Database-as-a-Service for Big Data: An Overview

Manar Abourezq[1], Abdellah Idrissi[2]
Computer Science Laboratory (LRI)
Computer Science Department, Faculty of Sciences
Mohammed V University
Rabat, Morocco

***Abstract*—The last two decades were marked by an exponential growth in the volume of data originating from various data sources, from mobile phones to social media contents, all through the multitude devices of the Internet of Things. This flow of data can't be managed using a classical approach and has led to the emergence of a new buzz word: Big Data. Among the research challenges related to Big Data there is the issue of data storage. Traditional relational database systems proved to be unable to efficiently manage Big Data datasets. In this context, Cloud Computing plays a relevant role, as it offers interesting models to deal with Big Data storage, especially the model known as Database as a Service (DBaaS). We propose, in this article, a review of database solutions that are offered as DBaaS and discuss their adaptability to Big Data applications.**

Keywords*—*Cloud Computing; Big Data; Database as a Service

I. INTRODUCTION

The volume of data stored in the world has been doubling every two years, and will reach a dazzling 40 billion terabytes (TB) by the year 2020 [1]. By means of comparison, the total size of data that existed in the digital universe in 2000 is 800 million TB, which means that the volume of data will be multiplied by 50 by 2020. This data is generated by various sources: Social Media, E-Commerce, Internet of Things, Sensors, etc. Organizations are also gathering more and more information, for various purposes: analysis to ameliorate their market position and offer better services to their customers, fraud detection, scientific projects like in genomics, legal reasons (for example, Moroccan firms are required by law to store ten years of financial data), etc.

This flow of data, which has been referred to as a flood or a tsunami, can't be managed using a classical approach and has led to the emergence of a new buzz word: Big Data. Almost all major IT leaders invested in various Big Data projects, from Google's BigQuery and Datastore, to Amazon's Elastic MapReduce, to Facebook's Cassandra, Yahoo!'s PNUTS, etc.

Cloud Computing has a leverage effect on Big Data, providing the computing and storage resources necessary to Big Data applications. The inherent characteristics of Cloud Computing, such as elasticity, scalability, automation, fault-tolerance, and ubiquity offer an ideal environment for the development of Big Data applications.

Cloud Computing is an established computing paradigm that gained in importance in the last decade. It refers to the utilisation of storage and computation resources as a utility.

There is a great tendency to opt for using IT as a service. It is estimated that more than 80% of Internet users use Cloud Computing in one form or another, from email services to different business applications as a service, all through data storage, development platforms, etc [2]. This usage percentage is even greater when it comes to companies: In a survey conducted by RightScale in January 2015, 93% of respondent companies confirmed using Cloud Computing [3], which shows that the latter is steadily advancing to become an integral part of companies and individuals use of IT.

Although the emergence of Cloud Computing is relatively new, the idea of delivering computing as a utility dates back to as far as the 1960s, when pioneers like John McCarthy, Leonard Kleinrock, and Douglas Parkhill predicted that, just like water, electricity, or the telephone, computing resources will someday be used as a public utility [4, 5, 6].

There is no consensual definition of Cloud Computing, yet. Many works have proposed their own as discussed in [7, 8]. One of the most cited definition is the NIST's, where Cloud Computing is defined as being a "*model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction*" [9].

Through the plethora of definitions, it emerges that cloud computing has several major characteristics, especially the following:

- Virtualization: physical resources are virtualized in order to optimize their utilization;
- Pooling: multiple users share access to the same pool of virtualized resources. This results in optimizing costs of infrastructure, installation, hosting, and maintenance for providers, who benefit from the economy of scale, and can offer more competitive prices;
- Ubiquity: cloud services are always accessible, anytime, anywhere, and from various computing devices;
- Remote access: cloud services are accessible via a network. It can be the Internet for cloud services that are destined to the general public, or LAN for private ones;
- Automation: users can get the resources they need without having to interact with the provider or require their intervention;

- Elasticity: resources are automatically and rapidly increased or decreased to accommodate the workload: when it increases, more resources are added to support it, and when it decreases, superfluous resources are removed. Thus, available resources are directly proportional to workload requirements, ensuring that client applications will have the exact amount of resources needed at any given time;
- Pay-as-you-go: users don't need to make any upfront investment in infrastructure, software licenses, etc. They pay only for the resources they consumed, without surplus. Although these resources are multi-tenant, providers strictly measure each client's resource consumption and bill them accordingly. Many billing plans are proposed, some based on the volume of resources used, others on the duration of usage (usually in hours), and others on "commitment" (paying per month, for example).

Cloud Computing's major deployment models are public, private, community, and hybrid (*Fig.* 1).

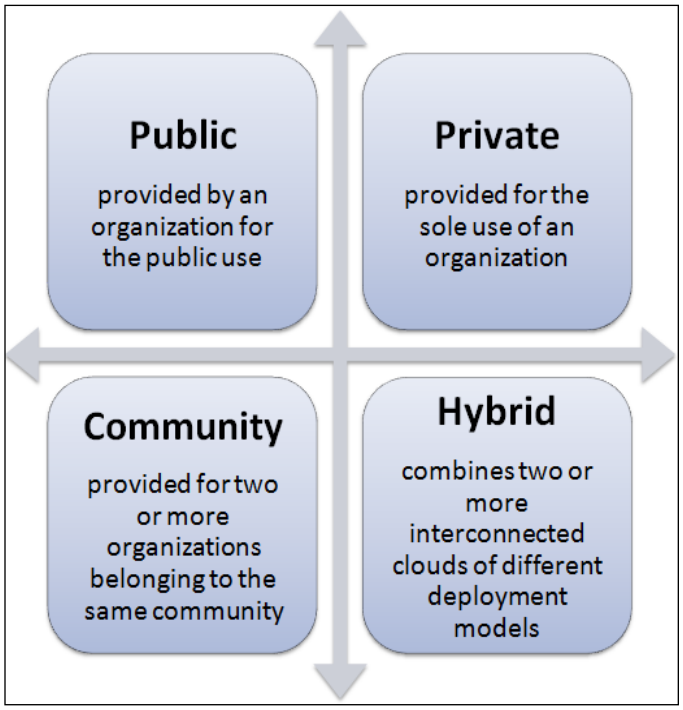

Fig. 1. Cloud deployment models

A Public Cloud is a deployment model in which cloud services are provided via a public network, usually the Internet. Examples include Amazon's Elastic Compute Cloud (EC2), Google's App Engine, and Microsoft's Azure.

A Private Cloud is provided for the sole use of an organization that can either choose to be responsible for managing it or delegate its management to a third-party. The organization can also choose to host it on-premise or off-premise. A variation of this deployment model is the On-Site Private Cloud, where the cloud is hosted and managed by the organization to which it is destined. The main advantage of both models is that there are no restrictions in bandwidth or resources, since all resources are exclusively intended for the sole use of the organization. It also allows organizations to manage themselves the security aspect of the cloud.

A Community Cloud is a private Cloud that is shared by organizations belonging to the same community, for examples, many departments belonging to the same University, or many companies that want to use a specific application that the provider is going to offer solely to them.

A Hybrid Cloud is composed of two or more of the Cloud models previously presented, interconnected by standard or proprietary technologies.

As for service models, the major ones are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (*Fig.* 2).

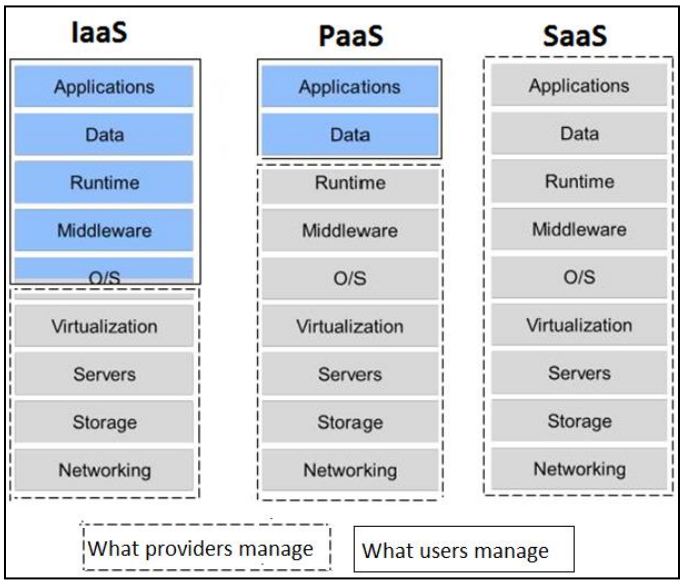

Fig. 2. Components of the main Cloud services models

IaaS provides basic virtualized resources, namely networking (network connections, bandwidth, IP addresses), virtual servers and virtual storage space. This infrastructure will be completed by clients with the various blocks necessary and used to run their applications. The provider manages the underlying infrastructure, while it is up to the user to handle anything other than the hardware part of the architecture. Although IaaS management is majorly incumbent to users, it is the model that satisfies best interoperability and portability needs, since users can compose the various blocks of the infrastructure used [10]. It is also used to build the other cloud service models. Prominent IaaS include Amazon Elastic Compute Cloud (EC2), Google App Engine, and Microsoft Azure.

PaaS is built on top of IaaS by adding a software layer to offer a development environment that can be used by clients to build and deploy their applications. It provides various development tools, such as APIs, for users to develop their applications. Clients can control the deployment and hosting environment of their applications without having to manage the underlying infrastructure. Prominent PaaS include Salesforce's Force.com, Google App Engine, and Microsoft Azure.

SaaS is arguably the most known and used cloud service model. It offers remote access to applications running in the Cloud, through various devices. Users seamlessly access "ready-to-go" applications without needing to invest or manage the underlying infrastructure, to buy software licenses, to handle updates and patches, etc. The provider is responsible for the smooth running of the applications and the maintenance of the underlying infrastructure. Prominent SaaS include Google Drive and Salesforce CRM.

Other service models are increasingly used, among which there is Network as a Service (NaaS), Logging as a Service (LaaS) for log files management, Security as a Service (SECaaS), Recovery as a Service (RaaS), etc. And one of the most promising service models is DataBase as a Service (DBaaS): a report by CISCO showed that if users had the choice to move only one application to the cloud, 25% would choose data storage [11].

Many factors contributed to the rise of Cloud Computing. The widespread use of mobile devices, for example, with their limited storage and processing capacities, led to delegating storage and processing to third parties. The various advantages that come from using the Cloud are also encouraging its rise, especially regarding elasticity, scalability, ubiquity, and cost efficiency, etc.

With Cloud Computing unlocking the barrier of storage and processing resources, developers could focus on their applications without fearing limitation. This led to an expansion of data-intensive applications where datasets are measured in terms of terabytes or petabytes, and the enhancement of Big Data.

We propose, in this work, a review of Cloud Computing solutions for Big Data storage, more precisely the model of DataBase as a Service (DBaaS).

Our paper is organized as follows. We present the definition and characteristics of Big Data in the next section. In section 3, we present some of the storage solutions for Big Data. Section 4 presents a review of several databases as a service, ensued by a discussion of the reviewed features in section 5.

II. Big Data: Definition and Characteristics

Throughout the last decade, the increasing use of new technological trends, such as Social Media, E-Commerce, E-Learning, video streaming, etc., resulted in a flood of data. For example, it is estimated that YouTube stores 1 000 TB of new data per day [12], Facebook 600 TB [13], eBay 100 TB [14], and Twitter 100 TB [15], to name but a few. Data thus generated can't be gathered, stored and analyzed easily using traditional storage and analytics tools. This data is referred to as Big Data.

One of the earliest works mentioning Big Data was in the 1990s, where Big Data is referred to as multisource, distributed data that is "*too large to be processed by standard algorithms and software*" [16]. This definition is also adopted by authors in [17], who define Big Data as "*information that can't be processed or analyzed using traditional processes or tools*" and in [18] where Big Data is a set of "*datasets which could not be captured, managed, and processed by general computers within an acceptable scope*".

Another definition of Big Data is proposed in [19] as a "phenomenon" that aims "maximizing computation power and algorithmic accuracy to gather, analyze, link, and compare large data sets" to "identify patterns in order to make economic, social, technical, and legal claims", while authors in [20] talk about "a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale", a definition that doesn't confine Big Data to the generated data only, but includes both the technology and the architecture related to data.

Cuzzocrea et al. [21] define Big Data as "*enormous amounts of unstructured data produced by high-performance applications*" belonging to various domains, from social media, to e-government, to medical information systems, etc. This data is highly-scalable and requires the applications that handle it to be highly-scalable as well.

Notorious consulting groups also attempted to define Big Data. McKinsey [22] talks about large datasets that can't be "*captured, communicated, aggregated, stored, and analyzed*" using traditional tools, while Experton Group [23] defines it as a "*collection of new information which must be made available to high numbers of users in near real time, based on enormous data inventories from multiple sources, with the goal of speeding up critical competitive decision-making processes*". Hortonworks defines Big Data as an ensemble of transaction data, interaction data, and observation data [24]. Transaction data is usually structured and stored in SQL databases, and results from applications such as ERP, CRM, transactional web applications, etc. Interaction data results from the interaction between users and applications, or users/applications with each other. This includes logs, social feeds, click streams, etc. As for observational data, it results from the Internet of Things, such as sensors, RFID chips, ATM machines, etc. Gartner [25] defines Big Data as being "high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.". This led to associating Big Data with the 3 Vs: Velocity, Variety, and Volume (Table I).

1) Volume: data sets easily reach hundreds of gigabytes, or terabytes. According to IBM, 2.5 million TB of data is created every day [26]. However, volume isn't always quantified by the size of data, but also by the number of transactions, the number of records, the number of files, etc.;

2) Velocity: data is generated and delivered at a very rapid pace. Sensors alone, for example, generate thousand TB of data every hour [27], and Wal-Mart is reported to collect 2 500 TB of customer transactions data per hour [28]. This flow of data can be in real time, near real time, batch, or streaming;

3) Variety: data comes from various sources, such as social media, blogs, business applications, sensors, mobile devices, etc. This data has different forms. It doesn't always have a specific format or respect a certain schema.

TABLE I. CLASSIFICATION OF THE 3 VS OF BIG DATA

Big Data's V	Classification	Definition
Volume	–	Data is characterized by a large volume, easily reaching Terabytes, or even Petabytes. This data deluge is due to, inter alia, the multiplication of data sources (where data is both human and machine induced), the widespread use of smartphones and applications in an increasingly connected world
Velocity	Real time	Data that is collected and then instantaneously made available for processing or analysis, such as data from GPS or ATM machines
	Near real time	Data that is collected and then is made available for processing or analysis with some delay. An example is data from Geographic information systems
	Batch	Data that is collected at a rather slow rate over a given period time of time, before being processed. Billing systems are an example of batch data
	Streaming	Data that has an interrupted flow, such as data from sensors
Variety	Structured	Data that respects a predefined data model, which makes it easy to collect and store. An example is data stored in relational databases
	Semi structured	Data that doesn't conform with a predefined formal data structure, but that has a certain level of data description, using tags (XML, HTML) or implementing a hierarchy (JSON) [29]
	Unstructured	Data that cannot be represented with a schema, such as text messages, tweets, blog entries, videos, etc.
	Hybrid	Data that combines two or more of the other data types

Other works emphasize on a fourth V, Veracity, to avoid the risk of obtaining a huge amount of poor quality data, or "data garbage" [30, 31, 32]. Authors in [32] define Big Data as "*the capture, management, and analysis of data that goes beyond typical structured data*" to "*any data not contained in records with distinct searchable fields*" and characterize it by the four Vs, namely Volume, Variety, Velocity, and Veracity. Thus, it is important to ensure good data quality by verifying its comprehensibility, completeness, and reliability. This represents a challenge because it is not always possible to validate data first-hand, especially as it is highly varied and comes from different sources, and in many cases entered by users.

Gantz et al. define Big Data in [33] as "a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis". This definition highlights a fifth V related to Big Data, namely Value, as it is not enough to store a large amount of data, but it is important to analyze it in order to extract value from it.

The NIST introduces another V, Variability, which describes any data change [34]. Thus, Big Data is defined as "extensive datasets - primarily in the characteristics of volume, variety, velocity, and/or variability - that require a scalable architecture for efficient storage, manipulation, and analysis".

Authors in [35] emphasize on the fact that Big Data has two important sides, namely the storage of large volume of data as well as the analysis of said data, while authors in [36] state that Big Data is a "*cultural, technological, and scholarly phenomenon*" that originates from the belief that the bigger the volume of data is, the more insight it would provide. It relies on technology and analysis to gather, store, analyze, and identify patterns in large datasets.

Deriving from these various definitions, we propose to define Big Data as large-scale datasets that originate from a plurality of sources at a rapid pace, aren't necessarily structured in a specific schema, can't be stored using typical database management systems, and can't be analyzed using conventional analytics tools.

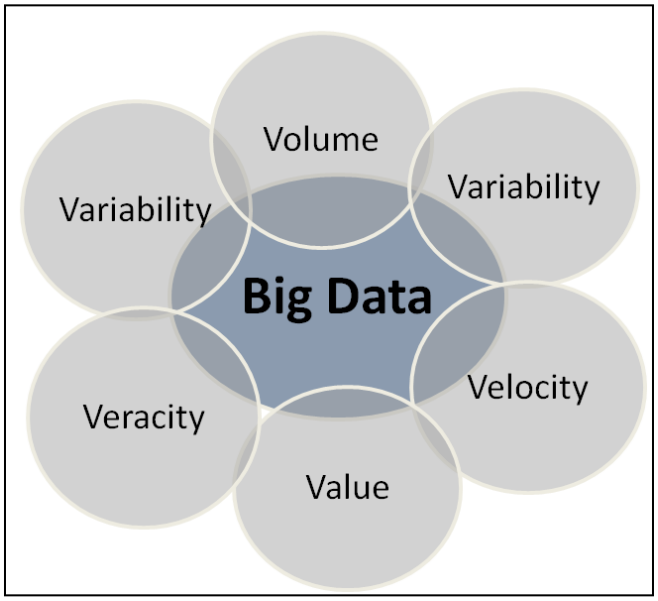

Fig. 3. Some of the V characterizing Big Data

Many factors influence the growth of the Big Data market. Horton identified seven key drivers falling into three categories, namely business drivers, technical drivers, and financial drivers [24]. Among these key drivers, there is the fact that Big Data enables innovative new business models to find adapted solutions to their needs, without requiring big investments in hardware or software, as it runs on commodity computers and offers a multitude of open source software. In fact, Big Data's influence is so tangible in business that some go as far as calling it a "*management revolution*" that challenges established conceptions of expertise, experience and management practice [37]. Many works have been trying to understand the source and nature of Big Data, and come up with new ways to address the challenges encountered in its different phases, from data collection to archiving, all through storage and analytics. Each one of Big Data's lifecycle's phases called for new solutions to be developed, as shown in *Fig.* 4.

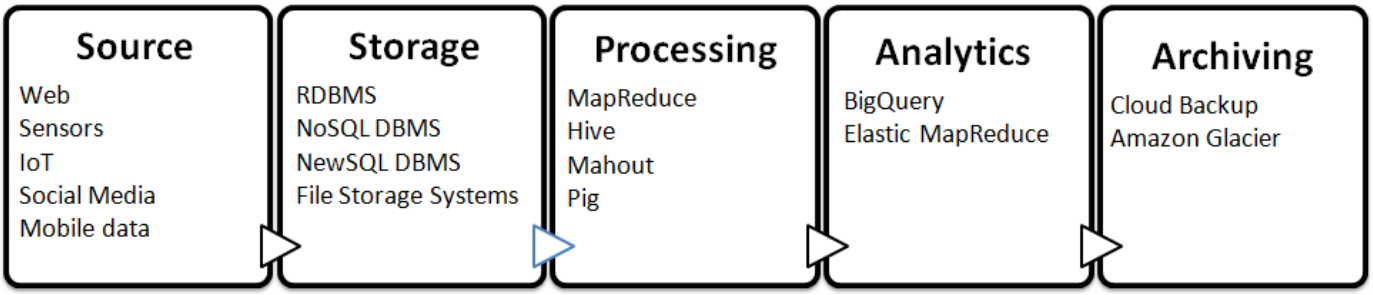

Fig. 4. Big Data's lifecycle

One of the challenges that rose with the growth of Big Data is the storage of the huge volume of generated data. We present in the next section the main storage systems used.

III. Big Data Storage

One of the challenges that face organizations dealing with Big Data is how and where to store the tremendous amount of data.

The most widespread data management technology is relational database management systems (RDBMS). However, with the rise of Big Data, these RDBMS became unfit for large, distributed data management, especially regarding data Velocity and Variety, since they require data to respect a relational schema before being imported in the database, while Big Data is about managing data of various formats and flow rate (streaming, real-time, etc.). Regarding data Volume, RDBMS are required to be distributed over multiple clusters, sometimes geographically distant. While most proprietary RDBMS scale to large amounts of data, open source ones, such as MySQL and PostgreSQL, are still far behind [38].

First approaches tried to adapt traditional RDBMS by using replication to scale reads, adding a caching layer, using vertical scaling (scale up) or horizontal scaling (scale out) to cope with said volume. Vertical scaling adds more resources to the machine that stores data. This needs powerful machines and can be expensive. Moreover, there is a physical storage limit that can't be exceeded (the current maximum size of a hard disk drive is 8 TB, with the project to reach 10 TB by 2017 [39]). Horizontal scaling, on the other hand, adds more machines to cope with the increasing data volume. Now that the cost of hardware is significantly less than it used to be, it is more interesting to add new servers to the cluster, whenever resources are needed. However, users would ultimately need to shard data across many clusters, which they would have to manage in the application layer.

A real-world example is the expansion of Twitter. Launched in 2006, Twitter knew an exponential growth leading to an average of 500 million tweets per day [40]. In order to manage the expansion of data volume, Twitter had to rethink its architecture, which was relying on MySQL for data storage, when sharding couldn't keep up with the increasing data traffic. This called for developing new adapted solutions used internally by Twitter, such as T-Bird and Snowflake [41]. In general, alternative database solutions are increasingly used in order to provide advantages in terms of performance, scalability, and suitability for Big Data environments. Among these solutions, there are NoSQL databases, NewSQL databases, and file storage systems like HDFS [50] and GFS [49].

A. NoSQL database systems

The term NoSQL, or Not Only SQL, was first coined in 1998 as the name of a relational database, based on the Unix Shell, and conceived to give better flexibility and optimize the use of resources compared with existing relational databases [42]. It was revived in 2009 with the rise of Cloud Computing and the presentation of Google's Bigtable [43], and has since been generalized to describe databases that model, store, and retrieve data in a different way than traditional relational databases. Many NoSQL databases are well-known today, such as MongoDB, HBase, Facebook's Cassandra, Linkedin's Voldemort, etc. One of the main features of NoSQL databases is that they are schema free, which means that the structure of data can be easily and quickly modified without needing to rewrite tables. This aims to overcome the inflexibility of traditional relational databases schemas. And while many NoSQL databases don't implement certain relational functionalities, such as JOINs, ordering, and aggregation, many offer support for SQL-like querying.

While relational databases permit handling data storage and management simultaneously, especially with implemented SQL-querying interfaces, NoSQL databases handle them separately. Data storage is done according to the adopted data model (key-value, document, etc.) with a primary focus on scalability. Data access is done using APIs. This renders NoSQL databases flexible for data modelling and easy for application development and deployment updates [44].

Relational databases guarantee ACID (Atomic, Consistent, Isolated, and Durable) transaction properties. However, CAP theorem (*Fig. 5*) states that at most two out of the three properties (Consistency, Availability, and Partition tolerance) can be achieved simultaneously in distributed environments [45]. While RDBMS do well on Consistency and Availability, they don't scale well. The main idea behind NoSQL databases is to loosen up on one of these two properties, namely Consistency and Availability, in order to enhance scalability. They provide what can be called BASE (Basically Available, Soft state, and Eventually consistent) [46] properties, in contrast with ACID. NoSQL database systems differ in which of the two properties they loosen, and how much they do loosen it. Many however provide eventual consistency to ensure high scalability and availability.

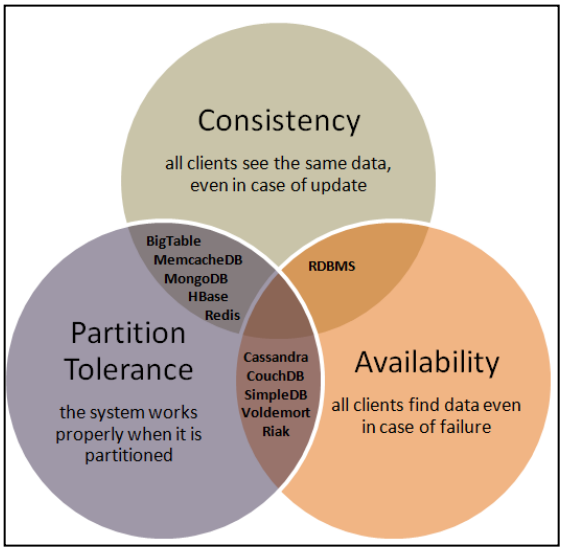

Fig. 5. The CAP theorem

NoSQL databases have many data models: Key-Value, Document, Column, and Graph, as shown in *Table II*.

Key-value databases store data as a collection of (*key*, *value*) pairs where a unique identifier, *key*, is used to access and retrieve data. They are schema-free, as values are independent from each other, with no restriction on their nature. As data is completely opaque to the system, the only way to access and retrieve it is by using the unique *key*. They support basic insert, read, and delete operations. Most are persistent while others like Memcached cache data in memory. Notorious examples include Redis, Memcached, and DynamoDB.

Document databases store data as documents that are based on a specific encoding (JSON, BSON, XML, etc.) and identified by a unique "ID". Document databases being schema-free, documents can store attributes of any kind. Most document databases generally support more complex data (such as nested documents) and offer more indexing and querying functionalities, but relatively less performance, than Key-Value ones.

Column databases are modelled after Google's Bigtable [43]. They store data using tables (columns and rows) but without any association between them. Columns consist of a unique identifier, a value, and a timestamp used for versioning. They are grouped in column families that have to be predefined, which affects flexibility.

Graph databases store data nodes interconnected with edges where each node and edge consists of key-value pairs. This allows graph databases to store not only data, but also relationships between data nodes. They are the tool of choice when dealing with heavily linked data. Some examples include Neo4J database, which supports ACID properties, and OrientDB.

Although they differ in their data model, all NoSQL databases allow a relatively simple storage of unstructured, distributed data and achieve high scalability. They are best adapted for applications that don't use a fixed schema, or don't require ACID operations, and for intensive read and update OLTP workloads [47].

B. NewSQL database systems

NewSQL originated from the affirmation that the relational model can be implemented to scale by retaining its key aspects and removing some of the general purpose ones [48]. NewSQL databases aim to answer Big Data storage needs, especially regarding volume and scalability, while providing the traditional functionalities of relational databases, especially regarding ACID transactions, querying operations such as JOINs and aggregations, etc. They are an attempt to realize the three properties featured in the CAP theorem, proving that Consistency and Availability can be achieved simultaneously in distributed environments.

NewSQL databases provide an SQL query interface, and clients (users and applications) interact with them the same way they interact with relational databases. They manage read/write conflicts using non-lock concurrency control [48].

TABLE II. NOSQL DATA MODELS

Data model	Definition	Use case	Advantages	Limitations	Examples
Key-Value	Stores data as a collection of (key,value) pairs	Applications with only one kind of object where search is performed based on one attribute	Simple to use	Relationships between data must be explicitly managed in the application layer	Memcached Redis DynamoDB
Document	Stores data as encoded documents	Applications with many kinds of objects where search is done on multiple attributes	Management of complex data structures	Relationships between data must be explicitly managed in the application layer	CouchDB MongoDB
Column	Stores data as columns consisting of a key, a value, and a timestamp	Applications with many kinds of objects where search is done on multiple attributes and that need data to be partitioned both horizontally and vertically	Allows high throughput and low latency	Less flexibility	Bigtable HBase Cassandra
Graph	Stores linked data as graphs	Applications that handles heavily connected data (social networks, location based services, etc.)	Seamless manipulation of graphs	Relatively high complexity and less scalability	Ne04j GraphDB OrientDB

Many NewSQL solutions extend existing relational databases to support high scalability, like Infobright, TokuDB, and MySQL cluster NDB, which are all built on MySQL. Other solutions retain existing relational databases and add a middleware for achieving high scalability through shading or clustering, such as ScaleArc, ScaleBase, dbShards, etc. There are also solutions that were developed from scratch to provide relational features in distributed environments, such as NuoDB.

NewSQL databases are relatively new compared to NoSQL ones. They are most adapted to use case scenarios that call for relational databases with more scalability. They try to combine the advantages of both relational and NoSQL databases, as detailed in *Table III*.

TABLE III. COMPARISON OF RELATIONAL, NOSQL, AND NEWSQL DATABASES

Feature	Relational databases	NoSQL databases	NewSQL databases
Relational schema	Yes	No	Yes
SQL Querying	Yes	No	Yes
ACID transactions	Yes	No	Yes
Big Data compatibility	No	Yes	Yes
Availability	Yes	Yes	Yes
Strong Consistency	Yes	No	Yes
Scalability	No	Yes	Yes

C. File Storage Systems

File storage systems are another solution to deal with large volume of data in distributed environments. The major ones are Google File Storage (GFS) [49] and Hadoop Data File Storage (HDFS) [50].

GFS is a scalable distributed file system developed by Google to meet the needs of its large distributed data-intensive applications [49]. It is designed for environments that are prone to failures, that manipulate huge data files by frequent read/append operations, and that need to process data in batch rather than in real-time. Thus, it is highly fault-tolerant and reliable, and emphasizes on high throughput rather than low latency.

GFS has a master-slave architecture (*Fig.* 6), a typical cluster consisting of one master and many chunkservers to which clients access directly after consulting the master. The master divides each file into 64 MB chunks and manages the mapping and replication of said chunks through the different chunkservers.

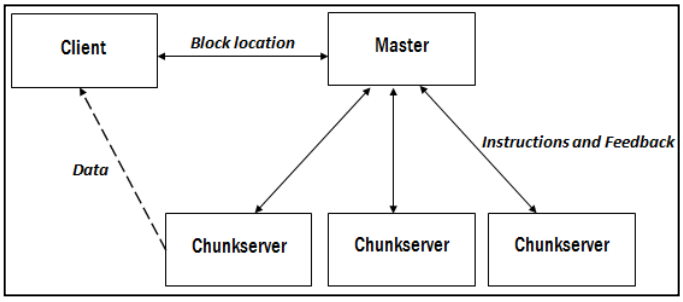

Fig. 6. GFS architecture

GFS maintains multiple replicas of each file, which leads to higher reliability and availability.

HDFS [50] is an open source implementation of GFS. It is part of the Apache Hadoop, an open source framework for distributed storage and distributed processing of large data sets. The biggest clusters implementing Hadoop are composed of 45 000 machines and store up to 25 petabyte of data [51].

HDFS is one of the four modules composing Hadoop, which are Hadoop commons, Hadoop YARN, and Hadoop MapReduce, the open source implementation of Google's Map/Reduce for the parallel processing of large distributed data.

HDFS is implemented based on the fact that moving computation is cheaper than moving data, providing interfaces to client applications to move where data is stored. Like GFS, HDFS has master-slave architecture (*Fig*. 7) consisting of a single master node, *NameNode*, and a slave for each node in the cluster, *DataNode*.

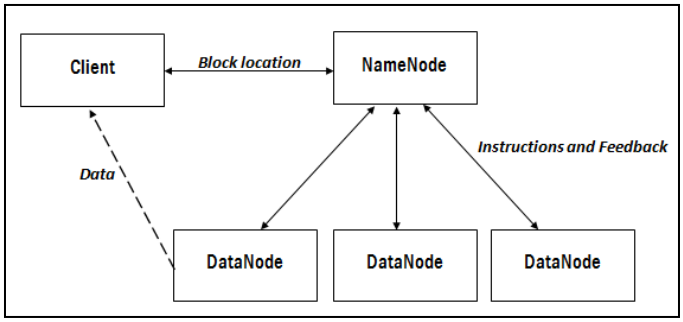

Fig. 7. HDFS architecture

The NameNode is the coordinator of HDFS. It divides files into fixed-sized blocks and maps them to DataNodes, and client applications consult it to know where to access data. The DataNode manages data storage in the node where it is installed. It can also create, delete, and replicate blocks when instructed by the NameNode.

The adoption of NoSQL, NewSQL and File Storage systems is mainly driven by six key factors, regrouped in the acronym SPRAIN [52]. These key drivers, which are the weak points of traditional RDBMS, are Scalability, Performance, Relaxed consistency, Agility, Intricacy, and Necessity. And while these new database systems are becoming the tool of choice to meet the demands of Big Data applications, it can be complicated and costly to run and manage them, especially at scale. One solution is to move them to the Cloud in order to take full advantage of the elasticity, scalability, availability, and performance of the latter, and meet the ever-growing storage and processing requirements of Big Data applications. And one of the currently most adapted Cloud Computing models to Big Data storage requirements is DataBase as a Service (DBaaS), as it can combine many of the aforementioned storage systems to offer scalable, on-demand, pay-as-you-go storage resources to organizations without any upfront investment.

We present, in the next section, a review of several DBaaS and discuss their suitability for Big Data storage.

IV. DATABASE AS A SERVICE (DBAAS) FOR BIG DATA

An ever growing number of companies found themselves swamped with the large amount of data generated and stored for different purposes (user based preference suggestions, business analysis...). Storing and retrieving data becomes a costly and complex operation, involving investments in infrastructure and database managers. It is only normal then that the question of outsourcing data was one of the earliest to surface with the emergence of Cloud Computing, which led to the DataBase as a Service (DBaaS) model.

DBaaS can be simply defined as "a paradigm for data management in which a third party service provider hosts a database and provides the associated software and hardware support" [53]. Companies using this model outsource all database management operations, from installation to backups, to the provider, and focus on developing applications. They can access their databases instances on-demand, using querying interfaces or programming tools.

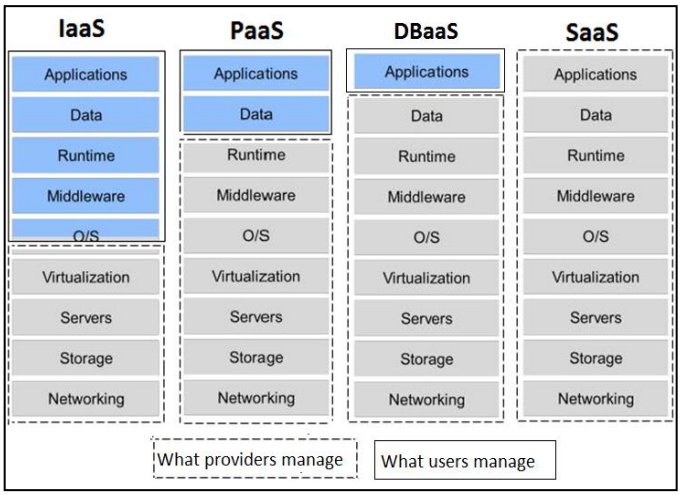

Fig. 8. DBaaS components

The increasing use of Cloud Computing, and especially SaaS, called for rethinking the persistency layer. The inherent characteristics of cloud computing, such as elasticity, scalability, self-service, and easy management make traditional RDBMS not fully adapted for applications that run in cloud environments. Early solutions tried extending existing DBMS to support high-scalability, but it only led to complex solutions with poor performance [54]. Leader IT operators, such as Google, Yahoo!, and Facebook, chose to implement their own data management solutions, respectively Bigtable, PNUTS, and Cassandra. Various other databases provided as DBaaS were developed from scratch to integrate the advantages of the cloud, with the exception of few providers who offer established relational or NoSQL databases, such as MySQL, PostgreSQL, MongoDB, and Redis, as a service.

Database as a Service (DBaaS) in one of the Cloud Computing models that is most suitable for Big Data. In this model, it is possible to use a database as a service and benefit from the high-scalability and storage capacity offered by the Cloud, without having to install, maintain, upgrade, backup or manage the database or the underlying infrastructure.

DBaaS is a different concept from the concept of cloud databases, which is beyond the scope of our paper. In this concept, users can either upload their machine image, with the database installed, to the cloud infrastructure or use a ready one offered by the provider. In both scenarios, the various database management operations are incumbent to users. Datawarehouse Cloud solutions are also beyond the scope of this paper.

We propose to review some of the most prominent databases that are DBaaS and discuss their adaptability to Big Data uses.

A. Cloud Bigtable

Cloud Bigtable is a DBaaS based on Bigtable [43], a highly-scalable, distributed, structured, and highly-available column database developed by Google that has been used internally since 2003 to store the data of numerous Google projects (Google Finance, Google Analytics, Google Earth, etc.). Bigtable was made publically available as Cloud Bigtable in May 2015 [55].

Bigtable stores data in tables, which are "*sparse, distributed, persistent sorted*" maps. [43]. These tables are sharded into tablets containing blocks of adjacent rows. Each cell is referenced by three dimensions: a row key, a column key, and a timestamp.

A row key is an arbitrary string and is the unit of transactional consistency in Bigtable. Rows with consecutive keys are grouped into tablets, which are the unit of distribution and load balancing. A column key is also an arbitrary string, and column keys are grouped into columns families, the unit of access control. Timestamps are used to manage data versioning. A cell can store different versions of the same data, each referenced by a timestamp. Older data is garbage-collected depending on the user's specifications.

Bigtable relies on Google File System (GFS), a scalable distributed file system presented in Section 4, for storing data in SSTable [43] file format. An SSTable is a file of key/value string pairs that is sorted by keys. It is used to map keys to values. Bigtable also uses Chubby, a highly-available and persistent distributed lock service, for synchronizing data access [56]. A Chubby service has four replicas and one master replica. The latter is used to serve requests. Bigtable architecture is composed of one master server, many tablet servers, and a library, as shown in *Fig.* 9.

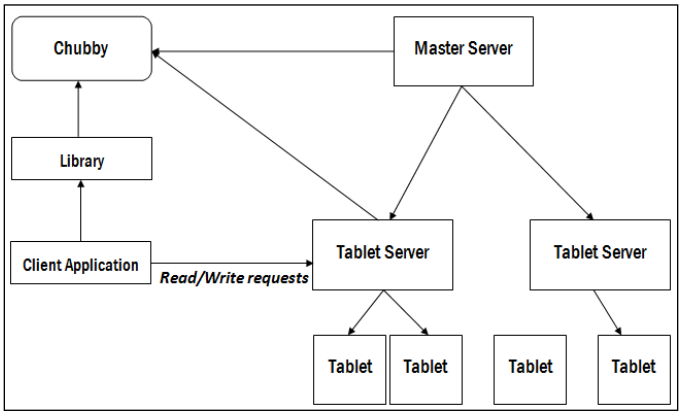

Fig. 9. Bigtable architecture

The library is linked to client applications and is used to retrieve the location of tablets. The master server performs many tasks: assigning tablets to tablet servers, load balancing, detecting new or expired tablets, detecting schema changes, and GFS garbage collection. A tablet server is responsible for managing a set of tablets, receiving read /writes requests from client applications, serving client requests that are directed to the tablets it manages, and splitting tablets when their size exceeds 1 GB.

Each tablet is assigned to one tablet server at a time. Tablet servers use Chubby to obtain an exclusive lock on the tablets they manage. The master server consults Chubby to discover tablet servers.

While being manipulated, tablets are stored in memory in a buffer called memtable. When the size of a memtable reaches a certain level, it is stored as an immutable SSTable in GFS. Tablet servers perform write operations on tablets in memtable, and read operations on views obtained from merging SSTables and the memtable.

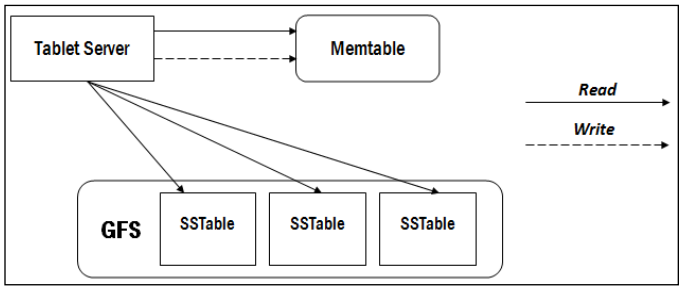

Fig. 10. Management of Read and Write operations

Bigtable maintains a high level of consistency. Reads are strongly consistent, since SSTables are immutable. As for writes, memtables perform a row copy each time there is a write operation in a row, ensuring that updates are seen by reads.

Client applications can connect to Cloud Bigtable using the Cloud Bigtable HBase client. The latter supports HBase shell, which can be used to perform queries and administrative tasks.

Cloud Bigtable was designed for Big Data applications that handle terabytes of data in clusters composed of thousands of nodes. Google recommends it for applications where the volume of data exceeds 1 TB. For Big Data applications with less than 1 TB data volume, Google recommends another solution, namely Cloud Datastore.

B. Cloud Datastore

Cloud Datastore is a NoSQL, schemaless, highly-scalable, and highly-reliable database for storing non-relational data developed by Google as a part of the App Engine. The main motivation for its development is to answer the need for high-scalability that couldn't be met by traditional relational databases. It supports basic SQL functionalities, including filtering and sorting. Other functionalities like table joins, sub queries and flexible filtering are not supported. Cloud Datastore is based on another Google's solution, namely Megastore, which is built on Bigtable. Thus, Cloud Datastore architecture is as shown in *Fig.* 11.

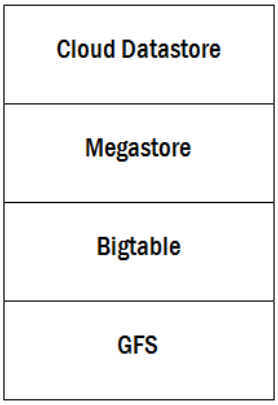

Fig. 11. Cloud Datastore architecture

Megastore [57] is a distributed data store that combines the scalability of NoSQL databases and some key features of relational databases, especially in terms of consistency and ACID transactions. It allows users to define tables just like in traditional SQL databases, and then maps them to Bigtable columns. It is used by more than 300 applications within Google [58].

Megastore ensures strong consistency. It replicates data across multiple geographically distributed datacenters using an algorithm based on a distributed consensus algorithm, Paxos [59], for committing distributed transactions. It also implements two-phase commit (2PC) [60] for committing atomic updates. Unlike 2PC, Paxos doesn't require a master node for committing transactions. Instead, it ensures that only one of the proposed values is chosen and, when it is, that all the nodes forming the cluster get the value. Thus, all future read and/or write access to the value will give the same result.

For each new transaction, Megastore identifies the last transaction committed and the responsible node then uses Paxos to get a consensus on appending the transaction to the commit log. Megastore is built on Bigtable to overcome the difficulty to use in applications that have relational schemas, or that need to implement strong consistency [86]. An amelioration to Megastore is Spanner [86], a highly-scalable, globally-distributed, semi-relational database where queries are done in an SQL-like language and offers better write throughput. Though Spanner is not offered as a service to developers, it is used internally by Google as the backend of F1, Google's distributed RDBMS supporting its online ad business. However, there is a project for building an open source version of Spanner, CockroachDB.

Cloud Datastore relies on Megastore to support transactions, ensuring strong consistency. The entity data, which is the equivalent of a row in relational databases, is written in two phases: the commit phase and the apply phase. In the commit phase, data is recorded in the transaction logs of a majority of replicas. It is also recorded in the transaction logs of all replicas in which it was not recorded and that are not up-to-date. In the second phase, the entity data and its index rows are written in each replica.

Cloud Datastore also relies on Bigtable's automatic sharding and replication to ensure high-scalability and

reliability. Performance is ensured by reducing lock granularity and allowing collocation of data to minimize the communication between nodes.

In Cloud Datastore, client applications perform queries and manipulate data using APIs, third-party implementations of the Java Data Objects (JDO) and Java Persistence API (JPA), or third-party frameworks such as Objectify, Slim3 or Twig.

Google intents to prove, with Cloud Datastore, that scalability can be achieved while keeping some features of traditional relational databases, especially transactions, ACID semantics, schema support, etc. It thus provides a highly-scalable and reliable cloud database that is adequate for Big Data applications that need to implement strong consistency.

C. Cloud SQL

Cloud SQL is a fully-managed, highly-available MySQL database hosted in Google's cloud and offered as DBaaS. It allows users to easily create, run, and manage MySQL databases in Google's infrastructure, with a promise of 99.95% uptime SLA [61]. It is simple to use and gives users the possibility to control the geographical location where their data is stored, the RAM capacity they need (ranging from 0.125 to 16 GB), the billing plan they prefer (based on the number of hours the database is accessed or based on the number of days the database exists), the backup frequency, the replication mode, the connection encryption mode, etc. Many companies opted for migrating their data into Cloud SQL, such as CodeFutures and KiSSFLOW.

Cloud SQL is distributed, and it replicates data across multiple datacenters in order to be fault-tolerant, using both synchronous and asynchronous replication. It supports all MySQL features with some exceptions (user defined functions, LOAD_FILE function, installing and uninstalling plugins). It is accessible via MySQL clients, standard MySQL database drivers, App Engine applications written in Java or Python, and third-party tools such as Toad for MySQL.

In Cloud SQL, the maximum size of an instance is 10 GB, with a total size limit of 500 GB. Moreover, it doesn't scale automatically, but it is up to the user to handle scalability, and it is not adapted to applications where data schema changes frequently. This makes Cloud SQL unsuited for Big Data applications.

D. Cloudant

Cloudant [62] is a scalable, distributed, NoSQL database as a service provided by IBM, with the assurance, through SLAs, of uninterrupted, highly-performant access to data. Cloudant's infrastructure consists of over 35 datacenters distributed in more than 12 countries all over the world. Data is stored in server nodes, grouped into clusters that can either be multi-tenant or single-tenant. Cloudant also offers users the possibility to deploy it on-premise, or to select other hosting providers such as Rackspace, SoftLayer, and Microsoft Azure. This is done in the optic of bringing Cloudant near to users' data, in the case where it is already hosted in the cloud. As for the billing, it is adaptable to the growth of the user's applications, offering a "pay-as-you-grow" billing plan.

Cloudant is interoperable with many open source solutions, which enhances its capabilities and features, as shown in *Fig.* 12.

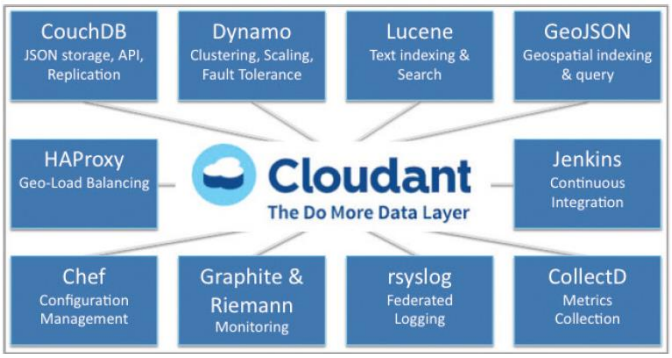

Fig. 12. An overview of Cloudant interaction with various open source solutions [62]

Cloudant is based on Apache CouchDB, with some additional features regarding data management, advanced geospatial capabilities, full-text search, and real-time analytics. It stores data as JSON documents (*Fig.* 13), which is a lightweight data-interchange format that is built on a collection of name/value pairs, and an ordered list of values.

```
{"menu": {
    "header": "SVG Viewer",
    "items": [
        {"id": "Open"},
        {"id": "OpenNew", "label": "Open New"},
        null,
        {"id": "ZoomIn", "label": "Zoom In"},
        {"id": "ZoomOut", "label": "Zoom Out"},
        {"id": "OriginalView", "label": "Original View"},
        null,
        {"id": "Quality"},
        {"id": "Pause"},
        {"id": "Mute"},
        null,
        {"id": "Find", "label": "Find..."},
        {"id": "FindAgain", "label": "Find Again"},
        {"id": "Copy"},
        {"id": "CopyAgain", "label": "Copy Again"},
        {"id": "CopySVG", "label": "Copy SVG"},
        {"id": "ViewSVG", "label": "View SVG"},
        {"id": "ViewSource", "label": "View Source"},
        {"id": "SaveAs", "label": "Save As"},
        null,
        {"id": "Help"},
        {"id": "About", "label": "About Adobe CVG Viewer..."}
    ]
}}
```

Fig. 13. An example of JSON-formatted documents

JSON documents are accessed using an HTTP-based RESTful API. Querying is done using Cloudant query, a declarative system based on MongoDB's declarative query. Cloudant assigns a unique identifier to each JSON document and uses a MapReduce-based framework to query data. Users write MapReduce functions in JavaScript, where the Map function defines which JSON documents are concerned by the Reduce function that specifies the operations to perform. Then Cloudant distributes the MapReduce functions to all nodes forming the cluster. It is noted that Cloudant allows MapReduce functions to be "chainable", meaning that the output of a MapReduce job can be used as input for other MapReduce jobs in the chain.

Data distribution is done by multi-master replication, ensuring a high fault-tolerance, and reducing latency by connecting users to data that is geographically closest. Users can replicate data not only through all nodes forming the cluster, but also to CouchDB, being able to benefit from an open source data storage solution to increase their datacenter size.

Cloudant is adapted to Big Data uses, especially for web, mobile, and the Internet of Things [63]. It is also suitable for applications that deal with unstructured data or that need to synchronously replicate data across multiple datacenters.

E. MongoLab

MongoLab is a fully-managed, highly-performant, highly-available MongoDB database offered as DBaaS that runs in major cloud infrastructures: Amazon WS, Google Cloud Platform, Rackspace, and Windows Azure, etc. It is also possible to integrate it with users' applications that run on other PaaS providers' platforms, like AppFog, Heroku, OpenShift, etc.

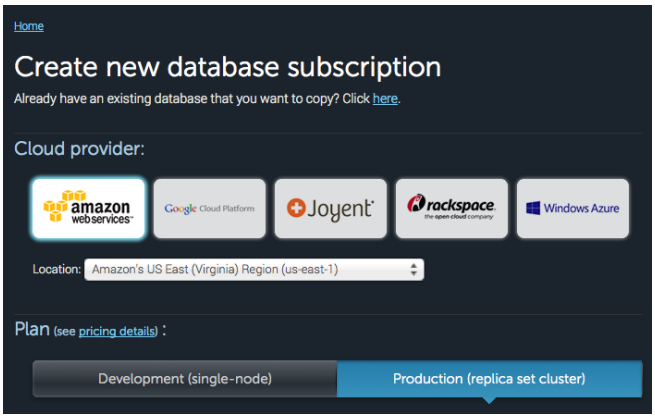

Fig. 14. MongoDB control panel

MongoDB is a schema-free, scalable document database that offers, along with the basic CRUD functions of traditional relational databases, many features such as indexing, aggregation, session-like data expiration management, native support of geo-spatial indexing, etc. Other features specific to relational databases, such as JOINs, are not supported.

MongoDB stores data as BSON documents, a lightweight, binary interchange format based on JSON. BSON represents data efficiently, optimizing storage space and scan speed, and rendering encoding and decoding data simple and fast. Data access, data requests and background management operations are performed by *mongod*, the primary daemon process of MongoDB.

Users can browse their data stored in MongoLab via the management portal, or the MongoDB shell, which is an interactive JavaScript shell. Applications can be connected to the MongoLab databases using a MongoDB driver, or MongoLab RESTful APIs.

MongoDB defines its own query language. Users can perform ad hoc queries using two functions like find() and findOne() that return a subset of documents. Queries can be performed with complex criteria (such as ranges or negatives), conditions, sorting, embedded documents, etc. It is also possible to use indexing, like in relational databases, which allows performing faster queries. In addition, MongoDB offers a wide range of commands to be used to manage servers and databases.

MongoDB handles replication using a master-slave strategy. Users define a replica set, which is composed of a primary server and many secondary servers. The primary server gets the requests from applications and users, and secondary servers store copies of the data contained in the primary server. This way, if the primary server becomes unavailable, one of the secondary servers is chosen by its peers to replace it. MongoDB also offers an interesting feature, slave delay, which sets a secondary server to lag by a predefined number of seconds to allow retrieving an earlier version of damaged data.

Scalability in MongoDB is ensured by autosharding. Mongos, MongoDB's routing service, is used to keep track of the location of data in the different shards. Applications connect to Mongos and send their queries the way they'd do with a stand-alone MongoDB instance, as shown in *Fig.* 15. This allows MongoDB to handle higher throughput in read and write operations than what a stand-alone instance can handle [64].

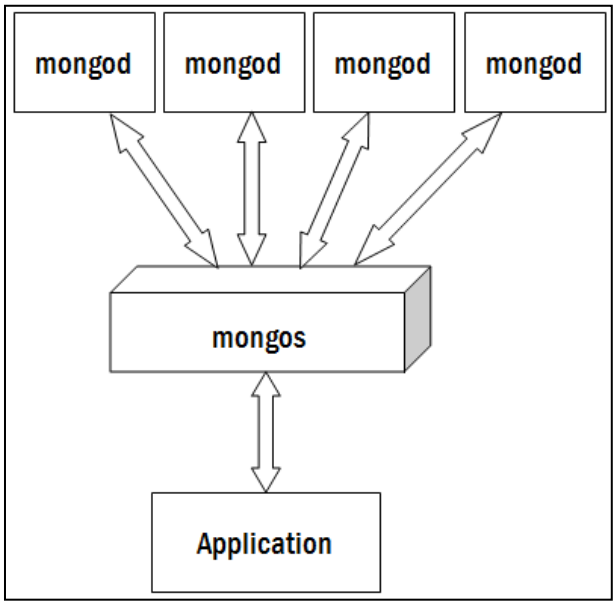

Fig. 15. Access by applications to sharded data in MongoDB

MongoDB's design makes it suitable for storing large volumes of heterogeneous, evolving collections of data.

F. Morpheus

Morpheus is a fully managed, highly-available DBaaS that provides access to SQL (MySQL), NoSQL (MongoDB), and cache (Redis) databases. It also offers a fully managed access to Elasticsearch, a full-text search engine.

As mentioned above, Morpheus offers a fully managed access to four databases. MongoDB and MySQL have been presented in previous chapters. We will present Elasticsearch and Redis.

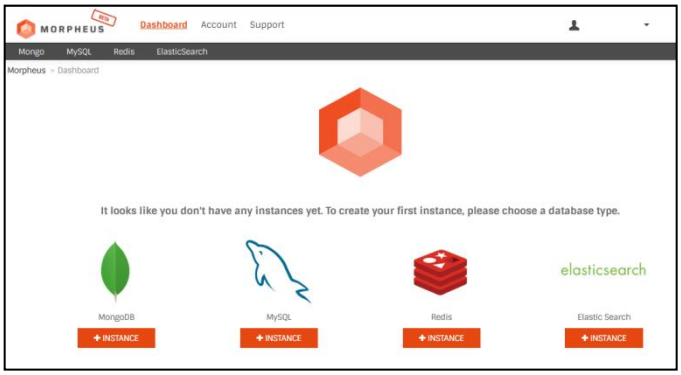

Fig. 16. Available databases in Morpheus[41]

Elasticsearch is an open source distributed, scalable, highly-available full-text search engine. It is built on Apache Lucene, an open source library for data retrieval.

Redis is an open source key-value cache and store that keeps data in memory for faster treatment, handling over 100 000 read/write operations per second [65]. Redis can also store data on hard disk asynchronously using snapshots or append-only logs.

Morpheus allows users to easily select one of the available databases and create an instance with a size ranging from 1 to 200 GB, as shown in *Fig.* 17. It supports many versions of each database and gives users the possibility to select one. Users can create many instances using disparate databases.

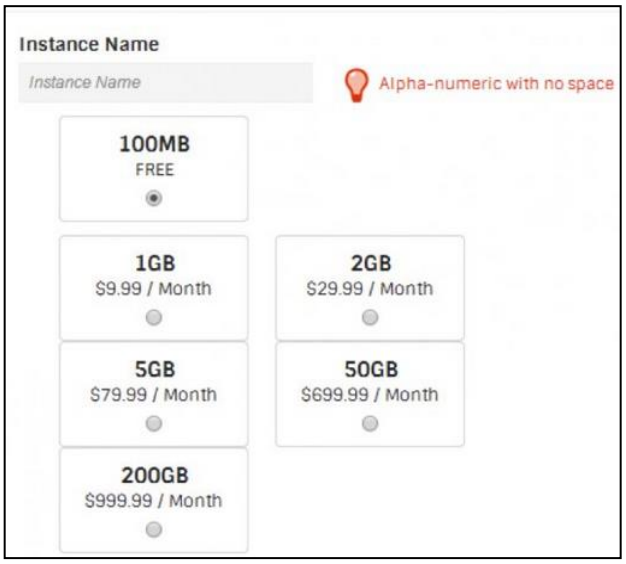

Fig. 17. Available instance sizes on Morpheus and their cost[41]

Morpheus uses Solid State Drives (SSD) for data storage, which improves the speed of data access. It also uses Amazon's datacenters. Replication is done using a master-slave strategy to ensure availability and fault-tolerance. Scalability is achieved using autosharding.

Use cases show that Morpheus allows creating up to 2000 instances, with a total data size of 400 TB [66]. This, along with its scalability and high availability, makes Morpheus suitable for Big Data uses.

G. Postgres Plus Cloud Database

Postgres Plus Cloud Database (PPCD) [67] is a fully-managed, highly-performant, highly-available, scalable access to PostgreSQL, an object-relational database management system. It supports relational databases ACID transactions, as well as NoSQL databases features.

The architecture of PPCD is composed of one server, and clusters, as shown in *Fig.* 18.

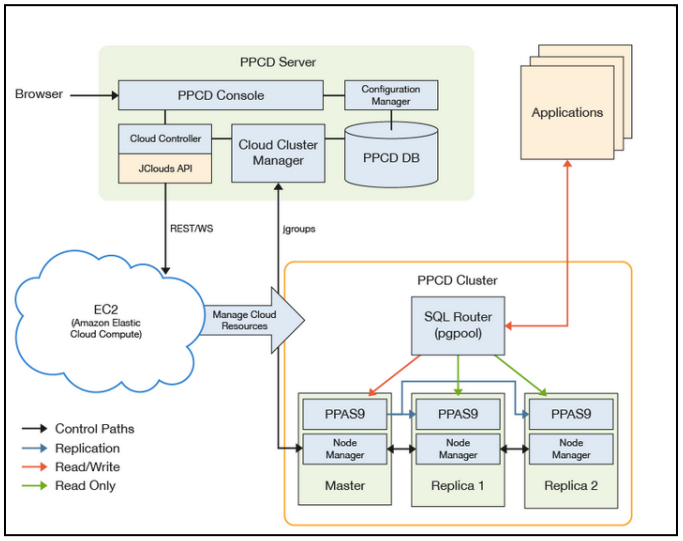

Fig. 18. The architecture of PPCD [67]

This architecture is for each cloud region. Users in a cloud region connect to a centralized console, the PPCD Console, to create clusters. The PPCD server deploys these clusters to the instances hosted by a Cloud provider (Amazon's EC2 [67], Amazon's VPC [68], etc) and connects to the cloud using JCloud APIs. The console uses jgroups, a toolkit for nodes messaging, to communicate with the various Cloud environments where clusters are deployed.

PPCD ensures reliability and availability using master-slave replication. The first database deployed by the console is designed as the master database, the other replicas are slaves and used for read-only operations. So PPCD clusters consist of a master and one or more replicas. They have built-in load balancers that receive incoming requests from applications and distribute them through the nodes.

The PPCD server manages the instances in the clusters using the Cloud Cluster Management (CCM). In case of failure, the CCM initiates automatic failover.

Automatic failover is implemented in two ways, as shown in *Fig.* 19. One way is to switch to a replica, which minimizes downtime, another is to migrate data from the failed master to a new one, which minimizes data loss.

PPCD offers, as a service, PostgreSQL databases that are hosted in the cloud, especially using Amazon's WS. This lets PPCD benefit from Amazon's powerful resources and makes it suitable for Big Data applications.

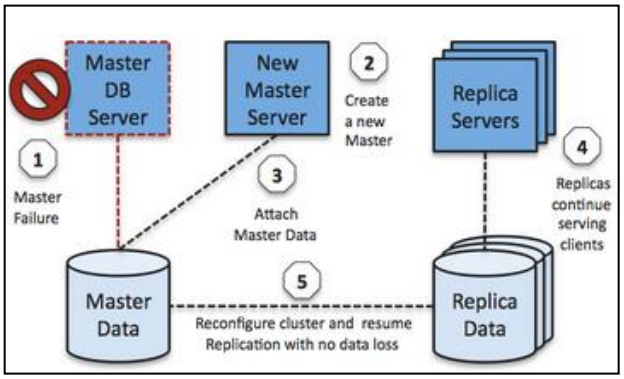

Fig. 19. Automatic failover scenario [67]

H. SimpleDB

SimpleDB is a highly available, scalable, schemaless non-relational document database that is part of Amazon's Web Services. It provides many of the functionalities provided by relational databases as a service in the cloud. SimpleDB is designed to run on other web services provided by Amazon. Developers that use SimpleDB can run their applications using Amazon's Elastic Compute Cloud (EC2) and store their data in Simple Storage Service (S3).

Data is structured in domains, which are the equivalent of tables in relational databases. Each domain is composed of attributes and items, and each attribute has one or more values for a given item, as shown in *Fig.* 20. Currently, users can store up to 10 GB of data per domain, and can create up to 250 domains [69]. However, they can request to create additional domains if needed.

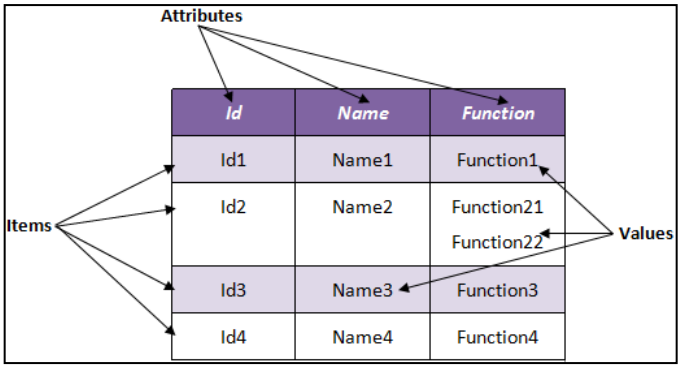

Fig. 20. Example of a domain in SimpleDB

SimpleDB provides a group of API calls to build applications [69], such as *CreateDomain* for creating domains, *DeleteDomain* for deleting domains, *PutAttributes* for adding, modifying, and removing data in domains, etc. Querying domains is done using an SQL-like *Select* query, but multi-domain querying is not supported.

SimpleDB implements automatic data indexing for a better performance. To ensure high-availability, asynchronous replication is implemented, and multiple copies of the domain are done after a successful write. Two consistency options are supported for read operations, namely strong consistency and eventual consistency. Strong consistency requires a majority of replicas to commit writes and acknowledge reads. Eventual consistency asynchronously propagates writes through the nodes, and any replica can acknowledge reads. Automatic data sharding is not supported, so users have to manually partition their data across multiple domains for better scaling. SimpleDB is optimized for parallel-queries.

SimpleDB is designed for fast reading and is a simple way to store data in a schema-free database offered as a DBaaS. However, it has many drawbacks, such as the storage limit of 10 GB per domain, the maximum attribute values of 256 per item, the limit response size of 1 MB per query [70], the performance setback due to the automatic indexing of all attributes, etc. For all these reasons, Amazon built upon SimpleDB to develop DynamoDB, which can be considered an improved version of SimpleDB that is more adapted to Big Data applications.

I. DynamoDB

Amazon's DynamoDB is a fully-managed, highly-available, highly-scalable, distributed NoSQL database. It is an answer to Amazon's need of a performant, reliable, efficient database able to scale up to meet the ever growing load on their servers, which simultaneously serve, at peak times, more than tens of millions of customers [71], with all the economical issues at stake. DynamoDB is fast and flexible, and supports document and key-value data models.

Since strong consistency and high availability are complementary (according to the CAP theorem), and one must be sacrificed in order to achieve the other in distributed environments, Amazon chose to privilege high availability. Thus DynamoDB supports eventual consistency, which is achieved by asynchronously propagating updates, and considering each update to be a new version of data. This versioning is done by using vector clocks [72]. DynamoDB uses sloppy quorum, a quorum-based technique, and hinted handoff, a decentralized replica synchronization protocol, to achieve consistency among replicas while ensuring availability in case of server failures [71].

Conflicts during updates needed to be addressed too. The classical approach is to resolve these conflicts during writes, committing them only when the majority of replicas can be reached. To be more suitable for Amazon's services, where rejecting a write can be prejudicial from the customer's perspective, DynamoDB opts for resolving conflicts during reads. However, DynamoDB leaves it up to developers to implement their own conflict resolution strategy at the application level. By default, DynamoDB uses "the last write wins" strategy [71].

DynamoDB scalability is designed using a variant of consistent hashing in order to partition data and scale incrementally [71]. This variant dynamically partitions data over all the nodes in the clusters, knowing that each node communicates with its immediate neighbours. Some of these nodes are used as coordinators to replicate data on many nodes. DynamoDB optimizes throughput and latency at any scale by using automatic partitioning and Solid State Drive (SSD).

As for querying and manipulating stored data, it is done using two functions: get(*key*) to retrieve all the versions of the object associated with the key "*key*" along with their context,

and put(*key*, *context*, *object*) to determine where to store the replicas of the object "*object*" and to write them to the disk. Data is stored as binary objects, or blobs.

Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

Fig. 21. A list of techniques used by DynamoDB as a response to some encountered problems and their advantages [71]

In DynamoDB, each node shares the routing table with the other nodes in the cluster in order to know what data is stored by which node. In the case of large clusters composed of thousands of nodes, the size of the routing table is significantly large. An improvement is suggested in [71] by using hierarchical extensions.

DynamoDB is Amazon's NoSQL solution for Big Data storage. It has been used by Amazon's services and given good performance, especially regarding availability and data loss. It is well-suited for many Big Data applications, from gaming to the Internet of Things.

J. Azure SQL Database

Azure SQL Database is a highly-available, scalable, relational database built on Microsoft SQL Server and hosted in Microsoft's cloud. It offers the main features of traditional relational databases (tables, views, indexes, procedures, complex queries, full-text search, etc.) as a service in the cloud. It also supports Transact-SQL, ADO.net, and ODBC. Azure SQL Database supports Microsoft SQL Server only, though it is not completely compatible with it. However, a recent version offers a near total compatibility [73].

Azure SQL Database is a TDS [74] proxy endpoint that routes the requests of client applications to the SQL server node that contains the primary replica of data. It has a four-layer architecture, as shown in *Fig.* 22. First, the infrastructure layer, which is Microsoft Azure datacenter, provides powerful computing and storage resources on which the other layers are built. Then there's the platform layer that contains at least three nodes of SQL server running in the infrastructure layer. Then there's the services layer that controls Azure SQL Database in terms of partitioning, billing, and connection routing. Last there's the client layer that contains various tools to allow client applications to connect to Azure SQL Database.

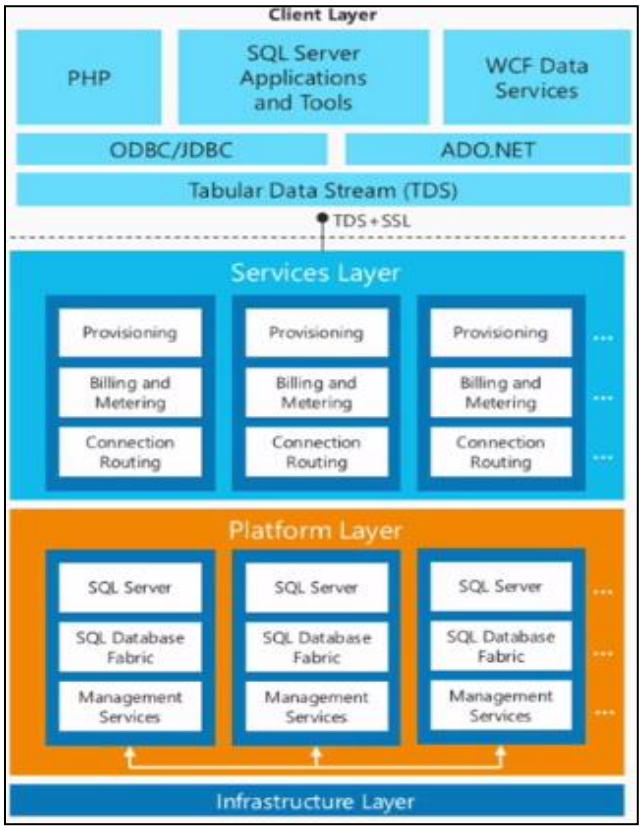

Fig. 22. Microsoft Azure SQL Database architecture [75]

Azure SQL Database organizes data in *table groups*, which are the equivalent of *databases* in SQL Server. A table group can be keyless or keyed. All tables in a keyed table group must have a common column called *partitioning key*. Rows that have the same partitioning key are grouped into *row groups*. However, Azure SQL Database doesn't support executing transactions on more than one table group and, if the table group is keyed, on more than one row group.

Azure SQL Database performs automatic scalability when the table groups are keyed. Each table group is partitioned based on its partitioning key in a way that each row group is contained in one partition. To ensure availability, partitions are replicated using a Paxos-based algorithm, and each partition is stored on a server.

As for consistency, it is ensured by taking snapshots of the table group to verify that committed transactions are reflected in the table group, and uncommitted ones aren't.

Azure SQL Database is used by many companies, including Xerox, Siemens, and Associated Press. However, it suffers from many limitations that render it unsuitable for Big Data applications. For example, the maximum database size supported is 500 GB, and the maximum database number

supported by a server is 150. So for Big Data applications, Microsoft's more adopted solution is DocumentDB.

DocumentDB is a fully-managed, scalable, NoSQL document database offered as a service. It supports SQL querying of JSON stored documents, which are all indexed by default to optimize query performance. Users can also query databases using JavaScript.

DocumentDB supports four levels of consistency, configurable by users. In addition to strong and eventual consistencies, there is session consistency, which is the default mode, and bounded staleness consistency. Session consistency asynchronously propagates writes, and sends read requests to the one replica that contains the requested version. Bounded staleness consistency asynchronously propagates writes, while reads are acknowledged by a majority of nodes, but may be lagged by a certain number of time or operations.

DocumentDB is still at its early stages and lacks many important features, such as backups and replication. Another solution developed by Microsoft and adapted to Big Data is SQL Server in Azure VM, which is not a DBaaS, but an IaaS to run SQL Server databases on virtual machines in the cloud.

K. Amazon RDS

Amazon Relational Database Service (RDS) offers a highly-available access to five distributed relational database management systems (MySQL, Oracle, Microsoft SQL Server, PostgreSQL, and Amazon Aurora) as a service in Amazon's Cloud. RDS aims to make setting up, running, and scaling relational databases simpler and easier, and to automate administrative tasks such as backups, point-in-time recoveries, and patching.

Scalability in RDS is achieved horizontally and vertically. RDS relies on sharding and read replicas to achieve horizontal scalability. As for vertical scalability, users can perform it by using command line tools, APIs, or AWS Management Console.

RDS supports automated backups. These backups can be used as point-in-time recoveries. In addition, users can program backups in the form of snapshots and that can be manually restored afterwards.

RDS replicates data synchronously using the Multi-AZ deployment [76] feature, where data is replicated between a primary instance and a standby instance, as shown in *Fig.* 23. Each one of these instances is stored in a different Availability Zone (AZ) to minimize downtime. If the primary instance fails, RDS performs an automatic failover to the standby instance.

RDS is most adapted to applications that already use one of the five supported database systems, or new applications that work with structured data and need relational features not supported by NoSQL databases, such as join operations [78]. It is also optimized for databases that support heavy I/O workloads. The size of databases stored in RDS can reach up to 3 TB and 30 000 IOPS [79], which makes it suitable for Big Data applications.

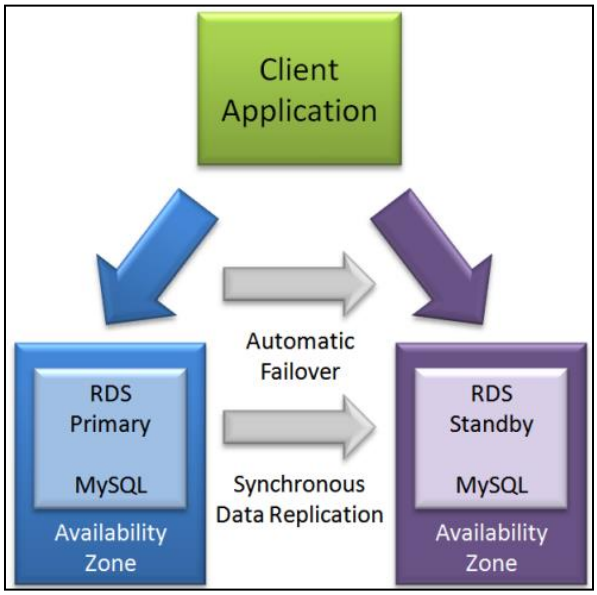

Fig. 23. Example of replication in RDS [77]

L. Other DBaaS solutions

There are various other DBaaS solutions, such as ClearDB, Clustrix, CumuLogic, Heroku, Percona, etc. They are meant for relatively small cloud deployment projects, not Big Data applications. Two prominent DBaaS solutions are HP Cloud Relational Database and Rackspace Cloud Database, two fully-managed, highly-available databases. Both support MySQL, with Rackspace Cloud Database supporting Percona Server, MariaDB also.

HP Cloud Relational Database is provided by HP and hosted in HP Helion Public Cloud. It is still in its early development stages, available in a beta version only for the users of HP Helion Public Cloud. Rackspace Cloud Database is provided by Rackspace. Both databases use OpenStack, an open source cloud computing platform. Users can manage their databases via the native OpenStack command-line interface tools, or APIs. HP Cloud Relational Database supports automated backup/restore operations to enhance fault-tolerance. Both databases offer the possibility for users to initiate backups. Availability is ensured by implementing snapshots and keeping replicas in different availability zones. Both databases are not suitable for Big Data applications, especially regarding data volume, HP Cloud Relational Database having a limiting size of 480 GB per database instance, and Rackspace Cloud SQL supporting a maximum size of 150 GB per database instance.

Rackspace acquired another DBaaS solution, Objectrocket, which is a fully-managed, highly scalable database that supports MongoDB and Redis. It offers the possibility of having instances of multiple TB. Another prominent DBaaS is Salesforce's Database.com, a fully-managed, highly-scalable relational database. It was first used as part of Salesforce's PaaS, force.com, before being available in a stand-alone version.

Database.com uses one large Oracle instance as the main data storage system. It arguably stores data in one wide table composed of hundreds of flex columns, which are columns storing various data types [80]. Salesforce doesn't disclose much of the technical details of Database.com's functionalities and architecture. For example, there are no resources detailing how Database.com handles scalability, replication, or consistency. The maximum supported data size isn't specified either.

We present, in tables IV, V, and VI hereafter, a summary of the databases as a service reviewed in this section.

TABLE IV. COMPARISON BETWEEN THE REVIEWED DATABASES (PART 1)

Name	Provider	Data Model	Supported databases	Data Storage Type	Querying
Cloud Bigtable	Google	Column database	N/A	Tables and Tablets	HBase Shell
Azure SQL Database	Microsoft	Relational	Microsoft SQL Server	Tables	SQL
Cloud Datastore	Google	NoSQL	N/A	Kinds (equivalent of relational tables)	API
HP Cloud Relational Database	HP	Relational	MySQL	Tables	OpenStack CLI API
Cloud SQL	Google	Relational	MySQL	Tables	SQL
Cloudant	IBM	Document datastore	N/A	JSON documents	API
DynamoDB	Amazon	Key-value store	N/A	Key-Value objects	API
MongoLab	MongoDB	Document datastore	MongoDB	BSON documents	MongoDB driver API
Morpheus	Morpheus	Relational or NoSQL	MySQL MongoDB Redis Elasticsearch	Tables JSON documents Key-Value objects	API
Postgres Plus Cloud Database	EnterpriseDB	Relational	PostgreSQL	Tables	API
Rackspace Cloud Database	Rackspace	Relational	MySQL Percona Server MariaDB	Tables	CLI API
RDS	Amazon	Relational	MySQL Oracle Microsoft SQL Server PostgreSQL Amazon Aurora	Tables	SQL
SimpleDB	Amazon	Key-value store	N/A	Key-Value objects	API
Database.com	Salesforce	Relational	N/A	Organizations (equivalent of relational tables)	SOQL SOSL
Objectrocket	Rackspace	Document and Key-value store	MongoDB Redis	Key-Value objects BSON	API

N/A: *NOT APPLICABLE* X: *NOT AVAILABL*

TABLE V. COMPARISON BETWEEN THE REVIEWED DATABASES (PART 2)

Name	Consistency	Replication	Scalability	Deployment Model	Clusters Tenancy	Interoperability with other cloud platforms	Geographical region choice
Cloud Bigtable	Strong Row Consistency	X	Horizontal	Cloud	Multi-tenancy	No	Yes
Azure SQL Database	Strong	Asynchronous	Horizontal	Cloud	Multi-tenancy	No	Yes
Cloud Datastore	Strong Eventual	Asynchronous	Vertical Horizontal	Cloud	Multi-tenancy	No	Yes
HP Cloud Relational Database	X	X	Vertical Horizontal	Cloud	Multi-tenancy	No	Yes
Cloud SQL	X	Synchronous Asynchronous	Vertical	Cloud	Multi-tenancy	No	Yes
Cloudant	Eventual	Synchronous	Horizontal	Cloud On-premise Hybrid	Single-tenancy Multi-tenancy	Rackspace Microsoft Azure SoftLayer	Depending on the provider
DynamoDB	Eventual	Asynchronous	Vertical Horizontal	Cloud	Single-tenancy Multi-tenancy	No	Yes
MongoLab	Eventual	Asynchronous	Horizontal	Cloud	Single-tenancy Multi-tenancy	Amazon Web Services Google Cloud Platform Windows Azure Rackspace Joyent	Depending on the provider

Name	Consistency	Replication	Scalability	Deployment Model	Clusters Tenancy	Interoperability with other cloud platforms	Geographical region choice
Morpheus	X	Asynchronous	Horizontal	Cloud On-premise Hybrid	Single-tenancy Multi-tenancy	Yes (through Morpheus Virtual Appliance)	No
Postgres Plus Cloud Database	Eventual	Asynchronous	Vertical Horizontal	Cloud On-premise	Single-tenancy Multi-tenancy	Amazon Web Services HP Cloud	Yes
Rackspace Cloud Database	Eventual	Asynchronous	Vertical	Cloud On-premise	Single-tenancy Multi-tenancy	No	X
RDS	Strong	Synchronous Asynchronous	Vertical Horizontal	Cloud	Single-tenancy Multi-tenancy	No	Yes
SimpleDB	Strong Eventual	Asynchronous	Vertical Horizontal	Cloud	Single-tenancy Multi-tenancy	No	Yes
Database.com	X	X	X	Cloud	Multi-tenancy	No	X
Objectrocket	Eventual	Asynchronous	Horizontal	Cloud On-premise	Single-tenancy Multi-tenancy	No	Yes

N/A: *NOT APPLICABLE* X: *NOT AVAILABLE*

TABLE VI. COMPARISON BETWEEN THE REVIEWED DATABASES (PART 3)

Name	Client libraries	License	billing	SLA commitment	Initial release date	Big Data compatible	Used by
Cloud Bigtable	HBase Client	Commercial	Per hour	Yes	2015	Yes	Google
Azure SQL Database	PHP ODBC/JDBC .NET SQL Server Tools WCF Data Services TDS	Commercial	Per hour	Yes	2009	Yes	Samsung easyJet
Cloud Datastore	Java JavaScript PHP Python .NET Objective-C	Commercial	Per hour	Yes	2013	Yes	Ubisoft GenieConnect
HP Cloud Relational Database	RESTful API	Commercial	Per hour	No	2012 (beta version)	No	X
Cloud SQL	Java JavaScript PHP Python .NET Objective-C	Commercial	Package Per hour	Yes	2013	No	BeDataDriven CodeFutures
Cloudant	Java Node.js	Commercial	Per month	Yes	2008	Yes	Adobe DHL
DynamoDB	Java PHP .NET Ruby	Commercial	Per month	Yes	2012	Yes	Elsevier Amazon Cloud Drive
MongoLab	Java PHP Python Ruby Note.js	Commercial	Per month	Yes	2011	Yes	Toyota Lyft
Morpheus	Not Available	Commercial	On estimate	No	2014	Yes	Spireon Rowmark
Postgres Plus Cloud Database	Java .Net	Commercial	Per hour Per month One Year subscription	Yes	2012	Yes	Bouygues Telecom Los Angeles Times
Rackspace Cloud Database	Java PHP .Net RESTful API	Commercial	Per hour	Yes	2012	No	InferMed
RDS	Java PHP .Net	Commercial	Per month	Yes	2009	Yes	Airbnb Unilever

Name	Client libraries	License	billing	SLA commitment	Initial release date	Big Data compatible	Used by
SimpleDB	Java PHP Python Ruby .NET	Commercial	Per hour	Yes	2007	Yes	MyMiniLife.com Issuu
Database.com	RESTful API SOAP API	Commercial	On estimate	Yes	2010	X	Cirrus Computing
Objectrocket	Java PHP Python Node.js cURL	Commercial	On estimate	Yes	2012	Yes	SponsorHub SumAll

N/A: *NOT APPLICABLE* X: *NOT AVAILABLE*

V. DISCUSSION

As presented in the previous section, there are various databases offered as a service by many Cloud providers. This model of use, namely DBaaS, offers many advantages both to users and providers. Users find themselves exempt from up-front investments and relieved from the burden of installing, running and administrating their databases. As for providers, the costs of providing their service are optimized, especially in the case of multi-tenancy.

However, there are several points to take into consideration when selecting a DBaaS, few of which we discuss hereafter.

A. Provider's reputation

Within the last decade, Cloud Computing has positioned itself as a primordial technology with an ever growing market, although big IT names still have a dominating position. In the first quarter of 2015 [81], Amazon held 29% of the market share, followed by Microsoft (10%), IBM (7%), Google (5%), Salesforce (4%), and Rackspace (3%). Every one of these providers has a DBaaS solution that benefit from their established Cloud platforms, whether relational (Amazon's RDS and SimpleDB, Microsoft's Azure SQL Database, Google's Cloud SQL, and Rackspace's Cloud SQL) or NoSQL (Amazon's DynamoDB and SimpleDB, IBM's Cloudant, and Google's Cloud Datastore).

In addition to these providers, other ones have positioned themselves quite successfully in the DBaaS market, such as Mongo inc. (MongoLab), Morpheus, and EnterpriseDB (Postgres Plus Cloud Database).

Users may be more confident confiding their data to well-established Cloud "pioneers", or choose to rely on other users' feedback, which every provider has on their website in the form of use cases.

B. Deployment

Users who are looking for a DBaaS should consider the deployment model to know whether their data will be stored on-premise or off-premise. For example, some users would choose to keep their data on-premise, for security concerns. Many providers don't offer the choice, as their databases are hosted in the Cloud only. This is the case for Amazon, Google, Microsoft, MongoDB, and Salesforce. Other providers give the possibility to choose between using their database as a hosted service in their Cloud or on-premise. This is the case of EnterpriseDB, HP, IBM, Morpheus, and Rackspace.

Another point regarding deployment is the interoperability of the DBaaS with other Cloud providers' solutions. In many cases, users' applications are already deployed, whether internally or in the Cloud. Thus, it would be more convenient when a DBaaS provider enables users to select the cloud platform they want to use, even if it is provided by another Cloud provider. This is not the case for providers like Amazon, Google, Microsoft, HP, Rackspace, and Salesforce, who compel customers to use their specific Cloud platforms, as their databases can't be used elsewhere.

Tenancy mode is also a point to consider when selecting a DBaaS. Customers desiring to optimize their database performance may want to opt for single-tenancy, where they get dedicated clusters and don't share resources with other customers. Not all DBaaS have this option. Database.com, for example, was specifically designed to be multitenant. Providers like Microsoft, Google, and HP don't offer this possibility either.

C. Database model

Providers who support many database systems give users the possibility to select a database to use from available databases. This way, customers can choose the database to which they are used or that they are most comfortable with. This can be particularly interesting for users who already have their applications deployed and running, because when a DBaaS offers access to a traditional database (MySQL or PostgreSQL for example), the codes that were designed to work with these databases can work seamlessly in the cloud, exempting users from rewriting their code.

Another point to study before choosing a DBaaS is the data model. Customers must have a clear idea of how they project to use their database, and especially the type of data they deal with. Although developers may benefit from the flexibility of NoSQL databases, due to their being schema free, they will have to explicitly manage data coherence in the application layer (relationships between data, for example, as there are no defined foreign keys in the database). Thus, if data is variably structured and can't be represented using the relational schema, then NoSQL databases will be more adapted. If not, then some relational DBaaS can offer good performance for Big Data applications, like Amazon RDS or Microsoft Azure SQL database.

D. Law and regulations

Data collection and storage are increasingly subject to regulations, whether directly, such as the "Data Protection

Directive" (DPD) [82] in the European Union, or indirectly, such as the "USA Patriot Act" in the United States of America. Such legislation affects the storage of data. The DPD, for example, requires personal data to be stored inside the EU, or only in countries outside the EU that ensure a certain level of data protection.

DBaaS physically store data in various datacenters in different locations. Moreover, to ensure availability, data is replicated across geographically distributed datacenters. Users in some cases may need to choose the geographical location where their data will be stored. This possibility is offered by the majority of the reviewed providers (except Morpheus), who have datacenters mainly in the USA and the EU. Other providers, like Salesforce and Rackspace, don't give details about the location of their datacenters. Another possibility is to opt for keeping data on-premise, which is possible for DBaaS like Cloudant, Postgres Plus Cloud Database, Rackspace Cloud Database, and Objectrocket.

E. Payment mode

One of the main characteristics of Cloud Computing is the concept of pay-as-you-go, where users strictly pay for the resources they consume. DBaaS users pay for the volume of data they store, according to several purchasing options. The majority of providers adopt a billing by the hour plan, where users pay for the volume of data stored during one hour. Examples include Google and Microsoft. Amazon, IBM, and MongoDB enlarge the time period to a month, while other providers like Morpheus, Salesforce, and Rackspace tailor their payment to customers, on a case-by-case basis.

F. Data volume

Choosing a DBaaS for Big Data applications implies to carefully consider the maximum supported size in order to ensure that it can scale to handle terabytes of data. While most reviewed DBaaS verify this condition, HP Cloud Relational Database, Cloud SQL, and Rackspace Cloud Database only offer a maximum instance size of 500 GB. Salesforce doesn't disclose information about Database.com maximum storage size.

G. Data consistency

Consistency, availability, and partition tolerance being complementary (as stated by the CAP theorem), most reviewed DBaaS chose to relax consistency in order to achieve high-availability in distributed environments. This is the case for Cloudant, DynamoDB, MongoLab, Postgres Plus Cloud Database, Rackspace Cloud Database, and Objectrocket. For applications that can't relax consistency, strong consistency is offered by DBaaS like Azure SQL Database, SimpleDB, and Cloud Datastore. The two latter ones implement both strong and eventual consistency, allowing users to choose the most adapted mode.

H. Scalability

Scalability allows adjusting computing resources and storage space to meet the increasing needs of applications. It is one of the inherent characteristics of cloud computing and one of the necessary requirements for Big Data applications.

Most reviewed databases scale horizontally to meet the levels required by Big Data applications. Databases like Cloud Datastore, DynamoDB, Postgres Plus Cloud Database, Amazon RDS and SimpleDB implement both vertical and horizontal scalability. Cloud SQL and Rackspace Cloud Database scale only vertically, which, added to their size limitations, makes them further unsuitable for Big Data applications. As for Salesforce's Database.com, there is no information on how it handles scalability.

I. SLA

A Service-Level Agreement (SLA) is a contractual document that governs the client's use of the provider's services.

SLAs help providers manage the services contracted and maintain the overall level of quality agreed on with their customers. The providers of the reviewed databases use SLAs, except for HP and Morpheus, who don't disclose their SLA policy. They all guarantee high availability, with an uptime of 99.9% at least.

J. Security and Privacy

One of the main concerns that keep organizations and individuals from moving their data to the cloud is the security and privacy aspects. Recent leaks and hacks (iCloud and Sony, to name but a few) only reinforced their reluctance to entrust data to the Cloud [83, 84].

The concern of security and privacy in cloud environments is enhanced by the large volume of datasets managed by Big Data. And just like DBaaS removes the burden of database installation and management, it also ensures the security of data. DBaaS providers implement different levels of security, starting from identity and access management, to data encryption, all through assuring the physical security and monitoring of datacenters. In addition to securing data while being stored in datacenters, it is crucial to ensure its transfer to and from client applications, which can be implemented using cryptographic protocols like TLS or SSL.

Providers like Amazon, Google, Microsoft, IBM, and Rackspace have achieved the ISO/IEC 27001 certification for their cloud platforms.

VI. CONCLUSION

Big Data has emerged as one of the most important technological trends for the current decade. It challenges the traditional approach to computing, especially regarding data storage. Traditional clustered relational database environments prove to be complex to scale and distribute to adapt to Big Data applications and new solutions are continually being developed.

One of the most adapted answers to Big Data storage requirements is Cloud Computing, and more specifically Database as a Service, which allows storing and managing tremendous volume of variable data seamlessly, without need to make large investments in infrastructure, platform, software, and human resources. In this context, our article presents a benchmark of the main database solutions that are offered by providers as DataBase as a Service (DBaaS). We studied the

features of each solution and its adaptability to Big Data applications.

Cloud Computing and Big Data are entwined, with Big Data relying on Cloud Computing's computational and storage resources, and Cloud Computing pushing the limits of these resources. New extensions of Cloud Computing are emerging to further enhance Big Data, especially Fog Computing and Bare-Metal Cloud. Fog Computing uses edge devices and end devices, such as routers, switches, and access points to host services, which minimizes latency. This proximity to end-users, along with its wide geographical distribution and support for mobility makes Fog Computing ideal for Big Data and the Internet of Things applications [85]. As for Bare-Metal Cloud, it aims to optimize performance for applications with high workloads by eliminating the virtualization layer and delivering "bare" servers without hypervisors installed. This way, there won't be too many virtual machines competing for physical resources and impeding the overall performance.

REFERENCES

[1] J. Gantz and D. Reinsel, "IDC: The Digital Universe in 2020: Big Data. Bigger Digital Shadows, and Biggest Growth in the Far East", 2012

[2] S. Radicati and Q. Hoang, "Email statistics report, 2012-2016", The Radicati Group, Inc., London, 2012

[3] "2015 State of the Cloud Report", RightScale, Inc., Retrieved from http://www.rightscale.com/lp/2015-state-of-the-cloud-report

[4] L. Kleinrock, "A vision for the Internet", ST Journal of Research, Volume 2, Issue 1, 2005

[5] J. McCarthy, MIT Centennial Speech of 1961 cited in "Architects of the Information Society: Thirty-five Years of the Laboratory for Computer Science at MIT", SL Garfinkel Ed, 1999

[6] D. Parkhill, "The Challenge of the Computer Utility", Addison-Wesley Publishing Company, 1966

[7] A. Idrissi and M. Abourezq, "Skyline in Cloud Computing", Journal of Theoretical and Applied Information Technology, Vol. 60, No. 3, February 2014

[8] M. Abourezq and A. Idrissi, "Introduction of an outranking method in the Cloud computing research and Selection System based on the Skyline", Proceedings of the International Conference on Research Challenges in Information Science (RCIS), May 2014

[9] P. Mell and T. Grance, "The NIST definition of cloud computing", National Institute of Standards and Technology, Issue 6, 2009

[10] S. Radack, "Cloud Computing: A Review of Features, Benefits, and Risks, and Recommendations for Secure, Efficient Implementations", NIST, ITL Bulletin, June 2012

[11] Cisco Global Cloud Networking Survey, 2012, Retrieved from http://www.cisco.com/c/en/us/solutions/enterprise-networks/global_cloud_survey.html

[12] YouTube statistics, Retrieved from http://www.youtube.com/yt/press/statistics.html

[13] P. Vagata and K. Wilfong, "Scaling the Facebook data warehouse to 300 PB", April 10, 2014, Retrieved from https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb

[14] L. Tay, "Inside eBay's 90PB data warehouse", May 10, 2013, http://www.itnews.com.au/News/342615,inside-ebay8217s-90pb-data-warehouse.aspx

[15] J. Lin and D. Ryaboy, "Scaling big data mining infrastructure: the twitter experience", ACM SIGKDD Explorations Newsletter, Volume 14, Issue 2, 2013

[16] M. Cox and D. Ellsworth, "Managing big data for scientific visualization", ACM Siggraph, Volume 97, 1997

[17] P. Zikopoulos and C. Eaton, "Understanding big data: Analytics for enterprise class hadoop and streaming data", McGraw-Hill Osborne Media, 2011

[18] C. Min, S. Mao, Y. Zhang, and V. Leung, "Big data: related technologies, challenges and future prospects", Springer, 2014

[19] D. Boyd and K. Crawford, "Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon", Information, communication & society, Volume 15, Issue 5, 2012

[20] I. Abaker, T. Hashem, I. Yaqoob, N. Badrul Anuar, S. Mokhtar, A. Gani, and S. Ullah Khan, "The rise of "big data" on cloud computing: Review and open research issues", Information Systems, Volume 47, January 2015

[21] A. Cuzzocrea, I. Song, and K. C. Davis, "Analytics over large-scale multidimensional data: the big data revolution", In Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP, pp. 101-104. ACM, 2011

[22] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers, and McKinsey Global Institute, "Big data: The next frontier for innovation, competition, and productivity", 2011

[23] H. Landrock, O. Schonschek, and A. Gadatsch, "Big Data Vendor Benchmark 2015 - A Comparison of Big Data Solution Providers", Experton Group AG, 2015

[24] S. Connolly, "7 Key Drivers for the Big Data Market", May 14, 2012, Retrieved from http://hortonworks.com/blog/7-key-drivers-for-the-big-data-market

[25] Big Data definition in the Gartner IT Glossary, Retrieved from http://www.gartner.com/it-glossary/big-data

[26] "What is big data?", Retrieved from http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[27] Geoinformatics, Department of Civil Engineering, IIT Kanpur, Retrieved from http://gi.iitk.ac.in/gi/geoinformatics

[28] A. Banafa, "The Future of Big Data and Analytics", School of Business and Information Technology, March 2014, Retrieved from http://www.kaplanuniversity.edu/information-technology/articles/future-of-big-data-analytics.aspx

[29] S. Abiteboul, "Querying semi-structured data", Springer Berlin Heidelberg, 1997

[30] H. U. Buhl, M. Röglinger, F. Moser, and J. Heidemann, "Big Data: A Fashionable Topic with(out) Sustainable Relevance for Research and Practice?", Business & Information Systems Engineering, Volume 5, Issue 2, 2013

[31] M. Walker, "Data Veracity", Data Science Central, November 28, 2012, Retrieved from http://www.datasciencecentral.com/profiles/blogs/data-veracity

[32] S. B. Siewert, "Big data in the cloud: Data velocity, volume, variety, veracity", IBM, July 9, 2013

[33] J. Gantz and D. Reinsel, "Extracting value from chaos", IDC iview 1142, 2011

[34] "Draft NIST Big Data Interoperability Framework: Volume 1, Definitions", NIST Special Publication 1500-1, April 6, 2015

[35] K. Fanning and E. Drogt, "Big Data: New Opportunities for M&A", Journal of Corporate Accounting & Finance, Volume 25, Issue 2, 2014

[36] D. Boyd and K. Crawford, "Critical questions for big data", Information, Communication & Society, Volume 15, Issue 5, 2012

[37] A. McAfee and E. Brynjolfsson, "Big data: the management revolution", Harvard Business Review, Volume 90, October 2012

[38] S. Madden, "From databases to big data", IEEE Internet Computing, Volume 16, Issue 3, 2012

[39] "HGST Unveils Intelligent, Dynamic Storage Solutions To Transform The Data Center", Retrieved from http://www.hgst.com/press-room/press-releases/HGST-unveils-intelligent-dynamic-storage-solutions-to-transform-the-data-center

[40] "Twitter Usage Statistics", Retrieved from http://www.internetlivestats.com/twitter-statistics/

[41] R. Krikorian, "New Tweets per second record, and how", August 16, 2013, Retrieved from https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

[42] G. Paterno, "NoSQL Tutorial: A comprehensive look at the NoSQL database", Linux Journal, Volume23, Issue 67, 1999

[43] F. Chang et al., "Bigtable: A distributed storage system for structured data", In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, 2006

[44] CL. Chen and C. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data", Information Sciences, Volume 275, 2014

[45] E. A. Brewer, "Towards robust distributed systems", In ACM Symposium on Principles of Distributed Computing, Volume 7, 2000

[46] D. Pritchett, "Base: An ACID alternative", ACM Queue, Volume 6, Issue 3, 2008

[47] R. Cattell, "Scalable SQL and NoSQL data stores", ACM SIGMOD, Volume 39, Issue 4, 2011

[48] A. Moniruzzaman, "NewSQL: Towards Next-Generation Scalable RDBMS for Online Transaction Processing (OLTP) for Big Data Management", arXiv preprint arXiv:1411.7343, 2014

[49] S. Ghemawat, H. Gobioff, and ST. Leung, "The Google file system", In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03), 2003

[50] D. Borthakur, "HDFS architecture guide", Hadoop Apache Project, 2008

[51] B. Antony, "HDFS Storage Efficiency Using Tiered Storage", January 12, 2015, Retrieved from http://www.ebaytechblog.com/2015/01/12/hdfs-storage-efficiency-using-tiered-storage/

[52] "NoSQL, NewSQL and Beyond: The drivers and use cases for database alternatives", April 15, 2011, 451 Research, Retrieved from https://451research.com/report-long?icid=1651

[53] D. Agrawal, A. El Abbadi, F. Emekci, and A. Metwally, "Database management as a service: Challenges and opportunities", In IEEE 25th International Conference on Data Engineering, 2009

[54] W. Lehner and KU. Sattler "Database as a service (DBaaS)", In IEEE 26th International Conference on Data Engineering, 2010

[55] "Google Launches Bigtable, A Big Managed Database In The Cloud", Forbes, May 6, 2015, Retrieved from http://www.forbes.com/sites/paulmiller/2015/05/06/google-launches-bigtable-a-big-managed-database-in-the-cloud

[56] M. Burrows, "The Chubby lock service for loosely-coupled distributed systems", In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, 2006

[57] J. Baker et al., "Megastore: Providing Scalable, Highly Available Storage for Interactive Services", In Conference on Innovative Data Systems Research, Volume 11, 2011

[58] J. Corbett et al., "Spanner: Google's globally distributed database", ACM Transactions on Computer Systems, Volume 31, Issue 3, 2013

[59] L. Lamport, "Paxos made simple", ACM Sigact News, Volume 32, Issue 4, 2001

[60] B. Lampson and D. Lomet, "Distributed transaction processing using two-phase commit protocol with presumed-commit without log force", U.S. Patent 5,335,343, issued August 2, 1994

[61] "SLA for availability", February 11, 2014, Retrieved from http://googlecloudplatform.blogspot.com.es/2014/02/google-cloud-sql-now-generally-available.html

[62] IBM white paper, "Technical Overview: Anatomy of IBM Cloudant DBaaS", Retrieved from http://www-01.ibm.com/software/data/cloudant/

[63] IBM software White Paper, "Build more and grow more with Cloudant DBaaS", Retrieved from https://cloudant.com/resources/white-papers/build-more-and-grow-more-with-cloudant-dbaas/

[64] P. Membrey, E. Plugge, and D. Hawkins, "The definitive guide to MongoDB: the NoSQL database for cloud and desktop computing", Apress, 2010

[65] J. Han et al., "Survey on NoSQL database", 6th IEEE international conference on Pervasive Computing and Applications, 2011

[66] "Award Winning Heterogeneous Database Provisioning & Management Platform for Private, Public, & Hybrid Clouds", Retrieved from http://www.gomorpheus.com/morpheus_appliance_datasheet.pdf

[67] "Overview of Postgres Plus Cloud Database", Retrieved from http://www.enterprisedb.com/Cloud

[68] J. Sullivan, "EnterpriseDB's Postgres Plus Cloud DB Auto-Scales In AWS", January 29, 2014, Retrieved from http://www.tomsitpro.com/articles/enterprisedb-postgresql-cloud-database-aws,1-1617.html

[69] "Amazon's SimpleDB Developer Guide", Retrieved from http://aws.amazon.com/documentation/simpledb/

[70] S. Sakr and M. Gaber, "Large Scale and Big Data: Processing and Management", CRC Press, 2014

[71] G. DeCandia et al., "Dynamo: Amazon's highly available key-value store", In ACM SIGOPS Operating Systems Review, Volume 41, Issue 6, ACM, 2007

[72] L. Lamport, "Time, clocks, and the ordering of events in a distributed system", Communications of the ACM 21, Issue 7, 1978

[73] G. Milener, "What's new in SQL Database V12", May 15, 2015, Retrieved from http://azure.microsoft.com/en-us/documentation/articles/sql-database-v12-whats-new/

[74] "Tabular Data Stream Protocol", Retrieved from https://msdn.microsoft.com/en-us/library/dd304523.aspx

[75] "Windows Azure SQL Database: SQL Database Fundamentals", Retrieved from http://channel9.msdn.com/Series/Windows-Azure-SQL-Database

[76] "Amazon RDS Multi-AZ Deployments", Retrieved from http://aws.amazon.com/rds/details/multi-az/

[77] J. Barr, "Amazon RDS – Multi-AZ Deployments For Enhanced Availability & Reliability", May 17, 2010, Retrieved from https://aws.amazon.com/blogs/aws/amazon-rds-multi-az-deployment/

[78] J. Baron and S. Kotecha, "Storage Options in the AWS Cloud", October 2013, Retrieved from http://aws.amazon.com/whitepapers/

[79] "Amazon RDS now supports 3TB and 30,000 Provisioned IOPS per database instance", March 13, 2013, Retrieved from http://aws.amazon.com/about-aws/whats-new/2013/03/13/amazon-rds-3tb-30k-iops/

[80] "The database architecture of salesforce.com, force.com, and database.com", September 15, 2011, retrieved from http://www.dbms2.com/2011/09/15/database-architecture-salesforce-com-force-com-and-database/

[81] "AWS Still Bigger than its Four Main Competitors Combined Despite Surging Growth", Synergy Research Group, April 27, 2015, Retrieved from https://www.srgresearch.com/articles/aws-still-bigger-its-four-main-competitors-combined-despite-surging-growth

[82] « Protection of personal data », European Commission, Retrieved from http://ec.europa.eu/justice/data-protection/

[83] G. Furukawa, "Sony's Two Big Mistakes: No Encryption, and No Backup", January 26, 2015, Retrieved from http://java.dzone.com/articles/sonys-two-big-mistakes-no

[84] « Apple denies iCloud, Find My iPhone security breach: Only 'very targeted attacks' », Tech Times, September 7, 2014, Retrieved from http://www.techtimes.com/articles/14717/20140907/apple-denies-icloud-find-my-iphone-security-breach-only-very-targeted-attacks.htm

[85] "Fog Computing, Ecosystem, Architecture and Applications", Research at CISCO, Retrieved from http://www.cisco.com/web/about/ac50/ac207/crc_new/university/RFP/rfp13078.html

[86] J. Corbett et al., "Spanner: Google's globally distributed database", ACM Transactions on Computer Systems (TOCS), Volume 31, Issue 3, 2013