TASSONOMY AND REVIEW OF BIG DATA SOLUTIONS NAVIGATION

PIERFRANCESCO BELLINI, MARIANO DI CLAUDIO, PAOLO NESI, NADIA RAUCH,

Distributed Systems and Internet Technology, Department of Systems and Informatics,
University of Florence, Firenze, Italy,
tel: +39-0554796523, fax: +39-055-4796363, http://www.disit.dsi.unifi.it
pbellini@dsi.unifi.it, diclaudi@dsi.unifi.it, paolo.nesi@unifi.it, rauch@dsi.unifi.it

ABSTRACT

In recent years more and more often, we hear about Big Data and issues related to the management of these huge volume of data, and considering that there is no definitive solution for their storage, querying, analysis. This discipline is in great evolution, multidisciplinary and very complex to be dominated for the several aspects to be addressed: architectural, data structure, data management, data analytics, protection and security, computational as parallel and distributed processing, etc. To cope with these problems a basic survey of main tips, which would allow the developer to orient themselves in the choice of the best solution, for the development of an architecture for the management of Big Data, in each specific case could be of great support and help. In fact a large number of fields, from industry to scientific research, are inquiring information and hints about what can be obtained through the analysis of Big Data and how these infrastructures can be created, by using what, etc. And, therefore, more powerful solutions are needed to cope with increasing complexity and variety of problems, for their better management and exploitation of the open accessible, produced and integrated data. In this paper, starting from the analysis of the existing solutions, particularly interesting and well documented use cases, we identified a group of main differentiating features which can mainly influence the choice of the solution to be set up; then we looked at different types of existing solutions and products to see how they are handled the identified features. Lastly, the results obtained with the analysis of the desirable features for each main domain and then the identification of the most suitable and/or adopted products for the application domains. The work cannot be exhaustive, and in many cases we had to decide to include and to exclude aspects and tools. The obtained results can be regarded as a model and main guidelines for big data solution navigation.

1. INTRODUCTION

The management of huge and growing volumes of data is a challenge since many years, whereas nowadays no long term solutions have been found. The term “Big Data” initially referred to huge volumes of data that have size beyond the capabilities of current database technologies, successively for “Big Data” problems one referred to the problems that present a combination of large volume of data to be treated in short time. When one establish that data have to be collected and stored at an impressive rate, it is clear that the biggest challenge is not only about the storage and management; their analysis and the extraction of meaningful values, deductions and actions is in reality the main challenge. Big data problems were mostly related to the presence of unstructured data, i.e. information that either do not have a default schema/template or do not adapt well to relational tables; it is therefore necessary to turn to analysis techniques for unstructured data, to address these problems.

Recently the big data problems are characterized by a combination of the so called 3V’s: *volume*, *velocity*, *variety*; and then a forth V has been added: *variability*. In substance, every day a large *volume* of information is produced and this data need a sustainable access, process and preservation according to the *velocity* of their arrival, therefore the management of large volume of data is not the only problem. Moreover, the *variety* of data, metadata, access rights and associating computing, formats, semantics and software tools for visualization, and the *variability* in structure and data models, significantly increase the level of complexity of these problems. The first V, *volume*, describes the large amount of data generated by individuals, groups and organizations. The volume of data being stored today is exploding. For example in the year 2000 were generated and stored about 800.000 Petabytes of data in the world [Eaton et al., 2012] and experts estimated that in the year 2020, about 35 ZettaByte of data will be

produced. The second V, *velocity*, refers to speed at which Big data are collected, processed and elaborated, may be handle constant flow of massive data that are impossible to be processed with traditional solutions. For this reason, it not important only to consider “where” the data are stored, but also “how” they are stored. The third V, *variety*, is concerned to the proliferation of data types from social, mobile sources, machine-to-machine and traditional data that are part of it. With the explosion of Social Networks, smart devices, sensors, data has become complex, because it includes raw, semi-structured and unstructured data from log files, web pages, search indexes, cross media, emails, documents, forums and so on. *Variety* represents all type of data and usually the enterprises must be able to analyse all them, if they want to gain advantages. Finally, *Variability*, the last V, refers to data unpredictability and to how these may change in the years, following the implementation of the architecture. Moreover, the concept of variability can be connected to assigning a variable interpretation to the data and to the confusions created in big data Analysis, referring for example, to different meanings in Natural Language that some data may have. These four properties can be considered orthogonal aspects of data storage, processing and analysis and it is also interesting that increasing variety and variability, also increases the attractiveness of data and their potentiality in providing hidden and unexpected information/meanings.

Especially in science, the need of new "*infrastructures for global research data*" that can achieve interoperability, and to overcome the limitations related to language, methodology and guidelines (policy), would be needed in short time. To cope with these types of complexities, several different techniques and tools may be needed, they have to be composed and new specific algorithms and solutions defined and implemented. The wide range of problems and the specifics needs make almost impossible to identify unique architectures and solutions adaptable to all possible applicative areas. Moreover, not only the number of application areas, so different from each other, but also the different channels through which data are daily collected, increases the difficulties of companies and developers to identify which is the right way to achieve relevant results from the accessible data. Therefore, this chapter can be a useful tool for supporting the researchers and technicians in making decisions about setting up some big data infrastructure and solutions. To this end, it can be very helpful to have an overview about big data techniques; it can be used as a sort of guideline to better understand possible differences and relevant best features among the many needed and proposed by the product as the key aspects of big data solutions. These can be regarded as requirements and needs according of which the different solutions can be compared and assessed, in accordance with the case study and/or application domain.

*To this end, and to better understand the impact of big data science and solutions, in the following, a number of **examples** describing major applicative domains taking advantage from the big data technologies and solutions are reported: education and training, cultural heritage, social media and social networking, health care, research on brain, financial and business, marketing and social marketing, security, smart cities and mobility, etc.*

Big data technologies have the potential to revolutionize education. **Educational data** like students' performance, mechanics of learning and answers to different pedagogical strategies, can provide an improved understanding of students' knowledge and accurate assessments of their progress. These data can also help identify clusters of students with similar learning style or difficulties, thus defining a new form of customized education based on sharing resources and supported by computational models. The proposed new models of teaching in [Woolf, Baker and Gianchandani, 2010] are trying to take into account, student profile and performance, pedagogical and psychological and learning mechanisms, to define personalized instruction courses and activities that meets the different needs of different individual students and/or groups. In fact, in recent years, it has been affirmed in the educational the approach to collect, mine and analyse large datasets, in order to provide new tools and information to the key stakeholders in education. This data analysis can provide an increasing understanding of students' knowledge, improve the assessments of their progress, and can help focus questions in education and psychology; such as the method of learning, or how different students respond to different pedagogical strategies. The collected data can also be used to define models to understand what students actually know and understand how to enrich this knowledge, and assess which of the adopted techniques can be effective in whose cases, and finally produce a case by case action plan. In terms of big data, a large variety and variability of data is present to take into account all

events in the students' career; the data volume is also an additional factor. Another sector of interest, in this field, is the e-learning domain, where are defined two mainly kinds of users: the learners and the learning providers [Hanna, 2004]. All personal details of learners and the online learning providers' information are stored in specific database, so applying data mining with e-learning can be able to realize teaching programs targeted to particular interests and needs through an efficient decision making.

For the management of large amounts of **cultural heritage** information data, Europeana has been created with over then 20 millions of content indexed which can be retrieve in real time. Each of them was early modelled with a simple metadata model, ESE, while a new and more complete models called EDM (Europeana Data Model) with a set of semantic relationships is going to be adopted in the 2013 [Europeana]. A number of projects and activities are connected to Europeana network to aggregate content and tools. Among them ECLAP is a best practice network, that collected not only content metadata for Europeana but real content files from over then 35 different institutions having different metadata sets and over than 500 file formats. A total of more than 1 million of cross media items is going to be collected with an average of some hundreds of metadata each, thus resulting in billions of information elements and multiple relationships among them to be queried, navigated and accessed in real time by a large community of users [ECLAP], [Bellini, Cenni and Nesi, 2012].

The volume of data generated by **social network** is great and with a highly variability in the data flow over time and space, due to human factor; e.g., Facebook receives 3 billion uploads per month, which corresponds to approximately 3600TB/year. Search engines companies like Google and Yahoo! collect every day trillions of bytes of data, around which real new business is developed, offering useful services to its users and companies in real time [Mislove, Gummandi and Druschel, 2006]. From these large amounts of data collected through social networks (e.g., Facebook, Twitter, MySpace), social media and big data solutions may estimate the user collective profiles and behaviour, analyse product acceptance, evaluate the market trend, keep trace of user movements, extract unexpected correlations, evaluate the models of influence, and perform different kinds of predictions [Domingos, 2005]. Social media data can be exploited by considering geo-referenced information and Natural Language Processing for analysing and interpreting urban living: massive folk movements, activities of the different communities in the city, movements due to large public events, assessment of the city infrastructures, etc. [Iaconesi and Persico, 2012]. In a broader sense by this information is possible to extract knowledge and data relationships, by improving the activity of query answering.

For example in **Healthcare/Medical field** large amount of information about patients' medical histories, symptomatology, diagnoses and responses to treatments and therapies is collected. Data mining techniques might be implemented to derive knowledge from this data in order to either identify new interesting patterns in infection control data or to examine reporting practices [Obenshain, 2004]. Moreover, predictive models can be used as detection tools exploiting **Electronic Patient Record** (EPR) accumulated for each person of the area, and taking into account the statistical data. Similar solutions can be adopted as decision support for specific triage and diagnosis or to produce effective plans for chronic disease management, enhancing the quality of healthcare and lower its cost. This activity may allow detecting the inception of critical conditions for the observed people over the whole population. In [Mans et al., 2009], techniques to the fast access and extraction of information from event's log from medical processes, to produce easily interpretable models, using partitioning, clustering and pre-processing techniques have been investigated. In medical field, especially hospital, run time data are used to support the analysis of existing processes. Moreover, to take into account genomic aspects and EPR for millions of patients leads to cope with big data problems. For genome sequencing activities (HTS, high throughput sequencing) that produce several hundreds of millions of small sequences, a new data structure for indexing called Gkarrays [Rivals et al., 2012], has been proposed, with the aim of improving classical indexing system such as hash table. The adoption of sparse hash tables is not enough to index huge collections of k-mer (sub-word of a given length k in a DNA sequence, which represent the minimum unit accessed). Therefore, new data structure have been proposed, that based on three arrays: the first for storing the start position of each k-mer, the second as an inverted array allows finding any k-mer from a position in a read, and the last records the interval of position of each distinct k-mer, in

sorted order. This structure allowed obtaining in constant time, the number of reads that contain a k-mer. A project of the University of Salzburg with the National Institute of sick of Salzburg studies how to apply machine learning techniques to the evaluation of large amounts of tomographic images generated by computer [Zinterhof, 2012]. The idea is to apply proven techniques of machine learning for image segmentation, in the field of computer tomography.

In several areas of **science and research** such as astronomy (automated sky survey), sociology (web log analysis of behavioural data) and neuroscience (genetic and neuro-imaging data analysis) the aim of big data analysis is to extract meaning from data and determine what actions take. To cope with the large amount of experimental data produced by research experiments, the University Montpellier started the ZENITH project [Zenith], that adopts a hybrid architecture p2p/cloud [Valduriez, Pacitti, 2005]. The idea of Zenith is to exploit p2p to facilitate the collaborative nature of scientific data, centralized control, and use the potentialities of computing, storage and network resources in the Cloud model, to manage and analyse this large amount of data. The storage infrastructure used in [De Witt et al., 2012] is called CASTOR, and allows the management of metadata related to scientific files of experiments at CERN. For example, the database of RAL (Rutherford Appleton Laboratory) uses a single table for storing 20GB (which reproduces the hierarchical structure of the file), that runs about 500 transactions per second on 6 clusters. With the increasing number of digital scientific data, one of the most important challenges is the digital preservation and for this purpose is in progress the SCAPE (SCAlable Preservation Environment) project [SCAPE Project]. The platform provides an extensible infrastructure to achieve the conservation of workflow information of large volume of data. The AzureBrain project [Antoniu et al., 2010] aims to explore cloud computing techniques for the analysis of data from genetic and neuroimaging domains, both characterized by a large number of variables. The Projectome project, connected with the **Human Brain** Project, HBP, aims to set up a high performance infrastructure for processing and visualizing neuro-anatomical information obtained by using co focal ultra-microscopy techniques [Silvestri et al., 2012], the solution is connected with the modelling of knowledge of and information related to rat brains. Here, the single image scan of a mouse is more than 1Tbyte and it is 1000 smaller than a human brain.

The task of finding patterns in **business data** in not new, today is getting a larger relevance because enterprises are collecting and producing huge amount of data including massive contextual information, thus taking into account a larger number of variables. Using data to understand and improve business operations, profitability and growth is a great opportunity and a challenge in evolving. The continuous collection of large amounts of data(business transaction, sales transaction, user behaviour), widespread use of networking technologies and computers, and design of big data warehouse and data mart have created enormously valuable assets. An interesting possibility to extract from these data meaningful information, could be the use of machine learning techniques in the context of mining business data [Bose et al., 2001], or also to use an alternative approach of structured data mining to model classes of customers in client databases using fuzzy clustering and fuzzy decision making [Setnes et al., 2001]. These data can be analysed in order to define prediction about the behaviour of users, to identify buying pattern of individual/group customers and to provide new custom services [Bose et al., 2001]. Moreover, in recent years, the major market analysts conduct their business investigations with data that are not stored within the classic RDBMS (Relational DataBase Management System), due to the increase of various and new types of information. Analysis of web users behaviour, customer loyalty programs, the technology of remote sensors, comments into blogs and opinions shared on the network are contributing to create a new business model called *social media marketing* and the companies must properly manage these information, with the corresponding potential for new understanding, to maximize the business value of the data [Domingos, 2005]. In financial field, instead, investment and business plans may be created thanks to predictive models derived using techniques of reasoning and used to discover meaningful and interesting patterns in business data.

Big data technologies have been adopted to find solutions to **logistic and mobility management** and optimization of multimodal transport networks in the context of Smart Cities. A data-centric approach can also help for enhancing

the efficiency and the dependability of a transportation system. In fact, through the analysis and visualization of detailed road network data and the use of a predictive model it is possible to achieve an intelligent transportation environment. Furthermore, through the merging of high-fidelity geographical stored data and real-time sensor networks scattered data, it can be made an efficient urban planning system that mix public and private transportation, offering people more flexible solutions. This new way of travelling has interesting implications for energy and environment. The analysis of the huge amount of data collected from the metropolitan multimodal transportation infrastructure, augmented with data coming from sensors, GPS positions, etc., can be used to facilitate the movements of people via local public transportation solutions and private vehicles [Liu, Biderman and Ratti, 2009]. The idea is to provide intelligent real time information to improve traveller experience and operational efficiencies (see for example the solutions for the cities of Amsterdam, Berlin, Copenhagen, and Ghent). In this way, in fact, is possible in order to use the big-data both as historical and real-time data for the applications of machine learning algorithms aimed to traffic state estimation/planning and also to detect unpredicted phenomena in a sufficiently accurate way to support near real-time decisions.

In security field, **Intelligence, Surveillance, and Reconnaissance** (ISR) define topics that are well suited for data-centric computational analyses. Using analysis tools for video and image retrieval, it is possible to establish alert for activity and event of interest. Moreover, intelligence services can use these data to detect and combine special patterns and trends, in order to recognize threats and to assess the capabilities and vulnerabilities with the aim to increase the security level of a nation [Bryant et al., 2010].

In the field of **energy resources optimization** and environmental monitoring, very important are the data related to the consumption of electricity. The analysis of a set of load profiles and geo-referenced information, with appropriate data mining techniques [Figueireido, Rodrigues and Vale, 2005], and the construction of predictive models from that data, could define intelligent distribution strategies in order to lower costs and improve the quality of life in this field, another possible solution is an approach that provides for the adoption of a conceptual model for a smart grid data management based on the main features of a cloud computing platform, such as collection and real-time management of distributed data, parallel processing for research and interpretation of information, multiple and ubiquitous access [Rusitschka, Eger and Gerdes, 2010].

In the above overview about some of the application domains for big data technologies, it is evident that to cope with those problems several different kinds of solutions and specific products have been developed. Moreover, the complexity and the variability of the problems have been addressed with a combination of different open source or proprietary solutions, since presently there is not an ultimate solution to the big data problem that includes in an integrated manner data gathering, mining, analysis, processing, accessing, publication and rendering. It would be therefore extremely useful a “map” of the hot spots to be taken into account, during the design process and the creation of these architectures, that helps the technical staff to orient themselves in the wide range of products accessible on internet and/or offered by the market. To this aim, we have tried to identify the main features that can characterize architectures for solving a big data problem, depending on the source of data, on the type of processing required, and on the application context in which should be to operate.

The paper is organized as follows. In section 2, the main requirements and features for big data solutions are presented by taking into account infrastructural and architectural aspects, data management, and data analytics aspects. Section 3 reports a brief overview of existing solutions for big data and their main application fields. In section 4,a comparison and analysis of the architectural features is presented. The analysis has permitted to put in evidence the most relevant features and different among the different solution. Section 5 is characterized by the description of the main application domains of the big data technologies and includes our assessment of these applicative domains in terms of the identified features reported in Section 3. Therefore, this section can be very useful to identify which are the major challenges of each domain and the most important aspects to be taken into account for each domain. This analysis allowed us to perform some comparison and consideration about the most commonly adopted tools in the different domains. Also in the same session, the identified application domains are

crossed with the solutions analysed in section 3, thus providing a short cut to determine whose products have already been applied to a specific field of application, that is, a hint for the development of future applications. Finally, in Section 6, conclusions are drawn.

2. MAIN REQUIREMENTS AND FEATURES OF BIG DATA SOLUTIONS

In this section, according to the above reported short overview of big data problems, we have identified a small number of main aspects that should be addressed by architectures for management big data problems. These aspects can be regarded as a collection of major requirements to cope with most of the issues related to the big data problems. We have divided the identified main aspects in three main categories which respectively concern with the infrastructure and the architecture of the systems that should cope with big data; with the management of the large amount of data and characteristics related to the type of physical storage; and with the accesses to data and techniques of data analytics, like ingestion, log analysis and everything else is pertinent to post-production phase of data processing. In some cases, the features are provided and/or inherited by the operating system or by the cloud/virtual infrastructure. Therefore, the specific big data solutions and techniques have to be capable to take advantage from the underlining operating system and the infrastructure.

2.1. INFRASTRUCTRALAND ARCHITECTURAL ASPECTS

The typical big data solutions are deployed on cloud exploiting the flexibility of the infrastructure. As a result, some of the features of big data solutions may depend on the architecture and infrastructure facilities from which the solution inherits/exploits the capabilities. Moreover, specific tool for data gathering, processing, rendering, etc., may be capable or not to exploit a different range of cloud based architectural aspects. For example, not all databases can be distributed on multiple servers, not all algorithms can be profitable remapped on a parallel architecture, not all data access or rendering solutions may exploit multilayered caches, etc. To this end, in the following paragraphs a set of main features are discussed, among them: scalability, multi-tiered memory, availability, parallel and distributed process management, workflow, self-healing, and data security and privacy. A summarization map is reported in Section 3.

Scalability: This feature may impact on the several aspects of the big data solution (e.g., data storage, data processing, rendering, computation, connection, etc.) and has to cope with the capability of maintaining acceptable performances coping from small to large problems. In most cases, the scalability is obtained by using distributed and/or parallel architectures, which may be allocated on cloud. Both computing and storage resources can be located over a network to create a distribute system where, managing also the distribution of workload.

As regards the **computational scalability**, processing a very huge dataset is important to optimize workload, for example with a parallel architecture, as proposed in Zenith project [Zenith], which may perform several operations simultaneously (on an appropriate number of tasks), or providing a dynamic allocation of computation resources(i.e., a process releases a resource as soon as is not more needed, thus it can be assigned to another process) technique used in the ConPaas platform [Pierre et al., 2011]. Usually, the traditional computational algorithms are not scalable and thus specific restructuring of the algorithms have to be defined and adopted. On the other hand, not all the algorithms can take advantage by parallel and/or distributed architectures for computing, specific algorithms have to be defined, provided that an efficient parallel and/or distributed solution exists. The evolution to distributed and parallel processing is just the first step, since processes have to be allocated and managed in some parallel architecture, which can be developed ad-hoc or generally setup. Semantic grid and parallel architectures can be used to the problem [Bellini et al., 2012a], [BlueGene],

Each system for big data may provide a **scalable storage** solution. In fact, the main problem could be to understand in which measure a storage solution has to be scalable to satisfy the worst operative cases or the most common cases

(and in general the most expensive cases). Moreover, for large experiments the data collection and processing may be not predictable with high precision in the long term, for example for the storage size and cost. For example, it is not clear how much storage would be needed to collected genomic information and EHR (Electronic Healthcare Records) for a unified European health system in 5 or 10 years. In any case because EHR contains a large amount of data, an interesting approach for their management could be the use of a solution based on HBase that builds a system distributed, fault-tolerant and scalable database on clouds, built on top of the HDFS, with random real-time read/write access to big data, overcoming the design limits of traditional RDBMS [Yang, Tang and Zhou, 2011]. Furthermore, to focus on this problem, a predictive model to understand how will increase the need for storage space should be made, while complexities and costs of this model are high. In most cases, it is preferable to have a pragmatic approach, first guess and work with the present problems by using cheap hardware and if necessary, increase the storage on demand. This approach obviously cannot be considered completely scalable, scalability is not just about the storage size, and then remains the need to associate the solution presented, with a system capable of scaling operationally [Snell, 2011].

A good solution to optimize the reaction time and to obtain a scalable solution at limited costs is the adoption of a **multi-tiered storage** system, including cache levels, where data passes from one level to another along the hierarchy of storage media having different response times and costs. In fact, a multi-tier approach to storage, utilizing arrays of disks for all backup with a primary storage and the adoption of an efficient file systems, allows to both provide backups and restores to online storage in a timely manner, as well as to scale up the storage when primary storage grows. Obviously each specific solution does not have to implement all layers of the memory hierarchy because their needs depends on the single specific case, together with the amount of information to be accessed per second, the deepness of the cache memories, binning in classes of different types of data based on their availability and recoverability, or the choice to use a middleware to connect separate layers. The structure of the multi-tiered storage can be designed on the basis of a compromise from access velocity to general storage cost. The multiple storages create as counterpart a large amount of maintenance costs.

Scalability may take advantage from the recent cloud solutions that implements techniques for dynamic and bursting on cloud storage and processes from private to public clouds and among the latters. Private cloud computing has recently gained much traction from both commercial and open-source interests [Microsoft, 2012]. For examples tools such as OpenStack [OpenStack Project] can simplifying the process of managing virtual machine resources. In most cases for small medium enterprises, there is the trend to migrate multi-tier applications into public cloud infrastructures (e.g., Amazon), which are delegated to cope with scalability via elastic cloud solutions. A deep discussion on cloud is out of the scope of this chapter.

High Availability: the high availability of a service (e.g., it may be referred to general service, to storage, process and network) is a key requirement in an architecture that can affect the simultaneous use of a large number of users and/or computational nodes located in different geographical locations [Cao, Wang and Xiong, 2009]. Availability refers to the ability of the community of users to access a system exploiting its services. A high availability leads to increased difficulties in guarantee data updates, preservations and consistency in real time, it is fundamental that a user perceives, during his session, the actual and proper reactivity of the system. To cope with these features, the design should be fault tolerant, as in redundant solution for data and computational capabilities to make them highly available despite to the failure of some hardware and software elements of the infrastructure. The availability of a system is usually expressed as a percentage of time (the nines method) that a system is up over a given period of time, usually a year. In cloud systems, for instance, the level of 5 nines (99.999% of time means HA, high availability) is typically related to the service at hardware level, and it indicates a downtime per year of approximately 5 minutes, but it is important to note that time does not always have the same value but it depends on the organization referred to by the critical system. The present solutions obtain the HA score by using a range of techniques of cloud architectures as **fault tolerant** capabilities for virtual machines, redundant storage for distributed database and balancing for the front end, and the dynamic move of virtual machines.

Computational Process Management: the computational activities on big data may take long time and may be distributed on multiple computational computers/nodes on some parallel architecture, in connection with some networking systems. Therefore, one of the main characteristics of most of the big data solutions has to cope with the needs of **controlling computational processes** by mean of: allocating them on a **distributed system**, putting them in execution on demand or periodically, killing them, recovering processing from failure, returning eventual errors, scheduling them over time, etc. Sometimes, the infrastructure that allow to put in execution parallel computational processes can work as a service, thus it has to be accessible for multiple users and/or other multitier architecture and servers. This means that sophisticated solutions for parallel processing and scheduling are needed, including the definition of Service Level Agreement, SLA, and in classical grid solutions. Example of solutions to cope these aspects are solutions for computational grid, media grid, semantic computing, distributed processing such as: AXCP media grid [Bellini at al., 2012b], general grid [Foster et al., 2002].The solution for parallel data processing has to be capable to dynamically exploits the computational power of the underlining infrastructure, since most of the big data problems may be computationally intensive for limited time slots. Cloud solutions may help to cope with the concepts of elastic cloud for implementing dynamic computational solutions.

Workflow Automation: Big data processes are typically formalized in the form of process workflows from data acquisition to results production. In some cases, the workflow is programmed by using simple XML (Extensible Markup Language) formalization or effective programming languages, for example in Java, JavaScript, etc. Related data may strongly vary in terms of dimensions and data flow (i.e., variability): an architecture that handles well with both limited and large volumes of data, must be able to full support creation, organization and transfer of these workflows, in single cast or broadcast mode. To implement this type of architectures, sophisticated automation systems are used. These systems work on different layers of the architecture through applications, API's (Application Program Interface), visual process design environment, etc. Traditional Workflow Management Systems, WfMS, may be not suitable for processing huge amount of data in real time, formalizing the stream processing, etc. In some big data applications the high data flow and timing requirements (soft real time) have made inadequate the traditional paradigm "*store-then-process*", so that the Complex Event Processing, CEP, paradigms been proposed [Gulisano et al., 2012]: a system that processes a continuous stream of data (event) on the fly, without any storage. In fact, the CEP can be regarded as an Event-Driven Architecture, EDA, dealing with the detection and production of reaction to events, that specifically has the task of filtering, match and aggregate low-level events in high-level events. Furthermore creating a parallel-distributed CEP, where data is partitioned across processing nodes, it is possible to realize an elastic system capable of adapting the processing resources to the actual workload reaching the high performance of parallel solutions and overcoming the limits of scalability.

An interesting application example is the Large Hadron Collider, LHC, the most powerful particle accelerator in the world, that is estimated to produce 15 million gigabytes of data every year [LHC], then made available to physicists around the world thanks to the infrastructure support "*worldwide LHC computing grid*" (WLCG). The WLCG connects more than 140 computing centres in 34 countries with the main objective to support the collection and storage of data and processing tools, simulation and visualization. The idea behind the operation requires that the LHC experimental data are recorded on tape at CERN before being distributed to 11 large computer centres (centres called "*Tier 1*") in Canada, France, Germany, Italy, the Netherlands, Scandinavia, Spain, Taiwan, UK and USA. From these sites, the data is made available to more than 120 centres "*Tier-2*", where you can conduct specific analyses. Individual researchers can then access the information using computer clusters or even their own personal computer.

Cloud Computing: The cloud capability allows to obtain seemingly unlimited storage space and computing power, that it is the reason for which cloud paradigm is considered a very desirable feature in each big data solution [Bryant et al., 2008]. It is a new business where companies and users can rent by using the "as a service" paradigm infrastructure, software, product, processes, etc., Amazon [Amazon AWS], Microsoft [Microsoft Azure], Google [Google Drive]. Unfortunately, these public systems are not enough to extensive computations on large volumes of data, due to the low bandwidth; ideally a Cloud Computing system for big data should be geographically dispersed,

in order to reduce its vulnerability in case of natural disasters, but also should have a high level of interoperability and data mobility. In fact, there are systems that are moving in this direction, such as the OpenCirrus project [Opencirrus Project], an international test bed that allows experiments on interlinked cluster systems.

Self Healing: this feature refers to the capability of a system to autonomously solve the failure problems, for example in the computational process, in the database and storage, and in the architecture. For example, when a server or a node fails, it is important to have the capability of automatically solve the problem to avoid repercussions on the entire architecture. Thus, an automated recovery from failure solution, that may be implemented by means of fault tolerant solutions, balancing, hot spare, etc., and some intelligence is needed. Therefore, it is an important feature for big data architectures, which should be capable to autonomously bypass the problem. Then, once informed about the problems and the performed action to solve it, the administrator may perform an intervention. This is possible for example, through techniques that automatically redirected to other resources, the work that was planned to be carried out by failed machine, which has to be automatically put offline. To this end, there are commercial products which allow setting up distributed and balanced architecture where data are replicated and stored in clusters geographically dispersed, and when a node/storage fails, the cluster can self-heal by recreating the missing data from the damage node, in its free space, thus reconstructing the full capability of recovering from the next problem. On the contrary, the breakdown results and capacity may decrease in the degraded conditions until the storage, processor, resource is replaced [Ghosh et al., 2007].

2.2. DATA MANAGEMENT ASPECTS

In the context of data management, a number of aspects characterize the big data solutions, among them: the maximum size of the database, the data models, the capability of setting up distributed and clustered data management solutions, the sustainable rate for the data flow, the capability of partitioning the data storage to make it more robust and increase performance, the query model adopted, the structure of the database (relational, RDF(Resource Description Framework), reticular, etc.), etc. Considering data structures for big data there is a trend to find a solution using the so called **NoSQL databases** (No SQL, Simple Query Language), even if there are good solutions that still use relational database [Dykstra, 2012]. In the market and from open source solutions, there are several different types of NoSQL databases and rational reasons to use them in different situations, for different kinds of data. There are many methods and techniques for dealing with big data, and in order to be capable to identify the best choice in each case, a number of aspects have to be taken into account in terms of architecture and hardware solutions, because different choices can also greatly affect the performance of the overall system to be built. Related to the database performance and data size, there is the so called CAP Theorem that plays a relevant role [Brewer, 2001], [Brewer, 2012]. The **CAP theorem** states that any distributed storage system for sharing data can provide only two of the three main features: *consistency*, *availability*, and *partition tolerance* [Fox and Brewer,1999]. Property of consistency states that a data model after an operation is still in a consistent state providing the same data to all its clients. The property of availability means that the solution is robust with respect to some internal failure, that is, the service is still available. Partition tolerance means that the system is going to continue to provide service even when it is divided in disconnected subsets, for example a part of the storage cannot be reached. To cope with CAP theorem, big data solutions try to find a trade-off between continuing to issue the service despite of problems of partitioning and at the same time attempting to reduce the inconsistencies, thus supporting the so called eventual consistency.

Furthermore in the context of relational database, the ACID (Atomicity, Consistency, Isolation and Durability) properties describe the reliability of database transactions. This paradigm does not apply to NoSQL database where, in contrast to ACID definition, the data state provides the so called BASE property: Basic Available, Soft state and Eventual consistent. Therefore, it is typically hard to guaranteed an architecture for big data management in a fault-

tolerant BASE way, since, as the Brewer's CAP theorem says, there is no other choice to take a compromise if you want to scale up. In the following, some the above aspects are discussed and better explained.

Database Size: in big data problems, the database size may easily reach magnitudes like hundreds of Tera Byte (TB), Peta Byte (PB) or Exa Byte (EB). The evolution of big data solutions has seen an increment of the amounts of data that can be managed. In order to exploit these huge volumes of data and to improve the productivity of scientific, new technologies and new techniques are needed. The real challenge of database size are the related to the indexing and to the access at the data. These aspects are treated in the following.

Data Model: to cope with huge data sets a number of different data models are available such as Relational Model, Object DB, XML DB or Multidimensional Array model that extend database functionality as described in [Baumann et al., 1998]. Systems like Db4o [Norrie, Grossniklaus and Decurins 2008] or RDF 3X [Schramm, 2012] propose different solutions for data storage can handle structured information or less and the relationships among them. The data model represents the main factor that **influences** the performance of the data management. In fact, the performance of indexing represents in most cases the bottleneck of the elaboration. Alternatives may be solutions that belong to the so called category of NoSQL databases, such as ArrayDBMS [Cattel, 2010], MongoDB [mongoDB], CouchDB [Couchbase], and HBase [Apache HBase], which provide higher speeds with respect to traditional RDBMS (relational database management systems). Belong to the category of NoSQL databases a large set of different kinds of databases, among them:

- Key-value stores: high scalable solution, which allows to obtain good speed in the presence of large lists of elements, such as stock quotes, examples are Amazon Dynamo [Amazon Dynamo], Oracle Berkeley [Oracle Berkeley].
- Wide column stores (big tables): are databases in which the columns are grouped, where keys and values can be composed (as HBase [Apache HBase], Cassandra [Apache Cassandra]). Very effective to cope with time series and with data coming from multiple sources, sensors, device and website, needing high speed. Consequently, they provide good performance in reading and writing operations, while are less suitable for datasets where the data have the same importance of the data relationships.
- Document stores: are aligned with the object-oriented programming, from clustering to data access, have the same behaviour of key-value stores, where the value is the document content. They are useful when data are hardly representable with a relational model due to high complexity; therefore, are used with medical records or to cope with data coming from social networks. Examples are MongoDB [mongoDB], CouchDB [Couchbase].
- Graph databases: they are suitable to model relationships among data. The access model is typically transactional and therefore suitable for applications that need transactions. They are used in fields like geospatial, bioinformatics, network analysis and recommendation engines. The execution of traditional SQL queries is not simple. Examples are:Neo4J [Neo4j], GraphBase [GraphBase], AllegroGraph [AllegroGraph].

Other NoSQL database categories are: object databases, XML databases, multivalue databases, multimodel databases, multidimensional database, etc. [NoSQL DB].

It is therefore important to choose the right NoSQL storage type during the design phase of the architecture to be implemented, considering the different features that characterize the different databases. In other words, it is very important to use the right tool for each specific project, because each storage type has its own weaknesses and strengths.

Resources: the main performance bottlenecks for NoSQL data stores correspond to main computer resources: network, disk and memory performance, and the computational capabilities of the associated CPUs. Typically the big data stores are based on clustering solutions in which the whole data set is partitioned in **clusters** comprised of a numbers of nodes, **cluster size**. The number of nodes in each cluster affects the completion times of each job, because a greater number of nodes in a cluster corresponds to a lower completion time of the job. In this sense, also

the memory size and the **computational capabilities** of each node influence the node performance [DeWitt et al., 2008]. Most of the NoSQL databases use persistent socket connections; while disk is always the slowest component for the inherent latency of non-volatile storages. Thus any high-performance database needs to have some form of memory caching or memory-based storage to optimize the **memory performance**. Another key point is related to the **memory size** and usage of the solution selected. Some solutions, such as HBase [Apache HBase], are considered memory-intensive, and in these cases a sufficient amount of memory on each server/node has to be guaranteed to cover the needs of the cluster that are located in its region of interest. When the amount of memory is insufficient, the overall performance of the system would drastically decrease [Jacobs, 2009]. The **network** capability is an important factor that affects the final performance of the entire big data management. In fact, network connections among clusters make extensive use during read and write operations, but there are also algorithms like Map-Reduce, that in shuffle step make up a high level network usage. It is therefore important to have a highly available and resiliency network, that is also able to provide the necessary redundancy and that could scale well, i.e. it allows the growth of the number of clusters.

Data Organization: the data organization impacts on storage, access and indexing performance of data [Jagadish et al., 1997]. In most cases, a great part of data accumulated are not relevant for estimating results, and thus they could be filtered out and/or stored in compressed size, as well as moved into slower memory along the multitier architecture. To this end, a challenge is to define **rules for arranging and filtering data** in order to avoid/reduce the loss of useful information preserving performances and saving costs [Olston, Jiang and Widom, 2003]. The distribution of data in different remote tables may be the cause of inconsistencies when connection is lost and the storage is partitioned for some fault. In general, it is not always possible to ensure **locally available data** on the node that would process them. It is evident that if this condition is generally achieved, the best performance would be obtained. Otherwise, it would be need to retrieve the missed data blocks, to transfer them and process them in order to produce the results with a high consumption of resources on the node requested them and on the node that owns them, and thus on the entire network; therefore, the time of completion would be significantly higher.

Data Access For Rendering: the activity of data rendering are related to the access of data for representing them to the users, and in some case by performing some pre-rendering processing. The presentation of original or produced data results may be a relevant challenge when the data size is so huge that their processing for producing a representation can be highly computational intensive, and most of the single data would not be relevant for the final presentation to the user. For example, representing at a glance the distribution of 1 billion of economical transactions on a single image would be in any way limited to some thousands of points; the presentation of the distribution of people flows in the large city would be based on the analysis of several hundreds of millions of movements, while their representation would be limited on presenting a map on an image of some Mbytes. A query on a huge dataset may produce enormous set of results. Therefore, it is important to know in advance the their size and to be capable to analyse big data results with scalable display tools, that should be capable to produce a clear vision in a range of cases, from small to huge set of results. For example, the node-link representation of the RDF graph does not provide a clear view of the overall RDF structure: one possible solution to this problem is the use of a 3D adjacency matrix as an alternate visualization method for RDF [Gallego et al., 2011]. Thanks to some graph display tools, it is possible to highlight specific data aspects. Furthermore, it should be possible to guarantee efficient access, perhaps with the definition of standard interfaces especially in business and medical applications on multichannel and multi-device delivering of results without decreasing data availability. An additional interesting feature for data access can be the save of user experience in data access and navigation (parameters and steps for accessing and filtering them).The adoption of semantic queries in RDF databases is essential for many applications that need to produce heterogeneous results and thus in those cases the data rendering is very important for presenting them and their relationships. Other solutions for data access are based on the production of specific indexes, such as Solr [Apache Solr] or in NoSQL databases. An example are the production of faceted results, in which the query results are divided into multiple categories on which the user can further restrict the search results, by composing by using

"and"/"or" different facets/filters. This important feature is present in solutions such as RDF-HDT Library, eXist project [Meier, 2003].

Data Security and Privacy: the problem of data security is very relevant in the case of big data solutions. The data to be processed may contain sensitive information such as EPR, bank data, general personal information as profiles, and content under IPR (intellectual property rights) and thus under some licensing model. Therefore, sensitive data cannot be transmitted, stored or processed in clear, and thus have to be managed in some coded protected format, for example with some encryption. Solutions based on conditional access, channel protection and authentication may still have sensible data stored in clear into the storage. They are called Conditional Access Systems, CAS, and are used to manage and control the user access to services and data (normal users, administrator, etc.) without protecting each single data element via encryption. Most big data installations are based on web services models, with few facilities for countering web threats, whereas it is essential that data are protected from theft and unauthorized accesses. While, most of the present big data solutions present only conditional access methods based on credentials only for accessing the data information, and not to protect them with encrypted packages. On the other hand, content protection technologies are sometimes supported by Digital Rights Management, DRM, solutions that allow to define and execute licenses that formalize the rights that can be exploited on a given content element, who can exploit that rights and at which condition (e.g., time, location, number of times, etc.). The control of the user access rights is *per-se* a big data problem [Bellini, Nesi and Pazzaglia, 2012]. The DRM solutions use authorization, authentication and encryption technologies to manage and enable the exploitation of rights at different types of users; logical control of some users with respect to each single pieces of the huge quantities of data. The same technology can be used to provide contribution to safeguard of the data privacy allowing keeping encrypted the data until they are effectively used by authorized and authenticated tools and users. Therefore, the access to data outside permitted rights and content would be forbidden. Data security is a key aspect of architecture for the management of such big quantities of data, and is excellent to define who can access to what. This is a fundamental feature in some areas such as health/medicine, banking, media distribution and e-commerce. In order to enforce data protection, some frameworks are available to implement DRM and/or CAS solutions exploiting different encryption and technical protection techniques (e.g., MPEG-21 [MPEG-21], AXMEDIS [Bellini et al., 2007], ODRL [Iannella, 2002]). In the specific case of EPR, several millions of patients with hundreds of elements has to be managed; where for each of them some tens of rights should to be controlled, thus resulting in billions of accesses and thus of authentications per day.

2.3. DATA ANALYTICS ASPECTS

Data Analysis aspects have to do with a large range of different algorithms for data processing. The analysis and review of the different data analytics algorithms for big data processing is not in the focus of this chapter that aims at analysing the architectural differences and the most important features of big data solutions. On the other hand, the data analytics algorithms may range on data: ingestion, crawling, verification, validation, mining, processing, transcoding, rendering, distribution, compression, etc., and also for the estimation of relevant results such as the detection of unexpected correlations, detection of patterns and trends (for example of events), estimation of collective intelligence, estimation of the inception of new trends, prediction of new events and trends, analysis of the crowd sourcing data for sentiment/affective computing with respects to market products or personalities, identification of people and folk trajectories, estimation of similarities for producing suggestion and recommendations, etc. In most of these cases, the data analytics algorithms have to take into account of user profiles, content descriptors, contextual data, collective profiles, etc.

The major problems of big data are related to how their "meanings" are discovered; usually this research occurs through complex modelling and analytics processes: hypotheses are formulated, statistical, visual and semantic models are implemented to validate and them and then new hypotheses are formulated again to take deductions, find unexpected correlations, produce optimizations. Also, in several of these cases, the specific data analytics algorithms are based on statistical data analysis; semantic modelling, reasoning and queries; traditional queries; stream and

signal processing; optimization algorithms; pattern recognition; natural language processing; data clustering; similarity estimation; etc.

In the following, key aspects are discussed and better explained.

Data Mining/Ingestion aspects are two key features in the field of big data solutions, in fact in most cases there is a trade-off between the speed of data ingestion, the ability to answer queries quickly and the quality of the data in terms of update, coherence and consistency. This compromise impacts the design of the storage system (i.e., OLTP vs OLAP, On-Line Transaction Processing vs On-Line Analytical Processing), that has to be capable to store and may be to index the new data at the same rate at which they reach the system, also taking into account that a part of the received data could not be relevant for the production of requested results. Moreover, some storage and file-systems are optimized to read and others for writing; while workloads generally involve a mix of both these operations. An Interesting solution is GATE, a framework and graphical development environment to develop applications and engineering components for language processing tasks, especially for data mining and Information Extraction [Cunningham et al., 2002]. Furthermore, the data mining process can be strengthened and completed by the usage of **crawling techniques**, now consolidated in the extraction of meaningful data from web pages richer information, also including complex structures and tags. The processing of a large amount of data can be very expensive in terms of resources used and computation time. For these reasons, it may be helpful to use a distributed approach of crawlers (with additional functionality) who works as distributed system under with a central control unit which manages the allocation of tasks between the active computers in the network [Thelwall, 2001].

Another important feature is the ability to get advanced **faceted** results from queries on the large volumes of available data: this type of queries allow the user to access the information in the store, along multiple explicit dimensions, and after the application of multiple filters. This interaction paradigm is used in mining applications and allows to analyse and browse data across multiple dimensions; the faceted queries are especially useful in e-commerce websites [Ben-Yitzhak O. et al, 2008]. In addition to the features already seen, it is important to take into account the ability to **process data in real-time**: today, in fact, especially in business, we are in a phase of rapid transition; there is also the need to faster reactions, to be able to detect patterns and trends in a short time, in order to reduce the response time to customer requests. This increases the need to evaluate information as soon as an event occurs, that is, the company must be able to answer questions on the fly according to real-time data.

Data Access for Computing: the most important enabling technologies are related to the data modelling and to the data indexing. Both these aspects should be focused on fast access/retrieve data in a suitable format to guarantee high performance in the execution of the computational algorithms to be used for producing results. The **type of indexing** may influence the speed of data retrieval operations at only cost of an increased storage space. Couchbase [Couchbase] offers an incremental indexing system that allows an efficient access to data at multiple points. Another interesting method is the use of Hfile [Aiyer et al., 2012] and the already mentioned Bloom filters [Borthakur et al., 2011]. It consists of an index-organized data file created periodically and stored on disk. However, in big data context there is the need to manage often irregular data, with a heterogeneous structure and do not follow any predefined schema. For these reasons could be interesting the application of an alternative indexing technique suitable for semi-structured or unstructured data as proposed in [McHugh et al., 1998]. On the other hands, for some specific problem the **management of data relationships** is relevant as the data itself or more. This is the case of new data types for social media which are formalized as highly interrelated content for which the management of multi-dimensional relationships in real-time is needed. A possible solution it is to store relationships in specific data structures that ensure good ability to access and extraction in order to adequately support predictive analytics tools. In most cases, in order to guarantee the demanded performance in the rendering and production of data results, a set of pre-computed partial results and/or indexes can be estimated. These pre-computed partial results should be stored into high speed **caches stores**, as **temporary data**. Some kinds of data analytic algorithms create enormous amounts of temporary data that must be opportunely managed to avoid memory problems and to save time for the successive computations. In other cases, however, in order to make some statistics on the information that is accessed more

frequently, it is possible to use techniques to create well-defined cache system or temporary files to optimize the computational process. With the same aim, some **incremental and/or hierarchical algorithms** are adopted in combination of the above mentioned techniques, for example the hierarchical clustering k-means and k-medoid for recommendation [Everitt, Landau and Leese, 2001], [Xui and Wunsch, 2009], [Bellini at al., 2012b].A key element of big data access for data analysis is the presence of **metadata as data descriptors**, i.e. additional information associated with the main data, which help to recover and understand their meaning it the context. In the financial sector, for example, metadata are used to better understand customers, date, competitors and to identify impactful market trends; it is therefore easy to understand that have an architecture that allows the storage of metadata, also represents a benefit for the following operations of data analysis. Metadata structures and organizes information, helping to make it discoverable and accessible; it also makes the data more valuable and more useful to analysts. A variety of attributes can be applied to the data, which may thus acquire greater relevance for users. For example, keyword, temporal, and geospatial information, pricing, contact details and anything else that improves the quality of the information that has been requested. In most cases, the production of suitable data descriptors could be the way to save time in recovering the real full data, since the matching and the further computational algorithms are based on those descriptors rather than on the original data. For example, the identification duplicated documents could be performed by comparing the document descriptors, the production of user recommendations can be performed on the basis of collective user descriptors or on the basis of the descriptors representing the centre of the clusters.

3. OVERVIEW OF BIG DATA SOLUTIONS

In this section, a selection of representative products for the implementation of different big data systems and architectures has been analysed and organized in a comparative table on the basis of the main features identified in the previous sections. To this end, the following paragraphs provide a brief overview of these considered solutions as reported in **Table 1**, described in the next Section.

The **ArrayDBMS** extends database services with query support and a multidimensional array modelling, because big data queries often involve a high number of operations, each of which is applied to a large number of elements in the array. In these conditions, the execution time with traditional database would be unacceptable. In the literature, and from the real applications, a large number of examples are available that use various types of ArrayDBMS, and among them, we can recall a solution that is based on ArrayDBMS Rasdaman [Baumann et al., 1998]: differently from the other types, Rasdaman ArrayDBMS provides support for domain-independent arrays of arbitrary size and it uses a general-purpose declarative query language, that is also associated with an optimized internal execution, transfer and storage. The conceptual model consists of arrays of any size, measures and types of cells, which are stored in tables named collections that contain an OID (object ID) column and an array column. The RaSQL language offers expressions in terms of multidimensional arrays of content objects. Following the standard paradigm *Select-from-where*, firstly the query process gathers collections inspected, then the *"where"* clause filters the array corresponding to the predicate, and finally, the "*Select"* prepares the matrices derived from initial query. Internally, Rasdaman decomposes each object array in "*tiles*" that form the memory access units, the querying units and the processing units. These parts are stored as BLOBs (Binary Large Object) in a relational database. The formal model of algebra for Rasdaman arrays offers a high potential for query optimization. In many cases, where phenomena are sampled or simulated, the results are data that can be stored, searched, and submitted as an array. Typically, the data arrays are outlined by metadata that describe them; for example, geographically referenced images may contain their position and the reference system in which they are expressed.

Couchbase [Couchbase] is designed for real time applications and does not support SQL queries. Its incremental indexing system is realized to be native to JSON (JavaScript Object Notation) storage format. Thus, JavaScript code can be used to verify the document and select which data are used as index key. Couchbase Server is an elastic and

open-source NoSQL database that automatically distributes data across commodity servers or virtual machines, and can easily accommodate changing data management requirements thanks to the absence of a schema to manage. Couchbase is also based on Memcached which is responsible for the optimization of network protocols and hardware, and allows obtaining good performance at the network level. Memcached [Memcached] is an open-source distributed caching system based on main memory, which is specially used in high trafficked websites and high performance demand web applications. Moreover, thanks to Memcached, CouchBase can improve its online users experience maintaining low latency and good ability to scale up to a large number of users. CouchBase Server allows managing in a simple way system updates, which can be performed without sending offline the entire system. It also allows realizing a reliable and highly available storage architecture, thanks to the multiple copies of the data stored within each cluster.

The **db4o** is an object based database [Norrie, Grossniklaus and Decurins 2008] which provides a support to make application objects persistent. It also supports various forms of querying over these objects such as query expression trees and iterator query methods, query-by-example mechanisms to retrieve objects. Its advantages are the simplicity, speed, and small memory footprint.

eXist [Meier, 2003] is grounded on an Open Source project to develop a native XML database system, that can be integrated into a variety of possible applications and scenarios, ranging from web-based applications to documentation systems. The eXist database is completely written in Java and maybe deployed in different ways, either running inside a servlet-engine as a stand-alone server process, or directly embedded into an application. eXist provides schema-less storage of XML documents in hierarchical collections. It is possible to query a distinct part of collection hierarchy, using an extended XPath syntax, or the documents contained in the database. The eXist's query engine implements efficient, index-based query processing. According to path join algorithms, a large range of queries are processed using index information. This database is available solution for applications that deal with both large and small collections of XML documents and frequent updates of them. eXist also provides a set of extensions that allow to search by keyword, by proximity to the search terms, and by regular expressions.

Google Map-Reduce [Yang et al., 2007] is the programming model for processing big data used by Google. Users specify the computation in terms of a map and a reduction function. The underlying system parallelizes the computation across large-scale clusters of machines and is also responsible for the failures, to maintain effective communication and the problem of performance. The Map function in the master node takes the inputs, partitioning them into smaller sub-problems, and distributes them to operational nodes. Each operational node could perform this again, creating a multi-level tree structure. The operational node processes the smaller problems, and returns the response to its parent node. In the Reduce function, the root node takes the answers from the sub-problems and combine them to produce the answer at the global problem is trying to solve. The advantage of Map-Reduce consists in the fact that it intrinsically parallel and thus it allows to distribute processes of mapping operations and reduction. The operations of Map are independent each other, thus can be performed in parallel (with limitations given from the data source and/or the number of CPU/cores near to that data); in the same way, a series of Reduce can perform the reduction step. Then performing the two processes of key values stored, it is possible potentially run queries in real time, very important feature in some work environments.

Hadoop [Hadoop Apache Project] is a framework that allows managing distributed processing of big data across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each of them offering local computation and storage. The Hadoop library is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. Hadoop was inspired from Google's Map-Reduce and Google File System, GFS, and in practical it has been realized to be adopted in a wide range of cases. Hadoop is designed to scan large data set to produce results through a distributed and highly scalable batch processing systems. Hadoop is composed of the Hadoop Distribute File System, HDFS, and of the programming paradigm Map-Reduce [Karloff, Suri and Vassilvitskii, 2010]; thus, it is capable to exploit the redundancy built into the environment. The programming

model is capable to detect failures and solve them automatically by running specific programs on various servers in the cluster. In fact, redundancy provides fault tolerance and capability to self-healing of the Hadoop Cluster. HDFS allows applications to be run across multiple servers, which have usually a set of inexpensive internal disk drives; the possibility of the usage of common hardware is another advantage of Hadoop. A similar and interesting solution is HadoopDB, proposed by a group of researchers at Yale. HadoopDB was conceived with the idea of creating a hybrid system that combines the main features of two technological solutions: parallel databases in performance and efficiency, and Map-Reduce-based system for scalability, fault tolerance, and flexibility. The basic idea behind HadoopDB is to use Map-Reduce as the communication layer above multiple nodes running single-node DBMS instances. Queries are expressed in SQL and then translated into Map-Reduce. In particular, the solution implemented involves the use of PostgreSQL as database layer, Hadoop as communication layer, and Hive as the translation layer [Abouzeid et al., 2009].

Hbase [Aiyer et al., 2012] is a large-scale distributed database build on top of the HDFS, mentioned above. It is a non-relational database developed by means of an open source project. Many traditional RDBMSs use a single mutating B-tree for each index stored on-disk. On the other hand, Hbase uses a Log Structured Merge Tree approach: first collects all updates into a special data structure on memory, and then, periodically, flush this memory on disk, creating a new index-organized data file, the called also Hfile. These indexes are immutable over time, while the several indices created on the disk are periodically merged. Therefore, by using this approach the writing to the disk are sequentially performed. HBase's performance is satisfactory in most cases and may be further improved by using Bloom filters[Borthakur et al., 2011]. Both HBase and HDFS systems have been developed by considering elasticity as fundamental principle, and the use of low cost disks has been one of the main goals of HBase. Therefore, to scale the system results is easy and cheap, even if it has to maintain a certain fault tolerance capability in the individual nodes.

Hive[Apache Hive] is an open-source data warehousing solution based on top of Hadoop. Hive has been designed with the aim of analysing large amounts of data more productively, improving the query capabilities of Hadoop. Hive supports queries expressed in an SQL-like declarative language - HiveQL- to extract data from sources such as HDFS or HBase. The architecture is divided into: Map-Reduce paradigm for computation (with the ability for users to enrich the queries with custom map-reduce scripts), metadata information for a data storage, and a processing part that receives a query from user or applications for execution. The core in/out libraries can be expanded to analyse customized data formats. Hive is also characterized by the presence of a system catalogue (Metastore) containing schemas and statistics, which useful in operations as data exploration, query optimization and query compilation. In Facebook, the Hive warehouse contains tens of thousands of tables and stores over 700TB of data and is being used extensively for both reporting and ad-hoc analyses by more than 200 users per month [Thusoo et al., 2010].

MonetDB [Zhang et al., 2012] is an open-source DBMS for data mining applications. It has been designed for applications with large databases and queries, in the field of Business Intelligence and Decision Support. MonetDB has been built around the concept of bulk processing: simple operations applied to large volumes of data by using efficient hardware, for large scale data processing. At the present, two versions of MonetDB are available and are working with different types of databases: MonetDB/SQL with relational database, MonetDB/XML with an XML database. In addition, a third version is under development to introduce RDF and SPARQL (SPARQL Protocol and RDF Query Language) supports. MonetDB provides a full SQL interface and does not allow a high-volume transaction processing with its multi-level ACID properties. The MonetDB allows performance improvement in terms of speed for both relational and XML databases thanks to innovations introduced at DBMS level, a storage model based on vertical fragmentation, run-time query optimization and of modular software architecture. MonetDB is designed to take advantage of the large amount of main memory and implements new techniques for an efficient support of workloads. MonetDB represents relational tables using the vertical fragmentation (column-stores), storing each column in a separate table, called BAT (Binary Association Table). The left column, usually the OID (object-id) is called the head and the right column, which usually contains the actual attribute valuesis called the tail.

MongoDB [MongoDB] is a document-oriented database that memorizes document data in BSON, a binary JSON format. Its basic idea consists in the usage of a more flexible model, like the "*document*", to replace the classic concept of a "*row*". In fact, with the document-oriented approach, it is possible to represent complex hierarchical relationships with a single record, thanks to embedded documents and arrays. MongoDB is open-source and it is schema free -- i.e., there is no fixed or predefined document's keys -- and allows defining indexes based on specific fields of the documents. In order to retrieve data, ad-hoc queries based on these indices can be used. Queries are created as BSON objects to make them more efficient, and are similar to SQL queries. MongoDB supports MapReduce queries and atomic operations on individual fields within the document. It allows realizing redundant and fault tolerant systems that can easily horizontally scaled, thanks to the sharing based on the document keys and the support of asynchronous and master-slave replications. A relevant advantage of MongoDB are the opportunities of creating data structures to easily store polymorphic data, and the possibility of making elastic cloud systems given its scale-out design, which increases ease of use and developer flexibility. Moreover, server costs are significantly low because MongoDB deployment can use commodity and inexpensive hardware, and their horizontal scale-out architecture, can also reduce storage costs.

Objectivity [Objectivity Platform] is a distributed OODBMS (Object-Oriented Database Management System) for applications that require complex data models. It supports a large number of simultaneous queries and transactions and provides high-performance access to large volumes of physically distributed data. Objectivity manages data in a transparent way and uses a distributed database architecture that allows good performance and scalability. The main reasons for using a database of this type include the presence of complex relationships that suggest tree structures or graphs, and the presence of complex data, that is, when there are components of variable length and in particular multi-dimensional arrays. Other reasons are related to the presence of a database that must be geographically distributed, and which is accessed via a processor grid, or the use of more than one language or platform, and the use of workplace objects. Objectivity has an architecture consisting of a single distributed database, a choice that allows achieving high performance in relation to the amount of data stored and the number of users. This architecture distributes tasks for computation and data storage in a transparent way through the different machines and it is also scalable and has a great availability.

OpenQM [OpenQM Database] is a DBMS that allows developing and run applications which includes a wide range of tools and advanced features for complex applications. Its database model belongs to the family of Multivalue and therefore has many aspects in common with databases Pick-descended and is transactional. The development of applications Multivalue is often faster than using other types of database and this therefore implies lower development costs and easier maintenance. This instrument has a high degree of compatibility with other types of systems with database Multivalue as UniVerse [UniVerse], PI/open, D3 and others.

The **RDF-HDT** (Header-Dictionary-Triples) [RDF-HDT Library] is a new representation format that modularizes data and uses structures of large RDF graphs to get a big storage space, and it is based on three main components: Header, Dictionary and a set of triples. Header includes logical and physical data that describes the RDF dataset; and it is the entry point to the data set. The Dictionary organizes all the identifiers in an RDF graph and provides a catalogue of the amount of information in RDF graph with a high level of compression. The set of Triples, finally, includes the pure structure of the underlying RDF graph and avoids the noise produced by long labels and repetitions. This design gains in modularity and compactness, and addresses other important characteristics: allows access addressed on-demand to the RDF graph and is used to design specific compression techniques RDF (HDT-compress) able to outperform universal compressors. RDF-HDT introduces several advantages like compactness and compression of stored data, using small amount of memory space, communication bandwidth and time. RDF-HDT is modular, since it keeps separates dictionary and triples, and includes a header with metadata about the data. Finally, it allows operations on the data base enabling access to parts of the graph without the needs to process them.

RDF-3X [Schramm, 2012] is an RDF store that implements SPARQL [SPARQL] that achieves excellent performance making a RISC(Reduced Instruction Set Computer) architecture with efficient indexing and query

processing. The design of RDF-3X solution completely eliminates the process of tuning indexes thanks to an exhaustive index of all permutations of subject-predicate-object triples and their projections unary and binary, resulting in highly compressed indexes and in a query processor that can provide data results with excellent performance. The query optimizer can choose the optimal join orders also for complex queries and a cost model that includes statistical synopses for entire join paths. RDF-3X is able to provide good support for online updates efficient, thanks to an architecture staging.

4. COMPARISON AND ANALYSIS OF ARCHITECTURAL FEATURES

A large number of products and solutions have been reviewed for analysing the most interested products for the readers and for the market, while a selection of solutions and products has been proposed in this paper with the aim of representing all of them. The analysis performed has been very complex since a multidisciplinary team has been involved in assessing the several aspects in multiple solutions including correlated issues in case of tools depending on other solutions (as reported in **Table 1**). This is due to the fact that, features of big data solutions are strongly inter-correlated and thus it was impossible to identify orthogonal aspects to provide a simple and easy to read taxonomical representation. Table 1 can be used to compare different solutions in terms of: infrastructure and the architecture of the systems that should cope with big data; data management aspects; and of data analytics aspects, like ingestion, log analysis and everything else is pertinent to post-production phase of data processing. Some of the information related to specific features of products and tools have not been clearly identified. In those cases we preferred to report that the information was not available.

Table 1 – Main features of reviewed big data solutions; where: Y = supported; N = no info; P = partially supported; A = available but supported by means a plug-in or external extension; NA = Not Available.

	ArrayDBMS	Couch Base	Db4o	eXist	Google MapReduce	Hadoop	HBase	Hive	Monet DB	Mongo DB	Objectivity	OpenQM	RdfHdt Library	RDF 3X
Infrastructural and Architectural Aspects														
Distributed	Y	Y	Y	A	Y	Y	Y	Y	Y	Y	Y	NA	A	Y
High Availability	A	Y	Y	Y	Y	Y	Y	Y	P	Y	Y	Y	A	A
Process Management	Computation insensitive	Auto distribution	NA	So high for update of entire files	configurable	configurable	Write intensive		Read dominated (or rapidly change)	Read dominated	More Possibility	Divided among more processes	NA	optimized
Cloud	A	Y	A	Y/A	Y	Y	Y	Y	A	Y	Y	NA	A	A
Parallelism	Y	Y	Transactional	Y	Y	Y	Y	Y	Y	Y	Y	NA	NA	Y
Data Management Aspects														
Data Dimension	100PB	PB	254GB	2^{31}doc.	10PB	10PB	PB	PB	TB	PB	TB	16TB	100mil Triples(TB)	50mil Triples
Traditional / Not Traditional	NoSQL	NoSQL	NoSQL	NoSQL	NoSQL	NoSQL	NoSQL	NoSQL	SQL	NoSQL	XML/ SQL++	NoSQL	NoSQL	NoSQL
SQL Interoperability	Good	SQL-like language	A	A	A	Low	A	Y	Y	JOIN not support	Y (SQL++)	NA	Y	Y
Data Organization	blob	blob 20MB	blob	NA	blob	chunk	A	Bucket	blob	Chunk 16MB	blob	Chunk 32KB	NA	blob
Data Model	Multidim. Array	1 document/ concept (Document Store)	objectDB+ B-tree for index	XML-DB + index tree	Big Table (CF, KV, OBJ, DOC)	Big Table (Column Family)	Big Table (Column Family)	Table - Partitions – Bucket (Column Family)	BAT (Ext SQL)	1Table for each collection (Document DB)	Classic Table in which define +models (GraphDB)	1file/table (data+ dictionary) (MultivalueDB)	3structures ,RDF graph for Header (RDF store)	1Table + permutations (RDF store)
Memory Footprint	reduces	documents + Bidimensional index	Objects + index	Documents + index	NA	NA	Optimized Use	NA	Efficient Use	Document + Metadata	Like RDBMS	No compression	-50% dataset	Less than dataset
Users Access Type	Web-interface	Multiple point	Remote user interface	Web interface, REST interface	Many types of interface	API, common line interface or HDFS-UI web app	Jython or scala interface, rest or thrift gateway	HiveQL queries	Full SQL interfaces	Command Line, Web Interface	Multiple access from different query application, AMS	Console or web application	Access on demand	Web interface (SPARQL)
Data Access Performance	much higher if	speed up access to a	various techniques	NA	NA	NA	NA	Accelerate queries	fast data access	over 50GB,	not provide any	NA	NA	NA

	ArrayDBMS	Couch Base	Db4o	eXist	Google MapReduce	Hadoop	HBase	Hive	MonetDB	MongoDB	Objectivity	OpenQM	RdfHdt Library	RDF 3X
	meta-data are stored in DBMS	document by automatically caching	for optimal data access performance					with bitmap indexes	(MonetDB/XQuery is among the fastest and mostscalable,)	10times faster than MySQL	optimisation for accessing replicas			
Data Analitycs Aspects														
Type of indexing	Multimensional-index	Y(incremental)	B-Tree field indexes	B+-tree (XISS)	Distributed Multileval-tree Indexing	HDFS	h-files	Bitmap Indexing	Hash index	Index RDBMS like	Y (function e objectivity/ SQL++ interfaces)	B-tree based	RDF-graph	Y (efficient triple indexes)
Data relationships	Y	NA	Y	Y	Y	A	Y	NA	Y	Y	Y	Y	Y	Y
Visual rendering and visualization	Y (rView)	NA	Y	NA	P	A	NA	NA	NA	A	A (PerSay)	Y	Y	P
Faceted Query	NA	A	P	Y	A (Lucene)	A (Lucene)	A(filter)	NA	NA	NA	**Y** (Objectivity PQE)	NA	Y	NA
Statistical Analysis Tools	Y	Y/A	A (optional library)	A (JMXClient)	A	Y	Y	y	A (SciQL)	Y (network traffic)	Y	A	A	Y
Log Analysis	NA	Y	NA	NA	Y	Y	P	Y	NA	A	Y	NA	Y	Y
Semantic Query	P	A (Elastic Search)	P	A	A	P (index for semantic Search)	Y	NA	NA	NA	NA	NA	Y	Y
Indexing Speed	more than RDBMS	non-optimal performance	5 – 10 times more than SQL	High speed with B+Tree	Y/A	A	Y	NA	More than RDBMS	High speed if DB dimension doesnot exceed memory	High Speed	Increased speed with alternate key	15 times faster than RDF	aerodynamic
Real Time processing	NA	Indexing + creating view on the fly	more suitable for real time processing of events	NA	A	A (streambased, Hstreaming)	A (HBaseHUT library)	A (Flume + Hive indexes our data and can be queried in real-time)	NA	Y	Y (release 8)	NA	NA	NA

5. APPLICATION DOMAINS COMPARISON

Nowadays, as reported in the introduction, big data solutions and technologies are currently used in many application domains with remarkable results and excellent future prospects to fully deal with the main challenges like data modelling, organization, retrieval, and data analytics. A major investment in big data solutions can lay the foundations for the next generations of advances in medicine, science, research, education and e-learning, business and financial, healthcare, smart city, security, info mobility, social media and networking.

In order to assess different fields and solutions, a number of factors have been identified. In **Table 2**, the relevance of the main features of big data solutions with respect to the most interesting applicative domains is reported. When possible, each feature has been expressed for that applicative domain in terms of relevance by expressing: High, Medium and Low relevance/impact. For some features, to grade was not possible, thus a list of commonly adopted specific solutions and/or technologies has been reported for each specific domain.

The main features are strongly similar to those adopted in Table 1 (the features whose not presented relevant differences in the different domains have been removed to make the table more readable). Table 2 could be considered a key a lecture to compare big data solutions on the basis of the relevant differences. The assessment has been performed according to the state of the art analysis of several big data applications in the domains and corresponding solutions proposed. The application domains should be considered as macro-areas rather than specific scenarios. Despite to the fact that the state of the art of this matter is in continuous evolution, the authors think that the work presented in this chapter can be used as a first step to understand which are the key factors for the identification of the suitable solutions to cope with a specific new domain. This means that the considerations have to be taken as examples and generalization of the analysed cases.

Table 2 can be read line by line. For example, considering the infrastructural aspect of supporting distributed architectures “Distributed management”, at the first glance, one could state that this feature is relevant for all the applicative domains. On the other hand, a lower degree of relevance has been notified for the educational domain. For example, the student profile analysis for the purpose of the personalized courses are typically locally based and are accumulating a lower number of information with respect to global market analysis, security, Social Media, etc. The latter cases are typically deployed planetary cases and information, on multisite distributed databases geographical distributed at worldwide level, while education is usually confined at regional and local levels. A similar consideration can be applied for the High Availability that could be less relevant for Educational and Social Media with respect to the demands of safety critical situations of energy management and of transportation. Moreover, the internal Parallelism of the solution can be an interesting feature that can be fully exploited only in specific cases depending on the data analytics algorithms adopted and when the problems can take advantage from a parallel implementation of the algorithm. This feature is strongly related to the reaction time, for example, in most of the social media solution, the estimation of suggestions and thus of clustering user profiles and content, is something that is performed off line updating values periodically but not in real time.

As regard the Data Management aspects, the amount of data involved is considerable huge in almost all the application domains (in the order of several petabytes and Exabyte). On the other hand, this is not always true for the size of individual files (with the exception of satellite images, medical images or other multimedia files that can also be several gigabytes in size). The two aspects are quite different and in general the former (number of elements) is that create major problems. For this reason, security, social media and smart cities have been considered the applicative domains with higher demand of big data solutions in terms of volume. Moreover, in many cases, the main problem is not their size, rather than the management and the preservation of the relationships among the various elements; they represent the effective semantic value of data set (the data model of Table 1 may help in comparing the solutions). For example, for the user profile (human relationships), traffic localization (service relationships, time relationships), medical patient records (events and data relationships), etc. Data relationships,

often stored in dedicated structures, and making specific queries and reasoning can be very important for some applications such as social media and networking. Therefore, for this aspect, the most challenging domains are again smart cities, social networks and health care. In this regard, in these last two application domains, the use of graph relationships navigation constitutes a particularly useful support to improve the representation, research and understanding of information and meanings explicitly not evident in the data itself.

In almost all domains the usage and the interoperability of both SQL and NoSQL database is very relevant, and some differences can be detected in the data organization. In particular, the interoperability with former SQL is a very important feature in application contexts such as healthcare, social media marketing, business and smart cities, to the widespread use of traditional RDBMS, rather than in application domains such as research scientific, social networks and web data, security and energy, mainly characterized by unstructured or semi-structured data. Some of the applicative domains intrinsically present a large variety and variability of data, while others presents more standardized and regular information. A few of these domains present both variety and variability of data such as the scientific research; security and social media which may involve content based analysis, video processing, etc.

Furthermore, the problems of data access is of particular importance in terms of performances and for the provided features related to the rendering, representation and/or navigation of produced results as visual rendering tools, presentation of results by using faceted facilities, etc. Most of the domains are characterized by the needs of difference custom interfaces for the data rendering (built ad-hoc on the main features) that provide safe and consistent data access. For example in health and scientific research it is important to take account of issues related to the concurrent access and thus data consistency, while in social media and smart cities is important to provide on-demand and multi-device access to information, graphs, real-time conditions, etc. A flexible visual rendering (distributions, pies, histograms, trends, etc.) may be a strongly desirable features to be provided, for many scientific and research applications, as well as for financial data and health care (for example for reconstruction, trend analysis, etc.). Faceted query results can be very interesting for navigating in mainly text based big data as for educational and cultural heritage application domains. Graph navigation among resulted relationships can be an avoidable solution to represent the resulted data in smart cities and social media, and for presenting related implications and facts in financial and business applications. Moreover, in certain specific contexts the data rendering has to be compliant with standards, for example in the health care.

In terms of data analytic aspects, several different features could be of interest in the different domains. The most relevant feature in this area is the type of indexing, which in turn characterizes the indexing performance. The indexing performance are very relevant in the domains in which huge amount of small data have to be collected and need to be accesses and elaborated in the short time, such as in: financial, health care, security and mobility. Otherwise, if the aim of the big data solution is mainly on the access and data processing the fast indexing can be less relevant. For example, the use of HDFS may be suitable in contexts requiring complex and deep data processing, such as the evaluation on the evolution of a particular disease in the medical field, or the definition of a specific business models. This approach, in fact, runs the process function on a reduced dataset thus achieving scalability and availability required for processing big data. In educations, instead, the usage of ontologies and thus of RDF databases and graphs provides a rich semantic structure better than any other method of knowledge representation, improving the precision of search and access for educational contents, including the possibility of enforcing inference in the semantic data structure.

Table 2 – Relevance of the main features of big data solutions with respect to the most interesting applicative domains.

	Data Analysis Scientific Research (biomedical…)	Educational and cultural heritage	Energy/Transportation	Financial/Business	Healthcare	Security	Smart cities and mobility	Social Media Marketing	Social Network Internet Service Web Data
Infrastructural and Architectural Aspects									
Distributed management	H	M	H	H	H	H	M	H	H
High Availability	H	M	H	H	H	H	H	M	H
Internal Parallelism (related to Velocity)	H	M	H	M	H	H	H	M	M
Data Management Aspects									
Data Dimension(data Volume)	H	M	M	H	M	H+	H+	H+	H+
Data Replication	H	L	M	H	H	H	H	H	M
Data Organization	Chuncks, Blob	Blob	Cluster, Blob	Cluster, Chuncks	Blob, Chuncks	Blob	Blob	Chuncks(16 MB), Blob	Chunks, Bucket
Data relationships	H	M	L	H	H	M	H	H	H
SQL Interoperability	M	M	L	H	H	L	H	H	M
Data Variety	H	M	M	M	M	H	H	H	H
Data Variability	H	H	M	H	H	H	H	H	H
Data Access Aspects									
Data Access Performance	H	H	L	M	H	L	H	L	H
Data Access Type	AQL (SQL array version), Full SQL Interfaces, Specific API	Standard Interfaces, Remote User interfaces	Remote user interface, Get API, Multiple Access query	HDFS-UI web app, API, common line interfaces, AMS	API, common line interfaces, concurrency access	Remote user Interfaces, Multiple access from different query application	Interfaces for On-Demand Access, ad-hoc SQL interfaces	API and Customized Interfaces, Command Line, Web Interfaces, AMS	Open ad hoc SQL access (es. HiveQL), Web interfaces, REST interfaces
Visual rendering and visualization	H	M	M	H	H	L	H	M	M
Faceted Query results	L	H	L	H	H	L	L	H	H
Graph relationships navigation	L	M	M	H	M	L	H	H	H

	Data Analysis Scientific Research (biomedical...)	Educational and cultural heritage	Energy/Transportation	Financials	Healthcare	Security	Smart cities and mobility	Social Media Marketing	Social Network Internet Service Web Data
Data Analytics Aspects									
Type of Indexing	Array MultiDimensional, Hash Index	Ontologies, Index RDBMS like,	Key Index Distributed, B-Tree field indexes	HDFS, Index, RDBMS like, B-tree Index	Multidimensional Index, HDFS, Index, RDBMS like,	Distributed multi-level tree indexing, HDFS, Index, RDBMS like,	Hash Index, HDFS, RDF-graph	Distributed Index, RDBMS like,	Inverted Index, MinHash, B+-tree, HDFS, RDF graph
Indexing Speed	L	L	L	H	M	H	H	M	H
Semantic Query	H	M	M	M	H	L	L	M	H
Statistical Analysis Tools in queries	H	H	H	H	H	L	M	H	H
CEP (Active Query)	H	L	M	H	H	M	H	L	H
Log Analysis	L	M	L	H	H	H	H	H	H
Streaming Processing	M	L	M	H	M	H	H	M	H(network monitoring)

The possibility of supporting statistical and logical analysis on data via specific queries and reasoning can be very important for some applications such as social media and networking. If this feature is structurally supported, it is possible to realize direct operations on the data, or define and store specific queries to perform direct and fast statistical analysis: for example for estimating recommendations, firing conditions, etc.

In other contexts, however, it is very important to the continuous processing of data streams, for example, to respond quickly to requests for information and services by the citizens of a "smart city", real time monitoring of the performance of financial stocks or report to medical staff unexpected changes in health status of patients under observation. As can be seen from the table, in these contexts, a particularly significant feature is the use of the approach CEP (Complex Event Processing), based on active query, which improves the user experience with a considerable increase in the speed of analysis; on the contrary, in the field of education, scientific research and transport speed of analysis is not a feature of primary importance, since in these contexts, the most important thing is the storage of data to keep track of the results of experiments or phenomena or situations occurred in a specific time interval, then the analysis is a passage that can be realized at a later time.

Lastly, it is possible to observe **Table 3** in which, the considered application domains are shown in relation to the products examined in the previous session and to the reviewed scenarios. Among all the products reviewed, MonetDB and MongoDB are among the most flexible and adaptable to different situations and contexts of applications. It is also interesting to note that RDF based solutions have been used mainly on social media and networking applications.

Table 3 – Relevance of the main features of big data solutions with respect to the most interesting applicative domains; where: Y = frequently adopted.

	ArrayDBMS	CouchBase	Db4o	eXist	Google MapReduce	Hadoop	HBase	Hive	MonetDB	MongoDB	Objectivity	OpenQM	RdfHdt Library	RDF 3X
Data Analysis Scientific Research (biomedical...)	X		X	X	X				X	X	X	X	X	X
Education and cultural Heritage				X		X		X		X				
Energy/Transportatio n		X	X	X					X		X			
Financial/Business	X	X				X		X	X	X	X			
Healthcare	X		X		X	X	X		X	X	X	X		
Security			X	X							X	Y		
Smart Mobility, smart cities		X	X	X					X	X	X			
Social Marketing	X	X				X	X	X	X	X			X	X
Social Media	X	X			X	X	X	X	X	X			X	X

Shown below the most effective features of each product analysed, and the main application domains in which their use is commonly suggested.

HBase over HDFS provides an elastic and fault tolerant storage solution and provides a strong consistency. Both HBase and HDFS are grounded on the fundamental design principle of elasticity. Facebook Messages [Aiyer et al. 2012] exploits the potential of HBase to combine services such as messages, emails and chat in a real-time conversation, that leads to manage approximately 14TB of messages and 11TB of chats, each month. For these reasons, it is successfully used in the field of social media internet services as well as on social media marketing.

RDF-3X is considerate one of the fastest RDF representations and it provides an advantage with the handling of the small data. RDF-3X physical design completely eliminates the need for index tuning thanks to highly compressed indexes for all permutations of triples and their binary and unary projections. Moreover, RDF-3X is optimized for queries, and provides a suitable support for efficient online updates by means of a staging architecture.

MonetDB achieves a significant speed improvement for both relational/SQL and XML/XQuery databases over other open-source systems; it introduces innovations at all layers of a DBMS: a storage model based on vertical fragmentation (column store), a modern CPU-tuned query execution architecture, automatic and self-tuning indexes, a run-time query optimization, and a modular software architecture. MonetDB is primarily used for the management of the large amount of images. For example, in astronomy, seismology and earth observations. These relevant features collocate MonetDB as on the best tools in the field of scientific research and scientific data analysis, thus defining an interesting technology on which to develop scientific applications and create interdisciplinary platform for the exchange of data in the world community of researchers.

HDT has been proven by experiments to be a good tool for compacting dataset. It allows to compacted more than fifteen times with respect to standard RDF representations; thus, improving parsing and processing while keeping a consistent publication scheme. Thus, RDF-HDT allows to improve compactness and compression, using much less space, thus saving storing space and communication bandwidth. for these reasons, this solution it is especially suited to share data on the web, but also in those contexts that require such operations as data analysis and visualization of results, thanks to the support of 3D Visualization of the RDF Adjacency matrix of the RDF Graph.

eXist's query engine implements efficient index structure to collect data for scientific and academic research, educational assessments and consumption in the energy sector, and its index-based query processing is needed to efficiently perform queries on large document collections. Experiments have moreover demonstrated linear scalability of eXist's indexing, storage and querying architecture. In general, the search expressions using full text index perform better with eXist, than the corresponding queries based on XPath.

In scientific/technical applications, ArrayDBMS is often used in combinations with complex queries, therefore the optimization results are fundamental. ArrayDBMS may be used with both hardware and software parallelisms, which make possible the realization of efficient systems in many fields, such as geology, oceanography, atmospheric sciences, gene data etc.

Objectivity/DB guarantees a complete support for ACID and can be replicated to multiple locations. Objectivity/DB is highly reliable and thanks to the possibility of schema evolution, provides advantages over other technologies that had a difficult time with change/update a field. Thus, it has been typically used for making date insensitive systems or real-time applications, that manipulate the large volumes of complex data. Precisely because of these features the main application fields are the healthcare and financial services, respectively for the real-time management of electronic health records and for the analysis of products with higher consumption, with also the monitoring of sensitive information to support intelligence services.

OpenQM enables the system development with reliability and also provides efficiency and stability. The choice of using OpenQM is usually related to the need for speed, security and reliability and also to the ability of easily build excellent GUI interfaces into the database.

Couchbase is a high-performance and scalable data solution supporting high availability, fault tolerance and data security. Couchbase may provide extremely fast response time. It is particularly suggested for applications developed to support the citizens in the new model of smart urban cities (smart mobility, energy consumption etc.). Thanks its low latency, Couchbase is mainly used in the development of gaming online. And, in applications where to obtain a significant performance improvements is very important, or where the extraction of meaningful information from the large amount of data constantly exchanged is mandatory. For example, in social networks like Twitter, Facebook, Flickr etc.

The main advantage of Hadoop is its ability to analyse huge data sets to quickly spot trends. In fact, most customers use Hadoop together with other types of software like HDFS. The adoption of Google MapReduce provides several benefits: the indexing code is simpler, smaller, and easier to understand, and it guarantees fault tolerance and parallelization. Both Hadoop and Google MapReduce are preferably used in applications requiring large distributed computation. The New York Times, for example, uses Hadoop to process row images and turn them into pdf format in an acceptable time (about 24 hours each 4TB of images). Other big companies exploit the potential of these products: Ebay, Amazon, Twitter and Google itself, that uses MapReduce to regenerate the Google's Index, to update indices and to run various type of analyses. Furthermore, this technology can be used in medical fields to perform large-scale data analysis with the aim of improve treatments and prevention of disease.

Hive significantly reduces the effort required for a migration to Hadoop, which makes it perfect for data warehousing and also it has the ability to create ad-hoc queries, using a jargon similar to SQL. These features make Hive excellent for the analysis of large data sets especially in social media marketing and web application business.

MongoDB provide a relevant flexibility and simplicity, which may reduce development and data modelling time. It is typically used in applications requiring insertion and updating in real time, in addition to a real-time query processing. It allows to define the consistency level that is directly related to the achievable performance. If high performance is not a necessity, it can be possible to obtain maximum consistency, waiting until the new element has been replicated to all nodes. MongoDB uses internal memory to store the working set, thus allowing faster access of data. Thanks to its characteristics, MongoDB is easily usable in Business and in Social Marketing fields and, it is actually successfully used in Gaming environment, thanks to its high performance for small operations of read/write. As many other bid data solutions, it is well suited for applications that handled high volumes of data where traditional DBMS might be too expensive.

Db4o does not need a mapping function between the representation in memory and what actually is stored on disk, because the applications schema corresponds with the data schema. This advantage allows to obtain better performance and good user experience. Db4o also permits to database access by using simple programming language (Java, .NET etc.), and thanks to its type safety, does not need to hold in check query against code injection. Db4o supports the paradigm CEP (see Table 2), and it is therefore very suitable for medical applications, scientific research, analysis of financial and real-time data streams, in which the demand for this feature is very high.

6. CONCLUSIONS

We have entered an era of big data. There is the potential for making faster advances in many scientific disciplines though better analysis of these large volumes of data, and also for improving the profitability of many enterprises. The need for these new-generation data management tools is being driven by the explosion of big data and by the rapidly growing volumes and variety of data that are collecting today from alternative sources such as social networks like Twitter and Facebook.

NoSQL Database Management Systems represents a possible solution to these problems, but unfortunately they are not a definitive solutions: these tools have a wide range of features, that can be further developed to create new products more adaptable to this huge stream of data constantly growing and to its open challenge like error-

handling, privacy, unexpected correlations detection, trend analysis and prediction, timeliness analysis and visualization. Considering this latter challenge, it is clear that, in a fast-growing market for maps, charts and other ways to visually sort through data, these larger volumes of data and analytical capabilities become the new coveted features; today, in fact in the “Big Data world”, static bar charts and pie charts just do not make more sense, more and more companies are demanding more dynamic, interactive tools and methods for line-of-business managers and information workers for viewing, understanding and operating on the analysis of big data.

Each product compared in this review, presents different features that may be needed in different situations with which we are dealing. In fact, there is still no definitive ultimate solution for the management of big data. The best way to determine on which product to base the development of your system may consist in analysing the available datasets carefully and determine what are the requirements to which you cannot give up. Then, an analysis of the existing products is needed to determine pros and cons, also considering other non-functional features as the programming language, the integration aspects, the legacy constraints, etc.

7. REFERENCES

[Abouzeid et al., 2009] Abouzeid A.; Bajda-Pawlikowski C.; Abadi D.; Silberschatz A.; Rasin A., *"HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads"*, Proceedings of the VLDB Endowment, Pages 922-933, Volume 2, Number 1, August 2009.

[Aiyer et al., 2012] Aiyer A.; Bautin M.; Jerry Chen G., Damania P.; Khemani P.; Muthukkaruppan K.; Ranganathan K.; Spiegelberg N.; Tang L.; Vaidya M., *"Storage Infrastructure Behind Facebook Messages Using HBase at Scale"*, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Pages 4-13, Volume 35, Number 2, June 2012.

[AllegroGraph] AllegroGraph - http://www.franz.com/agraph/allegrograph/

[Amazon AWS] Amazon AWS - http://aws.amazon.com/

[Amazon Dynamo] Amazon Dynamo - http://aws.amazon.com/dynamodb/

[Antoniu et al., 2010] Antoniu G.; Bougè L.; Thirion B.; Poline JB., "*AzureBrain: Large-scale Joint Genetic and Neuroimaging Data Analysis on Azure Clouds* ", 30 September 2010.

[Apache Cassandra] Apache Cassandra - http://cassandra.apache.org/

[Apache HBase] Apache HBase -http://hbase.apache.org/

[Apache Hive] Apache Hive - http://hive.apache.org/

[Apache Solr] Apache Solr - http://lucene.apache.org/solr/

[Baumann et al., 1998] Baumann P.; Dehmel A.; Furtado P.; Ritsch R.; "*The multidimensional database system RasDaMan*", SIGMOD '98 Proceedings of the 1998 ACM SIGMOD international conference on Management of data, Pages 575-577, ISBN:0-89791-995-5, June 1998

[Bellini, Cenni and Nesi, 2012] Bellini P., Cenni D.; Nesi P., *"On the Effectiveness and Optimization of Information Retrieval for Cross Media Content"*, Proceeding of the KDIR 2012 is part of IC3K 2012, International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Barcelona, Spain, 4-7 October 2012.

[Bellini at al., 2012a] Bellini P., I. Bruno, D. Cenni, A. Fuzier, P. Nesi, M. Paolucci, "Mobile Medicine: Semantic Computing Management for Health Care Applications on Desktop and Mobile Devices", Multimedia Tools and Applications, Springer, Vol.58, n.1, pp.41-79, May 2012. DOI 10.1007/s11042-010-0684-y. http://link.springer.com/article/10.1007/s11042-010-0684-y .

[Bellini at al., 2012b] Bellini P.; Bruno I.; Cenni D.; Nesi P., *“Micro Grids for Scalable Media Computing and Intelligence in Distributed Scenarios”*, IEEE MultiMedia, Pages 69 - 79, Volume 19, Number 2, February 2012.

[Bellini at al., 2007] Bellini P.; Bruno I.; Nesi P.; Rogai D., *"Architectural solution for interoperable content and DRM on multichannel distribution*", Proc. Of the International Conference on Distributed Multimedia Systems, DMS 2007, San Francisco Bay, USA, Organised by Knowledge Systems Institute, September 6-8, 2007.

[Bellini, Nesi and Pazzaglia, 2012] Bellini, P., P. Nesi, F. Pazzaglia, “Exploiting P2P Scalability for Grant Authorization in Digital Rights Management Solutions”, submitted on Multimedia Tools and Applications, Springer, 2012.

[Ben-Yitzhak O. et al, 2008] Ben-Yitzhak O., Golbandi N., Har'El N, Lempel R., Neumann A., Ofek-Koifman S., Sheinwald D., Shekita E., Sznajder B., Yogev S., *"Beyond Basic Faceted Search"*, Proc. of the 2008 International Conference on Web Search and Data Mining , Pages 33-44, 2008

[BlueGene] BlueGene IBM project, *http://www.research.ibm.com/bluegene/index.html*

[Borthakur et al., 2011] Borthakur D.; Muthukkaruppan K.; Ranganathan K.; Rash S.; SenSarma J.; Spielberg N.; Molkov D.; Schmidt R.; Gray J.; Kuang H.; Menon A.; Aiyer A., "*Apache Hadoop Goes Realtime at Facebook*", Proceedings of the 2011 International Conference on Management of Data, Athens, Greece, June 2011.

[Bose et al., 2001] Bose I.; Mahapatra R.K., *"Business Data Mining - a Machine Learning Prespective"*, Information & Management, Pages 211-225, Volume 39, Number 3, December 2001.

[Brewer, 2012] Brewer E., *"CAP Twelve Years later: How the Rules Have Changed"*, IEEE Computer, Pages 23-29, February 2012.

[Brewer, 2001] Brewer E., *"Lesson from Giant-Scale Services"*, IEEE Internet Computing, Pages 46-55, July/Aug 2001.

[Bryant et al., 2008] Bryant R.; Katz R.H.; Lazowska E.D.; "Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society", In Computing Research Initiatives for the 21st Century, Computing Research Association, 2008.

[Bryant et al., 2010] Bryant R.E.; Carbonell J.G.; Mitchell T., *"From Data to Knowledge to Action: Enabling Advanced Intelligence and Decision-Making for America's Security"*, Computing Community Consortium, Version 6, July 2010.

[Cao, Wang and Xiong, 2009] Cao L.; Wang Y.; Xiong J., *"Building Highly Available Cluster File System Based on Replication"*, International Conference onParallel and Distributed Computing, Applications and Technologies,Pages 94-101, December 2009.

[Cattel, 2010] Cattel R., " Scalable SQL and NoSQL Data Stores", ACM SIGMOND Record, Pages 12-27, Volume 39, Number 4, December 2010.

[Couchbase] Couchbase - *http://www.couchbase.com/*

[Cunningham et al., 2002] Cunningham H.; Maynard D.; Bontcheva K.; Tablan V., *"GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications"*, Proceeding of the 40th Anniversary Meeting of the Association for Computational Linguistics, 2002.

[DeWitt et al., 2008] DeWitt D.J.; Paulson E.; Robinson E.; Naugton J.; Royalty J.; Shankar S.; Krioukov A., *"Clustera: an integrated computation and data management system"*, Proceedings of the VLDB Endowment, Pages 28-41, Volume 1, Number 1, August 2008.

[DeWitt et al., 2012] DeWitt S.; Sinclair R.; Sansum A.; Wilson M., *"Managing Large Data Volumes from Scientific Facilities"*, ERCIM News, Page 15, Number 89, April 2012.

[Domingos, 2005] Domingos P.; *"Mining Social Networks for Viral Marketing"*, IEEE Intelligent Systems, Pages 80-82, Volume 20, Number 1, 2005.

[Dykstra, 2012] Dykstra D., *"Comparison of the Frontier Distributed Database Caching System to NoSQL Databases"*, Computing in High Energy and Nuclear Physics (CHEP) Conference, May 2012

[Eaton et al., 2012] Eaton C.; Deroos D.; Deutsch T.; Lapis G., *"Understanding Big Data: Analytics for enterprise class Hadoop and streaming Data"*, Mcgraw-HillPubl.Comp., ISBN: 978-0071790536, March 2012

[ECLAP] ECLAP: http://www.eclap.eu

[Europeana] Europeana Portal - http://www.europeana.eu/portal/

[Everitt, Landau and Leese, 2001] Everitt, B., Landau, S., Leese, M., "Cluster analysis," 4th edition, London, Arnold, 2001.

[Figueireido, Rodrigues and Vale, 2005] Figueireido V.; Rodrigues F.; Vale Z., *"An Electric Energy Consumer Characterization Framework Based on Data Mining Techniques"*, IEEE Transactions on Power Systems, Pages 596-602, Volume 20, Number 2, May 2005.

[Foster et al., 2002] Foster, I., C. Kesselman, J. M. Nick, S. Tuecke, "Grid Services for Distributed System Integration", IEEE Computer, June 2002.

[Fox and Brewer, 1999] Fox A.; Brewer E.A., *"Harvest, Yield, and Scalable Tolerant Systems"*, Proceedings of the Seventh Workshop on Hot Topics in Operating Systems, Pages 174-178, March 1999

[Gallego et al., 2011] Gallego M.A.; Fernandez J.D.; Martinez-Prieto M.A.; De La Fuente P., " *RDF Visualization using a Three-Dimensional Adjacency Matrix*", 4th International Semantic Search Workshop (SemSearch), March 2011.

[Ghosh et al., 2007] Ghosh D.; Sharman R.; Rao H. R.; Upadhyaya S.; *"Self-healing systems - survey and synthesis"*, Decision Support Systems, Pages 2164–2185, Volume 42, Number 4, January 2007.

[Google Drive] Google Drive - http://drive.google.com
[GraphBase] GraphBase - http://graphbase.net/
[Gulisano et al., 2012] Gulisano V.; Jimenez-Peris R.; Patino-Martinez M.; Soriente C.; Valduriez P., *"A Big Data Platform for Large Scale Event Processing"*, ERCIM News, Pages 32-33, Number 89, April 2012.
[Hadoop Apache Project] Hadoop Apache Project - http://hadoop.apache.org/
[Hanna, 2004] Hanna M., *"Data Mining in the e-learning domain"*, Campus-Wide Information Systems, Pages 29-34, Volume 21, Number 1, 2004.
[Iaconesi and Persico, 2012] Iaconesi S.; Persico O., *"The Co-Creation of the City, re-programming cities using real-time user generated content"*, 1st Conference on Information Technologies for Performing Arts, Media Access and Entertainment, May 2012.
[Iannella, 2002] Iannella, R., "Open Digital Rights Language (ODRL)", Version 1.1 W3C Note, 19 September 2002, http://www.w3.org/TR/odrl
[Jacobs, 2009] Jacobs A., *"The pathologies of big data"*, Communications of the ACM - A Blind Person's Interaction with Technology, Pages 36-44, Volume 52, Number 8, August 2009.
[Jagadish et al., 1997] Jagadish H.V.; Narayan P.P.S.; Seshadri S.; Kanneganti R.; Sudarshan S., *"Incremental Organization for Data Recording and Warehousing"*,Proceedings of 23rd International Conference on Very Large Data Bases, Pages 16-25, August 1997.
[Karloff, Suri and Vassilvitskii, 2010] Karloff H.; Suri S.; Vassilvitskii S., *"Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms"*, Pages 938-948, 2010.
[LAWA Project] LAWA Project - http://www.lawa-project.eu/
[LHC] LHC - http://public.web.cern.ch/public/en/LHC/LHC-en.html
[Liu, Biderman and Ratti, 2009] Liu L.; Biderman A.; Ratti C., "*Urban mobility landscape: Real time monitoring of urban mobility patterns"*, Proceedings of the 11th International Conference on Computers in Urban Planning and Urban Management, 2009.
[Mans et al., 2009] Mans R. S.; Schonenberg M. H.; Song M.; Van der Aalst W.M.P.; Bakker P.J.M., *"Application of Process Mining in Healthcare - A Case Study in a Dutch Hospital"*, Biomedical Engineering Systems and Technologies, Communications in Computer and Information Science, page 425-438, Volume 25, Part 4, 2009
[McHugh et al., 1998] McHugh J.; Widom J.; Abiteboul S.; Luo Q.; Rajaraman A., "Indexing semistructured data", Technical report, Stanford University CA, February 1998.
[Meier, 2003] Meier W., "*eXist: An Open Source Native XML Database*", Web, Web-Services, and Database Systems - Lecture Notes in Computer Science, page 169-183, Volume 2593/2003, 2009
[Memcached] Memcached - *http://memcached.org/*
Microsoft Azure - http://www.windowsazure.com/it-it/
[Microsoft, 2012] Microsoft *Microsoft private cloud. Tech. rep., 2012*
[Mislove, Gummandi and Druschel, 2006] Mislove A.; Gummandi K.P.; Druschel P., *"Exploiting Social Networks for Internet Search"*, Record of the Fifth Workshop on Hot Topics in Networks: HotNets V, Pages 79-84, August 2006.
[MongoDB] MongoDB - http://www.mongodb.org/
[MPEG-21] MPEG-21, http://mpeg.chiariglione.org/standards/mpeg-21/mpeg-21.htm
[Neo4J] Neo4J - *http://neo4j.org/*
[Norrie, Grossniklaus and Decurins 2008] Norrie M.C.; Grossniklaus M.; Decurins C., *" Semantic Data Management for db4o"*, Proceedings of 1st International Conference on Object Databases (ICOODB 2008), page 21-38, March 2008.
[NoSQL DB] NoSQL DB - *http://nosql-database.org/*
[Obenshain, 2004] Obenshain M.K., *"Application of Data Mining Techniques to Healthcare Data"*, Infection Control and Hospital Epidemiology, Pages 690-695,Volume 25, Number 8, 2004.
[Objectivity Platform] Objectivity Platform - *http://www.objectivity.com*
[Olston, Jiang and Widom, 2003] Olston C.; Jiang J.; Widom J., *"Adaptive filters for continuous queries over distributed data streams"*, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, Pages 563-574, 2003
[OpenCirrus Project] OpenCirrus Project - https://opencirrus.org/
[OpenQM Database] OpenQM Database - http://www.openqm.org/docs/
[OpenStack Project] OpenStack Project– *http://*www.openstack.org
[Oracle Berkley] Oracle Berkley - http://www.oracle.com/technetwork/products/berkeleydb/

[Pierre et al., 2011] Pierre G.; El Helw I.; Stratan C.; Oprescu A.; Kielmann T.; Schuett T.; Stender J.; Artac M.; Cernivec A., "*ConPaaS: an Integrated Runtime Environment for Elastic Cloud Applications*", ACM/IFIP/USENIX 12th International Middleware Conference, December 2011

[RDF-HDT Library] RDF-HDT Library - http://www.rdfhdt.org

[Rivals et al., 2012] Rivals E.; Philippe N.; Salson M.; Léonard M.; Commes T.; Lecroq T., *"A Scalable Indexing Solution to Mine Huge Genomic Sequence Collections"*, ERCIM News, Pages 20-21, Number 89, April 2012

[Rusitschka, Eger and Gerdes, 2010] Rusitschka S.; Eger K.; Gerdes C., *"Smart Grid Data Cloud: A Model for Utilizing Cloud Computing in the Smart Grid Domain"*, 1st IEEE International Conference of Smart Grid Communications, October 2010

[SCAPE] SCAPE Project - http://scape-project.eu/

[Schramm, 2012] Schramm M., "*Performance of RDF Representations*", 16th TSConIT, January 2012

[Silvestri et al., 2012] SilvestriLudovico (LENS), Alessandro Bria (UCBM), Leonardo Sacconi (LENS), Anna Letizia Allegra Mascaro (LENS), Maria Chiara Pettenati (ICON), SanzioBassini (CINECA), Carlo Cavazzoni (CINECA), Giovanni Erbacci (CINECA), Roberta Turra (CINECA), Giuseppe Fiameni (CINECA), Valeria Ruggiero (UNIFE), Paolo Frasconi (DSI - UNIFI), Simone Marinai (DSI - UNIFI), Marco Gori (DiSI - UNISI), Paolo Nesi (DSI - UNIFI), Renato Corradetti (Neuroscience - UNIFI), GiulioIannello (UCBM), Francesco SaverioPavone (ICON, LENS), *"Projectome: Set up and testing of a High Performance ComputationalInfrastructure for processing and visualizing neuro-anatomical information obtainedusingconfocal ultra-microscopytechniques"*, Neuroinformatics 2012 5th INCF Congress, September 2012

[Snell, 2011] Snell A., *"Solving Big Data Problems with Private Cloud Storage"*, White paper October 2011

[SPARQL] SPARQL at W3C - http://www.w3.org/TR/rdf-sparql-query/

[Thelwall, 2001] Thelwall M., *"A web crawler design for data mining"*, Journal of Information Science, Pages 319-325, Volume 27, Number 1, October 2001

[Thusoo et al., 2010] Thusoo A.; Sarma J.S.; Jain N.; Zheng Shao; Chakka P.; Ning Zhang; Antony S.; Hao Liu; Murthy R., *" Hive - A Petabyte Scale Data Warehouse Using Hadoop"*, IEEE 26th International Conference on Data Engineering (ICDE), Pages 996-1005, March 2010

[UniVerse] UniVerse - http://u2.rocketsoftware.com/products/u2-universe

[Valduriez, Pacitti, 2005] Valduriez P.; Pacitti E., *"Data Management in Large-scale P2P Systems"*, High Performance Computing for Computational Science Vecpar 2004 - Lecture Notes in Computer Science, Volume 3402, 2005

[Woolf, Baker and Gianchandani, 2010] Woolf B.P.; Baker R.; Gianchandani E.P., *"Enabling Personalized Education"*, Computing Community Consortium, Version 9, September 2010

[Xui and Wunsch, 2009] Xui R., Wunsch D. C. II, "Clustering", John Wiley and sons, USA, 2009.

[Yang et al., 2007] Yang H.; Dasdan A.; Hsiao R.L.; Parker D.S., *"Map-reduce-merge: simplified relational data processing on large clusters"*, SIGMOD '07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data, page 1029-1040, ISBN: 978-1-59593-686-8, June 2007

[Yang, Tang and Zhou, 2011] Yang J.; Tang D.; Zhou Y.; *"A Distributed Storage Model for EHR Based on HBase"*, International Conference on Information Management, Innovation Management and Industrial Engineering, November 2011

[Zenith] Zenith - http://www-sop.inria.fr/teams/zenith/

[Zhang et al., 2012] Zhang Y.; Kersten M.; Ivanova M.; Pirk H.; Manegold S., *"An implementation of ad-hoc array queries on top of MonetDB"*, TELEIOS FP7-257662 Deliverable D5.1, February 2012.

[Zinterhof, 2012] Zinterhof P., *"Computer-Aided Diagnostics"*, ERCIM News, Page 46, Number 89, April 2012.