IFRISSE

E-learning

Big Data

Master Informatique médicale et science des données

Animé par:

- GUIGUEMDÉ Rodrigue
- KALMOGO Roland

Plan du jour

- Hadoop, MapReduce, Spark & PySpark
- System local vs System distribué
- Écosystème de Hadoop
- Vue détaillée de Spark

Rappel: Big data

En général, nous utilisons des données qui conviennent à nos poste personnels: 0-8 GB juste la taille de nos RAM

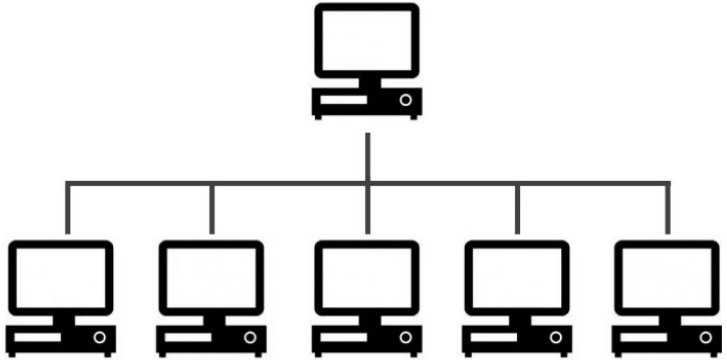
Que faire si nous désirons travailler avec données de taille plus grande (des TB ou TB) ?

Machines Distribuées

System local vs System distribué

Local:

Poste unique avec ses contraintes de resources



Distribué:

Machine maître contrôlant un ensemble de machine à travers un réseau

System local vs System distribué

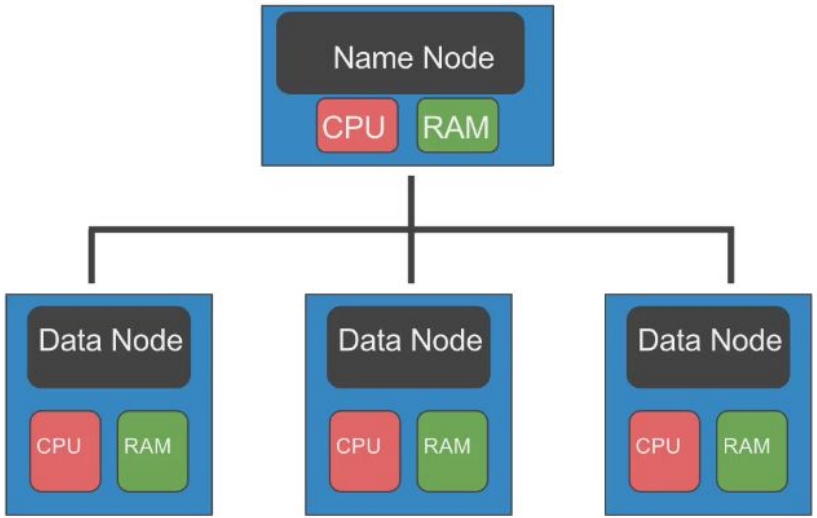
Local	Distribué
Ressources d’une seule machine	Ressource de plusieurs machines connectées
Difficile à étendre les capacités de la machine	Facile de combiner plusieurs machines pour atteindre les performances désirées
Non tolérant aux erreurs (Quand il y’a panne, Gros Problème)	Tolérances aux erreurs (Une machine peut avoir une panne mais le réseau est toujours fonctionnel)

Hadoop

- Architecture pour distribuer plusieurs fichiers entre plusieurs machines
- Utilisation HDFS (Hadoop Distributed File System)
- HDFS permet de travailler avec des données de taille énormes
- HDFS permet de dupliquer les blocs de données pour la tolérance aux erreurs
- HDFS permet d’utiliser MapReduce rendant possible les opérations sur les blocs de données

Stockage distribué: HDFS

- HDFS utilise des blocs de données de taille 128 MB (par défaut)
- Plus les blocs sont petits, plus on peut avoir beaucoup d'opération de parallélisation
- Chaque bloc est répliqué 3 fois
- Les blocs sont distribués dans les machines de données de façon à rendre possible la tolérance aux fautes/erreurs

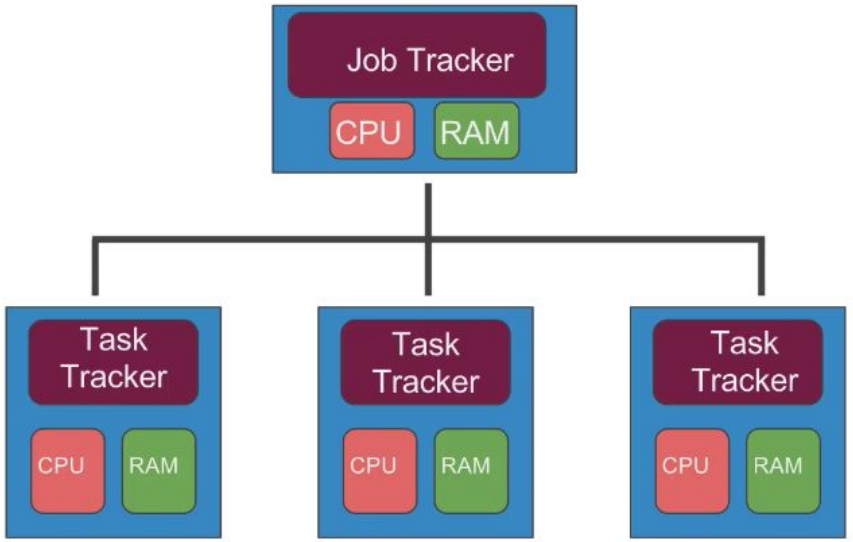


MapReduce

- MapReduce est une méthode/architecture qui permet de diviser une tâche de calcul en un ensemble de fichiers distribués (Comme le HDFS)
- Il est composé d’un Tracker de Job (job tracker) et de plusieurs trackers de tâches (task trackers)

MapReduce

- Job tracker envoie le code à exécuter aux task trackers
- Les tasks trackers allouent la mémoire le CPU pour les tâches et veillent sur elles sur les noeuds de travaux



Petit résumé

- Utiliser HDFS pour distribuer les données de taille énorme
- Utiliser MapReduce pour distribuer les opérations de calcul en des données distribuées

Spark est la dernière technologie adaptée pour ce types d’opérations

Questions?

Spark

- Dernière technologie utilisée pour rapidement et facilement gérer les données massives
- Open source (Familles des logiciel Apache)
- Très populaire dû à sa rapidité et à sa simplicité.
- La 1re version publiée en Février 2013
- Vu le jour dans un lab de l’université Berkeley Californie

Spark

- Une alternative de MapReduce
- Peut utiliser des données stockées dans une variété de format : Cassandra, AWS S3, HDFS, ...

Spark vs MapReduce

- MapReduce requiert que les fichiers soient stockés en HDFS, Spark non
- Spark est 100X plus rapide que MapReduce

Spark vs MapReduce

- MapReduce requiert que les fichiers soient stockés en HDFS, Spark non
- Spark est 100X plus rapide que MapReduce

Comment 100X plus rapide ?

- MapReduce stocke les données après chaque opération map et reduce.
- Spark garde les données en mémoire après chaque transformation
- Spark peut se retourner vers le disque dur si la mémoire est pleine

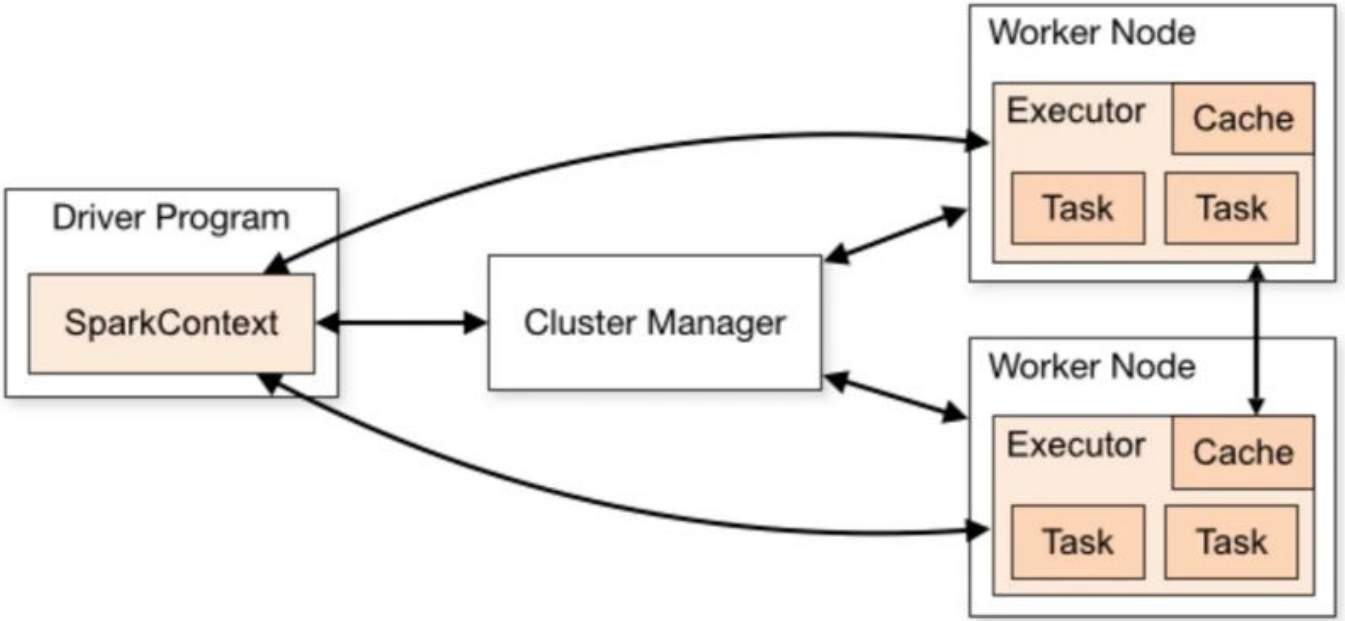
Spark RDDs

RDD (Resilient Distributed Dataset) est au coeur de la technologie Spark

Les RDDs ont 4 principales fonctionnalités:

- Données distribuées
- Tolérance aux fautes/Erreurs
- Parallélisation des opérations
- Habileté d’utiliser plusieurs source de données

Spark RDDs


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

Spark RDDs

Les RDDs sont immutables, peuvent être stockés dans la mémoire cache et évalués très rapidement

Deux types de RDDs

- Transformations
- Actions

Spark RDDs

- Actions de base:
  - First
  - Collect
  - Count
  - Take

Spark RDDs

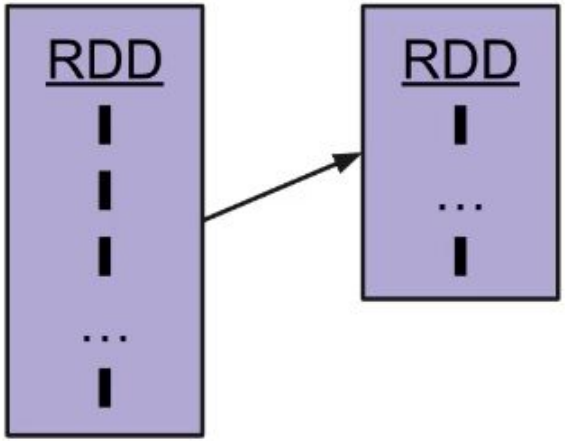
- Actions de base:
  - Collect: Retour tous les éléments du RDD comme un tableau
  - Count: Retourne le nombre d’éléments dans le RDD
  - First: Retourne le 1er élément du RDD
  - Take: Retourne sous forme de tableau, les n premiers éléments du RDD

Spark RDDs

- Transformations de base:
  - Filter
  - Map
  - FlatMap

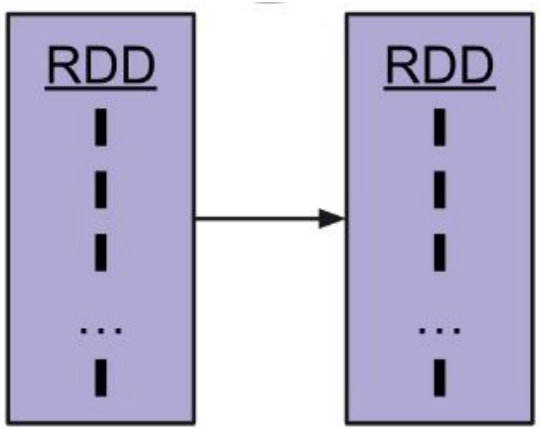
Spark RDDs

- Transformation de base:
  - RDD.filter: Applique une fonction à chaque élément et retourne les données dont le résultat de la fonction est “VRAI”



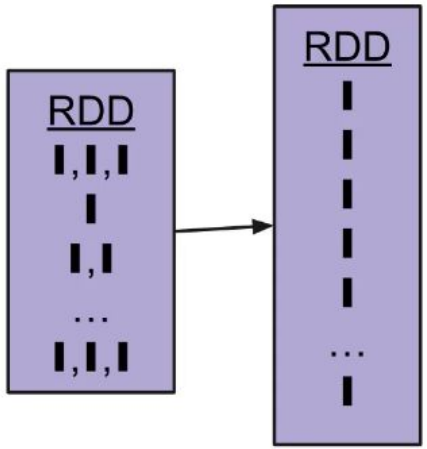
Spark RDDs

- Transformation de base:
  - RDD.map: Applique une fonction à chaque élément mais préserve le même nombre d’éléments avec chaque élément étant le résultat de la fonction sur ladite donnée



Spark RDDs

- Transformation de base:
  - RDD.flatMap: Transforme chaque élément en un ensemble d’éléments et change le nombre total d’éléments dans le RDD.



Ecosystem Spark

- Spark évolue très rapidement
- L'écosystème de Spark comprend:
  - Spark SQL
  - Spark Dataframes
  - MLlib
  - GraphX
  - Spark Streaming

Installation de Spark

- Il n’y a pas vraiment un grand intérêt à utiliser Spark en local
- Dans le context de big data, savoir mettre l’environnement distribué est important: AWS, Digital Ocean(DO)

Sur DO:

- https://m.do.co/c/91b74eb279b2
- Suivre les instructions de https://medium.com/@josemarcialportilla/getting-spark-python-and-jupyter-notebook-running-on-amazon-ec2-dec599e1c297