

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

185,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Towards Large Scale Image Retrieval System Using Parallel Frameworks

Saliha Mezzoudj

Abstract

Recently, the increasing use of mobile devices, such as cameras and smartphones, has resulted in a dramatic increase in the amount of images collected every day. Therefore, retrieving and managing these large volumes of images has become a major challenge in the field of computer vision. One of the solutions for efficiently managing image databases is an Image Content Search (CBIR) system. For this, we introduce in this chapter some fundamental theories of content-based image retrieval for large scale databases using Parallel frameworks. Section 2 and Section 3 presents the basic methods of content-based image retrieval. Then, as the emphasis of this chapter, we introduce in Section 1.2 A content-based image retrieval system for large-scale images databases. After that, we briefly address Big Data, Big Data processing platforms for large scale image retrieval. In Sections 5, 6, 7, and 8. Finally, we draw a conclusion in Section 9.

Keywords: big data processing platforms, image retrieval system, big data, parallel frameworks

1. Introduction

Computer vision (also called artificial vision or digital vision) is a branch of artificial intelligence whose main goal is to allow a machine to analyze, process and understand one or more images taken by an acquisition system (example: cameras, mobile, etc.) [1]. It is used to automate the tasks that the human visual system can do: recognition, motion analysis, scene reconstruction, and image restoration [1]. In this chapter, we are interested in the recognition task, there are several specialized applications based on recognition exist, such as content image search (CBIR) and image classification systems. Image classification is an important task in the field of computer vision, and it requires the development of robust classification systems, which can improve the performance of vision systems. Indeed, most image CBIR systems have three stages:

- The first step: it is the extraction of low-level characteristics of the images (extraction of descriptors). Indeed, the use of low-level image descriptors is the core of current image classification systems.
- the searching step, in which the feature vector of a query image is computed and compared to the image feature vectors of the database. As a result, the CBIR system returns the closest images to the user [2, 3].

For a long time, high calculation errands caused by calculating complexity and gigantic amount of image during indexing, and retrieving steps have been obstacles for building a CBIR systems [3, 4]. Furthermore, the conventional content-based image retrieval systems have focused on small databases of face images. Therefore, it is important to generalize and train these systems on large-scale databases [5, 6]. Therefore, in this chapter, we will present the basics of CBIR systems for large-scale databases, Big Data, Big Data processing platforms for large scale image retrieval.

2. The methods for extracting visual characteristics

The extraction of features from images is the basis of any computer vision system that does recognizing. These characteristics can contain both text (keywords; annotations, etc.), and visual characteristics (color, texture, shapes, faces, etc.). We will focus on techniques for extracting these visual features only. And for that the visual characteristics (descriptors) are classified in two categories general descriptors and specific domain descriptors [7, 8]:

2.1 General descriptors

They contain low-level descriptors that give a description of color, shape, regions, textures and movement.

Color: Color is one of the most used visual characteristics in facial recognition systems or anything like that. It is relatively robust to the complexities of the background and independently of the size and orientation of the image. The most well-known representation of color is the histogram, which denotes the frequencies of occurrence of the intensities of the three color channels. Many other representations of this characteristic exist: we speak especially of the moments of color. The mathematical basis of this approach is that each color distribution can be characterized by its color moments. Furthermore, most of the information on color is concentrated on lower order moments which are respectively: mean, standard deviation, color skewness, variance, median, etc.

Texture: A wide variety of texture descriptors have been proposed in the literature. These were traditionally divided into statistical, spectral, structural and hybrid [9] approaches. Among the most popular traditional methods are probably those based on histograms, Gabor filters [10], co-occurrence matrices [11] and models (lbp) [12]. These descriptors present various strengths and weaknesses, in particular as regards their invariance with respect to the acquisition conditions.

Shape: Over the past two decades, 2D shape descriptors have been actively used in 3D search engines and sketch-based modeling techniques. Some of the most popular 2D shape descriptors are curvature scale space (CSS) [13], SIFT [14], and SURF [15]. In fact, in the literature, 2D shape descriptors are classified into two main categories: contours and regions. Outline-based shape descriptors extract shape entities from the outline of a shape only. In contrast, region-based shape descriptors obtain shape characteristics of the entire region of a shape. In addition, hybrid techniques have also been proposed, combining techniques based on the contour and the [16] region.

Movement: Movement is related to the movement of objects in the sequence and to the movement of the camera. The latter information is provided by the capture device, while the rest is implemented by means of image processing. The set of descriptors is the following [7]: Motion Activity Descriptor (MAD), Camera Motion Descriptor (CMD), Motion Trajectory Descriptor (MTD), and Warp and Parametric Motion Descriptor (WMD and PMD).

Location: The location of items in the image is used to describe items in the spatial domain. In addition, elements can also be located in the [7] time domain: Region Locator Descriptor (RLD), Spatio Time Locator Descriptor (STLD).

3. Image classification

Image classification is an important step in the image recognition process. Indeed, many image classification techniques have been proposed to date. It is considered to be one of the main types of machine learning. Various studies have been carried out in order to choose the best technique for classifying [17] images.

3.1 What is machine learning?

It is one of the subdomains of artificial intelligence (AI) which uses a series of techniques to let computers learn, (that is, gradually improve the performance of the computer on a task specific) with data, without being explicitly programmed. Indeed, machine learning covers a vast field of tasks. Below are the types of machine learning described in this section [18]:

Supervised learning (classification): In this case, the entries are tagged by an expert, and the algorithm must learn from the tags of these entries in order to predict the class of each new entry. In other words, from a set of observations X and another set of measures Y , we seek to estimate the dependencies between X and Y .

Unsupervised learning (clustering): In this case, the entries are not labeled, no expert is available, and the algorithm must predict the class of each entry. The objective of this type of learning is to describe how the data is organized and to extract homogeneous subsets.

Semi-supervised learning: the algorithm combines labeled and unlabeled examples to generate an appropriate function or class.

Learning to learn: where the algorithm learns its own inductive bias based on previous experience.

4. A content-based image retrieval system for large-scale images databases

Indeed, the most of conventional CBIR systems are evaluated on small bases of images that fit easily in main memory, such as Caltech-101 [19], Caltech-256 [20] or PASCAL VOC [21].

Recently, the increase in images produced in different fields has enabled the acquisition and storage of a large amount of images, which offers new concepts such as Big Data, which are of huge volumes of images from a variety of sources, produced in real time and exceeding the storage capacity of a single machine. Indeed, these images are difficult to process with traditional image retrieval systems.

As digital cameras become more affordable and ubiquitous, digital images are growing exponentially on the Internet, such as ImageNet 1 [22] which consists of 14,197,122 images labeled for 21,841 classes. Indeed, this enormous quantity of images makes the task of classification of images much more complex and difficult to perform, especially since traditional processing and storage methods do not always manage to cope with this enormous quantity of images.

This challenge motivated us to develop a new image search and classification system allowing the storage, management and processing of large quantities of

images (Big Data), this imposes a parallelisation of calculations to obtain results in reasonable time, and optimum precision.

Massively parallel machines are more and more available at increasingly affordable costs, such is the case with multiprocessors. This justifies our motivation to direct our research efforts in large-scale image classification towards the exploitation of such architectures with new Big Data platforms that use the performance of these machines.

5. The basics of big data

Every day, we generate trillions of bytes of data (Big Data). This data comes from everywhere: from sensors used to collect climate information, messages on social media sites, digital images and videos posted online, transactional records of online purchases and GPS signals from phones mobile, to name a few sources.

Big Data is characterized by its volume (massive data); they are also known for their variety in terms of formats and new structures, as well as a requirement in terms of speed in processing. But until now, according to our research, no software is able to handle all this data which has many types and forms and which is growing very rapidly. So Big Data issues are part of our daily life, and more advanced solutions are needed to manage this mass of data in a short time.

Distributed computing is concerned with processing large amounts of data. This processing cannot be achieved with traditional data processing paradigms, it requires the use of distributed platforms. In the literature, there are several solutions, for the implementation of this paradigm. Among these solutions we find the example Google, which has developed a very reliable programming model for the processing of Big Data: it is the MapReduce model. This model is implemented on several platforms such as the Hadoop platform. Despite all these advantages, Hadoop suffers from latency problems which is the main cause of development of a new alternative to improve the performance of processing data, it is the Spark platform which is more powerful, more flexible and faster than Hadoop MapReduce.

In this chapter, we will explain the basics of Big Data, Big Data processing platforms, as well as storage.

5.1 Definition

Big Data refers to a very large volume of often heterogeneous data which has several forms and formats (text, sensor data, sound, video, route data, log files, etc.), and including heterogeneous formats: structured data, unstructured and semi-structured. Big Data has a complex nature that requires powerful technologies and advanced algorithms for its processing and storage. Thus, it cannot be processed using tools such as the traditional DBMS [23]. Most scientists and data experts define big data with the concept of 3Vs as follows [23]:

- **Velocity:** Data is generated quickly and must be processed quickly to extract useful information and relevant information. For example, Walmart (an international chain of discount retailers) generates over 2.5 petabytes (PB) of data every hour from its customers' transactions. YouTube is another good example of the fast speed of big data.
- **Variety:** Big data are generated from various sources distributed in multiple formats (e.g. videos, documents, commentaries, journals). Large data sets

include structured and unstructured data, public or private, local or remote, shared or confidential, complete or incomplete, etc.

- Volume: it represents the amount of data generated, stored and used. The volume of data stored today is exploding, it is almost 800,000 petabytes, Twitter generates more than 7 terabytes of data every day, Facebook generates more than 10 terabytes and the data volume in 2020 can reach 40 zeta bytes (**Figure 1**) [24].

Thereafter, the three original dimensions are widened by two other dimensions of big data (also known as the “5 V Big Data”):

- Truth: Truthfulness (or validity) of data is the reliability and accuracy of data, and the confidence that big data inspires in decision-makers. If the users of this data doubt its quality or relevance, it becomes difficult to invest more in it.
- Value: This last V plays a key role in Big Data, the Big Data approach only makes sense to achieve strategic goals of creating value for customers and for companies in all areas (**Figure 2**).

One of the reasons for the emergence of the concept of Big Data is the need to realize the technical challenge of processing large volumes of information of several types (structured, semi-structured and unstructured) generated at high speed. Big Data is based on four data sources [25]:

1. The logs (connection logs) from traffic on the company’s official website: These data sources are the paths taken by visitors to reach the site: search engines, directories, bounces from other sites, etc. Businesses today have a web storefront through its official website. The latter generates traffic that it is essential to analyze, so these companies have trackers on the different pages in order to measure the navigation paths, or the time spent on each page, etc. Some of the best-known analytics solutions include: Google Analytics, Adobe Omniture, Coremetrics.

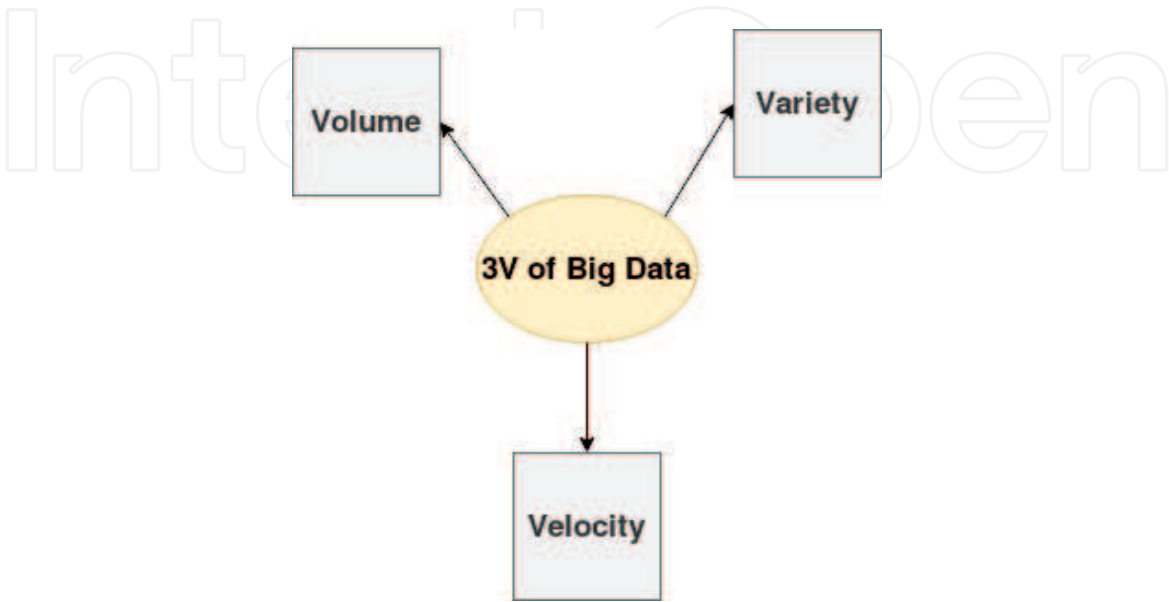


Figure 1.
The 3 V big data model.

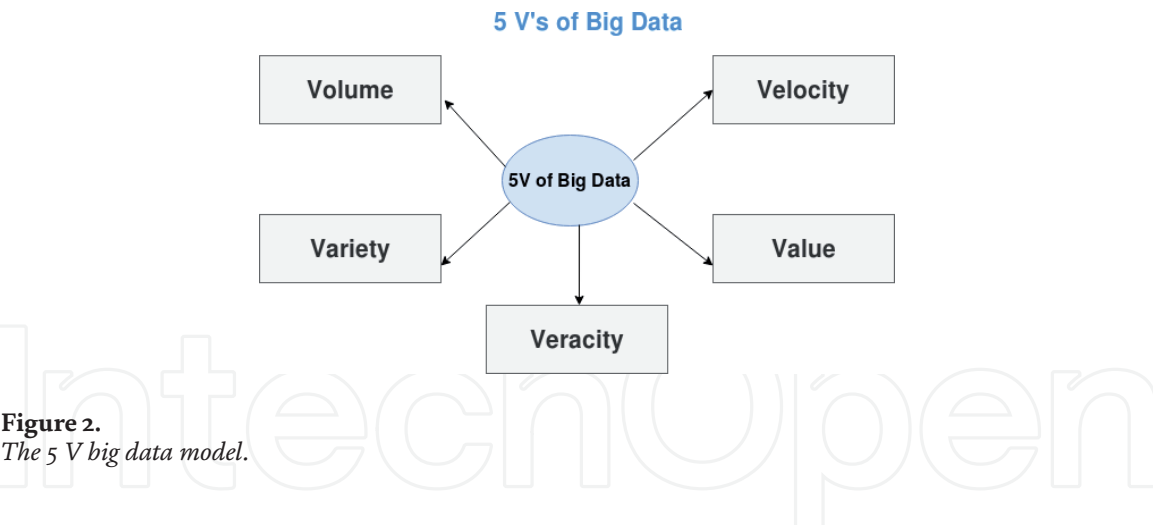


Figure 2.
The 5 V big data model.

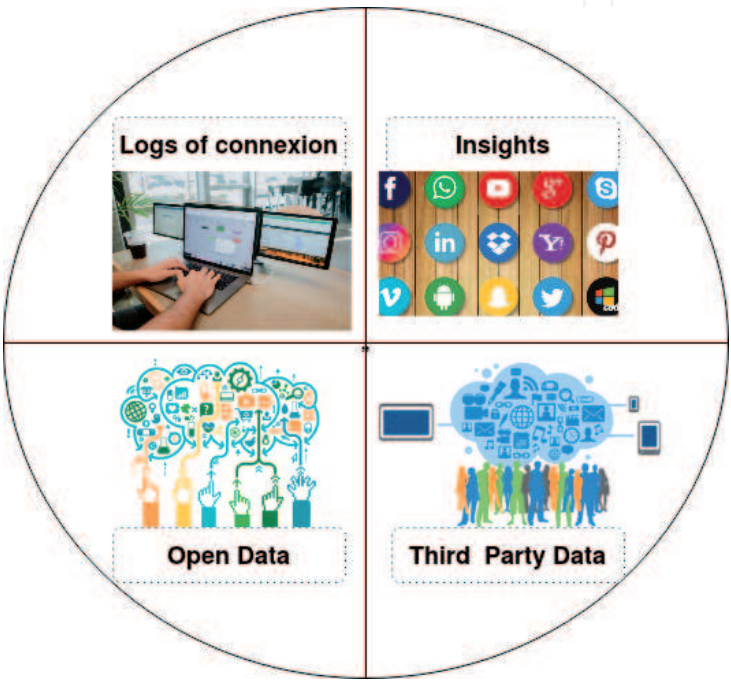


Figure 3.
The four sources of big data.

- 2. Social media insights: A complementary approach is to collect comments on posts and apply sentiment analysis algorithms to them. Let us mention a few avenues to follow our various accounts: Hootsuite, Radian6 or even the APIs made available and queried with the Power Query add-in for Excel, IRaMuTeQ for the analysis of textual data.
- 3. Behavioral data (third party data) These data are all data on Internet users collected via forms or cookies. Beyond traditional identity information (sex, age, CSP, etc.), it is now much more efficient to measure behavior (navigation, hardware configuration, time spent on pages, etc.). For this, there are specialized web players who help us collect information on our customers or prospects and thus improve communication campaigns. Some players in the field of third party data: Bluekai, Exelate, Weborama, Datalogix, etc.
- 4. Open and reusable data “Open data” are all open and reusable data, it makes possible to put open data online, to make the data more reliable and to make them reusable and usable, where openness consists in making the data public: free of rights, downloadable, reusable and free. The opening does not apply

to private data, sensitive and security information, documents protected by copyright, etc. (Figure 3).¹

5.2 Big data processing and storage technologies

Big Data requires redefining the data storage and processing systems that can support this volume of data. Indeed, several technologies have been proposed in order to represent this data, these technologies take at least one axis between the two, either improving storage capacities or improving computing power [23]:

- Improved computing power: the goal of these techniques is to allow processing on a large set of data, at considerable cost, and to improve execution performance such as processing time and tolerance breakdowns. Before the appearance of the Hadoop platform, there were several technologies such as Cloud Computing, massively parallel MPP architectures and In-Memory technologies.
- Improvement of storage capacities: improvement of storage of distributed systems, where the same file can be distributed over several hard drives, this allows storage volumes to be increased by using basic hardware. These storage technologies are always evolving to offer faster access to data such as NoSQL, HDFS from the Hadoop platform, HBase, Cloud Computing, etc.

6. MapReduce

6.1 Why MapReduce?

Traditional business systems normally have a centralized server to store and process data. The traditional model is certainly not suited to handling large volumes of scalable data and cannot be handled by standard database servers. In addition, the centralized system creates too much bottleneck when processing multiple files simultaneously. Google solved this bottleneck issue using MapReduce template.

6.2 MapReduce model definition

It was designed in the 2000s by Google engineers. It is a programming model designed to process several terabytes of data on thousands of computing nodes in a [26] cluster. MapReduce can process terabytes and petabytes of data faster and more efficiently. Therefore, its popularity has grown rapidly for various brands of companies in many fields. It provides a highly efficient platform for parallel execution of applications, allocation of data in distributed database systems, and fault tolerant network communications [27]. The main goal of MapReduce is to facilitate data parallelization, distribution, and load balancing in a simple [26] library.

6.3 The MapReduce model architecture

Google created MapReduce to process large quantities unstructured or semi-structured data, such as documents and logs of requests for web pages, on large clusters of nodes. It produced different types of data, such as inverted indices or

¹ The four sources of big data, <https://www.communication-web.net/2016/03/07/les-4-sources-du-big-data/>

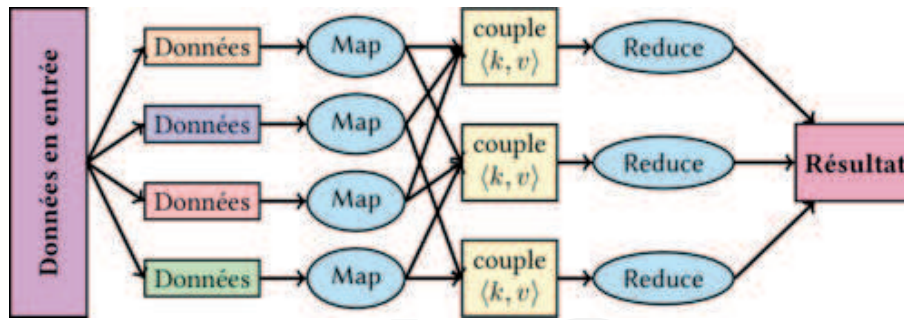


Figure 4.
An example of data flow in the MapReduce big data architecture [29].

URL access frequencies [28]. The MapReduce has three main parts, including the Master, the Map and reduce function. An example of this data flow is shown in **Figure 4**.

The Master is responsible for the management of the Map and Reduce functions and the provision of data and procedures, he organizes communication between mappers and reducers. The map function applies to each input record and produces a list of intermediate records. The Collapse function (also known as Reducer) is applied to each group of intermediate records with the same key and generates a value. Therefore, the MapReduce process includes the following steps:

- The input data are divided into records.
- Map functions process this data and produce key/value pairs for each record.
- All key/value pairs resulting from the Map function are merged together and grouped by a key, then they are sorted.
- The intermediate results are passed to the Reduce function, which will produce the final result [30].

7. Big data processing platforms

7.1 The Hadoop platform for the distributed computing of big data

First of all, Hadoop is a free framework, written in java, created and distributed by the Apache foundation, and intended for the processing of large data (of the order of petabytes and more) as well as for their intensive management. Inspired by several technical publications written by the giant Google, its goal is to provide a distributed, scalable and extensible storage and data processing system. It can handle a large number of data types (including unstructured data). We say that it is organized in a non-relational mode, it is more general than NoSQL, we can for example store data with two types of systems HDFS (Hadoop Distributed File System) and HBase which form a database management system oriented data, columns projected for servers distributed in clusters [31].

Hadoop parallelizes the processing of data across many nodes that are part of a cluster of computers, which speeds up calculations and hides the latency of input and output operations. Hadoop contains a reliable distributed file system that ensures fault tolerance through data replication.

7.2 The Spark platform for the distributed computing of big data

7.2.1 Motivation of Spark

Since its inception, Hadoop has become an important technology for Big Data. One of the main reasons for this success is its ability to manage huge amounts of data regardless of their type (structured, semi-structured, unstructured). However, users have been consistently complaining about the high latency issue with Hadoop MapReduce stating that the batch response to all of these real-time applications is very painful when it comes to processing and analysis data.

7.2.2 History of Spark

Spark is a high-speed compute cluster developed by contributions from nearly 250 developers from 50 AMPLab companies at UC Berkeley, to make data analysis faster and easier to write and thus run. Spark started in 2009 as a research project in the Berkeley Lab RAD, which would later become AMPLab. Researchers in the lab had previously worked on Hadoop MapReduce, and observed that MapReduce was ineffective for iterative and interactive computing jobs. So from the start Spark was designed to be fast for interactive queries and iterative algorithms, bringing ideas like in-memory storage support and efficient fault recovery. Research papers have been published about Spark at academic conferences and shortly after its inception in 2009 it was already 10–100 times faster than MapReduce for some jobs. Some of the early Spark users were other groups in UC Berkeley, including researchers, such as the Millennium Mobile Project, which used Spark to monitor and forecast traffic jams in San Bay. Francisco Machine Learning. In a very short time, however, many external organizations have started using Spark.

In 2011, AMPLab started developing high-level components on Spark, such as Shark and Spark streaming. These and other components are sometimes referred to as Berkeley Data Analytics Stack (ODB). The Spark was open source in March 2010, and it was transferred to the Apache Software Foundation on June 2013, where it is now a high level [32] project.

7.2.3 Definition

Apache Spark is an open source processing framework, it is built around speed, ease of use and the ability to handle large data sets, which are of diverse nature (text data, graph data, etc.), Spark extends the MapReduce model to efficiently support multiple types of computations, including iterative processing, interactive queries, and flow processing (**Figure 5**) [32].²

7.2.4 Advantages of Spark over Hadoop MapReduce

Spark is a strong framework for future large data applications that may require low latency queries, iterative computing, and real-time processing. The Spark has many advantages over the Hadoop MapReduce Framework among them we find [32, 33]:

7.2.4.1 Speed

Spark is an open source compute environment similar to Hadoop, but it has some useful differences that make it superior in some workloads, it allows loading the

² <https://meritis.fr/bigdata/larchitecture-framework-spark/>

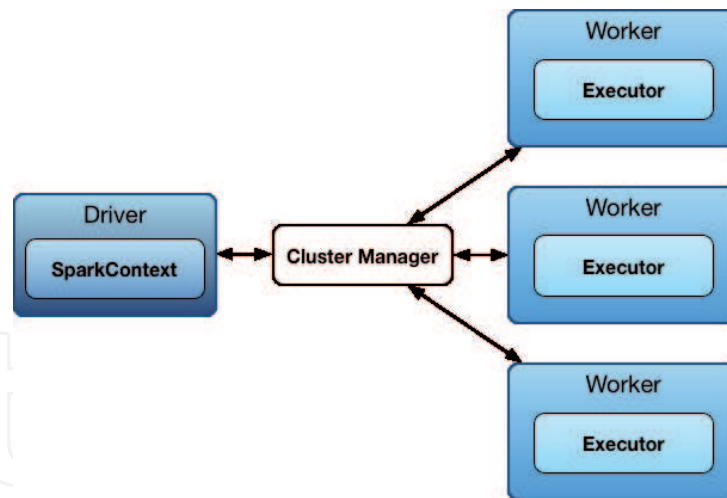


Figure 5.
Spark architecture [32].

dataset into distributed memory to optimize iterative workload and queries. Spark can run jobs 10 to 100 times faster than Hadoop MapReduce simply by reducing the number of reads and writes to disk.

7.2.4.2 Iterative processing

There are many algorithms which apply the same function to several steps. Like learning algorithms, Hadoop MapReduce is based on an acyclic data flow model, that is, the output of a previous MapReduce job is the input of the next MapReduce job. In this case we waste a lot of time in the I/O operation, so in Hadoop MapReduce between two MapReduce operations, there is a synchronization barrier and we need to keep the data on disk every time [33].

But with Spark, the concept of RDD (Resilient Distributed Datasets) allows data to be saved to memory and preserve disk only for result operations. So it does not have a whole synchronization barrier that could possibly slow down the process. So Spark allows to reduce the number of read/write on the disk.

7.2.4.3 Interactive queries

For processing in interactive data extraction algorithms where a user needs to run multiple queries on the same subset of data, Hadoop loads the same data multiple times from disk depending on the number of queries.

But Spark loads the data only once, it stores that data in distributed memory, then it does the proper processing. For processing in interactive data extraction algorithms where a user needs to run multiple queries on the same subset of data.

7.2.4.4 Richer

Spark provides concise and consistent APIs to Scala, Java and Python and Supports multiple functions (actions and transformations), unlike Hadoop, there are only two Map and Reduce functions.

7.2.4.5 Ease of use

Spark lets you quickly write applications in Java, Scala, or Python with simple, readable instructions.

7.2.4.6 General

On the general side, Spark is designed to cover a wide range of workloads that previously require separate distributed systems, including real-time processing applications, iterative algorithms, interactive queries, and streaming. By supporting these workloads in the same engine, Spark makes it easy and inexpensive to combine the different types of processing, which is often required in production data analysis pipelines.

7.2.4.7 Spark's real-time streaming method to process streams

In case of Hadoop MapReduce, it is just possible to process a flow of stored data, but with Apache Spark, it is thus possible to modify the data in real time thanks to Spark streaming [32].

7.2.4.8 Graphics processing

Developers can now as well make use of Apache Spark for graphics processing which maps relationships in data between various entities such as people and objects [32].

7.2.4.9 Learning algorithms

Spark comes with a learning library called MLlib, it provides several types of learning algorithms, including classification, regression, grouping and collaborative filtering, as well as supporting features like evaluation of the template and data import [32]. But in Hadoop you have to integrate a learning library called Mahout.

7.2.4.10 Quick management of structured data

Spark SQL is Spark's module for working with structured data, it allows querying data structured as a Distributed Data Set (RDD) in Spark, with built-in APIs in Python, Scala, and Java.³

7.2.4.11 Storage general

Spark uses the HDFS file system for data storage. It also works with any Hadoop compatible data source, including, HBase, Cassandra, etc.

7.2.4.12 Interactive

Offers an interactive console for Scala and Python. This is not yet available in Java.

7.2.5 Deployment

Executing heavy processing on a cluster, controlling the slave nodes, distributing the tasks for them fairly, and arbitrating the amount of CPU and memory that will be allocated to each process, this is the role of a cluster manager. Spark currently offers three solutions for this: Spark standalone, YARN and Mesos. Comes with Spark,

³ Spark Programming Guide-Spark 1.2.0 Documentation. [Online]. Available: <http://spark.apache.org/docs/1.2.0/programming-guide.html>

Spark Standalone is the easiest way to set up. This cluster manager relies on Akka for exchanges and on Zookeeper to guarantee the high availability of the master node. It has a console to supervise processing, and a mechanism to collect logs from slaves.

Alternatively, YARN the Hadoop cluster manager, Spark can run on it, and alongside Hadoop jobs. Finally, more sophisticated and more general, Mesos allows you to configure more finely the allocation of resources (memory, CPU) to different applications.

7.2.6 Components of Spark

Because Spark's core engine is both fast and versatile, it powers multiple specialized high-level components for various workloads, such as SQL or machine learning. These components allow you to combine them like libraries in a software project. Spark Core: Contains the basic functionality of Spark, including components for job scheduling, memory management, disaster recovery, interaction with storage systems, and more. Spark Core is also the API that defines Elastic Distributed Datasets (RDDs), which are the main programming abstractions in Spark. RDDs represent a collection of objects distributed over several compute nodes that can be manipulated in parallel. Spark Core offers many APIs for building and manipulating these collections.

Other than Spark Core API, there are additional libraries that are part of the Spark ecosystem and provide additional capabilities in big data analysis 6. These libraries are: Spark streaming, Spark SQL, Spark MLlib, Spark GraphX (**Figure 6**) [32].

7.2.7 RDD dataset resilient distributed

7.2.7.1 Definition

An RDD is a collection of objects partitioned across a set of machines, allowing programmers to perform in-memory calculations on large clusters in a way that provides fault tolerance.⁴

7.2.7.2 Characteristics

1. RDD achieves fault tolerance through a notion of lineage: if a partition of an RDD is lost, the RDD has enough information to simply rebuild that partition. This removes the need for replication to achieve fault tolerance.
2. There are two possibilities to create an RDD either to reference external data or to parallelize an existing collection. Spark allows you to create an RDD from any data source accepted by Hadoop (local file, HDFS, HBase, etc.).
3. You can modify an RDD with a transformation, but the transformation returns you a new RDD while the original RDD remains the same.
4. RDD supports two types of operations Transformations and Actions:

Transformation: Transformations do not return a single value, they return a new RDD. Nothing is evaluated when you call a transform function. The evaluation

⁴ Spark Programming Guide - Spark 1.2.0 Documentation. [Online]. Available: <http://spark.apache.org/docs/1.2.0/programming-guide.html>.

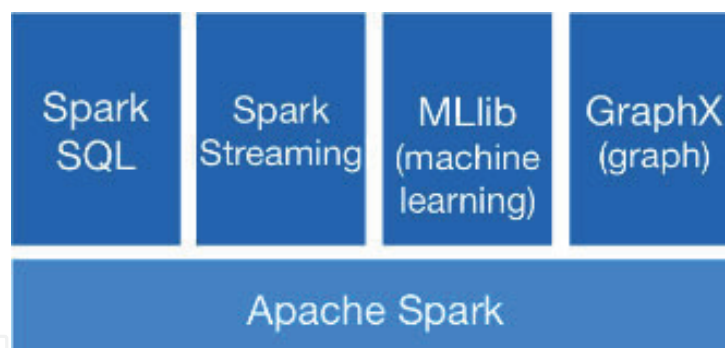


Figure 6.
Spark components [32].

of transformations is lazy, operations are performed only when a result must be returned. **Action:** an operation that evaluates and returns a new value. When an action function is called on an RDD object, all data processing requests were computed at that time and the result value is returned [33].

8. Storage

When processing a large amount of data, input data and results should be stored. Additionally, the performance of data intensive applications typically depends on the hardware and software infrastructure used for storage.

8.1 Classical storage

In this type of storage, the primary and easiest way to store data is a simple hard drive attached directly to the node. This type of storage system is sometimes referred to as a direct storage (DAS). On these disks, data is stored using a classic hierarchical file system like ext3 or ReiserFS. These file systems are typically implemented by an operating system driver as a sensitive part for security, performance, and reliability. This type of storage allows for fast read and write operations since everything is done locally. It is also simple to use as it is used with any operating system. However, there is no easy way to exchange data between multiple nodes.

8.2 Centralized network storage

A second way to store data is centralized network storage, usually referred to as networkAttached Storage (NAS). In this case, a node has one or more disk connected and allows other nodes to read and write files through a standard interface and serve them through the network. Network File System (NFS) is primarily a protocol for accessing files over the network. While the server is free to implement any means of accessing the actual data to be provided over the network, most implementations simply depend on whether the data is directly accessible on the server. One of the main advantages of this type of architecture is the ease of sharing data between multiple compute nodes. Since the data is stored on a server, it is easily maintained.

8.3 Parallel and distributed storage

In order to overcome the limitations of centralized network storage, data can be distributed across multiple storage nodes. Using multiple nodes allows access to multiple files at the same time without conflict. It also allows better throughput

to play the same file when there are replicas on multiple nodes. A distributed file system is usually designed to be better than centralized storing. Additionally, in theory, distributed storage systems can avoid the single point of the fault tolerance problem.

The distributed storage system often has a single entry point node that receives all requests to read or write data. As its role is central and critical, its work should be kept to a minimum. This master node generally only allows global collaboration among all the nodes involved in the storage system and can store metadata (file name, file size, access attributes, ...). Thus, when a request to read or write a file is received, the client is redirected to another node which will actually process his request. However, while the metadata can be stored on the master node, the actual data is still stored on other nodes and can be reproduced. The downside of distributed storage is that for the best performance, the application should consider locality. This is because even though it was thought that the default behavior of the storage system might be quite good, it is usually better to read or write data from the node closest to the system storage from a node with a high network cost.

An example of this storage system is the Hadoop Distributed File System (HDFS) and Tachyon.

8.3.1 HDFS architecture

HDFS has a master/slave architecture. An HDFS cluster consists of a single master called NameNode which manages the namespace of the file system and regulates access to files by clients (open, close, rename, etc.), as well as a set of DataNodes to manage the actual data storage (**Figure 7**) [30].

8.4 Definition of Tachyon

Tachyon is a memory-centric, distributed storage system that allows users to share data across platforms and perform read/write actions at memory speed across cluster processing platforms. It also achieves a write rate of 110x more than in HDFS [34]. To ensure fault tolerance, the lost output is recovered by rerunning the operations that created the output, called lineage [34]. Thus, the Tachyon lineage option is seen as a major challenge in Tachyon, and the lineage layer provides high throughput I/O and follows the job sequence and data lineage in the storage layer.

8.4.1 Tachyon architecture

Indeed, Tachyon uses a standard master-slave architecture similar to HDFS (see **Figure 8**)⁵, this architecture is called master-worker.

The master manages the metadata and contains a workflow manager, the latter interacts with a cluster resource manager to allocate resources and recalculate. Whereas, workers manage local resources and report status to the master, and each worker uses a RAM disk to store memory-mapped files.

8.4.2 The components of Tachyon

Tachyon's design uses a single master and multiple workers. Tachyon can be divided into three components, the master, the workers and the customers. The master and workers together constitute the Tachyon servers, which are the components that a system administrator would maintain and manage. Customers are

⁵ <https://www.slideshare.net/DavidGroozman/tachyon-meetup-slides>

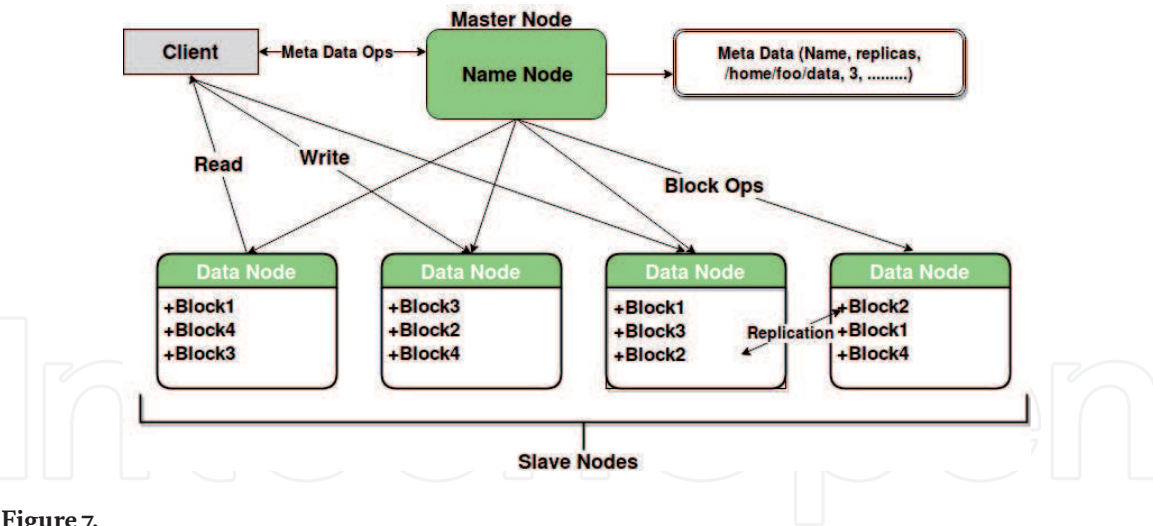


Figure 7.
HDFS architecture [30].

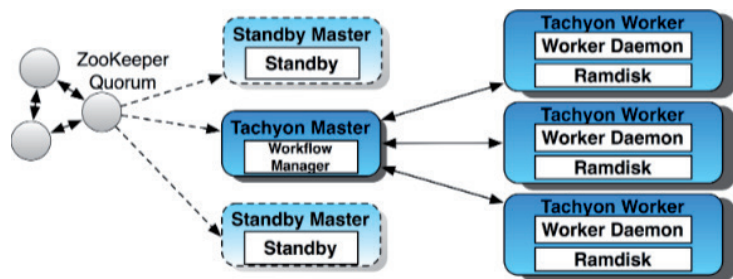


Figure 8.
The Tachyon architecture [34].

typically applications, such as Spark or MapReduce, or Tachyon command line users. So Tachyon users usually only need to interact with the client part of Tachyon [34].

- a. **Master:** Tachyon can be deployed in one of two main modes, a single master or several masters. The master is primarily responsible for managing the overall metadata of the system, for example, the file system tree. Clients can interact with the master to read or modify this metadata. In addition, all workers periodically poll the master to maintain their participation in the cluster. The master does not initiate communication with the components; it only interacts with components responding to requests.
- b. **Workers:** Tachyon workers are responsible for managing the local resources allocated to Tachyon. These resources could be local memory, SSD or hard drive and they are user configurable. Tachyon workers store data as blocks, and respond to customer requests to read or write data by reading or creating new blocks. However, the worker is only responsible for the data in these blocks.
- c. **Customer:** The Tachyon client provides users with a gateway to interact with Tachyon servers. It initiates communication with the master to carry out metadata operations and with workers for reading and writing data.

9. Conclusion

In this chapter, we presented the domain of content based image retrieval system for large scale images using parallel platforms, we covered the basic concepts of

content based image retrieval system. We introduced content based image retrieval system for large scale images (Big data) databases and its problems, the definition of Big Data and its platforms, these later allow us to go beyond classic tools, which fail to process these huge volumes of data, and move towards modern tools, which face to the large data size. Some of the tools we have introduced in this chapter include storage platforms like HDFS and Tachyon and processing platforms like Hadoop MapReduce and Spark.

IntechOpen

IntechOpen


Author details

Saliha Mezzoudj

Applied Mathematics Laboratory, Department of Computer Science, University of ALGER 1, Algeria

*Address all correspondence to: saliha.mezzoudj@yahoo.com

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Nicu Sebe, Ira Cohen, Ashutosh Garg, and Thomas S Huang. Machine learning in computer vision, volume 29. Springer Science & Business Media, 2005.
- [2] Ricardo da Silva Torres and Alexandre X Falcao. Content-based image retrieval: Theory and applications. *RITA*, 13(2):161-185, 2006.
- [3] Saliha Mezzoudj, Rachid Seghir, Yassmina Saadna, et al. A parallel content-based image retrieval system using spark and tachyon frameworks. *Journal of King Saud University-Computer and Information Sciences*, 2019.
- [4] Chunhao Gu and Yang Gao. A content-based image retrieval system based on hadoop and lucene. In *2012 Second International Conference on Cloud and Green Computing (CGC)*, Pages 684-687. IEEE, 2012.
- [5] Siyao Fu, Haibo He, and Zeng-Guang Hou. Learning race from face: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12): 2483-2509, 2014.
- [6] Mezzoudj Saliha, Behloul Ali, and Seghir Rachid. Towards large-scale face-based race classification on spark framework. *Multimedia Tools and Applications*, 78(18):26729-26746, 2019.
- [7] Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*, volume 1. John Wiley & Sons, 2002.
- [8] Khushboo Khurana and Reetu Awasthi. Techniques for object recognition in images and multi-object detection. *International journal of advanced research in Computer Engineering & Technology (IJARCET)*, 2(4):pp-1383, 2013.
- [9] Majid Mirmehdi. *Handbook of Texture Analysis*. Imperial College Press, 2008.
- [10] Francesco Bianconi and Antonio Fernández. Evaluation of the effects of gab or filter parameters on texture classification. *Pattern Recognition*, 40(12):3325-3335, 2007.
- [11] Robert M Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786-804, 1979.
- [12] Dong-Chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):509-512, 1990.
- [13] Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Efficient and robust retrieval by shape content through curvature scale space. In *Image Databases and Multi-Media Search*, Pages 51-58. World Scientific, 1997.
- [14] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91-110, 2004.
- [15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, Pages 404-417. Springer, 2006.
- [16] Wong, Ooi Qun and Rajendran, Parvathy. Image Segmentation using Modified Region-Based Active Contour Model. *J. Eng. Appl. Sci.* 2019;14:5710-5718
- [17] Maneela Jain and Pushpendra Singh Tomar. Review of image classification methods and techniques. *International journal of engineering research and technology*, 2(8), 2013.

- [18] W Chao. Machine Learning Tutorial. *National Taiwan University*, 2011.
- [19] Jean Ponce, Tamara L Berg, Mark Everingham, David A Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C Russell, Antonio Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pages 29-48. Springer, 2006.
- [20] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 Object Category Dataset. 2007.
- [21] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303-338, 2010.
- [22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211-252, 2015.
- [23] Ahmed Oussous, Fatima-Zahra Benjelloun, Ayoub Ait Lahcen, and Samir Belfkih. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4):431-448, 2018.
- [24] Raghavendra Kune, Pramod Kumar Konugurthi, Arun Agarwal, Raghavendra Rao Chillarige, and Rajkumar Buyya. The anatomy of big data computing. *Software: Practice and Experience*, 46(1):79-105, 2016.
- [25] Desamparados Blazquez and Josep Domenech. Big data sources and methods for social and economic analyses. *Technological Forecasting and Social Change*, 130:99-113, 2018.
- [26] Alberto Fernández, Sara del Río, Abdullah Bawakid, and Francisco Herrera. Fuzzy rule based classification systems for big data with mapreduce: Granularity analysis. *Advances in Data Analysis and Classification*, 11(4): 711-730, 2017.
- [27] Lu Lu, Xuanhua Shi, Hai Jin, Qiuyue Wang, Daxing Yuan, and Song Wu. Morpho: A decoupled mapreduce framework for elastic cloud computing. *Future Generation Computer Systems*, 36:80-90, 2014.
- [28] Seyed Nima Khezer and Nima Jafari Navimipour. Mapreduce and its applications, challenges, and architecture: A comprehensive review and directions for future research. *Journal of Grid Computing*, 15(3): 295-321, 2017.
- [29] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107-113, 2008.
- [30] Vignesh Prajapati. *Big Data Analytics with R and Hadoop*. Packt Publishing Ltd, 2013.
- [31] Aditya B Patel, Manashvi Birla, and Ushma Nair. Addressing big data problem using hadoop and map reduce. In *2012 Nirma University International Conference on Engineering (NUICONE)*, Pages 1-5. IEEE, 2012.
- [32] Holden Karau, Andy Konwinski, Patrick Wendell, and Matei Zaharia. *Learning spark: lightning-fast big data analysis*. “O’Reilly Media, Inc.”, 2015.
- [33] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.
- [34] Haoyuan Li, Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Tachyon: Reliable, memory speed storage for cluster computing frameworks. In *Proceedings of the ACM Symposium on Cloud Computing*, Pages 1-15. ACM, 2014.