

BEST PRACTICES FOR CREATING A LOW-COST STORAGE GRID FOR ORACLE DATABASES

Low-cost ATA storage arrays, low-cost storage networks, and Oracle Database 10g can in combination create a low-cost storage grid with excellent performance and availability.

In this paper, we provide best-practice guidelines for configuring a low-cost storage grid, based on our extensive evaluations of low-cost storage arrays from multiple major storage vendors that are participating in Oracle's Resilient Low-Cost Storage Initiative. Although this paper focuses on ATA disks and SAN networks, most of these guidelines also apply to Fibre and SCSI disks, iSCSI and Infiniband networks, and NAS storage.

For an overview of Oracle's Resilient Low-Cost Storage Initiative, please refer to <http://www.oracle.com/technology/depoy/availability/htdocs/lowcoststorage.html>. Also on this web site are Best Practice papers from storage vendors that are members of this initiative. These papers provide helpful vendor-specific advice for creating a successful low-cost storage grid.

CANDIDATE USES FOR LOW-COST STORAGE

DATABASE TYPES

Low-cost storage is most successfully deployed for databases with certain types of performance and availability requirements, as discussed in the low-cost storage white paper, http://www.oracle.com/technology/depoy/availability/pdf/lcs_OW.doc.pdf. Compared to traditional high-end storage arrays, low-cost storage arrays have excellent data throughput (i.e. excellent random 1 MB read and write performance) because of the fast transfer rates of ATA disks and their high ratio of network connections to disks. Low-cost storage arrays also provide storage at a superior price per gigabyte. However, low-cost storage arrays do not have better I/O rates (i.e. rate of random 8K reads or writes per second) than traditional storage. While the IOPS (I/Os per second) rate for low-cost storage is typically lower, the cost-per-IOPS is comparable.

Therefore, low-cost storage is best used for databases that are data throughput intensive or have moderate I/O request rate requirements. In general,

- development, test, and low-usage databases
- standby databases
- reporting databases
- data warehouses

are good candidates for low-cost storage. The data area for OLTP databases with a high I/O request rate is sometimes, but not always a good candidate.

FLASH RECOVERY AREA

Oracle's Flash Recovery Area is an ideal candidate for low-cost storage. The Flash Recovery Area is the on-line disk storage used by Oracle to store all recovery-related files, including full backups, incremental backups, archived logs, flashback logs, redo log members, etc. Using disk rather than tape to store recovery-related files results in considerable performance improvements for backup and restore at a reasonable price. Furthermore, because the Flash Recovery Area is accessed predominately with sequential 1 MB streams, the performance characteristics of low-cost storage are well suited to the Flash Recovery Area. The Flash Recovery Area can be configured to use low-cost storage while the data area remains on traditional storage.

USING AUTOMATIC STORAGE MANAGEMENT

While a low-cost storage grid can be configured to be very reliable, an individual storage array may not be as reliable as a traditional higher-end storage array. Using Oracle's Automatic Storage Management (ASM) to mirror data between storage arrays is a critical part of ensuring high-availability of the storage grid. In addition, ASM automatically

distributes data across storage arrays, dynamically rebalancing data as storage arrays and disks are added or removed from the storage grid.

Therefore, you should store your data on low-cost storage using ASM. ASM was first introduced in Oracle 10g Release 1. If you are using a previous version of Oracle, you must upgrade to version 10g Release 1 (or later) and migrate your data from its current storage to ASM. For data migration directions, see http://www.oracle.com/technology/deploy/availability/pdf/Technical_WP_ASM_Migration.pdf.

CONFIGURING ASM

ASM manages pools of disks as a single logical unit called a disk group. Two disk groups should be defined, the first for the database area, which contains the online data files and redo log files, and the second for the flash recovery area, which contains all recovery-related files such as disk backups, archived redo logs, and flashback logs. Each disk group should use ASM redundancy, either "normal redundancy" (2-way mirroring) or "high redundancy" (3-way mirroring), to protect the data. Disks should not be shared between disk groups.

A disk group is comprised of one or more failure groups. A failure group is a collection of disks that can become unavailable when a shared component fails. A disk should only belong to one failure group. A common example of a failure group is all disks within a single storage array. ASM uses the failure groupings to determine where to mirror the data so that component failures can be tolerated.

The example below creates two normal redundancy disk groups. The database area disk group consists of 2 failure groups where each failure group is a 4-disk storage array. The flash recovery area disk group also consists of 2 failure groups where each failure group is a 2-disk storage array.

```
SQL> create diskgroup DATA_AREA normal redundancy
      failgroup DATA_1 disk
        '/dev/raw/array01_01','/dev/raw/array01_02','/dev/raw/array01_03',
        '/dev/raw/array01_04'
      failgroup DATA_2 disk
        '/dev/raw/array02_01','/dev/raw/array02_02','/dev/raw/array02_03',
        '/dev/raw/array02_04';
SQL> create diskgroup RECOVERY_AREA normal redundancy
      failgroup RECOVERY_1 disk
        '/dev/raw/array03_01','/dev/raw/array03_02'
      failgroup RECOVERY_2 disk
        '/dev/raw/array04_01','/dev/raw/array04_02';
```

The figure below depicts the DATA_AREA disk group example. It shows how the data is always mirrored to a disk in a different failure group. It also shows that ASM uses a different layout on each mirror.

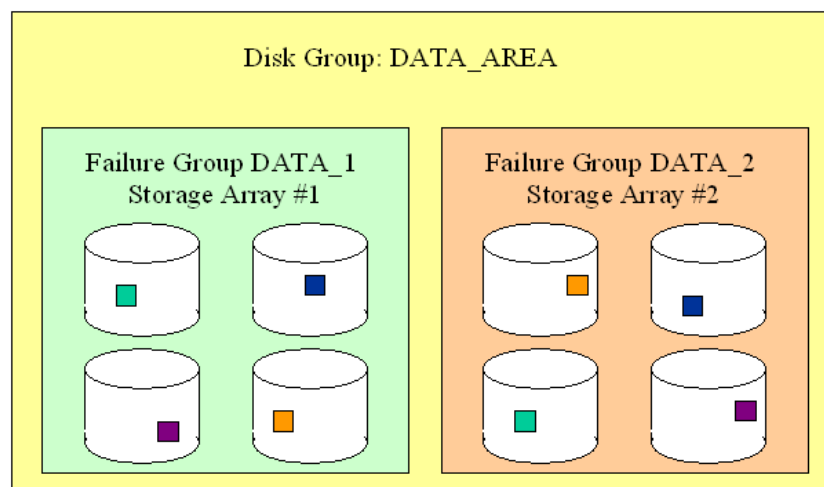


Figure 1: Disk Group with 3 Failure Groups

ASM automatically remirrors data when a disk fails. ASM also automatically rebalances the data within a disk group whenever disks are added or removed from the disk group. The rebalance is limited, however, by the amount of free space within the disk group. Therefore, you should ensure that the amount of free space within a disk group is at least the size of a single failure group.

NORMAL OR HIGH REDUNDANCY

When creating a disk group, you must specify how the data will be protected. External redundancy means that the storage array is responsible for data protection. Normal redundancy means that ASM uses 2-way mirroring. High redundancy means that ASM uses 3-way mirroring.

We recommend that you use normal or high redundancy because ASM can mirror across low-cost storage arrays, protecting you against controller failures. Whether you should use normal or high redundancy depends on many factors.

Normal redundancy uses less disk space than high redundancy. However, if your storage grid is sized to achieve performance goals, you may have extra, unused disk space that you can use for high redundancy.

While ATA disks have good reliability (500K hour MTBF), they are not as reliable as Fibre or SCSI disks for a variety of reasons. Less sturdy mechanical components and packaging make ATA disks more susceptible to complete disk failures. Less rigorous pre-ship testing, lower environmental isolation, and less sophisticated head alignment technology make ATA disks more susceptible to unrecoverable disk sector errors.

With normal redundancy, you may lose data under variations of the following scenarios:

- A disk in one disk group fails. A disk in the other disk group fails before ASM has a chance to rebalance (i.e. remirror) the first disk's data. This scenario will result in the loss of multiple ASM extents (an extent is 1 MB).
- A disk in one disk group fails. As ASM is rebalancing its data, it encounters a bad disk sector error in the other disk group. This scenario will result in the loss of a single ASM extent.
- The database gets a bad disk sector error or a data corruption during a read. Upon reading the block from the other disk group, both to get the data and to repair the first disk's bad disk sector, it encounters another bad disk sector error. This scenario will result in the loss of a single ASM extent.

With normal redundancy, you are more likely to lose a single ASM extent than an entire disk's worth of data.

Clearly, normal redundancy provides less data protection than high redundancy. The last section, Managing the Storage Grid, discusses ways of recovering from the loss of both a single ASM extent and an entire disk. To determine if normal redundancy is sufficient for your needs, you should decide if these recovery mechanisms are acceptable or if you need the extra reliability of high redundancy.

STORAGE FOR FILES STORED OUTSIDE OF ASM

ORACLE HOME

ASM does not currently support storing the Oracle Home directory. For a single instance database, you should place your Oracle Home on local storage. For a multi-instance RAC database, you should create an Oracle Home for each instance on its local storage.

ORACLE CLUSTERWARE FILES

Oracle Clusterware includes 2 important components: the voting disk and the Oracle Cluster Registry (OCR) file. The voting disk is a file that maintains information about node membership. The OCR file contains cluster and RAC database configuration information. Since ASM does not currently store this data, we recommend that you store it as follows.

For pre-Oracle Database 10 Release 2 databases, you should store the OCR file and voting disks in external RAID-protected storage.

For Oracle Database 10 Release 2 databases, you can configure Oracle to manage this data. You need to dedicate a section of multiple disks for this data. Oracle can use replication to tolerate disk failures. Ideally, the OCR file requires 2 disks and the voting disk requires 3 disks. We recommend that you create a small partition on the outside tracks of 5 disks (the partition should be at least 20 MB for the voting disk and 200 MB for the OCR file) in your low-cost storage grid. You can dedicate the remainder of the disk to ASM. If you have created separate disk groups for the data and flash recovery areas, we recommend the following configuration:

- From the storage dedicated to the data area, create
 - 1 partition for the OCR file
 - 1 partition for the voting disk
- From the storage dedicated to the flash recovery area, create
 - 1 partition for the OCR file
 - 2 partitions for the voting disk

STORAGE ARRAY CONSIDERATIONS

When configuring a low-cost storage grid, a key consideration is which low-cost storage array to use. This section describes functionality that you should carefully consider when selecting a storage array.

NETWORK CONNECTIVITY

Networked storage allows a low-cost storage array to be accessed by multiple database servers and multi-instance databases. The storage array should be attachable to a network via Fibre Channel, iSCSI, Infiniband, or as a NAS device. This is a Low-Cost Storage Initiative requirement.

The number of network connections that can be simultaneously active is a key determinant for the storage array's data throughput. Most low-cost storage arrays support at least 2 active network connections; some support up to 4.

SWAPPABLE COMPONENTS

Failure-prone components such as disks, fans, and power supplies should be hot swappable. This is a Low-Cost Storage Initiative requirement.

HOT SPARES

If a disk fails, ASM automatically remirrors the data that was stored on it to unused extents within its disk group. Therefore, you do not need hot spare drives in your low-cost storage array. Any extra disks should be added to ASM's disk groups so that the excess space can be used for remirroring data from failed disks.

CONTROLLER FAILOVER

Many low-cost storage arrays support dual controllers that can be configured such that if one controller fails, the other will take over all I/O requests directed to the failed controller. This feature allows you to seamlessly tolerate a controller failure.

DUAL-PORTED CONTROLLERS

If the controller is dual-ported, then you can configure your storage grid to tolerate a switch failure by connecting the second port to a redundant switch. If both ports can be active simultaneously, then you can also use the second port for increased throughput.

RAID SETTINGS

Low-cost storage arrays typically support multiple RAID configurations, e.g. RAID-0, RAID-1, RAID-5, and no redundancy. As explained above, you should use ASM to protect and layout the data. You should configure the storage array to use "no redundancy" (i.e. configure each disk as an independent LUN), for the following reasons.

RAID-0 (also known as striping) is superfluous if ASM is already distributing the data across disks and storage arrays using the optimal stripe depth of 1 MB, the size of an ASM extent. ASM uses a more sophisticated version of RAID-0 that doesn't require the data to be immediately restriped when disks are added or removed.

RAID-1 (also known as mirroring) is also superfluous for a low-cost storage grid if ASM is mirroring data between storage arrays. Because ASM mirroring protects against storage array failures as well as disk failures, it is superior to using RAID-1 within the storage array. For extra protection, ASM can triple-mirror the data.

If you use RAID-5 without ASM mirroring, you will be susceptible to storage array failures. If you use RAID-5 with ASM mirroring, you will pay a performance penalty. The Oracle database typically emits I/Os in 2 sizes: 8 KB (or the size of the database block) and 1 MB. 8 KB writes are slow in a RAID-5 configuration because a RAID stripe is typically larger than a database block and hence a write must be preceded by a read. 8 KB reads to a failed disk are slow because all other disks within the RAID set must be read. 1 MB I/Os are slow because the RAID stripe depth is typically smaller than 1 MB and multiple small disk I/Os are not as efficient as a single large disk I/O.

Not all low-cost storage arrays support no redundancy. As an alternative, you can use a RAID-0 configuration with a stripe depth of 1 MB (or the storage array's maximum stripe depth size).

NUMBER OF HOSTS

Most low-cost storage arrays limit the number of hosts that can be simultaneously connected to it. This number is important because it limits the number of hosts that an ASM server can manage and the maximum size of an Oracle RAC cluster accessing it.

REMOTE MANAGEMENT AND ALERTS

The storage array should automatically alert the administrator about any hardware failures via a remote alerting mechanism such as email and pager. The storage array should also send alerts via SNMP so that Oracle's Enterprise Manager can capture them. The storage array should also have remote management and monitoring tools. These are all Low-Cost Storage Initiative requirements.

INTER-OPERABILITY CERTIFICATION

A storage grid is often implemented within an existing SAN or LAN network. Therefore, the storage array must work well with existing HBAs, switches, and host operating systems. Our experience has shown that even though a storage array conforms to, for example, Fibre Channel specifications, it may not be compatible with other Fibre Channel conforming HBAs, switches, or drivers. This is a by-product of protocol versions, not with low-cost storage per se.

Therefore, you should verify that the storage array is certified with the network switches, host HBAs, drivers, and operating systems you are planning to use.

SIZING THE STORAGE GRID

Before creating a storage grid, you must decide how many storage arrays you need and the configuration of each storage array, e.g. number of controllers, network connections, disks, etc. To make these decisions, you must first decide if you are optimizing for storage capacity, data throughput, or I/O rate.

OPTIMIZING FOR CAPACITY

You can use two approaches to add storage to a low-cost storage grid. Many low-cost storage arrays allow you to expand by adding or daisy-chaining additional disk trays. We call this "scaling up". On the other hand, you can use storage arrays with a single disk tray and expand by adding additional storage arrays. We call this "scaling out".

If you are optimizing for storage capacity rather than performance, then you should consider using the scale-up approach to build your storage grid. The rest of this paper focuses on the scale-out approach.

OPTIMIZING FOR PERFORMANCE

If you are optimizing for performance, then you can configure your storage grid in 3 major steps, each of which is described in the following sections.

1. Characterize your database's performance requirements.
2. Characterize your storage array's performance capabilities.
3. Configure your storage grid.

CHARACTERIZING A DATABASE'S I/O WORKLOAD

To understand your database's performance requirements, you must consider:

1. Are your I/O requests primarily small I/Os (i.e. single-block) or large I/Os (i.e. multi-block)?

Databases issue large I/Os when performing the following types of operations: queries that require table or index scans, direct data loads, backups, and restores. In general, data warehouse environments issue large amounts of large I/Os whereas OLTP databases primarily issue small I/Os. However, batch operations and backups on OLTP databases also issue large I/Os.

For pre-10.2 databases, the number of blocks in a large read I/O is controlled by the database parameter "DB_FILE_MULTIBLOCK_READ_COUNT" and is limited to the operating system's maximum I/O size. For 10.2 databases, the large I/O size is set to a platform-specific limit, which is typically 1 MB.

A small I/O is typically 8 KB but can range from 2 KB to 32 KB. A large I/O is typically 1 MB.

2. What is your average and peak I/O rate (IOPS) requirement? What percentage are writes?

If your database is primarily issuing small I/Os, you should optimize for your average and peak I/O rate.

3. What is your average and peak throughput (MBps) requirement? What percentage are writes?

If your database is primarily issuing large I/Os, you should optimize for your average and peak data throughput rate.

For a 10g Release 2 database, you can characterize your I/O workload using the statistics in the GV\$SYSSTAT view specified below. These statistics are cumulative values. Therefore, they should be sampled during the beginning and end of typical and peak periods and transformed into rates.

- For small I/Os:
 - Number of small reads: "physical read total IO requests" minus "physical read total multi block requests"
 - Number of small writes: "physical write total IO requests" minus "physical write total multi block requests"
- For large I/Os:
 - Number of large reads: "physical read total multi block requests"
 - Total bytes read: "physical read total bytes"
 - Number of large writes: "physical write total multi block requests"
 - Total bytes written: "physical write total bytes"

For a pre-10g Release 2 database, the I/O statistics are in multiple views. The bulk of the I/O workload is measured as follows. These statistics do not account for I/Os corresponding to the control file and archiver. Unless specified otherwise, these statistics are cumulative values that should be sampled during the beginning and end of typical and peak periods and transformed into rates.

- For small I/O:
 - Number of large reads from data files: GV\$FILESTAT.SINGLEBLKRDS
 - Number of large writes to data files: GV\$FILESTAT.PHYWRTS
 - Number of redo log writes: in GV\$SYSSTAT, "redo writes" specifies the number of I/Os.
 - Number of backup I/Os: in GV\$BACKUP_ASYNC_IO and GV\$BACKUP_SYNC_IO, IO_COUNT specifies the number of I/Os. Each row of this view corresponds to a data file, the aggregate over all data files, or the output backup piece. Use the values from the last 2 row types.
 - Number of flashback log I/Os: in GV\$SYSSTAT, the "flashback log writes" statistic specifies the number of write I/Os to the flashback log.
- For large I/Os:

- Number of large reads from data files: `GV$FILESTAT.PHYRDS` minus `GV$FILESTAT.SINGLEBLKRDS`
- Number of large writes to data files: `GV$FILESTAT.PHYBLKWRT` specifies the number of DBWR and direct I/O block writes. Unfortunately, there isn't a way to derive the number of large I/Os, but you can assume that DBWR I/Os are small I/Os and direct I/Os are large I/Os. The number of direct I/O block writes is `GV$FILESTAT.PHYBLKWRT` minus `GV$FILESTAT.PHYWRTS`.
- Number of redo log writes: in `GV$SYSSTAT`, “redo blocks written” specifies the number of blocks written.
- Number of backup I/Os: in `GV$BACKUP_ASYNC_IO` and `GV$BACKUP_SYNC_IO`, `TOTAL_BYTES` specifies the number of bytes read or written. Each row of this view corresponds to a data file, the aggregate over all data files, or the output backup piece. Use the values from the last 2 row types.
- Number of flashback log I/Os: in `GV$FLASHBACK_DATABASE_STAT`, `FLASHBACK_DATA`, `DB_DATA`, and `REDO_DATA` specify the number of bytes read or written from the flashback logs, data files, and redo logs, respectively, in the given time interval.

For both 10g Release 2 and pre-10g Release 2 databases, you can characterize your database workload as follows:

- For I/O rates, calculate the number of small and large I/Os issued per second.
- For data throughput rates, calculate the number of bytes of small and large I/Os issued per second.

For a data warehouse application, you can roughly estimate your bandwidth requirements by assuming that each of today's CPUs requires about 200 MBps to feed it. Therefore, for a RAC cluster of 2 nodes, each containing 4 CPUs, you would need approximately 2 x 4 x 200 MBps, or 1.6 GBps of throughput.

ESTIMATING THE PERFORMANCE CAPABILITIES OF A STORAGE GRID

To understand a storage array's performance capabilities, you must obtain performance measurements over different workloads. You can download a tool called Orion from the Low-Cost Storage Initiative web site to calibrate a storage array's performance without installing or creating a database. Orion provides the following statistics:

- Multi-user sequential read throughput (in MBps): sustainable throughput of random large reads
- Multi-user sequential write throughput (in MBps): sustainable throughput of random large writes
- Random read I/O rate (in IOPS): sustainable rate of random small reads
- Random write I/O rate (in IOPS): sustainable rate of random small writes

You can also approximate a storage array's potential performance using the following guidelines:

- Its sustainable read throughput is bounded by the number and sustainable throughput of active network connections to it. The sustainable throughput of a connection is lower than its peak because of protocol overhead. For example, a 2 Gbit Fibre Channel can sustain about 180 MBps of throughput. If a storage array has multiple network connections, all of its connections may not be able to be active at the same time (some network connections are only for failover). In practice, storage arrays often have internal bottlenecks that further restrict the throughput.
- Its sustainable write throughput should, in theory, be higher than the read throughput because storage arrays with a cache can aggressively reorder I/Os to minimize disk seek time (i.e. use a disk elevator algorithm). In practice, the write throughput is usually lower than the read throughput because ATA disks process writes slower than reads and many storage arrays do not reorder I/Os aggressively.
- Its sustainable random read rate is bounded by the number of disks times the random read rate per disk. A 7200-rpm ATA disk can sustain about 79 IOPS. In practice, the random read rate may be only half of this upper bound because of other bottlenecks within the storage array.
- Its sustainable random write rate should, in theory, be higher than the random read rate with the use of write

reordering. In practice, the random write rate ranges from 60% to 200% of the random read rate.

- Many performance statistics reported by storage arrays reflect cached I/O performance. Because the Oracle database contains a buffer cache, read I/Os do not typically benefit from the storage array's cache. Write I/Os do benefit from the cache for write optimizations and if RAID-5 is enabled. However, since any sustained I/O workload requires that the writes be eventually written to disk, I/O performance of writes to disk rather than to cache is the more significant metric.

CONFIGURING THE STORAGE GRID

Once you have determined your database's performance requirements and your storage array's performance capabilities, you can decide how to configure your storage array from a performance and availability perspective.

If you are using ASM normal (2-way) or high (3-way) redundancy, ASM issues writes to all mirrors. However, ASM issues a read to only one of the mirrors. Therefore, for 'n' mirrors, the required write throughput or rate should be multiplied by 'n'.

If you are configuring for storage capacity, you can determine the required number of storage arrays by dividing your total required storage size, including storage for the mirrors, by the storage size of a single storage array.

If you are configuring for performance, you can determine the required number of storage arrays by dividing your total performance requirement (throughput or I/O rate) by the performance capability of a single storage array.

CONFIGURING THE STORAGE ARRAY

RAID

As previously explained in the Storage Considerations section, a low-cost storage array should not be configured for redundancy when using ASM. If a "no redundancy" option is not supported, you can use a RAID-0 configuration with a stripe depth of 1 MB (or the storage array's maximum stripe depth size).

CACHE

Most storage array controllers have a cache. If the cache is not implemented with NVRAM (i.e. battery backed), the cache must be disabled.

An NVRAM cache is critical for performance if the controller is protecting the data with RAID-5. Without RAID-5, its benefit for reads is marginal because most access locality is captured by the database's buffer cache. However, a cache can greatly help the performance of writes.

Therefore, you should use an NVRAM cache, but it does not need to be very large. You should dedicate more cache space for writes than for reads.

CACHE LINE SIZE

If the storage array controller has a cache, you can often configure the cache's line size. You should set the cache line size to be as large as possible and at least as large as the database block size. Properly configuring the cache line size is particularly important for large I/O workloads.

PREFETCHING

One benefit of a cache that vendors frequently tout is that it allows the controller to prefetch. In practice, all of the database's sequential accesses use large 1 MB I/Os. In addition, in situations where prefetching would improve performance, the database performs the prefetching itself.

If your host server can issue 1 MB I/Os to the storage array without fracturing by the operating system or host bus adapter (HBA) driver, consider disabling prefetching. The value of the prefetching performed by the controller is questionable and, in heavy mixed workloads, may even hurt performance.

However, if your host server fractures 1 MB I/Os, enable prefetching.

WRITE CACHE MIRRORING

If a storage array contains multiple controllers, each with their own write cache, then all write data is typically mirrored

across the caches so that in the event of a controller failure, another controller can flush the cached write data to disk. The mirroring of the write caches often greatly impairs performance. In our evaluation, the overhead of write cache mirroring degraded write performance by 50% on some storage arrays.

On some storage arrays, for improved performance, you can disable write cache mirroring. The database can withstand the loss of a controller with unflushed writes in its cache if ASM is mirroring across storage arrays and all disks are always accessed from the same controller. When ASM detects a controller failure, it marks the disks associated with the controller as failed. It will automatically rebuild the data on these disks elsewhere.

If you disable write cache mirroring, you should be prepared to tolerate the time and overhead for rebalancing the data on the disks mapped to a controller in the event of a controller failure.

CONFIGURING THE HOST

OPERATING SYSTEM

The configuration of the host server's operating system has a significant impact on performance. The section below focuses on tuning I/O on Linux. For all operating systems, the most important issue is if 1 MB I/Os are issued to the storage array as a single I/O request or are fractured into multiple I/O requests. If the request is fractured, the storage array cannot take advantage of the performance advantages of issuing one large I/O instead of multiple, sequential smaller I/Os.

HBA AND HBA DRIVER

The HBA and its driver have a significant impact on performance. The section below describes some issues particular to Linux. In general, be sure that you have loaded your vendor's most recently recommended driver.

The number of network connections from the host is one factor for determining its peak I/O throughput. Be sure to configure a sufficient number of network connections for your peak load.

LINUX

LINUX 2.4

In some versions of the Linux 2.4 kernel, large 1MB I/Os are broken up into 32K segments. This results in poor performance, particularly for queries that perform table or index scans, backup or restore jobs, and other workloads that issue large I/Os. Oracle has addressed these issues with changes in both the database and Red Hat Linux. In our internal benchmarks, these fixes resulted in a ten-fold speedup in sequential I/O performance.

For improved performance, we recommend that you do the following:

1. Call Oracle Support and reference bug #4039598. You will need your CSID (Customer Support ID) to initiate the call. Oracle will provide you with a fully-supported version of the Red Hat 3.0 Update 4 Linux kernel. The fixes enable Linux to issue I/Os based on the largest size the HBA can support or 128 KB, whichever is smaller.

2. Install the kernel:

```
# rpm -ivh kernel.rpm
```

3. Update the `aio-max-size` parameter.

Add the following line to `"/etc/sysctl.conf"`:

```
fs.aio-max-size = 1048576
```

This change will properly set this kernel parameter across reboots. To change it dynamically on a running system, issue the following command as root:

```
# echo 1048576 > /proc/sys/fs/aio-max-size
```

You can check the value of this parameter as follows:

```
# cat /proc/sys/fs/aio-max-size
```

4. Various changes were also made to the Oracle database to improve Linux performance. These improvements are available in Oracle 10g Release 2 and Oracle 10.1.0.4. Check the Oracle web site for availability.

LINUX 2.6

Linux 2.6 contains the performance improvements described above. Linux 2.6 is currently available as SUSE Linux Enterprise Server (SLES) 9 or Red Hat 4.0. The maximum I/O size is determined by the maximum number of 4KB pages the HBA driver supports, specified by the kernel parameter, `sg_tablesize`. Many HBAs are still limited to 32 pages, resulting in a maximum I/O size of 128 KB.

ACHIEVING HIGH AVAILABILITY

TOLERATING CONTROLLER FAILURES

See the sections above regarding controller failover and write cache mirroring.

TOLERATING SWITCH AND HBA FAILURES

In addition to tolerating failures within the individual arrays, a highly available storage grid must be able to tolerate failures of components that connect the database grid to the storage grid. These critical components include the network switches and the HBAs on the database servers and storage arrays. A highly available storage grid has redundant paths between each database server and storage array in the grid.

To tolerate connectivity failures, each database server and storage array should have at least two HBAs or a single dual-ported HBA, with each port connecting to a different switch. When a single HBA or switch fails, a database server will still be able to access any storage array. The following figure shows two paths from the servers to the storage arrays.

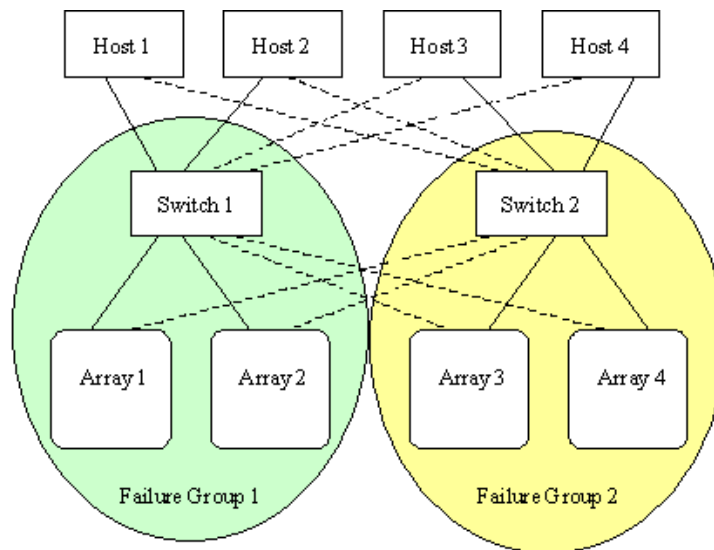


Figure 2: Low-Cost Storage Grid with Redundant Paths

MULTI-PATHING

Multi-pathing is available from various storage and operating system vendors. Multi-pathing contains multiple types of functionality that are important for a low-cost storage grid. It helps the database tolerate switch, HBA, connection, and controller failures (or upgrades) by masking these failures so long as at least one path to the storage array is operational. A multi-pathing package may only support a subset of the functionality described below.

Dynamic load-balancing provides the ability to utilize multiple paths to a disk. The paths may use differing HBAs, HBA ports, switches, and storage array controllers. Dynamic load-balancing helps database performance by intelligently directing I/Os across all available paths, resulting in even loads on these paths. This feature allows the database to achieve throughput that is greater than that of a single connection.

Multi-pathing software may also support the concept of a preferred versus failover path. The preferred path is always

used, until it fails at which point the failover path is used. This feature is useful for controllers that have 2 ports, only one of which can or should be active at a time.

Once a failed path has been repaired, multi-pathing software that implements failback will detect that the path is working via periodic checks. If it is dynamically load-balancing, it will utilize the path. If the path is a preferred path, then it will issue subsequent I/Os on the preferred path rather than the failover path.

Finally, the presence of multiple paths to a disk can be automatically discovered by some multi-path software. If not available, the onus remains on the administrator to properly configure the multi-path device.

When configuring ASM, you can use only one name to reference a disk. Therefore, when using multi-pathing, you must reference the disk by its multi-path device name. For further information, see <http://www.oracle.com/technology/tech/linux/asmlib/multipath.html>.

Linux 2.4 has limited support for multi-pathing. You can manually configure a multi-path disk name that is an alias for all device names for the same disk, e.g. /dev/md0. Failover from one path to the other only occurs for time-outs and errors. Some versions of Linux 2.4 (e.g. RedHat 3.0) support failback.

Linux 2.6 addresses the limitations in Linux 2.4. It proactively discovers multi-paths and load-balances between the paths. It proactively discovers problems with a path, fails-over to an alternate path, and fails-back when the problem has been rectified.²

MANAGING THE STORAGE GRID

BAD DISK SECTORS

A disk read may result in a transient error that the controller can overcome by retrying the operation multiple times. In some cases, the controller will not be able to succeed, despite multiple retries. This type of error is a result of a bad disk sector. A disk can have a bad sector that was undetected during the pre-ship disk evaluation test, that developed over time, or was the result of a bad write.

All types of disks (e.g. Fibre, SCSI, and ATA) can have bad sectors. However, ATA disks are more susceptible to bad sectors, due in part to their packaging which provides less environmental insulation and shorter pre-ship evaluation tests, as discussed earlier.

You can protect yourself from bad sector failures in multiple ways. You can configure ASM to mirror the data with either normal redundancy (2-way) or high redundancy (3-way). If ASM encounters a bad sector, it will automatically repair it from the mirror. By using high redundancy, you can reduce the probability of having bad sector and/or total disk failures on all mirrors.

If you lose a sector because of a double failure with normal redundancy or a triple failure with high redundancy, you can still recover from the error. For example, you can recover from a data block failure using RMAN BMR. Many files such as the redo log and control file are mirrored by the database. You can recover from failures to full and incremental backups using RMAN.³

Finally, most low-cost storage arrays have a disk scrubbing tool that scans the disks, looking for bad, unreadable disk sectors. You can typically configure the disk scrubbing tool to not interfere with the database workload by, for example, specifying a schedule or a maximum rate for the disk scrub.

If the storage array protects the data with RAID-1 or RAID-5, it can automatically remap the bad sector to a new location and populate the new sector, using the redundant data. If you are following the best practices and using ASM to mirror data between storage arrays, then you can ask ASM to rebuild the data by dropping the disk and immediately adding it back.

```
SQL> alter diskgroup DATA_AREA drop disk '/dev/raw/array01_01' rebalance wait;
```

² This Best Practice paper will be updated when a more detailed evaluation of Linux 2.6 multi-pathing has been completed.

³ This Best Practice paper will soon be updated with a reference to an upcoming paper that describes how to recover from block corruptions to any file.

```
SQL> alter diskgroup DATA_AREA add disk '/dev/raw/array01_01';
```

UNHEALTHY DISKS

Most low-cost storage arrays can check the health of each disk using SMART technology. SMART is an acronym for Self-Monitoring Analysis and Reporting Technology. SMART disks can monitor their status and report issues. SMART is an industry standard and is supported by many leading disk manufacturers such as Maxtor, Seagate, Western Digital, etc.

If the storage array alerts you that a disk is unhealthy, you can react proactively by asking ASM to drop the disk. ASM will then rebalance the data on the other disks. If you have a replacement disk, you can hot-swap the disk and ASM will rebalance the data onto this new disk. Because ATA disks are relatively inexpensive, especially compared to SCSI and Fibre disks, you should consider proactively replacing any disks that are unhealthy or incurring many bad sector errors. You can rebuild the data at your convenience during an off-peak period, rather than risk the problems associated with a full disk failure.

FLASHBACK

Unlike traditional storage, low-cost storage arrays do not typically support snapshots. If they do, it is not possible to coordinate the use of snapshots across multiple storage arrays. The Flashback functionality in Oracle Database 10g can be used in lieu of storage-side snapshots to recover from various human errors. Flashback Database allows you to recover the database to a previous point in time. Flashback Table allows you to recover a table to a previous point in time. Flashback Transaction allows you to identify changes made by a particular transaction or transactions within a specified period of time.

For additional information on Flashback functionality, you can refer to the Oracle Database 10g Backup and Recovery Advanced User's Guide.

DATA GUARD

You can protect your database against various disk, storage array, or network failures, as described in this paper. For additional protection, you should consider implementing a standby database with Oracle DataGuard. DataGuard protects against large-scale storage and connectivity failures. It also protects against human errors, corruptions, and software defects.

If you do decide to use DataGuard, you may determine that you can tolerate relatively infrequent storage-related failures in either your primary or standby database, such as switch failures and double disk failures. That is, you may decide that you would rather invest in a standby database than a redundant network infrastructure or triple storage redundancy.

SUMMARY

Before creating a low-cost storage grid, you should determine which databases and data areas you plan to deploy on it. You will be most successful by using it for throughput-intensive databases such as data warehouses or for moderate I/O workloads such as reporting databases or the Flash Recovery Area. You can get superior performance and availability from your low-cost storage grid through careful sizing and configuration of your host servers, storage network topology, ASM, and storage arrays.