ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Automne 2018)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Lundi le 22 octobre 2018

HEURE: 13h45 à 15h35

DUREE: 1H50

NOTE: Aucune documentation permise sauf un aide-mémoire, préparé par l'étudiant, qui consiste en une feuille de format lettre manuscrite recto verso, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Un service de message de groupe relie n ordinateurs. Quel est le nombre de messages envoyés (précisez pour chaque message s'il est en multi-diffusion ou à un seul destinataire) pour réaliser un message de groupe i) non fiable, ii) fiable, iii) atomique, iv) totalement ordonnancé. **(2 points)**

Pour un message de groupe non fiable, un seul message en multi-diffusion suffit. Pour un message de groupe fiable, un message en multi-diffusion rejoint chacun, et ensuite chacun des $n-1$ autres membres du groupe répond avec un message régulier d'accusé de réception. Pour un message atomique, on a un premier message en multi-diffusion pour envoyer l'information, ensuite $n-1$ messages des autres membres du groupe pour accepter l'envoi, et finalement un message de confirmation en multi-diffusion. Pour un message totalement ordonnancé, il faut un message au serveur d'ordonnancement et un message de réponse. Ensuite on envoie un message, étiqueté avec le numéro d'ordre obtenu, en multi-diffusion. Il n'est pas nécessaire d'avoir des accusés de réception de ces messages numérotés. En effet, si un client voit qu'il lui manque un message dans la séquence numérotée, il peut facilement déduire que le message correspondant lui manque.

b) Un client effectue une requête auprès d'un serveur. Le client prépare sa requête en 25ms, la requête transite sur le réseau en 1ms, elle est traitée en 8ms par le serveur qui ne fait que des calculs sur le CPU pendant ce temps et n'utilise aucun périphérique, la réponse transite sur le réseau en 1ms et elle est traitée sur le client en 15ms. Le client est alors prêt pour envoyer sa prochaine requête. Le serveur sert les requêtes avec un seul thread. Combien de requêtes par seconde est-ce qu'un client seul peut faire? Combien de clients ce serveur pourrait-il supporter avant de devenir saturé? **(2 points)**

Avec un seul client, le temps total est de 25ms+1ms+8ms+1ms+15ms = 50ms. Le client peut donc faire 1000ms/s / 50ms/requête = 20 requêtes / s. Le serveur n'est en fait occupé que pendant 8ms. Il peut donc servir 1000ms/s / 8ms/requête = 125 requêtes / s soit 125r/s / 20r/s = 6.25 donc 6 clients avant de devenir saturé.

c) Vous êtes en train de concevoir un service de multi-diffusion fiable. Vous avez le choix entre des accusés de réception positifs (accuser réception de chaque paquet reçu) ou négatifs (envoyer un message pour signifier chaque paquet manquant). Le meilleur choix dépend de la probabilité p qu'un client ne reçoive pas un paquet. Pour quelles valeurs de p est-ce que chacune des deux options sera préférable? **(1 point)**

Puisqu'on veut minimiser le nombre total de paquets envoyés, on prendra des accusés négatifs, si les paquets perdus sont peu probables ($0 \leq p \leq 0.5$), et des accusés positifs, si les paquets sont plus souvent perdus que non ($0.5 \leq p \leq 1$). On suppose ici que les accusés de réception étant très petits, il sont rarement perdus eux-mêmes.

Question 2 (5 points)

a) Un service d'interrogation de la base de donnée des employés utilise gRPC avec les définitions données plus bas. Un employé est embauché et ses coordonnées sont: id = 12, nom = "Haddock", prenom = "Archibald", adresse = "Chateau de Moulinsart", numero = 7654321. Quelle est la longueur du message de type EmployeFiche correspondant, étant donné l'encodage utilisé par protobuf dans gRPC? L'entreprise grossit et commence à embaucher des employés à l'international. Il faut donc ajouter le pays pour ceux dont l'adresse n'est pas au Canada. Comment peut-on ajouter ceci aux définitions actuelles? Est-ce que cela va demander de mettre à jour tous les logiciels client et serveur qui utilisaient ces définitions? **(2 points)**

```
service EmployeInfo {
  rpc EmployeEmbauche (EmployeFiche) returns (EmployeId);
  rpc EmployeNumero (EmployeId) returns (NumeroTelephone);
}
message EmployeId {
  int32 id = 1;
}
message NumeroTelephone {
  int32 numero = 1;
}
message EmployeFiche {
  int32 id = 1;
  string nom = 2;
  string prenom = 3;
  string adresse = 4;
  int32 numero = 5;
}
```

Pour tous les champs, le numéro de champ et le type peuvent entrer dans un seul octet. En effet, le type prend 3 bits et 4 bits restent disponibles dans le varint de 1 octet pour le numéro de champ; le numéro de champ le plus grand est de 5, ce qui prend 3 bits alors que 4 sont disponibles. Pour id, le nombre 12 requiert 1 octet, pour numero le nombre 7654321 requiert 23 bits, à 7 bits disponibles par octet dans un varint, cela veut dire 4 octets. Pour nom, la longueur prend 1 octet et les caractères 7 octets. Pour prenom, la longueur prend 1 octet et les caractères 9 octets. Pour adresse, la longueur prend 1 octet et les caractères 21 octets. Le total est donc de (1+1)+(1+1+7)+(1+1+9)+(1+1+21)+(1+4) = 50. Il suffirait d'ajouter un champ optionel, `string pays = 6 [default = "Canada"]` *à EmployeFiche. Ainsi, une nouvelle application qui cherche le pays trouvera Canada si rien n'est spécifié, même si l'autre partie n'a pas été mise à jour. Concrètement, les serveurs et les clients qui peuvent avoir à traiter des cas d'employés n'ayant pas le Canada comme pays doivent être mis à jour. Les autres n'ont pas à le faire, étant donné que les champs sont auto-identifiés avec gRPC et que la valeur par défaut de Canada est ajoutée.*

b) Une application de serveur de dessin est créée. Deux processus, P1 et P2, interagissent via Java RMI en utilisant les interfaces *Shape* et *ShapeList* définies ci-après. La méthode *allShapes* retourne un vecteur de *Shape*. Le processus P1 crée 4 formes, S1, S2, S3 et S4, du type *ShapeServant* qui est défini comme suit *ShapeServant extends UnicastRemoteObject implements Shape*. Ensuite P1 crée un dessin, SL1, du type *ShapeListServant* qui est défini comme suit *ShapeListServant extends UnicastRemoteObject implements ShapeList*. P1 ajoute S1, S2, S3 et S4 dans le dessin SL1 avec la méthode *addShape*. Avec le service *Naming*, P2 obtient une référence réseau (un proxy) à l'objet SL1, dénotée rrSL1. P2 crée alors un dessin, SL2, du type *ShapeListServant*. Il obtient avec la méthode *allShapes* de rrSL1 un vecteur et ajoute les 4 *Shape* du vecteur à l'objet SL2 avec la méthode `addShape`. Avec le service *Naming*, P1 obtient une référence réseau à SL2 dénotée rrSL2. Il obtient avec la méthode *allShapes* de rrSL2 un vecteur. Suite à ces opérations, dites quels objets réels et quelles références réseau sont contenus dans P1 et P2 pour les formes et les dessins (par exemple S1 pour l'objet réel et rrS1 pour une référence réseau à S1). **(2 points)**

```
public interface Shape extends Remote {
  GraphicalObject getAllState() throws RemoteException;
}

public interface ShapeList extends Remote {
  void addShape(Shape s) throws RemoteException;
  Vector allShapes() throws RemoteException;
}
```

Le processus P2 contient SL2, rrSL1, rrS1, rrS2, rrS3, rrS4. Le processus P1 contient S1, S2, S3, S4, SL1, rrSL2. Lorsque des références réseau à S1, S2, S3 et S4 sont reçues sur P1 par RMI suite à l'appel de méthode `allShapes` *de rrSL2 , elles sont automatiquement reconverties en des pointeurs aux objets réels, puisque ceux-ci sont des objets exportés par P1.*

c) Dans le cadre du premier travail pratique, la consigne suivante était spécifiée: "comme le serveur sera en mesure d'accepter des appels de plusieurs clients de façon simultanée, vous devez prendre les précautions nécessaires pour assurer la cohérence des données dans les structures partagées au sein du serveur". Expliquez comment il fallait implémenter ces précautions dans votre programme. **(1 point)**

Puisque plusieurs requêtes concurrentes de différents clients peuvent être traitées en parallèle sur différents threads, les méthodes qui accèdent les structures de données de l'objet doivent être protégées, par exemple avec l'attribut synchronized qui couple une prise de verrou à l'appel de la méthode. Autrement, si deux appels sont reçus en même temps par RMI de deux clients pour prendre le verrou sur un fichier, Les deux pourraient accéder en même temps la structure de donnée qui dit si le fichier est réservé pour un client, conclure que le fichier est libre, et le donner en même temps aux deux clients.

Question 3 (5 points)

a) Une machine virtuelle exécute une tâche urgente qui nécessite 1000 secondes de CPU sur un coeur (disponible à 100%) et le transfert par réseau de 200GiO de données (.2GiO par seconde de CPU à 100%); le transfert peut s'effectuer en parallèle avec les calculs sur le CPU. Cette machine virtuelle dispose sur le noeud physique actuel de 50% d'un coeur et d'une bande passante pour le réseau de 0.5GiO/s. Elle pourrait migrer vers un noeud physique moins occupé où elle bénéficierait d'un coeur à 100% et de 1.0GiO/s de bande passante de réseau. Toutefois, il faut qu'il reste de la bande passante disponible pour que la migration s'effectue (la migration est en plus basse priorité que l'aplication). De plus, la machine virtuelle subirait un délai associé à la migration, pendant l'arrêt total afin de finaliser la migration. L'image de la machine virtuelle à migrer occupe 8GiO. La migration prend un volume de bande passante correspondant mais ne consomme pratiquement aucun CPU. La machine virtuelle, pendant son exécution, modifie 0.2GiO de son image par seconde de CPU à 100%. La copie de l'image à migrer se fait en plusieurs itérations, jusqu'à ce que les modifications à transférer constituent moins de 0.5GiO; à ce moment, la machine est arrêtée et les dernières modifications sont transférées. Combien de temps prends la migration? Est-ce que la tâche urgente sera terminée plus tôt avec la migration? En combien de temps? **(2 points)**

Sur le noeud actuel, la tâche urgente prendra 1000s / 0.5 = 2000s de CPU. En parallèle, les entrées-sorties sur le réseau nécessitent 200GiO / 0.5GiO/s = 400s. Le CPU est le facteur limitant et cela requiert 2000s. Sur le nouvel ordinateur, le temps requis serait de 1000s, puisque le coeur est disponible à 100%. La migration demande de transférer 8GiO. L'application prend 0.1GiO/s à 50% de CPU. Il reste donc 0.4GiO/s pour transférer les 8GiO, ce qui demande 20s. Pendant 20s, 20s x .1GiO/s (à 50% de CPU) = 2GiO seront modifiés. Ceci prendra 2GiO/0.4GiO/s = 5s pour le transfert en seconde itération. L'itération suivante demandera 1.25s pour 0.5GiO. Il y aura ensuite 0.125GiO modifiés et la machine virtuelle sera arrêtée pendant 0.3125s afin de transférer ce restant, avant de démarrer sur le nouveau noeud. Le temps total pris par la migration est de 20s + 5s + 1.25s + .3125s = 26.5625s. Pendant 26.25s, la machine virtuelle s'est exécutée à 50% de CPU, il lui reste donc 1000s - 26.25s/2 = 986.875s à exécuter sur le nouveau noeud, pour un temps total de 26.5625s + 986.875s = 1013.4375s. La tâche urgente sera effectivement terminée bien plus vite avec la migration.

b) Trois machines virtuelles, A, B et C, s'exécutent sur un même noeud physique. Le noeud physique contient 4 coeurs et 4 disques. Chaque disque supporte 100 opérations d'entrée/sortie (IOP) par seconde. Chaque machine virtuelle sert des requêtes et répartit sa charge entre 4 coeurs virtuels et 4 disques virtuels. Les requêtes à la machine A prennent 50ms et 4 IOP, celle à la machine B 10ms et 8 IOP et celles à la machine C 100ms et 2 IOP. L'opérateur de la machine A a payé pour avoir une priorité absolue (même performance que si seul sur le noeud physique), celui de la machine B a payé pour une certaine priorité, et celui de la machine C a payé le minimum (la machine ne roule que si A et B ne font rien). Si A et B ne reçoivent aucune requête, combien de requêtes par seconde C peut-elle soutenir? Si A reçoit 40 requêtes par seconde et B 30 requêtes par seconde, combien de requêtes par seconde C peut-elle soutenir? **(2 points)**

Lorsque C est seule, elle peut soutenir (4 CPU x 1000ms/s) / 100ms/r = 40r/s au niveau du CPU et (4 disques x 100IOP/s) / 2 IOP/r = 200r/s pour les entrées-sorties. Elle peut donc effectuer

au maximum 40r/s. Lorsque A et B sont actives, il restera 4 CPU x 1000ms/s - 40r/s x 50ms/r - 30r/s x 10ms/r = 1700ms/s ainsi que 4 disques x 100IOP/s - 40r/s x 4 IOP/r - 30r/s x 8 IOP = 0 IOP/s. Tout les disques sont occupés à 100% et C ne pourra pas servir une seule requête!

c) OpenStack (avec virtualisation par KVM) et Kubernetes sont deux technologies très utilisées pour déployer des applications parallèles réparties. Peut-on rouler efficacement OpenStack au-dessus de Kubernetes? Kubernetes au-dessus de OpenStack? Kubernetes sur Kubernetes? OpenStack sur OpenStack? **(1 point)**

La virtualisation fonctionne assez bien à 1 niveau, en raison du support matériel dans les processeurs Intel et AMD. A deux niveaux, i.e. OpenStack sur OpenStack, le support matériel n'aide pas pour le second niveau et le surcoût est plus important, ce qui rend cela nettement moins efficace. Kubernetes est basé sur les conteneurs, ce qui n'implique pas de surcoût important, même lorsque 2 niveaux sont imbriqués. Toutes les solutions avec un seul (ou aucun) niveau de OpenStack sont donc efficaces.

Question 4 (5 points)

a) Un service de fichiers CODA est répliqué sur 3 serveurs (i.e. chaque fichier se retrouve en 3 copies). Chaque serveur possède 4 disques. Chaque disque peut effectuer 100 accès (lecture ou écriture) par seconde. Les clients, lors des ouvertures ou fermetures de fichiers, font des accès en lecture ou en écriture au serveur. Quel est le nombre maximal de lectures (s'il n'y a que des lectures) par seconde que pourrait soutenir ce service répliqué sur 3 serveurs, en supposant que la charge est répartie uniformément sur les serveurs et les disques, et que les disques constituent le facteur limitant? Quel est le nombre maximal d'écritures (s'il n'y a que des écritures)? Si on change pour un système avec 3 serveurs mais sans réplication, avec la charge uniformément répartie entre les 3, que devient le nombre maximal de lectures? Le nombre maximal d'écritures? **(2 points)**

En lecture, les serveurs opèrent en parallèle. Avec 3 serveurs de 4 disques à 100 accès/s, on a une capacité maximale de $3 \times 4 \times 100 = 1200$ *lectures par seconde. En écriture, chaque écriture doit être faite sur les 3 serveurs. On peut donc soutenir 3 fois moins d'écritures que de lectures soit 400 écritures par seconde. Si on enlève la réplication, la lecture reste la même, à 1200/s et les écritures deviennent semblables à 1200/s aussi.*

b) Une expérience a lieu au laboratoire du CERN. Pendant l'expérience, de nombreux noeuds reçoivent des données venant de nombreux capteurs et stockent cette information sur un système de fichiers réparti avec réplication en 3 copies. Tous les fichiers sont ouverts avant le début de la capture des données et chaque noeud sait donc déjà sur quels serveurs de fichiers il doit envoyer ses 3 copies. L'information générée sur chaque noeud est de 1GiO/s, et doit être stockée de manière répliquée sur les serveurs. Le service de fichiers est réparti sur 20 serveurs. Chaque serveur a une connexion au réseau de 20GiO/s et contient 4 contrôleurs de disque. Chaque contrôleur de disque est alimenté par un canal PCIe de 5GiO/s et est connecté à 8 disques. Chaque disque est capable de soutenir des écritures à un rythme de 2GiO/s. On suppose que la charge est parfaitement répartie entre les serveurs, les contrôleurs et les disques. Combien de noeuds est-ce que cette infrastructure est capable de supporter? **(2 points)**

Chaque noeud doit envoyer 3 x 1GiO/s = 3GiO/s. Sur chaque serveur, le réseau a une bande passante de 20GiO/s, et les contrôleurs de disques ont une bande passante de 4 x 5GiO/s = 20GiO/s. Pour chaque contrôleur, les disques une bande passante de 8 x 2GiO/s = 16GiO/s. Le facteur limitant est donc le contrôleur à 5GiO/s. Globalement, ce sont donc le réseau et les 4 contrôleurs qui l'un comme l'autre limitent la bande passante d'un serveur à 20GiO/s. Avec 20 serveurs, on peut soutenir 20 x 20GiO/s = 400GiO/s soit 400GiO/s / 3GiO/s = 133 clients.

c) Pour chacun de ces systèmes de fichiers répartis, NFS, AFS, CODA et CEPH, dites i) s'ils permettent la réplication en écriture, ii) s'ils permettent la mise à l'échelle à un vraiment très grand nombre de clients. **(1 point)**

CODA et CEPH permettent la réplication en écriture alors que NFS et AFS ne le permettent pas. Seul CEPH permet la grande mise à l'échelle en calculant la position du fichier de manière autonome avec un code de hachage, plutôt que d'avoir à consulter un serveur central pour cette méta-information.

Le professeur: Michel Dagenais