

Amazon S3 for Science Grids: a Viable Solution?

Mayur Palankar[#], Ayodele Onibokun[#], Adriana Iamnitchi[#], Matei Ripeanu^{*}

[#]*Computer Science and Engineering, University of South Florida*

4202 E. Fowler Ave., Tampa, FL 33620, USA

{mpalanka, aoniboku, anda}@cse.usf.edu

^{*}*Electrical and Computer Engineering, University of British Columbia*

2356 Main Mall Vancouver, BC V6T 1Z4, Canada

matei@ece.ubc.ca

Abstract– With the maturing of grid infrastructures, the interest in outsourcing the basic grid services is increasing. In science grids, data storage is one such fundamental service: indeed, in many virtual organizations terabytes of new data are produced each day and much more is processed.

Recently, Amazon.com has introduced a novel storage utility, the Simple Storage Service (S3). S3 aims to provide data storage as a low-cost, highly available service, with a simple ‘pay-as-you-go’ billing model. This article evaluates S3 as a black box and reasons whether S3 is an appropriate service for science grids. In the process, it identifies requirements for a storage service for this particular type of community, including security features and a more diversified pool of storage resources with different capabilities and a more flexible pricing scheme.

I. INTRODUCTION

The volume of data produced and shared by data-intensive scientific collaborations is rapidly growing. For example, modern high-energy physics experiments, such as DZero [1], typically generate more than one TeraByte (TB) of data per day and may move up to ten times as much [2]. As a result, significant resources, both human and material, are allocated to support the data-intensive operations of these communities, leading to high storage and management costs.

Recently, Amazon.com introduced the Simple Storage Service (S3) [3], a novel storage utility. S3 aims to provide data storage as a low-cost, highly available service, with a simple ‘pay-as-you-go’ billing model. In addition, Amazon uses open protocols and offers development tools to integrate S3 with diverse applications from remote backup of personal data to e-commerce applications that may use S3 to support their data-storage tier.

These characteristics make S3 a good candidate to offload storage support for data-intensive scientific collaborations. The goal of this paper is to evaluate

whether offloading storage from in-house maintained mass storage systems to S3 is a feasible and cost-effective alternative for today’s scientific collaborations like DZero [1], LHC [4], or SLAC [5]. To this end, we characterize S3’s observed availability and data access performance using a collection of our own nodes and geographically-distributed PlanetLab nodes [6]. We use this characterization in conjunction with more than two years of real traces from a scientific community, the DZero Experiment, a high-energy physics collaboration that spans 18 countries and has more than 500 active users. We evaluate the feasibility, performance, and costs of a hypothetical S3-supported DZero collaboration.

The contributions of this paper are:

- A characterization of S3 in terms of availability and user-observed data transfer performance.
- An evaluation of the costs of outsourcing DZero data requirements to S3 and an analysis of the potential ways to reduce these costs.
- A discussion of S3 security features in the context of data-intensive collaborative applications and suggestions for improved functionality for future storage utilities that might be dedicated to support science.
- A discussion of the support that a storage service such as S3 may want to provide in order to satisfy the data requirements of a potential customer base from the scientific community.

The rest of this paper is organized as follows: Section II gives an overview of S3’s core concepts, architecture, and functionality. Section III presents data usage characteristics in science grids and provides further insight on how science grids can be integrated with S3. Section IV presents a measurement-based study of the performance claims of S3. Section V estimates the S3 costs and discusses various models for using S3 in DZero-like environments. Section VI lists out future discussion

topics and suggestions for improving S3. Section VII summarizes our study and contributions and outlines future research directions.

II. AMAZON S3

S3 is supported by tens of thousands of computer systems around the world [7] to provide a scalable data storage infrastructure that aims to offer low data-access latency, infinite data durability, and 99.99% availability [3]. Since its launch in March 2006, S3 has acquired a large user base ranging from home users to small and large business enterprises [8]. It has been reported that S3 stores over 800 million data objects [9].

A. Concepts and Architecture

Data stored in S3 is organized over a two-level namespace. At the top of the namespace level are the *buckets*—similar to folders or containers. Buckets have a unique name in the S3 global namespace and serve several purposes: they allow users to organize their data; they identify the user to be charged for storage and data transfers; they play an important role in access control; and finally, they serve as the unit of aggregation for audit reports [9].

Buckets can store an unlimited number of *data objects*. Objects are composed from two parts: an opaque blob (of up to 5GB in size) and metadata, which includes user-specified key/value pairs for each object (up to 2KB) and a small number of predefined HTTP metadata entries (e.g., Last-Modified).

While users can create and access objects in buckets, subject to access control restrictions described in the next section, to modify or rename an existing object the user has to download the entire object, modify it, and finally re-transfer it back to S3, possibly with a new name.

Search is limited to a single bucket and is based on the object name only (either using the full object name or using prefixes). Metadata or content-based search capabilities are not provided.

Charging for the S3 service is for both storage (currently at a rate of \$0.15/GB/month) and access to data (at \$0.20/GB of data transferred¹). Regardless of the owner of an object or the identity of the user accessing the object, all charges are directed to the owner of the bucket that stores the object generating the charges.

¹ Since the writing of this report, S3 transfer costs decreased to between \$0.13/GB and \$0.18/GB, depending on volume transferred.

B. Security Model

Identities: When users register with Amazon's Web Services, they are assigned an *identity*, the 'AWS Access Key ID'. To prevent *whitewashing* attacks (i.e., attacks based on creating a large number of identities by the same physical user), identities are linked to user credit cards. Additionally, S3 includes the concept of anonymous requests, when no user identity information is associated with the request.

Authentication: Clients authenticate using a public/private key scheme and keyed-hash message authentication code (HMAC [10]). Together with their identity, users are assigned a private key: the 'AWS Secret Access Key' generated by Amazon during registration. Both keys are permanently stored at Amazon and can be recovered using web access mechanisms provided by Amazon's Web Services website. Thus, even though public/private key cryptography is used, the properties of the solution are those of a shared secret scheme as it will become evident later in this paper.

Access control: Access control is specified using access control lists (ACL) at the granularity of buckets or objects. Each ACL can specify the access attributes for up to 100 identities. The access control attributes supported are:

- **FULLCONTROL**: the owner preserves full control over the object, having all permissions below;
- **READ**: for buckets or objects, the user can read the object;
- **WRITE**: for buckets only, the user can create objects in the bucket;
- **READACL**: for buckets or objects, the user can read the ACL and the identity of the owner. Implicit for the owner;
- **WRITEACL**: the user can change the ACL – this is similar to full user control since the user can assign any right. Note that ACL's can not be modified: they can only be overwritten by a new version. Therefore, to modify an ACL, the user needs to read/download the ACL, modify it locally as desired, and then write/upload the updated ACL back to S3.

Auditing: Auditing can be implemented by configuring buckets to create and store access log records. These records contain details such as the request type, the object the request accessed, and the time and date when the request was processed.

C. Data Access Protocols

Currently, S3 supports three data access protocols:

- *Simple Object Access Protocol (SOAP)* is an open messaging framework based on a simple XML-

based communication protocol that allow applications to exchange information over a number of transport protocols such as HTTP and SMTP [11]. SOAP is supported by and standardized by the World Wide Web Consortium (W3C).

- *Representational State Transfer (REST)* protocol attempts to reduce the complexity of SOAP by using a limited set of HTTP commands to access and manipulate the server-side state or data records [12]. The actual message may be XML encoded and is contained within the HTTP request. Standard techniques (i.e., HTTPS) are used to provide server authentication, data confidentiality and a tamper-proof data channel between clients and the server.
- *BitTorrent* [13] is a file sharing protocol designed to work efficiently under flash crowd conditions (i.e., when large numbers of clients simultaneously attempt to download the same file). S3 makes publicly accessible data available through BitTorrent by providing tracker functionality. Essentially, a tracker assists multiple clients interested in the same data item in locating chunks of the data; a key functionality for enabling cooperative downloads.

III. CHARACTERISTICS OF SCIENCE GRIDS

Data produced, stored and used in science grids have particular characteristics in terms of scale and usage patterns. This section surveys the usage characteristics of data-intensive scientific collaborations and their implied requirements on the storage infrastructure. To quantify this characterization we focus the discussion on DZero [1], a representative high-energy physics collaboration whose main data generator is the particle accelerator at Fermi National Accelerator Laboratory.

A. Data Usage Characteristics

Particular to scientific communities is the intense usage of data: jobs submitted by hundreds of users process massive collections (TeraBytes), organized in hundreds to thousands of GigaByte-sized files. Moreover, new data is continuously produced. For example, modern high-energy physics experiments such as DZero [1] typically acquire more than one TB of data per day and move up to ten times as much. At the same time, access to data is shared by hundreds of users: for example, the 561 world-wide located DZero scientists submitted 113,062 jobs that processed more than 5.2 PetaBytes of data between January 2003 and March 2005. These numbers translate into accessing more than 1.1 million distinct

data files [2] generating requests for a sustained access rate of about 70MBps.

Finally, another particular data usage characteristic is *co-usage*: in scientific environments groups of files are often used together. Taking the high-energy physics project DZero as a case study again, each data analysis job accessed on average 102 files, with a maximum of more than 20,000 files. The need for simultaneous access to multiple files stresses the problems brought up by the large file size, requesting transfers of data collections in the order of TB. For example, the largest 10 datasets in the DZero traces analyzed in [2] are between 11 and 62 TB.

Workload analysis on the DZero traces shows that a significant part of the data is used for only limited time periods. This data can be archived on slower durable storage without an impact on performance...If we consider the lifetime of a file as the interval between the first and the last access to it as recorded in the 27 months of DZero traces, then about 30% of the files do not live longer than 24 hours, 40% not more than one week, 50% have a lifetime shorter than one month, while about 35% were still in use more than five months after the first recorded access. Consequently, out of the 4.54TB of data accessed each day, 30% will not be needed after 24 hours.

In addition, data is highly cacheable: experimental evaluations [14] with various caching techniques performed on the DZero traces show that caching data in local storage repositories with (now common) capacities of TBs may reduce data transferred to under 2.5% of the storage size.

Another characteristic of scientific data is its derivability from other data, allowing thus for various priorities in data preservation: processed data is derived from *raw* data typically generated by some unique or rare events (such as the recording of an astronomical phenomenon or particle collisions in a particle accelerator). This particularity allows thus for trading storage and transfer for computation costs: it sometimes may be more efficient to regenerate derived data than to store it or transfer it between remote locations.

B. Storage Service Requirements for Data Intensive Scientific Applications

This section presents the requirements for a storage infrastructure targeting data-intensive scientific communities:

- 1) *Data durability*. Depending on the specifics of each science project, losing experimental ('raw') data may be costly (since repeating a physics experiment may require reconfiguring and operating

expensive instruments) or even unacceptable. This results in strong durability requirements for raw data. Derived data can generally be reconstructed from raw data at the cost of additional computation.

2) *Data availability*: Data availability quantifies the successful access to previously stored data. Although most of the data is used for batch computations that do not require high availability in themselves, the fact that these operations require co-allocation of expensive resources (e.g., large compute resources, visualization equipment) increases the premium put on availability. Note that durability does not imply availability – data can be still stored but not available at the time of the request. However, availability requires durability. Finally, service availability for uploads rather than retrieval is important since data can be temporarily stored at experimental facilities only for limited periods of time.

3) *Access performance*. While data archival is an important use case, we expect that the predominant use case in our context is live remote storage. Thus, fast data access is key to support science applications.

4) *Usability*: Although ease of use can be quantified across multiple directions, the main characteristic of interest in the context of this paper is a set of protocols and APIs that allow composability with higher-level services for easy integration with science applications.

5) *Support for security and privacy*: Science applications are often collaborative with complex data sharing arrangements between multiple parties: users or institutions. The security infrastructure should enable defining and enforcing these complex sharing arrangements.

6) *Low cost*: Ultimately, cost is the main driver for adopting a storage utility to replace in-house storage systems. Utilities have the potential to benefit from economies of scale and we believe data storage is ripe for such development. However, lack of standardization and the particular requirements of data intensive science might delay or make inopportune the adoption of storage utilities.

IV. AMAZON S3 EVALUATION

Our evaluation of S3 is driven by the requirements of data-intensive scientific applications outlined in the previous section. In this section, we describe our experimental setup and present our quantitative evaluation of S3 service.

A. Experiment Setup

We conduct three different sets of experiments in order to evaluate the services provided by S3. For the data availability experiment, we use the Java APIs (jSh3ll [15]) while for the data access performance measurements, we use the Python APIs. Both APIs access the REST based protocols. Our choice for REST is motivated by its popularity: 85% of current S3 usage is based on the REST protocol [16].

We use five nodes for our experiments: four PlanetLab nodes and one dedicated machine at the University of South Florida in Tampa, Florida. We chose the PlanetLab nodes at locations that geographically approximate DZero user location: hence, two nodes were located in Europe (one in Germany and one in France) and two were located in the US (one in New York and one in California).

B. Data Durability

While characterizing data durability is not the main goal of our study, we mention that, during the (admittedly short) six-month time span of running S3 experiments, we have not observed any permanent data loss. An experimental study of different magnitude is required to characterize the data durability offered by the S3 service.

C. Data Availability

To test S3 data availability we download a small (1KB) object at regular, pre-defined time interval (15 minutes) from a dedicated machine at USF. If the download fails, we attempt to identify the cause of failure and exclude failures due to faults of the access link to our measurement node by issuing a HTTP get request for an object from USF website and a non-cacheable object from Amazon.com website. The first request checks the LAN connection, while the second request checks our campus connection. If either of these two requests fails, the request is not retried and the sample is eliminated from our data collection that reports S3 availability. Otherwise, failed requests are retried after an exponential back-off with at most four retries and an initial wait time of 2 minutes.

We ran this experiment for four weeks for a total of 2,688 access requests. We have observed an availability rate of 99.03% after the original download attempt, 99.55% after the first retry and a full 100% availability after two retries. Additional retries were never needed during the observed period. These results are convergent with Amazon’s stated 99.99% availability target [3].

We continue to run these experiment and an updated set of results will be available with a new version of this paper.

D. Data Access Performance

Our objective in this section is to evaluate the access performance for S3-stored data from a client perspective. We compare the download time of four different object sizes of 100KB, 1MB, 10MB, and 100MB. The data set includes 28 experiments (4 times a day over 7 days) to remove variability due to internet traffic. Our experiments confirm that the location of the client and the time of the day impact on the observed data access performance. Figure 1 presents the average access time and its variability for each client for the entire experiment duration.

The two main components that influence data access performance are connection establishment and download bandwidth.

1) *Connection Establishment time* depends on network latency and on authentication and SOAP/REST overheads. This initial start-up time is recorded for each of the data access experiments at each node and is small (1 – 30ms, depending on location) compared to the total download time.

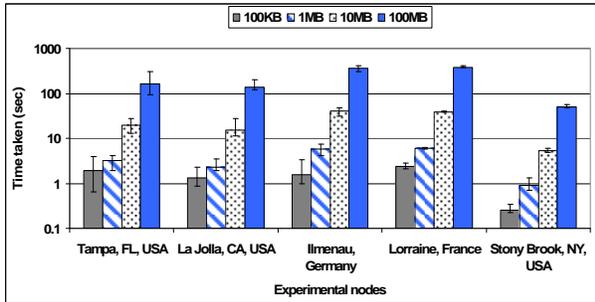


Figure 1: Average, minimum and maximum download time for different file sizes and locations. (Note the logarithmic scale on Y axis.)

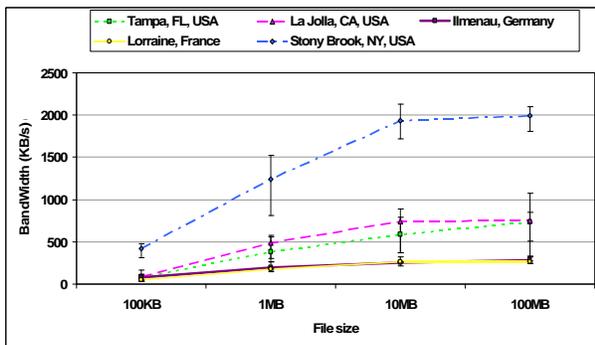


Figure 2: Average, minimum and maximum observed download bandwidth for different client locations and file sizes

2) *Access Bandwidth.* As expected, the observed data access bandwidth depends on client location and characteristics and the size of data downloaded. (As a result of the TCP slow start, the obtained download bandwidth for small objects is lower than that for large objects). Figure 2 shows the average, minimum

and maximum download bandwidth at each node as a function of the size of file downloaded.

We repeated all experiments to estimate the time to upload data to S3. We find that the upload time is largely similar to the download time for each location. Finally, concurrent requests to the same data objects do not have a noticeable impact on performance as our limited numbers of clients cannot stress-test S3.

E. Downloading Files via BitTorrent

Typically, multiple replicas of the same data item are available simultaneously at various sites participating in a virtual organization. BitTorrent enables partial parallel downloads from each of these replicas to serve an additional request for the data item, thus allowing for faster downloads. For the science community, the availability of data access through BitTorrent protocols is relevant as it enables simple integration of cooperative caching mechanisms to improve access performance. Additionally, a cooperative cache supported by BitTorrent is a direct solution to reduce cost (more precisely, S3 transfer charges) while preserving the data durability and availability offered by S3.

The main goal of our experiment is, first, to compare the BitTorrent-enabled data access performance with that offered by regular S3 transfers and, second, to understand the load balance between S3 and other data sources when multiple sources ('seeds' in BitTorrent parlance) are available.

To quantify the effectiveness of using BitTorrent, we download data from a single location (USF, Tampa, FL) while varying the number of seeds hosted on PlanetLab nodes and always maintaining a seed at S3. We measure the download time using BitTorrent and the amount of data downloaded from S3.

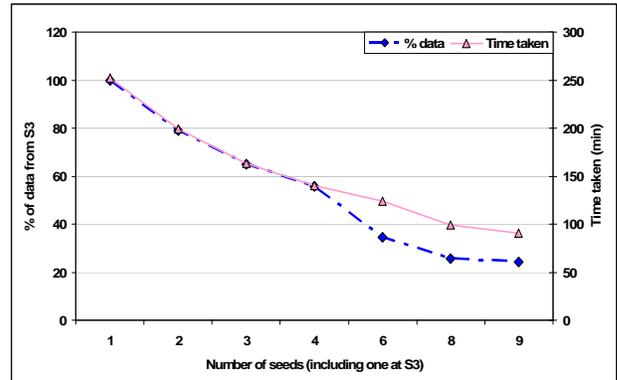


Figure 3: Time taken and percentage of data downloaded from S3 as a function of the number of seeds present in the system.

Figure 3 demonstrates that S3 contributes a large percentage of the load. To check for any unfair play

by S3, experiments were repeated on dedicated machines at USF instead of PlanetLab. In addition to better understanding the PlanetLab testbed (i.e., PlanetLab nodes impose an upload rates cap of 20KBps for BitTorrent), we infer an important lesson about S3: S3 limits its upload rate to any one BitTorrent client to 72KBps. Figure 4 shows the load balance when seeds with different bandwidth were used. (Note only one node apart from S3 is used in this experiment.)

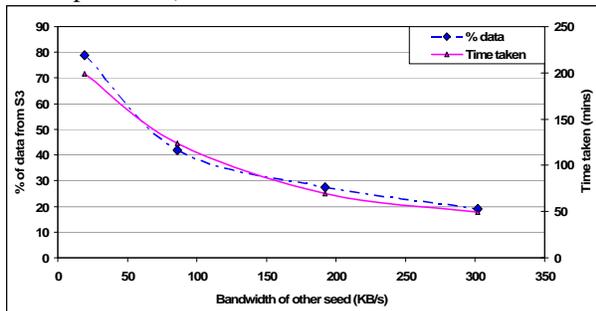


Figure 4: Effect of bandwidth on load balance and performance

To continue with our black-box exploration, we tried to freeride S3: we attempt to use the S3-provided BitTorrent tracker without generating any data transfer costs. It turns out that this freeriding behavior can be obtained by revoking the read permits from the S3-stored object (i.e., mark it as private) which results into faults when nodes attempt partial file downloads from S3 and thus makes it possible to download only from other seeds. Our experiments show (Figure 5) that S3 takes no countermeasure against this type of free-riding: by circumventing the data transfer costs from the S3 seed, the only performance cost is due to a smaller degree of parallelism when downloading data from fewer sources.

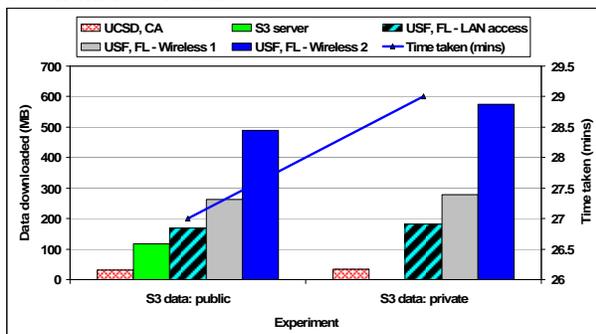


Figure 5: Change in performance while free-riding S3 tracker

To summarize, our S3 experimental results show good availability, variable download time depending on the node location and file size, and a fair use of the BitTorrent protocol.

V. AMAZON S3 FOR SCIENCE GRIDS

This section answers three intertwined questions: estimates the cost of using S3, and evaluates whether

S3 performance and security functionality are adequate to support scientific collaborations.

While answering these questions in general is a generous subject, our approach is based on a case study: we use DZero generated load and study different deployment scenarios that combine S3 and data caching. While our results can not be directly generalized, they provide a first case study and suggest an approach that can be easily applied to different scenarios. Moreover, our conclusions regarding the security infrastructure and the guidelines to design future storage services are largely application independent.

A. Cost

While the cost of storage systems and their management has continued to decrease in recent years, it still represents a major component, often the single most expensive item when running an IT infrastructure. At the same time, a data service provider such as S3 can reduce costs because of their ability to exploit economies of scale along multiple axes, both human and material.

In this section, we consider the purely hypothetical case of DZero’s using S3 for its data needs. Two main types of costs are thus to be considered: data access costs and data storage costs. Assuming all DZero data is entirely stored on and accessed from Amazon S3, the annual costs are \$675,000 per year for storage and \$462,222 per year for transfer, which adds up to \$1.13 million per year or \$94,768 per month.

However, each of these costs can be reduced in various ways. First, storage costs can be reduced by archiving “cold” data on low-cost storage and maintaining only the data most likely to be used on high-availability, low-latency storage. However, this approach needs more support from the S3 service than currently in place. Another way to reduce storage costs is to only store raw data and derive the rest of the data from raw data. The workloads we have do not contain sufficient information to allow us estimate the potential benefits of this approach.

Second, transfer costs can be reduced by using local caches. (In this discussion we will focus on the S3 costs, leaving out the costs of purchasing and maintaining the local caches). As presented in [14], various cache replacement algorithms that combine data request reordering with data pre-fetching lead to significant savings in data transfers for DZero. As an example, for a cache of 50TB, the amount of data that needs to be transferred on average per job is 0.013% of the cache size or 6.6GB. Given that over the 27 months, 113,062 jobs were submitted within the

subset of traces, the cost of transferring data from S3 to the cache is \$1.32 per job or \$149,241 for all the jobs in our traces. This leads to \$66,329 per year in transfer costs, which is an order of magnitude less than the scenario without caching.

Additionally, S3 data transfer costs can be reduced by using computation close to where data is stored. Since data transfers between S3 and Amazon's Elastic Computing Service (EC2) are not charged, EC2 hosted computations can be used to replace the data transfer costs for derived data with the cost to recompute this data on EC2 (\$0.1/hour of computation). Over the period of our workloads, the 973,892 hours of computation on data requested would cost \$3,607 per month using EC2.

Finally, another approach for reducing transfer costs is to use BitTorrent to download in parallel from multiple replicas. Transfer costs will thus be proportional with the amount of data downloaded from the S3 seed.

Although the ideas above are acknowledged methods in data-intensive computing, they require support from the application side or from S3, as discussed in Section VI.

B. Performance

A second question we are set to answer is whether the potential savings obtained by outsourcing data storage come at the price of performance degradation.

From our observed data-access performance, the download time varies with the location of the downloading node (Figure 1). We suspect that S3 optimizes the location of data according to the location of the user creating it. This assumption, if true, explains the high variability that we found in the download time that goes beyond the location of the downloading node or the time of day. It would also imply the need to change the S3 architecture to reduce the performance variability that users in scientific collaborations spread across the world might experience.

The implementation scenarios discussed in the previous section, such as caching or even proactive data replication, can be used to significantly improve performance. BitTorrent can also be used successfully, since typically multiple replicas of data items co-exist in science collaborations.

We also note that, for batch processing with little or no interactive user input in modifying the batch flow, the slower access to S3 stored data will not have a significant impact on user observed performance as long as jobs are specified in advance and S3 is able to provide data at overall average rate faster than DZero

compute resources consume it. An efficient system using S3-hosted data would only need limited local caching and a well-designed job management system that makes use of batch-job information to proactively download data while jobs are still in compute queues. Lastly, if EC2 is used for DZero computations, the data access time will be dependent on the transfer time between the two Amazon-provided services, S3 and EC2.

C. Security functionality evaluation

Risks: The assessment of the security functionality should start with an evaluation of the risks involved.

- Traditional risks with an outsourced data storage system: permanent data loss, temporary data unavailability (DoS), loss of data confidentiality, and malicious data modifications are still a concern. Some of these risks are prevented by the security scheme S3 uses (e.g., loss of confidentiality during transport is prevented by the TLS protocols used) while others can be mitigated by user-level solution not imposed by S3 (e.g., applications can cryptographically sign the data stored in S3 or use a hash-based scheme similar to that used in the self-certifying path names [18] to detect malicious or accidental data changes).
- S3 charging scheme introduces an additional risk: *direct monetary loss*. This risk is magnified by the fact that S3 does not provide a solution to limit the amount users stand to lose in case of an attack. For example, an attacker can attempt an attack similar to a denial of service (DoS) attack, but, instead of aiming to bring the down S3 service, he/she would simply aim to download/upload data from from/to a bucket owned by the target user to drive his S3 usage costs to the sky. Note that in the case of DoS the monetary loss is generally due to missed business opportunities or the cost of recovering data/configurations while now the monetary loss is direct.

The security model offered by S3 has the important merit of being simple. Simplicity however, comes at the price of limited support for large, collaborative applications that aim to control access to data resources offered by multiple participants. We summarize these limitations below:

- *Crude access control scheme:* The access control solution based on access control lists does not scale well to manage large systems with thousands of users and million of entities. Additionally S3 supports only a limited number of access rights (e.g., there is no write control at the individual object level but only at the bucket

level) and the access control lists are limited in size (to 100 principals)

- *Lack of support for fine-grained delegation:* Even moderately complex data-intensive scientific collaborations today make use of delegation for efficient data management and processing. For example, users delegate access to specific datasets to programs that operate on their behalf on remote computers. Higher risks (related to costs) make proper access control and delegation models even more necessary for S3 supported collaborations.
- *Implicit trust and lack of support for non-reputability:* Users have to trust S3 entirely as it does not provide unforgeable ‘receipts’ for transactions, i.e., signed certificates a user could present to a third party to demonstrate that she have stored a specific data item on S3.

Additionally, invocations recorded by S3 and presented in the audit trail are not signed by users. This makes the audit trail repudiable. Worse, with shared keys, or with a solution where the secret key is generated at Amazon, it is impossible to provide non-reputability.

- *Unlimited risk:* S3 does not offer any mechanism to support user-specified usage limits, e.g., quotas (per bucket owner or per delegated principal). As a result, the potential damage that an attacker (or simply a buggy program) can produce if it is able to read/write data that is billed to the user is limited only by the ability of the attacker to access S3 data and by the limit on the user’s credit card.

VI. DISCUSSION

While S3 could be used to support data intensive experiments like DZero, our investigation reveals an interesting characteristic: while S3 bundles at a single pricing point three highly desirable data characteristics (i.e., high durability, high availability, and fast access), many applications do not need all these three characteristics bundled together. One clear example is archival storage which puts a premium on durability but can survive with lower availability and access performance. In the case study we have considered, DZero, the large share of data that is infrequently used could be well stored on tapes. Similarly, application might be interested in S3-provided data caches, where availability and fast access are paramount while durability is less important. These observations makes us suggest that S3 should provide service through a number of limited classes of service that would allow users to choose their desired durability/availability/access performance mix to better control costs. We extend this idea in a separate position paper [17].

A second observation is that in addition to caching, application-level knowledge should be used to contain costs. Examples suggested previously limit long-term durable storage only to *raw* experimental data or transferring ‘cold’ data to cheaper media.

It is unlikely that these recommendations will be followed by S3 soon, but this experience allows us to recommend primitives for extended flexibility offered by novel storage infrastructures that could target the demands of data-intensive science applications. Our recommendations, listed below, could also prove valuable to large Grid deployments like TerraGrid or WestGrid are moving towards offering infrastructure services for science, similar in goals to those offered by Amazon’s S3 and EC2.

- *Enhanced security functionality to support complex collaborations:* Our analysis reveals the simple security model offered by S3 has does not support complex collaborations and exposes users to major risk. We believe that two key components are required: ability to limit potential damage in the case of an attack and support for fine-grained delegation.
- *Additional functionality for better usability:* A number of additional service functionalities would significantly simplify integration with applications; among them are metadata based searches; renaming objects; mutating access control lists.
- *Relaxed limitations:* Allowing larger groups of users (e.g., ACL can identify of at most 100 users).

VII. CONCLUSION

We would like to emphasize that S3 was not designed for the science community. Indeed, the science community has very specific requirements and extreme challenges regarding data usage. As such, our “criticism” of the S3 system should be read in fact as recommendations to any storage provider who would like to target the science community.

The contributions of this paper are in evaluating S3 as a black box and in formulating recommendations for better serving the science community with high data demands while maintaining the costs low. Costs can be reduced by using BitTorrent along with S3, exploiting data usage and application characteristics to improve performance, and, more importantly, by introducing user-managed collaborative caching in the system. In effect, our recommendations are driven by S3 billing structure: *we recommend using S3 for the costly tasks of providing high data availability and durability* (where costs are driven up by specialized hardware and nontrivial engineering effort) *and employ caching at the edges of the system*

to reduce the access volume when the usage patterns allow. These recommendations may not only reduce the S3 bill but will also significantly improve performance due to a cacheable workload specific to these collaborations.

Finally, we identify requirements that are not currently satisfied by S3. While S3 successfully supports relatively simple scenarios (e.g., personal data backup) and it can be easily integrated in the storage tier of a multi-tiered Web application (e.g., in Microsoft Software Development Network to serve its Direct Student downloads), its existing security functionality is strikingly inadequate to support complex, collaborative environments like the ones in today's scientific collaborations. More precisely, S3 lacks in terms of access control, support for delegation and auditing, and makes implicit trust assumptions between S3 and its clients. This lack of functionality is even more troubling when direct financial loss is at stake.

Acknowledgements

We acknowledge the financial support from the Department of Computer Science and Engineering at University of South Florida for the Amazon S3 service expenses.

VIII. REFERENCES

- [1] The DZero Experiment. <http://www-d0.fnal.gov>
- [2] A. Iamnitchi, S. Doraimani, G. Garzoglio., "Filecules in High-Energy Physics: Characteristics and Impact on Resource Management," 15th IEEE International Symposium on High Performance Distributed Computing (HPDC), June 2006.
- [3] Amazon Web Services. <http://s3.amazonaws.com>
- [4] The Large Hadron Collider. <http://lhc.web.cern.ch/LCG>
- [5] The Stanford Linear Collider. <http://www2.slac.stanford.edu/vvc/experiments/slc.html>
- [6] PlanetLab Consortium. <http://planet-lab.org>
- [7] S. Garfinkel, "Commodity Grid Computing with Amazon's S3 and EC2," Login, USENIX, February 2007.
- [8] M. Kirkpatrick. "Amazon releases early info on S3 storage use," <http://www.techcrunch.com/tag/s3>, July 2006.
- [9] Amazon S3: Developer Guide, March 2006. <http://developer.amazonwebservices.com/connect/servlet/KbServlet/download/123-102-1008/s3-dg-20060301.pdf>
- [10] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", IETF - Network Working Group, February 1997.
- [11] D. HinchCliffe. "REST vs. SOAP: The battle of the web service titans," website: http://webservices-system.com/read/79282_p.htm
- [12] Bohnsack et al., "Exploring RPC with SOAP and REST in a Distributed ISBN Lookup Application," May 2005.
- [13] BitTorrent. <http://www.bittorrent.com>
- [14] S. Doraimani., "Filecules: A New Granularity for Resource Management in Grids," University of South Florida Technical Report, 2007.
- [15] Java Shell. <https://jsh3ll.dev.java.net/>
- [16] T. O'Reilly. REST vs. SOAP at Amazon. <http://www.oreillynet.com/pub/wlg/3005?wlg=yes>

- [17] Matei Ripeanu, Adriana Iamnitchi S4: A Simple Storage Service for Sciences, , 16th IEEE International Symposium on High Performance Distributed Computing (HPDC) - Hot Topics Track, Monterey Bay, CA, June 2007.
- [18] David Mazières, M. Frans Kaashoek, Escaping the Evils of Centralized Control with Self-Certifying Pathnames, In Proceedings of ACM SIGOPS, September 1998.