Evaluation of Cloud-RAID: A Secure and Reliable Storage Above the Clouds

Maxim Schnjakin, Christoph Meinel
Hasso Plattner Institute
Potsdam University, Germany
Prof.Dr-Helmert-Str. 2-3, 14482 Potsdam, Germany
maxim.schnjakin, office-meinel@hpi.uni-potsdam.de

***Abstract*—Cloud Computing as a service-on-demand architecture has grown in importance over the previous few years. One driver of its growth is the ever increasing amount of data which is supposed to outpace the growth of storage capacity. The usage of cloud technology enables organizations to manage their data with low operational expenses. However, the benefits of cloud computing come along with challenges and open issues such as security, reliability and the risk to become dependent on a provider for its service. In general, a switch of a storage provider is associated with high costs of adapting new APIs and additional charges for inbound and outbound bandwidth and requests. In this paper, we present a system that improves availability, confidentiality and reliability of data stored in the cloud. To achieve this objective, we encrypt user's data and make use of the RAID-technology principle to manage data distribution across cloud storage providers.**

We conduct a proof-of-concept testbed experiment for our application to evaluate the performance and cost effectiveness of our approach. We deployed our application using eight commercial cloud storage repositories in different countries. Our approach allows users to avoid vendor lock-in, and reduces significantly the cost of switching providers. We also observed that our implementation improved the perceived availability and, in most cases, the overall performance when compared with individual cloud providers. Moreover, we estimated the monetary costs to be competitive to the cost of using a single cloud provider.

I. Introduction

Cloud Computing is a concept of utilizing computing as an on-demand service. It fosters operating and economic efficiencies and promises to cause an unanticipated change in business. Using computing resources as pay-as-you-go model enables service users to convert fixed IT cost into a variable cost based on actual consumption. Therefore, numerous authors argue for the benefits of cloud computing focusing on the economic value [9], [2].

However, despite of the non-contentious financial advantages cloud computing raises questions about privacy, security and reliability. Among available cloud offerings, storage services reveal an increasing level of market competition. According to iSuppli [7] global cloud storage revenue is set to rise to $5 billion in 2013, up from $1.6 billion in 2009. One reason is the ever increasing amount of data which is supposed to outpace the growth of storage capacity. Currently, it is very difficult to estimate the actual future volume of data but there are different estimates being published. According to IDC review [12], the amount of digital information created and replicated is estimated to surpass 3 zettabytes by the end of this year. This amount is supposed to more than double in the next two years. In addition, the authors estimate that today there is 9 times more information available than was available five years ago.

However, for a customer (service) to depend solely on one cloud storage provider (in the following provider) has its limitations and risks. In general, vendors do not provide far reaching security guarantees regarding the data retention [13]. Users have to rely on effectiveness and experience of vendors in dealing with security and intrusion detection systems. For missing guarantees service users are merely advised to encrypt sensitive content before storing it on the cloud. Placement of data in the cloud removes the physical control that a data owner has over data. So there is a risk that service provider might share corporate data with a marketing company or use the data in a way the client never intended.

Further, customers of a particular provider might experience vendor lock-in. In the context of cloud computing, it is a risk for a customer to become dependent on a provider for its services. Common pricing schemes foresee charging for inbound and outbound transfer and requests in addition to hosting the actual data. Changes in features or pricing scheme might motivate a switch from one storage service to another. However, because of the data inertia, customers may not be free to select the optimal vendor due to immense costs associated with a switch of one provider to another. The obvious solution is to make the switching and data placement decisions at a finer granularity then all-or-nothing. This could be achieved by distributing corporate data among multiple storage providers. Such an approach is pursued by content delivery networks (for example in [6], [8]) and implies significant higher storage and bandwidth costs without taking into account the security concerns regarding the retention of data.

A more economical approach, which is presented in this paper, is to separate data into unrecognizable slices, which are distributed to providers - whereby only a subset of the nodes needs to be available in order to reconstruct the original data. This is indeed very similar to what has been

done for years at the level of file systems and disks. In our work we use RAID like techniques to overcome the mentioned limitations of cloud storage in the following way:

1) **Security.** The provider might be trustworthy, but malicious insiders represent a well known security problem. This is a serious threat for critical data such as medical records, as cloud provider staff has physical access to the hosted data. We tackle the problem by encrypting and encoding the original data and later by distributing the fragments transparently across multiple providers. This way, none of the storage vendors is in an absolute possession of the client's data. Moreover, the usage of enhanced erasure algorithms enables us to improve the storage efficiency and thus also to reduce the total costs of the solution.
2) **Service Availability.** Management of computing resources as a service by a single company implies the risk of a single point of failure. This failure depends on many factors such as financial difficulties (bankruptcy), software or network failure, etc. In July 2008, for instance, Amazon storage service S3 was down for 8 hours because of a single bit error [24]. Our solution addresses this issue by storing the data on several clouds - whereby no single entire copy of the data resides in one location, and only a subset of providers needs to be available in order to reconstruct the data.
3) **Reliability.** Any technology can fail. According to a study conducted by Kroll Ontrack[1] 65 percent of businesses and other organizations have frequently lost data from a virtual environment. A number that is up by 140 percent from just last year. Admittedly, in recent times, no spectacular outages were observed. Nevertheless failures do occur. For example, in October 2009 a subsidiary of Microsoft, Danger Inc., lost the contracts, notes, photos, etc. of a large number of users of the Sidekick service [19]. We deal with the problem by using erasure algorithms to separate data into packages, thus enabling the application to retrieve data correctly even if some of the providers corrupt or lose the entrusted data.
4) **Data lock-in.** By today there are no standards for APIs for data import and export in cloud computing. This limits the portability of data and applications between providers. For the customer this means that he cannot seamlessly move the service to another provider if he becomes dissatisfied with the current provider. This could be the case if a vendor increases his fees, goes out of business, or degrades the quality of his provided services. As stated above, our solution does not depend on a single service provider. The data is balanced among several providers taking into account user expectations regarding the price and availability of the hosted content. Moreover, with erasure codes we store only a fraction of the total amount of data on each cloud provider. In this way, switching one provider for another costs merely a fraction of what it would be otherwise.

[1]http://www.krollontrack.com/resource-library/case-studies/

In recent months we conducted an extensive experiment for our application to evaluate the overall performance and cost effectiveness of the approach. In this paper we present the results of the experimental study. We show, that with an appropriate coding configuration Cloud-RAID is able to improve significantly the performance of the data transmission process, whereby the monetary costs are competitive to the cost of using a single cloud.

II. Architecture

The ground of our approach is to find a balance between benefiting from the cloud's nature of pay-per-use and ensuring the security of the company's data. As mentioned above, the basic idea is not to depend on solely one storage provider but to spread the data across multiple providers using redundancy to tolerate possible failures. The approach is similar to a service-oriented version of RAID (Redundant Arrays of Inexpensive Disks). While RAID manages sector redundancy dynamically across hard-drives, our approach manages file distribution across cloud storage providers. RAID 5, for example, stripes data across an array of disks and maintains parity data that can be used to restore the data in the event of disk failure. We carry the principle of the RAID-technology to cloud infrastructure. In order to achieve our goal we foster the usage of erasure coding technics (see chapter III). This enables us to tolerate the loss of one or more storage providers without suffering any loss of content [25], [11]. The system has a number of core components that contain the logic and management layers required to encapsulate the functionality of different storage providers. Our architecture includes the following main components:

- **User Interface Module.** The interface presents the user a cohesive view on his data and available features. Here users can manage their data and specify requirements regarding the data retention (quality of service parameters).
- **Resource Management Module.** This system component is responsible for intelligent deployment of data based on users' requirements. The component is supported by:
  - a registry and matching service: assigns storage repositories based on users requirements (for example physical location of the service, costs and performance expectations). Monitors the performance of participating providers and ensures that they are meeting the agreed SLAs
  - a resource management service: takes operational decisions regarding the content storage

  - a task scheduler service: has the ability to schedule the launch of operations at peak-off hours or after specified time intervals.

- **Data Management Module.** This component handles data management on behalf of the resource management module and is mainly supported by:
  - a data encoding service: this component is responsible for striping and encoding of user content
  - a data distribution service: spreads the encoded data packages across multiple providers. Since each storage service is only accessible through a unique API, the service utilizes storage "service-connectors", which provide an abstraction layer for the communication to storage repositories
  - a security service: manages the security functionality based on a user's requirements (encryption, secret key management).

Further details can be found in our previous work [23], [21], [20] and [14].

III. Evaluation

In this section we present an evaluation of our system that aims to clarify the main questions concerning the cost, performance and availability aspects when erasure codes are used to store data on public clouds.

A. Methodology

The experiment was run on Hasso Plattner Institute (HPI), which is located close to Berlin, Germany, over a period of over 377 (24x7) hours, in the middle of July 2012. As it spans seven days, localized peak times (time-of-day) is experienced in each geographical region. HPI has a high speed connectivity to an Internet backbone (1 Gb), which ensures that our test system is not a bottleneck during the testing. The global testbed spans eight cloud providers in five countries on three continents. The experiment time comprises three rounds, with each round consisting of a set of predefined test configurations (in the following sequences). Table I provides a summary of the conducted experiment. We used test files of different sizes from 100 kB up to 100 MB, deployed by the dedicated test clients.

Prior to each test round the client requires a persistent connection to the APIs of the relevant cloud storage providers, so that requests for an upload or download of test data can be send. In general, providers will refuse a call for the establishment of a new connection after several back-to-back requests. Therefore we implemented an API-connection holder. After two hours of an active connection the old connection is overwritten by a new one. Further, we determine a timeout of one second between two unsuccessful requests, each client waits for a think time before the next request is generated.

Category	**Description**
Cloud storage provider	8
Locations	Europe, USA, Asia
Total experiment time	about 15d 9h (377h)
Total number of test rounds	about 3 rounds
Total number of requests (read/write) / round	281,900
Service time out for each request	1 sec
Test file size	100 kB - 100 MB
Coding Method	cauchy_good
Coding configuration [k,m]	k=[2..4,6,10], m=[1..2], k>=m

Table I
Experiment Details

1) Machines for Experimentation: We employed three machines for experimentation. Neither is exceptionally high-end, but each represents middle-range commodity processor, which should be able to encode, encrypt, decrypt and decode comfortably within the I/O speed limits of the fastest disks. These are: Windows 7 Enterprise (64bit) system with an Intel Core 2 Duo E8400 @3GHz, 4 GB installed RAM and a 160 GB SATA Seagate Barracuda hard drive with 7200 U/min.

2) Experiment Setup: Figure 1 presents the workflow of the experiment. In general we use two machines to transfer test data to cloud storage providers. The first machine (the upper part of the graph) uses erasure codes. This means, upon receiving a write request the test system splits the incoming object into k data fragments of an equal size - $chunks$. These k data packages hold the original data. In the next step the system adds m further packages whose contents are calculated from the k chunks, whereby k and m are variable parameters [15]. With this, the act of encoding takes the contents of k data packages and encodes them on m coding packages. In turn, the act of decoding takes some subset of the collection of $n = k + m$ total packages and from them recalculates the original data. Any subset of k shares is sufficient to reconstruct the original data object [18].

In the next step, the application makes sure that each data package is sent to a different storage repository. In general, our system follows a model of one thread per provider per data package in such a way that the encoding, encryption, decryption, and provider accesses can be executed in parallel.

The second machine (the lower part of the graph in the figure 1) uploads the entire data object to a single provider without any modifications. As we are interested in the direct comparison between these two approaches, we want each data transmission to start simultaneously. Therefore we used the third machine as a "sync-instance" running a Tomcat 7 server with a self-written sync-servlet which controls the workflow of the experiment.

3) Erasure Configuration: In our experiment we make use of the Cauchy-Reed-Solomon algorithm for two reasons.

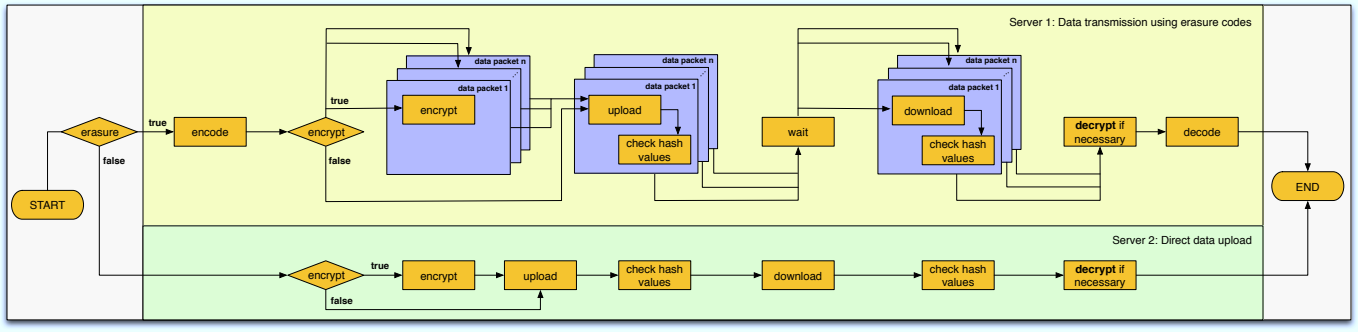


Figure 1. Workflow of the experiment

First, according to Plank et al. [17] the algorithm has a good performance characteristics in comparison to existing codes. In their work, the authors performed a head-to-head comparison of numerous open-source implementations of various coding techniques which are available to the public. Second, the algorithm allows free selection of coding parameters k and m, whereas other algorithms restrict the choice of parameters. Liberation Code [16] for example is a specification for storage systems with $n = k + 2$ nodes to tolerate the failure of any two nodes (whereby the parameter m is fix and is equal to two).

In our test scenario we tested more than 2520 combinations of k and m. We will denote them by $[k, m]$ in the course of the paper, whereby the present evaluation focuses on an encoding configuration [4,1]. Which means, that the setting provides data availability toward one cloud failure at the time of read or write request. Most of the providers have SLAs with 99% and 99.9% monthly up-time percentages. Thus, we believe that adding enough redundancy to tolerate one provider outage or failure at a time will be sufficient in most cases. The automated determination of the appropriate m and k values remains a subject of future work.

B. Schemes and Metrics

The goal of our test is to evaluate the performance of our approach. Mainly we are interested in availability of APIs, overhead caused by erasure codes and transmission rates. Therefore, we implemented a simple logger application to record the results of our measurements. In total we log 34 different events. For example, each state of the workflow depicted in figure 1 is captured with two log entries (START and END).

1) Erasure Overhead: Due to the nature of erasure codes, each file upload and download is associated with a certain overhead. On one hand this overhead is caused by the redundant m packages, which have to be stored, uploaded and sometimes downloaded (in the events of failure). The total size of all chunks (after encoding) can be expressed with the following equation: $s*(1+\frac{m}{k})$, whereby variable s is defined as the original file size. With this, the usage of erasure codes increases the total storage by a factor of $\frac{m}{k}$. Further, we need to encode data prior to its upload and accordingly decode the downloaded packets into the original file. Both operations cause an additional computational expense.

2) Transmission Performance and Throughput: We measure the throughput obtained from each read and write request. In general the throughput is defined as the average rate of successful message delivery over a communication channel. In our work we link the success of the message delivery to the success of the delivery of the entire data object. In our approach, a data object is completely transferred, when the last data package is being successfully transferred to the transfer destination. This means that in case of data upload, the transfer is only completed, when (upon a write request) our client receives a confirmation message in the form of individual digest values that correspond with the results of the local computation (this applies for all transferred data packages). In the event of a mismatch the system will delete the corrupted data and initiate a reupload procedure. With this, the value of throughput does not only represent the pure upload or download rate of the particular providers, as the measured time span includes also possible failures, latency and the bilateral processing of get-hash calls.

C. Empirical Results

This section presents the results in terms of read and write performance, as well as throughput, response time and availability based on over 281.000 requests. Due to space constraints, we present only some selected results from the conducted experiment.

1) Erasure Overhead: As described in III-B1 the erasure coding leads to a storage overhead of factor $\frac{m}{k}$. For instance, an $[k = 4, m = 1]$ encoding results in a storage overhead of $\frac{1}{4} * 100\% = 25\%$. In order to reduce the storage overhead, it would be advisable to define high k and preferably low m values. For example, an encoding configuration $[k = 10, m = 1]$ produces a storage overhead of only $\frac{1}{10} * 100\% = 10\%$. Erasure causes also a computational

overhead. During the experiment we scrutinized 12 different configurations. A selection of the results is presented in figure 2. The figure illustrates, that the computational expense increases with the file size regardless of the erasure configuration. As the encoding of a 100 MB data object takes approximately one second, the encoding overhead can be neglected in view of the significantly higher transmission times. In [14] we showed, that the average performance overhead caused by data encoding is less than 2% of the entire data transfer process to a cloud provider.

Using encryption, we can say that the total performance decreases as individual data packages have to be encrypted locally before moving them to the cloud. In our experiments the costs for encryption were less than 3% of total time which is also negligible in view of the overall transmission performance. This point has been addressed in our previous work [14] and [22].

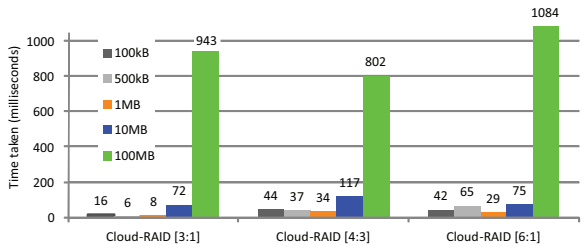


Figure 2. The computational overhead caused by erasure with different configurations and file sizes. In general, the overall overhead increases with growing file size regardless of the defined m and k parameters for the erasure configuration.

2) Transmission Performance and Throughput: Due to space constraints the current evaluation focuses on the Cloud-RAID configuration with $k = 4$ and $m = 1$. For performance comparison we experimented with different combinations among eight clouds, which are: Amazon US, Amazon EU, Azure, Box, Google EU, Google US, Nirvanix and Rackspace. The particular combinations are represented in table II.

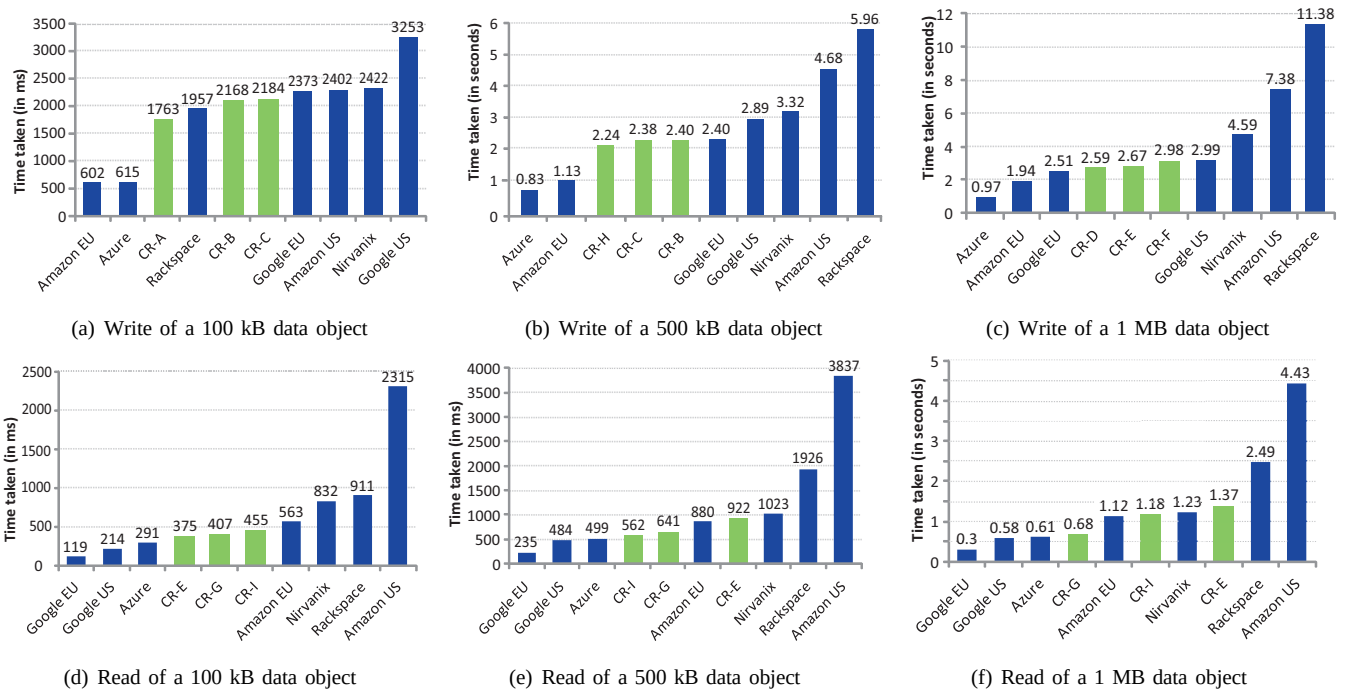
In general, we observed that utilizing Cloud-RAID for data transfer improves the throughput significantly when compared with cloud storages individually. This can be explained with the fact, that Cloud-RAID reads and writes a fraction of the original data (more specific $\frac{1}{4}$th with [4,1] setting, see III-B1) from and to clouds simultaneously. However, the total time of data transfer depends on the throughput performance of each provider involved into the communication process. The throughput performance of Cloud-RAID increases with higher performance values of cloud providers involved into the data distribution setting. During the performance evaluation we observed, that storage providers differ extremely in their upload and download capabilities. Moreover, some vendors seem to have optimized their infrastructure for large files, while others focused more on smaller data objects. In the following we will clarify this point.

As we mentioned above there is a striking difference in the up- and download capabilities of cloud services. Except Microsoft Azure all the tested providers are much faster in download than in upload. This applies to smaller and larger data objects. At one extreme, with Google EU or Google US services a write request of a 100 kB file takes up to 19 times longer than a read request (see figures 3a and 3d). This behavior can also be observed with larger data objects (although less pronounced). Here the difference in the throughput rate may range from 4 to 5 times, with the exception of the provider Rackspace, where an execution of a write request is up to 49 times slower than of a read request (e.g. an upload of a 100 MB file takes on average 17,3 minutes, whereas the download of the same file is performed in less than 21 seconds, see figures 4b and 4d). Then again, Google US service improves its performance clearly with the growing size of data objects (see figures 3a and 4a). The explanation for this could be that with larger files the relatively long reaction time of the service (due to the long distance between our test system and the service node) has less impact on the measuring results. Similar to the US service Google EU performs rather mediocre in comparison to other providers when it comes to read speeds for data objects up to 1 MB, (see figures 3a and 3b). In terms of performance for writing larger files, Google EU becomes the clear leader and even outperforms the fastest Cloud-RAID setting, which consists of the five fastest providers: Amazon EU, Azure, Google EU, Google US and Nirvanix (see figure 4b).

Similar phenomena have been observed by read requests. Microsoft Azure belongs to the leading providers for reading 100 kB data objects (see figure 3d) and falls back by reading 100 MB files (see figure 4d).

Hence, the performance of Cloud-RAID differs depending on the provider setting and file size. It is observed that our systems achieves better throughput values for read requests. The reason is that the test client fetches less data from the cloud (only k of n data packages) than in case of a write request, where all n packages have to be moved to the cloud.

As expected, we observe that the fastest read and write settings consist of the fastest clouds. Concerning writing 100 kB data objects, the fastest Cloud-RAID setting CR-A improves the overall throughput by an average factor of 3 (compared to the average throughput performance of the providers in the current Cloud-RAID setting). For reading 100 kB, CR-E achieves an improvement factor of 5. In terms of performance for writing 1 MB and 10 MB objects, Cloud-RAID setting CR-D and CR-E achieve already an average improvement factor of 7. Then again, for reading 10 MB, Cloud-RAID improves the average performance by a factor of 13 and even outperforms the fastest cloud providers (see figure 4c). By smaller data objects, execution



(a) Write of a 100 kB data object

(b) Write of a 500 kB data object

(c) Write of a 1 MB data object

(d) Read of a 100 kB data object

(e) Read of a 500 kB data object

(f) Read of a 1 MB data object

Figure 3. Average throughput performance in milliseconds and seconds observed on all reads and writes executed for the [4,1] Cloud-RAID configuration (4 of 5 data packages are necessary to reconstruct the original data, m = 1). The Cloud-RAID bars (CR) correspond to the complete data processing cycle: the encoding of a data object into data packages and the subsequent transmission of individual chunks in parallel threads.

Cloud-RAID	Provider Setting
CR-A	Amazon EU, Amazon US, Azure, Nirvanix, Rackspace
CR-B	Amazon EU, Amazon US, Azure, Google EU, Rackspace
CR-C	Amazon US, Azure, Google EU, Nirvanix, Rackspace
CR-D	Amazon EU, Amazon US, Azure, Google EU, Nirvanix
CR-E	Amazon EU, Azure, Google EU, Google US, Nirvanix
CR-F	Amazon EU, Google EU, Google US, Nirvanix, Rackspace
CR-G	Amazon EU, Amazon US, Azure, Google EU, Google US
CR-H	Amazon EU, Amazon US, Google EU, Google US, Nirvanix
CR-I	Amazon EU, Azure, Google EU, Google US, Rackspace
CR-K	Amazon EU, BoxNet, Google EU, Google US, Nirvanix
CR-L	Amazon EU, Amazon US, BoxNet, Google EU, Google US
CR-M	Amazon EU, Amazon US, Azure, BoxNet, Google EU

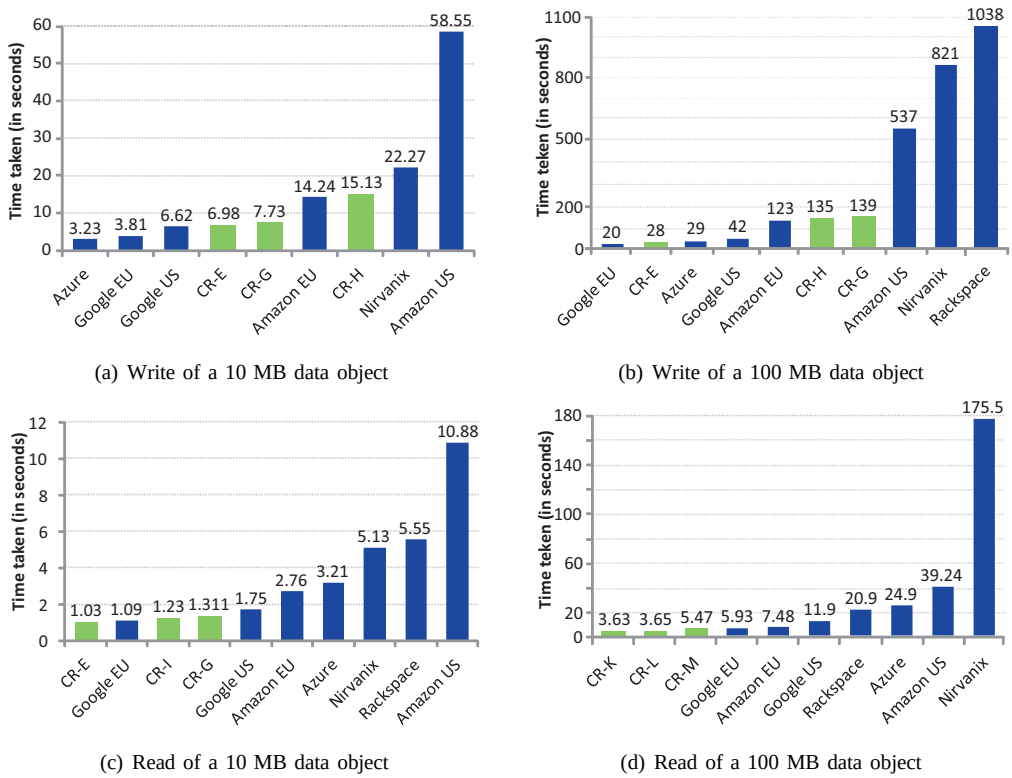
Table II
CLOUD-RAID SETTING WITH $k = 4$ AND $m = 1$.

of both read and write requests is highly affected by erasure overhead, DNS lookup and API connection establishment time. This can lead to an unusual behavior. For example, the transmission of a 100 kB data object to Google US can take our system more time than the transmission of a 500 kB or even 1 MB file (see figure 3a, 3b and 3c). Hence, increasing the size of data objects improves the overall throughput of Cloud-RAID. Concerning read and write speeds for 100 MB data objects, Cloud-RAID increases the average performance by a factor of 36 for writes (despite of the erasure overhead of 25 percent) and achieves an improvement factor of 55 for reads (see figures 4c and 4d).

There is also an observed connection between the throughput rate and the size of data objects. Charts 3a to 3f show results from performance tests on smaller files (up to 1 MB). Microsoft Azure and Amazon EU achieve the best results in terms of write requests. When writing 10 MB or 100 MB data objects Amazon EU falls back on the fourth place (see figures 4b and 4d). Form these observations, we come to the following conclusions. The overall performance of Cloud-RAID is not only dependent on the selection of k and m values, but also on the throughput performance of the particular storage providers. Cloud-RAID increases the overall transmission performance compared to the slower providers. Beyond that we are able to estimate, that the more providers are involved into the data distribution process, the less weight slower providers carry in terms of overall throughput performance. The underlying reason is again the size of individual data packages, which decrease with the growing number of k data packages (see chapter III-B1).

D. Observations and Economic Consequences

Finally, based on the measured observations, we determine users benefits from using our system. In order to assert the feasibility of our application we have to examine the cost structure of cloud storage services. Vendors differ in pricing scheme and performance characteristics. Some providers charge a flat monthly fee, others negotiate contracts with individual clients. However, in general pricing depends on the amount of data stored and bandwidth consumed in



(a) Write of a 10 MB data object

(b) Write of a 100 MB data object

(c) Read of a 10 MB data object

(d) Read of a 100 MB data object

Figure 4. Throughput observed in seconds on reads and writes executed for the [4,1] Cloud-RAID configuration. Here again, CR bars correspond to the complete data processing cycle.

transfers. Higher consumption results in increased costs. As illustrated in tables III and IV providers also charge per API request (such as read, write, get-hash, list etc.) in addition to bandwidth and storage. The usage of erasure codes increases the total number of such requests, as we divide each data object into chunks and stripe them over multiple cloud vendors. The upload and download of data takes on average two requests. Considering this, our system needs $(4+1)*2=10$ requests for a single data upload with a $[4,1]$ coding configuration. The download requires only $4*2=8$ requests, as merely 4 packets have to be received to rebuild the original data. Thus, erasure $[k,m]$ increases the number of requests by a factor of $k+m$ for upload and k for download. Consequently, the usage of erasure codes increases the total cost compared to a direct upload or download of data due to the caused storage and API request overhead. Tables III and IV summarize the cost in US Dollars of executing 10,000 reads and 10,000 writes with our system considering 5 data unit sizes: 100 kB, 500 kB, 1 MB, 10 MB and 100 MB. We observe, that the usage of erasure is not significantly more expensive than using a single provider. In some cases the costs can be even reduced.

IV. Related Work

The main idea underlying our approach is to provide RAID technique at the cloud storage level. In [5] the authors

Provider	Filesize in kB				
	100	500	1024	10240	102400
CR-B	0.15	0.55	1.07	10.21	101.61
CR-G	0.16	0.52	0.99	9.28	92.25
CR-I	0.15	0.55	1.07	10.21	101.61
CR [6,1][1]	3.61	4.12	4.78	16.50	133.69
Azure	0.11	0.53	1.08	10.74	107.42
Amazon/Google	0.13	0.59	1.19	11.74	117.21
Rackspace	0.17	0.86	1.76	17.58	175.78
Nirvanix	4.14	4.72	5.46	18.65	150.48

[1] The setting CR [6,1] consist of nearly all providers involved in the test setting: Amazon EU, Amazon US, Azure, Boxnet, Google EU, Nirvanix, Rackspace.

Table III
COSTS IN $ FOR 10,000 READS.

Provider	Filesize in kB				
	100	500	1024	10240	102400
CR-B	0.12	0.12	0.12	0.12	0.12
CR-G	0.16	0.16	0.16	0.16	0.16
CR-I	0.12	0.12	0.12	0.12	0.12
CR [6,1]	8.14	8.20	8.29	9.75	24.40
Azure	0.00	0.00	0.00	0.00	0.00
Amazon/Google	0.02	0.02	0.02	0.02	0.02
Rackspace	0.00	0.00	0.00	0.00	0.00
Nirvanix	4.10	4.48	4.98	13.77	101.66

Table IV
COSTS IN $ FOR 10,000 WRITES.

introduce the HAIL (High-Availability Integrity Layer) system, which utilizes RAID-like methods to manage remote file integrity and availability across a collection of servers or independent storage services. The system makes use of challenge-response protocols for retrievability (POR) [3] and proofs of data possession (PDP) [3] and unifies these two approaches. In comparison to our work, HAIL requires storage providers to run some code whereas our system deals with cloud storage repositories as they are. Further, HAIL does not provide confidentiality guarantees for stored data. In [10] Dabek et al. use RAID-like techniques to ensure the availability and durability of data in distributed systems. In contrast to the mentioned approaches our system focuses on the economic problems of cloud computing described in chapter I.

Further, in [1] authors introduce RACS, a proxy that spreads the storage load over several providers. This approach is similar to our work as it also employs erasure code techniques to reduce overhead while still benefiting from higher availability and durability of RAID-like systems. Our concept goes beyond a simple distribution of users' content. RACS lacks sophisticated capabilities such as intelligent file placement based on users' requirements or automatic replication. In addition to it, the RACS system does not try to solve security issues of cloud storage, but focuses more on vendor lock-in. Therefore, the system is not able to detect any data corruption or confidentiality violations.

The future of distributed computing has been a subject of interest for various researchers in recent years. The authors in [8] propose an architecture for market-oriented allocation of resources within clouds. They discuss some existing cloud platforms from the market-oriented perspective and present a vision for creating a global cloud exchange for trading services. The authors consider cloud storage as a low-cost alternative to dedicated Content Delivery Networks (CNDs).

There are more similar approaches dealing with high availability of data trough its distribution among several cloud providers. DepSky-A [4] protocol improves availability and integrity of cloud-stored data by replicating it on cloud providers using quorum techniques. This work has two main limitations. First, a data unit of size S consumes n x S storage capacity of the system and costs on average n times more than if was stored on a single cloud. Second, the protocol does not provide any confidentiality guaranties, as it stores the data in clear text. In their later work the authors present DepSky-CA, which solves the mentioned problems by the encryption of the data and optimization of the write and read process. However, the monetary costs of using the system is still twice the cost of using a single cloud. On top of this, DepSky does not provide any means or metrics for user centric data placement. In fact, our approach enables cloud storage users to place their data on the cloud based on their security policies as well as quality of service expectations and budget preferences.

V. Conclusion

In this paper we outlined some general problems of cloud computing such as security, service availability and a general risk for a customer to become dependent on a service provider. In the course of the paper we demonstrated how our system deals with the mentioned concerns. In a nutshell, we stripe users' data across multiple providers while integrating with each storage provider via appropriate service-connectors. These connectors provide an abstraction layer to hide the complexity and differences in the usage of storage services.

The main focus of the paper is an extensive evaluation of our application. From the results obtained, we conclude that our approach improves availability at costs similar to using a single commercial cloud storage provider (instead of 100% and more when full content replication is used).

We use erasure code techniques for striping data across multiple providers. The experiment proved, that given the speed of current disks and CPUs, the libraries used are fast enough to provide good performance - whereby the overall performance depends on the throughput performance of the particular storage providers. The throughput performance of Cloud-RAID increases with the selection of providers with higher throughput performance values. Hence, with an appropriate coding configuration Cloud-RAID is able to improve significantly the data transmission process when compared with cloud storages individually.

Further, performance tests showed that our system is best utilized for deployment of large files. Utilization of our system for storing of smaller data objects is subject to further test and analysis.

In the long term, our approach might foster the provision of new and even more favorable cloud storage services. Today, storage providers surely use RAID like methods to increase the reliability of the entrusted data to their customers. The procedure causes costs which are certainly covered by providers price structure. With our approach, the on-site backups might become redundant, as users data is distributed among dozens of storage services. Furthermore, we enable users of cloud storage services to control the availably and physical segregation of the data by themselves.

However, additional storage offerings are expected to become available in the next few years. Due to the flexible and adaptable nature of our approach, we are able to support any changes in existing storage services as well as incorporating support for new providers as they appear.

VI. Future work

Our performance testing revealed that some vendors have optimized their systems for large data objects and high upload performance, while others have focused on smaller files and better download throughput. We will use these observations to optimize read and write performance of our application. During our experiment we also observed that the

reaction time of read and get-hash requests may vary from provider to provider at different times of day. This behavior might be related to the usage of different consistency models and is subject of further analysis.

In addition, we are also planing to implement more service connectors and thus to integrate additional storage services. Any extra storage resource improves the performance and responsiveness of our system for end-users.

REFERENCES

[1] Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. Racs: A case for cloud storage diversity. *SoCC'10*, June 2010.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. *14th ACM CCS*, 2007.

[4] Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. Depsky: dependable and secure storage in a cloud-of-clouds. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 31–46, New York, NY, USA, 2011. ACM.

[5] Kevin D. Bowers, Ari Juels, and Alina Oprea. Hail: A high-availability and integrity layer for cloud storage. *CCS'09*, November 2009.

[6] James Broberg, Rajkumar Buyya, and Zahir Tari. Creating a 'cloud storage' mashup for high performance, low cost content delivery. *Service-Oriented Computing (Volume 5472), ICSOC'08 Workshops*, April 2009.

[7] Jeffrey Burt. Future for cloud computing looks good, report says. online, 2009.

[8] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, August 2008.

[9] Nicholas Carr. *The Big Switch*. Norton, 2008.

[10] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with cfs. *ACM SOSP*, October 2001.

[11] R. Dingledine, M. Freedman, and D. Molnar. The freehaven project: Distributed anonymous storage service. *The Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

[12] John Gantz and David Reinsel. Extracting value from chaos. online, 2009.

[13] Ponemon Institute. Security of cloud computing providers study. online, April, 2011.

[14] Implementation of a Secure and Reliable Storage Above the Untrusted Clouds. Platform for a secure storage-infrastructure in the cloud. *Proceedings of 8th International Conference on Computer Science and Education – ICCSE 2013*, 2013.

[15] J. S. Plank, S. Simmerman, and C. D. Schuman. Jerasure: A library in C/C++ facilitating erasure coding for storage applications - Version 1.2. Technical Report CS-08-627, University of Tennessee, August 2008.

[16] James S. Plank. The raid-6 liberation codes. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, FAST'08, pages 7:1–7:14, Berkeley, CA, USA, 2008. USENIX Association.

[17] James S. Plank, Jianqiang Luo, Catherine D. Schuman, Lihao Xu, and Zooko Wilcox-O'Hearn. A performance evaluation and examination of open-source erasure coding libraries for storage. In *Proccedings of the 7th conference on File and storage technologies*, FAST '09, pages 253–265, Berkeley, CA, USA, 2009. USENIX Association.

[18] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz. Maintenance free global storage in oceanstore. *IEEE Internet Computing*, September 2001.

[19] David Sarno. Microsoft says lost sidekick data will be restored to users. *Los Angeles Times*, October 2009.

[20] Maxim Schnjakin, Rehab Alnemr, and Christoph Meine. A security and high-availability layer for cloud storage. In *Web Information Systems Engineering – WISE 2010 Workshops*, volume 6724 of *Lecture Notes in Computer Science*, pages 449–462. Springer Berlin / Heidelberg, 2011.

[21] Maxim Schnjakin, Rehab Alnemr, and Christoph Meinel. Contract-based cloud architecture. In *Proceedings of the second international workshop on Cloud data management*, CloudDB '10, pages 33–40, New York, NY, USA, 2010. ACM.

[22] Maxim Schnjakin, Michael Goderbauer, Martin Krueger, and Christoph Meinel. Cloud storage and it-security. *Proceedings of the 13th Deutscher IT-Sicherheitskongress (Sicherheit 2013)*, 2013.

[23] Maxim Schnjakin and Christoph Meinel. Platform for a secure storage-infrastructure in the cloud. *Proceedings of the 12th Deutscher IT-Sicherheitskongress (Sicherheit 2011)*, 2011.

[24] The Amazon S3 Team. Amazon s3 availability event: July 20, 2008. online, 2008.

[25] H. Weatherspoon and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. *IPTPS*, March 2002.