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 DE L'EXAMEN FINAL

DATE: Vendredi le 16 décembre 2016

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) Sur les ordinateurs Linux, le fichier `resolv.conf` permet de lister plusieurs serveurs de nom (DNS), qui seront interrogés dans l'ordre selon lequel ils apparaissent dans le fichier, jusqu'à ce qu'une réponse soit obtenue. Ceci offre donc une tolérance aux pannes de serveur DNS par redondance, lorsqu'une traduction de nom à adresse IP est requise. Si 5 serveurs sont listés dans le fichier de configuration, combien de serveurs en panne par omission ce mécanisme peut-il tolérer avant de cesser de fonctionner? Si vous étiez dans un environnement hostile, où certains serveurs compromis fourniraient de mauvaises réponses, comment pourriez-vous modifier ce mécanisme pour aussi tolérer des pannes byzantines? Dans ce cas, combien de serveurs en panne (arrêtés ou compromis), pourriez-vous tolérer tout en maintenant le service opérationnel? **(2 points)**

Avec des pannes par omission, il suffit d'un seul serveur opérationnel pour fonctionner, soit jusqu'à 4 serveurs en panne qui peuvent être tolérés. Pour les pannes byzantines, il faudrait comparer les réponses et retenir celle qui apparaît le plus souvent. Dans un tel cas, jusqu'à 2 serveurs en panne pourraient être tolérés (3 bonnes réponses contre 2 mauvaises); avec 3 serveurs compromis, la mauvaise réponse pourrait l'emporter.

b) Un serveur de nom contient un CPU à un coeur ainsi qu'un disque. Chaque requête prend 150us (us pour microseconde) de temps CPU pour traiter la requête au complet lorsque la réponse est présente en mémoire centrale. Lorsque la réponse n'est pas présente en mémoire centrale, ce qui arrive pour 10% des requêtes reçues, il faut ajouter un 150us supplémentaire de temps CPU et une lecture au disque qui prend 500us (disque rapide de type SSD). Finalement, lorsque la réponse n'est pas en mémoire ni sur disque, ce qui arrive pour 1% des requêtes reçues, il faut ajouter à ces deux temps le temps pour une demande récursive au serveur plus haut dans la hiérarchie, ce qui ajoute 200us de temps CPU et 20ms (ms pour miliseconde) d'attente. Si le processus pour ce service traite séquentiellement les requêtes avec un seul fil d'exécution (thread), combien de requêtes par seconde peut-il servir? Si plusieurs fils d'exécution (thread) sont utilisés de manière à maximiser la performance, combien de requêtes par seconde peut-il servir et combien de fils d'exécution doit-on utiliser? **(2 points)**

*Ce service requiert en moyenne par requête: pour le CPU 150us + 150us * 0.1 + 200us * 0.01 = 167us, pour le disque 500us * 0.1 = 50us, pour l'attente 20000us * .01 = 200us, pour un total de 417us. Avec un seul fil d'exécution, ce système peut servir 1s / 417us / requête = 2398.08 requêtes / seconde. Le goûlot d'étranglement ici est le CPU plutôt que le disque. Avec assez de fils d'exécution, le système pourrait servir 1s / 167us / requête = 5988.02 requêtes / seconde. Il faut s'assurer d'avoir assez de fils pour couvrir les besoins du CPU (1) et le reste de la requête en proportion du temps pris par chaque composante. On regarde donc le total pris pour la requête sur le temps pris par le CPU. Ceci demanderait 1 fil * 417us / 167us = 2.497 fils, donc 3 fils d'exécution.*

c) Il y a eu, au cours des dernières années, quelques épisodes où des attaques en déni de service ont été lancées contre les serveurs de nom (DNS) principaux sur l'Internet. Dans un tel cas, il peut arriver que les serveurs racines, complètement submergés, soient pratiquement inopérants. Cependant, la plupart des utilisateurs n'ont rien remarqué et ils ont pu continuer à utiliser la plupart des sites populaires. Toutefois, certains sites moins populaires, mais aussi quelques sites

populaires qui font un usage plus dynamique des serveurs de nom, n'étaient effectivement pas accessibles en raison de l'attaque. Comment expliquez-vous cela? **(1 point)**

Le service de nom (DNS) fonctionne efficacement en grande partie en raison des caches maintenues dans les serveurs à différents niveaux. Généralement, les réponses fournies par les serveurs de nom en autorité viennent avec une échéance de 24 heures. Ainsi, les adresses populaires sont en cache et peuvent fonctionner pour 24 heures, même si le serveur de nom au plus haut niveau n'est pas disponible. Si une attaque devait réussir à se poursuivre pour plus de 24 heures, ceci pourrait avoir des conséquences plus importantes. Lorsqu'un site est moins populaire et que son adresse ne se trouve pas en cache du serveur interrogé, il peut effectivement y avoir un problème. Une autre catégorie de site à risque est celle des sites qui utilisent le service de nom de manière très dynamique, par exemple pour répartir la charge entre plusieurs serveurs. Souvent les adresses retournées auront une échéance très courte, par exemple de quelques minutes, et les valeurs en cache deviennent rapidement invalides.

Question 2 (5 points)

a) A l'occasion du nouvel an, vous désirez inviter à la maison les parents de votre conjoint. Cependant, ceux-ci, des ingénieurs, accordent une très grande importance à la précision du temps. A 10h05m00s, heure de votre domicile, vous envoyez un drone avec caméra lire le temps sur l'horloge atomique des parents qui n'habitent pas très loin. Rendu là, le drone filme l'horloge qui affiche 10h10m05s, se pose 5 minutes pour reposer ses moteurs, et filme à nouveau l'heure sur l'horloge atomique qui est maintenant 10h15m05s. Le drone revient à votre domicile à 10h14m00s, heure de votre domicile. Vous pouvez alors visionner le film et connaître les deux temps qui ont été filmés, à l'arrivée et au départ du drone de chez les parents. Quel ajustement apporterez-vous à l'horloge de votre domicile pour la synchroniser avec l'horloge atomique des parents suite à ces lectures? Quelle est l'incertitude sur cet ajustement? Si vous connaissez la direction et la force du vent ainsi que la localisation du domicile des parents, est-ce que cela vous permettrait d'obtenir un meilleur ajustement du temps? **(2 points)**

Nous pouvons ainsi calculer a = t(i-2) - t(i-3) = 10h10m05s - 10h05m00s = 5m05s et b = t(i-1) - t(i) = 10h15m05s - 10h14m00s = 1m05s. L'ajustement est donc de (a+b)/2 = (5m05s + 1m05s) / 2 = 3m05s (avancer l'heure d'autant). L'incertitude est de (a-b) = (5m05s - 1m05s) = 4m00s, soit plus ou moins 2 minutes. S'il n'y a pas de vent et que les temps d'aller et de retour sont strictement égaux, l'ajustement sera parfait. Puisque le drone est parti de 10h05 à 10h14, soit 9 minutes, et qu'il est resté là-bas 5 minutes, le temps de trajet total est de 4 minutes, soit 2 minutes dans chaque direction. Ainsi, puisque le drone part à 10h05 et lorsqu'il arrive 2 minutes plus tard (à 10h07 de votre heure) il voit 10h10m05s, on comprend que l'ajustement à appliquer soit effectivement de 3m05s. En connaissant la position des deux endroits, le temps de trajet total du drone et la vitesse du vent, on pourrait calculer la vitesse effective que le drone a eu dans chaque direction, en tenant compte du vent, et savoir quelle est l'asymétrie du temps d'aller par rapport à celui de retour. En tenant compte ainsi du temps de trajet réel, on peut faire un ajustement plus précis. Ainsi, si après 1 minute de trajet (10h06 de votre heure) on lit 10h10m05, cela signifierait que le bon ajustement serait plutôt de 4m05.

b) Le 31 décembre prochain, une seconde sera ajoutée en fin de journée de sorte qu'il sera éventuellement 23h59m60s, et que seulement une seconde plus tard il sera 0h00m00s. Pourquoi l'organisme en charge de maintenir le temps fait-il une telle modification qui pourrait perturber le décompte de nombreux fêtards lors de la veille du jour de l'an? La compagnie Google, soucieuse d'aplanir le problème, offre un service public de serveurs de temps (NTP) et compte cacher cette seconde supplémentaire, qui pourrait causer des ennuis à plusieurs logiciels, en ralentissant le temps d'une seconde sur une période allant de 10 heures avant et 10 heures après la fin de la journée du 31 décembre. Vous désirez profiter de ce ralentissement du temps pour améliorer votre record de 20s pour courir un 100 mètres. Que deviendra ce temps de 20s lorsque mesuré en utilisant le temps ralenti par Google? **(2 points)**

*L'organisme en charge de coordonner le temps veut aligner notre heure sur le mouvement de la Terre. Cependant, la vitesse de rotation de la terre varie un peu dans le temps en fonction notamment des marées et des mouvements dans le noyau de la Terre. Il faut donc ajouter une seconde de temps en temps pour maintenir le synchronisme avec le cycle planétaire. En ralentissant le temps pour enlever 1 seconde sur 20 heures, ceci donne pour 20s: 20s * (20h * 60m/h * 60s/m) / (20h * 60m/h * 60s/m + 1s) = 19.999722226s.*

c) Lors du travail pratique 2, un répartiteur envoyait des tâches à plusieurs serveurs. Il pouvait arriver que certains serveurs soient tués au beau milieu de l'exécution d'une tâche. Comment est-ce que le répartiteur pouvait s'apercevoir d'une telle panne et comment pouvait-il réagir afin de tolérer cette panne? **(1 point)**

Un serveur ainsi arrêté, tout comme un réseau qui tombe en panne, n'envoie pas de message pour prévenir que la réponse attendue n'arrivera pas. Il faut que le coordonnateur se mette une minuterie à chaque requête envoyée, de manière à conclure à une panne si la réponse n'arrive pas avant un temps raisonnable. A ce moment, il peut conclure que le serveur qui n'a pas encore répondu est en panne et redonner cette tâche à un serveur fonctionnel. Le coordonnateur pourrait aussi communiquer de temps en temps avec les différents serveurs pour suivre les progrès et découvrir plus tôt si un serveur tombe en panne.

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? Justifiez. **(2 points)**

T: Début
T: Read(a)
T: Write(b,1)
U: Début
U: Read(a)
U: Write(c,2)
V: Début
T: Compléter
U: Read(b)

V: Read(a)
V: Read(b)
V: Read(c)
V: Write(c,3)
V: Compléter
U: Compléter

Pour la validation en reculant, au moment où T termine, il ne peut y avoir de problème et T est accepté. Pour compléter V, on examine ce qui est lu (a, b, c) versus ce qui a été écrit par T (b) et V ne peut compléter. Rendu à compléter U, ce qui est lu (a, b) intersecte avec T (b) et U doit aussi être abandonné. Pour la validation en avançant, au moment où T termine, ce que T a écrit (b) n'intersecte pas avec ce que U a lu (a) ni V (rien encore) et T peut compléter. Au moment de compléter V, les écritures (c) n'intersectent pas avec les lectures de U (a,b) et V peut compléter. La dernière transaction, U, peut nécessairement compléter.

b) Une transaction répartie implique un coordonnateur, un client et plusieurs serveurs. Dans un premier cas, i), le client a ouvert une transaction (e.g., acheter des billets de spectacle pour sa famille), spécifié plusieurs opérations pour la transaction et demandé de compléter la transaction. Il est maintenant en attente de la confirmation que la transaction est acceptée. Il attend un certain temps et ne reçoit toujours pas de réponse. Peut-il en conclure que la transaction est effectuée ou non? Peut-il demander à ce point d'annuler la transation? Que peut-il faire? Dans un second cas, ii), le coordonnateur d'une transaction reçoit la demande du client de compléter la transaction et envoie à chaque serveur impliqué une demande à savoir s'ils sont prêts à commettre la transaction. Il reçoit des réponses (affirmatives) de tous les serveurs sauf 1 (dont le réseau est possiblement en panne). Peut-il relancer le serveur qui ne répond pas? Peut-il attendre indéfiniment? Que peut-il faire? **(2 points)**

Dans le premier cas, i), le client ne peut rien conclure. Peut-être que le coordonnateur ou le réseau est tombé avant que sa demande de compléter ne soit arrivée, ou après que la transaction soit complétée mais avant que la confirmation ne lui soit parvenue. Sa seule possibilité est de recontacter le coordonnateur jusqu'à ce qu'il obtienne une réponse, une fois que celui-ci aura recouvré ses données. Dans le second cas, ii), le coordonnateur peut encore sans problème abandonner la transaction à ce point-ci, puisqu'aucun message de confirmation n'a encore été envoyé. Peut-être que le serveur qui ne répond pas n'a pas reçu sa demande s'il est prêt, ou peut-être qu'une réponse positive a été envoyée mais ne lui est pas parvenue. Il peut relancer le serveur mais ne peut attendre indéfiniment, puisque pendant ce temps il fait aussi attendre le client et utilise des ressources sur les autres serveurs. Si la réponse tarde trop, il annule la transaction tout simplement et envoie des messages à cet effet.

c) Sachant que vous avez complété le travail pratique 3 et êtes un expert avec OpenStack et Heat, on vous consulte car le gabarit suivant comporte un problème. Dites quelle ligne devrait être corrigée et comment. **(1 point)**

```
resources:
   web_nodes:
       type: OS::Heat::ResourceGroup
```

```
        properties:
            count: 2
            resource_def:
            #Définition des serveur.

    pool:
        type: OS::Neutron::Pool
        #Définition du Pool

    lbalancer:
        type: OS::Neutron::LoadBalancer
        properties:
            protocol_port: 8000
            members: {get_resource: web_nodes}
            pool_id: {get_resource: pool}
```

La ligne "members: get_resource: web_nodes" devrait plutôt être "members: get_attr: [web_nodes, refs]".

Question 4 (5 points)

a) Le système informatique d'une centrale solaire commande l'inclinaison des panneaux solaires et doit offrir une très grande fiabilité. Un arrêt peut causer une diminution importante de la production électrique voire même concentrer les rayons du soleil au mauvais endroit et faire griller les maisons voisines. Le système est composé de 3 ordinateurs en redondance (Triple Modular Redundancy) qui reçoivent les mêmes entrées et dont les sorties sont passées par un voteur. Le voteur est très fiable et produit une sortie tant que deux des trois ordinateurs sont en bon ordre (donnent le même résultat). Chaque ordinateur a une probabilité de défaillance (donner un mauvais résultat) de 0.1. Quelle est la probabilité de défaillance de ce système de trois ordinateurs redondants? On vous propose de changer le voteur pour que, s'il obtient 3 réponses différentes, il propagera en sortie la sortie d'un des ordinateurs pré-sélectionné (e.g., l'ordinateur #1, car il est plus neuf). Est-ce que cela augmente la disponibilité du système? Est-ce qu'il y a des inconvénients à faire cela? **(2 points)**

Dans le premier cas, le système sera opérationnel tant qu'il y a au moins 2 ordinateurs fonctionnels (2 ou 3 sur 3). La probabilité d'avoir les 3 ordinateurs fonctionnels est de $0.9^3 = 0.729$. *Celle d'avoir 2 ordinateurs sur 3 est de* $3!/(2! \times 1!)0.9^2 \times 0.1 = 0.243$. *Le total est donc de* $0.729 + 0.243 = 0.972$. *Une autre manière de le calculer est que le tout est fonctionnel lorsqu'il y a 0 ou 1 ordinateur en panne, ce qui donne* $0.9^3 + 3!/(1! \times 2!)0.1 \times 0.9^2 = 0.972$. *Dans le cas où on modifie le voteur pour prendre une chance et utiliser l'ordinateur #1 en dernier recours si les 3 réponses sont différentes, on ajoute un fonctionnement correct dans le cas où l'ordinateur #1 est fonctionnel et les deux autres sont incorrects. La probabilité de ceci est de* $0.9 \times 0.1 \times 0.1 = 0.009$. *Nous n'utilisons pas le terme* $n!/(k! \times (n-k)!)$ *car nous ne voulons pas toutes les combinaisons de 1 ordinateur fonctionnel et 2 non fonctionnels mais seulement le cas où l'ordinateur #1 est fonctionnel et les deux autres non fonctionnels. La disponibilité dans*

ce cas devient $0.972 + 0.009 = 0.981$. *Cependant, dans le cas où les 3 ordinateurs donnent une réponse différente, il peut être plus intéressant de déclarer le système défectueux et effectuer un arrêt sécuritaire (e.g., pointer les miroirs vers le ciel ou vers le sol) plutôt que d'aller chercher quelques minces chances d'avoir le système #1 qui est peut-être fonctionnel.*

b) Un dicton dit que les configurations de serveurs actif-passif sont plus simples à gérer mais moins performantes. Sur le système de fichiers CODA, chaque fichier est stocké sur plusieurs serveurs (typiquement 3). Lorsqu'un client veut lire un fichier, il lui suffit de le lire sur un seul serveur. Pour écrire un fichier, il doit l'écrire sur tous les serveurs, ou du moins ceux qui sont rejoignables puisque ce système est tolérant aux pannes. Est-ce que CODA utilise une configuration actif-passif ou actif-actif? Est-ce que le dicton s'applique ici? Donnez un exemple concret de performance différente et de cas complexe à gérer par rapport à un serveur seul? **(2 points)**

Il s'agit d'un système actif-actif puisque tous les serveurs sont actifs de manière symétrique. Avec 3 serveurs en redondance, les lectures peuvent être réparties entre les serveurs et on peut donc servir 3 fois plus de requêtes, ce qui en fait un système 3 fois plus performant qu'un serveur seul ou actif-passif. Si le réseau est partitionné et que deux clients modifient en même temps le même fichier sur deux serveurs différents (chacun dans sa partition et ne réussissant pas à passer la modification aux autres serveurs dans l'autre partition), on se retrouvera avec des nouvelles versions incohérentes et il faudra résoudre le conflit (fusionner les modifications ou annoncer à un des clients que sa mise à jour sera abandonnée). Le dicton s'applique puisque les serveurs actif-actif sont plus performants mais moins faciles à gérer que les serveurs actif-passif.

c) Dans le processus de développement classique, chaque développeur d'un gros projet extrait du système de gestion de code source (e.g., SVN ou Git) les fichiers du programme et en obtient une copie sur son poste. Ensuite, celui-ci peut modifier quelques fichiers et effectuer la compilation sur son poste. Une compagnie qui dispose d'une grappe de serveurs de compilation pourrait permettre aux développeurs d'envoyer la demande de compilation sur ces serveurs (envoyer les fichiers source et recevoir les fichiers compilés). Avec le Google Build System, la situation est différente. Expliquez ces différences et en quoi permettent-elles de sauver du temps de compilation. **(1 point)**

Avec le Google Build System, les fichiers résident déjà sur les serveurs et les fichiers compilés seront aussi créés directement sur les serveurs, ce qui sauve beaucoup de transferts de fichiers potentiellement volumineux. Le principal attrait du Google Build System est sa cache. Si un fichier a déjà été compilé par un développeur, toute compilation subséquente avec le même fichier et les mêmes options est évitée car on utilise le fichier déjà compilé. Plus encore, si le fichier compilé n'a pas changé par rapport à la version antérieure (e.g., seulement des commentaires ont été édités dans le code source), les étapes subséquentes qui en dépendent, comme l'édition de lien, peuvent être évitées aussi.

Le professeur: Michel Dagenais