ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF4402: Systèmes répartis sur l'Internet (Automne 2012)
3 crédits (3-1.5-4.5)

CORRIGÉ DE L'EXAMEN FINAL

DATE: Dimanche le 16 décembre 2012

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) Un client envoie un message au serveur pour lui demander l'heure exacte. Ce message est envoyé à 10h00m00.000s et la réponse reçue à 10h00m00.542s, heure du client. La réponse indique que l'heure exacte du serveur au moment où il a répondu était de 10h00m05.041s. En utilisant l'algorithme de Christian, calculez le décalage à appliquer à l'heure du client et donnez l'intervalle d'incertitude sur cette valeur de décalage? Si le serveur avait fourni l'information additionnelle suivante, réception de la demande à 10h00m04.539 et envoi de la réponse à 10h00m05.041s, en utilisant l'algorithme de NTP qui tire parti de cette information, que deviendrait le calcul de décalage et l'intervalle d'incertitude associé? **(2 points)**

Avec la méthode de Christian, il suffit de calculer le délai de réponse vu du client, .542s, et d'assumer que cela représente le délai symétrique associé au réseau (.271s dans chaque direction). La nouvelle heure est donc 10h00m05.041 (heure du serveur) + .271s (délai réseau), duquel on soustrait 10h00m00.542s (heure de réception du client) pour calculer le décalage du client. Le décalage est donc de 4.77s +/- .271s.

Avec la méthode NTP, nous possédons de l'information supplémentaire. Le calcul donne 4.519s +/- .02s, ce qui est cohérent avec le premier calcul mais beaucoup plus précis.

a = 10h00m04.539s - 10h00mm00.000s = 00m04.539s
b = 10h00m05.041s - 10h00m00.542s = 00m04.499s
Ajustement = (a+b)/2 = (4.539s + 4.499s)/2 = 4.519 s
Précision = (a-b)/2 = (4.539s - 4.499s)/2 = 0.02s

b) Afin de détecter un interblocage réparti dans un système de transactions réparti, il faudrait idéalement prendre un cliché cohérent de tous les liens de dépendance entre les verrous et les transactions. Ceci n'est géneralement pas fait, les dépendances de chaque serveur sont prises en séquence avec un délai faible plutôt que d'essayer d'avoir simultanément tout l'état de tous les serveurs. Pourquoi? Quelles sont les conséquences d'avoir un graphe de dépendance global construit à partir des informations non simultanées (cohérentes) des différents serveurs? Donnez un exemple concret de situation problématique qui peut se présenter. **(2 points)**

Obtenir un cliché cohérent de l'état global serait très coûteux et pourrait demander de tout arrêter les processus impliqués le temps de récolter toute l'information. C'est pour cette raison qu'une approximation est faite en lisant toutes les informations rapidement mais sans garantie de cohérence.

Une première situation qui peut arriver est qu'un interblocage se constitue pendant le processus de collecte de donnée. Juste après que ses informations aient été lues, le premier processus demanderait un verrou détenu par le dernier processus lu. Ceci ne pose pas vraiment de problème car l'interblocage demeurera et sera facilement détecté au prochain tour. L'autre possibilité est que le premier processus relâche un verrou juste après que ses informations aient été lues. Ceci peut paraître difficile a priori qu'un processus qui semble impliqué dans un interblocage puisse relâcher un verrou; un processus ne relâche normalement pas un verrou avant de compléter et cela serait difficile s'il y a des dépendances circulaires. Cepen-

dant, il peut arriver que la transaction du premier processus soit annulée avant que d'autres liens de dépendances qui formeraient un cycle ne s'ajoutent.

c) Dans le travail pratique 2, chaque client numérotait ses demandes afin de permettre aux serveurs de maintenir un ordonnancement cohérent entre les requêtes (pour la même donnée d'un même client). Discutez brièvement des divers problèmes de sécurité de cette application et de solutions possibles. Par exemple, un client malicieux pourrait-il bloquer le système en envoyant par exprès des requêtes avec des numéros d'ordre manquants (e.g., requête 5, ensuite 7...)? **(1 point)**

Il faudrait premièrement être capable d'authentifier les clients auprès des serveurs. Autrement, n'importe qui peut se faire passer pour n'importe quel client et modifier n'importe quelle donnée. Puisque chaque serveur n'ordonnance que les requêtes par client, un client qui saute un numéro ne pourra voir ses requêtes servies tant que le numéro manquant ne sera pas vu par le serveur. Cela n'empêchera toutefois pas les autres clients d'être servis. Le client malicieux pourrait cependant remplir la queue du serveur de requêtes que le serveur retient. Si le serveur a été bien pensé, il aura une taille limite pour la queue de chaque client après laquelle il ignore toute nouvelle requête.

S'il n'y a pas d'authentification, le client malicieux pourrait se faire passer pour un autre client et envoyer des requêtes avec des numéros plus élevés (n+2, n+3...) que la requête courante (n) du client qu'il personnifie. Il pourrait ainsi remplir la queue de cet autre client. Le client réel va tout de même éventuellement envoyer la prochaine requête en séquence (n+1). Le serveur devrait idéalement accepter cette requête même si sa queue est pleine, puisque cela permettra de le débloquer. Ensuite, si le client réel envoie d'autres requêtes (n+2) le serveur sera perplexe car il croit avoir déjà reçu cette requête et devrait générer un message d'erreur. S'il refuse la seconde requête au même numéro, le client malicieux aura réussi à empêcher le client réel de se faire servir.

Question 2 (5 points)

Trois transactions, T, U et V, s'exécutent concurremment. Le séquencement des opérations est le suivant:

T: Début
U: Début
V: Début
T: Read(x)
U: Read(z)
T: Write(x,1)
T: Compléter
U: Read(x)
V: Read(w)
U: Read(y)
U: Write(y,2)

U: Compléter
V: Read(x)
V: Read(y)
V: Write(z,3)
V: Compléter

a) Lesquelles des transactions T, U et V pourraient être validées si une validation en reculant était utilisée pour vérifier la cohérence des transactions? Justifiez. **(1 point)**

La première transaction, T, est nécessairement validée. Pour U, il faut vérifier l'intersection entre ce que T a écrit (x) et ce que U a lu (x, y, z) ou écrit. Il y a intersection et il faut annuler U. Pour V, il ne reste plus que T à vérifier. T a écrit x alors que V a lu w, x et y. Là encore, l'intersection (x) fait qu'il faut annuler V.

b) Lesquelles des transactions T, U et V pourraient être validées si une validation en avançant était utilisée pour vérifier la cohérence des transactions? Justifiez. **(1 point)**

Au moment de compléter T, il faut vérifier ce qui est écrit par T (x) versus ce que U a lu (z, car à ce moment U n'a pas encore lu x). Il n'y a pas de conflit et T est validé. La transaction U écrit (y) alors que V a lu (w), elle peut donc compléter. Rendu à compléter V, il ne reste plus d'autre transaction et donc pas de problème.

c) Lesquelles des transactions T, U et V pourraient être validées si une validation par compteur de temps était utilisée pour vérifier la cohérence des transactions? Justifiez. Les écritures sont faites dans des valeurs tentatives jusqu'à ce que la transaction associée ne soit complétée, les lectures sont faites dans les valeurs tentatives. **(1 point)**

La transaction T se voit assigner le temps t1, U t2 et V t3 ($t0 < t1 < t2 < t3$).

T: Début, t1
U: Début, t2
V: Début, t3
T: Read(x), écrit en t0, lu en t1
U: Read(z), écrit en t0, lu en t2
T: Write(x,1), version tentative pour t1
T: Compléter, écrire x en t1, pas de conflit, validé
U: Read(x), écrit en t1, lu en t2
V: Read(w), écrit en t0, lu en t3
U: Read(y), écrit en t0, lu en t2
U: Write(y,2), version tentative pour t2
U: Compléter, écrire y en t2, pas de conflit, validé
V: Read(x), écrit en t1, lu en t3
V: Read(y), écrit en t2, lu en t3
V: Write(z,3), version tentative pour t3
V: Compléter, écrire z en t3, pas de conflit, validé

d) Vous devez concevoir un système d'achat de billets de spectacles où le client peut choisir plusieurs sièges et ensuite confirmer son choix et son achat par son paiement. De nombreux clients peuvent utiliser le système d'achat en même temps pour un même spectacle. Vous hésitez entre un contrôle optimiste de la concurrence et un contrôle strict de la concurrence. Quelles seront les conséquences de ce choix sur l'utilisation du site par un client (performance, déroulement de la procédure d'achat...)? Donnez un exemple concret de problème qui pourrait se poser. **(2 points)**

Un contrôle strict fait qu'il faut bloquer au fur et à mesure les sièges tentativement réservés par un client. Ceci demande de prendre un verrou à chaque nouveau siège ajouté au panier d'achat du client. Ceci est assez coûteux car la prise de verrou est une opération atomique sur un serveur central. De plus, on ne peut utiliser un contrôle strict sans mettre des restrictions importantes, faute de quoi le système sera très vulnérable aux abus. Par exemple, si un client réserve tentativement tous les billets de la salle, il pourrait bloquer tous les autres acheteurs. Pour cette raison, il faut imposer des limites possiblement sur le nombre de billets mais surtout sur le délai maximal pendant lequel une réservation tentative est conservée. Ainsi, un client peut se faire avertir qu'il a 2 minutes pour procéder au paiement, à defaut de quoi les sièges qu'il a placés dans son panier d'achat seront relâchés pour les autres clients.

Avec un contrôle optimiste, le client peut prendre tout le temps qu'il veut mais il peut éventuellement se faire avertir, en temps réel ou au moment de procéder au paiement, que les sièges qu'il avait identifiés viennent d'être achetés par un client plus rapide que lui. Cette procédure est plus efficace sur le serveur, en l'absence d'acquisition de verrous au fur et à mesure de la transaction. Par contre, le client pourrait être fort indisposé de de faire dire au dernier moment que quelqu'un a pris les sièges convoités avant lui.

Question 3 (5 points)

a) Un grand mélomane s'est procuré un système redondant pour jouer de la musique dans son salon. Il possède donc deux paires de haut-parleurs (HP_A et HP_B), chacune connectée à un ordinateur client (CL_A pour HP_A et CL_B pour HP_B) relié à un réseau sans fil différent ($WIFI_A$ pour CL_A et $WIFI_B$ pour CL_B). Il possède en outre deux serveurs redondants (SRV_1 et SRV_2) pour sa musique, chacun avec 4 disques en RAID 5 et deux cartes réseau sans fil. Chaque client peut aussi bien se connecter sur SRV_1 ou sur SRV_2. Les unités RAID 5 permettent de tolérer la panne d'un disque sur les quatre. Les probabilités d'être en panne à un moment donné sont de 0.00001 pour une paire de haut-parleurs, 0.00002 pour un client, 0.00003 pour un réseau sans fil, 0.00004 pour un serveur (électronique et logiciels mais sans les disques) et de .001 pour un disque. Quelle est la probabilité que ce système redondant soit non-disponible, empêchant d'écouter de la musique avec une paire de haut-parleurs à partir d'un serveur? **(2 points)**

Le RAID 5 sera disponible si les 4 disques sont disponibles ou si un seul disque est en panne (n'importe lequel des 4) et 3 disques fonctionnels. La probabilité est donc de $P_{RAID5} = (1 - 0.001)^4 + 4 \times (1 - 0.001)^3 \times 0.001 = 0.999994008$. *Un client musique est disponible si les haut-parleurs, le réseau et le client sont disponibles soit* $P_{CL} = (1 - .00001) \times (1 - .00002) \times (1 - .00003) = 0.999940001$. *Un serveur est disponible si le serveur et le RAID5*

sont disponibles $P_{SRV} = (1 - .00004) * P_{RAID5} = 0.999954008$. *Le service est disponible si au moins un des serveurs est disponible* $P_{Serveurs} = 1 - (1 - P_{SRV})^2) = 0.999999998$ *et au moins un des clients est disponible* $P_{Clients} = 1 - (1 - P_{CL})^2 = 0.999999996$. *Le service sera non-disponible sauf si un client et un serveur sont disponibles soit* $P = 1 - (P_{Serveurs} \times P_{Clients}) = 0.000000006$

b) Une entreprise avec 15 sites veut offrir des forums à ses employés, un peu comme le système Usenet qui utilisait le protocole NNTP. Sur chaque site, un serveur permettra aux employés de consulter une copie des messages et de contribuer de nouveaux messages sur les forums. Afin de propager chaque message sur chaque site, deux organisations sont envisagées. La première organisation se base sur un serveur central dans un des 15 sites, la maison mère, avec lequel chacun des serveurs sur les 14 autres sites communiqueront pour obtenir les nouveaux messages produits sur les autres sites et contribuer les nouveaux messages générés sur ce site. La seconde organisation se base sur un arbre avec 8 sites classés niveau 3, 4 sites classés niveau 2, 2 sites classés niveau 1 et un serveur racine de niveau 0 sur le site de la maison mère. Dans cette structure arborescente, chaque serveur de niveau n est alimenté par un parent de niveau n - 1 et alimente deux enfants de niveau n + 1; le niveau 4 n'a pas d'enfant et le niveau 0 pas de parent. Chaque serveur obtient de son parent les nouveaux messages produits dans les autres branches et lui contribue les nouveaux messages générés dans sa branche. Pour chaque nouveau message créé, comparez le nombre de messages échangés entre les serveurs pour chacune de ces deux organisations. **(2 points)**

Dans le premier scénario, lorsqu'un message est créé sur un site satellite, il faut 1 message pour l'envoyer au serveur central et 13 messages pour envoyer le message aux 13 autres sites satellites, pour un total de 14 messages (1 envoi de l'origine, 1 réception et 13 envois du central, 1 réception de chaque autre satellite). Avec le second scénario, chaque site sauf celui qui génère le nouveau message reçoit un message, pour un total de 14 aussi. Cependant, la répartition des messages est différente. Chaque noeud interne reçoit 1 message et en envoie 2 alors que les feuilles n'en reçoivent que 1 (sauf celui qui génère le nouveau message qui en envoie 1) et la racine en reçoit 1 et en envoie 1. Dans ce second scénario, la charge est beaucoup mieux répartie et risque de beaucoup mieux tolérer la mise à l'échelle. Il est peu probable que le multicast soit bien supporté sur un tel réseau externe entre différents sites. Avec le multicast, la première solution pourrait se faire avec 2 messages alors que la seconde demanderait entre 7 et 10 messsages dépendant de l'origine du message dans l'arbre.

c) Dans les bases de données conventionnelles, les modifications associées à une transaction sont premièrement écrites dans le journal, ensuite (si la transaction est confirmée) écrites dans la base de données, et finalement une entrée de confirmation de la transaction est ajoutée au journal. On vous propose plutôt une base de données structurée en journal où les modifications sont simplement écrites dans le journal de même que la confirmation de la transaction. Est-ce suffisant pour permettre de récupérer en cas de redémarrage, tout comme avec les bases de données conventionnelles? Quels sont les avantages et inconvénients d'une telle organisation pour les accès en lecture et en écriture à la base de données? **(1 point)**

Le journal peut effectivement suffire à assurer le recouvrement en cas redémarrage. Une telle écriture séquentielle est aussi très efficace car elle demande très peu de mouvement de la tête de lecture/écriture du disque. Par contre, l'accès efficace en lecture aux données peut

être problématique. La bonne valeur pour une variable (e.g. rangée dans une table de la base de donnée) est la dernière écrite dans le journal. Toutefois, il est possible de maintenir des index du contenu du journal. La mise à jour des index est moins coûteuse car elle peut être faite de manière asynchrone. En effet, il est toujours possible de reconstruire le fichier dérivé qu'est l'index; un arrêt imprévu qui cause une corruption de l'index ne constitue pas une perte d'information.

Question 4 (5 points)

a) Dans votre travail pratique 3, lorsqu'un client envoie un message à une salle de discussion, il envoie un message séparé à chacun des membres de cette salle. Lorsque le nombre de participants est très élévé, ceci peut poser des problèmes de performance. Proposez une amélioration qui permettrait de réduire le nombre total de messages transmis. **(2 points)**

Puisque le même message est envoyé à plusieurs destinataires possiblement en transitant par les mêmes noeuds intermédiaires, il serait plus économique d'envoyer le message une seule fois mais avec une liste de destinataires. Chaque noeud sur le chemin vers les destinataires ne recevrait le message qu'une seule fois. Le multicast a peu de chance d'être disponible sur un tel réseau externe et demanderait de révéler en partie la topologie des communications, ce qu'un réseau poste-à-poste cherche généralement à éviter. De la même manière, prendre un serveur central pour distribuer plus efficacement les messages d'une discussion irait à l'encontre de l'architecture distribuée.

b) Quelles sont les principales caractéristiques du Google Build System? En quoi chacune de ces caractéristiques permet-elle d'être plus rapide et efficace qu'un système de compilation conventionnel où chaque usager ou groupe recompile son logiciel avec des outils comme make (ou parallel make sur une grappe d'ordinateurs)? **(1 point)**

Le GBS conserve tous les fichiers sur les serveurs, ce qui évite des aller-retour inutiles vers la station client. Il vérifie le contenu de chaque fichier (e.g. avec une somme de contrôle) et sait ainsi reconnaître les doublons qu'il peut ne stocker qu'une seule fois. A chaque compilation, il note les options et le fichier d'entrée et vérifie si cette compilation a déjà été effectuée. De cette manière, chaque fichier n'est compilé qu'une seule fois (avec les mêmes options). Finalement, s'il voit que le fichier compilé n'est pas différent de la version précédente (e.g. fichier source dont seulement les commentaires ont été modifiés), il conserve l'ancien fichier compilé et évite ainsi possiblement du travail additionnel qui découlerait d'un nouveau fichier objet. De cette manière, Google augmente la performance autant en mettant plusieurs ordinateurs en parallèle qu'en évitant de refaire plus d'une fois chaque compilation.

c) Avec le format d'encodage sur le réseau des données CORBA CDR, l'envoyeur et le récepteur connaissent à l'avance l'ordre et le type des données échangées. En quoi cela diffère-t-il de ce qui arrive dans les Protocol Buffers de Google? Quels sont les avantages respectifs de chacune de ces manières d'encoder les données sur le réseau. **(1 point)**

L'encodage CDR n'a pas besoin d'encoder la position de ses champs ou leur type. Il peut donc être plus compact et plus rapide à encoder et décoder. Avec Protocol Buffers, le

fait d'identifier le champ et le type permet de tolérer des variations de versions de protocole en autant qu'un numéro de champ ne soit jamais réutilisé pour un même type d'objet. L'information de type peut aussi être utilisée pour varier le nombre d'octets alloués pour la valeur, ce qui peut finalement produire des données encore plus compactes. En effet, pour un entier 32 bits, il arrive souvent que la valeur utilisée soit assez petite et entre facilement dans un seul octet. Dans ce cas, CDR envoie toujours 4 octets alors que Protocol Buffers pourrait prendre moins de place avec un octet pour le numéro de champ, un octet pour le type (entier d'un octet), et l'octet contenant l'entier.

d) Dans votre travail pratique 3, à quoi correspond l'étape d'initialisation (bootstrap) dans Kademlia et pourquoi est-elle nécessaire? Concrètement, dans un tel système poste-à-poste décentralisé, comment est-ce qu'un client qui veut se connecter au réseau peut-il obtenir l'adresse d'un client existant? **(1 point)**

Chaque noeud dans le réseau poste-à-poste maintient des connexions avec quelques proches voisins. Il doit trouver ces voisins les plus proches. Pour cela, il doit entrer en communication avec au moins un noeud déjà connecté au réseau qu'il veut joindre. Ce noeud déjà connecté lui servira de porte d'entrée pour rechercher ses proches voisins. La manière d'amorcer le système est d'avoir une liste de postes connus pour être souvent connectés à ce réseau. Cette liste peut être obtenue de différentes manières, par exemple livrée avec le logiciel, disponible sur un site Web... Les premiers noeuds qui initient le réseau poste-à-poste doivent aussi bénéficier d'information pour les aider à se rejoindre. Parfois, quelques noeuds de départ ont des adresses statiques connues des autres.

Le professeur: Michel Dagenais