Archive storage system design for long-term storage of massive amounts of data

P. L. Bradshaw
K. W. Brannon
T. Clark
K. Dahman
S. Doraiswamy
L. Duyanovich
B. L. Hillsberg
W. Hineman
M. Kaczmarski
B. J. Klingenberg
X. Ma
R. Rees

A dramatic shift is underway in how organizations use computer storage. This shift will have a profound impact on storage system design. The requirement for storage of traditional transactional data is being supplemented by the necessity to store information for long periods. In 2005, a total of 2,700 petabytes of storage was allocated worldwide for information that required long-term retention, and this amount is expected to grow to an estimated 27,200 petabytes by 2010. In this paper, we review the requirements for long-term storage of data and describe an innovative approach for developing a highly scalable and flexible archive storage system using commercial off-the-shelf (COTS) components. Such a system is expected to be capable of preserving data for decades, providing efficient policy-based management of the data, and allowing efficient search and access to data regardless of data content or location.

Introduction

In order to help meet current and future long-term storage requirements [1], an archive storage system should be capable of storing, managing, and locating and retrieving multiple billions of documents over the deployment lifetime of the system. (Here, we use the term *document* to refer to both the file being archived and its associated metadata.) The system should support storing data from a wide variety of content management systems and embrace the following design goals. The system should 1) support standard interfaces for ingestion of documents from traditional content management systems as well as customer-developed applications; 2) allow access to previously stored documents in a flexible and secure manner; 3) provide efficient search on file content and metadata regardless of the interface used to ingest the documents; 4) support policy-driven storage management using multiple tiers of storage to meet service-level agreements and utilize the most cost-effective storage; 5) provide scalability for document ingestion and search to meet strict governmental and business requirements for retrieval of documents; 6) enable metadata specification, generation, and discovery, and be able to archive both the original files and the associated metadata; 7) dynamically increase or decrease the scale of the system to support changing business needs; 8) address compliance and data governance requirements, and provide appropriate security controls, nonrepudiation, and auditing; 9) provide business continuity and protection of archived documents; 10) provide storage media migration management to ensure documents stored in the archive system can be retrieved despite hardware and media aging and media obsolescence.

Figure 1 illustrates the structure of a storage archive system that meets these requirements. A variety of content management systems and customer-developed applications are able to send documents to the archive storage using standard interfaces. The archive storage system indexes and stores the documents according to policies in a cost-effective, secure, and reliable manner. Documents can be retrieved using standard interfaces when needed.

In creating a system designed to meet these objectives, two design approaches can be taken. A system can be designed and implemented by creating custom hardware and unique software components, or the system can be designed and assembled by integrating ready-made and generally available commercial off-the-shelf (COTS) [2] components with additional innovative technology. The use of COTS components enables the rapid

0018-8646/08/$5.00 © 2008 IBM

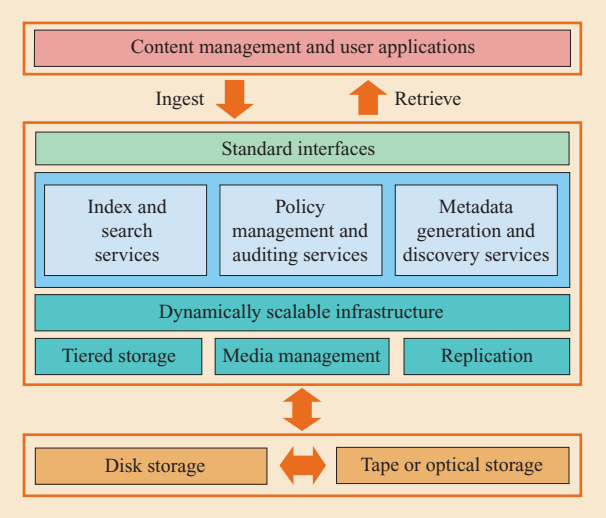

Figure 1

Archive storage system structure. Applications store and retrieve archived data through industry-standard interfaces. The archive storage system indexes and processes the data according to policies set by the storage administrator. Data is stored on disk and tape for later retrieval and is automatically migrated to avoid media obsolescence issues.

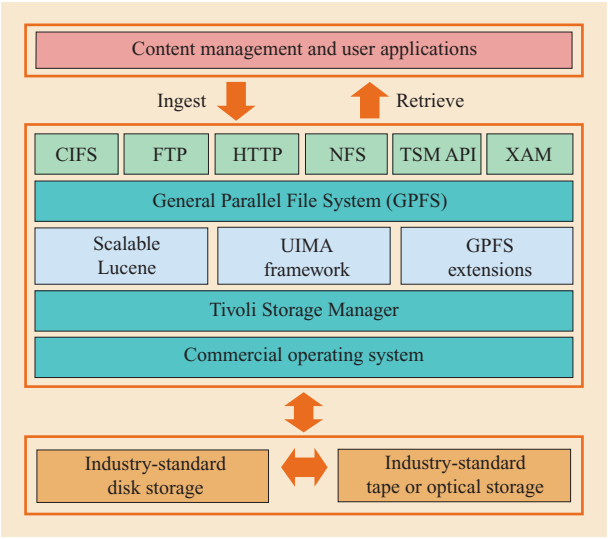

Figure 2

Archive storage system design using commercial off-the-shelf components. This system leverages open-source software and commercially developed components to create a high-quality, highly scalable, and flexible archive storage system. (CIFS: Common Internet File System; FTP: File Transfer Protocol; HTTP: Hypertext Transfer Protocol; NFS: Network File System; TSM API: Tivoli Storage Manager application programming interface; XAM: Extensible Access Method; UIMA: Unstructured Information Management Architecture.)

development of high-quality, flexible archive storage systems that can satisfy the long-term storage requirements. The hybrid approach of using COTS components and innovative technology can bring new capabilities to market much faster than the approach of creating new custom hardware and unique software.

The system described in this paper utilizes proven, long-standing, and mature components from the IBM product portfolio and the open-source community. The IBM components include the IBM General Parallel File System* (GPFS*) [3] and the IBM Tivoli* Storage Manager (TSM) [4]. GPFS was designed as a scalable file system for a high-performance computing environment and provides global access to data from multiple processing nodes. GPFS can support billions of files stored using multiple petabytes of storage, which makes it an ideal component for a highly scalable and high-performance archive system. TSM has been the open-systems storage hierarchy manager of IBM for more than 15 years and is utilized today by organizations around the world as a backup and recovery system. TSM provides the capability to manage data in a tiered storage hierarchy consisting of disk, tape, and optical storage.

The system also utilizes a new component based on the Apache Software Foundation Lucene** open-source indexing project [5]. A commercial operating system is used as the base runtime environment, and the Linus Torvalds Linux** operating system will be the initial commercial operating system used. Additional open-source components include a version of the Samba [6] network file system that has been enhanced to help improve scalability and performance.

These components are integrated with new technology to provide functions required to complete the archive system as shown in **Figure 2**. This COTS-based approach significantly benefits from the technology development investment that has been made in the component products and allows us to build low-cost, highly scalable, and flexible storage systems that can preserve data for years or decades, provide efficient policy-based management of the data, and provide efficient access to data regardless of data content. The remainder of this paper elaborates on the key requirements for a large-scale archive system and how COTS products can be used as building blocks.

Provide standard interfaces for ingesting and accessing archived documents

Managing an archive storage system through multiple generations of hardware and software technologies is

made possible by providing virtual ingestion and access of archived documents through standard interfaces (*ingestion* refers to the process of transmitting documents to the archive system, and *access* refers to the process of retrieving documents stored in the archive). An example of this kind of abstraction is when a user browses a Web site and need not be aware of the operating environment, operating system, processor model, or type of storage associated with the Web site. These components are all abstracted to the user through a set of standard interfaces. Similarly, an archive storage system should provide this abstraction through a variety of interfaces to accommodate a broad range of applications. The set of interfaces provided should include standards-based interfaces such as the Network File System (NFS) [7] and the Common Internet File System (CIFS) but can also include unique interfaces tailored for specific environments. Additionally, the system should provide the flexibility to allow more interfaces to be added easily, without affecting current users of the system.

Our system supports a number of industry-standard interfaces (**Figure 3**). Content management systems and user applications are able to use a variety of interfaces to store and retrieve archived documents. Documents can be stored using one interface and retrieved using another. Many of the interfaces are file based, which provides a one-to-one mapping with files in the file system of the archive. Applications that use or produce files—such as office productivity tools, Web servers, or multimedia applications—can easily use an archive system that provides file access over network-attached storage protocols such as NFS or CIFS [8]. Other protocols such as File Transfer Protocol (FTP) [9] and Hypertext Transfer Protocol (HTTP) [10] can also be easily deployed in this environment. Interfaces such as the TSM application programming interface [11] can also be provided, by mapping the underlying protocol streams to files in the file system. Additional interfaces may be used in certain environments, for example, interfaces that write data directly into the file system of the archive. By using a cluster file system such as GPFS, an instance of such a file may be run on an application server. In this way, the application can write directly into the archive, typically providing superior performance compared to the use of network file protocols.

The ability to ingest documents into the archive storage system using many diverse interfaces including file system interfaces is a key attribute because existing applications can be easily adapted to use the archive without complex programming changes. A file system interface approach exploits the inherent flexibility provided by the file system and utilizes the well-understood programming model for file system interaction. The biggest challenge in using a file system involves the handling of

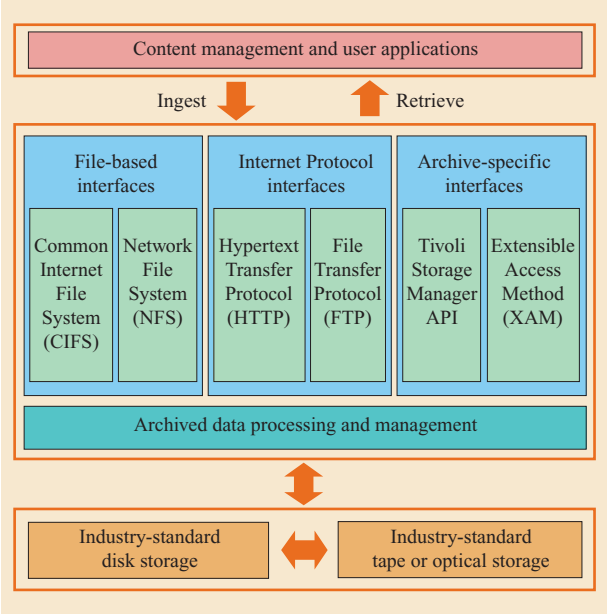

Figure 3

Archive storage system architecture using industry-standard interfaces and application programming interfaces (APIs) for storing and retrieving archived data. Applications can use well-understood interfaces to store and retrieve data. Data can be stored using one interface and retrieved by using another. Processing of the data is performed in a consistent manner regardless of the interface used to ingest the data.

new interfaces as they emerge. Some emerging interfaces, such as the Extensible Access Method (XAM) [12], may allow multiple complex data types within a single named entity or have semantics that may not easily map to file system data structures and semantics. Determining how to store data from this class of interfaces in the archive file system, as well as whether and how to allow access to this data from other interfaces, will be addressed on a case-by-case basis. For example, one may ask whether an NFS application should view a single item or multiple items when accessing a complex object stored by XAM. This challenge becomes part of the information preservation [13, 14] challenge once that long-term data storage has been successfully addressed.

Enable efficient search on data content and metadata regardless of interface

Locating documents stored in a large archive is difficult without basic search capabilities. All serious archive systems should provide this capability. With any inquiry to locate documents, most users will want to access the archived information using the Internet search paradigm. For example, users have become accustomed to locating the information needed through successive refinement of

Internet searches. By using metadata in addition to content, it is possible to locate documents more efficiently than is possible with content-based keyword search alone. For example, metadata makes it easier to find e-mails sent by a particular individual compared with searches of all e-mails where that individual happens to be copied or named in the body of the document.

However, unlike searching the Internet, searching an archive requires that access be controlled on the basis of subject matter and user role [15, 16] while meeting strict privacy requirements [17]. A user should not be able to view information to which they do not have access, even if that view simply verifies the existence of certain entities in the repository. Auditors, however, may need to search the entire repository. Additional information in the index of the archive or the post-processing of the search results can be used to control and filter access for information returned. This information should be maintained and updated over time as users' rights change or information is reclassified.

In addition, the archive index often represents the only means of access for locating salient documents. If the index becomes corrupt, discoveries might return partial information, and the integrity of the system would be compromised. Therefore, it is important that the integrity of the index be protected in a secure manner in addition to allowing for filtering of the content. In our system, we protect the archive index using the same mechanisms used for protecting ingested files and metadata.

The archive system is expected to also use the archive index and search mechanism to provide two key features in operating an archive: 1) policy binding and data management and 2) metadata annotation and search.

Support policy-driven tiered storage management

In order to store very large numbers of documents, the archive system should use tiered storage to help optimize the total cost of the system. Managing billions of archived documents in a tiered storage hierarchy requires that the system scales in terms of the selection and processing of documents to be moved between storage tiers.

In our system, policy statements are used to specify how archived documents are to be managed. The association of certain policies with specific documents is known as *policy binding*. Policies include declarations such as retention time or event, retrieval time objectives, data recovery point, and recovery time objectives. To help improve the effectiveness of applying policies to vast numbers of documents, the system combines search semantics with the policy specifications. For example, a search predicate can be used to identify the information to which specific policy declarations will be applied. The policy language allows the specification of search criteria as an input for identifying the documents to which the policy applies. This also implies that documents that enter the archive after the policy has been specified will be associated with the policy if their content or metadata matches the search criteria. The challenge with this behavior is that the administrator may need to audit and refine the policy search criteria over time to help ensure that it still has the intended effect. At the time that the policy was specified, the administrator may have been satisfied that the search identified the correct data. Documents added later may be inadvertently associated with the policy or excluded because additional qualifiers needed in the search criteria were overlooked. For example, policy that is bound to all documents that contain the term "confidential" would also inadvertently match documents that contain the phrase "not confidential." For example, the adaptive presentation approach taken in the IBM Classification Module for OmniFind* Discovery Edition [18] tries to present discovery search results in a way that prompts the user for additional refinement. Techniques such as this should be considered to assist the user with policy binding.

The use of search predicates to associate management policy with documents is a new approach for storage management. Most policy systems, such as the IBM Data Facility Storage Management Subsystem (DFSMS*) [19, 20] and TSM, use external attributes such as creation time, dataset size, owner, and last access date to identify which policies are to be applied. It is assumed that the policy administrator understands the business use of the information and can thus map these external attributes as needed to the appropriate policy. Because an increasing number of requirements are based on data content and context, modern archiving systems should use indexing and annotation techniques for policy binding. In addition, it is desirable to enable the ability for business experts to specify policy in terms that make sense for the business, instead of relying on technical experts to translate business policy to the intricacies of dataset naming and attribute information. Thus, data indexing and metadata are important not only in establishing a mechanism for locating information later, but also for exposing the appropriate content and context for application of business policy.

As a final example, consider the scenario in which a purchasing agent (business expert) specifies that information about a particular supplier involving transactions between certain dates in the ordering application needs to be treated a certain way. Some portions of the policy-specific action relate to content information, while others relate to the metadata or annotation information that was collected about the data that was archived.

Business requirements change over time, and this implies that policy binding may also change over time. Policy rebinding is a process that includes reevaluation of policy assertions to potentially reassign management attributes to archived information. In some cases, indices may need to be regenerated. The presence of certain content might have differing implications and thus need to be annotated in a different way. While it is desirable to avoid reannotation and regenerating the index because of the time and resource involved, the ability to do so must be considered in providing the archiving solution. Other reasons for index regeneration may include disaster recovery or recovery from a partial storage failure.

Provide scalability for ingest and search

While the total amount of storage managed is an important scalability metric for any archive system, the number of documents archived represents the primary challenge for archive system scalability. The need to retain large numbers of documents requires that the archive system be capable of storing, managing, and locating and retrieving billions of documents over its deployment lifetime. Furthermore, given the explosion of unstructured and semi-structured document formats of varying types and sizes, the archive system should store and manage both small and large document sizes in an efficient and cost-effective manner. (In this context, the term *small* refers to files of less than 1 KB, and *large* refers to files more than 1 GB.) As a result, we have the following two goals for scalability. First, the integrated archive platform should be able to ingest documents from multiple sources (e.g., applications) over multiple protocols (e.g., NFS, CIFS, FTP, HTTP, and XAM) in a timely fashion. As there are natural limits to the per-node ingestion rate of the various protocols, the archive system must include support for parallel ingestion across a set of cluster nodes.

For the second goal, the design of the system should take into account the management of very large indices in order to efficiently locate and retrieve documents by content or by metadata. The indexing operations themselves require that type-dependent, CPU-intensive parsing, tokenization, and analysis operations be performed on the ingested content. In our system, these indexing operations are performed in an asynchronous fashion after ingestion in order to optimize ingestion performance. As a result, a time delay may exist before the derived content is searchable from the index. It is important that the archive implementation be able to provide parallel indexing operations to minimize this latency. Similarly, index searches must be performed in parallel over multiple index instances to increase the speed of retrieval.

Our system uses GPFS as the base mechanism for providing highly parallel access to documents and document metadata. GPFS is the file system used in many of the largest supercomputer installations in the world and is a proven and robust component for archive processing.

Enable metadata specification, generation, discovery, and archiving of data and metadata

Organizations that must retain documents often resort to saving all documents rather than attempting to identify the subset that actually needs to be retained. While this approach is simple, it is costly in terms of the storage required. Management of the data can be time consuming, and it can be labor intensive to locate and produce documents.

Similar issues are encountered when documents are archived because of the long-term business value. In the petroleum industry, seismic data used for exploration costs tens of millions of dollars to produce and thus is kept for decades. Improvements in processing power have enabled new kinds of analysis for historic data. The ability to perform new analyses requires that data archived decades ago must be efficiently located and retrieved.

Locating documents in an efficient manner can require search capabilities beyond typical name- or content-based search techniques. The ability to specify, generate, and associate metadata with archived files is required. As discussed earlier, the need to manage archive data cost effectively requires the ability to have policy-driven tiered storage management [21]. Improving the effectiveness of these policies also requires search-like capabilities, and to make that possible, metadata must be stored with the files being archived.

The concept of metadata generation for archives is already in use in some industries. In health-care systems, standard formats such as DICOM** (Digital Imaging and Communications in Medicine) [22] and HL7 (Health Level 7) [23] include metadata such as patient name and type of image within the file, allowing archive systems to extract metadata directly from the files as they are ingested. Tools can also be used to scan and analyze data and produce metadata on the basis of file content. This concept can be extended to provide an enhanced archive of data for other industries and venues. When indexing is integrated in the archiving system, flexibility is needed in the specification of metadata. For well-defined file types or formats, the built-in indexing system can extract the salient metadata as described. For proprietary data formats, the metadata should be provided to the archiving system by the application submitting the file for archive. Content management systems that store files in

an archive system play a critical role in metadata specification. Content management systems provide the ability to classify data and generate metadata in a broader context and pass that metadata to the archive system, where it can be associated with the file and used for document location and tiered storage management.

It is important that the method used by the archive system for metadata generation employ open interfaces that can be used by subject-matter experts to provide annotations that are meaningful in their domains. Metadata generation with our archiving system is accomplished through use of the Unstructured Information Management Architecture (UIMA) framework [24]. UIMA was developed by IBM and now is available as an Apache open-source software development kit that can be downloaded [25]. UIMA provides an open platform that supports building and deploying applications that manage unstructured information and provides a runtime environment in which developers can plug in and run their UIMA component implementations, along with other independently developed components. In our system, UIMA plug-in annotators are used to extract semantic knowledge from documents as they are being ingested, and the system uses this information to drive retention and tiered storage policies. The architecture supports deployment of new and more sophisticated annotators. By leveraging UIMA, our system has the flexibility to leverage IBM-developed and non-IBM-developed annotators, thereby supporting metadata generation for a wide range of data formats [26, 27].

Regardless of how the metadata is generated, it must be associated with the ingested file and stored for later use in search requests and for tiered storage management. In our system, the metadata can be passed to the archive system in a variety of ways, depending on the protocol being used. In the case of CIFS and NFS version 4, the metadata can be sent as extended attributes. With NFS version 3, HTTP, and FTP protocols, which do not support passing extended metadata, metadata can be passed to the archive by submitting a companion file containing the metadata. When received by the archive system, the metadata is processed and becomes available for document search and tiered storage management.

Store data on equipment that can transition documents to new processing and storage systems

As discussed, digital archive storage systems often must keep data for years or decades. A thorough and automated approach is necessary in order to maintain data for long periods across new generations of server and storage technologies. Media age and use should be tracked, and the system should be able to exploit new devices and automatically transition documents to these new devices and interfaces without requiring an en masse document transfer to a new system. To accomplish this processing, the system should support 1) the transition to new processing components as they become available, 2) transition to new storage devices and technologies as they become available, and 3) new interfaces for document ingest while allowing previously stored documents to be accessed via these new interfaces.

In order to be able to transition to new processing components, the system should be designed to support a heterogeneous cluster environment. A heterogeneous cluster environment allows the system to continue to run on existing processors while new processors of potentially different architectures and operating system types are added to the cluster. Over time, old processors can be decommissioned and the associated workloads moved to the new processors. This allows the system to dynamically transition the computing components of the system over the lifetime of the archive. IBM, as well as other companies, has delivered applications on heterogeneous systems for many years, and we have leveraged our experience at IBM in the design of our system. Products such as TSM and GPFS all run across many system architectures, support dynamic system transformation to new platforms, and are used as components of our system.

The archive system must have the capability to attach and exploit new storage devices as they become available. Whether it is a new interface to a hard disk drive, a new tape cartridge format, or a radical new storage technology, any long-term archive system must have techniques for handling this on-going evolution. To help meet this requirement, our system employs techniques and capabilities that include the ability to dynamically add new storage devices to the system while the system is operational. Depending on the workload, new data can be allocated to the new device, or existing data can be rebalanced over the new device while normal archive ingest processing and access operations are performed. In a similar fashion, devices no longer in use can be marked as read-only while the existing data on the device is written to other devices in the system. Once all data from a device has been transferred to other devices, the device can be removed from the system.

Another key capability includes the ability to manage offline storage media in an efficient manner. Our system has the ability to move documents from one media type to a new media type of similar or dissimilar format or technology. Our system performs this action in the background and manages the action in an automatic or manual fashion.

Many of these capabilities are possible by using virtualization techniques. Access to the data in the archive system is through a number of interfaces or APIs (application programming interfaces) and not by direct access to the storage device where the document is stored. This level of indirection is key to providing the capabilities described above.

Address compliance requirements and provide security

The businesses and government entities of today are facing a higher degree of regulation and accountability than ever, and failure to comply could result in significant penalties. Among other things, when developing strategies for archiving data, customers may have to consider SEC (Securities and Exchange Commission) rule 17a-4 [28], the Sarbanes–Oxley Act [29], and the Health Insurance Portability and Accountability Act [30]. To aid in addressing compliance requirements, an archive system should prevent unauthorized access, modification, or deletion of documents [31, 32].

In our system, protection against unauthorized operations is provided by implementing retention policies, file and metadata immutability (e.g., protection against change), and auditing. To accomplish this, GPFS is enhanced with these new capabilities. To minimize the change to GPFS and provide flexibility, these enhancements have been designed so that a large part of the retention-management logic can be implemented outside of the file system, with GPFS enforcing only the fundamental immutability properties (e.g., the expiration timestamp of a file can never be moved backward). The design also provides flexible setting of retention attributes so that a user or application can set base retention attributes through standard file system interfaces or set sophisticated retention attributes, such as event-based retention and deletion holds, using either a namespace-based mechanism or a policy-based approach. (The term *deletion hold* refers to the prevention of the deletion of data at the end of its assigned retention period.) The system supports the concept of service classes for which retention attributes and storage management parameters can be specified for groups of documents. Whenever a document is protected by multiple overlapping retention settings, the system follows the most stringent one for each retention attribute. Additionally, our system ensures that the retention settings for a document are honored even if the document is migrated to another write protection-enabled storage subsystem, such as the IBM Write-Once Read-Many (WORM) tape technology [33]. Protection against unauthorized access is provided by exploiting the encryption capabilities of the underlying storage devices.

Another key requirement is support for tamperproof audit-log documenting, which indicates how and when documents are accessed during their life cycle [34]. Besides logging document events, the audit-log component is also responsible for logging system-level events (e.g., node or component failures), configuration events (e.g., policy updates), and query requests submitted through the search interface. Since logging requirements vary, the system design is flexible and allows users to specify which events to record. A minimum set of events is always logged; for example, the logging of changes to audit-log configurations should be mandatory so that attempts to disable auditing for certain events will be detected and reported. To protect the integrity of the audit log, the log files are stored on WORM storage technologies. The retention of a log file can be set independently from the related documents or can be set to "follow" the corresponding documents; for example, a log entry cannot be removed until the corresponding document expires. Note that this requires log entries stored within the same log file have similar expiration dates if immutability is enforced at file granularity.

Since the audit-log component has the ability to capture all updating events, the framework can be augmented to provide change logs that contain information about recent updates such as data object insertion and removal. Such change logs can be sent to the policy engine (e.g., for retention policy evaluation to determine the appropriate retention attributes of newly ingested objects) or to the indexer as part of the extract, transform, and load (ETL) process. This helps reduce the need for expensive scanning and crawling of the entire file system and can significantly shorten the delay between the ingestion time of a file and when the file is indexed by the indexer. The change-log mechanism can also be leveraged to help provide an efficient remote replication mechanism. In our system, we have extended and leveraged open-source Linux operating system components to provide audit-log capabilities.

Provide business continuity and protection of archived data

Business continuity and availability for the archive system is critical to help ensure that any failures in the archive system do not result in document loss. Document availability is achieved by leveraging the functionality in the COTS products. One type of failure is the failure of a node in the archive system. GPFS clustering provides recovery from a node failure by automatically reforming the cluster with the remaining nodes. Workload management for the archive system is integrated with GPFS clustering and recovery. A second type of failure is loss of a storage volume. GPFS provides block-based replication in the form of failure groups to recover from

this type of failure in the disk-resident data. Portions of the file system data may have been migrated to TSM for hierarchical storage management, leaving stub files in GPFS that refer to the data in TSM. TSM is configured to create multiple copies of migrated data to help ensure its availability.

Preventing loss of an entire archive system from a catastrophe requires disaster-recovery capabilities. Our system provides disaster-recovery capabilities by using file-based replication to duplicate documents at remote sites. Remote replication is configured as a policy on the archive system, indicating which sets of files should be replicated and to which remote archive systems. File replication is performed asynchronously. Document replication is managed as an adjunct to the audit-log process that identifies newly ingested documents. The identified documents are sent over the Internet Protocol (IP) network to a set of remote sites on the basis of defined policies. Since documents are immutable and never updated, sophisticated block-differencing and complex replication methods are unnecessary. Following loss of an archive system at one site, clients can refer to the redundant archive system or systems and continue to access data remotely via protocols over the IP network. Our system provides document replication by leveraging existing capabilities in GPFS and TSM.

Conclusion

We have taken the approach of designing and developing our archive system using COTS products as a base, extending those products, and integrating them with new innovations to create a highly scalable and flexible archive system. By using COTS-based products, we leverage the significant investments made in these products and benefit from the quality and reliability of these proven components. Our system can help meet both current and future requirements to store billions of documents on petabytes of storage with high performance ingest, search, and retrieval. This design approach can provide significant benefit to organizations as they deploy business-critical archives to store their data for years or decades. For related efforts concerning storage infrastructures, highly available storage systems, or the use of context to enhance search, the reader may consult References [35–37].

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of the Apache Software Foundation, Linus Torvalds, or National Electrical Manufacturers Association in the United States, other countries, or both.

References

1. J. McKnight, T. Asaro, and B. Babineau, "Research Report: Digital Archiving: End-User Survey & Market Forecast 2006–2010," Enterprise Strategy Group, Milford, MA, January 2006; see *http://www.enterprisestrategygroup.com/ViewSecureDocument.asp?ReportID=591&ReportType=Research&ReportField=Attachment2*.
2. GE Fanuc Intelligent Platforms, Inc., "Ruggedization for COTS Systems," white paper; see *http://www.gefanucembedded.com/news-events/whitepapers/1073/AFC-WIK*.
3. F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," *Proceedings of the First USENIX Conference on File and Storage Technologies*, Monterey, CA, January 28–30, 2002, pp. 231–244.
4. IBM Corporation, Tivoli Storage Manager; see *http://www.ibm.com/software/tivoli/products/storage-mgr/*.
5. Apache Software Foundation, Apache Lucene—Overview; see *http://lucene.apache.org/java/docs/*.
6. C. Hertel, "Samba: An Introduction"; see *http://us1.samba.org/samba/docs/SambaIntro.html*.
7. R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network File System," *Proceedings of the Summer USENIX Technical Conference*, Portland, OR, 1985, pp. 119–130.
8. P. J. Leach, "A Common Internet File System (CIFS/1.0) Protocol," *Network Working Group Technical Report*, December 1997.
9. J. Postel and J. Reynolds, "File Transfer Protocol (FTP)," IETF Request for Comments 959, Network Working Group (October 1985); see *http://www.faqs.org/rfcs/rfc959.html*.
10. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1," IETF Request for Comments 2068, Network Working Group (January 1997); see *http://www.ietf.org/rfc/rfc2068.txt*.
11. IBM Corporation, *IBM Tivoli Storage Manager: Using the Application Programming Interface*, Version 5, Release 2, Publication GC32–0793, December 2003; see *http://publib.boulder.ibm.com/tividd/td/TSMC/GC32-0793-02/en_US/PDF/ansa0000.pdf*.
12. Storage Networking Industry Association, XAM; see *http://www.snia.org/tech_activities/standards/curr_standards/xam/*.
13. R. A. Lorie, "Long Term Preservation of Digital Information," *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*, Roanoke, VA, 2001, pp. 346–352.
14. M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, and J. Satran, "The Need for Preservation Aware Storage: A Position Paper," *Operating Syst. Rev.* **41**, No. 1, 19–23 (2007).
15. M. J. Moyer and M. Ahamad, "Generalized Role-Based Access Control," *Proceedings of the 21st International Conference on Distributed Computing Systems*, Mesa, AZ, 2001, pp. 391–398.
16. J. S. Park, G.-J. Ahn, and R. Sandhu, "Role-Based Access Control on the Web Using LDAP," *Proceedings of the Fifteenth Annual Working Conference on Database and Application Security*, Niagara, Ontario, Canada, 2002, pp. 19–30.
17. R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," *Proceedings of the ACM SIGMOD Conference on Management of DataSIGMOD Record* **29**, No. 2, 439–450 (2000).
18. IBM Corporation, OmniFind Discovery Edition; see *http://www.ibm.com/software/data/enterprise-search/omnifind-discovery*.
19. IBM Corporation, IBM DFSMS Product Highlights; see *http://www.ibm.com/systems/storage/software/sms/index.html*.
20. J. P. Gelb, "System-Managed Storage," *IBM Syst. J.* **28**, No. 1, 77–103 (1989).

21. M. Beigi, M. Devarakonda, R. Jain, M. Kaplan, D. Pease, J. Rubas, U. Sharma, and A. Verma, "Policy-Based Information Lifecycle Management in a Large-Scale File System," *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, Washington, DC, June 6–8, 2005, pp. 139–148.
22. "Digital Imaging and Communications in Medicine (DICOM)," Rosslyn, VA, National Electrical Manufacturers Association, 2007; see *http://medical.nema.org/dicom/2007/07_01pu.pdf*.
23. Health Level Seven, Inc., HL7: Health Level Seven; see *http://www.hl7.org/*.
24. T. Götz and O. Suhre, "Design and Implementation of the UIMA Common Analysis System," *IBM Syst. J.* **43**, No. 3, 476–489 (2004).
25. Apache Software Foundation, Apache UIMA; see *http://incubator.apache.org/uima/*.
26. D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, Jr., "Semantic File Systems," *Operating Syst. Rev.* **25**, No. 5, 16–25 (1991).
27. P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton, "Extending Document Management Systems with User-Specific Active Properties," *ACM Trans. Inform. Syst.* **18**, No. 2, 140–170 (2000).
28. U.S. Securities Exchange Commission, "Final Rule: Applicability of CFTC and SEC Customer Protection, Recordkeeping, Reporting, and Bankruptcy Rules and the Securities Investor Protection Act of 1970 to Accounts Holding Security Futures Products"; see *http://www.sec.gov/rules/final/34–46473.htm*.
29. *Sarbanes–Oxley Act of 2002*, HR 3763; see *http://thomas.loc.gov/cgi-bin/query/z?c107:H.R.3763.ENR:%20*.
30. "Summary of the HIPAA Privacy Rule: Office for Civil Rights," U.S. Department of Health and Human Services; see *http://www.hhs.gov/ocr/privacysummary.pdf*.
31. G. Russo, S. Russo, and B. Pirenne, "An Operating System Independent WORM Archival System," *Softw. Pract. Exper.* **25**, No. 5, 521–533 (1995).
32. Z. Peterson and R. Burns, "Ext3cow: A Time-Shifting File System for Regulatory Compliance," *ACM Trans. Storage* **1**, No. 2, 190–212 (2005).
33. IBM Corporation, IBM 3592 Tape Cartridge; see *http://www.ibm.com/systems/storage/media/3592/index.html*.
34. J. Ramanathan, R. J. Cohen, E. Plassmann, and K. Ramamoorthy, "Role of an Auditing and Reporting Service in Compliance Management," *IBM Syst. J.* **46**, No. 2, 305–318 (2007).
35. The OceanStore Project; see *http://oceanstore.cs.berkeley.edu/*.
36. Lots of Copies Keep Stuff Safe (LOCKSS); see *http://www.lockss.org/lockss/Home*.
37. C. A. N. Soules and G. R. Ganger, "Connections: Using Context to Enhance File Search," ACM SIGOPS Operating Systems Review, *Proceedings of the Twentieth ACM Symposium On Operating Systems Principles* SOSP '05, Brighton, UK, pp. 119–132, October 2005.

Received September 21, 2007; accepted for publication October 16, 2007; Internet publication June 3, 2008

Paul L. Bradshaw *IBM Systems and Technology Group, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (paulbrad@us.ibm.com)*. Mr. Bradshaw is the Manager at the IBM Systems and Technology Group (STG) for storage for software architecture and standards. He has worked in the storage management area for more than 20 years. He is also one of the creators of the IBM Tivoli Storage Manager storage management product line (previously known as *ADSM*), and he has managed it through four major versions, which involved the transition of efforts from the IBM Research Division to the IBM Storage Group. He has also managed the IBM Storage Resource Management product group in the IBM Storage Division and led the IBM Strategy and Architecture team. More recently, he was Vice President of Engineering at Semio Corporation, where he focused on content management and categorization. In 2003, he rejoined IBM Research to manage the transition of research projects into the product groups. In 2005, he joined IBM STG in his current role of managing storage software architecture and standards teams.

Karen W. Brannon *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (kbrannon@almaden.ibm.com)*. Dr. Brannon is a member of the Computer Science Department at the IBM Almaden Research Center. Her current interests include health-care informatics and the incorporation of content management principles into storage systems. She has worked for several years in the area of content management systems and has contributed to the IBM Content Manager product. She has also made many key contributions to the IBM DB2* DataLinks technology. Prior to joining the IBM Almaden Research Center, Dr. Brannon worked at the IBM Scientific Center in Palo Alto, California, on numerically intensive computations for circuit simulation and semiconductor process modeling. Dr. Brannon received her Ph.D. degree from the University of California at San Diego.

Thomas Clark *IBM Tivoli Systems, 9000 South Rita Road, Tucson, Arizona 85744 (tkc@us.ibm.com)*. Mr. Clark is an IBM Distinguished Engineer and Chief Architect for IBM Storage Software. He spent more than 11 years at Informix Software developing parallel database systems and joined IBM in 2001. Since joining IBM, his work has focused on file systems, applications in storage, and storage management software. Mr. Clark received a B.S. degree in computer science and mathematics from Mt. Vernon Nazarene University and an M.S. degree in computer science from Portland State University.

Kirby Dahman *IBM Systems and Technology Group, 9000 South Rita Road, Tucson, Arizona 85744 (soybeans@us.ibm.com)*. Mr. Dahman is an IBM Senior Technical Staff Member and Architect for IBM modular storage. He joined IBM in 1980 at the Tucson Development Laboratory and has held staff and management positions in storage systems development and performance. For the past 15 years, he has served in architectural positions and focused on disk, tape, and optical storage for both mainframe and open systems. He is an IBM Master Inventor with numerous patents in storage system design. Mr. Dahman received a B.S. degree in mathematics from the California Institute of Technology and an M.S. degree in mathematics from California State University, and he did postgraduate study in computer science at the University of Illinois.

Sangeeta Doraiswamy *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120*

(dsang@us.ibm.com). Ms. Doraiswamy is a Senior Software Engineer in the Computer Sciences Department at the IBM Almaden Research Laboratory. She received her B.S. degree in mathematical statistics from Lady Sriram College, New Delhi, India, in 1984. After gaining 17 years of industry experience in software product development and managerial roles, including more than 11 years of database internals development at Sybase, Inc. (a relational DBMS vendor), Ms. Doraiswamy joined the IBM Research Division in October 2002. At IBM, she has worked on the IBM WebSphere* v6.0 messaging platform, RFID data cleansing, and mining and indexing of file content to aid in intelligent data storage. Her current interests are in building scalable solutions for mining, indexing, and searching very large document archives and in the area of health informatics.

Linda Duyanovich *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (lduyanov@us.ibm.com)*. Ms. Duyanovich is a Senior Manager for Storage Management and Solutions at the Almaden Research Center. She received an M.S. degree in operations research from Stanford University in 1977, and she has worked for IBM for more than 25 years in a variety of product development, architecture, performance, and product management roles prior to joining the IBM Research Division in 2003. Ms. Duyanovich currently leads several research teams working on new technologies concerning data archive and access, as well as storage and data management.

Bruce L. Hillsberg *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (blh@us.ibm.com)*. Mr. Hillsberg is the Director of Storage Systems Research at the Almaden Research Center and is responsible for IBM storage research activities worldwide. He joined IBM in 1981 and has held numerous technical and management positions throughout his career. Mr. Hillsberg has worked on systems and network management products, middleware integration and ease-of-use initiatives, business-to-business systems, IBM Year 2000 readiness, Internet appliances, and most recently, storage system strategy and storage technology for government applications. Mr. Hillsberg received a B.S. degree in computer science from Syracuse University.

Wayne Hineman *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (hiney@almaden.ibm.com)*. Mr. Hineman joined the IBM Almaden Research Center in 1989 as a systems programmer on mainframe operating systems. He is currently a Senior Software Engineer in the Storage Systems Department, working on advanced-technology storage-intensive solutions. His areas of interest include backup, archive, compliance, and file systems. Mr. Hineman has a B.S. degree in computer science from the University of Nebraska, Lincoln.

Michael Kaczmarski *IBM Tivoli Systems, 9000 South Rita Road, Tucson, Arizona 85744 (kacz@us.ibm.com)*. Mr. Kaczmarski is an IBM Distinguished Engineer, leading product integration efforts in Tivoli as the Chief Integration Architect. Deeply involved in bringing Tivoli Storage Manager to market from its birth in the IBM Research Division, Mr. Kaczmarski worked on storage products for more than 12 years. He has an M.S. degree in computer science from National Technological University and a B.S. degree in management information systems from the University of Arizona.

Bernhard J. (BJ) Klingenberg *IBM Software Group, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (bj.klingenberg@us.ibm.com)*. Mr. Klingenberg joined IBM in 1984 and is a Senior Technical Staff Member in the Tivoli Storage Software organization. He received a B.S. degree computer science from the University of Illinois and an M.S. degree from the University of Arizona. His current interests are in long-term archive, end-to-end management, and storage process management and automation.

Xiaonan Ma *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (xiaonma@us.ibm.com)*. Dr. Ma is a Research Staff Member in the Computer Science Storage Systems Department. He received his Ph.D. degree in electrical engineering from Texas A&M University in 2002. He subsequently joined IBM at the Almaden Research Center where he has worked on advanced object stores and reference storage solutions. His current research interests include advanced archival systems, regulatory compliance, and data security.

Robert Rees *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (rees@almaden.ibm.com)*. Mr. Rees is an IBM Distinguished Engineer in the Storage Systems Department at the IBM Almaden Research Center. He joined IBM in 1981 and has 26 years of experience in storage-related systems. His work has focused on data-protection software, scalable file systems, database management infrastructure, and archive systems. Mr. Rees was the Research Chief Architect for the Tivoli Storage Manager and IBM TotalStorage* SAN File System products.
Mr. Rees received a B.S. degree in computer engineering from the University of California, Santa Cruz.