
M. El malki & El haddadi & Arlind Koplku
UT2 -2024
Mohammed.El-Malki@irit.fr

# Règle 1



***Arrêtez moi si vous ne comprenez pas***

***ou si vous avez des questions!***

# Plan du cours

- **Quelles motivations ?**
- **Calcul distribué (Hadoop & MapReduce)**
- **Qu'est-ce que SPARK ?**
- **L'environnement SPARK**
- **Le concept SPARK**
  - **RDD (Resilient Distributed Dataset)**
  - **DataFrame**
- **Interaction avec Spark**

# Pourquoi s'intéresser aux Big Data?



- C'est la mode
  - ça grandit très vite
  - tout le monde en parle (privé, public, recherche)
  - une communauté active de plus de 300 personnes rien qu'à Toulouse
- Ca permet d'embaucher
  - http://www.information-management.com/gallery/Big-Data-Cloud-Best-Salaries-10026460-1.html
  - http://www.edureka.co/blog/10-reasons-why-big-data-analytics-is-the-best-career-move
- Ca mieux rémunéré

# Pourquoi c'est important?



- Parce que les technologies existantes, dites traditionnelles ne sont pas capables de faire face à la quantité et à la diversité croissante des données.
- Parce que l'entreprise a compris que donnée = $

# Quelles données?



- mes données
  - mes transactions, les logs de mon site Web, tout sur mon processus d'entreprise, tout sur mon marketing …
- mais aussi les données des autres (en vente ou pas)
- les données en libre accès (Open Data)
- le Web
- …

# Big data: definition 1

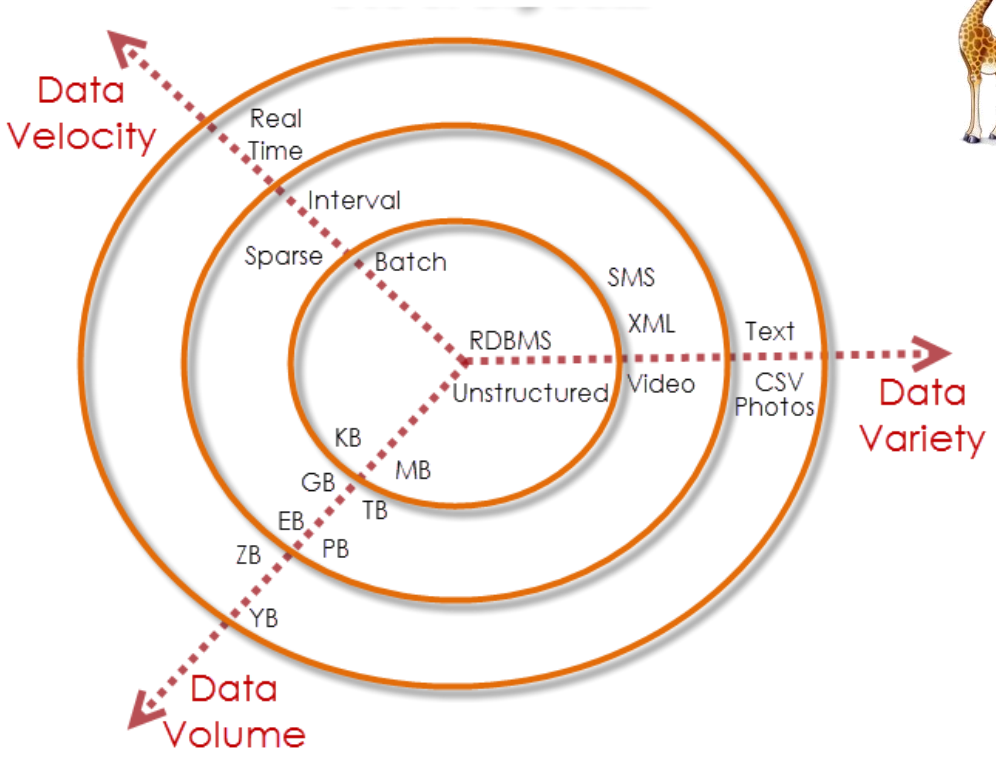


Big Data (Mégadonnées) sont des données qui excèdent la capacité de traitement de systèmes de bases de données classiques pour la capture, stockage, gestion, ou analyse.

Big data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze

# Big data: Les trois V



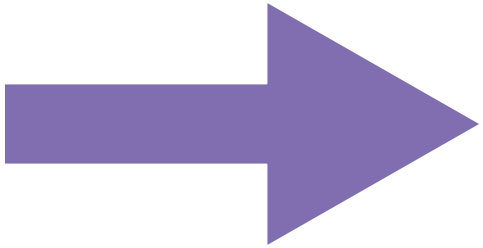



# Big data: Les 5 V





completeness, validity, coherence, quality of data
…




service, product improvement
cost reduction, new services
…

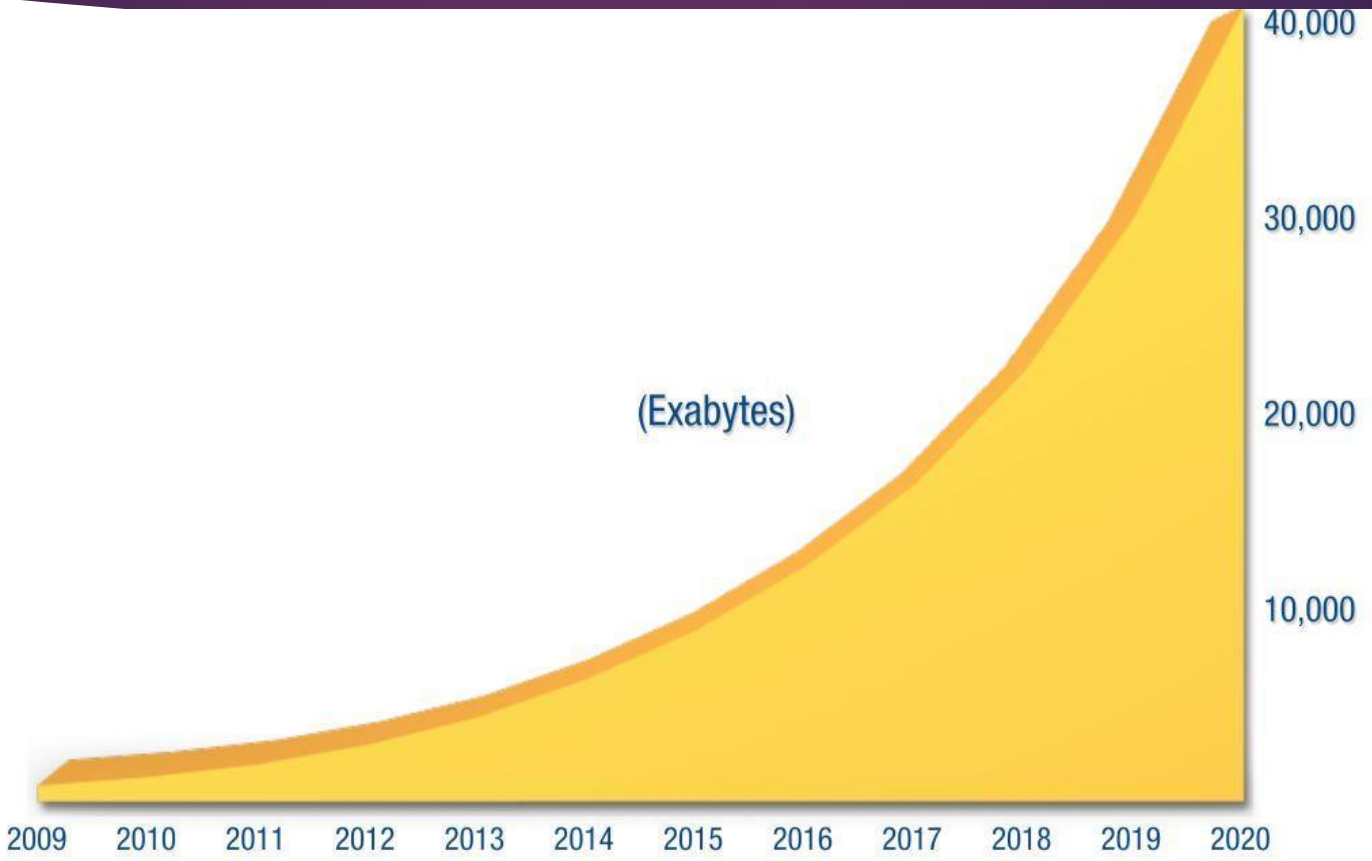
# Value?



- 100 milliards$, la valeur du Big Data en 2010
- Business grandissant de 10% par an
- 140000-190000 postes à remplir seulement aux Etats Unis jusqu'à l'année 2018 dans le domaine de Big Data Analytics et domaines connexes
- 125000$ le salaire moyen aux Etats Unis par an

# Big? Volume






Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

# Ok, mais quoi de neuf?



- calcul distribué: Hadoop, Spark, MapReduce
- analytics:, **Spark**, SAS, Knime
- langage de script + Hadoop: R, Python,

# Quelle philosophie de traitement?

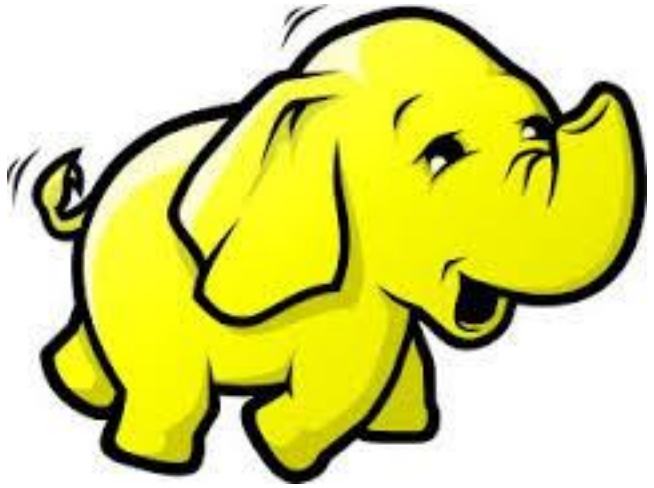


- **Le principe de localité des données avec les outils traditionnels**
  - Les données → le traitement

transfert par le réseau très pénalisant

- **Dans les nouvelle tendances**
  - Les données ← le traitement

Techniques de développement complexe



La première révolution : HADOOP

# Hadoop



- Hadoop est un framework open source développé en java, faisant partie des projets de la fondation Apache depuis 2009. Il a été conçu pour :
  - Stocker des volumes de données très importants ;
  - Supporter des données de formats variés, structurés, semi- structurés ou non structurés.
  - Hadoop est basé sur un ensemble de machines formant un cluster Hadoop. Chaque machine est appelé nœud

C'est l'addition des capacités de stockage et de traitement de ses nœuds qui lui assure un important système de stockage et une puissance de calcul

# Cluster Hadoop



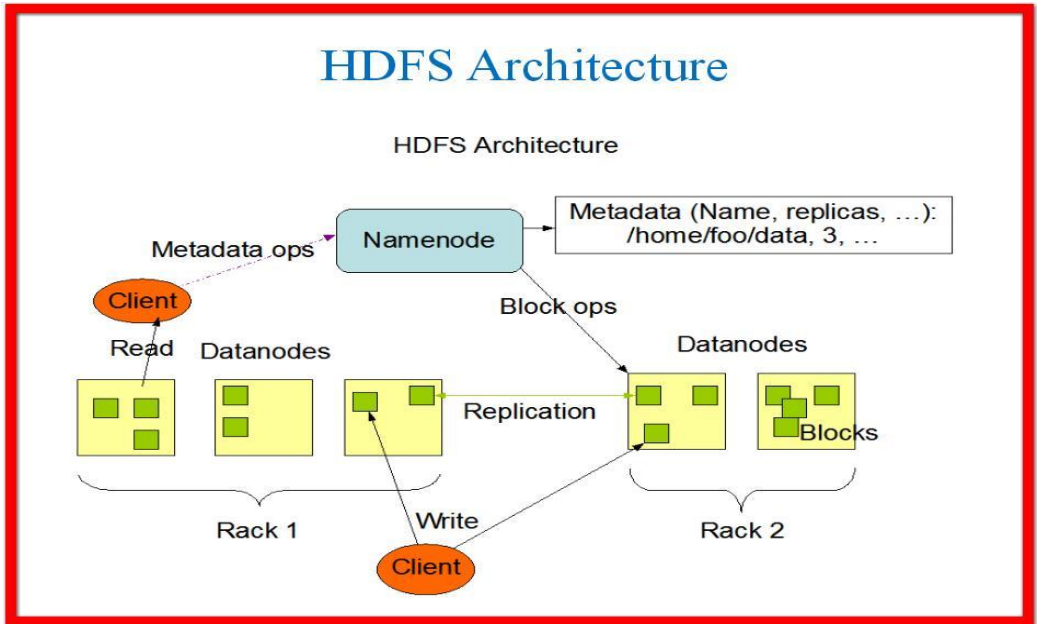
- Hadoop
  - Espace de stockage appelé HDFS (Hadoop Distributed File System)
  - Paradigme de calcul distribué appelé MapReduce

# HDFS



- HDFS est le composant chargé de stocker les données dans un cluster Hadoop
  - Basé sur une architecture maître -esclave
  - repose dans son fonctionnement sur trois types de nœuds :

- Namenode :
  - le nœud maître est le pilote du système HDFS, au moyen de métadonnées
- Le secondary namenode
  - effectuer à intervalle réguliers des sauvegardes du namenode
- DataNode
  - ils servent d'espace de stockage et de calcul des blocs de données




*Source Big Data week*

# Le paradigme MapReduce

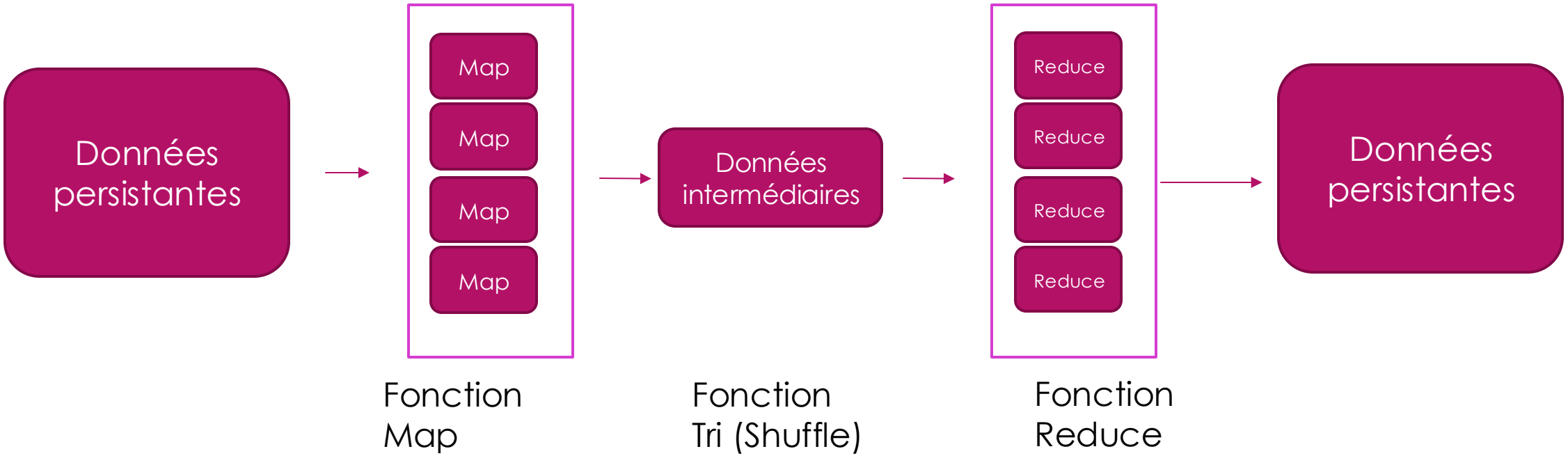


- MapReduce est un modèle de programmation parallèle développé spécifiquement pour
  - lire, traiter des volumes de données très importants
  - écrire des volumes de données très importants
  - dans un environnement distribué
- principe de fonctionnent est très simple :
  - décomposer une tâche en taches plus petites (N éléments) manipulées séparément
  - **notion de clé-valeur via deux fonctions Map et Reduce**

# Le paradigme MapReduce



Un job MapReduce est composé de trois phases :



# Le paradigme MapReduce : Map

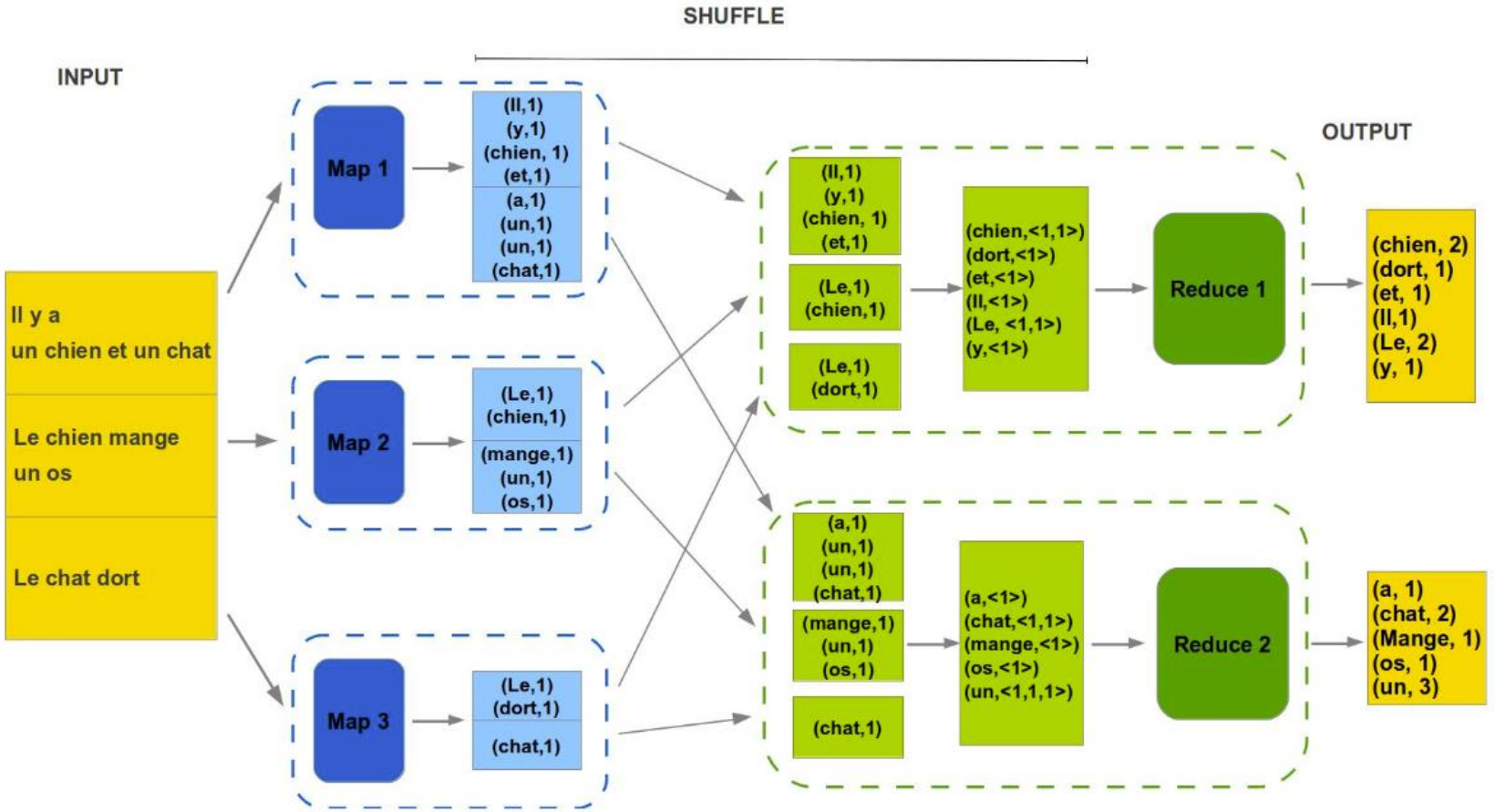


- La fonction Map
  - Lit les données sous forme de clé/valeur
  - Effectue le traitement souhaité
  - Retourne des données sous forme clé/valeur

- La fonction de tri (Shuffle)
  - Transfère les données produit par la fonction Map , tri et fusionne les données pour la fonction Reduce
- La fonction Reduce
  - Lit les données fournies  Map (après la phase Suffle)
  - Effectue le traitement d'agrégation souhaité (Somme, moyenne…)
  - Retourne un résultat final sous forme clé/valeur

# MapReduce : Exemple wordcount





# Cluster Spark

- 2009: Développé à l'université de Berkeley (AMPLAB)
- En 2014 : devenu une propriété d'APCHE
- 2014: sortie de la première version (1.0.0)

- Pour quel traitement :
  - traiter de gros volume de données
    - par des algorithmes parallèles robustes
  - spécialisés pour les applications qui nécessitent la réutilisation de données
    - Données intermédiaires en mémoire (mais aussi en disque)

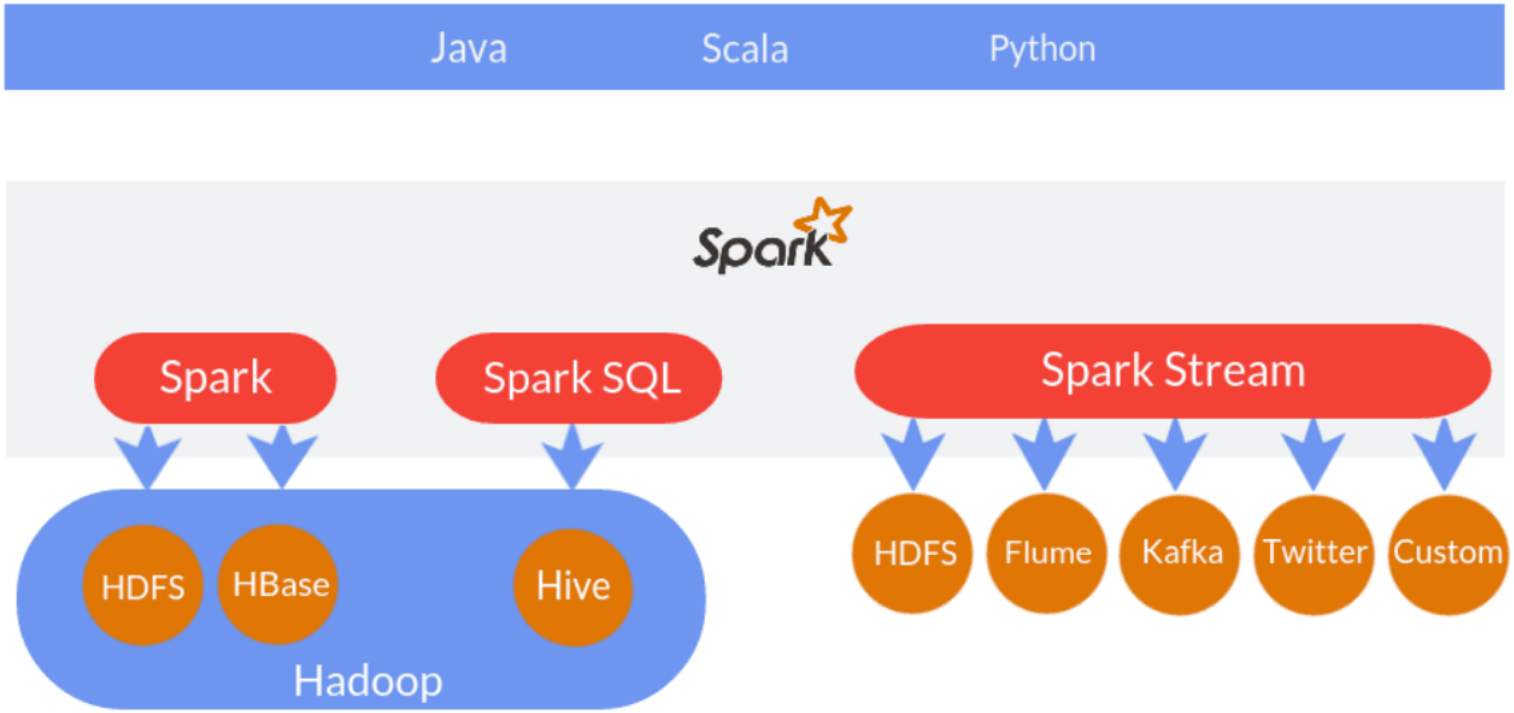
# Qu'est-ce que SPARK ?



- Les motivations
  - Moins de temps comparé à Hadoop
  - Moins de lignes comparé aux langage tels que java…
  - Plus interactif

# Architecture Spark

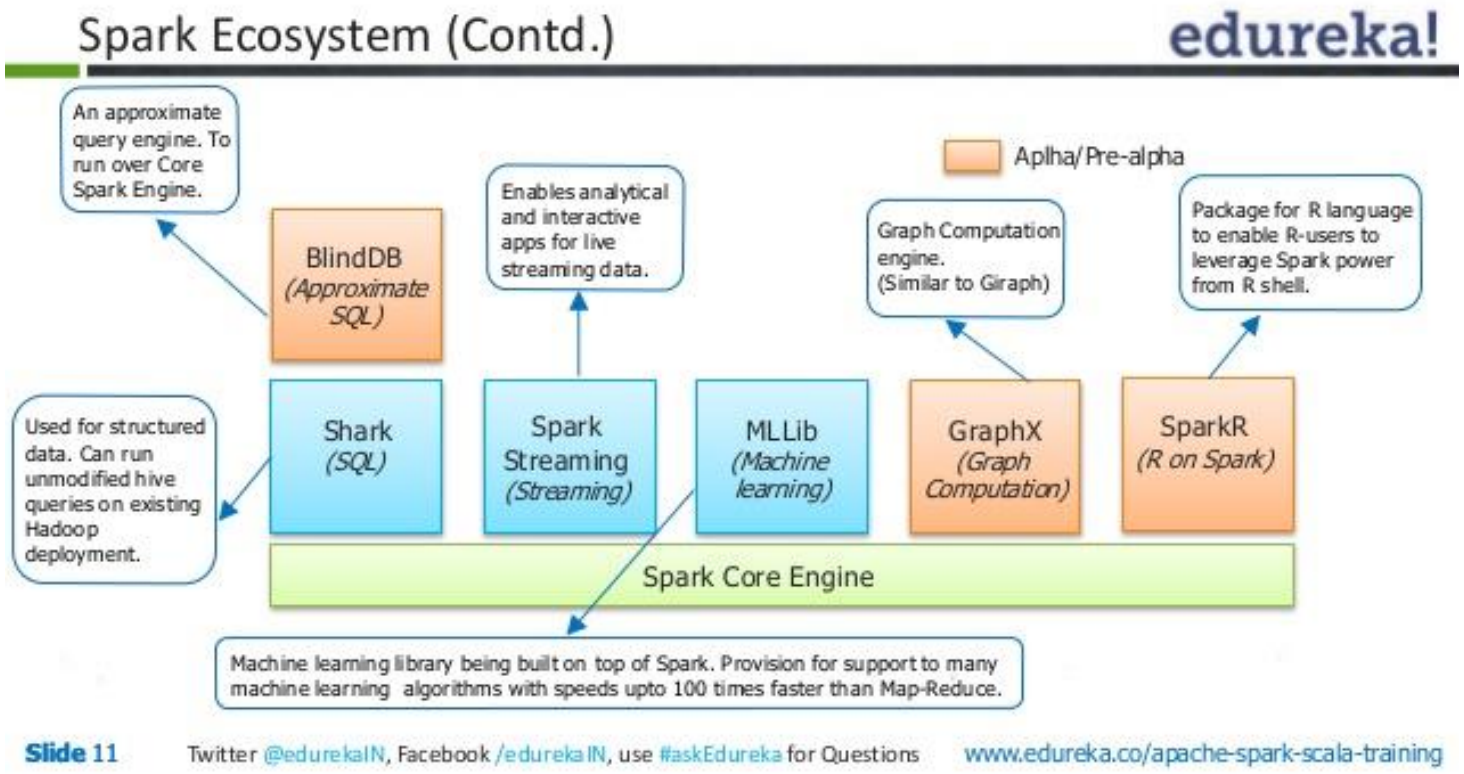




# L'environnement SPARK



- Spark
- Spark shell
- Spark Streaming
- Spark SQL
- Spark MLlib
- Spark GraphX



# L'environnement SPARK



- **Spark Streaming :**
  - utilisé pour traitement temps-réel des données en flux.
  - Il s'appuie sur un mode de traitement en "micro batch" et utilise pour les données temps-réel DStream,
- **Spark SQL :**
  - d'exposer les jeux de données Spark via API JDBC et
  - d'exécuter des requêtes de type SQL en utilisant les outils BI et de visualisation traditionnels.
  - permet d'extraire, transformer et charger des données sous différents formats (JSON, Parquet, base de données) et les exposer pour des requêtes ad-hoc.

# L'environnement SPARK

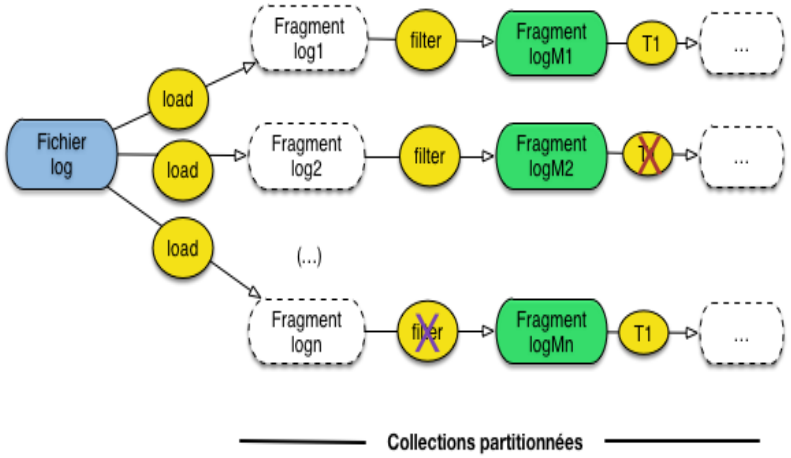


- **Spark MLlib**
  - est une libraire de machine learning
  - contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le clustering, le filtrage collaboratif, la réduction de dimensions, en plus des primitives d'optimisation sous-jacentes.
- **Spark GraphX**
  - la nouvelle API (en version alpha) pour les traitements de graphes et de parallélisation de graphes. De plus,
  - inclut une collection toujours plus importante d'algorithmes et de builders pour simplifier les tâches d'analyse de graphes

# Fonctionnement de spark

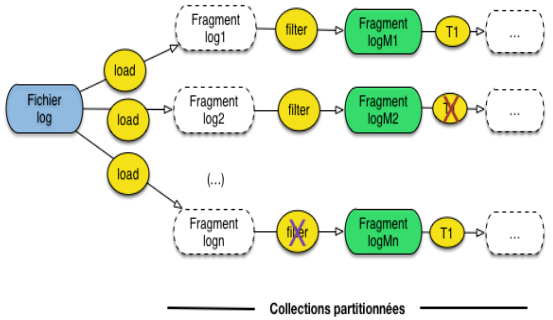


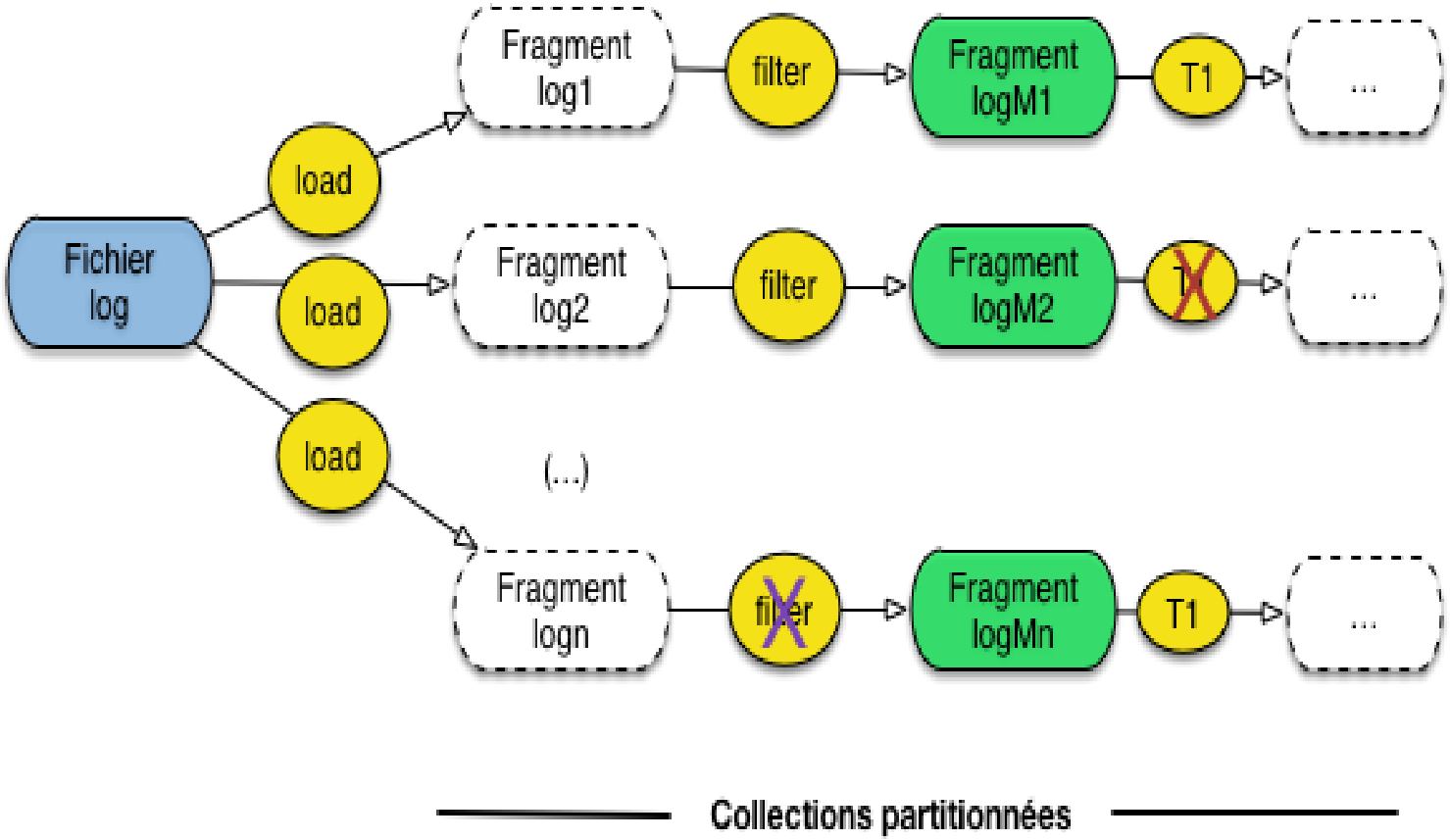
- Le cœur du concept
  - **RDD (Resilient Distributed Datasets)**
    - est une collection de données abstraite
    - résultant de la transformation d'un autre RDD ou d'une création à partir de données existantes.
    - L'historique des opérations (Ayant permis de constituer ou cré le RDD)  est conservé pour mieux gérer les pannes
- Traitement des données en parallèle
  - Un RDD est distribué sur l'ensemble des machines (paralléliser le traitement)
  - Chaque RDD est donc constitué de plusieurs blocs
  - Une panne affectant un fragment individuel peut donc être réparée (par reconstitution de l'historique) indépendamment des autres fragments, évitant d'avoir à *tout* recalculer



# RDD (Représentation)

- Le choix de la représentation des RDDs doit permettre de retrouver la situation initiale au travers des transformations appliquées. Pour ce faire, une représentation sous forme de graphe a été choisie. En fait, chaque RDD est représenté par une interface commune qui expose différents types d'informations :
  - un ensemble de partitions,
  - un ensemble de dépendances aux RDDs parents,
  - une fonction pour calculer l'ensemble de données en se basant sur ses parents,
  - et des métadonnées sur ses plans de distribution ainsi que sur l'emplacement des données.




Fichier log
load
load
load
Fragment log1
Fragment log2
Fragment logn
(...)
filter
filter
filter
Fragment logM1
Fragment logM2
Fragment logMn
T1
T1
T1
...
...
...
Collections partitionnées

# RDD : manipulations

- Un RDD repose sur deux types d'opérations:
  - Transformations
    - Pas de valeurs retournées, aucune évaluation
    - En entrée: un RDD, en sortie:  un autre RDD
  - Actions
    - Evaluation et retour de valeurs
    - Des requêtes de traitement des données sont calculées et un  résultat est retourné

# RDD : les actions:

- reduce(func) :
- collect() : retourne l'ensemble des items (enregistrements)
- count() : retourne le nombre d'items (enregistrements)
- take(n) : retourne le nombre n d'items (enregistrements)
- first() : retourne le premier item (enregistrement)
- countByKey() : retourne le nombre de clé/valeur
- takeSample(withReplacement, num, [seed])
- saveAsTextFile(path) : sauvegarde le resultat (appliqué sur un RDD) dans un fichier
- saveAsObjectFile(path) : Sauvegarde le résultat appliqué sur un RDD dans un Objet

# RDD : les transformations

| | |
|---|---|
| `map` | Prend un document en entrée et produit un document en sortie |
| `filter` | Filtre les documents de la collection |
| `flatMap` | Prend un document en entrée, produit un ou plusieurs document(s) en sortie |
| `groupByKey` | Regroupement de documents par une valeur de clé commune |
| `reduceByKey` | Réduction d'une paire *(k, [v])* par une agrégation du tableau *[v]* |
| `crossProduct` | Produit cartésien de deux collections |
| `join` | Jointure de deux collections |
| `union` | Union de deux collections |
| `cogroup` | Cf. la description de l'opérateur dans la section sur Pig |
| `sort` | Tri d'une collection |

# Persistance & partitionnement



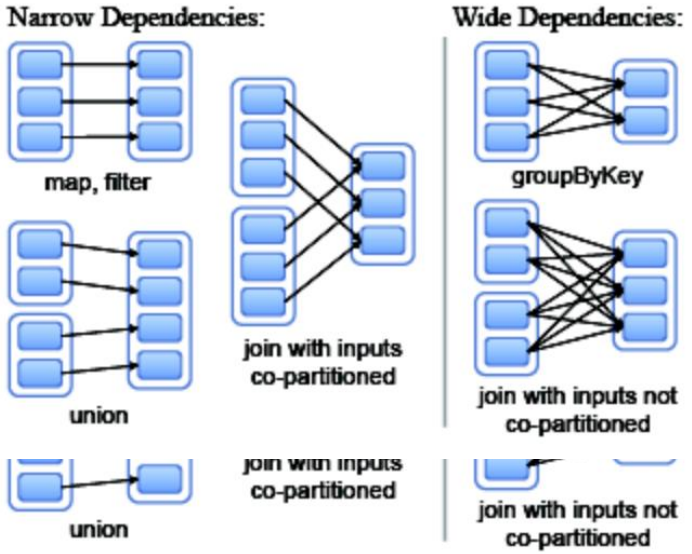
- les utilisateurs peuvent contrôler deux autres aspects des RDDs :
  - la persistance : il est possible de préciser quels sont les RDDs réutilisés ainsi que la stratégie de stockage,
  - le partitionnement: les éléments des RDDs peuvent être partitionnés entre les machines en se basant sur une clé dans chaque enregistrement.

# RDD : Persistance & partitionnement



Fonctions pour les métadonnées de partitions RDD

| Operation | Meaning |
|---|---|
| partitions() | Return a list of Partition objects |
| preferredLocations(*p*) | List nodes where partition *p* can be accessed faster due to data locality |
| dependencies() | Return a list of dependencies |
| iterator(*p*, *parentIters*) | Compute the elements of partition *p* given iterators for its parent partitions |
| partitioner() | Return metadata specifying whether the RDD is hash/range partitioned |



# Les variables partagées



- Deux types **de variables partagées** (au niveau du cluster Spark) :
  - Les variables Broadcast (Variable stockée en cache sur chaque machine)

```
val broadcastVar = sc.broadcast(Array(1, 2, 3))
broadcastVar.value
```

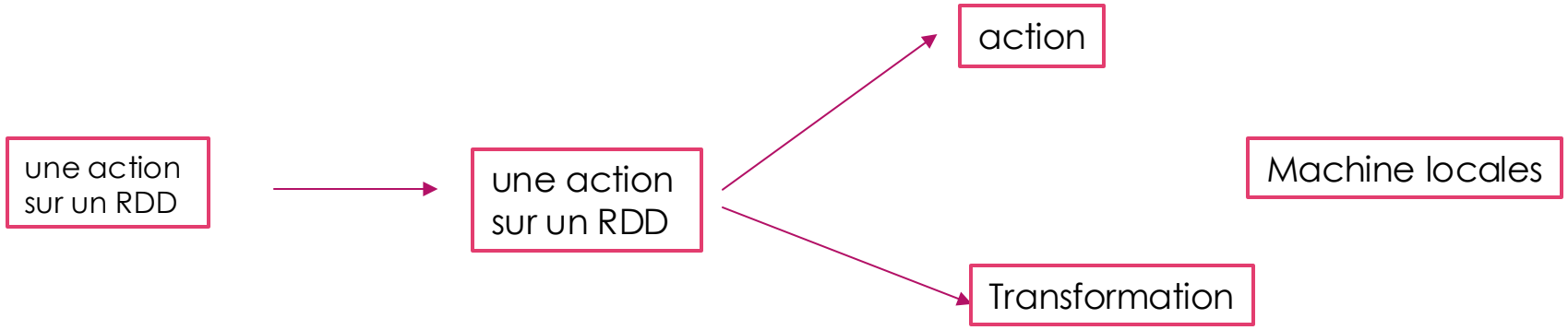
  - Les accumulateurs (Is peuvent être utilisés pour implémenter des compteurs (comme avec MapReduce) ou des sommes).

```
val accum = sc.accumulator(0, "My Accumulator")
sc.parallelize(Array(1, 2, 3, 4)).foreach(x => accum += x)
accum.value
```

# L'ordonnanceur de job



- L'ordonnanceur de Spark utilise la représentation des RDDs



# Création de RDD, stockage de RDD



- Il y a deux possibilités pour créer un RDD :
  - Référencer des données externes.
  - Paralléliser une collection existante
    - Spark permet de créer un RDD à partir de toute source de données acceptée par Hadoop (fichier local, HDFS, HBase, etc.)
      - SparkContext.textFile
      - SparkContext.wholeTextFiles
      - SparkContext.sequenceFile[K, V]
      - SparkContext.hadoopRDD
      - SparkContext.newHadoopRDD

# Création de RDD, stockage de RDD



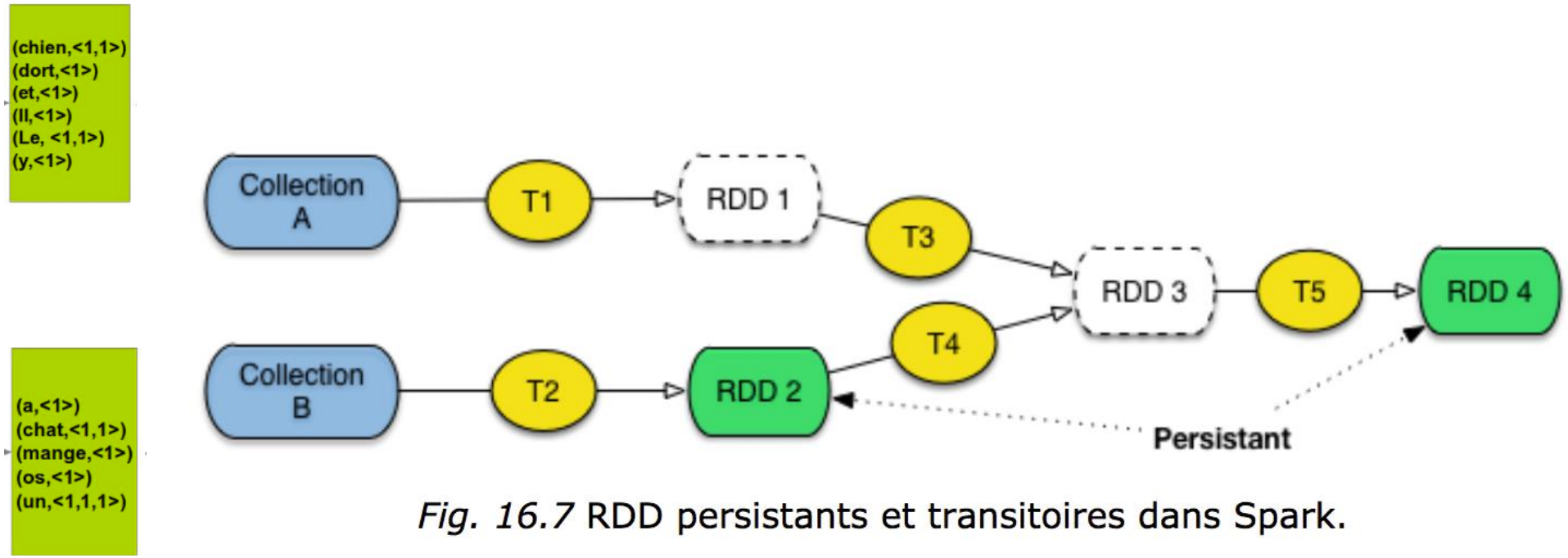
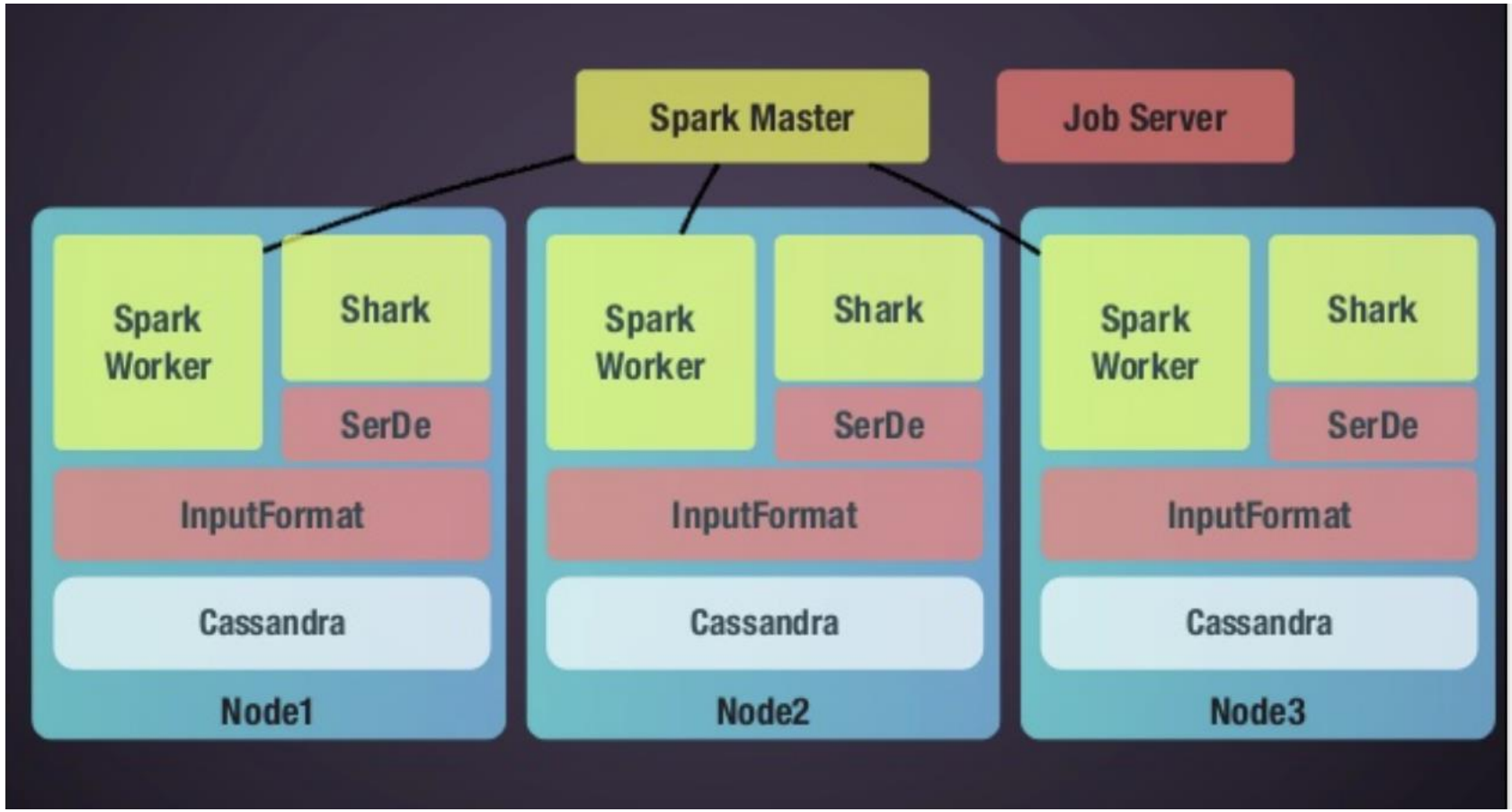



*Fig. 16.7* RDD persistants et transitoires dans Spark.

# Création de RDD (source externe)





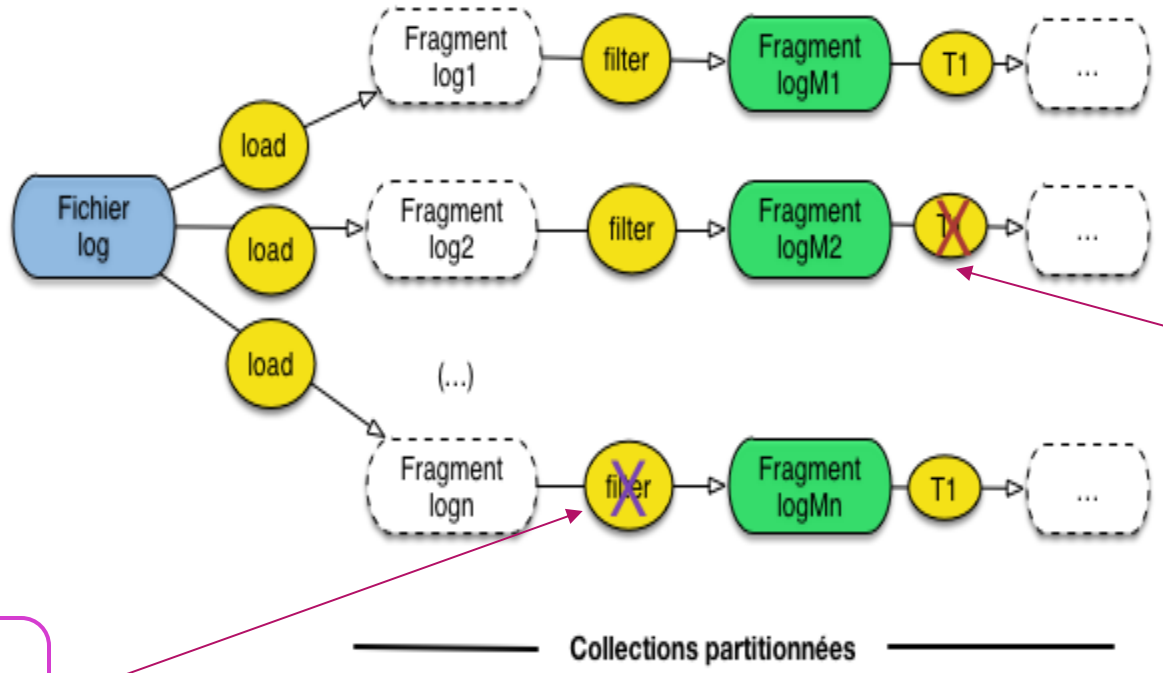
# Gestion d'une Panne



L'historique des opérations (Ayant permis de constituer ou créer le RDD) est conservé pour mieux gérer les pannes

Le gain en temps est plus importan t

Le gain en temps est considérable




relancer à partir de *F (Persistant)*

relancer à partir de *F (Persistant)*

# Spark SQL



- Les fonctionnalités (offertes pour interroger en Spark SQL):
  - **DataFrame :**
  - **Data Sources**
  - **JDBC Server (litre des données depuis des BD)**
- **Les composants de Spark SQL**
  - **DataFrame**
  - **SQLContext (crée par SparlContext pour e**ncapsuler les fonctions du monde relationnel dans Spark**)**

https://spark.apache.org/docs/1.3.0/api/scala/index.html#org.apache.spark.sql.DataFrame

# DataFrame



- **DataFrame :**
  - est une collection de données distribuées, organisées en colonnes nommées.
  - Peut être en convertie en RDD (en appelant la méthode rdd (On convertie une «  table /schéma relationnel» en sous forme de RDD de lignes), Peuvent être crées depuis
    - Des RDD existants
    - Des fichiers de données structurées
    - Des jeux de données JSON
    - Des tables HIVE
    - Des bases de données externes

# Les langages



- **Spark est écrit en Scala et peut être indifféremment utilisé en Java, en Scala et en Python** :
- **Scala** :
  - Un terminal Spark-shell reposant
  - écriture et exécution de traitements en direct.
- **Python** :
  - Un terminal Spark-shell reposant
  - écriture et exécution de traitements en direct.
  - (Quelques fonctionnalités manquantes)
- **Java** : La couche d'adaptation Java/Scala un peu plus lourde (mais reste acceptable).
- **R** : un package R pourde piloter les analyses depuis R

# Langage Spark



- Sous Python, il n'est pas nécessaire de définir le type des variables avant de pouvoir les utiliser. Il suffit d'assigner une valeur à un nom de variable pour que celle-ci soit automatiquement créée

# Déploiement Spark



- Spark offre trois solutions
  - Spark standalone,
  - YARN (e gestionnaire de cluster Hadoop),
  - Mesos (plus genraliste  avec allocation des ressources plus fine)
- Nativement intégré dans les distributions :
  - Cloudera
  - Hortoworks

# Discussion?

# Bibliographie




- https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf
- http://b3d.bdpedia.fr/files/slmapred.pdf
- http://avc.cnam.fr/univ-r_av/avc/courseaccess?id=2418
- http://b3d.bdpedia.fr/files/Payen-Cassandra-2016.pdf
- http://b3d.bdpedia.fr/calculdistr.html#s3-systemes-iteratifs-spark
- http://aptuz.com/blog/is-apache-spark-going-to-replace-hadoop/
- http://zenika.developpez.com/tutoriels/nosql/integration-spark-cassandra/#LII-A
- http://cedric.cnam.fr/vertigo/Cours/RCP216/installationSpark.html
- http://spark.apache.org/docs/latest/quick-start.html
- https://jcrisch.wordpress.com/2015/01/12/comprendre-et-debuter-avec-apache-spark/
- http://www.stat4decision.com/fr/spark-boite-a-outils-du-big-data/
- http://javaetmoi.com/2015/04/initiation-apache-spark-en-java-devoxx/
- http://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-tutor4-pyspark-mllib.pdf
- https://jcrisch.wordpress.com/2015/10/23/prise-en-main-de-cassandra-avec-le-connecteur-spark-en-scala/
- https://jcrisch.wordpress.com/?s=spark
- https://fr.tuto.com/connexion/?redirect=

# Bibliographie suite




- http://zenika.developpez.com/tutoriels/nosql/integration-spark-cassandra/#LI-B
- http://docplayer.fr/25977017-Installation-ation-d-un-cluster.html
- https://hortonworks.com/wp-content/uploads/2013/03/InstallingHortonworksSandboxonWindowsUsingVMwarePlayerv2.pdf
- https://www.infoq.com/fr/articles/apache-spark-introduction
- https://www.infoq.com/fr/articles/apache-spark-sql
- https://hortonworks.com/wp-content/uploads/2013/03/InstallingHortonworksSandboxonWindowsUsingVMwarePlayerv2.pdf
- http://b3d.bdpedia.fr/mapreduce.html

# Fonctions importantes



- Cache → appelée pour stocker les RDD créés en cache, de façon à ce que Spark n'ait pas à les recalculer à chaque fois, à chaque requête suivante