# Incorporating Oracle On-Line Space Management with Long-Term Archival Technology

**Steven M. Moran and Victor J. Zak**
Oracle Corporation
Advanced Programs Group
President's Plaza
Suite 200
196 Van Buren Street
Herndon, Virginia 22070
smmoran@us.oracle.com and vzak@us.oracle.com
+1-703-708-6778
Fax: +1-703-708-7919

## Abstract

The storage requirements of today's organizations are exploding. As computers continue to escalate in processing power, applications grow in complexity and data files grow in size and in number. As a result, organizations are forced to procure more and more megabytes of storage space. This paper focuses on how to expand the storage capacity of a very large database (VLDB) cost-effectively within a Oracle7 data warehouse system by integrating long term archival storage sub-systems with traditional magnetic media. The Oracle architecture described in this paper was based on an actual proof of concept for a customer looking to store archived data on optical disks yet still have access to this data without user intervention. The customer had a requirement to maintain 10 years worth of data on-line. Data less than a year old still had the potential to be updated thus will reside on conventional magnetic disks. Data older than a year will be considered archived and will be placed on optical disks. The ability to archive data to optical disk and still have access to that data provides the system a means to retain large amounts of data that is readily accessible yet significantly reduces the cost of total system storage. Therefore, the cost benefits of archival storage devices can be incorporated into the Oracle storage medium and I/O subsystem without loosing any of the functionality of transaction processing, yet at the same time providing an organization access to all their data.

## Introduction

As organizations rely more and more heavily on historic/legacy data for trend analysis and data mining for competitive advantage purposes, it is imperative that the organization has ready access to all its data. Maintaining data on-line, both historic and current, however, comes with the price of additional hardware costs (i.e., magnetic disk devices and their controllers). As data ages, it may not be accessed or updated as frequently as current data, yet still needs to be accessed on a periodic basis. An alternative means to effectively manage and store the data becomes necessary to ensure the organization has

access to its data. The use of lower cost archival storage media for long term archival storage provides the means to control costs and have ready access to all data. How will relational data stores, such as Oracle's architecture handle the slow response times that are typically associated with archival optical devices?.

This paper will discuss a prototype system which used both magnetic media and near-line optical technology with the Oracle7 relational database management system. The architectural design contained multiple Oracle tablespaces, storing on-line transaction data on magnetic devices and storing archival information/transactions on optical disks, both accessed from the same Oracle instance. An Oracle instance consists of an area of allocated memory named the System Global Area (SGA) and a number of Oracle processes. To test the prototype, a C program and Oracle's PL/SQL modules, managed the movement of aged data from on-line tablespaces to archival tablespaces.

The goal of the prototype was to prove to a customer that Oracle's relational database management system can effectively manage their proposed system architecture, consisting of 10 years worth of data (approximately 1.2 terabytes), stored on both magnetic and optical media. The customer needed to ensure that data stored on magnetic and optical media can be accessed transparently by Oracle in a reliable and feasible manner while meeting their performance criteria. The customer requires a cost-effective means to store and provide access to their data which is expected to grow over the years.

## Architecture

The prototype architecture was designed using the Oracle7 database, an optical juke box and the Archival Management and Storage System (AMASS™) file system software. AMASS is a product of EMASS Incorporated. The AMASS file system is completely transparent and provides direct access to both optical jukeboxes and high-speed tape libraries on workstations and departmental servers. The AMASS architecture implements a block-based, direct access paradigm for virtual storage, creating what appears to be unlimited disk capacity. AMASS makes the drives and media (volumes), normally considered off-line storage, appear as a single, on-line logical device with a single mounted file system. The AMASS file system is implemented at the virtual file system (VFS) layer of the UNIX Kernel. Incorporation of the AMASS file system at the VFS layer provides system call transparency to host applications. The core modules of the AMASS file system are: Metadata On-line Index, Cache I/O module. The AMASS architecture contain file information in a fnode structure which are maintained in an on-line index database index.

The Oracle architecture consisted of a total of four tablespaces. An Oracle database is divided into logical units called tablespaces. A tablespace is used to group related logical structures together. One or more datafiles (physical operating system files) are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace. The prototype architecture allocated one datafile per tablespace. Two of the Oracle tablespaces were based on datafiles created on a magnetic drive using the UNIX file system and the other two tablespaces' datafiles were created on the optical disks through AMASS. It should be noted that the datafiles on the optical disks were on separate disks.

## Oracle & AMASS I/O Architectures

Oracle and AMASS I/O architectures match up perfectly and allow coexistence of both products. Both systems use a cached block I/O management design to increase performance and minimize unnecessary I/O to a storage device.

Oracle manages its data in the database buffer cache section of the SGA. Database buffers store the most recently used blocks of database data. These buffers can contain modified data that have not yet been permanently written to disk. Users connect through user processes and communicates with the database through server processes. Oracle creates the server processes to handle requests from connected user processes. User I/O requests,

via application programs or dynamic access, are handled by system processes which handles read (server process) and write (database writer (DBWR)) requests. Oracle can be configured to vary the number of user processes per server process. In a *dedicated server* configuration, a server process handles requests for a single user process. A *multi-threaded server* configuration allows many user processes to share a small number of server processes.

To access the data residing on the optical disks, the AMASS file system handles I/O requests for the Oracle processes instead of UFS (UNIX File System). AMASS implements an I/O cache area similar to Oracle's SGA. The AMASS cache consists of raw partitions which are used as the staging area to handle read/write requests between the media and the file system. If the data the user requests resides in cache, the request is satisfied immediately, otherwise I/O processes request the appropriate media be loaded into a drive and then the data is subsequently read into the cache. The data is then returned to the requesting user process, which in this case, would be the Oracle server process. This block of data will also then be stored in the database buffer cache of the SGA until it is swapped out.

**Reads:** If a user's request for data exists in the database buffer cache of the SGA, results are returned immediately. If the requested data is not resident in the database buffer of the SGA, a server process requests the proper database blocks from either the UFS or AMASS datafiles and returns the data to the Oracle server process. If the data is physically resident on the AMASS datafile and the requested data blocks are in AMASS cache, the data is read into the database buffer, if not, the blocks need to be physically accessed from the optical media.

**Writes:**Write requests between both architectures (UFS and AMASS) are handled in a similar manner. The SGA and AMASS cache have similar queuing rules. All Oracle database transactions - inserts, updates or deletes - are processed within the SGA. The changed database blocks are not immediately written to their respective datafiles. All committed transactions are saved to Oracle redo log files to be used for database recovery if necessary. In all write operations, the Oracle server process, DBWR, flushes the dirty database blocks in the database buffer cache to the data's respective datafiles. Datafiles resident on the UNIX file system are updated immediately. The datafiles on the AMASS file system are initially written to the AMASS cache. AMASS has its own processes that are responsible for flushing its cache to its storage media, in this case, the optical disk. AMASS has one *libio_#* process for each drive within the library. AMASS maintains a sorted queue of write requests in which the *libio_#* process reads to determine which blocks are available to be flushed to the optical disk. Additionally, Oracle and AMASS have different schemas to guarantee the SGA or AMASS cached data is not lost during system anomalies.

Figures 1 illustrates the high level I/O flow between Oracle and AMASS. Figure 2 illustrates how data blocks are passed through AMASS cache, the UNIX VFS layer, the Oracle SGA and finally to the user.

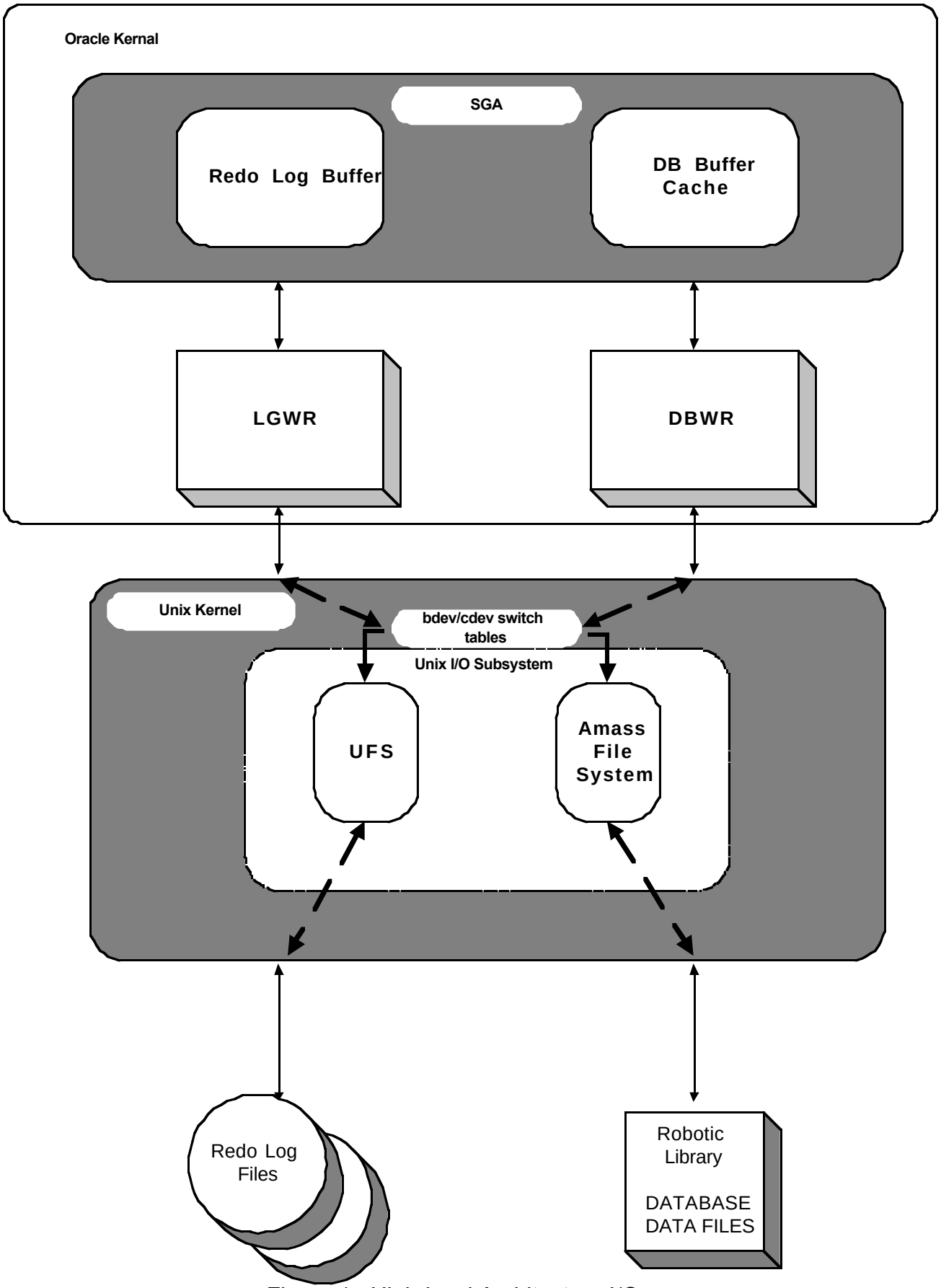


Figure 1: High-level Architecture I/O

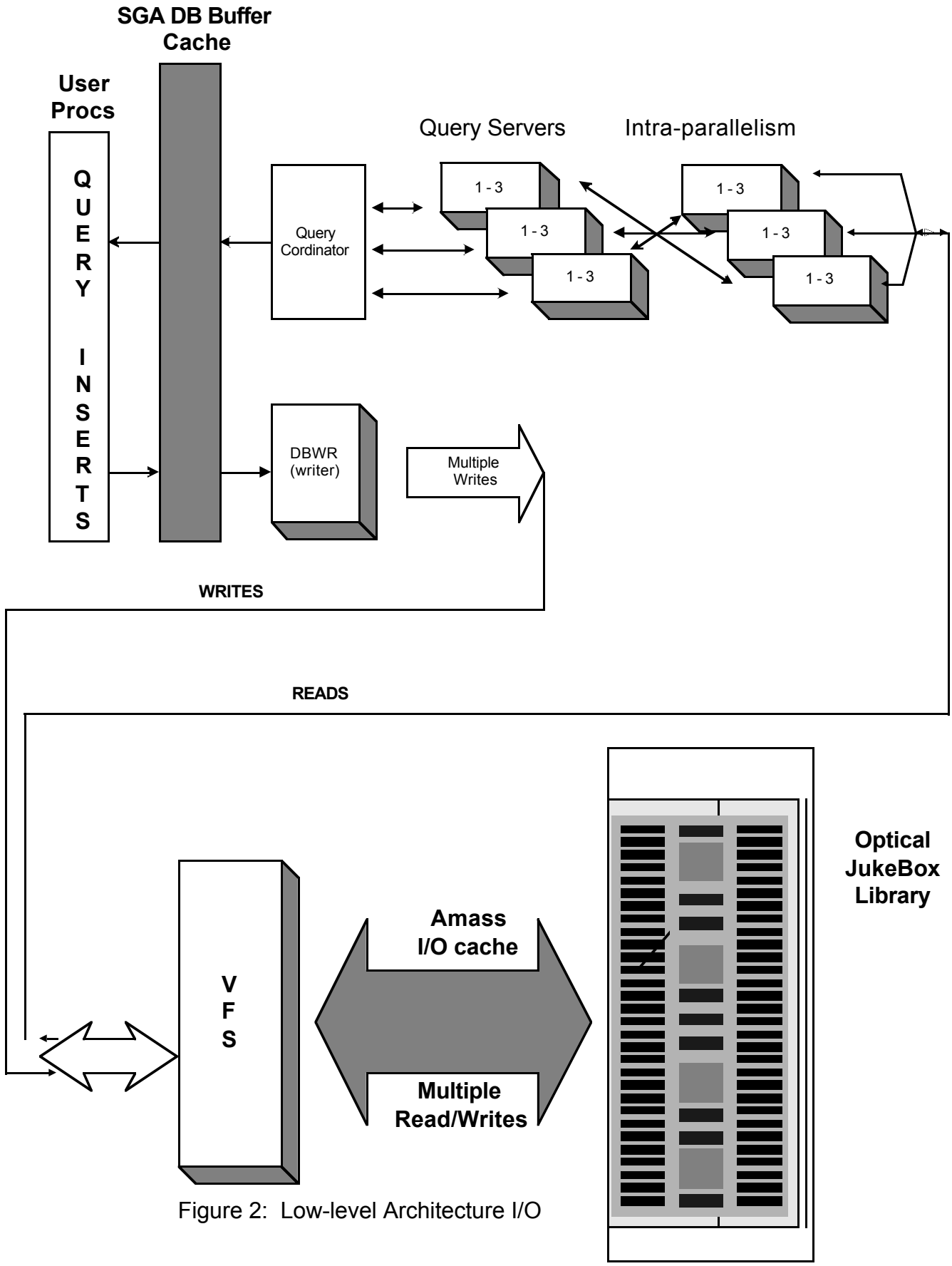


Figure 2: Low-level Architecture I/O

## Prototype Proof of Concept

### Prototype Environment

*Configuration 1: Functionality Test*

The first test was conducted on an SGI Challenge L Server. The server had 2 processors, 32 Megabytes of main memory and a 2 gigabyte disk drive sharing a controller with other drives. All the Oracle software, datafiles and redo logs were on the same disk drive. All monitoring and startups were initiated and processed on the server as opposed to a client.

*Configuration 2: Performance Test*

The second run was conducted on an SGI Challenge Server. The server had 4 processors, 756 Megabytes of main memory and six 2 gigabyte dual-headed barracuda disk drives striped across three controllers. All database monitoring and startup scripts were initiated from a client workstation.

Both configurations used the same optical disk system. The system consisted of eight 1.2 gigabyte drives, each one having its own SCSI ID. One controller was used for the optical disk system. The optical disks write at a rate of 300 KB/sec and can be read at a rate of 600 KB/sec. All transactions to and from the optical disk system are interfaced through an AMASS cache. The AMASS cache consisted of a 16 gigabyte RAID 5 system. The AMASS cache consists of 9 tunable cache blocks per open file/Oracle data file.

Both configurations used SGI's IRIX 5.3 UNIX operating system. The Oracle configuration consisted of Oracle7 release 7.2.2 of the DBMS, parallel query option and SQL*Net 2.2.

### Testing

The database contained two identical tables, T_ACTIVE and T_ARCHIVE, four indexes (two per table) and a single view, V_MENU. The view was a union of a common set of columns from the tables T_ACTIVE and T_ARCHIVE. Appendix 1 contains the DDL for the creation of these objects. The software designed to test the archive concept was composed of three parts.

> **Part I)** Loaded table T_ACTIVE, to a magnetic disk via a PL/SQL module with 10,000 rows, a sequence was used to ensure uniqueness. After the table was loaded, the PL/SQL code updated the inserted rows to randomly spread out the value of the column; 'changed_date.'
>
> **Part II)** A C program selected a number of rows from T_ACTIVE that met a date condition on the 'changed_date' column. The rows that met the condition were written to table T_ARCHIVE on an optical disk and its associated indexes were updated. The rows written to the optical disk were then deleted from T_ACTIVE. The tablespace that contains the archived table is usually maintained as a 'read-only' tablespace. Prior to running the second PL/SQL code, the tablespace in placed in 'write' mode. After the data to be archived was written, the tablespace was placed back in 'read only' mode.
>
> **Part III)** Issue a query that selects rows from the view V_MENU. The query was designed to select rows from both T_ACTIVE and T_ARCHIVE.

## Functionality Test

The first test used an Oracle configuration with 2 KB database blocks and all the default settings for the various database (init.ora) parameters. Minor tuning was conducted on the Oracle kernel to provide incremental improvements (i.e., increasing the number of rollback segments, etc.). The results of test one proved that the concept was viable, however, performance was unacceptable. The following test results are for the first test, a summary of the results for both tests are shown in Table 1.

**Loading the table T_ACTIVE --** 0:06.0 (read 6 minutes)
**Migrating data to optical drives --** 1:13.37
**Querying both active and archived data --** 0:01.13

The results of the functionality test precipitated discussions for the performance test. It was determined that the environment needed to be upgraded and tuned. The hardware architecture needed to be upgraded to increase main memory and add more disk drives. In addition, the Oracle database was not tuned on its initial installation (i.e., default settings were used) and significant improvement should be gained by altering a number of the adjustable parameters. The AMASS software and its associated cache have numerous adjustable parameters that were not fully utilized. It was determined that the performance test be conducted taking into account the following recommendations generated as a result of the functionality test:

- Rebuild the Oracle kernel with an increased block size
- Add memory to the SGI server and increase the number of database buffer blocks
- Separate indexes on the T_ARCHIVE table to a separate tablespace residing on a different optical disk
- Separate indexes on the T_ACTIVE table to a separate tablespace residing on a different magnetic disk
- Perform inserts with array processing
- Turn on read-ahead on the optical jukebox
- Tune the checkpoint interval on the database
- Have the SGI server dedicated to Oracle and not concurrently running the storage management software
- Initiate processing from a client workstation instead of a server

## Performance Test

The performance test was conducted on a new set of hardware as described above and through a client workstation. The Oracle instance was rebuilt with a 16 KB block size and larger redo log files and a larger temporary tablespace. With additional magnetic disks, separate data and index tablespaces were created leaving the Oracle software libraries on their own magnetic device. On the optical jukebox, two tablespaces were created, one for the data and one for the indexes. The tablespaces resided on different optical platters and thus different drives.

The first run of the performance test used the same software as in the functionality test. In this test, T_ACTIVE was first loaded with 10,000 records and then with 100,000 records to test loading times. The times to load T_ACTIVE were as follows:

**Load 10,000 rows --** 0:04.20
**Load 100,000 rows --** 0:09.29

The initial PL/SQL script used to load table T_ACTIVE was modified into a C program with no array processing and one commit point. This load took approximately 1 minute and 2 seconds (0:01.02). The code was then further modified by removing the database's sequence processing function and by not performing the update to the 'changed_date' column. The load under this scenario took 30 seconds (0:00.30). Since the test required various 'changed_date' values, the section of code used to vary the data in this column was replaced with Oracle's DECODE process. The resulting load with this modification took 50 seconds (0:00.50). The same modified code was used to load 100,000 records, this run took 6 minutes and 59 seconds (0:06.59). In terms of time, the two different loads were essentially linear, actually, the 100,000 record load was a bit more efficient than the smaller load. Comparing the 10,000 record loads between the functionality and performance test, the performance test showed an 86% improvement in load time. This improvement was attributed to application tuning and the hardware upgrade. Since the goal of this testing was to determine the feasibility of migrating data to optical disks, we accepted our loading results and shifted our attention to the migration process and did not revisit the loading process. The last load of T_ACTIVE was with 100,000 records. It is with this load that we tested the migration process. The results of the load portion of the performance test is as follows and an overall summary can be found in Table 1:

**Load of 10,000 rows --** 0:00.50
**Load of 100,000 rows --** 0:06.59

The migration PL/SQL code was run unmodified (i.e., same code as the functionality test) and took 44 minutes and 42 seconds (0:44.42). Instead of half the number of rows being migrated as in the functionality test, only a third of the records in the performance test were migrated. This difference was due to how we modified the loading script. It should be noted however, that we migrated 33,334 rows in the performance test vice the 4997 rows in the functionality test -- the results of this migration in itself was a significant improvement time per the number of rows loaded as shown below:

**Functionality Test Migration --** 67 rows/minute
**Performance Test Migration --** 750 rows/minute

The AMASS cache block was sized at 1 MB. Analysis of I/O monitoring suggested an increase of the AMASS cache block. Enlarging the block size should cause fewer writes to the optical disk system. A size of 64 MB was determined to be sufficient to continue testing. The migration process was rerun taking 5 minutes and 50 seconds to migrate 33,334 rows to the optical disk system. An 87% improvement from the previous run resulted.

**Migrate 33,334 rows with 1 MB AMASS cache block --** 0:44.42
**Migrate 33,334 rows with 64 MB AMASS cache block --**0:05.50 (5718 rows/minute)

Considering the modifications to the Oracle kernel and the AMASS cache, the migration results have improved substantially. The Oracle Trace and TKPROF utilities were then utilized to fine tune the process. Because the migration PL/SQL code included two select statements, both tables were altered to be parallelized with a degree of 8. The migration code was rerun with this modification with a resultant time of 5 minutes and 29 seconds (0:05.29). Reviewing the output of the TKPROF process, it was found that the temp tablespace was heavily utilized. As a result, we increased the Sort Size parameter to 10 MB and added 16 batch writes to the init.ora parameter file. The migration process was rerun with the resultant time of 5 minutes and 1 second (0:05.01).

Further investigation of the TKPROF output revealed that the 'SELECT COUNT(*) FROM V_MENU' SQL statements took on average 1 minute and 42 seconds to run. This SQL statement was run at the start and end of the migration process. This means that approximately 3 minutes and 24 seconds or 67% of the migration time was devoted to counting the number of rows in T_ACTIVE and T_ARCHIVE. This SQL statement has a GROUP BY clause which is part of the view and not the physical table and thus results in two full table scans and does not employ any indexes. Therefore, due to the inordinate amount of time and processing that was consumed when querying the view, we replaced the provided SQL statement with the following SQL statement:

```
select count (*), "Online"
from T_ACTIVE
UNION
select count(*), "Archive"
from T_ARCHIVE
```

The migration process was rerun with a resultant time of 1 minute and 51 seconds (0:01.51). The modification of the migration process produced another 63% improvement in run time. All of this was attributed to the two queries, before and after the migration, took a total of .5 seconds vice 3 minutes and 24 seconds. The final breakdown, in time, of the entire migration process is as follows:

| | |
|---|---|
| Alter Tablespace to read write -- | .13 seconds |
| SELECT COUNT(*) -- | .25 seconds |
| Select data to extract from T_ACTIVE -- | 29 seconds |
| Insert extracted data into T_ARCHIVE -- | 38 seconds |
| Delete extracted data from T_ACTIVE -- | 40 seconds |
| Alter Tablespace to read-only -- | .13 seconds |
| SELECT COUNT(*) -- | .25 seconds |

## Results

Reviewing the whole migration process, it was determined, that the migration from magnetic media to the optical media actually takes place when the optical media's tablespace is altered to read-only. When this occurs, the dirty pages in the SGA are written to the AMASS cache. At this point, the data is now archived. The actual write to the optical disk takes place when four of the nine AMASS cache blocks are full.

The results of the performance test were much more conclusive. The improved hardware and the extensive tuning of the Oracle kernel, AMASS system and the test software proved to be essential modifications for the test.

Note: The third part of the test (query using view V_MENU) was not conducted during the performance test. It was determined in the functionality test, that data on optical disk can be queried and subsequently selected along with data from magnetic media.

Table 1: Migration Test Results

| Operation | Functionality Test | Performance Test |
|---|---|---|
| Load of On-Line data table (10,000 rows) | 6 minutes | 50 seconds |
| Migration of 4997 rows from On-Line to Near-Line (optical) and deletion from the On-Line device | 1 hour<br>13 minutes<br>37 seconds | 1 minute<br>51 seconds<br><br>* migrated 33334 rows |
| Query of view (union of both On-Line and Near-Line tables)<br>-- After instance reboot | 1 minute<br>13 seconds | N/A (see Note below) |
| -- Data cached | 20 seconds | N/A |
| -- Flushed cache | 34 seconds | N/A |

## Considerations for employing the use of Long-Term Archival Technology

### (1) Write efficient application code and tune it for high performance!

Probably the most important aspect of the archive function is that the application and associated SQL and PL/SQL code be as efficiently written as possible. During the development phase, the SQL and PL/SQL should be subjected to numerous tuning exercises under varying conditions. Proper access to the database significantly reduces the run-time of the archival process as was indicated in testing.

Utilize the array interface when inserting a large number of rows into tables. Use to your advantage the decode statement to provide a way to avoid having to scan the same rows repetitively, or to join the same table repetitively.

After the code has been written, use the utilities provided by Oracle. Run the ANALYZE command on the tables you'll be querying. The ANALYZE command collects statistics about the tables and stores them in the data dictionary. Determine if the optimizer is selecting the most efficient access path for your SQL statements by running EXPLAIN PLAN. The EXPLAIN PLAN diagnostic statement gives you an inside look at how the optimizer is planning to process your SQL statement. The results of the analysis may provide the impetus to use hints in your SQL statements. Hints are a mechanism allowing you to manually tune individual SQL statements, overriding the optimizer's decisions for that statement by including your own optimization hints within the SQL statement.

Additionally, through the use of the parallel query option, you are able to scan intensive queries in a parallel fashion. Set the degree of parallelism close to the total number of disk heads on the drives containing the datafiles of the tables and indexes you are using during the archive process.

**(2) Strategic placement of database objects during physical database design!**

It is critical that the database objects (tables and indexes) associated with the archived data be placed on separate tablespaces. The tablespaces, if possible, should be accessed from different controllers. In addition, separate tablespaces for tables and indexes reduces the contention during the insertion of records and associated indexes. To further reduce contention, stripe the datafiles across multiple disks and controllers.

**(3) Tune the Oracle kernel (init.ora) for optimal performance!**

The Oracle kernel must be tuned to take advantage of the inherent tunable features of the Oracle database. The first parameter to be set should be the DB_BLOCK_SIZE. This parameter must be set prior to installing the database. For the performance test, we set this parameter to 16KB, we found that performance was enhanced when this was reset from its initial setting of 2KB. The following parameters should be set in the init.ora file:

OPTIMIZER_MODE -- COST

The OPTIMIZER_MODE parameter tells the query optimizer, when set to COST, to use the cost based optimizer. If you are selecting more than 10% of your data, it is optimal for Oracle to use a full table scan to satisfy NCUNITS is an integer used in combination with the MAXIOSZ parameter to define the AMASS cache block size. The cache block size is determined by the following equation:
your query. Couple this with Oracle's parallel query option, significant performance improvements will be gained.

SORT_AREA_SIZE -- 10 MB

The size in bytes a user process has available for sorting. Performance improvement can be substantial. Allocated on a per user basis.

ASYNC_IO -- TRUE

This will allow parallel disk writes and have the potential of increasing performance 20%.

DB_WRITERS -- 8

Ability to perform multiple database writes. At a checkpoint the master database writer determines which blocks in the database buffer cache need to be written to disk. The master database writer divides up the work and notifies slave database writers to write blocks. A good value would be equal to the number of disks containing data files.

DB_BLOCK_WRITE_BATCH -- 16

The number of blocks a database writer passes at one time to the operating system to write to different disks in parallel and to write adjacent blocks in a single I/O. A good value would be equal to twice the number of DB_WRITERS.

**(4) Tune the AMASS cache and associated system components for optimal performance!**

Significant performance gains will be achieved by properly tuning the AMASS cache. AMASS has configurable options broken into the following four areas: cache configuration, performance, jukebox scheduling and miscellaneous. In the test of the archive concept, we tuned parameters that affected the cache and performance areas. Tuning in the cache area proved to be the most beneficial for the purposes of this test.

MAXIOSZ maximum input/output size

MAXIOSZ is the size, in bytes, that AMASS uses internally to read data from and write data to the optical media. It is recommended that MAXIOSZ be kept as large as possible to achieve maximum throughput rates. Initially, this parameter was set to 1 MB and remained so throughout the test. This parameter is derived by the O/S vendors implementation of their scsi device driver.

NCUNITS number of cache units

NCUNITS is an integer used in combination with the MAXIOSZ parameter to define the AMASS cache block size. The cache block size is determined by the following equation:

**Cache Block Size = NCUNITS * MAXIOSZ**

The cache block is the basic unit used by AMASS to transfer data to and from the cache. When a file is read from or written to the AMASS file system media, it passes through the cache in cache block sized packets.

The initial test, had this parameter set to 1. It was subsequently set and maintained at 64. We found that a 64 MB cache block size increased the performance of the transfer to the optical disk .

NFNODES number of fnodes (file nodes)

The NFNODES parameter defines the number of files that can be open concurrently in the AMASS file system. This parameter is automatically calculated during AMASS configuration based on the number of cache blocks available in the cache disk.

The performance parameters define whether or not multiple cache blocks are read (READAHEAD) as a file is read and if more than one volume is a volume group can be written to at a time. For our purposes, we modified the READAHEAD parameter.

READAHEAD file read-ahead

The READAHEAD parameter is either set to enabled (1) or disabled (0). When enabled, AMASS will automatically read the requested block along with an additional three cache blocks of data from the AMASS file system media with every read request.

### (5) Plan the periodicity of the archival process!

The timing of the archival process should coincide during low level activities of the production system. If numerous tables are be queried and subsequently deleted from, this will result in an increased load to the production system. Logic calls for query and modification type activities of a batch nature to be conducted during relatively quiet periods of system inactivity.

### (6) Sizing of the Jukebox

The I/O bottleneck is directly related to the number of optical devices within your jukebox as related to Oracle users requests that do not have data in the SGA or the AMASS cache. Observed rates of 600 KB/sec on reads and 300 KB/sec on writes were observed and expected.

## Theoretical Considerations

The implementation of optical media in an Oracle database environment, to the knowledge of the authors, has never been attempted in a production environment. Therefore, it is necessary to test and evaluate various database activities within this new environment to verify all database functionality and administration features operate as they do in an all magnetic media environment. The customer proof of concept we conducted did not have the time to do a thorough investigation of all the different possible scenarios a typical IS organization may use. Following are some of the areas that need further investigation to fully ensure the viability of using long-term archival storage media.

### System Startup

The Oracle control file contains the names and locations of the datafiles which are opened and checked for integrity. Upon startup, Oracle opens and checks each datafile associated with a tablespace for database instance integrity. Therefore, for each tablespace space created, Oracle will cause the optical media to be loaded into the MO drive and verify the file's tablespace header information. Startup times will be slower the traditional Oracle disk instance.

### Archive Considerations

**Select & Insert (DML):** An Oracle implementation with optical disks is best suited for transactions that do queries and initial loading of data with inserts. Data inserted onto optical disks should be archived data which is typically static and will not be altered in the future. Queries on the archived data will successfully transfer data at the expected read rates of the optical media, 600 KB/sec. Inserts exceeds performance requirements due to the AMASS cache and write algorithms implemented within the AMASS cache I/O architecture.

**Update (DML):** AMASS does not handle dynamic column/row size updates the same way as a disk device. In archive architectures the media is formatted and the archive software can not update a used block on the media. The process for an update of a data block is to copy the block into memory, mark the current media block as unusable, in memory update the block with new results, then write the new memory block(s) to the optical media. The only case where a block is not marked invalid is when an update does not increase or decrease the data within a block.

Oracle database block **row chaining** and **row migration** should be avoided or further dead space issues on media will result.

### Performance

Writes are not limited by Oracle and AMASS, but enhanced, since Oracle allocates contiguous data blocks with in an extent. AMASS writes will be done contiguously since the creation of the tablespace within Oracle opens the datafile location(s) and allocates a contiguous set of blocks on media.

The AMASS I/O cache and Oracle database buffer cache both use an LRU algorithm which does not contend with the I/O among both software kernels. Sizing of the database buffer cache (shared memory) and AMASS cache (rdsk) are related when tuning and sizing the architectures to meet system throughput requirements.

Matching AMASS cache block size to Oracle's database block size did not significantly improve performance, rather having a larger cache block sizes proved I/O gains for the large insert transactions and full table scan queries. This results due to the manner in which AMASS handles its cache blocks for I/O.

Read times can be enhanced within Oracle by setting the initialization parameter, DB_FILE_MULTIBLOCK_READ_COUNT, to match your queries. Additionally, the AMASS architecture was tuned for read-ahead.

### Archive Technologies

The AMASS I/O architecture reads and writes data in blocks. Other archive architectures which use block I/O may work as well. Archive architectures that perform I/O using file format **may not** provide the same ease of use, ease of administration and good media space management.

### Indexes

Depending on queries and mechanical robotic movement, the use of indexes may reduce performance significantly. Using full table scans may be the correct approach. Further testing is necessary in this area. Functionally, indexes were proven to work. In our testing, and by the advice of database tuners, indexes should be created and placed on the same optical disks as its base table. Separation of indexes spreads the I/O request across media and removes media thrashing caused by having two data structures on the same media. Another alternative is to create the archived table's indices on magnetic media. This should result in improved performance by removing the extra robotic movement time of loading the index tablespace in the drive.

### Tape Technology

The use of tape technology was not tested but needs to be discussed. Optical testing was performed due to the customer's media life expectancy requirement. It is possible though, that tape technology could be used as the archival medium. Some of the factors that need to be considered and tested are as follows:

- The effect of Oracle startup and the effects of reading the header of each tablespace during the database integrity start up phase.
- Tape thrashing. Accessing a set of tables randomly, e.g., first table at end of tape, second in middle of tape. Additionally, user programming may be necessary when accessing table data in order to gain acceptable query and I/O transfer rates. User applications should consider querying tables in sequential order as they were written to tape.
- Use of indexes may not be beneficial at all, indexes would need to reside on disk.

### Robotic Hardware

Jukebox sizing is based totally on the user's requirements. Some of the hardware considerations are as follows:

- Response time
- Simultaneous insert transactions
- Simultaneous query transactions
- Off-line media management acceptability

## Conclusion

This exercise ***proved the concept*** of migrating data from magnetic devices to optical devices. Not only is it possible to do so, but performance-wise it is feasible. It was found that properly tuning the Oracle kernel, the AMASS cache system, system memory and I/O and the application code, the actual migration process is essentially a function of selecting requested data, inserting this data into another table and then deleting selected rows from the table on the magnetic media.

The prototype architecture, as shown in the proof of concept, provides the means to architect VLDB cost-effectively by using disk and optical storage devices using COTS products. This architecture should allow businesses to respond to customer needs by maintaining information on-line and near-online and having access to their data in a transparent manner.

## Acknowledgments

Vangard Technologies: Eric Eastman and Dave Donald
11211 East Arapahoe Road
Englewood, CO
800-840-6090
***Lab Environment and Optical Library***
***AMASS Documentation Suite***

Silicon Graphics Inc.: Liz Reynolds and Fred Beck
Denver, CO
***Challenge L hardware***

Emass Incorporated
10949 East Peakview Ave
Englewood, CO
800-654-6277
***AMASS Software***

Oracle Corporation: Joe Conway
Advanced Programs Group

## References

Oracle 7.2 Server Tuning
Oracle7 Server Concepts
Open Data Warehousing with Oracle 7 Parallel Data Management Technology

## Appendix 1

```
CREATE TABLE T_ACTIVE
(
MENU_NUM            NUMBER(9)
        CONSTRAINT CARMENUC_NUM_NN NOT NULL,
MENU_LABEL          VARCHAR2(2000)
        CONSTRAINT CARMENUC_MENU_LABEL_NN NOT NULL,
ADDED_USER_ID     VARCHAR2(15)
        DEFAULT SUBSTR(USER,1,15)
        CONSTRAINT CARMENUC_ADDED_USER_ID_NN NOT NULL,
ADDED_DATE          DATE
        CONSTRAINT CARMENUC_ADDED_DATE_NN NOT NULL,
CHANGED_USER_ID         VARCHAR2(15)
        DEFAULT SUBSTR(USER,1,15)
        CONSTRAINT CARMENUC_CHANGED_USER_ID_NN NOT NULL,
CHANGED_DATE     DATE
        DEFAULT SYSDATE
        CONSTRAINT CARMENUC_CHANGED_DATE_NN NOT NULL,
CONSTRAINT          IAMENU0
```

```
        PRIMARY KEY        (MENU_NUM)
        USING INDEX PCTFREE 5
        TABLESPACE ONLINE
        STORAGE (
                INITIAL 150K
                NEXT 150K
                PCTINCREASE 0
                )
)

CREATE INDEX IACTIVE_CHANGED_DATE ON T_ACTIVE
        (CHANGED_DATE)
        TABLESPACE AINDEX
        STORAGE (
                INITIAL 150K
                NEXT 150K
                PCTINCREASE 0
                )



CREATE TABLE T_ARCHIVE
(
MENU_NUM           NUMBER(9)
        CONSTRAINT CARMENUC_NUM_NN NOT NULL,
MENU_LABEL         VARCHAR2(2000)
        CONSTRAINT CARMENUC_MENU_LABEL_NN NOT NULL,
ADDED_USER_ID    VARCHAR2(15)
        DEFAULT SUBSTR(USER,1,15)
        CONSTRAINT CARMENUC_ADDED_USER_ID_NN NOT NULL,
ADDED_DATE         DATE
        CONSTRAINT CARMENUC_ADDED_DATE_NN NOT NULL,
CHANGED_USER_ID         VARCHAR2(15)
        DEFAULT SUBSTR(USER,1,15)
        CONSTRAINT CARMENUC_CHANGED_USER_ID_NN NOT NULL,
CHANGED_DATE    DATE
        DEFAULT SYSDATE
        CONSTRAINT CARMENUC_CHANGED_DATE_NN NOT NULL,
CONSTRAINT         IARMENUU0
        PRIMARY KEY        (MENU_NUM)
        USING INDEX PCTFREE 5
        TABLESPACE ARCHIVE
        STORAGE (
                INITIAL 150K
                NEXT 150K
                PCTINCREASE 0
                )
)

CREATE INDEX IARCHIVE_CHANGED_DATE ON T_ARCHIVE
        (CHANGED_DATE)
        TABLESPACE ARCHINDEX
        STORAGE (
                INITIAL 150K
```

```
NEXT 150K
PCTINCREASE 0
)
```