

Programma



01 Introduzione a Spark

02 Architettura Spark

03 RDD e Dataframes

04 Sottosistema SQL

A photograph of a woman with long blonde hair tied back, wearing a black blazer, sitting at a white desk. She is facing away from the camera, writing in a small notebook with a pen. On the desk, there is a small potted plant in a copper-colored pot, a pair of scissors, and some dried branches. In the background, there is a framed abstract painting on the wall. The right side of the image features a large blue rectangular area with a yellow vertical bar on its left edge. The text "Introduzione Spark" is written in white, bold, sans-serif font.

Introduzione Spark



Spark

Che cos'è?

Capiamone le basi

- Apache Spark è un motore di analisi open-source utilizzato per carichi di lavoro Big Data.
- È in grado di gestire l'analisi e l'elaborazione dei dati sia in batch sia in tempo reale.
 - **Batch Processing** (elaborazione in batch): In questa modalità, i dati vengono raccolti, processati e analizzati in lotti o gruppi.
 - **Real-time Processing** (elaborazione in tempo reale): In questa modalità, i dati vengono elaborati man mano che arrivano, senza dover aspettare che si accumulino in lotti.



Spark

Che cos'è?

Capiamone le basi e perché è nato

- Apache Spark è nato nel 2009 come progetto di ricerca presso l'Università di Berkeley, in California con l'obiettivo di trovare un modo per velocizzare l'elaborazione dei dati nei sistemi Hadoop.

Cos'è Hadoop?

- Apache **Hadoop** è un framework software e un motore di elaborazione parallela open-source basato su Java. Consente di suddividere le attività di analisi dei Big Data in attività più piccole che possono essere eseguite in parallelo utilizzando un algoritmo (come l'algoritmo **MapReduce**) e distribuendole su un cluster Hadoop.



Spark

Che cos'è?

MapReduce

- **Esigenza:** centinaia di computazioni al giorno che processano grandi quantità di dati, come documenti, logs o pagine web per ottenere dati strutturati.
- Una computazione può essere vista come una **sequenza finita di round**, ognuno dei quali è composto da una funzione di map e una di reduce
- Durante la computazione di ogni round:
 - La funzione di **Map** prende in input una coppia chiave/valore e restituisce una coppia di valori intermedi chiave/valore.
 - La funzione **Shuffle** raggruppa tutti i valori intermedi con stessa chiave e la passa in input alla funzione di Reduce.
 - La funzione di **Reduce** prende in input una coppia di chiave intermedia/lista di valori, fa operazioni su questi valori e restituisce un insieme di valori.

Input →

files con un documento per record

“document1”, “to be or not to be”

funzione Map →

$\langle nome_{doc}, contenuto_{doc} \rangle$
 $\langle parola, 1 \rangle$

input:

e output:

funzione Reduce →

$\langle parola, lista\ di\ 1 \rangle$
output: $\langle parola, occorrenze_{\#} \rangle$

input:

e

Input →

files con un documento per record

“document1”, “to be or not to be”



“to”, “1”
“be”, “1”
“or”, “1”
...

funzione Map →

$\langle nome_{doc}, contenuto_{doc} \rangle$
 $\langle parola, 1 \rangle$

input:
e output:

funzione Reduce →

$\langle parola, lista\ di\ 1 \rangle$
output: $\langle parola, occorrenze_{\#} \rangle$

input:
e

Input →

files con un documento per record

funzione Map →

$\langle \text{nome}_{\text{doc}}, \text{contenuto}_{\text{doc}} \rangle$
 $\langle \text{parola}, 1 \rangle$

input:
e output:

“document1”, “to be or not to be”



“to”, “1”
“be”, “1”
“or”, “1”
...

key = “be”
values = “1”, “1”

key = “not”
values = “1”

key = “or”
values = “1”

key = “to”
values = “1”, “1”

funzione Reduce →

$\langle \text{parola}, \text{lista di 1} \rangle$
output: $\langle \text{parola}, \text{occorrenze}_{\#} \rangle$

input:
e

Input →

files con un documento per record

funzione Map →

<nome_{doc}, contenuto_{doc}>
<parola, 1>

input:
e output:

funzione Reduce →

<parola, lista di 1>
output: <parola, occorrenze_#>

input:
e

“document1”, “to be or not to be”



“to”, “1”
“be”, “1”
“or”, “1”
...

key = “be”
values = “1”, “1”

“2”

key = “not”
values = “1”

“1”

key = “or”
values = “1”

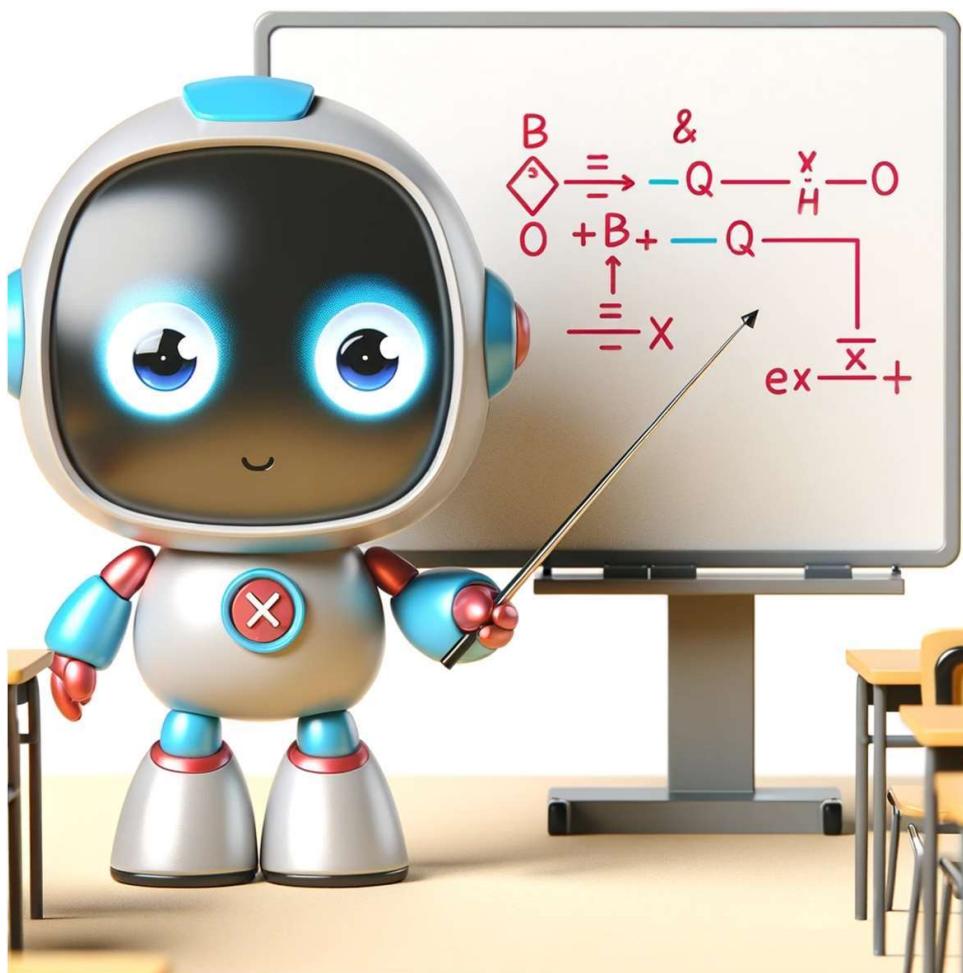
“1”

key = “to”
values = “1”, “1”

“2”

“be”, “2”
“not”, “1”
“or”, “1”
“to”, “2”

Esempio





Spark

Che cos'è?

Capiamone le basi

- Questo approccio trova applicazione in una vasta gamma di scenari e settori, ecco alcuni esempi MapReduce può essere utilizzato:
 - **Conteggio delle parole in grandi collezioni di testi.**
 - **Analisi dei dati di log.**
 - **Analisi dei dati scientifici.**
 - **Elaborazione di dati di sensori e IoT.**



Spark

Che cos'è?

Capiamone le basi

- Sebbene Hadoop fosse inizialmente una soluzione innovativa per l'elaborazione di grandi quantità di dati, aveva alcuni problemi chiave che Apache Spark mirava a risolvere:
 - **Velocità**
 - **Facilità di sviluppo**
 - **Diversità di Workload**
 - **Efficienza del cluster**



Un motore informatico
unificato e un insieme
di librerie per big data
i suoi componenti chiave.

Spark

Che cos'è?

[La filosofia dietro Spark](#)

- **Unificato:** Spark è progettato per supportare un'ampia gamma di attività di analisi dei dati, che vanno dal semplice caricamento dei dati e query SQL all'apprendimento automatico e al calcolo in streaming, sullo stesso motore di elaborazione e con un insieme coerente di API.
- Prima di Spark, nessun sistema open source ha cercato di fornire questo tipo di motore unificato per l'elaborazione parallela dei dati, il che significa che gli utenti dovevano mettere insieme un motore unificato partendo da diversi sistemi e diverse API.



Un motore informatico
unificato e un insieme di
librerie per big data
i suoi componenti chiave.

Spark

Che cos'è?

[La filosofia dietro Spark](#)

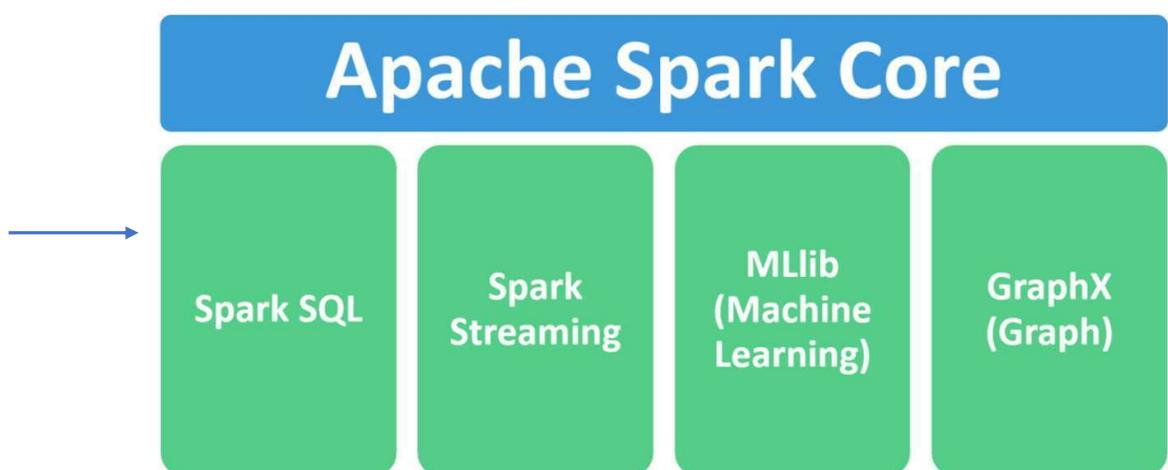
- **Motore di calcolo:** mentre Spark punta all'unificazione, Spark stesso limita attentamente il suo ambito ad essere un motore di calcolo.
- Spark può essere utilizzato con un'ampia varietà di storage persistente sistemi, inclusi sistemi di archiviazione cloud come Azure Storage e Amazon S3, file system distribuiti come Apache Hadoop, archivi di valori-chiave come Apache Cassandra e bus di messaggi come Apache Kafka.



Spark

Che cos'è?

Un motore informatico
unificato e un insieme di
librerie per big data
i suoi componenti chiave.





Spark

Che cos'è?

Capiamone le basi

- Apache Spark si basa su Hadoop **MapReduce** e ne estende il modello per utilizzarlo in modo efficiente durante altri tipi di operazioni, incluse query interattive ed elaborazione di flussi di dati.
- Spark offre binding nativi per i linguaggi di programmazione Java, Scala, Python e R. Inoltre, include diverse librerie per supportare la creazione di applicazioni per il machine learning [MLlib], l'elaborazione di dati in streaming [streaming Spark] e l'elaborazione di grafici [GraphX].



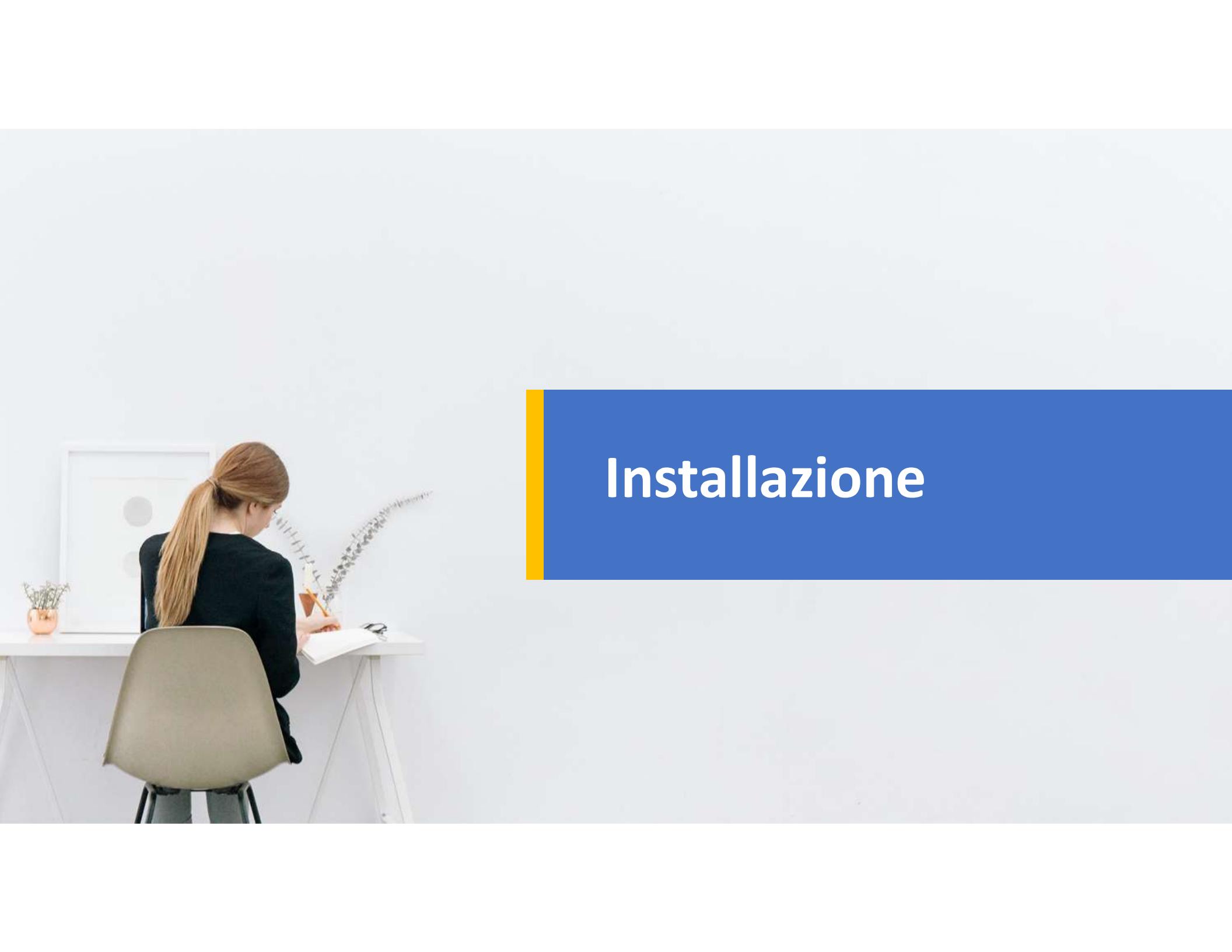
Spark

Quali sono i vantaggi di Apache Spark?



A photograph of a woman with long blonde hair tied back in a ponytail, wearing a black blazer over a white shirt. She is seated at a light-colored wooden desk, facing away from the camera. She is holding a pen and writing in a small open notebook. On the desk in front of her is a tall, thin vase containing a single, long, thin, dried plant. To her left, a white shelf holds a framed abstract painting with grey and white tones, a small copper-colored vase with dried flowers, and a pair of black headphones. The background is a plain, light-colored wall.

Spark Web UI

A photograph of a woman with long blonde hair tied back, wearing a dark green blazer, sitting at a white desk. She is focused on a small plant arrangement on the desk. The room is bright with white walls and a large window in the background. A small copper-colored vase with dried flowers sits on the desk. To the right of the image, there is a solid blue vertical bar with a thin yellow vertical bar to its left. The word "Installazione" is written in white, sans-serif font across the blue bar.

Installazione

Spark Web UI

Spark Web UI

Come è fatta?

Spark Jobs (2)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	count at SparkUIExample.scala:19	2020/01/29 18:03:28	0.2 s	2/2	1/1
0	copy at SparkUIExample.scala:19	2020/01/30 18:03:34	0.4 s	1/1	1/1

- Apache Spark fornisce una serie di interfacce utente web (Jobs, Stages, Tasks, Storage, Environment, Executors e SQL) per monitorare lo stato dell'applicazione Spark/PySpark, il consumo di risorse del cluster Spark e le configurazioni di Spark.
- Se si stai eseguendo l'applicazione Spark in locale, l'interfaccia utente di Spark può sarà fruibile utilizzando <http://localhost:4040/>.
- Di default, l'interfaccia utente di Spark viene eseguita sulla porta 4040.

Spark Jobs (?)

User: Mattia

Total Uptime: 3,5 h

Scheduling Mode: FIFO

Completed Jobs: 1

Spark Web Ui

Spark Web Ui

Spark Jobs Tab

- Scheduling Mode:
 - **Standalone**
 - **Mesos**
 - **YARN**
- Fornisce il numero dei Job completati.

Spark Web Ui

Spark Web Ui

Spark Stage Tab

- È possibile navigare nella scheda Stage in due modi:
 - Selezionare la Descrizione del rispettivo job Spark (Mostra solo gli stage per il job Spark selezionato)
 - Selezionare l'opzione Stages nella parte superiore della scheda Job Spark (Mostra tutti gli stage nell'applicazione)
- Nell'immagine sulla sinistra è possibile notare 4 stages.

The screenshot shows the 'Stages for All Jobs' section of the Spark Web UI. At the top, there are tabs for 'Jobs', 'Stages', 'Storage', 'Environment', 'Executors', and 'SQL'. The 'Stages' tab is selected. Below the tabs, it says 'Completed Stages: 4' and has a dropdown menu 'Completed Stages (4)'. There are two pagination controls: 'Page: 1' and 'Page: 1'. The main area displays a table of completed stages:

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
3	count at SparkUIExample.scala:20	+details 2020/07/20 21:41:35	57 ms	1/1		59.0 B		
2	count at SparkUIExample.scala:20	+details 2020/07/20 21:41:35	93 ms	1/1	296.6 kB		59.0 B	
1	csv at SparkUIExample.scala:18	+details 2020/07/20 21:41:35	0.2 s	1/1	296.6 kB			
0	csv at SparkUIExample.scala:18	+details 2020/07/20 21:41:34	0.3 s	1/1	64.0 kB			

Details for Stage 0 (Attempt 0)

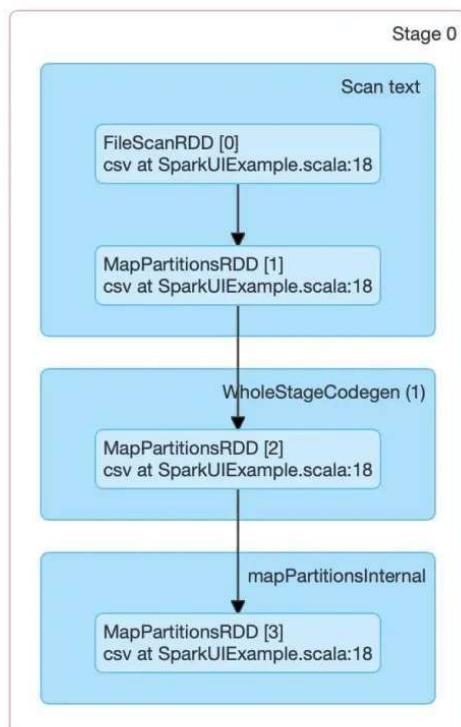
Total Time Across All Tasks: 94 ms

Locality Level Summary: Process local: 1

Input Size / Records: 64.0 KiB / 1

Associated Job Ids: 0

▼ DAG Visualization



Spark Web Ui

Spark Web Ui

Stage Detail

- I dettagli dello stage mostrano il Grafo Aciclico Diretto (DAG) di questo stage, dove i vertici rappresentano gli RDD o DataFrame e gli archi rappresentano un'operazione da applicare.
- All'interno di ogni Stage potete trovare i task singoli che permettono l'avanzamento dello stage.

Spark Web Ui

Spark Web Ui

Storage Tab

- La scheda Storage mostra gli RDD e i DataFrame persistenti, se presenti, nell'applicazione.
- La pagina di riepilogo mostra i livelli di memorizzazione, le dimensioni e le partizioni di tutti gli RDD, mentre la pagina dei dettagli mostra le dimensioni e gli executor per tutte le partizioni in un RDD o DataFrame.

Storage

« RDDs

ID	RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
1	rdd	Memory Serialized 1x Replicated	5	100%	236.0 B	0.0 B
4	LocalTableScan [count#7, name#8]	Disk Serialized 1x Replicated	3	100%	0.0 B	2.1 KiB

Spark Web Ui

Spark Web Ui

Tab Environment

- Questa pagina di ambiente ha cinque parti.

- **Informazioni sull'esecuzione:** contiene semplicemente le proprietà di runtime come le versioni di Java e Scala.
- **Proprietà di Spark:** elenca le proprietà dell'applicazione come 'spark.app.name' e 'spark.driver.memory'.
- **Proprietà di Hadoop:** mostra le proprietà relative a Hadoop e YARN. Nota: Le proprietà come 'spark.hadoop' non vengono mostrate in questa parte, ma in 'Proprietà di Spark'.
- **Proprietà di sistema:** mostra maggiori dettagli sulla JVM.
- **Voci del classpath:** elenca le classi caricate da diverse fonti, il che è molto utile per risolvere conflitti di classi.



Environment

- [Runtime Information](#)
- [Spark Properties](#)
- [Hadoop Properties](#)
- [System Properties](#)
- [Classpath Entries](#)

Spark Web UI

Spark Web UI

Tab Esecutori

The screenshot shows the Spark Web UI interface with the 'Executors' tab selected. At the top, there's a navigation bar with links for Jobs, Stages, Storage, Environment, Executors (which is highlighted in blue), and SQL. Below the navigation bar, the title 'SparkUIExample application UI' is displayed. The main content area is divided into two sections: 'Summary' and 'Executors'.

Summary: This section contains a single table with three rows: Active(1), Dead(0), and Total(1). The columns include RDD Blocks, Storage Memory, Disk Used, Cores, Active Tasks, Failed Tasks, Complete Tasks, Total Tasks, Task Time (GC Time), Input, Shuffle Read, Shuffle Write, and Blacklisted.

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(1)	0	0.0 B / 912.3 MB	0.0 B	3	0	0	4	4	0.7 s (0.0 ms)	657.1 KB	59 B	59 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	0.0 B / 912.3 MB	0.0 B	3	0	0	4	4	0.7 s (0.0 ms)	657.1 KB	59 B	59 B	0

Executors: This section contains two tables. The first table lists executors by ID, showing their address, status, and resource usage. The second table provides a detailed breakdown of task metrics for each executor.

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump	Thread Dump
driver	192.168.55.101:49668	Active	0	0.0 B / 912.3 MB	0.0 B	3	0	0	4	4	0.7 s (0.0 ms)	657.1 KB	59 B	59 B	Thread Dump	Thread Dump

Below the tables, a note indicates 'Showing 1 to 1 of 1 entries'.

- La scheda Executors visualizza informazioni di riepilogo sugli executor che sono stati creati per l'applicazione, inclusi l'utilizzo di memoria e disco e le informazioni sui task e sugli shuffle. La colonna Storage Memory mostra la quantità di memoria utilizzata e riservata per la memorizzazione nella cache dei dati.
- La scheda Executors fornisce non solo informazioni sulle risorse come la quantità di memoria, disco e core utilizzati da ciascun executor, ma anche informazioni sulle prestazioni.

Spark Web Ui

Spark Web Ui

Tab SQL

- Se l'applicazione esegue query Spark SQL, la scheda SQL visualizza informazioni come la durata, i job e i piani fisici e logici per le query. Qui includiamo un esempio di base per illustrare questa scheda:

The screenshot shows the Spark Web UI interface. At the top, there are tabs for Jobs, Stages, Storage, Environment, Executors, and SQL. The SQL tab is selected. Below the tabs, it says "Completed Queries: 2". A table lists two completed queries:

ID	Description	Submitted	Duration	Job IDs
1	count at SparkUIExample.scala:20	2020/07/20 21:41:35 +details	0.3 s	[2]
0	csv at SparkUIExample.scala:16	2020/07/20 21:41:33 +details	1 s	[0]

```
scala> val df = Seq((1, "andy"), (2, "bob"), (2, "andy")).toDF("count", "name")
df: org.apache.spark.sql.DataFrame = [count: int, name: string]

scala> df.count
res0: Long = 3

scala> df.createGlobalTempView("df")

scala> spark.sql("select name,sum(count) from global_temp.df group by name").show
+-----+
|name|sum(count)|
+---+-----+
|andy|      3|
| bob|      2|
+---+-----+
```

Spark Web Ui

Spark Web Ui

```
scala> val df = Seq((1, "andy"), (2, "bob"), (2, "andy")).toDF("count", "name")
df: org.apache.spark.sql.DataFrame = [count: int, name: string]

scala> df.count
res0: Long = 3

scala> df.createGlobalTempView("df")

scala> spark.sql("select name,sum(count) from global_temp.df group by name").show
+-----+-----+
|name|sum(count)|
+-----+-----+
|andy|      3|
| bob|      2|
+-----+
```

SQL

Completed Queries: 3

~ Completed Queries (3)

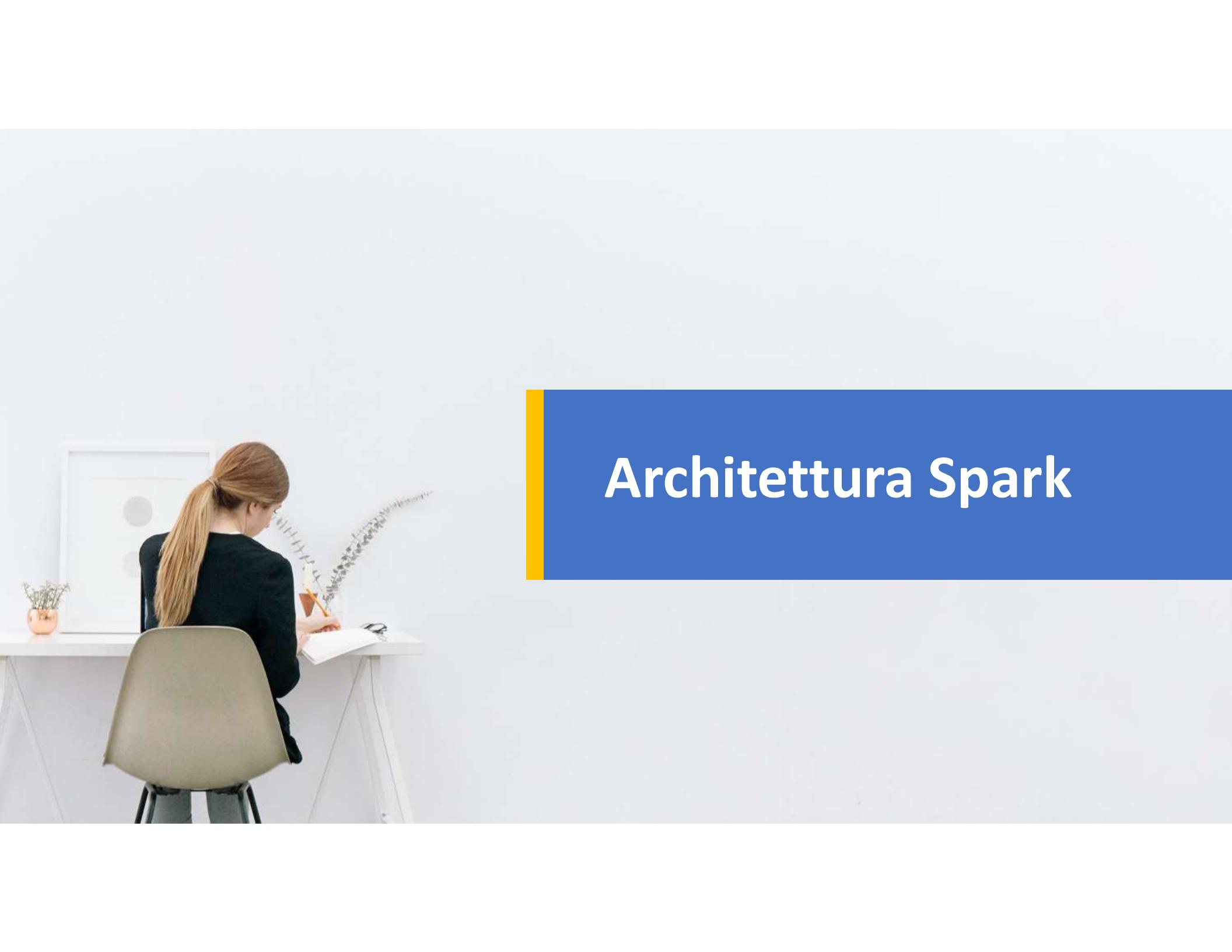
Page: 1

1 Pages. Jump to . Show items in a page.

ID	Description	Submitted	Duration	Job IDs
2	show at <console>:24	2019/08/04 15:23:15	2 s	[1][2][3][4][5]
1	createGlobalTempView at <console>:26	2019/08/04 15:23:09	0.3 s	
0	count at <console>:26	2019/08/04 15:23:01	2 s	[0]

Page: 1

1 Pages. Jump to . Show items in a page.

A photograph of a woman with long blonde hair tied back, wearing a black blazer, sitting at a white desk. She is facing away from the camera, drawing on a large sheet of paper with a pencil. On the desk, there is a small copper-colored vase with dried flowers and some other small objects. In the background, there is a framed abstract painting on the wall.

Architettura Spark



Spark

Struttura basica di spark

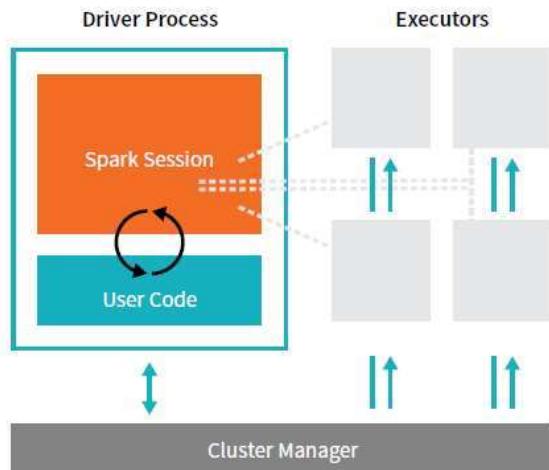
[Entriamo nel dettaglio](#)

- Un **cluster**, ossia un gruppo di macchine, mette insieme le risorse di molte macchine, permettendo di sfruttare tutte un insieme di risorse come se fossero una sola.
- Tuttavia, da solo, un gruppo di macchine non è sufficientemente potente; è necessaria una struttura per coordinarle e farle lavorare insieme in modo efficiente.
- **Spark** è uno strumento progettato appositamente per questo scopo: gestire e coordinare l'esecuzione di attività sui dati attraverso un gruppo di computer.

Spark

Struttura basica di spark

Entriamo nel dettaglio

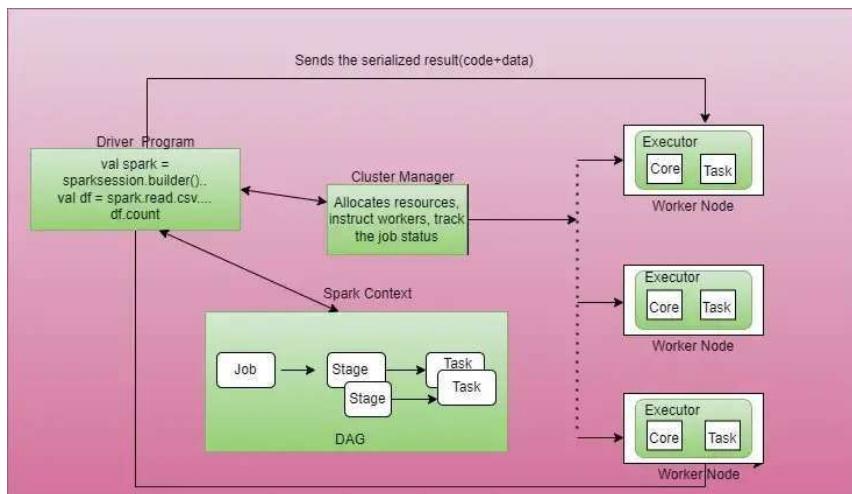


- Apache Spark funziona con un'architettura master-slave dove il master è chiamato "**Driver**" e gli slave sono chiamati **esecutori** o "**Workers**".
- Quando si esegue un'applicazione Spark, il **Driver** di Spark crea un contesto che è un punto di ingresso per l'applicazione, e tutte le operazioni (trasformazioni e azioni) vengono eseguite sui nodi **worker**, e le risorse sono gestite dal Cluster Manager.

Spark

Struttura basica di spark

Cos'è un Job?



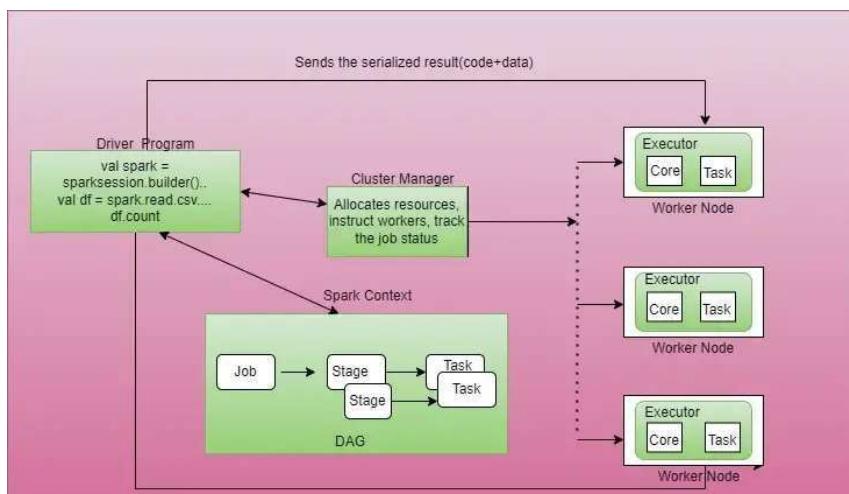
- Un **job** Spark può essere qualsiasi compito che deve essere eseguito su una grande quantità di dati troppo grande per essere elaborata su una singola macchina.
- L'immagine posta sulla sinistra vi mostra un job di spark.
- In Apache Spark, un job è diviso in **fasi** (o **stages**) Spark, dove ogni fase rappresenta un insieme di compiti che possono essere eseguiti in parallelo.
- Una fase consiste in un insieme di compiti che vengono eseguiti su un insieme di partizioni dei dati.

Spark

Struttura basica di spark

Cos'è un Job?

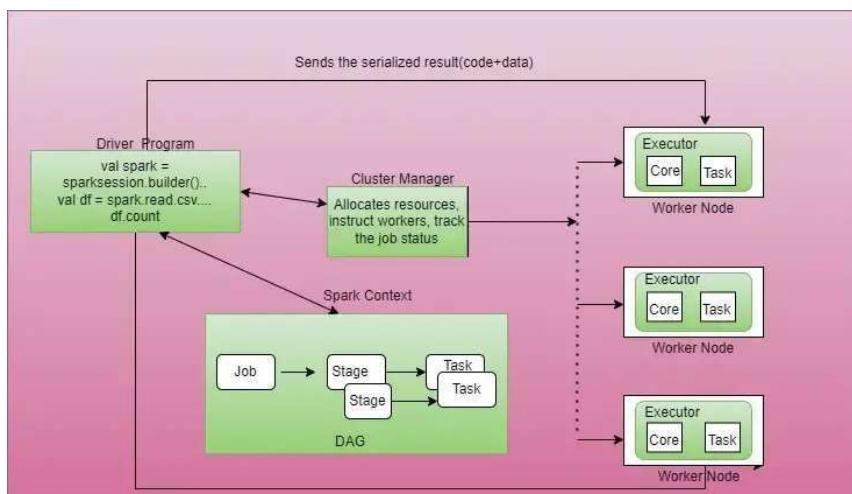
- Un lavoro Spark di solito coinvolge i seguenti passaggi:
 - Caricare i dati da una fonte di dati.
 - Trasformare o manipolare i dati utilizzando le API di Spark come Map, Reduce, Join, ecc.
 - Memorizzare i dati elaborati in un repository di dati.



Spark

Struttura basica di spark

Quando viene creato un job in Spark?



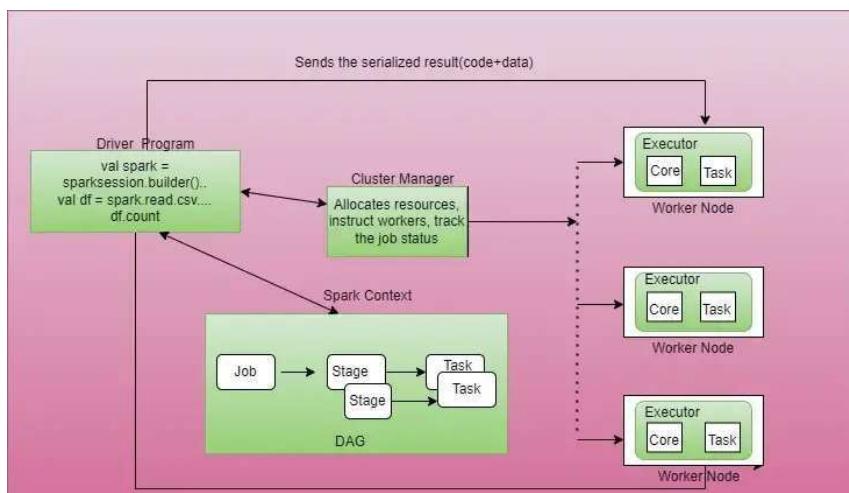
- In Apache Spark, un job viene creato quando viene chiamata un'azione Spark su un **RDD** (Resilient Distributed Dataset) o un **DataFrame**.
- Le trasformazioni costruiscono un piano di esecuzione logico chiamato **DAG** (Grafo Aciclico Diretto) che descrive il calcolo da eseguire sui dati.
- Quando viene chiamata un'azione, Spark esamina il DAG e pianifica le trasformazioni e i calcoli necessari da eseguire sui dati distribuiti.
- Questo processo crea un lavoro, che è una raccolta di compiti che vengono inviati ai nodi worker nel cluster per l'esecuzione.

Spark

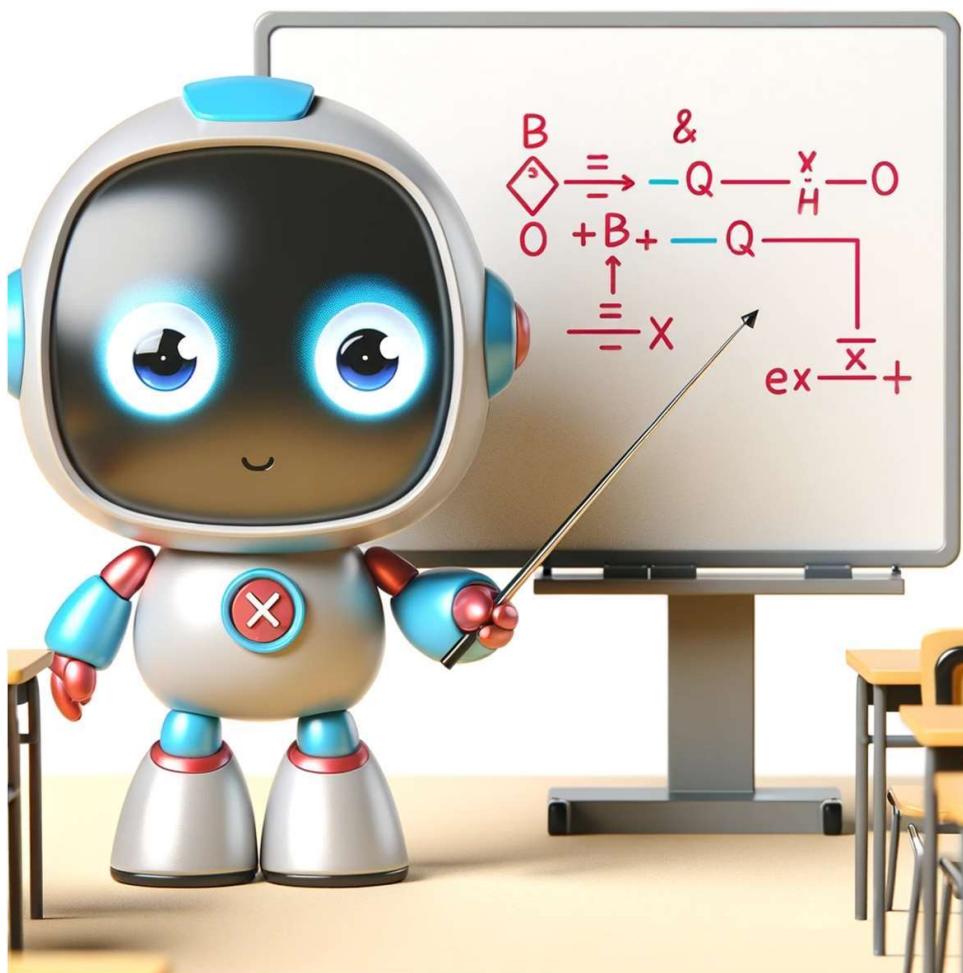
Struttura basica di spark

Azioni che triggerano la creazione di un job

- Ecco alcuni esempi di azioni in Spark che possono attivare la creazione di un lavoro:
 - **Count()**: Questa azione restituisce il numero di elementi nell'RDD o DataFrame. Quando viene chiamata, attiva il calcolo di tutte le trasformazioni che portano al conteggio finale e crea un lavoro per eseguire il calcolo.
 - **Collect()**: Questa azione restituisce tutti gli elementi dell'RDD o DataFrame come un array al programma driver. Quando viene chiamata collect(), attiva il calcolo di tutte le trasformazioni che portano alla raccolta finale e crea un lavoro per eseguire il calcolo.
- In generale, qualsiasi azione che richiede il calcolo delle trasformazioni applicate a un RDD o DataFrame attiverà la creazione di un lavoro in Spark.



Esempio



Spark

Struttura basica di spark

Azioni che triggerano la creazione di un job

- Nella nostra applicazione precedente, abbiamo eseguito 3 lavori Spark (0, 1, 2).

- Job 0: lettura del file CSV.
- Job 1: Inferire lo schema dal file.
- Job 2: Verifica del conteggio.

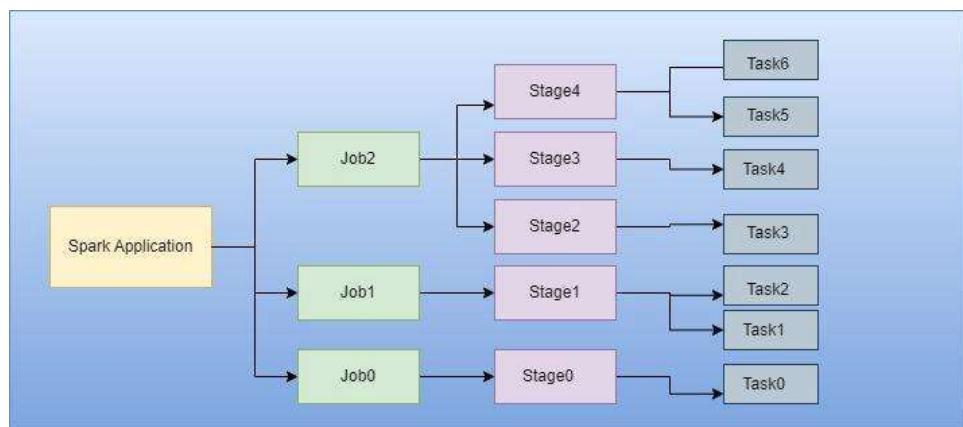
- Quindi, se guardiamo lo screenshot sulla sinistra, mostra chiaramente i risultati dei 3 lavori Spark relativi alle 3 azioni.

Completed Jobs (3)					
Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	count at SparkUIExample.scala:20 count at SparkUIExample.scala:20	2020/07/20 21:41:35	0.2 s	2/2	2/2
1	cav at SparkUIExample.scala:18 cav at SparkUIExample.scala:18	2020/07/20 21:41:35	0.3 s	1/1	1/1
0	cav at SparkUIExample.scala:18 cav at SparkUIExample.scala:18	2020/07/20 21:41:34	0.5 s	1/1	1/1

Spark

Struttura basica di spark

Cos'è uno Stage?

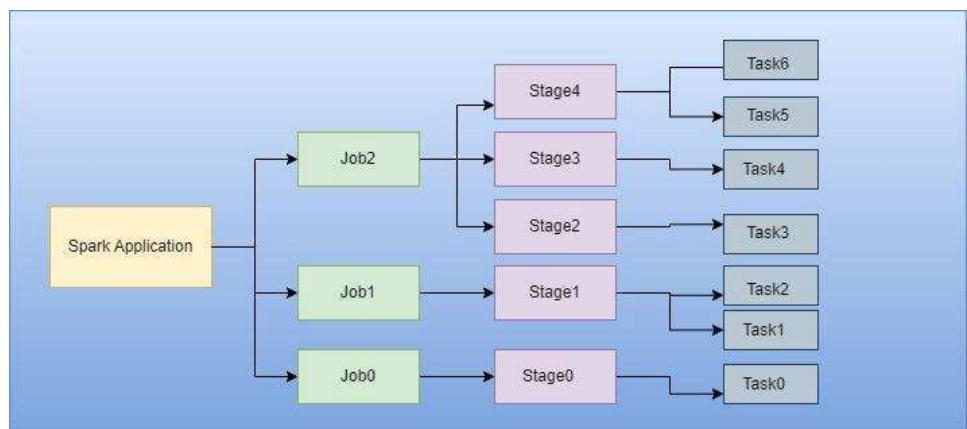


- Nel contesto di Apache Spark, uno **Stage** è un'unità di parallelismo in un lavoro Spark. Rappresenta un insieme di compiti che possono essere eseguiti insieme come parte di un singolo lavoro.
- L'applicazione Spark è suddivisa in diversi Job per ogni azione che verrà eseguita e i Job sono suddivisi in Stage, gli Stage infine sono suddivisi in task.
- Uno Stage è una collezione di compiti che condividono le stesse dipendenze, il che significa che devono scambiarsi dati l'uno con l'altro durante l'esecuzione.

Spark

Struttura basica di spark

Cos'è uno Stage?



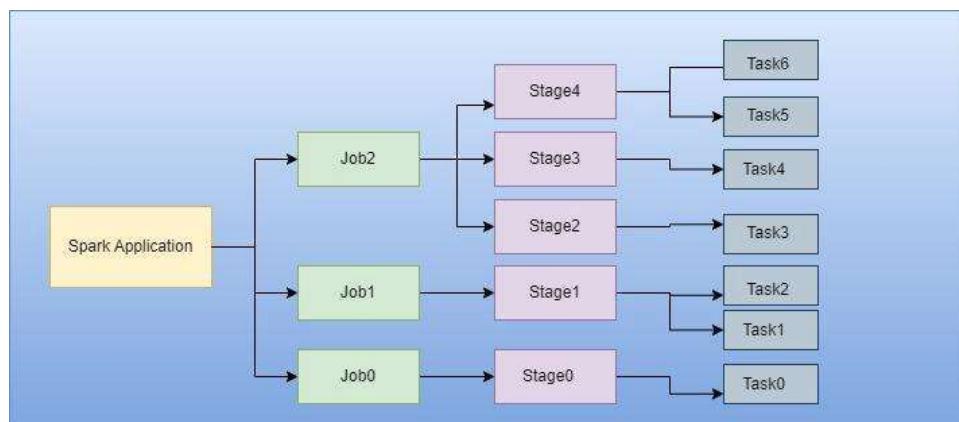
- Gli stage sono importanti perché consentono a Spark di eseguire calcoli paralleli in modo efficiente.
- Suddividendo un lavoro in stage, Spark può pianificare i compiti in modo da massimizzare il parallelismo riducendo al minimo la quantità di dati da scambiare tra i nodi. Questo può portare a significativi miglioramenti delle prestazioni, specialmente per lavori di elaborazione di dati su larga scala.
- Esistono due tipologie di Stage in Spark:
 - Narrow Stage
 - Wide Stage

Spark

Struttura basica di spark

Cos'è uno Stage?

- Un tipico lavoro Spark consiste in più stage. Ogni stage è una sequenza di trasformazioni e azioni sui dati in ingresso. Quando viene inviato un lavoro Spark, Spark valuta il piano di esecuzione e divide il lavoro in più stage.
- Spark esegue ogni stage in parallelo, dove ciascuno stage può avere più compiti in esecuzione su nodi diversi nel cluster.

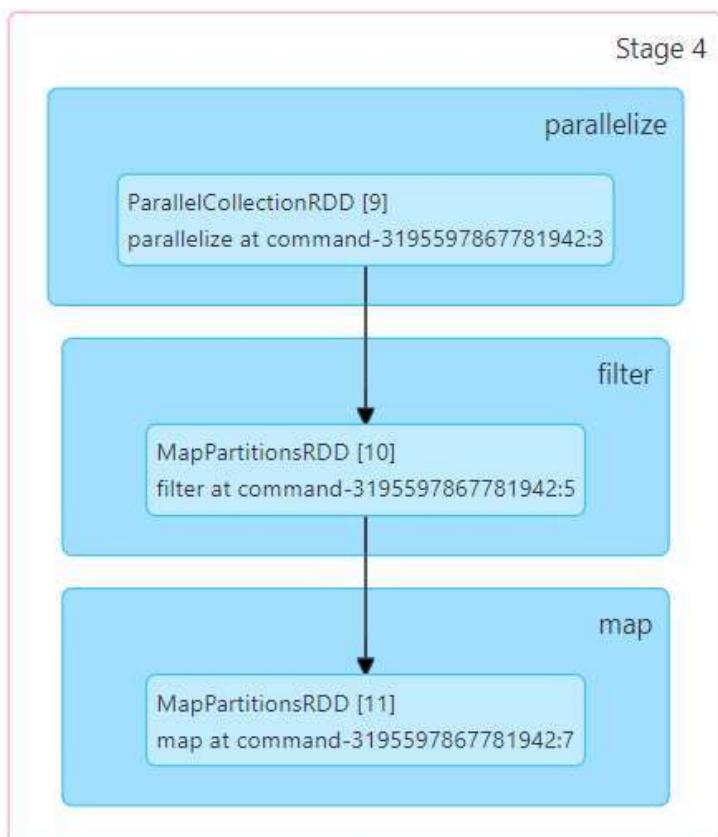


Spark

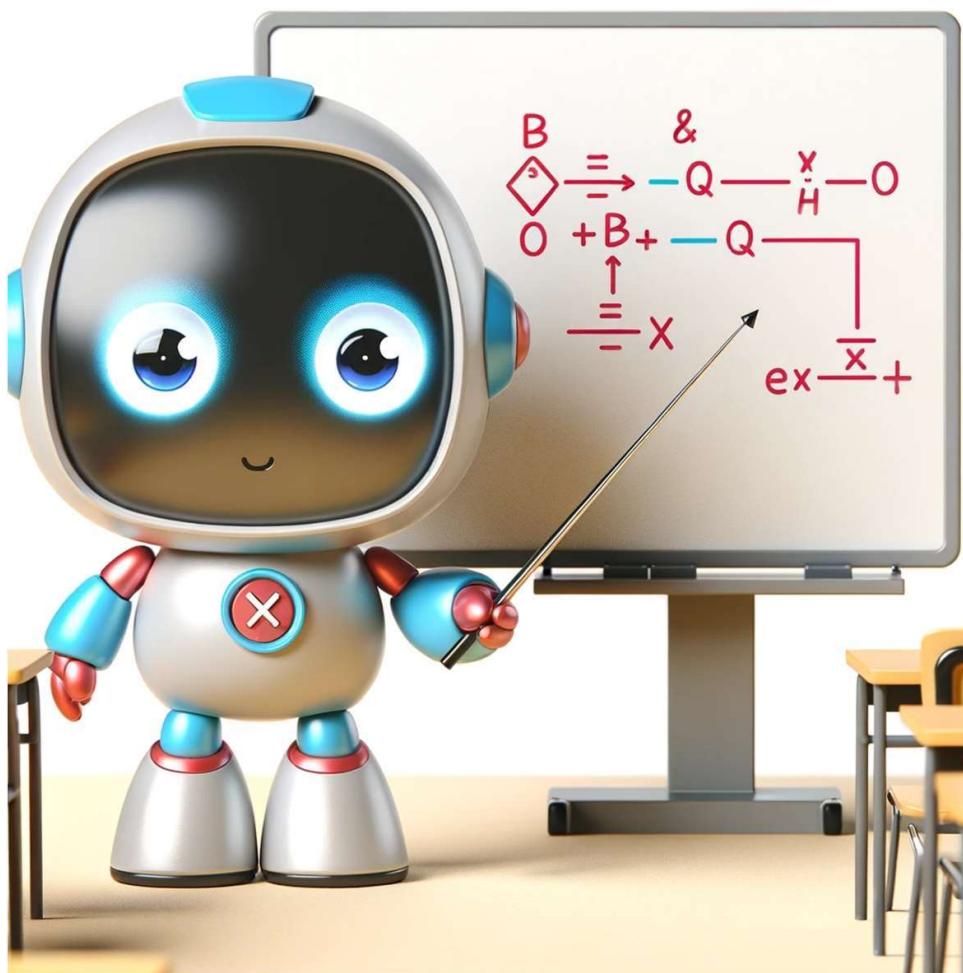
Struttura basica di spark

Cos'è uno Stage?

- In questo esempio, ci sono tre stage :
- Il primo stage è quando l'RDD viene creato usando parallelize.
- Il secondo stage è quando i dati vengono filtrati usando filter.
- Il terzo stage è quando i dati filtrati vengono mappati usando map.
- Il metodo collect attiva l'esecuzione di tutti e tre gli stadi.



Esempio

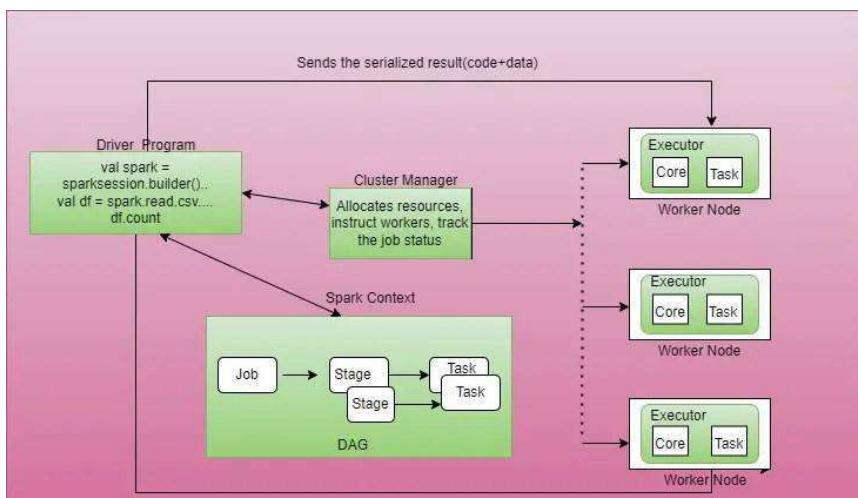


Spark

Struttura basica di spark

Spark Workers

- L'Executor di Spark è un processo che viene eseguito su un nodo **worker** in un cluster Spark ed è responsabile dell'esecuzione dei compiti assegnati ad esso dal programma driver di Spark.
- Punti chiave dell'Executor di Spark:
 - Esegue compiti sui nodi worker come indicato dal Driver.
 - Più Executor vengono eseguiti contemporaneamente in un'applicazione Spark.
 - Creato quando viene creato un `SparkContext` e viene eseguito fino alla terminazione dell'applicazione.
 - Comunica con il Driver per l'assegnazione dei compiti e segnala lo stato dei compiti.

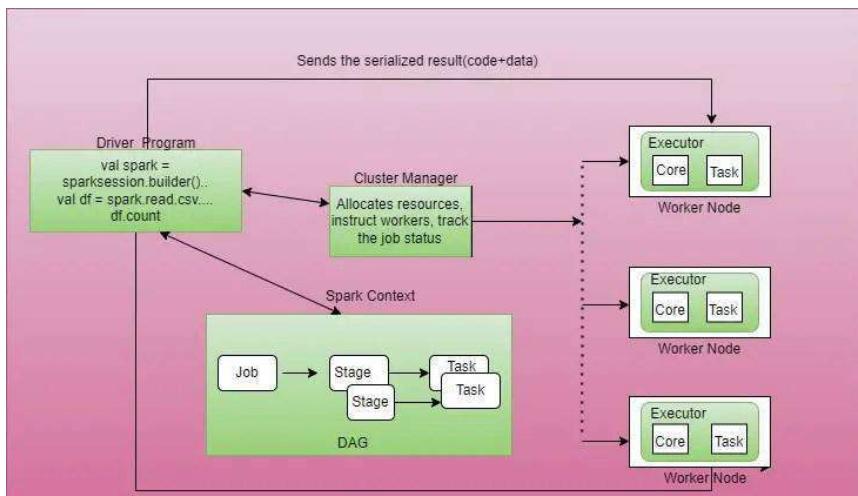


Spark

Struttura basica di spark

Spark Workers

- Gli Executor sono i pilastri di un'applicazione Spark, poiché eseguono le effettive computazioni sui dati.
- Quando un programma driver Spark invia un compito a un cluster, questo viene diviso in unità di lavoro più piccole chiamate "job".
- Ogni Executor viene allocato con una certa quantità di risorse di memoria e CPU quando viene avviato e utilizza questa memoria per memorizzare i dati in memoria per un accesso più veloce durante le computazioni.
- Gli Executor gestiscono anche i dati memorizzati nella cache e sul disco e gestiscono le operazioni di shuffle (quando i dati devono essere scambiati tra i nodi).

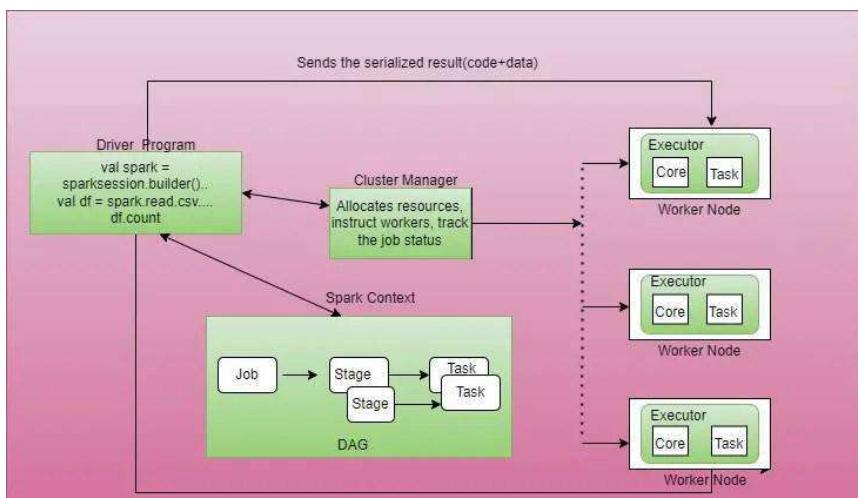


Spark

Struttura basica di spark

Tipi di Esecutori

- In Apache Spark, ci sono diversi tipi di Executor che possono essere utilizzati in base alle esigenze dell'applicazione. Questi sono:
 - **Default Executor.**
 - **Coarse-Grained Executor.**
 - **Fine-Grained Executor.**
 - **External Executors.**
- Ogni tipo di Executor ha vantaggi e svantaggi propri e la scelta dell'Executor dipende dalle esigenze dell'applicazione. Ad esempio, se l'applicazione ha un grande set di dati da elaborare, potrebbe essere più adatto un Coarse-Grained Executor, mentre se l'applicazione ha molti piccoli compiti, potrebbe essere più appropriato un Fine-Grained Executor.

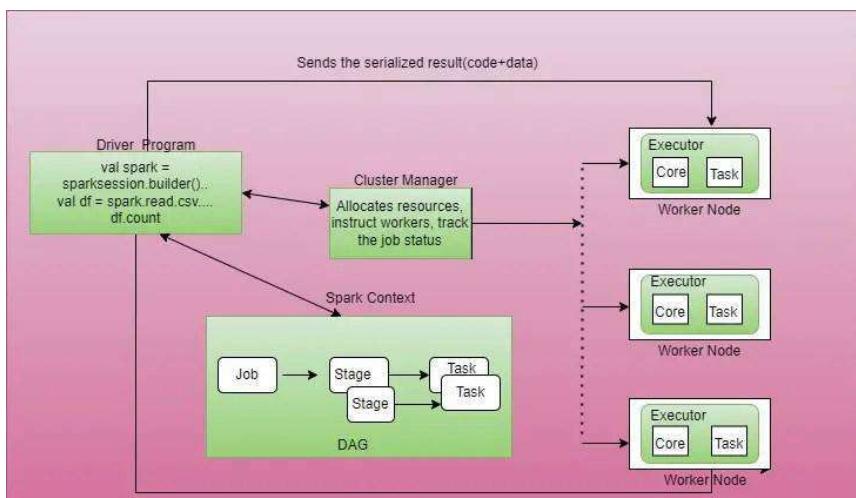


Spark

Struttura basica di spark

Parametri di configurazione

- Apache Spark fornisce una serie di opzioni di configurazione per gli Executor che possono essere utilizzate per ottimizzare le loro prestazioni e l'utilizzo delle risorse. Ecco alcune delle principali opzioni di configurazione:
 - **Memoria dell'Executor.**
 - **Core dell'Executor.**
 - **Numero di Executor.**
 - **Memoria di shuffle.**
- Queste sono solo alcune delle opzioni di configurazione disponibili per gli Executor di Spark. Ottimizzando queste impostazioni è possibile migliorare significativamente le prestazioni e l'utilizzo delle risorse delle applicazioni Spark.



Spark

Struttura basica di spark

Performance degli Esecutori di Spark

- Le prestazioni degli Executor di Spark possono avere un impatto significativo sulle prestazioni complessive di un'applicazione Spark. Ecco alcuni fattori che possono influenzare le prestazioni degli Executor di Spark:

- Memoria.
- CPU.
- Rete.
- Distribuzione dei dati.
- Granularità dei compiti.

