

HDFS

Hadoop Distributed File System

Fiche de Révision Examen

Pr. Btihal El Ghali

1 Systèmes de Fichiers

Définition

Un système de fichiers est utilisé pour lire et écrire sur un disque dur.

1.1 Types de Systèmes de Fichiers

- **Standalone FS** : NTFS (Windows), Ext (Linux), Mac FS (Mac)
- **Distributed FS** : HDFS

Point Clé

Un système de fichiers distribué s'installe au-dessus d'un SF standalone.

2 HDFS - Architecture

2.1 Caractéristiques Principales

Origine

HDFS est basé sur le système de fichiers Google (GFS)

Objectifs :

- Fonctionnement fiable et tolérant aux pannes
- Grands groupes de machines (jusqu'à milliers d'ordinateurs)
- Architecture maître/esclave

2.2 Architecture Maître/Esclave

Type	Composant	Rôle
2*Master	NameNode	Gère les métadonnées du système de fichiers
	Secondary NameNode	Points de contrôle périodiques (CheckPoint)
Slave	DataNode	Stocke les données réelles

3 Composants HDFS

3.1 NameNode (Master Daemon)

Rôle du NameNode

Le seul et unique NameNode du cluster - cœur du système HDFS

Fonctions :

1. Mappe un fichier avec son ensemble de blocs
2. Mappe un bloc avec les DataNodes où il réside
3. Moteur de réplication pour les blocs

3.2 DataNode (Slave Daemon)

Responsabilités :

- Lecture et écriture dans le système de fichiers
- Création de blocs
- Effacement
- Réplication basée sur instructions du NameNode

3.3 Secondary NameNode

Attention

Le Secondary NameNode NE PEUT PAS remplacer le NameNode principal en cas d'échec !

Fonctions :

- Effectue des points de contrôle périodiques (CheckPoint)
- En cas d'échec : récupération manuelle par administrateurs

3.4 Standby NameNode

Hadoop 2.0

Disponible uniquement dans Hadoop 2.0 et versions ultérieures

Caractéristiques :

- Résout le problème de SPOF (Single Point Of Failure) de Hadoop 1.0
- Fournit un basculement automatique en cas d'échec du NameNode actif

4 Mécanismes HDFS

4.1 Heartbeats (Battements de Cœur)

Mécanisme

DataNodes → NameNode : heartbeat toutes les **3 secondes**

Utilité :

- Le NameNode détecte les échecs des DataNodes
- Surveillance de l'état du cluster en temps réel

4.2 Blocs de Fichiers

Version Hadoop	Taille de Bloc par Défaut
Hadoop 1.x	64 Mo
Hadoop 2.x	128 Mo

Principe

Les données (fichiers) sont divisées en blocs de taille fixe

4.3 Réplication

Tolérance aux Pannes

Les pannes sont tolérées grâce à la réplication des blocs sur HDFS

Caractéristiques :

- Facteur de réplication par défaut : **3R** (3 répliques)
- Permet défaillance de nœud sans perte de données
- Configurable selon les besoins

5 Stockage de Fichiers sur HDFS

5.1 Processus d'Écriture

1. **Client** : Envoie demande via API client (ex : Write/1GB/nameF.txt)
2. **NameNode** : Génère métadonnées
 - Nombre de blocs et répliques
 - Nœuds contenant les blocs (Rack Awareness Algorithm)
3. **DataNodes** : Stockent les blocs et leurs répliques

5.2 Exemple Pratique

Fichier de grande taille :

- Division en Bloc 1, Bloc 2, Bloc 3
- Chaque bloc répliqué 3 fois
- Distribution selon Rack Awareness Algorithm

6 Configuration HDFS

6.1 Valeurs par Défaut

Paramètre	Valeur par Défaut
Block Size	64 MB (Hadoop 1.x) / 128 MB (Hadoop 2.x)
Replication Factor	3
Web UI Port	50070

6.2 Fichier de Configuration

Chemin : /etc/hadoop/conf/hdfs-site.xml

```
<property>
  <name>dfs.blocksize</name>
  <value>268435456</value>
</property>

<property>
```

```
<name>dfs.replication</name>
<value>3</value>
</property>

<property>
  <name>dfs.namenode.http-address</name>
  <value>itracXXX.cern.ch:50070</value>
</property>
```

7 Placement de Blocs

7.1 Stratégie Actuelle

Stratégie de Placement

1. **1ère réplique** : Nœud local
2. **2ème réplique** : Rack distant
3. **3ème réplique** : Même rack distant
4. **Répliques supplémentaires** : Placement aléatoire

Note : Les clients lisent à partir des répliques les plus proches.

7.2 Rack Awareness Algorithm

Fonctionnement :

- La bande passante réseau est calculée pour chaque cas
- La valeur minimale définit le rack de réplication

Pourquoi 2 répliques sur même rack ?

Probabilité très minime que deux racks tombent en défaillance simultanément → Optimisation de la bande passante

8 Opérations Read/Write

8.1 Processus d'Écriture

1. Client contacte NameNode
2. NameNode fournit liste de DataNodes
3. Client écrit sur 1er DataNode
4. Pipeline de réplication : DN1 → DN2 → DN3
5. Accusés de réception remontent le pipeline
6. Confirmation finale au Client

8.2 Processus de Lecture

1. Client contacte NameNode
2. NameNode fournit emplacements des blocs
3. Client lit depuis DataNode le plus proche
4. Lecture parallèle de plusieurs blocs possible

9 Gestion des Pannes

9.1 Types de Pannes

Type	Exemples
Temporaire	Défaillance réseau, Défaillance logiciel
Permanente	Défaillance matérielle

9.2 Panne d'un Nœud Esclave

Conséquences :

- Une copie de bloc est perdue
- Client ne le saura jamais (transparence)
- NameNode détecte via absence de heartbeat

Solution Automatique :

- **AF (Automatic Failover)** : Basculement automatique
- Hadoop remplace la copie perdue
- Respect du facteur de réplication
- Mise à jour des métadonnées

Recommandation

Supprimer le nœud en panne et recréer un nouveau dans tous les cas (temporaire ou permanente) pour éviter les blocs non identifiés.

9.3 Panne d'un Nœud Maître

Solutions selon version :

1. **Hadoop 1.x** : Secondary NameNode (récupération manuelle)
2. **Hadoop 2.x** : Standby NameNode (basculement automatique)
3. **Hadoop 2.x avec HA** : Passif NN (1 ou plusieurs)

10 HA Cluster (Haute Disponibilité)

10.1 Architecture HA

Composants Minimaux

- Au moins **2 NameNodes** (+ Resource Managers)
- Au moins **3 nœuds Zookeeper**
- **2 ou 3 Journal Nodes**
- Nœuds esclaves (DataNodes + Node Managers)

10.2 Fonctionnement

Mode Normal :

- PNN (Standby NN) reçoivent les heartbeats
- Seul ANN (Active NN) gère lectures/écritures

En cas de défaillance ANN :

1. Zookeeper fait l'élection du nouveau NN
2. Métadonnées actualisées récupérées des Journal Nodes
3. Basculement automatique et transparent

11 Interfaces d'Accès HDFS

11.1 Types d'Interfaces

- Java API (DistributedFileSystem)
- C wrapper (libhdfs)
- HTTP protocol
- WebDAV protocol
- **Shell Commands** (la plus simple et familière)

12 Commandes HDFS - À CONNAÎTRE

12.1 Commandes de Base

```
# Version de Hadoop
hadoop version

# Voir les processus Java
jps
```

12.2 Commandes Utilisateur (dfs)

Lister et Explorer :

```
# Lister contenu
hdfs dfs -ls <path>
hdfs dfs -ls /user/excode

# Lister recursif
hdfs dfs -ls -R

# Espace disque
hdfs dfs -du -h /
hdfs dfs -du -s /hbase/data/hbase/namespace/
```

Créer et Copier :

```
# Creer dossier
hdfs dfs -mkdir hdata

# Copier vers HDFS
hdfs dfs -put data/file.txt hdata
hdfs dfs -copyFromLocal data/file.txt hdata

# Copier depuis HDFS
hdfs dfs -get hdata/file.txt file.txt.hdfs
hdfs dfs -copyToLocal hdata/file.txt file.txt.hdfs

# Verifier integrite
md5sum file.txt file.txt.hdfs
```

Manipuler Fichiers :

```
# Copier dans HDFS
hdfs dfs -cp /hdata/file.txt /autreD

# Supprimer fichier
hdfs dfs -rm tdataset/tfile.txt

# Supprimer dossier recursif
hdfs dfs -rm -r tdataset/
hdfs dfs -rmr tdataset/

# Afficher contenu
hdfs dfs -cat tdataset/tfile.txt

# Ecrire dans fichier
echo "blah_blah_blah" | hdfs dfs -put - tdataset/tfile.txt
```

Statistiques :

```
# Facteur de replication (%r)
hdfs dfs -stat "%r" hdata/file.txt
```

12.3 Commandes fsck (File System Check)

```
# Verifier systeme de fichiers
hdfs fsck /

# Lister blocs et locations
hdfs fsck /user/cloudera/tdata/geneva.csv -files -blocks -locations

# Afficher blocs corrompus
hdfs fsck / -list-corruptfileblocks
```

12.4 Commandes Administration (dfsadmin)

```
# Rapport statut cluster
hdfs dfsadmin -report

# Arbre des racks
hdfs dfsadmin -printTopology

# Info DataNode specifique
hdfs dfsadmin -getDatanodeInfo localhost:50020
```

12.5 Commandes Avancées

```
# Lister NameNodes
hdfs getconf -namenodes
```

13 Points Clés pour l'Examen

À Retenir Absolument

1. **Architecture** : 1 NameNode, N DataNodes
2. **Blocs** : 64 Mo (Hadoop 1.x), 128 Mo (Hadoop 2.x)
3. **Réplication** : Facteur par défaut = 3
4. **Heartbeat** : Toutes les 3 secondes
5. **Placement** : Local, Rack distant, Même rack distant
6. **SPOF** : Résolu en Hadoop 2.x avec Standby NN
7. **HA Cluster** : 2 NN, 3 Zookeeper, 2-3 Journal Nodes
8. **Port Web UI** : 50070

Pièges Fréquents

- Secondary NameNode \neq Backup (ne remplace pas le NN!)
- Standby NameNode disponible uniquement Hadoop 2.x+
- AF peut créer des blocs non identifiés si panne temporaire
- Les clients lisent des répliques les plus proches (optimisation)

Formules de Calcul

Nombre de blocs :

$$\text{Nb blocs} = \left\lceil \frac{\text{Taille fichier}}{\text{Taille bloc}} \right\rceil$$

Espace total utilisé :

$$\text{Espace} = \text{Nb blocs} \times \text{Taille bloc} \times \text{Facteur réplication}$$

Temps transfert (théorique) :

$$\text{Temps} = \frac{\text{Taille données}}{\text{Débit disque (75 MB/s)}}$$

14 Configuration Cloudera

14.1 Cloudera Manager

```
# Lancer Cloudera Manager
sudo /home/cloudera/cloudera-manager --force --express

# Acces Web UI
http://quickstart.cloudera:7180
```

14.2 Services à Vérifier

- Apache Impala (requêtes interactives)
- Apache Hive (stockage structuré)
- HUE (interface utilisateur)
- HDFS (stockage distribué)
- YARN (framework de traitement)

Bonne révision et bon courage pour l'examen !