DATABASE LINEAR OF SCALABILITY AND HIGH AVAILABILITY WHILE MAINTAINING A SYSTEM PERFORMANCE

Dr Djordje Dihovični[1]; Dr Miroslav Medenica[2]
[1] Technical College of Professional Studies, Belgrade, SERBIA, dj_dihovicni@yahoo.com
[2] Technical College of Professional Studies, Belgrade, SERBIA, medenicam@yahoo.com

Abstract: *Every day increases the amount of data that is stored in data management systems (DBMS), which leads to the development of new solutions in order to bid farewell to the needs dictated by Big Date. Distributed systems are existing methods for storing large amounts of data to a new generation of web applications used by large corporations like Google, Amazon, Facebook, Twitter, Yahoo, etc. The need for very low-latency, linear scalability, global data distribution, constant availability and reduced operational costs and software led to the categories of NoSQL databases. Apache Cassandra falls into this category and is an open system for database management, designed to handle large amounts of data, which provides continuous availability, linear performance and easy distribution of data in multiple data centers. This paper gives an overview of Apache Cassandra on the part that is essential for distributed DBMS in the business system, provides a paradigm where it is used by large internet companies, characteristics, available tools and ongoing research to improve performance.*

Ključne reči: *NoSQL database, scalability,availability, model, safety*

1. INTRODUCTION

Nowadays, mainly when it comes to databases, it is referring to the relational database designed on a relational model in which separate logical and physical data structures, resulting in a physical data independence. Generally, the system for database management is software support that enables the creation, updating and administration of databases, which can be realized in communication with users and developers. Relational databases are subject to ACID standards that represent a set of features that guarantee reliability of transaction databases through:
atomicity, consistency, isolation and durability.
The tradition of relational database of nearly 50 years has made a habit that they are mostly used in data processing and in the main is set when it comes to creating databases.
The main language of the database is certainly inevitable SQL (Structured Query Language) language which is used for data management. It includes data entry, query, update and deletion, schema creation and modification, and data access control, etc.
Today, due to the development of technology, and because of the increasing number of users and the need for information leads to an enormous increase in the amount of data, which is particularly evident popularization of the Internet? It is introduced software Big Data under which is usually considered a large amount of structured and unstructured data that cannot be managed by conventional functional tools for data management. Ability to collect, search and correlations within a given data set characteristics is that characterize the Big Data paradigm. Volume, variety, velocity, and 3V model is used for description.
This enormous amount of data created difficulties in the database system to be accepted, processed and delivered in the shortest possible time.
It should always bear in mind that the system is as fast as the slowest part of his fast. The slowest part is often the database itself, and when we talk about relational fully trust, with all its large number of applications that support them, it should also include today's demands, such as scaling, distributing data across multiple nodes, etc..
This leads to a new type of database in which data is not organized according to the principles of relational models of care, according to much simpler and "freer" data model that does not require schema definition data. Over time, as the number of users of our systems grows, we have a greater load on the database and begin to think about solutions to scale the database system.
The rapid development of the Internet, especially the emergence of Web 2.0 and social networks that bring together a huge number of customers, including leading Facebook with over 500 million users, has been influencing the shake of SQL and seek alternative and innovative solutions. The answer to the question that will be the solution to use for storage will never be easy and there will be always more than one. In recent years have appeared NoSQL systems that are inspired by some of the shortcomings that have relational databases, and try to solve some of the major shortcomings of relational databases, as well as to specialize in a particular application domain. Although it brings some

advantages, it also brings certain drawbacks with it. In practice they meet applications that use more than one storage solutions. The reason is that each function requires special analysis and solution.
NoSQL databases are not trying to meet all the requirements and will not sacrifice performance for the sake of options. This means that the NoSQL solutions will not be respected for ACID transactions. This does not mean that it is worse than NoSQL SQL but is designed in a different way, and that has to change the way of thinking in the design and programming DAL layer applications.
Below will be given the deficiencies that have certain types of NoSQL databases, on the other hand we are not able to accept it in certain domains. So even relational database nor NoSQL are not the perfect choice, it must be viewed in accordance with the current requirements of the application and users should choose the technology that they prefer, or they can get the best of both spheres databases. On the other hand it is clear that NoSQL de facto become the standard when it comes to projects where on a daily basis deal with enormous amounts of information that are measured in terabytes.

2. NOSQL DATABASE

Historically the name NoSQL first gave Carlo Strozzi in 1998 to open a database that had nothing in common with the "NoSQL" in its present form. Then it was set up as an open database, which kept all the data as an ASCII file and use "shell scripts" instead of SQL data access.
In 2006 Google has given new solutions in this field by creating a system capable of performing parallel processing of large amounts of data. It is a "Bigtable" distributed system storage and data management, designed to be scaled up to high levels, where it is the petabytes of information over thousands of general computer applications. Google generates these first no relational database columns by model. In 2009 in San Francisco Johan Oskarsson organized a meeting where it was discussed about new technologies in the market of IT, storage and processing of data. The main stimulus for the meeting was new products like BigTable and Dinamo. For the meeting, it was necessary to find a short name for use as "Twitter hashtag". The term "NoSQL" suggested Eric Evans from Rackspace. Dat title is planned to be used only for this meeting and did not have deep meaning. Later it spread all over the world such as internet marketing and it turned out to become "de facto" the name of the trend in IT.
Name "NoSQ" has absolutely natural origin, from a relational database, but it must be said that there is no universally accepted definition. But we can say that NoSQL database that is not based on the relational data model or at least is not holding firmly, is easy to distribute and horizontally scalable. NoSQL is born from the need for processing large volume of data that have led to fundamental shifts in the large set of hardware platforms, and on the other hand it has created difficulties in the application "solid" relational database, and it has big problems in making the application code with the model of relational databases.
The reasons for the introduction of NoSQL databases, among other things, are that the relational database takes up a lot of resources, i.e. a relatively high price for non redundancy reading together caused to change the structure because of the connection structure with the use and optimization.
NoSQL databases implement different mechanisms for storing and linking data, unlike relational databases. If the data do not need to be stored in data tables or there are links that cannot be represented by classical SQL distances and without the need to quickly access the data, then there is need for using NoSQL database.
So the main benefit of NoSQL database is that it is very good for that for which they are intended. There is no need to force NoSQL database for things that are not intended.
An important feature of NoSQL databases is that they are designed for distributed usage - while relational databases are designed to work on a single computer. NoSQL systems are designed so that several of them are distributed to a larger number of computers where each working with its own set of data, which significantly speeds up the acceptance and processing of data. They support horizontal scalability, which represents the ideal solution for large systems. By adding new nodes, representing the server, the system continues to operate as before with more power and a new node in the chain.
Web applications today are faced with many problems, among other things; data processing is difficult, because users are always modifying data. The load is uneven, unpredictable increase of extremely large dynamic data, constantly adding new features, changes in existing components and etc.
Although the concept may resemble a relational database in the work setting relations, in records and key facts there are significant differences. The structure of the value usually is not strictly defined, is generally untied, does not insist on non redundancy, practically rarely have natural keys, and there is no referential integrity.
How NoSQL systems do not support ACID properties, with these are binding BASE properties:

- Basically Available - most data is available most of the time.
- Soft state - state of the system can be changed, even when we do not have the explicit query the database. The reason is that the update may cause the update node to which it belongs database.
- Eventually consistent - The system will be consistent over time.

Besides BASE properties for NoSQL databases it is infiltrated a CAP theorem, known as Brewer's theorem, which says that it is impossible to satisfy all three desirable properties in distributed computer systems.

- Consistency - that all nodes see the same data at the same time,
- Availability - each request receives a response if successfully executed or no,
- Acceptance of separation - the system continues to operate regardless of the failure message to any part of the system.

When developing the system for database management, it can be chosen only two of the three properties of distributed systems. Traditional systems management databases concentrate on CA - consistency and availability.

3. TYPES OF NOSQL DATABASE

To date, it is developed many as 150 types of NoSQL databases which testifies to the strength NoSQL technology. All of these species is developed by application needs in order to provide the required functionality. Yet group is allocated to the model-based data and is classified into four categories:

1. Key/Value,
2. Document,
3. Graph,
4. Column.

3.1 Key/value model

Key-value systems fall into one easier. Such systems store things in the form of pairs, wherein the first is member of the key, a second value is associated to that key. According to (Sadalage & Fowler, 2012) this data model is actually a simple hash table, which is primarily used when accessing data from the database only through a key. Do a little more simply, this model can be seen as a single table in a relational model, which has two columns. While storing and reading values per key, the system works extremely efficient because there is no special burden on the system as well as in a relational database. The value to be entered in this database is a blob (blob - binary large object) and the database itself is not interested in the content that is entered in other words, the database does not make any additional verification of data, such as validation and the like. These bases are characterized by two main features: unique keys and the impossibility of queries. All keys must be unique, i.e. it is not possible to have two of the same key. Unable to execute queries against the values in the database, because the operations are only possible over the keys. There are not many restrictions on what can be used as a key, as long as the string of reasonable length. Also, the introduction of value is limited only by the size of the space memory. As long as the database has enough space it can store anything, including a variety of multimedia content.
Today, the famous databases are: Amazon DynamoDB, Riak, Redis, LevelDB, scalaris, MemcacheDB, Kyoto Cabinet.

3.2 Document model

The popularity of these databases are acquired because their intuitive data model. The main element of this data model is a document designed as set of keys with associated values. Documents are placed in the collection, which would be something similar to tables in a relational database, if a document matches row of the table. Unlike relational databases, document bases have dynamic scheme, which means that each document in a collection can have a different structure.
All data within the document are indexed in addition to his base. If we know one property of a document, it can be found and other documents with the same property. It is also possible to find the exact location of the requested data within a document using the "document path", the type of key that opens access to the tree structure of the document. Documents are grouped into collections or collections to be easier to manage the growing base. Collections are comparable with the structure of folders in Windows operating systems. It can be used to easily navigate the hierarchical structure of documents, grouping documents by similarity or certain criteria, etc. Collections can contain within itself another collection.
Today known database: MongoDB, Couchbase, CouchDB, RethinkDB ...

3.3 Graph model

Bases are based on a model graph are used less, but they are extremely useful in dealing with data that are very connected. These databases are comprised of nodes and links between them, and allow efficient passage of data across nodes that follow their relationship. Nodes are objects, and the links between these objects to represent the lines that connect the two nodes. Requests for the graph database is usually depict a journey in the graph (graph traversal) to

reach the required information. Connections between nodes can contain various properties, but the base can provide detailed reports of their relationship.
Graph databases are used in applications that need to analyze the relationships between objects and go through all the nodes in a particular order. Their major use is for example in social networks or systems that need to quickly analyze complex network structure and find patterns in it. Unlike most NoSQL database, graph database support the ACID model.
Today, the famous base: Neo4j, FlockDB, Infogrid, OrientDB ...

3.4 Column) model

The column base using identifiers of rows and columns as the keys to find the data. It can roughly be likened to a spreadsheet in which a combination of a row number and the letter of colic in order is necessary to find a particular cell containing the value. As with key / value base, the cells in the column base can be almost any value. In contrast to the relational database, bubble column systems can access the individual columns independently of the other column, which is one of the main advantages of such systems from the standpoint of performance.
Due to its technical characteristics, the main advantage of these NoSQL systems is that they are suitable for the processing of very large amounts of data, because the load and analyze all the values in column quite effectively, but it is also the case with the adding or deleting columns.
On this model NoSQL database will specifically retain and further process it through consideration of Cassandre.
Today, the famous base: HBase, Cassandra, accumulé, Amazon SimpleDB ...

4. CASSANDRA

One of the first NoSQL database was Google BigTable. From this model they have developed two well-known databases HBase and Cassandra. Facebook in 2008 developed a system for Cassandra email search index. Cassandra just based it design on the Amazon distribution dynamo model and its data to Google "Big table", which is written in Java [1].
Cassandra is a distributed database developed by Apache. It is a NoSQL database that is highly scalable, designed to manage very large data structures across many servers, and it is extremely resistant to errors.
It also contains the features of the database key value and column-oriented database which enables extremely fast work with the data.
This model is in some ways reminiscent of relational, but there are huge differences between these two models. The key difference is in the way of data storage, where relational databases store data in rows and columns by Cassandra.

4.1 Database elements

The basic unit for storing data in Cassandra is column consisting of pairs of names and values, where the name poses the name of the function key. In the columns are entering data with a time stamp (timestamp) that is used to resolve conflicts when writing and doing similar things. Row is a set of columns that are associated with the key. A set of similar lines makes family of columns.
Row spans multiple columns, and identifies the key. Different lines do not necessarily have the same columns that can be added to any order at any time, without being tied exclusively to that line and not for others. Columns also may contain other columns (super column), and such over-columns can create a new family of columns (super family column).
Keyspace is the space that Cassandra is attributed to the family of columns that are related to a particular application. Cassandra records are stored in its internal memory called memtable. Records in Cassandra are autonomous at line level, meaning that the addition of new or updating existing columns related to one key is treated as one record that will be successful or unsuccessful.

4.2 Creating a database

The process to create the database is not very different in syntax of creating a database using other query languages. Before displaying the process of creating a database it will be mentioned the tools and programs used for creating and working with Cassandra database:
CQL – Cassandra Query Language is a query language to create, update, delete and search data within the Cassandra database:

- DataStax OpsCenter – the graphical user interface environment that helps users manage the cluster, Cassandra Bayom, furnishing the user the ability to manage the cluster, as well as the visualization of certain statistical measurement operation of a base [2].
- DataStax DevCenter – visual query tool, is easy to use, which allows developers to create queries. The users can create and execute DML, DQL DDL and queries through a GUI, which is far simpler than the terminal approach [2].

It is noted that after the installation, Cassandre immediately displays DataStax OpsCenter as shown in Figure 1.

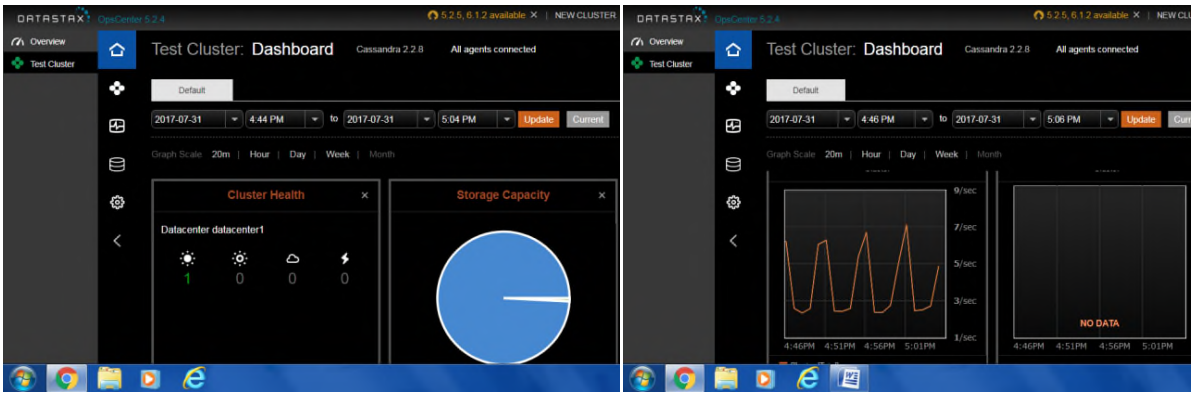

Figure 1: the graphical user interface environment - DataStax OpsCenter [2]

When creating a database first step is to run Cassandra CQL Shell tool and connect to the database, and then move to the very creation:

cqlsh>CREATE KEYSPACE VisokaSkola WITH REPLICATION = {'class' , 'Simple Strategy' , 'replication_factor' : 1};
cqlsh>USE VisokaSkola
cqlsh:VisokaSkola>

It can be seen here that the marking is different from the SQL database, and instead of the tag label it is used DATABASE key space.

A command *'WITH replication = {'class' : 'SimpleStrategy', 'replication_factor' : 1};* consists of several parts. 'SimpleStrategy' is the way that certain database is replicated by other nodes. Each replica is placed next to the next node in the clockwise direction, a replication factor shows us how many servers will be determined when base is replicated, 1 means only locally. When creating a database, it is necessary to choose a strategy. NetworkTopologyStrategy is used when the database is distributed across multiple centers.

After creating the database, by entering the command USE, it means that the database are going to be used for all subsequent queries. In this case, it is the database "VisokaSkola". A table is created with an example within previous created database. Table creates command „CREATE TABLE"A primary key of each table, is added at the end:

cqlsh:VisokaSkola>CREATE TABLE Studenti (
…name text,
…lastname text,
…social_number int,
…birth_date date,
…phone text,
…id int PRIMARY KEY);
cqlsh:VisokaSkola>

After completing the registration command creates a table in the database and populate with data is done via the command INSERT INTO ... .VALUES. As in a relational database with SQL SELECT command prints a table with grouped data.

Indices that are made above the columns are called secondary indexes. They are similar to hash function, but although they look similar to index of relational databases, they are inferior to them. Indices were placed in their family columns, and the user can not access it. Synchronized with the local data in the node, this means that the index is within the family of columns and will always be consistent with the local (but not outside the family column). The primary key (key order, the primary index) is a unique identifier of the order as well as relational databases and enables rapid access lines. As the lines are shared between servers in the ring, each server has only a part of the lines, and thus are keys distributed among them. Cassandra uses a divisor (partitioner) and a strategy for deploying copies (replicaplacement strategy) for locating the required nodes in the ring to access a specific order. The problem with primary keys is that they are at their location determined by the divisor. Divisor applied hash function to convert key

row (rowkey) into a single number (called a token) and then writes / reads the key to / from the node that owns the token.
The consistency height can be easily specified by:
quorum = new ConfigurableConsistencyLevel();
quorum.setDefaultReadConsistencyLevel(HConsistencyLevel.QUORUM);
quorum.setDefaultWriteConsistencyLevel(HConsistencyLevel.QUORUM);
Setting quorum as a measure of consistency ensures the response of the majority of nodes and restore data from the most recent time stamp when reading. Settings "ALL" can provide a maximum height consistency.

4.3 Authentication and authorization of users

Authentication is the process of identifying the user of the system while the authentication of the system determines what each user can do. To define the authentication within the Cassandra database firstly it is need to access cassanda.yaml file. After accession it must be opened in a text editor and change the text authenticator: AllowAllAuthenticator the authenticator. By editing the password it is achieved that database access is no longer allowed to all users. Also, since the driver DataStax OpsCenter is connected to the database it lost the right connections and needs to be upgraded to the standard username and password (Cassandra, Cassandra).

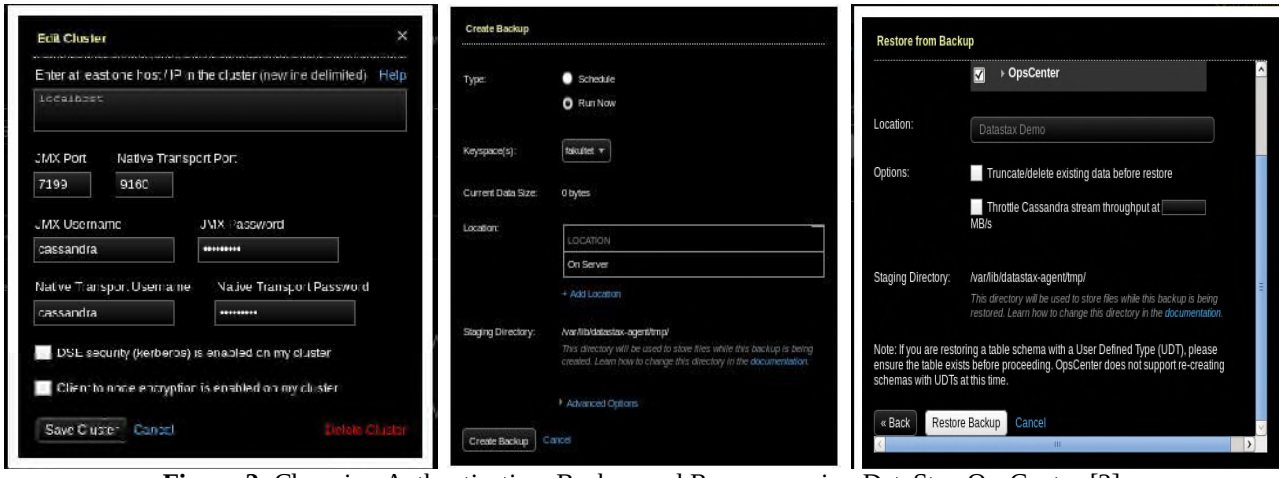

Figure 2: Changing Authentication, Backup and Recovery using DataStax OpsCenter [2]

After the above changes it must be implemented the system to reset. On restart, it is necessary to log in as superuser through command terminal are cqlsh cassandra -p cassandra which is entered within the terminal. Once when the user is entered, ie. and authenticates as superuser can add new users and define whether they will be superuser or not by using the following command:
CREATE USER user_name WITH PASSWORD 'password' NOSUPERUSER;
Adding authentication users, is also performed via editing cassandra.yaml (configuration file), because by default, people are allowed to do all the database or parts thereof, so that the "authorizer: AllowAllAuthorizer" needs to change into "authorizer: CasandraAuthorizer".
Also, the next step is to add a specific user authorization, or for example just to execute SELECT queries:
GRANT SELECT PERMISSION ON tablica TO user_name;

4.4 Backup and recovery

Backup is performed by storing snapshots (copy), which are taken when the system is online, in / var / lib / cassandra / data / snapshots directory. Name of the folder in which to store the copy is arbitrary, but by default this is snapshot. Cassandra with DataStax management tool provides two ways of data backup:
1. By using tool „nodetool" it is used statement in which it should be specified hostname, JMX port and keyspace (database name):
nodetool -h <host> -p <JMX port> snapshot <keyspace>
2. A process of using a backup of a Datastax OpsCenter is performed with the following sequence:
- From the left side in OpsCenter choose Services,
- At the top choose Backup->Configure

- At the top right side choose (Activity and Scheduled backup)

Activity backup allows current storage for all keyspaces or just some of them. Schedule Backup allows automatic storage, user just need to enter any of the day to go and specify how many days or weeks will perform backup and at what time. This can be determined and which keyspace user want to store.
It can also be adjusted automatically delete backups older than n days (Figure 2). It is used more incremental backups that backs up only what has changed since the last backup. By default it is turned off, or it can be activated via the configuration cassandra.yaml fila where necessary incremental backup's value changed from false to true.
Recovery database gives the possibility to load up data, ie. backup if the original database is damaged or if the database is unusable. It is also possible to restore data from any backup using DataStax OpsCentera sedentary procedure.

4.5 Performance of the system

Apache Cassandra is an open-source, distributed, decentralized, elastically scalable, highly available, fault-tolerant, constantly consistent, reliable, sustainable, column-oriented databases.
Cassandra is now used by companies such as Netflix, eBay, Twitter, Cisco and various other who are working with large amounts of data. The largest cluster is Cassandra where in use it contains more than 300 TB of data to over 400 machines. Data replication is automatic in many different places for full control of errors. In case of a node, it can be replaced without downtime. Provides control over synchronous or asynchronous replication for each update operation. Cassandra replication-in saves a redundant copy of data through the nodes in the ring. This means that if one node in the cluster drops, one or more copies of the data are widely available to other machines in the cluster. Clusters Cassandra base operating in the ring mode, wherein each node in the ring have the same role and responsibilities (Figure 3). Ring Cassandra cluster can be spread across multiple data centers (DC). Cassandra also supports virtual DC, as well as physical DC (Figure 3).

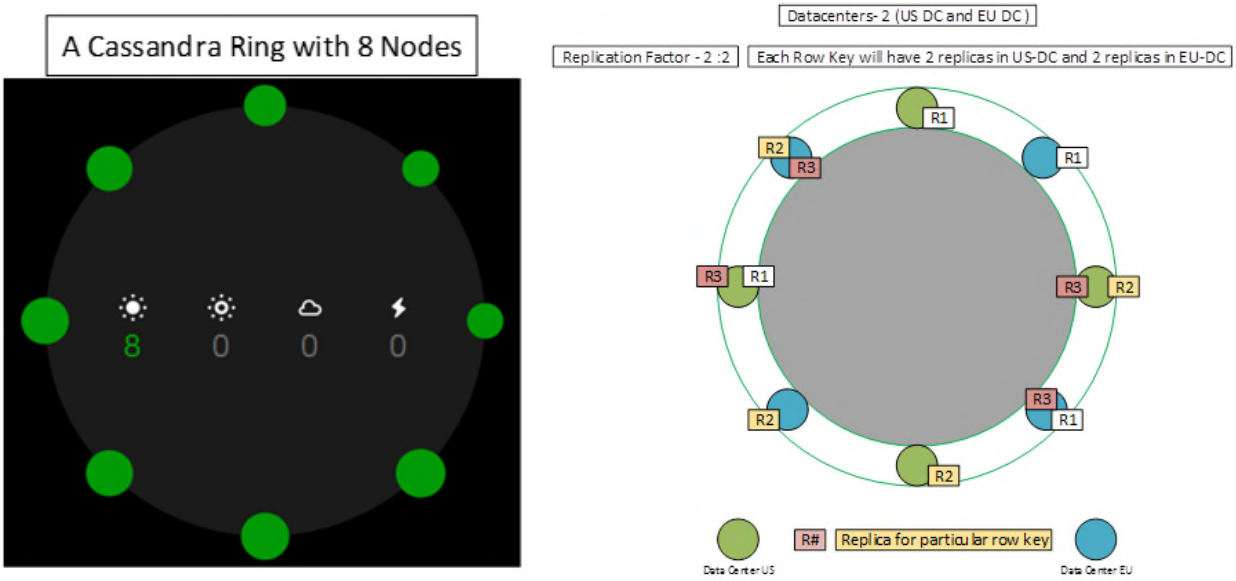

Figure 3: Cassandra ring and Datacenter replication [2]

High availability is ensured by using a peer-to-peer replication, setting the data center and the cluster ring structure, and automatic distribution of the information, as appropriate architecture.
Cassandra offers strong support for large amounts of data that circulate through multiple centers with asynchronous copy that enables low latency for all clients.
Cassandra is tested several times by well-known companies such as Facebook, Twitter and Netflix, and the results are still given over diagram.
Scalability is the ability to submit applications and the increasing demands of users, and that the application itself does not need to be changed. What is more scalable applications it will easier to bear an increased data flow. The goal to be pursued by all the designers of the system is to achieve linearity in the speed of response to the request and the amount of data with which to manipulate.
When we talk about the hardware there is a horizontal scalability and vertical scalability (Figure 4). Vertical scalability is when the application is located on a single server, and posses the increased flow to react so that the server adds memory, stronger processor, new core or additional hard disk (relational database). Horizontal scalability is more ideal solution, especially for large systems (NoSQL database). By adding new nodes system continues to operate as before

only with a new node (server) in the team. Cassandra is ranked as one of the most efficient and scalable among the currently existing NoSQL databases. Scalability means that adding nodes, we can have more demands served with the same performance and multiple nodes can lead to a reduction in execution time required. Scalability is an excellent base, read and write bandwidth grows linearly with the addition of new machines, without interrupting or slowing down applications.

Netflix has been tested for scalability at Cassandra base where it has proved that it is linear (Figure 4). Each client system is generated by writing about 17,500 requests per second and no bottlenecks while increasing traffic. Also clients have launched a 200 theme to generate traffic across clusters [3]. Sustainable development has the system and is carried out in several segments: a single-regeneration, Smooth Upgrade backup and Flexi Snapshot and Restore. Kassandra function of consistency makes it easy to return to the fallen node. If user damages the data in the node, he can put on the same offline, empty the damaged data, and then return to the ring node. Eventual consistency Cassandra will then propagate data from other nodes, according to the configuration. Cassandra feature consistency also supports smooth upgrades and even performs an upgrade version of the site. Cassandra supports automatic data backup / cluster snapshot, using tools such as nodetool or opscenter. OpsCenter allows complete or tablet levels or restoration after point.

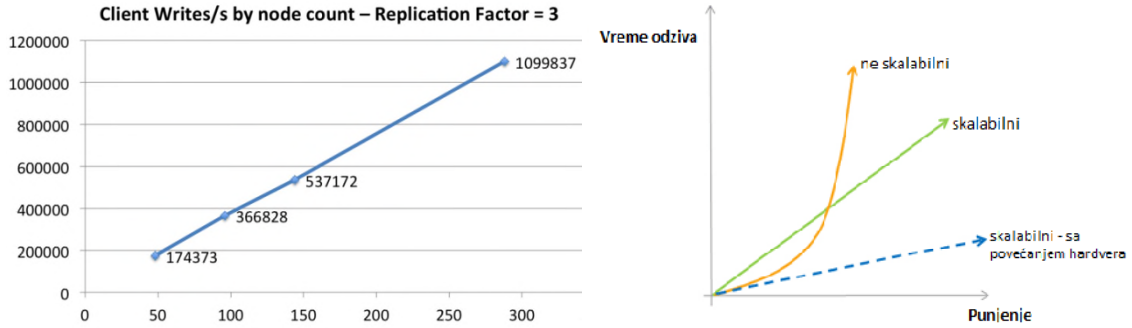

Figure 4: Scalability – Cassandra (Netfix) – all systems

5 . CONCLUSION

NoSQL database founds their place in the market not only due to its ability to solve certain problems, but also for the support of large social networks on the Web such as Google, Amazon, Facebook, and Twitter, which are developed and used by NoSQL products.

It is expected that in the future will be a large consolidation of the system that we have described and will be distinguished leaders in each category by which will be other NoSQL database corrected, and it is also expected the introduction of standardization in this area. Standardization would greatly facilitate the development of new and existing systems improvement, easier transition from one system to another, as well as ease of use. Cassandra database characterizes its high scalability when adding a large number of users, through the possibilities of introducing a large number of servers on which the base is located. Also, a big advantage is that users, regardless of their positions, can be connected to any of the available servers that have up to date information. Unlike traditional relational databases distributed DBMS, Cassandra will continue its work in the event of failure of one or more clusters. If we look to the future Cassandra will surely remain one of the best NoSQL database, because it has so far proved to be a very good base with a strong performance.

Cassandra is suitable as a base for various forms of electronic messaging, e-mail, chat, comments and so on. This database enables a high level of scalability, since it can handle a large number of data from different platforms. The need for distributed databases among users grows daily, so that NoSQL database, and Cassandra is expected that in the future will be more and more used.

LITERATURA

[1] http://cassandra.apache.org/
[2] https://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureIntro_c.html.
[3] https://www.netflix.com/
[4] N, Neeraj:*Mastering Apache Cassandra*, PacktPublishingLtd., UK, 2013.
[5]E, Hewitt: *Cassandra TheDefinitiveGuide*, O'Reilly Media Inc., USA, 2011.
[6] Aslett, Matthew:*How Will The Database Incumbents Respond To NoSQL And NewSQL?*. 451 Group, 2015.

[7] Kumar R, Gupta N, Maharwal H, Charu S, Yadav K: *Critical Analysis of Database Management Using NewSQL*, International Journal of Computer Science and Mobile Computing Vol. 3 (str. 434-438) 2014
[8] Janković, Olivera: *NoSQL dokument baza podataka: prikaz skladištenja podataka sa osvrtom na podatke sa senzora*", Infoteh-Jahorina, INFOTEH-JAHORINA Vol. 14 (str. 561-566) 2015.
[9] Grolinger K, Higashino W A, Tiwari A, Capretez M AM: *Data management in cloud environments: NoSQL and NewSQL data stores*, Jurnal of Cloud Computing, 2013.
[10] Robin Hecht Stefan Jablonski, University of Bayreuth: *NoSQL Evaluation A Use Case Oriented Survey*, International Conference on Cloud and Service Computing , 2011.
[11] Dietrich Featherston: *Cassandra: Principles and Application,* Department of Computer Science University of Illinois at Urbana-Champaign, 204.
[12] Ameya Nayak, Anil Poriya Dept. of Computer Engineering Thakur College of Engineering and Technology University of Mumbai: *Type of NOSQL Databases and its Comparison with Relational Databases*, International Journal of Applied Information Systems (IJAIS) – ISSN : 2249- 0868 Foundation of Computer Scie, 2012.
[13] Avinash Lakshman, Prashant Malik Cassandra-A Decentralized Structured Storage System, ACM SIGOPS Operating Systems Review archive, Volume 44 Issue 2, 2010.
[14] F. Bugiotti, L. Cabibbo, P. Atzeni, and R. Torlone: *Database design for NoSQL systems*, in Proceedings of the 33rd International Conference on Conceptual Modeling, pp. 223–231. , 2014.