ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF4410: Systèmes répartis et infonuagique (Hiver 2017)
3 crédits (3-1.5-4.5)

CORRIGÉ DU CONTRÔLE PÉRIODIQUE

DATE: Lundi le 27 février 2017

HEURE: 10h30 à 12h20

DUREE: 1H50

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Un groupe de N ordinateurs s'échangent des messages de groupe, en utilisant un ordonnancement causal. Chaque membre a un numéro (0 à N-1) qui lui est associé dans le groupe. Ainsi, chaque ordinateur maintient un vecteur de N entrées qui contient, pour chaque membre du groupe, le numéro du dernier message reçu de ce membre et livré aux applications. Chaque ordinateur *i* numérote séquentiellement les messages qu'il envoie au groupe et place dans le vecteur à sa position (*i*) le numéro du dernier message créé. Lorsqu'un message est envoyé au groupe par l'ordinateur *i*, l'ordinateur *i* joint au message son vecteur de N entrées qui inclut le numéro du présent message en position *i*. Le vecteur est utilisé par chaque ordinateur du groupe qui reçoit le message afin d'ordonnancer causalement les messages avant de les livrer aux applications. Les messages suivants, dans un groupe de 4 ordinateurs, sont envoyés et reçus par les différents ordinateurs dans l'ordre spécifié. Pour chaque message, indiquez le vecteur qui sera associé et son contenu, et pour chaque noeud indiquez dans quel ordre les messages seront livrés aux applications. Tous les ordinateurs sont initialemnent rendus à envoyer leur premier message (1_i où *i* est le numéro de l'ordinateur qui envoie le message). **(2 points)**

Ordinateur 0: envoi de 1_0, réception de 1_1, envoi de 2_0, réception de 2_1, envoi de 3_0, réception de 3_1
Ordinateur 1: envoi de 1_1, réception de 1_0, envoi de 2_1, réception de 2_0, envoi de 3_1, réception de 3_0
Ordinateur 2: réception de 1_0, réception de 2_0, réception de 2_1, réception de 3_0, réception de 3_1, réception de 1_1
Ordinateur 3: réception de 1_1, réception de 2_0, réception de 2_1, réception de 1_0, réception de 3_0, réception de 3_1

Lors de chaque envoi, le contenu du vecteur est indiqué. Lors de chaque réception, le vecteur reçu est indiqué. Si le vecteur reçu est inférieur ou égal au vecteur d'envoi pour toutes les positions, sauf celle de l'ordinateur envoyeur où il est supérieur de 1, tous les messages "connus" lorsque ce message avait été envoyé ont été reçus et celui-ci peut être livré aux applications. Sinon, le message reçu est retenu jusqu'à ce que les messages préalables manquants soient reçus. Lorsque le message est livré aux applications, le vecteur de ce message est fusionné au vecteur courant d'envoi (la valeur la plus grande est retenue pour chaque position) ce qui tient compte de la valeur pour l'ordinateur envoyeur qui est supérieure de 1.

Ordinateur 0: envoi de 1_0 *(1,0,0,0), réception de* 1_1 *(0,1,0,0) et livraison (1,1,0,0), envoi de* 2_0 *(2,1,0,0), réception de* 2_1 *(1,2,0,0) et livraison (2,2,0,0), envoi de* 3_0 *(3,2,0,0), réception de* 3_1 *(2,3,0,0) et livraison (3,3,0,0)*
Ordinateur 1: envoi de 1_1 *(0,1,0,0), réception de* 1_0 *(1,0,0,0) et livraison (1,1,0,0), envoi de* 2_1 *(1,2,0,0), réception de* 2_0 *(2,1,0,0) et livraison (2,2,0,0), envoi de* 3_1 *(2,3,0,0), réception de* 3_0 *(3,2,0,0) et livraison (3,3,0,0)*
Ordinateur 2: réception de 1_0 *(1,0,0,0) et livraison (1,0,0,0), réception de* 2_0 *(2,1,0,0), réception de* 2_1 *(1,2,0,0), réception de* 3_0 *(3,2,0,0), réception de* 3_1 *(2,3,0,0), réception de* 1_1 *(0,1,0,0) et livraison (1,1,0,0) qui permet la livraison de* 2_0 *(2,1,0,0) donnant (2,1,0,0), la livraison de* 2_1

(1,2,0,0) donnant (2,2,0,0), la livraison de 3_0 (3,2,0,0) donnant (3,2,0,0), et la livraison de 3_1 (2,3,0,0) donnant (3,3,0,0)
Ordinateur 3: réception de 1_1 (0,1,0,0) et livraison (0,1,0,0), réception de 2_0 (2,1,0,0), réception de 2_1 (1,2,0,0), réception de 1_0 (1,0,0,0) et livraison (1,1,0,0), qui permet la livraison de 2_0 (2,1,0,0) donnant (2,1,0,0), et la livraison de 2_1 (1,2,0,0) donnant (2,2,0,0), réception de 3_0 (3,2,0,0) et livraison (3,2,0,0), réception de 3_1 (2,3,0,0) et livraison (3,3,0,0)

b) Un système envoie des messages par multi-diffusion. Vous hésitez entre l'utilisation d'accusés de réception (positifs) pour les paquets reçus, ou l'utilisation de messages signalant chaque paquet perdu et donc manquant (accusés de réception négatifs). Dans un cas comme dans l'autre, il y aura un accusé par paquet (présent ou manquant selon le cas). Les paquets peuvent avoir une longueur choisie entre 1024 octets et 1048576 octets. La probabilité qu'un message ne soit pas reçu dans ce système (soit perdu) est donnée par: p = longueur en octets / 1200000. Pour quelles longueurs de paquets sera-t-il plus avantageux d'avoir des accusés de réception positifs? Négatifs? **(2 points)**

L'une ou l'autre solution permettra d'avoir éventuellement tous les paquets à destination. On ne peut que minimiser l'envoi des accusés de réception. Si la probabilité de perdre un paquet est inférieure à 0.5, il y aura moins de paquets manquants que présents et des accusés négatifs seront préférables. Si la probabilité est supérieure à 0.5, il sera plus avantageux de signaler les paquets reçus puisqu'ils seront moins nombreux que ceux manquants. Ceci veut donc dire que des accusés négatifs sont plus intéressants si la longueur est inférieure à 600000 octets et positifs autrements.

c) Il existe différents modèles de panne qu'un système peut avoir à détecter, afin d'avertir l'utilisateur et de ne pas fournir des résultats corrompus. Expliquez comment on peut détecter une panne par omission? Une panne de mauvaise réponse aléatoire? Donnez un exemple pour chaque cas. **(1 point)**

Une panne par omission se détecte à l'aide de l'expiration d'un délai. Par exemple, à chaque envoi de paquet, une minuterie peut être activée. Lorsqu'un accusé de réception est reçu, la minuterie est désactivée. Si la minuterie arrive à échéance, elle génère une interruption et le système sait alors que l'accusé de réception n'est pas arrivé et que quelque chose s'est perdu dans le système. Une panne par réponse aléatoire est plus difficile à détecter. Idéalement, il faut avoir une certaine redondance qui permet, avec une bonne certitude, de détecter tout problème. Par exemple, ce peut être plusieurs capteurs identiques en redondance, ou une somme de contrôle sur le contenu des blocs lus d'un disque optique. Autrement, il est peut-être possible d'essayer de détecter une certaine incohérence dans les résultats reçus pour suspecter un problème.

Question 2 (5 points)

a) Dans le cadre du premier TP, vous avez étudié la performance des appels, en fonction de la taille des arguments passés, pour 3 contextes différents: appel normal, appel RMI local et appel RMI distant. Décrivez comment le temps varie dans chaque cas et comment se comparent les résultats entre ces 3 contextes différents. Expliquez ces différences. **(2 points)**

Le temps pour exécuter un appel normal est minime, puisqu'il suffit de sauver quelques registres et sauter à la nouvelle adresse. Si l'argument est passé par pointeur, sa taille a peu d'impact sur la durée de l'appel. Pour un appel RMI local, il faut sérialiser les arguments et écrire dans un tube par des appels système, le processus serveur doit lire du tube et désérialiser les arguments, faire le travail demandé et retourner la réponse en la sérialisant et l'écrivant dans le tube, et finalement le processus appelant doit lire la réponse du tube et la désérialiser. Tout ceci se fait assez rapidement, puisque tout est en mémoire, mais cela demeure beaucoup plus lent que l'appel normal. De plus, le temps croît linéairement avec la taille de l'argument, possiblement avec des sauts lorsque la taille dépasse la taille de la mémoire cache. Pour un appel distant, le transfert par réseau s'ajoute avec toute la latence que cela comporte et l'effet des paquets. La durée varie donc peu lorsque des octets s'ajoutent dans un même paquet, puis un incrément plus important survient s'il faut un paquet de plus pour contenir l'argument.

b) Un serveur reçoit 1000 requêtes par seconde et utilise la sémantique au plus une fois. Chaque requête prend 20ms du client au serveur, 0.5ms dans le serveur, et la réponse prend 20ms pour revenir au client. Le client envoie un accusé de réception au serveur 100ms après avoir reçu la réponse et cet accusé prend 20ms pour aller au serveur qui le traite en temps négligeable. En régime permanent, avec un flot constant de requêtes au serveur, combien de résultats en moyenne sont conservés par le serveur, au cas où une retransmission serait requise, en attente de l'accusé de réception? Si pour .1% des requêtes, l'accusé de réception n'est jamais envoyé par le client, car il est tombé en panne, combien de résultats seront en attente d'accusé de réception après une journée d'opération par le serveur? **(2 points)**

*Une fois la requête reçue par le serveur et traitée, le résultat est sauvegardé en attente de l'accusé de réception, ce qui prend: 20ms pour revenir au client, 100ms pour que le client produise son accusé de réception et 20ms pour retourner au serveur, soit 140ms. Pendant ce temps, puisque nous avons 1s/1000 requêtes = 1ms/requête, nous allons recevoir 140 requêtes, soit autant de résultats conservés simultanément sur le serveur. Si .1% des requêtes restent prises dans le système, cela fait .001 * 1000 requêtes / s = 1 requête / s. Nous avons donc en 24 heures 24 * 60 * 60 = 86400 résultats qui traînent par erreur dans le système, en plus des 140 pour les requêtes actives.*

c) Quels sont les avantages du système d'appels à distance *gRPC* avec les *Protocol buffers* proposé par Google? **(1 point)**

Le système gRPC présente plusieurs avantages. Il est simple à utiliser, supporte plusieurs langages, est très efficace étant binaire avec compression des entiers, et les messages sont auto-décrits. Ceci permet de traiter des messages dont le type n'est pas connu à l'avance et surtout d'assurer une certaine compatibilité vers l'avant et vers l'arrière (vieux serveur qui reçoit et ignore des nouveaux champs inconnus, ou nouveau serveur qui prend des valeurs par défaut pour des nouveaux champs non fournis par le vieux client).

Question 3 (5 points)

a) Vous devez effectuer un calcul de rendu d'image par lancer de rayon, à partir d'une scène 3D, pour un travail en infographie. Vous décidez d'utiliser une grappe de calcul dans le nuage pour

ce faire et devez décider du nombre d'instances à activer afin d'aller le plus vite possible, étant donné que votre échéance vient rapidement. Un serveur doit tout d'abord envoyer séquentiellement à chaque instance la définition de l'espace 3D (éléments de la scène, mouvements, position de l'observateur), ce qui prend 5s pour chaque envoi. Par la suite, le calcul des 200000 images qui constitueront un film est réparti uniformément entre les différentes instances. Le calcul de chaque image sur une instance prend 4s. Ceci terminé, le serveur doit recevoir séquentiellement toutes les images calculées sur les instances, ce qui prend 1ms par image. Quel est le temps total pour compléter ce travail et obtenir toutes les images sur le serveur si la grappe contient n instances? Quelle est la valeur de n qui permet de minimiser ce temps? **(2 points)**

Le temps requis est de $n \times 5s + 200000 \times 4s/n + 200000 \times .001s$. La valeur optimale de n est obtenue lorsque la dérivée est nulle soit $5s - 200000 \times 4/n^2 = 0$ ou $5 \times n^2 = 800000$ et $n = 400$. La valeur optimale est donc de 400 instances.

b) Pour exécuter une machine virtuelle, trois approches sont possibles: la virtualisation complète, la paravirtualisation, ou les conteneurs. Quels sont les avantages et inconvénients de chaque approche? **(1 point)**

La virtualisation complète permet d'exécuter n'importe quelle image et présente donc le moins de contraintes. Par contre, le surcoût est le plus élevé des trois méthodes. La paravirtualisation demande une configuration particulière du noyau du système d'exploitation, qui est compatible avec le gestionnaire de machines (para)virtuelles utilisé. Le surcoût pour la (para)virtualisation est moins élevé que pour la virtualisation complète. Les conteneurs doivent utiliser le système d'exploitation de l'hôte. Il n'y a donc aucune flexibilité à ce niveau. Toutefois, il n'y a pas vraiment de surcoût par rapport à une exécution native sur une machine physique.

c) Sur Android, les applications peuvent être écrites en C/C++ et être compilées ou être écrites en Java. Quels sont les avantages de chaque approche? **(1 point)**

Une application native, compilée, offre normalement la meilleure performance. Elle doit toutefois être disponible pour le bon matériel et la bonne version du système d'exploitation et des librairies au niveau binaire. Une application en Java sera livrée sous forme de bytecode qui est indépendant du matériel. Il faut que la version des librairies installées soit compatible au niveau source, ce qui est beaucoup moins contraignant. L'application pourrait toutefois être un peu moins performante.

d) Quelle est l'utilité de pouvoir migrer des machines virtuelles d'un noeud physique à l'autre? Pourquoi décide-t-on de migrer une machine virtuelle? **(1 point)**

L'utilité de migrer des machines virtualles (VM) est de pouvoir facilement s'adapter aux circonstances. Si une VM a besoin de plus de ressources, il est possible de la migrer vers un noeud plus puissant. Si le taux d'utilisation des noeuds physiques dans un centre de données diminue, il est possible de consolider les VM sur un plus petit nombre de noeuds et d'en éteindre quelques-uns. Si un ordinateur requiert de l'entretien, il est possible de migrer vers d'autres noeuds les VM qu'il supporte. D'autres raisons peuvent aussi motiver une migration comme mettre ensemble deux VM qui communiquent beaucoup ensemble, rapprocher une VM du réseau avec lequel elle interagit, favoriser l'utilisation des noeuds physiques où l'électricité coûte moins cher ou est plus verte...

Question 4 (5 points)

a) Un réseau de clients est servi par 3 serveurs CODA répliqués. Chaque client ouvre en moyenne 5 fichiers par seconde et en ferme autant. Lors de l'ouverture, le fichier n'est pas présent localement dans 25% des cas et doit être lu à partir d'un serveur. Lors de la fermeture, le fichier a été modifié dans 10% des cas et doit alors être écrit sur chacun des 3 serveurs. Chaque requête de lecture prend 5ms de CPU sur un serveur et en plus, dans 30% des cas, une lecture du disque de 30ms. Chaque écriture prend sur chaque serveur 10ms de CPU et 40ms de temps du disque. Si chaque serveur possède 2 CPU et 2 disques, que les requêtes sont bien réparties entre les serveurs, les CPU et les disques, et que le service utilise plusieurs threads afin de servir en parallèle les requêtes, quel est le nombre maximal de clients possible avant que le service ne sature? Combien de thread devrait-on rouler sur chaque serveur? **(2 points)**

*Pour un client, on a 5 ouvertures / s générant 5 / s * .25 = 1.25 lecture / s. On a aussi 5 fermetures / s soit 5 / s * .1 = 0.5 écriture / s. Sur un des serveurs, cela donne 1.25 / 3 lecture / s (lectures réparties entre les 3 serveurs) soit 1.25 / 3 * 5ms = 2.08ms CPU et 1.25 / 3 * 0.3 * 30ms = 3.75ms disque. Cela donne aussi sur chaque serveur 0.5 écritures / s soit 0.5 * 10ms = 5ms CPU et 0.5 * 40ms = 20ms disque. Le total pour un serveur est donc 7.08 ms CPU et 23.75 ms disque. Le goûlot d'étranglement est au niveau des disques. Avec 2 disques, on peut supporter 2 * 1000ms / 23.75ms = 84.21 donc 84 clients. Il faut un thread par ressource, CPU ou disque, soit un total de 4 threads.*

b) Dans Glusterfs, plusieurs types de services de fichiers sont offerts. Supposons que nous ayons 4 fichiers à stocker sur un service de fichiers Glusterfs offert par 4 serveurs qui contribuent chacun une brique. Montrez quel fichier (ou morceau de fichier) pourrait se trouver sur chaque serveur pour chacune des configurations suivantes: i) serveurs répartis, ii) serveurs répliqués, iii) serveurs répartis et répliqués, iv) serveurs agrégés (striped). **(2 points)**

Pour le service réparti i), chaque fichier pourrait se retrouver sur un serveur différent puisqu'ils se répartissent la charge. Pour le service répliqué ii) une copie de chacun des 4 fichiers pourrait se retrouver sur chacun des 4 serveurs de manière à avoir une redondance quadruple. Dans le service réparti et répliqué, en supposant 2 serveurs répartis de 2 serveurs répliqués, on pourrait avoir 2 fichiers qui se retrouvent tous deux sur 2 des serveurs, et les 2 autres fichiers qui se retrouvent répliqués sur les 2 autres serveurs. Pour des serveurs agrégés, un même fichier très gros pourrait se retrouver avec 1/3 de son contenu sur chacun de 3 des serveurs et les 3 autres fichiers sur le quatrième serveur.

c) Lorsque le client d'un service de fichiers effectue une lecture, cette lecture peut être servie dans la cache d'E/S du client, sinon dans la cache d'E/S du serveur, ou sinon à partir du disque. Comment se compare la vitesse dans chaque cas? Est-ce qu'il y a un inconvénient à prendre la contenu à partir de la cache du client? De la cache du serveur? **(1 point)**

La cache du client sera la plus rapide, à la vitesse de la mémoire qui contient la cache d'E/S. La cache d'E/S sur le serveur est accédée à travers le réseau. Le délai de réseau ajoute donc au temps d'accès. Pour un accès servi à partir du disque du serveur, il faut attendre après le réseau et après le disque, ce qui ajoute autant au délai. Il n'y a pas de problème à lire de la cache d'E/S du serveur, puisqu'elle est nécessairement cohérente avec le contenu du disque (aussi à

jour ou plus à jour). La cache du client peut ne pas être au courant des dernières mises à jour et un problème de cohérence se pose. Sur NFS, on revalide le contenu dans la cache avant de l'accéder si la dernière validation date de plus de 3 secondes. Sur d'autres systèmes, on compte sur les notifications de changement en provenance du serveur pour être averti de tout bloc dont le contenu devient caduc.

Le professeur: Michel Dagenais