INF4410 – Systèmes répartis et infonuagique

TP2 – Services distribués et gestion des pannes

François Rajotte

11 octobre 2013

École Polytechnique de Montréal

Introduction

Le second travail pratique vous fera prendre conscience de plusieurs problèmes reliés à la gestion des pannes à l'intérieur d'un système réparti sur plusieurs machines. Les services distribués permettent de répartir le risque dû à un bris sur l'ensemble des machines plutôt que sur un seul point de rupture. Dans un système distribué, des pannes peuvent survenir à tout moment. Il faut donc que le service soit assuré même quand une anomalie survient. De plus, il faut parfois prévoir des mécanismes qui vérifient les résultats calculés par des machines non fiables. Par exemple, le projet Folding@home, qui permet à des ordinateurs personels de contribuer à une grappe de calul, doit s'assurer que les résultats ne sont pas entachés par des utilisateurs malicieux ou du matériel défectueux.

Prenez le temps de lire l'énoncé une fois attentivement avant de commencer, il y a beaucoup de petits détails à ne pas négliger.

Remise

- Méthode: Par Moodle, un seul membre de l'équipe doit remettre le travail, mais assurez vous d'inclure les deux matricules dans le rapport et le nom du fichier remis.
- Échéance: mercredi le 6 novembre 2013, avant 16h, voir le plan de cours pour les pénalités de retard.
- Format: Une archive au format .tar.gz contenant votre code et votre rapport contenant vos réponses aux questions. Ne remettez que les fichiers sources, pas les fichiers compilés. Pour les projets Eclipse, ne remettez que le projet, pas le workspace en entier.

Vous pouvez faire le laboratoire seul, mais le travail en binôme est encouragé. On vous demande d'inclure un *README* expliquant comment exécuter votre TP. Après la remise, il se peut que le chargé de laboratoire vous demande de faire une démonstration de votre travail, donc gardez votre code.

Barème

Respect des exigences et bon fonctionnement: 10 points
Rapport et réponses aux questions: 6 points
Clarté du code et commentaires: 4 points
Total: 20 points, valant pour 10% de la note finale du cours.

Jusqu'à 2 points peuvent être enlevés pour la qualité du français.

Spécifications

Généralités

On vous demande de développer un système distribué qui a comme tâche de compter l'incidence de chaque mot dans un texte. Un répartiteur devra s'occuper de distribuer le travail à faire entre les différents serveurs de calcul ainsi que de récupérer et colliger les résultats. Le répartiteur est également celui qui s'occupe de gérer la défaillance d'un des serveurs.

Répartiteur

Le répartiteur est le point d'entrée du système. C'est lui qui se connecte aux serveurs de calcul et soumet le travail que chacun d'eux doit effectuer. Il prend en paramètre le fichier d'entrée qui contient le texte à analyser et le fichier de sortie qui contiendra le résultat du calcul.

Exemple de fichier d'entrée:

```
bonjour un bonjour un deux
```

Exemple de fichier de sortie:

```
bonjour 2
deux 1
un 2
```

Modes de fonctionnement

Le système a la possibilité de travailler selon deux modes de fonctionnement.

- En mode sécurisé, le résultat des serveurs de calcul est considéré bon et valide. Le répartiteur n'a donc besoin que d'une seule réponse de la part d'un serveur.

- En mode non-sécurisé, le système ne fait pas confiance aux serveurs de calcul. Le répartiteur considère alors une réponse comme étant valide que si une majorité de serveurs de calcul sont d'accord sur la même réponse à un même travail.

Serveurs de calcul

Les serveurs de calcul sont responsables de recevoir des tâches du répartiteur, d'effectuer le calcul et de retourner la réponse au répartiteur. La tâche contient principalement la partie du texte à analyser. Lors de la réception d'une tâche, le serveur tente d'allouer les ressources nécessaires. Si les ressources sont insuffisantes, le serveur doit indiquer au répartiteur qu'il n'est pas en mesure d'effectuer le calcul.

Simulation des ressources

Pour les fins du travail pratique, les ressources disponibles à chaque serveur seront simulées à l'aide d'une fonction mathématique simple. Ces ressources sont mesurées en octets et chaque octet du texte contenu dans une tâche utilise un octet de ressources.

Chaque serveur *i* possède une quantité de ressources réservées q_i. Tant que la quantité de ressources utilisée est inférieure à q_i, la tâche peut s'effectuer sans problème. Si une tâche demande plus de ressources que la quantité réservée, le serveur tente d'utiliser plus de ressources mais cette opération peut échouer. Le taux d'échec croit linéairement de q_i(0%) à 10*q_i(100%). L'équation suivante permet de calculer ce taux d'échec (*T*) en fonction de la quantité de ressources réservées (q_i) et la quantité de ressources à utiliser (u_i) :

$$T = \frac{u_i - q_i}{9q_i} * 100\,\%$$

Le répartiteur ne connait pas les capacités de chaque serveur de calcul. Il doit donc s'assurer de distribuer des tâches de la bonne taille afin d'éviter de recevoir des échecs trop souvent. Notez également que les serveurs de calcul peuvent avoir des capacités de ressources qui varient d'un serveur à l'autre.

Pannes intempestives

En plus des tâches qui ne se réalisent pas à cause d'un manque de ressources, le système doit pouvoir réagir correctement si jamais un serveur de calcul est tué au beau milieu de l'exécution d'une tâche. Le répartiteur doit alors redistribuer le travail vers les autres serveurs toujours disponibles.

Serveur de calcul malicieux

En mode non-sécurisé, les serveurs de calcul peuvent retourner des réponses erronées. Parmi tous les serveurs de calcul, certains devront pouvoir être configurés de manière à agir malicieusement. Ceux-ci modifieront le décompte des mots pour donner un faux résultat. En autant qu'un nombre suffisant de serveurs agissent de bonne foi (>50%), le répartiteur devrait toujours être en mesure de récupérer le bon décompte. La fréquence à laquelle un serveur donne un faux résultat est configurable lors de son lancement allant de 0% (serveur de bonne foi) à 100% (serveur toujours malicieux)

Travail demandé

Pour ce travail pratique, vous êtes libres de choisir le langage et la technologie de communication que vous désirez, pourvu qu'on puisse exécuter votre application sur les ordinateurs du l4714.

Programme répartiteur

Vous devez implémenter le programme qui fait la répartition du travail entre les différents serveurs de calcul. Vous êtes libres de choisir comment découper le travail.

Programme serveur de calcul

Vous devez implémenter l'algorithme qui permet de compter les instances de chaque mot dans un texte. Assurez-vous que chaque serveur puisse être configuré indépendamment quant à sa quantité de ressources réservées (q_i) et son taux de réponse erronée (en mode non sécurisé).

Tests de performance – mode sécurisé

On vous demande de créer une instance du système en mode sécurisé en utilisant un répartiteur et trois serveurs de calcul. Choisissez un fichier d'entrée suffisamment gros pour que le calcul prenne au moins quelques secondes à effectuer.

Configurez la quantité de ressources réservées des serveurs de calcul de façon à ce que la quantité du premier soit deux fois plus grande que celle du deuxième et quatre fois plus grande que celle du troisième :

$q_1 = 2q_2 = 4q_3$

Faites varier la quantité de ressources disponibles des serveurs afin d'observer les changements sur le temps d'exécution, tout en respectant l'équation ci-haut.

Présentez le graphique du temps d'exécution en fonction de la quantité de ressources réservées du premier serveur de calcul. Expliquez les résultats en faisant des liens avec vos choix d'implémentation.

Tests de performance – mode non-sécurisé

Instanciez la même configuration de trois serveurs de calcul et un répartiteur mais cette fois en mode non-sécurisé. Chaque travail sera demandé à trois serveurs différents de manière à détecter les résultats malicieux. Choisissez une valeur de quantité de ressources réservées raisonnable selon les résultats obtenus précédemment, ni trop petite (exécution trop longue), ni trop grande (jamais d'échec).

Exécutez les trois cas suivants tout en mesurant le temps écoulé :

- Premier serveur malicieux 50% du temps, deux autres serveurs de bonne foi.
- Deuxième serveur malicieux 50% du temps, deux autres serveurs de bonne foi.
- Troisième serveur malicieux 50% du temps, deux autres serveurs de bonne foi.

Présentez et expliquez les temps obtenus. Comparez avec le temps obtenu en mode sécurisé.

Question 1: Le système distribué tel que présenté dans cet énoncé devrait être résilient aux pannes des serveurs de calcul. Cependant, le répartiteur demeure un maillon faible. Présentez une architecture qui permette d'améliorer la résilience du répartiteur. Quels sont les avantages et les inconvénients de votre solution? Quels sont les scénarios qui causeraient quand même une panne du système?

Rapport

Puisque vous avez beaucoup de liberté pour ce travail pratique, chaque équipe devrait obtenir un résultat assez différent. Votre court rapport devrait donc mentionner et expliquer les différents choix que vous avez faits pendant la conception, en commençant par le langage et la technologie de communication utilisés, mais aussi la façon dont le répartiteur divise les tâches et gère les tâches échouées, comment le système détecte une panne intempestive, etc. Pourquoi avez-vous fait ces choix de préférence à d'autres ?

Annexe – Conseil pour le travail au laboratoire

Entre les ordinateurs du laboratoire, seuls les ports de 5000 à 5050 sont ouverts pour la communication entre postes. Vous devez donc vous assurer que les processus qui écoutent sur le réseau écoutent sur un port dans cet intervalle.

Dans le cas de RMI, vous devrez d'abord modifier le port d'écoute de *rmiregistry* en lui fournissant comme paramètre en ligne de commande, par exemple:

```
rmiregistry 5001
```

L'appel à getRegistry devra refléter ce changement de port. Puis, en exportant un objet avec exportObject, un port entre 5000 et 5050 devra être fourni afin que le client puisse se connecter. Si vous utilisez une autre technologie que RMI, vous devrez aussi probablement vous adapter à cette contrainte.

Pour faire vos tests avec plusieurs instances, utilisez SSH entre les postes. Lorsque vous êtes sur un poste du labo, vous n'avez qu'à taper ssh l4714-xx pour vous connecter à un autre poste (par exemple, l4714-01). De plus, vos sessions sur les différents postes partagent le même système de fichiers. Vous n'avez donc pas à transférer vos fichiers entre les postes, ils seront là dès que vous vous connecterez par SSH.

Vous aurez plusieurs variables de configuration à modifier afin de connecter les différents serveurs entre eux : adresses des serveurs, numéro de port, mode de fonctionnement, quantité de ressources, etc. Vous pouvez utiliser un ou plusieurs fichiers de configuration pour contenir ces variables afin de facilement les modifier pour réaliser vos différents tests.

Vous aurez vraisemblablement besoin d'un fichier texte de grande taille afin de stresser suffisamment votre système. Le projet Gutemberg offre gratuitement des livres en format texte. Pour créer des fichiers encore plus gros, vous pouvez concaténer un fichier avec lui-même suffisamment de fois pour obtenir un fichier de la taille voulue.

Project Gutenberg: http://www.gutenberg.org/