Big Data Analysis on Clouds

Loris Belcastro, Fabrizio Marozzo, Domenico Talia, Paolo Trunfio

DIMES, University of Calabria, Rende, Italy
{lbelcastro,fmarozzo,talia,trunfio}@dimes.unical.it

Abstract. The huge amount of data generated, the speed at which it is produced, and its heterogeneity in terms of format, represent a challenge to the current storage, process and analysis capabilities. Those data volumes, commonly referred as Big Data, can be exploited to extract useful information and to produce helpful knowledge for science, industry, public services and in general for humankind. Big Data analytics refer to advanced mining techniques applied to Big Data sets. In general, the process of knowledge discovery from Big Data is not so easy, mainly due to data characteristics, as size, complexity and variety, that require to address several issues. Cloud computing is a valid and cost-effective solution for supporting Big Data storage and for executing sophisticated data mining applications. Big Data analytics is a continuously growing field, so novel and efficient solutions (i.e., in terms of platforms, programming tools, frameworks, and data mining algorithms) spring up everyday to cope with the growing scope of interest in Big Data. This chapter discusses models, technologies and research trends in Big Data analysis on Clouds. In particular, the chapter presents representative examples of Cloud environments that can be used to implement applications and frameworks for data analysis, and an overview of the leading software tools and technologies that are used for developing scalable data analysis on Clouds.

Keywords: Cloud computing; Big Data; Data Analytics; Data Mining

1 Introduction

In the last years the ability to produce and gather data has increased exponentially. In fact, in the Internet of Things' era, huge amounts of digital data are generated by and collected from several sources, such as sensors, cams, in-vehicle infotainment, smart meters, mobile devices, web applications and services. The huge amount of data generated, the speed at which it is produced, and its heterogeneity in terms of format (e.g., video, text, xml, email), represent a challenge to the current storage, process and analysis capabilities. In particular, thanks to the growth of social networks (e.g., Facebook, Twitter, Pinterest, Instagram, Foursquare, etc.), the widespread diffusion of mobile phones, and the large use of location-based services, every day millions of people access social network services and share information about their interests and activities. Those data

volumes, commonly referred as Big Data, can be exploited to extract useful information and to produce helpful knowledge for science, industry, public services and in general for humankind.

Although nowadays the term Big Data is often misused, it is very important in computer science for understanding business and human activities. As defined by Gartner[1]: "*Big Data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery, and process optimization.*" Thus, Big Data is not only characterized by the large size of data sets, but also by the complexity, by the variety, and by the velocity of data that can be collected and processed. In fact, we can collect huge amounts of digital data from sources, at a very high rate that the volume of data is overwhelming our ability to make use of it. This situation is commonly called "data deluge".

In science and business, people are analyzing data to extract information and knowledge useful for making new discoveries or for supporting decision processes. This can be done by exploiting Big Data analytics techniques and tools. As an example, one of the leading trends today is the analysis of big geotagged data for creating spatio-temporal sequences or trajectories tracing user movements. Such kind of information is clearly highly valuable for science and business: tourism agencies and municipalities can know the most visited places by tourists, the time of year when such places are visited, and other useful information [4][23]; transport operators can know the places and routes where is it more likely to serve passengers[58] or crowed areas where more transportation resources need to be allocated[57]; city managers may exploit social media analysis to reveal mobility insights in cities such as incident locations[24], or to study and prevent crime events [26][16].

But it must be also considered that just Twitter and Facebook produce about 20 TB of data every day. According to a study conducted by the International Data Corporation (IDC), the whole world produced about 165 exabytes (1 exabytes is equal to 10^{18} bytes) of data in 2007, 800 exabytes in 2009, and it is estimated that in 2020 the global amount of data produced will reach the 35 zettabytes (1 zettabytes is equal to 10^{21} bytes). Then to extract value from such kind of data, novel technologies and architectures have been developed by data scientists for capturing and analyzing complex and/or high velocity data. In this scenario data mining raised in the last decades as a research and technology field that provides several different techniques and algorithms for the automatic analysis of large data sets. The usage of sequential data mining algorithms for analyzing large volumes of data requires a very long time for extracting useful models and patterns. For this reason, high performance computers, such as many and multi-core systems, Clouds, and multi-clusters, paired with parallel and distributed algorithms are commonly used by data analysts to tackle Big Data issues and to reduce response time to a reasonable value.

Big Data analytics refer to advanced mining techniques applied to Big Data sets. In general, the process of knowledge discovery from Big Data is not so

[1] http://www.gartner.com/it-glossary/big-data

easy, mainly due to data characteristics, as size, complexity and variety, that require to address several issues. To overcame these problems and to get valuable information and knowledge in shorter time, high performance and scalable computing systems are used in combination with data and knowledge discovery techniques. In this context, Cloud computing is a valid and cost-effective solution for supporting Big Data storage and for executing sophisticated data analytic applications. In fact, thanks to elastic resource allocation and high computing power, Cloud computing represents a compelling solution for Big Data analytics, allowing faster data analysis, that means more timely results and then greater data value.

Actually, despite the Cloud is an affordable solution for many users, the number of analytics data solutions available is very limited. Most available solutions today are based on open source frameworks, such as Hadoop and Spark, but there are also some proprietary solutions, such as those proposed by IBM, EMC or Kognitio. Big Data analytics is a continuously growing field, so novel and efficient solutions (i.e., in terms of platforms, programming tools, frameworks, and data mining algorithms) spring up everyday to cope with the growing scope of interest in Big Data.

The remainder of the chapter is organized as follows. Section 2 introduces the main Cloud computing concepts. Section 3 describes representative examples of Cloud environments that can be used to implement applications and frameworks for data analysis in the Cloud. Section 4 provides an overview of the leading software tools and technologies used for developing scalable data analysis on Clouds. Section 5 discusses some research trends and open challenges on Big Data analysis. Finally, Section 6 concludes the chapter.

2 Introducing Cloud computing

This section introduces the basic concepts of Cloud computing, which provides scalable storage and processing services that can be used for extracting knowledge from Big Data repositories. In the following we provide basic Cloud computing definitions (Section 2.1) and discuss the main service distribution and deployment models provided by Cloud vendors (Section 2.2).

2.1 Basic concepts

In the last years, Clouds have emerged as effective computing platforms to face the challenge of extracting knowledge from Big Data repositories in limited time, as well as to provide effective and efficient data analysis environments to both researchers and companies. From a client perspective, the Cloud is an abstraction for remote, infinitely scalable provisioning of computation and storage resources. From an implementation point of view, Cloud systems are based on large sets of computing resources, located somewhere "in the Cloud", which are allocated to applications on demand [2]. Thus, Cloud computing can be defined as a distributed computing paradigm in which all the resources, dynamically scalable

and often virtualized, are provided as services over the Internet. As defined by NIST (National Institute of Standards and Technology) [37] Cloud computing can be described as: "A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". From the NIST definition, we can identify five essential characteristics of Cloud computing systems, which are on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Cloud systems can be classified on the basis of their service model and their deployment model.

2.2 Cloud service distribution and deployment models

Cloud computing vendors provide their services according to three main distribution models:

- *Software as a Service (SaaS)*, in which software and data are provided through Internet to customers as ready-to-use services. Specifically, software and associated data are hosted by providers, and customers access them without need to use any additional hardware or software. Examples of SaaS services are Gmail, Facebook, Twitter, Microsoft Office 365.
- *Platform as a Service (PaaS)*, in an environment including databases, application servers, development environment for building, testing and running custom applications. Developers can just focus on deploying of applications since Cloud providers are in charge of maintenance and optimization of the environment and underlying infrastructure. Examples of PaaS services areWindows Azure, Force.com, Google App Engine.
- *Infrastructure as a Service (IaaS)*, that is an outsourcing model under which customers rent resources like CPUs, disks, or more complex resources like virtualized servers or operating systems to support their operations (e.g., Amazon EC2, RackSpace Cloud). Compared to the PaaS approach, the IaaS model has a higher system administration costs for the user; on the other hand, IaaS allows a full customization of the execution environment.

The most common models for providing Big Data analytics solution on Clouds are PaaS and SaaS. IaaS is usually not used for high-level data analytics applications but mainly to handle the storage and computing needs of data analysis processes. In fact, IaaS is the more expensive delivery model, because it requires a greater investment of IT resources. On the contrary, PaaS is widely used for Big Data analytics, because it provides data analysts with tools, programming suites, environments, and libraries ready to be built, deployed and run on the Cloud platform. With the PaaS model users do not need to care about configuring and scaling the infrastructure (e.g., a distributed and scalable Hadoop system), because the Cloud vendor will do that for them. Finally, the SaaS model is used to offer complete Big Data analytics applications to end users, so that they can execute analysis on large and/or complex data sets by exploiting Cloud scalability in storing and processing data.

Regarding deployment models, Cloud computing services are delivered according to three main forms:

- *Public Cloud*: it provides services to the general public through the Internet and users have little or no control over the underlying technology infrastructure. Vendors manage their proprietary data centers delivering services built on top of them.
- *Private Cloud*: it provides services deployed over a company intranet or in a private data center. Often, small and medium-sized IT companies prefer this deployment model as it offers advance security and data control solutions that are not available in the public Cloud model.
- *Hybrid Cloud*: it is the composition of two or more (private or public) Clouds that remain different entities but are linked together.

As outlined in [27], users access Cloud computing services using different client devices and interact with Cloud-based services using a Web browser or desktop/mobile app. The business software and users data are executed and stored on servers hosted in Cloud data centers that provide storage and computing resources. Resources include thousands of servers and storage devices connected each other through an intra-Cloud network. The transfer of data between data center and users takes place on wide-area network. Several technologies and standards are used by the different components of the architecture. For example, users can interact with Cloud services through SOAP-based or RESTful Web services [42] and Ajax technologies allow Web interfaces to Cloud services to have look and interactivity equivalent to those of desktop applications. Open Cloud Computing Interface (OCCI)[2] specifies how Cloud providers can deliver their compute, data, and network resources through a standardized interface.

3 Cloud solutions for Big Data

At the beginning of the Big Data phenomenon, only big IT companies, such as Facebook, Yahoo!, Twitter, Amazon, LinkedIn, invested large amounts of resources in the development of proprietary or open source projects to cope with Big Data analysis problems. But today, Big Data analysis becomes highly significant and useful for small and medium-sized businesses. To address this increasing demand a large vendor community started offering highly distributed platforms for Big Data analysis. Among open-source projects, Apache Hadoop is the leading open-source data-processing platform, which was contributed by IT giants such as Facebook and Yahoo.

Since 2008, several companies, such as Cloudera, MapR, and Hortonworks, started offering enterprise platform for Hadoop, with greats efforts to improve Hadoop performances in terms of high-scalable storage and data processing. Instead, IBM and Pivotal started offering its own customized Hadoop distribution. Other big companies decided to provide only additional softwares and support

[2] OCCI Working Group, http://www.occi-wg.org

for Hadoop platform developed by external providers: for example, Microsoft decided to base its offer on Hortonworks platform, while Oracle decided to resell Cloudera platform. However Hadoop is not the only solution for Big Data analytics. Out of the Hadoop box other solutions are emerging. In particular, in-memory analysis has become a widespread trend, so that companies started offering tools and services for faster in-memory analysis, such as SAP, that is considered the leading company with its Hana[3] platform. Other vendors, including HP, Teradata and Actian, developed analytical database tools with in-memory analysis capabilities. Moreover, some vendors, like Microsoft, IBM, Oracle, and SAP, stand out from their peers for offering a complete solution for data analysis, including DBMS systems, software for data integration, stream-processing, business intelligence, in-memory processing, and Hadoop platform.

In addition, many vendors decided to focus whole offer on the Cloud. Among these certainly there are Amazon Web Services (AWS) and 1010data. In particular, AWS provides a wide range of services and products on the Cloud for Big Data analysis, including scalable database systems and solutions for decision support. Other smaller vendors, including Actian, InfiniDB, HP Vertica, Infobright, and Kognitio, focused their big-data offer on database management systems for analytics only. Following the approach in [48], the remainder of the section introduces representative examples of Cloud environments: Microsoft Azure as an example of public PaaS, Amazon Web Services as the most popular public IaaS, OpenNebula and OpenStack as examples of private IaaS. These environments can be used to implement applications and frameworks for data analysis in the Cloud.

3.1 Microsoft Azure

Azure[4] is the Microsoft Cloud proposal. It is environment providing a large set of Cloud services that can be used by developers to create Cloud-oriented applications, or to enhance existing applications with Cloud-based capabilities. The platform provides on-demand compute and storage resources exploiting the computational and storage power of the Microsoft data centers. Azure is designed for supporting high availability and dynamic scaling services that match user needs with a pay-per-use pricing model. The Azure platform can be used to perform the storage of large datasets, execute large volumes of batch computations, and develop SaaS applications targeted towards end-users. Microsoft Azure includes three basic components/services:

- *Compute* is the computational environment to execute Cloud applications. Each application is structured into roles: *Web role*, for Web-based applications; *Worker role*, for batch applications; *Virtuam Machines role*, for virtual-machine images.

[3] https://hana.sap.com
[4] https://azure.microsoft.com

- *Storage* provides scalable storage to manage: binary and text data (*Blobs*), non-relational tables (*Tables*), queues for asynchronous communication between components (*Queues*). In addition, for relational databases, Microsoft provides its own Cloud database services, called Azure SQL Database.
- *Fabric controller* whose aim is to build a network of interconnected nodes from the physical machines of a single data center. The Compute and Storage services are built on top of this component.

Microsoft Azure provides standard interfaces that allow developers to interact with its services. Moreover, developers can use IDEs like Microsoft Visual Studio and Eclipse to easily design and publish Azure applications.

3.2 Amazon Web Services

Amazon offers compute and storage resources of its IT infrastructure to developers in the form of Web services. Amazon Web Services (AWS)[5] is a large set of Cloud services that can be composed by users to build their SaaS applications or integrate traditional software with Cloud capabilities. It is simple to interact with these service since Amazon provides SDKs for the main programming languages and platforms (e.g. Java, .Net, PHP, Android).

AWS compute solution includes *Elastic Compute Cloud* (EC2), for creating and running virtual servers, and *Amazon Elastic MapReduce* for building and executing MapReduce applications. The Amazon storage solution is based on *S3 Storage Service*, with a range of storage classes designed to cope with different use cases (i.e., Standard, Infrequent Access, and Glacier for long term storage archive). A full set of database systems are also proposed: *Relational Database Service (RDS)* for relational tables; *DynamoDB* for non-relational tables; *SimpleDB* for managing small datasets; *ElasticCache* for caching data. Even though Amazon is best known to be the first IaaS provider (based on its EC2 and S3 services), it is now also a PaaS provider, with services like Elastic Beanstalk, that allows users to quickly create, deploy, and manage applications using a large set of AWS services, or Amazon Machine Learning, that provides visualization tools and wizards for easily creating machine learning models.

3.3 OpenNebula

OpenNebula [45] is an open-source framework mainly used to build private and hybrid Clouds. The main component of the OpenNebula architecture is the Core, which creates and controls virtual machines by interconnecting them with a virtual network environment. Moreover, the Core interacts with specific storage, network and virtualization operations through pluggable components called Drivers. In this way, OpenNebula is independent from the underlying infrastructure and offers a uniform management environment. The Core also supports the deployment of Services, which are a set of linked components (e.g., Web

[5] https://aws.amazon.com

server, database) executed on several virtual machines. Another component is the Scheduler, which is responsible for allocating the virtual machines on the physical servers. To this end, the Scheduler interacts with the Core component through appropriate deployment commands.

OpenNebula can implement a hybrid Cloud using specific Cloud Drivers that allow to interact with external Clouds. In this way, the local infrastructure can be supplemented with computing and storage resources from public Clouds. Currently, OpenNebula includes drivers for using resources from Amazon EC2 and Eucalyptus [40], another open source Cloud framework.

3.4 OpenStack

OpenStack[6] is an open source Cloud operating system realesed under the terms od the Apache License 2.0. It allows the management of large pools of processing, storage, and networking resources in a datacenter through a Web-based interface. Most decisions about its development are decided by the community to the point that every six months there is a design summit to gather requirements and define new specifications for the upcoming release. The modular architecture of OpenStack is composed by four main components, as shown in Figure 1.

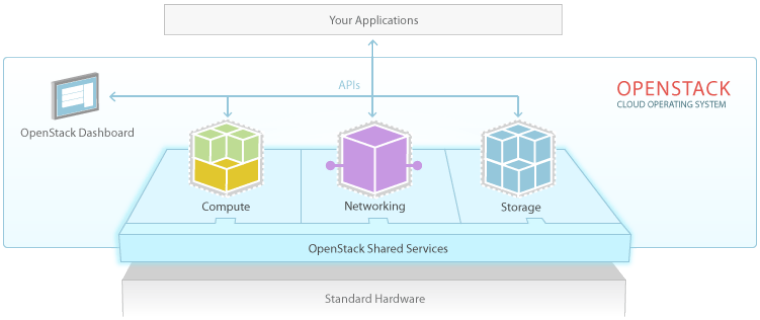

Fig. 1. OpenStack architecture (source: openstack.org).

OpenStack Compute provides virtual servers upon demand by managing the pool of processing resources available in the datacenter. It supports different virtualization technologies (e.g., VMware, KVM) and is designed to scale horizontally. *OpenStack Storage* provides a scalable and redundant storage system. It supports Object Storage and Block Storage: the former allows storing and retrieving objects and files in the datacenter. *OpenStack Networking* manages the networks and IP addresses. Finally, *OpenStack Shared Services* are additional services provided to ease the use of the datacenter, such as Identity Service for mapping users and services, Image Service for managing server images, and Database Service for relational databases.

[6] https://www.openstack.org/

4 Systems for Big Data Analytics in the Cloud

In this section we describe the most used tools for developing scalable data analysis on Clouds, such as MapReduce, Spark, workflow systems, and NoSQL database management systems. In particular, we discuss some frameworks commonly used to develop scalable applications that analyze big amounts of data, such as Apache Hadoop, the best-known MapReduce implementation, and Spark. We present also some powerful data mining programming tools and strategies designed to be executed in the Cloud for exploiting complex and flexible software models, such as the distributed workflows. Workflows provide a declarative way of specifying the high-level logic of an application, hiding the low-level details. They are also able to integrate existing software modules, datasets, and services in complex compositions that implement discovery processes. In this section we presented several data mining workflow systems, such as the Data Mining Cloud Framework, Microsoft Azure Machine Learning, and ClowdFlows. Moreover, we discuss about NoSQL database technology that recently became popular as an alternative or as a complement to relational databases. In the last years, several NoSQL systems have been proposed for providing more scalability and higher performance than relational databases. We introduce the basic principles of NoSQL, described representative NoSQL systems, and outline interesting data analytics use cases where NoSQL tools are useful. Finally, we present a brief overview of well known visual analytics tools, that help users in analytical reasoning by interactive visual interfaces.

4.1 MapReduce

MapReduce is a programming model developed by Google [11] in 2004 for large-scale data processing to cope efficiently with the challenge of processing enormous amounts of data generated by Internet-based applications.

Since its introduction, MapReduce has proven to be applicable to a wide range of domains, including machine learning and data mining, social data analysis, financial analysis, scientific simulation, image retrieval and processing, blog crawling, machine translation, language modelling, and bioinformatics. Today, MapReduce is widely recognized as one of the most important programming models for Cloud computing environments, being it supported by Google and other leading Cloud providers such as Amazon, with its Elastic MapReduce service[7], and Microsoft, with its HDInsight[8], or on top of private Cloud infrastructures such as OpenStack, with its Sahara service[9].

Hadoop[10] is the most used open source MapReduce implementation for developing parallel applications that analyze big amounts of data. It can be adopted for developing distributed and parallel applications using many programming

[7] http://aws.amazon.com/elasticmapreduce/
[8] http://azure.microsoft.com/services/hdinsight/
[9] http://wiki.openstack.org/wiki/Sahara
[10] http://hadoop.apache.org/

languages (e.g., Java, Ruby, Python, C++). Hadoop relieves developers from having to deal with classical distributed computing issues, such as load balancing, fault tolerance, data locality, and network bandwidth saving.

The Hadoop project is not only about the MapReduce programming model (Hadoop MapReduce module), as it includes other modules such as:

- *Hadoop Distributed File System (HDFS)*: a distributed file system providing fault tolerance with automatic recovery, portability across heterogeneous commodity hardware and operating systems, high-throughput access and data reliability.
- *Hadoop YARN*: a framework for cluster resource management and job scheduling.
- *Hadoop Common*: common utilities that support the other Hadoop modules.

In particular, thanks to the introduction of YARN in 2013, Hadoop turns from a batch processing solution into a platform for running a large variety of data applications, such as streaming, in-memory, and graphs analysis. As a result, Hadoop became a reference for several other frameworks, such as: Giraph for graph analysis; Storm for streaming data analysis; Hive, which is a data warehouse software for querying and managing large datasets; Pig, which is as a dataflow language for exploring large datasets; Tez for executing complex directed-acyclic graph of data processing tasks; Oozie, which is a workflow scheduler system for managing Hadoop jobs. Besides Hadoop and its ecosystem, several other MapReduce implementations have been implemented within other systems, including GridGain, Skynet, MapSharp and Twister [14]. One of the most popular alternative to Hadoop is Disco, which is a lightweight, open-source framework for distributed computing. The Disco core is written in Erlang, a functional language designed for building fault-tolerant distributed applications. Disco has been used for a variety of purposes, such as log analysis, text indexing, probabilistic modeling and data mining.

4.2 Spark

Apache Spark[11] is another Apache framework for Big Data processing. Differently from Hadoop in which intermediate data are always stored in distributed file systems, Spark stores data in RAM memory and queries it repeatedly so as to obtain better performance for some class of applications (e.g., iterative machine learning algorithms) [56]. For many years, Hadoop has been considered the leading open source Big Data framework, but recently Spark has become the more popular so that it is supported by every major Hadoop vendors. In fact, for particular tasks, Spark is up to 100 times faster than Hadoop in memory and 10 times faster on disk. Several other libraries have been built on top of Spark: *Spark SQL* for dealing with SQL and DataFrames, *MLlib* for machine learning, *GraphX* for graphs and graph-parallel computation, and *Spark Streaming* to build scalable fault-tolerant streaming applications.

[11] http://spark.apache.org

For these reasons, Spark is becoming the primary execution engine for data processing and, in general, a must-have for Big Data applications. But even though in some applications Spark can be considered a better alternative to Hadoop, in many other applications it has limitations that make it complementary to Hadoop. The main limitation of Spark is that it does not provide its own distributed and scalable storage system, that is a fundamental requirement for Big Data applications that use huge and continually increasing volume of data stored across a very large number of nodes. To overcome this lack, Spark has been designed to run on top of several data sources, such as Cloud object storage (e.g., Amazon S3 Storage, Swift Object Storage), distributed filesystem (e.g., HDFS), no-SQL databases (e.g., HBase, Apache Cassandra), and others. Today an increasing number of big vendors, such Microsoft Azure or Cloudera, offer Spark as well as Hadoop, so developers can choose the most suitable framework for each data analytic application.

With respect to Hadoop, Spark loads data from data sources and executes most of its tasks in RAM memory. In this way, Spark reduces significantly the time spent in writing and reading from hard drives, so that the execution is far faster than Hadoop. Regarding task recovering in case of failures, Hadoop flushes all of the data back to the storage after each operation. Similarly, Spark allow recovering in case of failures by arranging data in Resilient Distributed Datasets (RDD), which are a immutable and fault-tolerant collections of records which can be stored in the volatile memory or in a persistent storage (e.g., HDFS, HBase). Moreover, Spark's real-time processing capability is increasingly being used by Big Data analysts into applications that requires to extract insights quickly from data, such as recommendation and monitoring systems.

Several big companies and organizations use Spark for Big Data analysis purpose: for example, Ebay uses Spark for log transaction aggregation and analytics, Kelkoo for product recommendations, SK Telecom analyses mobile usage patterns of customers.

4.3 Mahout

Apache Mahout[12] is an open-source framework that provides scalable implementations of machine learning algorithms that are applicable on big input. Originally, the Mahout project provided implementations of machine learning algorithms executable on the top of Apache Hadoop framework. But the comparison of the performance of Mahout algorithms on Hadoop with other machine learning libraries, showed that Hadoop spends the majority of the processing time to load the state from file system at every intermediate step [44].

For these reasons, the latest version of Mahout goes beyond Hadoop and provides several machine learning algorithms for collaborative filtering, classification, and clustering, implemented not only in Hadoop MapReduce, but also in Spark, H2O[13]. Both Apache Spark and H2O process data in memory so they

[12] http://mahout.apache.org/
[13] http://www.h2o.ai

can achieve a significant performance gain when compared to Hadoop framework for specific classes of applications (e.g., interactive jobs, real-time queries, and stream data) [44]. In addition, the latest release of Mahout introduces a new math environment, called Samsara [29], that helps users in creating their own math providing general linear algebra and statistical operations. In the following, some examples for each algorithm's category are listed: analyzing user history and preferences to suggest accurate recommendations (collaborative filtering), selecting whether a new input matches a previously observed pattern or not (classification), and grouping large number of things together into clusters that share some similarity (clustering) [41]. In the future, Mahout will support Apache Flink[14], an open source platform that provides data distribution, communication, and fault tolerance for distributed computations over data streams.

4.4 Hunk

Hunk[15] is a commercial data analysis platform developed by Splunk for rapidly exploring, analyzing and visualizing data in Hadoop and NoSQL data stores. Hunk uses a set of high-level user and programming interfaces to offer speed and simplicity of getting insights from large unstructured and structured data sets. One of the key components of the Hunk architecture is the *Splunk Virtual Index*. This system decouples the storage tier from the data access and analytics tiers, so enabling Hunk to route requests to different data stores. The analytics tier is based on *Splunks Search Processing Language* (SPL) designed for data exploration across large, different data sets. The Hunk web framework allows building applications on top of the Hadoop Distributed File System (HDFS) and/or the NoSQL data store.

Developers can use Hunk to build their Big Data applications on top of data in Hadoop using a set of well known languages and frameworks. Indeed, the framework enables developers to integrate data and functionality from Hunk into enterprise Big Data applications using a web framework, documented REST API and software development kits for CSharp,Java, JavaScript, PHP and Ruby. Also common development languages such as HTML5 and Python can be used by developers.

The Hunk framework can be deployed on on-premises Hadoop clusters or private Clouds and it is available as a preconfigured instance on the Amazon public Cloud using the Amazon Web Services (AWS). This public Cloud solution allows Hunk users to utilize the Hunk facilities and tools from AWS, also exploiting commodity storage on Amazon S3, according to a pay-per-use model. Finally, the framework implements and makes available a set of applications that enable the Hunk analytics platform to explore, explore and visualize data in NoSQL and other data stores, including Apache Accumulo, Apache Cassandra, MongoDB and Neo4j. Hunk is also provided in combination with the Cloudera's enterprise data hub to develop large-scale applications that can access and analyze Big Data sets.

[14] https://flink.apache.org/
[15] http://www.splunk.com/en_us/products/hunk.html

4.5 Sector/Sphere

Sector/Sphere[16] is a Cloud framework designed at the University of Illinois-Chicago to implement data analysis applications involving large, geographically distributed datasets in which the data can be naturally processed in parallel [19]. The framework includes two components: a storage service called *Sector*, which manages the large distributed datasets with high reliability, high performance IO, and a uniform access, and a compute service called *Sphere*, which makes use of the Sector service to simplify data access, increase data IO bandwidth, and exploit wide area high performance networks. Both of them are available as open source software[17]. Sector is a distributed storage system that can be deployed over a wide area network and allows users to ingest and download large datasets from any location with a high-speed network connection to the system. The system can be deployed over a large number of commodity computers (called nodes), located either within a data center or across data centers, which are connected by high-speed networks.

In an example scenario, nodes in the same rack are connected by 1 Gbps networks, two racks in the same data center are connected by 10 Gbps networks, and two different data centers are connected by 10 Gbps networks. Sector assumes that the datasets it stores are divided into one or more separate files, called slices, which are replicated and distributed over the various nodes managed by Sector.

The Sector architecture includes a Security server, a Master server and a number of Slave nodes. The Security server maintains user accounts, file access information, and the list of authorized slave nodes. The Master server maintains the metadata of the files stored in the system, controls the running of the slave nodes, and responds to users' requests. The Slaves nodes store the files managed by the system and process the data upon the request of a Sector client. Sphere is a compute service built on top of Sector and provides a set of programming interfaces to write distributed data analysis applications. Sphere takes streams as inputs and produces streams as outputs. A stream consists of multiple data segments that are processed by Sphere Processing Engines (SPEs) using slave nodes. Usually there are many more segments than SPEs. Each SPE takes a segment from a stream as an input and produces a segment of a stream as output. These output segments can in turn be the input segments of another Sphere process. Developers can use the Sphere client APIs to initialize input streams, upload processing function libraries, start Sphere processes, and read the processing results.

4.6 BigML

BigML[18] is a system provided as a Software-as-a-Service (SaaS) for discovering predictive models from data and it uses data classification and regression algo-

[16] http://sector.sourceforge.net/
[17] http://sector.sourceforge.net
[18] https://bigml.com

rithms. The distinctive feature of BigML is that predictive models are presented to users as interactive decision trees. The decision trees can be dynamically visualized and explored within the BigML interface, downloaded for local usage and/or integration with applications, services, and other data analysis tools. Recently, BigML launched its PaaS solution, called *BigML PredictServer*, which is a dedicated machine image that can be deployed on Amazon AWS.

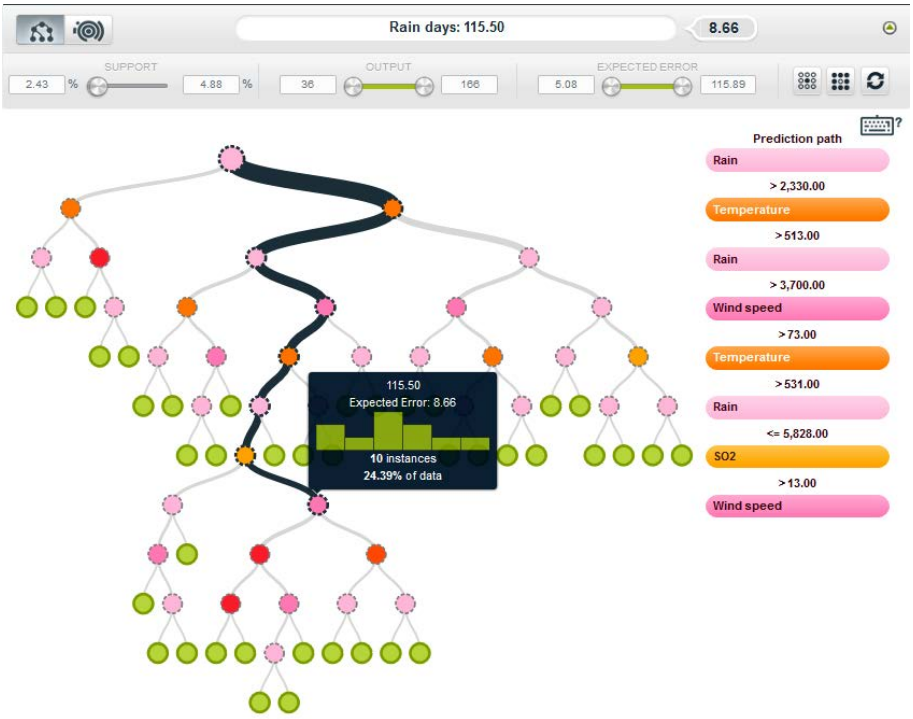

Fig. 2. Example of BigML prediction model for air pollution (source: bigml.com).

Extracting and using predictive models in BigML consists in multiple steps, as detailed as follows:

- *Data source setting and dataset creation.* A data source is the raw data from which a user wants to extract a predictive model. Each data source instance is described by a set of columns, each one representing an instance feature, or field. One of the fields is considered as the feature to be predicted. A dataset is created as a structured version of a data source in which each field has been processed and serialized according to its type (numeric, categorical, etc.).
- *Model extraction and visualization.* Given a dataset, the system generates the number of predictive models specified by the user, who can also choose

the level of parallelism level for the task. The interface provides a visual tree representation of each predictive model, allowing users to adjust the support and confidence values and to observe in real time how these values influence the model.
- *Prediction making.* A model can be used individually, or in a group (the so-called ensemble, composed of multiple models extracted from different parts of a dataset), to make predictions on new data. The system provides interactive forms to submit a predictive query for a new data using the input fields from a model or ensemble. The system provides APIs to automate the generation of predictions, which is particularly useful when the number of input fields is high.
- *Models evaluation.* BigML provides functionalities to evaluate the goodness of the predictive models extracted. This is done by generating performance measures that can be applied to the kind of extracted model (classification or regression).

4.7 Kognitio Analytical Platform

Kognitio Analytical Platform[19], available as Cloud based service or supplied as a pre-integrated appliance, allows users to pull very large amounts of data from existing data storage systems into high-speed computer memory, allowing complex analytical questions to be answered interactively. Although Kognitio has its own internal disk subsystem, it is primarily used as an analytical layer on top of existing storage/data processing systems, e.g., Hadoop clusters and/or existing traditional disk-based data warehouse products, Cloud storage, etc. A feature called *External Tables* allows persistent data to reside on external systems. Using this feature the system administrator, or a privileged user, can easily setup access to data that resides in another environment, typically a disk store such as the above-mentioned Hadoop clusters and data warehouse systems. To a final user, the Kognitio Analytical Platform looks like a relational database management system (RDBMS) similar to many commercial databases. However, unlike these databases, Kognitio has been designed specifically to handle analytical query workload, as opposed to the more traditional on-line transaction processing (OLTP) workload. Key reasons of Kognitios high performance in managing analytical query workload are:

- Data is held in high-speed RAM using structures optimized for in-memory analysis, which is different from a simple copy of disk-based data, like a traditional cache.
- Massively Parallel Processing (MPP) allows scaling out across large arrays of low-cost industry standard servers, up to thousands nodes.
- Query parallelization allows every processor core on every server to be equally involved in every query.
- Machine code generation and advanced query plan optimization techniques ensure every processor cycle is effectively used to its maximum capacity.

[19] www.kognitio.com

Parallelism in Kognitio Analytical Platform fully exploits the so-called shared nothing distributed computing approach, in which none of the nodes share memory or disk storage, and there is no single point of contention across the system.

4.8 Data Analysis Workflows

A workflow consists of a series of activities, events or tasks that must be performed to accomplish a goal and/or obtain a result. For example, a data analysis workflow can be designed as a sequence of pre-processing, analysis, post-processing, and interpretation steps. At a practical level, a workflow can be implemented as a computer program and can be expressed in a programming language or paradigm that allows expressing the basic workflow steps and includes mechanisms to orchestrate them.

Workflows have emerged as an effective paradigm to address the complexity of scientific and business applications. The wide availability of high-performance computing systems, Grids and Clouds, allowed scientists and engineers to implement more and more complex applications to access and process large data repositories and run scientific experiments in silico on distributed computing platforms. Most of these applications are designed as workflows that include data analysis, scientific computation methods and complex simulation techniques. The design and execution of many scientific applications require tools and high-level mechanisms. Simple and complex workflows are often used to reach this goal. For this reason, in the past years, many efforts have been devoted towards the development of distributed workflow management systems for scientific applications. Workflows provide a declarative way of specifying the high-level logic of an application, hiding the low-level details that are not fundamental for application design. They are also able to integrate existing software modules, datasets, and services in complex compositions that implement scientific discovery processes.

Another important benefit of workflows is that, once defined, they can be stored and retrieved for modifications and/or re-execution: this allows users to define typical patterns and reuse them in different scenarios [5]. The definition, creation, and execution of workflows are supported by a so-called Workflow Management System (WMS). A key function of a WMS during the workflow execution (or enactment) is coordinating the operations of the individual activities that constitute the workflow. There are several WMSes on the market, most of them targeted to a specific application domain. In the following we focus on some well-known software tools and frameworks designed implementing data analysis workflows on Clouds systems.

Data Mining Cloud Framework

The Data Mining Cloud Framework (DMCF) [32] is a software system that we developd at University of Calabria for allowing users to design and execute data analysis workflows on Clouds. DMCF supports a large variety of data analysis processes, including single-task applications, parameter sweeping applications,

and workflow-based applications [33]. A Web-based user interface allows users to compose their applications and to submit them for execution to a Cloud platform, according to a Software-as-a-Service approach. Recently, DMCF has been extended to include the execution of MapReduce tasks [3].

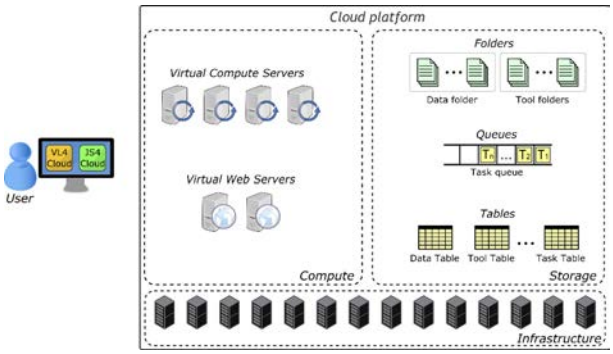

Fig. 3. DMCF architecture.

The DMCFs architecture includes a set of components that can be classified as storage and compute components (see Figure 3). The storage components include:

- A *Data Folder* that contains data sources and the results of knowledge discovery processes. Similarly, a Tool folder contains libraries and executable files for data selection, pre-processing, transformation, data mining, and results evaluation.
- The *Data Table*, *Tool Table* and *Task Table* that contain metadata information associated with data, tools, and tasks.
- The *Task Queue* that manages the tasks to be executed.

The compute components are:

- A pool of *Virtual Compute Servers*, which are in charge of executing the data mining tasks.
- A pool of *Virtual Web Servers* host the Web-based user interface.

The user interface provides three functionalities:

- App submission, which allows users to submit single-task, parameter sweeping, or workflow-based applications;
- App monitoring, which is used to monitor the status and access results of the submitted applications;
- Data/Tool management, which allows users to manage input/output data and tools.

The DMCF architecture has been designed as a reference architecture to be implemented on different Cloud systems. However, a first implementation of the framework has been carried out on the Microsoft Azure Cloud platform and has been evaluated through a set of data analysis applications executed on a Microsoft Cloud data center. The DMCF framework takes advantage of Cloud computing features, such as elasticity of resources provisioning. In DMCF, at least one Virtual Web Server runs continuously in the Cloud, as it serves as user front-end. In addition, users specify the minimum and maximum number of Virtual Compute Servers. DMCF can exploit the auto-scaling features of Microsoft Azure that allows dynamic spinning up or shutting down Virtual Compute Servers, based on the number of tasks ready for execution in the DMCFs Task Queue. Since storage is managed by the Cloud platform, the number of storage servers is transparent to the user.

For designing and executing a knowledge discovery application, users interact with the system performing the following steps:

1. The Website is used to design an application (either single-task, parameter sweeping, or workflow-based) through a Web-based interface that offers both the visual programming interface and the script.
2. When a user submits an application, the system creates a set of tasks and inserts them into the Task Queue on the basis of the application requirements.
3. Each idle Virtual Compute Server picks a task from the Task Queue, and concurrently executes it.
4. Each Virtual Compute Server gets the input dataset from the location specified by the application. To this end, file transfer is performed from the Data Folder where the dataset is located, to the local storage of the Virtual Compute Server.
5. After task completion, each Virtual Compute Server puts the result on the Data Folder.
6. The Website notifies the user as soon as her/his task(s) have completed, and allows her/him to access the results.

The set of tasks created on the second step depends on the type of application submitted by a user. In the case of a single-task application, just one data mining task is inserted into the Task Queue. If users submit a parameter sweeping application, a set of tasks corresponding to the combinations of the input parameters values are executed in parallel. If a workflow-based application has to be executed, the set of tasks created depends on how many data analysis tools are invoked within the workflow. Initially, only the workflow tasks without dependencies are inserted into the Task Queue.

In DMCF workflows may encompass all the steps of discovery based on the execution of complex algorithms and the access and analysis of scientific data. In data-driven discovery processes, knowledge discovery workflows can produce results that can confirm real experiments or provide insights that cannot be achieved in laboratories. In particular, DMCF allows to program workflow applications using two languages:

– VL4Cloud (*Visual Language for Cloud*), a visual programming language that lets users develop applications by programming the workflow components graphically [33].
– JS4Cloud (*JavaScript for Cloud*), a scripting language for programming data analysis workflows based on JavaScript [34].

Both languages use two key programming abstractions:

– Data elements denote input files or storage elements (e.g., a dataset to be analyzed) or output files or stored elements (e.g., a data mining model).
– Tool elements denote algorithms, software tools or complex applications performing any kind of operation that can be applied to a data element (data mining, filtering, partitioning, etc.).

Another common element is the task concept, which represents the unit of parallelism in our model. A task is a *Tool*, invoked in the workflow, which is intended to run in parallel with other tasks on a set of Cloud resources. According to this approach, VL4Cloud and JS4Cloud implement a data-driven task parallelism. This means that, as soon as a task does not depend on any other task in the same workflow, the runtime asynchronously spawns it to the first available virtual machine. A task T_j does not depend on a task T_i belonging to the same workflow (with $i \neq j$), if T_j during its execution does not read any data element created by T_i. In VL4Cloud, workflows are directed acyclic graphs whose nodes represent data and tools elements. The nodes can be connected with each other through direct edges, establishing specific dependency relationships among them. When an edge is being created between two nodes, a label is automatically attached to it representing the type of relationship between the two nodes. Data and Tool nodes can be added to the workflow singularly or in array form. A data array is an ordered collection of input/output data elements, while a tool array represents multiple instances of the same tool. Figure 4 shows an example of data analysis workflow developed using the visual workflow formalism of DMCF [6].

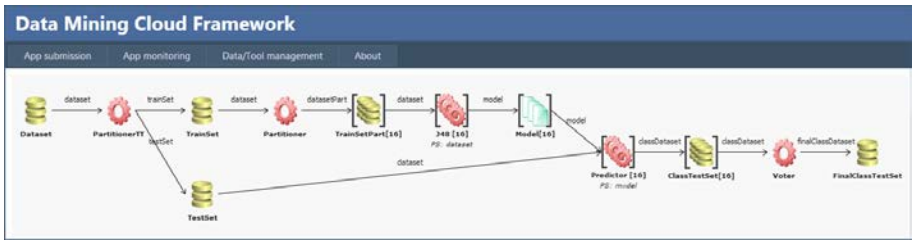

Fig. 4. Example of data analysis application designed using VL4Cloud.

In JS4Cloud, workflows are defined with a JavaScript code that interacts with Data and Tool elements through three functions:

– Data Access, for accessing a Data element stored in the Cloud;
– Data Definition, to define a new Data element that will be created at runtime as a result of a Tool execution;
– Tool Execution: to invoke the execution of a Tool available in the Cloud.

Once the JS4Cloud workflow code has been submitted, an interpreter translates the workflow into a set of concurrent tasks by analysing the existing dependencies in the code. The main benefits of JS4Cloud are:

1. It extends the well-known JavaScript language while using only its basic functions (arrays, functions, loops).
2. It implements both a data-driven task parallelism that automatically spawns ready-to-run tasks to the Cloud resources, and data parallelism through an array-based formalism.
3. These two types of parallelism are exploited implicitly so that workflows can be programmed in a totally sequential way, which frees users from duties like work partitioning, synchronization and communication.

Figure 6 shows the script-based workflow version of the visual workflow shown in Figure 4. In this example, parallelism is exploited in the for loop at line 7, where up to 16 instances of the J48 classifier are executed in parallel on 16 different partitions of the training sets, and in the for loop at line 10, where up to 16 instances of the Predictor tool are executed in parallel to classify the test set using 16 different classification models.

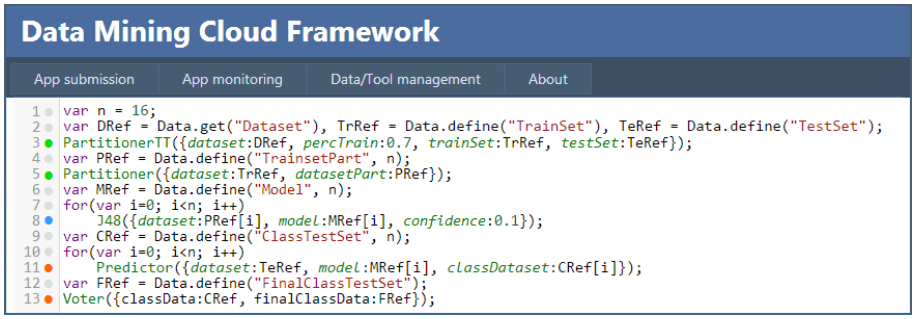

Fig. 5. Example of data analysis application designed using JS4Cloud.

Figure 6 shows a snapshot of the parallel classification workflow taken during its execution in the DMCFs user interface. Beside each code line number, a colored circle indicates the status of execution. This feature allows user to monitor the status of the workflow execution. Green circles at lines 3 and 5 indicate that the two partitioners have completed their execution; the blue circle at line 8 indicates that J48 tasks are still running; the orange circles at lines 11 and 13 indicate that the corresponding tasks are waiting to be executed.

Microsoft Azure Machine Learning

Microsoft Azure Machine Learning (Azure ML) is a SaaS that provides a Web-based machine learning IDE (i.e., integrated development environment) for creation and automation of machine learning workflows. Through its user-friendly interface, data scientists and developers can perform several common data analysis/mining tasks on their data and automate their workflows. Using its drag-and-drop interface, users can import their data in the environment or use special readers to retrieve data form several sources, such as Web URL (HTTP), OData Web service, Azure Blob Storage, Azure SQL Database, Azure Table. After that, users can compose their data analysis workflows where each data processing task is represented as a block that can be connected with each other through direct edges, establishing specific dependency relationships among them. Azure ML includes a rich catalog of processing tools that can be easily included in a workflow to prepare/transform data or to mine data through supervised learning (regression e classification) or unsupervised learning (clustering) algorithms. Optionally, users can include their own custom scripts (e.g., in R or Python) to extend the tools catalog. When workflows are correctly defined, users can evaluate them using some testing dataset.

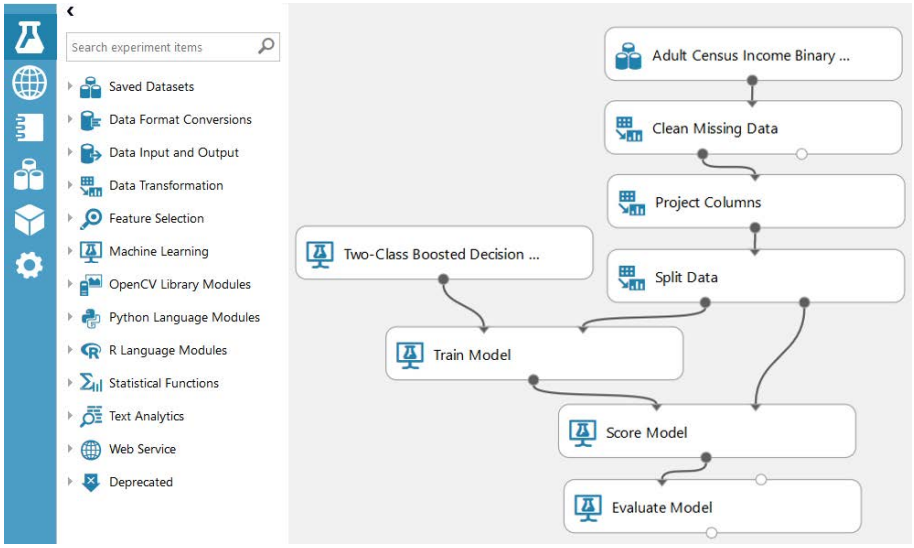

Fig. 6. Example of Azure Machine Learning workflow (source: studio.azureml.net).

Users can easily visualize the results of the tests and find very useful information about models accuracy, precision and recall. Finally, in order to use their models to predict new data or perform real time predictions, users can expose them as Web services. Always through a Web-based interface, users can monitor

the Web services load and use by time. Azure Machine Learning is a fully managed service provided by Microsoft on its Cloud platform; users do not need to buy any hardware/software nor manage virtual machine manually. One of the main advantage of working with a Cloud platform like Azure is its auto-scaling feature: models are deployed as elastic Web services so as users do not have to worry about scaling them if the models usage increased.

ClowdFlows

ClowdFlows [22] is an open source Cloud-based platform for the composition, execution, and sharing of data analysis workflows. It is provided as a software as a service that allows users to design and execute visual workflows through a simple Web browser and so it can be run from most devices (e.g., desktop PCs, laptops, and tablets). ClowdFlows is based on two software components: the workflow editor (provided by a Web browser) and the server side application that manages the execution of the application workflows and hosts a set of stored workflows. The server side consists of methods for supporting the client-side workflow editor in the composition and for executing workflows, and a relational database of workflows and data. The workflow editor includes of a workflow canvas and a widget repository. The widget repository is a list of all available workflow components that can be added to the workflow canvas. The repository includes a set of default widgets.

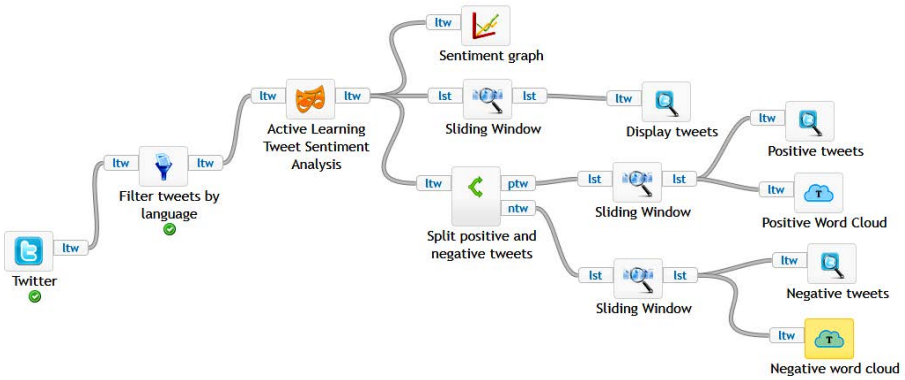

Fig. 7. Example of CloudFlow workflow (source: clowdflows.org).

According to this approach, the CloudFlows service-oriented architecture allows users to include in their workflow the implementations of various algorithms, tools and Web services as workflow elements. For example, the Weka's algorithms have been included and exposed as Web services and so they can be added in a workflow application. ClowdFlows is also easily extensible by importing third-party Web services that wrap open-source or custom data mining algorithms. To

this end, a user has only to insert the WSDL URL of a Web service to create a new workflow element that represents the Web service in a workflow application.

Pegasus

Pegasus [12] is a workflow management system developed at the University of Southern California for supporting the implementation of scientific applications also in the area of data analysis. Pegasus includes a set of software modules to execute workflow-based applications in a number of different environments, including desktops, Clouds, clusters and grids. It has been used in several scientific areas including bioinformatics, astronomy, earthquake science, gravitational wave physics, and ocean science. The Pegasus workflow management system can manage the execution of an application expressed as a visual workflow by mapping it onto available resources and executing the workflow tasks in the order of their dependencies. In particular, significant activities have been recently performed on Pegasus to support the system implementation on Cloud platforms and manage computational workflows in the Cloud for developing data-intensive scientific applications (Juve et al., 2010) (Nagavaram et al, 2011). The Pegasus system has been used with IaaS Clouds for workflow applications and the most recent versions of Pegasus can be used to map and execute workflows on commercial and academic IaaS Clouds such as Amazon EC2, Nimbus, OpenNebula and Eucalyptus (Deelman et al., 2015). The Pegasus system includes four main components:

- the Mapper, which builds an executable workflow based on an abstract workflow provided by a user or generated by the workflow composition system. To this end, this component finds the appropriate software, data, and computational resources required for workflow execution. The Mapper can also restructure the workflow in order to optimize performance, and add transformations for data management or to generate provenance information.
- the Execution Engine (DAGMan), which executes in appropriate order the tasks defined in the workflow. This component relies on the compute, storage and network resources defined in the executable workflow to perform the necessary activities. It includes a local component and some remote ones.
- the Task Manager, which is in charge of managing single workflow tasks by supervising their execution on local and/or remote resources.
- The Monitoring Component, which monitors the workflow execution, analyzes the workflow and job logs and stores them into a workflow database used to collect runtime provenance information. This component sends notifications back to users notifying them of events like failures, success and completion of workflows and jobs.

The Pegasus software architecture includes also an error recovery system that attempts to recover from failures by retrying tasks or an entire workflow, re-mapping portions of the workflow, providing workflow-level checkpointing, and using alternative data sources, when possible. The Pegasus system records

provenance information including the locations of data used and produced, and which software was used with which parameters. This feature is useful when a workflow must be reproduced.

Swift

Swift [53] is a implicitly parallel scripting language that runs workflows across several distributed systems, like clusters, Clouds, grids, and supercomputers. The Swift language has been designed at the University of Chicago and at the Argonne National Lab to provide users with a workflow-based language for grid computing. Recently has been ported on Clouds and exascale systems. Swift separates the application workflow logic from runtime configuration. This approach allows a flexible development model.

As the DMCF programming interface, the Swift language allows invocation and running of external application code and allows binding with application execution environments without extra coding from the user. Swift/K is the previous version of the Swift language that runs on the Karajan grid workflow engine across wide area resources. Swift/T is a new implementation of the Swift language for high-performance computing. In this implementation, a Swift program is translated into an MPI program that uses the Turbine and ADLB runtime libraries for scalable dataflow processing over MPI. The Swift-Turbine Compiler (STC) is an optimizing compiler for Swift/T and the Swift Turbine runtime is a distributed engine that maps the load of Swift workflow tasks across multiple computing nodes. Users can also use Galaxy [17] to provide a visual interface for Swift.

The Swift language provides a functional programming paradigm where workflows are designed as a set of code invocations with their associated command-line arguments and input and output files. Swift is based on a C-like syntax and uses an implicit data-driven task parallelism [54]. In fact, it looks like a sequential language, but being a dataflow language, all variables are futures, thus execution is based on data availability. When input data is ready, functions are executed in parallel. Moreover, parallelism can be exploited through the use of the foreach statement. The Turbine runtime comprises a set of services that implement the parallel execution of Swift scripts exploiting the maximal concurrency permitted by data dependencies within a script and by external resource availability. Swift has been used for developing several scientific data analysis applications, such as prediction of protein structures, modeling the molecular structure of new materials, and decision making in climate and energy policy.

4.9 NoSQL Models for Data Analytics

With the exponential growth of data to be stored in distributed network scenarios, relational databases exhibit scalability limitations that significantly reduce the efficiency of querying and analysis [1]. In fact, most relational databases

have little ability to scale horizontally over many servers, which makes challenging storing and managing the huge amounts of data produced everyday by many applications.

The NoSQL or non-relational database approach became popular in the last years as an alternative or as a complement to relational databases, in order to ensure horizontal scalability of simple read/write database operations distributed over many servers [8]. Compared to relational databases, NoSQL databases are generally more flexible and scalable, as they are capable of taking advantage of new nodes transparently, without requiring manual distribution of information or additional database management [46]. Since database management may be a challenging task with huge amounts of data, NoSQL databases are designed to ensure automatic data distribution and fault tolerance [15]. In the remainder of this section, we describe some representative NoSQL systems, and discuss some use cases for NoSQL databases, with a focus on data analytics.

NoSQL databases provide ways to store scalar values (e.g., numbers, strings), binary objects (e.g., images, videos), or more complex values. According to their data model, NoSQL databases can be grouped into three main categories [8]: Key-value stores, Document stores, Extensible Record stores.

Key-value stores provide mechanisms to store data as (key, value) pairs over multiple servers. In such kind of databases a distributed hash table (DHT) can be used to implement a scalable indexing structure, where data retrieval is performed by using key to find value [8].

Document stores are designed to manage data stored in documents that use different formats (e.g., JSON), where each document is assigned a unique key that is used to identify and retrieve the document. Therefore, document stores extend key-value stores because they provide for storing, retrieving, and managing semi-structured information, rather than single values. Unlike the key-value stores, document stores generally support secondary indexes and multiple types of documents per database, and provide mechanisms to query collections based on multiple attribute value constraints [8].

Finally, Extensible Record stores (also known as Column-oriented data stores) provide mechanisms to store extensible records that can be partitioned across multiple servers. In this type of database, records are said to be extensible because new attributes can be added on a per-record basis. Extensible record stores provide both horizontal partitioning (storing records on different nodes) and vertical partitioning (storing parts of a single record on different servers). In some systems, columns of a table can be distributed over multiple servers by using column groups, where pre-defined groups indicate which columns are best stored together.

A brief comparison of noSQL databases is shown in Table 1. For a more detailed comparison see also [20][28][39].

[20] Last queries can be lost as explained in http://redis.io/topics/persistence

	DynamoDB	Cassandra	Hbase	Redis	CouchDB	BigTable	MongoDB	Neo4j
Type	Key-Value	Column	Column	Key-Value	Document	Column	Document	Graph
Data Storage	MEM, FS	HDFS, CFS	HDFS	MEM, FS	MEM, FS	GFS	MEM, FS	MEM, FS
MapReduce	yes	yes	yes	no	yes	yes	yes	no
Persistence	yes	yes	yes	yes, with limits[20]	yes	yes	yes	yes
Replication	yes	yes	yes	yes	yes	yes	yes	yes
Scalability	high	high	high	high	high	high	high	high
Performance	high	high	high	high	high	high	high	high, variable
High availability	yes	yes	yes	yes	yes	yes	yes	yes
Language	Java	Java	Java	Ansi-C	Erlang	Java, Python, Go, Ruby	C++	Java
License	Proprietary	Apache 2.0	Apache 2.0	BSD	Apache 2.0	Proprietary	GNU AGPL3	GNU GPL3

Table 1. Comparison of some NoSQL databases. FS=File System; MEM=In-Memory

Google Bigtable

Google Bigtable[21] is a popular table store. Built above the Google File System, it is able to store up to petabytes of data and supporting tables with billions of rows and thousands of columns. Thanks to its high read and write throughput at low latency, Bigtable it is an ideal data source for batch MapReduce operations [9] and other applications oriented to the processing and analysis of large volumes of data.

Data in Bigtable are stored in sparse, distributed, persistent, multi-dimensional tables composed of rows and columns. Each row is indexed by a single row key, and a set of columns that are grouped together into sets called *column families*. Instead, a generic column is identified by a column family and a *column qualifier*, which is a unique name within the column family. Each value in the table is indexed by a tuple (row key, column key, timestamp). To improve scalability and to balance the query workload, data are ordered by row key and the row range for a table is dynamically partitioned into contiguous blocks, called *tablets*. These tablets are distributed among different Bigtable cluster's nodes (i.e., *Tablet Servers*). To improve load balancing, the Bigtable master is able to split larger and merge smaller tablets, redistributing them across nodes as needed. To ensure data durability, Bigtable stores data on Google File System (GFS) and protects it from disaster events through data replication and backup. Bigtable can be used into applications through multiple clients, including *Cloud Bigtable HBase*, a customized version of the standard client for the industry-standard Apache HBase.

Apache Cassandra

Apache Cassandra[22] is a distributed database management system providing high availability with no single point of failure. Born at Facebook and inspired by Amazon Dynamo and Google BigTable, Apache Cassandra is designed for managing large amount of data across multiple data centers and Cloud availability zones.

Cassandra uses a masterless ring architecture, where all nodes play an identical role, that allows any authorized user to connect to any node in any data

[21] https://cloud.google.com/bigtable/
[22] http://cassandra.apache.org/

center. This is a really simple and flexible architecture that allows to add nodes without service downtime. The process of data distribution across nodes is very simple and no programmatic operations are needed by the developers.

Since all nodes communicate each other equally, Cassandra has no single point of failure, that ensures continuous data availability and service uptime. Moreover, Cassandra provides very customizable data replication service that allows to replicate data across nodes that participate in a ring. In this manner, in case of node failure, one or more copies of the needed data are available on other nodes.

Cassandra also provides built-in and customizable replication, which stores redundant copies of data across nodes that participate in a Cassandra ring. This means that if a node in a cluster goes down, one or more copies of data stored on that node is available on other machines in the cluster. Replication can be configured to work across one data center, many data centers, and multiple Cloud availability zones. Focusing on performance and scalability, Cassandra reaches a quite linear speedup, that means the OPS (Operations Per Second) capacity can be increased by adding new nodes (e.g., if 2 nodes can handle 10,000 OPS, 4 nodes will support 20,000 OPS, and so on).

Many companies have successfully deployed and benefited from Apache Cassandra including some large companies such as: Apple (75,000 nodes storing over 10 PB of data), Chinese search engine Easou (270 nodes, 300 TB, over 800 million reqests per day), and eBay (over 100 nodes, 250 TB), Netflix (2,500 nodes, 420 TB, over 1 trillion requests per day), Instagram, Spotify, eBay, Rackspace, and many more.

Neo4j Graph Database

If we need to take into account real time data relationships (e.g. create queries using data relationships), NoSQL databases are not the best choice. In fact, relationship-based or graph databases has been created for naturally supporting operations on data that use data relationships. Graph databases provide a novel and powerful data modeling technique that does not store data in tables, but in graph models [43], with several benefits in storing and retrieving data connected by complex relationships.

There are several graph data models, such as Neo4j, OrientDB, Virtuoso, Allegro, Stardog, InfiniteGraph. Among all we focus on Neo4j. Neo4j is an open-source NoSQL graph database implemented in Java and Scala that is considered the most popular graph database used today. The Neo4j source code and issue tracking are available on GitHub, with a large support community. It is used today by a very large number of organizations working in different sectors, including software analytics, scientific research, project management, recommendations, and social networks.

In the Neo4j graph model, each node contains a list of relationship records that refer to other nodes, and additional attributes (e.g. timestamp, metadata, key-value pairs, and more). Each relationship record must have a name, a direction, a start node and an end node, and can contains additional properties. One

o more labels can be assigned both to nodes and relationships. In particular, such labels can be used for representing the roles a node plays in the graph (e.g., user, address, company, and so on) or for associating indexes and constraints to groups of nodes. Figure 8 shows an example of a graph model used for detecting bank fraud.

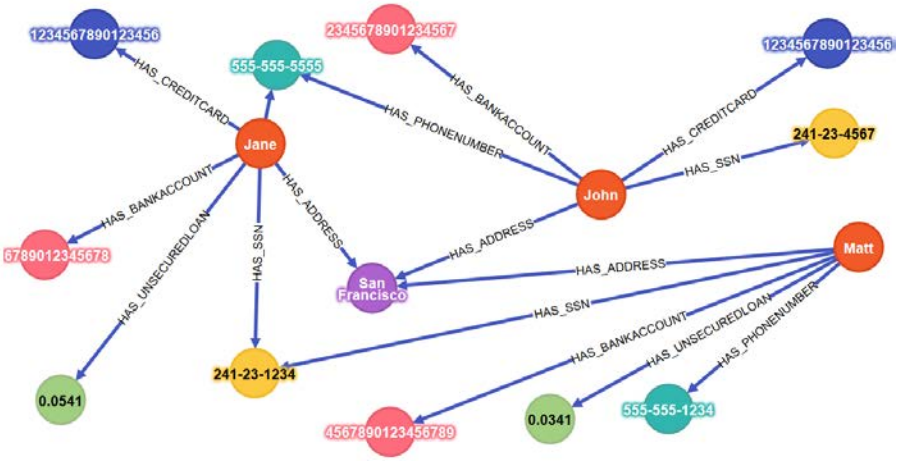

Fig. 8. Example of Bank Fraud Graph Dataset (source: neo4j.com).

Moreover, Neo4j clusters are designed for high availability and horizontal read scaling using master-slave replication. Focusing on performance, Neo4j is thousands of times faster than SQL in executing traversal operation. The traversal operation consists of visiting a set of nodes in the graph by moving along relationships (e.g., find potential friends in social network from user friendship). With such kind of operations, graph models allow to take into account only the data that is required, without doing expensive grouping operations as done by relational database during join operations [51]. Queries in Neo4j are written using *Cypher*, a declarative and SQL-based language for describing patterns in graphs. Cypher is a relative simple but very powerful language, that allows to execute queries in a easy way on a very complex graph database.

4.10 Visual Analytics

A primary problem in data analysis is to interpret results easily. To overcome this problem, in the last years, great progress has been made in the field of visual analytics. As defined by [50], visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces. Nowadays, people use visual analytics tools and methodologies to extract synthetic information from often confusing data and use them in further analysis or business operations. The

power of visual analytics techniques relies on human brain capabilities to process graphics faster than text. In particular, through a graphical data presentation, the human brain could be able to find complex and often hidden patterns and relationships in data that are difficult to discover using automatic methods. Also in the Big Data context, the tools used to visualize results and to interact with data play a key role. Thus, in order to support data presentation and interaction also in presence of Big Data, innovative methodologies (e.g, interactive charts, animations, diagrams, and much more) have been developed.

In particular, to ride the wave of visual analytics technologies, several big IT company, such as Microsoft, Google, and SAS, developed advanced data presentation and data visualization tools able to interact with existent Big Data platforms, including Hadoop-based ones. For example, Microsoft extended Excel functions to allow integration with its Big Data solution. In particular, Excel's users can be connected to Azure Storage associated to an Hadoop HDInsight cluster using the Microsoft Power Query for Excel add-in. Once data has been retrieved, users can exploit Excel functions to make more interesting charts or graphs.

Google Fusion Tables[23] is an other alternative for turning data into graphics in a very easy way. It allows to load tabular data, filter and summarize across hundreds of thousands of rows, and create geo maps, heat maps, graphs, charts, animations, and more. Also Google Charts[24] are a powerful Javascript library for making interactive charts for browsers and mobile devices. Google Charts allows to create several types of charts, from simple line charts to complex hierarchical tree maps. In the field of maps and location-based applications, advanced platforms, such as Google Maps[25], Mapbox[26], can be used to create interactive and dynamic maps, display additional layers on a map or generate routes. In the field of visual data analysis, several Big Data start-ups spring up in the last years. Tableau[27], for example, is a Big Data company from Stanford with multinational operations in fifteen cities, and more than 39,000 customer accounts in 150 countries. It developed software solutions for easily creating complex charts from huge amount of data. In fact, thanks to its Cloud analytics platform, Tableau allows users to manipulate data through a simple web control panel. In this way, users can interact directly with data to find interesting insights. Among all the competitors in this field, SAS[28] probably stands out among its peers.

SAS Visual Analytics, in fact, represents a complete solution for advanced data visualization and exploratory analyses. Thanks to its drag-and-drop capabilities and no code requirements, it allows users to easily solve complex issues using several sophisticated techniques for data analysis (e.g. decision trees, network diagrams, scenario analysis, path analysis, sentiment analysis) and business

[23] https://tables.googlelabs.com
[24] https://developers.google.com/chart
[25] https://www.google.com/maps
[26] https://www.mapbox.com/
[27] http://www.tableau.com
[28] https://www.sas.com

intelligence. In addition, exploiting in-memory processing, SAS software makes analytic applications faster.

4.11 Big Data funding projects

Open-source projects discussed in the previous sections (e.g., Hadoop, Spark, and NoSQL databases) have been widely used in several public funding projects. As examples:

- BigFoot project[29] is a cloud-based solution featuring scalable and optimized engines to store, process and interact with Big Data. It has received funding from the European Union's Horizon 2020 program.
- Optique[30] is a EU funding project with a total budget of about 14 million EUR. It is aims to provide a novel end-to-end OBDA (Ontology-Based Data Access) [38][7] solution for improving Big Data access. In particular, Optique platform allows to quickly formulate intuitive queries exploiting user vocabularies and conceptualizations, and executing them using massive parallelism.

Also government agencies invested large amount of money on Big Data technologies in many public sector fields, such as intelligence, defense, weather forecasting, crime prediction and prevention, and scientific research.

As example, US Administration invested more that 250 million USD for Big Data research and development initiative across multiple agencies and departments. Moreover, in 2014 UK government decided to invest about 73 million GBP in Big Data and other analytics technologies with the goals of creating 58,000 new jobs in Britain by 2017, contributing 216 billion GBP to the countrys economy.

4.12 Historical review

In this section a brief historical review of Big Data is presented. Undoubtedly, main events in Big Data evolution are due to big IT and Internet companies, like Google and Yahoo, who faced first the need of new solutions for tackling the rise of Big Data. A significant role in this context has been played by Hadoop and its related projects, that made Big Data analytics accessible also to a larger number of organizations.

Hadoop was created by Doug Cutting and it has its origins in Apache Nutch (2002), an open source web search engine, itself a part of the Lucene project (2000). After Google released the Google File System (GFS) paper (October 2003) and the MapReduce paper (December 2004), Cutting went to work with Yahoo and decided to build open source frameworks based on them: in 2006 Yahoo! created Hadoop based on GFS and MapReduce, and one year later, it started using Hadoop on a 1000 node cluster. In 2006, Yahoo Labs created

[29] http://bigfootproject.eu/
[30] http://optique-project.eu

Pig based on Hadoop, and then donated it to the *Apache Software Foundation* (ASF). In few years, several other projects was created around Hadoop and, in a short time, graduated to a Apache Top Level Project: HBase (2008), Hive (2008), Cassandra (2008), Storm (2011), Giraph (2011), and so on. At the same time, many Hadoop distributor was founded, such as Cloudera (2008), MapR (2009), Hortonworks (2011). A short history of Hadoop and related project is shown in Figure 9.

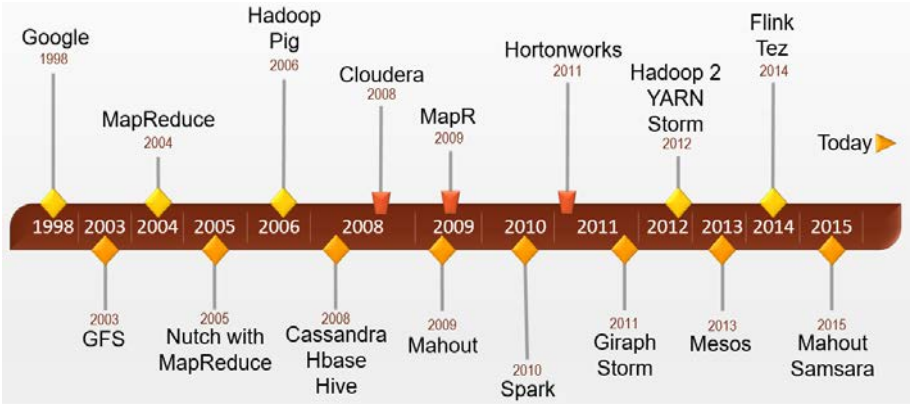

Fig. 9. A short Hadoop ecosystem's history.

Spark represents another milestone in Big Data analytics. Spark was initially created at UC Berkeley's AMPLab in 2009, open sourced in 2010 under a BSD license, and donated to the ASF in 2013. Finally, in February 2014, Spark became a Top-Level Apache Project and declared the most active ASF project. As discussed before, Spark is nowadays considered the primary execution engine for several Big Data applications, sometimes used to complement Hadoop.

4.13 Summary

It is not easy to summarize all the features of the systems discussed till now or to do a proof comparison among them. Some of those systems have common features and, in some cases, using one rather than another is an hard choice. In fact, given a specific data analytic task, such as a machine learning application, it is possible to use several tools. Some of those are widely used commercial tools, provided through cloud services, that can be easily used by no skilled people (e.g., Azure Machine Learning or Amazon Machine Learning); other are open-source frameworks that require skilled users who prefer to program their application using a more technical approach. In addition, choosing the best solution for developing a data analytic application may depend on many other factors, such as budget (e.g., often high-level services are easy-to-use but more expensive than

low-level solutions), data format, data source, the amount of data to be analyze and its velocity, and so on. Table 2 presents a brief comparison of the Big Data analytics systems.

Hadoop represents the most used framework for developing distributed Big Data analytics application. In fact, Hadoop-ecosystem is undoubtedly the most complete solution for any kind of problem, but at the same time it is thought for high skilled users. On the other hand, many other solutions are designed for low-skilled users or for low-medium organizations that do not want to spend resources in developing and maintaining enterprise data analytics solutions (e.g., Microsoft Azure Machine Learning, Amazon Machine Learning, Data Mining Cloud Framework, Kognitio Analytical, or BigML). Finally, other solutions have been created mainly for scientific research purposes and, for this reason, they are poorly used for developing business applications (e.g., Sector/Sphere, Pegasus).

Systems/ Tools	**Analytics**								**Open-source**	**Cloud model**
	Streaming	**Graph**	**In-Memory**	**Machine Learning**	**SQL**	**Data flow**	**Data processing**	**Workflow**		
Hadoop	x	x		x		x	x	x	x	IaaS
Spark	x	x	x	x	x		x		x	IaaS
Mahout			x	x					x	IaaS
Oozie								x	x	IaaS
Tez								x	x	IaaS
Giraph		x							x	IaaS
Storm	x								x	IaaS
Hive					x				x	IaaS
Pig						x			x	IaaS
Hunk										SaaS
Sector /Sphere							x		x	SaaS
BigML				x						SaaS, PaaS
Kognitio Analytical			x		x					PaaS
DMCF				x		x	x	x		SaaS, PaaS
Microsoft Azure ML				x			x	x		SaaS
Amazon ML	x		x	x			x			SaaS
Pegasus								x	x	IaaS
ClowdFlows								x	x	PaaS
Swift								x	x	IaaS

Table 2. A brief comparison of most common Big Data analytics systems.

Choosing the best database solution for creating a Big Data application is another key-step, so several aspects need to be considered. To decide what kind of database to adopt, the first aspect to be considered is probably the classes of queries will be run. So graph databases are probably the best solution for representing and querying highly connected data (e.g., data gathered from social network) or that have complex relationships and/or dynamic schema. In any other case, when non-graph data are analyzed, graph databases could result in

really bad performance. About that, summary considerations on graph databases are presented in Table 3.

Another aspect to be considered in choosing the best database solution should be the CAP (Consistency, Availability, and Partition) capabilities offered, because distributed NoSQL database systems can't be fully CAP compliant. In fact, the CAP theorem, also named Brewer's theorem[18], states that a distributed system can't simultaneously guarantee all three of the following properties:

- Consistency (C), that means all nodes see the same data at the same time;
- Availability (A), that means every request will receive a response within a reasonable amount of time;
- Partition (P) tolerance, that means the system continues to function also if arbitrary network partitions occur due to failures.

Thus if a distributed database system guarantees Consistency and Partitioning, it can never ensure Availability. Similarly, if you need a full Availability and Partition tolerance, you can't have Consistency, anyway not immediately. In fact, on a distributed environment, data changes on one node need some time to be propagated to the other nodes. During that time the copies will be mutually inconsistent, that may lead to the possibility of reading not updated data. To try to overcome this limitation, the *Eventual Consistency* property is usually provided: it ensures that the system, sooner or later, will become consistent. This is a weak property, so if the adopted database system only provides eventual consistency, the developer must be aware that exists the possibility of reading inconsistent data. NoSQL databases usually offer a balance among CAP properties, which is the key difference among the different available solutions. For each database family, some summary considerations are also provided for Key-Value databases (Table 4), Column-oriented (Table 5), and Document-oriented databases (Table 6).

Graph databases	
Horizontal scaling	Poor horizontal scaling.
When to use	For storing objects without a fixed schema and linked together by relationships; when users can done naturally their reasoning about data via graph traversals instead of using complex SQL queries.
CAP tradeoff	Usually prefer availability over consistency
Pros	Powerful data modeling and relationships representation; locally indexed connected data; easy to query.
Cons	Highly specialized query capabilities that make them the best for graph data, but not suitable for non-graph data.

Table 3. Summary considerations about graph databases.

Key-value databases	
Horizontal scaling	Very high scale provided via sharding.
When to use	When you have a very simple data schema or extreme speed scenario (like real-time)
CAP tradeoff	Most solutions prefer consistency over availability.
Pros	Simple data model; very high scalability, data can be accessed using query language like SQL.
Cons	Some queries could be inefficient or limited due to sharding (e.g., join operations across shards); no API standardization; maintenance is difficult; poor for complex data.

Table 4. Summary considerations about Key-Value databases.

Column-oriented databases	
Horizontal scaling	Very high scale capabilities.
When to use	When you need consistency and higher scalability performance than a single machine (i.e., usually using more than 1,000 nodes), without using indexed caching front end.
CAP tradeoff	Most solutions prefer consistency over availability.
Pros	Higher throughput and stronger concurrency when it is possible to partition data; multi-attribute queries; data is naturally indexed by columns; support semi-structured data.
Cons	More complex than the document stores; poor for interconnected data.

Table 5. Summary considerations about Column-oriented databases.

Document-oriented databases	
Horizontal scaling	Scale provided via replication or replication and sharding.
When to use	When your record structure is relatively small and it is possible to store all of its related properties in a single doc.
CAP tradeoff	In most cases prefer consistency over availability.
Pros	High scalability and simple data model; generally support secondary indexes, multiple types of documents per database, and nested documents or lists; MapReduce support for adhoc querying.
Cons	Eventually consistent model with limited atomicity and isolation; poor for interconnected data; query model is limited to keys and indexes.

Table 6. Summary considerations about Document-oriented databases.

5 Research Trends

Big Data analysis is a very active research area with significant impact on industrial and scientific domains where is important to analyze very large and complex data repositories. In particular, in many cases data to be analyzed are stored in Cloud platforms and elastic computing Clouds facilities are exploited to speedup the analysis. This section outlines and discusses main research trends in Big Data analytics and Cloud systems for managing and mining large-scale data repositories.

As we discussed, scalable data analytics requires high-level, easy-to-use design tools for programming large applications dealing with huge, distributed data sources. Moreover, Clouds are widely adopted by many organizations, however several existing issues remain to be addressed, so that Cloud solutions can improve their efficiency and competitiveness at each business size, from medium to large companies. This requires further research and development in several key areas such as:

- Programming models for Big Data analytics. Big Data analytics programming tools require novel complex abstract structures. The MapReduce model is often used on clusters and Clouds, but more research is needed to develop scalable higher-level models and tools. State-of-the-art solutions generated major success stories, however they are not mature and suffer several problems from data transfer bottlenecks to performance unpredictability. Several other processing models have been proposed as alternative to MapReduce, such as Dryad [21] or Pregel [30], but they have never been widely used by developers.
- Data storage scalability. The increasing amount of data generated needs even more scalable data storage systems. As discussed in the previously, traditional RDBMS systems are not the best choice for supporting Big Data applications in the Cloud, and that leads to the popularity of noSQL platforms[8]. Several noSQL solutions have been proposed, with good experimental results in term of performance gain, but several other improvements are still needed [52][47]. In fact, RDBMS systems have been around for a long time, are quite stable and offers lots of features. In the other hand, most noSQL systems are in its early version and several additional features have yet to be improved or implemented, such as integrating capabilities from DBMS (e.g., indexing techniques), facilities for ad-hoc queries, and more.
- Data availability. Cloud service provides have to deal with the problem of granting service and data availability. Especially in presence of huge amounts of data, granting high-quality service is an opened challenge. Several solutions have been proposed for improving exploitation, such as using a cooperative multi-Cloud model to support Big Data accessibility in emergency cases [25], but more studies are still needed to handle the continue increasing demand for more real time and broad network access to Cloud services.
- Data and tool interoperability and openness. Interoperability is a main issue in large-scale applications that use resources such as data and computing

nodes. Standard formats and models are needed to support interoperability and ease cooperation among teams using different data formats and tools. The *National Institute of Standards and Technology* (NIST) just released the Big Data interoperability framework[31], a collection of documents, organized in 7 volumes, which aim to define some standards for Big Data.

- Data quality and usability. Big Data sets are often arranged by gathering data from several heterogeneous and often not well-known sources. This leads to a poor data quality that is a big problem for data analysts. In fact, due to the lack of a common format, inconsistent and useless data can be produced as a result of joining data from heterogeneous sources. Defining some common and widely adopted format would lead to data that are consistent with data from other sources, that means high quality data. Since real-world data is highly susceptible to inconsistency, incompleteness, and noise, finding effective methodologies for data preprocessing is still an open challenge for improve data quality and the analysis results [10]. In this regard, an interesting discussion about challenges of data quality in the Big Data has been presented in [6].
- Integration of Big Data analytics frameworks. The service-oriented paradigm allows running large-scale distributed workflows on heterogeneous platforms along with software components developed using different programming languages or tools. Scalable software architectures for fine grain in-memory data access and analysis. Exascale processors and storage devices must be exploited with fine-grain runtime models. Software solutions for handling many cores and scalable processor-to-processor communications have to be designed to exploit exascale hardware [36][13].
- Tools for massive social network analysis. The effective analysis of social network data on a large scale requires new software tools for real-time data extraction and mining, using Cloud services and high-performance computing approaches [31][35]. Social data streaming analysis tools represent very useful technologies to understand collective behaviors from social media data. Tools for data exploration and models visualization. New approaches to data exploration and models visualization are necessary taking into account the size of data and the complexity of the knowledge extracted. As data are bigger and bigger, visualization tools will be more useful to summarize and show data patterns and trends in a compact and easy-to-see way.
- Local mining and distributed model combination. As Big Data applications often involve several local sources and distributed coordination, collecting distributed data sources to a centralized server for analysis is not practical or in some cases possible. Scalable data analysis systems have to enable local mining of data sources and model exchange and fusion mechanisms to compose the results produced in the distributed nodes [55]. According to this approach the global analysis can be performed by distributing the local mining and supporting the global combination of every local knowledge to generate the complete model.

[31] http://www.nist.gov/itl/bigdata/bigdatainfo.cfm

– In-memory analysis. Most of the data analysis tools query data sources on disks while, differently from those, in-memory analytics query data in main memory (RAM). This approach brings many benefits in terms of query speed up and faster decisions. In-memory databases are, for example, very effective in real-time data analysis, but they require high-performance hardware support and fine-grain parallel algorithms [49][59]. New 64-bit operating systems allow to address memory up to one terabyte, so making realistic to cache very large amount of data in RAM. This is why this research area is very promising.

6 Conclusions

In the last years the ability to gather data has increased exponentially. Advances and pervasiveness of computers have been the main driver of the very huge amounts of digital data that today are collected and stored in digital repositories. Those data volumes can be analyzed to extract useful information and producing helpful knowledge for science, industry, public services and in general for humankind. However, the huge amount of data generated, the speed at which it is produced, and its heterogeneity, represent a challenge to the current storage, process and analysis capabilities. Then to extract value from such kind of data, novel technologies and architectures have been developed by data scientists for capturing and analyzing complex and/or high velocity data. In this scenario was born also the Big Data mining field as a discipline that today provides several different techniques and algorithms for the automatic analysis of large data sets. But, the process of knowledge discovery from Big Data is not so easy, mainly due to data characteristics, and to get valuable information and knowledge in shorter time, high performance and scalable computing systems are needed. In many cases, Big Data are stored and analyzed in Cloud platforms.

Clouds provide scalable storage and processing services that can be used for extracting knowledge from Big Data repositories, as well as software platforms for developing and running data analysis environments on top of such services. In this chapter we provided an overview of Cloud technologies by describing the main service models (SaaS, PaaS, and IaaS) and deployment models (public, private or hybrid Clouds) adopted by Cloud providers. We also described representative examples of Cloud environments (Microsoft Azure, Amazon Web Services, OpenNebula and OpenStack) that can be used to implement applications and frameworks for data analysis in the Cloud. The development of data analysis applications on Cloud computing systems is a complex task that needs to exploit smart software solutions and innovative technologies. In this chapter we presented the leading software tools and technologies used for developing scalable data analysis on Clouds, such as MapReduce, Spark, workflow systems, and NoSQL database management systems. In particular, we particularly focused on Hadoop, the best-known MapReduce implementation, that is commonly used to develop scalable applications that analyze big amounts of data. As we discussed, Hadoop is also a reference tool for several other frame-

works, such as Storm, Hive, Oozie and Spark. Moreover, besides Hadoop and its ecosystem, several other MapReduce implementations have been implemented within other systems, including GridGain, Skynet, MapSharp, and Disco.

As such Cloud platforms become available, researchers are increasingly porting powerful data mining programming tools and strategies to the Cloud to exploit complex and flexible software models, such as the distributed workflow paradigm. Workflows provide a declarative way of specifying the high-level logic of an application, hiding the low-level details. They are also able to integrate existing software modules, datasets, and services in complex compositions that implement discovery processes. In this chapter we presented several data mining workflow systems, such as Data Mining Cloud Framework, Microsoft Azure Machine Learning, ClowdFlows.

Then we also discussed NoSQL database technology that became popular in the latest years as an alternative or as a complement to relational databases. In fact, NoSQL systems in several application scenarios are more scalable and provide higher performance than relational databases. We introduced the basic principles of NoSQL, described representative NoSQL systems, and outlined interesting data analytics use cases where NoSQL tools are useful. Finally, some research trends and open challenges on Big Data analysis has been discussed, such as scalable data analytics requirements of high-level, easy-to-use design tools for programming large applications dealing with huge distributed data sources.

Acknowledgement

This work is partially supported by EU under the COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS).

References

1. Abramova, V., Bernardino, J., Furtado, P.: Which nosql database? a performance overview. Open Journal of Databases (OJDB) 1(2), 17–24 (2014)
2. Barga, R., Gannon, D., Reed, D.: The client and the cloud: Democratizing research computing. Internet Computing, IEEE 15(1), 72–75 (Jan 2011)
3. Belcastro, L., Marozzo, F., Talia, D., Trunfio, P.: Programming visual and script-based big data analytics workflows on clouds. In: Big Data and High Performance Computing, Advances in Parallel Computing, vol. 26, pp. 18–31. IOS Press (2015)
4. Bermingham, L., Lee, I.: Spatio-temporal sequential pattern mining for tourism sciences. Procedia Computer Science 29(0), 379 – 389 (2014), 2014 International Conference on Computational Science
5. Bowers, S., Ludäscher, B., Ngu, A.H., Critchlow, T.: Enabling scientificworkflow reuse through structured composition of dataflow and control-flow. In: Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on. pp. 70–70. IEEE (2006)
6. Cai, L., Zhu, Y.: The challenges of data quality and data quality assessment in the big data era. Data Science Journal 14, 2 (2015)

7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The dl-lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
8. Cattell, R.: Scalable sql and nosql data stores. ACM SIGMOD Record 39(4), 12–27 (2011)
9. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS) 26(2), 4 (2008)
10. Che, D., Safran, M., Peng, Z.: Database Systems for Advanced Applications: 18th International Conference, DASFAA 2013, International Workshops: BDMA, SNSM, SeCoP, Wuhan, China, April 22-25, 2013. Proceedings, chap. From Big Data to Big Data Mining: Challenges, Issues, and Opportunities, pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
11. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6. pp. 10–10. OSDI'04, Berkeley, USA (2004)
12. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., da Silva, R.F., Livny, M., et al.: Pegasus, a workflow management system for science automation. Future Generation Computer Systems 46, 17–35 (2015)
13. Dongarra, J., et al.: The international exascale software project roadmap. International Journal of High Performance Computing Applications (2011)
14. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.H., Qiu, J., Fox, G.: Twister: A runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. pp. 810–818. HPDC '10, ACM, New York, NY, USA (2010)
15. Gajendran, S.K.: A survey on nosql databases. University of Illinois (2012)
16. Gerber, M.S.: Predicting crime using twitter and kernel density estimation. Decision Support Systems 61, 115 – 125 (2014)
17. Giardine, B., Riemer, C., Hardison, R.C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al.: Galaxy: a platform for interactive large-scale genome analysis. Genome research 15(10), 1451–1455 (2005)
18. Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News 33(2), 51–59 (2002)
19. Gu, Y., Grossman, R.L.: Sector and sphere: the design and implementation of a high-performance data cloud. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 367(1897), 2429–2445 (2009)
20. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of big data on cloud computing: Review and open research issues. Information Systems 47, 98 – 115 (2015)
21. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: Distributed data-parallel programs from sequential building blocks. SIGOPS Oper. Syst. Rev. 41(3), 59–72 (Mar 2007)
22. Kranjc, J., Podpečan, V., Lavrač, N.: Clowdflows: a cloud based scientific workflow platform. In: Machine Learning and Knowledge Discovery in Databases, pp. 816–819. Springer (2012)
23. Kurashima, T., Iwata, T., Irie, G., Fujimura, K.: Travel route recommendation using geotags in photo sharing sites. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management. pp. 579–588. CIKM '10, ACM, New York, NY, USA (2010)

24. Lee, R., Wakamiya, S., Sumiya, K.: Urban area characterization based on crowd behavioral lifelogs over twitter. Personal and Ubiquitous Computing 17(4), 605–620 (2013)
25. Lee, S., Park, H., Shin, Y.: Cloud computing availability: multi-clouds for big data service. In: Convergence and Hybrid Information Technology, pp. 799–806. Springer (2012)
26. Lemieux, A.: Geotagged photos: a useful tool for criminological research? Crime Science 4(1), 3 (2015)
27. Li, A., Yang, X., Kandula, S., Zhang, M.: Cloudcmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. pp. 1–14. ACM (2010)
28. Lourenço, J.R., Cabral, B., Carreiro, P., Vieira, M., Bernardino, J.: Choosing the right nosql database for the job: a quality attribute evaluation. Journal of Big Data 2(1), 1–26 (2015)
29. Lyubimov, D., Palumbo, A.: Apache Mahout: Beyond MapReduce. Chapman and Hall/CRC (February 2016)
30. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: A system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data. pp. 135–146. SIGMOD '10, ACM, New York, NY, USA (2010)
31. Marciani, G., Piu, M., Porretta, M., Nardelli, M., Cardellini, V.: Real-time analysis of social networks leveraging the flink framework. In: Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems. pp. 386–389. DEBS '16, ACM, New York, NY, USA (2016)
32. Marozzo, F., Talia, D., Trunfio, P.: A cloud framework for parameter sweeping data mining applications. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. pp. 367–374. IEEE (2011)
33. Marozzo, F., Talia, D., Trunfio, P.: Using clouds for scalable knowledge discovery applications. In: Euro-Par Workshops. Lecture Notes in Computer Science, vol. 7640, pp. 220–227. Rhodes Island, Greece (August 2012)
34. Marozzo, F., Talia, D., Trunfio, P.: Scalable script-based data analysis workflows on clouds. In: Proceedings of the 8th Workshop on Workflows in Support of Large-Scale Science. pp. 124–133. ACM (2013)
35. Martin, A., Brito, A., Fetzer, C.: Real-time social network graph analysis using streammine3g. In: Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems. pp. 322–329. DEBS '16, ACM, New York, NY, USA (2016)
36. Mavroidis, I., Papaefstathiou, I., Lavagno, L., Nikolopoulos, D.S., Koch, D., Goodacre, J., Sourdis, I., Papaefstathiou, V., Coppola, M., Palomino, M.: Ecoscale: Reconfigurable computing and runtime system for future exascale systems. In: 2016 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 696–701 (2016)
37. Mell, P.M., Grance, T.: Sp 800-145. the nist definition of cloud computing. Tech. rep., National Institute of Standards & Technology, Gaithersburg, MD, United States (2011)
38. Möller, R., Neumann, B.: Ontology-Based Reasoning Techniques for Multimedia Interpretation and Retrieval, pp. 55–98. Springer London, London (2008)
39. Moniruzzaman, A.B.M., Hossain, S.A.: Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. CoRR abs/1307.0191 (2013)

40. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on. pp. 124–131 (May 2009)
41. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in Action. Manning Publications Co., Greenwich, CT, USA (2011)
42. Richardson, L., Ruby, S.: RESTful web services. "O'Reilly Media, Inc." (2008)
43. Rodriguez, M.A., Neubauer, P.: The graph traversal pattern. CoRR abs/1004.1001 (2010)
44. Shahrivari, S.: Beyond batch processing: Towards real-time and streaming big data. CoRR abs/1403.3375 (2014)
45. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing 13(5), 14–22 (Sept 2009)
46. Stonebraker, M.: Sql databases v. nosql databases. Commun. ACM 53(4), 10–11 (Apr 2010)
47. Tai, A., Wei, M., Freedman, M.J., Abraham, I., Malkhi, D.: Replex: A scalable, highly available multi-index data store. In: 2016 USENIX Annual Technical Conference (USENIX ATC 16). pp. 337–350. USENIX Association, Denver, CO (Jun 2016)
48. Talia, D., Trunfio, P., Marozzo, F.: Data Analysis in the Cloud. Elsevier (2015), iSBN 978-0-12-802881-0
49. Tan, K.L., Cai, Q., Ooi, B.C., Wong, W.F., Yao, C., Zhang, H.: In-memory databases: Challenges and opportunities from software and hardware perspectives. SIGMOD Rec. 44(2), 35–40 (Aug 2015)
50. Thomas, J.J., Cook, K.A.: A visual analytics agenda. Computer Graphics and Applications, IEEE 26(1), 10–13 (2006)
51. Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., Partner, J.: Neo4j in action. Manning (2015)
52. Wang, Z., Chu, Y., Tan, K., Agrawal, D., El Abbadi, A., Xu, X.: Scalable data cube analysis over big data. CoRR abs/1311.5663 (2013)
53. Wilde, M., Hategan, M., Wozniak, J.M., Clifford, B., Katz, D.S., Foster, I.: Swift: A language for distributed parallel scripting. Parallel Computing 37(9), 633–652 (2011)
54. Wozniak, J.M., Wilde, M., Foster, I.T.: Language features for scalable distributed-memory dataflow computing. In: Data-Flow Execution Models for Extreme Scale Computing (DFM), 2014 Fourth Workshop on. pp. 50–53 (Aug 2014)
55. Wu, X., Zhu, X., Wu, G.Q., Ding, W.: Data mining with big data. IEEE Transactions on Knowledge and Data Engineering 26(1), 97–107 (2014)
56. Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I.: Shark: Sql and rich analytics at scale. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. pp. 13–24. SIGMOD '13, ACM, New York, NY, USA (2013)
57. You, L., Motta, G., Sacco, D., Ma, T.: Social data analysis framework in cloud and mobility analyzer for smarter cities. In: Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on. pp. 96–101 (Oct 2014)
58. Yuan, J., Zheng, Y., Zhang, L., Xie, X., Sun, G.: Where to find my next passenger. In: Proceedings of the 13th International Conference on Ubiquitous Computing. pp. 109–118. UbiComp '11, ACM, New York, NY, USA (2011)

59. Zhang, H., Chen, G., Ooi, B.C., Tan, K.L., Zhang, M.: In-memory big data management and processing: A survey. IEEE Transactions on Knowledge and Data Engineering 27(7), 1920–1948 (July 2015)