

**Oracle Database10<sup>g</sup>**  
**Automatic Storage Management**  
**Technical Best Practices**

*Nitin Vengurlekar*  
*Oracle Corporation*  
*January 2004*

## Table of Contents

|  |    |
|--|----|
| Introduction.....  | 3  |
| Introduction.....  | 3  |
| ASM Instances .....                                      | 5  |
| Disks .....  | 7  |
| DiskGroups .....   | 10 |
| Failure Groups .....                                     | 12 |
| CSS .....  | 12 |
| Database Instances .....                                 | 13 |
| Storage Management and Allocation.....                   | 16 |
| Rebalance and Redistribution .....                       | 17 |
| Files and Aliases .....                                  | 19 |
| Templates.....   | 22 |
| Conclusion .....   | 23 |
| Appendix A – Partitions, Views and Discover Strings..... | 24 |
| Appendix B- Migrating to ASM.....                        | 25 |

## Introduction

In Oracle10<sup>g</sup>, storage management and provisioning for the database has become much more simplified with a new feature called *Automatic Storage Management (ASM)*. ASM provides filesystem and volume manager capabilities built into the Oracle database kernel. With this capability, ASM simplifies storage management tasks, such as creating/laying out databases and disk space management. Since ASM allows disk management to be done using familiar create/alter/drop SQL statements, DBAs do not need to learn a new skillset or make crucial decisions on provisioning. Additionally, ASM operations can be completely managed with 10<sup>g</sup> Enterprise Manager.

Why is ASM important? ASM not only empowers the DBA to lower the cost of storage management, but also provides high performance capabilities and utilization.

ASM is a management tool specifically built to simplify the job of the DBA. It provides a simple storage management interface across all server and storage platforms. ASM provides the DBA flexibility to manage a dynamic database environment with increased efficiency. This feature is a key aspect of Grid Computing.

The following are some key benefits of ASM:

- I/O is spread evenly across all available disk drives to prevent hot spots and maximize performance.
- Inherent large file support.
- Performs automatic online redistribution after the incremental addition or removal of storage capacity.
- Maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor supplied reliable storage mechanisms.
- Provides database storage management for single SMP machines, or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.
- Leverage redundancy from intelligent storage arrays.
- For simplicity and easier migration to ASM, 10<sup>g</sup> will allow the co-existence of ASM and non-ASM files. Any new files can be created as ASM files whilst existing files can also be migrated to ASM.
- New RMAN commands enable non-ASM managed files to be relocated to an ASM disk group.
- 10<sup>g</sup> Enterprise Manager can manage most ASM disk and file management activities

This document will discuss the essentials of ASM and will cover the steps to add disks, create a diskgroup, and eventually create a database within ASM, discussing some best practices along the way. There will also be some comparisons with Veritas Volume Manager (VxVM), made merely for illustration.

The following basic components of ASM will be discussed:

- Instances – ASM instances and database instances
- Diskgroups
- Failure Groups
- ASM Disks
- Rebalancing Capabilities
- ASM Files, Templates and Aliases

Although CSS is not explicitly listed here, since it is used for process-to-process communication and is therefore required to run ASM. As such, it is discussed in this document for completeness.

## ASM Instances

In 10<sup>g</sup> there are two types of instances: database and ASM instances. The database instance will be discussed in a later section. The ASM instance, which is generally named +ASM, is started with the `INSTANCE_TYPE=ASM` `init.ora` parameter. This parameter, when set, signals the Oracle initialization routine to start an ASM instance and not a standard database instance. Unlike the standard database instance, the ASM instance contains no physical files; such as logfiles, controlfiles or datafiles, and only requires a few `init.ora` parameters for startup<sup>1</sup>. Upon startup, an ASM instance will spawn all the basic background processes, plus some new ones that are specific to the operation of ASM. An ASM instance can be started and shutdown just as any instance; however, ASM cannot mount or open<sup>2</sup> a database. The illustration in Figure 1 shows an initialized ASM instance.

ASM is the file and storage manager for all databases [that employ ASM] on a given node, similar to a Veritas volume manager. Therefore, only one ASM instance is required per node regardless of the number of database instances on the node. Additionally, ASM seamlessly works with the RAC architecture to support clustered storage environments. In RAC environments, there will be one ASM instance per clustered node, and the ASM instances will communicate with each other on a peer-to-peer basis.

Enabling the ASM instance is as simple as configuring a handful of `init.ora` parameters. The `init.ora` parameters specified in Figure 1 are the essential parameters required to start up ASM.

Access to the ASM instance is comparable to a standard instance; i.e., `SYSDBA` and `SYSOPER`. Note however, since there is no data dictionary, authentication is done from an Operating System level and/or an Oracle password file.

---

<sup>1</sup> The ASM instance only requires approximately 100Mb memory footprint.

<sup>2</sup> However, at startup ASM instance does mount disk groups specified by the `asm_diskgroups` parameter.

Figure 1.

```

Instance type
SQL> select instance_name from v$instance

INSTANCE_NAME
-----
+ASM

ASM init.ora parameters
*.background_dump_dest='/u06_10i/app/admin/+ASM/bdump'
*.core_dump_dest='/u06_10i/app/admin/+ASM/cdump'
*.instance_type='asm'
*.asm_diskgroups='+DGR14_A'
*.large_pool_size=12M
*.asm_diskstring='/dev/rdisk/c3t19d*s4'
*.remote_login_passwordfile='SHARED'
*.user_dump_dest='/u06_10i/app/admin/+ASM/udump'

ASM background processes3
oracle 1421      1 0 Sep 22 ?      0:00 asm_dbw0_+ASM
oracle 1423      1 0 Sep 22 ?      0:00 asm_lgwr_+ASM
oracle 1419      1 0 Sep 22 ?      0:00 asm_mman_+ASM
oracle 1417      1 0 Sep 22 ?      0:00 asm_pmon_+ASM
oracle 1427      1 0 Sep 22 ?      0:00 asm_smon_+ASM
oracle 1582      1 0 Sep 23 ?      0:00 asm_asmb_+ASM
oracle 1425      1 0 Sep 22 ?      0:00 asm_ckpt_+ASM
oracle 1429      1 0 Sep 22 ?      0:01 asm_rbal_+ASM
oracle 1584      1 0 Sep 23 ?      0:00 oracle+ASM
      (DESCRIPTION=(LOCAL=Y))
oracle 1615      1 0 Sep 23 ?      0:00 oracle+ASM
      (DESCRIPTION=(LOCAL=Y))
oracle 1586      1 0 Sep 23 ?      0:00 asm_o000_+ASM

```

Similarly, Veritas VxVM requires the following processes to maintain the disk configuration and metadata.

```

oracle@dal-ors4[/dev/rdisk]ps -ec |grep vx
  19  TS  59 ?      0:10 vxconfig
 860  TS  20 ?      0:00 vxrelocd
 894  TS  53 ?      0:00 vxnotify
 892  TS   0 ?      0:00 vxrelocd

```

<sup>3</sup> If an ASM instance fails, all Oracle database instances dependent on that ASM instance also fail. This is similar to a failure of a volume manager in the OS.

## Disks

Since ASM does not contain any datafiles, all metadata is stored on the ASM disks and in the ASM instance SGA.

The first task in building the ASM infrastructure is to discover and associate (adding) disks under ASM management. This step is best done with some coordination of the Storage Sysadmin. The Storage administrator will identify a set of disks from the storage array that will be presented to the host. The term *disk* may be used in loose terms. A disk can be partition of a physical spindle or refer to the entire spindle itself, this depends on how the storage array presents the logical unit number (LUN) to the OS. In this document we will refer generically to LUNs or disks presented to the OS as simply, *disks*.

On Solaris systems, disks will generally have the following SCSI name formats: *CwTxDySz*. Where the C is controller number; T is the target, D is the disk number, and S is the partition. Note, each OS will have its unique representation of disk naming.

On Solaris systems, for best practices, it is best to choose the s4 partition of disk, or a partition that skips the first 1Mb of the disk. This is done to skip the OS label/VTOC. See the Appendix A for a sample Sun Solaris partition table. Different Operating Systems will have varying requirements for the OS label; i.e., some may require an OS label before its used while others will not.

In SAN environments, it assumed that the disks identified are appropriately zoned and LUN masked. Once the disks are identified, they will need to be discovered by ASM. This requires that the disk devices (Unix filenames) have their ownership changed from root to oracle. These candidate disks must already have a valid label on it, and should not be managed (encapsulated) by any other LVM, such as Veritas.

In our example, disks c3t19d5s4, c3t19d16s4, c3t19d17s4, c3t19d18s4 were identified, and their ownership changed.

Now, these disks can be defined in the init.ora parameter, asm\_diskstring. In our example we used the following wildcard setting:

```
*.asm_diskstring='/dev/rdisk/c3t19d*s4'.
```

This will invoke ASM to scan all the disks with that string, and find any devices that it has permissions to open. Upon successful discovery, the V\$ASM\_DISK view will now show which disks were discovered.

Similarly, the Veritas `vxdisk` command provides the following information:

```
# vxdisk list | more
DEVICE      TYPE      DISK      GROUP      STATUS
c1t4d0s2    sliced    disk04     rootdg     online
c1t5d0s2    sliced    disk03     rootdg     online
c3t19d6s2    sliced    odm_vg04   odm_vg     online
c3t19d7s2    sliced    11i_ora01  11i_oracle_home_vg online
c3t19d8s2    sliced    arch_re01  arch_redo  online
c3t19d11s2   sliced    11i_ora02  11i_oracle_home_vg online
c3t19d12s2   sliced    11i_db_01  11i_db_vg  online
```

ASM, like Veritas, will generate and assign a disk name with a sequence number to each disk identified. The ASM disk name is different from the SCSI address. This allows for consistent naming across RAC nodes as well as protects from SCSI address name changes due to array re-configurations. The Disk Name is used when performing any disk management activities.

The `vxdiskadd` command line (and menu) utility can be used to configure disks. The `vxdiskadd` command will prompt the user for a disk group name and disk name for the disks. If no disk group name specified, the disks will be left as unassigned. For example, the following will place all disks with the stated string under VxVM control.

```
vxdiskadd c3t19d*s4
```

Since disk drives are mechanical devices, they have a tendency to fail. As drives begin to fail or have sporadic IO errors, the probability for database corruptions becomes more likely. ASM takes proactive measures with regards to IO errors. A permanent IO error is only signaled to the caller (Oracle IO process) after several retries in the device driver. If a permanent disk IO error is incurred during an Oracle write operation, then the affected disk is removed from the diskgroup by ASM, thus preventing more application failures<sup>4</sup>. If the loss of a disk results in data loss, ASM will automatically dismount the diskgroup to protect the integrity of the diskgroup data.

Although ASM does not maintain any database data, it does maintain metadata that is very important to the operation of the diskgroup. This metadata, as stated earlier, is kept in the ASM SGA as well as on the ASM disks. The metadata is kept within the disk group itself so that the disk group is self-describing.

---

<sup>4</sup> Disks are removed from the diskgroup only on write operation I/O error and not for read operations. For read errors, the block is read from the secondary (only for normal or high redundancy; for external redundancy)



The two important metadata structures are the Continuing Operation Directory (COD) and the Active Change Directory (ACD). The COD and ACD both house a log like structure of active structural changes to the diskgroup and metadata changes; respectively. The COD structure maintains the state of active ASM operations or changes, such as disk or datafile drop/add. For example, if a new disk is added to a diskgroup, a COD record is created to reflect this configuration change. The COD log record is either committed or rolled back based on the success of the operation. A successful operation adds the disk to the diskgroup. A failed operation rolls back the operation and returns an error message, preventing a dangling disk association.

The ACD is analogous to a redo log, where changes to the metadata are logged. The ACD log record is used to determine point of recovery in the case of ASM operation failures or instance failures. For example, if ASM fails during a rebalance operation, then upon restart of ASM, the metadata will reflect an influx operation, and then use ACD and COD to resume to unfinished operation.

Note, there is no need to backup the ASM, since there are no physical ASM files. The ASM metadata is persistently kept in the disk group and triple mirrored, thus, inherently providing high availability.

## DiskGroups

Once the disks are discovered and readied, a disk group can be created that will encapsulate one or more of these drives. A disk group, which is the highest-level data structure in ASM, is comparable to a LVM's volume group or a storage group. However, there are several other differentiators between typical LVM volume groups and ASM diskgroups:

- The ASM filesystem layer (which transparently sits atop the diskgroup) is not visible to users; i.e., The ASM logical volume / filesystem is a raw interface seen only by ASM and the interfacing database instance (more on that later).
- There are inherent automatic file-level striping and mirroring capabilities. A database file created within an ASM diskgroup will have its file extents (not to be confused with its database extents) distributed equally across all online disks in the diskgroup, providing an even IO load.

The disk group is created using SQL commands. See Figures 2 and 3 below.

Figure 2.

|   |         |        |          |         |
|---|---------|--------|----------|---------|
| SQL> create diskgroup DGR14_A external redundancy disk<br>'/dev/rdisk/c3t19d5s4', '/dev/rdisk/c3t19d16s4', '/dev/rdisk/c3t19d17s4',<br>'/dev/rdisk/c3t19d18s4'; |         |        |          |         |
| The output below shows the newly created diskgroup  |         |        |          |         |
| SQL> select name, state, type, total_mb, free_mb from v\$asm_diskgroup;   |         |        |          |         |
| NAME  | STATE   | TYPE   | TOTAL_MB | FREE_MB |
| -----   | -----   | -----  | -----    | -----   |
| DGR14_A   | MOUNTED | EXTERN | 34512    | 33479   |

The creation of a diskgroup involves the validation of the disks to be added. These disks cannot already be in use by another diskgroup, must not have a pre-existing ASM header, and cannot have an Oracle file header.

The successful creation of the diskgroup will create a valid ASM disk header on each disk, and the V\$ASM\_DISK view will indicate this with the online and normal indicators, for mode status and state, respectively. Once these disks are under ASM management, all subsequent mounts of the diskgroup will cause ASM to re-read and validate the ASM disk headers.

Figure 3.

| SQL> select name, path, mode_status, state, disk_number from v\$asm_disk |                       |         |        |             |
|--|-----------------------|---------|--------|-------------|
| NAME   | PATH                  | MODE_ST | STATE  | DISK_NUMBER |
| DGR14_A_0000   | /dev/rdisk/c3t19d5s4  | ONLINE  | NORMAL | 0           |
| DGR14_A_0001   | /dev/rdisk/c3t19d16s4 | ONLINE  | NORMAL | 1           |
| DGR14_A_0002   | /dev/rdisk/c3t19d17s4 | ONLINE  | NORMAL | 2           |
| DGR14_A_0003   | /dev/rdisk/c3t19d18s4 | ONLINE  | NORMAL | 3           |

In the example above, a DGR14\_A diskgroup is created using four disks, which reside in the EMC Storage Array, and the redundancy is handled externally via EMC mirrored sets. After the disk group is created, the metadata information, which includes creation date, diskgroup name, and redundancy type, is stored in the SGA and on each disk (in disk header) within the DGR14\_A diskgroup. When a diskgroup becomes mounted, ASM registers, the diskgroup name, the instance name, and the corresponding Oracle Home path name with Cluster Synchronization Services (CSS). This registered data is then used by the database instance to build the TNS connect string. This string is subsequently used by the database instance to connect into the ASM instance for volume management activities. This is discussed in the Database Instance Section.

A diskgroup can contain files from many different 10<sup>g</sup> Oracle databases. Alternatively, one Oracle database may also store its files in multiple disk groups managed by the same ASM instance. However, a disk and a database file can only be part of one diskgroup.

For manageability best practices, we're recommending 3 separate, independent storage areas for software, active database files, and recovery-related files.

- Software area: A location where software is installed and log and trace files are created. This must be in a filesystem created outside of a ASM diskgroup.
- Database area: A location where active database files such as datafiles, control files, online redo logs, and change tracking files used in incremental backups are stored
- Flash recovery area: A location where recovery-related files are created, such as multiplexed copies of the current control file and online redo logs, archived redo logs, backup sets, and flashback log files

Additionally, when mounting diskgroups, either at ASM startup or for subsequent mounts, it is advisable to mount all required diskgroups at once. This is done to minimize the overhead of multiple ASM discovery scans.

## Failure Groups

For systems that do not have or cannot use external redundancy, ASM provides its internal redundancy mechanism using Failure Groups. A Failure group by definition is a collection of disks that can become unavailable due to a failure of one of its associated components; e.g., disks or controllers. Thus, disks between two failure groups (for a given diskgroup) must be separated/isolated from a common failure component.

Redundancy for disk groups can be either *Normal redundancy* (the default); where files are two-way mirrored (requiring at least two failure groups), or *High redundancy*, which provides a higher degree of protection using three-way mirroring (requiring at least three failure groups).

In Figure 4 below is an example of creating a disk group using failure groups.

```
SQL> create diskgroup DBR15_B normal redundancy

Failure group flgrp1 disk

'/dev/rdisk/c3t19d3s4', '/dev/rdisk/c3t19d4s4', '/dev/rdisk/c3t19d5s4',
'/dev/rdisk/c3t19ds4'

Failure group flgrp2 disk

'/dev/rdisk/c4t19d3s4', '/dev/rdisk/c4t19d4s4', '/dev/rdisk/c4t19d5s4',
'/dev/rdisk/c4t19ds4' ;
```

## CSS

ASM was designed to work with single instance as well as with RAC clusters. ASM, even in single-instance, requires that Cluster Synchronization Services (CSS) is installed and available. In a single instance CSS maintains synchronization between the ASM and database instances. CSS, which is a component of Oracle's Cluster Ready Services (CRS), is automatically installed on every node that runs Oracle Database 10<sup>g</sup>. However, in RAC environments, the full cluster-ware (CRS) is installed on every RAC node.

Since CSS provides cluster management and node monitor management, it inherently monitors ASM and its shared storage components (disks and diskgroups). Upon startup, ASM will register itself and all diskgroups it has mounted, with CSS. This allows CSS across all RAC nodes to keep diskgroup metadata in-sync. Any new diskgroups that are created are also dynamically registered and broadcasted to other nodes in the cluster.

Database instances contact the CSS to lookup the TNS connect string (using the diskgroup name as an index). This connect string is then used to create a persistent connection into the ASM instance.

## Database Instances

A Database instance is the standard Oracle instance, and is the client of the ASM instance.

After the ASM disk group is created, DBCA can be used to create the database<sup>5</sup>. DBCA now has three options for database file structures: filesystem, raw, or ASM. If ASM is selected, then all available diskgroups for that ASM instance are listed. Selecting ASM and a diskgroup, will invoke DBCA to create a database within ASM. If no diskgroups exist, then DBCA offers the opportunity to create a new diskgroup. Note, an ASM diskgroup can house all the Oracle physical files; ranging from controlfiles and datafiles to spfiles and RMAN backup files; however, Oracle executables and non-database files cannot be housed in ASM diskgroups. Throughout this document all Oracle physical files will be generically referred to as *database files*.

Once the database is created, the database instance can operate just like a typical database instance; i.e., all file access is directly performed without extensive ASM intervention<sup>6</sup>. The database file level access (read/write) of ASM files is similar to pre-10<sup>g</sup>, except that any database file name that begins with a “+”, will automatically be handled and managed within the ASM code path. However, with ASM files, the database file access has the characteristics of raw devices; i.e., un-buffered (direct IO) with kernelized asynchronous IO (KAIO).

ASM load balances file activity by uniformly distributing file extents across all disks in a disk group. For this technique to be effective it is important that the disks used in a disk group be of similar size and performance characteristics.

There are two types of file extent distributions: coarse and fine. For coarse, ASM always spreads files in one allocation unit chunks (containing one file extent) across all of the disks in a disk group. Fine distribution interleaves 128k chunks in each AU across all disks. Fine distribution breaks up medium-sized I/O operations, into multiple smaller I/O operations that can execute in parallel.

As discussed above, ASM filesystem layer is transparent to the clients and users. Therefore all datafile access can only be done via the database instance and its utilities. For example, database backups of ASM files can only be performed with RMAN.

---

<sup>5</sup> DBCA can also be used to create the ASM instance and disk groups

<sup>6</sup> Database instances interact with the ASM instance when files are created or opened. At this time the file layout is read from the ASM instance and all subsequent I/O's are done using the layout stored in the database instance. ASM and DB instances also interact if the storage configuration changes (ie disks fail or are added or dropped).

In a database instance three new (sets of) processes are added to help manage the bookkeeping of ASM diskgroups. The following are the three new ASM-related background processes:

- RBAL – the RBAL process performs global opens of all the disks in the disk groups
- ASMB – The ASMB process contacts CSS using the diskgroup name, and acquires the associated ASM connect string. This connect string is then used to connect into ASM instance as a foreground process<sup>7</sup>. Using this persistent connection, periodic messages are exchanged to update statistics and provide a heartbeat mechanism. During operations that require ASM intervention, such as a file creation by a database foreground, the database foreground connects directly to the ASM instance to perform the operation. Also, database file extent maps are sent by ASM to ASMB via this pipe. Additionally, ASMB also sends database IO statistics to ASM instance.
- OOOx – A group of slave processes establish connections to the ASM instance. Through this connection pool database processes can send messages to the ASM instance. For example opening a file sends the open request to the ASM instance via a slave. However slaves are not used for long running operations such as creating a file. The use slave (pool) connections eliminate the overhead of logging into the ASM instance for short requests

---

<sup>7</sup> This connection is always a bequeath connection into ASM instance.

Figure 5 below, shows the [ASM related] init.ora parameters for a default database, named B2R6, created with DBCA

|                      |   |
|----------------------|---|
| spfile               | = +DGR14_A/b2r6/spfileb2r6.ora            |
| control_files        | = +DGR14_A/b2r6/controlfile/current.268.3 |
| compatible           | = 10.1.0.1.0                              |
| db_create_file_dest  | = +DGR14_A                                |
| background_dump_dest | = /opt/oracle/admin/B2R6/bdump            |
| user_dump_dest       | = /opt/oracle/admin/B2R6/udump            |
| core_dump_dest       | = /opt/oracle/admin/B2R6/cdump            |
| db_name              | = B2R6                                    |

A diskgroup can house files from one or more databases, and these databases can be from the same server or can reside on different servers (attached to the same Storage Array direct-attached or via SAN switch). Note, in the latter case where several databases [from different servers] are planning to use the same diskgroup, a RAC implementation is needed to maintain in-sync ASM metadata and requires that an ASM instance exist on each server. This will allow ASM to converse via inter-process communication to maintain metadata coherency.

Allowing multiple databases to share a diskgroup provides more possibilities for improved disk utilization and greater overall throughput.

As stated earlier, a database instance is a client to the ASM instance, this is reflected in the V\$ASM\_CLIENT view<sup>8</sup>. V\$ASM\_CLIENT contains one row for every instance that uses the ASM disk group<sup>9</sup>. See Figure 6 below.

|   |         |           |
|---|---------|-----------|
| SQL> select INSTANCE_NAME, DB_NAME, STATUS from v\$asm_client |         |           |
| INSTANCE  | DB_NAME | STATUS    |
| -----   | -----   | -----     |
| B2R6  | B2R6    | CONNECTED |

<sup>8</sup> This view only contains rows on a +ASM instance. For database instances, there are no entries in V\$ASM\_CLIENT.

<sup>9</sup> When a database instance (using an spfile) initially connects into ASM, there is one additional entry in V\$ASM\_CLIENT. This ASM loopback-connection, is transient and will generally disconnect after 3 minutes.

Since ASM manages the physical access to database files and its metadata, a shutdown of the ASM instance will cause all client database instances to shutdown as well. In normal shutdowns, the ASM instance will begin shutdown and wait for all sessions to disconnect; just as typical database instances. Note, however that ASM has a persistent database instance connection, thus the database instances must be shutdown first, in order for ASM to complete shutdown.

In case of ASM SHUTDOWN IMMEDIATE or ABORT, ASM will immediately terminate any open connections (including database instance connections), and all dependent databases will immediately abort as a consequence.

In a single ASM instance configuration, if the ASM instance fails while disk groups are open for update, then after the ASM instance re-starts, it reads the disk group's log and performs instance recovery. In RAC environments, with multiple ASM instances sharing disk groups, if one ASM instance should fail, another node's ASM instance automatically recovers transient ASM metadata changes caused by the failed instance; i.e. performs instance recovery.

## **Storage Management and Allocation**

A database created under the constructs of ASM will be striped and mirrored; as specified in the SAME methodology; i.e., the IO load is evenly distributed and balanced across all disks within the diskgroup. In our example, the database will be striped across four disks, and redundancy will be handled by EMC storage array. The striping is done on a file-by-file basis, using a 1MB stripe size, as opposed to other LVMs that do striping & mirroring at a disk-volume level. This 1MB stripe depth, also referred to as allocation unit (AU), is the smallest contiguous disk space that ASM allocates<sup>10</sup>.

Each disk in a diskgroup will have an allocation table structure in its disk header. This table structure contains an AU descriptor, one entry per on disk AU. The AU descriptor contains a file number and file extent number. A database file is broken up into file extents. Each file extent is a single allocation unit. Each database file has an associated file extent map, with each map entry referencing a Disk Number and AU number, which uniquely points to location of the (on disk) file extent.

---

<sup>10</sup> The AU size is currently not user configurable.



## Rebalance and Redistribution

With traditional volume managers, expansion/growth or shrinkage of striped filesystems has typically been difficult. Using ASM, these disk/volume activities are now seamless redistribution (rebalancing) of the striped data and can be performed online.

Any storage configuration change will trigger a rebalance. The main objective of the rebalance operation is to provide an even distribution of file extents and space usage across all disks in the diskgroup. Rebalancing is performed on all database files on a per file basis<sup>11</sup>. A new Oracle background process, RBAL, from the ASM instance is responsible for this activity.

The Rebalance process examines each file extent map, and the new extents are re-plotted on to the new configuration. For example, consider an 8-disk diskgroup, with a datafile with 40 extents (each disk will house 5 extents). When 2 new drives (of same size) are added, that datafile is rebalanced and spread across 10 drives, with each drive containing 4 extents. Only 8 extents need to move to complete the rebalance.

The following is a typical process flow for ASM rebalancing:

1. On the ASM instance, DBA adds (or drops) a disk to (from) a diskgroup.
2. This invokes the RBAL process to create the rebalance plan and then begin coordination of the redistribution
3. RBAL will calculate estimation time and work required to perform the task and then message the ARB0, ARB1, etc to actually process the request. The number of ARBx processes invoked is directly determined by the `asm_power_limit`.
4. The Continuing Operations Directory (metadata) will be updated to reflect a rebalance activity.
5. Each extent to be relocated is assigned to an ARBx process.
6. ARBx performs rebalance on these extents. Each extent is locked, relocated, and unlocked. This is shown as Operation *REBAL*.

Rebalancing involves physical movement of file extents. This impact is generally low because the rebalance is done one extent/AU at a time; therefore, there will be only be one outstanding I/O at any given time. This should not adversely affect online database activity. However, it is generally advisable to schedule the rebalance operation during off-peak hours.

---

<sup>11</sup> A weighting factor, influenced by disk size, affects rebalancing. A larger drive will consume more extents. This is done for even distribution based on overall size.

To influence the throughput and speed of the rebalance operation, a new init.ora parameter, called *asm\_power\_limit*, has been introduced<sup>12</sup>. The range of values for *asm\_power\_limit* are 1 to 11; where a value of 11 is full throttle and a value of 1 is low speed. However, the power value can also be set for a specific rebalance activity using the *alter diskgroup* command. This value is only effective for the specific rebalance task. See Figure 7. The important distinction between the *asm\_power\_limit* setting and the alter diskgroup power value, is that the power value can accept a value of 0 while the *asm\_power\_limit* cannot. A *power value of 0* indicates that no rebalance should occur for this rebalance. This setting is particularly important to set when adding or removing storage (that has redundancy), and then deferring the rebalance to a later time [scheduled] time.

Figure 7.

```
"Session1 SQL"> alter diskgroup DGR14_A add disk '/dev/rdsk/c3t19d39s4'
rebalance power 11
```

From another session

```
"Session2 SQL"> select * from v$asm_operation
```

| OPERA              | STAT       | POWER | ACTUAL | SO FAR | EST_WORK | EST_RATE | EST_MINUTES |
|--------------------|------------|-------|--------|--------|----------|----------|-------------|
| 1                  | REBAL WAIT | 11    | 0      | 0      | 0        | 0        | 0           |
| 1                  | DSCV WAIT  | 11    | 0      | 0      | 0        | 0        | 0           |
| (time passes.....) |            |       |        |        |          |          |             |
| OPERA              | STAT       | POWER | ACTUAL | SO FAR | EST_WORK | EST_RATE | EST_MINUTES |
| 1                  | REBAL REAP | 11    | 2      | 25     | 219      | 485      | 0           |

For best practices, it is best to add or remove drives at once, this will reduce the number rebalance operations that are needed for storage changes. Additionally, it is recommended to perform rebalance with the *asm\_power\_limit* or *power value* set to 2 times the number of drives (with a ceiling of 11).

With external redundancy, the dropping and adding of disks in the diskgroup is very seamless, and becomes more of an exercise of scheduling the rebalancing. However, with failure groups, some planning and forethought maybe required with respect to how disks are removed and added. The best practices for this exercise will be covered in a follow-up article.

<sup>12</sup> The *asm\_power\_limit* is specific to each ASM instance.

## Files and Aliases

Files are the objects that database instances access. Files come in the form of datafiles, controlfiles, spfiles, and redo logfiles<sup>13</sup>.

ASM files are no different operationally than non-ASM files. However, ASM files are not visible to the OS through the usual commands like *df*, *ls*, *dd*, etc<sup>14</sup>.

The ASM filename syntax is different than the typical naming standards; yet resembles the filesystem approach of the 9i OMF feature. ASM file names are derived and generated at the successful creation of a datafile. ASM file names are in the format of

*"+diskgroup\_name/database\_name/database file type/tag\_name.file\_number.incarnation id"*

For example, the system tablespace name is

*"+DGR14\_A/b2r6/datafile/system.258.3"*.

Note that the tag name in the datafile name corresponds to the tablespace name. For redo logfiles, the tag name is group number; e.g., *+DGR14\_A/b2r6/onlineolog/group\_3.264.3*. The file number is driving correlation between the database instance's file name and the ASM file.

The illustration below shows the relationship between a database instance's file identity and ASM's. Note, the file number from V\$ASM\_FILE is embedded in the file name.

---

<sup>13</sup> There are a total of 12 file types.

<sup>14</sup> RMAN has complete visibility into ASM infrastructure. To migrate from traditional raw/filesystem files, to ASM, RMAN must be used.

Figure 8.

**+ASM (ASM instance)**

```
SQL> select file_number , sum(bytes)/(1024*1024) from v$asm_file
2* group by file_number
```

| FILE_NUMBER | SUM(BYTES) / (1024*1024) |
|-------------|--------------------------|
| 256         | 360.007813               |
| 257         | 35.0078125               |
| 258         | 450.007813               |
| 261         | .002441406               |
| 262         | 150.007813               |
| 263         | 23.0078125               |
| 264         | 10.0004883               |
| 265         | 5.0078125                |
| 266         | 10.0004883               |
| 267         | 10.0004883               |
| 268         | 2.2109375                |

**B2R14 (database instance)**

```
SQL> select name from v$datafile
```

NAME

```
+DGR14_A/b2r6/datafile/sysaux.256.3
+DGR14_A/b2r6/datafile/system.258.3
+DGR14_A/b2r6/datafile/undotbs1.257.3
+DGR14_A/b2r6/datafile/users.265.3
+DGR14_A/b2r6/example01.dbf
/u01/oradata/b2r6/ishan01.dbf
```

```
SQL> select member from v$logfile;
```

MEMBER

```
+DGR14_A/b2r6/onlinelog/group_3.264.3
+DGR14_A/b2r6/onlinelog/group_2.266.3
+DGR14_A/b2r6/onlinelog/group_1.267.3
```

Note; this database contains ASM files and a non-ASM file named *ISHAN01.dbf*.

The ASM list, from the Figure 8, does not have an entry for the *ISHAN datafile* since it is not ASM managed.

The filename notation described thus far is called the Fully Qualified FileName (FQFN) notation. FQFN are generally long and awkward, therefore, to make file-naming convention easier to remember the ASM Alias name format was introduced.

ASM Aliases are essentially in hierarchical directory format, similar to the filesystem hierarchy: */u01/oradata/dbname/datafile\_name*

Alias names specify a disk group name, but instead of a file and incarnation number, a user-friendly string name is used. Alias ASM filenames are distinguished from fully qualified or numeric names because they do not end in a dotted pair of numbers.

The following is an example of an ASM Alias:<sup>15</sup>:

```
SQL> alter diskgroup DGR14_A add directory '+DGR14_A/oradata/B2R6';

SQL> alter diskgroup DGR14_A add alias '+DGR14_A/oradata/B2R6/ishan01.dbf'
for '+DGR14_A/b2r6/datafile/ishan.314.1'
```

For best practices, every database should implement the Oracle Managed feature. OMF will allow simpler file naming and also provides better file handling. For example if you drop a tablespace, ASM will delete the file only if it is an OMF datafile, if OMF is not used, then the DBA must drop the file manually from ASM instance. Additionally, OMF prevents the accidental dropping of in use files. OMF files are located in the DB\_CREATE\_FILE\_DEST directory<sup>16</sup>. Please review OMF documentation on usage and implementation.

Aliases are particularly useful when dealing with controlfiles; e.g., an Alias ASM filename is normally used in the CONTROL\_FILES initialization parameter.

To show the hierarchical tree of files stored in the diskgroup, use the following connect by clause SQL to generate the full path. Note that you need to have a sufficiently large shared\_pool\_size in your ASM instance to execute this query. A more efficient way to browse the hierarchy is to use EM.

```
SELECT concat(''||gname, sys_connect_by_path(aname, '/')) full_alias_path FROM
  (SELECT g.name gname, a.parent_index pindex, a.name aname,
    a.reference_index rindex FROM v$asm_alias a, v$asm_diskgroup g
    WHERE a.group_number = g.group_number)
  START WITH (mod(pindex, power(2, 24))) = 0
  CONNECT BY PRIOR rindex = pindex;
```

---

<sup>15</sup> An alias can be assigned during file creation, or an alias can be created for an existing file using the ALTER DISKGROUP <diskgroup> ADD ALIAS <alias name> for < file name> command.

<sup>16</sup> There are other \*\_DEST variables that can be used for other file types.

## Templates

ASM file templates are a named collections of attributes applied to files during file creation. Templates simplify file creation by housing complex file attribute specifications. When an ASM template is applied to a file, that file will have all the attributes of that template. When a disk group is created, ASM establishes a set of initial system default templates associated with that disk group. These templates contain the default attributes for the various Oracle database file types. The administrator may change attributes of the default templates. Additionally, administrators can add their own unique templates, as required. This enables specification of appropriate file creation attributes as a template for less sophisticated administrators to use. System default templates cannot be deleted. If you need to change an ASM file attribute after the file has been created, the file must be copied via RMAN, into a new file with the new attributes. This is the only method of changing a file's attributes.

```
SQL> alter diskgroup DGR14_A add template all_new_files attributes (unprotected  
fine)
```

*Once the template is created, it can be applied to a new tablespace.*

```
SQL> create tablespace ISHAN datafile '+DGR14_A/ishan(all_new_files)' size  
100M;
```

## **Conclusion**

ASM provides a point solution focused on unified interface database storage management. This complete solution enables Grid Computing, lowers overall manageability costs as well as helps to maintain serviceability levels. The following is a review of the high points of ASM:

### **Reduce administration complexity:**

- Simplifies or eliminates daily database administration tasks
- Automatic I/O tuning for all types of workloads
- Reduces the number of objects to manage, because vertical integration of file system and volume manager into Oracle kernel reduces complexity, as one disk group replaces multiple file systems. Also, with OMF, you do not have to worry about managing files, just tablespaces.
- Simplifies database storage configuration changes: Automatic data copy on disk add and drop. Online migration to new storage hardware.
- Reduced downtime: ASM prevents accidental file deletion because there is no file system interface, and ASM is responsible for file management.

### **Reduces costs of storage management and Increases Utilization**

- Reduces the cost of managing storage.
- Provides clustered volume manager and file system functionality integrated with the database.
- Makes block devices as simple to administer as file servers.
- Works with any type of disk from modular storage, NAS devices to SAN disk arrays.

### **Improves Performance, Scalability, and Reliability**

- Raw disk I/O performance for all files
- Stripe files across multiple storage arrays
- Higher storage resource utilization
- ASM-Lib API improves CPU utilization to reap I/O and provide priority and caching hints
- Overcomes file system size limitations
- Implements mirroring to protect against storage array failure: Two-way mirroring is supported with additional protection of Failure Groups.

## Appendix A - Partitions, Views and Discover Strings

Shown below is a sample Sun Solaris partition table. Note the use of the s4 partition.

```
partition> p
Current partition table (original):
Total disk cylinders available: 18412 + 2 (reserved cylinders)
```

| Part     | Tag        | Flag      | Cylinders        | Size          | Blocks                      |
|----------|------------|-----------|------------------|---------------|-----------------------------|
| 0        | unassigned | wm        | 0                | 0             | (0/0/0) 0                   |
| 1        | unassigned | wm        | 0                | 0             | (0/0/0) 0                   |
| 2        | backup     | wu        | 0 - 18411        | 8.43GB        | (18412/0/0) 17675520        |
| 3        | -          | wu        | 0 - 2            | 1.41MB        | (3/0/0) 2880                |
| <b>4</b> | <b>-</b>   | <b>wu</b> | <b>4 - 18411</b> | <b>8.43GB</b> | <b>(18408/0/0) 17671680</b> |
| 5        | unassigned | wm        | 0                | 0             | (0/0/0) 0                   |
| 6        | unassigned | wm        | 0                | 0             | (0/0/0) 0                   |
| 7        | unassigned | wm        | 0                | 0             | (0/0/0) 0                   |

The following table illustrates the various used and their relationship with the instance type.

| <b>ASM related V\$ Views</b> | <b>ASM Instance</b>   | <b>Database Instance</b>   |
|------------------------------|---|--|
| V\$ASM_DISKGROUP             | row/diskgroup discovered  | Row/diskgroup used by local database instance.                       |
| V\$ASM_CLIENT                | row/client connected  | row for local ASM instance if DB contains open ASM files             |
| V\$ASM_DISK                  | Row/disk discovered, across all diskgroups as well as disks that are not in disk groups | Row/disk, across all diskgroups used by the local database instance. |
| V\$ASM_FILE                  | Row/file allocated; across all client instances-diskgroups                              | N/A  |
| V\$ASM_TEMPLATE              | Row/template  | Row/template w/ associated-attached diskgroup                        |
| V\$ASM_ALIAS                 | Row/file alias  | N/A  |
| V\$ASM_OPERATION             | Row/per active ASM operation (could be more than 1 operation per invocation)            | N/A  |

The following table shows the various default ASM *diskstrings* that are used on the currently supported platforms. Note, The default search string specified below, is the string used when changed from the initial default parameter value of NULL.

| <b>Operating System</b> | <b>Default Search String</b> |
|-------------------------|------------------------------|
| Solaris (32/64 bit)     | /dev/rdisk/*                 |
| Windows NT/XP           | \\.\PhysicalDrive[1-9]*      |
| Linux (32/64 bit)       | /dev/raw/*                   |
| HPUX                    | /dev/rdisk/*                 |
| HPUX(Tru 64)            | /dev/rdisk/*                 |
| AIX                     | /dev/rhdisk/*                |



## Appendix B- Migrating to ASM

Migrating existing 10<sup>g</sup> databases to ASM, is very straightforward is performed simply through RMAN. The source database location can be on tape or on disk. ASM migration can be performed as either a whole-database; i.e., the entire database or piecemeal migration, where migration is performed on a tablespace or datafile basis. The procedure for either option is similar.

This section outlines the procedures to migrate existing databases into ASM volume management.

1. Check to see if block change tracking is enabled. If Block Change Tracking is enabled, then disable change tracking.

```
SQL> select * from V$BLOCK_CHANGE_TRACKING;
SQL> ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
```

2. Shut down the database consistently.  
SQL> SHUTDOWN IMMEDIATE

3. Modify the initialization parameter file of the target database as follows

Set DB\_CREATE\_FILE\_DEST and DB\_CREATE\_ONLINE\_LOG\_DEST\_[1234] to the desired ASM disk groups. If the database uses a server parameter file, then remove the CONTROL\_FILES parameter that specifies locations of the control file. The control file will be moved to the DB\_CREATE\_\* destination and the server parameter file will be automatically updated. If you are using a client-side server parameter file, then set the CONTROL\_FILES parameter to the ASM alias for the control file name using the disk group and the physical file name. For example, if the original CONTROL\_FILES setting was

```
'/u07/oradata/mig10i/MIG10I/controlfile/o1_mf_zsbzfq9j_.ctl'
```

Then the new ASM setting for CONTROL\_FILES would be

```
'+DG14_A/ o1_mf_zsbzfq9j_.ctl'
```

```
*.control_files=(+DGR14_A/o1_mf_zsbzfq9j_.ctl)
*.db_create_file_dest=+DGR14_A
*.db_create_online_log_dest_1 = +DG14_A
*.db_name='mig10i'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=mig10iXDB) '
*.log_archive_dest_1='LOCATION=+DGR14_A'
```

4. Invoke RMAN and connect to target database. Startup database in nomount mode.  
RMAN> CONNECT TARGET;  
RMAN> STARTUP NOMOUNT;
5. Restore the control file into new locations from location specified in the old Spfile/pfile. This migrates the current controlfile into the DGR14\_A ASM diskgroup.

```
RMAN> RESTORE CONTROLFILE FROM  
'/u07/oradata/mig10i/MIG10I/controlfile/o1_mf_zsbzf9j_.ctl' ;
```

6. Mount the database.

```
RMAN> ALTER DATABASE MOUNT;
```

7. Copy the database into the ASM disk group using the following command. Note, this retains the original files.

```
RMAN> BACKUP AS COPY DATABASE FORMAT '+disk_group';
```

*Optionally, you can speed up copy by increasing RMAN parallelism.*

*RMAN> configure device type disk parallelism 4;*

8. Switch all datafiles into new ASM disk group

```
RMAN> SWITCH DATABASE TO COPY;
```

There will be a switched message for each datafile.

```
datafile 1 switched to datafile copy "+DGR14_A/mig10i/datafile/system.271.1"  
datafile 2 switched to datafile copy "+DGR14_A/mig10i/datafile/undotbs1.259.3"  
datafile 3 switched to datafile copy "+DGR14_A/mig10i/datafile/sysaux.272.1"  
datafile 4 switched to datafile copy "+DGR14_A/mig10i/datafile/users.270.1"  
datafile 5 switched to datafile copy "+DGR14_A/mig10i/datafile/example.273.1"
```

After this operation all datafiles will be in ASM. The original datafiles (ones on old storage) will be catalogued as datafile copies, so you can use them as backup. Or, in case that you decide to migrate back to old storage, you can switch back.

9. RECOVER the DATABASE (if needed)

10. Open the database. Note that there is no need to do *open resetlogs*.

```
RMAN> ALTER DATABASE OPEN;
```

11. Manually create a temporary tablespace (tempfile). Since, the control files do not have a record for the temporary tablespace, RMAN does not migrate/copy over the tempfile. Thus, recreate the temporary tablespace manually in ASM.

```
SQL> create TEMPORARY tablespace temp_tempfile_local size 100M  
3 EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

12. Now, you can move your online logs into ASM. Use the following PL/SQL procedure to perform the redo log migration:

```
declare
cursor orlc is select lf.member, l.bytes
                from v$log l, v$logfile lf
                where l.group# = lf.group#
                and lf.type = 'ONLINE'
                order by l.thread#, l.sequence#;
type numTab_t is table of number index by binary_integer;
type charTab_t is table of varchar2(1024) index by binary_integer;
byteslist numTab_t;
namelist charTab_t;
procedure migrateorlfile(name IN varchar2, bytes IN number) is
    retry    number;
    stmt     varchar2(1024);
    als      varchar2(1024) := 'alter system switch logfile';
begin
    select count(*) into retry from v$logfile;
    stmt := 'alter database add logfile size ' || bytes;
    execute immediate stmt;
    stmt := 'alter database drop logfile ''' || name || '''';
    for i in 1..retry loop
        begin
            execute immediate stmt;
            exit;
        exception
            when others then
                if i > retry then
                    raise;
                end if;
            execute immediate als;
        end;
    end loop;
end;
begin
    open orlc;
    fetch orlc bulk collect into namelist, byteslist;
    close orlc;
    for i in 1..namelist.count loop
        migrateorlfile(namelist(i), byteslist(i));
    end loop;
end;
/
```

13. Now, optionally, you can delete old database files. RMAN has knowledge of the old datafiles, but not the control files or online redo logs. Hence, you must delete the control file and online redo logs manually.

```
RMAN> run {
# delete datafiles
DELETE COPY OF DATABASE;
HOST 'rm <old_online_redo_logs>';
HOST 'rm <old_control_file_copies>'; }
```

14. After migration is done, enable change tracking - if it was disabled.  
`SQL> ALTER DATABASE ENABLE BLOCK CHANGE TRACKING;`

### **Migrating non-ASM online datafiles to ASM.**

This section will describe how to migrate individual tablespaces to ASM, whilst the database is online. This illustration assumes that there is a tablespace named ISHAN located on a mounted filesystem.

1. Connect to RMAN  
`RMAN> connect target`
2. Make the target tablespace offline or read-only  
`RMAN> sql "alter tablespace ISHAN offline";`
3. Copy the ISHAN tablespace to the DGR14\_A diskgroup  
`RMAN> backup as copy tablespace ishan format '+DGR14_A';`
4. On successful copy of the tablespace, switch the ISHAN tablespace to ASM  
`RMAN> switch tablespace ishan to copy;`

When the following message is received, it is deemed that the tablespace is migrated successfully.

*datafile 6 switched to datafile copy "+DGR14\_A/b2r6/datafile/ishan.314.1"*

Note, the original filesystem copy still exists, and can be deleted.