Travaux pratiques: implémentation de Hadoop MapReduce “from scratch” en Java.

Etape 1: faire un programme séquentiel non parallélisé qui compte le nombre d'occurrences des mots dans un fichier.

Prérequis pour les questions suivantes:

- **programmer en Java**
  https://perso.telecom-paristech.fr/bellot/CoursJava/
- **manipuler l’environnement de développement Eclipse**
  https://perso.telecom-paristech.fr/bellot/CoursJava/tps/
- **lire et écrire dans un fichier en java**
  https://perso.telecom-paristech.fr/bellot/CoursJava/JavaIO.html

1. Premier comptage en séquentiel pur

Implémentez un logiciel en java qui compte le nombre d’occurrences des mots d’un fichier d’entrée de manière non parallélisée (monothread, une seul thread), en utilisant un seul processeur.
Quelle structure de donnée est la plus pertinente pour stocker les résultats: List, HashMap ou HashSet ou une autre ? Pour quelle raison ?

Testez votre programme avec un fichier d’entrée *input.txt* avec comme contenu:

```
Deer Beer River
Car Car River
Deer Car Beer
```

Résultat:

```
Deer 2
Beer 2
River 2
Car 3
```

2. Premier tri en séquentiel pur

Modifiez votre programme pour trier par nombre d'occurrences:

Résultat:

```
Car 3
Deer 2
```

```
Beer 2
River 2
```

3. Deuxième tri alphabétique en séquentiel pur

Modifiez le programme pour trier alphabétiquement pour les mots à égalité du nombre d'occurrences:

Résultat:
Car 3
Beer 2
Deer 2
River 2

4. Test du programme séquentiel sur le code forestier de Mayotte

Testez ensuite votre programme avec le code forestier de Mayotte disponible sur github *forestier_mayotte.txt* :
https://github.com/legifrance/Les-codes-en-vigueur
Votre programme a-t-il fonctionné du premier coup ?
Vérifiez en ouvrant le fichier texte qu'il contient bien du texte et non du code HTML.
Ne perdez pas de temps à corriger les éventuelles erreurs dues aux caractères spéciaux ou à des mots suspects ou illisibles (de toutes façons par la suite il y aura du chinois dans le texte).

5. Les 50 mots du code de la déontologie de la police nationale

Testez votre programme avec le code de déontologie de la police nationale disponible sur github *deontologie_police_nationale.txt* : https://github.com/legifrance/Les-codes-en-vigueur
De même ne perdez pas de temps à filtrer les caractères spéciaux ou autres mots bizarres. Pourquoi ? Car nous travaillerons ensuite sur des textes en chinois, japonais, arabe et d'autres langues. Si vous implémentez une étape de filtrage ici en français elle ne servira à rien par la suite. Quels sont les 5 premiers mots (qui ressemblent à des mots) parmi les 50 premiers de la liste triée résultat ? Gardez la réponse pour l'intégrer au rapport.

6. Les 50 mots du code du domaine public fluvial

Testez votre programme avec le code du domaine public fluvial *domaine_public_fluvial.txt*.
Quels sont les 5 premiers mots (qui ressemblent à des mots) parmi les 50 premiers de la liste triée résultat ? Gardez la réponse pour l'intégrer au rapport.

7. Les 50 mots du code de la santé publique

Testez votre programme avec le code de la santé publique *sante_publique.txt*.

Quels sont les 5 premiers mots (qui ressemblent à des mots) parmi les 50 premiers de la liste triée résultat ? Gardez la réponse pour l'intégrer au rapport.

8. Chronométrage du programme séquentiel

Chronométrer votre programme sur le code de la santé publique.

Chronométrage possible avec:
long startTime = System.currentTimeMillis();
...
long endTime   = System.currentTimeMillis();
long totalTime = endTime - startTime;

Combien de temps faut-il pour chacune des étapes:

- Compter le nombre d'occurrences
- Tri (par nombre d'occurrences et alphabétique)

Gardez la réponse pour l'intégrer au rapport.

9. Travailler sur des plus gros fichiers

Testez votre programme sur un cas réel: un extrait de toutes les pages internet transformées au format texte brut (format WET). Toutes les pages sur internet au format texte sont disponibles sur http://commoncrawl.org/the-data/get-started/ : chaque mois, environ 3 milliards de pages web, soit 250 To de données sont stockées. Ces données sont disponibles par tranche de moins d'1Go environ, vous travaillerez sur une tranche de 380Mo.
J'ai choisi une tranche en particulier pour avoir une comparaison entre nous (vous pouvez tester sur d'autres tranches si vous voulez). Téléchargez cette tranche ici:
https://commoncrawl.s3.amazonaws.com/crawl-data/CC-MAIN-2017-13/segments/1490218189495.77/wet/CC-MAIN-20170322212949-00140-ip-10-233-31-227.ec2.internal.warc.wet.gz
Décompressez et obtenez le fichier `CC-MAIN-20170322212949-00140-ip-10-233-31-227.ec2.internal.warc.wet`
Il s'agit d'une tranche contenant un ensemble de sites internet au format texte brut (WET).
Testez votre programme avec ce fichier en entrée. Chronométrez-le.
Gardez la réponse pour l'intégrer au rapport.
Si vous avez un erreur du type `java.lang.OutOfMemoryError` vous devez augmenter la taille mémoire de la machine virtuelle Java. Pour cela, sous Eclipse, suivez ce tutoriel: http://www.planetofbits.com/eclipse/increase-jvm-heap-size-in-eclipse/ . Par la suite, en ligne de commande, vous devrez ajouter l'option -Xms avec la valeur qui convient pour toutes les commandes java. Par exemple, java -Xms1000m pour avoir 1000Mo de mémoire alloués pour cette commande java.

10. Préparez un document de travail qu'il faudra rendre à la fin de l'unité d'enseignement.

Créez un document dans lequel vous allez expliquer votre implémentation au fur et à mesure. Vous obtiendrez également par la suite (dans les prochaines étapes) les mesures de chronométrage et calculerez le speedup (l'accélération) et la portion de code parallèle (le taux de parallélisation) à chaque fois avec vos nouvelles mesures. Pour comparer deux mesures, il faudra bien faire attention d'avoir le même jeu de données en entrée et le même résultat en sortie... Vous devez prouver que vous obtenez le même résultat en sortie en utilisant un autre logiciel (qui fait de la recherche du nombre de mots) et piocherez au hasard des mots pour savoir si dans le cas séquentiel et dans le cas réparti vous avez les même résultats.

Etape 2: travailler avec plusieurs ordinateurs en réseau.

Prérequis et documentation pour les questions suivantes :

- **avoir un compte et pouvoir se connecter aux machines de l'école (contacter la DSI de l'école dans le cas contraire)**
- **lancer un interpréteur de commande (console linux , shell) pour taper des commandes:**
  https://fr.wikipedia.org/wiki/Interpréteur_de_commandes
  https://fr.wikipedia.org/wiki/Shell_Unix
- **connaître quelques commandes de base sous Linux**
  https://fr.wikipedia.org/wiki/Commandes_Unix
- **https://prod-edx-mktg-edit.edx.org/course/linux-basics-the-command-line-interface**
- **savoir qu'un ordinateur a un nom d'hôte (hostname) et plusieurs adresses IP:**
  https://en.wikipedia.org/wiki/Hostname
  https://fr.wikipedia.org/wiki/Adresse_IP
- **savoir qu'un ordinateur peut faire partie d'un domaine (comme à l'école, le domaine enst.fr ou le domaine telecom-paristech.fr)**
  https://fr.wikipedia.org/wiki/Nom_de_domaine
- **savoir qu'un nom peut être transformé en adresse IP (et inversement) par un serveur qui gère le système du nom de domaine (DNS, Domain Name System)**
  https://fr.wikipedia.org/wiki/Domain_Name_System
- **se connecter à distance à un ordinateur avec SSH en ligne de commande**
  https://fr.wikipedia.org/wiki/Secure_Shell
  http://www.commentcamarche.net/faq/74-se-connecter-a-distance-avec-ssh-linux

1. Nom court, nom long

Quel est le nom COURT de votre ordinateur (le nom simple sans le domaine) ? quel est le nom LONG de votre ordinateur (le nom avec le domaine) ? Comment les connaître en ligne de commande ? Sur les ordinateurs de l'école, est-il possible d'obtenir ces noms autrement qu'en ligne de commande ? Ajoutez les réponses à votre rapport.

2. Adresse ip

Comment connaître les adresses (plusieurs) IP de votre ordinateur en ligne de commande ? Autrement (en passant par un site internet par exemple) ? Ajoutez les réponses à votre rapport.

3. Du nom vers l'IP

Comment à partir du nom d'un ordinateur, obtenir les adresses IP en ligne de commande ? Ajoutez les réponses à votre rapport.

4. De l'IP vers le nom

Comment, à partir d'une adresse IP, obtenir les noms associés en ligne de commande ? Ajoutez les réponses à votre rapport.

5. Ping pong à l'intérieur!

Testez la communication avec d'autres ordinateurs (pas le vôtre) depuis le réseau de l'école en utilisant la commande ping (pour arrêter le ping faire CTRL + C). suivi du nom court, du nom long, de l'IP. Les trois méthodes fonctionnent-elles ? Ajoutez les réponses à votre rapport.

6. Ping pong à l'extérieur

Si vous effectuez le ping depuis un réseau différent, il est possible que celui ne fonctionne pas (filtrage des accès vers le réseau de l'école depuis un réseau extérieur), contactez la DSI pour mettre en place une connection VPN / OpenVPN afin d'être sur le même réseau que les machines en salle de TP.

7. Calculer en ligne de commande sur l'ordinateur local

Comment lancer un calcul en ligne de commande sur votre ordinateur (par exemple 2 + 3) ? Parmi les multiples réponses possibles, lesquelles permettent de lancer le calcul et d'obtenir le résultat en appuyant une seule fois sur la touche <Entrée> ? Ajoutez les réponses à votre rapport.

8. Calculer en ligne de commande sur un ordinateur distant

Comment lancer un calcul (par exemple 2 + 3) en ligne de commande sur un autre ordinateur (à distance) ? Il faudra certainement vous authentifier avec un mot de passe. Comment obtenir le résultat du calcul immédiatement après avoir tapé son mot de passe ? Ajoutez les réponses à votre rapport.

9. Calculer à distance sans mot de passe

Comment lancer un calcul à distance en utilisant SSH sans taper le mot de passe et en une seule ligne de commande (c'est à dire qu'on appuie sur <Entrée> et on a le résultat directement)?
Attention: en utilisant vos ordinateurs personnels(a priori ce n'est pas le cas sur les ordinateurs de l'école), il faut parfois vérifier le "fingerprint" avec le message:

The authenticity of host 'mint.phcomp.co.uk (78.32.209.33)' can't be established.
RSA key fingerprint is 6a:de:e0:af:56:f8:0c:04:11:5b:ef:4d:49:ad:09:23.
Are you sure you want to continue connecting (yes/no)? yes

Pour ne plus avoir à rentrer quoi que ce soit dans le terminal, il faut donc désactiver la vérification de ce "fingerprint", sinon vous aurez l'erreur

Host key verification failed

Suivez ce guide pour désactiver la vérification de "fingerprint":

https://www.shellhacks.com/disable-ssh-host-key-checking/

De plus, si vous travaillez depuis votre ordinateur personnel, pour ne pas avoir à taper le mot de passe à chaque fois, il faut générer une clé privée et une clé publique SANS

PASSPHRASE (ne pas mettre de mot de passe lors de la génération des clés) et installer la clé publique sur un ordinateur de l'école, en copiant le contenu de la clé publique dans le fichier authorized_keys. Ce fichier authorized_keys est a placer dans le dossier .ssh (attention, il y a un point devant ssh ".ssh"), lui même situé dans votre dossier personnel sur les machines de l'école. Pour cela, suivez un guide sur internet de création de clés publiques privées pour SSH. Attention, votre clé privée doit également être située dans le dossier .ssh de votre ordinateur personnel. La bonne configuration est donc: avoir une clé privée sur l'ordinateur personnel dans le dossier .ssh de votre ordinateur personnel et avoir une clé publique sur un ordinateur de l'école dans le dossier .ssh d'un ordinateur de l'école. Attention: sur Windows, il est parfois compliqué de créer le dossier .ssh. Sur Windows, il est également compliqué de générer des clés avec puttygen car le format des clés n'est pas compatible avec openssh utilisé à l'école. Puttygen a une option de conversion du format des clés (dans le menu conversions) pour convertir vers le fomat openssh. Le plus simple est de générer les clés sur une machine de l'école puis de copier la clé privée sur votre ordinateur perso plutôt que de générer la clé sur votre ordi perso... Demandez de l'aide à vos confrères experts en cas de pb. Ne restez pas coincé longtemps sur cette étape!

Etape 3: travailler avec des fichiers locaux ou sur un serveur NFS.

Prérequis et documentation pour les questions suivantes :

- **pouvoir transférer un fichier d'un ordinateur à un autre en utilisant la commande SCP (Secure Copy): https://fr.wikipedia.org/wiki/Secure_copy**
- **connaître l'architecture d'un système de fichier NFS (Network File System) : https://fr.wikipedia.org/wiki/Network_File_System**

1. Chemin absolu

Quel est le chemin absolu de votre répertoire personnel, votre *home directory* ? (commandes "`cd`" puis "`pwd`")

2. Un fichier dans le répertoire personnel

Créez un fichier fperso.txt contenant le texte "bonjour" dans votre répertoire personnel (sur un ordinateur de l'école).
Vérifiez le contenu du fichier avec cette commande exactement:

```
cat ~/fperso.txt
```

3. Ou se trouve le fichier dans le répertoire personnel

Ce fichier est-il sur le disque dur de l'ordinateur ou autre part ? Comment savoir où est stocké physiquement ce fichier, à l'aide de quelle commande ?

4. Un dossier et un fichier dans le répertoire temporaire

Créez un dossier /tmp/<votre nom d'utilisateur> en remplaçant <votre nom d'utilisateur>.
Créez un fichier ftemp.txt dans le répertoire /tmp/<votre nom d'utilisateur> .
Vérifiez le contenu du fichier avec cette commande **exactement**:
`cat /tmp/<votre nom d'utilisateur>/ftemp.txt`
Ce dossier et ce fichier sont-ils sur le disque dur de l'ordinateur ou autre part ? Comment savoir où sont stockés physiquement ces éléments, à l'aide de quelle commande ?

5. Trois ordinateurs A B C. On commence avec A. Utilisation du serveur NFS.

Pour les questions suivantes, utilisez trois ordinateurs: A, B C.
Connectez vous physiquement (avec un clavier, une souris et un écran) sur l'ordinateur A.
Sur A, créez un fichier text.txt contenant le texte "mon texte sur NFS" dans votre répertoire personnel.
Vérifiez que le fichier existe et vérifiez son contenu. Pour cela, sur A, utilisez la commande :
`cat ~/text.txt`

6. Trois ordinateurs A B C. On continue sur B et sur C. Utilisation du serveur NFS.

Connectez-vous à B (physiquement ou à distance) et vérifiez que le fichier text.txt est également présent dans votre répertoire personnel. Pour cela, sur B, utilisez la commande :
`cat ~/text.txt`
De même, connectez-vous à C et vérifiez que text.txt est aussi présent.
Remarquez que vous n'avez pas copié le fichier mais qu'il est présent sur A, B et C grâce au serveur NFS.

7. Trois ordinateurs A B C. On commence avec A. Utilisation des disques locaux.

Déconnectez vous de B et de C et revenez sur l'ordinateur A.
Sur A, créez un dossier /tmp/<votre nom d'utilisateur> et un fichier local.txt contenant le texte "mon texte sur disque local" dans ce dossier /tmp/<votre nom d'utilisateur>.
Vérifiez que le fichier existe et vérifiez son contenu. Pour cela, sur A, utilisez la commande :
`cat /tmp/<votre nom d'utilisateur>/local.txt`

8. Trois ordinateurs A B C. On continue sur B et sur C. Utilisation des disques locaux.

Connectez-vous à B et C (physiquement ou à distance) et vérifiez que le dossier <votre nom d'utilisateur> ainsi que le fichier local.txt **ne sont pas** présent dans /tmp . Pour cela vérifiez avec la commande:
`ls /tmp`

9. Depuis A, copier de A vers B avec les disques locaux.

Comment, **à partir de A**, transférer le fichier /tmp/local.txt **sur B** (dans `/tmp/<votre nom d'utilisateur>/local.txt`) en utilisant scp ? Vérifiez que le fichier est bien présent sur B. Attention: si vous avez une erreur "no such file or directory" (ou l'équivalent français), vous devez d'abord créer le répertoire `/tmp/<votre nom d'utilisateur>/` avec la commande `mkdir -p` associée à un ssh pour l'ordinateur distant.

10. Depuis A, copier de B vers C avec les disques locaux.

Comment, **à partir de A**, transférer le fichier **de B** (depuis `/tmp/<votre nom d'utilisateur>/local.txt`) **vers C** (dans `/tmp/<votre nom d'utilisateur>/local.txt`) ? Vérifiez que le fichier est bien présent sur C. De même que la question précédentes, vous devez créer les répertoires `/tmp/<votre nom d'utilisateur>/` correspondants.

Etape 4: lancer des programmes java à distance manuellement.

Prérequis et documentation pour les questions suivantes :

- **exporter un .JAR exécutable (Runnable JAR File)**
- **utiliser les packages et savoir lancer un programme Java en ligne de commande**

Un premier programme SLAVE sous Eclipse
Faire un programme Java nommé "SLAVE" qui calcule 3+5, affiche le résultat, sous Eclipse (Pour lancer Eclipse: Menu applications>développement), lancer ce programme dans Eclipse (flèche verte "exécuter")

1. Exportation en JAR

Exporter le programme Java en slave.jar exécutable (Java ARchive dite Runnable) sous Eclipse. Attention de bien vérifier que le JAR est de type "Runnable"/"exécutable".

2. Exécution sur disque dur local

Créez le répertoire `/tmp/<votre nom d'utilisateur>/`
Copiez slave.jar exécutable dans le répertoire `/tmp/<votre nom d'utilisateur>/`
Testez et Lancer le slave.jar en ligne de commande sur votre ordinateur local.

3. Copie du JAR et exécution distante

Depuis la machine A contenant `/tmp/<votre nom d'utilisateur>/slave.jar`
Créez à distance sur la machine B (s'il n'existe pas) un répertoire /tmp/`<votre nom d'utilisateur>/`
Copiez slave.jar sur la machine B dans le répertoire /tmp/`<votre nom d'utilisateur>/`
Exécutez à distance (depuis A sur la machine B) le slave.jar.
Quelle est la commande tapée pour effectuer cette dernière action ?

Attention: si le JAR est créé sur votre machine personnelle, vous risquez de produire un JAR avec une version supérieure de JAVA que celle installée sur les ordinateurs de l'école. Pour remédier à ce problème, configurez votre projet JAVA pour qu'il soit compatible avec la version 1.8 de JAVA. Sous Eclipse, dans les propriétés de votre projet, allez dans Java Compiler et décochez la case "Use compliance from execution environment XXXX" et puis dans le menu déroulant choisissez 1.8. Vous devrez re-générer le JAR pour obtenir un JAR compatible avec la version 1.8 de Java.

Etape 5: lancer des programmes en ligne de commande depuis java et afficher la sortie standard et la sortie d'erreur.

Prérequis et documentation pour les questions suivantes :

- utiliser le processBuilder en java **http://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html .**
- lancer un exécutable (ou une commande) en ligne de commande depuis un programme écrit en java
- connaître les sorties standard et les sorties d'erreurs

1. Un programme MASTER java qui lance un autre programme en ligne de commande!

Ecrire un programme java nommé "MASTER" qui lance la commande suivante en utilisant ProcessBuilder:

```
ls -al /tmp
(vous pouvez aussi tester cette commande dans un terminal avant)
```

Récupérer et afficher la sortie de cette commande.
Vous devez utiliser ProcessBuilder de cette façon:

```
ProcessBuilder pb = new ProcessBuilder("ls", "-al", "/tmp");
```

2. Un programme MASTER java qui gère les erreurs de lancement d'un autre programme en ligne de commande.

Modifiez votre programme"MASTER" pour qu'il affiche la sortie d'erreur en cas d'erreur lors de l'exécution de la commande. Testez la sortie d'erreur avec une commande qui échoue avec un sortie d'erreur. Essayez par exemple d'exécuter la commande "ls /jesuisunhero".
Explications: si on tape la commande "ls /jesuiunhero", le dossier /jesuisunhero n'existant pas, on aura une erreur de type "impossible d'accéder à /jesuisunhero: aucun fichier ou dossier de ce type." qui s'affiche dans la sortie d'erreur. En effet, il y a deux sorties: les sorties standards (sans erreur) et les sorties d'erreurs.
Vous devez utiliser ProcessBuilder de cette façon:

```
ProcessBuilder pb = new ProcessBuilder("ls", "/jesuisunhero");
```

Sur `pb`, vous pouvez récupérer le flux de la sortie standard et le flux de la sortie d'erreur.

3. Un programme MASTER java qui lance un slave.jar en ligne de commande.

Modifiez votre programme “MASTER” pour qu’il lance “SLAVE”, c’est à dire slave.jar situé sur la même machine que “MASTER” dans le dossier
`/tmp/<votre nom d’utilisateur>/slave.jar`

Etape 6: gérer les timeout du MASTER.

1. Un SLAVE qui simule un calcul de 10 secondes.

Modifiez votre programme SLAVE pour qu’il simule une attente de 10 secondes avant d’afficher le résultat du calcul 3+5. Pour cela utilisez
`Thread.sleep(10000);`
Vérifiez le bon fonctionnement du SLAVE et constatez qu’il y a 10 secondes entre le démarrage du SLAVE et l’affichage du résultat. Attention de ne rien afficher avant les 10 secondes pour que la question suivante fonctionne correctement.
Générez de nouveau le slave.jar. Copiez-le en écrasant le slave.jar dans le dossier
`/tmp/<votre nom d’utilisateur>/slave.jar`
Testez le lancement à partir de MASTER.

2. Gérer les timeout au niveau du MASTER.

Modifier le MASTER pour qu’il attende que quelque chose soit écrit dans la sortie standard (sans erreur) ou dans la sortie d’erreurs du SLAVE pendant un certain temps maximum. Au bout du temps imparti le MASTER considère un timeout.
Il arrête les éventuels threads (si vous utilisez des threads - non obligatoire) s’occupant des sorties et/ou stoppe le processus créé avec ProcessBuilder.
Vous avez la solution ci-dessous pour implémenter les TESTs suivants:
TEST1 : Testez le bon fonctionnement du timeout en lançant le SLAVE avec un timeout de 2 secondes sur les sorties (standard et d’erreur). Le timeout étant plus court (au niveau du MASTER 2 secondes) que le temps de calcul du SLAVE (10 secondes), le MASTER doit arrêter les éventuels threads (si vous en utilisez) et le processus.
TEST 2: Testez ensuite avec un timeout de 15 secondes, il ne devrait pas y avoir de timeout.
TEST 3: Testez ensuite en changeant le SLAVE pour qu’il écrive non plus dans la sortie standard (sans erreur) mais dans la sortie d’erreur. Pour cela, remplacez dans le Slave les *System.out.print*… par *System.err.print*…

Aide:
SOLUTION 1 (très difficile) - passez à la SOLUTION 2 si vous êtes débutant : Vous utilisez deux threads (processus légers) pour chaque processus lourd (lancé avec ProcessBuilder). Un thread s’occupe de lire la sortie standard et l’autre la sortie d’erreur. Une solution pour gérer les timeout consiste à utiliser une structure *ArrayBlockingQueue* (taille fixe) ou *LinkedBlockingQueue* (taille dynamique) donnée en paramètre de chaque thread s’occupant de lire la sortie standard (sans erreur) ou la sortie d’erreur. Chaque thread va écrire ce qui est lu depuis les sorties dans cette structure en utilisant la méthode *put*. Le timeout de 2 secondes peut alors être paramétré lors de la récupération des éléments insérés dans la structure en utilisant la méthode *poll* sur l’*ArrayBlockingQueue* ou la *LinkedBlockingQueue* de cette manière: *poll(2, TimeUnit.SECONDS);*

SOLUTION 2 (très facile) : Une solution consiste à
1) rediriger la sortie d'erreur dans la sortie standard à l'aide de la méthode pb.redirectErrorStream(true) du ProcessBuilder
2) utiliser *pb.inheritIO();* pour rediriger les sorties du process dans le MASTER (c'est à dire que le MASTER va afficher les sorties du SLAVE)
3) démarrer le process avec Process p = pb.start();
4) attendre la fin du process avec un waitFor sur le process qui génère un timeout en utilisant
boolean b = p.waitFor(3, TimeUnit.SECONDS); //si b vaut false il y a un timeout
Attention: dans cette solution et l'utilisation de inheritIO() , vous ne pourrez plus récupérer et/ou traiter la sortie standard ou la sortie d'erreur du process: par exemple, `p.getInputStream()` renverra un flux vide et `p.getErrorStream()` renverra aussi un flux vide.

SOLUTION 3 (facile): il s'agit d'utiliser `pb.redirectErrorStream(true)` et une boucle pour lire les lignes reçues depuis `p.getInputStream()` sans utiliser inheritIO()

SOLUTION 4 (moins facile): il s'agit d'utiliser deux boucles pour lire les lignes reçues depuis `p.getInputStream()` et `p.getErrorStream()` sans utiliser inheritIO()

Etape 7: déployer automatiquement le programme SLAVE sur un ensemble de machines.

1. Un programme DEPLOY : Test de connection SSH multiple

Créer un fichier texte à la main contenant : les adresses IP et/ou les noms des machines que nous voulons utiliser pour notre système réparti (par exemple toutes les machines de cette salle de TP), avec un nom ou une IP par ligne dans le fichier.
Créer un nouveau programme java DEPLOY qui lit ce fichier ligne par ligne et teste si la connection SSH fonctionne bien sur chacune des machines. Attention, il s'agit bien d'un nouveau programme qui est séparé de MASTER ou SLAVE, vous ne l'exécuterez qu'en cas de mise à jour du SLAVE. Ceci permet de vérifier qu'une machine n'est pas éteinte ou qu'il y a un problème de connection (par exemple).
Pour vérifier que la connection fonctionne bien, vous pouvez faire afficher le nom de la machine distante (en exécutant la commande hostname à distance) et vérifier que l'affichage a effectivement lieu, sans erreurs. Réutilisez des parties de codes de la cinquième étape.
Votre programme DEPLOY lance-t-il les connections de manière séquentielle (les unes après les autres) ou de manière parallèle?

2. Un programme DEPLOY : copie de slave.jar multiple

Modifier votre programme DEPLOY pour qu'il copie le slave.jar dans `/tmp/<votre nom d'utilisateur>/` sur les ordinateurs dont la connection SSH fonctionne.
Pour cela, vous devez utiliser la commande `mkdir -p` pour créer les répertoires dans /tmp s'ils n'existent pas déjà, attendre que le mkdir se termine puis copier avec scp le fichier slave.jar. Comment faites-vous pour attendre que le mkdir se termine correctement?
Vérifiez ensuite manuellement que le fichier a bien été copié sur les ordinateurs distants.

Attention de bien attendre la fin du mkdir avant de lancer le scp (on ne veut pas avoir de copie avant que le dossier soit effectivement créé).
Lors des copies, faites attention au caractère " / " à la fin d'un chemin :
/tmp/toto est un chemin vers un fichier nommé toto
/tmp/toto/ est un chemin vers un dossier nommé toto.
Votre programme DEPLOY lance-t-il les copies de manière séquentielle (les unes après les autres) ou de manière parallèle?

Etape 8: nettoyer un ensemble de machines avec CLEAN.

1. Un programme "CLEAN" qui nettoie les machines distantes.

Créez un nouveau programme CLEAN (différent de MASTER, SLAVE ou DEPLOY) qui efface votre dossier `/tmp/<votre nom d'utilisateur>/` sur les ordinateurs dont la connection SSH fonctionne. Pour cela, vous utiliserez le même fichier texte écrit à la main et utilisé par DEPLOY contenant : les adresses IP et/ou les noms des machines que nous voulons utiliser pour notre système réparti (par exemple toutes les machines de cette salle de TP), avec un nom ou une IP par ligne dans le fichier.
CLEAN lit ce fichier ligne par ligne et efface sur chacune des machines votre dossier créé dans le dossier temporaire, en lançant la commande à distance `rm -rf /tmp/<votre nom d'utilisateur>/`. Attention de bien attendre la fin de la commande rm -rf pour être sûr que l'effacement a bien été effectué.
Votre programme CLEAN lance-t-il les commande d'effacement de manière séquentielle (les unes après les autres) ou de manière parallèle?

2. Vérification du DEPLOY et du CLEAN

Vérifiez manuellement que l'effacement des dossiers a bien lieu. Ce programme vous permet de nettoyer un ensemble de machines avant de relancer un calcul. A partir de maintenant, vous pouvez déployer votre application en utilisant DEPLOY et vous pouvez nettoyer votre application en utilisant CLEAN. Vérifiez donc que DEPLOY suivi CLEAN fonctionne correctement.

Etape 9: lancer le programme SLAVE sur un ordinateur à distance.

1. CLEAN et DEPLOY

Faites un CLEAN. Faites un DEPLOY sur une seule machine (modifier la liste des machines pour n'en mettre qu'une) . Vous devriez avoir la dernière version de slave.jar déployée sur une seule machine.

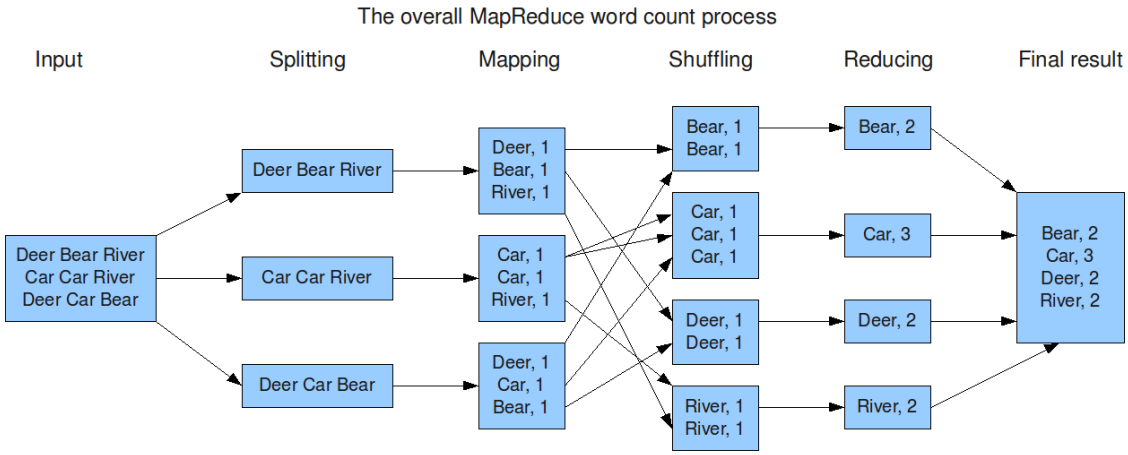
2. Master lançant Slave à distance.

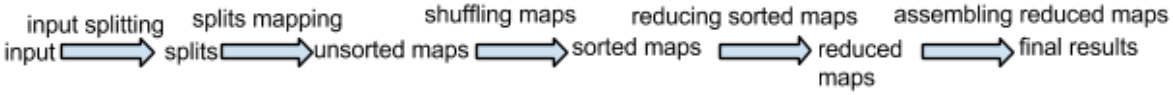
Modifier votre programme "MASTER" pour qu'il lance à distance le programme "SLAVE" déjà déployé avec le programme "DEPLOY" sur la seule machine distante.

Etape 10: MapReduce - SPLIT et MAP

Prérequis et documentation pour les questions suivantes :

- **connaître l'architecture de MapReduce**
- **Connaître la convention des nom des fichiers pour notre TP**



- 

- **connaître le vocabulaire: S UM SM RM (Splits, Unsorted Maps, Sorted Maps, Reduced Maps) - voir en cours**

1. Un MASTER qui déploie les splits

Créez trois fichiers correspondants à des "splits" dans le répertoire temporaire. Dans un premier temps, créez ces fichiers manuellement.
`/tmp/<votre nom d'utilisateur>/splits`
S0.txt S1.txt S2.txt.
S0.txt contient:
`Deer Beer River`
S1.txt contient:
`Car Car River`
S2.txt contient:
`Deer Car Beer`

Modifiez votre MASTER pour qu'il copie les fichiers de splits dans 3 ordinateurs différents. Pour cela vous utiliserez le fichier précédemment créé qui contient la liste des machines que vous voulez utiliser pour votre projet.
Attention, le répertoire `/tmp/<votre nom d'utilisateur>/splits` doit être créé sur les 3 ordinateurs s'il n'existe pas. Cette création peut se faire de manière automatique (en créant de manière programmatique ces répertoires). Attention de faire attention de bien attendre que la création des dossiers soit bien effectuée avant de lancer la copie des splits. Comment attendez-vous que la création des dossiers soit bien effectuée avant de copier véritablement les fichiers?

De la même manière que le programme DEPLOY, le MASTER va copier ces splits vers 3 ordinateurs dont la connection SSH fonctionne.
Votre programme MASTER lance-t-il les copies de manière séquentielle (les unes après les autres) ou de manière parallèle?

2. Un SLAVE qui fait la phase de map

Modifiez le SLAVE pour qu'il calcule un map à partir d'un split.
Pour cela, il prend un mode de fonctionnement en argument: 0 correspond au calcul du map à partir d'un split, puis un nom de fichier "Sx.txt" en entrée depuis le dossier `splits` et calcule un fichier "UMx.txt" en sortie dans le dossier `maps`, avec x variant (ici de 1 à 3). De la même manière que précédemment, le dossier `maps` doit être créé avant de pouvoir écrire des fichiers dedans. Vous devez attendre que le dossier `maps` soit créé avant de travailler avec. Comment attendez-vous que le dossier `maps` soit bien créé avant de travailler avec?
Le nom du fichier sera donné comme argument args du main:
public static void main (String[] args)

Testez dans un terminal le slave.jar comme suit:

```
cd /tmp/<votre nom d'utilisateur>/
java -jar slave.jar 0 /tmp/<votre nom d'utilisateur>/splits/S0.txt
```

Le fichier `/tmp/<votre nom d'utilisateur>/maps/UM0.txt` doit être créé contenant
Dear 1
Beer 1
River 1

Testez le fonctionnement de votre SLAVE avec le fichier S1.txt contenant
Car Car River
Testez dans un terminal le JAR comme suit:

```
cd /tmp/<votre nom d'utilisateur>/
java -jar slave.jar 0 /tmp/<votre nom d'utilisateur>/splits/S1.txt
```

Le fichier `/tmp/<votre nom d'utilisateur>/maps/UM1.txt` doit être créé contenant
Car 1
Car 1
River 1

Attention: Comme votre SLAVE est modifié, utilisez le DEPLOY pour déployer la nouvelle version.

Posez-vous la question: pourquoi retrouvons-nous deux lignes
Car 1
Car 1
Au lieu d'une seule ligne
Car 2 ?

Un indice: la phase de reduce qui arrivera plus tard, fera une addition des valeurs. Dans ce cas, la "fonction" de reduce est très simple: c'est une grande addition. Imaginez une

fonction de reduce beaucoup plus complexe qui effectue un algorithme complexe appliqué sur toutes les valeurs...

3. Un MASTER qui lance les SLAVE pour la phase de map.

Modifiez le MASTER pour qu'il lance la phase de map sur plusieurs machines et affiche "MAP FINISHED". Pour cela vous utiliserez le fichier précédemment créé qui contient la liste des machines que vous voulez utiliser pour votre projet.
Pour bien synchroniser le MASTER avec les SLAVES, veillez à afficher "MAP FINISHED" qu'une fois tous les SLAVEs terminés et uniquement quand tous les SLAVEs sont terminés, PAS AVANT! Le MASTER doit donc attendre que SLAVEs se terminent correctement.
Comment faites-vous pour qu'un process lancé avec ProcessBuilder en Java attende la fin de l'exécution du processus ?
Votre programme MASTER lance-t-il les SLAVE de manière séquentielle (les uns après les autres) ou de manière parallèle?

Etape 11: MapReduce - SHUFFLE

1. Le MASTER qui prépare les SLAVE à la phase de shuffle.

Modifiez le MASTER pour qu'il envoie le fichier précédemment créé qui contient la liste des machines que vous voulez utiliser pour votre projet à tous les SLAVES utilisés pour la phase de map. Copiez ce fichier dans la cible suivante pour tous les SLAVES:
`/tmp/<votre nom d'utilisateur>/machines.txt`

2. Le SLAVE qui prépare la phase de shuffle.

Modifiez le SLAVE pour qu'il prépare la phase de shuffle en regroupant les clés, en calculant le "hash" pour chacune des clés et en créant un fichier ayant pour nom <hash>-<hostname>.txt dans le dossier `shuffles`. De la même manière que précédemment, le dossier `shuffles` doit être créé avant de pouvoir écrire des fichiers dedans. Vous devez attendre que le dossier `shuffles` soit créé avant de travailler avec. Comment attendez-vous que le dossier `shuffles` soit bien créé avant de travailler avec?
Le nom du fichier sera donné comme argument args du main:
public static void main (String[] args)

Attention: si le fichier <hash>-<hostname>.txt existe déjà, il ne faut pas l'écraser mais plutôt continuer d'écrire dedans.
Ce nom de fichier correspond au hash, obtenu grâce à la fonction de hashage de la classe String, explications ici https://fr.wikipedia.org/wiki/Java_hashCode()#La_fonction_de_hachage_de_la_classe_java.lang.String , calculé à partir de la clé; Le nom de la machine , lui, est obtenu grâce à l'instruction java suivante: java.net.InetAddress.getLocalHost().getHostName()

Pour cela, votre prend un mode de fonctionnement en argument: 1 correspond au calcul du hash , puis un nom de fichier "UMx.txt" en entrée depuis le dossier `maps` et calcule un fichier "<hash>-<hostname>.txt " en sortie dans le dossier `shuffles`.

Testez le fonctionnement de votre SLAVE avec le fichier UM1.txt contenant
Car 1
Car 1
River 1

Testez dans un terminal le JAR comme suit:

```
cd /tmp/<votre nom d'utilisateur>/
java -jar slave.jar 1 /tmp/<votre nom d'utilisateur>/maps/UM1.txt
```

Les fichiers suivants doivent être créés:
`/tmp/<votre nom d'utilisateur>/shuffles /.txt` créé contenant
Car 1
Car 1
`/tmp/<votre nom d'utilisateur>/shuffles /.txt` créé contenant
River 1