Oracle SecureFiles: Prepared for the Digital Deluge

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

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

{Niloy Mukherjee, Amit.Ganesh, Vinayagam.Djegaradjane, Sujatha.Muthulingam, Wei.Zhang, Krishna.Kunchithapadam, Scott.Lynn, Bharath.Aleti, Kam.Shergill, Shaoyu.Wang}
@Oracle.com

ABSTRACT

Digital unstructured data volumes across enterprise, Internet and multimedia applications are predicted to surpass 6.023x10^{23} (Avogadro's number) bits a year in the next fifteen years. This poses tremendous scalability challenges for data management solutions in the coming decades. Filesystems seem to be preferred by data management application designers for providing storage solutions for such unstructured data volumes.

Oracle SecureFiles is emerging as the database solution to break the performance barrier that has kept unstructured content out of database management systems and to provide advanced filesystem functionality, while letting applications fully leverage the strengths of the RDBMS from transactions to partitioning to rollforward recovery. A set of preliminary performance results was presented at the 34th International Conference on Very Large Data Bases (VLDB 2008). It was claimed that SecureFiles would scale maximally as physical storage systems scale up. We legitimize our claims on SecureFiles scalability through this paper, presenting the scalability aspects of SecureFiles through a performance evaluation of I/O bound filesystem like operations on one of the latest high performance cluster of servers and storage.

We are presenting benchmark results that we believe represent a world record database insertion rate for any published result - at over 4.4GB/s using a cluster of seven servers. For 100 byte rows, that represents an insertion rate of 45 billion records a second in relational terms. In terms of unstructured data storage, the scale represents an insertion rate of more than 3.7 million 100 MB high-resolution multimedia videos a day.

1. INTRODUCTION

The rapid growth of web services as well as digital media volumes over the past decade has been driving software applications to deal with data objects of all sizes ranging from a few kilobytes to a few gigabytes. Most of these application data objects consist of unstructured content accompanied by relational or structured content. Application designers have the choice to use either filesystems or database management systems or a combination of both to manage such data. It has been observed that filesystems have been preferred over database management systems for providing storage solutions for unstructured data, even though database systems are equipped with advanced and secure data management features such as transactional atomicity, consistency, durability, manageability, availability and queriability that are not present in most filesystems. Databases on the other hand have been the universal choice for application designers to manage relational content. The reasons for such preferences are based on certain myths – databases are optimized to process small size objects that undergo frequent updates while filesystems are optimized to provide better throughput for larger data objects that are more archival in nature. Several studies have been conducted to explain the above decision tradeoffs with more soundness [1][2][3]. One of them suggests that 256 KB serves as a breakeven point and recommends filesystems as the more preferred option for sizes greater than 256 KB [2]. Given the fact that today's applications generate larger and larger objects, such a break-even point implies that around 85% of unstructured content will be staying out of databases [4].

Oracle SecureFiles [5] was introduced in 2007 as a completely new clustered filesystem-like data storage architecture to get rid of such break-even points. It was primarily designed to provide filesystem like or better throughput across all object sizes along with advanced filesystem features such as compression, encryption and de-duplication [6], without compromising on the advanced data management features available in the Oracle RDBMS. However, it was also designed to meet one urgent objective - to be able to meet and beat the scalability challenges of data management in the years to come resulting from explosion of the digital data universe.

2. EXPLOSION OF DATA VOLUMES

Growth of data volumes has been following patterns of Moore's law, however, at a more aggressive scale. An IDC research [7] demonstrates that 281 exabytes of data was ingested in the year 2007 alone. It predicts that the same will be around 1800 exabytes by 2011. This implies that in the next 15 years, the amount of data ingested per year will surface Avogadro's number in terms of numbers of bits (602,200,000,000,000,000,000,000). The abundance of data management applications has been contributing to the growth of physical hardware and storage devices. The growth in scale of physical hardware is also making it cheaper. The above cause-effect cycle is driving applications to ingest larger and larger data objects, thereby resulting in a recursive demand for more data ingestion.

Multimedia services, Internet access in emerging countries, sensor-based applications, datacenters, and social networks are contributing to the digital deluge, driving software applications to deal with a diverse range of object sizes, from several gigabytes of DVD quality videos to a few kilobytes of email messages. The number of data objects — documents, images, videos, and emails — is growing 50% faster than the number of gigabytes [7]. The information created in 2011 will be contained in more than 20 quadrillion of such data objects, thereby posing a tremendous management challenge for data management solutions. Michael Brodie [8] presented a big picture of data volumes and ingestion rates faced by data management software applications. In 2007 alone, Internet applications such as YouTube witnessed more than a million videos ingested per day. Users of MySpace loaded more than 1 million images, 25 million mp3 format songs, and 60000 videos per day across 250 servers. Cisco Systems [9] predict that Internet traffic will quintuple between now and 2011, resulting in around 11 exabytes per month of data. Out of all growing content, 25% are original while the rest is replicated. 95% of this data is unstructured; consumer generated and growing while the rest 10-15% of the data is structured [10].

The scenario has been posing an urgent requirement for a consolidated industry-strength data management solution to be able to scale to meet and beat this upcoming data explosion in the next few years. We claimed that Oracle SecureFiles architecture, introduced in 2007, could achieve such scale based on our preliminary throughput evaluations. In our previous papers [5][11], we had mentioned that SecureFiles architecture is capable to provide maximum scalability achievable with any hardware and storage system configuration. This paper builds on [5] to legitimize our claims on Oracle SecureFiles scalability, serving as the platform to explore whether the architecture is really capable of meeting the data explosion challenges.

The remainder of the paper is organized as follows. We present a brief review of SecureFiles architecture in section 3. Section 3 also discusses some of the architecture components as well as features inherited from the Oracle database server that contribute to its scalability. The primary focus of the paper is a performance evaluation of SecureFiles scalability on I/O bound filesystem like operations. We perform an exhaustive performance evaluation of data ingestion and retrieval operations like write and read on a high-end hardware platform, a cluster of seven server machines, each equipped with 4 quad-core Intel Xeon processors and 32GB of memory, connected to a disk array of 15 drive modules, each containing 16 drives of 146GB storage capacity. The experiments reveal that SecureFiles is able to maximally scale on the above system delivering an ingestion rate of more than 4.5 GB/s. This extreme scalability translates to around 3.7 million most high-resolution YouTube videos a day just using a single seven-node hardware cluster. Details of experimental setup, performance experiments and evaluations are presented in section 4. Section 5 enumerates a few published throughput results achieved by existing database solutions and compares them with the ones generated by SecureFiles. The paper is concluded in section 6 followed by acknowledgements and references.

3. SCALABLE ARCHITECTURE

The section provides a brief overview of Oracle SecureFiles and subsequently discusses some of its components as well as inherited features from the Oracle database server that contribute to the scalability of the system.

3.1 SecureFiles: A Brief Review

Oracle SecureFiles stores data as first-class objects within the database and supports all types of content and all file sizes. The underlying database storage serves the purpose of a physical filesystem-like repository. Oracle tables, indexes, table partitions, index organized tables and other logical data structures can be enhanced to define combined filesystem and relational metadata repositories. Unstructured data resides in SecureFiles segments [5] consisting of a set of Oracle database extents where each extent is a set of logically contiguous storage blocks. Unlike traditional filesystems, SecureFiles are not assigned fixed storage space. Instead, they grow dynamically with the workload requirements. Individual blocks in SecureFiles segments are mostly meant for file like data with a few percent of them are reserved for segment metadata operations. Application data objects are stored in the segment as SecureFiles objects. A SecureFiles object is a set of chunks allocated from the segment with a few blocks reserved for filesystem inode like metadata management. SecureFiles architecture follows ‘copy-on-write’ semantics, i.e., updates and overwrites of SecureFiles objects are never in-place. Multiple SecureFiles objects can be modified under a single database transaction. An individual SecureFiles object can be modified by a single transaction at a given point in time. The theoretical limit of a SecureFiles segment size is 128 terabytes that can be scaled out to around 8 exabytes using Oracle Partitioning [5]. An Oracle database can host multiple filesystem repositories using different SecureFiles segments. The physical storage of SecureFiles segments can vary from being SCSI, SATA, FLASH or TAPE without changing the logical view of the system. The database server serves the role of a filesystem server for SecureFiles. Clients for Oracle SecureFiles provide SQL standards compliant database interfaces.

Oracle Databases can be run on multi-node, shared-cache shared-disk clusters using the Real Application Clusters feature [12]. 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. Unlike traditional disk-based data coherency mechanisms, Real Application Clusters use Cache Fusion [13]. Cache Fusion employs interconnect network across the shared disks and the database nodes to maintain coherence in database buffer caches across all database nodes. 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, thereby leveraging the scalability possible with multi-node clustered systems. Several components of SecureFiles architecture have been designed to provide maximum scalability on multi-node clusters.

Given that scalability on multi-node clusters is an important objective to be met, scalability requirements on single node systems remain equally important. Multi-core multi-processor systems have been scaling up cost-effectively over the last few years, thereby, posing requirements for SecureFiles to be able to maximally scale on such systems.

The rest of this section details various components within SecureFiles as well as features inherited from the Oracle database server that contribute to prevent scalability bottlenecks during data manipulation as well retrieval operations in the system.

3.2 Scalability: Data Manipulation Ops

Figure 1 illustrates a logical setup of Oracle SecureFiles on a multi-node cluster for data manipulation operations. Each individual node is a multi-core system with a very high degree of concurrency of server processes performing SecureFiles object data manipulations (writes / updates / deletes / overwrites). The components responsible for providing scalability during these operations are described in details subsequently.

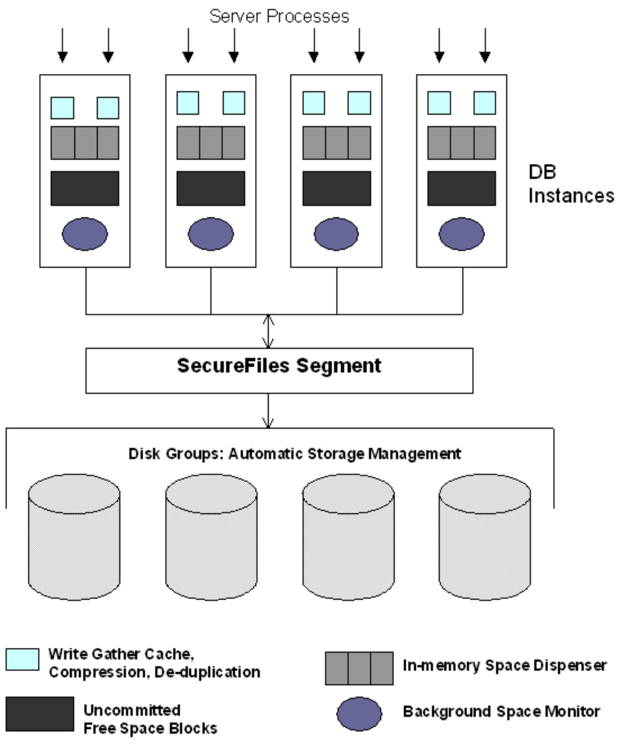

Server Processes
DB Instances
SecureFiles Segment
Disk Groups: Automatic Storage Management
Write Gather Cache, Compression, De-duplication
In-memory Space Dispenser
Uncommitted Free Space Blocks
Background Space Monitor

Figure 1. Scalability components in SecureFiles for data manipulation operations

3.2.1 Write Gather Cache

The Write Gather Cache (WGC) is a subset of the database cache private to Oracle SecureFiles object data. Individual server processes allocate buffers from process private memory pools. As a result, multiple concurrent server processes alleviate the need to contend on shared memory resources. Individual server processes 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. Buffering of in-flight data for every server process allows for fewer invocations of inode, space and I/O management layers. As a side effect, it also results in fewer but larger contiguous space allocations implying larger contiguous physical I/O. Overall, write performance throughputs scale up with the number of server processes running concurrently on single node systems as well as multi-node clusters.

3.2.2 Compression and De-duplication

Compression and de-duplication methods in SecureFiles also contribute to its overall scalability by reducing the amount of data to be materialized in physical storage. The Compression layer, when enabled, can automatically detect if SecureFiles 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 for a server process when buffered contiguous data staged in its Write Gather Cache exceeds a configured boundary threshold. Multiple contiguous compression subunits are encompassed within a larger unit whose size is determined by the Write Gather Cache flush threshold. Compression is performed piecewise in such a way that efficient random access of large files is possible. Compression not only results in maximization of storage utilization but also improves overall scalability of SecureFiles by reducing physical I/O sizes, data logging for media recovery, and encryption overheads, if encryption is enabled. The SecureFiles architecture provides varying degrees of compression that represent a tradeoff between physical I/O and CPU costs.

When de-duplication is enabled for a SecureFiles segment, checksums are generated on SecureFiles data objects staged by the segment. A secure prefix hash is generated for the first few kilobytes and a full hash is generated for the entire object. The hashes are stored in an Oracle index. During writes, as data gets staged in the Write Gather Cache for a server process, the prefix hash is generated and compared to the set of existing prefix hashes. If there is a prefix match, then the SecureFiles object associated with the original prefix hash 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. The component therefore contributes in scaling up throughput of applications that are required to store multiple instances of data objects, by preventing redundant physical I/O on the underlying storage system.

3.2.3 Free Space Management

The free space management is one of the major components responsible for scalability of SecureFiles throughput during data manipulation operations. The layer is responsible for allocating logical free space to SecureFiles objects from SecureFiles segments and de-allocating used space from SecureFiles objects back to SecureFiles segments keeping the real density and seek amortization trend in mind. The scalable space management features are listed as follows.

3.2.3.1 Variable Size Chunks

The space management layer supports allocation of variable sized chunks. With SecureFiles 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 scalable read and write access.

3.2.3.2 Background Free Space Monitor

The background free space monitor is responsible for managing space usage for SecureFiles segments. A SecureFiles segment comprises of a collection of database extents that are spread across the entire underlying storage system. As has been mentioned in section 3.1, SecureFiles segments are not pre-allocated with a fixed size, unlike typical filesystems. Individual nodes in clusters spawn respective background space monitors. The free space monitor checks space usage in the segment proactively collecting statistics from its individual node. Based on the amount of space consumed by SecureFiles objects as well as the amount of space reusable from old versions (resulting from deletes, truncates as well as 'copy-on-write' operations), the monitor decides to allocate necessary space in terms of database extents to grow the segments. Extents allocated to SecureFiles segments are first pre-split into chunks using methods that preserve the benefits of the Write Gather Cache. The metadata associated with the chunks are stored on various metadata blocks (Committed Free Space Blocks or CFS) in the segment that are handled through the database buffer cache. The CFS blocks are hashed on chunk sizes and meet space requests on a best-fit basis. The free space in SecureFiles segments mapped by these blocks is shared across all the instances in a distributed Oracle Real Application Cluster environment. The background space monitors perform their operations in a non-blocking manner. They do not proceed if they require waiting while acquiring latches, pins and other forms of locks on the space metadata blocks.

3.2.3.3 In-memory Space Dispenser

To achieve maximum scalability within a single node system under high concurrency environments, free space allocations to SecureFiles objects are managed in-memory through a shared-memory data structure known as the in-memory space dispenser. Individual database nodes create in-memory dispensers that are never shared across multiple nodes in a clustered system.

Once created, the in-memory dispenser requests the background space monitor to transfer free space entries from the underlying SecureFiles segment. This results in the distribution of cluster-wide free space in the segments to individual nodes. As a result, free space allocations requested by server processes are met by the local database instance, thereby reducing cluster wide network and disk traffic.

The design of the in-memory dispenser allows space allocation operations to scale with the degree of concurrency on a single database node. Private in-memory space dispensers in individual nodes prevent the need for server processes to communicate across nodes in a clustered system to maintain free space metadata coherence. The design therefore alleviates scalability bottlenecks of space allocation operations as the number of nodes in a cluster is scaled up.

3.2.3.4 Uncommitted Free Space Blocks

Operations such as full overwrites / rewrites, updates and deletes of SecureFiles 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. Uncommitted Free Space Blocks are affined to individual database nodes. The space monitor 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.

3.2.4 Automatic Storage Management

Automatic Storage Management (ASM) [14] is a feature in Oracle Database that assists manageability of underlying physical storage. Oracle SecureFiles extensively uses this feature to guarantee that the physical storage management is optimized to generate the maximum scalability of the storage system. ASM provides a simple storage management interface that is consistent across all server and storage platforms. Automatic Storage Management virtualizes the physical shared-disk database storage into disk groups. One or more disk groups can be assigned to SecureFiles segments. Automatic Storage Management automates the physical placement of the logical segments within those disk groups. It spreads the segment layout evenly across all available storage resources to scale performance and maximize utilization. This even distribution of segment layouts makes the manual I/O performance tuning obsolete. ASM provides three mirroring options for protection against disk failure: none, two-way, and three-way mirroring. As a vertically integrated volume manager, purpose-built for Oracle databases, ASM combined with SecureFiles provides the performance benefits of raw asynchronous I/O within database transaction boundaries. Furthermore, ASM allows change of physical storage configuration without having to take the database offline. ASM automatically rebalances SecureFiles segments across the disk group after disks have been added or dropped.

3.3 Scalability: Data Retrieval Ops

Figure 2 illustrates a logical setup of Oracle SecureFiles on a multi-node cluster for data retrieval operations. The components specifically responsible for providing scalability for read operations are described in details subsequently.

3.3.1 CR Mechanism and SecureFiles Inodes

Oracle's Consistent Read (CR) [15] is a fundamental mechanism built in Oracle database server that contributes to the maximum scalability of read operations. CR is a version-based concurrency control protocol that allows transactions to perform reads without acquiring any locks. Each transaction in Oracle is associated with a snapshot time, known as the System Change Number (SCN) [16], and the CR mechanism guarantees that any data read by a transaction is transactional consistent as of that SCN. When a transaction performs a change to a block, it stores the information required to undo that change in a rollback segment. When a transaction reads a block, the CR mechanism uses the stored undo information to create an earlier version of the block (a clone) that is consistent as of the reading transaction's SCN. Clones are created in-memory and are never written to disk. A read operation therefore never needs to wait for another transaction to commit or abort since the CR mechanism automatically reconstructs the version of the block required by the operation. This mechanism therefore allows high concurrency for read operations.

SecureFiles Inode management layer makes use of the Oracle CR mechanism to allow for maximum scalability of read operations on individual objects. The inode management layer is responsible for initiating on-disk storage and access operations on SecureFiles 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 SecureFiles 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 SecureFiles object, the metadata structure is also used to map logical offsets to physical offsets on disk. Modifications of metadata structures are transactionally managed and are recoverable after process, session and instance failures. Read operations on these metadata blocks follow the CR mechanism, thereby allowing multiple concurrent processes to retrieve them simultaneously. Each process reads a transactionally consistent copy of the inode metadata thereby returning consistent versions of data to the applications without waiting for write transactions to complete on the objects.

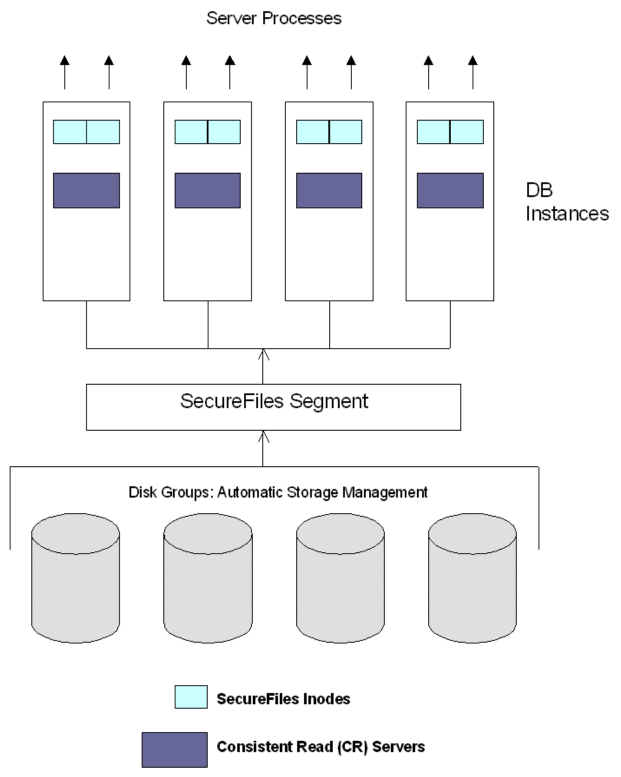

Server Processes
DB Instances
SecureFiles Segment
Disk Groups: Automatic Storage Management
SecureFiles Inodes
Consistent Read (CR) Servers

Figure 2. Scalability components in SecureFiles for data retrieval operations

SecureFiles objects maintain inodes independent of each other. This prevents single points of contention in concurrent environments during update, delete and append operations on SecureFiles 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 to store the data being written to disk. The metadata management structures can therefore scale to map terabyte-sized objects very efficiently. SecureFiles inode management layer contributes in further scale-up of read operations through an intelligent pre-fetching mechanism. The layer keeps track of access patterns and intelligently pre-fetches data before the request is actually made. Read latency is reduced by the overlap of network roundtrip with the disk I/O thereby scaling up read throughputs to greater extents.

3.3.2 CR Servers: Cache Fusion: Read-sharing

The mechanism for read sharing in Cache Fusion [13] exploits Oracle's *Consistent Read* (CR) mechanism. In RAC, when Instance A requires read access to a block that is present in the buffer cache in Instance B, it requests a copy of the block from Instance B without requiring any change of resource ownership. Instance B creates a transactional consistent CR clone of the block and ships it back to Instance A across the interconnect. Doing so has no impact on processes on Instance B since ownership of the block by Instance B is not affected. Only when the requested block is not present in any instance's cache is a disk I/O performed for the block. However, the Read-Sharing protocol guarantees that once a block is read from disk by any instance in RAC, subsequent read accesses to that block from any other instance do not require disk I/O or inter-node ownership changes.

Now that we have described details of various components and features associated with SecureFiles scalability, we demonstrate their contributions through an exhaustive set of performance evaluations in the subsequent section.

4. PERFORMANCE EVALUATION

In this section, we present a set of benchmarks and performance evaluation data to demonstrate the scalability of Oracle SecureFiles on one of the latest high-end server and storage clusters. The experiments are conducted on various workloads that cover most of the content types as well as object sizes. We demonstrate the scale up of SecureFiles on a single multi-core database node as well as scalability across the cluster.

4.1 Dataset

The dataset consists of four sets of workloads. The workloads include email documents, Microsoft Word and Portable Document Format documents, low-resolution images, high-resolution images and video files. Email documents range from 5 KB to 14 KB in size. Word and PDF documents range from 80 KB to 110 KB. The low-resolution images range from 900 KB to 1.2 MB, high-resolution bitmap images range from 8.5 MB to 11.3 MB while videos have a median size of 97 MB.

For experiments on email workloads, 1.6 million files are inserted and retrieved from a single database node. This results in approximately 17 GB of unstructured data. For experiments on Word and PDF documents, more than 1 million files are inserted from a single database node resulting in approximately 96 GB of data. 211000 low-resolution image files are used to insert around 202 GB, 22000 high-resolution images are used to insert 211 GB while 2184 video files are used to load around 208 GB of data from a single database node.

For multi-node experiments, the amount of data loaded is scaled up with the number of nodes used in the experiments. This results in a total of 730 GB in maximum across seven nodes.

4.2 Experiment Configuration

The database server is configured with a total cache size of 4 GB for a single database instances running on an individual server node. Out of the 4 GB, 2G B is assigned for shared memory data structures, and 67 MB is assigned for redo log buffers to host log records for all metadata and data modification operations in SecureFiles. 8192 bytes or 8 KB is selected as the data block size, i.e., SecureFiles chunks as well as other I/O units are multiple sets of contiguous 8 KB blocks. Each SecureFiles object is associated with a unique object number. The base table containing the metadata repository is indexed on object numbers as primary key. The index is maintained incrementally for every SecureFiles object insert.

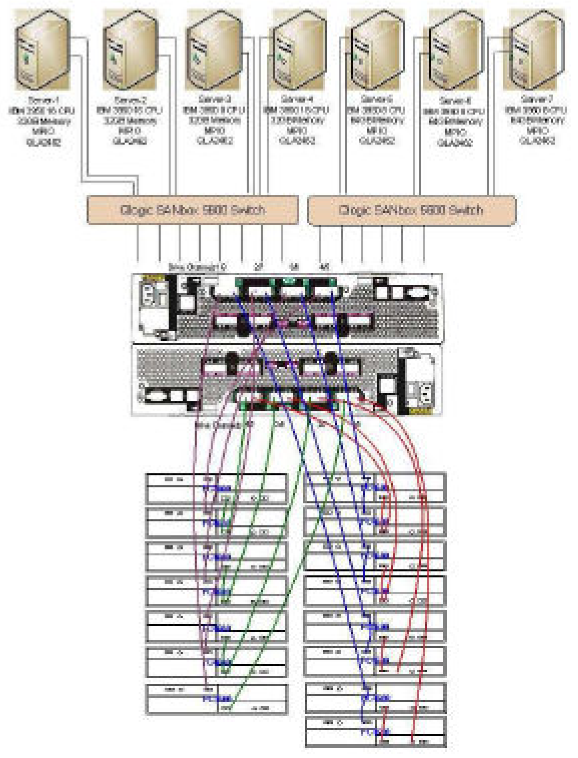


Figure 3. Hardware setup (courtesy of LSI Corporation)

The maximum number of processes as well as sessions per database node is set to 500. The flush threshold for the Write Gather Cache is set to 4 MB. SecureFiles is configured to issue direct I/Os for application data to the underlying storage bypassing the database buffer cache. However, metadata operations in SecureFiles (except the in-memory space management operations) as well as relational operations are always staged in the database buffer cache to prevent frequent accesses to underlying physical storage. The database is run in clustered mode to make use of Real Application Clusters, Cache Fusion and Read-Sharing features. The clustered database is also configured to take the advantages of Automatic Storage Management.

A single SecureFiles segment is used for the scope of the experiments. The segment is shared across all instances during multi-instance tests. The segment is configured to bypass logging of object data modifications as these operations are persisted to the underlying storage within database transaction boundaries. Even though the segment is configured to issue direct I/Os, the I/Os are configured to be asynchronous. This prevents SecureFiles operations getting blocked on I/O completions during data manipulation operations.

4.3 Hardware

Figure 3 describes the hardware setup for the experiment. It is one of the latest multi-node cluster setup by LSI Corporation using XBB-2 storage.

The setup consists of seven server nodes connected across a gigabit Interconnect. The servers are IBM 3950 machines consisting of four quad-core Intel Xeon (3.00GHz) processors, resulting in sixteen cores, 32 GB of RAM and use Red Hat Enterprise Linux 5.0. Each server is equipped with a Qlogic 4 GB dual port fiber channel Host Bus Adapter to connect to the underlying Storage Area Network. The Storage Area Network consists of 2 fiberchannel Qlogic SANbox 5600 SAN switches with a disk array. The disk array consists of a data cache of 4096 MB, 4Gb FC host interface, and 15 drive modules. Each drive module consists of 16 drives connected though fiberchannels where each drive has a capacity of 146 GB and 15000 RPMs. There are 15 Logical Unit Numbers or LUNs assigned to each server node. Each LUN is capable of driving a maximum throughput of 50 MB/s for contiguous writes of sizes 1 MB or more. Read throughputs supported are a bit higher in the range of 60 MB/s for contiguous reads of sizes 1 MB or more. Therefore, the theoretically maximal write throughput achievable by a single server node is around 750 MB/s for contiguous writes of 1MB or more. Similarly the maximal read throughput achievable by a server node is around 900 MB/s for contiguous reads of 1 MB or more.

4.4 Scalability Evaluations

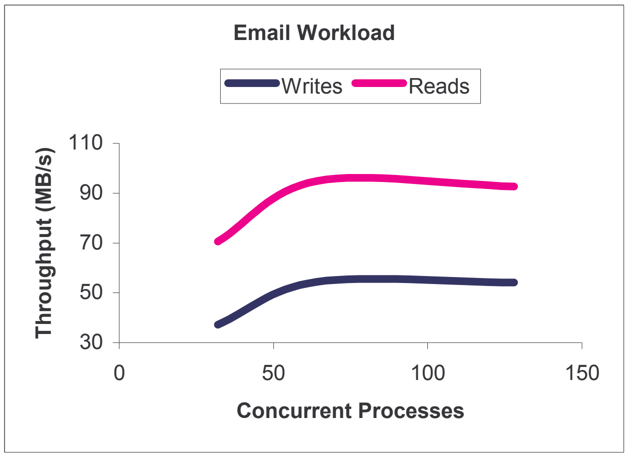
The scalability experiments are performed on the datasets described in section 4.1. For single node evaluations, clients as well as the database server are run on server 1 and the number of clients is varied till saturation of throughput is reached. The clients use OCI application program interfaces to insert and retrieve objects from the database. Multi-node experiments are conducted by varying the number of clients across several nodes in the cluster.

4.4.1 Single Node Results

This subsection reports evaluations of scalability of reads and writes operations on a single node. The operations are performed on the five distinct workloads in the dataset.

4.4.1.1 Experiments on Email Dataset

The application involves insert-only operations on the entire email dataset for archiving the email documents as SecureFiles objects. Degree of concurrency of inserts was varied from 32 to 128 streams. Average throughput of inserts was measured using the total number of objects inserted, their sizes and the time taken for the entire application to complete the loads along with index maintenance. As evident from figure 5(a), throughput of insert execution scales up with the degree of concurrency on the sixteen-core server machine. SecureFiles throughput for this specific workload is entirely CPU bound as the underlying physical I/Os comprise of maximal 2 contiguous data blocks. Following a scale up, the throughput therefore saturates after the number of processes is increased from 64. The throughput for the email archiving application is around 54.15MB/sec, thereby supporting data ingestion rate of approximately 5500 emails per second.



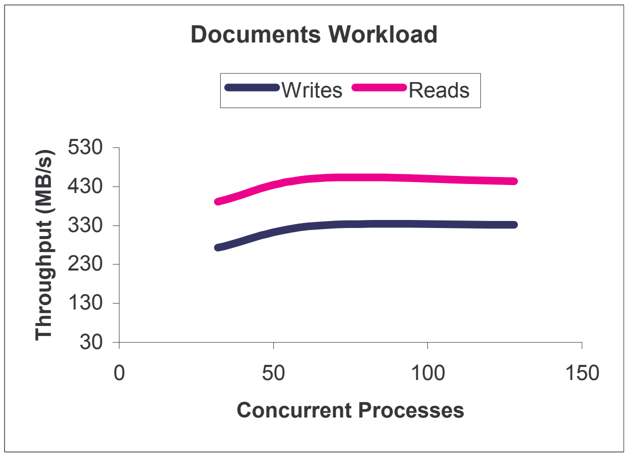
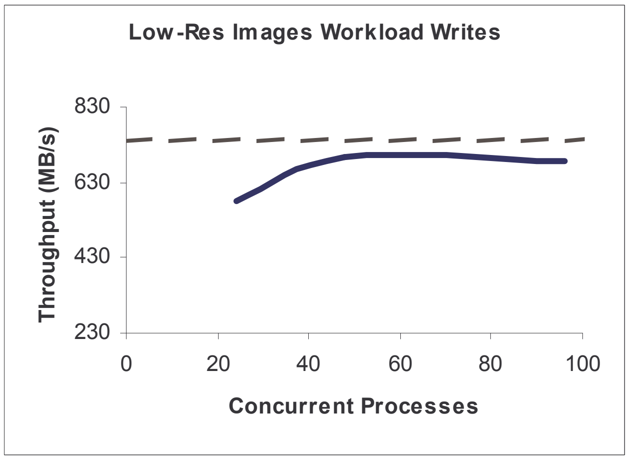


Figure 5. Throughput of SecureFiles (a) reads and writes on email workload ranging from 5KB to 14KB file sizes. (b) reads and writes on documents workload ranging from 80KB to 110KB file sizes.

Once all the objects are inserted into the database, read experiments are conducted on the objects. Degree of concurrency of inserts was varied from 32 to 128 streams. As evident from figure 5, throughput of read execution scales up with the degree of concurrency. Again, SecureFiles throughput for reads is entirely CPU bound. Following a scale up, the throughput therefore saturates after the number of processes is increased from 64. The throughput for email retrieval is around 92.7MB/sec, thereby supporting data retrieval rate of approximately 9300 emails per second.

4.4.1.2 Document Workload

The application involves insert-only operations on the entire Word and PDF documents dataset for archiving them as SecureFiles objects. Degree of concurrency of inserts was varied from 32 to 128 streams, similar to section 4.4.1.1. Multi-stream configurations were implemented to avoid conflicts on the same sets of rows across processes to prevent overwrites, thereby keeping the experiment insert-only. As evident from figure 5(b), throughput of insert execution scales up with the degree of concurrency on the server machine. SecureFiles throughput for this specific workload still remains entirely CPU bound, as the underlying physical I/Os comprise of maximal 64 contiguous data blocks. Following a scale up, the throughput therefore saturates after the number of processes is increased from 64. The throughput for the document archiving application is around 332MB/sec, thereby supporting data ingestion rate of approximately 3200 documents per second.



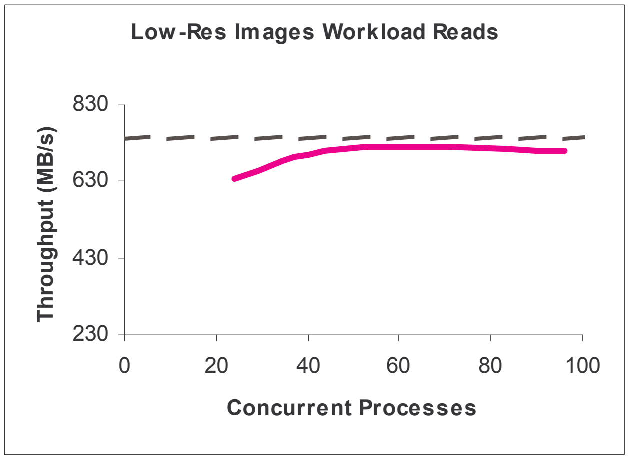
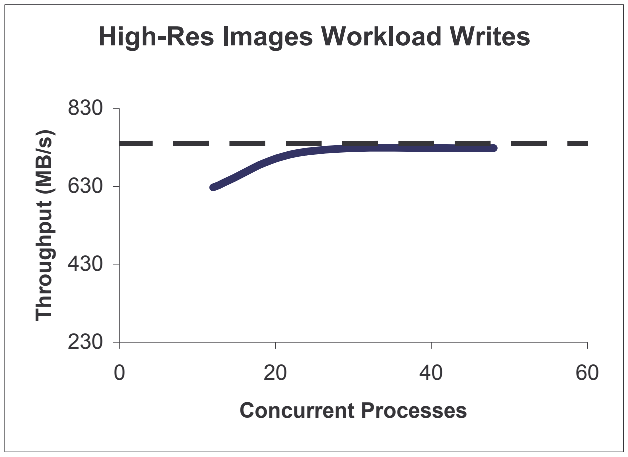


Figure 6. Throughput of SecureFiles (a) writes of low-res images ranging from 900KB to 1.2MB file sizes. (b) reads of low-res images

Once all the objects are inserted into the database, read experiments are conducted on the objects. Degree of concurrency of inserts was varied from 32 to 128 streams. As evident from figure 6, throughput of read execution scales up with the degree of concurrency. Again, SecureFiles throughput for reads is entirely CPU bound. Following a scale up, the throughput therefore saturates after the number of processes is increased from 64. The throughput for document retrieval is around 443.4MB/sec, thereby supporting data retrieval rate of approximately 4400 documents per second.



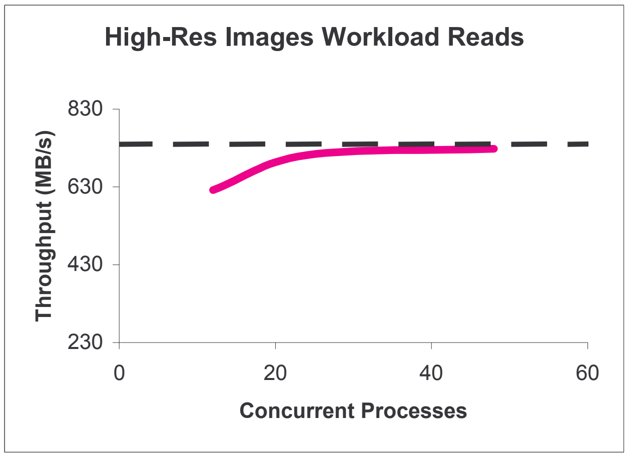
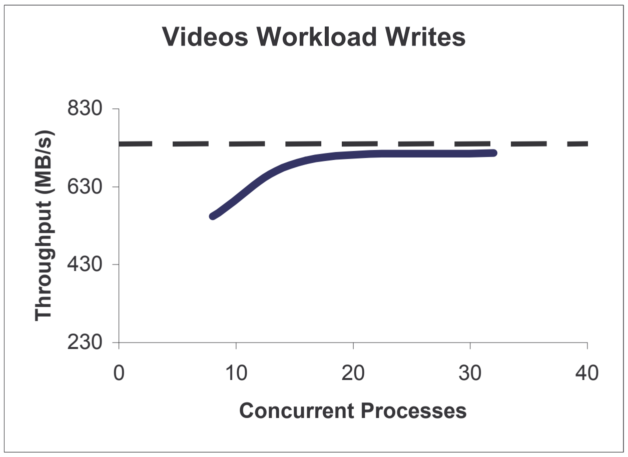


Figure 6. Throughput of SecureFiles (a) writes of high-resolution images ranging from 8.5MB to 11.3MB file sizes. (b) reads of high-resolution images.

4.4.1.3 Low-Resolution Images Workload

The application involves insert-only operations on the low-resolution images dataset comprising of images around 1MB in size for archiving them as SecureFiles objects. Degree of concurrency of inserts was varied from 24 to 96 streams in this case. As evident from figure 5(a), throughput of insert execution scales up with the degree of concurrency on the server machine. With the incoming sizes of writes ranging around 128 to 130 data blocks, optimized by free space management algorithms to be maximally contiguous, SecureFiles throughput for this specific workload starts becoming I/O bound. The throughput therefore increases irrespective of the number of cores in the server machine and saturates near the hardware limit. The throughput for this application is around 686.8MB/sec, thereby supporting data ingestion rate of approximately 700 images per second.

Once all the objects are inserted into the database, read experiments are conducted on the images. Degree of concurrency of inserts was varied from 24 to 96 streams. As evident from figure 5(b), throughput of read execution scales up with the degree of concurrency. As in the case of writes, SecureFiles throughput for reads starts becoming I/O bound. Following a scale up, the throughput therefore saturates near the hardware limit. The throughput for image retrieval is around 711MB/sec, thereby supporting data retrieval rate of approximately 730 images per second.



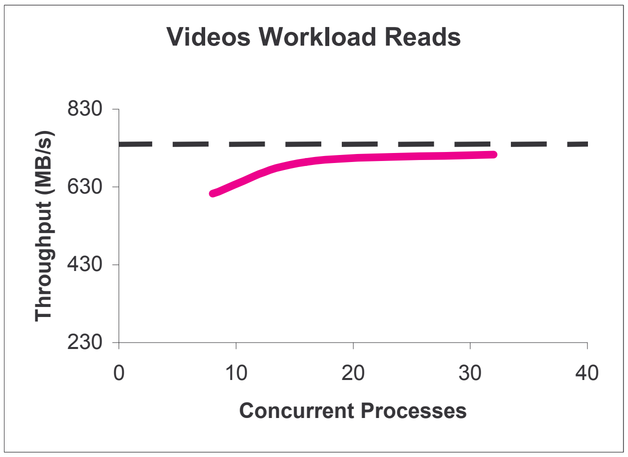
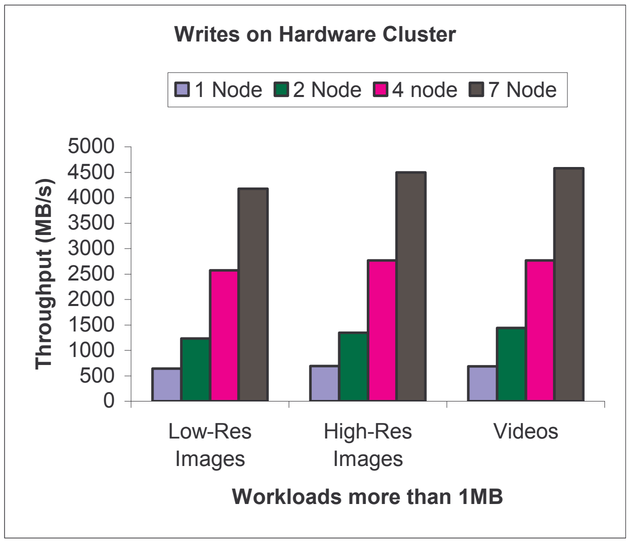


Figure 7. Throughput of SecureFiles (a) writes of high-resolution videos ranging from 90MB to 114MB file sizes. (b) reads of high-resolution videos.

4.4.1.4 High-Resolution Images Workload

The application involves insert-only operations on the high-resolution images dataset comprising of images around 10MB in size for archiving them as SecureFiles objects. Degree of concurrency of inserts was varied from 16 to 48 streams in this case. As evident from figure 6(a), throughput of insert execution scales up with the degree of concurrency on the server machine. With the incoming sizes of writes ranging around 1280 to 1300 data blocks, optimized through Write Gather Cache and free space management algorithms to be maximally contiguous, SecureFiles throughput for this specific workload starts becoming I/O bound at a lower degree of concurrency. The throughput therefore increases irrespective of the number of cores in the server machine and saturates near the hardware limit. The throughput achieved for this application is around 728MB/sec, close to the throughput limit of the hardware.

Once all the objects are inserted into the database, read experiments are conducted on the images. Degree of concurrency of inserts was varied from 16 to 48 streams. As evident from figure 6(b), throughput of read execution scales up with the degree of concurrency. As in the case of writes, SecureFiles throughput for reads becomes entirely I/O bound. Intelligent pre-fetching of object subsets enhances the scale up at a lower degree of concurrency. Following a scale up, the throughput saturates near the hardware limit. The throughput achieved is around 728MB/sec.



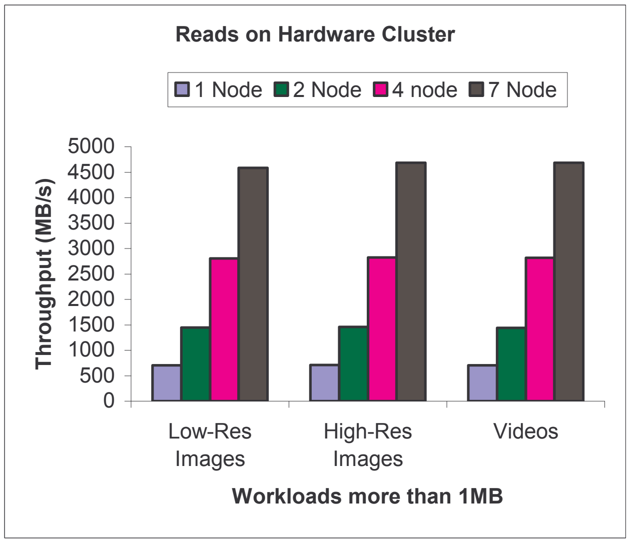


Figure 8. Write and read throughput scale-out on LSI cluster

4.4.1.5 Multimedia Videos Dataset

The application involves insert-only operations on the high-resolution images dataset comprising of images around 90-100MB in size for archiving them as SecureFiles objects. Degree of concurrency of inserts was varied from 8 to 32 streams in this case. As evident from figure 7(a), throughput of insert execution scales up with the degree of concurrency on the server machine. With the incoming sizes of writes ranging around 12800 to 13000 data blocks, optimized through Write Gather Cache and free space management algorithms to be maximally contiguous, SecureFiles throughput for this specific workload starts becoming I/O bound at even lower degrees of concurrency. The average throughput for this application is around 716MB/sec, close to the throughput limit of the hardware.

Once all the objects are inserted into the database, read experiments are conducted on the images. Degree of concurrency of inserts was varied from 8 to 32 streams. As evident from figure 7(b), throughput of read execution scales up with a smaller number of concurrent processes. As in the case of writes, SecureFiles throughput for reads is entirely I/O bound. Intelligent pre-fetching of object subsets enhances the scale up at even lower degrees of concurrency. The average throughput for image retrieval is around 714MB/sec.

4.4.2 Multi-Node Evaluations

Multi-node experiments are conducted using two, four as well as all the seven server machines. The experiments are conducted on workloads of 1MB and greater sizes as throughputs on these workloads are more storage hardware bound rather than being CPU bound. For each node involved in the experiments, the number of client per node is the one that provides the maximum throughput for the workload. Hence, the numbers of clients on each individual node in the cluster are chosen as 96 for low-resolution images, 48 for high-resolution images and 32 for videos dataset. As evident from figure 8, both read and write operations scale out with the number of instances across all workloads. Even though the underlying logical segment as well the physical storage is shared across nodes, free space management algorithms, consistent read mechanism, and cluster–wide read sharing contribute to the scale out of throughput. The write operations are driven with more than 4.4 GB/s ingestion rate while the reads generate a throughput of more than 4 GB/s across the cluster.

4.5 Evaluations Summary

The performance evaluations detailed in this section effectively demonstrate the scalability potential of SecureFiles. SecureFiles scale up with number of clients on a single database node across all workloads used, which cover object sizes from tens of kilobytes to hundreds of megabytes. For small sizes, the maximum throughput achieved by SecureFiles gets bound by CPU, while for sizes around IMB and larger, it maximizes to the limit of the hardware setup. For both data inserts as well as retrieval operations, SecureFiles scales maximally with the scale of the hardware used. SecureFiles ingests and retrieve several thousands of emails and documents per second using a single database node and is able to drive a throughput of more than 710MB/s, close to theoretical limits of the server machine. On a cluster of seven servers, SecureFiles scale out to provide more than 4.4GB/s for data insert operations and 4.4GB/s for read operations.

In real world applications, these results translate to support for extreme data ingestion rates across all object types and sizes. On a single node, data ingestion by SecureFiles is equivalent to around 480 million emails / 280 million Word documents / 63 million low-resolution images / 6.4 million high resolution images /

640000 high resolution videos a day We believe that insertion rates greater than 4.4 GB/s represent a world record in the area of databases. Such an insertion rate implies provision of more than 3.7 million highest resolution YouTube videos ingested a day just using a single seven-node hardware cluster.

5. COMPARISON WITH PUBLSIHED RESULTS

We try to compare our results with some published database benchmark results to highlight the extreme load scalability provided by SecureFiles, as has been claimed in the previous section. Unfortunately, benchmarks for unstructured data object loads using database management systems do not exist yet. Therefore, there are no published results that directly compare SecureFiles load performance with that of other database solutions. However, there exist database benchmarks that deal with ad-hoc queries and bulk loads of large volumes of purely relational data, TPC-H [17] being the most popular. Even though TPC-H benchmarks report query-per-hour performance, they provide public information about data load amounts and durations that can be used to infer load throughputs. The above scenario comes close to SecureFiles with respect to workloads used in the experiments. We therefore choose a few of the published results from these benchmarks to claim world-record SecureFiles load performance with respect to existing database solutions, with the caveat that the workloads compared are not similar.

TPC-H benchmark results report database load times that include creations of tables, loading of data from flat files and creation of indexes. Given the fact that creation of tables is a trivial operation, the reported time durations mainly comprise of data loads and index maintenance operations, somewhat similar to the experiment configuration described in section 4.

TPC-H council reports results for database sizes ranging from 100GB to 30TB database sizes. The maximum throughput among all the topmost database solutions for each category is inferred to be around 1.74GB/s, achieved on a system comprising of a cluster of 32 dual–core dual Power6 processors servers, each quipped with 32GB RAM [19]. SecureFiles, on the other hand, achieve more than 4.4GB/s on a cluster of seven sixteen-core servers.

6. CONCLUSION AND FUTURE WORK

The data management scenario has been evolving over the years with the growth of web services as well as multimedia applications. Software applications in enterprises, science and research, entertainment and other industries are dealing with data objects of all sizes ranging from a few kilobytes to a few gigabytes. It is predicted that data volumes will reach more than 6.023×10^{23} bits a year within the next fifteen years.

Filesystems have somehow been preferred over database management systems for providing storage solutions for unstructured data on grounds of performance. Oracle SecureFiles is emerging as the database solution to provide filesystem like or better throughput for storage of unstructured data across all data object sizes and types, breaking the performance barriers without compromising the strengths of the relational data management features. The paper demonstrates that the design is capable to meet the scalability challenges posed by the explosion of data volumes and ingestion rates in the coming years. The performance evaluation successfully proves the potential of Oracle SecureFiles technology to provide extreme scalability in massive data management environments for I/O bound operations, claiming a world record database insertion rate for any published result - at over 4.4GB/s.

Future work includes efforts to evaluate SecureFiles on HP Oracle Exadata Storage servers and database Machine [20], the latest high-performance hardware setup offered by Hewlett Packard Company and Oracle Corporation. Apart from testing the infrastructure using database interfaces, exhaustive evaluation is required to compare SecureFiles against traditional filesystems using standard filesystem interfaces and benchmarks.

7. ACKNOWLEDGEMENTS

We are thankful to LSI Corporation for providing us with the entire hardware setup. We thank Niraj Srivastava for assisting us with the hardware setup as well as guidelines for performance optimizations. We acknowledge all members of the Oracle SecureFiles team for conducting performance testing of Oracle SecureFiles. We also acknowledge the contributions of B. Baddepudi, Manosiz Bhattacharyya, Vipin Gokhale, Liwen Huang, Chao Liang, Xiaoming Liu, and Neil Macnaughton.

8. REFERENCES

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

[2] 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, 2006.

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

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

[5] Mukherjee, N., Aleti, B., Ganesh, A. et. al. *Oracle SecureFiles System.* Proceedings of the 34th Very Large Data Bases Endowment, 1(2), 1301-1312, 2008.

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

[7] *An Updated Forecast of Worldwide Information Growth Through 2011*. An IDC White Paper – Sponsored by EMC, 2008.

[8] Brodie, M. *Computer Science 2.0: A New World of Data Management.* Invited Industrial Talk, VLDB, 2007.

[9] *Cisco Visual Networking Index – Forecast and Methodology, 2007–2012.* A Cisco White Paper, 2008.

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

[11] Mukherjee, N., Ganesh, A., Kunchithapadam, K., Muthulingam, S. *Oracle SecureFiles - A Filesystem Architecture in Oracle Database Server.* ICSOFT (SE/MUSE/GSDCA), 60-63, 2008.

[12] Oracle Corporation. *Oracle9i Real Application Clusters Concepts Release 1 (9.0.1)*, Part Number A89867-01.

[13] 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, 2001.

[14] Manning, P. *Automatic Storage Management technical Overview.* An Oracle Technical White Paper, 2003.

[15] Bridge, W., Joshi, A., Keihl, M., Lahiri, T., Loaiza, J., and Macnaughton, N. *The Oracle Universal Server Buffer Manager*. Proceedings of the 23rd Very Large Data Bases Endowment, 1997.

[16] Joshi, A., Loaiza, J., and Lahiri, T. *Checkpointing in Oracle*. Proceedings of the 24th Very Large Data Bases Endowment, 1998.

[17] Transaction Processing Performance Council. http://www.tpc.org.

[18] Traeger, A., Zadok, E., Joukov, N., and Wright, C. *A Nine-Year Study of File System and Storage Benchmarking*. ACM Transactions on Storage, 4(2), 2008.

[19] *TPC Benchmark™ H Full Disclosure Report-IBM System p5 575 Using IBM DB2 Universal Database 8.2.* http://www.tpc.org/results/FDR/tpch/IBM_570_10000GB_20071015_FDR.pdf

[20] Weiss, R. A Technical Overview of the Oracle Exadata Storage Server. An Oracle Technical White Paper, 2008.