

ÉCOLE 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.

$$a = 10h00m04.539s - 10h00mm00.000s = 00m04.539s$$

$$b = 10h00m05.041s - 10h00m00.542s = 00m04.499s$$

$$\text{Ajustement} = (a+b)/2 = (4.539s + 4.499s)/2 = 4.519s$$

$$\text{Précision} = (a-b)/2 = (4.539s - 4.499s)/2 = 0.02s$$

- 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 généralement 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'ordonne 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 personifie. 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 débloquent. 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 séquençement 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 t_1 , U t_2 et V t_3 ($t_0 < t_1 < t_2 < t_3$).

T: Début, t_1

U: Début, t_2

V: Début, t_3

T: Read(x), écrit en t_0 , lu en t_1

U: Read(z), écrit en t_0 , lu en t_2

T: Write(x,1), version tentative pour t_1

T: Compléter, écrire x en t_1 , pas de conflit, validé

U: Read(x), écrit en t_1 , lu en t_2

V: Read(w), écrit en t_0 , lu en t_3

U: Read(y), écrit en t_0 , lu en t_2

U: Write(y,2), version tentative pour t_2

U: Compléter, écrire y en t_2 , pas de conflit, validé

V: Read(x), écrit en t_1 , lu en t_3

V: Read(y), écrit en t_2 , lu en t_3

V: Write(z,3), version tentative pour t_3

V: Compléter, écrire z en t_3 , 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 défaut 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 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 messages 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 élevé, 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'expéditeur 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 Kademia 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

ÉCOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF4410: Systèmes répartis et infonuagique (Automne 2013)

3 crédits (3-1.5-4.5)

CORRIGÉ DE L'EXAMEN FINAL

DATE: Samedi le 7 décembre 2013

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) Trois processus utilisent des horloges logiques qui sont incrémentées à chaque événement, incluant les envois et réceptions de messages entre processus. Les événements vus par chacun sont listés. Associez un vecteur de compteurs d'événements à chacun ($\langle p1, p2, p3 \rangle$), fournissant ainsi ces mêmes trois listes, mais avec des vecteurs de compteurs plutôt que des compteurs. À l'aide de ces vecteurs, que pouvez-vous dire de l'ordre relatif (avant, concurrent, après) des événements 2 de $p1$ versus 2 de $p3$, et 4 de $p1$ versus 4 de $p2$. Justifiez. (2 points)

$p1$: 1 lecture; 2 message A vers $p2$; 3 écriture; 4 écriture; 5 message B de $p3$

$p2$: 1 message A de $p1$; 2 écriture; 3 message C vers $p3$; 4 lecture; 5 message D de $p3$

$p3$: 1 lecture; 2 message C de $p2$; 3 écriture; 4 message B vers $p1$; 5 message D vers $p2$

Pour construire les vecteurs de compteurs d'événements, il faut initialiser les compteurs à 0 et joindre les contenus des vecteurs des envoyeurs lors de la réception d'un message, ceci donne ce qui suit.

$p1$: $\langle 1, 0, 0 \rangle$ lecture; $\langle 2, 0, 0 \rangle$ message A vers $p2$; $\langle 3, 0, 0 \rangle$ écriture; $\langle 4, 0, 0 \rangle$ écriture; $\langle 5, 3, 4 \rangle$ message B de $p3$

$p2$: $\langle 2, 1, 0 \rangle$ message A de $p1$; $\langle 2, 2, 0 \rangle$ écriture; $\langle 2, 3, 0 \rangle$ message C vers $p3$; $\langle 2, 4, 0 \rangle$ lecture; $\langle 2, 5, 5 \rangle$ message D de $p3$

$p3$: $\langle 0, 0, 1 \rangle$ lecture; $\langle 2, 3, 2 \rangle$ message C de $p2$; $\langle 2, 3, 3 \rangle$ écriture; $\langle 2, 3, 4 \rangle$ message B vers $p1$; $\langle 2, 3, 5 \rangle$ message D vers $p2$

En comparant $p1:3$, vecteur de $\langle 2, 0, 0 \rangle$, avec $p3:2$, vecteur $\langle 2, 3, 2 \rangle$, puisque toutes les entrées du second sont supérieures ou égales au premier, ce dernier est postérieur au premier. Dans l'autre cas, $p1:4$, vecteur $\langle 4, 0, 0 \rangle$, versus $p2:4$, vecteur $\langle 2, 4, 0 \rangle$, aucun vecteur n'est strictement supérieur ou égal à l'autre. On ne peut donc rien conclure et on doit considérer les événements comme concurrents.

- b) Un ordinateur A envoie un message à B à 9h00m10.200s pour obtenir le temps et reçoit une réponse à 9h00m10.600s. L'ordinateur B reçoit la requête de A à 9h00m05.300s à son heure, envoie une requête à C à 9h00m05.400s, reçoit la réponse de C à 9h00m05.500s et envoie sa réponse à A à 9h00m05.550s. Le serveur C reçoit la demande de B à 9h00m00.300s et lui répond à 9h00m00.350s, heure de C. La réponse finale à A contient toutes ces valeurs de temps ainsi que leur origine. A peut donc appliquer l'algorithme utilisé par NTP pour synchroniser son horloge avec celle de C, celle-ci étant très précise. Quel est le décalage à appliquer sur A? Quel est l'intervalle d'incertitude associé? (2 points)

Nous pouvons appliquer de manière imbriquée l'algorithme de NTP. En regardant B par rapport à C, on a:

$$a = 9h00m00.300s - 9h00m05.400s = -5.100s$$

$$b = 9h00m00.350s - 9h00m05.500s = -5.150s$$

$$\text{Ajustement} = (a+b)/2 = (-5.100s - 5.150s)/2 = -5.125s$$

$$\text{Précision} = (a-b)/2 = (-5.100s + 5.150s)/2 = 0.025s$$

Nous pouvons alors convertir les temps sur B en temps de C: $9h00m05.300 - 5.125 = 9h00m00.175$ et $9h00m05.550 - 5.125 = 9h00m00.425$. Le calcul de A par rapport à B (synchronisé à C) donne donc:

$$a = 9h00m00.175s - 9h00m10.200s = -10.025s$$

$$b = 9h00m00.425s - 9h00m10.600s = -10.175s$$

$$\text{Ajustement} = (a+b)/2 = (-10.025s - 10.175s)/2 = -10.100s$$

$$\text{Précision} = (a-b)/2 = (-10.025s + 10.175s)/2 = -0.075s$$

Le décalage à appliquer est ainsi de $-10.1s$ et l'incertitude est la somme des deux et est donc de $\pm (.075 + .025)$ soit $\pm .1s$. Nous pouvons aussi calculer le décalage de A à B sans convertir le temps de B:

$$a = 9h00m05.300s - 9h00m10.200s = -4.9s$$

$$b = 9h00m05.550s - 9h00m10.600s = -5.05s$$

$$\text{Ajustement} = (a+b)/2 = (-4.9s - 5.05s)/2 = -4.975s$$

$$\text{Précision} = (a-b)/2 = (-4.9s + 5.05s)/2 = -0.075s$$

Il suffit alors d'additionner les deux décalages à appliquer (C à B et B à A), ce qui donne $-5.125 - 4.975 = -10.1s$ ainsi que les imprécisions $.075 + .025 = .1s$, ce qui donne la même chose que le résultat précédent. Une troisième manière de résoudre le problème est de simplement soustraire le temps passé dans B à servir d'intermédiaire entre A et C et faire comme si A avait envoyé sa demande $9h00m05.400s - 9h00m05.300s = .1s$ plus tard et reçu la réponse $9h00m05.550s - 9h00m05.500s = .05s$ plus tôt. Ceci donnerait:

$$a = 9h00m00.300s - (9h00m10.200s + .100s) = -10s$$

$$b = 9h00m00.350s - (9h00m10.600s - .050s) = -10.200s$$

$$\text{Ajustement} = (a+b)/2 = (-10s - 10.200s)/2 = -10.1s$$

$$\text{Précision} = (a-b)/2 = (-10s + 9.8s)/2 = -.100s$$

- c) Lorsque la méthode de l'élection hiérarchique est utilisée, afin de trouver le serveur de plus haute priorité disponible pour être élu coordonnateur, on dit que le nombre de messages envoyés peut être de l'ordre de n^2 dans le pire cas. Décrivez un cas où cela se produirait? (1 point)

Si le réseau est lent et que chaque ordinateur déclenche une élection en même temps, chacun aura le temps d'envoyer un message à chaque autre ordinateur de plus haute priorité avant d'obtenir une réponse. Ceci représente $n-1 + n-2 + \dots + 1$ message, soit $n^2/2 - n$ messages. Ensuite, chaque ordinateur, ne recevant pas encore de message de réponse, peut vouloir se déclarer élu, avec $n - 1$ messages (ou un envoi à tous) pour se proclamer coordonnateur, ce qui représente $n \times (n - 1) = n^2 - n$ messages.

Question 2 (5 points)

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

T: Début
U: Début
T: Read(a)
U: Read(a)
T: Read(b)
T: Write(b,1)
T: Compléter
V: Début
V: Read(b)
U: Read(b)
U: Read(c)
U: Write(b,2)
U: Write(c,3)
U: Compléter
V: Read(c)
V: Write(c)
V: Compléter

- a) Lesquelles des transactions T, U et V pourraient s'effectuer ainsi, si un contrôle de la concurrence par prise de verrou (partagé pour la lecture et exclusif pour l'écriture) est utilisé? Justifiez. **(2 points)**

T acquière un verrou en lecture sur a, ensuite partagé avec U. T acquière un verrou en lecture sur b et le promeut en verrou d'écriture et peut compléter, relâchant ses verrous. V prend un verrou de lecture sur b, ensuite le partage avec U. U acquière un verrou sur c et veut ensuite promouvoir son verrou sur b pour l'écriture mais ne peut le faire à cause de V. S'il abandonne, cela laisse libre cours à V. S'il attend, U et V se bloqueront mutuellement car V voudra le verrou sur c.

- b) 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)**

En reculant, au moment de compléter T, il n'y a pas de problème. Pour compléter U, il faut vérifier ce qu'il a lu (a, b, c) versus ce qui a été écrit par les transactions précédentes (T qui a écrit b). Il y a intersection et donc potentiellement conflit. U est abandonné. Ceci laisse le libre champ à V, puisque V a commencé après la fin de T et n'est donc pas concurrent avec T.

En avançant, il faut vérifier ce qui est écrit par T (b) versus ce que U a lu à ce moment (a). Il n'y a pas d'intersection et T peut compléter. Au moment de compléter U, ses écritures (b, c) intersectent les lectures de V (b) et U doit être abandonné, laissant le champ libre à V.

- c) Un client effectue une série d'opérations dans le cadre d'une transaction avec plusieurs sous-transactions imbriquées. Indiquez lesquelles des variables a à e seront modifiées par la transaction ainsi que leur nouvelle valeur. **(1 point)**

```
Début T1
  Write(a, 2)
  Début T1.1
    Write(b, 4)
    Début T1.1.1
      Write(c, 6)
      Commettre T1.1.1
    Write(d, 8)
    Abandonner T1.1
  Write(b, 10)
  Début T1.2
    Write(e, 3)
    Début T1.2.1
      Write(e, 5)
      Abandonner T1.2.1
    Commettre T1.2
  Commettre T1
```

Après avoir enlevé les transactions abandonnées (et celles imbriquées). il reste:

```
Début T1
  Write(a, 2)
  Write(b, 10)
  Début T1.2
    Write(e, 3)
    Commettre T1.2
  Commettre T1
```

Les variables affectées sont donc a=2, b=10, et e=3.

Question 3 (5 points)

- a) Un jeune finissant veut offrir un nouveau service internet et doit se monter une infrastructure au plus bas coût possible. Son infrastructure est constituée de deux serveurs de base de données redondants et de 20 serveurs Web. Chaque serveur de base de données est constitué d'un ordinateur, et de deux disques redondants en miroir. Chaque serveur Web est constitué d'un ordinateur et d'un disque. Son fournisseur Internet s'occupe de répartir les requêtes entre ses serveurs Web. Le matériel utilisé a été recyclé et est en mauvais état. A chaque instant, chaque disque a une probabilité d'être en panne de 0.2 et chaque ordinateur (hormis

son ou ses disques) a une probabilité de 0.1. Quelle est la probabilité que tout (incluant chaque élément redondant) soit fonctionnel en même temps, (aucun disque ou ordinateur en panne)? Quelle est la probabilité que le service soit complètement non disponible?

(2 points)

Il y a 22 ordinateurs, chacun opérationnel avec une probabilité de .9, et 24 disques, chacun opérationnel avec une probabilité de .8. La probabilité que tout soit fonctionnel est de $.9^{22} \times .8^{24} = 0.000465$. Chaque serveur de base de données sera opérationnel si l'ordinateur est opérationnel ainsi que les disques. Les disques le sont sauf si les deux sont en panne, une probabilité de $1 - .2 \times .2 = .96$. Chaque serveur a donc une probabilité de $0.96 \times 0.9 = 0.864$ d'être opérationnel. Le service sera offert sauf si les deux serveurs de base de données sont en panne $1 - (1 - 0.864)^2 = 0.9815$. Ensuite, chaque serveur Web est opérationnel si son disque et ordinateur le sont, soit une probabilité de $0.8 \times 0.9 = 0.72$. Le service Web sera opérationnel sauf si tous les serveurs Web sont en panne, une probabilité de $1 - (1 - 0.72)^{20} = 8.7 \times 10^{-12}$. Le service sera non disponible sauf si les serveurs Web et les serveurs de base de données sont disponibles $1 - ((1 - 8.7 \times 10^{-12}) \times 0.9815) = 0.0185$.

- b) Une transaction répartie effectue les opérations suivantes sur un des serveurs impliqués. Expliquez qu'est-ce qui devra être écrit dans le journal qui permet la récupération en cas de panne, et à quel moment ceci devra être fait au plus tôt et au plus tard (i.e. après quelle opération)? **(2 points)**

- 1: lire a
- 2: écrire 40, b
- 3: lire c
- 4: écrire 60, d
- 5: préparer à commettre
- 6: commettre

Les lectures n'ont pas à être indiquées dans le journal. Les valeurs écrites pour b et d peuvent être écrites dès que reçues ou au plus tard juste avant d'écrire l'instruction de préparation dans le journal. L'opération de se préparer à commettre doit être écrite lorsque reçue, avant de répondre pour confirmer que le serveur est prêt à commettre. L'opération commettre doit être ajoutée dans le journal après avoir été reçue. Il est avantageux (mais pas strictement requis) de le faire rapidement, pour éviter d'avoir à révérifier son statut en cas de panne avant de l'avoir écrit.

- c) Dans le cadre du travail pratique 2, un répartiteur demandait à des serveurs de compter les incidences des mots dans des morceaux de texte. Lorsqu'un serveur faisait défaut et ne fournissait pas la réponse, le répartiteur pouvait demander à un autre serveur de refaire le travail, offrant ainsi une tolérance aux pannes. Cependant, le répartiteur demeure un maillon faible. Proposez une architecture qui permette d'améliorer la résilience du répartiteur. **(1 point)**

Le répartiteur pourrait écrire les résultats obtenus sur disque et les recouvrer après un redémarrage. Il pourrait y avoir deux répartiteurs redondants, tous deux actifs, qui se répartissent les serveurs et qui s'informent mutuellement des progrès réalisés.

Question 4 (5 points)

- a) Une entreprise opère 9 ordinateurs identiques, 3 pour chacun de trois départements très semblables. Chacun des 3 ordinateurs pour un département est affecté à un service particulier (serveur LDAP, serveur de fichiers, serveur de compilation) et ces trois serveurs ont un taux d'occupation moyen de 10%, 20% et 30% respectivement. Un administrateur de système, inspiré par le cours INF4410, propose de consolider ces serveurs en les virtualisant, ce qui permettrait de réduire le nombre d'ordinateurs requis. Le surcoût de la virtualisation est de 20%, ce qui prenait 1 seconde en prendrait 1.2. Deux solutions sont envisagées, conserver un ordinateur par département qui roule les trois machines virtuelles de ce département, ou avoir un nuage de 3 ordinateurs physiques sur lesquels seraient répartis les 9 machines virtuelles. Que deviendrait le taux d'utilisation moyen dans chaque cas, en supposant que la charge sur chaque serveur était assez uniforme dans le temps? Quelle solution vous semble la plus intéressante? Pourquoi? **(2 points)**

Sur 100 secondes, les trois serveurs en prenaient respectivement 10, 20 et 30, soit un total de 60. Avec le surcoût de la virtualisation, ceci deviendrait $60 \times 1.2 = 72$, soit un taux d'occupation de 72%. Il n'y a pas de différence de taux d'utilisation entre les deux cas, trois fois plus de charge sur trois ordinateurs au lieu d'un seul. La solution du nuage est plus intéressante car elle permet une certaine tolérance aux pannes. Par contre, elle peut être un peu plus difficile à mettre en oeuvre, et ne sépare pas tout à fait aussi bien les activités des trois départements au niveau de la sécurité informatique.

- b) Sur le nuage de la compagnie Amazon, un service de répartiteur existe qui envoie les requêtes reçues à tour de rôle à un des serveurs disponibles. Le répartiteur reçoit de l'information sur les différents serveurs (taux d'utilisation des serveurs, et temps de réponse aux requêtes). Le répartiteur peut aussi décider d'instancier des serveurs supplémentaires ou de retirer des serveurs instanciés. Quel critère est-ce que le répartiteur utilise pour choisir le prochain serveur auquel envoyer une requête reçue? Quel critère utilise-t-il pour décider d'activer une instance supplémentaire de serveur? Pour retirer une instance de serveur? **(2 points)**

Le répartiteur envoie la requête reçue au serveur qui répond le plus rapidement, ou les distribue à tour de rôle lorsque la différence de temps n'est pas importante. Lorsque le taux d'utilisation des serveurs est trop élevé, de nouvelles instances sont ajoutées. Lorsque le taux d'utilisation est trop faible, des instances sont retirées.

- c) Les solutions de virtualisation comme KVM offrent la virtualisation complète ou la paravirtualisation. Quels sont les avantages et limitations de ces deux alternatives? Dans quelle situation choisirait-on de préférence chacune? **(1 point)**

La paravirtualisation est plus rapide puisque l'opération demandée est directement déléguée à la machine physique, plutôt que d'émuler le comportement d'un dispositif physique particulier, comme une carte réseau, avant de déléguer le travail à la machine physique. Toutefois, la paravirtualisation n'est possible que lorsqu'on peut facilement configurer de cette manière la machine virtuelle. Lorsqu'on a plein contrôle sur la configuration de la machine virtuelle, la paravirtualisation est préférable puisqu'elle est plus rapide et ne requiert pas de support matériel. Par contre, s'il n'est pas possible de modifier la machine virtualisée, par exemple

parce qu'elle est fournie par un client qui ne sait pas que son ordinateur est virtualisé ou qui n'est pas intéressé ou capable de modifier sa configuration, alors la virtualisation complète sera nécessaire.

Le professeur: Michel Dagenais

ÉCOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF4410: Systèmes répartis et infonuagique (Automne 2014)

3 crédits (3-1.5-4.5)

CORRIGÉ DE L'EXAMEN FINAL

DATE: Vendredi le 19 décembre 2014

HEURE: 13h30 à 16h00

DUREE: 2H30

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

- a) Un ordinateur A envoie un message à B à 13h30m10.250, heure de A, pour lui dire qu'il est 13h30m10.250. L'ordinateur B reçoit ce message à 13h30m08.350, heure de B. Quel ajustement devrait appliquer B pour se synchroniser sur A? Énoncez vos hypothèses. Peut-on calculer l'incertitude sur cet ajustement? Si oui, quel est-il? **(2 points)**

Il n'y a pas moyen a priori de savoir quel est le délai sur le réseau. On peut donc assumer qu'il est négligeable. Ceci ferait qu'à 13h30m08.350 heure de B, on sait qu'il est près de 13h30m10.250 heure de A, et qu'un ajustement de $13h30m08.350 - 13h30m10.250 = -1.9s$ serait requis. Nous n'avons pas de donnée sur l'incertitude de cet ajustement. On pourrait faire des hypothèses sur le délai minimal et maximal requis pour l'envoi du message, surtout si on a plus d'information sur le type de réseau qui relie les deux ordinateurs. On pourrait alors utiliser la valeur moyenne de délai réseau et la soustraire de l'ajustement calculé. Par exemple, pour un délai minimal de .1ms et maximal de 10ms, on aurait une moyenne de 5.05ms, ce qui donnerait un ajustement de $-1.90505s$. On pourrait alors utiliser $\pm 5.05ms$ comme incertitude si ces valeurs minimales et maximales sont fiables.

- b) Trois processus utilisent des horloges logiques qui sont incrémentées à chaque événement, incluant les envois et réceptions de messages entre processus. Les événements vus par chacun sont listés ci-après. On veut fournir des clichés cohérents de l'état global du système. L'état global est déterminé par la liste des événements inclus, et donc pour chaque processus par la partition incluse de leur liste des événements. La liste complète des événements pour chaque processus, incluant tous les messages échangés, est disponible. Il faut donc choisir un point de coupure dans la liste de chacun des trois processus de sorte que le cliché résultant de l'état global soit cohérent. On vous propose deux clichés possibles (points de coupure pour chaque processus): (p1: entre les événements 2 et 3, p2: entre les événements 2 et 3, p3: entre les événements 2 et 3) et (p1: entre les événements 3 et 4, p2: entre les événements 3 et 4, p3: entre les événements 3 et 4). Pour chacun de ces deux clichés, dites s'il est cohérent ou non et expliquez pourquoi. **(2 points)**

p1: 1 message A vers p2; 2 message B de p3; 3 message C vers p3; 4 message D de p2; 5 message E vers p2, 6 message F de p2; 7 message G de p3

p2: 1 message A de p1; 2 message D vers p1; 3 message H vers p3; 4 message F vers p1; 5 message I de p3; 6 message E de p1

p3: 1 message B vers p1; 2 message C de p1; 3 message I vers p2; 4 message H de p2; 5 message G vers p1

Un cliché est cohérent si pour chaque événement inclus, tous les événements qui le précèdent sont aussi inclus. Ceci peut se résoudre graphiquement en vérifiant qu'aucun message n'a son origine après la frontière et sa destination avant la frontière. Une autre méthode est de vérifier le dernier événement inclus de chaque processus (les autres inclus sont plus antérieurs) versus le premier événement non inclus de chaque autre processus (les autres non inclus sont plus postérieurs) en utilisant les vecteurs d'horloges logiques.

$p1$: $\langle 1,0,0 \rangle$ message A vers $p2$; $\langle 2,0,1 \rangle$ message B de $p3$; $\langle 3,0,1 \rangle$ message C vers $p3$;
 $\langle 4,2,1 \rangle$ message D de $p2$; $\langle 5,2,1 \rangle$ message E vers $p2$; $\langle 6,4,1 \rangle$ message F de $p2$;
 $\langle 7,4,5 \rangle$ message G de $p3$

$p2$: $\langle 1,1,0 \rangle$ message A de $p1$; $\langle 1,2,0 \rangle$ message D vers $p1$; $\langle 1,3,0 \rangle$ message H vers $p3$;
 $\langle 1,4,0 \rangle$ message F vers $p1$; $\langle 3,5,3 \rangle$ message I de $p3$; $\langle 5,6,3 \rangle$ message E de $p1$

$p3$: $\langle 0,0,1 \rangle$ message B vers $p1$; $\langle 3,0,2 \rangle$ message C de $p1$; $\langle 3,0,3 \rangle$ message I vers $p2$;
 $\langle 3,3,4 \rangle$ message H de $p2$; $\langle 3,3,5 \rangle$ message G vers $p1$

Pour le premier cas, il faut comparer $\langle 3,0,1 \rangle$, $\langle 1,3,0 \rangle$ et $\langle 3,0,3 \rangle$ (exclus) qui ne doivent pas être antérieurs à $\langle 2,0,1 \rangle$, $\langle 1,2,0 \rangle$ et $\langle 3,0,2 \rangle$ (inclus). Ceci ne fonctionne pas car $\langle 3,0,1 \rangle$ est effectivement antérieur à $\langle 3,0,2 \rangle$ et le cliché n'est pas cohérent. Dans le second cas, il faut comparer $\langle 4,2,1 \rangle$, $\langle 1,4,0 \rangle$ et $\langle 3,3,4 \rangle$ (exclus) qui ne doivent pas être antérieurs à $\langle 3,0,1 \rangle$, $\langle 1,3,0 \rangle$ et $\langle 3,0,3 \rangle$ (inclus). C'est le cas et ce deuxième cliché est cohérent.

- c) Un professeur de géologie vous explique que la durée de la rotation de la terre sur son axe varie dans le temps en raison du mouvement dans le noyau liquide de la terre et même des marées. Plus encore, la vitesse de rotation de la terre diminue graduellement (la durée des jours augmente) en raison du frottement associé aux marées. Puisque les journées font généralement 24 heures, les heures 60 minutes et les minutes 60 secondes, comment tient-on compte de ce facteur? Est-ce que la durée des secondes varie d'une journée à l'autre pour correspondre à $24 \times 60 \times 60$ secondes par jour? Ou est-ce que le nombre de secondes par jour varie? Expliquez. **(1 point)**

La durée des journées varie tout de même très peu. Le temps universel coordonné (UTC) ajoute ou retranche une seconde de temps en temps, soit le 30 juin ou le 31 décembre à la fin de la journée, lorsque la différence avec le temps solaire est trop grande. En pratique, une seconde est ajoutée une fois par 3 ans environ. Le logiciel d'ajustement de l'heure sur l'ordinateur peut choisir de faire varier la durée des secondes afin de cacher cette seconde ajoutée qui pourrait confondre certains logiciels (e.g., calendrier).

Question 2 (5 points)

- a) Quatre transactions, T, U, V et W s'exécutent concurremment. Le séquençement des opérations est fourni ci-après. Lesquelles des transactions T, U, V et W pourraient être validées si une validation en reculant était utilisée pour vérifier la cohérence des transactions? Une validation en avançant? Justifiez. **(3 points)**

T: Début

U: Début

T: Read(a)

T: Read(b)

T: Read(c)

U: Read(c)
 U: Read(d)
 V: Début
 V: Read(e)
 T: Write(b)
 T: Write(f)
 T: Compléter
 W: Début
 W: Read(e)
 W: Read(f)
 U: Write(c)
 U: Write(d)
 U: Compléter
 V: Read(d)
 W: Write(a)
 W: Write(b)
 W: Compléter
 V: Read(g)
 V: Write(e)
 V: Compléter

En reculant, au moment de compléter T, il n'y a pas de problème. Pour compléter U, il faut vérifier ce qu'il a lu (c, d) versus ce qui a été écrit par les transactions précédentes concurrentes (T qui a écrit b, f). Il n'y a pas d'intersection et U peut compléter. Pour W, ses lectures (e, f) sont vérifiées versus les écritures de U (c, d), alors que T était terminé avant que W ne débute. Il n'y a pas de problème là non plus et W peut compléter. Pour V, les lectures (d, e, g) doivent être vérifiées versus les écritures de T, U et W (a, b, c, d, f). Il y a intersection (d) et la transaction V doit être abandonnée. En fait, U était complété avant que V ne lise la valeur de d mais cette information plus détaillée n'est pas utilisée par la méthode de base en reculant, elle est utilisée par la méthode par estampille de temps.

En avançant, il faut vérifier ce qui est écrit par T (b, f) versus ce que U et V ont lu à ce moment (c, d, e). Il n'y a pas d'intersection et T peut compléter. Au moment de compléter U, ses écritures (c, d) sont comparées aux lectures à ce moment de V et W (e, f); les lectures à venir de V et W ne sont pas encore connues à ce moment et n'ont de toutes manières pas d'impact sur la cohérence. Il n'y a pas d'intersection et la transaction U est validée. Pour W, ses écritures (a, b) sont comparées aux lectures de V à ce moment (d, e). Il n'y a pas d'intersection et W peut compléter. La dernière transaction, V, complète nécessairement et ainsi les quatre transactions sont validées.

- b) Pour les transactions réparties, un protocole à deux phases est utilisé. Lors de la première phase, on demande à chaque serveur de préparer la transaction et de confirmer qu'il est prêt à commettre. Lors de la seconde phase, on confirme à chaque serveur et au client que la transaction a été acceptée par tous et est commise. A quel moment au plus tôt peut-on envoyer le message au client que la transaction est confirmée? Au plus tard? A quel moment au plus tôt peut-on envoyer le message à chaque serveur impliqué dans la transaction que celle-ci est confirmée? Au plus tard? **(1 point)**

La confirmation peut être envoyée au client au plus tôt après avoir reçu de chaque serveur la confirmation qu'il est prêt à commettre. Il n'y a pas de borne pour le plus tard, mais il n'y a pas intérêt à faire attendre le client non plus. La situation est exactement la même pour confirmer la transaction aux serveurs. Le plus tôt est après avoir reçu de chaque serveur la confirmation qu'il est prêt à commettre et il n'y a pas de borne pour le plus tard. Là encore, il est désavantageux d'attendre trop longtemps puisque cela retarde la possibilité pour chaque serveur de clore la transaction, et de l'enlever de sa liste de transactions actives et éventuellement de son journal.

- c) Pour un système transactionnel, vous hésitez entre un système de maintien de la cohérence avec des verrous ou un système de validation de la cohérence avec la méthode en reculant. Comment est-ce que ces deux systèmes se compareront en termes de vitesse et de temps d'attente pour accéder aux ressources? Est-ce que d'autres facteurs sont à considérer? Le système le plus lent présente-t-il d'autres avantages? **(1 point)**

Les verrous imposent des opérations supplémentaires pendant la transaction (prise des verrous) et peuvent causer de l'attente pour l'obtention de ces verrous. L'utilisation des verrous est donc a priori plus lente que la validation optimiste de la concurrence par la méthode en reculant. Par contre, avec la validation optimiste de la concurrence, un certain nombre de transactions pourront devoir être abandonnées. Ceci peut enlever tout l'avantage procuré par l'exécution plus rapide sans verrou. Plus encore, dans certaines applications, le fait de devoir invalider une transaction demandée par un client peut être un désagrément majeur. Pour cette raison, la méthode avec les verrous demeure la plus intéressante dans un grand nombre d'applications.

Question 3 (5 points)

- a) On vous demande de choisir entre deux configurations pour le prochain serveur de fichiers de votre entreprise. La première configuration consiste en un ordinateur avec 4 disques en miroir. Chaque disque contient l'ensemble des données et un seul disque suffit donc. Le second système est constitué de deux ordinateurs, qui sont deux serveurs redondants, chacun étant connecté à deux disques en miroir et un seul disque suffit donc pour un serveur. La probabilité qu'un ordinateur soit opérationnel (hormis les disques) est de 0.95. La probabilité qu'un disque soit opérationnel est de 0.85. Quelle est la probabilité que le service soit disponible, pour chacune des deux configurations? **(2 points)**

Dans la première configuration, le système de disque est disponible sauf si les 4 sont indisponibles $1 - (1 - .85)^4 = 0.99949375$. Le service sera disponible si les disques et l'ordinateur le sont, $0.99949375 \times 0.95 = 0.949519063$. Dans le second cas, sur un serveur, les disques seront disponibles sauf si les 2 sont indisponibles $1 - (1 - .85)^2 = 0.9775$. Un serveur sera disponible si l'ordinateur et les disques le sont $0.9775 \times 0.95 = 0.928625$. Le service sera disponible sauf si les deux serveurs redondants sont indisponibles $1 - (1 - 0.928625)^2 = 0.994905609$. Cette deuxième configuration est donc nettement mieux pour réduire le temps d'indisponibilité. Elle est toutefois légèrement plus coûteuse puisqu'elle utilise deux ordinateurs plutôt qu'un, tout en conservant le même nombre de disques.

- b) Les interblocages posent un problème sérieux dans les systèmes de base de données répartis.
- i) Comment peut-on détecter un interblocage dans un tel système?
 - ii) Est-il possible de le faire sans simultanément arrêter tous les processus impliqués?
 - iii) Doit-on constamment vérifier la présence d'interblocages, ou peut-on le faire seulement à la demande (selon quel critère)?
 - iv) A défaut de faire une détection précise des interblocages, quel mécanisme peut-on utiliser pour s'assurer de ne pas laisser des transactions bloquées indéfiniment? **(2 points)**

i) Pour détecter les interblocages, il est possible de construire un graphe de dépendance entre les verrous, ceux qui les possèdent et ceux qui les attendent, et de vérifier l'existence d'un cycle dans le graphe. ii) A défaut de tout arrêter pour avoir un cliché exact du graphe réparti de dépendance, il est possible de le construire de manière incrémentale un graphe approximatif en interrogeant les processus impliqués les uns après les autres. Si un interblocage est présent, les dépendances en cause sont figées et ceci apparaîtra dans le graphe obtenu, même s'il est construit incrémentalement. Cependant, si une transaction est annulée au milieu de la construction de notre graphe, il est possible que nous détectons de manière erronée un interblocage. iii) La vérification des interblocages ne se fait pas sans arrêt puisque ce serait trop coûteux. Elle peut se faire à intervalle régulier ou seulement lorsqu'il est détecté qu'une transaction reste anormalement longtemps en attente d'un verrou. iv) Dans la plupart des cas, la détection des interblocages n'est pas faite directement. Chaque transaction est simplement annulée si elle reste bloquée trop longtemps.

- c) Vous avez besoin d'un service d'exclusion mutuelle pour votre système réparti. On vous propose un système symétrique réparti. Pour obtenir un verrou, un système demande la permission de chaque autre système. Celui qui détient le verrou attend d'en avoir terminé avant de donner la permission. Celui qui est en demande de verrou attend d'avoir obtenu et fini du verrou avant de donner la permission. Est-ce que ce système est sûr, vivace et respecte l'ordre? Expliquez. Est-ce que ce système est efficace? **(1 point)**

Ce système est sûr puisque la permission de chacun ne peut être obtenue tant qu'un autre processus détient le verrou. Ce système respecte l'ordre puisque chaque demande est traitée dans l'ordre par chaque processus. Le système est vivace puisque chacun passe dans l'ordre et que le système ne peut normalement bloquer; ainsi chacun verra éventuellement sa demande satisfaite. Ce système n'est pas efficace puisqu'il ne fonctionne que si les n processus répartis sont opérationnels, et puisque l'obtention de chaque verrou demande au moins n messages, contre 2 pour un serveur central.

Question 4 (5 points)

- a) Dans le travail pratique 3, à la question 3, on vous demande d'écrire une section pour votre gabarit heat qui définit une alarme qui est déclenchée lorsque le taux moyen d'utilisation du CPU d'une machine est supérieur à 90% sur une période de 1 minute. Ecrivez une version modifiée qui déclenche une alarme de la même manière, lorsque le taux moyen d'utilisation du CPU d'une machine est supérieur à CPU_limit_high sur une période de CPU_limit_period, où CPU_limit_high et CPU_limit_period sont des paramètres d'entrée pour le gabarit. Donnez la portion de gabarit à insérer dans la section *resources*: ainsi que la portion à ajouter dans la section *parameters*:. **(2 points)**

Il faut ajouter les sections de gabarit suivantes dans `parameters` : et `resources` : respectivement.

```
parameters:
  CPU_limit_high:
    type: number
    label: CPU usage
    description: CPU usage percentage high threshold value
    constraints:
      - range: { min: 0, max: 100 }
  CPU_limit_period:
    type: number
    label: CPU usage period
    description: CPU usage period in seconds

resources:
  cpu_alarm_high:
    type: OS::Ceilometer::Alarm
    properties:
      description: Augmentation de 1 si cpu > CPU_limit_high
      meter_name: cpu_util
      statistic: avg
      period: {get_param: CPU_limit_period}
      evaluation_periods: 1
      threshold: {get_param: CPU_limit_high}
      alarm_actions:
        - {get_attr: [web_server_scaleup_policy, alarm_url]}
    matching_metadata: {'metadata.user_metadata.stack':
      {get_param: "OS::stack_id"}}
    comparison_operator: gt
```

- b) Quels sont les principaux avantages de Protocol Buffers de Google, comparé à un système de RPC comme les Sun RPC? (1 point)

Protocol buffers est un protocole simple pour l'envoi de messages utilisé pour les appels de procédure à distance chez Google. Il peut fonctionner avec plusieurs langages et se distingue par le fait que le contenu des messages est compact et auto-descriptif. En effet, un message est une structure de donnée composée d'une séquence de champs. Chaque champ est numéroté et typé. On peut donc recevoir un message et décoder ses champs (séquence de: numéro de champ, type et valeur) sans en connaître la déclaration complète au préalable. Ceci contraste avec SUN RPC ou CORBA où un message ne peut être interprété sans avoir la déclaration exacte de sa structure, ce qui est très contraignant si on veut faire évoluer un système sans mettre à jour simultanément tous les clients et serveurs.

Les entiers utilisent le nombre d'octets minimal, soit 1 octet pour un nombre dont la valeur est inférieure à 128. Les entiers étant très fréquents et leur valeur étant souvent assez petite, ceci fait que les messages sont souvent plus compacts que ceux de SUN RPC ou CORBA en dépit du fait qu'ils viennent avec un nombre variable de champs et avec l'information

de type. Le nombre variable de champs permet par exemple à un client de lire un message d'une version plus nouvelle du protocole pour un service, en ignorant simplement les champs ajoutés. Il permet aussi de lire un message d'une version plus ancienne du protocole, pour laquelle il manque des champs, à condition que celui qui a défini le format du message ait spécifié des valeurs par défaut.

- c) Quelle est l'utilité pour les compagnies impliquées dans le consortium OpenStack.org de développer un logiciel comme OpenStack, alors qu'Amazon offre déjà un tel service de nuage élastique? **(1 point)**

Amazon offre la location de machines virtuelles en utilisant une infrastructure semblable à OpenStack. Cependant, le logiciel utilisé par Amazon lui est propre et n'est pas disponible pour les autres. Différentes compagnies collaborent donc pour développer un logiciel, OpenStack, qui leur permettra de développer un service comparable à celui d'Amazon, soit pour leurs besoins internes, soit pour leurs clients, possiblement en compétition avec Amazon. De plus, les membres de OpenStack ont décidé d'utiliser un modèle à code source libre afin de permettre à chaque utilisateur du logiciel d'avoir toute la liberté voulue pour modifier ou utiliser ce logiciel.

- d) Vous voulez rapidement mettre sur pied un service Web de type commerce en ligne, qui peut s'adapter à la charge et qui soit tolérant aux pannes, sans avoir à implémenter ou gérer ces fonctions, à s'occuper des copies de sécurité des données (backup) ou à gérer et entretenir le matériel. Si vous décidez d'utiliser le nuage d'Amazon à cette fin, quels services spécifiques offerts par Amazon utiliserez-vous? Décrivez rapidement l'architecture de votre système avec les serveurs et répartiteurs réseau associés. **(1 point)**

Le service de machines virtuelles d'Amazon permet de ne pas avoir à gérer de matériel et d'avoir accès à un nombre variable d'instances pour s'adapter à la charge. Le répartiteur des requêtes sert à équilibrer la charge entre les différentes instances et peut aussi redémarrer les instances bloquées et augmenter ou diminuer le nombre des instances en fonction de la charge observée. Finalement, il est possible de ne mettre sur les instances que la partie interaction et présentation, sans stocker de données qui nécessiteraient des copies de sécurité. À la place, toutes les informations sur le stock et les commandes peuvent être maintenues dans une base de données, service qui est offert par Amazon. Toute la gestion des mises à jour du logiciel de base de données ou des copies de sécurité pour assurer la tolérance aux pannes est alors prise en charge par Amazon. Ainsi, le répartiteur reçoit les requêtes et les envoie à l'une des instances démarrées pour satisfaire à la charge. L'instance choisie s'occupe du dialogue avec l'utilisateur et prend toutes ses données (items à vendre, description, prix, contenu du panier d'achat...) et envoie toutes ses mises à jour (nouvelle commande par un usager...) au serveur de données.

Le professeur: Michel Dagenais

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 milliseconde) 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 $1s / 417us / \text{requête} = 2398.08$ requêtes / seconde. Le goulot d'étranglement ici est le CPU plutôt que le disque. Avec assez de fils d'exécution, le système pourrait servir $1s / 167us / \text{requête} = 5988.02$ 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 $1 \text{ fil} * 417us / 167us = 2.497$ 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 transaction? 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

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 1 mégabit/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 $(a+b)/2 \pm (a-b)/2$ soit $-20.015s \pm .025s$ 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 $-19.99 \pm .02$ 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 $-20s \pm .01s$, 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 $-20s \pm .01s$.

- 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 surcoût 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

ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8480: Systèmes répartis et infonuagique (Automne 2018)

3 crédits (3-1.5-4.5)

CORRIGÉ DE L'EXAMEN FINAL

DATE: Vendredi le 14 décembre 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 5 questions pour 20 points

Question 1 (4 points)

- a) Un ordinateur contient un processeur à 8 coeurs, 4 disques de type SSD et 4 disques conventionnels. Il sert des requêtes DNS. Pour toutes les requêtes reçues, une recherche est effectuée en mémoire vive, ce qui prend 1ms de CPU. Ceci permet de servir 60% des requêtes. Pour les autres requêtes, dont la réponse ne se trouve pas en mémoire vive, une recherche est effectuée sur les disques SSD, ce qui prend 2ms de disque SSD. Ceci permet de servir la moitié des requêtes qui n'ont pas eu réponse en mémoire vive. Les requêtes restantes, dont la réponse n'a pas été trouvée en mémoire vive ni sur les disques SSD, seront servies à partir des disques conventionnels en 20ms de disque conventionnel. i) Quel est le nombre maximal de requêtes par seconde qui pourraient être servies si le logiciel traite séquentiellement les requêtes avec un seul thread? ii) Quel est le nombre maximal de requêtes servies si le logiciel utilise un grand nombre de threads et que les requêtes sont réparties uniformément sur les coeurs et les disques? Quel est le nombre minimal de thread requis pour maximiser le nombre de requêtes servies par seconde? **(2 points)**

Une requête demande en moyenne 1ms CPU, $0.4 \times 2\text{ms} = 0.8\text{ms}$ de disque SSD et $0.5 \times 0.4 \times 20\text{ms} = 4\text{ms}$ de disque conventionnel, pour un total de i) 5.8ms ou $1000\text{ms/s} / 5.8\text{ms/r} = 172\text{r/s}$. Avec de nombreux threads, les 8 coeurs pourraient traiter $8 \times 1000\text{ms/s} / 1\text{ms/r} = 8000\text{r/s}$, les disques SSD $4 \times 1000\text{ms/s} / 0.8\text{ms/r} = 5000\text{r/s}$, et les disques conventionnels $4 \times 1000\text{ms/s} / 4\text{ms/r} = 1000\text{r/s}$. Ce sont donc les disques conventionnels qui sont le facteur limitant et le nombre maximal de requêtes qui peuvent être servies est de ii) 1000r/s . Puisque chaque requête dure en moyenne 5.8ms sur un thread, cela donne $5.8\text{ms-t/r} \times 1000\text{r/s} = 5800\text{ms-t/s}$ ou 5.8s de thread par seconde. Il faut donc au moins 5.8, soit iii) 6 threads, pour soutenir ce débit maximal. Ceci est cohérent avec la règle conservatrice qui suggère 1 thread par ressource (coeur ou disque) et assure qu'avec $8 + 4 + 4 = 16$ nous aurions assez de threads.

- b) Un ordinateur doit traduire l'adresse symbolique www.polenord.com en adresse IP numérique. Il a accès à 2 serveurs DNS redondants dans son réseau local, et aux 13 serveurs DNS racine redondants. Il doit commencer par interroger les serveurs DNS du réseau local pour trouver l'adresse des serveurs de nom pour le domaine [polenord.com](http://www.polenord.com). Ces 2 serveurs ont exactement le même contenu et ont 60% de chances de contenir l'adresse recherchée. Toutefois, ils sont peu fiables et chaque serveur a 45% de chances d'être fonctionnel. Si ceux-ci sont en panne, ou n'ont pas ces adresses dans leur cache, l'ordinateur doit interroger les 13 serveurs racine. Ces 13 serveurs ont chacun l'information recherchée. Toutefois, ils sont eux aussi très peu fiables en ce moment en raison d'une attaque en déni de service, et chaque serveur a une probabilité de 10% d'être fonctionnel. Ensuite, ayant l'adresse des serveurs de nom pour le domaine [polenord.com](http://www.polenord.com), l'ordinateur pourra obtenir l'adresse du site www.polenord.com. i) Quelle est la probabilité que l'ordinateur puisse obtenir l'adresse des serveurs de noms de [polenord.com](http://www.polenord.com) auprès des 2 serveurs DNS de son réseau local? ii) Quelle est la probabilité que les 13 serveurs DNS racine ne soient pas disponibles? iii) Quelle est la probabilité que l'ordinateur ne puisse obtenir l'information à propos des serveurs DNS pour le domaine [polenord.com](http://www.polenord.com)? **(2 points)**

Les 2 serveurs DNS locaux seront non disponibles avec une probabilité de $(1 - .45)^2 = 0.3025$. S'ils sont disponibles, $1 - 0.3025 = 0.6975$, ils auront la réponse dans 60% des cas, soit i) $0.6975 \times 0.6 = 0.4185$. Les 13 serveurs racines seront indisponibles si les 13 sont simultanément non fonctionnels ii) $(1 - 0.1)^{13} = 0.254186583$. L'ordinateur n'obtiendra pas sa réponse si elle

n'est pas disponible sur le réseau local $1 - 0.4185 = 0.5815$ et les serveurs racine ne sont pas disponibles 0.254186583 , ce qui donne $iii) 0.5815 \times 0.254186583 = 0.147809498$.

Question 2 (5 points)

- a) Un ordinateur A envoie un message à B à 13h00m10.200s pour obtenir le temps et reçoit une réponse à 13h00m10.600s, ces deux temps étant mesurés avec l'horloge de A. L'ordinateur B reçoit la requête de A à 13h00m05.300s et retourne sa réponse à A à 13h00m05.550s, ces deux temps étant mesurés avec l'horloge de B. Quel est le décalage à appliquer sur A? Quel est l'intervalle d'incertitude associé? **(2 points)**

$$a = 13h00m05.300s - 13h00m10.200s = -4.900s$$

$$b = 13h00m05.550s - 13h00m10.600s = -5.050s$$

$$Ajustement = (a+b)/2 = (-4.900s - 5.050s)/2 = -4.975s$$

$$Précision = (a-b)/2 = (-4.900s + 5.050s)/2 = 0.075s$$

Le décalage à appliquer à A est de -4.975s et l'incertitude est de +/- 0.075s.

- b) Dans le nord du Canada, un serveur central d'exclusion mutuelle maintient les verrous qui servent à protéger les K fichiers qui représentent la liste de cadeaux de chaque personne. Ce serveur de verrous doit servir jusqu'à N threads répartis dans différents noeuds qui doivent accéder à ces fichiers. Chaque thread acquiert un après l'autre un certain nombre de verrous, suivant une discipline de verrouillage strict à deux phases, puis les relâche tous après la fin de la transaction. Lorsqu'un thread demande un verrou, cette opération est bloquante, le thread ne continue qu'après avoir obtenu le verrou. On suppose que les demandes de verrou sont toujours ordonnées correctement et qu'il n'y a pas d'interblocage. i) Combien de messages sont requis pour l'acquisition d'un verrou par un thread auprès du serveur central d'exclusion? ii) Si une case mémoire est requise dans le serveur pour chaque verrou en utilisation (numéro du thread qui détient le verrou) et pour chaque demande de verrou en attente (numéro du thread bloqué en attente pour ce verrou dans une liste), quel est le nombre maximal de cases mémoire qui peut être requis dans le serveur dans le pire cas? **(2 points)**

i) L'acquisition d'un verrou demande un message au serveur central par le thread et un message de réponse du serveur central vers le thread. Des accusés de réception pourraient être utilisés afin de ne pas bloquer indéfiniment en cas de perte d'un des deux messages sur le réseau. Il serait aussi possible de ne pas utiliser systématiquement d'accusé de réception mais plutôt que le thread envoie un message de vérification que sa demande a bien été reçue et est en attente si le délai est trop long. ii) Chaque verrou ne peut être détenu que par un seul thread. Chaque thread ne peut être en attente que d'un seul verrou. Ainsi, le nombre de cases mémoire pour les verrous en utilisation est au maximum de K et le nombre de cases pour les verrous en attente est au maximum de N (en fait N-1 puisqu'autrement on est en interblocage si tous les threads sont bloqués). Le nombre maximal de cases mémoire occupées pour cela est donc au total de K+N.

- c) Un groupe d'ordinateurs utilise l'élection hiérarchique. Lorsqu'un des ordinateurs du groupe ne réussit pas à contacter le serveur, il déclenche le processus d'élection. Quelles sont les étapes qu'il va effectuer et quels messages va-t-il envoyer à cet effet? **(1 point)**

L'ordinateur qui ne réussit pas à contacter le serveur courant déclenche une élection. Il commence par envoyer un message d'élection au serveur le plus prioritaire. S'il ne reçoit pas de réponse après un certain temps (et possiblement plus d'un essai), il contacte le second serveur le plus prioritaire, et continue ainsi jusqu'à ce qu'il reçoive une réponse ou qu'il ne reste plus de serveur plus prioritaire que lui. Si aucun serveur plus prioritaire ne lui a répondu, il s'autoproclame le serveur et envoie un message à cet effet à tous.

Question 3 (4 points)

- a) Lesquelles des transactions T, U, V et W 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
T: Read(a)
T: Read(b)
T: Read(c)
U: Read(c)
U: Read(d)
V: Début
V: Read(e)
T: Write(b)
T: Write(f)
W: Début
W: Read(e)
T: Compléter
W: Read(f)
U: Write(c)
U: Write(d)
U: Compléter
V: Read(d)
W: Write(a)
W: Write(b)
W: Compléter
V: Read(g)
V: Write(e)
V: Compléter

En reculant, au moment de compléter T, il n'y a pas de problème. Pour compléter U, il faut vérifier ce qu'il a lu (c, d) versus ce qui a été écrit par les transactions précédentes concurrentes, T (b, f). Il n'y a pas d'intersection et U peut compléter. Pour W, ses lectures (e, f) sont vérifiées versus les écritures de T (b, f) et U (c, d). Il y a intersection (f) et la transaction W doit être abandonnée. Pour V, les lectures (d, e, g) doivent être vérifiées versus les écritures de T (b, f) et U (c, d), mais pas de W qui a été abandonné. Il y a intersection (d) et la transaction V doit être

abandonnée. En fait, T était complété avant que W ne lise f, et U était complété avant que V ne lise d, mais cette information plus détaillée n'est pas utilisée par la méthode de base en reculant. Elle serait utilisée par la méthode avec les estampilles de temps.

En avançant, il faut vérifier ce qui est écrit par T (b, f) versus ce que U (c, d), V (e) et W (e) ont lu à ce moment. Il n'y a pas d'intersection et T peut compléter. Au moment de compléter U, ses écritures (c, d) sont comparées aux lectures à ce moment de V (e) et W (e, f); les lectures à venir de V et W ne sont pas encore connues à ce moment et n'ont donc pas d'impact sur la cohérence. Il n'y a pas d'intersection et la transaction U est validée. Pour W, ses écritures (a, b) sont comparées aux lectures de V (d, e) à ce moment. Il n'y a pas d'intersection et W peut compléter. La dernière transaction, V, complète nécessairement et ainsi les quatre transactions sont validées.

- b) Trois transactions concurrentes, T, U et V veulent effectuer les opérations suivantes. Les opérations pour chacune des transactions sont effectuées en séquence, dans l'ordre. Cependant, l'ordre entre les opérations des trois transactions peut varier, puisque chaque transaction est effectuée par un thread parallèle concurrent. La cohérence des opérations est assurée par des verrous ordinaires (i.e. un verrou est pris sur une variable par la transaction au premier accès à cette variable, indifféremment pour une lecture ou une écriture). Est-ce qu'un interblocage pourrait se produire? Si oui, donnez une séquence d'opérations qui mènerait à un interblocage. **(2 points)**

	Transaction T	Transaction U	Transaction V
0	Début	Début	Début
1	Read(a)	Read(c)	Read(d)
2	Read(b)	Read(d)	Read(e)
3	Read(f)	Write(c)	Read(f)
4	Write(c)	Write(c)	Write(e)
5	Write(f)	Write(d)	Compléter
6	Compléter	Compléter	

Oui, un interblocage est possible. Si T effectue ses opérations 0 à 3, il acquiert les verrous pour a, b et f. Ensuite, U avec ses opérations 0 à 1 pourrait acquérir c. Finalement, V pourrait effectuer ses opérations 0 à 2 et acquérir d et e. A ce moment, T voudra acquérir c (détenu par U), U voudra acquérir d (détenu par V) et V voudra acquérir f, détenu par T et nous avons un interblocage.

Question 4 (4 points)

- a) Deux scénarios sont comparés pour offrir un service de base de données. Dans le premier scénario a), 2 serveurs redondants viennent chacun avec une unité de disque RAID (4 disques dont au moins 3 sur 4 doivent être fonctionnels). Dans le second scénario b), 3 serveurs redondants viennent chacun avec 3 disques (qui doivent les 3 être fonctionnels). La probabilité d'être fonctionnel est 0.85 pour un serveur (hormis les disques) et de 0.9 pour un disque. Quel scénario sera le plus fiable? **(2 points)**

L'unité RAID a une probabilité de fonctionner de $0.9^4 = 0.6561$ (4 disques fonctionnels) et $4!/((4-3)!3!) \times 0.9^3 \times (1-0.9)^{4-3} = 0.2916$ (exactement 3 disques fonctionnels) pour un total de $0.2916 + 0.6561 = 0.9477$. Pour le scénario a) un serveur est opérationnel si le serveur est fonctionnel de même que son unité RAID, ce qui donne $0.9477 \times 0.85 = 0.805545$. Le service sera disponible sauf si les 2 serveurs sont en panne, ce qui donne $1 - (1 - 0.805545)^2 = 0.962187253$. Pour le scénario b), le serveur est fonctionnel si tout est fonctionnel (3 disques et le serveur) $0.9^3 \times 0.85 = 0.61965$. Le service sera fonctionnel sauf si les 3 serveurs sont en panne, ce qui donne $1 - (1 - 0.61965)^3 = 0.94497624$. Ainsi, le scénario a) est un peu plus fiable.

- b) Dans le travail pratique 3, un gabarit Heat, `autoscaling.yaml`, vous était fourni en exemple. Quel est le principe d'un service qui se met à l'échelle automatiquement sur OpenStack avec Heat? La section suivante est extraite de ce fichier. Expliquez à quoi sert cette section. (2 points)

```
cpu_alarm_high:
  type: OS::Ceilometer::Alarm
  properties:
    description:
    meter_name: cpu_util
    statistic: avg
    period: 60
    evaluation_periods: 1
    threshold: 50
    alarm_actions:
      - {get_attr: [web_server_scaleup_policy, alarm_url]}
  matching_metadata: {'metadata.user_metadata.stack':
                      {get_param: "OS::stack_id"}}
  comparison_operator: gt
```

La mise à l'échelle se base sur un répartiteur de requêtes, et un bassin de machines virtuelles identiques pouvant servir les requêtes. Lorsque la charge est trop élevée, de nouvelles instances de machines virtuelles sont ajoutées au bassin. Lorsque la charge est trop faible, des machines virtuelles sont retirées. Ainsi, le nombre de machines virtuelles disponibles s'ajuste en fonction de la charge. Le gabarit Heat permet de spécifier les différentes ressources requises pour implémenter un tel service, ainsi que leurs paramètres. La section de gabarit Heat fournie sert à instancier une ressource de type `OS::Ceilometer::Alarm`. Cette ressource monitor la métrique `cpu_util` qui est le taux d'utilisation du CPU. Si le taux d'utilisation du CPU monte au-dessus de 50% pendant 60 secondes, l'alarme exécute une action qui active la ressource `web_server_scaleup_policy` pour ajouter des machines virtuelles.

Question 5 (3 points)

Vous devez planifier un nouveau centre de données et comparer différents scénarios. Vous avez déjà prévu examiner la performance des systèmes (et les revenus qu'on peut en tirer), le coût des ordina-

teurs, le coût du bâtiment ainsi que les coûts d'opération et de renouvellement des équipements, afin de déterminer le projet le plus rentable sur la durée de vie anticipée. Devez-vous comme ingénieur aussi vérifier les aspects du développement durable de ce projet? i) Quelles sont les lois applicables? ii) Quelles sont les différentes phases du cycle de vie? iii) Quels sont les quatre différents types d'impact sur l'environnement? **(3 points)**

L'ingénieur doit tenir compte des conséquences de l'exécution de ses travaux sur l'environnement dans une perspective de développement durable. i) Ceci est prescrit par la loi canadienne de 2008 sur le développement durable, la loi québécoise de 2006 sur le développement durable, et le code de déontologie de l'Ordre des Ingénieurs du Québec. La procédure d'évaluation environnementale du BAPE ne s'applique normalement pas à un centre de données. ii) Les phases du cycle de vie sont la fabrication (construction du site et fabrication des équipements), le transport (des matériaux pour la construction et des équipements), l'opération du site (entretien, alimentation électrique...) et finalement la fin de vie (la démolition ou conversion du bâtiment, la restauration du site, et le recyclage ou la mise aux rebuts des matériaux et équipements). iii) 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 de l'équipement informatique et de 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. Étant donné l'importance de la consommation électrique, 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.

Le professeur: Michel Dagenais

ÉCOLE 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, $0.3 \times 15\text{ms} = 4.5\text{ms}$ d'attente après le disque et $1/3 \times 0.3 \times 200\text{ms} = 20\text{ms}$ d'attente après le serveur de plus haut niveau, pour un total de 25ms. Le serveur pourra donc traiter séquentiellement $1000\text{ms/s} / 25\text{ms/r} = 40$ requêtes / s. Chaque coeur du CPU peut traiter $1000\text{ms/s} / 0.5\text{ms/r} = 2000\text{r/s}$, soit 8000r/s pour les 4 coeurs. Chaque disque peut traiter $1000\text{ms/s} / 4.5\text{ms/r} = 222.22\text{r/s}$, 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 $1777.77\text{r} \times .025\text{s} = 44.44$ 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 Cristian. 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) = $13h00m09.005 \pm .005$. Le décalage à appliquer à ce moment sur l'heure du client qui était de $13h00m04.010$ est donc de $13h00m09.005 - 13h00m04.010 = 4.995s \pm .005$.

- 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 répartiteur 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 équipements informatiques et électriques, en particulier le raffinage de l'or et du cuivre requis pour ces équipements. Étant donné l'importance de la consommation électrique, 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 nœuds 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