Réponses Projet Données Réparties
Partie Hidoop

Axel GRUNIG
Thomas GUILLAUD

Département Sciences du Numérique - ASR
2020-2021

1 Réponses

Suite aux retours, nous avons ajouter du multithreading au niveau des Worker dans la classe WorkerImpl, afin qu'ils puissent continuer d'accepter des requêtes du client, malgré le fait qu'il soit en train d'en traiter une.

```java
public class WorkerImpl extends UnicastRemoteObject implements Worker {

    static public int PORT = 2000;


    protected WorkerImpl() throws RemoteException { }

    @Override
    public void runMap(Mapper m, Format reader, Format writer, CallBack cb) throws RemoteException {
        reader.setFname(Project.getDataPath()+reader.getFname());
        writer.setFname(Project.getDataPath()+writer.getFname());
        Thread t = new Thread(new RunMap(m, reader, writer, cb));
        t.start();
    }

    public static void main(String args[]){
        try {
            WorkerImpl worker = new WorkerImpl();
            LocateRegistry.createRegistry(PORT);
            Naming.rebind( name: "//localhost:" + PORT + "/worker", worker);
        } catch (RemoteException | MalformedURLException e) {
            e.printStackTrace();
            System.exit( status: 1);
        }
    }
}
```

Figure 1: Classe WorkerImpl améliorée

C'est donc la classe RunMap qui est lancé dans un autre thread qui va effectuer les différentes opérations pour exécuter le mapper.

```java
class RunMap implements Runnable {
    private final Mapper map;
    private final Format reader;
    private final Format writer;
    private final CallBack cb;

    public RunMap(Mapper m, Format reader, Format writer, CallBack cb){
        this.map = m;
        this.reader = reader;
        this.writer = writer;
        this.cb = cb;
    }

    @Override
    public void run() {
        reader.open(Format.OpenMode.R);
        writer.open(Format.OpenMode.W);
        map.map(reader, writer);
        reader.close();
        writer.close();
        System.out.println("Fini Map :"+writer.getFname().substring(Project.getDataPath().length()));
        try {
            cb.reveiller();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 2: Classe RunMap

De plus, nous avons décider de laisser le multithreading au niveau du client hidoop (classe Job) afin de paralléliser l'appel des Worker. Cela leur permet donc d'être lancées en même temps, et donc d'effectuer leur tache en même temps.

Concernant la piste d'amélioration fournie par l'autre groupe, nous l'avons suivie puisqu'elle nous a parue très pertinente. Nous avons donc rajoutée un timer d'attente pour la terminaison des Workers qui permet de ne pas attendre indéfiniment un Worker qui aurait un problème de connexion ou un autre dans problème du même type. Nous avons arbitrairement choisi la valeur de départ de ce timer que nous avons fixé à 30s. Ce temps peut totalement être modifié si nécessaire pour les tests.