2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications

Service Datastores in Cloud Governance

Adrian Copie*†, Teodor-Florin Fortiş*†, Victor Ion Munteanu*†, and Viorel Negru*†
*Institute e-Austria, Timişoara, Romania
†Faculty of Mathematics and Informatics
Department of Computer Science
West University of Timişoara, Timişoara, Romania
e-mail: {adrian.copie,fortis,vmunteanu,vnegru}@info.uvt.ro

***Abstract*—A direct consequence of adopting Cloud Computing is the creation of new marketplaces for small and medium-sized enterprises (SMEs). Having a strong support offered by Platform as a Service (PaaS) paradigm in terms of automation and management, combined with cloud governance, SMEs have a real chance to get back in competition with the large enterprises.**

The implementation of Cloud Governance solutions offers the necessary support for storing and manipulating data which is relevant for various aspects of applications, ranging from business to technical support for deployed services. Our paper focuses on analysing and detailing governance specific datastores and the way they are able to provide system-wide services.

***Keywords*-service datastore, cloud governance, cloud computing, distributed databases**

I. Introduction

Cloud Computing is a new paradigm that focuses on enabling resource utilization at a reduced cost. It is built around five essential characteristics: on-demand self-service, resource pooling, measured services, broad network access and rapid elasticity [1], as well as technical characteristics like service orientation, loosely coupling, strong fault tolerance, link with business models, virtualization, or ease of use [2].

These characteristics along with new business models [3], [4] offered by the cloud, enable Small and Medium-sized Enterprises (SMEs) easy migration to cloud environments by delegating infrastructure management to cloud providers and benefiting from the given pay per use economic model.

By benefiting from the 'as-a-service' paradigm, SMEs are free to engage in partnerships with each other in order to form virtual enterprises where their offerings are grouped together and exposed as a unique entity.

However, due to the large diversity in vendor proprietary technologies and the lack of standardization, cloud adoption has been made difficult, leading to what is know a as *vendor lock-in* problem.

Efforts have been made to bridge the gap between various provider technologies through the development of Platform-as-a-Service solutions (PaaS) which facilitate development, deployment and execution of applications in the cloud environment, as well as integration with components for cloud resources management.

These platforms are usually built on top of existing Infrastructure-as-a-Service (IaaS) or, in rare situations, on top of Hardware-as-a-Service (HaaS), and focus on Software-as-a-Service (SaaS) development by providing one or more of the following: cloud-independent approach to development and deployment (mOSAIC, Cloud Foundry[1], OpenShift[2], Morfeo 4CaaSt, ActiveState's Stackato), cloud resource management and scaling (mOSAIC, OpenShift, WSO2 Stratos[3], ActiveState's Stackato[4]), framework support (Cloud Foundry, OpenShift, ActiveState's Stackato), application services (Cloud Foundry, OpenShift, ActiveState's Stackato), application lifecycle management (Morfeo 4CaaSt[5]), or business ecosystem (Morfeo 4CaaSt).

While typical PaaS solutions offer a framework for the development and deployment of cloud applications, several aspects like resource management, business integration etc. will require specialized components. In the case of the mOSAIC project, the Cloud Agency (CA) handles resource management (provisioning, monitoring and reconfiguration) effectively making it a cloud management solution.

A set of complementary services that relate to Cloud Management is necessary, as specified in [5], [6], [7], as well as a solution that addresses integration issues, as identified by G. Gruman [8]. Cloud Governance could prove to be that solution by providing a mechanism to complement Cloud Management through a set of services that address its limitations. Cloud Governance also comes as a solution to the increasing demand of integration of different SaaS offerings [9], [10].

In order to enable cloud governance typical activities like service management (publishing, brokering, monitoring, billing), security and others, there is a need for the integration and manipulation of related data. However, given the diversity in data types and usage patterns, having a homogeneous approach for storage is infeasible leading to

[1]http://cloudfoundry.org/
[2]https://openshift.redhat.com/app/
[3]http://wso2.com/cloud/stratos/
[4]http://www.activestate.com/stackato
[5]http://4caast.morfeo-project.org/

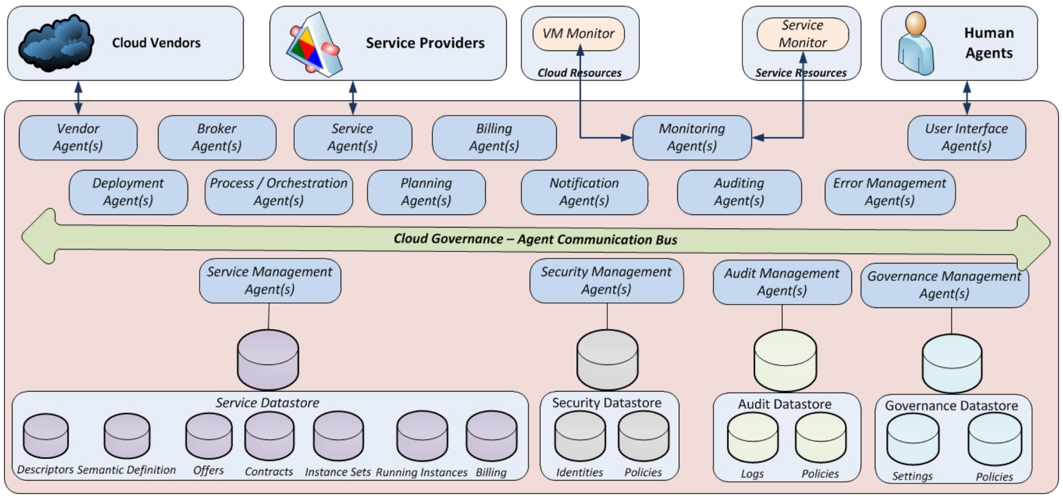

Figure 1. A multi-agent governance architecture

a need to identify specific requirements for each datastore category.

The remainder of this paper is structured as follows. The mOSAIC project is covered in Section II. A governance architecture is briefly discussed in Section III along with the associated service lifecycle in section IV. Our main work is presented in Section V and covers mainly the Service Datastore from the Cloud Governance architecture. Finally, conclusions and future work are presented in Section VI.

II. mOSAIC Project

'Open source API and Platform for multiple Clouds'[6] (mOSAIC) is a FP7-ICT project, that aims to facilitate cloud adoption through provider independent service development and deployment. mOSAIC API and Cloud Agency are the two main components of mOSAIC. While the API allows the development of multi-cloud oriented applications, the Cloud Agency[11], [12] covers management related functionality.

The API has been designed to provide a unified cloud API for building interoperable applications across different Cloud providers [13] by open and standard interfaces [14]. It is be best described though several use cases [15].

The Cloud Agency is a multi-agent system that has been designed to handle resource provisioning (buying resources from cloud vendors), monitoring (low level resources), re-configuration (at IaaS level) and dynamic discovery and mapping of cloud providers. It can work independent of or at IaaS level with mOSAIC's API.

III. Cloud governance

A cloud governance solution can be built in close relation with mOSAIC's Cloud Agency (CA) and closely following Distributed Management Task Force (DMTF)'s white papers [5], [6]. The CA is a cloud management solution exposing its functionality as services within cloud governance one.

[6]http://www.mosaic-cloud.eu

The proposed multi-agent system covers most of the cloud governance's functionality: service management, security and privacy management, service lifecycle management and monitoring. Each of these is covered by a subset of agents in the system.

The four interdependent subsystems, as identified in Figure 1, are as follows:

- Services – handles all service-related functionality, like publish, update, remove, query, and instance handling;
- Security – handles security issues like identity tracking, issuing security tokens and validating them;
- Audit – has the functionality of an archiver by holding all system and service logs and can alert in case a problem has been detected;
- Management – Contains information about the whole system, makes system wide decisions related to agents and functioning services;

These subsystems produce and consume essential data for the governance process. This information is not transient and it is also important to be accurate and always available as basis for subsequent governance decisions. The requirements for the underlying storage systems are mostly expressed in terms of availability and reliability but due to the complexity of the processed information, other general conditions occurs. They have to elastically scale up and down with the data load, they have to meet the service level agreements and they must offer performance even under heavy loads.

A. *Cloud Governance Datastores*

The Cloud Governance multi-agent system is in close connection with dedicated datastores that hold specific data which is critical for the entire component operation,

1) Governance Datastore: The Governance Datastore holds general information related to the name of the virtual machines on which it is running, together with the policies regarding the VMs and agents scalability.

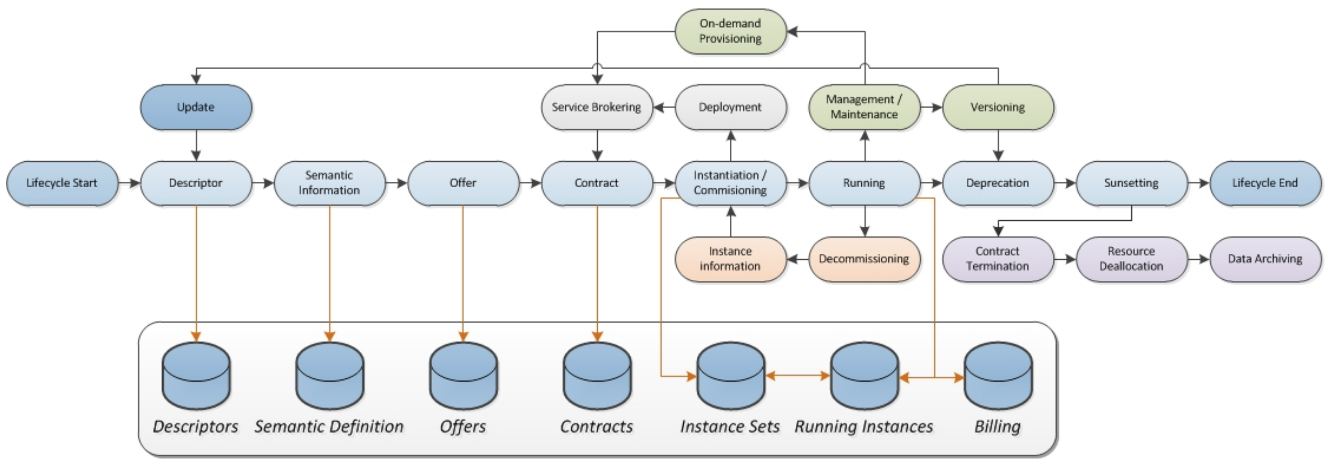

Figure 2. Service lifecycle and its relation with service datastores

2) Audit Datastore: Information about monitoring of services, billing and security audit is persisted in the Audit Datastore. Usually, cloud services are used on a pay-as-you go method, such that the consumer is directly interested in monitoring the SLA agreed with the provider. The monitored results are stored in this type of datastore and they are used in the billing process as well as for triggering events in the superior layers in case of SLAs violations.

3) Security Datastore: The Security Datastore holds critical information related to the Cloud Governance component in terms of accounts and access control lists. All the cloud service consumers keep their credentials in this datastore in order to be used further in the authentication and authorization processes.

4) Service Datastore: The Service Datastore can be seen itself as a special cloud service which must fulfil some basic requirements like high availability, reliability and scalability due to its fundamental importance for the entire cloud governance process. It is a particular type of datastore, built around a centralized repository that hosts all the existent information about the cloud services in the system.

Cloud governance could involve the management of a wide range of distinct or loosely coupled cloud services that are to be combined and orchestrated together in order to be used in various application, for solving different kind of problems.

IV. Service lifecycle

The ability to govern services is essential for a cloud environment. Handling the service lifecycle and lifecycle related functionality is a must for any real governance solutions. Cloud governance's approach to service lifecycle is similar to that of SOA governance's but is different in that it has to take into account service distribution across clouds and service scaling.

In designing a cloud service lifecycle, the following succession of operations provide a comprehensive approach:

- Design-time operations – these cover aspects of service description and semantic information; general information that allows the service to be published.
- Provisioning operations – these cover aspects of describing offers and service publishing, discovery and contracting.
- Deployment operations – these cover aspects of service instantiation and commissioning.
- Execution operations – these cover aspects of managing and billing.
- Retirement operations – these cover aspects of service retirement.

There is a closely connection between every stage of service lifecycle and the Service Datasores. This is better represented in Figure 2.

V. Service datastore

The information contained inside the Service Datastores is heterogeneous, every service being in relation with structured and also unstructured data. Its main purpose is to offer consumers a possibility to discover the best services that fit some specified requirements and maintaining other additional information. The service discovering process is realized through various searches based on different criteria, taking into consideration various metadata as parameters.

A. Components of the Service Datastores

Our proposal related to the Service Datastores component is to maintain a specialized database for every kind of information previously mentioned:

- *Service Descriptors* will contains general information about the services in the system;
- *Semantic Definitions* describes the services from the semantic point of view;
- The *Offers* database maintains information about the services offer;

- *Contracts* database keeps the already agreed contracts between the service providers and the consumers;
- *Instances* and *Running Instances* databases hold very similar data related to the instances of the services already existent in the system.
- Following the exploitation of the cloud services, financial data is generated that will be further used to invoice the consumers. This information is stored in the *Billing* database.

All these individual databases have a local scope, they belong to the Services Datastore component. However other catalogues are necessary in order to maintain relationships with the entities that use the services and they are called namely *Customers* and *Partners* databases. Their scope is global, being used also by other components of the Cloud Governance environment.

The interaction between the Service Datastores and the external environment represented by the service consumers is made through the Service Management agent which can issue various operations over the databases described above. These operations range from the service publishing and discovery to retrieving billing information for every running service in the system.

Table I
REQUIREMENTS FOR THE SERVICES DATASTORE COMPONENTS

Service Datastore Component	Highly Searchable	Optimized for	Specialized content	ACID support
Services Descriptors	Yes	Read	No	Yes
Semantic Definitions	Yes	Read	Yes	Yes
Offers	No	Read	No	Yes
Contracts	No	Read	No	Yes
Instances	No	Read	No	Yes
Running Instances	No	Read	No	Yes
Bills	No	Write	No	Yes

B. Requirements for the Service Datastore components

In real environments, the Service Datastores could host information about hundreds of services together with all their dependencies, which may lead to very large database size for which the scalability is not always simple to achieve in traditional implementations using a single database server. Adding supplementary storage servers leads to distributed databases solutions which do not guarantee that the three essential properties in a distributed system can be achieved at the same time: *consistency*, *availability* and *partition tolerance* [16], [17]. The Service Datastores are a mesh of various databases, each of these having individual requirements due to the specific of the stored data and their role inside the parent component. Table I summarize the most important specifications of the databases part of the Services Datastore.

C. Use cases related to Services Lifecycle

During its existence, a service passes through several phases, in each of these it interacts with various individual databases from the Services Datastore. In order to achieve different well defined goals, some use cases can be identified

1) Publishing a service: When a cloud service provider want to register a new cloud service, some steps must be followed. First of all, the service descriptor is stored inside the Services Descriptors database together with its semantic definition that will be placed in the Semantic Definitions. Every new cloud service comes with an offer that is also persisted in the Offers database.

2) Discovering and contracting a service: When a consumer needs a specific service, it will issue a request to the Service Descriptors database with a specific query. If the result of the query satisfies the consumer's conditions in terms of functional requirements, the Offers database is also consulted for the non-functional requirements, like SLAs and price. A positive result could lead to a contract establishment which represents an agreement between the cloud service provider and the service consumer. The resulted contract is stored inside the Contracts database and an entry with the characteristics of the service instance is added in the Instances database.

3) Instantiation and commissioning a service: When the cloud service starts, an entry related to the instance that is executed is added in the Running Instances database. During the execution phase of the cloud service, billing information is continuously generated and stored in the Billings database.

4) Retirement of a service: When a cloud provider decides to retire a service from the system, or the service reaches its end of life, it must be removed from the system. This action involves the termination of all the contracts between the service and its consumers, the removal of all the related information about the service from all the databases together with the resources de-allocation and archiving data.

D. Datastore implementation

In order to implement a service datastore in the cloud, we have analysed many types of storage systems: key-value stores, document databases and relational databases in the cloud.

1) Key-Value stores: Most of the key-value stores developments are based on [18]. The key-value stores have a simple data model consisting in a map or a dictionary, assigning or retrieving a value based on a unique key, having a very simple client interface, basically only *put*, *get* and *delete* operations are supported. The key is usually a string and the value is seen by the key-value store as a Binary Large Object (BLOBs) and it is the client's responsibility to interpret the content. These types of data stores are usually extremely scalable, they do not enforce any schema for the stored data and they are very reliable due to their replication mechanisms.

The information related to a service consists of many distinct chunks of data, from structured services descriptors to binaries representing the executable code or formatted data from the monitoring systems. Storing it into a key-value store involves using multiple keys with a mechanism of associating them to a specific service. Querying the Service Datastore is also a two-steps process, in the first phase all the keys must be fetched and next, the information associated to every key is interpreted. When a service subscribes in the datastore, a transactional mechanism must be implemented in order to allow the service dependencies to be written in an "all or nothing" manner.

2) Document databases: These types of databases are a step ahead compared with the key-value stores as they can persist more complex structures by allowing multiple key-values pairs in one document. The supported data types for keys and values is mostly the same as for the key-value stores but the document allows a better information management.

However they do not support document nesting. From the Service Datastores point of view, this solution is better in terms of grouping all the information in a single document, or splitting it in many documents but supporting a primitive linking mechanisms through document fields. Some of the document datastores like CouchDB (http://couchdb.apache.org) are ACID compliant which guarantees the data consistency during CRUD operations over the datastore, mainly when new services subscribe to the repository or other services unsubscribe from the datastore.

3) Graph databases: The data to be stored is represented using graph structures with nodes, edges and properties. The nodes are usually identified as objects while edges represent the relations between objects. The properties signify pertinent information related to nodes. The graph databases are mapped more straightforward to the object oriented applications. They scale very well horizontally, being able to offer good performance even for billions of records. They do not enforce a rigid schema, being suitable for dynamic and changing data. Many of them are fully transactional, fully ACID compliant.

4) Relational databases in the cloud: In the last years, despite the fact that the NoSQL movement was extremely active and brought a lot of storage products on the market, the relational database currently reinvented itself in the cloud, offering a so called NewSQL paradigm, basically the same functionalities as the traditional RDBMS systems but with dynamic elastic scalability basically for read operations, on a pay-as-you-go premise. From the Service Datastores point of view, using this solution enforces structuring the data, which is not possible all the time. The advantages consist in extremely flexible queries over the data and the ease of use due to the fact that SQL language is a very accessible to the developers.

Table II summarizes the characteristics of the studied database types and establishes a correspondence with the most relevant use cases discussed in this paper.

VI. Conclusions and future work

While cloud environments lack support for security, privacy and service management, the drive for cloud adoption goes on. Cloud governance plays an essential role and is vital for SMEs. Some companies already integrate cloud governance functionalities within their products, however they do not offer a complete cloud governance solution.

This paper addressed one core aspect of the Cloud Governance: its datastores with the accent placed on the Services Datastore. The requirements for Services Datastore components were identified in Table I and various possible distributed database types were discussed in order to establish an appropriate solution for each of them. The work conducted in this paper is essential for building a working prototype.

Based on the general characteristics of the most common types of distributed databases running in public or private cloud, and taking into account the individual requirements exposed in Table I together with the matching between the databases and the use cases exposed in Table II, we were able to establish the database types for the Service Datastores components:

- The Service Descriptors datastore is implemented as a

Table II
Correspondence between different distributed database types and general use cases

Database Type	ACID support	Searchability	Availability	Scalability	Usable for use case
Key-Value Store	No	No	Yes	Yes	-
Document Store	Yes	No	Yes	Yes	-
Graph Database	Yes	Yes	Yes	Yes	• Publishing a Service • Discovering and contracting a service • Instantiation and commissioning a service • Retirement of a service
NewSQL	Yes	Yes	Yes	Yes	• Publishing a service • Retirement of a service

relational database running in the public cloud due to its high availability requirements that could be better fulfilled by the specialized cloud providers.

- The Semantic Definitions database deals with specialized RDF data which also requires multiple searches, so an RDF repository allowing queries implemented in SPARQL language fit better the problem.
- The Offers, Contracts, Instances, Running Instances and Billings databases hold unstructured information but maintain various relations with the Services Descriptors database as well as each other. The best fit database for these requirements is graph database.

Our future work will focus on securing the cloud governance environment. This is achieved through having a system wide security and audit in place.

Acknowledgments

This work was partially supported by the grant of the European Commission FP7-ICT-2009-5-256910 (mOSAIC), and Romanian national grant PN-II-ID-PCE-2011-3-0260 (AMICAS). The views expressed in this paper do not necessarily reflect those of the corresponding projects' consortium members.

The work of first and third author was partially supported by the grant POSDRU 88/1.5/S/49516, "Creşterea calităţii şi a competitivităţii cercetării doctorale prin acordarea de burse". This project is co-funded by the European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by the West University of Timişoara in partnership with the University of Craiova and Fraunhofer Institute for Integrated Systems and Device Technology - Fraunhofer IISB.

References

[1] A. Mulholland, J. Pyke, and P. Fingar, *Enterprise Cloud Computing: A Strategy Guide for Business and Technology Leaders*. Tampa, FL, USA: Meghan-Kiffer Press, 2010.

[2] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *ICPP Workshops*, W.-C. Lee and X. Yuan, Eds. IEEE Computer Society, 2010, pp. 275–279.

[3] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer, "Cloud Computing– A Classification, Business Models, and Research Directions," *Business and Information Systems Engineering (BISE)*, vol. 1, no. 5, pp. 391–399, 2009, ISSN: 1867-0202.

[4] C. Weinhardt, A. Anandasivam, B. Blau, and J. Stößer, "Business Models in the Service World," *IEEE IT Professional, Special Issue on Cloud Computing*, vol. 11, no. 2, pp. 28–33, 2009, ISSN: 1520-9202. [Online]. Available: http://dx.doi.org/10.1109/MITP.2009.21

[5] DMTF. (2010, June) Architecture for Managing Clouds. Distributed Management Task Force. [Online]. Available: http://dmtf.org/sites/default/files/standards/documents/DSP-IS0102_1.0.0.pdf

[6] ——. (2010, June) Use Cases and Interactions for Managing Clouds. Distributed Management Task Force. [Online]. Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0103_1.0.0.pdf

[7] Cloud Computing Use Cases Group. (2010, July) Cloud computing use cases white paper. [Online]. Available: http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf

[8] G. Gruman. (2007, August) Integration issues may hinder saas adoption. CIO Update. [Online]. Available: http://www.cioupdate.com/trends/article.php/3695096/Integration-Issues-May-Hinder-SaaS-Adoption.htm

[9] S. Bennett, T. Erl, C. Gee, R. Laird, A. T. Manes, R. Schneider, L. Shuster, A. Tost, and C. Venable, *SOA Governance: Governing Shared Services On-Premise & in the Cloud*. Prentice Hall/PearsonPTR, 2011.

[10] T. Cecere. (2011, November) Five steps to creating a governance framework for cloud security. Cloud Computing Journal. [Online]. Available: http://cloudcomputing.sys-con.com/node/2073041

[11] S. Venticinque, R. Aversa, B. Di Martino, M. Rak, and D. Petcu, "A cloud agency for SLA negotiation and management," in *Proceedings of the 2010 conference on Parallel processing*, ser. Euro-Par 2010. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 587–594.

[12] R. Aversa, B. Di Martino, M. Rak, and S. Venticinque, "Cloud Agency: A Mobile Agent Based Cloud System," in *Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems*, ser. CISIS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 132–137.

[13] D. Petcu, C. Crăciun, M. Neagul, I. Lazkanotegi, and M. Rak, "Building an interoperability API for sky computing," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, july 2011, pp. 405–411.

[14] D. Petcu, S. Panica, and M. Neagul, "From grid computing towards sky computing. case study for earth observation," ser. Proceedings Cracow Grid Workshop 2010. Academic Computer Center, Poland, 2010, pp. 11–20.

[15] D. Petcu, "Identifying cloud computing usage patterns," in *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, sept. 2010, pp. 1–8.

[16] E. A. Brewer. (2000, July) Towards robust distributed systems. [Online]. Available: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

[17] S. Gilbert and N. Lynch, "Brewers conjecture and the feasibility of consistent, available, partition tolerant web services," *ACM SIGACT*, vol. 33, no. 2, jan. 2002.

[18] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazons highly available key-value store," in *IN PROC. SOSP*, 2007, pp. 205–220.