Towards Data Handling Requirements-aware Cloud Computing

Martin Henze, Marcel Großfengels, Maik Koprowski, Klaus Wehrle
Communication and Distributed Systems, RWTH Aachen University, Germany
Email: {henze,grossfengels,koprowski,wehrle}@comsys.rwth-aachen.de

***Abstract*—The adoption of the cloud computing paradigm is hindered by severe security and privacy concerns which arise when outsourcing sensitive data to the cloud. One important group are those concerns regarding the handling of data. On the one hand, users and companies have requirements how their data should be treated. On the other hand, lawmakers impose requirements and obligations for specific types of data. These requirements have to be addressed in order to enable the affected users and companies to utilize cloud computing.**

However, we observe that current cloud offers, especially in an intercloud setting, fail to meet these requirements. Users have no way to specify their requirements for data handling in the cloud and providers in the cloud stack – even if they were willing to meet these requirements – can thus not treat the data adequately. In this paper, we identify and discuss the challenges for enabling data handling requirements awareness in the (inter-)cloud. To this end, we show how to extend a data storage service, AppScale, and Cassandra to follow data handling requirements. Thus, we make an important step towards data handling requirements-aware cloud computing.

***Keywords*-Cloud Computing, Data Handling Requirements, Intercloud, Privacy**

I. Introduction

In recent years, the emergence of cloud computing with its vision of virtually unlimited, elastically scalable storage and processing resources has led to a variety of new, cloud-based services. The next iteration of this development is the vision of the intercloud paradigm [1], [2], where multiple clouds are used for providing a service in order to increase reliability as well as Quality of Service and at the same time reduce costs. Figure 1 illustrates a scenario where cloud resources distributed all over the world are utilized.

This distributed usage of resources is unnoticeable for the user. When moving sensitive data (e.g., customer records or sensed information) to the cloud, security and privacy concerns arise, which hinder the adoption of cloud-based services [3]–[7]. Data might be i) accessed by or handed over to third parties, ii) used for unintended purposes, iii) subject to data protection laws or contracts regarding customer data protection, and iv) not deleted once it is not needed anymore. Importantly, addressing these challenges becomes even more tremendous in an intercloud setting [8], [9].

In this paper, we discuss and address privacy concerns regarding the handling of data that arise when utilizing the cloud computing paradigm and especially the intercloud approach for outsourcing sensitive data. Users and lawmakers often impose data handling requirements when outsourcing

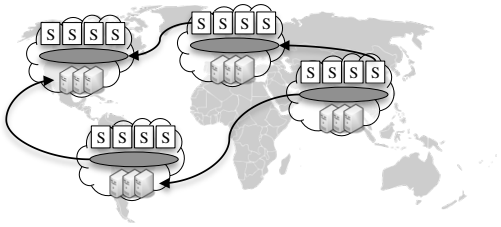

Figure 1. Intercloud Scenario

data, especially to the cloud. These requirements typically restrict, e.g., how long and where a specific piece of data might be stored. However, in current cloud offers it is impossible to meet these requirements adequately [9], as users cannot specify their requirements and cloud providers thus remain completely oblivious of these requirements. Even if they were willing to follow said requirements unconditionally, they are not able to do so. However, the ability to follow data handling requirements would allow cloud providers to enter new markets by addressing customers which want or have to adhere to data handling requirements.

Our contribution aims at making cloud computing data handling requirements-aware. First, we discuss data handling challenges when outsourcing data to the cloud. We then briefly present our vision of data handling aware cloud computing [9]. Essentially, we enrich data with data handling annotations before it is handed to the cloud. Based on this we discuss how to formalize data handling annotations using the PrimeLife Privacy Policy Language. Finally, we give insights into adapting a cloud stack consisting of AppScale and Cassandra to support data handling annotations and thus assist users *and* providers to meet data handling requirements.

II. Cloud Data Handling Challenges

Users and companies have certain requirements how their data should be handled. Companies, e.g., often want sensitive customer data to be stored in the jurisdiction of their headquarters. These preferences may either be intrinsic to the user or company, or driven by statutory regulations. The EU, e.g., demands that personal data of customers is only stored and processed within the EU or countries with comparable privacy laws (safe harbor principle). However, when outsourcing data to the cloud, users and companies essentially lose control over their data [4]–[6], [9].

In the following, we identify data handling challenges that have to be addressed technically when outsourcing data to

the cloud. Addressing these challenges allows to mitigate the anticipated loss of control over data. The two main challenges are location of storage and duration of storage.

A. Location of Storage

Storing data outside certain legislative boundaries (often even without noticing), raises severe concerns [4], [9]. One prominent reason for this is the *enforcement of data protection laws*. As already mentioned, the EU forbids the transfer of personal data to oversea jurisdictions with weaker privacy laws. The only exception is the safe harbor principle, which allows providers in countries with weaker privacy laws to voluntarily follow the EU privacy law and thus become eligible for receiving data. However, also *other legal requirements* besides data protection have an impact on the location of storage. In Germany, e.g., companies are not allowed to store any tax relevant data outside the EU.

Meeting these requirements with today's cloud services is virtually impossible. This essentially results from a lack of necessary information. In order to handle data compliant with these regulations, all involved entities need information where a specific data item is allowed to be stored and a way to communicate these restrictions.

B. Duration of Storage

For the duration of storage, we differentiate between maximum as well as minimum storage duration requirements. The *maximum duration of storage* specifies a point in time at which the data has to be deleted. This is driven by the perception of users who want their data to be deleted once it is not needed anymore. Recently, this approach has also been discussed as the "right to be forgotten" in the EU's regulation process [10]. The key challenge here inherently results from the desired redundancy (for reliability and performance) as well as the distributed nature of the cloud. If secure data erasure or physical destruction of the whole storage device are required, identifying and deleting all redundant copies of a data item in the cloud becomes a tremendous task. If the storage provider would know from the beginning when a data item has to be deleted, it could leverage this information in its storage allocation decision. Contrary, the *minimum duration of storage* specifies a point in time before which the data must not be deleted. Typical use cases origin from the accounting and tax domain where in many countries retention periods of documents have to be met. Thus, the corresponding data has to be stored persistently at least until the specified date. Again, in order to guarantee a minimum duration of storage, it is crucial that the storage provider knows in advance when the data should be deleted earliest.

III. Data Handling Requirements-aware Cloud

We showed that lawmakers on the one hand and users as well as companies on the other hand impose requirements how data should be handled when it is transferred to and stored in the cloud. Now, we discuss first steps towards meeting these data handling requirements in a cloud implementation. As we already proposed in earlier work [9], we use cross-layer data handling annotations in order to signal data handling requirements across the (inter-)cloud stack.

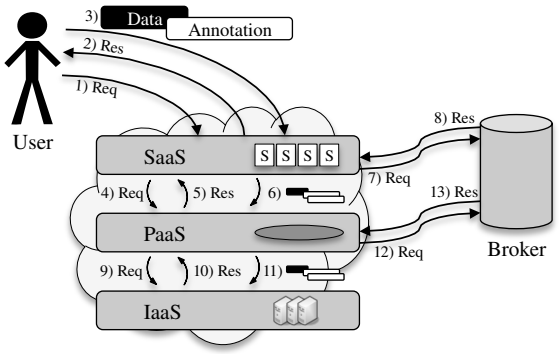

Figure 2. Data Handling Requirements-aware Cloud Stack

Figure 2 gives an overview over our envisioned architecture, which we discuss in the following example. Consider a user who uses a cloud service. Before she hands over her data to the cloud service, she creates a *data annotation* containing formalized data handling obligations. These obligations specify data handling requirements which have to be guaranteed by the cloud service (e.g., "delete after 30 days"). To prevent sending data to a cloud service that is not able to fulfill the stated obligations, at first only the data annotation is sent to the cloud service. Subsequently, the cloud service matches the data handling obligations against its data handling policies (i.e., data handling obligations it is able and willing to follow). In case of a positive match, the cloud service *signs the data annotation* and sends it back to the user. The user now has a proof that the cloud service will follow the stated obligations and can safely transfer the data together with the data annotation to the cloud service.

Now, the provider of the cloud service is responsible for following the data handling obligations. Additionally, it might itself impose additional obligations. A prominent example are data protection requirements in the EU, which require certain data to not leave the legislative boundaries of the EU. In our example, the cloud service adds another annotation ("do not store outside the EU"). Especially in an intercloud scenario, a cloud service often uses more than one PaaS offer to realize its service. Thus, it has to select a PaaS provider which is able and willing to follow the combined set of data handling obligations. We propose to utilize already existing cloud brokers [2] which determine the best provider with respect to QoS, SLAs, and pricing. These brokers have to be extended to also support matching of data handling obligations against policies. As soon as the SaaS finds a fitting PaaS offer, it will perform the previously described annotation matching handshake with this PaaS provider. This way, the SaaS provider also obtains a proof that the PaaS pledged to follow the annotated obligations.

Similarly, the PaaS provider is now responsible for the enforcement of the data handling obligations. Again, the PaaS provider might realize its service utilizing multiple IaaS providers, e.g., to decrease latency or costs. In order to find a IaaS provider that is capable of fulfilling the stated data handling obligations, it uses the services of a broker (as described above). Finally, the annotation matching handshake is performed and the PaaS provider receives a proof from the IaaS provider.

The IaaS maps data onto real hardware and is ultimately responsible for realizing the stated data handling obligations. Thus, it has to consider the location of storage when distributing data on hardware nodes. Additionally, it has to take measures to meet the duration of storage requirements.

IV. Formalizing Data Handling Obligations

In order to formalize data handling obligations, we leverage and extend the obligations concept of the PrimeLife Privacy Policy Language (PPL) [11], [12]. PPL is designed as an extension of the OASIS standardized eXtensible Access Control Markup Language (XACML) [13] to provide both access control and usage control functionalities in a data provider/data receiver scenario. Similarly to our approach, the actual data transfer should only happen after both sides agreed upon the handling of the data by comparing their respective data handling *preferences* (data owner) and *policies* (data receiver, e.g., service provider). Within the language schema, obligation elements enable the data provider to specify in her preferences how a piece of data is required to be handled by the receiver, e.g., "delete data within 7 days". The data receiver can in turn propose obligations that it is willing to adhere to in its policy. In the context of matching preferences against policies, a successful match requires the obligations of the policies to be less or equally permissive than those of the preferences.

Obligations in PPL are defined as a set of *triggers* and an *action* element in a context of "`do` {*action*} `when` {*trigger*}". For example, we can define the obligation "`do` {delete data} `when` {7 days passed}". One obligation can also have multiple triggers. Triggers are realized by capturing and storing contextual information (e.g., time) of the system. The interpretation and execution of actions however is entirely up to the implementation of the policy enforcement. Currently, PPL provides only a short, non-exhaustive list of predefined actions and triggers which can be combined to create obligation elements.

In the following, we discuss why PPL is a sensible choice for realizing data handling annotations in a cloud setting. For policy creation, evaluation and matching, PPL separates the handling of the original XACML access control part from the new obligation part, making the latter essentially an independent component of the language schema. This allows us to consider the PPL schema of obligations as a basis for the expression of our data handling annotations. PPL policies/preferences have a simple, human-readable syntax, which allows developers and even private users to utilize our approach. In terms of obligations and authorizations, PPL is easily extensible and can thus support a great variety of possible data handling use cases. PPL can be used by both data providers and data receivers to express their data handling preferences respectively policies in the same way, allowing for automated matching without prior translation of policies. This also facilitates the automated creation of merged policies, which contain a pair of successfully matched preferences and policies.

Now, we describe how we utilize the obligation concept of PPL to address the challenges identified in Section II. The trigger-based concept of obligations allows us to realize the *maximum duration of storage* and thus a guaranteed deletion of data, as it is, e.g., required by the right to be forgotten. This approach is also easily adaptable to an obligation defining a *minimum storage duration*. Although there is no predefined obligation regarding the *storage location*, PPL's obligation concept was designed to be easy extensible to include new obligations. We thus propose to extend the PPL obligation schema by defining an obligation element independent of the trigger/action-model, which specifies restrictions for the location of stored data. As the specification of location is possibly ambiguous and matching requires identical identifiers, this feature will be backed by a database providing alternative identifiers for each possible location. The location "Europe", e.g., contains the range of applicable IP addresses as well as countries, which can be accessed during the matching process.

V. Adapting a Cloud Stack

To validate and evaluate our proposed architecture, we are adapting a cloud stack in order to enable it to interpret the formalized data handling obligations and react accordingly. As underlying scenario, we chose a cloud service offering file storage and synchronization (similar to Dropbox). On the bottom, we built upon a Cassandra [14] cluster. As PaaS solution, we use AppScale [15] which utilizes Cassandra as storage provider [16]. The top of our exemplary cloud stack consists of a small, self-written program that runs on top of AppScale and realizes the actual file storage cloud service.

The main challenges when addressing the above data handling concerns involve the placement of data onto actual hardware. We therefore adapt those entities in Cassandra which handle the deployment and replication of data [14] to include data handling annotations.

One of these entities is the *snitch*, which essentially maps nodes to data centers as well as racks, and thus defines how nodes are grouped together in the overall network topology [17]. Cassandra itself provides several out-of-the-box snitches, some of them coming close to our needs regarding the deployment of data in specific locations. The snitches designed for Amazon EC2, e.g., map a node to a

whole region (e.g., South America). Because these mappings are determined and only changeable by Amazon, the existing snitches do not provide the flexibility necessary for our scenario. Our approach is a custom snitch which maps nodes to racks and data centers on a country level. The custom snitch needs a database with the mapping of IP to country in order to address the challenges regarding the *location of storage*. To address the *duration of storage* challenges, the provider can add secure data erasure cycles or planned disposal of the nodes' hard disks to this database.

The other two data distribution components of Cassandra we have to adapt are the *partitioner* and *replication strategy* [17]. The system is designed to partition the data first by assigning keys and computing tokens (the hash) of the keys and then creating copies of the data and distribute them among several nodes for reliability and fault tolerance. Again, Cassandra provides out-of-the-box partitioners and replication strategies. For *partitioning*, we can choose between an even distribution of data on the cluster (via consistent hashing) and an (lexically) ordered distribution. However, both are not feasible for addressing data handling challenges. As even distribution yields more advantages over an ordered distribution [17], we modify the even distribution approach. Our custom partitioner evenly distributes the data among those nodes of the cluster which are permitted by the data handling obligations. For this, our partitioner generates tokens which correspond to the data handling annotation by including the derived country code as a part of the token to determine the distribution. For the replication we adapt an existing *replication strategy* of Cassandra. Our adapted replication strategy randomly picks a certain number of nodes which suffice the given annotation and do not already store a replica of the data.

AppScale, which runs on top of Cassandra as our PaaS solution, has to be adapted as well. The major task of AppScale is to pass the annotations to Cassandra. For this, we extend AppScale's datastore API [15], [16] by adapting the `put()` command to pass down the annotation together with the data. Additionally, we change the AppScale configuration files for the datastores [16] to include our new storage entities. Similarly, our custom data storage cloud service asks the user for data handling annotations. It then compares the annotated obligations against its own policies. If they match, it receives the data, bundles it with the annotation, and hands it over to our adapted AppScale instance.

With our custom data storage service and these adaptations to Cassandra and AppScale we are able to built an early, functional prototype of a data handling-aware cloud stack. This prototype will then allow us to evaluate functionality, performance, and overhead of our approach.

VI. Conclusion and Outlook

Based on the identified data handling challenges, we discussed first steps towards implementing a data-handling aware cloud stack based on data handling annotations. We showed how we can utilize the PrimeLife Privacy Policy Language in order to formalize data handling obligations and policies. Based on this, we discussed how to adapt Cassandra and AppScale in order to interpret and effectuate these annotated data handling obligations and policies.

In the future, we plan to extensively evaluate our prototype cloud stack in order to justify functionality and overhead. Preliminary results indicate a modest performance overhead of our solution. Furthermore, we plan to further extend our approach. Here, we will especially focus on security as well as enforcing and tracing data handling obligations. Regarding security, we aim at cryptographically securing the annotation handshake process as well as binding annotations to data, e.g., using sticky policies [18]. Our main approach towards enforcement and tracking data handling is to create more transparency in the involved processes in the cloud.

Acknowledgment

This work has been funded by the Excellence Initiative of the German federal and state governments.

References

[1] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability," in *Proc. ICIW*, 2009.

[2] N. Grozev and R. Buyya, "Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey," *Software Pract Exper*, 2012.

[3] R. Hummen, M. Henze, D. Catrein, and K. Wehrle, "A Cloud Design for User-controlled Storage and Processing of Sensor Data," in *Proc. IEEE CloudCom*, 2012.

[4] S. Pearson and A. Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," in *Proc. IEEE CloudCom*, 2010.

[5] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud Data Protection for the Masses," *Computer*, vol. 45, no. 1, 2012.

[6] H. Takabi, J. Joshi, and G. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *IEEE Security Privacy*, vol. 8, no. 6, 2010.

[7] M. Henze, R. Hummen, R. Matzutt, D. Catrein, and K. Wehrle, "Maintaining User Control While Storing and Processing Sensor Data in the Cloud," *IJGHPC*, vol. 5, no. 4, 2013, in press.

[8] D. Bernstein and D. Vij, "Intercloud Security Considerations," in *Proc. IEEE CloudCom*, 2010.

[9] M. Henze, R. Hummen, and K. Wehrle, "The Cloud Needs Cross-Layer Data Handling Annotations," in *Proc. IEEE SPW*, 2013.

[10] J. Rosen, "The Right to Be Forgotten," *Stan. L. Rev. Online*, vol. 64, no. 88, 2012.

[11] C. A. Ardagna *et al.*, "PrimeLife Policy Language," W3C Workshop on Access Control Application Scenarios, 2009.

[12] S. Trabelsi, A. Njeh, L. Bussard, and G. Neven, "The PPL Engine: A Symmetric Architecture for Privacy Policy Handling," W3C Workshop on Privacy and Data Usage Control, 2010.

[13] "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, 2013.

[14] A. Lakshman and P. Malik, "Cassandra – A Decentralized Structured Storage System," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, 2010.

[15] N. Chohan *et al.*, "AppScale: Scalable and Open AppEngine Application Development and Deployment," in *Proc. CloudComp*, 2009.

[16] C. Bunch *et al.*, "An Evaluation of Distributed Datastores Using the AppScale Cloud Platform," in *Proc. IEEE CLOUD*, 2010.

[17] "Apache Cassandra™ 1.2 Documentation," DataStax, Inc., 2013. [Online]. Available: http://www.datastax.com/docs/1.2/

[18] S. Pearson and M. Mont, "Sticky Policies: An Approach for Managing Privacy across Multiple Parties," *Computer*, vol. 44, no. 9, 2011.