Università degli Studi di Palermo

dipartimento di ingegneria unipa

Spark

Corso di Big Data

a.a. 2022/2023

Prof. Roberto Pirrone

Laboratorio di Interazione Uomo-Macchina
CHILAB

Sommario

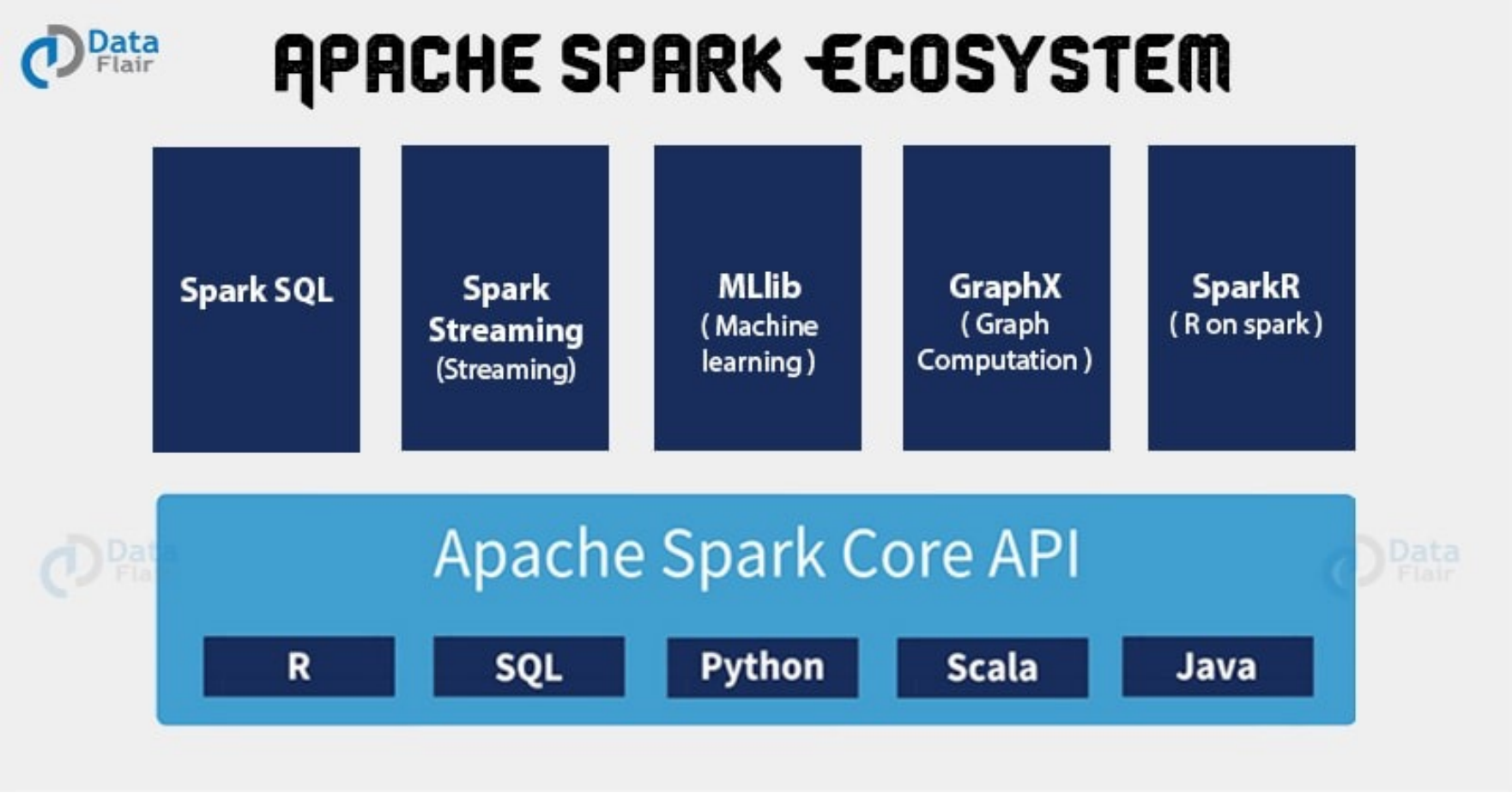
- Ecosistema Spark
- Architettura di Spark
- Introduzione a PySpark

Immagini e diagrammi da: [data-flair.training]

Ecosistema Spark

- Aapache Spark è un engine di calcolo su cluster di machine che si complementa con Hadoop, ma può essere anche utilizzato da solo
- Caratterizzato dalla capacità di eseguire processing di dati streaming
- Fornisce librerie dedicate ai diversi aspetti dell'analisi big data
- Possiede API di alto/basso livello rispetto all'astrazione sui dati in SQL, Java, Python, Scala e R

Ecosistema Spark



Ecosistema Spark

- Spark Core
  - Gestisce il flusso di I/O e la distribuzione del carico sugli esecutori
  - Fault recovery
  - Esegue i passi di computazione
    - Trasformazioni
    - Azioni
  - In memory computation

Ecosistema Spark

- Spark Core
  - Utilizza il *Resilient Distributed Dataset (RDD)* come modello dei dati che è l'unità base su cui si effettuano trasformazioni e azioni
    - Rappresentazione di basso livello
  - RDD: collezione immutabile dei dati in forma di record, distribuita sui nodi del cluster che consente la suddivisione «logica» in partizioni le quali sono oggetti mutable
    - Data lineage: Spark mantiene il flusso delle operazioni sui dati in forma di un DAG per ricostruire un RDD a partire dai dati o altri RDD che lo hanno generato in caso di fallimento

Ecosistema Spark

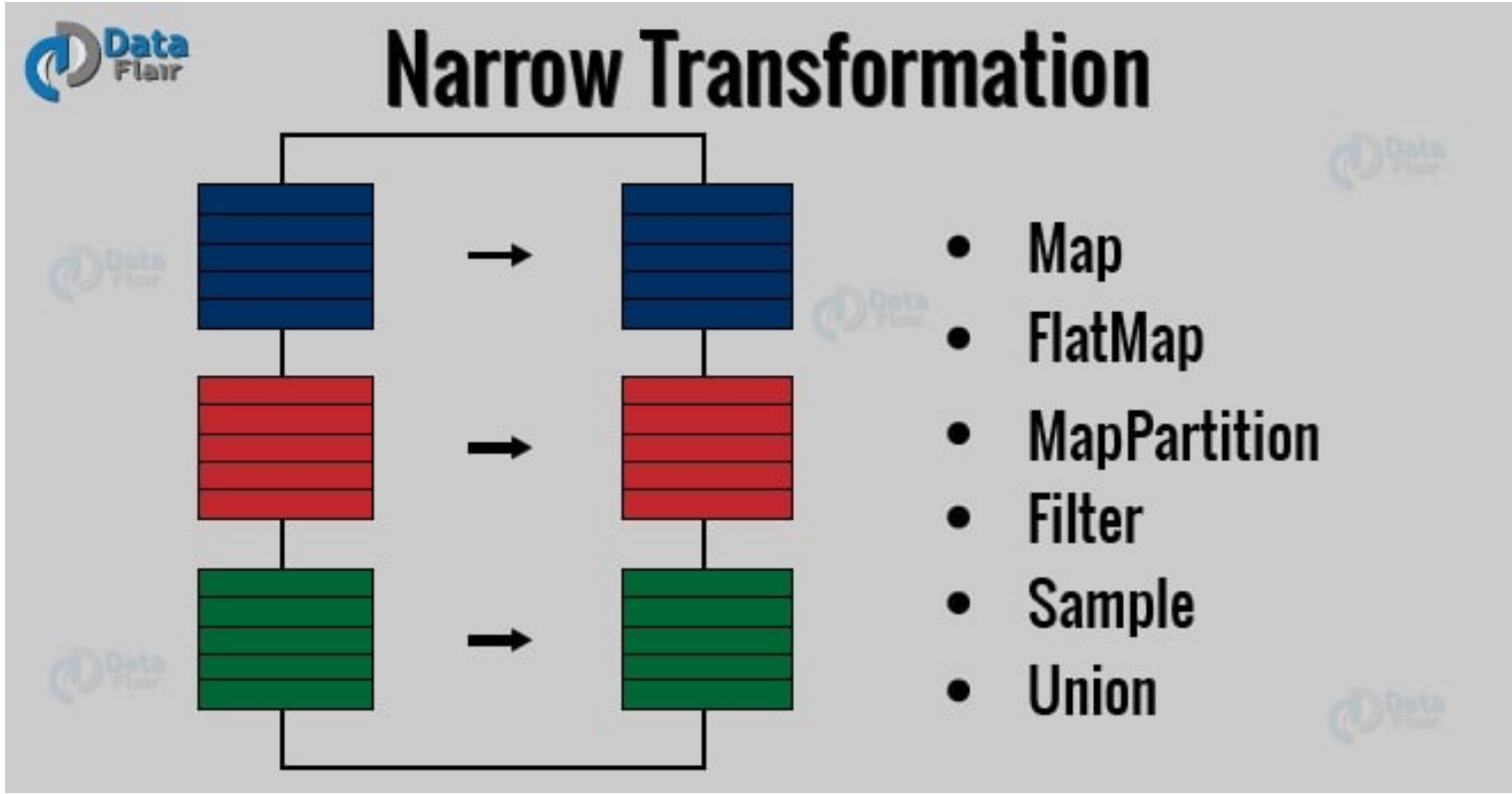
- Spark Core
  - Lazy Evaluation
    - I dati vengono sottoposti ad una sequenza di trasformazioni seguita da una azione
    - L’elaborazione avviene solo quando è necessario eseguire l’azione

Ecosistema Spark

- Spark Core
  - Trasformazioni
    - Trasformano un RDD in un altro senza modificarlo
    - Non vengono eseguite subito
    - Possono essere eseguite in cascata (pipelining)

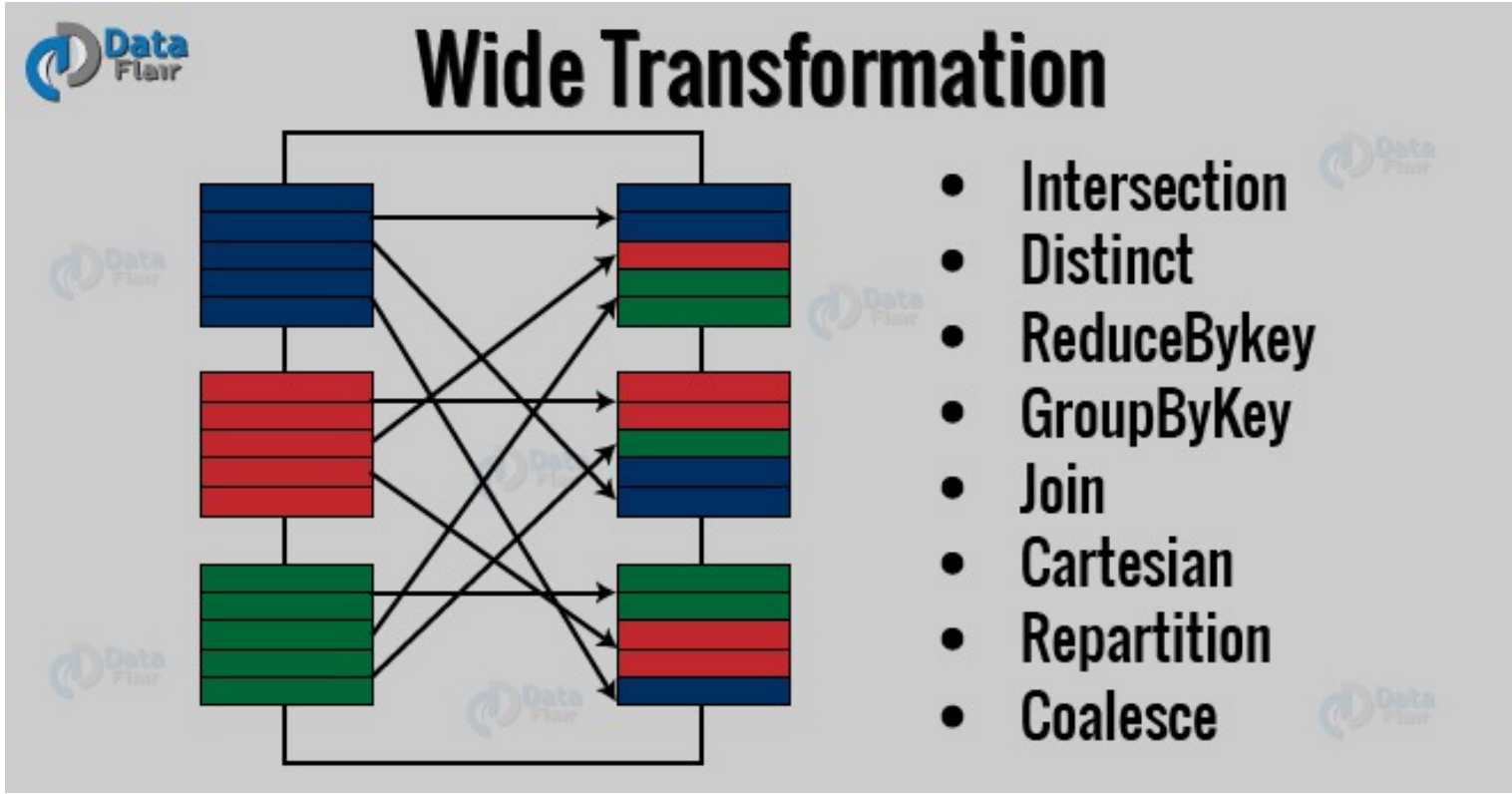
Ecosistema Spark

- Spark Core
  - Trasformazioni «narrow»
  - Non c’è bisogno di accedere a partizioni diverse da quella in cui si trova il dato
  - Eseguite in pielining



Ecosistema Spark

- Spark Core
  - Trasformazioni «wide» o «shuffle»
  - I dati possono trovarsi in partizioni diverse
  - Può essere necessario uno shuffling esplicito



Ecosistema Spark

- Spark Core
  - Azioni
    - Operazioni che *non producono* un RDD
    - Materializzano il risultato perché innescano tutto il flusso delle operazioni prescritte all'interno del garfo del RDD lineage, a partire dal caricamento dei dati fino alla fine
    - E' il momento in cui gli esecutori sui nodi restituiscono i risultati al driver dell'applicazione
    - Esempi: `first()`, `take()`, `reduce()`, `collect()`, `count()`

Ecosistema Spark

- Spark SQL
  - Orientato ai dati strutturati
  - Utilizza un linguaggio SQL like per eseguire query distribuite che vengono ottimizzate attraverso Spark Core
  - Connettività JDB/ODBC
  - Pieno interfacciamento per batch processing con Hive
  - Si basa sul concetto di *Dataframe* che è l'equivalente della tabella relazionale in questo contesto
    - Rappresentazione dei dati di alto livello *che astrae* il RDD

Ecosistema Spark

- Spark Streaming
  - Elaborazione dati streaming
  - Si interfaccia con diverse sorgenti quali Kafka, Flume, ma legge anche direttamente da socket TCP
  - Gestisce un proprio contesto operativo particolare rispetto a quello base definito in Spark Core

Ecosistema Spark

- Spark Streaming
  - Gathering – raccolta dei dati da sorgenti semplici (filesystem o socket TCP) ovvero complesse (altri framework per ETL)
  - Processing – applicazione di Trasformazioni e azioni su un modello dei dati apposito: il *Discretized Stream (DStream*) che internamente è una sequenza di RDD
  - Data storage – salvataggio su file system, strutture di dashboard, database

Ecosistema Spark

- Spark ML
  - Raccolta di algoritmi di machine learning
    - Preprocessing dei dati – cleaning, imputazione, estrazione di feature
    - Supervised learning
      - Classificazione
      - Regressione lineare / Regressione logistica
      - Decision tree
      - Random Forests
      - Naive Bayes

Ecosistema Spark

- Spark ML
  - Raccolta di algoritmi di machine learning
    - Unsupervised learning
      - K-means
      - Misture di gaussiane
      - Latent Dirichlet Allocation – clustering gerarchico
    - Recommendation
      - Collaborative filtering
      - Frequent pattern mining

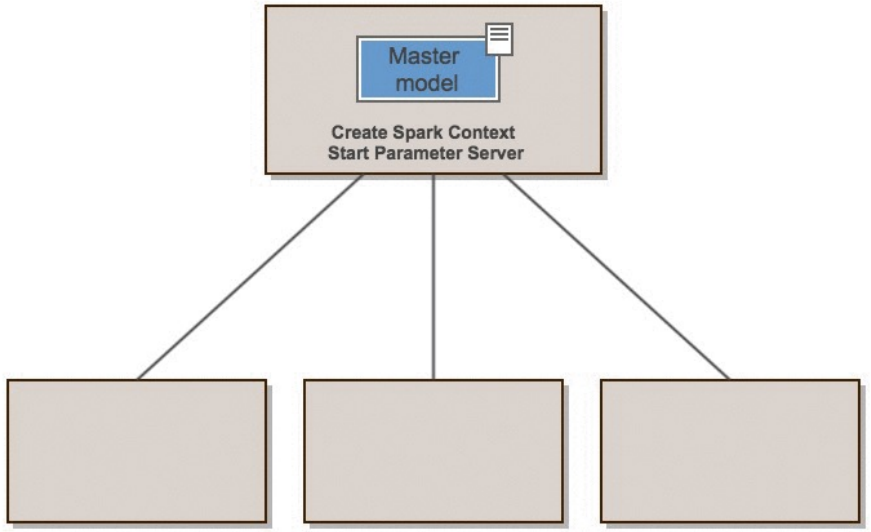
Ecosistema Spark

- Spark ML
  - Raccolta di algoritmi di machine learning
    - Analisi dei grafi
      - Ricerca di cammini tra nodi
      - Estrazione componenti connesse
      - PageRank
    - Deep Learning
      - Classificatore MLP nativo
      - Diverse librerie per integrare motori di DL con Spark

Ecosistema Spark

- Spark ML
  - Librerie per il Deep Learning
    - TensorFlowOnSpark (https://github.com/yahoo/TensorFlowOnSpark)
      - Distribuisce esplicitamente il job TensorFlow (*non Keras*) sul cluster Spark
      - Carica direttamente in TensorFlow i dati sia da RDD sia da DataFrame
      - Si appoggia al TensorFlow distribute mode

Ecosistema Spark



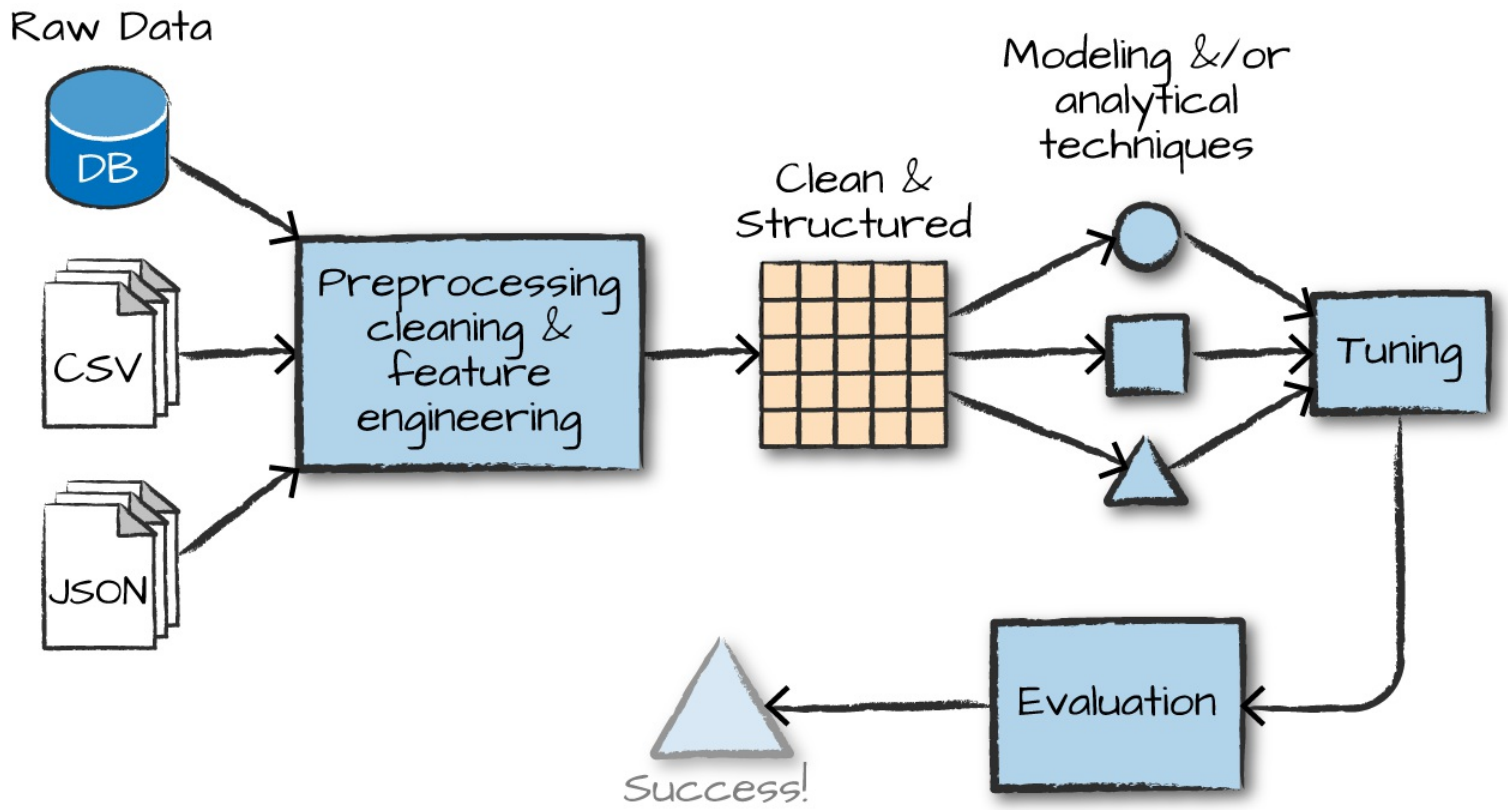
- Spark ML
  - Librerie per il Deep Learning
    - Elephas (https://github.com/maxpumperla/elephas)
      - Integra Keras con Spark
      - Implementa una suite di algoritmi di parallelizzazione dei dati tramite RDD
      - Il modello Keras viene inizializzato dal driver e quindi serializzato e inviato ai singoli workers

Ecosistema Spark

- Spark ML
  - Librerie per il Deep Learning
    - Deep Learning Pipelines (https://github.com/databricks/spark-deep-learning/tree/v1.6.0)
      - Integra le funzionalità TensorFlow/Keras al'interno di SparkML, rappresentandole come un o dei componenti standard di questa libreria
      - Il job viene distribuito di default sul cluster

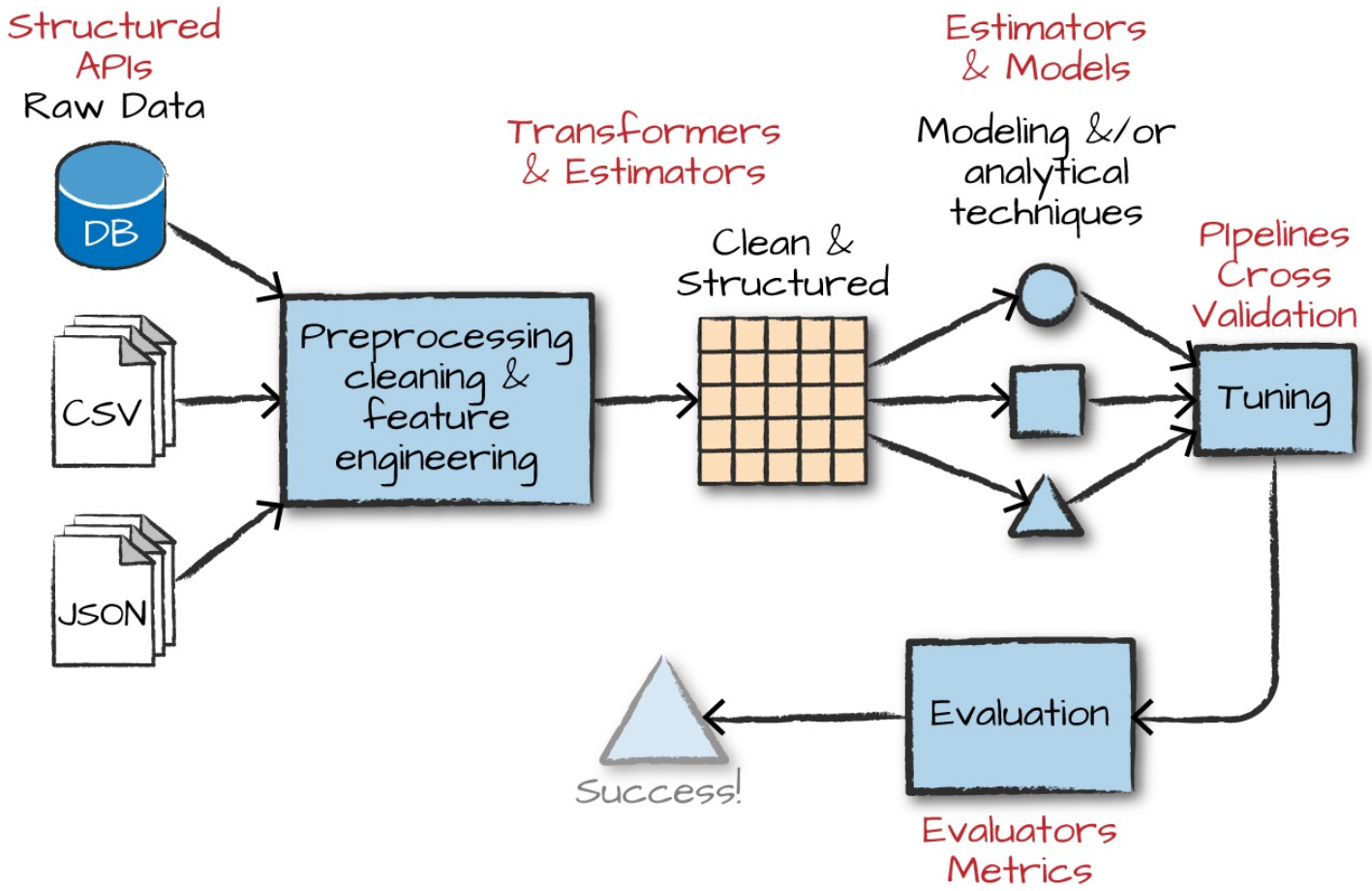
Ecosistema Spark

- Spark ML
  - Processo di analisi



Ecosistema Spark

- Spark ML
  - Processo di analisi
    - La libreria offre una serie di componenti software per ogni stadio del processo
      - Transformer
      - Estimator
      - Evaluator
      - Pipeline



Ecosistema Spark

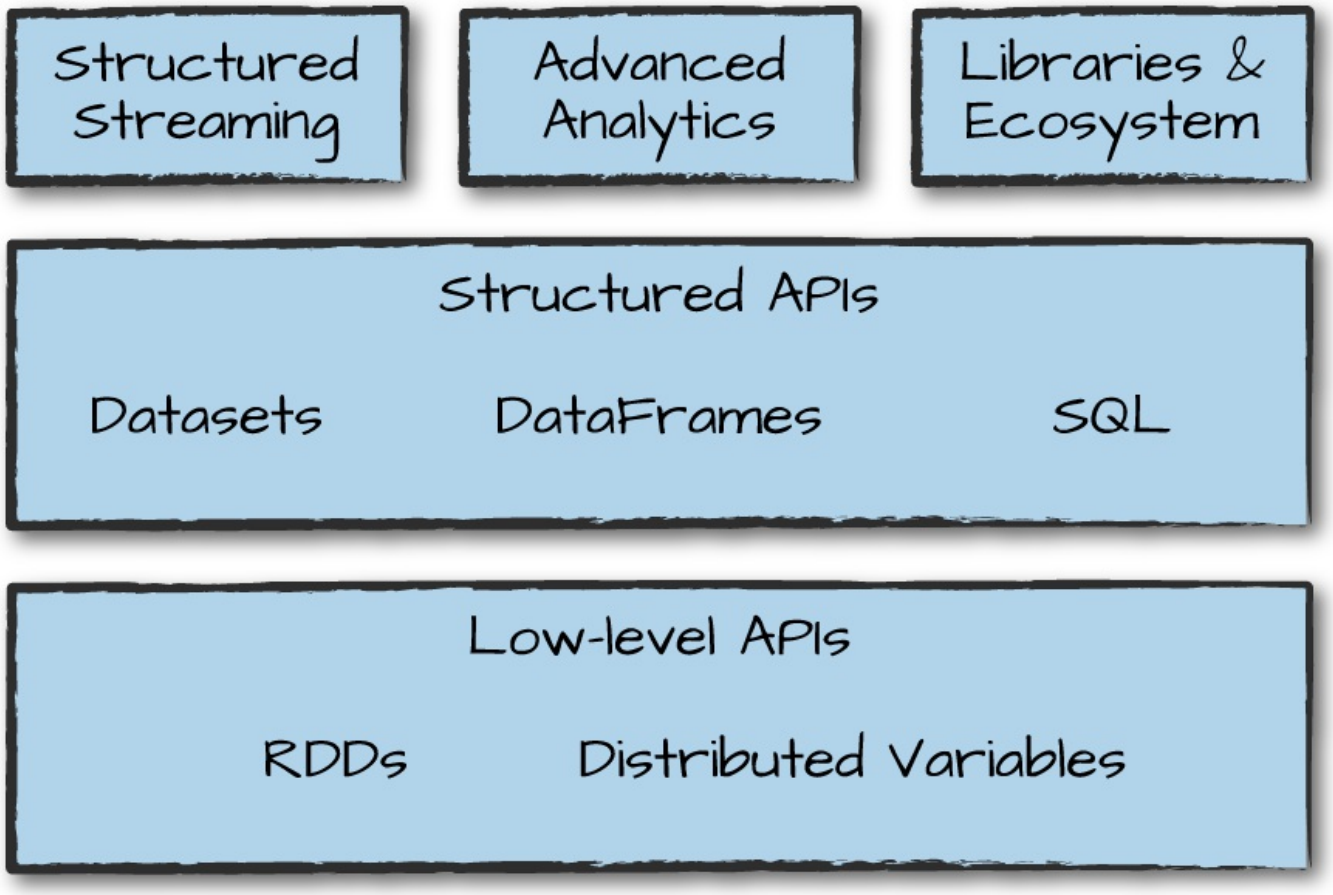
- Spark GraphX
  - API dedicata alla manipolazione di grafi
  - Algoritmi di ricerca di cammini, ricerca, attraversamento, clustering e classificazione su strutture dati a grafo
  - Gestisce una astrazione sugli RDD

Ecosistema Spark

- SparkR
  - API per l'interazione con Spark attraverso il linguaggio R che usa il proprio modello di DataFrame per gestire i dati
  - Conettività a database, machine learning, analisi dei dati

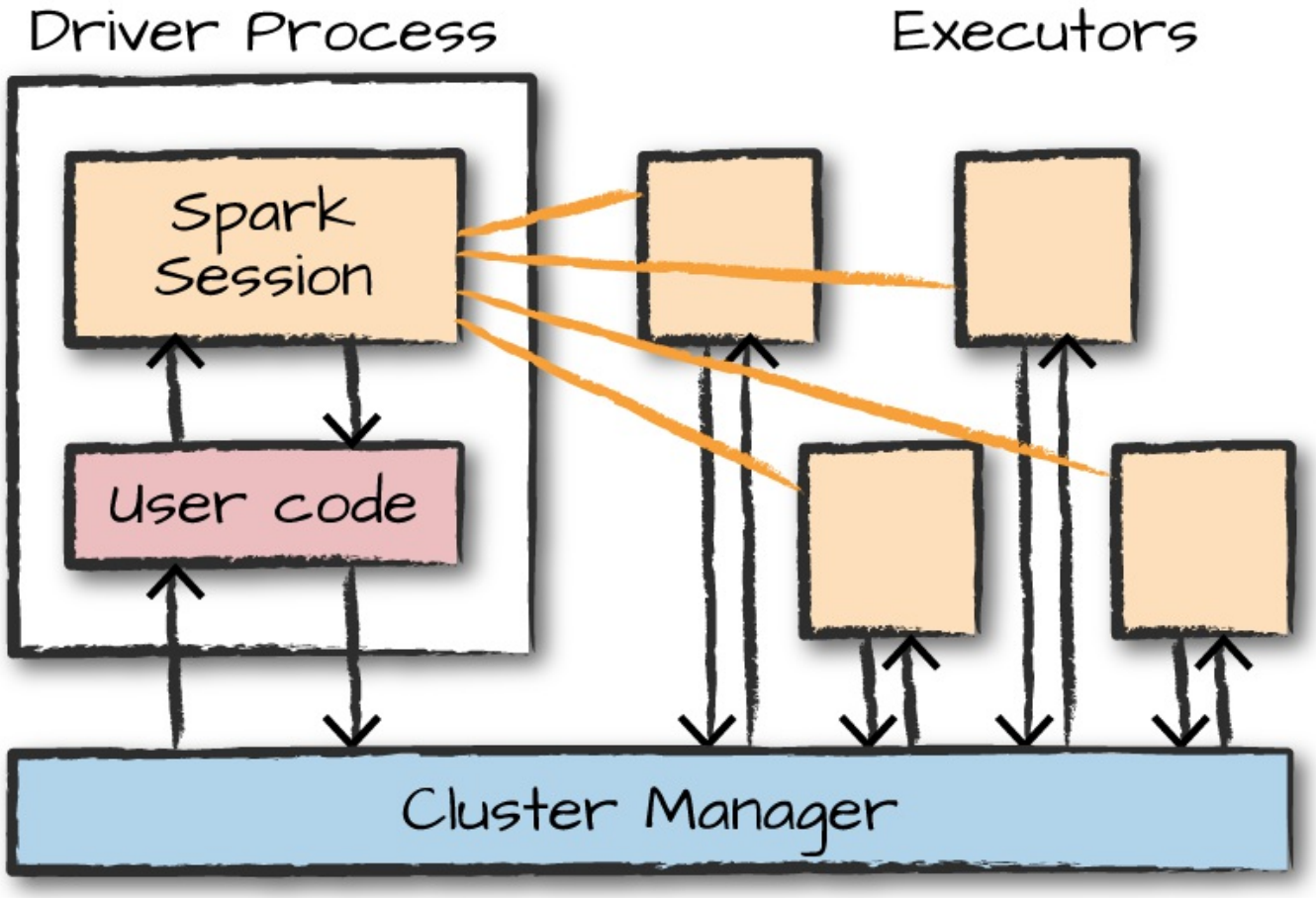
Architettura di Spark

- Da un punto di vista architetturale l'accesso alla API può avvenire
  - Attraverso i DataFrames e i Dataset
    - Dataset: DataFrame esplicitamente tipizzato
      - I tipi sono verificati a tempo di compilazione
      - Solo in Scala e Java
  - Utilizzando direttamente i RDD



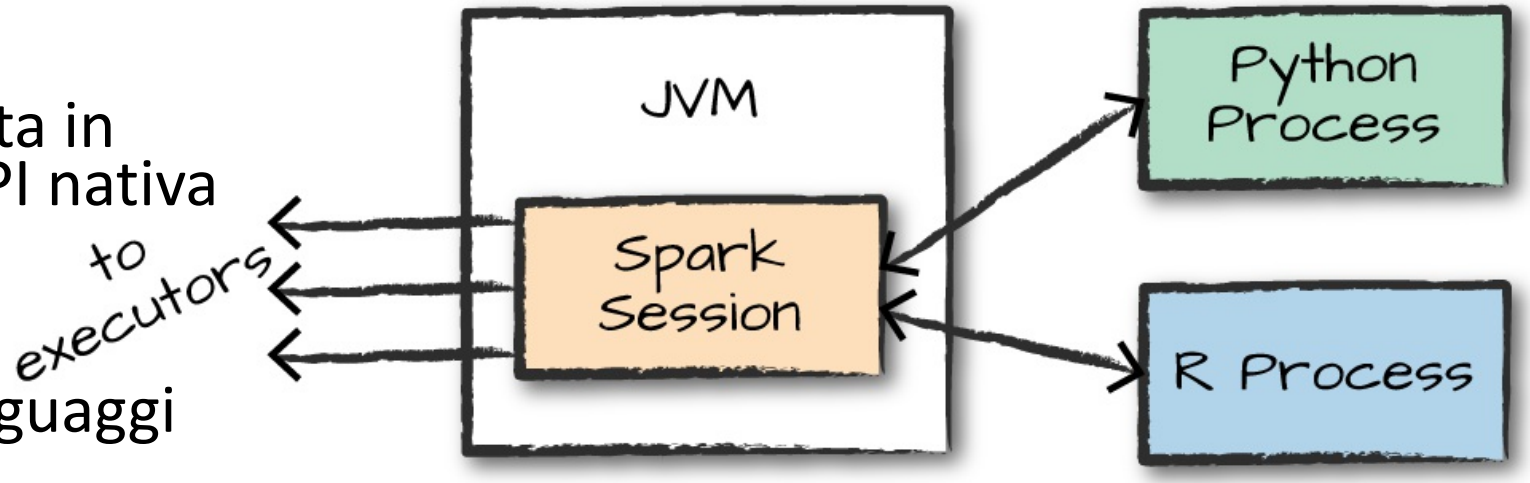
Architettura di Spark

- La struttura di una applicazione si basa sull'inizializzazione di una SparkSession da parte del codice dell'utente
- La `SparkSession` (Spark 2.X) controlla il processo driver che coordina gli esecutori
- Il cluster manager può essere stand-alone, YARN o Mesos



Architettura di Spark

- Spark è scritto in Scala e può essere programmato nativamente
- L'API Java è stata sviluppata in maniera coerente con l'API nativa Scala
- Il supporto per gli altri linguaggi si ottiene perché il codice Python/R viene tradotto in Java da Spark ed eseguito nelle JVM dei nodi del cluster



Architettura di Spark

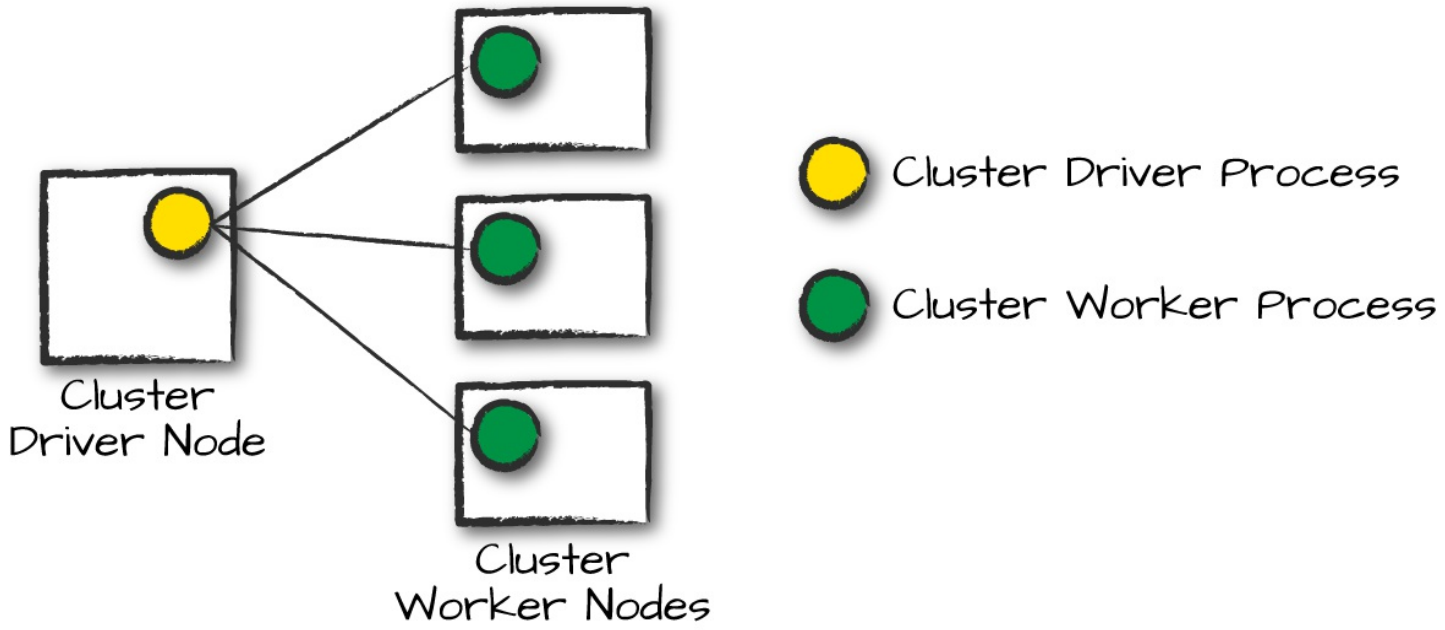
- `SparkSession` è il costrutto che inizializza l'applicazione e gestisce gli esecutori in Spark 2.X

- In Spark 1.X si utilizzavano `SparkContext`, `SQLContext` e `HiveContext` per gestire l'applicazione, l'interazione con SparkSQL e i DataFrames

- In Spark 2.X `SparkContext` è un oggetto di `SparkSession` che è responsabile direttamente della connessione al cluster, ma non è più necessario inizializzarlo esplicitamente dal codice perché lo fa la `SparkSession` quando viene invocata

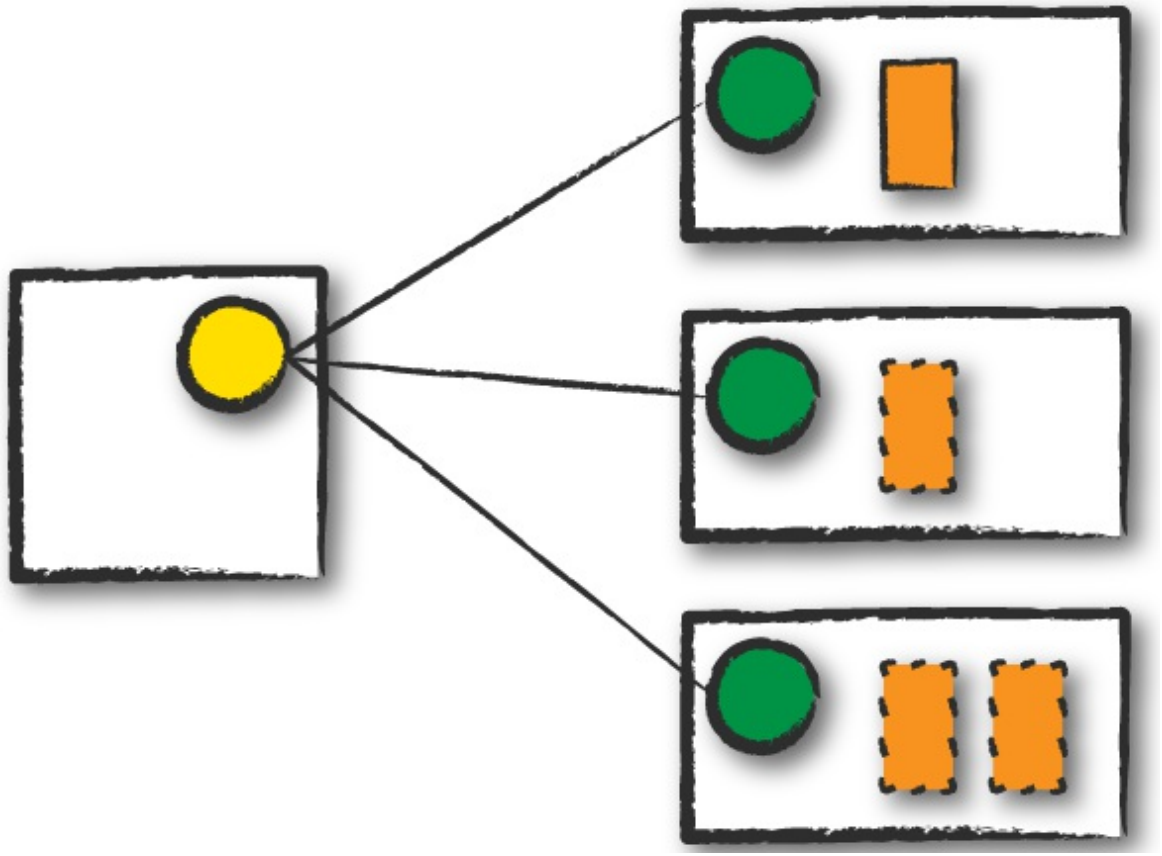
Architettura di Spark

- Modalità di esecuzione
  - Cluster mode
  - Client mode
  - Local mode



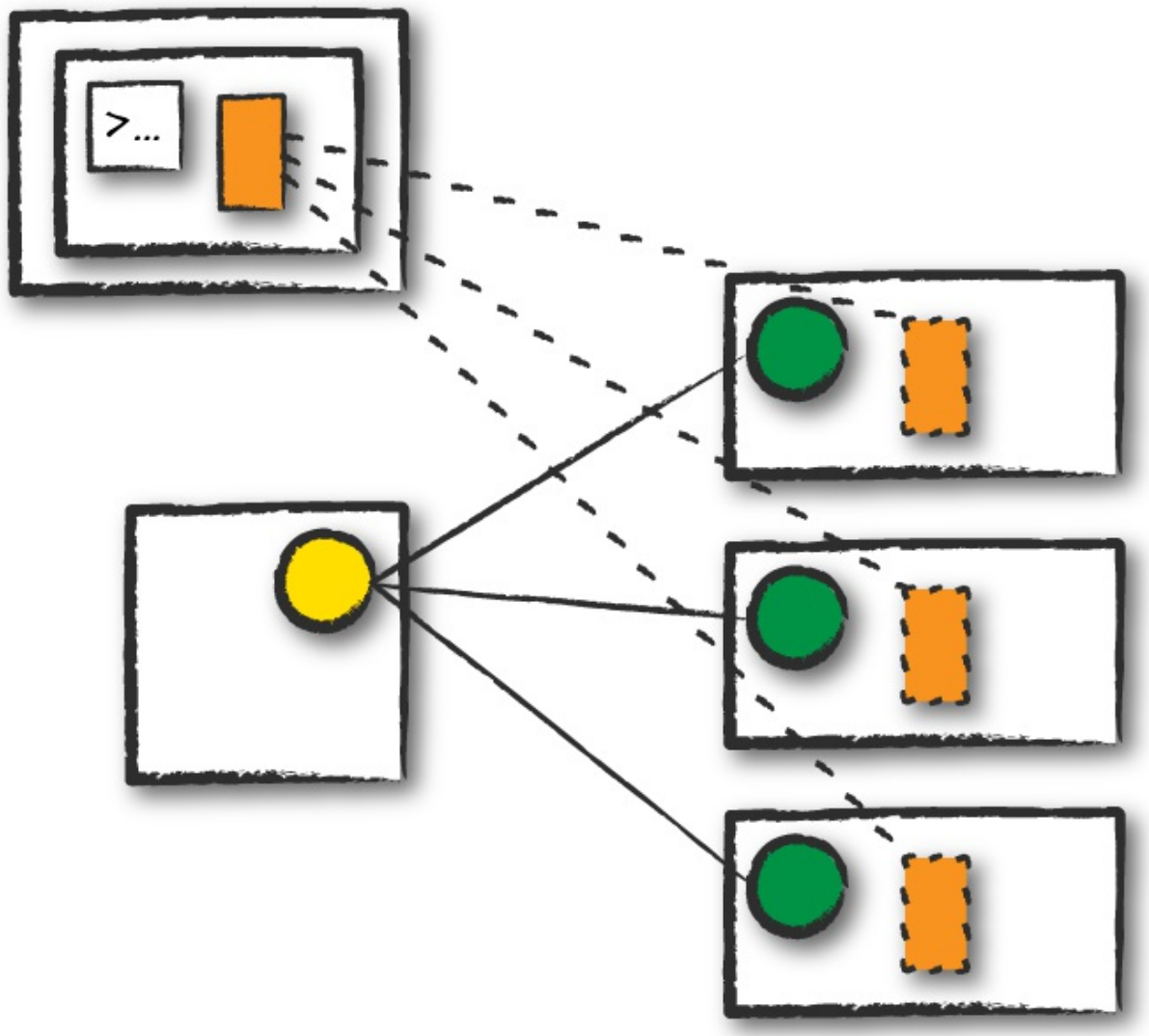
Architettura di Spark

- Modalità di esecuzione
  - Cluster mode
    - L'utente sottomette l'applicazione, cioè il driver, al cluster manager
    - Il cluster manager è direttamente responsabile della distribuzione degli esecutori



Architettura di Spark

- Modalità di esecuzione
  - Client mode
    - Il driver rimane presso la macchina dell'utente *fuori* dal cluster
    - Il client è responsabile del funzionamento degli esecutori
      - *Gateway machine*
      - *Edge node*

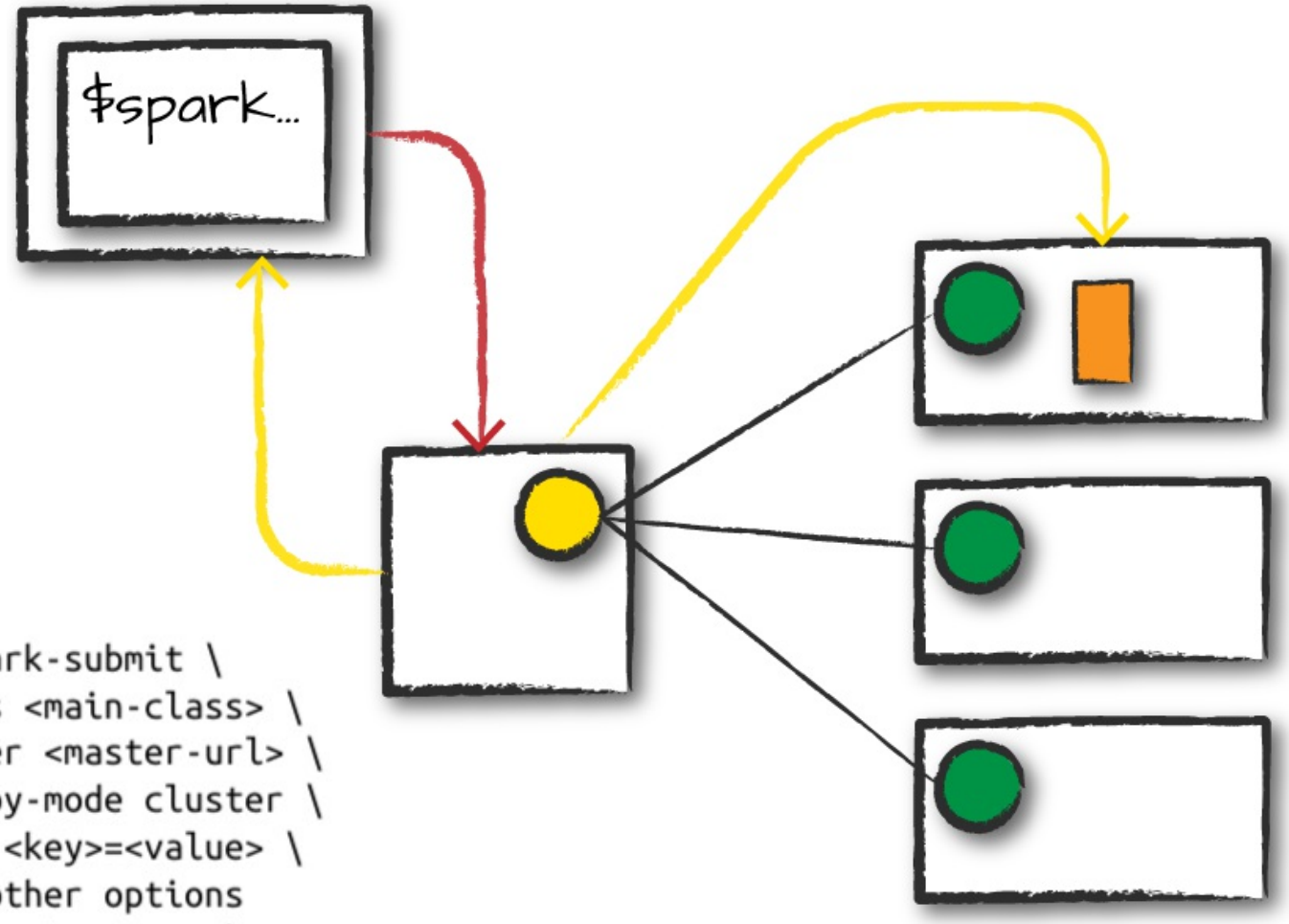


Architettura di Spark

- Modalità di esecuzione
  - Local mode
    - Esecuzione stand-alone su singola macchina
    - Parallellismo tramite thread
    - Usato per apprendimento, test e debug

Architettura di Spark

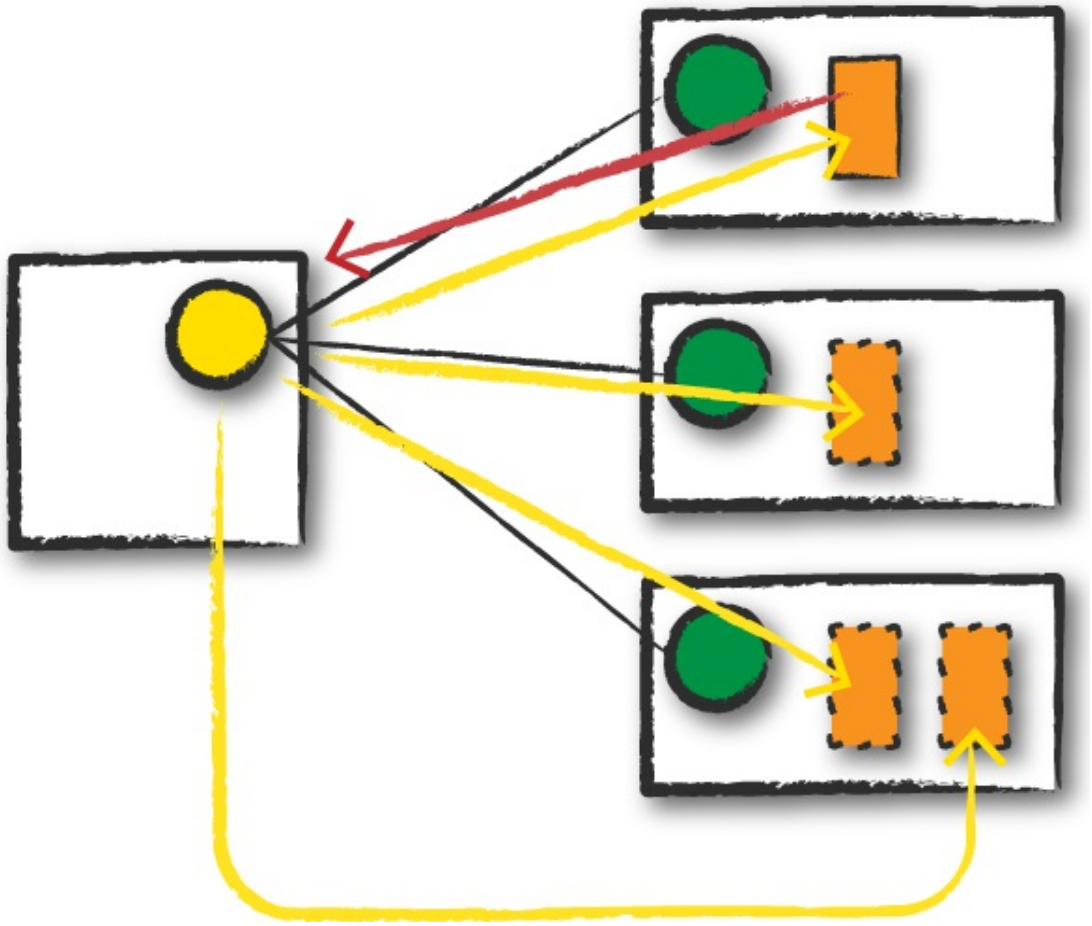
- Ciclo di vita dell'applicazione
  - L'utente invoca il batch e richiede le risorse per il processo driver
  - Il processo di invocazione termina dopo aver sottomesso il driver al cluster




```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode cluster \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

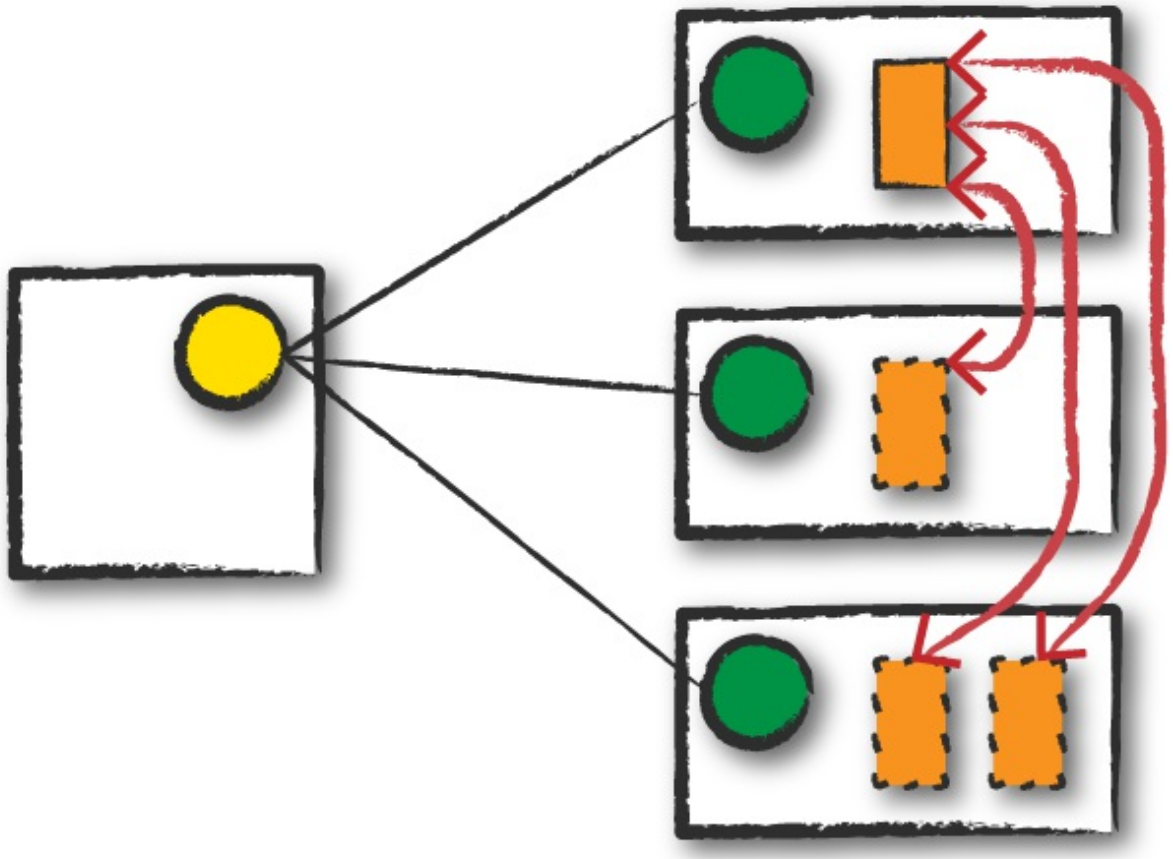
Architettura di Spark

- Ciclo di vita dell'applicazione
  - La `SparkSession` inizializza il cluster fatto da driver ed esecutori
  - Il driver negozia con il cluster manager l'allocazione degli esecutori
    - Configurazioni utente
  - Il cluster manager lancia effettivamente gli esecutori



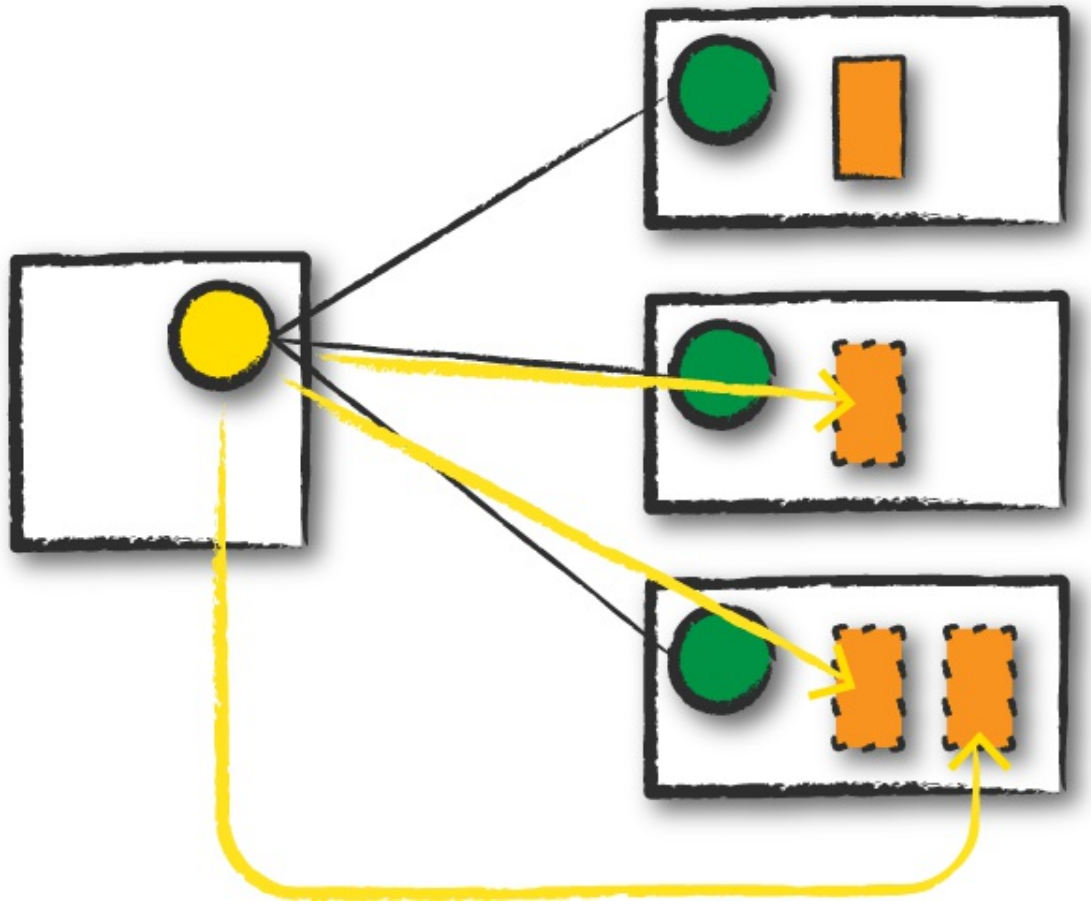
Architettura di Spark

- Ciclo di vita dell'applicazione
  - L'applicazione esegue attraverso comunicazioni dirette tra i processi del cluster (driver ed esecutori)



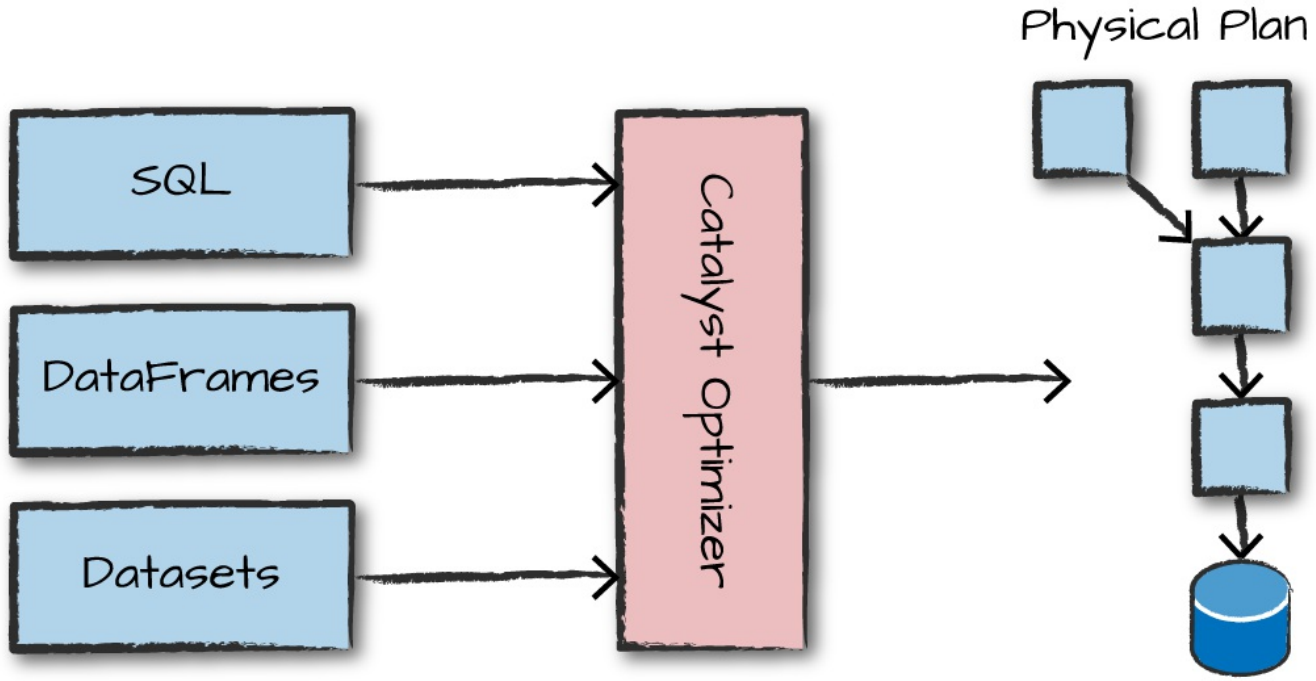
Architettura di Spark

- Ciclo di vita dell'applicazione
  - Al completamento
    - Il driver termina la sua esecuzione
    - Il cluster manager termina i processi esecutori



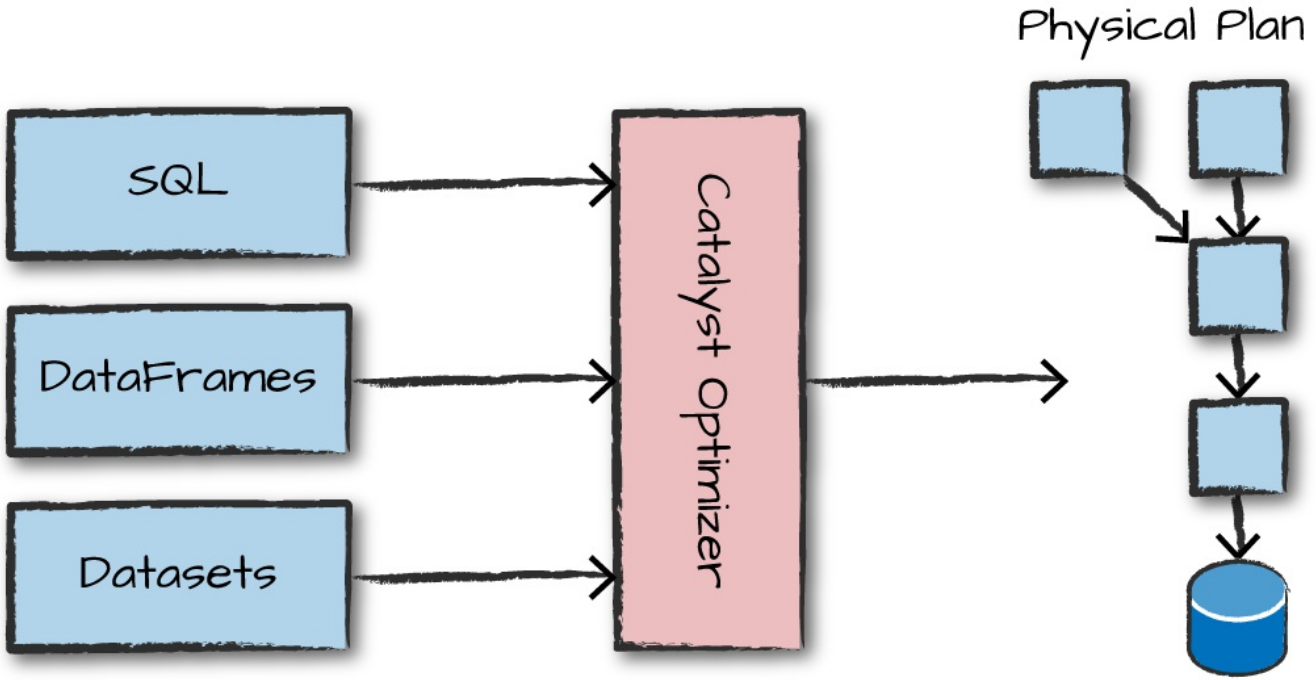
Architettura di Spark

- Esecuzione della API di alto livello
  - L'utente scrive codice in Python/Scala/R/Java che manipola DataFrames, Dataset ovvero codice SQL
  - Il codice viene validato da Spark per generare un *piano logico* di esecuzione



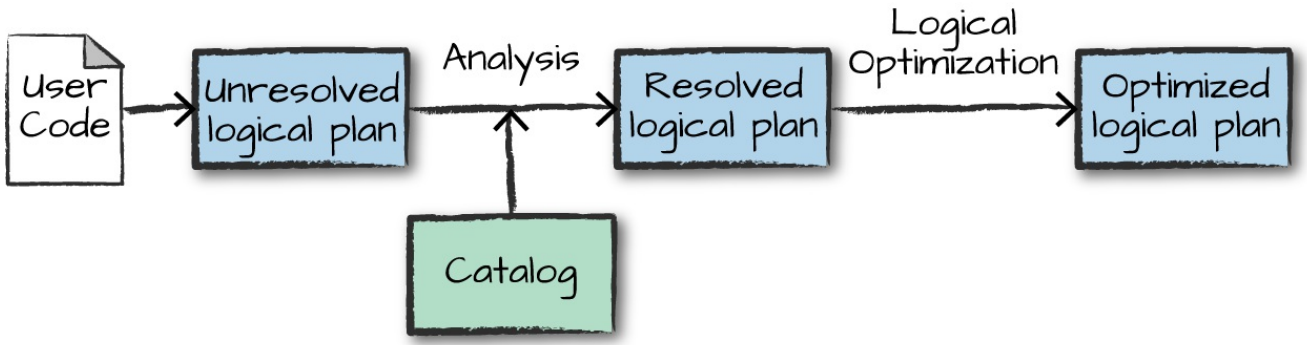
Architettura di Spark

- Esecuzione della API di alto livello
  - Catalyst ottimizza il piano logico, generando un *piano fisico*
  - Spark esegue il piano fisico che è una sequenza esplicita di trasformazioni tra RDD ed azioni



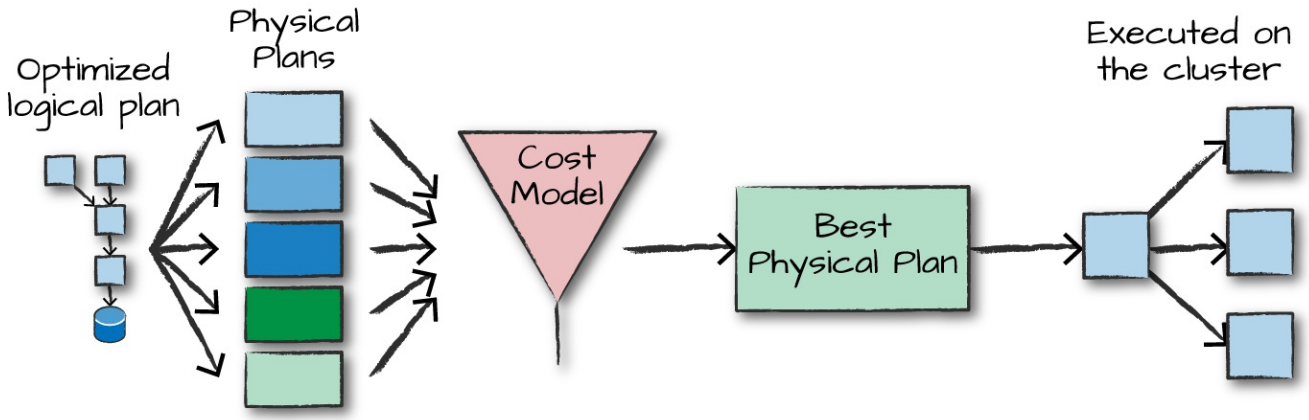
Architettura di Spark

- Esecuzione della API di alto livello
  - Il codice dell'utente genera un piano logico non risolto
    - Codice valido
    - Nessuna informazione sui DataFrame
  - Il Catalog è il repository di tutte le informazioni sui DataFrame che viene usato da Spark per verificare la validità del piano rispetto ai DataFrame



Architettura di Spark

- Esecuzione della API di alto livello
  - Il piano logico corrisponde ad una serie di possibili piani fisici di esecuzione
  - Vengono utilizzati dei modelli di costo di esecuzione per stabilire qual è il miglior piano fisico da selezionare



Introduzione a PySpark

- Interfaccia Python per Spark
- Si installa come pacchetto in un ambiente Pyhton
- Accessibile via CLI con il comando `pyspark`
- Integrabile in un'applicazione Python

Introduzione a PySpark

- Fornisce l'accesso a una `SparkSession`
  - `spark` è la `SparkSession` predefinita nella CLI
  - in un'applicazione si deve importare il package `pyspark`

```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('MyAppName').getOrCreate()
```

Introduzione a PySpark

- Per utilizzare PySpark in un IDE è necessario configurare la variabile di ambiente `$SPARK_HOME`
  - Dev'essere già installato Hadoop
  - Si può utilizzare il pacchetto `findspark` per inizializzare l'ambiente per utilizzare, poi, la `SparkSession`

```
import findspark

location = findspark.find()
findspark.init(location)
```

Introduzione a PySpark

- La `SparkSession` consente l'accesso alla DataFrame API
- La proprietà `spark.sparkContext` consente di accedere allo `SparkContext` associato alla sessione e, quindi alla RDD API
- Il package `pyspark.sql` consente l'accesso a SparkSQL
- Il package `pyspark.streaming` consente l'accesso a Spark Streaming
- I package `pyspark.ml/.mllib` consentono l'accesso a Spark ML rispettivamente per DataFrame e RDD

Introduzione a PySpark

- Sia la DataFrame API sia la RDD API usano la metafora della catena di trasformazioni con lazy evaluation e della azione finale, per cui la tipica pipeline di processo si presenta secondo il seguente pseudo-codice

```
myDataFrame/myRDD.trasf1(…)\
                 .trasf2(…)\
                 .trasf3(…)\
                     …
                 .azione()
```