**HDFS**

Table des matières

[Chapitre 1 DEFINITION & OBJECTIFS : 2](#_Toc99395068)

[Chapitre 2 : LA REPLICATION : 3](#_Toc99395069)

[Chapitre 3 : GESTION DES PANNES : 3](#_Toc99395070)

[Chapitre 4 : PROCESSUS DE LECTURE ET D’ECRITURE 3](#_Toc99395071)

[1 : ECRITURE : 3](#_Toc99395072)

[2 : LECTURE : 3](#_Toc99395073)

# 

# DEFINITION & OBJECTIFS :

Hdfs (Hadoop Distrubuted File System) est un système de fichiers distribué utilisé par le Framework Hadoop. Il est basé sur une architecture maitre/esclave.

Map reduce : est un système de calcul distribué.

Quand on installe HDFS sur plusieurs machines(nœuds) d’un cluster cela veut dire qu’on a installé plusieurs **data- nodes** et **un name-node**(ou **maximum 2** le deuxième c’est le **secondary Node**).

**nameNode (service)** : il est le nœud principal(maitre du système) du cluster et gère les métadonnées : chemin, nom du fichier, d’une façon centralisée.

Il est unique mais le secondary-node est un backup (sauvegarde) en cas de panne.

C’est le nameNode qui gère le droit à la lecture et/ou la lecture de blocs. C’est le namenode qui ordonne aux datanodes.

Il a toutes les informations du cluster et rien ne se passe sans qu’il le sache.

**Data-node(service) (esclave :** les blocs de fichiers sont stockés sur data-nodes et les data nodes se chargent de la réplication en blocs du fichier. (La réplication c’est En fonction du facteur de réplication qu’on rentre en paramètre). Ce paramètre est par défaut de l’ordre de 3(3 y compris lui-même) mais on peut le changer selon notre souhait pour chaque fichier. (L’ordre de granularité est le fichier).

C’est les data-nodes aussi qui s’occupent aussi de la suppression et/ou la création de blocs fichiers.

L’objectif de ce service HDFS est :

1. Grace à la réplication de chaque bloc principal d’un fichier sur d’autres data-nodes alors même si le data-node qui reçoit le bloc principal tombe en panne, on pourra toujours accéder au même bloc sur les autres data-nodes qui se sont chargés de le stocker.
2. Accès aux données en continue : on pourra à n’importe quel moment reconstruire le fichier principal et lire sur HDFS.
3. Grace au stockage distribué et au découpage de fichier on pourra stocker d’énormes volumes de données
4. Immuabilité (immutabilité) : les services HDFS n’ont pas besoin de mises à jour car à chaque fois qu’on fait une mise à jour on perd la cohérence définie au début (changements de paramètres par exemple) donc on se limite à une seule écriture mais on peut faire autant de lectures qu’on souhaite.
5. Un même algorithme (pèse quelques dizaines de méga au max) pet être facilement manipulé que des données éparpillées. Donc on préfèrerait déplacer le calcul (traitement) que de déplacer les données.

# : LA REPLICATION :

Supposons qu’un fichier de 10 To soit inséré dans hdfs, il sera subdivisé en plusieurs blocs de 128mo chacun ensuite ces blocs sont stockés sur différents nœuds (data nodes) (machine) du cluster. Ces data nodes répliquent les blocs entre eux selon le facteur qu’on rentre au paramètre pour chaque fichier. Les informations sur chaque cloc qu’il contiennent sont envoyées au namenode (Blockreport).

# : GESTION DES PANNES :

Grace au Heartbeat des datanodes (battements = pings) (qui informent continuellement le namenode de leur présence dans le cluster = signe de vie) si le datanode cesse pendant une période donnée d’émettre le signal il sera considéré par le namenode comme étant inactif ou défectueux, à ce moment-là le namenode ordonnera aux autres datanodes de répliquer tous les blocs de fichiers présents sur ce data-node défectueux.

# : PROCESSUS DE LECTURE ET D’ECRITURE

## 1 : ECRITURE :

Quand on une application client (toute application qui interagit avec le hdfs) sollicite l’écriture d’un fichier réellement on sollicite le namenode (demander l’autorisation).

Le namenode alors nous choisit un de datanodes sur lequel on rédige les blocs texte. A ce moment là à chaque fois qu’un data node reçoit le texte il informe le namenode et le réplique (suivant le facteur de réplication) sur d’autres datanodes.

## 2 : LECTURE :

Qaund un client sollicite