Projet Systèmes concurrents : HDFSv0

La concurrence

Cette version V0 gère la concurrence en lecture des fichiers grâce au système de fichier basique déjà fournit par l'OS. De plus la concurrence de transmissions est également gérée c'est à dire pour chaque écriture sur un socket un slave est lancé, ainsi nous pouvons transmettre des datas à tous nos serveurs hdfs en parallèle. Cependant la concurrence du problème des Lecteurs/Rédacteurs n'a pas encore été prise en compte. Pour la mettre en oeuvre l'idée serait de placer un système de moniteur sur un Daemon auquel le client demande l'autorisation avant de transmettre au HdfsServeur.

La fonction write

Dans le cas du LineFormat

Nous créons un slave (SlaveHdfsWrite) pour chaque serveur. Chaque slave enverra un bout du fichier calculée telle que tailleFichierSlave = tailleFichierClient/nombreServeursHdfs.
Nous donnons en paramètres au slave les numéros de la première et dernière ligne du fragement que le slave doit envoyé au serveur qui lui est associé (le hostname et le port sont donc également passés en paramètres). Le slave envoie successivement au serveur le nom de la commande sous forme de string, ici ("write"), puis le nom du fichier que le client veut écrire et enfin les lignes du fragment de fichier qui lui est associé sous forme de KV, avec comme clé, le numéro de la ligne dans le fragment, et comme valeur, la ligne en question.

Enfin chaque serveur crée un fichier avec le même nom que le fichier original (Attention
ceci est possible car le facteur de répblicatinousest égale à 1 dans cette version), avec les lignes du fraguement reçu. Concernant la structure de nos serveurs, nous définissons des dossiers "files" + "numPort" car nous sommes en "local".

Nos tests

Nous avons crée un fichier test.txt qui contient:

```
Je
suis
un
petit
poulet
scandinave.
Je
n
'
aime
pas
les
saucisses
turques.
```

nous lançons 4 serveurs de la manière suivante : HdfsServer numPort (avec les valeurs 8080, 8081, 8082, 8083).

```
HdfsServer 8080
HdfsServer 8081
HdfsServer 8082
HdfsServer 8083
```

Puis nous lançons la commande:

```
java HdfsClient write line test.txt
```

Dans les dossiers files8080, files8081, files8082, files8083, nous retrouvons des fichiers test.txt qui contiennent:

```
    Je
    suis
    un
    petit
------------------
    poulet
```

```
    scandinave.
    Je
    n
----------------
    '
    aime
    pas
    les
----------------
    saucisses
    turques.
```

Dans le cas du KvFormat

Si nous avons en entrée un fichier sous la forme d'un KvFormat, c'est-à-dire que chaque line est de la forme: clé<->valeur, le principe reste le même, nous modifions juste les KV que le slave envoie au serveur. À partir de la ligne a<->b nous envoyons un KV qui a pour clé "a" et pour valeur "b".

Nos tests

De même, nous lançons la commande :

```
java HdfsClient kv write sur le fichier test-kv.txt
```

et nous obtenons bien dans chaque dossiers un bout du fichier original.

La fonction read

Nous créons un slave (SlaveHdfsRead) pour chaque serveur, chaque slave conserve l'ObjectOutputstream associé à son serveur. Le client stocke dans une liste les différents slave pour reconstruire le fichier en écrivant successivement dans un fichier toutes les lignes de tous les ObjectOutputstream récupérés depuis la liste des slaves.

Le slave envoie successivement au serveur le nom de la commande sous forme de string, ici ("read"), puis le nom du fichier que le client veut lire. Le serveur envoie les lignes du fragment de fichier qu'il possède. Le slave se contente de transmettre l'ObjectOutputstream au client qui fera la reconstruction finale.

Nos tests

Une fois que nous avons mis les différents fragements d'un fichier dans les dossiers "files*port*" et que nous avons supprimé le fichier de départ, nous lançons la commande :

```
java HdfsClient read test.txt
```

Le fichier test.txt est généré et bien composé des différents fragements dans l'ordre.

La fonction delete

Nous créons un slave (SlaveHdfsDelete) pour chaque serveur, le slave envoie au serveur la commande sous forme de string, ici ("delete"), puis le nom du fichier à supprimer. Le serveur supprime donc son fichier qui a pour chemin : "files" + numPort + "/" + nomDuFichier

Nos tests

En lançant la commande:

```
java HdfsClient delete test.txt
```

Tous les dossiers file + numPort se vident.