Introduction to Big Data

1

Big data


2

Data on the Internet…

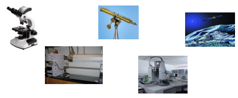
- Internet live stats
  - http://www.internetlivestats.com/

3

Who generates big data?

- User Generated Content (Web & Mobile)
  - E.g., Facebook, Instagram, Yelp, TripAdvisor, Twitter, YouTube


- Health and scientific computing



4

Who generates big data?

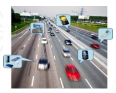
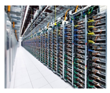
- Log files
  - Web server log files, machine system log files

- Internet Of Things (IoT)
  - Sensor networks, RFID, smart meters

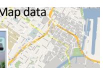
5

An example of Big data at work

- Crowdsourcing



Computing


Sensing


Map data


Real time traffic info

6

What is big data?

- Many different definitions
  - "Data whose scale, diversity and complexity require new architectures, techniques, algorithms and analytics to manage it and extract value and hidden knowledge from it"

7

7

What is big data?

- Many different definitions
  - "Data whose **scale**, **diversity** and **complexity** require new architectures, techniques, algorithms and analytics to manage it and extract value and hidden knowledge from it"

8

8

What is big data?

- Many different definitions
  - "Data whose scale, diversity and complexity require new **architectures**, **techniques**, **algorithms** and **analytics** to manage it and extract value and hidden knowledge from it"

9

9

The Vs of big data

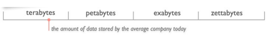
- The 3Vs of big data
  - **V**olume: scale of data
  - **V**ariety: different forms of data
  - **V**elocity: analysis of streaming data
- ... but also
  - **V**eracity: uncertainty of data
  - **V**alue: exploit information provided by data

10

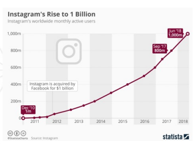
10

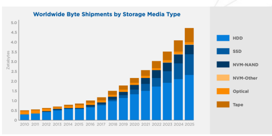
The Vs of big data

- **V**olume



  - Data volume increases exponentially over time
  - 44x increase from 2009 to 2020
    - Digital data 35 ZB in 2020



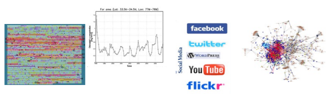


11

11

The Vs of big data

- **V**ariety
  - Various formats, types and structures
    - Numerical data, image data, audio, video, text, time series



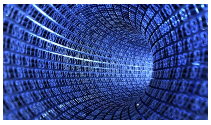
  - A single application may generate many different formats
    - Heterogeneous data
    - Complex data integration problem

12

12

The Vs of big data

- **V**elocity
  - Fast data generation rate
    - Streaming data
  - Very fast data processing to ensure timeliness



13

The Vs of big data

- **V**eracity
  - Data quality

14

The Vs of big data

- **V**alue
  - Translate data into business advantage

	Potential impact: 2011 research	Value captured %	Major barriers
Location-based data	• $100 billion+ revenues for service providers • Up to $700 billion value to end users	50–60	• Penetration of GPS-enabled smartphones globally
US retail[1]	• 60%+ increase in net margin • 0.5–1.0% annual productivity growth	30–40	• Lack of analytical talent • Siloed data within companies
Manufacturing[2]	• Up to 50% lower product development cost • Up to 25% lower operating cost • Up to 30% gross margin increase	20–30	• Siloed data in legacy IT systems • Leadership skeptical of impact
EU public sector[3]	• ~€250 billion value per year • ~0.5% annual productivity growth	10–20	• Lack of analytical talent • Siloed data within different agencies
US health care	• $300 billion value per year • ~0.7% annual productivity growth	10–20	• Need to demonstrate clinical utility to gain acceptance • Interoperability and data sharing

1 Similar observations hold true for the EU retail sector.
2 Manufacturing levers divided by functional application.
3 Similar observations hold true for other high-income country governments.

SOURCE: Expert interviews; McKinsey Global Institute analysis

15

Big data value chain

Generation → Acquisition → Storage → Analysis

- Generation
  - Passive recording
    - Typically, structured data
    - Bank trading transactions, shopping records, government sector archives
  - Active generation
    - Semi-structured or unstructured data
    - User-generated content, e.g., social networks
  - Automatic production
    - Location-aware, context-dependent, highly mobile data
    - Sensor-based Internet-enabled devices

16

Big data value chain

Generation → Acquisition → Storage → Analysis

- Acquisition
  - Collection
    - Pull-based, e.g., web crawler
    - Push-based, e.g., video surveillance, click stream
  - Transmission
    - Transfer to data center over high capacity links
  - Preprocessing
    - Integration, cleaning, redundancy elimination

17

Big data value chain

Generation → Acquisition → Storage → Analysis

- Storage
  - Storage infrastructure
    - Storage technology, e.g., HDD, SSD
    - Networking architecture, e.g., DAS, NAS, SAN
  - Data management
    - File systems (HDFS), key-value stores (Memcached, CEPH), column-oriented databases (Cassandra), document databases (MongoDB)
  - Programming models
    - **Map reduce**, **split apply combine**, stream processing, graph processing

18

Big data value chain

Generation → Acquisition → Storage → Analysis

- Analysis
  - Objectives
    - Descriptive analytics, predictive analytics, prescriptive analytics
  - Methods
    - Statistical analysis, data mining, text mining, network and graph data mining
    - Clustering, classification and regression, association analysis
  - Diverse domains call for customized techniques

19

Big data challenges

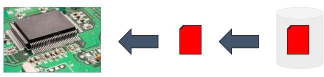
- Technology and infrastructure
  - New architectures, programming paradigms and techniques are needed
- Data management and analysis
  - New emphasis on "data"

➡ **Data science**

20

The bottlenecks

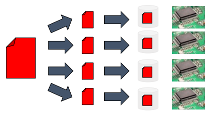
- Processors process data
- Hard drives store data
- We need to transfer data from the disk to the processor



21

The solution

- **Transfer the processing power to the data**
- Multiple distributed disks
  - Each one holding a portion of a large dataset
- Process in parallel different file portions from different disks



22

How a Big Data cluster work?

23

How to Handle the Big Data?

Traditionally there were compute-bound tasks
- Small datasets
- Complex algorithms

➢ Not suitable for large dataset

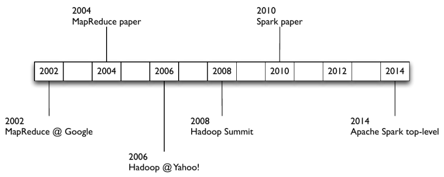
Opportunity: Performance is increased by
- Including more processors
- Investing in fast memory

Challenges:
- Split and distribute the task
- Synchronize threads
- Handle failures etc

➢ Born of the Hadoop framework

24

History of Hadoop



- S. Ghemawat, H. Gobioff, and S. Leung. “The Google File System”, ACM SIGOPS Operating Systems Review, Vol. 37, No. 5, 2003.
- J. Dean, and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”, OSDI, 2004.
- M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica. “Spark: Cluster Computing with Working Sets”, HotCloud, 2010.

Figure: Paco Nathan slides - http://training.databricks.com/workshop/sparkcamp.pdf

25

25

Key Ideas on Hadoop

- **Data locality principle**
  - Move algorithms to the data, not data to the algorithms
- **Failures** are the norm, not the exception
  - The framework takes care of splitting data, synchronizing tasks, recovering in case of failures of a task or a server etc.
- **Data intensive workloads**
  - A batch processing framework designed to perform full reads of the input, thus avoiding random access
- **Horizontal scalability** based on commodity servers
  - E.g., doubling the number of servers, halving processing time

26

26

Typical Architecture of Big Data Clusters

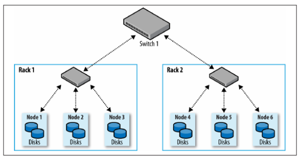


Figure 10-1. Typical two-level network architecture for a Hadoop cluster

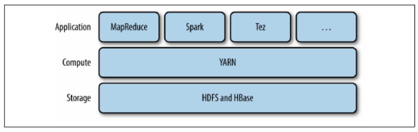
- Bunch of ordinary servers, switches etc
- Both storage and processing capacity at all servers
- Nodes play the role of masters, workers, etc.

Figure: Tom White - Hadoop: The Definitive Guide, 4th Edition, 2015

27

27

Basic Hadoop Ecosystem



- HDFS – Hadoop Distributed File System
- YARN – Yet Another Resource Negotiator
- Applications : MapReduce, Spark etc

Figure: Tom White - Hadoop: The Definitive Guide, 4th Edition, 2015

28

28

HDFS – Reading Logic

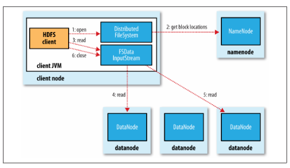


Figure 3-2. A client reading data from HDFS

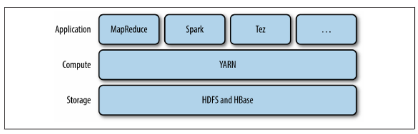
- Files are split in blocks (e.g., 64 MB)
- Blocks are stored across different disks/server/racks

Figure: Tom White - Hadoop: The Definitive Guide, 4th Edition, 2015

29

29

Basic Hadoop Ecosystem



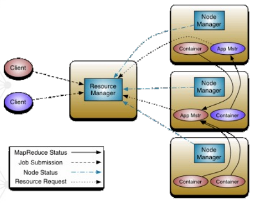
- HDFS – Hadoop Distributed File System
- **YARN – Yet Another Resource Negotiator**
- Applications : MapReduce, Spark etc

Figure: Tom White - Hadoop: The Definitive Guide, 4th Edition, 2015

30

30

YARN – Key Roles



Keywords
- Resource Manager
- Node Manager
- Application Master
- Containers

Managed resources
- CPU
- Memory

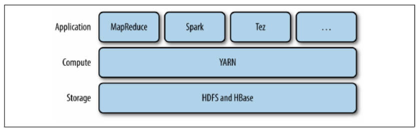
- (Max) CPU and memory usage is pre-allocated per application
- Disk and network usage is consequence of the application logic

Figure: Tom White - Hadoop: The Definitive Guide, 4th Edition, 2015

31

31

Basic Hadoop Ecosystem



- HDFS – Hadoop Distributed File System
- YARN – Yet Another Resource Negotiator
- Applications : MapReduce, **Spark** etc

Figure: Tom White - Hadoop: The Definitive Guide, 4th Edition, 2015

32

32

Spark

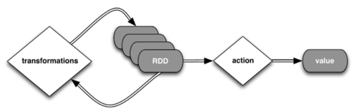
Key points
- Separate **What** from **How**
- Batch, interactive, and real-time within a single framework
- Integration with Java, Python, Scala
- Programming at a high level of abstraction, using functional programming

Practical aspect
- **Data loaded in memory → high speed and flexibility**

33

33

Spark – Basic Working



- RDD – Resilient Distributed Dataset
- Transformations – create a new RDD from an existing one
- Action – extract values from the RDD

Figure: Paco Nathan slides - http://training.databricks.com/workshop/sparkcamp.pdf

34

34

Spark – RDDs

Resilient
- In case of failures, the Spark environment knows how to rebuild a RDD

Distributed
- A collection of elements distributed in the cluster

They are **immutable** and static typed
- You **transform** a RDD into a new RDD

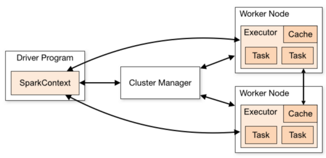
Lazy: RDDs are computed when an action is performed

RDDs can be **persisted in memory or disk**

35

35

Spark – Cluster Execution Overview



1. The application creates a driver process
2. The application gets its executor processes
3. It sends the code and tasks to the executors
4. **Key roles are played by the driver and the executors!**

Figure: http://spark.apache.org/docs/latest/cluster-overview.html

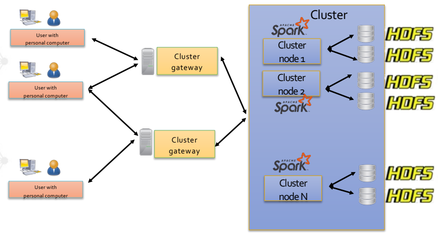
36

36

What can we do with Big Data?

37

Big Data Cluster - Architecture



38

The magic of Big Data technology

- How much time to get the result?
  - For 2 years of network log files

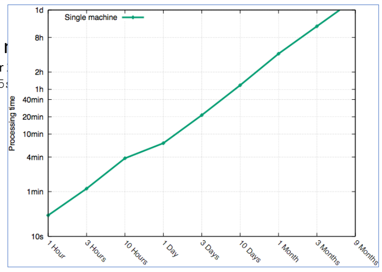
`1h~15s => 1d~3.5min => 1month~1.75h => 1year~1d`

39

The magic of Big Data technology

- How r
  - For

1h~15s



40

The magic of Big Data technology

- How much time to get the result?
  - For 2 years of network log files

`1h~15s => 1d~3.5min => 1month~1.75h => 1year~1d`

- What if we parallelize the computing?
  - N computing unit
  - Each unit count on 1/N of data
    - MAP data to computing unit
  - And sends the results back
    - REDUCE the data

  - Speedup of ~N

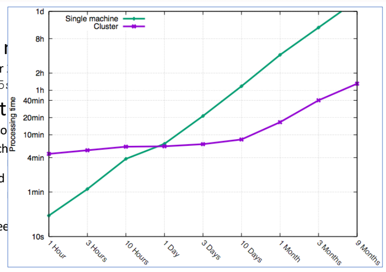
41

The magic of Big Data technology

- How r
  - For

1h~15s

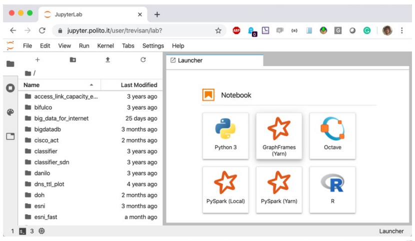
- What
  - N co
  - Each
  - And
  - Spee



42

Using a cluster

Go to https://jupyter.polito.it/, and login with your credentials



43

Same Interface, many services

Supported frameworks

- Simple notebooks (no Big Data)

Python 3 Octave R

- Work with BigData - Spark notebooks

PySpark (Local) PySpark (Yarn)

Prototyping (single machine) Scaling (all cluster)

44

Read data

With Spark, it is trivial to read **TBs of data**.

E.g., Read all the data of 30 k Instagram influencers over 1 year

```
comments      = spark.read.load("/data/SMARTDATA/social_networks/instagram_it/comments/*",          format="json")
profiles      = spark.read.load("/data/SMARTDATA/social_networks/instagram_it/profiles_periodic/*" , format="json")
medias        = spark.read.load("/data/SMARTDATA/social_networks/instagram_it/medias/*",            format="json")
```

With 3 lines of code, you read millions of comments:

created_time	created_time_str	id	media_code	mentioned_usernames	owner_id	owner_username	parent_comment_id	tags	text
1558946947	2019-05-27 10:49:07	17842793725463985	Bx9pIUWoOqP	[]	1532952670	fabry_il_marsy	null	[]	Bellissima
1558957621	2019-05-27 11:47:01	17842803448467320	BxZq4oZIVAl	[_mariachiaragreco]	495765650	marilusantonocito_	18036948625148574	[]	@_mariachiaragrec...
1558949682	2019-05-27 11:34:42	17842813900467387	Bx9sdP0CCFG	[]	423046839	alessandrotampieri_	null	[]	E poi pensi che a...
1558982879	2019-05-27 18:47:59	17842896571463223	Bx-czCuFOVr	[]	281232079	valerio261077	null	[]	Si trovano negli ...
1558978096	2019-05-27 19:28:16	17842902646463291	Bx9R7GMI1kp	[]	1512625090	kinga.matuszczak	null	[]	The best of the b...
1558936959	2019-05-27 08:02:39	17843057011467699	Bx9Ooh2CcOy	[silbo80di]	30925955	vintageblackboard	null	[]	@silbo80di la tua...

```
comments.count()
```

173701411

45

Process data

Do (simple) analytics on large data to extract knowledge

E.g., Who are the influencers that published more posts?

```
medias.groupby('owner_username').count().sort("count", ascending=False).limit(10).toPandas()
```

	owner_username	count
0	matteosalviniofficial	5066
1	lucatommassiniofficial	4625
2	napolimagazine	4229
3	__sonia69__	4163
4	blckthemall_italy	3368
5	tina.gia	3130
6	_luxury_fashion_style	3054
7	andrea_vento_viaggi	3021
8	passionedolomiti	2925
9	isaechia	2922

Spark offers simple Python API to process data
Two set of APIs:

1. RDD: based on functional programming
2. DataFrame: SQL-like data manipulation

The same simple code can run on you PC or on (our) huge cluster!

46

Visualize data

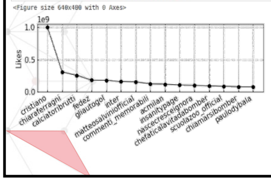
Support for data visualization:

- Classical plots (for writing succesful papers)
- Advanced charts (e.g., graphs)

```
likes = medias.groupby("owner_username").sum("likes_count").sort("sum(likes_count)", ascending=False).limit(15).toPandas()
```

```
fastplot.plot( (likes.owner_username.values, likes["sum(likes_count)"].values), None,
              ylabel = "Likes", **PLOT_ARGS).show()
```

<Figure size 640x480 with 0 Axes>



47

Visualize data

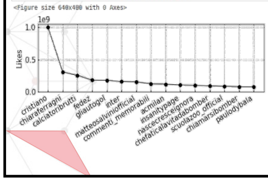
Support for data visualization:

- Classical plots (for writing succesful papers)
- Advanced charts (e.g., graphs)

```
likes = medias.groupby("owner_username").sum("likes_count").sort("sum(likes_cou
```

```
fastplot.plot( (likes.owner_username.values, likes["sum(likes_count)"].values),
              ylabel = "Likes", **PLOT_ARGS).show()
```

<Figure size 640x480 with 0 Axes>



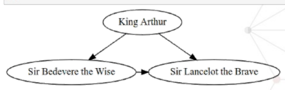

```
from graphviz import Digraph

dot = Digraph(comment='The Round Table')

dot.node('A', 'King Arthur')
dot.node('B', 'Sir Bedevere the Wise')
dot.node('L', 'Sir Lancelot the Brave')

dot.edges(['AB', 'AL'])
dot.edge('B', 'L', constraint='false')

dot
```



48

Big Data - Use Cases

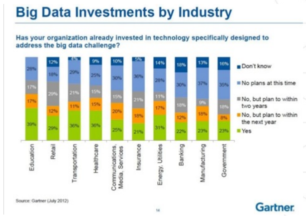
What you can do:
- **Quantitative statistics**: distributions, aggregations, counting, …
- **Build big graphs**: using the GraphFrames Spark library
- **Use simple machine learning**: using the Spark ML library

What you cannot do:
- **High Performance Computing**: use the HPC cluster instead
- **Train large-sized neural networks**: if no GPU available
- **Use polynomial algorithms**: if an algorithm is O(n^2) won't scale!

49

Conclusions

- Certainly not just hype
- … but not a panacea!



50

Perguntas
Fragen Domande Galdera Otázky
Questions
Spørgsmål Pertanyaan kysymykset
Frågor Spørsmål Cwestiynau
вопросы Preguntes Sorular
Въпроси
Vragen
Pytania

51