A Case Study in Distributed Collection Monitoring and Auditing Using the Audit Control Environment (ACE)

Michael Smorul, Joesph JaJa; Institute for Advanced Computer Studies, Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland, USA

Abstract

We describe the deployment of the Audit Control Environment (ACE)[1] on the Chronopolis distributed archive environment. The ACE system provides a scalable, auditable platform to actively ensure the availability and integrity of digital archival holdings over the lifetime of the archive. The core of ACE is a small integrity token issued for each monitored item that is part of a larger, externally auditable cryptographic system. Two components that describe this system, the Audit Manager and Integrity Management Service, have been developed and were released in October 2008.

ACE allows for the policy driven active monitoring of collections on a variety of disk and grid based storage systems. Each collection in ACE is subject to monitoring based on a customizable policy.

The Chronopolis archiving environment consists of three sites located at the University of Maryland(UMD), San Diego Supercomputing Center(SDSC)/UCSD Libraries, and the National Center for Atmospheric Research(NCAR). Data that has been ingested into storage at SDSC is replicated to UMD and NCAR. Current collections from data providers in Chronopolis range from a relatively few large files to collections containing millions of mostly small files. In total, these collections represent over 5.5 million files and 17 Terabytes of data.

With three copies of each object, the problem of ensuring the integrity of each copy across the three sites arises. An ACE Audit Manager at each site provides continuous monitoring of files to ensure no corruption occurs locally. In addition, periodic audits of each collection across the sites are made to ensure that the collection contents are consistent across all three sites.

We describe the deployment and performance of the Audit Manager at each of the three sites. Specifically, we compare peak theoretical ACE performance against varying collections.

ACE Overview

ACE is an integrity monitoring platform based on creating a small-size integrity token for each digital object upon its deposit into the archive (or upon registration of the object of an existing archive). This token is stored either with the object itself or in a registry at the archive as authenticity metadata.

These tokens are linked together through time spans by an auditable third party. For each time interval, cryptographic summary information(CSI) that depends on all the objects registered during that time interval is generated. The summary information is very compact and is size independent of the number or sizes of the objects ingested. The period of each round is currently defined in seconds but can adapted as needed by the archive.

At the end of each day, all CSI's generated are aggregated into a final witness value. This witness value is a single number that is used to verify all CSI's issued during the previous day. The value is expected to be stored in reliable, read-only media, and pubished over the internet. An independent auditor, given a trusted witness, may assert the integrity of all CSIs for a given time period. Once CSIs are certified, they may be used to validate all tokens covered by the summaries. Once tokens are validated, an auditor may assert that any file whose cryptographic digest matches its token has not been tampered with to a high probability.

Regular audits will be continuously conducted, which will make use of the integrity tokens and the summary integrity information to ensure the integrity of both the objects and the integrity information. In our implementation, audits can also be triggered by an archive manager or by a user upon data access. However we are assuming that the auditing services are not allowed to change the content of the archive even if errors are detected. The responsibility for correcting errors is left to the archive administrator after being alerted by the auditing service.

The ACE system consists of two components, first is an Integrity Management Service(IMS) which gathers token requests into rounds and generates Integrity Tokens(IT) at the end of each round. The IMS is also responsible for publishing nightly witness values. The University of Maryland currently hosts a publically available IMS for any party to use. The second component of ACE are multiple, independent Audit Managers(AM) that are installed locally at archives and periodically check the integrity of monitored objects according to a locally defined policy.

ACE Workflow

Two different workflows have been implemented in the first release of ACE. The first is a token registration workflow where new ITs are issued from an IMS. This occurs only once when a new file is detected. The second workflow is a validation workflow where previously issued tokens are used to validate the integrity of files and the stored digest of those files.

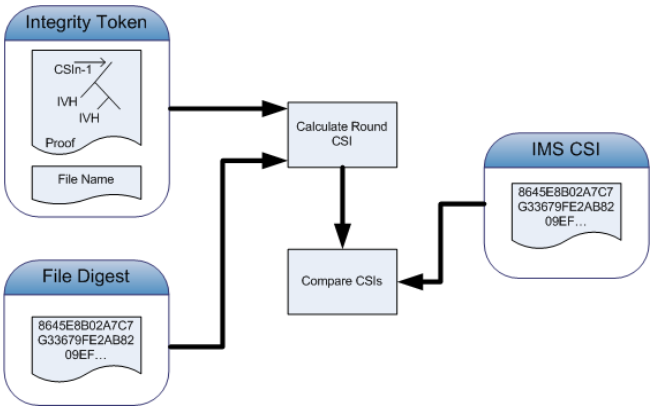
Both registration and validation are performed by an AM. This AM runs physically close to the data that is to be monitored. It is designed to have bit-level access to the data so that it may read all monitored files and generate cryptographic digests across those files. The AM defines policy that determines when an audit of holdings will occur. During the course of an audit, the AM will invoke both of these workflows unbeknownst to the end user.

Token Issuing

Tokens are issued as part of the registration workflow described earlier. The AM generates a SHA-256 digest of the file to be monitored. This generated digest and file name is submitted to the IMS for inclusion in the current round. It should be noted the IMS can handle requests for any digest, not only SHA-256.

The submitted token is aggregated with other requests during the same time interval. During an aggregation round, the hashes of all the objects submitted for registration as well as random hashes as necessary are aggregated using an authentication tree such as the Merkle's tree [2]. This resulting CSI is stored in a database to be later used for witness generation and IT validation. For each request, the IMS generates an IT which is returned to the AM. In the AM, communication with the IMS is handled out of band of the main audit process. This allows the AM to send batch requests and not to block on the IMS communication should the AM operate in a high latency environment. The token request flow is shown in Figure 1.

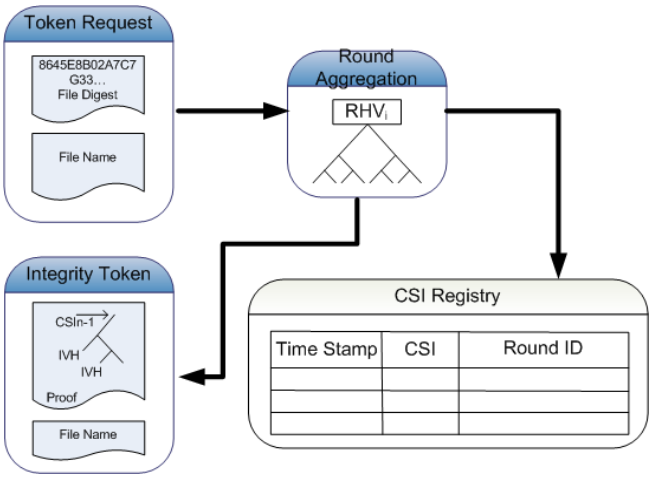
Figure 1: Token Request Flow



File and Token Validation

File and token validation occur on the AM subject to a specified policy. This policy may vary between collections. Periodically, each monitored item is read, a digest is calculated and compared with the stored value.

Figure 2: Validation Workflow



Token validation requires showing the stored digest has not been altered. Validation is done by linking the stored digest to an IMS CSI. This involves calculating the round CSI using the authentication tree stored in the IT and the stored item's digest. The IMS is then queried to retrieve the CSI for the round where the IT was issued. The returned CSI is compared to the calculated CSI and if it matches, the IT and digest are considered trustworthy to a high probability. As with registration, the AM has a batching mechanism to group CSI requests and minimize calls to the IMS.

Chronopolis Description

The Chronopolis Preservation Environment is a consortium of three data storage partners, and several data providers. Data storage partners include the University of Maryland(UMD), San Diego Supercomputing Center(SDSC)/UCSD Libraries, and the National Center for Atmospheric Research(NCAR).

Currently, Chronopolis uses the Storage Resource Broker (SRB)[3] to provide data grid services. The SRB provides a unified namespace and common functionality for data access and placement while abstracting access to underlying data resources. Each Chronopolis partner has an SRB installation backed by a metadata catalog(MCAT) and disk storage. These SRB installations are federated, so each site is able to see shared data at other partner sites.

Collections in Chronopolis are provided by a set of data providers. Data first staged into the SRB installation at SDSC, then periodically replicated to resources at UMD and NCAR. Once data is replicated, it is detected and monitored by installations of the ACE Audit Manager at each partner site.

Table 1: Chronopolis Collections

Provider	Files	Directories	Size(GB)
CDL	46,762	28	4,291
SIO-GDC	197,718	5,230	815
ICPSR	4,830,625	95,580	6,957
NC-State	608,424	42,207	5,465

Table 1 shows that collections supplied by data providers vary both in after file size and number of object in the collection.

ACE Audit Manager Installation

Each partner in the Chronopolis grid has an installation of the ACE Audit Manager locally auditing their collections. While it is possible to access the entire Chronopolis holdings from one partner, the effects of latency make remote auditing unfeasible. Therefore, each AM installation was configured to monitor only the local files.

Installation at each site required a unix compatible OS, MySQL database server, and Java 1.6 or greater. These requirements are standard for a lot of database driven library software including Fedora[4] and DSPace[5]. After installation, all additional configurations are handled through a web interface. This includes defining collection location, audit policy, and viewing audit logs and results.

Audit Manager Performance

The following is a description of the performance of the AM on the UMIACS Chronopolis storage node. First we describe the peak performance provided by the AM software, performance of the UMIACS and finally, observed audit performance.

Performance of the Audit Manager is affected by several factors. These are described below:

- Storage metadata performance. The AM performs several queries to list files in collections, and to determine on which storage resource they are held. For small files, these metadata operations may require more time than actually transferring a file.
- Data transfer speed. The rate at which data is read by the AM from a storage resource. In the case of distributed storage, there may be additional overhead for remotely connecting to resources holding data
- Digest Calculation. The AM calculates a SHA-256 digest for each object processed.
- ACE metadata overhead. Each registration operation generates several database operations to save an object's properties. Log events are saved for each state change an object generates. In addition, each audit requires that timestamps are updated for each tracked object.

Audit Manager Peak Performance

The throughput of ACE is determined by two factors, first AM metadata operations. These operations include item metadata retrieval, token registration, and event logging. Performance for these functions are determined by database performance. Webservice calls to the IMS are handled in a separate thread to allow for batched requests and to mitigate the effects of network latency. The second factor that affects AM performance is the time required to calculate cryptographic digests for all monitored data.

Two tests were performed to determine the effects of database overhead and digest calculation. A test collection containing 1.25million simulated files was created. Each file in this collection was 1MB in size, separated into 10 files per directory, at a depth of three directories from root of the collection. Digest calculation was performed serially with object registration. In most production drivers, file reading and digest calculation is performed in multiple, separate threads to mitigate storage system overhead.

The two tables show the registration and audit speed of an audit manager. The first test was performed registration and auditing was performed against simulated digests, the second test executed digest calculation on block of data stored in memory.

Table 2: Collection Registration

Digest source	Time(h)	Files/s
Auto-generated	3:06	122
Calculated	5:07	67.8

Table 3: Collection Auditing

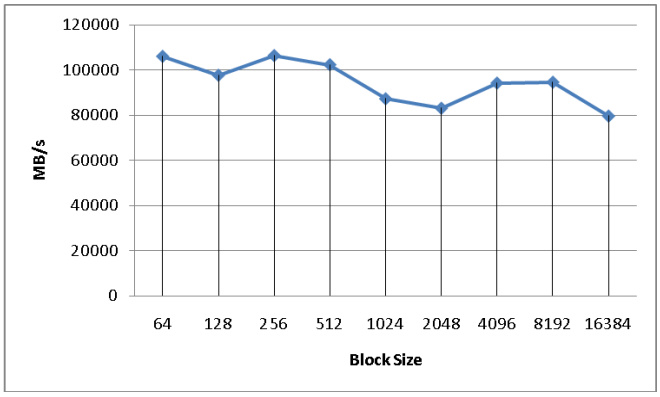
Digest source	Time(h)	Files/s
Auto-generated	1:15	277
Calculated	4:30	77.2

From the tables, we notice that registration time is more than 50% greater than auditing time; however this difference shrinks when the Audit Manager required to perform digest calculation. We will see in the following observed performance that in practice other limitations occur before AM performance becomes a bottleneck.

Storage Node Performance

UMD has several servers configured as storage nodes for Chronopolis. Each node is connected to two Apple XRaids by a 2Gbps storage area network(SAN) connection. Storage nodes are accessible through a 1 1Gbps Ethernet connection. Each storage node was configured to access 4 independent raid arrays (two arrays per XRaid). IOZone testing on the storage nodes showed an individual array is able to read data at close to 100MB/s. As an audit manager has four arrays, this easily allows for saturation of the 1Gbps Ethernet connection. The following graph shows the performance characteristics of the raids.

Figure 3: Raid Performance



During replication, a collection was assigned to a storage node; files were then dispersed equally across all arrays attached to that node. This file striping was performed so that the raids would not become a bottleneck when managing collections containing many small files.

Observed Performance

Final testing was performed against live collections at the UMD site. Each collection was audited while no other activity occurring on the storage nodes. Total audit time for all collections was a little over one week. During auditing we observed the largest bottleneck occurred when processing small files. This bottleneck is due to overhead involved in querying the MCAT, followed by opening connections to data nodes. The AM driver for the SRB was configured to use 5 threads to retrieve and calculate digests in parallel. The Audit Manager was installed on a two processor, 8 cpu core server with a 1Gbps connection to the storage nodes.

The following table shows the resulting throughput that was achieved during an audit.

Table 4: Collection auditing throughput

Provider	Time(h)	Files/s	Bandwidth(MB/s)
CDL	20:32	.63	59.44
SIO-GDC	6:49	8.05	34.00
ICPSR	122:48	10.93	16.11
NC-State	32:14	5.24	48.22

While audit times vary widely, we see that per object throughput on the Audit Manager is fewer than 20% available transaction capacity. For files with large collections, Chronopolis was able to supply data to the audit manager at slightly less than 50% of the 128MB/s available network capacity.

The most interesting characteristic of Chronopolis is how small files affect the overall audit rate. The following table shows file size compared to collection audit rate.

Table 5: Average File Size vs Throughput

Collection	Average Size(MB)	Bandwidth(MB/s)
ICPSR	1.47	16.11
SIO-GDC	4.22	34.00
NC State	9.19	48.22
CDL	93.96	59.44

While the preservation environment shows some bottlenecks when auditing small files, an audit policy could be devised where a collection containing small files was audited in parallel with a collection containing large files. Since the large file collection would not require significant MCAT or AM resources, overall audit performance could be significantly increased.

Conclusion

In this paper, we describe the ACE Audit Manager as it is installed on the Chronopolis preservation environment and the resulting performance characteristics resulting from that installation. ACE has been shown to provide an easy to use environment while providing a high performance integrity management platform.

References

[1] Song, S. and JaJa, J. ACE: A Novel Software Platform to Ensure the Integrity of Long Term Archives. in Archiving 2007. 2007: IS&T.

[2] Ralph Merkle. “Protocols for public key cryptosystems,” In Proceedings of the 1980 Symposium on Security and Privacy, IEEE Computer Society Press, 1980, pp 122–133.

[3] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC Storage Resource Broker. In Procs. of CASCON'98, Toronto, Canada, 1998

[4] Staples, Thornton, and Ross Wayland, "Virginia Dons Fedora: A prototype for a digital object repository," D- LIb Magazine, July 2000

[5] DSpace, http://dspace.org

[6] Iozone filesystem benchmark, http://www.iozone.org

Author Biography

Mike Smorul received his BS in computer science from the University Of Maryland. He has a background in network and high performance computing system administration. More recently, he has worked as lead programmer for the UMIACS ADAPT project. Current project include developing a modular set of tools to aid in ingestion and integrity monitoring in support of long term stewardship of digital objects.

Joseph JaJa currently holds the position of Professor of Electrical and Computer Engineering with a joint appointment at the Institute for Advanced Computer Studies at the University of Maryland, College Park. Dr. JaJa received his Ph.D. degree in Applied Mathematics from Harvard University and has since published extensively in a number of areas including parallel and distributed computing, combinatorial optimization, algebraic complexity ,VLSI architectures, and data-intensive computing. His current research interests are in parallel algorithms, digital preservation, and scientific visualization of large scale data. Dr. JaJa has received numerous awards including the IEEE Fellow Award in 1996, the 1997 R&D Award for the development of software for tuning parallel programs, and the ACM Fellow Award in 2000. He served on several editorial boards, and is currently serving as a subject area editor for the Journal of Parallel and Distributed Computing and as an editor for the International Journal of Foundations of Computer Science.