REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE BADJI MOKHTAR ANNABA
FACULTE DES SCIENCES DE L'INGENIORAT
DEPARTEMENT INFORMATIQUE

HADOOP MAPREDUCE

Master : Gestion et Analyse des Données Massives (GADM)

2^{eme}année

Dr. Klai Sihem

Avant propos...

Le Big Data est une science récente qui a surgie avec l'évolution et la variation des données et d'applications mises et échangées en ligne. Cette science consiste à prendre en charge d'une manière efficace un volume important des données hétérogènes en intégrant des techniques et outils nouveaux, vu que la technologie disponible ne répond plus aux besoins.
Ce polycopié est un support pédagogique qui permet d'initier l'étudiant au domaine des Big Data. Ce cours composé de plusieurs chapitres permet aux étudiants de comprendre la problématique et la motivation du domaine, et de maîtriser l'outil Hadoop avec le modèle MapReduce associé à ce domaine.

Chaque chapitre est élaboré pour répondre à un but pédagogique bien précis, se matérialisant par des explications, définitions accompagnées d'exemples et des illustrations par des figures suivies par des exercices, des solutions envisageables ou des fiches de travaux pratiques bien guidés.

1. Le chapitre I met l'étudiant dans le contexte du Big Data, consiste à lui donner des connaissances générales sur le domaine ;
2. Le chapitre II est consacré à l'étude de Hadoop, le framework qui permet le développement d'applications traitant les données massives. Ce chapitre donne les notions les plus générales avec la procédure d'installation du logiciel ;
3. Le chapitre III détaille la partie qui s'occupe du stockage des données "HDFS", avec la possibilité de la manipulation de ces données selon deux manières différentes à savoir : les commandes et l'API JAVA ;

4. Le chapitre IV étudie en détail la partie traitement des données massives "MapReduce", le modèle qui permet de traiter des blocs de données séparément et parallèlement dans des machines connectées. La modélisation selon le paradigme MapReduce est une étape importante avant le développement des programmes ;
5. Le chapitre V détaille l'implémentation des programmes MapReduce dans Hadoop. Dans le cadre de ce chapitre, nous étudions l'implémentation des programmes en utilisant le langage Java. D'autres langages peuvent être utilisés pour écrire des programmes mapreduce, mais cette partie n'est pas traiteé dans ce cours.

L'élaboration de ce polycopié a été inspirée de plusieurs documents, j'ai pris le soin de les citer dans la partie bibliographie, d'autres documents aussi ont été cité afin d'apporter aux lecteurs plus de commandes et plus de détails sur les parties traitées dans ce cours.

Terminologie. . .

JVM : Java virtuelle machine

HDFS : Hadoop Distributed File System

YARN : Yet Another Ressource Negociator

AM : Application Master

API java : Application Programming Interfaces java

Table des matières

Liste des figures

1 HDFS : Hadoop Distributed File System

Objectif Pédagogique

A la fin de ce chapitre, l'étudiant maîtrisera le fonctionnement de la partie "stockage des données de Hadoop (HDFS)". Il connaîtra :

- La façon dont sont organisés les fichiers dans HDFS ;
- La façon dont sont organisées les machines ;
- la manipulation des données HDFS en utilisant les commandes ou via des programmes java.

Pré-requis

L'étudiant doit avoir :

- Des connaissances sur le Système de Gestion des fichiers classiques ;
- Les bases du langage Java ;
- Installer Hadoop et maîtriser ses concepts.

HDFS (Hadoop Distributed File System) est un système de fichiers distribué, c'est la partie qui se charge du stockage et d'accès aux données. Il permet le stockage distribué des fichiers de très grande taille.

1.1 Organisation des fichiers dans HDFS

Dans HDFS, les fichiers sont organisés de la façon suivante : 10

— Ils sont stockés sur un grand nombre de machines de manière à rendre invisible la position exacte et l'accès d'un fichier ;

— Physiquement, les fichiers sont découpés en blocs de grande taille (jusqu'au 256Mo), par défaut elle est égale à 64 Mo mais elle est configurable. La taille d'un bloc dans HDFS est nettement plus importante que la taille d'un bloc sur les systèmes classiques. L'intérêt de fournir des tailles plus grandes permet de réduire le temps d'accès à un bloc ;

— Selon la taille d'un fichier, il lui faudra un certain nombre de blocs. Sur HDFS, le dernier bloc d'un fichier fait la taille restante. Les blocs sont numérotés et chaque fichier sait quels blocs il occupe (voir la figure 1.1).

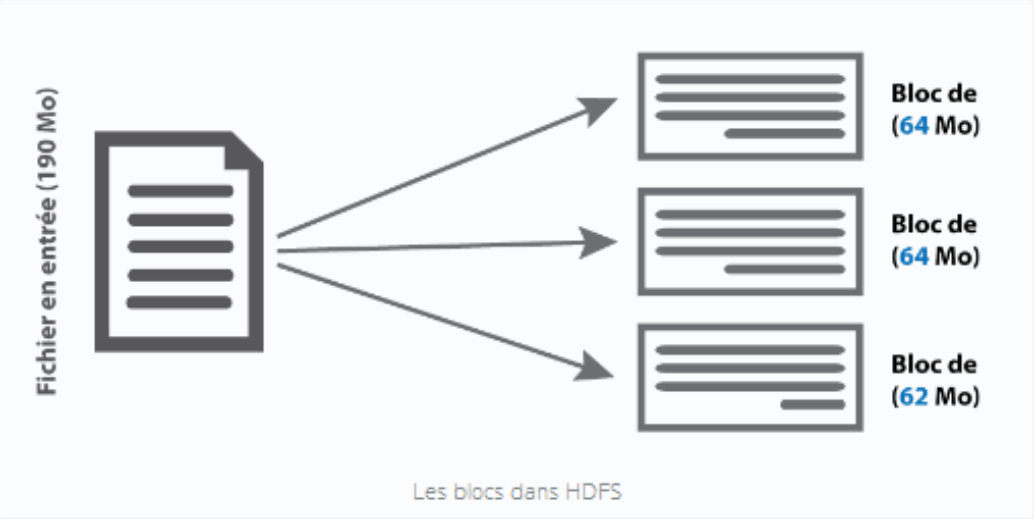

FIGURE 1.1 – *Exemple de découpage d'un fichier en blocs* 04

— Ces blocs sont ensuite répartis sur plusieurs machines, permettant de traiter un même fichier en parallèle. Cela permet de ne pas se limiter

par la capacité de stockage d'une seule machine et tirer parti de tout l'espace disponible du cluster de machines ;

— Pour garantir une tolérance aux pannes, les blocs de chaque fichier sont répliqués, de manière intelligente, sur plusieurs machines ;

— HDFS permet de voir tous les dossiers et fichiers de l'ensemble des machines du cluster comme un seul arbre, comme s'ils étaient sur le disque dur local.

1.2 Organisation des machines et Fonctionnement de HDFS

Le cluster HDFS est constitué de machines jouant différents rôles exclusifs entre eux selon une architecture maître-esclave de stockage distribué des données [1] :

1. Le NameNode (nœud maitre ou master node) : Dans un cluster Hadoop, le NameNode est la machine ou le nœud qui gère l'espace de noms, l'arborescence du système de fichiers et les métadonnées des fichiers et des répertoires :
   — Il centralise la localisation des blocs de données répartis dans le cluster. Quand un client sollicite Hadoop pour récupérer un fichier, c'est via le NameNode que l'information est extraite. Ce nœud va indiquer au client quels sont les DataNodes qui contiennent les blocs ;
   — Le NameNode reçoit régulièrement un rapport de bloc (Block report) de tous les DataNodes dans le cluster afin de s'assurer que les DataNodes fonctionnent correctement. Le rapport de bloc contient une liste de tous les blocs d'un DataNode .
   — En cas de de défaillance du DataNode , le NameNode choisit de nouveaux DataNodes pour de nouvelles réplications de blocs de données, équilibre la charge d'utilisation des disques et gère également le trafic de communication des DataNodes.

1. http://www.opentuto.com/fonctionnement-de-hdfs/

2. Le Secondary NameNode effectue des tâches de maintenance pour le compte du NameNode. Plus précisément, il met à jour le fichier "journalNode" à intervalles réguliers (par exemple une fois par heure).
3. Le DataNode : les datanodes contiennent des blocs, les mêmes blocs sont dupliqués sur différents datanodes, par défaut trois fois mais, c'est configurable. Cela assure :
   — Une fiabilité des données en cas de panne d'un datanode ;
   — Un accès parallèle par différents processus aux mêmes données.

La figure 1.2 illustre le fonctionnement de HDFS selon l'architecture maître-esclave.

Explication du schéma :

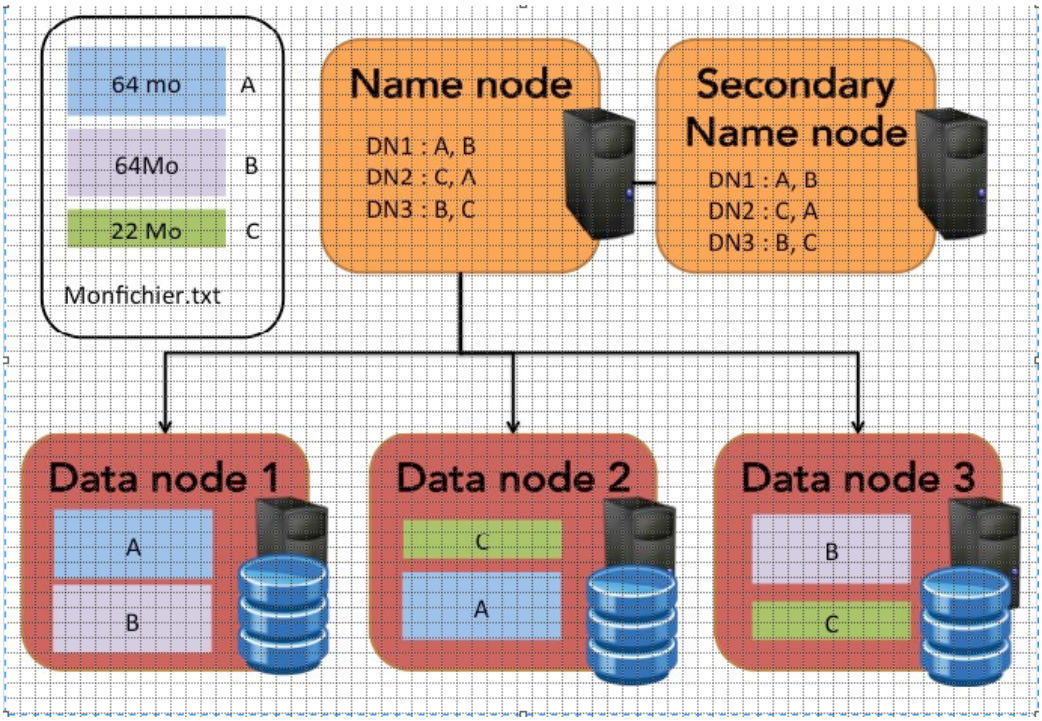

FIGURE 1.2 – *Les blocs en HDFS 06*

1. Le cluster est composé d'une machine Namenode, une machine Secondary Namenode et Trois machines DataNode ;
2. Le fichier "Monfichier.txt" est découpé en trois blocs (notés A,B,C) ;
3. Les datanodes contiennent des blocs (A,B,C). Les mêmes blocs sont dupliqués sur deux datanodes distincts ;

4. Le NameNode contient les métadonnées c.à.d les informations sur les blocs d'un fichier (référence bloc, son emplacement dans le DataNode, ...) ;
5. Le scondary NameNode contient une copie du contenu du NameNode.

1.3 Manipuler HDFS

Il ya deux possibilités pour manipuler HDFS :

— Soit directement depuis un terminal via les commandes ;

— Soit via l'API Java ;

1.3.1 Manipulation de HDFS via des commandes

Dans cette section, nous citons quelques commandes fournies par Hadoop, nous conseillons l'étudiant de consulter la documentation officielle de Hadoop pour plus de commandes et de détail [2]. Les étudiants sont amenés à les tester (c'est l'objet du TPN°3 en Annexes)

La commande hdfs dfs

La commande hdfs et ses options permet de gérer les fichiers et les dossiers :

hdfs dfs -ls [noms...] (pas d'option -l) : permet d'afficher le contenu d'un dossier.

hdfs dfs -cat nom : permet d'afficher le contenu d'un fichier.

hdfs dfs -mv ancien nouveau : permet de déplacer un fichier d'un répertoire à un autre.

hdfs dfs -cp ancien nouveau : fait la copie d'un fichier.

hdfs dfs -mkdir nomdossier : permet de créer un dossier.

hdfs dfs -rm -f -r nomdossier : permet de supprimer un dossier.

2. Apache Hadoop//hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

Echange entre HDFS et le monde

Pour placer un fichier dans HDFS (du fichier source "fichiersrc" au fichier destinataire "fichierdst"), deux commandes équivalentes :

hdfs dfs -copyFromLocal fichiersrc fichierdst

hdfs dfs -put fichiersrc [fichierdst]

Pour extraire un fichier de HDFS ("fichiersrc" au fichier destinataire "fichierdst"), deux commandes possibles :

hdfs dfs -copyToLocal fichiersrc dst

hdfs dfs -get fichiersrc [fichierdst]

Exemple 1.1 hdfs dfs -mkdir madoc *: création d'un dossier "madoc"*

hdfs dfs -put poly.pdf madoc *: place le fichier dans le dossier "madoc"*

hdfs dfs -ls madoc *: permet d'afficher le contenu du dossier "madoc"*

hdfs dfs -get poly.pdf/mapreduce.pdf *: permet de copier le fichier "poly.pdf" vers le fichier "mapreduce.pdf"*

1.3.2 Manipulation de HDFS via l'API Java

Hadoop propose une API Java complète pour accéder aux fichiers de HDFS. Elle repose sur deux classes principales 10 :

- FileSystem représente l'arbre des fichiers (file system). Cette classe permet de copier des fichiers locaux vers HDFS (et inversement), renommer, créer et supprimer des fichiers et des dossiers ;
- FileStatus gère les informations d'un fichier ou dossier tels que : récupérer la taille avec getLen() .

Ces deux classes ont besoin de connaître la configuration du cluster HDFS, à l'aide de la classe Configuration. D'autre part, les noms complets des fichiers sont représentés par la classe Path.

Exemple 1.2 *Exemple de programme de manipulations sur un fichier : Il s'agit de renommer le fichier "test.txt" en "monFichier.txt"*

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.Path;
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
Path nomP = new Path("/user/projet", "test.txt");
FileStatus infos = fs.getFileStatus(nomP);
System.out.println(Long.toString(infos.getLen()));
fs.rename(nomP, new Path("/user/projet","monFichier.txt"));
```

1.4 Ecriture et Lecture des données dans HDFS

1.4.1 Ecriture d'un fichier de données dans HDFS

Si on souhaite écrire un fichier dans HDFS, on utilise un client Hadoop, figure 1.3 04, 05, 09 :

1. Le client indique au name node qu'il souhaite écrire un bloc ;
2. Le name node indique le data node à contacter ;
3. Le client envoie le bloc au data node ;
4. Les data nodes répliquent les blocs entre eux ;
5. Le cycle se répète pour le bloc suivant.

Exemple 1.3 *Exemple de programme qui montre comment créer un fichier HDFS "donnee.txt", il contiendra le mot "MapReduce" :*

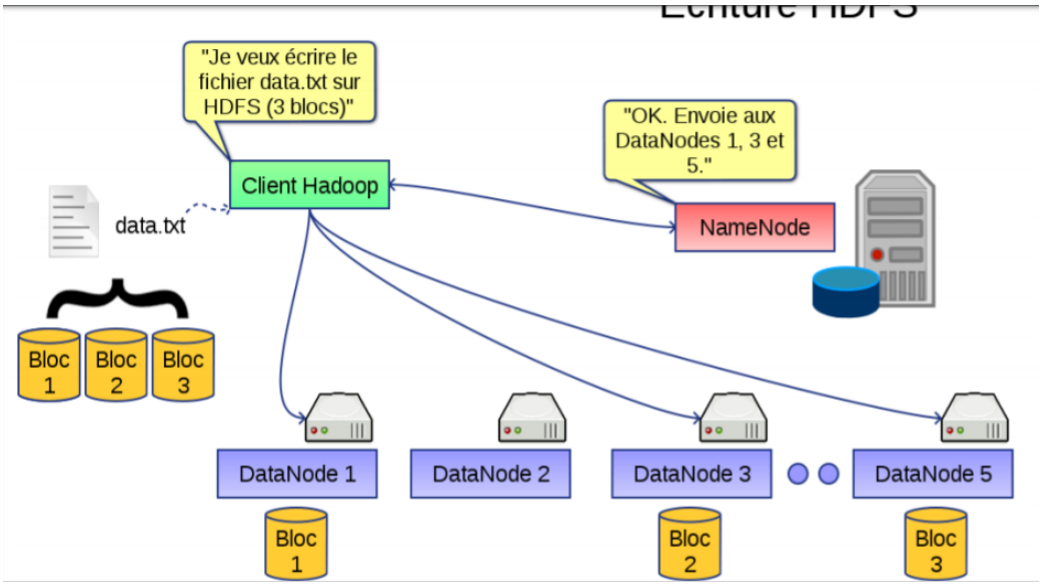

FIGURE 1.3 – *Ecriture d'un fichier HDFS*

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.Path;
public class HDFSwrite {
public static void main(String[] args) throws IOException {
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
Path nomP = new Path("donnee.txt");
if (! fs.exists(nomP)) {
FSDataOutputStream outStream = fs.create(nomP);
outStream.writeUTF("MapReduce ");
outStream.close();}
fs.close();}}
```

1.4.2 Lecture d'un fichier de données dans HDFS

Si on souhaite lire un fichier dans HDFS, figure 1.4 04, 05, 09 :

1. Le client indique au name node qu'il souhaite lire un fichier ;

2. Le name node indique sa taille ainsi que les différents data nodes contenant les blocs ;
3. Le client récupère chacun des blocs sur l'un des data nodes ;
4. Si le data node est indisponible, le client en contacte un autre.

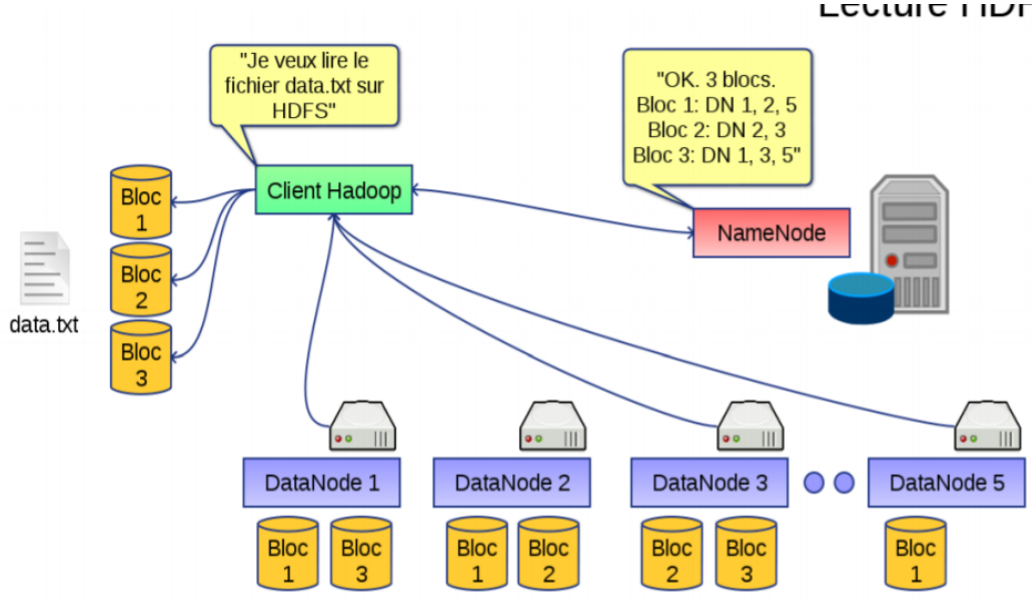

FIGURE 1.4 – *Lecture d'un Fichier HDFS*

Exemple 1.4 *Exemple de programme qui permet de lire le fichier texte "donnee.txt" :*

```
import java.io.*;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.Path;
public class HDFSread {
public static void main(String[] args) throws IOException {
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
Path nomP = new Path("donnee.txt");
FSDataInputStream inStream = fs.open(nomP);
InputStreamReader isr = new InputStreamReader(inStream);
BufferedReader br = new BufferedReader(isr);
String line = br.readLine();
System.out.println(line);
inStream.close();
fs.close();
}}
```

Annexes

A

Bibliographie

[01]. site : Le big data, 2019. URL https://www.lebigdata.fr/definition-big-data.

[02]. Stephane Vialle, CentraleSuplec. Big Data : Informatique pour les donnees et calculs massifs. 24 mai 2017 .

[03]. mohammed zuhair al taie. hadoop ecosystem : an integrated environment for big data. in big data, guest posts 2015. URL http://blog.agroknow.com/?p=3810. (Cité page 7.)

[04]. site : Hadoop. 2016. URL http://www.opentuto.com/category/web-2/big-data/hadoop/. (Cité pages 2, 7 et 8.)

[05]. amal abid. cours big data : Chapitre2 : Hadoop. 2017. URL https://fr.slideshare.net/AmalAbid1/cours-big-data-chap2. (Cité pages 7 et 8.)

[06]. celine hudelot, regis behmo. realisez des calculs distribue sur des donnees massives. 2019. URL https://openclassrooms.com/fr/courses/4297166-realisez-des-calculs-distribues-sur-des-donnees-massives. (Cité pages 7 et 4.)

[07]. Benaouda, Sid Ahmed Amine, implantation du modele mapreduce dans l'environnement distribue, 2015 . URL http://dspace.univ-tlemcen.dz.

[08]. Tom White, Hadoop : The Definitive Guide, June 2009 : First Edition.

[09]. Srinath Perera, Thilina Gunarathne, Hadoop MapReduce Cookbook, First published : February 2013. (Cité pages 7 et 8.)

[10]. Pierre Nerzic, Outils Hadoop pour le BigData, mars 2018. (Cité pages 2 et 6.)

[11]. marty hall. map reduce 2.0 (input and output). 2013. URL https://www.slideshare.net/martyhall/hadoop-tutorial-mapreduce-part-4-input-and-output.

[12]. build projects, learn skills, get hired. hadoop mapreduce- java-based processing framework for big data. URL https://www.dezyre.com/hadoop-course/mapreduce.

[13]. URL https://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/mapreduce/.

[14]. URL https://gist.github.com/kzk/712029/9d0833aac03b23ec226e034d98f5871d9580724e.

[15]. Jeffrey Dean and Sanjay Ghemawat, MapReduce : Simplified Data Processing on Large Clusters, 2004 .

[16]. Univ. Lille1- Licence info 3eme annee, Cours nÂ°1 : Introduction Ã la programmation fonctionnelle, 2013-2014 .

[17] dataflair team. hadoop architecture in detail hdfs,yarn et mapreduce. 2019. URL https://data-flair.training/blogs/hadoop-architecture/.

[18] vlad korolev. hadoop on windows with eclipse. 2008. URL http://v-lad.org/Tutorials/Hadoop/.

[19]. Donald Miner and Adam Shook. MapReduce Design Patterns. OReilly, 2012.

[20]. Tom White. Hadoop : The Definitive Guide, 4th Edition. OReilly, 2015.

[21]. Hadoop Training. 2018. URL https://www.slideshare.net/AnandMHadoop/session-19-mapreduce.