Open Access

Open Journal of Big Data (OJBD)
Volume 1, Issue 2, 2015

www.ronpub.com/ojbd
ISSN 2365-029X

Big Data in the Cloud: A Survey

Pedro Caldeira Neves [A,B], Jorge Bernardino [A,C]

[A] Polytechnic of Coimbra, Rua Pedro Nunes - Quinta da Nora, Coimbra, Portugal, pedrofilipeneves@gmail.com

[B] Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, USA

[C] CISUC - Centre for Informatics and Systems of the University of Coimbra, DEI – Pólo II, Coimbra, Portugal, jorge@isec.pt

ABSTRACT

Big Data has become a hot topic across several business areas requiring the storage and processing of huge volumes of data. Cloud computing leverages Big Data by providing high storage and processing capabilities and enables corporations to consume resources in a pay-as-you-go model making clouds the optimal environment for storing and processing huge quantities of data. By using virtualized resources, Cloud can scale very easily, be highly available and provide massive storage capacity and processing power. This paper surveys existing databases models to store and process Big Data within a Cloud environment. Particularly, we detail the following traditional NoSQL databases: BigTable, Cassandra, DynamoDB, HBase, Hypertable, and MongoDB. The MapReduce framework and its developments Apache Spark, HaLoop, Twister, and other alternatives such as Apache Giraph, GraphLab, Pregel and MapD – a novel platform that uses GPU processing to accelerate Big Data processing – are also analyzed. Finally, we present two case studies that demonstrate the successful use of Big Data within Cloud environments and the challenges that must be addressed in the future.

TYPE OF PAPER AND KEYWORDS

Research review: *Big Data, Cloud Computing, NoSQL, MapReduce, Graph based DBMS*

1 INTRODUCTION

Society is becoming deeply immersed in the use of electronic devices that generate Petabytes of data, a gold mine for knowledge extraction, with different *volume*, *velocity*, and *variety*. *Value* and *veracity* are also two important properties that specify the need of valuable and truthfulness data. These five properties are known as the 5 V´s model that supports the Big Data concept [10].

Big Data became a hot topic among computer researchers and business areas [60], providing organizations with a powerful tool to analyze large structured and unstructured data and make useful decisions through it. Knowledge extraction frequently requires sophisticated analytic solutions that mine structured and unstructured data helping organizations gaining insights over the information within their private and public data.

Cloud computing is currently one of the most discussed and promising topic in the information technology field and it was listed in Gartner's top ten technologies list for the last four consecutive years [9]. Cloud computing became a trend for researchers and organizations [21], allowing virtualizing resources and offering theoretically unlimited processing power and storage. In practice, cloud can easily scale up with two types of scaling: *vertical scaling*, which offers the possibility to upgrade servers; and *horizontal scaling* that allows adding new servers to a cluster [9].

Organizations that require a dynamic information technology infrastructure are moving to cloud due its scalability and effective pricing models. Cloud features allow startups (and others), that usually do not possess a large budget for IT investment, to hire computer resources in a *pay-as-you-go* model. From the Big Data perspective, cloud is an interesting environment, since it virtualizes distributed resources provide large storage capacities and high processing power (therefore, it can host and process big volumes of data).

In this paper we present Big Data in the cloud. Particularly we describe NoSQL databases; MapReduce and other variants such as: HaLoop, Twister and Apache Spark that address MapReduce's lack of interactivity; Pregel, GraphLab and Apache Giraph, that were designed to work with graphs, instead of data files; and MapD, which is a novel approach to accelerate Big Data processing by means of GPU processing power. As these systems run over distributed architectures they perfectly fit in the cloud paradigm and are often offered as Big Data as a service or used by several online applications. The paper presents two case studies showing how Big Data and Cloud computing match, the issues on moving big loads of data to the cloud and the existent solutions, disaster recovery plans, and existing Big Data challenges.

The remainder of this paper is structured as follows. Section 2 introduces Big Data and Cloud Computing. Section 3 presents NoSQL, MapReduce, Graph databases and GPU processing. Section 4 discusses the case studies and existent problems and solutions when moving Big Data to the cloud. Section 5 discusses disaster recovery, and Section 6 presents Big Data challenges, new trends and future directions. Finally, Section 7 concludes the paper.

2 BIG DATA AND CLOUD COMPUTING

The Big Data concept has been strongly leveraged and became a major force of innovation across academics, governments and corporates. The paradigm is regarded as an effort to understand and get information from data (Big Data Analytics), providing insights and information over huge datasets. Therefore, it is seen by governments as a way to improve cities (smart cities [56], [11]) and get proper insights over their people. Corporates regard this technology as a way to better know and understand their clients, to get closer to them and gain competitive advantage over their competitors. At last, Big Data is viewed by scientists as a mean to store and process huge amounts of data such as those yielded by CERN's Large Hadron Collider (LHC) in Switzerland [4].

Table 1. Examples of Big Data sizes [58]

Data Set/Domain	Description
Social Media	12+ Terabytes (10^{12}) of tweets every day and growing.
British Library UK Website Crawler	~110 Terabytes (10^{12}) per domain crawl to be delivered
LHC - Large Hadron Collider (CERN)	13-15 Petabytes (10^{15}) in 2010
Internet Communications (Cisco)	667 Exabytes (10^{18}) in 2013
Digital Universe	7.9 Zettabytes (10^{21}) (2015)

Big Data not only concerns the ability to storage huge amounts of data but also ways to process and extract knowledge from it [35]. Table 1 presents some examples of Big Data sizes in different domains. In practice, a big data database can contain structured and unstructured data that may come at different velocities, be varied and have different volumes. These are known by the three "V's" of big data [60]. Two other "V's" – veracity and value – are also important to explain that quantity is good but valuable and trustful data are also important (see Figure 1). The following paragraphs briefly describe the 5 V's model:

Volume concerns the huge loads that typically Big Data has to deal with. Processing and storing big volumes of data is rather difficult, since it concerns (among others): scalability (vertical, horizontal or both) in order to facilitate the storage and processing power growth; availability, which guarantees access to data and ways to perform operations over them; and bandwidth and performance, that guarantee the access to data at the right-time.

Variety concerns the different types of data from various sources that Big Data frameworks have to deal with (typically, different sources output different kinds of data). Big Data is a way to overcome these differences and unify data. Internet of Things (IoT) [43] is a Big Data related topic that studies data from individual objects of everyday life that can be very varied: Internet traffic, smartphones, wearable technology, and others. In order to process various types of data, Big Data must provide data-type abstraction frameworks.

Velocity concerns the different rates from each data source. For instance, an Enterprise Data Warehouse

(EDW) is typically updated once every day, whilst information from wireless sensor systems is constantly being updated. In order to aggregate data from several data sources, Big Data must be able to deal which data arriving at different velocities.

Value concerns the true value of data (i.e., the potential value of the data regarding the information they contain). Huge amounts of data are useless if they do not provide value for who is exploring it.

Veracity refers to the trustfulness of the data (i.e., it addresses the confidentiality, integrity, and availability of the data). Data are meaningless if their source is unreliable. Therefore, organizations need to ensure that the data is correct as well as the analyses performed on the data are correct.

The five V's of Big Data complement each other in order to provide solutions that are able to store and process data more efficiently.

Cloud computing is another modern movement that offers theoretically unlimited on-demand services to its users. Cloud's ability to virtualize resources allows abstracting from hardware, requiring little interaction with cloud providers and smoothly enabling users to access terabytes of storage, high processing power and high availability as a *pay-as-you-go* model [36]. Moreover cloud computing transfers all costs and responsibilities from the user to the cloud provider, leveraging companies in their early days.

Normally, it is a great endeavor for a startup company to start its business within IT market because they typically do not have the resources to buy their own data servers or machines. In addition to the hardware expenses, a company must consider several other costs such as software licenses, hardware, IT personnel and the maintenance of the infrastructure. Cloud computing provides an easy way to get resources on a pay-as-you-go basis, offering scalability and availability, which means that companies can easily negotiate resources with the cloud provider in order to operate their business.

Cloud providers usually offer three different basic services: Infrastructure as a Service (IaaS); Platform as a Service (PaaS); and Software as a Service (SaaS):

IaaS delivers storage, processing power, virtual machines, and so on. From the point of view of cost reduction, it makes sense to hire computer power as virtual machines. All that is needed is a couple of low-cost computers to serve as front-end to access the virtual machines stored in the cloud. The cloud provider satisfies the needs of the client by virtualizing resources according to the service level agreements (SLAs). Some examples of IaaS are Amazon EC2[1] and Google Compute Engine[2].

PaaS is built on top of IaaS. The service allows the user to deploy cloud applications created using the programming and runtime environments supported by the provider. Once more, by contracting this service, one is released from server maintenance and software updates, transferring those concerns to the cloud provider. Examples of PaaS are Google App Engine[3] and Microsoft Azure[4].

SaaS is one of the most known cloud models. It consists of applications running directly in the cloud provider. Some of the most used SaaS applications are Google Docs and Dropbox.

As shown in Figure 2, these three basic services are closely related: SaaS is developed over PaaS and ultimately PaaS is built on top of IaaS. Also, from these basic services several others emerged, including Database as a Service (DBaaS) [61] and BigData as a Service (BDaaS) [27]. DBaaS (DataBase as a Service), as well as BDaaS (Big Data as a Service), usually consist in a SaaS that allows users to hire database services. AaaS [17] (Analytics as a Service) is another service that allows users to hire analytics tools to perform calculations over data.

Since cloud virtualizes resources that are often distributed in clusters or datacenters, it is the most suitable framework for Big Data processing. By virtualizing thousands of machines we can create the high processing power and high storage levels to store and process big amounts of data.

Figure 1: 5 V's model of Big Data

[1] http://aws.amazon.com/ec2/
[2] https://cloud.google.com/compute/
[3] https://cloud.google.com/appengine/
[4] https://azure.microsoft.com/en-us/

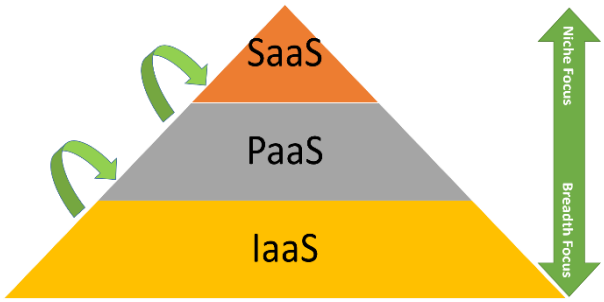

Figure 2: Relationship of cloud basic services

3 BIG DATA DBMS

Big Data and traditional entity-relation DBMS (Relational DBMS) are two incompatible concepts [10]. Firstly because the amount of data is too big to be managed by an entity-relational model; and secondly because traditional models tend to not work well (or be very expensive) on distributed systems, thus, availability and scalability are compromised. Commercial DBMSs tend to work better than open source but when processing big loads of data, they must run under complex hardware, which becomes expensive when scaling to clusters of machines.

Regarding scalability, Relational DBMS features make them less flexible – especially concerning scaling out. As such, several projects like MySQLCluster [8], VoltDB [67] and others were designed in order to provide scalability while still using basic MySQL properties. By using a sharded, (which is a mechanism that splits large datasets into smaller ones) and shared nothing architecture (a system where each node is independent) these projects accomplished to successfully scale-out SQL, separating tables over various servers. Nevertheless, as they use traditional MySQL, there are still some limitations:

- Use of small-scope operations: operations that spans many nodes – such as joins – do not scale well with sharding.
- Use of small-scope transactions: transactions that span many nodes are very inefficient.

Not Only SQL (NoSQL) DBMS bypass these problems by avoiding performing large operations, large transactions and join operations. NoSQL DBMS were developed as highly scalable databases that allow easy data distribution over a number of servers. Next, we provide an overview of NoSQL and Hadoop/MapReduce systems and several interactive alternatives. Additionally, we specify the characteristics of graph databases and MapD, which uses GPU processing for accelerate Big Data processing.

3.1 NoSQL

NoSQL (Not Only SQL) technology does not rely on entity-relation models; instead, information is stored as 'key-value' pairs, documents, columns or graphs, which supports an easy scale out process (horizontal scaling – the process of adding new machines to a cluster). These systems were developed to run over distributed and fault-tolerant architectures in which data resides in several redundant servers so that the system can be easily scalable. However, NoSQL systems offer little more than an efficient way to store and replicate data, providing only retrieval and appending operations. In the following paragraphs we overview the characteristics of some of the most used NoSQL systems: BigTable [28], DynamoDB [31], and the open source Cassandra [1], HBase [43], Hypertable [32], and MongoDB [47].

BigTable: was developed by Google in 2004 and is now used in more than 60 Google applications such as Google Earth, Web Indexing, Google Financing, Google Analytics and Personalized search. The system, which was developed to be a distributed, high efficient, proprietary system to manage structured data, was built upon the Google file system. It organizes tables as different groups of columns with variable dimensions and consisting in a sparse, distributed, persistent, multidimensional sorted map [28] in which the map is indexed by a row, a column key and a timestamp and each value in the map is an uninterpreted array of bytes.

Data in Big Table is organized in tablets that are assigned to a root table that contains the location of all tablets. When a write operation arrives at a tablet server, the server checks if it is well-formed and if the user has authorization to perform writing operations. If that is the case, the operation is written to a commit log. After the write has been committed, its contents are written to the memTable – Table stored in memory. When a read operation arrives to the tablet server, it is also checked if it is well-formed and if the user has proper authorization to get that data. Only then, the operation is executed.

Concerning the API, the system allows client applications to create and delete tables and column families, changing clusters (exploring locality), tables, and column family metadata [28].

Cassandra [25], [53] is one of the most known engines for NoSQL. The system provides "*automatic data distribution across all nodes that participate in a "ring" or database cluster*" [24]. Cassandra [54] supplies high performance at massive scale and high availability which, instead of using a legacy master-slave or a manual and difficult-to-maintain shared

design, uses a peer-to-peer (P2P) distributed architecture that is much more elegant, easy to set up and maintain and has no single point-of-failure.

Cassandra's built-for-scale architecture allows it to handle petabytes of information and thousands of concurrent users/operations per second (across multiple data centers) as easily as it can manage much smaller amounts of data and user traffic [24]. Similarly to BigTable, Cassandra implements a hierarchical architecture based on columns (name, value and timestamp) grouped by families that map the columns in each line. Cassandra's first dimension is a keyspace that contains the column families.

When a read or write operation arrives, the system identifies the nodes that own the specified key and route the requests to those nodes, waiting for a reply. If the reply does not come within a configured timeout, the request is considered to have failed. Updates are cached in memory and then written back to the disk.

DynamoDB [31], [30] was developed by Amazon to support its applications. The system was built to meet high reliability and availability requirements, applying techniques like data partitioning and replication using consistent hashing. DynamoDB provides consistency through object versioning: during updates, the consistency is maintained by a "quorum-like technique and a decentralized replica synchronization protocol" [31]. Query operations consist in read/write operations to data items that are uniquely identified by a key. States are stored as binary objects (blobs) identified by unique keys. The system allows distributing the load across multiple storage hosts and the possibility for the administrator to configure how many copies of the data items are created for fault tolerance purposes.

DynamoDB can execute two operations: *get()* and put(). A node that handles a *read()* or a *put()* is named coordinator and is typically the first node of a ranked preference list. Read and write operations involve the first healthy nodes of the preference list, skipping over those that are down or inaccessible. Upon receiving a *put()* request for a key, the coordinator generates a vector clock and stores it locally. The coordinator then sends it to the N highest-ranked reachable nodes. When receiving a *read()* request, the coordinator requests all existing versions of data for that key from the N highest-ranked reachable nodes in the list for that key. If the coordinator receives multiple versions of the same data, it returns all the unrelated versions. DynamoDB provides fault detection and allows nodes to be added and removed without any manual partitioning or redistribution.

HBase [43], [13] is an open-source, scalable, fault-tolerant, widely used system, built on top of Apache Hadoop and modeled around BigTable. HBase is used in many different systems and by many enterprises and entities such as Facebook, Twitter, and Mendeley. This system is a sparse, multidimensional, sorted map in which each cell is uniquely identified by row id, column id and timestamp set. Its architecture relies on Hadoop (HDFS) to store files and MapReduce to single-row or multi-row operations. Processing data requires mapping data into Hadoop nodes, shuffle data and reduce the outputs (MapReduce is detailed in section 3.2).

HBase is scalable and fault-tolerant and performs atomic row operations with row level-locking, featuring compression and in-memory operations. Partitioning and distribution methods are transparent as there is a multiple master support, to avoid a single point-of-failure.

Hypertable [10], [39] is an open source scalable database modeled from BigTable as a distributed system that allows parallelization and represents data as a multidimensional table. Hypertable is designed to use a highly available and scalable file system such as Hadoop (although it can run on top of any file system). This system follows a master slave architecture where the master does not handle data; instead, it coordinates slave nodes (range servers), which handle write/read operations.

When a write operation arrives, Hypertable inserts it in the Commit Log, before changing the in-memory CellCache. The system then sends an acknowledgment to the application that requested the write operation. When a read request arrives, Hypertable routes the request to the proper range server in order to get the data. This system provides a low-level API and has its own query language (Hypertable Query Language – HQL) that allows creating, modifying and querying the multidimensional table.

MongoDB [47], [59] is an open source, scalable cross-platform, document oriented database engine that aims at providing high performance, availability and easy scalability. A MongoDB deployment hosts a number of databases that consist in sets of collections that hold a set of documents. A document is a set of key-value pairs with a dynamic schema, which means that documents in the same collection do not need to have the same set of fields or structure. A document may hold very different and complex types of data, storing complex data types and large binary data files like videos and images.

A MongoDB cluster is made of one or two nodes and read/write operations are routed to the appropriate nodes. Concerning scalability, auto sharding [51] allows to scale clusters linearly by adding more machines, being possible to increase capacity without any downtime. MongoDB's replication is mostly used

for failover and not for scalability. The system supports master-slave replication with automatic failover and recovery, and replication (and recovery) is done at the level of shards.

These six DBMS share the goal of addressing RDBMS inefficiencies, although they present several differences. In a nutshell, they differ on the type of data stored. While BigTable, DynamoDB and HBase can only store structured data, Cassandra and MongoDB can contain structured, semi-structured and unstructured data and Hypertable, structured and unstructured data. Also, they address data differently – Cassandra, BigTable and Hypertable are column oriented DBMS whereas DynamoDB and HBase use key-value pairs and MongoDB is JSON document oriented.

Concerning schema, BigTable, Hypertable and HBase are schema-oriented, while DynamoDB is schema-free. Cassandra is schema-optional, which means that it may, or may not use schema, while MongoDB provides document dynamic schemas. In MongoDB users can easily add or remove fields from a document. Regarding the architecture, all systems follow a master-slave approach with the exception of Cassandra, which follows a P2P architecture.

Concerning the API, BigTable provides functions to change cluster, HBase and Cassandra provide in-memory capabilities and Hypertable has Hadoop compatibility. Table 2 presents a summary of each the features of each system. “Main type” stands for the type of architecture used. Here, the three types of data models are Column family, Document and Key-value. The types of data supported refer to the data that each DBMS can handle. Schema refers to the possibility of a DBMS to be schema-less or schemafull. Being a schemafull DBMS means that the DB must obey to a predefined schema to be able to manage data whereas schema-less means the opposite. Dynamic schema means that the schema can change very easily. Architecture stands for the type of architecture (models of communication) used in the design of the DBMS and consistency ensures that any transaction will bring the database from one valid state to another.

3.2 MapReduce

MapReduce (best known by its open source implementation: Hadoop) has become a *de facto* standard used by several commercial and non-commercial solutions in the context of Big Data. The system consists in an abstraction layer that handles hardware complexity and creates an interface between the programmer and the data management.

MapReduce was developed as a shared-memory system following a master/slave architecture and uses HDFS [40], which creates and replicates clusters (fault-tolerance), splitting and replicating datasets to nodes where they are more likely to be consumed by mappers. Exploring locality is a feature that improves MapReduce performance [21] (i.e., the performance is better if the distance between tasks and needed data is smaller).

The abstraction provided by this framework requires programming only two functions: “Map”, which is used for per-record computation; and “Reduce” that aggregates the output from the “Map” functions and gathers the final results. Typically, the programmer specifies both functions within a single job. The job then automatically divides the input dataset into several independent subsets that are then processed by the “Map” tasks in slave nodes. At last, MapReduce sorts (shuffle) the “Map” results, aggregating them through the “Reduce” function (job performed by a Master node).

The MapReduce framework has been scored as one of the best frameworks regarding Big Data. This is due to several features, namely: fault-tolerance, parallelism, locality exploration, throughput, and abstraction. However, the purpose of storing such amounts of data is to analyze and retrieve meaningful information from it. In this topic, MapReduce lacks essential features [66]: first, its performance is better than the traditional DMBSs only when it concerns high volumes of data and second, custom code has to be written even for the most common operations.

Many programmers are unfamiliar with the MapReduce framework. Thus, they rather prefer to use SQL or a similar language as a high-level declarative language to program tasks, leaving details to the backend engine. In this direction, several languages were developed to simplify the task of programming for MapReduce:

Hive [15] is a platform that uses an SQL-similar language (declarative) – HiveQL – to query MapReduce, translating queries into acyclic graphs of MapReduce jobs and submitting them to Hadoop for execution. It becomes a user-friendly SQL-like easy to use tool that serves as intermediate between MapReduce and the user, skipping the difficulties of programming directly MapReduce tasks. Hive was developed by Facebook and adds some other features to traditional SQL queries. Particularly, it introduces subqueries in the “from” clause, various types of “joins”, “group-by”, aggregations and “create table as select”.

Jaql [42] is a general-purpose dataflow language designed by IBM upon the JSON data model. It manipulates semi-structured data in the form of abstract JSON values, providing SQL-like operators as

Table 2. Features of NoSQL systems

Database system	*BigTable*	*Cassandra*	*DynamoDB*	*Hbase*	*Hypertable*	*MongoDB*
Main type	Column	Column	Key-value	Key-value	Column	Document
Types of data supported	Structured	structured, semi-structured, and unstructured data	Structured	Structured	Structured and unstructured data	structured, semi-structured, and unstructured data
Schema	Uses schema	Schema-optional	Schema-free	Uses schema	Uses schema	Dynamic schema
Architecture	Master-slave	P2P	Master-Slave	Master-Slave	Master-slave	Master-Slave
Consistency	Yes	Yes	Yes	Yes	Yes	Yes
Sharding	Yes	Yes	Yes	Yes	Yes	Yes
Replication	Synchronous and Asynchronous	Asynchronous	Asynchronous	Asynchronous	Synchronous	Synchronous
Type of license	Proprietary	Open-source	Proprietary	Open-source	Open-source	Open-source
Features	Allows users to create and delete tables and column families, changing clusters, table and column family metadata	In-memory operations as well as DB management	Create, update and delete of tables and data items	In-memory operations, Hadoop integration, Create, delete, append tables	Runs atop of Apache HDFS, GlusterFS or the Kosmos File System (KFS). Hadoop Compatibility. Full DB management	Secondary indexes, dynamic queries, sorting, rich updates, upserts (update if document exists, insert if it doesn't), and aggregation

Table 3: Programming Languages for MapReduce

Language	Name of the language	Developed by	Type of language	Types of Data structures	Schema optional
Jaql	Jaql	IBM	Data Flow	JSON	Yes
Hive	HiveQL	Facebook	Declarative	Complex	No
Pig	Pig Latin	Yahoo!	Data Flow	Complex, Nested	Yes

well as Hive and Pig. Moreover, it offers a rich set of built-in functions for processing unstructured or semi-structured data.

Pig [12] was developed by Yahoo and uses Pig-Latin language, a data flow scripting language to process data in Hadoop systems. It combines SQL with MapReduce, providing a way to express common SQL operations such as "select", "join" and so on. A Pig script is a query execution plan or a dataflow graph that is compiled and optimized by MapReduce. Pig has a flexible data model that supports complex types such as set or map and also provides code debugging.

Concerning the programmer's point of view, the frameworks above improve the use of MapReduce by offering rich easy-to-use APIs as either declarative or data-flow languages. Declarative languages are those that specify what one wants to do rather than how to do it, and do not allow procedural programming. On the contrary, data-flow languages allow instruction and procedural programming.

These frameworks abstract the programmer from the whole system, making her/him comfortable within the MapReduce distributed programming model. Table 3 presents a comparison between these three programming languages developed for MapReduce. Note that, although these languages were developed to improve MapReduce jobs programming, they are very different in nature. Concerning the type of language, Pig and Jaql are Data Flow languages while Hive is declarative. Pig operates over nested complex data while Jaql operates over JSON. Complex data structures stands for arrays, collections, scalars and hashes [3], while nested data structure simply means that a collection can contain objects of its own data type. JSON data structures refers to documents as a way to store values. Finally, Pig and Jaql are schema optional while Hive is not [7].

3.3 Enhancements to MapReduce

Concerning analytics' algorithms, MapReduce lacks built-in support for iterative process, which is a crucial feature in many applications. Thus, several alternatives to MapReduce were developed, such as HaLoop, Twister, and Apache Spark:

Apache Spark [48] is a MapReduce based implementation that allows data reuse across multiple iterations. It offers MapReduce native scalability and fault-tolerance, adding support for in-memory processes. Spark introduces the concept of Resilient Distributed Datasets (RDDs), which are fault-tolerant distributed collection items that allow users to control partitioning and preserve intermediate results in memory, optimizing data location and data manipulation, using a rich set of operators. RDDs can be cached in memory or in permanent storage, which overcomes the handicaps of Twister that only keeps intermediate data in memory. However, Apache Spark does not support group Reduce functions, gathering all results through only one reduce function.

HaLoop [68], [69] is a highly available, fault-tolerant and scalable framework that provides support for iterative algorithms by scheduling tasks across iterations and using several caching mechanisms. The system reuses mappers and reducers assigned to the same MapReduce job. Moreover, it attributes tasks to the same nodes in a loop-like way. By caching loop invariants it employs few resources, reloading repeated information several times. There is also a local "Reducer" that tests the loop condition. HaLoop distributed programming system maintains fault tolerance from MapReduce.

Twister [37], [18] is an in-memory MapReduce, scalable and fault-tolerant framework. Therefore, it performs loops in MapReduce normal execution, triggering another MapReduce iteration. The system is optimized at runtime for iterative computations and was developed to retain in memory (if possible) intermediate data to reduce computation overhead. Moreover, Twister allows configuring static data in both Map and Reduce tasks; uses a higher granularity for the Map tasks; offers a new operation (Combine), which is another reduction level; and allows the implementation of a set of programming extensions to MapReduce. However, when comparing to Hadoop, this system is weak fault-tolerance.

3.4 Graph Databases

Several alternatives to MapReduce are based in different computation models. Apache Giraph, GraphLab and Pregel, are graph parallel computation models that appeared to solve graph based problems:

Apache Giraph [38] is an open source, master/slave architecture, graph parallel computation model that loads and splits graphs across workers by using MapReduce's Map function. The master decides when workers shall start computing consecutive supersteps. Apache Giraph's computation starts by assigning vertices to the workers. The computation is then performed by each worker and stops when all vertices are inactive. Synchronization is maintained during operations. Since it implements a MapReduce's Map function, it is able to run on an Hadoop infrastructure while providing Pregel's API and middleware. The system is fault-tolerant, provides high-availability and allows in-memory processing, being able to compute iterative algorithms.

GraphLab [70] is an asynchronous peer-to-peer system developed at Carnegie Mellon University as a highly scalable and fault-tolerant graph processing system. It runs efficiently in both shared and distributed-memory systems and provides MapReduce-similar functions: Update and Sync. The Update function reads and modifies overlapping sets of data, whereas the Sync function performs reductions in the background while other computation is being performed. Also, by using scheduling primitives, GraphLab controls the order in which update functions are executed.

The GraphLab execution model follows a single loop semantics. A computation consists of a set of vertices V, set of edges E, and user-defined data D, defined in a Graph G = (V, E, D), an update function and an initial set of vertices V to be executed. The system keeps adding and removing vertices to the set of vertices V and ends when there are no vertices in V. The resulting values are returned when the algorithm ends. This framework also includes toolkits for graph analytics, clustering, collaborative filtering, graphical models and so on.

Pregel [33], [23] is a highly available, scalable, and fault-tolerant synchronous system developed by Google to address many practical computational problems such as Web mining or social network graph. The system provides support for iterative algorithms, since it preserves the state (and the graph structure) of each vertex between the iterations. The input to a Pregel computation is a direct graph in which a vertex is uniquely identified by a vertex identifier. Additionally, each vertex is associated with a modifiable, user value. The edges are associated with their sources vertices and each edge consists of a modifiable, user defined value and a target vertex identifier. In a Pregel computation, the first operation performed is graph initialization. Then, a chain of supersteps allows vertices to be executed in parallel, executing the same user-defined function that expresses the logic for a given algorithm. A vertex can modify its state or the state of its outgoing edges. The algorithm ends, when all vertices are inactive.

3.5 GPU Processing

While hardware has been evolving through the years, data analytics tools have not kept up. The need for better algorithms has been an issue and their performance have been relying entirely on the capabilities of DBMSs for scaling-out. Recently, [41] and [22] presented new approaches for storing Big Data by means of GPU processing. MapD [49] is perhaps one of the most successful approaches that brings GPUs into the Big Data ecosystem.

This framework pairs "*off-the-shelf video game GPU cards with a new design for parallel databases*" [49]. It is packed in both Server and Desktop versions: MapD Server allows analyzing multibillion-row datasets by multiple simultaneous users. It is designed to run in headless server environments supporting up to 16 GPUs per server – eight GPU cards with 192GB total GPU memory can be installed in a single server, allowing data to be queried at rates approaching three terabytes per second by almost 40,000 cores; and MapD Desktop, which allows on-site analysis of hundreds of millions of rows of data at great speeds.

Both versions are packaged with a web-based frontend that enables easy-to-use, interactive visual analytics. However, it also provides SQL and user defined functions (UDF) so that complex analytics such as regression and deep learning can be performed directly on GPU query results.

3.6 Discussion

All discussed systems present options to provide availability, fault-tolerance and scalability. MapReduce and its alternatives, MapReduce, HaLoop, Twister and Apache Spark, are key/value systems and all support in-memory iterative algorithms (with exception of MapRedue) but all are data oriented. Pregel, GraphLab and Apache Giraph are Graph oriented systems. Table 4 presents an overview of the main features of these systems. MapD is not mentioned in the comparison because it is still in its early day and little information is available.

Comparing all systems studied within the scope of this paper, we can see that all provide fault-tolerance,

availability and scalability, which are three major characteristics regarding Big Data needs. Concerning iterative Analytics, only HaLoop, Twister, Apache Spark, Pregel, GraphLab and Apache Giraph are eligible. Table 5 presents a comparison overview concerning fault-tolerance, availability, scalability, replication, and in-memory operations for all systems. Furthermore, it also presents some of the Big Data vendors that sell these systems as services in the cloud.

4 BIG DATA IN THE CLOUD

Storing and processing big loads of data requires scalability, easiness-to-growth, fault tolerance and availability. Cloud computing is able to deliver all these characteristics through hardware virtualization. As such, Big Data and Cloud computing are two compatible concepts: Cloud can make Big Data available, scalable and fault tolerant.

With the rapid increase of data and the demand for new ways to store, process and analyze data, a new business area has emerged. Companies started to view Cloud and, more recently, Big Data as a valuable business opportunity. Several new companies such as Cloudera[5], Hortonworks[6], and Teradata[7] and many others, are now focused in deliver Big Data as a Service (BDaaS) or DataBase as a Service (DBaaS). Companies such as Google, IBM, Amazon and Microsoft also provide ways for costumers to consume Big Data on demand. Next, we present two case studies, Nokia and RedBus, which discuss the successful use of Big Data within Cloud environments.

4.1 Case Studies

Nokia[8], was one of the first companies to understood the value of Big Data and Cloud computing technologies together [20]. Several years ago, the company used individual DBMS to accommodate each application requirement. Although, realizing, the advantages of integrating all data into one application, Nokia decided to migrate to Hadoop based systems, integrating data within the same domain to get proper insights on how its clients interact with their applications. As Hadoop uses commodity hardware, the cost per terabyte of storage is cheaper than a traditional RDBMS. Moreover, it allows both structured and unstructured data [20]:

> *"The benefits of Hadoop were clear – it offers reliable, cost-effective data storage and high performance parallel processing of multi-structured data at petabyte scale".*

As Cloudera Distributed Hadoop (CDH)[9] bundles the most popular open source projects in the Apache Hadoop stack into a single, integrated package, with stable and reliable releases, it represents a great opportunity for implementing Hadoop infrastructures, transferring all specialized processes, IT preoccupations and technical concerns onto the vendors' specialized teams. Nokia regarded BDaaS as an advantage and trusted Cloudera's expertise to deploy a Hadoop environment that cope with its requirements in a short time frame.

Hadoop, and in particular Cloudera's Distributed Hadoop, strongly helped Nokia to fulfill their needs [20]:

> *"Hadoop was absolutely mission critical for Nokia…"* which can now *"…understand how people interact with the apps on their phones to view usage patterns across applications… and we wouldn't have gotten our Big Data platform to where it is today without Cloudera's platform, expertise and support".*

Nokia's case study shows that Big Data can be advantageous and how partnering with Big Data vendors can leverage an easy deployment of a Big Data solution.

RedBus[10] is India's largest online bus ticket and hotel booking company. They wanted to implement a powerful data analysis to gain insights over its bus booking service [34]. RedBus datasets could reach up 2 terabytes in size and the application would have to be able to analyze booking and inventory data across hundreds of bus operators serving more than 10.000 routes. Furthermore, the company wanted to avoid setting up and maintaining a complex in-house infrastructure.

At first, the company considered using clusters of Hadoop servers to process the data, although they realized it would take too much time to set up and would require specialized personal in order to maintain such infrastructure in-house.

The company considers Google BigQuery as the perfect match for their needs: "*We explored several data analytics solutions. Nothing comes remotely close to the sheer power of Google BigQuery - It made large-scale data collection and crunching possible with little effort*…" [34].

[5] http://www.cloudera.com/
[6] http://hortonworks.com/
[7] http://www.teradata.com/
[8] http://www.nokia.com/
[9] http://www.cloudera.com/
[10] https://www.redbus.in/

Table 4: Map Reduce, enhancements and alternatives

Database system	*MapReduce*	*Apache Spark*	*HaLoop*	*Twister*	*Apache Giraph*	*GraphLab*	*Pregel*
Main type	key/value	key/value	key/value	key/value	Graph	Graph	Graph
Data Types supported	Structured, Unstructured and semi-structured data	Unstructured and Structured data (Spark SQL[10])	Structured, Unstructured and semi-structured data	Structured, Unstructured and semi-structured data	Unstructured and Structured	Unstructured and Structured	Unstructured and Structured
Support for iterative algorithms	Not supported	Supported	Supported	Supported	Supported	Supported	Supported
Type of DB	Data Files	Data Files	Data Files	Data Files	Graphs	Graphs	Graphs
Architecture	Master-Slave	Master-Slave	Master-Slave	Master-Slave	Master-Slave	P2P	Master-Slave
Computation model	Synchronous	Synchronous and Asynchronous	Synchronous	Synchronous	Bulk-synchronous	Asynchronous	Bulk-synchronous
Type of software	Open source	Open source	Open source	Proprietary	Open source	Not fully open source	Proprietary
Features	Explores locality. Map and Reduce functions.	Allows on Hadoop, standalone or in the cloud. Allows streaming.	Caching options for loop-invariant data access. Allows the reuse of Hadoop implementations concerning fault-tolerance mechanisms. Explores locality	Allows to configure static data in both Map and Reduce tasks; Weakly fault-tolerant; Explores locality	High scalability, computation, sharded aggregators, edge-oriented input, out-of-core computation, and more	Runs on Hadoop. Supports analytics over Graphs	Provides Java API to develop distributed computing solutions to graph problems. It also explores locality

Table 5: Comparison of Big Data systems

System / Features	**Fault-tolerance**	**Availability**	**Scalability**	**Replication**	**Available as a service by a cloud vendor**	**In-memory operations**
BigTable	✓	✓	✓	Synchronous and Asynchronous	Google	X
Cassandra	✓	✓	✓	Asynchronous	Google and others	X
DynamoDB	✓	✓	✓	Asynchronous	Amazon	X
Hbase	✓	✓	✓	Asynchronous	Microsoft Azure HDInsights	X
HyperTable	✓	✓	✓	Synchronous	X	X
MongoDB	✓	✓	✓	Synchronous	MongoDirector	X
MapReduce (Hadoop)	✓	✓	✓	Synchronous	Cloudera, Hortonworks and others	X
Apache Spark	✓	✓	✓	Synchronous and Asynchronous	IBM, Microsoft Azure HDInsights	✓ (allows streamming)
HaLoop	✓	✓	✓	Synchronous	X	✓
Twister	✓	✓	✓	Synchronous	X	✓
Apache Giraph	✓	✓	✓	Bulk-synchronous	Microsoft Azure HDInsights (allows installation)	✓
GraphLab	✓	✓	✓	Asynchronous	Dato	✓
Pregel	✓	✓	✓	Bulk-synchronous	X	✓

Using BigQuery, allows RedBus to [34]:

- Know how many times costumers tried to find an available seat but were unable to do it due bus overload;
- Examine decreases in bookings;
- Identify server problems by quickly analyzing data related to server activity;

Useless to say that moving towards Big Data brought RedBus business advantage. Google BigQuery provides RedBus with real-time data analysis capabilities at 20% of the cost of maintaining a complex Hadoop infrastructure. As affirmed in [34]: "*The fast insights gained through BigQuery are also making RedBus a stronger company. By minimizing the time it takes staff members to solve technical problems…*".

As supported by the use cases of Nokia and RedBus, and also other examples described in [2] and [6], switching towards Big Data enables organizations to gain competitive advantage. Moreover, Big Data as a Service (BDaaS), provided by Big Data vendors allows companies to leave the technical details for Big Data vendors and focus on their core business needs. Table 5 presents Big Data vendors that provide the discussed systems. Hadoop open-source is used mainly by Cloudera and Hortonworks, although several other implementations are used by other vendors such as Microsoft. BigTable and DynamoDB are proprietary and are provided by Google and Amazon. MongoDB is provided by MongoDirector, while the remaining systems are implemented on-demand by Microsoft Azure, a Microsoft's platform that sells Big Data and Analytics as a Service.

4.2 Moving Big Data to the Cloud

Although this technology enables the deployment of Big Data DBMS, moving Big Data to the Cloud presents issues that are not easy to resolve. Security and privacy are issues that typically concern Big Data clients. The major problem resides in the transfer of big loads of data through the Web. Uploading a 2 terabyte dataset through the Web is not a fast task and can be insecure without proper security configurations. Regarding bandwidth bottleneck the most pragmatic solution is to physically send the hard drives to the data center so that data can be uploaded onto the Cloud. However, this can be hard to accomplish without any loss of data. There are always the possibility of traffic accident and HDD damage during the trip. Plus, the cost of transfer would be enormous.

A better solution resides in improve bandwidth or create new algorithms and protocols for better data transfer. As bandwidth improvement passes through improved hardware from source to destination, a dedicated connection can be set up. However, this is not feasible in all cases, as the client would lose flexibility, since data cannot be uploaded from wherever s/he wants.

Software improvement consists in the best solution for work around this issue. Regarding software development for improving data paths, [44] presents the implementation of two algorithms that optimize at any given time the choice of the data center for data aggregation and processing, as well as the routes for transmitting data there. The first is an online lazy migration (OLM) algorithm and the second is a randomized fixed horizon control (RFHC) algorithm. Another contribution comes from Aspera [16] that built atop of its FASP™ transport technology, a proprietary suit of on demand transfer products [16].

5 DISASTER RECOVERY

Data is extremely important for a corporation since it provides information over the client's needs and through it, powers business and marketing teams to keep up with current demands and predict new trends. As data is very valuable, any loss of data results in the loss of money and competitiveness. In case of emergency or hazardous accidents such as floods, fires and others, data loss needs to be minimal. Disaster recovery techniques were developed to assure that data are quickly available with minimal downtime and loss. However, although this is a very important issue, the number of research articles focusing this particular area is relatively small [62], [64], [65].

By its definition [14], Cloud provides a good way to minimize data loss during emergencies. There are two important coefficients that are imperative to minimize: Recovery Point Objective, which refers the acceptable amount of application data that is acceptable to loose; and Recovery Time Objective, which refers to the acceptable downtime of the system. The author of [65] states that it is important to define a disaster recovery plan that not only relies on backups to reset data but also consists in a set of procedures that should be tested at least twice each year. Works such as [45], [64], [62] and [65] study this issue in depth from several perspectives.

From a technical perspective, [65] presents a very good methodology, proposing a "multi-purpose approach, which allows data to be restored to multiple sites with multiple methods", ensuring a recovery percentage of almost 100%. Furthermore, this study states that usually, data recovery methods use what they call "single-basket approach", which means there is only one destination from which to secure the

restored data. This area is of most importance and further research need to be done to minimize loss of data in case of accident.

6 BIG DATA CHALLENGES, NEW TRENDS AND FUTURE DIRECTIONS

This section discusses Big Data challenges, trends and future directions that must be taken into consideration within the next years to keep up with the rapid growth of data.

6.1 Challenges of Big Data

The current state of the art of Big Data, and Big Data platforms in particular, presents several challenges that must be addressed to keep up with the ever increasing rate of data and business needs. Moving huge amounts of data into a Cloud based Big Data platform, as discussed previously, is not simple and corporations often opt to physically send the hard drives to the data centers so that data can be uploaded. However, because this is neither the most practical solution nor the safest, several algorithms have been developed through the years ease data upload for cloud systems. This issue consists in a severe bottleneck when using Big Data in the cloud [44], [16].

Other problems such as, disaster recovery, scalability, locality exploration and fault tolerance are also big data challenges. Regarding scalability, there is some concerning about Exaflop computing [5], [26], [63]. Today's supercomputers and clouds can deal with Petaflop data sets, however, dealing with Exaflop size data sets is still under study and it rises concerns, since high performance and high bandwidth is required to transfer and process such huge volumes of data over the network. Cloud computing is slower than supercomputers since it is limited by the existent bandwidth and it is very expensive concerning complex projects. High performance computers (HPC) are the most promising solutions, however, the annual cost of such a computer is tremendous. There are several problems in designing Exaflop HPCs, especially regarding efficiency in power consumption – here, solutions tend to be more GPU based instead of CPU based. There are also problems related with the high degree of parallelism needed among hundred thousands of CPUs.

Analyzing big data loads requires the improvement of Big Data and analytics algorithms that are able to correlate and extract information from unstructured and structured data. Integration protocols and API standards would enable users to easily manage data and switch among solutions. Furthermore, the creation of such standards would also enable big data users to wisely choose a big data vendor solely based on cost and performance services, since API standardization reduces vendor lock-in algorithms.

With the rapid growth of real-time applications such as traffic monitoring [52], weather monitoring [44], CCTV monitoring like "Singapore Safe City Pilot" [57], social networks monitoring such as Google+ Ripples [29] and others, the development of stream analysis and iterative algorithms that efficiently manage processing power and memory consumption is a major challenge for research teams.

Efficiently recognize and store essential information (value) is very important since huge amounts of data are worthless if they do not have any value at all [46]. Human collaboration within the data visualization field is also an important issue, since there are patterns that humans can easily detect but that computer algorithms can hardly find (e.g.: captchas). Ideally, Big Data analytics would not be only computational. Another challenge is cloud elasticity capabilities, which allows cloud suppliers to accommodate user needs when there are peaks of information. This is a gap of cloud computing that is currently under study [55].

6.2 New Trends of Big Data

Big Data is an emerging technology broadly discussed both in business and academics areas. As a hot topic it enables corporations and governmental institutions to gain deep insights over its clients/citizens, there is a tremendous interest in a constant development of new DBMS and algorithms that enable corporations to predict more accurately their client's behavior. This technology has particular interest for marketing and CRM (Customer Relationship Management) teams that can target campaigns much more focused, implementing a concept known as digital marketing [50].

Big Data's interest from corporations and governmental institutions inevitably boosts research for new frameworks to provide further insights in an efficient and rapid way [50]. This necessity for new technology and high skilled personal do not show any signal of slowing down in the next three years. As stated by McKinsey in [50], the demand for Big Data expertise and specially data scientists will increase by 2018.

6.3 Future Research Directions

As data keeps constantly increasing and there is no signal that this trend will stop, new frameworks that are able to handle Exaflop computing in a fast and efficient

way should be taken into consideration. New ways to transfer data on to the cloud and API standardization through platforms are needed.

As future directions, we also point out: i) Big data API standardization to prevent vendor lock-in platforms; ii) improving iterative and streaming algorithms; iii) improving fault-tolerance, locality exploration, scalability and disaster recovery; iv) develop better protocols for file transferring over the WAN; and (v) develop cloud abilities to adapt load peaks by providing elasticity to its consumers.

7 CONCLUSIONS

The volume of data generated by applications and gadgets is being produced in an ever increasing rate. Big Data is seen by industries as an efficient way to store and explore data, whereas Cloud, due to its high flexibility and scalability, is the appropriate framework concerning the storage of Big Data solutions.

This study focused particularly on database engines that are able to run over distributed systems. In other words, systems that would fit cloud architectures. Concerning Big Data, some of the most known implementations are Cassandra, MongoDB and DynamoDB. However, they do not perform well comparing to MapReduce, which provides fault-tolerance and high availability and performance while abstracting programmers from the system details. MapReduce, within its open source implementation (Hadoop), is in fact the best known Big Data framework. However it presents some problems: it is very much static; it is difficult to program jobs; and it does not support graph computing. To overcome these problems several solutions like Twister, Spark, Pregel and GraphLab arose. Pregel and GraphLab provides a way to compute graph problems like web mining or social network trending, whereas Twister and Spark try to overcome the MapReduce's lack of interactivity.

With this study we aim at providing a better understand of Big Data, Cloud and how these two concepts fit together. To conclude, we show that, although Big Data and Cloud computing match very well together, there are some issues that should be addressed in a near future.

ACKNOWLEDGEMENTS

This work was hosted by Carnegie Mellon University under the program CMU-Portugal undergraduate internships and partially financed by iCIS – Intelligent Computing in the Internet Services (CENTRO-07-ST24 – FEDER – 002003), Portugal.

We would also like to thank Prof. Marco Vieira for their insightful comments on the paper, as these comments led us to an improvement of the work.We would like to thanks Instituto Superior de Engenharia de Coimbra (ISEC) from Polytechnic of Coimbra for hosting this research. Furthermore, we would also like to acknowledge Accenture for always encouraging our work.

REFERENCES

[1] Apache Cassandra Project "Apache Cassandra TM 1.2," pp. 1–144, 2013. http://cassandra.apache.org/.

[2] AWS, "AWS Case Studies: Big Data." [Online]. Available: https://aws.amazon.com/pt/solutions/case-studies/big-data/. [Accessed: 13-Aug-2015].

[3] "Complex data structures." [Online]. Available: http://theory.uwinnipeg.ca/programming/node38.html. [Accessed: 13-Aug-2015].

[4] "Computing | CERN." [Online]. Available: http://home.web.cern.ch/about/computing. [Accessed: 13-Jun-2015], CERN, 2015

[5] "Exaflop Computing Will Save the World ... If We Can Afford It - Industry Tap." [Online]. Available: http://www.industrytap.com/exaflop-computing-will-save-world-can-afford/15485. [Accessed: 26-May-2015].

[6] "Hadoop & Big Data Case Studies | MapR." [Online]. Available: https://www.mapr.com/resources/customer-case-studies. [Accessed: 13-Aug-2015].

[7] "JSON Example." [Online]. Available: http://json.org/example. [Accessed: 13-Aug-2015].

[8] "MySQL Cluster 7.2.1." Copyright © 2015 Oracle and/or its affiliates

[9] "Top 10 Strategic Technology Trends | Gartner." [Online]. Available: http://www.gartner.com/technology/research/top-10-technology-trends/. [Accessed: 13-Jun-2015].

[10] A. Fernández, S. del Río, V. López, A. Bawakid, M. J. del Jesus, J. M. Benítez, and F. Herrera, "Big Data with Cloud Computing: an insight on the computing environment, MapReduce, and programming frameworks," Wiley Interdiscip. Rev. Data Min. Knowl. Discov., vol. 4, no. October, pp. 380–409, 2014.

[11] A. Garzo, A. a. Benczur, C. I. Sidlo, D. Tahara, and E. F. Wyatt, "Real-time streaming mobility analytics," Proc. - IEEE Int. Conf. Big Data, Big Data, pp. 697–702, 2013.

[12] A. Gates, "Programming Pig", Yahoo!, Inc. First Edit. O'Reilly, 2011.

[13] A. Khetrapal, and V. Ganesh, "HBase and Hypertable for large scale distributed storage systems A Performance evaluation for Open Source BigTable Implementations," Evaluation, p. 8, 2006.

[14] A. Srinivas, Y. S. Ramayya, and B. Venkatesh, "A Study on Cloud Computing Disaster Recovery," pp. 1380–1389, 2013.

[15] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive," Proc. VLDB Endow., vol. 2, pp. 1626–1629, 2009.

[16] Aspera Inc., "Taking Big Data to the Cloud."

[17] B. D. Analytics, "EMC Accelerates Journey to Big Data with Business Analytics-as-a-Service an account of EMC IT's transformation to empower business and IT users with streamlined access to Big Data Analytics," no. January 2013, pp. 1–12. 01/13 EMC white Paper H11259.

[18] B. Zhang, Y. Ruan, T. Wu, J. Qiu, A. Hughes, and G. Fox, "Applying Twister to Scientific Applications." Indianapolis, pp. 25-32, 2010.

[19] C. C. Aggarwal, N. Ashish, and A. Sheth, "Chapter 12 the Internet of Things : A Survey from the Data-Centric," Manag. Min. Sens. Data, pp. 383–428, 2013.

[20] Cloudera, "Case Stundy Nokia: Using Big Data to Bridge the Virtual & Physical Worlds," 2012.

[21] D. A. Marcos, N. C. Rodrigo, B. Silvia, a. S. N. Marco, and B. Rajkumar, "Big Data Computing and Clouds: Challenges, Solutions, and Future Directions," pp. 1–44, 2013.

[22] D. Merrill, M. Garland, and A. Grimshaw, "Scalable GPU graph traversal," ACM SIGPLAN Not., vol. 47, no. 8, p. 117, 2012.

[23] D. Yan, J. Cheng, K. Xing, Y. Lu, W. Ng, and Y. Bu, "Pregel Algorithms for Graph Connectivity Problems with Performance Guarantees," vol. 7, no. 14, pp. 1821–1832, 2014.

[24] Datastax Corporation, "Introduction to Apache Cassandra", no. July, pp. 1–11, 2013.

[25] E. Dede, B. Sendir, P. Kuzlu, J. Hartog, and M. Govindaraju, "An evaluation of cassandra for hadoop," IEEE Int. Conf. Cloud Comput. CLOUD, pp. 494–501, 2013.

[26] E. Hpc, "The Challenge of Energy-Efficient HPC," SciDAC Rev., pp. 50–57, 2009.

[27] EMC Corporation, "Big Data-as-a-Service," no. July 2012, pp. 1–16, 2012.

[28] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. a. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," 7th Symp. Oper. Syst. Des. Implement, November 6-8, Seattle, WA, USA, pp. 205–218, 2006.

[29] F. Viégas and M. Wattenberg, "Google+ ripples: A native visualization of information flow," Proc. 22nd Int. Conf. World Wide Web, pp. 1389–1398, 2013.

[30] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo : Amazon's Highly Available Key-value Store," ACM SIGOPS Oper. Syst. Rev., vol. 41, pp. 205–220, 2007.

[31] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo," ACM SIGOPS Oper. Syst. Rev., vol. 41, p. 205, 2007.

[32] G. M. Carstoiu Dorin, Lepadatu Elena, "Hbase - non SQL Database, Performances Evaluation," Int. J. Adv. Comput. Technol., vol. 2, pp. 99–110, 2010.

[33] G. Malewicz, M. Austern, and A. Bik, "Pregel: a system for large-scale graph processing," Proc. 2010 ACM SIGMOD Int. Conf. Manag. Data, pp. 135–146, 2010.

[34] Google Cloud Platform, "Travel Agency Masters Big Data with Google BigQuery," 2006.

[35] I. A. 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," Inf. Syst., vol. 47, pp. 98–115, 2014.

[36] J. a. González-Martínez, M. L. Bote-Lorenzo, E. Gómez-Sánchez, and R. Cano-Parra, "Cloud computing and education: A state-of-the-art survey," Comput. Educ., vol. 80, pp. 132–151, 2015.

[37] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, "Twister : A Runtime for Iterative MapReduce," 19th ACM Int. Symp. High Perform. Distrib. Comput., pp. 810–818, 2010.

[38] J. Gonzalez, Y. Low, and H. Gu, "Powergraph: Distributed graph-parallel computation on natural graphs," 10th USENIX Conf. Oper. Syst. Des. Implement., pp. 17–30, 2012.

[39] J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL database," 6th Int. Conf. Pervasive Comput. Appl. ICPCA 2011, pp. 363–366, 2011.

[40] J. Shafer, S. Rixner, and A. L. Cox, "The Hadoop distributed filesystem: Balancing portability and performance," IEEE Int. Symp. Perform. Anal. Syst. Softw., pp. 122–133, 2010.

[41] J. Zhang, S. You, and L. Gruenwald, "Tiny GPU Cluster for Big Spatial Data: A Preliminary Performance Evaluation," IEEE 35th Int. Conf. Distrib. Comput. Syst. Work., pp. 142–147, 2015.

[42] K. Beyer, V. Ercegovac, and R. Gemulla, "Jaql: A scripting language for large scale semistructured data analysis," VLDB, pp. 1272–1283, 2011.

[43] L. George, HBase: The Definitive Guide. 2011.

[44] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving big data to the cloud," INFOCOM, pp. 405–409, 2013.

[45] J. S. P. Pokharel, "Disaster Recovery for System Architecture Using Cloud Computing," Appl. Internet, 2010.

[46] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. a. S. Netto, and R. Buyya, "Big Data computing and clouds: Trends and future directions," J. Parallel Distrib. Comput., vol. 79–80, pp. 3–15, 2014.

[47] "MongoDB Documentation," pp. 1–687, 2013.

[48] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark : Cluster Computing with Working Sets," 2nd USENIX Conf. Hot Top. Cloud Computing, p. 10, 2010.

[49] MapD. Corporation, "MapD," 2015.

[50] McKinsey & Company, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Glob. Inst., no. June, p. 156, 2011.

[51] MongoDB Documentation Project, "Sharding and MongoDB", pp. 1–80, 2015.

[52] N. J. Ferrier, S. M. Rowe, and a Blake, "Real-Time Traffic Monitoring," Workshop on Applications of Computer Vision, Sarasota, FL, USA, pp. 81-88, 1994.

[53] P. F. V. Abramova, J. Bernardino, "NoSQL databases: MongoDB vs cassandra," C3S2E, Porto, Portugal, 2013.

[54] P. F. V. Abramova, J. Bernardino, "Testing Cloud Benchmark Scalability with Cassandra," Services, Anchorage, AK, 2014.

[55] S. Das, "Scalable and Elastic Transactional Data Stores for Cloud Computing Platforms," no. December, pp. 1–278, 2011.

[56] S. Hurst. G. Simon, "Just how smart are Smart Cities? It's Time for Digital", Accenture, 2014.

[57] S. Industry and P. Office, "Singapore Government : Safe City Test Bed," 2014 Accenture, 2014.

[58] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, "Big Data: Issues and Challenges Moving Forward," 46th Hawaii Int. Conf. Syst. Sci., pp. 995–1004, 2013.

[59] S. Khan and P. V. Mane, "SQL Support over MongoDB using Metadata," vol. 3, no. 10, pp. 1–5, 2013.

[60] S. Sakr and M. M. Gaber, "Large Scale and Big Data", First Edit, 2014. Auerbach Publishers Inc.

[61] S. Sengupta, "Delivering Database as a Service (DBaaS) using Oracle Enterprise Manager 12c," Oracle, 2013.

[62] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," J. Netw. Comput. Appl., vol. 34, no. 1, pp. 1–11, 2011.

[63] T. Geller, "Supercomputing's exaflop target," Commun. ACM, vol. 54, no. 8, p. 16, 2011.

[64] T. Wood, E. Cecchet, K. Ramakrishnan, P. Shenoy, J. van der Merwe, and A. Venkataramani, "Disaster recovery as a cloud service: Economic benefits & deployment challenges," 2nd USENIX Work. Hot Top. Cloud Computing, pp. 1–7, 2010.

[65] V. Chang, "Towards a Big Data system disaster recovery in a Private Cloud," Ad Hoc Networks, pp. 1–18, 2015.

[66] V. Kalavri and V. Vlassov, "MapReduce: Limitations, optimizations and open issues," Proc. - 12th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust., pp. 1031–1038, 2013.

[67] VoltDB. Welcome to VoltDB: A Tutorial, VoltDB, Inc., 2015.

[68] Y. Bu, B. Howe, and M. D. Ernst, "HaLoop : Efficient Iterative Data Processing on Large Clusters," Proc. VLDB Endow., vol. 3, no. 1, pp. 285–296, 2010.

[69] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "The HaLoop approach to large-scale iterative data analysis," VLDB J., vol. 21, no. 2, pp. 169–190, 2012.

[70] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed GraphLab: A framework for machine learning and data mining in the cloud," Proc. VLDB Endow., vol. 5, pp. 716–727, 2012.

AUTHOR BIOGRAPHIES

Pedro Neves is affiliated with Instituto Superior de Engenharia de Coimbra (ISEC) - Polytechnic of Coimbra and is currently a visitor researcher at Carnegie Mellon University. Pedro received its bachelor degree in Computer Science from ISEC in 2012 and also the MSc degree in 2015. During his MSc he worked as a researcher in both in FCT and ISEC and afterwards, held the position of Business Intelligence and data analytics associate at Accenture Technology Solutions. His main interests are Data Science, Big Data, distributed data bases, Cloud computing and technics for scalability and elasticity within cloud environments.

Dr. Jorge Bernardino received the degree in computer engineering in 1987, the master's degree in systems and information technologies in 1994, and the PhD degree in computer science from the University of Coimbra in 2002. He is a Coordinator Professor at ISEC (Instituto Superior de Engenharia de Coimbra) of the Polytechnic Institute of Coimbra, Portugal. His main research fields are big data, data warehousing, business intelligence, database knowledge management, e-business and open source tools, subjects in which he has authored or co-authored dozens of papers in refereed conferences and journals. He has served on program committees of many conferences and acted as a referee for many international conferences and journals in data warehousing and databases. He was President of ISEC from 2005 to 2010. During 2014 he was Visiting Professor at CMU – Carnegie Mellon University.