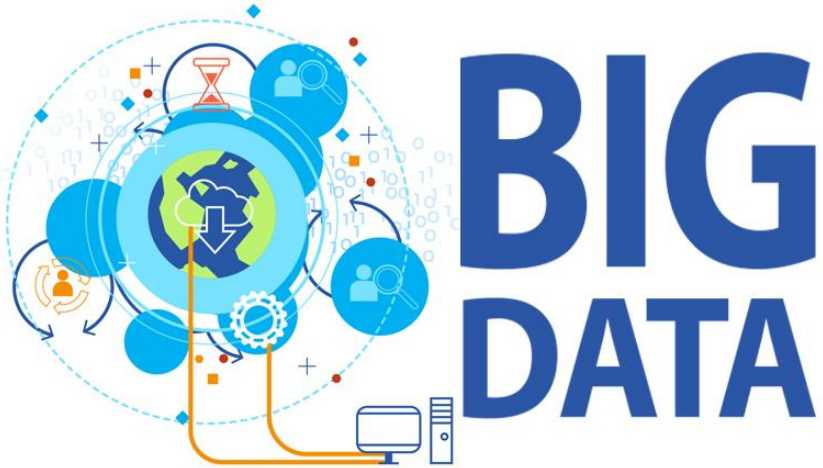


L'éco-Système HADOOP

Pr. HIBBI Fatima-Zohra

f.hibbi@emsi.ma

Plan

- Eco-système: Hadoop
- Le mode de fonctionnement
- L'architecture HDFS
- HDFS: Ligne de commande

Introduction

- Toutes Les machines sont connectées entre elles afin de partager **l'espace de stockage** et **la puissance de calcul**.
- Le Cloud est un exemple d'espace de stockage distribué : des fichiers sont stockés sur différentes machines, généralement en double pour prévenir une panne.
- L'exécution des programmes est également distribuée : ils sont exécutés sur une ou plusieurs machines du réseau.

Tout ce module vise à enseigner la programmation d'applications sur un cluster, à l'aide des outils Hadoop.

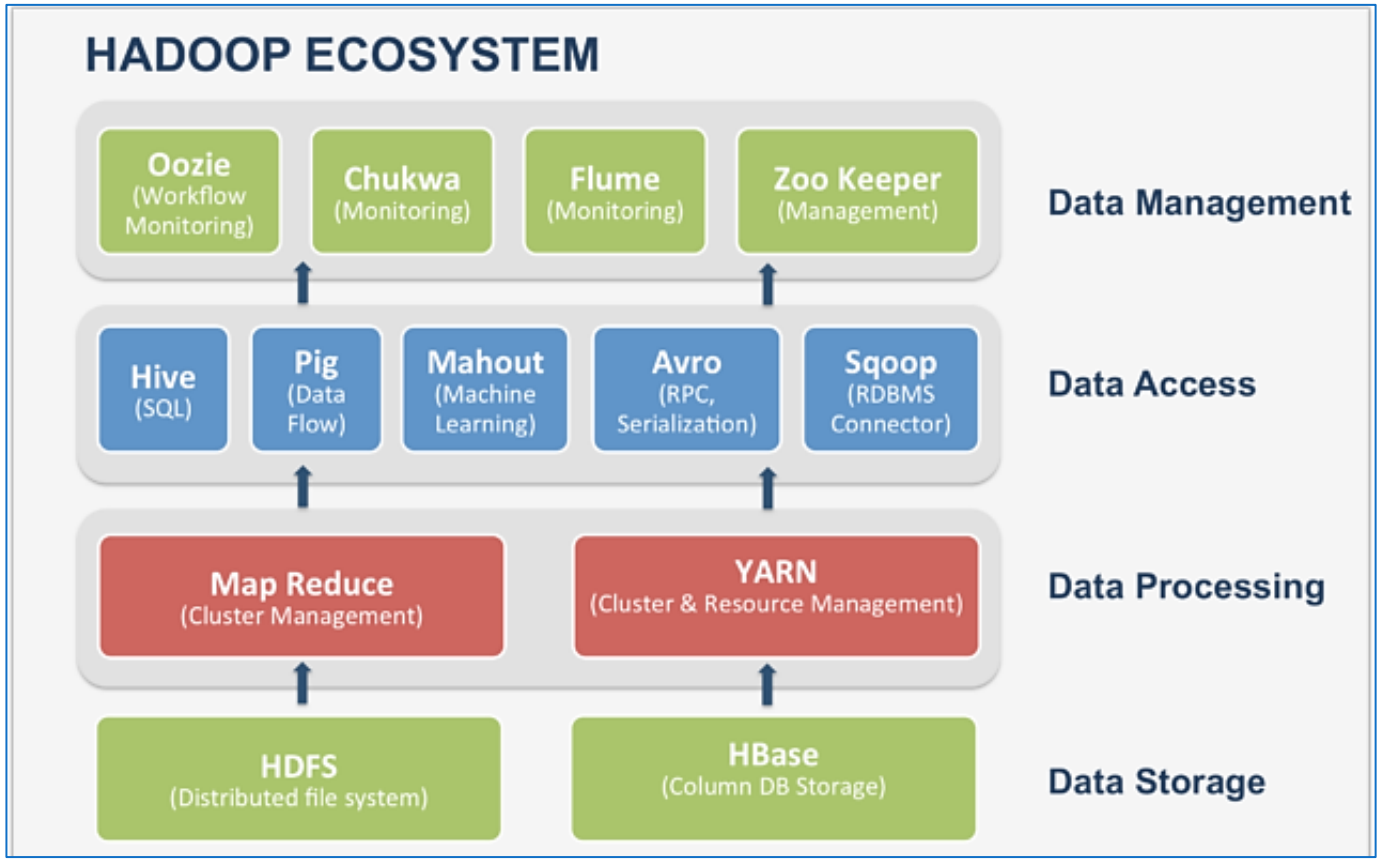
Hadoop ?

- Hadoop est une plateforme de gestion et de traitements de données distribués.
- Il contient de beaucoup de composants, dont :

  HDFS un système de fichier qui répartit les données sur de nombreuses machines,

  MapReduce/Yarn un framework de traitement distribué.

L'Architecture de la plateforme Hadoop



-Pig: Plateforme haut niveau pour le traitement de données, basée sur un langage de script Pig Latin

-Hive: Environnement de haut niveau pour le traitement de données, basé sur un langage proche de SQL (Hive QL)

-Avro: Sérialisation des données (*JSON*)

-Sqoop: Lecture et écriture des données à partir de BD externes

-Flume: Collecte de logs et stockage dans HDFS

-Ooziz, Chukwa, Zoo Keeper: L'ordonncement les job MapReduce.

By Ankita D Mendhekar

HADOOP: Mode de fonctionnement

Hadoop possède trois modes d'exécution :

- Mode local (standalone) : Hadoop fonctionne sur une seule machine et tout s'exécute dans la même JVM (Java Virtual Machine). En mode local le système de gestion de fichiers utilisé est celui du système hôte(Exemple Linux: ext3, ext4 ou xfs) et non HDFS.
- Mode pseudo-distribué: Hadoop fonctionne sur une seule machine, mais chacun des services principaux s'exécute dans sa propre JVM. Le système de fichier utilisé est HDFS dans ce mode.
- Mode totalement distribué: c'est le mode d'exécution réel d'Hadoop. Il permet de faire fonctionner le système de fichiers distribué et les services sur un ensemble de machines différentes.

HDFS: Hadoop File System

Introduction

HDFS : Système de fichiers pour **stocker** des fichiers très **volumineux** dans un **cluster** d'ordinateurs.

HDFS est un système de fichier distribué. C'est-à-dire:

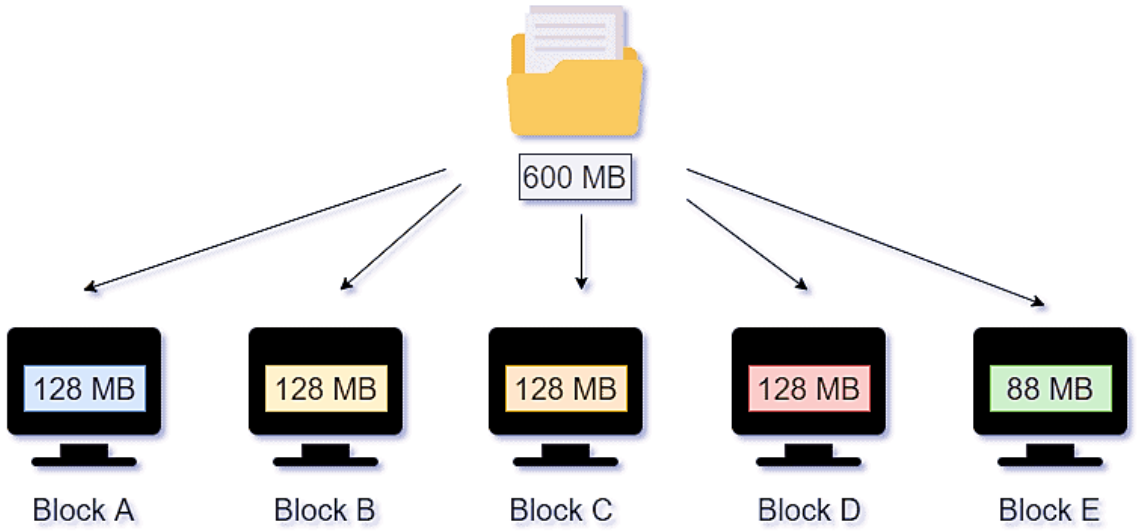
- Les fichiers sont organisé en arbre (linux)

-les fichiers sont copiés en plusieurs exemplaires pour assurer la fiabilité

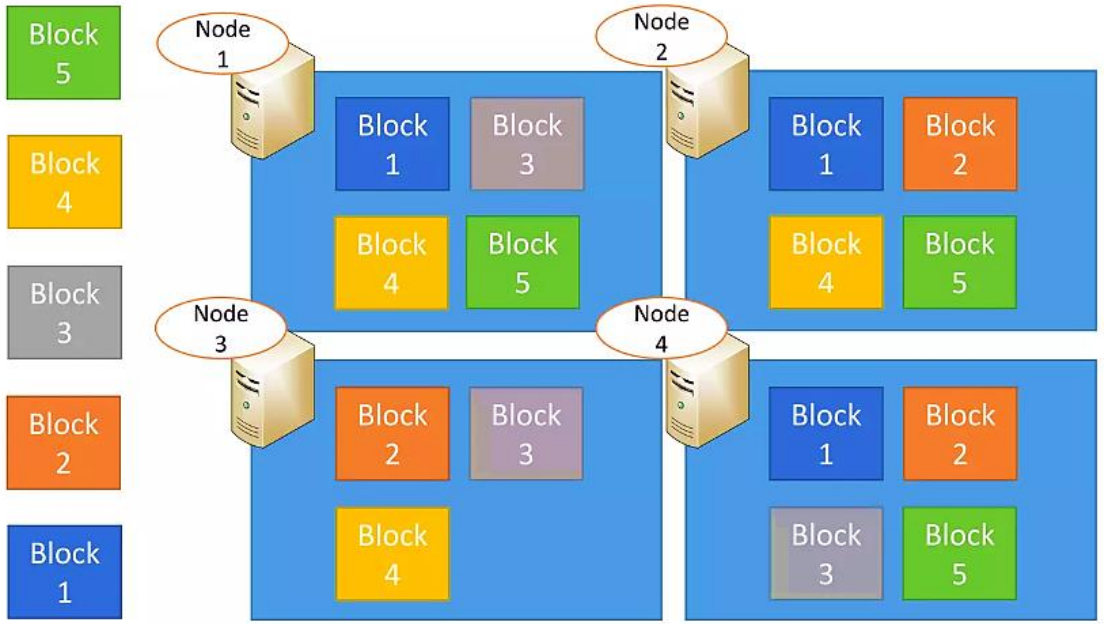
-HDFS stocke les données sur plusieurs nœuds qui forment un cluster

-WORM: write once read many times

La structure de HDFS

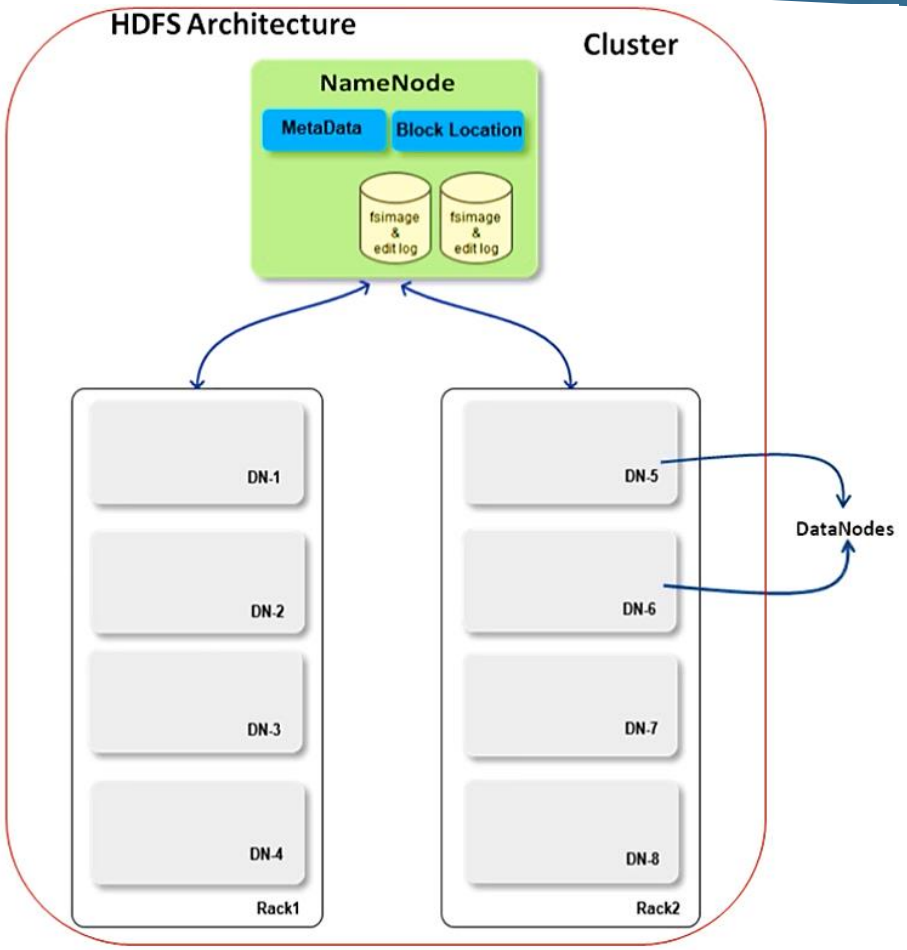


Chaque **fichier** est décomposé en plusieurs **blocs** qui seront répliqués sur les **nœuds.**



HDFS **réplique** les données sur plusieurs **nœuds (x3)** afin d'éviter leur perte en cas de panne de certains nœuds : **Fiabilité, Disponibilité**

L'architecture de HDFS: Maitre/Esclave

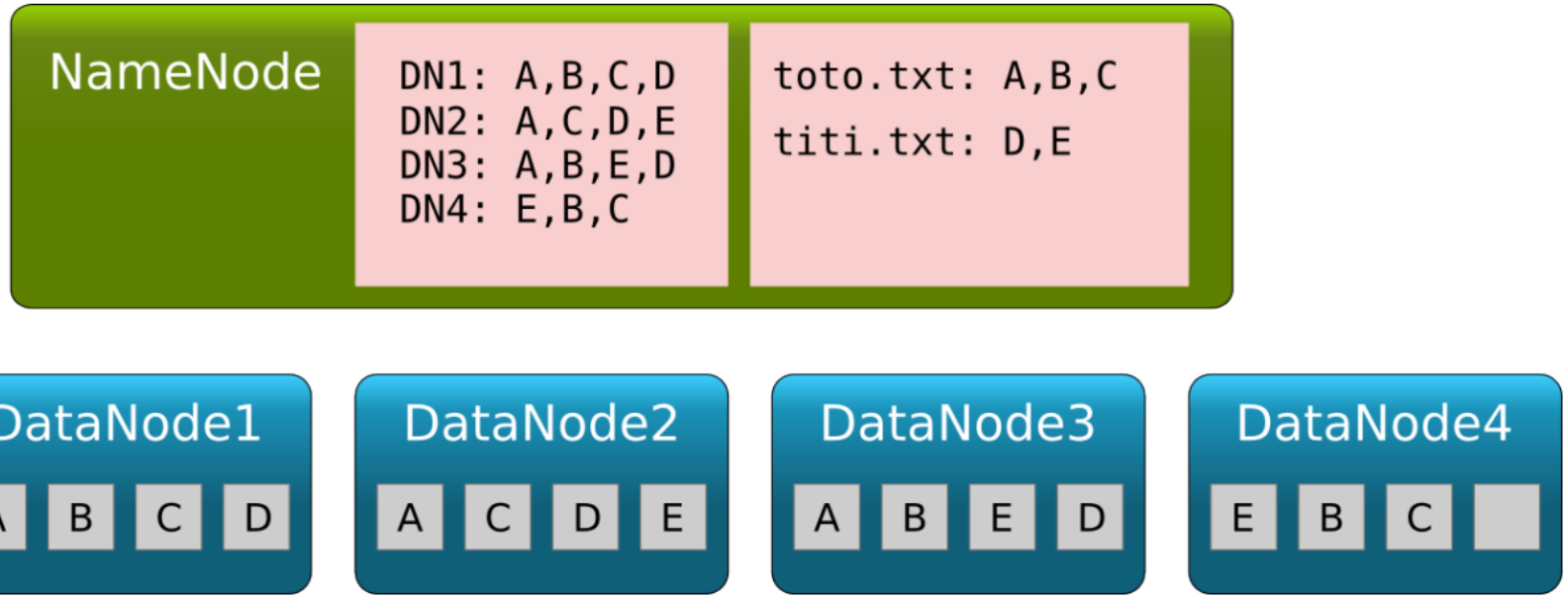


- Plusieurs Datanodes et un seule NameNode est active.
- MetaData: FileName, FileSize, N°Block, User, Group, etc.
- Le NameNode conserve les métadonnées en mémoire pour assurer un accès rapide ce qui nécessite une grande taille mémoire pour son fonctionnement.
- Ces informations sont stockées sur le disque local dans deux fichiers:

+FsImage (métadonnées) chargé en mémoire au démarrage du NameNode.

+EditLog : pour enregistrer chaque modification apportée aux métadonnées du système de fichiers.

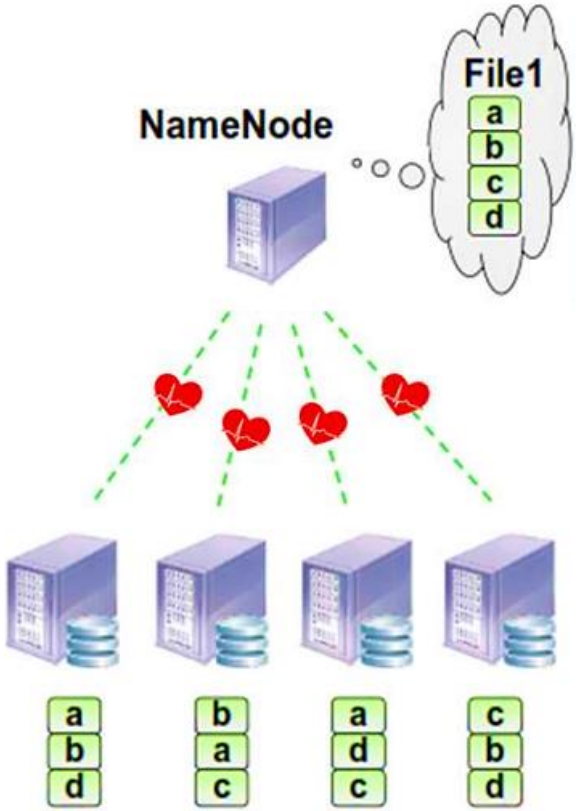
L'architecture de HDFS: DataNodes



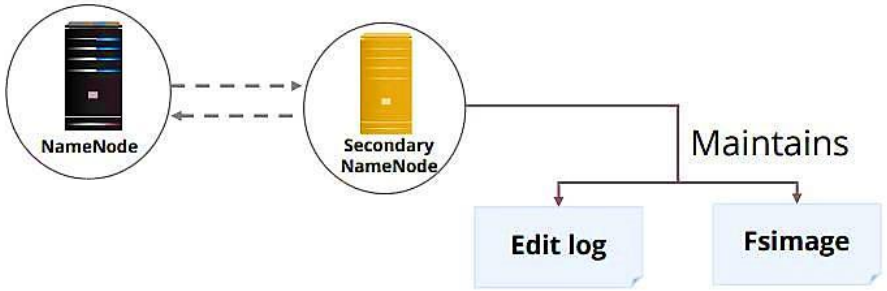
- Les dataNodes contiennent des blocs (A, B, C. . .), le NameNode sait où sont les fichiers : quels blocs et sur quels DataNodes.

L'architecture de HDFS: DataNodes

- HeartBeat: un concept utilisé par les NameNodes pour vérifier l'état des DataNodes (Active/Down)
- DataNodes envoie périodiquement (3sec) un signal au NameNode sous forme de heartbeat (id, espace stockage, espace utilisé, etc) avec un intervalle indiqué dans hdfs-site.xml
- DataNodes outofservice: Absence de HeartBeat et le temps d'attente dépasse 10 min.
- La création, la suppression et la réplication de blocs est effectué sur ordre du NameNode.



L'architecture de HDFS: Secondary NameNode



- Inconvénient majeur : panne du NameNode = mort de HDFS, c'est pour éviter ça qu'il y a le **secondary NameNode.**
- Le secondary NameNode archive les metadata, par exemple toutes les heures.

L'architecture de HDFS

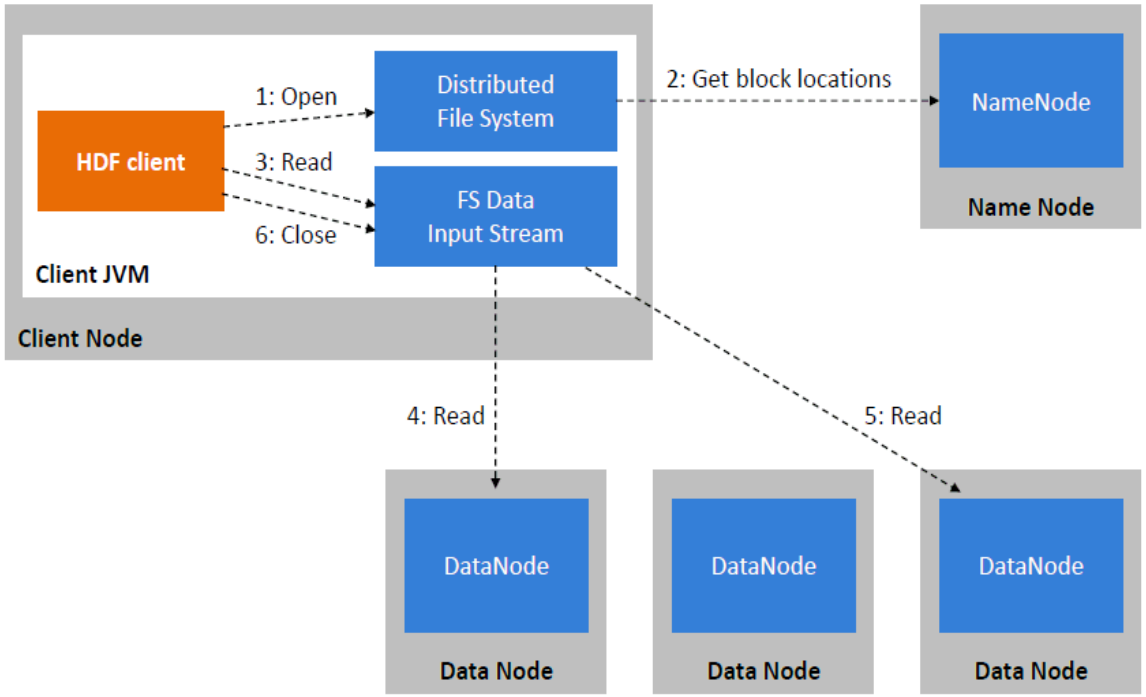
Secondary NameNode: Mode High availability

- Hadoop propose une configuration appelée high availability dans laquelle il y a 2 autres NameNodes en secours, capables de prendre le relais instantanément en cas de panne du NameNode initial.
- Les NameNodes de secours se comportent comme des clones. Ils sont en état d'attente et mis à jour en permanence à l'aide de services appelés JournalNodes.
- ZooKeeper a deux fonctions dans un cluster HDFShaute disponibilité:

  +Détecter une défaillance du NameNode actif

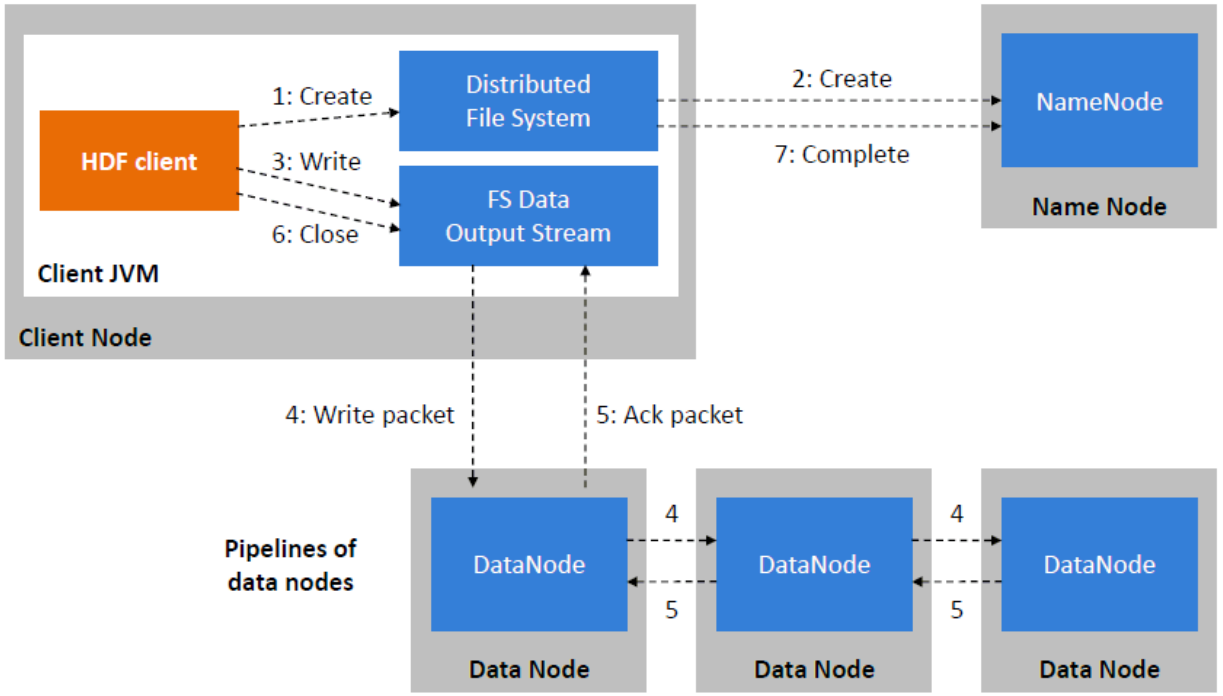
  +ZooKeeper fournit un mécanisme pour l'élection du NameNode actif lorsqu'il y a plusieurs candidats (NameNodes passfis)

Lecture d'un fichier HDFS




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

Ecriture d'un fichier HDFS




```java
import ...;
public class HDFSwrite {
  public static void main(String[] args) throws IOException
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(conf);
    Path nomcomplet = new Path("apitest.txt");
    if (! fs.exists(nomcomplet)) {
      FSDataOutputStream outStream = fs.create(nomcomplet);
      outStream.writeUTF("Bonjour tout le monde !");
      outStream.close();
    }
    fs.close();
  }
}
```

HDFS: Ligne de commande

- Syntaxe utilisé:

  $ hadoop fs –commande

  ou

  $ hdfs dfs -commande
- Pour avoir la liste des commandes:

  $ hdfs dfs -help
- Pour avoir la description d'une commande:

  $ hdfs dfs -help commande

  Exemple:

  $ hdfs dfs -help mkdir
- Pour avoir la syntaxe d'utilisation d'une commande:

  $ hdfs dfs -usage commande

  Exemple:

  $ hdfs dfs -usage put
- Pour vérifier l'état du contenu d'un dossier HDFS:

  $ hdfs fsck chemin-options

  Exemples:

  $ hdfs fsck /user/atelier tp -files

  $ hdfs fsck /user/atelier -files -blocks -locations
- Pour avoir la syntaxe détaillée de hdfs fsck:

  $ hdfs fsck -help

HDFS: Ligne de commande

Exemple:

- Afficher l'arborescence de la racine HDFS:

```
$ hdfs dfs -ls /
```

- Créer un dossier dans HDFS:

```
$ hdfs dfs -mkdir /user/test
```

- Créer un fichier dans HDFS:

```
$ hdfs dfs –touchz /user/filetest.txt
```

- Copier un fichier local vers HDFS:

```
$ hdfs dfs -copyFromLocal /home/cloudera/file1.txt /user/test/file1.txt
$ hdfs dfs -put /home/cloudera/file1.txt /user/test/file1.txt
```

- Copier le contenu d'un dossier local vers HDFS:

```
$ hdfs dfs -put /home/cloudera/bank /user/test/
```

- Copier un fichier vers HDFS en précisant la taille de bloc différente de celle par défaut (~32 MB):

```
$ hdfs dfs -D dfs.blocksize=33554432 -put /home/cloudera/file1.txt /user/test/file1.txt
```

HDFS: Ligne de commande

Exemple:

- Afficher le contenu d'un dossier HDFS:

```
$ hdfs dfs - ls /user/test
```

- Copier un fichier HDFS vers le système local

```
$ hdfs dfs -copyToLocal /hbase/hbase.id /home/cloudera/send.txt
```

- Afficher les dernières lignes (1Ko) d'un fichier:

```
$ hdfs dfs -tail /hbase/hbase.id
```

- Affiche un rapport sur les blocs des fichiers d'un dossier donné:

```
$ hdfs fsck /user/atelier -files -blocks –locations
```

- Avoir l'emplacement du FsImage:

```
$hdfs getconf -confKey dfs.namenode.name.dir
```