

The International Journal of Digital Curation

Volume 8, Issue 1 | 2013

Distributed Digital Preservation in the Cloud

David S.H. Rosenthal and Daniel L. Vargas,
LOCKSS Program,
Stanford University Libraries

Abstract

The LOCKSS system is a leading technology in the field of Distributed Digital Preservation. Libraries run LOCKSS boxes to collect and preserve content published on the Web in PC servers with local disk storage. They form nodes in a network that continually audits their content and repairs any damage. Libraries wondered whether they could use cloud storage for their LOCKSS boxes instead of local disks. We review the possible configurations, evaluate their technical feasibility, assess their economic feasibility, report on an experiment in which we ran a production LOCKSS box in Amazon's cloud service, and describe some simulations of future costs of cloud and local storage. We conclude that current cloud storage services are not cost-competitive with local hardware for long term storage, including for LOCKSS boxes.





Introduction

The LOCKSS¹ Program at the Stanford University Libraries pioneered the concepts of Distributed Digital Preservation about 14 years ago by building a peer-to-peer network of LOCKSS boxes in which libraries could collect and preserve content published on the Web. About 150 libraries currently run LOCKSS boxes, preserving e-journals and e-books in the Global LOCKSS Network (GLN), and databases, government documents, special collections and other content in Private LOCKSS Networks (PLNs). These boxes are typically modest PC servers with substantial amounts of local disk storage.

Some of these libraries asked whether they could use “affordable cloud storage” for their LOCKSS boxes. In this paper, we first describe the relevant features of typical cloud services, then discuss the various possible technical architectures by which a LOCKSS box could use such a service, ruling some out for performance reasons. Applying the typical cloud service’s charging model to the remaining architectures rules out others for economic reasons. We then describe an experiment in which we implemented the most cost-effective architecture and ran a production LOCKSS box to preserve GLN content in Amazon’s cloud service for several months. We report on the pricing history of cloud storage, using it and the costs from the experiment to compare the projected future costs of storing content preserved by LOCKSS boxes in cloud and local storage. We conclude with discussion of future work, some of which is already under way.


Features of Cloud Technology

Amazon, as a typical cloud service, provides the following components relevant here:

- A *compute service*, in which a customer can run a number of virtual machine instances. These instances have the normal resources of a physical PC, but their (virtual) disk storage is limited in size and is “evanescent” - vanishing without trace when the instance stops for any reason.² Amazon calls this service Elastic Cloud Computing (EC2).
- An *object storage service*, in which a customer can store arbitrary-sized byte strings. These objects persist, are named by URLs, and are accessed via an API using HTTP PUT, GET and HEAD requests. Depending on the service, the object store may be more or less reliable. Amazon calls this service Simple Storage Service (S3), and offers two levels of reliability. The standard level is designed for 11 nines durability, and Reduced Reliability Storage (S3-RRS) is designed for 4 nines durability.
- A *block storage service*, in which a customer can store a file system that can be mounted by virtual machine instances running in the compute service. These file systems persist across virtual machine restarts, but are designed for performance rather than extreme reliability. Facilities are normally provided by which snapshots of the file systems can be reliably preserved as objects in

¹ Lots Of Copies Keep Stuff Safe, a trademark of Stanford University.

² This is true for instances backed by their Simple Storage Service (S3), not by their Elastic Block Storage (EBS). But EBS is not reliable enough; see Experimental Results below.



the object storage service. Amazon calls this service Elastic Block Storage (EBS).

Technical Aspects

It is important to understand the interfaces between these components. Their performance characteristics determine the technical viability of the various system architectures possible for LOCKSS boxes in the cloud:

- The interface between the outside world and the compute service, which is typically as slow as the corresponding interface for a local computer;
- The interface between the outside world and the storage service, which is typically as slow as the corresponding interface for a local computer;
- The interface between the compute service and the object storage service, which is typically much slower than the interface between a local computer and its disk, or a local storage area network;
- The interface between the compute service and the block storage service, which is typically about the same speed as the interface between a local computer and its disk;
- The interface between the block storage service and the object storage service, whose performance is typically not critical.

Economic Aspects

The charging models for the various services determine the economic viability of the various architectures possible for LOCKSS boxes in the cloud:

- The *compute service* typically levies charges based on the size of the virtual machine and the time for which it is running;
- The *object storage service* typically levies charges based on the total size of the objects stored, the time for which they are stored, the GET, PUT and other HTTP operations invoked on them, and the durability level at which they are stored;
- The *block storage service* typically levies charges based on the total amount of storage consumed by the file systems, the time for which it is stored, and the number of I/O transactions between it and the compute service;
- The service typically also levies charges based on the amounts of data transferred in each direction across the interface between the compute and storage services and the outside world.

Technical Architectures

A LOCKSS box consists of some storage holding the preserved content in a repository, and a daemon – software running in a computer that accesses the repository to perform the technical preservation functions described in the OAIS

Reference Model (ISO, [2003](#)) including ingest, dissemination, integrity checking and management as part of a peer-to-peer network (Maniatis, Rousopoulos, Giuli, Rosenthal & Baker, [2005](#)).


The following architectures are feasible using current compute and storage service capabilities:

- **A LOCKSS daemon running in a local machine with storage in an object storage service.** We have modified the LOCKSS daemon's repository to use the S3 object storage interface and run experiments against S3, and also against Eucalyptus and the Internet Archive's object storage service, both of which are mostly compatible with S3. Extracting the content from S3 via its Web interface for integrity checking is too slow to be viable; the LOCKSS protocol requires fairly homogeneous I/O performance among the peers.
- **A LOCKSS daemon running in a compute service virtual machine instance with storage in the object storage service.** Even from the compute service the I/O performance of the object storage service is much slower than a local disk, violating the homogeneous I/O performance requirement.
- **A LOCKSS daemon running in a compute service virtual machine instance with storage in the virtual machine's disk.** We implemented this architecture in EC2 with an S3-backed virtual machine. It is disqualified for two reasons: firstly, if the instance stopped for any reason the preserved content would be entirely lost; secondly, the total space available is too small for practical use.
- **A LOCKSS daemon running in a compute service virtual machine instance with storage in the block storage service.** We implemented this architecture in EC2 with EBS storage, but concerns about the unknown reliability of EC2/EBS and the cost of recovering from a total loss of EBS data over the network from other LOCKSS boxes led us to prefer the next architecture.
- **A LOCKSS daemon running in a compute service virtual machine instance with storage in the block storage service and a snapshot preserved in the object storage service at regular intervals.** We implemented this architecture in Amazon's EC2 with EBS storage and snapshots in S3, and ran it for several months. The results are described in the Experimental Results section below.

We have examined some other architectures that require enhanced capabilities from the object storage service. They are discussed briefly in the Future Technology section below.

Economic Considerations

We chose Amazon for our experiment because Amazon has dominated the market for cloud services, with an estimated market share above 90%, and remains the price leader in storage (see Table 4). Applying the Amazon charging model to our preferred architecture, the following costs could be incurred:

- 
- EC2 charges for the virtual machine running the box;
 - EBS charges for storing the content;
 - EBS charges for I/Os to and from the content;
 - EBS charges for storing the backup snapshot;
 - Charges for inbound network usage for collecting the content – in Amazon’s case, inbound network usage is free;
 - Charges for outbound network usage for disseminating the content;
 - Charges for bidirectional network usage for the LCAP voting protocol – in Amazon’s case, inbound network usage is free;
 - Charges for outbound network usage for repairing content at other boxes;
 - Charges for inbound network usage for repairing content at this box from other boxes – in Amazon’s case, inbound network usage is free.

Experimental Results


We implemented an Amazon Machine Instance (AMI) containing our recommended configuration for a LOCKSS box. It is backed by S3 and configured by a bootbucket, which specifies these system parameters: AWS Access Key, AWS Secret Access Key, S3 bucket backing the image, name of .tar.gz archive in the S3 bucket containing the LOCKSS configuration, and a comma delineated list of volume ids to automatically attach and mount from EBS on startup.

The file systems it mounts from EBS contain the preserved content. Because EBS is not reliable over the long term, a snapshot of each entire EBS file system is preserved in S3. This snapshot is updated at regular intervals. Note that only content changed since the previous snapshot is transferred in this process.

We configured a LOCKSS box using this AMI to preserve a sample of open access content from the LOCKSS GLN so that it participated fully in the GLN, although with less content (82GB) than the median GLN LOCKSS box (1.58TB). Snapshots of the EBS file system containing this content were preserved in S3 at 12-hourly intervals. Once a snapshot had been successfully stored, older snapshots were deleted leaving two snapshots at the most.

We ran this box using a separate Amazon account created for the experiment and monitored the costs incurred by the account every week for 14 weeks. The results are shown in Table 1.

The experimental LOCKSS box was empty at the start, and was ingesting content during the experiment. During week 12 it finished ingesting, as reflected in the decrease in bandwidth charges, primarily for writing the new data to EBS and the snapshots to S3.



Week	Compute \$	Storage \$	Bandwidth \$	Total \$
2	53.76	2.43	0.16	56.36
3	53.76	2.65	0.16	56.57
4	53.76	2.83	0.18	56.77
5	53.76	2.98	0.19	56.93
6	53.76	3.13	0.22	57.10
7	53.76	3.14	0.28	57.18
8	53.76	2.94	0.24	56.94
9	53.76	3.58	0.25	57.59
10	53.76	3.71	0.44	57.92
11	53.76	3.86	0.39	58.02
12	53.76	3.24	0.45	57.45
13	53.76	3.99	0.33	58.08
14	53.76	4.00	0.40	58.16

Table 1. Costs incurred by experimental LOCKSS box in AWS. We eliminated Week 1 because there were some startup effects.

This experimental box was not representative in several ways:

- It preserved less content than the median LOCKSS box in the GLN and thus consumed less storage and less bandwidth than a production box.
- Out of caution the experimental box maintained two snapshots of its EBS volumes in S3; only one is necessary.
- As we were only running the experimental box for a short period, we used an on-demand instance so our compute charges were higher than they should have been. A production box would use a reserved instance, which requires an up-front payment and a commitment for either one or three years. Since LOCKSS boxes are continually active, this would need to be a heavy-utilization reserved instance with a three-year commitment. The May 2012 cost for a suitable instance would be \$1200 plus \$0.052/hr (Amazon, [2012](#)).
- There is the question of over-provisioning storage to cope with growth. As its collection grows, the disks of a physical LOCKSS box will fill up. Eventually, additional disks must be provided, and at that time it makes sense to buy the biggest disks available. Thus physical LOCKSS boxes add storage in large discrete increments, and are typically over-provisioned by at least half the size of their most recent disk addition. In fact, the median LOCKSS box is currently over-provisioned by about 2TB. In the cloud, storage can be added in smaller units more frequently, so the amount of over-provisioning can be less. We are not able to quantify this effect exactly.

We adjusted the results from Table 1 to model a production LOCKSS box having already ingested its content and using a reserved instance. To illustrate the range of costs implied by different amounts of over-provisioning, we modelled both minimal over-provisioning (Table 2) and over-provisioning matching that of the median LOCKSS box (Table 3).

Week	Compute \$	Storage \$	Bandwidth \$	Total \$
2	8.74	37.63	3.18	49.54
3	8.74	39.75	3.10	51.59
4	8.74	41.47	3.52	53.72
5	8.74	42.91	3.60	55.25
6	8.74	44.35	4.17	57.25
7	8.74	44.46	5.31	58.51
8	8.74	42.54	4.63	55.90
9	8.74	48.66	4.91	62.31
10	8.74	49.99	8.53	67.26
11	8.74	51.44	7.61	67.78
12	8.74	45.38	8.75	62.87
13	8.74	52.64	6.35	67.72
14	8.74	52.72	7.81	69.26

Table 2. Costs that would have been incurred by median LOCKSS box in AWS with minimal over-provisioning.

Week	Compute \$	Storage \$	Bandwidth \$	Total \$
2	8.74	85.14	3.18	97.05
3	8.74	87.26	3.10	99.10
4	8.74	88.98	3.52	101.23
5	8.74	90.42	3.60	102.75
6	8.74	91.86	4.17	104.76
7	8.74	91.97	5.31	106.01
8	8.74	90.05	4.63	103.41
9	8.74	96.17	4.91	109.82
10	8.74	97.50	8.53	114.77
11	8.74	98.95	7.61	115.29
12	8.74	92.89	8.75	110.37
13	8.74	100.14	6.35	115.23
14	8.74	100.23	7.81	116.77

Table 3. Costs that would have been incurred by median LOCKSS box in AWS with matching over-provisioning.

The projected total three-year cost in the minimal over-provisioning case would be approximately \$8,800, and in the matching over-provisioning case approximately \$18,100. We would expect actual costs between these two, and probably closer to the minimum. Current purchase cost for the matching over-provisioning case is about \$1,500. Thus non-hardware costs for the local case would have to be over 5/6 (minimal) or 11/12 (matching) of the three-year Total Cost of Ownership (TCO) for cloud to be cheaper.

Projecting the Cost of Cloud Storage

Table 4 shows the history of the prices charged by several major storage services. It shows that most have dropped less than 10% per year. This is in stark contrast with the 30-year history of raw disk prices, which have dropped at least 30% per year, as predicted by Kryder's Law (Walter, [2005](#)).

Service	Launch Date	Launch Price \$/GB/mo	December 2012 Price \$/GB/mo	Price Drop Per Year
Amazon S3	03/06	0.15	0.095	7%
Rackspace	05/08	0.15	0.100	9%
Azure	11/09	0.15	0.125	3%
Google	10/11	0.13	0.095	27%

Table 4. Price history of the base tier of some leading cloud storage services.

This comparison is somewhat unfair to Amazon S3. Amazon has used the decrease in storage costs to implement a tiered pricing model; over time larger and larger tiers with lower prices have been introduced. The price of the largest tier, now 5PB, has dropped about 10% per year; prices of each tier, once introduced, have been stable or dropped slowly. Google's rapid price drop is the anomalous result of its recent introduction and a late November 2012 price war.

Nevertheless, it is clear that the benefits of the decrease in raw storage prices are not going to cloud storage customers. Backblaze provides unlimited backup for personal computers for a fixed price, currently \$5/mo. Before the floods in Thailand, they documented the build cost of their custom 4U 135TB storage pods at under \$8K (Nufire, [2011](#)); using current retail prices for 3TB drives would make this \$7.1K. Dividing the hardware into 3 RAID6 arrays gives 117TB of usable capacity. Given S3's dominance of the cloud storage market, and thus purchasing volumes, it is very unlikely that their costs are higher than those of Backblaze. Despite this, 117TB in S3-RRS costs more than \$6.9K/mo. In the first month, an S3-RRS customer would pay almost as much as it would cost to buy the necessary hardware.

Work is under way to build economic models of long term data storage. Based on an initial model, and using interest rates from the last 20 years, Figure 1 compares the endowment in current dollars to fund storing 117TB for 100 years at varying rates of

←—————→

annual price drop for S3 and 3 RAID6 copies in Backblaze's storage pods. Both alternatives provide three geographically separate replicas in storage protected against more than one simultaneous disk failure.

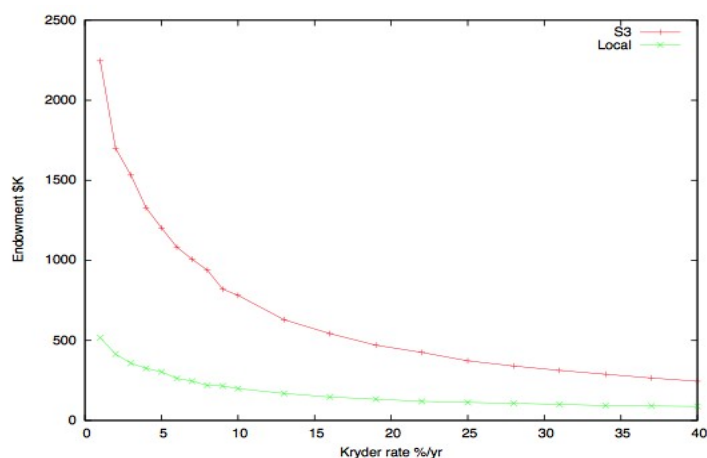


Figure 1. Projected 117TB 100 year endowment.

Shown in red, the cost of using Amazon's S3 starting at its current prices.

Shown in green, the cost of maintaining three copies in local storage using Backblaze's costs for the hardware and assuming (a) that the non-media costs are two-thirds of the total (Moore, D'Aoust, McDonald & Miller, 2007) and (b) that drives are replaced every three years. Note that Backblaze reports much lower non-media costs. Note also that non-media storage costs are typically per-drive or per-server, so they scale with the drive capacity.

It is evident that, in the long term, the rate of price drop dominates all other parameters. Note that Amazon S3 is not competitive with local storage at any Kryder rate. Unless the rate at which storage service price drops comes into line with that of raw media costs, these services cannot compete with local provision for long term storage; the endowment needed at the historic 7% rate for S3 is more than seven times that needed at the disk industry's roadmap projection of 20% rate for local storage.

Future Work

This investigation led to further work in both economics and technology.

Economics

Comparing the economics of local storage (which has both purchase and running costs) with cloud storage (which has only running costs) involves comparing expenditures through time. The standard technique for doing so is Discounted Cash Flow (DCF), which allows a series of incomes or expenditures through time to be reduced to a Net Present Value, subject to an assumed constant interest rate.

Recent research has thrown serious doubt upon both the practical usefulness and theoretical basis of DCF. Its practical usefulness is suspect because it involves choosing a discount rate – an interest rate that will apply for the duration. In practice, people applying DCF choose unrealistically high interest rates, making investment in long term projects excessively difficult (Haldane & Davies, [2011](#)). Its theoretical basis is suspect because the single constant interest rate averages out the effect of periods of very high or (as now) very low interest rates. This would be correct if the outcome was linearly related to the interest rate, but it is not (Doynes, Farmer & Geanakoplos, [2009](#)).

It became apparent that realistic projections of the costs of future storage technologies required more sophisticated techniques than DCF. An effort to build Monte Carlo models of storage costs is now under way, with participants from the LOCKSS Program, University of California Santa Cruz, Stony Brook University and Network Appliance. The early stages of this work involved two prototype economic models, one short term and one long term (Rosenthal, Rosenthal, Miller, Adams, Storer & Zadok, [2012](#)); the long term model was used to produce Figure 1.

Technology

A study of access patterns to data in digital archives (Adams, Miller & Storer, [2011](#)) showed that the majority of read operations were for integrity checking. This is also true of LOCKSS boxes. Checking the integrity of content in current cloud storage systems is problematic. Although it is possible to use the HTTP HEAD operation to ask the service for the checksum of an object, this is not useful as an integrity check (Rosenthal, [2010](#)). Although one might hope that the service would re-compute the hash for each HEAD request, it need not do so. The service could respond with a correct hash to this request by remembering the hash it computed when the object was created, without ever storing the object.

Thus, an integrity check has no option but to retrieve the entire object from the object storage service and compute its hash anew. This is slow and, if the system is running outside the compute service, expensive.

A simple enhancement to the object storage API would solve this problem. A client could supply a nonce with the HEAD request, which the object storage service would prepend to the object before hashing it (Rosenthal, [2010](#)). In this way any of a range of integrity check technologies (Maniatis, Roussopoulos, Giuli, Rosenthal & Baker, [2005](#); Shah, Baker, Mogul & Swaminathan, [2007](#); Song & Jaja, [2009](#)) could force the object storage service to prove that it currently contains a good copy of the object, without the need to retrieve it.

Our experiments suggest that, with some relatively simple additions to the LOCKSS daemon, this enhancement would make the following architecture economically feasible: A LOCKSS daemon running in a local machine with storage in an object storage service.

This would have two potential advantages. First, it would eliminate the need to pay to store the data twice (i.e. once in EBS for performance and once in S3 for reliability), as we did in our experiment. Second, it would eliminate the need for

←—————→
 over-provisioning storage to amortize the cost of adding storage over a reasonable amount of increased data.

Based on the costs incurred during our experiment after adjustment, Table 5 shows an estimate of the cloud service costs that would have been incurred by this configuration of a median LOCKSS box. This should be compared with Table 3.

Week	Compute \$	Storage \$	Bandwidth \$	Total \$
2	8.74	47.01	0.71	49.54
3	8.74	47.82	0.73	54.36
4	8.74	48.63	0.94	56.46
5	8.74	49.45	1.00	57.29
6	8.74	50.26	1.18	59.18
7	8.74	51.07	1.52	61.33
8	8.74	51.88	1.32	61.94
9	8.74	52.70	1.53	62.96
10	8.74	53.51	3.19	65.44
11	8.74	54.32	2.92	65.98
12	8.74	55.13	3.57	67.44
13	8.74	55.94	2.71	67.39
14	8.74	56.76	3.31	68.80

Table 5. Costs that would have been incurred by median LOCKSS box with storage in S3 using suggested API. These numbers assume that the box started in Week 1 with the 1.58TB content of the median box, and ingested at the same rate, governed by per-publisher crawl rate limits, as our experimental box.

Whether this architecture would deliver adequate performance would depend on the performance of the underlying service in computing the object's hashes. What is striking, however, is that using S3 directly is not significantly cheaper than using EBS backed by S3 with minimal over-provisioning. The compression and de-duplication capabilities of the mechanism for preserving snapshots of EBS volumes in S3 are remarkably effective at reducing the cost of doing so. Thus, given the performance questions surrounding using S3 directly, and the need for an API enhancement to address them, there is no compelling reason to pursue this architecture. Instead, the next step would rather be to ease the task of adding storage to the EBS volumes, so that this would be done frequently and thus the costs of over-provisioning minimized.

Conclusion

The 30-year history of raw disk costs shows a drop of at least 30% per year. The history of cloud storage costs from commercial providers shows that they drop, at most, by 3% per year. Until there is a radical change in one or other of these cost curves it clear that cloud storage is not even close to cost-competitive with local disk

storage for long term preservation purposes in general, and LOCKSS boxes in particular.

This makes the possible technical architectures by which LOCKSS boxes could use cloud storage irrelevant. Nevertheless, we have implemented and tested the most cost-effective architecture for a LOCKSS box in the current Amazon environment, and have made this implementation available for use by anyone who disagrees with our conclusions.


We repeat our earlier (Rosenthal, 2010) proposal for a simple extension to the current Amazon S3 API that would greatly improve the suitability of S3 and similar services, such as private cloud implementations, for digital preservation.

Acknowledgements

This work was performed under contract GA10C0061 from the Library of Congress' NDIIPP program. Special thanks are due to Leslie Johnston and Jane Mandelbaum of the Library of Congress; to Tom Lipkis and Thib Guicherd-Callin of the LOCKSS Program; and to Ethan Miller, Ian Adams and Daniel Rosenthal of University of California, Santa Cruz.

References

- Adams, I.F., Miller, E.L. & Storer, M.W. (2011). *Analysis of workload behavior in scientific and historical long- term data repositories*. Technical Report UCSC-SSRC-11-01, University of California, Santa Cruz. Retrieved from <http://www.ssrc.ucsc.edu/Papers/ssrcr-11-01.pdf>
- Amazon. (2012). *Amazon EC2 reserved instances*. Retrieved from <http://aws.amazon.com/ec2/reserved-instances/#3>
- Doyne, J., Farmer, J.D. & Geanakoplos, J. (2009). *Hyperbolic discounting is rational: Valuing the far future with uncertain discount rates*. Technical Report 1719, Cowles Foundation, Yale University. Retrieved from <http://cowles.econ.yale.edu/P/cd/d17a/d1719.pdf>
- Haldane, A.G. & Davies, R. (2011). The short long. *New Paradigms in Money and Finance?* Retrieved from <http://www.bankofengland.co.uk/publications/speeches/2011/speech495.pdf>
- ISO. (2003). Reference model for an open archival in formation system: ISO 14721:2003. Retrieved from http://www.iso.org/iso/catalogue_detail.htm?csnumber=24683
- Maniatis, P., Roussopoulos, M., Giuli, T.J., Rosenthal, D.S.H. & Baker, M.G. (2005). The LOCKSS peer-to-peer digital preservation system. *ACM Transactions on Computer Systems*, 23(1), 2–50. doi:10.1145/1047915.1047917

- 
- Moore, R.L., D'Aoust, J., Robert, H., McDonald, R.H. & Minor, D. (2007). Disk and tape storage cost models. *Archiving 2007*.
<http://www.imaging.org/IST/store/epub.cfm?abstrid=34413>
- Nufire, T. (2011). *Petabytes on a budget v2.0: Revealing more secrets*. Retrieved from
<http://blog.backblaze.com/2011/07/20/petabytes-on-a-budget-v2-0revealing-more-secrets>
- Rosenthal, D.S.H., Rosenthal, D.C., Miller, E.L., Adams, I.F., Storer, M.W. & Zadok, E. (2012). The economics of long-term digital storage. Paper presented at The Memory of the World in the Digital Age Conference, Vancouver, BC. Retrieved from
<http://www.lockss.org/locksswp/wp-content/uploads/2012/09/unesco2012.pdf>
- Rosenthal, D.S.H. (2010). LOCKSS: Lots Of Copies Keep Stuff Safe. Paper presented at NIST Digital Preservation Interoperability Framework Workshop. Retrieved from
<http://lockss.org/locksswiki/files/NIST2010.pdf>
- Shah, M.A., Baker, M.G., Mogul, J.C. & Swaminathan, R. (2007). Auditing to keep online storage services honest. *HOTOS XI: 11th Workshop on Hot Topics in Operating Systems*. Retrieved from
http://www.usenix.org/events/hotos07/tech/full_papers/shah/shah_html/
- Song, S. & Jaja, J. (2009). Techniques to audit and certify the long-term integrity of digital archives. *International Journal on Digital Libraries*, 10(2-3).
[doi.10.1007/s00799-009-0056-2](https://doi.org/10.1007/s00799-009-0056-2)
- Walter, C. (2005). Kryder's Law. *Scientific American*, 293.
<http://www.scientificamerican.com/article.cfm?id=kryders-law>