NoSQL Databases: A Paradigm Shift in the Storage and Management of Data in Databases

Ogheneovo, Edward E.
Department of Computer Science
University of Port Harcourt
E-mail- edward.ogheneovo@uniport.edu.ng

ABSTRACT

The world has witnessed rapid technological advancement in recent times especially with the growth of Internet technology and cloud computing. The quest for an efficient database management system that is scalable and very efficient in terms of performance in managing databases especially for Big Data in cloud computing environment using horizontal scalability is the major motivations for NoSQL existence. Prior to this time, the relational database system was the most popular database in use. But this system of data storage became inefficient in terms of scalability and performance in terms of storage, processing, and retrieval of the volume and varieties of data being generated on daily basis. NoSQL databases first started by big and multinational organizations like Amazon (Amazon Dynamo), Google (Google BigTable), LinkedIn (LinkedIn Voldemort), Twitter (Twitter FlockDB), Facebook (Facebook Cassandra), Yahoo! (Yahoo! PNUTS), etc. These technologies have revolutionized the volume, velocity, varieties, and values of data being generated and processed globally. It is against this backdrop that the NoSQL database technology was developed as a means of storing different forms of unstructured and semi-structured data. In this paper, we discussed the concept of NoSQL databases as a new paradigm, Big Data and their types, characteristics, CAP theory, and the benefits of NoSQL databases in the era of Big Data.

Keywords: NoSQL database, big data, relational database, technology, scalability, performance

iSTEAMS Multidisciplinary Conference Proceedings Reference Format
Ogheneovo, E.E. . (2019): NoSQL Databases: A Paradigm Shift in the Storage and Management of Data in Databases. Proceedings of the 20th iSTEAMS Multidisciplinary Trans-Atlantic Conference, KEAN University, New Jersey, United States of America. 10th – 12th October, 2019. Pp 191-206. www.isteams.net/usa2019 - DOI Affix - https://doi.org/ 10.22624/AIMS/iSTEAMS-2019/V20N1P15

1. INTRODUCTION

The world has witnessed rapid technological advancement in recent times especially with the growth of Internet technology and cloud computing. This has revolutionized the volume, velocity, varieties, and values of data being generated and processed globally in recent times. These data are collectively referred to as “big data”, a term coined in 1997 by Michael Cox and David Ellsworth [1] who were working with National Aeronautics and Space Administration (NASA) and later revisited in 2008 by Google [2]. Prior to this time, the relational database system was the most popular database in use. But this system of data storage became inefficient in terms of scalability and performance due to their storage, processing, and retrieval of the volume and varieties of data being generated on daily basis. It has been estimated that about 50,000GB of data are generated worldwide per second [3] [4].

However, for about a decade now, the concept of NoSQL has become increasingly popular due to the limitations of relational databases arising from volume and nature of data on the Web. With the development of Internet and Web 2.0 technologies [5], and with the concept of cloud computing, there has been proliferation of various forms of data in the cloud [6]. These data are of various forms such as structured, semi-structured, and unstructured data, simply referred to as big data. These data include documents, videos, images, etc. As the social media are increasingly becoming more and more popular among Internet users, so is the volume of data on the Internet are becoming larger on daily basis.

The term NoSQL database is a new paradigm in computer science and Information Technology (IT). The "NoSQL" stands for "not only SQL" database. As the name implies, it does not completely replace SQL but it compliments it such that the two databases can co-exist [7][8]. That is, proponents and users of NoSQL database admits that the goal is not to reject/replace relational "SQL" database but to overcome some of the technical deficiencies or limitations associated with relational databases [9]. Relational databases evolved over the years based on the kind of technology prevalent at that time. Today, there is growth and advancement in technology rapidly due to the volume and various types of data being deployed and used on the Internet and Web applications. Therefore, with the advancement of big data and cloud computing, there is indeed a paradigm shift in the way data and information are stored. Hence, the relational database is no longer efficient for managing data. Thus, NoSQL evolved in order to handle some of the problems inherent in relational database, thereby, providing better solutions for many of the modern storage problems. Therefore, NoSQL is an advanced database storage for cloud-based applications [10[11].

2. NoSQL DATABASES

The quest for an efficient database management system that is scalable and very efficient in terms of performance in managing databases especially for Big Data in cloud computing environment using horizontal scalability is the major motivations for NoSQL existence [12][13]. NoSQL databases first started by big and multination organizations like Amazon (Amazon Dynamo), Google (Google BigTable), LinkedIn (LinkedIn Voldemort), Twitter (Twitter FlockDB), Facebook (Facebook Cassandra), Yahoo! (Yahoo! PNUTS), etc. These organizations started using NoSQL even though they never rejected relational databases outrightly [14]. Having tried various options to optimize their relational database using a number of techniques such as adding more hardware or performing other upgrades using vertical integration by simplifying their database schema through de-normalization, relaxing durability and referential integrity, adding more query caching layers, separating read-only from write-dedicated replicas, and providing more partitions among their data, they soon discover that they have stretched their systems to a limit where further upgrades are no longer feasible since they cannot perform horizontal integration [15].

Most of the relational databases were designed; the major and most common model for hardware deployment involve using large serves attached to dedicated storage area networks (SANs). The intention of the designers then was to have a single machine that can manage the consistent state of the database on that system's connected storage. That is, databases should manage data in files by providing much concurrent access across multiple systems. However, this proved to be very unnecessary as most systems met designed goal with a single server and reliability goals with a stand-by ready to take over query processing in the event of master failure. Apart from the simple failure replication, there were only a few other options and they were all predicated on this same notion of completely consistent centralized data management [16][17].

Other technologies adopted include the two-phase locking and commit, and Oracle's RAC were also used. However, all these proved inefficient as they were difficult to manage and they were also found to be very expensive and costly. They were also limited in terms of usage to very few machines. Other solutions provided include logical SQL statement-level replication, single-master multi-replica log-based replication, and a handful of other approaches which had one limitation or the other ranging from cost overhead to administrative and technical overhead. Thus the relational databases failed to address the problem of scalability, latency, and availability requirements of many of the largest sites during the unprecedented growth of the Internet and Internet users [18].

The result is that there is a need for a new technology that can help solve the problem of scalability and performance of their database systems. It was this search for a new technology that is scalable and very efficient in terms of performance that gave birth to NoSQL. The NoSQL database paradigm started due to deficiencies and inefficiency of relational database in handling large and unstructured data. This was as a result of major shift in IT operations about a decade ago. Companies that generate large volumes of data began to develop their own NoSQL databases using different technologies. They started using NoSQL due to the when they soon discover that relational database can no longer cope with the volume and varieties of data they generate on daily basis in terms of scalability and performance. These organizations faced three (3) major challenges: (1) the unprecedented volumes of transactions, (2) expectations of low-latency access to large data sets, and (3) nearly perfect service while operating in an unreliable environment [19]. Today, there are several examples of NoSQL databases such as MongoDB, CouchDB, HBase, Cassandra, Riak, etc.

2.1 What are Big Data?

With the proliferation of various forms of data in the Internet and the Web due to social media (Facebook, YouTube, Twitter, Instagram, WhatsApp, Messeger, Palmchat, Imo, etc), Web browsers (Google, Yahoo!, LinkedIn, etc.,), ebusiness (Amazon, ebay, etc.), the relational database can no longer cope with the storage and manipulation of the volume, varieties, and velocity of data available across the globe [20][21]. One major handicap of the relational database is that most of these data are unstructured (e.g., video, images, social network data, geospatial, Internet search, genomics, astronomy, sensor network, etc.), the volume and nature of data generated on daily basis have become very massive that the relational database can no longer cope with their storage, processing and management. These data are called big data.

The term big data is used to describe the collection of complex and large data sets such that it's difficult to capture, process, store, search, and analyze using the relational database system. As noted by [22], big data helps organizations to store, manage, and manipulate vast amount of data at the right time in order to have a good understanding and ability to analyze such data such that organizations can effectively and efficiently manage and meet their business requirements in a timely fashion. Thus the concept of big data helps us in transforming the way we manage and leverage data due to technological advancement in Internet and Web applications especially with Web 2.0.

Definition (1): Big data is defined as any form of data source that has at least these three (3) characteristics: volume, velocity, and variety.

Definition (2): It is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.

Today, the number of Internet users have increased dramatically from 2.1 billion in 2013 to 3.7 billion in 2017 [23]. According to him, on the average the US alone uses 2, 657, 700 gigabyte of Internet data every minute. In 2016, report had it that more than 3.5 million text messages were sent every minute on the Internet and this has increased to about 15.2 million texts in 2017, a 334 percent increase. The report also has it that we tweet 456, 000 times, post 46, 700 Instagram photos, Google 3.6 million searches, publish 600 new page edits on Wikipedia, Facebook, WhatsApp, Amazon, ebay, and the Internet also cope with 103,447, 520 spam emails every minute (Data Never Sleeps, 5.0) [24]. Every day, hundreds of millions of take photos, make videos and video calls, and sent text messages across the Internet. Globally, business collects data on consumer preferences, purchases and trend, census data, crime data reports, songs, videos, images, and all sorts of data is growing fast. In 2013, the total amount of data sent across the Internet was 4.4 Zettabytes (Zettabyte is equivalent to 44 trillion gigabytes), and by 2020, it is estimated that the amount of data will rise to 44 Zettabytes [25]. Leochner [26] notes that 90 percent of the world data has been produced in the last two (2) years. Figure 1 shows the

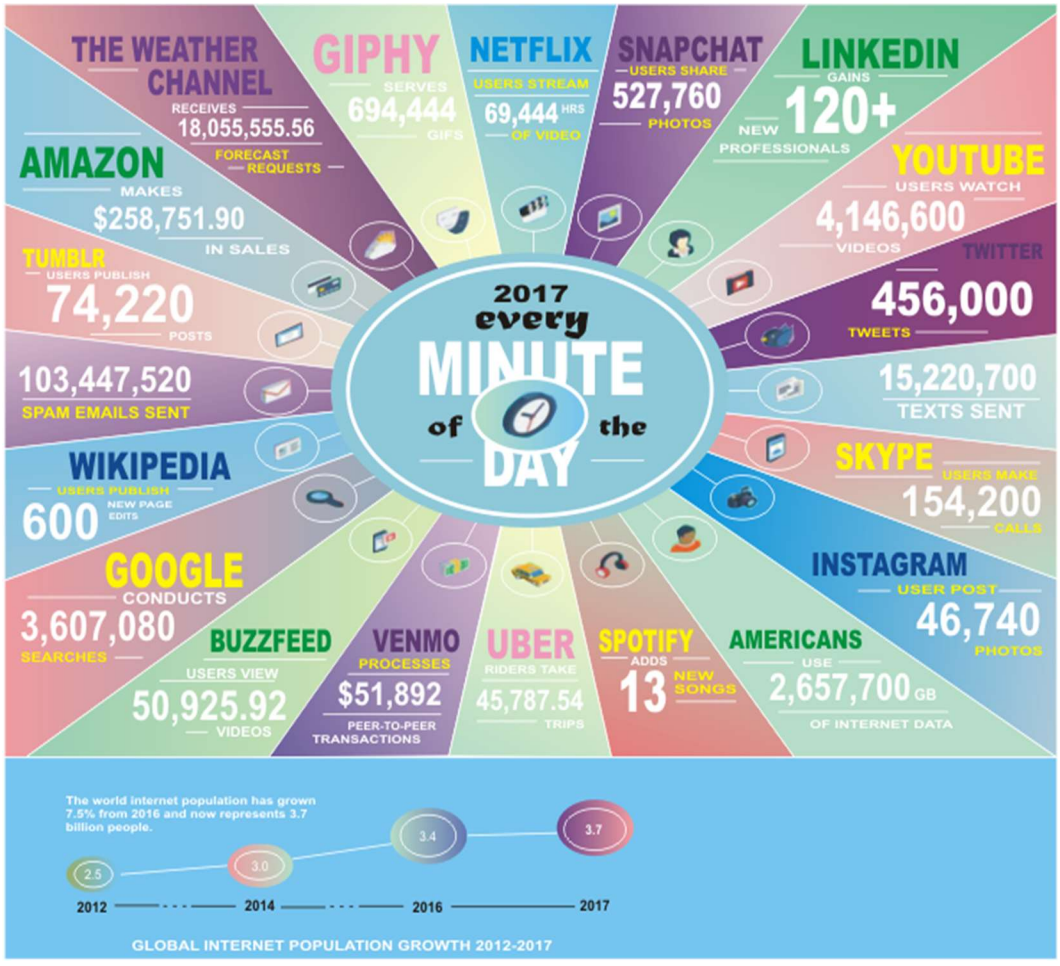

Fig. 1: The volume of data generated on the Internet
Source: Data Never Sleeps 5.0 (2017)

As noted by Walker (2018) [27], about 2.5 Quintillion (2.5×10^9) gigabytes of data are generated on daily basis. This volume of data would fill 10 million blu-ray disks, the height of which stacked, would measure the height of 4 Eiffel Towers on top of one another. It is estimated that 90% of these data are unstructured. These include tweets, photos, customers purchase history and even customer service call logs.

2.2 Why Big Data?

The explosion of data generation especially in social media coupled with the dropping of storage costs has created immense and large volume of data which continues to grow in size. In the past, the problem was simply managed by simply adding newer and better hardware, however, these days, it has become very infeasible to store the volume of data generated due to the proliferation of data as a result of Internet and Web 2.0 technologies since the volume of data generated today outpaces computer resources [28]. Thus the velocity or the rate at which data flows in and out of the system also plays a big factor. The constant influx of data on the Internet makes it difficult to distinguished between relevant data and noise. Data's usefulness especially at the rate which things change today, is extremely short lived making it practically difficult and near impossible to make value judgments in an acceptable amount of time. In big data environments, NoSQL does not require admins to separate operational and analytics databases into separate deployments. NoSQL is the operational database and hosts native analytics tools for business intelligence. In Hadoop environments, NoSQL databases ingest and manage incoming data and serve up analytic results.

2.3 Types of Data Available in the Internet

Various forms of data are available in the Internet these days ranging from texts, images, photos, videos, graphics, etc. These data can be grouped into two major categories. They are:

Structured Data

Structured data [29] are data that can be stored in tables in relational databases that are made up of rows (tuples) and columns (attributes). They are data that can be stored in structured query language (SQL). Structured data are usually strictly defined in terms of field and name and types. They often have relational key and are easily mapped into pre-designed fields. Structured data are mainly texts which makes them easy to be processed. Examples of relational database applications with structured data include airline reservation systems, inventory control, sales transactions, and ATM activity. Structured Query Language (SQL) enables queries of structured data types to be stored in relational databases. However, structured data is traditionally easier for Big Data applications to digest, yet today's data analytics solutions are making great strides in this area.

Unstructured Data

These are structured data that do not conform to the formal structure of relational databases. They usually contain tags or other markers which helps to separate semantic elements and enforce hierarchies of records and fields within the data. With the advent of Internet technology and Web 2.0, semi-structured data are very common these days. Examples include eXtensible Markup Language (XML) and HyperText Markup Language (HTML). JavaScript Object Notation (JSON), a human-readable text primarily used to transmit data-objects (objects consisting of attribute-value) between a server and the web application can also be used as an alternative to pairs XML [30].

The JSON is commonly used in NoSQL database mainly Mongo-DB and CouchDB. Semi-structured data model allows data and information from several sources with related but different properties to be integrated together and also enables the sharing of data and information on the web. Although, they have some structure, but they do not usually conform to a fixed schema [31].

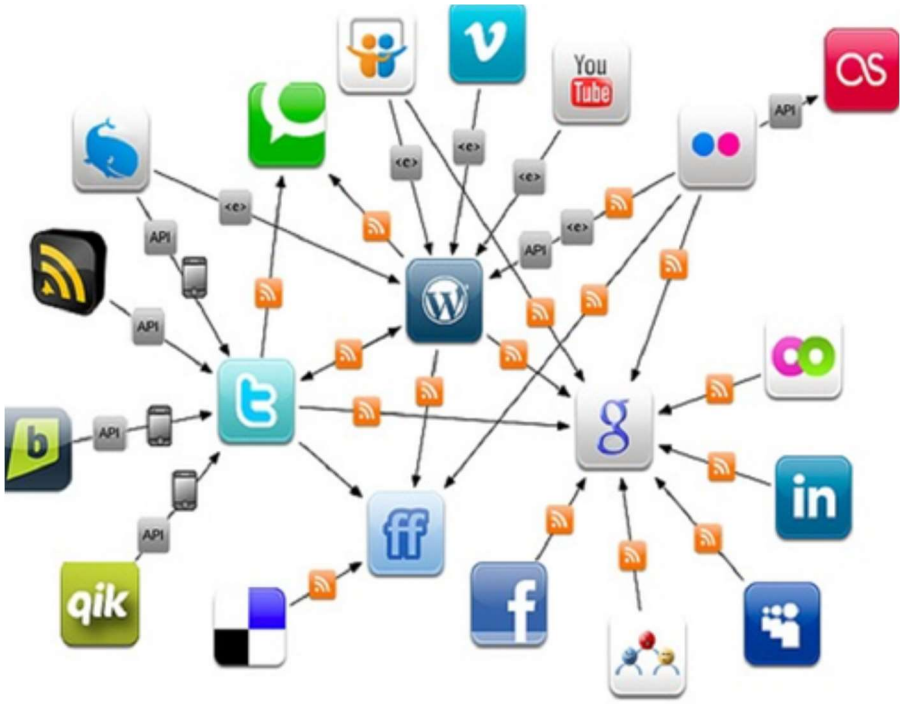

Fig. 2: Example of unstructured data
Source: Unstructured data, http://the-wacky-wordsmith.squarespace.com

They are therefore said to be schemaless. They are also self-describing in that they carry information about their own schemas. Unstructured data has internal structure but is not structured via pre-defined data models or schema. It can be textual or non-textual, and human- or machine-generated. It may also be stored within a non-relational database like NoSQL. Examples of human-generated unstructured data includes: Text files such as Word processing, spreadsheets, presentations, email, logs; Email: Email has some internal structure thanks to its metadata, and we sometimes refer to it as semi-structured. However, its message field is unstructured and traditional analytics tools cannot parse it; Social Media: Data from Facebook, Twitter, LinkedIn; Website: YouTube, Instagram, photo sharing sites; Mobile data: Text messages, locations; Communications: Chat, IM, phone recordings, collaboration software; Media: MP3, digital photos, audio and video files; Business applications: MS Office documents, productivity applications. Examples of machine-generated unstructured data includes: Satellite imagery: Weather data, land forms, military movements; Scientific data: Oil and gas exploration, space exploration, seismic imagery, atmospheric data; Digital surveillance: Surveillance photos and video; Sensor data: Traffic, weather, oceanographic sensors. Figure 3 shows the evolution of unstructured data ranging from traditional data such as numbers, texts, images to new data such as audio, video, and Hi-Res data growing 100X every year which must require new approach for handling them [32].

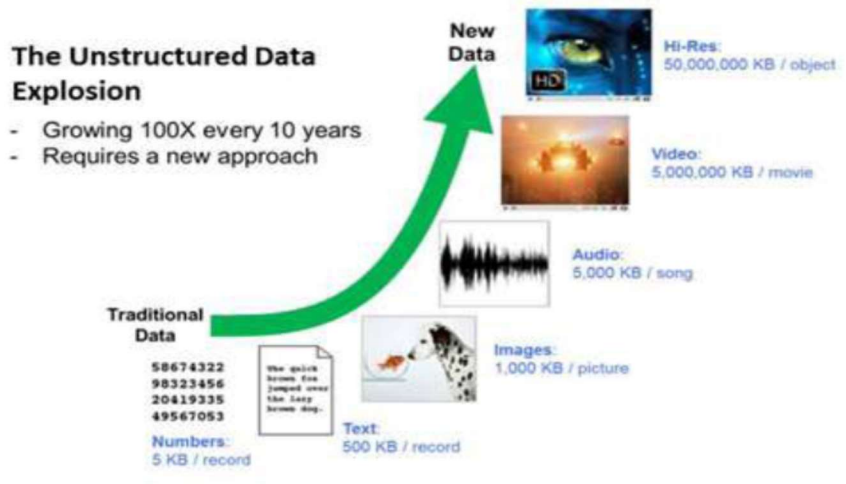

Fig. 3: Evolution of Unstructured Data
Source: Eberendu, A. C. [33]

Semi-Structured Data

Semi-structured data is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contain tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Therefore, it is also known as self-describing structure. Semi-structured data is information that does not reside in a relational database but that does have some organizational properties that make it easier to analyze. It must be noted that semi-structured data is considerably easier to analyze than unstructured data. Many Big Data solutions and tools have the ability to 'read' and process either JSON or XML. This reduces the complexity to analyze structured data, compared to unstructured data. Semi-structured data maintains internal tags and markings that identify separate data elements, which enables information grouping and hierarchies. Both documents and databases can be semi-structured. Email is a very common example of a semi-structured data type. Although more advanced analysis tools are necessary for thread tracking, near-dedupe, and concept searching; email's native metadata enables classification and keyword searching without any additional tools. Email is a huge use case, but most semi-structured development centers on easing data transport issues. Sharing sensor data is a growing use case, as are Web-based data sharing and transport: electronic data interchange (EDI), many social media platforms, document markup languages, and NoSQL databases. Examples of semi-structured data include Extra-Markup Languages (XML), JavaScript Object Notation (JSON), Yet Another Markup Language (YAML), and Binary JSON (BSON), NoSQL databases [34].

2.4 Characteristics of Big Data

There are three main characteristics of big data. They are: volume, velocity, varieties. However, Oracle added a fourth one, which is value.

Volume: This refers to the amount or quantity of data available or generated and processed. The size of the data generated determines the value and potential of the data under consideration and whether it can actually be considered as Big Data or not. The name big data is a term which is related to size of the data.

Velocity: The term velocity in this context refers to the speed of generation of data or how fast the data is generated and processed to meet the demands and the challenges of an organization. Thus in NoSQL, data can be accessed and streamed in a short time, almost at real-time. The implication is that the latency between read and write operations is considerably very small.

Variety: This has to do with different types of data being generated by individual or organization. That is, the category to which big data belongs to is also a very essential fact that needs to be known by the data analysts. This helps the people, who are closely analyzing the data and are associated with it, to effectively use the data to their advantage and thus upholding the importance of the big data. Data can be structured and unstructured. Structured data has semantic

Value: This property was proposed by Oracle Corporation. By data having value, we mean that such data … It is the intrinsic value that can be derived from data

Other properties include:
Variability: This is a factor which can be a problem for those who are analyzing the data. This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Complexity: By complexity of data we mean that data management can become a very complex process, especially when large volumes of data come from multiple sources. These data need to be linked, connected and correlated in order to be able to understand the information that is supposed to be conveyed by these data.

3. CAP Theorem

CAP Theorem is a concept which state that a distributed database system can only have 2 of the 3 properties: Consistency, Availability and Partition Tolerance. More formally, the CAP Theorem states that any networked shared-data system can have at most two (2) of three (3) desirable properties: Consistency, Availability and Partition Tolerance. The theorem relates the consistency model provided by replicated data store with its availability and tolerance to network partition. It was proposed by Eric Brewer (1998) and was proved by Lynch and Gilbert (2002) CAP Theorem is a very important concept in the Big Data world especially when we need to make. Figure 4 shows the relationship between the three properties of CAP [35].

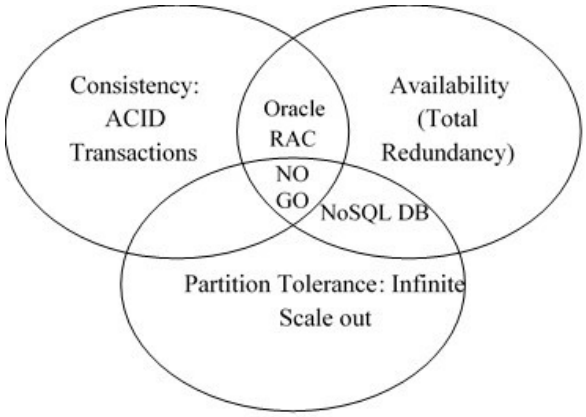

Fig. 4: CAP Theorem

As can be seen in the figure, consistency has to do with the ACID property of transactions which is a relational property whereas availability and partition tolerance are NoSQL properties. The partition tolerance property scales out infinitely. The implication is that even when a particular partition in a node is bad, it does not affect the entire network as the network will still continue to run even if one partition is malfunctioned.

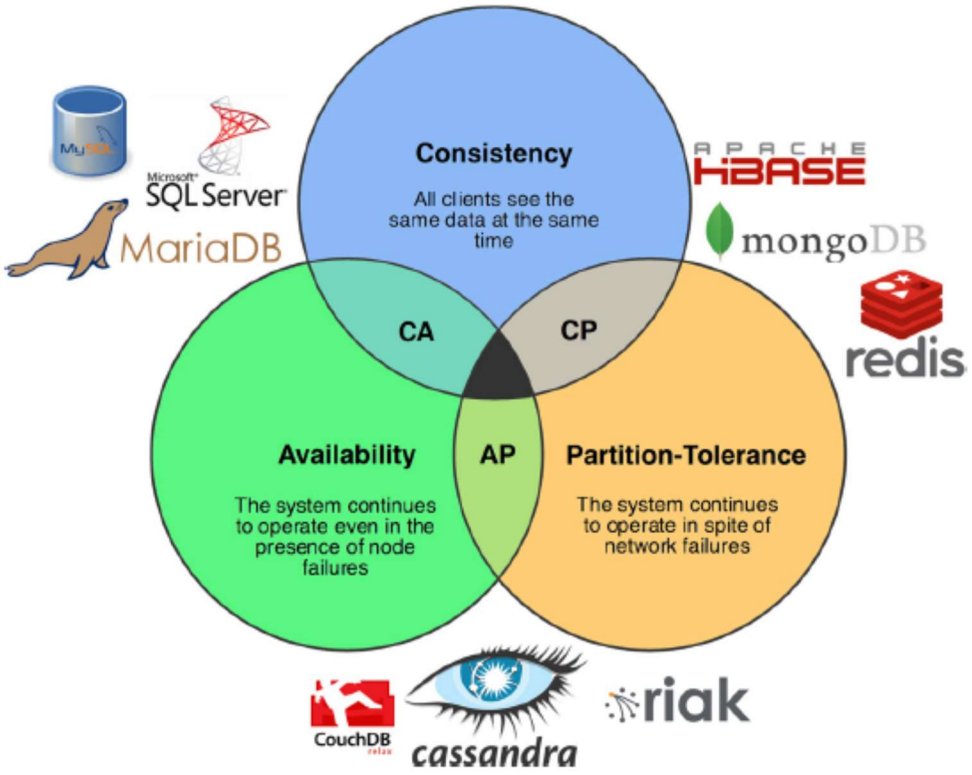

Fig. 5: CAP Theorem with databases that 'choose' CA, CP, and AP.
Source: Lorenço et al. [35]

Consistency

A system is consistent if all nodes see the same data at the same time. For example, if we perform a read operation on a consistent system, it should return the value of the most recent write operation. Thus the read should cause all nodes to return the same data, i.e., the value of the most recent write. In this case, every read operation receives the most recent write or an error. That is, all nodes see the same data at the same time. Therefore, consistency means serializability. By serializability we mean the system has a single up-to-date copy (i.e., replica) of the data.

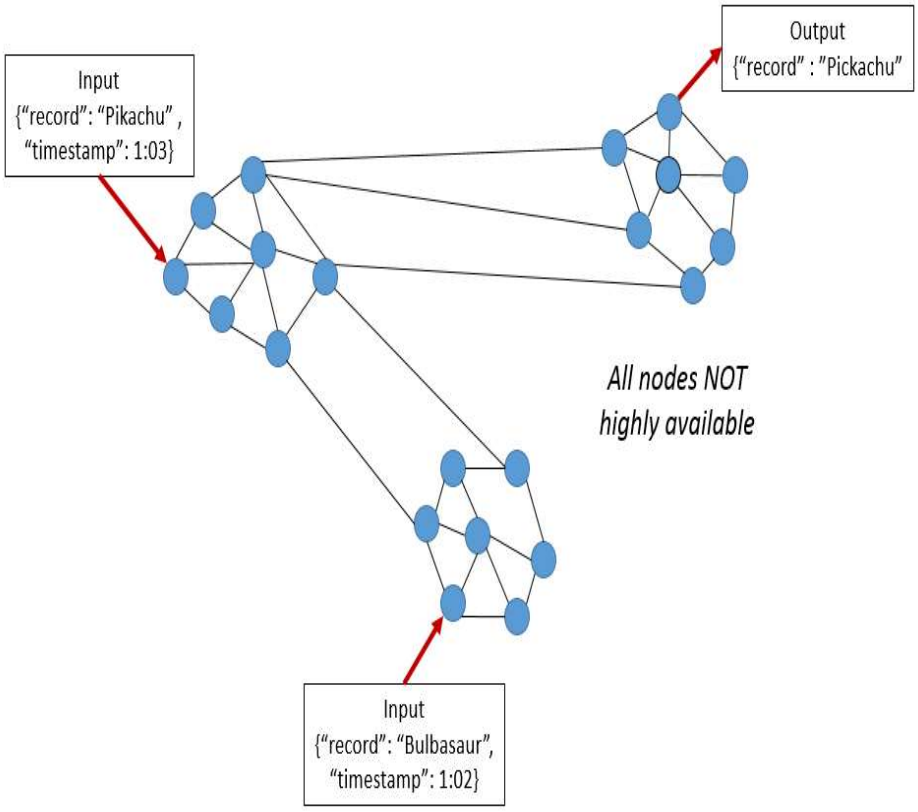

Fig. 6: Consistency property of CAP
Source: Nazrul, S. S. [36]

Availability

Availability refers to the proportion of time during which a service is able to successfully handle requests or the proportion of requests that receive a successful response. Thus every request receives a non-error response, without the guarantee that it contains the most recent write. The condition states that every request gets a response on success/failure. A response to a request is said to be successful if it is valid and arrives at the client machine within same time-out which may be specified in the service level agreement (SLA). Availability in a distributed system ensures that the system remains operational at all times. This is necessary as to ensure that every request gets a non-error response regardless of the individual state of a node. Gilbert and Lynch [37] describes availability as a system in which every request received by a non-failing mode in the system must result in a response. Therefore, n availability, if a request is sent by a client machine, the server and the service is not crashed, then the server must eventually respond to the client. The server is not allowed to ignore the client's requests. However, achieving availability in a distributed system requires that the system remains operational 100% of the time. Every client gets a response irrespective of the state of any individual node in the system.

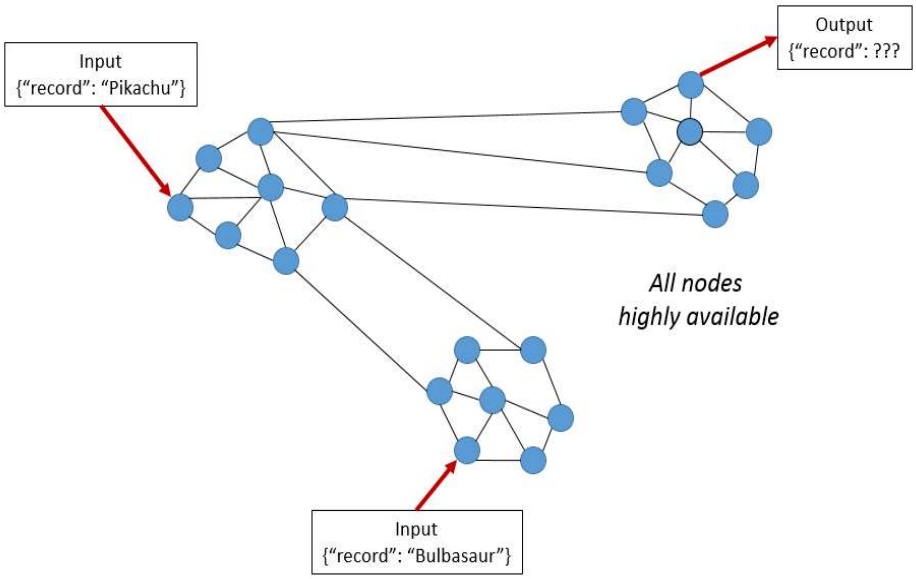

Fig. 7: High Availability property of CAP
Source: Nazrul, S. S. (2018)

Figure 7 shows a system that is highly available. As seen in the figure, we cannot say precisely if "Pikachu" or "Bulbasaur" was added first. The output could be either way.

Partition Tolerance

This property states that a system will continue to run despite the number of messages being delayed by the network between nodes. A system that is partition-tolerant can sustain any amount of network failure which will not result in the failure of the entire network. Figure 8 shows a network system having different nodes in which one partition in a network node is malfunctioned. However, the system is still working as there is no downtime due to the malfunction of only one partition in the node. That is, the system does not experience failure due the malfunction of only one node in the partition.

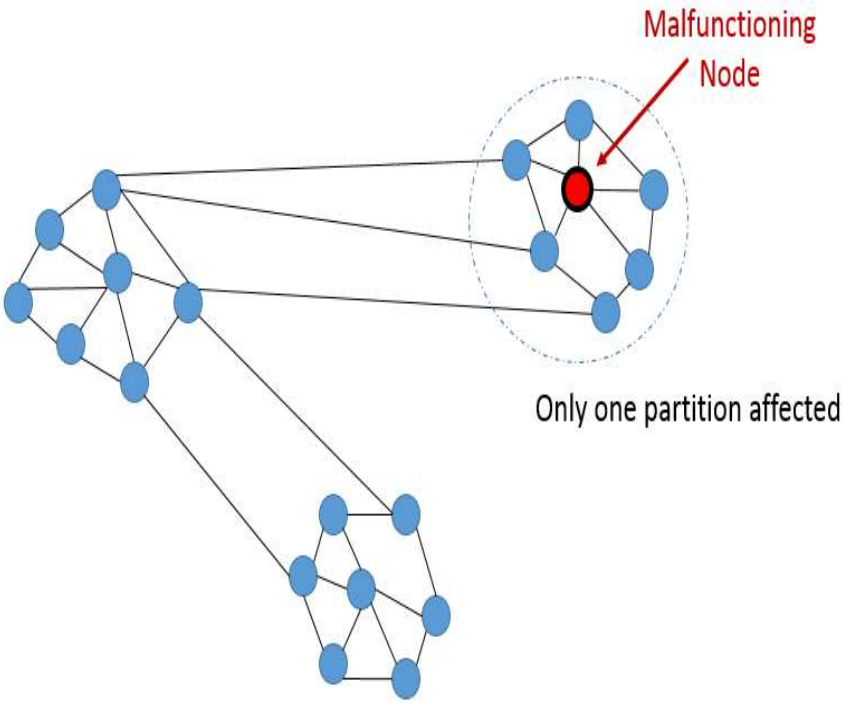

Fig. 8: Partition Tolerance property of CAP
Source: Nazrul, S. S. (2018)

Also, in this system, data records are sufficiently replicated across combinations of nodes and networks to keep the system up whenever there are intermittent outages. However, in modern distributed systems, partition tolerance is very necessary. Therefore, there is need to be trade-off between consistency and availability.
Usually, whenever a system is partitioned, the system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes. When a network partition failure occurs, there are two possible occurrences: 1) cancel the operation thereby decreasing availability and ensuring consistency or 2) proceed with the operation and then provide availability but risk inconsistency.

That is, the CAP theory states that there must be a trade-off between consistency and availability whenever a network is partitioned. However, since no network is guaranteed from failure, there is need for network partitioning such that failure in one partition dos not cause the failure of entire network thereby ensuring that the network is fault tolerant. Therefore, whenever there is partition, then only one option is left according to CAP theorem: consistency or availability. Whenever consistency is choosing over availability, the system will return an error or a time-out if particular information cannot be guaranteed to be up to date due to network partition. On the other hand, whenever availability is choosing over consistency, the system will always process the query and try to return the most recent available version of the information even if it cannot guarantee it is up to date due to network partitioning [38].

However, CAP theorem is often misunderstood such that one is often forced to think that one of the three properties must by abandoned at times. Thus we can understand that notion of "2 of 3" formulation of CAP theorem has been misused, misunderstood or misleading because it oversimplifies the relationship among the properties of CAP theorem. This is far from truth. In fact, choice is made only between consistency and availability only when a network is partitioned or failure occurs and at all other times, no trade-off has to be made between consistency and availability. This was shown by Eric Brewer himself in his paper 12 years later which he titled "CAP Twelve Years Later: How the 'Rule' Have Changed." According to him, CAP theorem only prohibits a tiny portion of the design space: perfect availability and consistency in the presence of partition are very uncommon. Thus there cannot be complete isolation or trade-off of any of the properties in CAP (Brewer, 2012).

4. TYPES OF NoSQL DATABASES

Different types of NoSQL databases exist depending on the

Key-Value Stores: This system provides a distributed index for object storage, where the objects are typically not interpreted by the system. Thus they are stored and handed back to the application as BLOBs, as in the popular memcached system. Also, the key-value stores provide object replication for recovery, partitioning of the data over many machines, and object persistence. Examples are: memcached, Redis, Riak, ArangoDB, InfinityDB, Oracle NoSQL Database, OrientDB, Scalaris, and Project Voldemart [39].

Document Stores: This system of NoSQL data store provides more functionality by recognizing the structure of the objects stored in it. Objects (or documents) may be structure, semi-structured or unstructured, i.e., they may have variable number of named attributes of various types (integer, strings, nested objects, images, etc.). Also, the system provides a simple query mechanism to search collections for objects with particular attribute values. Like the key-value stores, document stores can also partition the data over many machines, replicate data for automatic recovery and persistent the data. Document store is an extension of key-value databases where the value consists of a structured document. It organized data in a similar way as Extensible Markup Language (XML), Yet Another Markup Language (YAML), JavaScript object notation (JSON), and Binary JSON (BSON), which ensures one-to-one and one-to-many relationships in a single document. Thus a complex document can be retrieved or stored without using joint. Document store databases allow field indexes to be defined as well as advanced query features. Examples are: CouchDB, MongoDB, SimpleDB, and DynamoDB.

Graph Databases: Graph databases are used to store graphical data consisting of data interconnected with a finite number of relations between them. These data could be social relations, links road maps, network topologies, etc. Examples include Neo4j, OrientDB, AllegroGraph, FlockDB, InfiniteGraph, Sqrrl Enterprise, etc. However, not all systems fall exactly into these well-defined categories. For example, Riak might be categorized as a document store, since it does not provide some notion of fields. Also, it does not provide a field-based query mechanism. In the way, ArangoDB can also be categorized as key-value, graph, and multi-model databases. It must be noted that graph databases are very useful for consumer and campaign surveys, fraud detection, and social network applications. Using graph databases, it is feasible to identify the group of products often bought together by the same customers [40].

Extensible Record Stores: This system is provides a data model like the relational tables, but with a dynamic number of attributes, and like document stores, they provide higher scalability and availability made possible by database-wide ACID semantics. Examples are: BigTable, HBase, HyperTable, and Cassandra. The extensible record stores and some of the document stores come with (or are built on) a distributed computing infrastructures, e.g., for scattering/gathering work across nodes and determining group membership. For example, BigTable is used with a distributed Google File System (GFS), a distributed lock service (Chubby), and a MapReduce system [41].

Column-Family Databases: Column family stores data in a persistent, sparse, distributed hash maps. Thus arbitrary keys (rows) are applied to arbitrary key value pairs (columns). These columns can be extended further with arbitrary key value pairs since these key-value pairs lists can be organized into column families and keyspaces. Also, column family stores can appear in a very similar shape to relational databases on the surface, e.g. HBase and Cassandra both influenced by Google BigTable. Others are Accumulo, Vertica, etc. [42]

5. CONCLUSION

The world has witnessed rapid technological advancement in recent times especially with the growth of Internet technology and cloud computing. The quest for an efficient database management system that is scalable and very efficient in terms of performance in managing databases especially for Big Data in cloud computing environment using horizontal scalability is the major motivations for NoSQL existence. Prior to this time, the relational database system was the most popular database in use. But this system of data storage became inefficient in terms of scalability and performance in terms of storage, processing, and retrieval of the volume and varieties of data being generated on daily basis. NoSQL databases first started by big and multinational organizations like Amazon, Google, LinkedIn, Twitter, Facebook, Yahoo!, etc. These technologies have revolutionized the volume, velocity, varieties, and values of data being generated and processed globally. It is against this backdrop that the NoSQL database technology was developed as a means of storing different forms of unstructured and semi-structured data. In this paper, we discussed the concept of NoSQL databases as a new paradigm, Big Data and their types, characteristics, CAP theory, and the benefits of NoSQL databases in the era of Big Data.

REFERENCES

1. Cox, M. and Ellsworth, D. (1997). Application-Controlled Demand Paging for Out-of-**Core** Visualization. In Proceedings of the 8th Conference on visualization'97, Phoenix, Arizona, USA, October 18-24, 1997.
2. **Dontha, R. (2017).** The Origin of Big Data. https://www.kdnuggets.com/2017/02/origins-big-data.html.
3. Dietrich, A., Mohammad, S., Zug, S., and Kaiser, J. (2014). ROS Meets Cassandra: Data Management in Smart Environments with NoSQL. In Proceedings of the 11th Int'l Baltic Conference (Baltic DBIS), 2014.
4. Cattell, R. (2011). Scalable SQL and NoSQL Data Stores, SIGMOD Record, Vol. 39, No. 4, pp. 12-27.
5. Sharma, V. and Dave, M. (2012). SQL and NoSQL Databases. Int'l Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 8, pp. 20-27.
6. Moniruzzuman, A. B. M. and Hossain, S. A. (2013). NoSQL Database: New Era of Databases for Big Data Analytics – Classification, Characteristics, and Comparison. Int'l Journal of Database Theory and Application, Vol. 6, No. 4, pp. 1-13.
7. Nayak, A., Poriya, A. and Poojary, D. (2012). Type of NoSQL Databases and It's Comparison with Relational Databases. Int'l Journal of Applied Information Systems (IJAIS) –Foundation of Computer Science, Vol. 5, No. 4, New York.

9. Zvarenvashe, K. and Gotora, T. T. (2014). A Random Walk Through the Dark Side of NoSQL Databases in Big Data Analytics, IJSR, Vol. 3, Issue 6, pp. 506-509.
10. Abramova, V., Bernardino, J. and Furado, P. (2014). Which NoSQL Database: A Performance Overview, Open Journal of Databases (OJDB), Vol. 1, Issue 2, pp. 17-24.
11. Kumer, B., Gupta, N. Chara, S., and Jangir, S. (2014). Manage Big Data Through NewSQL, National Conference on Innovation in Wireless Communication and Networking Technology.
12. Fallon, M., Johannsson, H., Kaess, M., and Leonard, J. J. (2013). The MIT Stata Centre Dataset. The Int'l Journal of Robotics Research, Vol. 32, No. 14, pp. 1695-1699.
13. Niemueller, T., Lakemeyer, G., and Srinivasa, S. S. (2012). A Generic Robot Database and Its Application to Fault Analysis and Performance Evaluation. In Proceedings of the Int'l Conference on Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ, pp. 364-369.
14. Veen der van, J. S., Waaij, van der, B., and Meijer, R. J. (2012). Sensor Data Storage Performance SQL or NoSQL, Physical or Virtual. In Proceedings of the 5th IEEE Int'l Conference on Cloud Computing (CLOUD'2012)., pp. 431-438.
15. Tauro, C. J. M., Patil, B. R. and Prashanth, K. R. (2013). A Comparative Analysis of Different NoSQL Databases on Data Model, Query Model and Replication Model, Elsevier Publications.
16. Techopedia,Unstructured Data, https://www.techopedia.com/definition/13865/unstructured-data.
17. Shacklett , M. (2017). Unstructured Data: A Cheat. Sheet. https://www.techrepublic.com/article/unstructured-data-the-smart-persons-guide/.
18. Sint, R. Schaffert, S. Stroka, S. and Ferstl, R. (2009). Combining Unstructured, Fully Structured and Semi-Structured Information in Semantic Wikis. In Proceedings of 4th Workshop on Semantic Wiki - The Semantic Wiki Web 6th European Semantic Web Conference, Hersonissos, Create Greece, June 2009.
19. Bernkman, P. O. (2014). Unstructured Data: Practices in Polar Institutions and Networks: A Case Study with the Artic Options Projects, Data Science Journal, vol. 13, pp. 64-71.
20. Eberendu, A. C. (2016). Unstructured Data: An Overview of the Data of Big Data. Int'l Journal of Computer Trends and Technology (IJCTT), Vol. 38, No. 1, pp. 46-50.
21. Das, T. and Kumar, P. (2013). Big Data analytics: A Framework for Unstructured Data Analysis. Int'l Journal of Engineering and Technology (IJET), Vol. 5, No. 1, pp. 153-156.
22. Hurwitz, J., Nugent, A., Halper, F. and Kaufman, M. (2013) Big Data for Dummies. John Wiley & Sons, Hoboken.
23. Data Never Sleep 5.0 (2017). Data Never Sleep 5.0, https://www.domo.com/blog/data-never-sleeps-5/.
24. Straunch, C. (2019). NoSQL Databases: Selected Topics on Software – Technology Ultra-Large Scale Sites, Stuttgart Media University, Stuttgart, Germany.
25. Corbellini, A. Matreos, C. Zunim, A. Godey, D. and Schiaffino, S. (2007). Persisting Big-Data: The NoSQL Landscape. Information System, Vol. 63, pp. 1-23.
26. Loechner, J. (2016). 90% of Today's Data Produced in Two Years. https://www.mediapost.com/publications/article/291358/90-of-todays-data-created-in-two-years.html
27. Oussous, A., Benjelloun, F-Z., Lahcen, A. A. and Belfkih, S. (2018). Big Data Technologies: A Survey. Journal of King Saud University – Computer and Information Sciences, Vol. 30, Issue 4, pp. 431-448.
28. Hanig, C., schierle, M. and Trabold, D. (2010). Comparison of Structured vs. Unstructured Data for Industrial Quality Analysis. In Proceedings of The World Congress on Engineering and Computer Science,
29. Amir, G. (2013). Beyond the Hype: Big Data Concepts, Methods and Analysis. Int'l Journal of Information Management, vol. 35, No. 2, pp. 137-144.
30. Nazrul, S. S. (2018). CAP Theorem and Distributed Database Management Systems. https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e.

31. Gibber, S. and Lynch, N. (2002). Brewer's Conjecture and the Feasibility of Consistent, Available, Partitio-Tolerant Web Service. ACM SIGACT News, Vol. 33, Issue 2, pp. 51-59.
32. Abadi, D. (2010). DBMS Musings: Problems with CAP and Yahoo's Little Known NoSQL System, DBMS Musings. Retrieve 10th August, 2019.
33. Fox, A. and Brewer, E. (1999). Harvest, Yeild and Scalable Tolerant Systems. In Proceedings of the 7th Workshop Hot Topics in Operating Systems (HosOS'99), IEEE CS, pp. 174-178.
34. Lourenço, J. R. Cabral, B. Cerreiro, P. Vieira, M. and Bernardino,J. (2015). Choosing the Right NoSQL database for the Job: A Quality Attribute Evaluation. Journal of Big Data, Vol. 2, No. 18, pp. 1-26.
35. Litch, A. and Mattson, J. (2013). Investigating Storage Solutions for Large Data: A Comparison of Well Performing and Scalable Data Storage Solution for Real-Time Extraction MSc. Thesis, Department of Computer Science and Engineering, Chalmers University of Technology Göteborg, Sweden, 2010.
36. Mason, R. T. (2015). NoSQL Databases and Data Modeling Techniques for a Document-Oriented NoSQL Databases. In Proceedings of Informing Science & IT Education Conference (InSITE), pp. 259-268.
37. Zaki, A. K. (2014). NoSQL Databases: New Millennium Database for Big Data Big Users, Cloud Computing and Its Security Challenges, Vol. 3, Issue 3, pp. 403-409.
38. Chen, Y., Wang, W., Liu, Z. and Lin, X. (2009). Keyword Search in Structured and Semi-Structured Data. In Proceedings of the 2009 ACM SIGMOD Int'l Conference
39. Maluf, D. A. and Tran, P. B. (2008). Managing Unstructured Data with Structured Legacy Systems. In Aerospace Conference, 2008 IEEE, pp. 1-5.
41. Berezecki, M., Frechtenberg, E., Palecyry, M. Steele, K. (2011). Many-Core Key-Value Store. In Proceedings of the Int'l Green Computing Conference and Workshops (IGCC), Florida, USA, pp. 1-8.
42. Han, J., Sang, M. and Song, J. (2001). A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing. In Proceedings of the 10th IEEE/ACIS Int'l Conference in Computer and Information Science, Sanya, China.