


Data Development

III — Distribution du calcul et du stockage : les projets Hadoop et Spark

 **Patrick S. Kanmeugne**

 SupInfo

© 2025

Plan(1)

1. Le contexte du Big Data : les 4 Vs
2. Le projet Apache Hadoop
3. Le projet Apache Spark
4. Ateliers
5. Bibliographie

Le contexte du Big Data ?

ce qui change par rapport à l'analyse de données classique

Big Data ~ La nature et la taille des données à analyser obligent à adapter les ressources (calcul et stockage) et à repenser les méthodes !

Les quatre Vs

Volume, Variété, Vitesse, Véracité

La *complexité* de l'analyse de données évolue simultanément selon :

- Volume → une grande quantité de données
- Variété (diversité) → plusieurs types de données
- Vitesse → besoin d'interactivité dans l'analyse
- Véracité → crédibilité des sources, qualité des données

Plan(2)

1. Le contexte du Big Data : les 4 Vs
2. Le projet Apache Hadoop
 - 2.1 Le modèle Map Reduce
 - 2.2 Le système de fichier HDFS
3. Le projet Apache Spark
4. Ateliers
5. Bibliographie

Le projet Hadoop

un projet de la fondation Apache

Hadoop Framework

- Framework Open-Source de «*Big Data Processing*» distribué
- Calculs distribués sur des clusters de plusieurs ordinateurs
- Système de fichiers adapté pour stockage distribué
- Programmes écrits suivant un modèle simple

Map/Reduce

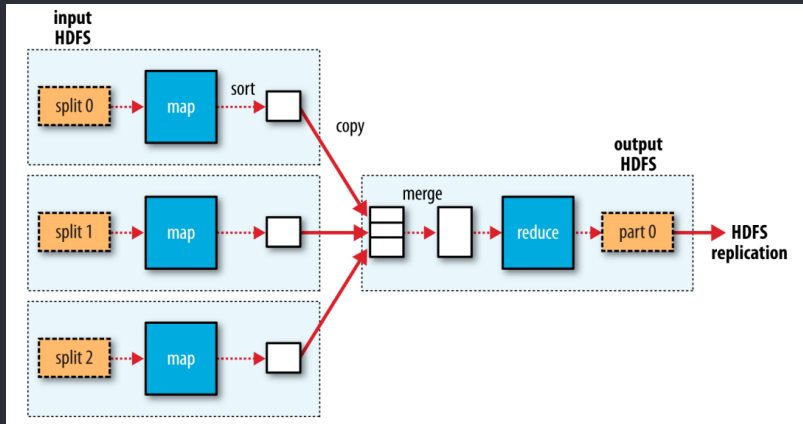
Un modèle de programme simple pour le framework Hadoop

Map/Reduce

- Modèle de programmation pour Hadoop
- 2 principales fonctions : *Map* et *Reduce*

Map/Reduce

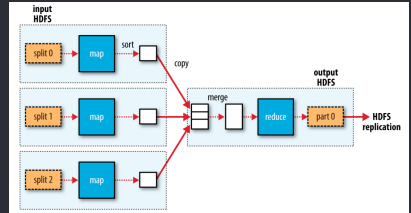
Architecture Map/Reduce [1] : avec une fonction «Map» et une fonction «Reduce»



Map/Reduce

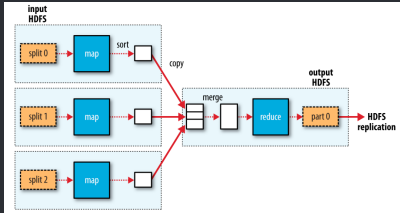
Map phase

- 1^{ère} phase du MapReduce
- Données d'entrée transformées en paires :
 < key, value >



Map/Reduce

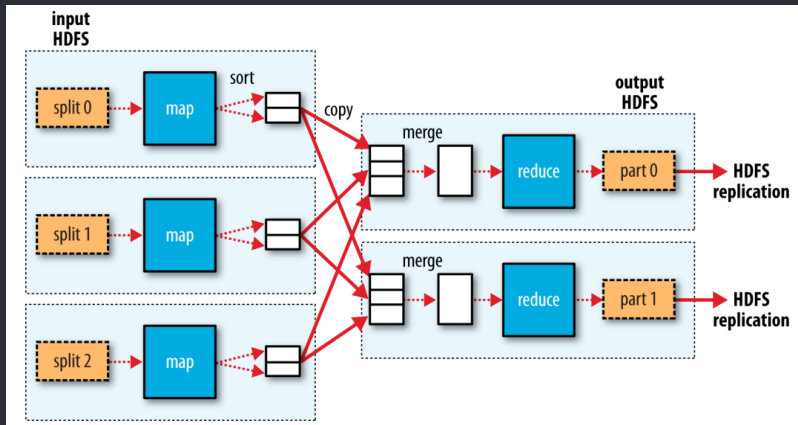
Reduce phase



- 2^{eme} phase du MapReduce
- les données $\langle key, value \rangle$ sont agrégées
- Production du résultat final

Map/Reduce

Architecture Map/Reduce [1] : avec ++ fonctions «Reduce»



Le système de fichier HDFS

Comprendre le système de fichier de Hadoop

HDFS : Hadoop Distributed FileSystem

- Système de fichier distribué pour des gros volumes de données
- Protocole d'accès i/o aux données à travers les clusters
- Compatible avec une infrastructure de stockage extensible à ∞

Le système de fichier HDFS

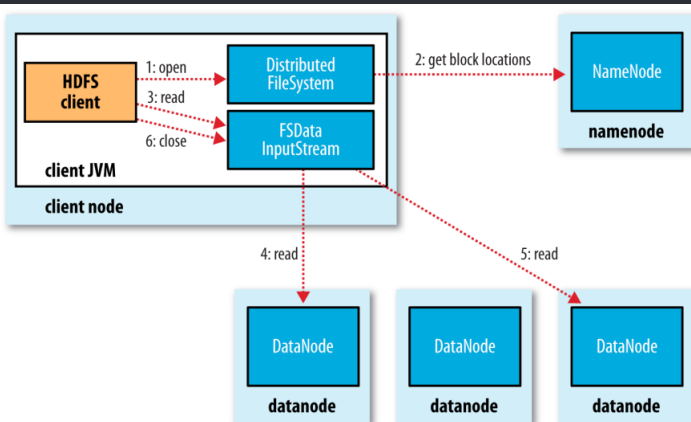
Les concepts

Hadoop : les concepts clefs

- *Namenode* – «master» : métadonnées, fichiers et répertoires
- *Datanode* – «slave» : transfert de blocks à la demande du «master»
- *Block* – unité de stockage (128 MB), répliqué $\geq 3 \times$

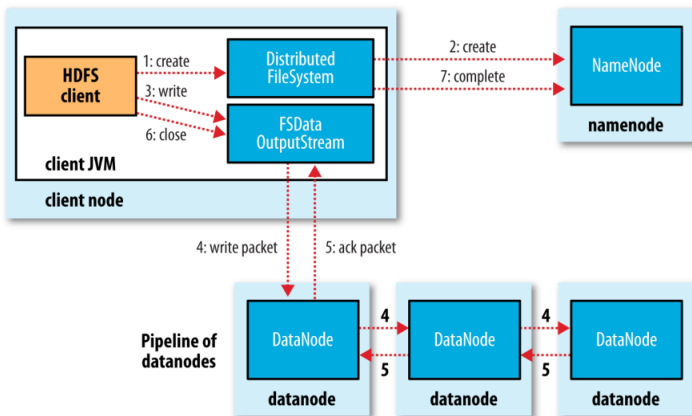
Le système de fichier HDFS

Lecture d'un fichier [1] : avec ++ fonctions «Reduce»



Le système de fichier HDFS

Écriture d'un fichier [1] : avec ++ fonctions «Reduce»



Le système de fichier HDFS

Les principales fonctionnalités

Hadoop HDFS : quelques fonctionnalités supplémentaires

- *High Availability* : duplication «namenodes» pour plus de robustesse
- *Block Caching* : mise en cache des «<blocks» les plus demandés
- *HDFS fédération* : répartition de l'administration entre «namenodes»

Plan(3)

1. Le contexte du Big Data : les 4 Vs
2. Le projet Apache Hadoop
3. Le projet Apache Spark
4. Ateliers
5. Bibliographie

Apache Spark : moteur de calcul distribué évolué

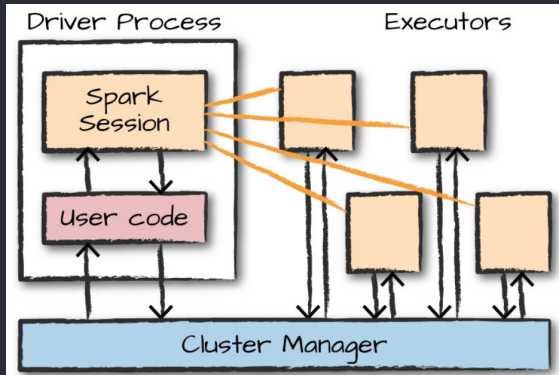
Matei Zaharia : créateur de Spark et co-fondateur de Databricks

Apache Spark : architecture basique

- *Cluster* : réseau d'ordinateurs dédié pour les calculs
- *Application Spark* : processus «pilote» et plusieurs «exécuteurs» qui manipulent des données
- *Exécuteur* : processus qui exécute une tâche allouée par le pilote et notifie
- *Dataframe* : collection de données distribuées organisées en colonnes ~ tables, feuilles de calcul

Apache Spark : Architecture basique d'une application Spark

Pilote, Application, Cluster, Exécuteurs (ou «workers») [2]



Autres concepts clefs

«*Lazy Evaluation*», «*Transformation*», «*Action*»

- *Lazy Evaluation* : le calcul est toujours différé jusqu'au dernier moment → orchestration optimale
- *Transformation* indique les manipulations à effectuer sur une collection de données
- *Actions* : déclenche l'exécution d'une série de transformations

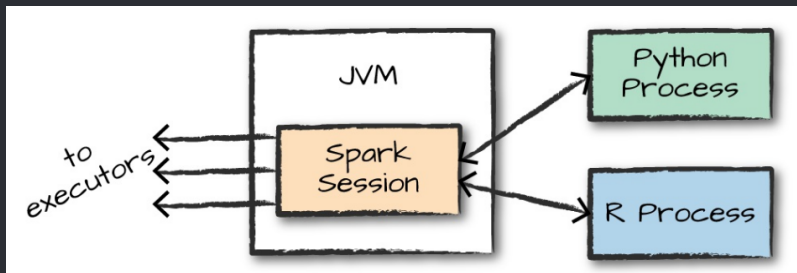
Spark API

Les API Spark existantes

- Scala : Spark est initialement écrit en Scala — le langage par défaut pour toutes applications Spark
- Java : compatibilité parfaite
- Python : compatibilité via la lib **PySpark**
- R : intégration de Spark via les lib **SparkR** et **SparklyR**

Les API Spark existantes

Spark via ses API [2]





Plan(4)

1. Le contexte du Big Data : les 4 Vs
2. Le projet Apache Hadoop
3. Le projet Apache Spark
- 4. Ateliers**
5. Bibliographie


Atelier : travail en groupe

Mini présentations 

Étudiez la question en groupe 

Contextualisez les challenges et mettez en perspective 

Présentation en 5 slides 

30' Maximum (préparation + présentation) 

Atelier : travail en groupe

Les sujets

1. Quel est l'avantage de l'utilisation des Dataframes dans Spark ~ aux méthodes traditionnelles ?
2. Dans quels sens les concepts d'«Action» et de «Transformation» diffèrent dans Spark – pourquoi cette distinction est importantes
3. Quels sont les principaux challenges associés aux systèmes distribués comme HDFS ~ systèmes de fichiers traditionnels ?
4. Expliquez le rôle du «Combiner» dans le modèle MapReduce et comment est-ce qu'il impacte les performances.

Plan(5)

1. Le contexte du Big Data : les 4 Vs
2. Le projet Apache Hadoop
3. Le projet Apache Spark
4. Ateliers
5. Bibliographie

Bibliographie

- [1] Tom White. *Hadoop The Definitive Guide, 4th Edition*. O'Reilly.
- [2] Bill Chambers and Matei Zaharia. *Spark : The Definitive Guide*. O'Reilly.