/bluetab

APACHE Spark™

Agosto 2021

/bluetab
an IBM Company

Antes de iniciar

Spark es un mundo muy grande.

Teoría

Práctica

Agenda

1. ¿Qué es Spark?
2. Principales características.
3. Arquitectura.
4. Dataframes / DataSet / RDD
5. Optimizador.
6. Componentes.
7. Transformaciones.
8. Acciones.
9. Lazy evaluation.
10. Manos a la obra.

¿Qué es Spark?

- Es una plataforma de cómputo en clúster (distribuido), diseñada para ser rápida.
- Es escalable y diseñada para soportar grandes volúmenes de procesamiento
- de datos.
- El nacimiento de Spark surge en los laboratorios AMPLab de la Universidad de Berkeley en 2009
- Basado en Scala

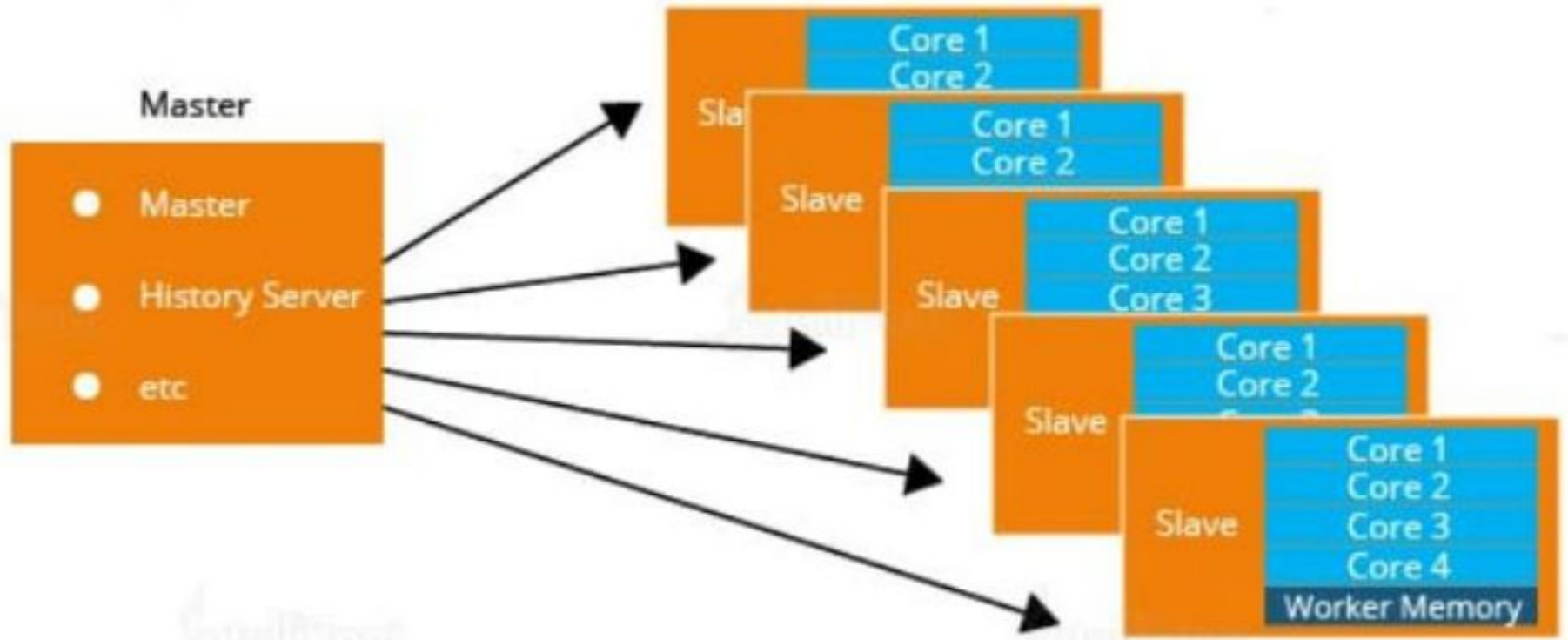
Principales características

a. Spark basa en el modelo "MapReduce" de Hadoop, implementando eficientemente los cálculos en RAM y el procesamiento de grandes volúmenes de información.
b. Spark permite ejecutar aplicaciones "en memoria" 100 veces más rápido que el modelo MapReduce y 10 veces más rápido ejecutando en "disco"
c. Tolerante a fallos.
d. Spark tiene APIs incorporadas para SQL, R, Scala, Python y Java.
e. Tiene componentes especializados para varias cargas de trabajo, como: Spark SQL, Spark Streaming, MLlib, GraphX, entre otras
f. Existe una gran flexibilidad e interconexión con otros módulos de Apache como Hadoop, Hive o Kafka

Arquitectura

Maestro - Esclavos

Administra y coordina la ejecución de tareas en un grupo de computadoras o recursos de manera distribuida



Dataframes / DataSet / RDD

Un RDD (Resilient Distributed Datasets), se define como una colección de elementos que es tolerante a fallos y que es capaz de operar en paralelo (distribuido).
Es la principal abstracción de datos dentro de Spark.
Prácticamente todo en Spark se basa en RDD.
Todo se construyen sobre RDD y se compilan en estas herramientas de bajo nivel para una ejecución distribuida conveniente y extremadamente eficiente.

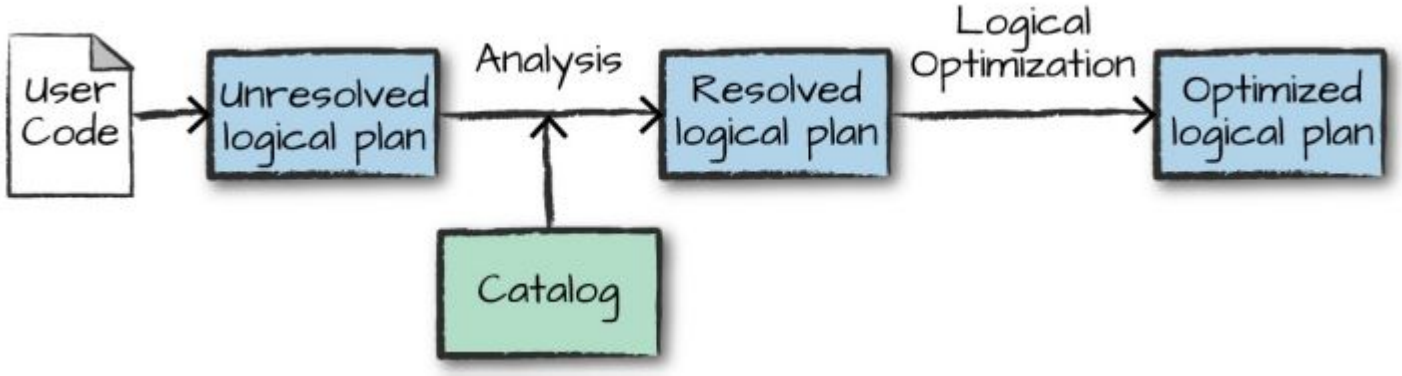
Dataframes / DataSet / RDD

Spark tiene dos nociones de colecciones estructuradas: **DataFrames** y **Datasets**. Son colecciones (distribuidas) en forma de tabla con filas y columnas bien definidas.
Cada columna debe tener el mismo número de filas que todas las demás columnas y cada columna tiene información de tipo que debe ser coherente para todas las filas de la colección.

Los dataframes Spark solo verifica si los tipos de datos se alinean con los especificados en el esquema en tiempo de ejecución.
Los dataset, por otro lado, comprueban si los tipos se ajustan a la especificación en el momento de la compilación.

Los dataset solo existen en lenguajes como Java/Scala

Optimizador

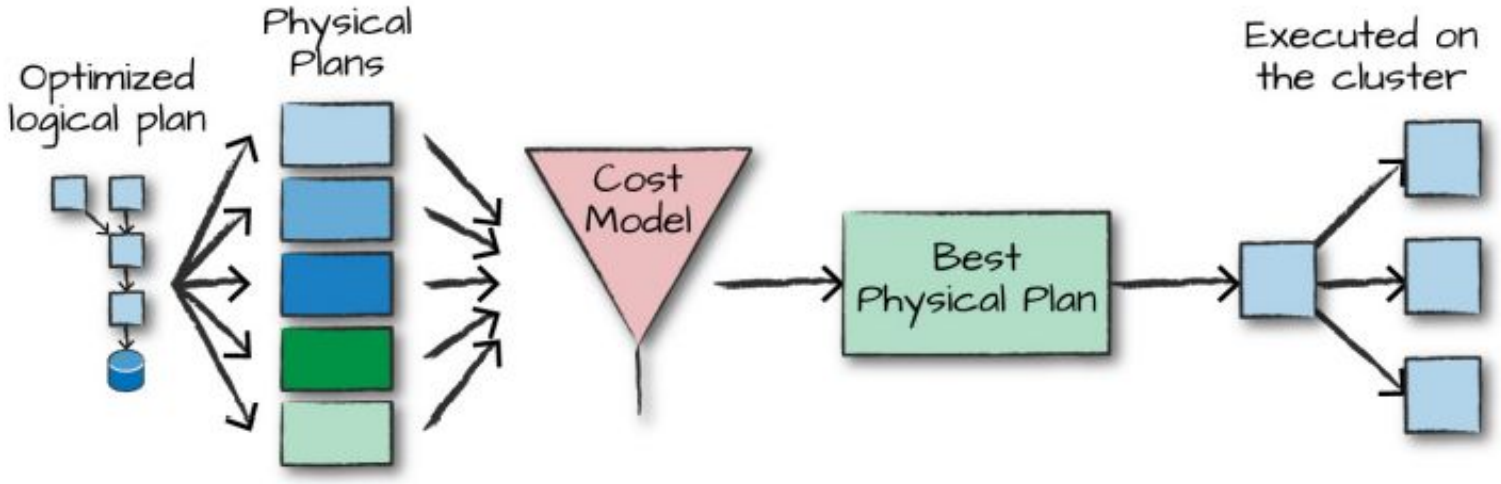


Busca convertir el conjunto de expresiones del usuario en la versión más optimizada, esto lo hace convirtiendo el código de usuario en un **plan lógico no resuelto**. Este plan no está resuelto porque, aunque su código puede ser válido, las tablas o columnas a las que hace referencia pueden existir o no.

Spark usa el **catálogo**, un repositorio de toda la información de tablas y DataFrame, para resolver columnas y tablas en el analizador. El analizador puede rechazar el plan lógico no resuelto si el nombre de columna o tabla requerida no existe en el catálogo. Si el analizador puede resolverlo, el resultado pasa a través de **Catalyst Optimizer**, una colección de reglas que intentan optimizar el plan lógico.

Esto no hace magia.

Componentes



Después de crear con éxito un plan lógico optimizado, Spark comienza el proceso de planificación física. El plan físico, especifica cómo se ejecutará el plan lógico en el clúster, generando diferentes estrategias de ejecución física y comparándolas a través de un modelo de costos. La planificación física da como resultado una serie de RDD y transformaciones. Este resultado que hace referencia a Spark como un compilador: toma consultas en DataFrames, Datasets y SQL y las compila en transformaciones RDD

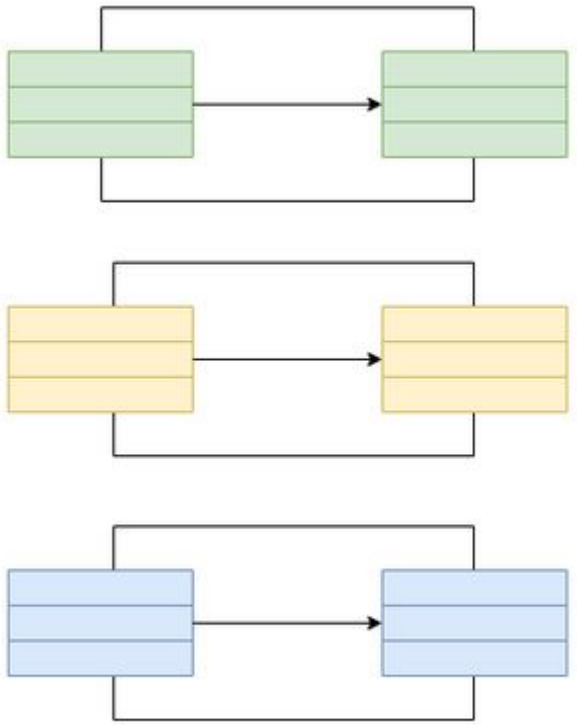
Transformaciones

En Spark, las estructuras de datos core son **inmutables**, lo que significa que no se pueden cambiar una vez creadas. Esto puede parecer un concepto extraño al principio: si no puede cambiarlo, **¿cómo se supone que debe usarlo?** Para "cambiar" un DataFrame, debe indicarle a Spark cómo le gustaría modificarlo para que haga lo que quiera. Estas instrucciones se denominan transformaciones.
Las transformaciones son el núcleo de cómo expresa su lógica empresarial con Spark.
Las transformaciones no son aplicadas sobre el DataFrame original si no que, se crea un nuevo DF que contiene los cambios realizados a los datos del origen.

Se obtiene como resultado un mismo dataframe / dataset / rdd

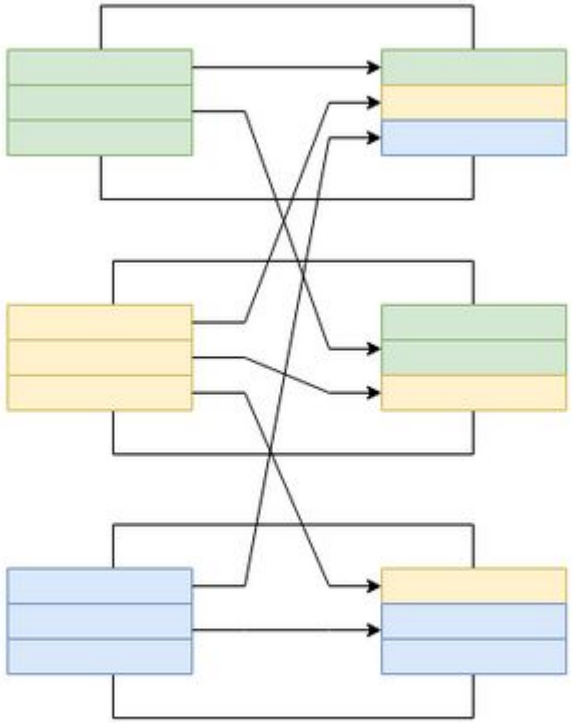
Transformaciones - Narrow

Se utiliza cuando los datos que se necesitan tratar están en la misma partición y no es necesario realizar una mezcla de dichos datos para obtenerlos todos. Ejemplo *Filter*, *union*.



Transformación narrow

Transformaciones - Wide



Transformación wide

Se utiliza cuando la lógica de la aplicación necesita datos que se encuentran en diferentes particiones y es necesario mezclar dichas particiones para agrupar los datos necesarios. Distinct, groupBy, join, etc

Acciones

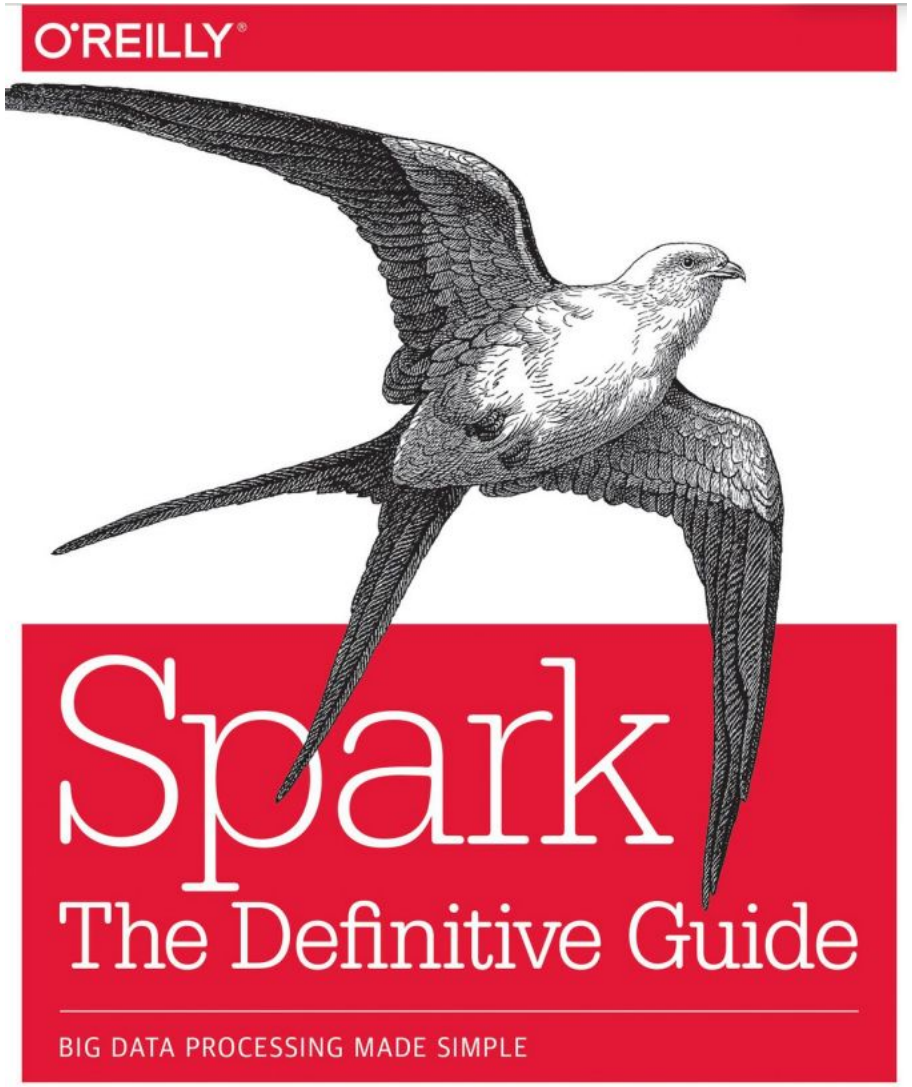
Una acción consiste simplemente en aplicar una operación y **obtener un valor** como resultado, que dependerá del tipo de operación. Por ejemplo: reduce, collect, **count**, first, foreach, la escritura, etc.

Lazy Evaluation

La evaluación perezosa significa que Spark esperará hasta el último momento para ejecutar el procesamiento. En lugar de modificar los datos inmediatamente cuando expresa alguna transformación, crea un plan que le gustaría aplicar a sus datos. Esto permite encontrar el plan más óptimo para la ejecución.

Aunque se encadenan las transformaciones, solo se ejecuta al realizar una acción.

Mi recomendación



Referencias

- Spark: un destello en el universo Big Data
  https://www.paradigmadigital.com/dev/spark-un-destello-en-el-universo-big-data/

- Apache Spark: Introducción, qué es y cómo funciona
  https://www.esic.edu/rethink/tecnologia/apache-spark-introduccion-que-es-y-como-funciona

- Spark the definitive guide. Big data processing made simple. Bill Chamber & Matei Zaharia.

- Apache Spark
  https://spark.apache.org/docs/latest/

Manos a la obra

Un formato de almacenamiento en columnas disponible para cualquier proyecto en el ecosistema de Hadoop, independientemente de la elección del framework de procesamiento de datos, modelo de datos o lenguaje de programación.

El almacenamiento en columnas está optimizado para seleccionar una sección de ellas.

Manos a la obra

La base fundamental del formato son los esquemas. Siempre que se lee un formato .avro está presente el esquema con el que han sido escritos, esto permite aumentar el rendimiento al escribir los datos, haciendo la serialización rápida y viable en espacio.

Los esquemas de Avro están definidos en JSON para facilitar la implementación con los lenguajes de programación.

Manos a la obra

Pero antes....

Conoce tus datos

https://github.com/JuanPabloArevalo/bluetabUniversitySpark

https://mvnrepository.com/artifact/org.apache.spark/spark-avro_2.12/3.1.2

/bluetab

an IBM Company

BLUETAB MADRID

Poeta Joan Maragall, 1
28020 Madrid
España
T: +34 914 571 697

BLUETAB CDMX

Av. Homero 440, Int. 503
Col. Polanco. Chapultepec
CP 11560, CDMX
México
T: +521 55 75771781

BLUETAB LIMA

Av. Javier Prado Este
560 Oficina
1401, San Isidro Lima
Perú
T: +51 (1) 391-8315

BLUETAB BOGOTÁ

Cra. 7 #71-21
Torre B, piso 12
Bogotá DC
Colombia
T: +57 (1) 3135831-32