2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing

Determining the performance of the databases in the context of Cloud Governance

Adrian Copie*

*West University of Timişoara,
Faculty of Mathematics and Informatics
bvd. V. Pârvan, 4, 300223
Timişoara, Romania
e-mail: adrian.copie@info.uvt.ro

Teodor-Florin Fortiş†* Victor Ion Munteanu*†

†Institute e-Austria Timişoara,
bvd. V. Pârvan, 4, 300223
Timişoara, Romania,
e-mail: fortis@info.uvt.ro, vmunteanu@info.uvt.ro

***Abstract*—The maturation of Cloud Computing technologies, cumulated with the opportunities for Small and Medium Enterprises to migrate their activities into the Cloud, have led to new marketplaces where the competition with the large enterprises can be sustained through Platform-as-a-Service solutions combined with cloud Governance. Having a Cloud Governance solution that offers support for the services lifecycle, it creates the framework for a reliable and complex persistence layer that will store the data produced and consumed during the cloud governance processes. Choosing the appropriate databases for the datastores implementations is a very important step in designing a reliable cloud governance system. This paper focuses on establishing the requirements for the subsystems composing the datastores and determining the performances of two representative database types that are candidates to be used in various datastores.**

***Index Terms*—Cloud Computing, Cloud Management, Cloud Governance, Cloud Datastores**

I. Introduction

Being built around a set of five essential characteristics [1], Cloud Computing offers an ideal environment for enterprises of all sizes through decreased costs, real scalability and availability, adaptable resources to service needs, and others. Reducing the costs and complexity for enterprises is a major challenge for cloud computing [2].

While through the *on-demand self-service* the effort of managing resources is mostly oriented towards the consumer, an effective implementation of these management requirements is strongly dependent on closely monitoring the applications' requirements and the way these requirements change over time. On the other hand, the characteristics of *resource pooling* and *rapid elasticity* are fundamental for planning and anticipating resource utilization. Last but not least, *measure services* makes use of monitoring information in order to cover sensitive aspects related to service functionality and utilization.

As one of the main advantages brought by the Cloud Computing is the potential decrease in IT-related expenses for enterprises. through it, "*developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it*", as the "Above the Clouds" Berkeley report mentions [3].

A simple migration in the Cloud does not offer any warranties over the efficient exploitation of these characteristics. In order to achieve increased efficiency, small and medium-sized enterprises (SMEs) must rely on additional means like infrastructure abstraction (IaaS), platform abstraction (PaaS), and/or cloud management solutions, through which they can achieve the desired efficiency and ease of use in utilizing and managing different types of Cloud resources. Key "*technological dimension of cloud desires*" and their potential influence in cloud adoption are described in in [2, Table 3].

Adoption of Cloud Computing enables SMEs not only to enter in direct and real competition with larger players on specific markets by grouping themselves into virtual clusters but also the possibility to gather them into virtual enterprises. For such an approach to be used there is a need for a high level support for implementing and monitoring business and operational aspects of these virtual enterprises.

The development of complex tailored and/or targeted solutions, based on services available from the different members of a virtual enterprise, is comprised of tasks related to both business aspects and resource management, their corresponding monitoring concerns being a central focus in order to achieve these targets. This development could be enabled by the creation of a "*simple marketplaces with common rules, that enable the dynamic selection and consumption of functionality*" [5].

Adding a governance level inside virtual enterprises is a must, thus complementing existing management and platform solutions. The governance framework is built around a typical *Evaluate*, *Direct* and *Monitor* cycle, where the monitoring step is complementing cloud management activities, in order to implement "*conformance to policies, and performance against the plans*" [6], and is in close relation with the set of six principles introduced by the IT governance standard ISO/IEC 38500 and through different associated frameworks, including COBIT and ValIT.

Different industry standards and specifications exists, in-

978-0-7695-5094-7/13 $31.00 © 2013 IEEE
DOI 10.1109/3PGCIC.2013.40

cluding ITIL[1] or or TOGAF[2], and they can further augment *cloud management* and *cloud governance* solutions through the enforcement of compliance with security standards, specific legislation aspects (e.g. privacy, spam, trade practices, environmental, accessibility), intellectual property rights (software licensing agreements), record keeping requirements, and social responsibility standards.

The typical governance cycle Monitor-Evaluate-Direct, as emphasized through ISO 38500 specification [6], is complemented at management level by a Monitor-Analyze-Plan-Execute cycle performed around a resource Knowledge base, closely following the IBM's MAPE-K model identified in [7].

The different reference papers for cloud computing identify a series of complementary services that are relevant in the context of Cloud Management[8], [9]. Cloud Governance, as a complementary solution to Cloud Management, could address the existing limitations at management level, while addressing the specific integration requirements of the different SaaS offerings [11], [12].

In order to enable cloud governance, a good support for cloud service lifecycle is required [13]. In the context of service lifecycle typical activities (like service management, security, operations support, and others), specific need for the integration and manipulation of related data exists. With a variety of data types and usage patterns involved in the context of cloud characteristics, there is a need to identify specific requirements for the usage domain of each datastore category.

II. Motivation and Related work

A. Cloud management and governance

Cloud governance and cloud management are directly interlaced. While governance "*ensures that enterprise objectives are achieved by evaluating stakeholders' needs, conditions and options through decision making, prioritization, monitoring of performance and compliance"* [14], cloud management assures that the enterprise objectives set by the governance layer are achieved through planning, deploying, running and monitoring current activities.

Three simple principles are set in relation with cloud governance and management tasks [6], [15]:

- Complete solution evaluation (current / future), including current strategies, proposals and supply arrangements (be it internal or external), trends that may affect the solution (technological, economical, social or political changes).
- Complete implementation of plans and policies to ensure that the solution meets its design objectives.
- Complete monitoring of conformance to policies and performance against the design objectives.

1) Management and Governance in the Cloud: In the white paper from Distributed Management Task Force (DMTF) [8], a series of Cloud Management issues and concerns were identified and set in correspondence with lifecycle aspects of cloud services. Core components of the architecture for clouds management were identified, including monitoring and evaluation, policy and security management, notification management, or key management. Specific interest in analyzing the management interfaces rather than detailed analysis of the cloud management issues exist in DMTF's documents. Being centered around a security-oriented architecture, the architecture identify requirements that are relevant for both management and governance levels.

2) Cloud governance: Even if there is a growing interest in cloud management solutions, little attention was paid to cloud governance. However, progress have been made in relation with relevant data and security concerns, that are most relevant to cloud governance: data and security concerns like data integration, data consistency, policy management and oversights, logging and event management, audit, and others.

Important issues in the context of Cloud governance were identified by Subashini and Kavitha [16], complementing the research challenges, as identified in the paper of Zhang et al [17]. Most of these open issues and research challenges are still active, and emphasize the clear convergence between SOA and cloud governance. They include *open standards interfaces*, *service level agreements* (SLA), *automated service provisioning*, *data security* and security in various service delivery models, *storage technologies* and *data management*, etc.

B. Cloud databases

The impetuous development of Web 2.0, the continuous evolution of software applications together with the need of worldwide presence of the companies on the Internet have led to huge amounts of data that must be stored in order to be further processed.

With the "Big Data" [18] paradigm, limitations of traditional transactional storage systems obeying the Atomicity, Consistency, Isolation, Durability (ACID) principles, and relying on single database were emphasized. Overall performance and scalability of these systems were forced by implementing various techniques for enforcing scalability: clustering, queue-based scalability, sharding, and others.

Because the distributed systems are wanted to be available all the time and also to well behave in case of various kinds of failures, in the most cases the consistency is sacrificed, leading to the *Basically Available*, *Soft state* and *Eventually consistent* (BASE) systems in opposition with ACID systems implemented by the relational databases. The databases created by using on the BASE principles are generally specialized in solving specific problems, with excellent performances while many of them are offered as services in the cloud.

a) Key-Value stores: Amazon's implementation for a high-available key-value store [19] is the reference system in this category. The data model for key-value stores is quite simple, consisting in a map (or a dictionary) where values are assigned to corresponding keys, usually strings. The value is not interpreted by the key-value store, it is only a byte array stored in a form of Binary Large OBject (BLOB) and it is the client's responsibility to give a meaning to that content. The

[1]http://www.itil-officialsite.com/
[2]http://www.opengroup.org/togaf/

operations supported by the key-value stores are limited to a small set: *put*, *get* and *delete*.

b) Graph Databases.: The development of relational networks have led to new concepts in the field of data storage too. Relations between entities are better described through graphs, so it comes naturally to imagine a storage system that is able to store and represent data in terms of graph concepts. Nodes, links an properties are the vocabulary of a graph database. These databases scale horizontally very well, does not enforce rigid schema and some of them offer ACID capabilities.

C. Cloud database benchmarking

Various researches have been conducted in order to analyze the overall performance of different database systems in the context of cloud computing or high performance computing. With these systems in the core of cloud data delivery, the different online transaction processing (OLTP) systems were tested for scalability, elasticity or high availability, with latency and throughput as target characteristics for the different operations under test.

With the growing interest for the noSQL paradigm, particular attention was offered for the noSQL approach in [20], [21]; for the key-value approaches in the context of big-data, in [19], [22], [23], in addition to the graph approaches.

The characteristics of the graphs were considered in [24] for drafting a set of guidelines for graph database management systems (GDMS). However, the work performed in [25], offered the very first results for GDMS, with an approach based on the *Scalable Synthetic Compact Applications* (SSCA) benchmark suite, for symmetric multiprocessors.

With a growing market for graph databases in the context of the big data paradigm, an extensive testing of several database systems, in an industry-like setting, was described in [23].

III. Cloud management and governance in relation with PaaS solutions

A. The mOSAIC approach

The mOSAIC project (Open source API and Platform for multiple Clouds (mOSAIC), was developed in the framework of FP7, and was targeted to ease cloud adoption by offering two independent components acting at cloud resource (the Cloud Agency) and programming level (mOSAIC API).

With the mOSAIC API, a unified approach for a cloud API, customers are able to develop flexible multi-cloud applications based on open and standard interfaces [26]. The functionality of the platform and the associated Cloud Agency were described by several use cases [27], [28].

mOSAIC's Cloud Agency [29], running on top of the Infrastructure-as-a-Service (IaaS), is complementing the services offered by the mOSAIC API – a Platform-as-a-Service solution –, and offers a cloud management solution. It was implemented as a multi-agent system (MAS), designed to solve different tasks like: resource provisioning, in relation with specific cloud vendors; monitoring, performed at level resources; reconfiguration, at IaaS level. Being developed as an independent solution, mOSAIC's Cloud Agency is available to different PaaS solutions, including the mOSAIC platform, by exposing a REpresentational State Transfer (REST) interface. Integrated with a semantic engine, the Cloud Agency also offers features for dynamic discovery and mapping of cloud providers.

B. Cloud Governance Authority

A Cloud Governance solution which is built around mOSAIC's Cloud Agency (CA) in order to allow usage and integration of software "*resources*", provided from the SaaS level, in a similar manner with cloud resources, is specified in Figure 1. The Cloud Agency itself can provide basic activities for negotiation, resource provisioning and monitoring, and solve basic scalability, interoperability and integration issues, as well as assuring and following SLA and QoS, and is acting as a broker between IaaS and PaaS implementations or client applications. Cloud governance provides activities that will enable negotiation, instantiation and commissioning of software services, complemented by management, monitoring, interoperability, and integration at SaaS level, thus relieving the CA of these tasks.

C. Service lifecycle

The service lifecycle support is the primary focus for a cloud governance solution which, together with a cloud management component, can offer a set of complementary services specifically designed to cover all service needs like high availability, scalability, billing, and others. Strong monitoring is thus a requirement and must be present at all architectural levels: at IaaS and PaaS it is covered by cloud management and at SaaS by cloud governance.

D. Cloud Governance and Lifecycle Support

It is essential for a cloud governance system to manage all the phases contained into the service lifecycle. During its lifecycle, the service passes through a succession of general operations namely *Design time operations*, *Provisioning*, *Deployment*, *Execution* and *Retirement*.

There is a strong relationship between the different lifecycle phases and core cloud governance datastores, with a special emphasis on Service Datastore components. These relations are specified in Figure 2, where the phases of a service lifecycle are exposed and their correspondence with the various databases contained in the Governance Datastores are specified. The *Service Datastore* is responsible with maintaining consistent and coherent information about the services lifecycle and every life phase has a corresponding database that traces the service's data.

IV. Datastores for supporting cloud governance

Every datastore inside the cloud governance solution has to maintain vital information for the governance process, fact that involves some special requirements for the composing databases. *Availability* is a very important general requirement as long as every datastore has its relevance for the governing

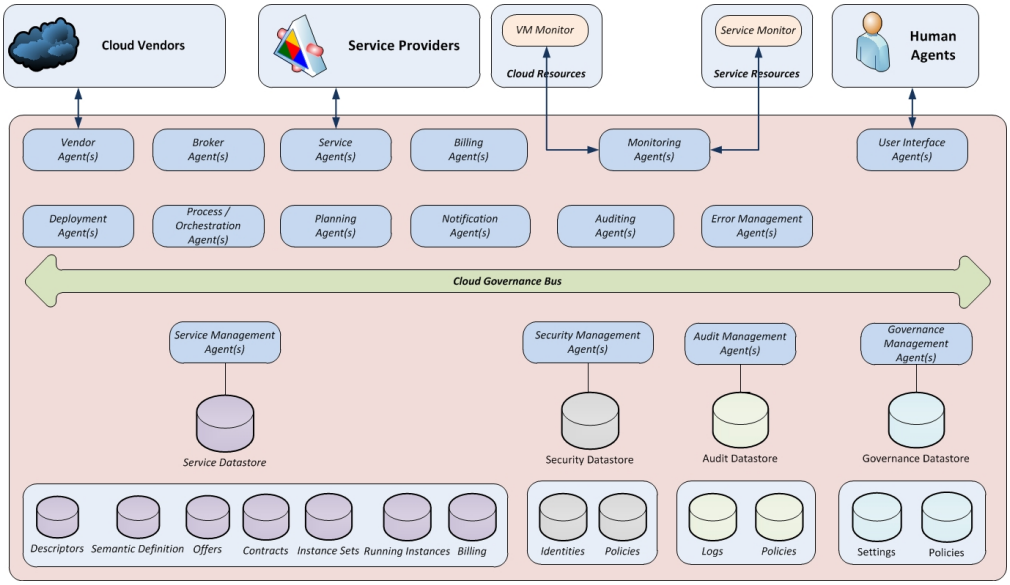

Fig. 1. Overall architecture of the Cloud Governance Authority

process and also because of the strong inter-dependency between all the storage system. *Security* is one of the biggest concern for every entity enrolled in the cloud government environment, this is why the security of the databases plays a crucial role when choosing and implementing the data persistence layer. Other general requirements associated with the databases are related to *high performances*, *cost optimization* and *simple APIs*.

A. Datastores for Cloud Governance

The architecture of the Cloud Governance Authority described in Figure 1 is based on a multi-agent system that is in close relation with specialized storage systems holding critical data for the cloud governance process.

1) Service Datastore: The most complex storage system in the cloud governance architecture, the `service datastore` can be seen as a service for conducting the internal governance processes. It is built around a centralized repository implementing *Service Descriptors* that maintain functional and non functional information about the registered services and exposes core functionality for supporting services discovery.

The semantic definitions for services are located in the *Semantic Definitions* database, also involved in the services discovery process. Every registered service comes with an offer which specifies its capabilities and also the price. Notice that this repository, together with *Service Descriptors*, offers a persistency layer for supporting semantic discovery.

The *Offers Datastore* is responsible with the offers management in the system. After the service discovery process, if the service consumer agrees with an existent offer, it signs a contract for a particular Service Level Agreement (SLA) and the information about the contract is stored in the *Contracts Database*. Registering a service means also adding information associated to the service itself but also related to its dependencies.

When an instantiation request is performed for a service, all its dependencies must be first solved. The information about services and related dependencies is persisted in the *Instances Datastore*. After service instantiation was solved, information about the running service is added in the *Running Instances* database and used to generate data that will be further used in the billing process, for the service consumers. The financial information related to the services is kept inside the *Billing Datastore*.

2) Security Datastore: The access to the services inside the cloud governance system has to be made in a secure way, only authorized entities (people or other services) must be entitled to use them as a whole or based on fragmented rights. The `security datastore` keeps vital information related to users credentials, certificates access rights an so on. In the same time it is involved in the generation of security tokens that will be used in the further access to the cloud governance resources.

3) Audit and Monitoring Datastore: Even the Cloud Management component offers a monitoring functionality, this is theoretically maintained only at the infrastructure level, represented by the virtual machines, networks, storage, etc. Cloud Governance needs a higher level of monitoring directly related to the registered services. All the data collected as a result of service monitoring, together with data that comes from the services audit actions are stored inside the Audit and Monitoring Datastore. They are further uses by other processes to determine if all the services are functioning according with the stipulated SLAs and in the same time to predict possible issues at the high level.

4) Governance/Configuration Datastore: The Cloud Governance system relies on multi-agents that interacts among

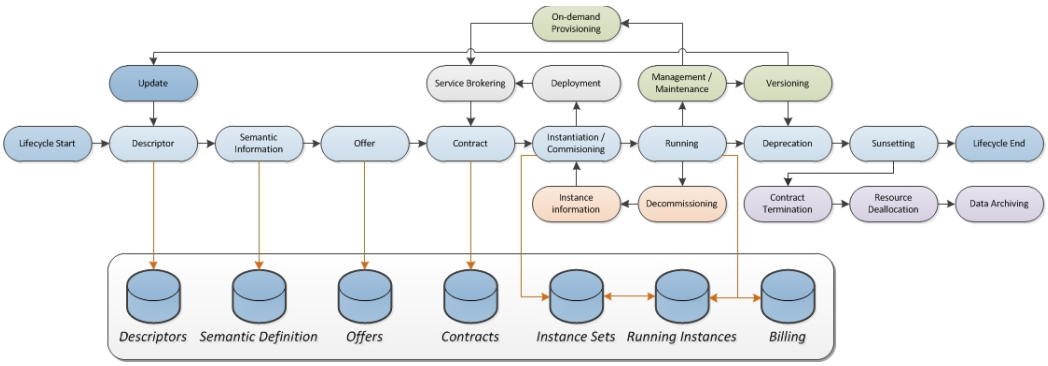

Fig. 2. The cloud service lifecycle in relation with the service datastore

them, with the specialized datastore or with the external environment. These agents run on various virtual machines, having different configurations, which can execute on different cloud infrastructure providers. The number of running agents and virtual machines can scale up an down based on previously established policies. All the information related to the IPs of the virtual machines, configurations, policies are persisted in the Governance Datastore.

V. Benchmarking databases for the requirements of cloud governance

A. Motivation

The amount of data generated during the cloud governance process can highly increase in direct relation with the number of registered services, their dependencies, the system configuration information, security and functional policies together with the data resulted during the audit and billing processes. This situation often leads to scalability issues, new servers must be added in the system in order to accommodate the amount of data and to maintain the performance of the system in the parameters specified by the SLAs.

The cloud governance storage layer is composed by many distinct subsystems, every one of them having different requirements and persisting heterogeneous data, so a predictable process for scaling must be implemented. Taking into account the type and the amount of the data which is expected for every subsystem, a benchmark process must be performed in order to determine the databases performances and to infer an algorithm able to predict the moment in which a database must scale by adding a new node in the system.

B. Chosen Databases

In order to perform the benchmarks, we focused on two representative storage systems from the NoSQL paradigm. For the Key-Value approach, Riak [30] was chosen as the open-source implementation of the well-known Amazon Dynamo [19] database. The graph databases are represented by Neo4j [31] which is one of the most used storage systems based on graph computation.

1) Riak: Riak is the open source, highly scalable, distributed and fault tolerant implementation of Amazon Dynamo [19] database. It is a key-value store, written in Erlang but offering a REST API and also a HTTP Protocol Buffers Client interface.

All the nodes composing the distributed database are equal, there is no one having special meaning, so there is not a single point of failure in the system. The number of replicas for the data is configurable, but for these benchmarks we have chosen to keep the default value, which is 3. Riak can be easily be installed in the Cloud, from sources of from its binary distribution and it is also easy to scale by adding the new node IPs to the Riak process.

2) Neo4j: Neo4j is a NoSQL database, based on the graph paradigm in which the information is internally represented in mathematical graphs but offering an object oriented interface for its clients. Neo4j is schema free, making possible to store various kinds of data in its nodes and also allowing different types of relations between its nodes. According to [24] the graph approach is very appropriate for a wide area of applications like: Social Network Analysis (SNA), proteomics, recommendation system, travel planning and routing, and so on.

C. How we tested

In this section, the testing environment, together with the data sets and benchmark program used to test the storage systems are presented. The tests were performed both on a single machine and in a cluster hosting the databases to be benchmarked.

The benchmark program architecture described in Figure 3 is based on the Cloud Governance Authority architecture (as described in Figure 1) and follows the producer-consumer paradigm. The set of producers resemble the mOSAIC's Cloud Agency while the set of Consumers resemble the agents found in the Cloud Governance environment.

The benchmark software was written in Java, by using the Akka[3] framework to model the multiagent system, and the Cloud Governance bus was replaced by the Apache Camel[4]

[3]http://www.akka.io
[4]http://camel.apache.org/

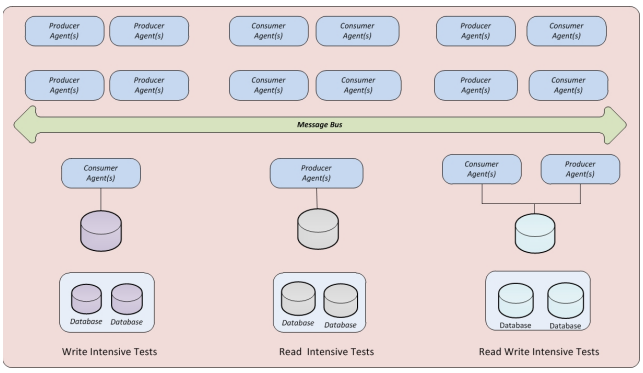

Fig. 3. Cloud Governance Authority - Benchmark Architecture

integration framework. In order to keep the bus as free as possible, message queues implemented in Simple-MQ [32] are attached to the consumers to be able to take very quickly the messages from the bus and process them.

Our benchmarking is focused on typical *write intensive* operations on databases, translated in *put* for key-value stores and different combinations between *read and write* operations translated in node and relations creation and also *traversal* operations for graph databases.

1) Infrastructure: In order to have a consistent and repeatable testing environment, we have opted to instantiate virtual machines based on small instances of Ubuntu Server 12.0 LT, with 1.7GB RAM to which Elastic Store Blocks of 160GB were attached to install the databases to be tested.

2) Databases configuration: The databases were configured keeping the default settings. All of them were installed as services and they run only one at a time. Because we have chosen to access the Neo4j database remotely, we have used the remote server configuration and the REST API exemplified in [31] The basic database operations are prone to a longer execution time than using the Embedded Neo4j [31] version, but it is the only way to access the database remotely. Riak key-value store was accessed through its Java API.

3) Data sets: The data sets used for these benchmarks are randomly generated, but having a parameterizable size, aiming to resemble the expected amount of data in real situations. However we have chosen the amount the data closer to the higher expected limit in order to perform the tests in the worst situation. We are interested to find out how the benchmarked databases respond to various sets of data in a high concurrent situations. Every storage component of the Cloud Governance system have to accommodate with different types and size of data, so an overview of the performances was absolutely necessary.

The key-value stores are populated with records that approximate the expected real information at their upper limits, having 4096 bytes. Finally, a write operation in the graph database inserts a graph having a depth of 10, building also a simple relation between all the nodes, resembling a service dependency.

We have performed tests using various amounts of actors and data sizes, aiming to bring the tested databases performances to their limits. Akka actors are used to simulate the agents living in a multi-agent system.

D. Benchmarking write-intensive operations

This scenario often occurs in the case of service auditing and notifications, when large amounts of information need to be collected from the registered services and stored in dedicated databases for further processing. Having in mind that all the notifications issued by the running services could be very important not only because of their meaning at the time they were raised but also from historical point of view, they must be correctly stored with respect of the issued time and no loses are allowed. This constraint leads to a highly available and writeable database, having time series capabilities, which is a key-value store.

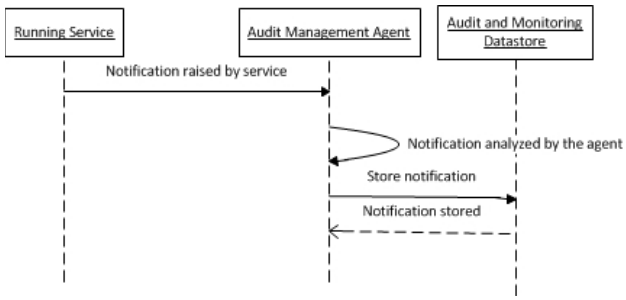

Fig. 4. Storing notification in Audit Datastore

The notifications storage mechanism is depicted in figure 4 which presents all the involved actors. The running service issues a notification which can be related to a simple logging information, or related to an audit action or even an information that will be used in the service billing process. This notification is sent to the *Audit Management Agent* which analyses it and decides in which particular database will be stored. After this analyze is performed, a `store` command is issued to the Audit and Monitoring datastore which will persist the notification and send back a notification stored confirmation to the management agent.

The selected key-value store (Riak) was configured to use LevelDB as back-end due to its strengths related to memory restrictions removal. Figure 5 depicts the case in which a single node holds the entire key-value store. The goal of the test was to determine the moment in which the database becomes saturated and must be scaled in order to offer a linear performance. The saturation is measured based on the *database latency*, which is defined as the time required to perform a certain operation over the database.

Figure 5 presents the situation in which a large number or intensive writes operations were focused on the running Riak database installed on a single node in the cloud. It can be noticed a relative stable latency (approx. 250 ms) until the number of operations reaches 155 000. After that, the latency starts to grow abruptly, indicating the fact that the database begins to be saturated and it needs to be scaled up.

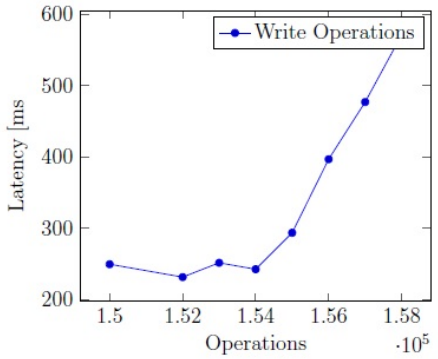

Fig. 5. Write Operations in Riak, single node, having leveldb as backend

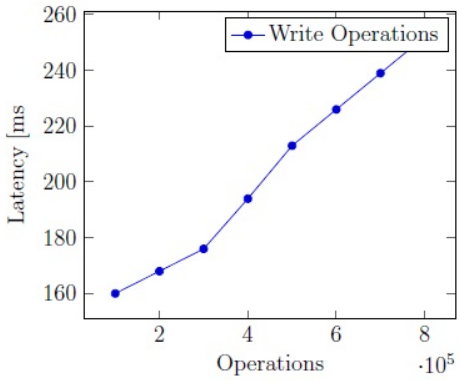

Fig. 6. Write Operations in Riak, cluster having leveldb as backend

Figure 6 presents the situation in which another node was added to the existing Riak configuration, in a cluster architecture. One can notice that in this situation the latency grows with the number of intensive writes, but the 250 ms value is reached only after 800 000 writes which is an improvement compared with the single node architecture.

E. Benchmarking read-write intensive operations

The intensive Read-Write operations are characteristic for the instantiation and service commissioning phase. This is a two-steps operation, primary it is the authentication and authorization, followed by the actual service instantiation.

The use case presented in Figure 7 focuses only on the instantiation phase, considering that the authentication and authorization succeeded. A client performs a service request to the *Service Management Agent* which in turn performs a query over the Running Instances database. If the requested service is found, it is returned to the client, otherwise another query is performed over the Instance Sets database. If the service is found, all its dependencies must be resolved prior instantiation. If all the conditions are met, the service is instantiated, its parameters are stored inside the Running Instances database and its endpoint is passed to the Service Management Agent

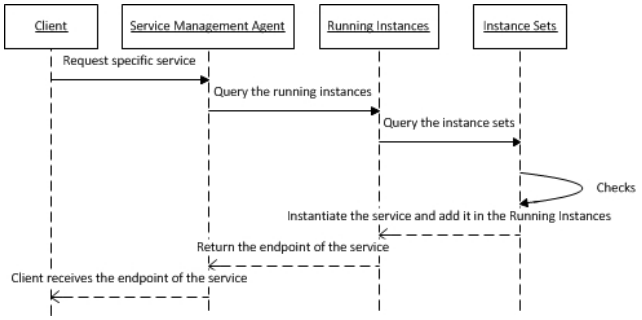

Fig. 7. Instantiation of a service

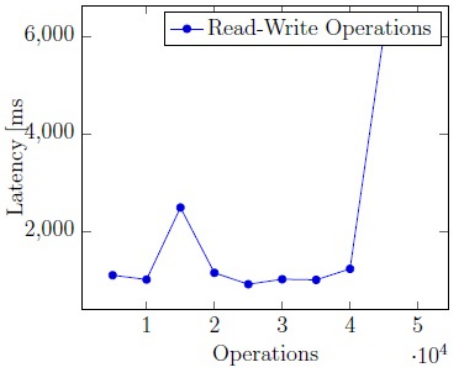

Fig. 8. Write Operations in Neo4j single node

and then to the client, which can start using the requested service.

The tests related to intensive read-writes operations were performed using Akka's actor system provided by the benchmark infrastructure. Even if the expected number of reads will be considerably greater that the number of writes, our choice was to maintain an even proportion between read and write operations.

Figure 8 presents the intensive read-write operations executed over the Neo4j graph database. In this case, the latency was also measured by inserting a graph with 10 level depth and then using a traversal to read information that satisfy some particular conditions. One can notice that the latency is stable until the number of read-write operations reached 40 000, then it grows abruptly, indicating the fact that the scaling threshold was reached.

The tests related to intensive operations were performed using Akka's actor system provided by the benchmark infrastructure. Even if the expected number of reads will be considerably greater that the number of writes, our choice was to maintain an even proportion between read and write operations.

VI. Conclusion

During the last years, Cloud Computing had reached a maturation level that allows companies of all sizes to migrate

their activities into the cloud. This is also the case for the Small and Medium size enterprises, which can now associate and collaborate through virtual enterprises in order to compete with the large solution providers on specialized markets. Managing the virtual enterprises involves cloud governance combined with platform-as-a-service solutions and support for service lifecycle. This paper focused on determining the performances for some cloud databases targeted to support various storage subsystems inside the proposed cloud governance architecture and also to make an idea related to their behavior correlated with the stored data volume.

ACKNOWLEDGEMENTS

The research leading to these results has received partial funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 318484 (FP7-ICT 2011-8-318484 MODAClouds) and Romanian Government grant PN-II-ID-PCE-2011-3-0260 (AMICAS), and refers to a platform developed in the frame of the grant of European Commission FP7-ICT-2009-5-256910 (mOSAIC). The views expressed in this paper do not necessarily reflect those of the corresponding projects' consortium members.

REFERENCES

[1] P. Mell and T. Grance. (2009, July) The NIST Definition of Cloud Computing. White paper. NIST. http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf. [Online]. Available: http://www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf

[2] W. Venters and E. A. Whitley, "A critical review of cloud computing: researching desires and realities," *Journal of information technology*, vol. 27, no. 3, pp. pp. 179–197, 2012. [Online]. Available: http://www.palgrave-journals.com/jit/journal/v27/n3/pdf/jit201217a.pdf

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. (2009, February) Above the clouds: A Berkeley view of cloud computing. Berkeley report,. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/\nullEECS-2009-28.pdf

[4] N. A. Sultan, "Reaching for the cloud: How smes can manage," *International Journal of Information Management*, vol. 31, no. 3, pp. 272 – 278, 2011.

[5] R. Vigne, J. Mangler, E. Schikuta, and S. Rinderle-Ma, "A structured marketplace for arbitrary services," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 48 – 57, 2012.

[6] International Organization for Standardization, *ISO/IEC 38500: Corporate governance of information technology*, ISO/IEC Std. 38 500, 2008, (retrieved Oct. 1st 2012). [Online]. Available: http://www.standardsdirect.org/iso38500.htm

[7] I. Naick. (2004, July) Make autonomic computing a reality with IBM Tivoli. (retrieved Oct. 1st, 2012). [Online]. Available: http://www.ibm.com/developerworks/library/ac-itito/index.html

[8] DMTF. (2010, June) Architecture for Managing Clouds. Distributed Management Task Force. (retrieved Oct 1st, 2012). [Online]. Available: http://dmtf.org/sites/default/files/standards/documents/\nullDSP-IS0102_1.0.0.pdf

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

[10] G. Gruman. (2007, August) Integration issues may hinder saas adoption. CIO Update. (retrieved Oct. 1st, 2012). [Online]. Available: \url{http://www.cioupdate.com/trends/article.php/3695096/\nullIntegration-Issues-May-Hinder-SaaS-Adoption.htm}

[11] 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.

[12] 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

[13] IBM Redbooks, *Service Lifecycle Governance With IBM Websphere Service Registry and Repository Advanced Lifecycle Edition*. Vervante, 2009.

[14] ISACA. (2012) Isaca® glossary of terms. online. (retrieved Oct 1st, 2012). [Online]. Available: http://www.isaca.org/Knowledge-Center/Documents/Glossary/glossary.pdf

[15] C. Rudd, *ITIL V3 Planning to Implement Service Management*, T. S. Office, Ed. Office of Government Commerce (OGC). The Stationery Office, 2010.

[16] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[17] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, 2010.

[18] N. Marz and J. Warren, *Big Data. Principles and best practices of scalable realtime data systems*. Manning Publications, 2012.

[19] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 205–220, Oct. 2007.

[20] R. Cattell, "Scalable sql and nosql data stores," *SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011.

[21] T. Dory, B. Mejias, P. V. Roy, and N.-L. Tran, "Measuring elasticity for cloud databases," in *Proceedings of the The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, 2011.

[22] J. Hugg. (2010, June) Key-value benchmarking. online. VoltDB. (retrieved Oct 1st, 2012). [Online]. Available: http://voltdb.com/company/blog/key-value-benchmarking

[23] T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, "Solving big data challenges for enterprise application performance management," *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 1724–1735, Aug. 2012.

[24] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey, "Survey of graph database performance on the HPC scalable graph analysis benchmark," in *Proceedings of the 2010 international conference on Web-age information management*, ser. WAIM'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 37–48.

[25] D. A. Bader, K. Madduri, J. R. Gilbert, V. Shah, J. Kepner, T. Meuse, and A. Krishnamurthy, "Designing scalable synthetic compact applications for benchmarking high productivity computing systems," *CTWatch Quarterly*, vol. 2, no. 4B, 2006.

[26] 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.

[27] B. Di Martino, D. Petcu, R. Cossu, P. Goncalves, T. Máhr, and M. Loichate, "Building a mosaic of clouds," in *Euro-Par 2010 Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, M. Guarracino, F. d. r. Vivien, J. Traff, M. Cannatoro, M. Danelutto, A. Hast, F. Perla, A. Knapfer, B. Di Martino, and M. Alexander, Eds. Springer Berlin / Heidelberg, 2011, vol. 6586, pp. 571–578.

[28] T.-F. Fortiş, G. Esnal, I. Padillo, T. Mahr, and G. Ferschl, "Cloud patterns for mosaic-enabled scientific applications," in *Euro-Par 2011 Workshops, Part I*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2012, vol. 7155, pp. 83–92.

[29] 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.

[30] Basho.com. (2012) Riak. (retrieved Oct 1st, 2012). [Online]. Available: http://basho.com/products/riak-overview/

[31] NeoTechnology. (2012) The Neo4j Manual. (retrieved Oct 1st, 2012). [Online]. Available: http://docs.neo4j.org/pdf/neo4j-manual-milestone.pdf

[32] Open Source. (2012) Simple-mq. (retrieved Oct 1st, 2012). [Online]. Available: http://code.google.com/p/simple-mq/