STRATIO®

SparkSQL:
de lo básico a la optimización

t3chfest 2019

Miguel Ángel Fernández Díaz

Presentación

- Formación:
  - Ingeniero Superior en Informática en la URJC
  - Erasmus: Universidad de Groningen
  - Executive Master en Innovación en la EOI
- Experiencia laboral:
  - Iberia L.A.E.: Desarrollador Java y Desarrollador BI
  - Cediant: Arquitecto Java, HPC y primer contacto con Hadoop
  - Stratio: Arquitecto Big Data
- Otros:
  - Profesor en el Master de Big Data de la U-Tad
  - Meetups
  - Codemotion

© Stratio 2018. Confidential, All Rights Reserved.

Agenda

- Introducción a Spark
- Spark SQL
- Optimizaciones
- Turno de preguntas

© Stratio 2018. Confidential, All Rights Reserved.

Introducción a Spark

STRATIO

Introducción: ¿Qué es Spark?

- Es un motor de procesamiento para entornos Big Data como Data Lakes que aprovecha el procesamiento en **memoria distribuido** para realizar análisis de datos de forma eficiente y con un alto rendimiento
- Es uno de los proyectos más populares dentro del Apache Foundation
- Incluído en las plataformas más importantes de Big Data como Cloudera, Hortonworks, MapR y Stratio
- Proyecto de **código abierto**, sostenido por Databricks principalmente, y que se ejecuta en la JVM

© Stratio 2018. Confidential, All Rights Reserved.

Introducción: ¿Cómo surge Spark?

Berkeley
UNIVERSITY OF CALIFORNIA

- Originalmente desarrollado por la UC Berkeley en 2009 como proyecto de investigación de Matei Zahara (AMPLab)
- Surgió como “ejemplo” de framework sobre Apache Mesos
- Su primera aplicación fue para monitorizar y predecir el tráfico en la Bahía de San Francisco
- En 2013, su incipiente comunidad le lleva a formar parte del programa Apache incubator
- En 2014, su estabilidad y ritmo de desarrollo le lleva a convertirse en un proyecto Apache Top Level

© Stratio 2018. Confidential, All Rights Reserved.

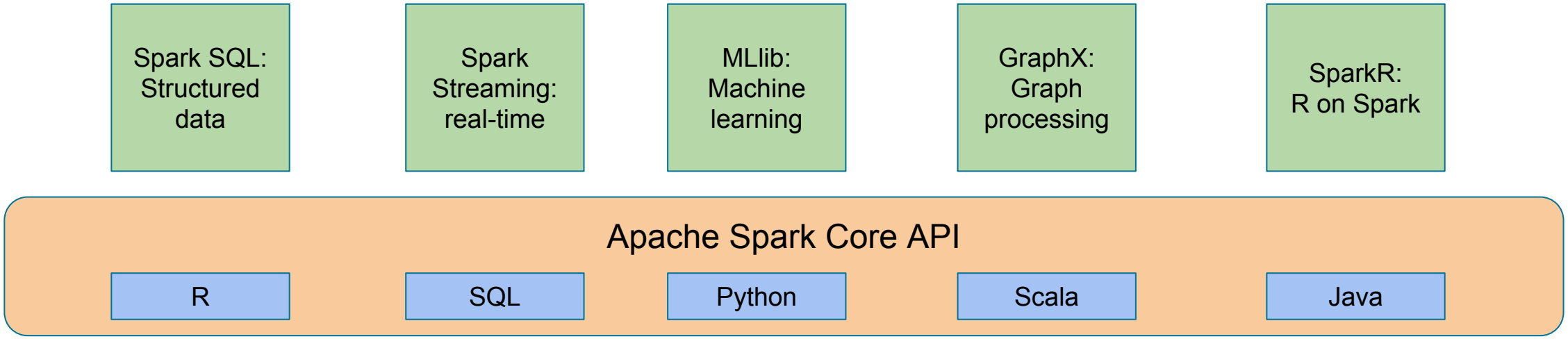
Introducción: ¿Por qué surge Spark?

- Nace como alternativa a la lentitud en algunos procesos Map-Reduce de Hadoop y como herramienta con una librerías más amigable y más transparente para los desarrolladores
- Los procesos iterativos e interactivos resultaban muy lentos dado que los cálculos intermedios se persistían en disco
- La desaceleración en la velocidad de los procesos hizo que muchos esfuerzos se centrasen en distribuir el procesamiento paralelamente entre varios nodos de computación
- Nace como necesidad de un framework con tolerancia a fallos automática

© Stratio 2018. Confidential, All Rights Reserved.

Introducción: Principales características de Spark

- Plataforma de computación distribuida de propósito general, que permite combinar, por ejemplo, sentencias SQL con algoritmos de Machine learning, gracias a su motor unificado de análisis

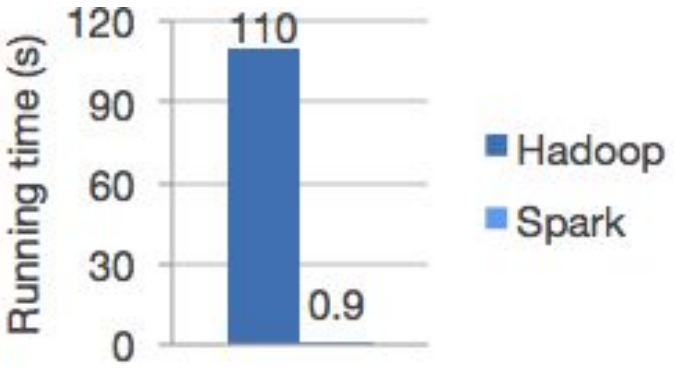


© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

Introducción: Principales características de Spark

- Acceso prácticamente a cualquier fuente de datos
- APIs en Java, Scala y Python
- Procesamiento 100x más rápido que en disco
- Tolerancia a fallos a través de la reconstrucción de RDDs
- Se puede apoyar en disco para persistencia temporal pero nunca para procesamiento

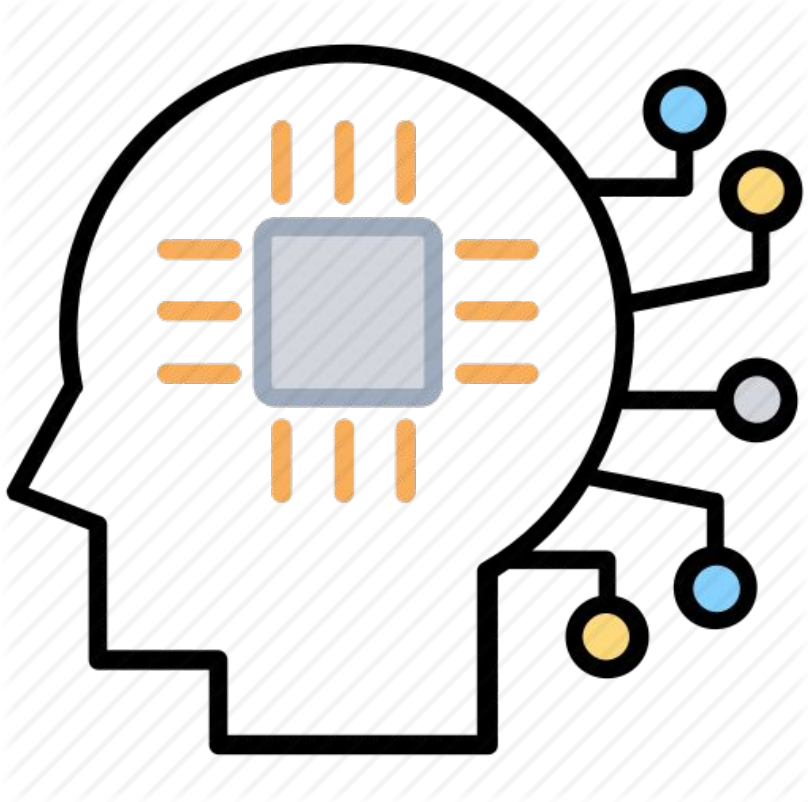


© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

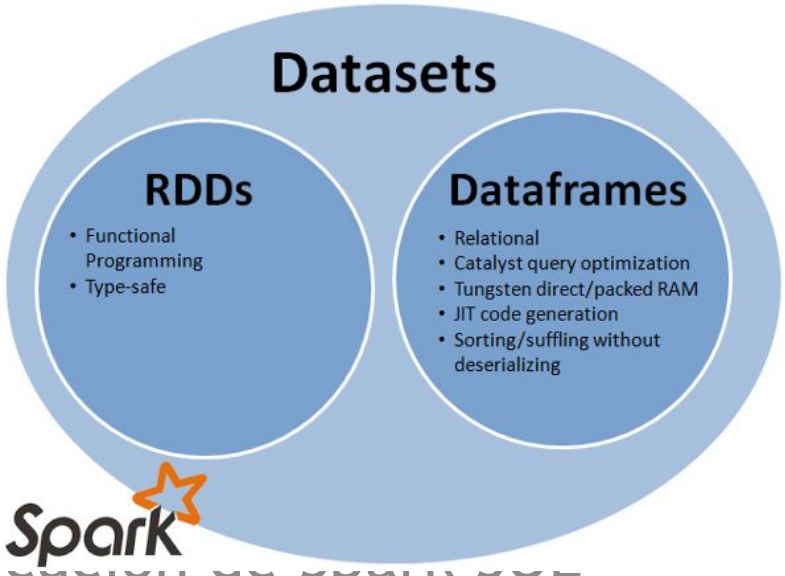
Introducción: Librerías de Spark

- Spark SQL
  - Datos estructurados
- Spark Streaming
  - Aplicaciones tolerante a fallos en streaming
- MLlib
  - Librería machine learning escalable
- GraphX
  - API para computación de grafos en paralelo
- SparkR
  - Paquete de R package con una interfaz ligera



© Stratio 2018. Confidential, All Rights Reserved.

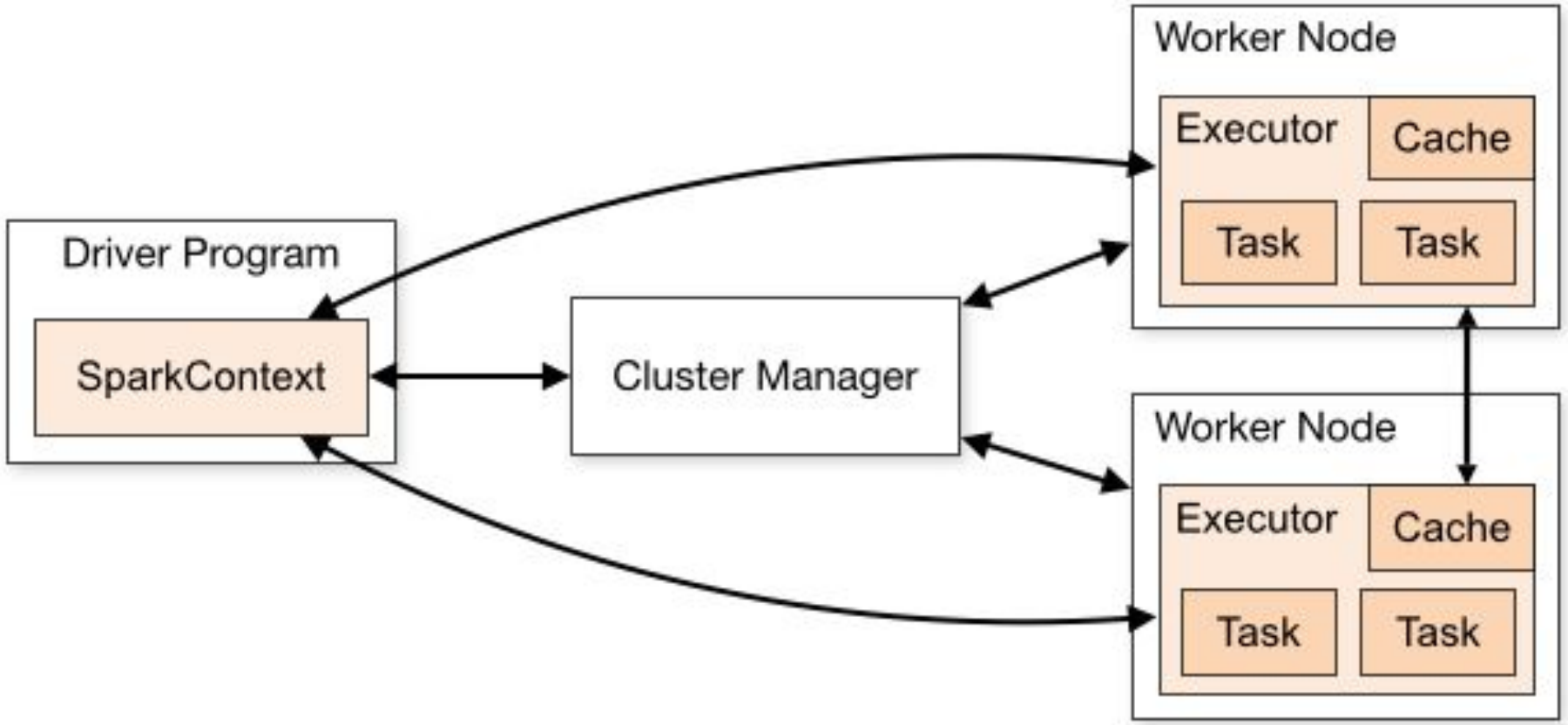
Introducción: RDD, DataFrame y Dataset



- RDD: Resilient Distributed Dataset
  - Abstracción de datos principal de Spark Core
  - Reparte datos entre las particiones
- Dataset
  - Estructura fuertemente tipada
  - Funciones lambda
  - Aprovecha las optimizaciones del motor de ejecución de Spark SQL
- DataFrame
  - Es un Dataset[Row] donde se proporciona un nombre a las columnas
  - Se puede construir a partir de un RDD, tablas de Hive, ficheros de datos estructurados o tablas de bases de datos relacionales

© Stratio 2018. Confidential, All Rights Reserved.

Introducción: Arquitectura



© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL

STRATIO

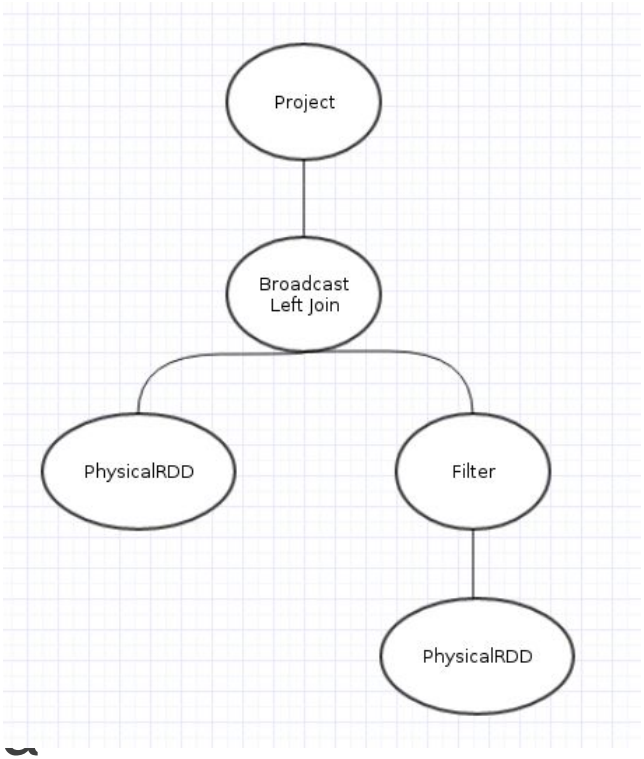
Spark SQL: ¿Qué es?

- Módulo de Spark que proporciona una abstracción de tabla proporcionando un esquema
- Acerca las capacidades de un framework de Big Data a herramientas de Business Intelligence
- Acceso uniforme a datos
- Conecta a cualquier fuente de datos de la misma manera
- Mezcla datos entre diferentes fuentes de datos
- Hereda la tolerancia a fallos y la computación distribuida de los RDDs y del Engine Core y las optimizaciones del Dataset

© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: ¿Por qué surge?

- El estándar SQL se impone ante el lenguaje NoSQL por su fuerte cohesión con las herramientas de BI
- La API de RDDs no tiene concepto de esquema
- Los RDDs no aprovechan información sobre el contexto de ejecución:
  - Optimiza la ejecución de las sentencias a través árboles de ejecución, cachés, estadísticas…
- Hive tenía carencias en cuanto a rendimiento y la tolerancia a fallos



© Stratio 2018. Confidential, All Rights Reserved.

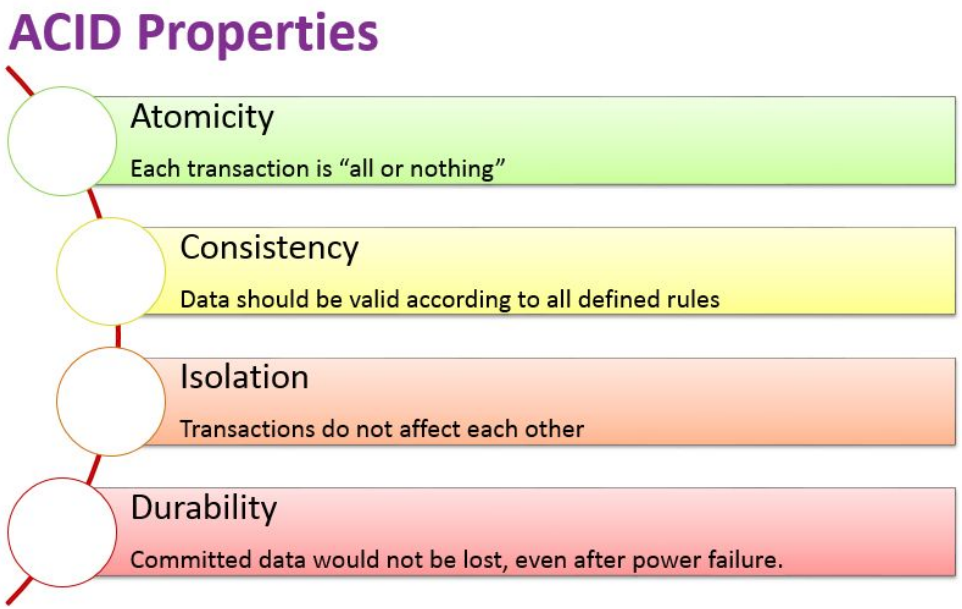
Spark SQL: Principales características

- Acceso unificado a una gran variedad de fuente de datos gracias a la API de Datasources
- Escalabilidad (RDDs)
- Tolerancia a fallos (RDDs)
- Motor de ejecución optimizado
- Integración con RDDs, DataFrames y Datasets
- Compatibilidad con sentencias Hive
- Compatibilidad con Hive metastore
- Extensibilidad

© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL vs Bases de Datos Relacionales

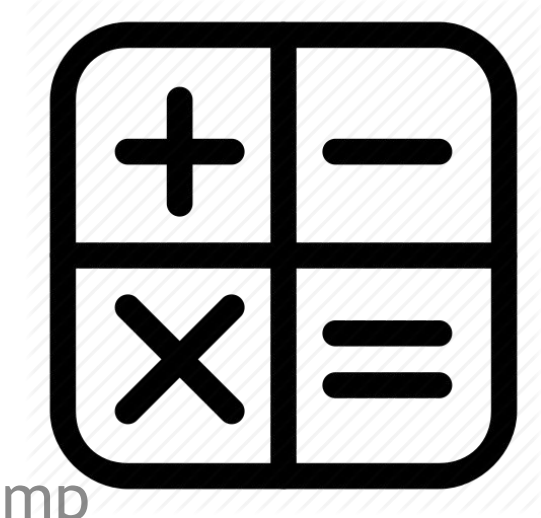
- Dado que accede a fuentes de datos muy heterogéneas:
  - No hay concepto de Primary Key
  - No hay concepto de Foreign Key
  - No hay concepto de índices
  - No hay concepto de triggers
  - No hay concepto de transacciones ACID
  - No hay concepto de roles
  - No hay concepto de esquema
  - No hay lenguaje procedural
- 2 tipos de tablas
  - Tabla externa
  - Table manejada



© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: Funciones

- Alrededor de 300 funciones y operadores aritméticos:
  - +, -, *, /, |, ~
  - max, min, avg, mean, count
  - cast, to_date, to_timestamp
  - ceil, floor
  - ifnull, isnull, exists
  - dayofyear, dayofmonth, dayofweek
  - filter, last, first
  - now, date, date_format, current:date, current_timestamp



```
scala> spark.sql("SELECT max(age) FROM example3").show
+--------+
|max(age)|
+--------+
|      99|
+--------+
```

© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

Spark SQL: UDFs

- User defined function
  - Aplica una función a una o más columnas a cada fila de una tabla

```
scala> spark.udf.register("square", (s: Long) => {s*s})
res84: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,LongType,Some(List(LongType)))

scala> spark.sql("SELECT square(id) FROM example3 LIMIT 1").show
+-----------------------------+
|UDF:square(cast(id as bigint))|
+-----------------------------+
|                           81|
+-----------------------------+
```

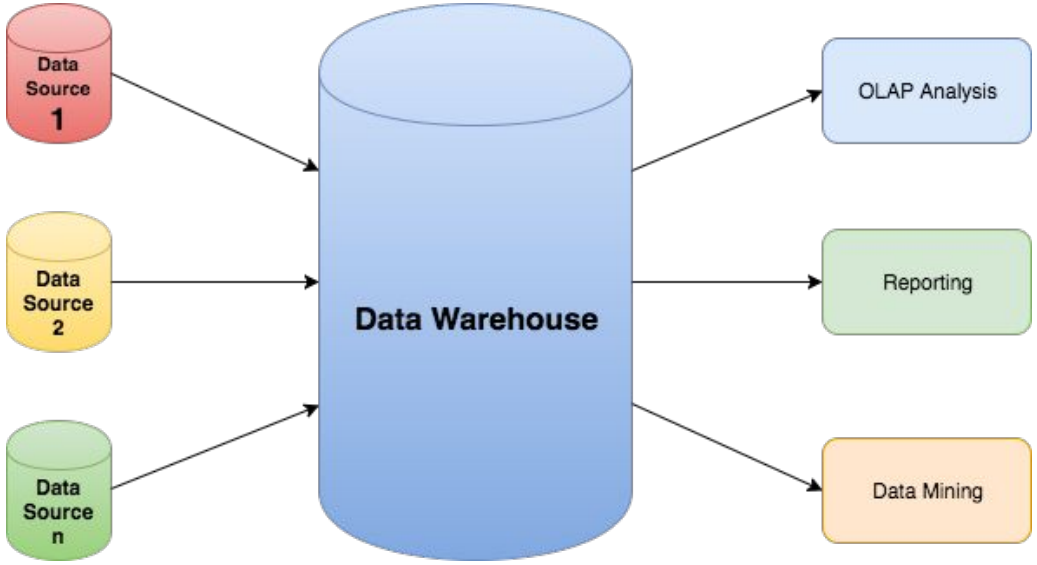
© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: UDFs y UDAFs

- User defined aggregate function
  - Permite realizar funciones de agregación teniendo en cuenta todos los valores de una columna
  - class GroupConcat extends UserDefinedAggregateFunction
    - initiliaze
    - update
    - merge
    - evaluate
  - spark.udf.register("group_concat", new GroupConcat)
  - spark.sql(“select group_concat(role) from example”)

© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: Que NO es

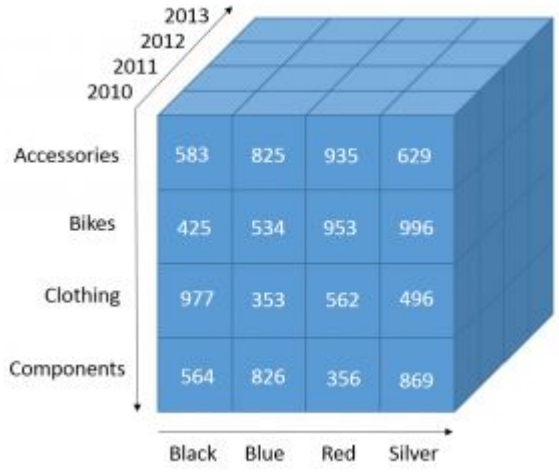


- NO es un data warehouse
- NO es una base de datos relacional
- NO es una herramienta transaccional
- NO es una herramienta comercial
- NO es una herramienta gráfica
- NO es una herramienta para sentencias online
- NO es una herramienta que solo utilice memoria
- NO es una cola para el envío de datos
- NO es solo una interfaz SQL

© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: Casos de uso

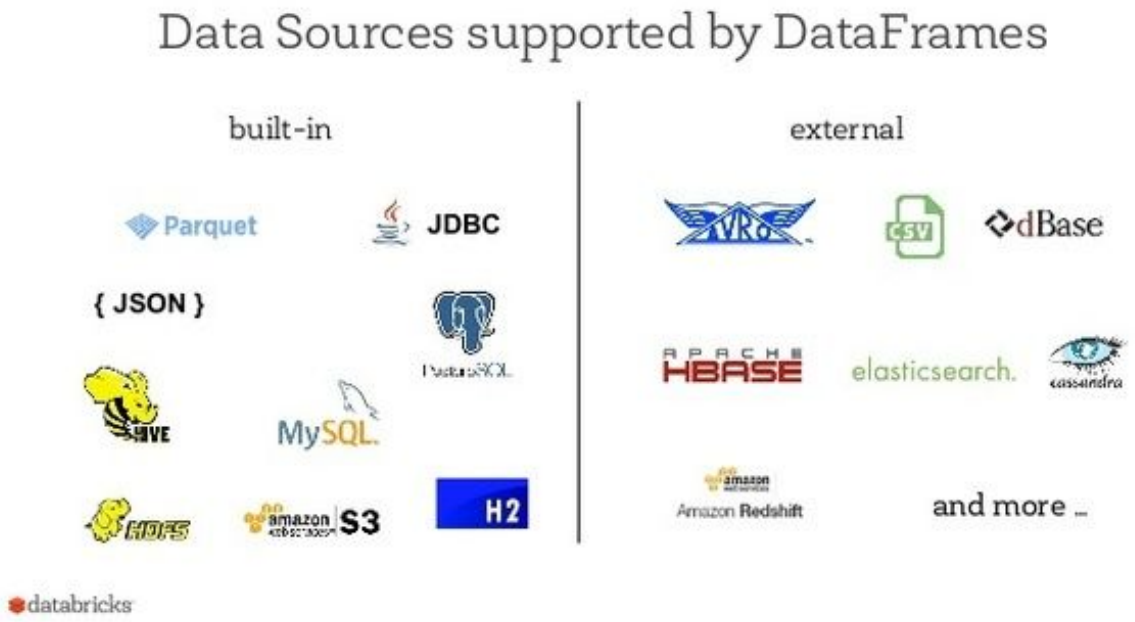
- Conectar herramientas de BI con un data lake
- Realizar analítica avanzada en Batch
- Realizar analítica avanzada en Streaming
  - Agregaciones en tiempo real (cubos OLAP)
- Machine learning a través de DataFrames
- Proporcionar esquema a datos desestructurados
- ETL
- Ingesta de datos
- Analítica avanzada en la nube



© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: Datasources

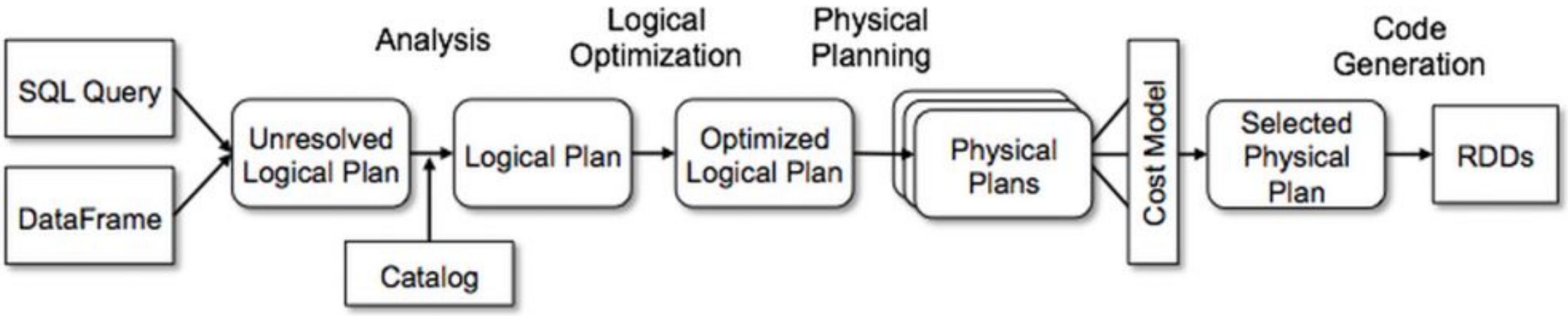
- Incluídos:
  - CSV
  - JSON
  - Parquet
  - JDBC
- Spark packages:
  - Cassandra
  - Mongo
  - Elasticsearch
  - HBase
- Comunidad:
  - Aerospike
  - Pig
  - Phoenix
  - Kudu
  - Arrow
  - Snowflake
  - Neo4j
  - Excel



© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: Componentes

- Catalyst



© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: Logical Plan y Spark Plan

```
scala> spark.sql("EXPLAIN EXTENDED SELECT * FROM example3").show(500, false)
```

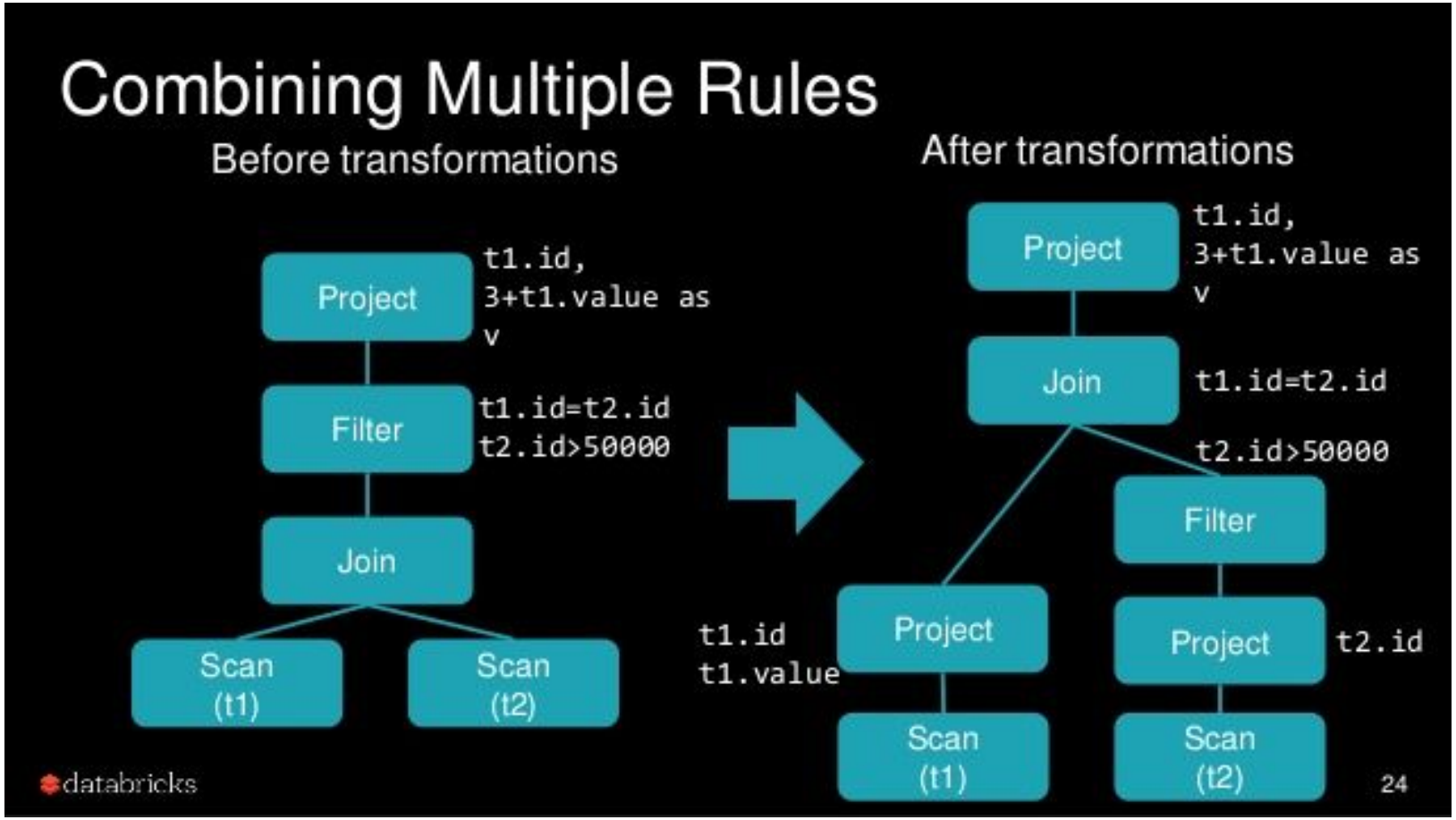
```
----------------------------------------+
|== Parsed Logical Plan ==
'Project [*]
+- 'UnresolvedRelation `example3`

== Analyzed Logical Plan ==
id: int, name: string, age: int
Project [id#1255, name#1256, age#1257]
+- SubqueryAlias example3
   +- Relation[id#1255,name#1256,age#1257] parquet

== Optimized Logical Plan ==
Relation[id#1255,name#1256,age#1257] parquet

== Physical Plan ==
*FileScan parquet default.example3[id#1255,name#1256,age#1257] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/tmp/example3], PartitionFilters: [], PushedFilters
: [], ReadSchema: struct<id:int,name:string,age:int>|
```

© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

Spark SQL: Optimizador de queries

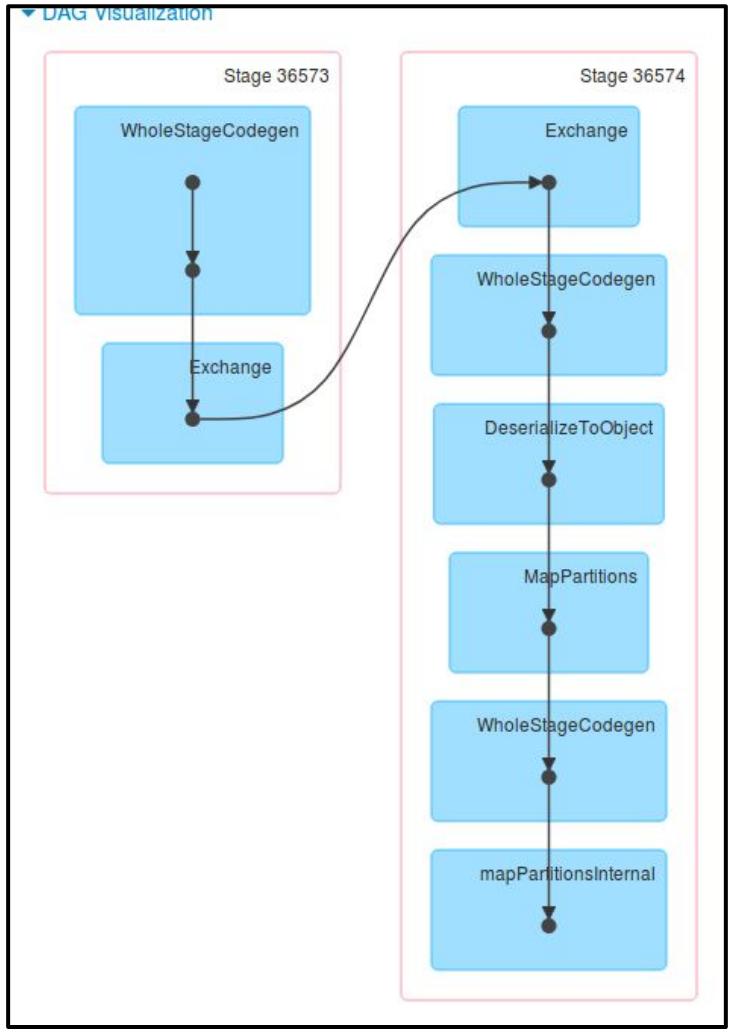
- Reglas
  - Genéricas
  - Específicas de cada datasource



© Stratio 2018. Confidential, All Rights Reserved.

Spark SQL: DAGs

- Job
- Stage
  - Fase de Shuffle
- Task



© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones

STRATIO

Optimizaciones: Configuración

- spark.sql.files.maxPartitionBytes
  - Default: 128MB
- spark.sql.files.openCostInBytes
  - Default: 4MB
- spark.sql.broadcastTimeout
  - Default: 300 seconds
- spark.sql.autoBroadcastJoinThreshold
  - Default: 10 MB
- spark.sql.shuffle.partitions
  - Default: 200

© Stratio 2018. Confidential, All Rights Reserved.

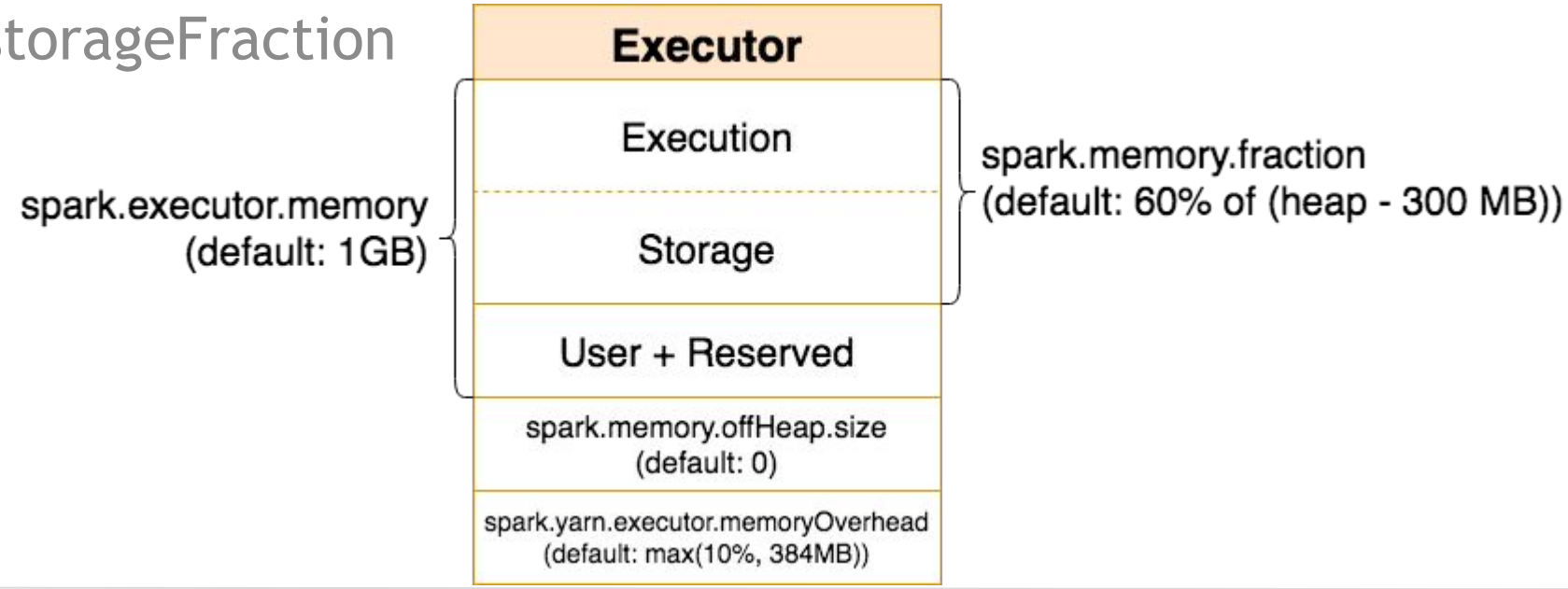
Optimizaciones: Hints

- El usuario puede introducir ciertas “pistas” para el optimizador de queries
  - SELECT /*+ COALESCE(2) */
  - SELECT /*+ REPARTITION(5) */
  - SELECT /*+ MAPJOIN(b) */
  - SELECT /*+ BROADCASTJOIN(b) */
  - SELECT /*+ BROADCAST(b) */

© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: Caché y memoria

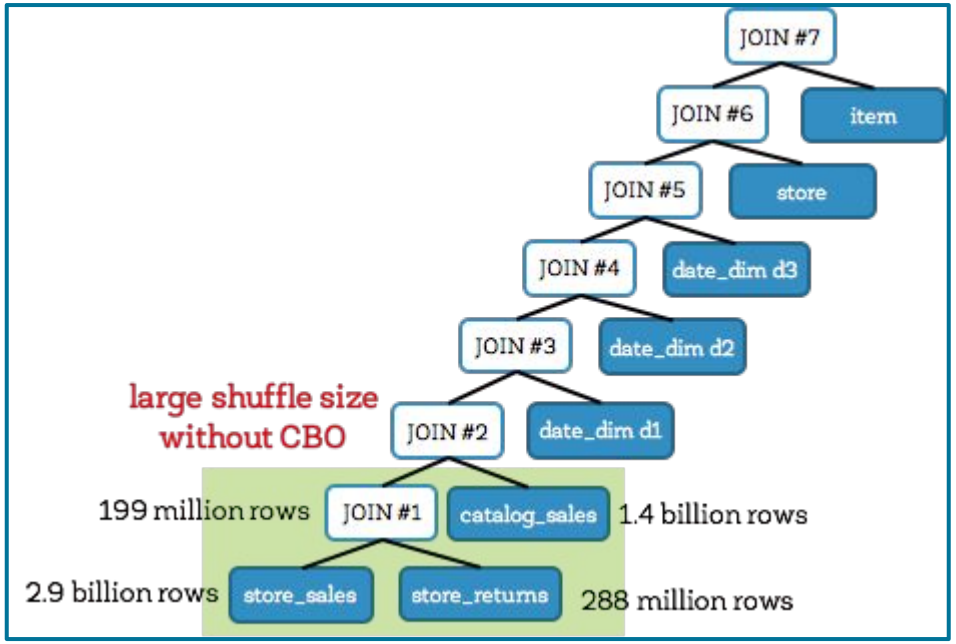
- Las tablas se pueden cachear en memoria (formato columnar)
- No se deben cachear tablas muy grandes
- Parámetros de configuración relacionados:
  - spark.memory.fraction
  - spark.memory.storageFraction

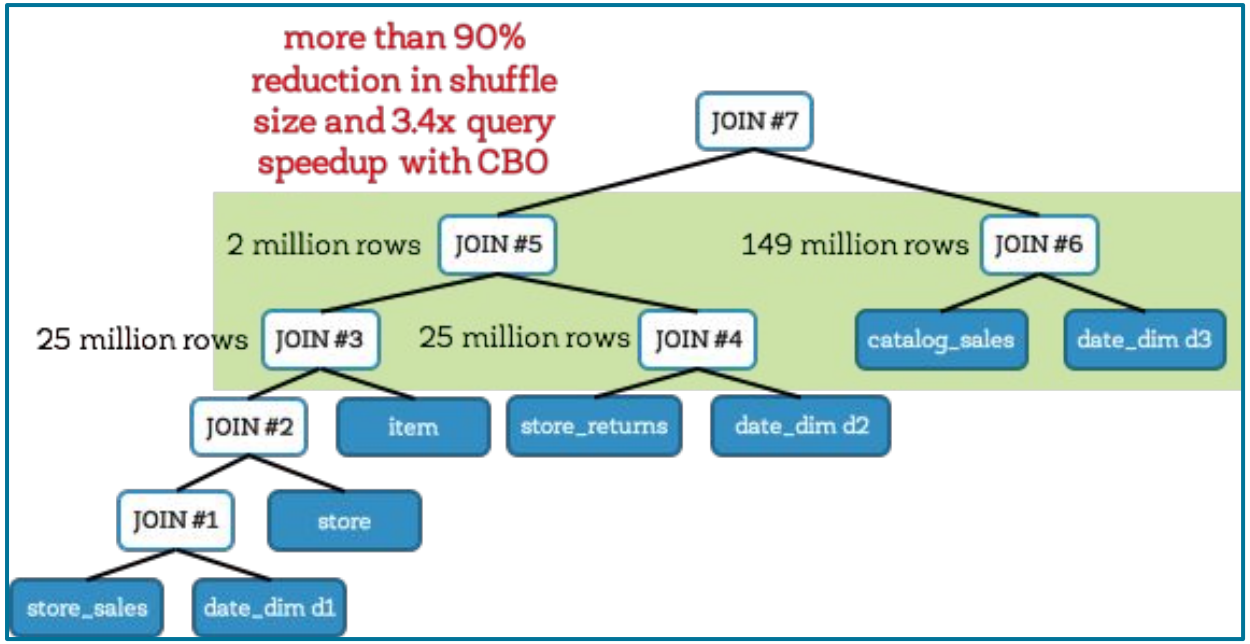


© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: CBO

- El Cost Based Optimizer está desactivado por defecto (spark.sql.cbo.enabled)
- Se basa en estadísticas sobre los datos para aplicar ciertas reglas de transformación a los árboles de ejecución





© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

Optimizaciones: CBO

- Se ha de popular las estadísticas con la instrucción ANALYZE TABLE

```
scala> spark.sql("ANALYZE TABLE example2 COMPUTE STATISTICS").show(500, false)
++
||
++
++


scala> spark.sql("DESCRIBE EXTENDED example2").show(50, false)
+----------------------------+-----------------------------------------------------------------------+-------+
|col_name                    |data_type                                                              |comment|
+----------------------------+-----------------------------------------------------------------------+-------+
|id                          |int                                                                    |null   |
|name                        |string                                                                 |null   |
|age                         |int                                                                    |null   |
|# Partition Information     |                                                                       |       |
|# col_name                  |data_type                                                              |comment|
|age                         |int                                                                    |null   |
|                            |                                                                       |       |
|# Detailed Table Information|                                                                       |       |
|Database                    |default                                                                |       |
|Table                       |example2                                                               |       |
|Owner                       |miguelangelfernandezdiaz                                               |       |
|Created                     |Thu Mar 07 11:19:57 CET 2019                                           |       |
|Last Access                 |Thu Jan 01 01:00:00 CET 1970                                           |       |
|Type                        |EXTERNAL                                                               |       |
|Provider                    |parquet                                                                |       |
|Table Properties            |[transient_lastDdlTime=1552561220]                                     |       |
|Statistics                  |1086 bytes, 2 rows                                                     |       |
|Location                    |file:/tmp/example2                                                     |       |
|Serde Library               |org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe            |       |
|InputFormat                 |org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat          |       |
|OutputFormat                |org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat         |       |
|Storage Properties          |[serialization.format=1]                                               |       |
|Partition Provider          |Catalog                                                                |       |
+----------------------------+-----------------------------------------------------------------------+-------+
```

© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

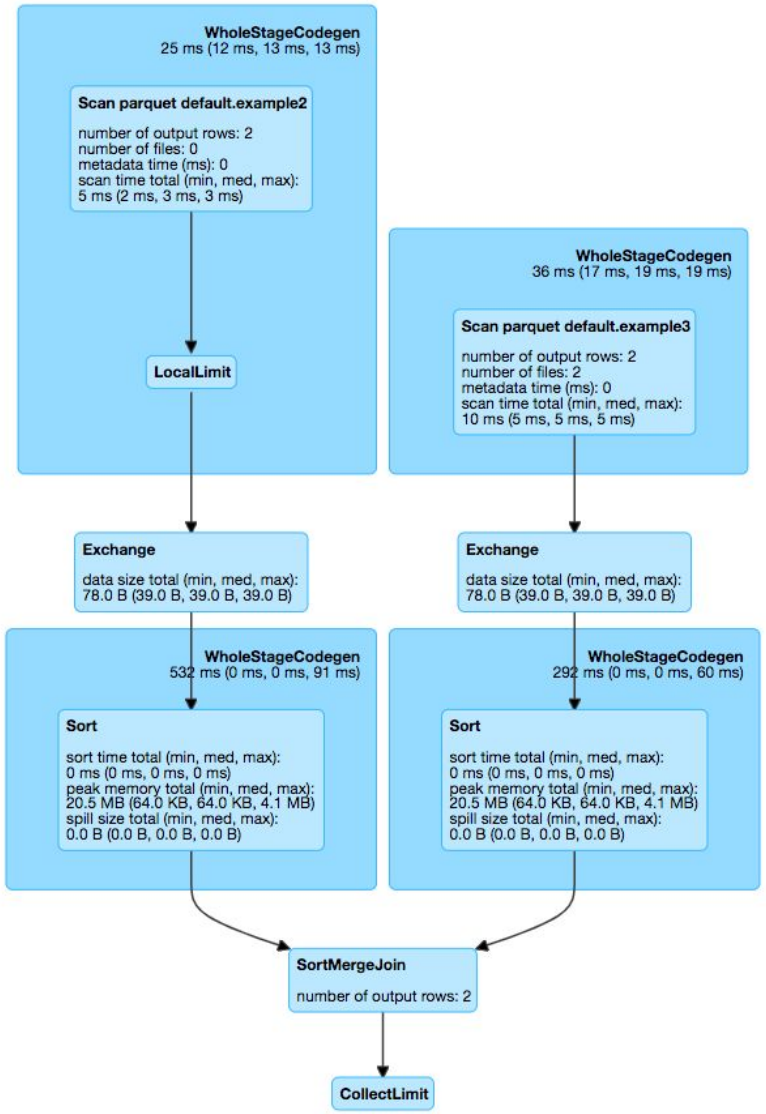
Optimizaciones: Sentencias pesadas

```
scala> spark.sql("SELECT * FROM example2 FULL OUTER JOIN example3 ON example2.name=example3.name").show(50, false)
+---+-----+---+---+-----+---+
|id |name |age|id |name |age|
+---+-----+---+---+-----+---+
|2  |Two  |22 |6  |Two  |66 |
|3  |Three|33 |9  |Three|99 |
+---+-----+---+---+-----+---+


scala> spark.sql("SELECT * FROM example2 LEFT JOIN example3 USING(name) UNION ALL SELECT * FROM example3 LEFT JOIN example2 USING(name) WHERE example2.name IS NULL").show(50, fals
e)
+-----+---+---+---+---+
|name |id |age|id |age|
+-----+---+---+---+---+
|Three|3  |33 |9  |99 |
|Two  |2  |22 |6  |66 |
+-----+---+---+---+---+
```

© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: Sentencias pesadas

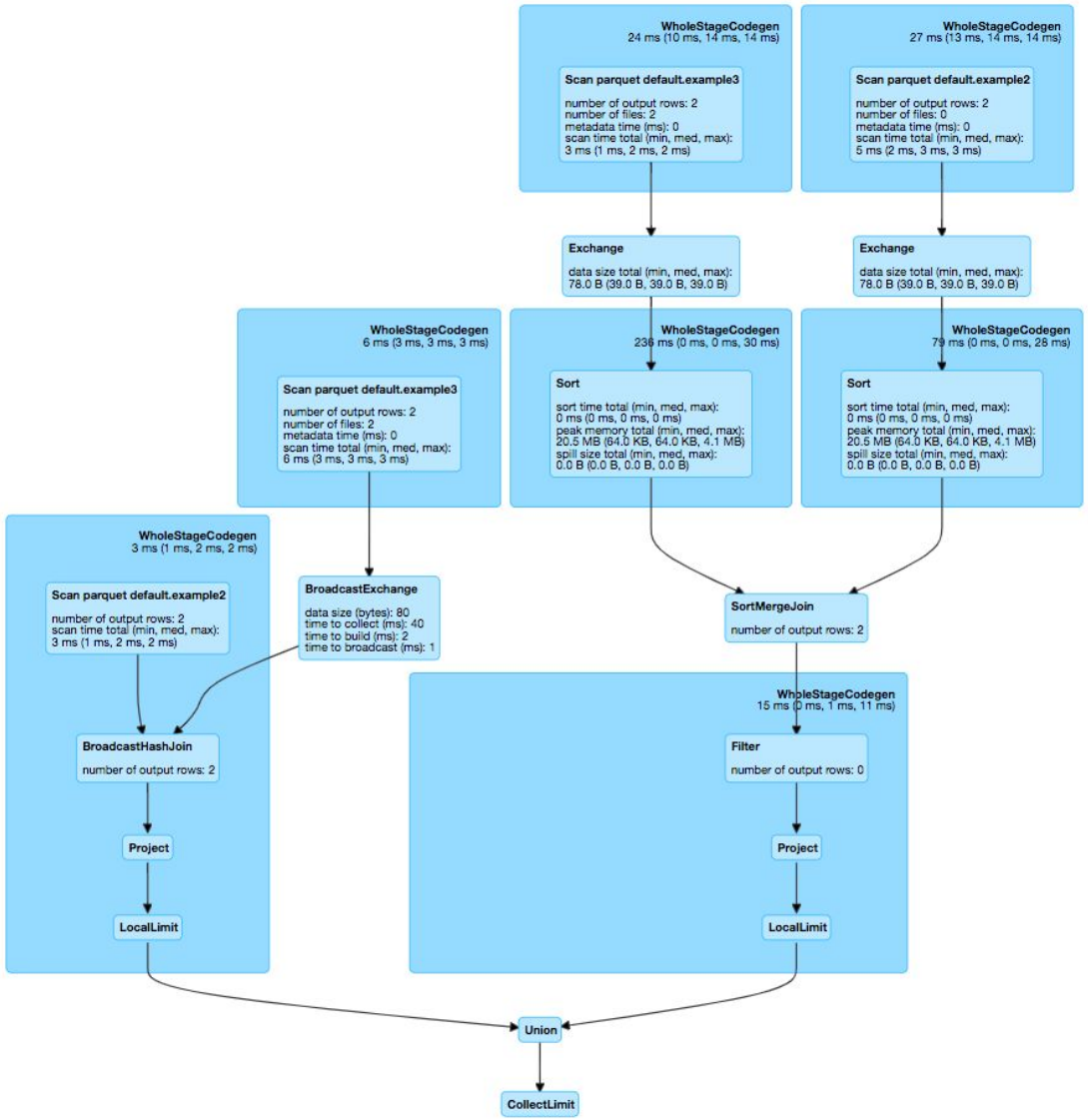
- Left outer join DAG



© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: Sentencias pesadas

- Union all DAG



© Stratio 2018. Confidential, All Rights Reserved.

STRATIO

Optimizaciones: Particiones

- El particionado de los datos puede ayudar a reducir el tiempo de consulta en sistemas de ficheros
- Los filtros y cláusulas WHERE que deben utilizar esta estructura jerárquica para beneficiarse de esta optimización
  - hdfs://namenode/departamentos/marketing/year=2012/pais=Spain
  - SELECT * FROM marketing WHERE year=2012 AND pais=Spain
- En HDFS, se debe evitar la fragmentación de ficheros dentro del mismo directorio ya que puede saturar tanto al namenode e introducir una sobrecarga importante en la planificación de las sentencias

© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: Big Data ≠ Grandes resultados

- Spark SQL está pensado para hacer analítica pesada sobre gran cantidad de datos pero… NO está diseñado para devolver resultados grandes
  - spark.driver.maxResultSize (default: 1GB)
- Alternativas:
  - Volcar el resultado a una capa de persistencia capaz de paginar
  - Usar la API de DataFrame:
    - toLocalIterator

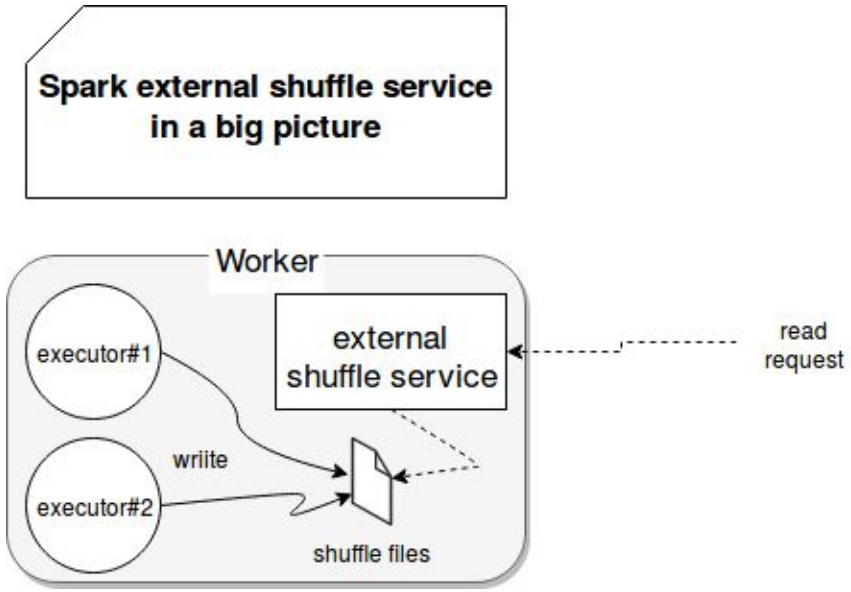
© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: TPC-DS

- Es un benchmark orientado a analítica Big Data
- Posibles problemas y sus soluciones:
  - No space left on device → spark.local.dir
  - Total size of serialized results of 381610 tasks (5.0 GB) is bigger than spark.driver.maxResultSize (5.0 GB) → limit, filter, saveAsParquetFile
  - Error executing query, currentState RUNNING,
  - org.apache.spark.sql.catalyst.errors.package$TreeNodeException: execute, tree → spark.sql.broadcastTimeout
  - Futures timed out after [300 seconds] → spark.network.timeout

© Stratio 2018. Confidential, All Rights Reserved.

Optimizaciones: Dynamic Allocation



- Es una forma de escalar elásticamente los ejecutores, añadiéndolos o quitándolos en función de la carga
- El servicio de shuffle externo debe estar activado
- Parámetros de configuración:
  - spark.dynamicAllocation.enabled (default: false)
  - spark.dynamicAllocation.executorIdleTimeout (default: 60s)
  - spark.dynamicAllocation.minExecutors (default: 0)
  - spark.dynamicAllocation.maxExecutors (default: infinity)

© Stratio 2018. Confidential, All Rights Reserved.

WE ARE HIRING

people@stratio.com

@StratioBD

Preguntas

© Stratio 2018. Confidential, All Rights Reserved.

STRATIO