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 DE L'EXAMEN FINAL

DATE: Jeudi le 20 avril 2017

HEURE: 9h30 à 12h00

DUREE: 2H30

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Le serveur DNS d'un fournisseur de service Internet peut servir des requêtes DNS soit de manière itérative ou de manière récursive. De manière itérative, le serveur traite la requête avec son CPU pendant 2ms, lit en plus son disque dans 20% des cas pendant 20ms, et retourne dans 40% des cas la réponse demandée et dans 60% des cas une redirection vers un serveur plus haut dans la hiérarchie. En recevant une redirection, le client doit aller chercher la réponse sur un autre serveur, ce qui lui prend 10ms. Si le serveur est configuré pour fonctionner de manière récursive, il prendra aussi 2ms de CPU, et en plus 20ms de disque dans 20% des cas. Ensuite, il retournera la réponse demandée dans 90% des cas et dans 10% des cas devra faire une requête et attendre la réponse avant de la retourner, ce qui lui demande en plus 1ms de CPU et 20ms d'attente. Quel sera le délai moyen vu par un client pour obtenir la réponse demandée dans chaque cas (récursif ou itératif)? Quel est le nombre de requêtes par seconde que peut soutenir le serveur, s'il n'utilise qu'un seul thread, de manière itérative? De manière récursive? **(2 points)**

De manière itérative, une requête prend $2ms + .2 \times 20ms = 6ms$*. A cela, il faut ajouter 10ms dans 60% des cas soit 6ms, pour un total de 12ms. De manière récursive, le délai est de* $2ms + .2 \times 20ms + .1 \times (1ms + 20ms) = 8.1ms$*. La manière récursive est donc plus intéressante de ce point de vue, puisqu'elle fait augmenter le taux de succès en cache du serveur. Dans le premier cas, le serveur peut traiter une requête en 6ms, puisqu'après le client est redirigé vers un autre serveur et le serveur du fournisseur Internet est libre pour la prochaine requête, soit* $1000ms/s/6ms/r = 166$ *requêtes par seconde. Dans le second cas, il peut fournir* $1000ms/s/8.1ms/r = 123$ *requêtes / seconde.*

b) Sur un serveur DNS en autorité, la base de donnée entre entièrement en mémoire et l'ingénieur responsable a réussi à faire traiter chaque requête en 3us (1us recevoir le paquet UDP de requête, 1us chercher la réponse en mémoire, 1us renvoyer la réponse). Chaque paquet UDP (requête ou réponse) occupe en moyenne 100 octets. Pour simplifier le calcul, on suppose que les transferts peuvent se faire à la bande passante indiquée, on néglige les autres effets comme la latence d'envoi pour les paquets. Le serveur est connecté à deux réseaux qui peuvent chacun soutenir 100 mégabits/s par seconde simultanément dans chaque direction. Un usager malicieux a réussi à prendre le contrôle d'un certain nombre de processeurs dans des caméras IP et les utilise pour effectuer une attaque en déni de service sur le serveur DNS. Chaque caméra IP est capable de générer 100000 paquets de requête par seconde et est connectée par un réseau à 1mégabit/s. Les requêtes sont émises avec de fausses adresses de retour et ne reviennent jamais aux caméras. Combien de caméras sont requises pour saturer ce serveur DNS, en tenant compte de la capacité de leur processeur et de leur réseau respectifs? **(2 points)**

Le CPU du serveur peut traiter $1000000us/s/3us/r = 333333r/s$*. Le réseau peut laisser passer* $2 \times 100000000bits/s/(100octets/p \times 8bits/octet) = 250000paquets/s$*. Le facteur limitant est le réseau et le serveur peut donc traiter 250 000 requêtes par seconde. Chaque caméra peut envoyer* $1000000bits/s/(100octets/p \times 8bits/octet) = 1250paquets/s$*. Le serveur sera donc saturé à partir de* $250000/1250 = 200$ *caméras malicieuses.*

c) Pour les requêtes de recherche avec le protocole LDAP, il est possible de spécifier une limite sur la longueur de la réponse ainsi que sur le traitement requis. Pourquoi de tels paramètres ont-ils été incorporés à ce protocole? Dans quelle situation est-ce utile? **(1 point)**

Les répertoires LDAP peuvent être très gros et les requêtes assez complexes (filtres compliqués). Le client peut donc vouloir spécifier des limites pour ne pas recevoir trop d'information ni surcharger le serveur. Le serveur peut aussi à l'interne imposer des quotas par requête, selon le niveau de privilège du client qui fait la requête. Par exemple, un client pourrait demander la liste des usagers, pensant qu'ils ne sont qu'une dizaine, et se retrouver avec une réponse qui en contient des dizaines de milliers.

Question 2 (5 points)

a) Un client A effectue deux requêtes auprès d'un serveur de temps B. La première requête part de A à 9h00m40.000 et arrive à B à 9h00m20.010, puis la réponse part de B à 9h00m20.060 et arrive à A à 9h00m40.100. La seconde requête part de A à 9h01m50.000 et arrive à B à 9h01m30.030, puis la réponse part de B à 9h01m30.130 et arrive à A à 9h01m50.140. Quelles sont les valeurs de décalage et d'incertitude calculées pour chaque requête? Laquelle valeur (i.e. de quelle requête) devrait-on utiliser? Peut-on combiner les informations des requêtes pour avoir un meilleur estimé du décalage? **(2 points)**

Pour la première requête on a $a = 9h00m20.010 - 9h00m40.000 = -19.990$ *et* $b = 9h00m20.060 - 9h00m40.100 = -20.040$ *donc le décalage est de (a+b)/2 +/- (a-b)/2 soit -20.015s +/- .025s ou de -19.99s à -20.04s. Pour la seconde requête on a* $a = 9h01m30.030 - 9h01m50.000 = -19.97$ *et* $b = 9h01m30.130 - 9h01m50.140 = -20.010$ *donc le décalage est de -19.99 +/- .02 ou de -19.97 à -20.01. La deuxième requête donne un résultat plus précis et pourrait être retenue. En combinant les deux valeurs, on voit que l'intersection des intervalles donne de -19.99 à -20.01 ou -20s +/- .01s, ce qui est encore plus précis. Il faut bien comprendre ici que les mesures de temps sont très précises. L'incertitude est sur le délai réseau dans chaque direction. Dans ce cas-ci, par chance, le délai a été très court dans une direction pour un échange et très court dans l'autre direction pour l'autre échange. En recoupant les informations des deux intervalles possibles, on parvient à profiter de toute cette information et raffiner nos bornes. Une autre manière de considérer le problème est d'utiliser l'envoi de la première requête et la réception associée à la seconde requête, ce qui donnerait* $a = 9h00m20.010 - 9h00m40.000 = -19.990$ *et* $b = 9h01m30.130 - 9h01m50.140 = -20.010$*, donc un décalage de -20s +/- .01s.*

b) Dans le cadre du travail pratique 2, vous avez implémenté un service de calcul sécurisé (pas de réponse malicieuse) et un service non sécurisé (réponses possiblement malicieuses). Quelle était votre stratégie pour détecter et laisser tomber les réponses malicieuses? Quel était le surcoût de cette détection (temps CPU pour calcul sécurisé versus calcul non-sécurisé, mais avec très peu de réponses malicieuses finalement)? Quelle était la probabilité de ne pas détecter une réponse malicieuse si les réponses malicieuses sont générées indépendamment sur chaque serveur de calcul? Si elles sont générées de manière concertée? **(2 points)**

Le surcoût de la détection est de faire chaque calcul deux fois, donc le double du temps CPU requis autrement. Si les réponses malicieuses sont générées de manière indépendante et sont réparties uniformément dans l'intervalle, la chance que la seconde réponse malicieuse donne le même résultat que la première réponse malicieuse est de par exemple $1/4000$ *si la réponse peut aller de 0 à 3999. La probabilité d'avoir une réponse malicieuse non détectée est donc la probabilité d'avoir deux réponses malicieuses fois la probabilité que la réponse malicieuse ne*

soit pas détectée. Par exemple, si le taux de réponse malicieuse est de 80% sur chaque serveur de calcul, ce serait $.8 \times .8 \times (1/4000)$*. Si les réponses malicieuses sont concertées, la probabilité sera celle d'avoir deux réponses malicieuses, soit dans l'exemple* $.8 \times .8$*.*

c) Nous avons vu deux algorithmes qui peuvent être utilisés pour une élection dans un système réparti, l'élection hiérarchique et l'algorithme de Paxos. L'élection hiérarchique est a priori beaucoup plus simple. Quels sont donc les avantages de l'algorithme de Paxos? Donnez un exemple où seul l'algorithme de Paxos fonctionnerait correctement? **(1 point)**

L'élection hiérarchique est basée sur un délai et est donc synchrone. Par exemple, en cas de long délai à répondre, ou en cas de partition du réseau, on peut se retrouver avec deux élus simultanément pendant une certaine période. L'algorithme de Paxos est plus complexe mais beaucoup plus robuste. Son fonctionnement correct ne sera pas affecté par un partitionnement ou des délais trop longs. Il est aussi plus général puisqu'il fournit un consensus, dont l'élection est un cas particulier.

Question 3 (5 points)

a) Lesquelles des transactions T, U et V pourraient être validées si une validation en reculant était utilisée pour vérifier la cohérence des transactions? Une validation en avançant? **(2 points)**

T: Début
U: Début
V: Début
T: Read(e)
T: Read(a)
T: Read(d)
U: Read(a)
U: Read(e)
T: Write(a,21)
T: Write(b,12)
T: Compléter
U: Read(b)
V: Read(b)
U: Write(c,3)
U: Write(f,8)
U: Compléter
V: Read(c)
V: Write(d,12)
V: Write(e,1)
V: Compléter

Pour la validation en reculant, la première transaction à terminer, T, est toujours validée. Au moment de terminer U, T qui précède a écrit (a, b), alors que U a lu (a, b, e). Ainsi U ne peut être validée. Rendu à V, seulement T a complété en écrivant (a,b), alors que V a lu (b, c). V ne

peut compléter. En avançant, la transaction T regarde ce qu'elle a écrit (a, b) versus ce que U et V ont lu (a, e). T ne peut compléter car elle met en péril U. Rendu à compléter U, ses écritures sont (c, f) alors que les lectures de V jusqu'ici sont (b). Ainsi, U peut compléter. V, qui est la dernière du groupe, peut naturellement compléter.

b) Supposons que les opérations listées en a) pour les transactions T, U et V sont en fait des transactions réparties. Un premier serveur X contient les variables a et b, un second serveur Y contient les variables c et d, alors que le troisième serveur Z contient les variables e et f. Le système utilise un protocole de fin de transaction atomique à deux phases. En supposant que les trois transactions puissent être validées avec les opérations telles que listées, que retrouverait-on dans le journal de chacun des serveurs (X, Y et Z)? Expliquez qu'est-ce qui est écrit à quel moment. **(2 points)**

Au moment de compléter T, les variables (a, b) sont écrites et se trouvent sur X. Le coordonnateur écrit (prepare T), envoie un message de préparation à X qui écrira ces nouvelles valeurs (T: a=21, b=12) ainsi que le fait qu'il accepte (prepare T). En recevant l'acceptation de X, le coordonnateur écrit (commit T) et envoie le message à X qui écrit (commit T). Au moment de compléter U, les variables (c, f) sont écrites et se trouvent sur Y et Z. Le coordonnateur écrit (prepare U) et envoie un message de préparation à Y et Z. Le serveur Y écrit la nouvelle valeur (U: c=3) ainsi que le fait qu'il accepte (prepare U) et Z fera de même, (U: f=8; Prepare U). Après avoir reçu les deux acceptations, le coordonnateur écrit (commit U) et envoie le message à Y et Z qui écrivent chacun dans leur journal (commit U). Finalement, pour V, les écritures sont (d, e) et se trouvent sur Y et Z. Le coordonnateur écrit (prepare V) et envoie un message de préparation à Y et Z. Le serveur Y écrit la nouvelle valeur (V: d=12) ainsi que le fait qu'il accepte (prepare V) et Z fera de même, (V: e=1; Prepare V). Après avoir reçu les deux acceptations, le coordonnateur écrit (commit V) et envoie le message à Y et Z qui écrivent chacun dans leur journal (commit V).

c) Pour les transactions locales, les algorithmes de contrôle optimiste de la concurrence, validation en avançant et validation en reculant, peuvent être intéressantes. Sont-elles applicables pour des transactions réparties? Expliquez. **(1 point)**

Pour les transactions réparties, le verrouillage strict à deux phases ou le contrôle par les estampilles de temps peuvent être utilisés. Le contrôle optimiste de la concurrence par validation de la cohérence (en avançant ou en reculant) n'est pas utilisé car il demande de tout figer pendant la vérification de la cohérence, ce qui n'est généralement pas acceptable dans un système réparti.

Question 4 (5 points)

a) Un système transactionnel en ligne, pour des ventes aux enchères, utilise de la redondance à plusieurs niveaux. Il est connecté à 2 fournisseurs Internet et peut fonctionner tant que l'un des deux est opérationnel (i.e. chaque serveur est connecté aux deux réseaux). Il y a ensuite 3 serveurs de façade alors qu'il suffit d'un seul pour fonctionner. Les serveurs de façade font des requêtes à 2 serveurs de base de données redondants alors qu'un seul suffit. Deux unités de disque RAID sont utilisées. Chaque unité de disque RAID fonctionne si au moins 3 de ses 4 disques sont fonctionnels. Deux configurations sont possibles. Dans la première, chaque unité RAID est connectée à un seul des deux serveurs de base de données; un serveur ne fonctionnera

que si son unité RAID fonctionne. Dans la seconde configuration, chaque serveur de base de données peut accéder aux deux unités RAID; un serveur ne fonctionnera que si au moins une des 2 unités est fonctionnelle. Si la probabilité de panne est de .1 pour un fournisseur Internet, .2 pour un serveur de façade, .15 pour un serveur de base de données et .25 pour un disque, et que la probabilité de panne est négligeable pour les autres composantes, quelle sera la probabilité que le service soit disponible aux clients pour chacune des deux configurations? **(2 points)**

Pour que le service fonctionne, il faut que simultanément fonctionnent le service Internet, le service façade et le service de base de données. Pour la première configuration, il faut associer la panne de l'unité RAID connectée à la panne du serveur de base de donnée. Dans la seconde configuration, le service de disque est dissocié et doit simplement lui aussi être disponible. Pour le service Internet, il y a panne si les deux fournisseurs sont en panne, une probabilité de $.1^2 = .01$. *La probabilité de fonctionner est donc de* $1 - .01 = 0.99$. *Pour les trois serveurs de façade, il y a panne si les trois serveurs sont en panne, une probabilité de* $.2^3 = .008$ *ou* $1 - .008 = .992$ *de fonctionner. Pour les unités RAID, la probabilité de fonctionner est celle d'avoir 0 ou 1 disque en panne, ou 4 ou 3 disques fonctionnels, soit* $(1-0.25)^4 + 4!/(3! \times (4-3)!) \times (1-0.25)^3 \times 0.25 = 0.73828125$. *Dans la première configuration, un serveur de base de donnée fonctionne si lui et son unité fonctionnent, soit* $(1 - 0.15) \times 0.73828125 = 0.627539063$. *Le service fonctionnera sauf si les deux sont en panne* $(1-0.627539063)^2 = 0.13872715$, *soit* $1-0.13872715 = 0.86127285$. *Ceci donne donc un service fonctionnel si toutes ces composantes sont fonctionnelles soit* $0.99 \times .992 \times 0.86127285 = 0.845838841$. *Dans la seconde configuration, le service de base de donnée a une probabilité de panne de* $0.15^2 = 0.0225$ *et de fonctionner de* $1 - 0.0225 = 0.9775$. *Le service de disque a une probabilité de panne de* $(1 - 0.73828125)^2 = 0.068496704$ *et de fonctionner de* $1 - 0.068496704 = 0.931503296$. *Le service sera fonctionnel si toutes ces composantes sont fonctionnelles soit* $0.99 \times 0.992 \times 0.9775 \times 0.931503296 = 0.894227515$. *La seconde configuration est donc plus robuste que la première.*

b) Dans le cadre du travail pratique 3, vous avez utilisé des gabarits Heat afin de spécifier le comportement d'un service. Que doit-on mettre dans le gabarit afin de répartir la charge entre plusieurs serveurs? Que doit-on ajouter au gabarit Heat afin d'ajuster selon la charge le nombre de serveurs entre lesquels le travail est réparti? Donnez le type des principales ressources requises et leur fonction. **(2 points)**

Pour répartir la charge, il faut un groupe de machines, OS::Neutron::Pool, une adresse IP associée au groupe, OS::Neutron::FloatingIP, un moniteur qui vérifie les noeuds fonctionnels, OS::Neutron::HealthMonitor, et finalement le répartiteur de charge, OS::Neutron::LoadBalancer. Pour ajuster le nombre de serveurs selon la charge, il faut des conditions de déclenchement, OS::Ceilometer::Alarm associés, qui activent des politiques pour augmenter ou diminuer le nombre de noeuds, OS::Heat::ScalingPolicy, et finalement le groupe avec mise à l'échelle automatique, OS::Heat::AutoScalingGroup. La mise à l'échelle du groupe est une activité séparée mais complémentaire à ce que fait le répartiteur de charge.

c) Le système Google Wide Profiling permet de comparer la performance fournie par les différentes versions d'un logiciel, par différents types de matériel, ou même de calculer le coût global (temps CPU) pour une fonction donnée. Comment cela fonctionne-t-il? Est-ce que le surcoût est important pour obtenir ces informations? Sont-elles fiables? Expliquez. **(1 point)**

Le système GWP échantillonne les contenus de la pile à intervalle régulier sur un échantillon de noeuds. Cette information permet de voir combien consomme en moyenne chaque fonction ou ligne de code. Ceci peut être corrélé avec la version des logiciels et avec le type de matériel. On peut donc savoir le coût global de chaque fonction ou l'efficacité relative de chaque version ou type de matériel (prend plus de temps pour effectuer le même travail en moyenne). Le surcoût est facilement contrôlé en ajustant la fréquence d'échantillonnage et le nombre de noeuds faisant partie de l'échantillon. Ceci nous donne donc un compromis entre le surcôut et la vitesse à laquelle on peut avoir assez de données pour obtenir une bonne précision. Avec de l'échantillonnage, il y a toujours une certaine imprécision, surtout pour détecter des événements rares. Pour les événements fréquents, qui sont souvent les plus importants, la moyenne sera faite sur un grand nombre d'échantillons et les données seront passablement fiables.

Le professeur: Michel Dagenais