**Système Hadoop**

1. **Introduction**

Hadoop est un [framework](https://fr.wikipedia.org/wiki/Framework" \o "Framework) [libre](https://fr.wikipedia.org/wiki/Logiciel_libre) et open source en [Java](https://fr.wikipedia.org/wiki/Java_(langage)) destiné à faciliter la création d'applications [distribuées](https://fr.wikipedia.org/wiki/Architecture_distribu%C3%A9e) (stockage et traitement) et [échelonnables (scalables)](https://fr.wikipedia.org/wiki/Scalability" \o "Scalability) permettant aux applications de travailler avec des milliers de nœuds et des [pétaoctets](https://fr.wikipedia.org/wiki/P%C3%A9taoctet" \o "Pétaoctet) de données de tous types.

Chaque nœud de données est constitué de machines standard regroupées en grappe (Cluster), chacune offrant un calcul et un stockage locaux.

Tous les modules de Hadoop sont conçus dans l'idée fondamentale que les pannes matérielles sont fréquentes et qu'en conséquence elles doivent être gérées automatiquement par le framework.

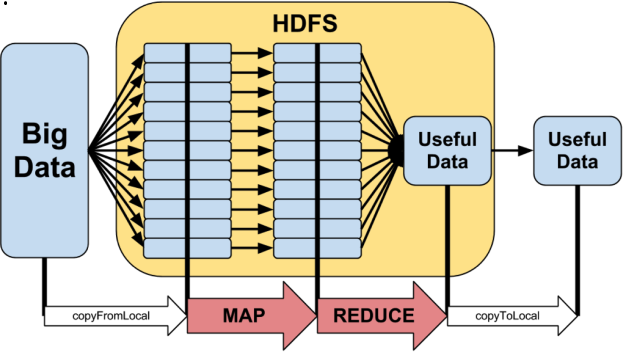
À l’heure actuelle, Hadoop est la principale plateforme du Big Data, il est utilisé pour le stockage et le traitement d’immenses volumes de données. Hadoop est utilisé par de nombreuses entreprises pour leurs projets Big Data.

L'objectif global de Hadoop est de résoudre les deux principaux problèmes de la manipulation de grandes quantités de données :

* stocker ces données (limitation de la taille des disques durs)
* rechercher rapidement dans ces données (limitation des puissances de calcul)

Pour résoudre ces deux problèmes, Hadoop se structure en deux principales couches :

* HDFS : Hadoop Filesystem, un système de fichiers virtuel agrégeant le stockage de plusieurs machines d'un cluster.
* Hadoop MapReduce : un framework logiciel en Java permettant de développer des programmes exécutables de manière distribués grâce à l'utilisation de l'algorithme MapReduce développé par Google

****

1. **Historique**

Lors de l’avènement du World Wide Web à la fin des années 90 et au début des années 2000, les moteurs de recherche et les index furent créés pour aider à localiser des informations pertinentes au sein de contenus textuels.

Au départ, les résultats de recherche étaient renvoyés par des humains. Mais **lorsque le nombre de pages a augmenté à plusieurs dizaines de millions, l’automatisation est devenue nécessaire**.

Les web crawlers ont alors été créés, principalement en tant que projets de recherche universitaires. Les moteurs de recherche comme Yahoo et AltaVista ont également commencé à apparaître. Leur objectif était de **proposer des résultats de recherche web plus rapidement en distribuant des données et des calculs sur différents ordinateurs pour accomplir des tâches multiples simultanément**.

Dans le même temps, le moteur de recherche Google était en développement. Ce projet était basé sur le même concept de stockage et de traitement de données d’une façon distribuée et automatisée pour proposer des résultats de recherche plus rapidement.

En 2008, Yahoo proposa Hadoop sous la forme d’un projet Open Source.

Après quatre ans de développement au sein de la communauté Open Source**, Hadoop 1.0 fut proposé au public en 2012 dans le cadre du projet Apache**. Depuis lors, le framework n’a cessé d’être développé et mis à jour.

La deuxième version **Hadoop 2 a permis d’améliorer la gestion de ressource et la planification**.

**Points forts d’Hadoop:**

* Distribution des traitements au plus près de la donnée (parallélisation).
* Reprise automatique sur erreur.
* Coût de stockage très concurrentiel.
* Écosystème très important.

**Inconvénients:**

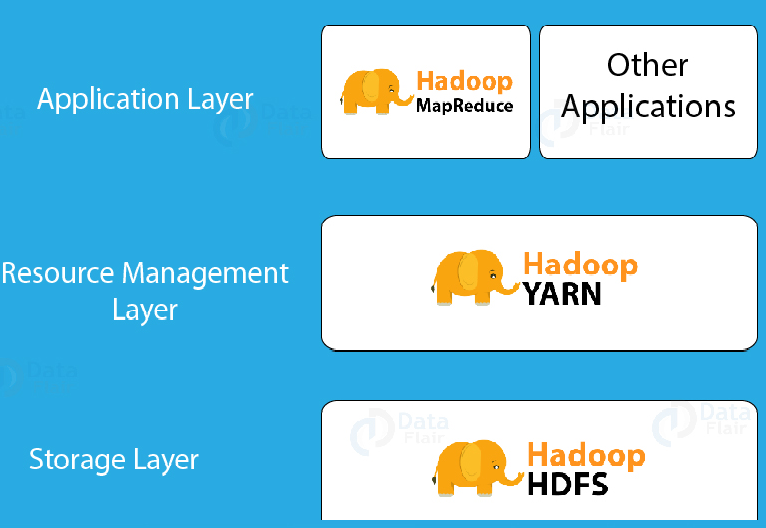
* Performance (par rapport à des bases de données NoSQL, à des grilles de données, etc.).
* Fait principalement pour traiter de très gros volume de données.

1. **Les fournisseurs de distributions Hadoop**

En plus d’Apache Hadoop, il existe quatre distributions Hadoop :

1. **Hortonworks** : la Plateforme de données Hortonworks (HDP:Hortonworks Data Platform) comprend le HDFS, MapReduce, Pig, Hive, HBase et ZooKeeper.
2. **Cloudera** : contient les principaux éléments de base du framework Hadoop (MapReduce et HDFS), ainsi que d’autres composants destinés aux entreprises qui assurent la sécurité, la haute disponibilité, et l’intégration avec le matériel et les autres logiciels.
3. **Pivotal** : en partenariat avec Hortonworks depuis 2015. Ensemble, ils concentrent leurs efforts autour d’un noyau cohérent de fonctionnalités basées sur Apache Hadoop comprenant des intégrations de produits, l’ingénierie conjointe et soutien à la production.
4. **MapR** : utilise un système de fichier Unix natif au lieu de HDFS. Il propose des fonctionnalités de haute disponibilité (snapshots, la réplication et le basculement avec état).
5. **Composants de base de Hadoop**

Le framework Hadoop de base comprend des composants Open Source, tous connectés à un ensemble de modules de cœur destinés à capturer, traiter, gérer et analyser d’importants volumes de données. Parmi ces composants on trouve : HDFS, MapReduce, YARN, Hadoop Common, …).



* **Hadoop Distributed File System (HDFS) :**  système de fichiers distribué sur des noeuds d’un cluster Hadoop.

HDFS est adapté au stockage et la réplication de fichiers de grande taille (>256MB).

* **MapReduce**. Il s’agit d’un modèle de programmation et un framework d’exécution pour traiter en parallèle des applications en mode batch. Division des tâches entre les nœuds puis Ordonnancement et surveillance de l’exécution des tâches.
* **YARN** (Yet Another Resource Negociator) est une technologie de gestion de clusters. Ce module prend en charge l’ordonnancement des tâches (jobs) et alloue les ressources au cluster pour faire fonctionner les applications, et arbitrer quand il y a un conflit de ressources. Il surveille également l’exécution des jobs.
* **Hadoop Common (**hadoop core) : contient des librairies et outils pour supporter les autres projets Hadoop (File system, RPC, Sérialisation).

Le système de fichiers distribué HDFS ou Hadoop est un système de fichiers conçu pour stocker des fichiers de données volumineux sur des machines matérielles de base.

Ces fichiers de données sont découpés en blocs de taille fixe et distribués sur le cluster.

Le bloc est la plus petite unité de données stockée par le système de fichiers. Il contient une quantité minimale de données pouvant être lues.

Ces blocs/morceaux résideront sur des data nodes différents en fonction de la configuration du cluster. Il est de 128 Mo par défaut dans Hadoop 2.0.

Hadoop a été conçu pour stocker des données à l'échelle du pétaoctet. La taille élevée des blocs réduira le coût du temps de recherche. Cette valeur est configurable mais elle n'est généralement pas modifiée.

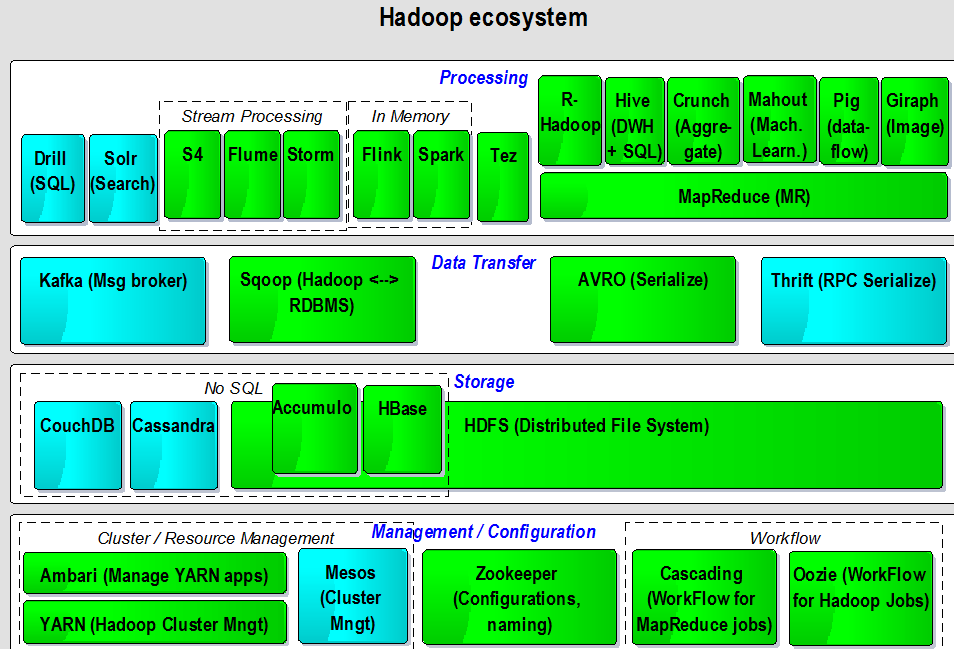
**Divisions**

Les fractionnements sont une représentation logique des données présentes dans les blocs. Les données à traiter par un Mapper individuel sont représentées par InputSplit. InputSplit ne contient pas de données réelles, mais une référence aux données.

1. [**Ecosystème**](https://fr.wikipedia.org/wiki/Cat%C3%A9gorie:%C3%89cosyst%C3%A8me_Hadoop) **Hadoop**

Le terme Hadoop se réfère non seulement aux modules de base cités ci-dessus, mais aussi à [son écosystème](https://fr.wikipedia.org/wiki/Cat%C3%A9gorie:%C3%89cosyst%C3%A8me_Hadoop) et à l'ensemble des logiciels qui viennent s'y connecter comme:

[Apache Pig](https://fr.wikipedia.org/wiki/Apache_Pig), [Apache Hive](https://fr.wikipedia.org/wiki/Apache_Hive), [Apache HBase](https://fr.wikipedia.org/wiki/HBase), Apache Phoenix, [Apache Spark](https://fr.wikipedia.org/wiki/Apache_Spark), [Apache ZooKeeper](https://fr.wikipedia.org/wiki/Apache_ZooKeeper), [Cloudera Impala](https://fr.wikipedia.org/wiki/Impala_(Cloudera)" \o "Impala (Cloudera)), [Apache Flume](https://fr.wikipedia.org/wiki/Apache_Flume), Apache Sqoop, [Apache oozie](https://fr.wikipedia.org/wiki/Apache_Oozie), Apache Storm.



1. **Hadoop Distributed File System (HDFS)**

HDFS est un système de fichiers distribué qui permet le stockage et la récupération de très grandes quantités de données (terra et peta bytes). Aussi il donne un accès haute-performance aux données réparties dans des clusters Hadoop.

Comme d’autres technologies liées à Hadoop, HDFS est un outil clé pour gérer des pools de Big Data et supporter les applications analytiques.

* Il s’agit d’un composant central du Framework hadoop, et plus précisément de son système de stockage.
* **intervention d’opérateur humain**. Il permet de bénéficier simultanément des avantages du calcul parallèle et du calcul distribué.
* Il permet l'abstraction de l'architecture physique de stockage comme s'il s'agissait d'un disque dur unique.
* **HDFS peut être lancé sur commodity hardware, ce qui le rend très tolérant aux erreurs**. Chaque donnée est stockée à plusieurs endroits, et peut donc être récupérée

Une architecture de machines HDFS (aussi appelée [cluster](https://fr.wikipedia.org/wiki/R%C3%A9seau_informatique) HDFS) repose sur deux types de composants majeurs un maître et d'autres esclaves : ***NameNode et DataNode.***

**NameNode (maître)**

Chaque cluster comporte un Namenode individuel (serveur principal), qui se charge d’ouvrir, fermer, renommer les fichiers ou même les dossiers.

C'est le nœud de noms, il gère le [File](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_fichiers) system, l'[espace de noms](https://fr.wikipedia.org/wiki/Espace_de_noms) et les [métadonnées](https://fr.wikipedia.org/wiki/M%C3%A9tadonn%C3%A9es) des fichiers et des répertoires.

Il centralise la localisation des blocs de données répartis dans le [cluster](https://fr.wikipedia.org/wiki/R%C3%A9seau_informatique). Les blocs de données sont cartographiés par le Namenode pour les Datanodes, ainsi, le NameNode a une connaissance des DataNodes dans lesquels les blocs sont stockés.

Quand un client sollicite Hadoop pour récupérer un fichier, c'est via le NameNode que l'information est extraite.

Toutes ces métadonnées, hormis la position des blocs dans les DataNodes, sont stockées physiquement sur le disque dans deux types de fichiers spécifiques edits\_xxx et fsimage\_xxx.

FsImage est un fichier stocké sur le système de fichiers qui contient la structure complète des répertoires (espace de noms) du HDFS avec des détails sur l'emplacement des données sur les [blocs de données](https://www.eginnovations.com/documentation/Hadoop/Hadoop-Data-Blocks-Test.htm) et quels blocs sont stockés sur quel nœud.

Ce fichier est utilisé par le NameNode lors de son démarrage.

EditLogs est un journal de transactions qui enregistre les modifications dans HDFS ou toute action effectuée sur le cluster HDFS comme l'ajout d'un nouveau bloc, la réplication, la suppression, etc.

En bref, il enregistre les modifications depuis la création du dernier FsImage.

La connaissance de la position des blocs dans les DataNodes est reconstruite à chaque démarrage du NameNode dans un mode appelé safe mode.

Pendant le safe mode, l'écriture sur HDFS est impossible, le NameNode charge les fichiers edits\_xxx et fsimage\_xxx et attend le retour des DataNodes sur la position des blocs.

Une fois toutes les opérations réalisées, le safe mode est relâché et l'accès en écriture est de nouveau autorisé.

Le NameNode est relativement gourmand en mémoire.

En effet, chaque fichier, répertoire et bloc dans HDFS est représenté comme un objet dans la mémoire et occupe 150 octets.

Si, par exemple, vous avez 10 millions de fichiers à gérer, le NameNode devra disposer d'un minimum de 1,5 Go de mémoire.

**Secondary NameNode**

Le NameNode dans l'architecture Hadoop est un point unique de défaillance (Single Point of Failure en anglais). Si ce service est arrêté, il n'y a pas moyen d'extraire les blocs d'un fichier donné.

Pour répondre à cette problématique, un NameNode secondaire appelé Secondary NameNode a été mis en place dans l'architecture Hadoop.

Son fonctionnement est de vérifier périodiquement l'état du NameNode principal et copie les métadonnées via les fichiers edits\_xxx et fsimage\_xxx. Si le NameNode principal est indisponible, le NameNode secondaire prend sa place.

**Secondary NameNode** vérifie périodiquement l'état du NameNode principal. A chaque vérification il copie les fichiers de backup : edits et fsimage. Sa structure des dossiers est donc quasi identique que celle du NameNode principal.

***DataNode (esclave)***

**Nœud de données, ce composant stocke et restitue les blocs de données.**

Les DataNodes sont sous les ordres du NameNode et sont surnommés les Workers. Ils sont sollicités par les NameNodes lors des opérations de lecture et d'écriture.

Les données sont stockées sur les **datanodes** (esclaves) tandis que les localisations des blocs de données sont répertoriées par le **namenode** (maître).

C'est le namenode qui sait comment sont décomposés les fichiers et sur quels datanodes sont stockés ces blocs.

Alors que dans les systèmes de fichiers non distribués la taille des fichiers est limitée par le nombre de bits utilisés pour indexer chacun des segments (4 Go pour [**FAT32**](https://fr.wikipedia.org/wiki/FAT32), 16 To pour [**NTFS**](https://fr.wikipedia.org/wiki/NTFS_(informatique)), 2 To à 8 Eo pour [HFS+](http://www.ntfs.com/hfs.htm)), dans HDFS [la limite théorique est de 512 Yo](https://stackoverflow.com/questions/5493873/hadoop-hdfs-maximum-file-size) (512x10¹² To).

Quant au nombre maximum de fichiers par répertoire, il est fixé par le paramètre fs.namenode.fs-limits.max-directory-items dont la valeur par défaut est 1048576.

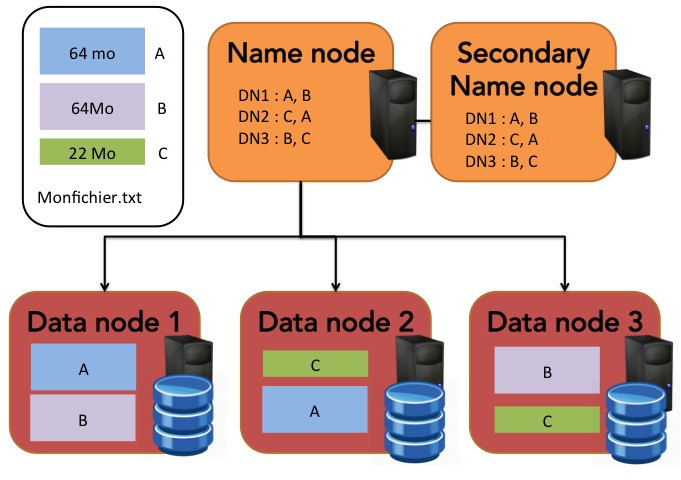
Les datanodes ont très peu d'intelligence et ils ne servent qu'à stocker les données. Les adresses des blocs ainsi que les noms des fichiers sont tous stockés par le namenode

* En lecture, les Datenodes vont transmettre au client les blocs correspondant au fichier à transmettre.
* En écriture, les DataNodes vont retourner l'emplacement des blocs fraîchement créés.
* Les DataNodes sont également sollicités lors de l'initialisation du NameNode et aussi de manière périodique, afin de retourner la liste des blocs stockés.

Lors du processus de lecture d'un fichier, le *NameNode* est interrogé pour localiser l'ensemble des blocs de données.

Pour chacun d'entre eux, le *NameNode* renvoie l'adresse du *DataNode* le plus accessible, c'est-à-dire le *DataNode* qui dispose de la plus grande [bande passante](https://fr.wikipedia.org/wiki/Bande_passante).

Les *DataNodes* communiquent de manière périodique au *NameNode* leurs statuts (la liste des blocs de données qu'ils hébergent).

Le **DataNode**  effectue les opérations demandées par le NameNode. C'est bien le DataNode qui stocke et retrouve les blocks demandés par NameNode. Egalement c'est lui qui apporte au NameNode une liste des blocks stockés. **Ce rapport est envoyé périodiquement.**  


Le HDFS stocke les fichiers de grande taille sur plusieurs machines. Il réalise la fiabilité en répliquant les données sur plusieurs hôtes.

Avec la valeur par défaut de réplication, les données sont stockées sur trois nœuds.

Hadoop fractionne les fichiers en gros blocs et les distribue à travers les nœuds du cluster.



En réalité, **le NameNode et le DataNode sont des codes de programmation Java pouvant être lancés sur des machines commodity hardware**.

**Les blocs HDFS**



Avec la valeur par défaut de réplication, les données sont stockées sur trois nœuds : deux sur le même support et l'autre sur un support différent.



**Composants de NameNode**

Chaque NameNode est composé des parties suivantes :   
- VERSION : le fichier qui contient les informations sur le NameNode. On y retrouve : l'identifiant pour le système de fichiers, le temps de création, le type de stockage, la version de la structure des données HDFS.  
  
- editslog : le fichier est très important, il regroupe toutes les opérations effectuées dans le système. Chaque fois quand une action d'écriture est effectuée, elle est d'abord placée dans ce fichier de logs. Ce n'est que par la suite que la mémoire est notifiée de cette nouvelle opération. Les données stockées dans la mémoire sont utilisées pour les opérations de la lecture.    
  
- fsimage : c'est un fichier binaire utilisé comme un point de contrôle pour les méta-données. Il joue aussi un rôle important dans la récupération des données dans le cas d'une défaillance de système.

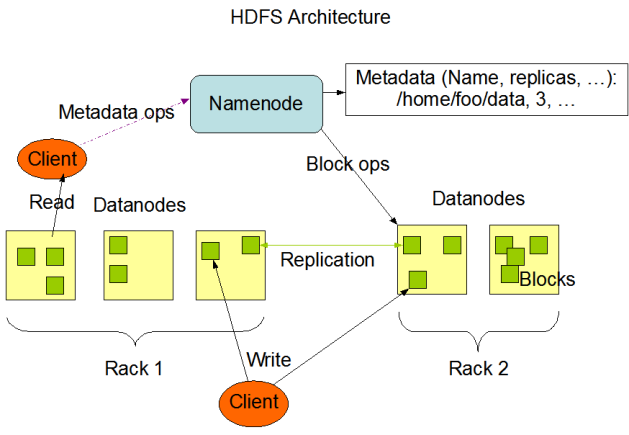
- fstime : il est aussi un fichier binaire. Il stocke les informations sur la dernière opération de contrôle sur les données.  
  
Quand NameNode est démarré, le safe mode  est activé. Cela veut dire que pendant un certain moment même long, si beaucoup de données sont à traiter, les fichiers ne peuvent être accédés qu'en lecture.

C'est parce que pendant le démarrage, NameNode charge ses fichiers fsimage et edits. Ensuite il attend que les DataNodes le tiennent informé quant à l'état des blocks stockés.

Une fois quand un certain pourcentage des DataNodes a rapporté les informations nécessaires, le safe mode est relaché et le système devient à nouveau disponible pour les requêtes d'écriture.



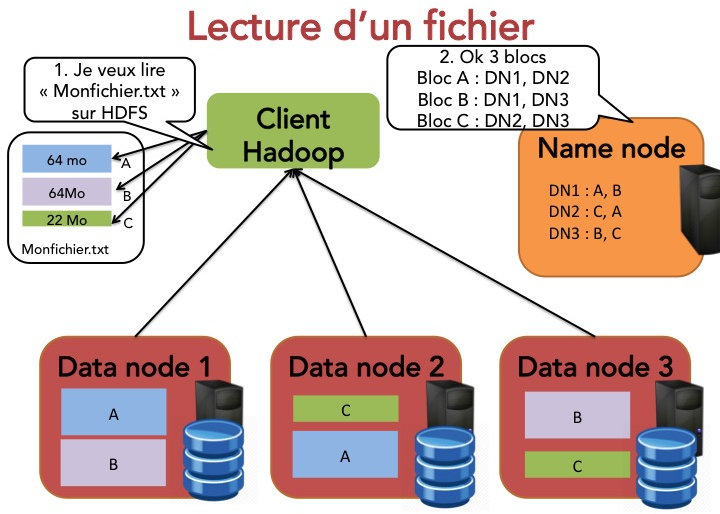
**Comment NameNode et DataNode communiquent ensemble ?**  
Deux opérations permettent de voir la façon de communication entre le namenode et les datanodes, la lecture et l'écriture.



**La lecture**

La lecture d'un fichier placé dans le système HDFS se base sur des interactions entre **NameNode** et **DataNode** :

* Le client se connecte à l'HDFS. Le système demande alors le NameNode.
* NameNode répond et indique la taille du fichier, les emplacements des blocks composant le fichier recherché par le client. Il retourne les adresses les plus proches de la machine du cluster qui effectue l'appel.
* Les DataNodes choisis retournent ensuite les données au client.



**L'écriture**

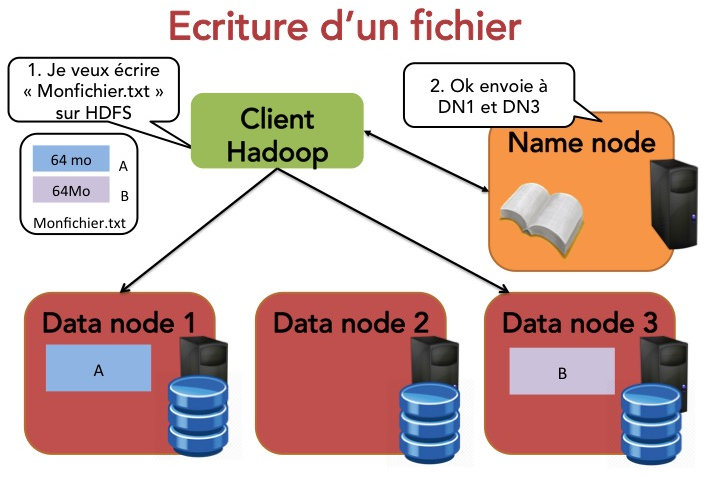
L'écriture est également fondée sur les interactions entre NameNode et DataNodes.

Elle se schématise par ces points :

* Le client demande la création d'un nouveau fichier. Le système informe alors **NameNode** de cette demande. **NameNode** crée le nouveau fichier sans aucun block associé (donc sans DataNodes).

* Si les données à écrire sont correctes (fichier n'existe pas et le client a les droits d'écrire dans le système), le fichier est divisé en plusieurs paquets. Ces paquets sont ensuite placés dans la queue interne appelée data queue (queue des données).
* L'écriture se fait ensuite par paquets, dans les **DataNodes**. Chaque paquet est répliqué dans plusieurs **DataNodes** (nombre déterminé par l'entrée de configuration **dfs.replication.min**). Une fois la réplication terminée, le paquet est alors supprimé d'une autre queue, acknowledged queue (la queue de reconnaissance).

Quand tous les blocks sont écrits, les DataNodes envoient les informations avec les emplacements de ces blocks au NameNode. Grâce à cette information, NameNode pourra à l'avenir retrouver les fichiers demandés par le client.



**Configuration**

Les principaux fichiers de configuration dans Hadoop sont les fichiers **core-site.xml, hdfs-site.xml, mapred-site.xml et yarn-default.xml**.

* + Le fichier **core-site.xml** contient les paramètres de configuration pour le noyau Hadoop tels que les paramètres d'entrées/sorties qui sont communs à HDFS et MapReduce. C'est ce fichier qui permet d'indiquer le nom d'hôte (Hostname) et le numéro de port du système HDFS.
  + Le fichier **hdfs-site.xml** permet de configurer les démons HDFS (NameNode, DataNode, NameNode secondaire). La configuration se fait grâce aux propriétés suivantes :
* dfs.name.dir, qui permet d'indiquer l'emplacement où le NameNode va stocker les métadonnées (FsImage et EditLogs).
* dfs.data.dir, qui permet d'indiquer l'emplacement où les DataNodes stockent les données.
* fs.checkpoint.dir, qui indique le répertoire dans lequel le NameNode secondaire stocke les images temporaires du système de fichier (FsImage) et les journaux de modification (EditLogs).

Le fichier de configuration hdfs-site.xml permet également de spécifier le facteur de réplication des blocs grâce à la propriété **dfs.replication.**

* + Le fichier **mapred-site.xml**est utilisé pour la configuration de MapReduce, et le fichier **yarn-default.xml** pour spécifier les paramètres de configuration pour le NodeManager et le ResourceManager.

**core-site.xml**

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

**hdfs-site.xml**.

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>C:\hadoop-2.8.0\data\namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>C:\hadoop-2.8.0\data\datanode</value>

</property>

</configuration>

**yarn-site.xml**

<configuration>

<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>

**Quelques commandes HDFS :**

* Créer un dossier dans HDFS : hadoop fs -mkdir
* Lister le contenu d'un dossier : hadoop fs -ls
* Charger un ou plusieurs fichiers du local à HDFS : hadoop fs -put
* Exporter un ou plusieurs fichiers de HDFS au local : hadoop fs -get
* Copier un ou plusieurs fichiers dans HDFS : hadoop fs -cp
* Déplacer un ou plusieurs fichiers dans HDFS : hadoop fs -mv
* Charger un ou plusieurs fichiers du local \_a HDFS : hadoop fs -copyFromLocal
* Exporter un ou plusieurs fichiers de HDFS au local : hadoop fs -copyToLocal
* Afficher le contenu d'un fichier : hadoop fs -cat <Path[Filename]>
* Afficher les dernières lignes d'un fichier : hadoop fs -tail <Path[Filename]>