published as: Maxim Schnjakin, Rehab Alnemr, Christoph Meinel: A Security and High-Availability Layer for Cloud Storage
In Proceedings of the 1st International Workshop (In conjunction with WISE 2010) on Cloud Information System Engineering (CISE 2010), Springer LNCS, Hong Kong, China, December 2010

A Security and High-Availability Layer for Cloud Storage

Maxim Schnjakin, Rehab Alnemr, and Christoph Meinel

Hasso Plattner Institute, Prof.-Dr.-Helmertstr. 2-3, 14482 Potsdam, Germany
{maxim.schnjakin,rehab.elnemr,meinel}@hpi.uni-potsdam.de

Abstract. Cloud Computing as a service on demand architecture has become a topic of interest in the last few years. The outsourcing of duties and infrastructure to external parties enables new services to be established quickly, scaled on demand, and with low financial risk. Cloud storage enables organizations to manage their data with low operational expenses. Nevertheless, several issues such as security and the risk to become dependent on a provider for its service should be considered before entering the cloud. 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 use the principle of RAID-technology in cloud infrastructure to manage data distribution across cloud storage providers. The distribution is based on users expectations regarding providers geographic location, quality of service, providers reputation, and budget preferences. Our approach allows users to avoid vendor lock-in, reduce cost of switching providers and increase security and availability of their data. We also explain on how the proposed system removes the complexity of interacting with multiple storage providers while maintaining security.

1 Introduction

Cloud Computing is a concept of utilizing computing as an on-demand service. It fosters operable and economic efficiencies and promises to cause an unanticipated change in business. Numerous authors argue for the benefits of cloud computing focusing on the economic value [11], [6]. Using computing resources as pay-as-you-go model enables companies to convert the fixed IT cost into a variable cost based on actual consumption. However, despite the non-contentious financial advantages cloud computing raises questions about privacy, security, reliability, and legislation. Among available cloud offerings, *storage service* reveals an increasing level of market competition. According to iSuppli [9] global cloud storage revenue is set to rise to $5 billion in 2013, up from $1.6 billion in 2009. The same market research states that the growth could be much better if concerns over privacy security were removed. Beyond the self-motivated interest in securing organization's data pool, several laws demand public and private organizations to protect the security of their information systems [19]. The European Union's Data Protection Directive (EU DPD), for instance, has clear

D.K.W. Chiu et al. (Eds.): WISE 2010 Workshops, LNCS 6724, pp. 449–462, 2011.

restrictions for the movement, processing, and access of specific types of data across political borders. Some of these laws cover markets such as the financial markets or health care industry. Any organization that does business in countries with existing legal restrictions regarding information security is subject to these laws.

In general, vendors do not provide far reaching security guarantees regarding the data retention. Users have to rely on vendors experience 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 many of direct physical controls of this data by the data owner. 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. Another problem is management of information life-cycle. Data destruction is extremely difficult in multi-tenant environment. Cloud computing introduces a level of abstraction that masks much of the physical infrastructure. Clients often do not know what really happens with their data when it is deleted at the application level or, more specifically, infrastructure level.

However, a company risks data/vendor lock-in when it depends solely on a single cloud storage provider. Common pricing schemes foresee charging for inbound and outbound transfer and requests in addition to hosting of the actual data. Changes in features or pricing scheme might motivate a switch from one storage service to onother. However, because of the data inertia, customers may not be free to select the optimal vendor due to immense costs of the switch. The obvious solution is to make the switching and data placement decisions at a finer granularity than all-or-nothing. This could be achieved by spreading of corporate data among multiple storage providers. Which leads to two problems: a] the customer is required to watch continuously for storage offerings and proper selection can be difficult because of market competitiveness, b] services can be difficult to use for non-developers, as each service is best used via particular web services or API, and has its own limitations.

Service selection -in its own- is an issue due to: the lack of general purpose and reusable framework to interact with multiple storage vendors, and the similarity of services functionalities. Therefore, selection should depend on the quality and the nature of the offerings, customer's preferences, and provider's reputation. In context of web services, this issue is tackled by *quality of service descriptions* within service level agreements (SLAs). However, the problem of handling service level management in inter-domain scenarios is not entirely solved up to present [15], [22].

For each potential cloud customer, it is both expensive and time consuming to handle these security and usability concerns. They will have to perform a scrutiny on the security capabilities of the service provider independently that include: studying security policies and service level agreements (SLA) that are usually written in a plain natural language[5], and inspecting the facilities of the service provider. Carrying on these two tasks is indeed inefficient as well as time consuming. Therefore, it is only logical to have a third party who is

specialized in legal and security matters to monitor and audit such tasks. In this paper, we propose an architecture that uses a trusted third party to ensure and to supervise the compliance of user's requirements. Furthermore, it acts as an intermediate layer between client and various storage services. We tackle the aforementioned problems by fragmenting and permuting the original data and later by spreading the fragments across multiple providers. This way, none of the storage vendors is in an absolute possession of clients data. Our approach tries to solve a number of general problems associated with cloud computing. One of which is the distribution of the data across several data providers which increases not only security, but also decreases the risk of data lock-in. The rest of this paper is structured as follows: starting by a motivation example and problems in 2, we follow by the proposed architecture in 3. In chapter 4 we describe how we deal with these general concerns. In the rest of the paper we show some related work and plans for future development.

2 Motivating Example

The presented example is for a financial consulting company which has several branches spread over European countries. The company is considering to take advantage of the economic benefits offered by maintaing parts of its data assets by a cloud service provider. Therefore, it carries on an extensive study on its data stocks to determine the appropriate resources to be transformed into the on-demand cloud computing model. The study showed that there is a need for an off-site backup storage, and a collaboration platform that facilitates file sharing and activity management. Some of the corporate data can be shared among employees and customers so it should not be protected by strong access measures. Other data contains personal information that can not be made public (e.g information about employees and customers), which requires protection from any unauthorized access.

Since the online storage is a competitive market, the company has to decide between a long list of service providers: Rackspace, GoGrid, Live Mesh, DropBox, Nirvanix, Amazon S3 storage service, etc.. According to [21] there are more than 100 vendors offering cloud storage. All of which have the same functionality.

The financial company -the customer- is of course interested in choosing the most reliable service, so it compares each offered solution independently which is, sufficient to say, a cumbersome task. This task includes studying: physical locations, legal provisions, security policies, and service level agreements (SLA).

2.1 Problems to Consider

1. **Security.** The most obvious concern is privacy which addresses the service providers responsibility to maintain and address security concerns for hosted infrastructure, and not sharing corporate data with a marketing firm or use the data in a way the client never intended. Network security as well as application security should be addressed.

2. **Service Availability.** Management of computing resources as a service by a single company implies the risk of single point of failure such as financial difficulties (bankruptcy), software or network failure, etc.. However, even if the vendor runs data centers in various geographic regions using different network providers, it may have the same software infrastructure. Therefore, a failure in the software in one center will affect all the other centers, hence affecting the service availability. In July 2008, for instance, Amazon storage service S3 was down for 8 hours because of a single bit error [20].
3. **Data lock-in.** In the context of cloud computing, it is a risk for a customer to become dependent on a provider for its services. Until today there are no standards for APIs or data import and export in cloud computing which limits data and applications portability between providers. The customer cannot seamlessly move the service to another provider if he becomes dissatisfied with the current provider (i.e. vendor increases his cost, goes out of business, or reduction in provided service quality).
4. **Missing QoS standardization.** After the selection of a service provider further market observation is also required. There is no guarantee, that the best vendor today will be the most reliable partner in the future. In the field of Web Services, this issue is addressed by formalizing service description and quality via Quality of Service (QoS) attributes. The problem of handling service level management in inter-domain scenarios is not entirely solved up to present. The scrutiny and the selection of services have to be performed independently which is both expensive and time consuming to handle this concern.
5. **Legislative Issue.** On the one hand there are laws and policies that allow government agencies freer access to data on a cloud than on private server. For example, in the USA the Stored Communication Act enables the FBI to access data without getting a warrant or the owner's consent. Furthermore, closed subpoenas may prohibit providers to inform their customers that data has been given to the government [21]. However, there are clear directives that demand public and private organizations to protect the security of their information systems concerning specific types of data assets (i.e. personal data) [19].

It is easily expected from the previous requirements that most companies will consider using cloud services as a security and legal hassle that is expensive just to avoid. Therefore, they may skip the idea altogether. The compromise is to neglect an in-depth provider assessment and out source it instead. In the next sections we describe a framework that semi-automizes most of the processes to spare the company the cumbersome efforts as well as to help realizing the cloud vision.

3 Architecture

The ground of our approach is to find a trade off between using the pay-per-use cloud services while ensuring the safety of the company's data. The goal is

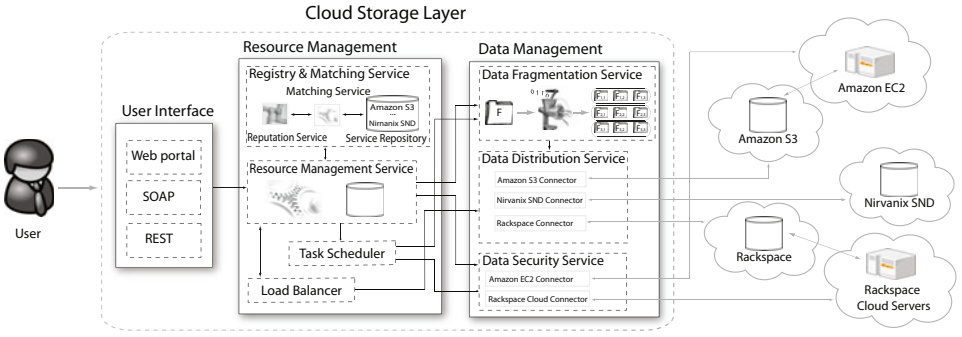

Fig. 1. Interaction of services in Cloud Storage Layer environment

to achieve such balance by the distribution of corporate data among multiple storage providers, automizing big part of the decision making process of selecting a cloud provider, and removing the auditing and administrating responsibility from the customer's side while keeping partial user control. The location of the data can be checked by the user whenever a legal check need to be made. The presented architecture (figure 1) is based on the following main components:

- **User Interface Module.** The interface presents a 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). User can upload, view modify or delete existing content. Further, user is presented with options to specify parameters regarding security or storage and transfer budget.
- **Resource Management Module.** This system component is responsible for intelligent deployment of data based on users requirements. The component is supported by:
  - Registry and matching service: assigns storage repositories based on users requirements. Monitors the performance of participating providers and ensures that they are meeting the agreed SLAs.
  - Resource management service: takes operational decisions regarding the content storage.
  - Task scheduler service: provides the ability to schedule the launch of operations at peak-off hours or after specified time intervals.
  - Load balancer component: the service helps to distribute workload across available storage resources taking into account unsers requirements.
- **Data Management Module.** The component handels data management on behalf of the resource management module and is mainly supported by:
  - Data fragmentation service: this component is responsible for permutation and striping of users content.
  - Data distribution service: spreads the fragmented data across multiple providers. Since each storage service is only accesible through a unique

API, the service utilizes storage "service-connectors", which provide an abstraction layer in communication to storage repositories.
- Security Service: enforces the security functionality based on user's requirements.

As mentioned above, the basic idea is not to depend solely on one storage provider but to spread the data across multiple providers using redundancy to tolerate possible failures. The approach is similar to the service-oriented version of RAID (Redundant Arrays of Inexpensive Disks) which manages sector redundancy dynamically across hard-drives. RAID 5, for instance, 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 use the same principle in cloud infrastructure by fostering the usage of erasure coding technics. This enables us to tolerate the loss of one ore more storage providers without suffering any loss of content [23], [14]. The system has a number of core components that contain the logic and management layers required to encapsulate the functionality of different storage providers. This is explained in details in the next section.

4 Design

Any application needs a model of storage, a model of computation and a model of communication. In this section we describe how we achieve the goal of a unified view on the data management system to the end-user.

4.1 User Interface

In general, there are two ways to interact with the storage management system: machine interpretable APIs (SOAP, REST) and a user focused web interface. First mentioned interfaces are under development at the time of writing this paper and are intended to deal with machine-based requests and requirement statements. Service interfaces are primarily aimed to facilitate the usage of our system for developers by dealing with more complex and frequent tasks.

The graphical user interface provides two major functionalities to an end-user: data administration and specification of requirements regarding the data storage. Administration relates to the creation of an account as well as uploading and viewing, modifying, or deleting the data. Further, users are presented with several options relating the data retention: security, geographic location, budget, availability and performance expectations in terms of quality of service assertions (e.g. bandwidth or response time). Some preferences can influence each other. For example, encryption causes a delay in availability of content as our approach foresees the encryption to be performed server-sided (see chapter 4.6). Hence, the requested data has to be decrypted prior its transmission to a client which increases storage costs. The reason is that extra cost arise for according computation which must be added to storage fees. Higher budget fosters replication of content to various vendors which increases performance and availability. The restriction of geographic hosting area can lead to higher storage and transfer costs

which decreases the availability of content in the case of low budget preferences. Due to higher costs of electricity, providers usually charge higher fees for hosting of data in Europe then in the USA (see table **??**). However, the dependences among the options are represented to the user visually as well as the according implications. Summarized, the user interface enables users to specify their requirements (regarding the placement and storage of user's data) manually in form of options:

- **budget-oriented** content deployment
- data placement based on **quality of service parameters** (i.e. availability, throughput, average response time)
- hosting of data based on users **security requirements**
- storage of data based on **geographical regions** of the user's choice. The restriction of data storage to specific geographic areas can be reasonable in the case of legal restrictions introduced in section 1.

4.2 Resource Management Service

This component tracks each user's actual deployment and is responsible for various housekeeping tasks:

1. **Tracking.** The service is equipped with a MySQL back-end data base to store crucial information needed for deploying and reassembling of users data. The data base tracks logistical details regarding the content storage. This includes information on each user's current deployment, according hash values, repositories used, replicas made of this content and credentials for utilized storage services. With this, the data base provides meta information on current content.
2. **Auditing.** The resource management service audits and tracks the performance of the participated providers and ensures, that all current deployments meet the relevant requirements specified by the user. In case of a breach of an agreed SLA or an unexpected change in pricing scheme, the resource management service avoids reading from the storage repository and removes the related shares. In this case the system reconstructs the original data from the redundant shares at other locations and reports to the reputation service. This way the management component represents a 'policy decision point' of the system.
3. **Scheduling.** The management component is also responsible for scheduling of non time-critical tasks. This includes the deployment of replicas and security functionality. Some providers (e.g. Amazon) offer discounts for large volumes and lower bandwidth rates for off-peak hours. In our approach we take advantage of these discounts to optimise the overall costs of data hosting. The management service delegates particular work-loads (e.g. file replication) to a system component named *task scheduler*.
4. **Budgeting.** The management component ensures that a user's budget has not been exceeded, e.g. by performing security related tasks such as data

encryption and integrity checks or by data transfer (i.e. up and download of content). In the event that current costs come close to a specified limit, our system starts removing replicated shares automatically.

4.3 Registry and Matching Service

At the present time, the capabilities of storage providers are created semi-automatically based on analysis of related SLAs that are usually written in a plain natural language[1]. Until now the claims stated in SLAs are to be translated and updated manually. Subsequently the formalized information is imported into a database of the system component named *service repository*. The data base tracks logistical details regarding the capabilities of storage services such as their actual pricing, SLA offered, and physical locations. With this, service repository represents a pool with available storage services. The selection of storage services for the data distribution occurs based on user preferences determined in the user interface.

After matching user requirements and provider capabilities, we use the reputation of the providers to produce the final list of potential providers to host parts of users data. A provider's reputation holds the details of his historical performance plus his ratings in the service registries and saved in a Reputation Object (introduced in our work in [4], [3], [2]). Simply, the object holds a profile of the behavior or performance of the provider in several contexts. These contexts are derived from the quality attribute used in the related SLAs. By reading this object, we know providers reputation concerning each performance parameter (e.g. has high response time, low price). In general, the number of storage repositories needed to ensure data striping depends on user's availability, security and performance requirements. Currently, our prototypical implementation supports three storage repositories: Amazons S3, Rackspace Cloud Files and Nirvanix SND.

4.4 Data Fragmentation Service

In compliance with [1] we mimic the data model of Amazon's S3 by the implementation of our fragmentation and distribution service. All data objects are stored in buckets. A bucket can not contain further buckets. Each bucket represents a flat namespace containing keys associated with objects. An object can be of an arbitrary size, up to 5 gigabytes. Objects must be uploaded entirely, as partial writes are not allowed in contradiction to partial reads.

The presented system establishes a set of n repositories for each data object of the user. These represent different cloud storage repositories. However, upon receiving a write request the system performs an initial permutation of the data object based on one of the predefined algorithms. The selection of a proper algorithm depends on the number of available storage repositories and the size of the source files. In general, if no security options are specified the particular

[1] Amazon Web Services SLA, http://aws.amazon.com/ec2-sla/

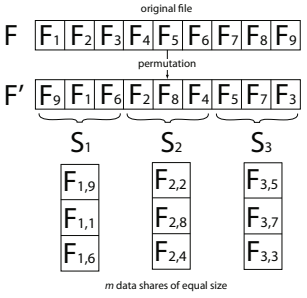

Fig. 2. Data fragmentation process

data shares (fragments) are stored in a plain text. Hence, the initial permutation is intended to prevent any party from being able to interpret the entrusted data. Otherwise a service provider would be able to read effortlessly the hosted data.

Following the permutation, the fragmentation service splits the incoming object into m data fragments of an equal size, whereby $m < n$ are variable parameters [17]. The system creates additional $(n - m)$ redundant shares, for a total of n shares. Redundant shares are the same size as data shares. Any subset of m shares is sufficient to reconstruct the original object. Hence, the usage of erasure-code technic allows us to recover original objects from any m fragments [12], [18]. The system makes sure that each share is sent to a different storage repository. Particular data shares contain also information about the original data object size and its hash value which is required for reassembling the data.

4.5 Data Distribution Service

Each storage service is integrated by the system by means of a service-connector. These provide an intermediate layer for the communication between resource management service (see section 4.2) and storage repositories hosted by storage vendors. This enables us to hide the complexity in dealing with access to unique APIs of each service provider. The basic connector functionality covers operations like creation, deletion or renaming of files and folders that are usually supported by every storage provider. Such a service-connector must be implemented for each storage service, as each cloud storage provider offers a unique interface to its repository. In some cases a higher overhead is needed to ensure the basic file management functionality. As we mentioned above, services differ in their usage. For example, Amazon's S3 lacks the feature of file renaming. Therefore, it requires the according S3 service-connector to delete and re-upload the according content by the execution of the rename-command on behalf of the data management component. When the service receives a *get* request, it fetches m shares and reassembles the data.

Further, the service is supported by a *load balancer* component, which is responsible for directing to the most appropriate repositories. Different load balancing and redirection policies are conceivable if parts of user's data are replicated to multiple providers. A user can be redirected to a random data

share or a physically closest service. Another thinkable approach is a minimal-cost aware redirection, which guides user to a cheapest source. Finally, users can be redirected to a replica that meet certain performance criteria (e.g response time or throughput).

4.6 Security Service

The initial level of security is provided by a primary permutation of users data objects and by its logical and physical segregation. These measures are taken no matter which security options are specified by the user. Some of cloud service providers offer computing resources (as a service) in addition to a mere storage infrastructure. For example, Amazon's Elastic Compute Cloud (EC2) provides a virtual computing environment that enables a user to run Linux- or Windows-based applications. In our approach we harness such computing services to ensure the security functionality in our system. In the following paragraph, we clarify the enforcement of security option *content encryption* for data shares placed on the S3 storage repository.

As mentioned above, Amazon enables customers to launch preconfigured instances. Whereby each instance appears to a user as a physical hardware. With this a user is capable of controlling nearly the entire software stack. Hence, we harness Amazon's image format to create our own virtual computer environment which contains software, encryption libraries and other configuration items. By placing the image in the EC2 environment we are able to launch, monitor and terminate any number of further instances. The management service conducts data to be secured to the according instances and determines the encryption algorithm. The encryption algorithm depends on the user's security requirements specified in the user interface. The usage and interaction with these instances is similar to the approach used by the implementation of service-connectors. The interaction is integrated by the system by means of security-connectors.

Security-connectors must be implemented for each computing platform independently, as each provider has different infrastructure and capabilities. For example, in the case of using Google's Megastore for data storage in conjunction with AppEngine for data processing, we would have to develop a dedicated Python application for data encryption. Consequently, we would have to implement a security-connector to communicate with the developed application.

Performance overhead. The specification of security requirements by a user influences the processing sequence described in the section above. The registry and matching component selects only vendors providing computing resources as a service along with storage services (e.g. Amazon, Rackspace). The data distribution process which includes the initial permutation and fragmentation remains unchanged. But following the data distribution an additional processing step arise: The management service assignes the task management component to encrypt the according data shares. The encryption assignment contains a priority

flag which depends on the user preferences specified in the user interface. The flag determines the period of time for encryption. For example, the setting *high* courses an immediate launch of the server-sided encryption instance (regardless of the current costs connected with its launch).

The settings *middle* and *low* are intended to support a cost optimised encryption. In this case encryption is performed during off-peak hours. For the encryption of data shares on S3 repository the system takes advantage of so-called spot instances. These enable users to bid for resources and thus control the balance of reliability versus monetary costs, availability and performance requirements.

5 Related Work

The main underlying idea of our approach is similar to provide RAID technique at the cloud storage level. In [7] authors introduce the HAIL 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-responce protocols for retrievability (POR) [16] and proofs of data possession (PDP) [16] and unifies these two approaches. In [13] 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 2.1. 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 the capabilities such as intelligent file placement based on users requirements, automatic replication, or security functionality.

The future of distributed computing has been a subject of interest for various researchers in the recent years. The authors in [10] 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 coud storage as a low-cost alternative to dedicated Content Delivery Networks (CNDs). In [8] Broberg and Buyya introduce a service for intelligent data placement and automatic replication which enables content creators to leverage the services of multiple cloud storage providers. However, this work does not address security and provider lock-in concerns which are mainly adressed in our approach. Further, in our work we don't aim to allocate resources from cloud providers to sell them to the customers. Our service acts as an abstraction layer between service vendors and service users automatising data placement processes. 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.

6 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 service provider. We demonstrated how our system deals with the mentioned concerns, introduced an architecture where a third party acts as an abstraction layer between users and cloud storage vendors with the services at their disposal. The three main components of the architecture are: User Interface (examines users expectations of geographic or security measures, and specifies QoS parameters), Resource Management Service (responsible for entire house keeping tasks and for intelligent deployment which includes the selection of cloud providers based on users expectations), and Data Distribution Service. The system stripes user's data across multiple providers. Thereby, it works by integrating with each storage provider via service-connectors thus providing an abstraction layer to hide the complexity and differences in the usage of storage services. Thus enabling users to avoid the risk of data lock-in and provide a low-level protection even without using far-reaching security functionality as none of the storage vendors is in an absolute possession of clients data.

However, full replication of users data is very costly but increases significantly the availability and the reliability of data storage. Therefore, in our approach we also consider the needs for budget-oriented users. We use erasure code techniques for striping data across multiple providers. This enables our system to tolerate one provider's failure by an added overhead cost of approximately 10% instead of 100% when full content replication is used. When using security options, we can say that the total costs increase with higher security requirements. It is up to each individual user to decide whether the additional costs caused by data encryption are justified.

7 Future Work

We plan to add to the implementation an additional feature; *service component* that is able to take from customers their formalized requirements and to translate them into formal electronic contracts (i.e. SLAs). This will enable full atomization of data hosting which primarily includes the identification of the suitable service providers based on user requirements and providers capabilities. At present, the maintenance of the database with providers capabilities is to be done manually as well as the specification of user requirements. Further, we are also planing to implement more service connectors and thus to integrate additional storage services.

Whilst the proposed system is still under development at present, we have to perform a comprehensive testing on its performance and reliability. This includes the predictability and sufficiency of response time and throughput as well as the validation of file consistency.

References

1. Abu-Libdeh, H., Princehouse, L., Weatherspoon, H.: Racs: A case for cloud storage diversity. In: SoCC 2010 (June 2010)
2. Alnemr, R., Meinel, C.: Enabling reputation interoperability through semantic technologies. In: ACM International Conference on Semantic Systems (2010)
3. Alnemr, R., Bross, J., Meinel, C.: Constructing a context-aware service-oriented reputation model using attention allocation points. In: Proceedings of the IEEE International Conference on Service Computing (2009)
4. Alnemr, R., Meinel, C.: Getting more from reputation systems: A context-aware reputation framework based on trust centers and agent lists. In: International Multi-Conference on Computing in the Global Information Technology (2008)
5. Amazon. Amazon ec2 service level agreement. online (2009)
6. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009, EECS Department, University of California, Berkeley (2009)
7. Bowers, K.D., Juels, A., Oprea, A.: Hail: A high-availability and integrity layer for cloud storage. In: CCS 2009 (November 2009)
8. Broberg, J., Buyya, R., Tari, Z.: Creating a 'Cloud storage' mashup for high performance, low cost content delivery. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 178–183. Springer, Heidelberg (2009)
9. Burt, J.: Future for cloud computing looks good, report says. online (2009)
10. Buyya, R., Yeo, C.S., Venugopal, S.: Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In: Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (August 2008)
11. Carr, N.: The Big Switch. Norton, New York (2008)
12. Chen, Y., Edler, J., Goldberg, A., Gottlieb, A., Sobti, S., Yianilos, P.: Prototype implementation of archival intermemory. In: IEEE ICDE (February 1996)
13. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with cfs. In: ACM SOSP (2001)
14. Dingledine, R., Freedman, M., Molnar, D.: The freehaven project: Distributed anonymous storage service. In: The Workshop on Design Issues in Anonymity and Unobservability (July 2000)
15. Keller, A., Ludwig, H.: The wsla framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management (2004)
16. Krawczyk, H.: LFSR-based hashing and authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)
17. Patterson, D., Gibson, G., Katz, R.: The case for raid: Redundant arrays of inexpensive disks. In: ACM SIGMOD (1988)
18. Rhea, S., Wells, C., Eaton, P., Geels, D., Zhao, B., Weatherspoon, H., Kubiatowicz, J.: Maintenance free global storage in oceanstore. In: IEEE Internet Computing (September 2001)
19. Smedinghoff, T.: Information Security: The Emerging Standard for Corporate Compliance. IT Governance Pub. (2008)

20. The Amazon S3 Team. Amazon s3 availability event: July 20, 2008. online (2008)
21. Velte, A.T., Velte, T.J., Elsenpeter, R.: Cloud Computing: A Practical Approach. McGraw Hill, New York (2009)
22. Venugopal, S., Chu, X., Buyya, R.: A negotiation mechanism for advance resource reservation using the alternate offers protocol. In: Proceedings of the 16th Int. Workshop on Quality of Service, IWQoS (June 2008)
23. Weatherspoon, H., Kubiatowicz, J.: Erasure coding vs. replication: A quantitative comparison. In: IPTPS (March 2002)