Volume 6, Issue 2, February 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Network Data Monitoring Using NoSQL on Hosted Database Systems

[1]Zainab Mirza, [2]Ayesha Bhimdiwala, [3]Ishrat Sultana, [4]Mohsiuddin Arafat Jagirdar
[1] Head, [2, 3, 4] Student (B. E.)
[1, 2, 3, 4] Department of Information Technology, MH Saboo Siddik College of Engineering, Mumbai, India

Abstract— Demand for storing, retrieving and managing unstructured data led to the advancement of NoSQL database systems. The generation of unstructured data has not doomed traditional relational databases but it has paved way for industries to manage semi-structured and unstructured data as per their needs. NoSQL systems are designed to serve millions of queries and updates simultaneously in contrast to traditional database systems. The most famous example of UID that currently runs on NoSQL database system has made its mark among the citizens of India.

Keywords—NoSQL; CAP; Hadoop; HDFS; Hive; ACID; BASE; Big Data; DynamoDB; Dydra; BigTable; RavenDB

I. INTRODUCTION

Relational databases (RDS's) established a strong hold in the industry for structured data. In the past few years, the contribution of web technologies, mobile applications and social media has engendered semi-structured and unstructured data. Demand for storing, retrieving and managing unstructured data led to the advancement of NoSQL database systems. The generation of unstructured data has not doomed traditional RDS's but it has paved way for industries to manage semi-structured and unstructured data as per their needs. NoSQL systems are designed to serve millions of queries and updates simultaneously in contrast to traditional database systems.

Although some of the early NoSQL solutions built their systems atop existing relational database engines, they quickly realized that such systems were designed for SQL-based access patterns and latency demands that are quite different from those of NoSQL systems, so these same organizations began to develop brand new storage layers. [1] Later, Map-Reduce processing, the Hadoop framework, and HDFS revolutionized the entire system.

First in this paper, section 2, we are attempting to elaborate the CAP theorem used for NoSQL, variations in ACID and BASE properties. In section 3, we discuss the types of data models used to fundamentally design a database. In section 4, we studied the existing network monitoring tools and proposed a system that can handle unstructured logs and store them in Hive database system. Further in section 5, we describe cloud computing environment, introduce everything as a service and see four types of hosted databases mapped back to the data models available for businesses that cannot afford a physical infrastructure during the initial phases.

II. NOSQL

This term was first used in 1998 as the name of a relational database that did not provide an SQL interface. It resurfaced in 2009, as an attempt to label a set of distributed, non-relational data stores that did not, necessarily, provide ACID guarantees. The "NoSQL(east)" conference in 2009, engendered the popularity of NoSQL. After which NoSQL has attracted a great number of companies and projects over the last couple of years. Web companies and businesses running large and highly-visited websites have switched from relational databases towards non-relational databases.

By observing different kinds of NoSQL databases we can conclude that – NoSQL is:

- Schema-less
- Running on distributed environment
- Capable of handling unstructured data efficiently

Since the inception of databases, ACID properties (atomicity, consistency, isolation, and durability) are checked to ensure accuracy and completeness of the data in the database. The evolution of NoSQL databases finds it difficult to maintain ACID properties. Thus new principles were developed, resulting in BASE properties (Basically available, Soft-state, Eventual consistency). Ippolito summarizes the BASE properties in the following way: an application works basically all the time (basically available), does not have to be consistent all the time (soft-state) but will be in some known state eventually (eventual consistency, cf. [2]).

In an ACM symposium, Principles of Distributed Computing, Eric. A. Brewer formulated the CAP theorem in the keynote "Towards Robust Distributed System" which is widely adopted by the NoSQL community and web based organizations. The CAP-Theorem is established for distributed systems. CAP stands for:

- Consistency: It means that the data is same across all the clusters, to perform read or write to/from any node and get the same data.
- Availability: It means the ability to access the cluster even if a node in the cluster goes down.
- Partition Tolerant: It means the cluster continues to function even if there is a communication break between two nodes. See figure 1.

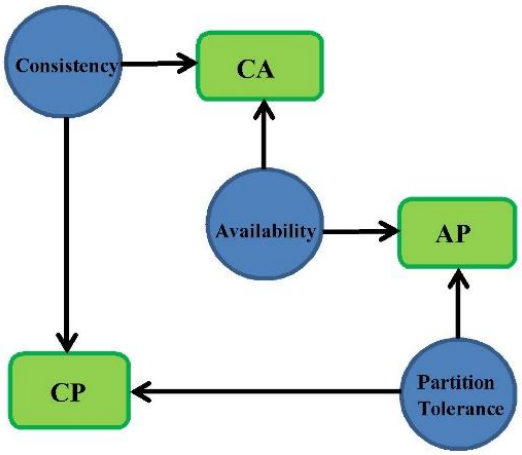

Figure 1.Characteristics of NoSQL databases

Of the above three characteristics, we can achieve at the most two of them in a distributed system. In order to get both consistency and partition tolerance, we will have to give up availability and so on. The diagram below shows the three possibilities that a database can fully achieve at a time. Theoretically, there are three options:

CA: The system is consistent between all nodes - as long as all nodes are online - and you can read/write from any node and be sure that the data is the same, but if you ever develop a partition between nodes, the data will be out of synchronization.

CP: In this, the data is consistent and between all nodes and maintains partition tolerance.

AP: The nodes are always online even if they can't communicate with each other. Once the partition is done, the nodes will resynchronize the data. It is not necessary that all the nodes will have the same data during or after the partition.

Thus, we can conclude from the theorem that, amongst the three CA, CP, AP it is difficult to achieve Consistency and Availability at the same time. Hence, it is clear that we can achieve Partition Tolerance and we need to decide between Consistency and Availability. This leads to BASE where we achieve Consistency or ACID where we achieve Availability. The theorem holds true for distributed systems which gives priority to Partition Tolerance and if partitioning is not possible then the system isn't distributed.

III. DATA MODELS

Before we learn some hosted databases, let us discuss the classical data models:

A. Key-value pair

Key-value store NoSQL database is a schema-less format database. The key can be of type synthetic or auto-generated and the value can be of String type, JSON or BLOB type. Fundamentally key-value type uses a hash table consisting of a pointer to the specific data and a unique key. Cache mechanism is followed that significantly enhances the performance of the system. There is a requirement of real key to read the data. The real key is a combination of both (bucket+ key) which user should know both the bucket and the key. A bucket stores a group of keys. Considering CAP theorem key-value stores facilitates with partition and Availability aspects and lacks in Consistency. There are some issues withkey-value stores like if the volume of the data increases, maintaining unique values as key becomes difficult. Fig 2 depicts the storage of details of an individual based on Adhaar number i.e., UID which is stored as key and its corresponding details as values in Key-value pair NoSQL database.

UID Database	
Key	Value
985647856987	ID: 985647856987 Name: Ram Verma DOB: 12/05/1985
965231478569	ID: 965231478569 Name: Rahul Mishra DOB: 02/12/1975
932547865263	ID: 932547865263 Name: Mahalaxmi Nair DOB: 02/12/1975

933214587546	ID: 933214587546 Name: Ravindra Gupta DOB: 06/01/1947

Figure 2 Key-value pair database example

B. Document based

A document store is similar to key-value store database except that the data which is stored is a collection of key-value pairs in a compressed form and the values stored are termed as documents provide some structure and encoding of the managed data. Common standards of document store are XML, JSON, and BSON (Binary Encoding of JSON Objects).

Document store embeds attribute metadata associated with stored content, which essentially provides a way to query the data based on the contents.

In relational databases, a record inside the same database will have same data fields and the unused data fields are kept empty, but in the case of document stores, each document may have similar as well as dissimilar data. Fig 3 shows how the tables which are not necessarily of uniform sized fields are linked to each other. For example, a detail of Ram Verma which is stored in UID Database is referenced through the ID database using its ID.

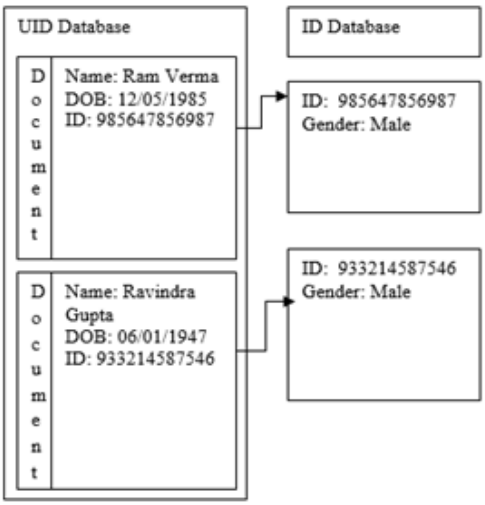

Figure 3Document based database example

Document-oriented databases should be used for applications in which data need not be stored in a table with uniform sized fields, but instead, the data has to be stored as a document having special characteristics document stores should be avoided if the database will have a lot of relations and normalization.

C. Column Store

Data in column-oriented NoSQL database is stored in cells grouped in columns of data rather than rows of data, in which columns are grouped logically into column families. Column families can contain a virtually unlimited number of columns that can be created at runtime and accordingly read and write is done using columns rather than rows.

Row-oriented databases are those databases in which all the rows are put together one by one as is the case of relational databases. Column-oriented databases are those databases in which all the values containing columns are put together.

The benefit of storing data in the column is fast search or access and data aggregation. Relational databases store a single row as a continuous disk entry. Different rows are stored in different places on disk while Column store databases store all the cells corresponding to a column as a continuous disk entry thus makes the search or access faster.

ID	Name	DOB
985647856987	Ram Verma	12/05/1985
965231478569	Rahul Mishra	02/12/1975
932547865263	Mahalaxmi Nair	02/12/1975
933214587546	Ravindra Gupta	06/01/1947

Figure 4 Row-oriented database example

ID	985647856987
	965231478569
	932547865263

	933214587546
Name	Ram Verma
	Rahul Mishra
	Mahalaxmi Nair
	Ravindra Gupta
DOB	12/05/1985
	02/12/1975
	02/12/1975
	06/01/1947

Figure 5Column store database example

D. Graph databases

Graph databases substitute relational tables with structured relational graphs of interconnected key-value pairings. Graph databases are useful as it concerns more in relationships between data rather than in the data itself. The graph-based database models use tree-like structures. These databases are commonly used by applications whereby clear boundaries for connections are needful to establish. For instance, when one register to a social network of any sort, one's friends' connections and their friends' friends' connections become much easier to work with using graph-based database management systems.

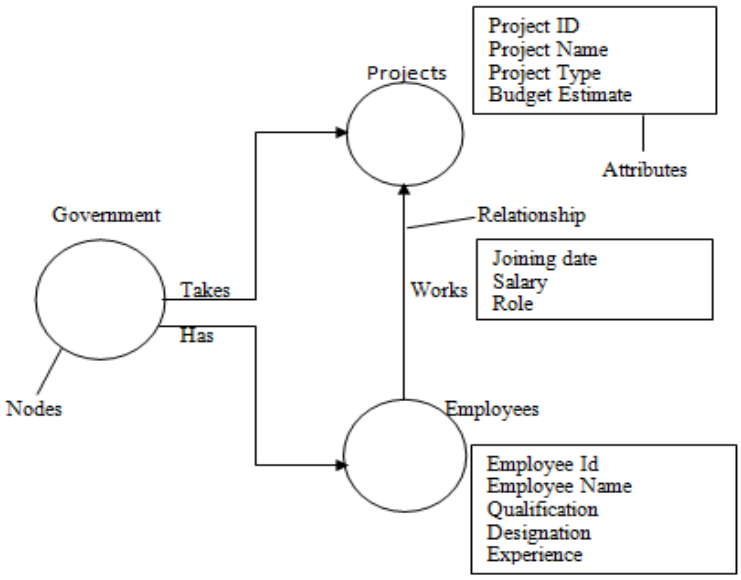

Figure 6 Graph database example

Independent of the total size of your data set, graph databases are best at managing highly connected data and complex queries. It comprises of nodes and edges. A node in a graph database can be defined as an instance of an object in its application. An edge represents the relation having properties. Fig 5 represents the relationship between Government and Projects taken which are completed by the employees employed by the Government.

IV. CASE STUDY

Nowadays, an enterprise network consists of myriad number of users with each user having dynamic IP's assigned. A log is a record of the events occurring within an organization's systems and networks. [3] The network administrator has the responsibility to check different parameters like measure the traffic, categorize them on the basis of access levels, monitor the users' behaviour and detect network anomalies, to avoid security breaches in the enterprise. The network administrator is prone to cause false positive and negative alarms at time due to human error. Originally, logs were used primarily for troubleshooting problems, but logs now serve many functions within most organizations, such as optimizing system and network performance, recording the actions of users, and providing data useful for investigating malicious activity [4]. Network Administrators are constantly striving to maintain smooth operation of their networks. In order to be proactive rather than reactive, administrators need to monitor traffic movement and performance.[5]

Thus, we studied the various available network log monitoring systems available for better analysis and less overhead on the network administrator. Over the past few decades, a lot of tools have been developed and widely used for Network traffic monitoring. In commercial environments, NetFlow is probably the de-facto standard for network traffic accounting. ntop includes both a NetFlow v5/v9/IPFIX probe and collector that can be used to play with NetFlow flows.[6] A snapshot of NetFlow is shown in the figure below.

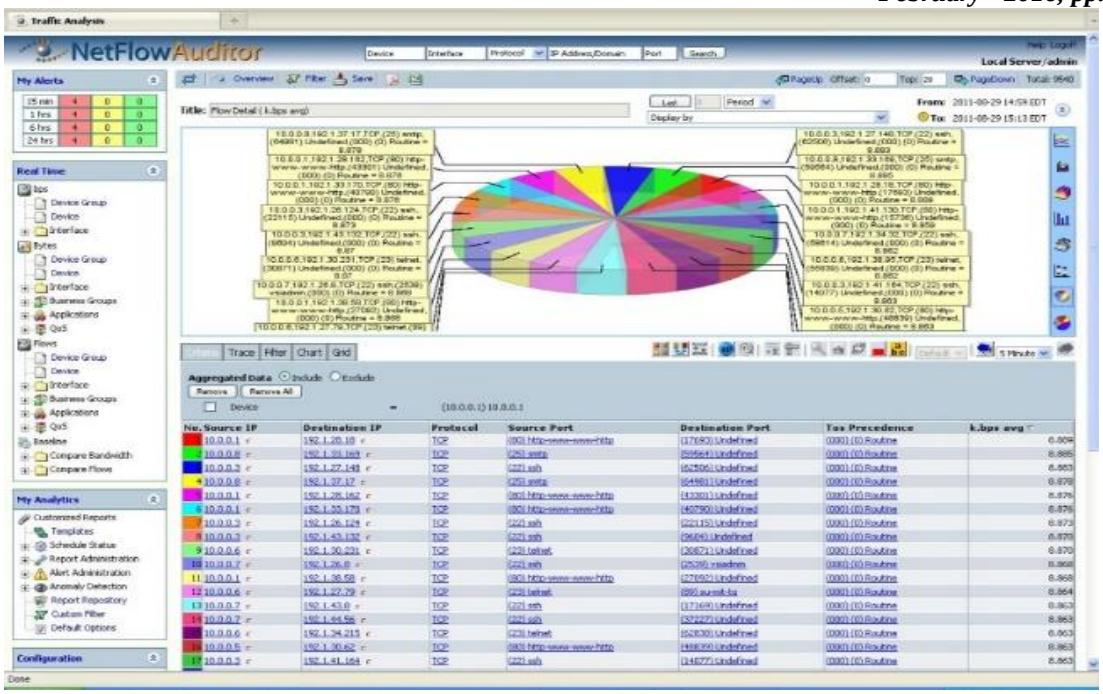

Figure 7NetFlow Auditor [6]

The above-mentioned systems provide a partial solution to the enterprise's requirement. The software's provide limited functionalities and are difficult to learn and use. Typically, it takes months to learn their features and capabilities. Most administrators use only a fraction of the features provided, performing only basic tasks such as traffic monitoring and fault monitoring. Further, most administrators change their job. This increases the overhead to retrain the new staff how to use the particular system. Consequently, a tool that is easy to learn and operate in a short period of time is desperately needed. Also, the given tools work on the limited amount of network traffic and have counterproductive features that all enterprises do not need. This resulted in the need to develop a robust application for the Network Administrator who shall monitor the user's behaviour and receive alerts if any anomaly is detected in the network.

A. Proposed System

To overcome the minor drawbacks and make a simple application for enterprises to monitor their network, we proposed a system that will manage the huge unstructured log files and ameliorate the current network scenarios.

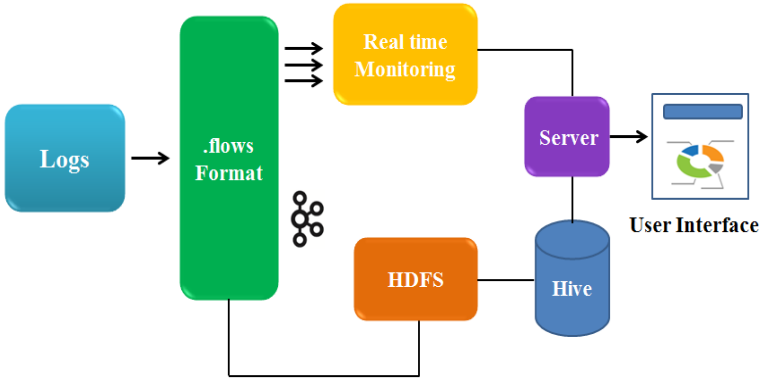

Figure 8 Proposed System

The data traveling in the network is stored on the server which generates binary logs. Since binary logs are not human readable, it is converted into .flows file (human readable can also be .txt). Now, .flows files are used for two purposes-

- To add the new logs generated from the network into the Hive database in HDFS of Hadoop Server. This will be used to perform historical analysis on the users to study their behaviour.
- These .flows files are being used to get the real time network results on web pages.

The files are segregated to obtain insights that will be beneficial for the enterprise. The result includes the maximum upload, download at the user, department and enterprise level, application using maximum bandwidth and traffic peak alerts.

This proposed system requires the enterprise to set up an infrastructure which includes log server, web servers, Hadoop HDFS environment and Hive ecosystem.

V. HOSTED DATABASES IN THE CLOUD COMPUTING ENVIRONMENT

According to the official NIST definition, "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The definition can be summarized as, cloud computing deals with computing and storage, the cloud service provider is responsible for providing the hardware resources and the end-user (consumer) is using these on demand, shared resources. The consumers are utilizing the services that the cloud service provider is providing. These services together fall in everything or anything as a service, generally known as XaaS. XaaS in the cloud means providing the organizations (i.e. consumers) flexibility to customize the environments according to their need.

We are emphasizing on Database as a Service (DBaaS) which is one of the services provided by the cloud environment. DBaaS provides services to store, retrieve, update and analyze data from the database provided by the cloud service provider. A hosted database, also called as cloud database is a database that runs in a virtualized environment on a cloud computing platform. The databases available on a cloud can be either SQL-based or NoSQL-based.

A. Dydra: (Graph-based)

Dydra[7] is one of the cloud-based graph databases. Dydra allows you to store your data natively as a property graph which represents relationships in data. Dydra is expressive as you can access and update your data with an industry-standard query language SPARQL. The captivating features of Dydra include its fast performance, its flexibility to explore everything from movie reviews and relationships to transaction logs. More importantly, it adds an aspect to your toolbox as it comes with the benefits of cloud services i.e., transparent scaling, fully managed and no lock-in.

B. RavenDB: (Document-based)

Raven[8] Database is a document-based hosted database which stores each document in JSON (JavaScript Object Notation) format. Raven Database eliminates the need for defining schema prior to store data. Instead, it provides the user with a facility that user only needs to give Raven Database an object and Raven will store it. There are sundry benefits of Raven Database over relational Database as it provides surpass performance in your application with faster development time. Additionally, it imparts user with better maintenance experience.

C. BigTable: (column-based)

BigTable [9] is sparse, distributed persistent multidimensional sorted map which is indexed by a row key, column key, and a timestamp. Row key is used to maintain data in lexicographic order. Each row range is termed as a tablet. Column keys are grouped into sets called column families and named using the syntax: family: qualifier.

D. DynamoDB: (Key value – based)

DynamoDB [10] is a completely supervised proprietary NoSql database service that is offered by Amazon.com as a part of the Amazon web services' portfolio. It is flexible and fast NoSql database suitable for consistent, single-digit millisecond latency at any scale. Due to its adaptable data model and reliable performance makes it a great fit for mobile, the web, Internet of things, ad-tech, gaming, etc.

VI. CONCLUSION

The main aim of this paper is NoSQL where we are emphasizing on CAP theorem, the current four classical data models being used, a case study for an enterprise to use huge unstructured network logs to analyze the user behavior and avoid security breaches. Further, we studied hosted databases based on the data models which can be used for SMBs (Small Medium Businesses) who cannot afford infrastructure during the initial phase of their business.
The future scope includes implementing our case study of network monitoring on a cloud-based hosted environment.

REFERENCES

[1] An Oracle White Paper, November 2012
[2] Strauch, Ch. 2011. NoSQL Databases. Lecture Selected Topics on Software-Technology Ultra-Large Scale Sites, Manuscript. Stuttgart Media University, 2011, 149 p., http://www.christof-strauch.de/nosqldbs.pdf
[3] Karen Kent ,MurugiahSouppaya, September 2006, Guide to Computer Security Log Management, *Recommendations of the National Institute of Standards and Technology*.
[4] Priyanka Sahu et al, System Monitoring and Security Using Keylogger, *International Journal of Computer Science and Mobile Computing* Vol.2 Issue. 3, March- 2013, pg. 106-1
[5] Alisha Cecil, A Summary of Network Traffic Monitoring and Analysis Techniques, http://www.cs.wustl.edu/~jain/cse567-06/ftp/net_monitoring/index.html
[6] NetFlow, www.ntop.org/products/netflow/nprobe/
[7] Dydra, *https://dydra.com*
[8] RavenDB, *https://ravendb.net*
[9] DynamoDB, *https://aws.amazon.com/documentation/dynamodb*
[10] Big Table *https://cloud.google.com/big*

