Apache Spark

Analyse complexe à grande échelle

Yannick Franck UM

Architecte, Consultant et Formateur

yannick.um@insightsunlocker.com

BIG DATA ?

De quoi s’agit-il ?

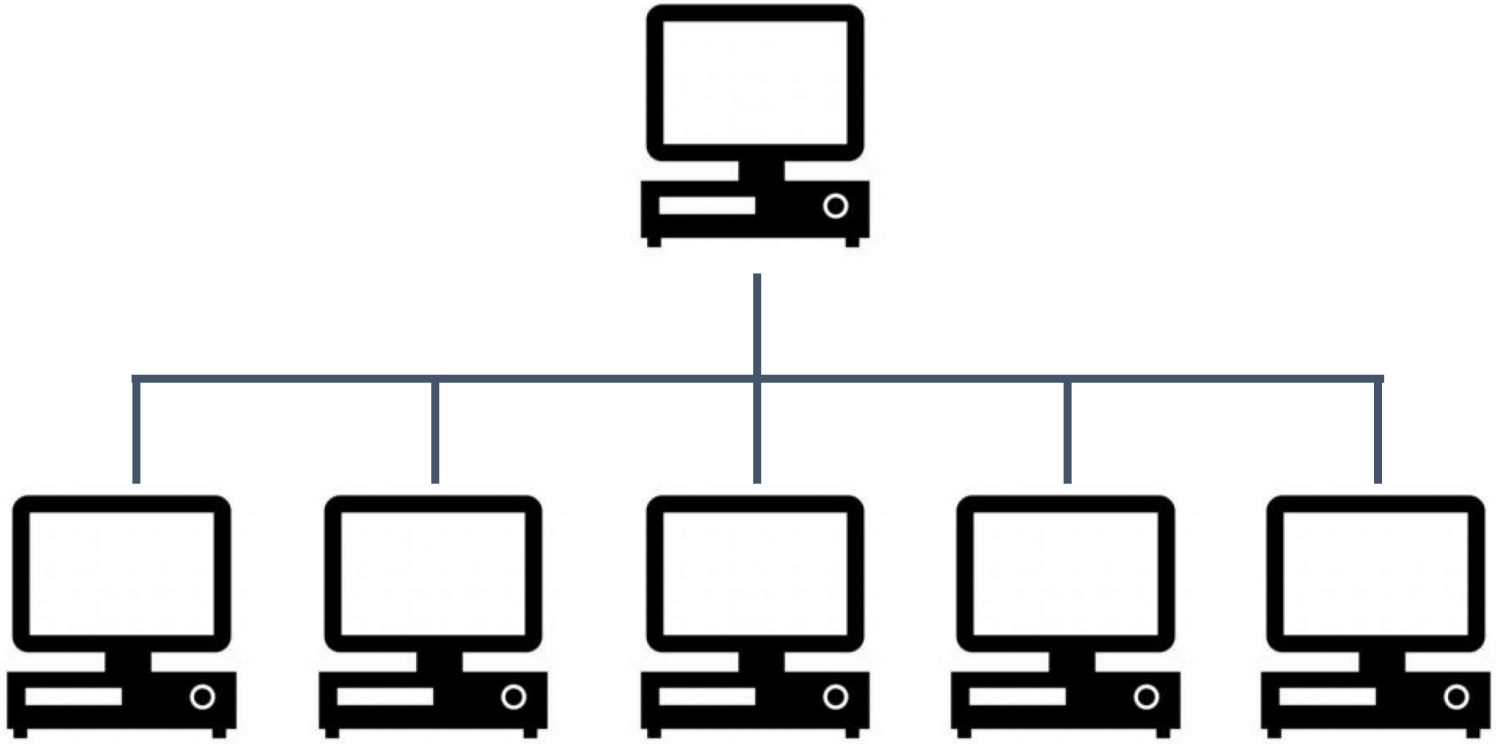
BIG DATA (Tableau blanc)

BIG DATA

- Données pouvant tenir sur un ordinateur local, dans une fourchette de 0 à 32 Go en fonction de la mémoire vive.
- Mais que pouvons-nous faire si nous disposons d'un plus grand nombre de données ?
  - Essayez d'utiliser une base de données SQL pour déplacer le stockage sur le disque dur au lieu de la mémoire vive.
  - Ou utiliser un système distribué, qui répartit les données sur plusieurs machines/ordinateurs.

Local ou distribué ?

Local



Distribué

BIG DATA

- Un processus local utilise les ressources de calcul d'une seule machine.
- Un processus distribué a accès aux ressources informatiques d'un certain nombre de machines connectées par le biais d'un réseau.

BIG DATA

- À partir d'un certain point, il est plus facile de passer à plusieurs machines dotées d'un processeur plus faible que d'essayer de passer à une seule machine dotée d'un processeur élevé.
- Les machines distribuées présentent également l'avantage d'être facilement extensibles : il suffit d'ajouter des machines supplémentaires.

BIG DATA

- Ils incluent également la tolérance aux pannes, c'est-à-dire que si une machine tombe en panne, l'ensemble du réseau peut continuer à fonctionner.
- Examinons le format typique d'une **architecture distribuée** qui utilise **Hadoop**.

BIG DATA

- **Hadoop** est un moyen de distribuer de très gros fichiers sur plusieurs machines.
- Il utilise le système de fichiers distribués Hadoop (HDFS).
- HDFS permet à l'utilisateur de travailler avec de grands ensembles de données.
- HDFS duplique également des blocs de données pour assurer la tolérance aux pannes.
- Il utilise également MapReduce
- MapReduce permet d'effectuer des calculs sur ces données

BIG DATA - Evolution

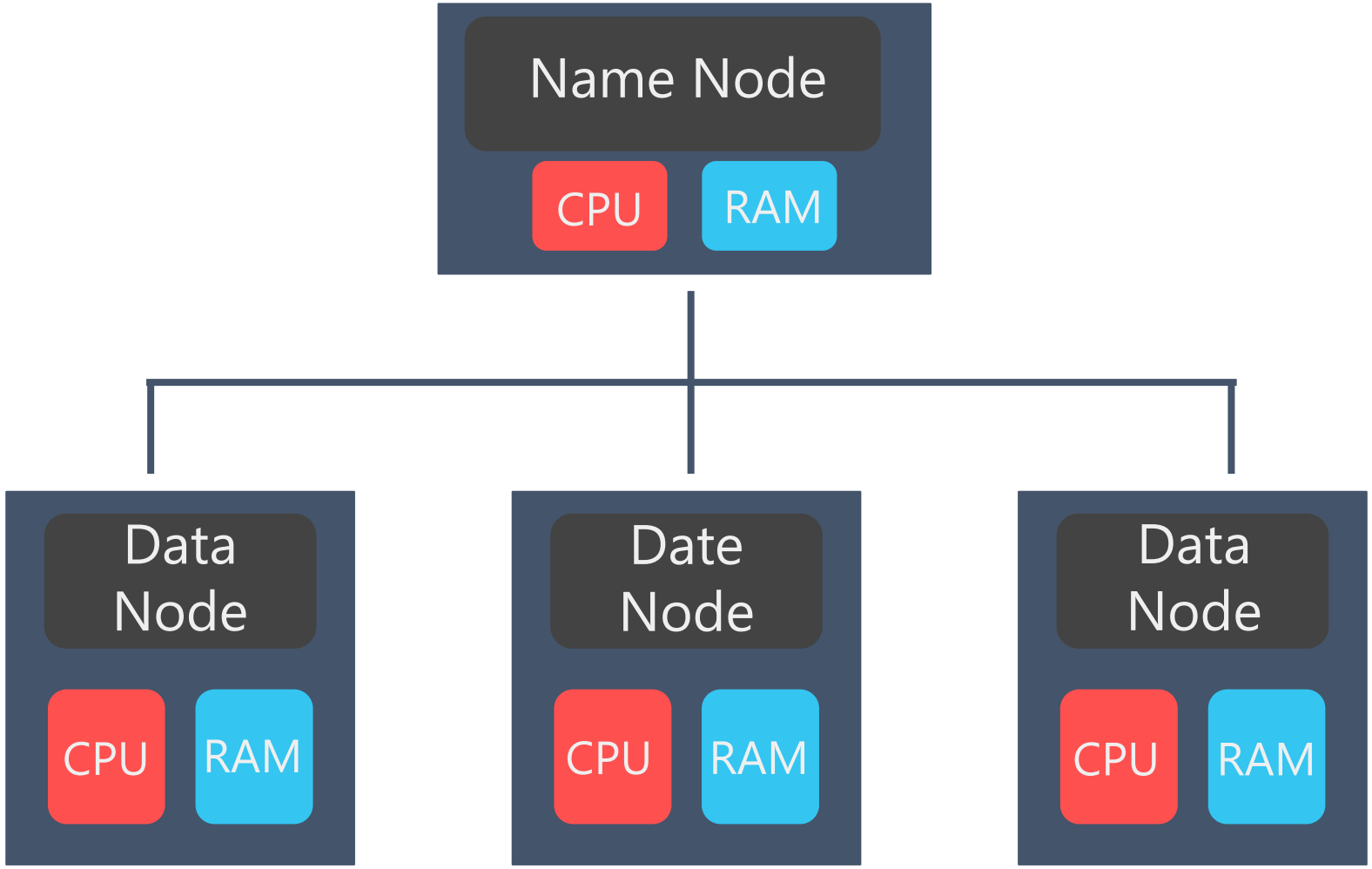
- **Hadoop & MR**
- **Spark**
- **Databricks & Microsoft Fabric**

HDFS

Du stockage distribué (diviser pour mieux régner)

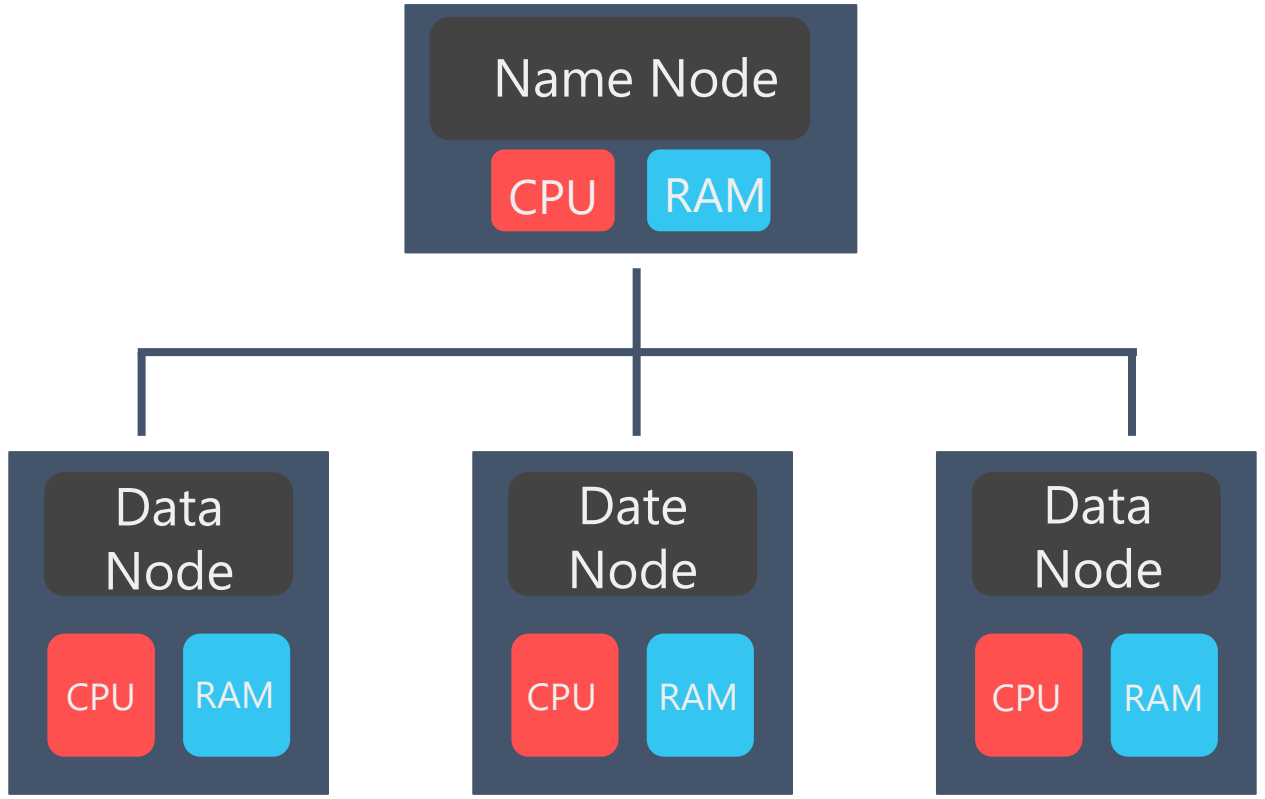
Stockage distribué – HDFS (Tableau blanc)

Stockage distribué - HDFS



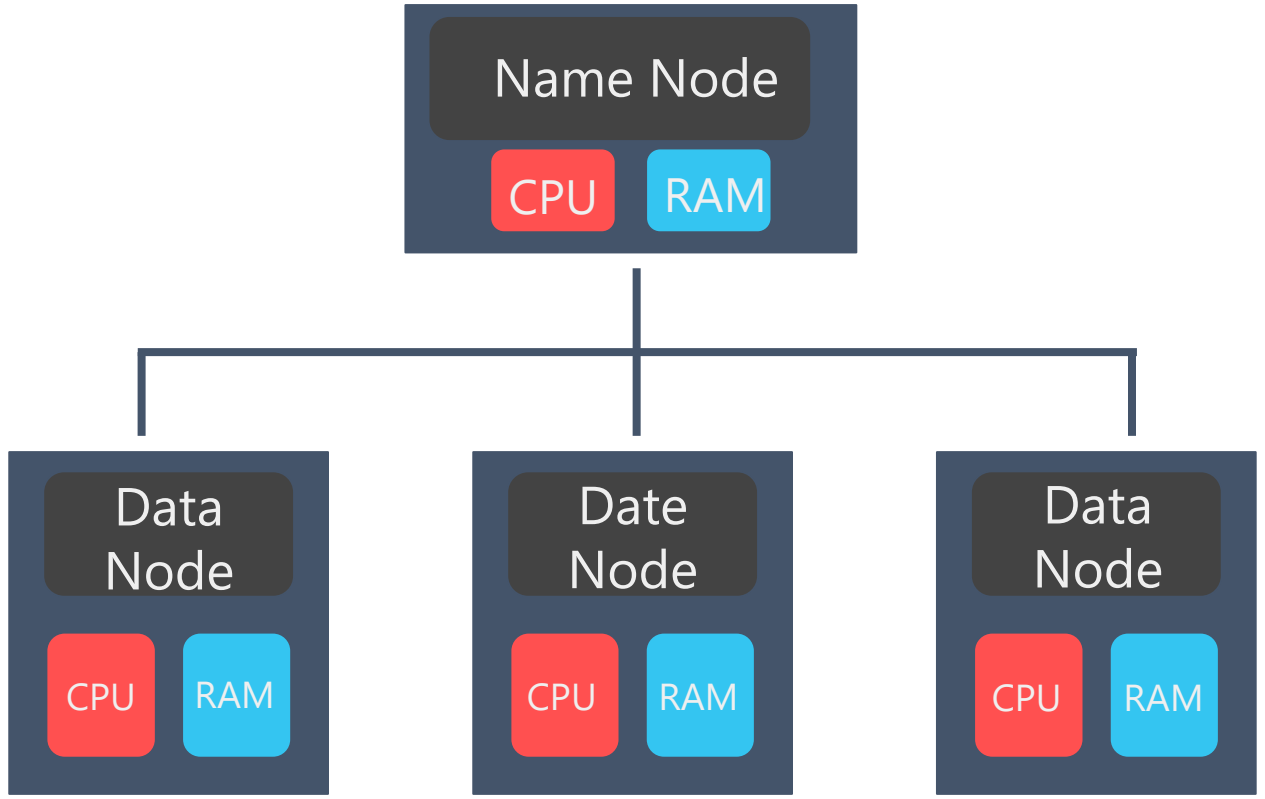
Stockage distribué - HDFS

- HDFS utilisera des blocs de données d'une taille de 128 Mo par défaut.
- Chacun de ces blocs est répliqué 3 fois
- Les blocs sont répartis de manière à favoriser la tolérance aux pannes.



Stockage distribué - HDFS

- Des blocs plus petits permettent une plus grande parallélisation lors du traitement.
- Les copies multiples d'un bloc évitent la perte de données en cas de défaillance d'un nœud.



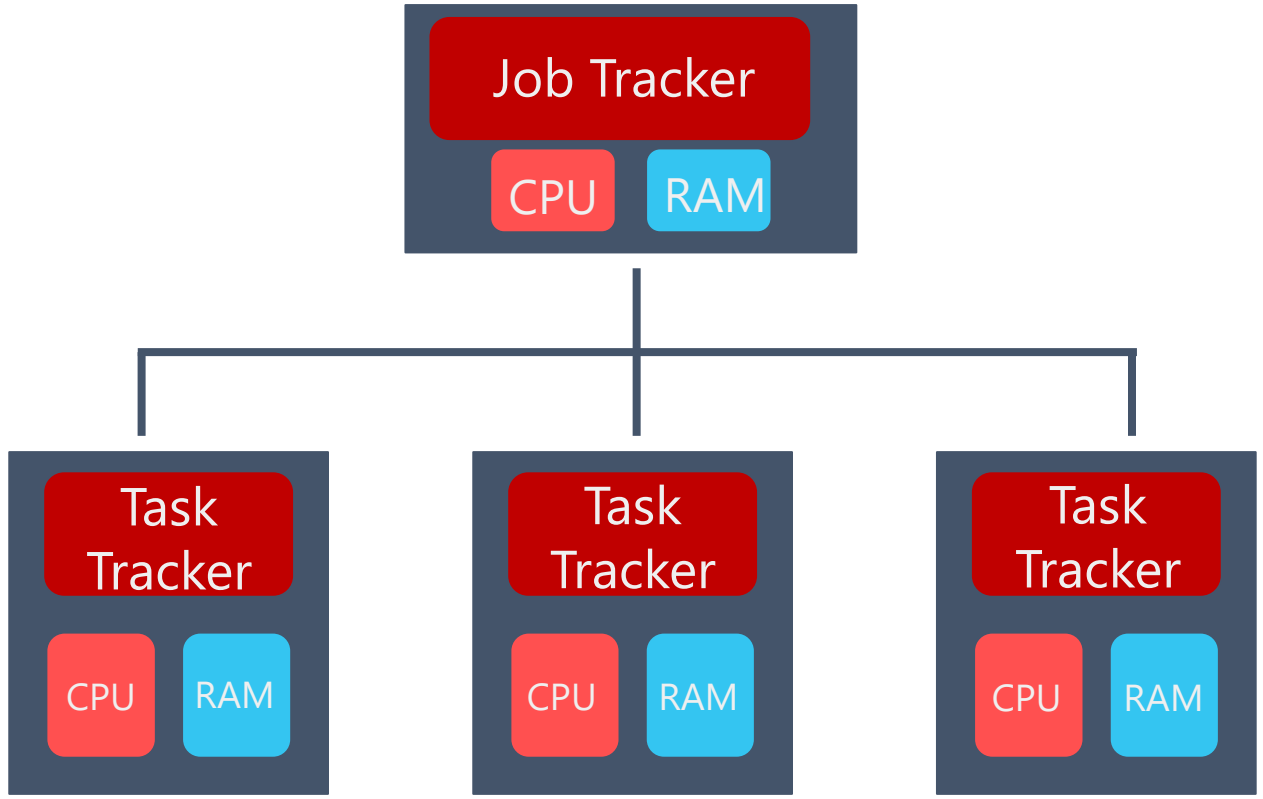
MapReduce

Du calcul distribué (diviser pour mieux régner)

MapReduce (Tableau blanc)

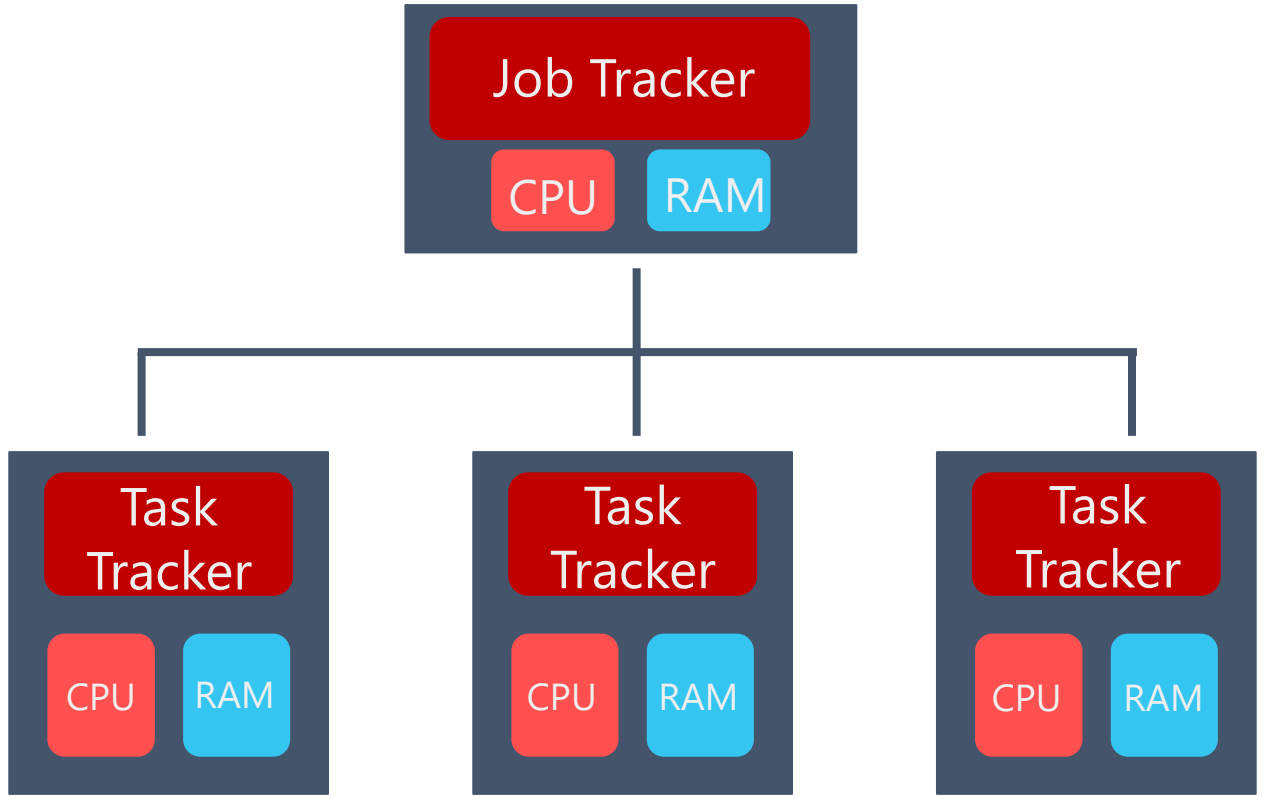
MapReduce

- **MapReduce** est un moyen de répartir une tâche de calcul sur un ensemble distribué de fichiers (tels que HDFS).
- Il se compose d'un outil de suivi des traitements (jobs) et de plusieurs noeuds de suivi des tâches.



MapReduce

- Le **Job Tracker** envoie le code à exécuter aux **Task Trackers**.
- Les traqueurs de tâches allouent l'unité centrale et la mémoire pour les tâches et surveillent les tâches sur les nœuds de traitements.



BIG DATA (EN SYNTHESE)

- Ce que nous avons couvert peut être considéré en deux parties distinctes :
  - Utiliser **HDFS** pour distribuer de grands ensembles de données
  - Utilisation de **MapReduce** pour distribuer une tâche de calcul à un ensemble de données réparties
- Ensuite, nous découvrirons l'une des dernières technologies en date dans ce domaine, appelée **Spark**.
- **Spark** améliore les concepts d'utilisation de la distribution

Spark

La révolution du calcul distribué !

De quoi allons-nous parler ?

- Des éléments ci-dessous :
  - Spark
  - Spark vs MapReduce
  - Spark RDDs
  - Spark DataFrames

Spark

- Spark est l'une des dernières technologies utilisées pour traiter **rapidement** et **facilement** les données volumineuses (Big Data).
- Il s'agit d'un projet open source sur Apache
- Il a été lancé pour la première fois en février 2013 et sa popularité a explosé en raison de sa facilité d'utilisation et de sa rapidité.
- Il a été créé à l'AMPLab de l'université de Berkeley.

Spark

Apache Spark est un **framework** de traitement des données distribué qui permet l'analytique de données à grande échelle en coordonnant le travail sur plusieurs nœuds de traitement dans un cluster.

En termes plus simples, Spark utilise une approche « diviser et conquérir » pour **traiter rapidement de grands volumes de données** en répartissant le travail sur plusieurs ordinateurs.

Le processus de distribution des tâches et de regroupement des résultats est géré pour vous par Spark. Vous envoyez un travail de traitement des données sous la forme d'un code qui lance un programme de pilote, qui utilise un objet de gestion de cluster appelé **SparkContext** pour gérer la distribution du traitement dans le cluster Spark. Dans la plupart des cas, ces détails sont abstraits. Il vous suffit d'écrire le code nécessaire pour effectuer les opérations de données dont vous avez besoin.

Spark

Spark peut exécuter du code écrit dans un large éventail de langages, notamment :

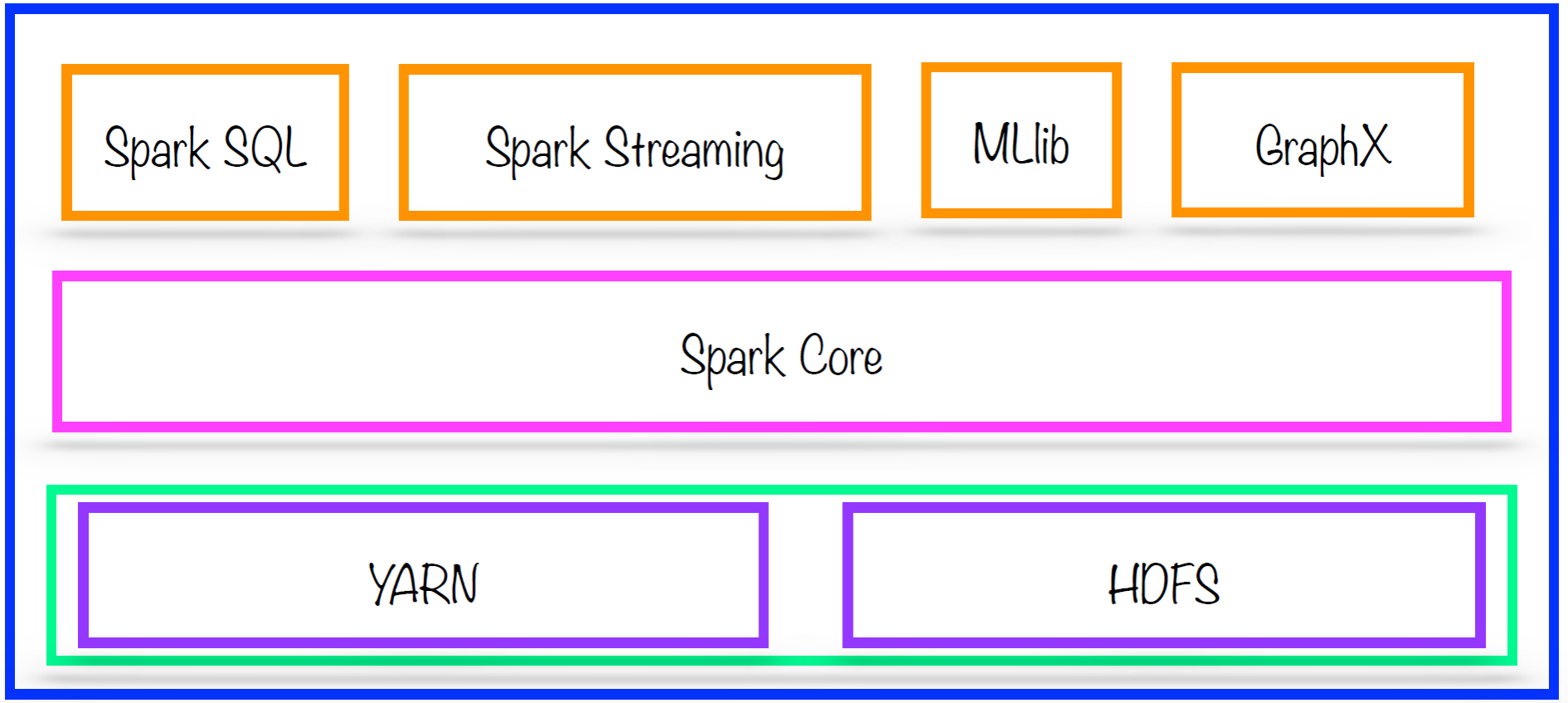
- Java
- Scala (un langage de script basé sur Java)
- Spark R
- Spark SQL
- PySpark (une variante de Python propre à Spark).

La plupart des charges de travail d'engineering données et d'analytique des données sont effectuées à l'aide d'une combinaison de **PySpark** et de **Spark SQL**

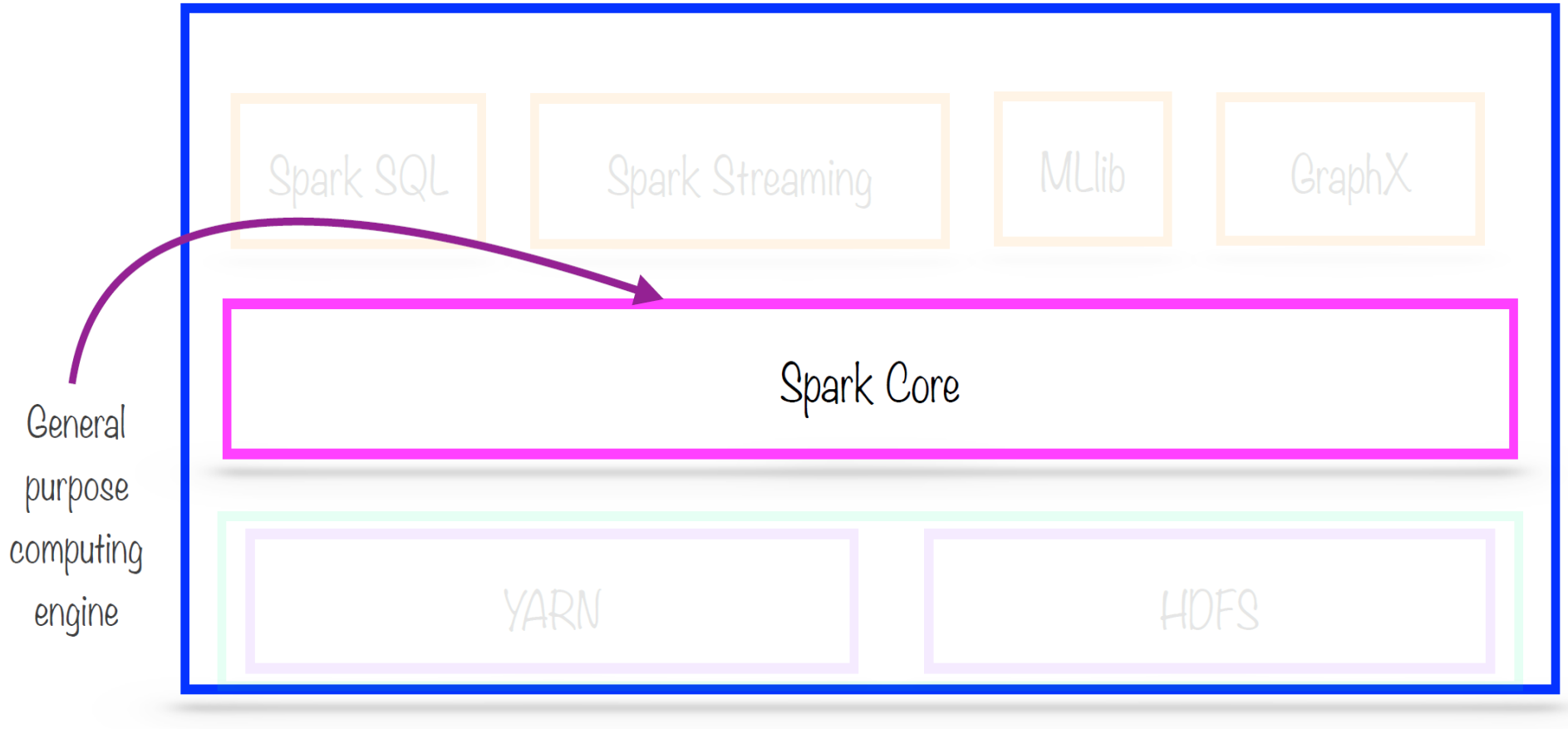
Spark

- Vous pouvez considérer Spark comme une alternative flexible à MapReduce.
- Spark peut utiliser des données stockées dans différents formats
  - HDFS
  - Azure Data Lake
  - Amazon S3
  - Lakehouse, etc.

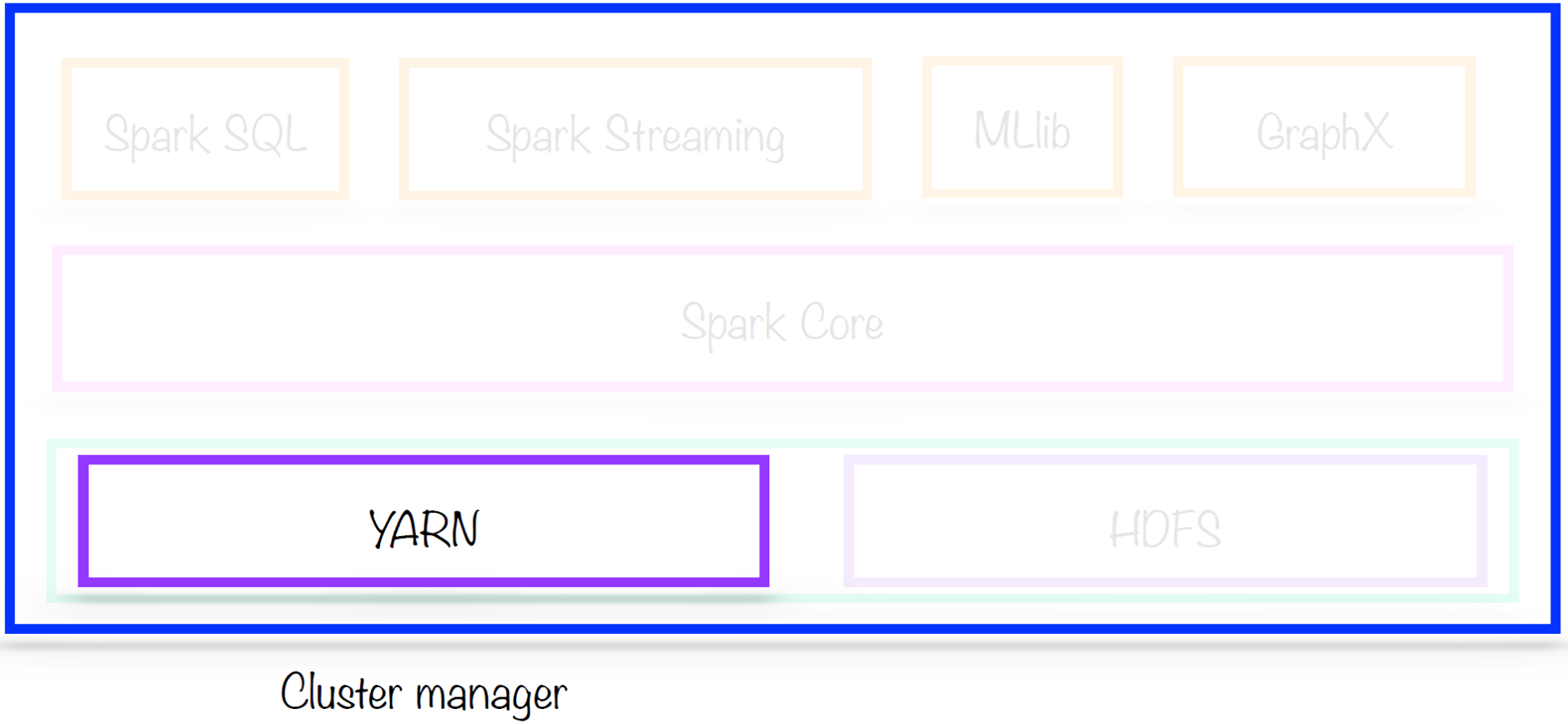
Architecture de Spark



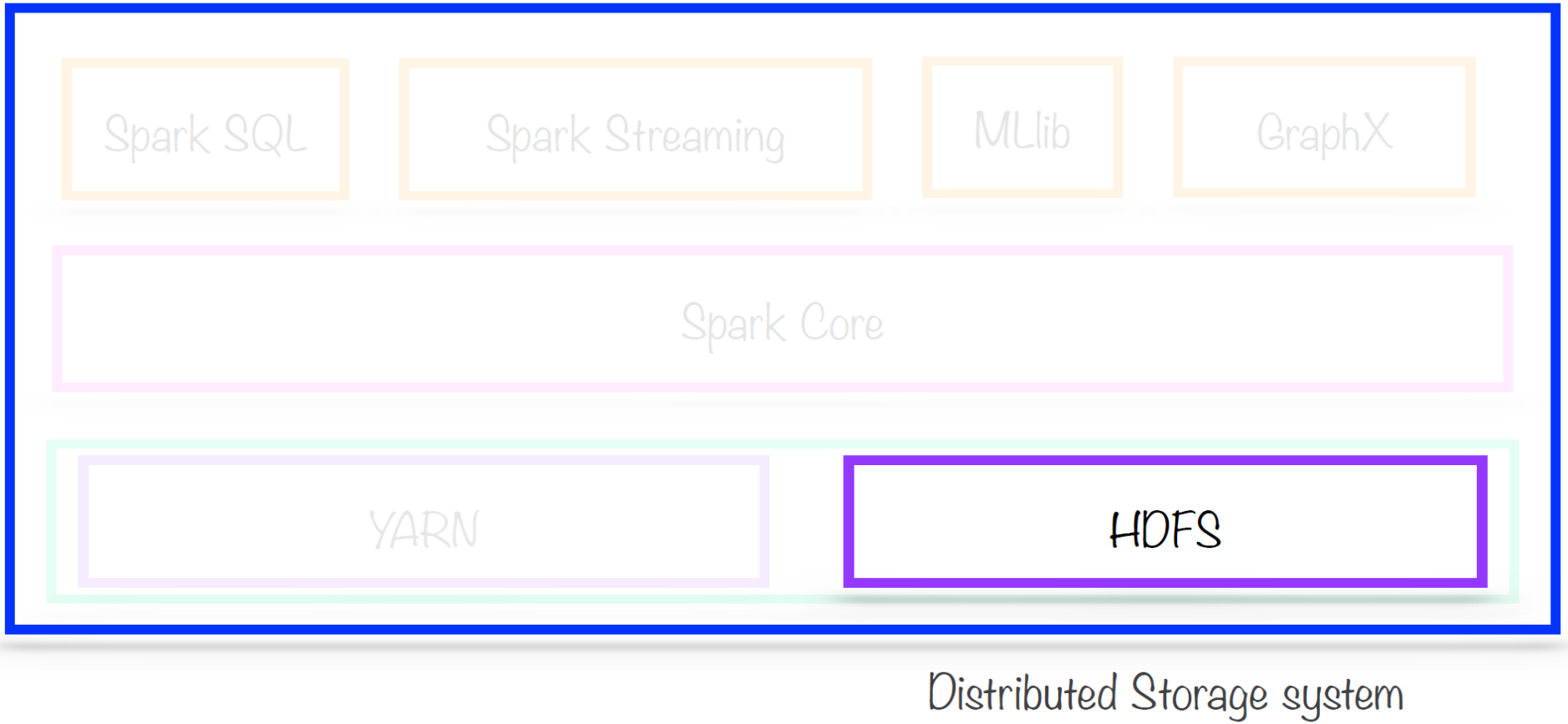
Architecture de Spark



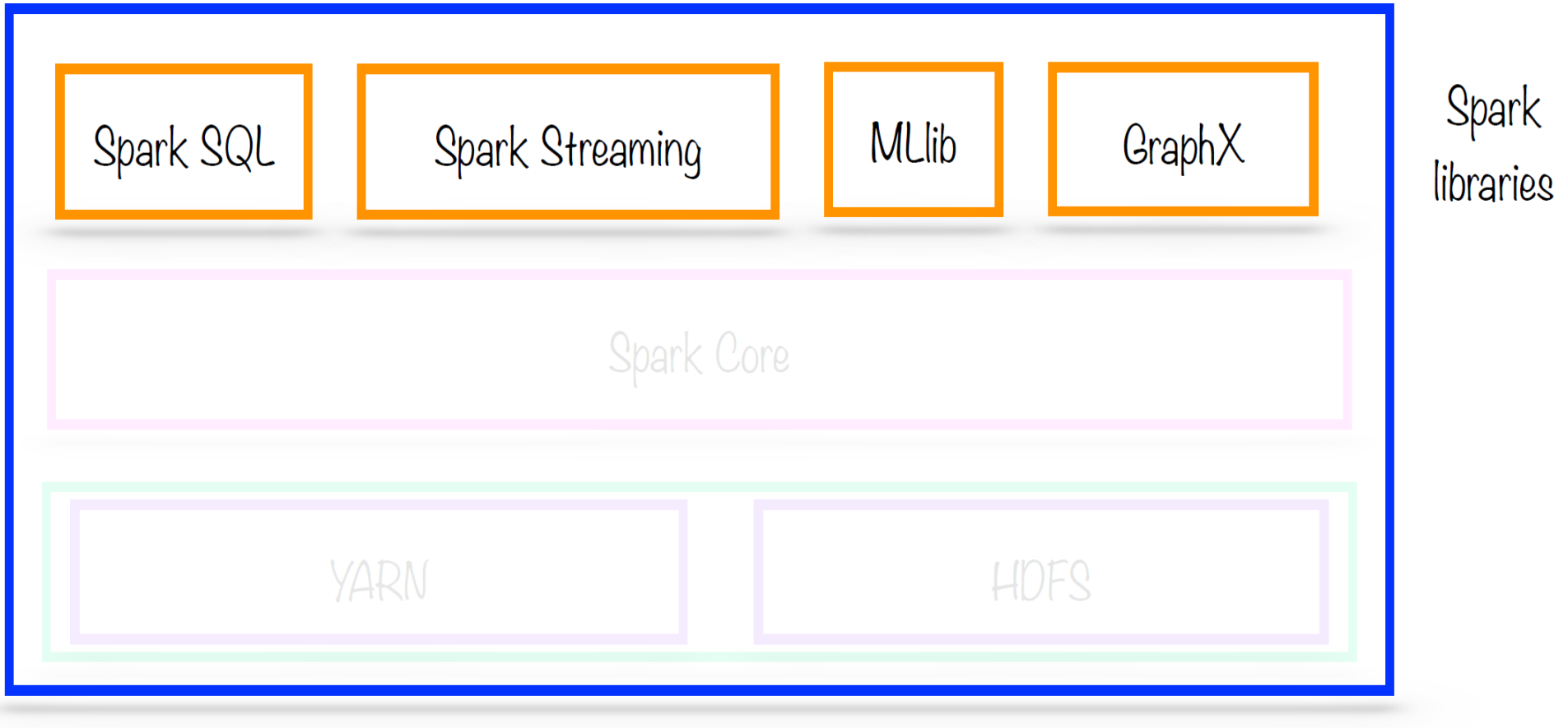
Architecture de Spark



Architecture de Spark



Architecture de Spark



Spark vs MapReduce

- MapReduce exige que les fichiers soient stockés dans HDFS, ce qui n'est pas le cas de Spark !
- Spark peut également effectuer des opérations jusqu'à **100 fois plus rapidement** que MapReduce.
- Comment atteint-il cette vitesse ?

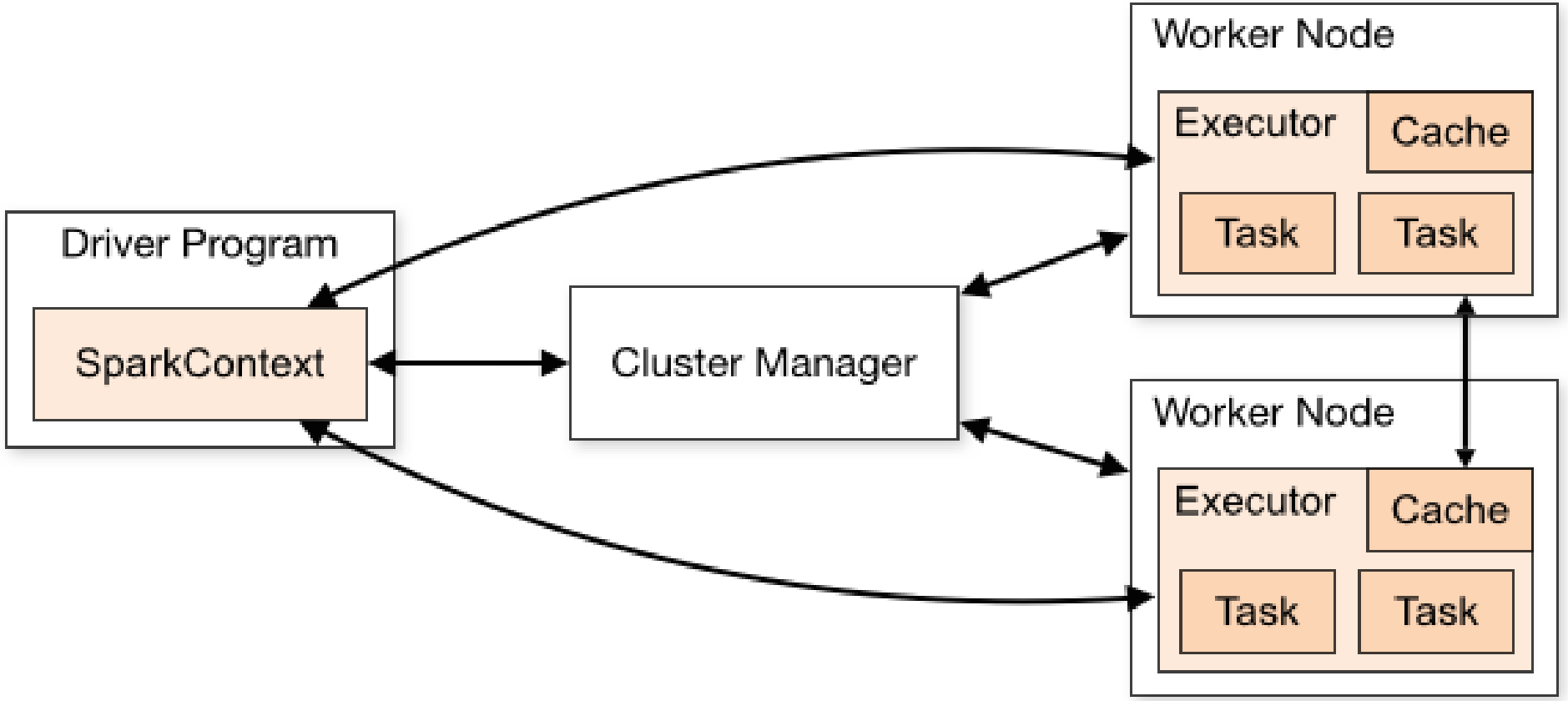
Spark vs MapReduce

- MapReduce écrit la plupart des données sur le **disque** après chaque opération de mappage et de réduction.
- Spark conserve la plupart des données en **mémoire** après chaque transformation
- Spark peut se déverser sur le disque si la mémoire est remplie.

Spark RDD

- Au cœur de Spark se trouve l'idée d'un ensemble de données distribuées résilientes (RDD).
- L'ensemble de données distribuées résilientes (RDD) présente quatre caractéristiques principales :
  - Collecte distribuée des données
  - Tolérance aux pannes
  - Fonctionnement en parallèle - partionné
  - Capacité à utiliser de nombreuses sources de données

Spark RDD


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

Spark RDD

- Les RDD sont **immuables**, **évalués paresseusement** et peuvent être **mis en cache**.
- Il existe **deux types d'opérations** Spark :
  - *Transformations*
  - *Actions*
- Les transformations sont essentiellement une recette à suivre.
- Les actions exécutent réellement ce que la recette demande de faire et renvoient quelque chose en retour.

Spark RDD

Resilient Distributed Datasets

All operations in Spark are performed on

in-memory objects

Spark RDD - charactéristiques

Partitioned	Immutable	Resilient
Split across data nodes in a cluster	RDDs, once created, cannot be changed	Can be reconstructed even if a node crashes

Spark RDD

- Ce comportement se répercute sur la syntaxe lors du codage.
- Souvent, vous écrivez un appel de méthode, mais vous ne verrez pas le résultat tant que vous n'aurez pas appelé l'action.
- C'est logique, car avec un grand ensemble de données, vous ne voulez pas calculer toutes les transformations avant d'être sûr de vouloir les effectuer !

Spark RDD

- Lorsque vous discuterez de la syntaxe Spark, vous verrez apparaître la syntaxe RDD par rapport à la syntaxe DataFrame.
- Avec la sortie de Spark 2.0, Spark évolue vers une syntaxe basée sur les DataFrame, mais gardez à l'esprit que la façon dont les fichiers sont distribués peut toujours être considérée comme des RDD, c'est juste la syntaxe typée qui change.

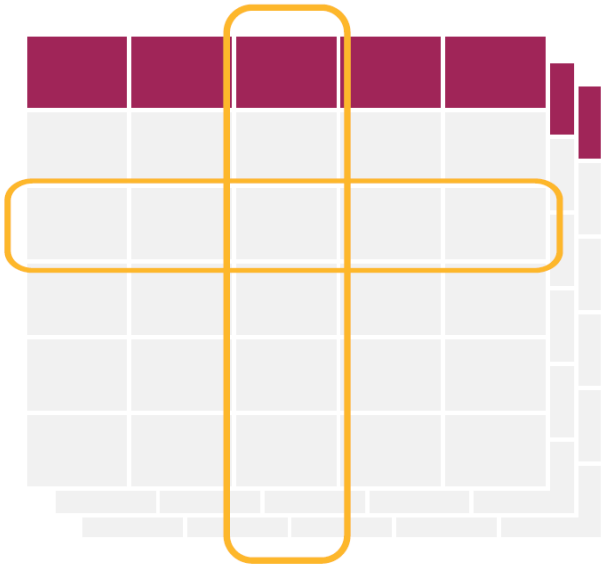
Spark RDD

- Nous avons couvert beaucoup de choses !
- Ne vous inquiétez pas si vous n'avez pas mémorisé tous ces détails, nous reviendrons sur beaucoup d'entre eux lorsque nous apprendrons à coder et à utiliser ces idées !

Spark Dataframes

- Les DataFrames Spark sont désormais le moyen standard d'utiliser les capacités d'apprentissage automatique de Spark.
- La documentation de Spark DataFrame est encore assez récente et peut être éparse.
- Faisons un tour d'horizon de la documentation !

Spark Dataframes



DataFrame = RDD + schema

Spark : site officiel

https://spark.apache.org/

Spark : Lab 01

Mise en place de l'environnement de travail (Spark sur le PC).

Anaconda & conda

Premières manipulations de Dataframes avec Spark.