Université Constantine 2
جامعة قسنطينة 2

Big Data & NoSQL (BDNS)

– Chapitre–5

Spark

Dr. GHEMMAZ W
Faculté NTIC
Wafa.ghemmaz@univ-constantine2.dz

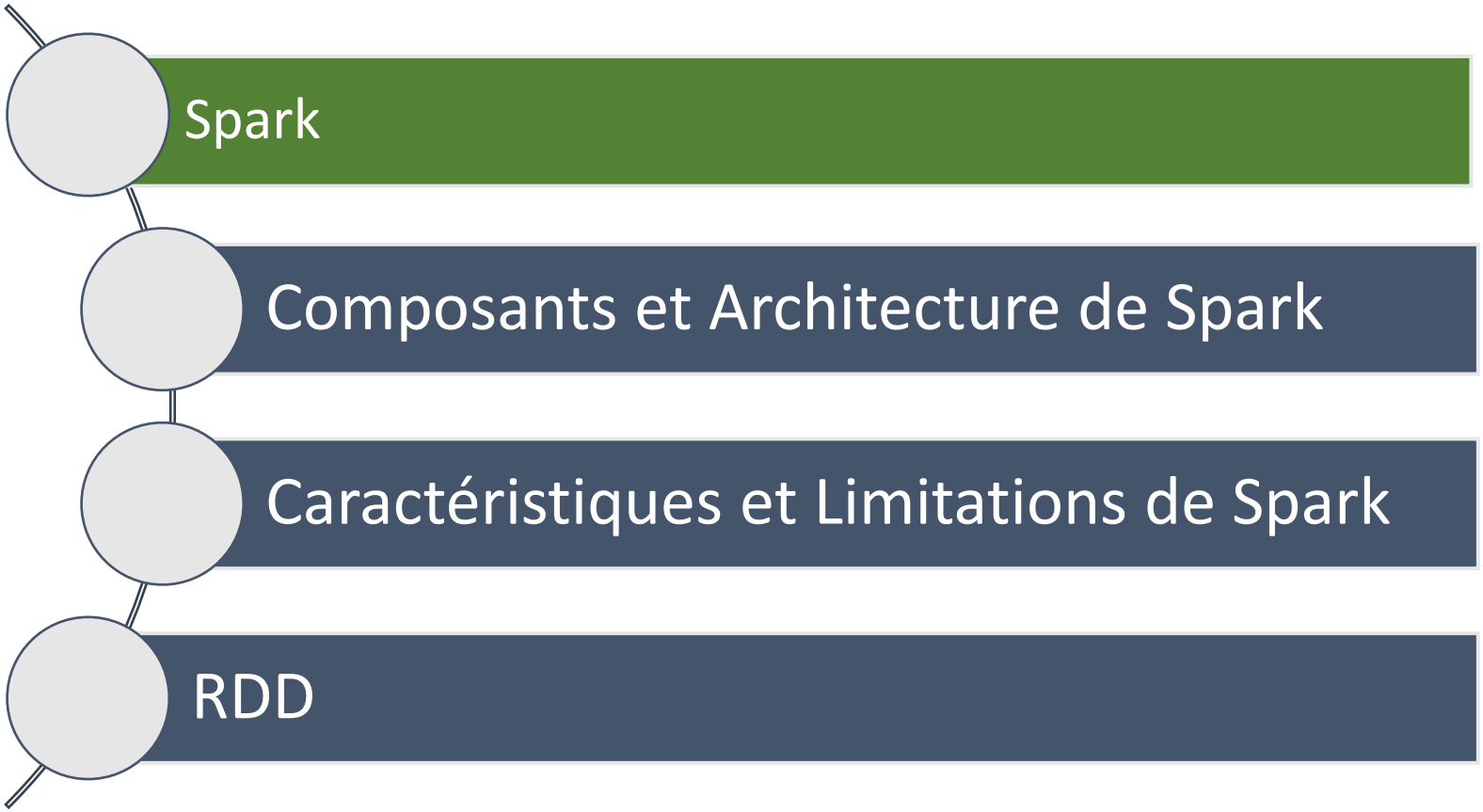
Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	TLSI	Master 1	SDSI

Plan


Spark
Composants et Architecture de Spark
Caractéristiques et Limitations de Spark
RDD

Plan


Spark
Composants et Architecture de Spark
Caractéristiques et Limitations de Spark
RDD

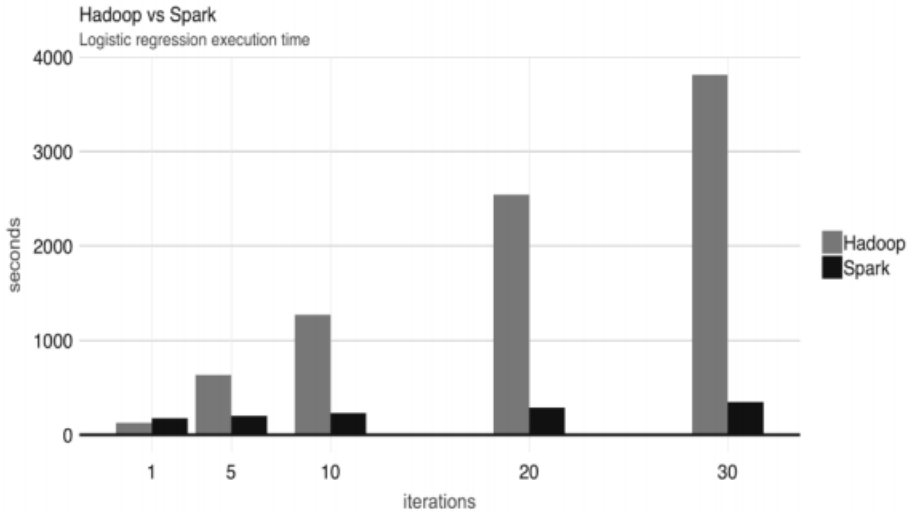
Spark

- Apache Spark est une plateforme de traitement sur cluster générique
- Apache Spark assure un traitement parallèle et distribué sur des données massives
- Apache Spark permet de réaliser des traitements par lot (batch processing) ou à la volée (stream processing).
- Apache Spark est conçu de façon à pouvoir intégrer tous les outils et technologies Big Data
- Apache Spark Supporte les taches itératives et interactives (grâce aux RDDs)
- Apache Spark offre des APIs de haut niveau en Java, Scala, Python et R

Spark

Spark

- Spark est **10x** plus rapide que Hadoop (données sur disque).
- Spark est **100x** plus rapide que Hadoop (données sur mémoire).

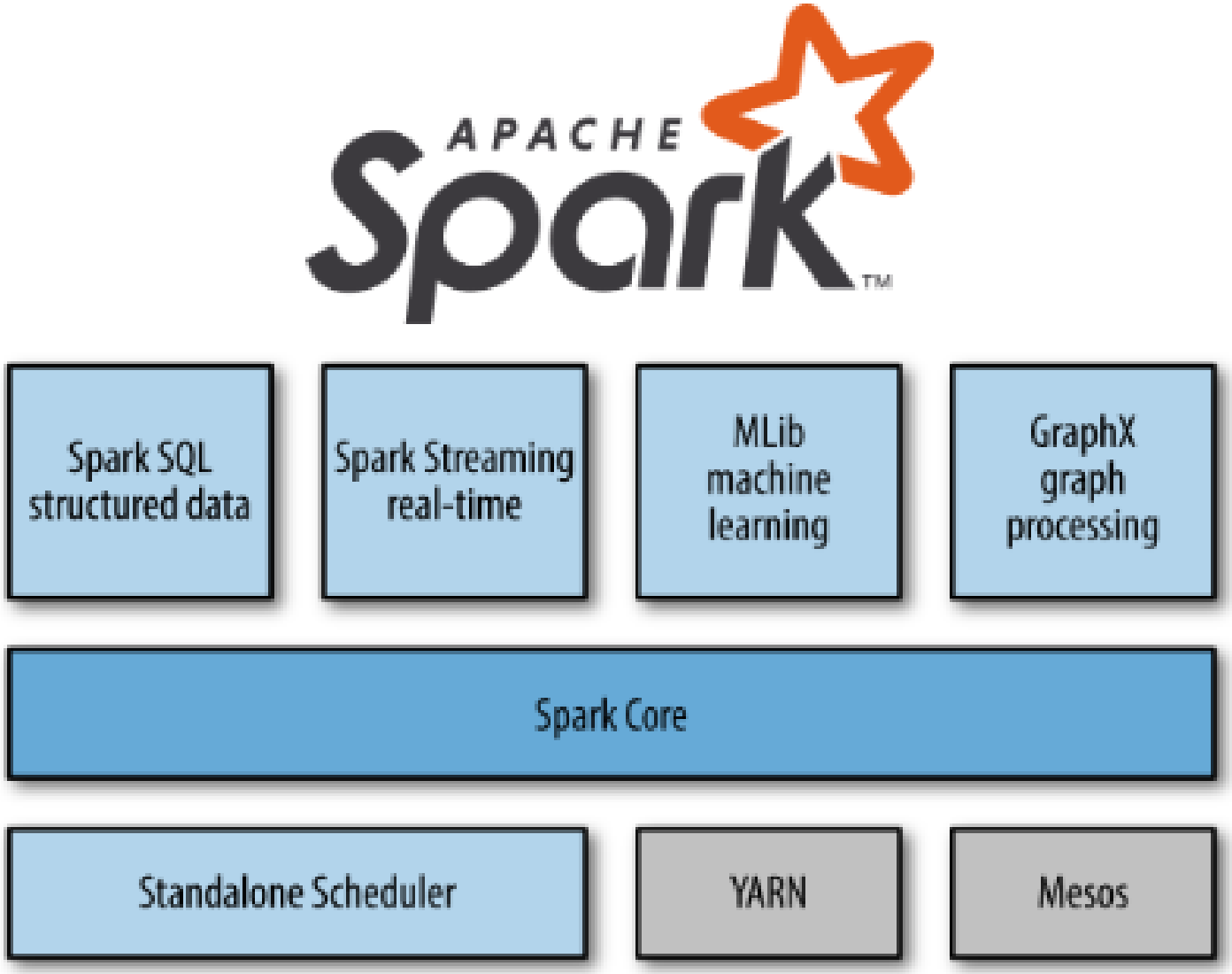


- Spark écrit les données (intermédiaires) sur la RAM et non pas sur le disque.

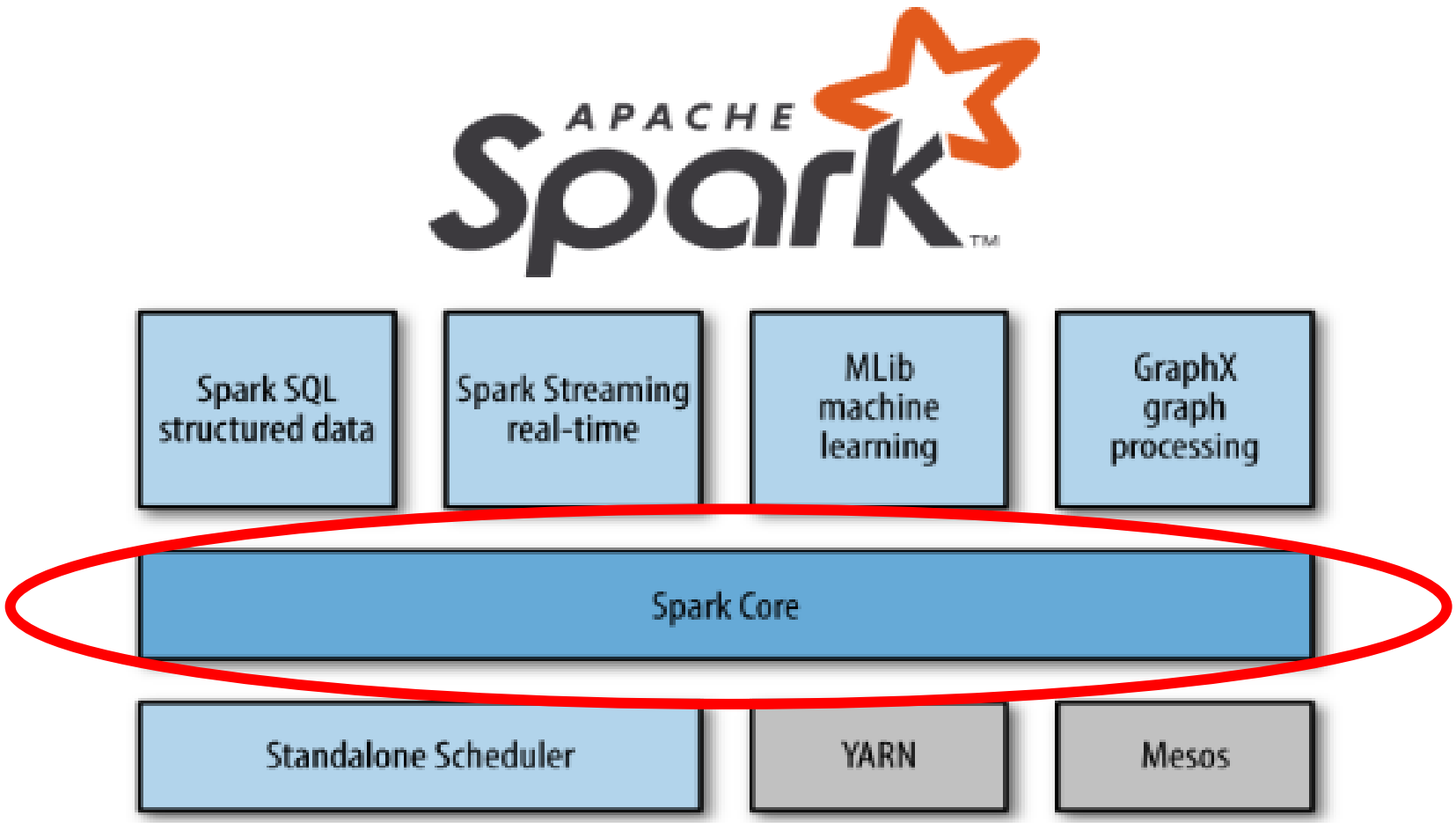
Plan


Spark
Composants et Architecture de Spark
Caractéristiques et Limitations de Spark
RDD

Composants de Spark

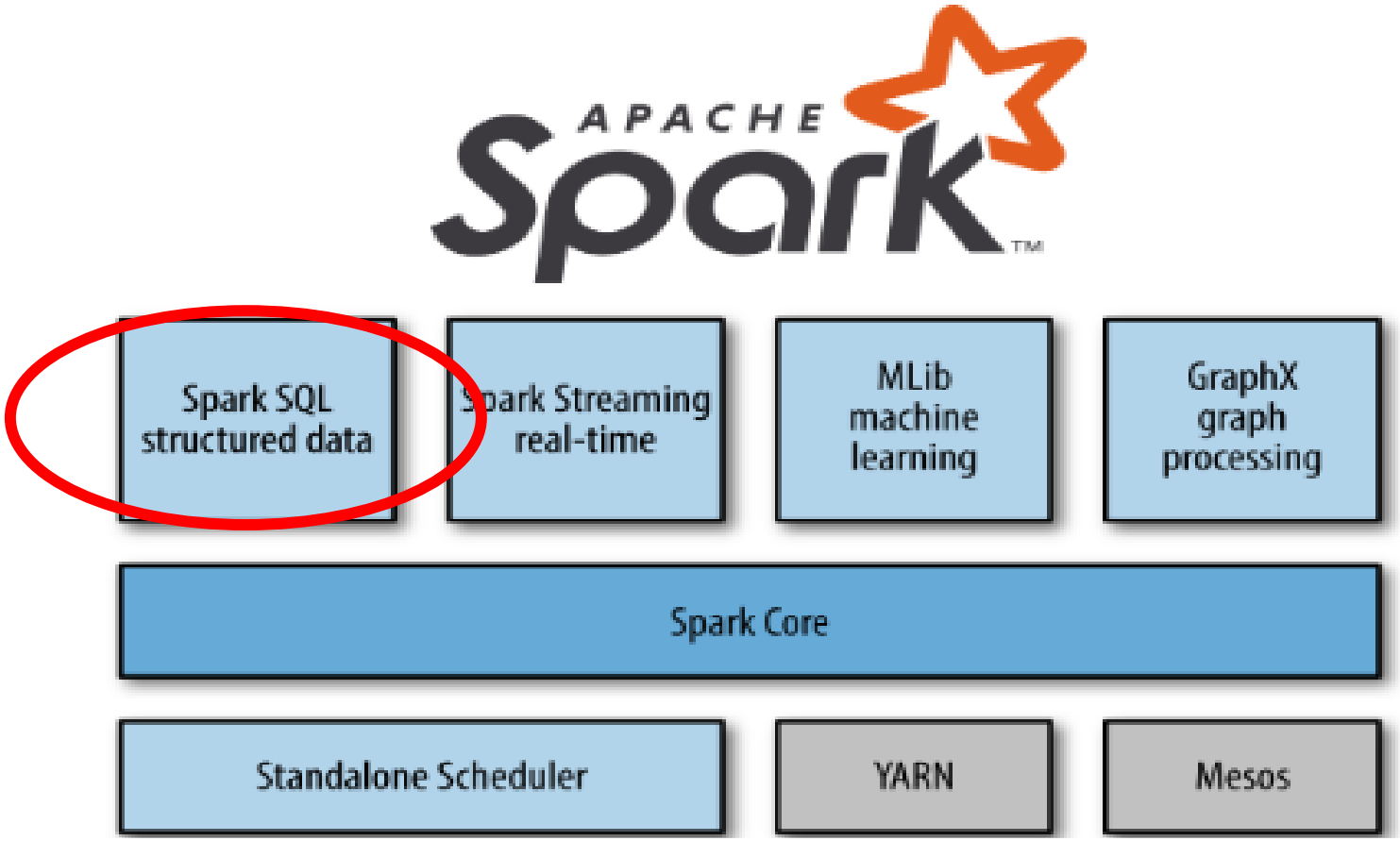


Composants de Spark



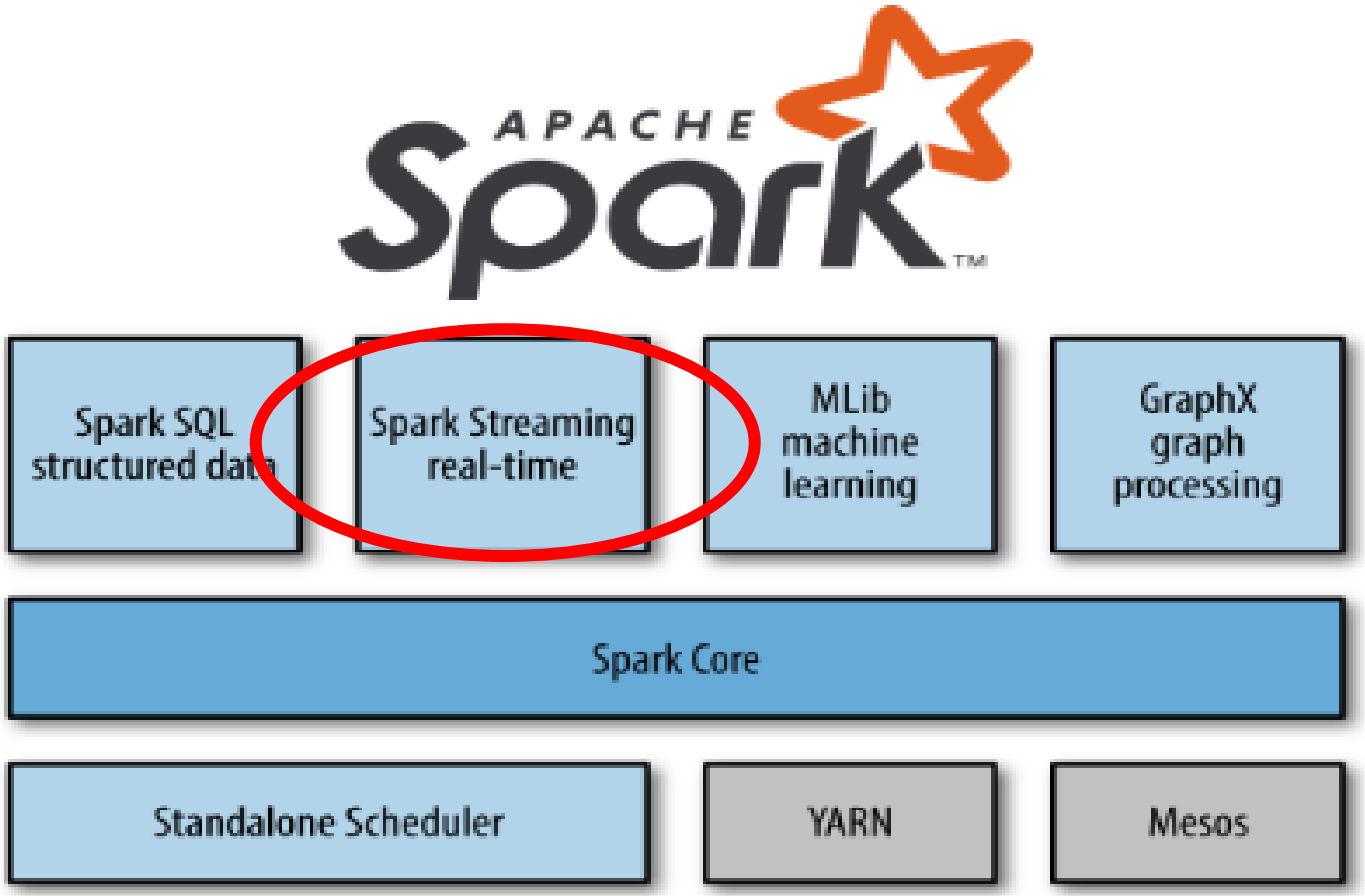
1. **Spark Core** Spark Core est le point central de Spark, qui fournit une plateforme d'exécution pour toutes les applications Spark. De plus, il supporte un large éventail d'applications.

Composants de Spark



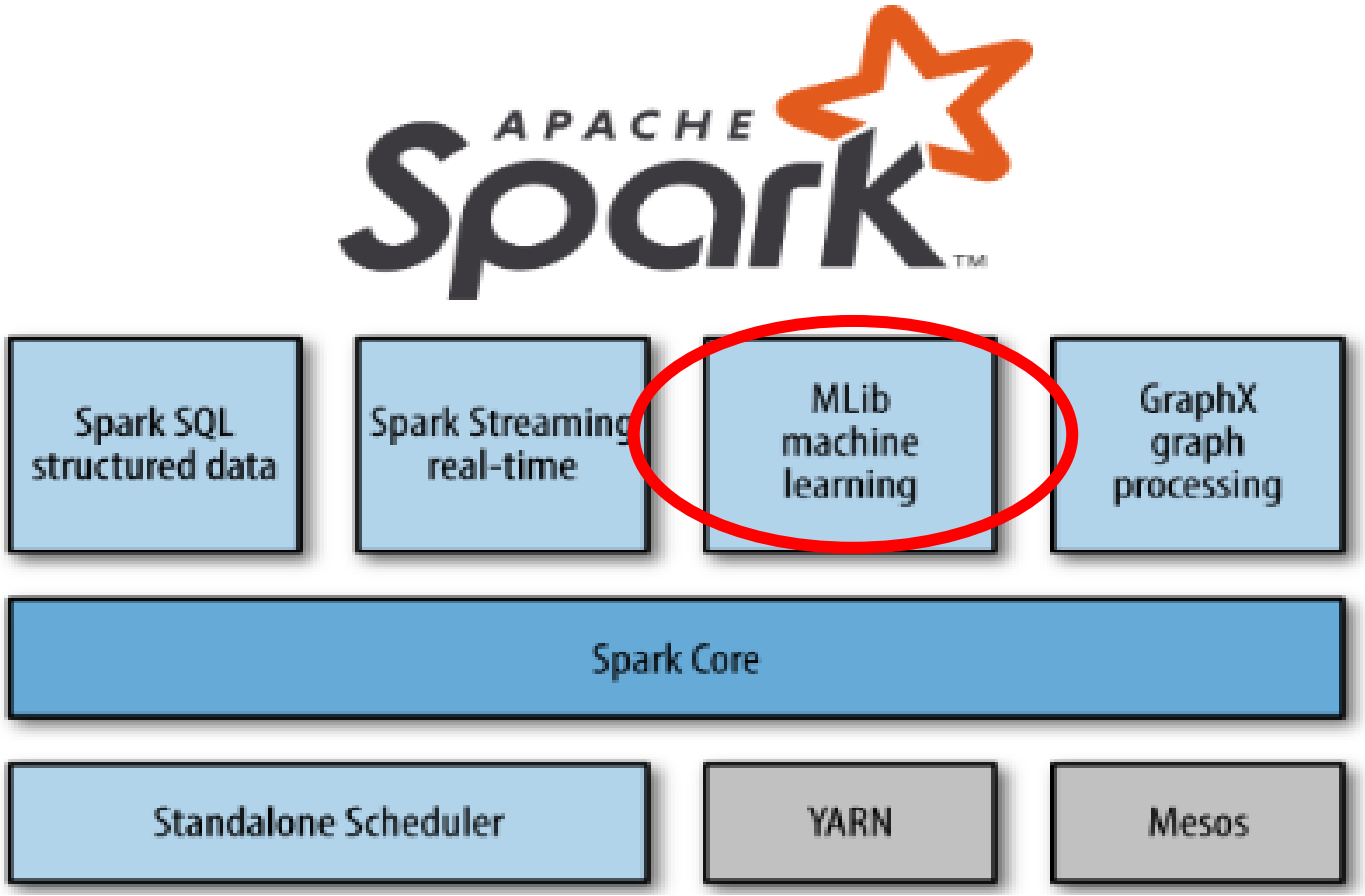
2. **Spark SQL** Spark SQL se situe au dessus de Spark, pour permettre aux utilisateurs d'utiliser des requêtes SQL/HQL. Les données structurées et semi-structurées peuvent ainsi être traitées grâce à Spark SQL, avec une performance améliorée.

Composants de Spark



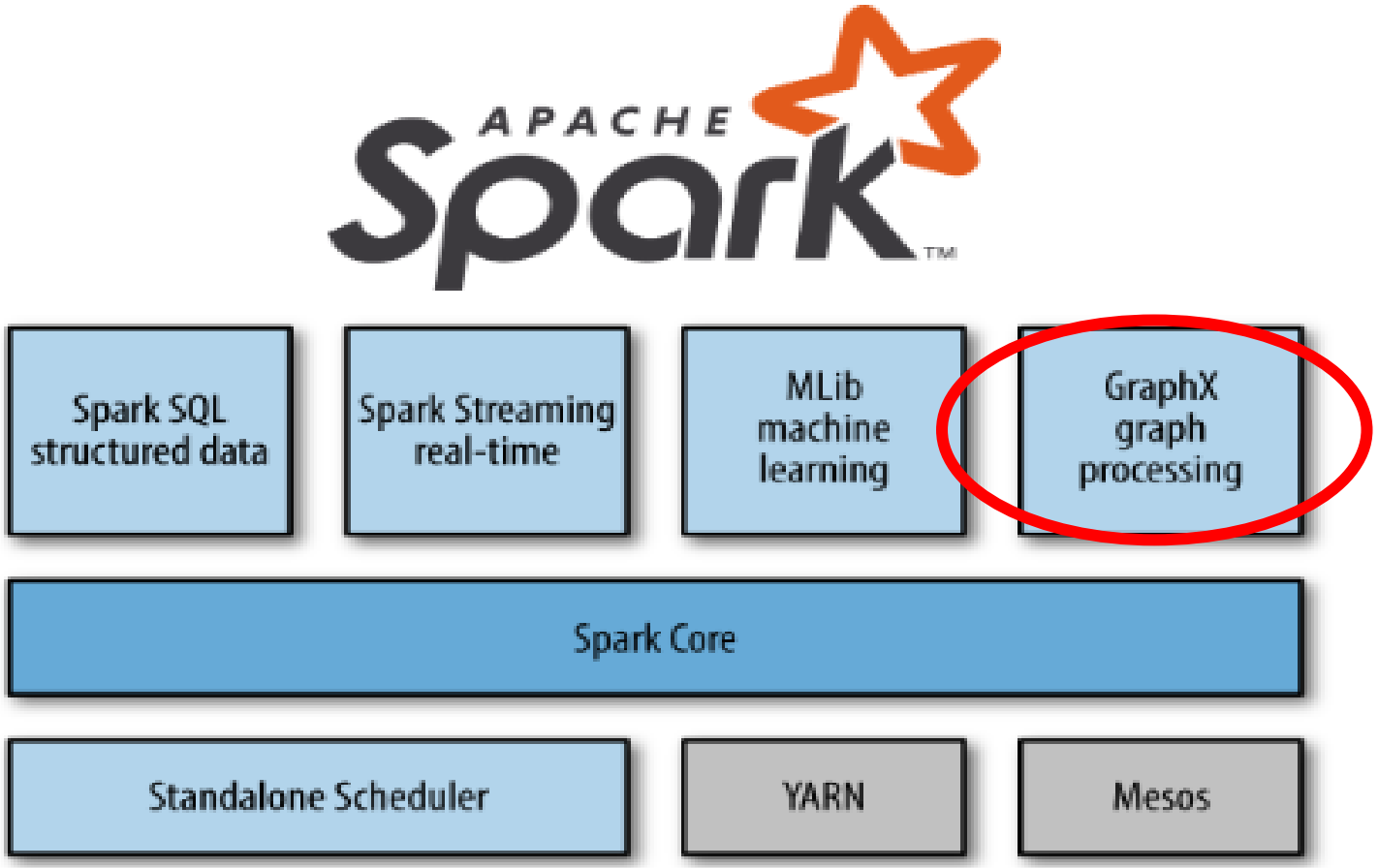
3. **Spark Streaming** Spark Streaming permet de créer des applications d'analyse de données interactives. Les flux de données sont transformés en micro-lots et traités par dessus Spark Core.

Composants de Spark



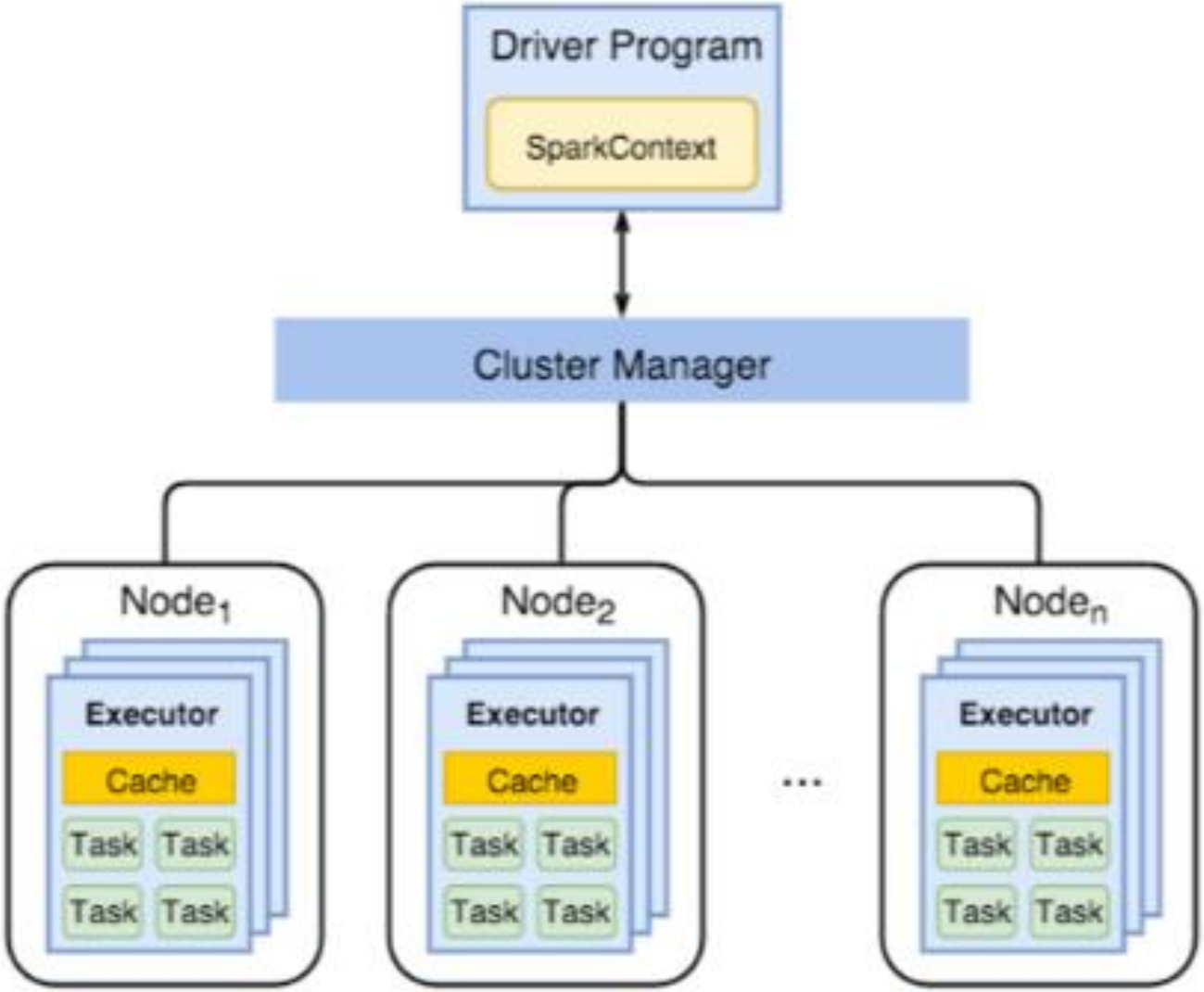
4. **Spark MLlib** La bibliothèque de machine learning MLlib fournit des algorithmes de haute qualité pour l'apprentissage automatique. Ce sont des libraries riches, très utiles pour les data scientists, autorisant de plus des traitements en mémoire améliorant de façon drastique la performance de ces algorithmes sur des données massives.

Composants de Spark



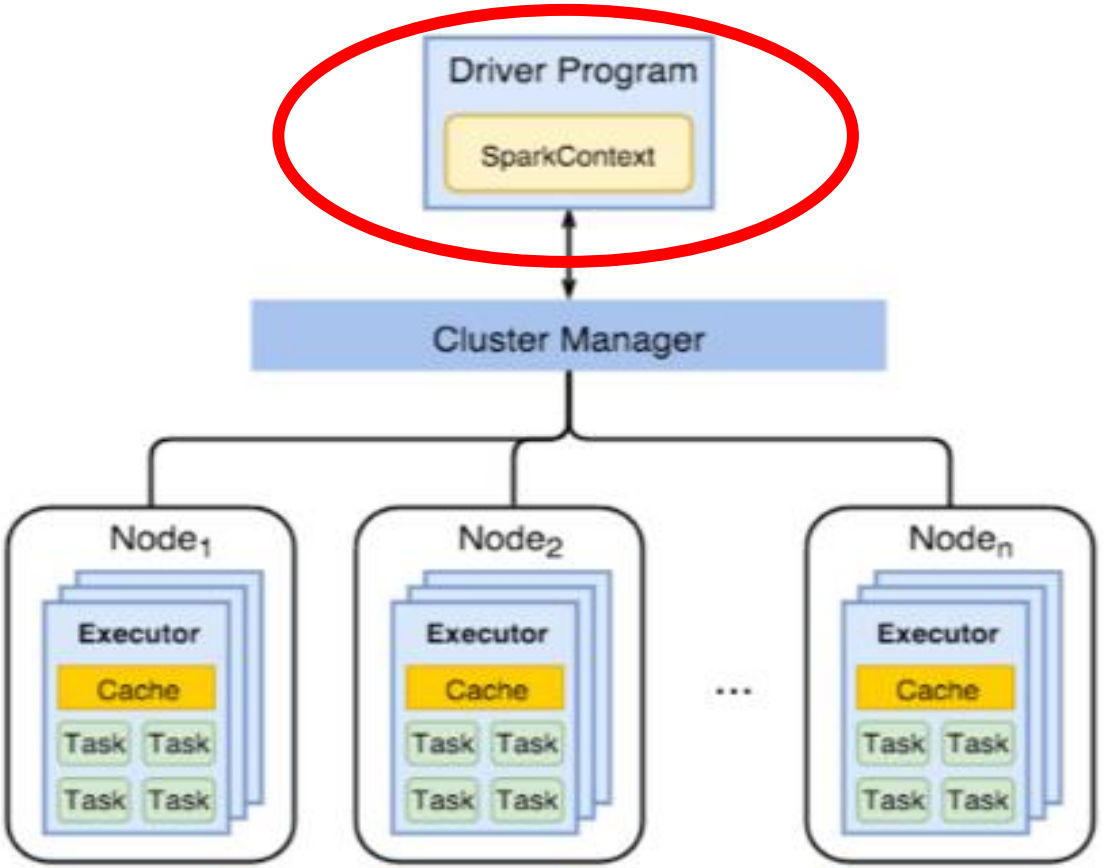
5. **Spark GraphX** Spark Graphx est le moteur d'exécution permettant un traitement scalable utilisant les graphes, se basant sur Spark Core.

Architecture Spark



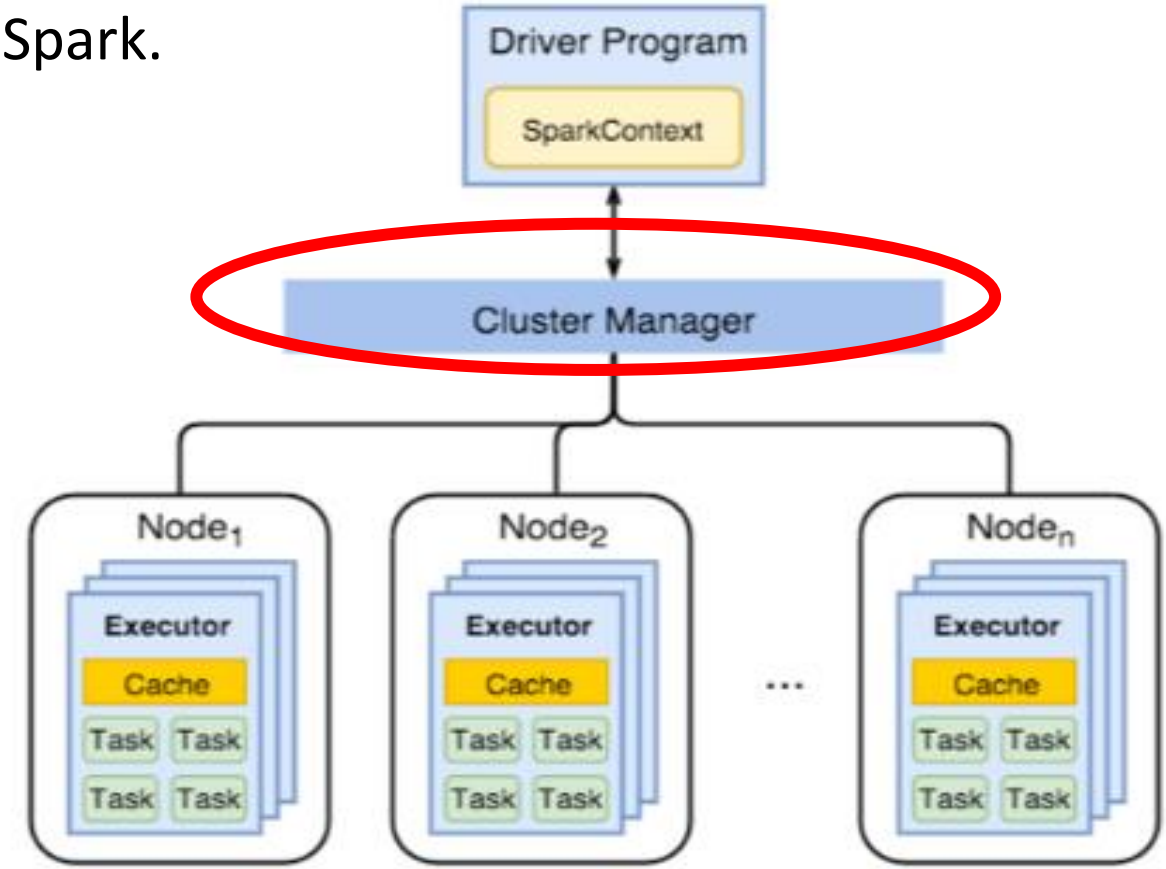
Architecture Spark

Les applications Spark s'exécutent comme un ensemble indépendant de processus sur un cluster, coordonnés par un objet SparkContext dans le programme principal, appelé driver program.



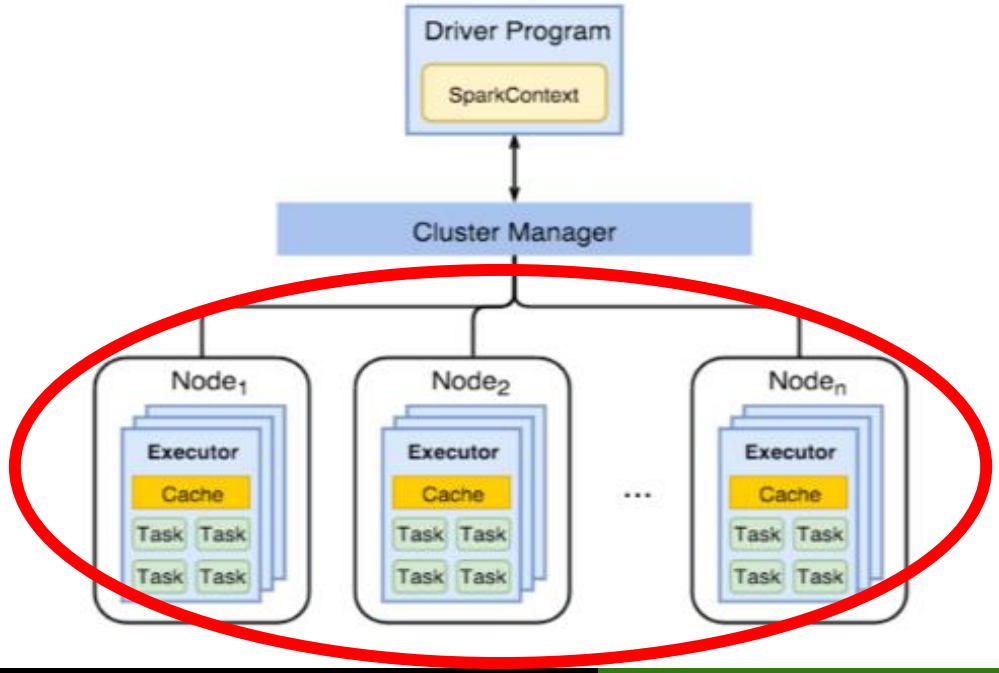
Architecture Spark

Pour s'exécuter sur un cluster, SparkContext peut se connecter à plusieurs types de gestionnaires de clusters (Cluster Managers): Standalone schedule, YARN, Mesos, Les gestionnaires permettent d'allouer les ressources nécessaires pour l'exécution de plusieurs applications Spark.



Architecture Spark

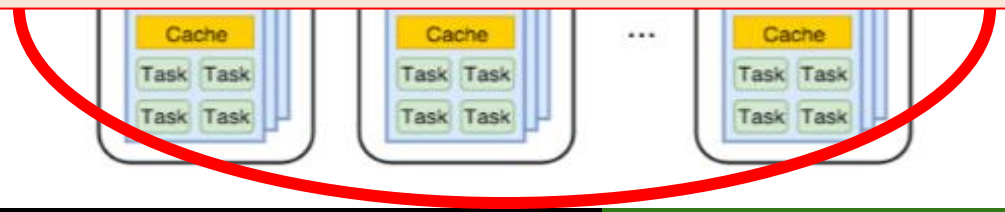
Une fois connecté, Spark lance des exécuteurs sur les nœuds du cluster, qui sont des processus qui lancent des traitements et stockent des données pour les applications. Il envoie ensuite le code de l'application (dans un fichier JAR ou Python) aux exécuteurs. Spark Context envoie finalement les tâches à exécuter aux exécuteurs.



Architecture Spark

Une fois connecté, Spark lance des exécuteurs sur les nœuds du cluster, qui sont des processus qui lancent des traitements et stockent

- **Chaque application a son lot d'exécuteurs, qui restent actifs tout au long de l'exécution de l'application.**
- **les applications sont isolées les unes des autres, du point de vue de l'orchestration (chaque driver exécute ses propres tâches), et des exécuteurs (les tâches des différentes applications tournent sur des JVM différentes).**
- **L'application principale (driver) doit être à l'écoute des connexions entrantes venant de ses exécuteurs**



Plan


Spark
Composants et Architecture de Spark
Caractéristiques et Limitations de Spark
RDD

Caractéristiques de Spark

- ✓ Performance de traitement.
- ✓ Dynamicité.
- ✓ Tolérance aux Fautes.
- ✓ Traitements à la volée.
- ✓ Évaluations Paresseuses (*Lazy Evaluations*)
- ✓ Support de plusieurs langages.
- ✓ Une communauté active et en expansion
- ✓ Support d'analyses sophistiquées
- ✓ Intégration avec Hadoop.

Limitations de Spark

- ✓ Pas de support pour le traitement en temps réel.
- ✓ Problèmes avec les fichiers de petite taille.
- ✓ Pas de système de gestion des fichiers.
- ✓ Coûteux.
- ✓ Nombre d'algorithmes limité
- ✓ Latence: traitement en streaming

Plan


Spark
Composants et Architecture de Spark
Caractéristiques et Limitations de Spark
RDD

RDD de Spark

RDD: Resilient Distributed Dataset

RDD : est une collection d'éléments tolérante aux fautes qui peut être gérée en parallèle. Les RDDs utilisent la mémoire et l'espace disque selon les besoins.

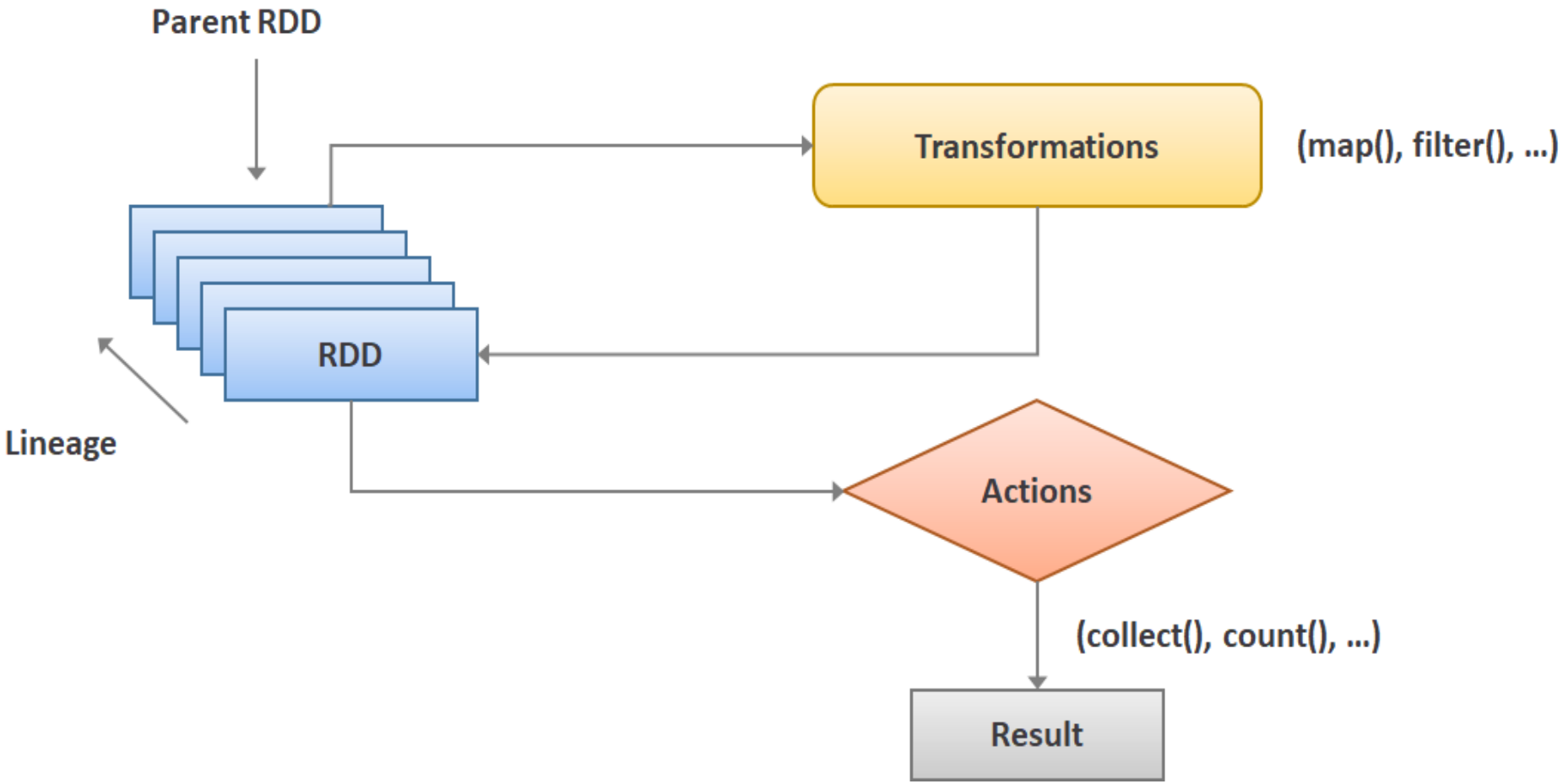
- **R**esilient: capable de récupérer rapidement en cas de problèmes ou de conditions difficiles,
- **D**istributed: partage les données sur les différents nœuds participants pour une exécution parallèle,
- **D**ataset: une collection de données composée d'éléments séparés mais qui sont manipulés comme une unité compacte

RDD: Resilient Distributed Dataset

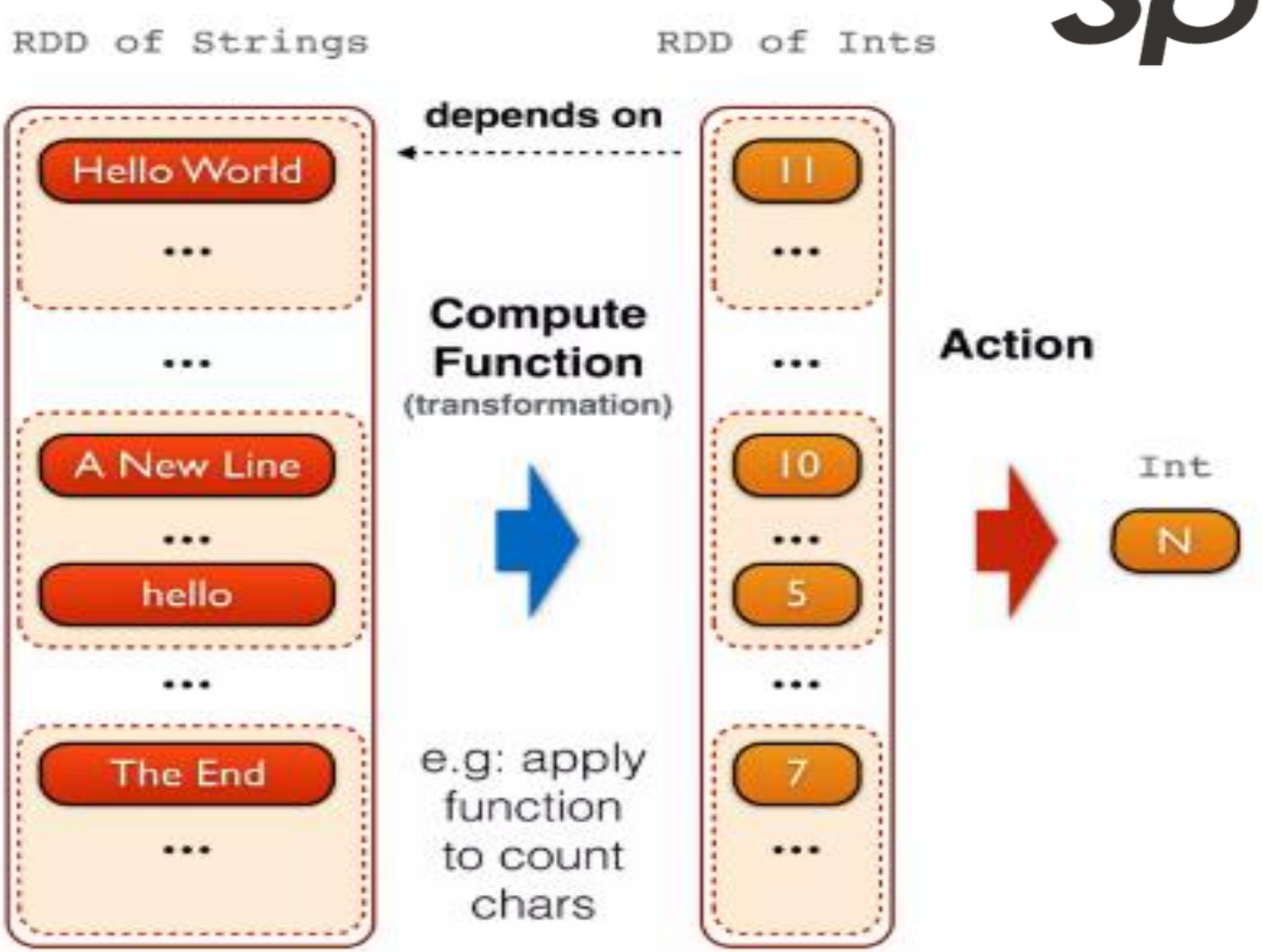
Les RDDs ont hérité les caractéristiques des collections (tableaux, listes, tuples) de scala:

- **Calcul paresseux (Lazy computation):** les opérations (transformations) exécutées sur les RDDs sont paresseuses c.à.d. ne sont pas exécutées telles qu'elles apparaissent au driver mais lorsqu'une action est effectuée.
- **Immuabilité (Immutability) :** en lecture seule (ne pas les modifier après leur création).
- **En mémoire (In-memory):** les RDDs sont exécutés en mémoire centrale (soit en mémoire cache) évitant ainsi la réplication des données sur disque.
- **Tolérance aux pannes**

Les opérations sur RDD



Les opérations sur RDD



Les opérations sur RDD

API Spark

Scala

```scala
val spark = new SparkContext()

val lines    = spark.textFile("hdfs://docs/")    // RDD[String]
val nonEmpty = lines.filter(l => l.nonEmpty())   // RDD[String]

val count = nonEmpty.count
```

Java 8

```java
SparkContext spark = new SparkContext();

JavaRDD<String> lines    = spark.textFile("hdfs://docs/")
JavaRDD<String> nonEmpty = lines.filter(l -> l.length() > 0);

long count = nonEmpty.count();
```

Python

```python
spark = SparkContext()

lines = spark.textFile("hdfs://docs/")
nonEmpty = lines.filter(lambda line: len(line) > 0)

count = nonEmpty.count()
```

Les opérations sur RDD

Transformations	Actions
map(func)	take(N)
flatMap(func)	count()
filter(func)	collect()
groupByKey()	reduce(func)
reduceByKey(func)	takeOrdered(N)
mapValues(func)	top(N)
…	…

Les opérations sur RDD

Il existe deux moyens de créer les RDDs:

1. Paralléliser une collection existante en mémoire dans le programme Driver.
2. Le générer à partir d'un fichier enregistré sur un support de stockage externe.

Parallélisation de Collections

Les collections parallélisées sont créées en appelant la méthode parallelize du JavaSparkContext sur une collection existante dans votre programme Driver.
Les éléments de la collection sont copiés pour former une structure distribuée qui peut être traitée en parallèle

```
List<Integer> data = Arrays.asList(1, 2, 3, 4, 5);
JavaRDD<Integer> distData = sc.parallelize(data);
```

```
list = spark.sparkContext.parallelize([1, 2, 3, 4, 5])
```

Parallélisation de Collections

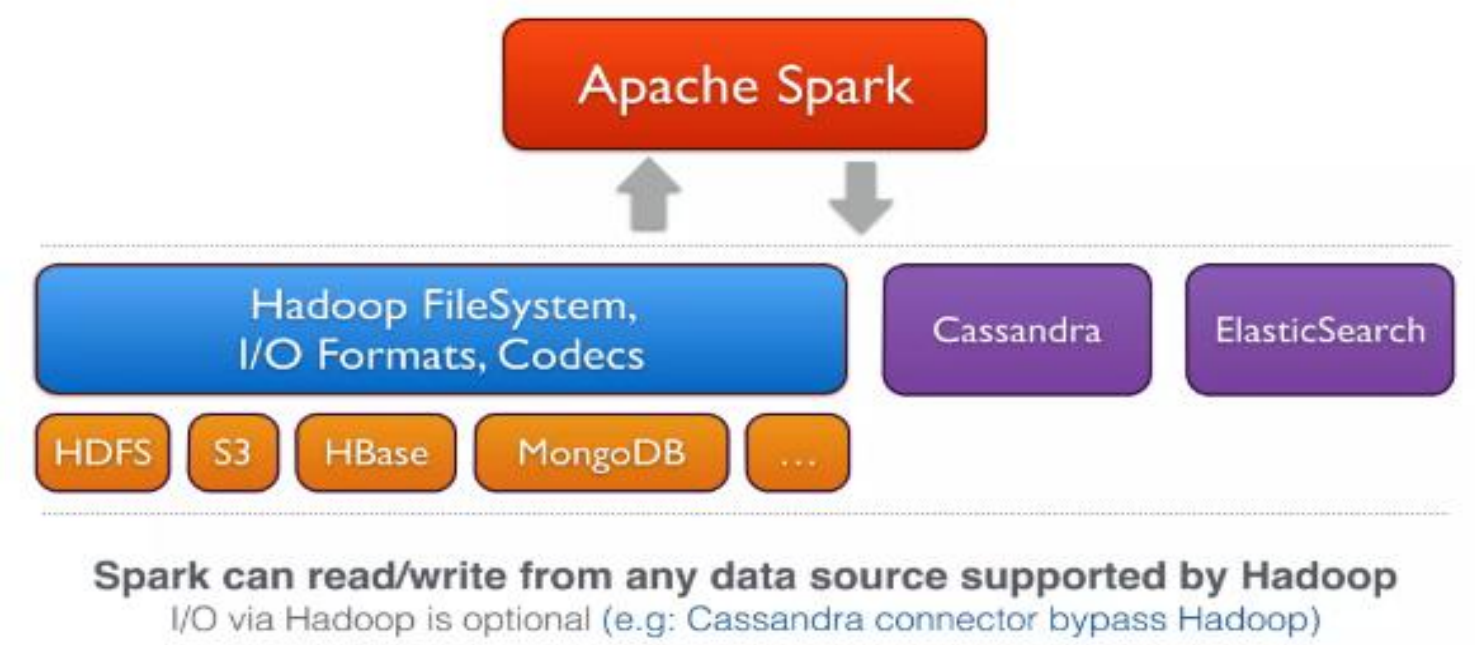
Un paramètre important à définir dans une collection parallélisée, c'est le nombre de partitions à utiliser pour diviser la collection. Spark exécute une tâche pour chaque partition du cluster. En temps normal, Spark essaiera de définir le nombre de partitions automatiquement selon votre cluster, cependant, il est possible de le définir manuellement en le passant comme second paramètre de la fonction parallelize:

```
sc.parallelize(data, 10)
```

Les opérations sur RDD

Génération à partir d'un fichier externe

Spark peut créer une collection distribuée à partir de n'importe quelle source de stockage supportée par Hadoop, incluant votre propre système de stockage, HDFS, Cassandra, HBase, Amazon S3, etc.



Génération à partir d'un fichier externe

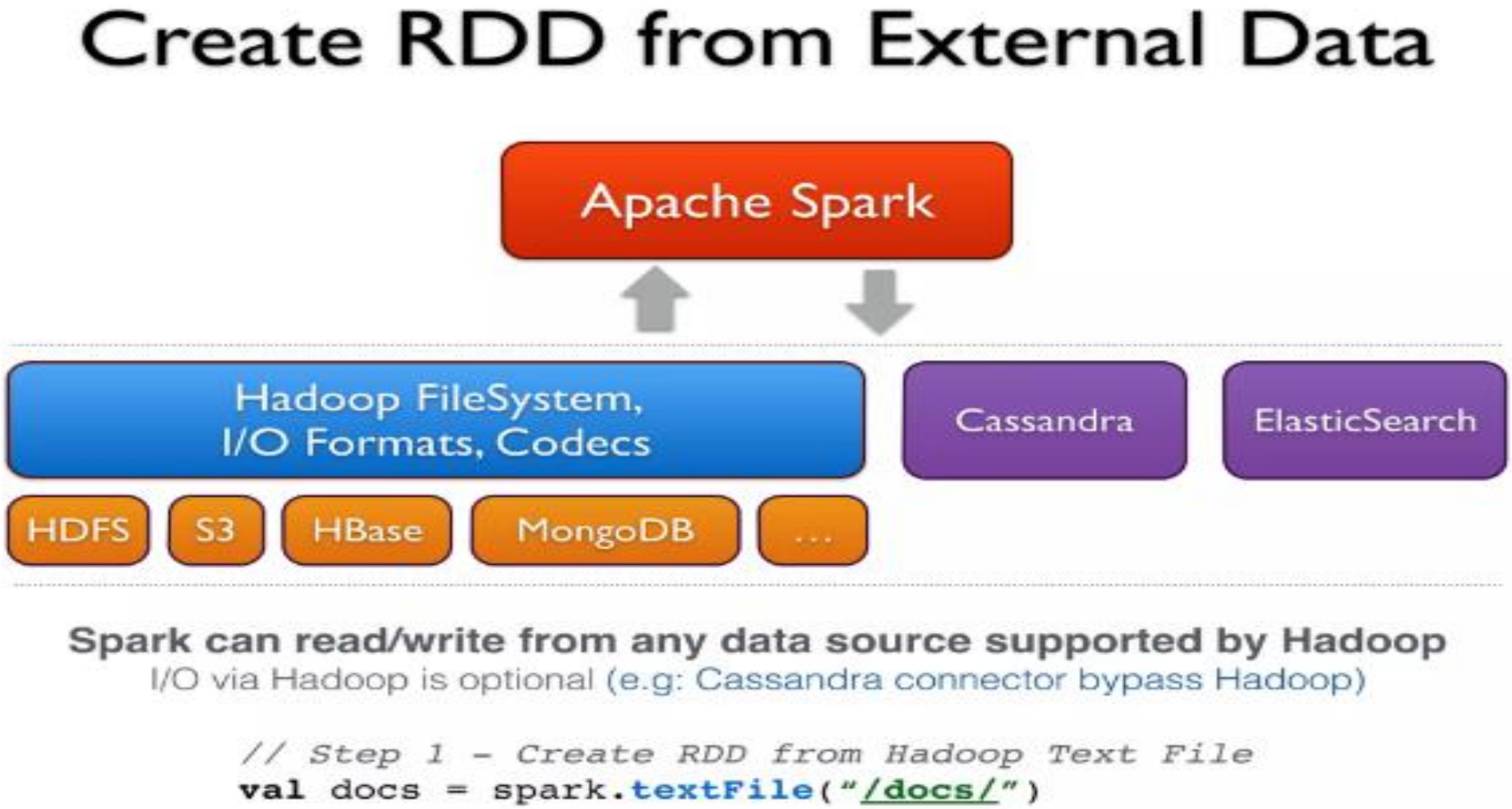
Il est possible de créer des RDDs à partir de fichiers texte en utilisant la méthode textfile du SparkContext. Cette méthode prend l'URI du fichier (chemin du fichier local, ou bien en utilisant hdfs:// ou s3://), et le lit comme une collection de lignes

```
JavaRDD<String> distFile = sc.textFile("data.txt");
```

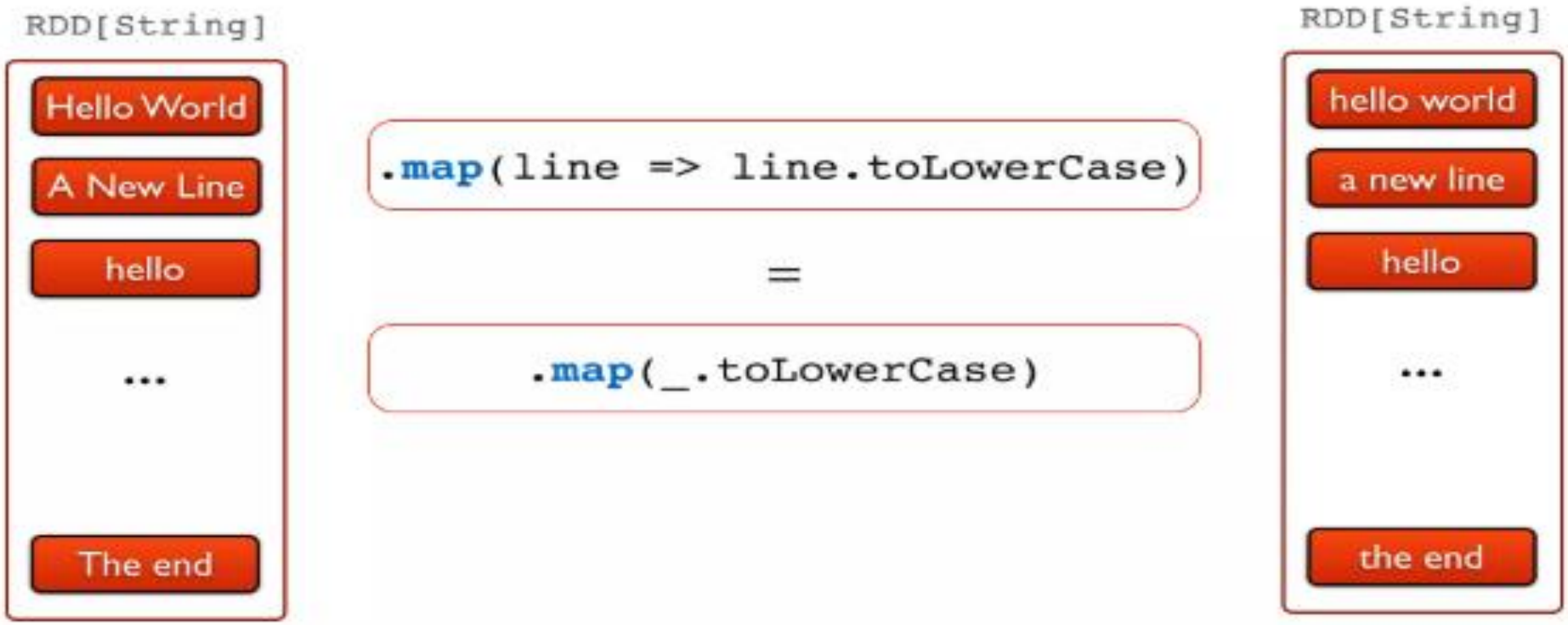
```
list = spark.sparkContext.textFile("input.txt")
```

Exemple: Traitement sur texte

Objectif: relever les mots les plus fréquents dans un fichier.

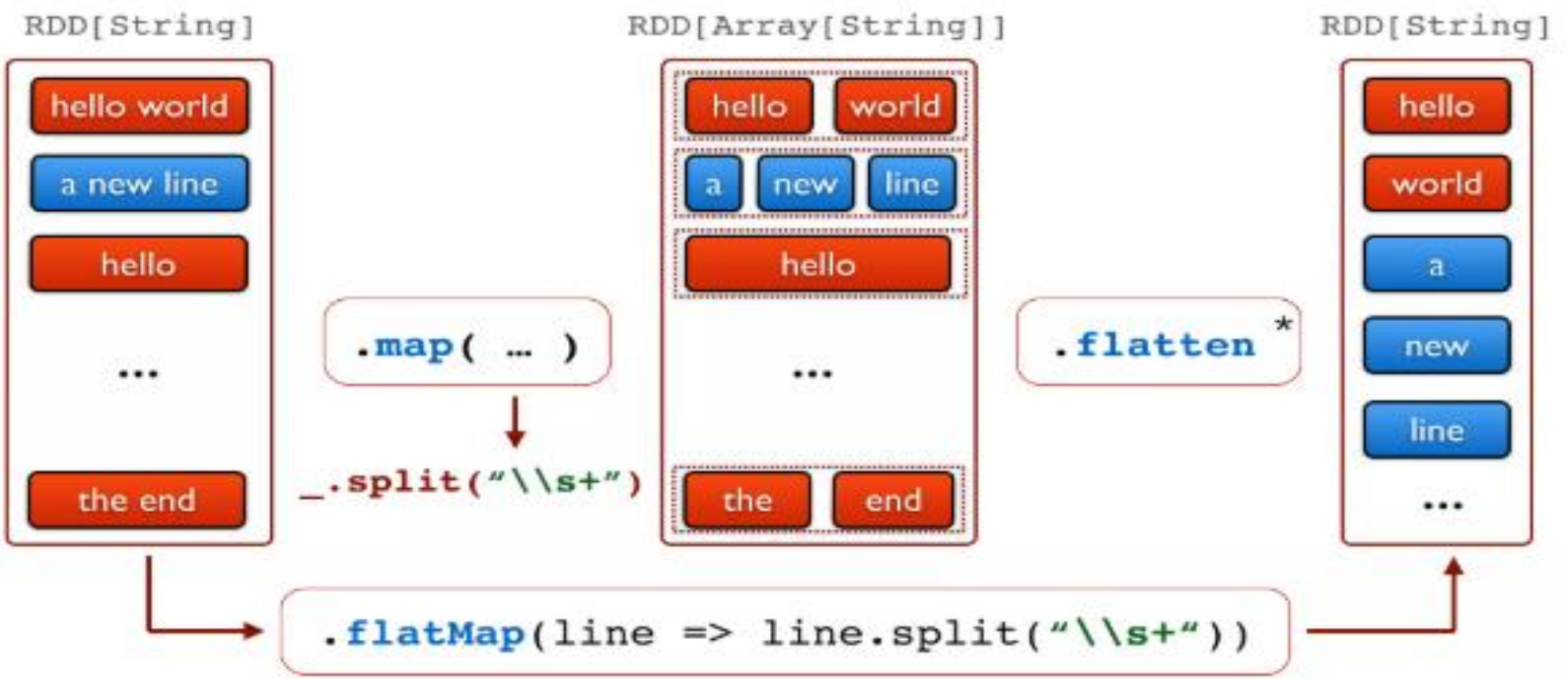


Exemple: Traitement sur texte




```
// Step 2 - Convert lines to lower case
val lower  = docs.map(line => line.toLowerCase)
```

Exemple: Traitement sur texte

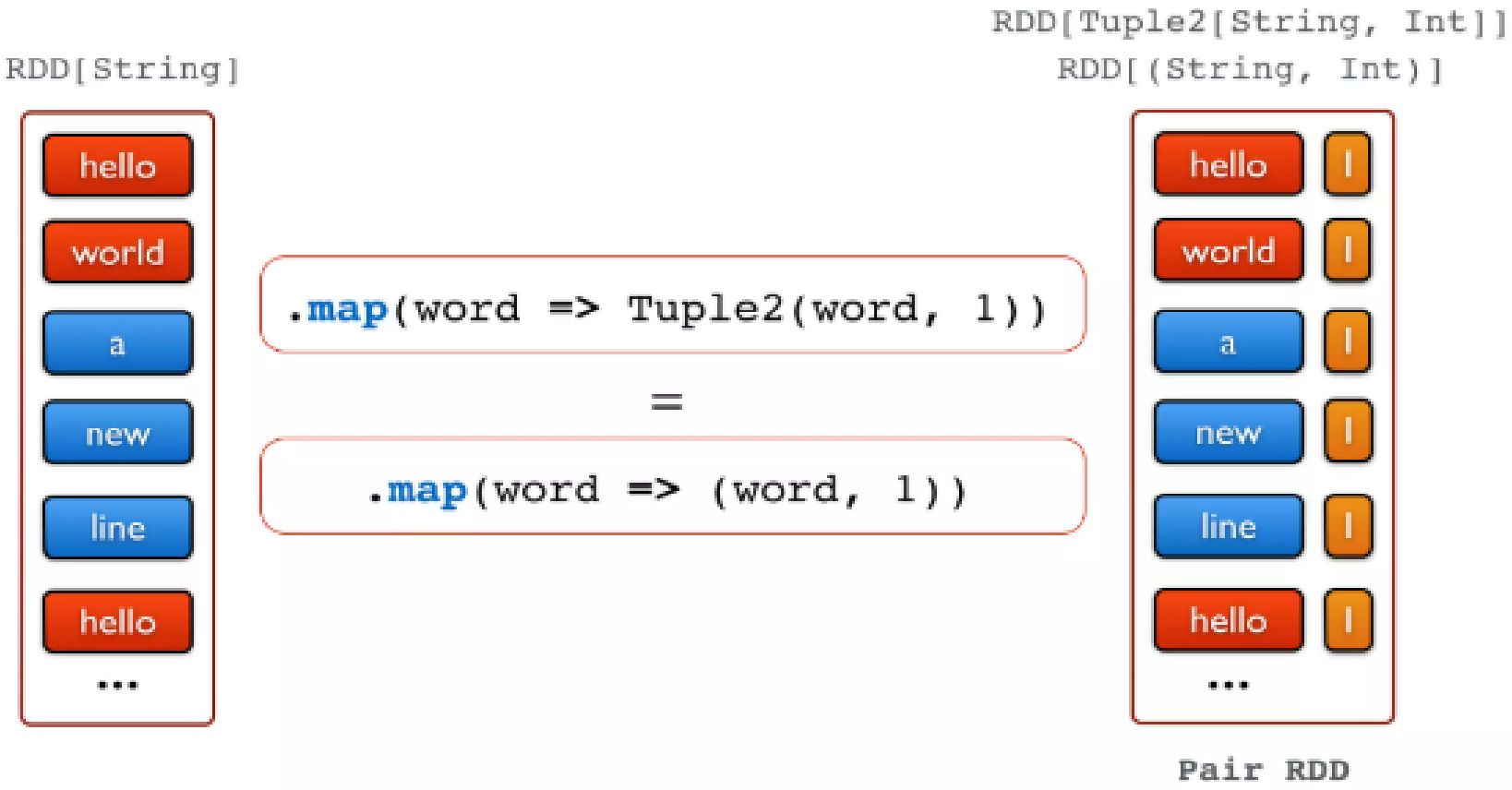



```
// Step 3 - Split lines into words
val words   = lower.flatMap(line => line.split("\\s+"))
```

Note: flatten() not available in spark, only flatMap

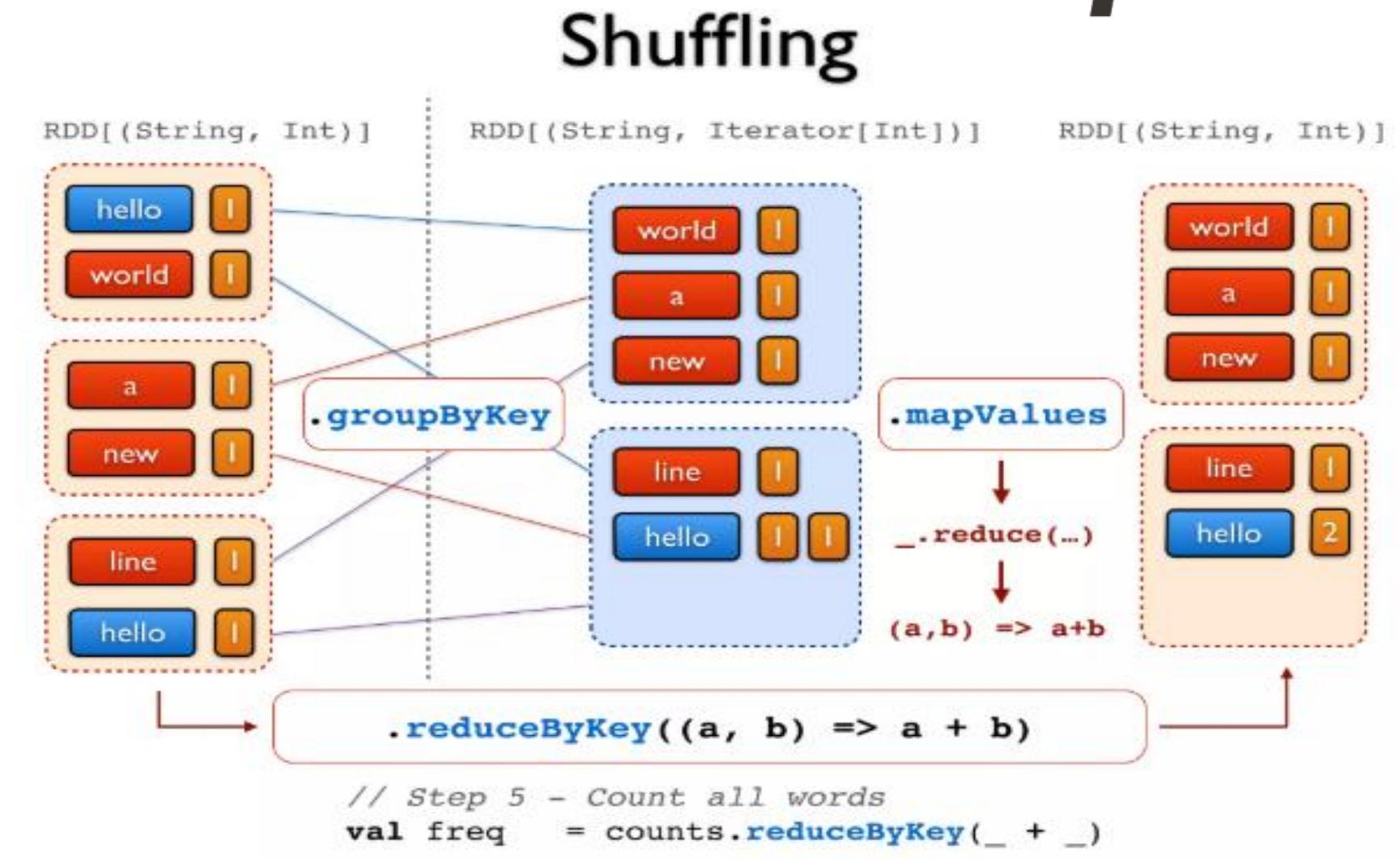
Exemple: Traitement sur texte

Key-Value Pairs



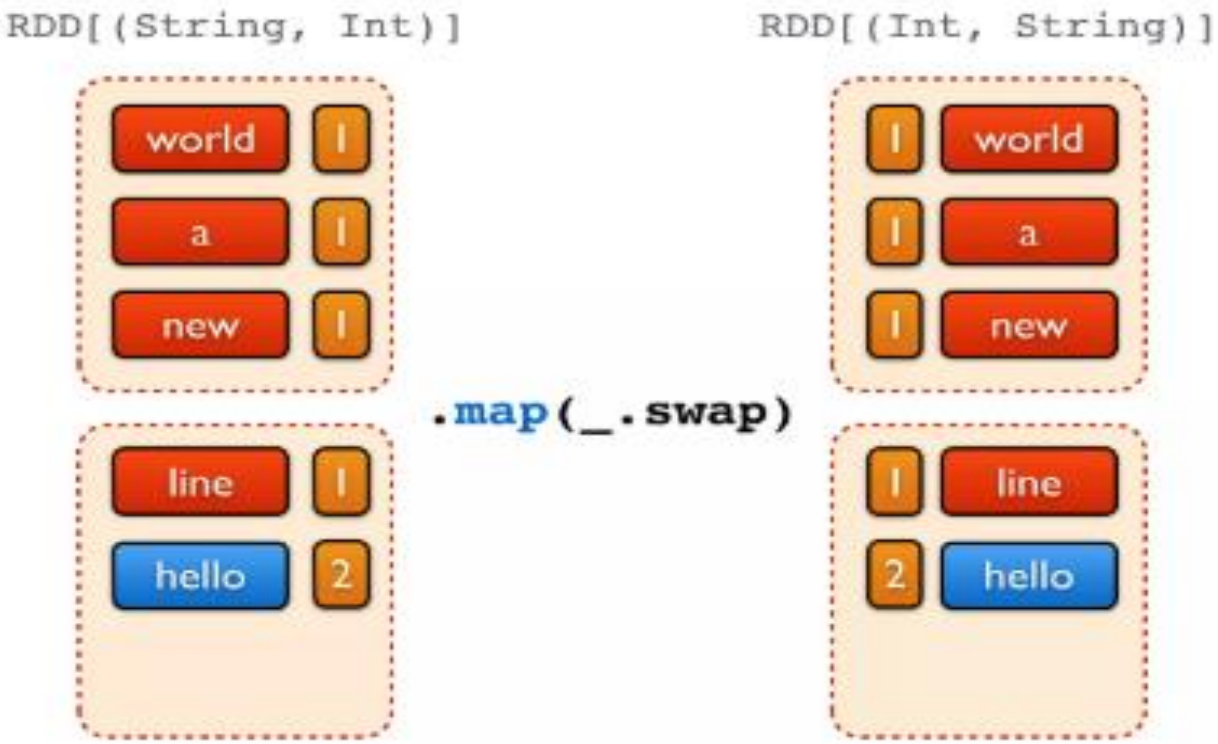

```
// Step 4 - Split lines into words
val counts = words.map(word => (word, 1))
```

Exemple: Traitement sur texte



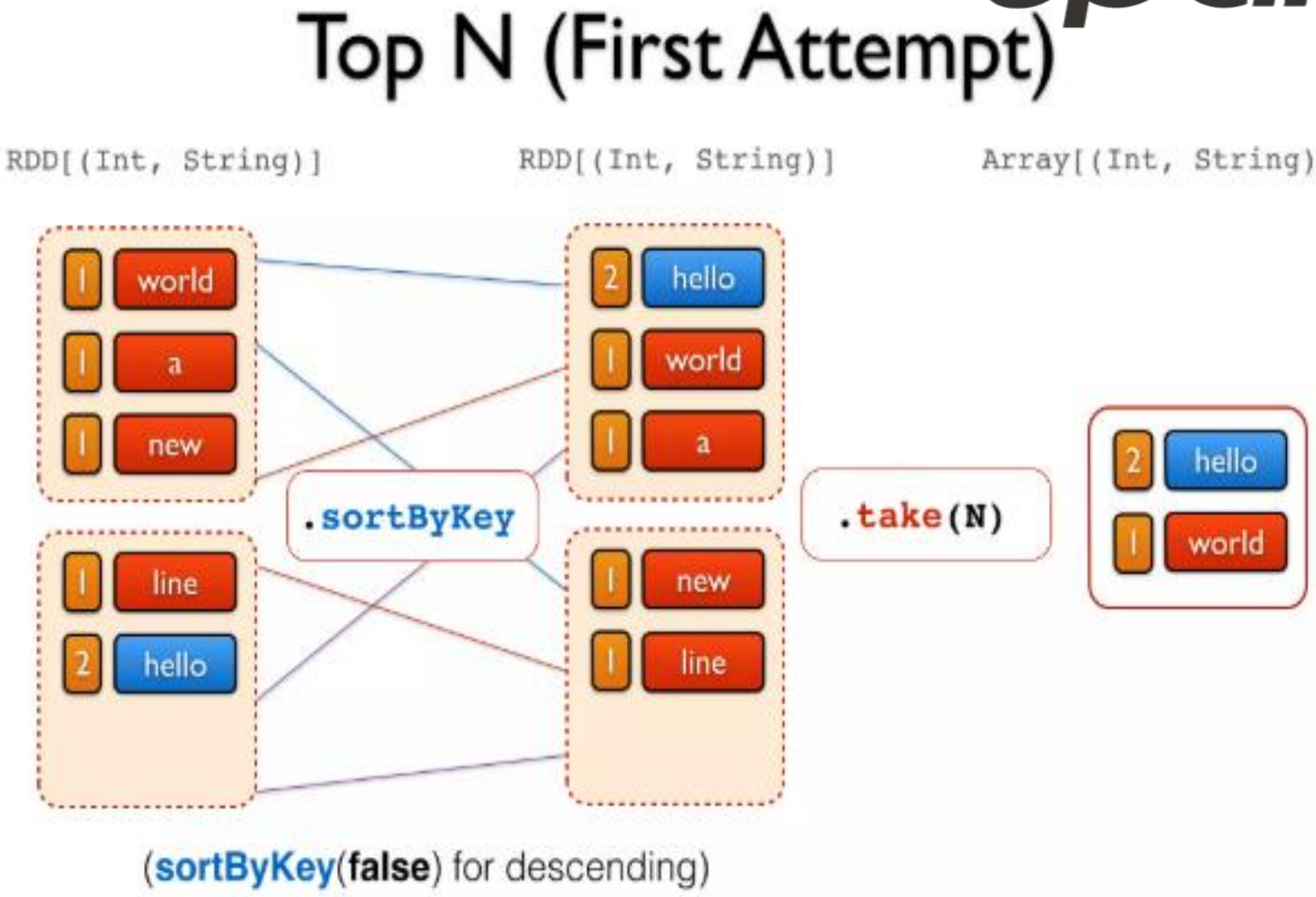
Exemple: Traitement sur texte

Top N (Prepare data)

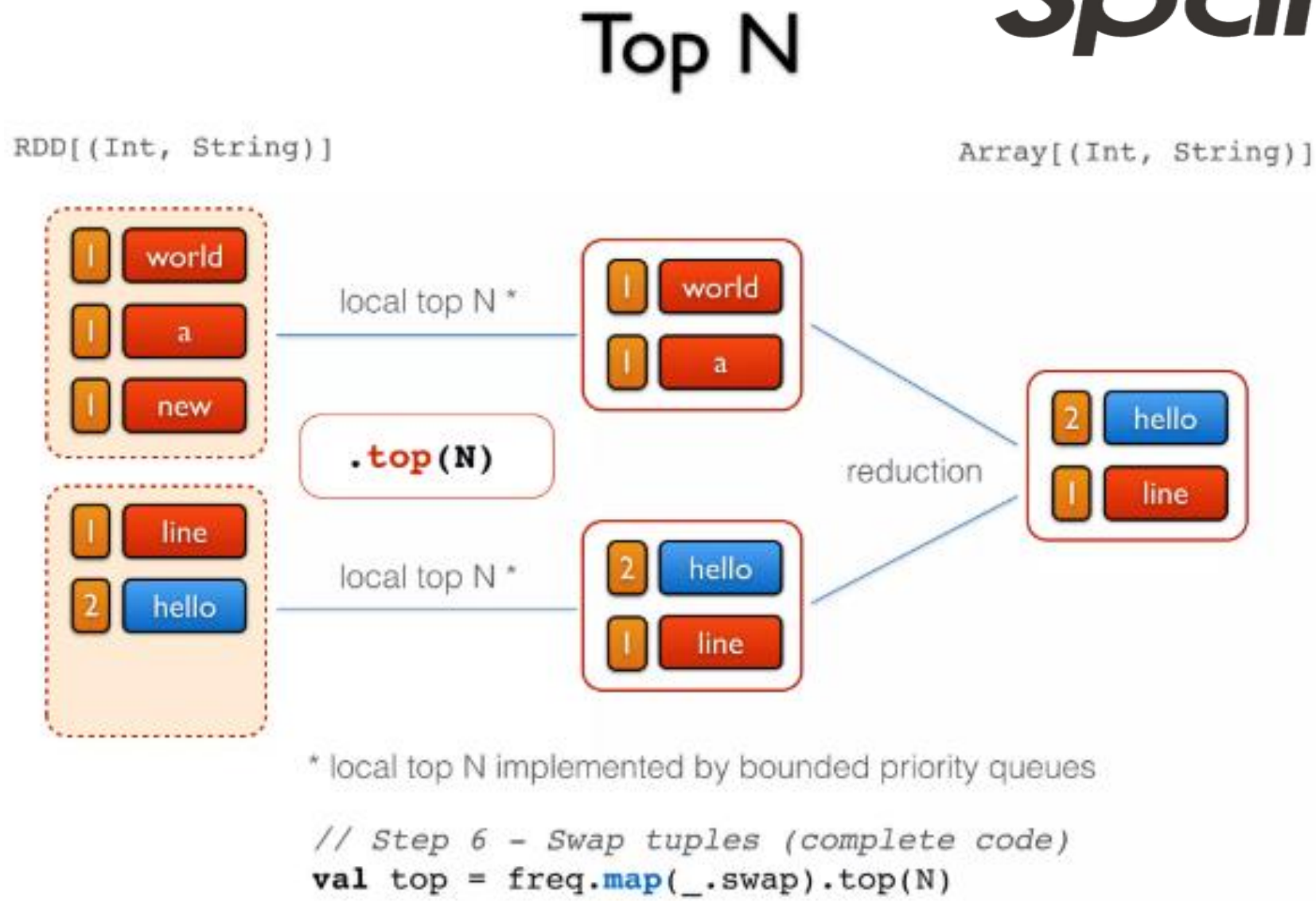



```
// Step 6 - Swap tuples (partial code)
freq.map(_.swap)
```

Exemple: Traitement sur texte



Exemple: Traitement sur texte



Références

- Fernando Rodrigez, (2014), « Apache Spark with Scala»
- Rudi Bruchez. (2015), « Les bases de données NoSQL et le Big Data », Editeur : Eyrolles, ISBN : 978-2-212-14155-9
- Rudi Bruchez (2021), « Les bases de données NoSQL », Editeur : Eyrolles ISBN : 978-2-212-67866-6
- Juvénal Chokogoue. (2017). « Hadoop - Devenez opérationnel dans le monde du Big Data », 8-2409007613