ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF4410: Systèmes répartis et infonuagique (Automne 2013)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Mardi le 29 octobre 2013

HEURE: 9h30 à 11h20

DUREE: 1H50

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

Un service de forum de discussion réparti permet à des usagers de se connecter à un serveur géographiquement proche et de participer à une discussion en y lisant ou ajoutant des messages. Les serveurs collaborant pour offrir ce service forment un groupe de serveurs qui se diffusent les messages. Il est possible d'ajouter un message sur un nouveau sujet de discussion, de répondre à un message dans une discussion existante, ou de lire les messages disponibles en listant les messages sur de nouveaux sujets ou en listant les réponses à un autre message.

a) Chaque message doit avoir un identifiant unique utilisé afin de demander un message, de lister les réponses à un message, ou pour répondre à un message. De plus, il ne faut pas afficher une réponse à un message avant d'afficher le message auquel il répond. Donnez i) le nom utilisé pour un tel ordonnancement de message de groupe, et proposez une méthode efficace ii) pour créer un identifiant unique pour chaque message, et iii) pour assurer le bon ordonnancement du traitement des messages dans chaque serveur. **(2 points)**

Il s'agit d'un ordonnancement causal. Il est facile d'assigner un identifiant unique en juxtaposant un identificateur unique de noeud (adresse IP ou adresse MAC de la carte réseau) avec un compteur de messages. Ainsi, lorsqu'un message d'un serveur est reçu, on peut

attendre pour les messages de compteur moins élévé pour le même serveur. Toutefois, il faut surtout attendre avant d'afficher un message reçu que le message auquel il répond soit reçu lui aussi.

b) Les messages du forum peuvent être diffusés entre les serveurs par TCP, ou par UDP en multi-diffusion. Chaque envoi de paquet par UDP ou TCP occupe le réseau, qui est ici une ressource partagée, pendant 100 microsecondes, plus le temps de transmission à 100Mbits/s. Un message moyen demande un paquet de 1500 octets tout compris. Par TCP, en plus du paquet du message d'envoi, le récipiendaire envoie un paquet d'accusé de réception de 100 octets. Combien de messages du forum par seconde le réseau peut-il supporter avec UDP et avec TCP s'il y a 4 serveurs? Proposez une organisation efficace pour détecter les messages perdus et les retransmettre en UDP avec multi-diffusion, puisque cette fonctionalité est offerte par TCP mais pas pour UDP. **(2 points)**

*Un paquet demande 100us / 1000000us/s + n octets * 8 bits / octet / 100Mbits/s. Ceci donne donc .00022s pour 1500 octets et .000108s pour 100 octets. Avec UDP, chaque message demande .00022s, pour un taux maximal de 1s/.00022s = 4545.45 messages / seconde. Avec TCP, chaque serveur doit envoyer le messages aux trois autres séparément et recevoir un accusé de réception, ce qui prend 3 * (.00022s + .000108s) = .000984s, pour un taux maximal de 1016.26 messages / seconde. Un serveur saura si un message est manquant s'il reçoit un message de valeur supérieure de compteur, l'accusé de réception n'est donc pas essentiel. Il demeure un problème de latence. Si un serveur A envoie le message numéro 85 et qu'il soit perdu avant d'arriver à B, et qu'ensuite le prochain message, 86, ne soit pas créé avant une heure, un long délai se passera avant que le serveur B ne réalise que le message 85 lui manque. Pour pallier à ce problème, un serveur qui ne reçoit rien d'un autre pour plus que 5 minutes pourrait le contacter afin de vérifier le numéro du dernier message envoyé.*

c) Dans un sujet de discussion du forum, portant sur une question politique délicate, un message incendiaire a été ajouté par un inconnu se faisant passer pour une autre personne, un candidat à une prochaine élection, afin de le mettre dans l'embarras. Expliquez brièvement ce qui est requis pour qu'un tel système de discussion soit sécuritaire à cet égard. **(1 point)**

Il faut que le message et son contenu soit proprement authentifié. Ceci peut se faire avec une signature (somme de contrôle cryptographique pour le contenu du message) ou l'encryption avec la clé privée de l'envoyeur (tous pouvant décrypter avec la clé publique associée de cet envoyeur). L'authentification peut se faire pour la connexion au serveur, pour le contenu de chaque message envoyé au serveur, ou même mieux au niveau du lecteur qui consulte le message.

Question 2 (5 points)

a) Lors du premier travail pratique, vous avez étudié le temps requis pour effectuer un appel à distance avec Java RMI en fonction de la taille des arguments de l'appel. Décrivez comment varie la durée de l'appel en fonction de la taille des arguments. Comment peut-on expliquer cette variation en fonction de la taille des arguments? **(2 points)**

Le temps de traitement et d'envoi possède une composante constante et une composante qui croit avec la taille. On peut donc approximer la fonction entre la taille et le temps par une droite. Toutefois, certains effets non-linéaires peuvent se produire localement en raison de tailles fixes pour des tampons ou pour les paquets sur le réseau. Un octet de plus dans le même paquet requiert moins de temps supplémentaire qu'un octet qui force le débordement dans un deuxième paquet. Le temps de transmission sur le réseau est la composante la plus importante. Pour des tailles inférieures à environ 1K, la dimension d'un paquet, il y a très peu de différence. Ensuite, le temps croit linéairement avec la taille des arguments.

b) Pour les appels à distance, les sémantiques *au moins une fois* de même que *au plus une fois* peuvent être utilisées. Par ailleurs, le protocole UDP et le protocole TCP peuvent être utilisés. Java RMI utilise généralement *au plus une fois* sur TCP alors que les Sun RPC utilisent habituellement *au moins une fois* sur UDP. Montrez dans chaque cas combien de paquets sont échangés pour un appel où aucun paquet n'est perdu, et dans le cas où le paquet de réponse à l'appel à distance est perdu. On suppose que les arguments pour l'appel de même que la réponse sont sans problème plus petits que la taille maximale pour un paquet. Pour chaque scénario, montrez bien la fonction de chaque paquet envoyé et indiquez à chaque fois lorsque la fonction à distance est exécutée. **(2 points)**

Avec RMI, en supposant que la connexion TCP est déjà établie, il y aura le paquet pour la requête, un accusé de réception, le traitement par la fonction à distance, l'envoi de la réponse et un accusé de réception de la réponse. Il est possible que l'accusé de réception de la requête soit jumelé par TCP au paquet contenant la réponse, donnant donc 3 paquets plutôt que 4. Si le paquet de la réponse est perdu, l'accusé de réception de la réponse ne parviendra pas et le serveur renverra sa réponse.

Avec UDP, la requête est envoyée dans un paquet, la fonction à distance est exécutée et le paquet de réponse revient, un total de 2 paquets. Si le paquet de réponse est perdu, la requête sera envoyée à nouveau, la fonction à distance exécutée à nouveau et la réponse reviendra.

c) Le langage C# offre les appels à distance avec l'interface *Remoting*. Une des difficultés avec les appels d'objets à distance est la gestion automatique de la mémoire. En effet, un objet peut être détruit et son espace recyclé seulement lorsqu'il n'est plus utilisé. Il faut donc savoir s'il existe localement des objets rejoignables qui utilisent cet objet, ou s'il en existe dans des applications distantes. Comment peut-on maintenir à jour l'information sur les utilisateurs distants? Comment est-ce que le C# se prémunit contre les utilisateurs distants qui disparaissent (e.g. ordinateur qui saute) sans prévenir qu'ils n'utilisent plus un objet? **(1 point)**

Chaque fois qu'un client utilise une référence réseau, il en avertit le serveur et obtient un bail (droit d'utilisation de durée limitée) sur cet objet. Lorsqu'il cesse d'utiliser l'objet, il peut envoyer un message pour le relâcher, en quelque sorte résilier son bail. S'il veut utiliser l'objet plus longtemps, il peut renouveler le bail. Si le client disparaît sans prévenir, le serveur peut libérer l'objet sans problème à la fin de la période associée au bail.

Question 3 (5 points)

a) Dans une université, le service de fichiers est fourni par trois serveurs CODA. Les trois serveurs redondants contiennent les mêmes fichiers en réplication et la charge est bien répartie entre les trois serveurs. Chaque client en moyenne ouvre 2 fichiers par seconde et ferme un fichier modifié (à envoyer aux serveurs) par 4 secondes. Lors d'une ouverture, le client vérifie si une copie est déjà présente et si elle est à jour (il n'a pas reçu de notification d'invalidation). Sinon, une copie du fichier est prise auprès d'un des serveurs. Lorsqu'un fichier est écrit sur un serveur, le serveur doit envoyer des messages de notification si d'autres clients en ont présentement une copie. Cependant, pour simplifier le problème, on suppose que peu d'usagers accèdent des fichiers différents et les messages de notification ou de validation seront rares et pourront être négligés. On suppose aussi que tous les fichiers ont une longueur inférieure à 64KiO et leur lecture ou écriture constitue un seul accès. Lors d'une ouverture d'un fichier sur un client, une copie du fichier est déjà présente dans 60% des cas et à jour dans 90% de ces cas. Si 200 clients sont actifs en parallèle, combien d'accès en lecture et en écriture par seconde chaque serveur verra-t-il? Si un disque permet d'effectuer un accès en 15ms, combien de disques devrait-on avoir sur chaque serveur, en supposant que la charge est bien répartie entre les disques. **(2 points)**

*Chaque client demande 0.25 écriture par seconde vers chaque serveur. Les lectures représentent .4 + .6 * .1 = .46 des ouvertures, soit 2 * .46 = .92 lectures par seconde sur un des serveurs, ou en moyenne 0.92 / 3 = .3066 par serveur par seconde. Le total est donc de .25 + .3066 = 0.5566 accès par seconde par client. A 200 clients, ceci donne sur chaque serveur 0.5566 * 200 = 111.33 accès/seconde. Un disque permet 1 / .015s = 66.66 accès / seconde. Il faut donc 111.33 accès /seconde / 66.66 accès / seconde / disque = 1.67 disque, donc 2 disques par serveur.*

b) Une compagnie offre des vidéos sur demande par Internet. Elle peut soit avoir des serveurs extrêmement bien connectés, soit utiliser les équipements et ressources des clients pour répartir la charge à l'aide de protocoles comme bittorrent. Un gros film doit sortir et la compagnie s'attend à voir 1000 clients le télécharger la première journée. Le film a une taille de 4GiO. La compagnie obtiendra 2$ par copie mais doit payer $3 par Megabit/s de bande passante pour la journée. Une solution alternative est de permettre à 100 clients de prendre une copie gratuitement, à condition qu'ils agissent comme serveurs (pairs) bittorrent pour les autres clients. On suppose que les téléchargements peuvent être uniformément répartis sur les 24 heures de la journée. Laquelle des deux solutions sera la plus payante pour la compagnie? Justifiez. **(2 points)**

*Dans le premier cas, il faut permettre le téléchargement de 4GiO * 1000 * 8 bits / octet / (24 heures * 3600 secondes / heure) = 397Mbit/s. Le revenu sera donc de $2 * 1000 clients - 397Mbits/s * 3$ / Mbit/s = $809. Dans le second cas, il faudra 4GiO * 100 * 8 bits / octet / (24 heures * 3600 secondes / heure) = 39Mbit/s et le revenu sera de $2 * 900 clients - 39Mbits/s * 3$ / Mbit/s = $1683. La seconde solution sera donc plus avantageuse.*

c) Avec le Global File System, plusieurs noeuds peuvent accéder directement des unités de stockage habituellement connectées par FibreChannel. Ces unités de stockage reçoivent de ces noeuds des commandes simples pour lire et écrire des blocs. Comment est-ce que cela

se compare avec un système de fichiers comme Lustre? Quels sont les avantages de cette approche? **(1 point)**

Avec Lustre, le processeur du serveur de fichiers est directement associé à ses disques. Si le processeur ou l'unité de disques sont défectueux, le serveur n'est plus accessible. Avec GFS, l'unité de disque peut être connectée à plusieurs processeurs qui se coordonnent pour mettre à jour les blocs de disque de manière cohérente afin de modifier le système de fichiers dessus. Si un processeur est en panne, un autre processeur connecté à la même unité de disque peut poursuivre le travail. Les processeurs sont généralement connectés à l'unité de disque via un réseau très rapide de type Fiber Channel.

Question 4 (5 points)

a) Un processus implémente un service de nom de manière récursive. Dans 90% des cas, le processus peut répondre immédiatement après 0.5ms de temps CPU. Autrement, il doit faire une requête à un serveur de nom plus haut dans la hiérarchie, ce qui prend 0.3ms de temps CPU et aussi un temps d'attente pour la réponse. Ce temps d'attente est de 7ms dans 50% des cas, 25ms dans 30% des cas et 75ms dans 20% des cas. Combien de requêtes par seconde ce service peut-il soutenir avec un seul fil d'exécution qui traite les demandes séquentiellement? Combien peut-il en soutenir avec de nombreux fils d'exécution (mais un seul CPU)? Combien de fils d'exécution doit-on avoir au minimum? **(2 points)**

*Le temps d'attente moyen est de .5 * 7ms + .3 * 25ms + .2 * 75ms = 26ms. Pour un traitement en séquentiel, le temps d'une requête sera en moyenne de .9 * .5ms CPU + .1 * (.3ms CPU + 26ms Attente) = .48ms CPU + 2.6ms Attente = 3.08ms. On peut donc soutenir 1000ms / 3.08ms = 324.67 requêtes / seconde. S'il y a toujours des fils d'exécution disponibles, le facteur limitant est le CPU et on peut soutenir 1000ms / .48ms = 2083.3 requêtes / seconde. Il faut 1 fil pour le CPU et 2.6 / .48 = 5.41 pour les requêtes en attente, soit un total de 6.41 donc 7.*

b) Pour une application particulière, vous désirez qu'une mise à jour du service de nom soit faite de manière atomique. Deux clients ne devraient jamais en même temps voir auprès de serveurs de noms différents l'ancienne et la nouvelle adresse associées à un nom. Pour ce faire, vous devez effectuer des requêtes auprès du serveur responsable de ce nom (ajouter, effacer ou modifier un attribut comme l'adresse IP associée à un nom), tout en sachant que cette valeur peut être maintenue en cache pendant un certain temps (spécifé par l'attribut TTL) par les autres serveurs de noms. Indiquez comment vous pourriez obtenir une telle mise à jour atomique dans ce contexte? **(2 points)**

S'il ne s'agissait que de serveurs répliqués, il serait possible d'enlever la valeur actuelle sur chaque serveur dans un premier temps, et d'insérer la nouvelle valeur sur tous les serveurs ensuite, pour obtenir le résultat désiré. La difficulté réside dans le fait que des valeurs sont maintenues en cache par les autres serveurs. Si le temps de validité (TTL) est de 24h, on pourrait enlever la valeur maintenant et ne pas insérer la nouvelle valeur avant 24h, au moment où il n'existerait plus de copie de l'ancienne valeur en cache. Il est aussi possible de diminuer le temps de validité graduellement (le diminuer d'une heure à chaque heure) de

sorte que l'ancienne valeur demeurerait valide dans les cache jusqu'à peu de temps avant le changement, au lieu d'avoir un trou qui pourrait aller jusqu'à 24h selons les disponibilités en cache.

c) Une entreprise veut se doter d'un annuaire comme service de nom pour ses listes d'ordinateurs, ses listes d'usagers, etc. Un premier ingénieur informaticien propose d'utiliser le protocole LDAP et le serveur OpenLDAP. Un autre collègue est d'avis qu'un service de nom n'est rien d'autre qu'une base de donnée et propose de simplement mettre une base de donnée de type SQL avec un service d'accès à distance comme ODBC. Qu'en pensez-vous? Qu'est-ce qu'un service de nom offre qui lui est spécifique? **(1 point)**

Un serveur de nom comme LDAP vient avec un protocole, un API et divers paramètres (cache en mémoire) qui ont été conçus et optimisés pour agir comme serveur de nom. Le déploiement d'un serveur LDAP est donc en général plus simple et plus efficace dans ce contexte. Un service de base de donnée demandera plus de configuration pour arriver à un service qui sera comparable mais possiblement moins efficace. Ceci dit, OpenLDAP peut utiliser une basse de donnée comme service de stockage sur disque pour son annuaire.

Le professeur: Michel Dagenais