Casado, R and Younas, M

Emerging trends and technologies in big data processing

Casado, R and Younas, M (2015) Emerging trends and technologies in big data processing. *Concurrency and Computation: Practice and Experience*, 27 (8). pp. 2078-2091.

doi: 10.1002/cpe.3398

Emerging Trends and Technologies in Big Data Processing

Rubén Casado[a], Muhammad Younas[b]

[a]*Department of Research & Innovation, Treelogic, Asturias, Spain* ruben.casado@treelogic.com
[b]*Department of Computing and Communication Technologies, Oxford Brookes University, Oxford, UK,* m.younas@brookes.ac.uk

Abstract: Big Data encompasses large volume of complex structured, semi-structured and unstructured and data which is beyond the processing capabilities of conventional databases. The processing and analysis of Big Data now play a central role in decision making, forecasting, business analysis, product development, customer experience and loyalty, to name but a few. In this paper, we examine the distinguishing characteristics of Big Data along the lines of the 3Vs: *variety*, *volume* and *velocity*. Accordingly, the paper provides an insight into the main processing paradigms in relation to the 3Vs. It defines a lifecycle for Big Data processing and classifies various available tools and technologies in terms of the lifecycle phases of Big Data, which include data acquisition, data storage, data analysis and data exploitation of the results. This paper is first of its kind that reviews and analyses current trends and technologies in relation to the characteristics, evolution, and processing of Big Data.

1. INTRODUCTION

Big Data is a collection of data sets which are enormously large and complex that conventional database systems cannot process within desired time. For instance, storing and processing of daily tweets at Twitter demand significant data storage, processing and data analytics capabilities — e.g., find correlations between millions of tweets or analyse the demographics of users. Though conventional SQL-based databases have proved to be highly efficient, reliable and consistent in terms of storing and processing structured (or relational) data, they fall short of processing Big Data which is characterized by large volume, variety, velocity, openness, inappropriate structure, and visualization among others [1].

Big Data is set to play a major role in various domains such as science, research, engineering, medicine, healthcare, finance, business, and ultimately society itself [2]. It can be used for analyzing and forecasting business trends, profit and loss, identify real time road traffic conditions, healthcare, weather information and so on.

Big Data is generally characterized by the 3Vs: *variety, volume* and *velocity*. *Variety* refers to the nature and structure of the information that constitute the Big Data. *Velocity* refers to the frequency of data generation as well as the dynamic aspects of the data. *Variety* refers to the multimodal nature of data such as different data schemas of data sources; structured data like ontologies and unstructured data like sensor signals [3].

Further, the processing, availability or acquisition of Big Data can be classified into different categories, including: batch processing, real time processing and hybrid processing. Batch processing is an efficient way of processing high volumes of data which is collected over a period of time. In this scheme, data is collected, stored/inserted into the data sources and processed. The batch results are then produced. However, several applications require real-time processing of data (streams) that is acquired from heterogeneous data sources. Real time processing involves continuous input, processing and output of data [3]. The low latency is the main goal of this processing paradigm. That is, data must be processed in a small (or near real) time period. Application domains include smart cities, entertainment, and disaster management.

Note that batch processing provides rigorous results since it can use more data and perform better training of predictive models. But it is not feasible for domains which need low response time. Real time processing generally ensures low response time. However, low response time can be achieved at the expense of less rigorous analysis of data. The hybrid approach is therefore required so that application domains (using Big Data) can benefit from both batch and real time processing. To obtain desired results under this approach, both batch and real-time results are queried. The results are then merged together, synchronized or composed. Data acquisition and analysis become more complicated under the approach.

The rest of the paper is organized as follows. Section 2 illustrates the characteristics of Big Data and related projects. Section 3 reviews and analyses various techniques used in processing and analyzing Big Data. Section 4 gives details on the data models and related Big Data storage systems. Section 5 describes the life cycle of Big Data processing and reviews existing tools and technologies according. Section 6 presents the summary and recommendations for future Big Data models, applications, and technologies.

2. BIG DATA CHARACTERISTICS AND ILLUSTRATIVE SCENARIO

Big Data is generally defined as a massive volume of structured, semi-structured and/or unstructured data, which may not be effectively managed and processed using traditional databases and software techniques [4]. Big Data is of high-volume, high-velocity, and contain a variety of information that require new management and processing methods so as to enable enhanced decision making, forecasting, business analysis, customer experience and loyalty, and process optimization in various organizations, industries, and online social networks. Traditional software cannot manage the Big Data due to the high volume, velocity and variety (3Vs) [47].

- **Volume:** refers to the size of the data to be processed. Volume of Big Data goes far beyond the conventional limits of megabytes or gigabytes and reaches the terabytes or even petabytes.
- **Velocity:** refers not only to frequency of the data generation, but also the dynamic aspects of the data as well as the need of generating the results in real-time.
- **Variety:** refers to the multimodal nature of data. That is, the different sources of information and the different data schemas of each source, e.g., structured data like ontologies and unstructured data like sensor signals.

Big Data has the potential to become the main enabler of decision making by penetrating in all walks of our modern society, including retail, manufacturing, healthcare, economics, finance, sciences and environment, road traffic and weather among others. In such domains, an enormous volume of data is generated on daily bases, for example, data streams of sensors network, online reviews and discussions, environmental data, weather data, and road traffic monitoring.

Realizing the benefits and importance of Big Data various development and research projects have been initiated by industry, academic institutes as well as governmental organizations. In the following we illustrate a scenario in order to explain and understand the characteristics and the processing paradigms of Big Data.

Big Air Quality Scenario:

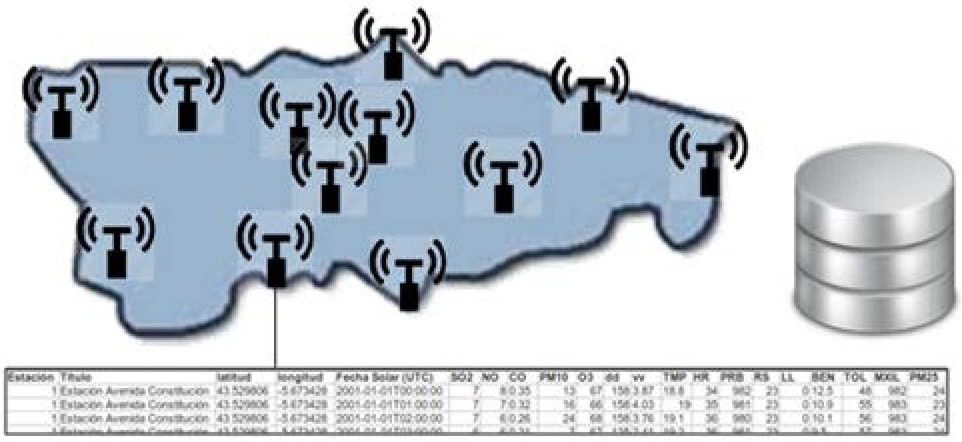
The Big Air Quality Data Scenario aims to research into the application of Big Data technology in order to capture, store and analyze the information about air quality in a location like Asturias — a region in the north of Spain with over 1 million of population.

In Asturias (and other regions), heavy transportation, heating, industry and waste incineration contribute heavily to the increasing levels of environmental pollution. Consequently, such pollution generates particles whose presence in the atmosphere has negative impact on people’s health. It is therefore necessary to analyze the presence of pollutants in the atmosphere. However to carry out such analysis it is required to generate, gather, store and process a very large volume of (big) data.

As depicted in Figure 1, in Asturias there are about one hundred stations that provide the information continuously. The data sent by the stations include: station id, name, location, time, and different measures of different components of air. It is expected that the volume of data will increase in future as the government plans to provide facilities for getting information from new sources such as satellite images, mobile sensors, meteorological data, social networks, etc. In addition, the government has the historical data of all stations from more than ten years ago.

Data acquired and processed in the Big Air Quality Scenario possess the characteristics of 3Vs. First the data has large volume given that it is acquired from different sources. Second, it is very dynamic and has high frequency of generating new data. Third, the data is of multimodal nature coming from different sources with different structures.

Further, in this scenario, three data processing paradigms are found very important. Batch processing is needed to analyse all historical data in order to identify patterns in the environment. Real-time processing is needed to monitor the status and send alerts to the authorities if necessary. A hybrid computation model is required to make predictions based on the previous behaviour and the current situation. These different processing domains are explained in the subsequent sections of this paper.



Estación	Título	latitud	longitud	Fecha Solar (UTC)	SO2	NO	CO	PM10	O3	dd	vv	TMP	HR	PRB	RS	LL	BEN	TOL	MXIL	PM25
1	Estación Avenida Constitución	43.529806	-5.673428	2001-01-01T00:00:00	7	8	0.35	13	67	158	3.87	18.8	34	982	23	0	12.5	48	982	24
1	Estación Avenida Constitución	43.529806	-5.673428	2001-01-01T01:00:00	7	7	0.32	16	66	158	4.03	19	35	981	23	0	10.9	55	983	23
1	Estación Avenida Constitución	43.529806	-5.673428	2001-01-01T02:00:00	7	6	0.26	24	68	158	3.76	19.1	36	980	23	0	10.1	56	983	24

Figure 1. Big Air Quality Scenario

3. PROCESSING PARADIGMS OF BIG DATA

This section describes the processing paradigm of Big Data and the time line. Big Data requires new methods, tools and techniques for solving new problems that emerge from their unique characteristics such as volume, variety and velocity. As depicted in Figure 2, the common processing paradigms (or solutions) for Big Data characterized by 3Vs, include: batch processing, real-time processing and hybrid processing models.

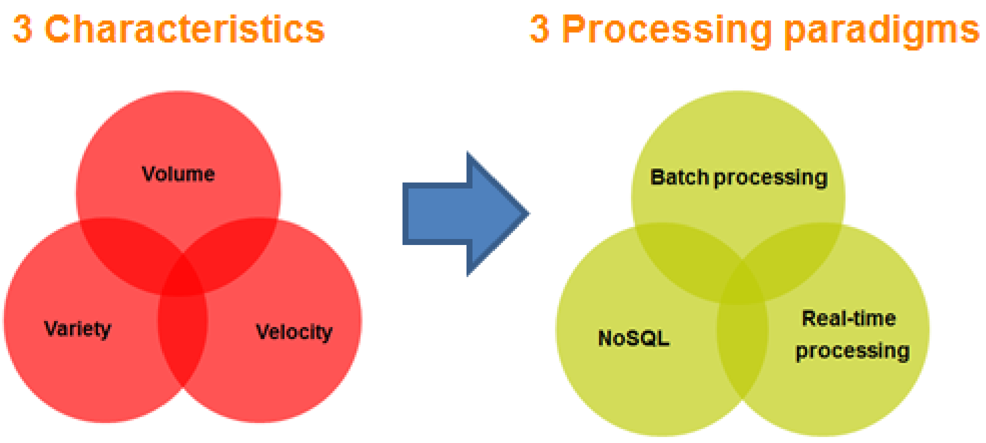


Figure 2. Big Data issues and Possible Solutions

Batch processing is a possible processing paradigm (or solution) for dealing with volume issue, whilst real-time processing can address the velocity issue. Hybrid approach can deal with both volume and velocity issues in application domains that require analysis of large amount of static as well as dynamic (or streaming) data. It combines the results coming from both batch and real-time processing. The characteristic of 'variety' is believed to be common to all the three processing paradigms.

Figure 3 shows the different generations of Big Data and the processing paradigms [5]. Batch processing (for Big Data) was started in 2003 when Google published its paper on Google File System (GFS) [6] and MapReduce framework [7]. But in those days businesses and organisations were not obviously faced with Big Data problems. So it can be said that the first generation of Big Data was started in 2006 when Hadoop [8] was born. Hadoop is one of the reliable technologies, that is, widely used for batch processing. Our research identifies that there are no latest developments in batch processing technologies, which marks the end of the first generation.

The second generation can be regarded as the real-time processing. Companies like Yahoo and Twitter had confronted situations wherein that they had to deal not only with big static data, but also with big real time (streaming) data. In order to deal with real time Big Data processing, Yahoo developed, in 2010, one of the first technologies, called S4 [9]. Other companies have relatively recently developed their own technologies for processing Big Data. For example, Linkedin developed Samza [10] and Google developed Millwheel [11]. We therefore see that the real time paradigm is still under development as new technologies are emerging and currently there is no de facto technology as that of Hadoop for batch processing.

The Hybrid processing model started with the definition of the Lambda Architecture [12] in 2012. But it is still under development. This is considered to be one of the challenging technologies in coming years.

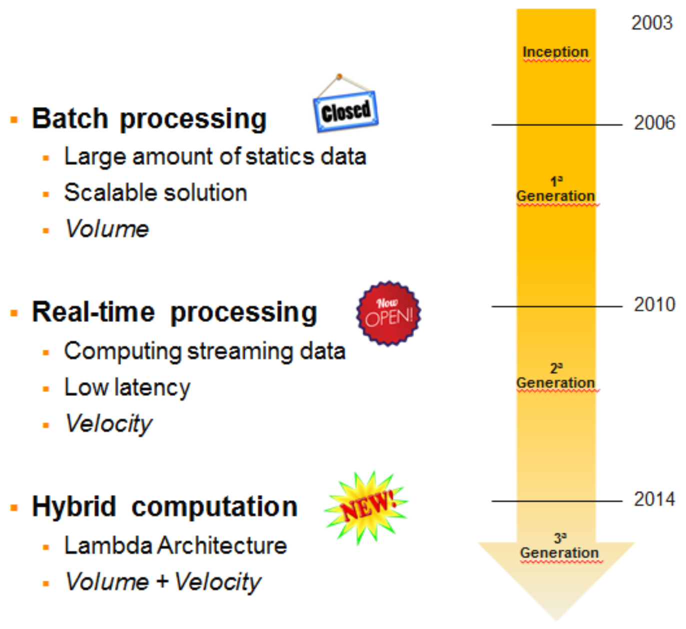
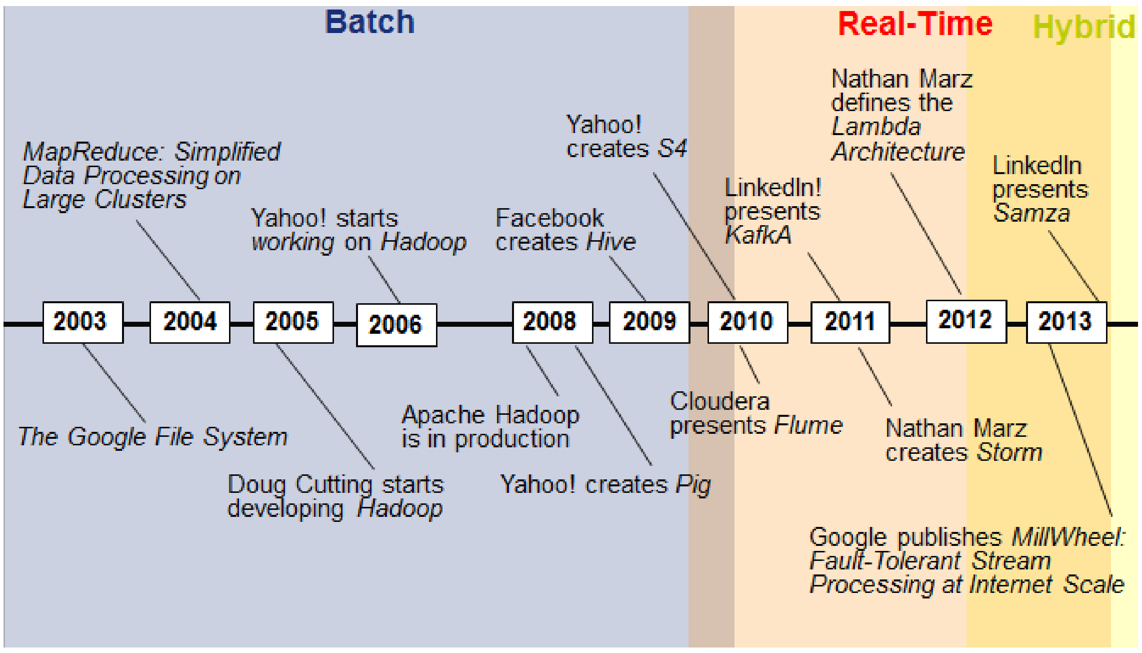


Figure 3. Processing Paradigms

Figure 4 shows the timeline of the Big Data processing paradigms and technologies. In 2003 and 2004 Google presented its papers on GFS [6] and MapReduce [7]. Doug Cutting based on those papers, started developing Hadoop in 2005 [13]. Doug Cutting moved to Yahoo!. At Yahoo, Cutting’s project was given high importance. The company liked the project and further worked on it. In 2008 Yahoo! released a stable version of Hadoop. This was followed by Facebook and Yahoo started working on abstract layers over MapReduce. Yahoo! presented Pig [14] in 2008 and Facebook presented Hive [15] in 2009.

In 2010, the second generation technologies for processing Big Data were emerged when Yahoo! developed S4 [9]; the first framework for real-time processing. Another key milestone in the 2nd generation was Storm [16]. This was created by Nathan Marz and was released as open source by Twitter. Other companies like Cloudera or Linkedin presented interesting technologies such as Flume [17] and Kafka [18]. In 2013, Google presented its paper on MillWheel [11] for dealing with real-time data processing. LinkedIn also released Samza [10], which is used for real-time data processing.

The inception of the third generation of Big Data processing is in 2012 when Nathan Marz developed the Lambda Architecture [12]. But it is still early to say that we have started the 3rd generation although there are some promising approaches as described in Section 4.



Batch
Real-Time
Hybrid
MapReduce: Simplified Data Processing on Large Clusters
Yahoo! starts working on Hadoop
Facebook creates Hive
Yahoo! creates S4
LinkedIn! presents KafkA
Nathan Marz defines the Lambda Architecture
LinkedIn presents Samza
2003
2004
2005
2006
2008
2009
2010
2011
2012
2013
The Google File System
Doug Cutting starts developing Hadoop
Apache Hadoop is in production
Yahoo! creates Pig
Cloudera presents Flume
Nathan Marz creates Storm
Google publishes MillWheel: Fault-Tolerant Stream Processing at Internet Scale

Figure 4. Big Data Processing Paradigms, Technologies and Timeline

3.1. BATCH PROCESSING

Batch processing is the solution to process large volume of static data. We say that batch processing uses static data because it works with data that is already in the system (data storages). This paradigm does not take into account new data once a batch processing has started.

The main feature of batch processing system is the scalability. To achieve high scalability and dealing with the volume problem, batch processing uses a parallel distributed processing framework such as MapReduce. MapReduce is the de facto standard technology for batch processing. It has several advantages: (i) it allows for a simple and unifying view of data; (ii) it is inherently scalable; (iii) it effectively hides the complexity of programming distributed software, which is challenging due to potential hardware failures, fluctuations in network quality, and device heterogeneity. MapReduce also has some limitations or constraints in certain settings that must be addressed. For instance, many analysis/mining tasks in real time systems or applications have to run iteratively or in multiple rounds. This is difficult to do in MapReduce. Several recent implementations try to address this shortcoming. Further, additional development for real-time and streaming computation, as well as optimisation of data access and indexing is required for efficient data analysis.

Overall the batch processing paradigm is more reliable, but batches can take longer to complete. Thus they are not suitable for low latency applications. Further, batches cannot be interrupted or reconfigured on-the-fly if new data arrive. An example of batch processing is to analyze web logs of a website in order to identify customers buying patterns. Currently batch Big Data analytics are applied to social networking applications, graph mining, scientific applications, and others.

3.2. REAL TIME PROCESSING

The goal of real-time processing paradigm is to deal with velocity of Big Data such as processing streaming data but with low latency.

This processing paradigm is based on more or less the same principles as those of batch processing such as distribution and parallelism. In order to achieve low latency, this processing paradigm analyses small sets of data that are stored in memory. So real-time processing is something like an infinite sequence of small batch processing where the information is in memory instead of disks (secondary storages) – in other words, it uses diskless approach. An example of real-time processing is to define current or trending topics at Twitter.

Several applications require real-time processing of data streams from heterogeneous sources. Examples include: *Smart cities* — manage transportation, energy supply, and garbage collection; *Disaster management* — especially through data gathered from emergency management sources, citizens' usage of social networks and mobile devices; *Production and logistics* — using factories' sensors for quality control and product optimization and saving resources; *Entertainment* — analyzing streaming data from music, TV and gaming platforms for recommendations, analysis of users, and advertisement.

3.3. HYBRID COMPUTATION

Many application domains require the combination of batch and real time processing paradigms. This is achieved through a hybrid model. This model is also known as Lambda Architecture [12], which contains:

Batch layer (batch processing) — manages the master dataset which is not changeable and is stored in a distributed file system; Serving layer (batch results) — loads and exposes the batch views in a datastore so that they can be queried; Speed layer (real-time processing) — deals only with new data that require low latency.

To obtain a complete result, the batch and real-time views must be queried and the results be merged together. Synchronization, results composition and other non-trivial issues have to be addressed at this stage, which is a part of the Combination layer.

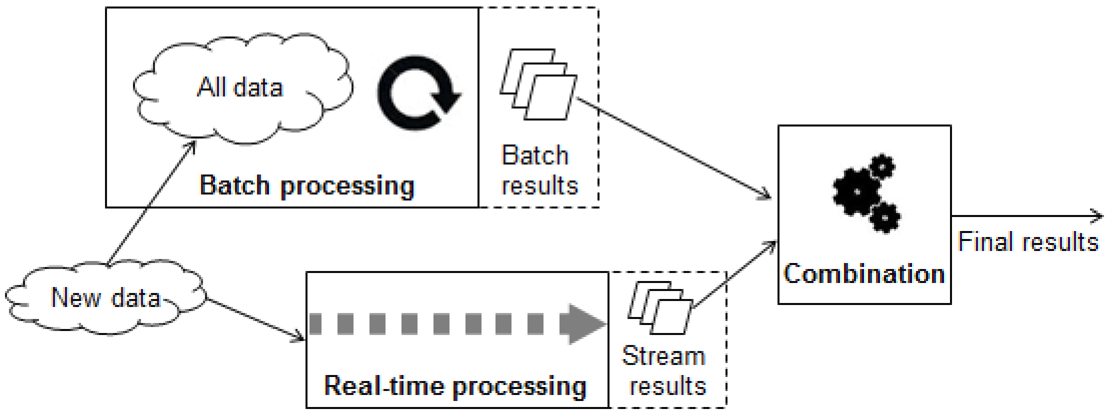


Figure 5. Hybrid Computation model

Figure 5 depicts a high level architecture of a hybrid processing model [12]. In this model there are three layers: batch, real-time and combination. The new data is duplicated and sent to both bath and real-time layers. Batch layer is processing in a loop the whole dataset. But a batch job takes long time to finish so new information may arrive during the process and such information is not taken into account by the batch layer. To compensate this delay, the real-time layer processes only the new data that has not been analyzed by the batch layer. Each layer stores its partial results in a database that is consulted by the Combination layer in order to obtain in real-time the final updated results.

4. DATA MODELS AND SYSTEMS

Compared to classical databases, Big Data applications are more demanding in terms of concurrency, latency, efficiency, economy of storage, access requirements and operational costs. To meet such needs, a variety of new types of databases have emerged which are different from traditional relational databases. They are generally referred to as NoSQL (Not only SQL) databases [19].

In 2007 Amazon, and other companies, experienced huge growth in their data and thus faced with the problem of managing and processing such data. Amazon therefore developed, Dynamo[20], which was one of the NoSQL databases. Another very important development came from Google. In 2008, they created a new NoSQL data store called, BigTable [21]. Though the overall problem of managing the huge amount of data was the same as that of Amazon, Google was more focused on bulk processing than on real time queries and processing. In 2011 they published a new technology, called Dremel [22] that provides a scalable, interactive ad-hoc query system with very low latency in Big Data sets. This technology complemented the BigTable storage systems and inspires other technologies like Apache Drill [23].

The objective of NoSQL databases is to provide good horizontal scalability for simple read/write database operations distributed over many servers. In contrast, traditional SQL databases have comparatively little or no ability to scale horizontally [24]. A key feature of NoSQL systems is shared nothing horizontal scaling – replicating and partitioning data over many servers. This allows them to support a large number of simple read/write operations.

The NoSQL systems, discussed in this paper, generally do not provide ACID transactional properties: updates are eventually propagated but with limited guarantees of consistency of read operations. Some authors suggest a BASE (Basically Available, Soft state, Eventually consistent) [25] model in contrast to the ACID (Atomicity, Consistency, Isolation, and Durability). The idea is that by giving up ACID constraints, one can achieve much higher performance and scalability. A key concept to understand the NoSQL properties is the so-called CAP theorem (Figure 6). In 2000, Professor Eric Brewer put forward the famous CAP theorem. That is, Consistency, Availability, and tolerance of network Partition. CAP theorem's core idea is that a distributed system cannot meet the three distinct needs simultaneously. Instead it can only meet two. Different NoSQL systems have been designed with the aim of achieving the two features specified in the CAP theorem.

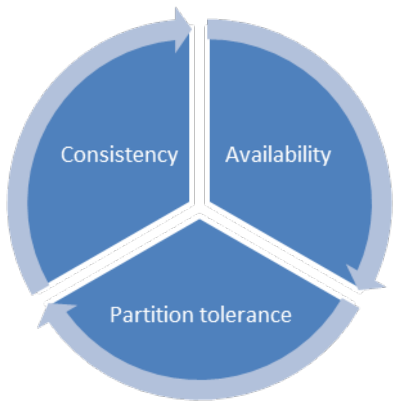


Figure 6. CAP theorem

Traditional databases are mainly based on relational data model. Their objective was to support associated class operations and ACID transactions. But in the NoSQL databases, the mainstream data models are as follow [26]:

- **Key-value**: It allows application developer to store schema-less data. This data consists of a key which is represented by a string and the actual data which is the value in key-value pair. The data can be any primitive of programming language, which may be a string, an integer or an array or it can be an object. Thus it loosens the requirement of formatted data for storage, eliminating the need for fixed data model. Example of key-value systems are HBase [27] and Redis [28].
- **Document-store**: Document Store, also commonly known as Document Oriented Database, is basically a software system used for storing, retrieving, updating data stored in database. The underlying storage structure used in such databases is a document. Each Document Store differs in its implementation of data. However each of it assumes that data is enclosed and encoded in some standard format which may be XML, JSON, BSON, PDF or Microsoft office. Each document is represented by a unique key which is a string (URI or path). An API or a query language is provided for fast retrieval of documents on the basis of its content. Examples of Document-oriented stores are MongoDB [29] and CouchDB [30].
- **Graph**: Graph databases are schema-less databases which use graph data structures along with nodes, edges and certain properties to represent data. Nodes may

represent entities like people, business or any other item similar to what objects represent in any programming language. Properties designate any information related to nodes. On the other hand, edges relate a node to other node or a node to some property. One can obtain some meaningful pattern or behavior after studying the interconnection between nodes, properties and edges. Examples of graph databases are Neo4J [31], Apache Giraph [32], and Google's Pregel [33].

- **Column-oriented**: Column Store Databases, unlike Row Databases, store their data in the form of columns. It serializes all the values of one column together. Column-oriented databases are comparatively efficient than row oriented ones [34]. Examples of column-oriented systems are Cassandra [35] and Hypertable [36].

5. TOOLS AND TECHNOLOGIES

In this section we discuss and analyse various tools and technologies according to the lifecycle of Big Data. We therefore first describe the lifecycle in order to clearly define the different phases involved in processing of Big Data irrespective of the use of a specific processing paradigm (presented in previous sections). Following the different phases of the lifecycle we then discuss and classify the tools and technologies related to each processing paradigm.

5.1. Big Data Lifecycle

Generally, in data processing environment, it is always required to acquire data, store data temporarily/permanently, analyze data and produce outputs/results. As shown in Figure 7, Big Data processing can generally be carried out in the following four phases: *data acquisition, data storage, data analysis, and data exploitation*. It is to be noted that the life cycle model does not strictly follow the sequence in which the phases appear. That is, some phases may have backward link to the previous phase. For instance, data storage phase can have backward link to Data Acquisition phase. This is possible in streaming data where data is continuously acquired and stored.

Data acquisition: In this phase Big Data is acquired from various sources. This is the first phase in all the main data processing paradigms (described above). Data can be acquired from batch data sources or from real time (streaming) data sources.
Data storage: This phase concern the storage of Big Data. Most of the data acquired needs to be stored somewhere (disk or diskless) for further processing or analysis.
Data analysis: This phase involves various models, techniques and algorithms which are used to process and analyse Big Data for various applications, for example, forecasting business trends, analysis of sales, analysing traffic data or weather related information.
Data exploitation (Results): This phase concerns the exploitation or results of Big Data. In other words, it involves the outcomes or observations of the analysis carried out in the preceding phases.

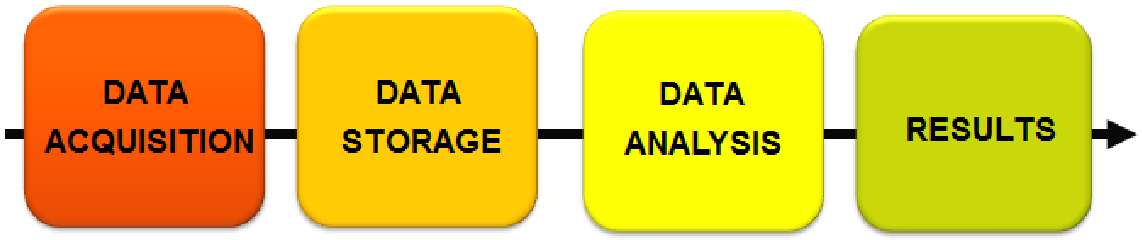


Figure 7. Big Data Phases

In the following, we classify the technologies of each processing paradigm according to the different phases of the Big Data lifecycle. This paper focus on the first three phases of the lifecycle since the data exploitation (results) phase is context-dependent and is not covered here.

5.2. BATCH PROCESSING: Tools and Technologies

This section reviews the following tools and technologies which we believe are closely related to the batch processing mode. However, it is possible that some of these tools and technologies may overlap with other processing models.

a) Data acquisition

Hadoop Distributed File System (HDFS): The original idea of creating Hadoop was to process large volume of unstructured data. But due to the ubiquity and amount of structured data stored in relational databases, Hadoop was also used alongside such databases. Thus data has to be ported from relational databases to Hadoop in order for it to be processed (and analysed) by the analytics tools. For example, HDFS [9] commands such as "hadoop dfs -copyFromLocal <path-to-local> <path-to-remote>" can be used to import data from local data stores to the cluster. The process of such data acquisition is easy and useful but it requires the availability of Hadoop (local) cluster.

Sqoop [37]: Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases [32]. It can be used to import data from external structured datastores into the Hadoop Distributed File System (HDFS) or related systems such as Hive and HBase. Conversely, Sqoop can be used to extract data from Hadoop and export it to external structured datastores such as relational databases and enterprise data warehouses. Unlike the HDFS commands, Sqoop allows for getting information from relational databases.

Flume [17]: Flume is a very useful tool that can be used for different tasks. Here, we perceive Flume as a data acquisition tool. It is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS). It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms for failover and recovery. The goal of flume is to collect data and move them to an appropriate storage system. The main different between Flume and the previous tools is that Flume can be configured to get streaming data and continuously move the information to HDFS. The architecture is based on three main elements:

Sources: are the elements which receive the new data, for example, from web server.

Sink: is the element which sends information to a data store system.

Channel: temporally stores information coming from a source until it is consumed by the Sink.

Scribe [38]: Scribe is a server for aggregating log data streamed in real time from a large number of servers. This is considered as another kind of data acquisition tool. At the higher level of abstraction, it is similar to Flume in terms of importing streaming data into the HDFS. Scribe is a collection of processes, running on different machines and listening to a specified port, in which one can push data in terms of messages and categories. For each category, users can define the way they wish to work with messages. There is a central Scribe server that receives messages from the Scribe nodes and writes messages to the final destination.

b) Data storage

HDFS [8]: HDFS is the file system of Hadoop. It defines an master-slave architecture. The data is stored in the Datanodes, and these Datanotes are managed by a central NameNode. To achieve better performance and reliability, the information is split into blocks of the same size and these blocks are replicated in different DataNodes.

HBase [27]: It is a NoSQL database which sits on top of HDFS based on the Google's BigTable [21]. The main difference between HDFS and HBase is that the latter allows random read and write access to the data, which is useful for accessing/modifying Big Data. But HBase does not provide the facility of storing data in a classical database format. Thus it cannot directly manipulate SQL.

c) Data analysis

MapReduce [7]**:** In order to achieve scalable solutions it is necessary to use parallel processing software. MapReduce provides developers with the power to write parallel distributed programs using Map and Reduce functions. The distributed processing behind this framework is transparent to users. Map function defines how the input data is split into key, value smaller problems. Reduce function generates the results for each key. With this simple model one can develop different design patterns, including joins, sorting, filtering and different type of functions such as average or top-k elements.

Hive [15]: MapReduce is a very powerful framework but sometimes is not easy to achieve desired outcome in terms of map and reduce functions. To deal with this problem, different technologies have been developed which provide abstraction layers on top of MapReduce. One such technology is Hive, which is developed by Facebook. The main feature is that it allows SQL users to work with Big Data. Hive can also be integrated with existing business intelligent tools that use SQL. Furthermore, Hive allows developers to include new functions. On the other hand, Hive has to translate the SQL code into MapReduce jobs so it is less efficient than native MapReduce jobs. In addition, SQL is not always the best approach to deal with data flow processing, for example, to develop machine learning algorithms. Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs.

Pig [14]: Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs. Pig's infrastructure layer consists of a compiler that produces sequences of MapReduce programs to be executed on Hadoop. While Hive uses a declarative language, Pig uses procedural language. Pig is powerful, flexible and is also extensible. Pig source code is easy to understand and easy to maintain. But it has similar problems like Hive. Some algorithms are not easy to develop and Pig code is less efficient than native MapReduce jobs. Furthermore, Pig is less efficient than Hive apart from some joins operations.

Cascading [39]: Cascading is a Java application framework that enables typical developers to quickly and easily develop rich Data Analytics and Data Management applications that can be deployed and managed across a variety of computing environments. Cascading works seamlessly with Apache Hadoop and API compatible distributions. It provides a more powerful abstraction layer on top of MapReduce for defining complex data workflows. It is developed and maintained by Concurrent but it is also an open source. The approach is similar to the one used by Pig, but its Cascading allows developers to define more complex workflows. On the other hand, it is more difficult to develop software applications using Cascading which is based on Hive or Pig.

Spark [40]**:** Apache Spark is a fast and general engine for large-scale data processing [41]. It is neither an abstract layer on top of MapReduce nor a modified version of Hadoop. It is a different processing engine that uses in-memory approach which makes it faster than Hadoop jobs. Spark is very useful for iterative algorithms. In Hadoop based technologies, once a job is finished, the information is written into the HDFS. In a workflow, the next job has to read again the information from disk. Such input/output tasks result in excessive delay. Spark avoids such delay because the information is loaded into memory. Spark is considered to be faster than Hadoop.

Shark (Spark SQL) [42]**:** Shark is a large-scale data warehouse system for Spark designed to be compatible with Apache Hive. The programing paradigm used by Spark is quite similar to MapReduce. Again it provides an abstraction layer on top of Spark. Shark basically follows the same idea as that of Hive, that is, to use SQL on top of Spark.

5.3. REAL TIME PROCESSING: Tools and Technologies

This section reviews the following tools and technologies which we believe are closely related to the real time processing mode. Again, it is possible that some of these tools and technologies may overlap with other processing models, e.g. Flume, etc.

a) Data acquisition

Flume [17]: For data acquisition in real-time, we can use Flume, in the same way we used it for batch processing. In this case, the destination of the information is not the HDFS but a temporal storage system like Kafka [18] .

b) Data storage

Kafka [18]**:** Apache Kafka is publish-subscribe messaging framework which is considered as a distributed commit log. It is basically a distributed producer-consumer architecture where the information is classified by topics. Kafka aims to unify offline and online processing by providing a mechanism for parallel load into Hadoop as well as the ability to partition real-time consumption over a cluster of machines.

Kestrel [43]: Kestrel is a simple, distributed message queuing system written on the Java Virtual Machine. Each server handles a set of reliable, ordered message queues, with no cross communication. This results in a cluster of k-ordered ("loosely ordered") queues. Kestrel is considered to be fast, smaller in size, and reliable.

c) Data analysis

Flume [17]**:** As discussed above, Flume is a technology which is used to import the data. But it is also used for data analysis. Flume allows developers to do simple analysis of the data thanks to its interceptors. An interceptor is an agent with some application logic that basically modifies or filters the information based on some criteria including time, source, content.

Storm [16]**:** Flume is considered to be not so powerful. In order to carry out rigorous real-time analysis Storm (Hadoop for real-time) has been developed. Storm is distributed real time computation system developed by Nathan Marz and is released as open source by Twitter. The architecture is simple. There are two types of elements: *Spouts* that reads information from the source and emits the data as K-V tuples. *Bolts* that processes information coming from the spouts or other bolts. By connecting bolts, Storm defines topologies that are similar to Jobs in MapReduce. The main difference is that a topology never ends, because the data to process is in streaming. Thus there is always new information to process.

Trident [16]**:** Trident is an abstraction layer on top of Storm which makes it easier to develop streaming processing software. Trident is included in the last releases of Storm. It has some similarity with Hive or Pig (for MapReduce). But Trident is used for Storm. It is quite powerful and easy to use but the set of built-in functions is limited.

S4 [9]: S4 is a general-purpose, distributed, scalable, fault-tolerant, pluggable platform that allows programmers to easily develop applications for processing continuous unbounded streams of data. S4 is, therefore, another technology for real-time processing. It was developed by Yahoo!. From a high level view, it is similar to Storm because both are inspired by the MapReduce framework. But there are differences between their programming models and recovery mechanisms.

Spark Streaming [40]: Similar to Spark for batch processing, there is a Shark technology for real-time processing, which is called Spark streaming. It defines the real-time processing similar to a sequence of very short batch jobs. Spark Streaming is claimed to be faster than Storm and S4 and it achieves sub-second latency. But the beauty of Spark is that it is the first approach to use the same programming model for both batch and real-time processing. On the other hand, Spark Streaming has the same problem as that of Spark and Shark. It is a relatively new technology and has not been tested in extremely large cluster.

5.4. HYBRID COMPUTATION: Tools and Technologies

As described in previous sections, hybrid computation model requires the usage of different layers. The technologies implementing this model are therefore more complex compared to batch or real time processing model. Though current technologies developed for hybrid model implicitly follow the main phases of the Big Data lifecycle but due to the nature of hybrid processing this paper avoids the separation between such phases.

Lambdoop [44]**:** Lambdoop is an abstraction layer over many open source technologies needed to build a lambda architecture. The goal of Lambdoop is to make easier the development of Big Data applications [45]. Lambdoop provides the same single programming model for all processing paradigm. It is not a MapReduce-like model. It is similar to Cascading and Pig for batch processing. It is easier to use than using Triden for Real-time processing. Lambdoop implements a whole hybrid computation model with the same programming model. Lambdoop represents the information as data objects independently of their nature: streaming or static. Developers define what operations they want to apply to the data, and their sequence of operation in a workflow. It uses intelligent agents which can process data according to the processing mode (batch workflow, streaming workflow or a hybrid workflow) being used. The main feature of Lambdoop is that it provides the same programming model for all processing paradigms. In addition, it also allows for a friendly and easier way to develop applications. Lambdoop is still an ongoing project and has not been open sourced at the time of writing this paper.

SummingBird [46]**:** SummingBird is being developed by Twitter and its earlier version has been open sourced recently. SummingBird is a library to write generic MapReduce jobs than can be executed in batch-processing using the Hadoop, in real-time using Storm or even in hybrid computation model using both platforms. The main elements of SummingBird are: (i) Hadoop is the batch layer, (ii) Storm is the Real-time layer, and (iii) there is a merge layer to combine the results. The main feature of SummingBird is that it provides the same programming model for all processing paradigm including the hybrid computation model.

6. SUMMARY AND RECOMMENDATIONS

One of main conclusions drawn from this review is that currently Big Data is a complex ecosystem that involves data of various characteristics, 3Vs, different processing modes, tools and technologies. Batch processing, powered by Hadoop, is the most common processing paradigm. In terms of technologies, Hadoop is the winner of the first generation although the Spark ecosystem provides interesting features for some kind of applications. Currently we are in the middle of the second generation, the real-time processing, where Storm seems to be the most promising technology. Google's MillWheel is another interesting and useful technology with good potentials. Similarly, the evolution of new technologies such as Summingbird and Lambdoop, is believed to be very beneficial for the hybrid computation model. Table 1 summarizes the key publications and technologies of Big Data.

Table 1. Big Data key publications and technologies

Dimension	**Key publications**	**Key technologies**
Batch processing	GFL [6]	Hadoop [8]

	MapReduce [7]	Sqoop [37] Pig [14] Hive [15] Cascading [39] Spark (and Shark) [40]
Real-time processing	S4 [47] MillWheel [11]	Flume [17] Kafka [18] S4 [9] Storm (and Trident) [16] Samza [10] Spark Streaming [40]
Hybrid computation	Lambda Architecture [12]	Lambdoop [44] Summingbird [46]
NoSQL systems	DynamoDB [20] BigTable [21] Pregel [33] Dremel [48]	HBase [27] Redis [28] MongoDB [29] Neo4j [31] Giraph [32] Cassandra [35] Hypertable [36] Drill [23]

It is evidenced from the literature[49][50][51][52][53][54][55] that Big Data is significantly important to various application domains. All kinds of private and public organizations are increasingly aware of the potential benefits of Big Data as an enabler to exploit their (potentially vast) data for different purposes. The IT industry has reacted by investing huge efforts in Big Data tools and technologies. However, current tools and technologies suffer from various limitations. From a technological point of view, the future of Big Data needs new solutions that provide new ways for data acquisition, storage, analysis and exploitation:

- New solutions that enable the combined analysis of both structured and unstructured, i.e., to be able to combine multiple data sources (from social media to data warehouses) in a way that is manageable, not only for experts or professionals, but also for non-professional users and groups.
- New tools and technologies for intelligent analysis that exploit streams of data in real time under strict resource constrains of computing capacity, storage, energy and communication bandwidth.
- New paradigms that super-seed the 'pure batch' and 'pure real-time' approach of present Big Data.
- New application frameworks able to squeeze all distributed computing resources, allowing to run different types of tasks (batch, stream analysis, hybrid computation) virtualizing all the underlying infrastructure and scheduling usage depending on the task requirements.
- New database systems able to handle huge datasets while keeping the transactional requirements of data operations available in traditional relational databases.
- New Big Data tools that guide and manage ethical and privacy issues in Big Data.

REFERENCES

[1] A. G. W. Paper, "'Big Data:' Big Challenge, Big Opportunity," pp. 1–6.

[2] I. O'Reilly Media, *Big Data Now: 2012 Edition*. O'Reilly Media, 2012.

[3] B. B. Taube, S. G. Solutions, and V. Corporation, "Leveraging big data and real-time analytics to achieve situational awareness for smart grids," pp. 1–20.

[4] A. Jacobs, "The pathologies of big data," *Commun. ACM*, vol. 52, no. 8, p. 36, 2009.

[5] R. Casado, "The three generations of Big Data processing," in *Big Data Spain*, 2013.

[6] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, p. 29, 2003.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Communications of the ACM*, 2008, vol. 51, no. 1, pp. 137–150.

[8] Apache, "Apache Hadoop," 2008. [Online]. Available: http://hadoop.apache.org/.

[9] Yahoo!, "S4: Distributed Stream Computing Platform," 2010. [Online]. Available: http://incubator.apache.org/s4/. [Accessed: 01-Feb-2014].

[10] Linkedin, "Samza," 2013. [Online]. Available: http://samza.incubator.apache.org/. [Accessed: 01-Feb-2014].

[11] T. Akidau, A. Balikov, K. Bekiroğlu, S. Chernyak, J. Haberman, R. Lax, S. McVeety, D. Mills, P. Nordstrom, and S. Whittle, "MillWheel: fault-tolerant stream processing at internet scale," *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1033–1044, Aug. 2013.

[12] N. Marz and J. Warren, *Big Data Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2014, p. 425.

[13] T. White, “Hadoop: The Definitive Guide,” Jun. 2009.

[14] Yah, “Apache Pig,” 2008. [Online]. Available: http://pig.apache.org/. [Accessed: 11-Jun-2013].

[15] Facebook, “Apache Hive,” 2009. [Online]. Available: http://hive.apache.org/. [Accessed: 11-Jun-2013].

[16] “Storm, distributed and fault-tolerant realtime computation.” [Online]. Available: http://storm-project.net/. [Accessed: 10-Jun-2013].

[17] Cloudera, “Apache Flume,” 2011. [Online]. Available: http://flume.apache.org/. [Accessed: 10-Jun-2013].

[18] Linkedin, “Apache Kafka, a high-throughtput distributed messaging system,” 2011. [Online]. Available: https://kafka.apache.org/. [Accessed: 01-Feb-2014].

[19] S. I. Corporation, “Survey on NoSQL Database,” *Pervasive Comput.*, no. 61072060, pp. 363–366, 2011.

[20] 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,” pp. 205–220, 2007.

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable,” *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26, Jun. 2008.

[22] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, “Dremel: interactive analysis of web-scale datasets,” *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 330–339, Sep. 2010.

[23] “Apache Drill.” [Online]. Available: https://incubator.apache.org/drill/. [Accessed: 01-Feb-2014].

[24] R. Cattell, “Scalable SQL and NoSQL Data Stores,” *ACM SIGMOD Rec.*, vol. 39, no. 4, p. 12, 2011.

[25] D. Pritchett, “BASE: AN ACID ALTERNATIVE,” *Queue*, vol. 6, no. 3, pp. 48–55, May 2008.

[26] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, “A Survey and Comparison of Relational and Non-Relational Database,” vol. 1, no. 6, pp. 1–5, 2012.

[27] “Apache HBase.” [Online]. Available: http://hbase.apache.org/. [Accessed: 10-Jun-2013].

[28] S. Sanfilippo, “Redis,” 2009. [Online]. Available: http://redis.io/. [Accessed: 01-Feb-2014].

[29] K. C. & M. Dirolf, *MongoDB: The Definitive Guide*, vol. 203. 2011, p. NP.

[30] D. Katz, “Apache CouchDB,” 2005. [Online]. Available: http://couchdb.apache.org/. [Accessed: 01-Feb-2014].

[31] Neo Technology, “Neo4j - The World’s Leading Graph Database,” 2010. [Online]. Available: http://www.neo4j.org/. [Accessed: 01-Feb-2014].

[32] Yahoo!, “Apache Giraph,” 2011. [Online]. Available: http://giraph.apache.org/. [Accessed: 01-Feb-2014].

[33] G. Malewicz, M. H. Austern, A. J. . Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, “Pregel,” in *Proceedings of the 2010 international conference on Management of data - SIGMOD ’10*, 2010, p. 135.

[34] D. J. Abadi, P. A. Boncz, and S. Harizopoulos, “Column-oriented database systems,” *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1664–1665, Aug. 2009.

[35] A. Lakshman and P. Malik, “The Apache Cassandra Project,” 2011. [Online]. Available: http://cassandra.apache.org/. [Accessed: 01-Feb-2014].

[36] Zvents, “Hypertable - Big Data. Big Performance,” 2008. [Online]. Available: http://hypertable.org/. [Accessed: 01-Feb-2014].

[37] Cloudera, “Apache Sqoop,” 2009. [Online]. Available: http://sqoop.apache.org/.

[38] Facebook, “Scribe,” 2008. [Online]. Available: https://github.com/facebook/scribe.

[39] C. Wensel, “Cascading | Application Platform for Enterprise Big Data,” 2008. [Online]. Available: http://www.cascading.org/. [Accessed: 01-Feb-2014].

[40] AMPLab, “Apache Spark - Lightning-Fast Cluster Computing,” 2012. [Online]. Available: http://spark.incubator.apache.org/. [Accessed: 01-Feb-2014].

[41] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing,” p. 2, Apr. 2012.

[42] AMPLab, “Shark,” 2013. [Online]. Available: https://github.com/amplab/shark/wiki.

[43] R. Pointer, “Kestrel,” 2008. [Online]. Available: https://github.com/twitter/kestrel.

[44] Treelogic, “Lambdoop,” 2013. [Online]. Available: http://www.lambdoop.com/.

[45] R. Casado, “Lambdoop, a framework for easy development of Big Data applications,” in *NoSQL Matters Barcelona*, 2013.

[46] Twitter, “Summingbird,” 2013. [Online]. Available: http://www.infoq.com/news/2014/01/twitter-summingbird.

[47] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, “S4: Distributed Stream Computing Platform,” in *2010 IEEE International Conference on Data Mining Workshops*, 2010, pp. 170–177.

[48] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, “Dremel,” *Commun. ACM*, vol. 54, no. 6, p. 114, Jun. 2011.

[49] S. P. Artur Krukowski (Intracom), Yiannis Kompatsiaris, L. R. M. Spiros Nikolopoulos (ITI), Francois Hanat, Nadia Echchihab (Cap Digital), P. D. K. M. (TU D. U. (Gradiant), Ruben Casado (Treelogic), R. G. (University C. Dimitrios Gunopoulos (University Athens), Carlos Bento, P. B. Joachim Köhler (Fraunhofer/IAIS), Dev Audsin, Yves Raimond, AndyBower (BBC), H. H. (Holken Brandtzæg (Sintef), Eric van Tol, Janienke Sturm (Fontys Univer sity), J. A. Consultants), Richard Jacobs (BT), Jean-Dominique Meunier (Technicolor), and (Telecom Italia), “Big and Open data Position Paper,” 2013.

[50] danah boyd and K. Crawford, “CRITICAL QUESTIONS FOR BIG DATA,” *Information, Commun. Soc.*, vol. 15, pp. 662–679, 2012.

[51] D. Agrawal, S. Das, and A. El Abbadi, “Big Data and Cloud Computing: Current State and Future Opportunities,” pp. 0–3, 2011.

[52] D. Version, “Critical Questions for Big Data: Provocations for a Cultural, Technological, and Scholarly Phenomenon,” no. 2012, pp. 662–679, 2005.

[53] D. Bollier, *The Promise and Peril of Big Data*. Aspen Institute, Communications and Society Program, 2010, pp. 1–66.

[54] M. Zhou, U. States, J. Grimmer, G. King, and Q. S. Science, “The Age of Big Data,” 2012.

[55] D. Boyd and K. Crawford, “Six Provocations for Big Data,” *Computer (Long. Beach. Calif).*, vol. 123, no. 1, pp. 1–17, 2011.