

Projet Fin d'étude 1ere version

Amini Makhlouf

Ben Aissa Tinhinane

2020/2021

Table des matières

I	État de l’art	3
II	Conception	4
III	Réalisation	5
1	Choix des outils technologiques	6
1.1	Hadoop :	7
1.1.1	Installation d’Hadoop :	8
1.2	Hbase :	12
1.2.1	Installation d’Hbase :	12
1.3	Langages et outils de développement	13
1.3.1	Javascript	13
1.3.2	vue	13
1.3.3	nodes	13
1.3.4	express js	13

Table des figures

1.1	Technologies utilisé	6
1.2	Logo Hadoop	7
1.3	Logo Java 8	8
1.4	Les fichier Hadoop	9
1.5	Modifier le fichier core-site.xml	9
1.6	Cr��er les deux fichiers dans le dossier data	9
1.7	Modifier le fichier hdfs-site.xml	10
1.8	Modifier le fichier yarn-site.xml	10
1.9	Modifier le fichier hadoop-env.cmd	10
1.10	<i>HADOOPHOME</i>	11
1.11	<i>JAVAHOME</i>	11
1.12	Logo Hbase	12
1.13	modifier le fichier hbase-config.cmd	12
1.14	Modifier le fichier hbase-site.xml	12

Première partie

État de l'art

Deuxième partie

Conception

Chapitre 1

Hbase

Introduction

Face à l'essor du Big Data, de nombreuses entreprises doivent désormais traiter des quantités massives de données. Dans ce contexte, les bases de données d'autrefois ne peuvent plus répondre à leurs besoins. Le volume de données à traiter est tel que les requêtes excèdent bien souvent les capacités d'un seul serveur. De même, les bases de données SGBDR ne peuvent prendre en charge des quantités massives de lectures et d'écritures.

Pour dépasser ces limites, de nouveaux SGBD dit "NoSQL" ont vu le jour. La particularité de ceux-ci est qu'ils n'imposent pas de structure particulière aux données, ils relâchent les contraintes qui empêchent les SGBDR de distribuer le stockage des données et sont linéairement scalables. HBase fait partie de cette catégorie de SGBD.

Nous présenterons dans ce chapitre Apache Hbase afin de montrer ses avantages et inconvénients et la façon dont ils gère cette masse de données qu'est le big data.

1.1 Apache Hbase

1.1.1 Apache HBase : qu'est-ce que c'est ?

Apache Hbase est une base de données non relationnelle (NoSQL), reprenant le concept et les fonctionnalités de Google BigTable¹. La différence est que HBase est Open Source. Cette base de données s'exécute généralement sur le système de fichiers distribués Hadoop (HDFS). Elle offre un accès d'écriture et de lecture en temps réel, aléatoire et cohérent, à des tables contenant des milliards de lignes et des millions de colonnes.



FIGURE 1.1 – Logo Hbase

Elle permet aussi combiner des sources de données reposant sur différentes structures et différents schémas. Il s'agit donc d'un très bon choix pour le stockage de données multi-structurées. Il est aussi possible d'effectuer des requêtes pour un repère temporel spécifique, ce qui permet de réaliser des requêtes "flashback"².

Nativement intégrée avec Hadoop, HBase fonctionne avec d'autres moteurs d'accès aux données par le biais de YARN. Son langage de programmation est le Java.

Apache HBase reprend toutes les fonctionnalités de Google BigTable. On retrouve ainsi les filtres Bloom, ou encore les opérations et la compression in-memory. Les tables de cette base de données peuvent servir d'entrée pour les tâches MapReduce dans l'écosystème Hadoop, et peuvent aussi servir de sortie après traitement des données par MapReduce.

Concrètement, HBase est un data store key-value orienté colonne. Il peut donc fonctionner avec toutes les données traitées par Hadoop. Il n'est pas possible de l'utiliser pour remplacer une base de données SQL, mais il est possible d'ajouter une couche SQL à cette data base pour l'intégrer avec les différents outils de Business Intelligence et d'analyse de données.

1. **Bigtable** est un système de gestion de base de données compressées, haute performance, propriétaire, développé et exploité par Google¹. C'est une base de données orientée colonnes, dont se sont inspirés plusieurs projets libres, comme HBase, Cassandra ou Hypertable.

2. **Bigtable**Le principe du FlashBack Query consiste à requêter une table via une clause SELECT + une clause "AS OF" permettant de préciser à quelle date (Time Stamp) ou à quel Numéro SCN on souhaiterait voir l'image de la table.

1.1.2 Concepts de base :

La base de données HBase est régie par les concepts suivants :

- **Map** : Le stockage se fait dans une map. Cette dernière est basée sur le principe de clé/valeur. Chaque valeur (tableau de bytes) est identifiée par une clé (tableau de bytes). L'accès à une valeur par sa clé est très rapide.
- **Map trié** : La map est triée par ordre lexicographique. Cette fonctionnalité de tri est très importante car elle permet de récupérer les valeurs par intervalle de clés.
- **Multidimensionnel** : La clé dans la map est une structure composée de row-key, column family, column, et d'un timestamp.
- **Multidimensionnel** : La clé dans la map est une structure composée de row-key, column family, column, et d'un timestamp.

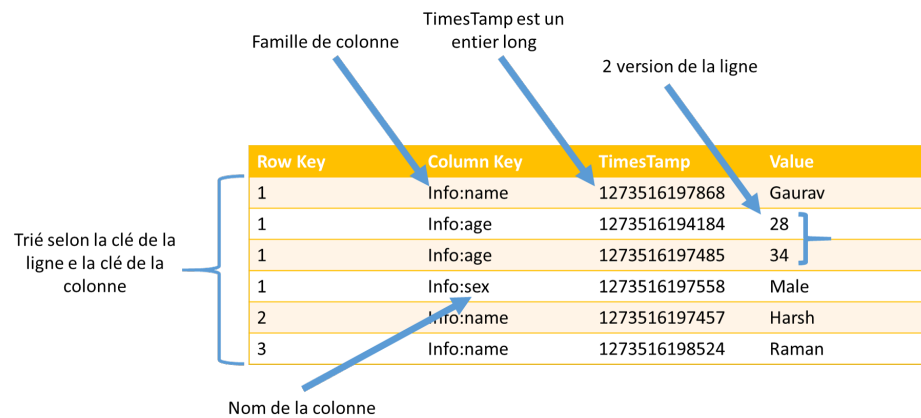


FIGURE 1.2 – Structure d'une clé dans la map

- **Null (Sparse)** : Contrairement aux bases de données relationnelles, une colonne qui n'a pas de valeur n'est pas matérialisée (aucun stockage n'est nécessaire en cas d'une valeur null pour une colonne).
- **Persistence** : Les données stockées dans la map sont sauvegardées durablement sur disque.
- **Consistance** : Toutes les modifications sont atomiques et les lectures se font toujours sur la dernière valeur validée (commit).
- **Système distribué** : Le système de base de données est construit sur un système de fichiers distribués afin que le stockage de fichiers sous-jacent soit repartitionné sur un ensemble de machines d'un cluster. Les données sont répliquées sur un certain nombre de nœuds permettant ainsi une tolérance aux pannes.

1.1.3 Architecture et fonctionnement

HBase est un SGBD distribué et en tant que tel, il s'installe sur un cluster d'ordinateurs. Comme Hadoop, HBase s'installe sur un cluster en architecture Maître/Esclave. Dans la terminologie HBase, le nœud maître s'appelle le HMaster, et les nœuds esclaves s'appellent les RegionServers. Le stockage des données est distribué sur les RegionServers qui sont gérés par le HMaster. Le HMaster gère les métadonnées des tables HBase et coordonne l'exécution des activités des RegionServers, tandis que les RegionServers effectuent les opérations de lecture/écriture de données dans le cluster. La gestion du volume de données se fait par l'ajout des RegionServers supplémentaires dans le cluster. La figure ci-après illustre l'architecture d'un cluster HBase.

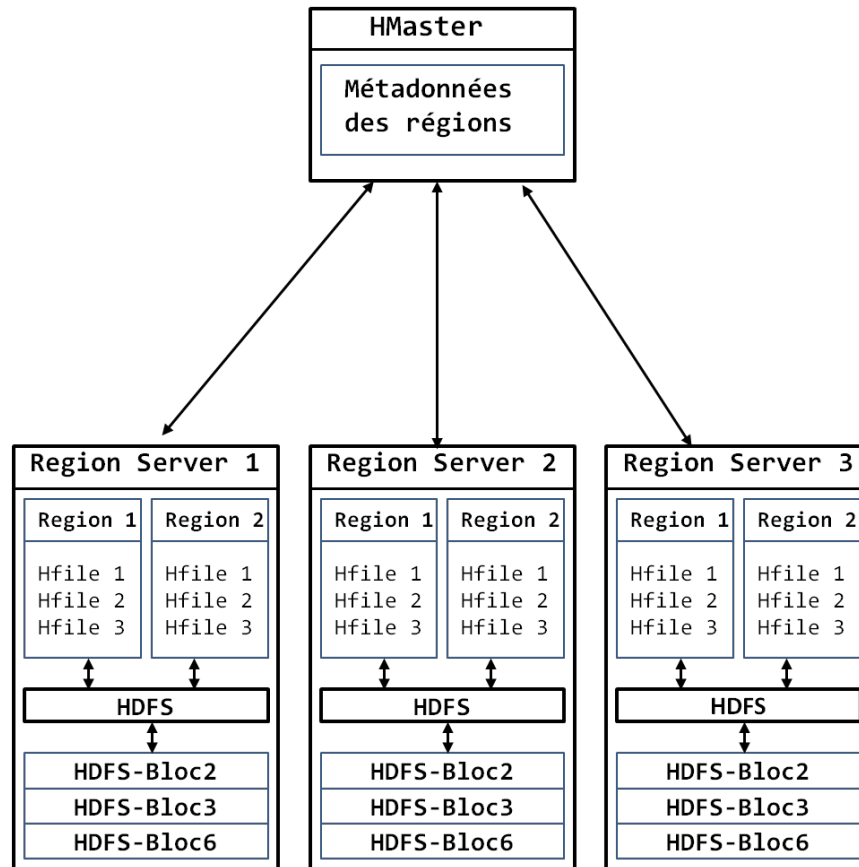


FIGURE 1.3 – Architecture d'un cluster HBase.

Comme vous le savez déjà, les tables HBase sont persistées sur le disque sous forme de fichiers HDFS appelés HFiles. Chaque HFile contient les données d'une et une seule famille de colonnes. Toutes les lignes de la table sont identifiées de façon unique à l'aide d'une valeur de la row key. Étant donné que la table fait office de base de données, pour une application métier donnée, toutes les données sont stockées dans la seule table HBase, qui pourra alors rapidement contenir des milliards de lignes, chiffrant sa taille en Téra octets voir Péta-octets. Cette taille phénoménale rend impossible le stockage de la table sur une seule machine. Pour résoudre ce problème, les tables HBase sont divisées en partitions appelées "régions" qui sont réparties entre les nœuds RegionServers pour le stockage. La taille de chaque région peut être paramétrée dans un fichier de configuration hbase-site.xml. Lorsque la taille d'une région dépasse la taille maximale que vous avez définie dans le fichier de configuration, la région se partitionne automatiquement en deux. La région est une partition de la table triée par valeurs de la row key. La table étant déjà physiquement partitionnée en HFiles, une région sera physiquement persistée sous forme d'un ou plusieurs HFiles. Les régions forment l'unité de stockage en HBase et sont la clé de la distribution du stockage et de la scalabilité du cluster HBase. La figure suivante illustre la façon dont HBase partitionne les tables en régions.

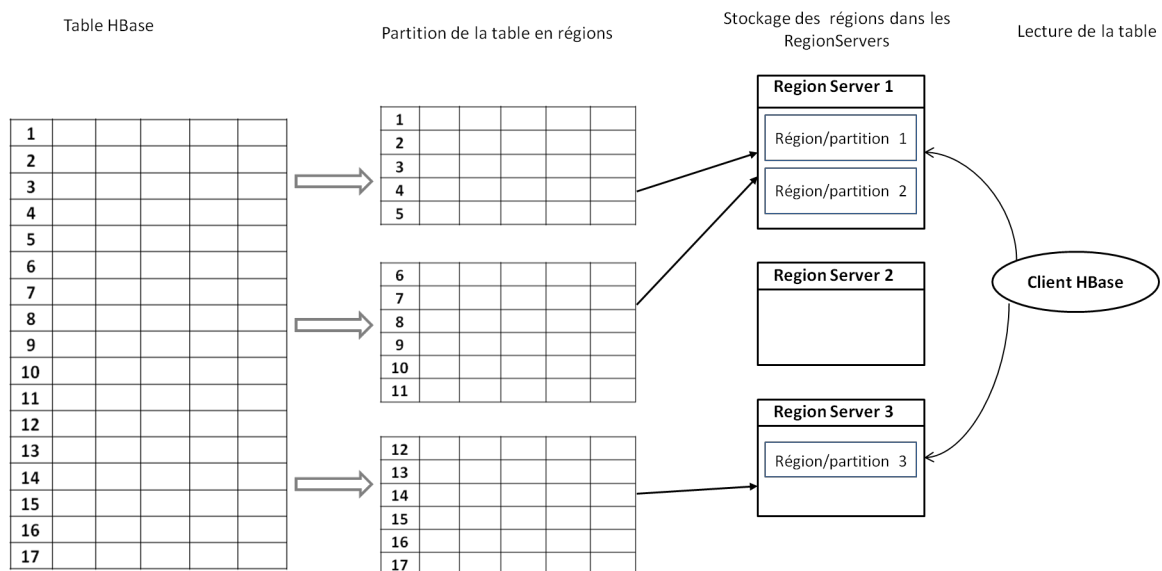


FIGURE 1.4 – Partition des tables HBase.

Les régions sont distribuées sur le cluster de façon aléatoire et chaque nœud RegionsServer peut stocker une ou plusieurs régions. Celle-ci sont répliquées entre les nœuds de façon à maintenir la disponibilité du cluster en cas de panne. Lors de l'ajout d'une ligne existante dans une table (nouvelle version de la ligne), HBase retrouve la région contenant la valeur de la row key de la ligne et l'insère dans cette région. Pour retrouver la région contenant la row key, HBase utilise une table de catalogue spéciale appelée "hbase :META". Cette table contient la liste des RegionsServers disponibles, et la liste des intervalles de valeurs de row key pour chaque région de table. Elle est stockée dans un composant de l'écosystème Hadoop appelé ZooKeeper, qui tourne sur un cluster différent du cluster sur lequel est installé Hbase. ZooKeeper est nécessaire parce qu'à la différence d'Hadoop où la communication entre le client et le cluster se fait à l'intermédiaire du nœud de référence, dans HBase, le client communique directement avec les nœuds RegionsServer sans passer par le HMaster. Le client n'a donc aucun moyen de connaître dans quel RegionsServer sont situées les données dont il a besoin. Lorsqu'un client fait une requête sur une row key précise, ZooKeeper pointe vers la table hbase : META pour récupérer les informations de la région contenant la row key, ensuite ZooKeeper renvoie cette information au client, qui va alors directement s'adresser au RegionsServer contenant la région. Finalement, la RegionsServer va traiter la requête et renvoyer au client les données de la row key. La figure ci-après illustre le fonctionnement d'HBase lors d'une opération de lecture de données.

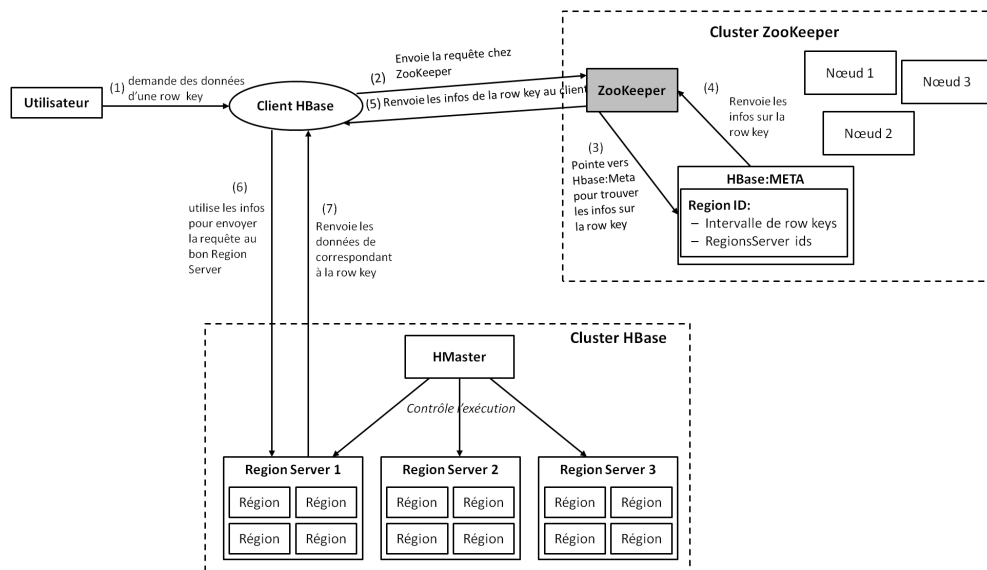


FIGURE 1.5 – Fonctionnement de HBase.

Remarque : Ce mode de fonctionnement est effectif uniquement à partir de la version 0.96 d'HBase. Antérieurement à cette version, HBase utilisait deux tables de catalogue pour gérer les références de régions : .META. et -ROOT-. .META. contenait la liste des RegionsServers disponibles, la liste des partitions dans chaque RegionsServer et leurs valeurs de row key. .META. était partitionnée en régions et lorsqu'un client envoyait une requête sur une row key précise, la requête n'accédait pas directement à la table .META., mais elle accédait à la table -ROOT-. La table -ROOT- était gérée par ZooKeeper et pointait vers la liste des régions .META. À partir de la version 0.96, la table -ROOT- a été supprimée et la table .META. a été remplacée par la table hbase :META actuelle. La table hbase :META n'est désormais plus gérée par HBase, mais par ZooKeeper.

1.1.4 Avantage de HBase

Les avantages de HBase sont nombreux, en plus de ceux que nous avons énoncé dans le chapitre précédent nous citons :

- **Assure l’extensibilité et la calculabilité.**
- **Importante tolérance aux pannes :** Les données sont répliquées sur les différents serveurs du Data Center, et un failover (basculement) automatique garantit une haute disponibilité.
- **Rapidité :** Il propose des lookups en temps réel ou presque, et un processing server side³ par le biais de filtres et de coprocesseurs. Un caching in-memory est également proposé.
- **Distribution transparente de la donnée :** Répartition de la charge est faite par le système lui même.
- **Les langages de programmation supportés :** Hbase supporte beaucoup de langage de programmation : C, C#, C++, Groovy, Java, PHP, Python, Scala.
- **Utile pour les données de capteurs :** HBase est très adapté aux données collectées de façon incrémentielle a partir de diverses sources. Cela inclut l’analytique sociale, les scieries chronologiques, la mise a jour des tableaux de bord interactifs avec les tendances et les compteurs, et la gestion de systèmes de journal d’audit.

1.1.5 Inconvénients de HBase :

- Ne supporte pas la transaction.
- Pas de permissions ou d’authentification intégrée.
- L’indexation et le tri se fait uniquement sur clé.
- Point de défaillance unique (lorsqu’un seul HMaster est utilisé).
- Ne supporte pas la structure SQL et XML.
- Problèmes de mémoire sur le cluster : Il y a une très faible capacité en mémoire.
- HBase ne vérifie pas le respect des contraintes d’intégrité référentielle et sémantiques.

³. **Server-side** : L’expression server-side (coté serveur) fait référence à des opérations qui sont effectuées par le serveur dans la communication entre client et serveur dans un réseau informatique.

1.2 Conclusion

A travers ce chapitre, nous avons expliqué comment Apache Hbase et Firebase peuvent être utilisés pour stocker et manipuler des données sémantiques récoltées à partir d'ensembles de données à grande échelle. Nous avons présenté pour chacun le modèle de données utilisé, l'architecture physique et logique ainsi que la façon dont les données sont structurées. Enfin nous avons parlé des points forts et inconvénients de ces deux bases de données.

Troisième partie

Réalisation

Chapitre 2

Choix des outils technologiques

Introduction

Pour la mise en œuvre de notre travail nous avons opté pour l'utilisation des technologies : Hadoop et Hbase pour la mise en place de la base de donnée NoSQL. Tout au long de cette partie nous présenterons les étapes d'installation de ces technologies ainsi que les outils de développement utilisés pour développer un simulateur qui génère des données pour la gestion d'électricité dans le cadre d'un smart grid.

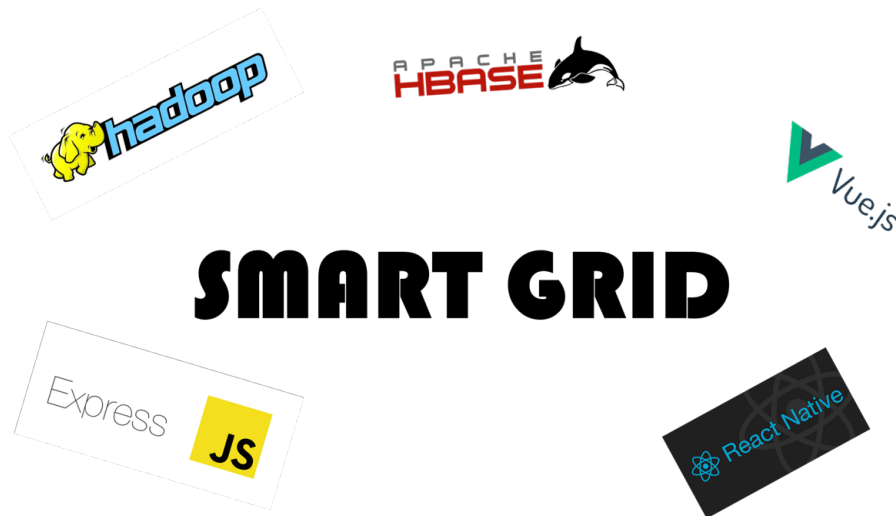


FIGURE 2.1 – Technologies utilisé

2.1 Hadoop :

Hadoop est un framework logiciel dédié au stockage et au traitement de larges volumes de données. Il s'agit d'un projet open source, sponsorisé par la fondation Apache Software Foundation.

Il ne s'agit pas d'un produit à proprement parler, mais d'un framework regroupant des instructions pour le stockage et le traitement de données distribuées. Différents éditeurs de logiciels ont utilisé Hadoop pour créer des produits commerciaux de gestion Big Data.

Les systèmes de données Hadoop ne sont pas limités en termes d'échelle, ce qui signifie qu'il est possible d'ajouter davantage de hardware et de clusters pour supporter une charge plus lourde sans passer par une reconfiguration ou l'achat de licences logicielles onéreuses.



FIGURE 2.2 – Logo Hadoop

2.1.1 Installation d'Hadoop :

Apache Hadoop peut être installée dans différents modes selon l'exigence. Ces différents modes sont configurés lors de l'installation. Nous avons :

- **Mode autonome** : C'est le mode de configuration par défaut d'Hadoop. Il n'utilise pas hdfs à la place, il utilise un système de fichiers local pour l'entrée et la sortie. Il est utile pour le et les tests.
- **Mode pseudo-distribué** : Il est également appelé cluster à nœud unique où Name-Node et DataNode résident sur la même machine. Tous les démons s'exécutent sur la même machine dans ce mode. Il produit un cluster entièrement fonctionnel sur une seule machine.
- **Mode entièrement distribué** : Hadoop s'exécute sur plusieurs nœuds dans lesquels il existe des nœuds séparés pour les démons maîtres et esclaves. Les données sont réparties entre un cluster de machines fournissant un environnement de production.

Pour notre travail nous allons installer un cluster hadoop pseudo-distribué à nœud unique sur Windows 10.

1. Prérequis :

Tout d'abord, nous devons nous assurer que les prérequis suivants sont installés :

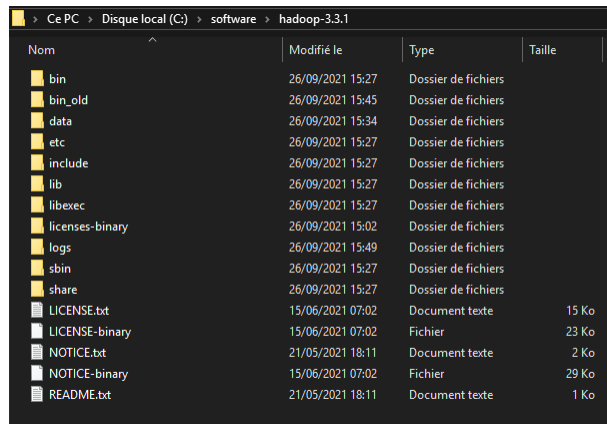
1. **Java 8 environnement d'exécution (JRE)** : Hadoop 3 nécessite une installation de Java 8. Je préfère utiliser le programme d'installation hors ligne. Lien de téléchargement https://www.java.com/en/download/windows_offline.jsp
2. **Java 8 development Kit (JDK)** Lien de téléchargement <https://www.oracle.com/java/technologies/downloads/#java8>
3. Pour décompresser les binaires Hadoop téléchargés, nous devons installer **7zip** . Lien de téléchargement <https://www.7-zip.org/download.html>



FIGURE 2.3 – Logo Java 8

2. Téléchargez les binaires Hadoop.

1. La première étape consiste à télécharger les binaires Hadoop depuis le site officiel <https://archive.apache.org/dist/hadoop/common/hadoop-3.1.0/hadoop-3.1.0.tar.gz>.
2. Après avoir terminé le téléchargement du fichier, nous devons décompresser le package en utilisant 7zip en deux étapes. Tout d'abord, nous devons extraire la bibliothèque `hadoop-3.1.0.tar.gz`, puis nous devons décompresser le fichier tar extrait :



Nom	Modifié le	Type	Taille
bin	26/09/2021 15:27	Dossier de fichiers	
bin_old	26/09/2021 15:45	Dossier de fichiers	
data	26/09/2021 15:34	Dossier de fichiers	
etc	26/09/2021 15:27	Dossier de fichiers	
include	26/09/2021 15:27	Dossier de fichiers	
lib	26/09/2021 15:27	Dossier de fichiers	
libexec	26/09/2021 15:27	Dossier de fichiers	
licenses-binary	26/09/2021 15:02	Dossier de fichiers	
logs	26/09/2021 15:49	Dossier de fichiers	
sbin	26/09/2021 15:27	Dossier de fichiers	
share	26/09/2021 15:27	Dossier de fichiers	
LICENSE.txt	15/06/2021 07:02	Document texte	15 Ko
LICENSE-binary	15/06/2021 07:02	Fichier	23 Ko
NOTICE.txt	21/05/2021 18:11	Document texte	2 Ko
NOTICE-binary	15/06/2021 07:02	Fichier	29 Ko
README.txt	21/05/2021 18:11	Document texte	1 Ko

FIGURE 2.4 – Les fichier Hadoop

3. Configuration Hadoop

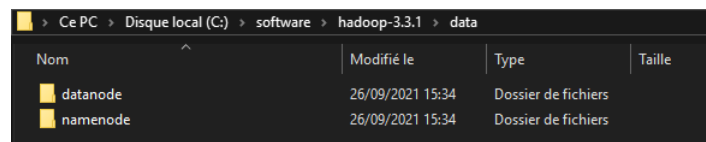
On doit impérativement maintenant éditer certains fichiers situés dans le répertoire `hadoop` du dossier `'etc'` où nous avons installé `hadoop`. Les fichiers qui doivent être modifiés ont été mis en évidence :

1. On modifie le fichier `core-site.xml` dans le répertoire `hadoop/etc/hadoop`. On copie cette propriété xml dans la configuration dans le fichier :

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

FIGURE 2.5 – Modifier le fichier `core-site.xml`

2. On crée un dossier `'data'` dans le répertoire `hadoop`. Puis on crée un dossier avec le nom `'datanode'` et un dossier `'namenode'` dans ce répertoire de données.



Nom	Modifié le	Type	Taille
datanode	26/09/2021 15:34	Dossier de fichiers	
namenode	26/09/2021 15:34	Dossier de fichiers	

FIGURE 2.6 – Créer les deux fichiers dans le dossier `data`

3. On édite le fichier `hdfs-site.xml` dans le répertoire `hadoop/etc/hadoop`, et on ajoute la propriété ci-dessous dans la configuration :

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///C:/software/hadoop-3.3.1/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>C:/software/hadoop-3.3.1/data/datanode</value>
  </property>
</configuration>
```

FIGURE 2.7 – Modifier le fichier `hdfs-site.xml`

4. On modifie le fichier `yarn-site.xml` dans le répertoire `hadoop/etc/hadoop`, et on ajoute la propriété ci-dessous dans la configuration

```
<configuration>

  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

FIGURE 2.8 – Modifier le fichier `yarn-site.xml`

5. On modifie `hadoop-env.cmd` et on remplace `JAVAHOME` par le chemin du dossier java où la jdk 1.8 est installé.

```
@rem The java implementation to use. Required.
set JAVA_HOME=C:\Java\jdk1.8.0_251
```

FIGURE 2.9 – Modifier le fichier `hadoop-env.cmd`

2. Configuration des variables d'environnement.

1. On ouvre le panneau de configuration pour modifier la variable d'environnement système, pour créer une nouvelle variable utilisateur et Mettre le nom de variable comme *HADOOPHOME* et la valeur de la variable comme chemin du dossier bin où on a extrait hadoop.

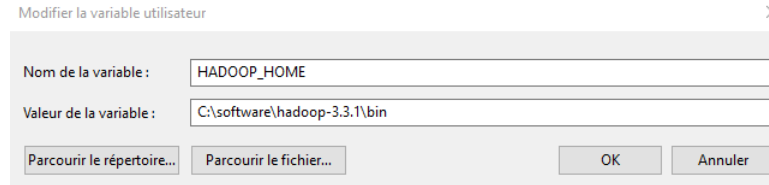


FIGURE 2.10 – *HADOOPHOME*

2. De même, on crée une nouvelle variable utilisateur avec le nom de la variable comme *JAVAHOME* et la valeur de la variable comme chemin du dossier bin dans le répertoire Java.

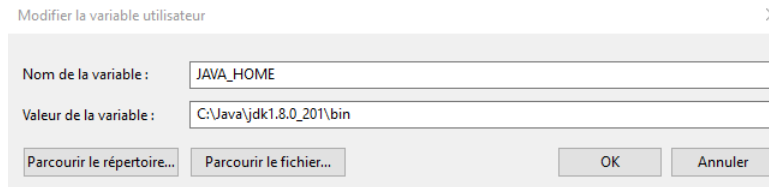


FIGURE 2.11 – *JAVAHOME*

3. On doit maintenant définir le répertoire bin Hadoop et le chemin du répertoire bin Java dans le chemin de la variable système.

2.2 Hbase :

HBase est un des composants additionnels d'Apache. Pour les systèmes de gestion de base de données (SGBD) orientés colonne, soit en "No-SQL", la notion de base de données et la façon dont les données sont stockées sont très différentes des systèmes de gestion de bases de données relationnelles.

Apache HBase est un data store orienté colonne utilisant des paires clé/valeur. La base de données HBase s'installe généralement sur le système de fichiers HDFS (pour Hadoop Distributed File System).



FIGURE 2.12 – Logo Hbase

2.2.1 Installation d'Hbase :

Pour l'installation de Hbase , il est très important de vérifier si Java et Hadoop ont été bien installés sur le système.

1. On commence par télécharger Hbase a partir de son archive officiel(<https://hbase.apache.org/downloads.html>)et On décompresse ce dernier (on a choisi la version 1.4.9).
2. On Crée deux dossiers dans le répertoire racine et les nommé "HBase" et "zookeeper".
3. Maintenant, nous devons modifier 2 fichiers, accédez à l'emplacement décompressé.
 - **1er fichier** : Modifiez `hbase-config.cmd` , situé dans le dossier bin sous l'emplacement décompressé et ajoutez la ligne ci-dessous pour définir :

```
set JAVA_HOME=C:\Java\jdk1.8.0_251
```

FIGURE 2.13 – modifier le fichier hbase-config.cmd

- **2eme fichier** : Modifiez `hbase-site.xml` , situé dans le dossier conf sous l'emplacement décompressé et ajoutez la section ci-dessous

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>file://home/hadoop/HBase/HFiles</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/hadoop/zookeeper</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>false</value>
  </property>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://localhost:9000/hbase</value>
  </property>
</configuration>
```

FIGURE 2.14 – Modifier le fichier hbase-site.xml

2.3 Langages et outils de développement

Pour développer notre propre simulateur on utilise les outils et les langages de programmation suivant :

2.3.1 Javascript

JavaScript est un langage de programmation qui permet d'implémenter des mécanismes complexes sur une page web. À chaque fois qu'une page web fait plus que simplement afficher du contenu statique — afficher du contenu mis à jour à des temps déterminés, des cartes interactives, des animations 2D/3D, des menus vidéo défilants, etc...

JavaScript a de bonnes chances d'être impliqué. C'est la troisième couche des technologies standards du web, les deux premières (HTML et CSS, Les trois couches se superposent naturellement.

2.3.2 vue

Vue.JS, ou simplement Vue, est un framework progressif pour les interfaces utilisateur pour les apps et sites JavaScript. Il s'agit d'un des frameworks front-end JS les plus populaires. On le compare souvent à React, Angular, Ember, etc. Par leur approche et leur ressemblance, Vue et React partagent de nombreux points communs. Le framework apparaît à l'été 2013. Il est développé par Evan You. Peu à peu, Vue va faire parler de lui et s'imposer chez les développeurs JS.

2.3.3 nodes

NodeJS est un outil libre codé en Javascript et orientée pour des applications en réseau. Si vous êtes sur cette page, c'est certainement parce que vous voulez avoir des explications plus détaillées sur NodeJS. Cet outil JavaScript est devenu célèbre dans l'univers du développement web depuis quelques années. D'ailleurs, il est très apprécié des géants du web comme Netflix, PayPal, LinkedIn, Uber, la NASA, etc. Cet article basé sur la définition de NodeJS se donne pour rôle de vous faire découvrir de long en large cette technologie. Vous y trouverez également tous les avantages liés à son utilisation.

2.3.4 express js

ExpressJS est un framework qui se veut minimaliste. Très léger, il apporte peu de surcouches pour garder des performances optimales et une exécution rapide. Express ne fournit que des fonctionnalités d'application web (et mobile) fondamentales, mais celles-ci sont extrêmement robustes et ne prennent pas le dessus sur les fonctionnalités natives de NodeJS.