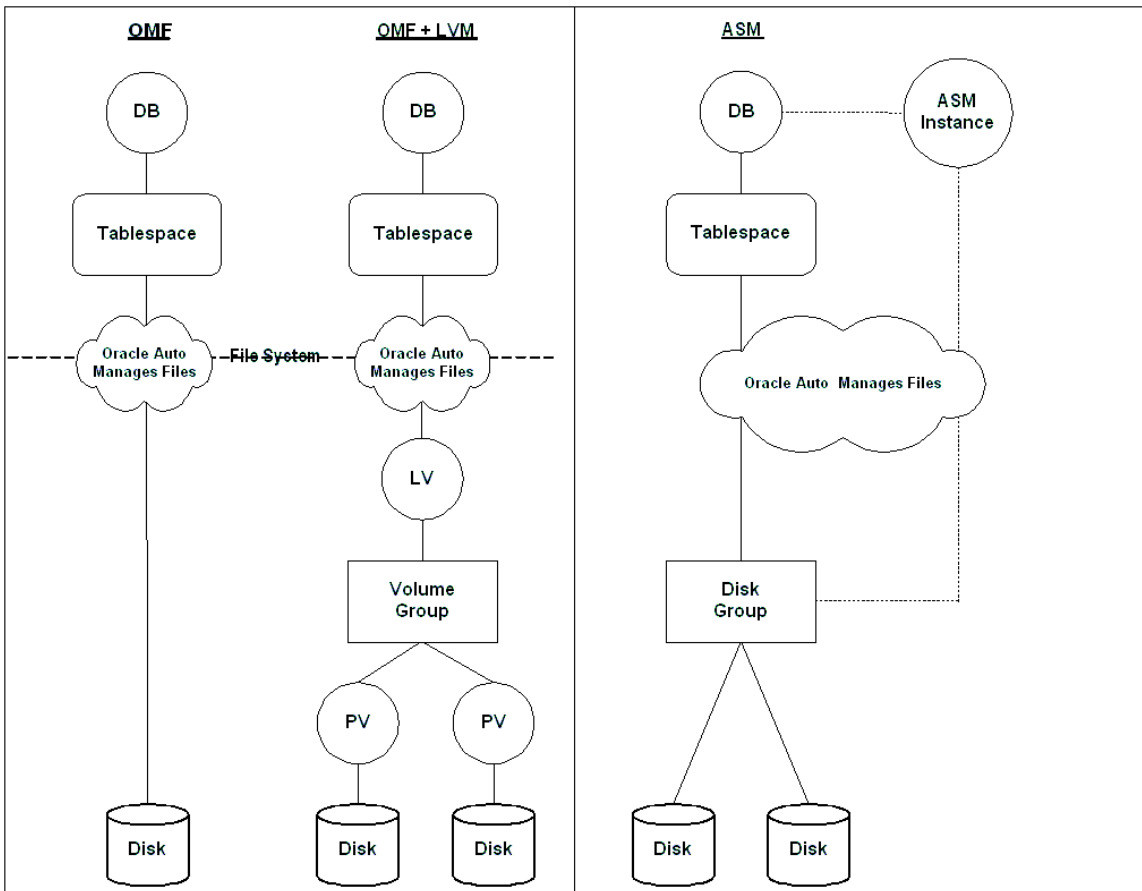


Figure 2

In many respects, OMF was a first step at internalizing within the Oracle database engine the concept of automatic file management. However even as just an incremental offering, it had some real usefulness. For those DBA's abstracting disk complexity away from the database (i.e. the black box extreme scenario from my introduction above), they merely had to provide Oracle with a single file destination – via the DB_CREATE_FILE_DEST parameter. That parameter value can only represent just a single file system based storage location – although it could be either a single pseudo-disk representing multiple physical disks hidden behind hardware RAID or a single logical volume combining numerous physical disks. Either way, Oracle 9i automatically manages all the file system files. The main disadvantages (besides the obvious single storage location) are that it does not work with raw devices and still requires using native operating system based file systems. Even the Oracle manuals suggest that OMF is limited as follows:

- Databases that are supported by the following:
 - A logical volume manager that supports striping/RAID and dynamically extensible logical volumes
 - A file system that provides large, extensible files
- **Low end or test databases** (*which I think kind of makes a statement in itself*)

Now look again at **Figure 2** and concentrate on ASM (right most example). There is no need for either a Logical Volume Manager or file systems. There are simply disk groups under ASM management and available for tablespace consumption. And while the file management is again automatic as with OMF, it is now 100% internal. There are no file systems to mount via the /etc/fstab file and no files to access using operating system file commands like ls, cp, tar, etc. You now query the database to see all such information.

Furthermore, ASM can stripe or mirror your data at the file level – versus by disk with other methods. And Oracle automatically manages keeping everything fully mirrored or striped for you whenever disks within the disk groups are added, dropped, or fail – and all with the database completely online. Oracle accomplishes this via new and highly efficient methods that are radically different than the simple arithmetic algorithms in use by many Logical Volume Managers. And although you might make the point that there's an additional ASM Oracle instance required, the LVM versus ASM example developed below will show it's tiny, very easy to setup and permits 100% effective, efficient and automatic storage management – and can do so for all your database instances. So ASM will be shown to be easier to create and manage, as well as providing all the performance expected. You can't lose with ASM.

EASIER TO SETUP

ASM is just plain easier to setup than its file system and LVM counterpart. Let's assume we have a simple database to create – just the required tablespaces and one for user data. Let's also assume the following hardware and software constraints:

- RAID 0 – stripe everything across all drives
 - Stripe Width = 4
 - Stripe Length = 64 K
- Four 20 GB IDE disks – each with single partition
- Linux ext3 file system (2 GB file size limit)
- Five Tablespaces
 - SYSTEM 2 GB 1 data files
 - SYSAUX 2 GB 1 data files
 - UNDO 8 GB 4 data files
 - TEMP 8 GB 4 data files
 - USER 60 GB 30 data files
- One Physical Volume (PV) per disk drive
- One Volume Group (VG) – VG01
- Four Logical Volumes (LV)
 - LV01 4 GB SYSTEM and SYSAUX
 - LV02 8 GB UNDO
 - LV03 8 GB TEMP
 - LV04 60 GB USER

Figure 3 shows the relative complexities of each environment. And remember, this was a simple case for just four disk drives. Now imagine doing this with thousands of drives!

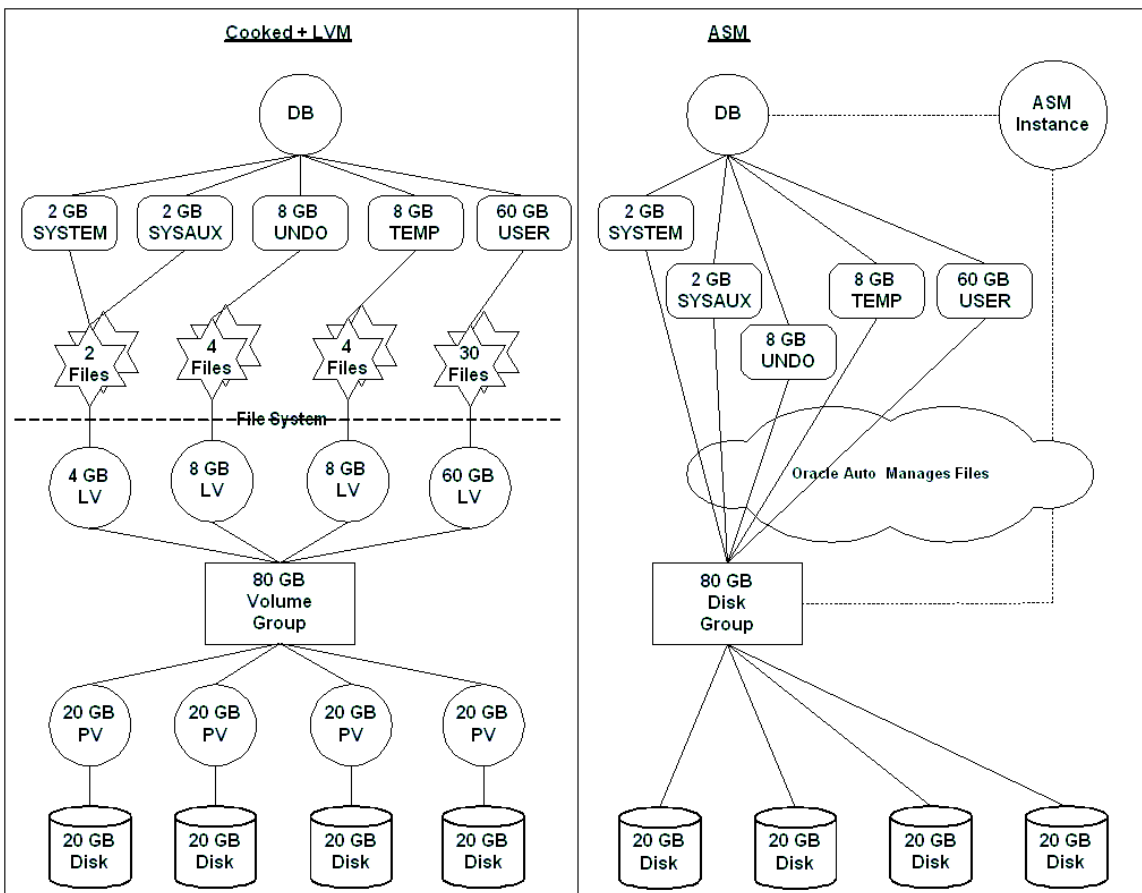


Figure 3

So now let's compare the actual steps to create the database in each environment. Note that in the LVM example I am purposefully avoiding the handling of various overhead issues with all size settings in order to keep the example simple. But be warned that for optimal disk space usage, you'll have to tackle this issue as well. It's yet another example of complex storage planning and management issues that you can avoid by using ASM.

Here are the steps using cooked files with a LVM (note that the IDE disk drives are assumed to be the 2nd through 5th drives in the example, hence the b-e device name designations of /dev/hdb through /dev/hde):

1. `fdisk /dev/hdb` set its type to 0x8e (LVM partition)
2. `fdisk /dev/hdc` set its type to 0x8e (LVM partition)
3. `fdisk /dev/hdd` set its type to 0x8e (LVM partition)
4. `fdisk /dev/hde` set its type to 0x8e (LVM partition)

5. pvcreate /dev/hdb /dev/hdc /dev/hdd /dev/hde
6. vgcreate VG01 /dev/hdb /dev/hdc /dev/hdd /dev/hde
7. lvcreate -L 4 G -i 4 -l 64 -n LV01 VG01
8. lvcreate -L 8 G -i 4 -l 64 -n LV02 VG01
9. lvcreate -L 8 G -i 4 -l 64 -n LV03 VG01
10. lvcreate -L 60 G -i 4 -l 64 -n LV04 VG01
11. mkfs -t ext3 /dev/VG01/LV01
12. mkfs -t ext3 /dev/VG01/LV02
13. mkfs -t ext3 /dev/VG01/LV03
14. mkfs -t ext3 /dev/VG01/LV04
15. mount /dev/VG01/LV01 /home/oracle/oradata/LVMDB/**system**
16. mount /dev/VG01/LV02 /home/oracle/oradata/LVMDB/**undo**
17. mount /dev/VG01/LV03 /home/oracle/oradata/LVMDB/**temp**
18. mount /dev/VG01/LV04 /home/oracle/oradata/LVMDB/**user1**
19. edit /etc/fstab and add the new mount point entries
20. Create initLVMDB.ora file
 - INSTANCE_TYPE = RDBMS
21. SQL Plus connect as SYSDBA for SID=LVMDB
22. STARTUP NOMOUNT PFILE=initLVMDB.ora
23. CREATE SPFILE FROM PFILE=initLVM.ora
24. Create Oracle database and user tablespace using SQL code below

```

create database LVMDB
controlfile reuse
logfile          '/home/oracle/oradata/LVMDB/redo_log01.dbf' size 16M,
                 '/home/oracle/oradata/LVMDB/redo_log02.dbf' size 16M
datafile         '/home/oracle/oradata/LVMDB/system/system01.dbf' size 2 G
sysaux datafile '/home/oracle/oradata/LVMDB/system/sysaux01.dbf' size 2 G
default temporary tablespace temp
tempfile         '/home/oracle/oradata/LVMDB/temp/temp01.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/temp/temp02.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/temp/temp03.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/temp/temp04.dbf' size 2 G
extent management local uniform size 64k
undo tablespace undo
datafile         '/home/oracle/oradata/LVMDB/undo/undo01.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/undo/undo02.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/undo/undo03.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/undo/undo04.dbf' size 2 G;

```

```

create tablespace USER
datafile          '/home/oracle/oradata/LVMDB/user1/user01.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user02.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user03.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user04.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user05.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user06.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user07.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user08.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user09.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user10.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user11.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user12.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user13.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user14.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user15.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user16.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user17.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user18.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user19.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user20.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user21.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user22.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user23.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user24.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user25.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user26.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user27.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user28.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user29.dbf' size 2 G,
                 '/home/oracle/oradata/LVMDB/user1/user30.dbf' size 2 G
extent management local uniform size 64k;

```

Now simply imagine that you'll need dozens of such user tablespaces, each with lots and lots of data files. It's not hard to see why this example does not scale well. Keeping track of all the tablespaces, data files, logical volumes, physical volumes and actual disk drives (for finding hot spots) is just too darn hard for anyone but superman.

Now here's the much simpler ASM example:

1. Create initASM.ora file
 - INSTANCE_TYPE = OSM
2. SQL Plus connect as SYSDBA for SID=ASM
3. STARTUP NOMOUNT PFILE=initASM.ora
4. CREATE SPFILE FROM PFILE=initASM.ora

5. CREATE DISKGROUP dgroup1 EXTERNAL REDUNDANCY DISK
'/dev/hdb','/dev/hdc','/dev/hdd','/dev/hde'
6. Create initASMDB.ora file
7. INSTANCE_TYPE = RDBMS
8. DB_CREATE_FILE_DEST = '+dgroup1'
9. SQL Plus connect as SYSDBA for SID=ASMDB
10. STARTUP NOMOUNT PFILE=initASMDB.ora
11. Create Oracle database and user tablespace using SQL code below

```

create database ASMDB
controlfile reuse
logfile          '+dgroup1' size 16 M
datafile         '+dgroup1' size 2 G
sysaux datafile '+dgroup1' size 2 G
default temporary tablespace temp
tempfile        '+dgroup1' size 8 G
undo tablespace undo
datafile        '+dgroup1' size 8 G;

create tablespace USER_LOCAL
datafile        '+dgroup1' size 60 G;

```

And that's the syntax if we still want to explicitly control the data allocation sizes (which really is no longer a concern as we're working now at the disk level). So we really could just let Oracle handle all the internal space needs and issue the much simpler syntax of

```
create database ASMDB;
```

Not only is this much shorter and thus easier to read – but note that we even got our redo logs created in our striped disk group (in the prior example they had just been placed on the file system). The main point is that the process of laying out the storage and creating the database is just so much simpler that if you're managing lots and lots of disk drives (i.e. SAN or NAS) that you can't go wrong with upgrading to Oracle 10g for ASM alone.

EASIER TO CHANGE

Of course you only create the database once – so you might argue that the savings shown above are not reason enough to undertake any changes. So now let's examine what would happen in each scenario if we add four disks. This is where ASM really shines through.

Let's assume that our single USER tablespace is nearly full containing just 10 tables and 10 indexes, where each table consumes 4 GB and each index consumes 2 GB. If we now need to create another table and index, we don't have enough room. So we are given four more disks identical to the first four to add to our storage design in order to accommodate additional space requests. In other words, we're going to add 80 GB to our single USER tablespace. Seems easy enough, right?

In the LVM example, we have three primary options:

- Create a new Volume Group VG02 with a new Logical Volume LV05
- Extend existing Volume Group VG01 with a new Logical Volume LV05
- Extend existing Volume Group VG01 by extending Logical Volume LV04



since we're assuming that all 80 GB is to be dedicated to the USER tablespace, there is no need to create more than a single new Logical Volume for first two options.

Most people will choose the third option, since we're merely trying to add space to our existing storage design. So here are the steps to implement that choice:

1. `fdisk /dev/hdf` set its type to 0x8e (LVM partition)
2. `fdisk /dev/hdg` set its type to 0x8e (LVM partition)
3. `fdisk /dev/hdh` set its type to 0x8e (LVM partition)
4. `fdisk /dev/hdi` set its type to 0x8e (LVM partition)
5. `pvcreate /dev/hdf /dev/hdg /dev/hdh /dev/hdi`
6. `vgextend VG01 /dev/hdf /dev/hdg /dev/hdh /dev/hdi`
7. `lvextend -L +80 G /dev/VG01/LV04`
8. `ext2online /dev/VG01/LV04`
9. SQL Plus connect as SYSDBA for SID=LVMDB
10. Add new space to the tablespace using SQL code below

```

alter tablespace USER
add datafile      '/home/oracle/oradata/LVMDB/user2/user01.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user02.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user03.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user04.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user05.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user06.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user07.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user08.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user09.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user10.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user11.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user12.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user13.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user14.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user15.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user16.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user17.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user18.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user19.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user20.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user21.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user22.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user23.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user24.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user25.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user26.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user27.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user28.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user29.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user30.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user31.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user32.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user33.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user34.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user35.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user36.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user37.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user38.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user39.dbf' size 2 G,
                  '/home/oracle/oradata/LVMDB/user2/user40.dbf' size 2 G;

```

Not only is this example very long, it has several problems as well. First, online file system resizing is very tricky business. The author of the ext2online utility states that “resizing a mounted file system is inherently dangerous and may corrupt file systems”. And the ext2online utility can only enlarge a mounted file system, to shrink or enlarge file systems one would use ext2resize instead – which only works on un-mounted file systems. Of course that would require taking the tablespace offline.

The LVM approach has some not so obvious drawbacks as well. We very likely expected that this solution would result in our data being striped across all eight of our drives – not true. While we can add space to a Logical Volume, we cannot change its striping nature on Linux (although some UNIX platform’s LVM do provide such capabilities). So our 10 old tables and 10 old indexes are striped across drives b-e, while our new table and index are striped across drives f-i (since the USER tablespace was already full, new objects will be created in the new space). Even if we exported the tablespace objects, dropped them, coalesced the tablespace, and then imported them back into the tablespace – the Logical Volume is still set for four way striping. We’d have to manually do the following if we really wanted eight way striping:

1. Export the objects in that tablespace (database in restricted session to be safe)
2. Drop the tablespace
3. Drop the Logical Volume
4. Create a new Logical Volume (with striping parameter set as –i 8)
5. Create the tablespace (this would have lots of data file lines for all 140 GB)
6. Import the objects into the tablespace

That’s where ASM steps in and makes life easy. Not only are the steps shorter:

1. SQL Plus connect as SYSDBA for SID=ASM
2. ALTER DISKGROUP dgroup1 ADD DISK
'/dev/hdf','/dev/hdg','/dev/hdh','/dev/hdi'

That’s it! But there’s more. ASM automatically rebalances both its striping and mirroring of a disk group whenever disks are added, dropped, or fail – and all with the database completely online. Therefore Oracle automatically takes care of keeping all of your objects fully striped. That’s why ASM can make the claim that it provides near optimal IO balancing without any manual tuning. It simply internalizes and automates that which DBA’s have been doing manually for years – trying to eliminate hot spots by spreading things across as many drives as possible. Note that you can control when and how Oracle performs that rebalancing via the OSM_POWER_LIMIT and other parameters (but that deserves an article on its own).

SOME PERFORMANCE TOO ...

Oracle 10g's mantra seems to be simpler and more automated database management. All the above exemplifies for some people why ASM by itself might be worth an upgrade to Oracle 10g. But this article would be incomplete without some examples comparing the relative performance of the LVM versus ASM scenarios. While we are eliminating the Logical Volume Manager and file system, we are still nonetheless utilizing more Oracle technology to do essentially much of the same thing. So initial expectations are to see fairly similar or slightly better performance – with the advantages once again being far greater simplicity to create and manage.

Here are some relative results for consideration (note – tested using 10g beta #1):

	ASM VS. LVM
Populate an 80 GB Database	11% faster
Build Indexes for the Database	9% faster
200 User Concurrent Access (OLTP)	5% faster

While these results are not earth shattering, roughly 10% improvements from something that makes the DBA's life easier is not a bad return on investment for the cost of doing an upgrade. And I'm optimistic that when Oracle 10g releases as production ready – those numbers might improve. Of course that's probably just the "Jerk" in me getting overly excited once again ☺

Bert Scalzo (Bert.Scalzo@Quest.com) is a product architect for Quest Software, which has been very active in the Oracle 10g beta program. Bert has a B.S., M.S., and Ph.D. in computer science, has worked as an Oracle DBA for over 18 years, and designed much of the DBA module for TOAD. Bert has also written two books: "Oracle DBA Guide to Data Warehousing and Star Schemas" (Prentice Hall) and "TOAD Handbook" (Sam's Publishing).

ABOUT THE AUTHOR

Bert Scalzo (Bert.Scalzo@Quest.com) is a product architect for Quest Software, which has been very active in the Oracle 10g beta program. Bert has a B.S., M.S., and Ph.D. in computer science, has worked as an Oracle DBA for over 18 years, and designed much of the DBA module for TOAD. Bert has also written two books: “Oracle DBA Guide to Data Warehousing and Star Schemas” (Prentice Hall) and “TOAD Handbook” (Sam's Publishing).

ABOUT QUEST SOFTWARE, INC.

Quest Software, Inc. delivers innovative products that help organizations get more performance and productivity from their applications, databases and infrastructure. Through a deep expertise in IT operations and a continued focus on what works best, Quest helps more than 18,000 customers worldwide meet higher expectations for enterprise IT. Quest Software, headquartered in Irvine, Calif., can be found in offices around the globe and at www.quest.com.

Contacting Quest Software

Mail:	Quest Software, Inc. World Headquarters 8001 Irvine Center Drive Irvine, CA 92618 USA
Web site	www.quest.com
Email:	info@quest.com
Phones:	1.800.306.9329 (Inside U.S.) 1.949.754.8000 (Outside U.S.)

Please refer to our Web site for regional and international office information. For more information on Quest Software solutions, visit www.quest.com.

Trademarks

All trademarks and registered trademarks used in this guide are property of their respective owners.