MapReduce and Spark: Big Data Processing

AMW School

Yudith Cardinale, PhD

Profesora Titular, Universidad Internacional de Valencia, España

Profesora Titular, Universidad Simón Bolívar, Venezuela

May 2023
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
The Big Data Boom

Data Explosion

- The production of data by users on the Web (blogs, social networks, etc.) and the sharing of ubiquitous information (sensors and mobile devices, cameras, microphones, photographs, etc.), drastically increases the amount of data that can be processed and the perspectives of interpretation.

- The growth in the amount of data available is skyrocketing in an unprecedented way:
  - In 2017 the IDC (International Data Corporation) predicted that by 2025 it would be reached 163 ZB (trillions of GB) of data.
  - In 2018, the IDC adjusted the prediction: by 2025 it would be reached 175 ZB of data.
The Big Data Boom: Data Explosion

Prediction in 2017:

Figure 2. Annual Size of the Global Datasphere

Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017
The Big Data Boom: Data Explosion

Prediction in 2018:

Figure 1 - Annual Size of the Global DataspHERE

Annual Size of the Global DataspHERE

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Zetabytes</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>80</td>
<td>85</td>
<td>90</td>
</tr>
</tbody>
</table>

2025: 175 ZB
### Data Explosion (cont.)

- According to IDC, only in 2020 64.2 ZB of data were created (product of the COVID-19 situation).
- The amount of digital data created over the next five years will be greater than twice the amount of data created since the advent of digital storage. The question is: **How much of them should be stored?**
- Only 2% of the data created in 2020 were kept and stored.
- In 2020, the IDC announced:
  - A growth of 23% per year for the period 2020-2025.
  - That **IoT data** (not including video surveillance cameras) are the **fastest growing data segment**, followed by **social networks**.
- **IDC’s Predictions in 2022:**
  
Data Explosion (cont.)

- What is generating such as data explosion? ⇒ Innovation
  - Business model transformation
    - Globalization and connectivity
    - From *product-oriented* to *service-oriented*, personalization of services
  - New data sources (social media, mobile devices, sensor networks, ...) ⇒ Social and cultural evolution
    - Every day we create 2.5 quintillion bytes of data; more than 90% of the existing data in the world today, were created only in the last two years!
  - Advanced technologies
    - Mobile devices
    - High volume of data processing networks
    - The "commoditization" of hardware
    - Cloud Computing, IoT, Data Science
    - Security, virtualization, *open-source software*, ...
"Big Data can be defined as volumes of data available in varying degrees of complexity, generated at different velocities and varying degrees of ambiguity, that cannot be processed using traditional technologies, processing methods, algorithms, or any commercial off-the-shelf solutions. Data defined as Big Data includes machine-generated data from sensor networks, nuclear plants, X-ray and scanning devices, and airplane engines, and consumer-driven data from social media. Big Data producers that exist within organizations include legal, sales, marketing, procurement, finance, and human resources departments"[1].

"Big Data refers to datasets and flows large enough that has outpaced our capability to store, process, analyze, and understand"[2].
The Big Data Boom

Definitions

- "Big Data are high–volume, high–velocity, and high–variety information assets that require new forms of processing to enable enhanced decision making, insight discovery, and process optimization" (Gartner 2012).
- "Building new analytic applications based on new types of data, in order to better serve your customers and drive a better competitive advantage" (David McJannet, Hortonworks).
- "Big Data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it..."

Dan Ariely
Professor of Psychologie
Duke University – NC - USA
The Big Data Boom

Big Data Dimensions

- In 2001, an analyst at the META Group (now Gartner) defined, in a research report, the challenges and opportunities of the data growth from a three-way perspective dimensions: The "3Vs" (Volume, Velocity, and Variety);
- In 2012, IBM added the fourth dimension: Veracity;
- The most important dimension (for the business) was identified more recently: Value ⇒ Data Science, "5Vs";
- "7Vs" version: "5Vs" + Variability, Visualization
- "10Vs" version: "5Vs" + Variability, Validity, Venue, Vocabulary, Vagueness
- Are you thinking in another V? For sure it will fix ... added it here

....
The Big Data Boom: 5 Vs

- **Velocity**
  - Batch
  - Real Time
  - Processing
  - Data streams

- **Volume**
  - Terabytes (TB)
  - Registering/Archives
  - Transactions
  - Tables, files

- **Variety**
  - **Sources**: internal, external, client behavior
  - **Content**: structured, semi-structured, non-structured, multi-dimensional

- **Value**
  - Statistics
  - Events
  - Correlations
  - Hypothesis

- **Veracity**
  - Confidence
  - Authenticate
  - Source, reputation
  - Availability
  - Responsible
The Big Data Boom: More Vs

Figure: https://www.learnbigdatatools.com/learn-big-data-databases/
The Big Data Boom

Challenges

- These data are presented in formats that are difficult to process with traditional DBMS:
  - They are not organised in tables and the structure can vary (from structured to unstructured);
  - They are generated in real time in continuous flows;
  - They come from different sources (mobile devices, sensors, PCs, Laptops, objects, social networks, ...) in a disorderly and unpredictable way;

- **Capture, storage, search, sharing, analysis, and data visualisation must be redefined:**
  - Collect large volumes of data, varied, to find new ideas;
  - Capture quickly the created data;
  - Store the data needed;
  - Process, analyse, and use these data.
The Big Data Boom: Challenges

Structured data
- Databases

Semi-structured data
- XML / JSON data
- Email
- Web pages

Unstructured data
- Audio
- Video
- Image data
- Natural language
- Documents
The Big Data Boom: Big Data Models

- Relational

<table>
<thead>
<tr>
<th>idTweet</th>
<th>contenido</th>
<th>fecha</th>
<th>numCompartidos</th>
<th>idAutor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>et temp...</td>
<td>07/08/2018</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Semp...</td>
<td>07/08/2018</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>est qua...</td>
<td>08/08/2018</td>
<td>178</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Phare...</td>
<td>09/08/2018</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>magna...</td>
<td>10/08/2018</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>idUsuario</th>
<th>nombreUsuario</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>flarcher0yo</td>
<td><a href="mailto:flarcher0@nytimes.com">flarcher0@nytimes.com</a></td>
</tr>
<tr>
<td>2</td>
<td>cbatchel1972</td>
<td><a href="mailto:cbatchelour1@exblog.jp">cbatchelour1@exblog.jp</a></td>
</tr>
<tr>
<td>3</td>
<td>serPaiITwo</td>
<td><a href="mailto:spail2@csmonitor.com">spail2@csmonitor.com</a></td>
</tr>
</tbody>
</table>
NoSQL: <key,value> oriented

<table>
<thead>
<tr>
<th>Clave de partición</th>
<th>Clave de ordenación</th>
<th>Atributos</th>
</tr>
</thead>
<tbody>
<tr>
<td>idUsuario 1</td>
<td>idTweet 1</td>
<td>contenido, et, temp...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fecha: 07/08/2018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numCompartidos: 20</td>
</tr>
<tr>
<td>idUsuario 2</td>
<td>idTweet 2</td>
<td>contenido, Sem...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fecha: 07/08/2018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numCompartidos: 12</td>
</tr>
<tr>
<td>idUsuario 1</td>
<td>idTweet 3</td>
<td>contenido, est, qua...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fecha: 08/08/2018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numCompartidos: 178</td>
</tr>
<tr>
<td>idUsuario 1</td>
<td>idTweet 4</td>
<td>contenido, Phare...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fecha: 09/08/2018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numCompartidos: 6</td>
</tr>
<tr>
<td>idUsuario 3</td>
<td>idTweet 5</td>
<td>contenido, magna...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fecha: 10/08/2018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numCompartidos: 2</td>
</tr>
</tbody>
</table>
**Big Data Models**

- **NoSQL: column oriented**

<table>
<thead>
<tr>
<th>Clave</th>
<th>Datos usuario</th>
<th>Datos Tweet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>idUsuario nombreUsuario email</td>
<td>contenido fecha numCompartidos</td>
</tr>
<tr>
<td>1</td>
<td>flarche0yo <a href="mailto:flarche0@nytimes.com">flarche0@nytimes.com</a></td>
<td>contenido1 et temp... fecha1 07/08/2018 numCompartidos1 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contenido3 est qua... fecha3 08/08/2018 numCompartidos3 178</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contenido4 Phare... fecha4 09/08/2018 numCompartidos4 6</td>
</tr>
<tr>
<td>2</td>
<td>cbatchel1972 <a href="mailto:cbatchelour1@exblog.jp">cbatchelour1@exblog.jp</a></td>
<td>contenido2 Semp... fecha2 07/08/2018 numCompartidos2 12</td>
</tr>
<tr>
<td>3</td>
<td>serPailTwo <a href="mailto:spail2@csmonitor.com">spail2@csmonitor.com</a></td>
<td>contenido5 magna... fecha2 10/08/2018 numCompartidos2 2</td>
</tr>
</tbody>
</table>
NoSQL: document oriented

```
{ id: 1, 
nombreUsuario: flarcher0yo, 
email: flarcher0@nytimes.com, 
tweets: [ 
  { 
    contenido: et temp..., 
    fecha: 07/08/2018, 
    numCompartidos: 20 
  } 
} 
```

```
{ id: 2, 
nombreUsuario: cbatchel1972, 
email: cbatchelour1@exblog.jp, 
tweets: [ 
  { 
    contenido: Semp..., 
    fecha: 07/08/2018, 
    numCompartidos: 12 
  } 
} 
```

```
{ id: 3, 
nombreUsuario: serPaliTwo, 
email: spail2@csmonitor.com, 
tweets: [ 
  { 
    contenido: magna..., 
    fecha: 10/08/2018, 
    numCompartidos: 2 
  } 
} 
```
NoSQL: graph oriented
The Big Data Boom: Processing Models

Processing Models

- Batch
- Streaming
- Transactional
- Lambda architecture: Streaming + Batch
  - Speed Layer: real time processing
  - Batch Layer: batch processing
  - Users can consult the speed views + batch
  - Drawback: manage both systems
- Kappa architecture: Streaming + Batch
  - Real time data: streaming
  - Historical data: big streaming
  - Users can consult the historical streaming views + real time
  - Drawback: If there is a fault, the historical processing should be re-executed – Some times it can be avoided
To obtain value from data it is necessary to process them, to analyze them;

*Data Science* is a multidisciplinary field whose goal is extract value from data;

*Data Science* is a process that transforms raw data into meanings, into interpreted information;

This process is a *pipeline* which includes data engineering, *machine learning*, and operations on the results.
Data Science: Big Data Analytics

Data engineering:
- Raw data
- Data wrangling
- Data cleansing
- Data preparation

Machine learning:
- Model learning
- Model validation

Operations:
- Model deployment
- Data visualization
Big Data Analytics

- Descriptive Analytics
- Diagnostic Analytics
- Predictive Analytics
- Prescriptive Analytics

Value

Information

Hindsight

Insight

Optimization

Foresight

Difficulty

http://www.gartner.com/it-glossary/predictive-analytics
Data engineering

- **Data ingestion:**
  - Identify and collect raw data;
  - Integrate data from different sources;
  - Represent them in a common and consistent format;

- **Data cleaning:**
  - Identify wrong values, inconsistencies, or insufficient parameters;
  - Data correction can be done manually or automatically; if the data cannot be repaired, it is deleted;
  - Identify outliers through statistical analysis, for example.

- **Data pre-processing or preparation:**
  - Although the data are "clean", they may require additional preparation before moving on to the *machine learning* phase;
  - Data normalization;
  - Convert categorical data to numeric values.
**Machine Learning**

- **Learning model:**
  
  ![Machine learning approaches](image)

  - Supervised learning:
    - Backprop neural networks
    - Decision tree learning
    - Bayesian statistics
    - Support vector machines
    - Random decision forests
  
  - Unsupervised learning:
    - K-means clustering
    - Principal component analysis
    - Generative adversarial network
    - Adaptive resonance theory
    - Hierarchical clustering
  
  - Reinforcement learning:
    - Q-learning
    - Temporal difference learning
    - SARSA
    - Monte Carlo methods
    - Inverse reinforcement learning

- **Model validation:**
  
  ![Model validation](image)
Operations on the results

- **Visualization:**
  - Graphs;
  - Reports.

- **Deployment of the model:**
  - Predictions;
  - Recommendations.
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Suppose we have 10,000 employees, whose job is to collect census forms and determine how many people live in each city;
How would you organize these tasks?
Suppose workers go on vacation, get sick, work at different rhythms;
Suppose some forms are filled out incorrectly and require corrections or need to be discarded;
What happens if the supervisor gets sick?
How big should the work piles be?
How can progress be monitored?
...

Big Data and MapReduce

A little bit of introspection

- What is the main challenge?
  - Are the individual tasks complicated?
  - If not, what is it that makes it complicated?
- How resistant is our solution?
- How well balanced is the work among the employees?
  - What factors affect this?
- How general is the set of techniques used?

We do not want to deal with all of this

- What about a system that would take care of all these details?
- Ideally, we would just tell the system what needs to be done
  ⇒ This is what MapReduce intends.
What we need

- Scalability of big amount of data: *scale out* instead of *scale up*
  - 1000’s of machines
  - 10,000’s of disks
  - … more machines, more disks? ⇒ *scale out* is easy.

- Commodity hardware: low costs machines and networks:
  - unreliable: large number of commodity server clusters
  - high latency

- Easy programming

- Fault tolerance
  - Cheap nodes fail, especially if there are a lot of them.
  - Mean Time to Failure (MTBF) for 1 node = 3 years
  - MTBF for 1000 nodes = 1 day
  - MTBF for 10000 nodes, 10 failures per day
  - Solution: implement fault tolerance in the system
What is MapReduce

There are two types of workers:

Those that take input data items and produce output items for the stacks, called map: takes pairs (item_key, value), produces one or more pairs (stack_key, value’)

Those that take the stacks and aggregate the results to produce outputs per stack, called reduce: take(stack_key, set of value’), produce one or more output results – typically (stack_key, agg_value)

It is a data-parallel programming model designed for scalability and fault tolerance in large commodity hardware systems

Combine Map and Reduce operations with an associated implementation

Initially proposed by Google (2004):

- Used in multiple operations
- Process several PB of data per day
Big Data and MapReduce

Big ideas behind MapReduce

- **Move processing where the data are:**
  - Commodity network = low bandwidth
  - Take advantage of data locality and avoid transferring large datasets across the network ⇒ Bring computation to data

- **Hiding system level details from application developers**
  - Programming distributed systems is difficult
  - Developers want to focus on their problems instead of dealing with distributed programming issues
  - **Solution**: Users write data-parallel functions **map** and **reduce** and the system handles work distribution and failures
Big Data and MapReduce: A simple example

Goal:

- Given a set of documents, count the frequency of each word:
  - **Input**: Key-value pairs (document: lineNumber, text)
  - **Output**: Key-value pairs (word, #occurrences)
  - What should be the intermediate key-value pairs?

```java
map(String key, String value) {
    // key: document name, line no
    // value: contents of line
    for each word w in value:
        emit(w, "1")
}
```

```java
reduce(String key, Iterator values) {
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += parseInt(v);
    emit(key, result)
}
```
Big Data and MapReduce: A simple example

En un lugar de la Mancha

Más vale la pena en el rostro que la mancha en el corazón

El amor es deseo de belleza
Big Data and MapReduce: A simple example

Entrada
En un lugar de la Mancha

Más vale la pena en el rostro que la mancha en el corazón

El amor es deseo de belleza

Map
map
map
map

Shuffle & Sort
en,1
rostro,1
la,1
mancha,1
en,1

Reduce
reduce
reduce
reduce

Salida
Big Data and MapReduce: A simple example

Entrada
En un lugar de la Mancha
Más vale la pena en el rostro que la mancha en el corazón
El amor es deseo de belleza

Map
map
map
map

Shuffle & Sort

Reduce
reduce
reduce
reduce

Salida

MapReduce and Spark
May 2023
39/98
Big Data and MapReduce: A simple example

Entrada

- En un lugar de la Mancha
- Más vale la pena en el rostro que la mancha en el corazón
- El amor es deseo de belleza

Map

- map
- map
- map

Shuffle & Sort

- amor, 1
- corazón, 1
- deseo, 1
- en, 1, 1, 1
- la, 1, 1, 1
- mancha, 1, 1
- pena, 1
- rostro, 1
- vale, 1

Reduce

- reduce
- reduce

Salida

- amor, 1
- corazón, 1
- deseo, 1
- en, 3
- la, 3
- mancha, 2
- pena, 1
- rostro, 1
- vale, 1
- belleza, 1
- de, 2
- el, 3
- es, 1
- lugar, 1
- más, 1
- que, 1
- un, 1
Big Data and MapReduce: A simple example

Coordinator

Map computation

Reduce computation

partitions

partitions

Data partitions by key

Redistribution by output's key ("shuffle")

Input files:
DFS

Map phase

Intermediate files
(on local disks)

Reduce phase

Output files:
DFS

Yudith Cardinale
MapReduce and Spark
May 2023
Big Data and MapReduce: Who uses it?

Image Credit: blog.galaxy.weblinks.com
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Apache Hadoop: Batch processing

Open-source implementation of MapReduce

- Huge amount of data processing in commodity clusters:
  - Scale: Petabytes of data on thousands of nodes
Apache Hadoop: Batch processing

Top Hadoop Ecosystem Components

- Oozie (Workflow)
- HCatalog (Table & Schema Management)
- Pig (Scripting)
- Hive (SQL Query)
- Mahout (Machine Learning)
- Drill (Interactive Analysis)
- Avro (JSON)
- Thrift (Cross Language Service)
- HBase (Column Store)
- Hadoop Distributed File System (HDFS)
- YARN (Cluster Resource Management)
- Zookeeper (Co-ordination)
- Apache Ambari (Management & Monitoring)
- Scoop (Data Collection)
- Hadoop (Data Collection)
Apache Hadoop: Batch processing
Installation:

- **Simple installation as a Java Application:**
  - Source: https://hadoop.apache.org
  - Preconfigured versions granted by enterprises like Cloudera/HortonWorks (https://www.cloudera.com/products/hdp.html) or mapR (mapR.com/products)

- **Operating modes**
  - Standalone: All components in one node (for tests)
  - Pseudo-distributed: All components in one node simulating distribution
  - Totally distributed: All components distributed in a cluster.

- **Virtualized installation**
  - Operating System in a VM: AlmaLinux, Ubuntu, Centos, ...
  - Hadoop installation: standalone, pseudo-distributed, with containers (docker-compose)
Installation (cont.):

- **Installation in a cluster**:
  - Configuration of hardware, operating system, ports, networks, ...
  - Edge node
  - Cluster configuration
    - Big data framework
    - HDFS configuration

- **Installation in AWS cloud**
  - IAM: account creation and permission
  - S3: massive storage
  - EMR: Elastic MapReduce cluster

- **Google collab**
  - Not needed to install anything locally
  - Notebook web
  - Install hadoop in google collab
  - Import programs and data
Hadoop Filesystems:

- Hadoop provides an **abstract notion of filesystems**
- Some supported filesystems are:
  - Local FS: local disk
  - HDFS: a particular FS of hadoop
  - HFTP: RO access to HDFS on HTTP
  - HSFTP: RO access to HDFS on HTTPS
  - WebHDFS: RW access to HDFS on HTTP
  - S3 (native): Access to native S3
  - S3 (block): Access to S3 in blocks
Apache Hadoop: HDFS

HDFS advantages:
- Distributed file system
- **Bulk persistence**: designed to store huge files in commodity hardware
- 128MB blocks (default)
- Replication on three servers (default): **reliability**
- API: programmatic, CLI (command line interface)

HDFS disadvantages:
- High latency
- Inefficient for small files
- Insertions only at the end of the files
- Single-writer/multiple-readers model
Apache Hadoop: HDFS

Hadoop MapReduce

Dataset

InputFormat (input splits)

InputSplit

InputSplit

InputSplit

InputFormat (recordReader)

MapReduce

Mapper

Mapper

Mapper

Mapper

Mapper

Mapper

Map

Map

Map

Map

Map

Map

InputSplit 1

InputSplit 2

InputSplit 3

HDFS

Bloque 1

Bloque 2

Bloque 3
Apache Hadoop: HDFS
Apache Hadoop: HDFS
Let’s see another material:

- An illustrative example: Calculate max temperature
- Installation of hadoop (standalone and pseudo-distributed – MacOs and Ubuntu)
- Resolve exercises
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Spark: Processing in memory

Batch Processing: MapReduce

MapReduce and Spark

May 2023
Spark: Processing in memory

Characteristics:
- Memory processing
- Efficient in iterative processing
- Languages: Scala, Python and Java
- Extends the MapReduce processing model
- Easy to optimize code: DAG – Code focused on analysts
RDD structure: for batch processing

- Distributed dataset:
  - Partitions = inputSplits
- The RDD dataset can be on disk or in memory
- Partitions can be changed:
  - with "repartition" transformation
- RDD is transformed into new RDDs (transformation) until the result (action) is obtained.
Spark: RDD operations

- Transformations generate new RDD

- Actions generate results
Spark: RDD transformations

Transformaciones sencillas:
- map
- flatMap
- mapPartitions
- filter
- sample
- union
- keyBy
- sortByKey
- pipe

Transformaciones complejas:
- intersection
- distinct
- reduceByKey
- groupByKey
- aggregateByKey
- join
- cartesian
- repartition
- coalesce
- cogroup
Spark: RDD actions

- Acciones:
  - reduce
  - collect
  - count
  - first
  - take
  - takeSample

- Acciones:
  - takeOrdered
  - saveAsTextFile
  - saveAsSequenceFile (java/scala)
  - saveAsObject (java/scala)
  - countByKey
  - Foreach
  - getNumPartitions
Let’s see another material:

- RDD transformations and actions
- Installation of Spark (standalone and pseudo-distributed – MacOs and Ubuntu)
- Resolve exercises
Streaming processing

Data Ingestor:
- Kafka
- Kinesis
- FLUME
- HTTP
- TCP
- ...

Streaming Processing

Batch Processing

DFS storing
Microbatches: RDD are generated in short times with the streaming data

Streaming processing with a native batch processing engine
Spark Streaming vs. Structured Streaming

Data stream → Spark Streaming → batches of input data → Spark Engine → batches of processed data

Data stream as an unbounded table

Unbounded Table

new data in the data stream = new rows appended to a unbounded table
Spark streaming transformations:

- Map, flatMap, filter: a function to each register
- Repartitions, union, join, cogroup: to manage partitions
- count, reduce: produce one result for RDD
- countByValue, reduceByKey: count the occurrence of each value, apply a function per key
- Window: manage static and slide windows (when to generate RDD and process them)
- \{count,reduce\}ByWindow
- reduceByKeyAndWindow, countByValueAndWindow
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Streaming Processing: Spark and Flink

**Spark Vs Flink**

- Spark SQL
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)

**Apache Spark Core**

Batch processing by default

**Difference between Spark & Flink**

Streaming engine by default

[www.apachespark.in](http://www.apachespark.in)
Flink: Streaming Processing

Streaming Processing: Spark and Flink

Data ingestor: Kafka, Kinesis, TCP sockets, ...

Spark

Spark Streaming

Batch_n, Batch_n-1, ... Batch_1

Spark Engine

Output: HDFS, Data Base, dashboards, ...

Transactions
Logs
IOT
Clicks
...

(Real-time) Events

Database, File System, KV-Store

Event-driven Applications

Streaming Pipelines

Stream & Batch Analytics

Resources | Storage
(K8s, Yarn, ...) | (HDFS, S3, NFS, ...)

Application
Event Log
Database, File System, KV-Store
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Case 1: Robots in museums

RUTAS

ROBOTS FOR URBAN TOURISTIC CENTERs AUTONOMOUS AND SEMANTIC WEB BASED

Case 1: Robots in museums

RUTAS

Big Data

Semantic Web

Ontologías

SLAM Ontology

Big Data

Robótica

Yudith Cardinale

MapReduce and Spark

May 2023

75 / 98
Case 1: Robots in museums

Multimodal emotion recognition

- Capture people through the robot’s sensors (camera, sound system)
- Analyse different sources (images: face, posture, gesture; audio: speech; text: translate speech to text);
- Analyse individual sources and apply a fusion method to aggregate individual results.
Case 1: Robots in museums

MULTIMODAL EMOTION RECOGNITION SYSTEM

- Outside-robot approach
  - More complex methods
  - Slower
  - No limited by Robot Hardware
A question for you:

How **MapReduce** or **Spark** can be used in this scenario?

Any idea?
Case 1: Robots in museums

Deep Learning Model 1 → Deep Learning Model n → Modalities fusion method → EmbraceNet+

Reading the latest data → Shared folder

CONTAINER

Input Data processing

SERVER

CONTAINER

Docker

image.jpg file

audio.wav files

Transcriber

text.txt files

jupyter

ROS

SentiLib

Spark NLP
Case 1: Robots in museums

Batch and Streaming Processing: Spark

Using RDD:

Using Data Frame:
Case 1: Robots in museums

Summary:
- We have implemented batch and streaming processing to prepare data for the deep learning models that recognise emotions (individual and fusion method);
- **Streaming processing**: To adapt the robot’s behaviour in real time according to the emotion recognised;
- **Batch processing**: For further analysis; to create datasets for more experiments;
- **Future work**: Implement the whole pipeline (processing and analysis) with Spark and Flink facilities. Extend this streaming processing approach for other tasks of the robot (SLAM, artworks description, HRI, navigation)
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce
3. Apache Hadoop MapReduce: Batch processing
4. Spark: Processing in memory
5. Flink: Streaming Processing
6. Case 1: Robots in museums
7. **Case 2: Smart homes for elder people**
8. Case 3: Twitter credibility measure
Case 2: Smart homes for elder people

Framework to detect Activities Daily Living (ADL) in Smart Homes
### Tableau 110.1 Les 6 items des activités de la vie quotidienne (ADL).

<table>
<thead>
<tr>
<th>Groupe</th>
<th>Niveau d’autonomie</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIR 1</td>
<td>Nécessite une aide en permanence</td>
</tr>
<tr>
<td>GIR 2</td>
<td>Nécessite une prise en charge pour la plupart des activités</td>
</tr>
<tr>
<td>GIR 3</td>
<td>Nécessite une aide pour les soins corporels plusieurs fois par jour</td>
</tr>
<tr>
<td>GIR 4</td>
<td>Nécessite un aide pour la toilette, l’habillage, voire les repas</td>
</tr>
<tr>
<td>GIR 5</td>
<td>Nécessite un aide ponctuelle pour la toilette, la préparation des repas et le ménage</td>
</tr>
<tr>
<td>GIR 6</td>
<td>Est autonome</td>
</tr>
</tbody>
</table>

#### 1. Hygiène corporelle
- Indépendance: 1
- Aide partielle: 0,5
- Dépendance: 0

#### 2. Habillage
- Indépendance pour le choix des vêtements et l’habillage: 1
- Autonomie pour le choix des vêtements et l’habillage, mais a besoin d’aide pour se chaussar: 0,5
- Dépendant: 0

#### 3. Aller aux toilettes
- Indépendance pour aller aux toilettes, se déshabiller et se rhabiller ensuite: 1
- Besoin d’aide pour se déshabiller ou se rhabiller aux toilettes: 0,5
- Ne peut aller aux toilettes seul: 0

#### 4. Transfert
- Indépendance: 1
- A besoin d’aide: 0,5
- Grabataire: 0

#### 5. Continence
- Continent: 1
- Incontinence urinaire ou fécale occasionnelle: 0,5
- Incontinence urinaire ou fécale: 0

#### 6. Repas
- Mange seul: 1
- Aide pour couper la viande ou peler les fruits: 0,5
- Dépendant: 0

**Total des points**
- Meilleur score = 6. Score < 3 = dépendance majeure; score = 0 : dépendance totale pour toutes ces activités.
Case 2: Smart homes for elder people

ICASA Simulator

ICASA

- Activities Daily Living (ADL)
- Independent people
- Non-independent people
Case 2: Smart homes for elder people

Another question for you:

How MapReduce or Spark can be used in this scenario?

Any idea?
Case 2: Smart homes for elder people

Batch and streaming processing: Flume, Flink, Spark

Data generation
Data ingestion
Data processing
Case 2: Smart homes for elder people

Summary:

- We have implemented batch and streaming processing to prepare data for the Analyzer Module that recognises ADL from sensors’ data and identifies people independence;
- **Streaming processing**: To recognise ADL and other events (emergency situations) in real time;
- **Batch processing**: For further analysis; to create datasets for more experiments;
- **Future work**:
  - Implement the whole pipeline (processing and analysis) with Spark and Flink facilities.
  - Recognise more complex activities.
Agenda

1. The Big Data Boom
   - Data Explosion
   - Definitions
   - Big Data Dimensions
   - Challenges
   - Data Science: Big Data Analytics

2. Big Data and MapReduce

3. Apache Hadoop MapReduce: Batch processing

4. Spark: Processing in memory

5. Flink: Streaming Processing

6. Case 1: Robots in museums

7. Case 2: Smart homes for elder people

8. Case 3: Twitter credibility measure
Case 3: Twitter credibility measure

\[ T-CR\text{E}o: \text{credibility model} \]

\[ TextCred(p.text) = w_{\text{SPAM}} \times is\text{SPAM}(p.text) + w_{\text{BadWords}} \times \text{bad\_words}(p.text) + w_{\text{MisspelledWords}} \times \text{misspelling}(p.text) \]

\[ UserCred(p.user) = \text{Verif\_Weight}(p.user) + \text{Creation\_Weight}(p.user) \]

\[ SocialCred(p.user) = \text{FollowersImpact}(p.user) + \text{FFProportion}(p.user) \]

\[ TopicCred(p.text) = 100 \times (1 - \frac{1}{n} \sum_{i=1}^{n} HDS(\text{NMF}(p.text), \text{NMF}(p.text\_hashtag_i))) \]

\[ UserCredBot(p.user) = \begin{cases} 
UserCred(p.user) & \text{if } p.user \text{ is not a bot,} \\
0.85 \times UserCred(p.user) & \text{if } p.user \text{ is a bot and } UserCred(p.user) \text{ is higher than 50,} \\
0.75 \times UserCred(p.user) & \text{if } p.user \text{ is a bot and } UserCred(p.user) \text{ is between 35 and 50,} \\
0 & \text{otherwise.} 
\end{cases} \]

\[ \text{Cred}(p) = \begin{cases} 
\text{weight}_{1_{\text{ext}}} \times TextCred(p.text) + \\
\text{weight}_{1_{\text{user}}} \times UserCred(p.user) + \\
\text{weight}_{1_{\text{social}}} \times SocialCred(p.user) + \\
\text{weight}_{1_{\text{topic}}} \times TopicCred(p.text) 
\end{cases} \quad \begin{cases} 
\text{weight}_{2_{\text{ext}}} \times TextCred(p.text) + \\
\text{weight}_{2_{\text{user}}} \times UserCred(p.user) + \\
\text{weight}_{2_{\text{social}}} \times SocialCred(p.user) 
\end{cases} \quad \text{if Topic analysis is possible,}
\]

\[ \text{Cred}(p) = \begin{cases} 
\text{weight}_{1_{\text{ext}}} \times TextCred(p.text) + \\
\text{weight}_{1_{\text{user}}} \times UserCred(p.user) + \\
\text{weight}_{1_{\text{social}}} \times SocialCred(p.user) + \\
\text{weight}_{1_{\text{topic}}} \times TopicCred(p.text) 
\end{cases} \quad \text{Otherwise.} \]
Case 3: Twitter credibility measure

Yudith Cardinale

MapReduce and Spark

May 2023

Credibility: 68.731812366

Dhall @dhall-lang · Jun 15, 2020

Version 17.0.0 of the standard is out:

- Quoted URLs no longer supported
- Optional/[build fold] no longer built-in
- Empty field labels permitted

Full changelog here:

Release v17.0.0 · dhall-lang/dhall-lang

Breaking changes: Remove the ability to quote paths in URLs After a deprecation period, quoted path ...

github.com

Credibility: 67.928262548

Dhall @dhall-lang · Jun 13, 2020

There is now a $250 bounty to generalize the dhall-kubernetes-generator project into a dhall-to-openapi project. If you are interested, comment on the following thread:

Expense proposal: Turn dhall-kubernetes-generator ...

I would like to propose funding $250 to factor out dhall-kubernetes-generator into a standalone op... 

discourse.dhall-lang.org

Credibility: 67.67539951
Case 3: Twitter credibility measure

- Twitter API
- HTML
- Data Extraction Features
  - API connection
  - Scraper
- Text
- Account age
- Verified
- Followings
- Followers
- Text Credibility
- User Credibility
- Social Credibility
- Tweet Credibility Level

Yudith Cardinale
MapReduce and Spark
May 2023
Case 3: Twitter credibility measure

Topic measure

Bot detection
Encore une question :

How **MapReduce or Spark** can be used in this scenario?

Any idea?
Case 3: Twitter credibility measure
Case 3: Twitter credibility measure

Summary:

- We have implemented batch and streaming processing to prepare data for the T-CREo credibility model;

  - **Streaming processing**: To calculate the credibility level in real time;
  - **Batch processing**:
    - To consider historic data/behaviour (Historical credibility);
    - To create datasets for more experiments.

- **Future work**:
  - Implement the whole pipeline (processing and analysis) with Spark facilities (Machine Learning).
  - Consider other parameters in the credibility model (retweets, likes, ...).
Last question:
Do you have another example in which MapReduce or Spark can be used?
Muchas gracias por la atención!
Yudith Cardinale
yudith.cardinale@campusviu.es
ycardinale@usb.ve