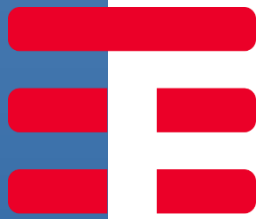

GRUPPO TELECOM ITALIA
Big Data Transformation
Big Data Transformation
TIM Academy



GRUPPO TIM

Big Data Transformation

Apache Spark Fundamentals & Query Fundamentals

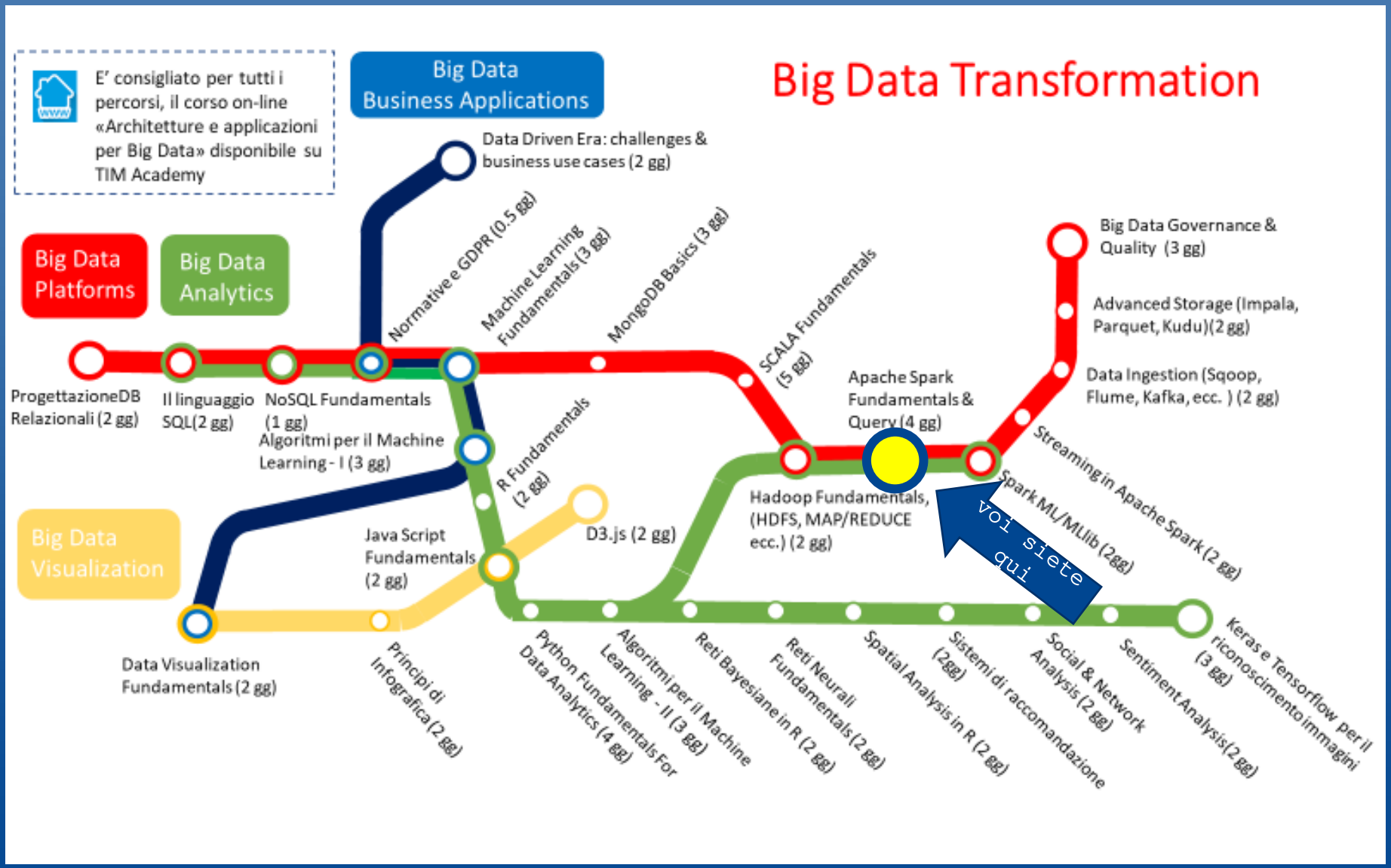
TIM Academy

GRUPPO TIM

Big Data Transformation

Apache Spark Fundamentals & Query

TIM Academy


E' consigliato per tutti i percorsi, il corso on-line «Architetture e applicazioni per Big Data» disponibile su TIM Academy
Big Data Business Applications
Big Data Transformation
Big Data Platforms
Big Data Analytics
Big Data Visualization
Data Driven Era: challenges & business use cases (2 gg)
Normative e GDPR (0.5 gg)
Machine Learning Fundamentals (3 gg)
MongoDB Basics (3 gg)
SCALA Fundamentals (5 gg)
Big Data Governance & Quality (3 gg)
Advanced Storage (Impala, Parquet, Kudu)(2 gg)
Data Ingestion (Sqoop, Flume, Kafka, ecc.) (2 gg)
Apache Spark Fundamentals & Query (4 gg)
Streaming in Apache Spark (2 gg)
Spark ML/MLlib (2gg)
voi siete qui
ProgettazioneDB Relazionali (2 gg)
Il linguaggio SQL(2 gg)
NoSQL Fundamentals (1 gg)
Algoritmi per il Machine Learning - I (3 gg)
R Fundamentals (2 gg)
Hadoop Fundamentals, (HDFS, MAP/REDUCE ecc.) (2 gg)
Java Script Fundamentals (2 gg)
D3.js (2 gg)
Data Visualization Fundamentals (2 gg)
Principi di Infografica (2 gg)
Python Fundamentals For Data Analytics (4 gg)
Algoritmi per il Machine Learning - II (3 gg)
Reti Bayesiane in R (2 gg)
Reti Neurali Fundamentals (2 gg)
Spatial Analysis in R (2 gg)
Sistemi di raccomandazione (2gg)
Social & Network Analysis (2 gg)
Sentiment Analysis(2 gg)
Keras e Tensorflow per il riconoscimento immagini (3 gg)

TIM Academy

Apache Spark

Uno dei tool più usati nel campo dei ***BIG DATA***

https://www.whizlabs.com/blog/big-data-tools/

1° APACHE hadoop

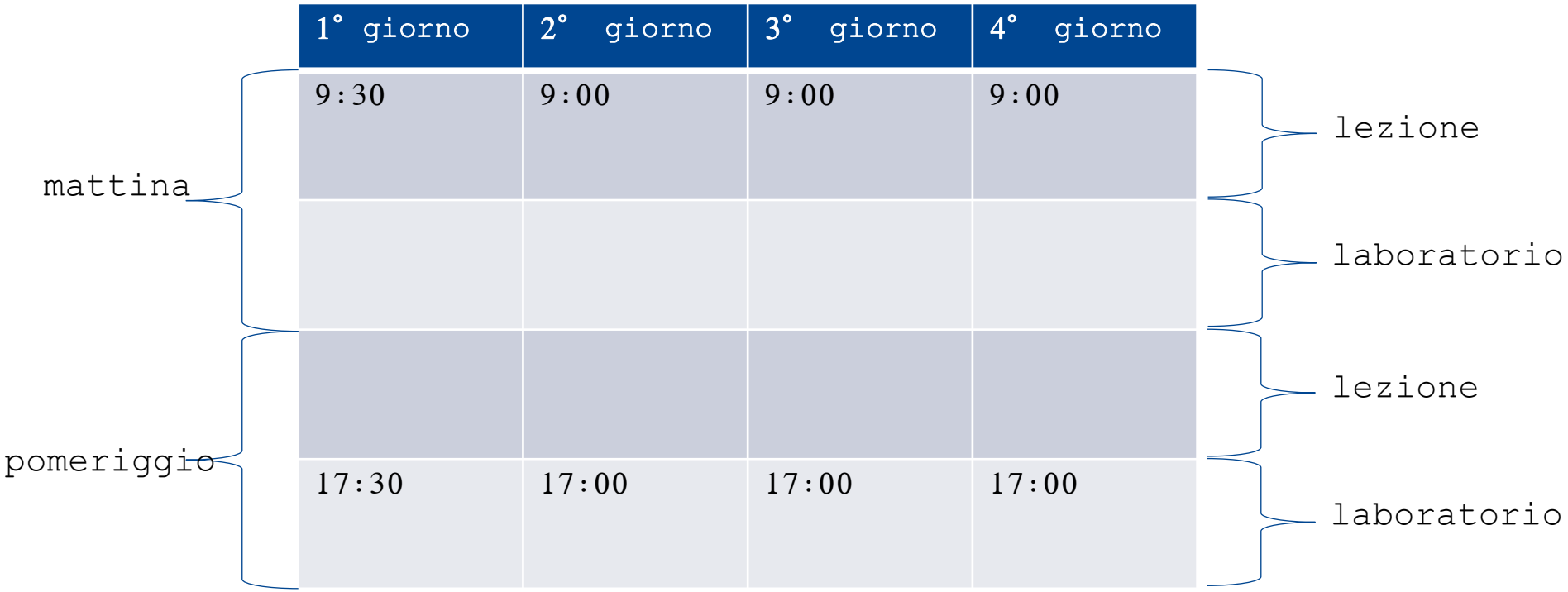
2° APACHE Spark™

Prerequisiti

- Conoscenza del linguaggio di programmazione Scala

- Aver seguito il corso Hadoop Fundamentals

Agenda del corso



Agenda 1/8

Introduzione

Presentazione del corso

Spark
Cos'è? A cosa serve? Chi lo usa? Punti di forza e debolezze

Contesto e storia
Come è nato? Chi l'ha fatto? Quale problema affronta?

Com'è fatto?
Una prima presentazione dell'architettura

E in pratica?
Come si usa? Come lo useremo noi in laboratorio?

Agenda 2/8

Le fondamenta

Strutture dati
RDDs, DataSets e DataFrames

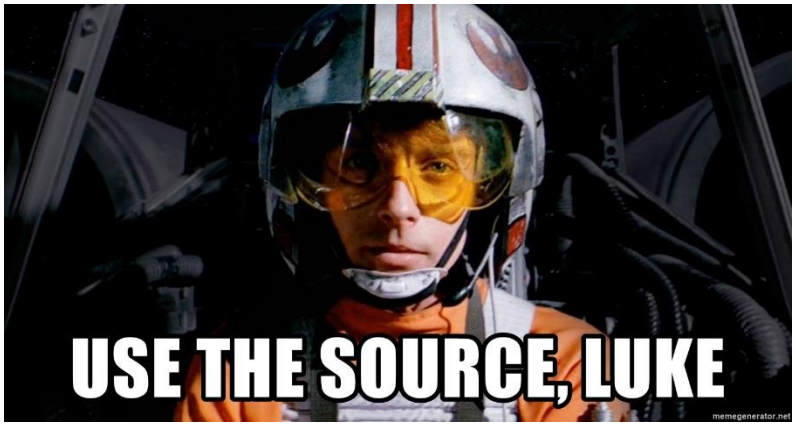
Operazioni sui dati
Transformazioni (*narrow* e *wide*) e Azioni; Valutazioni «pigre»; persistenza (cache)

Esecuzione
Tasks e Jobs; Spark UI; resilienza (lineage); DAG

Un pizzico di teoria
Programmazione funzionale per il calcolo parallelo

Coordinate
Documentazione; le varie distribuzioni: Cloudera, Hortonworks, ecc.; sorgente su github

Esercizi di laboratorio
Filtri e aggregazioni; i widget di Spark Notebook



Agenda 3/8

API

DataFrames e DataSet
Definizione e differenze

Gli schemi
Nome e tipo delle colonne; definiti manualmente o letti da un *data source*.

Catalyst
Spark SQL, piani logici e fisici.

I/O
Lettura e scrittura di file; HDFS, Hive, Local FS, S3, SequenceFile,…

Colonne ed espressioni
Vari tipi di trasformazioni su DataFrame e DataSet; selezioni, filtri, or

Gestire le stringhe
Trimming, Split (Explode), Espressioni regolari, Date e TimeStamp, Json

Funzioni definite dall'utente
Le UDF

RDD (forse va spostato)

SQL (forse va spostato)

Esercizi di laboratorio
Esercitazioni sugli argomenti trattati.

```
root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: double (nullable = true)
 |-- Country: string (nullable = true)
```

```
+---------+---------+--------------------+--------+-------------------+----...
|InvoiceNo|StockCode|         Description|Quantity|        InvoiceDate|Unit...
+---------+---------+--------------------+--------+-------------------+----...
|   536365|   85123A|WHITE HANGING HEA...|       6|2010-12-01 08:26:00|    ...
|   536365|    71053| WHITE METAL LANTERN|       6|2010-12-01 08:26:00|    ...
...
|   536367|    21755|LOVE BUILDING BLO...|       3|2010-12-01 08:34:00|    ...
|   536367|    21777|RECIPE BOX WITH M...|       4|2010-12-01 08:34:00|    ...
+---------+---------+--------------------+--------+-------------------+----...
```

Agenda 4/8

Aggregazioni

Tipi di aggregrazione
Intero DataFrame, *group by*, *window*, *grouping set*, *rollup*

$$\left(\sum_{i=1}^{n} p_i x_i^2\right) - \mu^2$$

Statistiche
Conteggi, somme, media, varianza, ecc.

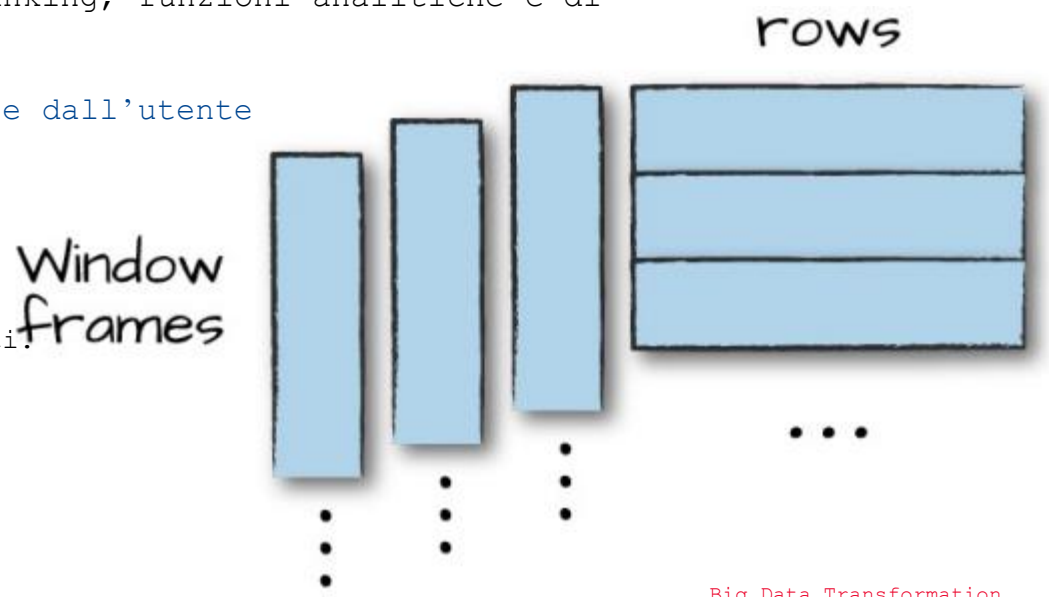
Aggregazione su tipi complessi
Insiemi e liste

Window
Funzioni su finestre di dati: ranking, funzioni analitiche e di aggregazione

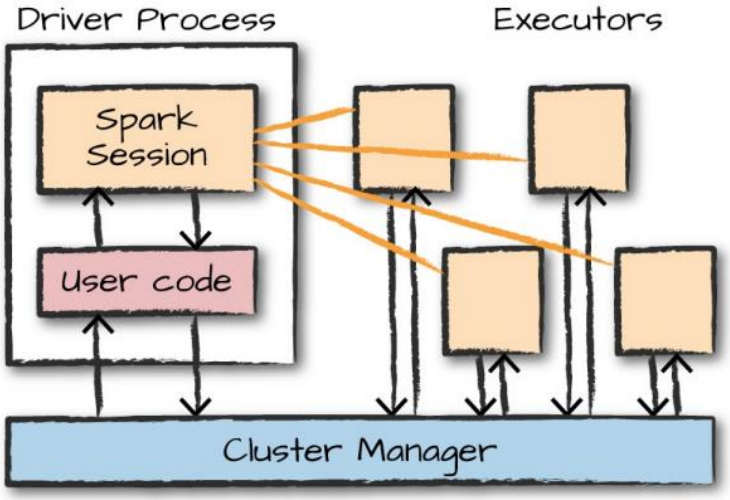
Funzioni di aggregazione definite dall'utente

Joins
Inner Joins, Outer Joins, etc.

Esercizi di laboratorio
Esercitazioni sugli argomenti trattati.



Agenda 5/8



Il cluster

Architettura di un'applicazione Spark
Spark Driver, Spark executors,
cluster manager (Spark standalone, YARN o Mesos)

Modi di esecuzione
Cluster mode, client mode, local mode

Ciclo di vita di un'applicazione Spark
Fuori e dentro Spark; tasks, jobs, stages

La sessione
SparkSession, SparkContext, SQLContext, HiveContext

Il programa
Struttura; compilazione (maven, sbt); spark-submit

Configurazione
SparkConf, argomenti di spark-submit;

Esercizi di laboratorio
Scrittura dei primi programmi in Spark; compilazione ed esecuzione.

Agenda 6/8

- DO e DON'T

Strumenti di monitoraggio e debugging
Spark UI; log; submit in modalità client

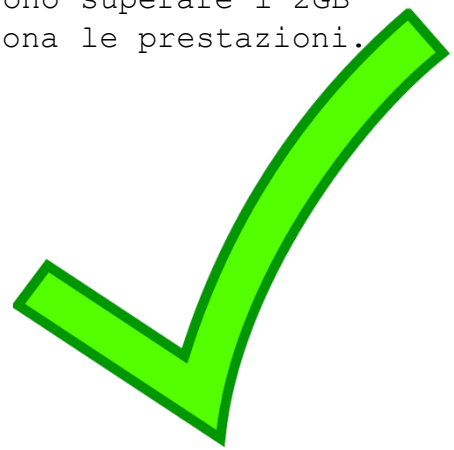
3 Ragioni per non usare i RDD
Sono obsoleti, più lenti e più complicati

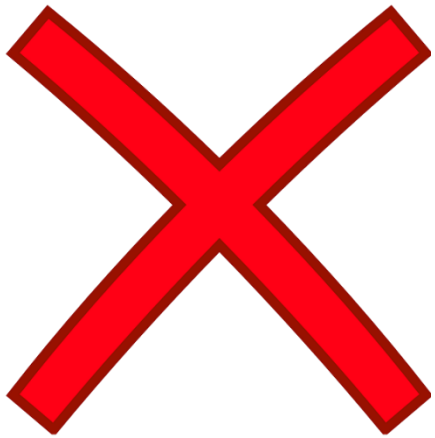
Decisioni, decisioni, decisioni!
Numero di esecutori? N. di core per esecutore? memoria per esecutore?

Numero di partizioni
I blocchi nello shuffle non devono superare i 2GB
Il numero di partizioni condiziona le prestazioni.

Controllare lo shuffling

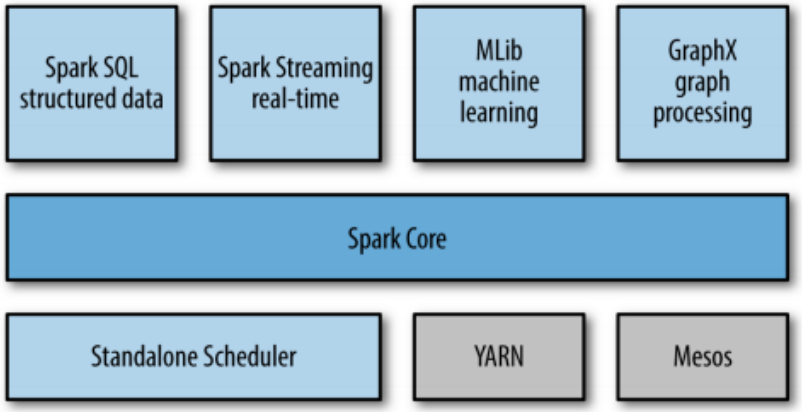
Controllare il DAG
GroupByKey => ReduceByKey
Reduce => TreeReduce
Uso di tipi complessi





Esercizi di laboratorio
Esercizi di recapitolazione

Agenda 7/8



- Spark stack

  Packages
  Spark SQL, Spark Streaming, Spark ML, GraphFra

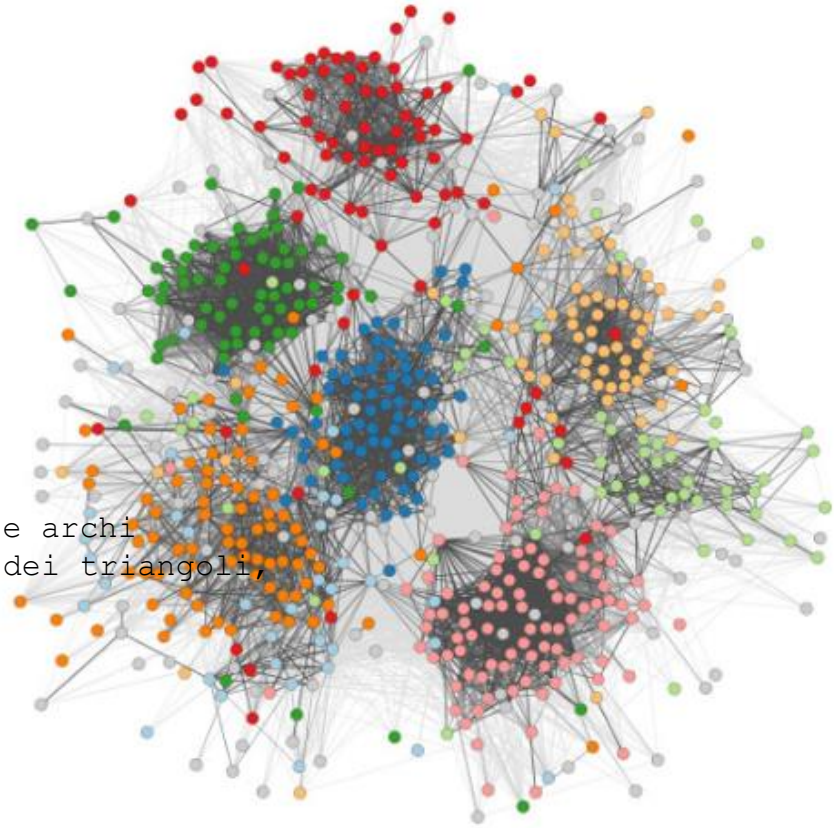
  Spark Streaming
  Breve introduzione: ci sarà un corso dedicato

  Spark ML
  Breve introduzione: ci sarà un corso dedicato
  Nota: ML (basato su DataFrames) vs. MLLib (basato su RDD)

  GraphFrames
  Breve introduzione: approfondimento nel pomeriggio
  Nota: GraphFrames (basato su DataFrames; 2016) vs. GraphX (basato su RDD; 2014)

  Esercizi di laboratorio
  Esercizi di recapitolazione

Agenda 8/8



GraphFrames

Cos'è un grafo?
Vertici e archi; concetti e algoritmi base.

GraphX vs GraphFrames

La classe GraphFrame
- Come creare un grafo a partire da vertici e archi
- Cercare le componenti connesse, conteggio dei triangoli, pageRank, ecc.
- Funzioni di ricerca del motif usando il Domain Specific Language (DSL)

Esercizi di laboratorio
Esercizi di recapitolazione

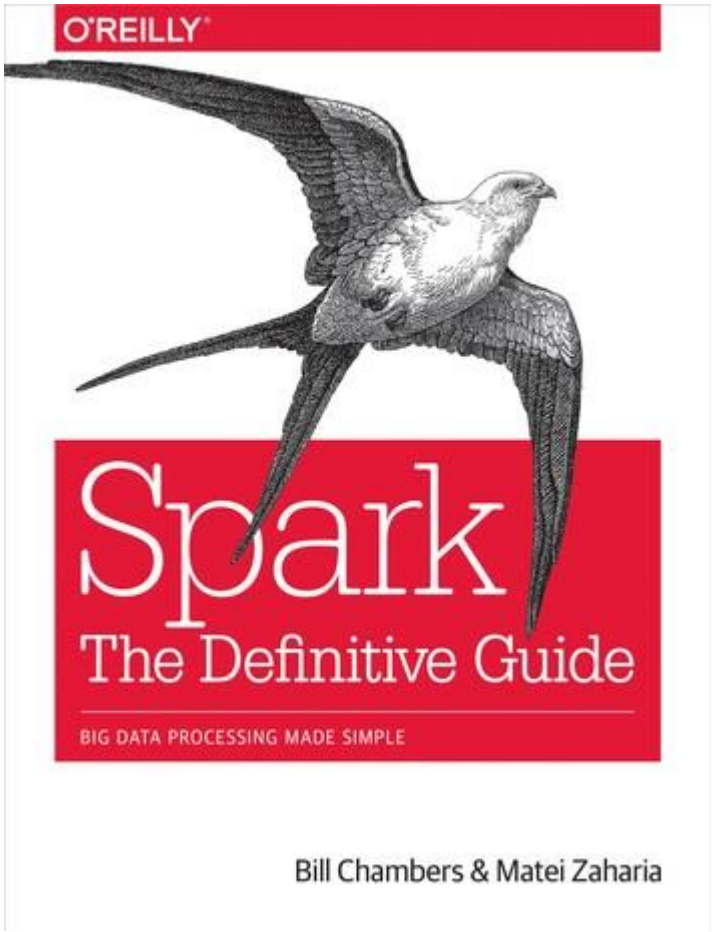
Conclusioni

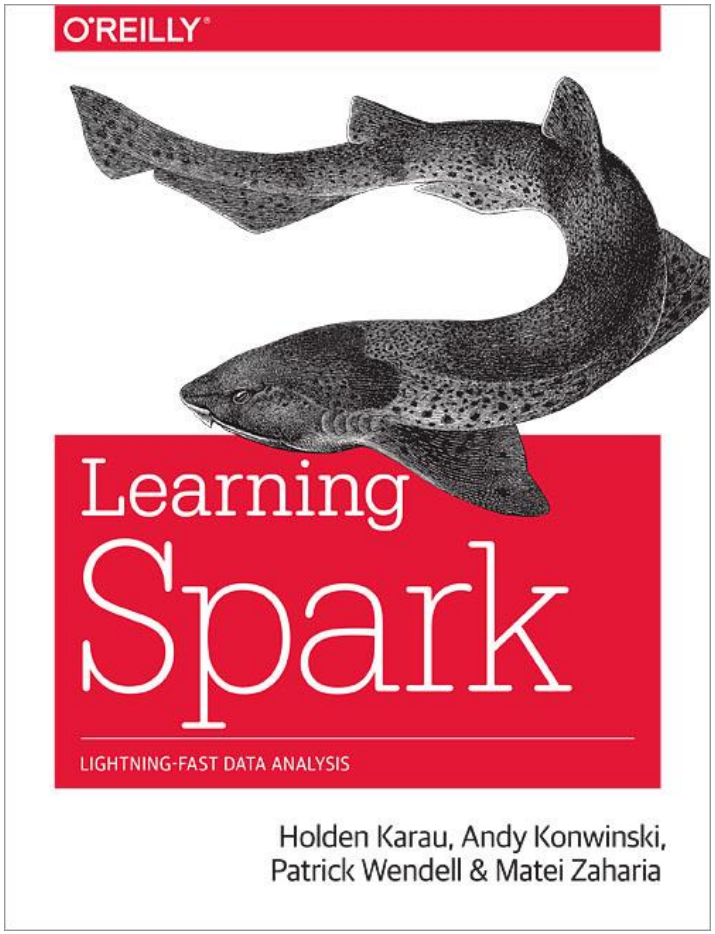
Recapitolazione e domande

Compilazione del questi 

Primo giorno: mattina

Riferimenti bibliografici





Cos'è Apache Spark?

Cos'è Apache Spark?

"Apache Spark is an **open-source distributed general-purpose cluster-computing framework**".

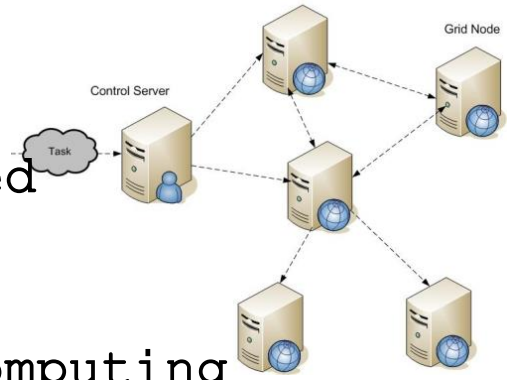
WIKIPEDIA
The Free Encyclopedia

- Open-source

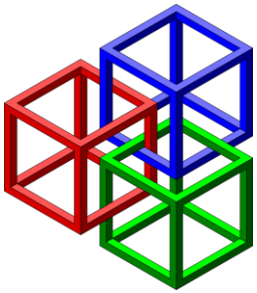
Open Source Initiative

- General-purpose

- Distributed



- Cluster-computing
- Framework



Computing… ad esempio?

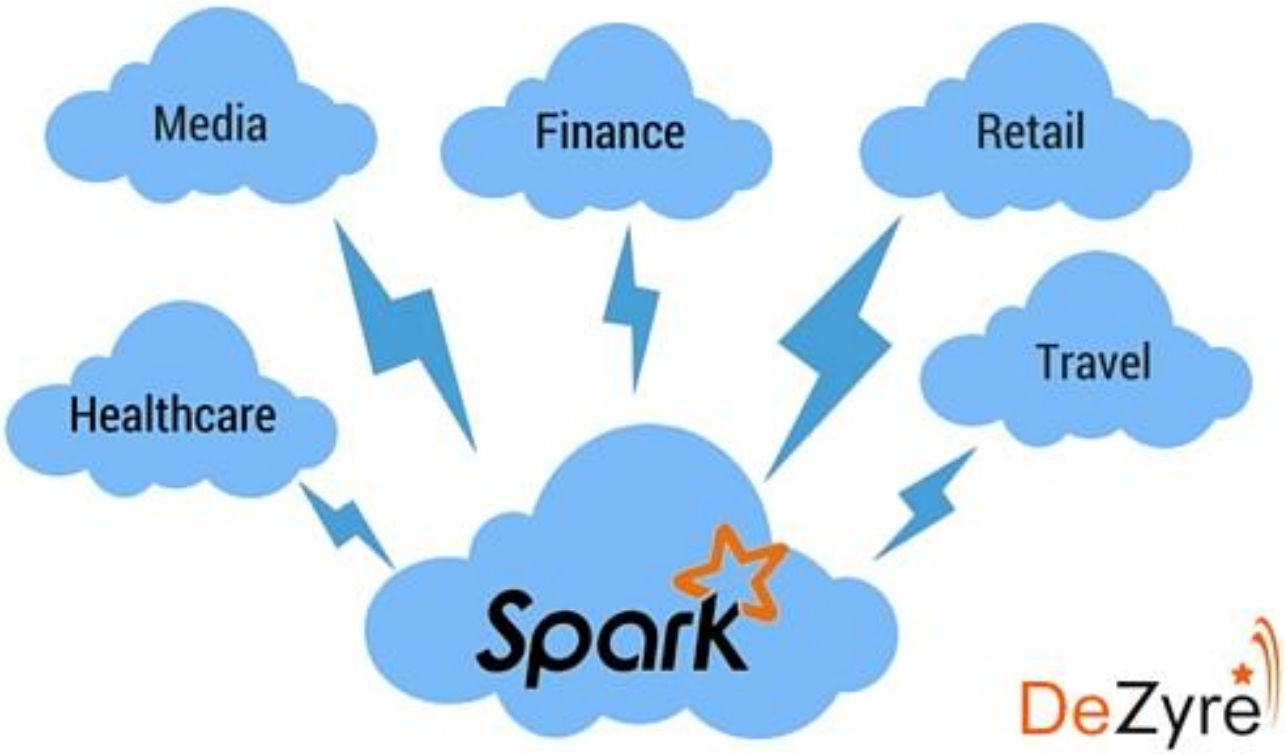
Vari utilizzi di Apache Spark

- Streaming
  - Streaming ETL (Extract, Transform, Load)
  - Data enrichment
  - Event detection (ad es. per individuare transizioni fraudolente, cambi potenzialmente pericolosi dei parametri vitali, ecc.)
  - Complex session analysis
- Machine learning
  - Marketing purposes and sentiment analysis
  - Network security
- Interactive analysis

https://www.qubole.com/blog/apache-spark-u

Computing… ad esempio?

Vari utilizzi di Apache Spark



https://www.dezyre.com/article/top-5-apache-spark-

Computing… ad esempio?

Test A/B

Click rate: 52 %

72 %

Qualche esempio concreto

Alibaba.com™

ebay

NASA Jet Propulsion Laboratory
California Institute of Technology

YAHOO!

Rif: https://spark.apache.org/powered-by.html

Qualche esempio concreto

Uber

Dal 2015 usa *Hadoop* + *Kafka* + *Spark (ecc.)*
100+ Petabytes con un minuto di latenza
10 000 job Spark / giorno

Cfr: https://eng.uber.com/uber-big-data-platform/

Pinterest

175 milioni di utenti mensilmente attivi
Test A/B
Kafka + Pinball + Spark Streaming + HBase &MemS
+ Presto

Cfr: https://medium.com/@Pinterest_Engineering/building-a-new-experiment-pipeline-wit

conviva®

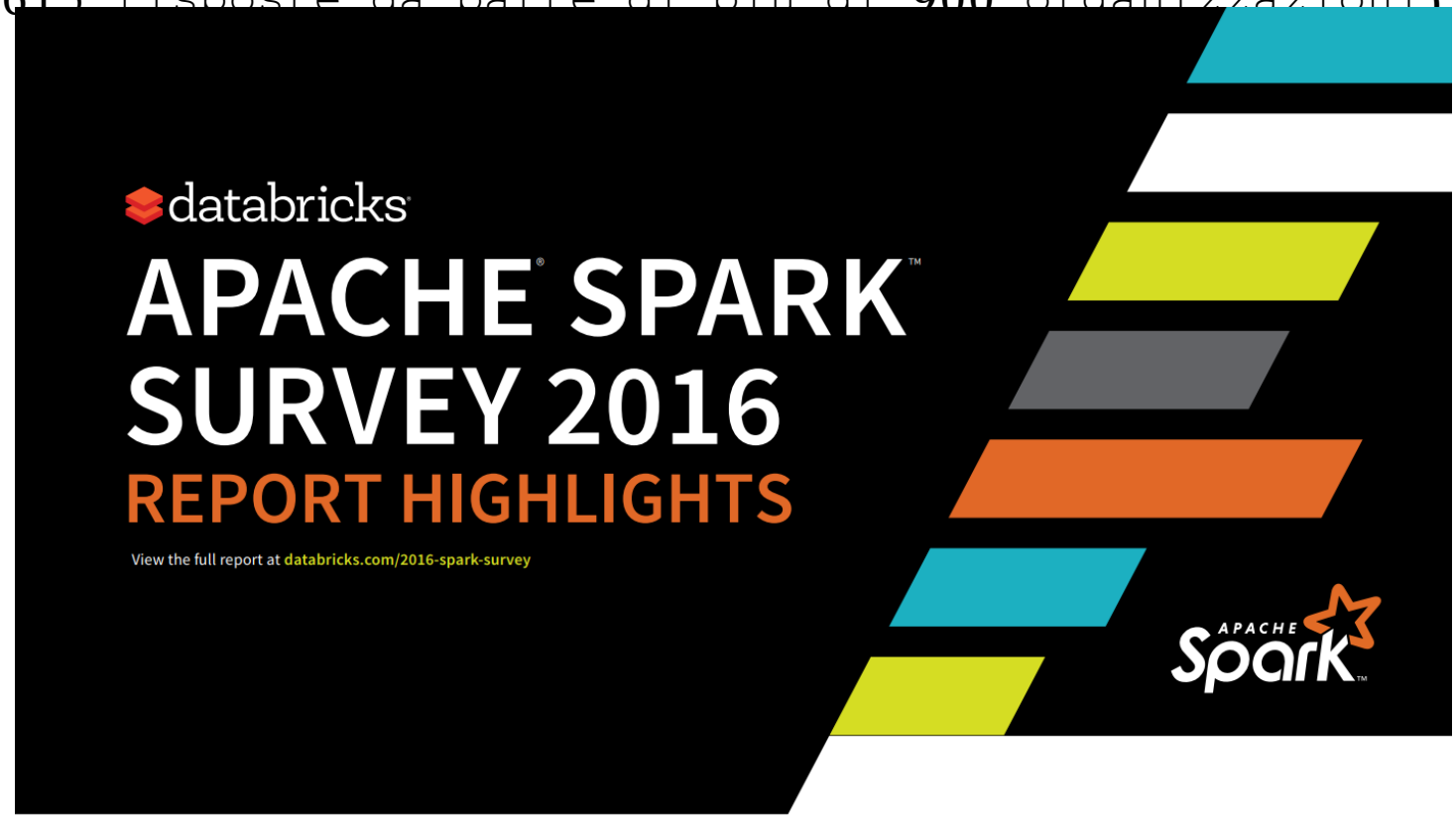
65 milioni di video stream / gior

http://ampcamp.berkeley.edu/wp-content/uploads/2012/06/dilip-joseph-amp-camp-2012-conviva-using-spark.pdf

https://www.qubole.com/blog/apache-spark-us

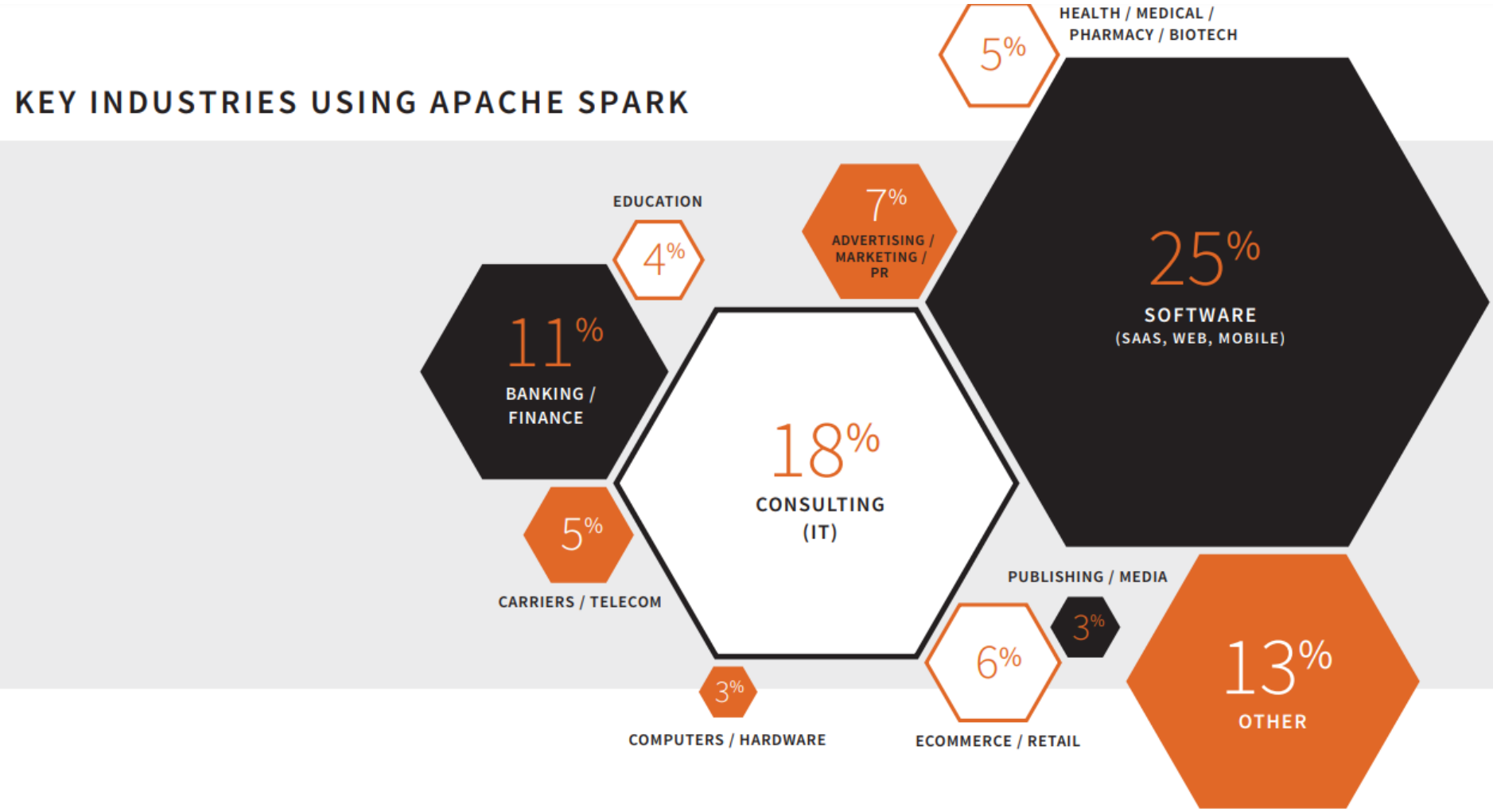
Chi usa Spark e perché

Nel 2016 i creatori di Spark hanno organizzato un sondaggio fra gli utenti.
(1615 risposte da parte di più di 900 organizzazioni)

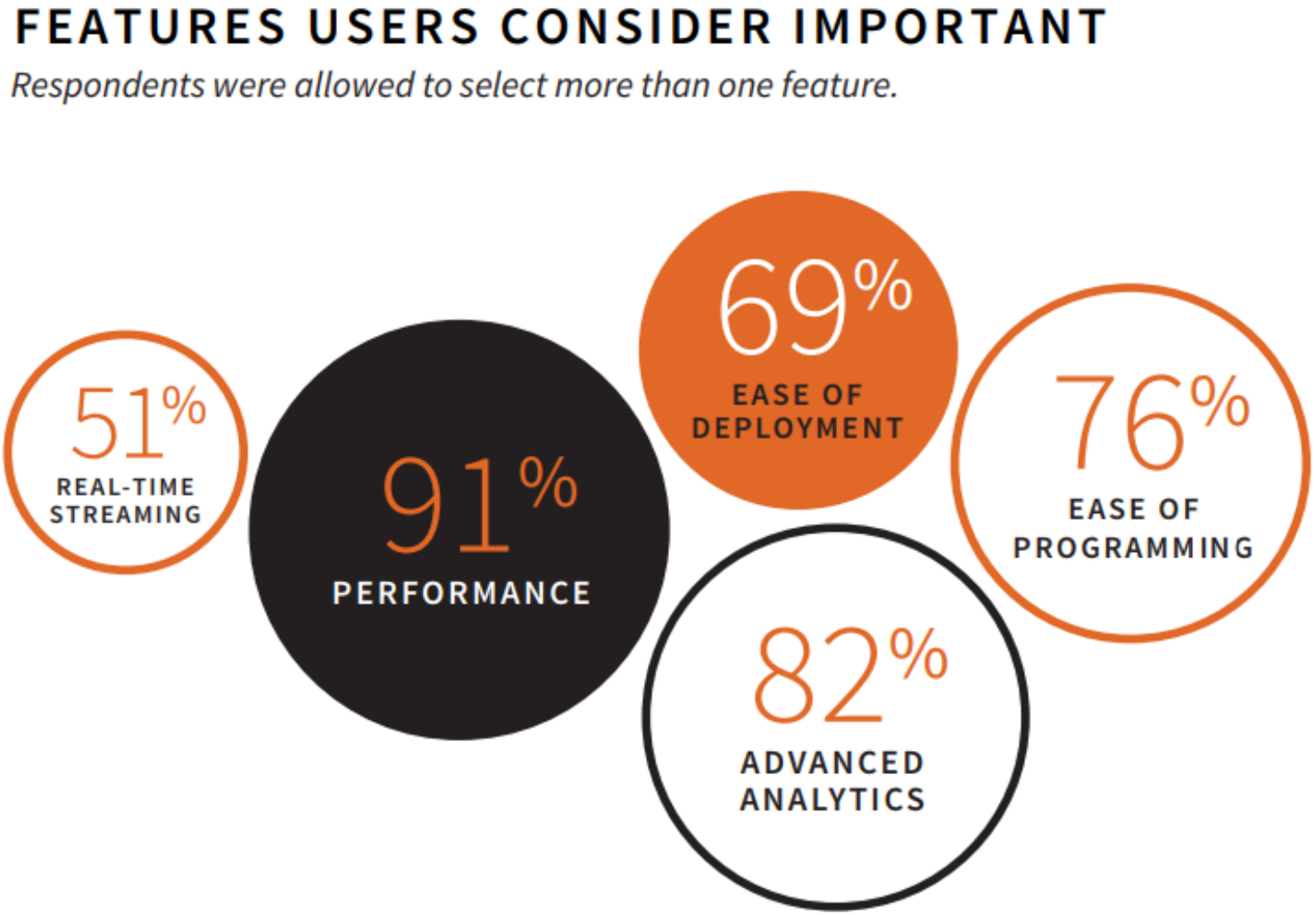


http://pages.databricks.com/rs/094-YMS-629/images/2016_Spark_Infographic.pdf

Chi usa Spark e perché



Chi usa Spark e perché



Perché Spark?

Principali punti di forza

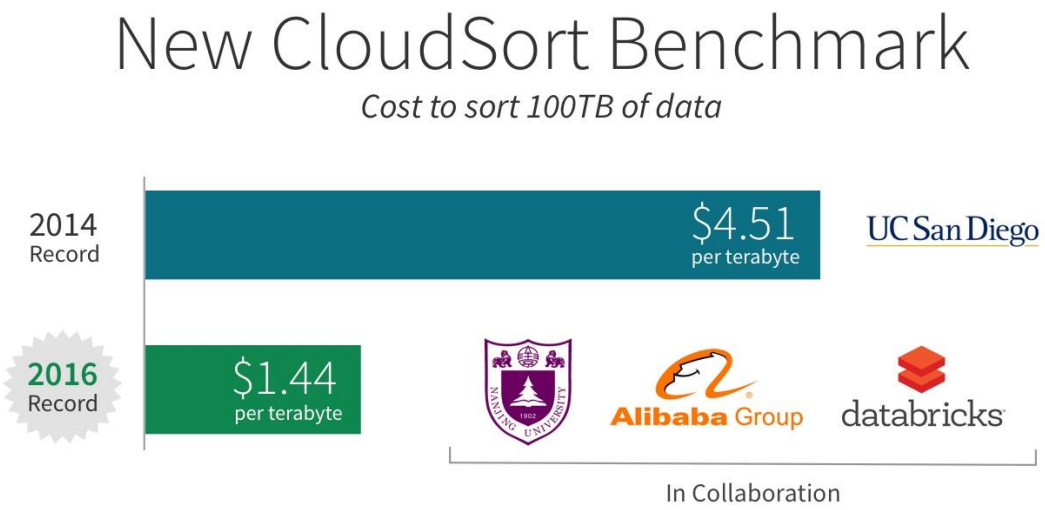
- Veloce
- Agnostico rispetto alla piattaforma
- Versatile
- Facile
- Capace di scalare

- Le «3 S» di Spark:
  - **Speed** = Velocità
  - **Semplicity** = Facilità di uso
  - **Support** Vari linguaggi e sistemi di storage; Provider commerciali + comunità grande, attiva e internazionale

https://bigdata-madesimple.com/what-is-3-ss-of-spark-and-its-effect-on-big-data/

Punti di forza: Velocità

- Ingegnerizzato dal basso per ottenere le migliori performance
- 100x più veloce di *Hadoop MapReduce* grazie all'elaborazione in memoria (e altre ottimizzazioni)
- Veloce anche su disco.
- Record del mondo nel 2014 e 2016 per ordinamento su disco



https://databricks.com/blog/2016/11/14/setting-new-world-record-apache-sp

Punti di forza: Facilità d'uso

- Approccio dichiarativo di alto livello ed API per agire su grandi datasets
- >100 operatori per trasformare data
- API basate su *DataFrame* e *Dataset* che permettono di manipolare facilmente dati strutturati e semi-strutturati

```
val errorCount = spark.read
      .text(fn)
      .filter($"value".like("%ERROR%"))
      .count()
```

Punti di forza: Facilità d'uso

WordCount in *MapReduce (Java)*

```java
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
          word.set(tokenizer.nextToken());
          output.collect(word, one);
        }
      }
    }
    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
      public void reduce(Text key, Iterator values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
          sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
      }
    }

    public static void main(String[] args) throws Exception {
      JobConf conf = new JobConf(WordCount.class);
      conf.setJobName("wordcount");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(Map.class);
      conf.setCombinerClass(Reduce.class);
      conf.setReducerClass(Reduce.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);
      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));
      JobClient.runJob(conf);
    }
}
```

WordCount in *Spark (Sc*

```scala
val file = spark.textFile("hdfs://...")
val counts = file
    .flatMap(line => line.split(" "))
    .map(word => (word, 1)).reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

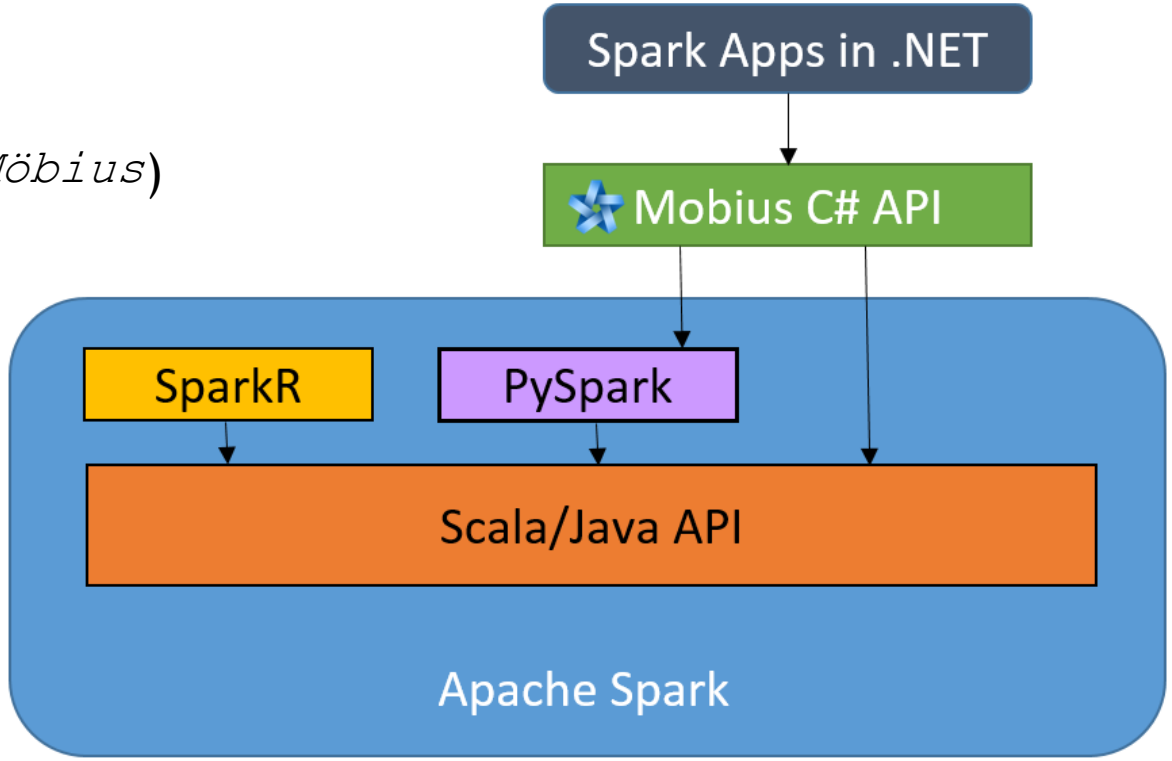
Punti di forza: Facilità d'uso

- Può essere eseguito interattivamente dalla shell o utilizzando un *NoteBook*
- Supporto per diversi linguaggi: **Scala, Java, Python, R, SQL**

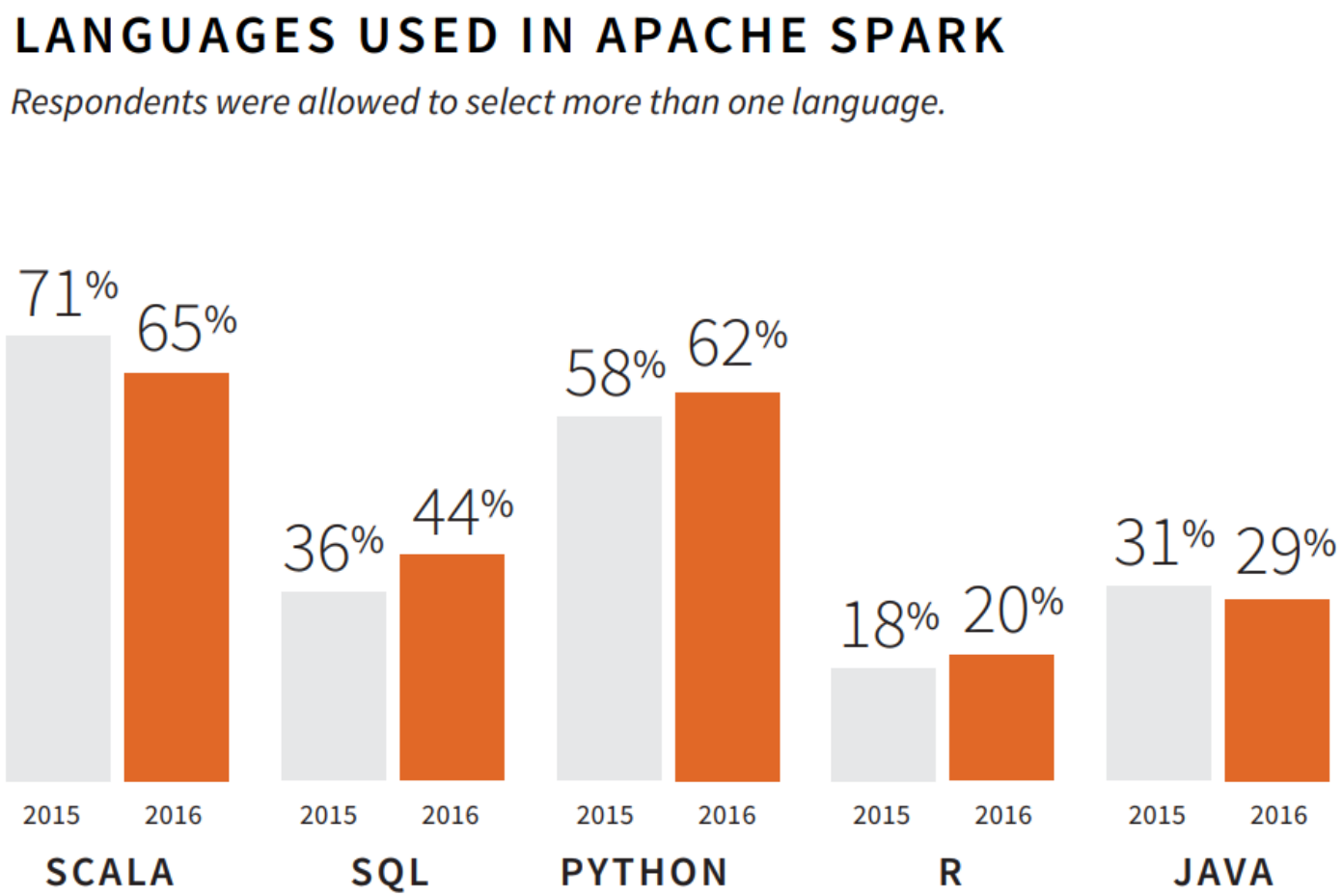
E stanno arrivando altri linguaggi!

(progetti di terze parti)

- *Clojure*
- *C#/.net* (progetto *Möbius*)
- *Groovy*
- *Julia*



Quale linguaggio usare?



http://pages.databricks.com/rs/094-YMS-629/images/2016_Spark_Infographic.pdf

Quale linguaggio usare?

Java

PRO:

- Potente e completo
- Molto diffuso, conosciuto e utilizzato: probabilmente il linguaggio più usato per i Big Data
- Alte prestazioni

CONTRO:

- Verboso
- Non supporta *Read-Evaluate-Print-Loop*

Quale linguaggio usare?

R

PRO:

- Il linguaggio di riferimento per Statistici e Analisti di Dati
- Buone librerie di visualizzazione

CONTRO:

- Non è considerato un linguaggio di utilizzo generale
- Prestazioni non alte
- C'è un certo ritardo nell'integrazione con Spark

Quale linguaggio usare?

SQL

PRO:

- Utilizzabile anche da non programmatori
- Trend di utilizzo in crescita

CONTRO:

- Inadatto per alcuni tipi di applicazioni

Quale linguaggio usare?

Python

PRO:

- Linguaggio molto noto e diffuso, anche nel mondo dei Big Data
- Facile da imparare
- Molto conciso ed espressivo
- Buone librerie di visualizzazione

CONTRO:

- Più lento
- «*Duck typing*». Controllo sui tipi a run-time

Quale linguaggio usare?

Scala

PRO:

- Spark è scritto in Scala: ogni feature del framework è immediatamente disponibile
- Compatibile con le librerie Java
- Controllo sui tipi durante la compilazione
- Alte prestazioni

CONTRO:

- Meno diffuso di Java o Python
- Marginalmente meno facile di Python

In questo corso useremo:


```
object HelloWorld {
        def main(args: Array[String]): Unit = {
                println("Hello, world!")
        }
}
```

Spark ha qualche punto debole

- Non ha un suo sistema di storage
- Grande consumo di risorse, in particolare la memoria ⇒ è dispendioso
- Problemi con dataset molto grandi
- Non ha supporto per il *real-time processing* (*Spark Streaming* usa micro-batches)
- Tende a generare tanti file relativamente piccoli. Questo è un male, ad es. per *Hadoop HDFS* o *Amazon S3*.
- Può richiedere ottimizzazione manuale
- Ha una latenza considerevole

Non solo Apache Spark!

Alcune alternative:

(creato da Twitter)

Apache Flink

TIBCO®

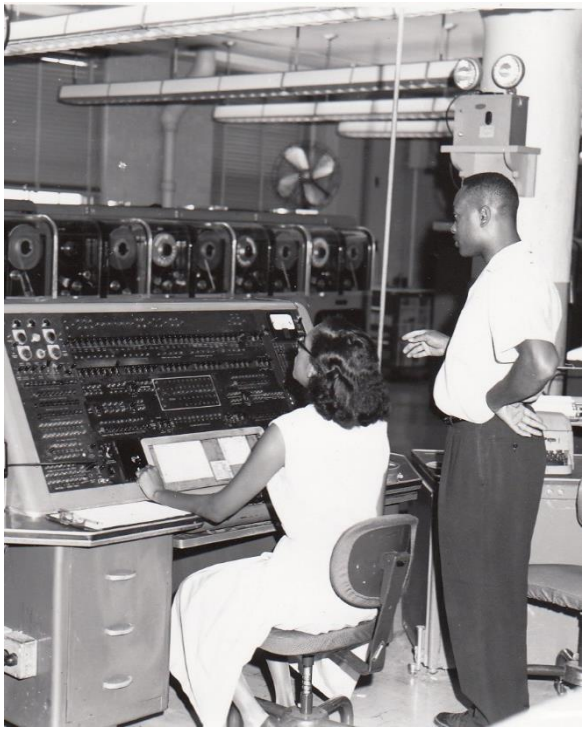
StreamBase

IBM InfoSphere Streams

Il contesto (e un po' di storia)

Contesto

La potenza di calcolo ha continuato a crescere ininterrottamente per ottant'anni



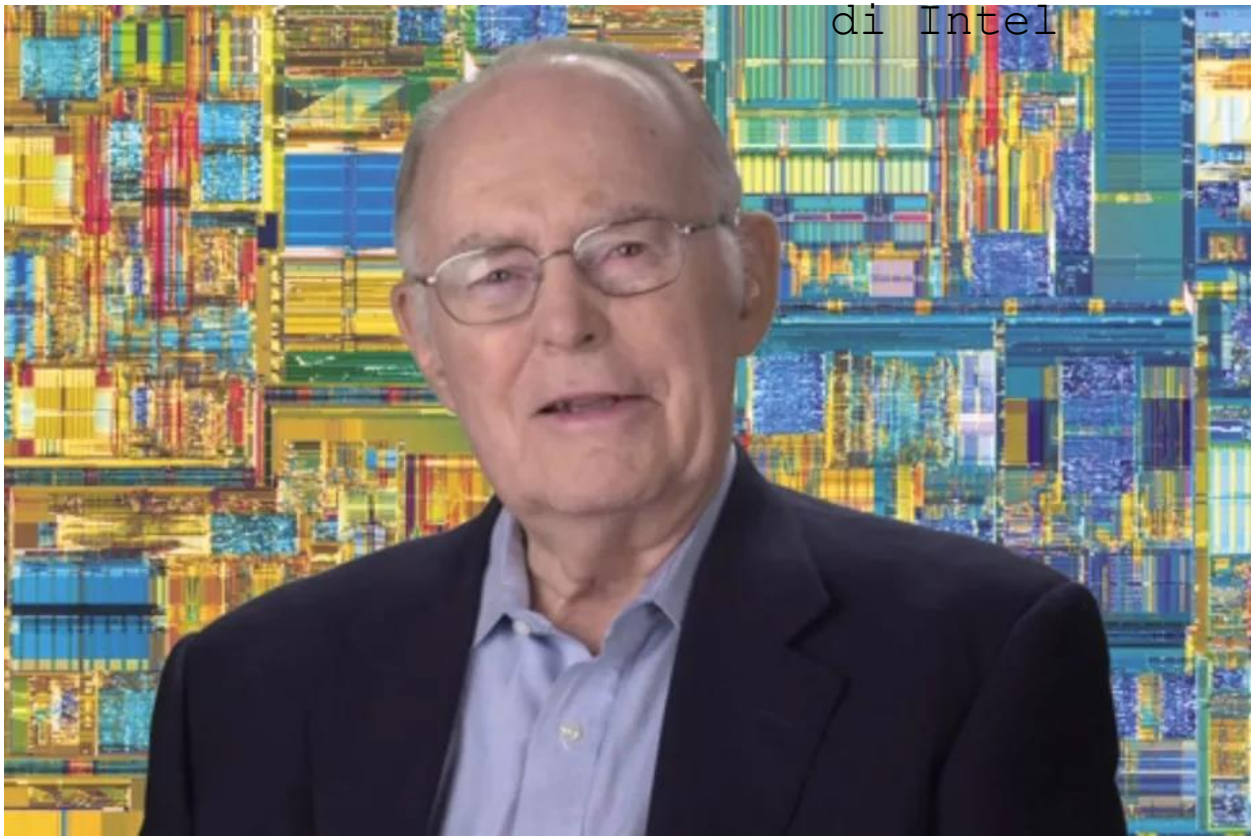
Univac I (1951)
~ 1 000 000$
1905 operazioni al secondo

Cray-1 (1976)
~ 10 000 000$, 115 Kw
1.9 GFLOPS

iPad 2 (2011)
<1 000$
1.6 GFLOPS

La legge di Moore

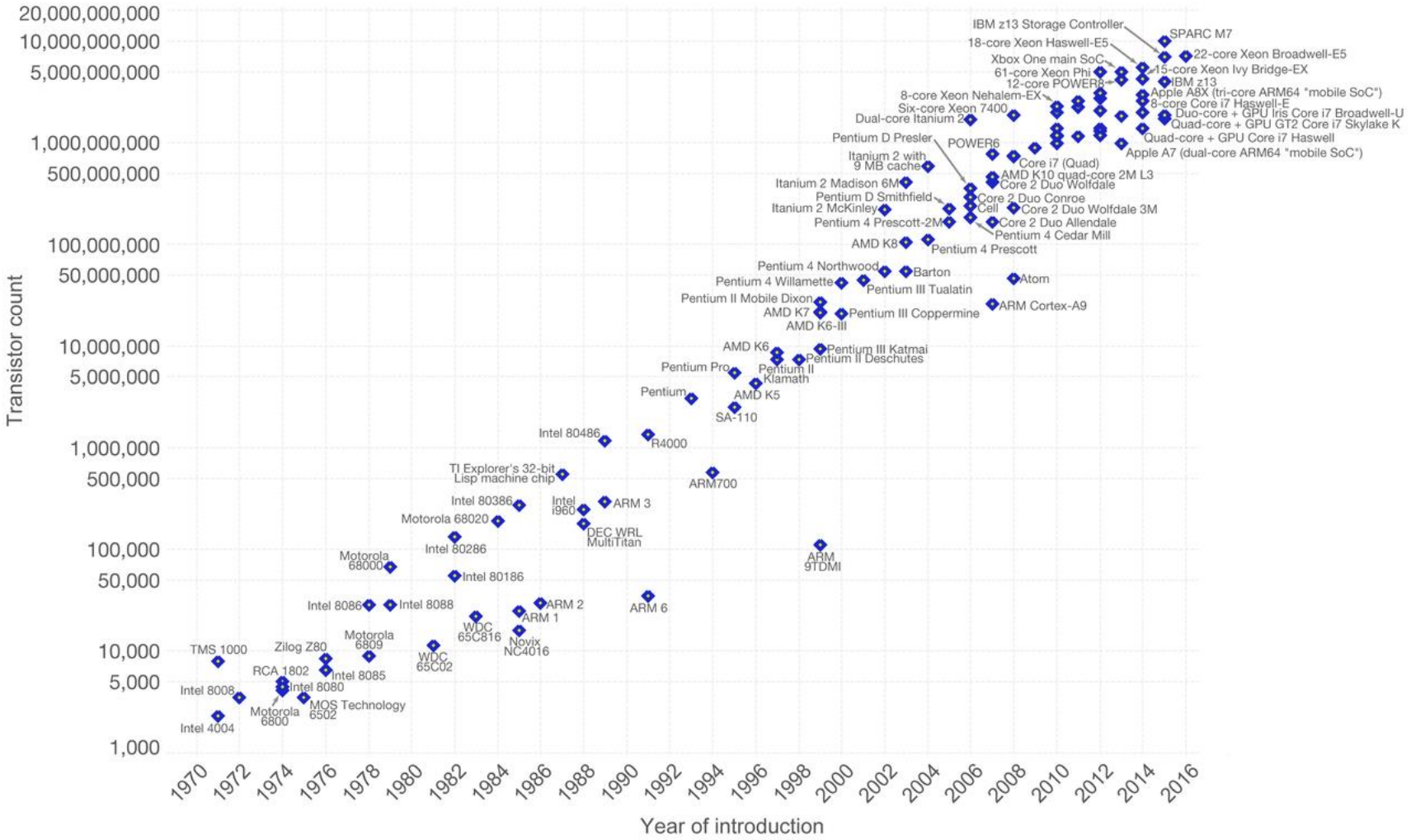
Gordon Earle Moore, nato nel 1929 cofondatore di Intel



Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

La legge di Moore è ancora valida?

DEATH NOTICE

Long beloved by both engineers and computer scientists because of ongoing performance benefits ceaselessly and seemingly effortlessly achieved. From the age of fifty, Moore's Law began to age rapidly, passing into senescence and then, at the beginning of this month, into oblivion. Moore's Law leaves a thriving multi-trillion dollar global semiconductor and electronics industry, along with a growing set of questions about how that industry will survive its passing. In lieu of flowers, donations should be lavished on Intel shares.

https://www.theregister.co.uk/2018/01/24/death_notice_fo

Un'altra strategia per andare veloci: il parallelismo

Largest Amazon EC2 instance: 128 virtual CPUs

Azul Systems Vega 3
Cores per chip: 54
Cores per system: 864

The Future is Parallel

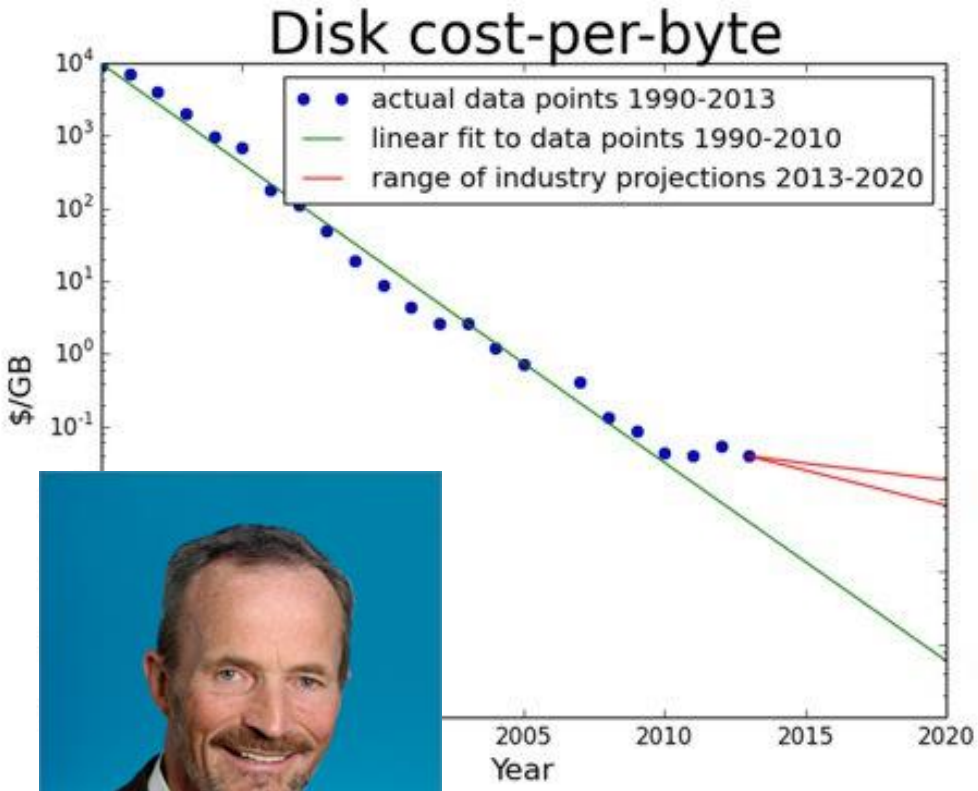
Intel Xeon
24 cores
48 threads

AMD Opteron
16 cores

Tilera Gx-3000
100 cores

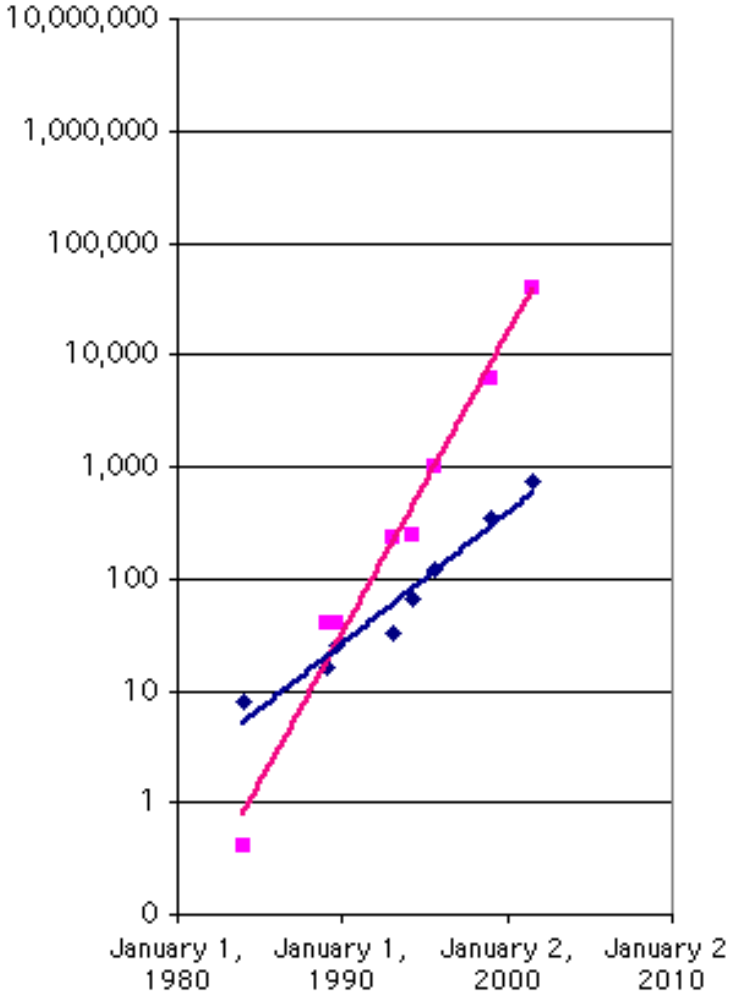
La legge di Kryder

Le prestazioni dei dischi crescono più velocemente della potenza di calcolo

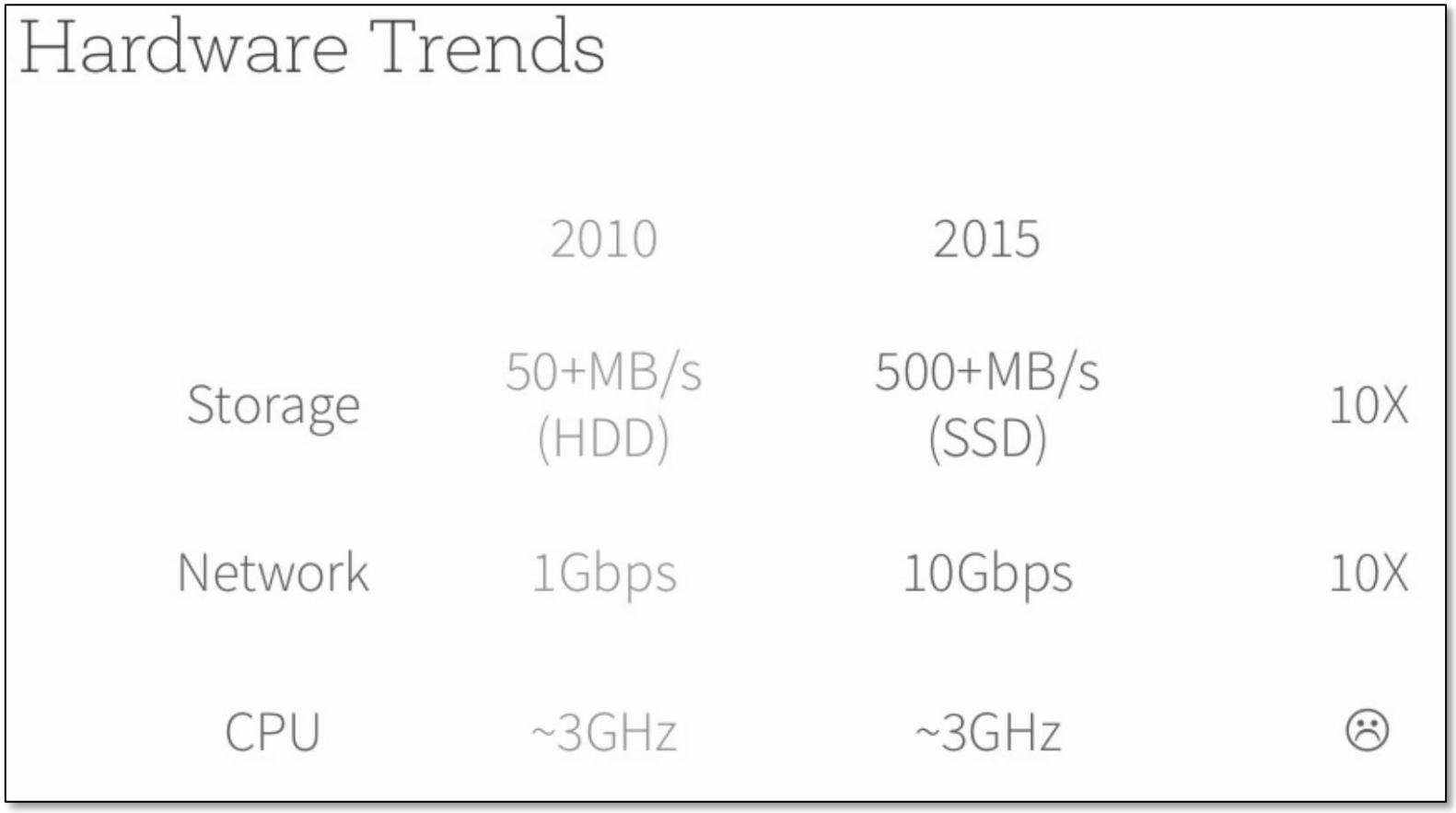


Mark H. Kryder

Legge di Kryder (viola)
vs.
Legge di Moore (blu)



Le prestazioni dei dischi crescono più velocemente della potenza di calcolo

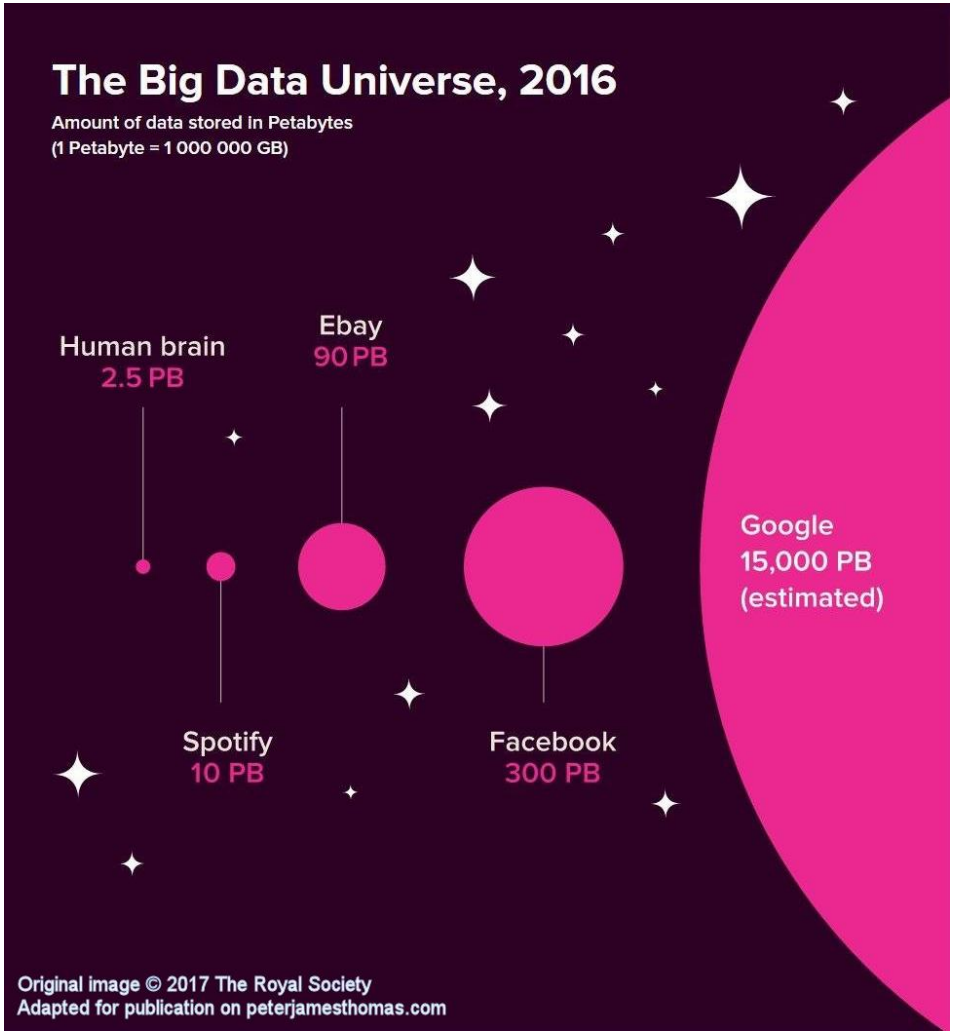


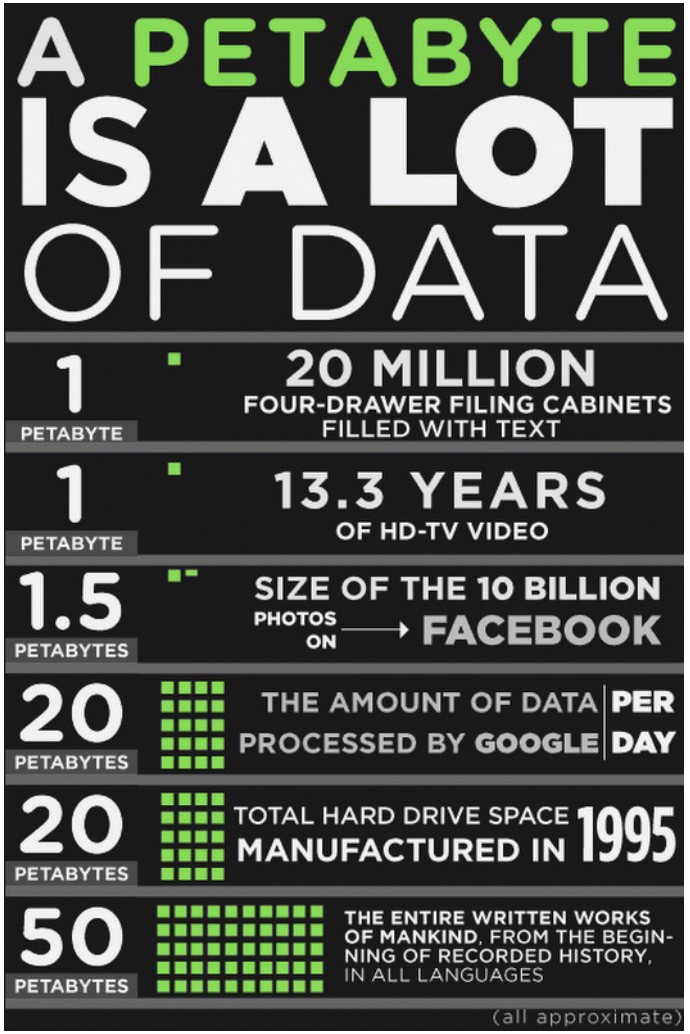
L'avvento dei BIG Data

BIG DATA:

- 500 milioni di tweets al giorno
- Traffico di quasi 2 GB/mese sui telefoni cellulari
- Amazon vende 600 prodotti al secondo
- 200 miliardi di email vengono scambiate ogni giorno
- MasterCard esegue 74 miliardi di transazioni ogni anno
- Ci sono quasi 6000 voli di linea al giorno.

Quanto sono «*BIG*» i Big Data?





Intermezzo : aspetta… quanti dati ha Google???

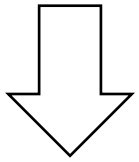
"If all digital data were stored on punch cards, how big would Google's data warehouse be?"

Randall Munroe

https://blog.ted.com/using-serious-math-to-answer-weird-questions-randall-munroe-at-ted2014/

Riassumendo

- La velocità di calcolo cresce, ma non tanto quanto vorremmo
- Lo storage cresce molto più velocemente
- Con uno storage economico e virtualmente illimitato la quantità di da



Il modo più efficace di analizzare questa enorme massa di dati è il *clus*

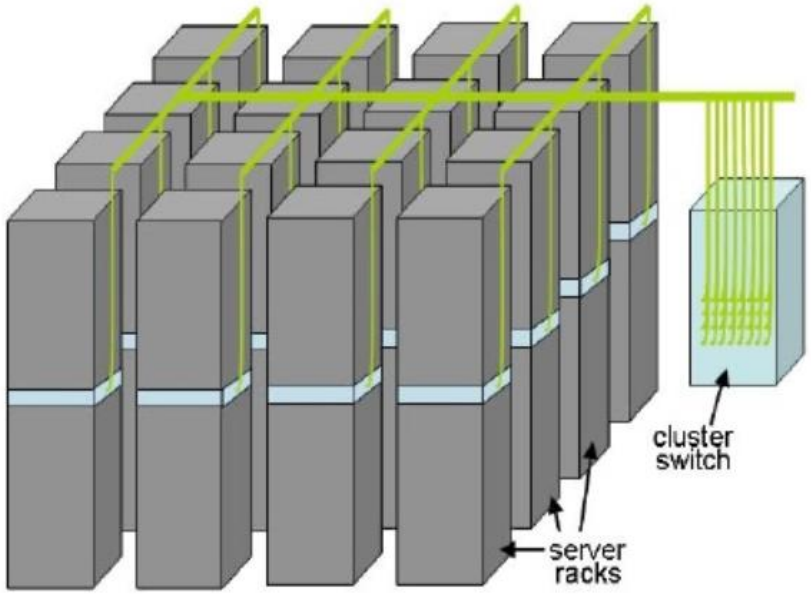
Esistono cluster con centinaia e anche migliaia di nodi!

Come si programma questa roba?

Come si programma questa roba?

Servers
- CPUs
- DRAM
- Disks

Racks
- 40-80 servers
- Ethernet switch



Clusters

Nel primo decennio del secolo Google pubblica tre articoli fondamentali

2003
GFS - Google File System

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google

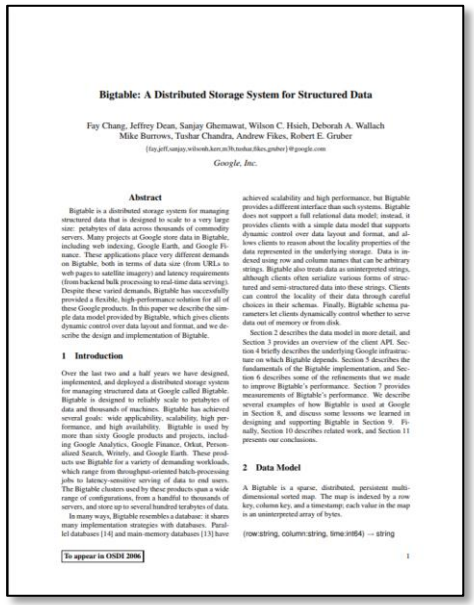
2004
MapReduce

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

Google, Inc.

2006
BigTable



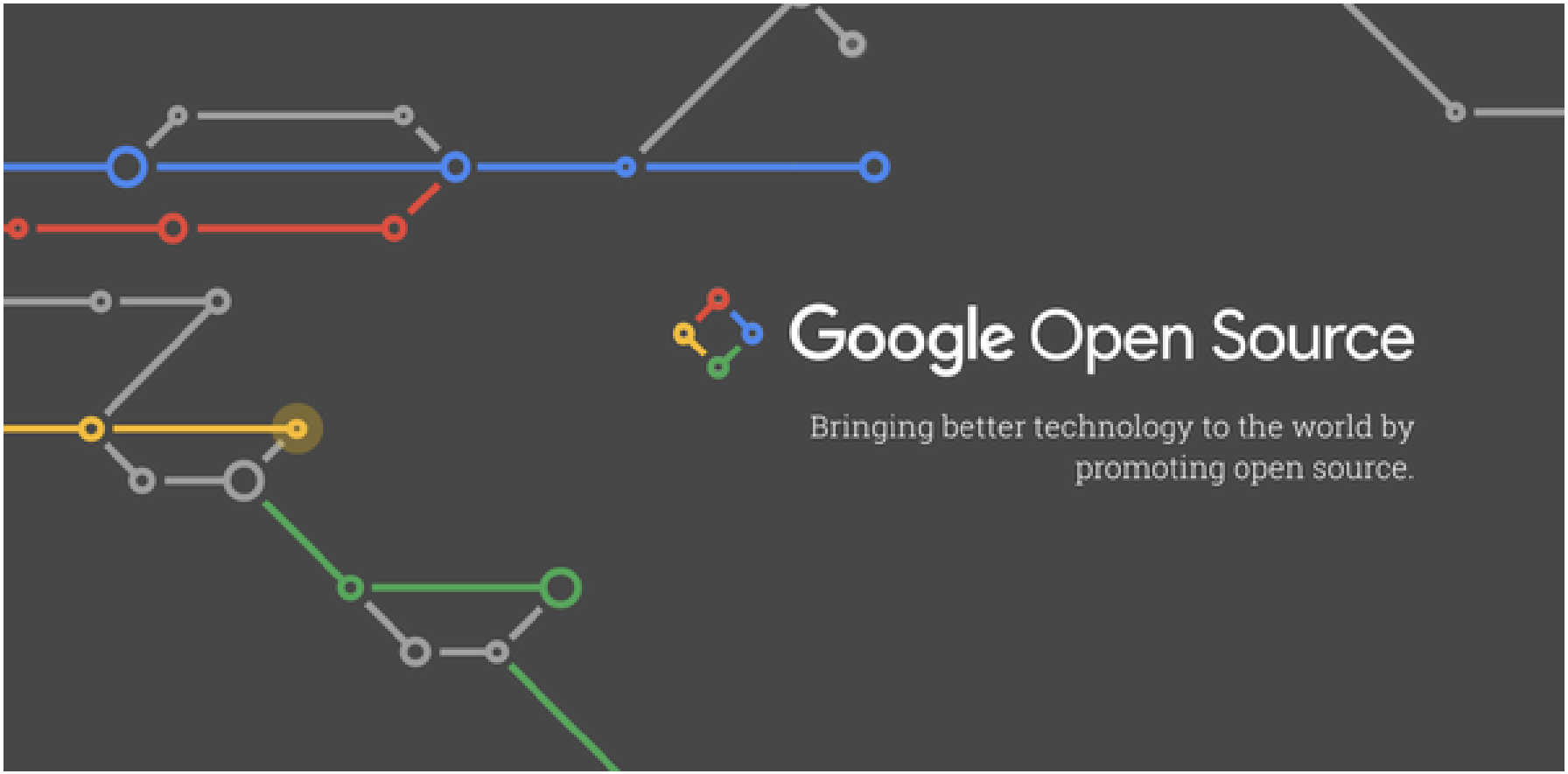
Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

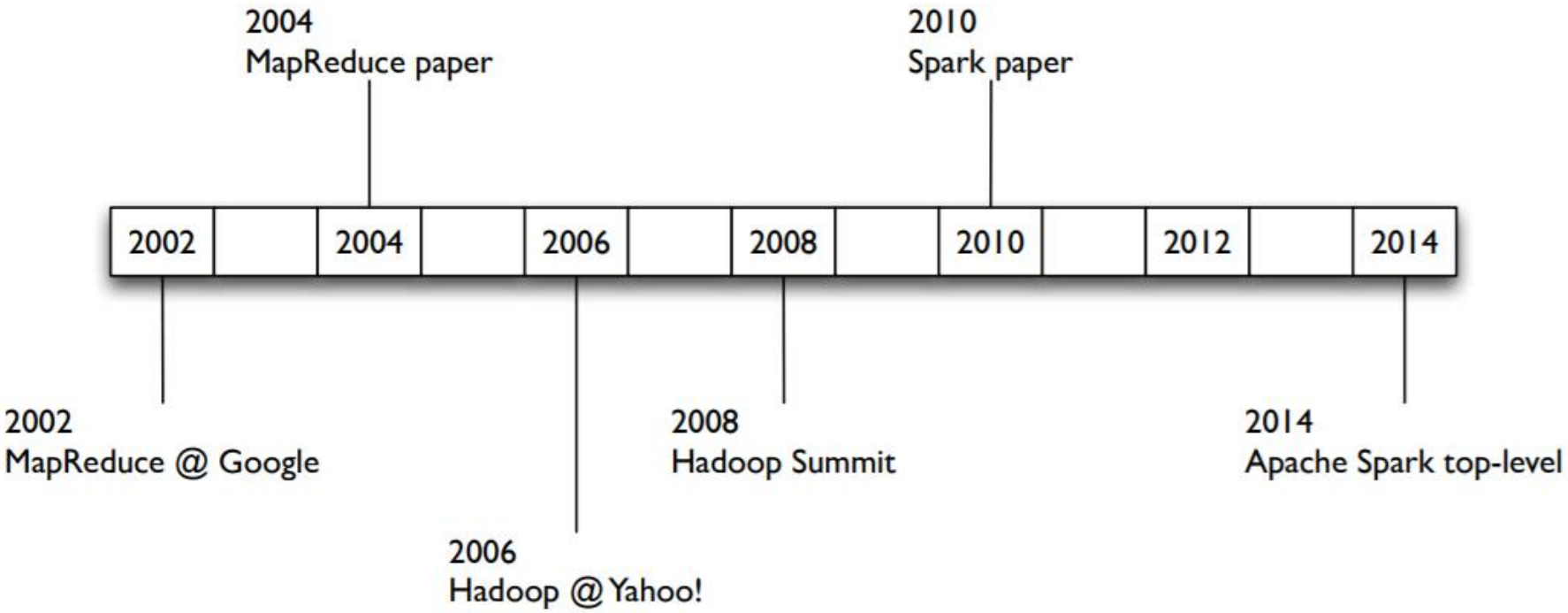
Google, Inc.

ma non pubblica il codice dell'imple

… poi (un po' in ritardo) abbraccia l'Open Source



Nel frattempo nascono Hadoop e Spark



Subito prima della nascita di Spark

Hadoop è la soluzione di riferimento per i Big Data:

- HDFS per lo storage e
- MapReduce per il calcolo

MapReduce ha delle limitazioni: è relativamente lento e difficile da programmare.

Inoltre gli utenti cominciano a richiedere:

- Algoritmi più complessi, basati su passi multipli
- Query ad-hoc interattive
- Processing in real time

Queste feature sono difficilmente implementabili in MapReduce

Subito prima della nascita di Spark

Per ovviare a queste limitazioni vengono sviluppati dei sistemi *specializzati*, uno per ogni tipo di esigenza:

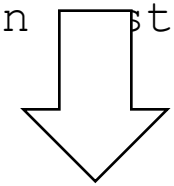
- Pregel
- Giraph
- Dremel
- Drill
- Impala
- Presto
- Storm
- S4
- ….

Subito prima della nascita di Spark

Problemi con i sistemi specializzati

- Molti sistemi da controllare, configurare e installare

- Difficile combinarli fra loro anche se spesso servirebbe (ad es. leggere dati con SQL e poi usare degli algoritmi di machine learning)
  ⇒ In molti casi il trasferimento dei dati fra due sistemi diventa un costo dominante



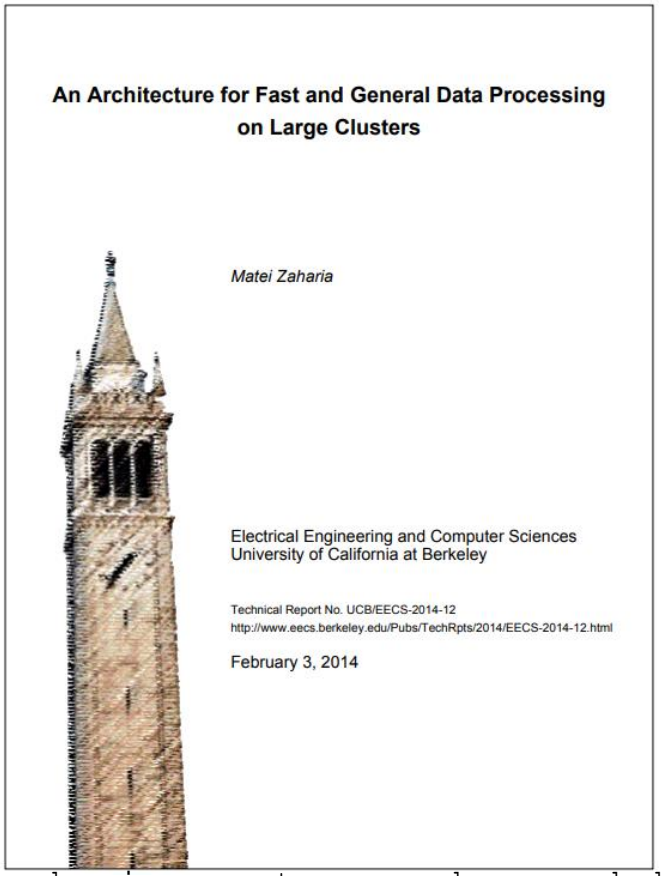
Spark nasce come un engine unificato

2009 : Nascita di Spark

Un gruppo di ricerca a UC Berkeley

Il gruppo lavora su un framework per il controllo di cluster che possa sostenere diversi tipi di cluster computing system. Spark nasce come esempio.

All'inizio focalizzato sul *machine learning*, poi diventa uno strumento generale



An Architecture for Fast and General Data Processing on Large Clusters

Matei Zaharia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-12
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-12.html

February 3, 2014

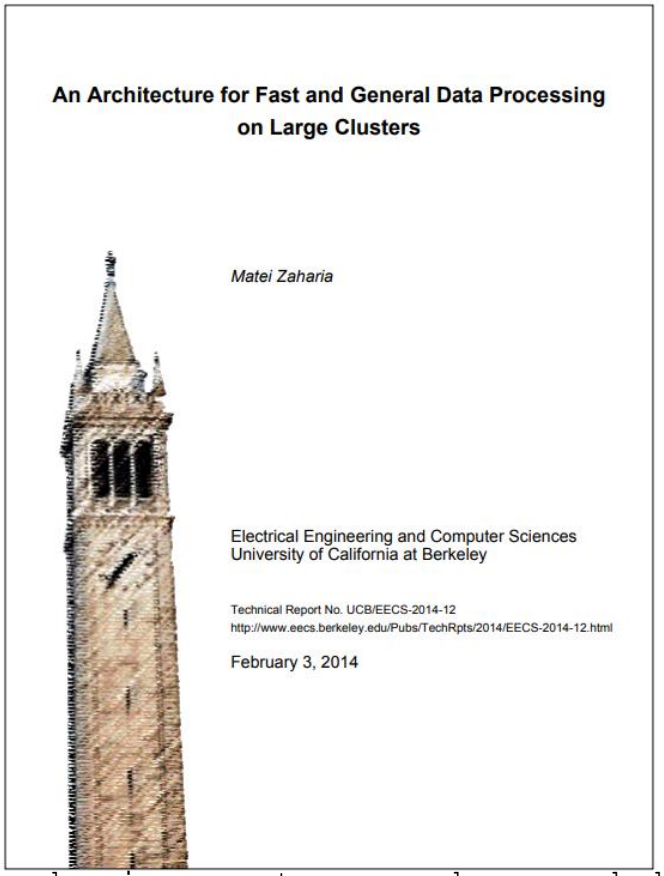
https://www.kdnuggets.com/2015/05/interview-matei-zaharia-creator-apache-spark.htm

2009 : Nascita di Spark

Un gruppo di ricerca a UC Berkeley

Due idee chiave:

- Tenere i risultati intermedi in memoria
- Fare uno strumento generale che possa essere utilizzato in contesti differenti (es. streaming e graph processing)



Nel 2013 si muove in Apache

Nel 2014 diventa un *Top-Level Project* (TLP)

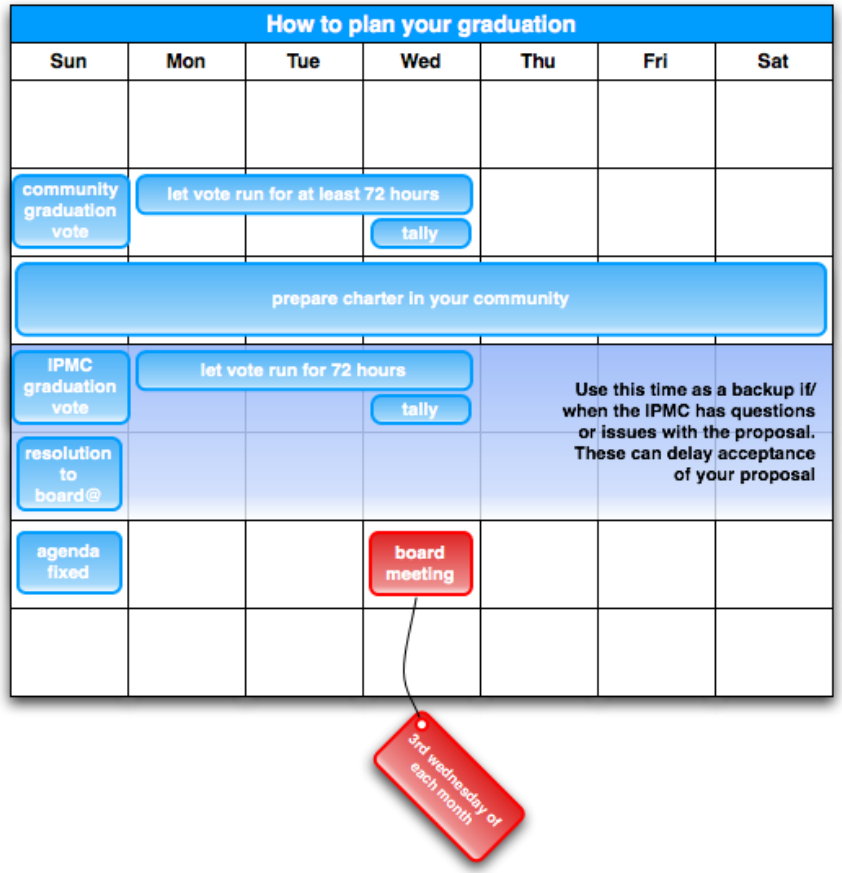
https://www.kdnuggets.com/2015/05/interview-matei-zaharia-creator-apache-spark.htm

2014: Spark è Apache Top Level Project

APACHE INCUBATOR

Diventare TPL è un processo complicato e laborioso.

Quando un progetto diventa TLP può contare su una comunità grande e attiva e su una grande visibilità



Apache is a well recognized brand

GOVERNOR ARNOLD SCHWARZENEGGER

November 5, 2009

Apache Software Foundation

It is a great pleasure to extend my greetings to all those attending ApacheCon and congratulations on your tenth anniversary.

I applaud your incredible work over the past decade and appreciate you choosing California as the place to celebrate this fantastic milestone. Our state is a land of innovation, and you have likewise fostered great technological advancements that have touched the lives of millions of people around the world.

Whether managing financial systems, positioning satellites or powering websites through the Apache HTTP Server, your open source projects play key roles in making our information age possible. Thank you for your extraordinary accomplishments and commitment to discovery.

On behalf of all Californians, I send my gratitude to everyone in attendance for your participation, and I offer my best wishes for a rewarding conference and continued success.

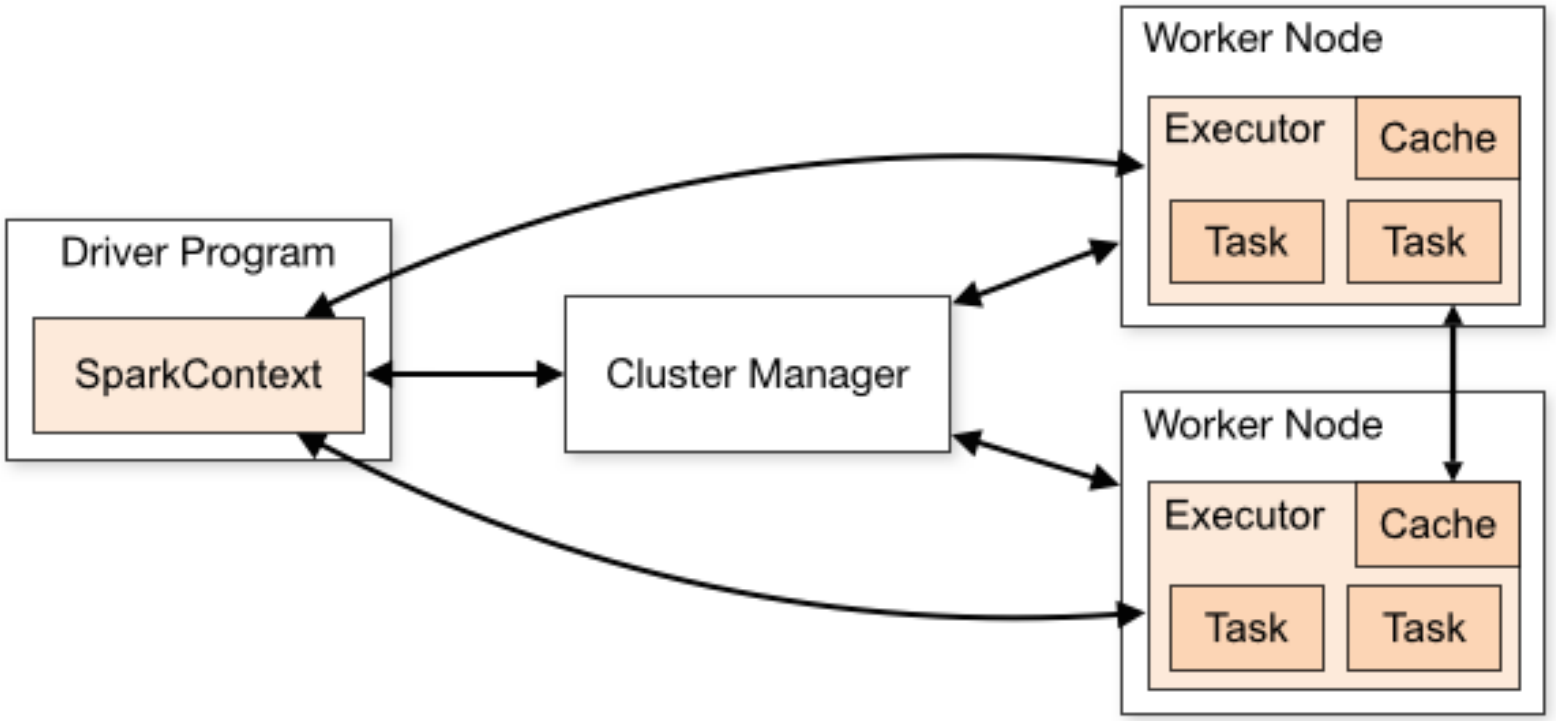
Sincerely,

Arnold Schwarzenegger

Spark,
com’è fatto?

Struttura di un programma Spark

L'elaborazione è distribuita su più nodi

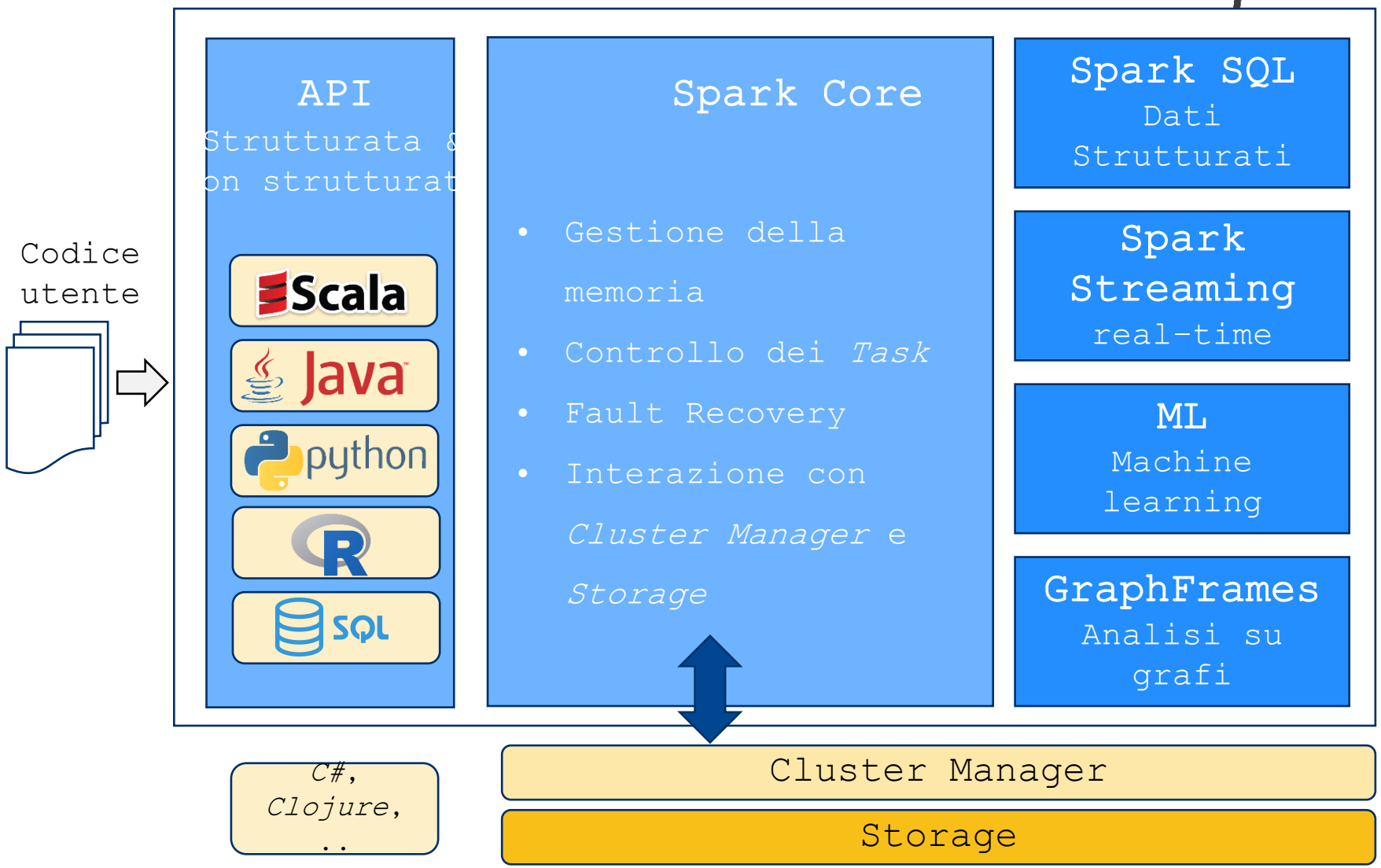


L'ecosistema di Spark

Cluster manager

Storage

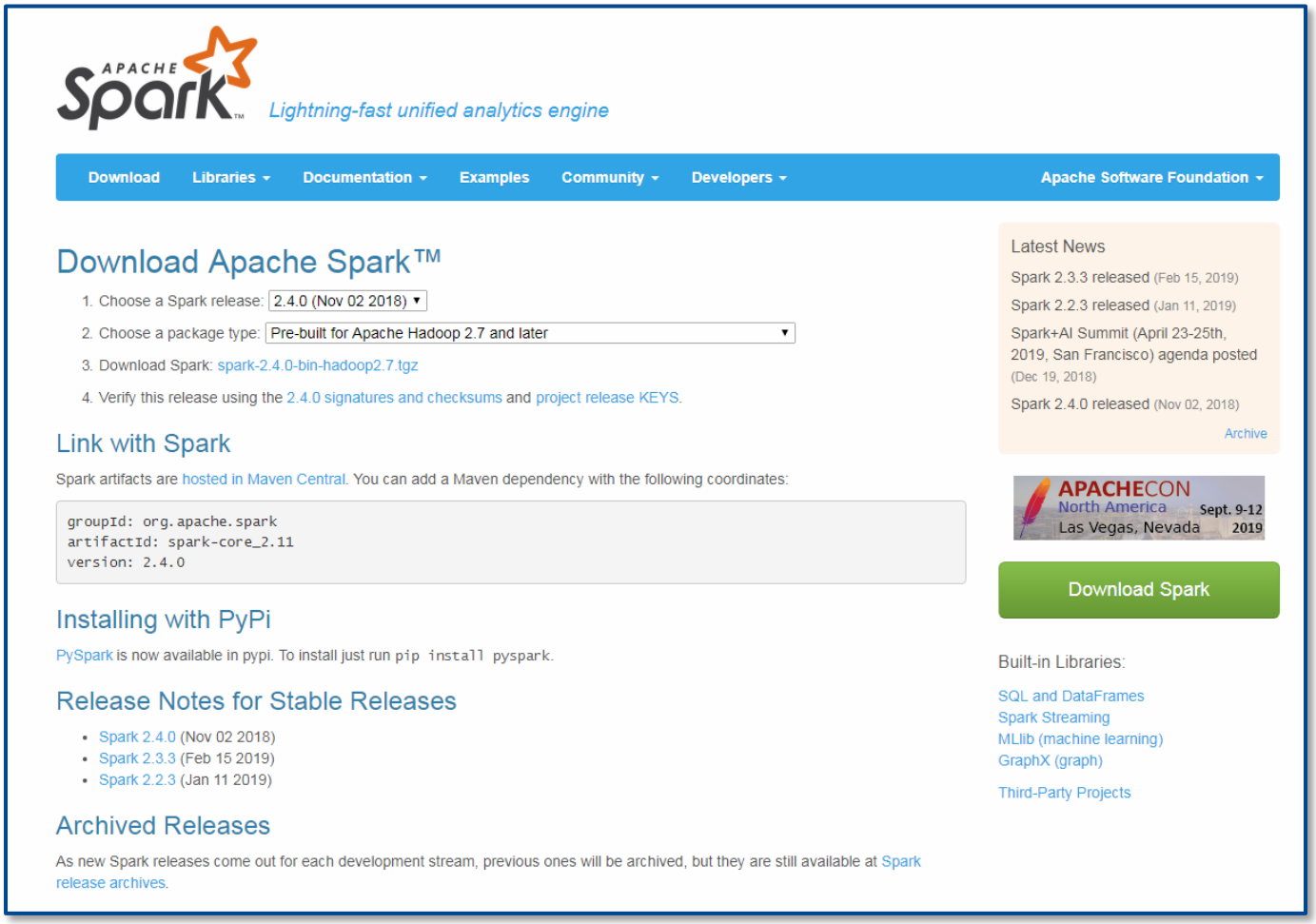
L'ecosistema di Spark



In pratica?

Per fare degli esperimenti a casa

https://spark.apache.org/downloa



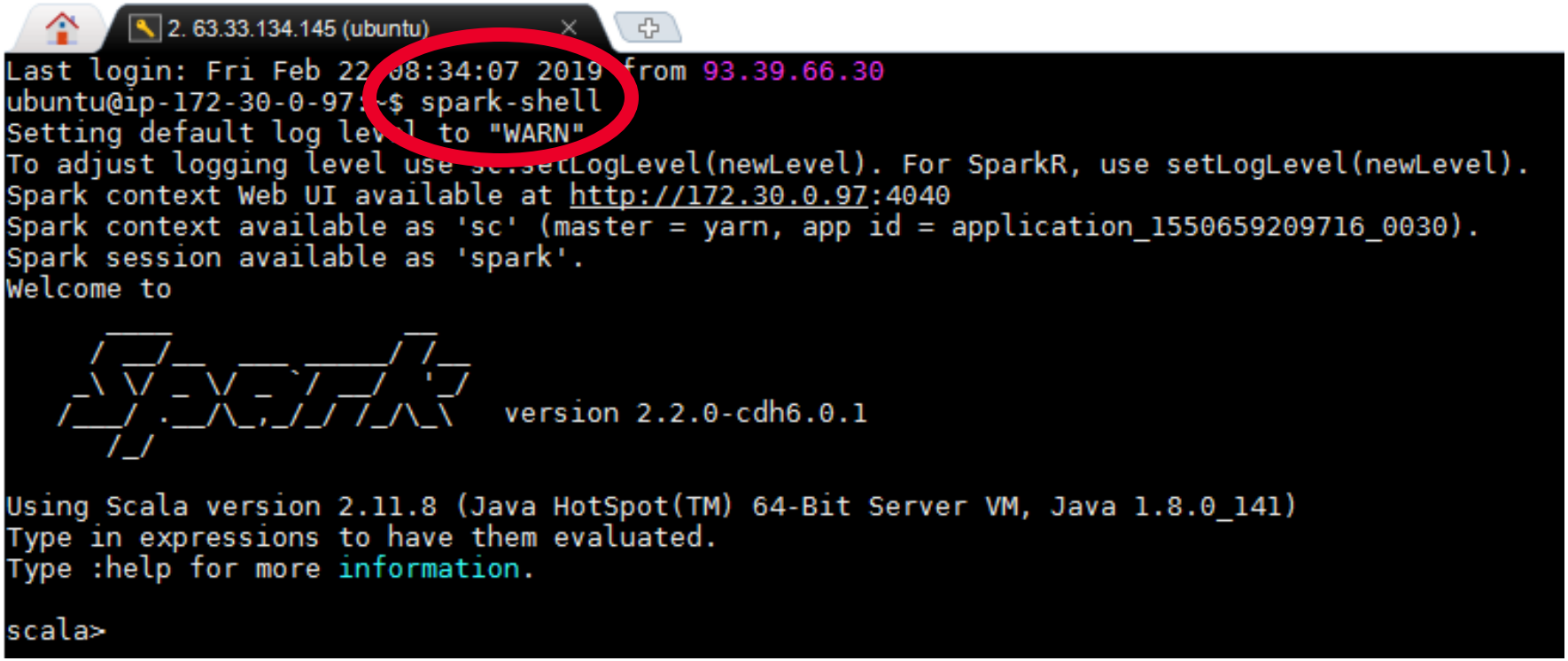
Per fare degli esperimenti a casa

Spark gira anche su un singolo computer

Lo stesso codice funziona su un cluster di 1000 nodi!

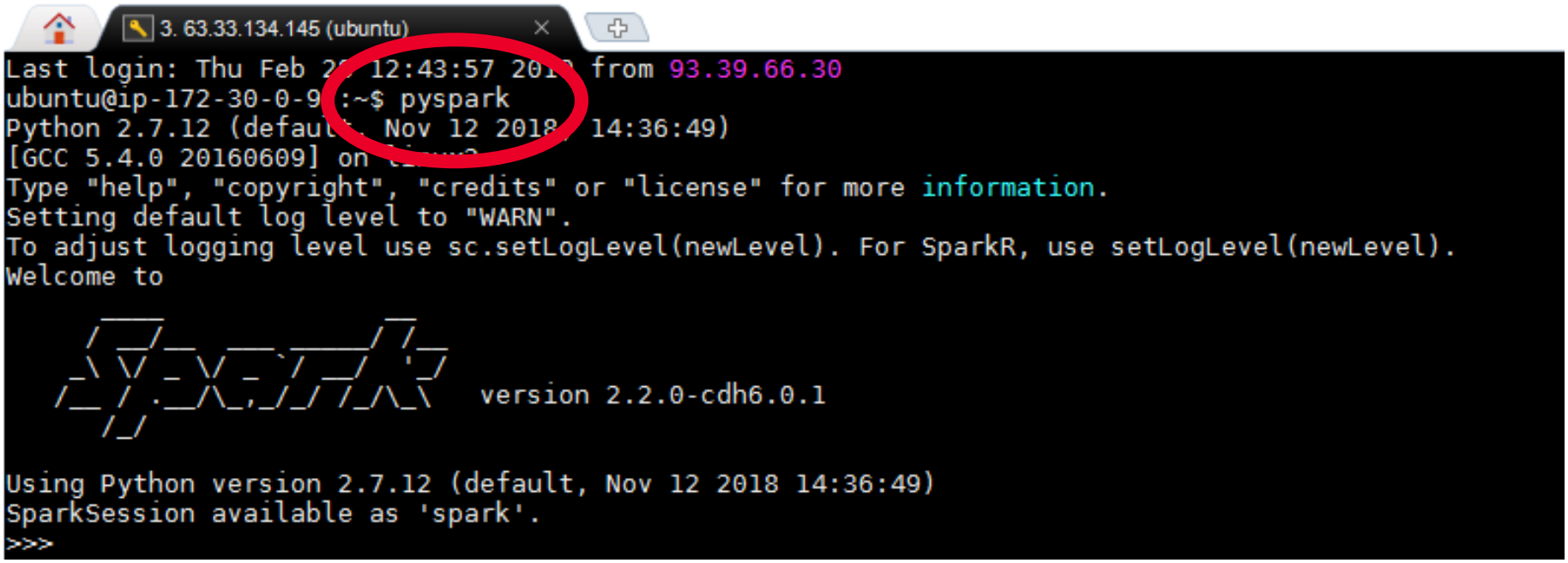
Come useremo Spark nel laboratorio di oggi?

La Shell di Spark



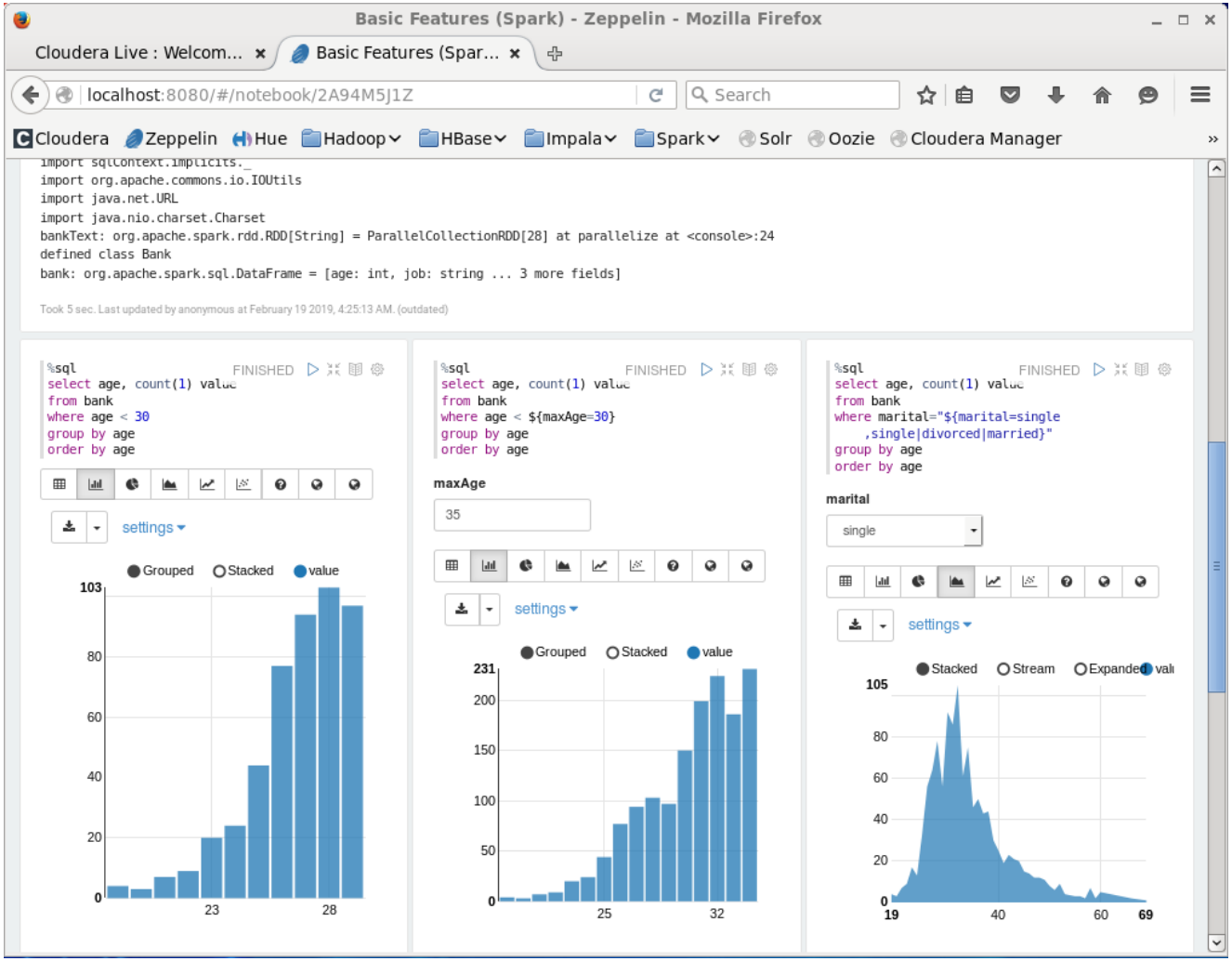
Come useremo Spark nel laboratorio di oggi?

Si può usare anche Python (pyspark)



Come useremo Spark nel laboratorio di oggi?

Zeppelin Notebook



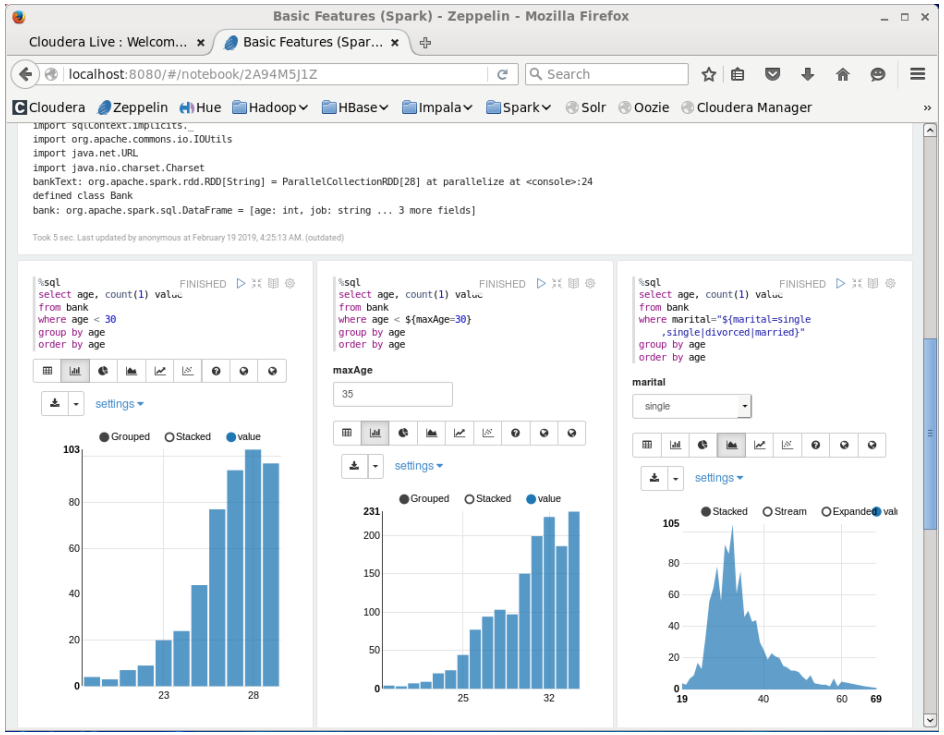
I Notebook

Un ambiente per il «literate programming»

Un'interfaccia che permette di mescolare frammenti eseguibili di programma con grafici, testo e altri contenuti.

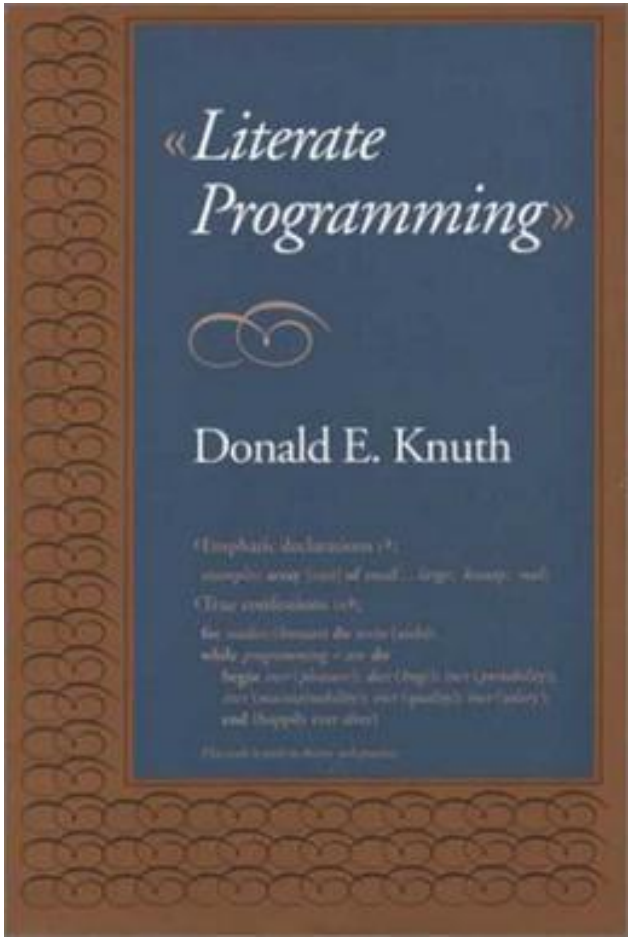
I notebook per Apache Spark

- *Jupyter Notebook*
- *Apache Zeppelin*



Literate programming

Donald E. Knuth



Laboratorio

Obiettivi per questa mattina

- Ripasso di Scala
- Prendere familiarità con *Zeppelin Notebook*

Scala Cheat Sheet

variances

Given T <: P
When C is covariant then C[T] <: C[P]
When C is contravariant then C[T] >: C[P]

class C[+T]	C is covariant
class C[-T]	C is contravariant
class C[T]	C is invariant

control constructs

if (condition) this else that	if then else
while (x < 5) { println(x); x += 1}	while loop
do { println(x); x += 1} while (x < 5)	do while loop
for (x <- xs if x%2 == 0) yield x*10	for comprehension
for (i <- 1 to 5) { println(i) }	for loop

variables and functions

val x = 1	constant, evaluated immediately
var y = 2	variable
def z = 2	evaluated when called
lazy val w = 2	evaluated once when needed
def f(x: Int) = { x*x }	define function
def f(x: R)	call by value
def f(x: => R)	call by name (lazy parameters)
(1 to 5).map(_*2)	anonymous function (positionally matched arguments)
(1 to 5).map(2*)	anonymous function (bound infix method)
(1 to 5).map(x => x*x)	anonymous function (to use an arg twice, have to name it)
def mapmake[T](g:T=>T)(seq: List[T]) = seq.map(g)	generic type
def sum(args: Int*) = args.reduceLeft(_+_)	varargs

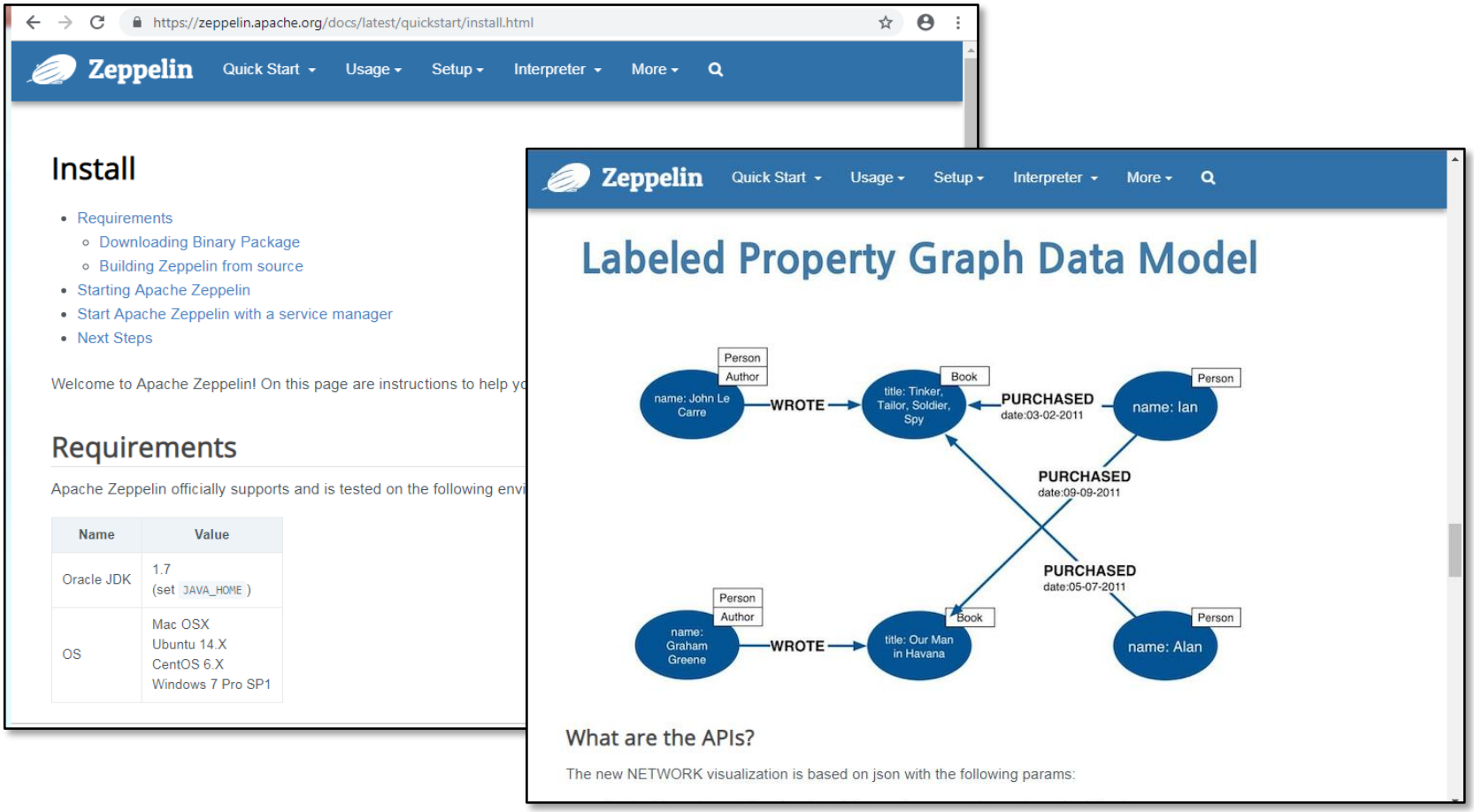
object orientation

class C(x: R)	class with private param
class C(val x: R)	class with public param
new { ... }	anonymous class
abstract class D { ... }	abstract class
class C extends D { ... }	inheritance
class C(x: R) extends D(x)	inheritance and constructor params
object O extends D { ... }	singleton
trait T { ... }	interfaces with implementation
class C extends D with T	using multiple traits

gutefrage.net

Zeppelin Notebook

Manuale online



https://zeppelin.apache.org/

KEEP CALM AND START CODING