ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

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

CORRIGÉ DE L'EXAMEN FINAL

DATE: Mercredi le 25 avril 2018

HEURE: 13h30 à 16h00

DUREE: 2H30

NOTE: Aucune documentation permise sauf un aide-memoire, 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 serveur DNS sert des requêtes de manière récursive. Lorsqu'une requête est reçue d'un client, elle requiert 0.5ms de CPU. Dans 30% des cas, la réponse n'est pas en mémoire centrale et un accès au disque est requis en plus, ce qui ajoute 15ms en attente du disque. Finalement, pour 1/3 des cas où un accès disque est requis, la réponse n'est pas sur disque non plus et un accès récursif est requis, ce qui ajoute 200ms en moyenne d'attente après la requête par réseau au serveur de plus haut niveau. Si ce serveur exécute toutes les requêtes séquentiellement avec un seul thread, combien de requêtes par seconde pourra-t-il traiter au maximum? Si ce serveur contient un CPU à 4 coeurs et 8 disques, et que de nombreux threads sont disponibles pour servir les requêtes en parallèle, combien de requêtes pourra-t-il traiter au maximum? Quel est le nombre de threads requis pour atteindre ce maximum? On suppose que la charge entre les disques est bien répartie, tout comme la charge entre les coeurs. **(2 points)**

Pour une requête moyenne, on a 0.5ms de CPU, 0.3 x 15ms = 4.5ms d'attente après le disque et 1/3 x 0.3 x 200ms = 20ms d'attente après le serveur de plus haut niveau, pour un total de 25ms. Le serveur pourra donc traiter séquentiellement 1000ms/s / 25ms/r = 40 requêtes / s. Chaque coeur du CPU peut traiter 1000ms/s / 0.5ms/r = 2000r/s, soit 8000r/s pour les 4 coeurs. Chaque disque peut traiter 1000ms/s / 4.5ms/r = 222.22r/s, soit 1777.77r/s pour les 8 disques. Avec 1777.77r/s, chaque requête occupant un thread pendant 25ms ou .025s, ceci donne un total de 1777.77r x .025s = 44.44 secondes de temps de thread à chaque seconde, ce qui requiert au minimum 45 threads en parallèle.

b) Un client interroge 10 serveurs DNS redondants et retient la réponse la plus fréquente. Combien de serveurs en panne ce client pourra-t-il tolérer avant de ne plus avoir la bonne réponse si les pannes sont par omission? Par réponse aléatoire? Par réponse byzantine? **(2 points)**

Par omission, le client peut fonctionner même si 1 seul serveur est fonctionnel (et 9 sont en panne). Par réponse aléatoire, le client peut avec un peu de chance fonctionner si au moins 2 serveurs sont fonctionnels (et 8 en panne). En présence de réponses byzantines, le client doit avoir plus de la moitié des serveurs fonctionnels, soit 6 (et un maximum de 4 serveurs en panne).

c) Comment se comparent les services de nom (Directory service) par LDAP et X.500? Lequel est le plus simple? Le plus utilisé? **(1 point)**

Le système X.500 était très compliqué, en ayant une architecture se voulant capable de s'adapter à un grand nombre de situations. Il y a eu très peu de tels serveurs en opération et il y en a encore moins aujourd'hui. Le système LDAP s'inspire de X.500 tout en conservant une organisation beaucoup plus simple. C'est le service de répertoire le plus utilisé, sauf dans les environnements qui utilisent exclusivement le système d'exploitation Windows de Microsoft où Active Directory est généralement utilisé.

Question 2 (5 points)

a) Un client interroge un serveur pour synchroniser son horloge avec la méthode de Christian. Il effectue 3 requêtes en espérant avoir plus de précision et obtient les résultats suivants. Envoi de la requête 1 à 13h00m00.000 et réception à 13h00m01.000 de la réponse disant que

l'heure du serveur était 13h00m05.000. Envoi de la requête 2 à 13h00m02.000 et réception à 13h00m03.000 de la réponse disant que l'heure du serveur était 13h00m07.000. Envoi de la requête 3 à 13h00m04.000 et réception à 13h00m04.010 de la réponse disant que l'heure du serveur était 13h00m09.000. A l'aide de ces données, calculez la valeur de décalage à appliquer à l'horloge du client qui minimise l'incertitude sur le décalage, et donnez cette incertitude. **(2 points)**

L'incertitude est déterminée par le temps d'aller-retour de la requête. La requête 3 est celle avec le temps d'aller-retour le plus court, 13h00m.04.010 - 13h00m04.000 = 0.010. Le nouveau temps à appliquer au client est donc de 13h00m09.000 (heure du serveur) + 0.010 / 2 (aller-retour / 2) = 13h00m09.005 +/- .005. Le décalage à appliquer à ce moment sur l'heure du client qui était de 13h00m04.010 est donc de 13h00m09.005 - 13h00m04.010 = 4.995s +/- .005.

b) Dans le cadre du travail pratique 2, vous avez implémenté un service réparti de calcul. Comment pouviez-vous tolérer une panne par omission de réponse? Comment pouviez-vous tolérer une panne de mauvaise réponse? Pour quel type de panne ou quelle combinaison de pannes simultanées est-ce que votre système ne réussirait plus à fonctionner correctement? **(2 points)**

Les pannes par omission sont tolérées grâce à un temps d'expiration. Si ce temps est dépassé, on suppose que le serveur est en panne et on soumet la même requête à un autre serveur. Ceci ajoute un peu de latence mais n'empêchera pas le calcul d'être finalement obtenu. Si le système n'est pas sécurisé et que des réponses incorrectes peuvent être reçues, le répartiteur considère qu'une réponse est correcte seulement si deux serveurs donnent la même réponse. Autrement, il faut resoumettre la requête à d'autres serveurs jusqu'à obtention de deux réponses identiques. Si deux serveurs malicieux s'entendent pour retourner la même mauvaise réponse, le système ne fonctionnera pas correctement. Le système comporte aussi plusieurs composantes qui ne sont pas redondantes, par exemple le repartiteur ou le réseau.

c) Nommez 3 algorithmes d'élection différents et expliquez brièvement les avantages et désavantages de chacun (simplicité, efficacité, robustesse). **(1 point)**

L'élection en anneau est simple mais exige que tous les participants soient fonctionnels, ce qui est peu robuste. Le délai associé à faire le tour de l'anneau à 3 reprises est aussi un peu long. L'élection hiérarchique est relativement simple et est assez efficace dans le cas courant où un ordinateur de haut niveau de priorité peut rapidement se déclarer le plus fort, et conséquemment élu. Cependant, l'élection hiérarchique ne fonctionne pas bien si les délais sont trop longs ou si le réseau est partitionné. Le consensus de Paxos est l'algorithme le plus complexe. Il est très robuste et fonctionne (ne donne pas de résultat incorrect) même en cas de partition de réseau. Dans le cas simple, il est raisonnablement efficace.

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
U: Read(w)
T: Read(x)
U: Read(z)
T: Write(x,10)
T: Compléter
U: Read(x)
V: Read(w)
U: Read(y)
U: Write(y,22)
U: Compléter
V: Read(z)
V: Read(y)
V: Write(z,9)
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 (x), alors que U a lu (w, z, x, y). Ainsi U ne peut être validée. Rendu à V, seulement T a complété en écrivant (x), alors que V a lu (w, z, y). V peut compléter. En avançant, la transaction T regarde ce qu'elle a écrit (x) versus ce que U et V ont lu (w,z). T peut compléter. Rendu à compléter U, ses écritures sont (y) alors que les lectures de V jusqu'ici sont (w). Ainsi, U peut compléter. V, qui est la dernière du groupe, peut naturellement compléter.

b) Une transaction répartie T effectue les opérations suivantes. Les variables a, b et c sont sur le serveur X, les variables d, e et f sur le serveur Y, et les variables g, h et i sur le serveur Z. Le système utilise un protocole de fin de transaction atomique à deux phases. Que retrouvera-t-on dans le journal de chacun des serveurs, X, Y et Z? Quelles requêtes et réponses est-ce que le coordonnateur échange avec les trois serveurs pour le protocole de fin de transaction? Chaque requête ou réponse à chacun des serveurs pour le protocole de fin de transaction arrive à quel moment par rapport aux écritures dans le journal de ce serveur? **(2 points)**

T: Début; Read(a); Read(b); Read(c); Read(d); Read(f); Read(h); Read(i); Write(c,0);
Write(b,1); Write(d,2); Read(e); Write(f,3); Write(h,4); Write(i,5); Read(e); Write(c,6);
Write(e,7);

Dans le journal de X, on aura: P0: Write(c,0); P1: Write(b,1); P2: Write(c,6); P3: Préparer T(P0, P1, P2); P4: Compléter T,P3. Selon les optimisations faites, il est possible que l'entrée P0 soit absente ou qu'elle ne soit pas listée dans la préparation P3, étant donné que cette valeur est écrasée par une autre écriture de c dans la même transaction. Dans le journal de Y on aura: P0: Write(d,2); P1: Write(f,3); P2: Write(e,7); P3: Préparer T(P0, P1, P2); P4: Compléter T,P3. Dans le journal de Z, on aura: P0: Write(h,4); P1: Write(i,5); P2: Préparer T(P0, P1); P3: Compléter T,P2. Chaque serveur, au moment où on lui demande s'il est prêt pour la transaction T, écrit toutes les informations à propos de T, incluant l'entrée pour "Préparer", dans son journal.

Il peut alors répondre qu'il est prêt et accepte la transaction. Le coordonnateur, ayant reçu l'acceptation de chaque serveur, peut alors envoyer la confirmation à chaque serveur qui pourra en conséquence ajouter l'entrée "Compléter".

c) Quelle est la différence entre une analyse économique classique et une analyse complète du cycle de vie? **(1 point)**

Dans une analyse économique classique, seule la rentabilité est évaluée en utilisant les chiffres des coûts directs. Plusieurs aspects plus difficiles à mesurer, dont l'impact sur l'environnement ou l'appauvrissement des ressources, ne sont pas tenus en compte autrement que via des taxes directes qui seraient imposées à cet effet. Avec l'analyse du cycle de vie, toutes les phases sont considérées, de la construction et la fabrication des intrants à l'opération et au démantèlement éventuel avec le recyclage ou la mise aux rebuts. Certains éléments comme l'impact sur l'environnement et la santé humaine sont quantifiés, afin de comparer entre eux différents scénarios. Ces impacts ne sont pas nécessairement convertis en valeur monétaire, puisque cette conversion n'est pas facile à réaliser et dépend de nombreux facteurs et hypothèses. L'analyse complète du cycle de vie est donc beaucoup plus complète que l'analyse économique classique.

Question 4 (5 points)

a) Un laboratoire informatique contient 26 postes de travail, reliés à 2 serveurs redondants. Chaque serveur contient à son tour deux disques redondants. Il est prévu pour accueillir 25 équipes. Il faut donc qu'au moins 25 postes de travail soient opérationnels. Il faut aussi qu'au moins 1 des 2 serveurs soit opérationnel. Pour qu'un serveur soit opérationnel, il faut que son électronique soit opérationnelle et qu'au moins 1 de ses 2 disques le soit. La probabilité de fonctionner à un moment donné est de 0.95 pour un poste de travail, 0.8 pour l'électronique d'un serveur et 0.7 pour un disque. Quelle est la probabilité que le laboratoire soit opérationnel pour accueillir 25 équipes? **(2 points)**

Pour que le service soit disponible à 25 équipes, il faut que 25 ou 26 postes soient opérationnels et au moins 1 serveur. Pour qu'un serveur soit opérationnel, il faut que son électronique fonctionne et au moins 1 disque. La probabilité que les 26 postes soient fonctionnels est de $0.95^{26} = 0.26352$. *La probabilité qu'exactement 25 sur 26 postes soient fonctionnels est de* $26!/((26-25)! \times 25!) \times 0.95^{25} \times (1-0.95)^{(26-25)} = 0.3606$. *Le total est* $0.26352 + 0.3606 = 0.62412$. *Les disques d'un serveur fonctionnent sauf si les 2 sont en panne, ce qui donne* $1-(1-0.7)^2 = 0.91$. *Le serveur fonctionne si son électronique et ses disques fonctionnent, soit* $0.8 \times 0.91 = 0.728$. *Les serveurs fonctionneront sauf si les 2 sont en panne, soit* $1-(1-0.728)^2 = 0.926$. *Le laboratoire sera opérationnel pour 25 équipes si les postes et les serveurs le sont, soit* $0.62412 \times 0.926 = 0.57794$.

b) Pour l'établissement d'un nouveau centre de données, on doit effectuer une analyse complète qui inclut l'analyse du cycle de vie. Quelles sont les différentes phases à considérer dans une analyse de cycle de vie? Quels sont les quatre différents types d'impact sur l'environnement à considérer? Quels sont les éléments d'un centre de données qui ont généralement le plus d'impact et pendant quelle phase? Est-ce que la source d'énergie et le climat ont un effet sur l'impact d'un centre de données? Expliquez. **(2 points)**

Il faut tenir en compte toutes les phases du projet, de sa construction et la fabrication des équipements qui s'y trouvent à l'opération et à la fin de vie. La fin de vie inclut la démolition ou la conversion du bâtiment, et le recyclage ou la mise aux rebuts des matériaux et équipements. Pour toutes les activités qui se retrouvent dans ces différentes phases, il faut examiner l'impact sur la santé humaine, sur l'écologie, sur les changements climatiques et sur l'appauvrissement des ressources. Pour un centre de données typique, une grande partie de l'impact est reliée à la consommation d'énergie pendant l'opération pour l'équipement informatique et pour la climatisation. Un autre impact non négligeable est la fabrication des équipement informatiques et électriques, en particulier le raffinage de l'or et du cuivre requis pour ces équipements. Etant donné l'importance de la consommation electrique, une partie étant reliée à la climatisation, une source d'énergie propre, et un climat froid qui ne requiert pas de climatisation, peuvent diminuer considérablement l'impact négatif d'un centre de données.

c) Dans le gabarit Heat du service Web avec répartition de charge du travail pratique 3, vous avez défini des propriétés permettant d'instancier plusieurs ressources. Expliquez brièvement le rôle de chacune des ressources suivantes dans le travail pratique, OS::Neutron::HealthMonitor, OS::Neutron::Pool, OS::Neutron::LoadBalancer **(1 point)**

Pour répartir la charge, il faut un groupe de machines, OS::Neutron::Pool (un bassin de machines virtuelles identiques pour servir des requêtes), un moniteur qui vérifie les noeuds fonctionnels, OS::Neutron::HealthMonitor (un moniteur qui interroge les serveurs à intervalle régulier pour détecter ceux qui ne sont pas fonctionnels). Finalement on retrouve le répartiteur de charge, OS::Neutron::LoadBalancer (en sachant quel serveur est fonctionnel et en mesurant le temps de réponse, les requêtes sont réparties de manière à partager équitablement la charge et minimiser le temps de réponse).

Le professeur: Michel Dagenais