SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

PC3R - TME3: MapReduce en *Go*

Equipe Enseignante PC3R

14/02/2020

Lien vers les ressources de l'UE: `https://www-master.ufr-info-p6.jussieu.fr/2019/PC2R`
Lien vers les horaires des trains: `https://frama.link/pc3r-sncf`

Objectif: Manipuler des mécanismes de passage de messages utilisant des canaux synchrones.

Rendu: Le rendu final pour ce TME doit être une unique archive contenant les sources qui peuvent être un ou plusieurs fichiers `.go`

Evaluation: Le rendu est évalué sur:
- la pertinence de l'utilisation des mécanismes d'utilisation des canaux.
- la lisibilité du code.
- la justesse du choix des paramètres (nombre de travailleurs, temps additionnels de travail et/ou de reduction), permettant de démontrer le fonctionnement concurrent du système.

Description Générale du Système

Le processus du système sont divisés ainsi:
- un *lecteur* unique, qui lit, ligne par ligne, un fichier contenant des données, il envoie chaque ligne de données à un travailleur.
- un nombre fixé de *travailleurs*, chacun reçoit une ligne de données, la convertit en un *paquet*, envoie le paquet au serveur de calcul, attend de recevoir un paquet résultat, envoie le paquet résultat au *réducteur*, et se relance.
- un unique *serveur de calcul*, qui peut être contacté par les travailleurs mais qui gère les requêtes de manière **concurrentes**: il peut accepter un nombre arbitraire de requêtes à la fois et crée des sous-processus à la volée pour les traiter ; chaque requête contient un paquet, qui est transformé par le serveur et renvoyé à l'expéditeur.
- un unique *réducteur*, qui, de manière séquentielle, reçoit des paquets transformés et calcule une information à partir du contenu de tous les paquets qu'il a reçu. Il tourne mais peut recevoir à tout moment un signal de *fin du temps* quand c'est le cas, il envoie la valeur qu'il a calculée au processus principal.
- un processus *principal* chargé de créer les canaux de communications initiaux, de lancer les autres processus, d'attendre un certain temps, puis d'envoyer le signal de fin du temps au réducteur et enfin de récupérer et d'imprimer le résultat final.

Il est conseillé de commencer par faire un schéma du système, des différents composants et des différentes communications entre composants.

Particularités

Dans ce TME, le fichier de données va être le fichier des horaires d'arrivée et de départ en gare des trains de la SNCF, récupérable dans `stop_times.txt` de l'archive donnée en en-tête de cet énoncé.
- le lecteur lit les lignes de ce fichier et les envoie (sous forme de chaînes de caractères) aux travailleurs,
- les travailleurs fabriquent des paquets contenant un champ `arrivee`, initialisé à l'heure d'arrivée du train, un champ `depart`, initialisé à l'heure de départ du train, et un champ `arret` initialisé à la valeur 0.

- le serveur de calcul reçoit des paquets, et les modifie en calculant la durée de l'arrêt (heure de départ moins heure d'arrivée) et modifiant le champ `arret` du paquet pour qu'il contienne cette valeur et renvoit les nouveaux paquers au travailleur **correspondant** (un travailleur ne doit pas recevoir un paquet modifié d'un autre travailleur, c'est un point crucial pour l'évaluation).
- le réducteur accumule les durées d'arrêt des paquets qu'il reçoit dans une variable entière ; à la fin du temps, il envoie au processus principal la *durée d'arrêt moyenne* qui a été calculée.
- le temps d'attente du thread principal avant de déclarer la fin du temps est passé en paramètre en à l'exécutable en ligne de commande.