CLOUD SAAS INFRASTRUCTURE

Bogdan CÂRSTOIU[1]

Lucrarea prezintă organizarea unei infrastructuri scalabile pentru livrarea SaaS, organizată pe niveluri şi uşor de administrat. După analiza cerinţelor SaaS am propus o infrastructură organizată pe trei niveluri: prezentare, producţie şi stocare. Nivelul de prezentare este cel care primeşte cererile de la clienţi pe care le distribuie nivelului de producţie cu ajutorul algoritmilor de nivel OSI 4/7 astfel încât încărcarea sa fie echilibrată. Software-ul de pe nivelul de producţie procesează cererile primite, cereri ce pot impune accesul la nivelul de stocare. Pentru nivelul de stocare s-a proiectat şi implementat baza de date Zatara, care prin organizarea nodurilor de stocare în grupuri asigură redundanta datelor. Performanţele Zatara scalează liniar cu numărul de calculatoare pe care rulează. Sunt prezentate o serie de teste ale bazei de date Zatara.

This paper describes a distributed, scalable, manageable, layered organized SaaS infrastructure. After reviewing SaaS requirements we designed an infrastructure organized in 3 abstracted layers: Presentation, Production, and Storage. The Presentation layer receives requests from customers and distributes them to the Production layer using OSI 4/OSI 7 level load balancing algorithms. The received requests are processed on the Production layer by the application software logic. In order to store information, the software running on the computers on the Production layer uses the Storage layer. On the Storage layer we introduce Zatara, our distributed database implementation that guarantees data redundancy using group replication. Zatara performance scales linearly with the number of computers it runs on. Several benchmarks of Zatara are presented in the paper.

Keywords: SaaS infrastructure, cloud computing, clustering, software provisioning, Zatara, load balancing

1. Introduction

To implement a Software as a Service (SaaS) infrastructure it is required to use cluster and grid technologies. While these technologies provide a hardware level implementation of the infrastructure, they only marginally cover the software running on the hardware platform. SaaS customers do not buy the infrastructure, but the software service that fits into their requirements, in most cases they are not even interested in the physical infrastructure.

[1] Assist., Control and Computers Faculty, University POLITEHNICA of Bucharest, Romania, e-mail: bcarstoiu@gmail.com

The main advantage of a cloud service is the fast resource scaling based on demand. These applications process large amounts of data, but are quite different to the software used in the cloud (e.g. Salesforce, Gmail). Modern cloud applications usually involve a web based interface and are able to answer to requests very fast, in several hundred milliseconds. The number of requests received from customers is also high, can be tens of thousands per second. The traditional enterprise software used by organizations of all sizes might require significant computational resources, but in most cases it does not run distributed on multiple computers.

Analysts predicted that cloud computing will bring major economical and social changes. A report published by the European Network and Information Security Agency (ENISA) highlighted the benefits and shortcomings of the cloud technology, but also made recommendations for cloud providers on how to minimize the risk of losing data. The report also evaluated the global financial impact; the turnover of the cloud services reached US $17.4 billion and will increase to US $44.2 billion in 2013 [1]. Cloud computing is a method to easily recover the financial investment and to reduce risk. It enables organizations to focus their attention and resources on the business instead of managing the vast technological resources. [2].

The Financial Services Club Autumn report entitled "The Impact of Cloud Computing on Financial Services," sponsored by Cisco and contributed by more than 230 finance specialists outlined the following facts [3]:

- Increasing use of cloud in financial institutions;
- Increased level of operational efficiency due to the cloud;
- Organizations have valid concerns regarding the security of the data stored by third parties;
- Cloud needs investments in SaaS and security;
- A market analysis suggested that over the next 3-5 years, the investment in security will be on the second place after SaaS investments;
- 56.9% of the IT service providers are turning to cloud computing.

Even during the financial crisis the cloud market was growing and created new opportunities such as cost reductions, lower investments in IT infrastructures and cut on unnecessary resources [4]. By using commercial and open source solutions organizations can build and manage a computing cloud, even if its scalability is lower than the ones developed by vendors like Amazon [5]. For most companies such a cloud satisfies all present and future demands.

2. Related Work

There are no major technical issues in building cloud architectures with tens or evens thousands of nodes, software packages like Eucalyptus can be

successfully used. NASA, one of the leading users of Eucalyptus, announced an alliance with Rackspace, Dell, AMD, Citrix and other vendors to build a software framework that enables companies to deploy cloud infrastructures quickly and with minimal costs [6]. NASA's initiative is based on Eucalyptus' current lack of flexibility. Eucalyptus cannot scale to millions of nodes and its architecture cannot be changed as it is not entirely open source.

By analyzing many successful web applications, we noticed the high scalability provided by these applications. The scalability was achieved after a relatively long time and with many efforts, because developers had to discover and learn new concepts [7]. Usually, scalability is an attribute of the application and it is difficult to build a generic framework for scalable infrastructures without knowing important facts such as: usage scenario, used technologies, interaction with third party applications, communication protocols, etc. A company that heavily uses the cloud is Twitter. Its user base grown faster than expected and the service went through difficult times during peak hours. If we investigate the way how Twitter works, we discover that it is basically a message exchange that could have used a standard protocol like AMQP. Obviously, the scalability level would have been limited to the one offered by a distributed AMQP server. Twitter chose to implement a proprietary messaging protocol from scratch, with its server and the necessary libraries.

Another company that heavily invested in its own software tools is Google; beside its search algorithms, Google implemented a distributed database that stored information gathered from the web. Compared with Twitter, that can start some debates on the opportunity of designing and implementing a proprietary messaging protocol, it is crystal that Google's success is based on its technology.

Twitter and Google are specialized applications, with a limited number of usage scenarios. Facebook is a multipurpose application that can also be extended by third party developers. Facebook development started using the MySQL database and PHP scripting language. The most important technical issue that had to be solved when the number of users grown was that the database could not scale. It was very difficult to replace the database, so they decided to introduce a caching layer to keep available the information most frequently requested from the database, in a similar way how the CPU's cache works. For this purpose Facebook chose Memcache, an open source program that keeps data in memory, responds to requests faster than MySQL and can be distributed on multiple nodes. Memcache cannot store complex data structures, only simple structures like key-value pairs. Regardless of how information is retrieved from the SQL database, it is added by the program logic in Memcache to be available there for the next requests[8] and the number of requests to the SQL database was reduced by

almost 90%, allowing the infrastructure to scale at a lower cost [9]. Memcached is distributed by design, so it is easy to add new servers to the cloud.

Gartner estimated global profits generated by using cloud services at US $63.3 billion in 2010 and US $150 billion in 2014, while spending for SaaS, PaaS and IaaS during the next five years is forecasted at US $112 billion.

3. The Requirements of SaaS Service Delivery Infrastructures

Scientific applications process large amounts of information, require a lot of computing power and high storage capacity for input, output, intermediate or final data. These applications do not provide an immediate response, in most scenarios it is acceptable to deliver the result after several hours/days or more [10].

SaaS web applications must be able to deliver results immediately following an interrogation. A SaaS service delivery infrastructure differentiates by a number of characteristics and requirements that are structured according to the role played by the observer (user, administrator, etc) [11].

- User perspective: quick access to information, resource availability, access to latest software features, stored information security, transaction security [12], [13].
- Administrator perspective: easy scalable by adding new machines, rapid detection of infrastructure failures, fast replacement of damaged machines, monitoring capabilities, automatic reorganization of the infrastructure on failure, upgrade software without service interruptions, ability to statistically determine when and how the infrastructure can fail or lose data [7], [5].
- Developer perspective: separation of the application code from the infrastructure code, availability of structured and unstructured storage services, availability of communication services in the infrastructure, infrastructure complexity not exposed to the application. [11].

There are also economical and business requirements such as: provider's independence from the hardware vendor, customer independence from the cloud provider, proprietary software independence, resistance to disasters, and predictable operating cost [14].

Developers have major issues with cloud infrastructures because they have to accommodate the existing software to the new infrastructures or to write new scalable software without any previous experience in distributed applications [7].

4. SaaS Cloud Architecture

The cloud is built using virtualization because it provides the necessary methods to divide resources and to adjust them in real-time. In most cloud implementations nodes are virtual machines, abstracted through the virtualization

technology. Virtual servers also provide resource control features, which is very important especially when powerful hardware machines with many processing units are used. This logical structure of the infrastructure allows a good scalability, reduces maintenance costs, and enables the application's integration in the hardware [16], [5]. The infrastructure is divided in three different layers, which will be detailed below:

- **Presentation layer** – on this layer we have the nodes that take the requests of the clients and distribute them to the Production layer. There are utilization scenarios that allow the presentation nodes to answer to requests directly instead of forwarding them to the Production Layer.
- **Production layer –** nodes on this layer implement the application logic. This logic is responsible for handling the requests received from the Presentation layer. This layer can auto-scale to handle the load during peak times. The application know-how can be found on this layer.
- **Storage layer**- is responsible for data storage. It acts as a service for the Production layer. It is a vital component of the infrastructure because only at this level information can be lost or tampered.

To meet these requirements we implemented a framework responsible with the automation, management and monitoring of the infrastructure. It also implements a semi-structured, distributed storage layer.Fig. 1 illustrates how a request from the client reaches the Presentation layer, and then it is distributed to the Production layer where it is processed by the software.

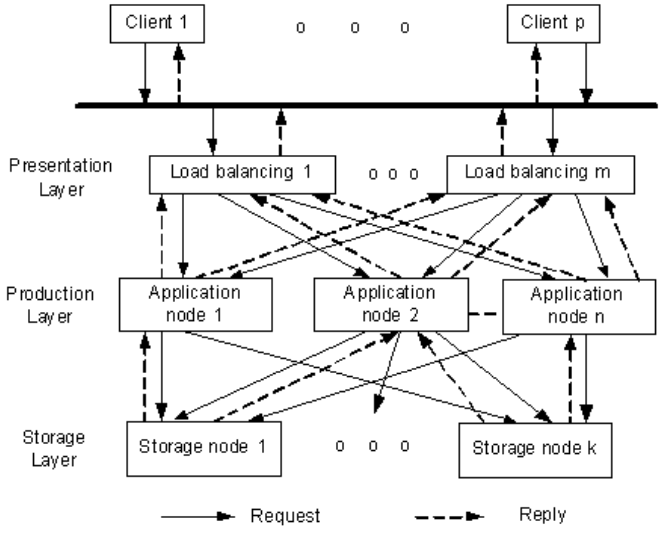

Fig. 1. SaaS Cloud Architecture

The answer to the request is then served to the customer.

4.1. Presentation Layer

Nodes on this level are able to process a large number of requests because all operations are very fast as they do not require important computing resources. A single node is able to distribute requests to tens or even hundreds of node on the Production layer. To dimension the Presentation layer it is necessary to know the number of requests received by the infrastructure and the capacity of the nodes on this layer. Scaling can be performed while infrastructure is operating and without service interruptions. There are several techniques used by the nodes on this layer to perform request distribution and load balancing. While there are many differences between them, most result from the OSI layer they operate on:

- **Layer 4 (transport)** – this is considered the least sophisticated technique, but it also consumes less resources. The Presentation layer node distributes the requests received on a specific port and protocol to the Production layer nodes. For some higher level protocols, problems can occur with this type of distribution because the requests need to preserve session information. This type of distribution can be implemented in hardware devices that use specialized processors designed for this purpose (there are devices capable of distributing up to 10 Gbs).
- **Layer 7 (application)** – the logic used to implement this technique is more complex. For example, in order to distribute HTTP requests we might use a specialized server for load balancing, a reverse proxy, a redirect server etc. Implementation is done using session cookies, browser applications recognize them and assigns them to a session. By using this method requests from the same client will get distributed to the same node on the production layer. For the SIP protocol the distribution is performed by a proxy server that can hide the complexity of the Production layer [17]. This type of distribution requires specialized software logic for each protocol. Their performance is moderate because there is a lot of processing involved in order to perform the distribution.

Both distribution techniques can be combined and used together, depending on the application. In the last years the presentation level implementation has not undergone dramatic improvements. Load balancing appeared with the first successful websites in the '90s. Initially, distribution was done mainly using dedicated network equipment, because regular servers were not powerful enough to handle the traffic. The dedicated load balancing devices are complex and expose programmable APIs, so they can be controlled by third party applications [18]. Unfortunately these solutions are not very flexible in configuration. For example, there is no problem to implement a cloud with one node on the Presentation layer and several nodes on the Production layer, but adding new nodes and removing existing nodes is a difficult and non-automated

process, especially on large implementations with thousands of nodes. Also, many implementations consider that a single node on the Presentation layer to handle all the traffic is enough, which is not correct for all deployments. When we deal with multiple nodes on the Presentation layer, the distribution of requests between them is usually performed using the DNS service. The DNS does not know anything about the availability of the Presentation layer nodes, and even if notified, it would be unfeasible to change DNS records due to client record caching. A more effective solution is for a note on the same layer to take over the IP from the failed machine and to distribute requests on its behalf. The operation is implemented using gratuitous ARP.

Choosing the most appropriate solution for the Presentation layer depends on the project, that's why the framework is opened for any distribution solution. Also it is possible to incorporate in the framework the services that can be used to identify the load on each Production layer node, in order to help the balancing algorithms perform a fair distribution.

4.2. Production Layer

Production layer does not involve special complications in terms of infrastructure. On this layer there are nodes that reply to customer requests and implement all the logic necessary to answer to these requests. This layer is the least affected by the infrastructure organization because it receives the requests forwarded by the Presentation layer and uses the services made available by the Storage layer. However, there are a number of particularities and differences compared to traditional architectures: more nodes answer to requests in parallel, the communication between the nodes on the Production layer should be avoided, access to storage structures is unified, the layer is isolated from the outside, it can use partitioning, can be management by a third party, etc.

4.3. Storage Layer

In cloud computing this layer brings the most important challenges. The huge amount of information accessible in real-time, the security constraints and the high availability are important issues that have to be addressed by the industry.

Traditional web applications use the structured storage offered by SQL relational databases. Unfortunately these databases are not capable to scale linearly on multiple servers. Due to the new wave of online applications that demanded more performance than SQL databases can offer, major changes on how structured information is stored and accessed are underway. The most important weakness of relational databases is their inability to scale horizontally. There are several expensive to license and implement databases which are able to scale vertically, but only up to a point given by the hardware platform it runs on.

Modern applications started to avoid using SQL structured databases, but there is still a huge demand on relational databases. This is satisfied by SQL as a service cloud offers, although such offers are limited to a maximum number of transactions per unit time [7]. The SQL provided as a service on the Storage layer might be highly available, but it cannot scale. Such services are usually implemented on virtual machines, in a very similar way to traditional hosting (examples: Amazon RDS MySQL, PostgreSQL, Heroku).

In order to deliver truly scalable services, providers consider other solutions. There are alternate systems that provide important advantages in terms of availability, speed and scalability, but with the price of sacrificing consistency. Google BigTable [19] is a non-relational, distributed database that was used as inspiration source for some open source solutions. Although cloud databases differ from each other, they share common characteristics: they expose a low number of operations, they do not have JOIN operations, the storage format is not column oriented, the information is distributed on multiple nodes and localized by the client application, etc.

Many cloud software applications use local storage, but the cloud storage layer should provide support for both local and remote storage via the Internet. Therefore, it is expected that a cloud storage service to be very attractive for many applications. Cloud storage platforms implement different techniques and answer to different needs. For example, Amazon's Simple Storage Service (S3) provides an unstructured storage facility accessible using the Internet. The model given to developers is simple: it allows storing objects as binary content in configurable containers. Software applications can create, read and delete objects and containers. Objects can also be updated, but in reality they are completely replaced. This simple, but limited storage service is more easily scalable than those that provide advanced functionality. Application developers have access to cheap storage space in the cloud, but they must work harder to use it.

Cloud information can be stored in structured or unstructured format. Structured information is strictly formatted and it is stored in such a way that various operations can be performed on it (eg: XML / XHTML, SQL databases). Structured information is organized to identify and separate content from context information [18]. Unstructured information is not organized in a way to allow any type separation. From unstructured information it is not possible to automatically extract properties and relationships. Unstructured information is the generated content such as audio, video, graphics and documents. Each type of content, structured or unstructured has different particularities when it comes to how storage and search operations are performed.

Our framework uses the distributed database Zatara, which was designed to scale horizontally with the number of nodes it is running on. Nodes are organized into groups and each node has assigned a NodeID and a GroupID. The

NodeID is represented by a 32-bit unique string, while the GroupID is represented by a 16-bit unique string. (Fig. 2). When working with a fixed number of hardware resources, it is important to organize nodes in groups to achieve maximum performance and data reliability. The database can store both persistent and caching only data, depending on the way how the customer wants to save an object. Caching mode does not guarantee data persistency, but it is useful for many applications that have to store information only for a limited time and they do not care when the information is lost. The persistent information stored on a node is replicated within the group to ensure redundancy. It is not recommended to group more than 4 nodes in a group.

To test and integrate Zatara we wrote a C client library that exposes an API available to software developers [20].

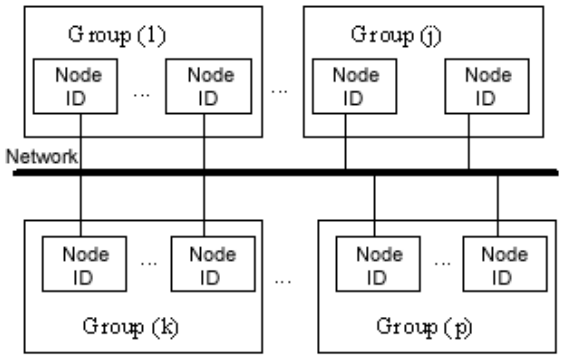

Fig. 2. Nodes organized in groups

Although the framework design is modularized and allows using any database system, Zatara offers important advantages [21], [22]. In designing Zatara, the biggest challenge was finding solutions to various compromises generally accepted by the designers of distributed databases, but hardly tolerated by application developers. An analysis of the existing distributed database implementations revealed that each implementation has particular strengths, but also notable weaknesses. A common weakness of most distributed databases is that many were designed with a specific software application in mind [23], thus affecting the database capabilities to fit into a more generic ecosystem. It has been already proved that a consistent, fault tolerant and scalable database cannot exist, but we consider that a single software application would want to store objects with different particularities. The developer should be able to choose how the object must be stored in the database [24]. The database speed is also a very important factor, even if the database is distributed a single node should be able to answer to many thousands of requests per second.

5. Zatara Perf*ormance*

Evaluating the performance of the storage layer is difficult because it is necessary to send enough requests to the database server in order to saturate its input. While this is possible on the same machine, it might be difficult over a shared network. In order to evaluate the performance, we used two Amazon EC2 Small Instance with 1.7Gb of memory and one EC2 processing unit. An instance has installed the test program, which is using the Zatara client library and the other instance is running the Zatara server. In the testing session we ran 20 consecutive tests in the same conditions, the worst 15% results were eliminated and we computed the average of the remaining 85% results. All operations performed on the database level have O(1) complexity. Should be noted that the software interaction with the database is usually blocking, this means that the software issues a request and waits for the answer to come from the database server. That's why the speed of the database directly affects the application response time. Testing was performed on caching type objects in order to avoid group replication. The performance is almost identical when storing persistent objects because information is committed to the disk asynchronously. Zatara always keeps caching type objects in the server RAM.

5.1. Storing Simple Objects

Using this test we can quickly evaluate the store operation speed. The names of the objects were generated by the client using a determinist algorithm, these are unique, 12 characters long. We evaluated the performance of the operation in two test sessions, in one storing 20 byte values (Fig. 3) and in the second one 64 byte values (Fig. 4). The results are not significant different, the network layer is also playing an important role on this test.

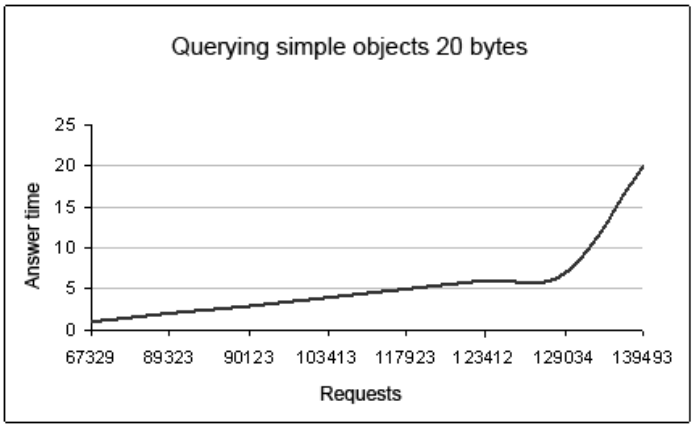

Fig. 3. Evolution of the response time with the number of store requests (simple objects, 20 bytes)

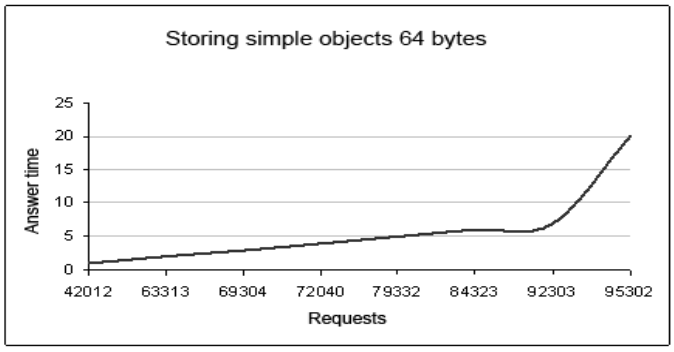

Fig. 4. Evolution of the response time with the number of store requests (simple objects, 64 bytes)

5.2. Query Simple Objects

On this test the objects stored in the previous test were retrieved from the database. The test is once again performed for 20 byte (Fig. 5) values and 64 byte values (Fig. 6).

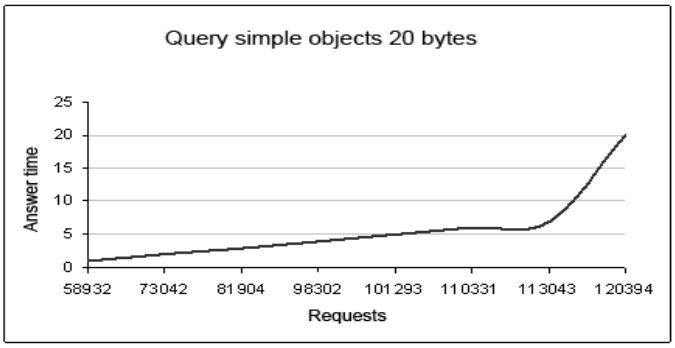

Fig. 5. Evolution of the response time with the number of retrieve requests (simple objects, 20 bytes)

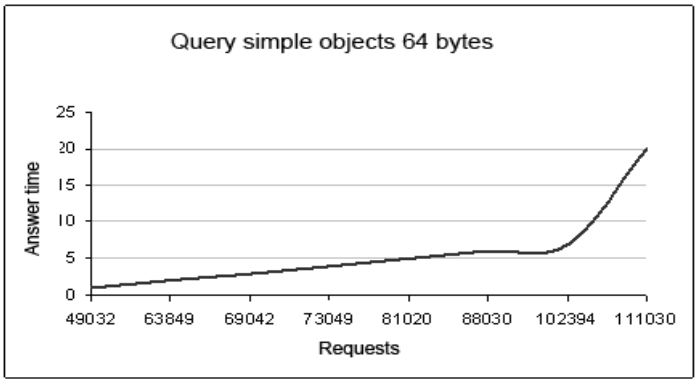

Fig. 6. Evolution of the response time with the number of retrieve requests (simple objects, 64 bytes)

5.3. Storing Complex Objects

On this test we evaluate the speed of the store operation on arrays. Arrays contain 512 elements and have unique names of 20 bytes. For each operation of storing information in the array the client sends a request to the database. Each element in the array gets a value of 20 bytes length. The results of the test can be found in Fig. 7.

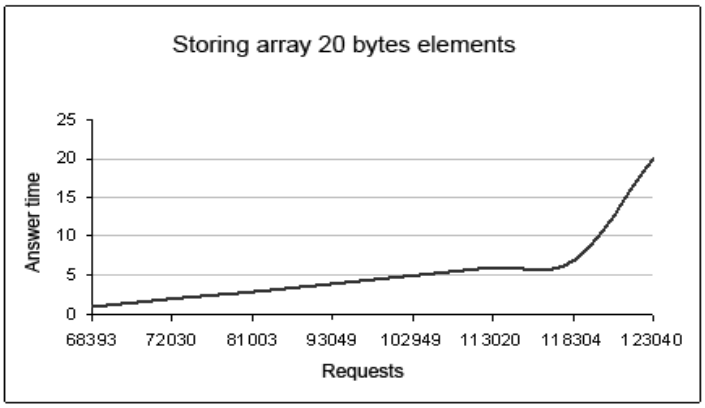

Fig. 7. Evolution of the response time with the number of store requests (complex objects, 20 bytes)

5.4. Query Complex Objects

On this test the client randomly queries the elements of the arrays. The names of the arrays are known; in each array the client will interrogate 128 elements. These are the elements previously stored, each one is 20 bytes. Results can be found in Fig. 8.

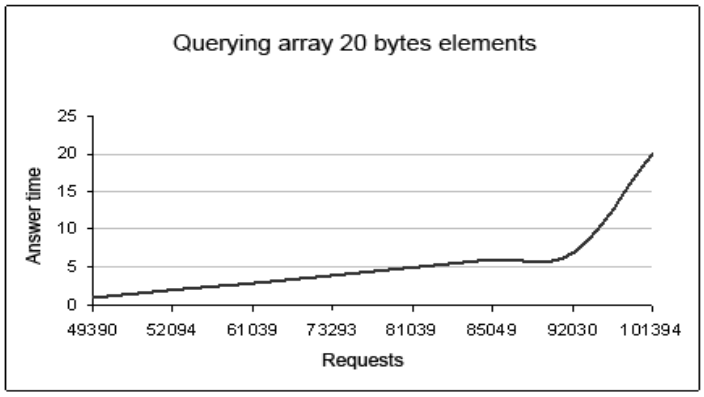

Fig. 8. Evolution of the response time with the number of query requests (complex objects, 20 bytes)

The previous tests revealed that there is no notable speed difference between the operations performed on simple and complex objects, when they deal with values of the same size.

6. Conclusions

From the infrastructure organization point of view, it does not matter what operating system is used on the nodes or what hardware architecture is involved. It is recommended to deal with virtual machines as nodes because they offer obvious advantages such as easy migration, centralized management, and better usage of physical resources. The logical organization of the infrastructure allows achieving good scalability, low maintenance costs and simplified application integration with the hardware infrastructure. The tiered organization with a presentation, production and storage level offers an abstraction level useful for infrastructure design and application developers.

The OSI layer 4 and 7 distribution methods allow requests to be distributed by the Presentation layer to the Production layer. The framework is flexible and does not impose any limitations on the methods used to balance the requests to the software logic. From the developer point of view, they are dealing only with the software implemented on the Production layer and they use the services provided by the Storage layer. Zatara, the distributed database designed to be used in the framework is flexible enough to be used by any software application, guarantees data integrity and achieves high performance and scalability. The storage classes implemented by the database and the online replication are innovative approaches applied for the first time on a cloud database.

The tests performed on Zatara using the Amazon cloud infrastructure revealed a good performance. The database is able to answer to almost 100k queries per second on a node with a basic hardware configuration. There are no notable performance differences when complex or simple objects are used.

R E F E R E N C E S

[1] *** Benefits, risks and recommendations for information security, raport ENISA. http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment/,

[2] *John Leese*, The Financial Benefits of Cloud Computing, http://www.the-financedirector.com/features/feature61743/

[3] ***http://thefinanser.co.uk/fsclub/2009/10/cloud-computing-needs-better-definition-to-succeed.html

[4] *Andrew R Hickey*, Cloud Computing, SaaS Boom Fueled By Recession, 2010, http://www.adaptiveplanning.com/docs/Cloud-Computing-SaaS-Boom-Fueled-By-Recession_CRN-Channel-Web.pdf)

[5] *K. C. Li, C. H. Hsu, L. T. Yang, J. Dongarra, H. Zima*, Handbook of Research on Scalable Computing Technologies, ISBN: 978-1-60566-661-7.
[6] *** Eucalyptus System http://www.eucalyptus.com/
[7] *Cal Henderson*, Building Scalable Web Sites: Building, Scaling, and Optimizing the Next Generation of Web Applications, ISBN: 0596102356.
[8] *Jure Petrovic*, Using Memcached for Data Distribution in Industrial Environment**,** Proceedings of the Third International Conference on Systems, Pages: 368-372, 2008, ISBN:978-0-7695-3105-2.
[9] *J. Sobel,* Needle in a Haystack: Efficient Storage of Billions of Photos, http://perspectives.mvdirona.com/2008/06/30/FacebookNeedleInAHaystackEfficientStorageOfBillionsOfPhotos.aspx
[10] *Edward P. Holden, Jai W. Kang, Dianne P. Bills, Mukhtar Ilyassov*, Databases in the cloud: a work in progress, **SIGITE '09:** Proceedings of the 10th ACM conference on SIG-information technology education, October 2009.
[11] *Wil A.H. Thissen*, *Paulien M. Herder*, Critical Infrastructures: State of the Art in Research and Application (International Series in Operations Research & Management Science, ISBN: 1402076010.
[12] *D. Amerheim*, et all Cloud Computing Use Case White Paper, 2009.
[13] *A. Weiss*. Computing in the Clouds. *netWorker*, 11(4):16-25, Dec. 2007.
[14] *Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal*, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia.
[15] *George Reese*, Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, ISBN: 0596156367.
[16] *Richard Chow, Philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, Jesus Molina*, Controlling data in the cloud: outsourcing computation without outsourcing control, Proceedings of the 2009 ACM workshop on Cloud computing security, Chicago, Illinois, USA, Pages: 85-90, 2009, ISBN:978-1-60558-784-4.
[17] *Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler*, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
[18] *K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman*. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169–182, 2005.
[19] *G. DeCandia, D. Hastorun, et al*, Dynamo: Amazon's highly available key-value store. In Proceedings of the 21st ACM Symposium on Operating Systems Principles (Stevenson, Washington, October 2007)
[20] *** Linux C and C++ Programmer's Guide http://www.comptechdoc.org/os/linux/programming/c/linux_pgc.html.
[21] *B. Carstoiu, D. Carstoiu*, High Performance Eventually Consistent Distributed Database Zatara, 6th International Conference on Network Computing, May 2010 Koreea, IEEE PDF files ISBN: 978-89-88678-19-0, IEEE Print version ISBN: 978-89-88678-20-6, pag 54-59.
[22] *Bogdan Carstoiu, Dorin Carstoiu*, Zatara, the Plug-in-able Eventually Consistent Distributed Database, Advances in Information Sciences and Service Sciences ISSN 1976-3700.
[23] *E. A. Brewer*, "Towards robust distributed systems." In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*, July 2000.
[24] *Bogdan Carstoiu, Dorin Carstoiu*, Web4Desktop, a Framework for Improving the Usability of Web Applications, pag 455-464, , CENTERIS 2010, Springer, Heidelberg.