ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF4410: Systèmes répartis et infonuagique (Automne 2016)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Lundi le 31 octobre 2016

HEURE: 13h45 à 15h35

DUREE: 1H50

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Une banque achète et vend des devises (e.g. dollars US ou Euros) à des taux très concurrentiels. Lorsque les prix sont mis à jour, il est essentiel que ces mises à jour rejoignent toutes les succursales en même temps et que la mise à jour se fasse simultanément. Il ne faut jamais que des prix différents, en raison d'une mise à jour en cours, soient vus dans deux succursales différentes, puisqu'alors des spéculateurs pourraient prendre de l'argent à la banque simplement en profitant de cette différence de prix. Il est aussi important de s'assurer que les messages sont valides (non modifiés) et authentiques (viennent bien de la maison-mère de la banque). Décrivez comment un tel service peut être implanté (messages envoyés et reçus pour une mise à jour, contenu de ces messages en termes de champs ajoutés et de transformation). **(2 points)**

Un message de groupe atomique permet de s'assurer que la mise à jour soit faite partout. On ne peut pas avoir une simultanéité parfaite de la mise à jour, mais on peut s'assurer que les deux prix ne sont jamais visibles simultanément. Une solution simple mais un peu inefficace est de faire un premier message atomique pour arrêter toute transaction (retirer l'ancien prix). Si ce message a fonctionné, un second message atomique transmet le nouveau prix. On peut bâtir là-dessus pour s'assurer plus efficacement que deux succursales ne puissent jamais montrer simultanément des prix différents. Un premier message pourrait prévenir qu'une mise à jour s'en vient et valider que toutes les succursales peuvent désactiver le prix courant. Une fois reçu la confirmation de chaque succursale, le serveur peut confirmer que la mise à jour est en route et faire désactiver le prix courant partout. Lorsque chaque succursale a confirmé avoir retiré le prix courant, le serveur peut envoyer un message (pas nécessairement atomique) à chaque succursale avec le nouveau prix. La première ronde de message, pour s'assurer que chaque succursale est prête, n'est pas vraiment requise. Cependant, elle évite de procéder à un changement si une des succursales n'est pas en mesure de répondre, ce qui bloquerait les autres avec un prix désactivé. Pour valider l'authenticité des messages, on peut utiliser un système d'encodage (par exemple à clés publiques avec double encodage pour assurer l'authenticité de l'envoyeur et la confidentialité du message). L'ajout de champs (avant encryption) avec une date, un numéro de séquence et une somme de contrôle permet de s'assurer que le message n'a pas été retardé, rejoué ou modifié.

b) Un gros fichier doit être transmis d'un serveur vers un grand nombre de clients. Chaque client est rejoignable soit par un réseau sans-fil commun, qui supporte la multi-diffusion et offre un débit de 100Mbit/s, soit par un réseau filaire commuté, dont chaque prise supporte un débit de 1Gbit/s, mais qui ne supporte que les communications point à point. Pour évaluer le temps requis pour les transmissions, on néglige la latence d'envoi sur le réseau, les paquets perdus et les bits requis pour les en-têtes de paquets, et on ne tient compte que du débit. Les choix disponibles sont soit de faire un envoi du fichier à tous les clients simultanément par multi-diffusion sur le réseau sans-fil, soit de faire une chaîne de distribution en arborescence en utilisant le réseau filaire (chaque client qui a une copie du fichier en envoie une copie à un autre client qui ne le possède pas encore). Quel est le temps pour transmettre un fichier de taille k (en bits) à n clients dans chacun des deux cas? Pour quelles valeurs de n et de k est-ce que chaque solution (sans-fil versus filaire) est plus rapide? **(2 points)**

Avec le réseau sans-fil, une seule transmission rejoint tout le monde en $t_{sf} = k/100M$. Avec le réseau filaire, après $t = k/1G$, 1 client est rejoint, ensuite 3, 7, 15... Donc $t_f = log_2(n+1) \times$

$k/1G$. Le ratio entre les deux est donc de $t_f/t_{sf} = log_2(n+1) \times k/1G \times 100M/k = log_2(n+1)/10$. Ainsi, pour de petites valeurs de n, le réseau filaire est plus rapide. Les deux seront identiques pour $t_f/t_{sf} = 1 = log_2(n+1)/10$ ou $log_2(n+1) = 10$ soit $n = 1023$. Ainsi, la valeur de k ne fait pas de différence et le réseau filaire est plus efficace lorsque n est inférieur à 512, équivalent lorsque n est entre 512 et 1023, et moins efficace autrement.

c) Quels sont les avantages et inconvénients de CORBA? En quoi pourrait-il être plus intéressant que SOAP? **(1 point)**

CORBA est un système complexe et donc un peu difficile à mettre en oeuvre. Par contre, il offre beaucoup de flexibilité, fonctionne avec de nombreux langages de programmation, et offre une excellente performance. SOAP est plus simple et offre aussi la possibilité de s'interfacer à plusieurs langages. Par contre, en raison de son format texte, les messages sont plus gros et il est moins efficace.

Question 2 (5 points)

a) Un service similaire à celui que vous avez programmé dans votre premier TP offre la lecture de fichiers à distance par Java RMI. Voici l'interface pour ce service qui retourne le contenu du fichier dont le nom est reçu en argument. Fournissez le contenu complet du fichier Server.java qui permettrait d'implémenter ce service en java, incluant la classe `Server`, les déclarations et les initialisations, mais sans les inclusions (`import`). **(2 points)**

```
package ca.polymtl.inf4410.tp1.shared;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServerInterface extends Remote {
byte[] ReadFile(String name) throws RemoteException;
}


public class Server implements ServerInterface {
  public static void main(String[] args) {
    Server server = new Server();
    server.run();
  }

  public Server() { super(); }

  private void run() {
    if (System.getSecurityManager() == null) {
      System.setSecurityManager(new SecurityManager()); }
    try {
      ServerInterface stub = (ServerInterface)
```

```
        UnicastRemoteObject.exportObject(this, 0);
      Registry registry = LocateRegistry.getRegistry();
      registry.rebind("server", stub);
    } catch (ConnectException e) {
      System.err.println("Impossible de se connecter au registre RMI");
    } catch (Exception e) { System.err.println(e.getMessage()); }
  }

  @Override
  public byte[] ReadFile(String name) throws RemoteException {
    return Files.ReadAllBytes(Paths.get(name));
}
```

b) Dans le premier TP, un client, un serveur et un processus `registry` s'exécutaient et utilisaient Java RMI. Dans le serveur, un objet de type `Server` était initialisé puis enregistré auprès du `registry`. Java fonctionne avec un ramasse-miettes (garbage collector) et libère la mémoire associée aux objets qui ne sont plus utilisés. A quel moment est-ce que l'objet de type `Server` du serveur ne sera plus utilisé et le système de support à l'exécution saura qu'il peut récupérer la mémoire associée? Toujours dans le client et le serveur fournis dans le premier TP, on retrouve les deux appels distants suivants, `UnicastRemoteObject.exportObject(this, 0);` pour exporter l'objet de type `Server` vers le registry et `distantServerStub.execute(a, b);` lorsque le client demande au serveur d'effectuer un calcul. Expliquez quelle information est transmise pour chaque appel entre les deux processus en cause, et en particulier pour chaque argument (this, a, b) dites si une copie distincte est effectuée ou si un autre mécanisme est employé. **(2 points)**

Lors de l'appel pour enregistrer l'objet de type `Server` *dans le* `registry`*, le système RMI du serveur mémorise que cet objet est en utilisation car exporté. Lorsque le* `registry` *cessera d'utiliser l'objet de type* `Server` *et qu'il avertira le RMI du serveur que cet objet n'est plus utilisé, s'il n'y a pas d'autres utilisateurs pour l'objet, il pourrait être mis aux rebuts.*

Lors d'un appel de méthode à distance, on envoie une référence à l'objet ciblé, un numéro identifiant la méthode voulue et une référence aux objets en arguments s'ils implémentent l'interface Remote (comme this) ou une copie distincte s'ils implémentent l'interface serializable (a et b).

c) Dans les fichiers XDR pour les SUN RPC, il est possible de spécifier un numéro de version pour un service. Quel usage peut-on en faire? **(1 point)**

Il est possible de fournir une implémentation séparée pour chaque version et ainsi d'avoir différentes versions du même service (avec leur implémentation spécifique) disponibles.

Question 3 (5 points)

a) Lors de la migration d'une machine virtuelle (VM) d'un serveur physique à l'autre, il faut copier toutes les pages qui constituent son image en mémoire virtuelle d'un serveur à l'autre. On

suppose que les même fichiers sont accessibles des deux serveurs et un peu d'état du logiciel de virtualisation (taille négligeable) doit aussi être copié. On vous propose deux techniques possibles pour effectuer la migration tout en minimisant le temps pendant lequel la machine virtuelle est complètement arrêtée. La première méthode consiste en tout copier sans arrêter la VM mais en notant les pages qui ont été modifiées après que cette itération de copie ait commencé. On continue ainsi pour quelques itérations jusqu'à ce que le nombre de pages encore modifiées soit petit. A ce moment, on arrête la VM, copie les pages restantes et redémarre la VM sur le nouveau serveur. La seconde méthode consiste en commencer dès le départ en même temps la copie des pages et l'exécution sur le nouveau serveur. Cependant, lorsque l'exécution de la VM requiert une page qui n'est pas encore copiée, la VM bloque, priorise la copie de la page après laquelle elle attend, et reprend dès que cette page arrive. Avec cette seconde méthode, il n'y a pas d'arrêt complet de la VM mais plusieurs pauses en attente de pages accédées qui n'ont pas encore été copiées. L'image en mémoire virtuelle contient 2 000 000 pages. Les pages sont copiées d'un serveur à l'autre au rythme de 20 000 pages par seconde. Pour la première méthode, la VM en exécution modifie 2 000 pages par seconde et initie l'arrêt et le transfert des pages restantes après trois intérations (la première copie et deux itérations pour les pages modifiées depuis lors). Pour la seconde méthode, la VM accède 4 000 pages par seconde non encore copiées. Pour la première méthode, quel est le temps pendant lequel la VM est arrêtée? Quel est le temps total de migration? Pour la seconde méthode, quel est le nombre de fois et la durée des arrêts en attente d'une page accédée non déjà copiée? Quel est le temps total de migration? **(2 points)**

L'ensemble des pages peut être copié en 2 000 000 / 20 000 = 100s. Pendant ce temps, $100 \times 2000 = 200000$ *pages ont été modifiées, ce qui demande 200 000 / 20 000 = 10s pour la seconde itération. Pendant ce temps,* $10 \times 2000 = 20000$ *pages ont été modifiées, ce qui demande 1s pour la troisième itération. Pendant ce temps, 2000 pages ont été modifiées. La VM est alors arrêtée et il faut 2000 / 20 000 = 0.1s pour finaliser le transfert. La VM est donc arrêtée pendant 0.1s. La migration totale a pris 100s + 10s + 1s + 0.1s = 111.1s. Avec la seconde méthode, chaque page n'est copiée qu'une seule fois, ce qui demande 100s et constitue le temps total de migration. Il n'y a pas d'arrêt complet. Cependant, pendant le transfert, il arrive à* $100 \times 4000 = 400000$ *reprises que la page désirée ne soit pas disponible et qu'il faille attendre le temps du transfert d'une page pour qu'elle parvienne en priorité sur le nouveau serveur. Ce temps est d'environ 1 / 20 000 = 0.00005 si la page est transférée immédiatement. S'il faut attendre après le transfert en cours d'une page, on pourrait ajouter la moitié de cette valeur (entre 0 et 1, soit une moyenne de 0.5 page à attendre pour le transfert déjà en cours, avant de pouvoir transférer la page spécifique après laquelle on attend). Ces 400 000 attentes de 0.00005s totalisent 20s mais n'ont pas un gros impact car ces faibles latences ajoutées sont plus difficilement perceptibles par les clients de la VM que l'arrêt complet pendant 0.1s.*

b) Lors de vos travaux pratiques, vous utilisez un système infonuagique basé sur OpenStack. Au moment de créer une nouvelle instance de machine virtuelle, OpenStack doit choisir sur quel noeud physique la placer. Quelle composante de OpenStack est responsable de choisir le noeud physique et démarrer l'instance virtuelle? Comment se fait le choix du noeud physique? **(1 point)**

C'est le module Nova qui s'occupe d'exécuter les instances de machines virtuelles. Le choix se

fait premièrement en déterminant les noeuds physiques qui remplissent les requis (architecture, zone géographique, quantité de mémoire, nombre de coeurs...) et ensuite en sélectionnant celui qui a le meilleur pointage parmi ceux-ci selon divers critères (charge, nombre d'instances déjà présentes, puissance du noeud...).

c) La fonctionalité KSM (Kernel Same page Merging) a été ajoutée au noyau Linux pour aider la virtualisation. Que fait KSM? Pourquoi est-ce particulièrement intéressant pour la virtualisation? **(1 point)**

KSM vérifie les pages (en mode lecture seulement) dont le contenu est identique en mémoire virtuelle et les consolide en n'en conservant qu'une seule copie. Ainsi, si plusieurs machines virtuelles utilisent les même fichiers, par exemple les exécutables du noyau Linux, de librairies ou d'applications populaires, il est possible de n'avoir qu'une seule copie, partagée, en mémoire physique, comme c'est le cas lorsque tout s'exécute sur un seul ordinateur, sans virtualisation. Ceci permet donc d'atténuer un des surcoûts associés à l'utilisation de machines virtuelles, à savoir la plus grande utilisation de mémoire lorsque des copies redondantes de certains fichiers populaires sont conservées en mémoire par chaque machine virtuelle.

d) Sur Amazon EC2, trois types de services de stockage (disque) sont disponibles, quels sont-ils? Quels sont les composantes de OpenStack qui offrent les services équivalents? **(1 point)**

Sur Amazon EC2, on retrouve le stockage d'instance, qui disparait lorsque l'instance est arrêtée, le stockage de blocs (Elastic Block Store, EBS), qui demeure lorsque l'instance est arrêtée mais qui ne peut être associé qu'à une seule instance à la fois, et le stockage d'objets (Simple Storage Service S3) qui sont accessibles de plusieurs instances à la fois. Sur OpenStack, Cinder offre le stockage d'instance et de blocs, et Swift offre le stockage d'objets.

Question 4 (5 points)

a) Un serveur de disque reçoit des requêtes à partir de clients. Chaque client requiert en moyenne des données pour 10 megabits/s. Le réseau est entièrement commuté. Le serveur est connecté au réseau par une prise qui fournit 10 gigabits/s. Son bus a une capacité de 16 gigaoctets/s et 6 disques y sont connectés. Les disques actuels fournissent chacun 100 megaoctets/s. Combien de clients est-ce que ce serveur peut supporter maintenant? Combien de clients peut-il supporter si on remplace les disques actuels par des disques SSD qui fournissent 1 gigaoctets/s chacun? **(2 points)**

Chaque client requiert 10 megabits ou 10/8 = 1.25 megaoctets/s. Le serveur peut recevoir 10 gigabits ou 1.25 gigaoctets/s. Le bus a une capacité de 16 gigaoctets/s et les 6 disques $6 \times 100 = 600$ *megaoctets/s. Le facteur limitant est donc les disques qui peuvent soutenir 600 megabits/s / 1.25 megabits/s / client = 480 clients. Avec des disques SSD, on passe à 6 gigaoctets/s et le facteur limitant devient le réseau à 1.25 gigaoctets/s, ce qui donne 1.25 gigaoctets/s / 1.25 megaoctets/s / client = 1000 clients.*

b) Sur un client NFS, le système d'exploitation reçoit 2 lectures et 1 écriture de page de 4Kio par seconde en moyenne. Pour une lecture, la page cherchée ne se trouve pas sur le client dans 25% des cas, se trouve déjà sur le client et a été validée depuis moins de 3 secondes dans 40% des

cas, et se trouve sur le client mais a été validée depuis plus de 3s dans le reste des cas, soit 35%. Lorsqu'une page est présente mais sa validation est trop vieille, une revalidation suffit dans 60% des cas et une relecture doit être faite en plus (car la validation indique que la page a changé) dans 40% des cas. Sur le serveur, une demande de validation demande 2ms de CPU, une lecture demande 4ms de CPU et dans 30% des cas un accès disque de 12ms, et une écriture demande 5ms de CPU et 12ms de disque. Si 25 clients accèdent ce serveur qui comporte 1 CPU et 1 disque, quel sera le pourcentage d'utilisation du CPU? Du disque? **(2 points)**

Un client génère 1 écriture/s sur le serveur. Il génère aussi 2 lectures/s $\times$ (0.25 + 0.35 $\times$ 0.4) = 0.78 lectures/s. Finalement, un client génère 2 lecture/s $\times$ 0.35 = 0.70 validations/s. Ceci génère sur le serveur à chaque seconde pour chaque client: 5ms CPU et 12ms disque (1 écriture), 0.7 $\times$ 2ms = 1.4ms CPU (0.7 validation), et 0.78 $\times$ 4ms = 3.12ms CPU et 0.78 $\times$ 0.3 $\times$ 12ms = 2.808ms disque (0.78 lecture). Le total est donc de 9.52ms CPU et 14.808ms disque. Avec 25 clients, le taux d'utilisation sera de 25 $\times$ 9.52ms / 1000ms/s = 23.8% CPU et 25 $\times$ 14.808ms / 1000ms/s = 37.02% disque.

c) Avec le système de fichiers CEPH, la répartition des groupes de placement (PGID) sur les serveurs de fichiers est déterminée par un algorithme de hachage (CRUSH). Quel est l'intérêt d'un tel algorithme pour déterminer le placement sur un service de fichiers réparti? **(1 point)**

Cet algorithme permet de calculer directement, avec un calcul simple effectué sur le client, le serveur de fichiers sur lequel se trouve le PGID. Il n'est ainsi pas nécessaire de consulter un service quelconque afin d'obtenir cette information. Ceci enlève un goûlot d'étranglement potentiel (serveur de métadonnées pour le placement des PGID) et permet à CEPH de plus facilement se mettre à l'échelle avec un très grand nombre de clients et de serveurs.

Le professeur: Michel Dagenais