

Model-Driven Cloud Data Storage

Juan Castrejón¹, Genoveva Vargas-Solar¹,
Christine Collet¹, and Rafael Lozano²

¹ Université de Grenoble, LIG-LAFMIA,
681 rue de la Passerelle, Saint Martin d'Hères, France

² Instituto Tecnológico y de Estudios Superiores de Monterrey,
Campus Ciudad de México, Calle del Puente 222, México, México

Abstract. The increasing adoption of the cloud computing paradigm has motivated a redefinition of traditional software development methods. In particular, data storage management has received a great deal of attention, due to a growing interest in the challenges and opportunities associated to the NoSQL movement. However, appropriate selection, administration and use of cloud storage implementations remain a highly technical endeavor, due to large differences in the way data is represented, stored and accessed by these systems. This position paper motivates the use of model-driven techniques to avoid dependencies between high-level data models and cloud storage implementations. In this way, developers depend only on high-level data models, and then rely on transformation procedures to deal with particular cloud storage details, such as different APIs and deployment providers, and are able to target multiple cloud storage environments, without modifying their core data models.

Keywords: Cloud computing, Model-driven engineering, Data storage

1 Introduction

Cloud computing represents one of the most promising paradigms for software development nowadays, due to its natural separation between users, applications and the services they require. In this model, computing resources are provided as services, easily accessible over a distributed network [11]. The focus of this new paradigm is to reach high levels of reliability, scalability and performance, by relying on a utility computing model [11].

Cloud storage represents a new paradigm to *store*, *retrieve* and *manage* large amounts of data, using highly scalable and reliable distributed infrastructures. This area has received a great deal of attention in recent years, due to a growing interest in the challenges and opportunities associated to the NoSQL movement [3]. However, unlike traditional environments, where the use of the *relational model* is pervasive, there is a wide variety of data models that can be used in cloud applications [3]. These data models include [3]: *key-value*, *document*, *extensible record*, *graph* and *relational* repositories. Each of these data models are designed for different use cases, and provide different support for functional and non-functional requirements of distributed systems [3], such as different degrees

of *consistency*, *scalability*, *replication* and *concurrency* [3]. Moreover, there is also a wide variety of both public and private providers for the distributed infrastructure that is required for cloud data storage [14]. These providers offer different combinations of *pricing*, *support*, *service levels*, and usually have different APIs to *store*, *retrieve* and *manage* data. These differences make it difficult to design and deploy applications targeting different cloud environments [15].

A key challenge in this heterogeneous environment is the appropriate selection of a data store that best matches the requirements of particular applications [14, 15]. This can be a daunting task, due to the high number of implementations in this environment, over 120 as of this writing [4], and the technical knowledge required to make an appropriate selection, as outlined in [14].

Furthermore, applications may require more than one type of data store, in order to support different use cases. In this regard, the appropriate use of data stores, either traditional or NoSQL, to support multiple use cases in a single application, is currently being studied as part of an emerging movement, named *polyglot persistence* [5, 16]. Nonetheless, the selection and administration of suitable storage systems for each use case, remain an open challenge [5].

This paper motivates the use of model-driven engineering (MDE) techniques [9], in order to characterize cloud data storage requirements, and to effectively encapsulate the *selection*, *administration* and *use* of cloud data storage implementations, specially, in polyglot persistence environments. We believe that MDE is a natural fit for this purpose, due to its emphasis in relying on different levels of modeling notations [9], which can be ultimately used to generate the implementation of software systems [1]. In particular, these multi-level structures can be used to avoid dependencies between high-level data models, cloud storage implementations and deployment providers.

The remainder of this paper is organized as follows. Section 2 outlines collaborations between cloud data storage and MDE. Section 3 describes related work. Finally, conclusions and future challenges are discussed in Section 4.

2 Model-driven cloud data storage

In this section, we outline a set of collaborations between cloud data storage and MDE, that are intended to avoid dependencies between high-level data models and cloud storage implementations, overcoming the problems described in the previous section. In particular, we strive for the following objectives: (i) provide adequate notations and environments to characterize cloud data storage requirements; (ii) selection of storage implementations and deployment providers; and, (iii) management of the required artifacts (configuration files for the deployment environments and cloud data management interfaces [6, 17, 19]) to work with different combinations of cloud storage implementations and providers.

2.1 Data modeling for the cloud

The specification of data models for software systems is traditionally performed using notations such as entity-relation (ER) or UML class diagrams. MDE tech-

niques can currently be applied to transform these diagrams into their corresponding relational database models and programming language entities.

However, these notations are usually not enough to characterize all the possible cloud data models. For instance, consider the *document* data model [3], that lacks a rigid schema and in which semi-structured information can be stored. Another example would be the adequate modeling of *families of attributes* [3], usually associated to *extensible record* scenarios [3]. In this regard, different techniques are currently being proposed to overcome these limitations [8, 13].

One of the objectives of our current work is the definition of adequate notations and environments for the modeling of datasets, and their associated functional and non-functional requirements, for cloud environments. For the specification of these requirements, we intend to rely on the ISO/IEC Software Product Quality Requirements and Evaluation (SQuaRE) standards [7], that already define software quality characteristics, such as *performance efficiency*, *portability* and *functional suitability* [7]. We propose to define these characteristics through the association of quality metrics relevant to cloud scenarios. At this moment, we can reuse previously proposed metrics [13, 14], such as *performance*, *cost* and *access latency*, but further validation of these quality metrics is also required.

Developers would then have to specify expected values for these metrics, according to the requirements of each of their datasets, and would not have to deal with low-level details of how to accomplish these expected values.

We intend to organise our modeling notations based on a traditional MDE structure of platform independent and specific models (PIM/PSM) [9], in regard to cloud storage implementations. In this way, we could integrate current research and industrial efforts, such as specification languages for modeling cloud environments [10], and different cloud data management interfaces [6, 17, 19].

2.2 Data storage selection

In order to ease the selection of data storage implementations and providers, we propose a decision process based on the analysis of historic data and usage patterns, both in test applications and within systems generated in our modeling environment. This analysis could be performed in a non-intrusive manner, by automatically generating aspect-oriented programming (AOP) monitoring artifacts, as outlined in [2]. In particular, dynamic crosscutting techniques can be used to monitor the behavior of the selected data stores, regarding the metrics associated to the SQuaRE quality standards. This monitoring information could then be used to compare with the expected values specified by the users of our modeling notations and environments. In turn, this analysis could be automatically integrated in applications designed with our modeling notations, with the objective of sharing the results in an open and collaborative environment, that could be exploited by new users of cloud data storage, and by our own modeling tools, as input for the data storage recommendation engine.

Developers could also generate the artifacts to work, at the same time, with multiple combinations of implementations and providers, as outlined in [2]. For instance, this would be helpful to compare their performance in real scenarios.

2.3 Cloud artifacts generation and management

Once the data storage implementations and providers are selected for the application datasets, we propose to use transformation procedures to generate the low-level artifacts to work with them, that is, configuration files for the deployment environments and cloud data management interfaces. This process could be performed using different levels of transformation procedures, each of them more dependant with particular cloud storage implementations and providers, using a similar approach as modern application development tools [18].

For example, an initial transformation could be defined between the graphical data models and an intermediate domain specific language (DSL), possibly extending the work in [10] and [18]. From this DSL, we could generate configuration files for the particular storage implementations, the AOP monitoring aspects, and the configuration of data management interfaces [6, 17, 19].

3 Related work

In recent years, collaborations between cloud computing and MDE have started to be acknowledged by the research community [1, 12]. These collaborations cover the main service models of cloud computing [11], that is, *Software as a Service* (SaaS), *Platform as a Service* (PaaS) and *Infrastructure as a Service* (IaaS).

In [1], the *Modeling as a Service* (MaaS) initiative is proposed as an approach to deploy and execute model-driven services over the Internet. This initiative is aligned with SaaS principles, since consumers do not manage the underlying cloud infrastructure and deal mostly with end-user systems. Our work deals with lower level service models (PaaS and IaaS) by allowing control over the deployed applications and the configuration settings of the deployment environments.

A model-driven approach for designing and deploying scalable applications on cloud platforms is described in [12]. This approach promotes the use of graphical models in order to capture cloud requirements, in particular, scalability features. These models are then bundled into a generic platform that automatically deploys them into PaaS and IaaS environments. Instead of striving for a generic platform, capable of managing all possible features of running scalable applications in the cloud, our work is focused only on data storage features.

An approach for the automatic selection of cloud storage services is proposed in [14]. This approach relies on the characterization of storage systems, based on capabilities, such as *performance* and *cost*, and on the specification of requirements for application datasets, such as *expected dataset size*, *access latency* and the *number of concurrent clients*. Based on this information, an assignment of datasets to the storage systems is proposed, for example, using a mathematical model that strives for optimal data allocation [15]. In comparison, we propose a recommendation engine based on the identification of common scenarios and usage patterns, both in test applications and within systems generated in our modeling environment. Our work also targets the automatic generation of the artifacts required to use and manage the proposed set of data storage implementations and providers.

4 Conclusions and Future work

This paper outlined collaborations between MDE and cloud data storage, intended to facilitate both the specification of cloud data storage requirements, and to encapsulate the *selection*, *administration* and *use* of cloud data storage implementations and deployment providers. We have also mentioned challenges that are required to make these collaborations possible. These challenges are currently being addressed by members of our research group.

References

1. Bruneliere, H., Cabot, J., Jouault, F.: Combining model-driven engineering and cloud computing. In: Modeling, Design, and Analysis for the Service Cloud Workshop. MDA4ServiceCloud '10 (2010)
2. Castrejón, J.: An aspect oriented approach for the synchronization of instance repositories in model-driven environments. RCS 52, 179–189 (2011)
3. Cattell, R.: Scalable sql and nosql data stores. SIGMOD Rec. 39, 12–27 (May 2011)
4. Edlich, S.: List of nosql databases. <http://nosql-database.org/> (March 2012)
5. Fowler, M.: Polyglot persistence. <http://martinfowler.com/bliki/PolyglotPersistence.html> (November 2011)
6. jclouds Inc.: jclouds. <http://www.jclouds.org/> (March 2012)
7. ISO/IEC:25010:2011: Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. ISO, Geneva, Switzerland (2011)
8. Katsov, I.: Nosql data modeling techniques. <http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/> (March 2012)
9. Kent, S.: Model driven engineering. In: Butler, M., Petre, L., Sere, K. (eds.) Integrated Formal Methods, LNCS, vol. 2335, pp. 286–298. Springer Berlin (2002)
10. Liu, D., Zic, J.: Cloud#: A specification language for modeling cloud. In: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing. pp. 533–540. CLOUD '11, IEEE Computer Society, Washington, DC, USA (2011)
11. National-Institute-Standards-Technology: The nist definition of cloud computing. <http://csrc.nist.gov/publications/PubsSPs.html> (September 2011)
12. Peidro, J.E., Muñoz-Escóí, F.D.: Towards the next generation of model driven cloud platforms. In: 1st International Conference on Cloud Computing and Services Science. pp. 494–500. CLOSER '11 (2011)
13. Piscopo, N.: Best practices for moving to the cloud using data models in the daas life cycle. <http://erwin.com/whitepapers/> (February 2012)
14. Ruiz-Alvarez, A., Humphrey, M.: An automated approach to cloud storage service selection. In: Proceedings of the 2nd international workshop on Scientific cloud computing. pp. 39–48. ScienceCloud '11, ACM, New York, NY, USA (2011)
15. Ruiz-Alvarez, A., Humphrey, M.: A model and decision procedure for data storage in cloud computing. In: Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. CCGrid '12 (2012)
16. Schmidt, S.: Nosql: The dawn of polyglot persistence. <http://codemonkeyism.com/nosql-polyglott-persistence/> (January 2010)
17. SpringSource: Spring data projects. <http://www.springsource.org/spring-data> (March 2012)
18. SpringSource: Spring roo. <http://www.springsource.org/spring-roo> (March 2012)
19. Storage-Networking-Industry-Association: Cloud data management interface. <http://www.snia.org/cdmi> (September 2011)