Oracle SecureFiles System

Niloy Mukherjee, Bharath Aleti, Amit Ganesh, Krishna Kunchithapadam, Scott Lynn, Sujatha Muthulingam, Kam Shergill, Shaoyu Wang, Wei Zhang

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

Niloy.Mukherjee@Oracle.com

ABSTRACT

Over the last decade, the nature of content stored on computer storage systems has evolved from being relational to being semi-structured, i.e., unstructured data accompanied by relational metadata. Average data volumes have increased from a few hundred megabytes to hundreds of terabytes. Simultaneously, data feed rates have also increased with increase in processor, storage and network bandwidths. Data growth trends seem to be following Moore's law and thereby imply an exponential explosion in content volumes and rates in the years to come. The near future poses requirements for data management systems to provide solutions that provide unlimited scalability in execution, availability, recoverability and storage usage of semi-structured content.

Traditionally, filesystems have been preferred over database management systems for providing storage solutions for unstructured data, while databases have been the preferred choice to manage relational data. Lack of consolidated semi-structured content management architecture compromises security, availability, recoverability, and manageability among other features. We introduce a system without compromises, the Oracle SecureFiles System, designed to provide highly scalable storage and access execution of unstructured and structured content as first-class objects within the Oracle relational database management system. Oracle SecureFiles breaks the performance barrier that has kept such content out of databases. The architecture provides capability to maximize utilization of storage usage through compression and de-duplication and achieves robustness by preserving transactional atomicity, durability, availability, read-consistent query-ability and security of the database management system.

1. INTRODUCTION

Traditionally, database management systems have been designed to provide maximum throughput of storage and access to relational data in transaction processing and data warehouse environments. The requirements primarily arose from the advent of e-commerce based enterprise applications during the mid-1990s. However, the rapid growth of the Internet has caused a huge increase in the amount of unstructured and semi-structured information generated and shared by organizations in almost every industry and sector. Current estimates by analysts show that eighty five percent of business information is semi-structured in the form of emails, research documents, presentations and news feeds [1]. Moreover, in recent years, multimedia applications accessing digital images and videos over the Internet, banking applications storing images of checks, geo-navigation applications accessing digital map data and scientific applications accessing mission-critical data are changing the dynamics of data ingestion in terms of both rate and volumes.

Analysts forecast semi-structured data volumes and rates to double year after year. Enterprise databases are expected to reach petabytes in sizes running on more than thousand core systems by the end of 2010 [2]. As data volumes and ingestion rates step up, a number of challenges arise in the area of database management. The challenges include provision for maximum throughput of storage and access operations, scalability, utilization of storage usage, highest degree of availability and security of critical data, and information lifecycle management of such huge data volumes.

Filesystems have been preferred over database management systems for providing storage solutions for unstructured data while databases have been preferred to manage accompanying relational data for indexing and querying purposes. While filesystems provide better throughput of storage and access operations, they lack secure database features such as atomicity, consistency, durability, manageability and availability. For example, a recent study [3] has found that most of the journaling filesystems do not provide the recoverability that one might expect from them. On the other hand, mission-critical corporate, scientific and financial applications that do not tolerate data loss corruptions heavily rely on enterprise database management systems that are not considered to provide filesystem like performance for unstructured data. Current and near future challenges in the area of semi-structured content management justifies the requirement for a consolidated industry-strength architecture and solution.

We present Oracle SecureFiles System, a novel data storage architecture within Oracle 11g database server [4] that aims to bridge the gap between unstructured and relation data management by providing clustered temporal filesystem-like or

PVLDB '08, August 23-28, 2008, Auckland, New Zealand

better throughput and scalability for unstructured content while preserving the same for relational content. Storage of content as first-class database objects within the Oracle database server preserves the rich set of data management features and benefits of the Oracle database server and hence provides a consolidated content management solution. Besides providing support for advanced database features, Oracle SecureFiles provides advanced filesystem features such as compression and de-duplication [6] for optimizing storage usage as well as encryption for maximum content security.

The remainder of the paper is organized as follows. Prior efforts in the area of consolidated content management are mentioned in the following section. Section 3 introduces Oracle SecureFiles System and provides a broad overview. Details of SecureFiles architecture are presented in section 4. The following section details the set of advanced database features inherited by SecureFiles. Sections on conclusion, contributing authors and acknowledgements follow Section 6, which presents description of in-house scalability and throughput experiments conducted on SecureFiles and their evaluations.

2. PRIOR WORK

Previous efforts in design and implementation of semi-structured content management can be classified in three broad categories. The first category consists of architectures that provide database solutions, where the database provides storage for unstructured content as first class objects. The second category consists of frameworks that provide hybrid filesystem/database solutions, storing unstructured content in filesystems and relational data in a database system to manage relational data and metadata maintaining transactional atomicity and backup for filesystem operations. The third category comprises of filesystem solutions, where filesystems provide frameworks that store both unstructured and relational data. The rest of the section details a few examples from each of the categories.

2.1 Database Solutions

Several database management solutions such as Oracle Database server, Microsoft SQL Server and IBM DB2 provide storage for unstructured content using LOBs (Large Objects) [7]. LOBs are database fields that provide underlying storage support for chunks of unstructured data as single file like entities within the database. LOBs aren't themselves a datatype, but rather a class of datatypes that consist of three ANSI SQL datatypes, BLOB (Binary Large Objects), CLOB (Character Large Objects) and NCLOBs (National Character Set Large Objects). We will use the term 'LOB' in this document to loosely refer to the concept of a 'Large Object' rather than a specific datatype. Data stored as LOBs do not share space with the accompanying relational table but are stored as separate first-class objects within the database. Being first-class database objects, unstructured data management using LOBs shares the rich feature set that comes with the database. In case of Oracle database server, LOBs provide support for capabilities like transactional atomicity, read consistency, flashback, consistent backup, and point in time recovery. However, the main disadvantage of LOBs has been the inability to provide filesystem-like performance in both throughput and scalability of execution. Sub-optimal performance for the storage and access of LOBs has resulted in semi-structured content management applications choosing filesystems as the primary storage option for unstructured data.

Olson et al implemented the Inversion Filesystem [8], a filesystem architecture built on top of the Postgres DBMS. Both relational and unstructured data is stored in Postgres DBMS. Storage and access occurs through POSTQUEL, the Postgres Query Language. However, Postgres is not SQL standards compliant and lacks advanced database features. The authors claim that their system could only achieve 30-80% of the throughput of a traditional network filesystem server backed by a non-volatile RAM cache. The authors do not provide performance comparisons eliminating the non-volatile RAM cache.

2.2 Hybrid Solutions

Architectural unification of databases and filesystems has been investigated in the past. Jim Gray along with Russell Sears and Catherine Van Ingen has investigated the possibilities of architectural confluence of filesystems and database systems [9][10]. They conducted experiments on storage and access of various file sizes using NTFS filesystem as well as SQL Server 2005 [11]. They concluded that file sizes less than 256 KB are efficiently handled by SQL Server using the BLOBs framework while NTFS is efficient for file sizes greater than 1MB.

There have been few attempts at design and implementation of filesystems providing database management features. Margo Seltzer et al [12] developed LIBTP, a UNIX library that provides basic transactional support functions for UNIX filesystems. Apart from transactional semantics, LIBTP does not provide other database features to the filesystem.

Gehani et al [13] designed and implemented OdeFS, a filesystem interface to an object data store. The design uses NFS clients to allow access to the OdeFS. The implementation of storage of objects in OdeFS is purely filesystem based and hence does not provide advanced data management features of the database.

Murphy et al [14] designed and implemented DBFS, a block structured user level filesystem that uses Berkeley DB as the backing store. Unstructured data storage in DBFS is built on top of the Berkeley Fast Filesystem rather than a database system that can make use of raw disk usage and performance. The authors chose Berkeley DB as the metadata store to primarily gain on performance. However, Berkeley DB is not SQL standards compliant and provides very only basic database features such as transactional support and recovery.

The Oracle database server provides a form a of hybrid solution by providing a datatype called a BFILE. A BFILE is a filesystem file that can be referenced and read from the database. BFILEs allowed applications to store data in a filesystem and still access it like it was a LOB. However, BFILEs have many restrictions. They cannot be written to or updated through the database. If updated from outside the database, there is no way of knowing from the database that the data has changed. If a file referenced by a BFILE is deleted, the BFILE must be maintained accordingly. Also, BFILEs are slower than accessing the filesystem file directly.

IBM DB2 provides support for the SQL compliant DATALINK type as a form of hybrid solution. The DATALINK type allows an application to reference a file external to the database and can be used like any other SQL data type to define columns in tables.

2.3 Filesystem Solutions

Among storage architectures that use filesystems for both unstructured and relational data management, Lustre [15] and Google File System (GFS) [16] are examples of distributed filesystems that aim to provide maximum scalability to meet data volume and ingestion requirements with provisions of fail-over and high availability through data replication. However, both Lustre and GFS are restricted to operate on top of Linux filesystems. In both cases, the degree of fail-over is proportional to the degree of replication and hence introduces trade off between data durability and utilization of storage usage.

Lustre can be integrated with ZFS file system [17] to provide data snapshots and copy-on-write transaction model. However, Luster combined with ZFS file system does not provide advanced data management features such as temporal query ability, metadata indexing, information lifecycle management, point in time recovery of data and query ability on standby systems. GFS uses BigTable [18], a distributed storage system for structured data, for relational data management. However, BigTable is proprietary in-house architecture used mainly for Google Applications. As in case of Lustre, GFS combined with BigTable also lacks the set of features provided by industry-strength database systems.

3. ORACLE SECUREFILES SYSTEM

In this paper we will give you an anatomy of the new Oracle Database 11g technology, Oracle SecureFiles, a next generation 'Database Solution' that represents a leap for databases into the area of high performance filesystems.

Oracle SecureFiles is a novel architecture that provides the most scalable execution of filesystem-like unstructured content and advanced filesystems features while preserving the rich features and benefits of the Oracle database server. Oracle SecureFiles stores data as first-class objects within the database and supports all types of content and all file sizes without compromising on the throughput and scalability aspects of performance. SecureFiles delivers comparable filesystem-like or better performance for basic reads and writes and provides maximum scalability of execution compared to traditional network filesystems. Oracle SecureFiles consists of highly optimized algorithms to scale performance up on single multi-core processor systems as well as scale out on distributed systems using Oracle Real Application Clusters. Being a component of the Oracle database server, SecureFiles can scale to store from petabytes to exabytes of unstructured data leveraging Oracle's Information Lifecycle Management Capability. Data can be managed with policies on multiple tiers of storage including SCSI, SATA or tape, without changing the logical view to the application.

Both database clients and filesystem clients can access data stored inside SecureFiles. Oracle SecureFiles is 100% backward compatible with all currently supported Oracle LOB APIs. These include, but are not limited to, SQL, PL/SQL, OCI, OCCI and Java. From the application/API perspective, there are no code changes required to take advantage of the improved performance of SecureFiles for a LOB based application. Apart from being SQL standards compliant, SecureFiles provides POSIX-compliant filesystem interfaces and data can be accessed through open data protocols such as HTTP, NFS and FTP using such interfaces. Oracle SecureFiles provides advanced filesystem compression to optimize utilization of cache and storage. Oracle SecureFiles also provides the option to detect duplicate file-level data and stores single instances of the redundant copies of file-like objects. The Oracle SecureFiles framework provides maximum security of semi-structured data by providing encryption to both relational and unstructured data using Transparent Data Encryption (TDE) semantics.

In addition to these advanced filesystem features, Oracle SecureFiles has been designed to provide basic Oracle database capabilities, such as atomicity, consistency, isolation and durability semantics on unstructured data management operations. Being part of the Oracle database server, the SecureFiles framework automatically inherits more advanced database features, for example, readability on standby systems, consistent backup, point in time recovery, XML indexing and XML queries using the Oracle XML DB framework [26], and information lifecycle management using the Oracle Partitioning technology.

Oracle SecureFiles introduces the concept of 'delta updates' that enables non-length-preserving updates of file-like objects without undergoing rewrites or overwrites of object data blocks. This is a major 'feature' differentiator with respect to filesystems or LOBs performing updates through length preserving overwrites.

Oracle SecureFiles is supported by Oracle Flashback Archive [19] to provide temporal management and query ability of unstructured and relational data [20] through complete historization of data. This serves as a major 'feature' differentiator with respect to current industry-strength file systems as well as databases.

Michael Stonebraker et al mention that implementation of a specialized solution to any given performance problem can result in 10x gain over a general implementation [21]. Managing files in the database has been an area of weakness in databases, as has been commonly documented. SecureFiles is the next generation of file technology, built into the database, to make them perform as fast as filesystems and scale as well as the high performance filesystems. SecureFiles have been tested in-house as well as by external customers such as National Ignition Facility at the Lawrence Livermore National Laboratory to match the performance of the hardware that they run on [22].

4. SYSTEM ARCHITECTURE

We present the architecture details of the Oracle SecureFiles System in this section. The section proceeds with the description of the structural design of Oracle SecureFiles System. Details of the design and advanced filesystem features follow next. The section concludes with a detailed discussion of methods incorporated in the design to extend secure database semantics such as transactional atomicity, read consistency, and data durability to unstructured data management operations in the system.

4.1 SecureFiles Schema

The structural design of SecureFiles is similar to that of filesystems. Figure 1 describes the structural components of Oracle SecureFiles.

Unstructured data associated with semi-structured content is stored as SecureFile objects. A SecureFile object is a collection of variable sized chunks allocated from and stored in the Oracle

database using the Oracle SecureFiles. Each chunk is a set of contiguous database blocks.

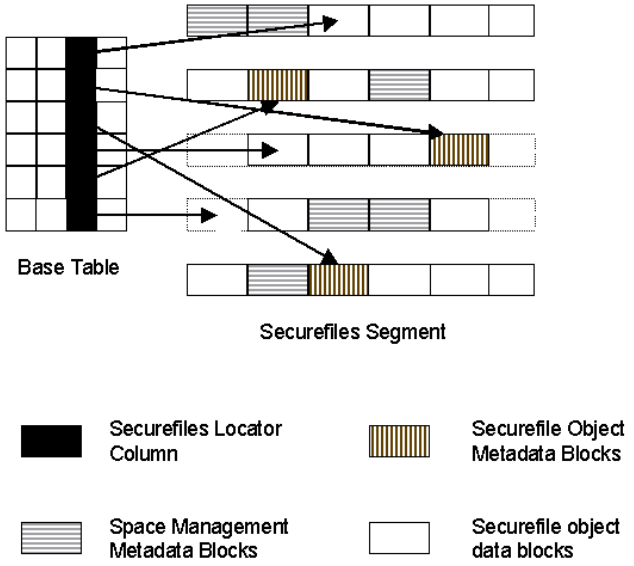


Figure 1. Structural Layout of SecureFiles

The base table is a typical Oracle table object that stores relational metadata associated with SecureFile objects. Besides the columns containing relational metadata, the table consists of one or more columns that hold locators providing reference pointers to the associated SecureFile objects. Each row-column intersection provides a distinct pointer to the very first block of an individual SecureFile object. Users can create unique and secondary indexes on the relational columns in the base table. Apart from being a placeholder for the locator, the row-column intersection contains metadata associated with the characteristics of the underlying SecureFile object, for example, the length of the file, whether the file is compressed, encrypted or de-duplicated. The row-column intersection also stores the starting block addresses and lengths of the first few chunks for the underlying data if sufficiently small.

The set of SecureFile objects contained in a single base table column shares a pool of free space within the database. Space is not shared across multiple SecureFiles columns to prevent complexities in load balancing and manageability. The free space pool is defined as a SecureFiles segment. Each SecureFiles segment is a collection of Oracle database extents that are contiguous ranges of data blocks. Logically, a SecureFiles segment consists of blocks that contain metadata for space management and blocks that are part of SecureFile objects. The first block of a SecureFile object may consist of an array of starting block addresses and lengths of the chunks if the number of chunks exceeds the maximum number of chunk metadata that can be kept in the row-column intersection. A SecureFile object may contain more than one such metadata block depending on the size and fragmentation of the object. Details about SecureFiles segment and object metadata management will be described in the next subsection.

4.2 Components of SecureFiles Architecture

Oracle SecureFiles architecture is layered into six major components, namely, Delta Update, Write Gather Cache, Transformation Management, Inode Management, Space Management and the I/O Management as shown in figure 2. The rest of the subsection details the design of each of these components as well methods contributing towards overall execution and scalability performance of the Oracle SecureFiles System.

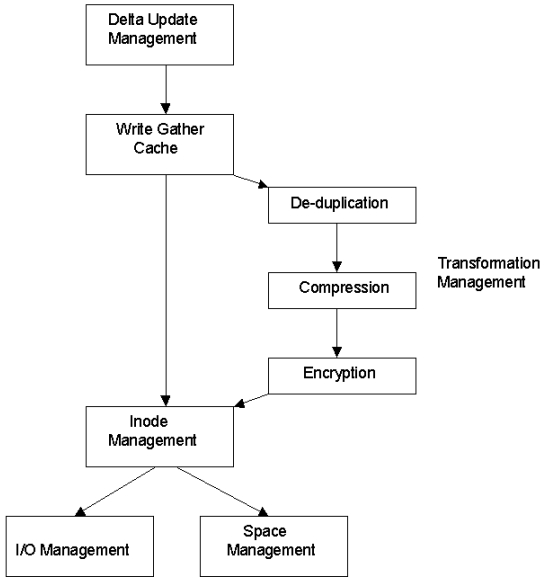


Figure 2. Architecture of SecureFiles

4.2.1 Delta Update

Updates to objects in filesystems as well as databases require rewrites of portions of objects that preserve length of the updates. The rewrites therefore span random lengths from being single bytes to being full object writes. This causes huge waste of I/O bandwidth even for small updates and affects performance for such operations. The Delta Update component provides the capability of non-length-preserving update operations on Oracle SecureFile objects through 'delta updates'. The feature provides special APIs to the user to specify the object to update, list of delta (content change), the length of the delta, and the start offset and end offsets to replace in the object and ensures I/O cost of update operations to be linear to the size of the delta.

The delta update component maintains its own metadata structures to record the mapping of source and destination offsets for each of the deltas. Queries on delta-updated objects first access the metadata structures to determine the correct offsets of deltas. Changes to metadata during delta update operations are transactionally managed, which guarantees correctness of query.

The delta update component provides substantial benefits in performance of XML storage frameworks such as Oracle XDB that depend internally on the performance of document inserts and updates by the underlying data management system.

4.2.2 Write Gather Cache

The Write Gather Cache (WGC) is a subset of the database buffer cache private to Oracle SecureFile object data. The gather cache can buffer large amounts of SecureFiles data up to a user-specified parameterized value during write operations before flushing or committing to the underlying storage layer. This buffering of in-flight data allows for large contiguous space allocation and large contiguous disk I/O. Write performance is greatly improved due to reduced disk seek costs. The WGC is allocated from the buffer cache and maintained on a per-transaction basis.

4.2.3 Transformation Management

The advanced data transformation management comprises of three subcomponents, compression, encryption and de-duplication. Oracle SecureFiles provide the option to enable/disable all possible combinations of these features.

De-duplication: Oracle database server automatically detects duplicate SecureFiles, and stores only a single physical copy on disk, thereby minimizing space usage. For every SecureFile object that has de-duplication enabled, a secure hash is generated for a subset of the object (prefix hash) and also for the whole object (full hash). During streaming writes, once generated, the prefix hash is compared to a set of prefix hashes stored in an index. If there is a prefix match, then the SecureFile object associated with the original prefix hash (master version) is read and byte-by-byte comparison is performed across the buffered data and the master version. At the end of the write, if the full hash matches and the full object matches on a byte-by-byte basis, then a reference pointer directing to the master version is maintained in the row column intersection. In this case, there have been no writes since the object is not materialized to disk. In the case of mismatch during comparison, then write gather cache buffers are communicated to the inode management component for materialization, and a new full hash will be inserted into the lookup index. Updates on a de-duplicated SecureFile object results in the materialization of the object on disk.

Compression: Oracle automatically detects if SecureFile object data is compressible and compresses using special multiple file compression algorithms. If compression does not yield any savings or if the data is already compressed, SecureFiles will automatically turn off compression for such objects. Compression is performed when buffered contiguous data exceeds a configured boundary threshold. These compressed data chunks are referred to as compression subunits. Multiple contiguous compression subunits are encompassed within a larger unit whose size is determined by the WGC flush threshold. Compression is performed piecewise in such a way that efficient random access of large files is possible. Compression not only results in significant savings in storage but also improves performance by reducing I/O sizes, database buffer cache requirements, data logging for media recovery, and encryption overheads. SecureFiles compression allows for random reads and writes to SecureFile data. Oracle SecureFiles architecture provides varying degrees of compression that represent a tradeoff between storage savings and CPU costs. Compression and De-duplication technology is designed with large RAM and Flash memory trends of the future in mind by enabling significantly better effective utilization of memory.

Encryption: Oracle SecureFiles uses Transparent Data Encryption (TDE) syntax for encryption of SecureFile objects along with the accompanying relational metadata. Oracle SecureFiles supports automatic key management for all SecureFile columns within a table and transparently encrypts/decrypts data as well as redo logs. SecureFile object buffers are encrypted/decrypted on database block size units. Oracle SecureFiles supports the following encryption algorithms: Triple Data Encryption Standard with a 168-bit key size, Advanced Encryption Standard with a 128 bit key size, Advanced Encryption Standard with a 192-bit key size (default) and Advanced Encryption Standard with a 256-bit key size.

4.2.4 Inode Management

The inode management layer is responsible for initiating on-disk storage and access operations on SecureFile object buffers being communicated by the upper layers in the SecureFiles architecture. As a client of the space management layer, the inode manager requests for on-disk free space to store the amount of data being flushed by the write gather cache. Based on the array of chunks returned by the space management layer, the inode manager stores the metadata either in the row-column intersection of the base table associated with the object, or in the most current header block of the SecureFile object. The metadata information includes start block address and length of a chunk as well as the start and end offsets of the object being mapped by the chunk. For compressed SecureFile object, the metadata structure is also used to map logical offsets to physical offsets on disk. The metadata structures are transactionally managed and are recoverable after process, session and instance failures.

SecureFile objects maintain inodes independent of each other. This prevents single points of contention in concurrent environments during update, delete and append operations on SecureFile objects. Metadata maintained in the inode can remain extremely compact because the space management layer provides the support to return a set of variable sized chunks, each scaled up to 64M each, to store the data being written to disk. The metadata management structures can therefore scale to map terabyte-sized objects very efficiently.

4.2.5 Space Management

The space management layer is responsible for allocating free disk space to SecureFile objects and de-allocating used space from SecureFile objects to the SecureFiles segment on disk keeping the real density and seek amortization trend in mind. The space management layer supports allocation of variable sized chunks. With SecureFile objects being cached in the Write Gather Cache, the space management layer is able to meet larger space requests from the inode manager through more contiguous layout on disk, therefore providing more efficient read and write access.

The space management layer is also responsible for managing SecureFiles segments. A SecureFile segment comprises of a collection of database extents that are spread across the entire underlying storage system. A specialized background process manages the growth of the segment based on heuristics of recent allocation demands on the segment.

Figure 3 demonstrates the flow of free space in Oracle SecureFiles system. Extents allocated to SecureFiles segments are first pre-split into chunks using methods that preserve the benefits of the WGC. The metadata associated with the chunks are stored

on various on-disk metadata blocks (Committed Free Space Blocks or CFS) in the segment. The CFS blocks are hashed on chunk sizes and meet space requests on a best-fit basis. The free space mapped by these blocks is shared across all the instances in a distributed Oracle Real Application Cluster environment. Based on allocation demands per instance in the cluster, the free space is distributed to each of the instances. Each database instance thereafter creates an in-memory dispenser in the database cache. The in-memory dispenser contains metadata mapping all free space assigned to the instance and meets future variable-sized space allocation requests from that instance. The dispenser scales with the concurrency of processes requesting allocation and performs in-memory operations during space allocations. As the in-memory dispenser runs out of space, the in-memory metadata cache is invalidated; check pointed to disk and recreated with new metadata entries. Although, the dispenser operations are in-memory, algorithms have been developed to support consistency of metadata after process, transaction or instance failures between successive checkpoints.

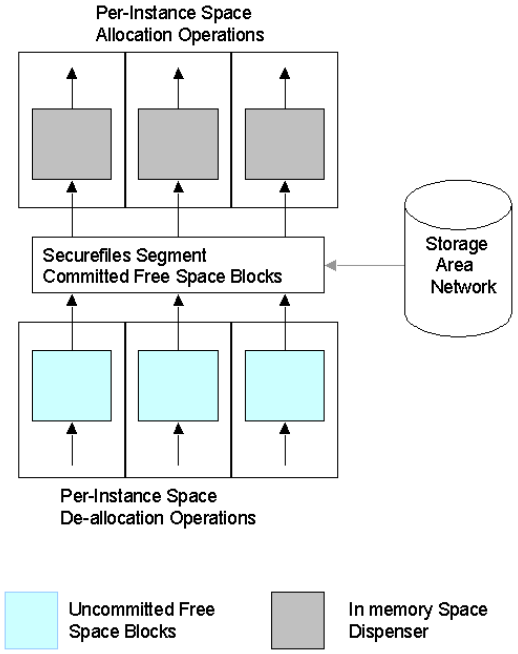


Figure 3. Space Management Architecture – Flow of Free Space

Operations such as full overwrites / rewrites, updates and deletes of SecureFile object follow 'copy-on-write' semantics resulting in de-allocation of space previously occupied by the offsets affected by the operation. In such cases, the space management metadata entries are inserted back to a different set of on-disk metadata blocks (Uncommitted Free Space Blocks or UFS) while maintaining transactional atomicity. The background process coalesces the free space entries if possible and moves them from UFS to CFS blocks if and only if the transactions associated with the generation of these free space entries have committed. The free space entries are transferred to the in-memory dispensers as and when requested.

4.2.6 I/O Management

The I/O Management Layer is responsible for satisfying I/O requests during reads and writes of SecureFile objects. During writes, the Inode Manager communicates the set of chunks obtained from the space layer as well as the write gather cache buffers to the I/O Manager. Based on a user parameter, the I/O Manager either copies the write gather cache buffers to database buffer cache buffers or schedules asynchronous disk writes for the set of chunks. The I/O Manager tries to further coalesce the set of chunks to optimize disk throughput. Transactions undergoing such writes do not commit as long as there are pending asynchronous I/Os

The I/O Manager supports read-ahead or pre-fetching data from disk. It keeps track of access patterns of SecureFile objects and issues intelligent pre-fetching of chunks before the request is actually made. Read latency is reduced by overlapping the network and storage throughput.

The design of the components in Oracle SecureFiles architecture enables the system to provide filesystem-like throughput and scalability of execution of read and write operations on objects of various sizes and types. Section 6 presents a few test cases that evaluate SecureFiles scalability and performance on reads and writes on a range of file sizes and types.

4.3 Extending Database Semantics to Oracle SecureFiles

The inode, space and I/O management components contain methods that assist in providing the essential relational database management features such as transaction atomicity, read consistency and data durability for unstructured data management in Oracle SecureFiles. The rest of this subsection details the methods implemented for each of the features

4.3.1 Transaction Atomicity

Oracle SecureFiles guarantees transaction atomicity on all filesystem-like operations on SecureFile objects.

Relational data associated with SecureFile objects is managed using the transaction semantics associated with the relational database kernel. The database kernel implements these semantics by generating undo records for all data and metadata operations. The undo records are stored as first-class objects within the database and are used to roll back database operations during failures thereby maintaining transactional consistency in the database.

Unstructured data manipulation operations in Oracle SecureFiles are responsible for modification of metadata maintained in inode and space management layers as well as inserts and updates of SecureFile objects. Metadata modifications follow transaction semantics similar to the methods described above. SecureFile objects undergo 'copy on write' semantics for larger update and overwrite operations. Such a semantic alleviates the requirement to store previous object images, partial or entire, for rollback purposes. However, small-size length-preserving updates on SecureFiles object data in ranges of few hundred bytes generate undo records for transaction atomicity purposes thereby preventing unnecessarily large I/O sizes. A single transaction can comprise of one or both forms of updates based on the length-preserving sizes.

When a transaction aborts, the relational metadata associated with SecureFile objects, inode metadata and space metadata roll back using the undo records. As a result, the SecureFile object locators point to the previous image of the inode metadata blocks that in turn point to the previous versions of the objects. Rollback of in-place updates of SecureFiles objects applies undo records to generate consistent previous data images. The rollback of space metadata prevents space leakage by freeing up space requested by the aborted transaction. Because of 'copy-on-write' semantics for large updates and overwrites, the rollback is not required to perform additional I/O to restore the previous object images. As a result, transaction recovery becomes independent of the sizes of the changes on the SecureFile objects

4.3.2 Read Consistency

Oracle database server supports multi-version read consistency for relational data. Queries retrieve data by re-creating snapshots of modified data blocks as of the time of their issuances. The snapshots or versions of relational data blocks are created through application of undo records that were generated during data manipulation operations.

Oracle SecureFiles supports similar read consistency semantics. While accompanying relational metadata in base tables and indexes use the above techniques to achieve read consistency, SecureFile objects achieve this through the following mechanism.

As has been mentioned in the previous subsection, data manipulation on SecureFile objects follow 'copy-on-write' semantics. The space management component maintains chunk metadata associated with object updates and deletes. The space freed during the update and delete operations map to old versions of data. The space management component retains such freed up space for a user-specified amount of time. Depending on the expiration of the retention period, the space management component either retains such space or reuses them for future allocations. Therefore, a query or a read operation issued on a SecureFile object at a point in time within the retention period is guaranteed to return the most consistent version of the object as of that point in time.

4.3.3 Data Durability

Oracle SecureFiles System design provides a range of data durability options. The I/O management component provides choice to the users to either use the database buffer cache to stage writes on SecureFile object buffers or to use the underlying storage for direct writes of SecureFile object buffers. The accompanying relational data, inode metadata and on-disk space metadata changes modify Oracle data blocks in the buffer cache itself and are logged in Oracle Redo logs. Metadata changes in the in-memory space management structures are not logged. However, such changes are recoverable after database instance failures thus guaranteeing consistency of on-disk space management metadata structures.

User data in SecureFile objects that are directly written to disk within transaction boundaries need not be separately logged into Oracle Redo Logs to achieve data durability across transaction and instance failures. However, users may choose to log operations on SecureFile object data for recovery from media failures and Point in Time Recovery purposes.

SecureFile objects that use the database buffer cache are required to log data operations in Oracle Redo Logs to achieve user data durability across transaction and instance failures. Relational data operations associated with SecureFile objects as well as inode and space management operations during data manipulation are always logged to keep the database consistent across failures.

5. ADVANCED FEATURES

Being stored as first-class objects within the database, Oracle SecureFiles System automatically inherits most of the current advanced data management features provided by the Oracle relational database server that are not provided by traditional filesystems, namely, temporal data access, readability in standby databases, secure backup, point in time recovery of SecureFiles data, usage in Real Application Clusters environments, and information lifecycle management. Apart from that, the system is also inherently capable to provide support for storage on low-cost flash devices in future.

5.1 A Temporal File System

Oracle Flashback framework is an extension of read consistency to provide capabilities to database users to query, retrieve as well as recreate relational data consistent as of any point in time in the past. The framework presented the concept of retention of undo records even though transactions generating them have committed. The current version of Oracle provides advanced enhancements called Flashback Archive that enable users to retrieve and recreate data as of several years before.

Oracle SecureFiles extends read consistency as described in the previous subsection to support Oracle Flashback and Flashback Archive features. Database users can set retention periods for SecureFile segments. If not explicitly specified by a user, previous versions of SecureFile objects are retained as long as their accompanying relational metadata is retained. This ensures consistency of SecureFiles data retrieval at any point in time as long as the accompanying relational data can be retrieved. SecureFiles with Flashback Archive provide a tamper-proof filesystem to applications that have many practical uses in the area of data security.

5.2 Data Retrieval in Standby Systems

Physical standby databases in Oracle function as standby systems which gather and apply Oracle Redo logs from current active database systems to produce a consistent snapshot of the entire database at a point in time so that they can replace the primary systems in case of failures. In previous versions of Oracle database server, users were not able to perform data management operations on standby databases.

The current version of Oracle database server provides the capability to query and retrieve database objects from physical standby database systems. Being first-class database objects, Oracle SecureFiles support query-ability of both unstructured and relational content on standby database systems if data manipulation operations on SecureFile objects are logged in the Oracle database Redo logs.

5.3 Secure Backup and Point in time Recovery

Oracle provides the option to encrypt data during database backup thereby retaining confidentiality of the backed up data. Being first class database objects, encrypted backup of the database system

ensures encrypted backup of SecureFile objects as well as accompanying relational data.

Oracle provides the capability to perform point in time recovery of the database by restoring backed up database and applying operations logged up to the chosen point in time in the Oracle database Redo logs. Point in time recovery can be performed on SecureFile objects if users choose to record manipulation operations on SecureFile object data.

5.4 Real Application Clusters

Real Application Clusters or RACs allow multiple instances of an Oracle database across multiple server systems to share access of the entire underlying disk subsystem staging the database. Apart from providing maximum availability and fail-over capacity in the database as all servers in the cluster have the capability to access the entire database, RAC provides opportunities for maximizing scalability of execution of database operations.

Oracle SecureFiles inherit the capabilities provided by Real Application Clusters. The design of the space management component in SecureFiles is tuned to provide scalability in throughput proportionally with the number of active database instances.

5.5 Information Lifecycle Management

Information Lifecycle Management has evolved as an important component in the area of content management. Growth in data volumes has posed requirements to monitor evolution lifecycle of data. Based on access activities, content management applications are required to migrate data between several storage devices in its lifetime to optimize storage utilization and cost.

Partitioning technology built in Oracle database server is a powerful tool to achieve effective lifecycle management of data. It enables applications to partition data based on various parameters. Applications can monitor temporal access patterns on such segments and may choose to migrate such segments from high-end storage devices to low end archiving devices.

Oracle SecureFiles makes use of similar partitioning techniques to achieve lifecycle management of SecureFile objects. Partitioning of base tables containing the relational and SecureFile metadata result in partitioning underlying SecureFiles segments. SecureFiles partitions can be independently migrated across storage devices.

5.6 Storage on Flash Devices

The exponential growth of flash memory has initiated active research to make use of such devices for database and filesystems storage [23][24]. Due to inherent properties of flash memory [25], these devices are optimized to support storage frameworks that simulate log-structured filesystem [24] like behavior.

The Oracle SecureFiles architecture provides a variant of a log-structured filesystem. Write operations on Oracle SecureFile objects follow 'copy on write' semantics, thereby preventing in place update operations on the storage system. Reclamation of storage space allocated to previous versions of SecureFile objects is based on several retention policies. The space management framework in Oracle SecureFiles assists the architecture to adapt to storage on flash devices. With current implementations of flash devices equipped with optimal wear leveling, content management on flash devices becomes highly feasible with Oracle SecureFiles.

6. PERFORMANCE EVALUATION

In this section, we present a set of benchmarks and performance evaluation data to showcase the impact of Oracle SecureFiles architecture on throughput and scalability of storage and access of semi-structured content. We compare performance of SecureFiles with a traditional network filesystem. We also demonstrate the scalability of SecureFiles on various content types and sizes in a single instance Oracle database as well as Real Application Clusters. Comparison with existing LOB architecture is out of scope for this paper.

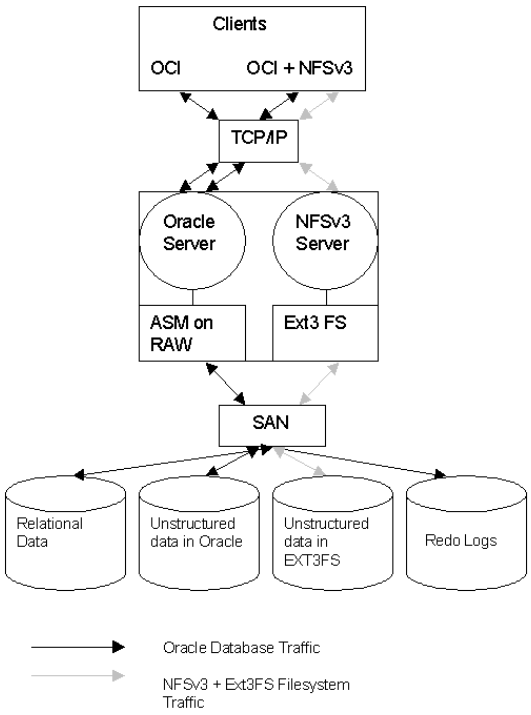


Figure 4. Hardware Setup

6.1 SecureFiles vs. Filesystem

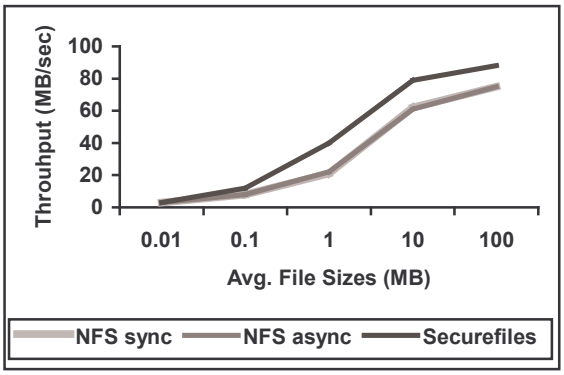
The experiment simulates a real world DICOM application consisting of digital diagnostic images accompanied by patient metadata. We compare read and write throughput of SecureFiles to that of an NFSv3 filesystem. In both cases patient metadata is stored in the Oracle database. In the case of filesystem, the images are stored on Ext3 FS file servers that are accessed using NFSv3. In case of SecureFiles, images are stored as SecureFile objects within the database.

6.1.1 Dataset and Configuration

The dataset consists of images ranging from 10kb to 100 MB. The experiment consists of tests individually run on sizes averaging 10 KB, 100 KB, 1 MB, 10 MB and 100 MB. For tests on sizes 100

MB, 10MB and 1 MB, the total amount of unstructured data inserted as files as well as SecureFile objects is 100 GB. For tests on size 100 KB, the total amount is 10GB and for size 10 KB, the total of data used is 1 GB. For sizes of 100 MB and 10 MB, write transactions are committed after every insert. For file sizes of 1 MB, 100 KB and 10 KB, write transactions are committed every 100 inserts to avoid test source-code specific overheads.

We configured Oracle SecureFiles to directly issue I/O to the underlying storage system without using the database buffer cache. We also set Oracle SecureFiles to enable logging of metadata and relational operations to Oracle Redo logs while disabling logging of operations on data. The setting is similar to filesystems that provide metadata journaling and disables recovery from hardware failures. However, the setting is more advanced compared to NFSv3 as it provides transactional atomicity, read consistency and data durability semantics across database failures. Multi-stream experiments were configured to avoid conflicts on the same sets of rows of the base table across processes.



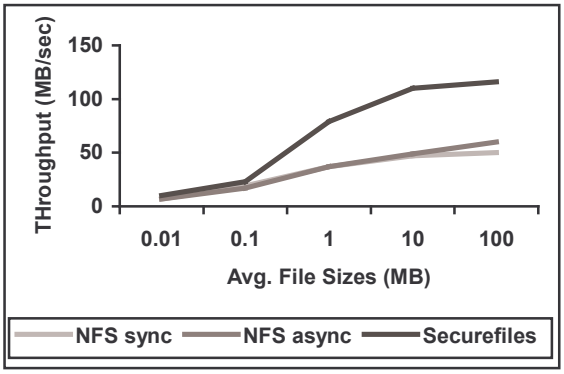


Figure 5. Throughput comparisons between NFS and Oracle SecureFiles: single process reads

Figure 6. Throughput comparisons between NFS and Oracle SecureFiles: four processes reads

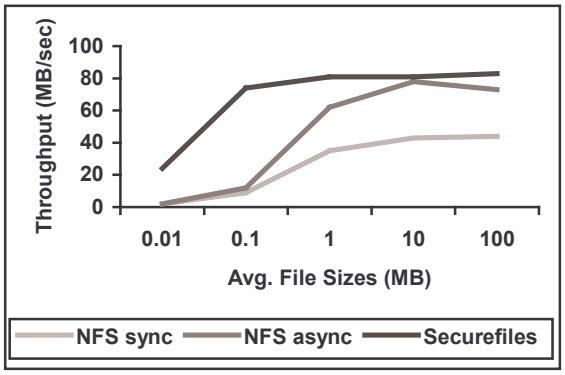
6.1.2 Setup

Figure 4 describes the hardware setup for the experiment. The setup consists of a client and a server connected across a gigabit Ethernet.

The Dell 2650 client machine consists of 2 hyperthreaded Intel Xeon 2.8 GHz processors with 0.5 MB processor cache each, 6GB of RAM and uses Red Hat Enterprise Linux 4.0. The Dell 2850 server consists of 2 hyperthreaded Intel Xeon 3.2 GHz processors with 2 MB processor cache each, 6GB of RAM, Red Hat Enterprise Linux 4.0 and 2Gbit Fiber Channel SAN Host Adapter. The client machine uses OCI interface and TCP/IP to communicate with the server for SecureFiles operations as well as database metadata operations for filesystem experiments. NFSv3 client is used for filesystem operations. Storage drives are allocated as two identically configured 2TB Raid 5 arrays. One of the units was allocated to Oracle and was managed using Oracle Automatic Storage Management. The other was configured as Ext3 FS made available to the client using NFSv3.

6.1.3 Experiment and Results

Single and multi-stream tests were performed to showcase both performance and scalability. The best performing options for NFSv3 (async and rwsize of 32KB) were used. Figures 5-8 demonstrate the throughput ratios between SecureFiles and NFSv3 on the tests performed.



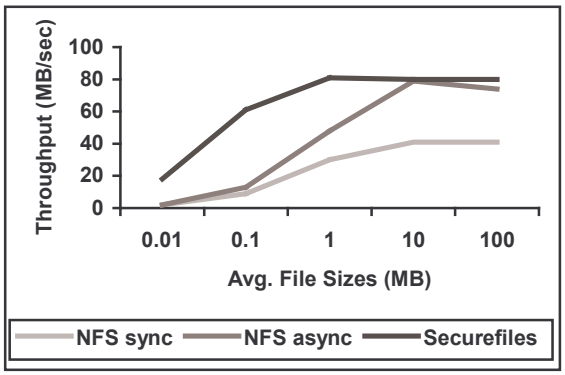


Figure 7. Throughput comparisons between NFS and Oracle SecureFiles: single process writes

Figure 8. Throughput comparisons between NFS and Oracle SecureFiles: four processes writes

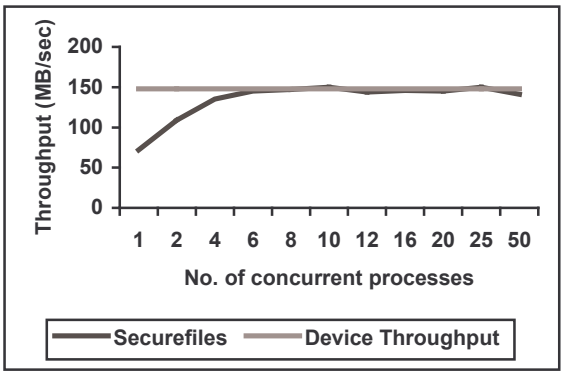
SecureFiles outperforms the NFSv3 access for all sizes with respect to read performance. Gains for the smaller file sizes are also due to reduced roundtrips where metadata and data is accessed in one roundtrip unlike the NFSv3 case where metadata and file data are accessed in separate roundtrips. Read performance for larger file sizes is contributed by intelligent pre-fetching, larger I/O sizes due to better contiguous space

allocations and network optimizations. SecureFiles outperforms the NFSv3 access for all sizes in write operations. Again as in the read case, for small file sizes the NFSv3 case has the overhead of writing metadata and file separately in two roundtrips. This demonstrates the improvements due to the write gather cache, larger contiguous I/O through in-memory space allocation and space pre-allocation optimizations. For smaller file sizes the throughput is limited by the roundtrip overheads and the disks are not utilized completely. Increasing the number of concurrent threads ensures increased throughput. For larger file sizes the throughput is limited by the network and I/O subsystem.

6.2 Scalability on SecureFiles on Single DB Instance

The test setup configuration for single instance scalability experiments is similar to that described in subsection 6.1.1. Oracle SecureFiles was configured to directly issue I/O to the underlying storage system for SecureFile object data bypassing the database buffer cache. Hardware setup as described in Section 6.1 was used for this set of experiments.

The scalability experiments were performed on three distinct sets of data. The first dataset consists of 1.5 GB of documents, each ranging between 60KB to 120 KB. The second dataset consists of 10GB of digital images, each ranging between 700 KB to 1.2 MB. The third set consists of 100GB of digital multimedia video files, each ranging between 60MB to 100MB.



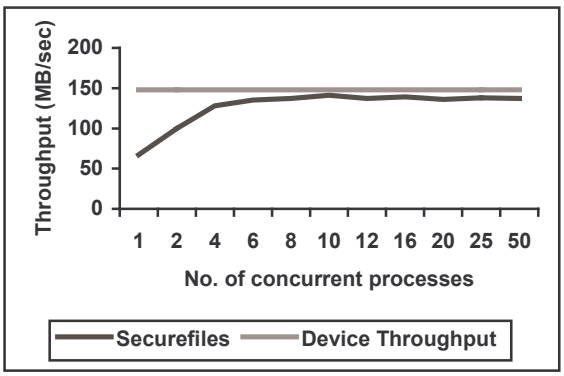


Figure 9. Average Throughput of Oracle SecureFiles on Document Archiving Workload

Figure 10. Average Throughput of Oracle SecureFiles on Document Modification Workload

6.2.1 Experiments on Documents Dataset

We simulated two types of document management applications on the dataset. The first application involves insert-only operations on the entire dataset for archiving the documents as SecureFile objects. Degree of concurrency of inserts was varied from 1 to 50 streams. The second application simulates modification operations on archived documents. The simulation consists of two phases. The first phase deletes 50% of the documents in random order and then re-inserts the documents. The second phase deletes and re-inserts the other 50% of the documents. Degree of concurrency of inserts was varied from 1 to 50 streams in both the cases. Multi-stream configurations were implemented to avoid conflicts on the same sets of rows across processes. Average throughput of inserts was measured for both the applications.

As evident from the figures 9 and 10, throughput of insert execution for both simulated applications scale up with the degree of concurrency on the hyper-threaded dual process server machine. Throughput of Oracle SecureFiles reach approximate hardware throughput at concurrency of 6 streams and converges thereafter with the device throughput in both cases. The hardware throughput was estimated using 'dd' Unix program. The average throughput for the archiving application is around 140MB/sec, thereby supporting data ingestion rate of approximately 1500 documents per second. The random insert-delete workload results in an average throughput of approximately 130 MB/sec, thereby supporting data ingestion rate of 1400 documents per second.

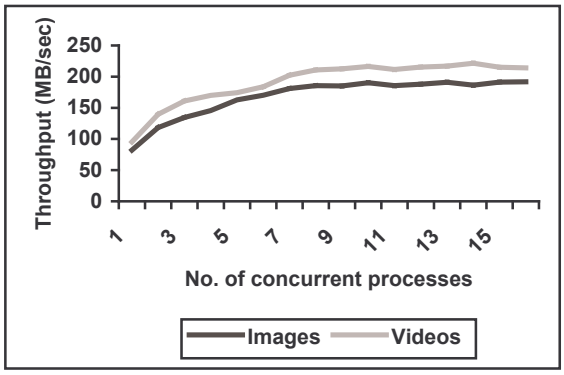


Figure 11. Average Throughput of Oracle SecureFiles on Image and Video Datasets

6.2.2 Experiments on Image and Video Dataset

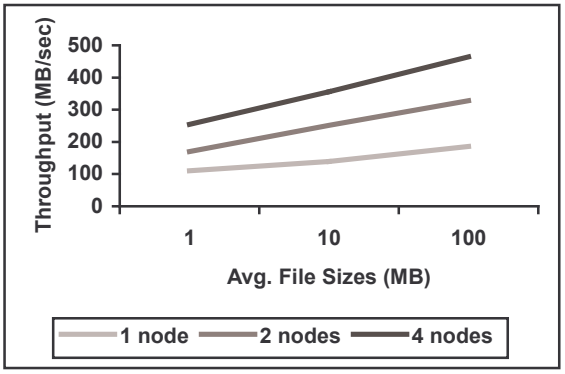
Insert-only experiments were performed for both image and video dataset. The hardware setup is similar to the previous experiments. However, for this set of experiments, the server machine acted as both the client as well as the server, thereby preventing network bandwidth bottlenecks on the system throughput. The degree of concurrency of loading the data in Oracle SecureFile objects was varied from 1 to 16. Figure 11 demonstrates that in both scenarios, SecureFiles write throughput scales with the number of clients loading the data. The SecureFiles throughput gets driven to asymptotically converge with the disk system throughput.

6.3 Scalability of SecureFiles on Real Application Clusters

The test setup and dataset configuration for this set of experiments is the same as mentioned in section 6.1.1. The dataset comprised of files of sizes ranging around 1MB, 10MB and 100MB for the scope of these experiments.

6.3.1 Hardware Setup

The hardware setup consists of four Real Application Cluster node set on four server machines, each consisting of two 3.4 GHz hyper-threaded Intel Xeon processors and 6 GB RAM. HBA is 2GBps that limits I/O bandwidth of the node to 276 MB/sec. The storage system consists of three EMC CX700 disk arrays connected through 2 switches with 12 LUNs spanning the three arrays.



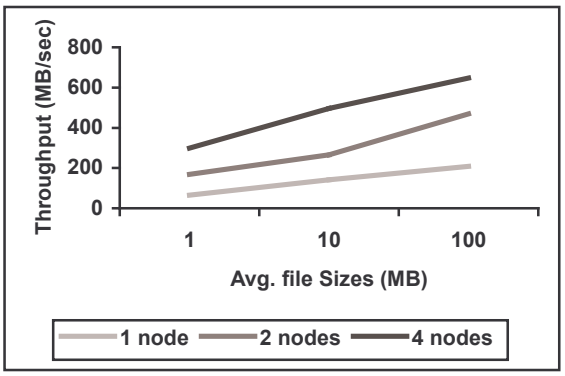


Figure 12. Throughput scalability of inserts operations of SecureFiles with number of DB instances in RAC

Figure 13. Throughput scalability of read operations of SecureFiles with number of DB instances in RAC

6.3.2 Experiments and Results

Experiments were performed on three different RAC configurations. Read and write operations were issued on a single database instance, on two instances and on all four instances. Single processes performed the operations from each of the nodes in all three configurations.

Figures 12 and 13 demonstrate throughput for all file sizes for read and writes. Both read and write throughputs of SecureFiles scale proportionally to the number of nodes across all file sizes.

7. CONCLUSIONS AND FUTURE WORK

In the past, Database management systems have been designed to provide scalable execution of storage and access of relational data. However, throughout the last decade, document management and multimedia applications have changed the dynamics of data ingestion in terms of rates, volumes and structures. This has resulted in around eighty five percent of data being non-relational that is not managed by database systems.

Current content management applications use filesystems to store unstructured data as they provide better throughput of data and access operations across all sizes and types and use database systems to manage accompanying relational metadata for indexing and querying purposes. This dichotomy in storage creates a need for compromises in one or more of high availability, scalability, performance or functionality. With Oracle 11g Database Server's SecureFiles capabilities we now have a next generation unified unstructured and relational data management platform without compromises. Our in-house performance evaluations demonstrate that Oracle SecureFiles is capable of providing optimal execution throughput and scalability for unstructured content while preserving the advantages of relational database management.

Oracle SecureFiles System is a consolidated industry-strength storage management solution for relational as well as unstructured data that does not compromise filesystem performance as well as secure database management features. New applications developed on top of Oracle SecureFiles will receive all the benefits of the framework. Future work includes efforts to provide efficient migration of existing applications based on LOBs. Efforts to support all existing filesystem interfaces are required to achieve higher adoption across filesystem based content management developers. Detailed evaluation experiments are required to be conducted on workloads comprising of hundreds of terabytes to petabytes of unstructured data under various hardware and system configurations.

8. ACKNOWLEDGMENTS

We acknowledge all members of the Oracle SecureFiles team for conducting research, design, development, implementation, functional testing and productization of Oracle SecureFiles. We also acknowledge the contributions of K. Baloglu, B. Baddepudi, J. Djegaradjane, Joy Forsythe, Vipin Gokhale, Liwen Huang, Nirman Kumar, Chao Liang, Xiaoming Liu, Karthik Rajan, Dheeraj Pandey and Niraj Srivastava.

9. REFERENCES

[1] Blumberg, R., Atre, S. *The Problem with Unstructured Data*. DM Review Magazine, Feb. 2003.

[2] Lallier, J. *Storage Management in the Year 2010*. Computer Technology Review, September 2004.

[3] Vijayan, P. *Iron File Systems.* Thesis Submitted for Doctor of Philosophy in Computer Sciences, University of Wisconsin-Madison, 2006.

[4] *Oracle Database 11g Product Family*. An Oracle White Paper, January 2008.

[5] Lahiri, T., Srihari, V., Chan, W., Macnaughton, N., Chandrasekaran, S. *Cache Fusion: Extending Shared-Disk*

Clusters with Shared Caches, Proceedings of the 27th VLDB conference, Roma, Italy, 2001.

[6] Biggar, H. *Experiencing Data De-Duplication: Improving Efficiency and Reducing Capacity Requirements*. A SearchStorage.com White Paper, Feb 2007.

[7] Shapiro, M., Miller, E. *Managing Databases with Binary Large Objects*. Proceedings of the 16th IEEE Mass Storage System Symposium, San Diego, CA, March 1999.

[8] Olson, M. A., *The Design and Implementation of Inversion Filesystem.* Proceedings of the winter 1993 USENIX Conference, Berkeley, CA, 1993.

[9] Gray, J. *Greetings! From a Filesystem User*. 4th USENIX Conference on File and Storage Technologies, San Francisco, CA, DEC 2005

[10] Sears, R., Ingen, C., Gray, J. *To BLOB or not to BLOB: Large object Storage in a database or a Filesystem?* Microsoft Research Technical Report, MSR-TR-2006-45, June 2006.

[11] Dumler, M. *Microsoft SQL Server 2005*. A Microsoft Product Guide, September 2005.

[12] Seltzer, M., Olson, M. *LIBTP: Portable, Modular Transactions for UNIX.* Proceedings of the 1992 Winter Usenix, San Francisco, CA, JAN 1992.

[13] Gehani, N., Jagadish, H. V., Roome, and W. D. OdeFS: *A Filesystem Interface to an Object-Oriented Database.* Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, Chile, 1994.

[14] Murphy, N., Tonkelowitz, M., and Vernal, M. *The Design and Implementation of the Database Filesystem*. www.eecs.harvard.edu/vernal/learn/cs261r/index.shtml, January 2002.

[15] *LUSTRE FILE SYSTEM: High Performance Storage Architecture and Scalable Cluster File System*. A Sun Microsystems White Paper, DEC 2007.

[16] Ghemawat, S., Gobioff, H., Leung, S. *The Google File System*. 19th ACM Symposium on Operating Systems Principles, NY, OCT 2003.

[17] *Architecture of ZFS for Lustre*. Sun Micorsystems, FEB 2008

[18] Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., Gruber, R. Bigtable*: A Distributed Storage System for Structured Data*.7th Usenix Symposium on Operating Systems Design and Implementation, Seattle, WA, Nov 2006.

[19] Rajamani, R. *Oracle Total recall/ Flashback Data Archive*. An Oracle White Paper, June 2007.

[20] Jensen, Christian S., Snodgrass, Richard T. *Temporal Data Management*. IEEE Transactions on Knowledge and data Engineering, Vol. 11, No. 1, January 1999.

[21] Stonebraker, M., Madden, S., Abadi, D., Harizopoulos, S., Hachem, N., Helland, P. *The End of an Architectural Era (It's Time for a Complete Rewrite). Proceedings of VLDB, Vienna, SEP 2007*

[22] Adams, P. *National Ignition facility and 11g SecureFiles.* Oracle Openworld, NOV 2007.

[23] Gray, J., Graefe, G. *The Five-Minute Rule Ten Years Later, and Other Computer Storage Rules of Thumb*. Proceedings of ACM SIGMOD, Tucson, AR, 1997.

[24] Nath, S, Kansal, M. FlashDB: *Dynamic Self-tuning Database for NAND Flash*. Proceedings of the International Conference on Information Processing in Sensor Networks, Cambridge, MA, APR 2007.

[25] *Flash Filesystems Overview*. An Intel White Paper, 2006.

[26] Geoff Lee. *Oracle Database 11g XML DB Technical Overview*. An Oracle White Paper, July 2007.