DATALEARNS247

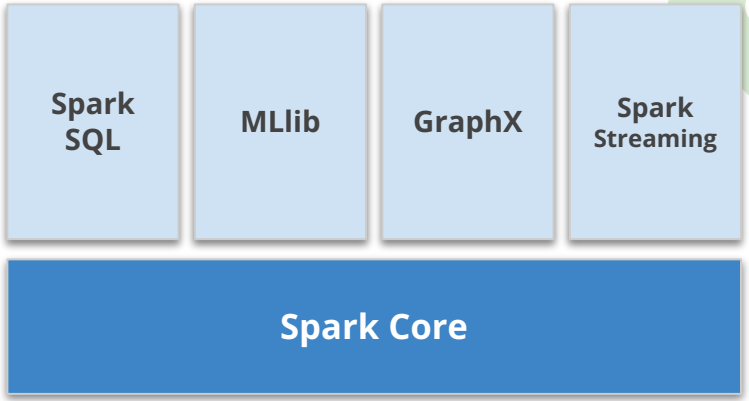
Apache Spark Data Processing

Big Data Engineering

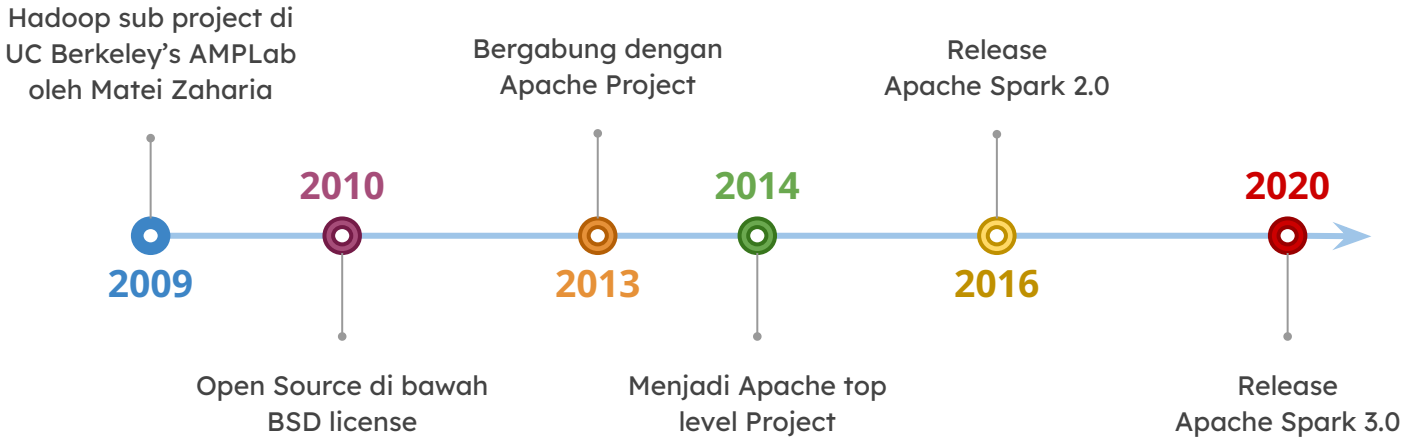
01 Overview

Apa itu Apache Spark ?

- Apache Spark™ adalah mesin multibahasa untuk data engineering, data science, dan machine learning, menggunakan node tunggal ataupun klaster.
- Mesin komputasi Spark disebut Spark Core, di atasnya dibangun berbagai library untuk SQL, Machine Learning, Streaming dan komputasi Graf.
- Fitur utama Spark adalah in-memory cluster computing.



Sejarah Apache Spark



Mengapa Apache Spark?

Fast Processing

Mencapai 100x lebih cepat dari Hadoop dengan in-memory computing, dan 10x lebih cepat dengan disk.

Fault Tolerant

Menggunakan **RDD (Resilient Distributed Dataset)** yang didesain untuk menangani kegagalan node dalam cluster.

Flexible

Mendukung berbagai bahasa pemrograman populer : Java, SQL, Python, dan R.

Unified Engine

Mendukung pemrosesan data batch, graph, streaming, dan query interaktif dalam satu mesin.

Extensible

Mendukung berbagai jenis data store, termasuk HDFS, Cassandra, HBase, MongoDB, Hive, RDBMS, dll.

Wide Support

Lebih dari 2000 kontributor. Digunakan oleh banyak perusahaan, termasuk 80% perusahaan Fortune 500.

02 Bagaimana Spark Bekerja ?

Spark Execution Mode

Spark menggunakan arsitektur master/slave. Setiap Spark program memiliki satu koordinator pusat disebut **driver** yang berkomunikasi dengan banyak worker yang terdistribusi (**executor**).

Spark memiliki 3 mode eksekusi

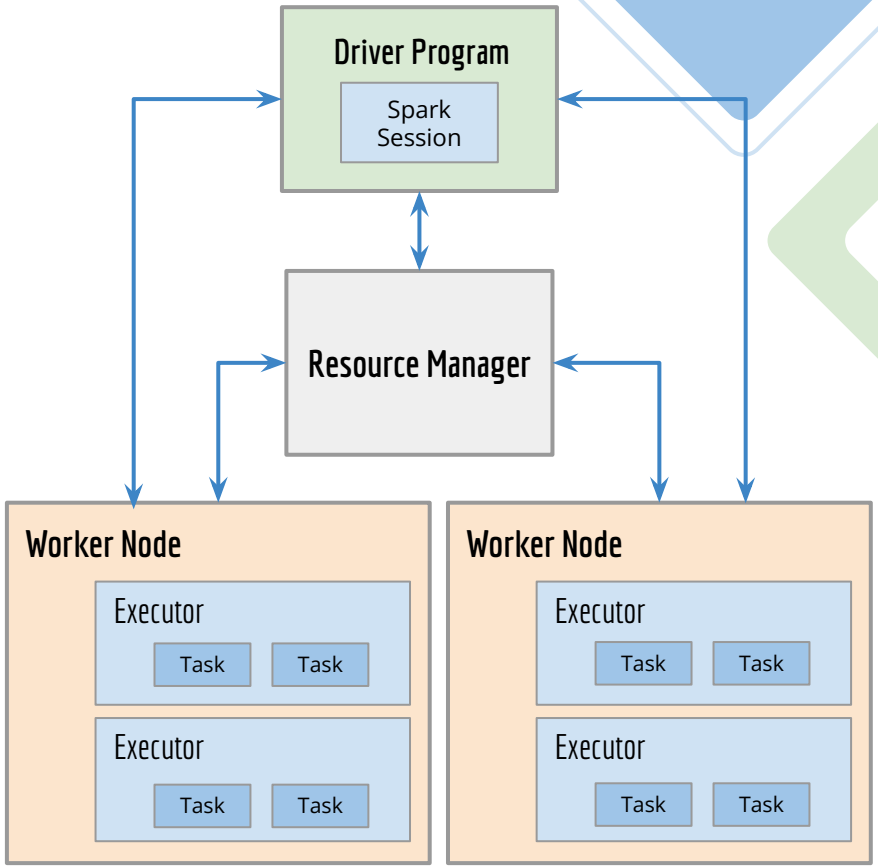
- Local
  Eksekusi program secara lokal, di notebook atau PC. Tidak menggunakan cluster yang terdistribusi

- Client mode
  Spark driver berada pada client di luar cluster, executor berada pada worker node

- Cluster mode
  Spark driver dan work berada pada node yang ada dalam cluster

Arsitektur Spark

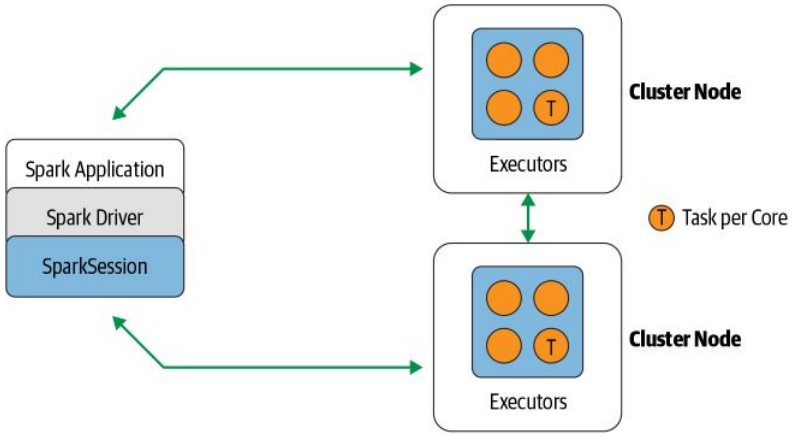
- Setiap aplikasi Spark memiliki satu koordinator pusat disebut **Driver**
- Driver membuat object **Spark Session** untuk berkomunikasi dengan **Resource Manager**
- Resource Manager mengalokasikan **Executor**, yaitu proses yang menjalankan komputasi dan menyimpan data untuk aplikasi
- Spark Session mengirimkan task untuk dieksekusi oleh executor
- Spark dapat digunakan dengan berbagai Resource Manager, diantaranya **Spark**, **YARN, Mesos,** dan **Kubernetes**,



SparkSession Code

Contoh kode untuk membuat SparkSession

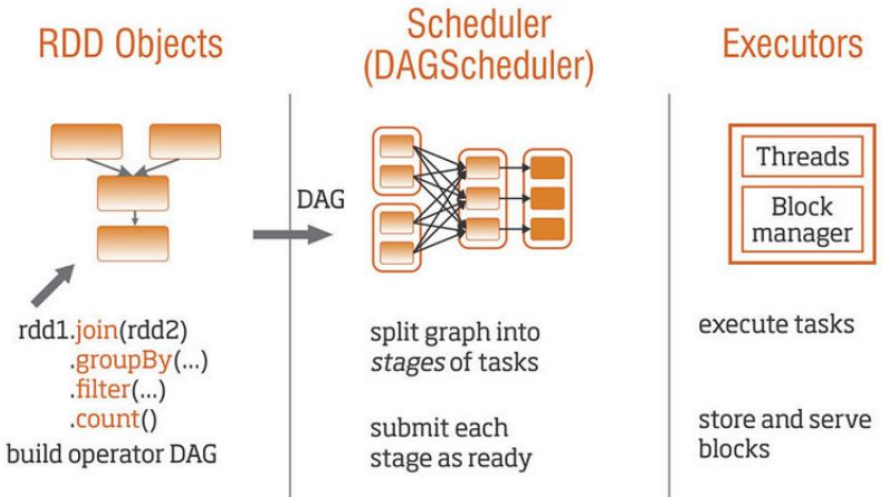
```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder
                    .appName('My Application')
                    .getOrCreate()

#read file as Spark dataframe
df = spark.read.csv(...)
```



Spark Abstractions

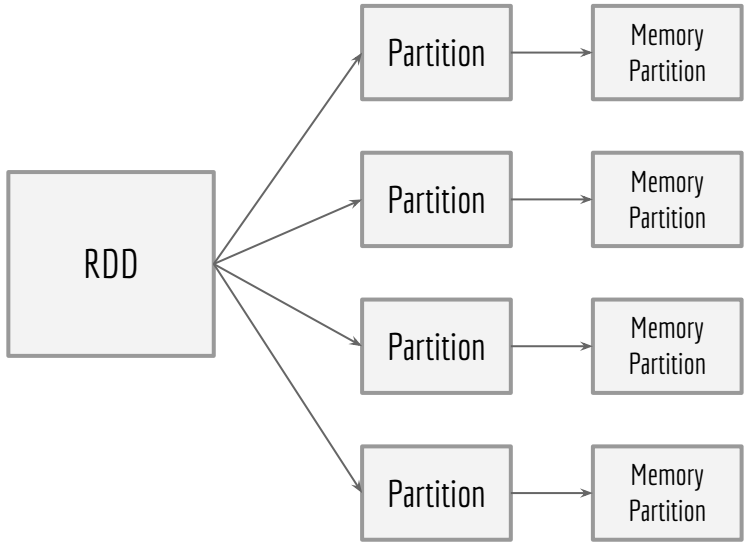
Proses dalam Spark didasarkan pada 2 konsep utama :

- Data Abstraction : **RDD** (Resilient Distributed Dataset)
- Process Abstraction : **DAG** (Direct Acyclic Graph) Execution Plan



Spark RDD

- Spark distributed data abstraction
- Immutable & resilient
- Lazy evaluation

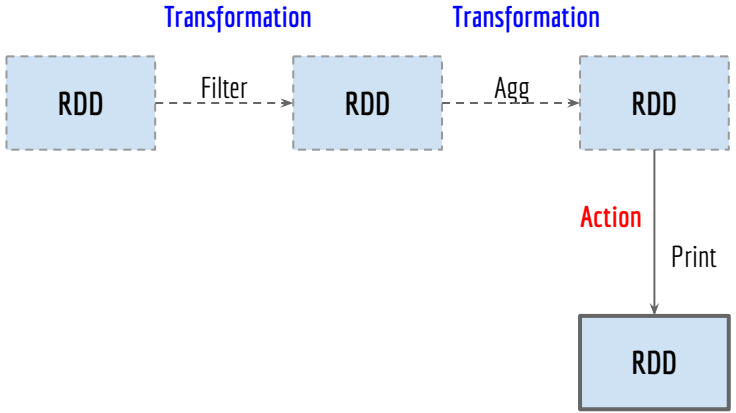


Immutability

- Immutable berarti sekali dibentuk, RDD tidak bisa diubah
- Ketika RDD diubah, Spark membuat RDD baru dan menyimpan RDD asli beserta proses transformasinya
- Bisa direkonstruksi kapanpun diperlukan, termasuk jika terjadi kegagalan proses → **Resilient**
- Apakah hal ini tidak menyebabkan pemborosan resource?

Lazy Evaluation

- Spark menunda eksekusi proses sampai benar-benar dibutuhkan
- Setiap transformasi terhadap RDD tidak langsung dieksekusi sampai data betul-betul dibutuhkan
- Perintah yang membutuhkan akses langsung ke data disebut **Action**
- Dengan metode ini Spark dapat melakukan optimasi terhadap rangkaian proses yang akan dieksekusi



Spark Transformations dan Actions

Spark Operations =

Fungsi yang menerima RDD sebagai input dan menghasilkan satu atau beberapa RDD baru dengan menerapkan operasi yang diwakilinya

TRANSFORMATIONS

filter(), map(), join(), distinct(), groupByKey(), etc.

+

Fungsi yang mengakses data di RDD.
Action mentrigger eksekusi semua transformasi terkait untuk mendapatkan data yang diperlukan.

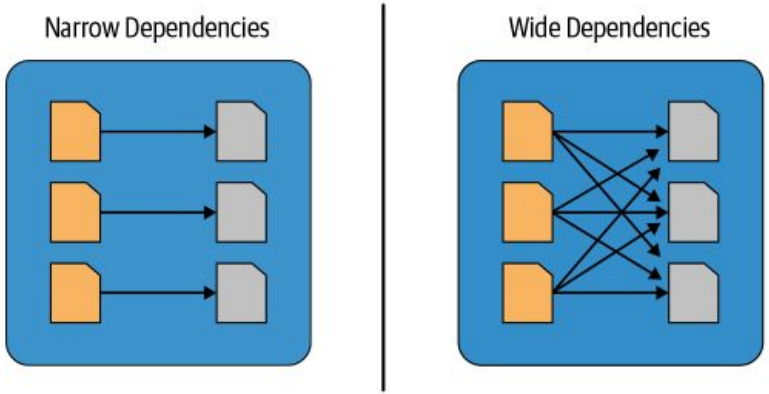
ACTIONS

show(), collect(), count(), first(), etc.

https://training.databricks.com/visualapi.pdf

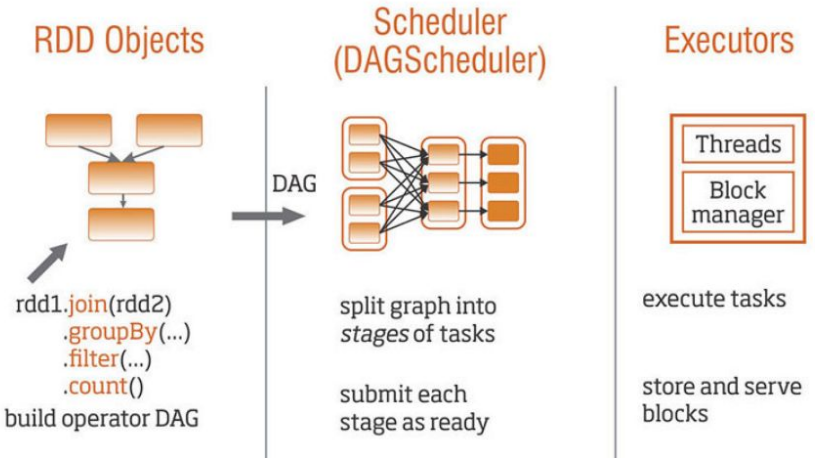
Narrow and Wide Transformations

- **Narrow Transformation** transformasi di mana sebuah partisi output hanya bergantung pada sebuah partisi input. Misalnya, `filter(), contains()`
- **Wide Transformation** transformasi yang melibatkan satu atau lebih partisi untuk melakukan proses. Misalnya `groupBy(), join()`
- Pertukaran data antar partisi disebut dengan **shuffle**



Spark DAG dan Execution Plan

- Spark membentuk rencana eksekusi (execution plan) yang berbentuk DAG (directed acyclic graph)
- DAG Scheduler dan Optimizer akan membentuk execution plan yang optimal, yang selanjutnya diterjemahkan menjadi serangkaian stage dan task
- RDD immutability + rangkaian instruksi dalam DAG menjadi dasar fault tolerance dalam Spark



03 Spark Structured API

Spark APIs

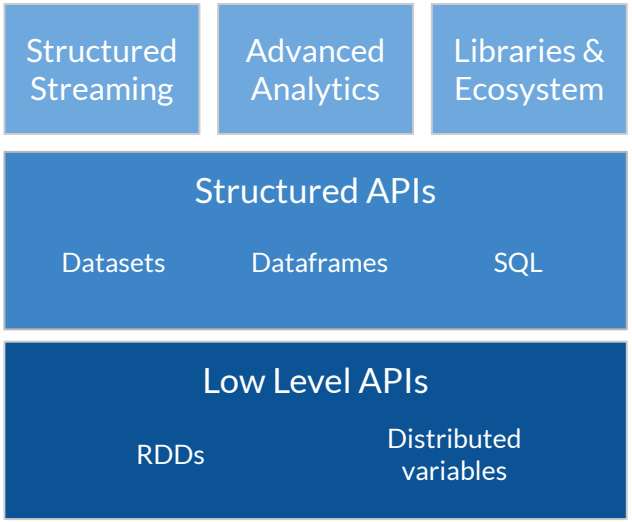
Spark memiliki dua set API dasar:

- API level rendah yang “tidak terstruktur”, yaitu :

  RDD dan **Distributed variables**

- API level lebih tinggi yang lebih terstruktur, yaitu :

  Dataset, **DataFrame** dan **SQL APIs**



Spark Structured APIs

- DataFrames

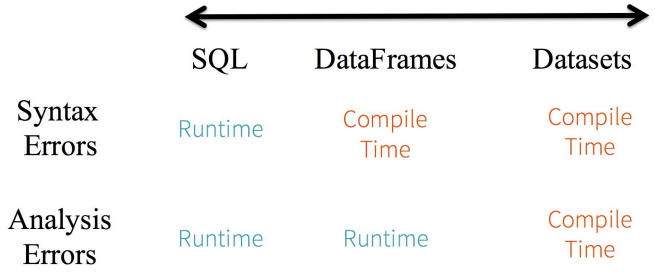
  Secara sederhana, DataFrame menggambarkan tabel dengan baris dan kolom. Konsepnya mirip dengan DataFrame R dan Python (Panda). Konversi Spark DataFrame ke R atau Python dapat dilakukan dengan mudah.

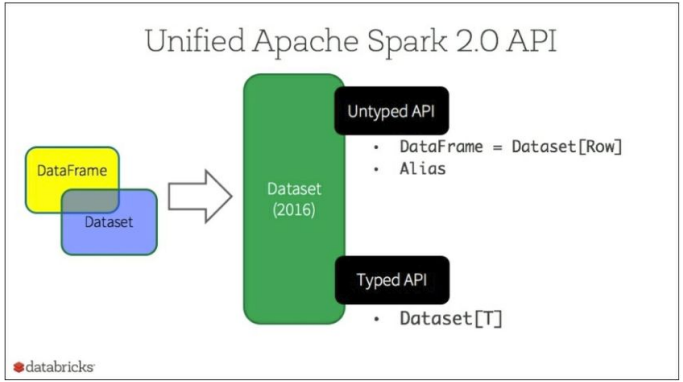
- Datasets

  Sama dengan DataFrame, DataSet memiliki struktur kolom dan baris. Bedanya, elemen Dataset merupakan objek yang *strongly-typed*.

- SQL tables and views

Note : Sejak Spark 2.0 Dataset dan DataFrame disatukan dalam satu API. DataFrame dianggap sebagai Dataset yang bertipe Row (untyped). Untuk implementasi di pyspark hanya ada DataFrame saja.

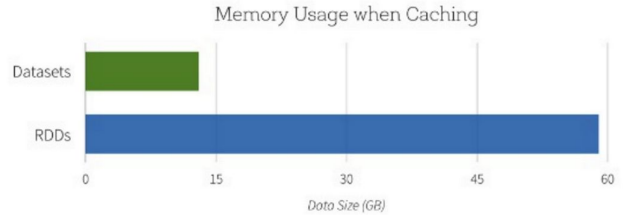


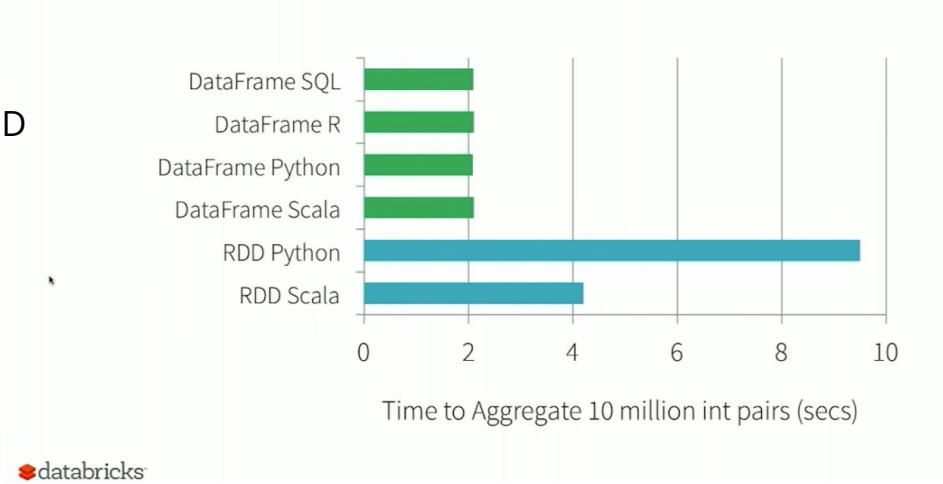


Mengapa Structured API?

- Mudah digunakan → tell Spark **what to do** not **how to do it**
- API yang konsisten untuk semua high level API : Machine learning, Deep learning, Graph, Streaming, dll
- Optimized storage and operations

  Berbeda dengan operasi dan data pada RDD yang bersifat transparan, Structured API memungkinkan Spark mengenali jenis operasi dan data yang diproses, sehingga dapat melakukan optimasi lebih lanjut, misalnya predicate pushdown, dll.





Kapan Menggunakan RDD?

Meskipun kita disarankan untuk menggunakan high level structured API, akan tetapi ada kasus di mana kita perlu menggunakan RDD atau low level API

- Jika perlu menggunakan fungsi yang tidak ada di *higher level API*
- Jika masih menggunakan legacy code yg menggunakan RDD
- Jika ingin melakukan custom partitioning (meskipun kita bisa melakukan ini dengan DataFrame sampai batas tertentu)
- Jika ingin melakukan manipulasi shared custom variable yang tidak ada di structured API

Karena pada dasarnya DataFrame akan dieksekusi oleh Spark sebagai RDD, memahami RDD akan sangat membantu dalam melakukan design, monitoring, maupun troubleshooting.

Hands-On

RDD vs DataFrame

Deskripsi

Dalam praktek ini, kita akan mempelajari perbedaan pemrograman menggunakan RDD dan DataFrame

Hands-On

Eksplorasi DataFrame

Deskripsi

Pada praktek ini, kita akan mempelajari :

- Beberapa cara untuk membentuk DataFrame dari beberapa sumber data:
  - Membentuk DataFrame dari list
  - Membentuk DataFrame dari Pandas DataFrame
  - Membaca file csv dan simple JSON file
- Operasi umum pada DataFrame:
  - Filtering
  - Sorting
  - Aggregation
  - Join

Hands-On

SQL Query dengan DataFrame

Deskripsi

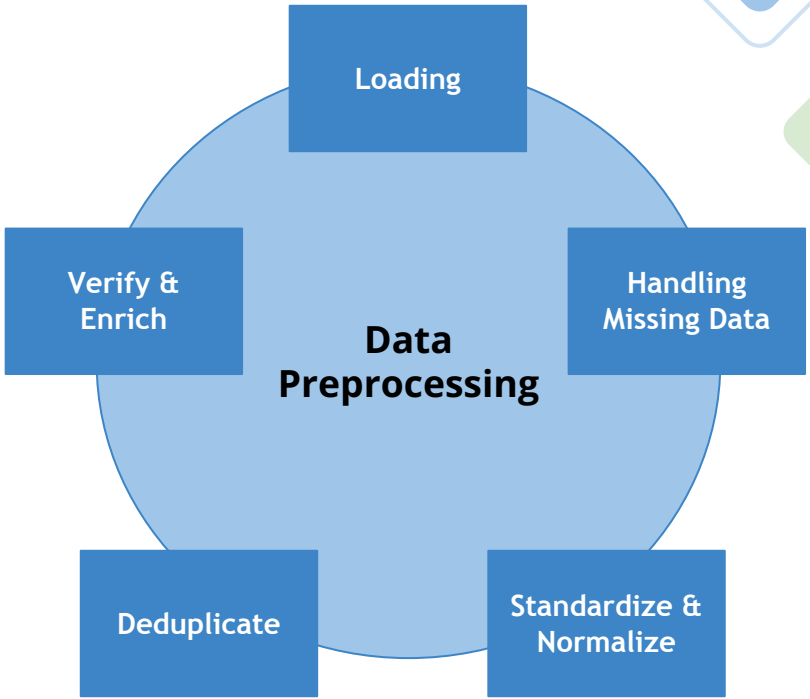
Pada praktek ini, kita akan mempelajari mengenai bagaimana mengolah data menggunakan SQL Query pada DataFrame

04 Data Cleaning and Enrichment

Data Preprocessing

Siapa yang perlu mempelajari data preprosesing

- Data Engineers
- Data Scientists/Data Analyst
- Machine Learning Engineers



Apa Yang Harus Diperhatikan?

- Invalid/Reject Records
- Null Values
- Invalid values
- Nonstandard values
- Duplicate records
- etc.

Null Values

- Masalah pertama yang ditemui biasanya adalah NULL values
- Perlu memahami data untuk menentukan penyebab/arti dari nilai NULLs : missing? unknown? broken records? dll
- Apa yang bisa dilakukan dengan nilai NULLs?
  - Menghapusnya
  - Menggantikan dengan nilai mean/median/mode
  - Memberikan unique category
  - Memprediksi nilai yang hilang tersebut
  - dll

Data Standardization

Beberapa hal yang harus diperhatikan untuk melakukan standarisasi data

- Extra Spaces dan Blank Cells
- Data type format : number, string, date
- Data format : date, time, etc
- Text formatting : upper/lower/proper case
- Spell check
- dll

Data Enrichment : Menggabungkan DataFrames

- Ada 2 cara untuk menggabungkan data : merge and join
- Join biasanya digunakan pada proses enrichment
- Untuk melakukan merge DataFrame, gunakan fungsi transformasi **`union()`**
- **`union()`** tidak memeriksa duplikasi data
- Operasi Join pada PySpark memiliki potensi masalah kinerja bila tidak dirancang dengan hati-hati karena dapat melibatkan shuffling data secara masif

Hands-On

Data Cleansing and Enrichment

Deskripsi

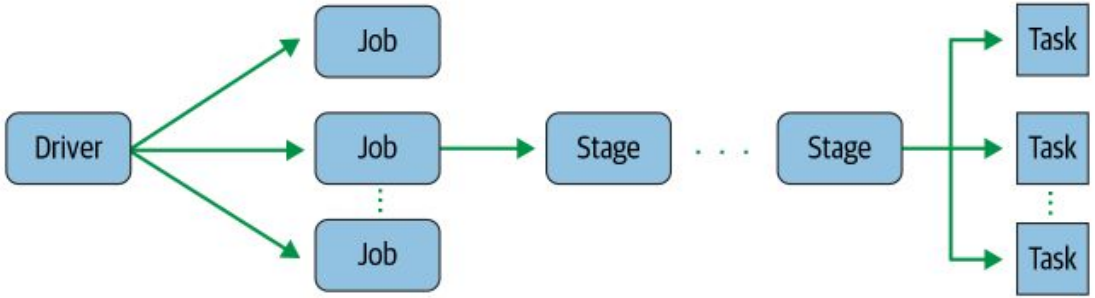
Pada praktek ini, kita akan mempelajari:

- Penggunaan spark dataframe untuk membaca file csv
- Membersihkan data, yaitu
  - Menangani missing value
  - Menangani data invalid
  - Menangani data duplikat
- Enrichment dengan data referensi

05 Monitoring

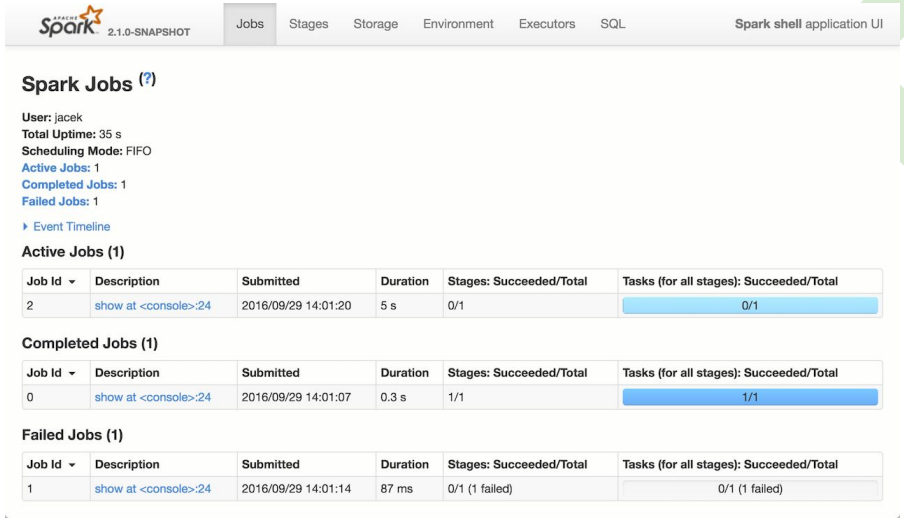
Spark Jobs, Stages and Tasks

- Driver mengubah aplikasi Spark User menjadi satu atau beberapa **Spark Job**
- **Job** : komputasi paralel yang terdiri dari banyak tugas (**task**)
- **Stages** : Setiap **Job** dibagi beberapa kumpulan **task** yang disebut **Stage**. Sebuah stage dibuat setiap kali terjadi shuffle
- **Task** : unit kerja terkecil yang akan dikirimkan ke **executor**. Setiap stage memiliki beberapa task, satu task per partisi



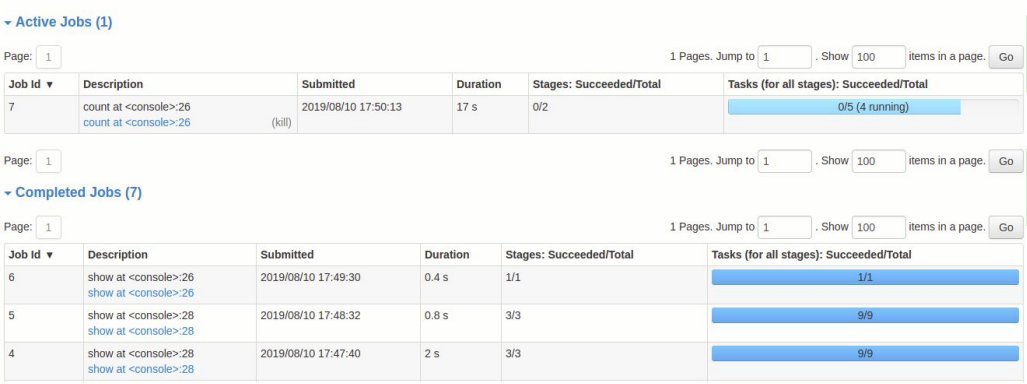
Spark Web UI

- **Jobs** : status tiap job.
- **Stages** : status tiap stage.
- **Storage** : informasi tentang *persisted* RDD dan DataFrame, jika ada.
- **Environment** : nilai untuk berbagai variabel environment dan konfigurasi, termasuk JVM, Spark, dan properti sistem.
- **Executors** : informasi tentang executor, termasuk penggunaan memori dan disk, serta informasi *task* dan *shuffle*.
- **SQL** : informasi tentang query SQL, seperti durasi, job, serta logical dan physical plan.



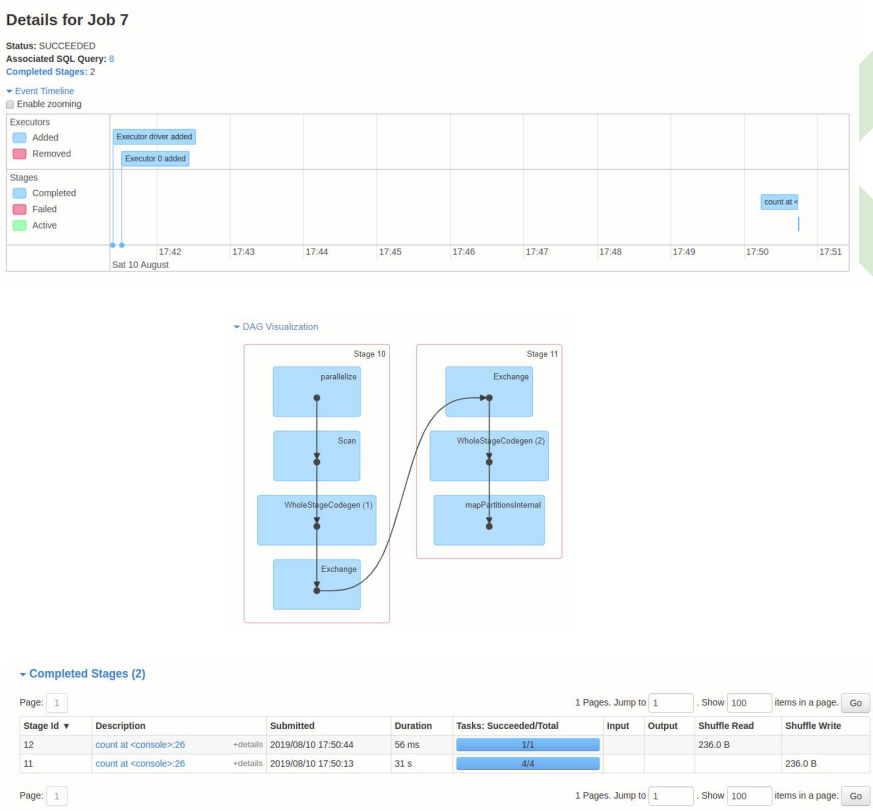
Jobs Tab

- Menampilkan summary job yang aktif maupun sudah selesai.
- Menunjukkan informasi status, durasi, dan progress semua job, dan timeline event secara keseluruhan.
- Untuk melihat informasi detail sebuah job, klik salah satu job yang dimaksud



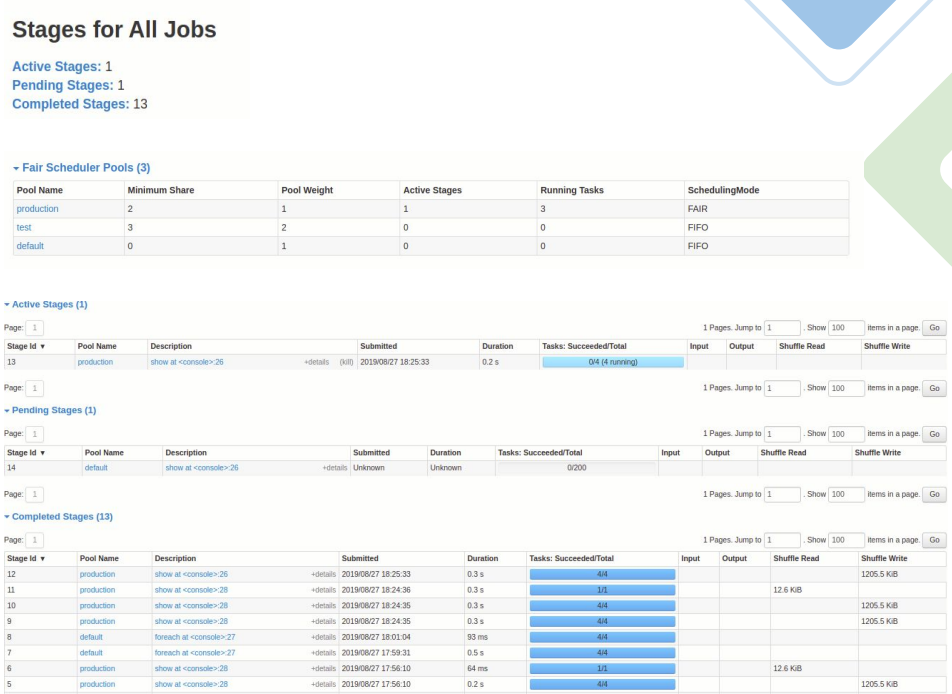
Jobs Detail

- Menampilkan detail info dari job tertentu, yaitu :
  - Status: running, succeeded, atau failed
  - Jumlah stage per status (active, pending, completed, skipped, failed)
  - SQL Query terkait : Link ke the sql tab
  - Event timeline: Displays secara kronologis event-event yang terkait dengan executor (added, removed) dan stage dari job ybs
  - Visualisasi DAG dari job tersebut
  - List dari stage per status stage



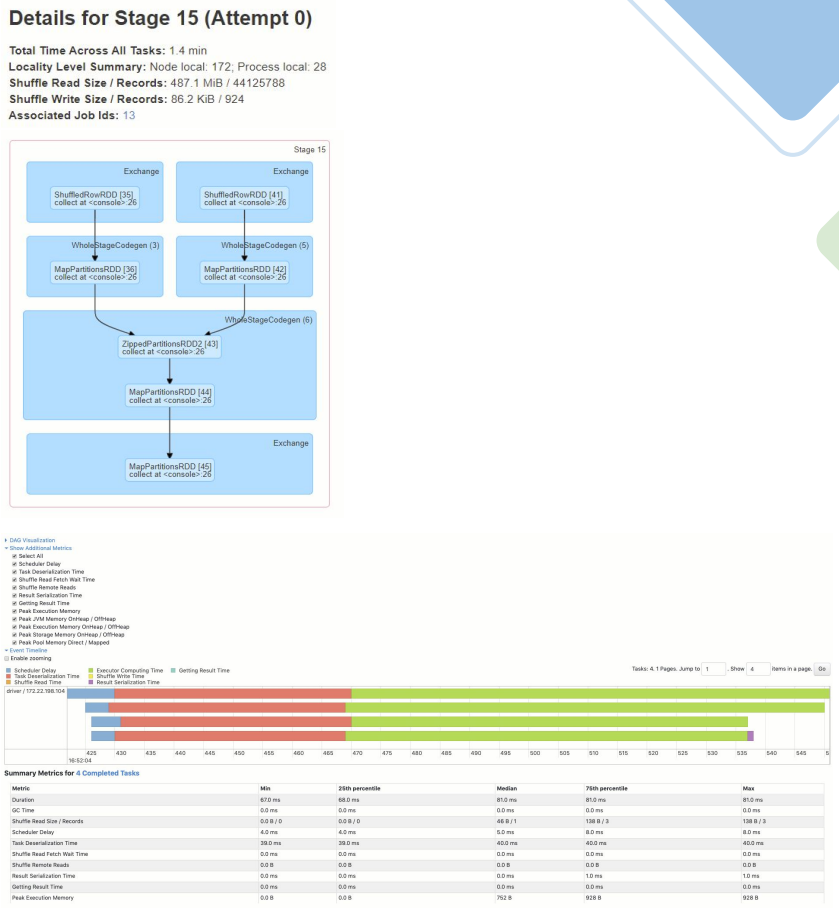
Stages Tab

- Menampilkan status seluruh stage dari seluruh job yang ada
- Menampilkan jumlah seluruh stages per status stage (active, pending, complete), serta list dari seluruh stage tersebut
- Untuk melihat informasi detail sebuah stage, klik salah satu stage yang dimaksud



Stage Detail

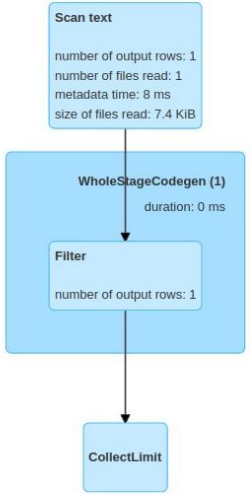
- Menampilkan informasi mengenai
  - Total waktu untuk menjalankan semua task
  - Locality level summary
  - Shuffle Read Size / Records
  - ID Job terkait
  - Visualisasi DAG untuk stage tersebut
  - Summary metrics untuk seluruh task, ditampilkan dalam bentuk tabel dan timeline



Details for Query 2

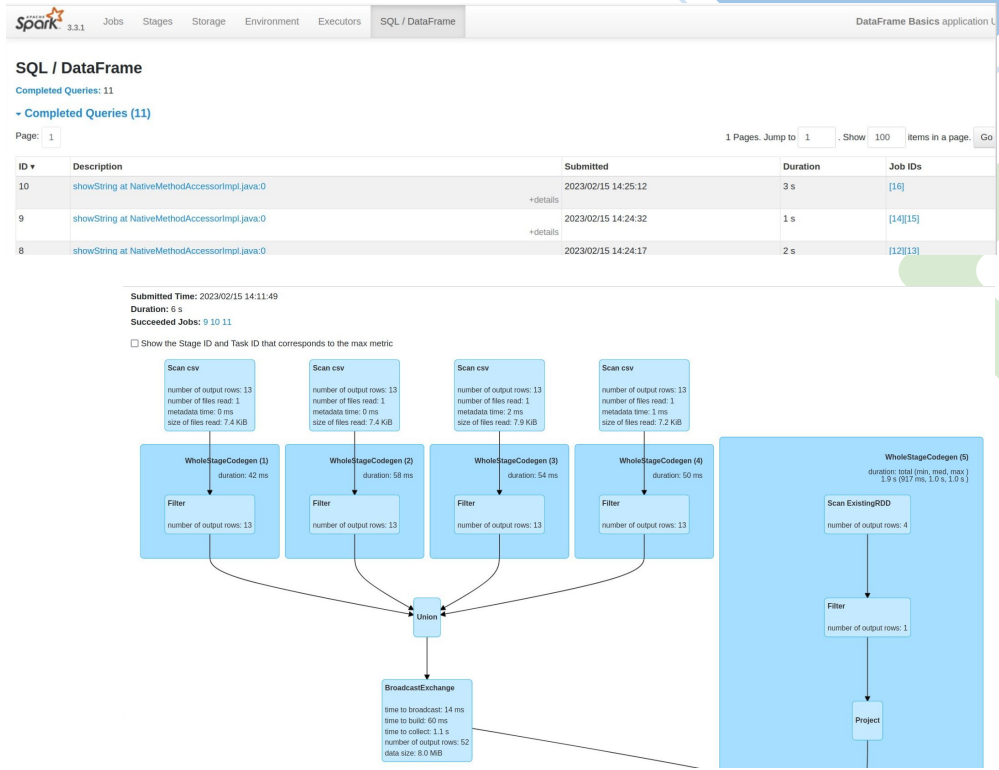
Submitted Time: 2023/02/15 14:11:26
Duration: 0.6 s
Succeeded Jobs: 3

☐ Show the Stage ID and Task ID that corresponds to the max metric



▾ Details

```
== Physical Plan ==
CollectLimit (3)
+- * Filter (2)
   +- Scan text  (1)


(1) Scan text
Output [1]: [value#65]
Batched: false
Location: InMemoryFileIndex [file:/home/hadoop/dataset/penumpang-09-2021.csv]
ReadSchema: struct<value:string>

(2) Filter [codegen id : 1]
Input [1]: [value#65]
Condition : (length(trim(value#65, None)) > 0)

(3) CollectLimit
Input [1]: [value#65]
```



06 Improving Performance

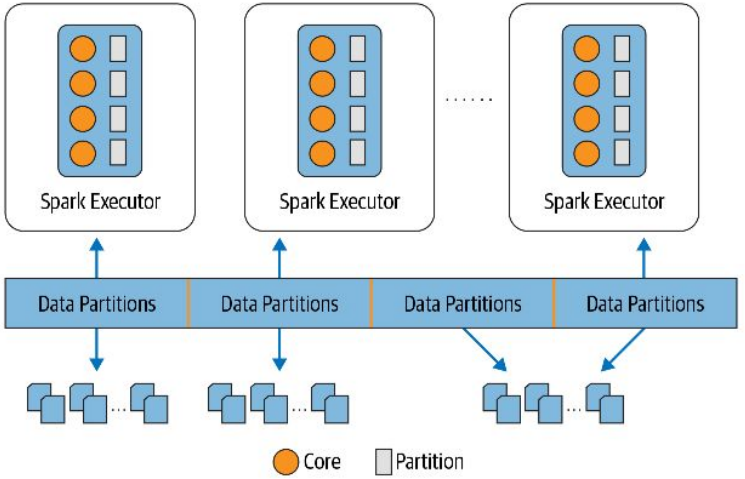
Pendahuluan

Ada beberapa konsep terkait kinerja Spark yang akan kita bahas secara singkat pada bagian ini, yaitu :

- Partisi
- Shuffle
- Caching

Spark Partitions

- Sebuah logical chunk dari dataset yang besar dan terdistribusi
- Satuan dasar paralelisme pada Apache Spark
- Perlu dipertimbangkan keseimbangan antara jumlah cpu core dan partisi untuk memaksimalkan paralelisasi
- Lebih banyak partisi menghasilkan *chunk* yang lebih kecil akan tetapi memerlukan lebih banyak task untuk menyelesaikan pekerjaan



Bagaimana Partisi Dibuat

- Created by Spark on Loading

  Ketika RDD/DataFrame dibentuk dari sumber data terdistribusi, Spark akan membuat satu partisi untuk setiap file blok secara default. Ukuran partisi ditentukan oleh **`spark.sql.files.maxPartitionBytes`** dengan nilai default 128MB.

- Created by User Command :

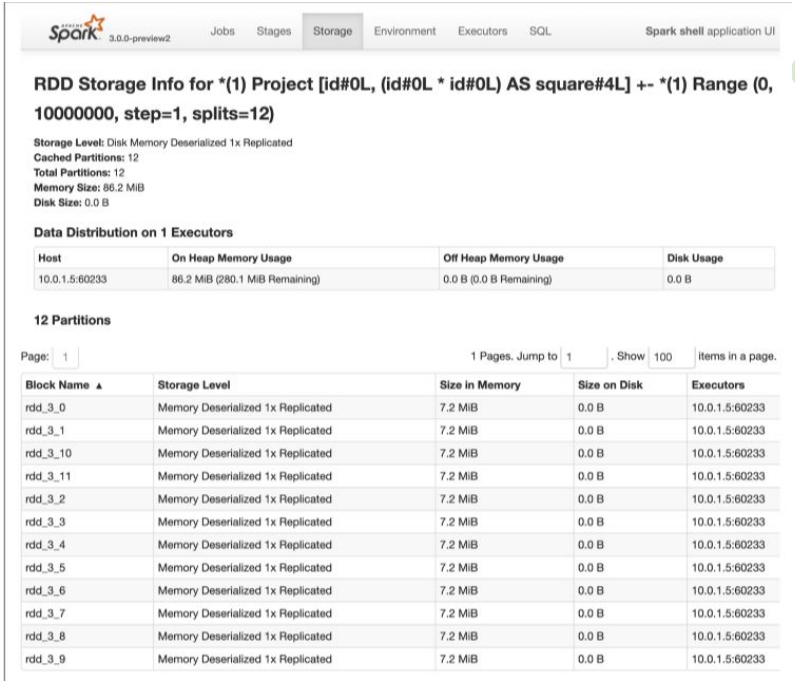
  `repartition()` and **`coalesce()`**. Repartition dapat meningkatkan atau mengurangi jumlah partisi, sedangkan coalesce hanya mengurangi jumlah partisi

- Shuffle partitions, created during the shuffle stage

  Dibuat selama operasi wide transformation seperti **`groupBy()`** or **`join()`**. Dikonfigurasi melalui parameter **`spark.sql.shuffle.partitions`**. Nilai defaultnya adalah 200.

Caching and Persistence of Data

- Untuk meningkatkan kinerja, Spark memungkinkan caching atau persisting DataFrames atau tabel yang sering diakses.
- Untuk melakukan caching, gunakan `dataFrame.cache()` atau `spark.catalog.cacheTable("tableName")`
- `persist()` memberikan kontrol lebih besar atas bagaimana dan di mana data disimpan: dalam memori atau disk, serialized atau unserialized



Caching and Persistence of Data

Cache atau Persist digunakan :

Jika perlu mengakses kumpulan dataset yang besar berulang kali untuk query atau transformasi. Sebagai contoh:

- DataFrames yang biasa digunakan selama pelatihan machine learning
- Ketika komputasi RDD mahal, caching dapat membantu mengurangi biaya pemulihan jika salah satu executor gagal

Cache tidak perlu digunakan:

Beberapa skenario yang mungkin tidak menjamin caching DataFrames Anda meliputi:

- DataFrames yang terlalu besar untuk dimuat ke dalam memori
- Transformasi ringan pada DataFrame yang tidak perlu sering digunakan

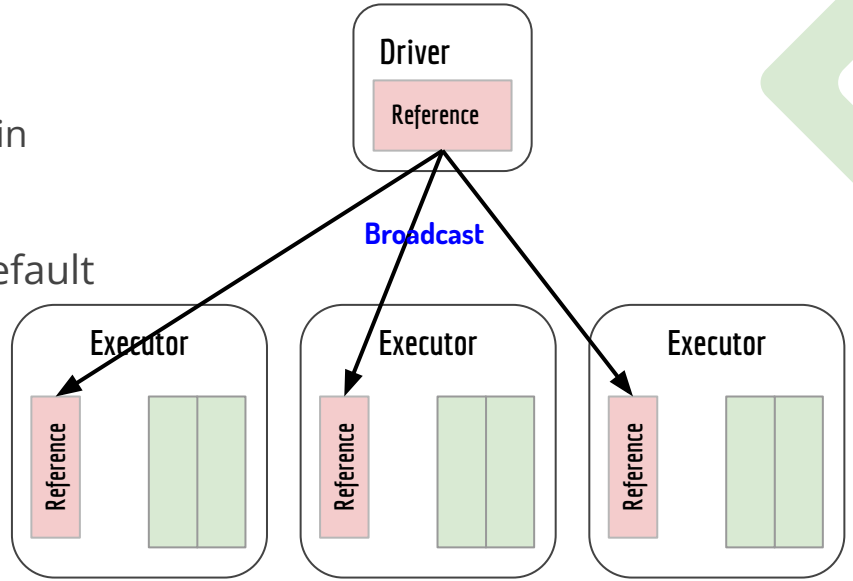
Optimasi Join

Spark memiliki beberapa strategi join, di antaranya :

- **Broadcast Hash Join** : join dataset dengan data reference berukuran kecil. Sebuah copy dari data reference akan di-broadcast ke seluruh executor untuk di-join
- **Shuffle Hash Join** : join dua dataset berukuran relatif besar. Kedua dataset akan di-shuffle berdasar join key, salah satu di-hash dan di-join
- **Shuffle Sort-Merge Join** : join dua dataset berukuran relatif besar. Kedua dataset akan di-shuffle berdasar join key, kemudian keduanya di-sort

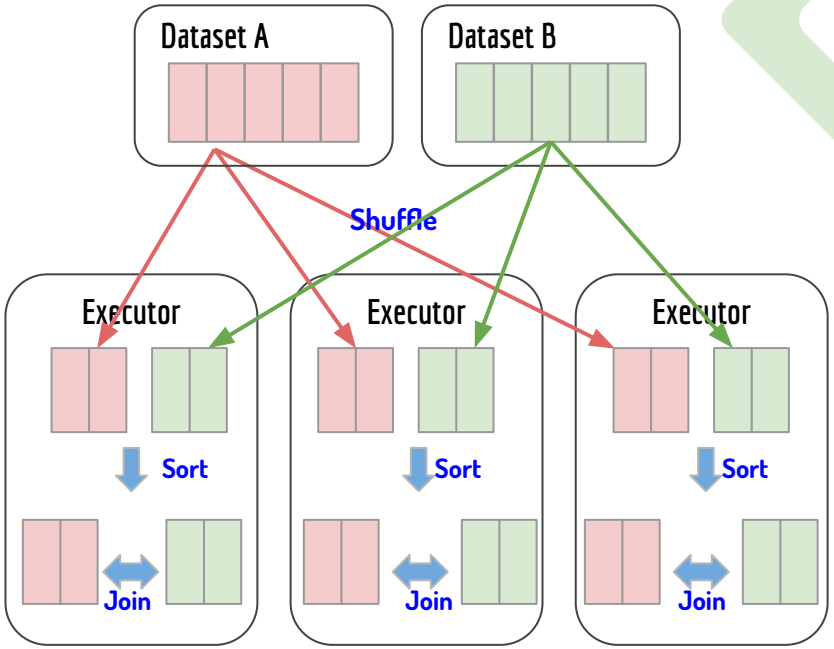
Broadcast Hash Join

- Langkah :
  1. Broadcast dataset kecil ke seluruh node
  2. Hash dataset kecil di setiap executor dan join
- Dataset yang akan di-broadcast harus < **spark.sql.autoBroadcastJoinThreshold** (default = 10 MB)
- Pros : tidak ada shuffle, tidak ada masalah skew
- Cons : Resiko out of memory (OOM) di driver node



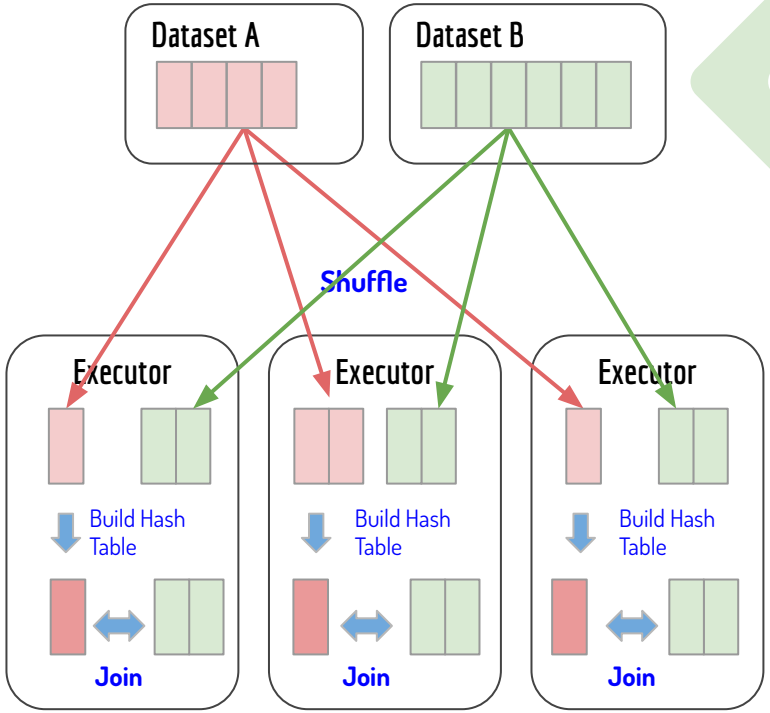
Shuffle Sort-Merge Join

- Langkah :
  1. Shuffle kedua dataset berdasarkan join-key
  2. Sort kedua dataset di tiap executor berdasarkan join-key
  3. Join berdasarkan join-key
- Merupakan metode default, yang diset melalui parameter **spark.sql.join.preferSortMergeJoin = True**
- Pros : Sesuai untuk join dataset yang besar
- Cons : Perlu shuffle dan sort, terpengaruh oleh data skew



Shuffle Hash Join

- Langkah :
  1. Shuffle kedua dataset berdasarkan join-key
  2. Hash dataset yang lebih kecil, dan lakukan join
- Satu dataset harus lebih kecil dari yang lain (sekitar 1: 3), dan berukuran < (**spark.sql.autoBroadcastJoinThreshold * spark.sql.shuffle.partitions**)
- Pros : Sesuai untuk join dua dataset yang salah satunya lebih besar, tidak perlu sort
- Cons : Perlu shuffle, terpengaruh oleh data skew, dan ada resiko OOM

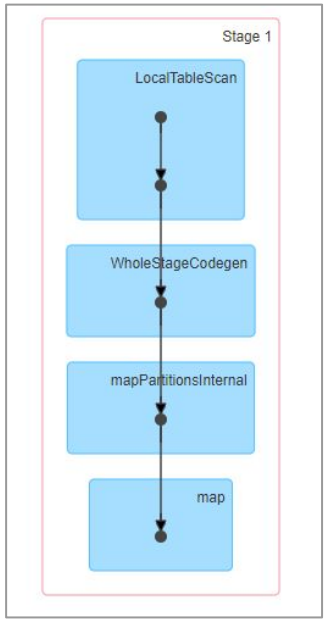


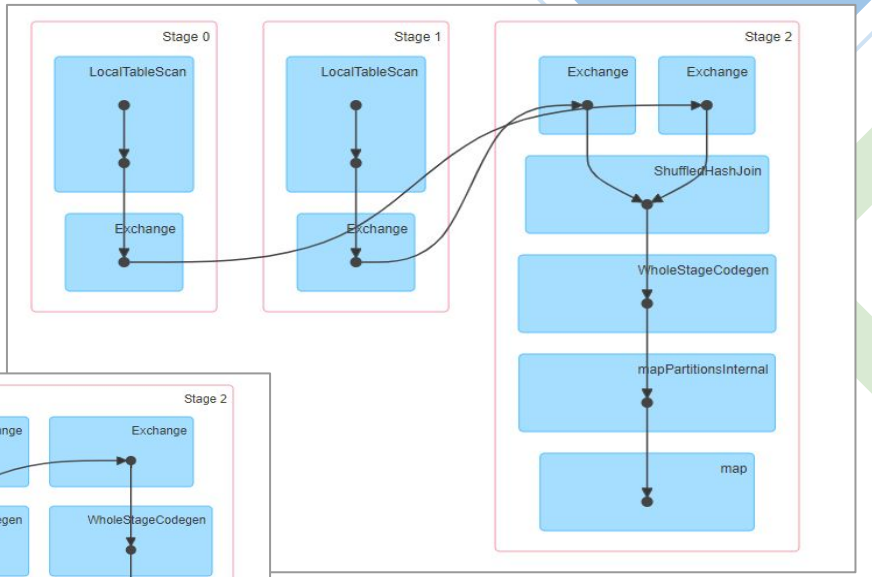
Broadcast Join : Code Example

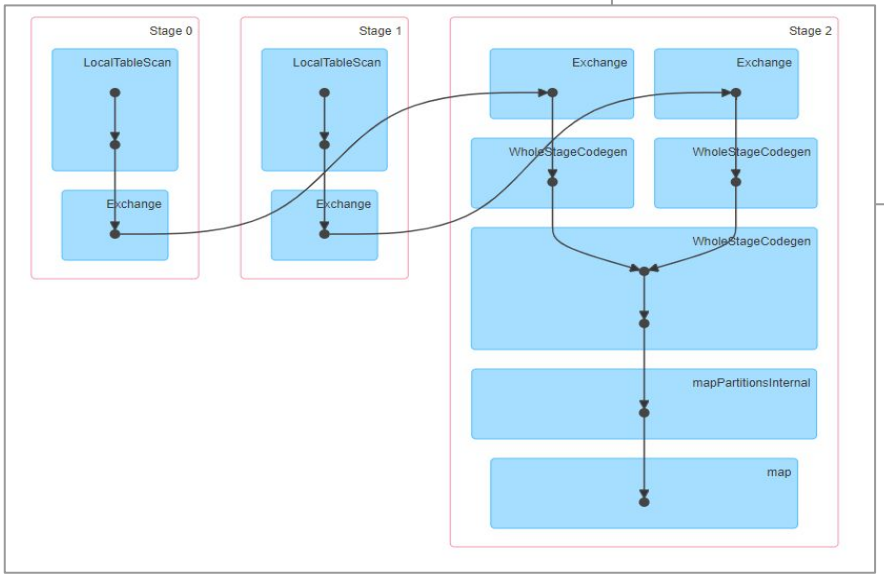
```
#Enable broadcast Join dan set limit ukuran dataframe yang bisa di-broadcast
spark.conf.set("spark.sql.autoBroadcastJoinThreshold", 100663296)

#Disable broadcast Join
spark.conf.set("spark.sql.autoBroadcastJoinThreshold", -1)
```

```
data_DF.join(
  broadcast(reference_DF),
  data_DF("join_key") <=> reference_DF("join_key")
)
```

Perbandingan Execution Plans







THE REAL TRAINING BEGINS WHEN THE CLASS ENDS

Module development team

M. Urfah
Sigit Prasetyo

document version : 1.00.2011.23

DATALearns247 is educational program developed by Solusi247 focusing on building Indonesian data talents through curriculum based on the real world experience in big data and artificial intelligence implementation