

Apache Spark Data Processing

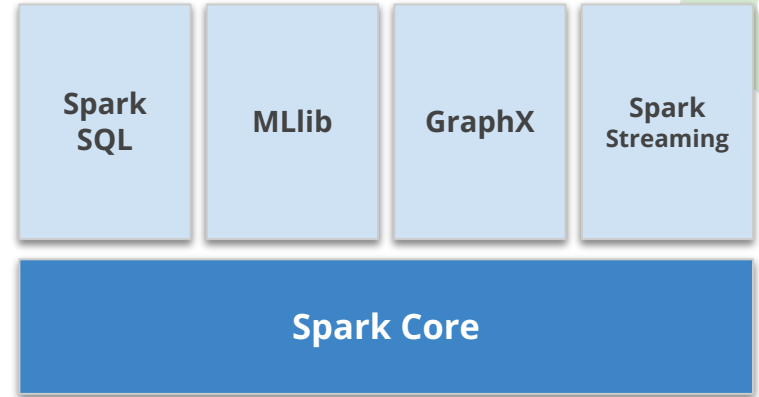
Big Data Engineering



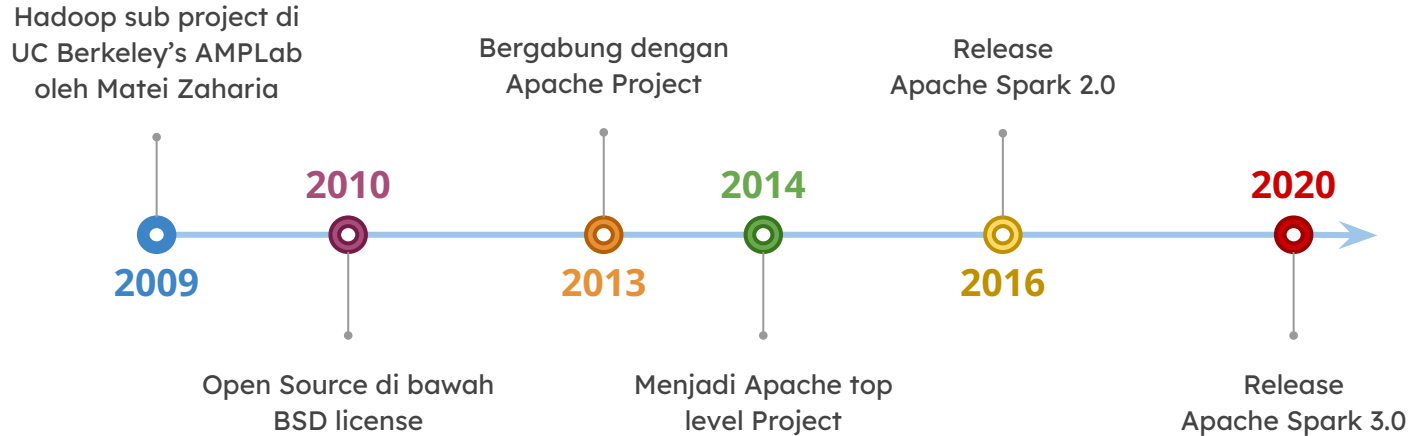
01 | Overview

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

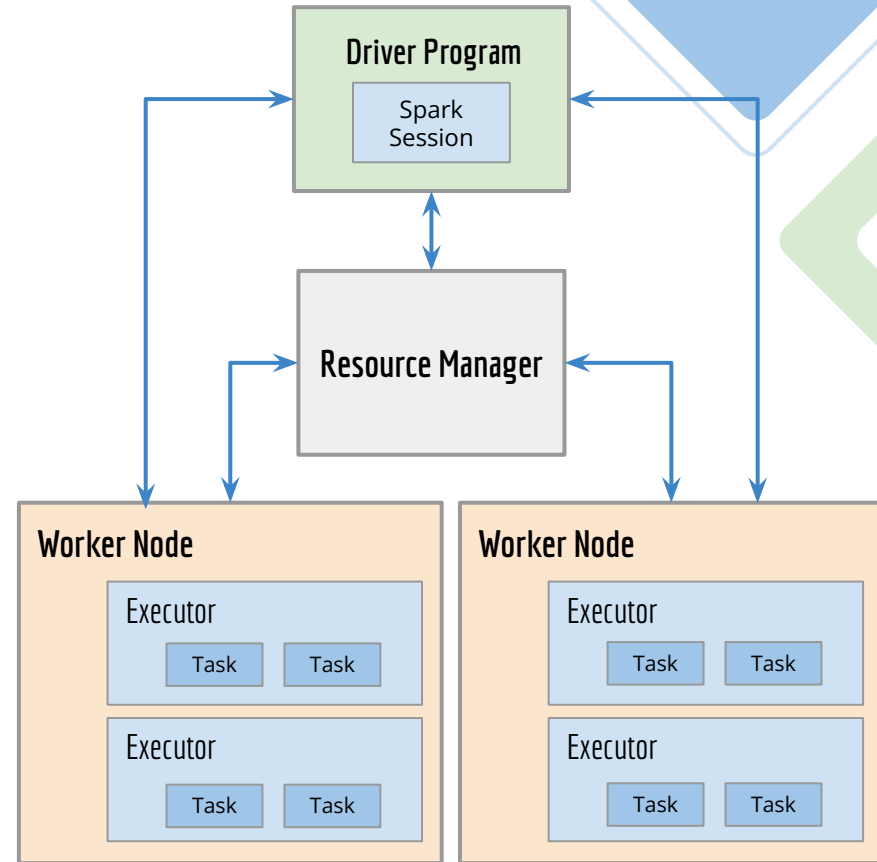
Spark menggunakan arsitektur master/slave. Setiap Spark program memiliki satu koordinator pusat disebut **driver** yang berkomunikasi dengan banyak worker yang terdistribusi (**executor**).

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**,



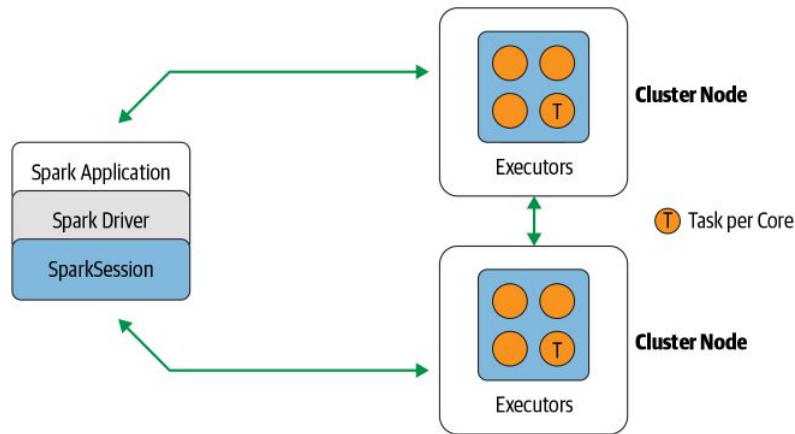
SparkSession Code

Contoh kode untuk membuat SparkSession

```
import pyspark
from pyspark.sql import SparkSession

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

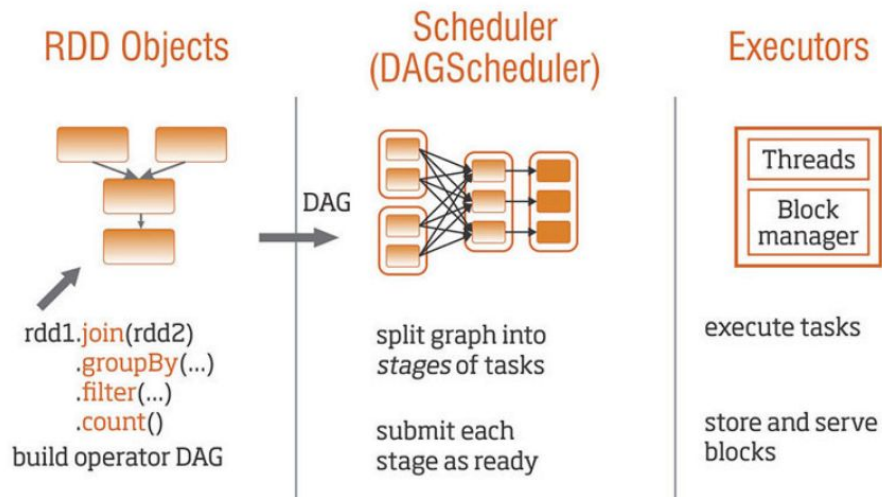
#read file as Spark dataframe
df = spark.read.csv(...)
```



Spark Abstractions

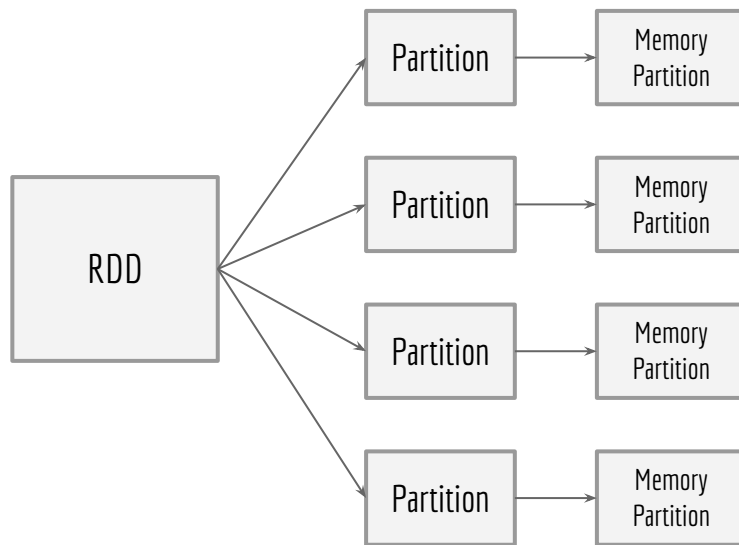
Proses dalam Spark didasarkan pada 2 konsep utama :

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



Spark RDD

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



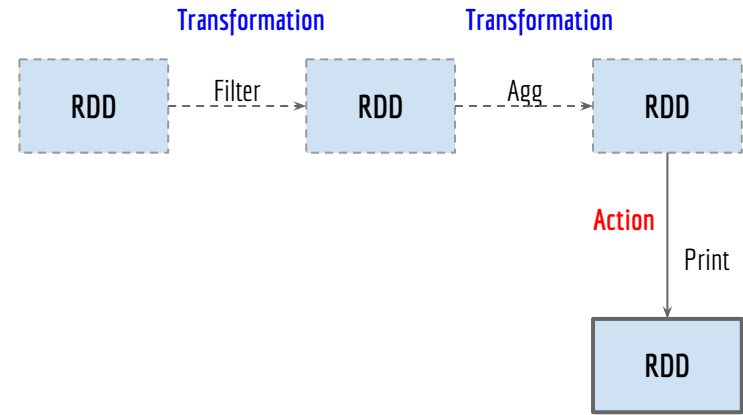
Immutability

- 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

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.

Spark Operations =

+

Fungsi yang mengakses data di RDD.

Action mentrigger eksekusi semua transformasi terkait untuk mendapatkan data yang diperlukan.



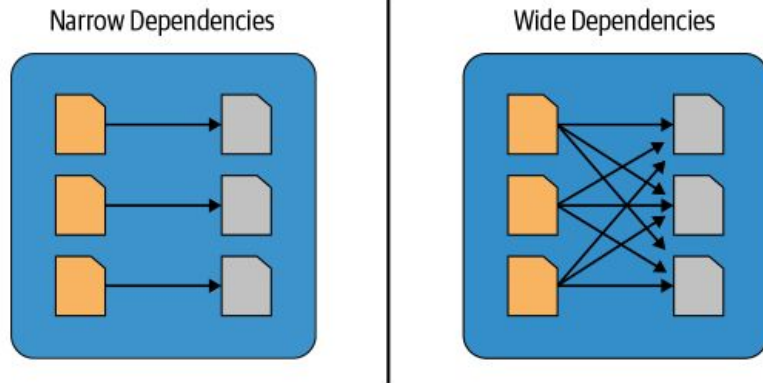
ACTIONS

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

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

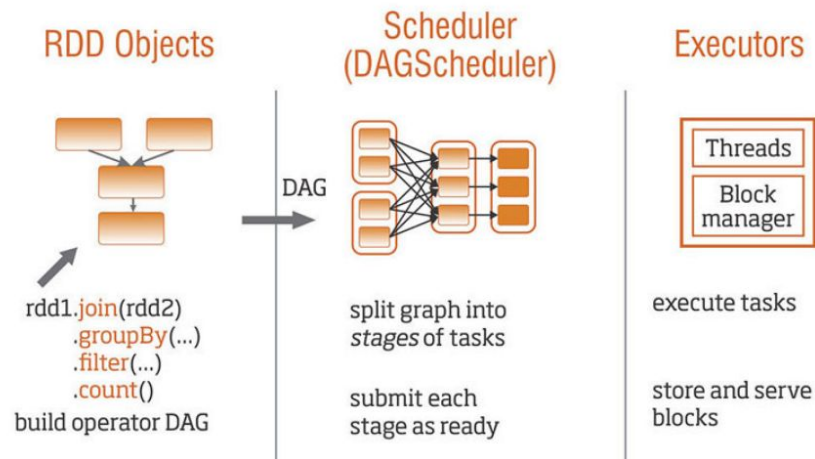
Narrow and Wide Transformations

- **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

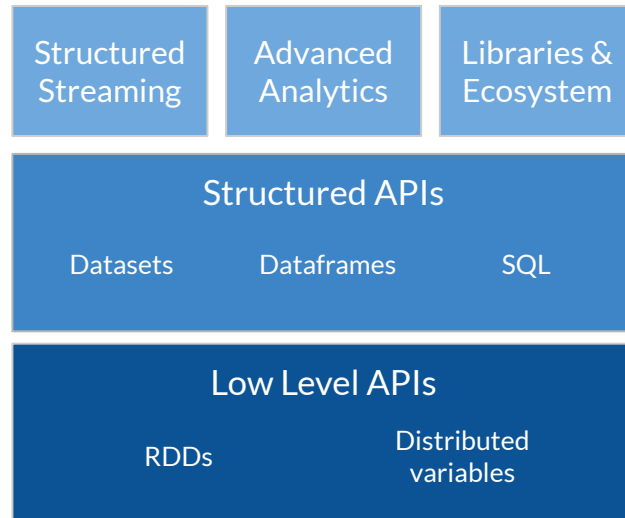
Spark memiliki dua set API dasar:

- 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.

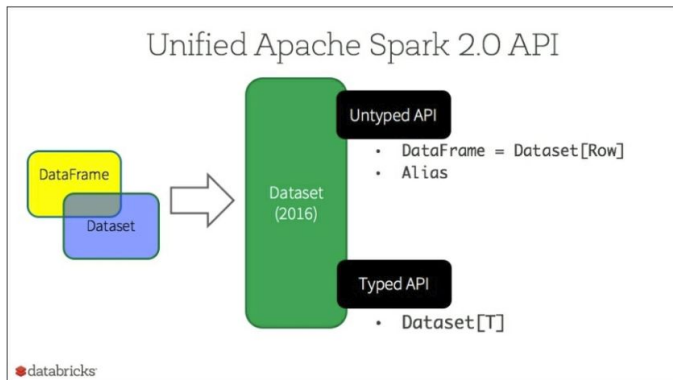
■ Datasets

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.

	SQL	DataFrames	Datasets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time



Mengapa Structured API?

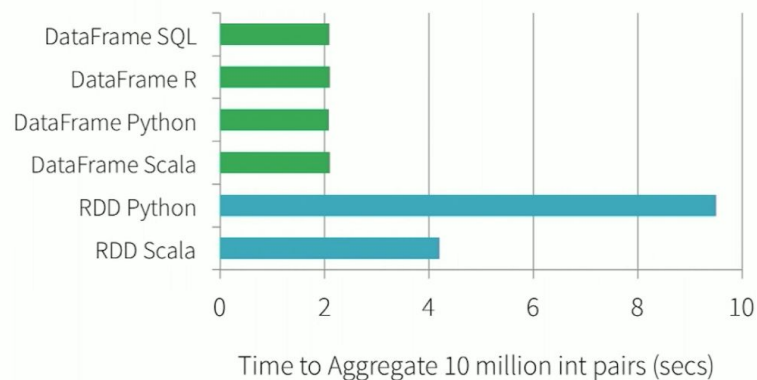
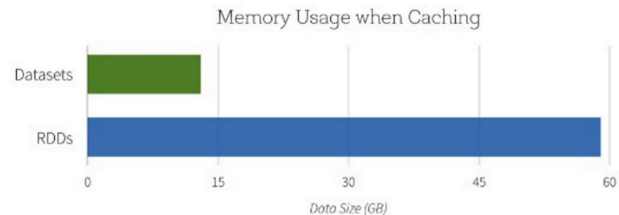
- Mudah digunakan → tell Spark **what to do** not **how to do it**

- 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.

databricks



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

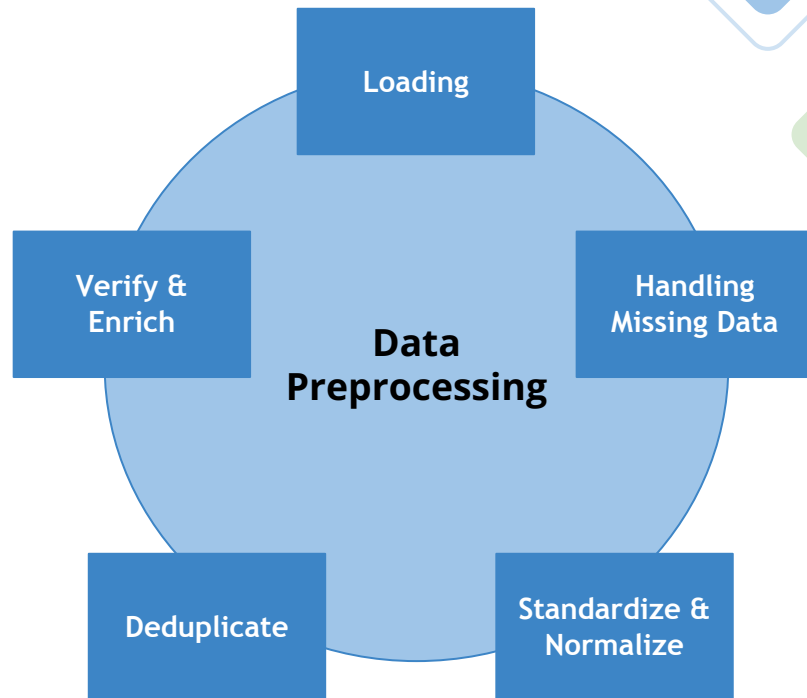


04 | Data Cleaning and Enrichment

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:

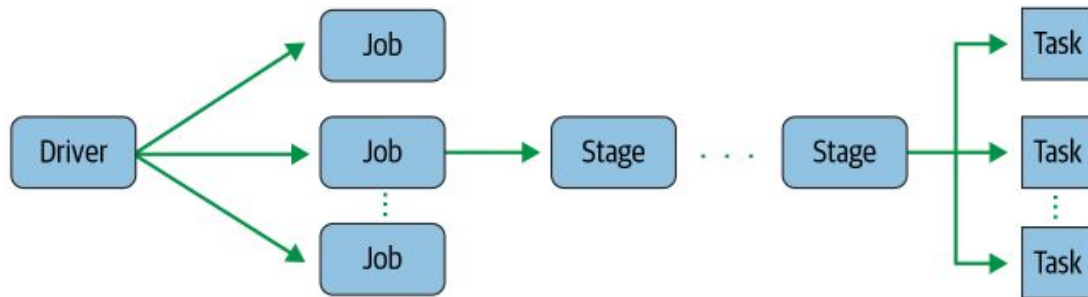
- Penggunaan spark dataframe untuk membaca file csv
- Membersihkan data, yaitu
 - Menangani missing value
 - Menangani data invalid
 - Menangani data duplikat
- Enrichment dengan data referensi



05 | Monitoring

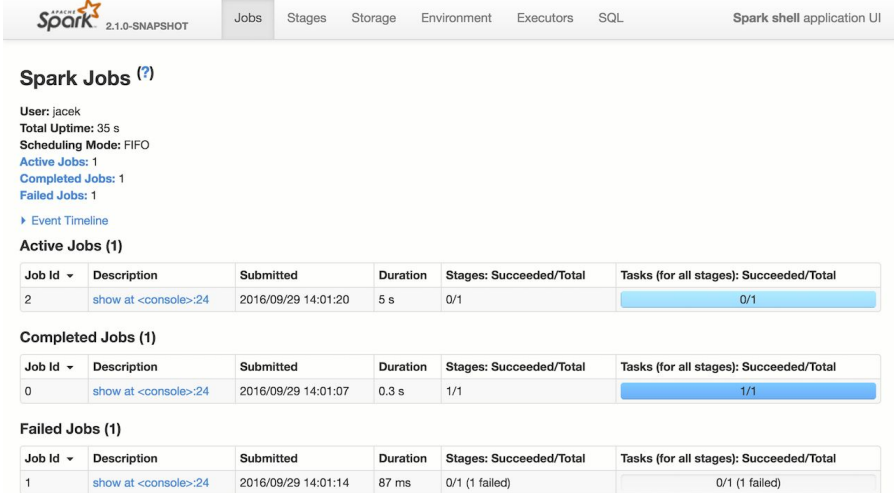
Spark Jobs, Stages and Tasks

- 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.



Spark 2.1.0-SNAPSHOT

Jobs Stages Storage Environment Executors SQL Spark shell application UI

Spark Jobs (?)

User: jacek
Total Uptime: 35 s
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 1
Failed Jobs: 1
[Event Timeline](#)

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	show at <console>:24	2016/09/29 14:01:20	5 s	0/1	0/1

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	show at <console>:24	2016/09/29 14:01:07	0.3 s	1/1	1/1

Failed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	show at <console>:24	2016/09/29 14:01:14	87 ms	0/1 (1 failed)	0/1 (1 failed)

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

▼ Active Jobs (1)

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
7	count at <console>:26 count at <console>:26	2019/08/10 17:50:13 (kill)	17 s	0/2	0/5 (4 running)

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

▼ Completed Jobs (7)

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
6	show at <console>:26 show at <console>:26	2019/08/10 17:49:30	0.4 s	1/1	1/1
5	show at <console>:28 show at <console>:28	2019/08/10 17:48:32	0.8 s	3/3	9/9
4	show at <console>:28 show at <console>:28	2019/08/10 17:47:40	2 s	3/3	9/9

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

Details for Job 7

Status: SUCCEEDED

Associated SQL Query: 8

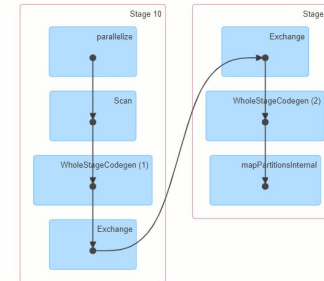
Completed Stages: 2

▼ Event Timeline

Enable zooming



▼ DAG Visualization



▼ Completed Stages (2)

Page: 1

1 Pages. Jump to 1

Show 100 items in a page

Go

Stage Id ▼	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
12	count at <console>-26 +details	2019/08/10 17:50:44	56 ms	1/1			236.0 B	
11	count at <console>-26 +details	2019/08/10 17:50:13	31 s	4/4				236.0 B

Page: 1

1 Pages. Jump to 1

Show 100 items in a page

Go

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

Stages for All Jobs

Active Stages: 1
Pending Stages: 1
Completed Stages: 13

▼ Fair Scheduler Pools (3)

Pool Name	Minimum Share	Pool Weight	Active Stages	Running Tasks	SchedulingMode
production	2	1	1	3	FAIR
test	3	2	0	0	FIFO
default	0	1	0	0	FIFO

▼ Active Stages (1)

Page: 1					1 Pages. Jump to 1 . Show 100 items in a page Go				
Stage Id ▼	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
13	production	show at <console>26	+details (vii) 2019/08/27 18:25:33	0.2 s	2019/08/27 18:25:33 0/4 (4 running)				
Page: 1					1 Pages. Jump to 1 . Show 100 items in a page Go				

▼ Pending Stages (1)

Page: 1		1 Pages. Jump to 1 . Show 100 items in a page Go							
Stage Id ▼	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
14	default	show at <console>26	+details Unknown	Unknown	0/200				
Page: 1		1 Pages. Jump to 1 . Show 100 items in a page Go							

▼ Completed Stages (13)

Page: 1

1 Pages. Jump to 1

Show 100

Items in a page

Go

Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
12	production	show at <console>26	+details 2019/08/27 18:25:33	0.3 s	4/4				1205.5 KIB
11	production	show at <console>28	+details 2019/08/27 18:24:36	0.3 s	1/1			12.6 KIB	
10	production	show at <console>28	+details 2019/08/27 18:24:35	0.3 s	4/4				1205.5 KIB
9	production	show at <console>28	+details 2019/08/27 18:24:35	0.3 s	4/4				1205.5 KIB
8	default	foreach at <console>27	+details 2019/08/27 18:01:04	93 ms	4/4				
7	default	foreach at <console>27	+details 2019/08/27 17:59:31	0.5 s	4/4				
6	production	show at <console>28	+details 2019/08/27 17:56:10	64 ms	1/1			12.6 KIB	
5	production	show at <console>28	+details 2019/08/27 17:56:10	0.2 s	4/4				1205.5 KIB

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 Stage 15 (Attempt 0)

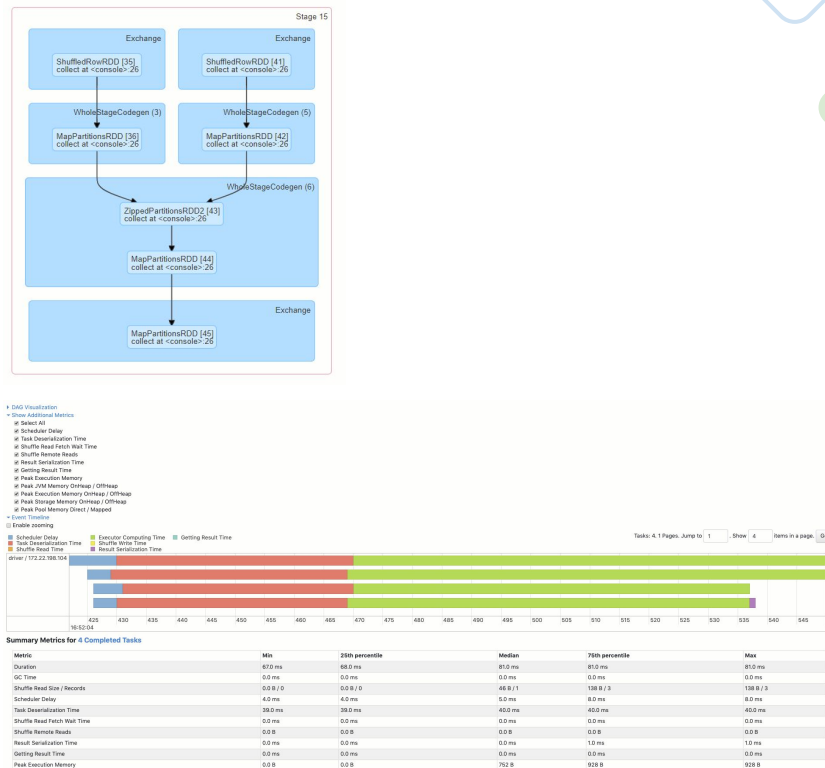
Total Time Across All Tasks: 1.4 min

Locality Level Summary: Node local: 172; Process local: 28

Shuffle Read Size / Records: 487.1 MiB / 44125788

Shuffle Write Size / Records: 86.2 KiB / 924

Associated Job Ids: 13



The slide features several decorative geometric elements: a light green double-lined chevron pointing down at the top center; a blue triangle pointing up at the bottom left; and a blue double-lined chevron pointing left on the right side, with a thin green line nested inside it.

06 | Improving Performance

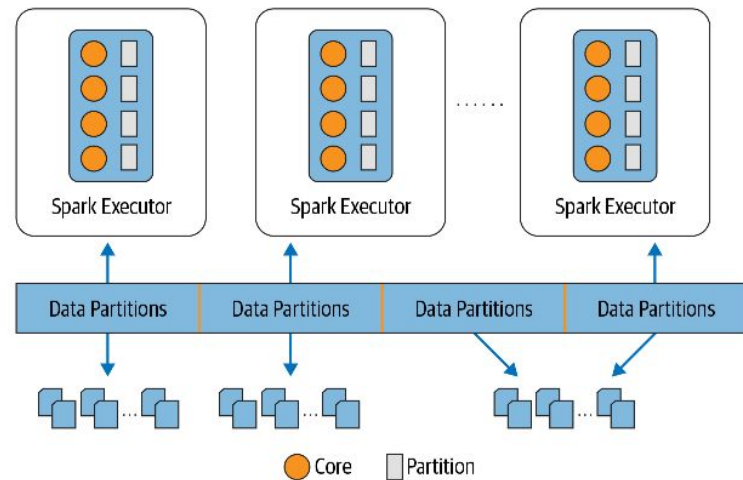
Pendahuluan

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

Storage Level: Memory Deserialized 1x Replicated
Cached Partitions: 12
Total Partitions: 12
Memory Size: 86.2 MiB
Disk Size: 0.0 B

Data Distribution on 1 Executors

Host	On Heap Memory Usage	Off Heap Memory Usage	Disk Usage
10.0.1.5:60233	86.2 MiB (280.1 MiB Remaining)	0.0 B (0.0 B Remaining)	0.0 B

12 Partitions

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page.

Block Name	Storage Level	Size in Memory	Size on Disk	Executors
rdd_3_0	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_1	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_10	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_11	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_2	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_3	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_4	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_5	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_6	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_7	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_8	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233
rdd_3_9	Memory Deserialized 1x Replicated	7.2 MiB	0.0 B	10.0.1.5:60233

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

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