

Rapport TP SLR207

Aurélien Blicq

Le début de ce document reprend les questions du sujet de TP. Cependant, à partir de la question 41, j'ai arrêté de répondre aux questions pour me concentrer sur l'implémentation de l'algorithme. A la fin du rapport, il y a une section Résultats et Conclusion qui reprend les tests effectués à la fin du projet.

I- programme séquentiel

1. Un HashMap est la structure la plus adaptée car elle permet de lier des mots (objets de type String) à un nombre d'occurrences (objets de type Integer)
 2. Avec le fichier *sante_publique.txt*, on a un temps de calcul de 2.5 secondes
 3. Avec le fichier de pages web, on a un temps de calcul de 65 secondes
-

II- ordinateurs en réseau

10. En tapant la commande `nslookup`, on peut récupérer le long d'un ordinateur à partir de son nom court.

Exemple:

Note: dans la suite, les lignes commençant par \$ sont des commandes et les lignes suivantes en sont le résultat

```
$ nslookup c45-01
```

```
Server:      137.194.2.16
Address:     137.194.2.16#53
```

```
Name:   c45-01.enst.fr
Address: 137.194.34.192
```

Avec mon ordinateur personnel:

```
$ nslookup pegASUS
```

```
Server:      137.194.2.16
Address:     137.194.2.16#53
```

```
** server can't find pegASUS: NXDOMAIN
```

Ce qui signifie que mon ordinateur n'appartient à aucun domaine.

11. En utilisant la commande `ifconfig` on obtient les adresses IP de l'ordinateur.

```
$ ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 2198 bytes 156733 (153.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 2198  bytes 156733 (153.0 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 137.194.92.119  netmask 255.255.248.0  broadcast 137.194.95.255
inet6 fe80::bbd4:a144:a3e8:b535  prefixlen 64  scopeid 0x20<link>
inet6 2001:660:330f:16:29d8:420d:5d31:2ac0  prefixlen 64  scopeid 0x0<global>
ether f8:94:c2:29:6e:db  txqueuelen 1000  (Ethernet)
RX packets 65049  bytes 74310406 (70.8 MiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 13379  bytes 2480537 (2.3 MiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

On peut voir par exemple que l'adresse IPv4 de l'ordinateur est 137.194.92.119. On peut également voir l'adresse IPv6, etc.

On peut aussi avoir ces informations sur de nombreux sites internet comme *www.adresseip.com*, *www.mon-ip.com* ou *www.localiser-ip.com*.

14. on fait un ping sur un des ordinateurs de l'école et on obtient le résultat suivant:

```
$ ping -c 10 c45-19
```

```
PING c45-19.enst.fr (137.194.34.210) 56(84) bytes of data.
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=1 ttl=64 time=1.53 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=2 ttl=63 time=1.17 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=3 ttl=63 time=4.97 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=4 ttl=63 time=3.39 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=5 ttl=63 time=3.57 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=6 ttl=63 time=2.83 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=7 ttl=63 time=4.25 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=8 ttl=63 time=3.94 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=9 ttl=63 time=4.15 ms
64 bytes from c45-19.enst.fr (137.194.34.210): icmp_seq=10 ttl=63 time=3.91 ms
```

```
--- c45-19.enst.fr ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 36ms
rtt min/avg/max/mdev = 1.174/3.372/4.966/1.144 ms
```

16. pour faire de l'arithmétique, on peut par exemple utiliser `expr`, qui donne un résultat immédiat après un seul appui sur <Entrée>.

```
$ expr 2 + 3
5
```

17. en utilisant une connexion ssh, on peut demander à une machine distante de faire ce calcul (un mot de passe est nécessaire).

18. la commande suivante permet d'enregistrer sa clé ssh public sur le server de l'école et de ne pas avoir à entrer de mot de passe pour l'authentification :

```
cat ~/.ssh/id_rsa.pub | ssh ablicq@ssh.enst.fr 'cat >> .ssh/authorized_keys'
```

III- Fichiers locaux/distants

19. On execute la commande suivante:

```
$ cd && pwd
/home/aurelien
```

20. La commande `echo bonjour > fperso.txt` permet de créer un fichier `fperso.txt` et d’y insérer le texte bonjour.

21. La commande `df` permet d’afficher et d’avoir des informations sur la localisation du fichier indiqué.

```
$ df fperso.txt
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sda1          915G      36G  832G   5% /home
```

Notre fichier est donc créé sur le disque dur de l’ordinateur.

22. Après avoir créer un fichier contenant “some text”, on exécute les commandes:

```
$ cat /tmp/aurelien/ftemp.txt
some text
```

```
$ df /tmp/aurelien/ftemp.txt
Sys. de fichiers blocs de 1K Utilisé Disponible Uti% Monté sur
tmpfs              4023980   43828    3980152   2% /tmp
```

Le système `tmpfs` (temporary file system) résidant dans la RAM, c’est là que notre fichier est stocké.

23. On crée dans le répertoire personnel le fichier `text.txt` contenant “mon texte sur NFS”.

```
$ cat ~/text.txt
mon texte sur NFS
```

24. On vérifie bien que depuis la machine *c128-32*, en se connectant en `ssh` sur les machines *c128-26* et *c128-34*, le fichier `text.txt` est présent et contient le texte “Mon texte sur NFS”.

25. En utilisant la commande adéquate, on créer le fichier `/tmp/ablicq/local.txt` contenant le text “Mon fichier local”.

26. En effectuant la commande `ls /tmp` sur les machines B et C, le dossier créé sur la machine A n’apparaît pas.

27. La commande `scp src:path dest:path` permet de transférer un fichier depuis la source vers la destination dans les chemins indiqués.

Par exemple, la commande `scp /tmp/ablicq/local.txt ablicq@c128-26.enst.fr:/tmp/ablicq/local.txt` transfère depuis l’ordinateur local, un fichier vers l’ordinateur *c128-26*

V- Ligne de commande depuis Java

34. La constante `ProcessBuilder.Redirect.INHERIT` permet de faire hériter le processus construit Java appelant. Par exemple, `pb.redirectError(ProcessBuilder.Redirect.INHERIT)`; redirige la sortie d'erreur du `ProcessBuilder pb` vers la sortie d'erreur du programme Java.

VI- Une gestion des timeout par le master

37. On implémente un thread `ReadThread` comme suit:

```
public class ReadThread extends Thread {

    /**
     * the InputStream from which to read data
     */
    private BufferedInputStream in;

    /**
     * the queue on which to put data
     */
    private LinkedBlockingQueue<String> queue;

    /**
     * a boolean to stop the thread when wanted
     */
    private boolean isRunning = true;

    /**
     * the constructor of the thread
     * @param in the InputStream from which to read data
     * @param queue the queue on which to put data
     */
    public ReadThread(BufferedInputStream in, LinkedBlockingQueue<String> queue) {
        this.in = in;
        this.queue = queue;
    }

    /**
     * read the input from the given input stream and relays the data to the given queue
     */
    @Override
    public void run() {
        while (isRunning) {
            try {
                if (in.available() > 0) {
                    // parse the input as a character and not an integer
                    char read = (char) in.read();
                    queue.put(Character.toString(read));
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * sets the isRunning variable to false to stop the thread

```

```

    */
    public synchronized void stopRun(){
        isRunning = false;
    }
}

```

VII- Deploy

38. On commence par créer une classe Deployer dont le rôle sera de déployer les fichier slave.jar aux ordinateurs. Cette classe possède deux méthodes: une pour tester la connexion aux hôtes indiqués, et une pour déployer un fichier jar passé en argument:

```

public class Deployer {
    /**
     * the list of hosts indicated in the configuration file
     */
    ArrayList<String> hostsList = new ArrayList<>();

    /**
     * Constructor of the Deployer.
     * Parse the given config file to the hostsList
     * @param configFile the config file containing the hosts to which we wish to deploy
     */
    public Deployer(String configFile) {
        parseHosts(configFile);
    }

    /**
     * Parse the given config file to the hostsList
     * @param configFile the config file containing the hosts to which we wish to deploy
     */
    private void parseHosts(String configFile){
        // read the hosts list from the config file
        try (Scanner in = new Scanner(new FileInputStream(configFile))) {
            while (in.hasNextLine()){
                String s = in.nextLine();
                hostsList.add(s.split("\\s")[0]);
            }
        } catch (Exception e) { e.printStackTrace();}
    }

    /**
     * Send the command 'hostname' to every host to test the connection
     */
    public void runTest(){
        for(String host : hostsList){
            ProcessBuilder pb = new ProcessBuilder(
                "ssh",
                "-o", "UserKnownHostsFile=/dev/null",
                "-o", "StrictHostKeyChecking=no",
                "ablicq@"+host,
                "hostname");

            pb.redirectOutput(ProcessBuilder.Redirect.INHERIT);
            pb.redirectError(ProcessBuilder.Redirect.INHERIT);
        }
    }
}

```

```

        try {
            Process p = pb.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/**
 * copy the given jarFile to every host at /tmp/ablicq/slave.jar
 * create the directory if needed
 * @param jarFile the jar file to deploy
 */
public void deploy(String jarFile){
    for (String host : hostsList){
        ProcessBuilder pb1 = new ProcessBuilder(
            "ssh",
            "-o", "UserKnownHostsFile=/dev/null",
            "-o", "StrictHostKeyChecking=no",
            "ablicq@"+host,
            "mkdir", "-p", "/tmp/ablicq",
        );

        ProcessBuilder pb2 = new ProcessBuilder(
            "scp", jarFile, "ablicq@"+host+":/tmp/ablicq/slave.jar"
        );

        pb1.redirectOutput(ProcessBuilder.Redirect.INHERIT);
        pb1.redirectError(ProcessBuilder.Redirect.INHERIT);

        pb2.redirectOutput(ProcessBuilder.Redirect.INHERIT);
        pb2.redirectError(ProcessBuilder.Redirect.INHERIT);

        try {
            Process p1 = pb1.start();
            p1.waitFor();
            pb2.start();

        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}

public static void main(String[] args) {
    Deployer deployer = new Deployer("config.txt");
    deployer.runTest();
    deployer.deploy("/tmp/ablicq/SLAVE.jar");
}
}

```

Tel qu'écrit ci-dessus, le programme s'exécute séquentiellement en raison du `p1.waitFor()` ; qui attend que le dossier soit créé avant d'y copier le fichier jar.

41. Cette question est très similaire à la précédente. Il faut cependant récupérer les fichiers de splits individuellement car la notation `splits/*` du shell ne fonctionne pas avec les `processBuilder`. J'ai également utilisé

`parallelStream().forEach()` qui permet d'effectuer des opérations parallèles sur les éléments d'une collection.

Résultats et Conclusion

Après implémentation du projet, j'ai effectué un test avec le fichier *forestier.txt* du dépôt *les codes en vigueur*, le séparant 16 en splits de taille 100K, et exécutant l'algorithme sur un panel de 28 slaves.

Les tests ont été peu convaincants car l'exécution a duré près de 40 minutes, contre quelques secondes pour l'algorithme séquentiel.

Durant l'exécution, j'ai remarqué que la structure `hosts.parallelStream.forEach(...)` que j'ai utilisée n'est capable de gérer que quatre éléments en parallèle, ce qui correspond au nombre de cœurs de ma machine master. L'exécution était donc séquentielle et, en ajoutant les délais de synchronisation entre le master et les slaves, cela explique les performances faibles de cette implémentation.

Afin de remédier à cela, il faudrait utiliser des Threads pour gérer la communication avec chaque slave. On pourrait aussi utiliser des sockets pour communiquer entre le master et les slaves ou entre les slaves. J'ai malheureusement manqué de temps à la fin du projet pour mettre en œuvre ces solutions.