LOCKSS Boxes in the Cloud

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

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, 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 LOCKSS boxes.

1 Introduction

> "I really worry about everything going to the cloud," [Wozniak] said. "I think it's going to be horrendous. I think there are going to be a lot of horrible problems in the next five years. ... With the cloud, you don't own anything. You already signed it away. ... I want to feel that I own things, ... A lot of people feel, 'Oh, everything is really on my computer,' but I say the more we transfer everything onto the web, onto the cloud, the less we're going to have control over it." [12]

About 14 years ago the LOCKSS[1] Program at the Stanford University Libraries started building tools for libraries to collect and preserve content published on the Web. About 150 libraries currently run LOCKSS boxes preserving e-journals, e-books, databases, special collections and other content. These boxes are typically modest PC servers with substantial amounts of local disk.

Some of these libraries are members of the US National Digital Stewardship Alliance run by the Library of Congress. These and other libraries asked whether they could use "affordable cloud storage" for their LOCKSS boxes. The Library of Congress funded the LOCKSS team to investigate this question, and this is the final report of this work.

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 service's charging model to the remaining architectures rules others out for economic reasons. We then describe an experiment in which we implemented the most cost-effective architecture and ran a production LOCKSS box preserving 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.

2 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 reliability, and Reduced Reliability Storage (S3-RRS) is designed for 4 nines reliability.

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

[2] This is true for instances backed by S3, not by EBS. But EBS is not reliable enough; see Section 5.

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

2.1 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.

2.2 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 reliability 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.

3 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 preservation functions described in the OAIS Reference Model [9] including ingest, dissemination. integrity checking, damage repair and management as part of a peer-to-peer network [13].

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. The performance we observe disqualifies this architecture; extracting the content from S3 via its Web interface for integrity checking and dissemination is too slow.
- A LOCKSS daemon running in a compute service virtual machine instance with storage in the object storage service. The performance of this architecture led us to reject it; even from the compute service the I/O performance of the object storage service is much slower than local disk.
- 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. If the instance stopped for any reason the preserved content would be entirely lost, and 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 EBS and the cost of recovering from a total failure of EBS 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 EC2 with EBS storage and snapshots in S3 and ran it for several months, see Section 5.

We have examined some other architectures that require enhanced capabilities from the object storage service. They are discussed briefly in Section 7.2.

4 Economic Considerations

We now apply the Amazon charging model to the preferred 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.

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.

5 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 the following system parameters:

- AWS Access Key.
- AWS Secret Access Key.
- S3 bucket backing the image.
- Name of .tar.gz archive in S3 bucket containing LOCKSS configuration.
- A comma delineated list of volume ids to automatically attach/mount from EBS.

It mounts file systems from EBS containing 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 at most two.

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[3]. The experimental LOCKSS box was empty at the start, and was ingesting content during the experiment. During week 11 it finished ingesting, as reflected in the decrease in bandwidth charges, primarily for writing the new data to EBS and the snapshots to S3.

This experimental box was not representative in several ways:

[3]We eliminated week 1 because there were some start-up effects. The anomalous storage cost drop in weeks 8 and 12 of Tables 1, 2 and 3 was caused by a glitch in Amazon's accounting system that resulted in missing accounting records.

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.

- It preserved less content than the median LOCKSS box in the GLN and thus consumed less storage and less bandwidth than a production box would have.
- Also, out of caution the experimental box maintained two snapshots of its EBS volumes in S3; only one is necessary.
- Because 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 1 and 3 years. Since LOCKSS boxes are continually active, this would need to be a heavy-utilization reserved instance with a 3-year commitment. The current cost for a suitable instance would be $1200 plus $0.052/hr [2].
- Finally, 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 modeled both minimal over-provisioning (table 2), and over-provisioning matching that of the median LOCKSS box (table 3). The total 3-year cost in the minimal over-provisioning case would be about $8,770 and in the matching over-provisioning case about $18,130 We would expect actual costs between these two, and probably closer to the minimum.

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.

Service	Launch	Launch $/GB/mo	Now $/GB/mo	% Drop per yr
Amazon S3	03/06	0.15	0.125	3
Rackspace	05/08	0.15	0.15	0
Azure	11/09	0.15	0.14	3
Google	10/11	0.13	0.13	0

Table 4: Price history of storage services (base tier).

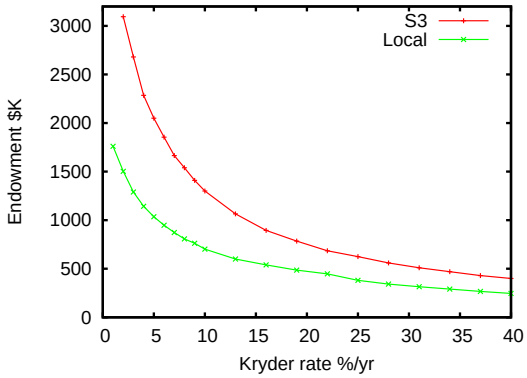

Figure 1: Projected 135TB 100yr endowment for S3 and local storage.

6 Projecting The Cost Of Cloud Storage

Table 4 shows the history of the prices charged by several major storage services. It shows that they drop at most 3%/yr. This is in stark contrast with the 30-year history of raw disk prices, which have dropped at least 30%/yr, as predicted by Kryder's Law [21].

This comparison is somewhat unfair to 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.

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 storage hardware at under $8K for 135TB [15]; correcting for the current 60% increase in disk prices since the floods [11] would make this $11.2K. Given S3's dominance of the cloud storage market, and thus purchasing volumes, it is very unlikely that their costs are higher than Backblaze's. Despite this, 135TB in S3-RRS costs more than $10K/mo. In the first month, an S3-RRS customer would pay almost as much as it would cost after the floods to buy the necessary hardware.

Work, discussed in Section 7.1, 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 135TB for 100 years at varying rates of annual price drop:

- In red using Amazon's S3 assuming storage charges only.
- In green maintaining three copies in local storage using Backblaze's costs for the hardware and assuming that the non-media costs are 2/3 of the total [14].

It is evident that, in the long term, the rate of price drop dominates all other parameters. Note that S3 is not competitive with local storage at any plausible Kryder rate. Unless the rate at which storage service price drops come 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 3% rate for S3 is more than five times that needed at the disk industry's roadmap projection of a 20% rate for local storage.

7 Future Work

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

7.1 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 [8]. 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 isn't [5].

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, U.C. 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 [16]; the long-term model was used to produce the graphs of Section 6.

7.2 Technology

A study of access patterns to data in digital archives [1] 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 in fact useful as an integrity check [17]. 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.

Week	Compute $	Storage $	Bandwidth $	Total $
2	8.74	47.01	0.71	56.46
3	8.74	47.82	0.73	57.29
4	8.74	48.63	0.94	58.31
5	8.74	49.45	1.00	59.18
6	8.74	50.26	1.18	60.17
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.

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 [17]. In this way any of a range of integrity check technologies [18, 19, 13] 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, 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 (Section 5), Table 5 shows an estimate of the cloud service costs that would have been incurred by this configuration of a median LOCKSS box [4]. Compare with table 2.

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 in reducing the cost of doing so. Thus, given

[4] These numbers were computed by assuming 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.

Week	Compute $	Storage $	Bandwidth $	Total $
1	7.64	3.70	0.52	11.86
2	8.74	3.76	0.71	13.21
3	8.74	3.83	0.73	13.29
4	8.74	3.89	0.94	13.56
5	8.74	3.96	1.00	13.69
6	8.74	4.02	1.18	13.93
7	8.74	4.09	1.52	14.35
8	8.74	4.15	1.32	14.21
9	8.74	4.22	1.53	14.48
10	8.74	4.28	3.19	16.21
11	8.74	4.35	2.92	16.00
12	8.74	4.41	3.57	16.71
13	8.74	4.48	2.71	15.92
14	8.74	4.54	3.31	16.59

Table 6: Costs that would have been incurred by median LOCKSS box with storage in Glacier.

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.

7.3 Amazon's Glacier

Just as we were finishing this paper, Amazon announced Glacier [3], a service specifically addressing the long-term storage market. By doing so, they vindicated the arguments above.

Glacier has some excellent features. In return for accepting a significant access latency (Amazon says typically 3.5 to 4.5 hours, but does not commit to a maximum latency), the service provides one low price of $0.01/GB/mo. 5% of the data can be accessed each month with no charge, which provides for local data integrity checks at 20-month intervals at no cost other than the compute costs in EC2 needed to perform them. Other than that, the charging model is complex [6], but each access above this costs at least as much as a month's storage. There is a $0.05 fee for each 1000 requests, making it important that archived objects are large.

Given the limited information about Glacier available, some significant assumptions were needed to generate Table 6, but the cost reduction it shows relative to S3 (Table 5) is plausible. Among these were that the LOCKSS box based on Glacier was able to operate under Glacier's limit of 5% of the data being accessed in a month, which means that content would on average be polled less frequently than once every $20 * (Q + 1)$ months, where Q is the poll quorum.

The analogous computation to those in Section 5 based on these numbers suggests that the total 3-year cost of ownership of a hypothetical LOCKSS box based on Glacier would be about $2,730. However, the assumptions above mean that this is a considerable under-estimate. For example, even though Glacier is taking aggressive measures to protect the data underneath the LOCKSS box, if the quorum were 5 content would

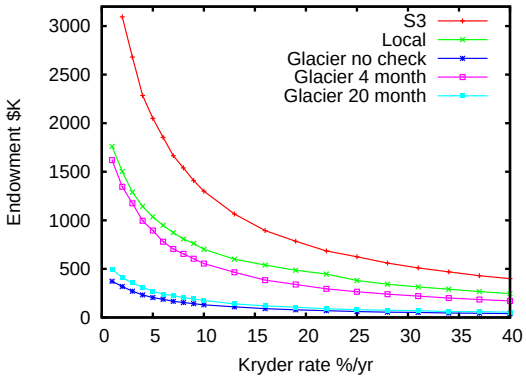

Figure 2: Projected 135TB 100yr endowment for S3, Glacier (integrity checked every 4 and 20 months and never) and local storage.

be polled on only once in 10 years, which is clearly inadequate. Further, implementing a LOCKSS daemon that used Glacier as a repository would require addressing two complex issues caused by the long delay and high cost of accessing content:

- Most of the methods by which a LOCKSS box can disseminate content, including proxying, serving, URL resolvers [7] and Memento [20] are optimized for direct access by readers to small, individual files. Both the long delay and the high per-request cost make Glacier unsuitable for this purpose whatever preservation technology is employed. LOCKSS boxes can export entire AUs as a WARC [10] archive. The user interface for doing so would have to be changed to cope with the delay, and to provide the user with some indication of the cost, but this would be a viable way to disseminate content.
- LOCKSS has a sophisticated mechanism for scheduling its operations but the current policy that drives the mechanism would not be suitable for a repository with a 3-5 hour access latency and sever cost penalties for spikes in request rates. An alternate policy would need to be developed. The current asynchronous nature of LOCKSS boxes in a network might need to be changed so that boxes operated synchronously, all on the same AU at the same time. Private LOCKSS networks, in which the quorum is close to the total number of peers, already have a *de facto* synchronous policy.

The complexities of Glacier's pricing for access to data [6] make it difficult to include integrity checking in a model. In order to compute Figure 2 (compare with Figure 1) we made the following assumptions:

- No accesses to the content other than for integrity checks.
- Accesses to the content for integrity checking are generated at a precisely uniform rate. This is important because Glacier's data access charges for each month are based on the *peak* hourly rate during that month.
- Each request is for 1GB of content. This is important because Glacier charges for each request in addition to the charge for the amount of data requested.

- In each month 5% of the content may be accessed without an access charge, but the requests to do so are charged the normal request fee.

Amazon pitches Glacier at the digital preservation market:

> Digital preservationists in organizations such as libraries, historical societies, non-profit organizations and governments are increasing their efforts to preserve valuable but aging digital content such as websites, software source code, video games, user-generated content and other digital artifacts that are no longer readily available. These archive volumes may start small, but can grow to petabytes over time. Amazon Glacier makes highly durable, cost-effective storage accessible for data volumes of any size. This contrasts with traditional data archiving solutions that require large scale and accurate capacity planning in order to be cost-effective.

They are right that Glacier's advantage in this market is that it avoids the need for capacity planning. The announcement does not, however, address the point illustrated in Section 7.1, which is that the long-term cost-competitiveness of cloud storage services such as Glacier depends not on their initial pricing, but on how closely their pricing tracks the Kryder's Law decrease in storage media costs.

Although Glacier with 4-monthly integrity checks is competitive with local storage at all Kryder rates, this assumes that they both experience the same Kryder rate. If Glacier experiences Amazon's historic 3% rate and local storage the industry's projection of 20%, Glacier is nearly 2.5 times more expensive.

It is anyone's guess how quickly Amazon will drop Glacier's prices as the underlying storage media costs drop. Based on the history of Amazon's pricing, we surmise that it will follow the pattern of S3's pricing. That is:

- The initial price will be set low, even at a loss, so that competitors are deterred and Amazon captures the vast bulk of the market.
- Subsequent media price drops will not be used to reduce prices all round, but will be used to create tiered prices, so that the part of the value of lower media prices not captured by Amazon itself goes to their largest customers.

If this surmise is correct, then the only way to make Glacier competitive with local storage is to extend the interval between integrity checks enough that all accesses to data are covered by the 5% monthly free allowance.

The shortest interval that can possibly achieve this is 20 months, although in practice some margin for error would be needed and thus a more practical interval would be 24 months. The 20 months line in Figure 2 suggests that this makes Glacier at a 3% Kryder rate somewhat cheaper than local storage at a 20% rate, but even if the assumptions above were to be true, this is not an apples-to-apples comparison.

The Blue Ribbon Task Force [4] and other investigations of the sustainability of digital preservation emphasize that preservation cannot be justified as an end in itself, only as a way to provide access to the preserved content. The local disk case provides practical access; the Glacier case does not. The long latency between requesting access and obtaining it, and the severe economic penalties for unpredictable or large volume accesses mean that Glacier cannot alone be a practical digital preservation system. At least one copy of the content must be in a system that is capable of:

- Providing low-latency access for users of the content. Otherwise the preservation of the content cannot be justified.
- Being a source for bulk transfer of the content to a Glacier competitor. Getting bulk data out of Glacier quickly is expensive, equivalent to between 5 months and a year of storage, which provides a powerful lock-in effect.

As an example of the costs of a practical system using Glacier but providing access and guarding against lock-in, if we maintain one copy of our 135TB example in Glacier with 20-month integrity checks experiencing a 3% Kryder rate, and one copy in local storage experiencing a 20% Kryder rate (instead of the three in our earlier local storage examples), the endowment needed would be $517K. The endowment needed for three copies in local storage at a 20% Kryder rate would be $486K. Given the preliminary state of our economic model, this is not a significant difference. Replacing two copies in local storage with one copy in Glacier would not significantly reduce costs, instead it might increase them slightly. Its effect on robustness would be mixed, with 4 versus 3 total copies (effectively triplicated in Glacier, plus local storage) and greater system diversity, but at the cost of less frequent integrity checks.

Customers, expecially those with a short planning horizon, are likely to look only at the short-term costs and choose a Glacier-only solution that will lock them in, be no cheaper in the long run, and will fail to provide adequate access to the content to justify its preservation. But it will be difficult for other vendors to compete with Amazon in this space; they are trading on the short-termism identified by Haldane and Davies [8].

8 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 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 [17] proposal for a simple extension to the current S3 API that would greatly improve the suitability of S3 and similar services, such as private cloud implementations, for digital preservation.

Amazon's introduction of Glacier, a cloud-based archival storage service, supports our conclusion that existing cloud storage services are not competitive for long-term preservation. If customers can tolerate substantial delays and costs for user access to data, and if either Glacier's prices drop in line with

media costs, or customers are satisfied with infrequent integrity checks, it will be very competitive.

Acknowledgments

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 U.C. Santa Cruz.

References

[1] Ian F. Adams, Ethan L. Miller, and Mark W. Storer. Analysis of workload behavior in scientific and historical long-term data repositories. Technical Report UCSC-SSRC-11-01, University of California, Santa Cruz, March 2011.

[2] Amazon. Amazon EC2 Reserved Instances. `https://aws.amazon.com/ec2/reserved-instances/#3`, 2012.

[3] Amazon. Amazon Glacier. `http://aws.amazon.com/glacier/`, August 2012.

[4] Blue Ribbon Task Force on Sustainable Digital Preservation and Access. Sustainable Economics for a Digital Planet. `http://brtf.sdsc.edu/biblio/BRTF_Final_Report.pdf`, April 2010.

[5] J. Doyne Farmer and John Geanakoplos. Hyperbolic Discounting is Rational: Valuing the Far Future With Uncertain Discount Rates. Technical Report 1719, Cowles Foundation, Yale University, Aug 2009.

[6] Klint Finley. Is There a Landmine Hidden in Amazon's Glacier? *Wired*, August 2012.

[7] Philip Gust. Accessing LOCKSS Content Through OPACS and Link Resolvers. Technical report, LOCKSS Program, Stanford University Libraries, November 2010.

[8] Andrew G Haldane and Richard Davies. The Short Long. In *29th Societe Universitaire Europeene de Recherches Financieres Colloquium: New Paradigms in Money and Finance?*, May 2011.

[9] ISO. Reference model for an open archival information system: Iso iso 14721:2003. `http://www.iso.org/iso/catalogue_detail.htm?csnumber=24683`, 2003.

[10] ISO. ISO 28500:2009 Information and documentation - WARC file format. `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=44717`, May 2009. Draft at `http://bibnum.bnf.fr/WARC/warc_ISO_DIS_28500.pdf`.

[11] Paul Kunert. Hard disk drive prices quick to rise, slow to fall. *The Register*, May 2012.

[12] Robert MacPherson. Apple co-founder Wozniak sees trouble in the cloud. `http://phys.org/news/2012-08-apple-co-founder-wozniak-cloud.html`, August 2012.

[13] Petros Maniatis, Mema Roussopoulos, T. J. Giuli, David S. H. Rosenthal, and Mary Baker. The LOCKSS peer-to-peer digital preservation system. *ACM Trans. Comput. Syst.*, 23(1):2–50, 2005.

[14] Richard L. Moore, Jim D'Aoust, Robert H. McDonald, and David Minor. Disk and Tape Storage Cost Models. In *Archiving 2007*, May 2007.

[15] Tim Nufire. Petabytes on a Budget v2.0: Revealing More Secrets. `http://blog.backblaze.com/2011/07/20/petabytes-on-a-budget-v2-0revealing-more-secrets`, July 2011.

[16] Daniel C. Rosenthal, David S. H. Rosenthal, Ethan L. Miller, Ian F. Adams, Mark W. Storer, and Erez Zadok. Toward an Economic Model of Long-Term Storage. In *FAST2012 Work-In-Progress*, February 2012.

[17] David S. H. Rosenthal. LOCKSS: Lots Of Copies Keep Stuff Safe. In *NIST Digital Preservation Interoperability Framework Workshop*, March 2010.

[18] Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, and Ram Swaminathan. Auditing to Keep Online Storage Services Honest. In *HOTOS XI, 11th Workshop on Hot Topics in Operating Systems*, May 2007.

[19] Sangchul Song and Joseph JaJa. Techniques to audit and certify the long-term integrity of digital archives. *Int. J. Digit. Libr.*, 10(2-3), August 2009. DOI: 10.1007/s00799-009-0056-2.

[20] Herbert van de Sompel, Michael L. Nelson, Robert Sanderson, Lyudmila L. Balakireva, Scott Ainsworth, and Harihar Shankar. Memento: Time Travel for the Web. `http://arxiv.org/abs/0911.1112`, November 2009.

[21] Chip Walter. Kryder's Law. *Scientific American*, 293, August 2005.