DATALEARNS247

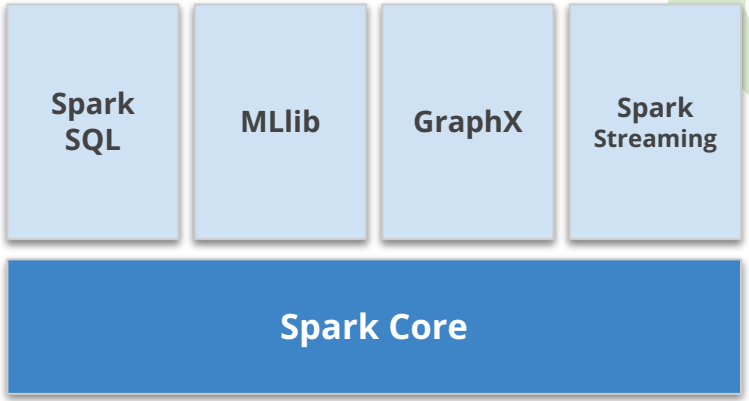
Apache Spark Data Processing

Big Data Engineering

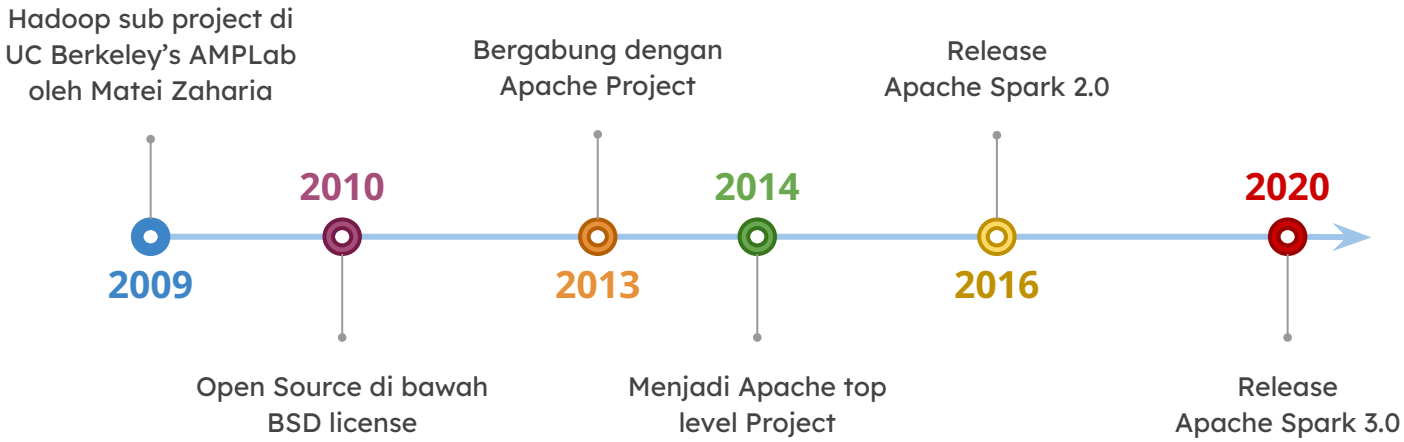
01 Overview

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.

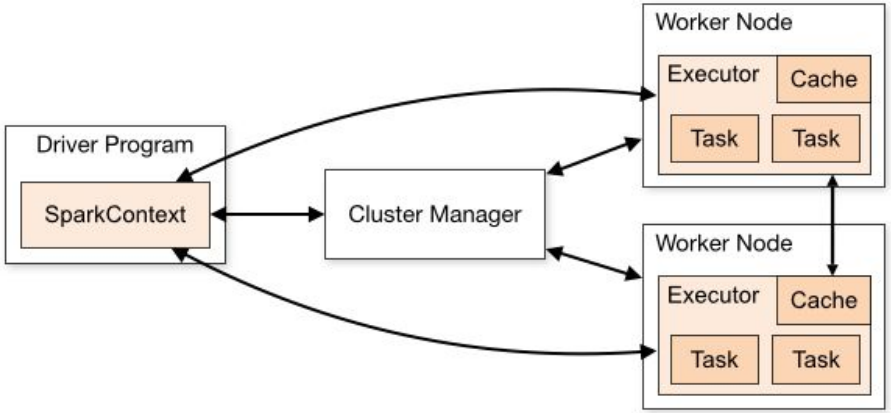
Wide Support

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

02 Bagaimana Spark Bekerja ?

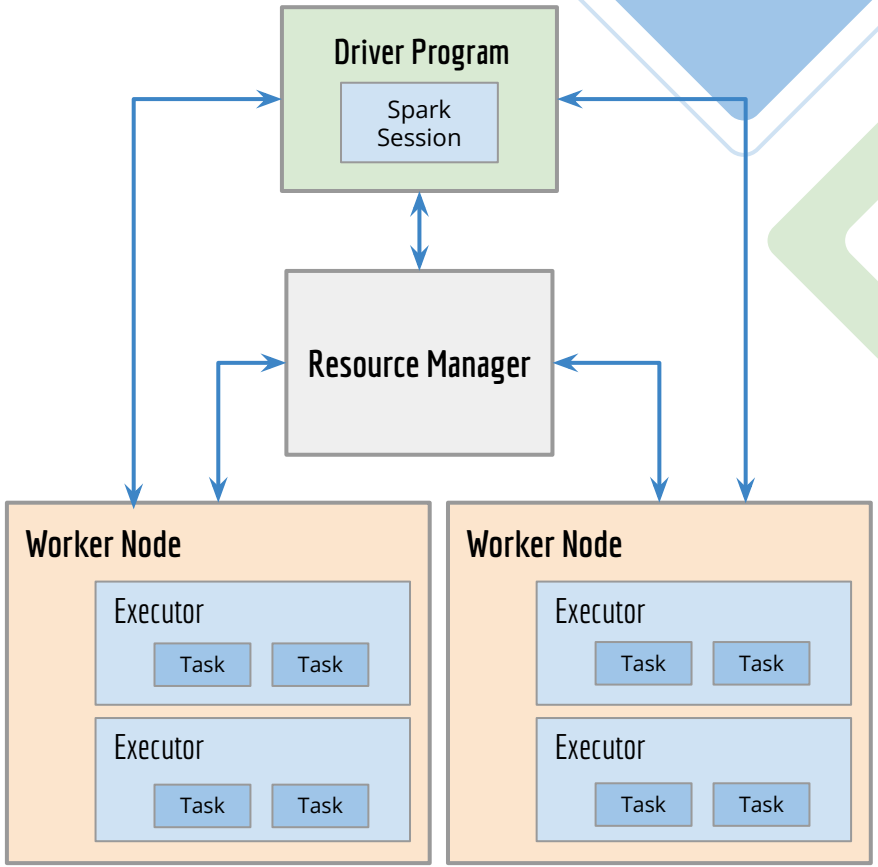
Arsitektur Spark

- Apache Spark memiliki arsitektur master/slave, dengan 3 komponen utama : *driver program, executor,* dan *cluster manager.*
- **Driver program** berfungsi sebagai master, sedangkan **executor** bertindak sebagai slave.
- **Cluster manager** merupakan komponen eksternal yang mengatur dan membagi resource dalam klaster spark ke aplikasi-aplikasi yang berjalan di atasnya.



Aplikasi Spark

- Setiap aplikasi Spark memiliki satu *driver program* sebagai koordinator pusat
- Driver membuat object **Spark Session/SparkContext** untuk berkomunikasi dengan *Resource Manager*
- Resource Manager mengalokasikan *executor*, yang akan menjalankan komputasi dan menyimpan data untuk aplikasi
- Spark Session mengirimkan task untuk dieksekusi oleh executor
- Resource Manager yang dapat digunakan : **Spark**, **YARN, Mesos,** dan **Kubernetes**,



Jenis Cluster Manager

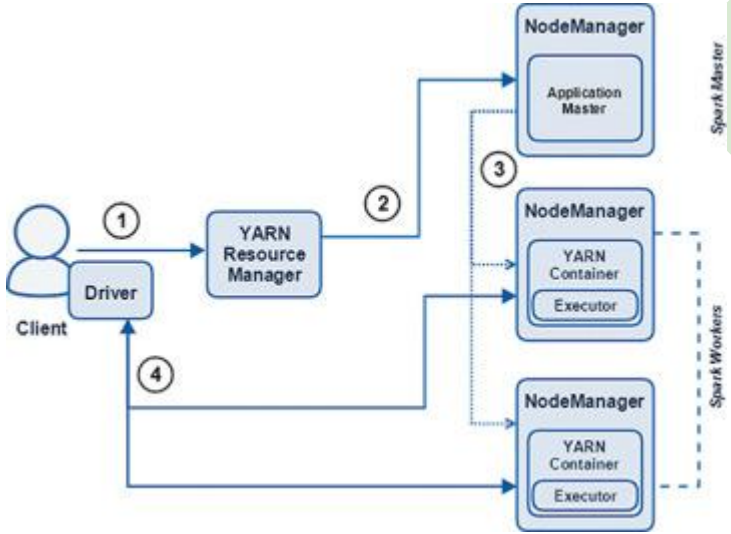
- **Standalone** – cluster manager sederhana, disertakan dengan Spark, untuk memudahkan setup klaster Spark tanpa cluster lain.
- **Apache Mesos** – cluster manager umum yang juga dapat menjalankan Hadoop MapReduce dan aplikasi layanan. (Deprecated)
- **Hadoop YARN** – resource manager di Hadoop 3.
- **Kubernetes** – sistem open-source untuk mengotomatiskan deployment, scaling, dan manajemen aplikasi dalam container.

Spark Execution Mode

Spark memiliki 3 mode eksekusi/deployment

- **Local** : Eksekusi program secara lokal, di notebook atau PC. Tidak menggunakan cluster yang terdistribusi
- **Client mode** : Spark driver berada di client di luar cluster, executor berada pada worker node.

  Client harus terus online selama driver program berjalan. Sesuai untuk mode interaktif, seperti notebook dan shell (`spark-shell, pyspark, spark-sql, dll`)



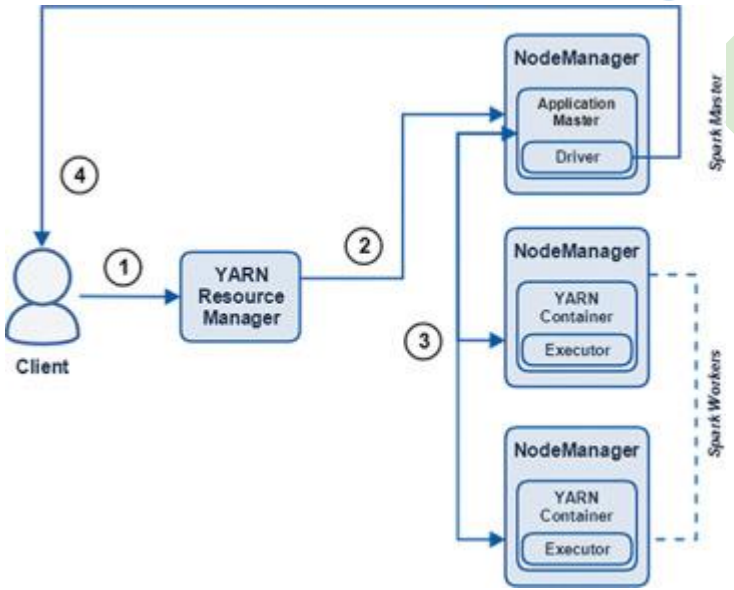
Client mode execution

Spark Execution Mode

- **Cluster mode** : Spark driver dan worker berjalan di node yang ada di dalam cluster.

  Client tidak perlu online selama program berjalan. Sesuai untuk job non-interaktif

```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```



Cluster mode execution

Contoh Spark Deployment

```
spark-submit \
   --master yarn \
   --deploy-mode cluster \
   --conf spark.executor.memory 4G \
   --class org.apache.spark.examples.SparkPi \
   /path/to/examples.jar \
   10
```

```
spark-submit --master yarn --deploy-mode cluster --py-files file1.py,file2.zip myPythonApp.py
```

Spark Config

Ada 3 cara setting konfigurasi spark :

- Melalui program

```
SparkSession.builder
      .config("spark.some.config.option1", "some-value1")
      .config("spark.some.config.option2", "some-value2")
```

- Melalui parameter spark-submit

```
spark-submit \
      --executor-memory 20G \
      --total-executor-cores 100 \
      --conf "spark.some.config.option1"="some-value1"
```

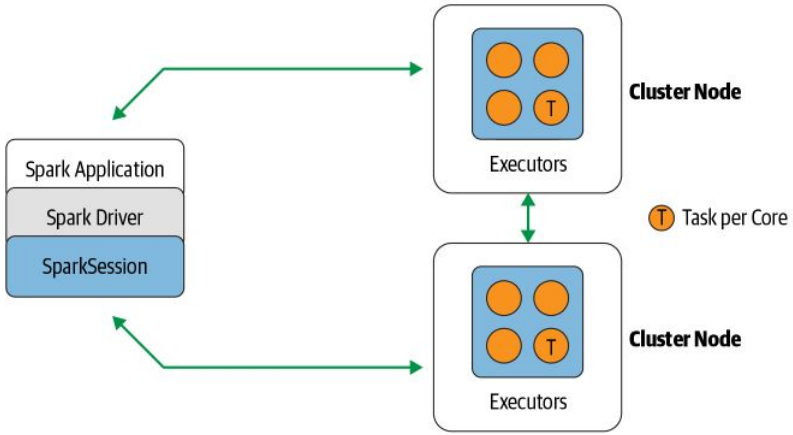
- Melalui file spark-default.conf

```
#spark-default
spark.io.compression.lz4.blockSize  128kb
spark.master      yarn
```

SparkSession Code

Contoh kode untuk membuat SparkSession

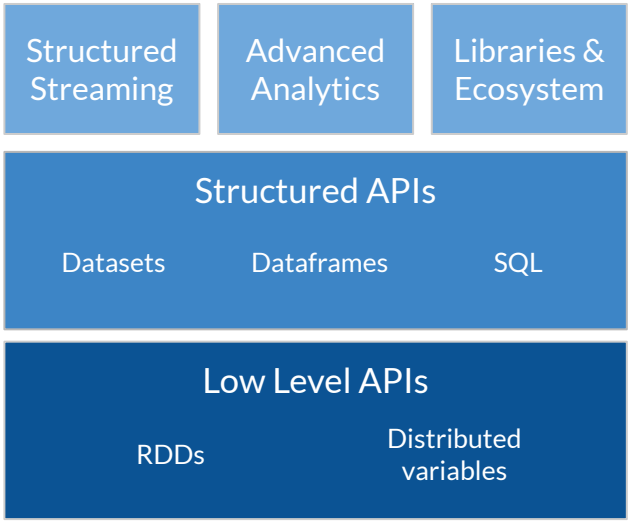
```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder
                    .appName('My Application')
                    .config("option1", "value1")
                    .getOrCreate()

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



Spark APIs

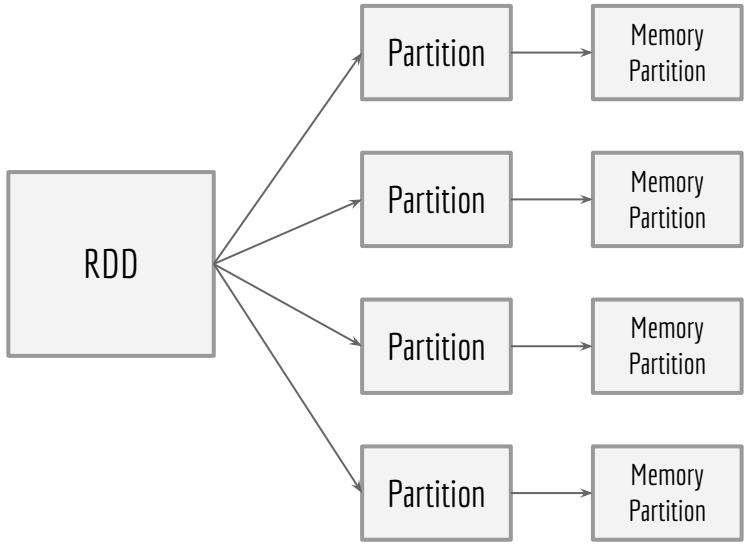
Spark memiliki dua set API dasar:

- API level rendah yang tidak terstruktur : **RDD** dan **Distributed variables**
- API level lebih tinggi yang lebih terstruktur : **Dataset**, **DataFrame** dan **SQL APIs**



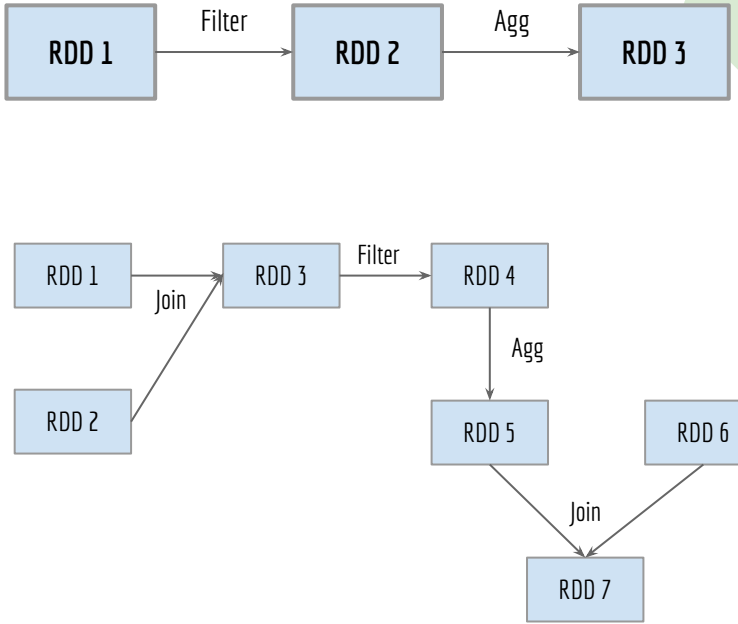
Spark RDD (Resilient Distributed Dataset)

- Abstraksi data terdistribusi
- Menyimpan referensi ke objek Partition
- Setiap object Partition menunjuk ke subset data
- Partition di-assign ke node eksekutor di dalam cluster
- Secara default, tiap partisi disimpan di memory



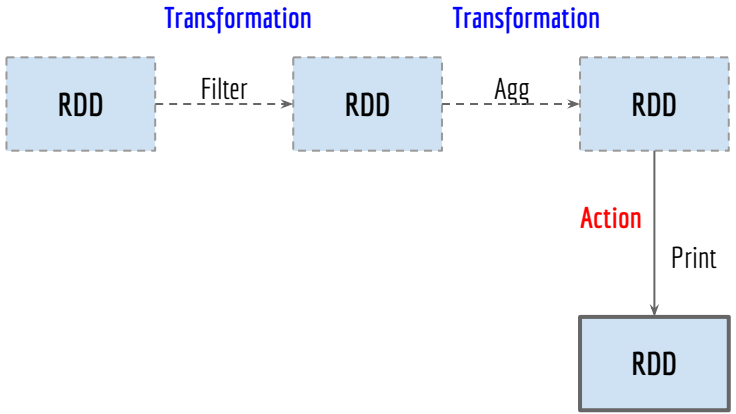
Immutability

- 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**



Lazy Evaluation

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



Spark Actions vs Transformations

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.

+

ACTIONS

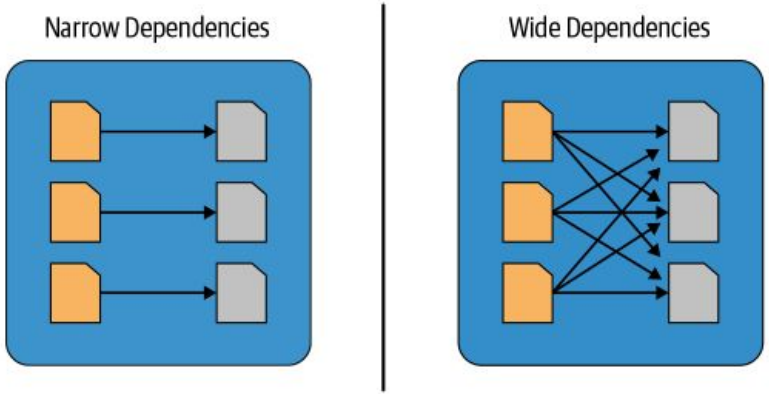
show(), collect(), count(), first(), etc.

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

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**

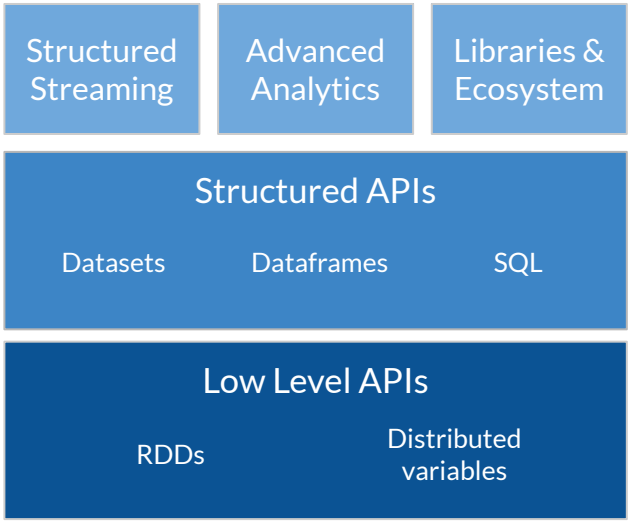


03 Spark Structured API

Spark APIs

Spark memiliki dua set API dasar:

- API level rendah yang tidak terstruktur : **RDD** dan **Distributed variables**
- API level lebih tinggi yang lebih terstruktur : **Dataset**, **DataFrame** dan **SQL APIs**

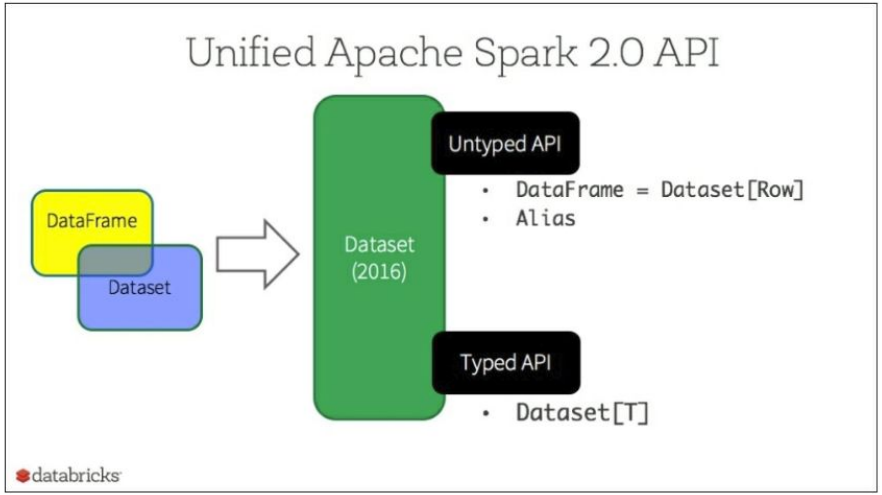


Spark Structured APIs

- DataFrames
  DataFrame menggambarkan tabel dengan baris dan kolom. Konsepnya mirip dengan DataFrame R dan Python (Pandas).

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

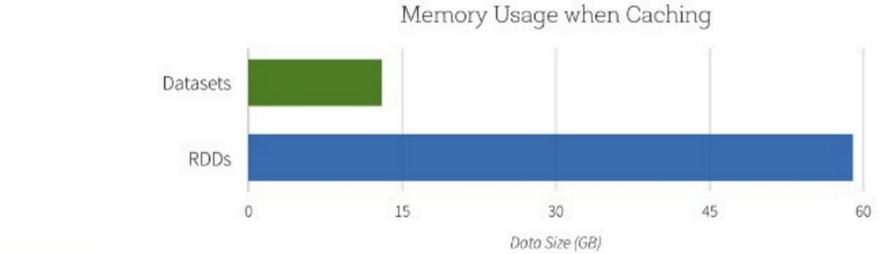
- SQL tables and views

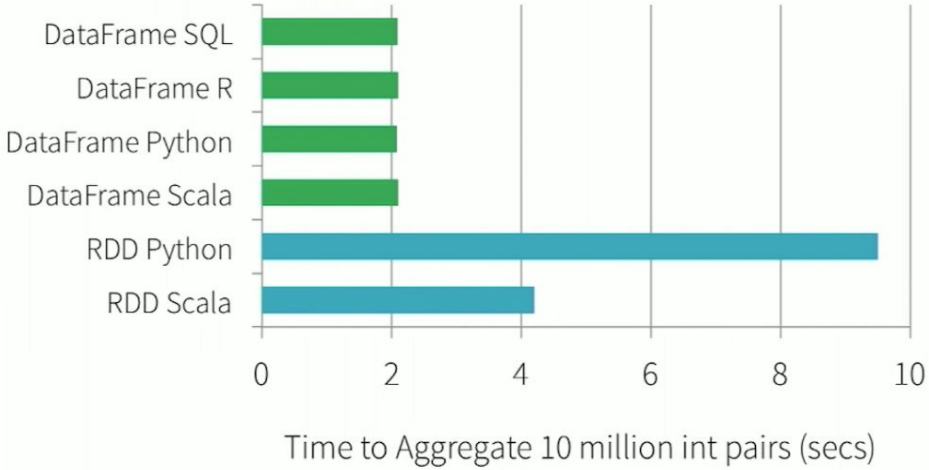


Mengapa Menggunakan Structured API?

- 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

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





RDD vs DataFrame

Create

```
rdd = spark.sparkContext.parallelize(listData)
rdd.collect()
```

```
df = spark.createDataFrame(list, ["col1", "col2"])
df.show()
```

Filter

```
rdd.filter(lambda x: x[0] == "value1")
```

```
df.filter(df["col1"] == "value1")
```

Aggregate

```
rdd.map(lambda x: (x[0], (x[1], 1)))
   .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))
   .map(lambda x: (x[0], x[1][0]/x[1][1]))
```

```
df.groupBy("col1").agg(avg("col2"))
```

Spark vs Pandas DataFrame

Spark DataFrame	Pandas DataFrame
Relatively difficult to use	Easier to use
Distributed	Not distributed
Lazy Execution. A task is not executed until an action is performed	Eager Execution. A task is executed immediately
Immutable	Mutable
Scalable	Not scalable
Assures fault tolerance	Does not assure fault tolerance
Column index	Column & row index

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

Spark SQL

- Fungsi `spark.sql` digunakan untuk melakukan pemrosesan data dengan perintah SQL
- Spark mendukung query SQL ANSI dan HiveQL
- Untuk menggunakan SQL di atas dataframe, buat temporary view

Spark SQL on DataFrame

- Kita bisa menjalankan SQL query dengan spark.sql() tanpa menggunakan database
- Ketika temporaryView disimpan dengan perintah saveAsTable tanpa database, spark menyimpannya sebagai file parquet di direktori `spark_warehouse/nama_tabel`

Create

```
df.createOrReplaceTempView("viewname")
```

Execute SQL

```
spark.sql(“select * from viewname”)
spark.sql(“select col1, sum(col2) as sum_col2 from viewname group by col1”)
```

Save as Table

```
df.write.saveAsTable(name="tablename", mode="overwrite")
```

Hands-On

SQL Query dengan DataFrame

Deskripsi

Salah satu fitur yang sangat memudahkan di Spark adalah penggunaan SQL Query untuk melakukan pemrosesan data.

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

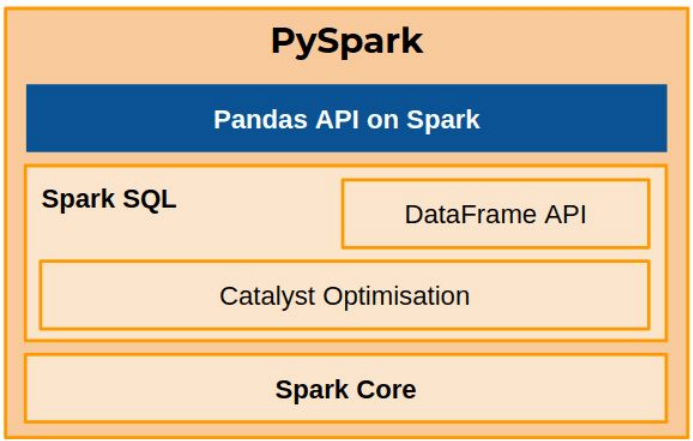
04 Pandas API on Spark

Mengapa Pandas dan Spark

- Pandas adalah library pemrosesan data yang lengkap dan powerful, namun Pandas hanya dapat berjalan di satu mesin, sehingga tidak *scalable*.
- Spark DataFrame saat ini sudah cukup mudah penggunaannya
- Tetapi, perbedaan antara Pandas DataFrame dan Spark DataFrame menyebabkan ‘biaya’ dalam hal **waktu belajar** maupun **migrasi pipeline**
- Pandas API on Spark memadukan keunggulan Pandas dan Spark, dan mengurangi ‘biaya’ tersebut

Pandas API on Spark

- Awalnya merupakan library terpisah yang dibuat oleh Databricks pada tahun 2019, bernama **Koalas**
- Sejak Spark 3.2 dimasukkan ke dalam main package Spark sebagai **Pandas API on Spark**
- Memiliki sintaksis yang sama dengan Pandas tetapi menggunakan PySpark DataFrame di bawahnya

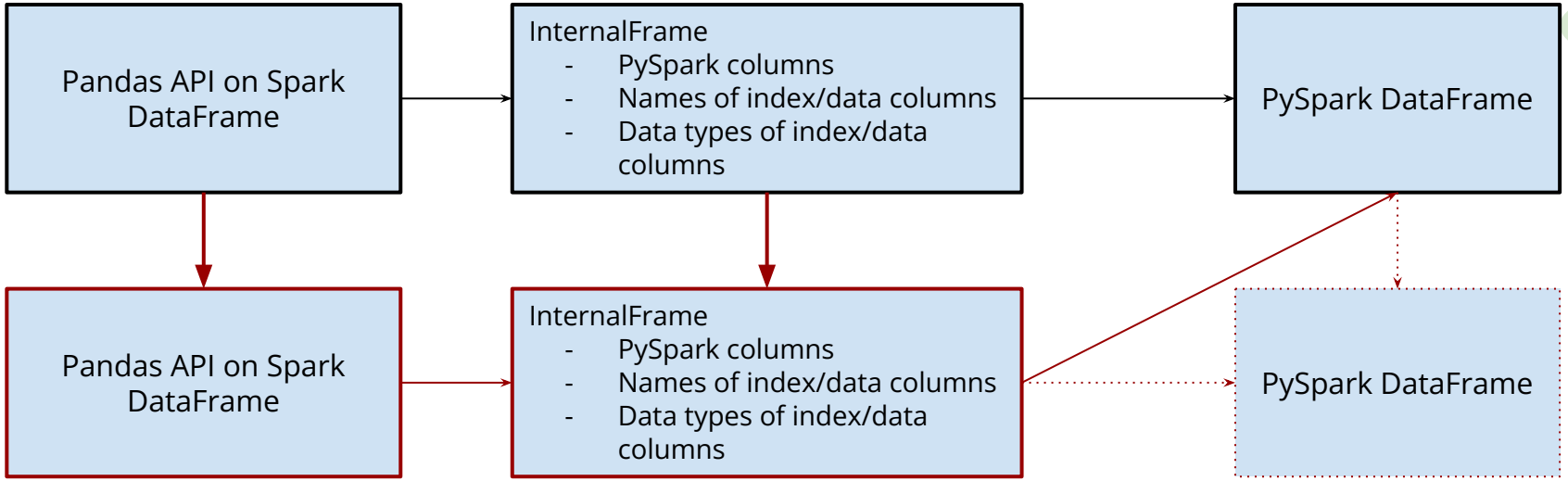



```
from pyspark.pandas import read_csv

psdf = read_csv("data.csv")
psdf.columns = ["x", "y", "z"]
psdf["x2"] = psdf.x * psdf.x
```

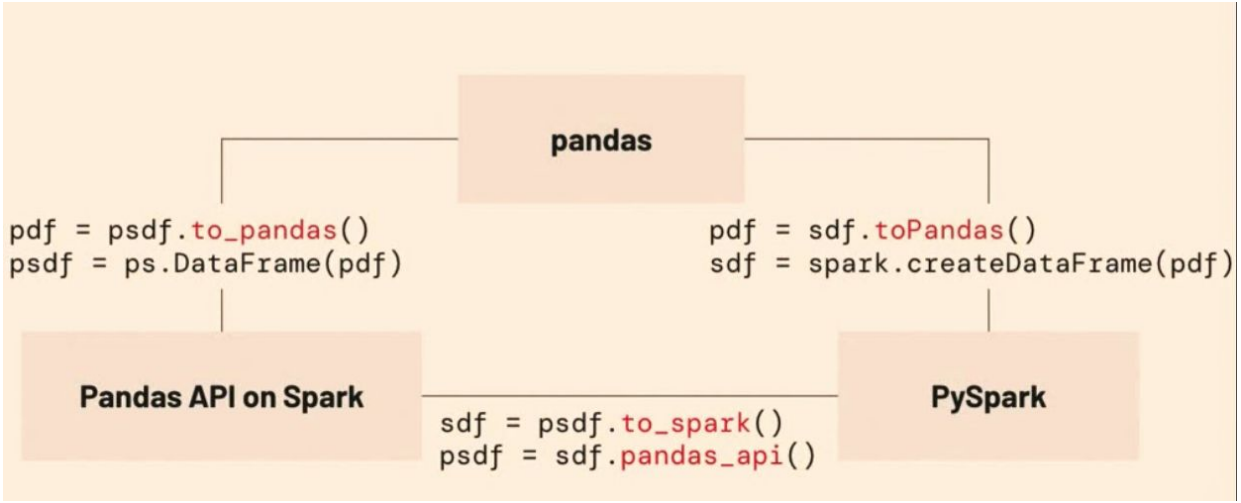
Pandas API vs PySpark DataFrame

	Pandas API on Spark	PySpark DataFrame
Compliance	Pandas DataFrame	Relations (tables)
Identifier (index)	Rows, Columns	Column
Mutability	Mutable	Immutable
Execution Engine	Spark SQL	Spark SQL

Bagaimana Pandas API on Spark Bekerja



Konversi PySpark - Pandas - Pandas API on Spark



Indexing

- dataFrame pandas-on-Spark mereplikasi fungsi pengindeksan Pandas.
- Jika kolom indeks tidak ditentukan, akan digunakan salah satu dari 3 jenis indeks default berikut :
  - Sequence
  - Distributed-sequence
  - Distributed

	sequence (default)	**distributed sequence**	**distributed**
Distributed computation	No, in a single worker node	Yes	Yes
Performance	Bad for large dataset	Good enough	Good
Sequential indexing	Yes	Yes	No
Operation on different dataframes	Yes	Yes	No

Best Practises

- Hindari komputasi pada satu partisi
- Jangan gunakan indeks default *sequential*
- Tentukan kolom indeks setiap kali indeks diperlukan
- Ingat bahwa data yang diproses terdistribusi
- Gunakan “spark.explain()” untuk menampilkan Physical Plan dan mendeteksi apakah ada shuffle

Best practise secara detail :
https://spark.apache.org/docs/latest/api/python/user_guide/pandas_on_spark/best_practices.html

THE REAL TRAINING BEGINS WHEN THE CLASS ENDS

Module development team

M. Urfah
Sigit Prasetyo

document version : 2.00.0203.23

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