Institute of Architecture of Application Systems

A Taxonomy for Cloud Data Hosting Solutions

Steve Strauch, Oliver Kopp, Frank Leymann, Tobias Unger

Institute of Architecture of Application Systems, University of Stuttgart, Germany, lastname@iaas.uni-stuttgart.de
gridsolut GmbH + Co. KG, Stuttgart, Germany, unger@gridsolut.de

BibTeX:

```
@inproceedings{Strauch2011,
  author    = {Steve Strauch and Oliver Kopp and Frank Leymann and
               Tobias Unger},
  title     = {A Taxonomy for Cloud Data Hosting Solutions},
  booktitle = {Proceedings of the IEEE International Conference on Cloud
               and Green Computing, CGC 2011,
               12-14 December 2011, Sydney, Australia},
  year      = {2011},
  pages     = {577-584},
  doi       = {10.1109/DASC.2011.106},
  publisher = {IEEE Computer Society}
}
```


Universität Stuttgart
Germany

2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing

A Taxonomy for Cloud Data Hosting Solutions

Steve Strauch[◕], Oliver Kopp[◕], Frank Leymann[◕], Tobias Unger[◕,†]
[◕]*Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany, lastname@iaas.uni-stuttgart.de*
[†]*gridsolut GmbH + Co. KG, Stuttgart, Germany, unger@gridsolut.de*

***Abstract*—Cloud computing allows reducing capital expenditure by using resources on demand. We investigate how to build a database layer in the Cloud and present pure and hybrid Cloud data hosting solutions. The solutions are organized in a taxonomy. The properties used for organization are: application layer, deployment model, location, service model, data store type, and compatibility. Using the taxonomy, existing Cloud data hosting solutions are categorized.**

***Keywords*-cloud data hosting solution; taxonomy; distributed application architecture; database layer; cloud computing**

I. Introduction

Cloud computing has started to be applied in industry. This is substantially based on the advantage of reducing capital expenditure (CAPEX) and transforming it into operational costs [1]. "Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e. g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [2]. Two main Cloud computing characteristics are on-demand self-service and rapid elasticity. On-demand self-service allows that a Cloud user can unilaterally provision computing capabilities (e. g., server time) as required automatically without needed human interaction with the corresponding service provider. Rapid elasticity denotes that computing capabilities can rapidly be provisioned and deprovisioned according to the current demand. To deploy an application in the Cloud, there are four deployment models available: *Private Cloud*, *Public Cloud*, *Community Cloud*, and *Hybrid Cloud* [2].

Applications are typically built using a three layer architecture model consisting of a presentation layer, a business logic layer, and a data layer [3]. The presentation layer treats the interaction with the user. The business logic is realized within the business layer. The data layer is responsible for data storage and is in turn subdivided into data access layer (DAL) and database layer (DBL). The DAL is an abstraction layer encapsulating the data access functionality. The DBL is responsible for data persistence and data manipulation. The subdivision of the data layer finally leads to a four layer application architecture. Figure 1 presents this architecture. Until today, migrating and hosting an application in the Cloud has been limited to full virtualization: the application as a whole is put into a virtual machine image.

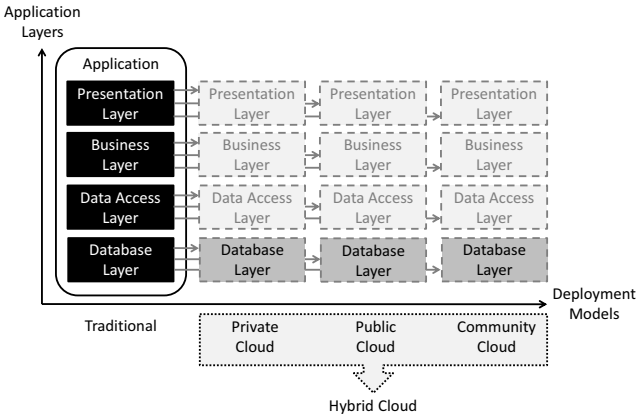

Figure 1. Overview of Cloud Deployment Models and Application Layers

Current available virtual image capture products are PlateSpin Migrate [4], VMware vCenter Converter [5], and Microsoft Visual Server Migration Toolkit [6], for instance. Recently, new types of Cloud services have been introduced. For example, RunMyProcess enables business process modeling and execution in the Cloud [7]. Amazon S3 enables storage in the Cloud [8].

Figure 1 shows all potential possibilities for the distribution of an application consisting of four layers. The traditional application not using any Cloud technology is shown on the left. It is neither distributed nor replicated or running on a computer cluster. The acceptance and application of Cloud computing in industry is still limited to hosting a Private Cloud infrastructure in-house: The infrastructure is shared only with the different divisions of the company. Data privacy, data security, data compliance, and quality of services (QoS) such as availability are the main concerns hampering the application of Cloud computing in the area of enterprise applications. Currently, there is little support for decision making for data hosting in the Cloud. The taxonomy presented in this paper supports decision making for migrating or building a database layer in the Cloud.

The contribution of this paper is a new taxonomy for *Cloud data hosting solutions*. The term *Cloud data hosting solution* denotes the choice among the concrete deployment model, service model, and the implied capabilities such as a centralized or distributed data store. The taxonomy is

978-0-7695-4612-4/11 $26.00 © 2011 IEEE
DOI 10.1109/DASC.2011.106

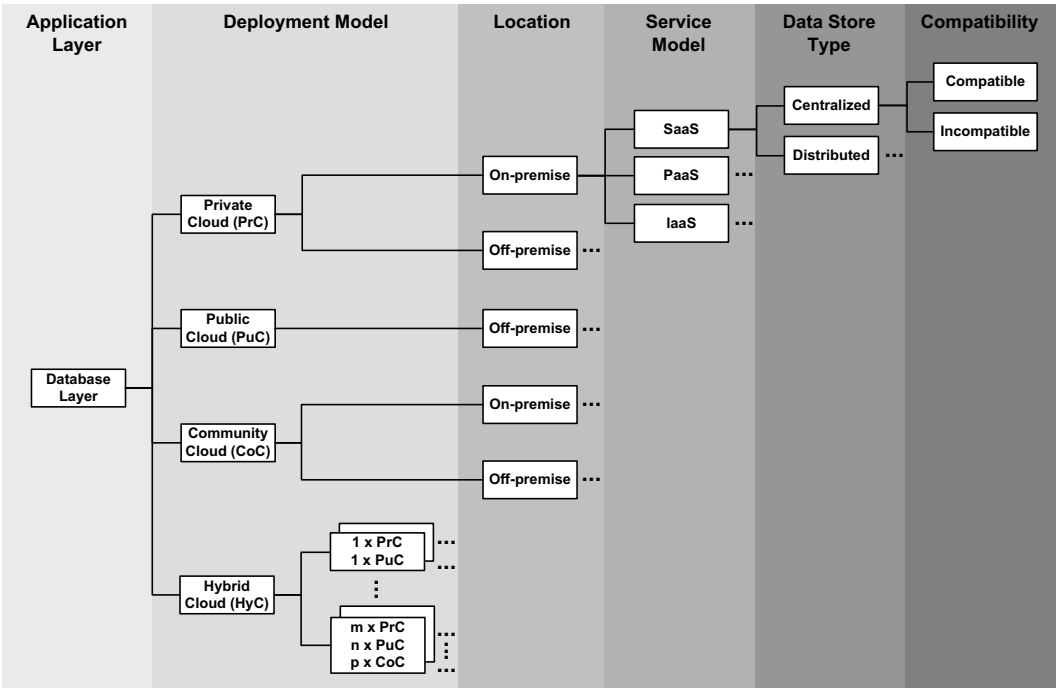

Figure 2. Taxonomy for Cloud Data Hosting Solutions

presented in Section II. Selected existing approaches are classified using the taxonomy in Section III. The taxonomy and its limitations are discussed in Section IV. Related work in the field of classification of Cloud data hosting solutions is presented in Section V. Finally, Section VI concludes and provides an outlook on future work.

II. Taxonomy for Cloud Data Hosting Solutions

The taxonomy for Cloud data hosting solutions is presented in Figure 2. The six distinguishing properties are:

- *Application Layer* (1 option)
- *Deployment Model* (4 options)
- *Location* (2 options)
- *Service Model* (3 options)
- *Data Store Type* (2 options)
- *Compatibility* (2 options)

The properties have been derived from our experience gained from industry and research projects. They have been proven to be essential for choosing a concrete Cloud data store. There are more properties regarding technical aspects such as virtualization, which have not been regarded as supportive for decision making.

Without regarding the Hybrid Cloud, there are $1 \cdot 3 \cdot 2 \cdot 3 \cdot 2 \cdot 2 = 72$ possibilities. A Public Cloud, however, is per definition always hosted off-premise. Thus, for the Public Cloud properties, $3 \cdot 2 \cdot 2 = 12$ of the 72 entries are invalid. This leads to a total of 60 pure Cloud data hosting solutions. By using "pure" we denote that no Hybrid Cloud deployment model is used. A Hybrid Cloud allows for arbitrary combinations of pure deployment models as long as at least two distinct pure deployment models are used. As a result, the number of Cloud data hosting solutions is infinite. The properties of each pure deployment model used in a Hybrid Cloud setting are the same as shown for the respective pure deployment model in the taxonomy (denoted by "..." in Figure 2).

In the following, the properties are described in the order they appear in the taxonomy. First, the application layer is presented. It is followed by the presentation of the Cloud-related properties. Finally, properties concerning functionality of the database layer are tackled. An alternative solution is to switch Cloud-related properties with the functionality. We opted for the first solution to ease selection of Cloud offerings based on the corresponding Cloud-related properties.

A. Application Layer

This taxonomy treats the database layer only. To indicate that the taxonomy can be extended to other application layers, we explicitly included the property application layer.

B. Deployment Model

First of all, the option of the deployment model has to be taken. It answers the questions who is administrating the Cloud infrastructure for whom. NIST distinguishes four deployment models [2]:

- *Private Cloud* – The Cloud infrastructure is run for a single organization. It may be administrated by the company itself and may exist on-premise or off-premise.
- *Public Cloud* – The Cloud infrastructure is provided to the general public or a large industry group and is owned by an organization selling Cloud services. It exists off-premise only.
- *Community Cloud* – The Cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns. It may be maintained by the organizations or a third party. Moreover, it may exist on-premise or off-premise.
- *Hybrid Cloud* – The Cloud infrastructure is a composition of at least two or more Private, Public, and Community Clouds that are still independent but integrated to a certain extend by using standardized or proprietary technology enabling data and application portability between them. It may exist on-premise, off-premise, or both on-premise and off-premise.

C. Location

The location is directly related to the deployment model. It states whether the Cloud infrastructure is operated within the company using it (*on-premise*) or outside the company (*off-premise*).

D. Service Model

With respect to the capabilities provided to the Cloud service consumer the following three service models are distinguished [2]:

- *Cloud Software as a Service (SaaS)* – The capability provided to the consumer is to utilize the provider's applications running on a Cloud infrastructure. The applications are accessible through a thin client interface such as a Web browser. With the possible exception of limited user-specific application configuration settings, the consumer does not manage or control the underlying Cloud infrastructure or even individual application capabilities.
- *Cloud Platform as a Service (PaaS)* – The capability provided to the consumer is to deploy consumer-created or acquired applications onto the Cloud infrastructure. The consumer does not manage or control the underlying Cloud infrastructure, but has control over the deployed applications and possibly application hosting environment configurations.
- *Cloud Infrastructure as a Service (IaaS)* – The capability provided to the consumer is to provision fundamental computing resources such as processing, storage, or networks where the consumer is able to deploy and run arbitrary software. The software can include operating systems and applications. The consumer does not manage or control the underlying Cloud infrastructure, but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components.

E. Data Store Type

The *data store type* is used for distinguishing *centralized data stores* (*centrDS*) and *distributed data stores* (*distrDS*). We use the term "data store" instead of "database" to include the data stores and Cloud data services emerged in the field of NoSQL [9].

F. Compatibility

For migration purposes compatibility of the existing traditional database layer and the Cloud data hosting solution is important. Thus, the taxonomy has to be further refined into compatible (*comp*) and incompatible (*incomp*) infrastructures. In case a new application is built, the option compatible is the only available option.

G. Unique Identifiers for Cloud Data Hosting Solutions

For abbreviating a concrete pure Cloud data hosting solution, we use unique identifiers derived from the regular expression shown in Listing 1. We concatenated abbreviations for valid particular values and use abbreviations for each property. For instance, "centrDS" stands for "centralized data store". The delimiter "–" is escaped as "–" itself is a meta character.

```
unique identifier = DBL \-
  ((PrC \- (onP|offP)) |
   (PuC \- offP) |
   (CoC \- (onP|offP)) \-
  (IaaS|PaaS|SaaS) \-
  (centrDS|distrDS) \-
  (comp|incomp)
```

Listing 1. Regular Expression for Unique Identifiers of Pure Cloud Data Hosting Solutions

Hybrid Cloud Data Hosting Solutions (HCDHS) consist of combinations of pure deployment models. Thus, the unique identifier for a HCDHS begins with `DBL-HyC-` and states which pure deployment models are used with their location, service model, data store type, and compatibility properties. There may be more than one deployment model with the same properties used. For instance when using similar products of different vendors. The number of deployment models used is given by a number. As a result, a unique identifier for a HCDHS is as follows: `DBL-HyC-#1*(PrC-offP-IaaS-centrDS-comp)-#1*(PuC-offP-PaaS-centrDS-comp)`. This denotes a combination of Amazon VPC with Amazon RDS, which are classified in the next section.

Although the value of the *application layer* property is the same for every Cloud data hosting solution, we integrated it in the regular expression for the sake of extensibility to other application layers.

III. Classfication of Existing Approaches

In this section, we evaluate the offerings of Amazon Inc., Eucalyptus Inc., Google Inc., Microsoft, MongoLab, Rackspace, SalesForce.com, and SuccessBricks Inc. For evaluating the compatibility property, we assume a concrete existing database layer. It is hosted on-premise using the traditional deployment model for a single, non-clustered, relational MySQL [10] database, version 5.1.

Table I presents a classification of existing Cloud data store offerings.

Amazon Elastic Block Storage (Amazon EBS [11]) by Amazon Inc. offers to create storage volumes behaving like unformatted block devices, where a file system can be created and used. Amazon EBS is offered on a Public Cloud (PuC-offP) as IaaS. We regard Amazon EBS as distributed data store (distrDS) as EBS internally replicates the storage volume within the same availability zone. As Amazon EBS does not offer a database system, it is incompatible (incomp).

Amazon Elastic Compute Cloud (Amazon EC2 [12]) by Amazon Inc. offers hosting of virtual images. The format of the used images is called "Amazon Machine Image" (AMI). Amazon offers pre-configured AMIs. Amazon EC2 is offered on a Public Cloud (PuC-offP) as IaaS. The concrete classification depends on the chosen AMI image. In case a single image with MySQL 5.1 is chosen, the additional properties are centralized data store (centrDS) and compatible (comp). In case a single image with DB2 Express 9.5 is chosen, the additional properties are centralized data store (centrDS) and incompatible (incomp). Amazon offers the possibility to configure a cluster of instances to increase performance. For instance, the usage of images offering nodes of a MySQL 5.1 cluster leads to the two properties distributed database (distrDB) and compatible (comp). We denote the dependency on the concrete choice as "DBL-PuC-offP-IaaS-*-*" in Table I.

Amazon Relational Database Service (Amazon RDS [13]) by Amazon Inc. offers a relational database in the Cloud. The functionality is compatible with MySQL 5.1 and Oracle 11g. It is offered on a Public Cloud (PuC-offP) as PaaS. Amazon RDS offers the deployment of an own database schema. We regard the schema as the application in the NIST definition. Thus, Amazon RDS is PaaS. The concrete classification depends on the chosen database instance type and its configuration. Amazon RDS allows to choose among MySQL and Oracle instance types and to configure read replicas and multi availability zone deployment. In case no multi availability zone deployment is chosen, the database is not distributed. Choosing MySQL 5.1 with no read replicas and no multi availability zone deployment leads to a centralized (centrDS) and compatible (comp) data store. Choosing Oracle 11g database instance type with read replicas and multi availability zones leads to a distributed (distrDS) and incompatible (incomp) data store. We denote the dependency on the concrete choice as "DBL-PuC-offP-PaaS-*-*" in Table I.

Amazon Simple Storage Service (Amazon S3 [8]) by Amazon Inc. is a replicated object store. Objects can be stored and retrieved using a key. Amazon S3 is offered on a Public Cloud (PuC-offP) as SaaS. The customer may configure the Reduced Redundancy Storage (RRS) option and authentication mechanisms. The user has no further control over the infrastructure. As the objects are always replicated, we classify Amazon S3 as distributed data store (distrDS). The concept of key-value storage is different from MySQL leading to an incompatibility (incomp).

Amazon SimpleDB [14] is a non-relational data store offered by Amazon Inc. It is offered on a Public Cloud (PuC-offP). Amazon SimpleDB offers no configuration option, thus it falls in the category SaaS. It is a distributed data store (distrDS) as it always creates geographically distributed copies of the data in order to provide high availability. It is incompatible (incomp) with MySQL.

Amazon Virtual Private Cloud (Amazon VPC [15]) by Amazon Inc. offers hosting of virtual images (PaaS) in a Private Cloud (PrC) hosted off-premise (offP) by Amazon Inc. As Amazon VPC is also based on AMIs, the data store properties are dependent on the database and the image(s) chosen. We denote that with "DBL-PrC-offP-PaaS-*-*" in Table I.

ClearDB [16] by SuccessBricks Inc. offers a relational database in the Cloud. It is offered on a Public Cloud (PuC-offP). ClearDB provides an own language for triggers (incomp), but is otherwise compatible with MySQL. The database schema is deployed on the ClearDB instance. ClearDB automatically replicates the data internally (distrDS).

Cloud Files [17] by Rackspace is a replicated file store and content delivery network (CDN). Cloud Files is offered on a Public Cloud (PuC-offP) as SaaS. Files up to five gigabyte can be uploaded and managed using the online control panel or RESTful API. Uploaded files can be configured to be distributed to several endpoints across the world. The data is organized in containers. The user can configure to use private containers or public containers. When using private containers a secure and encrypted connection between Cloud Files and the users application is established. Configuring containers as public implies, that a URL is provided for every file in the corresponding container for public sharing. The user has no further control over the infrastructure. It is a distributed data store (distrDS) as automatic replication of three full copies of the data across multiple computers in multiple zones is provided. The concept of file storage and CDN is different from MySQL leading to an incompatibility (incomp).

Database.com [18] by SalesForce.com offers a relational database in the Cloud. It is offered on a Public Cloud (PuC-offP). Database.com offers Salesforce Object Query

Table I
CLASSIFICATION

Cloud Data Store	Configuration	Classification
Amazon EBS [11]	–	DBL-PuC-offP-IaaS-distrDS-incomp
Amazon EC2 [12]	AMI running MySQL 5.1 relational database on OpenSolaris	DBL-PuC-offP-IaaS-centrDS-comp
	AMI running IBM DB2 Express 9.5 on Linux	DBL-PuC-offP-IaaS-centrDS-incomp
	...	DBL-PuC-offP-IaaS-⋆-⋆
Amazon RDS [13]	MySQL 5.1 DB instance, no read replicas, no multi availability zone deployment	DBL-PuC-offP-PaaS-centrDS-comp
	Oracle Database 11g DB instance, read replicas and multi availability zone deployment	DBL-PuC-offP-PaaS-distrDS-incomp
	...	DBL-PuC-offP-PaaS-⋆-⋆
Amazon S3 [8]	–	DBL-PuC-offP-SaaS-distrDS-incomp
Amazon SimpleDB [14]	–	DBL-PuC-offP-SaaS-distrDS-incomp
Amazon VPC [15]	AMI running MySQL 5.1 relational database on OpenSolaris	DBL-PrC-offP-IaaS-centrDS-comp
	AMI running IBM DB2 Express 9.5 on Linux	DBL-PrC-offP-IaaS-centrDS-incomp
	...	DBL-PrC-offP-IaaS-⋆-⋆
ClearDB [16]	–	DBL-PuC-offP-PaaS-distrDS-incomp
Cloud Files [17]	–	DBL-PuC-offP-SaaS-distrDS-incomp
Database.com [18]	–	DBL-PuC-offP-PaaS-distrDS-incomp
Eucalyptus [19]	EMI running MySQL 5.1 relational database on OpenSolaris hosted solely for a single organization operated by the organization itself	DBL-PrC-onP-IaaS-centrDS-comp
	EMI running IBM DB2 Express 9.5 on Linux hosted solely for a single organization operated by the organization itself	DBL-PrC-onP-IaaS-centrDS-incomp
	...	DBL-⋆-⋆-IaaS-⋆-⋆
Google App Engine Blobstore [20]	–	DBL-PuC-offP-SaaS-distrDS-incomp
Google App Engine Datastore [21]	–	DBL-PuC-offP-SaaS-distrDS-incomp
Microsoft Live SkyDrive [22]	–	DBL-PuC-offP-SaaS-distrDS-incomp
Microsoft SQL Azure [23]	–	DBL-PuC-offP-PaaS-centrDS-incomp
MongoLab [24]	–	DBL-PuC-offP-PaaS-distrDS-incomp

Language (SOQL) which is a query-only language not fully compatible with SQL (incomp) [25]. The database schema and triggers are deployed on database.com. Thus, it is a PaaS. Database.com automatically replicates the data internally (distrDS).

Eucalyptus [19] by Eucalyptus Inc. offers a software solution for hosting virtual images (IaaS). The format of the used images is called "Eucalyptus Machine Image" (EMI). The EMI may be freely chosen. A single MySQL 5.1 image leads to a centralized (centrDS) and compatible (comp) data store. A single IBM DB2 Express 9.5 image leads to centralized (centrDS) and incompatible (incomp) data store. The infrastructure may hosted on-premise (onP) by an organization and may be offered the service for itself only (PrC). These combinations are listed explicitly in Table I. The software also allows for hosting the infrastructure off-premise (PrC-offP) or offering the service to other consumers (PuC-offP). This free choice is listed as "DBL-*-*-IaaS-*-*" in Table I.

Google App Engine Blobstore [20] by Google Inc. is a replicated object store (distrDS). Objects can be stored and retrieved using a key. Blobstore is offered on a Public Cloud (PuC-offP) as SaaS. The customer cannot configure any property. The concept of key-value storage is different from MySQL leading to an incompatibility (incomp).

Google App Engine Datastore [21] by Google Inc. is a non-relational replicated data store (distrDS) offered by Google Inc. It is offered on a Public Cloud (PuC-offP) as SaaS. The customer cannot configure any property. For querying data, the Google Query Language (GQL) supporting only a subset of functionality from SQL is provided, leading to an incompatibility (incomp).

Microsoft Live SkyDrive [22] by Microsoft offers a shared folder. SkyDrive is offered on a Public Cloud (PuC-offP) as

SaaS. Access rules and sharing options can be configured. We assume that the data is replicated (distrDS). The shared folder is incompatible (incomp) with MySQL.

Microsoft SQL Azure [23] is a Cloud-enabled SQL-database solution offered by Microsoft. It is offered on a Public Cloud (PuC-offP) as PaaS. Currently, there is no synchronization offered (centrDB), but a Microsoft Sync Framework is planned. Microsoft SQL Azure is incompatible with MySQL (incomp).

MongoLab [24] offers hosting MongoDB in the Cloud. MongoDB is a NoSQL database. The MongoLab service is offered on a Public Cloud (PuC-offP) as SaaS. The data is replicated (distrDB). NoSQL is incompatible with MySQL (incomp).

IV. Discussion

In this section, we discuss aspects concerning the distribution of an application in the Cloud and limitations of the approach.

To find an optimal hosting for each application layer enabling the distribution applying the Cloud computing paradigm there are requirements concerning the application architecture. The adjacent layers have to be loosely coupled to ease distribution. Loosely coupling and dynamic binding are fundamental characteristics of the architectural paradigm Service-Oriented Architecture (SOA [26]). The strong relationship between SOA and Cloud computing is described by Behrendt et al. [27]. Within SOA the concept of a service is defined as publicized package of functionality that is composable and discoverable based on its description as well as terms and conditions of use. As a consequence, each application layer shown in Figure 1 can be seen as a service. Each service itself might be again structured into the different application layers. This consideration is out of scope of this paper.

There are two different approaches available when splitting an application for distribution and migration: vertical splitting and horizontal splitting [3], [28], [29]. Vertical splitting of an application is done by sub-dividing the application in separate layers as shown in Figure 1. For instance, horizontal splitting denotes that each layer of the application can itself also be horizontally subdivided. As a consequence separate parts of one layer can be deployed using separate deployment models, cf. Figure 1. Thus, confidential data can be hosted in the part of the database layer deployed in the Private Cloud. Uncritical data that might have to be public available, can be stored in the part of the database layer deployed in the Public Cloud. Horizontal splitting of an application directly leads to the Hybrid Cloud deployment model. Different Cloud data hosting solutions based on the Hybrid Cloud can be classified using the taxonomy.

The presentation of the taxonomy does not go into detail of each Cloud data hosting solution. We see business conditions, incentives, challenges, limitations, impact on application architecture, and an examination of technical aspects as noteworthy discussion points. The technical aspects should include hosting type, tenancy, configuration, as well as management effort. These details support building an architectural decision model [30], which may support application architects in decision making.

This paper does not deal with database architectures enabling scalability. A recent discussion regarding MySQL is presented by Zawodny and Balling [31]. A discussion on general database architectures is provided by Delis and Roussopoulos [32].

V. Related Work

This section presents the related work building the basis for the taxonomy for Cloud data hosting solutions. We examined publications on architecture best practices of Cloud applications as well as proven approaches of currently available Cloud applications and Cloud services.

IBM's Cloud Computing Reference Architecture 2.0 [27] defines the basic architecture elements and their relationships. Moreover, fundamental architecture principles are defined being essential to manage and provide Cloud services.

Adler [33] provides following three main contributions regarding best practices for scalable applications in the Cloud: First of all the challenges implied when building a scalable application in the Cloud are examined. Additionally architectural designs and techniques are explained to cope with the challenges on the level of the architecture as well as underlying infrastructure. Afterwards a reference architecture for scalable applications in the Cloud is introduced that is used by various RightScale[1] customers and enables its customer specific customization to a certain extend. The whole paper presumes a multi-layer architecture as we do in this publication and especially investigates the layer specific characteristics as well as the question how to achieve scalability in each layer.

Netflix[2] is a company delivering movies and TV shows over the Web. Anand gives a detailed report how Netflix switched from data hosting in an on-premise data center to the usage of Amazon SimpleDB and Amazon S3 for storing their data off-premise in the Cloud [34]. They investigated the compatibility and whether the Cloud solution offers a distributed database. These criteria are re-used in our taxonomy.

Kossmann and Kraska [35] analyze the offerings of the main PaaS storage provider Amazon, Google, and Microsoft and examine the common features and differences. They classify along three dimensions: deployment type, service type, and supported workloads. Supported workloads have been measured. We also use deployment type and service type as classification criteria and show all possible classes not bound to existing solutions.

[1] http://www.rightscale.com
[2] http://www.netflix.com

An evaluation of a Web application using different architectures with respect to the data layer taking into consideration partitioning, replication, and cashing is presented in [36]. For the realization of the data layer Cloud services from Amazon, Google, or Microsoft are used. Afterwards, the Web application variants are tested and compared regarding performance and costs. Though, the focus is both on the performance of the corresponding data layer and on the end-to-end performance of the whole Web application variant. The focus on our work is on general architectural possibilities of building a data layer in the Cloud and not a detailed investigation of performance issues.

A Cloud computing vendors taxonomy [37] enabling a comparison of different Cloud services is provided by OpenCrowd[3]. The four main categories provided are based on the three Cloud service models SaaS, PaaS, and IaaS with the additional category Cloud software. Within these categories the Cloud services are grouped based on their application domains, e. g., backup and recovery, database, content management, and Cloud integration. We re-used the service model criteria in our taxonomy, but we provide a more detailed taxonomy by considering six properties geared towards Cloud data hosting solutions.

Architectural decisions on the choice among NoSQL and SQL databases are presented by Hoff [38]: The capabilities of each type of product is examined and questions guiding through the choice are presented. Cloud-related factors are not regarded. Our work classifies available products using a taxonomy. Questions guiding through the taxonomy are our next step in our future work.

A list of all available NoSQL databases is provided by Edlich [39]. The list is categorized using the type of the NoSQL database such as document store, key value store, and graph database. The databases are not categorized into different Cloud data hosting solutions as we do.

All in all, there currently is no taxonomy offering a categorization of data hosting solutions in the Cloud. This paper provides such a taxonomy.

VI. Conclusion and Future Work

Cloud computing offers different possibilities to store data. This paper made these possibilities explicit by classifying them into *different Cloud data hosting solutions* by using *six properties*. If Hybrid Clouds are disregarded, *sixty different pure Cloud data hosting solutions* have been discovered. Otherwise, the taxonomy is infinite as Hybrid Clouds allow arbitrary combinations of pure Cloud deployment models. We classified existing Cloud data store offerings applying the taxonomy.

Currently, focusing on vertical splitting of the application into layers it is also possible to split each layer itself horizontally, cf. Section IV. We will investigate the horizontal splitting and distribution of application layers in our future work. For example, what are the consequences and implications of splitting the database layer to store the confidential part of the data on-premise in the Private Cloud and the uncritical data off-premise in the Public Cloud?

The database layer is not the only layer in applications today. Our future work is to research on different deployment models for these layers, especially data access and business layer. For instance, what does it mean if the database layer is moved to the Private Cloud and the business layer is moved to the Public Cloud and the other layers remain in a non-Cloud setting? The presented taxonomy for pure Cloud data hosting solutions will be expanded to Cloud-enabled application hosting topologies, where the other application layers are also regarded.

[3]http://www.opencrowd.com

Acknowledgment

The research leading to these results has partially received funding from the 4CaaSt project (http://www.4caast.eu) part of the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 258862. This paper expresses the opinion of the authors and not necessarily the opinion of the European Commission. The European Commission and the authors are not liable for any use that may be made using the information contained in this paper. We thank Gerd Breiter (IBM) and Ralph Retter (T-Systems) for their valuable input.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[2] P. Mell and T. Grance, "Cloud Computing Definition," *National Institute of Standards and Technology, Version 15*, July 2009.

[3] J. Dunkel, A. Eberhart, S. Fischer, C. Kleiner, and A. Koschel, *Systemarchitekturen fuer Verteilte Anwendungen: Client-Server, Multi-Tier, SOA, Event Driven Architectures, P2P, Grid, Web 2.0*. Hanser Verlag, 2008.

[4] PlateSpin (subsidiary of Novell), "PlateSpin Migrate." [Online]. Available: http://www.novell.com/products/migrate/

[5] VMware, "VMware vCenter Converter." [Online]. Available: http://www.vmware.com/products/converter/

[6] Microsoft, "Microsoft Visual Server Migration Toolkit." [Online]. Available: https://www.microsoft.com/windowsserversystem/virtualserver/overview/vsmtdatasheet.mspx

[7] RunMyProcess, "RunMyProcess." [Online]. Available: http://www.runmyprocess.com/

[8] Amazon.com, Inc., "Amazon Simple Storage Service (Amazon S3)." [Online]. Available: http://aws.amazon.com/s3/

[9] C. Strauch, "NoSQL Databases," February 2011. [Online]. Available: http://www.christof-strauch.de/nosqldbs.pdf

[10] Oracle Corporation, "MySQL." [Online]. Available: http://www.mysql.com

[11] Amazon.com, Inc., "Amazon Elastic Block Store (EBS)." [Online]. Available: http://aws.amazon.com/ebs/

[12] ——, "Amazon Elastic Compute Cloud (Amazon EC2)." [Online]. Available: http://aws.amazon.com/ec2/

[13] ——, "Amazon Relational Database Service (Amazon RDS)." [Online]. Available: http://aws.amazon.com/rds/

[14] ——, "Amazon SimpleDB." [Online]. Available: http://aws.amazon.com/simpledb/

[15] ——, "Amazon Virtual Private Cloud (Amazon VPC)." [Online]. Available: http://aws.amazon.com/vpc/

[16] SuccessBricks, Inc., "ClearDB." [Online]. Available: http://cleardb.com/

[17] Rackspace, "Cloud Files." [Online]. Available: http://www.rackspace.com/cloud/cloud_hosting_products/files/

[18] Salesforce.com, "Database.com." [Online]. Available: http://www.database.com

[19] Eucalyptus Systems, Inc., "Eucalyptus." [Online]. Available: http://www.eucalyptus.com/

[20] Google, "Google App Engine Blobstore." [Online]. Available: https://code.google.com/intl/en/appengine/docs/java/blobstore/

[21] ——, "Google App Engine Datastore." [Online]. Available: https://code.google.com/intl/en/appengine/docs/java/datastore/

[22] Microsoft, "Microsoft Live SkyDrive." [Online]. Available: http://explore.live.com/windows-live-skydrive

[23] ——, "Microsoft SQL Azure." [Online]. Available: http://www.microsoft.com/windowsazure/sqlazure/

[24] MongoLab, "MongoLab." [Online]. Available: http://mongolab.com

[25] SalesForce.com, Inc., "An Introduction to the Force.com Database." [Online]. Available: http://wiki.developerforce.com/index.php/An_Introduction_to_Force_Database

[26] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[27] M. Behrendt, B. Glasner, P. Kopp, R. Dieckmann, G. Breiter, S. Pappe, H. Kreger, and A. Arsanjani, "Introduction and Architecture Overview IBM Cloud Computing Reference Architecture 2.0," February 2011. [Online]. Available: https://www.opengroup.org/cloudcomputing/uploads/40/23840/CCRA.IBMSubmission.02282011.doc

[28] B. Zeller and A. Kemper, "Experience report. exploiting advanced database optimization features for large-scale sap r/3 installations," in *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002.

[29] S. Agrawal, V. Narasayya, and B. Yang, "Integrating vertical and horizontal partitioning into automated physical database design," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM, 2004.

[30] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, "Managing architectural decision models with dependency relations, integrity constraints, and production rules," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1249–1267, 2009.

[31] J. Zawodny and D. Balling, *High Performance MySQL: Optimization, Backups, Replication, Load-balancing, and More*. O'Reilly & Associates, Inc. Sebastopol, CA, USA, 2004.

[32] A. Delis and N. Roussopoulos, "Performance and scalability of client-server database architectures," in *Proceedings of the 18th International Conference on Very Large Data Bases*, 1992.

[33] B. Adler, "Building Scalable Applications In the Cloud: Reference Architecture & Best Practices, RightScale Inc." 2011. [Online]. Available: http://www.rightscale.com/info_center/white-papers/building-scalable-applications-in-the-cloud.php

[34] S. Anand, "Netflix's Transition to High-Availability Storage Systems," October 2010. [Online]. Available: https://sites.google.com/site/practicalcloudcomputing/index/Netflix%E2%80%99sTransitiontoaKey_v3.1.pdf?attredirects=0&d=1

[35] D. Kossmann and T. Kraska, "Data management in the cloud: Promises, state-of-the-art, and open questions," *Datenbank Spektrum*, vol. 10, no. 3, pp. 121–129, December 2010.

[36] D. Kossmann, T. Kraska, and S. Loesing, "An Evaluation of Alternative Architectures for Transaction Processing in the Cloud," in *Proceedings of the 2010 International Conference on Management of Data*. ACM, 2010.

[37] OpenCrowd, "Cloud Computing Vendors Taxonomy." [Online]. Available: http://cloudtaxonomy.opencrowd.com/

[38] Todd Hoff, "35+ Use Cases for Choosing Your Next NoSQL Database," June 2011. [Online]. Available: http://highscalability.com/blog/2011/6/20/35-use-cases-for-choosing-your-next-nosql-database.html

[39] Stefan Edlich, "List of NoSQL Databases," July 2011. [Online]. Available: http://nosql-database.org

All links were last followed on October 11, 2011.