¡¡DELIVERABLE UNDER REVIEW PROCESS!!

VELaSSCo

Visual Analysis for Extremely Large-Scale Scientific Computing

D2.1 – State-of-the-art of Big Data

Version 1

Deliverable Information

Grant Agreement no	619439
Web Site	http://www.velassco.eu/
Related WP & Task:	WP 2, D2.1, T2.1
Due date	June 30, 2014
Dissemination Level	
Nature	
Author/s	Benoit Lange, Toàn Nguyên,
Contributors	Olav Liestol

*The research leading to these results has received funding from the European Community's Seventh Framework Programme managed by REA-Research Executive Agency http://ec.europa.eu/research/rea [FP7/2007-2013] under grant agreement n° **619439***

Approvals

	Name	Institution	Date	OK
Author	Benoit Lange Toàn Nguyên	**INRIA** **INRIA**	30/06/2014	
Task Leader	Toàn Nguyên	**INRIA**		
WP Leader	Toàn Nguyên	**INRIA**		
Coordinator				

Change Log

Version	Description of Change
Version 0	First version of the document
Version 1	Major update

Table of Contents

1. Introduction

As stated in [20], scientists hope to reach Exascale computing in 2015. Exascale computers are machines, which can reach one exaFlops (10^{18} floating point operations per second). Such amount of computations brings new kinds of problems for large-scale computing. To be able to reach Exascale computing, scientists have to deal with massive parallel architectures and apply computations on heterogonous nodes (variable granularity of computation). Such a paradigm is a shift in mind: traditional parallelism is not enough to express Exascale requirements. ExaFlops are expected to be reached in 2018; to achieve this goal new hardware is developed, see [108].

Storage of Data is another important field of research in computer science. Data growing scale has followed a similar path than computing (based on Moore's law) [28]: quantity doubles every 2 years, see Figure 1. This fast growing is driven by two evolutions: personal computers and Internet access. Data is produced by everything; any object from our life can now produce information. Moreover, data is now collected through many communication channel. To manage efficiently this amount of information, it is necessary to produce metadata, this new information is used to explain existing inforamtion.

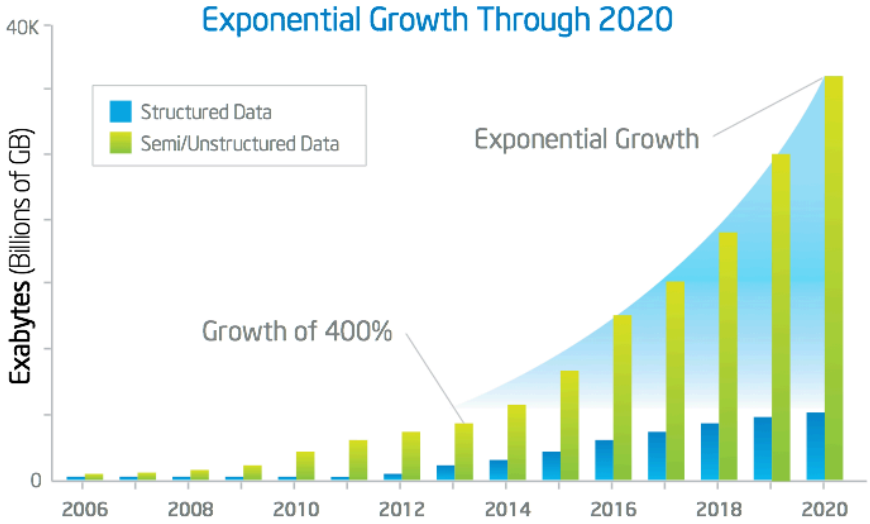


Figure 1. Current and forecasted growth of big data. [220]

As an example, LSST produce 30 terrabytes of astrophysics data every night [22]. In 2008, servers around the world have produced 9.57 zettabytes (10^{21} bytes) of information [27]. Massive data brings new computation dilemmas, but it is not a new problem: the first processing equipment was deployed in 1890; this system used punched cards to store information. Now, compute systems have evolved and are massively distributed to perform fast parallel computations. This intensive data growth has an energy impact: data analysis and management needs more energy [26]. In the case of the VELaSSCo project, this amount of data is produced by simulation software. For example, the solver computes a particles simulation on a set of particles and produces files. These files are composed of timestamps and new values for particles. When a new time step is computed, a new file is generated. To understand a simulation, it is necessary to store the maximum number of time steps, this statement imply to deal with large amount of information: Big Data.

In the earlier days, the term Big Data has been proposed to refer to large datasets, which required supercomputing capabilities [14][29]. But after a while, these datasets could fit into a standard computer, and analysis could be performed on a desktop computer. From this statement this definition of Big Data is considered as a poor term [13]. A new definition has been provided: Big Data is more correlated to capacities of search, aggregate and cross-reference datasets: the number of datasets is less important. Data processing needs to address three key points, named 3V: Volume, Velocity and Variety. The term 3V is presented in [11][29]. The data volume concerns the amount of produced data (byte quantity). Rate and frequency of the data production represent the data velocity. And finally, data variety means which data is produced. Now storage systems need to handle with these three dimensions [12]. To be able to manage these datasets, data scientists need to provide new ODP (Organize, Package and Deliver data) strategies to databases [12].

An important event brings Big Data in the front part of the scientific community [159]. Google Scientists have produced a paper published in Nature, which presents the dissemination of flu across USA in 2008. This analysis has been performed without any medical knowledge; only search engine queries have been used. Five years later, the same analysis was performed again, and then the results have diverged from the reality, the flu was overstated by a factor of two. This example shows an important thing with Big Data: the result is not always true. In 2010, the economist has published a paper entitles: “the data deluge”, see [230], in this paper, authors present business development of this new domain. Big data is now an evolving domain, which surround all domains. This document referees to most common problems with big data: the amount of information, transformation of traditional business, data visualization, etc. This paper presents what big data means, and what is evolving with big data.

In this document, the first part is dedicated to distribute systems (Section 2). The second part presents cloud infrastructures (Section 3). The third part is devoted to best practices concerning Big Data (Section 4). The fourth part presents some cloud tools (Section 5) and the following part discusses some applications (Section 6). The next section discusses the future of cloud systems (Section 7). The last section focuses on the VELaSSCo project (Section 8).

2. Distributed Systems

A distributed system (DS) is a part of the High Performance Computing field, and HPC is a part of the computer science domain. A distributed system is focused on how to deal with distributed components over networks. Computation is coordinated by a message passing system or job scheduling. DS can be divided into several sub-domains as stated in [1]. These subdomains are: Supercomputing, Grids, Clusters, Web 2.0 and clouds. Figure 2 shows this decomposition, on abscissa, systems are classified between Application and Services orientation, and the ordinate axis is dedicated to describe scale (size of computing system). Subclasses of DS are disseminated on this 2D space and some overlapping exists between these subdomains.

In this document, our interest is focused on grid and cloud computing.

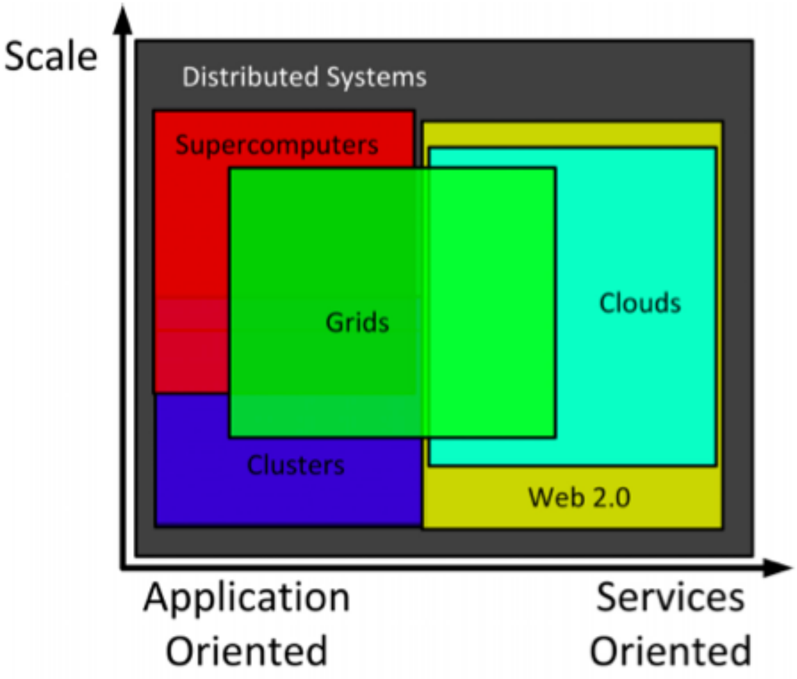


Figure 2. Global overview of Distributed Systems [1]

2.1. Supercomputing

To reduce the scope of this document, we make a short introduction to this subdomain. And more precisely, we deal with the BigTable approach presented by Google. The presented solution cannot be defined as a cloud or grid computing.

Google has developed this solution to index the Web, and provide an efficient WebCrawler. This solution is based on 3 components: GFS [215], BigTable [56] and MapReduce. The main component is GFS (Google File System). It is a virtual distributed File System (FS), which has been designed to fit efficiently to web crawl requirements (small amounts of writes, huge amounts of reads). GFS has been designed to be suitable with a highly distributed system. This system works with two different nodes: Master and Chunkservers. This solution provides a high reliability and high redundancy with at least 3 instances of the same data chunks. Then Google has provided a database system named BigTable. This DB is based on GFS, and was a data column store. All the computation is based on MapReduce. This programming model is a distributed computation model based on two functions Map and Reduce. Figure **3** presents the MR paradigm. Multiple Map functions are executed on several nodes. Results of the Map function are written in files, and the Reduce functions merge these files into other files. The original MR implementation is only suitable for batch processing; interactive processing is not suitable.

Scientists have developed an open-source framework named Hadoop and based on Java [95][94][107]. This strategy enables to deploy a reliable, scalable and distributed application among several computers. Instead of using Google File Systems, Scientists have developed a specific file system named HDFS (Hadoop File System). This File System enables similar functionalities than GFS. This system can store any data, without any schema requirements. To apply computation among several nodes, they also implement MapReduce. As an open source framework, this solution has massively evolved, and a plethora of extensions has been proposed and implemented. In the next section of this document, we present some of them. An example of these extensions is presented in [142]: MongoDB can be used to store Hadoop data instead of HDFS.

2.2. Grid computing

Grids are part of distributed systems (see Figure 2). Grids provide a set of resources by sharing and coordinating them. Grids architectures can be composed by different kinds of resources, hardware (x86, SPARC, ARM, etc.) and operating systems. All these features make Grids heterogeneous. Grids are also shared between users; each one has a specific amount of time and resources. And grids can be distributed among several locations. When a scientist has to deal with a grid system, he has to think with a possible heterogeneity of the hardware, shared resources (resources are not always available). Resources of a Grid can be distributed among countries (this implies specific network issue). An example of a grid cluster is named: GRID 5000 (France) [2]. This system is distributed among 10 sites and connected by a specific high-speed French network. This grid system provides to scientists a highly configurable, reconfigurable and manageable platform.

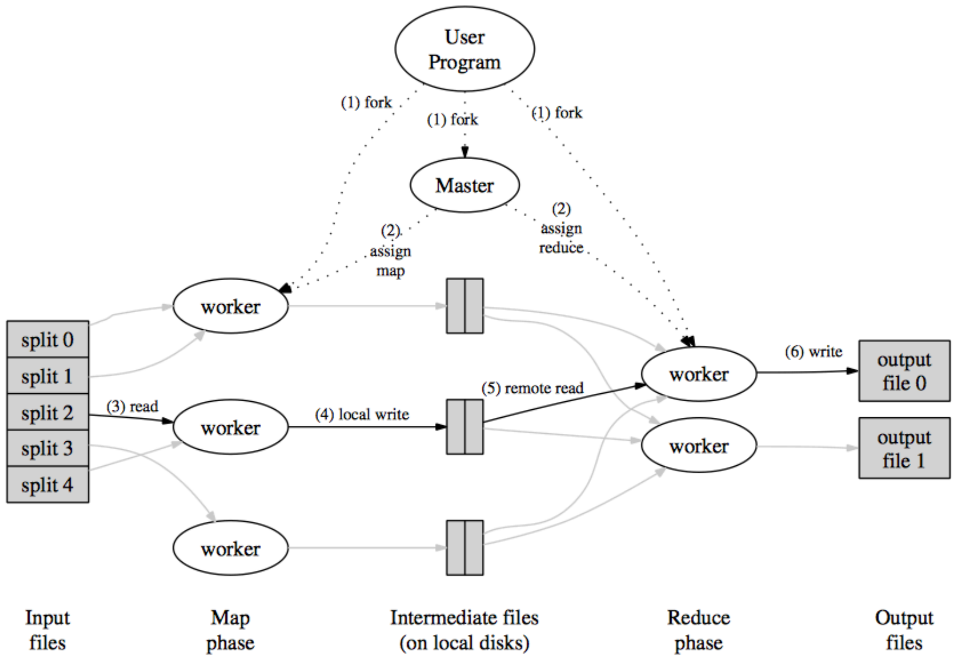


Figure 3. MapReduce execution workflow

In [3], authors describe the requirements, of grid systems: the coordination of resources with a standard protocol. They also present the most important problem with this system: fluctuant quality of service (due to the distribution and failure of resources). To manage Grid resources, a centralized system is used; and a specific protocol to enable communication among nodes. The distributiveness of the Grid system is the most complex task; it is not easy to deal with this problem. From a developer point of view, a grid system can be assimilated as a very large super computer. These super computers are virtual, because nodes are not located at the same location. Compared to a traditional super computer, network is the critical point of such systems.

Grid systems have a specific business model; it is a project-oriented model. The cost unit is represented by CPU hours, allocated resources and wall time. To manage these resources, GRID systems use batch submission software: PBS (Portable Batch System), Torque, SGE, etc. Grid can deal with both low hardware level and virtualized environment. Virtualized environments are used to reduce the complexity to deploy application on this system.

The next section presents an overview of cloud architectures; we present a comparison with grid because they share lot of common points. As stated in [1], the differences between both architectures are not so easy to understand.

2.3. Clouds

In 1961, John McCarthy said, "computation may someday be organized as a public utility" [10][9]. This quote lays the first milestone for cloud computing. With this paradigm, computation is brought outside of the desktop computer. Running fewer larger clusters is cheaper than running more small clusters [53].

Multiple definitions have been proposed for cloud computing, see [4][5]. For example, cloud computing is described as a powerful service and applications, integrated and packed on the web in [4]. Cloud systems have brought a new point of view concerning requirements of IT systems [163]. Clouds are becoming one of the most used metaphors to describe Internet needs. Internet usage brings new usage and also new data structures, quantity and velocity. For example, Internet services bring new data like: text, rich text, graphs, etc [143]. Some examples of cloud services provided on Internet are storage systems like: Dropbox, box.net or Google Drive. Another example provides an email service like Gmail, or some web crawlers used on Google or Yahoo. In [174], 15 services are referenced to be used with clouds.

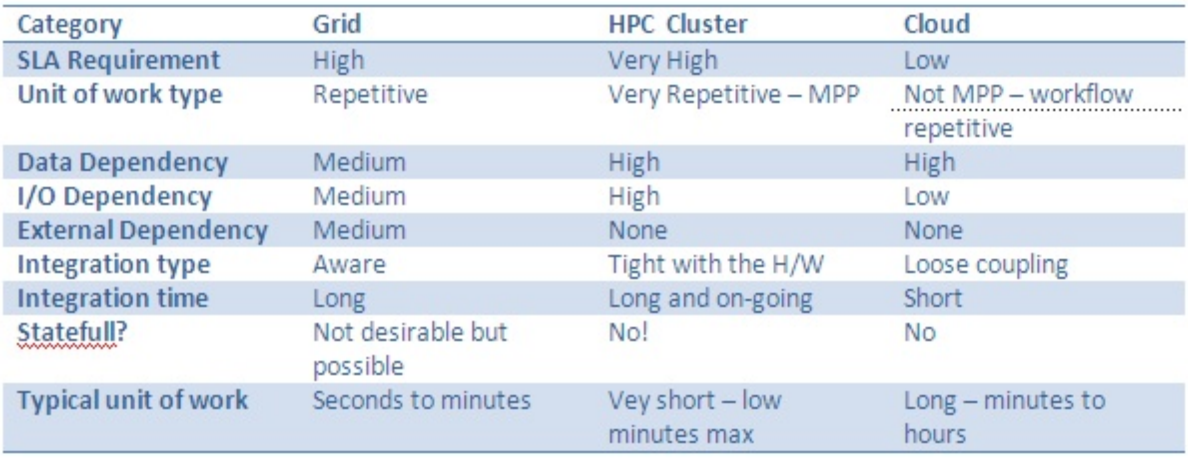
Cloud is considered as a cornerstone of Web 2.0. It can be used to host services, resources, applications, etc. Cloud computing refers both to the application and the hardware layers. As stated in [1], Cloud computing is a large-scale distributed computing driven by the economies of scale: the final price is linked with the quantity of CPU used.

Three important features compose cloud systems: processing, storage and network. To deliver resources on demand, abstract layers are used to manage cloud. Abstraction is provided by the intensive use of virtualization. With virtualization, it is possible to produce dynamically the most suitable architecture for each specific purpose. This strategy reduces the complexity for deployment. Virtualization is used on public, private and hybrid clouds. For example, on public clouds, a customer can extend his CPU resources or storage system with a credit card. This extensibility is provided by new virtual machines. The public cloud business model is based on computing and storage requirements. Resources of cloud systems are managed on demand, and they are released as soon as possible. With this strategy, Clouds systems response time is drastically reduced and users can expect a short response time (within 100ms [6]).

An example of a cloud public provider is Amazon. Amazon provides mainly two cloud services: one is dedicated for computing purposes: EC2 (Amazon Elastic compute Cloud) [199] and S3 is dedicated to storage (Amazon Simple Storage Service) [200].

In the earlier days of clouds, some companies have introduced a huge mistake on Grid and clouds. They decided to switch some of their product names from grid to cloud. This strategy led to a global mistake of the term Cloud. Moreover, as stated in Figure 2 some overlaps exist between both strategies, and a separation is not so clear. A cloud system is defined as massive, parallel, scalable and on-demand computing capabilities [7][10]. Differences between clouds and grids are deeply presented in [140]. Company mostly own Clouds, while grids are owned by public institutions (Universities). But cloud systems are not only restricted to companies: researchers also develop and deploy their own cloud facilities, for example using the Hadoop framework [39][44]. With this framework they are able to provide storage, elasticity and fault-tolerance features of a big data system. This strategy can replace their traditional data warehouse solutions [42][43]. Hadoop is also becoming the new data and computational standard for scientific needs.

To identify the best solution between Grids and Clouds, it is necessary to identify the requirements of the target applications. In [163], the authors provide some questions to identify which solution is the most suitable: information classification, computation needed, which data will be stored and which cloud provider is the most suitable. To extend these requirements we can use different categories provided in [142] as shown in Figure 4.



Category	Grid	HPC Cluster	Cloud
SLA Requirement	High	Very High	Low
Unit of work type	Repetitive	Very Repetitive – MPP	Not MPP – workflow repetitive
Data Dependency	Medium	High	High
I/O Dependency	Medium	High	Low
External Dependency	Medium	None	None
Integration type	Aware	Tight with the H/W	Loose coupling
Integration time	Long	Long and on-going	Short
Statefull?	Not desirable but possible	No!	No
Typical unit of work	Seconds to minutes	Vey short – low minutes max	Long – minutes to hours

Figure 4. Comparison between: Grids, HPC and cloud.

3. Cloud infrastructures

When we want to talk about clouds, it is necessary to deal with: architecture, virtualization, management, resources on demand, services, fault-tolerance and analysis mechanisms. Clouds have also to be classified between public, private and hybrid. Private companies host public clouds; private clouds are hosted by a specific organization (universities, companies) and hybrid clouds use both methodology [8][10].

Private clouds are managed by a specific organization. These cloud systems have a strict complexity, a dedicated bandwidth, dedicated computation capabilities and a controlled security.

Public clouds are more traditional. Some companies sell cloud services: Amazon, Google, etc. These clouds are more traditional and suffer from different problems: service is provided over the Internet, security is related to the provider, resources are shared between a wild set of users, etc. The last cloud architecture is named hybrid and is based on multiple public and private clouds. A virtualization layer manages services provided by these different clouds. This layer is used to create resources dynamically, in [10] these additional resources are designed to add computation or storage on demand. This virtualized layer provides the necessary resources to different cloud infrastructures. Virtualization is used as an abstraction layer between hardware, operating system and services. This strategy improves agility, flexibility and cost of a cloud system. Virtualization is used at different levels: network, storage or server. Virtualization is used to provide dynamically all the necessary resources. It brings elasticity to traditional IT systems [9]. To perform high reliability of cloud systems, these systems have been designed to be fault-tolerant and support an efficient load-balancing. Fault-tolerance is guaranteed by redundancy, histories and backup of data. The load-balancing is used to minimize job latency, avoid failover and downtime of services. When a component fails the manager will no longer send traffic to this element, this feature has been inherited from grid systems.

The next section describes different services provided by cloud systems. The second part presents storage methodologies of cloud systems. The last part presents computational methodology used with clouds.

General usage of cloud systems is based on a simple paradigm named Software as a Service (SaaS). Here, the goal is to provide an application as a global service. Two other main services have to be discussed: PaaS and IaaS [146]. But cloud can also provide some other services; the literature regroups all these services under the XaaS paradigm (Everything as a Service) [85][86][146]. Some taxonomy has been proposed to present XaaS [8][146].

3.1. XaaS

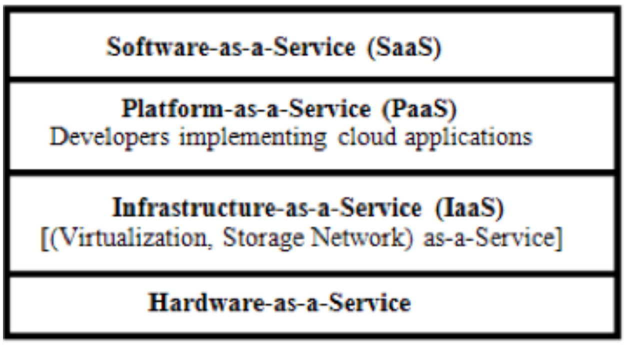

Software-as-a-Service (SaaS)
Platform-as-a-Service (PaaS)
Developers implementing cloud applications
Infrastructure-as-a-Service (IaaS)
[(Virtualization, Storage Network) as-a-Service]
Hardware-as-a-Service

Figure 5. Cloud Layer Architecture [8]

New XaaS paradigms are proposed every day. VMware introduced an example in a recent proposal. In [119], they have introduced a DaaS (Desktop as a Service) which enables users to deploy a window for different tiny clients (desktop computer, tablet, smartphones, …). As stated in [85][86], XaaS counts at least 35 different services.

Moreover each provider (Amazon, Google, Microsoft, etc.) has produced its own taxonomy of services, thus it is not easy to provide a global taxonomy. Some examples of XaaS services are: Software as a Service (SaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Communications as a Service (CaaS) or Monitoring as a Service (Maas), etc. In this document, we use the smaller taxonomy of [8]. They merge most of XaaS into three sub-classes: SaaS, PaaS and IaaS. And these different classes can be stacked into layers, see Figure 5. A stricter classification is provided in [10], two classes are used: SaaS and IaaS. As stated in Figure 7, these services can be encapsulated, IaaS can run PaaS and PaaS can run SaaS.

All these three sub-classes are presented in [8] and they have common points: these services are reachable through an Internet connection and through a single access point.

Software as a Service is the most common cloud service. It uses common resources and a single application instance [8]. An end-user can only access a single application. For example, Microsoft proposes this kind of service: Live mesh [1] (a file sharing software). GoToMeeting is another example; this application provides a cloud based meeting and collaborative tool. Now, most of Internet services are provided using SaaS: Saleforce, ConstantContact, NetSuite, etc.

Platform as a Service provides to developers an environment necessary to deploy applications. PaaS is a set of tools and services, which are able to scale with a system [8]. Google's App Engine is an example of this service [1], a developer is able to build a scalable Web application based on the architecture of Google applications. Other PaaS providers are Force.com [212], AWS Elastic [199], Beanstalk [213] or Heroku [214].

IaaS provides dynamic hardware, software and equipment to user. This subclass of service is brought by massive use of virtualization (hardware, automation). This model is suitable for companies who do not want to buy expensive data centers. Common examples of such services are Amazon EC2 and S3, which provide dynamic storage and computation [1]. Most of IaaS providers use closed-source virtualization environment like: VMware or Citrix. But it is also possible to use other open source solutions like Openstack with: Linux container or KVM virtualization software.

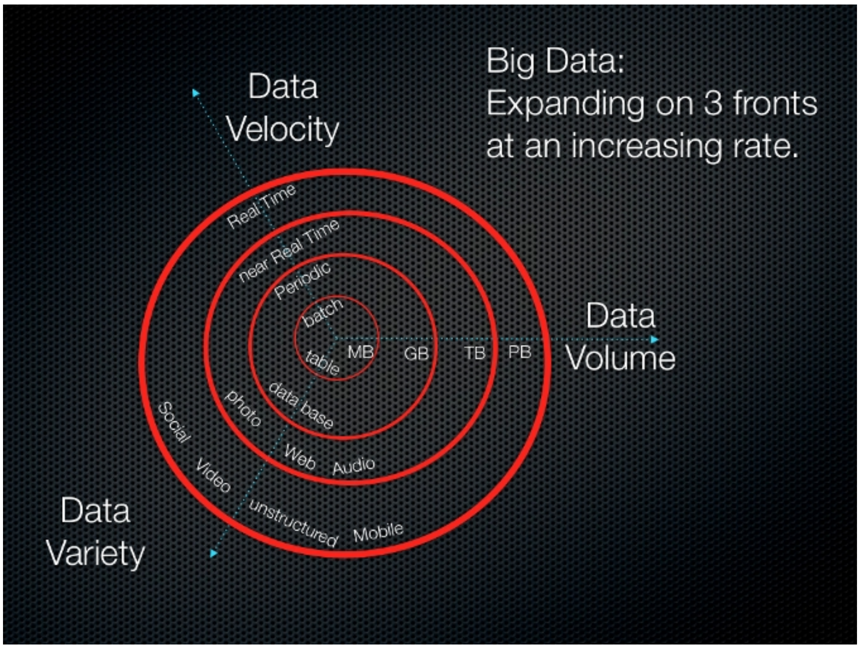

Big Data:
Expanding on 3 fronts
at an increasing rate.

Figure 6. A star glyph of 3V, definition of BigData [87][144]

3.2. Storage

Storage systems have followed the evolution regarding the amount of produced data. In the earlier days of data storage, the CRUD paradigm has been defined to set requirements of traditional database systems (Create, Read, Update and Delete). A second evolution of DataBases was brought by the definition of the ACID requirements (Atomicity, Consistency, Isolation and Durability). This requirement rules the most widely used DB like RDBMS (Relational Data Base Management System). A comparison between ACID and BASE (Basically Available, Soft and Eventual Consistency) approaches is presented in Figure 8.

But regarding to the fast evolution of the data produced, the ACID paradigm is now too restrictive to be used with new data. Each year, the amount of produced data is increasing, and meta-data is used to enhance data understanding. As stated in the introduction, 3 dimensions, named 3Vs [11][142], define this fast growing: Volume, Variety and Velocity. These 3 dimensions are represented in Figure 6.

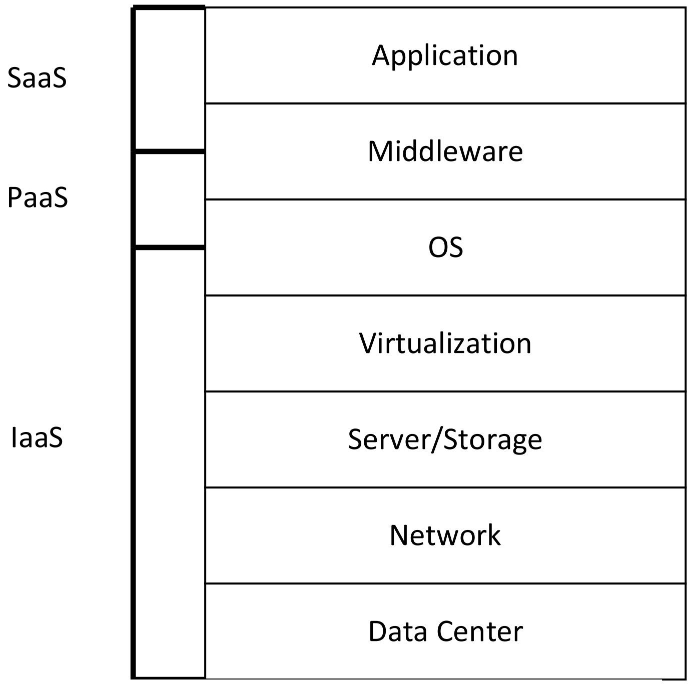


Figure 7. Positioning of IaaS, PaaS and SaaS.

Before choosing a Data Base, it is necessary to study the structure of the target data. DB is designed to store one of thee three kinds of data: structured, unstructured and semi-structured.

In the next sections, we present a classification, and some examples of DBMS, based on information in [15]. This classification is composed by: traditional RDBMS, other RDBMS (Distributed DBMS, Object-oriented, …) and non-relational DB.

ACID	BASE
Strong consistency	Weak consistency
Isolation	Availability first
Focus on “commit”	Best effort
Nested transaction	Approximated answers
Availability	Aggressive
Conservative	Simple
Difficult evolution	Faster
	Easier evolution

Figure 8. ACID vs Basic [201]

3.2.1. RDBMS

RDBMS are the most popular database management systems. RDBMS means: Relational Data Base Management System. The first description of RDBMS has been produced in 1970, [20]. This DB is based on a table-oriented data model; the storage of data is constrained by a schema. Data records are represented as rows of table, i.e., tuples. These DBMS have some basic relational algebra operations: union, intersection, difference, selection, projection and join. On top of these DB, a language named SQL (Structured Query Language) is used. The evolution of these DB has produced a large number of DBMS: DB-engine [18] presents a wide DB list. This web site references different kinds of DB (NoSQL, Object-oriented, …). If we take a look at only RDBMS, 79 RDBMS are referenced on this website, which contains 216 references. Four RDBS are in the top 5 of this webpage: Oracle, Mysql, Sql Server and PostgreSQL.

MySQL is an open source RDBMS. It is one of the most famous RDBMS because it has been massively used with web applications. This DB has been developed to answer to direct requirements of users. It supports multi-threading and multi-user accesses. This DB has been created in 1995, and the implementation is done in C and C++. It is possible to access to this DB using several languages: Ada, C, C++, C#, D, … A distributed version of this DB exists and was named MySQL cluster. Replication is also provided.

Oracle DB is a database marketed by the Oracle Corporation. It is a closed-source DB, which can be deployed on any OS. This DB has been developed in 1980, using C and C++. It is possible to use it with any programing language. This DB supports partitions and replication methods. This DB supports also multiple users at the same time.

SQL Server is a commercial product delivered by Microsoft. It has been developed since 1989. It is a commercial DB, developed in C++. It can only run on a Windows instance. It is possible to use this database with .NET language or Java. Data can be distributed across several files and replication depends on which version of SQL Server is running. Multiple users can use this DB concurrently.

3.2.2. Object Oriented DataBase

An object database is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases, which are table-oriented.

Object-oriented database management systems (OODBMSs) combine database capabilities with object-oriented programming language capabilities. OODBMSs allow object-oriented programmers to develop the product, store them as objects, and replicate or modify existing objects to make new objects within the OODBMS. Because the database is integrated with the programming language, the programmer can maintain consistency within one environment, in that both the OODBMS and the programming language will use the same model of representation. Relational DBMS projects, by way of contrast, maintain a clearer division between the database model and the application.

Express Data Manager™ (EDM) (see [225] for more information) from Jotne EPM Technology is an object-oriented database that is well suited to handle both structured and unstructured FEM and DEM data. The reason for this is that ISO has standardized an object oriented schema for storing and exchange of FEM and CFD data, ISO 10303-209,Multidisciplinary analysis and design (AP209). The schema is written in the modelling language EXPRESS (ISO 10303-11, see white paper on [226]). EDM is designed to handle EXPRESS objects and is by that a good choice database system for standards based and, thus, interoperable FEM, DEM and CFD data.

EDM is a database system that is well suited for deployment on HPC clusters; see Figure **9**, below. The EDM database server, EDMserver™ consists of one database server process and several application server processes, as indicated on the right hand side of the sketch below. The database server performs initialization functions, like compile EXPRESS schemas, create users, repositories, models (datasets) etc., and synchronisation functions between EDM application servers. It is the EDM application servers that execute the database queries, and these EDM application servers can be deployed on several machines concurrently.

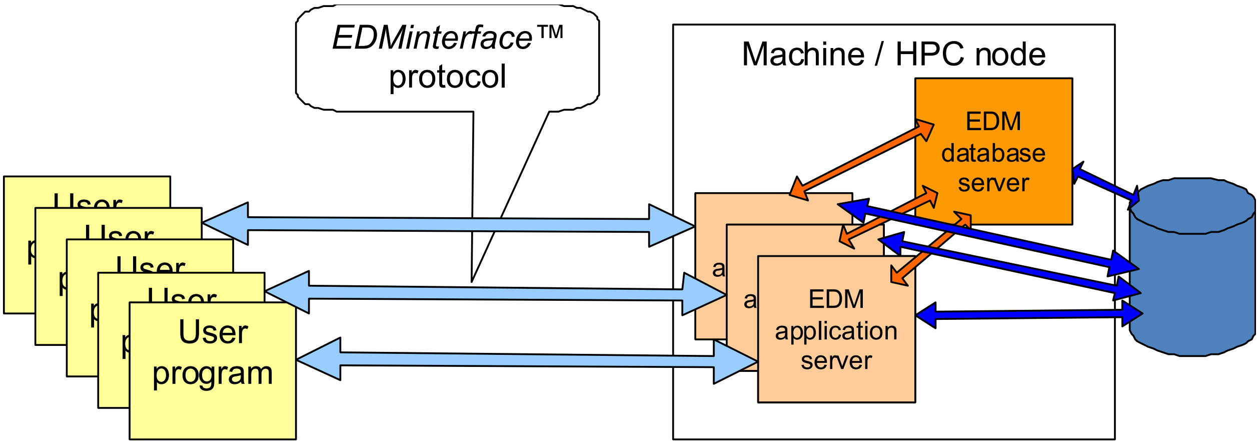


Figure 9: EXPRESS Data Manager (EDM) in the context of a HPC cluster

The communication between the EDM application servers and the database server is implemented by messages over TCP/IP. By that it is easy to distribute the application servers on many machines. The TCP/IP approach is better suited for HPC clusters than communication solutions using, for example, shared memory messages, semaphores etc. .

An EDM database can have many independent datasets, in EDM terminology called models. Each EDM application server may update one data model; therefore, this architecture allows parallel updates of the database. If the scenario requires a database that consists of one huge dataset, all EDM application servers may read this model concurrently. Queries may be split into subqueries, and each subquery may be assigned to one EDM application server. The result is an implementation of the Hadoop Map/Reduce pattern.

The EDM database is an object oriented database system where each object has a "database global" object id; the object id is a 64 bits integer. The uppermost part of the object id is the model number of the model that the object belongs to. The lowermost part of the object id is the object number within the model. By this great object address space it is only disk size on the database server that limits the size of the database.

These characteristics make the EDM database system a good starting point for implementing efficient FEM/DEM queries on a HPC cluster.

OoDBMS (Object oriented DBMS) was developed to improve storage applications. Objects produced by software are directly stored in OODBMS. Schema of data is provided by the applications. Some existing DBs have evolved to support this new paradigm, and others have been developed. Some examples of these DB are: Cache, DB4o, Versant Object Database or PostgresSQL.

PostgresSQL is developed since 1989. It was developed in C and can run on any operating system. This DB can be used with several programming languages: .NET, C, C++, Java. This DB is mostly Object-oriented, and it does not support the partitioning of data. Replication is performed using a traditional master-slave solution. Multiple users can use this DB.

A deeper comparison between RDBMS and OODBMS is provided in [231]. Data of OODBMS contains both execution part and information. In a RDBMS, only data are stored into a two dimensional view (table). In an OODBMS, data are stored into structured objects. Authors of [231] say that OODBMS must be used for complex data and/or complex data relationship. OODBMS are not recommended when only a few join operations are necessary and when there is large volume of simple transactional data. This document also provide information regarding to advantages and disadvantages of RDBMS compared to OODBMS. These information are quite useful to determine which solution is the most suitable regarding to data and operation over data.

3.2.3. Other RDBMS

To be more suitable with new data requirements, RDBMS have also evolved. DBs have evolved to support multiple instance of the DB.

To increase performance of DB, solutions have been proposed to use multiple nodes. Thus two solutions arise: the first one uses distributive properties (like the solution powered by Google: BigTable), the second one uses parallel databases. To be able to deal with BigData, it is necessary to use distributed DBMS. In 2006, BigTable has been presented in [56], it is a distributed file system management, which can be used to store data in a distributed way. These parallel DBs are designed to run on multiple nodes, with multiple CPU and using parallel storage systems.

3.2.4. NoSQL

NoSQL describes DataBase, which do not only deal with SQL. A clear definition of NoSQL does not exist, and the points of view can diverge. An example is presented in [16]; NoSQL is described as a distributed DB without any fixed schema. Sometime, NoSQL refer to a non-relational system. Another definition of NoSQL DB refers to a structured DB. A common property with all NoSQL definitions is Web 2.0, huge amount of data and huge amount of users.

This DB has been introduced in 1998 and has reached a growing interest in 2009 [16]. These DBs do not use ACID schema, thus they are able to achieve higher performance and higher availability, compared to traditional RDBMS. Store and retrieve mechanisms are also different.

As stated in [15][16], NoSQL DBs can be classified into different types: Key-Value, Document, Column, and Graph. Some other classifications are available, see [17]. In this document, we use the most common classification: Key-Value, Document, Column, and Graph. This classification of DBs is based on [14][18].

3.2.4.1. Key-Values

The Key-Value strategy allows the application developer to store data with a schema-less approach. Two fields represent data: a key and a value. This strategy allows storing data using an arbitrary index based on a key. Both fields of this DB depend on the applications [88]. This mechanism can be found in several DB: Redis [24], CouchDB [88], Cassandra [88]. A more complete list of Key-Value databases is presented in [17][202].

3.2.4.2. Document

Document DBs store data with named documents. The engine is used to manipulate these documents and improve storing, retrieving and managing. Using a document storage approach allows to deal with semi-structured data. Access of data is done using a unique key for each document. This methodology have been implemented into: MongoDB [24], CouchDB [24], Riack [24], etc.

3.2.4.3. Column DB

In a column DB, data is stored in tables. Each table is structured with a column schema. This strategy serializes all data onto columns. This strategy has been used in several DBs: BigTable [17], Hbase [24] and Cassandra [24].

3.2.4.4. Graph

The last kind of NoSQL DB is named Graph DB. This DB uses a graph structure to store data, composed by: nodes and edges. With this strategy, every element contains direct access to adjacent elements. With this method, no indexes are necessary. Examples of graph DB are: Neo4j [24], FlockDB [17], OrientedDB [17], Pegassus [31] or Pregel[30].

3.2.5. The case for VELaSSCo

In this section, we will propose a summary of VELaSSCo needs concerning the storage system. Data of this project will be produced by two specific simulations: FEM and DEM. For Discrete Element Methods, particles and contacts can be created and lost during the whole simulation. For both solutions, a solver produces relative small files, which contain the different time steps of a simulation (See D1.3, section 2.5, page 18). These files will be structured regarding to a specific format. The amount of data produced by these solvers depends on the number of generated time steps, it is also depends on the number of nodes/Elements (FEM) or particles (DEM) involved in the simulation. The velocity of produced data vary regarding to the simulation engine.

As the data already exists in distributed files at day 0 of the project, it is necessary to find a distributed system, which enables to inject the data based on a key value or document oriented approach into the DB (key = analysis_name + time_step + result_info + partition_info, value = result_values). With this methodology, the complexity of integration is drastically reduced. Data storage needs to be directly reachable by a HPC facility. If the DB system is distributed, thus the location of information has to be transparent.

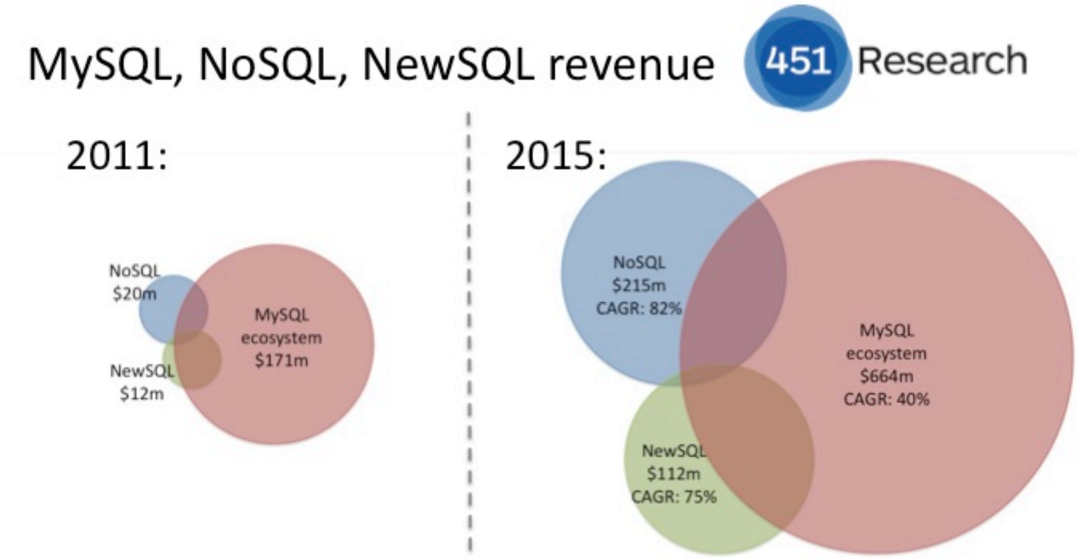


Figure 10. Expected revenue of different DBMS [221].

3.3. Computation

Computational models of DBMS can be divided into two different strategies: complex and simple queries.

Complex queries apply complex operations (join, selection, projection, etc.) on data. A simple query uses a simple model: extract data from the dataset (without any result enhancement). Complex queries can be expressed by simpler queries.

From these two models, some specific query languages have been proposed. In this paper we present only two of the most used models: SQL and MapReduce. SQL can express complex and simple queries, while MR is used to express simple queries (some extensions of MR have been proposed to improve this simple query model).

This section is dedicated to compare these strategies. We present how computational models work, describe when these methods have to be used, and which drawbacks arise from these models.

3.3.1. SQL

Most of traditional RDBMS systems use complex queries to manage their data. SQL is a normalized language used to exploit DBMSs. This language has been built on top of CRUD (Create, read, update and delete). SQL also supports more complex queries like: joins, transactions, etc. This language has been developed to run one-time queries. Most common SQL systems do not provide suitable tool to run long queries, and the results of these executions are sent to the users at the end of the computation.

Unfortunately, the fast growth of datasets introduces new requirements for data computations. An answer for these requirements has been presented in [74]. Authors proposed a Data Stream Management System (DSMS), as an alternative to the traditional DBMS systems. This project deals with three important features: the first one concerns streamed data where data is sent at runtime, and it is not necessary to wait the end of a computation. The second feature concerns load variability (system deals with variability of queries) and the last part concerns the management of resources. To perform these continuous queries, authors have introduced a new language named CQL (Continuous Queries Language). This modern paradigm brings new challenges: necessity to provide suitable stream structures, and traditional declarative queries need to be translated into specific continuous jobs.

A DSMS needs specific features; it is necessary to deal with coarse data (sometime accurate results are not possible and degraded results have to be sent). In [74], authors provide a centralized monitor and manager of queries. With this methodology, system workload is always under control. This project is unfortunately no longer maintained, and has been stopped in January 2006.

3.3.2. MapReduce

The second computational methodology is for simple computational tasks. The most used model is MapReduce. This methodology is composed by two functions: Map and Reduce. Map function is used to filter and sort the data, while the Reduce function is used to aggregate and summarize data produced by Maps. This methodology is described in several papers: [19] [35][58]. In the original paper, MapReduce was implemented using the Sawzall functional language [35]. Google has popularized this methodology with one paper: [89]. The Hadoop framework provides an open-source implementation of MapReduce [49].

MapReduce has massively been used for words search and count, but this simple computational model is not suitable for complex queries, and expressing more complex computations requires drastic code modifications. In the scientific community (Machine Learning for example), MR has some difficulties to express iterative algorithms, thus it is necessary to find new solutions. Problems that cannot handle efficiently with MapReduce are presented in [25].

3.3.3. Procedural approach

The EDMopenSimDM™ system (see: [227] or [228]) offers AP209 engineering queries to the end user. Examples of AP209 queries are Model Query, Node Query, Element Query, Element Property Query, Displacement Query and Stress and Strain Query.

These AP209 queries are built upon the EDM C++ Express API. This is a generated C++ interface to the EDM database objects of AP209. Each EXPRESS entity (object class) is mapped to a corresponding C++ class. The generated C++ classes have put and get methods for all attributes. The EDM C++ EXPRESS API is an extension to the EDMinterface™ API, which is shown in Figure **9**, above. See also the EDMinterface™ chapter in EDMassist in [229].

In the AP209 engineering queries each query is implemented along the following pattern:

1. The objects to be queried are either all of one specific class, a range of objects, objects with specific ids or objects that satisfy pattern matching search criteria.
2. For each object of interest, the database object is located by different access methods dependent on the search criteria. If all objects of one specific class shall be visited, the query traverses the list of all objects of that class. If specific objects ids are the search criteria, attribute hash tables are used.
3. When an object of interest is found, the C++ get methods are used to retrieve information about the object and all other relevant objects for the query.

The AP209 engineering queries developed for the EDMopenSimDM™ system are a good starting point for developing visualization queries for FEM/DEM objects.

3.3.4. The case for VELaSSCo

To identify the most suitable strategy for the VELaSSCo platform, it is necessary to identify its needs. Discussions with the partners highlight some requirements: an efficient computational model, a distributed computational model, a simple interface to access data and a computational model suitable for batch and online queries.

To store data, the most common habit is based on SQL storage; developers have been formed to use SQL engine. Thus switching to a new paradigm like MapReduce is not a simple task. To deal with Big Data, it is necessary to evaluate the performance of both model, see [19]. The rest of this section describes some differences between these two models regarding to the VELaSSCo project.

EDM, on the other hand, is a strongly object oriented DB, in which associations between objects are very important, are another key value in the system. The format followed is the one defined in AP209. As the input data does not have this information, it should be created when the simulation data is injected into the system, like defining the mesh-groups, identify the distributed parts that belong together, group the results of the same analysis and time-step, etc.

The first difference is on data structures: MR allows arbitrary data formats, while DBMS requires well-defined data formats. MR and parallel DB share some common elements, but some noticeable differences appear in both systems. This missing schema can produce data corruption, while a strict schema makes SQL DBs less flexible.

Another difference between DBMS and MapReduce resides in the indexing of data. The index of DBMS is based on hash or B-tree, while MR framework does not provide a built-in index. MR programming model needs a higher language to be more user-friendly, and MR can be compared to a low-level language like assembler. Hive, Pig, [42][73][123] have emerged to propose this higher language, and in [158] other SQL abstractions are presented. These solutions bring SQL abstractions on top of MR, thus it becomes less difficult to interact with the MR computational model. SQL queries are translated into a set of MR jobs. Hive, [46], brings also an improved file storage system [122]. This storage strategy (ORCFile) compresses data and brings good compression results (78% smaller than raw information).

Outputting results from RDBMS and cloud solution is also different (in their traditional methodology). With the MR computation, it is necessary to create manual data filter. In the case of RDBMS, data is already filtered. This issue is also similar regarding to intermediary steps, RDBMS uses raw data, while MR writes intermediary files.

Finally, MR provides a native sophisticated fault-tolerance model compared to RDBMS; this advantage comes from the built-in high redundancy of data [21]. From these different key points, [19] have proposed several benchmarks based on the original MR task: Grep Task, and four HTML specific tasks (selection, aggregation, join, UDF aggregation (user defining function)). Then the authors make a comparison based on user-level aspects. This benchmark shows that in most cases, parallel DB are an order of magnitude faster than MR. But MR is more user-friendly, and deploying this system is easier than a parallel DB. Extensibility is also in favor of MR; DBMS are not well suited for such tasks. Some of the discussed VELaSSCo queries, which requires the processing of the distributed information and the fusion of partial results, which also may require some distribution-fusion calculation iterations, can be harder to implement in EDM than in map-reduce.

In the specific case of VELaSSCo, a hybrid computational model can be more suitable. It is important to provide an extensible approach which can answer to well define requirements and which can be extended to future needs. One of the requirements for the VELaSSCo platform is to support special particle shapes, or cluster of particles, which can be implemented on other systems, but requires an extension of the AP209 standard so that the EDM systems support them too. In the meantime may be they can be treated as mesh of particles, for the cluster, so that a 'mesh of particle clusters' would be represented as a 'mesh of mesh of particles' in EDM/AP209.

The next section presents some improvements concerning cloud solutions.

4. Best practices

With a major evolution of usage, cloud systems need some improvements, evolutions. These modifications have to be focused on cpu usage, long run time computation, etc.

To manage this new amount of data and computation, industries and scientists use a cloud approach to perform both computation and storage. Cloud systems have to be scalable, see [191]. Three scale strategies exist: scale out, scale up and scale side-by-side. Scale out is used to refer to the increase of number of nodes (machines). Scale up increases the capacity of a system. And the last methodology is used to deploy a temporary system, used for test and development. Regarding to actual state-of-the-art, a large variety of cloud architectures are based on Hadoop; see [34][40][42][50][80][177]. Computations presented in most cloud systems use the MR methodology.

In this section, we present some best practices used to improve cloud architectures. Our interest is focused on virtualization, storage and computations improvements. The last part presents software stacks.

4.1. Hardware

IT systems used for Big Data have multiple tradeoffs: data storage, network resources, computation capabilities and energy efficiency, see [203]. This section presents optimizations provided by scientists at different hardware levels.

4.1.1. Storage

Computer hardware is composed of two different storage systems: main memory and secondary storage systems. Both solutions have their own advantages and drawbacks. The main memory (RAM) is mostly used to store cached data or current used data. This system has a high throughput, directly linked to CPU. This storage system cannot be used as a long-term storage system; some energy is required to keep data available; the cost of this memory is expensive. Hard Disk (HD) is the most suitable solution to store high data volume for long periods. It is also the cheapest solution. But this system suffers many drawbacks: lifetime (Hard disks are mechanical systems), speed efficiency (it is a low speed storage system) and failure (this system is quite a sensitive to hardware issues).

Some improvements have been developed for this hardware, for example SSD HD, SCSI HD or PCI-express HD. These new storage systems move I/O bottleneck to computation units. But, for most IT systems, the main issue remains I/O of HD, distributed systems have been partly an answer to this issue [81].

In [25], the authors present a solution based on SSD, this methodology allows to increase the amount of available I/O operations per seconds. Optimized hardware can improve performance. Hyder is a new transactional record manager. The main idea of Hyder is to provide a data-sharing architecture without requirement of portioning database or applications. Transactions are applied on a snapshot and a log is produced. Then, this log is broadcast to all nodes of the server. Nodes are now able to apply these modifications to their local data. Hyder is composed by three main components: log, index, and roll-forward. Hyder is designed to simplify application development by providing a set of tools:

partitioning, distributed programming, layer of caching and load balancing. The architecture of Hyder has also been improved and four main key points have been identified: high performance network, large main memory, many-core processors and solid-state storage.

Storage facilities are mostly led by two technologies: regular HD and SSD. SSD can provide more I/O operations than a mechanical HD system. But their lifetime is limited: SSD have a limited number of Read and Write operations. Moreover, SSD cost is drastically more important than mechanical HD cost. In [72], authors proposed to use Blu-ray disk to store cold data. Cold Data is a specific set of information that is quite never accessed. They have evaluated their methodology on a real Facebook dataset and they are able to save 50% of hardware cost and 80 percent of energy. This storage strategy is only used into experimental data center, and none production data center are based on this strategy.

4.1.2. Computation

The popular computational model for cloud systems is MapReduce. This computational model is implemented among several frameworks. One of them is Hadoop, it is a Java framework that did not originally deal with parallelism on modern architectures. However, hardware has evolved, and now standard architectures are composed by multi- or many-core nodes, see [25][81]. In this section, we present improvements based on Multi-core architecture and GPU. The native implementation of Hadoop was not suitable for multi-core architecture. Now, some researchers have developed MR solution which used new hardware capabilities.

The first example is presented in [33] and is named Nephele. It provides a new processing framework designed for cloud environments. This system can dynamically allocate or deallocate resources on clouds. The computation is expressed as a directed acyclic graph. In this case, jobs are represented as vertex and edges of the graph define communication flows. With this strategy, it is possible to express a complex computation using a parallel processing approach. This tool is also composed of a specific scheduler, which allows dealing efficiently with parallelism. This solution is very similar to Dryad [45].

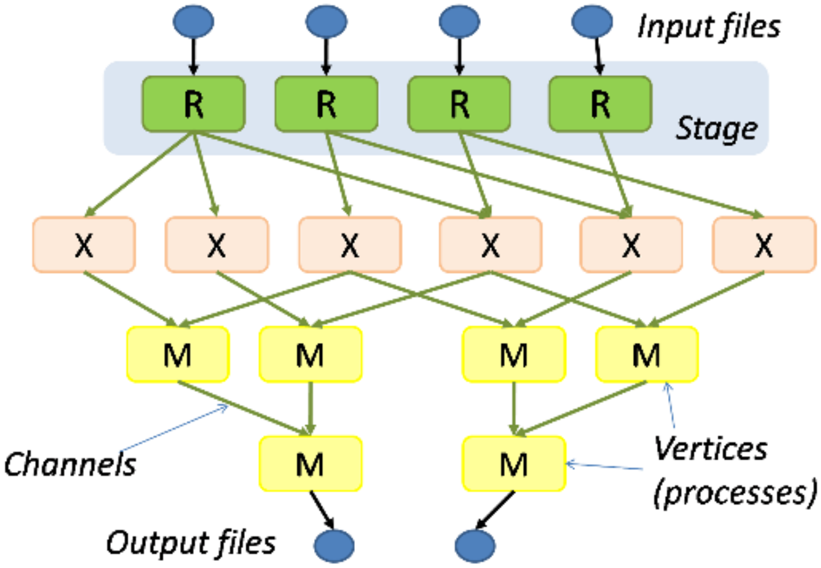


Figure 11. Structure of Dryad.

Dryad, see Figure 11. Structure of Dryad., has been developed to solve two problems: efficient parallel applications and distributed applications. This framework is presented in [45] and was motivated by the emergence of large-scale Internet services. Dryad provides an execution engine able to handle large distributed and concurrent applications. This strategy is based on several features: efficient scheduling of CPU and hardware usage, strategies of recovering from communication and computer failures and moving data among vertices. Dryad is based on a graph flow control. Communication is allowed between each node through a specific pipeline. This method alleviates the difficulty to program large distributed systems, and concurrent applications. A Dryad job is designed by its communication flows. Jobs are acyclic graphs where each vertex is a program and each edge is a data channel. It is a logical computation graph mapped onto physical resources. This application gives to developer the opportunity to express some queries in a more complex way than the traditional MR model. But this solution brings more complexity to the MR model.

Another solution to increase computation capabilities is based on existing multicore task scheduler for Hadoop systems, see [47]. Hadoop employs a JAVA runtime, which is not been optimized for parallel computing or specific hardware instruction (SIMD). Some authors have proposed a hierarchical MapReduce application, named Azwraith, which use the optimized MapReduce runtime: Ostrich. This runtime is able to deal efficiently with task parallelism and data locality. At the opposite of the actual Hadoop implementation, they are able to reduce computation using a caching system. This solution answers the requirements of the original Hadoop; the workflow has been preserved. Regarding to actual Hadoop framework, authors are able to achieve higher speedup.

Another hardware suitable for improving the performance of computations is provided by GPU. These devices are composed by large set of computational units.

Hadoop has been extended to use GPU capabilities. MARS is an implementation of MR using GPU [64]. This approach is implemented for data- and computation-intensive tasks. It removes the complexity of GPU programing by providing a familiar MR model to the developers. Developers can write MR tasks without any knowledge on GPGPU. To validate their results, authors implement six classic tasks of web applications and show that their strategy is better than the traditional CPU-based methodologies. This approach gives promising results, but last updates to this implementation have been released in 2009. Figure 12 presents the Mars Workflow; grey parts are executed on the GPU.

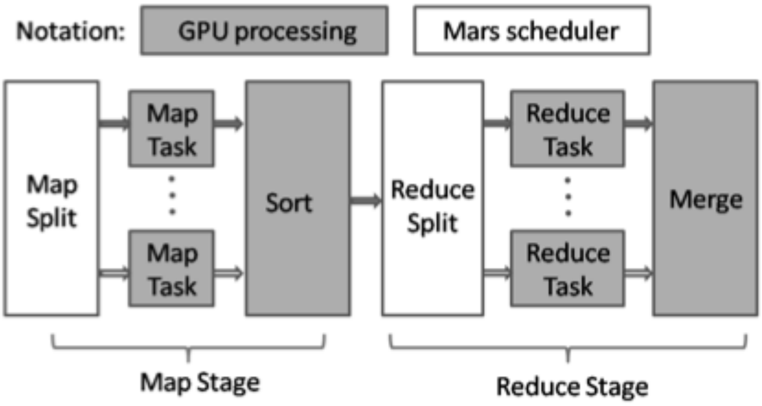


Figure 12. MARS workflow

4.1.3. Others

We have presented two of the most important hardware strategies to improve cloud systems: computation and storage. But, other solutions exist but have a minor impact for this project, thus this section regroups them.

Network is an important aspect for public clouds. Nodes are disseminated all around the world (for example Amazon cloud). Thus to improve a cloud system, it is necessary to study the topology of the architecture. Network infrastructure has evolved and follows the same path than computation capabilities. Now, with new networks and compression of data, it is possible to transfer more information than the existing methods. The gap between private and public cloud is reduced gradually [25] [81].

The second part of this section concerns improvements on power efficiency [6][26][48]. Modern hardware consume more and more energy, thus power has to be taken into account during the cloud deployment. The energy bill of IT equipment depends on the number of elements used. Energy consumption depends on the computation done on the system. Regarding to the computation model used, an online model is not suitable for energy savings because the necessary computation is often hard to predict. Architectures have to be design with a low energy footprint. Intel provides an example of low energy architecture [157]. Intel has designed most of the hardware presented, but in this paper, they also propose some optimizations to improve OS, Hadoop System and hardware parameters.

Consumption power of modern data center has become an important issue. In [232], authors provide an overview of different techniques used to improve energy efficacy of data center. Consumption of data center is a new challenge; not only to reduce the cost of data center exploitation but also to reduce it is energy footprint. This document suggests also some new way to follow most related to cloud system.

4.2. Virtualization

Virtualization is a key feature for IaaS. Virtualization allows deploying dynamically virtual machines configured for specific needs. As stated before and in [25], virtualizations now lead the deployment of the architectures (location of computation, storage, and so on).

Management of Virtual machines is performed using a manager named: VMM (Virtual Machine Monitor). This tool allows the system administrator to share hardware resources into several virtual machines. Three important elements characterize virtual machines: VM have to run existing applications without modifications, VM have to run subsets of instructions directly on CPU and VM need to access to resources.

IaaS can now deal efficiently with some scientific requirements: long runtime and high resource utilization. From these requirements, some improvements have been proposed, and an example of one of them is presented in [77], BlobCR. Here, the authors present a solution to minimize the storage space and performance overhead on a checkpoint/restart strategy. This methodology implements VM snapshots, by inserting checkpoint-restarts. An experimentation of this methodology has been provided on G5K test bed [2]. This solution uses MPI to provide a message passing system. A part of the distributed file system is used to store the checkpoints. Snapshots of the VM are saved in the checkpoint repository, the

VM is suspended and the application state can be saved. Then, while only a small set of virtual disks have been modified; BlobCR needs a specific strategy to avoid useless data replication. Authors have used a transparent solution for the versioning system. BlobCR is available.

Vmware also provides a solution to deploy a Hadoop ecosystem, which is based on a virtual environment. In [233], authors present their distribution based on their virtualization software. This document shows benefit of virtualization for Hadoop deployment. This strategy allows deploying as fast as possible a whole ecosystem. Fault tolerance of this system is performed through virtual machine. Energy and computation efficiency is optimized; it is possible to use more resources from the IT system. With their strategy, it is possible to provide a Hadoop as a service platform.

4.3. Data Storage

Cloud systems have eliminated the burdening maintenance of the traditional HPC systems. Scientists are now using these modern infrastructures to run their specific workloads. However, scientific workloads exhibit some performance issues on cloud compute systems [83]. Authors analyze performance of the Amazon EC2 platform for scientific computations. Amazon EC2 is an IaaS, the virtualization of this system is provided by Xen. In [83], authors highlight difficulties of using scientific tools and they provide an evaluation for using IaaS in the scientific context. Two parts are proposed: a specific cloud methodology and an infrastructure approach. This paper claims that there does not exist a “best”-performing instance for scientific computing. Computation workloads may exhibit unstable behaviors when applications run over virtualized resources. To obtain reasonable performance, the users need to tune the execution plans of the applications and choose carefully the virtualization infrastructure. Amazon EC2 is generally not suitable for scientific computations, but some providers have more specific services for such kind of applications (Penguin on Demand or HPC as a service).

Clouds are often used as massive storage systems with versioning, redundancy, etc [48][169]. To ensure the best service, specific cluster file systems are used [97]. These specific file systems are mainly split in two subclasses: Shared FS and distributed FS. Both systems have their drawbacks and their specific usage. Shared FS provide direct access level to multiple computers at the block level, while the Distributed FS does not share any blocks. In most cloud solutions, the FS used is based on distributed FS. The fast growth of data has highlighted the importance to use distributed systems [25]. A cloud system needs tools to extend its file system dynamically, with the same redundancy capabilities, distribution approaches, etc. In this section, we present some example of distributed FS and improvements provided for the read and write I/O of these FS.

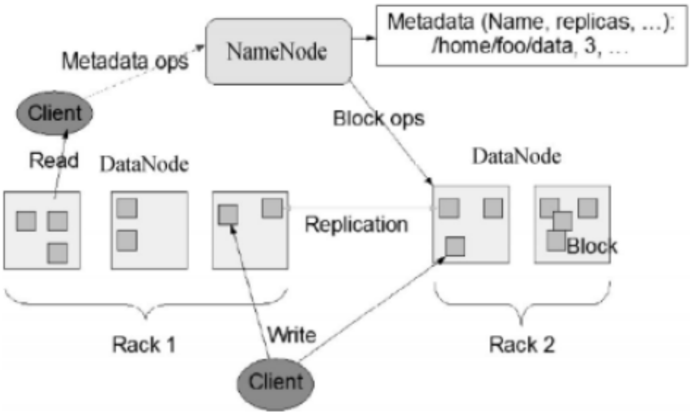


Figure 13. HDFS Architecture [93].

Most cloud systems use a distributed file systems to manage data, and this strategy is named share-nothing: data is split across the file system. A distributed file system implies to use synchronization barriers. In [38], authors do not use synchronization; they use a logical partitioning of data and a specific query processing. This method avoids synchronization barriers and keeps consistency of the file system. The modern FS uses atomic operations to perform this synchronization (an atomic operation is a lightweight operation). With this method, authors of [38] have built a storage-distributed system based on both RDBMS and NoSQL DB. This solution stores data into a centralized node called: *shared record store*. To improve query, the system decouples queries processing and transactions management. In the context of this paper, atomic operations are used to implement a multi-version concurrency control. An evaluation is presented by the authors and provides a comparison with MySQL cluster DB. The methodology presented in this paper is claimed more efficient than most common RDBMS.

Distributed FS have been massively used in the case of cloud architectures. In [56], authors present BigTable. The provided data model is a sparse, distributed persistent and multi-dimensional sorted map. Cells contain multiple versions of the same data; this approach is used to improve indexing methodology used for such a DB. This methodology has been designed to provide a high availability, and machines can be added dynamically regarding to necessary resources.

Hadoop Distributed File System (HDFS) is one of the most used FS with a Hadoop framework. It is a distributed and extensible file system. It has been designed to store large datasets on large clusters of computers. This solution is a virtual FS, deployed on a top of a regular File System.

As stated in [93], HDFS architecture deals with two different components, see Figure 13. These machines are named: a *NameNode* and a *DataNode*. The *DataNode* is in charge of the storage. And the *NameNode* provides File tree, namespace, metadata and directory. To store large files, HDFS split them and send data chunk to some Datanodes. To increase reliability of the system, each block is repeated among the whole system. *DataNode*s can communicate together to dispatch chunks among them. Unfortunately, this system is not POSIX compliant, and the *NameNode* is a single point of failure, it is necessary to duplicate it to increase the security of the system.

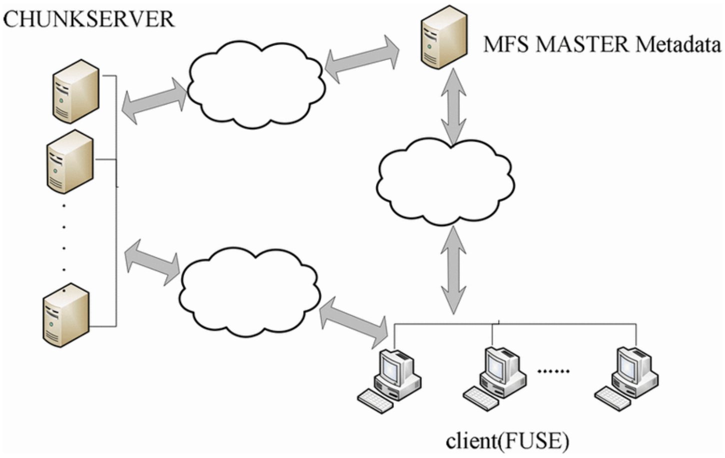


Figure 14. MooFs Architecture [93].

Another example of FS is named Moose File System (MooseFS). It is a distributed FS, which enable fault-tolerance, scalable, POSIX compliant, and suitable with any kind of data or hardware. This system is inspired by GFS [215], it is based on three nodes: a *Master Server* (master server), some *Data Servers* (chunk servers) and a/some *Metadata Backup Servers* (metalogger server). The first node is used to manage the whole FS. It stores the metadata. A chunk server is used store data and synchronizes them, if multiple copies of the same files are stored. The last part store metadata and change logs: this element periodically updates the main metafile. Figure 14 presents the architecture of this FS.

In [37], authors present NetCDF, a distributed file designed to provide an efficient parallel I/O interface. This File system has a set of tool that provides a portable and efficient file format and programming methodology. The programming interface is built on top of MPI-IO, to perform efficient parallel I/O. NetCDF is currently under development and some releases are proposed to end-users.

Ceph is another well known distributed FS, [76]. This FS provides a separation between data and meta-data with a specific data distributed function called CRUSH designed to run efficiently over heterogeneous and dynamic clusters. With this solution, authors address some critical challenges: scalability, performance and reliability. Ceph provides a single uniform directory hierarchy based on the POSIX semantics.

Lustre FS, presented in [91], is a parallel-distributed FS. This FS is often used in Supercomputing: it provides a scalable method to use multiple computers and store data. This system is composed by two elements: a metadata server, which stores namespaces, metadata, filenames, directories, permissions and file layouts, and object storage servers which store the data in one or more objects. The architecture of Lustre is presented in Figure 15. This file system is POSIX compliant.

A comparison between Lustre and HDFS is provided in [92], see Figure 16. The authors present a strategy, which combines Lustre FS and HDFS. As stated in this paper, using Lustre introduces some overhead, compared to HDFS.

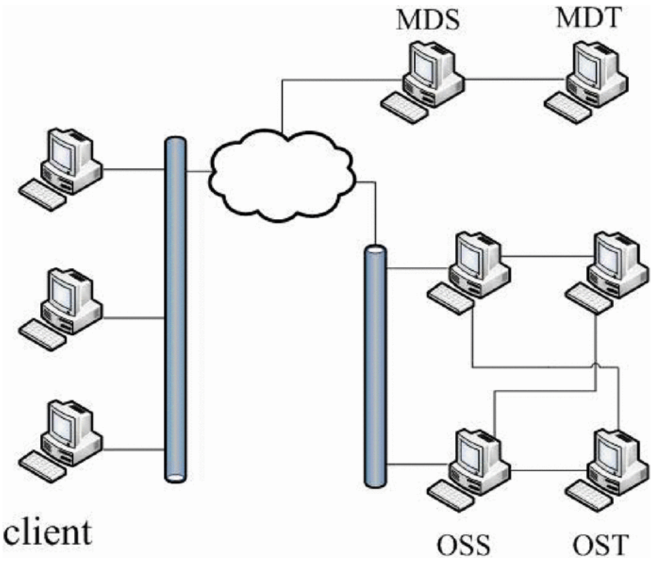
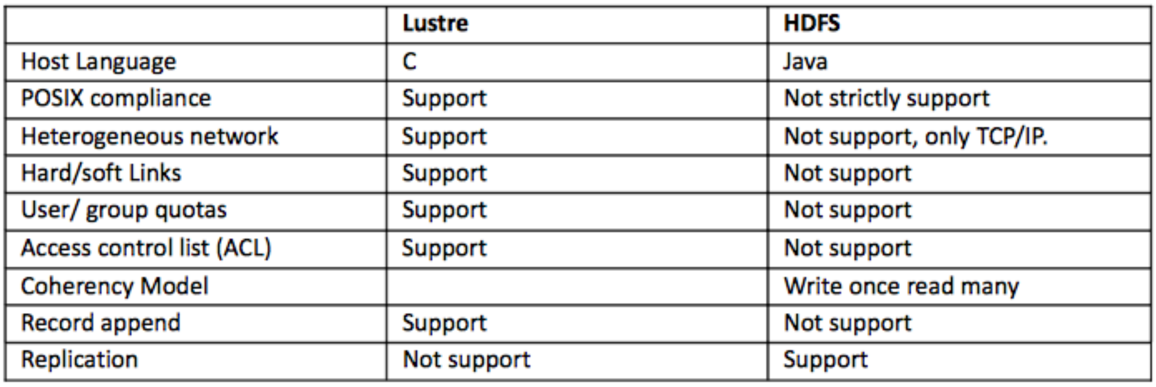


Figure 15. Architecture of Lustre [93].

Another comparison between HDFS, Lustre and Moosefs is presented in Figure 17, [93]. This benchmark evaluates I/O performance of Lustre, HDFS and Moosefs. This paper also evaluates more common features: installation complexity, data backup, failure support, and redundancy of data, etc. These requirements have to be taken into account before choosing the file system. File systems have their own advantages and are not suitable for similar tasks. For example Lustre can be used for massive writes, while HDFS is more suitable for massive reads.



	Lustre	HDFS
Host Language	C	Java
POSIX compliance	Support	Not strictly support
Heterogeneous network	Support	Not support, only TCP/IP.
Hard/soft Links	Support	Not support
User/ group quotas	Support	Not support
Access control list (ACL)	Support	Not support
Coherency Model		Write once read many
Record append	Support	Not support
Replication	Not support	Support

Figure 16. Comparison between Lustre and HDFS [92]

Read and write operations can be optimized with specific I/O operations. High distribution and high redundancy implies important I/O operations, and thus it is important to provide improvements to the system. I/O operations have to be tuned regarding to the hardware, but this methodology is time consuming. To avoid this long study, [36] presents a new library to avoid code complexity (adapting I/O regarding to the architecture), this library is named ADIOS. ADIOS API provides a simple standard I/O model, extended by a XML file, which described data and the process flow. This strategy gives to developers the ability to change execution parameters to his application using an XML file. This API provides suitable I/O operations regarding to the architecture. The source code does not require any changes. The XML file is used to change parameters of the execution. Adios has evolved, and seems to be maintained. Now, it is possible to add other features for the execution plan: manipulation of the workflow, visualization, and auxiliary tasks.

Some cloud architectures are able to handle data locality, they have to ensure a compromise between where the data is stored and where the data is processed. The current Hadoop runtime is based on a JVM and this virtual machine is not optimized to deal with the location of data and multi-core features on modern processors. In [47], the authors present a hierarchical approach that improves the data locality and the task parallelism. A cache system, based on RAM, is used to improve the location of data. This caching methodology enables to reuse data, and avoid networks and disks accesses. This system is presented in the Azwraith software, which is not available for download. The locality of data is also important in the case of heterogeneous environments.

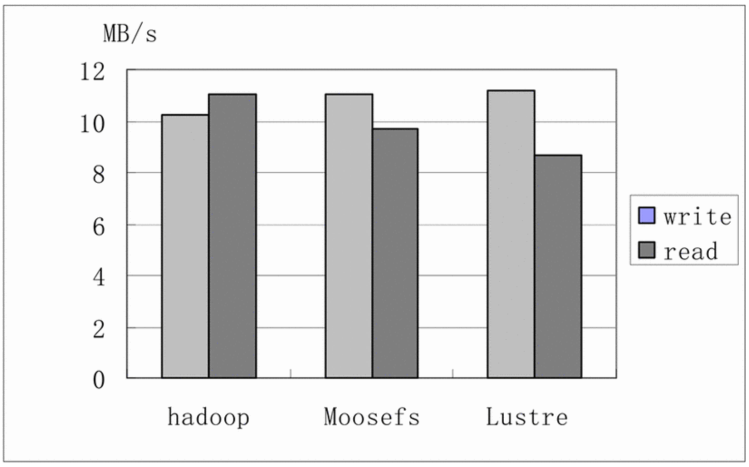


Figure 17. I/O performance comparison between three file systems [93]

Hadoop has been designed for homogenous clusters. Thus in heterogeneous environments, it is necessary to study the data locality to preserve a more efficient processing, and this issue is studied in [63]. This paper presents a load balancing approach to store the data on the appropriate computational nodes. The main idea is to improve processing operations, but moving large sets of data can lead to excessive network loads. Two algorithms are presented in this paper: an efficient initial placement in a distributed file system and a reorganization methodology. The first algorithm splits data regarding node capabilities. The second algorithm addresses dynamic load balancing. An evaluation of this approach is presented, and this study is applied on two well-established functions: *grep* and *word count*. Compared to the traditional Hadoop system, the proposed method can achieve better performance on heterogeneous architectures.

Storage for the Hadoop frameworks has largely evolved in the past few years. New file systems have been developed like CloudStore, which have been designed to run with Hadoop jobs [177]. The main HDFS branch has also evolved and now proposes new versions named HDFS 2.0 and presented in [116]. This new version brings HDFS to new dimensions by improving automatic failover and full stack resiliency. It also adds standard support for NFS read/writes to HDFS through a dedicate gateway: HDFS is becoming POSIX compliant. HDFS 2.0 also brings an encryption protocol for data transfer and snapshot capabilities for HDFS (save the current status of HDFS).

4.4. Data computation

In this section, we present how computation on cloud systems can be improved. This section is dedicated to the limitations and improvements for current computational models used on clouds. These improvements bring new capabilities to the actual cloud computation models, for example: SQL support, HPC capabilities or online-computing. MapReduce was one of the most traditional computational models for cloud systems. Without modifications, this model is not enough to express complex or scientific computing [25][29][60].

The MapReduce computation model has been designed to run batch jobs. These jobs are mining operations, like word count or extraction of specific contents. But this model is not designed to run scientific or more complex workloads: iterative computations, recursive computations, etc. In [82], the authors provide a classification of scientific problems. This classification is based on how a scientific problem can be converted into a MapReduce function. Four classes have been defined. The first class represents an algorithm, which can be adapted as a single execution of a MR. The second class represents algorithms that can be adapted as sequential executions of a constant number of MR. The third kind represents algorithms where the content of one iteration is represented as an execution of a single MR. And the last one represents algorithms where the content of one iteration is represented as an execution of multiple MR. [82] presents an evaluation of two runtimes: Hadoop and Twister. This benchmark evaluates which solution is the most suitable regarding to scientific workloads. Twister is an alternative MR framework designed to run iterative algorithms. The first and second classes are more suitable for Hadoop systems and data intensive algorithms. The third and fourth classes are more adapted to Twister.

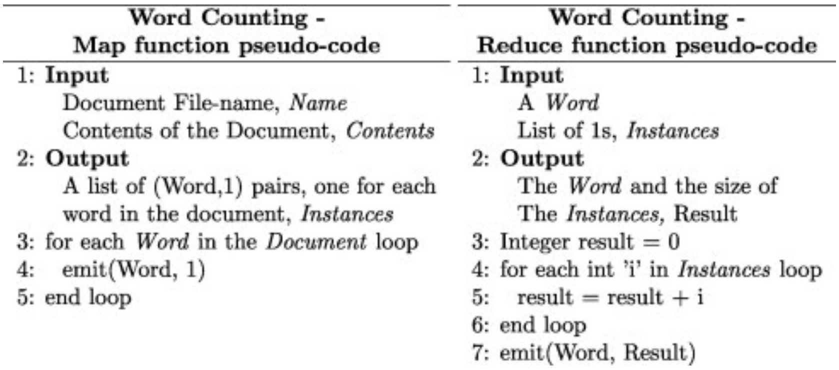
As shown in the previous example, identifying the right computational model is not a trivial task. And the adaptation of a MR paradigm requires understanding of the complete computation workflows. Specific computational models will have some difficulty to be adapted to the MR model, like recursive algorithms. This computational model delivers its output when all computations are applied at the end of the recursion. A failure may imply a complete restart of its tasks. In [59][59], the authors provide an efficient implementation of recursions in MR environments. But they are not able to fully implement this strategy: a complete optimized recursion of a MR function requires a considerable amount of work.

After the identification of the computational workloads, it is necessary to manage these jobs and also manage efficiently the accesses to the data. In [57], authors present HFSP: a new scheduler for Hadoop systems. This scheduler is based on the jobs size and on the evaluation of the required time to accomplish them. This evaluation is produced on-line. Authors introduce the notion of virtual time and age. This scheduler provides good performance for either large or small jobs.

The information retrieval is conducted by an efficient indexing strategy, and the choice of the best indexing methodology is not obvious. In [50], the authors present a study of four indexing strategies. Each strategy has been implemented on a Hadoop system and applied to different datasets. The first and second indexing strategies concern the solution presented in [89]. This indexing method can be implemented in two different ways. The first method is based on a map function which outputs a set of <term, doc-ID> pairs for each token. Then, for each unique doc-IDS, a reduce function aggregates unique terms to obtain a frequency measure. Figure 18 presents the pseudo code of MR functions for this specific method. The second strategy emits <term, doc-ID, term frequency>. This solution reduces the number of messages, as they are now done only once per unique term in each document. Figure 19 shows the pseudo-code of the indexing solution. The third strategy is presented in [50]: instead of emitting terms, this method tokenizes documents during the map phase. Then the reduce phase is in charge of writing all the index structures. Compared to the previous solutions, this method emits less information; see Figure 20 for the pseudo-code.

And the last one is presented in [205]. This strategy splits the process into multiple tasks; each map task is in charge of a subset of the data. This benchmark shows that the two first methods are not well suited (they generate important set of intermediate map data). The last method seems to be the most efficient: this strategy uses efficiently the local computer memory and tries to reduce the MR traffic (using compression strategies). The pseudo code of this strategy is presented in Figure 21.

To reduce the complexity of cloud programming, it is necessary to use existing or find new ways to express queries. The SQL language is well disseminated, and scientists have provided a SQL layer on top of Hadoop (queries are compiled into MR jobs). A recent report [42] explains that some companies also combine a Hadoop system with a traditional DBMS. This combination is used to increase performance of analytic tools.



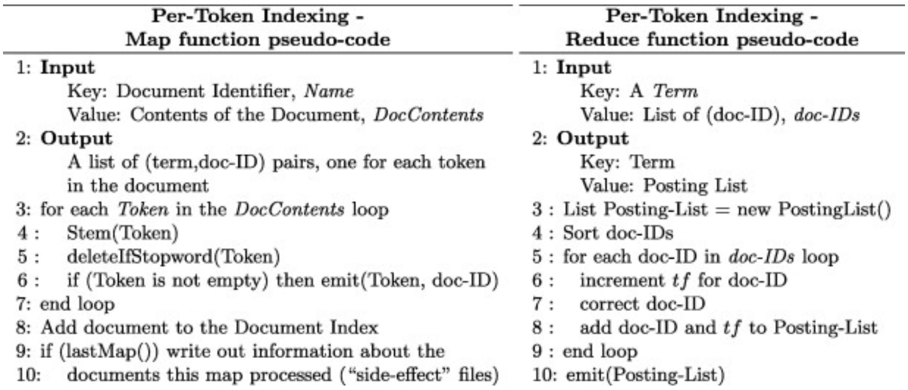
Word Counting - Map function pseudo-code

```
Input
    Document File-name, Name
    Contents of the Document, Contents
Output
    A list of (Word,1) pairs, one for each
    word in the document, Instances
for each Word in the Document loop
    emit(Word, 1)
end loop
```

Word Counting - Reduce function pseudo-code

```
Input
    A Word
    List of 1s, Instances
Output
    The Word and the size of
    The Instances, Result
Integer result = 0
for each int 'i' in Instances loop
    result = result + i
end loop
emit(Word, Result)
```

Figure 18. Pseudo-code for the word-counting operation, expressed as map and reduce functions [50]



Per-Token Indexing - Map function pseudo-code

```
Input
    Key: Document Identifier, Name
    Value: Contents of the Document, DocContents
Output
    A list of (term,doc-ID) pairs, one for each token
    in the document
for each Token in the DocContents loop
    Stem(Token)
    deleteIfStopword(Token)
    if (Token is not empty) then emit(Token, doc-ID)
end loop
Add document to the Document Index
if (lastMap()) write out information about the
    documents this map processed ("side-effect" files)
```

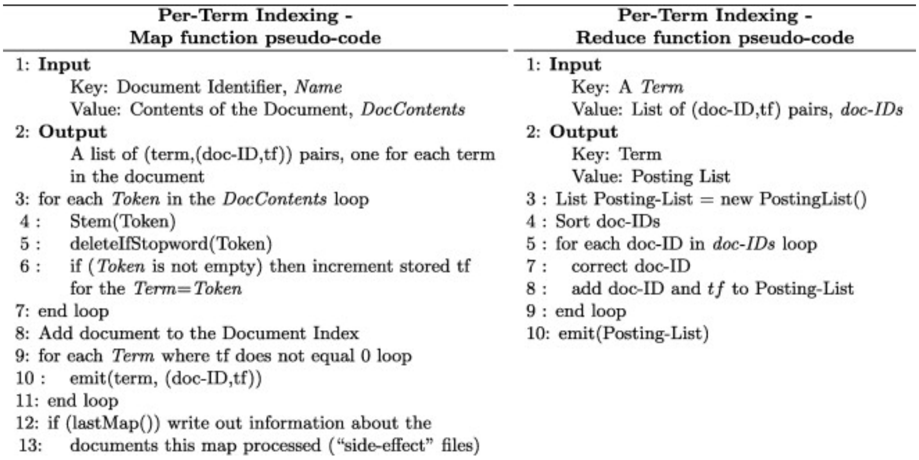
Per-Token Indexing - Reduce function pseudo-code

```
Input
    Key: A Term
    Value: List of (doc-ID), doc-IDs
Output
    Key: Term
    Value: Posting List
List Posting-List = new PostingList()
Sort doc-IDs
for each doc-ID in doc-IDs loop
    increment tf for doc-ID
    correct doc-ID
    add doc-ID and tf to Posting-List
end loop
emit(Posting-List)
```

Figure 19. Pseudo-code interpretation of the per-token indexing strategy (map emitting <term,doc-ID>) [50]

Hive has been presented to convert a SQL-like query into MR jobs. This SQL language is named HiveQL and can be used to define custom MR scripts. It can express computations into higher levels than MR. This language has been used on real Facebook datasets. Now, Hive has been integrated into the Hadoop framework. An improvement for Hive is presented in [138]. This optimization increases execution plans (increase performance of 100x compared to original implementation). It also introduces a SQL windowing function (Rank, Lead, Lag). Now, this optimization has been integrated to modern implementations of Hive

[139]. Hcatalog is another improvement for Hive [123]. This solution is used to easily read and write data on the cloud through Pig, MapReduce and Hive. Hcatalog can display data from any suitable format: RCFile, text, sequence etc. Display of the data is done through tabular views. It also provides a REST API. Hcatalog is now integrated into the Hive ecosystem.



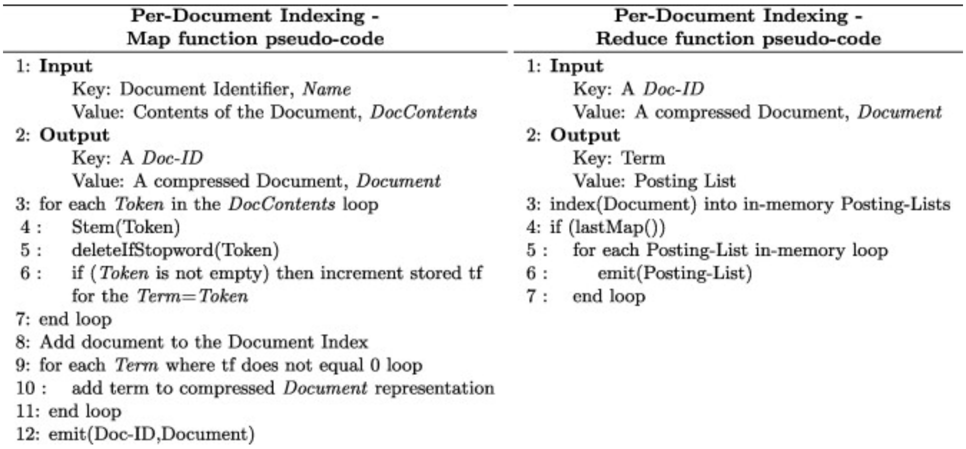
Per-Term Indexing - Map function pseudo-code

```
Input
    Key: Document Identifier, Name
    Value: Contents of the Document, DocContents
Output
    A list of (term,(doc-ID,tf)) pairs, one for each term
    in the document
for each Token in the DocContents loop
    Stem(Token)
    deleteIfStopword(Token)
    if (Token is not empty) then increment stored tf
    for the Term=Token
end loop
Add document to the Document Index
for each Term where tf does not equal 0 loop
    emit(term, (doc-ID,tf))
end loop
if (lastMap()) write out information about the
    documents this map processed ("side-effect" files)
```

Per-Term Indexing - Reduce function pseudo-code

```
Input
    Key: A Term
    Value: List of (doc-ID,tf) pairs, doc-IDs
Output
    Key: Term
    Value: Posting List
List Posting-List = new PostingList()
Sort doc-IDs
for each doc-ID in doc-IDs loop
    correct doc-ID
    add doc-ID and tf to Posting-List
end loop
emit(Posting-List)
```

Figure 20. Pseudo-code interpretation of the per-term indexing strategy (map emitting <term,(doc-ID,tf)>) [50]



Per-Document Indexing - Map function pseudo-code

```
Input
    Key: Document Identifier, Name
    Value: Contents of the Document, DocContents
Output
    Key: A Doc-ID
    Value: A compressed Document, Document
for each Token in the DocContents loop
    Stem(Token)
    deleteIfStopword(Token)
    if (Token is not empty) then increment stored tf
    for the Term=Token
end loop
Add document to the Document Index
for each Term where tf does not equal 0 loop
    add term to compressed Document representation
end loop
emit(Doc-ID,Document)
```

Per-Document Indexing - Reduce function pseudo-code

```
Input
    Key: A Doc-ID
    Value: A compressed Document, Document
Output
    Key: Term
    Value: Posting List
index(Document) into in-memory Posting-Lists
if (lastMap())
    for each Posting-List in-memory loop
        emit(Posting-List)
    end loop
```

Figure 21. Pseudo-code interpretation of per-document style indexing (map emitting <Doc-ID,Document>) [50]

HadoopDB is a hybrid MR system; this approach uses a traditional Hadoop system and a set of DBMSs. This solution has been proposed in [40] and takes both advantages of RDBMS and MR. This prototype is able to achieve similar performance compared to a parallel DB, and has the advantages of MR: scalability, fault-tolerance and flexibility. This strategy uses Hadoop and Hive. A schema of the architecture is presented in Figure 22. Queries are expressed in SQL, then they are translated into MR jobs, and finally this job is transferred to the right nodes. An evaluation of this system is provided, but this solution does not match the performance of parallel DBs. The authors hope to increase performance of HadoopDB with the next releases of Hadoop. Moreover, they have planned to use a column-store DB instead of the actual Postgresql DB. This implementation seems not to be developed anymore; the last update has been performed in 2013.

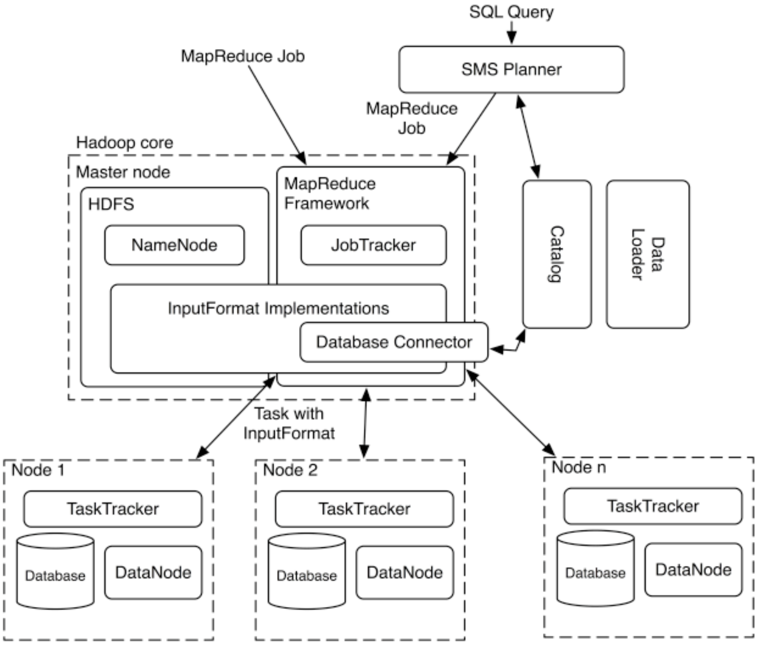


Figure 22. The architecture of HadoopDB

The traditional MR model is based on batch scheduling. Some scientists have extended this model to support online and hierarchical computations. Now, the Hadoop framework also provides a tool named Streaming [117]. This utility allows an user to create and run an application as a MR job. This application only needs to interact with the Hadoop framework through standard I/O (stdin or stdout). But this solution does not fill the critical requirements for interactive jobs: a query cannot be modified before the end of the computation.

An on-line MR has been presented in [80]. This method can pipeline data among operators. This strategy sends earlier information from MR tasks, thus users do not have to wait the end of a computation. With this methodology, continuous queries are available and events monitoring is possible. This methodology has been tested in HOP (Hadoop Online Prototype), but authors do not anymore update this software. Another continuous processing MR implementation has been proposed in [70]. A huge quantity of data can be assimilated to an unbounded data stream. The architecture presented in this paper is named S4; it is a distributed stream-processing engine inspired by the MR model and the actor model. Yahoo! has initially released S4, and now the development seems to be stopped, the last version has been published in June 2013.

Another streaming and distributed system is presented in [28]. It is the first streaming-mining application based on a subscription model. This solution can analyze streamed data by tracking and analyzing changes. In their paper the authors present implementation of SPC and provide an experimental evaluation. This communication strategy is not only designed to deal with stream operators but with any other operators. It provides a unique feature such as the flow specification that allows stream mining applications to catch ever-changing states of the data. Authors show that their solution can manage a large-scale application with a non-trivial structure.

Another requirement for the MR model is to handle hierarchical computations. Map and Reduce functions are not enough to express all the potential of complex queries. Node locations have also an important impact on the computational model, for example in public clouds, nodes are not necessary located at the same location, IT systems can be distributed all around the world. To answer these requirements, hierarchical methods have been developed.

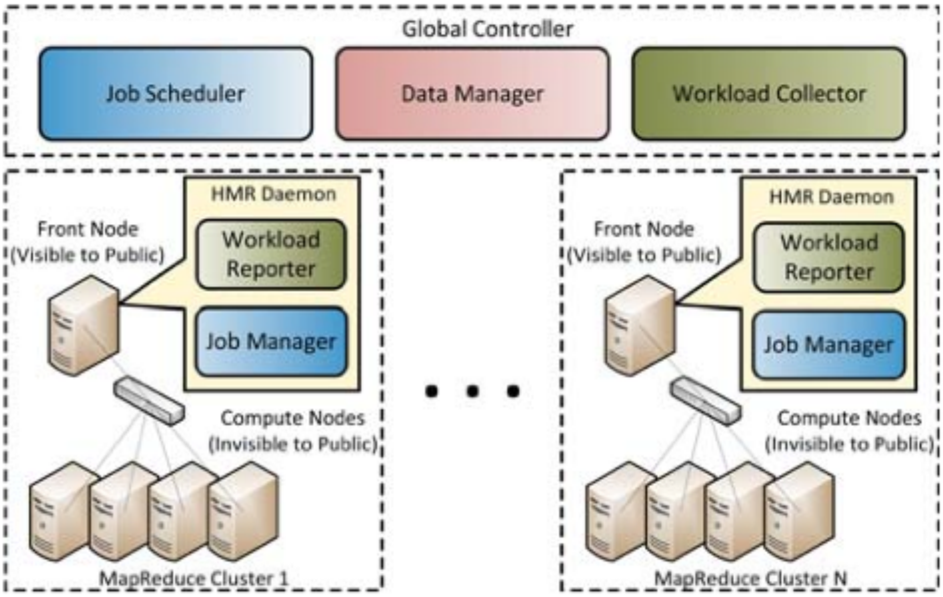


Figure 23. Map-Reduce-GlobalReduce model.

In [55], the proposed hierarchical method improves the reduction step. When a specific number of map operations have been achieved, a reduction operation is launched. Then, regarding to a tree structure, the system aggregates new results of the reduction tasks. This paper makes a comparison with the regular Hadoop model. This hierarchical decomposition of the reduction phase is more efficient than the traditional MR implementation. When applications use a public cloud, performance may vary during the runtime. These changes come from the global distribution of public clouds (hardware is potentially distributed all around the world). Thus, it is necessary to propose some efficient solutions to improve the traditional MR model.

In [39][53], a hierarchical MR model has been proposed, it can gather resources from different clusters and run computations across them. The proposed method is based on a Map-Reduce-GlobalReduce model. Thus, three layers compose this solution. With their hierarchical MR programming model, the authors are able to coordinate multiple clusters to act collaboratively. The proposed method has two components: a traditional HMR (Hadoop MapReduce, which is deployed over a cluster) and a Global Controller (which manages clusters). Figure 23 presents a representation of this architecture.

Another hierarchical approach is presented in [65]. This paper provides a set of recommendations to deploy this architecture. MapReduce software has to be deployed over the same local clusters to reach good performance. With a widely distributed system, MR is becoming less efficient. The authors of this paper present three new architectures to solve this issue: LMR (Local MapReduce), GMR (Global MapReduce) and DMR (Distributed MapReduce). The first strategy moves all the data into one centralized cluster; all computations are performed locally. The second method requires the deployment of a global MR cluster over all clusters. The last solution is based on multiple MapReduce clusters. These three strategies have to be used carefully regarding to: the network topology, node bandwidth and latency. From this paper, the DMR solution is more efficient than the LMR and the GMR: transfers of data are avoided. With zero-aggregation (MR output is the same size as the input) or ballooning (MR output is larger than the input) data, the LMR obtains better results than both other solutions.

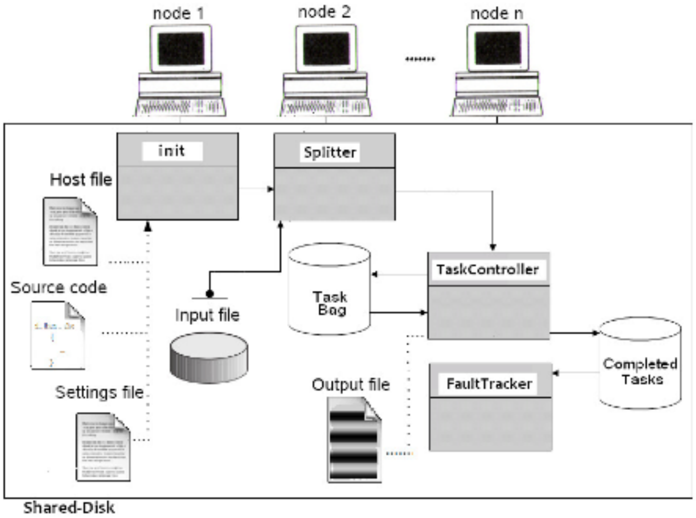


Figure 24. MARLA architecture.

Cloud systems are not necessarily designed to achieve high performance computing. These systems are mainly deployed to answer to high-density storage or deal with large numbers of users. Cloud systems can be used as a public facility (leasing resources) or deployed as private clouds. But in both cases, HPC and cloud systems are different entities [30][48]. HPC and cloud systems do not have the same goals, and these systems do not have similar computational power. Deploying these two IT systems is not simple, and these solutions are expensive. Thus, scientists have tried to find solutions to deploy cloud systems over existing HPC facilities.

MRAP is one of them. In [176], the authors analyze the computation gap between clouds and HPC. The authors have extended the traditional MR model to eliminate unnecessary pre-processing features needed for a transfer from MR to HPC. This solution provides HPC data analytics, which reduces the overhead of MR on HPC systems. This solution has been implemented using a dedicate framework.

In the literature, Hadoop has been delpoyed to HPC facilities [104][105]. For example, this has been realized using myHadoop [34]. This application runs Hadoop over a HPC system. HPC resources are different than a traditional Hadoop system. HPC deals with small and powerful hardware, while Hadoop frameworks run on large clusters with less powerful hardware. The last update of the original myHadoop framework was realsead in 2011. But a new version of myHadoop is available [100], and a full deployment instruction guide is presented in [101].

MR models have been implemented in other software as well; Dryad is one of them. This framework is introduced in [45] and was motivated by the emergence of large-scale Internet services. Dryad provides an execution engine, which can handle large distributed and concurrent applications. This application is based on several features: a specific scheduler, and a communication strategy. This scheduler was designed to manage computers and CPUs on clouds. The communication strategy is used for recovering information when a failure occurs. The communication between nodes is performed through a specific pipeline. The Dryad jobs are designed by a communication flow, jobs can be represented as acyclic graphs, where each vertex is a program and each edge is a data channel. It is a logical computation graph mapped on some physical resources. This application gives developer the opportunity to express queries as more complex structures than MapReduce, but the simplicity of the MR has been sacrified. Dryad has been recently released to support Hadoop ecosystem. Now, it is possible to use Dryad has an extension of Hadoop, all features are available for HDFS and YARN [236].

In [234], authors provide an evaluation of several cloud deployment over different facilities: AWS, Azure and GCE (google compute engine). This document provides all information and benchmarks of deployment of cloud system over these services. These benchmark shows that deployments of cloud system are more relevant with Linux OS, compared to Windows performances.

[235] owns more references concerning deployment of Hadoop ecosystem over HPC. They classified these distributions into four specific kinds: Commercial software (with IBM platform symphony), research project with myHAdoop, MR job adaptor. The third kind is named on the shelf with: MR+ and Hadoop on demand. And the last category is for Mesos and Yarn.

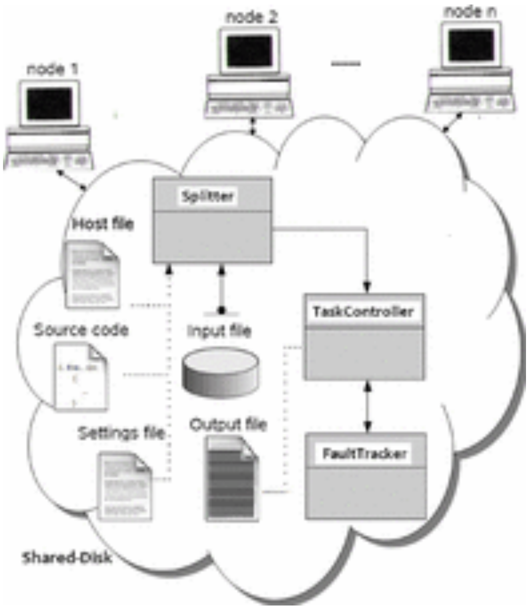


Figure 25. MARIANE architecture [99].

Another solution to deploy a Cloud on top of a HPC is presented in [187]. Here, authors have study how to integrate MapReduce jobs into a HPC cluster. Both systems use a scheduler to perform their tasks, but cloud systems do not required any parameters to launch jobs, while HPC jobs needs configuration to run. This solution is based around a MR adaptor, which provides a specific HPC workflow for this computational model. With this methodology, components are optimized to execute MR tasks.

However, providing extensions to Hadoop is sometime not enough. Thus MapReduce computations have been directly implemented on top of HPC facilities.

In [102], the authors present MARISSA. This solution provides a POSIX compliant framework, which can support iterative executions, the ability to run different applications on nodes, and to process different datasets on each node, etc. Restrictions of the traditional Hadoop system are avoided, and a deployment on a regular HPC is facilitated. This solution cannot be compared to new solutions, because this method is not available for download. The traditional MapReduce approach is used through the Hadoop framework. But, it is based on HDFS, which was not POSIX compliant. Thus a deployment on a traditional cluster is not an easy task. In [103], the authors provide a solution named MARLA. This method has been designed to provide a MR framework, which supports POSIX features. Some heterogeneous nodes compose traditional HPC systems. This paper presents a dynamic load-balancing strategy on top of a MR application. The architecture of Marla is composed by three components: Splitter, taskcontroller and faulttracker, see Figure 24. The MARLA framework does not use a chunk strategy to store the data. As stated in the paper, MARLA is able to reach a higher efficiency than the regular Hadoop framework, on a HPC. MARLA is not directly available, and the maintenance is not clear.

In [99], the authors present a modern approach to use MR model on traditional HPC. HDFS is not POSIX compliant, thus it is necessary to develop a new MR framework adapted to a HPC context. From this assumption, authors have released: MARIANE. This solution does not only provide a MR implementation over a HPC files system, it is also brings better performance. MARIANE is designed to work with various clusters, shared-disk solutions, POSIX file systems and parallel file systems. The architecture of MARIANE is presented in Figure 25. Three modules manage the whole system: a Splitter, a task controller and a fault tracker.

With the redesign of MR, the authors avoid to use an additional file system like HDFS. [99] shows significant improvements in term of performance compared to the traditional dual file system methodology. Unfortunately this API is not directly available for download, and MARIANE cannot be used in combination with the original Hadoop framework.

The last solution is a new implementation of the MR library. In [173], the authors have developed a solution based on message passing (MPI). The library described in this paper is a lightweight implementation of a basic MR. The motivations to create this library were to provide a solution which enables graph algorithms on for MR frameworks. Several algorithms has been implemented with this library, the goal is to validate all features of this solution.

Several tools have been produced regarding this computational model, see [106]. In [198], developers have proposed a MapReduce implementation based on bash programming. A JavaScript implementation also exists and was named: Meguro [198]. This tool can be used to extract content and traces from datasets. A MPI implementation has also been ported: MR-MPI [197]. This implementation provides a C/C++ and Python wrapper, and it has been designed to provide MR to distributed parallel machines.

Another MR library is named FlumeJava. It is a Java library designed by Google. This library can be used for more complex problems by creating specific pipelines. This Library is presented in [192]. This library optimizes the execution plans of the applications; the library abstracts details of how data is represented and where the data is stored (Mysql, Bigtable, etc).

Phoenix [41][51] is another reimplementation of MR. This software has evolved with 3 different distributions (1, 2 and ++). Phoenix now provides a full implementation of MR model with modular properties. It can be used efficiently to support particular workloads. This MR implementation is suitable to run on shared memory machine, and also run on multi-core architectures. Compared to the MR implementations on Hadoop, Phoenix is able to achieve higher performance. The implementation was done in C++. Their optimizations are designed to improve runtime on NUMA systems; they are able to achieve average speedups of x2.5 and peak speedups of x19.

4.5. In the case of VELaSSCo

VELaSSCo has to follow some of the best practices highlighted in this document.

The platform has to deal with hardware optimization. These optimizations have to deal with multi-threading and hardware extensibility. Thus to deploy the system, a suitable solution is to use commodity hardware. This solution can be used to increase the capacity of the cloud infrastructure. It is an economic solution, which can be used to deal with more computational and storage nodes.

On top of this commodity system, it is necessary to deploy virtual machines dynamically regarding to the computation requirements. Load balancing of the system has to be dynamic; the platform manager is in charge of the creation or release of virtual machines.

In the VELaSSCo project, simulation engines output their results into files. Thus, to simplify the development of our storage platform, a solution can be based on file system. Alternative solutions can also be based on database systems instead of accessing files, and computational models could be based on higher languages like SQL.

As stated in the computing section, it is necessary to support scientific computational workflows for the VELaSSCo platform. The current requirements do not consider heavy computations. But with an extensible approach, it is possible to support future needs later. For example extensions can be: online queries, recursive and iterative algorithms. Moreover, instead of using this platform with fixed computational facilities, it is suitable to connect it to a HPC system.

Finally, the data that is stored in the VELaSSCo platform will be composed by different time step for the same inputs. Thus, storage needs to be studied for redundant information.

In the next section, we present some clouds frameworks designed mainly to store files, to compute information and for both cases.

5. Cloud tools

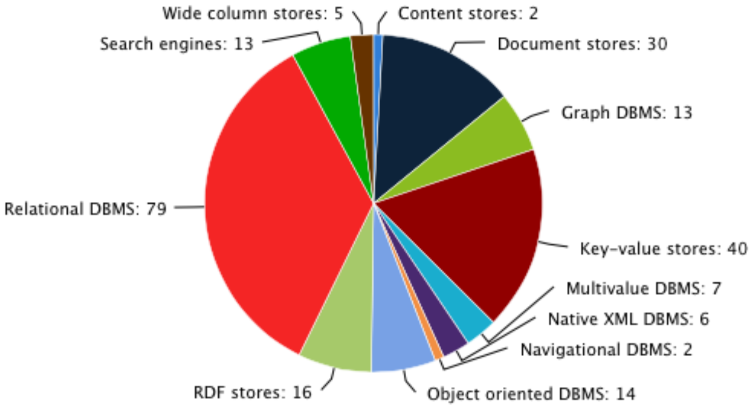


Figure 26. Shows the number of references DBMS in Db-engine.com

The recent interest for cloud systems led to the development of a large number of software stacks. Thus, cloud software can be classified into three different kinds: private, public and open source. Private software is described in scientific papers, but these solutions are not accessible to the community, a well-known example is BigTable from Google. The second kind represents tools, which are sold by companies. The last kind regroups stacks brought by the community.

5.1. Storage

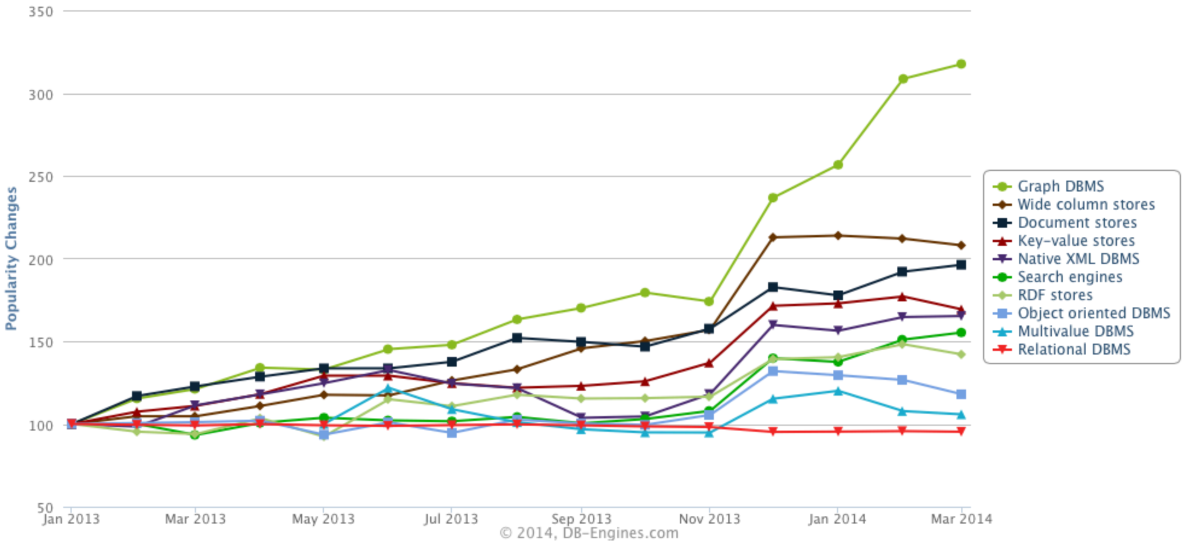


Figure 27. With this figure, db-engine shows the popularity of some DB-engine.

Cloud systems use mainly two methods to store data: file systems and databases. A benchmark of both solutions is not a trivial task: their storage and compute model are too different. Thus to provide a benchmark, it is necessary to produce adapted tools.

Firstly we will focus on DB systems. The web site: http://db-engines.com, provides an interesting comparison between a large set of DBMSs. This website compare SQL and NoSQL DB. It has been developed to give advices on DBMSs. As stated in [98][98], NoSQL systems are not so easy to sort. Most of DBMSs are based on a single data model, but the recent NoSQL systems use multiple models to increase performance. This trend comes along the fast growing of datasets: NoSQL DBMSs need to find new solutions to deal with this massive amount of information. DB engines provide comparison of some features, for example: license, implementation language, DB model, SQL support, transactions, MapReduce support, etc.

The website also provides two interesting statistics presented in Figure 26 and Figure 27.

Figure 26 shows the huge number of RDBMS compared to other classes. All other classes can be regrouped in NoSQL DBs. This specific type of DB represents now a large set of database systems. Figure 26 gives an overview of which NoSQL models are available. The second figure (Figure 27) represents the popularity of DB engines. This figure presents the popularity growth of NoSQL DBs. Another overview of a set of NoSQL DB is presented in [67].

Different methodologies can be used when a computer scientist wants to use a DBMS to store cloud data. The first solution is to use a traditional RDBMS, while the second one use a NoSQL DB.

The first example presented in here is named ScaleDB (http://www.scaledb.com). It is a MySQL plugin. This tool extends a MySQL DB into a shared-disk clustering DB. With this extension, DBs can perform simultaneous read/write operations on data: any node can reach any data. ScalDB has a special indexing mechanism to deliver superior performance. This indexing strategy removes the necessary use of a master/slave paradigm.

A second solution is named OrientDB (http://www.orientechnologies.com/orientdb/). It was developed to store both documents and graphs. This NoSQL DB uses a SQL layer to query the data. This DB can perform high-speed queries using a specific tree structure.

ThruDB is a service built on top of Thift's framework (https://code.google.com/p/thrudb/). It provides a specific indexing and a document storage service. This system has been developed to store website data. This DB service works with multiple storage back-ends: for example it can be deployed on EC2. But this system is no longer developed.

Dynamo is a new key-value store approach presented in [23]. This solution was built on top of the S3 Amazon service. It is provides a high-level of availability (an "always-on" experience). This software is based on an extensive use of object versioning. Data consistency is facilitated by versioning. Dynamo has already been applied on Amazon services. Dynamo provides improved technics for partitioning, replication, versioning, membership, failure handling and scaling. For the partitioning approach, this method is based on consistent hashing, which provide incremental and scalable performance. A vector clock provides high availability which guaranty read consistency with write operations. A Merkle tree is used to recover data from permanent failure and finally, membership and failure detections are provided by gossip-based membership protocol and failure detection.

Data stored in cloud systems have also evolved. Two specifics datasets have been massively used: spatio-temporal and graph datasets.

In [52], the authors present a new indexing structure developed for geographical data sets. This solution was based on top of a NoSQL DB. Some databases have been specially developed to store geographical data into their DB (with Geohashes). [52] presents a new non-relational distributed DB, inspired by BigTable. This NoSQL DB is named Accumulo. To improve search for spatio-temporal data, authors have proposed a specific index structure based on a geohash. This approach allows to compute keys for each piece of data independently. With this strategy, filters over data can be improved; for example, temporal data is included into the index of data.

Another example, PNUTS is a massive parallel and geographically distributed DB system presented by Yahoo! [75]. This system provides a data storage organization based on a hashed and an ordered table. This system supports all cloud requirements: load-balancing, failover, central management, etc. The data model of PNUTS is presented as a simple relational model. This system achieves redundancy at multiple levels: data, meta-data, serving components, etc. The communication system is based on a publish-subscribe message system.

Due to recent changes in Internet usage, scientists have to deal with graph-based datasets. This new data is produced by social networks like Facebook, Twitter, etc. These social networks produce each day huge amounts of new information. In [24], authors analyze how several applications deal with large social graphs. Comparison is based on 8 applications: MySQL, Redis, Riak, MongoDB, CouchDB, HBase, Cassandra and Neo4j. MySQL is representative of Relational Data Base Model, Redis and Riak are key-value repositories, MongoDB and CouchDB use Document to store data, Cassandra and HBase use a column oriented model and Neo4j is a graph DB.

All these storage systems have their own query language and processing capacities. This study shows that the most suitable DB for graph data is Neo4j. This DB is designed to deal with graph and not social data, thus constraints on query and processing of social data arise. The second problem of Neo4J is the important size of the social data.

The nature of social data (semi-structured) is the main problem for RDBMS, join queries are complex to perform. Moreover, RDBMS are not suitable to perform computation on complete datasets. Tested Key-value DB software is not able to apply query or process over data. Document and column stores suffer the same problem. From [24], column store seems to be the best choice to store social data.

Instead of using DBMSs, some approaches use simple file system as a storage system. One solution proposed by the science community is based on large virtual FS, which enable to deal with big data.

The Google file system (GFS) was introduced in [215]. This system is designed to store data on a virtual file system; this data is used for web crawling. It has been optimized for processing data required for Google applications. This system has been designed to be fault tolerant (standard computers are used and failures can occur frequently). GFS splits files into tiny blocks to store information. Each block is stored among different nodes. These tiny files are named chunk and their size is limited to 64 Mo. A unique key and version identify each block (version is used to avoid update problems). Each file is copied at least 3 times across the datacenter. GFS is a private software, regarding to statement of this method, HDFS has been developed.

GFS and HDFS are virtual file system; they do not use low-level storage systems, and direct access of the disk is not provided. Both systems are designed to be transparent to the end-user. But these FS have not been developed to be POSIX compliant, thus a traditional tool to manage files are not suitable. To reduce complexity of file access, some improvements have been performed for HDFS. For example, HBase provides random and real-time read/write access to the HDFS. But virtual file systems are not well suited to run with HPC, it is necessary to use POSIX compliant FS. An example of a *real* file system is presented in [37]. This FS is named parallel NetCDF (http://en.wikipedia.org/wiki/NetCDF). It has been designed to fit especially with scientific data. This system stores data into an array-oriented methodology.

HDFS has also been improved to be used with HPC facilities see [118]. Here authors present a NFS gateway, which extend the actual Hadoop ecosystem. This solution brings to HDFS support for the NFS protocol. Figure 28 presents how this gateway works. It can be deploy on several nodes, to increase the availability of NFS systems. With this strategy, HDFS cluster can be mounted on any machine and interaction with stored files is possible through standard command lines, script or UI explorer. Some improvements are scheduled to manage with the security of this gateway and support for other protocols (like NFS 4).

Another approach to improve storage system is based on hardware. In [147], authors describe a new scalable design to store a distributed FS. This solution is based on the combination of SSD and mechanical HD. In this project, the Bigfoot team has performed a benchmark to evaluate the best strategy to store data (HDD-only, SSD-only and a hybrid SSD/HDD storage configuration). This paper also presents an efficient strategy to store data supporting data locality.

Microsoft has proposed a prototype implementation of a distributed data storage system, [79]. This tool is called Boxwood. This project explores the use of data structure or abstraction at the storage level. In this case, storage is considered as a simple abstract layer. This strategy covers all requirement of Distributed file system.

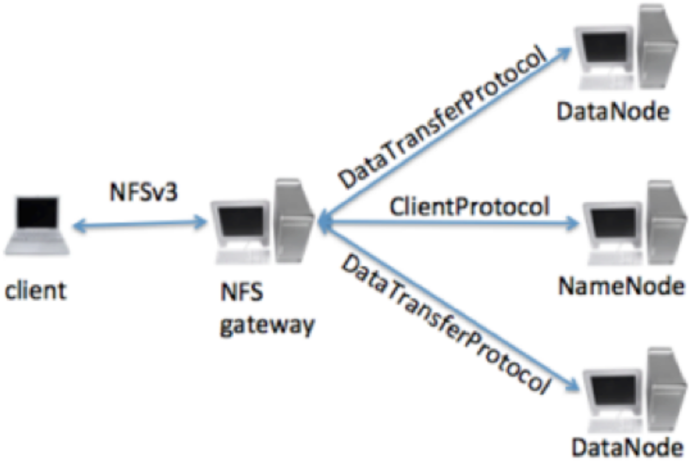


Figure 28. NFS gateway for HDFS.

5.2. Computing

Cloud systems can also perform computation over data. Some examples of these computation functionalities are: information retrieval, data analyses, statistics, etc. The traditional computational model used on cloud system is the MapReduce. But this model is too simple, and some scientific workloads need more complex algorithms than a simple MR. In this section, we present four specific scientific workloads extracted from literature. These workloads are: iterative, recursive and graph algorithm. The last part concerns machine learning. Our first example is based on a set of tools for bioinformatics applications. In [66], authors present a new solution called CloudBLAST. It provides a full distribution dedicated to bioinformatics requirements. This solution brings all tools to run bioinformatics applications on a distributed system.

Specific algorithms are also developed, in the next part we will present iterative algorithms. These algorithms need to re-run the execution of a specific function multiple times. Original implementation of MR is not optimized for this purpose.

Twister (http://www.iterativemapreduce.org) is a solution to perform iterative executions of MR in a more efficient way. Twister is new MR runtime which enhances the actual MR model. The authors have added the support for static and variable data, it is possible to cache MR tasks. This solution is based on a publish/subscribe paradigm for communication. Authors have also improved the combination phase which grab data from many reduce phases, etc. This application is a lightweight implementation compared to the traditional MR in Hadoop. And this tool is suitable with any regular MR jobs, and the associated management tools.

Another solution presented in this document is named HaLoop, [90]. This solution can be downloaded at: https://code.google.com/p/haloop/). HaLoop extends the MapReduce model by providing a programing support for iterative applications, caching options for loop invariant data access and it is designed to reuse major building blocks from Hadoop. This solution also supports similar requirement than the current Hadoop system (reliability, distributive system, etc). To evaluate their solution a benchmark has been presented in [89]. They compare their solution with a traditional Hadoop system. Conclusions of this paper show that HaLoop is able to compute more efficiently iterative algorithm than Hadoop.

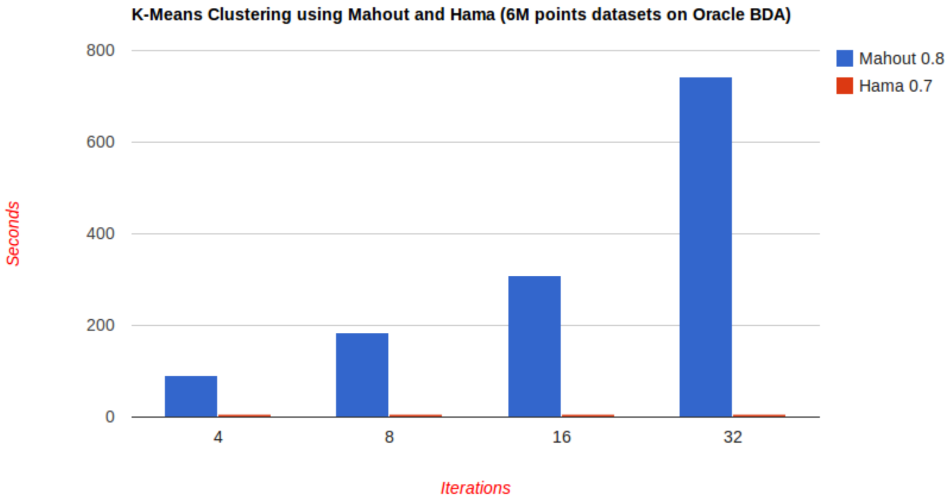


Figure 29. K-means Clustering using Mahout and Hama [130].

The traditional MapReduce model was in most case not suitable for machine learning. ML has some specific workloads, which cannot be splitted into efficient MR jobs. Mahout has been developed to fill the gap of machine learning (ML) with Hadoop [96]. This software can apply machine-learning techniques in Hadoop. These machine-learning problems are limited, and Mahout only proposes a sub set of ML problems: a recommendation engine, a classification engine and a clustering engine.

Apache Hama (https://hama.apache.org), is another ML tool based on top of the MR model. This method provides a pure Bulk Synchronization Parallel (BSP) computing engine to perform complex analyses. This model is based on three components, which can express a wide range of scientific problems. In a comparative benchmark, the authors show that this solution is more efficient than Mahout: [130]. BSP is an answer to MPI problems: data partitioning and data locality aware for task scheduling, job fault-tolerance, deadlock and race condition avoidance and complexity of interface.

In [73], the authors present a use-case of Hadoop frameworks for a machine learning workloads. This system uses PIG as an analysis tool. A recent version of PIG is able to provide predictive analysis. This methodology incorporates machine-learning capabilities to PIG and is mainly focused on supervised classification.

Graph algorithms have also evolved to run on Hadoop platforms.

In [30], the authors present a computational model for large graph problems. The Massive use of social networks brings graph problems to the Hadoop ecosystem. Efficient processing of this data has become an important challenge. This application has been designed to express programs as sequences of iterations, in which the computation is done using a message passing approach centered on vertices. This model has been designed to be efficiently scalable and fault-tolerant.

Dealing with DAGs (Direct Acyclic Graphs) is also a difficult task for MR. Tez, presented in [113], has been designed to handle interactive queries. Tez is a speed layer for Hive and Pig. This solution removes unnecessary tasks, synchronization barriers and useless HDFS reads/write. Tez is new software, and it was not yet widely used by the community.

Pegasus, presented in [31], is a petascale graph-mining library over the Hadoop platform. Authors address a solution to design an efficient MR algorithm able to handle massive graphs. Authors have observed that mining operations on graphs can be described by a repetition of matrix-vector multiplications. They define a primitive called GIM-V, which is a highly optimized Matrix-vector multiplication iteration. To validate this approach, a benchmark has been done on a super cluster [32]. PEGASUS is a scalable algorithm, which can mine billion-scale graphs. Pegasus can achieve 43x smaller storage, and 9,2x faster run time, compared to a naïve solution. With their tool, they have been able to discover some patterns and anomalies (for example validate 7-degree of separation in a web graph).

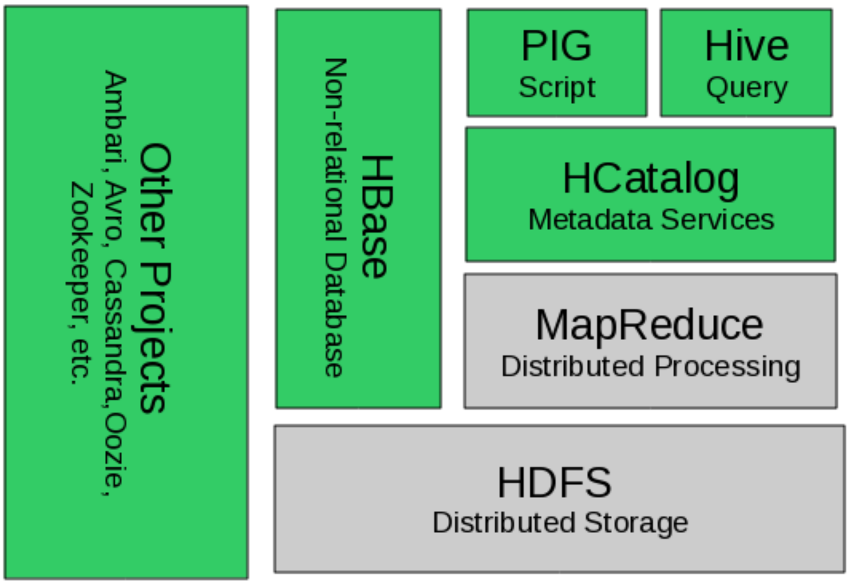


Figure 30. Hadoop 1.0 ecosystem.

Finally, we present a specific evolution of the MR model, which has been implemented in Hadoop. To enable more complex computation, Yahoo! provides a resource scheduling and negotiator to Hadoop.

The main evolution of Hadoop has been introduced with YARN. As stated in [42][44], this update brings Yarn and a new MR workload: MapReduce v2.0. YARN is the modern computational engine for Hadoop platforms. This evolution brings a new architecture composed by a programming model, a resource management infrastructure and a scheduling function. This evolution brings a new computational model to Hadoop; MR is no longer the only available computational model. The programming model provided by YARN is able to coordinate application communication, execution flows and dynamic optimization. In [44], the authors show that the decoupling strategy is efficient, and provides greater scalability, higher efficiency and enable different frameworks to efficiently share a cluster.

As stated in [155], Yarn can be considered as an operating system for big data. With this tool, Hadoop can deal with any kind of computational model; MR is not anymore the only one suitable computational model. In Hadoop 2.0, the JobTracker task is splitted in two major components: the RessourceManager and the ApplicationManager. In this document, the authors present a comparison view of both solutions; see Figure 30 and Figure 31.

Using Yarn is not an easy task. In [111][112], the authors present a solution named Reef; this tool has been designed to run on top of Yarn. This extension allows running efficiently some specific workloads: complex transformations, iterative machine learning or declarative query processing. Reef uses a persistent cache system to reduce computation time. Yarn is not the ultimate solution, Reef developers bring the last stone to this framework: retainability (increases performance of iterative and workflow computations), composability (writes in a component oriented for reuse), cost modeling (store data locally with a coarse-grain parallelism), fault handling (including checkpointing of tasks), elasticity (dynamic adaptation as available resources). With Reef, development of applications based on Yarn is simplified. This software is open-source and frequently updated.

DataTorrent is a cloud provider, which bring interactive and real-time query on its own distribution of Hadoop, [126]. They have produced a tool, which simplify the development for running real-time applications on Hadoop. It is a commercial software, which can handle with prediction of load, control and data flow.

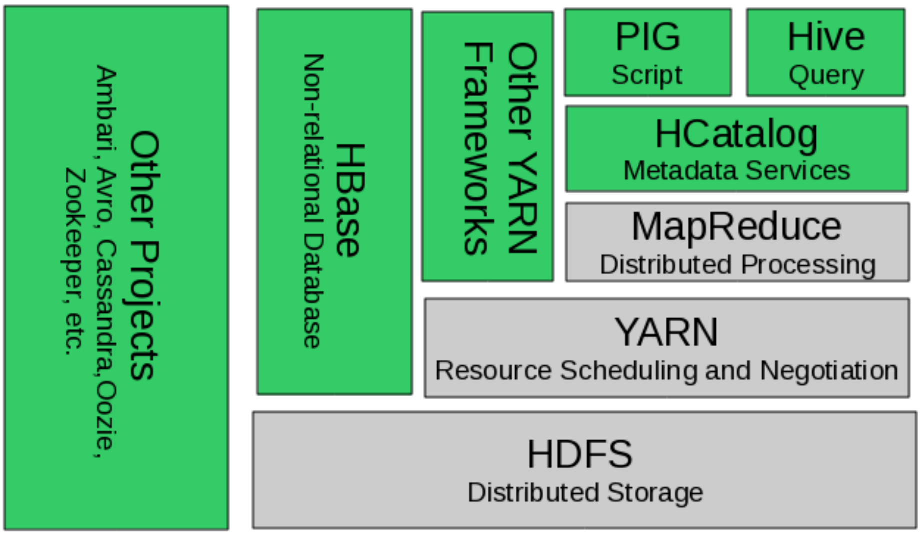


Figure 31. Hadoop 2.0 ecosystem

Limitations of the actual Hadoop ecosystem have also been highlighted by other actors of cloud computing. And similar solutions have been developed by Twitter (Mesos), Google (Omega), Facebook (Corona) and Microsoft (Cosmos). These systems are based on the same behavior than Yarn. They try to match similar requirements than Yarn: scalability, latency and a more flexible programming model.

Omega [224] is designed to deal with distributed, multi-level scheduling. This solution is mostly designed to be suitable to run on a closed source environment with specific resources, which are well known. This solution is only designed to run with well know parameters, and there is no room for heterogeneous architecture, and software. Corona [222]is based on a communication protocol instead of a control-plane solution used in Yarn. Other solutions are based on the similar behavior than YARN like Mesos. This solution is based on a two layers scheduler. Mesos [223] is an offer-based resource manager while Yarn uses a request-based method. The second difference is on usage of task scheduler; Mesos is similar to the regular Hadoop strategy (a pool of central schedulers) while Yarn is enabling late binding contender. Cosmos is the closest solution regarding to Yarn architecture. Cosmos has a main difference: it does not own a central resource manager.

5.3. Storage and computing

To provide more suitable solutions for cloud systems, the most efficient methods are to produce optimized solutions for both storage and computing, [25]. This last section presents some framework designed to improve all the features of a cloud system.

SciDB (http://scidb.org) is a data management and analytics software for scientific research. This DB includes most of the needed features for scientific DB: versioning, provenance tracking, support of uncertain data with error bars, [62][82]. SciDB has been designed to involve large datasets produced by scientific applications. In [62], the authors present specification of this DB, which is based on a new storage manager, array data model, query language and an extensible framework. The data model used for this DB reflects the common scientific requirements, use-cases, which determine the needs of this DB. They are presented here: http://scidb.org/use/. This DB is organized as collection of-n-dimensional arrays. Each value in SciDB can express any kind of numerical data, or a fixed string length. But scientific data also bears some error intervals, thus each of SciDB can itself contain another SciDB arrays. This DBMS have been developed to adopt a shared-nothing design. Thus, SciDB can be deployed over a network. Versioning features of this DB are benchmarked in [68]. The features are implemented using the update function, each update creates a new array version. This strategy is useful to make comparisons with data at different times. A strategy is used to reduce data space required for the system.

DataBase Managements Systems have also largely evolved, and this evolution has conducted these systems to deal with improvements on the computational models and storage facilities. VoltDB (http://voltdb.com) is a DB system that deals with high velocity data. It provides a high performance database with ACID-compliant capabilities and also provides JSON flexibility. It uses memory and disk to store the data, and transactions are performed in a couple of milliseconds. Analytics tools of this DB have efficient methods able to provide speed queries and the ability to apply automatic decisions. This DB can be deployed on a shared-nothing system.

Terrastore is a document store DB, with advanced features on scalability and elasticity (https://code.google.com/p/terrastore/), but the consistency has been eluded from this DB. This software is based on Terracotta (which provides fast clustering technology). This DB supports single- and multiple-cluster deployments. It is a schema-less DB, which provides some standard processing features: range, MapReduce, update function, event processing, etc.

Cloudata (http://www.cloudata.org) is a distributed large-scale structured data, inspired by the traditional Google BigTable. It is a DBMS system, without a relational data model. This DBMS support various file systems (not only HDFS). It provides some computational services: single row operation, multi row operation, data upload, MapReduce, and simple queries.

5.4. Future of clouds

The traditional MR framework is showing its limits, [189]. It is time to use another programming model. The bottleneck of this approach is the JobTracker: to avoid problems linked with this function, Hadoop 2 has been developed. Hadoop 2 is based on Yarn, which splits the JobTracker into two separate daemons. This new framework keeps all features of Hadoop 1: reliability, availability, scalability, etc. Moreover, this new version brings new capacities: support alternative programming paradigms and support for short-lived services.

The actual MR framework has also reached its limits (Around 4000 computers), [54]. The actual Hadoop system imposes to upgrade the whole IT system for any minor or major changes. All nodes of a cluster need to be updated at the same time. The next generation of MR needs to deal with high computability.

Scientific workloads have also specific behaviors; most of them are MTC-based (Many-Task computing). In [71], authors proposed an empirical evaluation of performance of four commercial cloud services for scientific purposes. Clouds systems claim to become the next generation of scientific computation tools instead of the current grids, clusters, etc. For their study, authors also evaluate the cost of such tools for scientific purposes. In this paper, authors provide a benchmark between different infrastructure: Amazon EC2, GoGRID, ElasticHosts and Mosso. Conclusions of this study show that actual cloud computing services are not enough for scientific computing, but it can be a temporary solution to store and compute data.

Big Data brings new interesting questions; one of them is presented in [190]. In this paper authors have evaluated usage of a modern clusters compared to a traditional commodity server used for Big Data. They evaluate 11 jobs, which represent traditional Hadoop jobs and they show that the HPC solution can be competitive in any case. From this statement it is necessary to evaluate in the earlier day of the project the evolution of hardware and also which jobs will run on the Big Data system. Cloud infrastructures have to scale-up and -out at the same time to reach the highest performance.

6. Applications

In this section, we present some use-cases of big data architectures, then we show some commercial and non-commercial products.

6.1. Use cases

Big data has been used in a variety of domains. Now, scientists do not try anymore to filter data, they usually store information, and try later to extract the content and concepts from them.

Risk analyses are an use-case of big data. In [168], authors present Hadoop integration to predict risk from huge datasets. They use a Hadoop framework instead of a RDBMS due to the traditional bottleneck of standard DBMSs. Most risk analysis solutions are based on ETL, but these systems are no able to deal with billions of records. Analyses provided by the authors show that the Hadoop solution is more suitable for simple risk analyses queries. For complex queries, it is necessary to adopt another storage framework, the access speed provided by Hadoop can be too restrictive.

Storage of files has been one of the most prevalent usages for cloud. These systems allow users to store files, and retrieve them anywhere. Files are accessed by network connections. Moreover, this solution provides to users a fully secure file storage system. This security is brought by data versioning and redundancy.

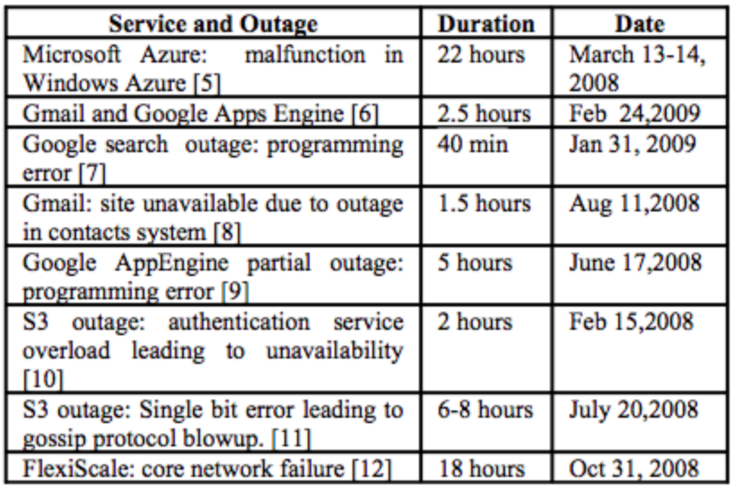
Cloud systems can also be used to deal with seasonal activities. This issue is a typical use case for resellers. And it is necessary to provide the most scalable solutions, regarding to the user flow and network traffic. The activity flow constrains the deployment of IT systems. All the hardware will not be used if the activity is low. Thus, it is necessary to deal efficiently with activity, and find a suitable solution to support elastic workloads. Amazon is a leader in on-line sales. This company has to deal with periodic annual peaks. Amazon has deployed a huge cloud solution, which can extend dynamically the IT system. When the planned workload is low, they can lease the unused parts of their cloud systems.

Cloud systems can be used to deliver CPU time to the users. This approach can extend the computation power of existing IT systems and can provide the necessary tools to extend a system by providing CPU monitoring. Clouds can bring thousands of nodes with at least the same number of CPU.

Cloud systems are also designed to solve network latency problems. Regarding to private, public or hybrid cloud, network capabilities are totally different. The computational pipeline used can avoid latency between nodes of these systems: multi-queries over a cloud system is an efficient solution to reduce high latency issues.

Using cloud architectures can reduce drastically the cost of IT equipment. Buying modern HPC facilities is expensive. And this kind of IT equipment has to be updated regularly. Cloud systems can be leased, thus it is not necessary to buy a complex HPC infrastructure. Moreover, deploying a cloud solution can be a cheaper solution if computational efficiency is not necessary.

Cloud systems also provide fully reliability. Outage of these systems is drastically controlled and service down time does not often happen. Examples of outage are presented in Figure 32.



Service and Outage	Duration	Date
Microsoft Azure: malfunction in Windows Azure [5]	22 hours	March 13-14, 2008
Gmail and Google Apps Engine [6]	2.5 hours	Feb 24,2009
Google search outage: programming error [7]	40 min	Jan 31, 2009
Gmail: site unavailable due to outage in contacts system [8]	1.5 hours	Aug 11,2008
Google AppEngine partial outage: programming error [9]	5 hours	June 17,2008
S3 outage: authentication service overload leading to unavailability [10]	2 hours	Feb 15,2008
S3 outage: Single bit error leading to gossip protocol blowup. [11]	6-8 hours	July 20,2008
FlexiScale: core network failure [12]	18 hours	Oct 31, 2008

Figure 32. Outage of some cloud services [8]

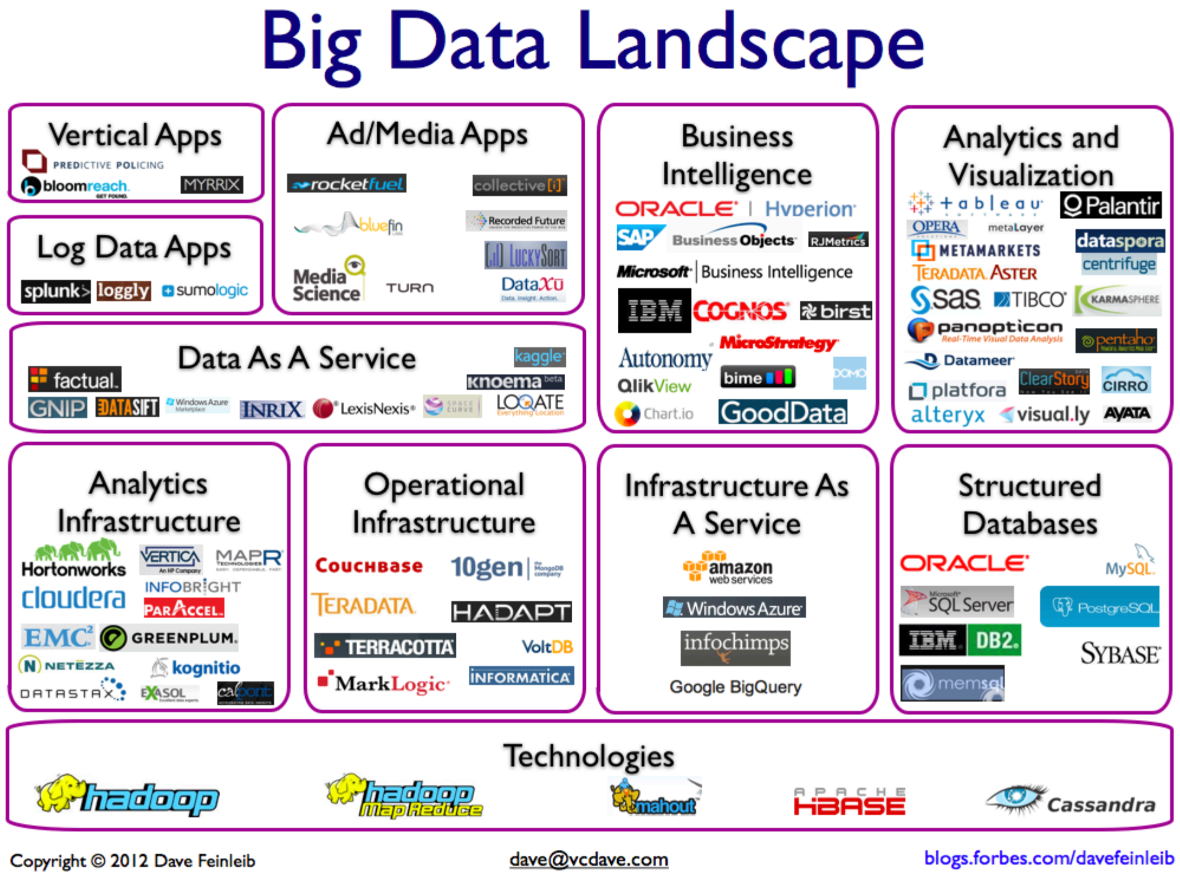
Molecular dynamic simulation has also been influenced by big data. HiMatch is a software developed for this purpose, [172]. This tool is inspired from the MR paper presented by Google. Authors have tried to bridge the gap between molecular dynamics simulations. Their tool enables users to write trajectory analyses program in a sequential fashion and the toolkit is in charge of deployment of the parallel application. This framework has been implemented in Python and support multi-core processing.

Scientific community is starting to bring problems to modern clouds. For example Openfoam has been ported to Amazon. Improvement of CloudFlu allows deploying OpenFoam on AWS services [210][211].

6.2. Involved companies

Big Data is also an industrial problem, see Figure 33, not only scientists have to deal with this domain. This section presents some industrial project, which involved with Big Data. [237] presents a set of product which include Hadoop as a part of their infrastructure.

IBM is a major company for Big Data. One of their famous systems is named *Watson*. This super computer is able to crawl data over the Internet, to extract knowledge. This super computer gathers information over Internet, and uses it to play *Jeopardy.* To evaluate capacities of this system, Watson has been opposed to two best players of this game. At the end of the party, Watson wins. Watson is not a storage system; it is an efficient crawler system. Internet is full of correct and incorrect information. Thus, Watson has to deal with this incorrectness. It has been necessary to filter information: slang language or abbreviations. Another issue for Watson was the massive use of Wikipedia. With this super computer, IBM has been able to beat two of the best Jeopardy players [218]. Now Watson has evolved and has been used to help nurses to diagnostic health problems.



Copyright © 2012 Dave Feinleib dave@vcdave.com blogs.forbes.com/davefeinleib

Figure 33. Example of companies, which involved in Bigdata.

EDF, a French company involved in power management, also deals with big data. EDF is currently deploying a new measurement tool. It is a new way to manage and monitor power usage by consumers. This solution grabs information every 10 minutes from 35 millions of homes. Raw data produced by this solution represent for each time step 120 TB. But to deal with this huge amount of data it has been necessary to use a specific data warehouse. The first solution was designed to run a Hadoop system, but they preferred finally to use the UDA solution (Unified Data Architecture) based on Hadoop, Teradata and Aster Data. To improve knowledge, they also grab information from social networks to evaluate the feeling of their users [206].

Facebook is also a well know company which deals with Big Data. To store their massive data, FB needs to use a cloud architecture. Their solution is based on Hadoop, with on top layers some SQL possibility. In the earlier days these possibilities were provided by Hive, but this solution is not efficient enough for their needs. Thus they have developed their own SQL engine on top of Hadoop. This architecture is used to store data from multiple users and also to apply analytics on data.

Yahoo! has also largely involved in the Big Data community. They have developed the Hadoop ecosystem, and a spin off (Hortonworks [216]) has been created which also involved in the Hadoop community. They provide some plugin, which can be used to extend Hadoop. They use this specific architecture to store data for web crawling, or some analyses of users information, genome, etc.

Google has published some keystone papers [56] and [215], which brings to big data their first key features. Their first solution is closed source, and only usable by Google developers. They use their technology to deploy some services: Google search engine, Gmail, Google drive etc. They also have opened their cloud infrastructure and now it is possible to deploy applications on top of the Google architecture. They also provide Google cloud tools (BigQuery). They also propose some cloud services: App engine, Cloud Database, Cloud SQL, Cloud Storage, etc.

Amazon is another leader in the Big Data industry. Through Amazon Web Services, they provide a set of facilities to deal with big data and clouds. Their product is dedicated to some specific use case: Intensive computation, storage, database, analysis monitoring and deployment, and software facilities. For example with the EC2 services, Amazon provides a cloud infrastructure based on virtual machines, and designed to run intensive computations. With their S3 solution, they provide an evolving cloud system to store data. They also provide facilities to use a Big Data architecture and HPC facilities at the same time.

Twitter is more a Big Data user than a service provider. They have developed some extensions to Hadoop, but mainly focused on their needs. They try to store and analyze large data extracted from network links between users. Their data is mainly designed to be graph-structured.

Aster is a company involved in BigData analytics. They have developed a tool based on SQL-MapReduce approach, which can be used to perform Big Data analyses. This methodology combines a standard Database and a MapReduce solution to deliver a high-speed analytics tool.

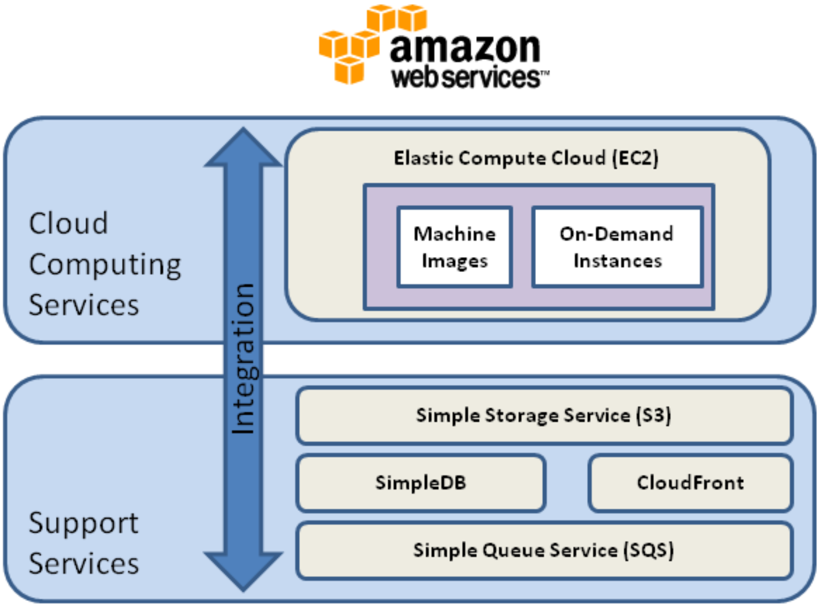


Figure 34. Amazon AWS.

JASMIN is a big data analytic system, it was developed to run complex computation over large data set. [78] presents the initial configuration of JASMIN, and the first experiment of this system. Requirement of Big Data analysis have not been achieved on JASMIN, this system does not yet own enough compute and storage capabilities. The JASMIN architecture is composed by six major components: a low latency network, a Panasas storage sub-system, a batch compute system, a data compute system, a high memory system and two images of virtual machines.

6.3. Which datasets

Datasets in Big Data systems involve in many domains. People are usually interested to store all the information, and understand them later. The processing of data is applied in a second step. In this section we present some use-cases, like data monitoring [167].

A first set concerns health data. This dataset is coming from many sources: Hospital measurement tools, quantified self-measurement tools and diagnostics provided by health personnel. All these information are extracted at different time step, and different frequencies. Their validity can also be noised by some external sources: human factor, hardware issues. Governments of different countries have decided to group national health information into a unique health DB. This aggregation of data brings security, and access issues. It is also necessary to design the most suitable solutions, which can deal with these different datasets: images, numerical values, and human diagnostics. Analytics and storage solutions need to be improved to fit with all these sources.

New buildings also produce more and more data. These entities have largely evolved in the past few years. In the earlier days, buildings are equipped by BMS (Building Management Systems), which provide all necessary tools to manipulate the whole ecosystem of the buildings: sensor, valves, actuators, etc. These systems are only reactive. To improve the usage of buildings, it is necessary to propose new kind of devices able to improve the building behavior through the use of massive dataset. These data sets are composed by values of the equipment. These new BMSs provide sensor values at different frequencies; these systems also provide massive datasets (more than 1.000 of sensors in a regular buildings). All the time steps of sensor data are stored to improve the building management.

The next generation of scientific discoveries will undoubtedly be provided by usage of Big Data. More and more equipment and simulations create huge datasets, which have to be effectively managed. Also, the next generations of discoveries need to be compatible with Big Data, Big Data analytics, Data-intensive simulations and computation. The science has to follow an important evolution: at the beginning this field was lead by empirical descriptions of natural phenomena. After some year theoretical models are used to describe real natural phenomena. At the beginning of 21e, computational model bring science to a new dimension. then models are proposed, then computational models are used to simulate complex phenomena and now science uses big data to extract and infer new theories, new evidence, new perspectives and better understanding of natural phenomena.

6.4. Products

As stated before, many company are today involved in Big Data, see Figure 33. Example of companies, which involved in Bigdata. In this section, we present some existing products, which are dedicated to Big Data and Clouds. This section is decomposed in two parts: closed-sourced software and open-source software.

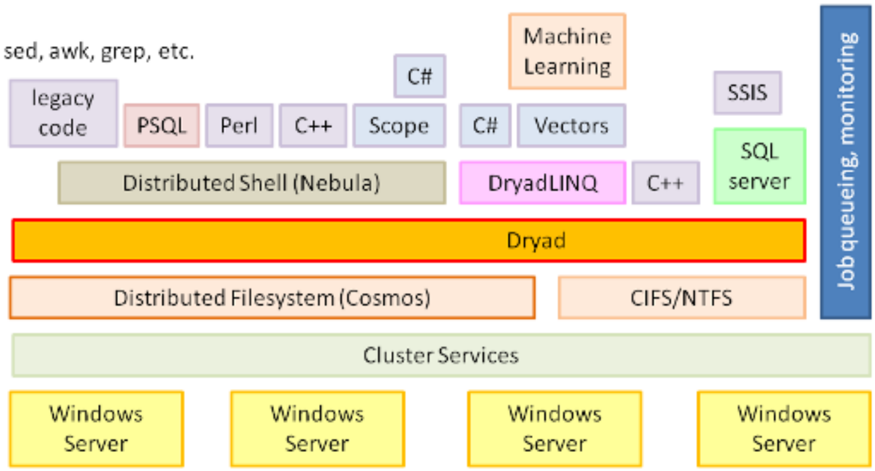


Figure 35. Dryad Stack [207]

Miscrosoft has been an important actor in the cloud area. They have developed a cloud platform named Azure. It is a PaaS cloud system, [150]. This service provides similar capabilities compared to the Amazon cloud system. And this system is designed to be suitable with Microsoft tools. As a proof of concept, they also have deployed some of their popular apps in the cloud environment: SQL Server or Office 365.

Microsoft also provides it is own MR system named Dryad. Their solution is based on the traditional distributed computational model (MapReduce), which can scale from a small to a large cluster. Dryad has been designed to help developer use all resources of a computer cluster or data center for running parallel programs. Usability of this solution has been provided by several tools, which enhance the traditional MR paradigm. They have developed SSIS, which can run multiple instances of a SQL server. Dryadlinq generates computation using a specific language: LINQ. A full description of this MR toolkit is presented in [143], and a presentation of the Dryad ecosystem is shown in Figure 35. As stated in [236], a new version of their solution is available, and now it is support Hadoop with YARN.

Google is also another important company, which deals with big Data and Clouds. As stated in [150], Google was born in the cloud. This company uses basic functionalities of cloud systems to provide the most efficient and suitable search engine. They have developed a set of tools which are not available to public, and are able to reach high performance. Now they have extended their framework, and provide new tools on top of it. They have used their own cloud solution to produce some of their bestseller tools: Google docs, Google drive, etc. Google now provides solution to deploy applications through their cloud architecture. Now, Google can offer three engines: Google App Engine, Google Cloud Storage and Google Big Query. They also provide demonstration of their capabilities, for example, they have extend the NoSQL DB named: Cassandra, and with this solution they are able to reach one million writes per second [145]. This solution has been deployed on Google Cloud Platform; the architecture of the solution is presented in Figure 36.

Intel (http://www.intel.com) was recently involved in Cloud systems. This company has produced some customized Linux distributions to run efficiently Hadoop. These distributions were splitted in several distributions regarding the needs of the IT system. Reference [156] presents one distribution of this ecosystem. These solutions provide efficient tools to decompose a data center into computation nodes and storages nodes. To be suitable with HPC, they extend Hadoop to use the Lustre FS. They distribution also integrates some of the most important tools for HPC usage. The architecture of this distribution is presented in Figure 37.

IBM (http://www.ibm.com) is a key partner in the development of OpenStack. OpenStack is an Operating system, which deals with deployment of virtual machines, like VMware or Citrix solution. IBM has planned to only use OpenStack for the deployment of their clouds [150].

HP (http://www.hp.com) also owns a cloud solution, see [166]. There solution is composed by a set of tool for storage, computing, DB, load balancing and so on.

VMware (http://www.vmware.com) is a software developer, which is mainly involved in virtualization software. They have developed a large set of tools to deploy and run virtual machines. They also have built software named vCloud, able to provide all the functionalities to deploy a full cloud system [150].

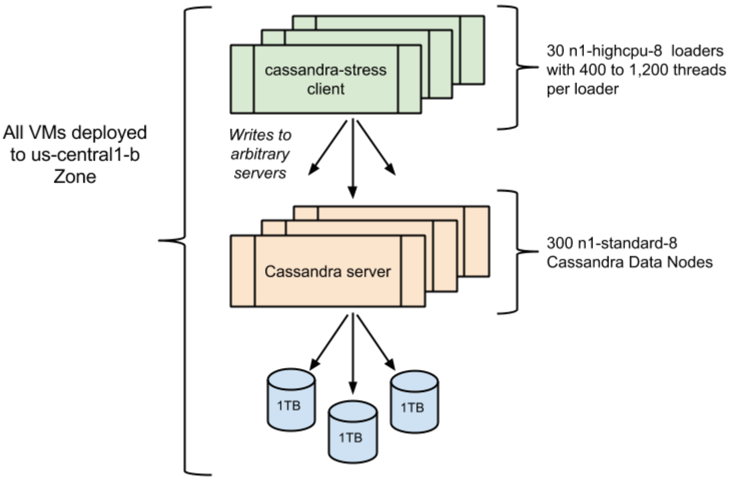

All VMs deployed to us-central1-b Zone
cassandra-stress client
30 n1-highcpu-8 loaders with 400 to 1,200 threads per loader
Writes to arbitrary servers
Cassandra server
300 n1-standard-8 Cassandra Data Nodes
1TB
1TB
1TB

Figure 36. Example of Cassandra deployment on the Google Cloud Platform.

Amazon (http://www.amazon.com) is the most well know company involved in cloud systems. They provide a set of tools through there platform AWS (Amazon Web Service). Tools provided by Amazon concern several domain of cloud computing. They propose a tool for storage, DB, analytics, SaaS monitoring. In [150], Amazon was described has the inventor of IaaS market. Amazon covers all the pipeline of cloud computing, they can deliver from storage facilities to super computer systems. Two services of Amazon are mainly used by their users: S3 and EC2. S3 is dedicated to provide a cloud storage solution, and EC2 is dedicated to computation [143].

All of these companies provide solutions to deploy cloud systems with associated fees. Open source software solutions have been developed to allow anybody to deploy a cloud system.

Apache is a software foundation, which host an important set of these tools. Hosted projects deal with Database, big data and clouds [208]. This foundation host a large part of open source tools from the Hadoop community.

VMware has developed a tool to deploy a whole Hadoop ecosystem: Serengeti. This solution allows running multiple Hadoop distributions on virtual servers. With Serengeti, it is possible to deploy a whole Hadoop ecosystem in a couple of minutes, one command is enough. All configurations parameters required for Hadoop clusters is produced by Serengeti.

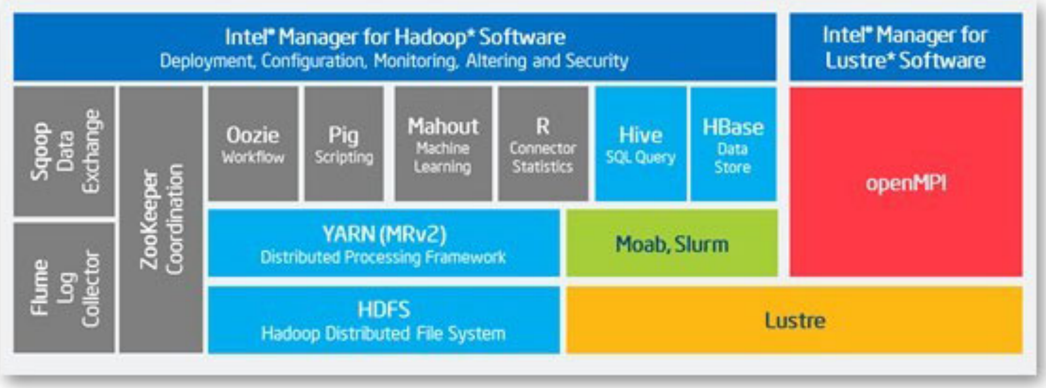

Intel* Manager for Hadoop* Software
Deployment, Configuration, Monitoring, Altering and Security
Intel* Manager for Lustre* Software
Sqoop Data Exchange
Flume Log Collector
ZooKeeper Coordination
Oozie Workflow
Pig Scripting
Mahout Machine Learning
R Connector Statistics
Hive SQL Query
HBase Data Store
openMPI
YARN (MRv2) Distributed Processing Framework
Moab, Slurm
HDFS Hadoop Distributed File System
Lustre

Figure 37. Architecture of the Intel distribution of Hadoop.

Instead of using a closed-source virtual manager system, it is possible to use an open-source solution. Sahara [196], previously named Savanna project, can be used to deploy a whole Hadoop ecosystem on top of an OpenStack platform. This solution provides all necessary tools to be suitable with this virtual machine manger.

Mesos is an Apache project, which provides resource isolation and distribution strategies to run distributed algorithms. With this software it is possible to deploy a whole ecosystem of Hadoop, Jenkins, Spark, etc. This tool provides a scalable system suitable for 10.000 nodes; virtualization is done using Linux containers.

The Apache Ambari is another project that aims to provide a simple tool to deploy Hadoop ecosystems. This software is a management tool to simplify development, provision management, and monitoring of Hadoop clusters.

Whirr is another project that can provide convenient ways to launch clusters. This solution can deploy a whole Hadoop system anywhere on a: private cloud or AWS. This solution provides a set of library for running cloud services. This solution provides a neutral way to run a cloud service with any provider.

OpenStack [128] is a major tool in the Big Data community. Openstack is a cloud OS, which controls large pools of compute, storage and network nodes [195]. This software provides all the necessary tools to monitor efficiently the deployment of a complete cloud system. It has been designed to provide cloud flexibility and supports most of common hardware and does not impose any software requirements. This solution is designed to manage and automate pools of compute resources. Different kinds of supervisors can be used to deploy cloud resources (KVM, XenServer and LXC). This stack also supports alternative hardware architectures like ARM. It can be used to process Big Data using Hadoop. This software is designed to provide virtual machines needed to compute results of BigData queries. Several plugins have been developed. Hortonworks provides one of these, which enable to run Hadoop on top of Openstack [128]. All the parameters settings and deployment of virtual machines are performed automatically with this tool.

7. The Hadoop Ecosystem

The last part of this section is a discussion of Hadoop 2.0, [121]. It is the most recent version of Hadoop. It brings modern optimization to the Hadoop ecosystem, some of them have been presented before: HDFS 2.0, Yarn, etc. With this novel ecosystem, Hadoop can achieve higher performance: uses more compute/storage nodes, a simplified deployment, etc. This solution does not have lost his extensibility, and most of existing extensions can be used. The ecosystem is still evolving and new extensions are proposed.

7.1.1. Introduction

Hadoop is a well-deployed system, which has been released by the open-source community. This system has been developed to reproduce the computational pipeline presented by Google, this computational model is named MapReduce. The Hadoop system has largely evolved and now new possibilities are available: it provides all necessary stuff to process (in any manner) distributed data.

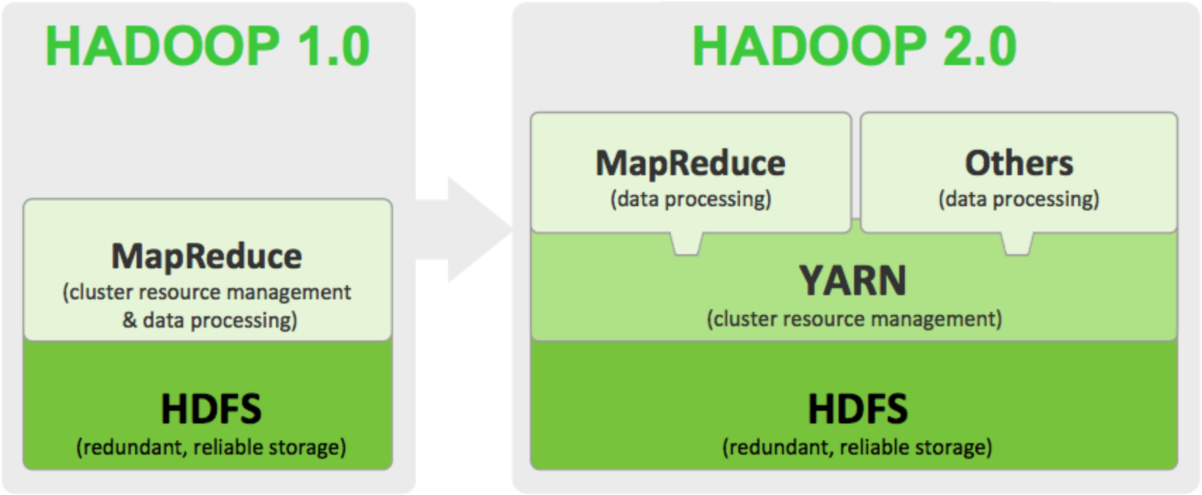

HADOOP 1.0
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(data processing)
Others
(data processing)
YARN
(cluster resource management)
HDFS
(redundant, reliable storage)

Figure 38. evolution of Hadoop [216].

7.1.2. Evolution of Hadoop 1.x to Hadoop 2.x

The first version of Hadoop is based on the traditional MapReduce strategy. This solution decomposes queries in two functions: Map and Reduce. The Map function will gather information from several nodes, write results on a file, and then the reduce function reads these file and produces the final result. This strategy is well suited for some use cases, but it is necessary to propose a modified version of this model to solve some complex analyses. Generally, this MR model is compared to a low level language, thus to be more usable, a high-level language is necessary. The traditional MR model is not anymore enough to express complex computation needed by Big Data system. Computation with this model can take more time than any other computation workload.

To simplify Hadoop, some extensions have been proposed. We will present some of them in the next section.

Recently, scientists have made a major revision of the Hadoop ecosystem. This major revision changes the Hadoop version from 1.x to 2.x. This new version is based on Yarn. The fundamental idea with YARN is to split two of the functions into separate job. This separation has been presented earlier in this document. With this solution any kind of computational model can be used with Hadoop. MapReduce has been ported to Yarn, as a proof of concept, and to provide backward compatibility. Now Hadoop 2 is following the same path than the previous version, and a large set of specific extensions are available.

The next section will present some of the extensions available for Hadoop; most of extensions developed with Hadoop 1 are also usable with Hadoop 2.

7.1.3. Hadoop extensions

The Hadoop community has provided an important set of extensions. The list of these extensions is available on many online resources [153] [158][160]. The classification of these extensions is not a simple task and different classes can be presented. The Figure 43, from [153], presents some extensions that have been developed for Hadoop 1 ecosystem. They concern all parts of the Hadoop framework. Each of these extensions brings improvements at different levels, for example the computational model, the storage file system, the possible analyses, etc.

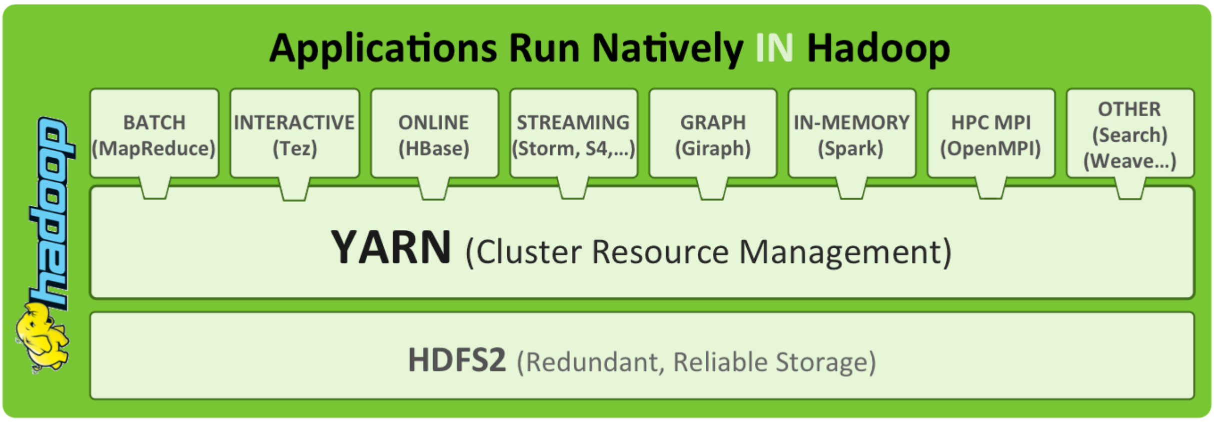

Applications Run Natively IN Hadoop
BATCH (MapReduce)
INTERACTIVE (Tez)
ONLINE (HBase)
STREAMING (Storm, S4,...)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave...)
YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)
hadoop

Figure 39. Example of applications, which run on Hadoop [216].

Some Hadoop distributions provide all tools to deploy a complete system. Two leaders of Big Data propose this service: Cloudera and Hortonworks. Cloudera distribution provides 25 tools related to Hadoop (with Avro, Crunch, DataFu, Flume, HBase, Hue, Hive, etc.). And 12 tools only compose the Hortonworks distribution.

In the next sections, we present some of these extensions. We will present some compute models; analyze tools, storage solution and management tools.

7.1.4. Hadoop Computation

In this section, we present some computational improvements of Hadoop and also deployment facilities for this model: Hadoop has not been designed to run efficiently on HPC system.

The MapReduce paradigm has brought simplicity to distributed computations. Unfortunately, this strategy is too simple to express specific workloads. But most of computational problems are not batch suitable. Thus, researchers have developed extensions on top of Hadoop to simplify application development.

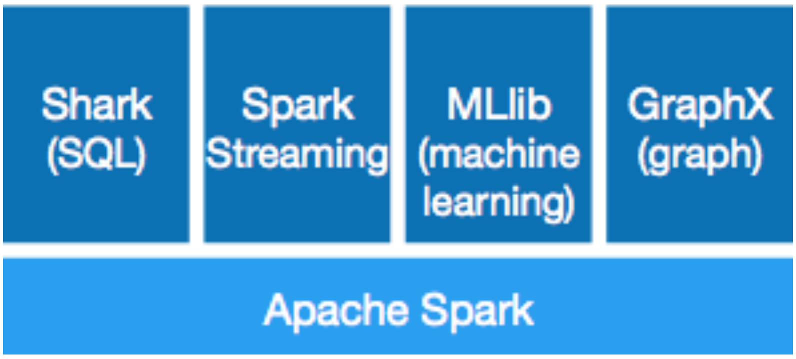


Figure 40. Spark software stack [108]

Spark is presented in [108][109]; it is a recent Apache project. In the earlier days of this project, it was developed under the incubation platform of Apache. Spark can run on top of Hadoop 2.0, using Yarn manager. With Yarn, the previous applications run faster than the first version of Hadoop framework. This extension can provide an efficient tool for analytics based on SQL, a streaming engine, a Machine learning library and a graph library. Figure 40 presents a schema of the stack layer for Spark. This software is evolving and releases are provided regularly. On top of Spark, they have implemented a query processing engine based on SQL. This engine is called Shark. In [162], authors provide a description of this query engine. Shark provides a streamlined interface, which enable deep data analysis. A schema of this stack layer is presented in Figure 41.

For graph datasets, Giraph has been developed, [131]. Giraph is an extension, which enables to process large graphs in an efficient way. This solution is an open-source implementation of Pregel. Giraph has been improved compared to the traditional Pregel model. This solution brought a master computation node, a share aggregator; an edge oriented input and also out-of-core computation capabilities.

Another example of computational model for Hadoop is named Reef. Reef has been presented in [161]. This model is used to compute Singular Value Decomposition of a dataset. This algorithm is scalable and can run on structured or unstructured data sets. To validate their solution, authors benchmark this computational model on a real Twitter dataset.

To reduce complexity of YARN, some other software is developed. Twill, presented in [114], brings a new abstraction layer for YARN. If we consider Yarn as an operating system for clusters, Twill brings ease of development for customers: they only deal with business logic. Twill allows expressing Yarn jobs similar to thread programming.

Spring for Apache Hadoop provides another simple way to deploy a Hadoop Yarn system. It produces an unified configuration model for any part of the Hadoop ecosystem (HDFS, MR, Pig or Hive). This solution is presented in [115].

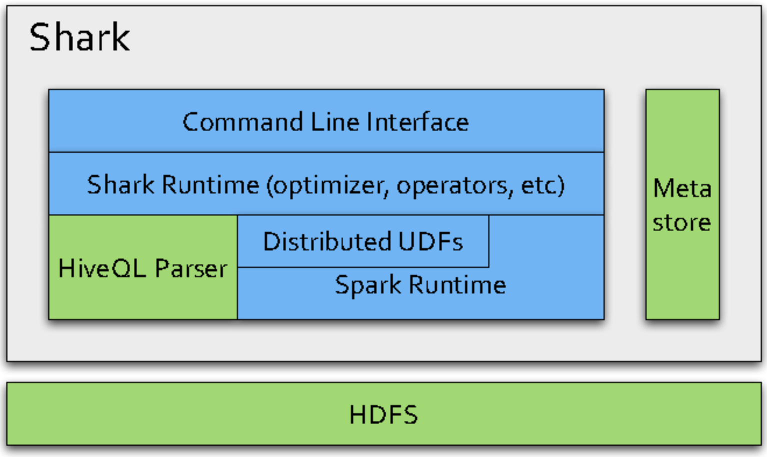


Figure 41. Shark architecture

Kafka [127] is a publish-subscribe messaging solution. It is provided a distributed system to transfer logs. This solution has been designed to handle hundreds of megabytes of reads and writes per second. A message passing system is a low-level communication tool. This solution brings specifics problems, which needs to solve on processing layer. In [129], authors present a solution to avoid issue of message passing system. This solution is a top layer of message system called stream processing. Stream used in Samza is a more complex system compared to the traditional message passing system. This stream engine provides buffers requirements and isolation of processing from each component (application and system).

The Hadoop ecosystem can be deployed on HPC facilities with specific tools. In [175], the authors present an evaluation of Hadoop in the context of HPC. They have evaluated the impact of the file system, network and programming model for scientific workloads. This paper describes the usability of MR for three data intensive operations: filter, merge and reorder. Filter operations return a subset of the entire data set. Reorder operation is designed to reorder data in an output dataset (identical size of the input data set). Merge brings two datasets into one data set. This paper has studied different workloads: palindrome processing, data processing, Wikipedia data processing and Terragen. This study shows that HDFS is well suited with minimal writes.

To deploy a Hadoop framework on a HPC two solutions have been presented. The first one is named Hadoop on demand and the second one myHadoop.

Hadoop on demand is described in [179]. This tool has been designed to make deployment of Hadoop easier. It provides a tool to deploy virtual Hadoop over large clusters. This solution can allocate nodes, and start all necessary daemons (Hadoop, MR and HDFS). It also provides the appropriate configuration file.

MyHadoop presented in [34], [100] is another tool to deploy Hadoop on a HPC. This tool is an aggregation of script, which starts Hadoop services over a HPC. This method dumps data from HPC nodes into the Hadoop, and then applies computation with Hadoop framework. Then at the end of computation, all HDFS partitions are scratched. And HPC nodes can be reused as HPC nodes.

7.1.5. Hadoop Analysis

Hadoop MR is not the most efficient tool to analyze complex data. But some extensions have been developed to perform these computations.

Bioinformatics is a domain, which needs a lot of computation. Analysis of genomic sequences is a complex task. Crossbow presented in [180] is a scalable software pipeline for genome alignment. This framework allows deploying a whole computing system in the Hadoop environment. This solution can bring faster results, compared to the traditional single node approach.

Another Bioinformatics tool is named SeqPig, [181]. This tool is built on top of Pig which relies on a Hadoop system. This extension is a distributed analysis tools for large sequencing datasets. This solution takes advantage of the Hadoop framework using the high-level language Pig. The authors have created a library and a collection of tools to manipulate, analyze and query sequencing datasets in a scalable and simpler manner. This set of tool is based on the scripting engine Pig.

CloudBusrt is another analytics tool for bioinformatics. This tool has been released in 2010, and presented in [182]. It is an optimized algorithm for next-generation of sequence mapping for the human genome. This solution uses the Hadoop implementation of the MapReduce to parallelize executions of the algorithm. This solution is able to achieve linear speedup regarding to the number of read mapped.

MRQL is a query-processing engine optimized for large-scale, distributed systems. This extension is built on top of Hadoop, Spark and Hama. This query engine is based on SQL-like query language for large-scale data analysis. This extension can process data through three models: a map-reduce job, a BSP mode using Hama and a Spark mode. This extension is presented in [194]. With MRQL, users can express complex analysis like: PageRank, K-means, matrix factorization, etc.

In [209], a comparison between the most important SQL engines for Hadoop is presented.

7.1.6. Hadoop Storage

Storage of Hadoop frameworks has also been widely studied. Scientists have proposed a lot of extension to use other file systems instead of the traditional HDFS. To store files, two strategies are mainly used: a standard File System [240] or a Data Base. These extensions of FS are for example: Lustre[239], Ceph, Gluster FS and Tachyon. All of these systems have some specific advantages, and it is possible to use them for Hadoop deployments. For example, Tachyon is an open-source, distributed, fault-tolerant system that stores data in memory. The second strategy uses Databases to store the data, and extensions allow to use Cassandra or mongo DB as basis of the File System. This section presents two approaches that deal with such storage strategies.

In [178], authors have proposed a strategy to use a Hadoop ecosystem with a HPC infrastructure. They have study performance of both distributed FS: CloudStore (Hadoop FS) and a specific HPC FS named: Parallel Virtual File System. To perform their evaluation, they have used two different workloads: one designed for the HPC FS and the other one suitable for the Hadoop FS. For this benchmark, they have used a record trace mechanism to provide I/O traces of both workloads. Their workload was named TFIDF (Term Frequency-Inverse Document Frequency). They have shown that the Hadoop FS runs slower than the HPC FS. They have also discovered that the Hadoop job Scheduler made different placement decisions for mixed workloads. Thus it is necessary to provide a wider study for why Hadoop job scheduler tries to use data locality in some cases and not for others.

To improve the accesses of storage solutions, it is necessary to find the most efficient way to select the data. In [193], authors have developed a new solution: Hadoop++. This solution can boost access to files without any modification of the Hadoop framework. The strategy used by the authors injects specific information at the right place. No changes are performed on the Hadoop ecosystem. This paper shows the efficiency of this method compared to the traditional Hadoop methodology. Hadoop++ is composed by a new indexing strategy based on RDBMS indexing. This solution adds an additional header to the files.

Before a storage phase, it is necessary to bring data to clouds. Different solutions exist for this purpose. Two tools are mainly used: Flume and Sqoop. Both tools gather information from specific storage systems. Flume has been developed to collect, aggregate and move large data into HPC system. Data is streamed to the cloud systems [135]. On the other hand, Sqoop is a bulk synchronization tool, which allows transferring data from a structured repository (a RDBMS) to a Hadoop server [136].

Hadapt is commercial software, which cover the needs of SQL storage on Hadoop. This tool is a hybrid storage system that uses a SQL DB and HDFS to store the data [124]. It also brings a native SQL engine on top of the Hadoop system. This tool comes with an ODBC connector to simplify access to the data.

The TeraData company, [223], provides a storage distribution for Hadoop which uses their database to store the information. This solution is presented in Figure 42, and shows the architecture of their solution. This strategy is based on a top-level plugin. With this methodology, they can use both products to deal with Big Data: Aster a discovery tool and Teradata a storage facility.

In [238], Quirks is presented. Author explains how he implements an abstract file system layer for Hadoop. He uses Quirks as a default file system for Hadoop. This document provides most of necessary information to extend Hadoop with a complementary File System.

7.1.7. Hadoop Management

This last section is dedicated to the management tools for Hadoop systems. Some extensions proposed by the community are designed to extend the Hadoop management strategy. For this purpose, we will present three kinds of extensions, some of them are dedicated to the global management of the ecosystem, other are used to improve and simplify its deployment, and the last strategy are used to monitor jobs.

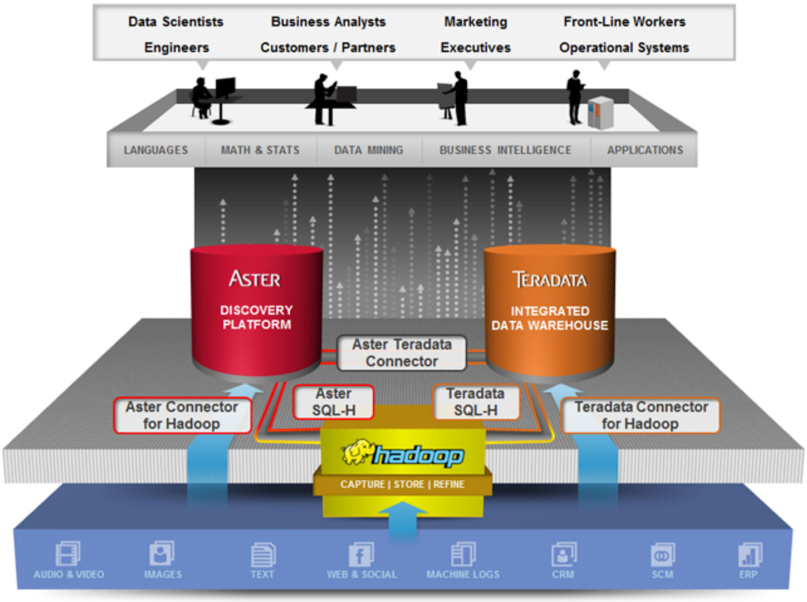


Figure 42. Teradata, data integration on Hadoop [219]

The Bigfoot project is a FP7 project, which involved with 9 partners. The goal is to design, implement and evaluate a scalable way to process and interact with big data application footprints. In [149], they present their contribution to the Hadoop community. They have provided some modifications to the global Hadoop ecosystem using an analysis tool, a protocol scheduler, a k-mean cluster solution for Spark, an improved simulator, an Openstack measurement framework and an artificial workload injector for Hadoop. All of these improvements are developed to provide a better analysis of the Hadoop behavior.

To reduce the complexity of the deployment HPC systems, it is necessary to provide a standard communication protocol. One of the most standard communication protocols between nodes is based on MPI. In the thread [183], the authors expected to bring support of MPI in Hadoop in a couple of releases. This strategy has already a name: Hamster. This integration will be possible due to the recent separation of the JobTracker introduced by Yarn.

To improve the deployment of an application on top of Hadoop, some scientists have also brought new tools. An example is Hoya, [125]. Hoya allows to deploy an existing application on YARN. This tool provides all necessary tools to monitor these jobs, and it is also possible to extend dynamically the number of nodes used by the cluster. This tool has similar requirements to Hadoop ecosystems and can be extend with existing plug-ins. For example it is possible to use this tool with Hbase and Accumulo.

The last discussion of this section presents monitoring tools. To understand Hadoop behavior, it is necessary to understand workflow of this ecosystem. Thus, it is necessary to use Monitoring tools. They need to be efficient to provide the full capability for the management of Hadoop.

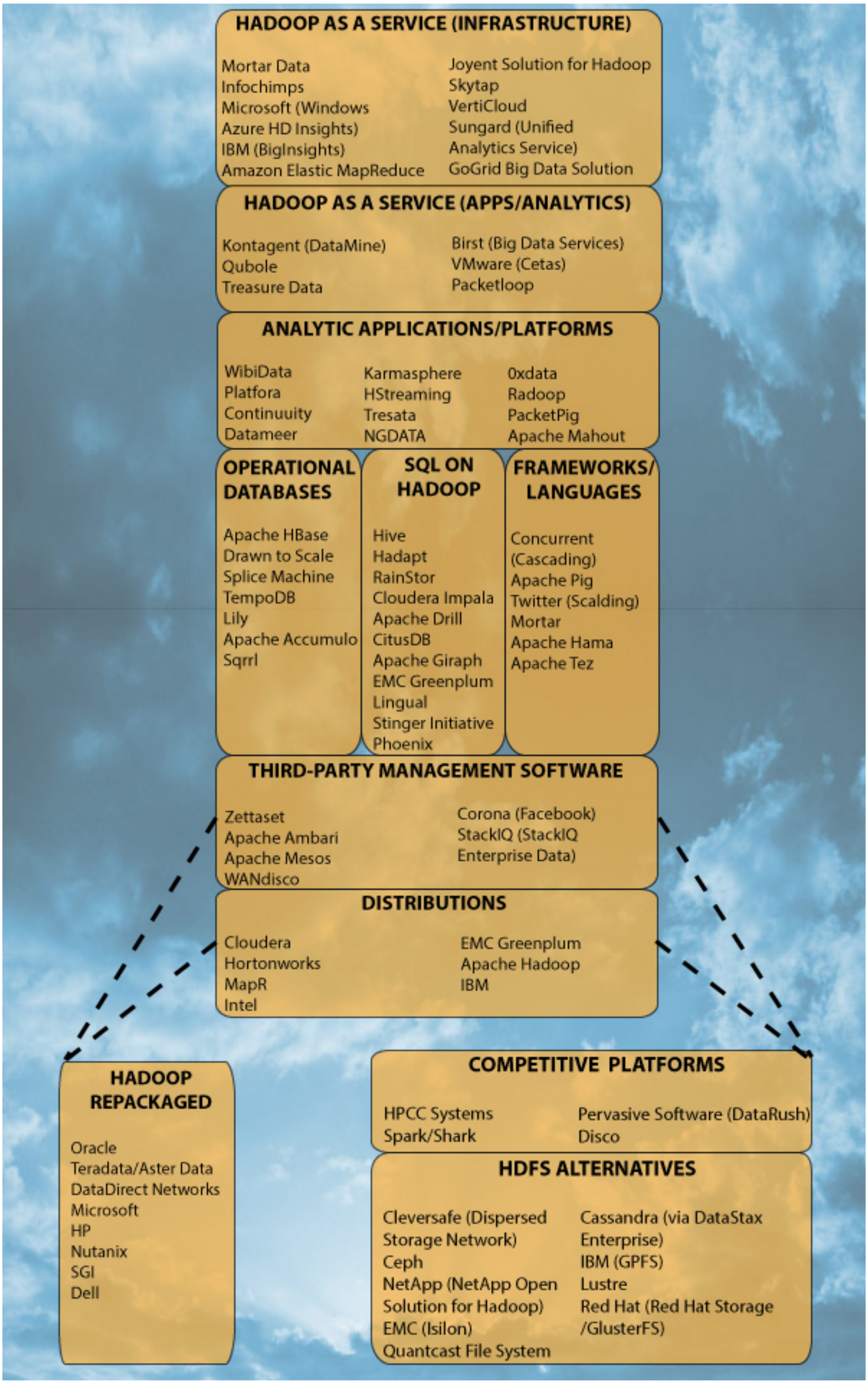
Ambari is a project, which aims to deploy and monitor a Hadoop platforms; this solution is presented in [133]. This software provides a simple way to manipulate or monitor Hadoop clusters. This solution is based on a web UI, and it can support multiple existing components of Hadoop: MR, Hive, Hcatalog, Hbase, Zookeeper, Oozie, Pig and Sqoop. This tool can be used for starting, stopping or reconfiguring Hadoop services. This system can be used to monitor the health of Hadoop clusters.

Another GUI is provided by Hue, [132]. Hue is a web interface for analyzing data provided by Hadoop, [132]. This GUI is a web application, which enables to manage files, jobs, and workflows in Hadoop. This GUI allows querying the Hadoop system through the SQL engine. This tool also provides a workflow monitor based on Zookeeper tasks.

Oozie, [134], is a workflow scheduler for the Hadoop ecosystem. This tool can express workloads of applications as Direct Acyclic Graphs. OOzie is able to deal with recurrent tasks to reduce execution time. This software can be used to express Hadoop jobs through any extensions: MR, Streaming, Pig, Hive or Sqoop.

Another well know tool is named Zookeeper. This tool is an old tool, which was brought by Yahoo! in 2008, [137]. This tool has been designed to transform a Hadoop ecosystem into a highly reliable distributed coordination system. It was used to save all states of any part of the Hadoop system, saving these state enables to restore them in case of failure.

Some tools are designed to automate the tuning of Hadoop. In [165], the authors provide an optimized technique to tune parameters of the Hadoop distribution. It has developed a tool named: Starfish, which deals with new extensions of Hadoop: for example a new query language, an iterative execution engine and a new storage system with random key access. This solution has been designed to reduce the overhead of the profiling tool.



HADOOP AS A SERVICE (INFRASTRUCTURE)

Mortar Data
Infochimps
Microsoft (Windows Azure HD Insights)
IBM (BigInsights)
Amazon Elastic MapReduce
Joyent Solution for Hadoop
Skytap
VertiCloud
Sungard (Unified Analytics Service)
GoGrid Big Data Solution

HADOOP AS A SERVICE (APPS/ANALYTICS)

Kontagent (DataMine)
Qubole
Treasure Data
Birst (Big Data Services)
VMware (Cetas)
Packetloop

ANALYTIC APPLICATIONS/PLATFORMS

WibiData
Platfora
Continuuity
Datameer
Karmasphere
HStreaming
Tresata
NGDATA
0xdata
Radoop
PacketPig
Apache Mahout

OPERATIONAL DATABASES

Apache HBase
Drawn to Scale
Splice Machine
TempoDB
Lily
Apache Accumulo
Sqrrl

SQL ON HADOOP

Hive
Hadapt
RainStor
Cloudera Impala
Apache Drill
CitusDB
Apache Giraph
EMC Greenplum
Lingual
Stinger Initiative
Phoenix

FRAMEWORKS/ LANGUAGES

Concurrent (Cascading)
Apache Pig
Twitter (Scalding)
Mortar
Apache Hama
Apache Tez

THIRD-PARTY MANAGEMENT SOFTWARE

Zettaset
Apache Ambari
Apache Mesos
WANdisco
Corona (Facebook)
StackIQ (StackIQ Enterprise Data)

DISTRIBUTIONS

Cloudera
Hortonworks
MapR
Intel
EMC Greenplum
Apache Hadoop
IBM

HADOOP REPACKAGED

Oracle
Teradata/Aster Data
DataDirect Networks
Microsoft
HP
Nutanix
SGI
Dell

COMPETITIVE PLATFORMS

HPCC Systems
Spark/Shark
Pervasive Software (DataRush)
Disco

HDFS ALTERNATIVES

Cleversafe (Dispersed Storage Network)
Ceph
NetApp (NetApp Open Solution for Hadoop)
EMC (Isilon)
Quantcast File System
Cassandra (via DataStax Enterprise)
IBM (GPFS)
Lustre
Red Hat (Red Hat Storage /GlusterFS)

Figure 43. [153] a graphical representation of Hadoop ecosystem.

7.2. Benchmarks

As stated in this document, IT systems for Big Data can be deployed on three different clouds: private, public or hybrid. To choose the right architecture, it is necessary to identify the needs and requirements of a project. Used tools, architectures and infrastructures depend on which goals are expected. This choice is also determined by an evaluation of performance and workloads. To evaluate them, some benchmark tools have been designed.

To deploy an efficient cloud system, the first step is to identify which version of a toolkit can be used. Regarding different distributions, the evolution of the implementation can bring new performance or overheads. For example in [154], the authors present advantages and drawbacks of Hadoop 2.0 and Hadoop 1.0. The main advantage of Hadoop 2.0 is the scalability of the system; it is now possible to deal with more than 1.000 nodes. To choose the right version, the right set of tools and the right solution for the deployment, benchmarks have also been proposed to cloud users.

Intel has produced two benchmark tools presented in [157]. The first one is named Hibench, it is a Hadoop benchmark suite used to evaluate parameters of the Hadoop platform through an evaluation stage. This strategy can be used to tune, and optimize the whole system. This tool is composed by a set of micro-benchmarks: Sort, Word Count, Terrasort. This tool is also designed to study the behavior of an enhanced file system (a high performing FS which can perform numerous read and write simultaneously).

Some other benchmarks are available regarding which extensions are used: for example Hive query benchmark or Mahout Bayesian classification. They have also developed another benchmark tool named HiTune. It is another performance analyzer for Hadoop. This tool is composed by three major components: Tracker (collect runtime information), Aggregation engine (merge results of the tracker and an analysis engine). This strategy allows analyzing workload of the cloud facilities by grabbing system information. In [61], authors perform some benchmarks based on standard tools: TerraSort, GraySort or MinutSort. They have evaluated sorting performance of Hadoop 0.23.

To find the best solution, it is also necessary to study the workload. In [151], authors bring a new study of specific workloads. This document is an empirical analysis of six-business contexts from real datasets. These datasets are extracted from Cloudera or FB. From these datasets, authors have described a new MR workload driven by an interactive analysis. Their analyses are based on three conceptual components: data, temporal and compute patterns. This study raises important question to identify workload behavior, for example: data access, data locality regarding to temporal query, regularity of query, common jobs, etc.

Benchmarks are used to study behavior of Hadoop systems; this methodology is presented in [152]. In this paper, authors present a standard benchmark suite to quantify and compare performance of different MR frameworks. These benchmarks are based on representative use cases of MR workloads. With this tool, authors are able to compare some applications like: Hadoop, Twister or Lemo MR.

Simgrid is a generic framework to enable simulation of large distributed systems. This solution is presented in [185][186]. This tool is built to study how big systems work (Grid, Cloud, HPC, etc.).This software can be used to evaluate some heuristics, it also allows to prototype applications.

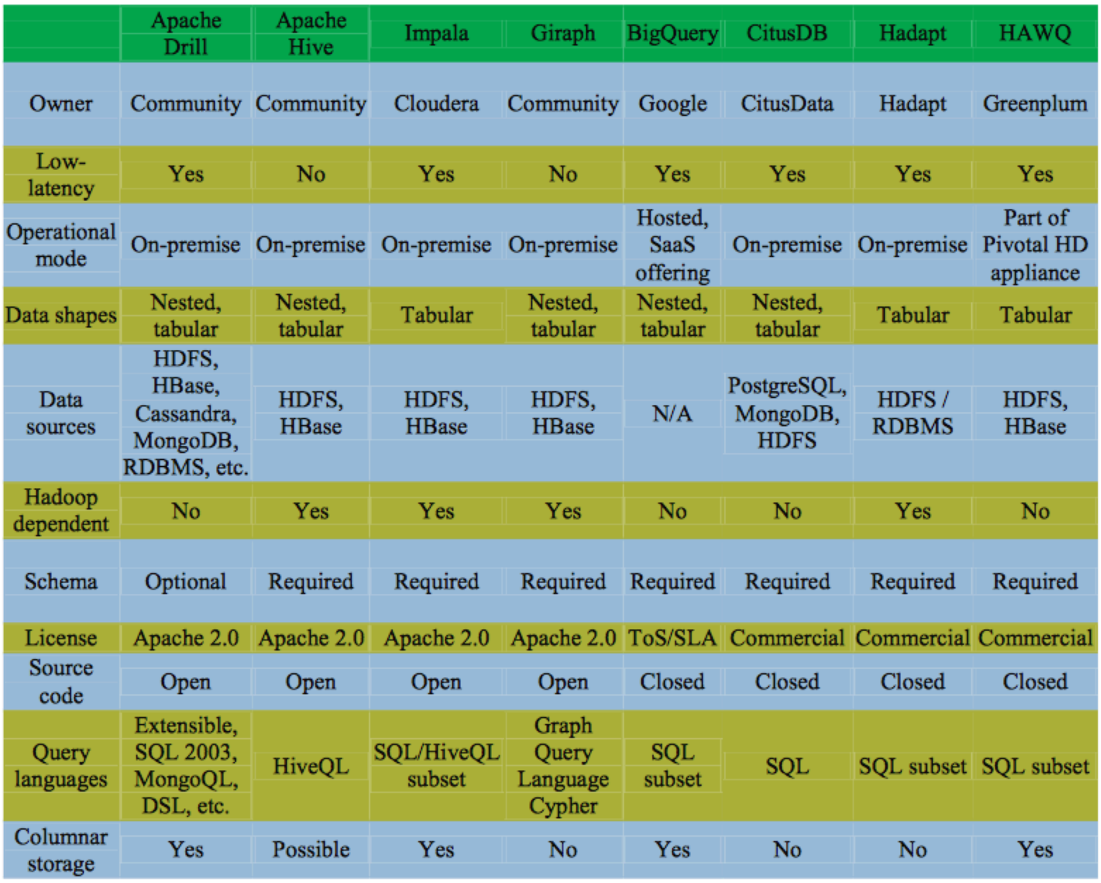
Applications can be simulated through dedicated tools. These simulations can be focused on different tasks: network, workload, etc. These simulation tools are more frequently used to identify the needs of an application. One of these simulation tools is presented in [184]. CloudSim has been designed to simulate and model cloud system infrastructures. This solution has been developed to help future cloud users to control the execution workflow of an application. With this tool, customers have the opportunity to tune performance of an application before a real deployment. The goal of this tool was to provide a generic and extensible framework to compute simulations.

Simulation of the Hadoop behavior can be evaluated with existing workloads. For example, logs are loaded to a simulation engine, which emulates the behavior of a specific computation on a virtual Hadoop system. In [148], authors present a modification of the Mumak simulator. This simulator is designed to run on top of the Hadoop framework. To run a simulation, it is necessary to provide a real workload trace. This tool only simulates MR tasks. Authors of this paper have extended it with a MRPref extension. This tool is a network simulator, which can be used to simulate the runtime of tasks and the delays of a network. Thus, authors are able to add these characteristics to Mumak, with this strategy they can predict the runtime of an execution with various parameters.

Another benchmark is named CloudCmp and was presented in [84]. This method has been developed to measure some parameters of cloud systems: elasticity, storage persistence and network capabilities. This evaluation is based on a set of metrics, to allow users to select the best provider: the one which can deliver the highest performance. Another target of this solution aims to help cloud providers with a standard metrics to compare cloud systems (identify where competitors are better, for example). Obviously, a complete coverage of cloud services features is not possible for a benchmark (all services are not available in all cloud solutions). But authors have provided fair metrics. These measures can perform metrics each hour to determine the most accurate model. Before starting an evaluation of a cloud system, a user needs to select a couple of those metrics compatible with the targeted cloud. This kind of benchmark can be used to propose a meta-cloud system able to combine the advantages of difference services.

For cloud systems based on NoSQL DB, a benchmark has been proposed in [69]. This solution has been developed to identify the best raw performance engine among NoSQL systems. Authors have developed some standardized tools to evaluate some criteria of such systems. This benchmark has been designed to perform an evaluation for key-value systems. To experiment this benchmark, several DBMS have been selected: Cassandra, Couchbase, Aerospike and MongoDB. This evaluation has been applied to a real-world hardware: without virtualized system, data can be stored in RAM or SSD. The results of this benchmark show that Cassandra and Aerospike are in a near-tie in term of performance. Aerospike outperformed all solutions for a high volume workload.

Finally, a website compare existing distributed DB systems, in [119]. This solution compares: Redshift, Hive, Shark, Impala and Tez. In this document, authors compare the most recent analytics frameworks. This tool evaluates different queries on these frameworks. This benchmark is fully deployed on a cluster, to evaluate which solutions are the most suitable with the infrastructures. Another comparison is provided at [170]. Here, authors have produced a table, which compares a large set of Big Data tools. This table is presented in Figure 44.



	Apache Drill	Apache Hive	Impala	Giraph	BigQuery	CitusDB	Hadapt	HAWQ
Owner	Community	Community	Cloudera	Community	Google	CitusData	Hadapt	Greenplum
Low-latency	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Operational mode	On-premise	On-premise	On-premise	On-premise	Hosted, SaaS offering	On-premise	On-premise	Part of Pivotal HD appliance
Data shapes	Nested, tabular	Nested, tabular	Tabular	Nested, tabular	Nested, tabular	Nested, tabular	Tabular	Tabular
Data sources	HDFS, HBase, Cassandra, MongoDB, RDBMS, etc.	HDFS, HBase	HDFS, HBase	HDFS, HBase	N/A	PostgreSQL, MongoDB, HDFS	HDFS / RDBMS	HDFS, HBase
Hadoop dependent	No	Yes	Yes	Yes	No	No	Yes	No
Schema	Optional	Required	Required	Required	Required	Required	Required	Required
License	Apache 2.0	Apache 2.0	Apache 2.0	Apache 2.0	ToS/SLA	Commercial	Commercial	Commercial
Source code	Open	Open	Open	Open	Closed	Closed	Closed	Closed
Query languages	Extensible, SQL 2003, MongoQL, DSL, etc.	HiveQL	SQL/HiveQL subset	Graph Query Language Cypher	SQL subset	SQL	SQL subset	SQL subset
Columnar storage	Yes	Possible	Yes	No	Yes	No	No	Yes

Figure 44. Comparison chart for Big Data analytical Tools [164][170][171]

8. Discussion for the VELaSSCo Project

This last section summarizes this document. This discussion tries to find which solution will be the most suitable strategy for the VELaSSCo project.

In this project, our goal is to define requirements of a storage platform for data produced by simulation engines. These simulations are based on solvers, which solve specific equations. These equations describe some physical phenomena. The computational model is based on FEM and DEM.

FEM (Finite Element Method) is a computational approach, which uses solvers to compute solutions to partial differential equations on a domain defined by a mesh and a surrounding domain. For example for a car simulation, the domain can be the air surrounding the car.

The second model is composed by DEM simulations (Discrete Element Method). In this approach, the discrete elements are defined as particles. The particles interact with their neighbors through contact forces. The data of DEM simulations is very dynamic because particles and contacts can be created or lost during the simulation.

These simulations have already reached their limits. Computation and storage of these simulations are too important. It is now necessary to compute simulation on multiple nodes to reduce the computation time or storage space. The second main issue for scientists is the storage of produced data. Currently, data produces by simulation are analyzed and scratched. Storing all available information will bring simulations to a new dimension. Moreover, actual personal computers are not anymore suitable for massive simulations: runtime take too much time, and these systems do not own enough memory. Thus, it is necessary to use more powerful systems, like HPC. The amount of data is also an issue; data needs to be split among nodes of the HPC. Each node computes its data and at the end of each time-step, results files are aggregated, and sent to the visualization tools.

This platform has also to deal with query. Two different complexities of queries have been identified: simple and complex one. Simple queries will extract direct content from database, while complex queries need a computation phase. These queries will be performed at two levels: batch and real time.

A big mistake with Big Data is to try to provide a solution, which enables simultaneously HPC and Big Data (computation and storage are located on the same cluster). Requirements of both systems are not similar. Thus for optimal solution, the preconized approach is based on a dedicate VELaSSCo storage platform. But to be suitable with any existing architecture, our platform will be deployable on any kind of IT system. With this scalable methodology, we will able to deploy a adapted platform, which deal with any hardware capabilities. This solution enables to work with the best and the worst hardware architecture.

For the best architecture, our proposed Big Data architecture can follow the most common strategy: use commodity nodes, with virtualization. With this method, VELaSSCo platform deploys the wider architecture, at the lower price.

Development of this platform can be suitable with two strategies: use an existing framework or develop a new framework from scratch. The first solution seems to be the more suitable one; development cost and complexity will be drastically reduced. Moreover, with an existing framework, it is possible to use existing plugins. The chosen architecture must be modular (plug-in concepts). The chosen framework has to fit the proposed architecture: be deployed on an important set of nodes, and support for virtual machines. The solution has also to be suitable with HPC facilities. For an existing solution, it is necessary to choose an evolving framework with regular updates. Using a deprecated solution can bring many design and implementation issues, increase the amount of work and result in poor performance.

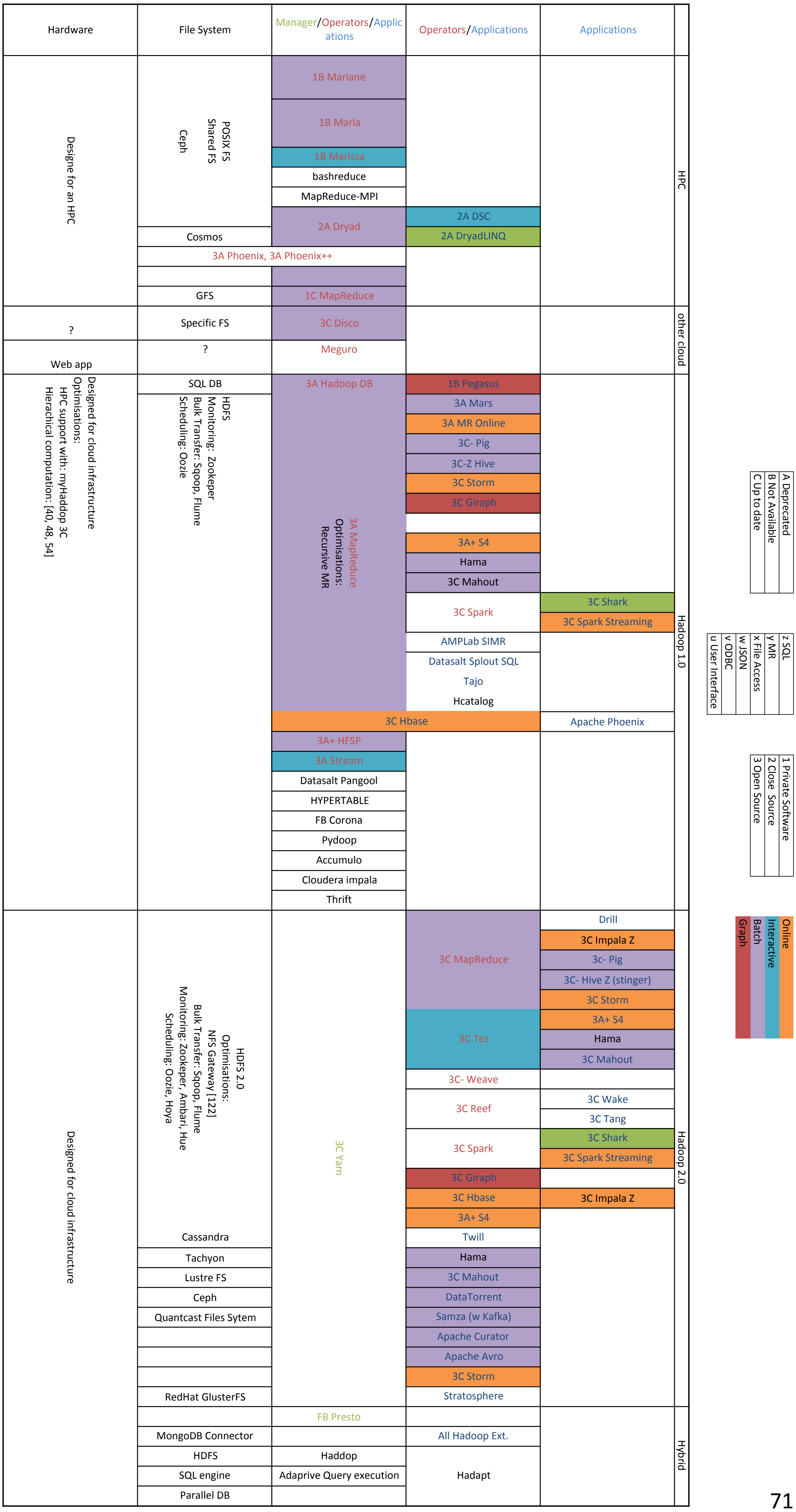
The Cost of the platform has also to be discussed; we plan to provide two different platforms, one designed around the EDM software, and the second one developed around a full open-source software stack. Regarding [234], Linux seems to be the most suitable operating system, where lower execution times are achieved. With this strategy, the global cost of our platform will be drastically reduced. It can be deployed on a larger cluster (cost will be mainly dedicated to hardware). The two possible platforms differ on some of the modules, but the skeleton of the architecture will be the same (specification of the architecture will be presented in deliverable D2.2). The main idea is to take advantage of commercial modules (like EDM) for the project, using partners' contributions, and let the door open to set up another configuration using open source software.

From these elements, the most suitable solution is the Hadoop ecosystem with some extensions; this framework has already been validated for astrophysical data [241]. As a first step, we can use a full distribution of Hadoop, which includes a set of pre-installed extensions, likes Cloudera [217] or Hortonworks [216]. Then extends this ecosystem, and disable unnecessary extensions. This minimal distribution will be used to deploy over extensions, and develop new one, some examples can be: storage for DEM and FEM data, support for storage using the Step format, study energy impact, and improve data locality.

As stated in previous deliverable, VELaSSCo project needs to deal with security. Now, new version of Hadoop can provide all necessary stuff to support security at different level. Security is provided at the file system level.

If we choose to use this framework, it will be necessary to answer some open questions:

- How to integrate EDM database on a Hadoop cluster?
- How to deal with interactive query needed by visualization tool?
- Hierarchical decomposition of simulation can be used to store data, is it an efficient strategy to use this approach to store the data?
- Is it necessary to use a specific data locality model to improve the whole system?
- How does the actual communication protocol deals with interactive visualization?
- The frameworks has to deal with multi-resolution of data, it has to grab data at a coarse grain and refines it during the query runtime.
- Etc.



Hardware	File System	Manager/Operators/Applications	Operators/Applications	Applications	
Designe for an HPC	POSIX FS Shared FS Ceph	1B Mariane			HPC
		1B Marla			
		1B Marissa			
		bashreduce			
		MapReduce-MPI			
		2A Dryad	2A DSC		
	Cosmos		2A DryadLINQ		
	3A Phoenix, 3A Phoenix++				
	GFS	1C MapReduce			
?	Specific FS	3C Disco			other cloud
Web app	?	Meguro			
Designed for cloud infrastructure Optimisations: HPC support with: myHaddop 3C Hierachical computation: [40, 48, 54]	SQL DB	3A Hadoop DB	1B Pegasus		Hadoop 1.0
	HDFS Monitoring: Zookeper Bulk Transfer: Sqoop, Flume Scheduling: Oozie	3A MapReduce Optimisations: Recursive MR	3A Mars		
			3A MR Online		
			3C- Pig		
			3C-Z Hive		
			3C Storm		
			3C Giraph		
			3A+ S4		
			Hama		
			3C Mahout		
			3C Spark	3C Shark	
				3C Spark Streaming	
			AMPLab SIMR		
			Datasalt Splout SQL		
			Tajo		
			Hcatalog		
		3C Hbase		Apache Phoenix	
		3A+ HFSP			
		3A Stream			
		Datasalt Pangool			
		HYPERTABLE			
		FB Corona			
		Pydoop			
		Accumulo			
		Cloudera impala			
		Thrift			
Designed for cloud infrastructure	HDFS 2.0 Optimisations: NFS Gateway [122] Bulk Transfer: Sqoop, Flume Monitoring: Zookeper, Ambari, Hue Scheduling: Oozie, Hoya	3C Yarn	3C MapReduce	Drill	Hadoop 2.0
				3C Impala Z	
				3c- Pig	
				3C- Hive Z (stinger)	
				3C Storm	
			3C Tez	3A+ S4	
				Hama	
				3C Mahout	
			3C- Weave		
			3C Reef	3C Wake	
				3C Tang	
			3C Spark	3C Shark	
				3C Spark Streaming	
			3C Giraph		
			3C Hbase	3C Impala Z	
			3A+ S4		
	Cassandra		Twill		
	Tachyon		Hama		
	Lustre FS		3C Mahout		
	Ceph		DataTorrent		
	Quantcast Files Sytem		Samza (w Kafka)		
			Apache Curator		
			Apache Avro		
			3C Storm		
	RedHat GlusterFS		Stratosphere		
		FB Presto			Hybrid
	MongoDB Connector		All Hadoop Ext.		
	HDFS	Haddop	Hadapt		
	SQL engine	Adaprive Query execution			
	Parallel DB				

A Deprecated
B Not Available
C Up to date

z SQL
y MR
x File Access
w JSON
v ODBC
u User Interface

1 Private Software
2 Close Source
3 Open Source

Online
Interactive
Batch
Graph

Figure 45. Summary of Big Data software

9. References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In Grid Computing Environments Workshop, 2008. GCE â€™08, pages 1 to 10, Nov 2008.

[2] https://www.grid5000.fr

[3] I. Foster. What is the grid? a three point checklist, july 2002. ThreePoint-Check. pdf, 2006.

[4] A. Weiss. Computing in the clouds. networker, 11(4), 2007.

[5] http://cloudcomputing.sys-con.com/node/612375

[6] J. Dean and L. A. Barroso. The tail at scale. Communications of the ACM, 56(2):74 to 80, 2013.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. Communications of the ACM, 53(4):50 to 58, 2010.

[8] B. Rimal, E. Choi, and I. Lumb. A taxonomy and survey of cloud computing systems. In INC, IMS and IDC, 2009. NCM â€™09. Fifth International Joint Conference on, pages 44 to 51, Aug 2009.

[9] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. Commun. ACM, 53(4):50 to 58, Apr. 2010.

[10]M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali. Cloud computing: distributed internet computing for it and scientific research. Internet Computing, IEEE, 13(5):10 to 13, 2009.

[11]D. Laney. 3d data management: Controlling data volume, velocity and variety. META Group Research Note, 6, 2001.

[12]http://www.datascientists.net/what-is-data-science

[13]D. Boyd and K. Crawford. Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. Information, Communication & Society, 15(5):662 to 679, 2012.

[14]L. Manovich. Trending: the promises and the challenges of big social data. 2011.

[15]http://www.cse.wustl.edu/~jain/cse570-13/m_10abd.htm

[16]B. Tudorica and C. Bucur. A comparison between several nosql databases with comments and notes. In Roedunet International Conference (RoEduNet), 2011 10th, pages 1 to 5, June 2011.

[17]http://en.wikipedia.org/wiki/NoSQL

[18]http://db-engines.com

[19]A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A com- parison of approaches to large-scale data analysis. In SIGMOD â€™09: Proceedings of the 35th SIGMOD international conference on Management of data, pages 165 to 178, New York, NY, USA, 2009. ACM.

[20]Software challenges at extreme scale. SciDAC Review, 2010.

[21]S. Sakr, A. Liu, and A. G. Fayoumi. The family of mapreduce and large scale data processing systems. CoRR, abs/1302.2966, 2013.

[22]http://www.lsst.org/lsst/about/technology

[23]G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazonâ€™s highly available key-value store. In ACM SIGOPS Operating Systems Review, volume 41, pages 205 to 220. ACM, 2007.

[24]N. Ruflin, H. Burkhart, and S. Rizzotti. Social-data storage-systems. In Databases and Social Networks, pages 7 to 12. ACM, 2011.

[25]P. A. Bernstein, C. W. Reid, and S. Das. Hyder-a transactional record manager for shared flash. In CIDR, volume 11, pages 9 to 20, 2011.

[26]K. Kambatla, G. Kollias, V. Kumar, and A. Grama. Trends in big data analytics. Journal of Parallel and Distributed Computing, 2014.

[27]J. E. Short, R. E. Bohn, and C. Baru. How much information? 2010 report on enterprise server information. Technical report, UCSD - Global Inforamtion Industry Center, 2011.

[28]L. Amini, H. Andrade, R. Bhagwan, F. Eskesen, R. King, P. Selo, Y. Park, and C. Venkatramani. Spc: A distributed, scalable platform for data mining. In Proceedings of the 4th international workshop on Data mining standards, services and platforms, pages 27 to 37. ACM, 2006.

[29]W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. ACM SIGKDD Explorations Newsletter, 14(2):1 to 5, 2013.

[30]G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 135 to 146. ACM, 2010.

[31]U. Kang, D. H. Chau, and C. Faloutsos. Pegasus: Mining billion-scale graphs in the cloud. In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, pages 5341 to 5344. IEEE, 2012.

[32]Yahoo! reaches for the stars with m45 supercomputing project. http://research.yahoo.com/node/1879.

[33]D. Warneke and O. Kao. Nephele: efficient parallel data processing in the cloud. In Proceedings of the 2nd workshop on many-task computing on grids and supercomputers, page 8. ACM, 2009.

[34]S. Krishnan, M. Tatineni, and C. Baru. myhadoop-hadoop-on-demand on traditional hpc resources. San Diego Supercomputer Center Technical Report TR-2011-2, University of California, San Diego, 2011.

[35]R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the data: Parallel analysis with sawzall. Scientific Programming, 13(4):277 to 298, 2005.

[36]J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin. Flexible io and integration for scientific codes through the adaptable io system (adios). In Proceedings of the 6th international workshop on Challenges of large applications in distributed environments, pages 15 to 24. ACM, 2008.

[37]J. Li, W.-k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale. Parallel netcdf: A high-performance scientific i/o interface. In Supercomputing, 2003 ACM/IEEE Conference, pages 39 to 39. IEEE, 2003.

[38]S. Loesing, M. Pilman, T. Etter, and D. Kossmann. On the design and scalability of distributed shared- memory databases. Technical report, Eidgen Ì^ossische Technische Hochschule Zu Ì^rich, 2013.

[39]Y. Luo and B. Plale. Hierarchical mapreduce programming model and scheduling algorithms. In Pro- ceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pages 769 to 774. IEEE Computer Society, 2012.

[40]A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin. Hadoopdb: An archi- tectural hybrid of mapreduce and dbms technologies for analytical workloads. Proc. VLDB Endow., 2(1):922 to 933, Aug. 2009.

[41]J. Talbot, R. M. Yoo, and C. Kozyrakis. Phoenix++: modular mapreduce for shared-memory systems. In Proceedings of the second international workshop on MapReduce and its applications, pages 9 to 16. ACM, 2011.

[42]K. Saroj. Hadoop 2.0 to a big step for big data - http://cloudtimes.org/2014/01/24/hadoop-2-0-a-big- step-for-big-data-momentum/.

[43]K. Saroj. Idc report to hadoop leads the big data analytics tool for enterprises - http://cloudtimes.org/2013/11/06/idc-report-hadoop-leads-the-big-data-analytics-tool-for-enterprises/.

[44]V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. Oâ€™Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC â€™13, pages 5:1 to 5:16, New York, NY, USA, 2013. ACM.

[45]M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review, 41(3):59 to 72, 2007.

[46]A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment, 2(2):1626 to 1629, 2009.

[47]Z. Xiao, H. Chen, and B. Zang. A hierarchical approach to maximizing mapreduce efficiency. In PACT, pages 167–168, 2011.

[48]G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang. Introducing map-reduce to high end computing. In Petascale Data Storage Workshop, 2008. PDSWâ€™08. 3rd, pages 1 to 6. IEEE, 2008.

[49]A. Monsen. Introduction to MapReduce with Hadoop on Linux - http://www.linuxjournal.com/content/introduction-mapreduce-hadoop-linux.

[50]R. McCreadie, C. Macdonald, and I. Ounis. Mapreduce indexing strategies: Studying scalability and efficiency. Information Processing & Management, 48(5):873 to 888, 2012.

[51]R. M. Yoo, A. Romano, and C. Kozyrakis. Phoenix rebirth: Scalable mapreduce on a large-scale shared- memory system. In Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on, pages 198 to 207. IEEE, 2009.

[52]A. Fox, C. Eichelberger, J. Hughes, and S. Lyon. Spatio-temporal indexing in non-relational distributed databases. In Big Data, 2013 IEEE International Conference on, pages 291 to 299. IEEE, 2013.

[53]Y. Luo, Z. Guo, Y. Sun, B. Plale, J. Qiu, and W. W. Li. A hierarchical framework for cross-domain mapreduce execution. In Proceedings of the second international workshop on Emerging computational methods for the life sciences, pages 15 to 22. ACM, 2011.

[54]The Next Generation of Apache Hadoop MapReduce - http://developer.yahoo.com/blogs/hadoop/next-generation-apache-hadoop-mapreduce-3061.html.

[55]M. Elteir, H. Lin, and W.-c. Feng. Enhancing mapreduce via asynchronous data processing. In Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on, pages 397 to 405. IEEE, 2010.

[56]F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.

[57]M. Pastorelli, A. Barbuzzi, D. Carra, M. Dellâ€™Amico, and P. Michiardi. Hfsp: size-based scheduling for hadoop. In Big Data, 2013 IEEE International Conference on, pages 51 to 59. IEEE, 2013.

[58]R. Lammel. Googleâ€™s mapreduce programming modelâ€”revisited. Science of computer programming, 70(1):1 to 30, 2008.

[59]F. N. Afrati, V. Borkar, M. Carey, N. Polyzotis, and J. D. Ullman. Map-reduce extensions and recursive queries. In Proceedings of the 14th international conference on extending database technology, pages 1 to 8. ACM, 2011.

[60]V. Granville. What MapReduce can't do- http://www.analyticbridge.com/profiles/blogs/what-mapreduce-can-t-do.

[61]T. Graves. Graysort and minutesort at yahoo on hadoop 0.23. Technical report, Yahoo!, 2013.

[62]P. G. Brown. Overview of scidb: large scale array storage, processing and analysis. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 963 to 968. ACM, 2010.

[63]J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares, and X. Qin. Improving mapre-duce performance through data placement in heterogeneous hadoop clusters. In Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, pages 1 to 9. IEEE, 2010.

[64]B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang. Mars: a mapreduce framework on graphics processors. In Proceedings of the 17th international conference on Parallel architectures and compilation techniques, pages 260 to 269. ACM, 2008.

[65]M. Cardosa, C. Wang, A. Nangia, A. Chandra, and J. Weissman. Exploring mapreduce efficiency with highly-distributed data. In Proceedings of the second international workshop on MapReduce and its applications, pages 27 to 34. ACM, 2011.

[66]A. Matsunaga, M. Tsugawa, and J. Fortes. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In eScience, 2008. eScienceâ€™08. IEEE Fourth In- ternational Conference on, pages 222 to 229. IEEE, 2008.

[67]K. Kovacs, http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis

[68]A. Seering, P. Cudre-Mauroux, S. Madden, and M. Stonebraker. Efficient versioning for scientific array databases. In Data Engineering (ICDE), 2012 IEEE 28th International Conference on, pages 1013 to 1024. IEEE, 2012.

[69]D. Nelubin and B. Engber. Ultra-high performance nosql benchmarking. Technical report, Thumbtack Technology, 2013.

[70]L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In Data Mining Workshops (ICDMW), 2010 IEEE International Conference on, pages 170 to 177. IEEE, 2010.

[71]A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema. Performance anal- ysis of cloud computing services for many-tasks scientific computing. Parallel and Distributed Systems, IEEE Transactions on, 22(6):931 to 945, 2011.

[72]D. Harris, Facebook has built a prototype system for storing petabytes on Blu-ray, http://gigaom.com/2014/01/28/facebook-has-built-a-prototype-system-for-storing-petabytes-on-blu-ray/

[73]J. Lin and A. Kolcz. Large-scale machine learning at twitter. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pages 793 to 804. ACM, 2012.

[74]A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. Stream: the stanford stream data manager (demonstration description). In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 665 to 665. ACM, 2003.

[75]B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!â€™s hosted data serving platform. Proceedings of the VLDB Endowment, 1(2):1277 to 1288, 2008.

[76]S. A. Weil, S. A. Brandt, E. L. Miller, D. D. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In Proceedings of the 7th symposium on Operating systems design and implemen- tation, pages 307 to 320. USENIX Association, 2006.

[77]B. Nicolae and F. Cappello. Blobcr: efficient checkpoint-restart for hpc applications on iaas clouds using virtual disk image snapshots. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, page 34. ACM, 2011.

[78]B. Lawrence, V. Bennett, J. Churchill, M. Juckes, P. Kershaw, S. Pascoe, S. Pepler, M. Pritchard, and A. Stephens. Storing and manipulating environmental big data with jasmin. In Big Data, 2013 IEEE International Conference on, pages 68 to 75. IEEE, 2013.

[79]J. MacCormick, N. Murphy, M. Najork, C. A. Thekkath, and L. Zhou. Boxwood: Abstractions as the foundation for storage infrastructure. In OSDI, volume 4, pages 8 to 8, 2004.

[80]T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. Mapreduce online. In NSDI, volume 10, page 20, 2010.

[81]T. Honjo and K. Oikawa. Hardware acceleration of hadoop mapreduce. In Big Data, 2013 IEEE Inter- national Conference on, pages 118 to 124. IEEE, 2013.

[82]S. N. Srirama, P. Jakovits, and E. Vainikko. Adapting scientific computing problems to clouds using mapreduce. Future Generation Computer Systems, 28(1):184 to 192, 2012.

[83]S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. A performance analysis of ec2 cloud computing services for scientific computing. In Cloud Computing, pages 115 to 131. Springer, 2010.

[84]A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pages 1 to 14. ACM, 2010.

[85]H. Schaffer. X as a service, cloud computing, and the need for good judgment. IT Professional, 11(5):4 to 5, Sept 2009.

[86]X. Xu. From cloud computing to cloud manufacturing. Robotics and computer-integrated manufacturing, 28(1):75 to 86, 2012.

[87]ivy Wigmore, http://whatis.techtarget.com/definition/3Vs, 2013

[88]M. Seeger and S. Ultra-Large-Sites. Key-value stores: a practical overview. Computer Science and Media, page 23, 2009.

[89]J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Commun. ACM, 51(1):107 to 113, Jan. 2008.

[90]Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst. Haloop: Efficient iterative data processing on large clusters. Proceedings of the VLDB Endowment, 3(1-2):285 to 296, 2010.

[91]http://lustre.opensfs.org

[92]Using lustre with apache hadoop. Technical report, Sun Microsystems Inc.

[93]S. Bai and H. Wu. The performance study on several distributed file systems. 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 0:226 to 229, 2011.

[94]http://en.wikipedia.org/wiki/Apache_Hadoop

[95]http://hadoop.apache.org

[96]https://mahout.apache.org

[97]http://en.wikipedia.org/wiki/Clustered_file_system

[98]http://db-engines.com/de/blog_post/29

[99]Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan. Mariane: Mapreduce implementation adapted for hpc environments. In Grid Computing (GRID), 2011 12th IEEE/ACM International Con- ference on, pages 82 to 89. IEEE, 2011.

[100] https://github.com/glennklockwood/myhadoop

[101] http://users.sdsc.edu/~glockwood/di/hadoop-hpc.php

[102] E. Dede, Z. Fadika, J. Hartog, M. Govindaraju, L. Ramakrishnan, D. Gunter, and R. Canon. Marissa: Mapreduce implementation for streaming science applications. In E-Science (e-Science), 2012 IEEE 8th International Conference on, pages 1 to 8, Oct 2012.

[103] Z. Fadika, E. Dede, J. Hartog, and M. Govindaraju. Marla: Mapreduce for heterogeneous clusters. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Comput- ing (ccgrid 2012), pages 49 to 56. IEEE Computer Society, 2012.

[104] http://www.hpcwire.com/2014/02/11/hpc-hacking-hadoop/

[105] http://www.hpcwire.com/2014/02/14/adapting-hadoop-hpc-environments/

[106] http://users.sdsc.edu/~glockwood/comp/mapreduce.php

[107] http://www.glennklockwood.com/di/hadoop-overview.php

[108] https://spark.incubator.apache.org

[109] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, pages 10 to 10, 2010.

[110] http://hothardware.com/Reviews/Intel-Announces-MIC-Xeon-Phi-Aims-For-Exascale-Computing/

[111] B.-G. Chun, T. Condie, C. Curino, C. Douglas, S. Matusevych, B. Myers, S. Narayanamurthy, R. Ra- makrishnan, S. Rao, J. Rosen, et al. Reef: Retainable evaluator execution framework. Proceedings of the VLDB Endowment, 6(12):1370 to 1373, 2013.

[112] https://github.com/Microsoft-CISL/REEF

[113] http://hortonworks.com/hadoop/tez/

[114] http://twill.incubator.apache.org/index.html

[115] http://projects.spring.io/spring-hadoop/

[116] http://hortonworks.com/blog/hdfs-2-0-next-generation-architecture/

[117] http://hadoop.apache.org/docs/r1.2.1/streaming.htm

[118] http://hortonworks.com/blog/simplifying-data-management-nfs-access-to-hdfs/

[119] http://amplab.cs.berkeley.edu/benchmark/

[120] http://www.ciol.com/ciol/news/210540/vmware-launches-hybrid-desktop-service-daas

[121] http://slideshare.net/hortonworks/nextgen-apache-hadoop-mapreduce

[122] http://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/

[123] http://hortonworks.com/hadoop/hcatalog/

[124] https://hadapt.com

[125] http://hortonworks.com/blog/hoya-hbase-on-yarn-application-architecture/

[126] https://www.datatorrent.com

[127] http://kafka.apache.org

[128] http://hortonworks.com/labs/openstack/

[129] http://samza.incubator.apache.org

[130] https://hama.apache.org

[131] https://giraph.apache.org

[132] http://gethue.com

[133] http://ambari.apache.org

[134] https://oozie.apache.org

[135] http://flume.apache.org

[136] http://sqoop.apache.org

[137] http://zookeeper.apache.org

[138] http://hortonworks.com/labs/stinger/

[139] http://www.dbms2.com/2013/08/06/hortonworks-hadoop-stinger-and-hive/

[140] http://www.cloud-lounge.org/EN/clouds-and-grids-compared.html

[141] http://www.mongodb.com/learn/big-data

[142] http://www.mongodb.com/learn/hadoop

[143] http://map-reduce.wikispaces.asu.edu

[144] http://www.datasciencecentral.com/forum/topics/the-3vs-that-define-big-data

[145] http://googlecloudplatform.blogspot.fr/2014/03/cassandra-hits-one-million-writes-per-second-on-google-compute-engine.html

[146] AHRONOVITZ, M., AMRHEIN, D., ANDERSON, P., et al. Cloud computing use cases white paper. 2010.

[147] BigFoot project, D4.1, Scalable Distributed File and Database Systems: Design and Specification.

[148] Fei Dong, Tianyu Feng, Hong Zhang, Hadoop System simulation with Mumak, 2010

[149] http://bigfootproject.eu/software.html

[150] http://www.businessinsider.com/10-most-important-in-cloud-computing-2013-4?op=1

[151] Chen, Y., Alspaugh, S., & Katz, R. (2012). Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads. Proceedings of the VLDB Endowment, 5(12), 1802-1813.

[152] Fadika, Z., Dede, E., Govindaraju, M., & Ramakrishnan, L. (2011, September). Benchmarking mapreduce implementations for application usage scenarios. In Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on (pp. 90-97). IEEE.

[153] http://gigaom.com/2013/03/05/the-hadoop-ecosystem-the-welcome-elephant-in-the-room-infographic/

[154] G. Rommel, Hadoop 1.x vs Hadoop 2.

[155] A. Murthy, J. Markham, V. K. Vavilapalli and D. Eadline, apache hadoop yarn moving beyond mapreduce and batch processing with apache hadoop 2.

[156] http://hadoop.intel.com/products/hadoop-hpc

[157] IntelÂ® Distribution for Apache Hadoop* Software: Optimization and Tuning Guide

[158] http://hadoopecosystemtable.github.io

[159] Harford, T.,Big data: are we making a big mistake?, http://www.ft.com/intl/cms/s/2/21a6e7d8-b479-11e3-a09a-00144feabdc0.html#axzz2xcBtEODA

[160] http://www.revelytix.com/?q=content/hadoop-ecosystem

[161] Kumar, A., Karampatziakis, N., Mineiro, P., Weimer, M., & Narayanan, V. K. Distributed and Scalable PCA in the Cloud.

[162] Engle, C., Lupher, A., Xin, R., Zaharia, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012, May). Shark: fast data analysis using coarse-grained distributed memory. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (pp. 689-692). ACM.

[163] Moving to the Cloud, A white paper produced by the Cloud Computing Use Cases Discussion Group

[164] http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=13251p

[165] Dong, F. (2012). Extending Starfish to Support the Growing Hadoop Ecosystem (Doctoral dissertation, Duke University).

[166] http://www.hpcloud.com

[167] Georgiou, A. (2013). Storing Data Flow Monitoring in Hadoop (No. CERN-STUDENTS-Note-2013-144).

[168] Bamotra, P. Hadoop and Risk Analytics.

[169] Padhy, R. P. (2012). Big Data Processing with Hadoop-MapReduce in Cloud Systems. International Journal of Cloud Computing and Services Science (IJ-CLOSER), 2(1), 16-27.

[170] Huddar, M. M. G., & Ramannavar, M. M. A Survey on Big Data Analytical Tools.

[171] http://www.infoivy.com/2013/08/comparison-table-of-interactive.html

[172] Tu, T., Rendleman, C. A., Borhani, D. W., Dror, R. O., Gullingsrud, J., Jensen, M. O., ... & Shaw, D. E. (2008, November). A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories. In High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for (pp. 1-12). IEEE.

[173] Plimpton, S. J., & Devine, K. D. (2011). MapReduce in MPI for large-scale graph algorithms. Parallel Computing, 37(9), 610-632.

[174] http://www.hightechdad.com/2013/03/22/15-top-cloud-computing-use-cases/

[175] Fadika, Z., Govindaraju, M., Canon, R., & Ramakrishnan, L. (2012, June). Evaluating hadoop for data-intensive scientific operations. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 67-74). IEEE.

[176] Sehrish, S., Mackey, G., Wang, J., & Bent, J. (2010, June). Mrap: a novel mapreduce-based framework to support hpc analytics applications with access patterns. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (pp. 107-118). ACM.

[177] http://en.wikipedia.org/wiki/CloudStore

[178] Molina-Estolano, E., Gokhale, M., Maltzahn, C., May, J., Bent, J., & Brandt, S. (2009, November). Mixing Hadoop and HPC workloads on parallel filesystems. In Proceedings of the 4th Annual Workshop on Petascale Data Storage (pp. 1-5). ACM.

[179] https://hadoop.apache.org/docs/r0.18.3/hod.html

[180] http://bowtie-bio.sourceforge.net/crossbow/index.shtml

[181] http://www.ncbi.nlm.nih.gov/pubmed/24149054

[182] http://sourceforge.net/apps/mediawiki/cloudburst-bio/index.php?title=CloudBurst

[183] https://issues.apache.org/jira/browse/MAPREDUCE-2911

[184] http://www.cloudbus.org/cloudsim/

[185] http://simgrid.gforge.inria.fr

[186] Casanova, H., Legrand, A., & Quinson, M. (2008, April). Simgrid: A generic framework for large-scale distributed experiments. In Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on (pp. 126-131). IEEE.

[187] Veiga Neves, M., Ferreto, T. and De Rose, C., Scheduling MapReduce Jobs in HPC Clusters, Euro-Par 2012 Parallel Processing, pp 179-190, 2012

[188] http://www.asterdata.com/product/advanced-analytics.php

[189] Murthy, A. C., Douglas, C., Konar, M., Oâ€™Malley, O., Radia, S., Agarwal, S., & KV, V. (2011). Architecture of next generation Apache Hadoop MapReduce framework. Tech. rep., Apache Hadoop.

[190] Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., & Rowstron, A. (2013, October). Scale-up vs Scale-out for Hadoop: Time to rethink?. In Proceedings of the 4th annual Symposium on Cloud Computing (p. 20). ACM.

[191] http://blog.codingoutloud.com/2010/07/27/three-types-of-scaling-in-the-cloud-scale-up-scale-out-and-now-scale-side-by-side-with-juxtaposition-scaling/

[192] Chambers, C., Raniwala, A., Perry, F., Adams, S., Henry, R. R., Bradshaw, R., & Weizenbaum, N. (2010, June). FlumeJava: easy, efficient data-parallel pipelines. In ACM Sigplan Notices (Vol. 45, No. 6, pp. 363-375). ACM.

[193] Dittrich, J., QuianÃ©-Ruiz, J. A., Jindal, A., Kargin, Y., Setty, V., & Schad, J. (2010). Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). Proceedings of the VLDB Endowment, 3(1-2), 515-529.

[194] http://mrql.incubator.apache.org

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

[196] https://wiki.openstack.org/wiki/Sahara

[197] http://mapreduce.sandia.gov

[198] http://www.sevenforge.com/meguro/

[199] https://aws.amazon.com/ec2/

[200] http://aws.amazon.com/s3

[201] Brewer E. A., Towards Robust Distributed Systems

[202] http://nosql-database.org

[203] Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications, 1(1), 7-18.

[204] http://moosefs.org/

[205] M. Cafarella, D. Cutting, Building Nutch: Open source search, Queue, 2 (2) (2004), pp. 54 à 61.

[206] http://www.greentechmedia.com/articles/read/edfs-big-data-vision-for-france

[207] http://research.microsoft.com/en-us/projects/dryad/

[208] http://projects.apache.org/indexes/category.html

[209] http://www.infoivy.com/2013/11/comparison-table-of-hive-pig-shark.html

[210] http://aws.amazon.com/fr/hpc-applications/getting-started/

[211] http://sourceforge.net/apps/mediawiki/cloudflu/index.php?title=Main_Page

[212] http://www.salesforce.com

[213] http://beanstalkapp.com

[214] https://www.heroku.com

[215] Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). The Google file system. In ACM SIGOPS Operating Systems Review (Vol. 37, No. 5, pp. 29-43). ACM.

[216] http://www.hortonworks.com

[217] http://www.cloudera.com

[218] http://en.wikipedia.org/wiki/Watson_(computer)

[219] http://teradata.com

[220] Philippe Botteri of Accel Partners, Feb. 2013

[221] http://blogs.the451group.com/opensource/2012/05/23/451-research-delivers-market-sizing-estimates-for-nosql-newsql-and-mysql-ecosystem/

[222] Chaiken, R., Jenkins, B., Larson, P. Å., Ramsey, B., Shakib, D., Weaver, S., & Zhou, J. (2008). SCOPE: easy and efficient parallel processing of massive data sets. Proceedings of the VLDB Endowment, 1(2), 1265-1276.

[223] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R., ... & Stoica, I. (2011, March). Mesos: A platform for fine-grained resource sharing in the data center. In Proceedings of the 8th USENIX conference on Networked systems design and implementation (pp. 22-22).

[224] Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., & Wilkes, J. (2013, April). Omega: flexible, scalable schedulers for large compute clusters. In Proceedings of the 8th ACM European Conference on Computer Systems (pp. 351-364). ACM.

[225] http://www.jotneit.no/products/express-data-manager-edm

[226] http://www.jotneit.no/images/pdf/EXPRESS_White_Paper.pdf

[227] http://www.jotneit.no/products/edmopensimdm

[228] www.cimdata.com/de/component/docman/doc_download/1659-jotne-epm-s-simdm-program-managing-simulation-and-analysis-data-and-activities

[229] http://edmserver.epmtech.jotne.com/EDMAssist/WebHelp/EDMassist.htm.

[230] The data deluge, the economist, 2010

[231] http://www.comptechdoc.org/independent/database/basicdb/dataobject.html

[232] Yuan, H., Kuo, C. C., & Ahmad, I. (2010, August). Energy efficiency in data centers and cloud-based multimedia services: An overview and future directions. In Green Computing Conference, 2010 International (pp. 375-382). IEEE.

[233] Virtualizing Apache Hadoop, vmware, 2012

[234] Cloud Benchmark – round 1, Techila, 2014

[235] http://www.hadoopsphere.com/2013/06/options-for-mapreduce-with-hpc.html

[236] https://github.com/MicrosoftResearch/Dryad

[237] http://bigdatacompilation.blogspot.fr/2014/06/products-that-include-apache-hadoop-or.html

[238] http://blog.gopivotal.com/pivotal/products/usage-and-quirks-of-fs-default-name-in-hadoop-filesystem

[239] Kulkarni, O., Hadoop MapReduce over Lustre, Intel, 2013

[240] https://wiki.apache.org/hadoop/HCFS

[241] Loebman, S., Nunley, D., Kwon, Y. C., Howe, B., Balazinska, M., & Gardner, J. P. (2009, August). Analyzing massive astrophysical datasets: Can Pig/Hadoop or a relational DBMS help?. In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on (pp. 1-10). IEEE.

10.Index

11.List of figures