

Databricks y Apache Spark para Big Data: de cero a experto



1

Introducción a Apache Spark

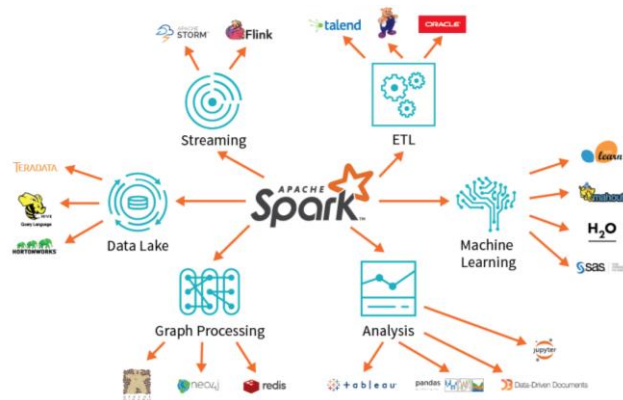


2

Apache Spark

Spark es una solución **Big Data** de **código abierto**. Desarrollado por el laboratorio RAD de **UC Berkeley** (2009).

Se ha convertido en una **herramienta de referencia** en el campo del Big Data.



3

Apache Spark vs MapReduce

Más fácil y rápida que Hadoop MapReduce.

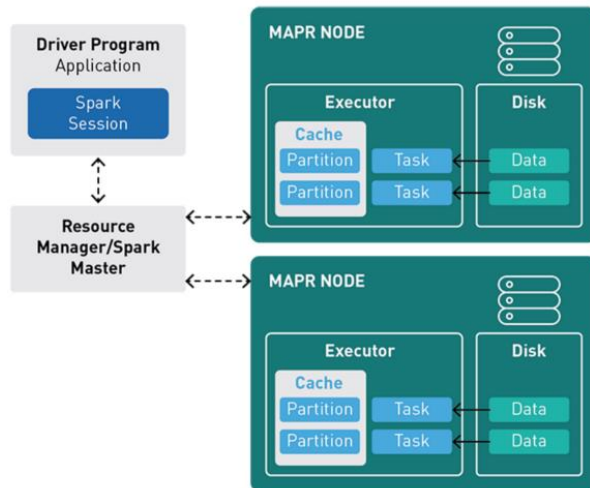
Diferencias:

- **Spark** mucho **más rápido** al almacenar en caché los datos en la **memoria** vs **MapReduce** en el **disco duro** (más lectura y escritura)
- Spark optimizado para un mejor **paralelismo**, utilización **CPU** e inicio más rápido
- Spark tiene modelo de **programación funcional** más rico
- Spark es especialmente útil para **algoritmos iterativos**



4

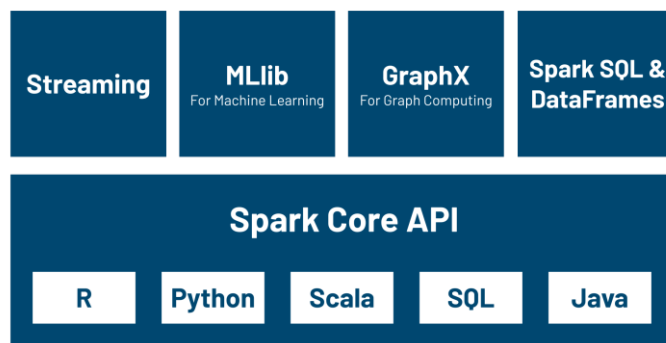
Cómo se Spark en un clúster



5

Componentes de Spark

Spark contiene un **ecosistema** de herramientas **muy completo**.



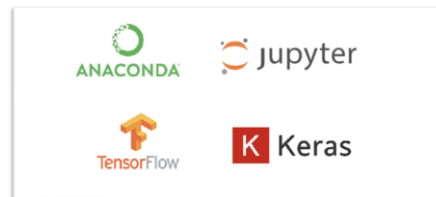
6

PySpark

PySpark es una biblioteca Spark **escrita en Python** para ejecutar la aplicación Python usando las **capacidades de Apache Spark**.

Ventajas de PySpark:

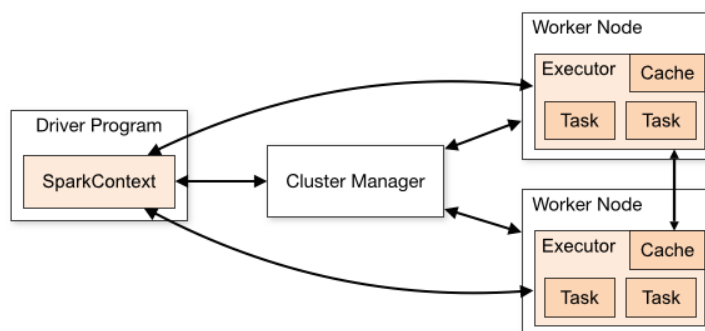
- **Fácil** de aprender
- Amplio conjunto de librerías para **ML y DS**
- Gran apoyo de la **comunidad**



7

Arquitectura de PySpark

Apache Spark funciona en una **arquitectura maestro-esclavo**. Las **operaciones** se ejecutan en los **trabajadores**, y el **Cluster Manager** administra los recursos.



8

Tipos de administradores de clústeres

Spark admite los siguientes administradores de clústeres:

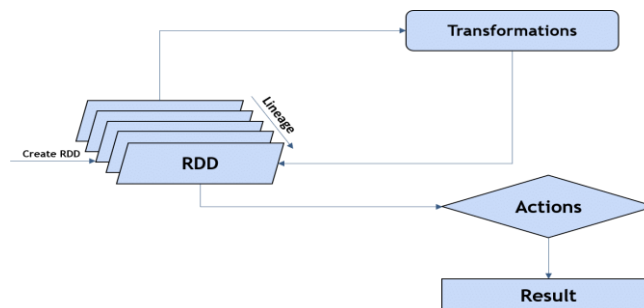
- **Standalone** : administrador de clúster simple
- **Apache Mesos** : es un administrador de clústeres que puede ejecutar también Hadoop MapReduce y PySpark.
- **Hadoop YARN** : el administrador de recursos en Hadoop 2
- **Kubernetes**: para automatizar la implementación y administración de aplicaciones en contenedores.

RDDs de Apache Spark

Apache Spark RDDs

Los RDD son los componentes básicos de cualquier aplicación Spark. RDD significa:

- **Resiliente:** es tolerante a fallos y es capaz de reconstruir datos en caso de fallo.
- **Distribuido:** los datos se distribuyen entre los múltiples nodos de un clúster.
- **Conjunto de datos:** colección de datos particionados con valores.



11

Operaciones en RDDs

Con los RDD, puede realizar dos tipos de operaciones:

- **Transformaciones:** estas operaciones se aplican para crear un nuevo RDD.
- **Acciones:** estas operaciones se aplican en un RDD para indicarle a Apache Spark que aplique el cálculo y devuelva el resultado al controlador.

```

1 num = [1,2,3,4,5]
2
3 num_rdd = sc.parallelize(num)
4 num_rdd.collect()
[1, 2, 3, 4, 5]

```

12

DataFrames en Apache Spark



13

Introducción a DataFrames

Los **DataFrames** son de naturaleza **tabular**. Permiten varios formatos dentro de una misma tabla (**heterogéneos**), mientras que cada variable suele tener valores con un único formato (**homogéneos**).
Similares a las tablas SQL o a las hojas de calculo.

		Column Index		
		2018	2019	2020
Row Index	English	85	60	90
	Math	73	80	64
	Science	98	58	74
	French	88	96	87

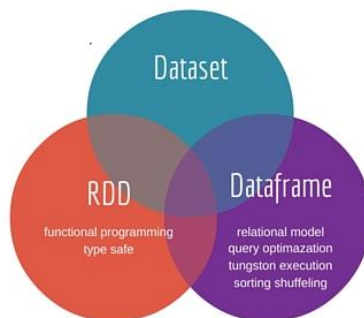


14

Ventajas de los DataFrames

Algunas de las ventajas de trabajar con Dataframes en Spark son:

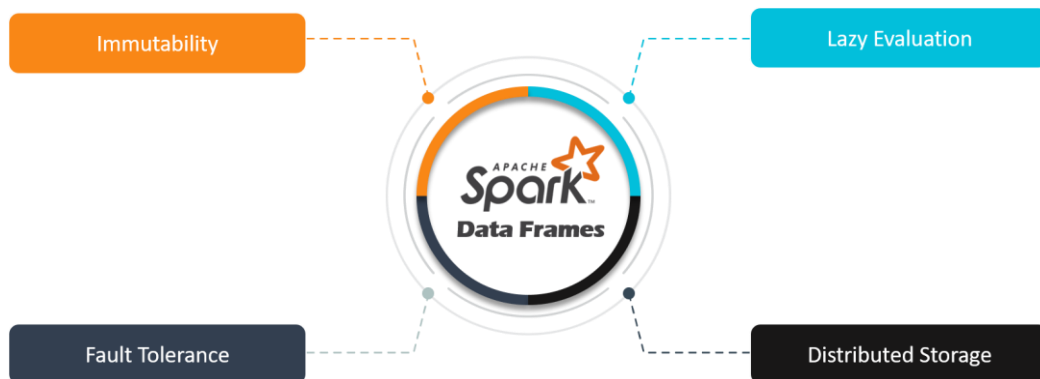
- Capacidad de procesar una **gran cantidad de datos** estructurados o semiestructurados
- Fácil **manejo de datos** e imputación de valores faltantes
- Múltiples formatos como **fuentes de datos**
- Compatibilidad con **múltiples lenguajes**



15

Características de los DataFrames

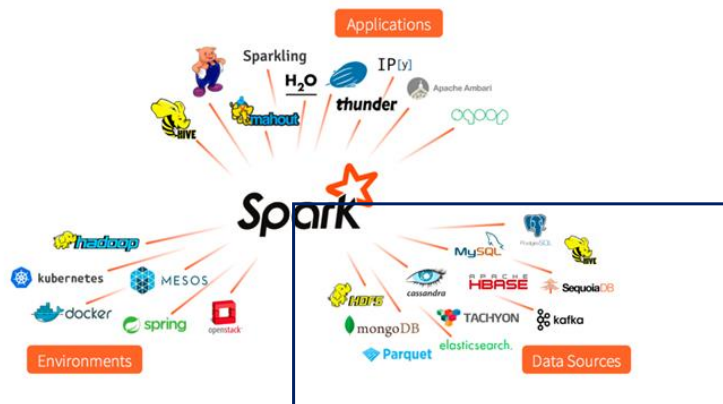
Los **DataFrames** de Spark **se caracterizan** por: ser distribuidos, evaluación perezosa, inmutabilidad y tolerancia a fallos.



16

Fuentes de datos de DataFrames

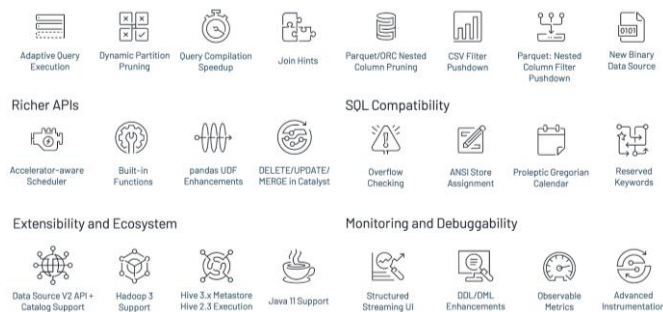
Los marcos de datos en Pyspark se pueden crear de varias formas: a través de archivos, utilizando RDDs o a través de bases de datos.



Funciones avanzadas de Spark

Funciones avanzadas

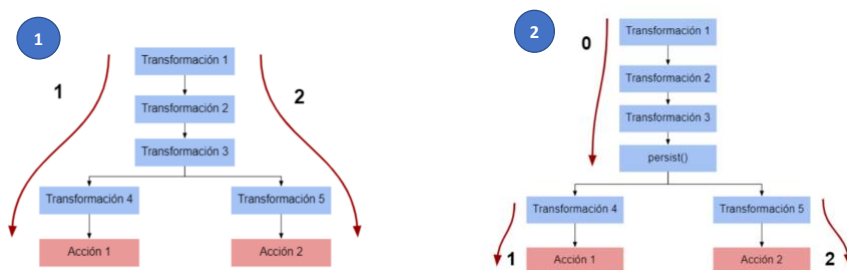
Spark contiene numerosas **funciones avanzadas** para optimizar su rendimiento y realizar transformaciones complejas en los datos. Algunas de ellas son: las expresiones de selectExpr(), UDF, cache(), etc



19

Optimización del rendimiento

Una de las **técnicas de optimización** son los métodos **cache()** y **persist()**. Estos métodos se usan para **almacenar un calculo intermedio** de un RDD, DataFrame y Dataset para que puedan reutilizarse en acciones posteriores.



20

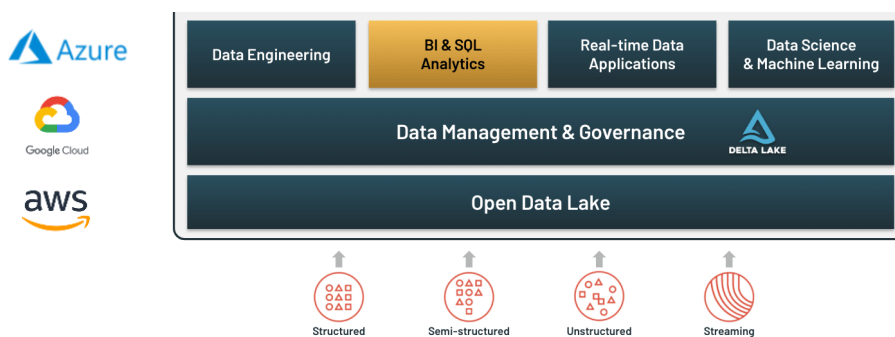
Databricks

21

Introducción a Databricks

Databricks es la **plataforma analítica de datos** basada en Apache **Spark** desarrollada por los precursores de Spark. Permite analítica avanzada, Big Data y ML de forma **sencilla** y **colaborativa**.

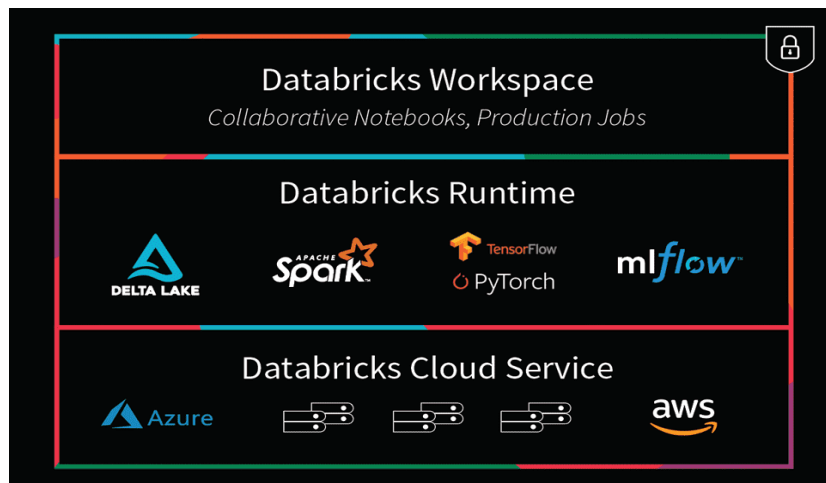
Disponible como **servicio cloud** en **Azure**, **AWS** y **GCP**.



22

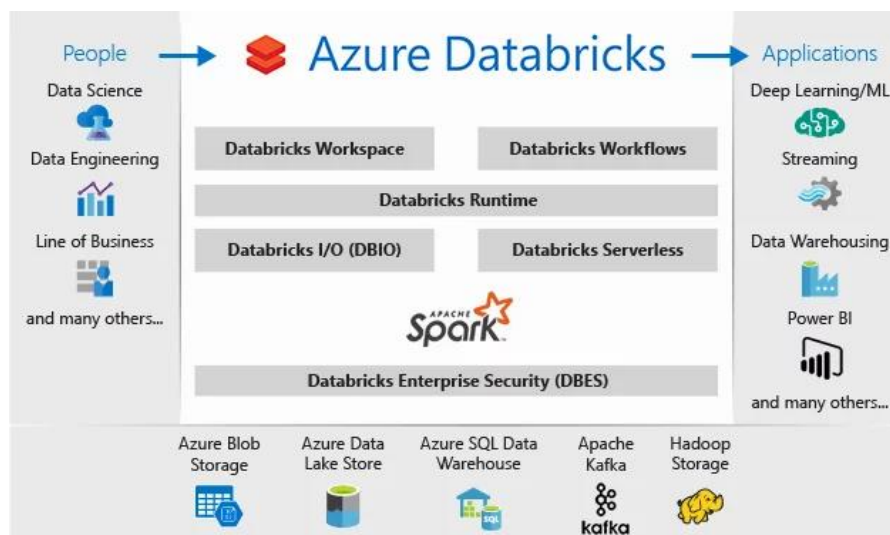
Características de Databricks

Permite **auto-escalar** y dimensionar **entornos de Spark** de **forma sencilla**. Facilita los despliegues y se acelera la instalación y configuración de los entornos.



23

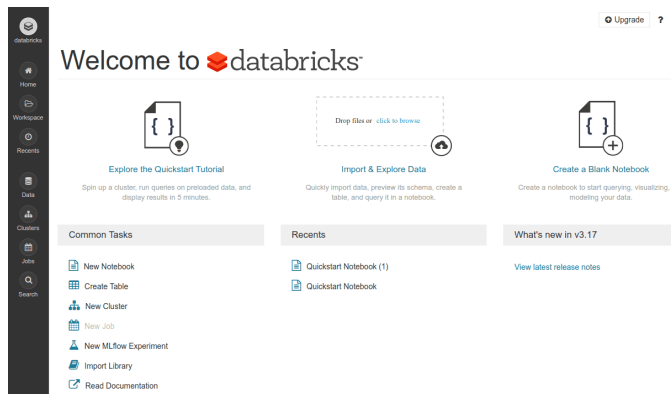
Arquitectura de Databricks



24

Databricks Community

Databricks community es la versión **gratuita**. Permite usar un **pequeño clúster** con recursos limitados y **notebooks no colaborativos**. La versión de pago aumenta las capacidades.

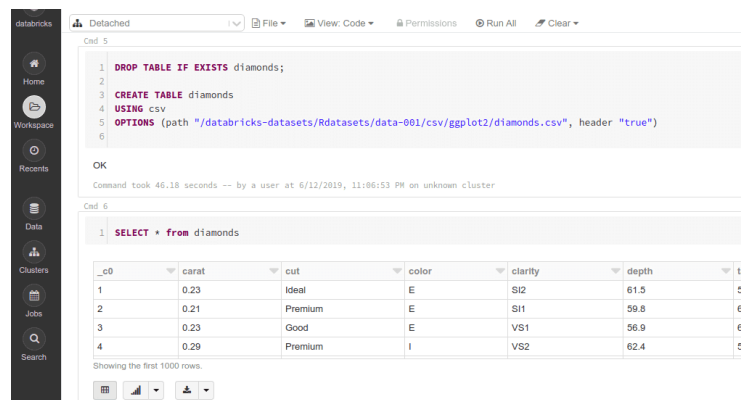


25

Terminología

Términos importantes que debemos conocer:

1. Workspaces
2. Notebooks
3. Librerías
4. Tablas
5. Clusters
6. Jobs



26

Machine Learning con Spark

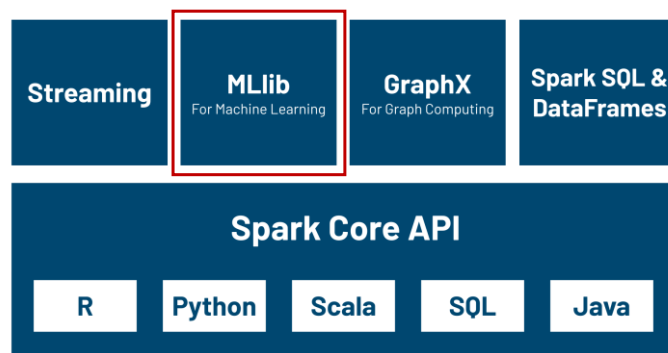


27

Spark Machine Learning

Machine Learning: es la construcción de **algoritmos** que pueden aprender de los datos y hacer predicciones sobre ellos.

Spark MLlib se usa para realizar aprendizaje automático en Apache Spark. MLlib consta de algoritmos y funciones habituales.

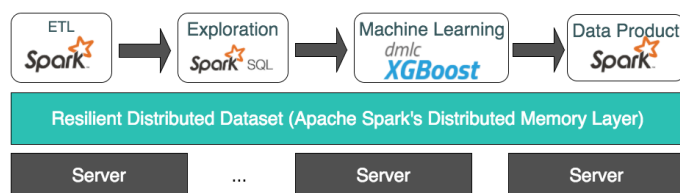


28

Herramientas Spark Machine Learning

Herramientas de MLlib:

- **spark.mllib** contiene la API original construida sobre RDD
- **spark.ml** proporciona una API de nivel superior construida sobre DataFrames para construcción de pipelines de ML. La API de ML principal.



Fuente: <https://www.r-bloggers.com/>

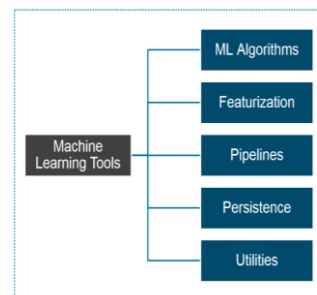


29

Componentes Spark Machine Learning

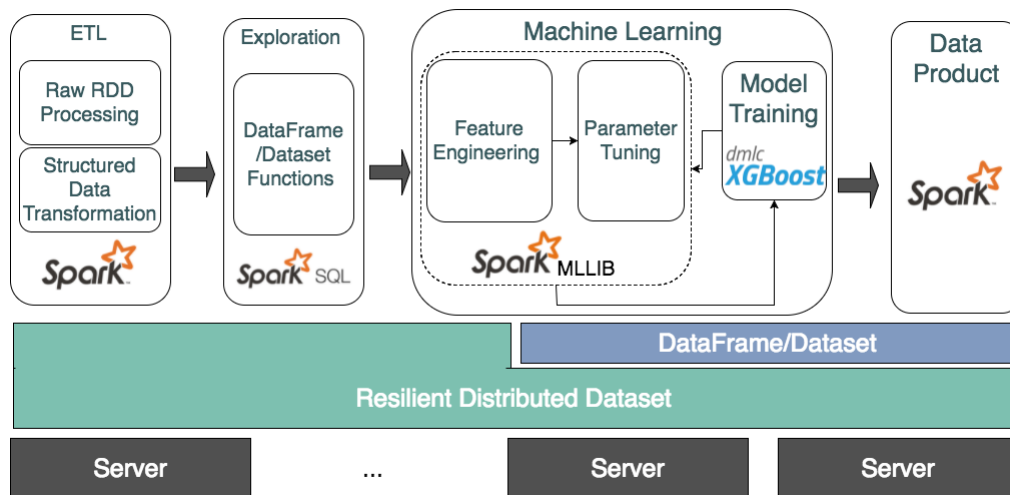
Spark MLlib proporciona las siguientes herramientas:

- **Algoritmos ML:** Incluyen algoritmos de aprendizaje comunes como clasificación, regresión, agrupamiento y filtrado colaborativo.
- **Caracterización:** Incluye: extracción, transformación, reducción de dimensionalidad y selección de características.
- **Pipelines:** son herramientas para construir modelos de ML en etapas.
- **Persistencia:** permite guardar y cargar algoritmos, modelos y pipelines.
- **Utilidades:** para álgebra lineal, estadística y manejo de datos.



30

Proceso de Machine Learning



Fuente: <https://www.r-bloggers.com/>



31

Ingeniería de características con Spark

Las técnicas de preprocesamiento de datos más utilizadas en los enfoques de Spark son las siguientes

- VectorAssembler
- Agrupamiento
- Escalado y normalización
- Trabajar con características categóricas
- Transformadores de datos de texto
- Manipulación de funciones
- PCA



32

Ingeniería de características con Spark

- **Vector Assembler:** Se utiliza básicamente para concatenar todas las características en un solo vector que se puede pasar al estimador o al algoritmo ML.
- **Agrupamiento:** es el método más sencillo para convertir las variables continuas en variables categóricas. Se puede realizar con la clase Bucketizer.
- **Escalado y normalización:** es otra tarea común en variables continuas. Permite que los datos tengan una distribución normal.
- **MinMaxScaler y StandardScaler:** estandarizan las características con una media cero y una desviación estándar de 1.
- **StringIndexer :** para convertir características categóricas en numéricas.

```
+ ---+ + ---+ + ---+ + -----+
| int1 | int2 | int3 | caracteristicas |
+ ---+ + ---+ + ---+ + -----+
| 7 | 8 | 9 | [7.0,8.0,9.0] |
| 1 | 2 | 3 | [1.0,2.0,3.0] |
| 4 | 5 | 6 | [4.0,5.0,6.0] |
+ ---+ + ---+ + ---+ + -----+
```



33

Pipelines en PySpark

En los **Pipelines** (canalizaciones) las diferentes **etapas del trabajo** de aprendizaje automático se pueden agrupar como una sola entidad y se pueden considerar como un flujo de trabajo ininterrumpido.

Cada etapa es un **Transformador** . Se ejecutan en **secuencia** y los datos de entrada se transforman mientras pasan por cada etapa.



```
tokenizer = Tokenizer(inputCol="SystemInfo", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.01)

# Build the pipeline with our tokenizer, hashingTF, and logistic regression stag
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

model = pipeline.fit(training)
```



34

Apache Spark Koalas



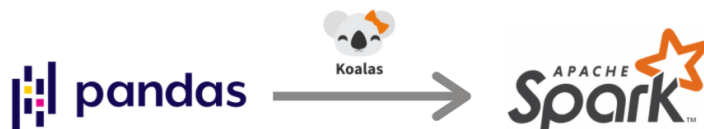
35

Introducción a Koalas

Koalas proporciona un **reemplazo directo de Pandas**, lo que permite un escalado eficiente a cientos de nodos para la ciencia de datos y el Machine Learning.

Pandas no se escala a Big data.

PySpark DataFrame es más compatible con **SQL** y **Koalas DataFrame** está más cerca de **Python**

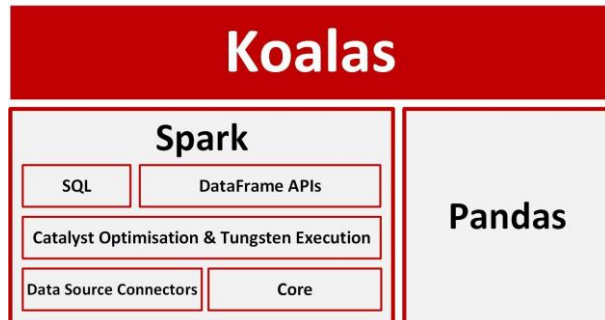


36

Koalas y PySpark DataFrames

Koalas y PySpark DataFrames son diferentes. **Koalas** DataFrames sigue la **estructura de Pandas** e implementa un **índice**. El **PySpark DataFrame** es más compatible con las tablas en las **bases de datos relacionales** y no tiene índices.

Koalas traduce las API de pandas al plan lógico de **Spark SQL**.



37

Ejemplo: Ingeniería de características con Koalas

En ciencia de datos a menudo se necesita la función de **get_dummies()** de **pandas** para codificar variables categóricas como variables ficticias (numéricas).

Gracias a Koalas se puede hacer esto en Spark con solo unos pocos ajustes.

Pandas

```
import pandas as pd
data = pd.read_csv("fire_department_calls_sf_clean.csv", header=0)
display(pd.get_dummies(data))
```

Koalas

```
import databricks.koalas as ks
data = ks.read_csv("fire_department_calls_sf_clean.csv", header=0)
display(ks.get_dummies(data))
```

	Call_Me?	Money	Target		Money	Call_Me?_Maybe	Call_Me?_No	Call_Me?_Yes
0	Yes	5	10	→	0	5	0	1
1	No	3	4		1	3	1	0
2	Maybe	5	5		2	5	1	0
3	Yes	10	7		3	10	0	1
4	Yes	9	9		4	9	0	1

38

Ejemplo: Ingeniería de características con Koalas

En ciencia de datos a menudo se necesita trabajar con **datos de tiempo**. Pandas permite trabajar con este tipo de datos de forma fácil, en PySpark es más complicado.

	End_date	Start_date
0	2013-03-17 21:45:00	2012-01-31 12:00:00
1	2013-03-24 21:45:00	2012-02-29 12:00:00
2	2013-03-31 21:45:00	2012-03-31 12:00:00
3	2013-04-07 21:45:00	2012-04-30 12:00:00
4	2013-04-14 21:45:00	2012-05-31 12:00:00
5	2013-04-21 21:45:00	2012-06-30 12:00:00
6	2013-04-28 21:45:00	2012-07-31 12:00:00



	End_date	Start_date	diff_seconds
0	2013-03-17 21:45:00	2012-01-31 12:00:00	35545500.0
1	2013-03-24 21:45:00	2012-02-29 12:00:00	33644700.0
2	2013-03-31 21:45:00	2012-03-31 12:00:00	31571100.0
3	2013-04-07 21:45:00	2012-04-30 12:00:00	29583900.0
4	2013-04-14 21:45:00	2012-05-31 12:00:00	27510300.0
5	2013-04-21 21:45:00	2012-06-30 12:00:00	25523100.0
6	2013-04-28 21:45:00	2012-07-31 12:00:00	23449500.0

Pandas

```
df['diff_seconds'] = df['End_date'] - df['Start_date']
df['diff_seconds'] = df['diff_seconds']/np.timedelta64(1,'s')
print(df)
```

Koalas

```
import databricks.koalas as ks
df = ks.from_pandas(pandas_df)
df['diff_seconds'] = df['End_date'] - df['Start_date']
df['diff_seconds'] = df['diff_seconds'] / np.timedelta64(1,'s')
print(df)
```

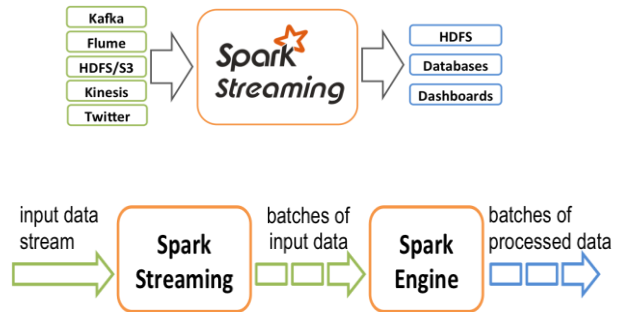
Spark Streaming

Fundamentos Spark Streaming

PySpark Streaming es un sistema escalable y tolerante a fallos que sigue el paradigma de **lotes RDD**.

Opera en intervalos de lotes, recibiendo un **flujo de datos de entrada continuo** de fuentes como Apache Flume , Kinesis, Kafka, sockets TCP, etc.

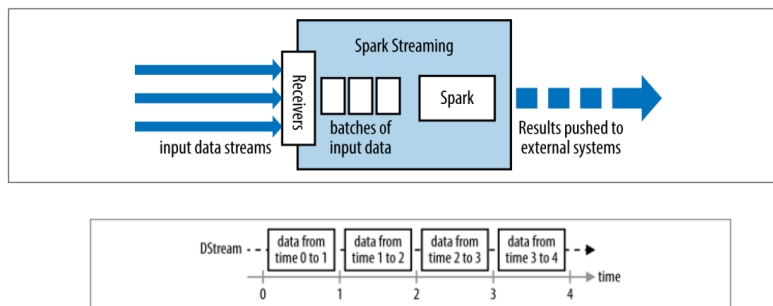
Spark Engine se encarga de procesarlos.



41

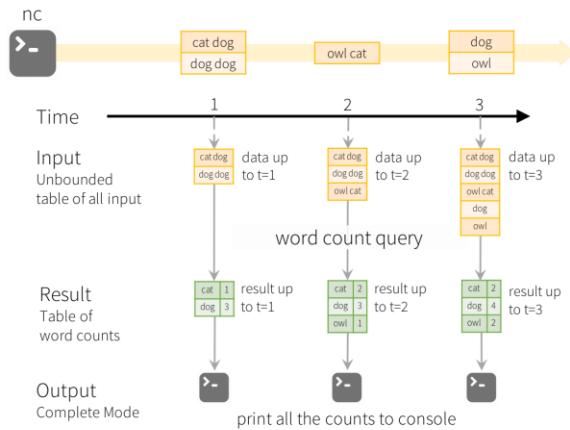
Funcionamiento Spark Streaming

Spark Streaming recibe datos de varias fuentes y los agrupa en pequeños lotes (**Dstreams**) en un intervalo de tiempo. El usuario puede definir el **intervalo**. Cada lote de entrada forma un RDD y se procesa mediante trabajos de Spark para crear otros RDD.



42

Ejemplo: contar palabras



43

Modos de salida

Spark usa varios modos de salida para almacenar los datos:

- **Modo completo (Complete):** toda la tabla se almacenará
- **Modo de adición (Append):** solo las nuevas filas del último proceso se almacenarán. Solo para las consultas en las que no se espera que cambien las filas existentes.
- **Modo de actualización (Update):** solo las filas que se actualizaron desde el último proceso se almacenarán. Este modo solo genera las filas que han cambiado desde el último proceso. Si la consulta no contiene agregaciones, será equivalente al modo append.

Complete,
Append,
Update

```
query = wordCounts \
    .writeStream \
    .outputMode("complete") \
    .format("console") \
    .start()
```

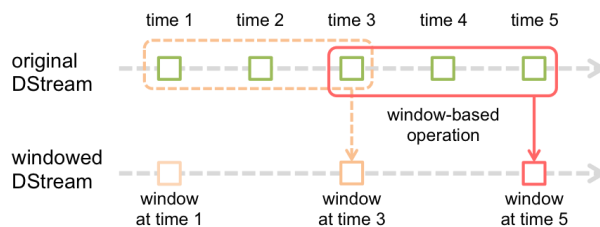
44

Tipos de transformaciones

Para **tolerancia a fallos** los datos recibidos se copian en dos nodos y hay también un mecanismo llamado **checkpointing**.

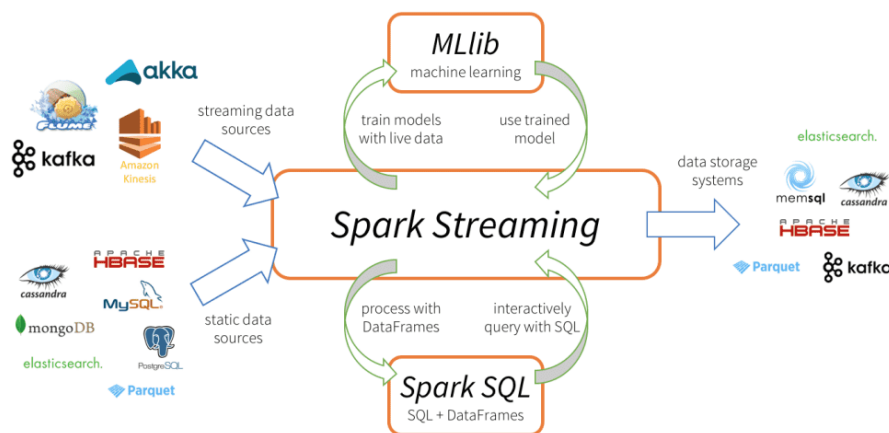
Las transformaciones se pueden agrupar en :

- **sin estado**: no depende de los datos de lotes anteriores.
- **con estado**: utilizan datos de lotes anteriores



45

Capacidades de Spark Streaming



46

Delta Lake



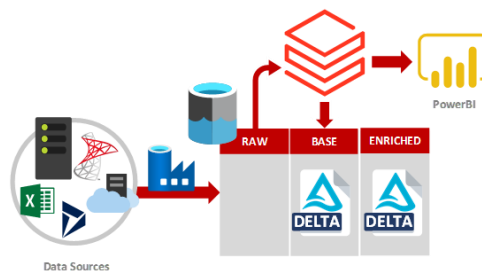
47

Introducción a Delta Lake

Delta Lake es la **capa de almacenamiento** open source desarrollada para **Spark** y **Databricks**.

Proporciona **transacciones ACID** y gestión avanzada de **metadatos**.

Incluye un motor de consultas compatible con Spark que permite **acelerar las operaciones** y mejoran el rendimiento. Los datos almacenados en formato **Parquet**.

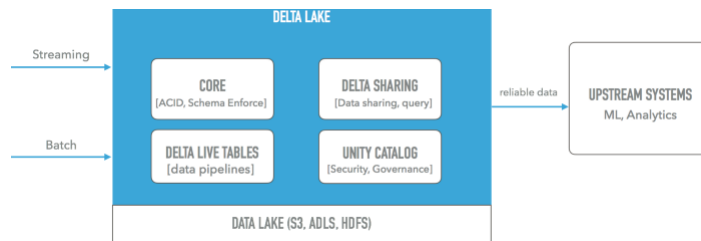


48

Funcionamiento Delta Lake

Delta Lake se basa principalmente en dos tecnologías: **Apache Spark y Apache Parquet**.

- De Spark tenemos las **APIs**. Estas aseguran que se mantenga la confiabilidad de los datos mediante transacciones en ACID, aplicación de esquemas, etc.
- Apache **Parquet** se utiliza para almacenar los metadatos utilizados para la calidad de los datos, transacciones ACID, gobernanza, control de versiones, etc.

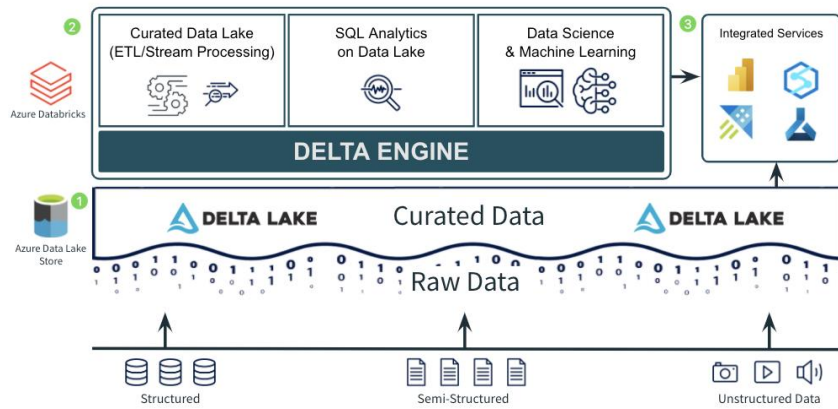


Funcionamiento Delta Lake

Delta Lake ofrece:

- **Transacciones ACID:** garantizan que no haya datos inconsistentes.
- **Manejo escalable de metadatos:** utiliza el procesamiento distribuido de Spark para manejar todos los metadatos de petabytes de datos
- **Transmisión y unificación de lotes**
- **Aplicación del esquema:** gestiona las variaciones del esquema y evita la inserción de registros incorrectos
- **Viaje en el tiempo:** tiene control de versiones que permite reversiones, auditoría de históricos, experimento de ML reproducibles
- **Actualizaciones y eliminaciones:** admite operaciones de fusión, actualización y eliminación

Arquitectura Delta Lake con Azure



51

Recursos

52

Recursos:

- <https://spark.apache.org/docs/2.2.0/index.html> Documentación oficial de Spark
- <https://colab.research.google.com/> Google Colab para poder tener capacidad de computo adicional