POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF8480

Systèmes répartis et infonuagique

Rapport de TP2

Présenté à

Houssem Daoud

Par

Anthony Abboud

Et

Luc Courbariaux

20 mars 2018

Introduction

Le but de ce deuxième laboratoire est de nous introduire à la gestion des pannes et des modes de sécurisation à l'intérieur d'un système réparti sur plusieurs machines. Dans notre cas, nous devions implémenter un répartiteur (le client) ainsi que des serveurs de calculs (les serveurs) servant à résourdre une longue liste d'opérations contenu dans un fichier texte.

Présentation du système

Notre système réparti est divisé en 3 parties: le répartiteur, les serveurs de calcul et le service de répertoire de noms.

- Le répartiteur

Le répartiteur est le point d'entrée du système. Il s'authentifie d'abord auprès du service de répertoire de noms afin de lui demander la liste de serveurs disponibles. Il se connecte ensuite à ces derniers (fonction loadServerStub()). Après, il ira lire les opérations à traiter dans le fichier texte (envoyé en paramètres) et le stockera dans un conteneur afin de facilitter les manipulations; ceci est la fonction loadData().

Ensuite, le répartiteur passe à la tâche de distribution (fonction distribution()). Notre fonction de distribution répartit les opérations en fonction de la capacité relative de chaque serveur. Par exemple, si on a 100 opérations, que le serveur 1 a une capacité q1 de 4 opérations, et que la capacité totale de tous les serveurs de calcul est de 40, alors 10% (4/10) des opérations seront envoyées à ce serveur.

Une fois envoyée sur différents fils d'exécution, le répartiteur attend les résultats. Si un serveur de calcul est tué lors de l'exécution ou qu'un serveur refuse de faire son calcul, la fonction recommence la distribution parmi les serveurs restants. Si un résultat est retourné, on passe à la fonction de vérification de sous-résultats.

La fonction de vérification des sous-résultats est très simple: elle vérifie si l'attribut 'accepted' de chaque sous-résultat de type Result envoyé par les serveurs sont vrais ou non. Cela est utile lorsque le système roule en mode non-sécurisé.

Enfin, si tous les sous-résultats sont bons, on passe à la dernière tâche, le rassemblement et affichage du résultat final. Cette fonction fait donc la somme de tous les sous-résultats, applique le dernier modulo 4000 afin d'éviter un débordement, et puis affichage le résultat à l'écran de l'usager.

- Les serveurs de calcul

Les serveurs de calcul reçoivent les tâches du répartiteur, chacun ayant un nombre d'opérations différentes à effectuer.

D’abord, le serveur effectuer l’acceptation du travail à l’aide de la formule de pourcentage de refuse mentionné si haut. Si jamais le travail est refusé, le serveur enverra au client une réponse 0 avec attribut ‘accepted’ à false.

Ensuite, le serveur effectue les opérations de pell et prime. En mode sécurisé, il effectue directement le calcul et envoit la réponse au client avec ‘accepted = true’. En mode non-sécurisé, il enverra probablement une variable faussée, que le répartiteur devra détecter dans sa fonction de vérification.

- Le service de répertoire de noms

Le service de répertoire de noms sera un LDAP simple : c’est un arbre de répertoires contenant des entrées, chaque entrée constituée d'attributs pouvant contenir plusieurs valeurs et où l’on peut observer des requêtes et réponses asynchrones. Ce service sera utiliser pour authentifier le client (répartiteur) pour lui donner la liste des adresses IP des serveurs de calcul.

Tests de performance – mode sécurisé

Nous n’avons malheureusement pas pu tester notre système pour plus d’un serveur.
Pour un serveur de capacité de 60, le fichier operations-588.txt prend un temps de 436 ms, et ce avec des print() du côté du serveur (qui ralentissent le temps d’exécution).

Tests de performance – mode non-sécurisé

Nous n’avons malheureusement pas pu tester notre système pour plus d’un serveur, et alors il semble que notre implémentation ne renvoit pas d’erreurs tel qu’elle le devrait quel que soit le taux choisi.

Réponse à la question 1

Théoriquement, la solution la plus simple serait de faire en sorte que chaque serveurs de calculs soient aussi des répartiteurs.

Les avantages de cette solution sont que chaque membre du système peut vérifier et redistribuer les tâches elle-mêmes, rendant l’exécution plus rapide et plus solide. Aussi, si jamais le répartiteur tombe en panne, les autres serveur de calcul peuvent prendre la relève facilement.

Par contre, on peut observer quelques inconvénients. D’abord, le système deviendrait décentralisé puisque plus personne ne possède une autorité. Cela fait ensuite en sorte que si jamais un des serveurs de calcul se fait pirater, le problème ne peut plus être isolé et l’ensemble du système serait affecté.

Il est à noter qu'un tel système pourrait quand même tomber en panne dans les cas où ils utilisent un LDAP, et celui-ci tombe en panne. Personne ne pourra s'authentifier et accéder à la liste des serveurs/répartiteurs disponibles. Un autre cas où le système peut tomber en panne est si tout le système se trouve sur le même réseau, et que ce dernier tombe en panne. Ainsi, pas de connexion, pas de communication possible.

Explications supplémentaires

Afin d'optimiser la quantité d'opérations à envoyer par travail à chaque serveur, nous devons nous fier à la formule du taux de refuse suivante :

$$T = \frac{u_i - q_i}{5 q_i} * 100\,\%$$

Puisqu'on veut que ce facteur soit le plus petit possible, on cherche donc à maximiser la formule suivante :

$$\left(1 - \frac{u_i - q_i}{5 q_i}\right) * u_i$$

Pour trouver le maximum, on doit calculer sa dérivée :

$$1 - \frac{2u_i - q_i}{5 q_i} = 0$$
$$2u_i - q_i = 5 q_i$$
$$2u_i = 6 q_i$$
$$u_i = 3 q_i$$

On enverra donc des messages 3 fois plus long que la capacité des serveurs, offrant un taux de refus maximal de 60%. Cela optimise l'exécution des opérations.